ARE 264-b Problem Set 1

Problem 1

See the code appendix at the end for function definitions.

1.1 Temperature Aggregation

1.1.1 Construct 4 temperature response variables

See next section.

1.1.2 Aggregate to year-county

Construction and aggregation are both handled in the `create_temperature_vars()` function.

```
if !isfile(joinpath(root, "CountyAnnualTemperature_aaron.csv"))
    create_temperature_vars()
end
```

This creates the new file of county-year temperature response variables to regress on (degree days, temperature bins, cubic splines, and piecewise linear). I have compared the values of these county-year variables with the CountyAnnualTemperature1950to2012.dta file for Autauga County and they match.

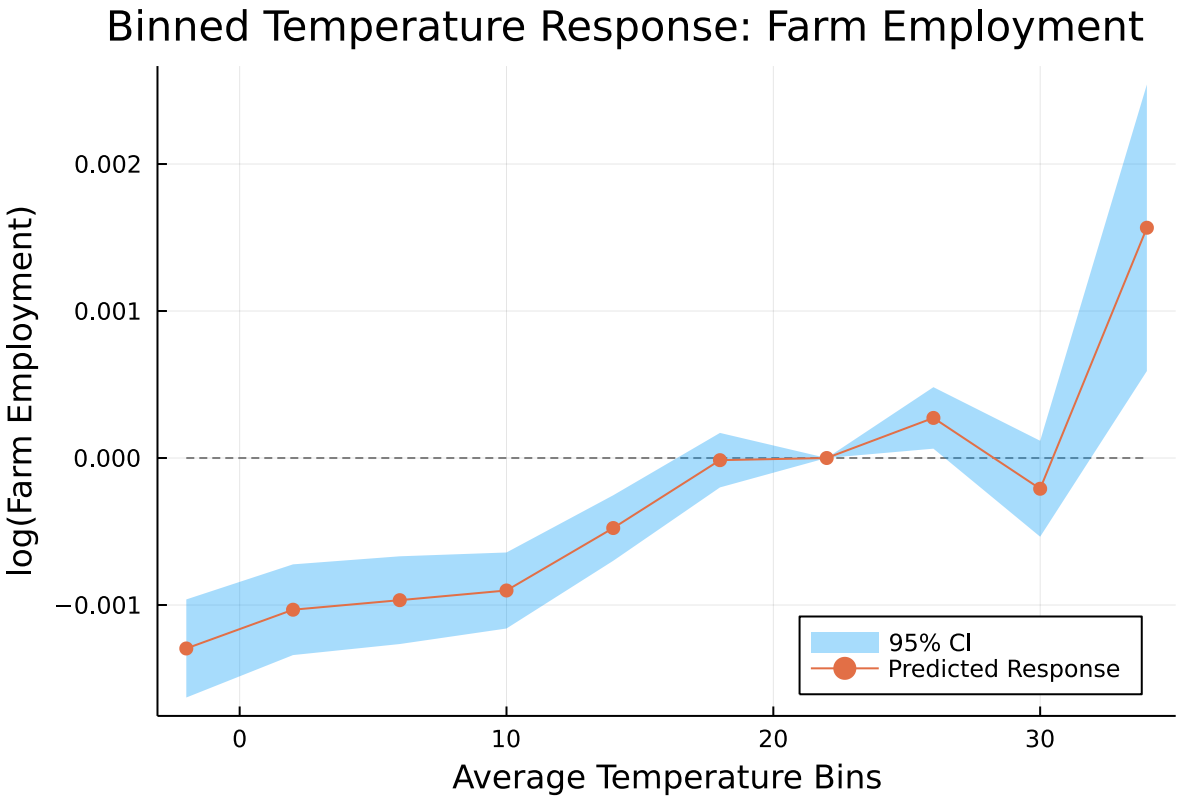
1.2 US Climate Impacts: County-Year Damages

1.2.1 log-transformed farm employment vs binned temperature

> Explore the relationship between log tranformed emp_farm and the vector of binned temperature controls. Include additional controls for county FE, year FE. Provide an interpretation of the coefficient of the 32+ bin.

