Causal Model Selection Hypothesis Tests in Systems Genetics: a tutorial

Elias Chaibub Neto[*] and Brian S Yandell[†]

July 2, 2012

1 Motivation

Current efforts in systems genetics have focused on the development of statistical approaches that aim to disentangle causal relationships among molecular phenotypes in segregating populations. Model selection criterions, such as the AIC and BIC, have been widely used for this purpose, in spite of being unable to quantify the uncertainty associated with the model selection call. In this tutorial we illustrate the use of software implementing the causal model selection hypothesis tests proposed by Chaibub Neto et al. (2012).

2 Overview

This tutorial illustrates the basic functionality of the `cmst` R package in few simulated toy examples, and reproduces the analysis of a yeast genetical genomics data-set presented in Chaibub Neto et al. (2012). The `cmst` package builds over the `R/qtl` package (Broman et al. 2003), and we assume the reader is familiar with it.

3 Basic functionality

Here, we illustrate the basic functionality of the `cmst` package in a toy simulated example.

```
library(cmst)
```

We first use the `SimCrossCausal` function to simulate a `cross` object with 3 phenotypes, y_1, y_2 and y_3, where y_1 has a causal effect on both y_2 and y_3. The simulated cross data set, `Cross`, is composed of: 100 individuals (`n.ind = 100`); 3 chromosomes of length 100cM (`len = rep(100, 3)`); 101 unequally spaced markers per chromosome (`n.mar = 101` and `eq.spacing = FALSE`); additive genetic effect set to 1 (`add.eff = 1`); dominance genetic effect set to 0 (`dom.eff = 0`); residual variances for y_1 (`sig2.1`) and the other phenotypes (`sig2.2`) set to 0.4 and 0.1,

[*]Department of Computational Biology, Sage Bionetworks, Seattle WA
[†]Department of Statistics, University of Wisconsin-Madison, Madison WI

respectively; backcross cross type (`cross.type = "bc"`); and phenotype data transformed to normal scores (`normalize = TRUE`). The argument `beta = rep(0.5, 2)`, represents the causal effect of y_1 on the other phenotypes (i.e., coefficients of the regressions of $y_2 = 0.5\,y_1 + \epsilon$ and $y_3 = 0.5\,y_1 + \epsilon$). The length of beta controls the number of phenotypes to be simulated.

```
set.seed(987654321)
Cross <- SimCrossCausal(n.ind = 100,
                        len = rep(100, 3),
                        n.mar = 101,
                        beta = rep(0.5, 2),
                        add.eff = 1,
                        dom.eff = 0,
                        sig2.1 = 0.4,
                        sig2.2 = 0.1,
                        eq.spacing = FALSE,
                        cross.type = "bc",
                        normalize = TRUE)
```

We compute the genotype conditional probabilities using Haldane's map function, genotype error rate of 0.0001, and setting the maximum distance between positions at which genotype probabilities were calculated to 1cM.

```
Cross <- calc.genoprob(Cross, step = 1)
```

We perform QTL mapping using Haley-Knott regression (Haley and Knott 1992), and summarize the results for the 3 phenotypes. Figure 1 presents the LOD score profiles for all 3 phenotypes. The black, blue and red curves represent the LOD profiles of phenotypes y_1, y_2 and y_3, respectively.

```
Scan <- scanone(Cross, pheno.col = 1 : 3, method = "hk")

summary(Scan[, c(1, 2, 3)], thr = 3)
         chr  pos     y1
c1.loc55   1 55.0 12.618

summary(Scan[, c(1, 2, 4)], thr = 3)
         chr pos   y2
c1.loc55   1  55 5.27

summary(Scan[, c(1, 2, 5)], thr = 3)
      chr  pos   y3
D1M50   1 55.5 7.58

plot(Scan, lodcolumn = 1 : 3, ylab = "LOD")
```

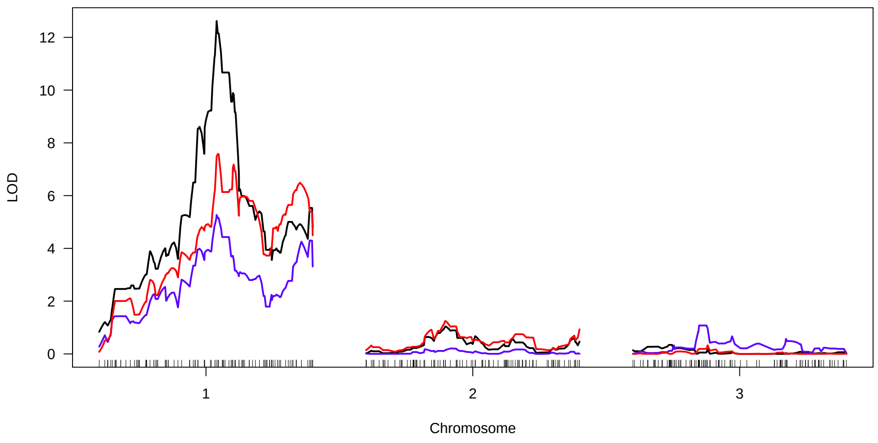

Figure 1: LOD score profiles for phenotypes y_1 (black curve), y_2 (blue curve) and y_3 (red curve).

