INSPEcT - INference of Synthesis, Processing and dEgradation rates from Transcriptomic data

de Pretis S. - Furlan M. - Pelizzola M.

April 4, 2019

Contents

1 Introduction

INSPEcT is an R/Bioconductor compliant solution for the study of RNA transcriptional dynamics from RNA-seq data. It is based on a system two of ordinary differential equations (ODEs) that describes the synthesis (k_1) and processing (k_2) of premature RNA (P) and the degradation (k_3) of mature RNA ($M = T - P$):

$$\begin{cases} \dot{P} = k_1 - k_2 \cdot P \\ \dot{T} = k_1 - k_3 \cdot (T - P) \end{cases} \quad (1)$$

The total RNA (T) is measured from the exonic quantification of transcripts in RNA-seq experiment, while premature RNA (P) is measured by the intronic quantification. The model is based on two commonly accepted assumptions: premature RNAs are not degraded, but only converted to mature RNA, and nuclear export occurs at a rate considerably faster than other rates and can be disregarded.
The package supports the analysis of several experimental designs, such as steady-state conditions or time-course data, and the presence or absence of the nascent RNA library. According to the experimental design, the software returns different outputs. In particular, when the nascent RNA is present, the rates of synthesis, processing and degradation can be calculated both in time-course and in steady-state state conditions and tested for differential regulation. Without the information from the nascent RNA, the rates of synthesis, processing and degradation, as well as their significant variation, can be calculated only in time-course, while only a variation in the ratio between processing and degradation (post-transcriptional ratio) can assessed between steady states. A schema of the possible configurations and related outputs is reported in Table ??.

Table 1: Table reporting the experimental designs suitable for INSPEcT

	Total and nascent RNA		
	Synthesis	*Processing*	*Degradation*
Steady state	Yes	Yes	Yes
Time-course	Yes	Yes	Yes

	Total RNA		
	Synthesis	*Processing*	*Degradation*
Steady state	No	Ratio $\frac{Processing}{Degradation}$	
Time-course	Yes	Yes	Yes

2 Quantification of Exon and Intron features

The *INSPEcT* analysis starts with the quantification of exonic and intronic quantifications and the estimation of their variance from the replicates in the different conditions of the analysis. *INSPEcT* can estimate transcripts abundances and asso-

ciated variances starting from different entry points according to the source of data available: SAM (or BAM) files, read counts or transcripts abundances. The user can choose to estimate the variance by means of two different softwares: *DESeq2* or *plgem*. By default, *DESeq2* is used when starting from BAM or read counts, while only *plgem* can be used when starting from transcripts abundances.

```
library(INSPEcT)
```

2.1 From BAM or SAM files

In case the user has access to the raw aligned data, *INSPEcT* can quantify transcripts abundaces and variances directly from SAM or BAM files. Transcrips annotation should be provided with an annotation object of class `TxDb`. *INSPEcT* retrieves the genomic coordinates of the exons and defines introns as inter-exonic regions. By default, *INSPEcT* collapses the exons of the transcripts that belong to the same gene (argument `by` set equal to `'gene'`), but can work also at the transcript level (argument `by` set equal to `'tx'`). When working at the transcript level, *INSPEcT* cannot discriminate betweeen transcript, therefore we suggest to assign reads to all the overlapping features by setting the argument `allowMultiOverlap` to `TRUE`. *INSPEcT* manage strandness of the reads with the argument `strandSpecific`, which can be set to `0`, in case of non-stranded reads, `1` for stranded and `2` for reverse-stranded experiments. *INSPEcT* prioritizes the exon annotation, meaning that only the reads that do not overlap with any of the annotated exon are eventually accounted for introns. Here we report an example where the data from 4 sample BAM files (2 time points, 2 replicates each, as specified with the argument `experimentalDesign`) is quantified in a non-stranded specific way, at the gene level, leaving un-assigned the reads mapping to more than one feature.

```
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene

bamfiles_paths <- system.file('extdata/',
  c('bamRep1.bam','bamRep2.bam','bamRep3.bam','bamRep4.bam'),
  package='INSPEcT')

exprFromBAM <- quantifyExpressionsFromBAMs(txdb=txdb
           , BAMfiles=bamfiles_paths
           , by = 'gene'
           , allowMultiOverlap = FALSE
           , strandSpecific = 0
           , experimentalDesign=c(0,0,1,1))
```

The function returns a list containing exonic/intronic expressions and variances, as well as exonic/intronic feature widths extracted from the annotation, exonic/intronic counts and counts statitics.

```
names(exprFromBAM)
```

```
## [1] "exonsExpressions"   "intronsExpressions" "exonsVariance"
## [4] "intronsVariance"    "exonsWidths"        "intronsWidths"
## [7] "exonsCounts"        "intronsCounts"      "countsStats"
```

```
exprFromBAM$countsStats
```

```
##                       0_rep1 0_rep2 1_rep1 1_rep2
## Unassigned_Ambiguity       5      0      0      0
## Assigned_Exons          2515   2763   2729   2672
## Assigned_Introns         862    609    654    706
## Unassigned_NoFeatures      0      0      0      0
```

2.2 From read counts

In case the matices of intronic and exonic reads have been already calulated, *INSPEcT* calulates the mean expression and the associated variance using, by default, the software *DESeq2*. The package *INSPEcT* contains the read counts associated to the intronic and exonic features of 500 genes profiled both for the nascent and total RNA in 11 time-points and replicated 3 times each following the induction of Myc in 3T9 cells [ref to the paper]. As a complementary information, the width of the intronic and exonic features for the same set of genes, as well as the sequencing depth of all the samples is provided within the package. The nascent and total quantification evaluated in this section of the vignette will be used later to generate the synthetic dataset for the benchmarking of the method.

```
data('allcounts', package='INSPEcT')
data('featureWidths', package='INSPEcT')
data('libsizes', package='INSPEcT')

nascentCounts<-allcounts$nascent
matureCounts<-allcounts$mature
expDes<-rep(c(0,1/6,1/3,1/2,1,1.5,2,4,8,12,16),3)

nasExp_DESeq2<-quantifyExpressionsFromTrCounts(
        allcounts=nascentCounts
        ,libsize=nascentLS
        ,exonsWidths=exWdths
        ,intronsWidths=intWdths
        ,experimentalDesign=expDes)

matExp_DESeq2<-quantifyExpressionsFromTrCounts(
        allcounts=matureCounts
        ,libsize=totalLS
```

```
              ,exonsWidths=exWdths
              ,intronsWidths=intWdths
              ,experimentalDesign=expDes)
```

2.3 From transcripts abundances

In order to exemplify the quantification of mean transcripts abundances and variances starting from their replicated quantification, we transform the read counts loaded from the package in the section above into RPKMs using the width of intron and exon and the library sizes

```
trAbundancesFromCounts <- function(counts, widths, libsize)
        t(t(counts/widths)/libsize*10^9)
nascentTrAbundance <- list(
  exonsAbundances=trAbundancesFromCounts(
        nascentCounts$exonsCounts, exWdths, nascentLS),
  intronsAbundances=trAbundancesFromCounts(
        nascentCounts$intronsCounts, intWdths, nascentLS))
```

Once we have obtained transcripts abundances for introns and exons in the different conditions and replicates of the experimental design, we can calulate the mean expression and variance (using *plgem*) by:

```
nasExp_plgem<-quantifyExpressionsFromTrAbundance(
        trAbundaces = nascentTrAbundance
        , experimentalDesign = expDes)
```