Below, I produce a plot of the coefficients on the temperature bin variables and their 95% confidence intervals, where the points are plotted at the midpoint of their respective bins. I used the 20-24°C bin as the reference bin in the regression, so all coefficients represent percent of annual Farm Employment in the county changed by moving a single day's average temperature from the 20-24°C bin to each respective bin. For example, the second point from the left should be interpreted as: *On average, if we move a single day that has a daily average of 22°C to a daily average of 2°C, we expect to see a decrease in that county's farm employment by roughly 0.1 percentage points, all other days held constant.*

Note that the left-most and right-most bin represent *below 0°C* and *above 32°C* average temperature, respectively. Thus the right most point in the plot represents: *% change in farm employees for the average county for each additional day above 32°C that would have been between 20°C and 24°C*



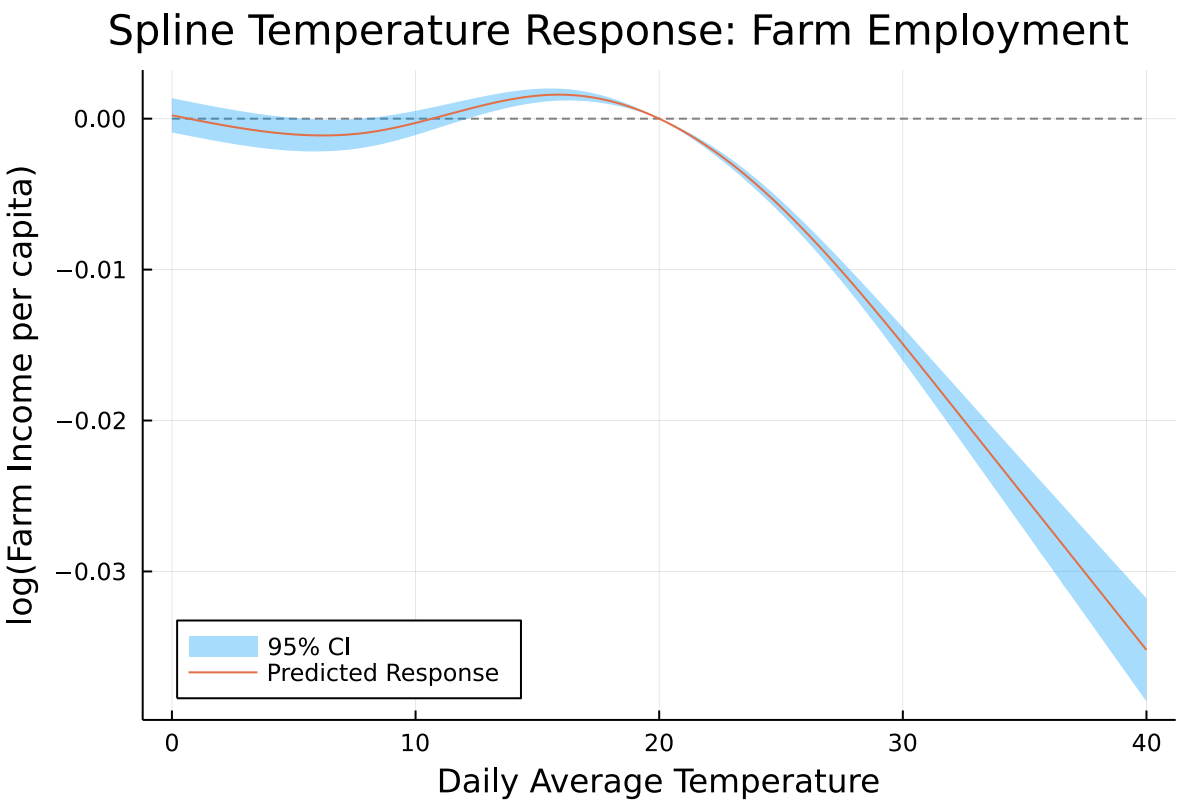
1.2.2 per capita farm property income vs cubic spline temperature

> Explore the relationship between log tranformed `inc_farm_prop_income` / population (i.e. log(per capita farm prop income) using the restricted cubic spline. Include additional controls for county FE and year FE. Plot the predicted marginal effects with associated confidence intervals, and compare to the binned temperature response function above.

Let's compare the shift from 20°C days to cooler days between the plot above and the spline plot below. In the previous plot, we saw that farm employment decreases when shifting to cooler days, but in the below plot, the farm's income (normalized to the county population) seems to be fairly steady when moving moderate days to cooler days. Perhaps this is because it takes fewer hands to harvest when the season is cooler because the product ripens slower.

In the plot above, farm employment incrases when shifting 20°C days to hotter days, and in the plot below, the farm's income seems to be signficantly decreasing when more moderate days are made hot. Similarly, perhaps hotter seasons require more intense (faster) harvesting of the produce. So more people could be employed in the county, but the farm may not make as much due to more product over-ripening.

