

Tree growth

Elmera Azadpour, Grace Lewin and Desik Somasundaram

```
source(here("R", "treegrowthcalc.R"))
```

```
treegrowthcalc
```

```
## function (time, Cinitial, parms)
## {
##   if (Cinitial < parms$thresh) {
##     treegrowth = parms$r * Cinitial
##   }
##   else if (Cinitial >= parms$thresh) {
##     treegrowth = parms$g * (1 - (Cinitial/K))
##   }
##   treegrowth = ifelse(Cinitial > parms$K, 0, treegrowth)
##   return(list(treegrowth))
## }
```

```
# lets start with sobel
library(sensitivity)

# come up with first set of sample parameters
# we will assume that we know the initial population,

Cinitial=10
thresh = 50
# want to learn about sensitivity to growth rate (r) and carrying capacity
# 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)
thresh = thresh
X1 = cbind.data.frame(r=r, K=K, g=g, thresh=thresh)

# repeat to get our 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)
thresh = thresh
X2 = cbind.data.frame(r=r, K=K, g=g, thresh=thresh)

# create our sobel object and get sets of parameters for running the model

sens_C = sobolSalt(model = NULL, X1, X2, nboot = 300)
```

```
# our parameter sets are
head(sens_C$X)
```

```
##           [,1]      [,2]      [,3] [,4]
## [1,] 0.008246684 227.6487 1.987210 50
## [2,] 0.008877500 236.9182 1.511347 50
## [3,] 0.009841286 255.4445 1.673317 50
## [4,] 0.010865544 254.0898 1.905075 50
## [5,] 0.007967250 268.7551 1.991064 50
## [6,] 0.008929733 229.3996 1.451742 50
```

```
# lets add names
colnames(sens_C$X) = c("r", "K", "g", "thresh")
```

```
# run our differential equation and keep the output
# BUT
```

```
# what output do we want to keep
```

```
# how about maximum population if we run the model for 200 years, and how many years to get to the carry
```

```
# for illustration lets look at running just one parameter sets and summarizing results
sens_C$X[1,]
```

```
##           r           K           g      thresh
## 8.246684e-03 2.276487e+02 1.987210e+00 5.000000e+01
```

```
# recall ODE needs ALL of our parameters in a single list
# initial population and times for which we want output
Cinitial
```

```
## [1] 10
```

```
# gets results for 200 years (evaluating every year)
```

```
simtimes = seq(from=1, to=300)
```

```
parms = list(r=sens_C$X[1,"r"], K=sens_C$X[1,"K"], g=sens_C$X[1,"g"], thresh=sens_C$X[1,"thresh"])
```

```
# or
```

```
#parms = list(r=as.data.frame(sens_C$X)$r[1], K=as.data.frame(sens_C$X)$K[1], g=as.data.frame(sens_C$X)$g[1], thresh=as.data.frame(sens_C$X)$thresh[1])
```

```
result = ode(y=Cinitial, times=simtimes, func=treegrowthcalc, parms=parms)
```

```
head(result)
```

