Characterization of ancient and modern genomes by SNP detection and phylogenomic and metagenomic analysis using PALEOMIX

Mikkel Schubert[1], Luca Ermini[1], Clio Der Sarkissian[1], Hákon Jónsson[1], Aurélien Ginolhac[1], Robert Schaefer[2], Michael D Martin[1], Ruth Fernández[1], Martin Kircher[3], Molly McCue[4], Eske Willerslev[1] & Ludovic Orlando[1]

[1]Centre for GeoGenetics, Natural History Museum of Denmark, University of Copenhagen, Copenhagen, Denmark. [2]Biomedical Informatics and Computational Biology Graduate Program, University of Minnesota Rochester, Rochester, Minnesota, USA. [3]Department of Genome Sciences, University of Washington, Seattle, Washington, USA. [4]College of Veterinary Medicine, University of Minnesota, St. Paul, Minnesota, USA. Correspondence should be addressed to L.O. (Lorlando@snm.ku.dk).

Published online 10 April 2014; doi:10.1038/nprot.2014.063

Next-generation sequencing technologies have revolutionized the field of paleogenomics, allowing the reconstruction of complete ancient genomes and their comparison with modern references. However, this requires the processing of vast amounts of data and involves a large number of steps that use a variety of computational tools. Here we present PALEOMIX (http://geogenetics.ku.dk/publications/paleomix), a flexible and user-friendly pipeline applicable to both modern and ancient genomes, which largely automates the *in silico* analyses behind whole-genome resequencing. Starting with next-generation sequencing reads, PALEOMIX carries out adapter removal, mapping against reference genomes, PCR duplicate removal, characterization of and compensation for postmortem damage, SNP calling and maximum-likelihood phylogenomic inference, and it profiles the metagenomic contents of the samples. As such, PALEOMIX allows for a series of potential applications in paleogenomics, comparative genomics and metagenomics. Applying the PALEOMIX pipeline to the three ancient and seven modern *Phytophthora infestans* genomes as described here takes 5 d using a 16-core server.

INTRODUCTION

The recent development of next-generation sequencing (NGS) technologies has revolutionized genomics, delivering complete genome sequences of model and non-model organisms in a cost- and time-efficient manner[1,2]. The impact of this technology is vast, with many applications in biosciences, including genome-wide SNP[3] and copy number variation (CNV)[4] discovery, nucleosome mapping[5] and genome-wide methylation profiling[6], genome re-sequencing[7] and *de novo* genome assembly[8]. Several sequencing platforms have been developed over the years, including the Roche 454, the Illumina and the SOLiD lines of sequencers. Such platforms are based on various principles, but they commonly require short DNA fragments as templates from which libraries are built and, in turn, sequenced in a massively parallel fashion. As sequencing technologies have advanced, developments have also been made toward so-called third-generation sequencing technologies (e.g., Helicos, Pacific Biosciences and Oxford Nanopore Technologies) that achieve single-molecule sequencing in the absence of pre-sequencing DNA library amplification.

The common preference of most NGS technologies for short templates is particularly attractive for ancient DNA (aDNA) research, as postmortem DNA degradation fragments chromosomes into molecules that are often shorter than 100 bp in size[9]. NGS was used on aDNA extracts soon after the first NGS platforms became available (Roche 454)[10], and it revealed 13 Mb of mammoth nuclear sequence information in one sequencing run[11]. It took no more than 2 years before a first draft of the mammoth genome could be characterized[12]. The massive throughput of Illumina platforms subsequently enabled the sequencing of the first ancient human genome at 20× coverage by using hair shafts preserved in Greenland for ~4,000 years[13]. Soon after, the first draft of the Neandertal genome was completed by using bone extracts of three specimens[14] and even a new group of archaic hominins, the Denisovans, was discovered on the basis of genome sequence data[15,16]. Clearly the paleogenomic era had been unleashed, and today a series of additional ancient genomes have been characterized, including those of an ~100-year-old Australian Aborigine[17], the Copper Age Iceman[18], an ~24,000-year-old ancient Siberian from Mal'ta[19], as well as high-quality genomes from Neandertals[20] and Denisovans[21]. Recently, the first draft genome of an early Middle Pleistocene horse was characterized by a combination of second- and third-generation platforms (Illumina and Helicos true single-molecule sequencing, respectively)[22], paving the way toward a 1 million–year-old genome[23]. NGS shotgun sequencing of ancient remains has also been essential for recovering genome-wide information to address long-standing questions regarding recent human migrations[24], and to unveil the evolutionary history of the polar bear[25,26]. In addition, target-enrichment approaches have revealed a plethora of complete ancient mitochondrial genome sequences[27–35], the sequence of the Neandertal exome[36], a complete chromosome of an ~50,000-year-old human from China[31] and the complete genome sequence of microbial pathogens responsible for massive historical outbreaks, such as the *Yersinia pestis* bacterium responsible for the Black Death[37].

The analysis of millions to billions of short-sequencing reads delivered by NGS platforms requires high-performance computational resources, large-scale data storage and efficient analytical tools. In recent years, many such bioinformatic tools have been developed and provided to the scientific community[38–41]. However, executing an end-to-end analysis, from raw reads to complete mitogenome and/or whole-genome sequences, carrying a list of high-quality annotated variants, still remains a challenging task. Moreover, in the case of NGS data sets generated from ancient specimens, analytical pipelines should take into account the molecular features of aDNA. In particular, postmortem DNA damage not only introduces characteristic nucleotide misincorporation signatures that can reduce the sensitivity of sequence alignment to reference genomes[42] but also offers a great opportunity to authenticate sequence data sets as ancient, rather than as the by-products of modern contamination[43]. In addition, for most NGS data sets from archaeological remains, the majority of DNA reads do not originate from the studied specimens, but from microbes that colonized samples during deposition. An important but often neglected aspect of the analysis of NGS aDNA data sets is the characterization of the microbial metagenomic composition of the ancient samples.

Here we present the flexible and open-source pipeline called PALEOMIX, combining all associated tools that were developed while characterizing the Middle Pleistocene horse draft genome[22]. In a nutshell, the PALEOMIX pipeline starts with NGS sequencing reads and proceeds through a series of analytical stages before delivering a final set of alignments against one or more reference genomes. In addition, the PALEOMIX pipeline supports the characterization of DNA damage patterns[44], the quantification of DNA damage parameters[45], the identification of the library metagenomic composition and can perform super-matrix maximum-likelihood phylogenetic inference.

Our pipeline is a flexible set of steps and tools designed to work efficiently and reproducibly with NGS data sets. A variety of user-defined options allows for compatibility with a full range of experimental situations. As a result, the PALEOMIX pipeline can equally well be applied to aDNA and modern NGS sequence data sets, and can process shotgun as well as target-enrichment sequence data. In the original publication[22], it was not only successfully applied to the characterization of two ancient horse genomes but also to that of five modern horse domestic breeds, the Przewalski's horse and the domestic donkey. In addition, it enabled the characterization of complete mitochondrial genomes from 29 ancient horses.

Here we present a protocol describing the use of the PALEOMIX pipeline and illustrate its versatility by applying it to a series of ancient and modern genomic data sets. These data sets were obtained from two studies[46,47] that report genome sequences of *Phytophthora infestans*, the oomycete pathogen responsible for the potato blight and the Irish famine of the mid-19th century. These genomes were characterized by using shotgun-sequencing data sets generated from historical herbarium potato leaf samples, in addition to a number of modern samples included for comparison with the historical samples.

For convenience, the protocol presented here is broken down into three major parts, corresponding to each set of tools used in the analyses (**Fig. 1**); part one includes pre-processing and reading, including adapter removal and quality filtering, followed by the generation of alignments against one or more reference genomes. Part two consists of variant detection and annotation, followed by genome-wide phylogenetic inference, and part three is the examination of the metagenomic contents of the sample by using sequences produced in part one of the protocol. The first part is similar to a previously published protocol[48], and it is typically carried out for each sample to be examined.

The alignment and phylogenetic parts of the pipeline are designed to be executed with little user intervention, automatically rerunning analytical steps in response to changes to input files and parallelizing tasks as much as possible. These parts are designed to resist failure during execution, by verifying completion of individual processes to avoid the inclusion of partial results, and by allowing the pipeline to easily be resumed from any step in the process. The metagenomic third part, in contrast, is implemented as a series of shell and R commands and scripts.

Although the pipeline software for parts one and two is primarily intended for stand-alone servers, and handles the parallelization of individual steps in the analyses, support for breaking down the individual steps of the pipeline is offered through the use of command-line options to facilitate the use of the pipeline on a cluster environment in which the total allowable run time is limited. The pipeline software does not, however, support distributing a single project over multiple hosts in an automated fashion. As the software is designed to perform an almost fully automatic genome analysis on a dedicated server, no effort has been made to make PALEOMIX compatible with Galaxy platforms[49].

As the pipeline software relies on existing tools to handle major computational tasks, the scalability of the pipeline is mainly limited by the performance of these tools; where possible, the pipeline itself will parallelize tasks, e.g., running different lanes or libraries in parallel for the binary alignment/map (BAM) pipeline, and also allow for the easy use of 'chunked' data sets to further parallelize the processing of individual lanes. For example, this pipeline has been used successfully both to map >40 samples against a mammalian (nuclear and mitochondrial) genome in a single run and to map multiple samples against several hundreds of different bacterial or viral genomes in a single run.

However, neither the execution of the Genome Analysis Toolkit (GATK)[40] nor the genotyping steps in the phylogenetic pipeline are parallelized, although different genomes or sets of regions may be run in parallel. This results in a suboptimal use of resources when few genomes are to be genotyped and when large parts of a genome are to be genotyped. These steps will be parallelized in the near future.

Preparation of samples

During NGS library preparation, adapters are appended to the ends of the inserts. Depending on the inserts' size distribution, and the number of bases sequenced, adapters may be either absent from final sequences sequenced in part or in whole. Removing adapter sequences from sequencing reads is therefore a requirement before subsequent attempts are made at aligning sequences to reference genomes and/or identifying biologically meaningful variation[50]. Given the highly fragmented nature of most aDNA samples, the latter category of sequences is expected to be more frequent than those for modern samples in which generally larger insert sizes are used.

PALEOMIX makes use of AdapterRemoval to trim adapter sequences from single-ended and/or paired-ended sequences, as this software not only allows for the removal of specified adapter sequences but also for the trimming of low-quality and/or ambiguous bases (N) and the collapse of overlapping paired-ended reads into a single (potentially longer) sequence[50]. The latter has been shown to decrease the error rate for the resulting collapsed reads[48]. After trimming of adapters and merging overlapping paired-ended reads, very short sequences may be discarded (<25 bp by default, unless changed by the user) in order to reduce the fraction of spurious alignments[42].

Some authors have chosen to analyze only the collapsed sequences for highly fragmented aDNA samples[21,35] in an effort to exclude modern contamination under the assumption that modern contamination would exhibit lower levels of fragmentation[43]. Another advantage of collapsed reads is that they reflect the true size distribution of DNA inserts[22], which can be examined by software such as mapDamage[45]. The automated protocol described here will collapse reads that overlap for at least 11 bases, selecting the most probable nucleotide in the case of mismatches in the overlapping region[14,51]. This behavior can be modified by the end user, and it allows for the optional exclusion of noncollapsed reads to accommodate different experimental setups.

After adapter trimming and sequence collapsing, samples may be examined for their metagenomic content as described below; alternatively, trimmed sequences may be mapped against reference sequences or genomes of interest. The standard BAM format[39] is used to store alignments against the reference genome in a space-efficient manner, which allows for random access. This format is supported by a number of cross-platform tools, including the suite of reference tools SAMtools[39], the Picard suite of tools and the GATK[40]. In addition, APIs exist for a number of languages, including C/C++, Python, Haskell and many others, supporting an easy integration into new and existing tools.

