INSPEcT - INference of Synthesis, Processing and dEgradation rates in Time-course analysis

Stefano de Pretis

April 7, 2016

Contents

1 Introduction

INSPEcT provides an R/Bioconductor compliant solution for the study of dynamic transcriptional regulatory processes. Based on RNA- and 4sU-seq data, which can be jointly analyzed thanks to a computational normalization routine, *INSPEcT* determines mRNA synthesis, degradation and pre-mRNA processing rates over time for each gene, genome-wide. Finally, the *INSPEcT* modeling framework allows the identification of gene-level transcriptional regulatory mechanisms, determining which combination of synthesis, degradation and processing rates is most likely responsible for the observed mRNA level over time.

INSPEcT is based on the estimation of total mRNA levels, pre-mRNA levels (from RNA-seq), synthesis rates and processing rates (from 4sU-seq), and degradation rates from the combined analysis of these two data types. 4sU-seq is a recent experimental technique developed to measure the concentration of nascent mRNA and for the genome-wide inference of gene-level synthesis rates. During a short pulse (typically few minutes), cells medium is complemented with 4sU, a naturally occurring modified uridine that is incorporated within growing mRNA chains with minimal impact on cell viability. The chains which have incorporated the uridine variant (the newly synthesized ones) can be isolated from the total RNA population by biotinylation and purification with streptavidin-coated magnetic beads, followed by sequencing. The the main set of steps in the *INSPEcT* workflow is as follows:

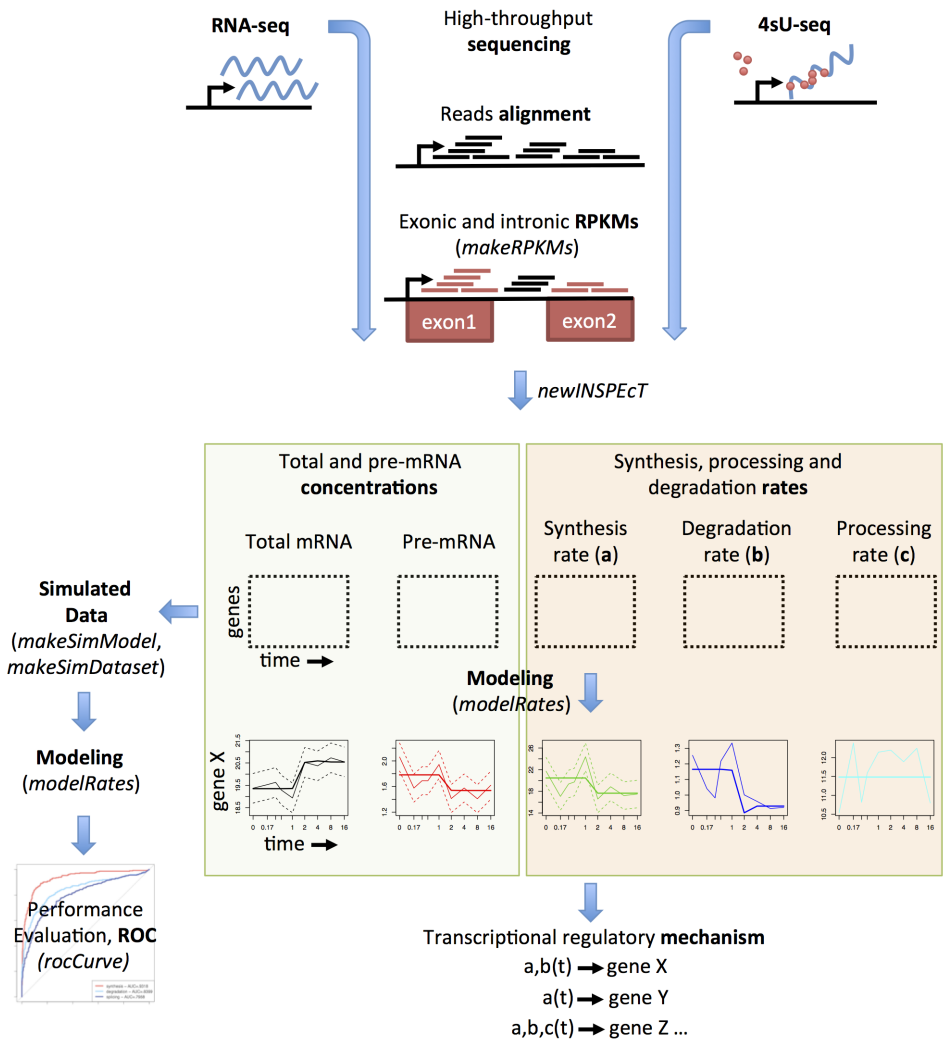

Figure 1: Representation of the INSPEcT workflow

- Exonic and intronic RPKM for both RNA- and 4sU-seq are determined for each gene. Exonic and intronic RNA-seq RPKM allow to quantify the total mRNA and pre-mRNA, respectively (`makeExonsGtfFromDb`, `makeIntronsGtfFromDb`, `makeRPKMs`).
- Normalized synthesis, processing and degradation rates are obtained by integrating RNA- and 4sU-seq data (`newINSPEcT`).
- Rates, total mRNA and pre-mRNA concentrations are modeled for each gene to assess which of the rates, if any, determined changes in mRNA levels (`modelRates`).
- Simulated data that recapitulate rate distributions, their variation over time and their pair-wise correlations are created and used to evaluate the performance of the method (`makeSimModel`, `makeSimDataset`, `rocCurve`).

Within this vignette, a complete *INSPEcT* analysis is presented (Figure 1). For details regarding *INSPEcT* extended methods description, refer to de Pretis S. et al., Bioinformatics (2015).

2 Quantification of Exon and Intron features

The INSPEcT framework includes function to quantify Exon and Intron features, both in 4sU- and RNA-seq experiments. `makeRPKMs` function builds an annotation for exons and introns, either at the level of transcripts or genes, and count reads falling on the two different annotations in each provided file in BAM or SAM format. This function prioritize the exon annotation, meaning that reads which fall on exon are not counted for introns, in case of overlap. Canonical RPKMs are then evaluated in order to provide expression values. RPKMs, read counts, and the annotation is returned as output.

