Module 3

Goals of module 3:
* Regression analysis overview * Understanding purpose and how to use it to address your questions * Relationship between variable types and regression interpretation * Stock and flow variables * Logging data * Lagging and differencing data * Lag() and data.table::shift() * Special considerations for: * Policy evaluation * Difference-in-differences * Time series data * Panels * Survey data * Recap on using dummies * Controlling analysis output * Stargazer options * Broom package * Making maps * ggmap * Regression * Variable transformation * Log of price * price per square foot (?) * Interaction effects (?) - may not find any significant ones

Day One

For the past month we've looked at housing data from Redfin to see how housing prices in the Washington, D.C. metro area differ based on distances to public transit. This module will shift attention to employment and take us through a variety of numerical analyses starting with policy evaluation. We will examine how minimum wage laws affect labor conditions in local areas using data collected by David Card and Alan Krueger on fast-food job wages.

The 1994 Card and Krueger Study * A 1992, $0.80 minimum wage increase by New Jersey prompted researchers to revisit the debate on the aggregate effect of the minimum wage on labor markets. * Card and Krueger collected information from fast-food restaurants in New Jersey and eastern Pennsylvania. * They found no indication of a reduction in employment. * An even bigger impact of the study was the introduction of differences-in-differences. * Intuitive tool suitable for quasi-experimental studies

First, a refresher on when to use regressions.

When to use regression analysis in economics * Trying to identify causation * Correlation vs. causation * **Inductive** reasoning * Inference based on observed data

Regression analysis defined * Used to describe the relationship between: * A single response variable Y and * Any number of predictor variables X_1, X_2, . . . , X_n * The goal is to show Y is determined by X_1, . . . , X_n * The response variable for OLS must be continuous (but sometimes economists cheat) * There are no restrictions on the predictor variables * Predictors can be continuous, discrete, or categorical

Steps to take before you put your data into a regression * Check for: * Missing values * Outliers * Asymmetric distributions * Clustering of values * Unexpected patterns * Numerical summaries * Mean, min, max, variance, etc. * Correlations * Graphical summaries * Scatter plots * Histograms * Box plots

We are going to start by reading in the Card and Kreuger data and doing some basic analysis in the same way that we have been since the beginning of class. We want to get an idea of what the data looks like before we perform any sort of regressions.

Find the data file with the name 'public', and you will see this is saved as a .dat file: a generic data file format that can be read into R using the "fread" function from the data.table package. Once you've read in the data, take a look at the variable names.

```
# read in the employment data
cardKrueger <- fread("Data/public.dat")

# look at how reading in data went
glimpse(cardKrueger)
```

```
## Observations: 410
## Variables: 46
```

```
## $ V1  <int> 46, 49, 506, 56, 61, 62, 445, 451, 455, 458, 462, 468, 469...
## $ V2  <int> 1, 2, 2, 4, 4, 4, 1, 1, 2, 2, 3, 1, 1, 1, 1, 2, 2, 3, 3, 3...
## $ V3  <int> 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0...
## $ V4  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V5  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V6  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V7  <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V8  <int> 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V9  <int> 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ V10 <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ V11 <int> 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 2, 2, 0, 0, 1, 2...
## $ V12 <chr> "30.00", "6.50", "3.00", "20.00", "6.00", "0.00", "50.00",...
## $ V13 <chr> "15.00", "6.50", "7.00", "20.00", "26.00", "31.00", "35.00...
## $ V14 <chr> "3.00", "4.00", "2.00", "4.00", "5.00", "5.00", "3.00", "5...
## $ V15 <chr> ".", ".", ".", "5.00", "5.50", "5.00", "5.00", "5.00", "5....
## $ V16 <chr> "19.0", "26.0", "13.0", "26.0", "52.0", "26.0", "26.0", "5...
## $ V17 <chr> ".", ".", "0.37", "0.10", "0.15", "0.07", "0.10", "0.25", ...
## $ V18 <int> 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1...
## $ V19 <chr> ".", ".", "30.0", "0.0", "0.0", "45.0", "0.0", "0.0", "0.0...
## $ V20 <int> 2, 2, 2, 2, 3, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2...
## $ V21 <dbl> 6.5, 10.0, 11.0, 10.0, 10.0, 10.0, 6.0, 0.0, 11.0, 11.0, 9...
## $ V22 <dbl> 16.5, 13.0, 10.0, 12.0, 12.0, 12.0, 18.0, 24.0, 10.0, 10.0...
## $ V23 <chr> "1.03", "1.01", "0.95", "0.87", "0.87", "0.87", "1.04", "1...
## $ V24 <chr> "1.03", "0.90", "0.74", "0.82", "0.77", "0.77", "0.88", "0...
## $ V25 <chr> "0.52", "2.35", "2.33", "1.79", "1.65", "0.95", "0.94", "0...
## $ V26 <chr> "3", "4", "3", "2", "2", "2", "3", "6", "2", "4", "4", "4"...
## $ V27 <chr> "3", "3", "3", "2", "2", "2", "3", "4", "2", "4", "4", "3"...
## $ V28 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1...
## $ V29 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ V30 <int> 111792, 111292, 111292, 111492, 111492, 111492, 110792, 11...
## $ V31 <chr> "1", ".", ".", ".", ".", ".", ".", "2", ".", "1", "1", "."...
## $ V32 <chr> "3.50", "0.00", "3.00", "0.00", "28.00", ".", "15.00", "26...
## $ V33 <chr> "35.00", "15.00", "7.00", "36.00", "3.00", ".", "18.00", "...
## $ V34 <chr> "3.00", "4.00", "4.00", "2.00", "6.00", ".", "5.00", "6.00...
## $ V35 <chr> "4.30", "4.45", "5.00", "5.25", "4.75", ".", "4.75", "5.00...
## $ V36 <chr> "26.0", "13.0", "19.0", "26.0", "13.0", "26.0", "26.0", "2...
## $ V37 <chr> "0.08", "0.05", "0.25", "0.15", "0.15", ".", "0.15", "0.20...
## $ V38 <chr> "1", "0", ".", "0", "0", "0", "0", "0", "0", "0", "0", "1"...
## $ V39 <chr> "2", "2", "1", "2", "2", "2", "2", "2", "2", "1", "2", "2"...
## $ V40 <chr> "6.50", "10.00", "11.00", "10.00", "10.00", "10.00", "6.00...
## $ V41 <chr> "16.50", "13.00", "11.00", "12.00", "12.00", "12.00", "18....
## $ V42 <chr> "1.03", "1.01", "0.95", "0.92", "1.01", ".", "1.04", "1.11...
## $ V43 <chr> ".", "0.89", "0.74", "0.79", "0.84", "0.84", "0.86", "0.84...
## $ V44 <chr> "0.94", "2.35", "2.33", "0.87", "0.95", "1.79", "0.94", "0...
## $ V45 <chr> "4", "4", "4", "2", "2", "3", "3", "6", "4", "4", "6", "4"...
## $ V46 <chr> "4", "4", "3", "2", "2", "3", "3", "3", "3", "3", "3", "3"...
```

