bio-protocol

Open Access

Published: Jun 05, 2023

Polysome Profiling in Adult Mouse Testes

Jun-Yan Kang[1, #], Ai Zhong[1, #], Ze Wen[1, #], Xinghai Yu[2, #], Yu Zhou[2], and Mo-Fang Liu[1, *]

[1]State Key Laboratory of Molecular Biology, State Key Laboratory of Cell Biology, Shanghai Key Laboratory of Molecular Andrology, Shanghai Institute of Biochemistry and Cell Biology, Center for Excellence in Molecular Cell Science, Chinese Academy of Sciences; University of Chinese Academy of Sciences, Shanghai, China
[2]State Key Laboratory of Virology, Modern Virology Research Center, College of Life Sciences, Frontier Science Center for Immunology and Metabolism, Wuhan University, Wuhan, China
*For correspondence: mfliu@sibcb.ac.cn
[#]Contributed equally to this work

Abstract

Polysome profiling is widely used to isolate and analyze polysome fractions, which consist of actively translating mRNAs and ribosomes. Compared to ribosome profiling and translating ribosome affinity purification, polysome profiling is simpler and less time consuming in sample preparation and library constructions. Spermiogenesis, i.e., the post-meiotic phase of male germ cell development, is a highly coordinated developmental process in which transcription and translation are decoupled because of nuclear condensation, resulting in translation regulation as the major mode for the regulation of gene expression in post-meiotic spermatids. To understand the translation regulation during spermiogenesis, an overview of translational state of spermiogenic mRNAs is required. Here, we describe a protocol to identify translating mRNAs using polysome profiling. Briefly, mouse testes are gently homogenized to release polysomes containing translating mRNAs, following polysome-bound mRNAs isolated by sucrose density gradient purification and characterized by RNA-seq. This protocol allows to quickly isolate translating mRNAs from testes and analyze the discrepancy of translational efficiency in mouse testes from different mouse lines.

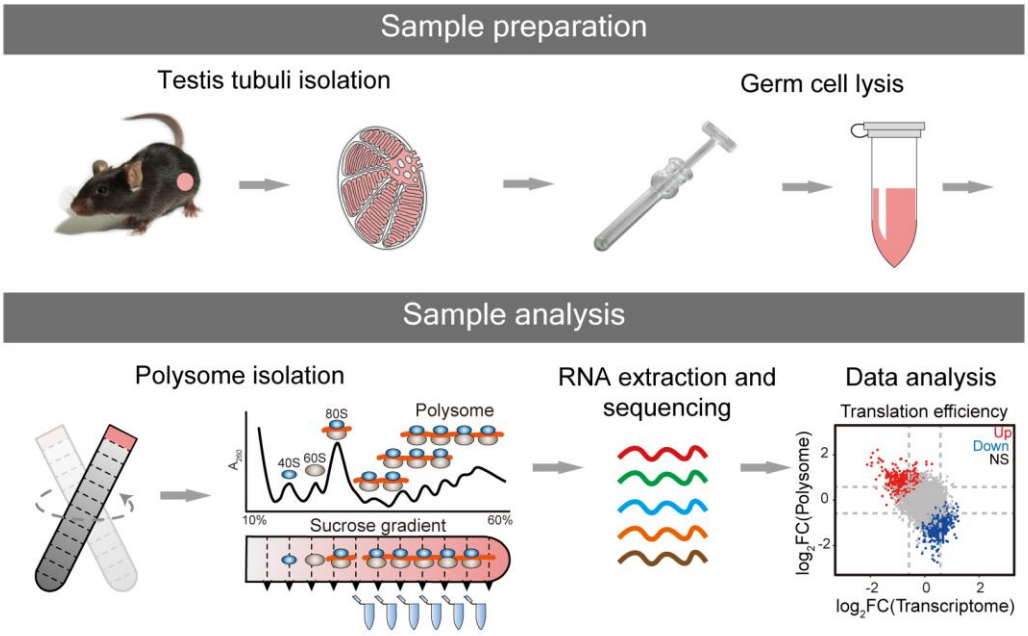
Key features

- Quickly obtain polysome RNAs from testes.
- Omit RNase digestion and RNA recovery from gel.
- High efficiency and robustness compared to ribo-seq.

Keywords: Polysome profiling, RNA extraction, mRNA translation, Translation efficiency, Mouse testes, Spermatid, Germ cell, Testicular lysate preparation, Sucrose density gradient purification

This protocol was validated in: Science (2022), DOI: 10.1126/science.abj6647

Graphical overview



Schematic illustrating the experimental design for polysome profiling in mouse testes. Mouse testes are homogenized and lysed in *Sample preparation*, and polysome RNAs are enriched by sucrose gradient centrifugation and used to calculate translation efficiency in *Sample analysis*.

Background

Gene expression is regulated at multiple levels, including epigenetic modifications, transcription, splicing, translation, etc. Among these, translation is the most important step in determining the level of protein expression (Schwanhäusser et al., 2011). An in-depth study of the translation process may connect the transcriptome with the proteome, allowing us to understand the regulatory mechanisms underlying the dynamic gene expression. Polysome profiling is a highly repeatable method used to identify mRNAs associated with polysomes, which can be isolated by subjecting cytosolic extracts to a continuous sucrose gradient (Chassé et al., 2017). Compared to polysome profiling, western blot is low throughput and should be coupled with RT-qPCR to qualitatively analyze the translation efficiency of specific genes; ribo-seq produces a precise snapshot of translatome by sequencing the RNA fragments protected by ribosomes from RNase-mediated degradation, which renders ribo-seq less efficient in identifying differences in translation efficiency. During male germ cell development, transcription and translation of spermiogenic mRNAs occur at distinct developmental stages, known as the uncoupling of transcription and translation (Steger, 1999; Sassone-Corsi, 2002). However, how those mRNAs transcribed in earlier developmental stages of spermatocytes and/or how round spermatids are translationally activated in late spermatids remain open questions. In our recently published work (Kang et al., 2022), we identified FXR1 as a potential translation activator of spermiogenic mRNAs. To verify the necessity of FXR1 for translation, we need to compare the translation efficiency in control and *Fxr1* knockout mouse testes. To this end, we established the protocol for polysome profiling in adult mouse testes based on the method previously described (Ingolia et al., 2012).

