High dimensional data analysis

Cavan Reilly

October 16, 2019

Table of contents

Data mining

Data mining is the process of searching through large data sets for associations.

If one uses conventional statistical testing methods then one is guaranteed to find some sort of association if one searches hard enough.

However these associations frequently won't be found in another independent sample.

Many methods have been developed for this process, and the field of inquiry that seeks to develop and apply these tools is referred to as *machine learning*.

Data mining

We will consider 3 appraoches to high dimensional data analysis here:

- random forests
- logic regression
- multivariate adaptive regression splines

However there are many other algorithms that have the same purpose.

Hence our treatment is selective and far from comprehensive.

Random forests

Random forests generate a collection of trees.

The result from applying random forests is not a final, best tree, but is a guide to which explanatory variables have the greatest impact on the outcome.

Thus it tells us about variable importance rather than the exact nature of the relationship between the set of predictors and the trait under investigation.

Random forests

Here are the steps of the algorithm.

1. Randomly select about 1/3 of the data with replacement and call it the *out of bag data* (OOB data), call the rest of the data the *learning sample* (LS).
2. Use the LS data to fit a tree without any pruning. However, at each split only use a randomly chosen set of predictors (about 1/3 of the data).
3. Use the OOB data to determine the impurity at all terminal nodes, sum these and call this the *tree impurity*.
4. For each predictor used to construct the tree, permute the subject indicators, recompute the tree impurity, and find the difference between this tree impurity and the tree impurity from the previous step to get the variable importance for each predictor.

Random forests

One then repeats the whole process many times and at the end computes the average variable importance for each predictor.

By only selecting a subset of variables to use at each split the algorithm potentially avoids problems due to high levels of LD among SNPs.

By permuting each predictor used and recomputing the tree impurity the algorithm can assess if a predictor is important for the overall classification process in a nonparametric fashion.

In the R package randomForest, variable importance is reported as the average over all iterations divided by its standard error.

This package also reports the mean of the total node impurity based on splits involving each predictor.

Random forests in R

Here we again set up the Virco data set and examine how all of the mutations predict the difference in the fold change for NFV and IDV.

We need to get rid of missing data before using the `randomForest` function as it doesn't allow missing data.

```
> library(randomForest)
> Trait <- NFV.Fold - IDV.Fold
> VircoGeno <- data.frame(virco[,substr(names(virco),1,1)=="P"]!="-")
> Trait.c <- Trait[!is.na(Trait)]
> VircoGeno.c <- VircoGeno[!is.na(Trait),]
> RegRF <- randomForest(VircoGeno.c, Trait.c,
+   importance=TRUE)
```

