A Data Analysis Tutorial for DSPR data

Elizabeth King
Department of Ecology & Evolutionary Biology
University of California, Irvine
egking@uci.edu

06 June 2013

1 Prerequisites

If you do not have R, navigate to http://www.r-project.org to download and install it. New users should download and read the reference manual.

2 Installing the DSPRqtl Analysis Package

To install the analysis package, within R, execute:

```
install.packages("DSPRqtl",
                 repos = "http://wfitch.bio.uci.edu/R/",
                 type = "source")
```

Note: you must use the statement `repos = "http://wfitch.bio.uci.edu/R/"`.

3 Installing the DSPRqtlData Packages

1. Install one or both data packages (`DSPRqtlDataA` and/or `DSPRqtlDataB`). If only one population was used, only one data package (A or B) needs to be installed.
2. The data packages are ~3.2 GB each and will take several minutes (to hours) to download and install depending on your connection speed. These packages are too large to be stored on and installed from CRAN. If you do not install these packages, the functions in the `DSPRqtl` package will fetch each position file from the internet. This method takes considerably longer to run.

3.1 Installation within R

Within R (not recommended in RStudio), type:

```
install.packages("DSPRqtlDataA",
                 repos = "http://wfitch.bio.uci.edu/R/",
                 type = "source")
```

and/or

```
install.packages("DSPRqtlDataB",
                 repos = "http://wfitch.bio.uci.edu/R/",
                 type = "source")
```

for the pA and pB data packages, respectively.

Note: you must use the statement `repos = "http://wfitch.bio.uci.edu/R/"`.

3.2 Installation from the Downloaded Package Source

If installation within R fails, the packages can be downloaded manually and installed from within R from local source. Download the package source files from:

- http://wfitch.bio.uci.edu/R/src/contrib/DSPRqtlDataA_2.0-1.tar.gz
- http://wfitch.bio.uci.edu/R/src/contrib/DSPRqtlDataB_2.0-1.tar.gz

using a web browser or command line tool (e.g., `wget`).

After the files have downloaded, execute the following commands in R (one, the other, or both as necessary):

```
install.packages("DSPRqtlDataA_2.0-1.tar.gz", repos = NULL, type = "source")
install.packages("DSPRqtlDataB_2.0-1.tar.gz", repos = NULL, type = "source")
```

Note: You may have to modify the path to the package file.

4 Loading and Using the Data Packages

Once the data packages are installed, they can be loaded using:

```
library(DSPRqtlDataA)
```

and/or

```
library(DSPRqtlDataB)
```

Then, the set of additive HMM probabilities and raw HMM probabilities for any given position can be obtained with:

```
data(A_chromosome.arm_position.in.base.pairs)
```

e.g.,

```
data(A_X_12000000)
# This gives a data.frame named A_X_12000000
```

A list of regularly spaced positions every 10 kb is available at http://FlyRILs.org/Data or within the `DSPRqtl` package (see above for install instructions).

```
library(DSPRqtl)
data(positionlist_wgenetic)
# This gives a data.frame named poslist
```

5 Data Analysis Tutorial

Before you begin, please also obtain the DSPRqtl manual (see http://FlyRILs.org/Tools/Tutorial) and refer to it throughout this tutorial. Also load the `DSPRqtl` package:

```
library(DSPRqtl)
```

5.1 Phenotype Data

- Format your phenotype file and read it into R. All phenotype files must have at a minimum a column named `"patRIL"` (and in the case of crossing designs, also one named `"matRIL"`) containing the numeric RIL IDs (e.g., 11001, 11002, ...) and a column with phenotype measurements (with a name chosen by the user). Cross designs must also contain a column specifying the sex of the individuals measured so the X chromosome is handled correctly.
- If covariates are to be used, these should also be in this file (or added to it within R before beginning the analysis).
- To view an example, load the example ADH data in the `DSPRqtl` package. Throughout this tutorial, the example dataset `ADHdata` (included in the R package) will be used to illustrate each step.

```
data(ADHdata)
# a data.frame named ADHdata

head(ADHdata)
```

```
##   patRIL matRIL    adh
## 1  11001  11001 -44.09
## 2  11002  11002 -55.76
## 3  11003  11003 -47.58
## 5  11005  11005 -39.89
## 6  11006  11006 -56.91
## 7  11007  11007 -73.39
```

