Effi-VO.R

arcs

Fri Dec 1 14:32:39 2017

```
########################################################################
###### Objective of this program is to ##############################
###### study normalised efficiency of jobs and ######################
###### To study normalised efficiency for each VO ##################
########################################################################

library(data.table)
library(ggplot2)

setwd("/home/arcs/Oct14/DataCSV")
getwd()
```

```
## [1] "/home/arcs/Oct14/DataCSV"
```

```
jobs <<- fread("Oct2017Efficiency_VO.csv")
```

```
##
Read 76.1% of 5876000 rows
Read 5876000 rows and 8 (of 8) columns from 0.193 GB file in 00:00:03
```

```
############### Function to print values ##########################
printf <- function(...) cat(sprintf(...))



########################################################################
############# Studying the structure of Data ######################
########################################################################
names(jobs)
```

```
## [1] "RequestCpus"         "MATCH_HEPSPEC"       "MATCH_TotalCpus"
## [4] "RemoteWallClockTime" "ExitCode"            "RemoteSysCpu"
## [7] "RemoteUserCpu"       "x509UserProxyVOName"
```

```
str(jobs)
```

```
## Classes 'data.table' and 'data.frame':   5876000 obs. of  8 variables:
##  $ RequestCpus        : int  8 8 8 8 8 8 8 1 1 8 ...
##  $ MATCH_HEPSPEC      : chr  "None" "None" "None" "None" ...
##  $ MATCH_TotalCpus    : chr  "None" "None" "None" "None" ...
##  $ RemoteWallClockTime: chr  "None" "None" "None" "None" ...
##  $ ExitCode           : chr  "None" "None" "None" "None" ...
##  $ RemoteSysCpu       : int  0 0 0 0 0 0 0 97 182 25311 ...
##  $ RemoteUserCpu      : int  0 0 0 0 0 0 0 49122 663 1323662 ...
##  $ x509UserProxyVOName: chr  "cms" "cms" "cms" "cms" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
summary(jobs)
```

```
