Performance comparison of qpgraph with qtlhot

(code reproducing part of the results in *Mapping eQTL networks with mixed graphical Markov models*)

Inma Tur[1,2], Alberto Roverato[3], Robert Castelo[1,2,*]

September 16, 2014

[1]Dept. of Experimental and Health Sciences, Universitat Pompeu Fabra, Barcelona, Spain.

[2]Research Program on Biomedical Informatics, Institut de Recerca Hospital del Mar, Barcelona, Spain.

[3]Dept. of Statistical Sciences, Università di Bologna, Italy.

This vignette runs the Causal Model Selection Hypothesis Tests (CMSTs) introduced by Chaibub Neto et al. (2013) and implemented in the qtlhot package, on the yeast cross data from Brem and Kruglyak (2005) processed by Tur et al. (2014) to compare the performance of CMSTs with the method introduced by Tur et al. (2014) and named "qpgraph" hereafter. The steps and code shown below are following a vignette reproducing Figures 7 and 8 from Chaibub Neto et al. (2013) and entitled "Hotspots and causal inference for yeast data" available at `http://pages.stat.wisc.edu/~yandell/statgen/software/qtlyeast/yeast_cmst.pdf` as of September 2014.

To speed up the production of the PDF of this vignette the result of some computations are cached in `*.Rdata` files. They should be removed to force re-calculating the results. We start by loading the packages qtlhot implementing the CMST method, qtlyeast with annotation data employed in the analysis below, and parallel and doParallel to perform the more computational intensive calculations in parallel.

```
> library(qtlyeast)
> library(qtlhot)
> library(parallel)
> library(doParallel)
```

1 Re-run of qtlhot on a yeast cross data set

We load the yeast cross data processed by Tur et al. (2014) and calculate LOD scores at every 2cM using Haley-Knott regression.

```
> load("cross.Brem.Rdata")
> cross.Brem <- calc.genoprob(cross.Brem, step=2)
> if (!"scan.orf.Rdata" %in% list.files()) {
    scan.orf <- scanone(cross.Brem, pheno.col=seq(nphe(cross.Brem)), method="hk")
    save(scan.orf, file="scan.orf.Rdata")
  } else
    load("scan.orf.Rdata")
```

Determine permutation LOD thresholds for the null hypothesis of no eQTL anywhere in the genome at three different statistical levels. We use a seed to facilitate reproducing the numbers.

```
> cross <- cross.Brem
> cross <- calc.genoprob(cross, step=2)
> set.seed(12345)
> cross$pheno <- data.frame(norm=rnorm(nind(cross)))
> set.seed(12345)
> if (!"perm.orf.Rdata" %in% list.files()) {
    perm.orf <- scanone(cross, method="hk", n.perm=1000)
    save(perm.orf, file="perfm.orf.Rdata")
  } else
    load("perfm.orf.Rdata")
Doing permutation in batch mode ...
> summary(perm.orf, alpha=c(0.01, 0.05, 0.1))
LOD thresholds (1000 permutations)
     lod
1%  4.17
5%  3.41
10% 3.08
```

Select and save high LOD scores.

```
> if (!"highlod.orf.Rdata" %in% list.files()) {
    lod.thr <- c(summary(perm.orf, alpha=0.05))
    highlod.orf <- highlod(scan.orf, lod.thr=lod.thr, drop.lod=1.5)
    save(lod.thr, highlod.orf, file="highlod.orf.Rdata")
  } else
    load("highlod.orf.Rdata")
> names(highlod.orf)
[1] "highlod" "chr.pos" "names"
> head(highlod.orf$highlod)
   row phenos      lod
1 3203      6 6.484966
2 3204      6 7.083250
3 3205      6 7.571993
4 3206      6 7.898943
5 3207      6 8.032767
6 3208      6 7.972421
> dim(highlod.orf$highlod)
[1] 105074      3
> head(highlod.orf$chr.pos)
               chr      pos
11265_i_at_x01   1 0.217350
11265_i_at_x02   1 0.217801
11276_at_x12     1 1.449002
11276_at_x09     1 1.455755
11276_at_x01     1 1.477361
c1.loc2          1 2.217350
```

```
> dim(highlod.orf$chr.pos)
[1] 3843    2
> head(highlod.orf$names)
[1] "YDR407C" "YDR180W" "YAR050W" "YKL129C" "YOR328W" "YJR138W"
> length(highlod.orf$names)
[1] 6216
```

Load a yeast annotation *data.frame* object with chromosomal position for each gene, which forms part of the qtlyeast package.

```
> data(yeast.annot)
> 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
```

In the qtlyeast package as well there is a list of gene-signatures derived from KO experiments conducted by Hughes et al. (2000) and Zhu et al. (2008).

```
> data(ko.list)
> length(ko.list)
[1] 247
> head(lapply(ko.list, head))
$YOR128C
[1] "YAR073W" "YBL013W" "YBL032W" "YBL042C" "YBL054W" "YBL064C"

$YMR282C
[1] "YAL039C" "YAL044C" "YAL061W" "YBL005W" "YBL043W" "YBL064C"

$YER017C
[1] "YAL061W" "YAL062W" "YBL005W" "YBL043W" "YBL045C" "YBL049W"

$YEL036C
[1] "YAL044C"   "YAR009C"   "YAR064W"   "YBL002W"   "YBL003C"   "YBL005W.B"

$YHR013C
[1] "YAR015W"   "YCL064C"   "YCL074W"   "YCL026C.A" "YDR222W"   "YEL068C"

$YCR048W
[1] "YIR013C"   "YLR056W"   "YDR170W-A" "YFL034C-A"
```

