demo_yeast

April 19, 2014

Contents

In this demo we show how to use LIMIX to build genetic models and perform different anlysis of single and multiple traits. We consider a yeast dataset (Smith et al, 2008, PLoS Biology) consisting of 109 individuals with 2,956 marker SNPs and expression levels for 5,493 in glucose and ethanol growth media respectively. Such a dataset is ideal to showcase LIMIX and multiple trait analysis, containing two different axis of multi-trait variation (multiple genes and multiple environments). In particular, in these demos, we focus on the subset of the 11 genes in the Lysine Biosynthesis KEEG pathway considered in the paper.

We start discussing GWAS, multi locus models and variance decomposition models for single trait analysis and then we show how there models are extended for joint analysis of multiple traits. Then we go on to discuss how the prediction tool made available by LIMIX can be used to monitor overfitting and perform model selection. Finally, we show how LIMIX can be used to build model that infer and account for hidden confounders from high-dimensional phenotype data both in single trait and multi trait analysis.

1 Setting up

In [1]:
```
%matplotlib inline
```

In [2]:
```
import scipy as SP
import pylab as PL
from matplotlib import cm
import h5py
import pdb
SP.random.seed(0)
```

In [3]:
```
# import limix
import sys
sys.path.append('./../build/release.darwin/interfaces/python')
import limix.modules.varianceDecomposition as VAR
import limix.modules.qtl as QTL
import limix.modules.data as DATA
```

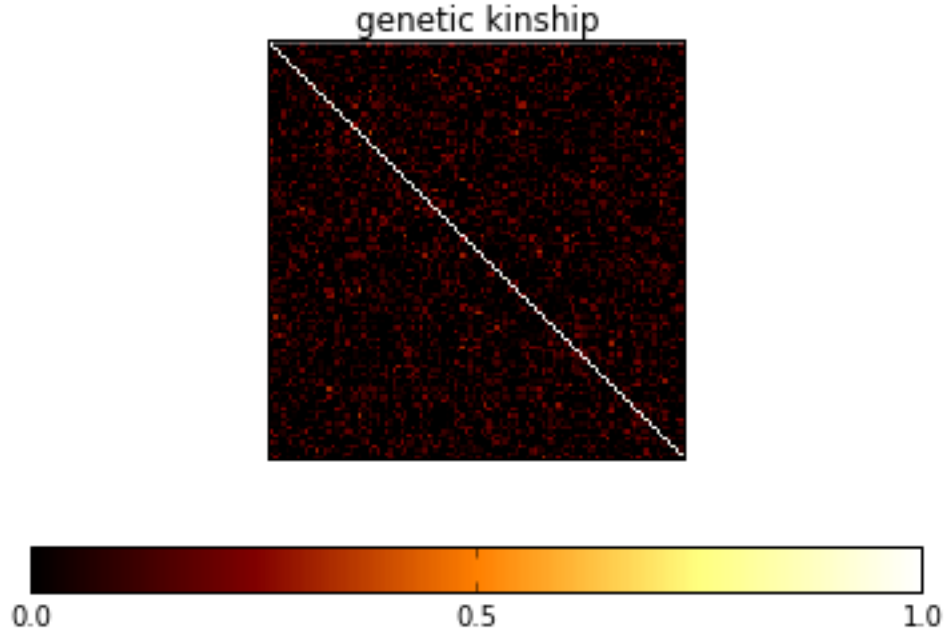
In [4]:
```
# temporary
from include.utils import *
from include.manhattan_script import plot_manhattan
```

In [5]:
```
#genes from lysine biosynthesis pathway
lysine_group = ['YIL094C', 'YDL182W', 'YDL131W', 'YER052C', 'YBR115C', 'YDR158W',
                'YNR050C', 'YJR139C', 'YIR034C', 'YGL202W', 'YDR234W']
```

In [6]:
```
#import data
fname = './data/smith_2008/smith08.hdf5'
data = DATA.QTLData(fname)
#getting genotypes
X = data.getGenotypes()
cumPos = getCumPos(data)
chromBounds = getChromBounds(data)
# non-normalized and normalized sample relatedeness matrix
R0 = SP.dot(X,X.T)
R  = R0/R0.diagonal().mean()
```

In [7]:
```
# plot sample relatedeness matrix
plt = PL.subplot(1,1,1)
PL.title('genetic kinship')
PL.imshow(R,vmin=0,vmax=1,interpolation='none',cmap=cm.afmhot)
PL.colorbar(ticks=[0,0.5,1],orientation='horizontal')
plt.set_xticks([])
plt.set_yticks([])
```

Out[7]: []



2 Single Trait Analysis

2.1 Single Locus Analysis

2.1.1 The Genetic Model

Indicating with N the number of samples, the standard LMM considered by LIMIX is

$$\mathbf{y} = \mathbf{F}\boldsymbol{\alpha} + \mathbf{x}\beta + \mathbf{g} + \boldsymbol{\psi}, \quad \mathbf{g} \sim \mathcal{N}\left(\mathbf{0}, \sigma_g^2 \mathbf{R}\right), \ \boldsymbol{\psi} \sim \mathcal{N}\left(\mathbf{0}, \sigma_e^2 \mathbf{I}_N\right) \tag{1}$$

where

$$\begin{aligned}
\mathbf{y} &= \text{phenotype vector} \in \mathcal{R}^{N,1} && (2)\\
\mathbf{F} &= \text{matrix of } K \text{ covariates} \in \mathcal{R}^{N,K} && (3)\\
\boldsymbol{\alpha} &= \text{effect of covariates} \in \mathcal{R}^{K,1} && (4)\\
\mathbf{x} &= \text{genetic profile of the SNP being tested} \in \mathcal{R}^{N,1} && (5)\\
\boldsymbol{\beta} &= \text{effect size of the SNP} \in \mathcal{R} && (6)\\
\mathbf{R} &= \text{sample relatedeness matrix} \in \mathcal{R}^{N,N} && (7)\\
& && (8)
\end{aligned}$$

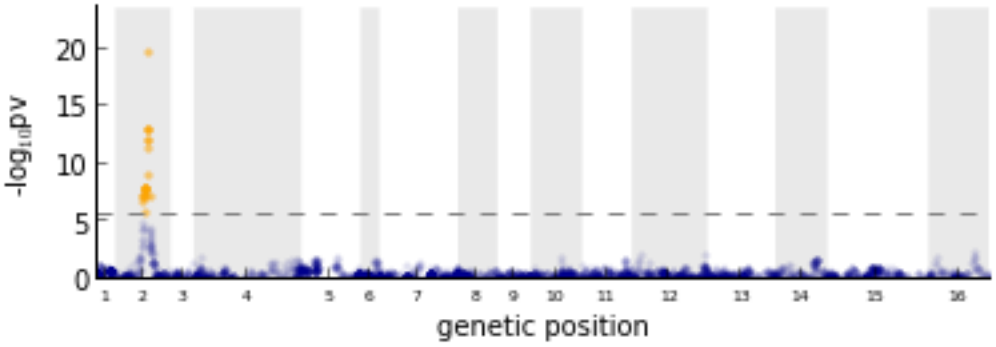
Association between phenotypic changes and the genetic markers is tested by testing $\beta \neq 0$. The model can be rewritten using the delta-representation

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{W}\boldsymbol{\alpha} + \mathbf{x}\beta, \sigma_g^2 \left(\mathbf{R} + \delta \mathbf{I}\right)\right) \tag{9}$$

where $\delta = \sigma_e^2 / \sigma_g^2$ is the signal-to-noise ratio. and is refitted SNP-by-SNP during the scan.

