A collective analysis for 126 simulations

Ranvir Singh Virk

2023-11-14

Loading and joining the data

Since we have our data in 126 different files, we will be joining it in one csv and pre-processing the data to work with.

Let's have a **process_file** function that takes the file and file_id as inputs and returns data after doing the following conversions for us:
- Set's file_number for each file
- Convert the time from seconds to year
- Convert the calcite to avg_calcite (Averaging out the calcite values for first where value for soln is between 1 to 10)
- Convert the values mole/liter to ton CO2/Ha

```
# Create a funtion to prepocess and join all the files
process_file <- function(file, file_id){
  file_number <- str_extract(file, "(?<=/)[^/]+(?=\\.out$)") # Extracting the file number
  data <- read.table(file, header = TRUE) |>
    filter(state == "transp") |> # Get the files which have state as transp
    mutate(year = time/(3600*24*365)) |> # Convert form sec to year
    rename(`C(4)` = "C.4.") |>
    select(c('soln', 'Calcite', 'Sr', 'C(4)', 'year')) |>
    group_by(year) |>
    mutate(avg_calcite = ifelse(soln %in% 1:10, mean(Calcite[soln %in% 1:10]), Calcite)) |> # Getting t
    ungroup() |>
    # Conversions from mole/liter to CO2/Ha
    mutate(Calcite = avg_calcite,
           Calcite = Calcite * 500000 * 44 / 1000000 / ifelse(soln %in% 1:10, 1, 10),
           `C(4)` = `C(4)` * 500000 * 44 / 1000000 / 10,
           file_id = file_id) |>
    select(-avg_calcite)
  return(data)
}
```

Now that we have this function to process the files, let's read and combine all the files we have

```
# Seting the base path for alljobs directory
base_path <- "../Data/alljobs"

# Getting all the simulation folders from the base directory
folders <- list.dirs(path = base_path, full.names = TRUE, recursive = FALSE)

all_data <- list()

file_counter <- 1

for (folder in folders) {
  # Adjust the pattern to match your .out files, if they have a specific naming convention
  file_paths <- list.files(path = folder, full.names = TRUE, pattern = "*.out$")
  data <- process_file(file_paths, file_counter)
  all_data <- bind_rows(all_data, data)
  file_counter <- file_counter + 1
}
```

Get the data for Total CO2 Capture

We will be partitioning the data in 3 different variables for calcite, total_capture and the data where the value for soln = 11
The Calcite and soln_11_data will help us get the total_capture data required for the Total_CO2_Capture graphs.