Materials and reagents

1. 1.5 mL microcentrifuge tube
2. 2 mL Eppendorf tubes
3. 20 mL syringe
4. C57BL/6N male mice
5. Cycloheximide (CHX) (MCE, catalog number: HY-12320)
6. cOmplete™ EDTA-free protease inhibitor cocktail tablets (Roche, catalog number: 04693132001)
7. SUPERase•In™ RNase inhibitor (Invitrogen, catalog number: AM2694)
8. Triton X-100 (EMD Millipore, catalog number: 648466)
9. Sucrose, RNase- & DNase-free (AMRESCO, catalog number: 0335)
10. RNAiso Plus (Takara, catalog number: 9109)
11. Tris-HCl (1 M, pH 7.0, RNase-free) (Invitrogen, catalog number: AM9851)
12. NaCl (5 M) (Invitrogen, catalog number: AM9760G)
13. $MgCl_2$ (1 M) (Invitrogen, catalog number: AM9530G)
14. Chloroform (Sinopharm Chemical Reagent, catalog number: 10006818)
15. Isopropanol (Sinopharm Chemical Reagent, catalog number: 80109218)
16. Ethanol (Sinopharm Chemical Reagent, catalog number: 10009218)
17. Polysome buffer (see Recipes)
18. Lysis buffer (see Recipes)
19. 60% sucrose grading solution (see Recipes)
20. 10% sucrose grading solution (see Recipes)
21. 75% ethanol (see Recipes)

Equipment

1. Ultra-clear centrifuge tube, 14 × 89 mm (Beckman Coulter, catalog number: Z90805SCA)
2. Biocomp Gradient Maker and Fractionator (Biocomp)
3. Beckman Coulter Optima L-80 XP ultracentrifuge (with rotor SW41 Ti) (Beckman Coulter)
4. Dounce homogenizer, 2 mL (Sangon, catalog number: F519062)
5. Tweezers (Sangon Biotech, catalog number: F519022)
6. Scissor (Sangon Biotech, catalog number: F519234)
7. 100 mm cell culture dish, DNase- & RNase-free (WHB-100)

Software

We perform all sequencing data analyses under the Linux system. We use the conda program to create the environment for polysome profiling data analysis and build computational pipelines using the snakemake program with the following software:

1. Sratoolkit (https://github.com/ncbi/sra-tools/wiki/01.-Downloading-SRA-Toolkit)
2. Fastqc (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/)
3. STAR (https://github.com/alexdobin/STAR)
4. Samtools (https://github.com/samtools/samtools)
5. RSeQC (https://rseqc.sourceforge.net/index.html)
6. R (https://www.r-project.org/)
7. Rsubread (https://bioconductor.org/packages/release/bioc/html/Rsubread.html)
8. DESeq2 (https://bioconductor.org/packages/release/bioc/html/DESeq2.html)
9. Xtail (https://github.com/xryanglab/xtail)

Databases

The genome sequences and gene annotation can be downloaded from the Genecode database (mm10 version 23), and tRNA and rRNA sequences can be downloaded using the UCSC Table Browser.

1. Genecode (https://www.gencodegenes.org/)
2. UCSC Genome Browser (https://genome.ucsc.edu/)

Procedure

A. Prepare mouse testicular lysate

1. C57BL/6N male mice were sacrificed by cervical dislocation and the testes were collected after abdominal anatomy (see also Note 4).
2. Use tweezers to peel off the tunica albuginea covering the surface of the testis.

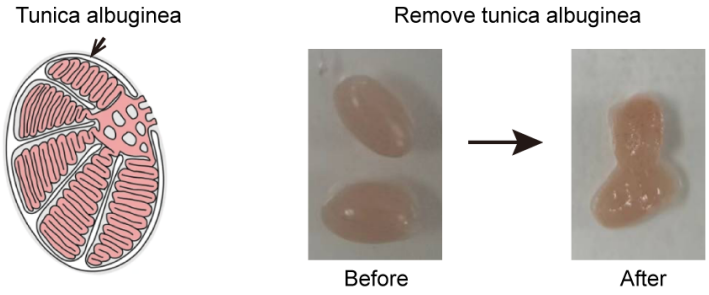


Figure 1. Schematic diagram illustrating the tunica albuginea's location and shape and remaining testicular tissues

3. Weigh the testicular tissues and transfer them to a pre-cooled Dounce homogenizer (see Note 4).
4. Homogenize the testicular tissues with lysis buffer (1 mL per 100 mg of testes) by 10–15 strokes.
5. Transfer homogenate to a 1.5 mL microcentrifuge tube and incubate on ice for 15 min.
6. Centrifuge at 13,000× *g* for 2 min at 4 °C.
7. Transfer the supernatant to a fresh 1.5 mL microcentrifuge tube on ice.

B. Prepare sucrose gradient

1. Mark the ultra-clear centrifuge tube with the marker block provided by Biocomp and place it in the MagnaBase holder.
2. Add 10% sucrose up to the marker line with a 20 mL syringe.
3. Then, add an equal volume of 60% sucrose at the bottom of the tube with a fresh syringe.
4. Apply the short cap provided by Biocomp and remove the excess solution.
5. Turn on the Biocomp Gradient Station and level the rotating platform using a round bubble level.
6. Place the MagnaBase holder in the center of the plate.
7. Select the pre-programmed 10%–60% (w/v) gradients and press USE.
8. The station will beep when finished. Sucrose gradients should be used immediately.

C. Isolate polysome fractions from the total lysate

1. Take 10% of the lysates prepared in section A to a new 1.5 mL Eppendorf tube, add 500 µL of RNAiso Plus, and mix samples well. Label this sample as TOTAL and store it at -20 °C, or extract RNA following the procedure in section D.
2. Fill the tube prepared in section B with residual lysates.

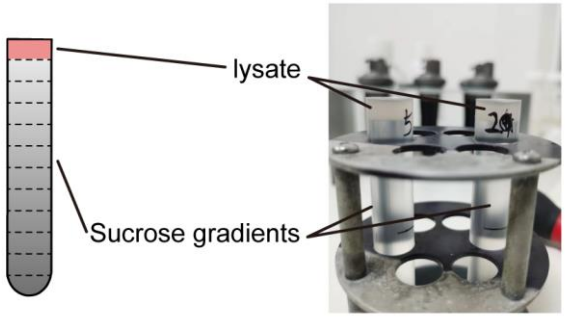


Figure 2. Loading of lysates on sucrose gradients

3. Centrifuge the gradient at 38,000× *g* for 2 h at 4 °C in an SW41 Ti rotor.
4. After ultracentrifugation, gently remove tubes from the rotor and collect 12 fractions with Piston Gradient Fractionator.
   a. Start the Gradient Station Fractionator and clean the upper valve assembly and the collection tubing.
   b. Insert the centrifuge tube into the tube holder and lock it in.
   c. Place the tube holder on the fractionator and turn it back 90° to lock it.
   d. Cut out the cap of 2 mL Eppendorf tubes and place them onto the Fraction Collector.
   e. Open the software, select the user profile, and zero the detector.
   f. After blanking, press the AIR button according to the operation manual to force air through the entire system until no liquid comes out at the end of the collection tubing.
   g. Run the fractionator and collect 12 fractions. The system will plot the absorbance in real time as the sample passes through the flow cell. Export data following software instructions.