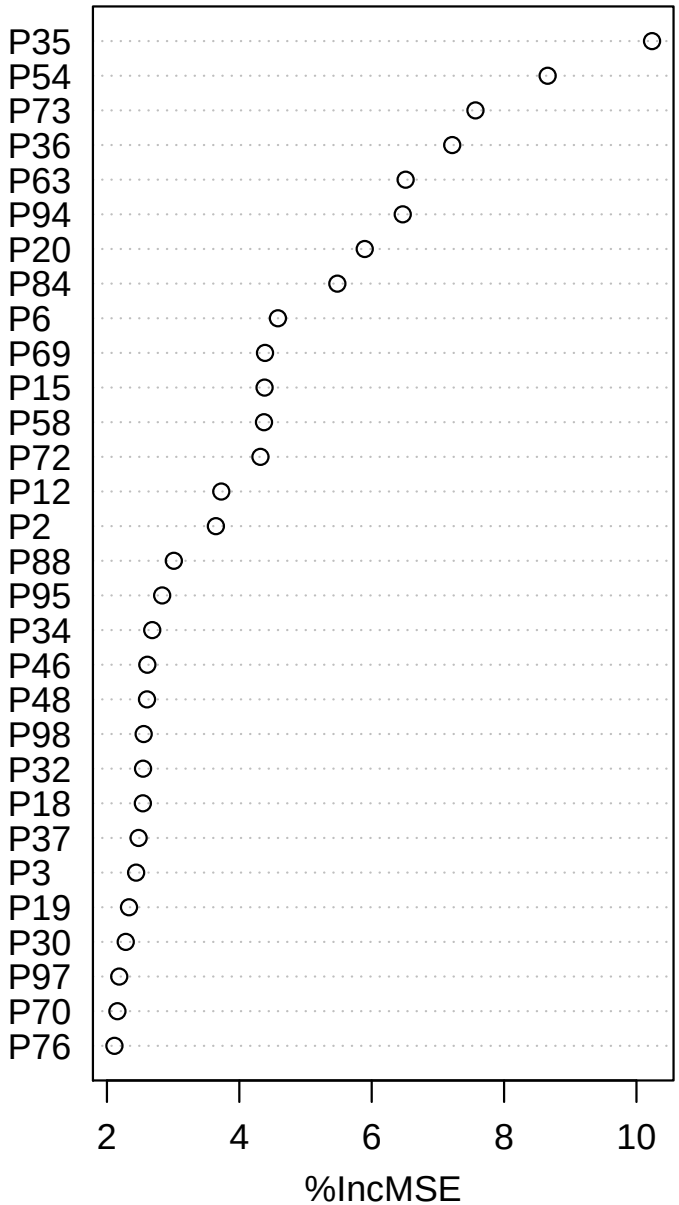
Random forests in R

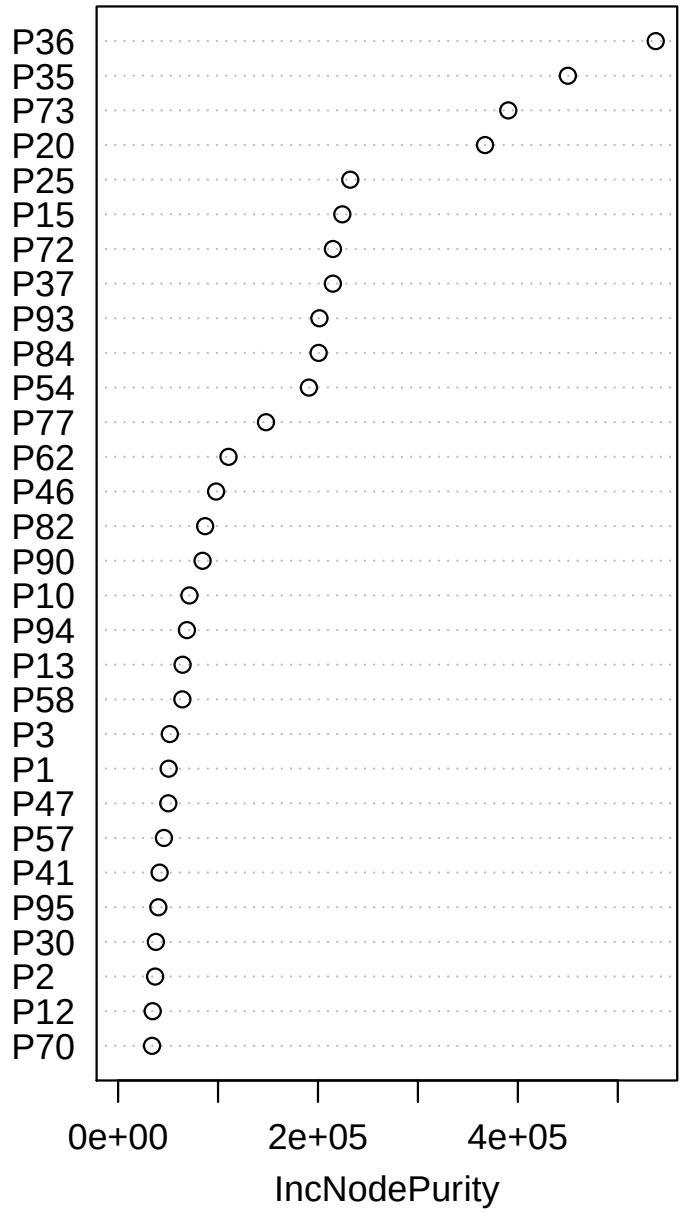
First we examine the output, then generate a plot to see which are the important variables.

```
> RegRF
Call:
 randomForest(x = VircoGeno.c, y = Trait.c, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 33
          Mean of squared residuals: 5547.21

                    % Var explained: 15.9
```

```
> varImpPlot(RegRF,main="")
```


P35
P54
P73
P36
P63
P94
P20
P84
P6
P69
P15
P58
P72
P12
P2
P88
P95
P34
P46
P48
P98
P32
P18
P37
P3
P19
P30
P97
P70
P76
2
4
6
8
10
%IncMSE


P36
P35
P73
P20
P25
P15
P72
P37
P93
P84
P54
P77
P62
P46
P82
P90
P10
P94
P13
P58
P3
P1
P47
P57
P41
P95
P30
P2
P12
P70
0e+00
2e+05
4e+05
IncNodePurity

Random forests in R

The plot indicates that the 5 most important variables are P35, P36, P54, P63 and P73.

Recall that the regression based approach identified P35, P46, P54, P58 and P73, so P36 and P63 are novel.

Not clear if any of these effects are statistically significant, and while one can inspect the individual trees using `getTree` it is not clear how one would use that information.

Missing data and random forests

It is not uncommon to have missing genotype data, especially in studies examining many variants.

In our example we had to dispose of 90 observations due to missingness, which was almost 10% of the sample.

A simple fix for this is to simply substitute the most commonly observed SNP for missing values.

This should be conducted within ethnic groups as its validity relies on Hardy Weinberg equilibrium.

Missing data and random forests in R

The `randomForest` package has functions that allow one to do this.

```
> Trait <- NDRM.CH[Race=="Caucasian" & !is.na(Race) & !is.na(NDRM.CH)]
> NamesAkt1Snps <- names(fms)[substr(names(fms),1,4)=="akt1"]
> FMSgeno <- fms[,is.element(names(fms),NamesAkt1Snps)]
+   [Race=="Caucasian" & !is.na(Race) & !is.na(NDRM.CH),]
> dim(FMSgeno)
```

```
[1] 777 24
```

So we have 777 subjects out of 1154 Caucasians under consideration at this point.