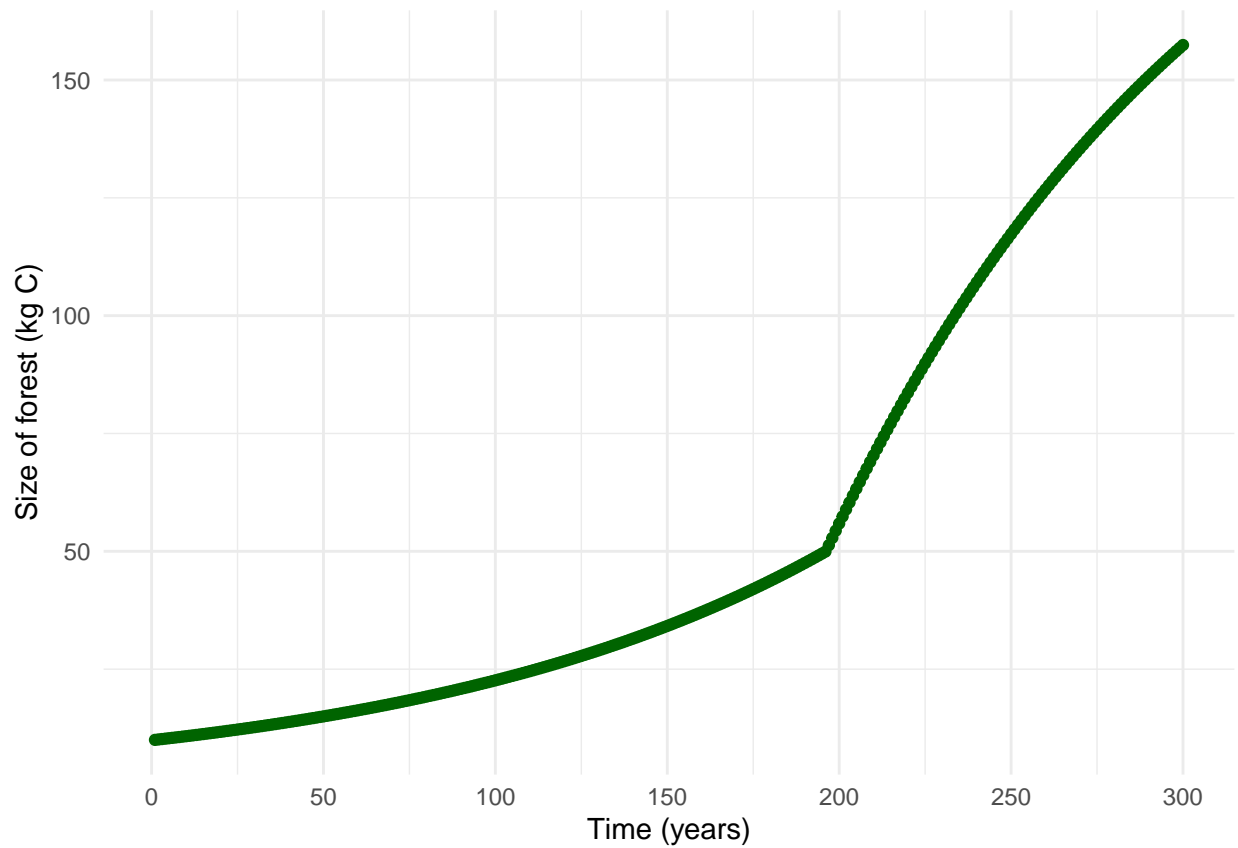
```
##      time      1
## [1,] 1 10.00000
## [2,] 2 10.08281
## [3,] 3 10.16630
## [4,] 4 10.25049
## [5,] 5 10.33537
## [6,] 6 10.42095
```

```

colnames(result)=c("time","C")
# turn it into a data frame
result = as.data.frame(result)

ggplot(result, aes(time, C))+
  geom_point(color = "dark green") +
  labs(x = "Time (years)", y = "Size of forest (kg C)") +
  theme_minimal() +
  scale_x_continuous(breaks = seq(from = 0, to = 300, by = 50)) +
  scale_y_continuous(breaks = seq(from = 0, to = 250, by = 50))

```



```

# extra metrics of interest from this
# maximum population it gets to
maxsize = max(result$C)
maxsize

```

```
## [1] 157.4126
```

```

# mean population
meansize = mean(result$C)
meansize

