Tidy Tuesday: Modeling Student Debt Inequality

Shafqat Shafiq

2025-03-09

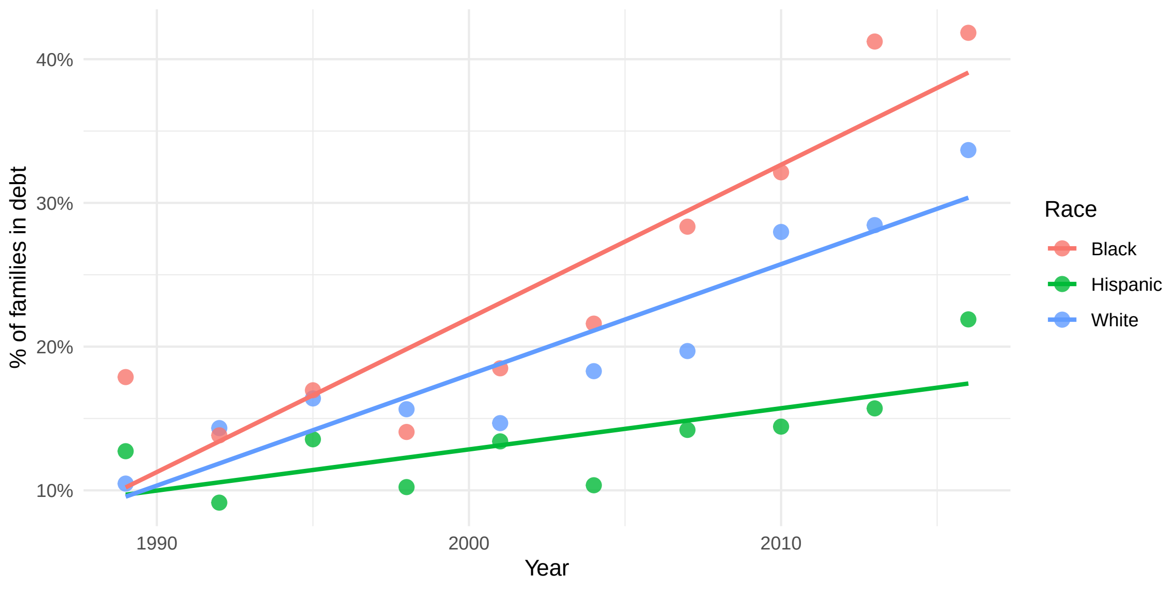
1 Explore Data:

```r
# Load data
url <- paste0(
  "https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/",
  "2021/2021-02-09/student_debt.csv"
)

student_debt <- read_csv(url)
```

Let's visualize the data.

```r
# Visualize
student_debt %>% ggplot(aes(x = year, y = loan_debt_pct, color = race)) +
  geom_point(size = 3, alpha = 0.8) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(y = "% of families in debt",
       color = "Race", x = "Year") +
  scale_y_continuous(labels = scales::percent)
```



2 Build a Linear Model to capture the effect of Race and Year:

```
lmod <- lm(loan_debt_pct ~ 0 + year*race, data = student_debt)

lmod %>% tidy()
```

```
# A tibble: 6 x 5
  term               estimate std.error statistic      p.value
  <chr>                 <dbl>     <dbl>     <dbl>        <dbl>
1 year                0.0107    0.00129      8.29 0.0000000166
2 raceBlack         -21.2       2.58        -8.20 0.0000000204
3 raceHispanic       -5.60      2.58        -2.17 0.0402
4 raceWhite         -15.2       2.58        -5.90 0.00000437
5 year:raceHispanic  -0.00783   0.00182     -4.29 0.000250
6 year:raceWhite     -0.00299   0.00182     -1.64 0.114
```

