# SNP Ranking by Function R package (SuRFR)

Niamh M. Ryan and Stewart W. Morris

June 2014

## 1 Introduction

Genomics projects such as linkage and genome wide association (GWAS) studies have been extremely successful at identifying genomic regions that harbour genetic variants contributing to a phenotype of interest. Over 90% of disease-associated variants from GWAS fall within non-coding regions (Maurano et al., 2012). However, while there are several tools that identify putatively pathological coding variants, pinpointing causal non-coding variants is still a challenge. This is partly because the genomic signals that characterise functional ragulatory variants are not fully defined, and partly due to our incomplete understanding of regulatory architecture. However, the situation is improving.

A number of large scale projects are currently underway with the aim of identifying functional elements on a genome-wide scale. These resources (such as the Encyclopaedia of DNA Elements (ENCODE) project) are providing a wealth of genomic and epigenomic annotation data for functional predictions. However, there is a need for a systematic, scalable, benchmarked method to combine these datasets, along with other genomic functional measures, to prioritise candidate variants for follow-up analysis.

To help address this, we are introducing the SNP Ranking by Functionality R package (SuRFR), which we have designed to aid in the identification of functional variants from genomics studies. Variants are ranked in order of likelihood of functionality. This is achieved by ranking SNPs from least likely to indicate functionality through to most likely, for a variety of functional annotation categories. The individual ranks are then combined into a final prioritisation rank-of-ranks using a weighting system. Central to this approach is a well validated weighting model that apportions the relative importance of each annotation type (a vector of multipliers, one for each annotation data source).

The SuRFR package provides three weighting models: a general model for any analysis (ALL); a model designed specifically for prioritizing (rare) disease vari-

ants (DM); and a model for prioritising complex disease variants (DFP). In addition, SuRFR allows users to specify their own custom model.

This method has been tested extensively using a combination of synthetic and real datasets and ten-fold cross-validation. For more information on model training and testing, see our paper: SuRFRing the genomics wave: an R package for prioritising SNPs by functionality.

For more details on the content of the SuRFR package, see the User Manual.

## Annotation Categories

SuRFR incorporates information relating to a range of genomic and epigenomic annotation parameters known to correlate with regulatory elements and non-coding disease variants.These are:

Minor Allele Frequencies: from the 1000 Genomes phase-1 populations.

Conservation: Genomic Evolutionary Rate Profiling (GERP) estimates position-specific evolutionary rates and identifies candidate constrained elements (Cooper et al., 2005). Constraint is measured in terms of Rejection Substitutions scores based on a comparison of the number of observed versus expected substitutions. GERP scores for the entire genome were obtained from the UCSC Genome Browser's ftp site.

DNase hypersensitivity: Genome-wide DNase HS data assayed in 125 cell types generated by the University of Washington (Sabo et al, 2004) and Duke University (Song et al., 2010) ENCODE groups (wgEncodeRegDnaseClusteredV2).

DNase footprints: Deep sequencing DNase footprinting data from ENCODE project, generated by the University of Washington ENCODE group, defines regions of the genome that interact with and bind DNA binding proteins (Sabo et al., 2006).

Chromatin states: Ernst et al. (2011) systematically mapped nine chromatin marks to nine cell lines and used a multivariate hidden Markov model to distinguish different chromatin states through recognition of combinatorial patterns of chromatin marks. These data have been rigorously tested and confirmed by in vitro assays. Fifteen chromatin states are predicted, including active promoter, weak promoter, strong enhancer, transcriptional elongation, polycomb repressed, repetitive /copy-number-variant. Each of the nine cell lines are annotated as part of the SNP annotation.

Position: SNP annotations across a range of positional categories: whether they are located in exons; introns; splice sites; CpG islands; CpG shores; promoters (defined as being within 1kb of a transcription start site); 10kb upstream or downstream of a gene; or intragenic (beyond 10kb up or downstream of a gene). Genic data was sourced from both UCSC's gene annotation data and the FANTOM5 CAGE dataset (defining novel transcription start sites and therefore novel promoters).