3 Analysis of RNA dynamics in time-course experiments

3.1 Analysis of Total and Nascent RNA

In case of the joint analysis of total and nascent RNA data, for each transcript with at least one intron, the exonic and intronic quantifications are available in the Total and in the Nascent RNA libraries. The system of equations 1 describes the exonic and intronic quantifications of the Total library, but when integrated between zero and the time of labeling used to generate the nascent RNA transcripts (t_L) it describes the exonic and intronic quantifications of the Nascent library. For matter of simplicity

and to avoid unrealistic high rates of synthesis, processing and degradation, *INSPEcT* assumes by default that the nascent transcripts are not degradaed during the labeling time. The set of equations to describe Total and Nascent RNA is:

$$\begin{cases} \dot{P}_R = k_1 - k_2 \cdot P_R \\ \dot{T}_R = k_1 - k_3 \cdot (T_R - P_R) \\ s_F \cdot P_L = \frac{k_1}{k_2} - (1 - e^{k_2 \cdot t_L}) \\ s_F \cdot T_L = k_1 \cdot t_L \end{cases} \quad \text{2}$$

where P_R and T_R are the premature and total RNA levels, respectively, estimated from the total RNA library, P_L and T_L are premature and total RNA levels, respectively, estimated from the nascent RNA library. This system describes a single condition or time-point. In case of steady-state analysis, $\dot{P}_R$ and $\dot{T}_R$ are equal to zero, while in case of the time-course analysis they are estimated from the interpolation of $P_R(t)$ and $T_R(t)$ during the entire time-course via cubic splines. *INSPEcT* exploit the fact that the system is overdetermined (three unknowns - k_1, k_2 adn k_3, and four equations) to calculate a scaling factor between the Total and Nascent RNA quantifications, s_F, that multiplies P_L and T_L. To avoid overfitting, a single scaling-factor representative of the entire population of transcripts is used, which represents the normalization factor between the Nascent and Total libraries (ref to INSPEcT paper).

3.1.1 Estimation of rates from intronic and exonic quantifications

Once obtained the intronic and exonic quantifications from the Total and Nascent library, the rates of synthesis, processing and degradation are calculated following the procedure described in the section above during the initialization of the *INSPEcT* object with the function `newINSPEcT`:

```
tpts<-c(0,1/6,1/3,1/2,1,1.5,2,4,8,12,16)
tL<-1/6
nascentInspObj<-newINSPEcT(tpts=tpts
                           ,labeling_time=tL
                           ,nascentExpressions=nasExp_DESeq2
                           ,matureExpressions=matExp_DESeq2)
```

Once created the *INSPEcT* object, the method `ratesFirstGuess` returns the expressions of premature and total RNA for the analyzed genes, as well as the rates of synthesis, processing and degradation. Additionally, the method `ratesFirstGuessVar` returns the variances of premature and total RNA, and of the synthesis rates.

```
round(ratesFirstGuess(nascentInspObj,'total')[1:5,1:3],3)
```

```
##           total_0 total_0.166666667 total_0.333333333
```

```
## 100503670 605.805           609.366           601.865
## 101206      5.107             5.356             5.196
## 101489     15.629            15.767            15.606
## 102436      1.543             1.107             1.305
## 104458     86.966            93.632            88.837
```

```
  round(ratesFirstGuessVar(nascentInspObj,'total')[1:5,1:3],3)
```

```
##           total_0 total_0.166666667 total_0.333333333
## 100503670 775.569           784.570           764.586
## 101206      0.382             0.417             0.392
## 101489      1.192             1.208             1.179
## 102436      0.281             0.145             0.202
## 104458     24.917            28.736            25.866
```

```
  round(ratesFirstGuess(nascentInspObj,'synthesis')[1:5,1:3],3)
```

```
##           synthesis_0 synthesis_0.166666667 synthesis_0.333333333
## 100503670     231.119               211.859               210.099
## 101206          4.875                 4.412                 3.964
## 101489          7.749                 7.412                 7.421
## 102436         32.087                28.312                28.645
## 104458         47.333                46.424                50.867
```

The *INSPEcT* object can be subsetted to focus on a specific set of genes. For the sake of speeding up the downstream analysis, we will focus on the first 10 genes of the INSPEcT object. The complete pattern of total and pre-mRNA concentrations variation together with the synthesis, degradation and processing rates can be visualized using the `inHeatmap` method (Figure 1).

```
nascentInspObj10<-nascentInspObj[1:10]
inHeatmap(nascentInspObj10, clustering=FALSE)
```

In case of a long t_L (more than 30 minutes), it can be useful to set the argument `degDuringPulse=TRUE` to estimate the rates of the RNA life-cycle without assuming that nascent transcripts are not degradaed during the labeling time. The longer the labelling time is the weaker this assumption gets, however, taking into account nascent RNA degradation involves the solution of a more complicated system of equations and can originate unrealistic high rates of synthesis, processing and degradation.

```
  nascentInspObjDDP<-newINSPEcT(tpts=tpts
                           ,labeling_time=tL
                           ,nascentExpressions=nasExp_DESeq2
                           ,matureExpressions=matExp_DESeq2
                           ,degDuringPulse=TRUE)
```

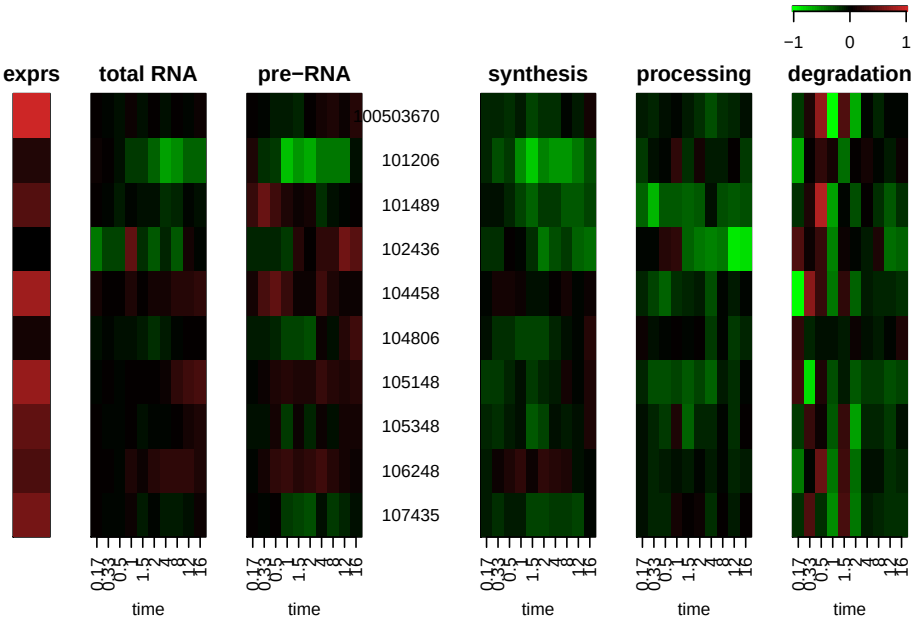


Figure 1: inHeatmap of the ratesFirstGuess
Heatmap representing the concentrations of total RNA, of premature RNA and the first guess of the rates of the RNA life cycle