Phenotypes y_1 and y_2 map to exactly same QTL at position 55 cM on chromosome 1. Phenotype y_3 maps to a QTL at position 55.5 cM. Whenever two phenotypes map to close, but not exactly identical, positions we are faced with the question of which QTL to use as causal anchor. Instead of making a (sometimes) arbitrary choice, our approach is to compute the joint LOD profile of both phenotypes and use the QTL detected by this joint mapping approach as the causal anchor. The function `GetCommonQtls` performs the joint QTL mapping for phenotypes whose marginal LOD peak positions are higher than a certain LOD threshold (`thr`), and are less than a fixed distance apart (`peak.dist`). The function can also handle separate additive and interacting covariates for each phenotype (`addcov1`, `intcov1`, `addcov2`, `intcov2`). In this simulated example the QTL detected by the joint analysis agreed with phenotype's y_1 QTL.

```
commqtls <- GetCommonQtls(Cross,
                          pheno1 = "y1",
                          pheno2 = "y3",
                          thr = 3,
                          peak.dist = 5,
                          addcov1 = NULL,
                          addcov2 = NULL,
                          intcov1 = NULL,
                          intcov2 = NULL)
commqtls
         Q Q.chr Q.pos
1 c1.loc55     1    55
```

Now, we fit our causal model selection tests for phenotypes y_1 and y_2 using the `CMSTtests` function. The `Q.chr` and `Q.pos` arguments specify the chromosome and position (in cM) of the

QTL to be used as a causal anchor. The argument `method` specify which version of the CMST test should be used. The options `"par"`, `"non.par"` and `"joint"` represent, respectively, the parametric, non-parametric, joint parametric versions of the CMST test. The option `"all"` fits all three versions. The `penalty` argument specifies whether we should test statistics based on the AIC (`"aic"`), BIC (`"bic"`), or both (`"both"`) penalties. In this particular call we computed all 3 versions using both penalties fitting 6 separate CMST tests.

```
nms <- names(Cross$pheno)
out1 <- CMSTtests(Cross,
                  pheno1 = nms[1],
                  pheno2 = nms[2],
                  Q.chr = 1,
                  Q.pos = 55,
                  addcov1 = NULL,
                  addcov2 = NULL,
                  intcov1 = NULL,
                  intcov2 = NULL,
                  cross.type = "bc",
                  method = "all",
                  penalty = "both")
```

The output of the `CMSTtests` function is composed of a list with 17 elements. It returns the names of the phenotypes and number of individuals (`n.ind`):

```
out1[1:3]
$pheno1
[1] "y1"

$pheno2
[1] "y2"

$n.ind
[1] 100
```

The log-likelihood scores (`loglik`) of models M_1, M_2, M_3, and M_4 (see Chaibub Neto et al. 2012 for details):

```
out1[4]
$loglik
[1] -123.5318 -140.4604 -141.5803 -123.4834
```

The dimensions of the models (`model.dim`):

```
out1[5]
$model.dim
[1] 6 6 6 7
```

The R^2 values (`R2`) relative to the regression of phenotypes 1 and 2 on the causal anchor:

```
out1[6]
$R2
[1] 0.4407170 0.2153583
```

The covariance matrix (`S.hat`) with the variances and covariances of the penalized log-likelihood ratios of models $M_1 \times M_2$, $M_1 \times M_3$, $M_1 \times M_4$, $M_2 \times M_3$, $M_2 \times M_4$, and $M_3 \times M_4$:

```
out1[7]
$S.hat
            [,1]        [,2]         [,3]         [,4]         [,5]        [,6]
[1,]  0.26221327 -0.01323094  0.010924311 -0.275444212 -0.251288963  0.02415525
[2,] -0.01323094  0.36275299  0.012080993  0.375983930  0.025311930 -0.35067200
[3,]  0.01092431  0.01208099  0.001115354  0.001156681 -0.009808958 -0.01096564
[4,] -0.27544421  0.37598393  0.001156681  0.651428142  0.276600893 -0.37482725
[5,] -0.25128896  0.02531193 -0.009808958  0.276600893  0.241480006 -0.03512089
[6,]  0.02415525 -0.35067200 -0.010965639 -0.374827248 -0.035120888  0.33970636
```

