Sensitivity Analysis of the Two-layer Atmosphere Energy Balance Model

This activity explores the sensitivity of the two-layer atmosphere energy balance model we used last week. Each step of the activity will take you through the steps of a typical Sensitivity Analysis workflow. There will be instructions of how and where to modify the code for each step. Make sure you understand what each step of the code does before you modify it!

1. Which three do you want to vary and why do you think those are interesting?
   (a) epsilon1 (the amount of radiation that the earth surface absorb can change),
   (b) epsilon2 (the amount of radiation that the atmosphere can absord can change)
   (c) alpha (the amount of radiation that atmosphere can reflect changes)
2. What is the output parameter that we plan to measure? T_surf_0 / Earth's surface temperature.

Step 1: Define Input and Output Parameters

Below are all the input parameters for your two layer atmosphere model you used last week. Choose three that you want to test the model's sensitivity to, and comment out those variables! Make sure you answer the following questions:

```
# Constant values that you probably shouldn't vary
# (at least without breaking physics or making the model for a different planet)
sigma <- 5.67 * 10^(-8) #W/m^2/K^4 or J/s/m^2/K^4
C_surf <- 1.176 * 10^(10) #J/K
C_atm <- 1.4 * 10^7 #J/K

# Parameters that you may want to vary
R_sun <- 342 #W/m^2 or J/s/m^2
alpha <- 0.3 #fraction
epsilon1 <- 0.8 #fraction
epsilon2 <- 0.5 #fraction
C_atm1 <- 0.5 * C_atm #J/K
C_atm2 <- 0.5 * C_atm #J/K
T_surf_0 <- 5 #C
T_atm1_0 <- -20 #C
T_atm2_0 <- -20 #C
```

Step 2: Choose Sensitivity Analysis Methods

Here you don't have much choice - we will be doing a qualitative sensitivity analysis using *both* one-at-a-time and all-at-a-time methods for your three parameters. However, depending on the parameter you chose above, it may make more sense to do the sensitivity analysis locally or globally. Before you move on, list whether you will do a local sensitivity analysis or global sensitivity analysis and why.

Step 3: Define Input Variability Space

For each of the three parameters above, choose the minimum and maximum value you want to explore, bearing in mind whether you are doing a global or local sensitivity analysis. Make sure you note which parameter below corresponds to which parameter named above.

```r
# Parmater 1
# NAME OF PARAMETER: epsilon1
# LOCAL OR GLOBAL: GLOBAL
p1_min <- 0
p1_max <- 1

# Parmater 2
# NAME OF PARAMETER: epsilon2
# LOCAL OR GLOBAL: GLOBAL
p2_min <- 0
p2_max <- 1

# Parmater 3
# NAME OF PARAMETER: alpha
# LOCAL OR GLOBAL: GLOBAL
p3_min <- 0
p3_max <- 1
```

Step 4+5: Choose a Sampling Strategy and Sample Size

For each parameter, you need to decide what values you want to probe within the range you determined above and thus how you plan on doing the sampling. In order to implement these samples below for each variable you selected above, you will also need to choose the number of samples to make.

```r
# Number of values to probe within each parameter
# (the same for all three parameters)
N_samples <- 50

# Define your parameter samples
p1_samples <- seq(p1_min, p1_max, length.out = N_samples)
p2_samples <- seq(p2_min, p2_max, length.out = N_samples)
p3_samples <- seq(p3_min, p3_max, length.out = N_samples)
```