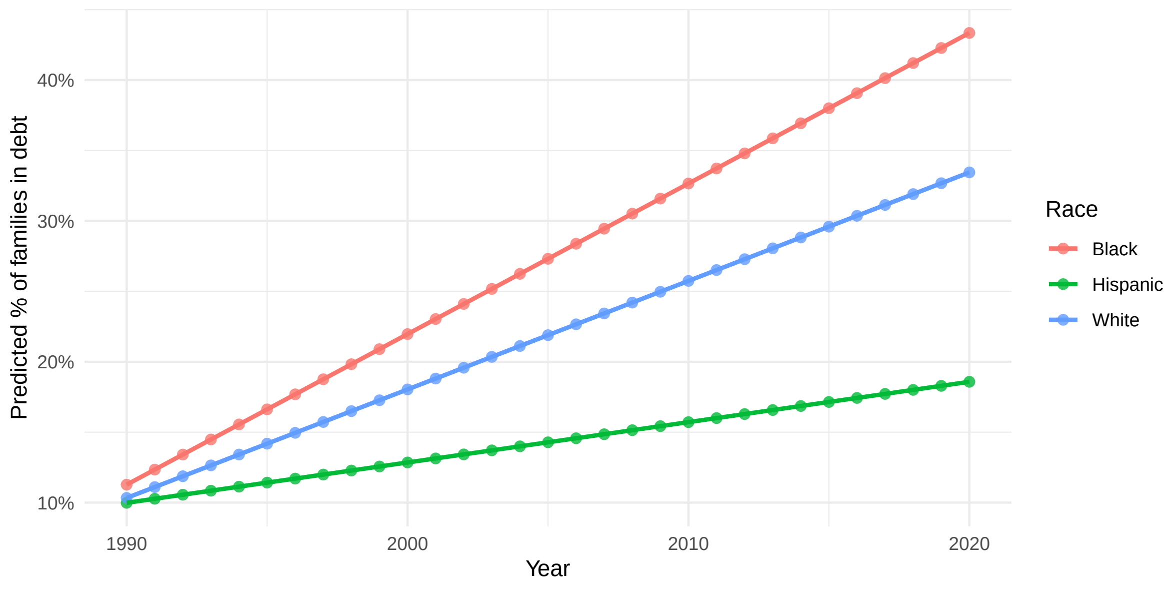
Interaction terms are often difficult to interpret in general. The best way to deal with them is to not just look at the coefficients directly, but instead by exploring the model results using simulated data, visualizations, and tables.

Let's first use the `augment()` function to fit the model on existing data.

```
# Create a new dataframe comprised of simulated year and race.
new_points <- crossing(
  race = c("Black", "White", "Hispanic"),
  year = 1990:2020
)
```

Let's now visualize the fitted data.

```
# Augment lmod to new dataframe and then visualize
augment(lmod, newdata = new_points) %>%
  ggplot(aes(x = year, y = .fitted, color = race)) +
  geom_point(size = 2, alpha = 0.8) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(y = "Predicted % of families in debt",
       color = "Race", x = "Year") +
  scale_y_continuous(labels = scales::percent)
```



3 Visualize the effects:

Let's now visualize the effects.

```
# Hold the results in a dataframe called results
results <- lmod %>% tidy()

results
```

```
# A tibble: 6 x 5
  term               estimate std.error statistic      p.value
  <chr>                 <dbl>     <dbl>     <dbl>        <dbl>
1 year                 0.0107    0.00129      8.29 0.0000000166
2 raceBlack          -21.2       2.58        -8.20 0.0000000204
3 raceHispanic        -5.60      2.58        -2.17 0.0402
4 raceWhite          -15.2       2.58        -5.90 0.00000437
5 year:raceHispanic  -0.00783   0.00182     -4.29 0.000250
6 year:raceWhite     -0.00299   0.00182     -1.64 0.114
```

We have the following columns in the table:

- `term`
- `estimate`
- `std.error`
- `statistics`
- `p.value`

We want to create a plot that shows:

1. The estimate coefficient (`estimate`)
2. The uncertainty around those estimates using `std.error`
3. A visual indication of statistical significance