Find eQTLs for the KO genes in the `ko.list` object:

```
> cand.reg <- GetCandReg(highlod.orf, yeast.annot, names(ko.list))
> dim(cand.reg)
```

```
[1] 144   6
> head(cand.reg, n=3)
     gene phys.chr phys.pos peak.chr  peak.pos peak.lod
1 YOR128C       15 255.8677       13  16.96703 3.575424
2 YMR282C       13 473.2316       14 171.35745 4.474679
3 YER017C        5 152.3216       14 169.35745 7.199227
```

Get candidate regulators mapping in *cis*:

```
> cis.cand.reg <- GetCisCandReg(highlod.orf, cand.reg)
> dim(cis.cand.reg)
[1] 3 7
> head(cis.cand.reg, n=3)
       gene phys.chr  phys.pos  peak.pos  peak.lod peak.pos.lower
31  YNL186W       14 161.79199 173.35745  7.013452      159.35745
233 YCL009C        3  66.45451  37.06192  6.087373       33.06192
236 YOL084W       15  58.21656  56.39798 22.533157       52.39798
    peak.pos.upper
31       183.35745
233       67.06192
236       59.16584
```

For each one of the 144 candidate KO-genes determine which other genes also co-mapped to the same QTL of the KO-gene

```
> if (!"forFitAllTests.Rdata" %in% list.files()) {
    comap.targets <- GetCoMappingTraits(highlod.orf, cand.reg)
    save(cand.reg, comap.targets, file="forFitAllTests.Rdata")
  } else
    load("forFitAllTests.Rdata")
> summary(sapply(comap.targets, length))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   80.75  223.00  262.10  507.20  591.00
> length(unlist(comap.targets))
[1] 37748
```

Perform the causality tests of each candidate regulator KO-gene to its putative targets. This is performed in parallel, and therefore, one should adapt the number of cores to the available hardware.

```
> registerDoParallel(cores=4)
> foreach (k=1:nrow(cand.reg)) %dopar% {
    cat("trait=", k, "\n")
    out <- FitAllTests(cross=cross.Brem,
                       pheno1 = cand.reg[k ,1],
                       pheno2 = 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)
  }
```

The previous computation generates results in different files. The following function call joins all the results in one single object.

```
> ko.tests <- JoinTestOutputs(comap.targets, file="output_ko_validation")
> names(ko.tests)
 [1] "R2s"          "AIC.stats"    "BIC.stats"    "pvals.j.BIC"  "pvals.p.BIC"
 [6] "pvals.np.BIC" "pvals.j.AIC"  "pvals.p.AIC"  "pvals.np.AIC" "pvals.cit"
[11] "phenos"
> head(ko.tests$pvals.j.BIC)
                   pval.1    pval.2    pval.3    pval.4
YOR128C_YML104C 0.9997715 0.9802490 0.1317432 1.0000000
YOR128C_YBL051C 0.9967955 0.9696993 0.1404016 1.0000000
YOR128C_YLR089C 0.9908934 0.9471378 0.2565940 1.0000000
YOR128C_YKL174C 0.9047424 0.7958462 0.9593991 0.8539408
YOR128C_YMR300C 0.9850849 0.5330988 0.9646496 0.9998041
YOR128C_YBR068C 0.9977548 0.9762158 0.1440231 1.0000000
> dim(ko.tests$pvals.j.BIC)
[1] 37748     4
> head(ko.tests$pvals.j.AIC)
                   pval.1    pval.2    pval.3    pval.4
YOR128C_YML104C 0.9997715 0.9802490 0.1317432 1.0000000
YOR128C_YBL051C 0.9967955 0.9696993 0.1404016 1.0000000
YOR128C_YLR089C 0.9930598 0.9484553 0.5013241 0.9992298
YOR128C_YKL174C 0.9680634 0.9076010 0.9833731 0.5904668
YOR128C_YMR300C 0.9944541 0.8130588 0.9808841 0.9340783
YOR128C_YBR068C 0.9977548 0.9762158 0.1440231 1.0000000
> dim(ko.tests$pvals.j.AIC)
[1] 37748     4
```

