NEffi_final.R

arcs

Thu Dec 21 15:20:56 2017

```
#Aim: To compute the normalised efficiency and study the low efficiency jobs

library(data.table)
library(ggplot2)


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

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

```
jobs <- fread(input = "Nov2017Efficiency_final1.0.csv", sep = ",", fill = TRUE)
```

```
## Warning in fread(input = "Nov2017Efficiency_final1.0.csv", sep = ",",
## fill = TRUE): Bumped column 1 to type character on data row 673341, field
## contains 'None'. Coercing previously read values in this column from
## logical, integer or numeric back to character which may not be lossless;
## e.g., if '00' and '000' occurred before they will now be just '0', and
## there may be inconsistencies with treatment of ',,' and ',NA,' too (if they
## occurred in this column before the bump). If this matters please rerun and
## set 'colClasses' to 'character' for this column. Please note that column
## type detection uses a sample of 1,000 rows (100 rows at 10 points) so
## hopefully this message should be very rare. If reporting to datatable-help,
## please rerun and include the output from verbose=TRUE.
```

```
##
Read 7.6% of 8986213 rows
Read 26.5% of 8986213 rows
Read 44.8% of 8986213 rows
Read 63.4% of 8986213 rows
Read 82.3% of 8986213 rows
Read 8986213 rows and 18 (of 18) columns from 0.672 GB file in 00:00:08
```

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


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

```
## 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
```

```r
# Removing jobs with NA in Particular Col
jobs <- jobs[!is.na((jobs$RemoteWallClockTime))]
jobs <- jobs[!is.na(jobs$MATCH_HEPSPEC)]
jobs <- jobs[!is.na(jobs$MATCH_TotalCpus)]

# Setting default values for MATCH_HEPSPEC and MATCH_TotalCpus

index <- jobs$MATCH_HEPSPEC == 0
jobs$MATCH_HEPSPEC[index] <- 80
jobs$MATCH_TotalCpus <- 8

# Computation of efficiency
jobs$CPUTime <- jobs$RemoteSysCpu + jobs$RemoteUserCpu
jobs <- jobs[!is.na(jobs$CPUTime)]
jobs$WallTime <- jobs$RemoteWallClockTime
jobs <- subset(jobs, jobs$WallTime != 0) #Removing jobs with WallTime = 0
jobs <- subset(jobs, jobs$MATCH_TotalCpus != 0)
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: 7128525
```

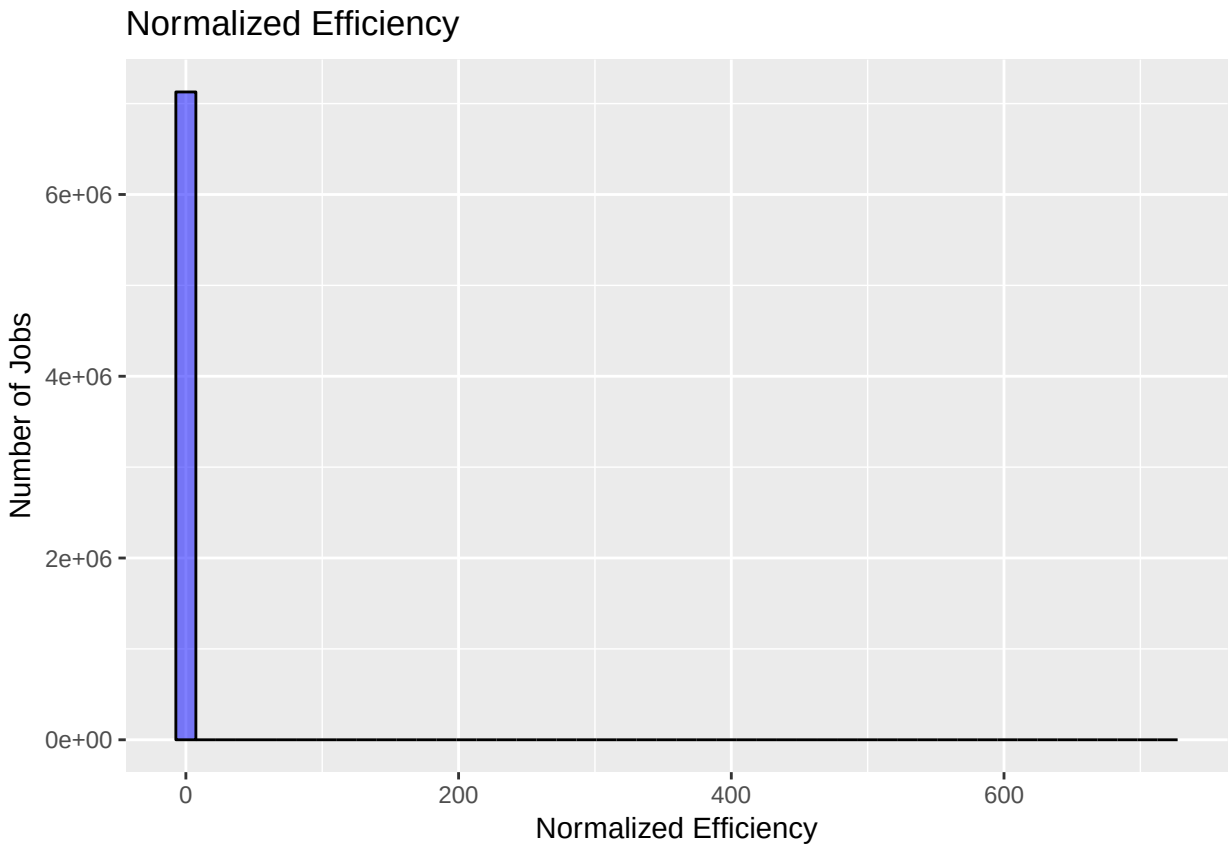
```r
jobs$NEfficiency <- jobs$NCPUTime/jobs$NWallTime
Total_NEfficiency <- sum(jobs$NCPUTime)/sum(jobs$NWallTime)

