Parallel Computing

Ed Gonzalez

2022-12-01

I had to move onto the desktop version of R Studio for this as the cloud-based program did not allow me to use more than 0.5GB of RAM

```
install.packages("readr", repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/tz/sh20cj15711657_9_1d4v6m00000gn/T//Rtmpau3axm/downloaded_packages
```

```
library(readr)

airlines09 <- read_csv("/Users/ed/Downloads/2009.csv")
```

```
## Rows: 6429338 Columns: 28
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (4): OP_CARRIER, ORIGIN, DEST, CANCELLATION_CODE
## dbl  (22): OP_CARRIER_FL_NUM, CRS_DEP_TIME, DEP_TIME, DEP_DELAY, TAXI_OUT, W...
## lgl   (1): Unnamed: 27
## date  (1): FL_DATE
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

Installing some packages for tidying the data set and for parallel computing

```
install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/tz/sh20cj15711657_9_1d4v6m00000gn/T//Rtmpau3axm/downloaded_packages
```

```
install.packages("sparklyr", repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/tz/sh20cj15711657_9_1d4v6m00000gn/T//Rtmpau3axm/downloaded_packages
```

```
install.packages("multidplyr", repos = "http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/tz/sh20cj15711657_9_1d4v6m00000gn/T//Rtmpau3axm/downloaded_packages
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v dplyr   1.0.10
## v tibble  3.1.8      v stringr 1.5.0
## v tidyr   1.2.1      v forcats 0.5.2
## v purrr   0.3.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(sparklyr)
```

```
##
## Attaching package: 'sparklyr'
##
## The following object is masked from 'package:purrr':
##
##     invoke
##
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(multidplyr)
library(dplyr)
```

Timing some code using dplyr that finds the mean arrival delay of each carrier

```
system.time(
  airline_delay <- airlines09 %>%
  group_by(OP_CARRIER) %>%
  summarize(mean = mean(ARR_DELAY, na.rm = TRUE)) %>%
    print())
```

```
## # A tibble: 19 x 2
##    OP_CARRIER   mean
##    <chr>       <dbl>
##  1 9E          0.950
##  2 AA          5.67
##  3 AS          1.32
##  4 B6          5.08
##  5 CO          5.51
##  6 DL          4.84
##  7 EV         11.7
##  8 F9          5.69
##  9 FL          8.20
## 10 HA          0.261
## 11 MQ          5.97
## 12 NW          3.92
## 13 OH         10.7
## 14 OO          3.30
## 15 UA          1.03
## 16 US          2.09
## 17 WN          1.66
## 18 XE          5.75
## 19 YV          6.04
```

```
##    user  system elapsed
##   0.127   0.031   0.159
```

Same as before, but now utilizing the for %do% process and timing it

```
library(foreach)
```

```
##
## Attaching package: ’foreach’
```

```
## The following objects are masked from ’package:purrr’:
##
##     accumulate, when
```

```
carriers<-split(airlines09,airlines09$OP_CARRIER)

system.time(
  {foreach(i=carriers,.combine=c) %do%
  mean(i$ARR_DELAY, na.rm = T) %>%
  print()}
)
```

```
##  [1]  0.9500134  5.6708979  1.3170298  5.0792806  5.5138186  4.8419736
##  [7] 11.7207813  5.6856234  8.1978418  0.2607874  5.9679675  3.9242081
## [13] 10.6877569  3.3035038  1.0341901  2.0899596  1.6638597  5.7545910
## [19]  6.0439209
##    user  system elapsed
##   0.034   0.018   0.053
```

Using the doParallel package where I essentially just change the %do% to %dopar% and assign some cores

```
library(doParallel)
```

```
## Loading required package: iterators

## Loading required package: parallel
```

```
n.cores <- detectCores() - 1
registerDoParallel(n.cores)

system.time({
  foreach(i=carriers,.combine=c) %dopar%
  mean(i$ARR_DELAY, na.rm = T) %>%
  print()}
)
```

```
##  [1]  0.9500134  5.6708979  1.3170298  5.0792806  5.5138186  4.8419736
##  [7] 11.7207813  5.6856234  8.1978418  0.2607874  5.9679675  3.9242081
## [13] 10.6877569  3.3035038  1.0341901  2.0899596  1.6638597  5.7545910
## [19]  6.0439209
```

