Tutorial for R package proRate

Yu Liu

2025-5-9

## Introduction

The dynamics of transcriptional elongation can influence various post-transcriptional processes, such as splicing, polyadenylation, and nuclear export *[1-5]*. To quantify the elongation rate, a typical method is to treat cells with drugs able to inhibit Polymerase II (Pol II) from entering the gene bodies and starting transcription, such as DRB (5,6-dichloro-1-beta-d-ribofuranosylbenzimidazole), and then track Pol II using Pro-seq or Gro-seq *[6, 7]*. As the DRB-dependent blocking of Pol II entry persists, a read blank region will form on the gene body because few Pol II enter it. Downstream of this blank region (Pol II depleted state) is the intact read region (Pol II occupied state) formed by the unaffected Pol II having entered the gene body before the drug blocking. Hence, the length of the blank region is considered the distance Pol II should have gone through during the blocking period, and the corresponding Pol II transcription rate can be obtained based on this distance and the blocking time.

To get the distance, the transition point between the Pol II depleted state and the occupied state must be identified. Traditionally, a 2-state hidden Markov model (HMM) is used to infer this elongation rate from Pro-seq or Gro-seq data *[6]*. However, this method is complicated with many parameters to be estimated and the model’s hidden variable to be solved by the expectation-maximization (EM) iteration *[8]*. In addition, it typically assumes that the observed continuous data follow a normal distribution, which cannot always be fulfilled. Hence, we tried to solve the problem using a different method and developed the R package *proRate*. It identifies the transition point via a novel least sum of squares (LSS) method, which is more efficient than HMM and has no requirement on the data distribution. In addition, this package also offers other functions frequently used when studying transcription dynamics, such as metagene plotting, pausing index calculation, and differential gene identification.

## Package installation

Code and documentation of *proRate* are freely available at <https://github.com/yuabrahamliu/proRate>, or <https://github.com/FADHLyemen/proRate>.

The following commands can be used to install this R package.

library(devtools)  
  
install\_github("yuabrahamliu/proRate")

## Data preparation

To demonstrate the functions of *proRate*, this tutorial uses some data accompanying the package, including four bam files.

All four bam files are from a previous work studying the influence of the elongation factor Paf1C on RNA polymerase II transcription rate *[6]*. Shortly, gene editing was used to produce mouse myoblast C2C12 cells with the factor Paf1C conditionally knocked out (KO cells), while some wild C2C12 cells with intact Paf1C expression were used as controls (WT cells). Then, both the KO cells and WT cells were divided into two groups; one of them was treated with the drug DRB to block RNA polymerase II from entering the gene bodies for 15 min (15 min group), while the other was not treated with DRB (0 min group). Hence, the four bam files represent four experimental conditions. They are WT cells + 0 min DRB (wt0), WT cells + 15 min DRB (wt15), KO cells + 0 min DRB (ko0), and KO cells + 15 min DRB (ko15). After the DRB treatment step, Pro-seq was used to sequence nascent RNA for the C2C12 cells in these four conditions, and the bam files were transformed from these Pro-seq data. To simplify the analysis in this tutorial, only reads from mouse chromosome 19, the smallest autosome in the mouse genome, are included in the four bam files.

Next, attach *proRate* to the R session and get the directories of its accompanied example files above.

library(proRate)  
  
wt0file <- system.file("extdata", "wt0.bam", package = "proRate")  
wt15file <- system.file("extdata", "wt15.bam", package = "proRate")  
ko0file <- system.file("extdata", "ko0.bam", package = "proRate")  
ko15file <- system.file("extdata", "ko15.bam", package = "proRate")

## Rate inference

We first use the function calrate to calculate the transcription elongation rates for genes in WT C2C12 cells.

The two most important parameters are time1file and time2file. Here, time1file corresponds to the Pro-seq bam file used as a reference, i.e., with a DRB treatment time of 0 min, and the directory of the wt0 file taken above should be transferred. The other time2file corresponds to the bam file used as an experimental group with a DRB treatment for more than 0 min. Here, the directory of the wt15 file should be transferred. The time difference between these 2 files is 15 min, so for another parameter time, its value is 15. In addition, because the Pro-seq sequencing libraries of these bam files were prepared using the directional ligation method, the parameter strandmethod should be set as 1. On the other hand, if the dUTP method is used to construct libraries, it should be set as 2.

Then, the parameter genomename is set as “mm10”, which means that in the above time1file and time2file, the genes in the mouse “mm10” genome should be analyzed. In addition, the parameters genelencutoff and fpkmcutoff, which are set as 40000 and 1, respectively, further require only the genes with a length greater than 40000 bp and an FPKM expression value greater than 1 in time1file will be analyzed.

Because bam file processing is time-consuming, calrate provides a parallelization option. Its parameter threads can be set to a multiple-thread number to save computing time.

Then, the parameters startshorten and endshorten are set to 1000, which means that before inferring a gene’s transcription rate, its first 1000 bp and last 1000 bp regions will be discarded to avoid unstable reads at the transcription starting and ending stages. In addition, these two parameters also set a cutoff for the genes, i.e., only those with a length greater than 2\*(startshorten + endshorten) will be included in the analysis.

After discarding the first and last regions, a gene will be divided into 40 bins, which is indicated by setting the parameter window\_num as 40. Then, for each bin, the normalized read count ratio between the treatment time2file and the reference time1file will be calculated, generating a vector with 40 ratios. Then, the LSS method will be used to find the transition bin between the gene’s transcription-inhibited region and the normal read region. After that, this identified transition bin and its downstream neighbor will be merged and expanded to the single-base level, and the LSS method will be further used on them to find the transition base pair in this region.

Because the LSS method is used for the rates inference, the corresponding parameter method should be set as “LSS”. In addition, it can also be set as “HMM”, so the traditional hidden Markov model will be used instead. However, because HMM needs to communicate with some *Python* functions while performing the inference, another parameter, pythonpath, should be set for the HMM case. It is the directory of the *Python* interpreter to be used, and two *Python* modules should be installed in this *Python* environment, i.e., numpy and hmmlearn.

Finally, the parameter difftype should be set as 1 here, which means the treatment and reference Pro-seq files are from experiments treating cells with transcription inhibitors, such as DRB, so that the normal transcription will be repressed for a specific time, generating a read depleted region upstream of the normal transcription region. However, this parameter can also be set as 2 if the treatment and reference files are from experiments treating cells with transcription activators, e.g., treating MCF-7 human breast cancer cells with E2 (17-beta-estradiol) and making the read depleted region downstream rather than upstream, of the normal transcription region.

Hence, the transcription rates for the genes in WT C2C12 cells can be calculated using the following command.

