Stat 341, Homework 1

Ryan Sheehan & Brad Smallwood

2017-01-25

`r knitr::opts_chunk$set(echo <- TRUE)`

We will work with the `gapminder` data and will use functions from `dplyr` for this homework. The `gapmider` dataset is a `tbl_df` data structure, which we will coerce to a `data.frame`.

1. (1 mark) Write a code chunk that uses the basic `[`, `[[` or `$` operators to extract all data from Canada and the United States and saves them as a new dataset called `CanUS1`.

Answer

```
View(gapminder)
#1609-1620 USA, 241- 252 Canada
head(gapminder)
Can1 <- gapminder[241:252,c("country", "continent", "year", "lifeExp", "pop", "gdpPercap")]
Can1

US1 <- gapminder[1609:1620, c("country", "continent", "year", "lifeExp", "pop", "gdpPercap")]
US1

CanUS1 <- rbind(Can1, US1)
CanUS1
```

2. (3 marks) Repeat the subsetting in (1) with the `filter()` function from `dplyr` to create a dataset `CanUS2`. Verify column-by-column that all elements of `CanUS1` and `CanUS2` equal using the `all.equal()` function and a `for` loop over columns. What difference does `all.equal(CanUS1,CanUS2)` report?

Answer

```
Can2 <- filter(gapminder, country == "Canada")
US2 <- filter(gapminder, country == "United States")
CanUS2 <- rbind (Can2, US2)

dim(CanUS1)
dim(CanUS2)

test=numeric(ncol(CanUS1))
for(i in 1:ncol(CanUS1)){
  test[i]<-all.equal(CanUS1[,i],CanUS2[,i])
}
View(test)
```

3. (2 marks) Extract the columns `year`, `lifeExp`, `pop` and `gdpPercap` and save this dataset as `gm2` (1 mark). Also coerce `gm2` to a matrix and save as `gm3` (1 mark).

Answer

```
gm2 <- data.frame(CanUS1$year, CanUS1$lifeExp, CanUS1$pop, CanUS1$gdpPercap)
gm2
gm3 <- as.matrix(gm2)
```

4. (2 marks) Create a larger dataset by stacking `gm2` `n<-100` times over. That is, if `nrg` is the number of rows of `gm2` and `ncg` is the number of columns, the larger dataset should have `100*nrg` rows and `ncg` columns. Call your stacked dataset `biggm2`. To create the stacked dataset, initialize with `biggm2 <- NULL` and use a `for` loop to build up `biggm2` one layer at a time. Time this code using the `system.time()` function. An example use of `system.time()` to time an R command, e.g., `x <- rnorm(100000)` is:

```
system.time({
  x <- rnorm(100000) # Could put multiple lines of R code here
})
```

Use the first element of the output (`user` time) as your measure of execution time.

Answer

```
biggm2 <- NULL
system.time(
  for(i in 1:100){
    biggm2<-rbind(biggm2,gm2)
  }
)

biggm2
gm2
```

5. (2 marks) Repeat (4) to create `biggm3` by stacking `gm3` 100 times, and compare the timings for constructing `biggm2` versus `biggm3`.

Answer

```
biggm3 <- NULL
system.time(
  for(i in 1:100){
    biggm3 <- rbind(biggm3,gm3)
  }
)
biggm3
#Making biggm3 was slightly faster at user: 0.02 < 0.08, system: 0.00 = 0.00, elapsed: 0.01 < 0.
08
```

6. (3 marks) Now build `biggm3` by (i) initializing an empty matrix of appropriate dimension, and (ii) looping 100 times and inserting `gm3` into successive layers of `biggm3`. Time this code and compare the timing to that of part (5). You may find the following R function useful:

```
layerInds <- function(layerNum,nrow) {
  ((layerNum-1)*nrow + 1):(layerNum*nrow)
}
# Example use:
inds <- layerInds(layer<-1,nrow<-nrow(gapminder))
range(inds)
inds
nrow(gapminder)
ncol(gapminder)
```

Answer

```
biggm3 <- matrix(, nrow <- nrow(gm3)*100, ncol <- ncol(gm3))
system.time(
  for(i in 0:99){
      for(k in 1:nrow(gm3)){
        biggm3[(i*nrow(gm3)) + k, ] <- gm3[k, ]
      }
  }
)
biggm3

#user  system elapsed
#0.03    0.00    0.04
```

7. (3 marks) Write a function called `stackmat` that implements the faster of (5) and (6) for stacking matrices. The function should:

- Take a matrix `mat` as input and the number `nstack` of times it is to be stacked, with default number `nstack<-1` .
- Test whether `mat` is a matrix; if not, stop execution and issue an error message. For the error, use the `stop()` function, as in `stop("argument` mat `must be a matrix")`
- Stack `mat` `nstack` times and return the result.

Test that your function can replicate the `biggm3` matrix you created earlier.

Answer

```
stackmat <- function(mat, nstack){
  if(is.matrix(mat) == FALSE){
    stop("argument `mat` must be a matrix")
  }
  if(nstack < 1 || is.na(nstack)){
    nstack <- 1
  }

outputmat <- NULL
  for(i in 1:nstack){
    outputmat <- rbind(outputmat,mat)
  }
  return(outputmat)
}

test_Matrix <- stackmat(mat <- gm3, nstack <- 100)
test_Matrix
biggm3
```