Forest Growth - ODE and Sobel Sensitivity

Anna Abelman, Margaret Brickner, & Hannah Garcia

5/6/2021

```r
# read in function
source("forest_growth.R")
```

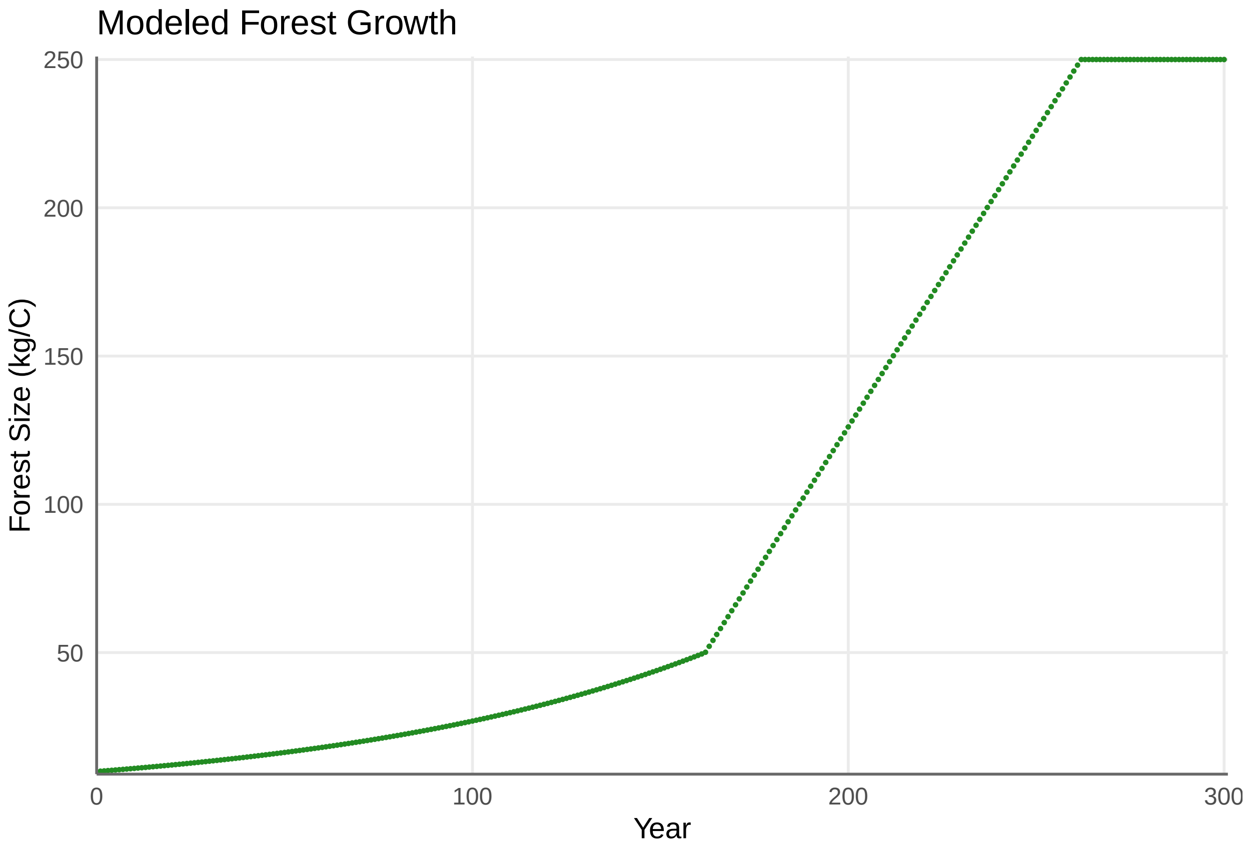
ODE for 300 years

Run the model for 300 years (using the ODE solver) starting with an initial forest size of 10 kg/C, and using the following parameters: canopy closure threshold of 50 kgC , K = 250 kg C (carrying capacity) , r= 0.01 (exponential growth rate before before canopy closure), g = 2 kg/year (linear growth rate after canopy closure)

```r
#set all of the parameters so they're easier to update and easier to track.
forest_parms = list(K=250, r = 0.01, g = 2, Ct = 50)

#set the years and initial size
years = seq(from=1, to=300)
initial_size = 10

#use ODE solver to run the model for 300 years
forest_300_result = ode(y = initial_size, times = years, func = forest_growth, parms = forest_parms)

#check out the results
colnames(forest_300_result)=c("year","size")
head(forest_300_result)
```

```
##      year     size
## [1,]    1 10.00000
## [2,]    2 10.10050
## [3,]    3 10.20202
## [4,]    4 10.30455
## [5,]    5 10.40811
## [6,]    6 10.51271
```

```r
#graph the Results
ggplot(as.data.frame(forest_300_result), aes(year, size))+