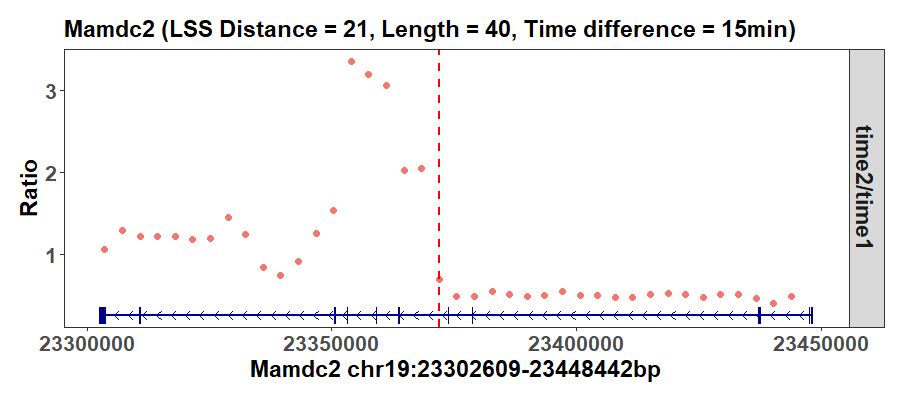
wtrates <- calrate(time1file = wt0file,   
 time2file = wt15file,   
 time = 15,   
 strandmethod = 1,   
   
 genomename = "mm10",   
 lencutoff = 40000,   
 fpkmcutoff = 1,   
   
 threads = 1,   
   
 startshorten = 1000,   
 endshorten = 1000,   
 window\_num = 40,   
   
 method = "LSS",   
 pythonpath = NULL,   
   
 difftype = 1)

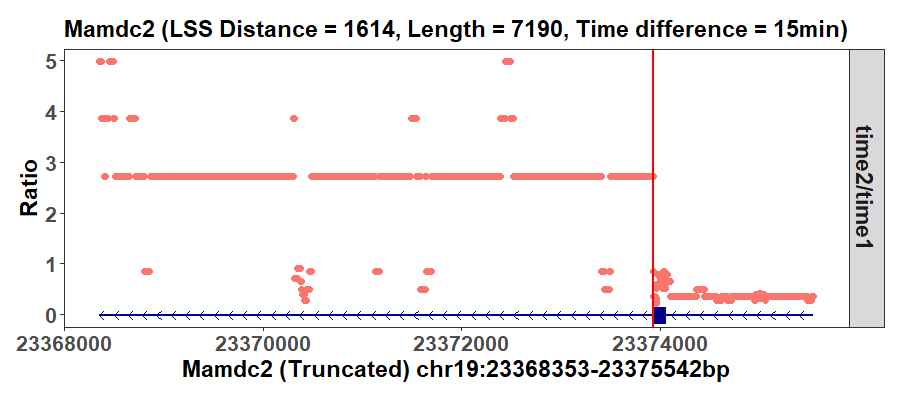
The result wtrates is a list, and its most important element is a data frame named “report”. It contains the distance that Pol II passes (i.e., the length of the Pol II depleted region) on each gene, which is inferred using the LSS method here. In addition, it also includes the Pol II travel time (15 min here for all genes), the final rate, the statistical significance of the result, and many other information.

head(wtrates$report)  
#> gene\_id distance time rate significance binpadj binpval  
#> 1 Mamdc2 74514 15 4967.6000 significant 2.651728e-10 1.523433e-11  
#> 2 Cpt1a 27323 15 1821.5333 significant 2.651728e-10 2.253969e-11  
#> 3 Dagla 35638 15 2375.8667 significant 2.651728e-10 3.182073e-11  
#> 4 Kmt5b 25652 15 1710.1333 significant 4.606572e-08 7.370516e-09  
#> 5 Cemip2 14617 15 974.4667 significant 5.176325e-07 1.072756e-07  
#> 6 Mark2 54574 15 3638.2667 significant 1.628291e-06 5.210532e-07  
#> frontbinratio latterbinratio diffbinratio chr start end strand  
#> 1 0.5046868 1.579635 1.0749482 chr19 23302609 23448442 -  
#> 2 0.5561864 1.535600 0.9794137 chr19 3322334 3385733 +  
#> 3 0.6606073 1.488996 0.8283882 chr19 10245265 10304877 -  
#> 4 0.7112802 1.213760 0.5024796 chr19 3767421 3818303 +  
#> 5 0.2198139 1.577981 1.3581669 chr19 21778342 21858360 +  
#> 6 0.8921235 1.722870 0.8307461 chr19 7275396 7341860 -  
#> extendpadj extendpval frontextendratio latterextendratio  
#> 1 0.000000e+00 0.000000e+00 0.3846782 2.669113  
#> 2 0.000000e+00 0.000000e+00 0.6691627 2.415501  
#> 3 0.000000e+00 0.000000e+00 0.9296687 1.889518  
#> 4 1.062347e-201 9.773596e-202 0.9027819 1.275751  
#> 5 0.000000e+00 0.000000e+00 0.2945463 1.648696  
#> 6 2.367902e-227 2.083754e-227 0.7124585 1.510239  
#> diffextendratio genewidth GC exon fpkm  
#> 1 2.2844351 145834 0.4232621 0.02694845 5.043759  
#> 2 1.7463384 63400 0.4972871 0.12053628 42.632969  
#> 3 0.9598489 59613 0.5415430 0.09640515 57.940221  
#> 4 0.3729695 50883 0.4357054 0.28992001 58.911502  
#> 5 1.3541492 80019 0.4390207 0.10647471 68.830831  
#> 6 0.7977802 66465 0.4751674 0.11949146 75.033895

As mentioned above, to infer a gene’s transition point between the Pol II depleted region and the occupied region, *proRate* first divides the gene body into 40 bins, which can also be set as other bin numbers via the parameter window\_num, and then calculates the normalized read ratio between time2file and time1file for each bin to identify the transition bin. After that, the single-base-level transition point can be further identified within the merged region of this bin and its downstream neighbor. Both the LSS and HMM methods can be used to infer the transition bin and transition base pair. From the “report” slot of wtrates above, the gene Mamdc2 has the most significant difference between its two regions. Then, another function, addtrack, can be used to show the plots for its bin-level and base-level ratios and the inferred transition bin and base pair. Its parameter genedat accepts a subset of wtrates’s “report” slot. Here, it is the one only containing the row for the gene Mamdc2. The parameters binplotdat and expandplotdat accept the plot data for the bin level and base level, which can be reached from wtrates’s other two slots, i.e., “binplots” and “expandplots”. The parameters genomename and method are the ones previously used by calrate, and the parameters textsize, titlesize, face are used to control the font size and face of the texts and title of the plots.

