R Notebook

Woah! Welcome back to having to think after the collective food coma of the Thanksgiving break! Remember the good times we had talking about how the atmospheric signal of water vapor got incoded into the ice sheet? If you don't, here is a quick reminder:

We are in this "viscous sublayer" or VSL (also called the laminar layer in the Craig-Gordon model). In this layer, transport of different isotopes is entirely diffusion! In these exercises we are going to explore what the means for the top of the ice sheet (or "firn"). More concisely, we are going to use a toy diffusion model to see the effect that the VSL's thickness has on what the firn "sees" from the atmosphere. From the paper:

"A positive linear relation between B_best and delta_z is expected since the influence of the bottom boundary condition is more attenuated for a thicker VSL."

What the hell is B_best anyways? Well we have snow measurements and atmospheric measurements but it's hard to make a measurement **AT** the snow surface. Madsen et al. assume that there is some dirunal variation B, a mean value A, and phase offset C. In the end, the value at the snow surface will look something like this:

$$\delta^*(z=0,t) = A^* + B^* sin(\frac{2\pi t}{T_{day}} + C^*)$$

Exercise 1: Go ahead and make a function called "dirurnal_isotope" as a function of A, B, C, and t. Then lets do a sanity check by plotting it in a tibble called "isotope_model_data". Go ahead and use the values from the 26th-29th of June from the paper for

$$\delta^{18}O$$

which so happens to be

$$A_{best} = -43.9, B_{best} = 3.3, C_{best} = 14.4$$

.

```
diurnal_isotope <- function(A, B, C, t) {
   isotope <- A + B*sin((2*pi*t)/24 + C)
   return(isotope)
}
```

Here is the plot.

```
isotope_model_data <- tibble(time = seq(from = 0, to = 24, by = 1), snow_value = diurnal_isotope(-43.9,

isotope_model_data %>%
    ggplot(aes(x = time, y = snow_value)) +
    geom_line(lwd = 1) +
    theme(text = element_text(size = 20))
```