```

```
## [1] 55.05547
```

Compute our metric for all the parameter sets

What if we want to run for all parameters

Lets create two additional functions that will help us

- a function that computes the metrics we want
- a function that runs our ode solver and computes the metrics (I call it a wrapper function as it is really just a workflow/wrapper to call ode solver and then compute metrics)

```
# turn computing our metrics into a function
```

```
compute_metrics = function(result) {  
  maxsize = max(result$C)  
  meansize = mean(result$C)  
  return(list(maxsize=maxsize, meansize=meansize))  
}
```

```
# try it on our first parameter set
```

```
compute_metrics(result)
```

```
## $maxsize  
## [1] 157.4126  
##  
## $meansize  
## [1] 55.05547
```

```
# great but we need to apply the ode and this function for all of our parameters
```

```
# define a wrapper function to do everything we need - run solver and compute metrics - and send back r
```

```
p_wrapper = function(r, K, g, thresh, Cinitial, simtimes, func) {  
  parms = list(r=r, K=K, g=g, thresh=thresh)  
  result = ode(y=Cinitial, times=simtimes, func=func, parms=parms, method="daspk")  
  colnames(result)=c("time", "C")  
  # get metrics  
  metrics=compute_metrics(as.data.frame(result))  
  return(metrics)  
}
```

```
# now use pmap as we did before
```

```
allresults = as.data.frame(sens_C$X) %>% pmap(p_wrapper, Cinitial=Cinitial, simtimes=simtimes, func=tre
```

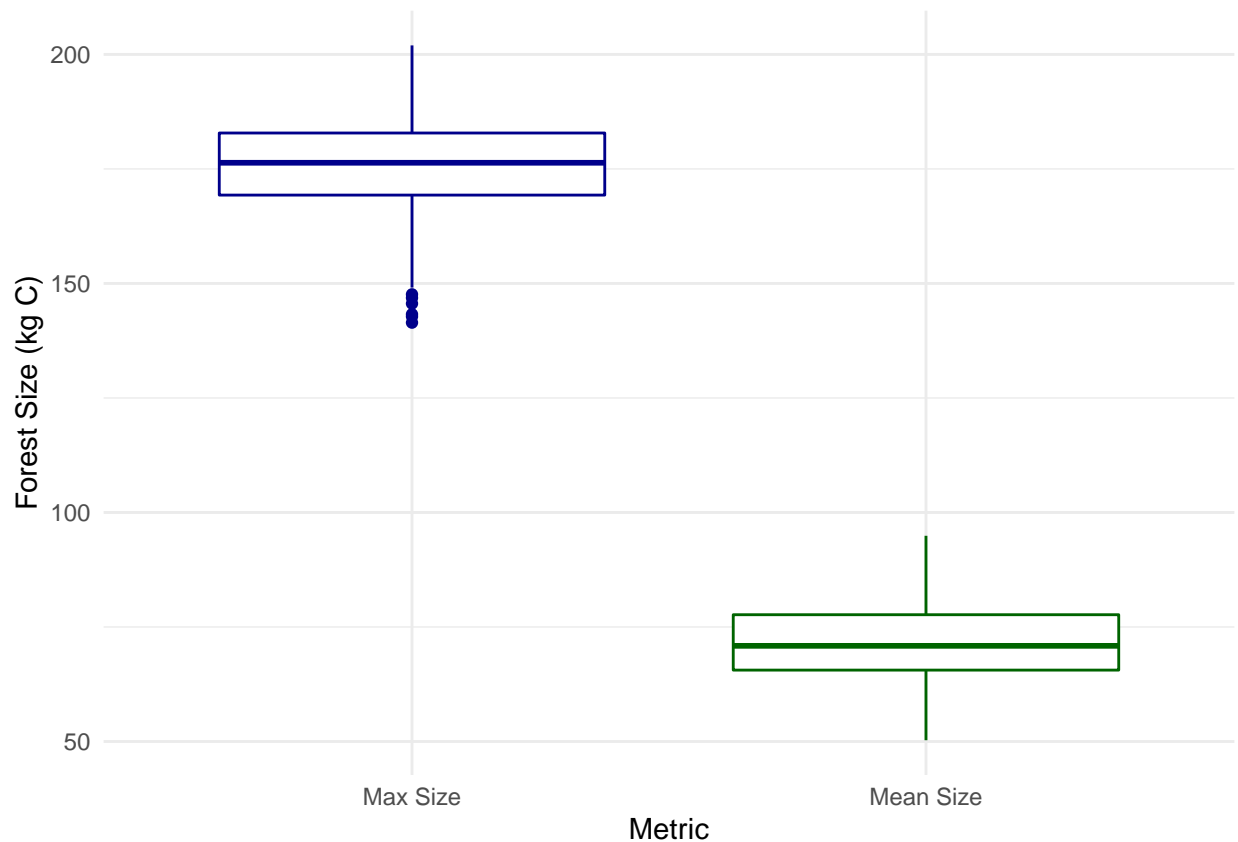
```
# extract out results from pmap into a data frame
```

```
allres = allresults %>% map_dfr(``, c("maxsize", "meansize"))
```

```
# create boxplots
```

```
tmp = allres %>% gather(key="metric", value="value")
```

```
ggplot(tmp, aes(metric, value, col=metric))+
  geom_boxplot() +
  theme_minimal() +
  labs(x = "Metric", y = "Forest Size (kg C)") +
  scale_x_discrete(labels = c("Max Size", "Mean Size")) +
  theme(legend.position="none") +
  scale_color_manual(values=c("darkblue", "darkgreen"))
```