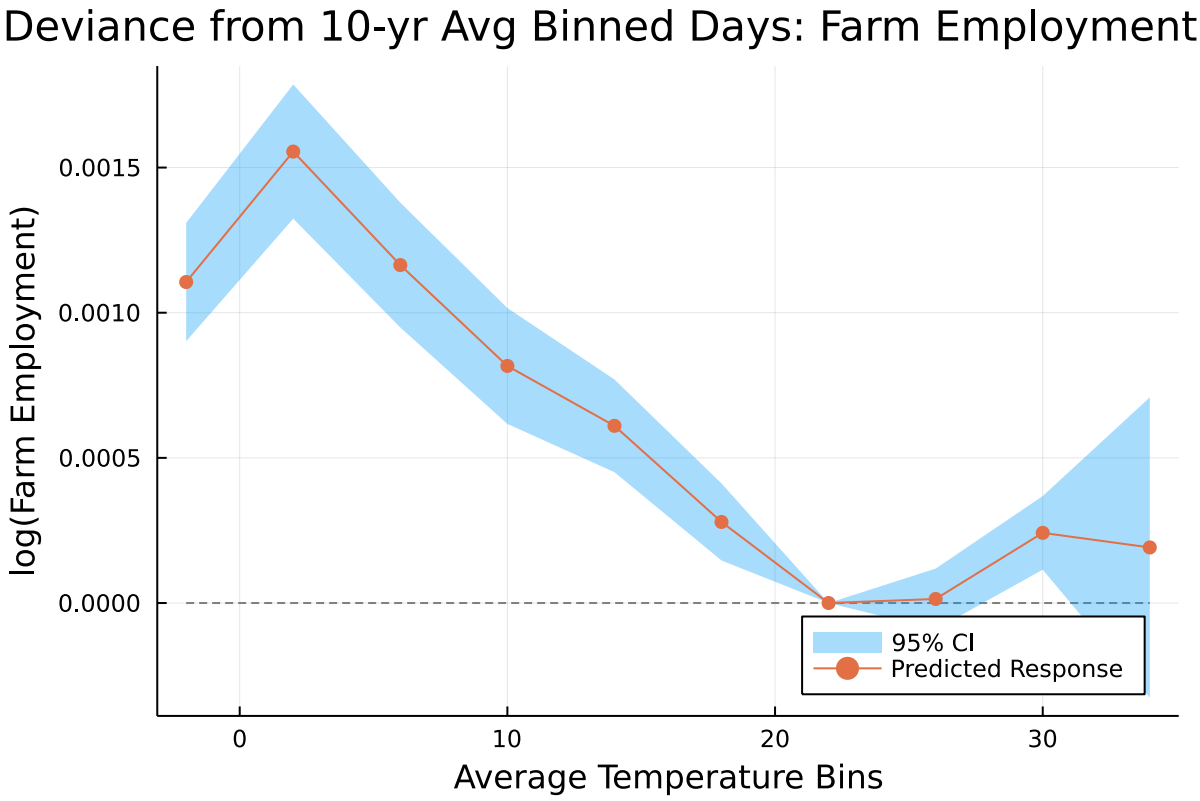
The confidence intervals were created using the delta method. Bootstrapping was also explored – clustered bootstrapping (at the county level,w ith 10,000 iterations) resulted in confidence intervals that were nearly identical to the delta method confidence intervals. The bootstrapping results are omitted here for brevity.

