CursedForest - A Random Forest Machine Learning Implementation for “Big” and “Wide” Data

Aidan O’Brien1,2, Piotr Szul3, Robert Dunne4, Paul Leo5, Emma L. Duncan5,6,7, Natalie A. Twine1,8, Stephanie Li9, James Doecke9, Nick Ellis10, Oscar J. Luo1 and Denis C. Bauer1\*

1 Health & Biosecurity, CSIRO, Sydney, NSW, Australia

2 ANU, Canberra, Australia

3 Data61, CSIRO, Brisbane, QLD, Australia

4 Data61, CSIRO, Sydney, NSW, Australia

5 Institute of Health and Biomedical Innovation, Queensland University of Technology, Brisbane, QLD, Australia

6 Department of Endocrinology, Royal Brisbane and Women's Hospital, Brisbane, QLD, Australia

7 Faculty of Medicine and Biomedical Sciences, University of Queensland, Brisbane, QLD, Australia

8 MQ, QLD, Australia

9 Health & Biosecurity, CSIRO, Brisbane, QLD, Australia

10 Oceans & Atmosphere, CSIRO, Brisbane, QLD, Australia

**Modern genomic sequencing studies have seen increasingly large cohort in the past decade. This lead to genomic sequencing datasets with not only bigger sample sizes, but also, more crucially, exponentially growing variant space. Traditional data analysis approaches are unable to cater such kind of big and wide datasets. Even the machine learning library in the highly-parallelised Spark computing engine suffer from this “the curse of dimensionality”. We introduce CursedForest, a tailored implementation of random forests, designed to handle data with extremely large number of variables per sample. With public genome sequencing and SNP array datasets, we demonstrate the utilities of CursedForest for accurately predicting sample ethnicity and identifying known and novel variants associated with the phenotype of interest.**

The digital revolution is seeing a dramatic increase in data collected about almost every aspect oflife~\cite{Loebbecke2015}. These data sets are not only growing vertically, by capturing more events, but also horizontally by capturing more information about these events. The challenge of “big” and “wide” data is especially pronounced in the biomedical space where, for example, whole genome sequencing (WGS) technology enables researchers to interrogate all 3 billion base pairs of the human genome.

Identifying the relevant or disease specific genetic variations between individuals is the focus of genome wide association studies (GWAS). These analyses are typically performed on only the most frequently differing base pairs, termed single nucleotide polymorphisms (SNPs), by applying linear or logistic regression analysis to each SNP separately~\cite{CCC2007}. It has since been demonstrated that not taking interactions between SNPs into account when filtering for potential disease mutations is inadequate~\cite{Manolio2009,Yang2011} and the approach should in-fact be extended to interrogating all genomic positions[CITATION WES vs WGS?].

Hence using more sophisticated machine learning (ML) approaches, in particular tree-based models, have been successful for taking the interaction of variables into account~\cite{Wright.et.al.2016}. In addition, random forests \cite{Breiman2001} are well suited for processing “wide” genomic data for two reasons: firstly, while other machine learning applications have the propensity to overfit data sets with more features $p$ than samples $n$ (a consequence of the “curse of dimensionality''~\cite{Bauer2014, bellman1961adaptive}), decision trees are resistant to overfitting; secondly, random forests are easy to parallelise, as the forest is a sum of decision trees, it is possible to grow separate trees on different compute nodes and combine the trees.

The most popular random forest software packages to date are R-based, although it has been reported that current implementations do not scale well with increasing feature size~\cite{Wright.and.Ziegle.2016}. This is despite R now supporting long vectors (length greater than $2^{31}-1$), and the \randomforest implementation in R lending itself to parallelisation by growing multiple trees simultaneously and combining them for the final result~\cite{Liaw.and.Weiner.2002}. More suitable implementations for “wide” genomic data have been developed in other programming languages. \textsc{CloudForest} \cite{Bressler2015}, written in Go, achieves fast run times by effective use of the CPU cache, optimising for different classes of features and efficiently

multithreading. Similarly, \ranger~\cite{Wright.and.Ziegle.2016}, written in C++ with an R front-end, is specifically optimised for dealing with large data sets.