  geom_point(colour = "forestgreen", size = .3) +
  labs(title = "Modeled Forest Growth", x = "Year", y = "Forest Size (kg/C)")+
  theme_minimal() +
  theme(axis.line = element_line(color = "dimgrey"),
        panel.border = element_rect(fill = NA, color = NA),
        panel.grid.minor.y = element_blank(),
      panel.grid.minor.x = element_blank()) +
  scale_x_continuous(expand = c(0.0,1)) +
  scale_y_continuous(expand = c(0.0,1))
```



Sobel Sensitivity

Run a sobol sensitivity analysis that explores how the estimated maximum and mean forest size (e.g maximum and mean values of C over the 300 years) varies with the pre canopy closure growth rate (r) and post-canopy closure growth rate (g) and canopy closure threshold and carrying capacity(K)

Assume that they are all normally distributed with means as given above and standard deviation of 10% of mean value

```
#initial population value
Pinitial=10

#remembering the initial parameters
#forest_parms = list(K=250, r = 0.01, g = 2, Ct = 50)

#first set the number of parameters
np=100
K = rnorm(mean=250, sd=25, n=np)
r = rnorm(mean=0.01, sd=0.001, n=np)
g = rnorm(mean=2, sd=0.2, n=np)
X1 = cbind.data.frame(r=r, K=K, g=g, Ct=50)
```