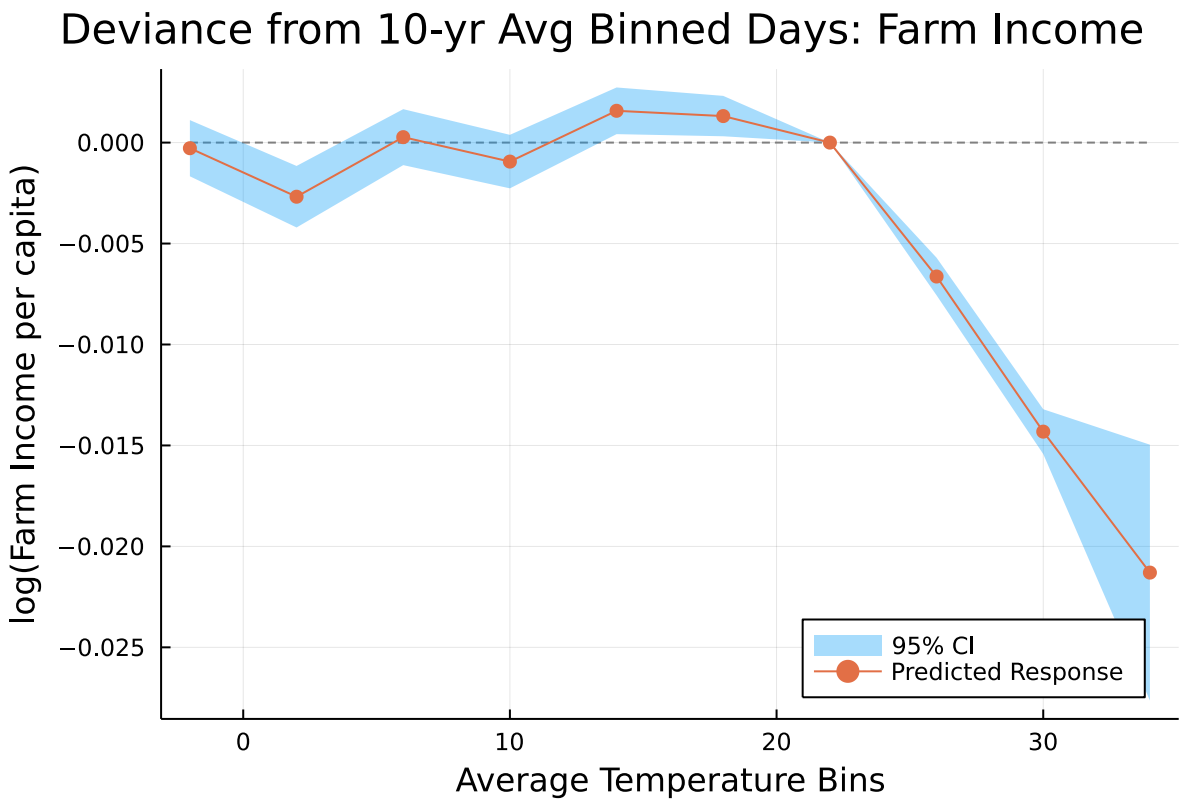


1.2.3 Treatment heterogeneity

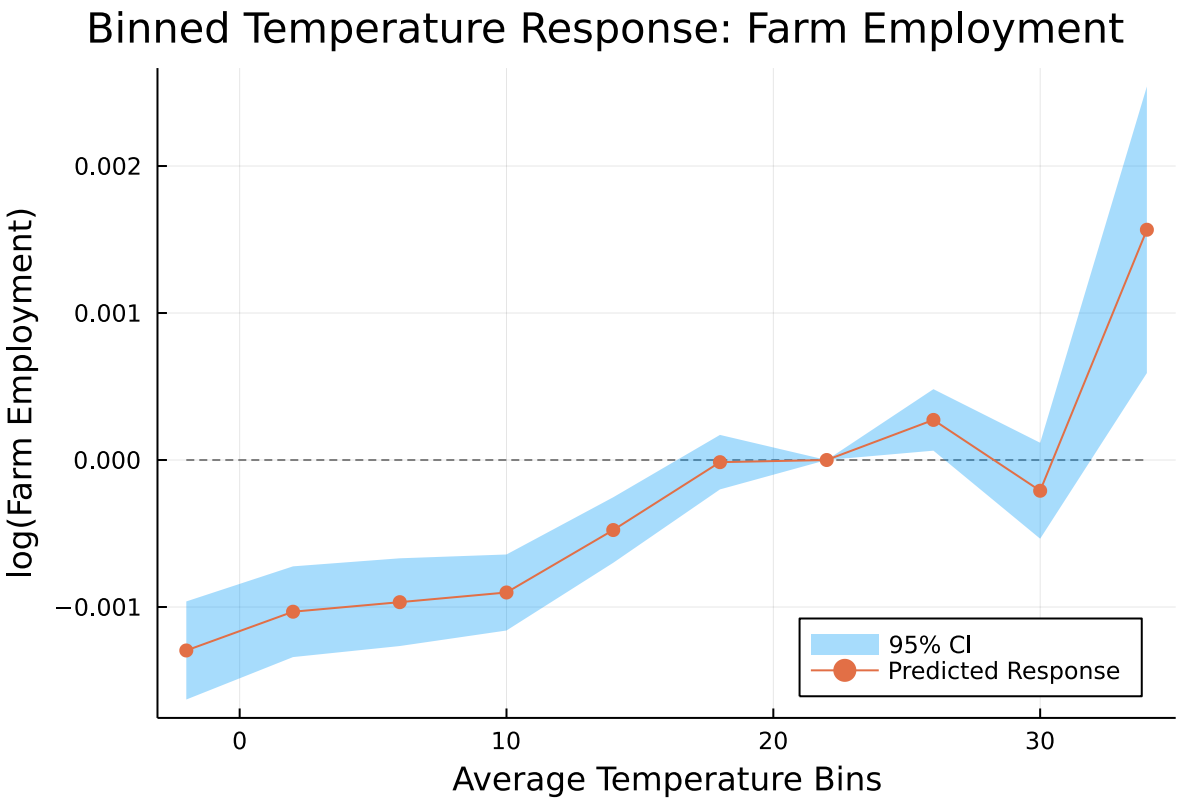
> Use the binned temperature estimator to design a test for whether we observe treatment effect heterogeneity. One possibility for this test is to interact the temperature bins with the average # of days in a county for which the temperature falls into each respective bin (i.e. are counties that experience more 32+ days more/less responsive to temperature).