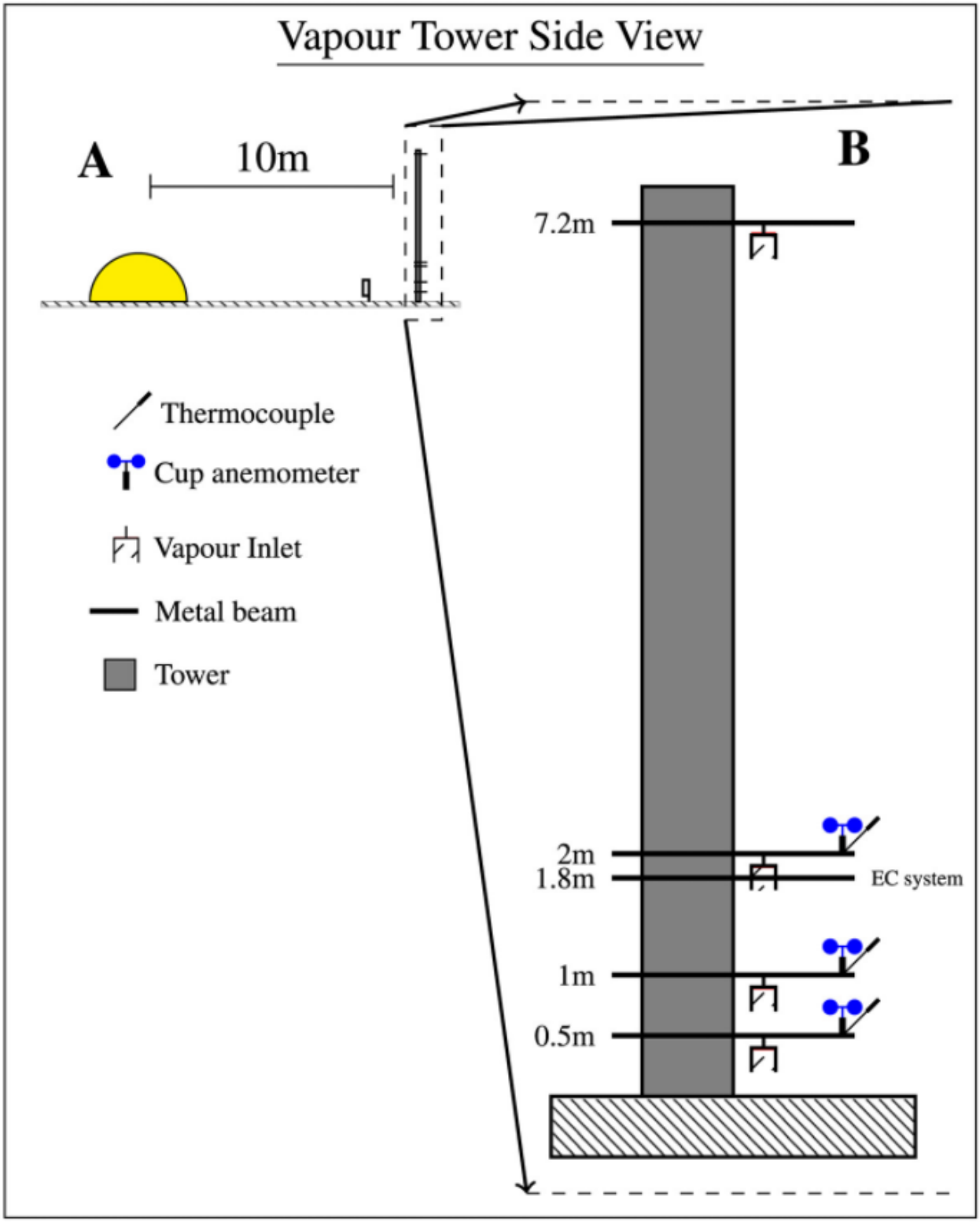
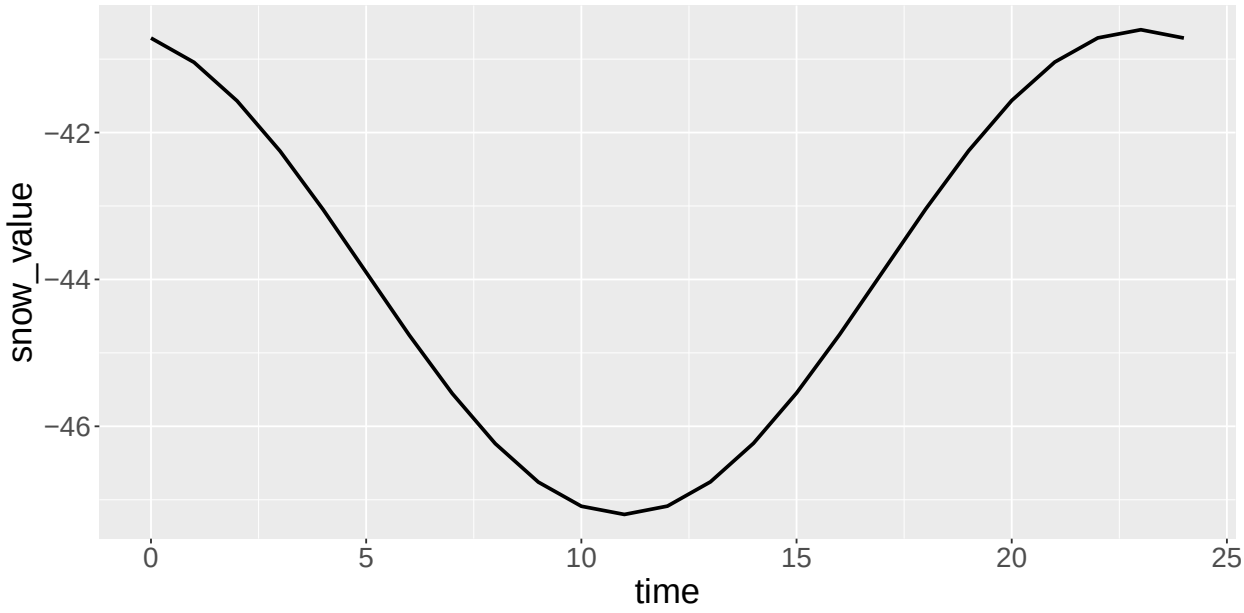


Figure 1: A little reminder of where we are in the system.