Transcribed Enhancers: An atlas of active, transcribed enhancers across the majority of human tissues and cell types was produced using data from the FANTOM5 project (Andersson et al., 2014). These CAGE-defined enhancers were shown to be more likely to be validated by enhancer assays than predicted

enhancers identified using methods such as mapping of chromatin marks or DNase HS sites.

Transcription Factor Binding Sites: TFBSs identified using ChiP-seq experiments for 161 transcription factors across 91 cell types and predicted transcription factor binding motifs from the ENCODE Factorbook repository (wgEncodeRegTfbsClusteredV3) (Wang et al., 2012; Wang et al., 2013).

### Basic Input

SuRFR takes in an tab delimited text or bed file comprised of chromosome number, start and end coordinates (in hg19 format) for the SNPs to be investigated (no header row). It can also take in a pre-compiled tab delimited text file with the required annotation categories for each SNP and run in straight through the prioritisation function.

## 2 Getting Started

SuRFR is dependent on an additional annotation package, SAILR (SNP Annotation Information List R package), which provides pre-compiled annotation tables for all the SNPs from the 1000 Genomes project for each of the four main populations: AFR, AMR, ASN and EUR (See the 1000 Genomes Browser for more information on their population databases).

SuRFR also provides a function to build annotation tables for any variants across the genome. To do this it uses bedtools, which is unix specific. If you wish to annotate variants not present in the 1000 Genomes database you must therefore work in a unix/linux/mac environment.

However, if you only wish to annotate known variants, SuRFR also works in the Windows envirmonment.

To install SuRFR you must first have installed SAILR:

### Intallation

#### 2.0.1 Unix/Linux/MacOS:

> R CMD INSTALL SAILR_some_version.tar.gz
> R CMD INSTALL SuRFR_some_version.tar.gz

#### 2.0.2 Windows:

In your RGui application:

1. Go to Packages: Install package(s) from local zip files
2. Select the downloaded package (example: SAILR_some_version.tar.gz)
3. Select the downloaded package (example: SuRFR_versionx.tar.gz)

**For all platforms:**

Once you have installed the packages, you can then use the following command in the R console:

```
> library(SuRFR)
```

# 3 Reading Data

SuRFR builds a table of SNP annotation data in conjunction with its sister annotation package, SAILR, which comprises pre-compiled annotation files for all variants from the 1000 Genomes project for each of the four main populations (AFR, AMR, ASN, EUR) from which users can extract a subset of SNPs of interest.

SNP lists containing variants not catalogued by the 1000 Genomes project can be annotated using a second function Denovo_anno_table, which uses tabix to extract data from a remote source and bedtools to build the functional table.

SuRFR can also take in a premade annotation table, with the correct column headers, as a tab-delimited text file.

## 3.1 Accessing SNP annotation data from SAILR

This function takes in a basic bed file (chr, start, end: no header row) and uses tabix to select pre-compiled SNP annotation data from SAILR to builds a functional annotation dataframe. The dataframe is also exported and saved as a text file so it can be run through the analysis step at a later time.

The user must provide:

1. The filename of a tab delimited file containing chromosome number, start coordinates and end coordinates (hg19 assembly) for each SNP (this file must NOT contain a header row).
2. The name of a population from which to pull the data of interest.

We use a 20 line example bedfile, "test.data.bed" to show how SuRFR works from start to finish.