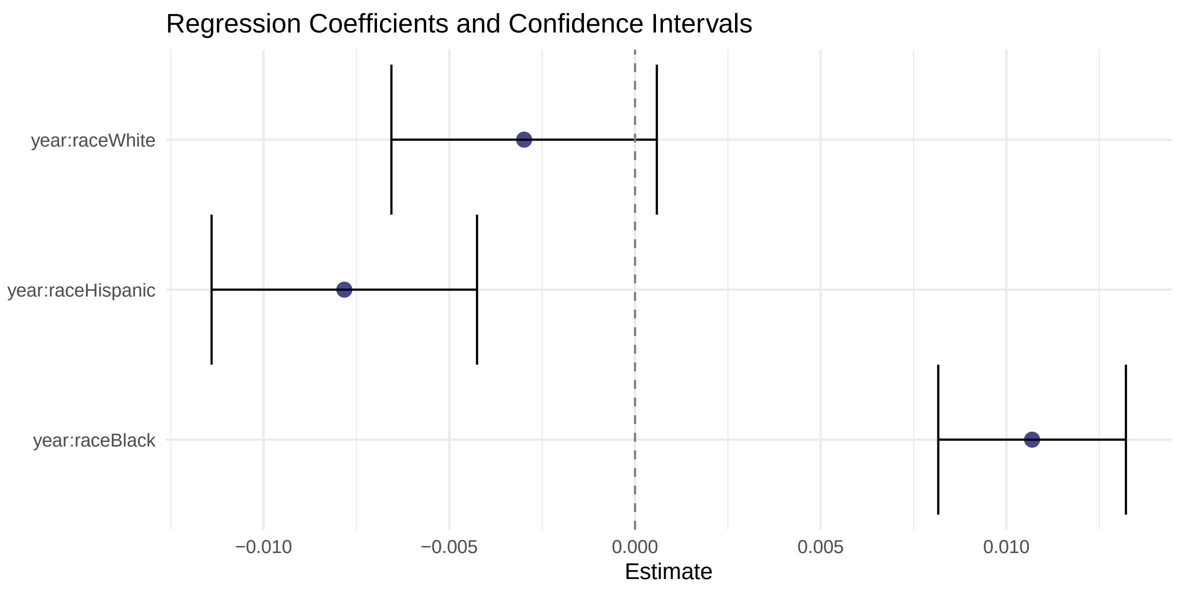
We do it using:

- `ggplot()` with the y-axis displaying the `term`, and x-axis the `estimate`

- `geom_point()` for the grammar of graphics
- `geom_errorbar()` for the uncertainty; use `xmin()` and `xmax()`
- `geom_hline()` for the y-intercept (to be at 0)

```r
# Visualize the results

p1 <- results %>%
  filter(str_detect(term, "year")) %>%
  mutate(term = case_when(term == "year" ~ "year:raceBlack",
                          TRUE ~ term)) %>%
  ggplot(aes(x = estimate, y = term)) +
  geom_point(size = 3, alpha = 0.8) +
  geom_errorbar(aes(xmin = estimate - 1.96 * std.error,
              xmax = estimate + 1.96 * std.error,
              width = 1)) + # Corrected linewidth
  geom_vline(xintercept = 0, linetype = "dashed", color = "gray50") +
  labs(title = "Regression Coefficients and Confidence Intervals",
       y = NULL,
       x = "Estimate"); p1
```



4 Interpretation of Regression Coefficients:

Model: `loan_debt_pct ~ 0 + year*race`

Coefficients:

- **`year` (0.0107):**
  - For Black families, the percentage of families with student loan debt increases by approximately 1.07 percentage points each year.
- **`raceBlack` (-21.2):**
  - At the year zero point, Black families have a loan debt percentage of -21.2 percentage points.
- **`raceHispanic` (-5.60):**
  - At the year zero point, Hispanic families have a loan debt percentage of -5.60 percentage points.
- **`raceWhite` (-15.2):**
  - At the year zero point, White families have a loan debt percentage of -15.2 percentage points.
- **`year:raceHispanic` (-0.00783):**
  - The yearly increase in loan debt percentage for Hispanic families is about 0.78 percentage points less than the yearly increase for Black families.
- **`year:raceWhite` (-0.00299):**
  - The yearly increase in loan debt percentage for White families is about 0.3 percentage points less than the yearly increase for Black families.