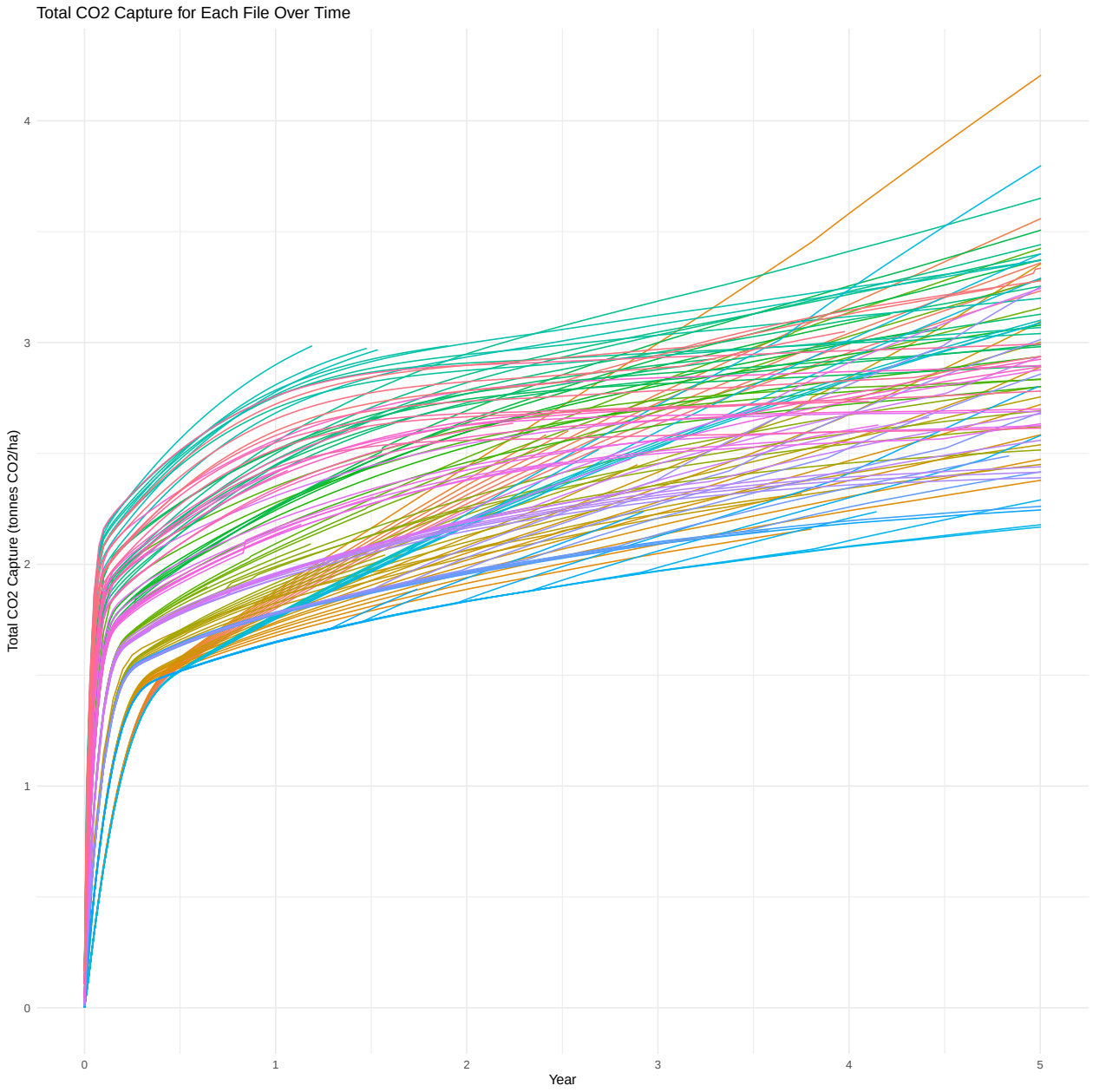
```
# So let's get the data where the soln value is 11
# We remove the Sr column as we do not need the Sr data for total CO2 caputre data.
soln_11_data <- all_data |>
  filter(soln == 11) |>
  group_by(file_id)|>
  mutate(`C(4)` = cumsum(`C(4)`)) |>
  ungroup()|>
  dplyr::select(-"Sr")

calcite <- all_data |>
  filter(soln %in% 1:10) |>
  dplyr::select(-"Sr")


# Now we can use the claclite and soln_11_data to create the total_capture data that we need to plot fo
total_capture <- full_join(calcite, soln_11_data, by = c("year", "file_id"), suffix = c(".soil", ".efflu
  mutate(Total_CO2_capture = Calcite.soil + `C(4).effluent` + Calcite.effluent) |>
  filter(soln.soil == 1) |>
  dplyr::select(c(Total_CO2_capture, year, file_id))
```