printf("\n\n Normalised Efficiency(For all jobs):")
```

```
##
##
##  Normalised Efficiency(For all jobs):
```

```r
print(Total_NEfficiency)
```

```
## [1] 0.7432982
```

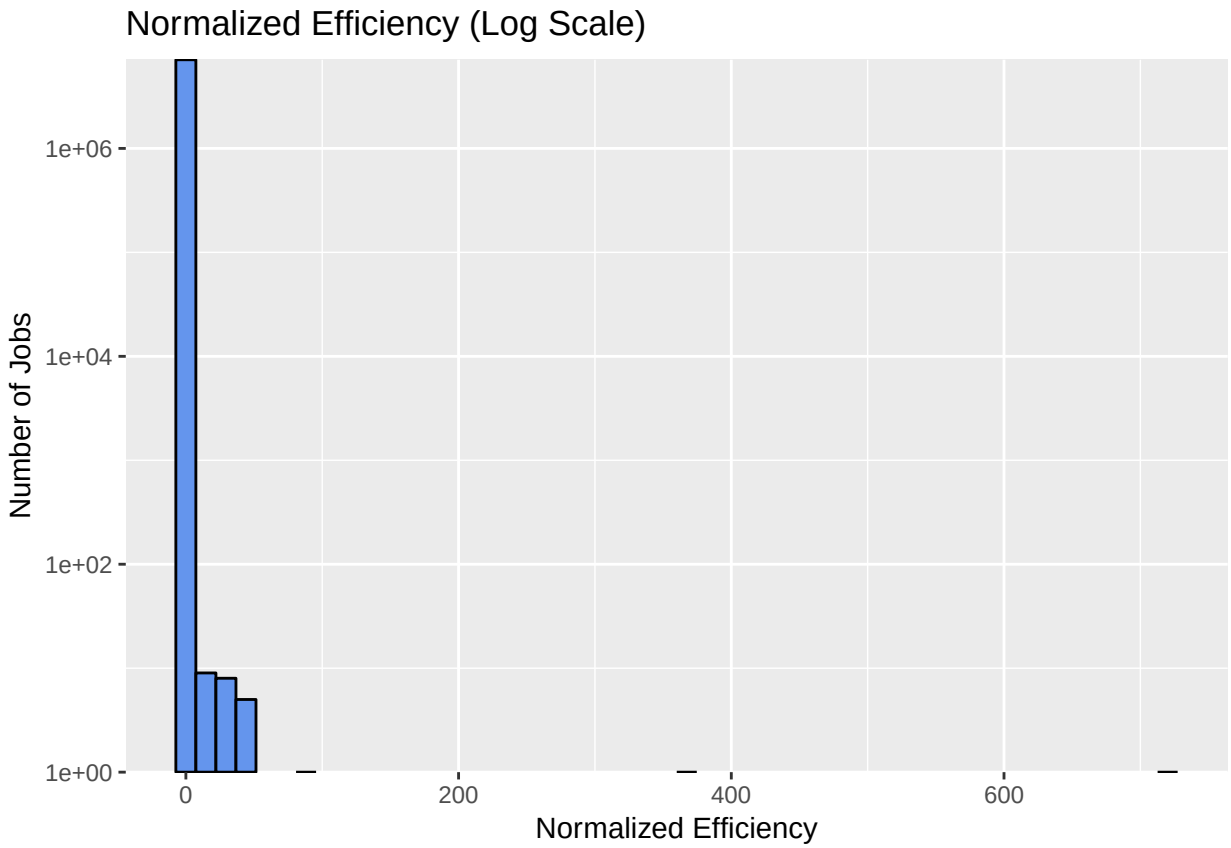
```r
# Visiualisation of Efficiency of jobs
graph1 <- ggplot(jobs, aes(x = NEfficiency)) +
  geom_histogram( color = "Black", fill = "Blue", bins = 50, alpha = 0.5 )
graph1 + labs(title= "Normalized Efficiency", x= "Normalized Efficiency", y = "Number of Jobs")
```




```
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 43 rows containing missing values (geom_bar).