The variable names do not appear very helpful, and many of our variables were read in as factors. To understand how our dataset is set up, we will need to look at the codebook. You can open up the codebook in R from the file panel. We will need to rename and recode our variables to match the codebook so that we can more easily keep track of variables in our dataset. * Can you identify the variables that would be best treated as categorical? * How about numerical?

Why did many of our variables get read in as factors?

```r
# try reading in the employment data again, but avoid variables that should not be factors getting turn
cardKrueger <- fread("Data/public.dat", na.strings = c("NA", "."))

# change variable names according to code book
names(cardKrueger) <- c("SHEET", "CHAIN", "CO_OWNED", "STATE", "SOUTHJ",
                        "CENTRALJ", "NORTHJ", "PA1", "PA2", "SHORE", "NCALLS",
                        "EMPFT", "EMPPT", "NMGRS", "WAGE_ST", "INCTIME",
                        "FIRSTINC", "BONUS", "PCTAFF", "MEALS", "OPEN",
                        "HRSOPEN", "PSODA", "PFRY", "PENTREE", "NREGS",
                        "NREGS11", "TYPE2", "STATUS2", "DATE2","NCALLS2",
                        "EMPFT2", "EMPPT2", "NMGRS2", "WAGE_ST2", "INCTIME2",
                        "FIRSTIN2", "SPECIAL2", "MEALS2", "OPEN2R", "HRSOPEN2",
                        "PSODA2","PFRY2", "PENTREE2", "NREGS2", "NREGS112")
```

