R-Codes

Junhuang Xue

October 16, 2018

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

penissssssssssssssssssssssssss

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:

```
##############################################################################
#                                                                            #
# Simulation                                                                 #
#                                                                            #
##############################################################################
# general hints cor coding:
# monitor time each chunk code took,
# dont have to optimise chunks that used insignificant amount of time
# use for loop isf it doesnt take too long,
# use interaction function
# debug and testing codes are contained within an empty  functions
#
# V1: rand.day.time
#     simdata (sim.number, time.start, time.end, cat1n, cat2n)
#     check simulation with cumulative plot and count plot
#
# V2: Fixed the bug that caused as.POSIXct to outout different time zone
#     simdata (cat2.val:Able to add custome matrix as terminal roots)
#
# V3: rand.day
#     simdata(added leaf variable, various debugging, cleanning)
#     contain test and debug codes in an empty function
#     cumdata
#     tabulatedata
```

```
#
# Todo
# V4: fig bug in tabulatedata
#     anomaly
#
#packages
#qpcr
#minpack.lm
#########################################################################################################

### Function 1 ###

###Create a number of random times

# rand.day.time originally by Dirk Eddelbuettel 2012
# Debugged by Thomas Lumley on 28 Oct 2018
# https://stackoverflow.com/questions/14720983/efficiently-generate-a-random-sample-of-times-and-dates-

rand.day.time <- function(N, st="2006/01/01 00:00:01", et="2018/12/31 23:59:59") {
         st <- as.POSIXct(strptime(st, format="%Y/%m/%d %H:%M:%S", tz="Pacific/Auckland"))
         et <- as.POSIXct(strptime(et, format="%Y/%m/%d %H:%M:%S", tz="Pacific/Auckland"))
       dt <- as.numeric(difftime(et, st, unit="sec"))
       ev <- sort(runif(N, 0, dt))
       rt <- st + ev
}

### testing
rand.day.time.test <- function(){

time.start = "2006/01/01 00:00:01"
time.end = "2018/12/31 23:59:59"
print(rand.day.time(5,time.start,time.end))

# system.time(rand.day.time(1000))
# system.time(rand.day.time(1000000))

test0 = rand.day.time (100000, time.start, time.end)
head(test0)
tail(test0)

t.char = as.character(test0)                          # it is taking a while
t.day = as.character(as.Date(t.char, units = "days"))  # it is taking a while
dailycount = tabulate(as.factor(t.day))

plot(dailycount,type="l", col="blue")
abline(h=mean(dailycount), col="red", lwd=2)
points(dailycount[1], pch=19, col="red")
points(length(dailycount), dailycount[length(dailycount)], pch=19, col="red")

}
#########################################################################################################

### Function 2 ###
```

```r
### Create random day

rand.day <- function(N = 1, st = "2006/01/01 00:00:01", et = "2018/12/31 23:59:59"){
day = as.character(as.Date(rand.day.time(N, st, et)), units = "days")
return(day)
}

### testing
rand.day.test <- function(){

rand.day()
rand.day(1)
str(rand.day(1))

rand.day(2)
length(rand.day(2))

}
##########################################################################################################################

### Function 3 ###

### Function to simulate raw data

# 1,000 to 1,000,000 simulation recommended
# time consuming parts
   # create id
   # as.character(as.date(time))
   # ifelse to assign leaf dummies

#####################################################

### codes for debugging simdata
simdata.debug <- function(){

sim.number = 1000000
time.start = "2006/01/01 00:00:01"
time.end = "2018/12/31 23:59:59"

cat2.val = qpcR:::cbind.na(
  c(2,2),
  c(2,2))

cat2.val = qpcR:::cbind.na(
  c(2,2),
  c(2))

cat2.val = qpcR:::cbind.na(
  c(200,200,100),
  c(200,200,  0),
  c(100,  0,  0))

cat2.val
```

```r
str(cat2.val)
}
#####################################################

simdata <-function(sim.number = 1000000,                    # number of simulations
                   repeats =1,                              # how many time we run simulation and take m
                                                            # how to do this????
                   time.start = "2006/01/01 00:00:01",      # start and end of period
                   time.end = "2018/12/31 23:59:59",
                   cat2.val = qpcR:::cbind.na(              # number in leafs, sum to 1000
                     c(250,250),                            # Use 0 for empty values in matrix
                     c(250,250))){

###error checking

  if (sim.number > 10000000)
    stop("too many simulations, may be too slow")
