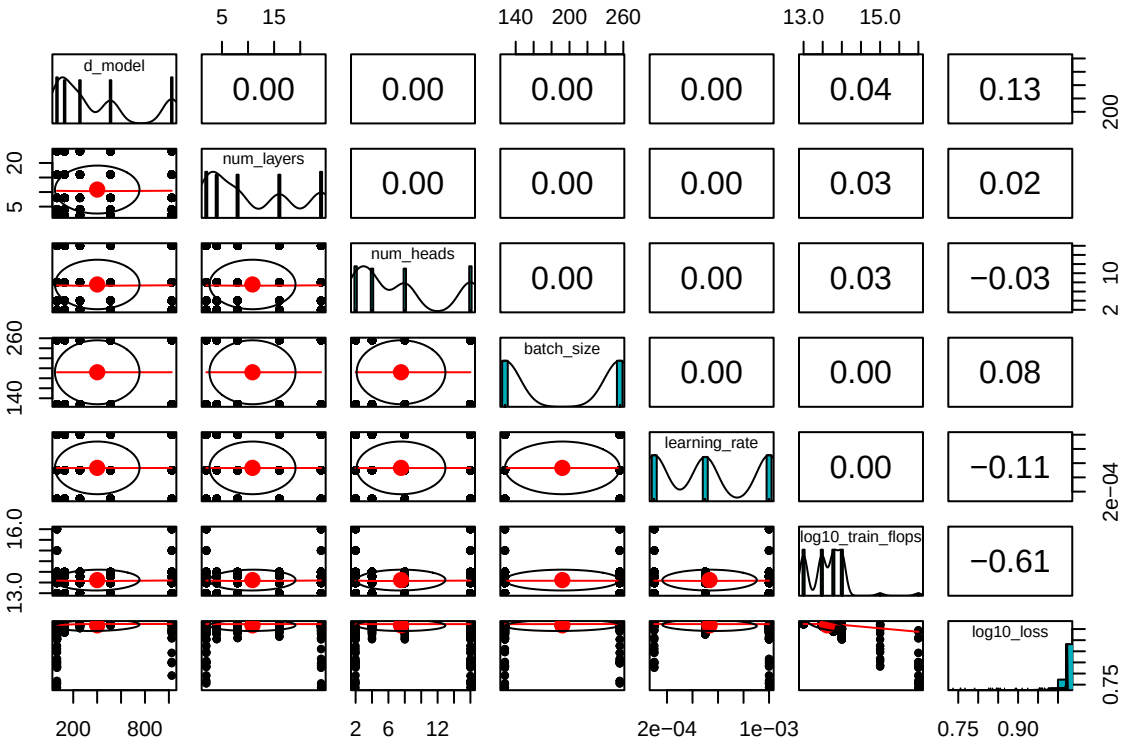
EDA

2024-05-07

cor plots

We first query select some data with training_flops less then 1e16, which corresponds to 23.6% of total budget. Let's look at the correlation plots.

```
pairs.panels(df,
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
)
```



As expected, training flops is the biggest contributor.

argmin cor

Let's take the argmin over all hyperparameters for each training flops, see what the result look like.

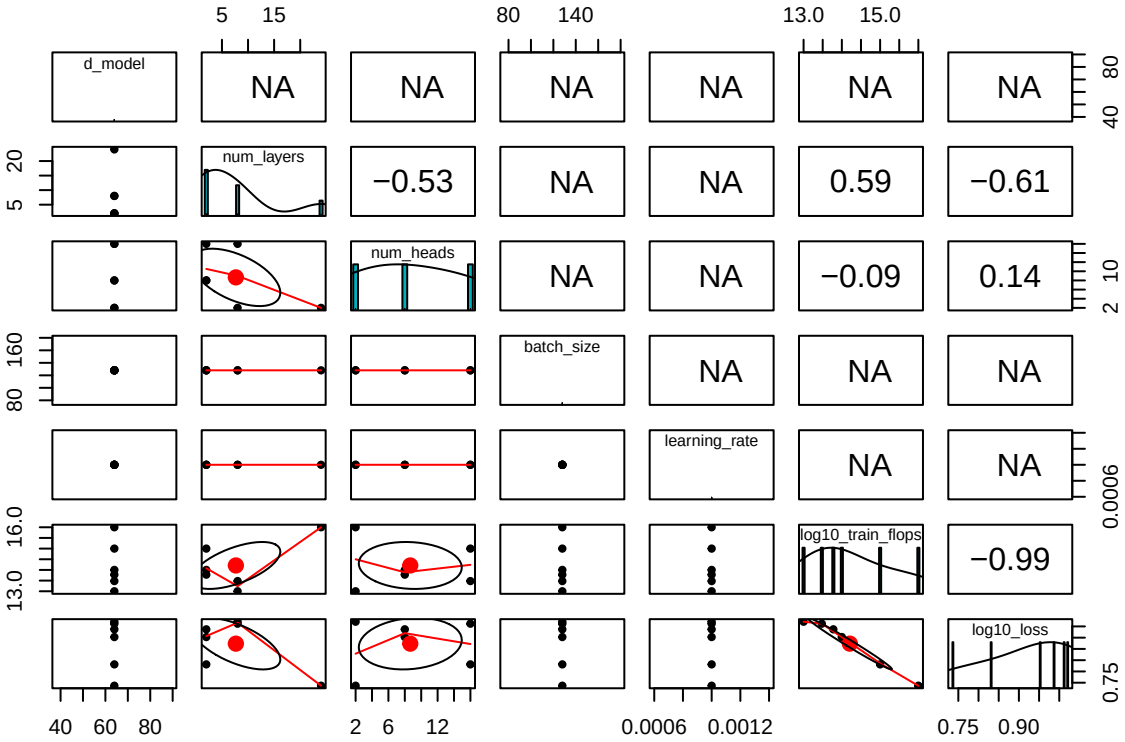
```
# Function to find the row with the smallest loss for each unique log10_train_flops
get_min_loss_per_flop = function(df) {
  # Find unique values of log10_train_flops
  unique_flops = unique(df$log10_train_flops)
```

```
  # Initialize an empty data.table to store results
  result = data.table()

  # Iterate over each unique log10_train_flops value
  for (flop in unique_flops) {
    # Subset the data.table for the current log10_train_flops
    subset_dt = df[log10_train_flops == flop]

    # Find the row with the smallest loss within the subset
    min_loss_row = subset_dt[which.min(log10_loss)]

    # Bind the row with the smallest loss to the result data.table
    result = rbind(result, min_loss_row)
  }

  return(result)
}

# Assuming df is your data.table
argmin_df = get_min_loss_per_flop(df)
argmin_df
```

```
##    d_model num_layers num_heads batch_size learning_rate log10_train_flops
## 1:      64          8         2        128         0.001          13.00000
## 2:      64          8        16        128         0.001          13.47712
## 3:      64          2         8        128         0.001          13.77815
## 4:      64          2         8        128         0.001          14.00000
## 5:      64          2        16        128         0.001          15.00000
## 6:      64         24         2        128         0.001          16.00000
##    log10_loss
## 1:  1.0208374
## 2:  1.0120776
## 3:  0.9870799
## 4:  0.9527515
## 5:  0.8313305
## 6:  0.7365705
```

We observe the model consistently favors small batch_size, large learning rate, and small d_model.

```
pairs.panels(argmin_df,
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = TRUE # show correlation ellipses
)
```



As we can see, `loss ~ train_flops` follows an almost perfect line.