joda Package Vignette

Ewa Szczurek

October 14, 2013

Contents

1 Introduction

This document gives a short introduction to gene deregulation analysis using the **R** package *joda*. The package implements an algorithm called JODA, which is designed to quantify how strongly regulation of genes changes between two different cell populations. JODA analyzes a given set of regulators, which are interconnected in a common signaling pathway. The algorithm utilizes two types of input: (1) regulator knockdown data, as well as (2) knowledge about pathways that interconnect the regulators and about genes that are expected to be regulated by those pathways in both cells. For each regulator the analysis yields gene deregulation scores. The scores reflect how strongly the effect of the regulator's knockdown on the genes changed between the cells.

The input to the algorithm and processing steps and are illustrated in Figure 1.

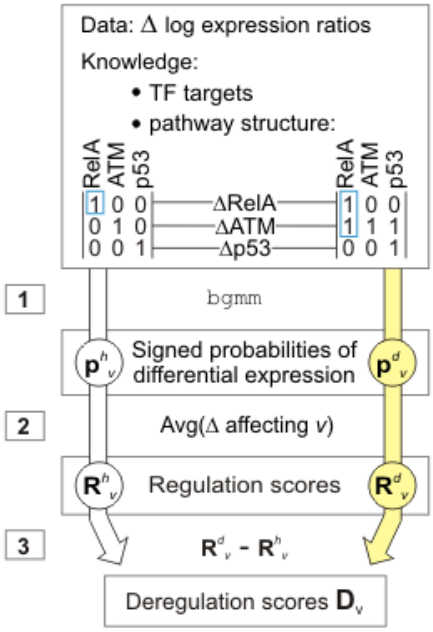

Figure 1: The input and steps of the JODA algorithm

2 Input to the JODA algorithm.

■inputdata■

We present the application of the *joda* package to deregulation analysis on an example dataset of Elkon *et al.*, 2005. The JODA algorithm computes deregulation scores for a given set of regulators (in the example dataset ATM, RelA and p53). The input data has to contain gene expression measurements upon knockdown of each regulator in each cell population. The example input is stored in the `damage` dataset.

```
> library(joda)
> data(damage)
```

The example dataset contains transcriptional effects of silencing the regulators ATM, RelA and p53, performed on two cell populations, referred to as the healthy and the damaged cells (together six knockdown experiments). The data for each knockdown experiment are preprocessed log expression ratios of a regulator knockdown versus control in a given cell population. The damaged cells (denoted *d*) are a population of cells that were treated with

a DNA-damage inducing drug neocarzinostatin (NCS). NCS triggers a cellular pathway, where the central kinase ATM signals down to transcription factors RelA and p53. This pathway is inactive in the healthy cells (denoted h).

The knockdown data are (in two data frames) `data.healthy` and `data.damage`.

```
> head(data.healthy)
```

```
                  ATM        RelA         p53
1007_s_at -0.65999832 -0.70467161 -0.44830942
1053_at    0.15601284  0.06511664  0.08027362
117_at     0.01485424 -0.11931042 -0.11213425
121_at    -0.22150756 -0.25566903 -0.23788410
1255_g_at -0.06630502 -0.04021388 -0.06735836
1294_at   -0.05957592 -0.13969689 -0.17432387
```

```
> head(data.damage)
```

```
                  ATM         RelA         p53
1007_s_at -0.40918959 -0.573389287 -0.28255737
1053_at    0.04582026  0.005923906 -0.02656703
117_at    -0.06416793 -0.136964585 -0.04523239
121_at    -0.02032572  0.027913098  0.01190413
1255_g_at  0.03004704  0.001573532  0.09873156
1294_at   -0.02158214 -0.072783759 -0.03449833
```

■inputknowledge■

Additional to the knockdown data, the input to the JODA algorithm consists of two kinds of knowledge. First, topologies of the pathways that connect the given set of regulators and are active in the two cell populations. This knowledge is formalized in two binary matrix models, one per each cell population. For each regulator, the model defines a set of knockdown experiments which affect this regulator's activity. Example model matrices for the ATM pathway in the healthy and in the damaged cells are shown in Figure 1 and stored in objects `model.healthy` and `model.damage`.

```
> print(model.healthy)
```

```
     ATM RelA p53
ATM    1    0   0
RelA   0    1   0
p53    0    0   1
```

```
> print(model.damage)

     ATM RelA p53
ATM    1    1   1
RelA   0    1   0
p53    0    0   1
```

Here, the first row of `model.damage` tells that in the damaged cells the knockdown of ATM affects not only ATM, but also RelA and p53, which are downstream of ATM.