Differential equationsin R isn't crazy mature, though it has some great basic packages that will help us get pretty close. We are going to solve the diffusion equation in 1D with the help of ReacTran (which will give us an equation) and deSolve (which will do our solving). We will get a steady state solution at each time step, go much more forward in time and solve again with the new atmoshperic boundary conditions. The very first thing to do is get comfortable with solving the diffusion equation and inform our model spin-up time.

Note: We are ignoring the advection term here. Though Madesen et al. includes it, it is pretty damn complicated and outside the scope of these excersies.

```
# Lets make a grid in the spatial dimensions. This will represent a VSL of 0.1 m
Grid <- setup.grid.1D(N = 10000, L = 0.01)

# The function for solving
pde1D <-function(t, C, parms) {
  tran <- tran.1D(C = C, D = D,
  C.down = Cdown, C.up = Cup, dx = Grid)$dC
  list(tran) # return value: rate of change
}

D <- 0.00000009723 # diffusion constant for 18O in 0.1m/sec
Cdown <- diurnal_isotope(-43.9, 3.3, 14.4, 0) #snow value
Cup <- diurnal_isotope(-45, 10, 14.4, 0) #air value
Cave <- (Cdown + Cup) / 2 # we need an inital condition somehow, lets do the average
diff_time <- 180 #number of time steps in units of what our diffusion constant is in time

times <- seq(0, diff_time, by = 1) #make time array
system.time(
  out <- ode.1D(y = rep(Cave, Grid$N),
  times = times, func = pde1D,
  parms = NULL, nspec = 1)
)
```

```
##    user  system elapsed
##    1.36    0.01    1.37
```

```
tail(out[, 1:10], n = 10) #sample the out file from the solver
```

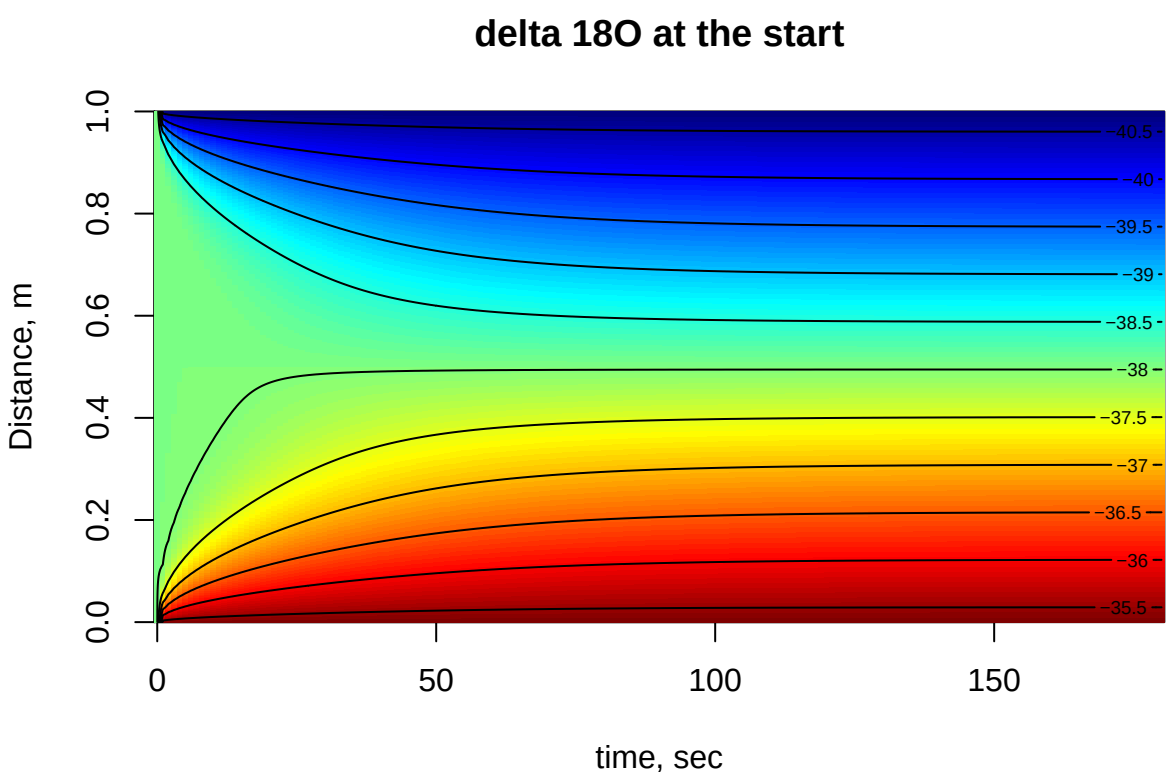
```
##        time         1         2         3         4         5         6
## [172,]  171 -35.34369 -35.34423 -35.34477 -35.34531 -35.34585 -35.34638
## [173,]  172 -35.34369 -35.34423 -35.34477 -35.34531 -35.34585 -35.34638
## [174,]  173 -35.34369 -35.34423 -35.34477 -35.34531 -35.34585 -35.34638
## [175,]  174 -35.34369 -35.34423 -35.34477 -35.34531 -35.34584 -35.34638
## [176,]  175 -35.34369 -35.34423 -35.34477 -35.34531 -35.34584 -35.34638
## [177,]  176 -35.34369 -35.34423 -35.34477 -35.34531 -35.34584 -35.34638
## [178,]  177 -35.34369 -35.34423 -35.34477 -35.34531 -35.34584 -35.34638
## [179,]  178 -35.34369 -35.34423 -35.34477 -35.34531 -35.34584 -35.34638
## [180,]  179 -35.34369 -35.34423 -35.34477 -35.34531 -35.34584 -35.34638
## [181,]  180 -35.34369 -35.34423 -35.34477 -35.34531 -35.34584 -35.34638
##                7         8       9
## [172,] -35.34692 -35.34746 -35.348
## [173,] -35.34692 -35.34746 -35.348
## [174,] -35.34692 -35.34746 -35.348
## [175,] -35.34692 -35.34746 -35.348
## [176,] -35.34692 -35.34746 -35.348
## [177,] -35.34692 -35.34746 -35.348
## [178,] -35.34692 -35.34746 -35.348
## [179,] -35.34692 -35.34746 -35.348
## [180,] -35.34692 -35.34746 -35.348
## [181,] -35.34692 -35.34746 -35.348
```