To import your data into R, you must read the file in. R will accept many types of files. For help reading files into R, see http://cran.r-project.org/doc/manuals/R-data.html. The code to read in a text file is shown below as an example.

```
phenotype.dat <- read.table("/path/to/your/file.txt",
                            header = TRUE)
```

5.2 Genome Scan

One can perform a genome scan using the `DSPRscan()` function for experimental designs where the inbred RILs are measured directly, crosses to a standard, or the pA-pB cross design. For users requiring more flexibility, see the *Advanced Analysis* section below. Handling interactive covariates and analyzing multiple phenotypes at once is in development. The `DSPRscan()` function performs a genome scan that regresses the 8 founder genotype probabilities (or 16 in the case of the pA-pB cross) on the measured phenotype at evenly spaced 10 kb positions across the genome:

```
DSPRscan(model,design,phenotype.dat,id.col,batch,sex)
```

- `model` is the null model in formula notation. For a phenotype with no covariates, it is of the form `phenotype ~ 1`. With a single additive covariate, it is: `phenotype ~ covariate`.

- `design` is either `"inbredA"` or `"inbredB"` for inbred designs pA and pB RILs and crosses to a standard. For the pA-pB cross, use `"ABcross"`.
- `phenotype.dat` is the `data.frame` read in the section *Phenotype Data* containing the RIL IDs, phenotype, and any covariates.
- `id.col` identifies the column to use as unique ids for the measurements. For an inbred design, this column could be the patRIL column. For a cross design, a unique id column should be included.
- `batch` is the number of positions tested at one time. Defaults to 1,000. If memory use is a problem, reduce this number.
- `sex` is the sex of the individuals measured and must be specified for the ABcross. It is either `"m"` or `"f"`. For inbred designs, it can be left out.

For a single phenotype with the data package installed, a genome scan should take ~15–20 min.
Using the ADH example data:

```
data(ADHdata)
scan.results <- DSPRscan(adh ~ 1,
                         design = "inbredA",
                         phenotype.dat = ADHdata,
                         id.col='patRIL')
```

The output of `DSPRscan()` is a list containing:

- `LODscores` This is a `data.frame` with positions and LOD scores.
- `model` This is the model statement.
- `design` This is the design specified.
- `phenotype` This is the phenotype data.
- `sex` This is the sex of the individuals measured.

There is example output from a finished genome scan of the ADH data in this package as well. Type:

```
data(ADHscan)
```

To extract the LOD scores `data.frame`:

```
ADH.lod.scores <- ADHscan$LODscores

ADH.lod.scores[100:105, ]
```

```
##     chr    Ppos   Gpos   LOD
## 100   X 1150000 0.1359 1.662
## 101   X 1160000 0.1419 1.661
## 102   X 1170000 0.1479 1.661
## 103   X 1180000 0.1540 1.661
## 104   X 1190000 0.1601 1.663
## 105   X 1200000 0.1663 1.664
```

5.3 Permutation Test

Perform a permutation test using the function `DSPRperm()` to obtain a significance threshold. For initial data exploration, the value 6.8 can be used for inbred designs and 10.1 for the ABcross. This threshold seems to be fairly stable for multiple phenotypes that we have tested, but we recommend each user perform a permutation test for their specific data set.

```
DSPRperm(model, design, phenotype.dat, id.col, batch, niter, alpha, sex)
```

For `model, design, phenotype.dat, id.col, batch, sex` see above description for `DSPRscan()`. The two additional arguments to `DSPRperm()` are:

- `niter` The number of permutations to perform. Default is 1,000.
- `alpha` The desired Type I error rate.

