R Notebook to reproduce the results

```
library(pulseR)
replicateNum <- 3
time <- 12
geneNum <- 100
set.seed(259)
```

Generate test data

Define experiment with three fractions: - total - 4sU labelled - flow-through.

Let's generate 3 replicates for every condition for a single time point 12.

```
formulas <- MeanFormulas(
  total = mu,
  lab   = mu * (1 - exp(-d * time)),
  unlab = mu * exp(-d * time)
)
conditions <- data.frame(condition = rep(names(formulas),
                                         each = replicateNum),
                         time = time)
knitr::kable(conditions)
```

condition	time
total	12
total	12
total	12
lab	12
lab	12
lab	12
unlab	12
unlab	12
unlab	12

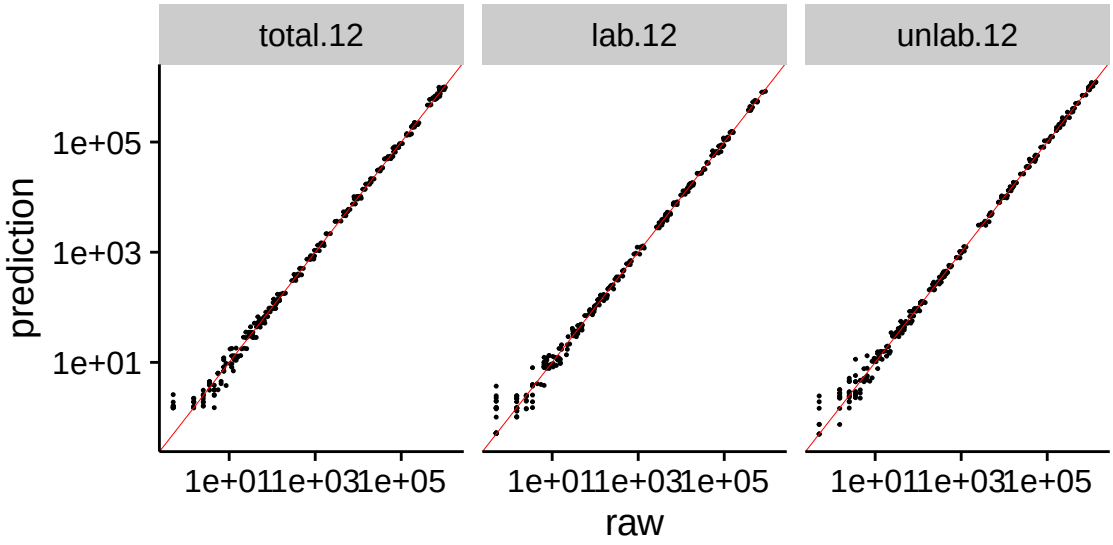
Here we create parameters for 100 genes:

```
rownames(conditions) <- paste0("sample_", seq_along(conditions$condition))
t <- pulseR:::addKnownShared(formulas, conditions)
formulas_known <- t$formulas
conditions_known <- data.frame(condition = t$conditions)

fractions <- factor(conditions_known$condition)
par <- list(size = 1e2)
par$individual_params <- data.frame(mu = 10^runif(geneNum, 0, 6),
                                    d = -log(runif(geneNum, .1, .9))/12)
rownames(par$individual_params) <- paste0("gene_", 1:geneNum)

par$fraction_factors <- 1 * (1:(length(levels(fractions)) - 1))
```

```
counts <- generateTestDataFrom(formulas_known, par,
                               conditions_known,
                               fractions)

pd <- PulseData(
    count_data = counts,
    conditions = conditions,
    formulas   = formulas,
    fractions  = ~condition+time)
```