addtrack(genedat = subset(wtrates$report, gene\_id == "Mamdc2"),   
 binplotdat = wtrates$binplots$Mamdc2,   
 expandplotdat = wtrates$expandplots$Mamdc2,   
 genomename = "mm10",   
 method = "LSS",   
 titlesize = 17,   
 textsize = 16,   
 face = "bold")





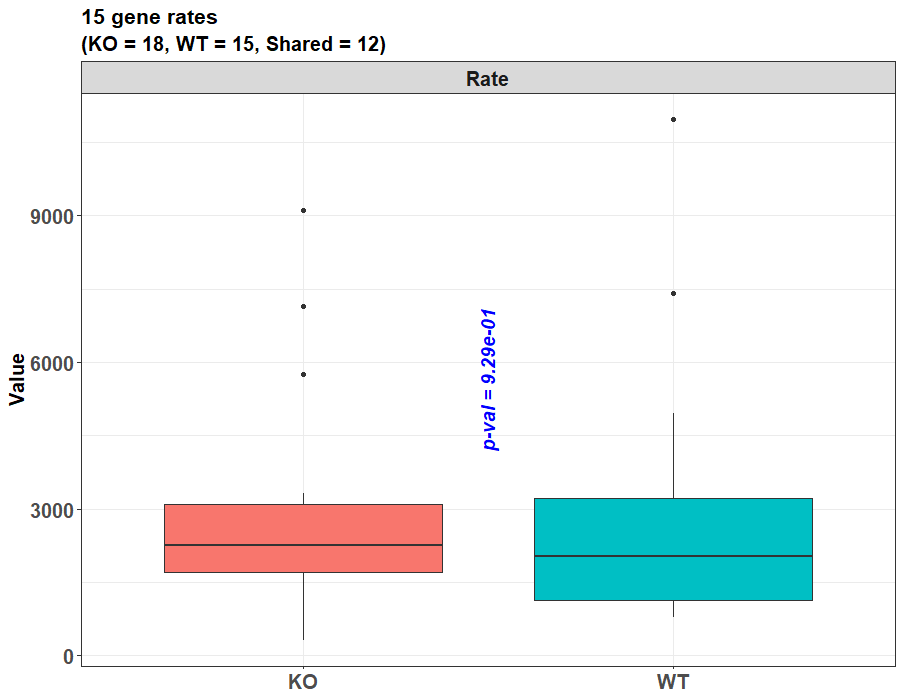
The vertical lines in the plots indicate the inferred transition bin and base pair, and the plots’ titles show their coordinate on the bin and base levels.

Similarly, KO cells’ gene transcription rates can be calculated via calrate.

korates <- calrate(time1file = ko0file,   
 time2file = ko15file,   
 time = 15,   
 strandmethod = 1,   
   
 genomename = "mm10",   
 lencutoff = 40000,   
 fpkmcutoff = 1,   
   
 threads = 2,   
   
 startshorten = 1000,   
 endshorten = 1000,   
 window\_num = 40,   
   
 method = "LSS",   
 pythonpath = NULL,   
   
 difftype = 1)

Then, the genes with a significant transition point can be extracted from the WT and KO cells’ results, respectively, and used to compare the WT and KO cells’ rates. The plot below shows that the KO ones without the Paf1C transcriptional factor have smaller rates than the WT ones. Although the p-value is insignificant given the small gene number, we can still get the trend, which demonstrates that Paf1C knockout reduces genes’ transcription rates, consistent with the original study’s conclusion *[6]*.

interrates <- function(ratesreports,   
 conditions,   
 titlesize,   
 textsize,   
 face = "bold"){  
   
 ratesdat <- do.call(rbind, ratesreports)  
   
 ratesdat$condition <- rep(conditions, c(nrow(ratesreports[[1]]), nrow(ratesreports[[2]])))  
   
 keptcols <- c("gene\_id", "rate", "condition")  
   
 ratesub <- ratesdat[keptcols]  
   
 row.names(ratesub) <- 1:nrow(ratesub)  
   
 ratesub <- ratesub[!is.na(ratesub$rate), , drop = FALSE]  
   
 ratesub <- ratesub[order(ratesub$rate), , drop = FALSE]  
   
 row.names(ratesub) <- 1:nrow(ratesub)  
   
 pval <- wilcox.test(subset(ratesub, condition == conditions[1])$rate, subset(ratesub, condition == conditions[2])$rate)$p.val  
   
 names(ratesub) <- c("gene\_id", "Value", "Condition")  
   
 ratesub$Metric <- "Rate"  
   
 annotext <- data.frame(Metric = unique(ratesub$Metric),   
 value = pval,   
 x = 1.5,   
 y = min(ratesub$Value) + (max(ratesub$Value) - min(ratesub$Value))/2,   
 color = "red",   
 stringsAsFactors = FALSE)  
   
 annotext$color[as.numeric(annotext$value) >= 0.05] <- "blue"  
   
 titleprefix <- paste(unique(ratesdat$Condition), collapse = "\_")  
   
 titlestr <- paste0(titleprefix, unique(ratesdat$time), " gene rates")  
   
 subtitlestr <- paste0("(",   
 paste(paste0(unique(ratesub$Condition), " = ", as.character(table(ratesub$Condition)[unique(ratesub$Condition)])), collapse = ", "),   
 ", Shared = ",   
 length(intersect(subset(ratesub, Condition == conditions[1])$gene\_id, subset(ratesub, Condition == conditions[2])$gene\_id)),   
 ")")  
   
 p <- ggplot2::ggplot(ratesub)  
   
 p <- p + ggplot2::geom\_boxplot(ggplot2::aes(x = Condition, y = Value, fill = Condition)) +   
 ggplot2::ylab("Value") +   
 ggplot2::ggtitle(label = titlestr, subtitle = subtitlestr) +   
 ggplot2::scale\_fill\_discrete(scales::hue\_pal()(length(unique(ratesub$Condition))), guide = FALSE) +   
 ggplot2::facet\_wrap(ggplot2::vars(Metric), scales = "free") +   
 ggplot2::theme\_bw() +   
 ggplot2::theme(  
 plot.title = ggplot2::element\_text(size = titlesize, face = face),   
 plot.subtitle = ggplot2::element\_text(size = textsize, face = face),   
 axis.title.x = ggplot2::element\_blank(),   
 axis.title = ggplot2::element\_text(size = textsize, face = face),   
 axis.text = ggplot2::element\_text(size = textsize, face = face),   
 legend.title = ggplot2::element\_text(size = textsize, face = face),   
 legend.text = ggplot2::element\_text(size = textsize, face = face),   
 strip.text = ggplot2::element\_text(size = textsize, face = face)) +  
 ggplot2::geom\_text(data = annotext, ggplot2::aes(x = x, y = y), label = paste0('bolditalic("p-val = ', formatC(annotext$value, format = 'e', digits = 2), '")'), color = annotext$color, size = floor(textsize/2.75), angle = 90, parse = TRUE)  
   
 print(p)  
   
 cat(paste0('WT mean = ',   
 round(mean(subset(ratesub, Condition == 'WT')$Value), 3), "\n"))  
   
 cat(paste0('KO mean = ',   
 round(mean(subset(ratesub, Condition == 'KO')$Value), 3), "\n"))  
   
 cat(paste('Wilcox p-val = ', round(pval, 3), '\n'))  
   