##   RequestCpus    MATCH_HEPSPEC      MATCH_TotalCpus    RemoteWallClockTime
##  Min.   :1.000   Length:5876000     Length:5876000     Length:5876000
```

```
##  1st Qu.:1.000   Class :character   Class :character   Class :character
##  Median :1.000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :2.018
##  3rd Qu.:1.000
##  Max.   :8.000
##    ExitCode          RemoteSysCpu      RemoteUserCpu
##  Length:5876000     Min.   :     0.0   Min.   :      0
##  Class :character   1st Qu.:     0.0   1st Qu.:      2
##  Mode  :character   Median :     2.0   Median :      5
##                     Mean   :   294.6   Mean   :  15690
##                     3rd Qu.:   110.0   3rd Qu.:   9335
##                     Max.   :298748.0   Max.   :1989119
##  x509UserProxyVOName
##  Length:5876000
##  Class :character
##  Mode  :character
##
##
##
```

```
printf("\nTotal number of jobs: %d\n", nrow(jobs))
```

```
##
## Total number of jobs: 5876000
```

```
#########################################################################
############# Conversion to numeric values ##############################
#########################################################################
jobs[,"RemoteWallClockTime"] <- as.numeric(unlist(jobs[,"RemoteWallClockTime"])) #RemoteWallClockTime
```

```
## Warning: NAs introduced by coercion
```

```
jobs[, "ExitCode"] <- as.numeric(unlist(jobs[, "ExitCode"]))
```

```
## Warning: NAs introduced by coercion
```

```
jobs[, "MATCH_HEPSPEC"] <- as.numeric(unlist(jobs[, "MATCH_HEPSPEC"]))
```

```
## Warning: NAs introduced by coercion
```

```
jobs[, "MATCH_TotalCpus"] <- as.numeric(unlist(jobs[, "MATCH_TotalCpus"]))
```

```
## Warning: NAs introduced by coercion
```

```
#########################################################################
###################### Data Cleansing ###################################
#########################################################################
jobs <- na.omit(jobs)