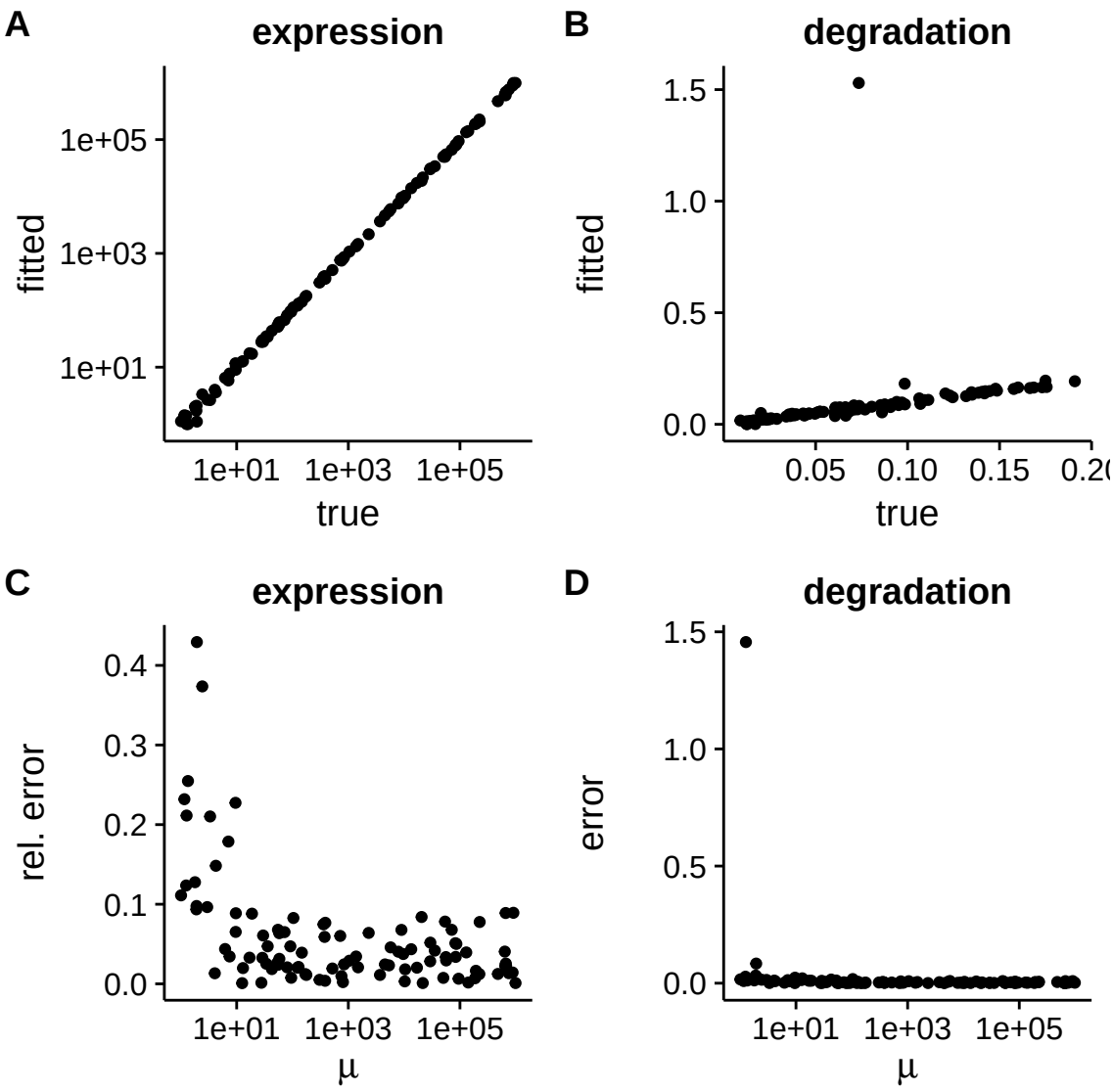
```
par2 <- par
guess <-  apply(counts[, conditions$condition == "total"], 1, mean)
par2$individual_params$mu <- guess
par2$individual_params$d <- runif(geneNum,0,5)/time
par2$size <- 1e4
par2$fraction_factors <- rep(1, length(par$fraction_factors))

options <- list(
  lower_boundary = c(1,1e-3),
  upper_boundary = c(1e10, 5),
  lower_boundary_size = 1,
  upper_boundary_size = 1e9,
  lower_boundary_fraction = .1,
  upper_boundary_fraction = 10,
  cores = 2
)
options$parscales <- c(1e5,1)

fit <- fitModel(pd, par2, options)
```

```
pr <- predictExpression(fit$par, pd)
```




```
q_d <- qplot(
  y = fit$par$individual_params$d,
  x = par$individual_params$d,
  xlab = "true",
  ylab = "fitted",
```

```
  #main="d"
  main="degradation"
  )

q_mu <- qplot(
  y = fit$par$individual_params$mu,
  x = par$individual_params$mu,
  log = 'xy',
  xlab = "true",
  ylab = "fitted",
  #main=expression(mu)
  main="expression"

)

delta <- par$individual_params - fit$par$individual_params
delta_d <- qplot(
  x = par$individual_params$mu,
  y = abs(delta$d),
  log = 'x',
  #main="d"
  main="degradation"
  ) +
  xlab(expression(mu)) +
  ylab("error")
delta_mu <- qplot(
  x = par$individual_params$mu,
  y = abs(delta$mu) / par$individual_params$mu,
  log = 'x',
  #main=expression(mu)
  main="expression"
  ) +
  xlab(expression(mu)) +
  ylab("rel. error")

q <- plot_grid(q_mu,
               q_d,
               delta_mu,
               delta_d,
               align = "hv",
               labels = LETTERS[1:4])
q
```




```
save_plot(filename = "../paper/fig/parameters.tiff",
          plot = q,
          base_height = 6,
          base_width = 6)
save_plot(filename = "../paper/fig/parameters.pdf",
          plot = q,
          base_height = 6,
          base_width = 6)
```