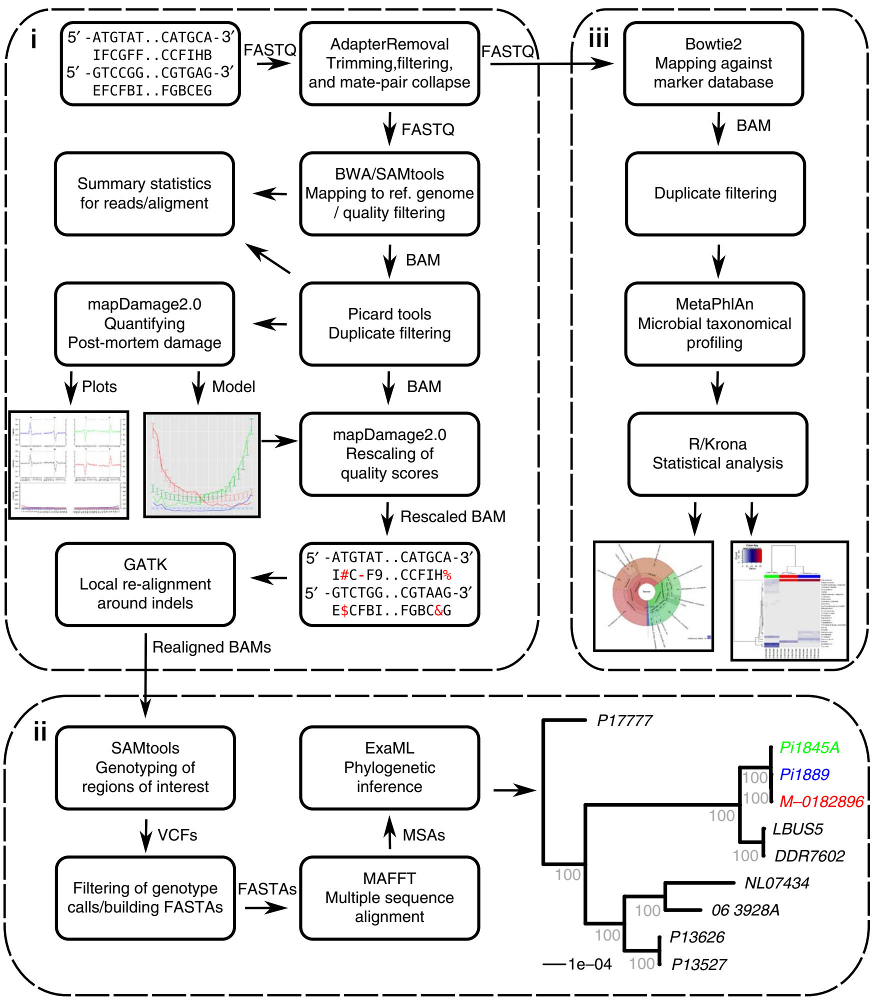

Figure 1 | Overview of major analytical parts of the protocol: (i) initial processing of raw reads and generation of alignments against a reference genome; (ii) phylogenetic analyses; and (iii) microbial taxonomical profiling.

The automated protocol described here uses the popular Burrows-Wheeler alignment (BWA) software to align sequences against reference genomes[38], and also supports Bowtie2 (ref. 52). One drawback of BWA is the presumption that few differences between the query sequence and the reference will be observed within the first 32 bp (i.e., it performs read seeding by default in order to speed up alignment to the reference sequence). However, aDNA sequences often show an excess of nucleotide misincorporations at read termini, owing to the presence of DNA overhangs in which post-mortem cytosine deamination rates are inflated[53]. For highly damaged aDNA templates, mismatch expectations of a seeding approach can be too conservative and result in the loss of a (minor) fraction of endogenous DNA sequences[42]. The automated protocol described here allows for the seeding heuristic to be turned on and off on a per project/sample/library basis as disabling this feature results in a many-fold increase in run time[42].

PCR amplification of DNA libraries can also result in a biased representation of DNA inserts, with size differences and GC content percentages as well-characterized sources of bias[30]. Sequences with identical coordinates on reference genomes probably represent sequences of the same insert rather than independent inserts and are therefore removed before downstream analyses. Although filtering these sequences is not unproblematic and many programs have been developed to filter out PCR duplicates by using sequence coordinates (e.g., Picard MarkDuplicates and SAMtools rmdup), these typically consider collapsed reads as single-ended sequences in which template molecules are identified solely by the 5′ position of the alignment against a reference sequence. Yet as collapsed sequences represent the original template molecule, both alignment termini can be considered specific features of an original template molecule, following the same logic as that used for filtering paired-ended duplicate reads. The protocol described here therefore makes use of a modified version of a program originally developed to filter PCR duplicates among collapsed reads for the Neandertal draft genome[14]. Collapsed reads that have been truncated from either end of the collapsed sequence owing to low-quality bases, single-ended reads and noncollapsed paired-ended reads are filtered with the standard MarkDuplicates program. In all cases, the programs attempt to select the highest quality duplicate on the basis of individual base quality scores.

Evaluating the presence of postmortem DNA damage signatures from read alignments has now become part of a standard procedure to authenticate aDNA data sets[43,44]. After mapping and filtering duplicates, nucleotide misincorporation and DNA fragmentation patterns can be plotted, and DNA damage parameters can be estimated with mapDamage2.0. The posterior distributions of damage parameters may also be useful for sample comparisons[45]. Subsequently, these estimates may be used to recalculate base quality scores in aligned reads and to downscale bases that are likely to be by-products of postmortem DNA damage, thereby reducing noise in downstream analysis. In addition, mapDamage2.0 provides various statistics for the aligned sequences, including the base composition of each aligned position and the insert-size distribution as inferred from collapsed sequences. These features have all been incorporated into the PALEOMIX pipeline.

Owing to the alignment of sequences being carried out with fast mapping software, alignments overlapping indels may be suboptimal; to achieve higher quality alignment around indels, the Indel Realigner from the GATK is (optionally) used in PALEOMIX in a two-step process, first detecting the regions containing likely indels and then realigning reads mapping across those indels both to achieve a consensus indel suitable for downstream analysis and to minimize spurious mismatches resulting from misalignments[40].

Analysis of BAM files

After raw sequencing reads are processed into the final sequence alignments in BAM format, a set of genomic regions are selected, which represent one or more sets of regions of interest (a set of genes, the exome, RNAs, whole organelle genomes and so on). These are genotyped by using SAMtools in the pipeline described here; although GATK[40] offers another compelling alternative for calling SNPs, it relies on the existence of high-quality data sets of known variation for the organism of interest to achieve high specificity, which is typically not available for nonhuman, nonmodel organisms. Because of this, the initial release of the PALEOMIX pipeline only supports genotyping with SAMtools.

Consequently, a set of variant call format (VCF) files containing the regions of interest are produced (including SNPs, indels and bases matching the reference sequence), and filtered according to a set of user-specified criteria, including but not limited to SNP quality, read depth and the number of alternative alleles observed. These VCF files may subsequently be analyzed by using any number of standard tools. For example, SNPs in genomes that are part of the Ensembl database may be analyzed by using the variant effect predictor (VEP) script to determine the genes affected and their effect on the coding sequence, the location of SNPs relative to annotations, sorting intolerant from tolerant (SIFT) and PolyPhen scores for the resulting protein sequences and more[54].

For each position that passes the VCF filters, the most likely diploid genotype is determined and FASTA sequences are produced for each region of interest (typically, genes). To accommodate different types of regions (e.g., genomic versus organelle, coding versus noncoding), filtering criteria may be specified globally or per set of regions. The resulting sets of sequences are collected for each sample and (optionally) aligned by using the multiple sequence alignment based on fast Fourier transform (MAFFT) suite of multiple sequence alignment algorithms[55]. These sequences and alignments are then merged into a super-matrix[56] and partitioned according to some desired partition scheme (e.g., one partition per codon position); a maximum-likelihood phylogeny is then inferred by using ExaML, an Message-Passing Interface (MPI)-based member of the randomized 'axelerated' maximum likelihood (RaxML) family of phylogenetic software[57] optimized for large-scale, partitioned multigene and whole-genome data sets. In addition, bootstrap alignments may be generated from the partitioned super matrix, in order to estimate support for individual branches in the resulting phylogeny.

Microbial taxonomical profiling

In addition to mapping sequences to a specific set of reference genomes, the trimmed reads may also be used to create a taxonomic profile of the bacterial metagenomic contents of the sample, both in terms of identifying which microbial taxonomical groups are present and in terms of quantifying the relative abundance of DNA of microbial origin, potentially providing insights into taphonomic processes and contamination occurring during the handling and analysis of the samples. This approach maximizes the amount of information that may be retrieved from aDNA NGS shotgun data sets, in which the majority of DNA sequencing reads typically originates from exogenous microbial sources, and is also applicable for modern environmental samples and microbiomes.

A versatile suite of statistical tools allows for the comparison of microbial profiles from DNA extracts and libraries produced using different starting materials (e.g., under different environmental conditions) and/or methodologies. Here we showcase how the metagenomic module can be used to compare intra- and inter-specimen variability of the microbial DNA population of the samples. The PALEOMIX pipeline is thus suited for various applications in molecular ecology, such as environmental microbial biodiversity monitoring[58], dietary analysis[59] and profiling of microbes associated with a living host[60].

The corresponding taxonomic profile is then assigned through a series of stages by using published programs and scripts (**Fig. 1**). The four main stages are: (i) mapping trimmed reads to a database of phylogenetically informative metagenomic markers; (ii) removing PCR duplicates; (iii) calculating the taxonomical composition of the microbial communities; and (iv) statistically analyzing the microbial profiles (e.g., principal component analysis (PCA) and hierarchical clustering).

The microbial communities are profiled by metagenomic phylogenetic analysis (MetaPhlAn)[61]; MetaPhlAn relies on a homology-based classification of metagenomic reads by comparison with preidentified and curated clade-specific reference marker sequences[61], rather than on local homology searches against a full catalog of reference genomes (i.e., by using nucleotide BLAST (BLASTN; wherein BLAST stands for basic alignment search tool)[62] and metagenome analyzer (MEGAN)[63]). The specificity of the reference markers used here limits possible bias related to evolutionary conserved regions and horizontal gene transfer, making it particularly suitable for microbial identification. MetaPhlAn has been successfully applied to the profiling of clinical microbiomes[64] and to that of ancient fossil remains preserved on frozen soils[65].

To identify the presence of taxonomic groups, the collapsed reads are mapped against the MetaPhlAn database of genomic markers. Although the mapping process may be carried out in either Bowtie2 (ref. 52) or BLASTN[62], the former substantially reduces the computational resources required and is clearly preferred for this protocol.

As the nonhomogeneous and artifactual over-representation of sequences arising during presequencing PCR amplification of libraries can skew the calculation of microbial taxon relative abundances, PCR duplicates should be removed from the data sets. PCR duplicates are identified and removed by using the same method as that used for the second part of the protocol, taking both alignment coordinates into consideration. The filtered hits are then converted into a format that is usable by the MetaPhlAn program (see Steps 33–37) and analyzed to produce tables of relative abundances of taxonomical groups.

MetaPhlAn microbial profiles can be visualized through heat maps or stacked bar plots; alternatively, they can be viewed by using the hierarchical data browser Krona[66]. Multiple microbial profiles can then be characterized and compared by using standard statistical analyses in R[67], including PCA, principal coordinate analysis (PCoA) and hierarchical clustering. The R package pvclust[68] may be used to assess uncertainty by performing bootstrap resampling analyses and calculating *P* values for clusters.