```




```
# Jobs with very high efficiency- Error
error_jobs <- subset(jobs, NEfficiency > 1.2)
error_fraction <- nrow(error_jobs)/nrow(jobs)

printf("\n\n Fraction of very high efficiency jobs:")
```

```
##
##
##  Fraction of very high efficiency jobs:
```

```
print(error_fraction)
```

```
## [1] 0.0001970955
```

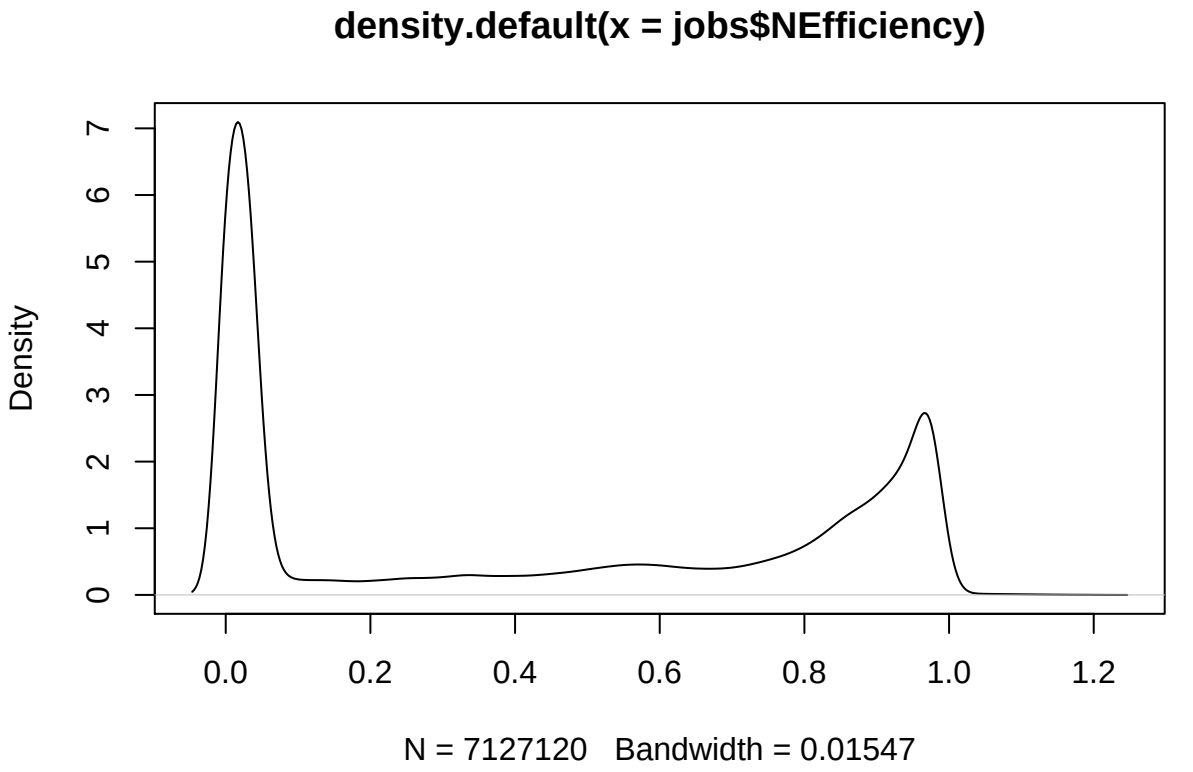
```
# Correction by eliminating jobs with high efficiency
jobs <- subset(jobs, NEfficiency <= 1.2)
corrected_eff <- sum(jobs$NCPUTime)/sum(jobs$NWallTime)

