Visualization and interaction with the RNA dynamics of single genes

de Pretis S., Pelizzola M.

May 2, 2019

Contents

1 Introduction 2

2 Run the application from an R session. 2

3 Presentation of the Graphical User Interface 3

3.1 Interaction with an object of class INSPEcT 4

3.2 Visualization of the RNA dynamics for a single gene 5

3.3 Model minimization 6

3.4 Interaction with individual parameters of RNA rates 7

4 De novo hypothesis generation - Case studies 8

4.1 Constant kinetic rates. 9

4.2 Modulation of processing rate 10

4.3 Modulation of the synthesis rate 11

4.4 Concomitant modulation of synthesis and degradation rates . 12

5 About this document 13

1 Introduction

The life cycle of RNAs is composed of three main steps, i.e. transcription and processing of the premature RNA (P) and degradation of the mature (M). The kinetic rates governing these steps define the dynamics of each transcript (k_{1-3} for synthesis, processing and degradation, respectively), and their role in transcriptional regulation is often underestimated. A complete understanding of the effects of the rates of the RNA life-cycle on premature and mature RNA requires mathematical and/or computer skills to solve the corresponding system of differential equations:

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

This system of differential equations is used by INSPEcT to estimate the rates of the RNA-life cycle when transctiptomic data and (possibly) newly-synthesized RNA are available. INSPEcT aims at assessing the dynamics of each gene by modeling the temporal behavior of the RNA kinetic rates with either constant or variable functions. In order to visualize and interact with output of the classification procedure of INSPEcT, and to ease the understanding of the role of the RNA life-cycle kinetic rates in shaping the dynamics of premature and mature RNA, we developed a Graphical User Interface (GUI). Specifically, the GUI allows to: (i) interact with the results of INSPEcT analysis on single genes, refining a model, or testing modes of regulation that are alternative to the one identified by INSPEcT, and (ii) explore how a given combination of rates results in the dynamical behavior of premature and mature RNA species.

2 Run the application from an R session

The GUI is distributed within the INSPEcT package, and starts with the following command line operations:

```
library(INSPEcT)
runRNAdynamics()
```

3 Presentation of the Graphical User Interface

The GUI is divided into 4 sections (Fig. 1), which account for specific functions:

- Section 1: interaction with an object of class INSPEcT;
- Section 2: visualization of the RNA dynamics for a single gene;
- Section 3: model minimization;
- Section 4: interaction with individual parameters of RNA rates.

Figure 1: Representation of the GUI divided into its 4 main sections

3.1 Interaction with an object of class INSPEcT

At startup, the software loads a predefined INSPEcT object, which contains 10 genes and can be used to explore the software functionalities. This object can be replaced by any INSPEcT dataset generated by the user, and saved in the "rds" format ("Choose INSPEcT file", Fig. 2). Genes that are part of the INSPEcT object are divided according to their regulation class. This is encoded by a three digit string, where each digit represent a step in the RNA life-cycle (in the order: synthesis, processing and degradation), and the character represents the regulation status of the rate ("K"=constant, "V"=variable). For example: "KVK" represent a gene regulated in its processing rate, while constant in its synthesis and degradation rates. Once a regulation class is selected via "Select class", a specific gene can be chosen from the list that appears in "Select gene" (Fig. 2).

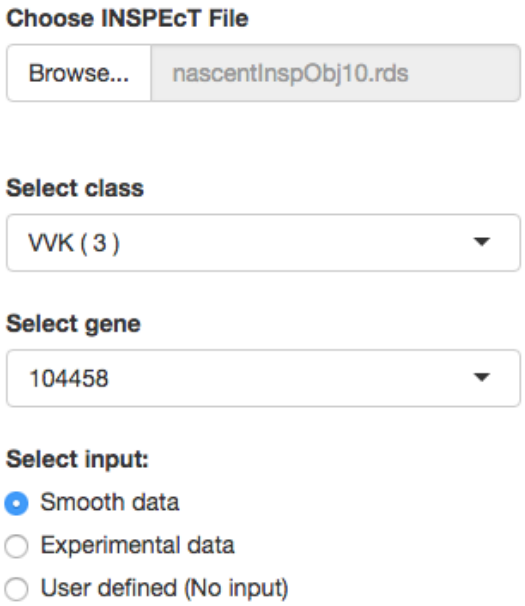

Figure 2: Interaction with an object of class INSPEcT (section 1)