The BIC scores (`BICs`):

```
out1[8]
$BICs
[1] 274.6946 308.5518 310.7917 279.2030
```

The BIC-based penalized log-likelihood test statistics (`Z.bic`):

```
out1[9]
$Z.bic
     [,1]     [,2]      [,3]      [,4]
[1,]   NA 3.305926 2.9966507  6.749745
[2,]   NA       NA 0.1387598 -2.986200
[3,]   NA       NA        NA -2.709873
[4,]   NA       NA        NA        NA
```

The BIC-based model selection p-values for the parametric CMST (`pvals.p.BIC`), non-parametric CMST (`pvals.np.BIC`) and joint parametric CMST (`pvals.j.BIC`):

```
out1[10:12]
$pvals.p.BIC
[1] 0.001364817 0.999526684 0.998635183 1.000000000

$pvals.np.BIC
[1] 6.289575e-06 9.999977e-01 9.999999e-01 1.000000e+00

$pvals.j.BIC
[1] 0.003779558 0.999946885 0.999669186 1.000000000
```

The analogous AIC-based quantities:

```
out1[13:17]
$AICs
[1] 259.0636 292.9208 295.1606 260.9668

$Z.aic
     [,1]     [,2]      [,3]      [,4]
[1,]   NA 3.305926 2.9966507  2.849429
[2,]   NA       NA 0.1387598 -3.251273
[3,]   NA       NA        NA -2.933361
[4,]   NA       NA        NA        NA

$pvals.p.AIC
[1] 0.002189889 0.999526684 0.998635183 0.997810111

$pvals.np.AIC
[1] 6.289575e-06 9.999977e-01 1.000000e+00 9.999977e-01

$pvals.j.AIC
[1] 0.005993868 0.999946885 0.999669186 1.000000000
```

The `cmst` package also provides the function `CMSTtestsList` that computes CMST tests of a single phenotype against a list of phenotypes. Its output is less detailed though. In this particular call we test y_1 against y_2 and y_3.

```
out2 <- CMSTtestsList(Cross,
                      pheno1 = nms[1],
                      phenos = nms[-1],
                      Q.chr = 1,
                      Q.pos = 55.5,
                      addcov1 = NULL,
                      addcov2 = NULL,
                      intcov1 = NULL,
                      intcov2 = NULL,
                      cross.type = "bc",
                      method = "all",
                      penalty = "both")
out2
$R2s
      R2.Y1 ~ Q R2.Y2 ~ Q
y1_y2 0.4286585 0.2112760
y1_y3 0.4286585 0.2945801

$AIC.stats
```

```
         AIC.1    AIC.2    AIC.3    AIC.4     z.12     z.13      z.14      z.23
y1_y2 261.1967 293.4397 297.8127 263.0819 3.136952 3.034372 2.6436961 0.2659898
y1_y3 256.9466 278.0272 311.4368 258.2783 2.177343 3.876750 0.8229369 2.0030490
           z.24      z.34
y1_y2 -3.084095 -2.975873
y1_y3 -2.329987 -4.023391

$BIC.stats
         BIC.1    BIC.2    BIC.3    BIC.4     z.12     z.13     z.14      z.23
y1_y2 276.8278 309.0707 313.4437 281.3181 3.136952 3.034372 6.297065 0.2659898
y1_y3 272.5777 293.6583 327.0678 276.5145 2.177343 3.876750 2.432884 2.0030490
           z.24      z.34
y1_y2 -2.819431 -2.752652
y1_y3 -2.022629 -3.826214

$pvals.j.BIC
           pval.1    pval.2    pval.3 pval.4
y1_y2 0.003366003 0.9998807 0.9997011      1
y1_y3 0.035842249 0.9974598 0.9999899      1

$pvals.p.BIC
           pval.1    pval.2    pval.3    pval.4
y1_y2 0.001205187 0.9991464 0.9987948 1.0000000
y1_y3 0.014727493 0.9852725 0.9999471 0.9925105

$pvals.np.BIC
            pval.1    pval.2 pval.3    pval.4
y1_y2 2.346206e-06 0.9999992      1 1.0000000
y1_y3 1.758821e-03 0.9991050      1 0.9999607

$pvals.j.AIC
          pval.1    pval.2    pval.3 pval.4
y1_y2 0.01109575 0.9998807 0.9997011      1
y1_y3 0.38662989 0.9985143 0.9999950      1

$pvals.p.AIC
           pval.1    pval.2    pval.3    pval.4
y1_y2 0.004100312 0.9991464 0.9987948 0.9958997
y1_y3 0.205271925 0.9900966 0.9999713 0.7947281

$pvals.np.AIC
            pval.1    pval.2 pval.3    pval.4
y1_y2 1.608001e-05 0.9999992      1 0.9999937
y1_y3 4.431304e-02 0.9991050      1 0.9715560
```