Now that the variable names match our variables in the codebook, we can transform some of the variable values according to the codebook. Start by looking at the class of our variables using one of the functions we learned during the first module.

```r
# Check variable classes
glimpse(cardKrueger)
```

```
## Observations: 410
## Variables: 46
## $ SHEET    <int> 46, 49, 506, 56, 61, 62, 445, 451, 455, 458, 462, 468...
## $ CHAIN    <int> 1, 2, 2, 4, 4, 4, 1, 1, 2, 2, 3, 1, 1, 1, 1, 2, 2, 3,...
## $ CO_OWNED <int> 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1,...
## $ STATE    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ SOUTHJ   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ CENTRALJ <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ NORTHJ   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ PA1      <int> 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ PA2      <int> 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ SHORE    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ NCALLS   <int> 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 2, 2, 0, 0,...
## $ EMPFT    <dbl> 30.0, 6.5, 3.0, 20.0, 6.0, 0.0, 50.0, 10.0, 2.0, 2.0,...
## $ EMPPT    <dbl> 15.0, 6.5, 7.0, 20.0, 26.0, 31.0, 35.0, 17.0, 8.0, 10...
## $ NMGRS    <dbl> 3, 4, 2, 4, 5, 5, 3, 5, 5, 2, 3, 3, 5, 3, 3, 3, 1, 2,...
## $ WAGE_ST  <dbl> NA, NA, NA, 5.00, 5.50, 5.00, 5.00, 5.00, 5.25, 5.00,...
## $ INCTIME  <dbl> 19, 26, 13, 26, 52, 26, 26, 52, 13, 19, 13, 13, 39, N...
## $ FIRSTINC <dbl> NA, NA, 0.37, 0.10, 0.15, 0.07, 0.10, 0.25, 0.25, 0.1...
## $ BONUS    <int> 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1,...
## $ PCTAFF   <dbl> NA, NA, 30, 0, 0, 45, 0, 0, 0, 0, 5, 0, 80, 0, 0, 0, ...
## $ MEALS    <int> 2, 2, 2, 2, 3, 2, 2, 2, 1, 1, 2, 2, 2, 2, 2, 3, 2, 2,...
## $ OPEN     <dbl> 6.5, 10.0, 11.0, 10.0, 10.0, 10.0, 6.0, 0.0, 11.0, 11...
## $ HRSOPEN  <dbl> 16.5, 13.0, 10.0, 12.0, 12.0, 12.0, 18.0, 24.0, 10.0,...
## $ PSODA    <dbl> 1.03, 1.01, 0.95, 0.87, 0.87, 0.87, 1.04, 1.05, 0.73,...
## $ PFRY     <dbl> 1.03, 0.90, 0.74, 0.82, 0.77, 0.77, 0.88, 0.84, 0.73,...
## $ PENTREE  <dbl> 0.52, 2.35, 2.33, 1.79, 1.65, 0.95, 0.94, 0.96, 2.32,...
## $ NREGS    <int> 3, 4, 3, 2, 2, 2, 3, 6, 2, 4, 4, 4, 3, 3, 3, 5, 4, 6,...
## $ NREGS11  <int> 3, 3, 3, 2, 2, 2, 3, 4, 2, 4, 4, 3, 2, 3, 1, 4, 3, 3,...
## $ TYPE2    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1,...
## $ STATUS2  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ DATE2    <int> 111792, 111292, 111292, 111492, 111492, 111492, 11079...
## $ NCALLS2  <int> 1, NA, NA, NA, NA, NA, NA, 2, NA, 1, 1, NA, 1, 2, 1, ...
## $ EMPFT2   <dbl> 3.5, 0.0, 3.0, 0.0, 28.0, NA, 15.0, 26.0, 3.0, 2.0, 1...
```

```
## $ EMPPT2   <dbl> 35, 15, 7, 36, 3, NA, 18, 9, 12, 9, 25, 32, 39, 10, 2...
## $ NMGRS2   <dbl> 3, 4, 4, 2, 6, NA, 5, 6, 2, 2, 4, 4, 4, 3, 3, 3, 3, 3...
## $ WAGE_ST2 <dbl> 4.30, 4.45, 5.00, 5.25, 4.75, NA, 4.75, 5.00, 5.00, 5...
## $ INCTIME2 <dbl> 26, 13, 19, 26, 13, 26, 26, 26, 13, 13, 13, 26, 41, 1...
## $ FIRSTIN2 <dbl> 0.08, 0.05, 0.25, 0.15, 0.15, NA, 0.15, 0.20, 0.25, 0...
## $ SPECIAL2 <int> 1, 0, NA, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0...
## $ MEALS2   <int> 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2,...
## $ OPEN2R   <dbl> 6.5, 10.0, 11.0, 10.0, 10.0, 10.0, 6.0, 0.0, 11.0, 11...
## $ HRSOPEN2 <dbl> 16.5, 13.0, 11.0, 12.0, 12.0, 12.0, 18.0, 24.0, 11.0,...
## $ PSODA2   <dbl> 1.03, 1.01, 0.95, 0.92, 1.01, NA, 1.04, 1.11, 0.94, 0...
## $ PFRY2    <dbl> NA, 0.89, 0.74, 0.79, 0.84, 0.84, 0.86, 0.84, 0.84, 0...
## $ PENTREE2 <dbl> 0.94, 2.35, 2.33, 0.87, 0.95, 1.79, 0.94, 0.94, 2.32,...
## $ NREGS2   <int> 4, 4, 4, 2, 2, 3, 3, 6, 4, 4, 6, 4, 3, 3, 3, 3, 4, 6,...
## $ NREGS112 <int> 4, 4, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 3, 3, 1,...
```

```
# create a clean dataset using information from the code book
cardKrueger_clean <- cardKrueger %>%
  mutate(chainName = case_when(CHAIN == 1 ~ "Burger King",
                               CHAIN == 2 ~ "KFC",
                               CHAIN == 3 ~ "Roys",
                               CHAIN == 4 ~ "Wendys"),
         stateAbbrv = ifelse(STATE == 1, "NJ", "PA"),
         location = case_when(SOUTHJ == 1 ~ "Southern NJ",
                              CENTRALJ == 1 ~ "Central NJ",
                              NORTHJ == 1 ~ "Northern NJ",
                              PA1 == 1 ~ "Northeast suburbs of Philadelphia",
                              PA2 == 1 ~ "Easton",
                              SHORE == 1 ~ "NJ Shore")) %>%
  # subset to useful variables
  select(stateAbbrv, chainName, location, EMPFT, EMPPT, NMGRS, WAGE_ST, HRSOPEN,
         PSODA, PFRY, PENTREE, EMPFT2, EMPPT2, NMGRS2, WAGE_ST2, HRSOPEN2,
         PSODA2, PFRY2, PENTREE2)

# Tidy up workspace
rm(cardKrueger)
```

