EPC: question 7

Madeline Chun

2024-10-27

combine the columns

```
get_mode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

is_outlier <- function(x) {
  q1 <- quantile(x, 0.25)
  q3 <- quantile(x, 0.75)
  iqr <- q3 - q1
  (x < (q1 - 1.5 * iqr)) | (x > (q3 + 1.5 * iqr))
}

times <- times %>%
  rowwise() %>%
  mutate(
    combined_value = ifelse(
      any(is_outlier(c_across(minutes_after:...10))),
      get_mode(c_across(minutes_after:...10)),  # Use mode if there are outliers
      mean(c_across(minutes_after:...10), na.rm = TRUE)  # Use mean if no outliers
    )
  )

times <- times %>%
  select(-c(3:10))
```

merging with presidential race margin

```
pres <- pres %>%
  filter(candidatevotes >= 10000)

dem_votes <- pres %>%
  filter(party_simplified == "DEMOCRAT") %>%
  select(year, state, state_po, totalvotes, d_votes = candidatevotes)

rep_votes <- pres %>%
  filter(party_simplified == "REPUBLICAN") %>%
  select(year, state, state_po, totalvotes, r_votes = candidatevotes)

combined_data <- left_join(
  dem_votes, rep_votes, by = c("year", "state", "state_po", "totalvotes")
  )

combined_data <- combined_data %>%
  mutate(percentage_margin = ((r_votes - d_votes) / (totalvotes)) * 100)

combined_data <- combined_data %>%
  filter(state_po != "DC")

# Merge the two data frames by year and state
times <- times %>%
  mutate(state = toupper(state))

merged_data <- merge(combined_data, times, by = c("year", "state"))

# Assuming merged_data already contains the percentage_margin column
merged_data <- merged_data %>%
  mutate(absolute_percentage_margin = abs(percentage_margin))
```

outliers

```
# Calculate Z-scores for combined_value
merged_data <- merged_data %>%
  mutate(z_score = (combined_value - mean(combined_value, na.rm = TRUE)) / sd(combined_value, na.rm = T

# View outliers where z-score > 3
outliers <- merged_data %>%
  filter(z_score > 3)

# Print the outliers
print(outliers)
```

```
##   year          state state_po totalvotes d_votes r_votes percentage_margin
## 1 2020         ALASKA       AK     359530  153778  189951        10.0611910
## 2 2020 NORTH CAROLINA       NC    5524802 2684292 2758773         1.3481207
## 3 2020      WISCONSIN       WI    3298041 1630866 1610184        -0.6270995
##   combined_value absolute_percentage_margin  z_score
## 1      10074.000                 10.0611910 8.600339
## 2      11130.000                  1.3481207 9.523883
## 3       4199.625                  0.6270995 3.462798
```

```
# Optionally, filter out the outliers from the original data
cleaned_data <- merged_data %>%
  filter(z_score <= 3)
```