Missing data and random forests in R

Let's next examine the proportion of missing data for our SNPs.

```
> round(apply(is.na(FMSgeno),2,sum)/dim(FMSgeno)[1],3)
        akt1_t22932c         akt1_g15129a         akt1_g14803t
               0.069                0.067                0.021
akt1_c10744t_c12886t akt1_t10726c_t12868c akt1_t10598a_t12740a
               0.071                0.075                0.071
 akt1_c9756a_c11898t          akt1_t8407g          akt1_a7699g
               0.066                0.069                0.067
  akt1_c6148t_c8290t   akt1_c6024t_c8166t   akt1_c5854t_c7996t
               0.021                0.066                0.021
   akt1_c832g_c3359g          akt1_g288c   akt1_g1780a_g363a
               0.021                0.021                0.021
   akt1_g2347t_g205t    akt1_g2375a_g233a         akt1_g4362c
               0.066                0.066                0.069
        akt1_c15676t         akt1_a15756t         akt1_g20703a
               0.021                0.021                0.021
        akt1_g22187a         akt1_a22889g         akt1_g23477a
               0.071                0.021                0.021
```

Missing data and random forests in R

We then use the single imputation approach. Here are genotypes at one SNP before and after imputation.

```
> FMSgenoRough <- na.roughfix(FMSgeno)
> table(FMSgeno$"akt1_t22932c")
 CC  TC  TT
  3  55 665
> table(FMSgenoRough$"akt1_t22932c")
 CC  TC  TT
  3  55 719
```

Now use the `randomForest` function on the imputed data set.

```
> RandForRough <- randomForest(FMSgenoRough,Trait,
+   importance=TRUE)
```

Missing data and random forests in R

Here is the output that we previously viewed graphically.

```
> RandForRough$"importance"[order(RandForRough$"importance"[,1],
+   decreasing=TRUE),]
                          %IncMSE IncNodePurity
akt1_t10726c_t12868c 234.365556     26555.977
akt1_t8407g          197.084466     15232.624
akt1_g288c           130.835952     23870.986
akt1_g14803t         122.216074     19274.661
akt1_g15129a         118.016707     14555.595
akt1_c6148t_c8290t   111.196225     16282.021
akt1_a15756t         109.983082     23851.596
akt1_a22889g          99.801940     26576.974
akt1_c832g_c3359g     95.813145     11138.666
akt1_t10598a_t12740a  94.835043     17957.746
akt1_g2347t_g205t     85.283846     17545.905
akt1_g22187a          82.328928     22391.776
akt1_c6024t_c8166t    63.121531      8805.539
akt1_a7699g           62.914052      8162.597
akt1_g23477a          54.096923     19348.334
akt1_c9756a_c11898t   49.616864      8575.560
akt1_g2375a_g233a     49.140841     17405.400
```

Missing data and random forests in R

akt1_c5854t_c7996t	49.026789	19192.551
akt1_c15676t	41.848416	20593.258
akt1_g4362c	33.724391	7866.557
akt1_c10744t_c12886t	5.698555	4253.520
akt1_g1780a_g363a	5.291269	2947.077
akt1_g20703a	-1.958886	28478.770
akt1_t22932c	-5.089253	16526.413

Missing data and random forests

There are more sophisticated approaches available than single imputation.

The *proximity score* between 2 individuals is the proportion of trees for which those 2 subjects end up in the same terminal node, we denote this p_{ij} for subjects i and j.

A better imputation algorithm starts by fitting a random forest using single imputation.

The proximity scores are then calculated.

Missing data and random forests

Then, if genotype data was missing for subject i, for each possible genotype, we compute the average proximity score of subjects with that genotype to subject i.

We then assign the genotype with the highest average proximity score.

Then repeat fitting random forests and imputing.

After a while fit the final random forest.

It's not clear if this process converges to a final answer or how many times one should repeat the process.

Missing data and random forests in R

We return to the previous example and use some things we've already set up.

```
> FMSgenoMI <- rfImpute(FMSgeno, Trait)
     |      Out-of-bag   |
Tree |      MSE  %Var(y) |
 300 |     1260   111.05 |
     |      Out-of-bag   |
Tree |      MSE  %Var(y) |
 300 |     1241   109.40 |
     |      Out-of-bag   |
Tree |      MSE  %Var(y) |
 300 |     1257   110.82 |
     |      Out-of-bag   |
Tree |      MSE  %Var(y) |
 300 |     1253   110.46 |
     |      Out-of-bag   |
Tree |      MSE  %Var(y) |
 300 |     1241   109.39 |
> RandForFinal <- randomForest(FMSgenoMI[,-1], Trait, importance=TRUE)
```