The object `ko.tests` contains the raw p-values of the tests for the joint-parametric, parametric and non-parametric CMSTs. We will compare all of them later to the approach introduced by Tur et al. (2014).

```
> idPairs <- paste(ko.tests$phenos[, 1], ko.tests$phenos[, 2], sep="_")
> genes <- unique(c(ko.tests$phenos[, 1], ko.tests$phenos[, 2]))
> length(genes)
[1] 2691
> pvalsCMST <- c("pvals.j.BIC", "pvals.j.AIC", "pvals.p.BIC", "pvals.p.AIC",
                 "pvals.np.BIC", "pvals.np.AIC")
> ## for some unknown reason row 1960 has no row name
> lapply(ko.tests[pvalsCMST], function(x) x[1958:1962, ])
$pvals.j.BIC
                   pval.1    pval.2    pval.3    pval.4
YMR055C_YLR081W 0.8055439 0.9994046 0.8313796 1.0000000
YMR055C_YEL068C 0.9566140 0.9978892 0.2226569 1.0000000
                0.9315503 0.9960868 0.3191286 1.0000000
YHL038C_YJR138W 0.9949585 0.8954224 0.6580078 0.9903824
```

```
YHL038C_YGR184C 0.9376368 0.9282437 0.8264628 0.8711602

$pvals.j.AIC
                   pval.1    pval.2    pval.3    pval.4
YMR055C_YLR081W 0.8055439 0.9994046 0.8313796 1.0000000
YMR055C_YEL068C 0.9640110 0.9986651 0.5179330 0.9991040
                0.9338901 0.9960868 0.4898972 0.9990712
YHL038C_YJR138W 0.9979876 0.9596546 0.8824255 0.7949448
YHL038C_YGR184C 0.9807013 0.9699696 0.9577722 0.4792273

$pvals.p.BIC
                   pval.1    pval.2    pval.3    pval.4
YMR055C_YLR081W 0.5060401 0.9993912 0.4939599 1.0000000
YMR055C_YEL068C 0.8941371 0.9901963 0.1058629 0.9731210
                0.8480126 0.9874181 0.1519874 0.9788915
YHL038C_YJR138W 0.9849631 0.7724653 0.4206004 0.5793996
YHL038C_YGR184C 0.8380825 0.8579150 0.6093657 0.3906343

$pvals.p.AIC
                   pval.1    pval.2    pval.3    pval.4
YMR055C_YLR081W 0.5060401 0.9993912 0.4939599 1.0000000
YMR055C_YEL068C 0.9081664 0.9931673 0.2956742 0.7043258
                0.8519251 0.9874181 0.2580431 0.7419569
YHL038C_YJR138W 0.9930212 0.8842921 0.6789350 0.3210650
YHL038C_YGR184C 0.9295258 0.9290801 0.8285368 0.1714632

$pvals.np.BIC
                   pval.1    pval.2      pval.3    pval.4
YMR055C_YLR081W 0.5376125 0.9999998 0.537612453 1.0000000
YMR055C_YEL068C 0.9999009 0.9999967 0.000210484 0.9999009
                0.9766325 0.9999804 0.036063193 0.9999009
YHL038C_YJR138W 0.9997895 0.8507036 0.388482376 0.6816001
YHL038C_YGR184C 0.9219668 0.9219668 0.946108860 0.1095656

$pvals.np.AIC
                   pval.1    pval.2     pval.3     pval.4
YMR055C_YLR081W 0.5376125 0.9999998 0.53761245 1.00000000
YMR055C_YEL068C 0.9999009 0.9999987 0.02336753 0.98534889
                0.9853489 0.9999804 0.14929643 0.89043443
YHL038C_YJR138W 0.9999009 0.9766325 0.98534889 0.03606319
YHL038C_YGR184C 0.9639368 0.9461089 0.99839153 0.07803315
> ## luckily every other row names match the 'phenos' element
> idPairs[1958:1962]
[1] "YMR055C_YLR081W" "YMR055C_YEL068C" "YMR275C_YLR019W" "YHL038C_YJR138W"
[5] "YHL038C_YGR184C"
> stopifnot(identical(rownames(ko.tests$pvals.j.BIC)[-1960], idPairs[-1960]))
> stopifnot(identical(rownames(ko.tests$pvals.j.AIC)[-1960], idPairs[-1960]))
> stopifnot(identical(rownames(ko.tests$pvals.p.BIC)[-1960], idPairs[-1960]))
> stopifnot(identical(rownames(ko.tests$pvals.p.AIC)[-1960], idPairs[-1960]))
```

```
> stopifnot(identical(rownames(ko.tests$pvals.np.BIC)[-1960], idPairs[-1960]))
> stopifnot(identical(rownames(ko.tests$pvals.np.AIC)[-1960], idPairs[-1960]))
> ## select tests and make sure the IDs are the same (ID for pair 1960
> ## is the empty string "") therefore we are going to assume the row 1960
> ## at the 'phenos' element contains the missing identifiers in
> ## 'pvals.j.BIC' and 'pvals.j.AIC'
> ko.tests.CMST <- lapply(ko.tests[pvalsCMST],
                          function(x, ids) { rownames(x) <- ids ; x }, idPairs)
> head(lapply(ko.tests.CMST, head, 3))
$pvals.j.BIC
                   pval.1    pval.2    pval.3 pval.4
YOR128C_YML104C 0.9997715 0.9802490 0.1317432      1
YOR128C_YBL051C 0.9967955 0.9696993 0.1404016      1
YOR128C_YLR089C 0.9908934 0.9471378 0.2565940      1

$pvals.j.AIC
                   pval.1    pval.2    pval.3    pval.4
YOR128C_YML104C 0.9997715 0.9802490 0.1317432 1.0000000
YOR128C_YBL051C 0.9967955 0.9696993 0.1404016 1.0000000
YOR128C_YLR089C 0.9930598 0.9484553 0.5013241 0.9992298

$pvals.p.BIC
                   pval.1    pval.2     pval.3    pval.4
YOR128C_YML104C 0.9991300 0.9557354 0.04426456 1.0000000
YOR128C_YBL051C 0.9878903 0.9477637 0.05223632 1.0000000
YOR128C_YLR089C 0.9707494 0.8787884 0.12121165 0.9843119

$pvals.p.AIC
                   pval.1    pval.2     pval.3    pval.4
YOR128C_YML104C 0.9991300 0.9557354 0.04426456 1.0000000
YOR128C_YBL051C 0.9878903 0.9477637 0.05223632 0.9895151
YOR128C_YLR089C 0.9764188 0.8810976 0.27566875 0.7243312

$pvals.np.BIC
                   pval.1    pval.2       pval.3    pval.4
YOR128C_YML104C 0.9999009 0.9995697 0.0008473411 1.0000000
YOR128C_YBL051C 0.9911171 0.9983915 0.0146511115 1.0000000
YOR128C_YLR089C 0.9999551 0.9995697 0.0008473411 0.9997895

$pvals.np.AIC
                   pval.1    pval.2       pval.3    pval.4
YOR128C_YML104C 0.9999009 0.9995697 0.0008473411 1.0000000
YOR128C_YBL051C 0.9911171 0.9983915 0.0146511115 0.9991527
YOR128C_YLR089C 0.9999551 0.9995697 0.2542565900 0.8024033
```

Here, `pval.1`, `pval.2`, `pval.3` and `pval.4` correspond, respectively, to the p-values derived from the application of four separate joint-parametric CMST tests to the causal, reactive, independence and full models (denoted by M_1, M_2, M_3 and M_4 in Chaibub Neto et al. (2013). Only the independence M_3 model, with p-values in the `pval.3` column, is relevant to our comparison.