regress

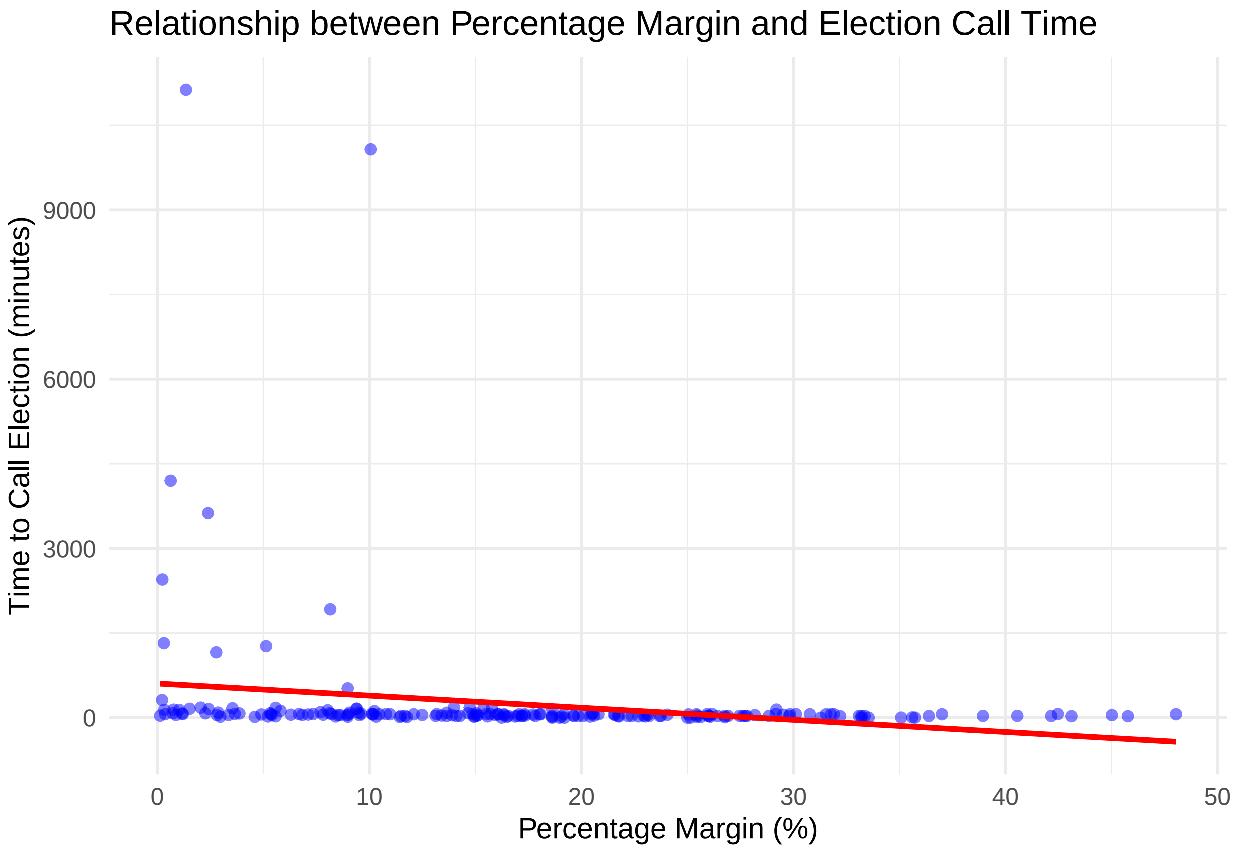
```
model <- lm(combined_value ~ absolute_percentage_margin, data = merged_data)
model1 <- lm(combined_value ~ absolute_percentage_margin, data = cleaned_data)
model2 <- lm(combined_value ~ poly(absolute_percentage_margin, 2), data = merged_data)

merged_data$combined_value_adjusted <- merged_data$combined_value + 0.0001
```

```r
# Fit the exponential model using the adjusted combined value
exp_model <- lm(log(combined_value_adjusted) ~ absolute_percentage_margin, data = merged_data)

stargazer(model, model1, model2, exp_model,
          type = "text",   # You can use "html" or "latex" for other output formats
          title = "Comparison of Model and Model1",
          dep.var.labels = "Time to Call Election (minutes)",
          column.labels = c("Model", "Model1", "Polynomial", "Exp"),
          model.numbers = FALSE,  # Suppresses model numbers in the output
          star.cutoffs = c(0.05, 0.01, 0.001)  # Significance stars
          )
```

```
## 
## Comparison of Model and Model1
## =====================================================================================================
##                                                                             Dependent variable:      
##                                       ---------------------------------------------------------------
##                                                         Time to Call Election (minutes)              
##                                               Model                Model1               Polynomial   
## -----------------------------------------------------------------------------------------------------
## absolute_percentage_margin                  -21.475**             -8.722***                          
##                                              (7.277)               (2.320)                           
##                                                                                                      
## poly(absolute_percentage_margin, 2)1                                                   -3,310.665**  
##                                                                                         (1,111.156)  
##                                                                                                      
## poly(absolute_percentage_margin, 2)2                                                    2,446.409*   
##                                                                                         (1,111.156)  
##                                                                                                      
## Constant                                   604.957***            264.772***              240.187**   
##                                             (146.878)             (47.149)               (78.571)    
##                                                                                                      
## -----------------------------------------------------------------------------------------------------
## Observations                                  200                   197                     200      
## R2                                           0.042                 0.068                   0.065     
## Adjusted R2                                  0.037                 0.063                   0.056     
## Residual Std. Error                   1,121.900 (df = 198)    353.361 (df = 195)    1,111.156 (df = 19
## F Statistic                           8.708** (df = 1; 198) 14.135*** (df = 1; 195) 6.862** (df = 2; 1
## =====================================================================================================
## Note:
```