```
##    user  system elapsed
##   0.053   0.088   0.043
```

Same procee utilizng the multidplyr package

```
library(multidplyr)
cl <- new_cluster(7)

system.time(
  {
  airline_delay <- airlines09 %>%
    group_by(OP_CARRIER) %>%
    partition(cl) %>%
    summarize(mean = mean(ARR_DELAY, na.rm = TRUE)) %>%
    collect() %>%
    print()}
  )
```

```
## # A tibble: 19 x 2
##    OP_CARRIER   mean
##    <chr>       <dbl>
##   1 9E          0.950
##   2 NW          3.92
##   3 YV          6.04
##   4 AA          5.67
##   5 XE          5.75
##   6 AS          1.32
##   7 F9          5.69
##   8 HA          0.261
##   9 OH         10.7
##  10 WN          1.66
##  11 B6          5.08
##  12 FL          8.20
##  13 US          2.09
##  14 CO          5.51
##  15 MQ          5.97
##  16 DL          4.84
##  17 UA          1.03
##  18 EV         11.7
##  19 OO          3.30
```

```
##    user  system elapsed
##   3.399   1.997  17.210
```

Creating a table with times for each computation

```
tidyverse.time <- c(0.128, 0.045, 0.233)
foreach.do.time <- c(0.032, 0.021, 0.052)
foreach.dopar.time <- c(0.004, 0.025, 0.085)
multidplyr.time <- c(3.73, 1.834, 7.336)

rbind(tidyverse.time, foreach.do.time, foreach.dopar.time, multidplyr.time)
```

```
##                     [,1]  [,2]  [,3]
## tidyverse.time     0.128 0.045 0.233
## foreach.do.time    0.032 0.021 0.052
## foreach.dopar.time 0.004 0.025 0.085
## multidplyr.time    3.730 1.834 7.336
```

Using dplyr to select for specific variables and performing a summary function to find the mean of arrival delay, counting unique instances, and finding a linear regression model between arrival delay and taxi in/out times

```
system.time({
  taxi.times <- airlines09 %>%
  select(ORIGIN, DEST, OP_CARRIER, ARR_DELAY, TAXI_IN, TAXI_OUT) %>%
  drop_na() %>%
  group_by(ORIGIN, DEST, OP_CARRIER) %>%
  summarise(Count = n(),Mean = mean(ARR_DELAY, na.rm = T),
            fit = summary(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT))$r.squared)
  print(head(taxi.times))
})
```

```
## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable
## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable

## Warning in summary.lm(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT)): essentially perfect
## fit: summary may be unreliable
```

```
## ‘summarise()‘ has grouped output by ’ORIGIN’, ’DEST’. You can override using
## the ‘.groups‘ argument.
```

```
## # A tibble: 6 x 6
## # Groups:   ORIGIN, DEST [5]
##   ORIGIN DEST  OP_CARRIER Count  Mean    fit
##   <chr>  <chr> <chr>      <int> <dbl>  <dbl>
## 1 ABE    ATL   EV           766 17.0  0.116
## 2 ABE    CLE   XE           502 -5.89 0.201
## 3 ABE    CLT   US           354 -1.40 0.0790
## 4 ABE    CLT   YV            35  2.43 0.108
## 5 ABE    DTW   9E           969 -1.05 0.134
## 6 ABE    FLL   FL            62 10.2  0.0191
```

```
##    user  system elapsed
##   3.976   0.161   4.197
```

Doing the same as before but utilizing multidplyr functions to compare times

```
system.time({
  taxi.times2 <- airlines09 %>%
  select(ORIGIN, DEST, OP_CARRIER, ARR_DELAY, TAXI_IN, TAXI_OUT) %>%
  drop_na() %>%
  group_by(ORIGIN, DEST, OP_CARRIER) %>%
    partition(cl) %>%
    summarise(Count = dplyr::n(),Mean = mean(ARR_DELAY, na.rm = T),
            fit = summary(lm(ARR_DELAY ~ TAXI_IN + TAXI_OUT))$r.squared)
  print(head(taxi.times2))
})
```