}  
  
interrates(ratesreports = list(subset(wtrates$report, significance == "significant"), subset(korates$report, significance == "significant")),   
 conditions = c("WT", "KO"),   
 titlesize = 16,   
 textsize = 15,   
 face = "bold")  
#> WT mean = 3023.484  
#> KO mean = 2859.515  
#> Wilcox p-val = 0.929



## Compare between the LSS and the HMM methods

As mentioned above, the function calrate can infer the gene transcription rates not only with the LSS method but also with the traditional HMM method. To perform the latter, the parameter method should be set as “HMM”, rather than “LSS”. At the same time, another parameter, pythonpath, should be set. It is because the conduction of HMM depends on the communication with some *Python* functions, and the pythonpath parameter is the directory of the *Python* interpreter to be used. In addition, two *Python* modules, numpy and hmmlearn, should be installed there.

If the *Python* interpreter is installed with *Anaconda*, its path can be found within RStudio. After opening the RStudio interface, select its “Tools” menu list, and then select the “Global Options…” button. Next, select the “Python” button on the left panel, and finally, tap the “Select…” button on the right panel. As a result, the candidate *Python* interpreter paths will be shown in the “Conda Environments” card on the right panel. Select the one that will be used for the HMM method, and run the following command lines in RStudio, so that this path can be stored by the reticulate R package. For example, if the last candidate *Python* interpreter shown in the card is selected in RStudio, the command lines to find it with reticulate is like this.

library(reticulate)  
  
pythonpaths <- conda\_list()  
pythonpath <- pythonpaths$python[nrow(pythonpaths)]

Then, the above variable pythonpath can be transferred to the parameter pythonpath, which indicates the *Python* interpreter to be used. Another parameter method can be set as “HMM” to start calrate’s HMM inference function. For example, the transcription rates of the wt15 sample can be inferred with HMM using the command below. Compared with the LSS inference before, only the parameters method and pythonpath need to be changed.

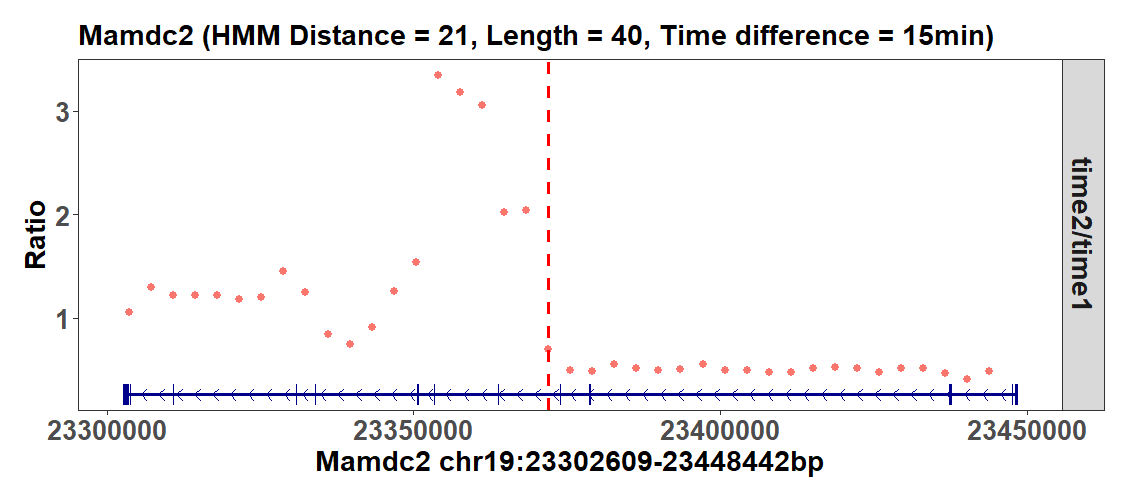
wtrates.hmm <- calrate(time1file = wt0file,   
 time2file = wt15file,   
 time = 15,   
 strandmethod = 1,   
   
 genomename = "mm10",   
 lencutoff = 40000,   
 fpkmcutoff = 1,   
   
 threads = 1,   
   
 startshorten = 1000,   
 endshorten = 1000,   
 window\_num = 40,   
   
 method = "HMM",   
 pythonpath = pythonpath,   
   
 difftype = 1)

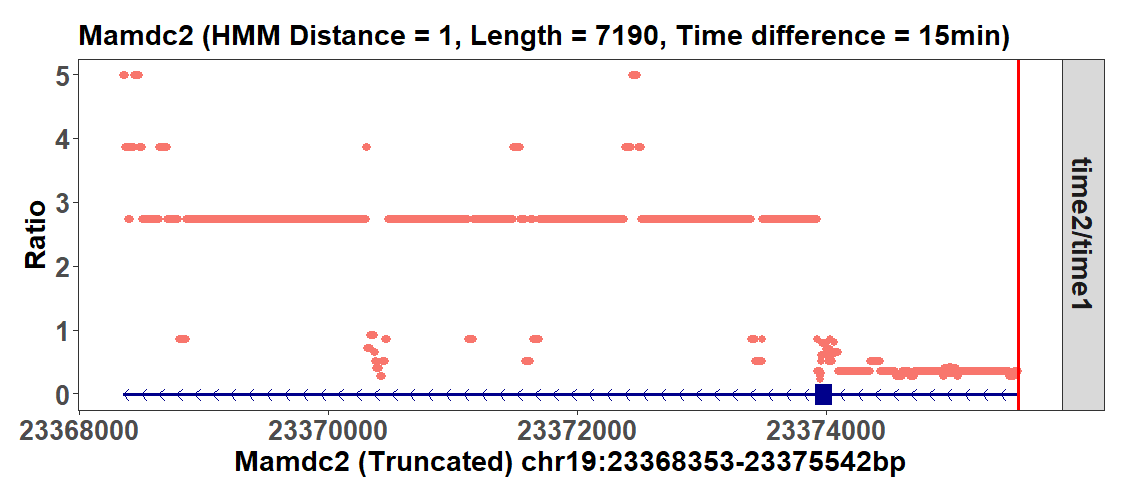
The result wtrates.hmm is also a list, containing the “report” data frame. And the transcription distances and rates inferred by HMM can be found here.