Moreover, to compare a CMST with the mixed GMM approach implemented by qpgraph we need to transform the CMST into what one may call a Conditional Independence Model Selection Test (CIMST). It can be shown that in the case of the M_3 model, any version of a CIMST (joint-parametric, parametric or non-parametric) can be calculated as one minus the p-value of the corresponding CMST (column `pval.3`). Therefore, we will use this particular CIMST for comparison with the approach by Tur et al. (2014) and implemented in the R/BioC package qpgraph.

```
> ko.tests.CIMST <- sapply(ko.tests.CMST, function(x) 1 - x[, 3])
```

Since qpgraph scores undirected edges and stores its results into squared symmetric matrices, we will also store the CIMST p-values into square matrices to facilitate their comparison.

```
> pvals.CIMST <- vector(mode="list", length=ncol(ko.tests.CIMST))
> names(pvals.CIMST) <- colnames(ko.tests.CIMST)
> for (cimst in names(pvals.CIMST)) {
    pvals.CIMST[[cimst]] <- matrix(NA_real_, nrow=length(genes),
                                   ncol=length(genes),
                                   dimnames=list(genes, genes))
    pvals.CIMST[[cimst]][ko.tests$phenos] <- ko.tests.CIMST[, cimst]
  }
```

Every CIMST p-value tests the null hypothesis of the absence of a directed edge going from a KO-gene to a putative target. When the putative target is also a KO-gene then the matrix has the two corresponding symmetric cells. However, when the putative target is not a KO-gene, the cell containing the p-value does not have its symmetric counterpart. In these cases we assign the p-value of the KO-gene to its putative target to the cell that would correspond to the reverse relationship of the putative target to the KO-gene, making effectively this edge undirected to enable the comparison with qpgraph.

```
> tg2koEdges <- paste(ko.tests$phenos[, 2], ko.tests$phenos[, 1], sep="_")
> missingEdges <- !tg2koEdges %in% rownames(ko.tests.CIMST)
> tg2koEdges <- cbind(ko.tests$phenos[missingEdges, 2],
                      ko.tests$phenos[missingEdges, 1])
> for (cimst in names(pvals.CIMST)) {
    pvals.CIMST[[cimst]][tg2koEdges] <- ko.tests.CIMST[missingEdges, cimst]
    stopifnot(isSymmetric(is.na(pvals.CIMST[[cimst]]))) ## QC
  }
```

We want to reproduce Figure 7 and 8 from (Chaibub Neto et al., 2013) on the yeast cross data set processed by Tur et al. (2014) to verify that calculations have been correctly done. For this purpose we should continue now adjusting the p-values by FDR.

```
> adj.ko.tests <- p.adjust.np(ko.tests)
```

Finally, performances of these methods as function of the nominal significance level are calculated as follows and shown in Figure 1 which reproduce Figures 7 and 8 from Chaibub Neto et al. (2013) with the yeast cross data processed by Tur et al. (2014).

```
> if (!"roc.aux.Rdata" %in% list.files()) {
    roc.aux <- PrecTpFpMatrix(alpha=seq(0.01, 0.1, by=0.01),
                              val.targets=ko.list,
                              all.orfs=highlod.orf$names,
                              tests=adj.ko.tests,
                              cand.reg=cand.reg, cis.cand.reg=cis.cand.reg)
    save(roc.aux, file="roc.aux.Rdata")
  } else
    load("roc.aux.Rdata")
```