3.2 Visualization of the RNA dynamics for a single gene

For the selected gene, the experimental quantifications of the premature and mature RNA levels (estimated from RNA-seq data) are plotted together with their standard deviations (Fig. 3). In case the nascent RNA have been profiled, also the synthesis rate is considered part of the experimental data and plotted with its standard deviation. Otherwise, the rate of synthesis can be inferred from total RNA-seq data. In this case, nascent RNA data is not necessary but the rate of synthesis lacks the standard deviation. Most of the times, due to noise associated with this kind of data, smoothing the experimental profiles ("Smooth data" in "Select input", Fig. 2) can lead to more precise identification of the regulation pattern. Alternatively, the user can use raw data by selecting "Experimental data" in "Select input" (Fig. 2). In addition to experimental or smoothed data, the results of the INSPEcT modeling are plotted with continuous lines within the synthesis, pre-RNA, processing, mature RNA and degradation panels (Fig. 3). The "User defined" mode in "Select input" will be covered in section 4.

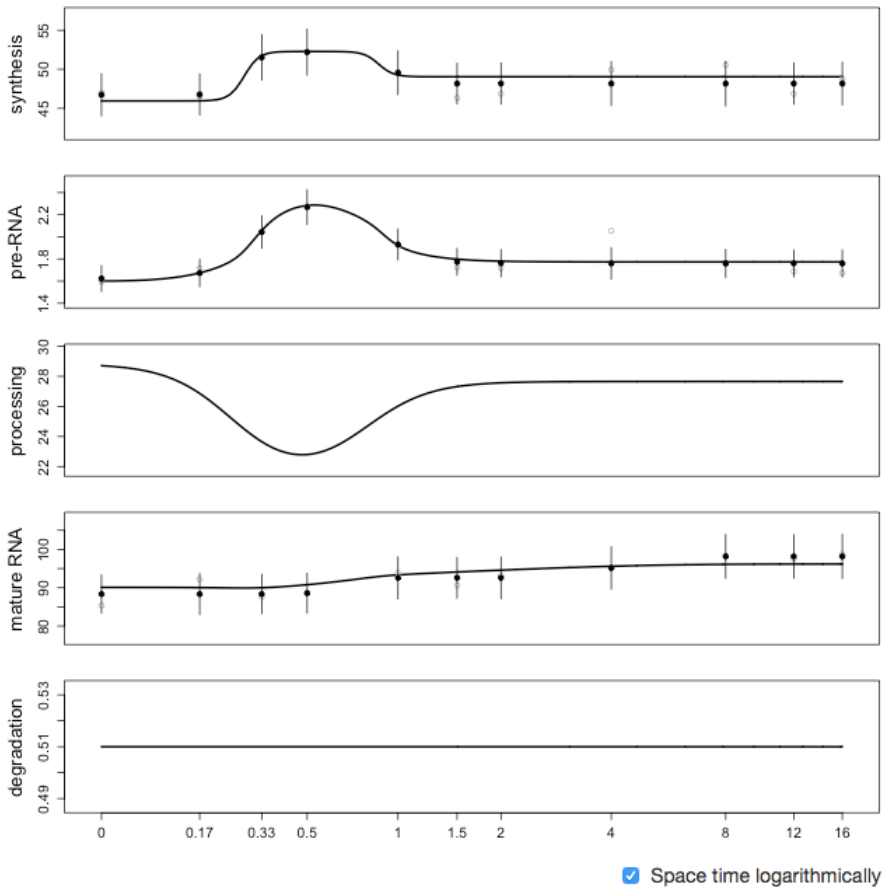

Figure 3: Visualization of the RNA dynamics for a single gene (section 2)

3.3 Model minimization

The minimization status corresponding to the modeling are reported in section 3 of the GUI (Fig. 4). In particular, the p-value associated to the chi-squared statistic and the Akaike information criterion indicate the goodness of the fit. Both these metrics are penalized for the complexity of the model, meaning that they measure a trade-off between performance and simplicity and can be used to compare models with different complexity. The complexity of a model depends on the functional forms that describe the RNA life-cycle kinetic rates: a constant rate has a complexity of 1, a sigmoidal 4, and impulsive 6, i.e. the number of their parameters. In practice, when two models explains the data adequately well, the simpler one is selected due to its simplicity (lower p-value of the chi-squared statistic and lower AIC). Additionally, the chi-squared p-value can be used to assess whether the model under consideration adequately explains the experimental (or smooth) data (when $p<0.05$). Finally, the number of iterations spent trying to reach the minimum is reported, as well as whether the minimization converged or not to a local minimum. In case the minimization status is not satisfactory, supplementary iterations can be provided to search for a better minimum, either using the Nelder Mead simplexes method (NM, used from INSPEcT) or the quasi-Newton BFGS method, by pushing the button "Optimize - Run".