head(wtrates.hmm$report)  
#> gene\_id distance time rate significance binpadj binpval  
#> 1 Mamdc2 72901 15 4860.067 significant 3.351553e-10 1.523433e-11  
#> 2 Ppp6r3 33671 15 2244.733 significant 3.937816e-09 3.579833e-10  
#> 3 Mark2 54497 15 3633.133 significant 7.866881e-07 1.072756e-07  
#> 4 Ostf1 27620 15 1841.333 significant 3.280890e-05 5.965255e-06  
#> 5 Ptar1 29513 15 1967.533 significant 4.618113e-05 1.049571e-05  
#> 6 Cemip2 31040 15 2069.333 significant 2.784713e-04 8.860452e-05  
#> frontbinratio latterbinratio diffbinratio chr start end strand  
#> 1 0.5046868 1.579635 1.0749482 chr19 23302609 23448442 -  
#> 2 0.6014897 1.313899 0.7124097 chr19 3454928 3575749 -  
#> 3 0.8828105 1.648096 0.7652853 chr19 7275396 7341860 -  
#> 4 0.7232864 1.929708 1.2064215 chr19 18516137 18631823 -  
#> 5 0.8391100 1.968977 1.1298671 chr19 23687429 23731668 +  
#> 6 0.7282955 1.748306 1.0200101 chr19 21778342 21858360 +  
#> extendpadj extendpval frontextendratio latterextendratio  
#> 1 3.902620e-01 1.084061e-01 0.3569581 2.1565571  
#> 2 5.455118e-01 3.030621e-01 0.6185415 0.7365628  
#> 3 0.000000e+00 0.000000e+00 0.7357869 1.4219637  
#> 4 0.000000e+00 0.000000e+00 0.7723933 1.9553504  
#> 5 4.699205e-01 1.827469e-01 0.5847388 3.2371199  
#> 6 3.655874e-280 8.124164e-281 0.9012480 5.5374018  
#> diffextendratio genewidth GC exon fpkm  
#> 1 1.7995990 145834 0.4232621 0.02694845 5.043759  
#> 2 0.1180213 120822 0.4277036 0.07761831 52.810645  
#> 3 0.6861768 66465 0.4751674 0.11949146 75.033895  
#> 4 1.1829571 115687 0.4313795 0.04417091 14.822840  
#> 5 2.6523811 44240 0.4089060 0.28955696 24.338632  
#> 6 4.6361538 80019 0.4390207 0.10647471 68.830831

This result can be compared with the previous one generated by LSS. From their “report” data frames, it can be seen that HMM and LSS return different distance results for many genes. This is because they identify different transition points between the Pol II depleted and occupied states. For example, for the gene Mamdc2, LSS returns its distance as 74514 bp, as shown before, which means it identifies this point as the transition point. On the other hand, HMM returns Mamdc2’s distance as 72901 bp, and this transition point is different from the LSS one. The bin-level plot drawn by addtrack shows that HMM’s transition point is located in the same bin as the LSS one because their bin-level plots are the same. However, on the single-base level, HMM’s point is different from the LSS one, as shown by the HMM plots below and the LSS plots before. It can be seen directly that LSS’s single-base-level point is more reasonable than HMM’s.

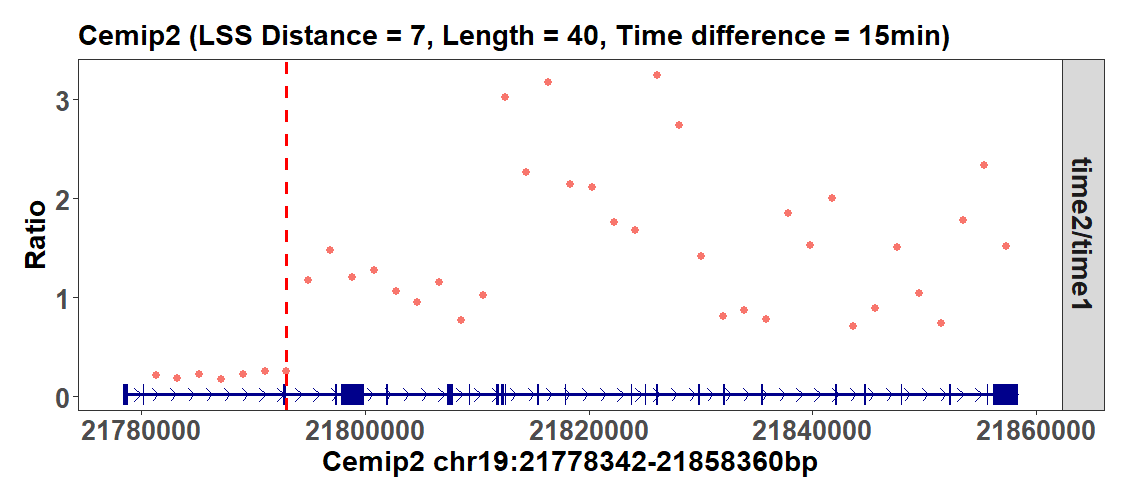
addtrack(genedat = subset(wtrates.hmm$report, gene\_id == "Mamdc2"),   
 binplotdat = wtrates.hmm$binplots$Mamdc2,   
 expandplotdat = wtrates.hmm$expandplots$Mamdc2,   
 genomename = "mm10",   
 method = "HMM",   
 titlesize = 17,   
 textsize = 16,   
 face = "bold")