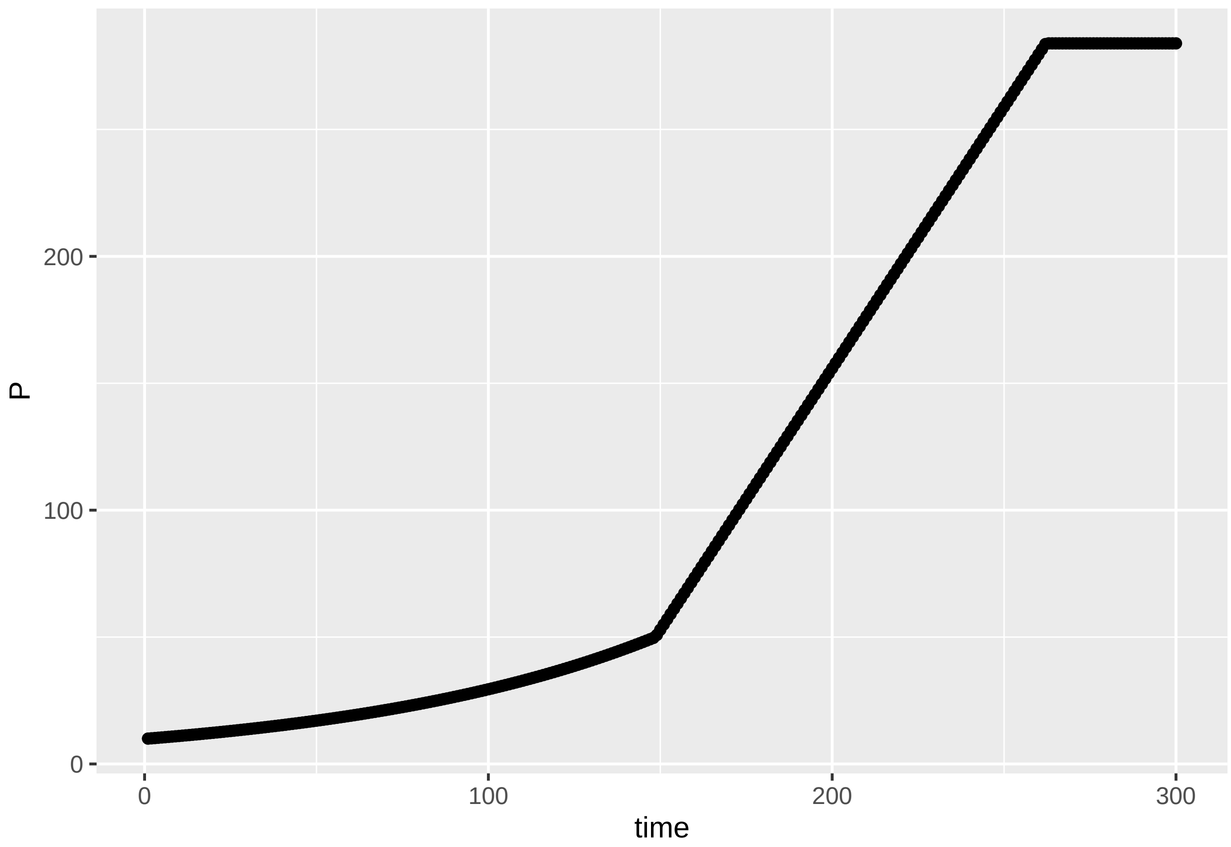
```
#second set of samples
K = rnorm(mean=250, sd=25, n=np)
r = rnorm(mean=0.01, sd=0.001, n=np)
g = rnorm(mean=2, sd=0.2, n=np)
X2 = cbind.data.frame(r=r, K=K, g=g, Ct=50)

#create sobel object that holds the 2 sample datasets
sens_parms = soboljansen(model = NULL, X1, X2, nboot = 300)

#checking out the first values
head(sens_parms$X)
```

```
##             r        K        g Ct
## 1 0.010904089 283.9155 2.060225 50
## 2 0.010547315 254.6151 1.957811 50
## 3 0.008265334 247.8220 1.996492 50
## 4 0.010818514 214.7073 1.756996 50
## 5 0.009372979 228.7451 2.332695 50
## 6 0.010768549 222.0624 1.919034 50
```

```
# gets results for each year for 300 years
times = seq(from=1, to=300) #number of simulations runs
parms = list(r=sens_parms$X$r[1], K=sens_parms$X$K[1], g=sens_parms$X$g[1], Ct=sens_parms$X$Ct[1]) #fir
forest_result = ode(y=Pinitial, times=times, func=forest_growth, parms=parms) #run ODE to get results f

#checking out first values
head(forest_result)
```

```
##      time        1
## [1,]    1 10.00000
## [2,]    2 10.10964
## [3,]    3 10.22048
## [4,]    4 10.33254
## [5,]    5 10.44582
## [6,]    6 10.56034
```

```
colnames(forest_result)=c("time","P") #change column names so easier to use

#change to a dataframe to plot
forest_result = as.data.frame(forest_result)
ggplot(forest_result, aes(time, P))+
  geom_point()
```



```
#this plot confirms our ODE results make sense so now we can begin running with the other parameters to
```

Sensitivity Analysis of Maximum Forest Size

Graph the results of the sensitivity analysis as a box plot of maximum forest size and a plot of the two sobol indices (S and T)

```
#max population from first set of parameters
max_pop = max(forest_result$P)
max_pop
```