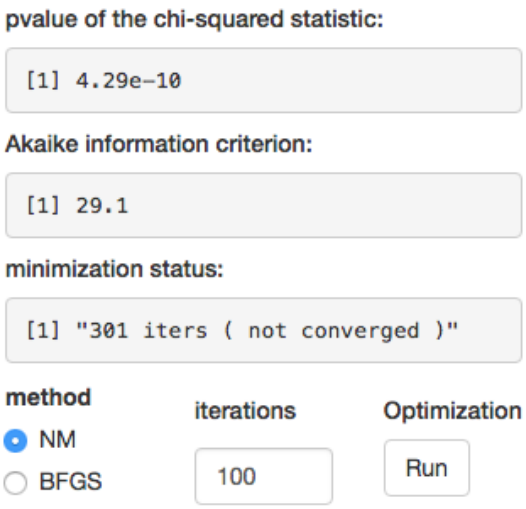

Figure 4: Model minimization (section 3)

3.4 Interaction with individual parameters of RNA rates

Parameters of the modeling can be directly modified "by hand". In fact, for each kinetic rate of the RNA life-cycle, the parameters describing the selected functional form are provided in the bottom part of the GUI (Fig. 5). Constant rates are described by a single parameter, which correspond to the value of the rates throughout the time-course, alternatively, variable rates can be described either by sigmoidal or impulsive functions. Sigmoidals are S-shaped functions described by four parameters: starting levels, final levels, time of transition between starting and final levels, and slope of the response. Impulsive functions allow more complex behaviors with two additional parameters that describe time and levels of a second transition, possibly encoding for bell-shaped responses. Minimum and maximum levels can be set individually per each rate, allowing the user maximum flexibility in the definition of rate levels. At startup, these ranges are set to include all the parameters of the sample dataset. Each time a new dataset is loaded, ranges are updated accordingly.