2.1.2 Example: Single-trait eQTL mapping for gene YBR115C in environment 0

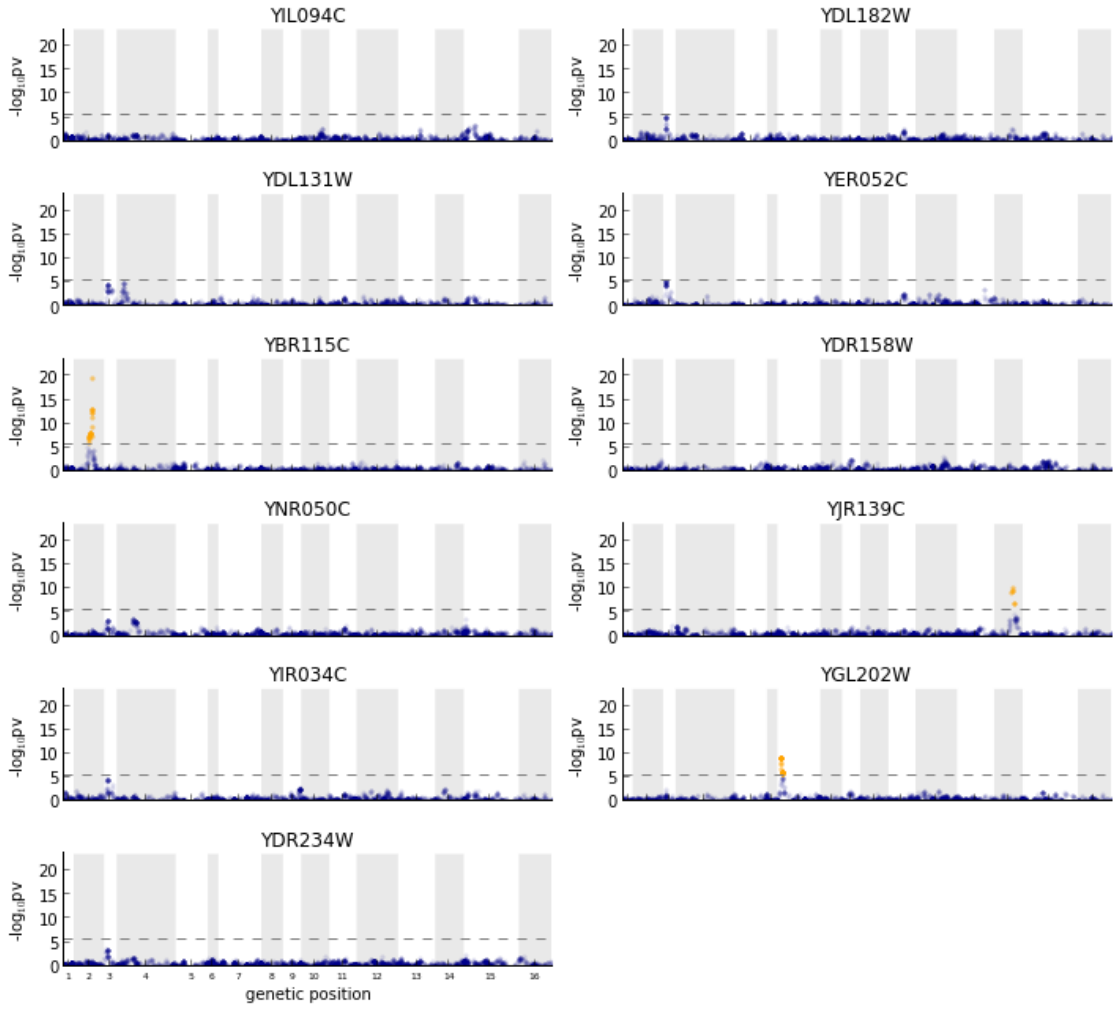
```
In [8]: # getting data
        Y = data.getPhenotypes(geneIDs='YBR115C',environments=0)[0]
        N,G = Y.shape

        # Running the analysis
        # when cov are not set, LIMIX considers an intercept (F=SP.ones((N,1)))
        lmm = QTL.simple_lmm(X,Y,K=R,covs=None)

        # get p-values
        pv = lmm.getPv()          # 1xS vector of p-values (S=X.shape[1])
        beta = lmm.getBetaSNP()   # 1xS vector of effect sizes (S=X.shape[1])

finished GWAS testing in 0.04 seconds

In [9]: plt = PL.subplot(2,1,1)
        plot_manhattan(plt,cumPos,pv[0,:],chromBounds)
```



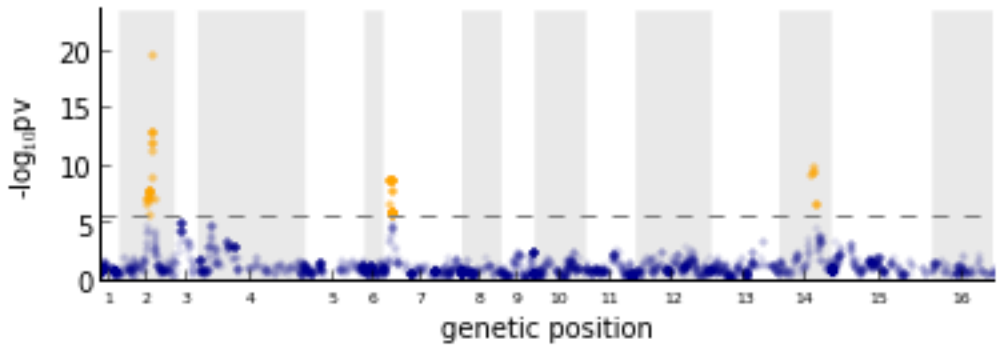
2.1.3 Example: Single-trait eQTL mapping for all genes in the pathways in environment 0

```
In [10]: # getting data
         Y = data.getPhenotypes(geneIDs=lysine_group,environments=0)[0]
         N,G = Y.shape

         # Running the analysis
         # when a matrix phenotype si given
         # LIMIX performs single-trait GWAS on all columns (traits) one-by-one
         lmm = QTL.simple_lmm(X,Y,K=R,covs=None)

         # get p-values
         pv = lmm.getPv()          # GxS vector of p-values (G=Y.shape[1]=#genes,S=X.shape[1],#SNPs)
         beta = lmm.getBetaSNP()   # GxS vector of effect sizes

finished GWAS testing in 0.27 seconds

In [11]: # plotting all manhattan plots
         PL.figure(1,figsize=(10,16))
         lim = -1.2*SP.log10(pv.min())
         for g in range(G):
```

```
        plt = PL.subplot(G,2,g+1)
        PL.title(lysine_group[g])
        if g!=G-1:   xticklabels = False
        else:        xticklabels = True
        plot_manhattan(plt,cumPos,pv[g,:],chromBounds,lim=lim,xticklabels=xticklabels)
    PL.tight_layout()
```



In [12]:
```
# plotting minimum pv across genes
pv_min = pv.min(0)
plt = PL.subplot(2,1,1)
plot_manhattan(plt,cumPos,pv_min,chromBounds)
```



2.2 Multi locus Single Trait Model

2.2.1 The Genetic Model

The model LIMIX emplys for standard GWAS can be extended to include multiple loci by step-wise forward selection. This is done by performing the following two steps: - a genome-wide scan is performed using the standard linear mixed model - if the most associated marker has P-value lower than a certain threshold the marker is added as covariate and the algorithm return to step 1, otherwise it stops here.