For short labeling times, as 10 minutes can be considered, the absence of degradation during the pulse is a good assumption and similar rates are estimated in both cases. In Figure 2 we compared the degradation rates estimated with or without the assumption that degradation of mature RNA occurs during the labeling time.

```
k3 <- ratesFirstGuess(nascentInspObj, 'degradation')
k3ddp <- ratesFirstGuess(nascentInspObjDDP, 'degradation')
plot(rowMeans(k3), rowMeans(k3ddp), log='xy',
     xlim=c(.2,10), ylim=c(.2,10),
     xlab='no degradation during pulse',
     ylab='degradation during pulse',
     main='first guess degradation rates')
abline(0,1,col='red')
```

3.1.2 Selection of the regulative scenario

The rates evaluated so far are highly exposed to the experimental noise, in particular processing and degradation rates which sum up the noise coming from different sources of experimental data (such as the signal from the Total and the nascent libraries). For this reason, it could be challenging to distinguish real variations of a rate from noise. Once priors have been computed, the method `modelRates` add a second step of modeling to identify the rates that are significantly regulated over the time course. This method tests wether the variation of a rate over time (synthesis,

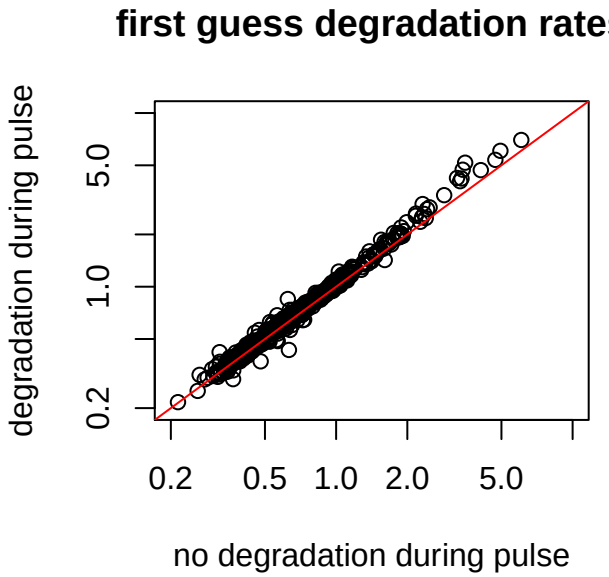


Figure 2: Dotplot of synthesis rates calculated with or without the assumption of no degradation during pulse
First guess of the synthesis rates calcualted with or without degDuringPulse option. In red is represented the line of equation y = x.

processing, degradation, or a combination of them) is required to obtain a good fit of the experimental data. To this purpose, 8 different models are tested for each gene, representing all the combinations of constant or variable synthesis, processing and degradation rates, eventually choosing the best trade off between goodness of fit and simplicity of the model. During this step, constant functions, sigmoid functions or impulse models are used to represent the rates over time. This evaluation can be computationally intense and some parameters are initialized randomly, therefore the argument `seed` guarantees the reproducibility of the results.

```
nascentInspObj10<-modelRates(nascentInspObj10, seed=1)
```

The rates that are computed through this second modeling procedure can be accessed through the method `viewModelRates` and visualized with the `inHeatmap` method, setting the argument `type='model'` (Figure 3).

```
round(viewModelRates(nascentInspObj10, 'synthesis')[1:5,1:3],3)
```

```
##           synthesis_0 synthesis_0.17 synthesis_0.33
## 100503670     214.613        214.613        214.613
## 101206          4.887          4.507          4.136
## 101489          7.286          7.286          7.277
## 102436         30.151         30.151         30.151
## 104458         45.458         45.464         50.729
```

```
inHeatmap(nascentInspObj10, type='model', clustering=FALSE)
```

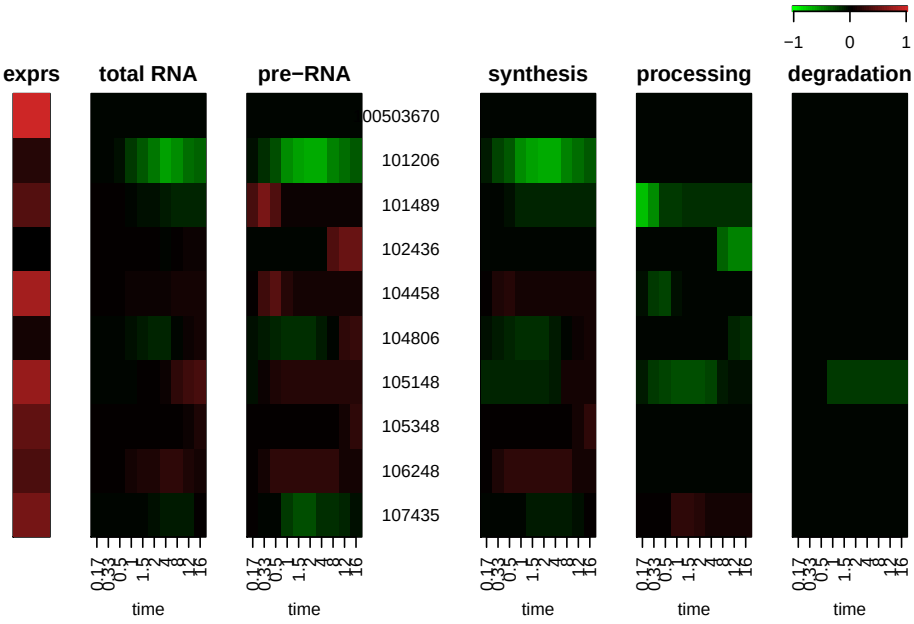


Figure 3: inHeatmap of the rates after the second step of modeling
Heatmap representing the concentrations of total RNA, of premature RNA and the rates of the RNA life cycle after the second step of modeling.

The metohd `geneClass` returns 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: "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(nascentInspObj10)
```

```
## 100503670    101206    101489    102436    104458    104806    105148
##       "0"       "a"      "ac"       "c"      "ac"      "ac"     "abc"
##    105348    106248    107435
##       "a"       "a"      "ac"
```

Additionally, the metohd `plotGene` can be used to fully investigate the profiles of RNA concentrations and rates of a single gene. Estimated synthesis, degradation and processing rates, premature RNA and total RNA concentrations are displayed with solid thin lines, while their standard deviations are in dashed lines and the modelled rates and concentrations are in thick solid lines. This example shows a gene of class "ab", indicating that its expression levels are controlled by synthesis and degradation rates (Figure 4).

```
plotGene(nascentInspObj10, ix="101206")
```

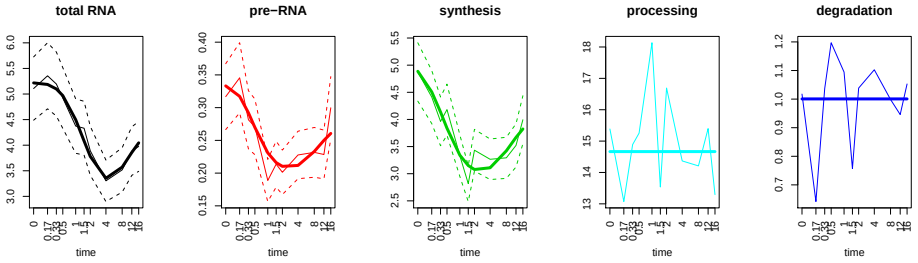


Figure 4: Plot of a single gene resolved using nascent and total RNA
Plot of concentrations and rates over time for the given gene 101206. Total and nascent RNA were used to resolve the whole set of rates and concentrations.

The quality of each model is evaluated through log likelihood and chi-square, to take into account the different complexities of the regulatory scenarios we compute also the chi-square p value; it can visualized as a histogram (Figure 5) and eventually used to discard badly fitted genes.