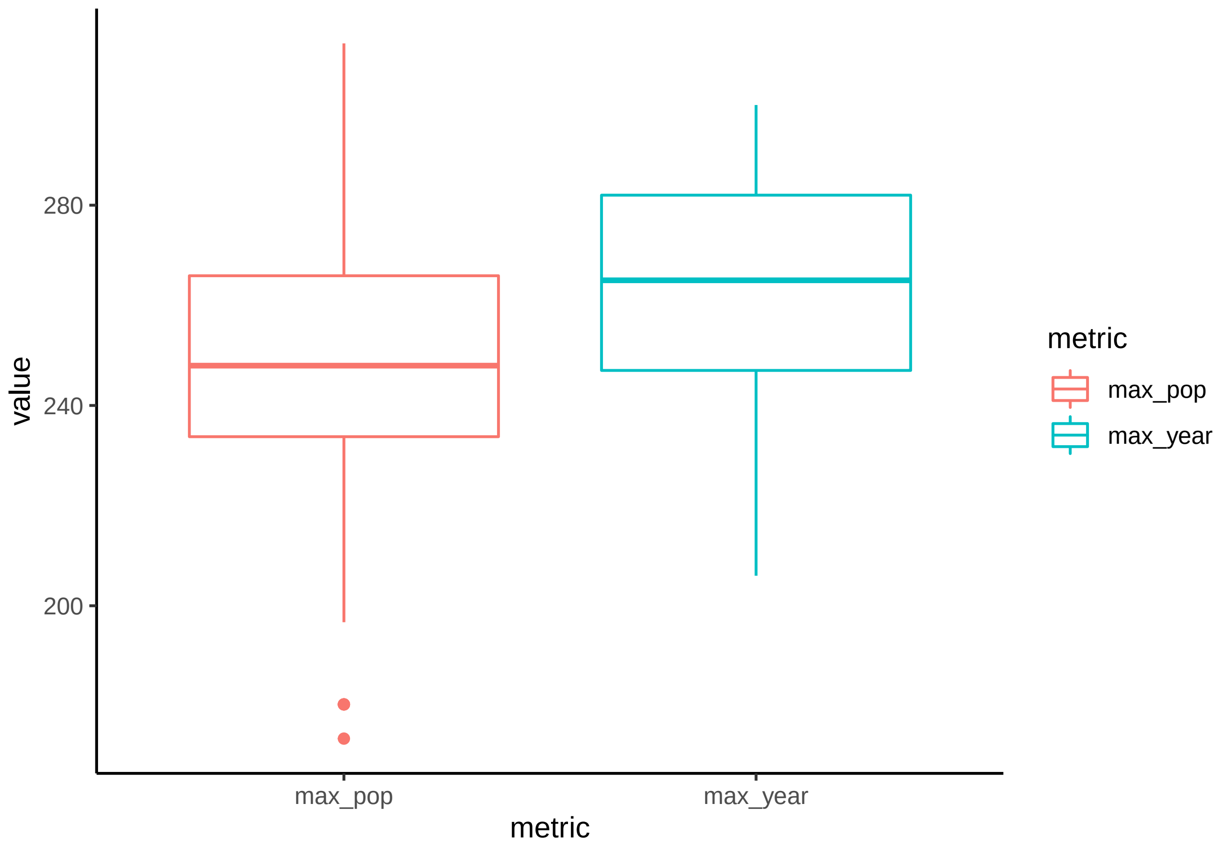
```
## [1] 283.9156
```

```
#how many years required to get to the maximum popuation
max_year = which.max(forest_result$P)
```

```
#create a function to pull out max_year and max_pop
compute_max_metrics = function(forest_result) {
  max_pop = max(forest_result$P)
  idx = which.max(forest_result$P)
  max_year = forest_result$time[idx]
return(list(max_pop=max_pop, max_year=max_year))
}
```

```
#try it on our first parameter set
compute_max_metrics(forest_result)

#create a wrapper function that will solve and return the max_pop and max_year for each set of paramete
forest_wrapper = function(r, K, g, Ct, Pinitial, times, func) {
    parms = list(r=r, K=K, g=g, Ct=Ct) #list parameters needed
    forest_result = ode(y=Pinitial, times=times, func=forest_growth, parms=parms) #solve function
    colnames(forest_result)=c("time","P") #change column names
  max_metrics=compute_max_metrics(as.data.frame(forest_result)) #find max_pop and max_year
  return(max_metrics)
}

#use pmap tp return all results for all 100 sets of parameters
all_forest_results = sens_parms$X %>%
  pmap(forest_wrapper, Pinitial=Pinitial, times=times, func=forest_growth)

#use results from above (pmap) and turn into nice dataframe
all_forest_results_df = all_forest_results %>% map_dfr(`[`,c("max_pop","max_year"))


#turn into tidy format
final_forest = all_forest_results_df %>%
  gather(key="metric", value="value")
```

```
#plot it!
ggplot(final_forest, aes(metric, value, col=metric))+
  geom_boxplot()+
  theme_classic()
```