LIMIX allows the user to set - a threshold either over the P-value or the Q-value of the most associated SNP as stopping criterion; - a maximum number of iterations as stopping criterion;

2.2.2 Example: Single-trait eQTL mapping for gene YBR115C in environment 0

```
In [13]: # getting data
         Y = data.getPhenotypes(geneIDs='YBR115C',environments=0)[0]
         N,G = Y.shape

         # Running the analysis
         thr = 0.01/X.shape[1] # threshold on pvalues (Bonferroni corrected significance level of 1%)
         maxiter = 10 # maximum number of iterations
         lmm, RV = QTL.forward_lmm(X,Y,K=R,qvalues=False,threshold=thr,maxiter=maxiter)
         # RV contains:
         # - iadded:   array of indices of SNPs included in order of inclusion
         # - pvadded:  array of Pvalues obtained by the included SNPs in iteration before inclusion
         # - pvall:   [maxiter x S] SP.array of Pvalues for all iterations

finished GWAS testing in 0.04 seconds
finished GWAS testing in 0.03 seconds

In [14]: # TODO: plotting
```

2.3 Variance Component Models

Here we show how the LIMIX variance decomposition module can be used in genetic anlysis of variance for single trait analysis.

2.3.1 The Genetic Model

The model considered by the LIMIX variance decomposition module is an extension of the genetic model employed in standard GWAS in to include different random effects terms:

$$\mathbf{y} = \mathbf{F}\boldsymbol{\alpha} + \sum_x \mathbf{u}^{(x)} + \boldsymbol{\psi}, \quad \mathbf{u}^{(x)} \sim \mathcal{N}\left(\mathbf{0}, {\sigma^{(x)}}^2 \mathbf{R}^{(x)}\right), \ \boldsymbol{\Psi} \sim \mathcal{N}\left(\mathbf{0}, \sigma_e^2 \mathbf{I}_N\right) \tag{10}$$

where

$$\begin{aligned} \mathbf{y} &= \text{phenotype vector} \in \mathcal{R}^{N,1} && (11)\\ \mathbf{F} &= \text{matrix of } K \text{ covariates} \in \mathcal{R}^{N,K} && (12)\\ \boldsymbol{\alpha} &= \text{effect of covariates} \in \mathcal{R}^{K,1} && (13)\\ \mathbf{R}^{(x)} &= \text{is the sample covariance matrix for contribution } x \in \mathcal{R}^{N,N} && (14)\\ & && (15) \end{aligned}$$

If $\mathbf{R}$ is a genetic contribution from set of SNPs $\mathcal{S}$, with a bit of abuse of notation we can write $\mathbf{R} = \frac{1}{C}\mathbf{X}_{:,\mathcal{S}}\mathbf{X}_{:,\mathcal{S}}{}^T$ where $C = \frac{1}{N}\text{trace}\left(\mathbf{X}_{:,\mathcal{S}_i}\mathbf{X}_{:,\mathcal{S}_i}{}^T\right)$.

LIMIX allows the user to flexibly build the model by adding fixed and random effect terms.

2.3.2 Example: Cis/Trans Variance Decomposition in environment 0

Here we use the LIMIX variance decomposition module to quantify the variability in gene expression explained by proximal (cis) and distal (trans) genetic variation. To do so, we build a linear mixed model with a fixed effect intercept, two random effects for cis and trans genetic effects and a noise random effect:

$$\mathbf{y} = \mathbf{1}_N\mu + \mathbf{u}^{(cis)} + \mathbf{u}^{(trans)} + \boldsymbol{\psi}, \quad \mathbf{u}^{(cis)} \sim \mathcal{N}\left(\mathbf{0}, {\sigma^{(x)}}^2 \mathbf{R}^{(cis)}\right), \ \mathbf{u}^{(trans)} \sim \mathcal{N}\left(\mathbf{0}, {\sigma^{(x)}}^2 \mathbf{R}^{(trans)}\right), \ \boldsymbol{\Psi} \sim \mathcal{N}\left(\mathbf{0}, \sigma_e^2 \mathbf{I}_N\right) \tag{16}$$

where $\mathbf{R}^{(\text{cis})}$ and $\mathbf{R}^{(\text{trans})}$ are the local and distal relatedeness matrices, built considering all SNPs in cis and trans (i.e., not in cis) respectively. As cis region we considered a window of 500kb downstream and upstream the gene.

The gene-model is fitted to gene expression in environment 0 for all genes in the Lysine Biosynthesis pathway and variance components are averaged thereafter to obtain pathway based variance components.