```
chisq<-chisqmodel(nascentInspObj10)
hist(log10(chisq), main='', xlab='log10 chi-squared p-value')
```

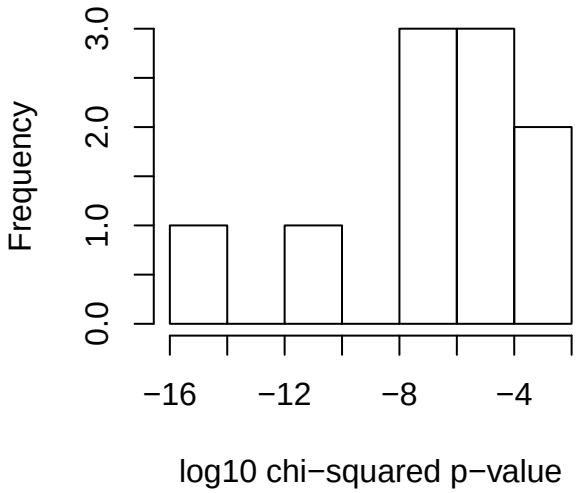


Figure 5: Goodness of the fit
Histogram of the p-values from the goodness of fit test for modeled genes.

```
discard<-which(chisq>1e-4)
featureNames(nascentInspObj10)[discard]

## [1] "100503670" "107435"

nascentInspObj_reduced<-nascentInspObj10[-discard]
```

The model selection is a fundamental and subtle stage of the analysis, it influences all the results shown above and should be carefully triggered according to the specific dataset under analysis. See the **Parameters setting** and **Evaluation of time-course model on simulated data** sections for further details.

3.2 Analysis of Total RNA without Nascent

The absence of the nascent RNA component makes the identification of a unique solution of the system more difficult. Nonetheless, the joint analysis of premature and mature RNA time-course RNA-seq analysis is an important source of information in regards of RNA dynamics. The main idea behind this relies in the fact that genes that are modulated transcriptionally (i.e. by the synthesis rate) respond in their RNA levels following a precise pattern, that is premature RNA and mature RNA are modulated to the same extent (same fold change compared to unperturbed condition) but with different timing. In fact, the mature RNA follow the modulation of the premature RNA by a delay that is indicative to the mature RNA stability (i.e. degradation rate). Deviations from this behavior are signs of post-transcriptional regulation. We developed a computational approach that explots these concepts and is able to quantify RNA dynamics based on time-course profiling of total RNA-seq data with good approximation. Without nascent RNA, the sole information of total (T, exonic quantification) and premature (P, intronic quantification) from the Total library is available and used to model the RNA dynamics. In particular, we model premature RNA (P) and mature RNA ($M = T - P$), using the equation:

$$\dot{M} = k_2\, P(t) - k_3\, M(t) \quad \boxed{3}$$

We start solving the equations assuming constant processing and degradation rates over the time-course and interpolating P with a linear piecewise in order to find the $k2$ and $k3$ that minimize the error over M. While $k1(t)$ is obtained as follows:

$$k_1(t) = \dot{P(t)} + k_2\, P(t) \quad \boxed{4}$$

After that, $k_1(t)$ is used in combination with $P(t)$ and $T(t)$ to obatain $k_2(t)$ and $k_3(t)$ using the same procedure implemented for the joint analysis of nascent and total RNA.

3.2.1 Estimation of rates from intronic and exonic quantifications

The procedure described above is implemented in the method `newINSPEcT`, which creates the *INSPEcT* object for the analysis and gives a first estimation of the synthesis, processing and degradation rates along the time-course, for those genes with enough signal for both introns and exons. In comparison to the mature and nascent configuration, from the practical point of view, nothing changes except for the absence of t_L and new synthesized expression data.

```
tpts<-c(0,1/6,1/3,1/2,1,1.5,2,4,8,12,16)

matureInspObj<-newINSPEcT(tpts=tpts
                          ,labeling_time=NULL
                          ,nascentExpressions=NULL
                          ,matureExpressions=matExp_DESeq2)
```

All the methods that apply to the `INSPEcT` object created with the nascent RNA apply also to the object created with the sole total RNA. For example, we can compare the time-averaged rates of synthesis estimated in this first step of modeling with or without the nascent RNA, by extracting themwith the `ratesFirstGuess` (Figure 6).

```
cg <- intersect(featureNames(nascentInspObj),
                featureNames(matureInspObj))
k1Nascent <- ratesFirstGuess(nascentInspObj[cg,], 'synthesis')
k1Mature <- ratesFirstGuess(matureInspObj[cg,], 'synthesis')
smoothScatter(log10(rowMeans(k1Nascent)),
              log10(rowMeans(k1Mature)),
              main='synthesis rates',
              xlab='nascent + mature RNA',
              ylab='mature RNA')
abline(0,1,col='red')
```

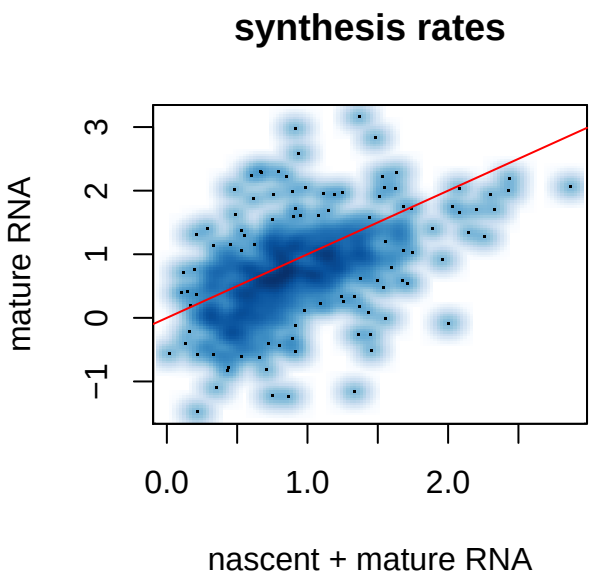


Figure 6: Dotplot of synthesis rates calculated with or without nascent RNA
First guess of the synthesis rates calcualted on the same set of genes with or without the nascent RNA information. In red is represented the line of equation $y = x$.

3.2.2 Selection of the regulative scenario

Similarly to the procedure with the nascent RNA, inspect provide an additional step of modeling, where the software assigns to the rates of the RNA cycle a precise functional form to describe their behavior over time (see the corresponding section in "Analysis of Total and Nascent RNA"). In this case, RNA dynamics can be modeled either an impulse functions or constant functions. During this procedure 8 independent models for each gene are tested (all the combinations of contant/impulsive functions for each of the three rates), and the best model is chosen based on the trade-off between the goodness of fit and the simplicity. This modeling procedure refines the estimation of the rates and gives a statistical confidence about the variation of each rate during the time-course.

```
matureInspObj10 <- matureInspObj[1:10, ]
matureInspObj10 <- modelRates(matureInspObj10, seed=1)
```

In figure 7, the same gene represented in figure 4 is plotted. Also in this case, estimated synthesis, degradation and processing rates, premature RNA and total RNA concentrations are displayed with solid thin lines, while their standard deviations are in dashed lines and the modelled rates and concentrations are in thick solid lines.