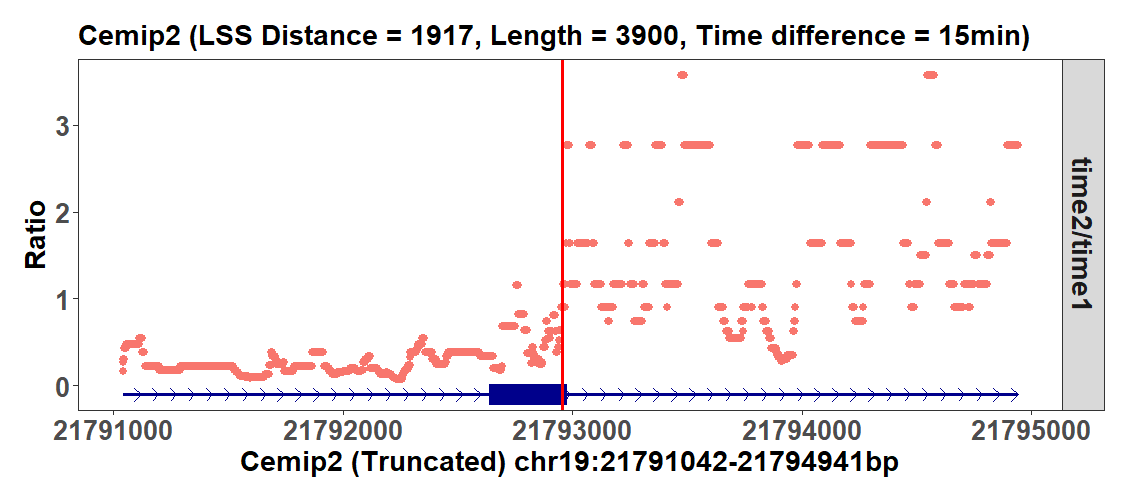




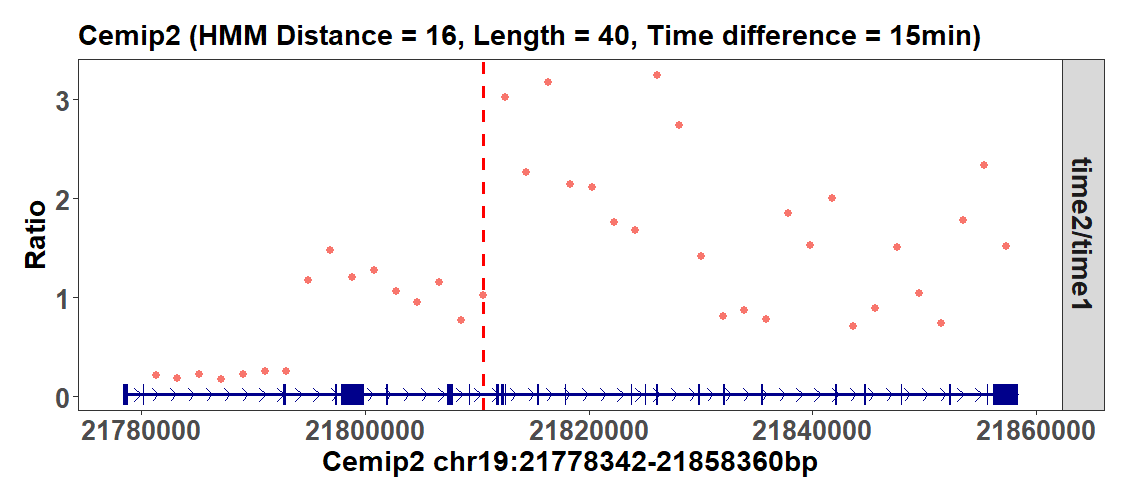
Another example is the gene Cemip2. LSS and HMM also return different distances and transition points, which can be seen from addtrack’s plots below. LSS’s point is more reasonable than HMM’s.

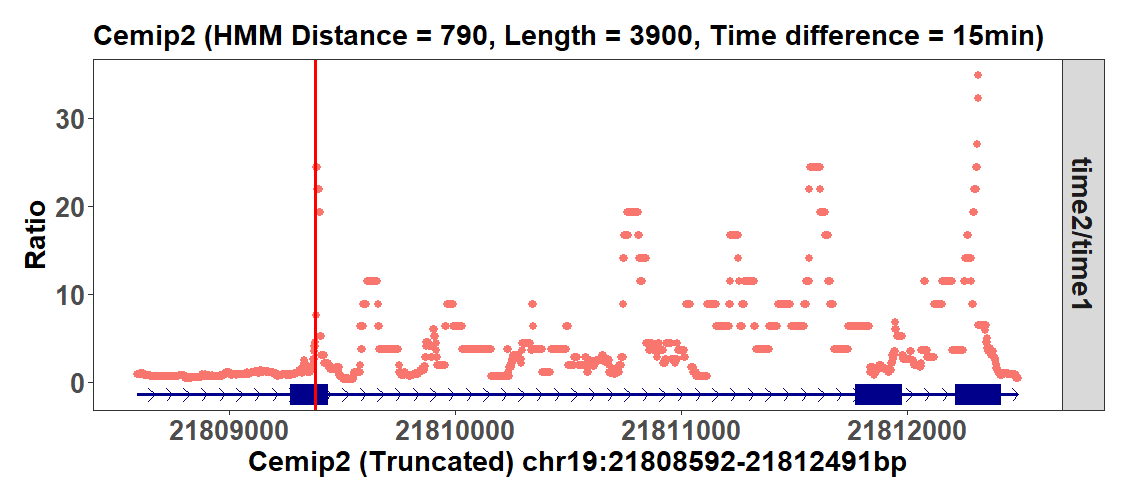
#LSS results  
addtrack(genedat = subset(wtrates$report, gene\_id == "Cemip2"),   
 binplotdat = wtrates$binplots$Mamdc2,   
 expandplotdat = wtrates$expandplots$Mamdc2,   
 genomename = "mm10",   
 method = "LSS",   
 titlesize = 17,   
 textsize = 16,   
 face = "bold")





#HMM results  
addtrack(genedat = subset(wtrates.hmm$report, gene\_id == "Cemip2"),   
 binplotdat = wtrates.hmm$binplots$Mamdc2,   
 expandplotdat = wtrates.hmm$expandplots$Mamdc2,   
 genomename = "mm10",   
 method = "HMM",   
 titlesize = 17,   
 textsize = 16,   
 face = "bold")





## Metagene plotting

Metagene plotting is always needed for transcriptional dynamic studies to check the gene reads distribution on a metagene level. Hence, *proRate* contains the functions metaplot and mmetaplot to do this. To include the metagene curves of multiple experimental conditions, such as the wt0 and ko0 groups, into one plot, mmetaplot should be used.

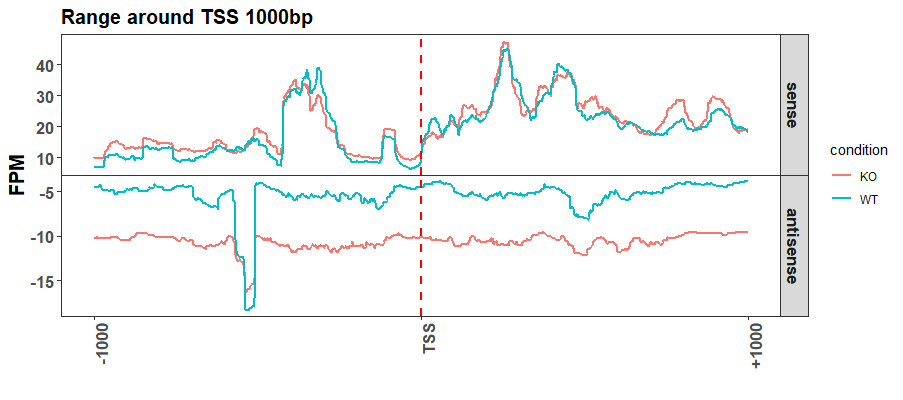
For its parameter metafiles, the bam file directories of different experimental conditions should be transferred as a vector, so here, set it as c(wt0file, ko0file). Correspondingly, the parameter labels should be set as c(“WT”, “KO”) so that the final metagene curves can be labeled with these conditions in the plot.