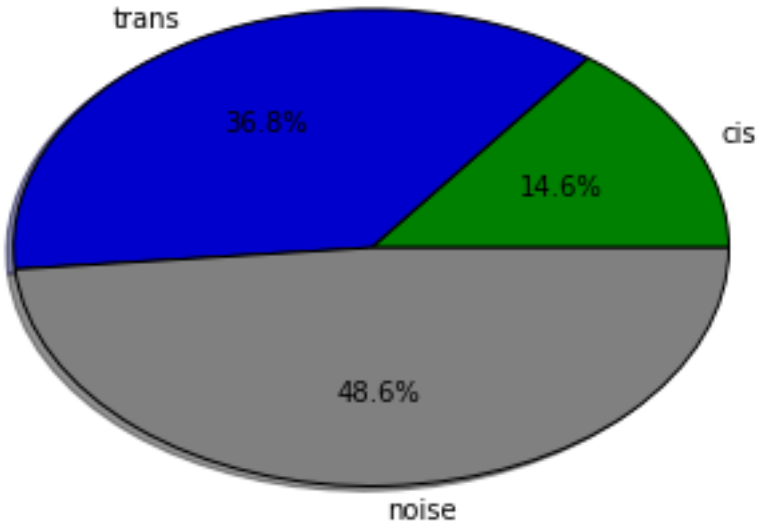
```
In [15]: # loop over genes and fit variance component model
         var = []
         G = len(lysine_group)
         for g in range(G):

             print '.. fit gene %d' % g

             #get phenotype for gene g and environment 0
             geneID = lysine_group[g]
             Y = data.getPhenotypes(geneIDs=geneID,environments=0)[0]

             #build kinships
             Icis = data.getIcis_geno(lysine_group[g],5e5)
             if Icis.sum()==0:       continue
             Xcis = X[:,Icis]
             Rcis    = SP.dot(Xcis,Xcis.T)
             Rtrans  = R0-Rcis # R0 = XX.T = non-normalized relatedness matrix
             Rcis   /= Rcis.diagonal().mean()
             Rtrans /= Rtrans.diagonal().mean()

             # variance component model
             vc = VAR.CVarianceDecomposition(Y)
```

```
            vc.addFixedEffect()
            vc.addRandomEffect(K=Rcis)
            vc.addRandomEffect(K=Rtrans)
            vc.addRandomEffect(is_noise=True)
            vc.findLocalOptimum()

            # get variances
            # vc.getVariances() returs a vector of variances explained
            # by the three random effects in order of addition (cis,trans,noise)
            _var = vc.getVariances()

            var.append(_var)

        var = SP.array(var)
```

```
.. fit gene 0
Local minimum found at iteration 0
.. fit gene 1
Local minimum found at iteration 0
.. fit gene 2
Local minimum found at iteration 0
.. fit gene 3
Local minimum found at iteration 0
.. fit gene 4
Local minimum found at iteration 0
.. fit gene 5
Local minimum found at iteration 0
.. fit gene 6
Local minimum found at iteration 0
.. fit gene 7
Local minimum found at iteration 1
.. fit gene 8
Local minimum found at iteration 0
.. fit gene 9
Local minimum found at iteration 0
.. fit gene 10
Local minimum found at iteration 0
```

In [16]:
```
#normalize variance component and average
var/=var.sum(1)[:,SP.newaxis]
var_mean = var.mean(0)
labels = ['cis','trans','noise']
colors = ['Green','MediumBlue','Gray']
PL.pie(var_mean,labels=labels,autopct='%1.1f%%',colors=colors,shadow=True, startangle=0)
```

Out[16]:
```
([<matplotlib.patches.Wedge at 0x1156fe610>,
  <matplotlib.patches.Wedge at 0x116398510>,
  <matplotlib.patches.Wedge at 0x1155ae410>],
 [<matplotlib.text.Text at 0x1163986d0>,
  <matplotlib.text.Text at 0x116398e10>,
  <matplotlib.text.Text at 0x1155aec10>],
 [<matplotlib.text.Text at 0x116398990>,
  <matplotlib.text.Text at 0x1155ae190>,
  <matplotlib.text.Text at 0x1155aec90>])
```



3 Multi Trait Analysis

Denoting with N and P the number of samples and phenotypes in the analysis, LIMIX models the $N \times P$ matrix-variate phenotype $\mathbf{Y}$ as a sum of fixed and random effects in a multivariate mixed model framework.

Single-trait models only consider the across-sample axis of variation and thus - fixed effects need only specification of a sample design (denoted by $\mathbf{F}$ so far) - random effects need only specification of a sample covariance matrix (denoted by $\mathbf{R}$ so far), which describes the correlation across samples induced by the random effect

Conversely, multi-trait models also consider the across-trait axis of variation and thus - fixed effects also require a sample design (denoted with $\mathbf{A}$ in the following) - random effects also require a trait covariance matrix (denoted with $\mathbf{C}$ in the following), which describes the correlation across traits induced by the random effect

Different forms of trait designs for fixed effects and trait covariances for random effects reflect different hypothesis on how these contributions affect phenotypes (e.g., shared and trait-specific effects). In the following, we will show how LIMIX can be used for different analysis of multiple traits wisely choosing trait designs and covariances.

We decided to start with multi-trait variance components models before going to multi-trait GWAS analysis, including single-locus and multi-locus analysis. In the next section we will be discussing more subtle problems as regularization and model selection.

3.1 Variance decomposition

3.1.1 Genetic Model

The general multi-trait linear mixed models considered in the LIMIX variance decomposition module can be written as

$$\mathbf{Y} = \sum_i \mathbf{F}_i \boldsymbol{W}_i \boldsymbol{A}_i^{(\mathrm{cov})} + \sum_x \mathbf{U}^{(x)} + \boldsymbol{\Psi}, \quad \mathbf{U}^{(x)} \sim \mathrm{MVN}\left(\mathbf{0}, \mathbf{R}^{(x)}, \mathbf{C}^{(x)}\right), \ \boldsymbol{\Psi} \sim \mathrm{MVN}\left(\mathbf{0}, \mathbf{I}_N, \mathbf{C}^{(\mathrm{noise})}\right) \quad (17)$$

where

$$\begin{aligned}
\mathbf{Y} &= \text{phenotype vector} \in \mathcal{R}^{N,P} && (18)\\
\mathbf{F}_i &= \text{sample designs for fixed effects} \in \mathcal{R}^{N,K} && (19)\\
\mathbf{W}_i &= \text{effect size matrix of covariates} \in \mathcal{R}^{K,L} && (20)\\
\mathbf{A}^{(\text{cov})} &= \text{trait design for the fixed effect} \in \mathcal{R}^{L,P} && (21)\\
\mathbf{R}^{(x)} &= \text{sample covariance matrix for contribution } x \in \mathcal{R}^{N,N} && (22)\\
\mathbf{C}^{(x)} &= \text{trait covariance matrix for contribution } x \in \mathcal{R}^{P,P} && (23)\\
\mathbf{C}^{(\text{noise})} &= \text{residual trait covariance matrix} \in \mathcal{R}^{P,P} && (24)\\
& && (25)
\end{aligned}$$

The variance decomposition module in LIMIX allows the user to build the model by adding any number of fixed and random effects, specify different designs for fixed effects and choose covariance models for random effects. In case of only two random effects LIMIX uses speedups beaking the $O(N^3P^3)$ complexity in $O(N^3 + P^3)$.

In the following we show two different example. In the first example we construct interpretable trait covariances for random effects and build a model that dissects gene expression variability across environmental and cis and trans genetic effects, and their interactions (cis GxE, trans GxE). In the second example we show how to use the fast implementation for a two-random-effect model, one for genetic effects and the other for residuals. Such a model is the natural extention to multi-trait analysis of the null model in single-trait GWAS and it is the first step of any multi-trait GWAS analysis performed in LIMIX.

3.1.2 Example 1: GxE variance decomposition for Lysine Biosynthesis

Indicating with N and $E = 2$ respectively the number of samples and the number of environments, we considered for each gene a model consisting of an envoronment-specific intercept term, a random effect for cis genetic effects, a random effect for trans genetic effects, and a noise random effect term.