```
plotGene(matureInspObj10, ix="101206")
```

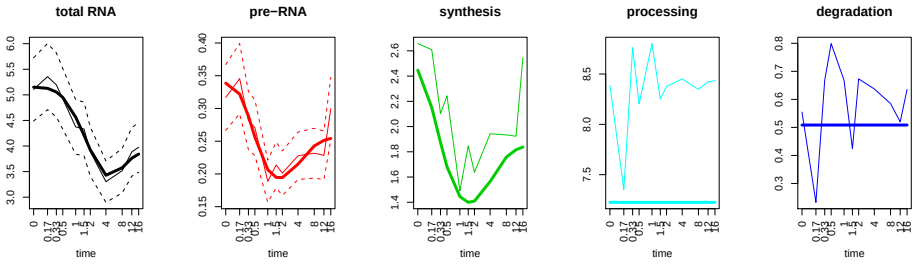


Figure 7: Plot of a single gene resolved using only total RNA
Plot of concentrations and rates over time for the given gene 101206. Only total RNA was used to resolve the whole set of rates and concentrations.

3.3 Performance evaluation with simulated data

INSPEcT provides functionalities 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, to estimate the number of time points and replicates necessary to achieve a given performance and to configure the model selection parameters.
The first step consists in sampling from an *INSPEcT* object: absolute values of the rates, fold changes, correlations between these two elements and associated variances. Once these parameters distributions have been estimated, they are used to simulate a given number of genes ($nGenes = ...$). Optionally, the user can provide

the probabilities for a rate to be modelled as a constant, sigmoid or impulse function; by default they are respectively 0.5, 0.2 and 0.3.

```
simRates<-makeSimModel(nascentInspObj, 1000, seed=1)
```

`makeSimModel` produces an object of class *INSPEcT_model*, to be analysed it must be transformed in an *INSPEcT* object using the `makeSimDataset` method. It takes as arguments the number of replicates required and the time points at which the data should be virtually collected. Regarding the latter point, the new time-course must be designed as a sampling of the original experimental time window (same initial and final conditions). The object created by this method can be modelled via `modelRates` as any other object of class *INSPEcT*, and in this case the results could be compared with the ground truth and the perfromace of the procedure with given number of replicates and time points can be estimated. In order to avoid long computation time, the next two chunks of code have been previously evaluated and will be not computed within this vignette.

```
newTpts <- c(0.00, 0.13, 0.35, 0.69, 1.26, 2.16, 3.63,
             5.99, 9.82, 16.00)
nascentSim2replicates<-makeSimDataset(object=simRates
                                      ,tpts=newTpts
                                      ,nRep=2
                                      ,NoNascent=FALSE
                                      ,seed=1)
nascentSim2replicates<-modelRates(nascentSim2replicates[1:100]
                                  ,seed=1)

newTpts <- c(0.00, 0.10, 0.26, 0.49, 0.82, 1.32, 2.06,
             3.16, 4.78, 7.18, 10.73, 16.00)
nascentSim3replicates<-makeSimDataset(object=simRates
                                      ,tpts=newTpts
                                      ,nRep=3
                                      ,NoNascent=FALSE
                                      ,seed=1)
nascentSim3replicates<-modelRates(nascentSim3replicates[1:100]
                                  ,seed=1)
```

Starting from the very same `simRates`, it is also possible to generate *INSPEcT* objects without information about the nascent RNA (argument `NoNascent` set to TRUE). However, it is not allowed to produce artificial gene sets starting from *INSPEcT* object without nascent RNA because the experimental estimation of the synthesis rate is mandatory to guarantee the good quality of the simulated data.

```
newTpts <- c(0.00, 0.13, 0.35, 0.69, 1.26, 2.16, 3.63,
             5.99, 9.82, 16.00)
```

```
matureSim2replicates<-makeSimDataset(object=simRates
                                     ,tpts=newTpts
                                     ,nRep=2
                                     ,NoNascent=TRUE
                                     ,seed=1)
modelSelection(matureSim2replicates)$thresholds$chisquare <- 1
matureSim2replicates<-modelRates(matureSim2replicates[1:100]
                                 ,seed=1)

newTpts <- c(0.00, 0.10, 0.26, 0.49, 0.82, 1.32, 2.06,
             3.16, 4.78, 7.18, 10.73, 16.00)
matureSim3replicates<-makeSimDataset(object=simRates
                                     ,tpts=newTpts
                                     ,nRep=3
                                     ,NoNascent=TRUE
                                     ,seed=1)
modelSelection(matureSim3replicates)$thresholds$chisquare <- 1
matureSim3replicates<-modelRates(matureSim3replicates[1:100]
                                 ,seed=1)
```

Once the simulated data have been produced and analysed, it is possible to compare the results obtained by the method `modelRates` and the object `simRates`, which contains the ground truth of rates.
The method `rocCurve`, for example, measures the performace of the constant/variable classification individually for the rates of synthesis, processing and degradation, using the receiver operating characteristic (ROC) curve. (Figure 8).

```
data("nascentSim2replicates","nascentSim3replicates",
  "matureSim2replicates","matureSim3replicates",package='INSPEcT')

par(mfrow=c(2,2))
rocCurve(simRates[1:100],nascentSim2replicates)
title("2rep. 10t.p. Total and nascent RNA", line=3)

rocCurve(simRates[1:100],nascentSim3replicates)
title("3rep. 12t.p. Total and nascent RNA", line=3)

rocCurve(simRates[1:100],matureSim2replicates)
title("2rep. 10t.p. Total RNA", line=3)

rocCurve(simRates[1:100],matureSim3replicates)
title("3rep. 12t.p. Total RNA", line=3)
```

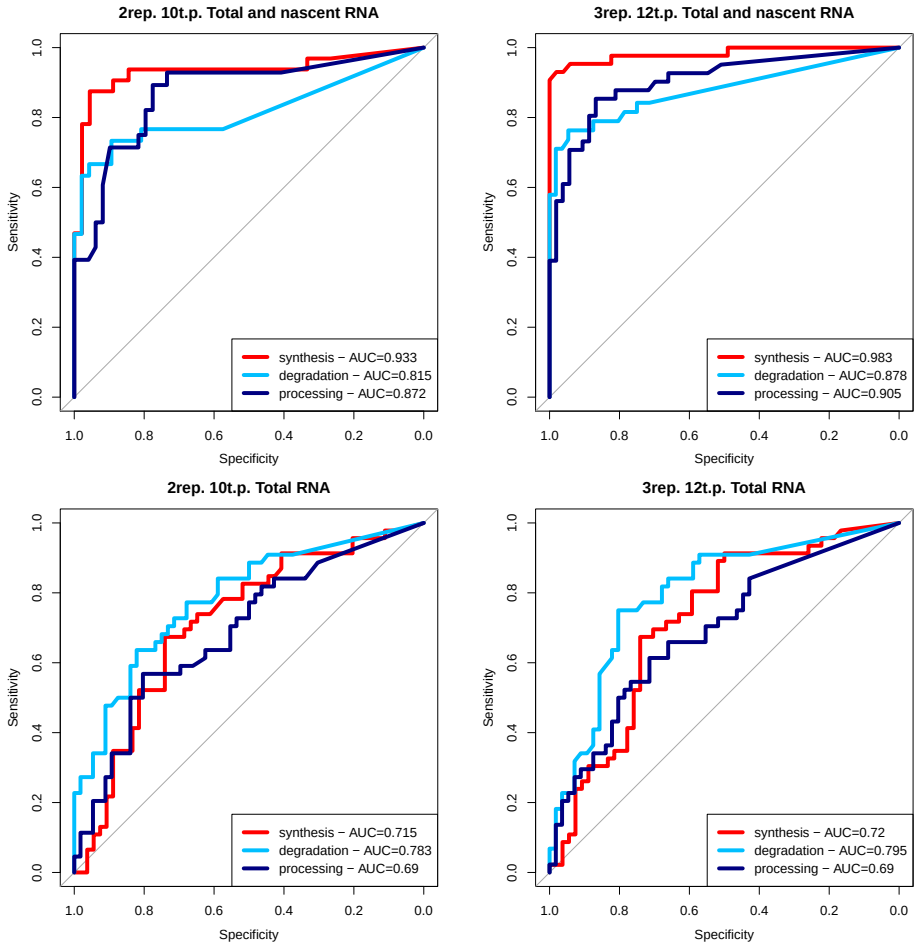


Figure 8: **ROC analysis of INSPEcT classification**
For each rate, INSPEcT classification performance is measured in terms of sensitivity, TP / (TP + FN), and specificity, TN / (TN + FP), using a ROC curve analysis; false negatives (FN) represent cases where the rate is identified as constant while 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; sensitivity and specificity are computed using increasing thresholds for the Brown method used to combine multiple p-values derived from the log-likelihood ratio tests

Further, the method `rocThresholds` can be used to assess the sensitivity and specificity that is achieved with different thresholds of the chi-squared and Brown tests (Figure 9).