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
paths_4su <- system.file('extdata', '4sURNA_0h.bam', package="INSPEcT")
paths_total <- system.file('extdata', 'totalRNA_0h.bam', package="INSPEcT")
makeRPKMsOut <- makeRPKMs(txdb, paths_4su, paths_total)
rpkms <- makeRPKMsOut$rpkms
```

In case intronic and exonic RPKMs have been computed using alternative methods, they can be provided for the following analyses.

3 Time-course analysis

3.1 Estimation of rates

INSPEcT is based on a simple model of differential equations that describes the process of synthesis and processing of pre-mRNA and the degradation of mature mRNA. Equations model the synthesis of new pre-mRNAs which then decay into mature mRNAs, which in turn exponentially degrade and are removed from the system. The model is based on two main assumptions that are widely used in the description of mRNA life cycle: pre-mRNAs are not degraded, and translocation of mRNAs from nucleus to cytoplasm occurs immediately after maturation, or at a rate considerably faster than the rate of degradation (Rabani M. et al., Nature Biotechnology, 2011; Sun M. et. al, Genome Research, 2012). The model lacks any spatial assumption, like segregation of mRNAs into cellular compartments that could impact the degradation rate, but this is a consequence of the non-spatial nature of the data:

$$\begin{cases} \dot{P} = a(t) - c(t)\,P \\ \dot{T} = a(t) - b(t)\,(T - P) \end{cases}$$

where T is total RNA, P is pre-mature RNA, $a(t)$ is the synthesis rate, $b(t)$ is the degradation rate and $c(t)$ is the processing rate. After having quantified data from RNA-seq (R) and 4sU-seq (labeled, L) libraries into intronic and exonic RPKMs (`makeRPKMs` function), the `newINSPEcT` method is used to estimate synthesis, processing and degradation rates by solving the above system of differential equations applied at every time point (t) to both the total and labeled fractions. When applied to the labeled RNA fraction, the system can be solved and integrated between $t - t_L$ and t, assuming that no labeled molecules existed before the labeling pulse (t_L). At each time point, *INSPEcT* solves a system of four equations:

$$\begin{cases} \dot{P}_{R_t} = a_t - c_t\, P_{R_t} \\ \dot{T}_{R_t} = a_t - b_t\,(T_{R_t} - P_{R_t}) \\ P_{L_t} = \frac{a_t}{c_t} - (1 - e^{c_t\, t_L}) \\ T_{L_t} = a_t\, t_L \end{cases}$$

with three unknowns a_t, b_t and c_t which are respectively the synthesis, degradation and processing rates at time t. P_{R_t} is equal to the pre-mRNA level (intronic RNA-seq RPKM), T_{R_t} is equal to the total mRNA level (exonic RNA-seq RPKM), and P_{L_t} is equal to the pre-mRNA level as quantified in the labeled fraction (intronic 4sU-seq RPKM). Finally, T_{L_t} is equal to the total mRNA level as quantified in the labeled fraction (exonic 4sU-seq RPKM). $\dot{P}_{R_t}$, $\dot{T}_{R_t}$ are estimated from the interpolation of $P_R(t)$ and $T_R(t)$. The overdetermination of the system is used to calculate a time-point specific scaling factor between RNA- and 4sU-seq RPKMs that can be visualized using the `sfPlot` method.

In order to create an object of class *INSPEcT* and to calculate the first estimates of rates and concentrations for each gene, the specific time points of the time-course, the 4sU labeling time and the RPKMs corresponding to labeled and total fraction of the intronic and exonic regions of each gene have to be provided (here time is in $hours$). All results are stored in an object of class *INSPEcT* and rates can be accessed with the `ratesFirstGuess` method. This sample code calculates rates and concentrations on a sample set of 500 genes:

```
data('rpkms', package='INSPEcT')
tpts <- c(0, 1/6, 1/3, 1/2, 1, 2, 4, 8, 16)
tL <- 1/6
mycerIds <- newINSPEcT(tpts, tL, rpkms$foursu_exons, rpkms$total_exons,
        rpkms$foursu_introns, rpkms$total_introns, BPPARAM=SerialParam())
```

```
## For some genes only synthesis and degradation will be evaluated because they have zero
valued features in more than 2/3 of the time points in their intronic features:  333193;
94067; 230866; 68961; 100042464; 667250; 59288; 100038734; 100113398; 100040591
```

```
## Some genes have only exons RPKMs, on them only synthesis and degradation will be evaluated.
```

```
## Number of genes with introns and exons:  490
```

```
## Calculating scaling factor between total and 4su libraries...
```

```
## Estimating degradation rates...
```

```
## Estimating processing rates...
```

```
## Number of genes with only exons:  10
```

```
## Estimating degradation rates...
```

```
head(ratesFirstGuess(mycerIds, 'synthesis'))
```

```
##       synthesis_0 synthesis_0.17 synthesis_0.33 synthesis_0.5 synthesis_1 synthesis_2
## 77595    7.528854       4.980970       5.876078      5.658038    4.855749    4.002908
## 66943    5.000007       3.492451       4.341861      5.217094    5.399309    3.888288
## 16969   17.386770      13.751558      14.963087     16.621702   16.138508   12.980762
## 76863   44.986626      41.850756      44.267417     45.791007   48.873205   37.520985
## 56372   32.434462      27.747296      29.820630     33.162516   33.290951   26.140684
```

```
## 64540   47.722321      36.832310      43.235743     46.420279   61.259644   47.602034
##       synthesis_4 synthesis_8 synthesis_16
## 77595    4.661779    3.858624     5.182438
## 66943    3.504487    2.677346     3.030138
## 16969   14.367606   11.858020    12.744490
## 76863   41.582881   38.809257    36.507584
## 56372   31.190347   25.787364    26.545396
## 64540   50.681092   45.600072    44.062532
```

In case of a long 4sU labeling time (longer than 10-15 minutes), it could be useful to activate the `degDuringPulse` option, as shown below. This option estimates all the rates of the RNA life-cycle without assuming that no degradation of the newly synthesized transcripts occurs during the pulse. The longer the labeling time is, the weaker this assumption gets. This option, however, involves solving a more complicated system of differential equations and for this reason it is not recommended for short labeling times.