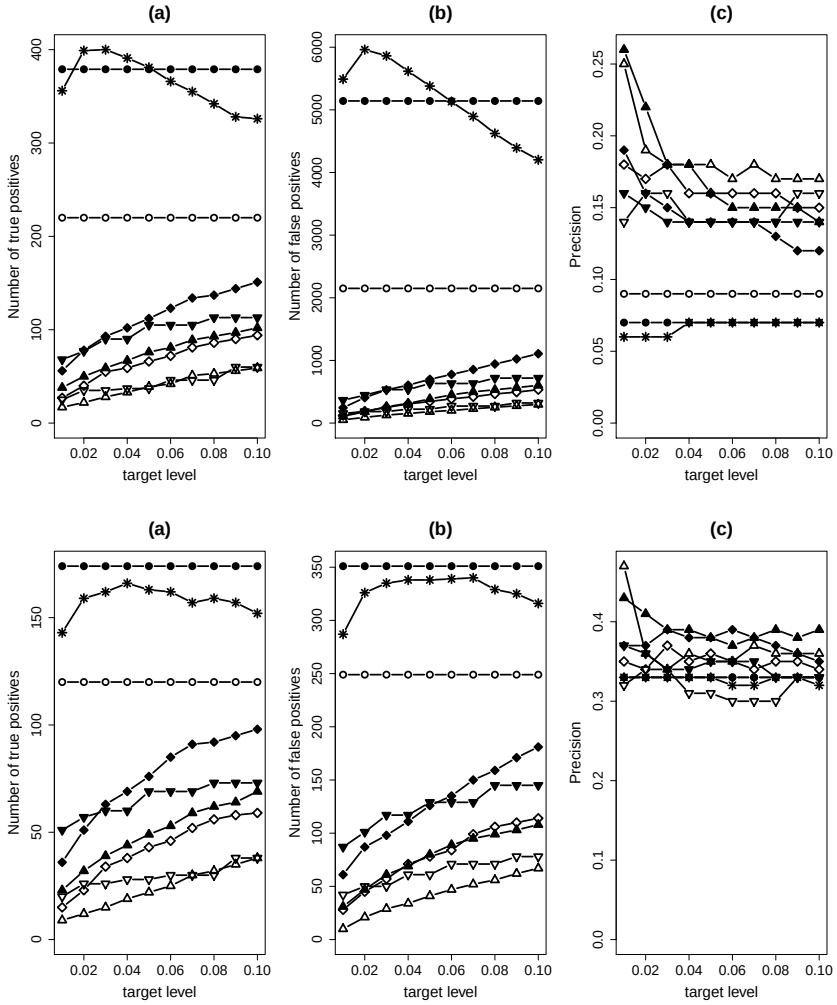

Figure 1: Results of the qtlhot package with the yeast cross data set from Brem and Kruglyak (2005) processed by Tur et al. (2014). Top and bottom panels, respectively, are analogous to Figures 7 and 8 from Chaibub Neto et al. (2013).

2 Comparison of qtlhot with qpgraph

We are going now to compare the results from the joint-parametric, parametric and non-parametric CMSTs with the approach introduced by Tur et al. (2014). We have to consolidate

first the set of genes employed in both analyses and start loading the qpgraph package and the result object of the analysis on the yeast cross.

```
> library(qpgraph)
> load("eqtlnet.q0.fdr.nrr.sel.RData")
> param
eQTLnetworkEstimationParam object:
  Organism: Saccharomyces cerevisiae
  Genome: sacCer3
  Gene annotation: UCSC Table sgdGene
  1857 markers 6141 genes
> eqtlnet.q0.fdr.nrr.sel
eQTLnetwork object:
  Organism: Saccharomyces cerevisiae
  Genome: sacCer3
  Gene annotation: UCSC Table sgdGene
  Input size: 1857 markers 6141 genes
  Model formula:  ~marker | gene(q = 0, 25, 50, 75, 100)
  G^(0,25,50,75,100,*): 1781 vertices and 2300 edges corresponding to
                        500 eQTL and 1799 gene-gene associationsmeeting
                        a fdr-adjusted p-value < 0.01,
                        a non-rejection rate epsilon < 0.10,
                        a forward eQTL selection significance level alpha < 0.05
                        and involving 1391 genes and 389 eQTLs
> ko.list.present <- ko.list[intersect(names(ko.list), geneNames(param))]
> length(ko.list.present)
[1] 235
> ko.list.present <- lapply(ko.list.present,
                            function(x, g)
                              intersect(make.names(x), g), geneNames(param))
> ## ko.list.present <- ko.list.present[sapply(ko.list.present, length) > 0]
> length(ko.list.present)
[1] 235
> ko.genes <- unique(c(names(ko.list.present),
                       unlist(ko.list.present, use.names=FALSE)))
> length(ko.genes)
[1] 5143
```

The package qpgraph filters out genes in the yeast cross that have no current annotation for this reason we need still to get the common set of genes between CMST and qpgraph, and then with the genes in `ko.genes`.

```
> genes.CIMST <- rownames(pvals.CIMST[[1]])
> genes.qpgraph <- geneNames(param)
> common.genes <- intersect(ko.genes, intersect(genes.CIMST, genes.qpgraph))
> length(common.genes)
[1] 2381
```

Filter out genes in `ko.list.present` not common to CMST and qpgraph results:

```
> ko.list.present <- ko.list.present[intersect(names(ko.list.present), common.genes)]
> length(ko.list.present)
[1] 145
> ko.list.present <- lapply(ko.list.present, function(x, g)
                            intersect(make.names(x), g), common.genes)
> length(ko.list.present)
[1] 145
> ko.genes <- unique(c(names(ko.list.present),
                       unlist(ko.list.present, use.names=FALSE)))
> length(ko.genes)
[1] 2199
```

Build a *bronze standard* of gene associations between a KO-gene and its targets discarding duplicated and self-regulatory associations:

```
> elen <- sapply(ko.list.present, length)
> presentEdges <- cbind(rep(names(ko.list.present), times=elen),
                        unlist(ko.list.present, use.names=FALSE))
> ## discard duplicated edges
> presentEdges <- t(apply(presentEdges, 1, sort))
> presentEdges <- unique(presentEdges)
> ## discard loops
> presentEdges <- presentEdges[presentEdges[,1] != presentEdges[, 2], ]
> nrow(presentEdges)
[1] 10656
> bronzeStd <- matrix(FALSE, nrow=length(ko.genes), ncol=length(ko.genes),
                      dimnames=list(ko.genes, ko.genes))
> bronzeStd[presentEdges] <- TRUE
> bronzeStd <- bronzeStd | t(bronzeStd)
> diag(bronzeStd) <- FALSE
> stopifnot(isSymmetric(bronzeStd)) ## QC
> stopifnot(sum(bronzeStd[upper.tri(bronzeStd)]) == nrow(presentEdges)) ## QC
```

Restrict the results of the compared methods to the genes in `ko.genes`:

```
> pvals.CIMST <- lapply(pvals.CIMST, function(x, g) x[g, g], ko.genes)
> nrr <- eqtlnet.q0.fdr.nrr.sel@nrr[ko.genes, ko.genes]
```