However, the use of traditional compute infrastructure limits the parallelisation strategies that can be employed. The programs are limited to utilising only CPUs that are on the same computer node (multithreading) or farm out independent tasks to CPUs distributed across nodes that do not require communication between the processes (a separate tree is grown on each node). Hadoop/Spark overcomes these limitations by enabling programs to scale beyond compute-node boundaries and hence enable more sophisticated parallelisation strategies. In the case of random forests, the computations for each node of a tree can hence be handed off to separate processors.

Despite overcoming the node-boundary limitation, the standard implementation of random forest in Spark ML is not able to handle the extremely ``wide'' genomic data as it was developed for a large number of samples with only modest dimensionality~\cite{NIPS2016\_6366}. Although Spark ML can build a random forest model on a subset of the data (chromosome 1), we show that the time taken is excessive due to the large amount of data being aggregated and processed by the driver node during intermediate stages of building the model. This unbalanced work load where the driver node becomes the bottleneck and worker nodes being idle prevents a seamless scaling to larger data sets. We also show that the memory requirements per executor increases with dimensionality due to the data types Spark ML uses.%, which we elaborate on in the discussion.

Here we introduce \cursedforest, a tailored Hadoop/Spark-based implementation of random forests specifically designed to cater for “big” (many samples) and “wide” (many features) data sets. \cursedforest extends our previously developed variant interpretation framework, VariantSpark~\cite{OBrien2015}, to now offer supervised as well as unsupervised ML algorithms in the Spark framework. In our implementation, a Spark application runs on a ``driver'' node and distributes tasks to many ``worker'' nodes, or ``executors''. By also utilising Spark to read in and manipulate the standard genomic variant format (VCF) directly, \cursedforest outperforms existing tools even on small data sets where multithreading generally performs well. Harnessing the virtually unlimited capability to parallelise tasks, \cursedforest can hence explore the solution space faster by building a larger number of diverse models to generate a consensus classifier.

Using this facility, \cursedforest is capable of parallelising the split for each node in a tree thereby handling millions of features, as required to process whole genome sequencing data or SNP array data with unobserved genotypes imputed~\cite{Howie2012}. This provides the potential to generate data sets of hundreds of thousands of individuals with millions of variants (imputing the GWAS catalog), highlighting the need for modern compute paradigms in the genomics space.

VariantSpark~\cite{OBrien2015} with the \cursedforest extension therefore offers acomprehensive analysis toolkit that can scale to future data demands. To showcase the framework's ability we demonstrate

a classification as well as feature-selection task on synthetic data in the first section. In the second section, we demonstrate the ability of \cursedforest to successfully replicate findings from a previous GWAS study as well as identify novel variants associated with bone mineral density (BMD). Thirdly, we demonstrate the scalability of \cursedforest in respect to the dimensionality of data by building a random forest model on whole-genome data from the 1000 Genomes Project~\cite{1KG2012} to predict ethnicity. Finally, given the role different parameter values can play in model construction, we explore the effect that tuning these parameters can have on the prediction accuracy of the model.

**RESULTS**

**Cursedforest**

VariantSpark made use of Spark's machine learning algorithms ``Spark ML''. While Spark ML algorithms demonstrate scalability when dealing with a large number of samples, this scalability eventually breaks down as we include more features. As is standard with Spark applications, we store our data in a Resilient Distributed Dataset (RDD), where an RDD is essentially a collection of elements. In the case of Spark ML, each element in the RDD is a sample. RDDs contribute to the scalability of Spark as they can be distributed across multiple nodes and operated on in parallel. Even as we add more samples to a dataset, Spark can simply schedule extra tasks to handle the additional items in the RDD.

However, within an RDD, Spark ML stores each sample as a vector. Unlike RDDs, which can be partitioned and distributed across multiple nodes, each vector must be present in its entirety on any node accessing it. This is no problem with typical data sets; however as dimensionality increases, the vectors eventually reach a size where they can no longer fit into a single node's memory.

So in the case of adding more samples, Spark ML can simply create more tasks, keeping memory consumption within the cluster's bounds. However, as the dimensionality of each sample grows, the memory requirements of the job increases to enable these increasingly large vectors to be loaded into memory.