```
mycerIdsDdp <- newINSPEcT(tpts, tL, rpkms$foursu_exons, rpkms$total_exons,
        rpkms$foursu_introns, rpkms$total_introns, BPPARAM=SerialParam(), degDuringPulse=TRUE)
head(ratesFirstGuess(mycerIdsDdp, 'synthesis'))
```

```
##       synthesis_0 synthesis_0.17 synthesis_0.33 synthesis_0.5 synthesis_1 synthesis_2
## 77595    9.411477       5.684722       6.576053     7.268387    5.798410    4.635642
## 66943    5.516929       3.678602       4.391283     5.485265    5.832666    4.140901
## 16969   19.265293      14.723916      15.120875    17.089119   17.956414   14.084861
## 76863   50.378856      44.349792      47.247940    51.622971   54.222735   41.575164
## 56372   35.637688      29.171039      31.029189    35.696777   36.080218   28.961868
## 64540   52.665742      38.638173      44.004380    47.962549   66.994180   52.041814
##       synthesis_4 synthesis_8 synthesis_16
## 77595    5.463453    4.356661     6.502008
## 66943    3.753895    2.787467     3.188385
## 16969   15.285097   12.679985    13.474854
## 76863   45.005148   41.861676    38.939693
## 56372   33.073911   27.377715    27.788418
## 64540   53.538485   47.952355    45.834433
```

It is possible to subset the *INSPEcT* object and focus on a specific set of genes. For the sake of speeding up the downstream analysis, we are going to focus on the first 10 genes of the obtained *INSPEcT* object. We can now display the total and pre-mRNA concentrations together with the synthesis, degradation and processing pre-modeling rates they originated from using the `inHeatmap` method (Figure 2).

```
mycerIds10 <- mycerIds[1:10]
inHeatmap(mycerIds10, clustering=FALSE)
```

3.2 Modeling of rates to determine transcriptional regulatory mechanism

Once a prior estimate is obtained for synthesis, processing and degradation rates over time for each gene, *INSPEcT* tests different models of transcriptional regulation to identify the most likely combination of rates explaining the observed changes in gene expression (`modelRates` method). To this purpose, a parametric function is fit to each rate over time, through minimization of residual sum of squares. Once the parametric functionalization for synthesis, degradation and processing rates are obtained, it is possible to test how those

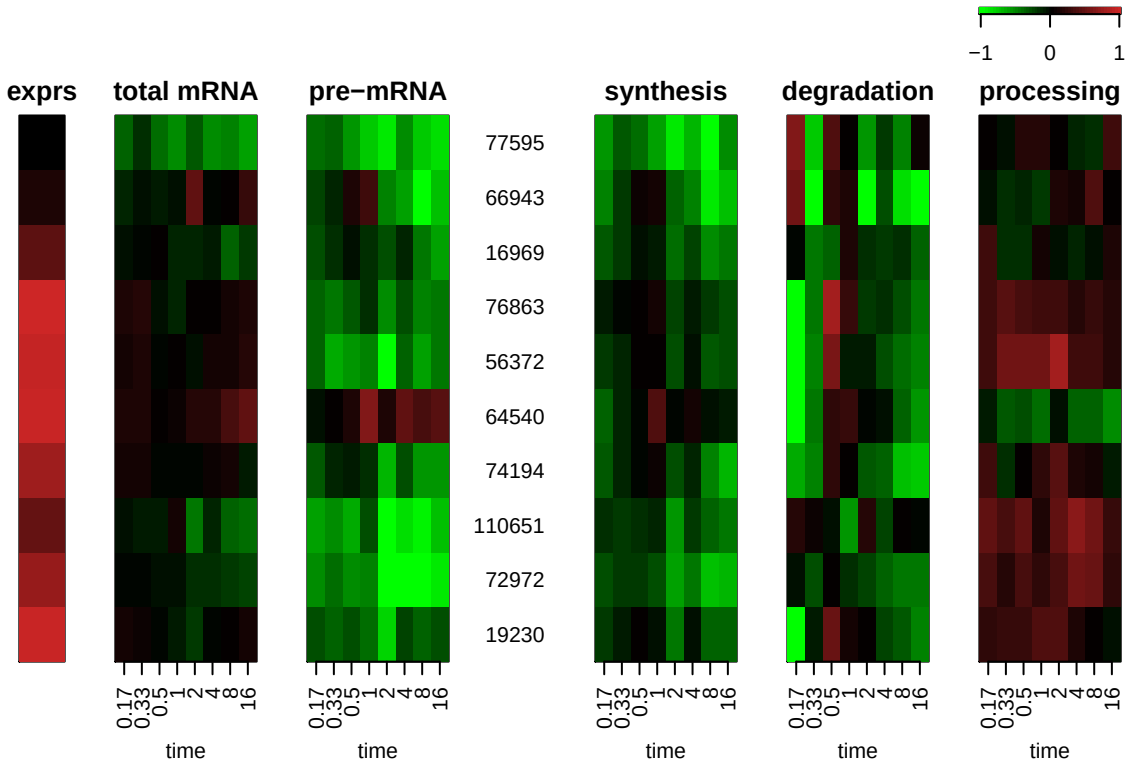

Figure 2: Heatmap of the pre-model rates representing the concentrations of total mRNA and pre-mRNA and the rates of the mRNA life cycle

parametric functions recapitulate the experimental data they originated from after an additional minimization step. To identify the most likely mechanism of transcriptional regulation, *INSPEcT* tests the possibility that each rate is constant during the time course by building models that alternatively set as constant one, two or all the three rates. Due to the fact the the initial parameters for each rate function are initialized randomly, the `seed` argument in `modelRates` can be set to obtain reproducible results.

