WaveQTL

June 14, 2014

1 Overview

The `WaveQTL` software implements a wavelet-based approach for genetic association analysis of functional phenotypes (e.g. sequence data arising from high-throughput sequencing assays) described in [2]. A key factor that distinguishes a wavelet-based approach from typical association analysis is to better exploit high-resolution measurements provided by high-throughput sequencing assays (see [2] for motivations). The implemented wavelet-based methods aim to

- test for genetic association of functional phenotype;
- provide explanations for observed associations such as which parts and features of the data are driving the association;
- make applications to large-scale genetic association analyses computationally tractable.

Although wavelet methods were motivated primarily by genetic association studies for high-throughput sequencing data, they could also test for association between functional data and other covariates, either continuous or discrete. For example, in a genomics context, it could be used to detect differences in gene expression (from RNA-seq data) or TF binding (from ChIP-seq data) measured on two groups (e.g. treatment conditions or cell types). Or it could be used to associate a functional phenotype, such as chromatin accessibility, with a continuous covariate, such as "overall" expression of a gene. It could also be used for genome-wide association studies of functional phenotypes unrelated to sequencing.

2 How to cite WaveQTL

Heejung Shim and Matthew Stephens (2014). Wavelet-based genetic association analysis of functional phenotypes arising from high-throughput sequencing assays. arXiv. 1307.7203.

3 Functional phenotypic data preprocessing

The `WaveQTL` software takes normalized wavelet coefficients (WCs) as an input phenotype. We provide multiple R functions in `R/WaveQTL_preprocess_funcs.R` that preprocess functional phenotypic data as described in [2] and produce inputs required by the `WaveQTL`. Here, we describe how to use the main function `WaveQTL_preprocess` using dsQTL data from [2] as an example. `R/WaveQTL_preprocess_example.R` contains R scripts used in this section. For details of preprocessing procedure, please read our paper [2], and for details of the function `WaveQTL_preprocess` and other functions called by `WaveQTL_preprocess`, see the description of functions in `R/WaveQTL_preprocess_funcs.R`.

The main function `WaveQTL_preprocess` takes functional data ("Data") and optionally library read depth ("library.read.depth"), a list of covariates ("Covariates"), minimum read count for filtering ("meanR.thresh") as an input. If "library.read.depth" is provided, the function standardizes the functional data by the library read depth to account for different library read depths across individuals. Then, the function decomposes the (standardized) functional data into WCs using a `wavethresh` R package and normalizes the WCs. If "Covariates" are provided, the function corrects the WCs for the Covariates during the normalization. See the description of the function `Normalize.WCs` in `WaveQTL_preprocess_funcs.R` for details of normalization. Users can specify which type of wavelet transform should be applied by using "filter.number" and "family" in input arguments. They are arguments to the function `wd` in the R package `wavethresh` (see their manual for details). For now, the function doesn't allow users to specify the level of wavelet decomposition and the function uses the maximum level decomposition. In addition to wavelet transform, this function filters out some WCs that are computed based on very low counts. The function considers WCs to have low count if the total counts used in their computation are less than or equal to "meanR.thresh" per individual on average. Finally, the function `WaveQTL_preprocess` returns quantile transformed (standardized and covariates corrected) WCs ("WCs") and filtering status for each WCs ("filtered.WCs") as an output.

We start with making a directory to save output from the function `WaveQTL_preprocess`.

```
cd WaveQTL
mkdir test
cd test
mkdir dsQTL
```

Now open `R` session, set working directory to `WaveQTL/R` and read functions in `WaveQTL_preprocess_funcs.R`.

```
> setwd("~/WaveQTL/R")
> source("WaveQTL_preprocess_funcs.R")
```

Then, set a path to the directory containing dsQTL data that is shown in Figure 2 of [2] and read data on 70 individuals.

```
> data.path = "../data/dsQTL/"
# read functional phenotypic data
> pheno.dat = as.matrix(read.table(paste0(data.path,
+ "chr17.10160989.10162012.pheno.dat")))
> dim(pheno.dat)
[1] 70 1024
# read library read depth
> library.read.depth = scan(paste0(data.path, "library.read.depth.dat"))
> length(library.read.depth)
[1] 70
# read Covariates
> Covariates = as.matrix(read.table(paste0(data.path, "PC4.dat")))
> dim(Covariates)
[1] 70 4
```

Each row of `pheno.dat` contains functional phenotypic data for each individual on a region of 1024bp (B = 1024bp). For example, this functional phenotypic data can be the number of reads starting at each location of the region. In our dsQTL analysis of [2], we obtained functional phenotypic data after applying multiple procedures into read counts to avoid potential biases. We describe how to replicate those procedures in Section ??. Each element of `library.read.depth` contains library read depth for each individual. `Covariates` contains multiple covariates to be corrected for (here, four principal components we used to control for confounding factors in our dsQTL analysis of [2]).

We set two to "meanR.thresh" for filtering of low count WCs and run `WaveQTL_preprocess`.

```
> meanR.thresh = 2
> res = WaveQTL_preprocess(Data = pheno.dat, library.read.depth =
+ library.read.depth, Covariates = Covariates, meanR.thresh = meanR.thresh)
> str(res)
List of 2
 $ WCs         : num [1:70, 1:1024] -1.415 0.126 0.347 -1.323 0.674 ...
 $ filtered.WCs: num [1:1024] 1 1 1 1 1 1 1 1 0 0 ...
```

`WaveQTL_preprocess` returns a list of `WCs` and `filtered.WCs`. `WCs` is a matrix of the number of individuals (70) by the number of WCs (1024). Each row contains (standardized, covariates corrected, and quantile transformed) WCs for each individual. WCs are ordered from low resolution WC to high resolution WC. For example, with a Haar wavelet transform, the first column contains WC (precisely speaking, scaling coefficient) that corresponds to sum of data in the region. The second column contains WC that contrasts the data in the first half vs second half of the region. The last (1024) column contains WC that contrasts the data in the 1023-th base vs 1024-th base. `filtered.WCs`, a vector of length 1024, contains indicators to show whether t-th WC in output `WCs` is filtered (0) or not (1) due to low count.

Finally, set a path to the output directory we created at the beginning and save output as files.

```
> output.path = "../test/dsQTL/"
> write.table(res$WCs, file= paste0(output.path, "WCs.txt"), row.names=FALSE,
+ col.names = FALSE, quote=FALSE)
> cat(res$filtered.WCs, file = paste0(output.path, "use.txt"))
```

4 WaveQTL

4.1 Input file format for WaveQTL

`WaveQTL` requires two input files containing genotypes and phenotype (WCs), and optionally two input files containing filtering status of WCs and information on which WCs share the same hyper parameter π (see [2] for details on π).

4.1.1 Genotype file

`WaveQTL` takes genotype file in the BIMBAM format `http://stephenslab.uchicago.edu/software.html` [1].

5 Software

We can use the DHMT package from here: `http://dsp.rice.edu/software/WHMT`

They have both c and matlab implementations of HMT. One possibility is just changing their code. It is prefferable to use the c code as it will be faster. One problem is that their code works for 2-dimensional wavelets (for images) and we need a 1-dimensional wavelets (for genomic data)

References

[1] Bertrand Servin and Matthew Stephens. Imputation-based analysis of association studies: candidate regions and quantitative traits. *PLoS genetics*, 3(7):e114, July 2007.

[2] H. Shim and M. Stephens. Wavelet-based genetic association analysis of functional phenotypes arising from high-throughput sequencing assays. *ArXiv e-prints*, July 2013.