Step 6: Assess Robustness and Convergence

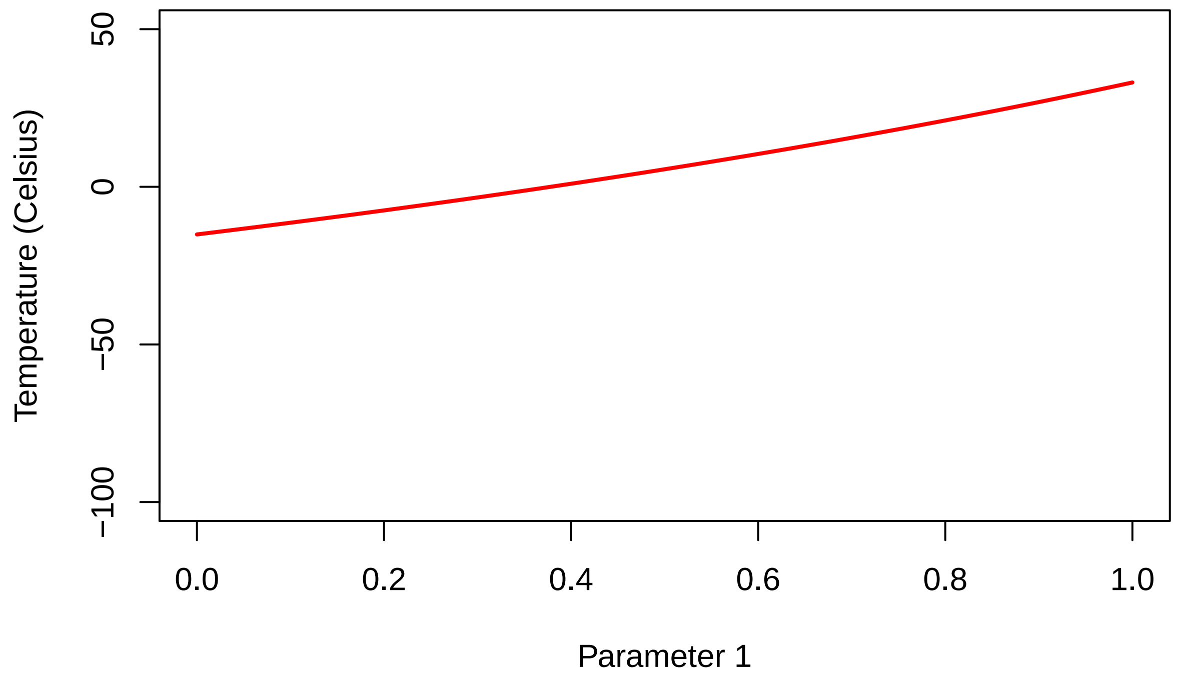
Recall that *robustness* assesses whether or not sensitivity indices change with different samples of a fixed size, and *convergence* assesses whether or not sensitivity indices change by increasing/decreasing the size of a sample. Before you move on, determine whether you output parameter (decided upon in Part 1) is *robust* to samples of a fixed size and/or *converges* to the same result if you increase or decrese the number of samples.