Since the results of the CIMSTs are also restricted to pairs of genes with common eQTLs, we also set to `NA` the corresponding cells in the matrix of non-rejection rates calculated with qpgraph:

```
> missingMask <- is.na(pvals.CIMST[[1]])
> nrr[missingMask] <- NA_real_
> nrr[1:5, 1:5]
```

```
5 x 5 Matrix of class "dgeMatrix"
        YOR128C   YMR282C   YER017C YHR013C YER069W
YOR128C      NA        NA        NA      NA      NA
YMR282C      NA        NA 0.6094606      NA      NA
YER017C      NA 0.6094606        NA      NA      NA
YHR013C      NA        NA        NA      NA      NA
YER069W      NA        NA        NA      NA      NA
```

Now calculate precision-recall curves using the function `qpPrecisionRecall()` from the qpgraph package. This function will take the missing `NA` entries into account to properly calculate recall and precision rates. Although the `NA` entries do not make it necessary, we also set explicitly the constraint that the bronze standard is formed by relationships restricted to pairs involving at least one KO-gene, using the arguments `pairup.i` and `pairup.j`:

```
> CIMST.pr <- vector(mode="list", length=length(pvals.CIMST))
> names(CIMST.pr) <- names(pvals.CIMST)
> for (cimst in names(CIMST.pr))
    CIMST.pr[[cimst]] <- qpPrecisionRecall(measurementsMatrix=pvals.CIMST[[cimst]],
                                           refGraph=bronzeStd,
                                           pairup.i=names(ko.list.present),
                                           pairup.j=ko.genes,
                                           decreasing=FALSE,
                                           recallSteps=c(seq(0, 0.1, by=0.05),
                                                         seq(0.2, 1, by=0.1)))
> qpgraph.pr <- qpPrecisionRecall(measurementsMatrix=nrr,
                                  refGraph=bronzeStd,
                                  pairup.i=names(ko.list.present),
                                  pairup.j=ko.genes,
                                  decreasing=FALSE,
                                  recallSteps=c(seq(0, 0.1, by=0.05),
                                                seq(0.2, 1, by=0.1)))
```

Calculate the area under the precision-recall curve (AUC) as a summary measure of performance of each method.

```
> fauc <- function(f) integrate(approxfun(f), 0, 1)$value
> CIMST.pr.auc <- sapply(lapply(CIMST.pr, "[", , c("Recall", "Precision")), fauc)
> CIMST.pr.auc
 pvals.j.BIC  pvals.j.AIC  pvals.p.BIC  pvals.p.AIC pvals.np.BIC pvals.np.AIC
   0.1237574    0.1219209    0.1235473    0.1223287    0.1144099    0.1122646
> qpgraph.pr.auc <- fauc(qpgraph.pr[, c("Recall", "Precision")])
> qpgraph.pr.auc
[1] 0.1231494
> qpgraph.pr.auc / CIMST.pr.auc
 pvals.j.BIC  pvals.j.AIC  pvals.p.BIC  pvals.p.AIC pvals.np.BIC pvals.np.AIC
   0.9950872    1.0100769    0.9967799    1.0067096    1.0763884    1.0969575
```

We observe that the AUC is nearly identical between qpgraph and all the CIMSTs. This can be clearly seen in Figure 2A. A likely reason is that the bronze standard includes many indirect

associations resulting from differential expression cascades, which can be appropriately ranked by a marginal model (i.e., without conditioning) such as the CIMST M_3. To increase the discriminative power of the bronze standard we are going restrict it to those KO-gene and putative target associations that also exist in the database Yeastract (Teixeira et al., 2014), available at `http://www.yeastract.com`, which contains curated regulatory associations between transcription factors and target genes in *Saccharomyces cerevisiae*. We start building this more restricted bronze standard by first downloading the flat file with all curated transcriptional regulatory relationships:

```
> download.file("http://www.yeastract.com/download/RegulationTwoColumnTable_Documented_2
                "RegulationTwoColumnTable_Documented_2013927.tsv.gz", method="curl")
```

We read the downloaded file into R and map the gene symbols to ORF identifiers using the R/BioC gene-centric annotation package for yeast org.Sc.sgd.db and a helper function defined here below called `sym2orfID()`:

```
> library(org.Sc.sgd.db)
> sym2orfID <- function(twocol) {
    require(org.Sc.sgd.db)

    twocolmat <- as.matrix(twocol)
    stopifnot(ncol(twocolmat) == 2) ## QC
    twocolmatID <- select(org.Sc.sgd.db, columns="ORF",
                          keys=twocolmat[, 1], keytype="GENENAME")
    twocolmat[!is.na(twocolmatID$ORF), 1] <- twocolmatID$ORF[!is.na(twocolmatID$ORF)]
    twocolmatID <- select(org.Sc.sgd.db, columns="ORF",
                          keys=twocolmat[, 2], keytype="GENENAME")
    twocolmat[!is.na(twocolmatID$ORF), 2] <- twocolmatID$ORF[!is.na(twocolmatID$ORF)]
    twocolmat
  }
> yeastract <- read.csv(gzfile("RegulationTwoColumnTable_Documented_2013927.tsv.gz"), se
                        stringsAsFactors=FALSE, header=FALSE)
> yeastract <- sym2orfID(yeastract)
> dim(yeastract)
[1] 201974      2
> head(yeastract)
     V1        V2
[1,] "YKL112W" "YKL112W"
[2,] "YKL112W" "YAL054C"
[3,] "YKL112W" "YGL234W"
[4,] "YKL112W" "YOL086C"
[5,] "YKL112W" "YMR169C"
[6,] "YKL112W" "YGL156W"
```