On the other hand, \cursedforest is specifically designed to handle wide ``cursed'' data. It avoids the relation between memory and dimensionality by avoiding calculations that rely on entire feature vectors and taking the parallelization work down to the level of the individual features. For each node of a tree, \cursedforest will distribute tasks that consist of single features (variants), for every individual. Each of these tasks will calculate the information gain for that specific feature. Once these tasks have completed, the results are reduced to return the feature which gives the greatest information gain. This process is then repeated until \cursedforest has created the entire decision tree.

The current implementation of \cursedforest uses a ``Gini impurity'' criteria for splitting. Let $f\_q$ be the fraction of items labeled with value $q$, where $q = 1, \ldots, Q$ at a node. The Gini impurity is

$$

I\_G(f) = \sum\_{q = 1}^Q f\_q ( 1 - f\_q ),

$$

which is at a minimum when all observations at the node are in the same class.

**GWAS-style analyses using \cursedforest faithfully identify known disease-accociated genes**

In this section we use \cursedforest to perform feature selection on over 7.2 million genomic variants and identify the genomic locations associated with bone mineral density (BMD)~\cite{Duncan.2011}. We rank variants by their Gini impurity, with the most important one ranked at position 1, and return the top 100,000 variants.

To test whether the associated variants have biological meaning, we investigate if variants in or near BMD-associated genes are enriched toward the top of our ranked list. To form the list of ``known genes'', we combine the 26 known and 6 novel BMD genes listed by Duncan {\it et al.}~\cite{Duncan.2011} with the genes identified by Estrada {\it et al.}~\cite{Estrada2012} and Lui {\it et al.}~\cite{Liu2008}, resulting in 102 genes. All genes are covered within the top 40,000 variants and a gene name is assigned to the variant if it falls within 100 kb of this gene.

The known genes rank significantly higher than variants from other locations in the genome (Mann-Whitney U, $p=4.5\times10^{-14}$, Supplemental File 1), supporting our hypothesis that biologically relevant features are contributing more information to our classification model.

As shown in Figure 1a, the top 30 features are dominated by variants from five well established BMD genes; TNFRSF11B (TNF Receptor Superfamily Member 11b or Osteoprotegerin), CTNNB (Catenin Beta 1), ZBTB40 (Zinc Finger And BTB Domain-Containing Protein 40), IBSP (Integrin Binding Sialoprotein) and MEF2C (Myocyte Enhancer Factor 2C). The BMD association has been independently replicated for four of the five genes: TNFRSF11B~\cite{Rivadeneira2009,Zhang2014,Kemp2014,Paternoster2013}, CTNNB1~\cite{Pei2016, Rivadeneira2009}, ZBTB40~\cite{Rivadeneira2009, Zhang2014,Nielson2016}, and MEF2C~\cite{Pei2016a}. The association of IBSP has not been replicated in other GWAS studies, but homozygous knockouts of this gene have a known skeletal phenotype in mice. Mice deficient in IBSP have impaired bone growth and mineralization, with dramatically reduced bone formation~\cite{Malaval2008}. CTNBB1, MEF2C and TNFRSF11B also have known skeletal phenotypes. CTNNB1 is a key downstream member of the Wnt signaling pathway, and deletion of CTNNB1 in mice results in severe osteopenia, while constitutive activation of CTNNB1 results in massively increased bone deposition~\cite{Holmen2005}. MEF2C is a transcription factor which controls bone mass by regulating osteoclastic bone resorption, where mice with a MEF2C deletion in osteocytes had increased bone mass compared with normal controls~\cite{Kramer2012}. TNFRSF11B (or OPG) is a secreted glycoprotein which regulates bone mass~\cite{Simonet1997}. ZBTB40 has no known biological or molecular involvement in skeletal phenotypes~\cite{Styrkarsdottir2008}, even though it has been associated with BMD in multiple GWAS and is expressed in bone. SNPs in linkage disequilibrium (LD) with each other at these loci also show a high importance score, as exemplified by TNFRSF11B in Figure 3b (Supplemental Figure 2 and 3 for the other genes).