Step 7a: Implement & Visualize Results for the One-at-a-time Analysis

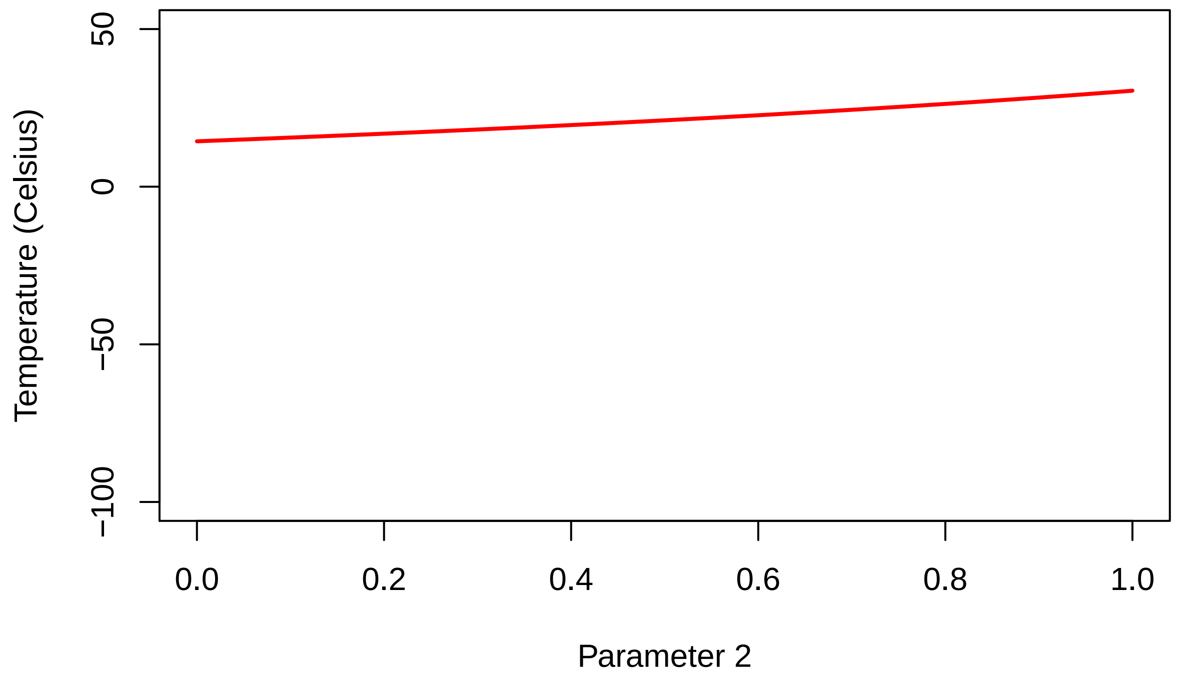
First, let's see how the output parameter depends on the three parameters that we chose to vary, but independent of one another. This means you will need to produce three plots, for each one you let one parameter vary, and the other two parameters are held fixed. Be sure to choose those fixed parameters wisely! (Should they be the minimum or maximum value? Or somewhere in between?) Pay attention to the comments below to determine what needs to be changed.

```r
# THINGS YOU DON'T NEED TO CHANGE
# Set up a sequence of times to calculate the temperatures. Since the temperature
# of the Earth changes slowly, increment the times by weeks instead of seconds.
num_steps <- 10000
times_weeks <- seq(num_steps)

# Since the units for the rates of change are J/s and K/s, we
# need to express convert weeks to seconds for the calculations.
times <- times_weeks * 60*60*24*7

# Create dummy lists to store the temperatures over time.
Ts_surf <- matrix(0,1,num_steps)
Ts_atm1 <- matrix(0,1,num_steps)
Ts_atm2 <- matrix(0,1,num_steps)

# Define a function to calculate the rates of change of the different
# temperatures in terms of the parameters and current temperatures.
change_rates <- function(T_surf, T_atm1, T_atm2, C_surf, C_atm1, C_atm2,
                         R_sun, alpha, sigma, epsilon1, epsilon2){

   A_surf <- (1-alpha)*R_sun + epsilon1*sigma*T_atm1^4
                             + epsilon2*(1-epsilon1)*sigma*T_atm2^4
   R_surf <- sigma*T_surf^4

   A_atm1 <- epsilon1*sigma*T_surf^4 + epsilon2*epsilon1*sigma*T_atm2^4
   R_atm1 <- 2*epsilon1*sigma*T_atm1^4

   A_atm2 <- epsilon2*(1-epsilon1)*sigma*T_surf^4 + epsilon2*epsilon1*sigma*T_atm1^4
   R_atm2 <- 2*epsilon2*sigma*T_atm2^4

   T_dot_surf <- (A_surf - R_surf) / C_surf
   T_dot_atm1 <- (A_atm1 - R_atm1) / C_atm1
   T_dot_atm2 <- (A_atm2 - R_atm2) / C_atm2

   return( c(T_dot_surf, T_dot_atm1, T_dot_atm2) )
}

# Create dummy lists to hold the final output temperatures
p1_Tsurf_finals <- matrix(0,1,N_samples)
p2_Tsurf_finals <- matrix(0,1,N_samples)
p3_Tsurf_finals <- matrix(0,1,N_samples)


# THINGS YOU MAY NEED TO CHANGE FOR PARAMETER 1:

# Iterate over all the samples for this parameter
for (ii in 1:N_samples){
  # Insert the initial temperatures into the storage lists.
  Ts_surf[1] <- T_surf_0
  Ts_atm1[1] <- T_atm1_0
  Ts_atm2[1] <- T_atm2_0

  # Iteratively fill in the temperatures in the storage lists.
  for(t in 2:num_steps){
    # Calculate the time step.
    delta <- times[t] - times[t-1]

    # Convert the previous temperatures from Celsius to Kelvin.
    T_surf_K <- Ts_surf[t-1] + 273
    T_atm1_K <- Ts_atm1[t-1] + 273
    T_atm2_K <- Ts_atm2[t-1] + 273

    # Input the previous temperatures into the change_rates function.
    # THINK CAREFULLY HERE...
    rates <- change_rates(T_surf_K, T_atm1_K, T_atm2_K, C_surf,
                          C_atm1, C_atm2, R_sun, alpha,
                          sigma, p1_samples[ii], epsilon2)