Summary:

- Black families experience a consistent annual increase in student loan debt percentage.
- The `race` coefficients directly represent the estimated loan debt percentage for each racial group at the year zero point.
- The interaction terms represent the difference in the rate of change of loan debt over time when comparing Hispanic and White families to Black families.

5 Bootstrap Estimates and Confidence Intervals:

More often than not, in real-world data analysis, the Gauss-Markov assumptions—which are the foundation of Ordinary Least Squares (OLS) regression—fail to hold. These assumptions include linearity, no perfect multicollinearity, exogeneity of errors, homoscedasticity (constant variance of errors), and no autocorrelation. When these assumptions are met, OLS provides the Best Linear Unbiased Estimator (BLUE), meaning it minimizes variance among all linear estimators.

However, real-world data is rarely ideal. Skewed distributions, heteroscedastic errors, omitted variable bias, measurement errors, and autocorrelated data frequently violate these conditions. For instance, economic and financial data often exhibit non-constant variance, while time series data suffers from autocorrelation. In such cases, OLS estimates become inefficient, biased, or misleading. This is where alternative estimation techniques, such as bootstrap resampling, become essential, as they allow for robust inference without relying on strict parametric assumptions.

Bootstrap estimates and confidence interval building involve resampling the observed data with replacement to create multiple simulated datasets. Each bootstrap sample is used to recompute the statistic of interest (e.g., regression coefficients), generating an empirical distribution of the estimator. This process allows for a more flexible approach to quantifying uncertainty without relying on strong parametric assumptions like normality or homoscedasticity.

Confidence intervals are then constructed by taking percentiles from the bootstrap distribution (e.g., the 2.5th and 97.5th percentiles for a 95% confidence interval). Unlike traditional OLS-based confidence intervals, which assume normality, bootstrap intervals remain valid even when the data is skewed, heteroscedastic, or contains outliers, making them a powerful alternative in real-world applications.

Bootstrapping regression coefficients not only provides more robust estimates by reducing sensitivity to violations of OLS assumptions, but it also leads to tighter confidence interval (CI) bounds by better capturing the true variability in the data. Traditional OLS confidence intervals rely on asymptotic normality and homoscedastic errors, which can lead to overly wide or misleading intervals when these assumptions are violated.

In contrast, bootstrap confidence intervals are data-driven and reflect the actual distribution of coefficient estimates, rather than relying on theoretical approximations. By repeatedly resampling from the observed dataset, bootstrapping accounts for small sample effects, skewness, and heteroscedasticity, leading to more precise and realistic confidence intervals. As a result, these intervals often provide a more accurate reflection of parameter uncertainty, improving inference in cases where OLS methods struggle.

5.0.1 Steps in Bootstrapping Regression Coefficients

1. **Extract Original Coefficients:**
   - Compute OLS estimates from the model (`coef(lmod)`).
   - These serve as the baseline for comparison.
2. **Set the Number of Bootstrap Resamples:**
   - Define how many times to resample the data (`n_resamples <- 1000`).
   - A higher number improves accuracy but increases computation time.
3. **Initialize Storage for Bootstrap Estimates:**
   - Create a matrix (`boot_matrix`) to store resampled coefficients.
   - Rows = Number of bootstrap samples, Columns = Number of regression coefficients.
4. **Prepare for Resampling:**
   - Capture the original dataset size (`original_sample_size <- nrow(student_debt)`).
   - This ensures each resample matches the original dataset size.

```r
set.seed(1000)

original_coefs <- coef(lmod)

# set up the number of bootstrap resamples
n_resamples <- 1000

# set up a matrix to hold the bootstrap coefficients
  # nrows = number of resamples
  # ncols = number of regression coefficients

boot_matrix <- matrix(NA, nrow = n_resamples,
                      ncol = length(original_coefs))

# rename the columns of the boot_matrix as the coefficient names
colnames(boot_matrix) <- names(original_coefs)

# collect the number of rows/length of original sample
original_sample_size <- nrow(student_debt)
```