```
## # A tibble: 6 x 6
## # Groups:   ORIGIN, DEST [6]
##   ORIGIN DEST  OP_CARRIER Count  Mean    fit
##   <chr>  <chr> <chr>      <int> <dbl>  <dbl>
## 1 ABE    ATL   EV           766 17.0  0.116
## 2 ABQ    DEN   F9          1042  3.28 0.0560
## 3 ABQ    HOU   WN          1000 -1.60 0.0258
## 4 ABQ    LBB   WN           363  2.15 0.0476
## 5 ABQ    MAF   WN           362  9.80 0.0183
## 6 ABQ    MDW   WN           724 -5.25 0.0312
```

```
##    user  system elapsed
##   1.589   0.566  14.289
```

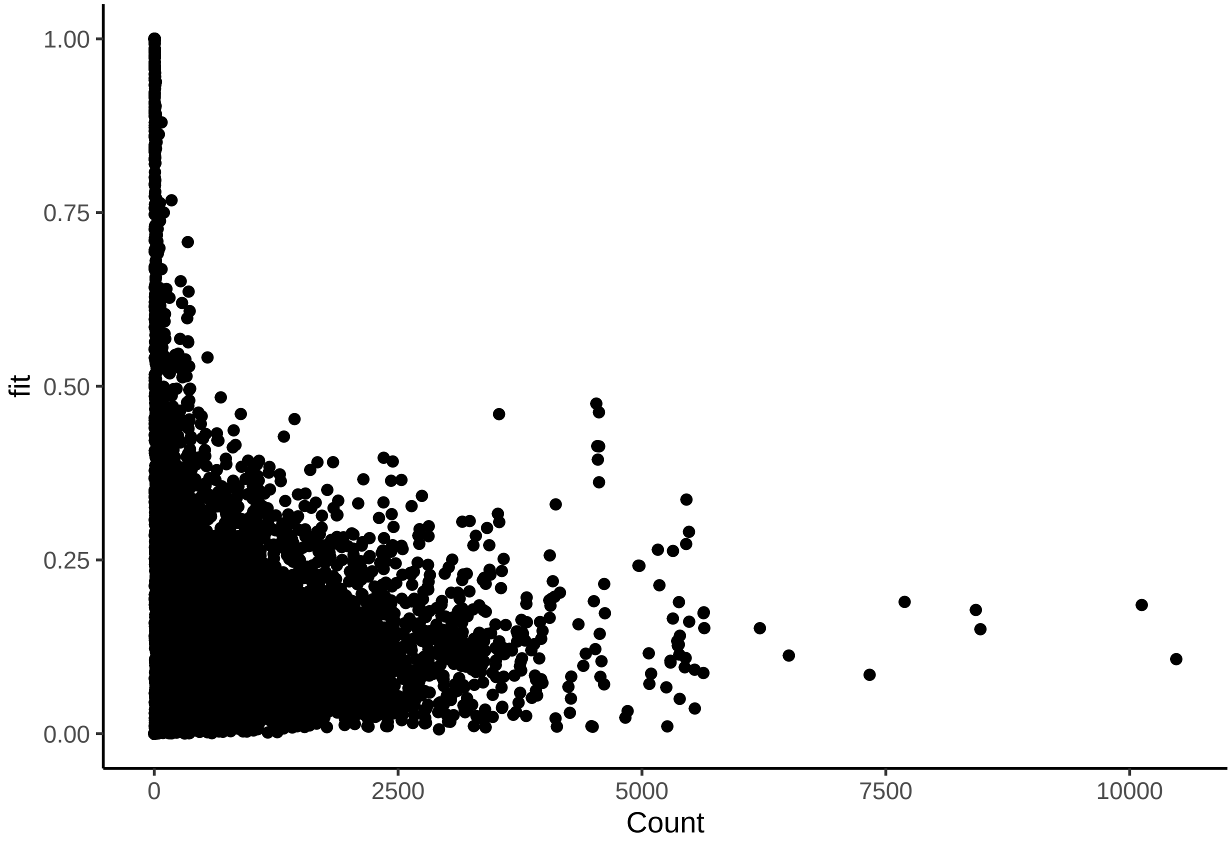
Using ggplot2 package to create a scatter plot