Missing data and random forests in R

The results have the same top SNP, but with a higher level of importance.

```
> RandForFinal$"importance"[order(RandForFinal$"importance"[,1],
+   decreasing=TRUE),]
                          %IncMSE IncNodePurity
akt1_t10726c_t12868c 275.972895     26379.565
akt1_t8407g          215.239603     18653.158
akt1_g288c           125.970201     23006.316
akt1_g14803t         118.484596     18590.184
akt1_g15129a         112.634668     14430.548
akt1_c6148t_c8290t   110.098681     14128.753
akt1_a15756t         108.350798     25081.994
akt1_a22889g         102.888684     27519.912
akt1_g2347t_g205t     99.874095     18860.208
akt1_t10598a_t12740a  99.668040     15841.751
akt1_g22187a          87.859637     23192.396
akt1_c832g_c3359g     86.295125     11718.348
akt1_a7699g           80.272263      9685.794
akt1_c6024t_c8166t    71.193225      8303.793
akt1_g2375a_g233a     56.545555     19601.788
akt1_g23477a          54.657235     19967.984
```

Missing data and random forests in R

akt1_c5854t_c7996t	53.464080	19031.417
akt1_c9756a_c11898t	50.022642	8412.683
akt1_c15676t	42.705935	22458.086
akt1_g4362c	37.384479	9252.566
akt1_c10744t_c12886t	13.630497	9215.804
akt1_t22932c	6.783107	19136.158
akt1_g1780a_g363a	6.335833	2903.492
akt1_g20703a	5.588045	31654.745

Haplotype importance

We also may be interested in assessing the impact of haplotypes on the trait.

As we've already discussed how the EM algorithm can be used to compute the posterior probability of subjects having various haplotypes, we can use those ideas to generate haplotypes that we then supply to a random forest.

There is a package called mirf that has functions that allow on to use multiple imputation to sample haplotypes, compute variable importance measures via random forests, then average over the multiply imputed data sets.

But as of the time of the drafting of these notes the package doesn't appear to be supported.

Logic regression

In logic regression we model the mean of the trait as depending on a sequence of Boolean expressions multiplied by parameters that indicate the effect of the expression on the mean.

A Boolean expression is a sequence of "and"s, "or"s and "not"s that results in a binary outcome.

For example, consider 4 variants that are binary and labeled v_1, v_2,v_3 and v_4, and suppose that the most common variant is coded as 0 with the less common variant coded 1.

Then ($v_1 = 0$ and $v_2 = 0$) or ($v_3 = 1$ and not $v_4 = 1$) describes a complicated multilocus haplotype.

Associated with this haplotype is a parameter that gives the effect of this haplotype on the mean.

Logic regression

Except for cases where there are a very small number of variants under investigation, the number of possible expressions is too large to explicitly consider each.

Hence in practice, researchers have used either a greedy search algorithm or simulated annealing.

A greedy search algorithm is an algorithm that always tries to increase the value it is trying to optimize.

Simulated annealing is a very general algorithm that sometimes accepts losing progress towards the goal of optimizing something.

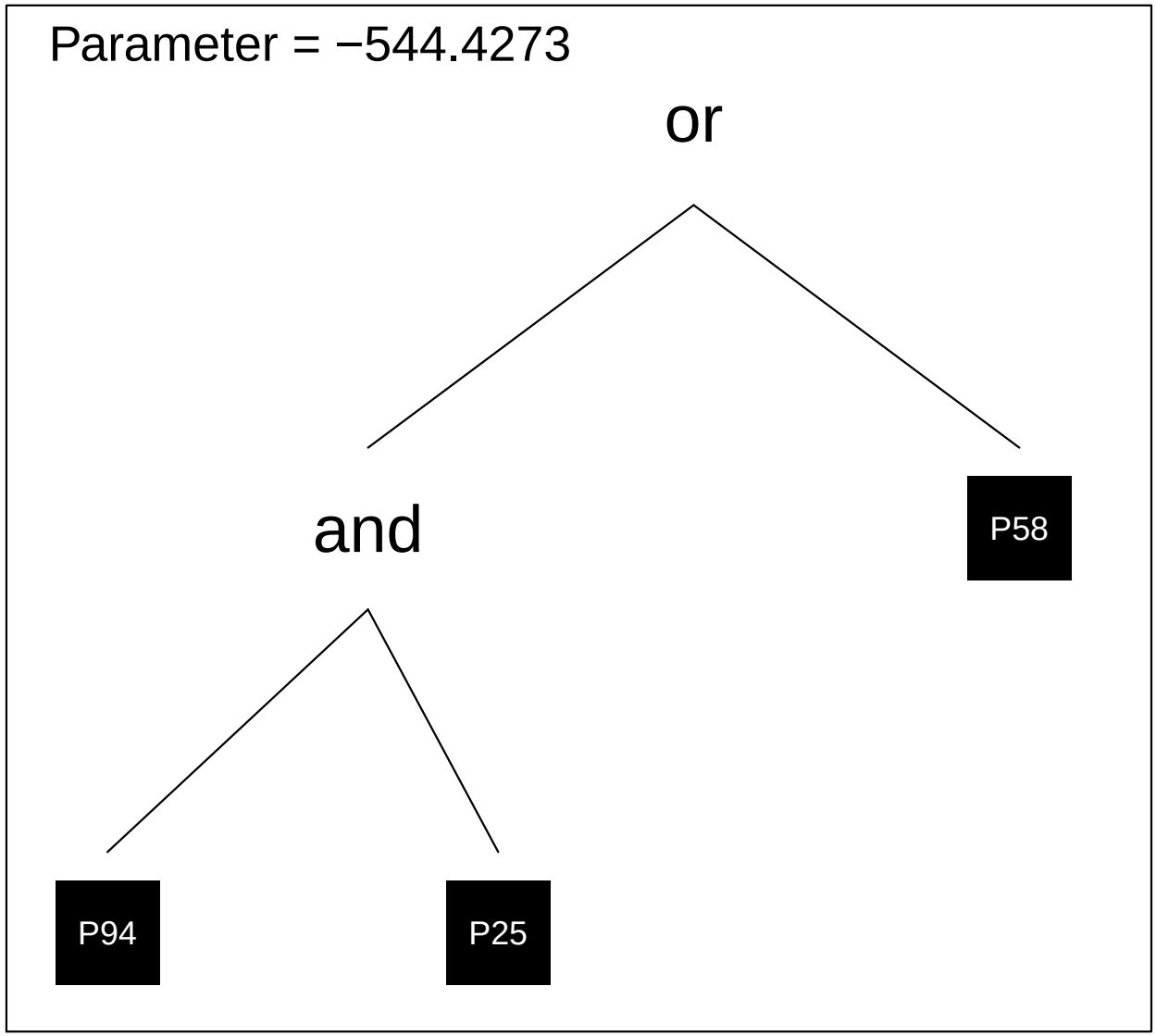
Due to this property, simulated annealing can avoid getting stuck at a local optimum unlike greedy search algorithms.