MATERIALS

EQUIPMENT

▲ **CRITICAL** Software version numbers given here specify the minimum version required, not the specific version required. Java archive (JAR) files must be located at ~/install/jar_root/. MetaPhlAn is assumed to be installed in the folder '~/install/metaphlan'. The location of executables must be specified in the user's PATH variable. For example, to add the folder ~/install/bin to the current user's PATH, use the following shell command:

```
$ export PATH=~/install/bin/:$PATH
```

Operating system and hardware

- A computer running a 64-bit Linux with at least 16 cores, at least 32 GB of RAM, at least 800 GB of free disk space (for the example data set), and a high-speed internet connection

General software requirements

- Bash (https://www.gnu.org/software/bash/bash.html)
- Git (http://git-scm.com/)
- Python v2.7.x (http://python.org/)
- Pysam v0.7.5 (https://code.google.com/p/pysam/)
- Perl (http://www.perl.org/)
- A Java runtime environment (http://www.java.com/)
- Curl (http://curl.haxx.se/)

Software requirements for BAM pipeline

- AdapterRemoval v1.5 (ref. 50) (https://code.google.com/p/adapterremoval/)
- BEDTools v2.16.0 (ref. 69) (https://code.google.com/p/bedtools/)
- BWA v0.5.10 (ref. 38) (http://bio-bwa.sourceforge.net/). Newer versions (0.6.x and 0.7.x) are not recommended at the time of writing
- GATK[40] (http://www.broadinstitute.org/gatk/)
- mapDamage v2.0.2 (ref. 45) (http://ginolhac.github.io/mapDamage/)
- Picard tools v1.82 (http://picard.sourceforge.net/)
- SAMtools v1.18 (ref. 39) (http://samtools.sourceforge.net/)
- Preseq 0.0.5 (ref. 70) (optional; http://smithlabresearch.org/software/preseq/)

Software requirements for phylogenetic pipeline

- BEDTools v2.16.0 (ref. 69) (https://code.google.com/p/bedtools/)
- ExaML v1.0.5, parser v1.0.2 (https://github.com/stamatak/ExaML). The ExaML binary must be named 'examl', and the parser binary must be named 'examlParser'. Compiling and running ExaML requires MPI (e.g., Open-MPI; http://www.open-mpi.org/)
- MAFFT v7 (ref. 55) (http://mafft.cbrc.jp/alignment/software/). The algorithm-specific MAFFT commands (e.g., 'mafft-ginsi', 'mafft-fftnsi') must be installed. These are automatically created by the 'make install' command
- RAxML v7.3.2 (ref. 57) (https://github.com/stamatak/standard-RAxML). The pipeline expects a single-threaded binary named 'raxmlHPC'; therefore do not use the 'PTHREADS' makefiles
- SAMtools v1.18 (ref. 39) (http://samtools.sourceforge.net/)
- Tabix v0.2.5 (ref. 71) (http://samtools.sourceforge.net/). Both the 'tabix' and the 'bgzip' executables must be installed

Software requirements for metagenomic pipeline

- Bowtie2 (ref. 52) v2.1.0 (http://bowtie-bio.sourceforge.net/bowtie2/index.shtml)
- R v3.0 statistical computing environment (http://www.r-project.org/)

R packages

- permute (http://cran.r-project.org/web/packages/permute/)
- vegan (http://cran.r-project.org/web/packages/vegan/)
- MASS[72] (http://cran.r-project.org/web/packages/MASS/)
- gplots (http://cran.r-project.org/web/packages/gplots/index.html)
- ape[73] (http://cran.r-project.org/web/packages/ape/)
- pvclust[68] (http://cran.r-project.org/web/packages/pvclust/index.html)
- ggplot2 (ref. 74) (http://cran.r-project.org/web/packages/ggplot2/index.html)
- MetaPHlAn[61] (http://huttenhower.sph.harvard.edu/metaphlan)
- Krona[66] 2.4 (http://sourceforge.net/p/krona/home/krona/)

Input file formats

- One or more FASTQ files containing sequencing reads produced by NGS platforms
- One or more FASTA files containing the reference sequences against which the FASTQ files are to be mapped. When using the human genome, it is important to ensure that the ordering of chromosomes in the FASTA file corresponds to the historical ordering; please refer to the GATK FAQ for more information (http://www.broadinstitute.org/gatk/guide/topic?name=faqs)

Example data

- Example data are taken from recent studies[46,47]. The data consist of NGS reads produced from shotgun sequencing of potato (*Solanum tuberosum*) leaves infected with *P. infestans*, also known as late potato blight. These data include three historical samples (M-0182896, Pi1845A and Pi1889) and seven modern samples (06_3928A, DDR7602, LBUS5, NL07434, P13527, P13626 and P17777), listed in **Supplementary Table 1**
- The *P. infestans* nuclear genome[75] (European Nucleotide Archive accession no. GCA_000142945.1) and the *P. infestans* mitochondrial genome[76] (accession AY894835.1) in FASTA format

EQUIPMENT SETUP

Install required R packages Type the following:

```
$ R
  > install.packages(c("permute", "vegan",
  "MASS", "gplots", "ape", "pvclust", "ggplot2"))
```

Install required python modules Type the following:

```
$ mkdir -p ~/install
$ cd ~/install
$ curl -O https://pysam.googlecode.com/files/
pysam-0.7.5.tar.gz
$ tar xvzf pysam-0.7.5.tar.gz
$ cd pysam-0.7.5
$ python setup.py install -user
```

Install the PALEOMIX pipeline Detailed instructions may be found at http://geogenetics.ku.dk/publications/paleomix. Enter the following:

```
  $ mkdir -p ~/install
  $ cd ~/install
  $ git clone https://github.com/MikkelSchubert/
pypeline.git
  $ cd ~/install/pypeline/
  $ python setup.py install -user
  $ echo "export PATH=~/.local/bin:\$PATH" >>
~/.bashrc
  $ source ~/.bashrc
```

Download the example data Use the provided script to download the FASTA reference sequences:

```
  $ cd ~/install/pypeline/examples/nature_protocols/
alignment/000_prefixes
  $./setup.sh
```

Use the provided script to download the FASTQ sequencing reads:

```
  $ cd ~/install/pypeline/examples/nature_protocols/
alignment/000_rawreads
  $ ./setup.sh
```

PROCEDURE

Set up and run the BAM pipeline ● TIMING 72 h

1| Use the provided small example project to test whether the pipeline has been correctly installed. At the prompt terminal, type (note that $ is the command prompt and # indicates a comment):

```
$ cd ~/install/pypeline/examples/bam_pipeline
# Carry out dry-run of the pipeline; displays dependency graph or errors
$ bam_pipeline dryrun 000_makefile.yaml
# Run the pipeline; output is placed in the current directory
$ bam_pipeline run 000_makefile.yaml
```

? TROUBLESHOOTING

2| (Optional) Record your preferred settings: e.g., default locations of temporary files, and the maximum number of active processes. Use the '--write-config-file' switch, which writes the current (globally settable) values to ~/.pypeline/bam_pipeline.ini:

```
$ bam_pipeline run --write-config-file
$ cat ~/.pypeline/bam_pipeline.ini
[Defaults]
max_threads = 4
...
```

3| Create initial makefile in YAML format (**Box 1**):

```
$ cd ~/install/pypeline/examples/nature_protocols/alignment
$ bam_pipeline mkfile > 000_makefile_template.yaml
```

Open the '000_makefile_template.yaml' and the '000_makefile.yaml' files (located in the same folder) in a suitable text editor; the second file contains the final makefile, and may be used as a reference for the following steps.

4| Review the global options in the 'Options' section as described in **Box 1**. Set the minimum mapping quality to 25 by using Options:Aligners:BWA:MinQuality to eliminate low-quality hits from the analyses:

```
Options:
  ...
  Aligners:
   ...
   BWA:
     MinQuality: 25
```

Box 1 | BAM pipeline makefiles

The BAM pipeline makes use of a 'makefile' in YAML format (http://www.yaml.org), a human-readable data serialization format, to specify input files and settings for the processing of these files. A template makefile may be created by using the 'mkfile' command (Step 3), which prints the template to stdout:

```
$ bam_pipeline mkfile > makefile.yaml
```

The use of a text editor meant for editing source code (emacs, vim, xcode, gedit, kate and so on) is recommended when working with the makefiles; a rich text editor such as Microsoft Word is not recommended, as these may easily generate invalid YAML files. In addition, the editor must use spaces for indentation, not tabs.

The makefile is documented in the template file itself, and reviewing this documentation carefully is highly recommended when starting a new project. The following options are of particular importance, and failing to set them to appropriate values will lead to unexpected results:

Options:QualityOffset: The ASCII value used to encode the Phred scores in the input FASTQ reads[77]. It may be identified by manual inspection of the FASTQ reads, or by using tools such as FastQC. (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/).

Options:AdapterRemoval:--pcr1 and **--pcr2**: Adapter sequences to be removed by AdapterRemoval. The correct sequences must be used to prevent contaminating the final sequences with the adapters[50]. Please refer to the AdapterRemoval manual for correct usage.

For ancient samples, the following options are of special interest:

Options:Aligners:BWA:UseSeed: By default, BWA uses the first 32 bp of reads as a high-fidelity seed region, which is not recommended for aDNA data sets[42]. Use this option to disable this behavior.

Options:RescaleQualities: To mitigate the effect of postmortem deamination of cytosines, mapDamage2.0 may be used to rescale per-base quality scores in mapped reads[45]. Use this option to enable rescaling of quality scores.

Options:ExcludeReads: For highly fragmented DNA material, it may be desirable to exclude paired-ended reads that do not overlap, and are therefore not collapsed by AdapterRemoval. Use this option to accomplish this.

5| List reference sequences for alignment specifying the downloaded reference sequences under 'Prefixes', to indicate which sequences the sample data should be mapped against. To ease interoperability with the second part of the protocol, use the names of the FASTA files without the extension as the name of the prefixes:

```
Prefixes:
  # Phytophthora infestans nuclear genome:
  # Name of the prefix; is used as part of the output filenames
  Pi_nucl:
    # Path to .fasta file containing a set of reference sequences.
    Path: 000_prefixes/Pi_nucl.fasta
    # Label for prefix: One of nuclear, mitochondrial, chloroplast,
    # plasmid, bacterial, or viral. Is used in the .summary files.
    Label: nuclear

  # Phytophthora infestans mitochondrial genome:
  # Name of the prefix; is used as part of the output filenames
  Pi_mito:
    # Path to .fasta file containg a set of reference sequences.
    Path: 000_prefixes/Pi_mito.fasta
    # Label for prefix: One of nuclear, mitochondrial, chloroplast,
    # plasmid, bacterial, or viral. Is used in the .summary files.
    Label: mitochondrial
```

6| Record each mapping by specifying a ‘target’ for each sample (or related set of samples) at the root level (bottom) of the makefile; each target is given a name (the top level key) and consists of one or more samples, for each of which there are one or more libraries, and for each of which one or more ‘lanes’ have been sequenced:

```
# Target name
M-0182896:
  # Sample name
  M-0182896:
    # Library name
    M-0182896_NO_UDG:
      # "Lane" name and file path
      ERR267888: "000_rawreads/M-0182896/ERR267888_{Pair}_*.fastq.gz"

    M-0182896_UDG:
      ERR267889: "000_rawreads/M-0182896/ERR267889_{Pair}_*.fastq.gz"
```

Repeat for each of the remaining libraries for M-0182896 and for the remaining samples included in the example data. Samples and libraries are listed in **Supplementary Tables 1** and **2**. Use the sample name for both the sample and target keys, as shown above.
▲ CRITICAL STEP The target name is used to generate output filenames and is typically the same as the sample name. The sample, library and lane names are used to tag the reads and are also used to inform certain steps such as indel realignment, which is carried out per sample, and PCR duplicate removal, which is carried out per library. Each name should be at least two characters long. The path may contain wildcards to include multiple files (as shown above); these files (or pairs of files) are processed independently by default but are tagged with the same information. The string ‘{Pair}’ is used to signify that these are paired-ended data, and it is replaced with ‘1’ and ‘2’ by the pipeline before searching for files matching the path. For single-ended data, this string is simply omitted.

7| (Optional) Override the default adapter sequence for lanes sequenced using nonstandard adapters (changes highlighted in bold text):

```
# Target name
M-0182896:
  # Options that apply to all data in this target
  Options:
    AdapterRemoval:
      # Adapters that differ from AdapterRemoval defaults
      --pcr1: "AGATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNNATCTCGTATGCCGTCTTCTGCTTG"
      --pcr2: "AATGATACGGCGACCACCGAGATCTACACNNNNNNNACACTCTTTCCCTACACGACGCTCTTCCGATCT"
```

Repeat for each sample that uses nonstandard adapter sequences, by using the adapters listed in **Supplementary Table 3**.
▲ CRITICAL STEP Samples produced by using adapter sequences that differ from the AdapterRemoval default sequences must have these sequences specified either at the global level (Step 4) or for the individual sample (as shown here). The correct sequences must be used to prevent contaminating the final sequences with the adapter sequences[50].

8| (Optional) Set options suitable for ancient samples (changes highlighted in bold text):

```
# Target name
M-0182896:
  # Options that apply to all data in this target
  Options:
    AdapterRemoval:
      # Adapters that differ from AdapterRemoval defaults
      --pcr1: "AGATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNNATCTCGTATGCCGTCTTCTGCTTG"
      --pcr2: "AATGATACGGCGACCACCGAGATCTACACNNNNNNNACACTCTTTCCCTACACGACGCTCTTCCGATCT"

  # Sample name
  M-0182896:
    # Library name
    M-0182896_NO_UDG:
      # Options that apply only to library "M-0182896_NO_UDG"
      # The remaining libraries have been treated with UDG, which removes the
      # signature of postmortem DNA damage
      Options:
        # Rescale base qualities to account for postmortem damage
        RescaleQualities: yes

        Aligners:
          BWA:
            # Disable seed for ancient DNA
            UseSeed: no

       ERR267888: "000_rawreads/M-0182896/ERR267888_{Pair}.fastq.gz"

    M-0182896_UDG:
       ERR267889: "000_rawreads/M-0182896/ERR267889_{Pair}.fastq.gz"
```

Repeat for sample Pi1845A and Pi1889, setting both options at the target level, enabling the use of rescaling, and disable the use of a seed by BWA.

9| Run the BAM pipeline, using at most 16 cores in total, and at most four cores per BWA instance, by using the following command line:

```
$ bam_pipeline run --max-threads 16 --bwa-max-threads 4 000_makefile_template.yaml
```

▲ **CRITICAL STEP** It is highly recommended to examine the Fragmisincorporation_plot.pdf created for each library in the *.mapDamage folders (see Anticipated Results) to detect issues with adapter trimming, alignment and so on. For ancient samples that have been rescaled by using mapDamage2.0, examine the trace plot 'Stats_out_MCMC_trace.pdf' to detect any undesirable trends and the 'Stats_out_post_pred.pdf' to ensure that the observed position-specific substitution frequencies are largely contained in the posterior predictive intervals.

? TROUBLESHOOTING

10| (Optional) Perform a quick estimate of the molecular complexity of the 'Pi1889' library 'CTTGTA' and potential gains associated with further sequencing efforts using preseq, and then save the results in the file 'Pi1889.Pi_mito.duphist/Pi1889_id_CTTGTA_predicted_yields.txt':

```
$ cd ~/install/pypeline/examples/nature_protocols/alignment
$ lc_extrap -Q -H Pi1889.Pi_mito.duphist/Pi1889_id_CTTGTA.txt -o
Pi1889.Pi_mito.duphist/Pi1889_id_CTTGTA_predicted_yields.txt
```

Repeat for other samples and/or libraries as desired.
▲ **CRITICAL STEP** To generate the histogram files used as input by preseq, uncomment the 'DuplicateHist' line in the 'Features' list in the makefile before running the makefile as shown in Step 9; by default, these files are not generated.
? TROUBLESHOOTING

Set up and run the phylogenetic pipeline ● TIMING 38 h
11| Use the provided small example project to test whether the pipeline has been correctly installed. At the prompt terminal, type:

```
$ cd ~/install/pypeline/examples/phylo_pipeline/alignment
# Generate synthetic data
$ ./setup.sh
# Align synthetic data against reference sequence
$ bam_pipeline run 000_makefile.yaml
$ cd ../phylogeny
# Carry out dry-run of the pipeline; displays dependency graph or errors
$ phylo_pipeline genotype+msa+phylogeny 000_makefile.yaml --dry-run
# Run the pipeline; output is placed in the current directory
$ phylo_pipeline genotype+msa+phylogeny 000_makefile.yaml
```

? TROUBLESHOOTING

12| (Optional) Record your preferred settings: e.g., default locations of temporary files, data files and the maximum number of active processes. Use the '--write-config-file' switch to write the current command-line options to '~/.pypeline/phylo_pipeline.ini':

```
$ phylo_pipeline --write-config-file
$ cat ~/.pypeline/phylo_pipeline.ini
[Defaults]
max_threads = 4
...
```

13| Create the basic directory structure for the project:

```
$ mkdir -p ~/install/pypeline/examples/nature_protocols/phylogeny/data
```

Please see **Box 2** for a description of the file structure used by the phylogenetic pipeline.

14| Reuse the prefixes from the BAM pipeline for the phylogenetic pipeline:

```
$ cd ~/install/pypeline/examples/nature_protocols/phylogeny/data
$ ln -sf ../../alignment/000_prefixes prefixes
```

15| Use the alignments created by the BAM pipeline as input for the phylogenetic pipeline:

```
$ ln -sf ../../alignment samples
```

Box 2 | Phylogenetic pipeline makefiles

As in the BAM pipeline, the phylogenetic pipeline makes use of a 'makefile' in YAML format (**Box 1**). This makefile is used to specify the current set of samples, the genomic regions of interest and settings for programs used by the pipeline. The makefile is documented in the template file itself, and reviewing this documentation carefully is highly recommended when starting a new project.

Of special note is the 'Gender' key for samples; this key may be set to any string value (e.g., male, female) and is matched by a list given for each prefix, in which chromosomes or contigs that are haploid for that gender may be specified and subsequently used to filter heterozygous SNPs at those chromosomes as sequencing errors. The following demonstrates this for a XY sex-determination mammal (all extraneous options have been removed):

```
Project:
  Samples:
    Sample_A:
      Gender: male
    Sample_B:
      Gender: female

  RegionsOfInterest:
    Regions_A:
      HomozygousContigs:
        # Two haploid chromosomes for male genomes
        male: [chrX, chrY]
        # No haploid chromosomes for female genomes
        female: []
```

Note that lists must be specified for each gender used in the samples, even if these do not have any haploid chromosomes. Such genders are specified by using an empty list as shown above: [].

To ease interoperability with the BAM pipeline, and to reduce the need to manually specify paths, the phylogenetic pipeline expects files to be located at paths derived from the names of samples, prefixes and areas of interest and so on:

For a sample with name S and a prefix with name P, the pipeline expects .bam files to be located at './data/samples/S.P.bam'. If the Realigned option is enabled for the prefix, the path is expected to be './data/samples/S.P.realigned.bam', corresponding to the files produced by the BAM pipeline.

For a prefix with name P, the pipeline expects FASTA files to be located at './data/prefixes/P.fasta'.

For set of regions of interest with name R for prefix P, the pipeline expects a .bed file located at './data/regions/P.R.bed'. Similarly, for a selection of genes from this file named S, the pipeline expects text-file containing a name per line located at './data/regions/P.R.S.names'.

The default locations can be changed by using the command-line options and/or the configuration file (Step 12 of the main protocol).

16| Create .bed files with genomic coordinates for the regions of interest by using release 20 of the Ensembl protists database:

```
$ mkdir regions
$ cd regions
$ curl -O ftp://ftp.ensemblgenomes.org/pub/release-
20/protists/gtf/phytophthora_infestans/Phytophthora_infestans.ASM14294v1.20.gtf.gz
$ gunzip Phytophthora_infestans.ASM14294v1.20.gtf.gz
# The pipeline expects BED files to start with the prefix name:
$ conv_gtf_to_bed Phytophthora_infestans.ASM14294v1.20.gtf Pi_nucl.Ensembl.v20
```

17| Create an initial makefile with the 'mkfile' command (**Box 2**):

```
$ cd ~/install/pypeline/examples/nature_protocols/phylogeny
$ phylo_pipeline mkfile > 000_makefile_template.yaml
```

Open the '000_makefile_template.yaml', and the '000_makefile.yaml' file (located in the same folder) in a suitable text editor; the second file contains the final makefile and may be used as a reference for the following steps.

18| Set a name of the project in the makefile by using the 'Title' key:

```
Title: P_infestans
```

19| Specify each sample to be analyzed in the 'Samples' subsection by using the same sample name as that in the BAM pipeline:

```
  Samples:
    06_3928A:
      Gender: NA
    DDR7602:
      Gender: NA
    LBUS5:
      Gender: NA
    M-0182896:
      Gender: NA
    NL07434:
      Gender: NA
    P13527:
      Gender: NA
    P13626:
      Gender: NA
    P17777:
      Gender: NA
    Pi1845A:
      Gender: NA
    Pi1889:
      Gender: NA
```

As no filtering of SNPs will be performed here on the basis of the gender of the specimens, the 'Gender' is set to an arbitrary value ('NA').

20| Specify genomic regions to be analyzed for the *P. infestans* nuclear genome in the 'RegionsOfInterest' subsection by using the name of the .bed file created in Step 16 (**Box 2**):

```
  RegionsOfInterest:
    Ensembl.v20.protein_coding.CDS: # The name of the set of regions
      Prefix: Pi_nucl      # Determines FASTA / BED filename prefix
      Realigned: yes       # Tells the pipeline to expect .realigned.bam files
      IncludeIndels: yes # Include indels in the resulting sequences
      ProteinCoding: yes # Require that included indels are divisible by 3
      HomozygousContigs:
        NA: [] # Empty list for homozygous contigs determined by gender
```

21| Set a minimum read-depth and quality with 'vcf_filter' command-line options, and set a maximum read-depth for genotyping by using the 'MaxReadDepth' option and the maximum depth calculated by the BAM pipeline (c.f. the *.depth tables produced at Step 9) to account for the heterogeneous coverage for the different samples:

```
  VCF_Filter:
    MaxReadDepth:
      06_3928A:           110
      DDR7602:             73
      LBUS5:               85
      M-0182896:           93
      NL07434:            133
      P13527:              87
      P13626:             117
      P17777:             114
      Pi1845A:             17
      Pi1889:              41
    --min-read-depth:       6
    --min-quality:         20
```

22| (Optional) Set the additional command-line options used when genotyping the samples under the 'Genotyping' section; modify the 'Mpileup' section to change the behavior of the 'samtools mpileup' command during pileup construction; modify the 'BCFTools' section to modify the behavior of the 'bcftools view' command during genotyping; and modify the 'VCF_Filter' section to modify behavior of the 'vcf_filter' command following the genotyping of areas of interest.

23| Run the genotyping step of the phylogenetic pipeline with at most ~16 cores in total.

```
$ phylo_pipeline genotype --max-threads 16 000_makefile_template.yaml
```

? TROUBLESHOOTING

24| Tabulate the fraction of homozygous- and/or heterozygous-derived sites:

```
$ cd ~/install/pypeline/examples/nature_protocols/phylogeny/
$ ./summarize_heterozygosity.py data/regions/Pi_nucl.Ensembl.v20.protein_coding.CDS.bed
results/P_infestans/genotypes/Pi1889.Pi_nucl.Ensembl.v20.protein_coding.CDS.filtered.
vcf.bgz

17073318 sites kept after filtering:
  17004051 homozygous sites containing the reference allele (99.59%)
    41388 heterozygous sites containing the reference and a non-reference allele (0.24%)
    27827 homozygous sites containing a single non-reference allele (0.16%)
        52 heterozygous sites containing two different non-reference alleles (0.00%)
```

25| (Optional) Download and install VEP[54] release 73; the dependencies for VEP will be automatically downloaded by the installer provided by Ensembl:

```
$ cd ~/install/pypeline/examples/nature_protocols/phylogeny
$ curl -o vep73.tgz "http://cvs.sanger.ac.uk/cgi-bin/viewvc.cgi/ensembl-
tools/scripts/variant_effect_predictor.tar.gz?view=tar&root=ensembl&pathrev=branch-
ensembl-73"
$ tar xvzf vep73.tgz
$ cd variant_effect_predictor/
# Follow instructions, but do NOT install any cache/FASTA files
# (applies to this example only!):
$ perl INSTALL.pl
```

26| (Optional) Fetch genome annotation from Ensembl. For genomes accessible through the main Ensembl website, this step may be performed automatically via the INSTALL.pl script; otherwise, use the following commands:

```
$ mkdir -p ~/.vep/phytophthora_infestans/73
$ cd ~/.vep/phytophthora_infestans/73
$ curl -O ftp://ftp.ensemblgenomes.org/pub/protists/release-20/fasta/
phytophthora_infestans/dna/Phytophthora_infestans.ASM14294v1.20.dna.toplevel.fa.gz
$ gunzip Phytophthora_infestans.ASM14294v1.20.dna.toplevel.fa.gz
$ curl -O ftp://ftp.ensemblgenomes.org/pub/protists/release-20/vep/
phytophthora_infestans_vep_20.tar.gz
$ tar xvzf phytophthora_infestans_vep_20.tar.gz
$ mv phytophthora_infestans/20/* .
```

27| (Optional) Run the VEP script on SNPs that pass the filtering criteria used by the pipeline:

```
$ cd ~/install/pypeline/examples/nature_protocols/phylogeny/variant_effect_predictor
$ perl ./variant_effect_predictor.pl --offline --cache --format vcf --species
phytophthora_infestans -i <(zgrep -w -e "^#" -e "PASS" ../results/P_infestans/
genotypes/Pi1889.Pi_nucl.Ensembl.v20.protein_coding.CDS.filtered.vcf.bgz) -o OUTPUT
```

▲ **CRITICAL STEP** The ‘zgrep’ command is used to select only those SNP calls that passed the filters implemented by the ‘vcf_filter’ tool in addition to any headers. Note that owing to the use of padding to filter by indels (10 bp by default), this list will contain some SNPs that fall outside coding regions.
? TROUBLESHOOTING

28| (Optional) Select the MAFFT algorithm and set command-line options in the ‘MultipleSequenceAlignment’ section:

```
  MAFFT:
    # Select alignment algorithm; valid values are 'mafft', 'auto', 'fft-ns-1',
    # 'fft-ns-2', 'fft-ns-i', 'nw-ns-i', 'l-ins-i', 'e-ins-i', and 'g-ins-i'.
    Algorithm: G-INS-i
    # Command line options passed to the MAFFT commands
    --maxiterate: 1000
```

29| Execute the command below to create multiple-sequence alignments in FASTA format:

```
$ cd ~/install/pypeline/examples/nature_protocols/phylogeny
$ phylo_pipeline msa --max-threads 16 000_makefile_template.yaml
```

? TROUBLESHOOTING

30| (Optional) Carry out quality filtering of aligned genes before phylogenetic inference. This is accomplished by creating a list of the names of sequences that are to be used in subsequent analyses and saving the list in the same location as the corresponding '.bed' file but using the extension '.NAME.names' instead of '.bed'. To select genes with at least 80% of all bases covered, use the following command:

```
$ cd ~/install/pypeline/examples/nature_protocols/phylogeny
$ ./select_highly_covered_genes.py --min-coverage 0.8 \
  results/P_infestans/alignments/Ensembl.v20.protein_coding.CDS/ \
  > data/regions/Pi_nucl.Ensembl.v20.protein_coding.CDS.HighlyCovered.names
```

31| Create an entry for the phylogenetic inference in the makefile, here named 'HighlyCoveredGenes', and set options as required:

```
PhylogeneticInference:
  # Name of run; results are thereby placed in the folder
  # ./results/P_infestans/phylogenies/HighlyCoveredGenes/
  HighlyCoveredGenes:
    # Root on the clade containing these samples; if not
    # specified, root on the midpoint of the tree.
    RootTreesOn:
      - P1777

    # Include one or more regions of interest
    RegionsOfInterest:
      Ensembl.v20.protein_coding.CDS:
        # Apply different models by codon position (see below)
        Partitions: "112"
        # Limit analysis to the genes selected above
        SubsetRegions: HighlyCovered

    # Settings for examl
    ExaML:
      # Number of replicates of the full phylogeny
      Replicates: 1
      # Number of bootstrap trees
      Bootstraps: 100
      # Model of rate heterogeneity
      Model: GAMMA
```

▲ **CRITICAL STEP** Restricting the phylogeny to the highly covered genes selected in the previous step is accomplished by setting the 'SubsetRegions' value to the name defined in that step, namely, 'HighlyCovered'. The 'Partitions' above allow the user to specify up to ten models per gene (using numbers 0–9) in a position-dependent manner; thus in the above, codon positions 1 and 2 are assigned to model '1', whereas codon position 3 is assigned to model '2'. The sequence of models is repeated across the entire gene, thus a single model per gene may be specified by using '1', or '11' or '111' and so forth. In addition, positions can be excluded by using the label 'X', as in '11X' to exclude the third position.

32| Run the phylogenetic inference (**Fig. 2**):

```
$ phylo_pipeline phylogeny --max-threads
16 000_makefile_template.yaml
```

▲ **CRITICAL STEP** When running ExaML with more than one thread (set with '--examl-max-threads'), it is crucial that the total number of threads does not exceed the current capacity of the server (set with '--max-threads'); doing so will markedly decrease the performance of ExaML processes, leading to markedly longer run times.
? TROUBLESHOOTING

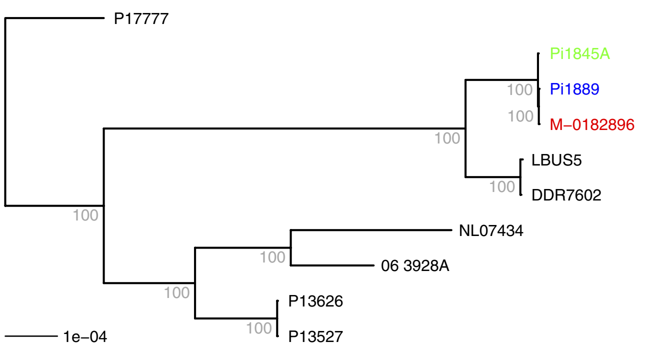

Figure 2 | The *P. infestans* phylogeny produced by the phylogenetic pipeline on the basis of coding sequences with at least 80% of bases covered in the multiple sequence alignment. Node labels show bootstrap support; sample names of ancient samples are colored as in **Figure 3**.

Microbial taxonomic profiling ● TIMING 7 h

33| Decompress FASTQ files and map the sequences by using the Bowtie2 v2.1.0 indexed version of the MetaPhlAn marker database and write the resulting output to a SAM alignment file with the following Unix command:

```
$ cd ~/install/pypeline/examples/nature_protocols/profiling
$ bzcat ../alignment/Pi1889/reads/Pi1889/Pi1889_id_CTTGTA/*/reads.collapsed.bz2 | bowtie2 -x
~/install/metaphlan/bowtie2db/mpa -U - -S Pi1889.id.CTTGTA.bowtie2.out.sam --no-unal
```

34| Sort the input SAM file and convert it to BAM with the following command:

```
$ java -jar ~/install/jar_root/SortSam.jar I=Pi1889.id.CTTGTA.bowtie2.out.sam
O=Pi1889.id.CTTGTA.sorted.bam SO=coordinate
```

? TROUBLESHOOTING

35| Filter the duplicates on the basis of both the 5′ and the 3′ alignment coordinates by using the provided script:

```
$ bam_rmdup_collapsed --remove-duplicates < Pi1889.id.CTTGTA.sorted.bam >
Pi1889.id.CTTGTA.noduplicates.bam
```

36| Collect the names of reads and markers with the following command:

```
$ samtools view Pi1889.id.CTTGTA.noduplicates.bam | awk '{print $1 "\t" $3}' >
Pi1889.id.CTTGTA.noduplicates
```

37| Profile the taxonomy of the microbial metagenomic reads identified:

```
$ ~/install/metaphlan/metaphlan.py Pi1889.id.CTTGTA.noduplicates > Pi1889.id.CTTGTA.txt
```

38| Profile the remaining libraries for samples M-0182896, Pi1845A and Pi1889, as described in Steps 33–37, using the automated shell script provided with the pipeline:

```
$ cd ~/install/pypeline/examples/nature_protocols/profiling
$ ./build_all_profiles.sh
```

? TROUBLESHOOTING

39| Create a 'results' folder in the 'profiling' directory and merge the MetaPhlAn profiles for each library produced in Step 38 by using the 'merge_metaphlan_tables.py' script provided with the MetaPhlAn package[61]; save the resulting table as 'Potato_merged.csv'.

```
$ cd ~/install/pypeline/examples/nature_protocols/profiling
$ mkdir results
$ ~/install/metaphlan/utils/merge_metaphlan_tables.py *.txt > results/Potato_merged.csv
```

40| Import the merged table of taxonomic abundances in R as the variable 'input':

```
$ cd ~/install/pypeline/examples/nature_protocols/profiling/results
$ R
> input <- read.csv("Potato_merged.csv", header=TRUE, row.names=1, sep="\t")
```

The R commands shown here, as well as subsequent R commands, are listed in the 'metagenomic_profile.R' file in the 'profiling/' directory.

41| Select records representing relative abundances at the taxonomical level of interest (here, genus level, expressed as 'g__'), and save the resulting table to the file 'Relative_abundances_genus.csv'.

```
> abundances <- input[row.names(input)[grep("g__[\\w_]*$", row.names(input), perl=TRUE)],]
> row.names(abundances) <- sapply(strsplit(row.names(abundances), "g__"), get("["), 2)
> write.csv(abundances, "Relative_abundances_genus.csv")
```

42| Calculate the number of genera identified and the Shannon diversity index for each profile, followed by the Bray-Curtis distances among profiles. Save summary tables for each calculation as 'Taxon_count_genus.txt', 'Diversity_genus.txt' and 'Distances_genus.txt', respectively.

```
# Load required packages
> library(permute)
> library(vegan)
> library(MASS)
# Calculate the number of genera identified for each profile
> taxon_table <- specnumber(t(abundances))
# Export a summary table
> write.table(taxon_table, "Taxon_count_genus.txt", sep="\t", col.names=FALSE)
# Calculate the Shannon diversity index for each profile at the genus level
> diversity_table <- diversity(t(abundances), index="shannon")
# Export a summary table
> write.table(diversity_table, "Diversity_genus.txt", sep="\t", col.names=FALSE)
# Calculate Bray-Curtis distances among profiles at the genus level
> distances <- vegdist(t(abundances), method="bray")
# Export a summary table
> write.matrix(as.matrix(distances), "Distances_genus.txt", sep="\t")
```

43| Visualize taxonomic profiles performed at the genus level by using a heat map saved as 'Heatmap_genus.pdf', and assess inter-profile relationships by clustering (**Fig. 3a**):

```
# Load required packages
> library(gplots)
# Save heatmap as .pdf file
> pdf("Heatmap_genus.pdf")
# Draw the heatmap (see below)
> heatmap.2(as.matrix(abundances), trace="none", col=colorpanel(50, low="gray91",
mid="darkblue", high="red"), dendrogram="both", ColSideColors=rep(c("red", "green",
"blue"), times=c(5,1,3)), margins=c(12, 13))
# Close the .pdf file
> dev.off()
```

44| Perform a PCoA on the basis of the genus-level Bray-Curtis distances calculated at Step 42; plot and save the first two dimensions in the 'PCOA_genus.pdf' file (**Fig. 3b**).

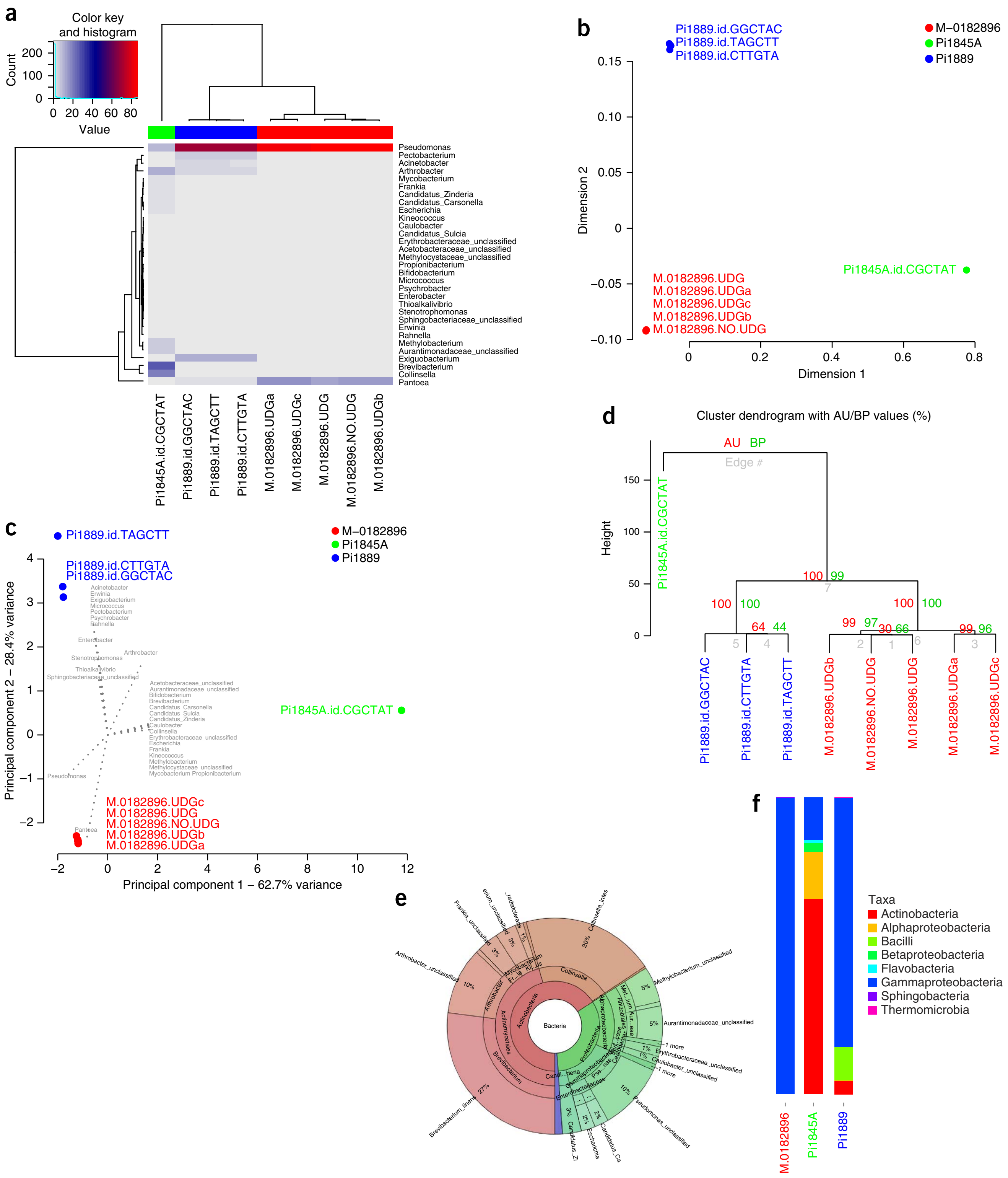

Figure 3 | Analyses of the microbial taxonomical profiles of three *Phytophthora*-infected historical potato samples. (**a**) Heat map of relative abundances of microbial genera. Profiles are color-coded according to samples: M–0182896 in red, Pi1845A in green and Pi1889 in blue. (**b**) PCoAs of Bray-Curtis distances between genus-level microbial profiles (dimensions 1 and 2). (**c**) PCA of genus-level microbial profiles (components 1 and 2; '_u', unclassified). (**d**) Hierarchical clustering of Manhattan distances between library microbial profiles at the genus level (1,000 bootstraps). AU, approximately unbiased *P* values; BP, bootstrap probabilities. (**e**) Krona hierarchical pie chart for sample Pi1889. (**f**) Bar plot of relative abundances of microbial classes per sample. Note that the labels have been tweaked manually to increase legibility.

```
# Perform the Principal Coordinate Analysis
> library(ape)
> pcoa <- pcoa(distances)
# Save the PCA plot as a pdf file.
> pdf("PCOA_genus.pdf")
# Plot the first two dimensions
> plot(pcoa$vectors[,1], pcoa$vectors[,2], pch=16, cex=1.5, cex.axis=0.9, font.lab=2,
font.axis=2, xlab="Dimension 1", ylab="Dimension 2", col=rep(c("red", "green", "blue"),
times=c(5, 1, 3)))
# Add profile name labels, colored using the same color scheme as above.
> text(x=pcoa$vectors[,1], y=pcoa$vectors[,2], labels=row.names(pcoa$vectors), font=2,
cex=0.8, pos=3, col=rep(c("red", "green", "blue"), times=c(5, 1, 3)))
# Add a legend showing the correspondence between profiles and samples.
> legend('topright', legend=c("M-0182896", "Pi1845A", "Pi1889"), pch=16, col=c('red',
'green', 'blue'), bty='n', cex=.75)
> dev.off()
```

45| Visualize similarities among profiles by PCA; plot the first two components and save the PCA plot as 'PCA_genus.pdf' (**Fig. 3c**).

```
# Perform Principal Component Analysis
> pca <- prcomp(t(abundances), scale.=T)
# Calculate the percentage of the variance accounted by principal components 1 and 2
> PC1 <- round (100 * (summary(pca)$importance[2,1]), 1)
> PC2 <- round (100 * (summary(pca)$importance[2,2]), 1)
# Plot PCA scores for principal component 1 and 2
> pdf("PCA_genus.pdf")
# Plot the first two principal components
> plot(pca$x[,1], pca$x[,2], pch=16, cex=1.5, cex.axis=0.9, font.lab=2, font.axis=2,
xlab=paste("Principal component 1 - ", PC1, "% variance", sep=""), ylab=paste("Principal
component 2 - ", PC2, "% variance", sep=""), col=rep(c("red", "green", "blue"),
times=c(5, 1, 3)))
# Add profile name labels, colored using the same color scheme as above.
> text(pca$x[,1], pca$x[,2], colnames(abundances), font=2, cex=0.8, pos=3,
col=rep(c("red", "green", "blue"), times=c(5, 1, 3)))
# Add a legend showing the correspondence between profiles and samples.
> legend('topright', legend=c("M-0182896", "Pi1845A", "Pi1889"), pch=16, col=c('red',
'green', 'blue'), bty='n', cex=.75)
# Plot PCA loadings and their labels
> vectors.x <- (pca$rotation[,1]) * 8
> vectors.y <- (pca$rotation[,2]) * 8
> points(cbind(vectors.x, vectors.y), col="grey50", type="n")
> text(cbind(vectors.x, vectors.y), rownames(cbind(vectors.x, vectors.y)), cex=0.6,
font=2, pos=3, col="grey50")
> for (v in 1:length(vectors.x)) {
segments(0,0, vectors.x[v], vectors.y[v], col="grey50", lty=3, lwd=2.5)
}
> dev.off()
```

46| Perform bootstrapped hierarchical clustering with 1,000 bootstraps by using ‘Manhattan’ distances and the ‘average’ linkage clustering method (‘method.dist’ and ‘method.hclust’ options); save the hierarchical clustering dendrogram as ‘Clustering_genus.pdf’ (**Fig. 3d**):

```
> library(pvclust)
> clustering <- pvclust(abundances, method.dist="manhattan", method.hclust="average",
nboot=1000)
> pdf("Clustering_genus.pdf")
> plot(clustering)
> dev.off()
```

Finally, end the R session.

47| Infer the general microbial profile per sample by merging the information gathered from all libraries.

```
$ cd ~/install/pypeline/examples/nature_protocols/profiling
# Create a directory
$ mkdir -p krona
# Merge libraries per sample
$ cat Pi1889.*.noduplicates > krona/Pi1889.all.noduplicates
$ cat M.0182896_*.noduplicates > krona/M.0182896.all.noduplicates
$ cd krona
# Run MetaPhlAn
$ ~/install/metaphlan/metaphlan.py Pi1889.all.noduplicates > Pi1889.all.txt
$ ~/install/metaphlan/metaphlan.py M.0182896.all.noduplicates > M.0182896.all.txt
```

▲ **CRITICAL STEP** Step 47 can be performed only if a minimum of two libraries are available per sample, if libraries cluster per sample and if hierarchical clustering is statistically significant. The last two conditions imply that the intra-sample variability is significantly lower than the inter-sample variability.

48| Visualize resulting MetaPhlAn profiles with Krona (**Fig. 3e**):

```
$ cd ~/install/pypeline/examples/nature_protocols/profiling/krona
# Convert each of the MetaPhlAn profiles into an input file readable by Krona
$ ~/install/metaphlan/conversion_scripts/metaphlan2krona.py -p M.0182896.all.txt -k
Krona_M.0182896.all.txt
$ ~/install/metaphlan/conversion_scripts/metaphlan2krona.py -p ../Pi1845A.id.CGCTAT.txt -k
Krona_Pi1845A.all.txt
$ ~/install/metaphlan/conversion_scripts/metaphlan2krona.py -p Pi1889.all.txt -k
Krona_Pi1889.all.txt
# Generate one Krona hierarchical pie chart per profile
$ ktImportText Krona_M.0182896.all.txt -o Figure_Krona_M.0182896.all.html
$ ktImportText Krona_Pi1845A.all.txt -o Figure_Krona_Pi1845A.all.html
$ ktImportText Krona_Pi1889.all.txt -o Figure_Krona_Pi1889.all.html
# Visualize the krona pie chart in a web browser (i.e. firefox)
$ firefox Figure_Krona_M.0182896.all.html
$ firefox Figure_Krona_Pi1845A.all.html
$ firefox Figure_Krona_Pi1889.all.html
```

49| Visualize resulting MetaPhlAn profiles in bar plots (**Fig. 3f**):

```
$ cd ~/install/pypeline/examples/nature_protocols/profiling
# Merge the sample profiles into a single table of relative abundances
$ ~/install/metaphlan/utils/merge_metaphlan_tables.py krona/M.0182896.all.txt
Pi1845A.id.CGCTAT.txt krona/Pi1889.all.txt > results/Potato_merged_profiles.csv
$ cd ~/install/pypeline/examples/nature_protocols/profiling/results
$ R
# Import the table of relative abundances
> input2 <- read.csv("Potato_merged_profiles.csv", header=TRUE, row.names=1,
sep="\t")
# generate a table of relative abundances at the class level.
> abundances2 <- input2[row.names(input2)[grep("c__[\\w_]*$", row.names(input2),
perl=TRUE)],]
> row.names(abundances2) <- sapply(strsplit(row.names(abundances2), "c__"), get("["), 2)
> data_table <- data.frame(samples=rep(colnames(abundances2), each=dim(abundances2)[1]),
taxa=rep(row.names(abundances2), dim(abundances2)[2]), datavect=unlist(abundances2))
> library(ggplot2)
> library(grid)
# Draw the stacked bar plot
> ggplot(data_table, aes(x=samples)) + geom_bar(aes(weight=datavect, fill=taxa),
 position='fill') + scale_y_continuous("", breaks=NULL) + scale_fill_manual(values=rainbow
(dim(abundances2)[1])) + theme(axis.text.x=element_text(angle=90, hjust = 0,
 color="black"), legend.text=element_text(size = 8)) + theme(legend.key.size=unit
(0.6, "lines"))
> ggsave("Barplot_class.pdf")
```

? TROUBLESHOOTING
Troubleshooting advice can be found in **Table 1**.

TABLE 1 | Troubleshooting table.

Step	Problem	Possible reason	Solution
1, 9, 11, 23, 29, 32	Not enough space in the '/tmp' partition	By default, the pipeline saves files produced while running tasks in '/tmp/username/' before moving them to their final destination	Ensure that this partition has sufficient space, or change the location using '–temp-root', to prevent filling out /tmp
	The pipeline reports missing input files	Use of relative paths in makefile	The pipeline resolves relative paths from the folder from which the pipeline is run, not from the folder containing the makefile. Change the current working directory to that implied by the paths, or modify the paths in the makefile
	The pipeline reports missing executables	Executables are not installed	Please refer to the documentation for the pipeline and install all required software
		Executables are not in PATH	Please ensure that the installed executable is accessible via the PATH variable in set for the current shell

(continued)

TABLE 1 | Troubleshooting table (continued).

Step	Problem	Possible reason	Solution
	The pipeline reports that version requirements are not met for required software	The program is out of date	Update the required software to meet the minimum requirements specified in the error message
		Multiple versions are installed	Ensure that the up-to-date version of the required software has precedence in the PATH variable. Use the shell command 'which -a EXECUTABLE' to display the location (and precedence) of installed copies of a given executable
1, 9	Java command fails with 'OutOfMemoryError'	Maximum heap-space is insufficient for the current task	Increase the maximum heap-space used by the JVM (default: 4 GB) to, e.g., 8 GB, by using the '–jre-option' switch to pass options to the JRE: `$ bam_pipeline run [...] -jre-option -Xmx8g`

	BWA fails with '[gzclose] buffer error' message	The version of zlib is older than 1.2.4	Upgrade to zlib v1.2.4 or later, or recompile BWA by using the up-to-date version of zlib:
			`$ cd ~/install/`
			`$ curl -O http://zlib.net/zlib-1.2.8.tar.gz`
			`$ tar xvzf zlib-1.2.8.tar.gz`
			`$ cd zlib-1.2.8`
			`$ ./configure`
			`$ make`
			`$ cd ~/install/`
			`$ curl -L -O`
			`http://downloads.sourceforge.net/project/`
			`bio-bwa/bwa-0.5.10.tar.bz2`
			`$ tar xvjf bwa-0.5.10.tar.bz2`
			`$ cd bwa-0.5.10`
			`$ sed -e's#INCLUDES=#INCLUDES=-I../zlib-1.2.8/`
			`#' -e's#-lz#../zlib-1.2.8/libz.a#' Makefile >`
			`Makefile.zlib`
			`$ make -f Makefile.zlib`
			`Add the resulting 'bwa' executable to your PATH`

	The mapDamage modeling step fails because of an error in 'dyn.load(libLFile)'	GNU Scientific Library (GSL) is not installed correctly	Reinstall GSL:
			`$ cd ~/install`
			`$ curl -O ftp://ftp.gnu.org/gnu/gsl/gsl-1.16.`
			`tar.gz`
			`$ tar xvzf gsl-1.16.tar.gz`
			`$ cd gsl-1.16`
			`$ ./configure`
			`$ make`
			`$ sudo make install`

	The mapDamage modeling step fails because of missing RcppGSL	RcppGSL package is installed, but the library is missing	Check whether GSL is missing; if so, follow steps outlined in the previous row:
			`$ R`
			`> library(RcppGSL)`
			`Loading required package:`
			`Rcpp`
			`sh: gsl-config: command not found ...`

10	Preseq 'lc_extrap' fails to run, giving the error 'single estimate failed'	Quick estimate enabled by use of the '-Q' option	Call preseq without the '-Q' option to carry out a complete inference analysis using bootstraps. Note that this markedly increases the running time
	Preseq 'lc_extrap' fails to run, giving the error 'sample not sufficiently deep or duplicates removed'	Library contains insufficient information	The library does not contain sufficient information to extrapolate the results of subsequent sequencing runs. Add additional lanes to library before attempting to carry out the extrapolation
		lc_extrap called without the '-H' option	To signify that the input files are histograms, include the '-H' command-line option when calling lc_extrap

(continued)

TABLE 1 | Troubleshooting table (continued).

Step	Problem	Possible reason	Solution
11, 32	RAxML/ExaML run for a very long time with low CPU utilization	Project folder or temporary directory is located on the network file system	Due to IO usage patterns involving the repeat opening and closing of files, RAxML and ExaML may take an exceedingly long time to complete when running over a network file system; to avoid this issue, move the project folder or temporary folder to a local partition before running the pipeline
27	VEP 'INSTALL.pl' script times out downloading list of genomes	User behind proxy	Set the 'ftp_proxy' environmental variable when running 'INSTALL.pl': `$ env ftp_proxy=hostname:port perl INSTALL.pl`
		Active FTP connection prevented by firewall or proxy	Set the 'FTP_PASSIVE' environmental variable when running 'INSTALL.pl': `$ env FTP_PASSIVE=1 perl INSTALL.pl`
	VEP 'OUTPUT' file does not contain annotations	Annotations are not installed correctly	Please ensure that the annotation files have been installed in the folder corresponding to the current version of the VEP script. For example, for v73: `$ ls ~/.vep/phytophthora_infestans/73/ Phytophthora_infestans.ASM14294v1.20.dna.toplevel.fa Phytophthora_infestans.ASM14294v1.20.dna.toplevel.fa.index info.txt phytophthora_infestans_vep_20.tar.gz supercont1.1 supercont1.10 [...]`
32	examlParser fails, giving the error message 'Empirical base frequency for state number X is equal to zero in DNA data partition NAME_PARTITION'	Bootstrapping has produced a partition missing one or more nucleotides	Either delete the resulting PHYLIP file to force the creation of a new bootstrap alignment, or exclude the gene from the analysis as described in Step 30 by using the NAME part of the name printed by examlParser (this will cause all phylogenetic analytical steps to be rerun)
34, 38	SortSam fails with 'OutOfMemoryError'	Default maximum heap-space is insufficient to carry the out task	Increase the available amount of heap-space for the JVM using the '-Xmx' option to, e.g., 4 GB: `$ java -Xmx4g -jar [...]`

● **TIMING**

Steps 1–10, set up and run the BAM pipeline: 72 h
Steps 11–32, set up and run the phylogenetic pipeline: 38 h
Steps 33–49, microbial taxonomic profiling: 7 h

ANTICIPATED RESULTS

BAM pipeline

Summary files, coverage tables, depth tables and mapDamage2.0 results for the samples Pi1845A, Pi1889 and M-0182896 are included in the pipeline repository in the folder '~/install/pypeline/examples/nature_protocols/example_results/alignment'.

When executed (Steps 1 and 9), the BAM pipeline produces the following files and folders for a target 'T':

- T.summary: summary table of the complete run, including information on the number of reads pre- and post-trimming, coverage, clonality, fraction of endogenous reads and more, aggregated for each library, sample and target. Note that these numbers are estimated after filtering for mapping quality.
- T.*.realigned.bam: final alignment files (for each prefix; Pi_nucl and Pi_mito) after local realignment of indels with the GATK[40].

- T.*.coverage: report of average coverage for a given prefix; coverages are reported for the entire prefix, for each sequence in the prefix (if <100), and by samples and libraries.
- T.*.depths: report of the fractions of sites that have a given minimum depth of coverage (i.e., the percentage of sites covered at least one time, two times, ..., up to 200 times); these numbers are reported for the entire prefix, per sequence (if <100 sequences), and by samples and libraries.
- T.*.mapDamage/: directories (one per prefix) containing mapDamage plots for each library. If rescaling is enabled, these folders also contain the results of parameter estimation for postmortem damage. To estimate parameters for a library for which only plots have been built, use the following command, replacing 'PREFIX' with the path to the FASTA file and replacing 'FOLDER' with the path to the folder containing the output from running mapDamage2.0: `$ mapDamage --stats-only -r PREFIX -d FOLDER`
- T.*.duphist/: directories (one per prefix) contain histogram files for each library. These may be analyzed by using preseq[70] to estimate the complexity of the libraries, and predict yields from additional sequencing efforts. To compute the expected yield of future experiments, use the following command, replacing 'INPUT_HISTOGRAM' with the path to a histogram file generated by the pipeline: `$ lc_extrap -Q -H INPUT_HISTOGRAM`
- T/: folder containing intermediate files produced by the pipeline, with a subfolder for each prefix, and one for (trimmed) reads, the structure of which mirrors the structure in the makefile: Sample/Library/Lane/. It may be removed to save space, at the cost of (potentially) having to rerun the entire pipeline, if any changes need to be made to the project.

Phylogenetic pipeline: genotyping step
After execution of the genotyping step of the phylogenetic pipeline (Steps 10 and 23), the following main files are produced for each target T with prefix P, and they are placed in ./results/Title/genotypes/:

- T.P.protein_coding.CDS.vcf.bgz: compressed VCF file with sample genotype.
- T.P.protein_coding.CDS.filtered.vcf.bgz: filtered VCF file produced by using the 'vcf_filter' tool included with the pipeline, and the command-line options specified in the makefile.
- T.P.protein_coding.CDS.fasta: FASTA records of each named sequence.

The .bgz files are compressed by using the 'bgzip' tool from the Tabix package, and they may be unpacked by using any gzip-compatible tool (gzip, zcat, zless, etc.).

Phylogenetic pipeline: multiple sequence analysis
After the execution of the MSA step (Steps 11 and 29), the following files are produced for each gene G proscribed for regions of interest R, and they are placed in './results/Title/alignments/R/':

- G.fasta: multi-FASTA containing unaligned sequences for each sample.
- G.afa: multi-FASTA containing sequences aligned by using MAFFT.

Each sequence is named according to the name of the sample it was derived from, with the sequence name included as meta-information.

Phylogenetic pipeline: phylogenetic inference
After the execution of the phylogenetic inference step (Steps 11 and 32), the following main files are produced for run 'R', and they are located in './results/Title/phylogenenies/R/':

- bootstraps.newick: file containing the phylogenetic trees inferred from the bootstrap alignments in Newick format; by default the pipeline will produce 100 bootstrap alignments from the original super-matrix and perform maximum-likelihood phylogenetic inference by using ExaML.
- replicates.newick: file containing the phylogenetic trees inferred from the original super-matrix in Newick format; by default the pipeline will perform one maximum-likelihood phylogenetic inference using ExaML.
- replicates.support.newick: the trees from 'replicates.newick' with support values in percent, which are calculated by using the trees in 'bootstraps.newick'.

All Newick trees are either rooted on one or more clades specified in the makefile, or on the midpoint of the tree if no such clades have been specified. The Newick tree produced from the example data, corresponding to the 'replicates.support.newick' file described above, is visualized in **Figure 2** by using FigTree (http://tree.bio.ed.ac.uk/software/figtree/).

Metagenomic pipeline
The output files used in and resulting from the metagenomic analyses of the analyzed example libraries (with the exception of the BAM files) are included in the pipeline repository in the folder '~/install/pypeline/examples/nature_protocols/example_results/profiling'.

TABLE 2 | Files produced by the PALEOMIX metagenomic module.

M-0182896	Pi1845A	Pi1889
M-0182896.UDG.txt	Pi1845A.CGCTAT.txt	Pi1889.id.CTTGTA.txt
M-0182896.UDGa.txt		Pi1889.id.GGCTAC.txt
M-0182896.UDGb.txt		Pi1889.id.TAGCTT.txt
M-0182896.UDGc.txt		
M-0182896.NO_UDG.txt		

TABLE 3 | Diversity statistics of the genus-level microbial profiles.

Profile	Number of taxa identified	Shannon diversity index
M-0182896.NO_UDG	2	0.41
M-0182896.UDG	2	0.40
M-0182896.UDGa	5	0.45
M-0182896.UDGb	3	0.42
M-0182896.UDGc	7	0.47
M.0182896.all	9	0.45
Pi1845A.id.CGCTAT	19	2.30
Pi1889.id.CTTGTA	15	1.12
Pi1889.id.GGCTAC	14	1.11
Pi1889.id.TAGCTT	17	1.13
Pi1889.all	24	1.15

The regular MetaPhlAn output file is a tab-delimited file with two columns, containing the taxonomy of the taxon identified (k: kingdom; p: phylum; c: class; o: order; f: family; g: genus; s: species) and its relative abundance in percentages. The final MetaPhlAn output text files containing the microbial profile of each sample are listed in **Table 2**.

The 'Potato_merged.csv' file obtained after combining the MetaPhlAn profiles of each of our example files (Step 39) contains 10 columns and 126 lines. The first column contains the identified taxa, and the remaining columns contain the percentage-wise abundances of each taxon for each library. The columns are ordered alphabetically, so that the five first columns correspond to the profiles obtained for sample 'M-0182896', the next column to sample 'Pi1845A' and the last three columns to sample 'Pi1889'. As a result of the Metagenomic pipeline, summary tables are created for each library, including the number of taxa identified and Shannon diversity indexes, as shown in **Table 3**.

In addition, a whole set of graphical outputs in the form of .pdf files are generated. In our example, using genus as the selected taxonomical level, graphical outputs are shown in **Figure 3** for the samples M-0182896, Pi1845A and Pi1889, including:

- A heat map of relative abundances of microbial genera (**Fig. 3a**)
- PCoAs of Bray-Curtis distances between genus-level microbial profiles (dimensions 1 and 2; **Fig. 3b**)
- PCA of genus-level microbial profiles (components 1 and 2; **Fig. 3c**)
- Hierarchical clustering of Manhattan distances between microbial profiles at the genus level (**Fig. 3d**)
- Krona hierarchical pie chart (one per sample, sample Pi1845A shown only; **Fig. 3e**)
- Bar plot of relative abundances of microbial classes (**Fig. 3f**)

In our example of bootstrapped hierarchical clustering (**Fig. 3d**), for the edge segregating the Pi1845A profiles from the Pi1889 and M-0182896 profiles (edge #8) and the edge separating the Pi1889 profiles from the M-0182896 profiles (edge #7), the approximately unbiased (AU) P values were 1.000 ± 0.000 and 0.997 ± 0.002, respectively. Increasing the bootstraps to 10,000 iterations led to AU P values of 1.000 ± 0.000 (edge #8) and 0.997 ± 0.001 (edge #7). This suggests that the clustering according to samples detected at Steps 44–46 is significantly supported. Note that the exact values resulting from the bootstrap analysis will differ from run to run owing to the random nature of bootstrapping.

The table of relative abundances for the merged samples created in Step 49 (variable 'input2') can be used to perform all the statistical analyses described in Steps 41–48. Steps 43–46 can be performed independently from each other after completing Step 41, but Step 42 is required before proceeding to Step 44.

Note: Any Supplementary Information and Source Data files are available in the online version of the paper.

ACKNOWLEDGMENTS This work was supported by the Danish Council for Independent Research, Natural Sciences (FNU); the Danish National Research Foundation (DNRF94); a Marie Curie Career Integration grant (FP7 CIG-293845); and the Marie Curie FP7 Initial Training Network (EUROTAST). The work of M.S. was made possible thanks to the support of the Lundbeck foundation (R52-A5062). A.G. and L.E. were supported by Marie Curie Intra-European Fellowships (FP7 IEF-299176 and IEF-302617, respectively). R.F. was supported by a postdoctoral grant from AXA Research Fund (32983).

AUTHOR CONTRIBUTIONS M.S., H.J., A.G. and L.O. designed the BAM pipeline, with feedback from R.S. and M.M. M.S., A.G. and L.O. designed the phylogenetic pipeline. M.S. wrote the scripts for both pipelines, based in part on code written by M.K. L.E. and C.D.S. designed the metagenomic pipeline and scripts.

M.D.M. guided selection and analysis of the example data sets. H.J., R.S., A.G., M.D.M., R.F., M.K., M.M. and L.O. tested the pipelines. L.O. coordinated the work. M.S., L.E., C.D.S., R.F. and L.O. wrote the manuscript with contributions from all the authors.

COMPETING FINANCIAL INTERESTS The authors declare no competing financial interests.

1. Keane, T.M. *et al.* Mouse genomic variation and its effect on phenotypes and gene regulation. *Nature* **477**, 289–294 (2011).
2. McVean, G.A. *et al.* An integrated map of genetic variation from 1,092 human genomes. *Nature* **491**, 56–65 (2012).
3. Nielsen, R., Paul, J.S., Albrechtsen, A. & Song, Y.S. Genotype and SNP calling from next-generation sequencing data. *Nat. Rev. Genet.* **12**, 443–451 (2011).
4. Alkan, C., Coe, B.P. & Eichler, E.E. Genome structural variation discovery and genotyping. *Nat. Rev. Genet.* **12**, 363–376 (2011).
5. Thurman, R.E. *et al.* The accessible chromatin landscape of the human genome. *Nature* **489**, 75–82 (2012).
6. Ball, M.P. *et al.* Targeted and genome-scale strategies reveal gene-body methylation signatures in human cells. *Nat. Biotechnol.* **27**, 361–368 (2009).
7. Wheeler, D.A. *et al.* The complete genome of an individual by massively parallel DNA sequencing. *Nature* **452**, 872–876 (2008).
8. Li, R. *et al.* The sequence and *de novo* assembly of the giant panda genome. *Nature* **463**, 311–317 (2010).
9. Overballe-Petersen, S., Orlando, L. & Willerslev, E. Next-generation sequencing offers new insights into DNA degradation. *Trends Biotechnol.* **30**, 364–368 (2012).
10. Margulies, M. *et al.* Genome sequencing in microfabricated high-density picolitre reactors. *Nature* **437**, 376–380 (2005).
11. Poinar, H.N. *et al.* Metagenomics to paleogenomics: large-scale sequencing of mammoth DNA. *Science* **311**, 392–394 (2006).
12. Miller, W. *et al.* Sequencing the nuclear genome of the extinct woolly mammoth. *Nature* **456**, 387–390 (2008).
13. Rasmussen, M. *et al.* Ancient human genome sequence of an extinct Palaeo-Eskimo. *Nature* **463**, 757–762 (2010).
14. Green, R.E. *et al.* A draft sequence of the Neandertal genome. *Science* **328**, 710–722 (2010).
15. Krause, J. *et al.* The complete mitochondrial DNA genome of an unknown hominin from southern Siberia. *Nature* **464**, 894–897 (2010).
16. Reich, D. *et al.* Genetic history of an archaic hominin group from Denisova Cave in Siberia. *Nature* **468**, 1053–1060 (2010).
17. Rasmussen, M. *et al.* An Aboriginal Australian genome reveals separate human dispersals into Asia. *Science* **334**, 94–98 (2011).
18. Keller, A. *et al.* New insights into the Tyrolean Iceman's origin and phenotype as inferred by whole-genome sequencing. *Nat. Commun.* **3**, 698 (2012).
19. Raghavan, M. *et al.* Upper Palaeolithic Siberian genome reveals dual ancestry of Native Americans. *Nature* **505**, 87–91 (2014).
20. Prufer, K. *et al.* The complete genome sequence of a Neanderthal from the Altai Mountains. *Nature* **505**, 43–49 (2014).
21. Meyer, M. *et al.* A high-coverage genome sequence from an archaic Denisovan individual. *Science* **338**, 222–226 (2012).
22. Orlando, L. *et al.* Recalibrating *Equus* evolution using the genome sequence of an early Middle Pleistocene horse. *Nature* **499**, 74–78 (2013).
23. Millar, C.D. & Lambert, D.M. Ancient DNA: towards a million-year-old genome. *Nature* **499**, 34–35 (2013).
24. Skoglund, P. *et al.* Origins and genetic legacy of Neolithic farmers and hunter-gatherers in Europe. *Science* **336**, 466–469 (2012).
25. Lindqvist, C. *et al.* Complete mitochondrial genome of a Pleistocene jawbone unveils the origin of polar bear. *Proc. Natl. Acad. Sci. USA* **107**, 5053–5057 (2010).
26. Miller, W. *et al.* Polar and brown bear genomes reveal ancient admixture and demographic footprints of past climate change. *Proc. Natl. Acad. Sci. USA* **109**, E2382–E2390 (2012).
27. Gilbert, M.T. *et al.* Intraspecific phylogenetic analysis of Siberian woolly mammoths using complete mitochondrial genomes. *Proc. Natl. Acad. Sci. USA* **105**, 8327–8332 (2008).
28. Gilbert, M.T. *et al.* Paleo-Eskimo mtDNA genome reveals matrilineal discontinuity in Greenland. *Science* **320**, 1787–1789 (2008).
29. Bon, C. *et al.* Coprolites as a source of information on the genome and diet of the cave hyena. *Proc. Biol. Sci.* **279**, 2825–2830 (2012).
30. Dabney, J. *et al.* Complete mitochondrial genome sequence of a Middle Pleistocene cave bear reconstructed from ultrashort DNA fragments. *Proc. Natl. Acad. Sci. USA* **110**, 15758–15763 (2013).
31. Fu, Q. *et al.* DNA analysis of an early modern human from Tianyuan Cave, China. *Proc. Natl. Acad. Sci. USA* **110**, 2223–2227 (2013).
32. Lari, M. *et al.* The complete mitochondrial genome of an 11,450-year-old aurochsen (*Bos primigenius*) from Central Italy. *BMC Evol. Biol.* **11**, 32 (2011).
33. Vilstrup, J.T. *et al.* Mitochondrial phylogenomics of modern and ancient equids. *PLoS ONE* **8**, e55950 (2013).
34. Haus, T. *et al.* Mitochondrial diversity and distribution of African green monkeys (*Chlorocebus* Gray, 1870). *Am. J. Primatol.* **75**, 350–360 (2013).
35. Meyer, M. *et al.* A mitochondrial genome sequence of a hominin from Sima de los Huesos. *Nature* **505**, 403–406 (2014).
36. Burbano, H.A. *et al.* Targeted investigation of the Neandertal genome by array-based sequence capture. *Science* **328**, 723–725 (2010).
37. Bos, K.I. *et al.* A draft genome of *Yersinia pestis* from victims of the Black Death. *Nature* **478**, 506–510 (2011).
38. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25**, 1754–1760 (2009).
39. Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25**, 2078–2079 (2009).
40. McKenna, A. *et al.* The genome analysis toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.* **20**, 1297–1303 (2010).
41. Wang, K., Li, M. & Hakonarson, H. ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic Acids Res.* **38**, e164 (2010).
42. Schubert, M. *et al.* Improving ancient DNA read mapping against modern reference genomes. *BMC Genomics* **13**, 178 (2012).
43. Krause, J. *et al.* A complete mtDNA genome of an early modern human from Kostenki, Russia. *Curr. Biol.* **20**, 231–236 (2010).
44. Ginolhac, A., Rasmussen, M., Gilbert, M.T., Willerslev, E. & Orlando, L. mapDamage: testing for damage patterns in ancient DNA sequences. *Bioinformatics* **27**, 2153–2155 (2011).
45. Jonsson, H., Ginolhac, A., Schubert, M., Johnson, P.L. & Orlando, L. mapDamage2.0: fast approximate Bayesian estimates of ancient DNA damage parameters. *Bioinformatics* **29**, 1682–1684 (2013).
46. Martin, M.D. *et al.* Reconstructing genome evolution in historic samples of the Irish potato famine pathogen. *Nat. Commun.* **4**, 2172 (2013).
47. Yoshida, K. *et al.* The rise and fall of the *Phytophthora infestans* lineage that triggered the Irish potato famine. *Elife* **2**, e00731 (2013).
48. Kircher, M. Analysis of high-throughput ancient DNA sequencing data. *Methods Mol. Biol.* **840**, 197–228 (2012).
49. Goecks, J., Nekrutenko, A. & Taylor, J. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.* **11**, R86 (2010).
50. Lindgreen, S. AdapterRemoval: easy cleaning of next-generation sequencing reads. *BMC Res. Notes* **5**, 337 (2012).
51. Briggs, A.W. *et al.* Removal of deaminated cytosines and detection of *in vivo* methylation in ancient DNA. *Nucleic Acids Res.* **38**, e87 (2010).
52. Langmead, B. & Salzberg, S.L. Fast gapped-read alignment with Bowtie 2. *Nat. Methods* **9**, 357–359 (2012).
53. Briggs, A.W. *et al.* Patterns of damage in genomic DNA sequences from a Neandertal. *Proc. Natl. Acad. Sci. USA* **104**, 14616–14621 (2007).
54. McLaren, W. *et al.* Deriving the consequences of genomic variants with the Ensembl API and SNP Effect Predictor. *Bioinformatics* **26**, 2069–2070 (2010).
55. Katoh, K. & Standley, D.M. MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Mol. Biol. Evol.* **30**, 772–780 (2013).
56. de Queiroz, A. & Gatesy, J. The supermatrix approach to systematics. *Trends Ecol. Evol.* **22**, 34–41 (2007).
57. Stamatakis, A. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics* **22**, 2688–2690 (2006).
58. Fierer, N. *et al.* Cross-biome metagenomic analyses of soil microbial communities and their functional attributes. *Proc. Natl. Acad. Sci. USA* **109**, 21390–21395 (2012).
59. Cotillard, A. *et al.* Dietary intervention impact on gut microbial gene richness. *Nature* **500**, 585–588 (2013).
60. Schloissnig, S. *et al.* Genomic variation landscape of the human gut microbiome. *Nature* **493**, 45–50 (2013).

61. Segata, N. *et al.* Metagenomic microbial community profiling using unique clade-specific marker genes. *Nat. Methods* **9**, 811–814 (2012).
62. Altschul, S.F. *et al.* Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389–3402 (1997).
63. Huson, D.H., Richter, D.C., Mitra, S., Auch, A.F. & Schuster, S.C. Methods for comparative metagenomics. *BMC Bioinformatics* **10** (suppl. 1), S12 (2009).
64. Human Microbiome Project Consortium. Structure, function and diversity of the healthy human microbiome. *Nature* **486**, 207–214 (2012).
65. Der Sarkissian, C., Ermini, L., Jónsson, H., Alekseev, A.N., Crubezy, E., Shapiro, B. & Orlando, L. Shotgun microbial profiling of fossil remains. *Mol. Ecol.* doi:10.1111/mec.12690 (2014).
66. Ondov, B.D., Bergman, N.H. & Phillippy, A.M. Interactive metagenomic visualization in a Web browser. *BMC Bioinformatics* **12**, 385 (2011).
67. R Development Core Team. *R: A Language and Environment for Statistical Computing*, http://www.r-project.org/ (2013).
68. Suzuki, R. & Shimodaira, H. Pvclust: an R package for assessing the uncertainty in hierarchical clustering. *Bioinformatics* **22**, 1540–1542 (2006).
69. Quinlan, A.R. & Hall, I.M. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26**, 841–842 (2010).
70. Daley, T. & Smith, A.D. Predicting the molecular complexity of sequencing libraries. *Nat. Methods* **10**, 325–327 (2013).
71. Li, H. Tabix: fast retrieval of sequence features from generic TAB-delimited files. *Bioinformatics* **27**, 718–719 (2011).
72. Venables, W.N. & Ripley,, B.D. *Modern Applied Statistics with S* (Springer, 2002).
73. Paradis, E., Claude, J. & Strimmer, K. APE: Analyses of Phylogenetics and Evolution in R language. *Bioinformatics* **20**, 289–290 (2004).
74. Wickham, H. *ggplot2: Elegant Graphics for Data Analysis* (Springer, 2009).
75. Haas, B.J. *et al.* Genome sequence and analysis of the Irish potato famine pathogen *Phytophthora infestans*. *Nature* **461**, 393–398 (2009).
76. Avila-Adame, C. *et al.* Mitochondrial genome sequences and molecular evolution of the Irish potato famine pathogen, *Phytophthora infestans*. *Curr. Genet.* **49**, 39–46 (2006).
77. Cock, P.J., Fields, C.J., Goto, N., Heuer, M.L. & Rice, P.M. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res.* **38**, 1767–1771 (2010).