4 Yeast knock-out data analysis

Here we reproduce the analysis of the budding yeast genetical genomics data-set presented in Chaibub Neto et al. (2012). The data represents a cross of a standard yeast laboratory strain, and a wild isolate from a California vineyard (Brem and Kruglyak 2005). It consists of expression measurements on 5,740 transcripts measured on 112 segregant strains with dense genotype data on 2,956 markers. Processing of the expression measurements raw data was done as described in Brem and Kruglyak (2005), with an additional step of converting the processed measurements to normal quantiles by the transformation $\Phi^{-1}[(r_i - 0.5)/112]$, where Φ is the standard normal cumulative density function, and the r_i are the ranks.

In order to evaluate the precision of the causal predictions made by the methods we used validated causal relationships extracted from a data-base of 247 knock-out experiments in yeast (Hughes et al. 2000, Zhu et al. 2008). In each of these experiments, one gene was knocked-out, and the expression levels of the remainder genes in control and knocked-out strains were interrogated for differential expression. The set of differentially expressed genes form the knock-out signature (ko-signature) of the knocked-out gene (ko-gene), and show direct evidence of a causal effect of the ko-gene on the ko-signature genes.

We first load the yeast cross object (`yeast.orf`), and compute the conditional genotype probabilities using Haldane's map function, genotype error rate of 0.0001, and setting the maximum distance between positions at which genotype probabilities were calculated to 2cM.

```
load("yeast_orf_ns_cross.RData")
yeast.orf <- calc.genoprob(yeast.orf, step = 2)
```

Next we count the number of missing phenotype observations for each one of the 5,740 transcripts (`na.counts`), and record the indexes of the phenotype with no missing values (`no.na.pos`), and phenotypes with missing values (`na.pos`).

```
na.counts <- apply(apply(yeast.orf$pheno, 2, is.na), 2, sum)
no.na.pos <- which(na.counts == 0)
na.pos <- which(na.counts > 0)
```

We perform QTL-mapping using Haley-Knott regression (Haley and Knott 1992) in two steps. First, we handle the phenotypes without missing values as a single block (`no.na.scan`). Next, we run separate analysis for each of the phenotypes with missing data. Finally, we join and save the results of both analyzes in a single `scanone` object called `scan`. This step takes a few minutes to run. Hence, for the sake of time we prefer to skip it, and we simply load the pre-computed results.

```
########## don't run
no.na.scan <- scanone(yeast.orf, pheno.col = no.na.pos, method = "hk")
na.scan <- matrix(0, nrow(no.na.scan), length(na.pos))
dimnames(na.scan) <- list(row.names(no.na.scan), names(na.pos))
for (i in 1 : length(na.pos)) {
  na.scan[, i] <- scanone(yeast.orf, pheno.col = na.pos[i], method = "hk")[, 3]
  cat("trait ", i, "\n")
}
na.scan <- data.frame(na.scan)
names(na.scan) <- names(na.pos)
scan <- data.frame(no.na.scan, na.scan)
class(scan) <- c("scanone", "data.frame")
save(scan, file = "scan_orf_ns.RData", compress = TRUE)
##########

load("scan_orf_ns.RData")
```

Next, we load a yeast annotation data.frame, `yeast.annot`, that provides the orf, gene symbol, and chromosome location (in both Mb and cM) of each one of the 5,740 transcripts. (This information will be needed to determine which ko-genes show significant QTLs.)

```
load("yeast_annot_Mb_cM.RData")
head(yeast.annot)
         orf  gene chr   Mb.pos   cM.pos
3952 YAL001C  TFC3   1 0.151168 102.4066
3951 YAL002W  VPS8   1 0.143709 101.3745
3950 YAL003W  EFB1   1 0.142176 101.1623
1330 YAL005C  SSA1   1 0.141433 101.0595
3934 YAL007C  ERP2   1 0.138347 100.1245
3933 YAL008W FUN14   1 0.136916 100.1245
```

