ESM 232 - Homework 6

David Segan, Simone Alburqueque, Siya Qiu and Mauricio Collado

5/7/2021

1. Objective

The objective of the current assignment is to build a forest size forecast model that considers changes in its growth rate according to the following thresholds: canopy closure and carrying capacity. The measuring unit for forest size is carbon.

The default values for all parameters are shown below:

```
# forest growth function
source(here("fgfx.R"))

# initial forest size
C = 10 # initial forest size of 10 kg C

# given default parameters (def means default)
K_def = 250 # carrying capacity of 250 kg C
f_def = 50 # canopy closure threshold of 50 kg C
r_def = 0.01 # exponential growth rate before canopy closure
g_def = 2 # linear growth rate after canopy closure of 2 kg/year
```

The second part of the assignment evaluates the sensitivity of the output (forest size) considering a normal distribution for all the parameters above (except initial forest size). In this case, we kept the default values as the mean, and the standard deviation is 10% of the mean value.

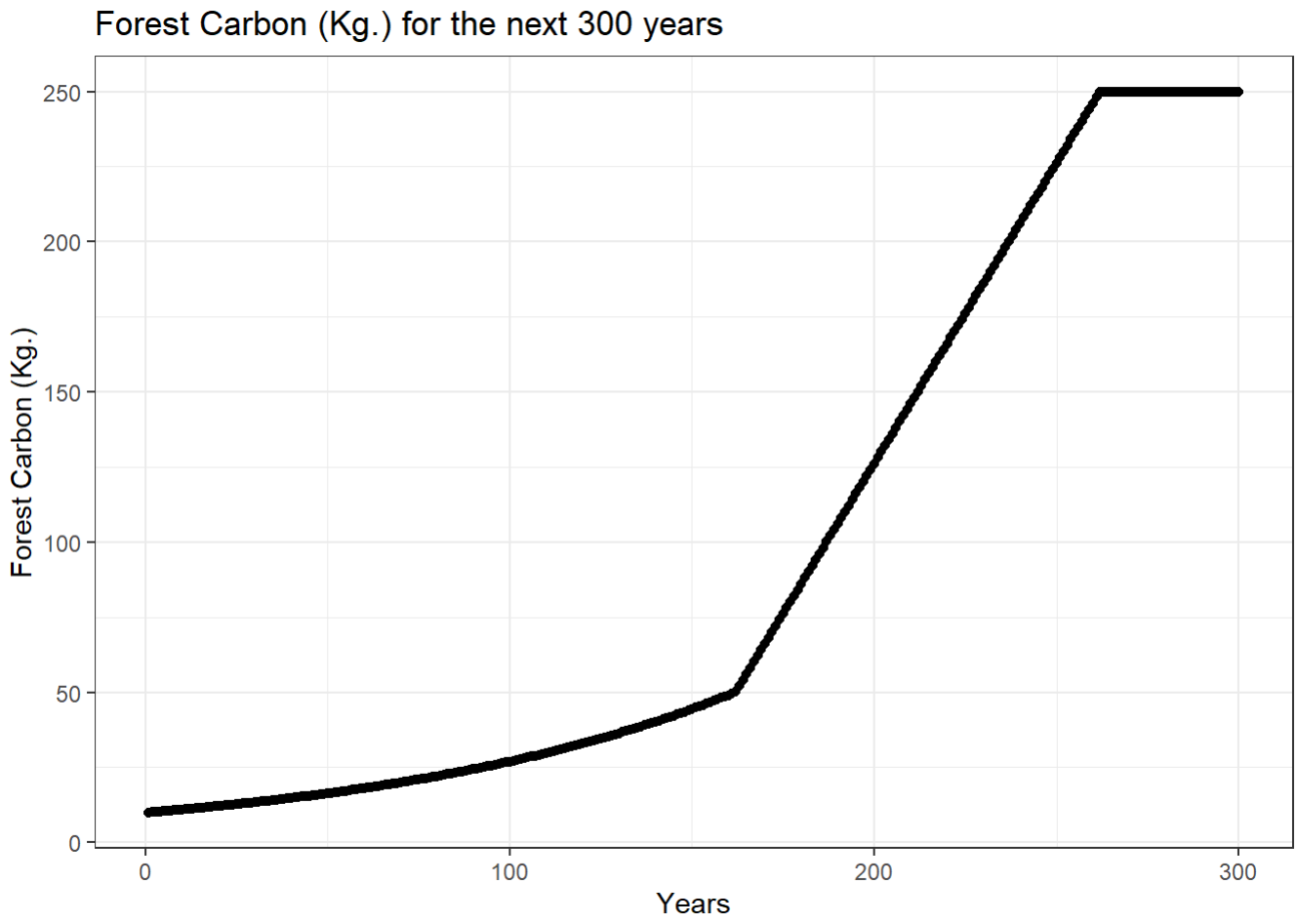
2. Forest size prediction

We evaluate our forest prediction for the next 300 years. The **Appendix 1** has tur function to estimate the forest growth.

```
# run model for 300 years and graph result

# create parameter list
initialcarbon=C
years = seq(from=1, to=300, by=1) # number of year to evaluate

# insert our default values
parms = list(r=r_def,
             K=K_def,
             f = f_def,
             g = g_def)

#apply solver
results=ode(initialcarbon, years, forest_growth, parms)

# add more meaningful names
colnames(results)=c("year","C")

# save results in a dataframe
results_df <- as.data.frame(results)
```

Our results show that around the year 160, our forest reaches the canopy closure threshold and starts a linear growth rate. Around the year 260, the forest size is stable over time (steady-state). The results are not surprising, considering that our canopy closure growth rate (g) is pretty high (2 kg/year).

```
# plot
ggplot(results_df, aes(year, C))+
  geom_point()+
  labs(title="Forest Carbon (Kg.) for the next 300 years",
       y="Forest Carbon (Kg.)",