Each row in this matrix corresponds to a curated transcriptional regulatory relationship where the first column contains the transcription factor and the second the regulated target. We store the unique set of transcription factor gene identifiers:

```
> yeastractTFs <- unique(yeastract[, 1])
> length(yeastractTFs)
```

```
[1] 307
```

Next, we remove discard duplicated and self-regulatory associations:

```
> yeastract <- t(apply(yeastract, 1, sort))
> yeastract <- unique(yeastract)
> yeastract <- yeastract[yeastract[, 1] != yeastract[, 2], ]
> dim(yeastract)
[1] 201230      2
```

Finally, we build another bronze standard with curated transcriptional regulatory relationships among the genes we analyze:

```
> yeastract <- yeastract[yeastract[,1] %in% ko.genes & yeastract[,2] %in% ko.genes, ]
> bronzeStdYeastract <- matrix(FALSE, nrow=length(ko.genes), ncol=length(ko.genes),
                               dimnames=list(ko.genes, ko.genes))
> bronzeStdYeastract[yeastract] <- TRUE
> bronzeStdYeastract <- bronzeStdYeastract | t(bronzeStdYeastract)
> diag(bronzeStdYeastract) <- FALSE
> stopifnot(isSymmetric(bronzeStdYeastract)) ## QC
> stopifnot(sum(bronzeStdYeastract[upper.tri(bronzeStdYeastract)]) == nrow(yeastract)) #
```

Now, calculate again the precision-recall curves using the intersection of the KO-gene and putative target bronze standard and the Yeastract bronze standard. We have to take care to restrict the comparison to associations that involve at least a transcription factor gene with the arguments `pairup.i` and `pairup.j` in the call to `qpPrecisionRecall()`:

```
> CIMST.yeastract.pr <- vector(mode="list", length=length(pvals.CIMST))
> names(CIMST.yeastract.pr) <- names(pvals.CIMST)
> for (cimst in names(CIMST.yeastract.pr))
    CIMST.yeastract.pr[[cimst]] <-
      qpPrecisionRecall(measurementsMatrix=pvals.CIMST[[cimst]],
                        refGraph=bronzeStd & bronzeStdYeastract,
                        pairup.i=intersect(names(ko.list.present), yeastractTFs),
                        pairup.j=ko.genes, decreasing=FALSE,
                        recallSteps=c(seq(0, 0.1, by=0.05),
                        seq(0.2, 1, by=0.1)))
> qpgraph.yeastract.pr <-
    qpPrecisionRecall(measurementsMatrix=nrr,
                      refGraph=bronzeStd & bronzeStdYeastract,
                      pairup.i=intersect(names(ko.list.present), yeastractTFs),
                      pairup.j=ko.genes,
                      decreasing=FALSE,
                      recallSteps=c(seq(0, 0.1, by=0.05),
                                    seq(0.2, 1, by=0.1)))
```

Calculate the AUC for each of the previous curves:

```
> CIMST.yeastract.pr.auc <-
    sapply(lapply(CIMST.yeastract.pr, "[", , c("Recall", "Precision")), fauc)
> CIMST.yeastract.pr.auc
```

```
 pvals.j.BIC  pvals.j.AIC  pvals.p.BIC  pvals.p.AIC pvals.np.BIC pvals.np.AIC
   0.1278524    0.1332908    0.1249618    0.1274540    0.1146765    0.1103481
> qpgraph.yeastract.pr.auc <- fauc(qpgraph.yeastract.pr[, c("Recall", "Precision")])
> qpgraph.yeastract.pr.auc
[1] 0.1981544
> qpgraph.yeastract.pr.auc / CIMST.yeastract.pr.auc
 pvals.j.BIC  pvals.j.AIC  pvals.p.BIC  pvals.p.AIC pvals.np.BIC pvals.np.AIC
    1.549869     1.486632     1.585720     1.554713     1.727943     1.795720
> qpgraph.inc <- 100*(qpgraph.yeastract.pr.auc/CIMST.yeastract.pr.auc-1)
> qpgraph.inc
 pvals.j.BIC  pvals.j.AIC  pvals.p.BIC  pvals.p.AIC pvals.np.BIC pvals.np.AIC
    54.98685     48.66318     58.57199     55.47129     72.79426     79.57204
```

We can see that when using a more direct set of regulatory relationships as bronze standard the AUC for qpgraph is between 49% and 80% higher than using any of the CIMSTs. This improvement can be graphically seen in Figure 2 that appears in (Tur et al., 2014).

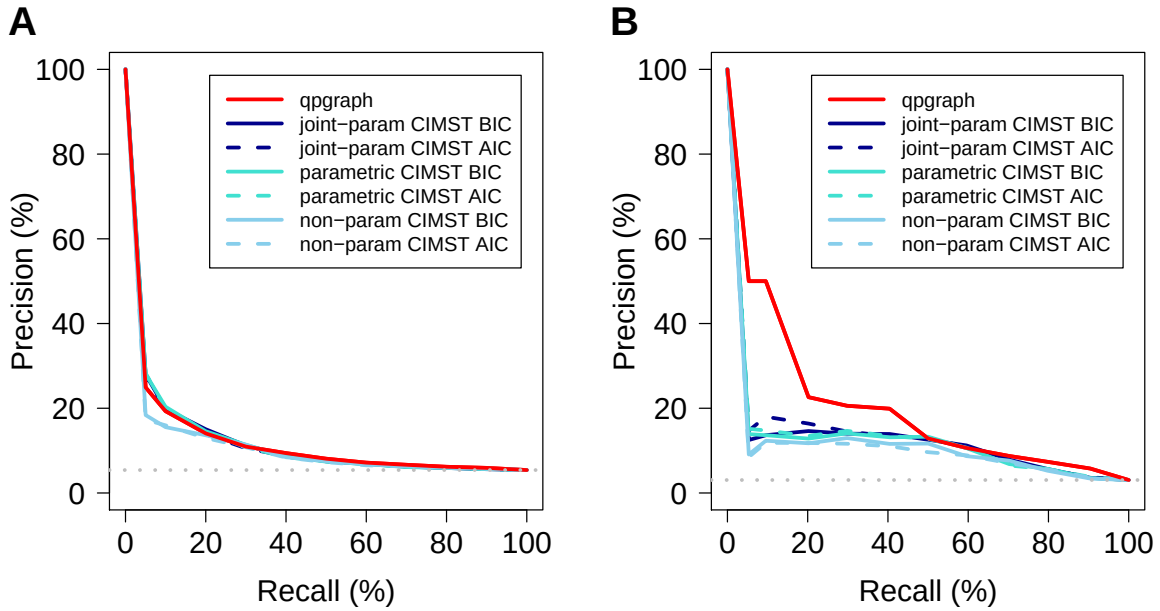

Figure 2: Comparison of qpgraph with CIMST. Precision-recall curves calculated from predicted regulatory relationships inferred from a yeast cross data set using the R/BioC package qpgraph (Tur et al., 2014) and 6 different configurations of the CMST method implemented in the R package qtlhot (Chaibub Neto et al., 2013). In (A) predictions are compared against a bronze standard formed by relationships formed by knocked-out genes and their putative targets derived from differential expression. In (B) this bronze standard is further restricted to relationships also present in the Yeastract database (Teixeira et al., 2014) of curated transcriptional regulatory associations. The horizontal gray dotted line indicates the baseline precision attained by a random predictor.

3 Session Information

```
> toLatex(sessionInfo())
```

- R version 3.1.0 (2014-04-10), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF8`, `LC_COLLATE=en_US.UTF8`, `LC_MONETARY=en_US.UTF8`, `LC_MESSAGES=en_US.UTF8`, `LC_PAPER=en_US.UTF8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.27.10, Biobase 2.25.0, BiocGenerics 0.11.5, colorout 1.0-2, corpcor 1.6.6, DBI 0.3.0, doParallel 1.0.8, foreach 1.4.2, GenomeInfoDb 1.1.19, IRanges 1.99.28, iterators 1.0.7, lattice 0.20-29, mnormt 1.5-1, org.Sc.sgd.db 2.14.0, qpgraph 1.21.22, qtl 1.33-7, qtlhot 0.9.0, qtlyeast 1.0.4, RSQLite 0.11.4, S4Vectors 0.2.4, setwidth 1.0-3, vimcom 0.9-93
- Loaded via a namespace (and not attached): annotate 1.43.5, BatchJobs 1.3, BBmisc 1.7, BiocParallel 0.99.19, biomaRt 2.21.1, Biostrings 2.33.14, bitops 1.0-6, brew 1.0-6, checkmate 1.4, codetools 0.2-9, digest 0.6.4, fail 1.2, genefilter 1.47.6, GenomicAlignments 1.1.29, GenomicFeatures 1.17.14, GenomicRanges 1.17.40, GGBase 3.27.3, graph 1.43.0, grid 3.1.0, Matrix 1.1-4, mvtnorm 1.0-0, RCurl 1.95-4.3, Rgraphviz 2.9.1, Rsamtools 1.17.33, rtracklayer 1.25.16, sendmailR 1.1-2, snpStats 1.15.4, splines 3.1.0, stringr 0.6.2, survival 2.37-7, tools 3.1.0, XML 3.98-1.1, xtable 1.7-4, XVector 0.5.8, zlibbioc 1.11.1

References

Brem, R. B. and Kruglyak, L. (2005). The landscape of genetic complexity across 5,700 gene expression traits in yeast. *P Natl Acad Sci USA*, 102:1572–1577.

Chaibub Neto, E., Broman, A. T., Keller, M. P., Attie, A. D., Zhang, B., Zhu, J., and Yandell, B. S. (2013). Modeling causality for pairs of phenotypes in systems genetics. *Genetics*, 193:1003–1013.

Hughes, T. R., Marton, M. J., Jones, A. R., Roberts, C. J., Stoughton, R., Armour, C. D., Bennett, H. A., Coffey, E., Dai, H., He, Y. D., et al. (2000). Functional discovery via a compendium of expression profiles. *Cell*, 102(1):109–126.

Teixeira, M. C., Monteiro, P. T., Guerreiro, J. F., Gonçalves, J. P., Mira, N. P., dos Santos, S. C., Cabrito, T. R., Palma, M., Costa, C., Francisco, A. P., Madeira, S. C., Oliveira, A. L., Freitas, A. T., and Sá-Correia, I. (2014). The yeastract database: an upgraded information system for the analysis of gene and genomic transcription regulation in saccharomyces cerevisiae. *Nucleic acids research*, 42(D1):D161–D166.

Tur, I., Roverato, A., and Castelo, R. (2014). Mapping eQTL networks with mixed graphical Markov models. *submitted*.

Zhu, J., Zhang, B., Smith, E. N., Drees, B., Brem, R. B., Kruglyak, L., Bumgarner, R. E., and Schadt, E. E. (2008). Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks. *Nature genetics*, 40(7):854–861.