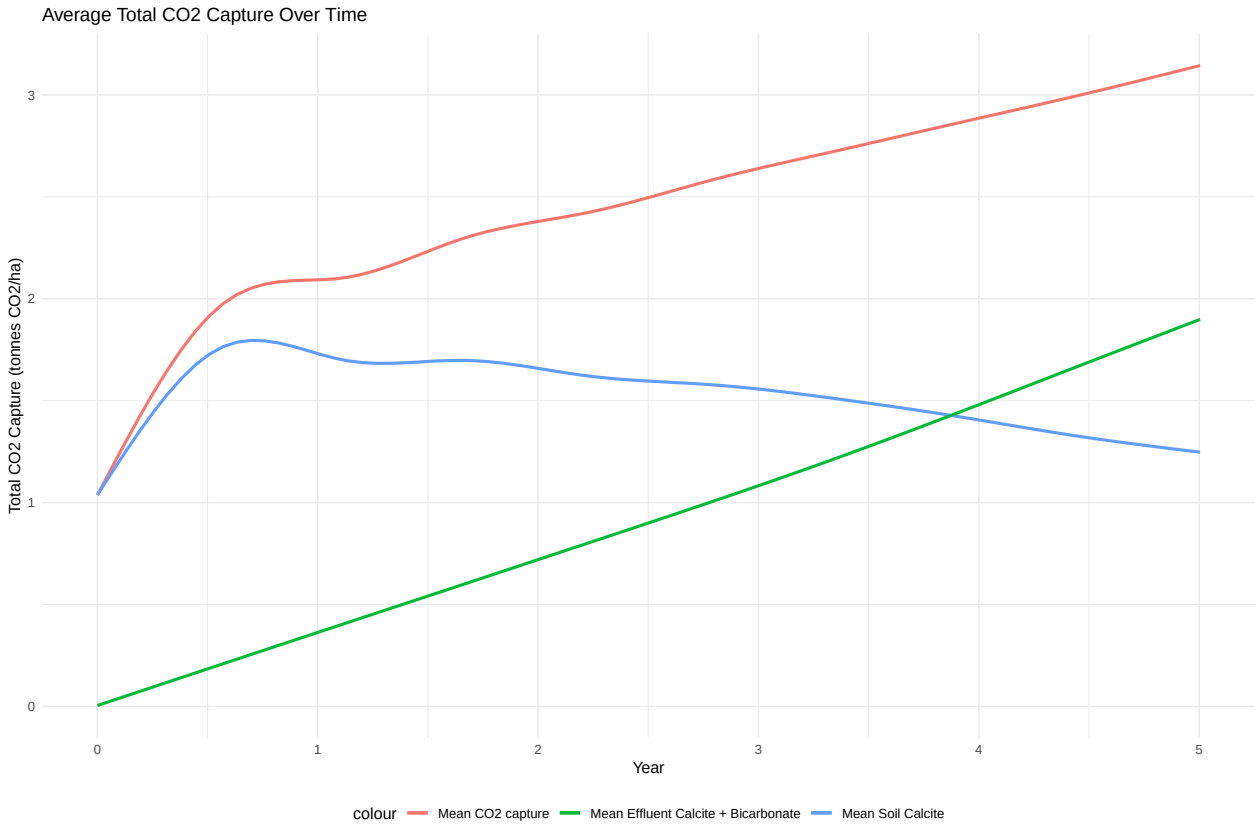
Plots

Let's take a look at the plot for total co2 capture for the 126 files separately.



We can see a trend where the Total CO2 Capture is increasing rapidly for about 0.3 years, and then slows down giving us a elbow shaped curve.

Let's also take a look at an graph for mean values of Calcite, Bicarbonate and Total CO2 Capture.



There seems to be a relation between the Mean CO2 Capture and the Mean Effluent Calcite + Bicarbonate. Although a somewhat reverse relation between CO2 Capture and Mean Soil Calcite

IPCC like diagram

We need to group these files on similarity and get the mean values of those groups.

K-Means We can use the clustering techniques, cluster refers to a collection of data points, aggregated together based on certain similarity metrics. For clustering we define a target number k, which is the number of centroids we want for the dataset. A centroid is the imaginary or real location representing the center of the cluster.

Let's start with K-mean clustering due to it's simplicity and scale-ability.