       x="Years")+
  theme_bw()
```



3. Sobol sensitivity analysis

We perform a Sobol sensitivity analysis for the maximum value and mean value of the projected forest size considering the variability of K, r, f, and g.

In the first part, we generate the normally distributed values for our parameters and test if the ODE performs well.

```
# run a sobol sensitivity analysis

# parms are normally distributed with sd 10% of mean value

# set the number of parameters
np=100

# first set of samples
K = rnorm(mean=K_def, sd=0.1*K_def, n=np)
r = rnorm(mean=r_def, sd=0.1*r_def, n=np)
f = rnorm(mean=f_def, sd=0.1*f_def, n=np)
g = rnorm(mean=g_def, sd=0.1*g_def, n=np)

# bind the samples for each parameter
X1 = cbind.data.frame(r=r, K=K , f=f, g=g)

# repeat to get our second set of samples
K = rnorm(mean=K_def, sd=0.1*K_def, n=np)
r = rnorm(mean=r_def, sd=0.1*r_def, n=np)
f = rnorm(mean=f_def, sd=0.1*f_def, n=np)
g = rnorm(mean=g_def, sd=0.1*g_def, n=np)

# bind the samples for each parameter
X2 = cbind.data.frame(r=r, K=K , f=f, g=g)

# create our sobel object and get sets ofparameters for running the model
sens_C = soboljansen(model = NULL, X1, X2, nboot = 300)

# gets results for 300 years (evaluating every year)
simtimes = seq(from=1, to=300)

# take parameters from the sobel
parms = list(r=sens_C$X$r[1],
                   K=sens_C$X$K[1],
                   f=sens_C$X$f[1],
                   g=sens_C$X$g[1])

# verify our results
result_2 = ode(y=initialcarbon, times=simtimes, func=forest_growth, parms=parms)

# turn it into a data frame
head(result_2)
```

```
##      time        1
## [1,]    1 10.00000
## [2,]    2 10.11886
## [3,]    3 10.23914
## [4,]    4 10.36085
## [5,]    5 10.48400
## [6,]    6 10.60862
```

```
colnames(result_2)=c("year","C")
result_df2 = as.data.frame(result_2)
```

The second step is to create a function the calculates the maximum and mean value of the forest size for each simulated period. We also evaluate if it works.

```
# turn computing our metrics into a function
compute_metrics = function(result) {
 max_c = max(result$C)
 mean_c = mean(result$C)

return(list(max=max_c, mean=mean_c))}

# check if it works
compute_metrics(result=result_df2)
```