```
mycerIds10 <- modelRates(mycerIds10, seed=1)
```

Following this modeling procedure, new rates are computed and they can be accessed through the `viewModelRates` method and visualized with the `inHeatmap` method (Figure 3).

```
data('mycerIds10', package='INSPEcT')
head(viewModelRates(mycerIds10, 'synthesis'))
```

```
##       synthesis_0 synthesis_0.17 synthesis_0.33 synthesis_0.5 synthesis_1 synthesis_2
## 77595    7.126555       5.344305       5.343918      5.317712    4.746869    4.744761
## 66943    4.515331       4.515581       4.587750      5.149937    5.248445    3.556650
## 16969   16.194439      15.375145      15.374862     15.374862   15.374834   13.547124
## 76863   42.262296      42.262296      42.262296     42.262296   42.262296   42.262296
## 56372   29.577186      29.577186      29.577186     29.577186   29.577186   29.577186
## 64540   41.863501      41.863973      42.035407     46.599833   50.022497   50.024394
##       synthesis_4 synthesis_8 synthesis_16
## 77595    4.744761    4.744761     4.744761
## 66943    3.105208    3.105143     3.105143
```

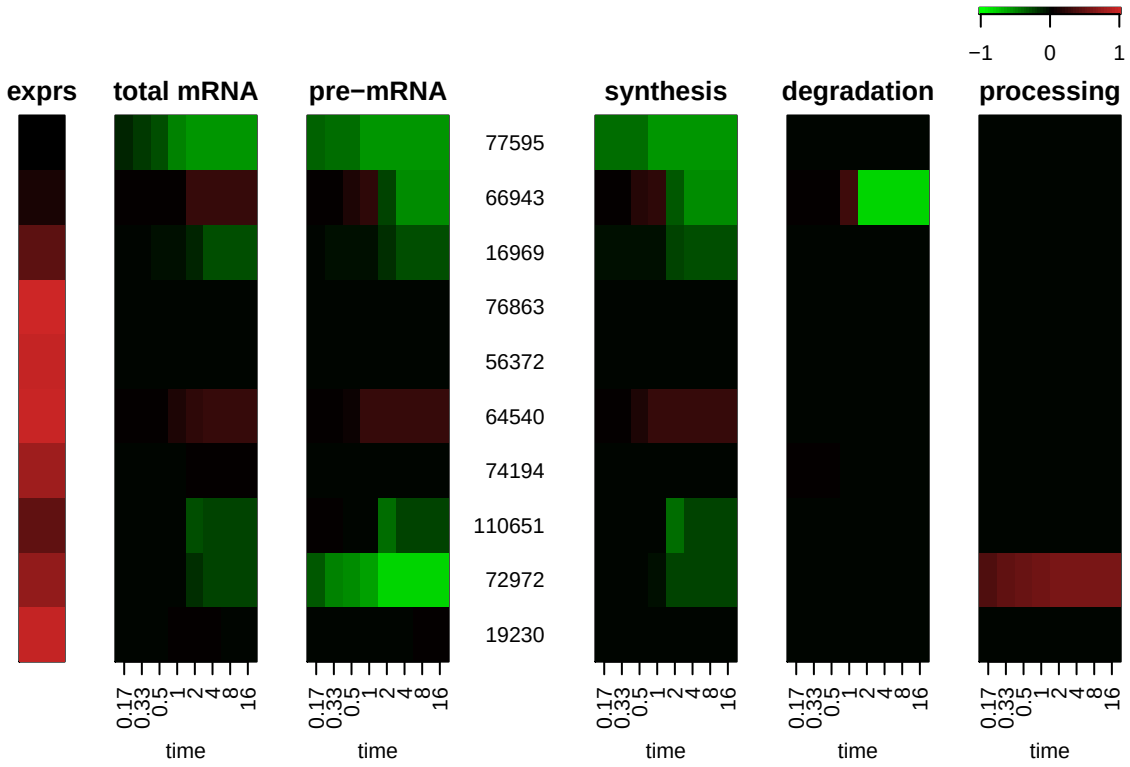

Figure 3: Heatmap of the modeled rates representing the concentrations of total mRNA and pre-mRNA and the rates of the mRNA life cycle

```
## 16969   13.280429   13.280429    13.280429
## 76863   42.262296   42.262296    42.262296
## 56372   29.577186   29.577186    29.577186
## 64540   50.024394   50.024394    50.024394
```

```
inHeatmap(mycerIds10, type='model', clustering=FALSE)
```

The `geneClass` method can be used to recapitulate the transcriptional regulatory mechanism assigned to each modeled gene. In particular, each gene is assigned to a class named after the set of varying rates, if any ("0" denotes a gene whose rates are constant over time, "a" denotes a gene whose synthesis changes over time, "b" denotes a gene whose degradation changes over time, "c" denotes a gene whose processing changes over time).