To estimate treatment heterogeniety, I first calculated the 10-year moving average (backward looking) for each year-county-temperature bin. This is functionally a medium-term historical average of days of the year with average temperatures in each bin. So the first 10 years of data are now unused for these plots. For each year-county-bin, I then calculated the difference of the bin from the 10-year average (the deviance from the average). I then regressed both Farm Employment and Farm Income per capita on the new deviance bins.

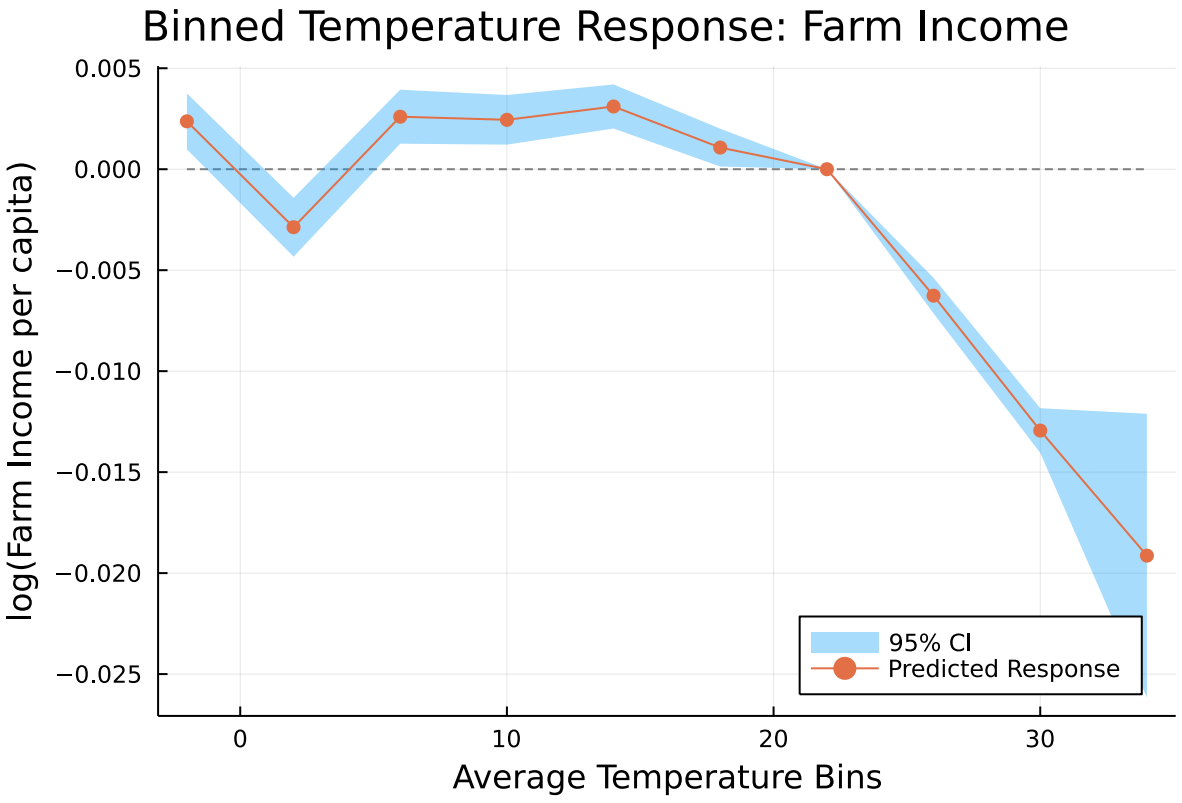




To compare, I again plot the basic temperature bins' effects, but for both Farm Employment and Farm Income