Logic regression in R

First we set up the data and remove missing data, then we run a logic regression and create a plot.

```
> library(LogicReg)
> Trait <- NFV.Fold - IDV.Fold
> VircoGeno <- data.frame(virco[,substr(names(virco),1,1)=="P"]!="-")
> Trait.c <- Trait[!is.na(Trait)]
> VircoGeno.c <- VircoGeno[!is.na(Trait),]
> VircoLogicReg <- logreg(resp=Trait.c, bin=VircoGeno.c,
+   select=1)
```

tree 1 out of 1



Logic regression in R

We can also examine the output.

```
> VircoLogicReg
score 77.475
548 -544 * (((not P94) and (not P25)) or (not P58))
```

Note that running the algorithm multiple times gives different answers as it uses a randomized search algorithm (simulated annealing).

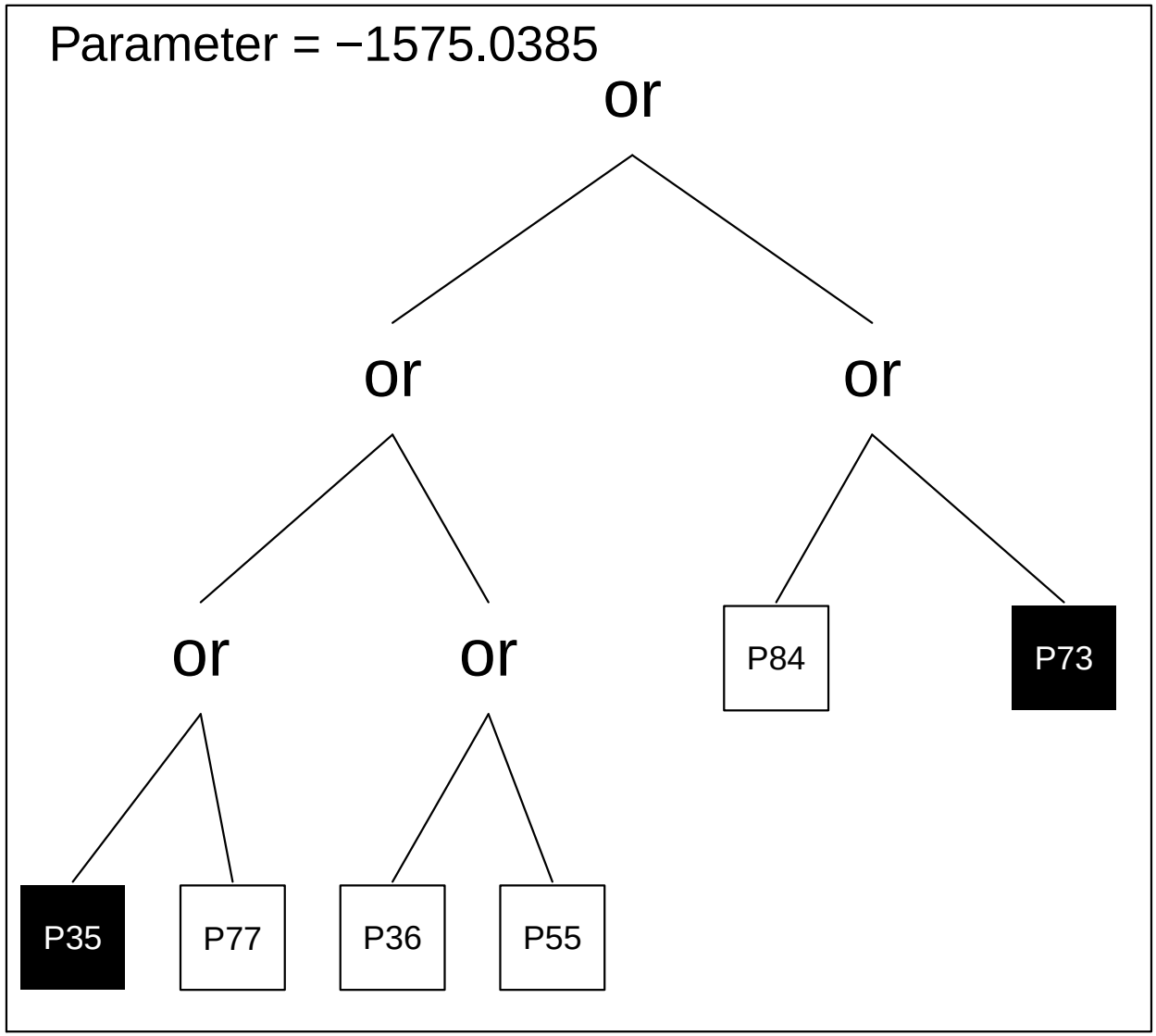
This implies that we are not using enough iterations, check the help file for `logreg` for extensive treatment of monitoring the progress of the algorithm.