```
rocThresholds(simRates[1:100],nascentSim2replicates,
  bTsh=c(.01,.01,.05),cTsh=.1)
```

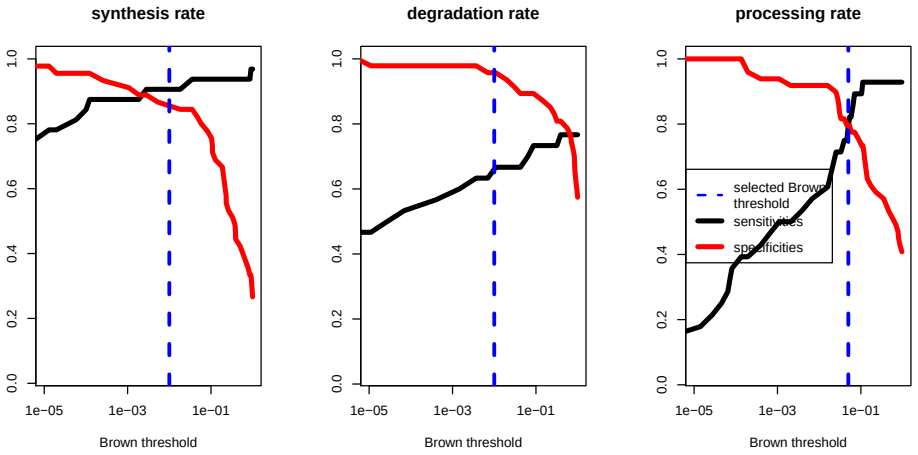


Figure 9: Effect of chi-squared and Brown tests thresholds in INSPEcT classification
Plot of the sensitivity (black curve) and specificity (red curve) that is achieved after performing the log-likelihood ratio and Brown method for combining p-values with selected thresholds; thresholds that can be set for chi-squared test to accept models that will undergo the log-likelihood ratio test and for Brown p-value to assess variability of rates

3.4 Parameters setting

If desired, different parameters can be set for both the modelling and the testing part. Regarding the modelling part, we might want to 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 improve the performance of the method, but also the needed computational time; the impact of these options on the quality of the modelling can be evaluated using simulated datasets.

```
nascentInspObj10<-removeModel(nascentInspObj10)

modelingParams(nascentInspObj10)$nInit<-20
modelingParams(nascentInspObj10)$nIter<-1000
```

Instead, 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 or not. The chi-squared test is used to discard bad models that will not enter in the model selection. By default, the threshold of acceptance is set to 0.2 and in this example we decide to be more stringent and set the threshold to 0.1. In case of low number of time points, few or no models can result in a low chi-square. In these case, it could be necessary to use larger values for the chi-squared test threshold, or be completely comprehensive and set the threshold to 1. The threshold relative to the Brown test is used to call differential regulation and can be set independently for each rate. Low threshold of the Brown test reflect in a more strict call for differential regulation of the rate. In this example, we are changing the thresholds of both the chi-squared test and the

Brown method for combining p-values. We are also imposing that only the constant models ("0") and processing varying model ("c") are tested one agains each other used to to assess the variability of the processing rates using log-likelihood ratio test. Usually, all nested models that differ for the processing rate are used ("0"vs"c", "c"vs"bc", "c"vs"ac" and "ab"vs"abc").

```
modelSelection(matureInspObj10)$thresholds$chisquare<-.1
modelSelection(matureInspObj10)$thresholds$brown<-c(synthesis=.01
                                    ,processing=.01
                                    ,degradation=.05)
```

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

```
## modelling
modelingParams(matureInspObj10)
```

```
## $nInit
## [1] 10
##
## $nIter
## [1] 300
##
## $na.rm
## [1] TRUE
##
## $Dmax
## [1] 10
##
## $Dmin
## [1] 1e-06
##
## $verbose
## [1] TRUE
##
## $estimateRatesWith
## [1] "int"
##
## $useSigmoidFun
## [1] TRUE
##
## $testOnSmooth
## [1] TRUE
##
## $seed
## [1] 1
```

```
## model selection and testing framework
```

```
modelSelection(matureInspObj10)
```

```
## $modelSelection
## [1] "llr"
##
## $preferPValue
## [1] TRUE
##
## $padj
## [1] TRUE
##
## $thresholds
## $thresholds$chisquare
## [1] 0.1
##
## $thresholds$brown
##   synthesis  processing degradation
##        0.01        0.01        0.05
##
##
## $limitModelComplexity
## [1] FALSE
```

4 Analysis of RNA dynamics in steady state RNA-seq data

4.1 Analysis of Total and Nascent RNA

INSPEcT is also designed to analyze RNA dynamics in steady state conditions and to compare the RNA dynamics of two steady state at time. In order to exemplify the procedure, we treat the data relative to the time-course analyzed above as 11 individual steady states. *INSPEcT* recognizes that the analysis is at steady state when a character vector of condtions is given as time-points.

```
conditions <- letters[1:11]
nasSteadyObj <- newINSPEcT(tpts=conditions
                        ,labeling_time=tL
                        ,nascentExpressions=nasExp_DESeq2
                        ,matureExpressions=matExp_DESeq2)
```

Once created the steady-state *INSPEcT* object, the first guess of the rates can be extracted with the method `ratesFirstGuess`, as usual. If we want to test for differential regulation in the RNA dynamics between two conditions, the method `com`

pareSteady can be used. The *INSPEcT* object created with 11 conditions can be subsetted in order to select the ones that we want to compare. In this case, we selected the first and the last conditions, tested for differential regulation and visualized the results relative to individual rates using the methods synthesis, processing and degradation. These methods return the rates computed by the modeling in the two conditions, the log transfromed geometric mean of the rate, the log2 fold change between the two conditions and the statistics relative to the test. In case a rate was not assessed as differentially regulated the same value is reported for the two conditions. Also in this case, the p-values thresholds of the chi-squared and Brown test can be modified by the modelSelection.