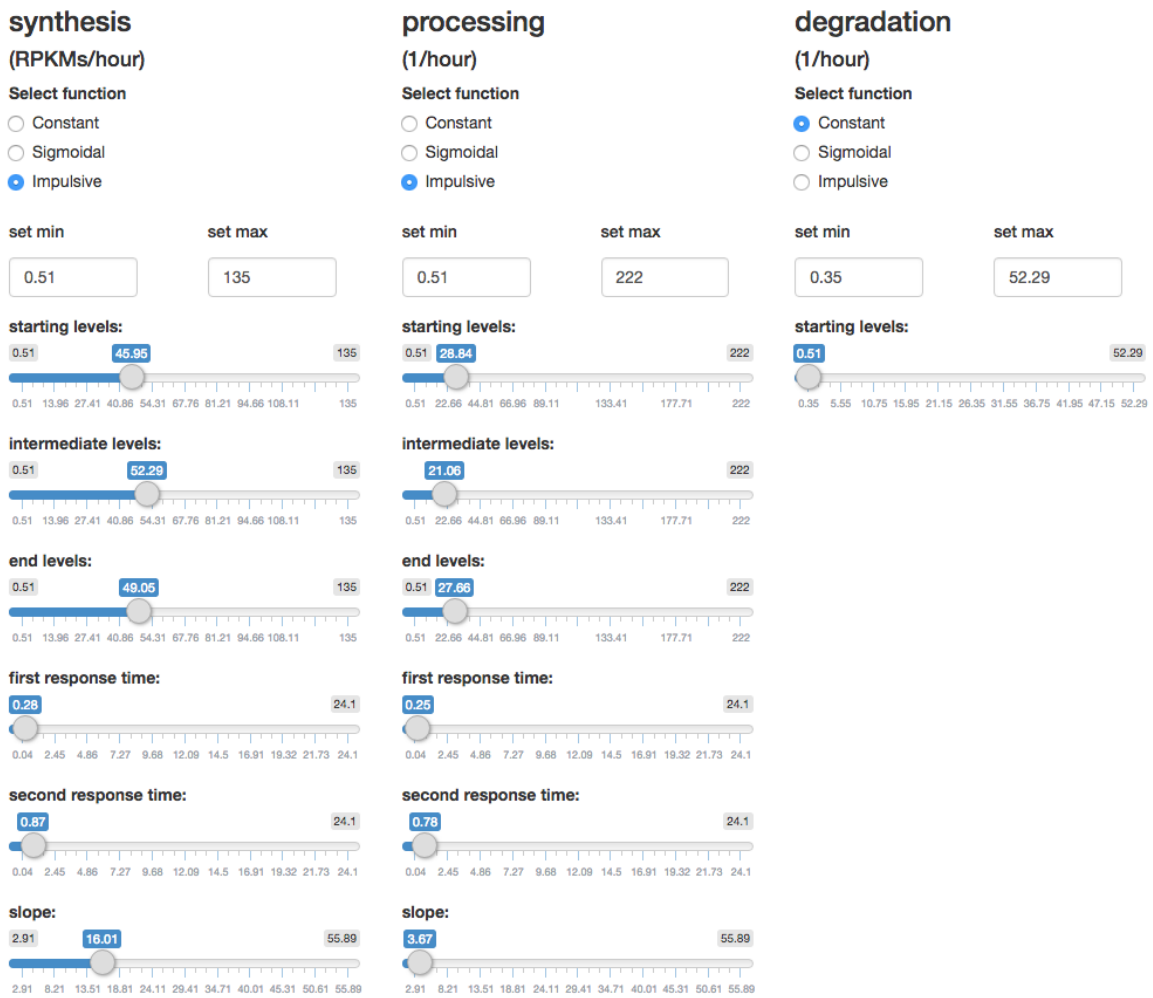

Figure 5: Direct interaction with RNA life-cycle kinetic rates (section 4)

The user can test different regulative scenarios from the one selected by INSPEcT, by tuning each function parameter or switching the functional form assigned to one or more rates. In the latter case, the new parameter set can be searched via the minimization of the error over the data (3.3) and/or "by hand" the best combination of parameters. Each time the model is modified, its plot and minimization status are updated.

4 De novo hypothesis generation - Case studies

The procedure explained in the previous sections mostly refers to the interaction with experimental data. Nonetheless, the software is designed also to test hypothesis independently from experiments. This can be easily achieved by choosing "User defined (No input)" in "Select input" (Fig. 2). In this mode, experimental or smooth data are not represented, as well as the minimization status. The user is therefore free to change both the functional forms and the parameters associated to the RNA life-cycle kinetic rates, directly assessing the effects on premature and mature RNA dynamics. In the following subsections, we report some case studies to exemplify the role of the RNA life-cycle in the definition of premature and mature RNA dynamics.

4.1 Constant kinetic rates

The combination of the RNA kinetic rates establishes the levels of mature and pre-mature RNA: the ratio between the rates of synthesis and processing sets premature RNA levels, and the ratio between synthesis and degradation sets the mature ones. When no perturbations occur, this long standing condition is called steady state. (Set rates to: synthesis = constant{10}; processing = constant{20}; degradation = constant{2})