Understanding the movement in wages

What is the average wage by state, by fast food chain, before and after the policy goes into effect?

```
# use dplyr to check average wage by state and chain
avgWage <- cardKrueger_clean %>%
  group_by(chainName, stateAbbrv) %>%
  summarise(avgWageBefore = mean(WAGE_ST, na.rm = TRUE),
            avgWageAfter = mean(WAGE_ST2, na.rm = TRUE)) %>%
  # Sort by chain to get like-restaurants next to each other
  arrange(chainName) %>%
  # Remove grouping attribute
  ungroup()
avgWage
```

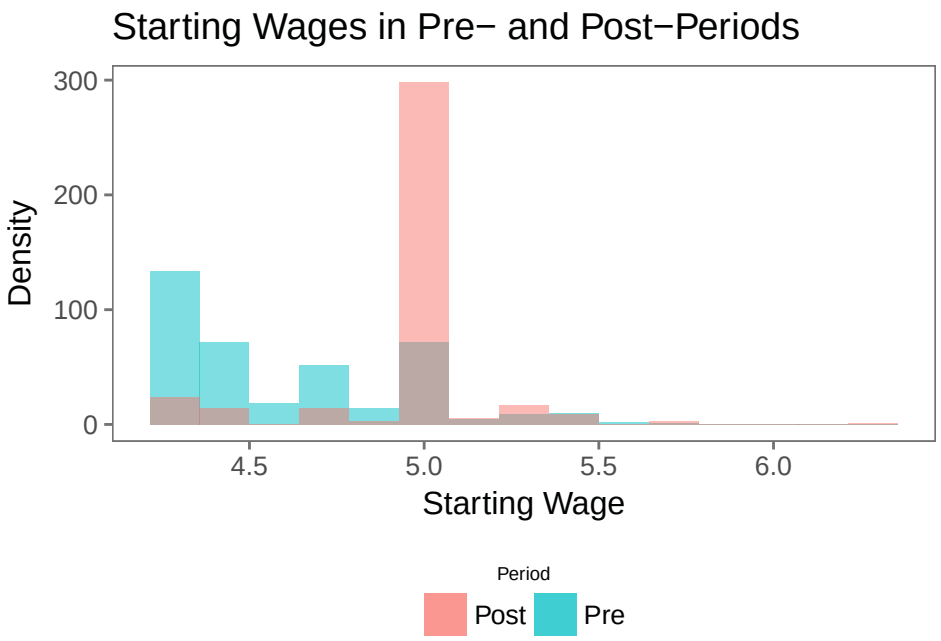
```
## # A tibble: 8 x 4
##     chainName stateAbbrv avgWageBefore avgWageAfter
##         <chr>      <chr>         <dbl>        <dbl>
```

```
## 1 Burger King         NJ      4.533228     5.076364
## 2 Burger King         PA      4.521176     4.579697
## 3         KFC         NJ      4.575224     5.075758
## 4         KFC         PA      4.725000     4.683333
## 5        Roys         NJ      4.669241     5.063462
## 6        Roys         PA      4.794118     4.660667
## 7      Wendys         NJ      4.806829     5.135238
## 8      Wendys         PA      4.628000     4.600000
```

There seems to be significant movement in starting wage between these two periods in New Jersey, while Pennsylvania stays relatively constant for each chain. Is this enough to know whether the minimum wage increase in New Jersey had an impact on the labor market? * Do we see differences between chains in the wages paid to employees? * Consider how this may or may not impact our analysis * Do you think stores with higher salaries have cheaper or more expensive food? * What does this table tell us about employment?

Make a histogram plot of wages in the two periods and use color to indicate the time period. * Pay attention to the shape of the data and how that impacts ggplot layering

```
# try out ggthemes to change the look of the plot
ggplot(cardKrueger_clean) +
  # Need to use geom_ in the following way due to the structure of our data
  geom_histogram(bins = 15, aes(x = WAGE_ST, fill = "Pre"), alpha = 0.5) +
  geom_histogram(bins = 15, aes(x = WAGE_ST2, fill = "Post"), alpha = 0.5) +
  labs(x = "Starting Wage",
     y = "Density",
     fill = "Period",
     title = "Starting Wages in Pre- and Post-Periods") +
  theme_few() +
  theme(legend.position = "bottom", legend.title = element_text(size = 7)) +
  guides(fill = guide_legend(title.position = "top", title.hjust = 0.5))
```



Evident from the plot is the effect of the policy on wages. What are some important nuances in the graph important for our research topic?

Understanding the simultaneous shifts in employment levels

What is the total number of employees by state, by fast food chain, before and after the policy goes into effect?