```
diffrates <- compareSteady(nasSteadyObj[,c(1,11)])

head(round(synthesis(diffrates),2))
```

```
##                a      k log2mean log2fc pval padj
## 100503670 231.12 248.14     7.90   0.10 0.19 0.28
## 101206      4.87   3.99     2.14  -0.29 0.01 0.02
## 101489      7.75   6.48     2.83  -0.26 0.18 0.27
## 102436     32.09  24.05     4.80  -0.42 0.59 0.72
## 104458     47.33  49.28     5.59   0.06 0.13 0.20
## 104806      3.50   3.98     1.90   0.18 0.60 0.72
```

```
head(round(processing(diffrates),2))
```

```
##                a      k log2mean log2fc pval padj
## 100503670  20.87  19.87     4.35  -0.07 0.07 0.38
## 101206     15.39  13.33     3.84  -0.21 0.18 0.49
## 101489     10.43   8.52     3.24  -0.29 0.58 0.71
## 102436    265.80 147.00     7.63  -0.85 0.00 0.22
## 104458     29.70  29.40     4.88  -0.01 0.62 0.73
## 104806     16.70  15.30     4.00  -0.13 0.03 0.31
```

```
head(round(degradation(diffrates),2))
```

```
##               a     k log2mean log2fc pval padj
## 100503670  0.39  0.40    -1.33   0.06 0.28 0.63
## 101206     1.02  1.09     0.07   0.09 0.34 0.64
## 101489     0.52  0.46    -1.03  -0.18 0.24 0.61
## 102436    22.56 17.52     4.31  -0.36 0.70 0.79
## 104458     0.55  0.50    -0.93  -0.15 0.04 0.34
## 104806     0.99  1.12     0.08   0.17 0.66 0.79
```

Following this, the method plotMA can be used to plot the results of the comparison for each indiviadual rate. In figure (Figure 10), we show the results for the synthesis rate. All the points that do not lie on the line $y = 0$ corresponds to genes called differentially regulated by the method compareSteady. This method select differentially regulated genes based on the p-value of the Brown test previosly chosen. Additionally,

it is possible to visualize genes that are differentially according to a certain threshold of the Benjamini and Hochberg correction of the p-values (argument **padj**). Those genes will be visualized as orange triangles.

```
plotMA(diffrates, rate='synthesis', padj=1e-3)
```

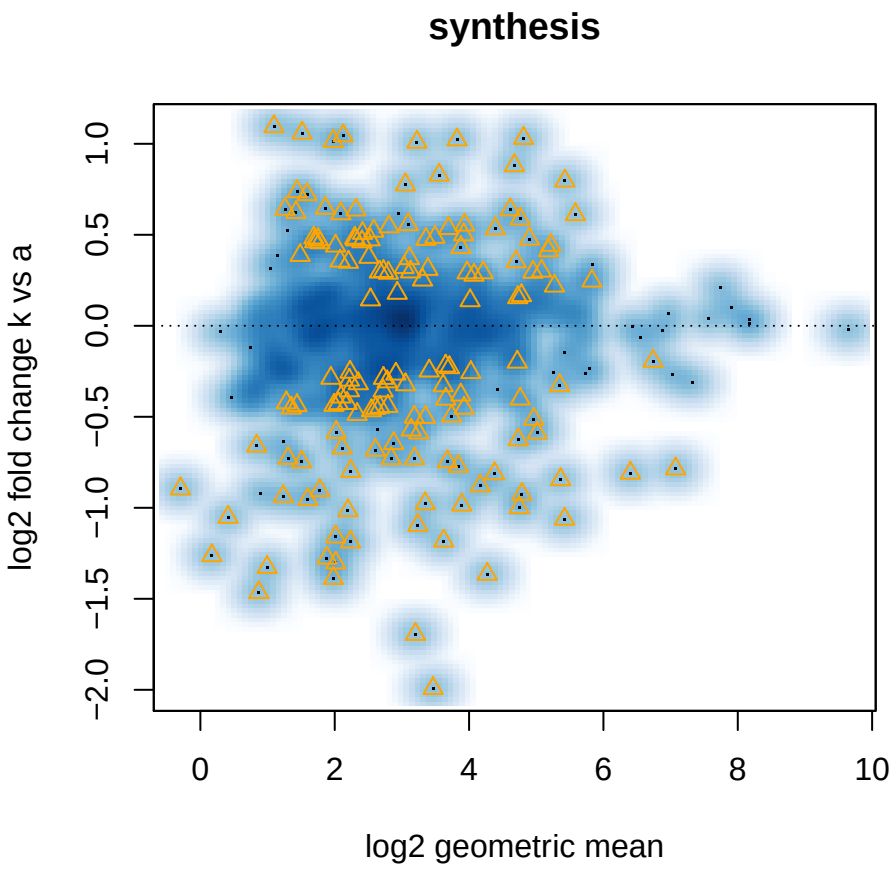


Figure 10: **Representation of rate geometric mean adn variation between conditions** plotMA generated image. Orange triangles correspond to genes whose rates are differentially used between the two conditions, blue cloud correspond to the whole distribution of rates.

4.2 Analysis of Total without Nascent RNA

It is not possible to estimate the kinetic rates from steady state data without nascent RNA expression data because of the underdetermination of the system (Equations 1). The only information that can be extracted is the ratio between the premature (P) and mature (M) RNA:

$$\frac{P}{M} = \frac{P_T}{T_T - P_T} = \frac{k_3}{k_2}$$

where P_R and T_R are the premature and total RNA levels, respectively, estimated from the total RNA library. However, even from this aggregated quantity, valuable information regarding the differential post transcriptional regulation of genes in different samples can still be obtained.
This analysis is performed by the metohd `compareSteadyNoNascent` which takes as input an object of class *INSPEcT* created witouh nascent RNA, an expression threshold for premature and total RNA, to select reliable data to be analysed (*expressionThreshold*) and the distance, expressed in log2, from the expected behaviour beyond which a gene should be considered as post-transcriptionally regulated.
We noticed that the ratio P over M decreases along with the expression following a power law, which can be easily fitted in the log log space by a linear model. This power law is calculated in the first step of the `compareSteadyNoNascent` analysis. Following that, genes that deviates from this trend across conditions are identified as differentially post-transcriptionally regulated.