K-means clustering: K means finds the cetroid through the mean or averaging data points. And then allocating the closest data points to the nearest cluster.

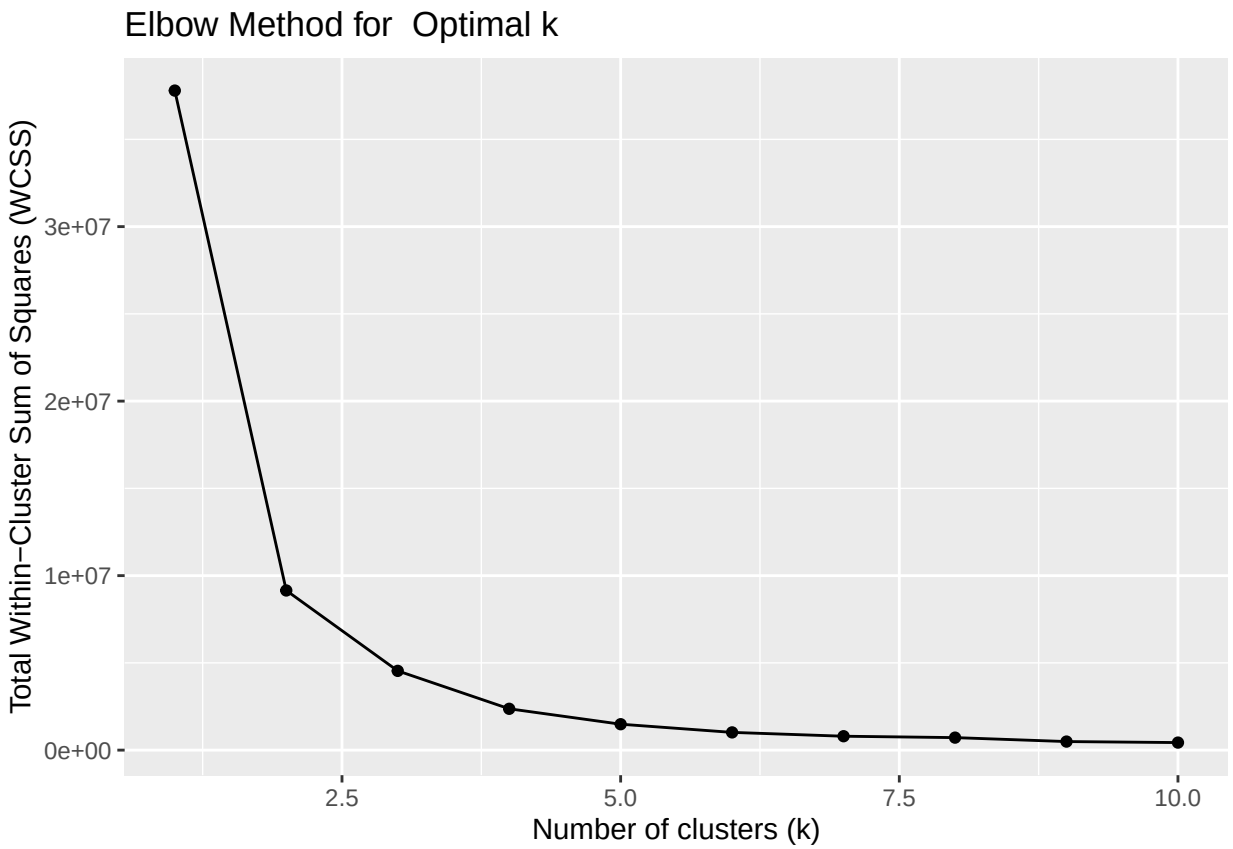
To determine the appropriate number of clusters, we will need to use the elbow method.
What elbow method does is, plots the graph for the **"Within Cluster Sum of Squares"**(WCSS), i.e. sum of the square distance between points in a cluster and the cluster centroid, for values of k from 1 to n.