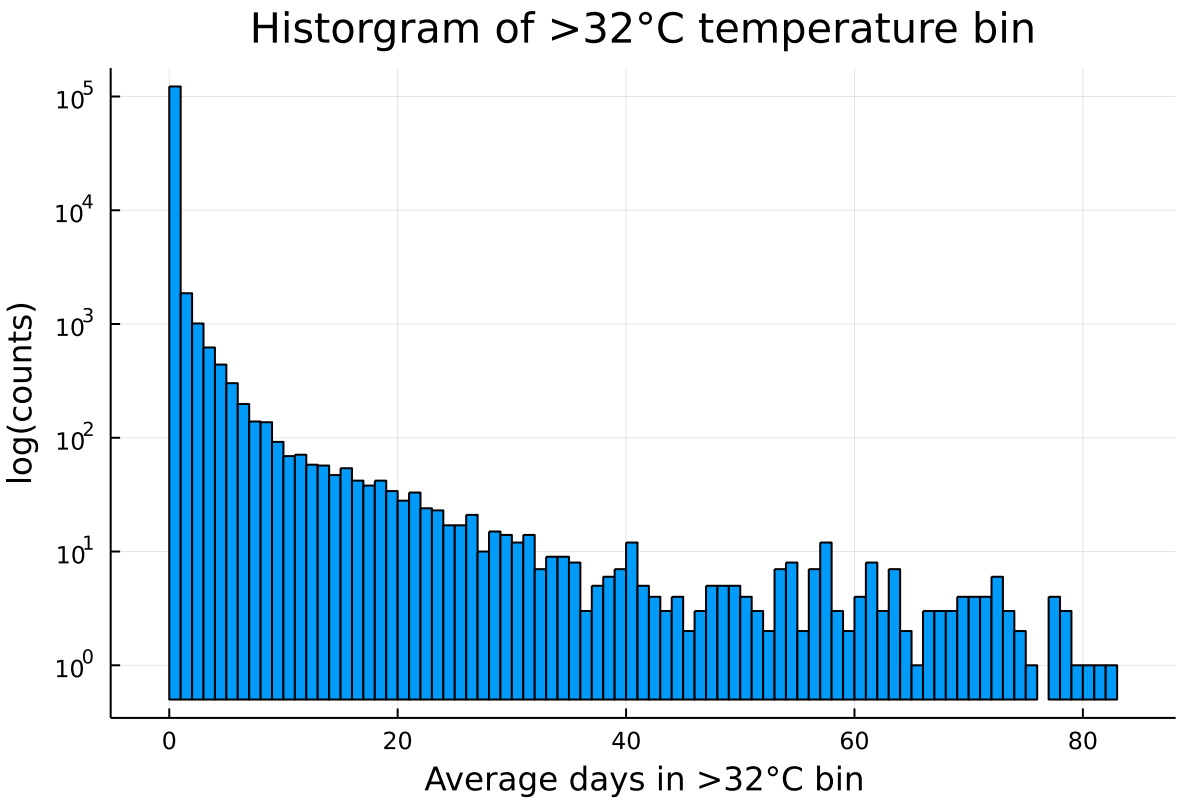



```
plot123b(:emp_farm_ln, df)
```



1.2.4 Within-bin Treatment heterogeneity

In the presence of heterogeneous treatment effects, the OLS coefficient estimate is a weighted average of the heterogeneous treatment effects, where the weights depend on the relative size of the groups and the conditional variance within each group. Derive the OLS regression weights for the 32C+ regressor using methods outlined in Angrist and Krueger (1999), Section 2.3. Replicate figure 3b in Angrist and Krueger, replacing schooling with the support of the 32C+ regressor. Where is most of the weight coming from within the 32C+ regressor? How does this affect our understanding of the the temperature-outcome dose-response relationship?



I haven't produced the weighting function plot yet, but based on the histogram above, we can see that the majority of the county-year observations have 0's in the 32+ bin. (Note that the y-axis is log scaled and the 0-1 bin has nearly 100 times as many observations than even the second most populated bin.) In fact, **the median of the 32+ bin is 0 days**. Angrist and Krueger (1999) discuss that the weighting function will tend to be larest near the median treatment values. This means that the treatment estimate for the 32+ bin will be heavily weighted based on the treatment effect felt by county-years who had zero treatment of 32+ temperatures. I think this means that the the average treatment effect we estimate within the 32+ bin will be underestimating (in magnitude) the true effects of all feasible temperatures above 32°C because we will be mostly estimating the effects with observations that have zero or very few days in the 32+ bin.

Problem 2

2.2 Hedonic Air Quality Analysis

Load pollution data

2.2.1 Housing prices vs air pollution

> Estimate the relationship between changes in air pollution and housing prices, both not adjusting and adjusting for other housing price determinants (use pop7080 as weights). What do your estimates imply and do they make sense? Describe the potential omitted variables biases. What is the likely relationship between economic shocks and pollution and housing price changes? Using the observable measures of economic shocks (dincome, dunemp, dmnfcg), provide evidence on this.

	dlhouse		
	(1)	(2)	(3)
dgtsp	0.003***	0.001	0.000
	(0.001)	(0.000)	(0.000)
dlhouse Mean	1.075	1.076	1.076
Controls	N	Y	Y
Econ. Shocks	N	N	Y
N	1,001	994	994
R2	0.079	0.455	0.572

Both regressions (1) and (2) seem to indicate that increased pollution increases housing values! This doesn't makes sense since air pollution should be a disamenity. This result could be caused by OVB, if the omitted variable has correlation with housing price in the same direction as the correlation with pollution (ie, if x2 is the OV, cov(x2, pollution) and cov(x2, houseprice) are both the same sign). For example, if there is a recession, and the economy slows, this generally decreases the amount of pollution being produced by firms supplying goods to the economy. So an indicator of positive economic activity would be positively correlated with pollution. At the same time, a recession would likely cause the average price of housing to decrease.