Logic regression in R

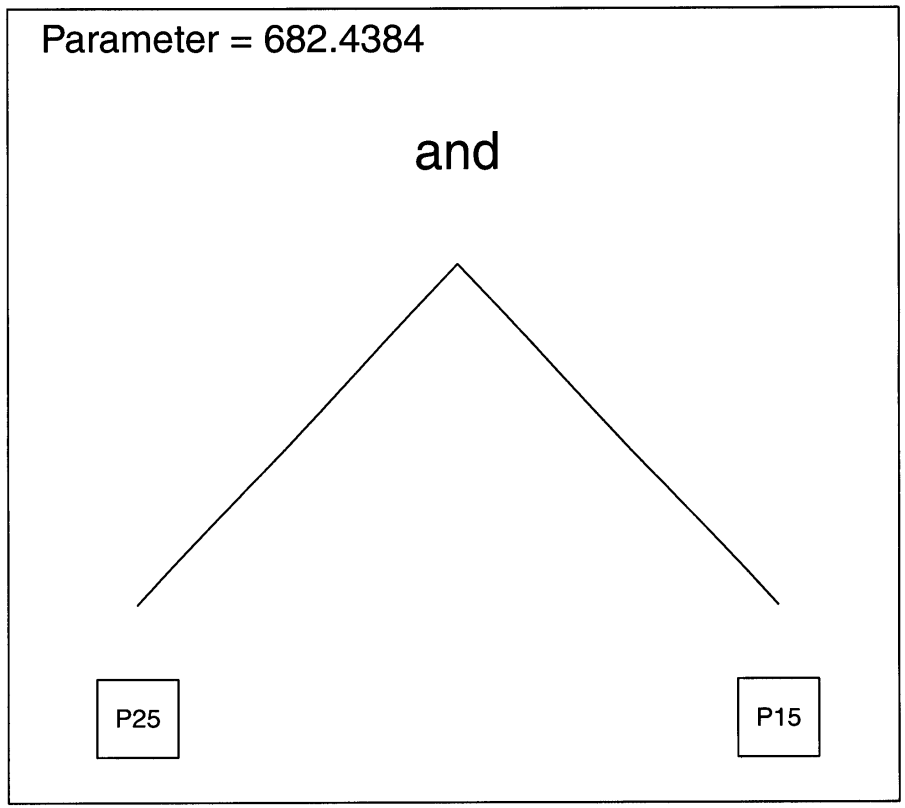
We can also specify multiple trees in the model as follows.

```
> VircoLogicRegMult <- logreg(resp=Trait.c, bin=VircoGeno.c, select=2,
+   ntrees=2, nleaves=8)
The number of trees in these models is 2
The model size is 8
The best model with 2 trees of size 8 has a score of 32.4689
> plot(VircoLogicRegMult)
```

tree 1 out of 2 total size is 8



tree 2 out of 2 total size is 8



Logic regression in R

Here is the description of the model.

```
> VircoLogicRegMult
2 trees with 8 leaves:  score is 32.469
1580 -1580 * ((((not P35) or P77) or (P36 or P55)) or (P84 or (not P73)))

+682 * (P25 and P15)
```

There is also an MCMC based implementation that samples trees.

This method determines the importance of a variable by determining how frequently it is included in the model.

Logic regression in R

We again use the virco data set but now determine which mutations in the protease region predict Saquinavir fold change.