    # Calculate the current temperatures using the previous temperatures,
    # time step, and rates of change.
    Ts_surf[t] <- Ts_surf[t-1] + delta * rates[1]
    Ts_atm1[t] <- Ts_atm1[t-1] + delta * rates[2]
    Ts_atm2[t] <- Ts_atm2[t-1] + delta * rates[3]
  }

  # Extract the final temperature for the Earth and store it in the list
  p1_Tsurf_finals[ii] <- Ts_surf[num_steps]
}

# THINGS YOU MAY NEED TO CHANGE FOR PARAMETER 2:

# Iterate over all the samples for this parameter
for (ii in 1:N_samples){
  # Insert the initial temperatures into the storage lists.
  Ts_surf[1] <- T_surf_0
  Ts_atm1[1] <- T_atm1_0
  Ts_atm2[1] <- T_atm2_0

  # Iteratively fill in the temperatures in the storage lists.
  for(t in 2:num_steps){

    # Calculate the time step.
    delta <- times[t] - times[t-1]

    # Convert the previous temperatures from Celsius to Kelvin.
    T_surf_K <- Ts_surf[t-1] + 273
    T_atm1_K <- Ts_atm1[t-1] + 273
    T_atm2_K <- Ts_atm2[t-1] + 273

    # Input the previous temperatures into the change_rates function.
    # THINK CAREFULLY HERE...
    rates <- change_rates(T_surf_K, T_atm1_K, T_atm2_K, C_surf,
                          C_atm1, C_atm2, R_sun, alpha,
                          sigma, epsilon1, p2_samples[ii])

    # Calculate the current temperatures using the previous temperatures,
    # time step, and rates of change.
    Ts_surf[t] <- Ts_surf[t-1] + delta * rates[1]
    Ts_atm1[t] <- Ts_atm1[t-1] + delta * rates[2]
    Ts_atm2[t] <- Ts_atm2[t-1] + delta * rates[3]
  }

  # Extract the final temperature for the Earth and store it in the list
  p2_Tsurf_finals[ii] <- Ts_surf[num_steps]
}

# THINGS YOU MAY NEED TO CHANGE FOR PARAMETER 3:

# Iterate over all the samples for this parameter
for (ii in 1:N_samples){
  # Insert the initial temperatures into the storage lists.
  Ts_surf[1] <- T_surf_0
  Ts_atm1[1] <- T_atm1_0
  Ts_atm2[1] <- T_atm2_0

  # Iteratively fill in the temperatures in the storage lists.
  for(t in 2:num_steps){

    # Calculate the time step.
    delta <- times[t] - times[t-1]

    # Convert the previous temperatures from Celsius to Kelvin.
    T_surf_K <- Ts_surf[t-1] + 273
    T_atm1_K <- Ts_atm1[t-1] + 273
    T_atm2_K <- Ts_atm2[t-1] + 273

    # Input the previous temperatures into the change_rates function.
    # THINK CAREFULLY HERE...
    rates <- change_rates(T_surf_K, T_atm1_K, T_atm2_K, C_surf,
                          C_atm1, C_atm2, R_sun, p3_samples[ii],
                          sigma, epsilon1, epsilon2)

    # Calculate the current temperatures using the previous temperatures,
    # time step, and rates of change.
    Ts_surf[t] <- Ts_surf[t-1] + delta * rates[1]
    Ts_atm1[t] <- Ts_atm1[t-1] + delta * rates[2]
    Ts_atm2[t] <- Ts_atm2[t-1] + delta * rates[3]
  }

  # Extract the final temperature for the Earth and store it in the list
  p3_Tsurf_finals[ii] <- Ts_surf[num_steps]
}

# Now lets plot the results!
# Make sure you update "xlab" to a proper desciptor for that parameter (with units!)
# You may need to update the "ylim" parameter to change the range plotted in the y-axis
```