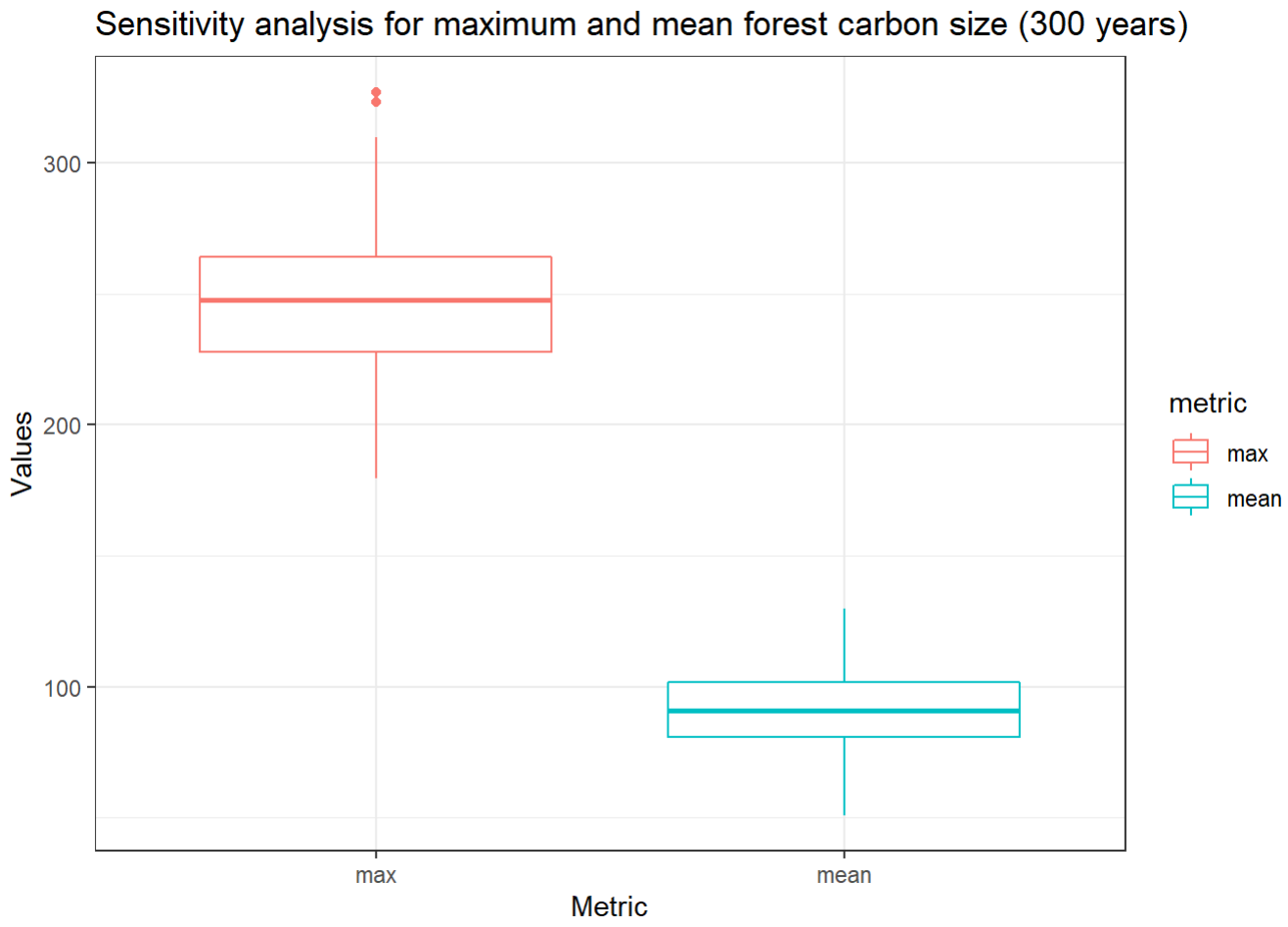
```
## $max
## [1] 246.7673
##
## $mean
## [1] 112.6709
```

Our third step is to create a wrapper function that combines our functions (forest growth and metrics) and keeps our results in a data frame. Then, we apply the new wrapper function to the parameters' values selected by our Sobol's sensitivity.

```
# define a wrapper function to do everything we need - run solver and compute metrics - and send
back results for each parameter

# our # of simulations is already defined, 300 years

c_wrapper = function(r, K, f, g, C, simtimes, func) {
    parms = list(r=r, K=K, f=f, g=g) # define parameters
    result = ode(y=C, times=simtimes, func=func, parms=parms) # rund ODE
    colnames(result)=c("year","C") # name columns
  # get metric function and keep results as a dataframe
  metrics=compute_metrics(as.data.frame(result))
  return(metrics)
}

# use pmap to run our wrapper functions for parameter value of our sensitivity analysis
allresults = sens_C$X %>% pmap(c_wrapper, C=C, simtimes=simtimes, func=forest_growth)

# extract out results from pmap into a data frame
allres = allresults %>% map_dfr(`[`,c("max","mean"))
```

Our boxplot visualization for the key metrics shows that significant difference between our maximum an average value. For all simulations, it means that it will take a long time to recover the forest size.

```
# create boxplots
tmp = allres %>% gather(key="metric", value="value")
ggplot(tmp, aes(metric, value, col=metric))+
  geom_boxplot() +
  labs(title="Sensitivity analysis for maximum and mean forest carbon size (300 years)",
       y="Values",
       x="Metric")+
  theme_bw()
```



Finally, the main effect analysis shows that the forest size has higher sensitivity to the carrying capacity (ignoring interaction with other parameters). For the total effect (taking into account the interaction with other variables), the K is also predominant. This result is logical because the K determines the “ceiling” for the forest size.