The second kind of knowledge are regulator-gene relations, given for some regulators, which are also transcription factors (shortly, TFs), and for some remaining genes. This knowledge is cell-population specific. The known TF targets are expected (but rarely sure) to show an effect to the knockdown experiments, and serve as examples of genes that are differentially expressed upon their TF knockdown. This type of knowledge is uncertain and is given as probabilities. The example input stores certainties (beliefs) about known targets of RelA or p53 being differentially expressed upon their regulator knockdown in the two cell populations.

```
> str(beliefs.healthy)

List of 2
 $ RelA: num [1:60, 1:2] 0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.95 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:60] "200762_at" "201329_s_at" "201642_at" "201719_s_at" ...
  .. ..$ : chr [1:2] "differential" "unchanged"
 $ p53 : num [1:85, 1:2] 0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.95 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:85] "200632_s_at" "200766_at" "201041_s_at" "201069_at" ...
  .. ..$ : chr [1:2] "differential" "unchanged"

> str(beliefs.damage)

List of 1
 $ p53: num [1:33, 1:2] 0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.95 0.95 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:33] "200736_s_at" "200802_at" "200921_s_at" "201202_at" ...
  .. ..$ : chr [1:2] "differential" "unchanged"

> head(beliefs.healthy[["p53"]])
```

```
            differential unchanged
200632_s_at         0.95      0.05
200766_at           0.95      0.05
201041_s_at         0.95      0.05
201069_at           0.95      0.05
201099_at           0.95      0.05
201426_s_at         0.95      0.05
```

■steps■

3 Processing steps

The algorithm proceeds in three steps. In each step, we compute the following scores for each regulator:

1. For each gene, in each cell population: signed probabilities of differential expression upon the regulator's knockdown,
2. For each gene, in each cell population: regulation scores,
3. For each gene: deregulation scores.

■step1■

3.1 Step 1: computing signed differential expression probabilities

In the first step, the input data from each regulator's knockdown is processed to estimate the effect of the knockdown on the genes. To this end, JODA utilizes our belief-based differential expression analysis method, implemented in an **R** package *bgmm*. The method assigns each gene a probability that it was differentially expressed in the experiment. In this step, the knowledge about the known TF targets is used. To improve the estimation, the known targets of the perturbed regulator are given a high prior of differential expression in the experiment. Each returned probability is signed, i.e., multiplied by 1 or -1 to indicate whether the effect of the knockdown was up- or down-regulation.

To compute the signed differential gene expression probabilities for each knockdown experiment, the `differential.probs` function is used. When the argument `verbose` is set to `TRUE` the function prints the parameters of the fitted two-component models (one component for the differential and

one for the unchanged genes). Setting the argument `plot.it` to `TRUE` yields a plot of the models' components. For the knockdown data and knowledge in the healthy cells the function call reads (the generated plot is presented in Figure 2):

```
> probs.healthy=differential.probs(data=data.healthy, beliefs=beliefs.healthy, verbos

joda: Input correctly defined

Inferring probabilities of differential expression under the knockdown of  ATM ...
Applying unsupervised mixture modeling
The parameters of the model for ATM:
                    differential   unchanged
Mixing proportions:   0.35661277  0.64338723
Means:                0.07343407 -0.03710922
Variances:            0.14760688  0.02181053

Inferring probabilities of differential expression under the knockdown of  RelA ...
Applying belief-based mixture modeling
The parameters of the model for RelA:
                    differential    unchanged
Mixing proportions:   0.52374510  0.476254896
Means:                0.07521450 -0.074466149
Variances:            0.09389958  0.008102053

Inferring probabilities of differential expression under the knockdown of  p53 ...
Applying belief-based mixture modeling
The parameters of the model for p53:
                    differential    unchanged
Mixing proportions:   0.50634787  0.493652126
Means:                0.05503342 -0.072353721
Variances:            0.04807592  0.007986199
```

For the knockdown data and knowledge in the damaged cells the function call reads (the generated plot is presented in Figure 3):

```
> probs.damage=differential.probs(data=data.damage, beliefs=beliefs.damage, verbose=T

joda: Input correctly defined

Inferring probabilities of differential expression under the knockdown of  ATM ...
```

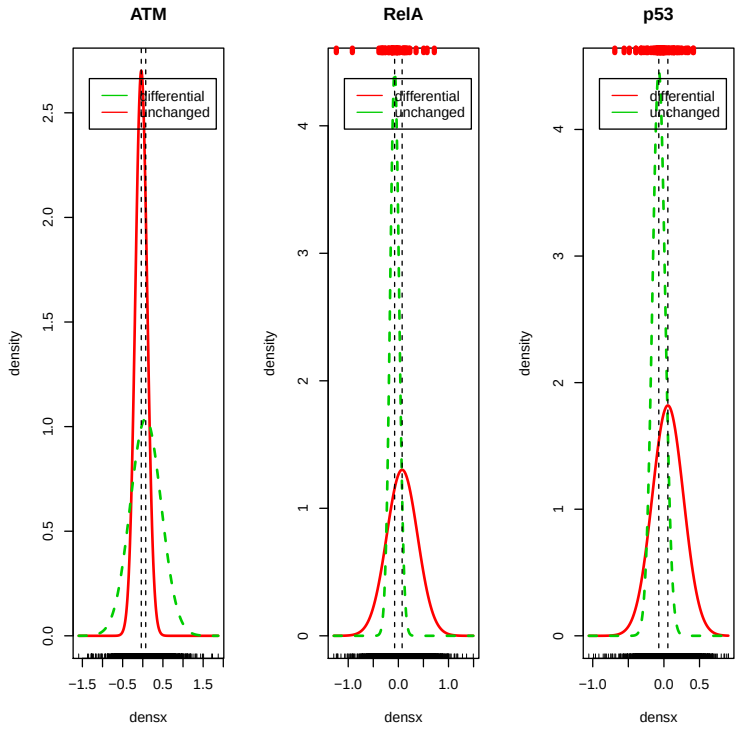

Figure 2: Output of the differential.probs function applied to the data and knowledge for the healthy cells.

```
Applying unsupervised mixture modeling
The parameters of the model for ATM:
                    differential    unchanged
Mixing proportions:   0.35358709  0.646412907
Means:                0.02230456 -0.002159813
Variances:            0.16242875  0.015909231

Inferring probabilities of differential expression under the knockdown of  RelA ...
Applying unsupervised mixture modeling
The parameters of the model for RelA:
                    differential   unchanged
Mixing proportions:  0.268037955 0.731962045
Means:              -0.006308146 0.004366857
```

```
Variances:           0.124130999 0.018442192

Inferring probabilities of differential expression under the knockdown of  p53 ...
Applying belief-based mixture modeling
The parameters of the model for p53:
                    differential    unchanged
Mixing proportions:   0.26116122  0.738838780
Means:               -0.01600267 -0.002041848
Variances:            0.06562652  0.010953522
```

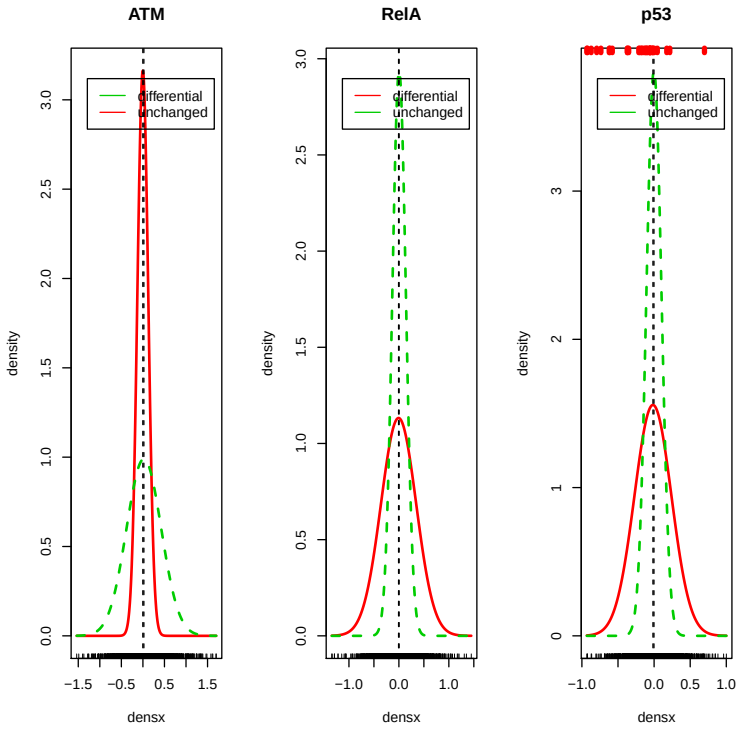

Figure 3: Output of the `differential.probs` function applied to the data and knowledge for the damaged cells.

■step2■

3.2 Step 2: computing regulation scores

In the second step, for each regulator and for each cell population, we obtain a vector of regulation scores that quantify the effect of the regulator on the genes in this population. In this step, the given pathway models are used. For a given cell population and regulator, regulation scores are computed as an average over the probabilities of differential expression in all knockdown experiments that affect this regulator in this cell population. The affecting experiments are defined using the pathway model as both the knockdown of the regulator itself, and knockdowns of its upstream activators in the pathway. For example, the regulation scores for RelA in the damaged cells are an average of signed probabilities for the knockdowns of RelA and of its upstream activator ATM. In the healthy cells, only its own knockdown affects RelA, and its regulation scores are the same as its signed probabilities.

```
> regulation.healthy= regulation.scores(probs.healthy, model.healthy, TRUE)

joda: the model is correctly defined
joda: getting the regulation scores...

> head(regulation.healthy)

                ATM       RelA        p53
1007_s_at -0.9960349 -1.0000000 -0.9952370
1053_at    0.3286554  0.5179488  0.6409970
117_at     0.1830380 -0.2301634 -0.2566034
121_at    -0.2570529 -0.5776955 -0.4877506
1255_g_at -0.1689931 -0.2444295 -0.2637883
1294_at   -0.1687471 -0.2472470 -0.3168580

> regulation.damage= regulation.scores(probs.damage, model.damage, TRUE)

joda: the model is correctly defined
joda: getting the regulation scores...

> head(regulation.damage)

                ATM        RelA         p53
1007_s_at -0.9462780 -0.97162479 -0.84971693
1053_at    0.1552072  0.13941915  0.01302862
117_at    -0.1588039 -0.17172624 -0.14696014
121_at    -0.1467651 -0.01097676 -0.01012653
1255_g_at  0.1502600  0.13697255  0.16110760
1294_at   -0.1469091 -0.14350978 -0.13909837
```

■step3■

3.3 Step 3: computing deregulation scores

In the third step, to quantify deregulation of genes by a given regulator, we compute a vector of deregulation scores as a difference between the regulation scores for this regulator in the two cell populations.

```
> deregulation= deregulation.scores(reg.scores1=regulation.healthy, reg.scores2=regul

joda: calculating deregulation scores
subtracting reg.scores1 from reg.scores2

> head(deregulation)

                  ATM        RelA        p53
1007_s_at  0.04975690  0.02837521  0.1455201
1053_at   -0.17344820 -0.37852969 -0.6279684
117_at    -0.34184195  0.05843713  0.1096432
121_at     0.11028783  0.56671869  0.4776241
1255_g_at  0.31925309  0.38140201  0.4248959
1294_at    0.02183797  0.10373723  0.1777596
```

Genes which obtain negative deregulation scores are interpreted as more activated by the regulator in the cells corresponding to the second argument (here, the damaged cells). Genes more activated in the other cells (here, healthy) obtain positive scores. For example, to see the top genes most activated by p53 in the damaged cells, we sort the genes by their deregulation scores:

```
> head(rownames(deregulation)[order(deregulation[,"p53"])])

[1] "200921_s_at" "201939_at"   "208811_s_at" "202081_at"
[5] "221962_s_at" "200668_s_at"
```