The model can be written

$$\mathbf{Y} = \mathbf{FWA} + \mathbf{U}^{(\text{cis})} + \mathbf{U}^{(\text{trans})} + \mathbf{\Psi}, \quad \mathbf{U}^{(\text{cis})} \sim \text{MVN}(\mathbf{0}, \mathbf{R}^{(\text{cis})}, \mathbf{C}^{(\text{cis})}),\ \mathbf{U}^{(\text{trans})} \sim \text{MVN}(\mathbf{0}, \mathbf{R}^{(\text{trans})}, \mathbf{C}^{(\text{trans})}),\ \mathbf{\Psi} \sim \text{MVN}(\mathbf{0}, \mathbf{I}_N, \tag{26}$$

where $\mathbf{R}^{(\text{cis})}$ and $\mathbf{R}^{(\text{trans})}$ are the local and distal kinships, built considering all SNPs in cis and trans (i.e., not in cis) respectively. As in the single-trait analysis, we considered as cis region the region of 500kb downstream and upstream the gene.

To dissect persistent and specific variance compoents we considered a 'block+diagonal' form for $C^{(x)}$:

$$C^{(x)} = {a^{(x)}}^2 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} {c_1^{(x)}}^2 & 0 \\ 0 & {c_2^{(x)}}^2 \end{bmatrix} = \begin{bmatrix} {a^{(x)}}^2 + {c_1^{(x)}}^2 & {a^{(x)}}^2 \\ {a^{(x)}}^2 & {a^{(x)}}^2 + {c_2^{(x)}}^2 \end{bmatrix} \tag{27}$$

Variance of persistent and specific (GxE) effects from term x are ${a^{(x)}}^2$ and $\frac{1}{2}\left({c_1^{(x)}}^2 + {c_2^{(x)}}^2\right)$ while the variance explained by pure environmental shift can be calculated as $\text{var}\,(W)$.

```
In [17]: # loop over genes and fit variance component model
         var = []
         G = len(lysine_group)
         for g in range(G):

             print '.. fit gene %d' % g

             #get phenotype for gene g
             geneID = lysine_group[g]
```

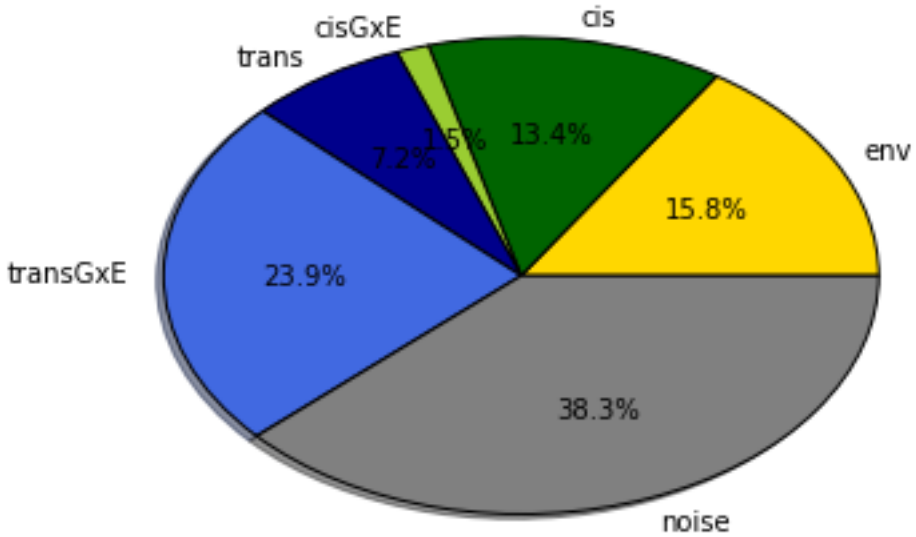
```
            Y = data.getPhenotypes(geneIDs=geneID,center=False,impute=False)[0]
            Y /= Y.std()

            #build kinships
            Icis = data.getIcis_geno(lysine_group[g],5e5)
            if Icis.sum()==0:      continue
            Xcis = X[:,Icis]
            Rcis    = SP.dot(Xcis,Xcis.T)
            Rtrans  = R0-Rcis # R0 = XX.T = non-normalized relatedness matrix
            Rcis   /= Rcis.diagonal().mean()
            Rtrans /= Rtrans.diagonal().mean()

            # variance component model
            vc = VAR.CVarianceDecomposition(Y)
            vc.addFixedEffect()
            vc.addRandomEffect(K=Rcis,covar_type='block_diag')
            vc.addRandomEffect(K=Rtrans,covar_type='block_diag')
            vc.addRandomEffect(is_noise=True,covar_type='block_diag')
            vc.findLocalOptimum()

            # environmental variance component
            vEnv = vc.getFixed().var()
            # cis and cisGxE variance components
            Ccis  = vc.getEstTraitCovar(0)
            a2cis = Ccis[0,1]
            c2cis = Ccis.diagonal().mean()-a2cis
            # trans and cisGxE variance components
            Ctrans  = vc.getEstTraitCovar(1)
            a2trans = Ctrans[0,1]
            c2trans = Ctrans.diagonal().mean()-a2trans
            # noise variance components
            Cnois = vc.getEstTraitCovar(2)
            vNois = Cnois.diagonal().mean()

            # Append
            var.append([vEnv,a2cis,c2cis,a2trans,c2trans,vNois])

        var = SP.array(var)
```

```
.. fit gene 0
Local minimum found at iteration 0
.. fit gene 1
Local minimum found at iteration 0
.. fit gene 2
Local minimum found at iteration 0
.. fit gene 3
Local minimum found at iteration 0
.. fit gene 4
Local minimum found at iteration 0
.. fit gene 5
Local minimum found at iteration 0
.. fit gene 6
Local minimum found at iteration 0
.. fit gene 7
```

```
Local minimum found at iteration 0
.. fit gene 8
Local minimum found at iteration 0
.. fit gene 9
Local minimum found at iteration 0
.. fit gene 10
Local minimum found at iteration 0
```

In [18]:
```
#normalize variance component and average
var/=var.sum(1)[:,SP.newaxis]
var_mean = var.mean(0)
labels = ['env','cis','cisGxE','trans','transGxE','noise']
colors = ['Gold','DarkGreen','YellowGreen','DarkBlue','RoyalBlue','Gray']
PL.pie(var_mean,labels=labels,autopct='%1.1f%%',colors=colors,shadow=True, startangle=0)
```

Out[18]:
```
([<matplotlib.patches.Wedge at 0x1156cc490>,
  <matplotlib.patches.Wedge at 0x11564c310>,
  <matplotlib.patches.Wedge at 0x11564c550>,
  <matplotlib.patches.Wedge at 0x114264650>,
  <matplotlib.patches.Wedge at 0x114268e90>,
  <matplotlib.patches.Wedge at 0x114268d10>],
 [<matplotlib.text.Text at 0x1156c8890>,
  <matplotlib.text.Text at 0x11564c250>,
  <matplotlib.text.Text at 0x114264550>,
  <matplotlib.text.Text at 0x114268650>,
  <matplotlib.text.Text at 0x114268990>,
  <matplotlib.text.Text at 0x1156e9650>],
 [<matplotlib.text.Text at 0x11564cf10>,
  <matplotlib.text.Text at 0x11564c890>,
  <matplotlib.text.Text at 0x114264a50>,
  <matplotlib.text.Text at 0x114268dd0>,
  <matplotlib.text.Text at 0x114268790>,
  <matplotlib.text.Text at 0x1156e97d0>])
```



3.1.3 Example 2: fast variance decomposition

In this example we consider a model with only two random effects: one genetic and a noise term and show how to perform fast inference to learn the trait covariance matrices.