For our case let's try with n = 10, so we will be getting the WCSS values from 1 to 10, and will pick the value of K at elbow point. Elbow point refer to the point after which the change in WCSS is not significant, or after which point the plot moves almost parallel to the x-axis.

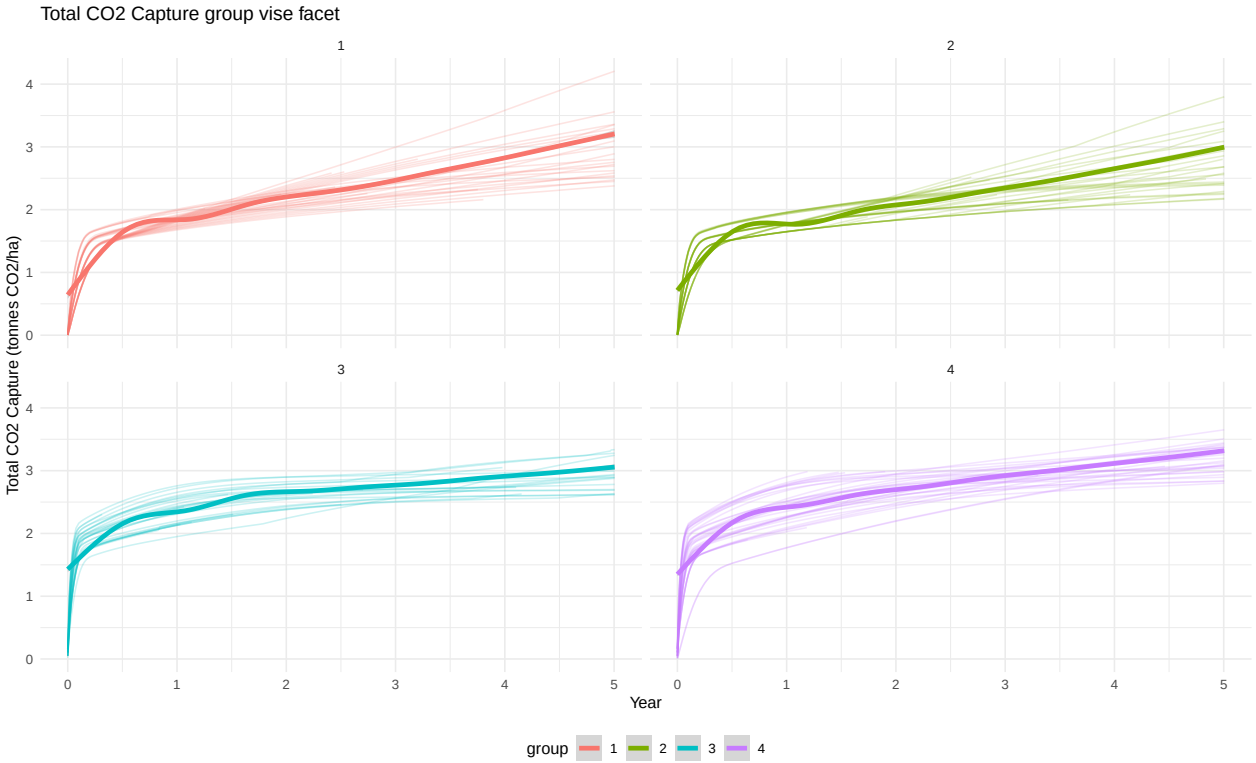
```
data_for_clustering <- total_capture[, c("Total_CO2_capture", "file_id")]
```