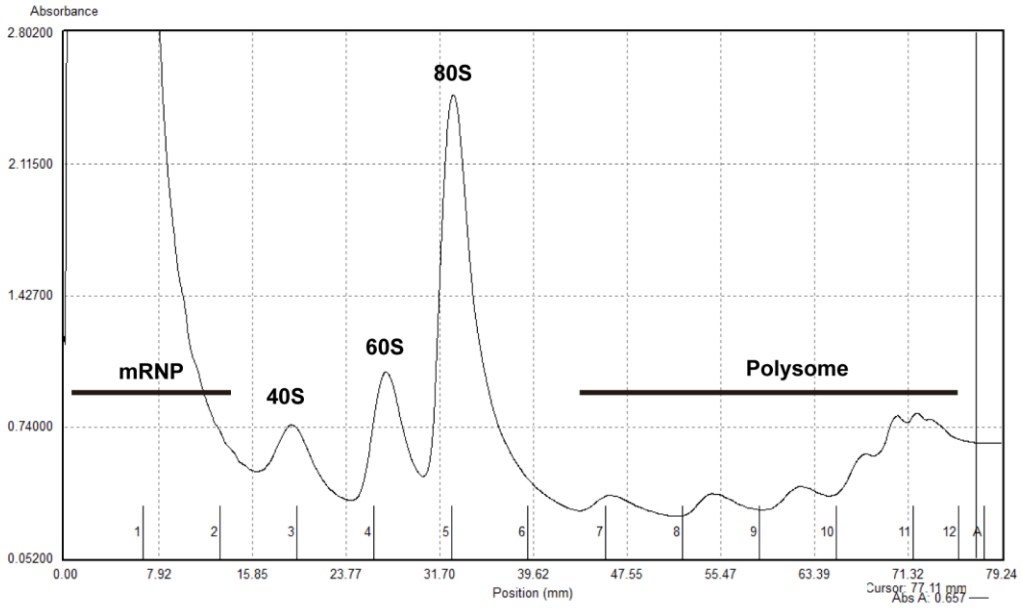


Figure 3. An absorbance graph generated by Biocomp Gradient Maker and Fractionator

5. Transfer 700 µL of each fraction to a new 1.5 mL Eppendorf tube, add 500 µL of RNAiso Plus, and mix samples well. Label these samples as FRT#1–12 and store them at -20 °C, or extract RNA following the procedure in section D.

D. Extract polysome RNAs

1. Vortex for 10 s and keep the homogenate at room temperature for 5 min.
2. Centrifuge at 13,000× *g* for 5 min at 4 °C.
3. Transfer the supernatant to a new centrifuge tube and add 100 µL of chloroform.
4. Vortex for 10–15 s and keep at room temperature for 5 min.
5. Centrifuge at 13,000× *g* for 15 min at 4 °C.
6. Transfer the upper layer to a new centrifuge tube.
7. Add one volume of isopropanol and keep it at room temperature for at least 10 min.
8. Centrifuge at 13,000× *g* for 15 min at 4 °C.
9. Remove the supernatant and wash the RNA precipitate with 1 mL of pre-cooled 75% ethanol.
10. Centrifuge at 13,000× *g* for 5 min at 4 °C.
11. Remove the supernatant and keep the RNA precipitate air dried.
12. Dissolve the RNA precipitate with an appropriate amount of preheated nuclease-free water (see Note 5).

E. High-throughput sequencing

1. According to the absorbance graph, combine the RNA solutions extracted from polysome fractions (usually FRT#6–12 or FRT#7–12) and label it as **POLY**.
2. Subject comparable **TOTAL** and **POLY** RNAs to high-throughput sequencing with the Illumina platform (performed by Novogene Co, Ltd). Briefly, the concentration and purity of extracted RNAs were checked using Nanodrop and agarose gel electrophoresis, and integrity was further determined using Agilent 2100. Then, the libraries were constructed using NEBNext® Ultra™ RNA Library Prep kit for Illumina®.

Data analysis

A. Make sure that conda is installed in your Linux system

B. Create a conda environment named *PPIMT* and install the snakemake and required software (sratoolkit, fastqc, STAR, samtools, rseqc, R, Rsubread, and DESeq2). Create directories named *scripts* and *snakemake* in your working directory.

C. Prepare a CSV file of sample information and the following scripts for downstream analysis (Table 1)

Table 1. Sample list

raw	samples	rep	cond	contrast
SRR14139008	Hetero_polysome_rep2	rep2	Hetero_polysome	None
SRR14139009	Hetero_polysome_rep1	rep1	Hetero_polysome	None
SRR14139016	cKO_polysome_rep2	rep2	cKO_polysome	Hetero_polysome
SRR14139017	cKO_polysome_rep1	rep1	cKO_polysome	Hetero_polysome
SRR14139014	cKO_total_rep2	rep2	cKO_total	Hetero_total

SRR14139015	cKO_total_rep1	rep1	cKO_total	Hetero_total
SRR14139018	Hetero_total_rep2	rep2	Hetero_total	None
SRR14139019	Hetero_total_rep1	rep1	Hetero_total	None

1. Sample information: *PP_sample_info.csv*

2. Scripts:
   a. Meta.py

```
import os
import pandas as pd

## input
DATA_BASE = "../data"
GENOME_DIR = "../../anno"
G_INDEX_DIR = os.path.join(GENOME_DIR, "star.idx")
r_INDEX_DIR = os.path.join(GENOME_DIR, "star.idx_rRNA")
ANNO_DIR = GENOME_DIR
SCRIPT_DIR = "../scripts"
SAMPLE_INFO = "../PP_sample_info.csv"
REF_GTF = os.path.join(GENOME_DIR, "gencode.vM23.annotation.gtf")
REF_FA = os.path.join(GENOME_DIR, "mm10.fasta")
rRNA_FA = os.path.join(GENOME_DIR, "mm10_rRNA.fa")
SIZES = os.path.join(GENOME_DIR, "mm10.sizes")
Rscript = "Rscript"
sample_info = pd.read_csv(SAMPLE_INFO, sep=",")
samples = sample_info['samples']
cond = sample_info[sample_info['contrast']!="None"].cond.unique()

## output
RESULTS_BASE = "../results"

## RNAseq
def get_control(cond, samp_info):
    contrast = samp_info[samp_info.cond == cond].contrast.unique()
    contrast = "".join(list(contrast))
    samp = samp_info[samp_info['cond']==contrast].samples
    samp = list(samp)
    return(samp)

def get_samp(cond, samp_info):
    samp_info = samp_info[samp_info.cond == cond]
    samp = samp_info.samples
    samp = list(samp)
   return(samp)
```

   b. featurecount.R

```
#!/usr/bin/env Rscript
library("Rsubread")
args <- commandArgs(T)

## input
bam <- args[1]
```

```
gtf <- args[2]
strandspecf <- args[3]
name <- gsub("_uniq_sort.bam", "", basename(bam))
base <- dirname(dirname(dirname(bam)))
Nthread <- 8

## output
DESdir <- paste(base, "featurecount", sep = "/")
txt_feacount <- paste(DESdir, paste(name, "featureCounts.txt", sep
= "_"), sep = "/")
log_feacount <- paste(DESdir, paste(name, "featureStat.log", sep =
"_"), sep = "/")

##run
fc_SE <-featureCounts(
   bam,
   annot.ext = gtf,
   isGTFAnnotationFile = T,
   isPairedEnd = TRUE,
   strandSpecific = as.integer(strandspecf),
   nthreads = Nthread
  )
write.table(fc_SE$counts,txt_feacount, quote = F,col.names = F,sep
= "\t")
write.table(fc_SE$stat,log_feacount, quote = F,col.names = F,sep =
"\t")
```