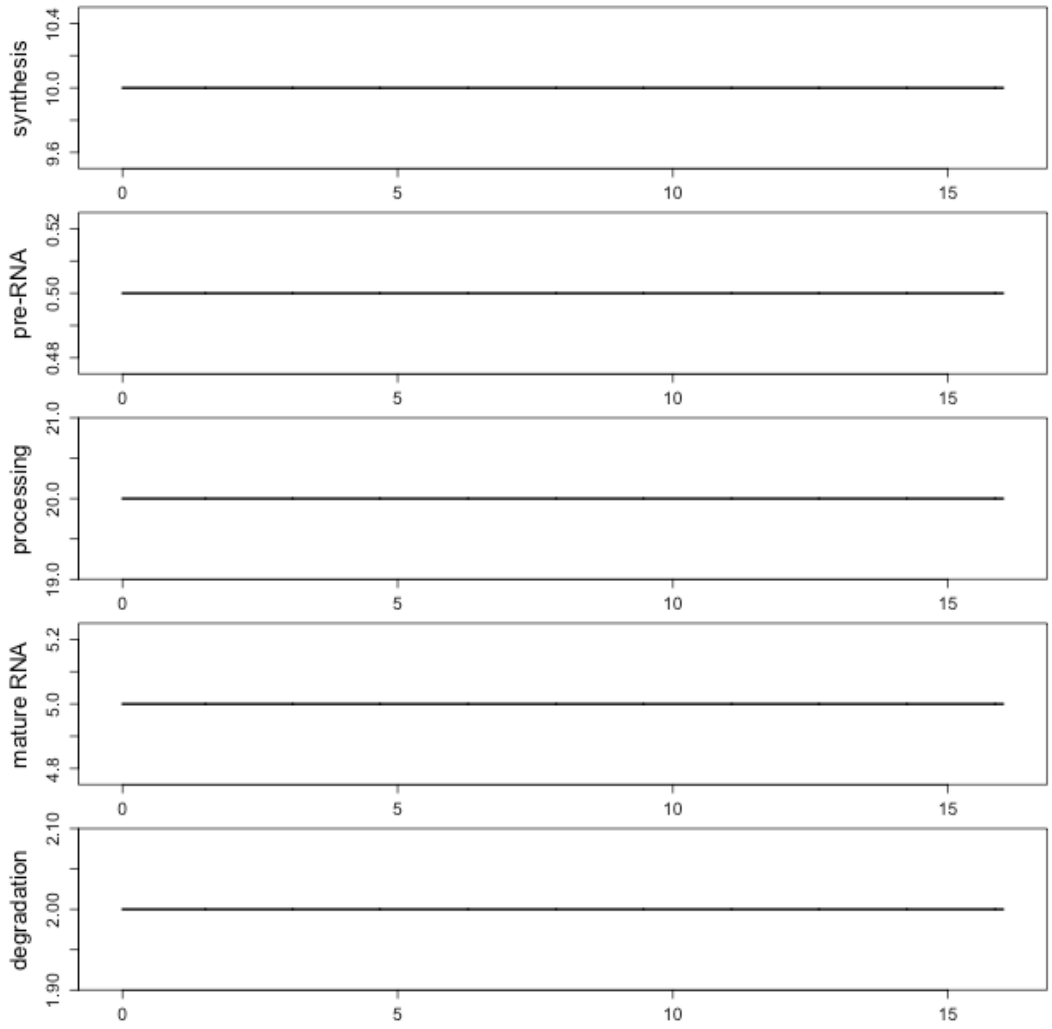

Figure 6: Case study 1: constant kinetic rates

As a consequence of this, steady state premature RNA levels are independent from degradation rates, and (more importantly and less intuitively) steady state mature RNA levels are independent from processing rates.

4.2 Modulation of processing rate

As mentioned above, steady state mature RNA levels are independent from processing rates. Nonetheless, a perturbation in the processing dynamics leads to a variation in mature RNA levels, which is just temporary. Conversely, it produces a lasting effect on premature RNA levels. (Set rates to: synthesis = constant{10}; processing = sigmoidal{5,20,8,5}; degradation = constant{2})