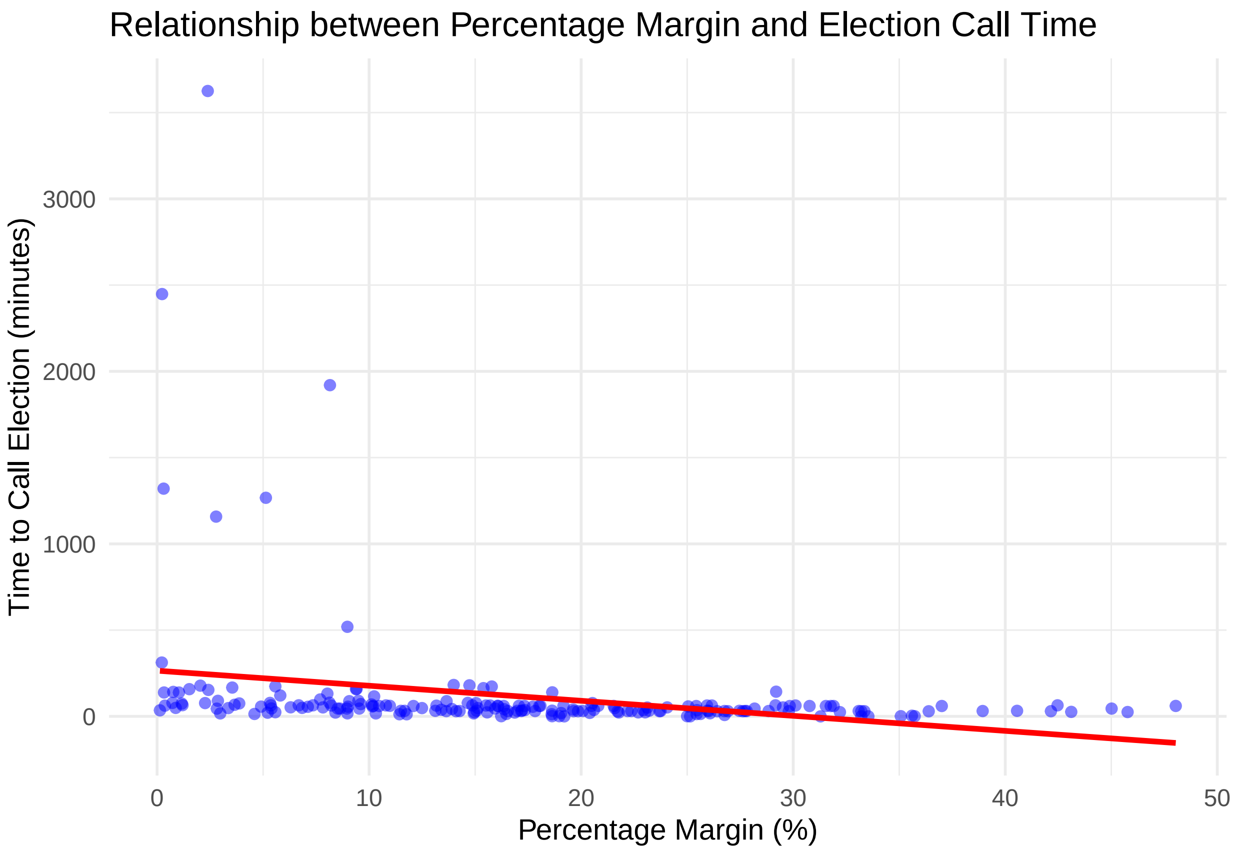
```r
# Create a scatter plot of percentage margin vs. times
ggplot(merged_data, aes(x = absolute_percentage_margin, y = combined_value)) +
  geom_point(color = "blue", alpha = 0.5) +  # Add points
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  labs(title = "Relationship between Percentage Margin and Election Call Time",
       x = "Percentage Margin (%)",
       y = "Time to Call Election (minutes)") +
  theme_minimal()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```r
ggplot(cleaned_data, aes(x = absolute_percentage_margin, y = combined_value)) +
  geom_point(color = "blue", alpha = 0.5) +  # Add points
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  labs(title = "Relationship between Percentage Margin and Election Call Time",
       x = "Percentage Margin (%)",
       y = "Time to Call Election (minutes)") +
  theme_minimal()
```