```
library(ggplot2)

scatter.p <- ggplot2::ggplot(taxi.times, aes(Count, fit))

scatter.p + geom_point() + theme_classic()
```

```
## Warning: Removed 2 rows containing missing values (`geom_point()`).
```

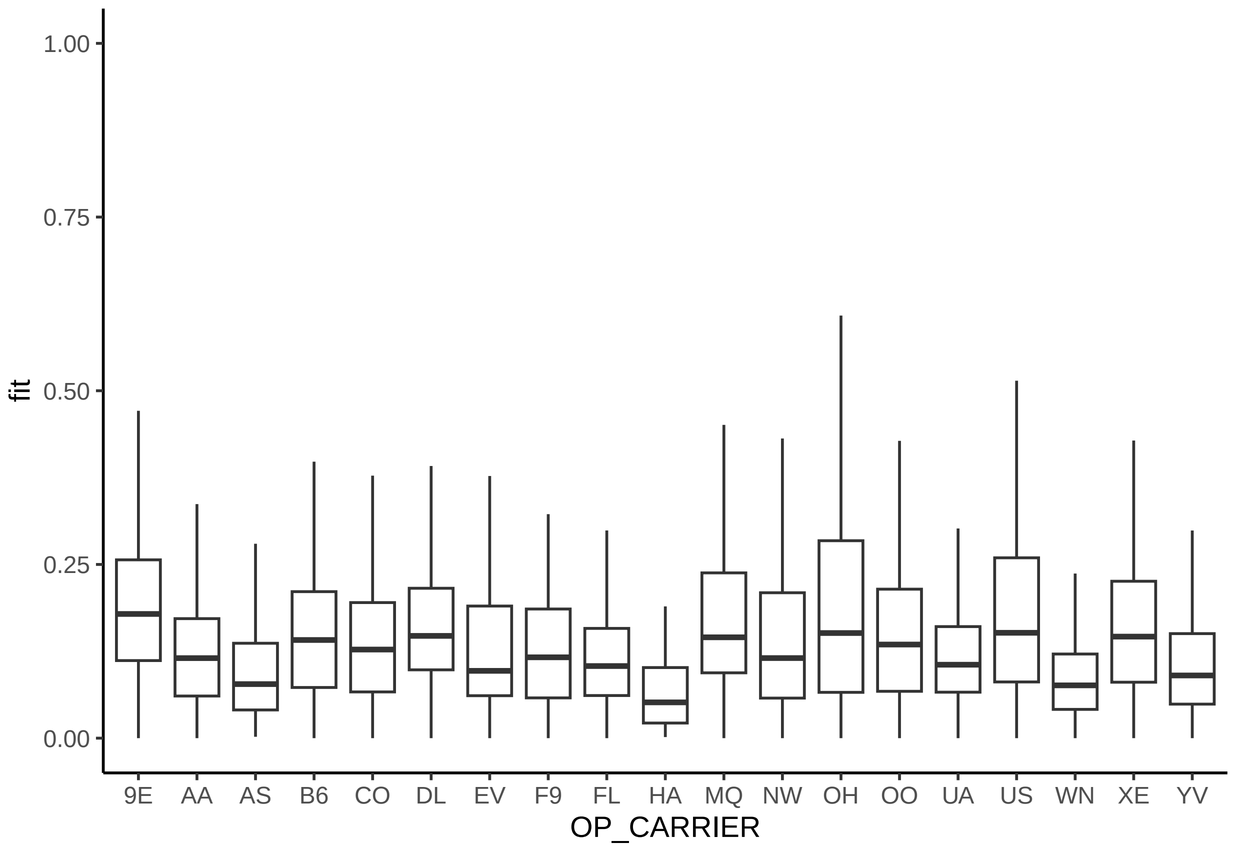


# Creating boxplots for each individual airline

```
box.p <- ggplot2::ggplot(taxi.times, aes(OP_CARRIER, fit))

box.p+geom_boxplot(outlier.shape = NA ) + theme_classic()
```

```
## Warning: Removed 2 rows containing non-finite values (‘stat_boxplot()‘).
```


1.00
0.75
0.50
0.25
0.00
fit
9E
AA
AS
B6
CO
DL
EV
F9
FL
HA
MQ
NW
OH
OO
UA
US
WN
XE
YV
OP_CARRIER