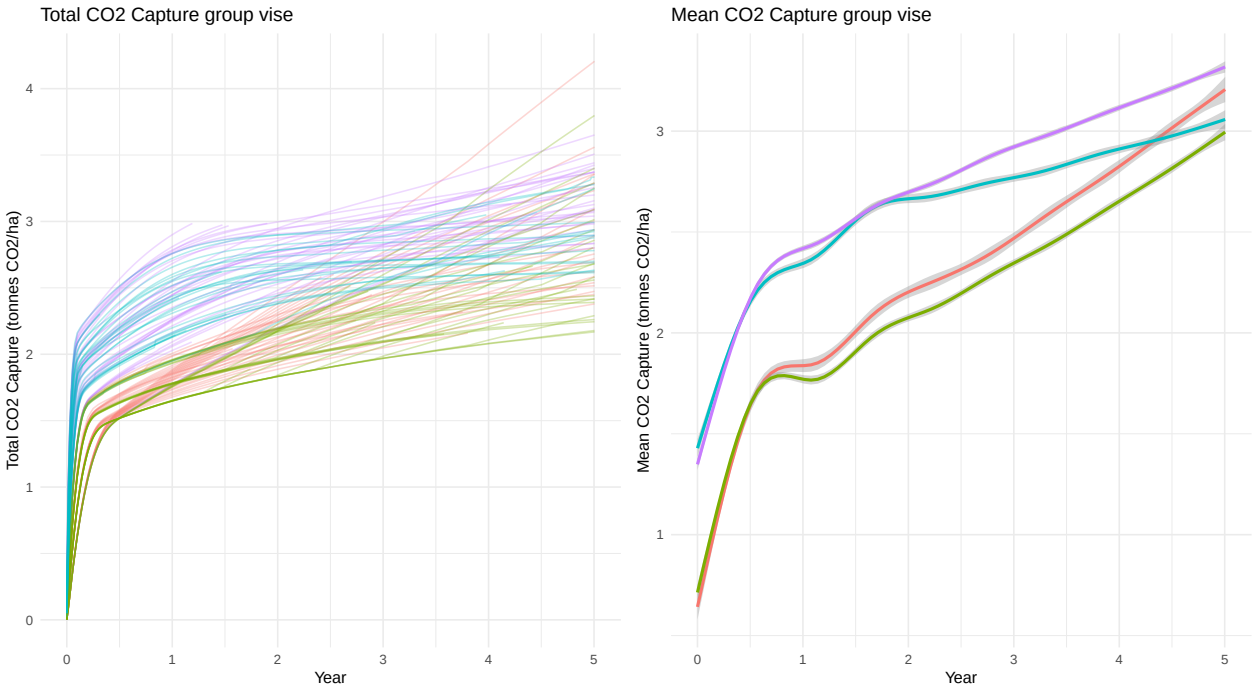
```r
wcss <- sapply(1:10, function(k){
  kmeans(data_for_clustering, centers = k)$tot.withinss
})
```

Now let's take a look at the plot we get from this:



From the plot above, 4 seems to be the elbow point, so let's use it as the value of K.