Next, we load the list of ko-signatures derived from the knock-out experiments in Hughes et al. (2000) and Zhu et al. (2008). We show below the first knock-out signature.

```
load("ko_list_all_orf.RData")
length(ko.list.all)
[1] 247
ko.list.all[1]
$YOR128C
 [1] "YAR073W"   "YBL013W"   "YBL032W"   "YBL042C"   "YBL054W"   "YBL064C"
 [7] "YBR013C"   "YBR054W"   "YBR072W"   "YBR126C"   "YBR155W"   "YBR186W"
[13] "YCL030C"   "YDL038C"   "YDL234C"   "YDL244W"   "YDR001C"   "YDR018C"
[19] "YDR055W"   "YDR077W"   "YDR085C"   "YDR399W"   "YDR518W"   "YDR533C"
[25] "YDR534C"   "YER055C"   "YER062C"   "YFL014W"   "YFL030W"   "YFL058W"
[31] "YGL156W"   "YGL162W"   "YGL187C"   "YGL234W"   "YGR032W"   "YGR043C"
[37] "YGR138C"   "YGR161C"   "YGR171C"   "YGR213C"   "YGR250C"   "YHL040C"
[43] "YHR087W"   "YHR096C"   "YHR104W"   "YHR216W"   "YIL125W"   "YJL034W"
[49] "YJL054W"   "YJL116C"   "YJR151C"   "YKL029C"   "YKL090W"   "YKL097W.A"
[55] "YKL163W"   "YKL165C"   "YKR061W"   "YLL019C"   "YLL060C"   "YLR120C"
[61] "YLR121C"   "YLR142W"   "YLR178C"   "YLR194C"   "YLR350W"   "YLR359W"
[67] "YML130C"   "YML131W"   "YMR040W"   "YMR090W"   "YMR173W"   "YMR181C"
```

```
[73] "YMR300C"   "YNL112W"   "YNL134C"   "YNL160W"   "YNL220W"   "YOL151W"
[79] "YOL031C"   "YOR173W"   "YOR289W"   "YOR338W"   "YOR382W"   "YPL088W"
[85] "YPL277C"   "YPR156C"   "YAR075W"   "YDR243C"   "YFR024C.A" "YOL053C.A"
```

Next, we determine which of the 247 ko-genes also showed a significant QTL in our data set, according to a permutation test (Churchill and Doerge 1994) aiming to control GWER < 0.05. For each one of the ko-genes with a significant QTL, that is, with LOD score above `lod.thr` = 3.47, the function `GetCandReg` returns the ko-gene's chromosome (`phys.chr`) and physical position in cM (`phys.pos`), as well as, the LOD score (`peak.lod`) at the peak position (`peak.pos`), and the chromosome where the peak is located (`peak.chr`). In total, we observed 135 ko-genes with significant QTLs. These ko-genes are our candidate regulators. We show below the information on the first 10 candidate regulators. Note that some ko-genes map to the same chromosome where they are physically located, while other map to different chromosomes.

```
cand.reg <- GetCandReg(scan = scan,
                       annot = yeast.annot,
                       traits = names(ko.list.all),
                       lod.thr = 3.47,
                       drop = 1.5)
dim(cand.reg)
[1] 135   6
cand.reg[1:10,]
      gene phys.chr phys.pos peak.chr    peak.pos peak.lod
2  YMR282C       13 473.2316       14 236.0138450 3.692560
3  YER017C        5 152.3216       14 238.0138450 6.597231
7  YER069W        5 211.7280        3  54.0140660 3.975861
9  YOR058C       15 188.5460        8   0.9067482 3.569372
10 YGL017W        7 227.0394        7 221.6439074 5.894020
14 YMR055C       13 235.2625       13 246.0276440 5.578000
16 YMR275C       13 467.3183       13 460.0276440 5.508846
18 YER061C        5 203.6281       13  23.4283870 3.713512
19 YOR028C       15 176.6140       15  61.1420708 5.408553
24 YGR109C        7 339.0973        2 236.1133803 5.299293
```

Genes that map to positions close to their physical locations are said to map in *cis* (local-linkages). Genes that map to positions away from their physical locations are said to map in *trans* (distal-linkages). There is no unambiguous way the determine how close a gene needs to map to its physical location in order to be classified as cis. Our choice is to classify a gene as cis if the 1.5-LOD support interval (Manichaikul et al. 2006) around the LOD peak contains the gene's physical location, and if the LOD score at its physical location is higher the the LOD threshold. The function `GetCisCandReg` determines which of the candidate regulators map in cis. We see that only 27, out of the 135 candidate regulators, show cis-linkages. (The additional columns `peak.pos.lower` and `peak.pos.upper` show, respectively, the lower and upper bounds of the 1.5-LOD support interval around `peak.pos`.)

```
cis.cand.reg <- GetCisCandReg(cand.reg, scan, drop = 1.5, lod.thr = 3.47)
dim(cis.cand.reg[[1]])
[1] 27  8
cis.cand.reg[[1]][1:10,]
       gene phys.chr phys.pos peak.chr peak.pos peak.lod peak.pos.lower
10  YGL017W        7 227.0394        7 221.6439 5.894020       210.0037
14  YMR055C       13 235.2625       13 246.0276 5.578000       144.0276
16  YMR275C       13 467.3183       13 460.0276 5.508846       420.0276
48  YLR342W       12 402.5087       12 402.5087 8.666742       400.0196
61  YNL021W       14 278.5199       14 242.0138 4.359721       216.0138
63  YOR038C       15 179.1709       15 174.0012 5.060326       154.0012
77  YGR040W        7 251.4701        7 246.2146 8.558638       234.0037
79  YJL030W       10 157.0201       10 157.0890 8.976738       154.8556
97  YKL043W       11 151.0933       11 152.0006 9.323551       144.0006
101 YDR004W        4 237.5420        4 238.0177 4.574174       222.0177
    peak.pos.upper
10        230.7461
14        260.0276
16        472.0276
48        410.0196
61        284.0138
63        184.9256
77        254.0037
79        162.0222
97        158.0006
101       248.0177
```