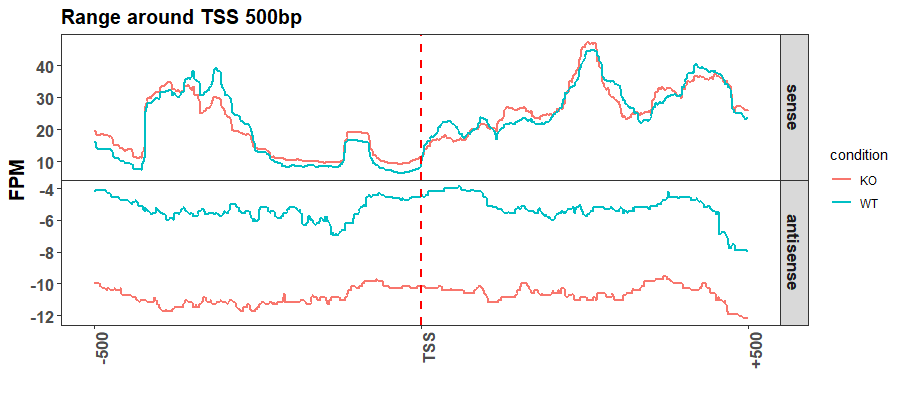
The final metagene plot can focus on different gene regions. To focus on the promoter, the parameter tssradius can be set using a single numeric value or a numeric vector. For example, if set with the vector c(1000, 500), then two metagene plots will be generated. One covers the promoter region from 1000 bp upstream to 1000 bp downstream of the TSS point; the other covers the promoter from 500 bp upstream to 500 bp downstream of TSS. The parameter ttsradius can also be set in this way to focus on the gene tail region.

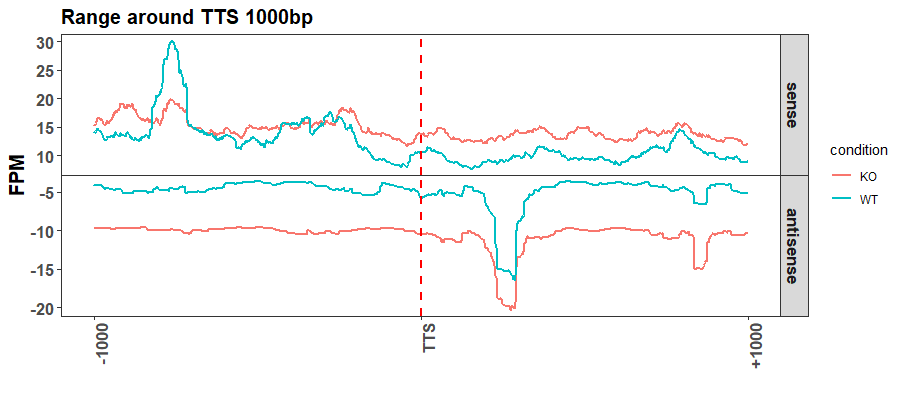
If need to generate a metagene plot on the whole gene body, the parameter genebodylen should be given a numeric value, for example, 2000. Because the length of the whole gene body varies largely for different genes, mmetaplot will first unify them according to the value of genebodylen. If a gene has a length greater than the defined value of 2000 bp here, its gene body will be compressed, while if a gene is shorter than 2000 bp, it will be extended, so finally, all the genes will be scaled to 2000 bp, and the metagene plot can be generated. The command is shown below.

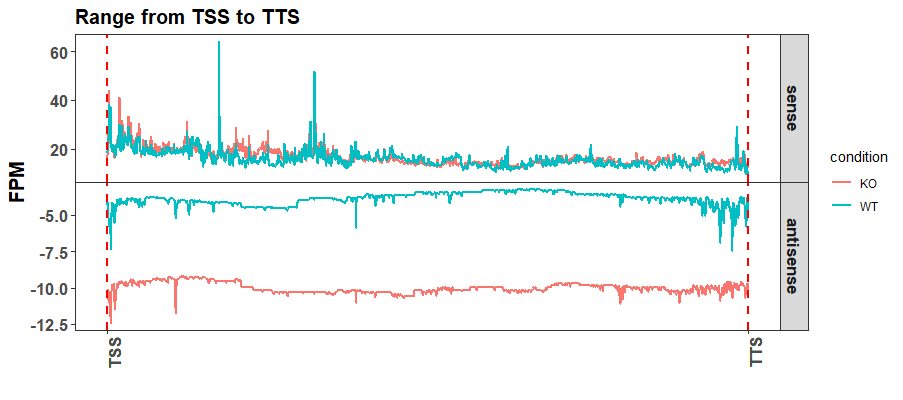
metareslist <- mmetaplot(metafiles = c(wt0file, ko0file),   
 labels = c("WT", "KO"),   
 tssradius = c(1000, 500),   
 ttsradius = c(1000),   
 genebodylen = 2000,   
 strandmethod = 1,   
 genomename = "mm10",   
 genelencutoff = 40000,   
 fpkmcutoff = 1)

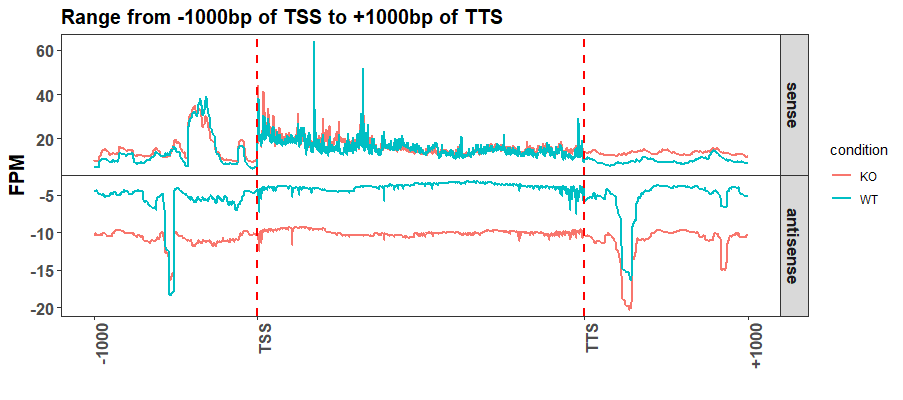
As shown below, several metagene plots can be returned, focusing on the various gene regions defined.