So an indicator of positive economic activity would be positively correlated with housing price. Then bias from an omitted economic variable that is positively correlated with both pollution and housing prices would cause an upward bias in our estimate of the partial derivative of house price changes w.r.t. pollution changes. If the bias is large enough, it could flip the sign of the coefficient, and since our coeffience on pollution is nearly statistically indistinguishable from 0 after controlling for other observed housing determinants, upward OVB could be likely here.

There was a large recession in the 70s, so variables describing the impact of the recession on the supply and demand for housing seem like they would be important to control for. For example, many manufacturing jobs were lost during the recession (more relative to other sectors) which affects the housing market. We can test three of our economic indicators to see how correlated with pollution they are in order to see if we should be concerned about OVB.

We could further control for unemployment more generally and average county income changes to ensure we are comparing willingness to pay across similar counties. After controlling for three economic indicators, regression (3) above shows that the hedonic estimate of willingness to pay for changes in pollution become statistically indistinguishable from zero.

	dgtsp
	(1)
(Intercept)	-18.008***
	(2.883)
dincome	0.008***
	(0.002)
dunemp	-42.063
	(66.889)
dmnfcg	105.580***
	(31.109)
Estimator	OLS
N	1,001
R2	0.159

Above is the regression of changes in pollution levels on changes in various eocnomic indicators. Economic shocks can change pollution because pollution is generated by economic activity (i.e., if there is a recession, there will be less pollution) and economic shocks can change what people are able/willing to pay for a house (ie, recession -> house prices decrease), which creates endogeniety (house price is correlated with the error term).

2.2.2 Attainment status as IV for pollution changes

> Suppose that federal EPA pollution regulation is a potential instrumental variable for pollution changes during the 1970s. What are the assumptions required for mid-decade regulatory status (tsp7576) to be a valid instrument for pollution changes when the outcome of interest is housing price changes?

Assumptions:

1. Relevance: tsp7576 needs to be correlated with pollution changes.
2. Exogeneity: tsp7576 only affects housing prices through it's effect on changes in pollution.

> Provide evidence on the relationship between the regulatory status indicator and the observable economic shock measures. Interpret your findings.

```
----------------------
               tsp7576
              --------
                   (1)
----------------------
(Intercept)   0.474***
               (0.132)
dincome         -0.000
               (0.000)
dunemp          -0.084
               (2.976)
dmnfcg          -0.800
               (0.899)
----------------------
Estimator          OLS
----------------------
N                1,001
R2               0.006
----------------------
```

The above regression of the regulator status indicator on our measures of economic shock shows that there is vary little statistical relationship between the regulatory status indicator and economic shocks. This supports our assumption of exogeneity of the regulatory status as an IV for changes in pollution.

2.2.3 First stage, reduced form

> Document the first-stage relationship between regulation and air pollution changes and the reduced form relationship between regulation and housing price changes, both not adjusting and adjusting for other covariates. Interpret your findings. How does two-stage least squares use these two equations? Now estimate the effect of air quality changes on housing price changes using two-stage least squares and the tsp7576 indicator as an instrument (not conditioning and conditioning on other observables). Interpret the results.

First-stage (relevance) relationship between regulation (`tsp7576`) and air pollution changes (`dgtsp`); and *Reduced form* relationship between regulation (`tsp7576`) and housing price changes (`dlhouse`). (SE in parentheses)

	dgtsp		dlhouse	
	(1)	(2)	(3)	(4)
tsp7576	-8.774***	-7.424***	0.049	0.055**
	(2.256)	(1.912)	(0.038)	(0.021)
Dependent Var Mean	-8.300	-8.233	1.075	1.076
Controls	N	Y	N	Y
N	1,001	994	1,001	994
R2	0.052	0.177	0.014	0.467

How does two-stage least squares use these two equations?

The 2SLS regression coefficient is the ratio of the reduced form relationship to the first-stage relationship. Formally, if the first stage relationship is given by:

$$x = \alpha z + u$$

and the reduced form relationship is given by

$$y = \gamma z + \varepsilon$$

then the IV (2SLS) coefficient β that we want to estimate from

$$y = \beta x + e$$

will be

$$\beta = \frac{\gamma}{\alpha}$$

So, without using statistical controls, our IV estimate of the effect of pollution on house prices can be calculated by dividing the coefficient from regression (3) above by the coefficient from regression (1). With statistical controls, we would divide the coefficient from (4) by that from (2).