Let's take the solution and plot it. Also we'll make sure to take the last time file as our future

$$t = 0$$

boundary condition and call it steady_state_solution.

```
sss_temp <- out[121,]
steady_state_solution <- sss_temp[-1]
image(out, xlab = "time, sec",
  ylab = "Distance, m",
  main = "delta 18O at the start", add.contour = TRUE)
```



To do this at each time step we'll pack this into one big function, "isotope_vapor_diff". We just care about the end values so that's the output. For the data book-keeping, tibbles can't hold multi-dimenional data at a point unless it is a list. I have packed away the output as a list and unpacked it from the "starting_values" input variable.

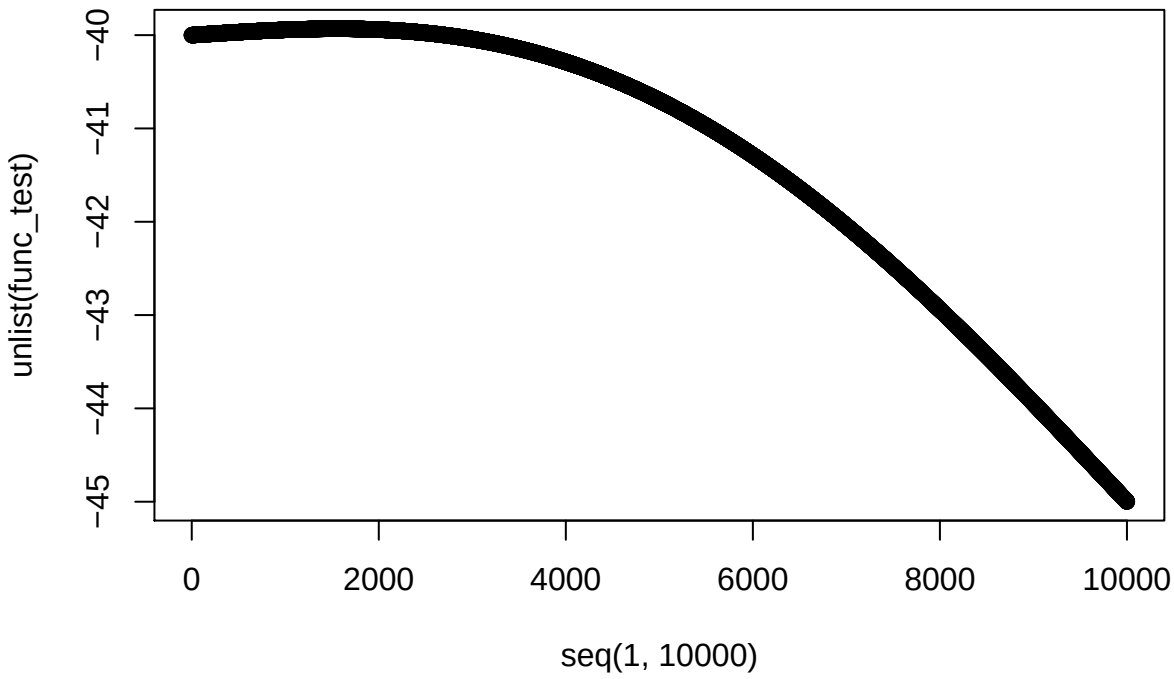
```
isotope_vapor_diff = function(starting_values, snow_value, atmos_value) {
  if(is.list(starting_values)){
    starting_values <- unlist(starting_values)
  }

  D <- 0.00000009723 # diffusion constant
  Cdown <- snow_value
  Cup <- atmos_value
  diff_time <- 600


  Grid <- setup.grid.1D(N = 10000, L = 0.01)

  pde1D <-function(t, C, parms) {
    tran <- tran.1D(C = C, D = D,
    C.down = Cdown, C.up = Cup, dx = Grid)$dC
    list(tran) # return value: rate of change
    }


  times <- seq(0, diff_time, by = 1)
```

```
  out <- ode.1D(y = starting_values,
    times = times, func = pde1D,
    parms = NULL, nspec = 1)

  sss_temp <- out[121,]
  steady_state_solution <- list(sss_temp[-1])
  return(steady_state_solution)
}
```

Exercise 2: Use this fancy new function to explore how the VSL would look in three minutes if you changed the surface snow value and air value from our previous steady state solution to two new values. Plot your results.

Note: Remember that the function returns a list. You can unpack that bad boy with a statement that looks like "unlist(your_list_here)".

```
func_test <- isotope_vapor_diff(steady_state_solution, -45, -40)
plot(seq(1,10000), unlist(func_test))
```



On to the main event. Now we need to add in our first steady state solution to serve as the t = 0 boundary of our model!

```
isotope_model_data <- isotope_model_data %>%
  add_column(isotope_space = NA)
isotope_model_data$isotope_space[1] = list(steady_state_solution)
isotope_model_data
```

```
## # A tibble: 25 x 4
##     time snow_value air_value isotope_space
##    <dbl>      <dbl>     <dbl> <list>
##  1     0      -40.7     -35.1 <dbl [10,000]>
##  2     1      -41.0     -35.1 <lgl [1]>
##  3     2      -41.6     -35.7 <lgl [1]>
##  4     3      -42.3     -37.0 <lgl [1]>
##  5     4      -43.0     -38.9 <lgl [1]>
##  6     5      -43.9     -41.1 <lgl [1]>
##  7     6      -44.8     -43.6 <lgl [1]>
##  8     7      -45.6     -46.2 <lgl [1]>
##  9     8      -46.2     -48.8 <lgl [1]>
## 10     9      -46.8     -51.0 <lgl [1]>
## # ... with 15 more rows
```

Looks good. Now we use the diffusion function to fill out the rest of the isotope_space column using the end of previous step as the starting value.

```
interator = length(isotope_model_data$isotope_space) - 1
system.time(
for (i in 1:interator) {
  isotope_model_data$isotope_space[i+1] = isotope_vapor_diff(isotope_model_data$isotope_space[i], isotop
}
)
```