Looking at the sensitivity for each parameter on `max_pop`

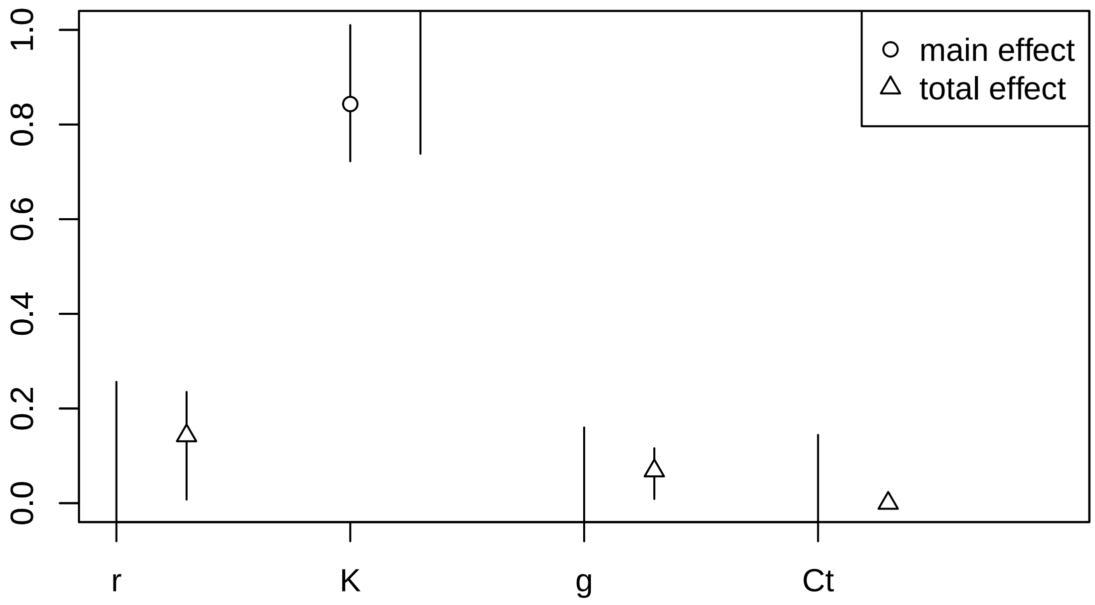
```
sens_parms_max_pop = sensitivity::tell(sens_parms,all_forest_results_df$max_pop)
#first-order indices(S)
sens_parms_max_pop$S
```

```
##      original         bias std. error  min. c.i. max. c.i.
## r  -0.1341640 -0.017145383 0.16107418 -0.3968508 0.2563122
## K   0.8409070 -0.002404924 0.07178501  0.7222841 1.0101068
## g  -0.2264189 -0.012382709 0.17434967 -0.4934881 0.1599492
## Ct -0.2410337 -0.012224688 0.17377753 -0.5120999 0.1442669
```

```
#total sensitivity index (T)
sens_parms_max_pop$T
```

```
##      original          bias std. error   min. c.i. max. c.i.
## r  0.14229229 -0.0002777412  0.0607141 0.007451146 0.2350878
## K  1.07315860  0.0177890261  0.1474811 0.738235868 1.3123207
## g  0.07042281  0.0019831531  0.0283318 0.008716293 0.1160795
## Ct 0.00000000  0.0000000000  0.0000000 0.000000000 0.0000000
```

```
#plot to see the difference
plot(sens_parms_max_pop)
```