We must first load the example bedfile and then run it through the SuRFR annotation function:

```
> bedfile <- system.file("extdata", "test.data.bed", package = "SuRFR")
> annotation_table <- SuRFR_annotation(bedfile, pop = "EUR")

Input file:        /usr/lib64/R/library/SuRFR/extdata/test.data.bed
Population:        EUR

> # Lets have a look at the output:
> annotation_table[1:5,]
```

```
  chr   start     end         RS refbase altbase DAF.G1K GeneUnique     Exon
1 chr1 1853587 1853588  rs2144686       C       T 0.04354   C1orf222 C1orf222
2 chr1 1946590 1946591 rs13303016       G       A 0.82982   AK054708     <NA>
3 chr1 1946808 1946809  rs4648752       C       T 0.82850   AK054708 AK054708
4 chr1 1981117 1981118  rs3795277       A       C 0.06332       <NA>     <NA>
5 chr1 2513104 2513105   rs735000       C       T 0.15303       <NA>     <NA>
  Nearest.exon Nearest.centro.gene Nearest.telo.gene   Promoter  CpG CpG_shore
1     C1orf222              TMEM52            CALML6       <NA> <NA>      <NA>
2     AK054708               GABRD              <NA>       <NA> <NA>      <NA>
3     AK054708               GABRD              <NA>       <NA> <NA>      <NA>
4         <NA>               PRKCZ              <NA> uc001aiq.3 <NA>  CpG: 195
5         <NA>             FAM213B             MMEL1       <NA> <NA>      <NA>
  wgEncodeRegDnaseClustered DNase_footprint wgEncodeBroadHmmGm12878HMM
1                        NA               0          13_Heterochrom/lo
2                      1000               5               12_Repressed
3                      1000               0               12_Repressed
4                      1000               2            2_Weak_Promoter
5                       447               1          5_Strong_Enhancer
  wgEncodeBroadHmmH1hescHMM wgEncodeBroadHmmHepg2HMM wgEncodeBroadHmmHmecHMM
1              11_Weak_Txn          2_Weak_Promoter       13_Heterochrom/lo
2              11_Weak_Txn             12_Repressed         2_Weak_Promoter
3              11_Weak_Txn             12_Repressed         2_Weak_Promoter
4          2_Weak_Promoter        4_Strong_Enhancer         2_Weak_Promoter
5        13_Heterochrom/lo              11_Weak_Txn       13_Heterochrom/lo
  wgEncodeBroadHmmHsmmHMM wgEncodeBroadHmmHuvecHMM wgEncodeBroadHmmK562HMM
1      13_Heterochrom/lo        13_Heterochrom/lo            12_Repressed
2           12_Repressed          6_Weak_Enhancer            12_Repressed
3           12_Repressed          6_Weak_Enhancer            12_Repressed
4           12_Repressed             12_Repressed             8_Insulator
5      13_Heterochrom/lo        4_Strong_Enhancer            12_Repressed
  wgEncodeBroadHmmNhekHMM wgEncodeBroadHmmNhlfHMM GERP.UCSC Splice
1      13_Heterochrom/lo       13_Heterochrom/lo     1.320     NA
2           12_Repressed            12_Repressed     1.300     NA
3           12_Repressed            12_Repressed     0.542     NA
4        2_Weak_Promoter            12_Repressed    -3.000     NA
5           12_Repressed            12_Repressed    -0.636     NA
                          FANTOM5_promoter_peaks FANTOM5_promoter_genes
1                                           <NA>                   <NA>
2                                           <NA>                   <NA>
3                                           <NA>                   <NA>
4 chr1:1981834..1981887,+,chr1:1981890..1981945,+                  PRKCZ
5                                           <NA>                   <NA>
                                                        FANTOM510kb_peaks
1                                                                    <NA>
2 chr1:1950743..1950748,+,chr1:1950763..1950815,+,chr1:1950820..1950853,+
3 chr1:1950743..1950748,+,chr1:1950763..1950815,+,chr1:1950820..1950853,+
```

```
4                         chr1:1981834..1981887,+,chr1:1981890..1981945,+
5 chr1:2517953..2517968,+,chr1:2518202..2518291,+,chr1:2518488..2518555,+
  FANTOM5_10kb_genes nearest_TSS nearest_FANTOM_TSS Downstream_dist
1               <NA>           0              -2870               0
2              GABRD           0              -4155               0
3              GABRD           0              -3937               0
4              PRKCZ        -791               -732           18926
5            FAM213B       -4794              -4857            8976
  Enhancer_Atlas TFBSs                   Pos
1             NA    NA chr1:1853587-1853588
2             NA   440 chr1:1946590-1946591
3             NA    NA chr1:1946808-1946809
4             NA  1000 chr1:1981117-1981118
5             NA   234 chr1:2513104-2513105
```