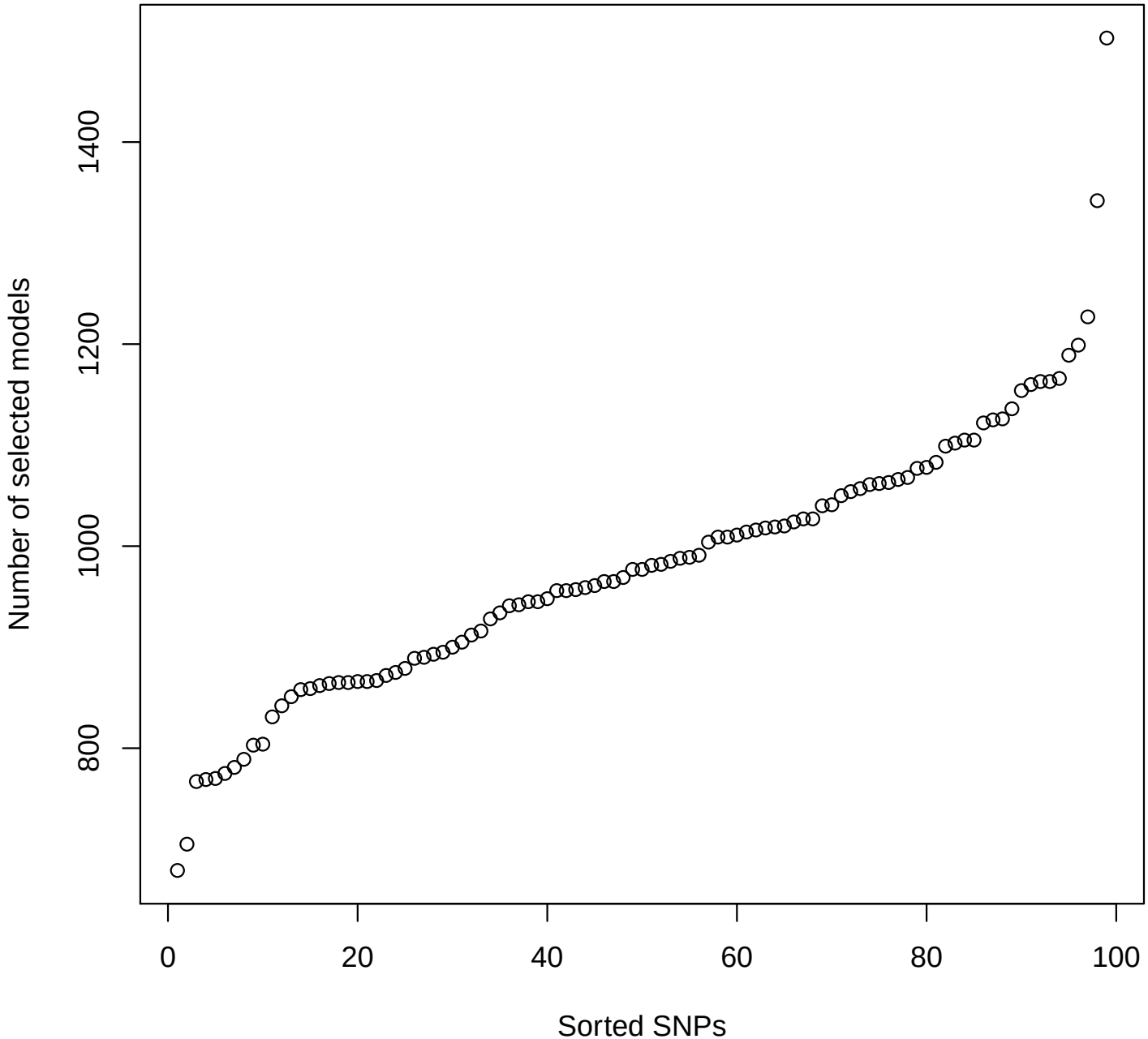
```
> Trait <- SQV.Fold
> VircoGeno <- data.frame(virco[,substr(names(virco),1,1)=="P"]!="-")
> Trait.c <- Trait[!is.na(Trait)]
> VircoGeno.c <- VircoGeno[!is.na(Trait),]
> VircoLogicRegMCMC <- logreg(resp=Trait.c,
+   bin=VircoGeno.c, select=7)
```

This can take a while.

Logic regression in R

We then plot the output to examine which locations are important.

```
plot(sort(VircoLogicRegMCMC$single), xlab="Sorted SNPs",
+   ylab="Number of selected models")
```


Number of selected models
800
1000
1200
1400
0
20
40
60
80
100
Sorted SNPs

Logic regression in R

To determine the identity of the top 5 scoring loci, we print the names of the mutations from least to most important.

```
> names(VircoGeno)[order(VircoLogicRegMCMC$single)]
 [1] "P28" "P98" "P8"  "P5"  "P99" "P27" "P22" "P70" "P49" "P83" "P39" "P63"
[13] "P64" "P52" "P56" "P42" "P38" "P40" "P72" "P9"  "P55" "P89" "P32" "P51"
[25] "P76" "P88" "P23" "P14" "P86" "P97" "P34" "P96" "P87" "P67" "P44" "P79"
[37] "P24" "P10" "P59" "P68" "P46" "P92" "P11" "P95" "P77" "P20" "P71" "P65"
[49] "P7"  "P43" "P36" "P29" "P15" "P73" "P62" "P30" "P69" "P19" "P91" "P3"
[61] "P81" "P57" "P6"  "P66" "P35" "P1"  "P26" "P90" "P31" "P80" "P60" "P47"
[73] "P58" "P2"  "P93" "P78" "P50" "P82" "P21" "P84" "P37" "P53" "P12" "P13"
[85] "P75" "P61" "P33" "P16" "P17" "P25" "P41" "P18" "P45" "P4"  "P85" "P74"
[97] "P94" "P54" "P48"
```

So the top scoring loci are P48, P54, P74, P85 and P94.