```
geneClass(mycerIds10)
```

```
##  77595  66943  16969  76863  56372  64540  74194 110651  72972  19230
##    "a"   "ab"    "a"    "0"    "0"    "a"    "b"    "a"   "ac"    "0"
```

The `plotGene` method can be used to investigate profiles of mRNA concentrations and rates for a given gene. Estimated synthesis, degradation and processing rates, pre-mRNA and total mRNA concentrations are displayed with solid thin lines, while their variances are in dashed lines and the modeled rates and concentrations are in thick solid lines. This example shows a gene of class "ab", indicating that its levels are controlled by both synthesis and degradation. In this case, the variation of multiple rates determines opposite trends for total and pre-mRNA concentrations over time: despite a decrease in the synthesis rate over time, the levels of

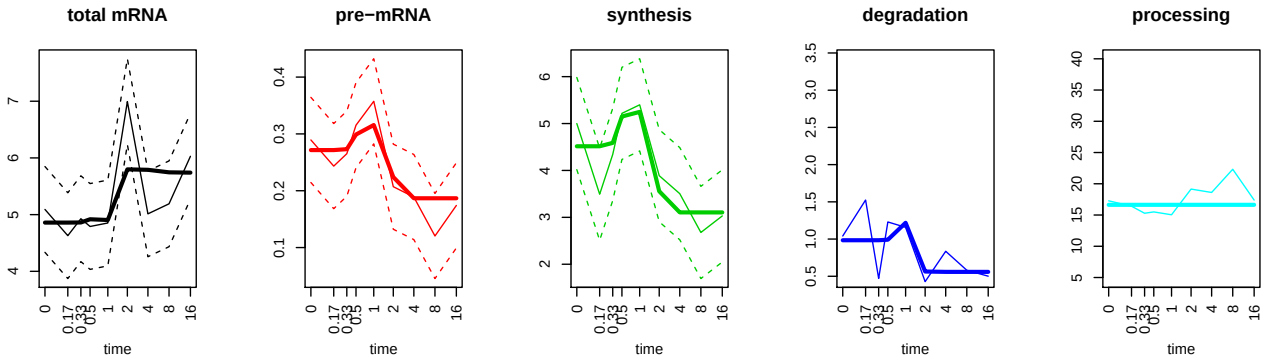

Figure 4: Pre-model and modeled concentrations and rates for a selected gene with standard deviation, where available

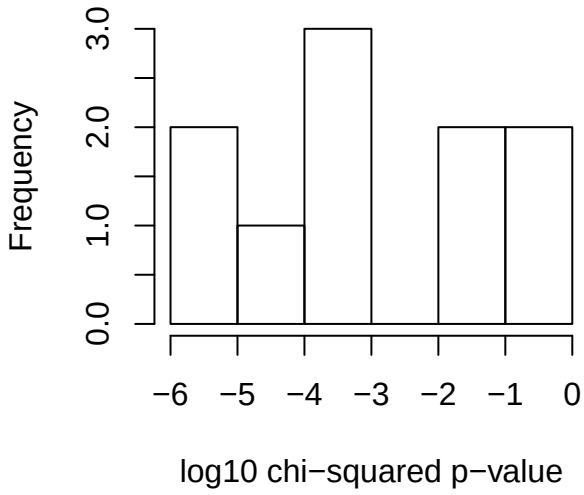

Figure 5: Histogram of the p-values from the goodness of fit test for selected models

total mRNA increase mostly due to a decrease in the degradation rate (Figure 4).

```
plotGene(mycerIds10, 2, fix.yaxis=TRUE)
```

For each model, the chi-squared statistic that measures the goodness of the fit is calculated. In order to evaluate how good the models are able to recapitulate the experimental data, the chi-squared test p-values of the model that better represents the transcriptional scenario for each gene can be visualized as a histogram (Figure 5). Eventually, models with a p-value of the chi-squared test higher than a selected threshold can be discarded.

```
chisq <- chisqmodel(mycerIds10)
hist(log10(chisq), main='', xlab='log10 chi-squared p-value')
```

```
discard <- which(chisq>.05)
featureNames(mycerIds10)[discard]
```

```
## [1] "56372" "74194"
```

```
mycerIds10new <- mycerIds10[-discard]
```

3.3 Evaluation of performance via simulated data

INSPEcT provides functionality to build a synthetic dataset for which the transcriptional scenario is known for each gene. Simulated data can be used to evaluate the performance of *INSPEcT* in classifying each rate as constant or variable over time, and to estimate the number of time points and replicates necessary to achieve a given performance. The method `makeSimModel` takes as arguments an *INSPEcT* object and the number of genes that have to be sampled. The *INSPEcT* object is used to sample absolute values of the rates, fold changes, correlations between absolute values and fold changes and variance of the noise to be added to each feature. Optionally, the user can provide a new set of time points where the synthetic dataset will be sampled, and the ratio between the constant rates, the rates modeled with an impulse model and the rates modeled with a sigmoid function. By default the ratio is .5 constant, .3 impulse, .2 sigmoid.

```
simRates <- makeSimModel(mycerIds, 1000, newTpts=NULL, seed=1)
```

The `makeSimModel` method generates an object of class *INSPEcT_model* which can be used to generate an object of class *INSPEcT* using the `makeSimDataset` method. The `makeSimDataset` method takes as arguments the timepoints at which the dataset has to be simulated and the number of replicates that need to be simulated. The object created by this method can be modeled via `modelRates` as any other object of class *INSPEcT*.

```
simData1rep <- makeSimDataset(simRates, tpts, 1, seed=1)
simData1rep <- modelRates(simData1rep, seed=1)
newTpts <- c(0, 1/6, 1/3, 1/2, 1, 1.5, 2, 4, 8, 12, 16, 24)
simData3rep <- makeSimDataset(simRates, newTpts, 3, seed=1)
simData3rep <- modelRates(simData3rep, seed=1)
```

It is possible to compare now the performance of the modeling, by comparing the `simRates` object, which contains the ground truth of rates, to the `simData1rep` or `simData3rep` objects, which contain the predictions made by *INSPEcT* on datasets that have been simulated with one replicate of 9 time points or three replicates of 12 time points. Modeled data have been previously computed and stored within the package for computational time reasons and are not evaluated directly within this vignette. The evaluation of the performance is done using a ROC curve analysis and measured with the area under the curve (AUC) (Figure 6).