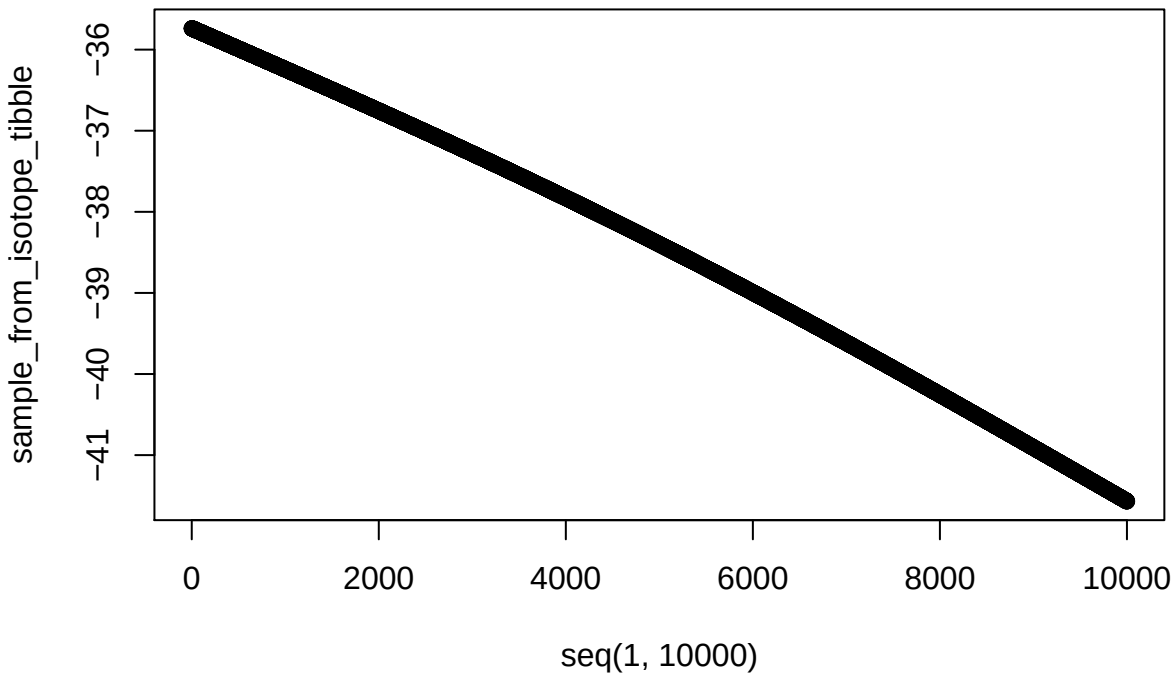
```
##    user  system elapsed
##   46.14    0.50   46.83
```

```
isotope_model_data
```

```
## # A tibble: 25 x 4
##     time snow_value air_value isotope_space
##    <dbl>      <dbl>     <dbl> <list>
##  1     0      -40.7     -35.1 <dbl [10,000]>
##  2     1      -41.0     -35.1 <dbl [10,000]>
##  3     2      -41.6     -35.7 <dbl [10,000]>
##  4     3      -42.3     -37.0 <dbl [10,000]>
##  5     4      -43.0     -38.9 <dbl [10,000]>
##  6     5      -43.9     -41.1 <dbl [10,000]>
##  7     6      -44.8     -43.6 <dbl [10,000]>
##  8     7      -45.6     -46.2 <dbl [10,000]>
##  9     8      -46.2     -48.8 <dbl [10,000]>
## 10     9      -46.8     -51.0 <dbl [10,000]>
## # ... with 15 more rows
```

This is how we might look at a single solution. In this case, hour 4.

```
sample_from_isotope_tibble <- unlist(isotope_model_data$isotope_space[4])
plot(seq(1,10000), sample_from_isotope_tibble)
```



At each time, we can look at the isotope value at different parts of the space as a rough approximation of what value the snow would see if the VSL was that thick.