printf("\n\n Normalised Efficiency after correction:")
```

```
##
##
##  Normalised Efficiency after correction:
```

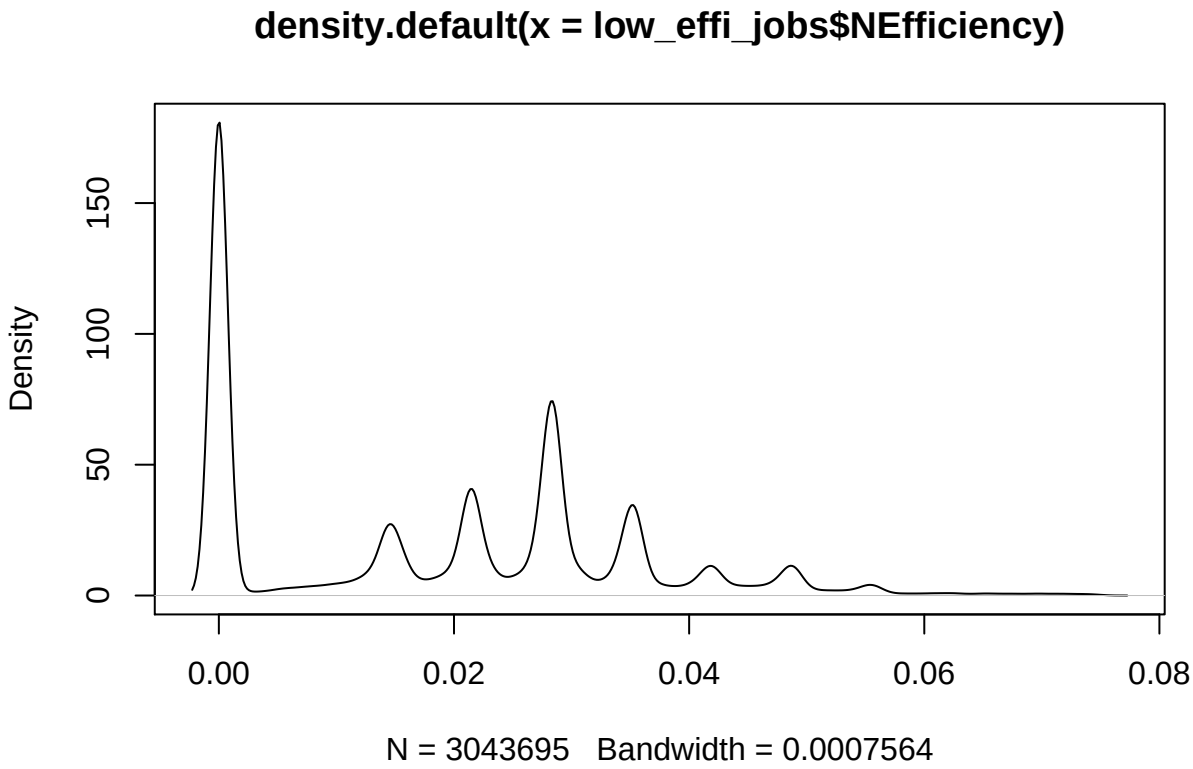
```
print(corrected_eff)
```