```
data('simRates', package='INSPEcT')
data('simData1rep', package='INSPEcT')
data('simData3rep', package='INSPEcT')
par(mfrow=c(1,2))
rocCurve(simRates, simData1rep); title("1 replicate - 9 time points", line=3)
rocCurve(simRates, simData3rep); title("3 replicates - 12 time points", line=3)
```

The method `rocThresholds` can be used to assess the sensitivity and specificity that is achieved thanks to the given thresholds for the chi-squared test and for the Brown's test. If thresholds are not provided, default values are used (Figure 7).

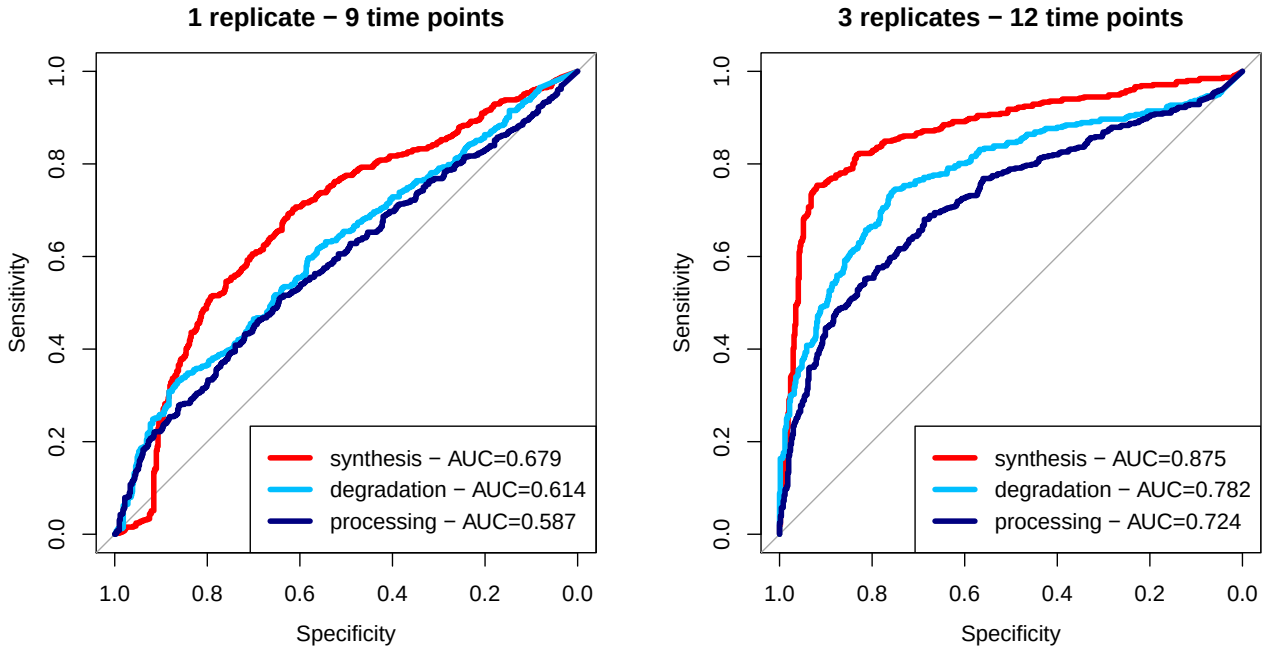

Figure 6: For each rate, INSPEcT classification performance is measured in terms of sensitivity and specificity using a ROC curve analysis (rocCurve method). False negatives (FN) represent cases where the rate is identified as constant while it was simulated as varying. False positives (FP) represent cases where INSPEcT identified a rate as varying while it was simulated as constant. On the contrary, true positives (TP) and negatives (TN) are cases of correct classification of varying and constant rates, respectively. Consequently, sensitivity and specificity are computed using increasing thresholds for the Brown's method used to combine multiple p-values derived from the log-likelihood ratio tests

```
rocThresholds(simRates, simData3rep, bTsh=c(.01,.01,.05), cTsh=.1)
```

3.4 Parameter settings

If desired, different parameters can be set for both the modeling and the testing part. Regarding the modeling part, we might want to exclude testing sigmoid functions (all evaluated smooth function will be impulse model functions, `useSigmoidFun=FALSE` option), increase the number of different initializations that are performed for each gene (`nInit` option), or increase the maximum number of steps in the rates optimization process (`nIter` option). All these choices could increase the performance of the method, but also the needed computational time. Nevertheless, the use of sigmoid functions can reduce over-fitting problems and is highly recommended when the number of data points within the time-course is lower than 7. The impact of these options can be evaluated using a synthetic dataset.

```
modelingParams(mycerIds10)$useSigmoidFun <- FALSE
modelingParams(mycerIds10)$nInit <- 20
modelingParams(mycerIds10)$nIter <- 1000
```

Alternatively we might want to change the thresholds for chi-squared and log-likelihood ratio tests, or define the specific set of models to be compared with log-likelihood ratio test while assessing if a given rate is variable

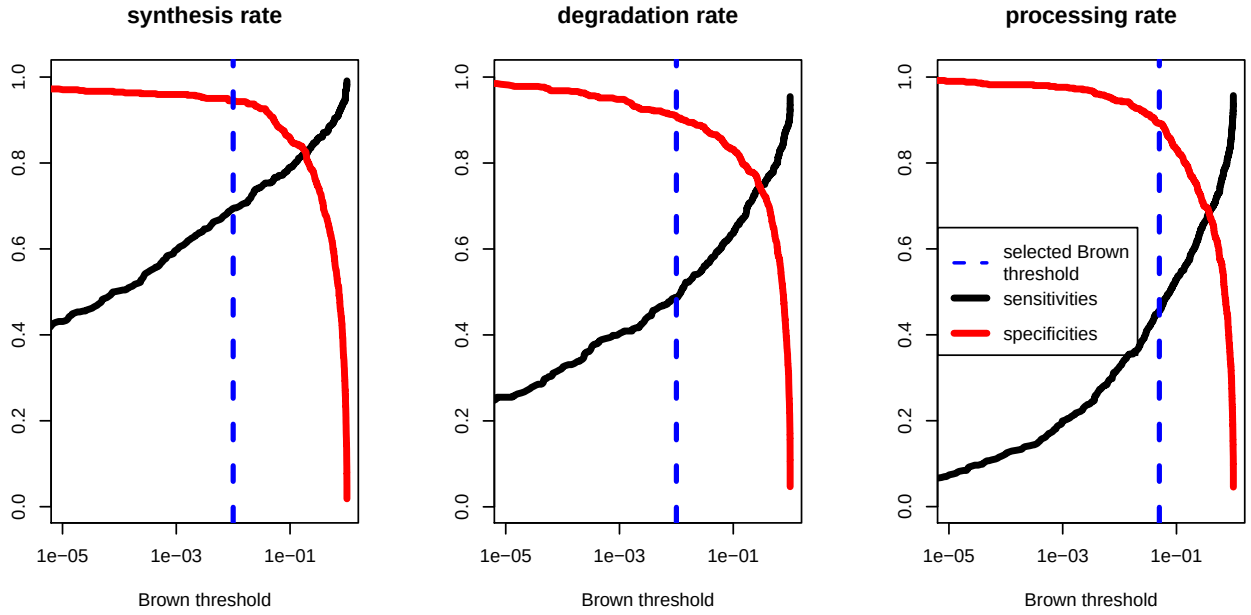

Figure 7: Plot of the sensitivity (black curve) and specificity (red curve) that is achieved after performing the log-likelihood ratio and Brown's method for combining p-values with selected thresholds. Thresholds that can be set for chi-squared test to accept models that will undergo the log-likelihood ratio test and for Brown's p-value to assess variability of rates.

or not. In this example, we are changing the thresholds of both the chi-squared test and the Brown's method for combining p-values. Regarding the processing rates, only the models in which all rates are constant ("0") will be compared to the one in which only processing varies ("c") to assess the variability of the rates using log-likelihood ratio test.