For each one of the 135 candidate ko-genes, we determined which other genes also co-mapped to the same QTL of the ko-gene. The co-mapping genes represent the putative targets of a ko-gene. The function `GetCoMappingTraits` returns a list with the putative targets of each ko-gene. A gene is included in the putative target list of a ko-gene when its LOD peak is greater than `lod.thr` and the 1-5 LOD support interval around the peak contains the location of the ko-gene's QTL. The number of targets vary from ko-gene to ko-gene (from 1 to 570), and we show below the putative targets of one ko-gene (YMR275C) with 4 putative targets. In total, the 135 candidate regulators have 31,936 targets.

```
comap.targets <- GetCoMappingTraits(traits = cand.reg,
                                    scan = scan,
                                    lod.thr = 3.47,
                                    drop = 1.5)
summary(unlist(lapply(comap.targets, length)))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0    63.5   188.0   236.6   480.0   570.0
comap.targets[7]
```

```
$YMR275C
[1] "YGL254W" "YML069W" "YMR247C" "YDL013W"
length(unlist(comap.targets))
[1] 31936
```

Next, we use the function `FitAllTests` to fit the causality tests of each candidate regulator ko-gene (`pheno1`) to its putative targets (`phenos`). We use the candidate regulator's QTL (`Q.chr` and `Q.pos`) as a causal anchor. This function fits: the AIC and BIC model selection criterions (Schadt et al. 2005); the AIC- and BIC-based versions of the joint, parametric and non-parametric CMST tests (Chaibub Neto et al. 2012); and the CIT test (Millstein et al. 2009). We do not run it here because this step can take a few hours, as we perform a total of 31,936 tests for each of the 9 approaches. The function `JoinKoOutputs` joins together the outputs of the 135 separate fits of the `FitAllTests` function.

```
###### don't run
set.seed(123456789) # we fix a seed because cit uses bootstrap
for (k in 1 : 135) {
  cat("trait=", k, "\n")
  out <- FitAllTests(cross = yeast.orf,
                     pheno1 = cand.reg[k, 1],
                     phenos = comap.targets[[k]],
                     Q.chr = cand.reg[k, 4],
                     Q.pos = cand.reg[k, 5])
  save(out, file=paste("output_ko_validation", cand.reg[k, 1], "RData",
       sep = "."), compress = TRUE)
}
######
JoinKoOutputs(x = comap.targets)
```

After loading the joined results we use the function `PrecTpFpMatrix` to summarize the performance of the different methods in terms of "biologically validated" true positives, false positives and precision, of the inferred causal relations. Since we already have the results of the knock-out experiments (recall that `ko.list.all` holds the ko-signatures of the ko-genes), we define a true positive as a statistically significant causal relation between a ko-gene and a putative target gene, when the putative target gene belongs to the ko-signature of the ko-gene. Similarly, we define a false positive as a statistically significant causal relation between a ko-gene and a putative target gene when the target gene doesn't belong to the ko-signature. (For the AIC and BIC methods, that do not provide a p-value measuring the significance of the causal call, we simply use the detected causal relations in the computation of true and false positives). The "validated precision", is computed as the ratio of true positives by the sum of true and false positives. The `PrecTpFpMatrix` computes these measures to both all ko-genes, and to cis ko-genes only. The argument `alpha` sets the significant levels at each the summaries are computed.

```
load("joined_ko_output.RData")
```

```
aux <- PrecTpFpMatrix(alpha = seq(0.01, 0.10, by = 0.01),
                      nms = cand.reg[, 1],
                      val.targets = ko.list.all,
                      all.orfs = names(yeast.orf$pheno),
                      to.load = "joined_ko_output.RData",
                      cis.index = cis.cand.reg[[2]])
Prec1 <- aux[[1]]
Prec2 <- aux[[2]]
Tp1 <- aux[[3]]
Tp2 <- aux[[4]]
Fp1 <- aux[[5]]
Fp2 <- aux[[6]]
```