## 3.2 Unique variants

If your data includes variants that are not present in the 1000 Genomes database, we have a second function that uses a unix shell script (rebuild) to build functional tables from scratch.

All you need, again, is a tab delimited text file (no header) with chr, start and end coordinates for each SNP of interest and provide a file name (here we are providing a test file, so need to use system.file to find it).

This command also allows you to split your job over multiple threads and parallelise the operation. If you have a large dataset, this will significantly speed up the running time.

```
> file <- system.file("extdata","test.data.bed", package="SuRFR")
```

Then, on the command line type:

```
> ann_table2 <- Denovo_anno_table(file, pop="EUR", threads=1)
> ann_table2[1,]
```

```
   chr   start     end        RS refbase altbase DAF.G1K GeneUnique     Exon
1 chr1 1853587 1853588 rs2144686       C       T 0.04354   C1orf222 C1orf222
  Nearest.exon Nearest.centro.gene Nearest.telo.gene Promoter  CpG CpG_shore
1     C1orf222              TMEM52            CALML6     <NA> <NA>      <NA>
  wgEncodeRegDnaseClustered DNase_footprint wgEncodeBroadHmmGm12878HMM
1                        NA               0          13_Heterochrom/lo
  wgEncodeBroadHmmH1hescHMM wgEncodeBroadHmmHepg2HMM wgEncodeBroadHmmHmecHMM
1               11_Weak_Txn          2_Weak_Promoter       13_Heterochrom/lo
  wgEncodeBroadHmmHsmmHMM wgEncodeBroadHmmHuvecHMM wgEncodeBroadHmmK562HMM
1       13_Heterochrom/lo        13_Heterochrom/lo            12_Repressed
  wgEncodeBroadHmmNhekHMM wgEncodeBroadHmmNhlfHMM GERP.UCSC Splice
1       13_Heterochrom/lo       13_Heterochrom/lo      1.32     NA
  FANTOM5_promoter_peaks FANTOM5_promoter_genes FANTOM510kb_peaks
```

```
1                   <NA>                    <NA>              <NA>
  FANTOM5_10kb_genes nearest_TSS nearest_FANTOM_TSS Downstream_dist
1               <NA>           0              -2870               0
  Enhancer_Atlas TFBSs                  Pos
1             NA    NA chr1:1853587-1853588
```

"file" is your input file

"EUR" is the population you want to get allele frequencies for and can be set to EUR, AMR, AFR or ASN.

# 4 SNP Ranking

SuRFR contains two versions of the ranking and weighting function: one designed to take in a data frame from the annotation step (Reading Data section) and one to read in a pre-compiled tab delimited text file containing the annotation table.

### Setting the weighting model

As explained in the Introduction, you can define your own custom model, changing the relative contribution of each annotation feature with respect to each other, as a vector of the format (MAF, Conservation, Chromatin States, DNase HSs,Position, DNase footprints, Enhancers, TFBSs). If you wish to leave out any of the annotation classes, set the weighting to "0". Similarly, if you want to reverse the order of the weighting of any annotation, set the weight to a negative value (ie if you want least conserved to rank above most conserved, changed position 2 to -x)

Three pre-defined weighting models are included in the R package: ALL, DM and DFP. Each of these models has been trained and tested using benchmarking datasets and ten-fold cross-validation. For more information on model assessment see our paper.