```
thresholds(mycerIds10)$chisquare <- .1
thresholds(mycerIds10)$brown <- c(alpha=.01, beta=.01, gamma=.05)
llrtests(mycerIds10)$processing <- list(c('0','c'))
```

To have a sense of all parameters that can be set, type:

```
## modeling
modelingParams(mycerIds10)
## model selection and testing framework
modelSelection(mycerIds10)
thresholds(mycerIds10)
llrtests(mycerIds10)
```

4 Steady-state analysis

Synthesis, processing and degradation rates are the determinants of the levels of pre-mRNAs and mature mRNAs. At steady-state the ratio between synthesis and processing rates determine the pre-mRNA levels, while the ratio between synthesis and degradation determine the mature mRNA levels. Different conditions

might express different levels of mature mRNA or pre-mRNA for each gene and this difference could arise from the different usage of one, two or all three rates. In order to address the problem, rates from the two different experimental conditions are compared at the gene level by a simple t-test between rate values. Rate variances are estimated using the properties of variance using the approximation that synthesis rate (a), degradation rate (b) and processing rates (c) are computed as:

$$a = \frac{T_L}{t_L \cdot s_f}$$
$$b = \frac{a}{T_T - P_T}$$
$$c = \frac{c}{P_T}$$

where T_L is the 4sU-seq-exonic RPKM of the gene under consideration, T_T is the RNA-seq-exonic RPKM, P_T is the RNA-seq-intronic RPKM, t_L is the labeling time and s_f is a scaling factor that INSPEcT calculated to linearly scale the 4sU library. Considered that we know the variance of T_L, T_T, and P_T, we can also infer the variance of a, b and c. In order to test log scaled means and variances, we transform the variance using the formula:

$$Var[f(X)] \approx (f'(E[x]))^2 \cdot Var(X)$$

In order to exemplify the steady state analysis task, we generate a second set of simulated data with 3 replicates from the object *INSPEcT_model* simRates and compare it with the one which was previously generated for the time course analysis.

```
newTpts <- c(0, 1/6)
simData3rep_2 <- makeSimDataset(simRates, newTpts, 3, seed=2)
```

We have now two datasets with 3 replicates. The comparison between the two datasets is performed by the `compareSteady` method. This method take as input two objects of class INSPEcT and check that each of have been profiled with replicates, at least for the steady state condition (identified as the first temporal condition).

```
diffrates <- compareSteady(simData3rep, simData3rep_2)
```

Results of the comparison can be seen at a glance by typing the name of the variable which contains the results:

```
diffrates
```

```
## Object of class INSPEcT_diffsteady
```

```
## Head of slot synthesis:
```

```
##   condition1   variance1 condition2   variance2 samplesize1 samplesize2 log2mean
## 1   3.854221 0.028188427   4.042793 0.055307735           3           3 3.948507
## 2   3.049255 0.019488760   3.203442 0.010301351           3           3 3.126349
## 3   4.354099 0.023354702   4.223092 0.010076628           3           3 4.288595
## 4   4.213518 0.033074733   4.358450 0.048251461           3           3 4.285984
## 5   4.032123 0.017870494   3.551907 0.043142614           3           3 3.792015
## 6   2.423534 0.005658333   2.272448 0.001584867           3           3 2.347991
##       log2fc       pval      padj
## 1  0.1885720 0.32152260 0.7637116
## 2  0.1541865 0.19670314 0.6602565
```

```
## 3 -0.1310071 0.28240819 0.7361887
## 4  0.1449326 0.42842817 0.8286812
## 5 -0.4802160 0.02811316 0.3896046
## 6 -0.1510857 0.03711851 0.4316105

## ...  and other 994 hidden genes.

##

## Head of slot degradation:

##   condition1  variance1 condition2    variance2 samplesize1 samplesize2   log2mean
## 1  1.2351141 0.02221613   1.185528 0.0356923424           3           3  0.2750872
## 2  2.0374740 0.19135120   1.788392 0.0655318835           3           3  0.9327224
## 3  1.7411531 0.06523060   1.528905 0.0361781861           3           3  0.7062708
## 4  1.1491568 0.01718102   1.024288 0.0221003258           3           3  0.1175988
## 5  5.0943609 1.77347847   3.341619 1.7320857077           3           3  2.0447241
## 6  0.6476028 0.01320938   0.541821 0.0002022625           3           3 -0.7554653
##        log2fc      pval      padj
## 1 -0.05911428 0.7404902 0.9421642
## 2 -0.18811845 0.4308459 0.8395259
## 3 -0.18754450 0.3067238 0.7729986
## 4 -0.16595378 0.3411773 0.8054793
## 5 -0.60835412 0.1972160 0.6911809
## 6 -0.25729285 0.1602124 0.6766952