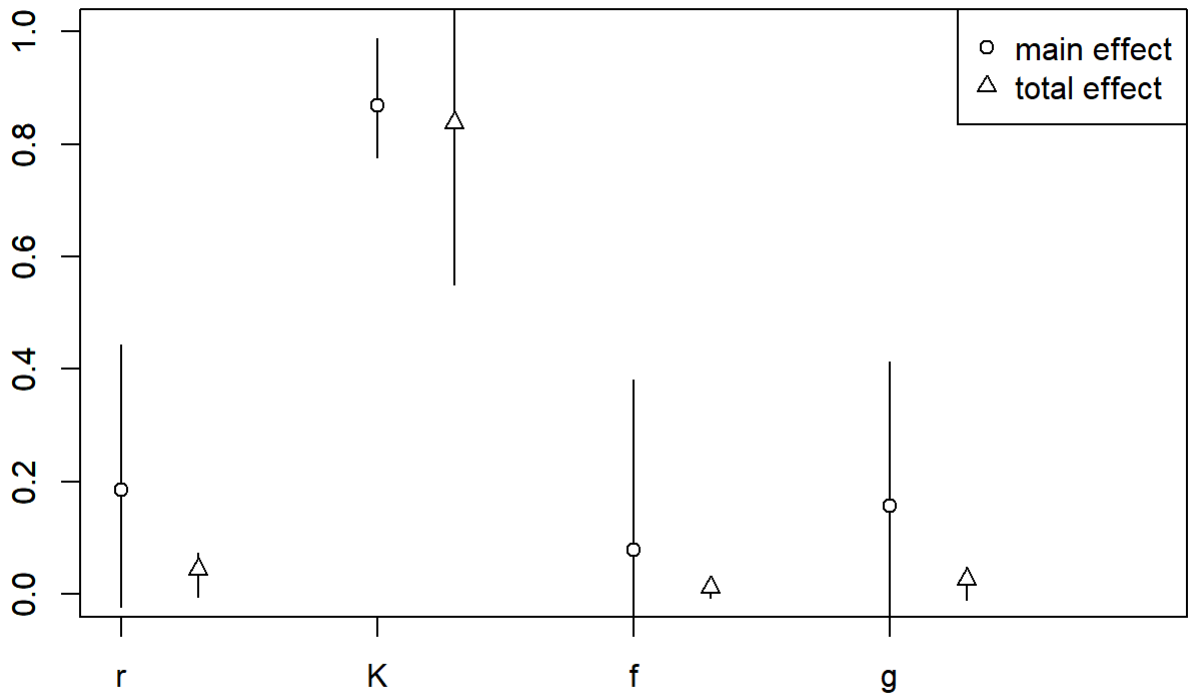
```
# sobol indice

sens_C_max = sensitivity::tell(sens_C, allres$max)

# first-order indices (main effect without co-variance)
sens_C_max$S
```

```
##     original         bias std. error   min. c.i. max. c.i.
## r 0.16120574 -0.024371834 0.11837479 -0.02261325 0.4441453
## K 0.86329617 -0.006268323 0.05129799  0.77572722 0.9876896
## f 0.05178235 -0.026127295 0.13307037 -0.14529378 0.3812502
## g 0.13095530 -0.025629289 0.12043208 -0.06189046 0.4130036
```

```
# total sensitivity index -note that this partitions the output variance - so values sum to 1
sens_C_max$T
```

```
##     original          bias  std. error    min. c.i.  max. c.i.
## r 0.04469155  1.492214e-05 0.019117299 -0.005395768 0.07383780
## K 0.86337245  2.467700e-02 0.139084573  0.551060606 1.07803793
## f 0.01255353  1.294369e-04 0.008983994 -0.006824310 0.02366631
## g 0.02695337 -5.842193e-05 0.014617129 -0.009474597 0.04874324
```

```
plot(sens_C_max)
```



Appendix 1: Forest growth function

```
#'  Forest growth
#' @param T period of growth
#' @param C size of the forest
#' @param parms$r - early exponential growth rate
#' @param parms$g - linear growth rate
#' @parms parms$f - canopy closure threshold
#' @parms parms$K - carrying capacity
#' @return change in population
#' Authors: David Segan, Mauricio Collado, Simone Alburqueque and Siya Qiu

forest_growth = function(time, C, parms) {

  # compute rate of change of forest
  dexfor = parms$r*C


  # establish the conditionals between carrying capacity and canopy closure
  dexfor =ifelse(C > parms$K, 0, # set rate of change to 0 if C > K
                 ifelse(C > parms$f, parms$g, # else, set rate of change to g if C > f
                        dexfor)) # else, default rate of change is r*C

  return(list(dexfor))
}
```