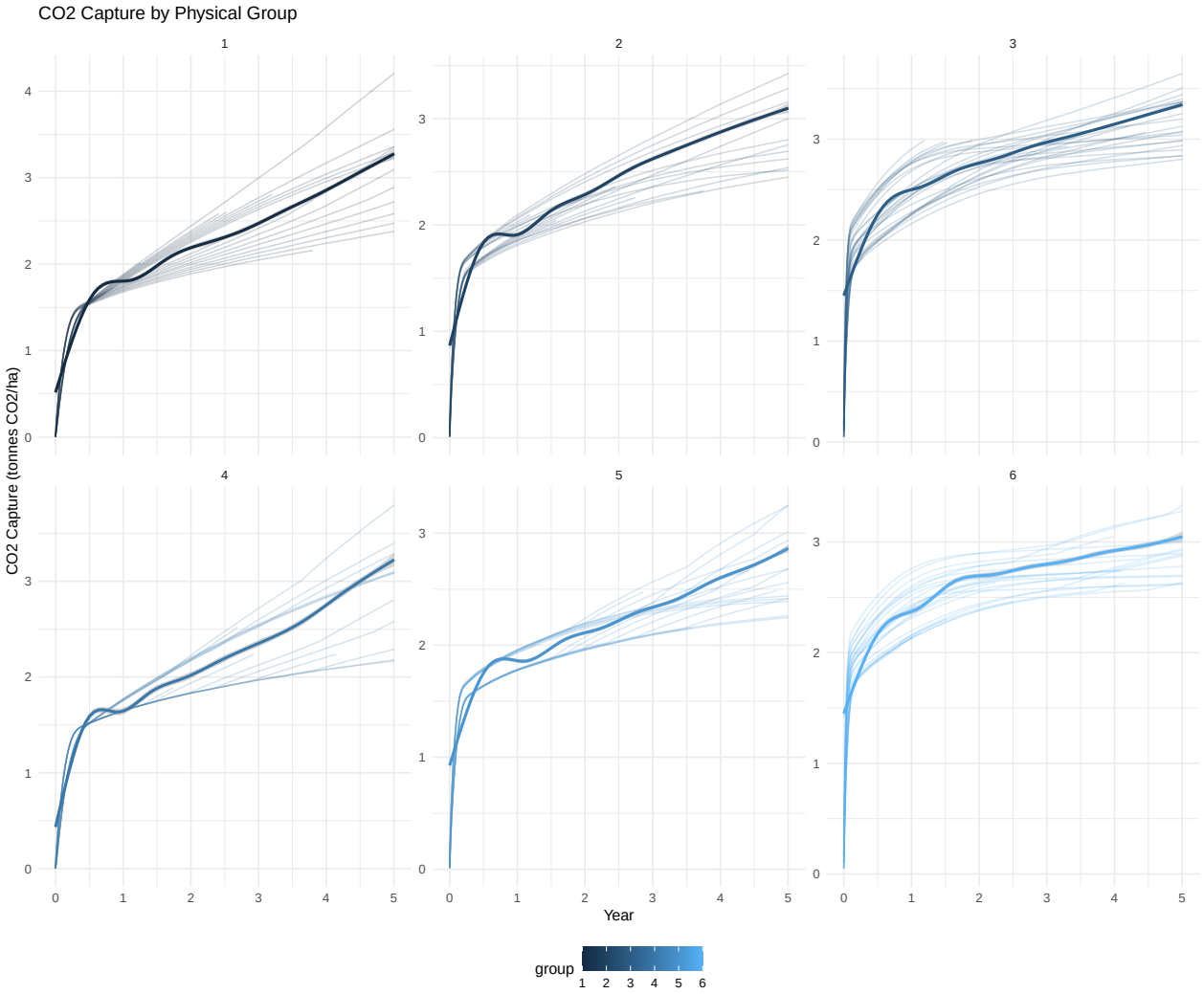


Let's also take a look at the groups created by physical meaning

Plot from Physical Groups

We can also cluster the files based on their physical properties. That means using the values from columns **Solution**(2 distinct values) and **Temperature**(3 distinct values), we can get 6(2 x 3) different clusters.

We can then take a look at the similarities between the k-means clusters and the physical clusters.