printf("\nTotal no of jobs after removing NA:%d \n", nrow(jobs))
```

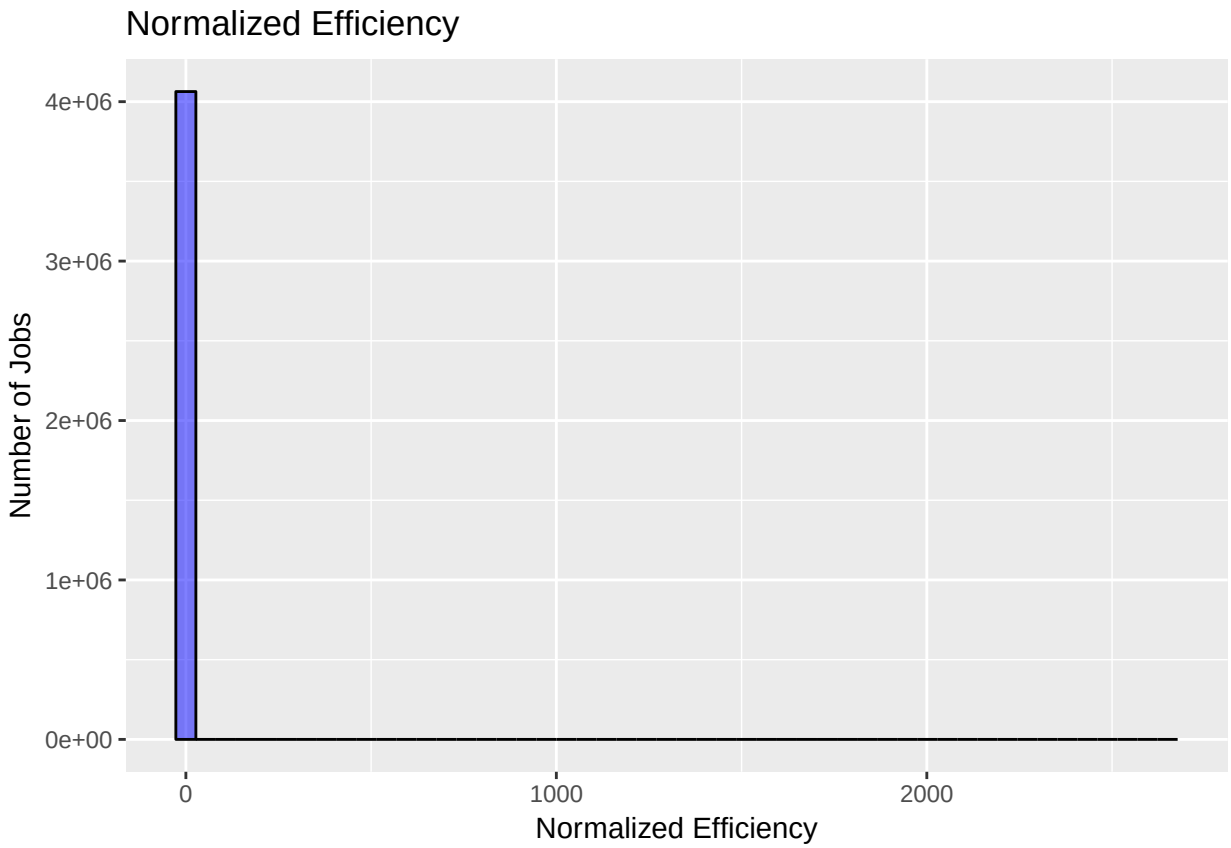
```
##
## Total no of jobs after removing NA:4882661
```

```
#########################################################################
############### Computation of efficiency ###############################
#########################################################################
jobs$CPUTime <- jobs$RemoteSysCpu + jobs$RemoteUserCpu
jobs$WallTime <- jobs$RemoteWallClockTime
jobs$HEPSPEC_TotalCpus <- jobs$MATCH_HEPSPEC/ jobs$MATCH_TotalCpus
```

```
jobs$NWallTime <- jobs$WallTime * jobs$RequestCpus * jobs$HEPSPEC_TotalCpus
jobs$NCPUTime <- jobs$CPUTime * jobs$HEPSPEC_TotalCpus
jobs <- subset(jobs, NWallTime != 0)
printf("\nTotal no of jobs after removing jobs with normalized walltime = 0: %d\n", nrow(jobs))
```

```
##
## Total no of jobs after removing jobs with normalized walltime = 0: 4063470
```

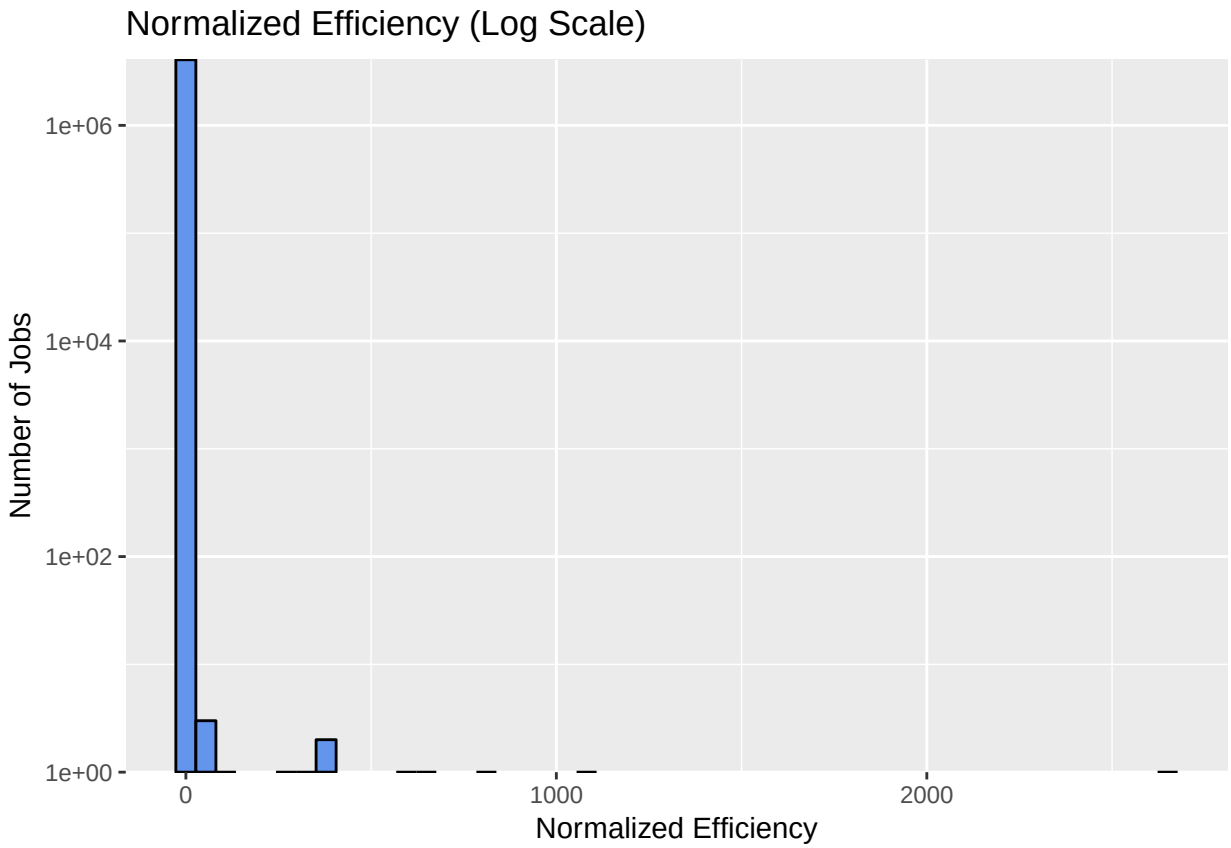
```
jobs$NEfficiency <- jobs$NCPUTime/jobs$NWallTime

graph1 <- ggplot(jobs, aes(x = NEfficiency)) +
  geom_histogram( color = "Black", fill = "Blue", bins = 50, alpha = 0.5 )
graph1 + labs(title= "Normalized Efficiency", x= "Normalized Efficiency", y = "Number of Jobs")
```




```
graph2 <- ggplot(jobs, aes(x = NEfficiency)) +
  geom_histogram(color = "Black", fill = "cornflowerblue", bins = 50 ) +
  scale_y_continuous(trans="log10", expand=c(0,0))
graph2 + labs(title= "Normalized Efficiency (Log Scale)", x= "Normalized Efficiency", y = "Number of Jo
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 39 rows containing missing values (geom_bar).
```



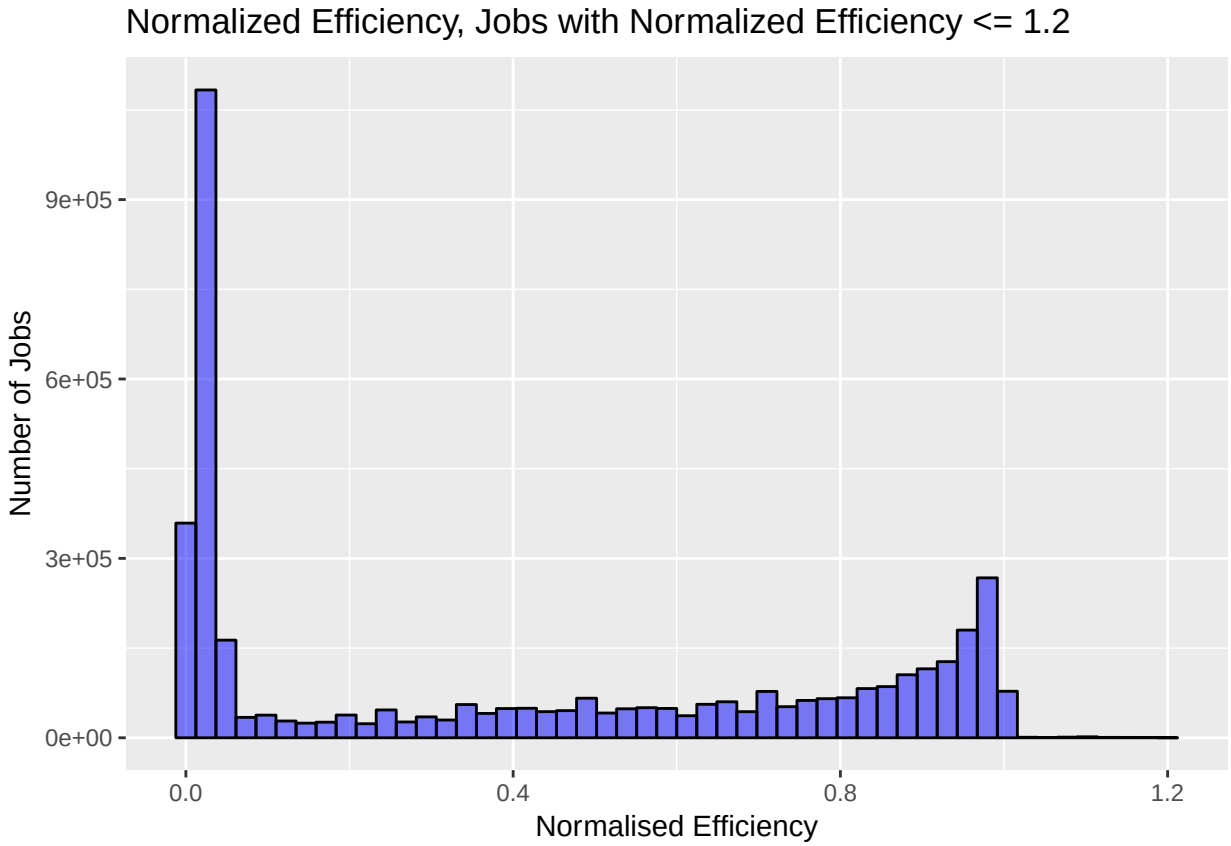

```
printf("\nTotal no of jobs with normalized efficiency <= 1.2: %d\n", nrow(subset(jobs, NEfficiency <= 1
```

```
##
## Total no of jobs with normalized efficiency <= 1.2: 4062607
```

```
printf("\nTotal no of jobs with normalized efficiency > 1.2: %d\n", nrow(subset(jobs, NEfficiency > 1.2)
```

```