The output of `DSPRperm()` is a list containing:

- `maxLODs` A vector of maximum LOD scores for each permutation.
- `alpha` The specified alpha level
- `threshold` The significance threshold.

After a permutation test is performed, the `maxLODs` can also be used to determine the significance threshold at another alpha level. For example,

```
perm.test <- DSPRperm(adh ~ 1,
                      design = "inbredA",
                      phenotype.dat = ADHdata)
```

For $\alpha = 0.01$:

```
quantile(perm.test$maxLODs, 1-0.01)
```

5.4 Identify Significant QTL

Get a summary of the significant peaks. Finding and summarizing significant peaks can be done in a single step using the function `DSPRpeaks()`. The individual functions to get the values are also available (see the `DSPRqtl` manual). After peaks are identified, it is important for the user to confirm these represent distinct peaks.

```
DSPRpeaks(qtldat,method,threshold,LODdrop,BCIprob)
```

- `qtldat` Output from `DSPRscan()`.
- `threshold` The threshold found with the permutation test. Defaults to 6.8 or 10.1 for inbred and ABcross designs respectively.
- `LODdrop` The desired LODdrop for support intervals. Defaults to 2.
- `BCIprob` The desired nominal Bayes fraction for support intervals. Defaults to 0.95.

The output of `DSPRpeaks()` is a list of peaks. Each peak is a list containing:

- `threshold` The specified threshold.
- `peak` The peak position and LOD score.
- `LODdrop` The specified LODdrop.
- `BCIprob` The specified BCIprob.
- `CI` The confidence interval.
- `founderNs` The number of RILs with each founder genotype at the peak.
- `geno.means` The founder means and standard errors at the peak.
- `perct.var` The percent variation explained by the QTL.
- `entropy` The proportion missing information.

Using the `ADHscan` output:

```
peaks <- DSPRpeaks(ADHscan, threshold = 6.8, LODdrop = 2)
```

And the main QTL is found at position 26 in the list:

```
peaks[[26]]
```

```
## $threshold
## [1] 6.8
##
## $peak
##    chr     Ppos  Gpos   LOD
## 26  2L 14500000 49.91 107.4
##
## $LODdrop
## [1] 2
##
## $CI
##             chr     Ppos  Gpos   LOD
## Lower Bound  2L 14410000 49.79 105.4
## Upper Bound  2L 14620000 50.07 105.4
##
## $founderNs
##        A1        A2        A3        A4        A5        A6        A7
##        94        21         1       355       186        20         9
##        A8      Hets Uncertain
##        42        10        84
##
## $geno.means
##    Estimate Std. Error
## A1   -29.03     0.8126
## A2   -32.61     1.7962
## A3   -58.39     6.2789
## A4   -50.40     0.4272
## A5   -49.55     0.5805
## A6   -51.30     1.7197
## A7   -55.61     2.4104
## A8   -53.16     1.2484
##
## $perct.var
## [1] 45.22
##
## $entropy
## [1] 0.03688
```

The `DSPRscan()` results can be plotted using the `DSPRplot()` function. Multiple scan results can be plotted on the same plot. Pass the scan results as a `list()` to the `DSPRscan()` function.

```
DSPRplot(list(ADHscan), threshold=6.8)
```

```
## Error: '...' used in an incorrect context
```

5.5 Local Interval Mapping

The user may wish to perform local interval mapping to compare the peak locations and confidence intervals. The `LocalInt()` function will perform interval mapping for a range of positions given by the user. `FindCI()` can then be used to re-estimate confidence intervals. This function is only available for the `"inbredA"` or `"inbredB"` designs.

```
LocalInt(peakChr, peakPos, range, phenotype.dat, pheno.name, design)
```

- `peakChr` Chromosome arm at the peak.
- `peakPos` Position in base pairs at the peak.
- `range` The range of positions to examine. Default is 100 (on either side, positions are 10 kb apart)
- `phenotype.dat` The phenotype `data.frame`. See the section *Phenotype Data*.
- `pheno.name` The name of the column containing the phenotype.
- `design` `"inbredA"` or `"inbredB"`

The output of `LocalInt` is the same as the LODscores from `DSPRscan` but only for the specified set of positions.