Compute the sobol indices for each metric

```
# sobol can only handle one output at a time - so we will need to do them separately

sens_C_maxsize = sensitivity::tell(sens_C,allres$maxsize)

# first-order indices (main effect without co-variance), note: order "r", "K", "g", "thresh" (X1-X4)
sens_C_maxsize$S
```

```
##      original      bias std. error  min. c.i. max. c.i.
## X1  0.631708437 -0.01336244 0.05921864  0.5390245 0.7604032
## X2 -0.005883536 -0.01224598 0.08751886 -0.1599402 0.1827028
## X3  0.431525191 -0.01654284 0.07808197  0.3022125 0.6089747
## X4 -0.013027789 -0.01232715 0.08755616 -0.1653175 0.1717605
```

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

```
##          original          bias  std. error    min. c.i.    max. c.i.
## X1  6.064758e-01  1.521968e-02  9.543650e-02  3.836633e-01  7.652341e-01
## X2  4.936851e-03  8.582402e-05  4.055820e-03 -3.950048e-03  9.873702e-03
## X3  4.468355e-01  4.228041e-03  6.813965e-02  2.954293e-01  5.552333e-01
## X4 -6.492584e-13  6.969595e-13  4.580137e-13 -2.338322e-12 -4.949548e-13
```

```
# create another one for mean size
sens_C_meansize = sensitivity::tell(sens_C,allres$meansize)
# first-order indices (main effect without co-variance)
sens_C_meansize$S
```

```
##          original          bias std. error    min. c.i. max. c.i.
## X1  0.925726559 -0.0006277517  0.01317800  0.90261231  0.9557071
## X2  0.008996605 -0.0084985731  0.08683700 -0.14129601  0.1750952
## X3  0.115156231 -0.0071603888  0.08877687 -0.05470719  0.2876303
## X4  0.008624303 -0.0085385376  0.08683791 -0.14159161  0.1749688
```

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

```
##          original          bias  std. error    min. c.i.    max. c.i.
## X1  9.082823e-01  1.407774e-02  9.990965e-02  6.940565e-01  1.078188e+00
## X2  5.116897e-06  7.557589e-07  5.549715e-06 -1.051039e-05  1.023379e-05
## X3  8.452255e-02  2.281402e-03  1.534124e-02  5.000880e-02  1.087537e-01
## X4  2.420286e-14 -3.768837e-14  9.769682e-14 -1.322608e-13  2.590702e-13
```

