RegressHaplo Example

Sivan Leviyang

February 12, 2017

The folder **data/** contains a BAM and associated index file for a paired-end NGS dataset. The reads are synthetic, constructed using the ART simulation package.

Running the Pipeline

Make sure you load RegressHaplo

```
library(RegressHaplo)
```

then run the RegressHaplo pipeline on the BAM file.

```
bam_file <- "data/example.bam"
out_dir <- "output/"
full_pipeline(bam_file, out_dir, start_pos=500, end_pos=1500, num_trials=700)
```

We have chosen to restrict reconstruction to reference positions 500 through 1500 and to optimize the penalized regression 700 times. The haplotypes returned will cover the full consensus, composed of 2500 positions, but positions outside the $500 - 1500$ region will be set to consensus values. (Here we restrict to such a short region for the sake of an example with relatively short run time.)

Parsing RegressHaplo Results

In this case, we directed RegressHaplo to place output files in the **output/** directory. The final output file is **output/final_haplo.fasta**. The fasta file can be accessed directly; here we use Biostrings to read in and see the sequences.

```
haps <- readDNAStringSet("output/final_haplo.fasta")
haps
```

```
##   A DNAStringSet instance of length 4
##     width seq                                               names
## [1]  2499 ATGGGATGTCTTGGGAATCAG...GGAGCTATTTCCATGAGGCG haplotype1_0.5488
## [2]  2499 ATGGGATGTCTTGGGAATCAG...GGAGCTATTTCCATGAGGCG haplotype2_0.2611
## [3]  2499 ATGGGATGTCTTGGGAATCAG...GGAGCTATTTCCATGAGGCG haplotype3_0.1243
## [4]  2499 ATGGGATGTCTTGGGAATCAG...GGAGCTATTTCCATGAGGCG haplotype4_0.0658
```

But RegressHaplo also provides several functions to view and analyze the reconstrution.

```
# determine the positions on the reference that were considered variable
get_variable_positions.pipeline(out_dir)
```

```
##  [1]  558  588  634  648  677  705  796  860  973  986 1003 1031 1043 1085
## [15] 1153 1279 1300 1307 1337 1491
```

```
# get haplotype information
info <- get_fasta.pipeline(out_dir)
# info is a list containing the elements haplotypes and freq
# freq gives the frequency of the reconstructed haplotypes
info$freq
```

```
## [1] 0.5488 0.2611 0.1243 0.0658
```

```
# info$haplotype is a character vector containing the haplotypes.
class(info$haplotypes)
```

```
## [1] "character"
```

```
length(info$haplotypes)
```

```
## [1] 4
```

```
# we can use Biostrings to see the haplotypes since the sequences are rather long
DNAStringSet(info$haplotypes)
```

```
##   A DNAStringSet instance of length 4
##     width seq                                               names
## [1]  2499 ATGGGATGTCTTGGGAATCAG...GGAGCTATTTCCATGAGGCG haplotype1_0.5488
## [2]  2499 ATGGGATGTCTTGGGAATCAG...GGAGCTATTTCCATGAGGCG haplotype2_0.2611
## [3]  2499 ATGGGATGTCTTGGGAATCAG...GGAGCTATTTCCATGAGGCG haplotype3_0.1243
## [4]  2499 ATGGGATGTCTTGGGAATCAG...GGAGCTATTTCCATGAGGCG haplotype4_0.0658
```

Evaluating Regression Performance

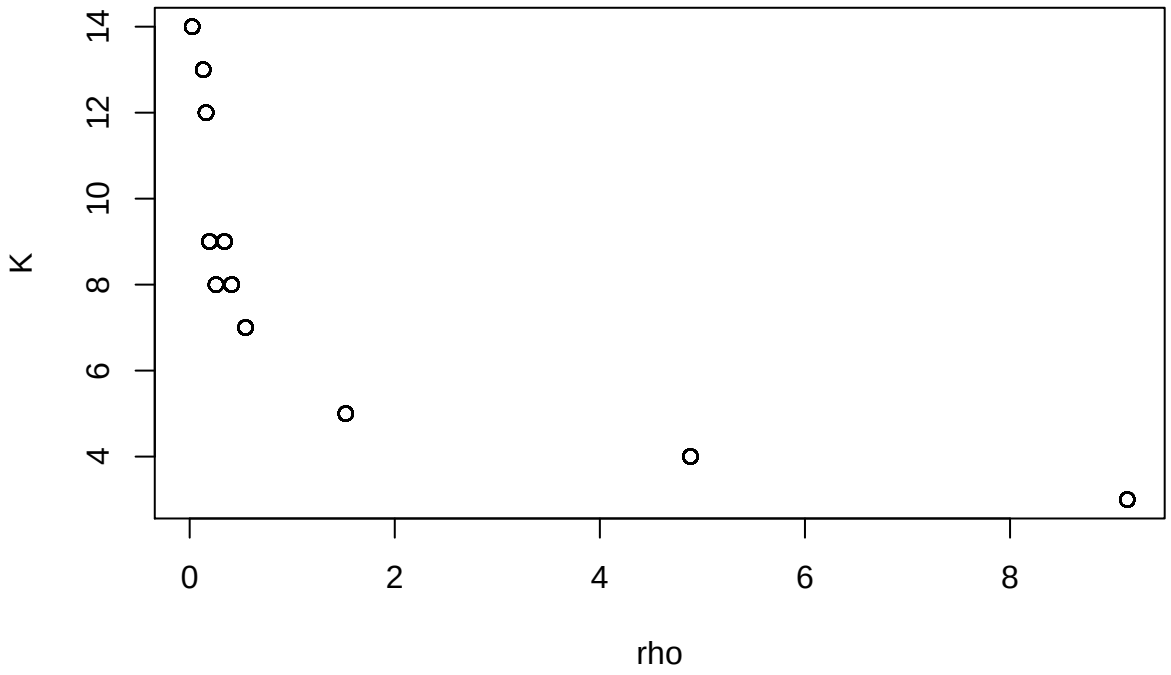
After running RegressHaplo, check if the solutions have captured a range of haplotype reconstructions. The solutions summary data.frame describes the results of the 700 trials. Each row in the data.frame corresponds to a single trial and provides K (the number of reconstructed haplotypes), fit (a fit measure, lower is better), and rho (the penalty parameter). Typically many trials generate the same result - meaning they find the same local minimum.

```
df <- get_solutions_summary.pipeline(out_dir)
# let's look at every 100th solution
df[seq(from=100,to=700,by=100),]
```

```
##           rho  K       fit solution_number
## 100 0.1325711 13 0.1913357             100
## 200 0.1930698  9 0.1925051             200
## 300 0.3393222  9 0.1929382             300
## 400 0.4094915  8 0.1936661             400
## 500 0.5452668  7 0.1942780             500
## 600 4.8831237  4 0.2529478             600
## 700 9.1432139  3 0.4715639             700
```