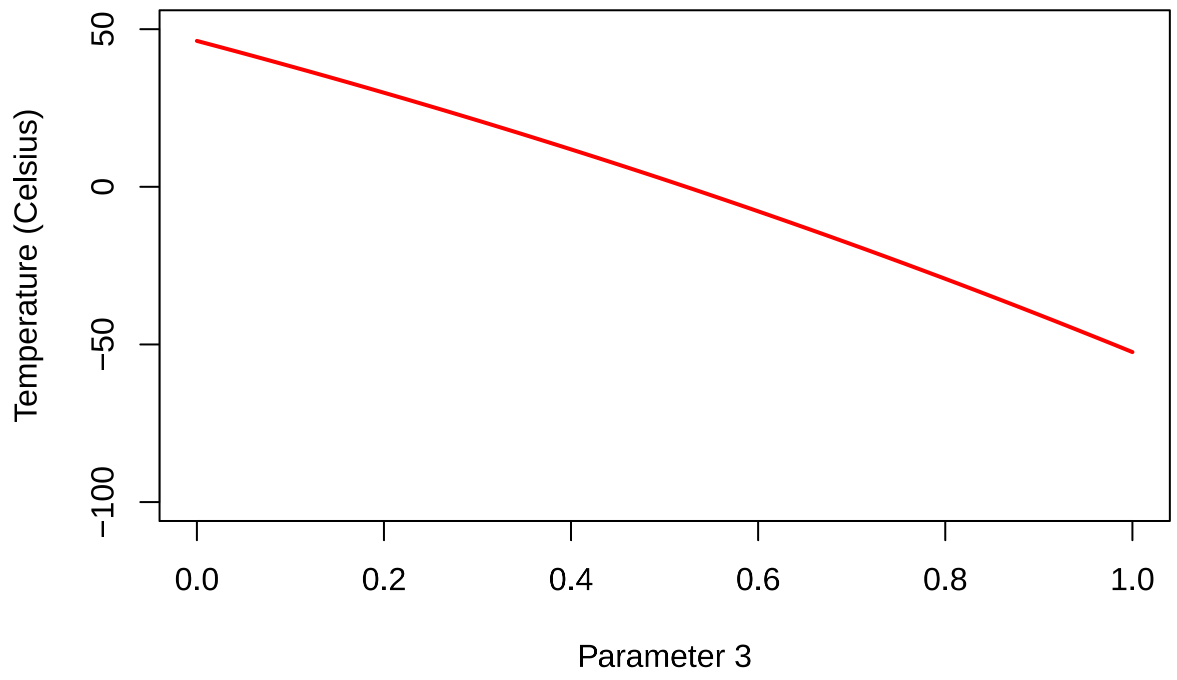
```
# Parameter 1:
plot(p1_samples,p1_Tsurf_finals,type="l",ylim=c(-100,50),col="red",lwd=2,
     xlab="Parameter 1",ylab="Temperature (Celsius)")
```



```r
# Parameter 2:
plot(p2_samples,p2_Tsurf_finals,type="l",ylim=c(-100,50),col="red",lwd=2,
     xlab="Parameter 2",ylab="Temperature (Celsius)")
```



```r
# Parameter 3:
plot(p3_samples,p3_Tsurf_finals,type="l",ylim=c(-100,50),col="red",lwd=2,
     xlab="Parameter 3",ylab="Temperature (Celsius)")
```