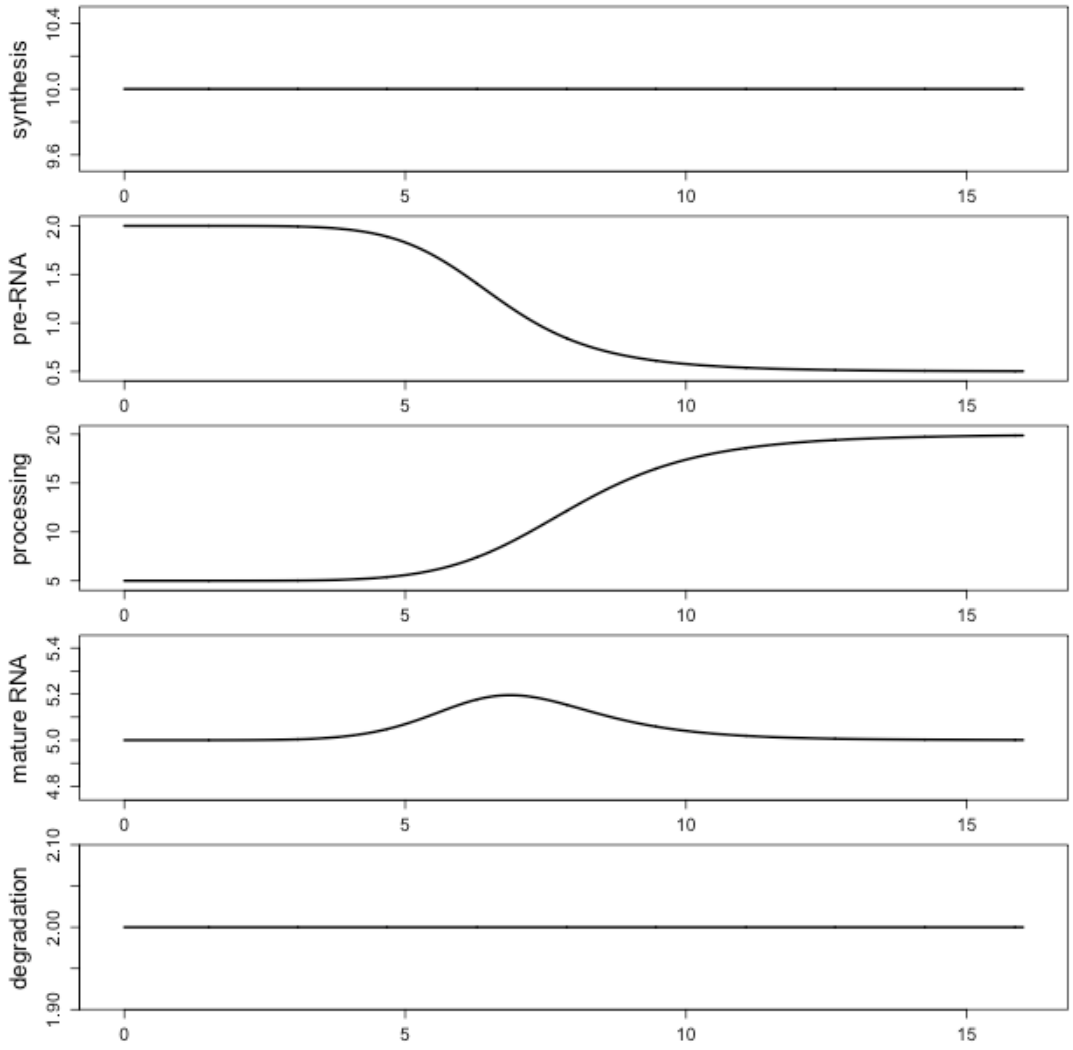

Figure 7: Case study 2: modulation of processing rate

Note that the decrease of the degradation rate makes the perturbation of the mature RNA less visible. (Set degradation = constant{0.5})

4.3 Modulation of the synthesis rate

A modulation in the synthesis rate determines a similar modulation for premature and mature RNAs (in terms of fold change compared to the untreated condition). (Set rates to: synthesis = sigmoidal{10,20,8,5}; processing = constant{20}; degradation = constant{2})