  if (any(is.na(as.vector(cat2.val))) == T)
    stop("cat2.val should not contain NA, replace NA with 0")

#  x=as.vector(cat2.val)
#  if (!is.numeric(x) || !all(is.finite(x) || x < 0))
#    stop("invalid matrix values, use real numbers")

### identifiers/key variables

  index = 1:sim.number
  EventID = sprintf("%08d", sample(1:paste(rep(9,8), collapse=""),
                    sim.number,replace=F))

### select random times
#
rand.day.time  <- function(N, st=time.start, et = time.end) {
    st <- as.POSIXct(strptime(st, format="%Y/%m/%d %H:%M:%S", tz="Pacific/Auckland"))
    et <- as.POSIXct(strptime(et, format="%Y/%m/%d %H:%M:%S", tz="Pacific/Auckland"))
    dt <- as.numeric(difftime(et, st, unit="sec"))
    ev <- sort(runif(N, 0, dt)) #<------------------------------ add seasonality here???
    rt <- st + ev
}

time = rand.day.time(sim.number, time.start, time.end)
time.char = as.character(time)
time.day = as.character(as.Date(time.char, units = "days"))

#-----------------------------------------------------------#
### simulate level 1 and 2 brunch using terminal root matrix

cat1.val = colSums(cat2.val, na.rm=T)
cat2.tot = sum(colSums(!is.na(cat2.val)))

cat1.n = length(cat1.val)
cat2.n = nrow(cat2.val)
```

```
cat1.let = c(LETTERS[1:cat1.n])
cat2.let = c(LETTERS[1:cat2.n])

sim.cat1 = rep(cat1.let, cat1.val)
cat1 = sample(sim.cat1 , sim.number, replace = TRUE)
cat2 = rep(NA,sim.number)
tabulate(as.factor(sim.cat1))
tabulate(as.factor(cat1))

Category = cbind(cat1, cat2)
head(Category)

for(i in 1:cat2.n){
  Category[which(Category[,1] == LETTERS[i]),2] = sample(rep(cat2.let, cat2.val[,i]),
          sum(Category[,1] == LETTERS[i]), replace = TRUE)
}
head(Category)
table(Category[,1], Category[,2])

Category = as.data.frame(Category)
Category$leaf = paste(Category$cat1, Category$cat2, sep = "")
head(Category)

###assign name and value to dummy variable, name same as value

cat1.name = cat1.let
cat1.name
cat2.name = levels(interaction(cat1.let, cat2.let, sep = "", lex.order = TRUE))
cat2.name
names = c(cat1.name, cat2.name)
names

substr(cat2.name,1,1)
substr(cat2.name,2,2)

for(i in 1:(cat1.n+cat1.n*cat2.n)){
Category[,i+3] = rep(names[i],sim.number)
}
colnames(Category) = c("cat1","cat2","leaf",cat1.name,cat2.name)
head(Category)

### use ifelse to assign dummies variables

for(i in 1:cat1.n){
  Category[,i+3] = ifelse(Category$cat1 == Category[,i+3], 1, 0)
}

for(i in 1:(cat1.n*cat2.n)){
  Category[,i+3+cat1.n] = ifelse(Category$cat1 == substr(Category[,i+3+cat1.n],1,1) &
                                 Category$cat2 == substr(Category[,i+3+cat1.n],2,2), 1, 0)
}
head(Category)
```

```
### delete columns if column sum is 0

dummies = Category[-(1:3)]
cat2.val
colSums(dummies)

emptyroot = which(colSums(dummies) == 0)
emptyroot2 = which(colSums(dummies) == 2) # <------------------------fix here for no col == 0
emptyroot
emptyroot2
test1 = Category[-(emptyroot+3)]
head(test1)
test2 =  Category
head(test2)

Category = Category[-(emptyroot+3)]
head(Category)


### create dataframe

sim.data.df = data.frame(cbind(index, EventID, time.char, time.day, Category))
sim.data.df$index     = as.numeric  (sim.data.df$index)
sim.data.df$EventID   = as.character(sim.data.df$EventID)
sim.data.df$time.char = as.character(sim.data.df$time.char)
sim.data.df$time.day  = as.Date     (sim.data.df$time.day)

return(sim.data.df)
}


###Extras

#add categorical variables by random sampling?
#add contineous variables by random sampling??? using some kind of model??

##############################################################################################################

###Testing
simdata.test <- function(){

test1 = simdata (1000)
head(test1)
str(test1)
colSums(test1[,7:ncol(test1)])

v2 = cbind(                        # number in terminal roots, sum to 1000
  c(200,200,100),
  c(200,200,  0),
  c(100,  0,  0))

test2 = simdata (1000,cat2.val = v2)
head(test2)
```

```
str(test2)
colSums(test2[,7:ncol(test2)])

test3 = simdata (1000000,cat2.val = v2)
head(test3)
str(test3)

system.time(simdata (1000))
system.time(simdata (1000000))

}
#########################################################################################################################

#########################################################################################################################

# Function 5, 6

## functions to check simulation
## use ggplot 2 later if we want a good presentable plot

# create a function for tabulation
#####################################################

#debug for simulation checking codes
simcheck.debug <- function(){

v2 = cbind(
  c(200,200,100),
  c(200,200,  0),
  c(100,  0,  0))

sample1 = simdata (100000,cat2.val = v2)
head(sample1)
str(sample1)
x = sample1
}
#####################################################
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.