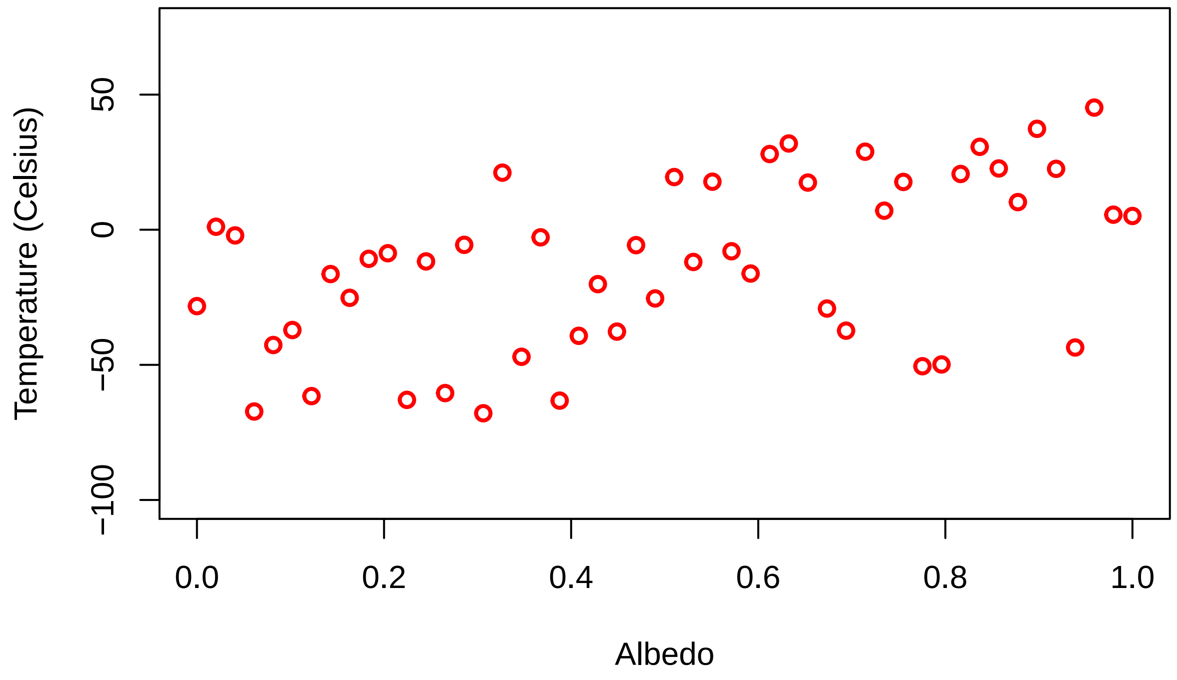


Step 7b: Implement & Visualize Results for the All-at-a-time Analysis

Sometimes the whole is more than the sum of its parts! In order to test how these parameters interact with each other, we'll now do an all-at-a-time analysis. The easy (but dumb) way to implement this is to calculate the output parameter for every possible combination of the three input parameters, but that would be *very* slow since it would involve many nested for-loops. Instead we'll loop over just one of the parameters (as above), and for each iteration we'll randomly select one of the possible values for each of the other two parameters.

```r
# Create dummy lists to hold the final output temperatures
pAll_Tsurf_finals <- matrix(0,1,N_samples)

# YOU MAY NEED TO CHANGE THINGS FROM HERE DOWN

# Iterate over all the samples for this parameter
for (ii in 1:N_samples){
  # Insert the initial temperatures into the storage lists.
  Ts_surf[1] <- T_surf_0
  Ts_atm1[1] <- T_atm1_0
  Ts_atm2[1] <- T_atm2_0

  # Randomly select a value for each of the other two parameters
  # For R experts: one can just use the "sample" function to get a random value,
  # but make sure you set replace=True!
  p2_rand_position <- floor(runif(1, min=1, max=N_samples))
  p3_rand_position <- floor(runif(1, min=1, max=N_samples))
  p2_rand_value <- p2_samples[p2_rand_position]
  p3_rand_value <- p3_samples[p3_rand_position]

  # Iteratively fill in the temperatures in the storage lists.
  for(t in 2:num_steps){
    # Calculate the time step.
    delta <- times[t] - times[t-1]

    # Convert the previous temperatures from Celsius to Kelvin.
    T_surf_K <- Ts_surf[t-1] + 273
    T_atm1_K <- Ts_atm1[t-1] + 273
    T_atm2_K <- Ts_atm2[t-1] + 273

    # Input the previous temperatures into the change_rates function.
    # THINK CAREFULLY HERE...
    rates <- change_rates(T_surf_K, T_atm1_K, T_atm2_K, C_surf,
                          C_atm1, C_atm2, R_sun, p3_rand_value,
                          sigma, p1_samples[ii], p2_rand_value)

    # Calculate the current temperatures using the previous temperatures,
    # time step, and rates of change.
    Ts_surf[t] <- Ts_surf[t-1] + delta * rates[1]
    Ts_atm1[t] <- Ts_atm1[t-1] + delta * rates[2]
    Ts_atm2[t] <- Ts_atm2[t-1] + delta * rates[3]
  }

  # Extract the final temperature for the Earth and store it in the list
  pAll_Tsurf_finals[ii] <- Ts_surf[num_steps]
}
```

```
# Now lets plot the results!
# Make sure you update "xlab" to a proper desciptor for that parameter (with units!)
plot(p1_samples,pAll_Tsurf_finals,type="p",ylim=c(-100,75),col="red",lwd=2,
     xlab="Albedo",ylab="Temperature (Celsius)")
```


Temperature (Celsius)
Albedo