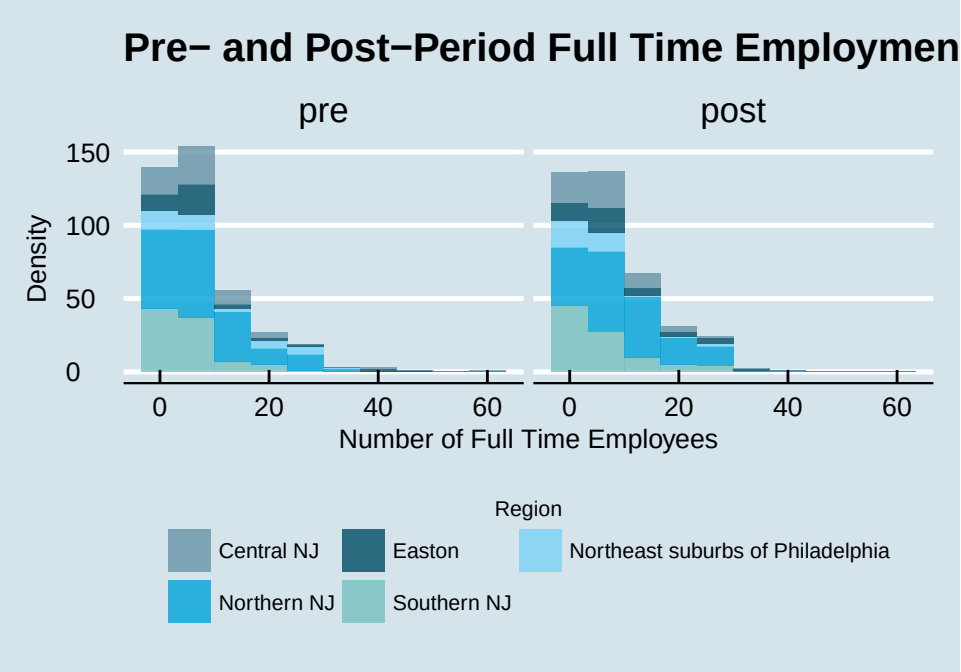
```
# use dplyr to check number of employees by state and chain
numEmp <- cardKrueger_clean %>%
  group_by(chainName, stateAbbrv) %>%
  # Card and Krueger weigh managers half as much as other employees
  summarise(empBefore = sum(EMPFT, EMPPT, 0.5*NMGRS, na.rm = TRUE),
            empAfter = sum(EMPFT2, EMPPT2, 0.5*NMGRS2, na.rm = TRUE)) %>%
  # Sort by chain to get like-restaurants next to each other
  arrange(chainName) %>%
  # Remove grouping attribute
  ungroup()
numEmp
```

```
## # A tibble: 8 x 4
##     chainName stateAbbrv empBefore empAfter
##         <chr>      <chr>     <dbl>    <dbl>
## 1 Burger King         NJ    4073.4  4229.75
## 2 Burger King         PA    1336.5  1259.50
## 3         KFC         NJ    1145.5  1169.50
## 4         KFC         PA     175.0   211.00
## 5        Roys         NJ    2649.0  2427.50
## 6        Roys         PA     463.0   383.50
## 7      Wendys         NJ    1328.5  1380.50
## 8      Wendys         PA     479.5   407.00
```

Make a histogram of employment by region before the minimum wage changes and after.

```
# reshape the data to make an indicator variable for the time period
empl_PrePost <- cardKrueger_clean %>%
  gather(-chainName, -stateAbbrv, -location,
         key = "variable", value = "value") %>%
  filter(variable %in% c("EMPFT", "EMPFT2")) %>%
  mutate(period = ifelse(variable == "EMPFT2", "post", "pre")) %>%
  mutate(period = factor(period, levels = c("pre", "post")))

# plot the data
ggplot(empl_PrePost) +
  geom_histogram(bins = 10, aes(x = value, fill = location), alpha = 0.8) +
  labs(x = "Number of Full Time Employees",
       y = "Density",
       fill = "Region",
       title = "Pre- and Post-Period Full Time Employment") +
  scale_fill_economist() +
  theme_economist() +
  # We can use multiple calls to theme
  theme(legend.position = "bottom", legend.text = element_text(size = 8),
        legend.title = element_text(size = 8)) +
  facet_grid(. ~ period) +
  guides(fill = guide_legend(nrow = 2, byrow = TRUE, title.position = "top", title.hjust = 0.5))
```



We have seen through our data exploration exercise a number of relationships between variables of interest. Different states had different patterns in wage changes, chains seemed to have different levels in the wages offered to new employees, and regions have varying levels of employees per store. What we have not seen, however, is any proof of causal effect.

Our research goal lends itself to causal inference. Surely, we do care about movements in wages and employment in general; our goal is to exactly find the effect of one on the other, and in particular among low-wage jobs (which is the only type of job we observe in this dataset). Now we turn to regressions.