In contrast, one of the lowest ranked gene with literature implied BMD association is DCDC5 (Doublecortin Domain Containing 5) ranked 23,842 (see Supplemental Figure 1). Similar to the conclusions drawn by Duncan {\it et al.}~\cite{Duncan.2011}, our results also indicate that this gene does not have a strong association with BMD. DCDC5 is not highly expressed in bone, and any mechanisms by which this gene might regulate BMD are unclear \cite{Thakker2012}. Although variants near DCDC5 are associated with BMD \cite{Rivadeneira2009}, it is questionable as to whether the gene itself contributes to BMD. Similarly, the lowest ranked gene with annotated BMD links, MAPT (microtubule associated protein tau) at 36,262, is primarily expressed in the nervous system and has not been identified in bone tissue. It is therefore likely that the association is through a co-morbidity with Alzheimer's disease~\cite{Dengler-Crish2017} and not a major contributing factor in this cohort.

**\cursedforest amplifies signal over traditional methods to identify novel associations**

Figure 1b shows a traditional GWAS analysis using logistic regression~\cite{Duncan.2011} where the resulting $p$-values do not clearly distinguish these known BMD genes from the noise. In contrast, \cursedforest seems to amplify the signal to identify associated features (Figure 1a). As such, this approach allows to employ a very stringent cut-off for identifying strongly associated feature. Here we chose a 99.98 False Discovery Rate (FDR) from 60 random experiments (see Methods).

Above this FDR cut-off we identify four novel locations as annotated in Figure 1a.

Most notably a lincRNA (LINC01006) on chromosome 7, which has been associated with serum creatinine levels in two independent GWAS studies~\cite{Pattaro2010,Chambers2010}. Subjects with low serum creatinine levels were at a higher risk for low BMD~\cite{Huh2015} and in a rat study high creatine monohydrate diet resulted in greater lumbar BMD~\cite{Antolic2007}, with creatinine being the breakdown product of creatine. The variants overlapping the 5' end of the lincRNA are the highest ranked features in this region and all LD SNPs also show high importance (see Fig 3d). Also amongst the top 30 features is PRODH (proline dehydrogenase) located on 22q11, a loci observed by Stagi {\it et al.}~\cite{Stagi.2010} to significantly reduce bone mass in individuals with a micro-deletion in this region. PRODH catalyses the first step in proline degradation, and as proposed by Phang {\it et al.}~\cite{Phang.2008}, proline is made available by collagen degradation. Again, all LD SNPs also show high importance (see Figure 3c). In contrast rs17195090 and rs112558362 are single feature loci (Supplementary Figure 4) hence likely due to erroneous imputation or genotyping.

To evaluate whether the here identified features are significantly associated with BMD, we train a traditional linear mixed model to return a $p$-value for each included feature. Interestingly, while the three known BMD genes with the highest Importance score are indeed significantly associated (TNFRSF11B $p$-value=XX, CTNNB1 XX, ZBTB40 XX), IBSP and MEF2C were not significant. All of our newly associated features, on the other hand, were deemed significant, and in-fact rs17195090 was the most significant feature evaluated with $p$-value=XX (rs112558362 XX, LINC01006 XX, PRODH XX).

**\cursedforest outperforms existing methods for classification problems**

In this section we demonstrate \cursedforest's classification accuracy on biological data and compare its performance against other published methods. We do this by predicting the ethnicity of the individuals in the 1000 Genomes Project from their genomic profiles and evaluate against the annotated label. We record the prediction error (OOB) as well as the runtime for each method. We limit the number of executors available to \cursedforest to 12 to keep the comparison fair for all methods.

