GAPS/CoGAPS Users Manual

Elana J. Fertig (ejfertig@jhmi.edu) and Michael F. Ochs (ochsm@tcnj.edu)

October 13, 2014

Contents

Chapter 1

Introduction

Gene Association in Pattern Sets (GAPS) infers underlying patterns in a matrix of measurements that can be interpreted as arising from the multiplication of two lower dimensional matrices. The first development of this code in R/Biocondcutor was focused on gene expression analysis, however the original concept was used in spectral imaging. The approach is a general form of matrix factorization using a stochastic algorithm. In this vignette, we will focus on gene expression analysis for concreteness, but the factorization is applicable more broadly.

The Markov chain Monte Carlo (MCMC) matrix factorization that infers patterns also infers the extent to which individual genes belong to these patterns. The CoGAPS algorithm extends GAPS to infer the coordinated activity in sets of genes for each of the inferred patterns based upon (5) and to refine gene set membership based upon (2).

The GAPS algorithm is implemented in C++ and compiled and integrated into R using the Rcpp package. GAPS is licensed under the GNU General Public License version 2. You may freely modify and redistribute GAPS under certain conditions that are described in the top level source directory file `COPYING`.

The R package CoGAPS is designed to facilitate the corresponding analysis of microarray measurements by calling the GAPS C++ library. With the migration to C++ code, the installation as noted in Chapter 2 should now be automatic. Running instructions for the GAPS and CoGAPS analyses are provided in Sections 3.1 and 3.2, respectively. CoGAPS and GAPS are freely available at `https://github.com/ejfertig/CoGAPS` and through the CoGAPS Bioconductor package.

If you use the `CoGAPS` package for your analysis please cite: (1) EJ Fertig, J Ding, AV Favorov, G Parmigiani, and MF Ochs (2010) CoGAPS: an R/C++ package to identify patterns and biological process activity in transcriptomic data. *Bioinformatics* **26**: 2792-2793.

To cite the CoGAPS algorithm use: (3) MF Ochs (2003) Bayesian Decomposition in *The Analysis of Gene Expression Data: Methods and Software* G Parmigiani, E Garrett, R Irizarry, and S Zeger, ed. New York: Springer Verlag.

To cite the gene set statistic use: (5) MF Ochs, L Rink, C Tarn, S Mburu, T Taguchi, B Eisenberg, and AK Godwin (2009) Detection of treatment-induced changes in signaling pathways in gastrointestinal stromal tumors using transcriptomic data. *Cancer Research* **69**: 9125-9132.

To site the set-membership refinement statistic use: (2) EJ Fertig, AV Favorov, and MF Ochs (2012) Identifying context-specific transcription factor targets from prior knowledge and gene expression data. *2012 IEEE International Conference on Bioinformatics and Biomedicine*, B310, *in press*.

Please contact Elana J. Fertig `ejfertig@jhmi.edu` or Michael F. Ochs `ochsm@tcnj.edu` for assistance.

Chapter 2

Installation Instructions

The GAPS and CoGAPS algorithms are implemented in an open source C++ software and an R package to facilitate analysis, visualization, and integration with other R tools (CoGAPS, available through Bioconductor).

With this version of CoGAPS, installation should be automatically completed through Bioconductor package installation:

```
source("http://www.bioconductor.org/biocLite.R")
biocLite("CoGAPS")
```

The C++ software will be automatically compiled, linking to required boost libraries distributed as part of the CRAN `BH` package (`http://cran.r-project.org/web/packages/BH/index.html`) and Rcpp (`http://cran.r-project.org/web/packages/Rcpp/index.html`).

Chapter 3

Executing CoGAPS

In this chapter, we describe how to run both the GAPS and CoGAPS algorithms.

3.1 GAPS

GAPS seeks a pattern matrix ($\mathbf{P}$) and the corresponding distribution matrix of weights ($\mathbf{A}$) whose product forms a mock data matrix ($\mathbf{M}$) that represents the expression data $\mathbf{D}$ within noise limits ($\boldsymbol{\varepsilon}$). That is,

$$\mathbf{D} = \mathbf{M} + \boldsymbol{\varepsilon} = \mathbf{AP} + \boldsymbol{\varepsilon}. \tag{3.1}$$

The number of rows in $\mathbf{P}$ (columns in $\mathbf{A}$) defines the number of biological patterns that GAPS will infer from the measured microarray data or equivalently the number of nonorthogonal basis vectors required to span the data space. As in the Bayesian Decomposition algorithm (4), the matrices $\mathbf{A}$ and $\mathbf{P}$ in GAPS are assumed to have the atomic prior described in (6). In the GAPS implementation, α_A and α_P are corresponding parameters for the expected number of atoms which map to each matrix element in $\mathbf{A}$ and $\mathbf{P}$, respectively. The corresponding matrices $\mathbf{A}$ and $\mathbf{P}$ are found by MCMC sampling.

3.1.1 Methods

The GAPS algorithm is run by calling the `gapsRun` function in the CoGAPS R package as follows:

```
> results <- gapsRun(data, unc, nFactor = "5", simulation_id = "simulation",
                     nEquil = "1000", nSample = "1000", nOutR = 1000,
                     output_atomic = "FALSE",
                     alphaA = "0.01",  nMaxA = "100000",
                     max_gibbmass_paraA = "100.0",
                     lambdaA_scale_factor = "1.0",
                     alphaP = "0.01", nMaxP = "100000",
                     max_gibbmass_paraP = "100.0",
                     lambdaP_scale_factor = "1.0")
```

Input Arguments The inputs that must be set each time are only the data and standard deviation matrices, with all other inputs having default values. However, in reality it is essential to set at least nFactor, nEquil, and nSample based on the expected dimensionality of the data and the number of iterations needed to explore the posterior distribution. The arguments are as follows:

data The matrix of m genes by n samples of expression data. The input should be a matrix object and row names and column names will be retained in the output matrices as appropriate.

unc The matrix of m genes by n samples of standard deviations for the expression data. The values will be used in estimating the Likelihood, presently under an assumption of normal error distribution.

nFactor Number of patterns into which the data will be decomposed. The value must be less than the number of genes and number of samples in the data to avoid instability from an ill-posed problem.

simulation_id The base name to attach to files with atoms if created.

nEquil The number of iterations for "burning in" the sampler before beginning to gather samples for the statistics. The equilibration is done using a simulated annealing approach that slowly lowers the "temperature" to more thoroughly explore the posterior distribution prior to sampling.

nSample The number of iterations for sampling. Each iteration refers to a number of proposals chosen from a Poisson distribution with mean equal to the present number of atoms.

nOutR The number of iterations between reporting the present status of the sampler back to R.

output_atomic This determines whether to write atom files to disk (large).

alphaA A sparsity parameter reflecting the expected number of atoms per element of the amplitude **A** matrix in the decomposition. To enforce sparsity, this parameter should typically be less than one. (optional; default=0.01)

alphaP A sparsity parameter reflecting the expected number of atoms per element of the pattern matrix in the decomposition. To enforce sparsity, this parameter should typically be less than one. (optional; default=0.01)

max_gibbmass_paraA This provides an upper limit on the size of the truncated normal used during Gibbs sampling steps on the A domain. This avoids the creation of an unusually large mass during sampling when the sampler is in the tail of the truncated normal.

max_gibbmass_paraP This is the corresponding limit for the P domain.

nMaxA This will set a maximum number of atoms in the A domain, however it is not yet implemented in this version.

nMaxP This will set a maximum number of atoms in the P domain, however it is not yet implemented in this version.

lambdaA_scale_factor This is the lambda factor for the penalized likelihood in A.

lambdaP_scale_factor This is the lambda factor for the penalized likelihood in P.

The algorithm returns in the results a list with

Amean A matrix with the sampled mean value for the amplitude matrix **A**.

Asd A matrix with the sampled standard deviation for the amplitude matrix **A**.

Pmean A matrix with the sampled mean value for the pattern matrix **P**.

Psd A matrix with the sampled standard deviation for the pattern matrix **P**.

atomsAEquil A vector with the number of atoms in the A domain throughout the equilibration steps.

atomsASamp A vector with the number of atoms in the A domain throughout the sampling steps.

atomsPEquil A vector with the number of atoms in the P domain throughout the equilibration steps.

atomsPSamp A vector with the number of atoms in the P domain throughout the sampling steps.

chiSqValues A vector with the sample chi-squared values throughput sampling.

meanChi2 χ^2 value of the final mean result, i.e. $\chi^2 = frac(D - AP)^2\sigma^2$.

Once the GAPS algorithm has been run, the inferred patterns and corresponding amplitude can be displayed using the `plotGAPS` function as follows:

```
> plotGAPS(Amean, Pmean, outputPDF="")
```

where setting outputPDF to a string will redirect output to a PDF file from the screen. The command

```
> plotP(Pmean,Psd)
```

will create a plot of just the rows of the $mathbf P$ matrix with standard errors, which is equivalent to plotting the basis vectors for the matrix factorization.

Input Arguments

Amean The amplitude matrix **Amean** obtained from GAPS.

Pmean The pattern matrix **Pmean** obtained from GAPS.

outputPDF Name of an `pdf` file to which the results will be output. (Optional; default="" will output plots to the screen.)

Psd The standard deviation of the mean for **Pmean**.

Side Effects

- Save the plots of **Amean** and **Pmean** to the `pdf` file `outputPDF`.

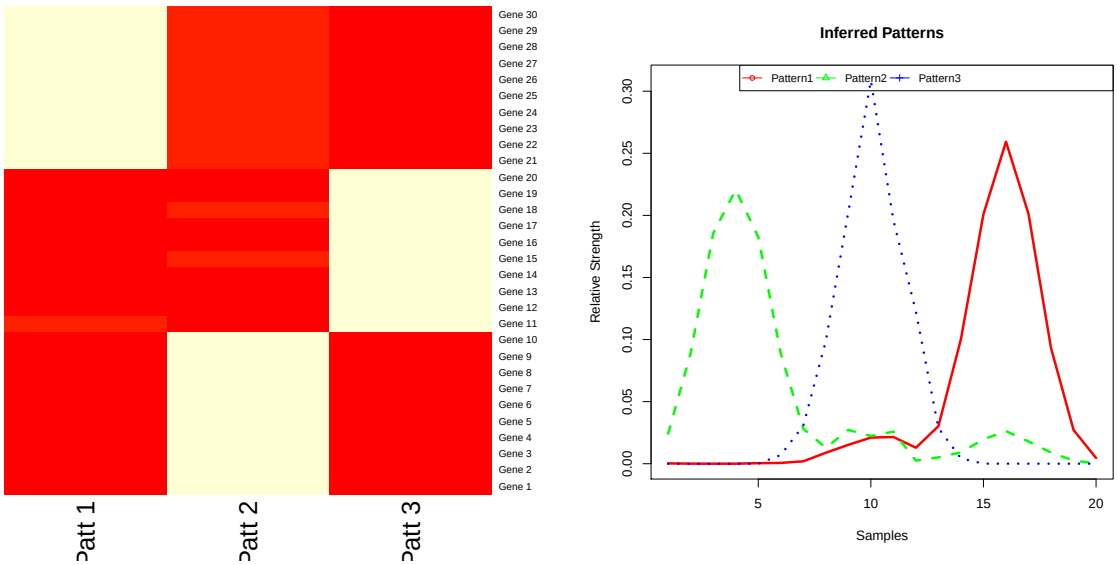
3.1.2 Example: Simple Simulation

In this example, we have simulated data in SimpSim (`SimpSim.D`) with three known patterns (`SimpSim.P`) and corresponding amplitude (`SimpSim.A`) with specified activity in two gene sets (GSets). In this data set, each gene set is overexpressed in one the simulated patterns and underexpressed in one.

```
> library('CoGAPS')
> data('SimpSim')
> nIter <- 5000
> results <- gapsRun(SimpSim.D, SimpSim.S, nFactor=3,
+                 nEquil=nIter, nSample=nIter)

Equil:1000 of 5000, Atoms:66(57)  Chi2 = 1686.03
Equil:2000 of 5000, Atoms:71(59)  Chi2 = 1523.92
Equil:3000 of 5000, Atoms:77(64)  Chi2 = 1493.69
Equil:4000 of 5000, Atoms:84(67)  Chi2 = 1493.27
Equil:5000 of 5000, Atoms:76(65)  Chi2 = 1483.7
Samp: 1000 of 5000, Atoms:80(67)  Chi2 = 1487.81
Samp: 2000 of 5000, Atoms:84(61)  Chi2 = 1530.93
Samp: 3000 of 5000, Atoms:76(67)  Chi2 = 1465.68
Samp: 4000 of 5000, Atoms:86(67)  Chi2 = 1469.54
Samp: 5000 of 5000, Atoms:80(63)  Chi2 = 1469.14
 *** Check value of final chi2: 1469.14 ****
[1] "Chi-Squared of Mean: 1383.72990954499"

> plotGAPS(results$Amean, results$Pmean, 'ModSimFigs')
```



(a) Inferred amplitude matrix

(b) Inferred patterns

Figure 3.1: Results from GAPS on simulated data set with known true patterns.

```
null device
          1
```

Figure 3.1 shows the results from plotting the GAPS estimates of $\mathbf{A}$ and $\mathbf{P}$ using `plotGAPS`, which has a fit to $\mathbf{D}$ of $\chi^2 = 1383.72990954499$. Figure 3.2 displays the true patterns used to create the SimpSim data, stored in `SimpSim.P`.

Figure 3.4 shows the results from running CoGAPS on the SimpSim data.

3.2 CoGAPS

CoGAPS infers coordinated activity in gene sets active in each row of the pattern matrix $\mathbf{P}$ found by GAPS in a single step, by running both `GAPS` and then performing the statistical analysis of `calcCoGAPSStat`. Specifically, CoGAPS computes a Z-score based statistic on each column of the $\mathbf{A}$ matrix developed in (5). The resulting Z-score for pattern p and gene set i, $\mathcal{G}_i$, with G elements is given by

$$Z_{i,p} = \frac{1}{G} \sum_{g \in \mathcal{G}_i} \frac{\mathbf{A}_{gp}}{\mathbf{Asd}_{gp}} \tag{3.2}$$

where g indexes the genes in the set and $\mathbf{Asd}_{gp}$ is the standard deviation of $\mathbf{A}_{gp}$ obtained from the MCMC sampling in GAPS. CoGAPS then uses random sample tests to convert the Z-scores from eq. (3.2) to p values for each gene set.

3.2.1 Methods

The CoGAPS algorithm is run by calling the `CoGAPS` function in the CoGAPS R package as follows:

```
> results <- CoGAPS(data, unc, GStoGenes, nFactor = "5", nEquil=10000,
                    nSample=10000, nOutR=1000, output_atomic="false",
```

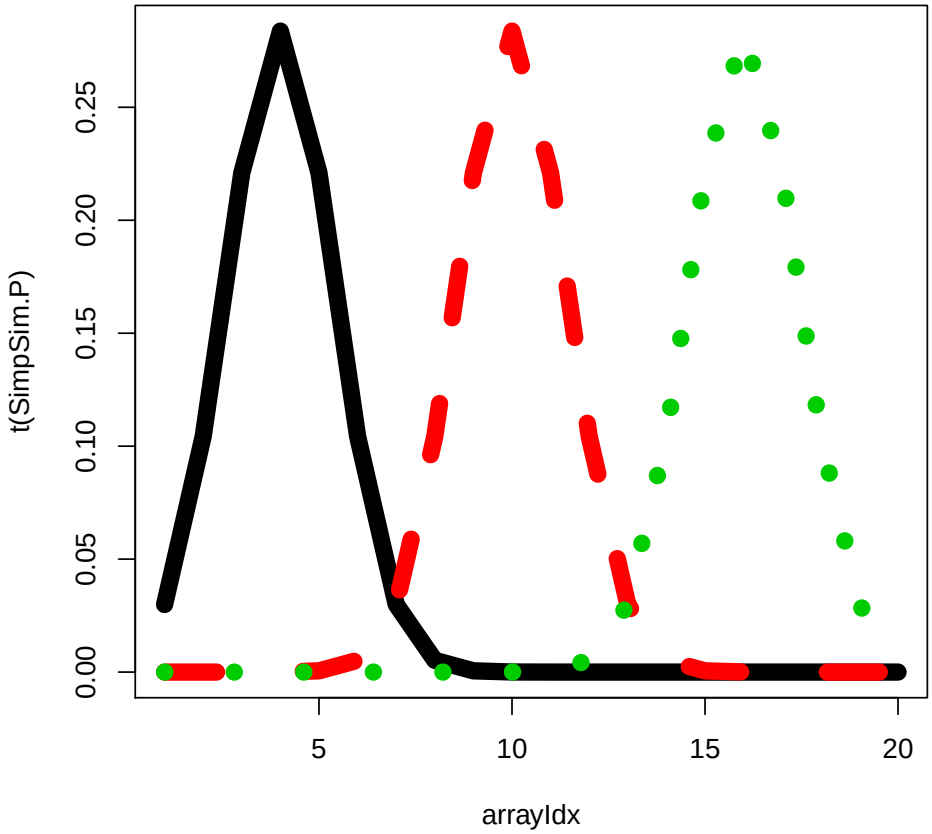


Figure 3.2: Known true patterns used to generate SimpSim data.

```
simulation_id="simulation", plot=TRUE, nPerm=500,
 alphaA = "0.01",  nMaxA = "100000",
 max_gibbmass_paraA = "100.0", lambdaA_scale_factor = "1.0",
 alphaP = "0.01", nMaxP = "100000",
 max_gibbmass_paraP = "100.0", lambdaP_scale_factor = "1.0")
```

Input Arguments

... Input arguments from GAPS.

GStoGenes List or data frame containing the genes in each gene set. If a list, gene set names are the list names and corresponding elements are the names of genes contained in each set. If a data frame, gene set names are in the first column and corresponding gene names are listed in rows beneath each gene set name.

nPerm Number of permutations used for the null distribution in the gene set statistic. (optional; default=500).

plot Use `plotGAPS` to plot results from the run of `GAPS` within `CoGAPS`.

List Items in Function Output

... Output list from GAPS except vectors of sample atom numbers and individual χ^2 values.

D The input data matrix.

Sigma The input standard deviation matrix.

GSUpreg p-values for upregulation of each gene set in each pattern.

GSDownreg p-values for downregulation of each gene set in each pattern.

GSActEst Conversion of p-values to activity estimates of each gene set in each pattern (see (5) for details on conversion. Essentially values near 1 indicate high activity for the set and near -1 indicate low activity.

The CoGAPS algorithm can also be run manually by first running the GAPS algorithm described in Section 3.1 and then calling the function `calcCoGAPSStat` as follows:

```
> calcCoGAPSStat(Amean, Asd, GStoGenes, numPerm=500)
```

The input arguments for `calcCoGAPSStat` are as described in the previous sections. This function will output a list containing `GSUpreg`, `GSDownreg`, and `GSActEst`.

3.2.2 Example: Simulated data

In this example, we have simulated data in SimpSim with three known patterns (SimpSim.P) and corresponding amplitude (SimpSim.A) with specified activity in two gene sets (GSets). In this data set, each gene set is overexpressed in two of the simulated patterns and underexpressed in one.

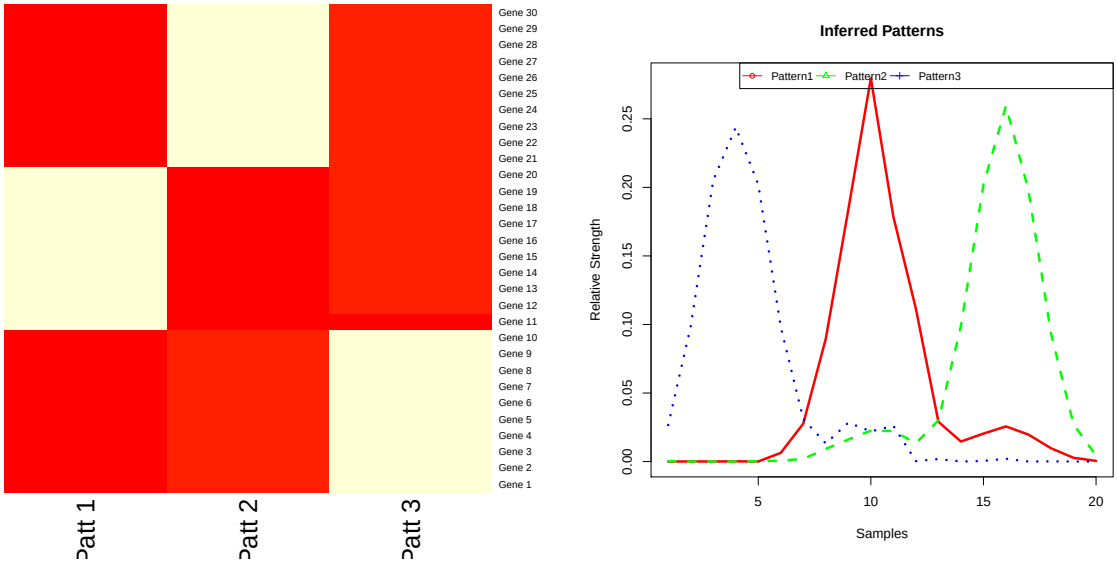
```
> library('CoGAPS')
> data('SimpSim')
> nIter <- 5000
> results <- CoGAPS(data=SimpSim.D, unc=SimpSim.S,
+                   GStoGenes=GSets,
+                   nFactor=3,
+                   nEquil=nIter, nSample=nIter,
+                   plot=FALSE)

Equil:1000 of 5000, Atoms:72(49)  Chi2 = 1681.34
Equil:2000 of 5000, Atoms:73(63)  Chi2 = 1606.52
Equil:3000 of 5000, Atoms:80(68)  Chi2 = 1529.97
Equil:4000 of 5000, Atoms:78(67)  Chi2 = 1506.9
Equil:5000 of 5000, Atoms:75(69)  Chi2 = 1538.77
Samp: 1000 of 5000, Atoms:83(73)  Chi2 = 1534.79
Samp: 2000 of 5000, Atoms:81(72)  Chi2 = 1520.98
Samp: 3000 of 5000, Atoms:72(66)  Chi2 = 1513.51
Samp: 4000 of 5000, Atoms:86(72)  Chi2 = 1541.79
Samp: 5000 of 5000, Atoms:75(74)  Chi2 = 1546.8
 *** Check value of final chi2: 1546.8 ****
[1] "Chi-Squared of Mean: 1412.69421179564"

> plotGAPS(results$Amean, results$Pmean, 'GSFigs')
```

```
null device
          1
```

Figure 3.3 shows the results from running CoGAPS on the GIST data in (5) with the option plot set to true.



(a) Inferred amplitude matrix

(b) Inferred patterns

Figure 3.3: Results from GAPS on data of simulated gene set data.

Moreover, the gene set activity is provided in `results$GSActEst` including p-values for upregulation in `results$GSUpreg` and downregulation in `results$GSDownreg`.

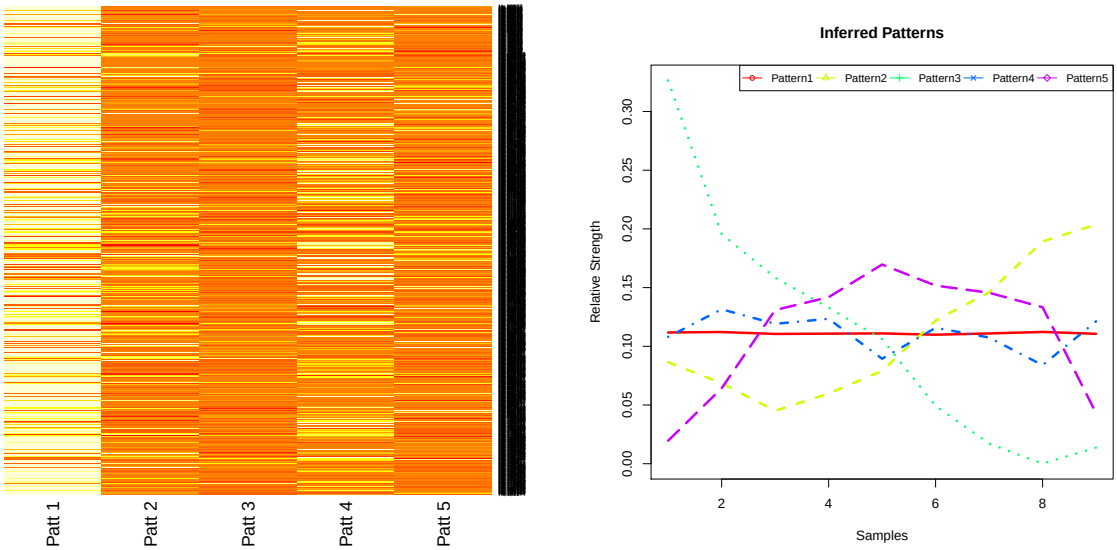
3.2.3 Example: GIST Analysis

We also provide the code that repeats the CoGAPS analysis of GIST data (GIST_TS_20084) with gene sets defined by transcription factors (TFGSList), as in the DESIDE analysis of (5). To minimize time in running the vignette, this is not done live. However, running the code will complete the analysis and require on the order of an hour on a fast machine.

```
> library('CoGAPS')
> data('GIST_TS_20084')
> data('TFGSList')
> nIter <- 10000
> results <- CoGAPS(GIST.D, GIST.S, tf2ugFC,
+                   nFactor=5,
+                   nEquil=nIter, nSample=nIter,
+                   plot=FALSE)
> plotGAPS(results$Amean, results$Pmean, 'GISTFigs')
```

Figure 3.4 shows the results from running CoGAPS on the GIST data in (5) with the option plot set to true.

Moreover, the gene set activity is provided in `results$GSActEst` including p-values for upregulation in `results$GSUpreg` and downregulation in `results$GSDownreg`.



(a) Inferred amplitude matrix

(b) Inferred patterns

Figure 3.4: Results from GAPS on data of simulated gene set data.

3.3 Using CoGAPS-based statistics to infer gene membership in annotated gene sets

As we describe in the previous section, the GAPS matrix factorization can be used to infer gene set activity in each pattern using the function `calcCoGAPSStat` (5). The `computeGeneGSProb` function extends this gene set statistic to compute a statistic quantifying the likely membership of each gene annotated to a set based upon its inferred activity (2). This statistic is formulated by comparing the expression pattern computed with CoGAPS of a given gene g annotated as a member of $\mathcal{G}$ to the common expression pattern of all annotated members of $\mathcal{G}$. This similarity is quantified with the following summary statistic

$$S_{g,\mathcal{G}} = \frac{\sum_p -log\left(\Pr_{\mathcal{G}p}\right) \mathbf{A}_{gp}/\mathbf{Asd}_{gp}}{\sum_p -log\left(\Pr_{\mathcal{G}p}\right)}, \tag{3.3}$$

where $\Pr_{\mathcal{G}p}$ is the probability of upregulation of the geneset, returned from `calcCoGAPSStat` as `GSActEst` based upon eq. (3.2). Similar to the gene set statistics, p-values for the set membership are computed with permutation tests that compare the value of $S_{g,\mathcal{G}}$ from eq. (3.3) to the statistic formulated for a random gene set of the same size that also contains gene g.

The set membership statistic is computed from the results from the GAPS matrix factorization, computed with either the `GAPS` function described in Section **??** or the `CoGAPS` function described in Section 3.2 as follows:

```
> computeGeneGSProb(Amean, Asd, GStoGenes, numPerm=500)
```

Input Arguments

Amean Mean of the amplitude matrix estimated from the GAPS matrix factorization

Asd Standard deviation of the amplitude matrix estimated from the GAPS matrix factorization

GStoGenes List or data frame containing the genes in each gene set. If a list, gene set names are the list names and corresponding elements are the names of genes contained in each set. If a data frame, gene set names are in the first column and corresponding gene names are listed in rows beneath each gene set name.

nPerm Number of permutations used for the null distribution in the gene set and set membership statistics. (optional; default=500).

Function Output

p-value of set membership for each gene specified in `GStoGenes`.

3.3.1 Example on GIST Data

Although not run in the interest of installation time, the following examples were used to generate some of the results in (2), with the complete analysis code available from `http://astor.som.jhmi.edu/~ejfertig/ejfertig/Publications.html`.

This example refines transcription factor targets annotated in TRANSFAC (TFGSList) to identify context-specific targets from gene expression data (GIST_TS_20084)from (5).

```
> # define transcription factors of interest based on Ochs et al. (2009)
> TFs <- c("c.Jun", 'NF.kappaB', 'Smad4', "STAT3", "Elk.1", "c.Myc", "E2F.1",
+          "AP.1", "CREB", "FOXO", "p53", "Sp1")
> # take the results from the previously run analysis
> # set membership statistics
> permTFStats <- list()
> for (tf in TFs) {
+      genes <- levels(tf2ugFC[,tf])
+      genes <- genes[2:length(genes)]
+      permTFStats[[tf]] <- computeGeneTFProb(Amean = results$Amean,
+                                             Asd = results$Asd, genes)
+ }
```

Chapter 4

Feedback

Please send feedback to Elana Fertig `ejfertig@jhmi.edu` or Michael Ochs `ochsm@tcnj.edu`.

If you want to send a bug report, please first try to reproduce the error. The code is stochastic and we have seen many transient errors arising in the boost libraries which rarely repeat. Send the data and please describe what you think should have happened, and what did happen.

Chapter 5

Acknowledgments

This new version of CoGAPS relies on extensive recoding by Waishing Lee in C++ and work for C++ streamlining and R integration by Ondrej Maxian and Conor Kelton. John Stansfield provided additional R code for visualization. Special thanks to paper co-authors Jie Ding, Alexander V. Favorov, and Giovanni Parmigiani for statistical advice in developing the GAPS / CoGAPS algorithms. Additional thanks to Simina M. Boca, Ludmila V. Danilova, Genevieve Stein-O'Brien, Jeffrey Leek, and Svitlana Tyekcheva for their useful feedback.

This work was funded by NIH/NLM R21LM009382, NIH/NLM R01LM011000 and NSF Grant 0342111.

Bibliography

[1] EJ Fertig, J Ding, AV Favorov, G Parmigiani, and MF Ochs. CoGAPS: an R/C++ package to identify patterns and biological process activity in transcriptomic data. *Bioinformatics*, 26(21):2792–3, Nov 2010.

[2] EJ Fertig, AV Favorov, and MF Ochs. Identifying context-specific transcription factor targets from prior knowledge and gene expression data. In *IEEE International Conference on Bioinformatics and Biomedicine*, number B310, Philadelphia, PA, USA, 2012.

[3] MF Ochs. Bayesian decomposition. In G Parmigiani, E Garrett, R Irizarry, and S Zeger, editors, *The Analysis of Gene Expression Data: Methods and Software*. Springer-Verlag, New York, 2003.

[4] M.F. Ochs. *The Analysis of Gene Expression Data The Analysis of Gene Expression Data: Methods and Software*, pages 388–408. Statistics for Biology and Health. Springer-Verlag, London, 2006.

[5] M.F. Ochs, L. Rink, C. Tarn, S. Mburu, T. Taguchi, B. Eisenberg, and A.K. Godwin. Detection of treatment-induced changes in signaling pathways in gastrointestinal stromal tumors using transcriptomic data. *Cancer Res*, 69(23):9125–9132, 2009.

[6] S. Sibisi and J. Skilling. Prior distributions on measure space. *Journal of the Royal Statistical Society, B*, 59(1):217–235, 1997.