# Part V: Profiling and Parallel Processing

```
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(forcats)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(tibble)
library(profvis)
```

```
name_basics      <- read_rds("data/name_basics.rda")
title_basics     <- read_rds("data/title_basics.rda")
title_principals <- read_rds("data/title_principals.rda")
title_ratings    <- read_rds("data/title_ratings.rda")
```

##Part V: Profiling and Parallel Processing

Profiling the Genre-Proportions Pipeline

```
# Profile known-for median rating pipeline
profvis({
  # Expand knownForTitles
  known_for_expanded <- name_basics %>%
    select(nconst, primaryName, knownForTitles) %>%
    filter(!is.na(knownForTitles)) %>%
    separate_rows(knownForTitles, sep = ",") %>%
    rename(tconst = knownForTitles) %>%
    inner_join(title_ratings, by = "tconst")

  # Summarize median rating
  known_for_median <- known_for_expanded %>%
    group_by(nconst, primaryName) %>%
    summarise(median_knownfor_rating = median(averageRating, na.rm = TRUE), .groups = "drop") %>%
```

```r
    arrange(desc(median_knownfor_rating))
})
```

2. Parallelizing the “Known-For” Count

```r
library(parallel)

# Sequential version
seq_time <- system.time({
  counts_seq <- sapply(
    name_basics$knownForTitles,
    function(x) length(strsplit(x, ",")[[1]])
  )
})

# Parallel version using mclapply
cores <- detectCores() - 1
par_time <- system.time({
  counts_par <- mclapply(
    name_basics$knownForTitles,
    function(x) length(strsplit(x, ",")[[1]]),
    mc.cores = cores
  )
})

# Timing results
seq_time
```

```
##    user  system elapsed
##   4.635   0.093   4.727
```

```r
par_time
```

```
##    user  system elapsed
##   4.016   3.324   2.289
```

3. Benchmarking String-Counting Functions

```r
library(bench)
library(stringr)

# Approach 1: strsplit + lengths
f1_count <- function(x) {
  lengths(strsplit(x, ","))
}

# Approach 2: str_count + 1
f2_count <- function(x) {
  str_count(x, ",") + 1
}

# Benchmarking
bm <- mark(
  split_lengths = f1_count(name_basics$knownForTitles),
  str_count     = f2_count(name_basics$knownForTitles),
  iterations    = 20,
  check = FALSE
)

bm
```

```
## # A tibble: 2 x 6
##   expression         min   median `itr/sec` mem_alloc `gc/sec`
##   <bch:expr>    <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
## 1 split_lengths    897ms    1.01s     0.975    5.76MB    0.244
## 2 str_count        135ms  136.5ms     7.13      5.8MB    0
```