Pre-defined weighting models:

```
> ALL

[1]  0  1  4  0 13  1  0  2

> DM

[1] 11  3  5  0 15  1  0  4

> DFP

[1]  0  0  3  0 11  3  3  1
```

Setting custom model:

```
> custom1 <- c(4,0,1,10,3,0,1,1)
> custom1

[1]  4  0  1 10  3  0  1  1

> custom2 <- c(-2,-4,1,5,2,1,0,0)
```

**Setting the optimum MAF**

The optimum MAF can also be tuned. Default MAF is unique (1e06) as this annotation is very informative for disciminating rare disease variants from background variants and is central to the DM model's functionality.

However, if the causal variant for a particular disease was known to be likely to be present in the population with a MAF of 0.1, ie 10 percent, then the MAF can be changed accordingly.

```
> freq <- "10"
> freq

[1] "10"
```

**Setting the position and chromatin state ranks**

Two additional parameters that can be customised are the rank orders for the Position and Chromatin state annotation categories. Unless set by the user as function arguments, these variables run using their default rankings. These default ranks have been chosen using multivariable regression on a training dataset of known regulatory variants and background variants.We recommend using the default settings, however, if you wish to adjust them, edit their arguments.

chrom: ten position vector (promoters, strong enhancers, weak enhancers, repressed, insulators, transcription translation, transcription elongation, weak transcription, heterochromatin, unsure(repetitive/CNV and NA's))

pos_ranks: seven position vector (introns, CpG shores, CpG islands, 10kb upstream/downstream, promoters, splice sites, exons)

Both of these vectors rank SNPs on most important to least important (high to low numbers). For more details on these data, look at "Ernst_states" and "Position_ranks" in the User Manual.

Below is an example using all of these variables:

```
> custom <- c(4,0,1,10,3,5,1,1)
> out <- "bed.file.custom1.freq1"
> mydata <- SuRFR_analysis_data.frame(ClinVar_Test, out, model=custom, MAF=10,
```

```
+
+ chrom=c(10,9,6,7,8,5,4,3,2,1),pos_rank=c(1,2,3,3,3,4,5))
> mydata[1:2,]

                        Pos GERP_raw GERP.rank GERP_av_rank Conservation.rank
247 chrX:100662900-100662901        0         1  0.003597122       0.003597122
82   chr15:72978551-72978552        0         1  0.003597122       0.003597122
    DNaseHS_raw DNaseHS.rank DNaseHS.av.rank DNase_foot_raw DNase.foot.rank
247        1000          278               1             30             278
82         1000          278               1             13             274
    DNase.foot.av.rank Ernst_score Ernst.Av.new.rank Position.rank
247          1.0000000          10                 1             1
82           0.9856115          10                 1             1
    Position_score DAF_raw DAF.rank DAF.rank_normalised Enhancers_raw
247              5 0.06897      271           0.9748201             0
82               5 0.12665      267           0.9604317             0
    Enhancers.rank TFBSs_raw TFBSs.rank Grand_Total Grand_Total.rank
247    0.003597122      1000  0.1654676    23.06835                1
82     0.003597122      1000  0.1654676    22.93885                2
```

## SNP ranking from a dataframe

This function is for data taken straight from the Annotation step. To run the prioritisation function the user must define the dataframe to be analysed; a weighting model to use; an optimal MAF to rank SNPs on and; an name to embed into the output file name:

Here we are using the ALL model and setting the MAF to the default, "MAF" (which prioritises unique variants over common), and setting the output file tag to "bed.file.ALL.unique". This option is to make sure you can save each ranked list of SNPs using a unique name every time. Use print to see the values of these variables:

```
> ALL

[1]  0  1  4  0 13  1  0  2

> MAF

[1] 1e-06

> out <- "bed.file.ALL.unique"
> mydata2 <- SuRFR_analysis_data.frame(annotation_table,out, ALL, MAF=unique)
> mydata2[1:5,]

                    Pos GERP_raw GERP.rank GERP_av_rank Conservation.rank
12 chr1:6052580-6052581    0.352        13         0.65              0.65
1  chr1:1853587-1853588    1.320        18         0.90              0.90
```

```
2  chr1:1946590-1946591    1.300        17         0.85              0.85
4  chr1:1981117-1981118    0.000         1         0.05              0.05
11 chr1:6051265-6051266    1.490        19         0.95              0.95
   DNaseHS_raw DNaseHS.rank DNaseHS.av.rank DNase_foot_raw DNase.foot.rank
12        1000           20            1.00             32              20
1            0            1            0.05              0               1
2         1000           20            1.00              5              19
4         1000           20            1.00              2              18
11        1000           20            1.00              1              17
   DNase.foot.av.rank Ernst_score Ernst.Av.new.rank Position.rank
12               1.00          10                 1          1.00
1                0.05          10                 1          1.00
2                0.95          10                 1          0.80
4                0.90          10                 1          0.85
11               0.85          10                 1          0.80
   Position_score DAF_raw DAF.rank DAF.rank_normalised Enhancers_raw
12              5 0.80607        3                0.15             0
1               5 0.04354       20                1.00             0
2               3 0.82982        2                0.10             0
4               4 0.06332       19                0.95             0
11              3 0.36939       13                0.65             0
   Enhancers.rank TFBSs_raw TFBSs.rank Grand_Total Grand_Total.rank
12           0.05      1000       1.00       20.65                1
1            0.05         0       0.05       18.05                2
2            0.05       440       0.90       18.00                4
4            0.05      1000       1.00       18.00                4
11           0.05       306       0.85       17.90                5
```

## SNP ranking from an input file

This function works the same way as the last function except that it takes in a tab delimited text file instead of a dataframe. This input file must contain the relevant column headings or the function will not run.

This example uses a test file which is a tab delimited bed file containing the required functional annotation data for 39 non-coding, pathogenic variants for Beta thallasemia, proximal to the HBB gene:

```
> filename <- system.file("extdata","HBB_TP_noncoding.bed", package="SuRFR")
> HBB_noncoding <- SuRFR_analysis_tab_file(filename, DM, unique)
```

And that is it. If you have any problems, please check out our user manual.

**Happy SuRFing!**

# References


Maurano, M.T., et al., Systematic localization of common disease-associated variation in regulatory DNA. Science, 2012. 337(6099):p1190-5

Cooper, G.M., et al., Distribution and intensity of constraint in mammalian genomic sequence. Genome Res, 2005. 15(7): p. 901-13.

Sabo, P.J., et al., Discovery of functional noncoding elements by digital analysis of chromatin structure. Proc Natl Acad Sci U S A, 2004. 101(48): p. 16837-42.

Song, L. and G.E. Crawford, DNase-seq: a high-resolution technique for mapping active gene regulatory elements across the genome from mammalian cells. Cold Spring Harb Protoc, 2010. 2010(2): p. pdb prot5384.

Sabo, P.J., et al., Genome-scale mapping of DNase I sensitivity in vivo using tiling DNA microarrays. Nat Methods, 2006. 3(7): p. 511-8.

Ernst, J., et al., Mapping and analysis of chromatin state dynamics in nine human cell types. Nature, 2011. 473(7345): p. 43-9.

Andersson, R., et al., An atlas of active enhancers across human cell types and tissues. Nature, 2014. 507(7493): p. 455-61.

Wang, J., et al., Sequence features and chromatin structure around the genomic regions bound by 119 human transcription factors. Genome Res, 2012. 22(9): p. 1798-812.

Wang, J., et al., Factorbook.org: a Wiki-based database for transcription factor-binding data generated by the ENCODE consortium. Nucleic Acids Res, 2013. 41(Database issue): p. D171-6.