Below we reproduce Figure 5 of Chaibub Neto et al. (2012). This figure presents the number of inferred true positives, number of inferred false positives and the prediction precision across varying significance levels for each one of the methods. The results were computed using all 135 ko-gene/putative target lists.

```
lwd <- 2
xaxis <- seq(0.01, 0.10, by=0.01)
my.lty <- c(rep(1, 4), rep(2, 4), 1)
my.lty <- rep(1, 9)
my.pch <- c(1, 21, 24, 23, 25, 2, 5, 6, 8)
par(mfrow=c(1, 3))
par(mar=c(5, 4.1, 4, 2) + 0.1)
ymax <- max(Tp1)
yaxis <- seq(0, ymax,length.out = length(xaxis))
plot(xaxis, yaxis, type = "n", ylab = "Number of true positives", cex = 1.5,
     xlab = "Target significance level", cex.axis = 1.5,
     cex.lab = 1.7, main = "(a)", cex.main = 2)
for (k in 1 : 9) {
  lines(xaxis, Tp1[k,], type="b", lwd=lwd, pch=my.pch[k], cex=1.5,
        col = "black", bg = "black")
}
ymax <- max(Fp1)
yaxis <- seq(0, ymax, length.out = length(xaxis))
plot(xaxis, yaxis, type = "n", ylab = "Number of false positives", cex = 1.5,
     xlab = "Target significance level", cex.axis = 1.5,
     cex.lab = 1.7, main = "(b)", cex.main = 2)
for (k in 1 : 9) {
  lines(xaxis, Fp1[k,], type = "b", lwd = lwd, pch = my.pch[k], cex = 1.5,
        col = "black", bg = "black")
}
ymax <- max(Prec1)
```

```
yaxis <- seq(0, ymax, length.out = length(xaxis))
plot(xaxis, yaxis, type = "n", ylab = "Precision", cex = 1.5,
     xlab = "Target significance level", cex.axis = 1.5,
     cex.lab = 1.7, main = "(c)", cex.main = 2)
for (k in 1 : 9) {
  lines(xaxis, Prec1[k,], type = "b", lwd = lwd, pch = my.pch[k], cex = 1.5,
        col = "black", bg = "black")
}
```

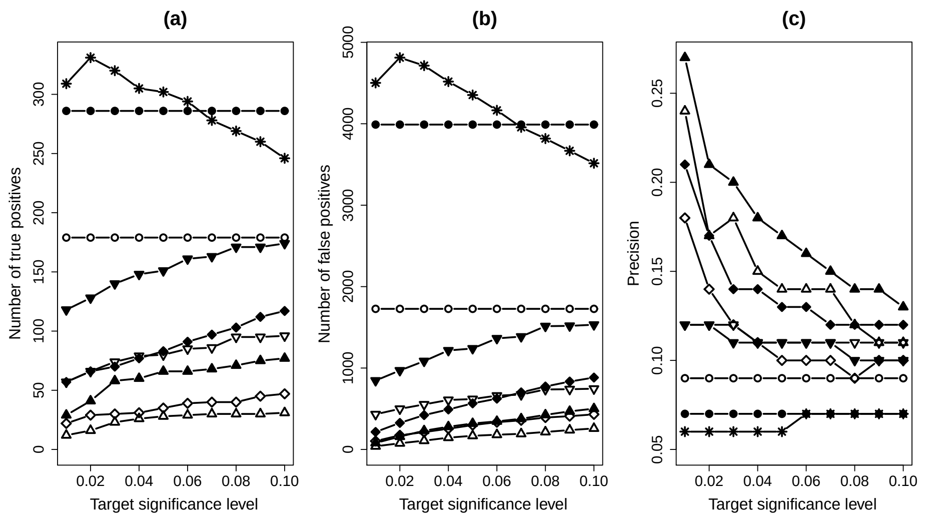

Figure 2: Reproduction of Figure 5 on Chaibub Neto et al. 2012. Overall number of true positives, number of false positives and precision across all 135 ko-gene/putative target lists. Asterisk represents the CIT. Empty and filled symbols represent, respectively, AIC- and BIC-based methods. Diamonds: parametric CMST. Point-down triangles: non-parametric CMST. Point-up triangles: joint-parametric CMST. Circles: AIC and BIC.