OLS and Fixed Effects

```
# straightforward linear regression on pre-period employment and wages
preReg <- lm(EMPFT ~ WAGE_ST, cardKrueger_clean)
```

preReg contains the output of a regression that proposed the following hypothesis: full-time employment is linearly determined by starting wages (plus unobserved noise). Let's see how well this hypothesis performed on the pre-policy-change period. You have already seen the stargazer package at this point, and we will continue to use it here.

```
# Call stargazer to create output table; type = "text" will let us review it right in the markdown file
stargazer(preReg, type = "text")
```

```
##
## ================================================
##                         Dependent variable:
##                     ---------------------------
##                                 EMPFT
## ------------------------------------------------
## WAGE_ST                        3.193**
##                                (1.255)
##
## Constant                       -6.468
```

```
##                                (5.807)
##
## ------------------------------------------------
## Observations                     385
## R2                              0.017
## Adjusted R2                     0.014
## Residual Std. Error       8.500 (df = 383)
## F Statistic            6.472** (df = 1; 383)
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```

Wage is coming out weakly significant as a determinant of employment; however, the model as a whole does not do the greatest job in explaining variations in the employment variable. * What other variables did we find to be impactful on employment? * Are any of them categorical variables?

Fixed effects

You already have some exposure to the use of fixed effects from module 2, so the goal here will be to build an intuition for recognizing when you can and cannot employ fixed effects and how to construct variables that will make it easy to do so.

Fixed effects let you limit your analysis to within-group variation. In a panel data setting this is most commonly used to account for heterogeneity **between** individual subjects, resulting in a model that can get you consistent estimates even with structural differences between groups/subjects. * As an analogy, think about how group_by() in the dplyr package works. * Similar to running lm() after using group_by()

For our use cases, there is no meaningful differences between a fixed effects estimator and a least squares dummy variables estimator; they are computationally equivalent. This means we will think of indvidual/group fixed effects and the use of dummies as similar or identical.

Some reasons to use fixed effects: * The variation you see when pooling all observations together might not be of interest to you. * You believe a variable, or any transformation of one, should not enter your model linearly but still does explain variation. * There is a non-numeric variable that you want to include in your model.

When you cannot employ fixed effects with a variable: * There is not enough variation in each group of the variable

As an example, could we use a fixed effect with a different group for each observation of our dataset? How many observations would each group have (is this enough variation to run a regression)?

Specifying fixed effects in lm():

```
# If class of x is character, factor, logical, or otherwise only has two levels
lm(y ~ x)

# If class of x is numeric, convert to factor
lm(y ~ as.factor(x))
```

```
# adding state fixed effects on pre-period variables
preReg_fixedState <- lm(EMPFT ~ WAGE_ST + stateAbbrv,
                        cardKrueger_clean)
stargazer(preReg_fixedState, type = "text")
```

```
##
## ================================================
##                         Dependent variable:
##                     ---------------------------
```

```
##                                EMPFT
## ---------------------------------------------
## WAGE_ST                       3.129**
##                               (1.250)
##
## stateAbbrvPA                  2.196**
##                               (1.090)
##
## Constant                      -6.601
##                               (5.784)
##
## ---------------------------------------------
## Observations                    385
## R2                             0.027
## Adjusted R2                    0.022
## Residual Std. Error      8.466 (df = 382)
## F Statistic           5.293*** (df = 2; 382)
## =============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

Stargazer Options Review

We have seen and used stargazer for over a month, but to recap: the purpose of stargazer is to create publication-ready tables. Both regression tables and summary tables can be made in the same, user-friendly format. Virtually any number of model objects (the type of output from lm()) for regression tables or any number of data frames/vectors/matrices for summary tables can be input into stargazer. Let us see what happens when we add a second lm object to the stargazer call from the first regression we ran; i.e. let's add both preReg and preReg_fixedState to the same stargazer call.

```
stargazer(preReg, preReg_fixedState, type = "text")
```

```
##
## ===============================================================
##                                 Dependent variable:
##                     ------------------------------------------
##                                        EMPFT
##                               (1)                   (2)
## ---------------------------------------------------------------
## WAGE_ST                     3.193**               3.129**
##                             (1.255)               (1.250)
##
## stateAbbrvPA                                      2.196**
##                                                   (1.090)
##
## Constant                    -6.468                -6.601
##                             (5.807)               (5.784)
##
## ---------------------------------------------------------------
## Observations                  385                   385
## R2                           0.017                 0.027
## Adjusted R2                  0.014                 0.022
## Residual Std. Error   8.500 (df = 383)      8.466 (df = 382)
## F Statistic         6.472** (df = 1; 383) 5.293*** (df = 2; 382)
## ===============================================================
```

```
## Note:                                *p<0.1; **p<0.05; ***p<0.01
```

Two columns of coefficients appear in the regression table, the first belonging to the model used in preReg and the second from the model in preReg_fixedState. Using the summary tables feature of stargazer is very similar. Now let us try one of the data frames we created earlier when we were exploring data, numEmp. Recall that this data frame already contains summary statistics that we calculated; therefore, we want to specify to stargazer that we do not want to compute summary statistics as we make the table (since we already have the statistics we want). The way to handle this option is with the summary argument: setting summary = FALSE will command stargazer to output the content of the data frame.

```
stargazer(numEmp, type = "text", summary = FALSE)
```

```
##
## ==========================================
##    chainName  stateAbbrv empBefore empAfter
## ------------------------------------------
## 1 Burger King     NJ      4073.4   4229.75
## 2 Burger King     PA      1336.5    1259.5
## 3     KFC         NJ      1145.5    1169.5
## 4     KFC         PA        175      211
## 5    Roys         NJ       2649     2427.5
## 6    Roys         PA        463     383.5
## 7   Wendys        NJ      1328.5    1380.5
## 8   Wendys        PA       479.5     407
## ------------------------------------------
```