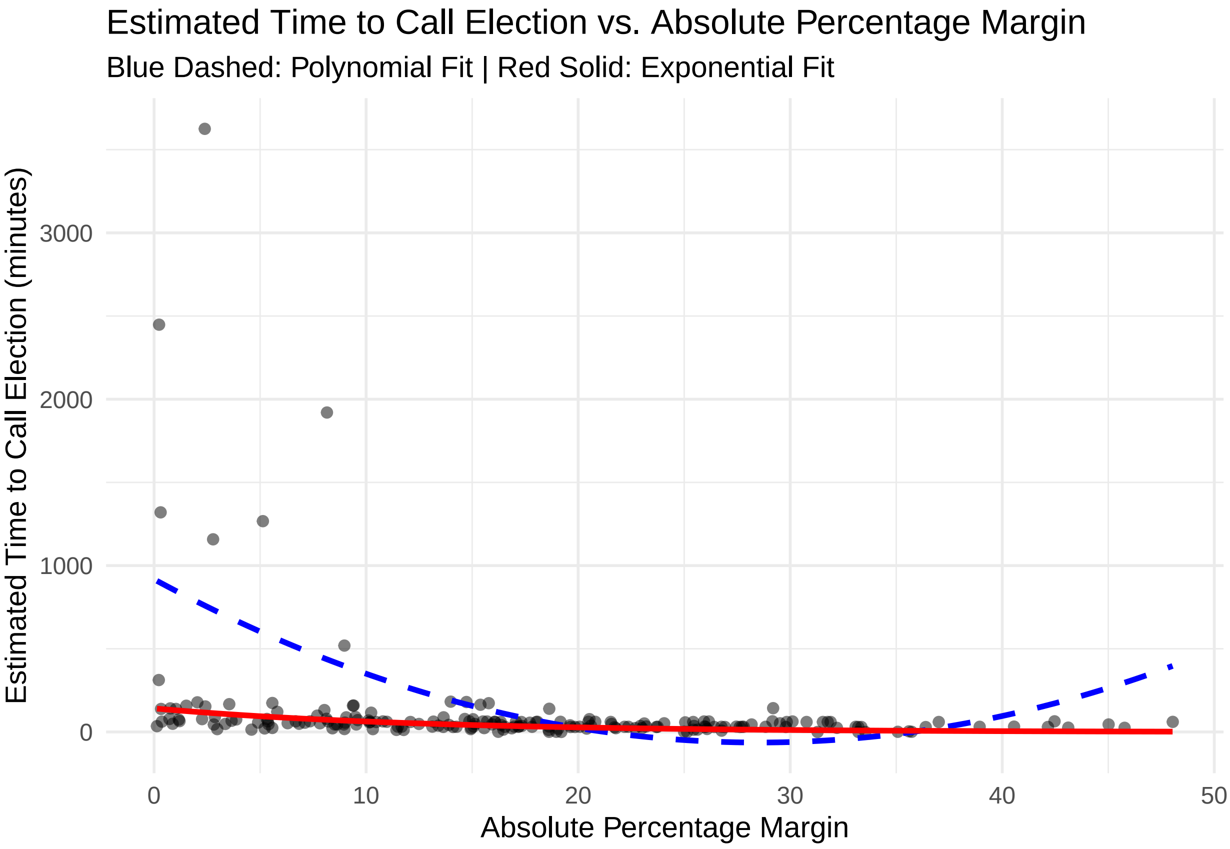
```
## `geom_smooth()` using formula = 'y ~ x'
```