```
## [1] 0.7431711
```

```
plot(density(jobs$NEfficiency))
```




```
# Study of low efficiency jobs
low_effi_jobs <- subset(jobs, jobs$NEfficiency < 0.075)

plot(density(low_effi_jobs$NEfficiency))
```



```r
low_effi_jobs$TotalWallTime <- low_effi_jobs$WallTime * low_effi_jobs$RequestCpus

# Classification of low efficient jobs into 2 Classes on the basis of CPU Time

# Set of jobs that never get CPU Time
low_effi_jobs_ZeroCPU <- subset(low_effi_jobs, CPUTime == 0)
# Set of jobs with low CPU Time and high Wall time
low_effi_jobs_grt_CPUTime <- subset(low_effi_jobs, CPUTime > 0)

# Contribution based on VO Zero efficiency jobs
VO = unique(jobs$x509UserProxyVOName)

for (vo in VO){
  printf("\n\n\n************ Zero CPU Time - VO Name: %s ***************\n", vo)
  sub_Data <- subset(low_effi_jobs_ZeroCPU, x509UserProxyVOName == vo)
  printf("\nNumber of observation: %d", nrow(sub_Data))
  printf("\nPercentage of data: %f", (nrow(sub_Data)/nrow(jobs))*100)
}
```

```
##
##
##
## ************ Zero CPU Time - VO Name: None ***************
##
## Number of observation: 369583
## Percentage of data: 5.185587
##
##
## ************ Zero CPU Time - VO Name: cms ***************
##
## Number of observation: 30285
## Percentage of data: 0.424926
##
##
## ************ Zero CPU Time - VO Name: atlas ***************
##
## Number of observation: 32060
## Percentage of data: 0.449831
##
##
## ************ Zero CPU Time - VO Name: lhcb ***************
##
## Number of observation: 7750
## Percentage of data: 0.108740
##
##
## ************ Zero CPU Time - VO Name: vo.compass.cern.ch ***************
##
## Number of observation: 790
## Percentage of data: 0.011084
##
##
## ************ Zero CPU Time - VO Name: ilc ***************
##
```