5.0.2 Steps in Bootstrapping Regression Coefficients (Resampling & Model Fitting)

1. **Iterate Over Bootstrap Resamples:**
   - Loop runs `n_resamples` times to generate multiple resampled datasets.
2. **Generate Bootstrap Indices:**
   - Randomly sample row indices (`boot_indices`) from the original dataset with replacement.
   - Ensures each resampled dataset is the same size as the original.
3. **Create Bootstrap Sample:**
   - Use `boot_indices` to extract a new dataset (`boot_sample`).
   - Some original data points may appear multiple times, while others may be absent.
4. **Fit Regression Model to Resampled Data:**
   - Run `lm(loan_debt_pct ~ 0 + year*race, data = boot_sample)`.
   - The interaction model estimates how loan debt percentage varies across years and racial groups.
5. **Store Bootstrap Coefficients:**
   - Extract regression coefficients and save them in `boot_matrix[i, ]`.
   - Each row in `boot_matrix` corresponds to a set of regression coefficients from one bootstrap iteration.
6. **Inspect Bootstrap Results:**
   - Use `boot_matrix %>% head()` to check the first few rows of stored coefficients.

```r
# set up bootstrap for-loop, where we sample 1:n_resamples

for (i in 1:n_resamples){

  # create new bootstrap indices
  boot_indices <- sample(1:original_sample_size,
                         size = original_sample_size,
                         replace = TRUE)

  # create a new bootstrap sample using the bootstrapped indices
  boot_sample <- student_debt[boot_indices, ]
  # fit model to bootstrap sample
  boot_model <- lm(loan_debt_pct ~ 0 + year*race, data = boot_sample)

  # collect coefficients and store them in the i-th row of boot_matrix
  boot_matrix[i, ] <- coef(boot_model)

}

boot_matrix %>% head()
```

```
            year raceBlack raceHispanic raceWhite year:raceHispanic
[1,] 0.009222815 -18.19720    -4.087293 -15.09075      -0.007109655
[2,] 0.010879324 -21.55897    -6.779619 -15.12654      -0.007422247
[3,] 0.010195065 -20.18246    -6.728430 -16.44023      -0.006769613
[4,] 0.014604093 -29.02701   -10.770852 -15.47747      -0.009167521
[5,] 0.008736174 -17.23104    -4.262190 -11.32076      -0.006539078
[6,] 0.009652203 -19.07581    -7.049085 -15.21164      -0.006062523
     year:raceWhite
[1,]   -0.001590596
[2,]   -0.003226611
[3,]   -0.001895372
[4,]   -0.006768941
[5,]   -0.002989323
[6,]   -0.001955838
```

5.0.3 Steps involved with computing Bootstrap Confidence Intervals

1. **Determine Number of Coefficients:**
    - `n_coefs <- length(original_coefs)` retrieves the number of regression coefficients from the original model.
2. **Initialize Vectors for CI and Estimates:**
    - `upper_cis`, `lower_cis`, and `bootstrap_estimates` are created as empty numeric vectors to store the upper and lower confidence intervals and bootstrap estimates for each coefficient.
3. **Iterate Over Each Coefficient:**
    - A for loop runs from `1` to `n_coefs`, allowing for the processing of each coefficient individually.

4. **Extract Bootstrap Values:**
   - `coef_values <- boot_matrix[, coef]` retrieves all bootstrap values corresponding to the current coefficient.
5. **Compute Lower Confidence Interval:**
   - The lower confidence interval for the coefficient is calculated using `quantile(coef_values, 0.025, na.rm = TRUE)`, which gives the 2.5th percentile.
6. **Compute Upper Confidence Interval:**
   - The upper confidence interval is calculated similarly with `quantile(coef_values, 0.975, na.rm = TRUE)`, providing the 97.5th percentile.
7. **Compute Bootstrap Estimate:**
   - The mean of the bootstrap values is computed using `mean(coef_values)`, storing the result in `bootstrap_estimates`.
8. **Store Results in a Data Frame:**
   - The results are compiled into a data frame called `results`, which includes the coefficient names, original estimates, bootstrap estimates, and the calculated lower and upper confidence intervals.