Relationship between Percentage Margin and Election Call Time

```
# Fit the polynomial model
poly_model <- lm(combined_value ~ poly(absolute_percentage_margin, 2), data = merged_data)

# Add a small constant to handle zero values
merged_data$combined_value_adjusted <- merged_data$combined_value + 0.0001

# Fit the exponential model using the adjusted combined value
exp_model <- lm(log(combined_value_adjusted) ~ absolute_percentage_margin, data = merged_data)

# Create a sequence for absolute_percentage_margin
margins <- seq(min(cleaned_data$absolute_percentage_margin), max(cleaned_data$absolute_percentage_margi

# Predict using polynomial model
poly_predictions <- predict(poly_model, newdata = data.frame(absolute_percentage_margin = margins))

# Predict using exponential model
exp_predictions <- exp(predict(exp_model, newdata = data.frame(absolute_percentage_margin = margins)))


# Combine predictions into a data frame for plotting
predictions_df <- data.frame(
  absolute_percentage_margin = margins,
  polynomial_fit = poly_predictions,
  exponential_fit = exp_predictions
)

# Create the plot
```

```
ggplot(cleaned_data, aes(x = absolute_percentage_margin, y = combined_value)) +
  geom_point(alpha = 0.5) +  # Original data points
  geom_line(data = predictions_df, aes(x = absolute_percentage_margin, y = polynomial_fit), color = "blu
  geom_line(data = predictions_df, aes(x = absolute_percentage_margin, y = exponential_fit), color = "re
  labs(title = "Estimated Time to Call Election vs. Absolute Percentage Margin",
       x = "Absolute Percentage Margin",
       y = "Estimated Time to Call Election (minutes)",
       subtitle = "Blue Dashed: Polynomial Fit | Red Solid: Exponential Fit") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



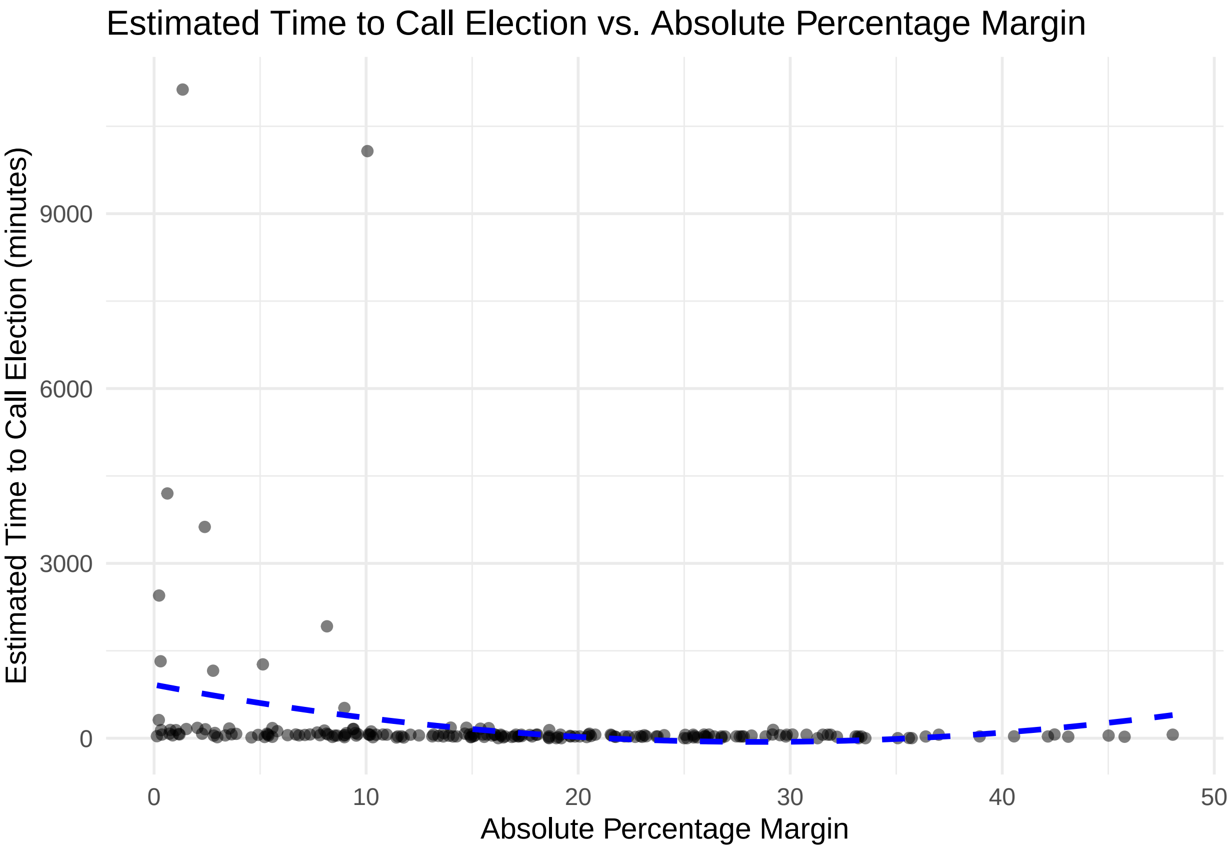
predict

