Package 'lazybar'

April 28, 2020

Type Package

Title Progress Bar with Remaining Time Forecast Method

Version 0.1.0

Description A simple progress bar showing estimated remaining time. Multiple forecast methods and user defined forecast method for the remaining time are supported.

License GPL-3

Encoding UTF-8

LazyData true

URL https://pkg.yangzhuoranyang.com/lazybar/, https://github.com/FinYang/lazybar/

BugReports https://github.com/FinYang/lazybar/issues/

Imports R6

Suggests forecast

RoxygenNote 7.1.0

Language en-AU

NeedsCompilation no

Author Yangzhuoran Yang [aut, cre] (<https://orcid.org/0000-0002-1232-8017>)

Maintainer Yangzhuoran Yang <Fin.Yang@monash.edu>

Repository CRAN

Date/Publication 2020-04-28 11:20:02 UTC

R topics documented:

`lazyProgressBar` *Progress bar with customisable estimated remaining time*

Description

Display a progress bar displaying the estimated time. The purpose of having various estimation methods is to provide a more accurate estimation when the run time between ticks is assumed to be different, e.g., online estimation, time series cross validation, expanding window approach, etc.

Usage

```
lazyProgressBar(n, method = "average", fn = NULL, ...)
```

Arguments

`n`	Integer. Total number of ticks
`method`	Character. The embedded forecasting method of remaining time: `drift` (default), `average`, `naive`, or `snaive`. Ignored if `fn` is not `NULL`.
	`average` **(default)** Average method. The run time between future ticks are assumed to be the average run time of the past ticks. This is the most common estimation method for remaining time.
	`drift` Drift method. The run time between future ticks are assumed to increase (decrease), and the level changed is set to be the average change of the run time of the past ticks. This is to assume the run time between ticks is linearly increasing or decreasing.
	`naive` Naive method. The run time between future ticks are assumed to be the run time between the last two ticks,
	`snaive` Seasonal naive method. If this method is chosen, an argument of `s` needs to be supplied in the `...`. The run time between future ticks is set to be the run time `s` times before. By default `s` is set to be 1/10 of the total number of ticks.
`fn`	Function. User defined function to estimate the remaining time. The function should predict the remaining time using the arguments and return a scalar. It should have at least three arguments in the order of `dtime`, `i`, and `n`, which represent the status of the progress bar at the current tick:
	`dtime` A numeric vector of the run time between past ticks.
	`i` The number of the current tick.
	`n` The number of total ticks.
`...`	Other arguments to pass to estimation method. The arguments need to be named.

Details

Four simple forecasting methods are available for the estimation of the remaining time: Average method (default), Drift method, Naive method and Seasonal naive method. For the summary of the simple methods, see Chapter 3 of `References`. User can also supply their customised estimation method as a function. See `Arguments` and `Examples`.

Value

An R6 object with methods `tick()` and `print()`.

Author(s)

Yangzhuoran Fin Yang

References

Hyndman, R.J., & Athanasopoulos, G. (2018) Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2. Accessed on 24/04/2020.

Examples

```
pb <- lazyProgressBar(4)
pb$tick()
pb$tick()
pb$tick()
pb$tick()

# With linearly increasing run time
pb <- lazyProgressBar(4, method = "drift")
for(i in 1:4){
  Sys.sleep(i * 0.2)
  pb$tick()$print()
}

# With user defined forecast function
# The forecast function itself will
# require certain computational power
forecast_fn <- function(dtime, i, n, s = 10){
  # When the number of ticks is smaller than s
  # Estimate the future run time
  # as the average of the past
  if(i<s){
    eta <- mean(dtime)*(n-i)
  }

  # When the number of ticks is larger than s
  # Fit an arima model every s ticks
  # using forecast package
  if(i>=s){
    if(i %% s ==0){
      model <- forecast::auto.arima(dtime)
    }
    runtime <- forecast::forecast(model, h=n-i)$mean
    if(i %% s !=0){
      runtime <- runtime[-seq_len(i %% s)]
    }
    eta <- sum(runtime)
  }
```

```
  return(eta)
}

pb <- lazyProgressBar(10, fn = forecast_fn, s=3)
for(i in 1:10){
  Sys.sleep(i * 0.2)
  pb$tick()$print()
}
```

Index