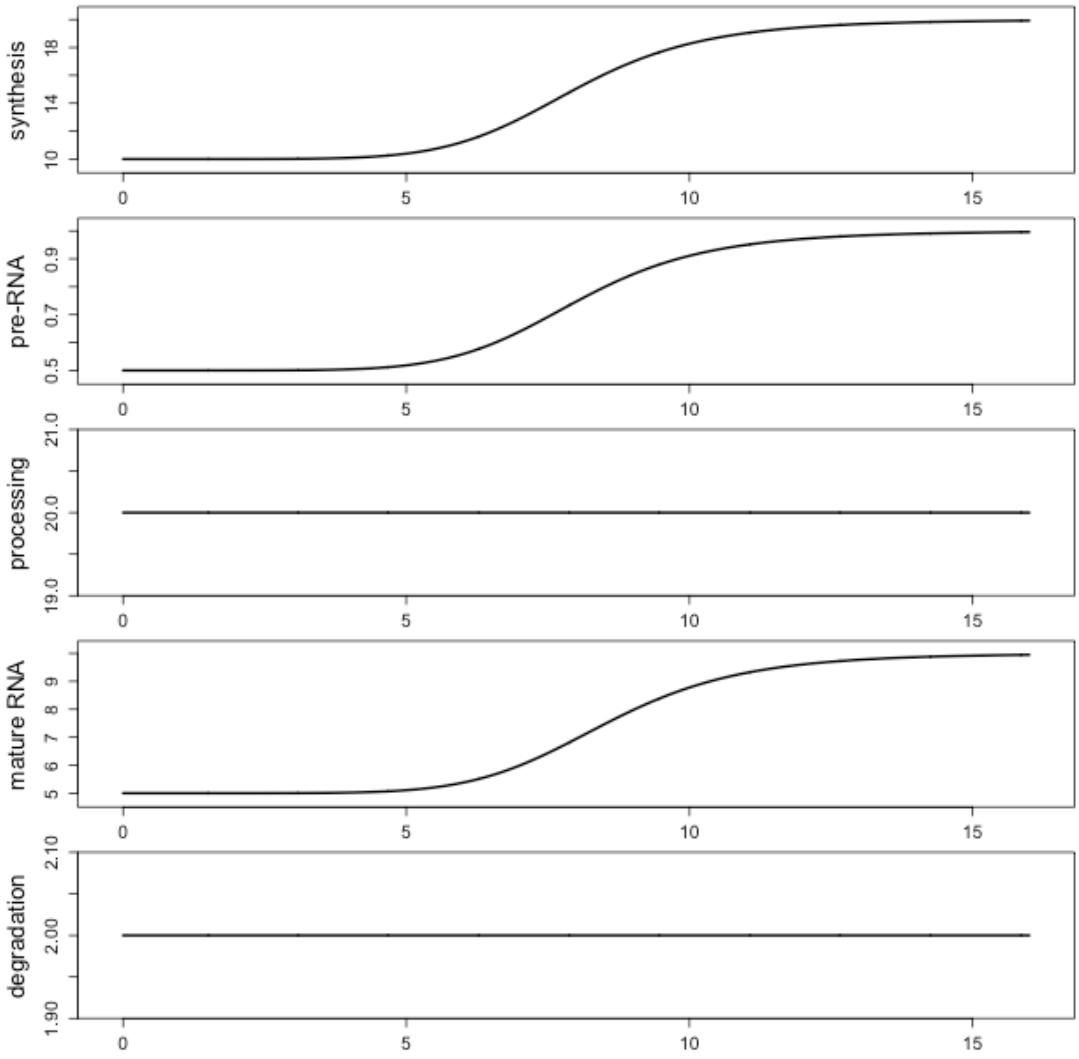

Figure 8: Case study 3: modulation of syntheis rate

Note that the time lag between the synthesis and premature RNA response is inversely proportional to the processing rate (Set processing = constant{10}). Following the same logic, the time lag between the premature and the mature RNA response is inversely proportional to the degradation rate (Set degradation = constant{0.5}).

4.4 Concomitant modulation of synthesis and degradation rates

When synthesis and degradation rates are modulated in the same direction they apply an opposite effect on mature RNA levels, leaving trace of the transcriptional regulation only on premature RNA levels (a similar response have been observed in yeasts, during heat shock). (Set rates to: synthesis = sigmoidal{10,20,8,5}; processing = constant{20}; degradation = sigmoidal{0.25,0.5,8,5}). Rather, their opposite modulation reinforces the transcriptional response of mature RNA. (Set degradation = sigmoidal{0.5,0.25,8,5})