As shown in Fig.~\ref{figure:1000genomes}, only \cursedforest is able to scale to the full phase3\\_chr1-22 dataset (2,504 samples x 19,328,051 variants), with the standard random forest implementation in R running out of resources on the relatively low-dimensional phase1\\_chr22 dataset (1,092 x 490,036). Ranger's c++ implementation and the R front-end version of it are able to process the second largest dataset (phase3\\_chr1-3 with 2,504 x 19,328,051), however, both are unable to process larger data sets successfully. While the c++ implementation runs out of memory the R front-end, which also covers the loading and pre-processing, needs an unreasonable compute time for this task (>20h), see Supplementary Table 1. The standard implementation of random forest in SparkML already runs out of memory after processing the third largest dataset, phase3\\_chr1 (2,504 x 6,450,364). This demonstrates that Spark ML, although designed to handle many samples, is unable to efficiently cope with large numbers of features.

All tested methods have a worse runtime than \cursedforest for the data sets they processed successfully, except for the smallest dataset where the overhead of Spark's distributed platform made ranger's R implementation faster with 1 min 10 seconds compared to 1 min 50 seconds. For example, on phase3\\_chr1-3 \cursedforest finishes in 1 hour, with ranger(R) requiring 4 hours and SparkML already needing 8 hours on a smaller dataset. Extrapolating from this, all comparable implementations hence would require more than 24 hours to finish, assuming the use of larger memory nodes.

As our implementation can efficiently utilise large numbers of commodity nodes, the performance of \cursedforest can be substantially improved further. Table~\ref{cursedforesttable} shows the runtime when utilising the full Spark cluster (130 worker nodes).

Building a model with \cursedforest on phase3\\_chr1, which previously took 20 minutes, can be reduced to 4 minutes. Subsequently, phase3\\_chr1-3 takes about 11 minutes, and building a model on the entire genome is 37 minutes. Not only does the job complete successfully using \cursedforest, but building a random

forest model on all 22 autosomes (phase3\\_chr1-22) leads to a more accurate result. The error can be reduced from OOB=0.04 when using only data from Chr1, which represents processing power limit for most tools, down to OOB=0.01 when using the full dataset.

**cursedforest can evaluate variable importance and perform classification for millions of features}**

In this section we test the limits of \cursedforest's association testing and classify ability on high-dimensional data. We therefore generate a synthetic dataset with 2.5 million variables ($p=2,500,000$), of which 5 are related to the response variable, and 5000 samples ($n=5000$), as discussed in the methods section.

We fit the random forest model and estimate the classification accuracy by capturing the out-of-bag error (OOB) and the variable selection performance by capturing the rank-biased overlap (RBO) \cite{Webber2010} measure. RBO compares two lists (in this case, the parameters $w\_i, i=1,\ldots , 5$ (Equation~\ref{equation:synthetic.data}), and the top 1000 ranked variable importances from the \cursedforest model). It is a mesaure of similarity between rankings that handles lists of different lengths, weighs high ranks more heavily than low, and is monotonic with increasing depth of evaluation. We expect to retrieve the 5 features in order of the weight that was assigned to them and with a higher variable importance measure than the noise variables. RBO provides a quantiative measure of how close we come to this on a scale from 0 (no feature recovered) to 1 (fully recovered). See supplementary information (S\*) for more details.

We are running Apache Spark 1.6.1 on a YARN cluster with 12 worker nodes each with 16 Intel Xeon E5-2660@2.20GHz CPU cores and 128 GB of RAM.

As shown in Fig~\ref{figure:out-of-bag-prediction-error-prod.png} the default value for \mtry (number of variables selected at each node) does not result in a good classification performance for this large feature dataset. Note that the plot shows the proportion, $n/p,$ for \mtry with the default being $\sqrt{p} \approx 1581$ ($n/p=0.0006$). The OOB for this value of \mtry does not drop below 0.5, even when the number of trees are increased (\ntree). Increasing \mtry in combination with \ntree yields the best performance with the OOB error essentially constant around 0.4 across a large range of of \mtry and \ntree values. This is in contrast to the feature-selection performance, where the RBO measure heavily depends on \ntree and gives better results with lower values of \mtry.

This may be because a large \mtry leads to more correlated trees as the same important features have a higher chance of being selected in all, not yielding good performance outcomes. This issue is less pronounced for classification error where random features can mimic the response variable, hence resulting more in a performance plateau. Increasing the number of trees on the other hand improves performance especially when the trees are kept diverse (small \mtry) but appropriate for large feature data sets (\mtry larger than default). \cursedforest's feature of parallelizing trees at the node level hence caters perfectly to the requirement of large feature data sets as more trees can be built given a fixed time budget.