You have already explored and used many of these options, but here is a list of some commonly used arguments for stargazer:

Main arguments used for regression tables: * covariate.labels: Labels for the variables listed on the far left, top to bottom * dep.var.labels: Labels for the variables listed at the top of each column, left to right * title: Title of table * keep/omit: Lets you suppress printing out variables you do not want, based on text patterns * order: Lets you control the order of dependent variables (from top to bottom), based on text patterns * intercept.bottom: If set to FALSE, the intercept term is put at the top; the default is TRUE and the intercept is listed last * column.labels: Labels for columns. These show up right below dep.var.labels and let you provide a very short description of the model in that column. * . . . : And many more!

Main arguments used for summary statistics tables: * summary: If TRUE, calculate number of observations, mean, median, max, and min of numeric variables in the input object. If FALSE, output the contents of the input object. * rownames: If TRUE, output the rownames (by default these are the row number). If FALSE, suppress rownames from the output. * digits: Lets you control how many decimal places to print. * . . . : Some more!

To find a description of EVERY argument to stargazer try the manual. * ?stargazer

```
# adding fixed effects and controls on pre-period variables
preReg_controls <- lm(EMPFT ~ WAGE_ST + stateAbbrv + HRSOPEN +
                        PSODA + PFRY + PENTREE, cardKrueger_clean)
stargazer(preReg, preReg_fixedState, preReg_controls, type = "text", intercept.bottom = FALSE)
```

```
##
## ================================================================================================
##                                          Dependent variable:
##                     ---------------------------------------------------------------------------
##                                                  EMPFT
##                               (1)                 (2)                    (3)
## ------------------------------------------------------------------------------------------------
## Constant                    -6.468              -6.601                -13.311*
```

```
##                             (5.807)              (5.784)              (7.801)
##
## WAGE_ST                    3.193**              3.129**              3.065**
##                            (1.255)              (1.250)              (1.186)
##
## stateAbbrvPA                                    2.196**               2.107*
##                                                 (1.090)              (1.155)
##
## HRSOPEN                                                             0.812***
##                                                                     (0.212)
##
## PSODA                                                                -7.579
##                                                                     (6.597)
##
## PFRY                                                                 4.293
##                                                                     (5.265)
##
## PENTREE                                                              -0.593
##                                                                     (0.824)
##
## ------------------------------------------------------------------------------------------
## Observations                 385                  385                  364
## R2                          0.017                0.027                0.117
## Adjusted R2                 0.014                0.022                0.102
## Residual Std. Error   8.500 (df = 383)     8.466 (df = 382)     7.751 (df = 357)
## F Statistic         6.472** (df = 1; 383) 5.293*** (df = 2; 382) 7.861*** (df = 6; 357)
## ==========================================================================================
## Note:                                                  *p<0.1; **p<0.05; ***p<0.01
```

```
# simple linear regression on post-period variables
PostReg <- lm(numFullTimeEmp2 ~ startingWage2, cardKrueger_clean)
```

```
## Error in eval(predvars, data, env): object 'numFullTimeEmp2' not found
```

```
stargazer(PostReg, type = "text")
```

```
## Error in .stargazer.wrap(..., type = type, title = title, style = style, : object 'PostReg' not found
```

```
# adding fixed effects on post-period variables
PostReg_fixedEffects <- lm(numFullTimeEmp2 ~ startingWage2 + state,
                           cardKrueger_clean)
```

```
## Error in eval(predvars, data, env): object 'numFullTimeEmp2' not found
```

```
stargazer(PostReg_fixedEffects, type = "text")
```

```
## Error in .stargazer.wrap(..., type = type, title = title, style = style, : object 'PostReg_fixedEffe
```

```
# adding fixed effects and controls on post-period variables
PostReg_controls <- lm(numFullTimeEmp2 ~ startingWage2 + state + hoursOpen2 +
                         priceSoda2 + priceFry2 + priceEntree2, cardKrueger_clean)
```

```
## Error in eval(predvars, data, env): object 'numFullTimeEmp2' not found
```