##
## Total no of jobs with normalized efficiency > 1.2: 863
```

```
graph3 <- ggplot(subset(jobs, jobs$NEfficiency <= 1.2), aes(x = NEfficiency)) +
  geom_histogram( color = "Black", fill = "Blue", bins = 50, alpha = 0.5 )
graph3 + labs(title= "Normalized Efficiency, Jobs with Normalized Efficiency <= 1.2", x= "Normalised Ef
```




```
VO = unique(jobs$x509UserProxyVOName)

for (vo in VO){
  printf("\n\n\n************ VO Name: %s ***************\n", vo)
    sub_Data <- subset(jobs, x509UserProxyVOName == vo)
  printf("\nNumber of observation: %d", nrow(sub_Data))
  printf("\nPercentage of data: %f", (nrow(sub_Data)/nrow(jobs))*100)
  NEfficiency_sub <- sum(sub_Data$NCPUTime)/sum(sub_Data$NWallTime)
  printf("\nNormalized Efficiency: ")
  print(NEfficiency_sub)
}
```

```
##
##
##
## ************ VO Name: atlas ***************
##
## Number of observation: 1125330
## Percentage of data: 27.693818
## Normalized Efficiency: [1] 0.7800236
##
##
##
## ************ VO Name: cms ***************
##
## Number of observation: 148123
## Percentage of data: 3.645234
```

```
## Normalized Efficiency: [1] 0.6291599
##
##
##
## ************ VO Name: vo.compass.cern.ch ***************
##
## Number of observation: 1440857
## Percentage of data: 35.458783
## Normalized Efficiency: [1] 0.664588
##
##
##
## ************ VO Name: lhcb ***************
##
## Number of observation: 142273
## Percentage of data: 3.501269
## Normalized Efficiency: [1] 0.9386083
##
##
##
## ************ VO Name: ilc ***************
##
## Number of observation: 115228
## Percentage of data: 2.835704
## Normalized Efficiency: [1] 0.9015873
##
##
##
## ************ VO Name: alice ***************
##
## Number of observation: 1091655
## Percentage of data: 26.865093
## Normalized Efficiency: [1] 0.7649069
##
##
##
## ************ VO Name: None ***************
##
## Number of observation: 4
## Percentage of data: 0.000098
## Normalized Efficiency: [1] 0.9920202
```