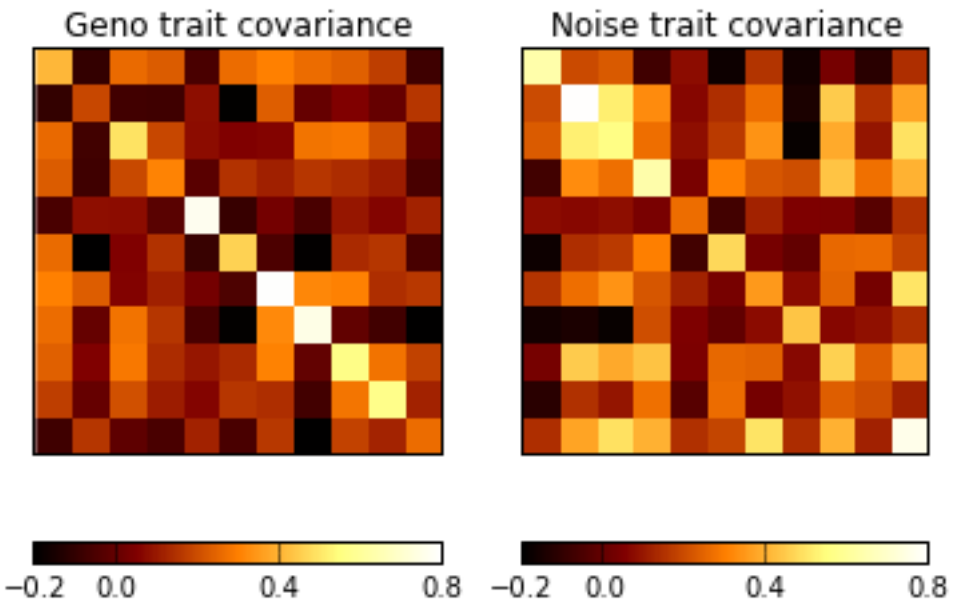
We apply the model to the expression in environoment 0 of the 11 genes of the lysine biosynthesis pathway.

```
In [19]: Y = data.getPhenotypes(geneIDs=lysine_group,environments=0)[0]
         # variance component model
         vc = VAR.CVarianceDecomposition(Y)
         vc.addFixedEffect()
         vc.addRandomEffect(K=R,covar_type='lowrank_diag',rank=4)
         vc.addRandomEffect(is_noise=True,covar_type='lowrank_diag',rank=4)
         vc.findLocalOptimum(fast=True,init_method='diagonal')
         # retrieve geno and noise covariance matrix
         Cg = vc.getEstTraitCovar(0)
         Cn = vc.getEstTraitCovar(1)
```

```
Local minimum found at iteration 0
```

```
In [22]: # plot trait covariance matrices
         plt = PL.subplot(1,2,1)
         PL.title('Geno trait covariance')
         PL.imshow(Cg,vmin=-0.2,vmax=0.8,interpolation='none',cmap=cm.afmhot)
         PL.colorbar(ticks=[-0.2,0.,0.4,0.8],orientation='horizontal')
         plt.set_xticks([])
         plt.set_yticks([])
         plt = PL.subplot(1,2,2)
         PL.title('Noise trait covariance')
         PL.imshow(Cn,vmin=-0.2,vmax=0.8,interpolation='none',cmap=cm.afmhot)
         PL.colorbar(ticks=[-0.2,0.,0.4,0.8],orientation='horizontal')
         plt.set_xticks([])
         plt.set_yticks([])
```

```
Out[22]: []
```



3.2 Multi Trait GWAS

The genetic model considered by LIMIX for multivariate GWAS consists of fixed effects for covariates, a fixed effect for the SNP being tested, a random effect for the polygenic effect and a noisy random effect term.

The model can be written

$$\mathbf{Y} = \sum_i \mathbf{F}_i \boldsymbol{W}_i \boldsymbol{A}_i^{(\text{cov})} + \mathbf{X}\mathbf{B}\mathbf{A}_1^{(\text{snp})} + \mathbf{G} + \boldsymbol{\Psi}, \qquad \mathbf{G} \sim \text{MVN}(\mathbf{0}, \mathbf{K}, \mathbf{C}_g),\ \boldsymbol{\Psi} \sim \text{MVN}(\mathbf{0}, \mathbf{I}_N, \mathbf{C}_n), \tag{28}$$

where

$$\begin{aligned}
\mathbf{Y} &= \text{matrix-variate phenotype} \in \mathcal{R}^{N,P} && (29)\\
\mathbf{F_i} &= \text{sample design for covariates} \in \mathcal{R}^{N,K} && (30)\\
\mathbf{W_i} &= \text{effect size matrix of covariates} \in \mathcal{R}^{K,L} && (31)\\
\mathbf{A_i}^{(\text{cov})} &= \text{trait design for the fixed effect} \in \mathcal{R}^{L,P} && (32)\\
\mathbf{X} &= \text{genetic profile of the SNP} \in \mathcal{R}^{N,1} && (33)\\
\mathbf{B} &= \text{effect sizes of the SNP} \in \mathcal{R}^{1,M} && (34)\\
\mathbf{A}_1^{(\text{snp})} &= \text{trait design for the fixed effect} \in \mathcal{R}^{M,P} && (35)\\
\mathbf{K} &= \text{sample relatedeness matrix} \in \mathcal{R}^{N,N} && (36)\\
\mathbf{C}_g &= \text{genetic trait covariance matrix} \in \mathcal{R}^{P,P} && (37)\\
\mathbf{I}_N &= \text{sample noise matrix} \in \mathcal{R}^{N,N} && (38)\\
\mathbf{C}_n &= \text{noise trait covariance matrix} \in \mathcal{R}^{P,P} && (39)
\end{aligned}$$

Prior to any multi-trait GWAS analysis, LIMIX learns the trait covariates matrices on the mixed model without the fixed effect from the SNP. However, LIMIX allows the user to set refitting of the signal-to-noise ratio δ SNP-by-SNP during the GWAS as in single-trait analysis.

In general, LIMIX tests for particular trait designs of the fixed effect $\mathbf{A}_1^{(\text{snp})} \neq \mathbf{A}_0^{(\text{snp})}$. As shown extensively below, specifying opportunately $\mathbf{A}_1^{(\text{snp})}$ and $\mathbf{A}_0^{(\text{snp})}$ different biological hypothesis can be tested.

3.2.1 Any effect test

Association between any of the phenotypes and the genetic marker can be tested by setting

$$\mathbf{A}_1^{(\text{snp})} = \mathbf{I}_P, \quad \mathbf{A}_0^{(\text{snp})} = \mathbf{0} \tag{40}$$

This is a P degrees of freedom test.

In this context, multi-trait modelling is considered just to empower detection of genetic loci while there is no interest in the specific design of the association.