Logic regression in R

However there is extensive variability-when you rerun there will be a different set of top scoring mutations.

To use this in practice one needs to increase the number of iterations using a command like the following prior to running this.

```
mymccontrol <- logreg.mc.control(nburn = 250000,
niter = 500000)
```

then one would issue the command

```
> VircoLogicRegMCMC <- logreg(resp=Trait.c,
+   bin=VircoGeno.c, select=7, mc.control=mymccontrol)
```

Logic regression in R

This would run a half million iterations rather than the default of 25000 which evidently is not enough for this problem.

It would take a long time to run this job, but that's not uncommon when using MCMC.

The best way to determine if one has run enough iterations is to run the job twice and see if the results agree to within the desired level of precision.

For a problem like this that means we should choose enough iterations so that the top few most important loci are the same for 2 different runs.

Multivariate adaptive regression splines

Multivariate adaptive regression splines (MARS) is a method, similar to CART, for building models that explain the variation in a trait.

The primary difference is that MARS seeks to build additive models based on the set of predictor variables and their statistical interactions with one another.

The algorithm does this recursively, where at each stage, it incorporates the covariate or pairwise interaction among covariates, that is the best predictor.

After it finishes building up the model, it does some backward elimination to get rid of terms which it deems to be overfitting.

Multivariate adaptive regression splines in R

It is easy to implement using the earth package in R.

Here we install and load the package then set up some data from the virco data set to conduct the analysis.

We create binary predictors as this is what the algorithm is designed for, but we can encode SNP data using multiple indicator variables.

```
> install.packages("earth")
> library(earth)
> Trait <- NFV.Fold - IDV.Fold
> VircoGeno <- data.frame(virco[,substr(names(virco),1,1)=="P"]!="-")
> Trait.c <- Trait[!is.na(Trait)]
> VircoGeno.c <- VircoGeno[!is.na(Trait),]
```

Multivariate adaptive regression splines in R

Then we use the earth function with degree set to 2 to look for 2-way interactions

```
VircoMARS <- earth(Trait.c~., data=VircoGeno.c,
degree=2)
> summary(VircoMARS) Call:  earth(formula=Trait.c~.,
data=VircoGeno.c, degree=2)
                    coefficients
(Intercept)            -1.23166
P25TRUE               748.10125
P35TRUE                37.86730
P76TRUE               -32.35013
P1TRUE * P25TRUE    -784.98992
P1TRUE * P73TRUE      -31.73965
P10TRUE * P35TRUE      28.78594
P10TRUE * P73TRUE      71.69144
P15TRUE * P35TRUE     -32.09370
```

Multivariate adaptive regression splines in R

```
P15TRUE * P54TRUE     30.96957
P15TRUE * P73TRUE    -58.99329
P20TRUE * P35TRUE     48.89864
P20TRUE * P54TRUE    -38.47681
P20TRUE * P73TRUE     80.14481
P30TRUE * P73TRUE    254.30888
P30TRUE * P77TRUE     38.92757
P35TRUE * P36TRUE    -41.63952
...
Selected 40 of 100 terms, and 23 of 99 predictors
Importance: P25TRUE, P1TRUE, P35TRUE, P36TRUE, P73TRUE, P54TRUE, P94TRUE, ...
Number of terms at each degree of interaction: 1 3 36

GCV 5173.102    RSS 4081272    GRSq 0.2173564    RSq 0.3660587
```

So the algorithm originally selected 40 terms and 23 remain after pruning.

Multivariate adaptive regression splines in R

We can see the order of importance of these terms as follows.

```
> evimp(VircoMARS)
                  nsubsets   gcv    rss
P25TRUE                 39 100.0  100.0
P1TRUE                  38  96.3   97.1
P35TRUE                 36  81.8   87.4
P36TRUE                 36  81.8   87.4
P73TRUE                 36  81.8   87.4
P54TRUE                 34  72.3   81.0
P94TRUE                 34  72.3   81.0
P10TRUE                 34  71.8   80.6
P84TRUE                 34  71.8   80.6
P77TRUE                 33  70.0   79.0
P72TRUE                 31  61.9   73.3
...
P85TRUE-unused           6   9.9   26.1
P65TRUE-unused           2   3.6   14.6
```

Multivariate adaptive regression splines in R

Note that each of these machine learning approaches identifies different subsets of mutations, although some sites are frequently seen.