Next, we reproduce Figure 6 of Chaibub Neto et al. (2012). This figure was generated using the results of the 27 cis ko-gene/putative targets lists.

```
ymax <- max(Tp2)
yaxis <- seq(0, ymax, length.out = length(xaxis))
plot(xaxis, yaxis, type = "n", ylab = "Number of true positives", cex = 1.5,
     xlab = "Target significance level", cex.axis = 1.5,
     cex.lab = 1.7, main = "(a)", cex.main = 2)
for (k in 1 : 9) {
  lines(xaxis, Tp2[k,], type = "b", lwd = lwd, pch = my.pch[k], cex = 1.5,
        col = "black", bg = "black")
}
ymax <- max(Fp2)
```

```
yaxis <- seq(0, ymax, length.out = length(xaxis))
plot(xaxis, yaxis, type = "n", ylab = "Number of false positives", cex = 1.5,
     xlab = "Target significance level", cex.axis = 1.5,
     cex.lab = 1.7, main = "(b)", cex.main = 2)
for (k in 1 : 9) {
  lines(xaxis, Fp2[k,], type = "b", lwd = lwd, pch = my.pch[k], cex = 1.5,
        col = "black", bg = "black")
}
ymax <- max(Prec2)
yaxis <- seq(0, ymax, length.out = length(xaxis))
plot(xaxis, yaxis, type = "n", ylab = "Precision", cex = 1.5,
     ylim = c(0.18, 0.6), xlab = "Target significance level", cex.axis = 1.5,
     cex.lab = 1.7, main = "(c)", cex.main = 2)
for (k in 1 : 9) {
  lines(xaxis, Prec2[k,], type = "b", lwd = lwd, pch = my.pch[k], cex = 1.5,
        col = "black", bg = "black")
}
```

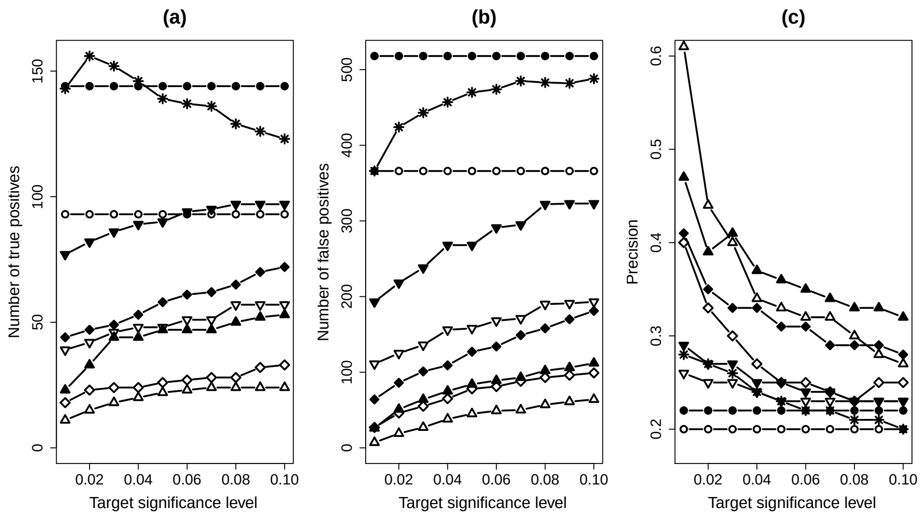

Figure 3: Reproduction of Figure 6 on Chaibub Neto et al. 2012. Overall number of true positives, number of false positives and precision restricted to 27 cis ko-gene/putative target lists. Asterisk represents the CIT. Empty and filled symbols represent, respectively, AIC- and BIC-based methods. Diamonds: parametric CMST. Point-down triangles: non-parametric CMST. Point-up triangles: joint-parametric CMST. Circles: AIC and BIC.

5 References

1. Brem R., L. Kruglyak, 2005 The landscape of genetic complexity across 5,700 gene expression trait in yeast. PNAS **102:** 1572-1577.

2. Broman K., H. Wu, S. Sen, G. A. Churchill, 2003 R/qtl: QTL mapping in experimental crosses. Bioinformatics **19**: 889-890.

3. Chaibub Neto et al. (2012) Causal model selection hypothesis tests in systems genetics. Genetics (under review)

4. Churchill G. A., R. W. Doerge, 1994 Empirical threshold values for quantitative trait mapping. Genetics **138**: 963-971.

5. Haley C., S. Knott, 1992 A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. Heredity **69**: 315-324.

6. Hughes T. R., M. J. Marton, A. R. Jones, C. J. Roberts, R. Stoughton, et al, 2000 Functional discovery via a compendium of expression profiles. Cell **102:** 109-116.

7. Manichaikul A., J. Dupuis, S. Sen, and K. W. Broman, 2006 Poor performance of bootstrap confidence intervals for the location of a quantitative trait locus. Genetics **174:** 481-489.

8. Schadt E. E., J. Lamb, X. Yang, J. Zhu, S. Edwards, et al., 2005 An integrative genomics approach to infer causal associations between gene expression and disease. Nature Genetics **37**: 710-717.

9. Zhu J., B. Zhang, E. N. Smith, B. Drees, R. B. Brem, L. Kruglyak, R. E. Bumgarner, E. E. Schadt, 2008 Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks. Nature Genetics **40**: 854-861.