Comparisons for k-means and physical clusters

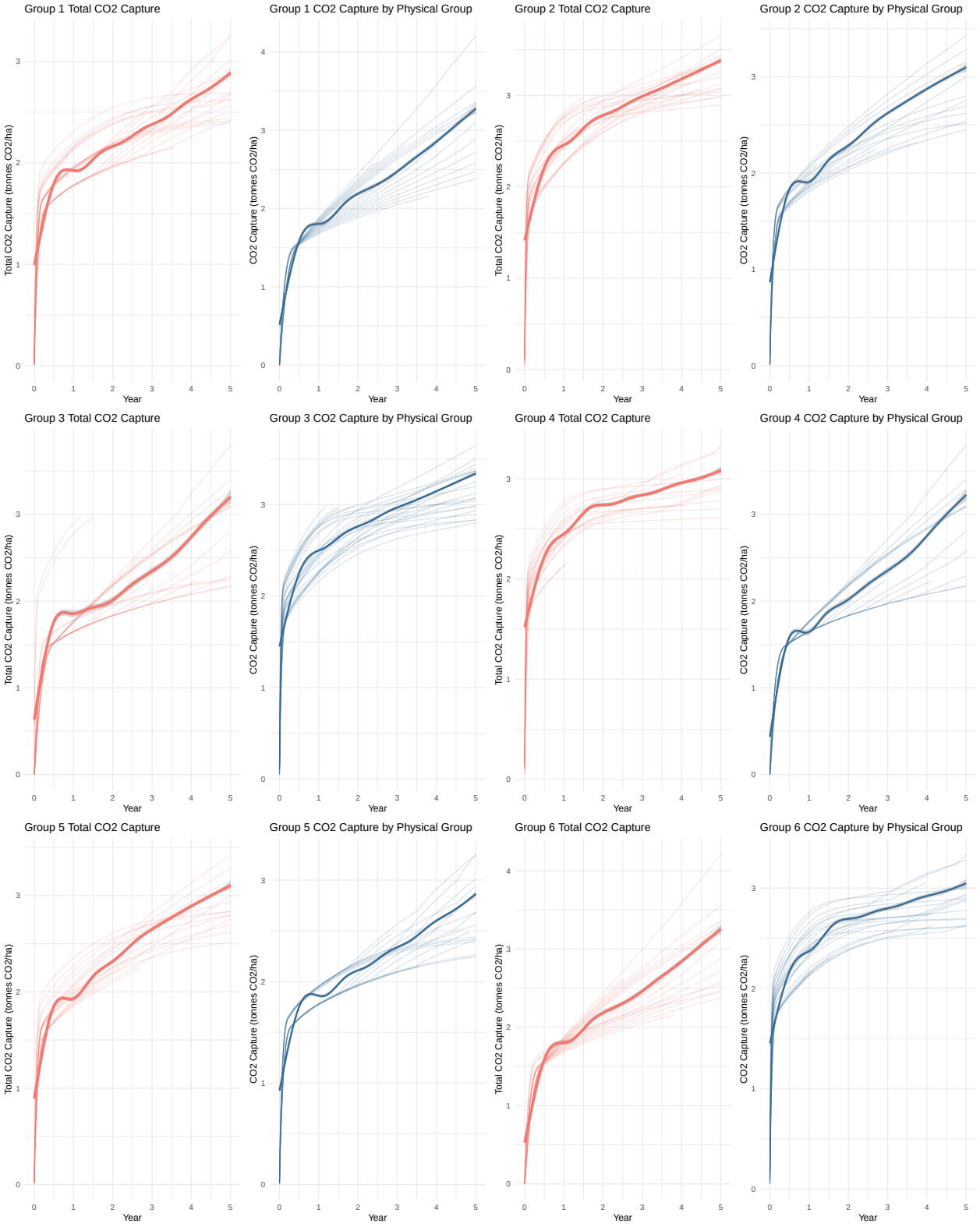
Since we have 6 different clusters for the *physical clustering*, let's change our k-means clusters to 6 for the sake of comparison:

```
clusters <- kmeans(data_for_clustering, centers = 6)

kmean_total_caputer.revised <- total_capture %>%
  mutate(group = as.factor(clusters$cluster[match(file_id, data_for_clustering$file_id)]))

group_mean <- kmean_total_caputer.revised |>
  group_by(year, group) |>
  summarize(mean_CO2 = mean(Total_CO2_capture, na.rm = TRUE), .groups = 'drop')
```

Now let's take a look at the side by side facet plot:



We can see that although there might be some differences in the grouping number, but mostly the k-means clusters are identical to the ones we get through the physical clustering.

Now let's take a look from these clusters what can we predict about the importance of different parameters. And take a look at what are the more important parameters for the prediction of the CO2 absorption levels.

Correlation