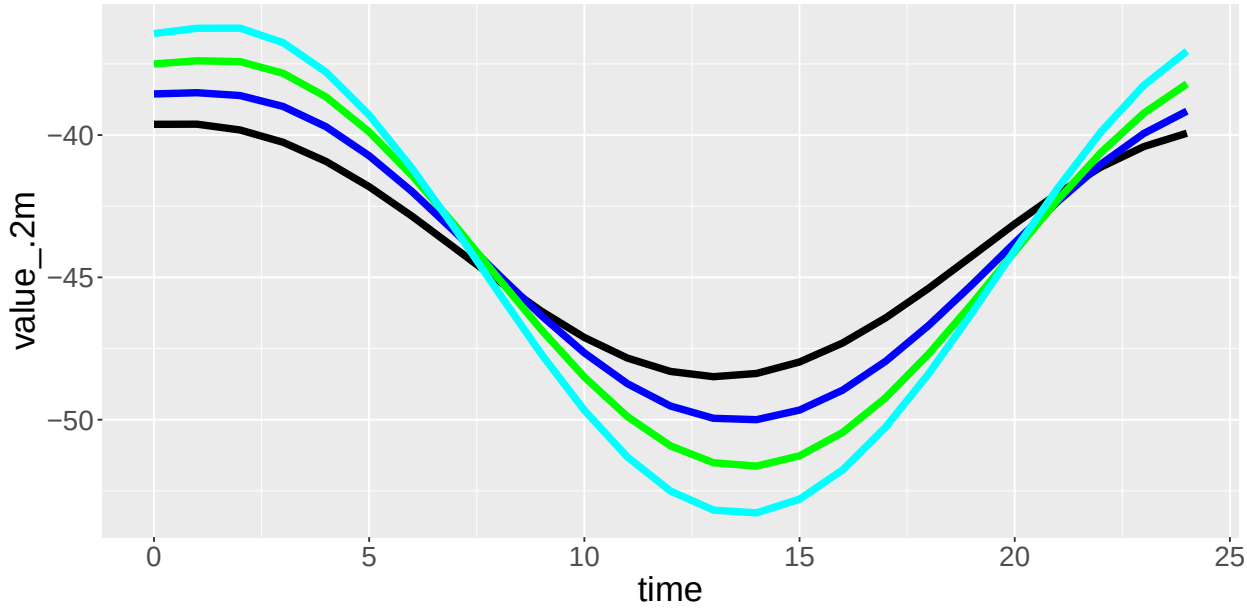
Exercise 3: Add to the tibble four new columns for the value at four different heights of your choosing. Is your result linear with respect to VSL height? How could you tell?

```
isotope_model_data <- isotope_model_data %>% rowwise() %>%
  mutate(value_.2m = unlist(isotope_space)[8000])

isotope_model_data <- isotope_model_data %>% rowwise() %>%
  mutate(value_.4m = unlist(isotope_space)[6000])

isotope_model_data <- isotope_model_data %>% rowwise() %>%
  mutate(value_.6m = unlist(isotope_space)[4000])

isotope_model_data <- isotope_model_data %>% rowwise() %>%
  mutate(value_.8m = unlist(isotope_space)[2000])

isotope_model_data
```

```
## # A tibble: 25 x 8
## # Rowwise:
##     time snow_value air_value isotope_space value_.2m value_.4m value_.6m
##    <dbl>      <dbl>     <dbl> <list>            <dbl>     <dbl>     <dbl>
##  1     0      -40.7     -35.1 <dbl [10,000~     -39.6     -38.6     -37.5
##  2     1      -41.0     -35.1 <dbl [10,000~     -39.6     -38.5     -37.4
```

```
##  3     2      -41.6     -35.7 <dbl [10,000~     -39.8     -38.6     -37.4
##  4     3      -42.3     -37.0 <dbl [10,000~     -40.3     -39.0     -37.8
##  5     4      -43.0     -38.9 <dbl [10,000~     -40.9     -39.7     -38.7
##  6     5      -43.9     -41.1 <dbl [10,000~     -41.8     -40.7     -39.9
##  7     6      -44.8     -43.6 <dbl [10,000~     -42.9     -42.0     -41.4
##  8     7      -45.6     -46.2 <dbl [10,000~     -44.0     -43.4     -43.2
##  9     8      -46.2     -48.8 <dbl [10,000~     -45.1     -44.9     -45.0
## 10     9      -46.8     -51.0 <dbl [10,000~     -46.2     -46.3     -46.8
## # ... with 15 more rows, and 1 more variable: value_.8m <dbl>
```

```
diff_heights <- isotope_model_data %>%
  ggplot(aes(x = time)) +
  geom_line(aes(y = value_.2m), lwd = 2) +
  geom_line(aes(y = value_.4m), color="blue", lwd = 2) +
  geom_line(aes(y = value_.6m), color="green", lwd = 2) +
  geom_line(aes(y = value_.8m), color="cyan", lwd = 2) +
  theme(text = element_text(size = 20))
diff_heights
```



This seems roughly linearly attenuated based on VSL thickness!