Looking at the sensitivity for each parameter on `max_year`

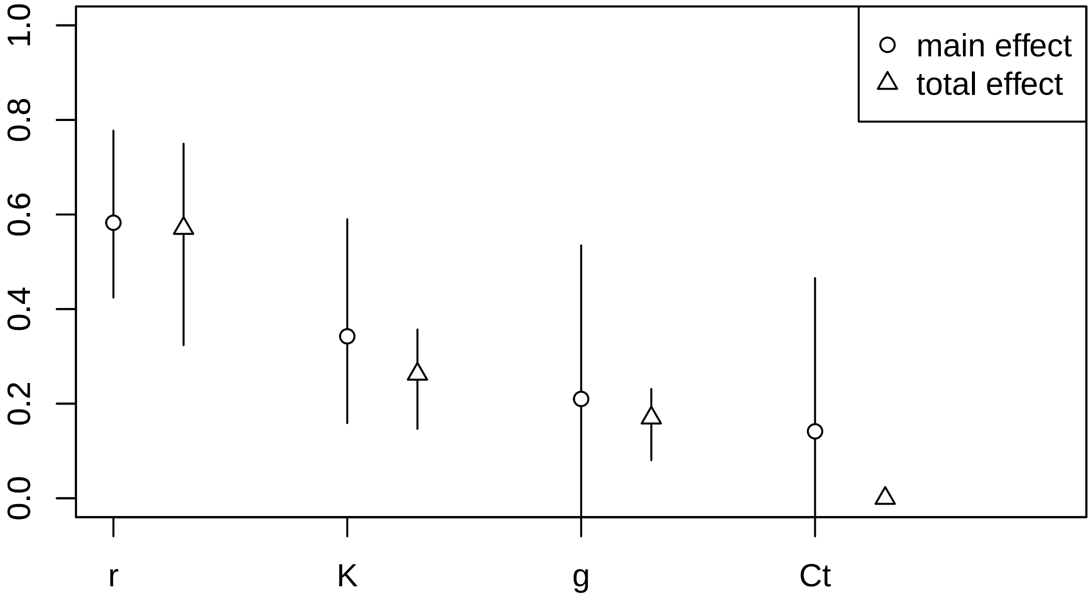
```
sens_parms_max_year = sensitivity::tell(sens_parms,all_forest_results_df$max_year)
#first-order indices (S)
sens_parms_max_year$S
```

```
##     original        bias std. error   min. c.i. max. c.i.
## r  0.5658314 -0.01678753 0.08615606  0.42430336 0.7774243
## K  0.3278000 -0.01458798 0.11296990  0.15890784 0.5900181
## g  0.1883102 -0.02168581 0.14527379 -0.05717011 0.5348569
## Ct 0.1185351 -0.02294119 0.14236213 -0.11667120 0.4655453
```

```
#total sensitivity index (T)
sens_parms_max_year$T
```

```
##     original        bias std. error  min. c.i. max. c.i.
## r  0.5926150 0.021750843 0.10766171 0.32372060 0.7496982
## K  0.2704255 0.007916714 0.05099959 0.14657418 0.3568815
## g  0.1738449 0.003973078 0.03768578 0.08034908 0.2311454
## Ct 0.0000000 0.000000000 0.00000000 0.00000000 0.0000000
```

```
#plot to see the difference
plot(sens_parms_max_year)
```


1.0
0.8
0.6
0.4
0.2
0.0
○ main effect
△ total effect
r
K
g
Ct