The list result metareslist records the FPKM values for various gene regions. For example, its slot TSSFPKMstatses contains a data frame with each row representing a gene and each column representing a promoter region on either the transcriptional forward (sense) or reverse (antisense) strand. The corresponding FPKM values are recorded in this data frame.

head(metareslist$TSSFPKMstatses)  
#> gene\_id TSS1000\_fwdfpkm.x.WT TSS1000\_revfpkm.x.WT TSS500\_fwdfpkm.x.WT  
#> 1 Ahnak 231582.3497 8994.997 290781.0250  
#> 2 Atl3 9479.5780 18315.676 11334.0699  
#> 3 Cemip2 25295.8433 2100.310 24560.8366  
#> 4 Chka 26606.1188 10339.816 27373.3487  
#> 5 Cpt1a 192.5574 4431.033 150.3544  
#> 6 Dagla 10409.1654 2321.861 7225.8565  
#> TSS500\_revfpkm.x.WT TSS1000\_fwdfpkm.y.KO TSS1000\_revfpkm.y.KO  
#> 1 10022.997 175919.49 6374.969  
#> 2 20174.494 22388.74 16492.841  
#> 3 3917.033 23051.50 10136.386  
#> 4 10656.635 52228.07 26641.570  
#> 5 4431.033 6260.71 9497.655  
#> 6 2428.206 15015.22 12916.564  
#> TSS500\_fwdfpkm.y.KO TSS500\_revfpkm.y.KO  
#> 1 238008.31 8096.766  
#> 2 19099.05 16588.497  
#> 3 21119.20 10701.061  
#> 4 35007.74 28597.877  
#> 5 6331.45 9256.974  
#> 6 12539.72 12762.281

The exact FPM values shown in the metagene plots are also included in the metareslist. For example, metareslist$WT$TSSfwdmeans is a vector with a length of 2001, covering the FPM values from 1000 bp upstream of TSS, then to the single TSS point, and finally to 1000 bp downstream of TSS, on the forward (sense) strand, in the “WT” condition.

head(metareslist$WT$TSSfwdFPMmeans)  
#> [1] 7.089653 7.089653 7.089653 6.978877 6.978877 6.978877

The FPM values recorded in these slots can be further transferred to another function, plotprocessing, so that the extremely large FPM outliers in the original metagene plots can be removed.

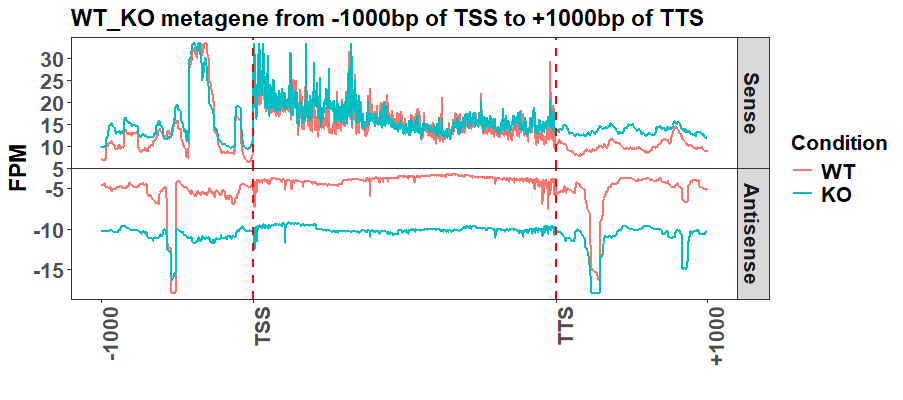
For example, plotprocessing can remove the outlier FPMs in the complete metagene that starts from the gene promoter and ends at the gene tail region. To do this, the FPM values in the original metareslist should be extracted as follows so that the new list combinefwdlist records the forward strand FPM values for the two conditions “WT” and “KO”, and the other combinerevlist records the reverse strand ones.

combinefwdlist <- list()  
  
combinerevlist <- list()  
  
for(i in seq(1, 2, 1)){  
   
 groupname <- c("WT", "KO")[i]  
   
 combinefwdlist[[i]] <- metareslist[[groupname]]$combinefwdFPMmeans  
   
 combinerevlist[[i]] <- metareslist[[groupname]]$combinerevFPMmeans  
   
}

Then, these two lists can be transferred to plotprocessing to remove their FPM outliers. If want to remove the FPM values greater than their 99% quantile, the parameter cutoff can be set as 0.01, indicating the cutoff is the 1 – 1% (0.01) = 99% quantile of the FPMs, and any larger values will be reduced to it. The parameter groupnames should be set as c(“WT”, “KO”), meaning the first elements in combinefwdlist and combinerevlist are from the “WT” condition, and the second ones are from the “KO” condition. The parameter labels is the four points, including the start, middle, and endpoints, that need to be indicated by vertical lines in the final plot. The transferred vector c(“-1000”, “TSS”, “TTS”, “+1000”) means the start point of the plot will start from the 1000 bp position upstream of the TSS point, and end in the 1000 bp position downstream of the TTS point. For the two middle points, i.e., “TSS” and “TTS”, their coordinates in the metagene should be further indicated by the parameter lineposes. When it is c(1001, 3000), it means that if we see the start point of the metagene as 1, then the “TSS” point should have a coordinate of 1001, and the “TTS” point should have a coordinate of 3000.

plotprocessing(fwdlist = combinefwdlist,   
 revlist = combinerevlist,   
   
 cutoff = 0.01,   
 groupnames = c("WT", "KO"),   
 labels = c("-1000", "TSS", "TTS", "+1000"),   
 lineposes = c(1001, 3000),   
   
 title = "WT\_KO metagene from -1000bp of TSS to +1000bp of TTS",   
 titlesize = 17,   
 textsize = 16)

Then, the function will return the new metagene plot with the extremely large FPM values removed.



## References

1. Lei EP, Krebber H, Silver PA. Messenger RNAs are recruited for nuclear export during transcription, Genes & Development 2001;15:1771-1782.
2. Bentley DL. Coupling mRNA processing with transcription in time and space, Nature Reviews Genetics 2014;15:163-175.
3. Wallace EWJ, Beggs JD. Extremely fast and incredibly close: cotranscriptional splicing in budding yeast, RNA 2017;23:601-610.
4. Muniz L, Nicolas E, Trouche D. RNA polymerase II speed: a key player in controlling and adapting transcriptome composition, The EMBO Journal 2021;40:e105740.
5. Debes C, Papadakis A, Gronke S et al. Ageing-associated changes in transcriptional elongation influence longevity, Nature 2023;616:814-821.
6. Hou L, Wang Y, Liu Y et al. Paf1C regulates RNA polymerase II progression by modulating elongation rate, Proceedings of the National Academy of Sciences 2019;116:14583-14592.
7. Fuchs G, Voichek Y, Benjamin S et al. 4sUDRB-seq: measuring genomewide transcriptional elongation rates and initiation frequencies within cells, Genome Biology 2014;15:R69.
8. Rabiner LR. A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE 1989;77:257-286.