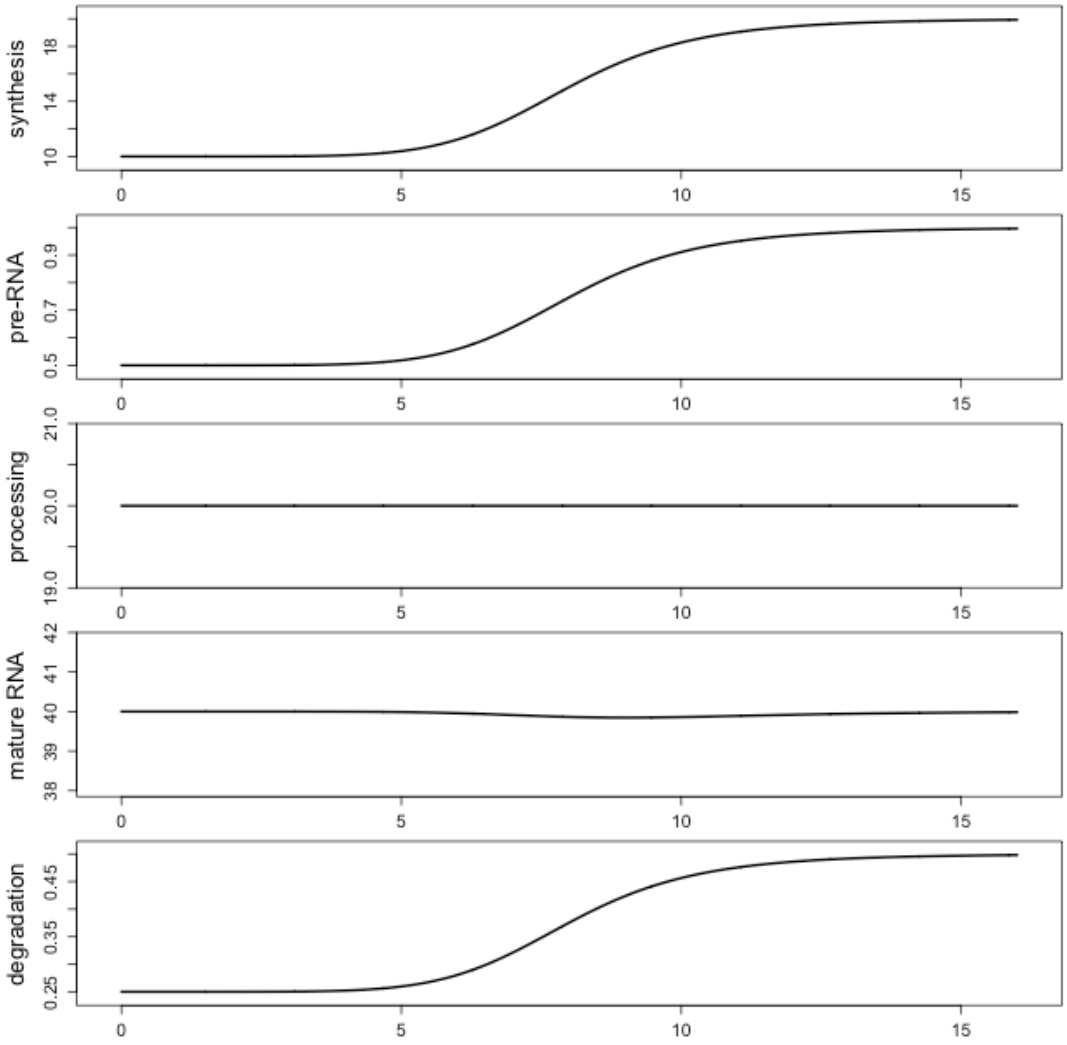

Figure 9: Case study 4: concomitant modulation of synthesis and degradation rates

5 About this document

```
sessionInfo()
```

```
## R version 3.6.0 (2019-04-26)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.2 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.9-bioc/R/lib/libRblas.so
## LAPACK: /home/biocbuild/bbs-3.9-bioc/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## 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_1.36.0
##  [3] AnnotationDbi_1.46.0
##  [4] GenomicRanges_1.36.0
##  [5] GenomeInfoDb_1.20.0
##  [6] IRanges_2.18.0
##  [7] S4Vectors_0.22.0
##  [8] INSPEcT_1.14.0
##  [9] BiocParallel_1.18.0
## [10] Biobase_2.44.0
## [11] BiocGenerics_0.30.0
##
## 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_3.6.0                backports_1.1.4
##  [9] R6_2.4.0                   KernSmooth_2.23-15
## [11] rpart_4.1-15               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_1.24.0
## [21] preprocessCore_1.46.0       bit_1.1-14
## [23] compiler_3.6.0              htmlTable_1.13.1
## [25] DelayedArray_0.10.0         rtracklayer_1.44.0
## [27] scales_1.0.0                checkmate_1.9.1
## [29] genefilter_1.66.0           stringr_1.4.0
## [31] digest_0.6.18               Rsamtools_2.0.0
## [33] foreign_0.8-71              rmarkdown_1.12
## [35] XVector_0.24.0              base64enc_0.1-3
## [37] pkgconfig_2.0.2             htmltools_0.3.6
## [39] highr_0.8                   htmlwidgets_1.3
## [41] rlang_0.3.4                 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_1.2.1
## [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_1.56.0
## [57] rootSolve_1.7               MASS_7.3-51.4
## [59] SummarizedExperiment_1.14.0 zlibbioc_1.30.0
## [61] plyr_1.8.4                  grid_3.6.0
## [63] blob_1.1.1                  crayon_1.3.4
## [65] lattice_0.20-38             Biostrings_2.52.0
## [67] splines_3.6.0               annotate_1.62.0
## [69] hms_0.4.2                   locfit_1.5-9.1
## [71] knitr_1.22                  pillar_1.3.1
## [73] geneplotter_1.62.0          biomaRt_2.40.0
## [75] XML_3.98-1.19               glue_1.3.1
## [77] evaluate_0.13               latticeExtra_0.6-28
## [79] data.table_1.12.2           BiocManager_1.30.4
## [81] deSolve_1.21                gtable_0.3.0
## [83] purrr_0.3.2                 assertthat_0.2.1
## [85] ggplot2_3.1.1               xfun_0.6
## [87] xtable_1.8-4                survival_2.44-1.1
## [89] tibble_2.1.1                GenomicAlignments_1.20.0
## [91] memoise_1.1.0               cluster_2.0.9
## [93] BiocStyle_2.12.0
```