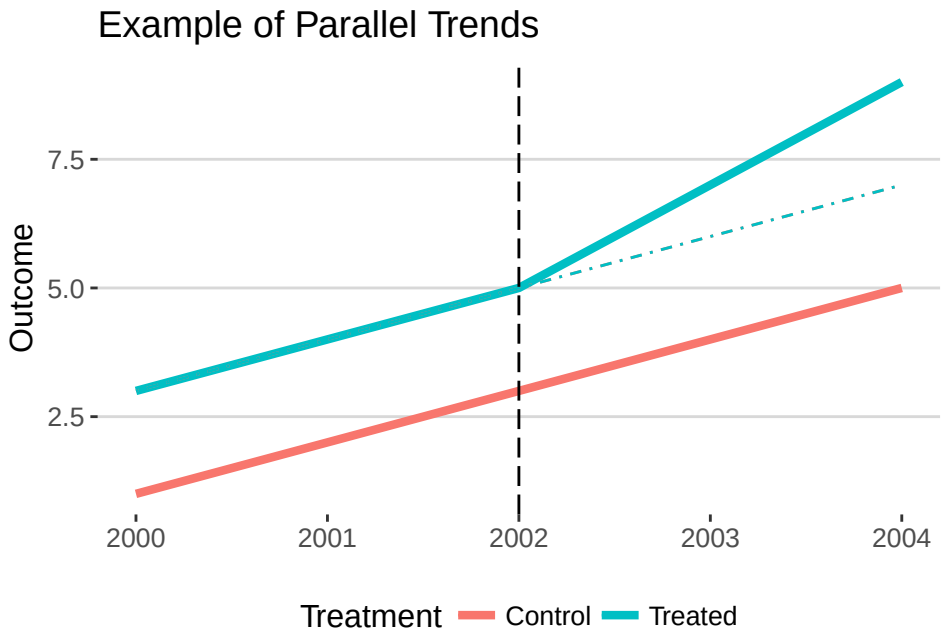
```
stargazer(PostReg_controls, type = "text")
```

```
## Error in .stargazer.wrap(..., type = type, title = title, style = style, : object 'PostReg_controls'
```

HOW MUCH OF THE WAGE CAN YOU EXPLAIN USING LOCATION? WHAT ABOUT CHAIN?

Difference-in-differences (DID)

With the DID technique you are able to make use of a treatment and a control groups to make causal inferences about the effect of some treatment. You cannot call on this tool if you do not observe both groups; the treatment group so you can observe the result of treatment and the control group so you have a baseline comparison to estimate the full effect of the treatment. * The control should serve as a counterfactual - it would tell you how the treatment group would look like if no treatment ever occurred. * Admissible data structures include aggregated data, pooled cross-sections, or longitudinal A useful tool to use in quasi-experiments, DID addresses concerns about random assignment (since it is not necessary) while also accounting for confounders. * Confounders are variables that affect both sides of your equation. * The major assumption: parallel trends * Differences in original characteristics between control and treatment groups okay - differences in *trends* between groups *NOT* okay * Violation of parallel trends leads to biased estimates



The additional difference in the treatment group in the post-treatment period (after 2002) from the baseline counterfactual (the dotted line) is the DID estimate.

```
# recreate the differences-in-differences done in the Card and Krueger paper

# create a total employment variable: part time employees count as half, and
## include number of managers as full time employees. Then find the change
## in total employment between periods
regression_data <-
  cardKrueger_clean %>%
  mutate(totalEmp = numFullTimeEmp + numManagers + 0.5*numPartTimeEmp,
         totalEmp2 = numFullTimeEmp2 + numManagers2 + 0.5*numPartTimeEmp2) %>%
  mutate(changeTotalEmp = totalEmp2 - totalEmp)
```

```
## Error in mutate_impl(.data, dots): Evaluation error: object 'numFullTimeEmp' not found.
```

```
# find the average employment in both periods and the average change
empAvgs <- group_by(regression_data, state) %>%
  summarise(preAvgEmp = mean(totalEmp, na.rm = TRUE),
            postAvgEmp = mean(totalEmp2, na.rm = TRUE),
            avgEmpChange = mean(changeTotalEmp, na.rm = TRUE))
```

```
## Error in group_by(regression_data, state): object 'regression_data' not found
```

```
empAvgs
```

```
## Error in eval(expr, envir, enclos): object 'empAvgs' not found
```

Day Two

Lagging data

What you may very well encounter in the next dataset or model you use, sometimes a situation arises when one variable is influenced by the **lag** of another variable or even by its own lag.

```
##
## ===============
## Bank Year Y  X1
## ---------------
## A    1990 15 11
## A    1991 0  10
## A    1992 0  5
## B    1990 5  9
## B    1991 -8 8
## B    1992 0  1
## ---------------
##
## ============================
## Bank Year Y  Y_lag X1 X1_lag
## ----------------------------
## A    1990 15  NA   11   NA
## A    1991 0   15   10   11
## A    1992 0    0   5    10
## B    1990 5   NA   9    NA
## B    1991 -8   5   8    9
## B    1992 0   -8   1    8
## ----------------------------
```

Two lagging tools (among many) * Lag(x, k, . . .) - base R * x is the numerical series you want to lag * k is the number of periods to lag by * Shift(x, n, type) - data.table * x is again the numerical series * n is the number of periods * type is whether you want to "lead" or "lag" and takes a character

```
unlagdf <- data_frame(x = 1:5)

lagdf <- unlagdf %>%
  mutate(x_lag = lag(x, 1))
lagdf
```

```
## # A tibble: 5 x 2
##       x x_lag
##   <int> <int>
## 1     1    NA
## 2     2     1
## 3     3     2
## 4     4     3
## 5     5     4
```

Question: How might you lag a variable in a *panel* or *cross-sectional* dataset?

Let's try creating a lagged variable in cardKrueger.

Day Three

Day Four