Check that the different rho have produced a range of K values

```
plot(df$rho, df$K, xlab="rho", ylab="K")
```



We have captured solutions with K values (the number of haplotypes reconstructed) ranging from 3 to 14, although $K = 6, 10, 11$ are missing. This is a good spread of K values; often a particular K value will be missed.

We can check the *rho* values used

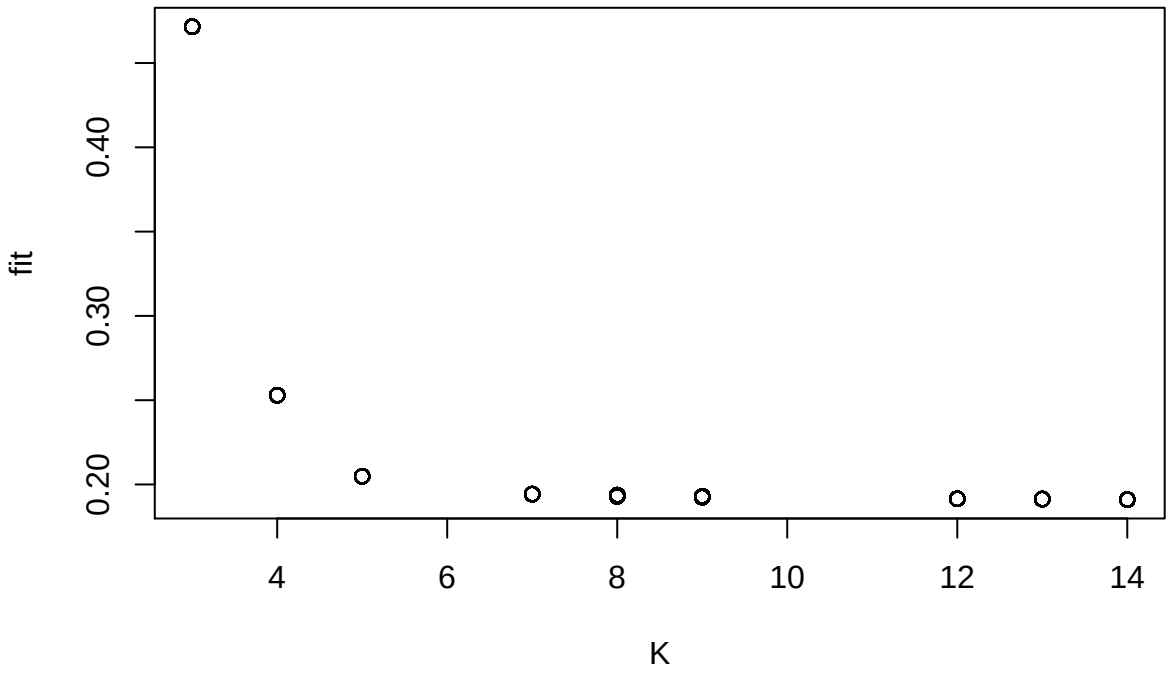
```
unique(df$rho)
```

```
##  [1] 0.02447103 0.13257114 0.15998587 0.19306977 0.33932218 0.25708603
##  [7] 0.40949151 0.54526683 1.52191820 4.88312366 9.14321386
```

If further K values are desired, additional rho values can be specified through the `full_pipeline` rho parameter, see help for specifics. Increasing rho will lower K.

We can also check whether the final solution chosen, $K = 4$, reflects a good tradeoff between over and under fitting.

```
#  K values vs fit
plot(df$K, df$fit, xlab="K", ylab="fit")
```



Here $K = 5$ would have produced a better fit, but $K = 4$ seems reasonable.