```
intercept <- coef(exp_model)[1]  # Intercept (beta0)
slope <- coef(exp_model)[2]       # Slope (beta1)

predict_combined_value <- function(absolute_percentage_margin) {
  # Calculate the predicted log value
  log_predicted_value <- intercept + slope * absolute_percentage_margin

  # Exponentiate to get the combined value and subtract the constant
  predicted_value <- exp(log_predicted_value) - 0.0001  # Subtract the small constant

  return(predicted_value)
}

# Example usage of the function
# Replace 'some_margin' with the actual margin value you want to predict for
some_margin <- 0.1
predicted_combined_value <- predict_combined_value(some_margin)
print(predicted_combined_value)
```

```
## (Intercept)
##    141.5771
```

alternate polynomial prediction

```
# Fit the polynomial model
poly_model <- lm(combined_value ~ poly(absolute_percentage_margin, 2), data = merged_data)

# Extract coefficients from the polynomial model
poly_coefficients <- coef(poly_model)  # Change the variable name to avoid conflict

# Create a sequence for absolute_percentage_margin for predictions
margins <- seq(min(merged_data$absolute_percentage_margin), max(merged_data$absolute_percentage_margin)

# Predict using the polynomial model
poly_predictions <- predict(poly_model, newdata = data.frame(absolute_percentage_margin = margins))

# Combine predictions into a data frame for plotting
predictions_df <- data.frame(
  absolute_percentage_margin = margins,
  polynomial_fit = poly_predictions
)

# Create the plot
ggplot(merged_data, aes(x = absolute_percentage_margin, y = combined_value)) +
  geom_point(alpha = 0.5) +  # Original data points
  geom_line(data = predictions_df, aes(x = absolute_percentage_margin, y = polynomial_fit), color = "blu
  labs(title = "Estimated Time to Call Election vs. Absolute Percentage Margin",
       x = "Absolute Percentage Margin",
       y = "Estimated Time to Call Election (minutes)") +
  theme_minimal()
```



```
# Example margin to predict
some_margin <- 2.0  # Example absolute percentage margin

# Create a function to predict combined_value based on absolute_percentage_margin
predict_combined_value_poly <- function(absolute_percentage_margin) {
  # Calculate the predicted combined_value using the polynomial equation
  # Use the renamed coefficients variable
  predicted_value <- poly_coefficients[1] +
                     poly_coefficients[2] * absolute_percentage_margin +
                     poly_coefficients[3] * (absolute_percentage_margin^2)

  return(predicted_value)
}

# Use predict function to get predicted value from the model directly
predicted_value_from_model <- predict(poly_model, newdata = data.frame(absolute_percentage_margin = som

# Use custom function to predict value
predicted_combined_value_poly <- predict_combined_value_poly(some_margin)

# Print both values for comparison
cat("Predicted value from model (predict function):", predicted_value_from_model, "\n")
```

```
## Predicted value from model (predict function): 784.6278
```

```
cat("Predicted value from custom function:", predicted_combined_value_poly, "\n")
```

```
## Predicted value from custom function: 3404.495
```