c. Distance.R

```
#!/usr/bin/env Rscript
## input
RAW_DIR <- "../results/featurecount"
## output
outDir <- "../results/plot_sample_cor"
args <- commandArgs(TRUE)
if (length(args) == 2) {
   RAW_DIR <- args[1]
   outDir <- args[2]
} else if (length(args) > 0) {
 stop("
============================================================
Usage:  'Rscript  01_distance.R  [/path/to/featurecounts_files_dir]
[/out_dir]'
Or:    'Rscript 01_distance.R'  (use default parameters)
 ============================================================
", call.=FALSE)
}

library(RColorBrewer)
library(ComplexHeatmap)
library(circlize)
library(dplyr)

samples <- list.files(RAW_DIR, pattern = "_featureCounts.txt")
```

```
t_samples <- samples[grep("total", samples)]
p_samples <- samples[grep("polysome", samples)]

pdf_poly <- file.path(outDir, "All_polysome_cor.pdf")
pdf_total <- file.path(outDir, "All_total_cor.pdf")

## run
plot_cor <- function(RAW_DIR, samples, pdf_cor) {
 COLS <- rev(brewer.pal(n = 5, name = "RdYlBu"))
 samp_base <- sapply(samples, function(x) strsplit(x, split =
"_featureCounts")[[1]][1])
 samp_cond <- sapply(as.vector(samp_base), function(x) strsplit(x,
split = "_rep")[[1]][1])
 full_name <- file.path(RAW_DIR, samples)
 tsv <- lapply(full_name, function(x) read.csv(x, sep = "\t", header
= F)$V2)
 GENEID <- read.csv(full_name[1], sep = "\t", header = F)$V1
 df <- do.call(cbind, tsv)
 df <- df[apply(df, 1, sum) > 40, ]
 colnames(df) <- as.vector(samp_base)
 cor_mat <- cor(df, method = "spearman")
 print(cor_mat)
 p <- Heatmap(cor_mat, name = "Correlation", col = COLS,
show_row_names = T,
 show_column_names = T, cluster_columns = T, cluster_rows = T,
clustering_distance_rows = "spearman", clustering_distance_columns
= "spearman")
pdf(pdf_cor, width = 5, height = 4)
print(p)
dev.off()
}

plot_cor(RAW_DIR, p_samples, pdf_poly)
plot_cor(RAW_DIR, t_samples, pdf_total)
```

d. DESeq.R

```
#!/usr/bin/env Rscript
library(DESeq2)
args <- commandArgs(T)

## input
samples <- args[1]
# samples <- gsub(" ","", samples)
samples <- strsplit(samples, split = ",")[[1]]
samp_cond <- strsplit(args[2], split = ",")[[1]]

## output
tsv_DE <- args[3]

## run
run_DESeq2 <- function(samples, samp_cond, tsv_DE) {
  samples_base <- sapply(samples, basename)
  samp_base <- sapply(samples_base, function(x) strsplit(x, split =
```

```
"_featureCounts")[[1]][1])
  tsv <- lapply(samples, function(x) read.csv(x, sep = "\t", header
= F)$V2)
  GENEID <- read.csv(samples[1], sep = "\t", header = F)$V1
  df <- do.call(cbind, tsv)
  rownames(df) <- GENEID
  colnames(df) <- as.vector(samp_base)
  colData <- data.frame(condition = samp_cond)
  colData$condition <- factor(samp_cond, levels = unique(samp_cond))
  dds <- DESeqDataSetFromMatrix(
    countData = df,
    colData = colData,
    design = ~condition
  )
  dds <- DESeq(dds)
  res <- results(dds)
  print(head(res))
  # write all expression gene name to a file
  all_exp <- res[res$baseMean != 0, ]
  all_exp <- as.data.frame(all_exp)
  all_exp$name <- rownames(all_exp)

  write.table(all_exp, tsv_DE, sep = "\t", quote = F, row.names = F)
}

run_DESeq2(samples, samp_cond, tsv_DE)
```

e. polysome_volcan.R

```
library(ggplot2)
library(readr)
## function
volcan_plot <- function (deg_p, name) {
  library(ggplot2)
  cutoff <- 0.58
  PValue <- 0.05
  Ylim <- 20
  pdf_Vocan <- paste(name, "volcan.pdf", sep = "_")
  deg_p <- deg_p[!is.na(deg_p$padj),]
  deg_p$padj[(-log10(deg_p$padj) > Ylim)] <- 10 ^ (-Ylim)
  ##Construct the plot object
  # with legend

  deg_p$type  <-  ifelse((deg_p$log2FoldChange  <  (-cutoff))  &
(deg_p$padj < 0.05) , "Down",
                    ifelse((deg_p$log2FoldChange     >cutoff)     &
(deg_p$padj < 0.05), "Up", "NS"
                    ))
  label_tab <- table(deg_p$type)
  label_stat <- paste(names(label_tab), label_tab, sep = ":")
  p <- ggplot(data = deg_p,
              aes(x = log2FoldChange,y = -log10(padj),colour = type))
+
    geom_point(size = 0.2) +
```

```
    scale_color_manual(labels=label_stat, values = c("blue", "grey",
"red")) +
    geom_vline( xintercept = c(-0.58, 0.58),lty = 4,col = "grey", lwd
= 0.6 ) +
    geom_hline( yintercept = -log10(0.05),lty = 4,col = "grey",lwd =
0.6) +
    xlim(c(-6, 6)) +
    ylim(c(0, Ylim)) +
    theme_bw() +
    theme(
      legend.position = c(0.8, 0.8),
      panel.grid = element_blank(),
      legend.title = element_blank())+
    labs(x = "log2(KO/WT)", y = "-log10(FDR)")
  pdf(pdf_Vocan, width = 3.5, height = 4)
  print(p)
  dev.off()
}
## input
raw_DIR <- "../results/DESeq2"
OV <- file.path(raw_DIR, "cKO_polysome_DEG.tsv")
## output
OUTDIR <- "../results/figure"
pdf_volcan <- file.path(OUTDIR, "polysome")

##
deg_p <- read_tsv(OV)
volcan_plot(deg_p, pdf_volcan)
```