5.0.4 Key Outcome:

The resulting data frame `results` provides a comprehensive view of the original and bootstrap estimates alongside their corresponding confidence intervals, offering insight into the variability and reliability of the regression coefficients derived from the bootstrap sampling process.

```
# get the number of coefficients
n_coefs <- length(original_coefs)

# set up empty vectors to hold upper, lower & boot estimates for each coefficient
upper_cis = lower_cis = bootstrap_estmates = numeric(n_coefs)

# run a for loop, iterating over each coefficient i

for (coef in 1:n_coefs) {

  # Extract the bootstrap values for the coefficient
  coef_values <- boot_matrix[, coef]

  # Compute lower CI for the coefficient
  lower_cis[coef] <- quantile(coef_values, 0.025, na.rm = TRUE)

  # Compute upper CI for the coefficient
  upper_cis[coef] <- quantile(coef_values, 0.975, na.rm = TRUE)

  # Computer mean/Bootstrap estimate
  bootstrap_estmates[coef] <- mean(coef_values)

}

# hold the results in a dataframe

boot_results <- data.frame(

  # Add coefficient names
  term = names(original_coefs),

  # Add original coefficient estimates
  estimate = original_coefs,

  # Add bootstrap estimates
  bootstrap_estimate = bootstrap_estmates,

  # Add Lower CI
  lower_ci = lower_cis,

  # Add Upper CI
  upper_ci = upper_cis

) %>% as_tibble()
```

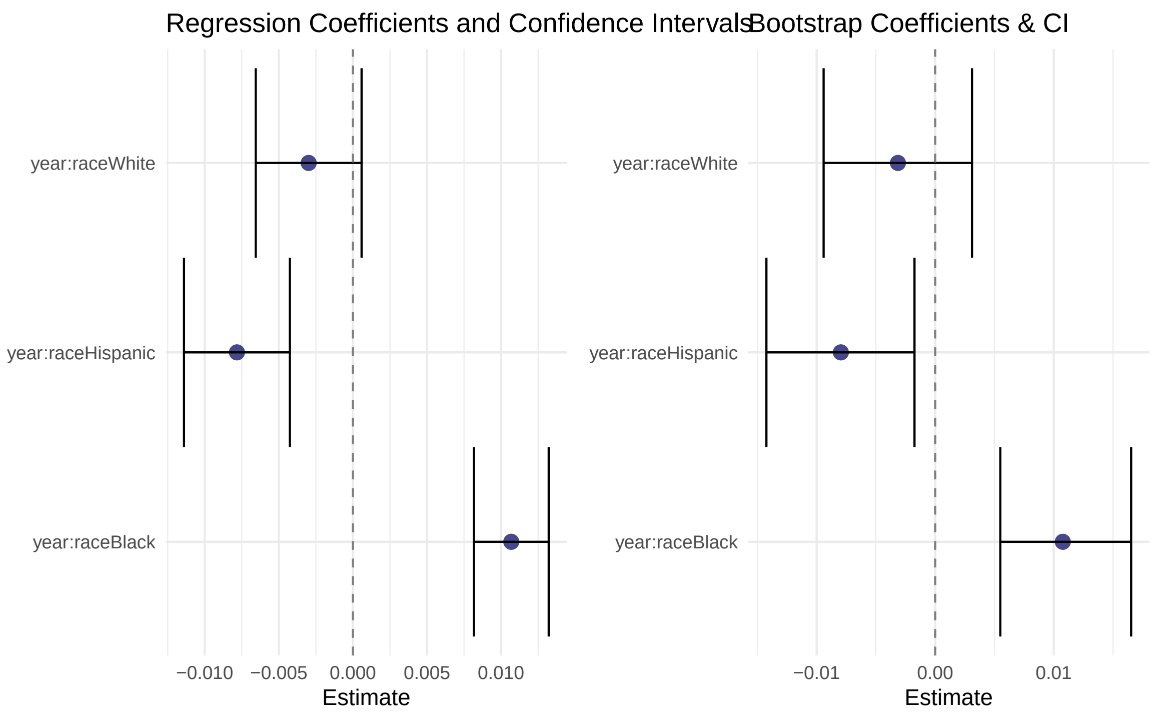
Now that we have computed our Bootstrap estimates and Confidence intervals, we'd like to visualize it the results. We'll use the technique as before when we visualized the original coefficient estimates and their confidence intervals.

```
p2 <- boot_results %>%
  filter(str_detect(term, "year")) %>%
  mutate(term =
           case_when(term == "year" ~ "year:raceBlack",
                          TRUE ~ term)) %>%
  ggplot(aes(x = bootstrap_estimate,
             y = term)) +
```

```r
  geom_point(size = 3, alpha = 0.8) +
  geom_errorbar(aes(
    xmin = lower_ci,
    xmax = upper_ci,
    width = 1)) +
  geom_vline(xintercept = 0, linetype = "dashed", color = "gray50") +
  labs(title = "Bootstrap Coefficients & CI",
       y = NULL,
       x = "Estimate")
```