From the synthetic data we can conclude that \cursedforest is able to perform accurate feature selection as well as accurate classification for large numbers of samples with large feature vectors. We can also conclude that fine tuning the model to accurately classify the dataset does not ensure good feature selection performance as there seems to be little correspondence between the OOB and RBO measure.

**Scalability analysis**

In this section we explore the performance of \cursedforest in more detail by testing its ability to scale to different sizes of data and computational resources.

In order to asses these characteristics, we run \cursedforest classification on synthetic data sets with varying numbers of variables (features) and samples, similar to the dataset used in \cite{Wright.and.Ziegle.2016} to evaluate \ranger, allocating varying number of CPU cores to the \cursedforest and also varying the computational complexity of the random forests by using a range of \mtry values.

We investigate the different synthetic data sets generated for section \ref{section:synthetic\_data} and measured the time taken to build a random forest model of 100 trees. The results reported below are averages of 5 runs, and all the cases were executed with the same random seed, to improve the consistency of measurements.

First we look at \cursedforest horizontal scalability for a medium size dataset of 2.5 million variables and 5000 samples, by varying the \mtry fraction and the number of CPU cores allocated to the execution.

Regardless of the number of cores used, \cursedforest displays approximately linear dependency between the execution

time and \mtry (Fig~\ref{figure:synthetictiming.a}).

\cursedforest scales almost linearly with the number of CPU cores for medium values of \mtry fraction but for both lower

and higher values the performance degrades slightly (Fig~\ref{figure:synthetictiming.b}). In the latter case the likely

cause is communication overhead (with lower \mtry values the proportion of time for parallelizable computation to the

time for internode communication is lower) while in the latter case it is most likely caused by reaching the clusters

computational capacity.

Next we investigate \cursedforest scalability with regards to the size of data, by varying the number of variables and sample for a fixed \mtry fraction of 0.25 and execution of 128 CPU cores. The results are visualized in Fig~\ref{figure:synth2} below (please note log scale on the axes and the values on y axes are expressed as trees per hour). Generally, the number of trees per hour decreases with an increased number of variables and samples sizes. Some irregularities in the graph can be attributed to computation vs communication tradeoff. It is also worth noting thatkeeping the \mtry fraction constant results in higher \mtry values with the growing number of variables, and this is what drives the performance down rather that the increase of dataset size itself. To conclude \cursedforest is capable of processing 60 trees per hour on a data sets with 50 million variables and 10,000 samples, which is the size range for whole genome sequencing experiments of clinically relevant cohort sizes.

**DISCUSSION**

Using a novel parallelization approach enabled by Spark, we extended random forests to cope with extremely high dimensional data (large number of variables). We have demonstrated that for data sets with more variable ($p$) than samples ($n$), such as GWAS applications, \cursedforest can perform effective variable selection replicating previous findings and uncovering novel associations in a bone mineral density study. We are able to elevate the signal compared to previous approaches using logistic regression due to random forests also modelling interacting features.

In fact, \cursedforest's ability to almost exhaustively sample the combinatorial space by efficiently building large numbers of trees puts it closer than any other method to the theoretical boundary for accurate recovery of informative variables. Donoho and Tanner~\cite{Donoho.and.Tanner.2009} define this as the ``universal phase change'', which depends on underdetermination, $\delta = n/p$, and the sparsity, $\rho =k/n$ (where $k$ is the number of informative variables). By effectively operating in the difficult area of this space with $p >> n$ and small number of informative features, \cursedforest is pushing the practical limits of this paradigm.

To showcase the breadth of application scope, we also demonstrate that \cursedforest's implementation aids in classification problems such as predicting the ethnicity of the 1000 genome project. By comparing \cursedforest to other implementations (including those optimised for large data sets) we have demonstrated that runtime can be improved by 80\% allowing analysis in timeframes relevant for point-of-care diagnostics such as for "Patients like mine" scenarios thereby enabling evidence based self learning health care.