```
conditions <- letters[1:11]
matureInspObj <- newINSPEcT(tpts=conditions
                            ,matureExpressions=matExp_DESeq2)
regGenes<-compareSteadyNoNascent(inspectIds=matureInspObj
                 ,expressionThreshold=0.25
                 ,log2FCThreshold=2.)
regGenes[1:6,1:5]
```

```
##           total_a total_b total_c total_d total_e
## 100503670   FALSE   FALSE   FALSE   FALSE   FALSE
## 101206      FALSE   FALSE   FALSE   FALSE      NA
## 101489      FALSE   FALSE   FALSE   FALSE   FALSE
## 102436         NA      NA      NA      NA      NA
## 104458      FALSE   FALSE   FALSE   FALSE   FALSE
## 104806         NA      NA      NA      NA      NA
```

```
table(regGenes, useNA='always')
```

```
## regGenes
## FALSE  <NA>
##  3722  1558
```

The output of the function is a matrix of booleans, as large as the expression input data, where TRUE means that a specific gene, in a given condition, is post-transcriptionally regulated in a peculiar way.
The entire analysis is based on the estimation of a standard behaviour of a set of genes which means that benefits from the size of the dataset under analysis. The example is, obviously, just a proff of concept to explicit arguments and output of `compareSteadyNoNascent`; for this reason, we have decided to use time-course data. A real application does not require data to be temporally related in any way.

5 Analysis of the delay introduced by the rates of processing to RNA responsiveness

During the transition between steady-states, RNA kinetic rates define the speed at which the mature form of a transcript can be set at a new level, i.e. the responsiveness. The rate of degradation is typically considered the major determinant of responsiveness: the lower is transcript stability, the higher is its responsiveness. Nonetheless, RNA processing rate influences RNA responiveness and specifically introduces a delay to it, which is commonly considered non-influent due to the speed at which the processing of the transcripts occur. To better elucidate the role of processing rate in this context, we devised two metrics, τ and Δ, that measure the additional time required to double mature RNA levels following a doubling in synthesis rates, compared to a scenario where processing occurs instantaneously (τ), and the number of RNA molecules involved in this delay (Δ). These metrics can be calculated from the kinetic rates of individual genes, or can be approximated with accuracy from premature and mature RNA abundances. The responsiveness of genes with high values for both metrics is thus significantly burdened by the processing step.
The two metrics, as well as the genes delayed by the processing rates, can be easily calculated from the *INSPEcT* object.

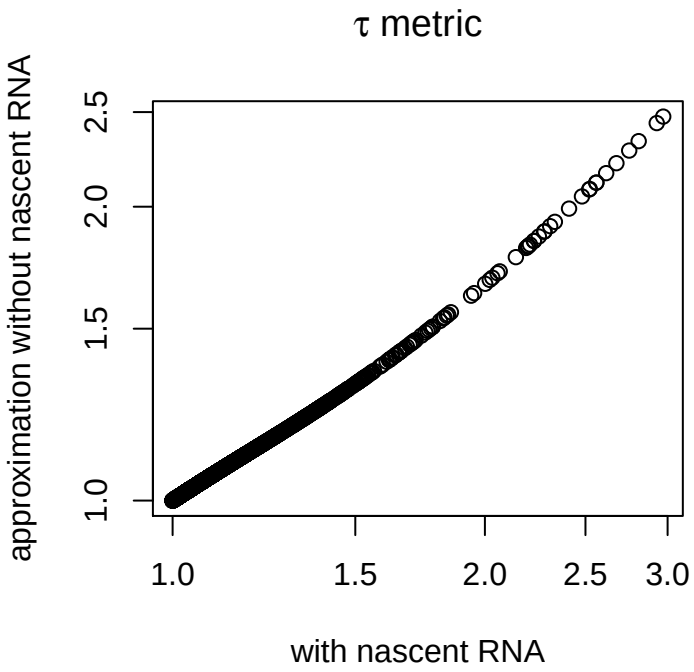
```
tau <- calculateTau(nasSteadyObj)

procDelay<- processingDelay(inspectIds=nasSteadyObj
                            ,tauThreshold=1.2
                            ,deltaThreshold=1.0)
table(procDelay, useNA='always')
```

```
## procDelay
## FALSE  TRUE  <NA>
##  5156    69     0
```

In case the object was created without the nascent RNA information, approximated metrics will be calculated.

```
tau_approx <- calculateTau(matureInspObj)
cg <- intersect(featureNames(nasSteadyObj), featureNames(matureInspObj))
plot(tau[cg,], tau_approx[cg,], log='xy'
    , main=expression(paste(tau,' metric'))
    , xlab='with nascent RNA'
    , ylab='approximation without nascent RNA')
```



6 About this document

```
sessionInfo()
```

```
## R version 3.5.3 (2019-03-11)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: OS X El Capitan 10.11.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] TxDb.Mmusculus.UCSC.mm9.knownGene_3.2.2
##  [2] GenomicFeatures_1.34.7
##  [3] AnnotationDbi_1.44.0
##  [4] GenomicRanges_1.34.0
##  [5] GenomeInfoDb_1.18.2
```

```
##  [6] IRanges_2.16.0
##  [7] S4Vectors_0.20.1
##  [8] INSPEcT_1.12.2
##  [9] BiocParallel_1.16.6
## [10] Biobase_2.42.0
## [11] BiocGenerics_0.28.0
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-6                matrixStats_0.54.0
##  [3] bit64_0.9-7                 RColorBrewer_1.1-2
##  [5] progress_1.2.0              httr_1.4.0
##  [7] tools_3.5.3                 backports_1.1.3
##  [9] R6_2.4.0                    KernSmooth_2.23-15
## [11] rpart_4.1-13                Hmisc_4.2-0
## [13] DBI_1.0.0                   lazyeval_0.2.2
## [15] colorspace_1.4-1            nnet_7.3-12
## [17] tidyselect_0.2.5            gridExtra_2.3
## [19] prettyunits_1.0.2           DESeq2_1.22.2
## [21] preprocessCore_1.44.0       bit_1.1-14
## [23] compiler_3.5.3              htmlTable_1.13.1
## [25] DelayedArray_0.8.0          rtracklayer_1.42.2
## [27] scales_1.0.0                checkmate_1.9.1
## [29] genefilter_1.64.0           stringr_1.4.0
## [31] digest_0.6.18               Rsamtools_1.34.1
## [33] foreign_0.8-71              rmarkdown_1.12
## [35] XVector_0.22.0              base64enc_0.1-3
## [37] pkgconfig_2.0.2             htmltools_0.3.6
## [39] highr_0.8                   htmlwidgets_1.3
## [41] rlang_0.3.3                 rstudioapi_0.10
## [43] RSQLite_2.1.1               acepack_1.4.1
## [45] dplyr_0.8.0.1               RCurl_1.95-4.12
## [47] magrittr_1.5                GenomeInfoDbData_1.2.0
## [49] Formula_1.2-3               Matrix_1.2-17
## [51] Rcpp_1.0.1                  munsell_0.5.0
## [53] stringi_1.4.3               pROC_1.14.0
## [55] yaml_2.2.0                  plgem_1.54.1
## [57] rootSolve_1.7               MASS_7.3-51.3
## [59] SummarizedExperiment_1.12.0 zlibbioc_1.28.0
## [61] plyr_1.8.4                  grid_3.5.3
## [63] blob_1.1.1                  crayon_1.3.4
## [65] lattice_0.20-38             Biostrings_2.50.2
## [67] splines_3.5.3               annotate_1.60.1
## [69] hms_0.4.2                   locfit_1.5-9.1
## [71] knitr_1.22                  pillar_1.3.1
## [73] geneplotter_1.60.0          biomaRt_2.38.0
```

```
## [75] XML_3.98-1.19              glue_1.3.1
## [77] evaluate_0.13              latticeExtra_0.6-28
## [79] data.table_1.12.0          BiocManager_1.30.4
## [81] deSolve_1.21               gtable_0.3.0
## [83] purrr_0.3.2                assertthat_0.2.1
## [85] ggplot2_3.1.0              xfun_0.6
## [87] xtable_1.8-3               survival_2.44-1.1
## [89] tibble_2.1.1               GenomicAlignments_1.18.1
## [91] memoise_1.1.0              cluster_2.0.7-1
## [93] BiocStyle_2.10.0
```