```
## Number of observation: 417
## Percentage of data: 0.005851
##
##
## ************ Zero CPU Time - VO Name: alice ***************
##
## Number of observation: 590988
## Percentage of data: 8.292101
##
##
## ************ Zero CPU Time - VO Name: na62.vo.gridpp.ac.uk ***************
##
## Number of observation: 13
## Percentage of data: 0.000182
##
##
## ************ Zero CPU Time - VO Name: dune ***************
##
## Number of observation: 8
## Percentage of data: 0.000112
```

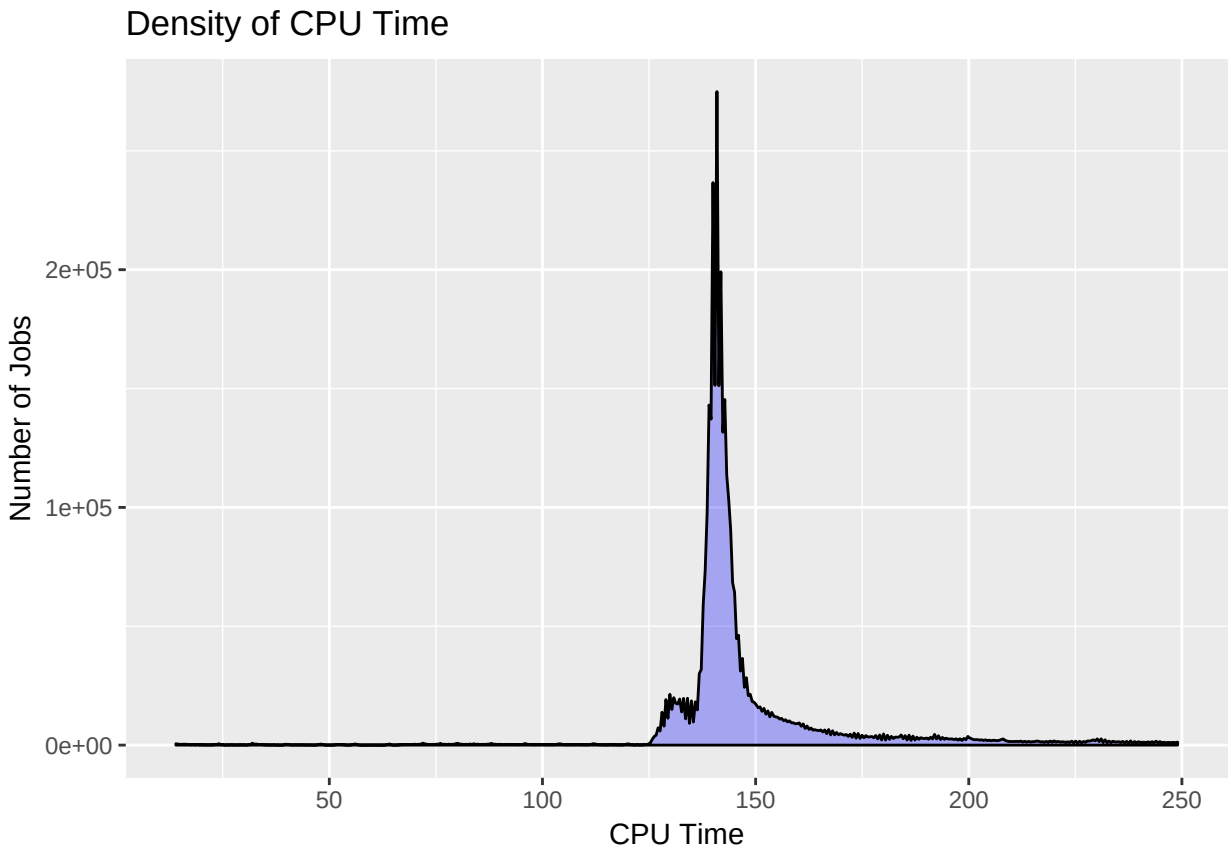
```
# Contribution based on VO low efficiency jobs
for (vo in VO){
  printf("\n\n\n************ VO Name: %s ***************\n", vo)
  sub_Data <- subset(low_effi_jobs_grt_CPUTime, x509UserProxyVOName == vo)
  printf("\nNumber of observation: %d", nrow(sub_Data))
  printf("\nPercentage of data: %f", (nrow(sub_Data)/nrow(jobs))*100)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

```
##
##
##
## ************ VO Name: None ***************
##
## Number of observation: 1
## Percentage of data: 0.000014
## Normalized Efficiency: [1] 0.01844262
##
##
##
## ************ VO Name: cms ***************
##
## Number of observation: 10758
## Percentage of data: 0.150945
## Normalized Efficiency: [1] 0.01084576
##
##
##
## ************ VO Name: atlas ***************
##
## Number of observation: 69463
## Percentage of data: 0.974629
```

```
## Normalized Efficiency: [1] 0.004512994
##
##
##
## ************ VO Name: lhcb ***************
##
## Number of observation: 8926
## Percentage of data: 0.125240
## Normalized Efficiency: [1] 0.004520315
##
##
##
## ************ VO Name: vo.compass.cern.ch ***************
##
## Number of observation: 1563614
## Percentage of data: 21.938932
## Normalized Efficiency: [1] 0.02483906
##
##
##
## ************ VO Name: ilc ***************
##
## Number of observation: 7262
## Percentage of data: 0.101892
## Normalized Efficiency: [1] 0.05029955
##
##
##
## ************ VO Name: alice ***************
##
## Number of observation: 351730
## Percentage of data: 4.935093
## Normalized Efficiency: [1] 0.03066118
##
##
##
## ************ VO Name: na62.vo.gridpp.ac.uk ***************
##
## Number of observation: 1
## Percentage of data: 0.000014
## Normalized Efficiency: [1] 0.06551298
##
##
##
## ************ VO Name: dune ***************
##
## Number of observation: 46
## Percentage of data: 0.000645
## Normalized Efficiency: [1] 0.005019873
```

```
# Peak in Total Wall Time - low efficient jobs
j5 <- subset(low_effi_jobs_grt_CPUTime, TotalWallTime < 250)

ggplot(j5, aes(x=TotalWallTime)) +
```

```
  stat_density(aes(y=..count..), color="black", fill="blue", alpha=0.3) +
  labs(title = "Density of CPU Time ", x = "CPU Time", y = "Number of Jobs")