## ...  and other 994 hidden genes.

##

## Head of slot processing:

##   condition1  variance1 condition2 variance2 samplesize1 samplesize2 log2mean     log2fc
## 1  15.203271 10.3922961  17.644199 70.427082           3           3 4.033716  0.2148122
## 2  43.991826  7.5759882  47.567206  5.606298           3           3 5.515529  0.1127318
## 3   9.036237  0.7149832   9.133954  1.792719           3           3 3.183481  0.0155174
## 4  12.453646  8.9077566  17.403105  6.436234           3           3 3.879885  0.4827765
## 5   9.412245  5.3344776   7.625994 10.621764           3           3 3.082732 -0.3036136
## 6  29.621827  8.3876154  26.296310 60.171196           3           3 4.802689 -0.1718002
##        pval      padj
## 1 0.6464040 0.9846846
## 2 0.1657527 0.8164551
## 3 0.9198490 0.9947881
## 4 0.1076860 0.7406042
## 5 0.5005656 0.9541432
## 6 0.5431765 0.9679932

## ...  and other 994 hidden genes.

##
```

The complete datasets regarding synthesis, processing and degradation can be accessed via the accessor methods:

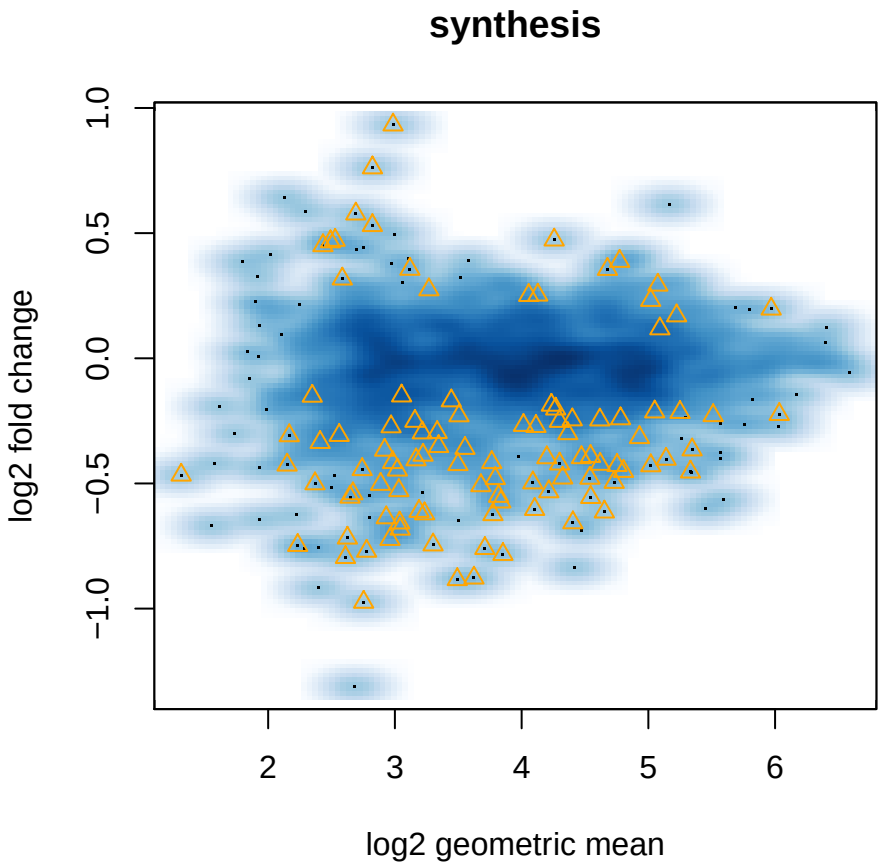

Figure 8: Example of the plotMA generated image. Orange triangles correspond to genes whose rates are differentially used between the two conditions, blue cloud correnspond to the whole ditribution of rates.

```
head(synthesis(diffrates))
head(processing(diffrates))
head(degradation(diffrates))
```

A method for plotting results for each rate has been implemented ((Figure 8)):

```
plotMA(diffrates, rate='synthesis', alpha=.5)
```

Enjoy!

5 About this document

```
sessionInfo()
```

```
## R version 3.2.4 Revised (2016-03-16 r70336)
## Platform: x86_64-pc-linux-gnu (64-bit)
```

```
## Running under: Ubuntu 14.04.4 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets  methods
## [9] base
##
## other attached packages:
##  [1] TxDb.Mmusculus.UCSC.mm9.knownGene_3.2.2 GenomicFeatures_1.22.13
##  [3] AnnotationDbi_1.32.3                    GenomicRanges_1.22.4
##  [5] GenomeInfoDb_1.6.3                      IRanges_2.4.8
##  [7] S4Vectors_0.8.11                        INSPEcT_1.0.2
##  [9] BiocParallel_1.4.3                      Biobase_2.30.0
## [11] BiocGenerics_0.16.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.4                compiler_3.2.4             formatR_1.3
##  [4] futile.logger_1.4.1        plyr_1.8.3                 highr_0.5.1
##  [7] XVector_0.10.0             futile.options_1.0.0       bitops_1.0-6
## [10] tools_3.2.4                zlibbioc_1.16.0            biomaRt_2.26.1
## [13] evaluate_0.8.3             RSQLite_1.0.0              preprocessCore_1.32.0
## [16] rootSolve_1.6.6            DBI_0.3.1                  rtracklayer_1.30.4
## [19] stringr_1.0.0              knitr_1.12.3               pROC_1.8
## [22] Biostrings_2.38.4          deSolve_1.13               XML_3.98-1.4
## [25] lambda.r_1.1.7             magrittr_1.5               Rsamtools_1.22.0
## [28] GenomicAlignments_1.6.3    SummarizedExperiment_1.0.2 BiocStyle_1.8.0
## [31] KernSmooth_2.23-15         stringi_1.0-1              RCurl_1.95-4.8
```