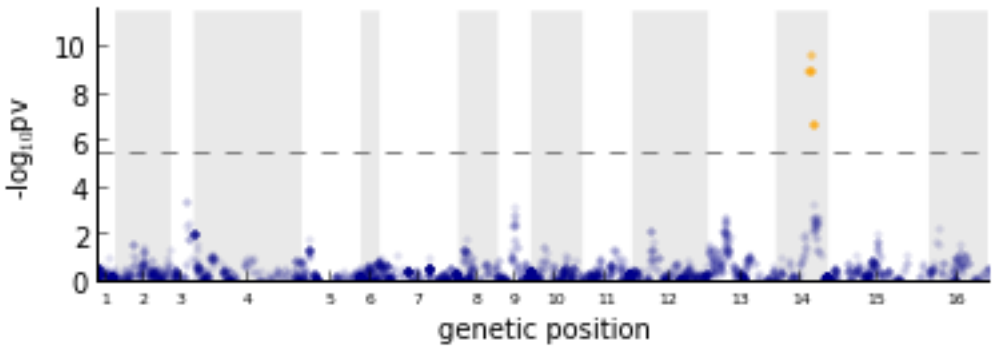
Example: any effect test for gene YJR139C across environments Here we test for genetic effects on expression of one gene either in environment 0 or in environment 1.

```
In [30]: # Get data
         Y = data.getPhenotypes(geneIDs='YJR139C')[0]
         N,P = Y.shape
         # Any effect test
         F    = SP.ones((N,1))
         Acov = SP.eye(P)
         Asnp = SP.eye(P)
         K1r  = R             # sample relatedness matrix
         ct   = 'freeform'    # model for trait covariance matrix
         lmm, pv = QTL.kronecker_lmm(X,Y,covs=F,Acovs=Acov,Asnps=Asnp,K1r=R,covar_type=ct)
```

```
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.05 s
```

REMARKS: - kronecker_lmm always considers Asnp0=0 thus only Asnp1 needs to be specified - if trait covariance matrices (arguments K1c and K2c are not specified) kronecker_lmm implements the variance decomposition module to estimate them. In this case, the user can specify the trait covariance model to considered (covar_type and rank).

```
In [31]: plt = PL.subplot(2,1,1)
         plot_manhattan(plt,cumPos,pv[0,:],chromBounds)
```



3.2.2 Common effect test

A common effect test is a 1 degree of freedom test and can be done by setting

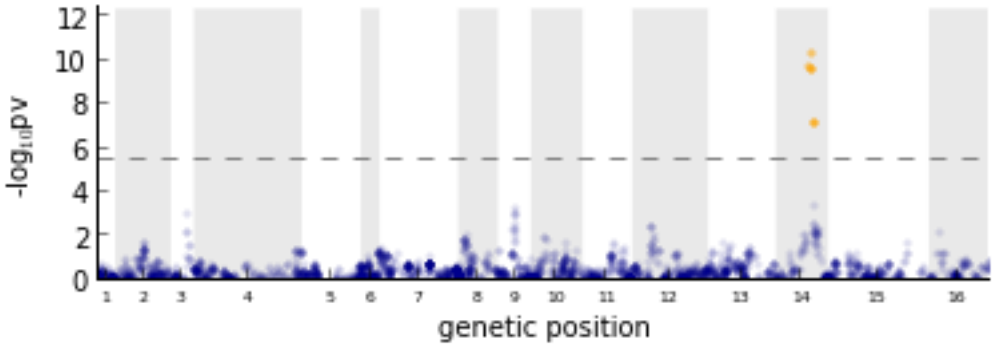
$$\mathbf{A}_1^{(\text{snp})} = \mathbf{1}_{1,P}, \quad \mathbf{A}_0^{(\text{snp})} = \mathbf{0} \tag{41}$$

Example: common effect test for gene YJR139C across environments

```
In [32]: # Get data
         Y = data.getPhenotypes(geneIDs='YJR139C')[0]
         N,P = Y.shape
         # Common effect test
         F    = SP.ones((N,1))
         Acov = SP.eye(P)
         Asnp = SP.ones((1,P))
         K1r  = R             # sample relatedness matrix
         ct   = 'freeform'    # model for trait covariance matrix
         lmm, pv = QTL.kronecker_lmm(X,Y,covs=F,Acovs=Acov,Asnps=Asnp,K1r=R,covar_type=ct)
```

```
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.09 s
```

```
In [43]: plt = PL.subplot(2,1,1)
         plot_manhattan(plt,cumPos,pv[0,:],chromBounds)
```



3.2.3 Specific effect test

For a specifc effect test for trait p the alternative model is set to have both a common and a specific effect for transcript p from the SNP while the null model has only a common effect.

It is a 1 degree of freedom test and, in the particular case of $P = 3$ traits and for $p = 0$, it can be done by setting

$$\mathbf{A}_1^{(\text{snp})} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad , \mathbf{A}_0^{(\text{snp})} = \mathbf{1}_{1,3} \tag{42}$$

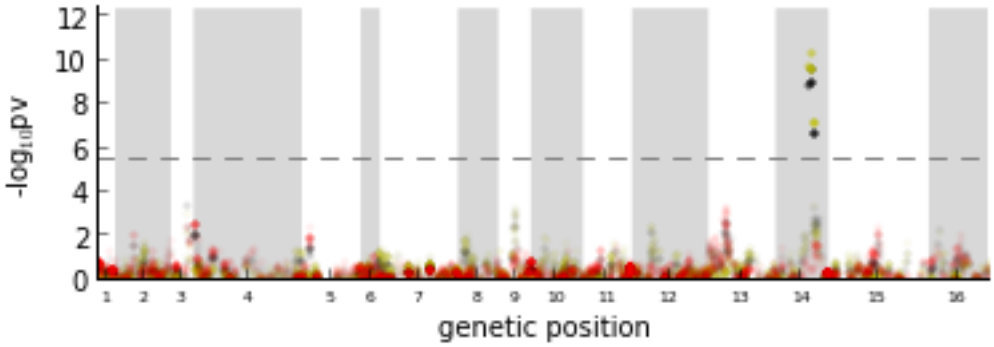
Example: specific effect test for gene YJR139C across environmentsExample

```
In [41]: Y = data.getPhenotypes(geneIDs='YJR139C')[0]
         # Any effect test
         F     = SP.ones((N,1))
         Acov  = SP.eye(P)
         Asnp  = SP.eye(P)
         Asnp1 = SP.eye(P)
         Asnp0 = SP.ones((1,P))
         K1r   = R             # sample relatedness matrix
         ct    = 'freeform'    # model for trait covariance matrix
         pv_spec,pv_comm,pv_any=QTL.simple_interaction_kronecker(X,Y,covs=F,Acovs=Acov,Asnps1=Asnp1,Asnp
```

```
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.05 s
```