f. getExp_mRNA.R

```
#!/usr/bin/env Rscript
library(readr)

# args <- commandArgs(T)
## input
RAW_DIR <- "../results/featurecount/"
samples <- list.files(RAW_DIR, pattern = "_rpkm", full.names = T)[7:8]

## output
outDir <- "../results/figure "
tsv_mRNA <- file.path(outDir, "mRNAexp.tsv")

## run
samp_base <- sapply(samples, function(x) strsplit(basename(x), split
= "_rpkm")[[1]][1])
samp_cond <- sapply(as.vector(samp_base), function(x) strsplit(x,
split = "_rep")[[1]][1])
tsv <- lapply(samples, function(x) read.csv(x, sep = "\t", header =
F)$V2)
GENEID <- read.csv(samples[1], sep = "\t", header = F)$V1
df <- do.call(cbind, tsv)
# df <- round(t(t(df/glt$length)))
rownames(df) <- GENEID
```

```
colnames(df) <- as.vector(samp_base)
df <- df[rowSums(df) > 0, ]
df_mean <- data.frame(
    ID = rownames(df),
    rpkm_mean = apply(df, 1, mean)
)
df_mean$ID <- sapply(df_mean$ID, function(x) strsplit(x, split =
"\\.")[[1]][1])
write_tsv(df_mean, tsv_mRNA)
```

g. Xtail.R

```
## input
P_DEG <- ".. /DESeq2/cKO_polysome_DEG.tsv"
T_DEG <- ".. /DESeq2/cKO_total_DEG.tsv"
## output
outDir <- "../results/figure"
args <- commandArgs(TRUE)
if (length(args) == 3) {
  P_DEG <- args[1]
  T_DEG <- args[2]
  outDir <- args[3]
} else if (length(args) > 0) {
  stop("
=============================================================
Usage:     'Rscript     02_xtail.R     [/path/to/polysome/DEG_file]
[/path/to/total/DEG_file] [/out_dir]'
Or:    'Rscript 02_xtail.R'  (use default parameters)
=============================================================
 ", call.=FALSE)
}

library(xtail)

DEtsv <- file.path(outDir, "DE_TE.tsv")
## run
## function
get_baseNmame <- function(x) {
  strsplit(basename(x), split = "_featureCounts")[[1]][1]
}
get_mat <- function(x) {
  count <- read.csv(x, sep = "\t", header = F)$V2
  names(count) <- read.csv(x, sep = "\t", header = F)$V1
  return(count)
}
## get data matrix
countDir <- "../results/featurecount/"

p_samps  <-  list.files(countDir,  full.names  =  T,  pattern  =
"polysome_rep[12]_featureCounts.txt")
t_samps  <-  list.files(countDir,  full.names  =  T,  pattern  =
"total_rep[12]_featureCounts.txt")
p_mat <- do.call(cbind, lapply(rev(p_samps), get_mat))
colnames(p_mat) <- sapply(rev(p_samps), get_baseNmame)
```

```
t_mat <- do.call(cbind, lapply(t_samps, get_mat))
colnames(t_mat) <- sapply(t_samps, get_baseNmame)
cond <- rep(c("control", "treat"), each = 2)
test.results <- xtail(t_mat, p_mat, cond, bins = 1000)
res_table <- test.results$resultsTable
res_table$ID <- row.names(res_table)

write.table(res_table, DEtsv, sep = "\t", row.names = F, quote = F)
```

h. DE_TE_scatter.R

```
library(readr)
library(ggplot2)
library(ggrepel)
library(clusterProfiler)
library(org.Mm.eg.db)

 outDir <- "../results/figure"
## input
DEtsv <- file.path(outDir, "DE_TE.tsv")
P_DEG <- "../results/DESeq2/cKO_polysome_DEG.tsv"
T_DEG <- "../results/DESeq2/cKO_total_DEG.tsv"
## get data matrix
countDir <- "../results/featurecount/"
## output
pdf_volcan <- file.path(outDir, "DE_TE_scatter.pdf")
tsv_volcan <- file.path(outDir, "DE_TE_scatter.tsv")
## function
get_baseNmame <- function(x) {
  strsplit(basename(x), split = "_featureCounts")[[1]][1]
}
get_mat <- function(x) {
  count <- read.csv(x, sep = "\t", header = F)$V2
  names(count) <- read.csv(x, sep = "\t", header = F)$V1
  return(count)
}

###
## filter low exp genes
test.results <- read_tsv(DEtsv)

p_samps <- list.files(countDir, full.names = T, pattern =
"polysome_rep[12]_featureCounts.txt")
t_samps <- list.files(countDir, full.names = T, pattern =
"total_rep[12]_featureCounts.txt")
samps <- c(p_samps, t_samps)
fpkm_mat <- do.call(cbind, lapply(samps, get_mat))
colnames(fpkm_mat) <- sapply(samps, get_baseNmame)
ExpGenes_bool <- apply(fpkm_mat, 1, function(x) {
  all(x > 1)
})
ExpGenes <- rownames(fpkm_mat)[ExpGenes_bool]

p_DEG <- read_tsv(P_DEG)
```

```
t_DEG <- read_tsv(T_DEG)
useGenes <- intersect(
  p_DEG$name[!is.na(p_DEG$padj)],
  t_DEG$name[!is.na(t_DEG$padj)]
)
useGenes <- intersect(useGenes, ExpGenes)
deg_p <- test.results[test.results$ID %in% useGenes, ]
cutoff <- 0.58
deg_p$type <- ifelse((deg_p$log2FC_TE_final < (-cutoff)) &
(deg_p$pvalue.adjust < 0.05), "Down",
  ifelse((deg_p$log2FC_TE_final > cutoff) & (deg_p$pvalue.adjust <
0.05), "Up", "NS")
)
deg_p$Gene <- sapply(
  deg_p$ID,
  function(x) strsplit(x, split = "\\.")[[1]][1]
)
label_tab <- table(deg_p$type)
label_stat <- paste(names(label_tab), label_tab, sep = ":")
HL_lst <- c(
  "Gpd2", "Stat4",
  "Hk1", "Akap3"
)
eg <- bitr(deg_p$Gene, fromType = "ENSEMBL", toType = "SYMBOL", OrgDb
= "org.Mm.eg.db")
deg_p <- merge(deg_p, eg, by.x = "Gene", by.y = "ENSEMBL", all.x =
T)
p <- ggplot(deg_p, aes(x = mRNA_log2FC, y = RPF_log2FC, color = type))
+
  geom_point(size = 0.1) +
  xlab(expression("log2FC(Transcriptome)")) + # x-axis label
  ylab(expression("log2FC(Polysome)")) + # y-axis label
  geom_vline(xintercept = c(-cutoff, cutoff), lty = 4, col = "grey",
lwd = 0.6) +
  geom_hline(yintercept = c(-cutoff, cutoff), lty = 4, col = "grey",
lwd = 0.6) +
  scale_color_manual(labels = label_stat, values = c("blue", "grey",
"red")) +
  geom_text_repel(
    data = subset(deg_p, SYMBOL %in% HL_lst),
    aes(mRNA_log2FC, RPF_log2FC, label = SYMBOL),
    box.padding = 1.0, max.overlaps = 50,
    col = "black"
  ) +
  xlim(c(-3, 3)) +
  ylim(c(-3, 3)) +
  theme_bw() +
  theme(
    legend.position = c(0.8, 0.8),
    panel.grid = element_blank(),
    legend.title = element_blank()
  )
pdf(pdf_volcan, width = 3.5, height = 3)
```

```
print(p)
dev.off()
write.table(deg_p, tsv_volcan, sep = "\t", row.names = F, quote = F)
```