In summary, \cursedforest offers the broadly needed capability of performing sophisticated machine learning tasks on big, complex life science data. Specifically, the here demonstrated ability to uncover novel biological information from previously analysed GWAS studies showcases the enormous potential of re-analysing the GWAS catalogue given this unprecedented computational capacity to perform joint-loci association testing and identify the much-anticipated cumulative multi-gene disease contribution.

**METHODS**

**Data Sets**

**1000 Genomes Project**

We obtained the 1000 Genomes Project data as VCF files from their FTP site. Each VCF file contains the variants for every individual across one chromosome. We use the phase 3 dataset which contains 2,504 individuals with approximately 81 million variant positions (i.e. a matrix of $2,504 \times 81$ million). For compatibility with the majority of the algorithms we demonstrate, we convert the original variant representations to double representations. For example, we store the absence of a variant (0\textbar0) as 0, a heterozygous variant (1\textbar0 or 0\textbar1) as 1, and a homozygous variant (1\textbar1) as 2, so we have an ``additive'' model. However, this is not the only possible encoding \cite{Goldstein.et.al.2011}. Note that from the encodings listed in Table~\ref{table:encodings}, the additive encoding will present the most difficulty for a random forest model.

**Bone mineral density dataset**

We obtained a VCF file from the authors of \cite{Duncan.2011}, which contains variant data from 1,936 postmenopausal women. This dataset covers approximately 7 million variant positions. We have information about the bone mineral density (BMD) for each individual, where a low BMD is a major predisposing factor to fracture \cite{Duncan.2011}.

**Synthetic data**

Each dataset consists of $n$ samples and $p$ variables where $n << p$, and values for each variable are ordinal variables with three levels represented as numbers $\{0, 1, 2\}$ (which correspond to an additive effect encoding of genomic variation) randomly generated from a uniform distribution with equal probabilities.

**Parameter settings**

We consider the parameter settings for the random forest algorithm. We use the R notation from the \randomforest package \cite{Liaw.and.Weiner.2002} which incorporates the original Fortran code by Brieman and Cutler. We incorporatethe advice of \cite{Liaw.and.Weiner.2002}, which we have found mirrors our own experience:

\item \ntree{} -- the number of trees. The number of trees necessary for good performance grows with the number of predictors. \cite{Liaw.and.Weiner.2002} suggest that a high \ntree is necessary to get stable estimates of variable importance and proximity; however, even though the variable importance measures may vary from run to run, we note that it is possible for a random forest model to have a poorer fit and still have an accurate ranking of variable importance;

\item \mtry{} -- the number of variables considered at each split (if \mtry=$p$, we have a boosted decision tree model). If one has a very large number of variables but expects only very few to be “important”, using larger \mtry may give better performance;

\item the size and complexity of the individual trees is controlled in \randomforest by setting \texttt{nodesize}, the minimum size of terminal nodes. It is controlled in Spark ML by setting \texttt{maxDepth}, the maximum depth of each tree in the forest.\footnote{The Spark ML documentation \cite{Spark.2016} sets \texttt{maxDepth} to 4 in their classification example. This may give an estimate with a higher bias and, as such, may be a poor choice. See the discussion in \cite{Dietterich.2002} where it is shown that the action of a RF is complicated than first thought. They show that bagging may reduce bias as well as variance but give a better result for low bias learners. \cite{Dietterich.2002} notes that if the bootstrap replicate approximation were correct (i.e.~if the bootstrap sample came from an identical distribution to the data), then bagging would reduce variance without changing bias.}

**ACKNOWLEDGEMENTS**

**AUTHOR CONTRIBUTIONS**

AO and PS implemented the method, AO conducted the BMD and 1000genomes experiments while PS generated the scaling data, analysed by RD with input from NE. PL and ELD aided in the interpretatio nof the BMD data, while SL and JD contributed to the 1000genomes analysis. DCB visualised the BMD and 1000genomes results. All authors contributed to the manuscript. The authors would like to acknowledge CSIRO, specifically PS, for setting up and maintaining the in-house Spark cluster.

**COMPETING FINANCIAL INTERESTS**

The authors declare competing financial interests: details are available in the online version of the paper.