Simple_interaction_kronecker not only compares the alternative model (where the trait design of the SNP is Asnp=Asnp1) versus the null model (where the trait design of the SNP is Asnp=Asnp0) but also compares the alternative model and the null models versus the no association model (Asnp=0). Three pvalues are then retured: - pv for alternative vs null (specific effect test) - pv for null vs noAssociation (common effect test) - pv for alternative vs noAssociation (any effect test)

```
In [53]: tests = ['Any effect test','Interaction effect test']
         lim = -1.2*SP.log10(SP.array([pv_any.min(),pv_comm.min(),pv_spec.min()]).min())
         plt = PL.subplot(2,1,1)
         plot_manhattan(plt,cumPos,pv_any[0,:],chromBounds,colorS='k',colorNS='k',lim=lim,alphaNS=0.05)
         plot_manhattan(plt,cumPos,pv_comm,chromBounds,colorS='y',colorNS='y',lim=lim,alphaNS=0.05)
         plot_manhattan(plt,cumPos,pv_spec[0,:],chromBounds,colorS='r',colorNS='r',lim=lim,alphaNS=0.05)
```



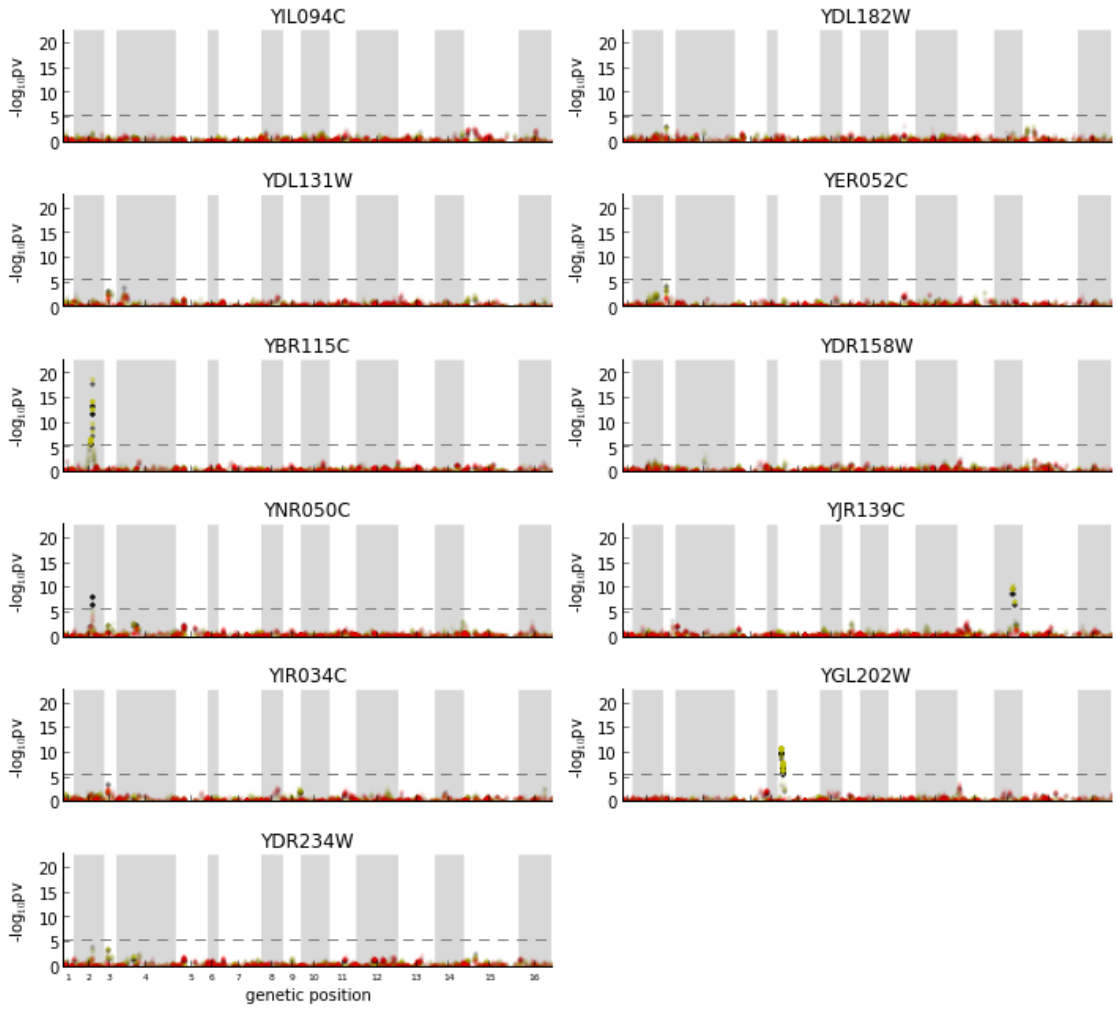
3.2.4 Example: any, common and specific effect tests across environments for all genes in the pathway

```
In [61]: G = len(lysine_group)
         pv_any  = SP.zeros((G,X.shape[1]))
         pv_comm = SP.zeros((G,X.shape[1]))
         pv_spec = SP.zeros((G,X.shape[1]))
         # for loop over genes
         for g in range(G):
             Y = data.getPhenotypes(geneIDs=lysine_group[g])[0]
             # any, common and specific effect tests for gene g
             _pv_spec,_pv_comm,_pv_any=QTL.simple_interaction_kronecker(X,Y,covs=SP.ones((N,1)),Acovs=SP
             pv_any[g,:]  = _pv_any[0,:]
             pv_comm[g,:] = _pv_comm
             pv_spec[g,:] = _pv_spec[0,:]
```

```
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.04 s
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.12 s
.. Training the backgrond covariance with a GP model
```

```
Local minimum found at iteration 0
Background model trained in 0.04 s
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.06 s
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.04 s
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.10 s
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.04 s
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.05 s
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.05 s
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.41 s
.. Training the backgrond covariance with a GP model
Local minimum found at iteration 0
Background model trained in 0.04 s
```

In [62]:
```
# plotting all manhattan plots
PL.figure(1,figsize=(10,16))
lim = -1.2*SP.log10(SP.array([pv_any.min(),pv_comm.min(),pv_spec.min()]).min())
for g in range(G):
    plt = PL.subplot(G,2,g+1)
    PL.title(lysine_group[g])
    if g!=G-1:   xticklabels = False
    else:        xticklabels = True
    plot_manhattan(plt,cumPos,pv_any[g,:],chromBounds,colorS='k',colorNS='k',lim=lim,alphaNS=0
    plot_manhattan(plt,cumPos,pv_comm[g,:],chromBounds,colorS='y',colorNS='y',lim=lim,alphaNS=
    plot_manhattan(plt,cumPos,pv_spec[g,:],chromBounds,colorS='r',colorNS='r',lim=lim,alphaNS=
PL.tight_layout()
```



All any-effect eQTL are explained by environmental persistent signal with the exception of the peak on chromosome 2 for gene YNR050C that has an almost significant specific component. These results together with those obtained from the GxE variance decomposition analysis suggest that the strong transGxE signal of the genes in the pathway is due to interaction of the poligenic effect with the environment.

3.3 Multi trait Multi locus

In [57]:

4 Predictions and Model Selection

In []:

5 PANAMA

In []: