A guide to using *ISOpureR*

Catalina V Anghel

November 16, 2014

1 Background

ISOpure is a statistical model for deconvolution of mRNA microarray profiles of mixed tumour samples into the constituent normal and cancer profiles, as well as estimating the proportion of cancer content. The model was developed by Quon et al. in [1] and first implemented in MATLAB. The R package *ISOpureR* keeps as close to the MATLAB implementation as possible. We will use 'ISOpure' to refer to the algorithm in general and *ISOpureR* to refer to the R package.

The full description of the model details (inputs, outputs, computation) is in [1]. A summary of the changes made for the R implementation is given in [3]. In particular, *ISOpureR* is not yet tested to be back-compatible with ISOLATE [2], the precursor to ISOpure.

Briefly, the inputs are as follows:

- A matrix where the columns $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N$ represent the tumour microarray profiles of N patients, where the preprocessing of the data is described in [1]. In particular, the intensities should *not* be log-transformed. The size of the matrix is $G \times N$ where G is number of transcripts/features and N is the number of patients.
- A matrix where the columns $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_K$ represent the microarray profiles from K normal (*i.e.* healthy tissue) samples. The size of the matrix will be $G \times K$, where K is expected to be less than N.

The ISOpure algorithm runs in two steps. The outputs of *ISOpureR* at each step are lists that we have called `ISOpureS1model` and `ISOpureS2model`, respectively. They contain all the intermediate and hyper-parameters estimated by the models as well as the desired outputs. The most important entries are the following.

- The tumour 'purity' estimates, `ISOpureS1model$alphapurities` and `ISOpureS2model$alphapurities`. (The α's are estimated in Step 1; the values of `alphapurities` from Step 1 are transferred to the model in Step 2, so that `ISOpureS1model$alphapurities` and `ISOpureS2model$alphapurities` will be identical.) These are numerical vectors containing N entries, $\alpha_1, \alpha_2, \ldots, \alpha_N$, of the estimated proportion of RNA in the tumour sample that was contributed by the cancer cells, for each patient.
- A matrix `ISOpureS2model$cc_cancerprofiles` where the columns $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N$ represent the purified cancer profiles corresponding to each mixed tumour profile. The size of the matrix will be $G \times N$, as for the tumour profiles.

The idea is that a patient's particular tumour sample, $\mathbf{t}_n$ is estimated as a combination of a cancer profile, $\mathbf{c}_n$ and a normal healthy profile $\mathbf{h}_n$

$$\mathbf{t}_n = \alpha_n \mathbf{c}_n + (1 - \alpha_n)\mathbf{h}_n.$$

Since patients do not necessarily have a matched tumour-normal sample, and the normal sample itself may have a different composition from the healthy component inside the tumour sample, $\mathbf{h}_n$ is estimated from the reference normal samples $\mathbf{b}_i$'s.

2 Applying ISOpureR to an example

We will show how to use *ISOpureR* with a small part of the lung adenocarcinoma expression data from Beer et al. [4], which is included with the package. Although the full dataset contains measured expression levels of 5151 transcripts in 86 patients, only data from 2500 transcripts and 43 patients is included with *ISOpureR* to reduce the file size. The dataset also contains 10 reference healthy samples.

2.1 Load and format data

The first step is to load the data and make sure that both the tumour data and the normal data are in matrix form, with the required dimensions.

```
# Load the library and the data included with the package
# Data is not the full Beer dataset due to memory constraints
library(ISOpureR);
path.to.data <- paste0(file.path(system.file(package = "ISOpureR"), 'extdata', 'Beer'));
load(file.path(path.to.data , 'beer.normaldata.2500.transcripts.RData'))
load(file.path(path.to.data , 'beer.tumordata.2500.transcripts.43.patients.RData'))

# Check what the data looks like
# The tumordata is rather large, so just look at the normal data for this example
str(beer.normaldata)

## 'data.frame': 2500 obs. of  10 variables:
##  $ V1 : num  231 1200 159 850 195 ...
##  $ V2 : num  222 1434 154 1072 188 ...
##  $ V3 : num  214 1185 173 1051 164 ...
##  $ V4 : num  221 1174 152 1077 175 ...
##  $ V5 : num  213 1055 165 1044 182 ...
##  $ V6 : num  222 1289 142 980 179 ...
##  $ V7 : num  191 1056 148 907 156 ...
##  $ V8 : num  223 1333 162 964 199 ...
##  $ V9 : num  234 1140 150 880 181 ...
##  $ V10: num  226 1086 145 997 177 ...

# Make sure that everything is in matrix form
beer.normaldata <- as.matrix(beer.normaldata);
beer.tumordata <- as.matrix(beer.tumordata);

# Check what the data looks like
str(beer.normaldata)

##  num [1:2500, 1:10] 231 1200 159 850 195 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:2500] "1" "2" "3" "4" ...
##   ..$ : chr [1:10] "V1" "V2" "V3" "V4" ...

str(beer.tumordata)

##  num [1:2500, 1:43] 218 1399 155 893 170 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:2500] "1" "2" "3" "4" ...
##   ..$ : chr [1:43] "V1" "V2" "V3" "V4" ...
```

2.2 Run ISOpure Step 1

The ISOpure model runs in two steps. To perform the first step of ISOpure, which will estimate the proportion of cancer cells, α_n, for each patient, type the following:

```
# For reproducible results, set the random seed
set.seed(123);
# Run ISOpureR Step 1
ISOpureS1model <- ISOpureS1.model_core.learnmodel(beer.tumordata, beer.normaldata);
```

The first step runs a long time (*i.e.* about an hour; for the full Beer dataset it takes a few hours). The output to the screen will look something like this:

```
# [1] "---------------------"
# [1] "Initializing..."
# [1] "MIN_KAPPA set to  24446.9895636363"
# [1] "--- optimizing mm..."
# [1] "--- optimizing theta..."
# [1] "--- optimizing vv..."
# [1] "--- optimizing kappa..."
# [1] "--- optimizing omega..."
# [1] "Total log likelihood: -868015829.449002"
# [1] "iter:  1 / 35 , loglikelihood:  -868015829.449002 , change:  Inf"
# [1] "--- optimizing mm..."
# [1] "--- optimizing theta..."
# [1] "--- optimizing vv..."
# [1] "--- optimizing kappa..."
# [1] "--- optimizing omega..."
# [1] "Total log likelihood: -867925503.209553"
# [1] "iter:  2 / 35 , loglikelihood:  -867925503.209553 , change:  0.000104071419856478"
# [1] "--- optimizing mm..."
# [1] "--- optimizing theta..."
# [1] "--- optimizing vv..."
# [1] "--- optimizing kappa..."
# [1] "--- optimizing omega..."
# [1] "Total log likelihood: -867906153.934941"
# [1] "iter:  3 / 35 , loglikelihood:  -867906153.934941 , change:  2.22942014227049e-05"
```

The optimization of the loglikelihood will run for at least 35 iterations, and if the change in loglikelihood is greater than 10^{-7}, up to 100 iterations. At the end of this process you may see warnings as below.

```
warnings()
# Warning messages:
# 1: In sqrt(B * B - A * d1 * (x2 - x1)) : NaNs produced
# 2: In sqrt(B * B - A * d1 * (x2 - x1)) : NaNs produced
# 3: In sqrt(B * B - A * d1 * (x2 - x1)) : NaNs produced
```

These are nothing to worry about. They are part of the optimization calculations, and when a `NaN` is produced, the algorithm detects that it has not converged and simply adjusts the step size.

The list `ISOpureS1model` which is returned will contain all the information on parameters estimated from the first step. If you would like to see what the list looks like without performing all the calculations, you can load the saved result of Step 1 from the data folder. The most important list entry is vector of estimated fractions of cancer content, the `alphapurities`.

```
# Load the saved ISOpureS1model for this example, if time is an issue
load(file.path(path.to.data , 'beer.ISOpureS1model.2500.transcripts.43.patients.RData'))
ls()

## [1] "ISOpureS1model"  "beer.normaldata" "beer.tumordata"  "path.to.data"

# Check that what ISOpureS1model looks like
str(ISOpureS1model)

## List of 14
##  $ vv                : num [1, 1:11] 1 1 1 1 1 ...
##  $ log_BBtranspose   : num [1:10, 1:2500] -9.4 -9.48 -9.46 -9.47 -9.51 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:10] "V1" "V2" "V3" "V4" ...
##   .. ..$ : chr [1:2500] "1" "2" "3" "4" ...
##  $ PPtranspose       : num [1:10, 1:2500] 8.29e-05 7.61e-05 7.77e-05 7.71e-05 7.38e-05 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:10] "V1" "V2" "V3" "V4" ...
##   .. ..$ : chr [1:2500] "1" "2" "3" "4" ...
##  $ kappa             : num 24447
##  $ theta             : num [1:43, 1:11] 2.12e-01 1.36e-01 1.47e-01 3.05e-06 4.16e-02 ...
##  $ omega             : num [1:10, 1] 7.49e-01 5.02e-57 6.64e-73 2.51e-01 4.67e-63 ...
##  $ log_all_rates     : num [1:11, 1:2500] -9.4 -9.48 -9.46 -9.47 -9.51 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "V1" "V2" "V3" "V4" ...
##   .. ..$ : chr [1:2500] "1" "2" "3" "4" ...
##  $ MIN_KAPPA         : num 24447
##  $ total_loglikelihood: num [1, 1] -8.68e+08
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr ""
##  $ mm_weights        : num [1, 1:2500] -9.46 -7.69 -9.8 -7.88 -9.58 ...
##  $ theta_weights     : num [1:43, 1:11] -3 -3 -3 -3 -3 ...
##  $ omega_weights     : num [1:10, 1] -2.3 -131.6 -168.2 -3.4 -145.5 ...
##  $ mm                : Named num [1:2500] 7.85e-05 4.62e-04 5.59e-05 3.84e-04 7.00e-05 ...
##   ..- attr(*, "names")= chr [1:2500] "1" "2" "3" "4" ...
##  $ alphapurities     : num [1:43] 0.546 0.67 0.656 0.397 0.69 ...

# Look at the alphapurities vector in particular
ISOpureS1model$alphapurities

##  [1] 0.5462532 0.6701689 0.6560692 0.3971465 0.6900217 0.4697611 0.6332442 0.7640138
##  [9] 0.4865317 0.8176220 0.5937556 0.8152299 0.5703189 0.5351764 0.5279974 0.3299588
## [17] 0.7496806 0.5286267 0.5108075 0.4610482 0.2749277 0.6213010 0.7164460 0.7617311
## [25] 0.5369314 0.6746044 0.5660559 0.6806280 0.5235685 0.5615519 0.4809694 0.6790250
## [33] 0.5922468 0.7219368 0.5378330 0.8059887 0.9158957 0.5681175 0.5498978 0.8379697
## [41] 0.6464301 0.7058618 0.6770253
```

2.3 Run ISOpure Step 2

Once Step 1 is complete, to perform the second step of ISOpure, which will estimate the patient-specific cancer mRNA expression profiles, call the following function:

```
# For reproducible results, set the random seed
set.seed(456);

# Run ISOpureR Step 2
ISOpureS2model <- ISOpureS2.model_core.learnmodel(
                                        beer.tumordata,
                                        beer.normaldata,
                                        ISOpureS1model
                                        );
```

The screen output will look very similar to the output in Step 1.

```
# [1] "---------------------"
# [1] "Initializing..."
# [1] "MIN_KAPPA set to  2106511.36827368"
# [1] "--- optimizing cc..."
# [1] "--- optimizing theta..."
# [1] "--- optimizing vv..."
# [1] "--- optimizing kappa..."
# [1] "Total log likelihood: -866065642.895818"
# [1] "iter:  1 / 35 , loglikelihood:  -866065642.895818 , change:  Inf"
# [1] "--- optimizing cc..."
# [1] "--- optimizing theta..."
# [1] "--- optimizing vv..."
# [1] "--- optimizing kappa..."
# [1] "Total log likelihood: -864323005.847771"
# [1] "iter:  2 / 35 , loglikelihood:  -864323005.847771 , change:  0.00201618727750749"
```

Again, `ISOpureS2model` which is returned by the function `ISOpureS2.model_core.learnmodel` will contain all the information on parameters estimated in Step 2. The matrix `ISOpureS2model$cc_cancerprofiles` will contain the patient-specific cancer profiles and is of the same dimension as the `tumordata`. It is also of the same scale, (*i.e.* although ISOpureS2 treats purified cancer profiles as parameters of a multinomial distribution, we re-scale them to be on the same scale as the input tumour profiles). The n-th column corresponds to the profile for the n-th patient.

```
# Load the saved ISOpureS2model for this example, if time is an issue
# Note that the entries in the model have been rounded to 5 significant digits
# (due to memory constraints)
load(file.path(
                path.to.data,
                'beer.ISOpureS2model.2500.transcripts.43.patients.rounded.RData'
                )
        )

# Check that what ISOpureS2model looks like
str(ISOpureS2model)

## List of 13
##  $ vv                 : num [1, 1:11] 1 1 1 1 1 1 1 1 1 1 ...
##  $ log_BBtranspose    : num [1:10, 1:2500] -9.4 -9.48 -9.46 -9.47 -9.51 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:10] "V1" "V2" "V3" "V4" ...
##   .. ..$ : chr [1:2500] "1" "2" "3" "4" ...
##  $ PPtranspose        : num [1, 1:2500] 7.85e-05 4.62e-04 5.59e-05 3.84e-04 7.00e-05 ...
##  $ kappa              : num [1, 1:43] 2106511 2106511 2106511 2106511 2106511 ...
##  $ omega              : num [1:43, 1] 1 1 1 1 1 1 1 1 1 1 ...
##  $ log_cc             : num [1:43, 1:2500] -8.66 -8.66 -8.71 -8.67 -8.66 ...
##  $ MIN_KAPPA          : num 2106511
##  $ total_loglikelihood: num [1, 1] -8.64e+08
##  $ theta              : num [1:43, 1:11] 0.21461 0.15943 0.16297 0.00384 0.0497 ...
##  $ cc_weights         : num [1:43, 1:2500] 7.85e-05 7.85e-05 7.85e-05 7.85e-05 7.85e-05 ...
##  $ theta_weights      : num [1:43, 1:10] -1.55 -1.99 -1.92 -12.7 -3.18 ...
##  $ alphapurities      : num [1:43] 0.546 0.67 0.656 0.397 0.69 ...
##  $ cc_cancerprofiles  : num [1:2500, 1:43] 467 1331 153 986 180 ...

# Check what the cancer profiles look like
str(ISOpureS2model$cc_cancerprofiles)

##  num [1:2500, 1:43] 467 1331 153 986 180 ...

# Look at the first entries in the cancer profile for a particular patient,
# say patient 3
head(ISOpureS2model$cc_cancerprofiles[ ,3])

## [1]  480.07 1410.30  152.14 1051.70  183.20 1329.90
```

Generating gene signatures

Purified cancer profiles have been shown to generate better prognostic gene signatures compared to mixed tumour profiles [1]. The purified cancer profiles $\mathbf{c}_n$ (rather than the mixed tumour profiles $\mathbf{t}_n$) were used to train a Cox proportional hazards (CPH) model to predict survival data for each patient. To test, another dataset of samples were purified using ISOpure and then used to compute the risk for each patient, using the CPH model parameters learned on the training set.

References

[1] Quon, G., Haider, S., Deshwar, A.G., Cui, A., Boutros, P.C., Morris, Q. Computational pu- rification of individual tumor gene expression profiles leads to significant improvements in prognostic prediction. Genome Medicine, 5:29 (2013). http://www.ncbi.nlm.nih.gov/pubmed/23537167.

[2] Quon, G., Morris, Q. ISOLATE: a computational strategy for identifying the primary origin of cancers using high-throughput sequencing. Bioinformatics, 25:2882-2889 (2009) http://www.ncbi.nlm.nih.gov/pubmed/19542156

[3] (submitted) Anghel C.V., Quon, G. Haider S., Nguyen F., Deshwar A.G., Morris Q.D., Boutros P.C. ISOpureR: an R implementation of a computational purification algorithm of mixed tumour profiles. BMC Bioinformatics. (2014)

[4] Beer, D.G., Kardia, S.L., Huang, C.C., Giordano, T.J., Levin, A.M., Misek, D.E., Lin, L., Chen, G., Gharib, T.G., Thomas, D.G., Lizyness, M.L., Kuick, R., Hayasaka, S., Taylor, J.M., Iannettoni, M.D., Orringer, M.B., Hanash, S. Gene-expression profiles predict survival of patients with lung adenocarcinoma. Nat. Med. 8(8), 816-824 (2002) http://www.ncbi.nlm.nih.gov/pubmed/12118244