```




```
for (vo in VO){
  printf("\n\n\n************ VO Name: %s ***************\n", vo)
  sub_Data <- subset(j5, x509UserProxyVOName == vo)
  printf("\nNumber of observation: %d", nrow(sub_Data))
  printf("\nPercentage of data: %f", (nrow(sub_Data)/nrow(jobs))*100)
}
```

```
##
##
##
## ************ VO Name: None ***************
##
## Number of observation: 0
## Percentage of data: 0.000000
##
##
## ************ VO Name: cms ***************
##
## Number of observation: 7729
## Percentage of data: 0.108445
##
##
## ************ VO Name: atlas ***************
##
```

```
## Number of observation: 6463
## Percentage of data: 0.090682
##
##
## ************ VO Name: lhcb ***************
##
## Number of observation: 4144
## Percentage of data: 0.058144
##
##
## ************ VO Name: vo.compass.cern.ch ***************
##
## Number of observation: 1424085
## Percentage of data: 19.981213
##
##
## ************ VO Name: ilc ***************
##
## Number of observation: 11
## Percentage of data: 0.000154
##
##
## ************ VO Name: alice ***************
##
## Number of observation: 276563
## Percentage of data: 3.880431
##
##
## ************ VO Name: na62.vo.gridpp.ac.uk ***************
##
## Number of observation: 0
## Percentage of data: 0.000000
##
##
## ************ VO Name: dune ***************
##
## Number of observation: 0
## Percentage of data: 0.000000
```

```
# To check the job universe. Exit code depends on Job universe. Exit code is always Zero for grid jobs.
# For others When a user job exits by means other than a signal, this is the exit return code of the us
unique(jobs$JobUniverse) # Super set of J5
```

```
## [1] 5
```

```
exit_code <- unique(j5$ExitCode)
for (ec in exit_code){
  printf("\n\n\n************ Exit Code Value: %s ***************\n", ec)
  sub_Data <- subset(j5, ExitCode == ec)
  printf("\nNumber of observation: %d", nrow(sub_Data))
  printf("\nPercentage of data: %f", (nrow(sub_Data)/nrow(jobs))*100)
}
```

```
##
##
##
```

```
## ************ Exit Code Value: 0 ***************
##
## Number of observation: 1446837
## Percentage of data: 20.300444
##
##
## ************ Exit Code Value: 137 ***************
##
## Number of observation: 268821
## Percentage of data: 3.771804
##
##
## ************ Exit Code Value: None ***************
##
## Number of observation: 3326
## Percentage of data: 0.046667
##
##
## ************ Exit Code Value: 1 ***************
##
## Number of observation: 1
## Percentage of data: 0.000014
##
##
## ************ Exit Code Value: 3 ***************
##
## Number of observation: 5
## Percentage of data: 0.000070
##
##
## ************ Exit Code Value: 127 ***************
##
## Number of observation: 5
## Percentage of data: 0.000070
```