Statistical analysis code for analysis of CASTxB6 F2 mouse cross

2. QTL analysis

Peter Langfelder

May 3, 2011

Contents

1 Preliminaries

In this document we detail the eQTL analysis of the CASTxB6 cross. The first step is to set up the R session and load the requisite genotype data and expression data. Missing genotype data have been imputed using a standard imputation procedure. Quality analysis of genotypes indicated that many of the B6xCAST animals had genotypes not consistent with expected proportions of homozygous and heterozygous alleles in an F2 cross. For this reason we exclude all B6xCAST animals from this study.

```
# Load the WGCNA library and the qtl library
library(WGCNA)
library(qtl)
# Source additional custom functions (for producing QTL plots)
source("networkFunctions-extras-05.R");
# Load the imputed snp data
file = bzfile("../../../../Data-CXB/CXB_BXC_MF_clinical_traits_GT_imputed_031408.csv.bz2");
geno0 = read.csv(file);
# Display the first few rows and columns
geno0[1:5, 1:20]
# Isolate marker names
markerNames = names(geno0)[substring(names(geno0), 1, 2) %in% c("rs", "mC") ];
split = sapply(strsplit(markerNames, split = ".", fixed = TRUE), I);
markerID = split[1, ];
chromo = as.numeric(substring( split[2, ], 4));
basePair = as.numeric(substring( split[3, ], 3));
SNPinfo = rbind( c("", chromo), c("", basePair));
colnames(SNPinfo) =c("Mice_id", markerID);
genoTable = geno0[, c(1, match(markerNames, names(geno0))) ];
```

We load the pre-processed expression data.

```
exprMice = list();
x = load("../../CxBOnly-Liver-outliersRemoved-exprFemaOR-pValFemaOR.RData");
exprMice[[1]] = rownames(exprFemaOR);
x = load("../../CxBOnly-Liver-outliersRemoved-exprMaleOR-pValMaleOR.RData");
exprMice[[2]] = rownames(exprMaleOR);
exprMFMice = c(exprMice[[1]], exprMice[[2]]);
gtMice = geno0$Mice_id;
```

```
# all mice with expressions are also present in the genotype table:
table(exprMFMice %in% gtMice)
```

Next we restrict the genotype data to mice with valid expression measurements and write out the corresponding csv files that the `readCross` function requires. We do not remove any outliers here.

```
genoWithExpr = genoTable[ match(exprMFMice, gtMice), ];
colnames(genoWithExpr) = colnames(SNPinfo);
genoForQTL = rbind(SNPinfo, genoWithExpr);
write.csv(genoForQTL, "CxB-genotypesForQTL-miceWithExpression.csv",
         row.names = FALSE, quote = FALSE);
# Prepare the corresponding trait data - these will be replaced by the expression data below.
traits0 = geno0[, -match(markerNames, names(geno0))];
traits1 = traits0[, -c(2:7, 9:17) ];
traitsForQTL = traits1[match(exprMFMice, traits1$Mice_id), ];
nTraits = ncol(traitsForQTL);
write.csv(traitsForQTL, "CxB-traitsForQTL-miceWithExpression.csv",
         row.names = FALSE, quote = FALSE);
```

We now use the `read.cross` function of the qtl package to read and process the data and ready it for QTL analysis.

```
# Read the cross that was prepared for the cQTL analysis
cross = read.cross(format = "csvs",
                 genfile = "CxB-genotypesForQTL-miceWithExpression.csv",
                 phefile = "CxB-traitsForQTL-miceWithExpression.csv",
                 genotypes = c(1,2,3), alleles = c("B", "C"));
```

We prepare gene expression data into a single data frame and use a bit of a hack to replace the trait phenotypes with the expression data.

```
exprAll = rbind(exprFemaOR, exprMaleOR);
nGenes = ncol(exprAll);
exprForQTL = data.frame(Mice_id = exprMFMice, exprAll);
# Replace phenotypes in the cross by the expression data
cross$pheno = exprForQTL;
summary(cross)
```

Since a sex effect can be expected in gene expression levels, we use sex as a covariate. The actual calculations of the eQTLs will take several hours, possibly several days. We save the results of the calculation for use in conjunction with the QTL analysis.

```
sex = as.numeric(factor(traitsForQTL$sex));
system.time( {eqtl = scanone(cross, pheno.col = c(1:nGenes)+1, addcovar = sex, method = "mr") });
save(eqtl, file = "eqtl-eqtl.RData");
```