Let's look at the two visualizations side-by-side and compare.

```r
print(p1 + p2)
```



It looks like the Bootstrap CIs are much wider than the OLS CIs. Let's try to understand why.

Topic	Description
OLS Confidence Intervals	OLS confidence intervals are typically constructed as: `Estimate ± (Critical Value * Standard Error)` Where: `Estimate`: coefficient estimate from OLS regression. `Critical Value`: from t-distribution (or Z-distribution for large samples). `Standard Error`: estimated standard deviation of the coefficient. The standard error in OLS is based on assumptions of homoscedasticity and normality. If violated, it can be underestimated.
Bootstrap Confidence Intervals	Bootstrap confidence intervals are derived from the empirical distribution of coefficient estimates obtained from repeated resampling. **Percentile Bootstrap:** The common method is to use the percentiles of the bootstrap distribution to form the confidence interval. For a 95% CI, use the 2.5th and 97.5th percentiles of bootstrap estimates. The bootstrap standard error is the standard deviation of the bootstrap estimates.
Why Bootstrap CIs Are Wider	**Violation of Assumptions:** OLS standard errors rely on theoretical formulas that assume homoscedasticity and normality. If violated, OLS standard errors may be smaller than true standard errors. Bootstrapping estimates standard error directly from the data, without these assumptions. **Empirical Distribution:** The bootstrap percentile method captures the shape of the empirical distribution of coefficient estimates. If this distribution is non-normal or skewed, it produces wider intervals than OLS. **Variance Estimation:** The variance of bootstrap estimates is calculated as the sample variance of bootstrap coefficients. This empirical estimate can be larger than the theoretical variance estimated by OLS, especially if OLS assumptions are violated. **Interaction Terms:** The variance of interaction terms is usually larger than that of single terms. Bootstrapping captures this increase in variance, while OLS can underestimate it.

In summary:

- OLS confidence intervals are based on theoretical approximations that can underestimate uncertainty when assumptions are violated.

- Bootstrap confidence intervals are based on empirical distributions and direct variance estimation, providing a more accurate and often wider representation of uncertainty.

6 Conclusion:

This analysis explored the trends in student loan debt percentage across racial groups over time, utilizing both Ordinary Least Squares (OLS) regression and bootstrap resampling techniques. The OLS model provided initial insights into the relationship between year, race, and loan debt percentage, revealing significant differences in debt accumulation patterns. However, the bootstrap analysis, which relaxes the restrictive assumptions of OLS, demonstrated a more robust and realistic assessment of uncertainty.

The wider confidence intervals obtained through bootstrapping highlight the limitations of relying solely on OLS, particularly when dealing with complex datasets and potential violations of model assumptions. These wider intervals underscore the importance of considering empirical distributions and directly estimating variability, especially in studies involving interaction terms and sensitive demographic data.

The findings indicate that Black families experience a consistent and statistically significant annual increase in student loan debt percentage. Additionally, while Hispanic families also show a significant increase, the rate of increase is statistically lower compared to Black families. Conversely, the difference in the rate of increase between White and Black families was not found to be statistically significant.

Ultimately, this analysis reinforces the importance of using robust statistical methods to accurately capture uncertainty and provide reliable insights into complex social phenomena. The bootstrap approach offers a valuable tool for researchers seeking to move beyond the limitations of traditional parametric methods, providing a more nuanced and data-driven understanding of student debt inequality.