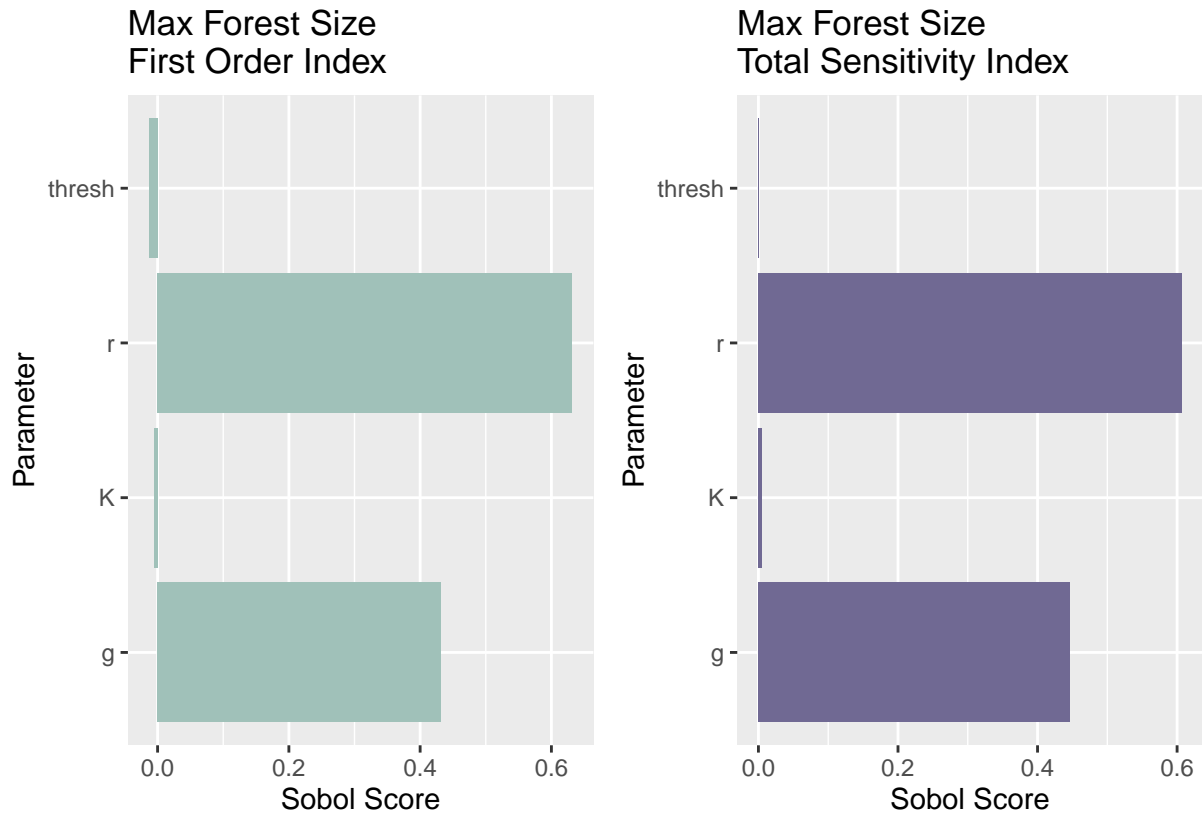
```
#max forest size sobol indices
S_sens_C_maxsize <- sens_C_maxsize$S %>%
  mutate(parameter = c("r", "K", "g", "thresh"))

T_sens_C_maxsize <- sens_C_maxsize$T %>%
  mutate(parameter = c("r", "K", "g", "thresh"))

S_max <- S_sens_C_maxsize %>%
  ggplot(aes(original, parameter)) +
  geom_col(fill = "#A0C1B9") +
  labs(x = "Sobol Score",
       y = "Parameter",
       title = "Max Forest Size \nFirst Order Index")

T_max <- T_sens_C_maxsize %>%
  ggplot(aes(original, parameter)) +
  geom_col(fill = "#706993") +
  labs(x = "Sobol Score",
       y = "Parameter",
       title = "Max Forest Size \nTotal Sensitivity Index")

S_max+T_max
```