i. DE_TE_scatter_0.01.R

```
library(readr)
library(clusterProfiler)
library(ggplot2)
library(ggrepel)
## input
outDir <- "../results/figure/"
DEtsv <- file.path(outDir, "DE_TE_scatter.tsv")
## output

pdf_volcan <- file.path(outDir, "DE_TE_scatter0.01.pdf")
tsv_volcan <- file.path(outDir, "DE_TE_scatter0.01.tsv")
## run
## function

deg_p <- read_tsv(DEtsv)
deg_p <- deg_p[!duplicated(deg_p$Gene), ]
cutoff <- 0.58
pcut <- 0.01

deg_p$type1 <- ifelse((deg_p$log2FC_TE_final < (-cutoff)) &
(deg_p$pvalue.adjust < pcut), "Down",
 ifelse((deg_p$log2FC_TE_final > cutoff) & (deg_p$pvalue.adjust <
pcut), "Up", "NS"))
label_tab <- table(deg_p$type1)
label_stat <- paste(names(label_tab), label_tab, sep = ":")
HL_lst <- c( "Gpd2", "Stat4",  "Hk1", "Akap3")
p <- ggplot(deg_p, aes(x = mRNA_log2FC, y = RPF_log2FC, color =
type1)) +
 geom_point(size = 0.1) +
 xlab(expression("log2FC(Transcriptome)")) + # x-axis label
 ylab(expression("log2FC(Polysome)")) + # y-axis label
 geom_vline(xintercept = c(-cutoff, cutoff), lty = 4, col = "grey",
lwd = 0.6) +
 geom_hline(yintercept = c(-cutoff, cutoff), lty = 4, col = "grey",
lwd = 0.6) +
 scale_color_manual(labels = label_stat, values = c("blue", "grey",
"red")) +
 geom_text_repel(
   data = subset(deg_p, SYMBOL %in% HL_lst),
   aes(mRNA_log2FC, RPF_log2FC, label = SYMBOL),
   box.padding = 1.0, max.overlaps = 50,
   col = "black"
 ) +
 xlim(c(-3, 3)) +
 ylim(c(-3, 3)) +
 theme_bw() +
 theme(
   legend.position = c(0.8, 0.8),
```

```
    panel.grid = element_blank(),
    legend.title = element_blank()
  )
pdf(pdf_volcan, width = 3.5, height = 3)
print(p)
dev.off()
write_tsv(deg_p, tsv_volcan)
```

D. Prepare the references for alignment

1. Download the directory "anno" with the required Fasta files from our polysome_profiling project, including the mouse genome (mm10) (https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M23/GRCm38.p6.genome.fa.gz), tRNA and rRNA sequences (generated from https://genome.ucsc.edu/cgi-bin/hgTables), and genecode v23 gtf annotation file (https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M23/gencode.vM23.annotation.gtf.gz).
2. Build rRNA and tRNA index.

```
STAR --runThreadN 2 --runMode genomeGenerate \
--genomeDir r_INDEX_DIR \
--genomeFastaFiles mm10_rRNA.fa \
--outFileNamePrefix r_INDEX_DIR
```

3. Build genome index.

```
STAR --runThreadN 20 --runMode genomeGenerate \
--genomeDir g_INDEX_DIR \
--genomeFastaFiles mm10.fasta \
-sjdbGTFfile anno/gencode.vM23.annotation.gtf \
--outFileNamePrefix g_INDEX_DIR
```

E. Store the Fastq files of sequencing reads in a directory with the name *data*

Note: If you would like to download the data presented in our original article, use the sratoolkit program and the accession number list that you can find on our PPIMT project page.

```
mkdir data
while read line
do
name=$line
fastq-dump -O data --gzip $name
done < "SRR_Acc_List.txt"
```

F. Check the quality of raw sequencing data using fastqc as below

```
rule fastqc:
    input:
        fq1 = os.path.join(DATA_BASE, "{sample}_R1.fq.gz"),
        fq2 = os.path.join(DATA_BASE, "{sample}_R2.fq.gz"),
    output:
```

```
        fq1                    =                    os.path.join(RESULTS_BASE,
"fastqc/{sample}/{sample}_R1_fastqc.html"),
        fq2                    =                    os.path.join(RESULTS_BASE,
"fastqc/{sample}/{sample}_R2_fastqc.html"),
    shell:
        """
        fastqc       -f       fastq       -t       2       --noextract       -o
{RESULTS_BASE}/fastqc/{wildcards.sample} {input.fq1} {input.fq2}
        """
```

G. Align the raw data to tRNA and rRNA, using the STAR program as below

```
rule map_rRNA:
    input:
        R1 = os.path.join(DATA_BASE, "{sample}_R1.fq.gz"),
        R2 = os.path.join(DATA_BASE, "{sample}_R2.fq.gz"),
        rRNA_index = "%s/SAindex" % r_INDEX_DIR,
    output:
        "%s/map_rRNA/{sample}/{sample}Unmapped.out.mate1" % RESULTS_BASE,
        "%s/map_rRNA/{sample}/{sample}Unmapped.out.mate2" % RESULTS_BASE,
    threads: 14
    log:
        "%s/map_rRNA/{sample}/{sample}.log" % RESULTS_BASE,
    shell:
        """
        STAR --genomeDir {r_INDEX_DIR} --readFilesIn {input.R1} {input.R2}
\
        --readFilesCommand gunzip -c --runThreadN {threads} \
        --outFileNamePrefix
{RESULTS_BASE}/map_rRNA/{wildcards.sample}/{wildcards.sample} \
        --outReadsUnmapped     Fastx     --outFilterMatchNmin     40     --
outFilterScoreMinOverLread 0 \
        --outFilterMatchNminOverLread 0 1>{log} 2>&1
        """
```

H. Align the unmapped reads to genome, using the STAR program as below

```
rule map_genome:
    input:
        f_fastq1   =   "%s/map_rRNA/{sample}/{sample}Unmapped.out.mate1"   %
RESULTS_BASE,
        f_fastq2   =   "%s/map_rRNA/{sample}/{sample}Unmapped.out.mate2"   %
RESULTS_BASE,
        genome_index = "%s/SAindex" % G_INDEX_DIR,
    output:
        "%s/map_genome/{sample}/{sample}Aligned.out.bam" % RESULTS_BASE,
    threads: 14
    log:
        "%s/map_genome/{sample}/{sample}.log" % RESULTS_BASE,
    shell:
        """
```

```
        STAR --genomeDir {G_INDEX_DIR} --readFilesIn {input.f_fastq1}
{input.f_fastq2} \
        --runThreadN {threads} \
        --outFileNamePrefix
{RESULTS_BASE}/map_genome/{wildcards.sample}/{wildcards.sample} \
        --outSAMtype BAM Unsorted \
        --outFilterMatchNmin 40 --outFilterScoreMinOverLread 0 --
outFilterMatchNminOverLread 0 \
        1>{log} 2>&1
        """
```