Using the ADH sample data:

```
# The main QTL
main.peak <- peaks[[26]]
peakChr <- main.peak$peak$chr
peakPos <- main.peak$peak$Ppos
```

```
peak.int <- LocalInt(peakChr,
                     peakPos,
                     phenotype.dat = ADHdata,
                     pheno.name = "adh",
                     design = "inbredA")
```

6 Advanced Analysis

6.1 Generate Genotype Information

This package helps automate analysis for simple DSPR designs. Many users (especially those with complex interactions or complex cross designs) will require more flexibility. To aid those users, the `DSPRgenos()` function is available to set up the genotype information using a given phenotype data file. This function can handle the following designs: * `"inbredA"` and `"inbredB"` Direct measurements on the RILs or crosses to a standard * `"AAcross"` and `"BBcross"` Crosses within the pA or pB RIL panels (e.g. round robin design) * `"ABcross"` Crosses between the pA and pB RIL panels.

The function can handle multiple measurments, a mix of males and females, and a mix of cross directions in the case of the pA-pB cross (i.e. A males to B females and vice versa). The function outputs either a list or a 3 dimensional array and the user can then use the many built in analysis tools in R to fit a model across the genome.

```
DSPRgenos(design,phenotype.dat,id.col,output)
```

- `design` `"inbredA"`,`"inbredB"`,`"AAcross"`,`"BBcross"`, or `"ABcross"`.
- `phenotype.dat` The phenotype `data.frame`. See the section *Phenotype Data*.

- `id.col` identifies the column to use as unique ids for the measurements. For an inbred design, this column could be the patRIL column. For a cross design, a unique id column should be included.
- `output` Format for the genotype output. Either `"list"` or `"array"`. Default is `"list"`.

The output of `DSPRgenos()` is a list containing:

- `genolist` a list containing the matrix of additive genotype probabilities at each position in the genome. The list is in the same order as the list of positions described below. Column names are the different DSPR haplotypes and row names are the unique ids provided in id.col. Alternatively, if output='array', the output is a 3 dimensional array [samples,haplotypes,positions]. Haplotypes and samples are named while positions are in the order of the list of positions described below.
- `positions` a `data.frame` containing regularly spaced positions (every 10KB) in the genome where the genotype probabilities are calculated. Columns are: chr = chromosome, Ppos = physical position (zero offset), Gpos = genetic position, Gaxis = cummulative genetic position.
- `phenotype` the phenotype `data.frame` ordered by the specified id column and in the same order as the genotype information at each position in the genome in the geno list described above.

The user can easily use apply (in the case of an array), lapply (in the case of list), or a loop to do a genome scan of whatever type they wish. Future tutorials will provide sample code for specific senarios (e.g. the round robin design).

The DSPRgenos function is somewhat time consuming to run (~30min - 1 hour). However genome scans using the output should be fast (2-10 min for simple designs). The user will likely wish to save the output especially if planning to run genome scans with multiple different models. Because the output is a large object (~11,000 positions in the genome), it is much more efficient to use the `save` function in R as shown below:

```
myGenos<-DSPRgenos(design,phenotype.dat,id.col,output)
save(myGenos, file="my file path")
```

To return to these results later, use `load`:

```
load(file="my file path")
# The object will load into the workspace with the same name as
# when save was used. In this example, this is myGenos.
```

7 Questions?

If you have any questions or have trouble with this tutorial, please contact flyrils@gmail.com.