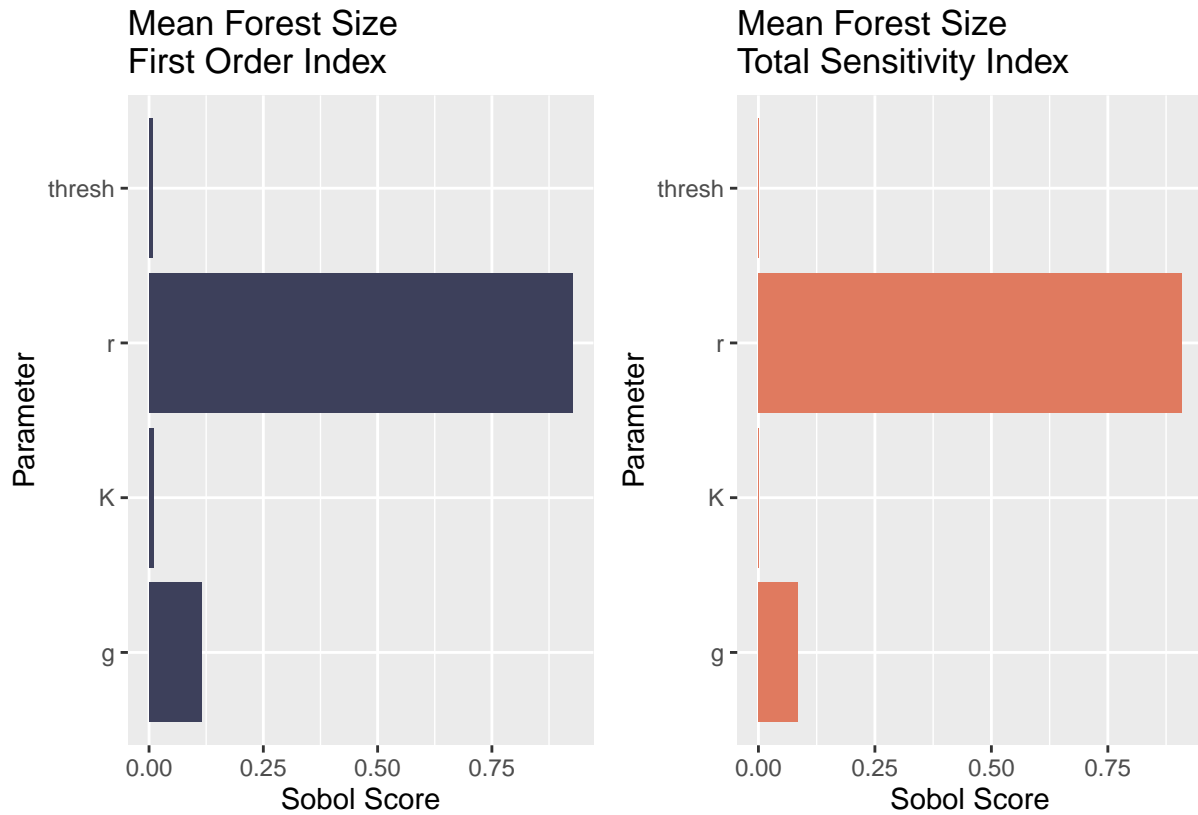
```
S_sens_C_meansize <- sens_C_meansize$S %>%
  mutate(parameter = c("r", "K", "g", "thresh"))

T_sens_C_meansize <- sens_C_meansize$T %>%
  mutate(parameter = c("r", "K", "g", "thresh"))

S_mean <- S_sens_C_meansize %>%
  ggplot(aes(original, parameter)) +
  geom_col(fill = "#3d405b") +
  labs(x = "Sobol Score",
       y = "Parameter",
       title = "Mean Forest Size \nFirst Order Index")

T_mean <- T_sens_C_meansize %>%
  ggplot(aes(original, parameter)) +
  geom_col(fill = "#e07a5f") +
  labs(x = "Sobol Score",
       y = "Parameter",
       title = "Mean Forest Size \nTotal Sensitivity Index")

S_mean+T_mean
```



Climate change impacts on forest growth

- K is highly sensitive under the total sensitivity index for max size. r and K are most sensitive under first-order indices for mean size.
- Climate change will influence the growth rate (r) of forests by influencing forest disturbance events such as wildfires, storms, pest and pathogen outbreaks, drought conditions, and more. Additionally, climate change will influence forest carbon cycling (GPP, NPP, etc.) leading to shifts in the carrying capacity of forest ecosystems. This will ultimately affect the productivity of forests, shifting resource management, economic processes, and forest product harvest.