I. Filter out uniquely mapped reads to bam file and make an index using samtools program

```
rule sort_index_uniqbam:
    input:
        rules.map_genome.output
    output:
        unique_bam =
temp("%s/sort_index_uniqbam/{sample}/{sample}_uniq.bam" % RESULTS_BASE),
        sort_bam = "%s/sort_index_uniqbam/{sample}/{sample}_uniq_sort.bam" %
RESULTS_BASE,
        bai = "%s/sort_index_uniqbam/{sample}/{sample}_uniq_sort.bam.bai" %
RESULTS_BASE,
    threads: 8
    log:
        log1 = "%s/sort_index_uniqbam/{sample}/{sample}sort.log" %
RESULTS_BASE,
        log2 = "%s/sort_index_uniqbam/{sample}/{sample}index.log" %
RESULTS_BASE,
    shell:
        """
        {samtools} view -@ {threads} -h -bq 255 {input} > {output.unique_bam}
        {samtools} sort -@ {threads} {output.unique_bam} -o {output.sort_bam}
1>{log.log1} 2>&1
        {samtools} index {output.sort_bam} 1>{log.log2} 2>&1
        """
```

J. Generate bigwig files from indexed bam files, using the bam2wig.py script from rseqc package

```
rule bam2bw:
    input:
        "%s/sort_index_uniqbam/{sample}/{sample}_uniq_sort.bam" %
RESULTS_BASE,
    output:
        "%s/bam2bw/{sample}.bw" % RESULTS_BASE,
    threads: 1
    log:
        "%s/bam2bw/{sample}.log" % RESULTS_BASE,
    shell:
        """
```

```
        bam2wig.py -i {input} -o {RESULTS_BASE}/bam2bw/{wildcards.sample} -
t 1000000000 -s {SIZES} -u \
         1>{log} 2>&1
        """
```

K. Count mapped reads for all annotated genes using the featureCounts program from Rsubreads package

```
rule featurecount:
    input:
        bam  =  "%s/sort_index_uniqbam/{sample}/{sample}_uniq_sort.bam"  %
RESULTS_BASE,
        gtf = REF_GTF,
    output:
        "%s/featurecount/{sample}_featureCounts.txt" % RESULTS_BASE,
        "%s/featurecount/{sample}_featureStat.log" % RESULTS_BASE,
    threads: 8
    params:
        Rscript = Rscript,
    log:
        "%s/featurecount/{sample}.log" % RESULTS_BASE,
    shell:
        """
        {params.Rscript} {SCRIPT_DIR}/featurecount.R {input.bam} {input.gtf}
0 1>{log} 2>&1
        """
```

L. Analyze differential genes based on the reads counts of all genes in different samples with DESeq2 as below

```
rule DESeq2:
    input:
        control = lambda wildcards: ["%s/featurecount/%s_featureCounts.txt" %
(RESULTS_BASE, samp)
            for samp in get_control(wildcards.sample, sample_info)],
        treat = lambda wildcards: ["%s/featurecount/%s_featureCounts.txt" %
(RESULTS_BASE, samp)
            for samp in get_samp(wildcards.sample, sample_info)]
    output:
        tsv = "%s/DESeq2/{sample}_DEG.tsv" % RESULTS_BASE,
    params:
        Rscript = Rscript,
    run:
        samples  =  ",".join(list(map(str,  input.control)))  +  ","  +
",".join(list(map(str,input.treat)))
        group = ",".join(["WT"]*2+["TREAT"]*2)
        cmd2 = " ".join([params.Rscript, SCRIPT_DIR + "/DESeq2.R", samples,
group, output.tsv])
        print(cmd2)
        shell(cmd2)
```

M. Calculate sample correlation based on the reads counts of all genes in different samples with distance.R as below

```
rule sample_cor:
    input:
        "%s/featurecount/{sample}_featureCounts.txt" % RESULTS_BASE,
    output:
        poly= "%s/plot_sample_cor/All_polysome_cor.pdf" % RESULTS_BASE,
total= "%s/plot_sample_cor/All_total_cor.pdf" % RESULTS_BASE,
    params:
        Rscript = Rscript,
    shell:
        """
        {params.Rscript} {SCRIPT_DIR}/distance.R
        """
```

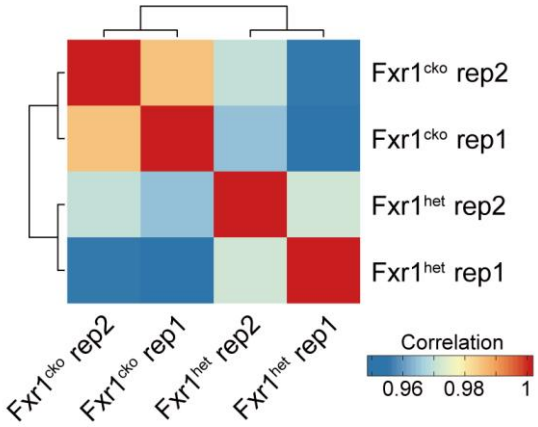


Figure 4. Correlation analysis. The Spearman correlation coefficient among four polysome profiling experiments for *Fxr1*^{cko} and control testes with two replicates. The correlation coefficient was calculated based on reads counts of all genes between two different samples.

N. Visualize polysome profiling results with volcano plot based on the results of differential expression analysis between WT and FXR1 KO with polysome_volcan.R as below

```
rule plot_polysome_volcan:
    input:
        tsv = "%s/DESeq2/{sample}_DEG.tsv" % RESULTS_BASE,
    output:
        "%s/figure/polysome_volcan.pdf" % RESULTS_BASE,
    params:
        Rscript = Rscript,
    shell:
        """
        {params.Rscript} {SCRIPT_DIR}/polysome_volcan.R
        """
```

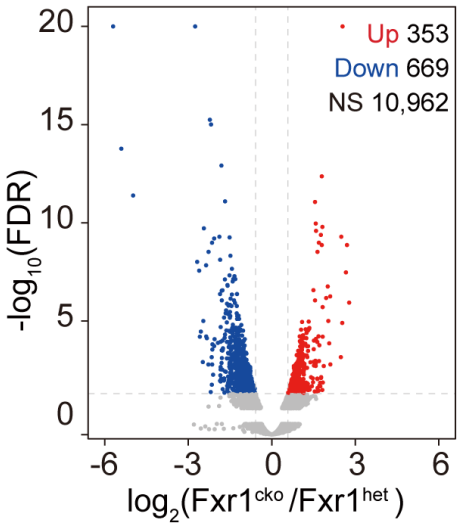


Figure 5. Volcano plot showing the significantly changed genes. The cutoff is FDR < 0.05 and fold-change > 1.5 between *Fxr1*^{cko} and control testes.

