Periodic_growth_of_df.R

jason

Mon Jan 14 10:38:57 2019

```r
# adpated from R Inferno https://www.burns-stat.com/documents/books/the-r-inferno/
#
# This script is an example of one way to overcome the issue of fragmented memory caused by 'growing ob
#
#
#
n <- as.numeric(10)
current.N <- 10 * n
my.df <- data.frame(a = character(current.N),
                    b = numeric(current.N),
                    stringsAsFactors = FALSE)
count <- 0
set.seed(58008)
for (i in 1:n){
        this.N <- rpois(1, 10)

        if (count + this.N > current.N) {
                old.df <- my.df
                current.N <- round(1.5 * (current.N + this.N))
                my.df <- data.frame(a = character(current.N),
                            b = numeric(current.N),
                            stringsAsFactors = F)
                my.df[1:count, ] <- old.df[1:count,]
                print(my.df)
        }
my.df[count + 1:this.N, ] <- data.frame(a = sample(letters,
                                                    this.N, replace = TRUE),
                                        b = runif(this.N))
count <- count + this.N
}
my.df <- my.df[1:count, ] #this operation trims the zero rows from the bottom
my.df
```

```
##     a          b
## 1   6 0.64801413
## 2   2 0.04366762
## 3   3 0.48040822
## 4   1 0.42206178
## 5   5 0.90812082
## 6   2 0.77015892
## 7   4 0.79140401
## 8   1 0.05346619
## 9   6 0.44667432
## 10  8 0.49053550
## 11  3 0.83089149
## 12  5 0.29899754
## 13  9 0.19913514
## 14  7 0.47829378
## 15  4 0.44364721
## 16  2 0.86301979
## 17  6 0.39113905
## 18  4 0.51590165
## 19  2 0.96513782
## 20  2 0.11371298
## 21  1 0.65068486
## 22  3 0.91074943
## 23  6 0.82118152
## 24  5 0.90568649
## 25  1 0.55378980
## 26  2 0.52428599
## 27  3 0.14103984
## 28  5 0.80584355
## 29  4 0.82329404
## 30  3 0.97114221
## 31  2 0.37256942
## 32  4 0.96468122
## 33  7 0.36004371
## 34  5 0.09276328
## 35  6 0.37750584
## 36  3 0.15017776
## 37  1 0.40081375
## 38  5 0.13548491
## 39  3 0.69106309
## 40  4 0.78927234
## 41  7 0.95818570
## 42  6 0.82635954
## 43  8 0.44558268
## 44  4 0.34659587
## 45 10 0.23460693
## 46  1 0.57090711
## 47  8 0.53801591
## 48  2 0.78816427
## 49  6 0.91450563
## 50  9 0.30497877
## 51  1 0.79220645
## 52  2 0.94722950
## 53  3 0.33908431
## 54  2 0.15528266
## 55  4 0.83520241
## 56  9 0.70175676
## 57  8 0.06129689
## 58  5 0.39433048
## 59  1 0.30217609
## 60  6 0.28251845
## 61  7 0.57349477
## 62  6 0.49657262
## 63  8 0.79972108
## 64  3 0.95627945
## 65  3 0.59886888
## 66  2 0.85920856
## 67  1 0.10914945
## 68  3 0.15933006
## 69  6 0.67720759
## 70  5 0.73743137
## 71  1 0.99194860
## 72  4 0.99172174
## 73  1 0.86793250
## 74  3 0.73559785
## 75  2 0.67818143
## 76  7 0.94993673
## 77  6 0.64223726
## 78  5 0.29313204
```