O. Analyze differential translational efficiency based on the reads counts of expressed genes between WT and FXR1 KO using the R package Xtail between WT and FXR1 KO as below

```
rule plot_TE_scatter:
    input:
        total = "%s/DESeq2/cKO_total_DEG.tsv"" % RESULTS_BASE,
polysome = "%s/DESeq2/cKO_poly_DEG.tsv"" % RESULTS_BASE,
    output:
        "%s/figure/DE_TE_scatter0.01.pdf" % RESULTS_BASE,
    params:
        Rscript = Rscript,
    shell:
        """
 {params.Rscript} {SCRIPT_DIR}/getExp_mRNA.R
        {params.Rscript} {SCRIPT_DIR}/Xtail.R
 {params.Rscript} {SCRIPT_DIR}/DE_TE_scatter.R
        {params.Rscript} {SCRIPT_DIR}/DE_TE_scatter_0.01.R
       """
```

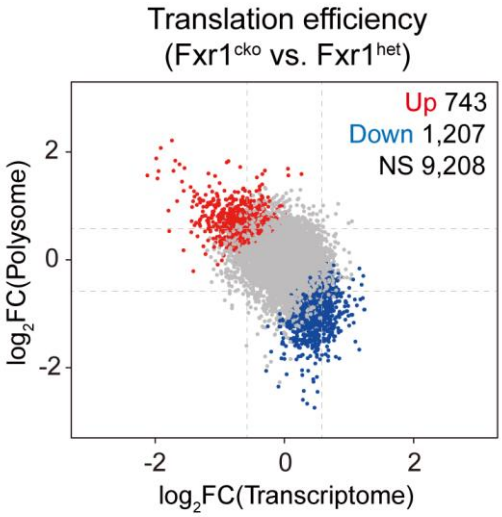


Figure 6. Differential translational efficiency analysis. Scatter plot showing mRNA level changes (x-axis) against polysome profiling changes (y-axis) between adult *Fxr1*^{cko} and control testes, with translation-deficient genes in *Fxr1*^{cko} testes shown in blue.

Notes

1. Once testes are harvested, all manipulations should be performed on ice or at 4 °C.
2. Fill the ultra-clear centrifuge tube with lysate to approximately 2–3 mm from the border to prevent the tube from collapsing during centrifugation.
3. When collecting fractions with Piston Gradient Fractionator, the collection tube needs to be flushed with ddH_2O. We recommend preheating the water to avoid clogging the tube with the sucrose solution.
4. Each sample requires two mice aged 25 days or one mouse older than 35 days. One testis of a 25-day-old mouse weighs approximately 30 mg and one testis of a 35-day-old mouse weighs approximately 40 mg.
5. The amount of water used to dissolve the RNA depends on the abundance of the RNA, or in other words, on the number of testes initially lysed. As a rule of thumb, 50–100 µL of water for INPUT yields approximately 1 µg/µL RNA, and 20–50 µL of water for FRT yields 80–800 ng/µL RNA for polysome fractions and 1–2 µg/µL RNA for RNP fractions. The volume of water used to dissolve 12 fractions should be consistent.

Recipes

1. Polysome buffer (50 mL)

50 mM Tris-HCl (pH 7.0)
100 mM NaCl
5 mM $MgCl_2$
Nuclease free water

2. Lysis buffer (10 mL)

1% Triton X-100
100 µg/mL CHX
cOmplete™ EDTA-free protease inhibitor cocktail

SUPERase•In™ RNase inhibitor
Polysome buffer (up to 10 mL)

3. **60% (w/v) sucrose grading solution (10 mL)**
   60% (w/v) sucrose
   100 μg/mL CHX
   Cocktail protease inhibitor
   RNase inhibitor
   Polysome buffer (up to 10 mL)

4. **10% (w/v) sucrose grading solution (10 mL)**
   10% (w/v) sucrose
   100 μg/mL CHX
   Cocktail protease inhibitor
   RNase inhibitor
   Polysome buffer (up to 10 mL)

5. **75% (v/v) ethanol (50 mL)**
   75% (v/v) ethanol
   Nuclease-free water

Acknowledgments

This protocol was originally applied in our study "LLPS of FXR1 drives spermiogenesis by activating translation of stored mRNAs" published by Science (Kang et al., 2022). We are grateful for funding by the National Key R&D Program of China (2022YFA1303300, 2022YFC2702600 and 2021YFC2700200), National Natural Science Foundation of China (3191101352, 91940305, 31830109, and 31821004), Science and Technology Commission of Shanghai Municipality (2017SHZDZX01, 19JC1410200 and 17JC1420100), Innovative research team of high-level local universities in Shanghai (SHSMU-ZDCX20210902), National Postdoctoral Program for Innovative Talent grant (BX20180331), China Postdoctoral Science Foundation grant (2018M642018), Open Fund of State Key Laboratory of Reproductive Medicine (SKLRM-K202101), and the Foundation of Key Laboratory of Gene Engineering of the Ministry of Education. Furthermore, we would like to thank Ming-Shun Sun from the Cell Biology Core Facility in Shanghai Institute of Biochemistry and Cell Biology (SIBCB), Chinese Academy of Sciences (CAS) for technical assistance.

Competing interests

The authors declare no competing interests.

Ethics

All animal studies were approved by the Institutional Animal Care and Research Advisory Committee in SIBCB, CAS.

References

Chassé, H., Boulben, S., Costache, V., Cormier, P., and Morales, J. (2017). Analysis of translation using polysome profiling. *Nucleic Acids Res* 45(3): e15.

Ingolia, N. T., Brar, G. A., Rouskin, S., McGeachy, A. M. and Weissman, J. S. (2012). The ribosome profiling strategy for monitoring translation in vivo by deep sequencing of ribosome-protected mRNA fragments. *Nat Protoc* 7(8): 1534-1550.

Kang, J. Y., Wen, Z., Pan, D., Zhang, Y., Li, Q., Zhong, A., Yu, X., Wu, Y. C., Chen, Y., Zhang, X., et al. (2022). LLPS of FXR1 drives spermiogenesis by activating translation of stored mRNAs. *Science* 377(6607): eabj6647.

Sassone-Corsi, P. (2002). Unique chromatin remodeling and transcriptional regulation in spermatogenesis. *Science* 296(5576): 2176-2178.

Schwanhäusser, B., Busse, D., Li, N., Dittmar, G., Schuchhardt, J., Wolf, J., Chen, W. and Selbach, M. (2011). Global quantification of mammalian gene expression control. *Nature* 473(7347): 337-342.

Steger, K. (1999). Transcriptional and translational regulation of gene expression in haploid spermatids. *Anat Embryol (Berl)* 199(6): 471-487.