I run the 2SLS model below, and then test if the ratio of the above coefficients are equal to the 2SLS coeficients below. They do indeed match, both with and without controls. However, the 2SLS model below corrects the estimates the standard errors for the ratio.

Looking at regression (2) in the table below, the coefficient of -0.007 indicates that a 1-unit decrease in *the change in mean TSPs* from 1970 to 1980 results in an average housing price change from 1970 to 1980 *being 0.7 percentage points larger* (more positive).

To make the interpretation more concrete, let's look at an example county. First note that the dependent variable (`dlhouse`) is the natural log of the ratio of housing prices in the county (1970 / 1980). So, for a given value of `dlhouse` ,

$$dlhouse = log\left(\frac{hprice_{1980}}{hprice_{1970}}\right)$$

a percentage increase would be

$$\% \text{ increase from 1970 to 1980} = 100 \cdot (e^{dlhouse} - 1)$$

Then, take an average county (County X) that saw a decrease of 8.2 units of pollution (an improvement in air quality) and an increase in housing prices by about $100(e^{1.076} - 1)$ = 193.3% between 1970 and 1980. If instead they saw one unit more of pollution improvement (a 9.2-unit improvement instead), then we would expect that County X housing prices would have instead increased by

193.3% - (100β_{IV})% $\approx$ 193.3% + 0.7% = 194.0%

showing evidence that, on average, pollution decreases caused by the regulation have increased house prices more than they would have otherwise increased.

2SLS Regression of changes in housing prices on changes in pollution, using an indicator for regulation as an IV for pollution changes (SE in parentheses)

	dlhouse	
	(1)	(2)
dgtsp	-0.006	-0.007*
	(0.005)	(0.004)
dlhouse Mean	1.075	1.076
dgtsp Mean	-8.300	-8.233
Controls	N	Y
N	1,001	994
R2	-0.564	-0.093
Adjusted R2	-0.566	-0.107
F	1.280	10.801
First-stage F statistic	15.129	15.071

Testing the 2SLS regression coefficients (regressions `223e` and `223f`) for equality with the ratio of coefficients from the reduced-form and first-stage regressions:

```
true
```

```
last(reg223e.coef) ≈ last(reg223c.coef) / last(reg223a.coef)
```

```
true
```

```
last(reg223f.coef) ≈ last(reg223d.coef) / last(reg223b.coef)
```

2.2.4 Mean of TSPs as IV for pollution changes

In principle, the regulation indicator variable should be a discrete function of pollution levels in 1974. Specifically, the EPA is supposed to regulate those counties in 1975 who had either an annual geometric mean of TSPs above 75 units (mg/m3) or a 2nd highest daily concentration above 260 units in 1974. Based on this notion, redo part (3) using mtspgm74 as an instrument for pollution changes. Interpret your findings.

First stage and reduced form regressions (SE in parentheses)

	dgtsp		dlhouse	
	(1)	(2)	(3)	(4)
above75	-11.917***	-10.421***	0.054	0.061*
	(2.049)	(1.864)	(0.054)	(0.025)
dlhouse Mean	-8.249	-8.181	1.075	1.076
Controls	N	Y	N	Y
N	974	967	974	967
R2	0.077	0.193	0.014	0.465

2SLS Regression of changes in housing prices on changes in pollution, using an indicator for above 75 pollution units as an IV for pollution changes (SE in parentheses)

	dlhouse	
	(1)	(2)
dgtsp	-0.005	-0.006*
	(0.005)	(0.003)
dlhouse Mean	1.075	1.076
dgtsp Mean	-8.249	-8.181
Controls	N	Y
N	974	967
R2	-0.424	0.104
Adjusted R2	-0.425	0.091
F	0.960	13.729
First-stage F statistic	33.838	31.260

2.2.5 Discontinuity in treatment assignment

> Describe how one could use this discontinuity in treatment assignment to derive an alternative estimate of the capitalization of pollution changes. Under what conditions will this estimate be valid? Based on this concept, estimate the nonparametric bivariate relationships between pollution changes and 1974 TSPs levels and housing price changes and 1974 TSPs levels. Do this using the lowess STATA command or a similar command in R (experiment with relatively small bandwidths between 2-4). Plot the two conditional mean functions on either side of the discontinutiy (changes in air quality for discrete 1974 values on either side of threshold). Interpret your findings and relate them to the results in (4).

I can imagine using either regression discontinuity (RD) or differences-in-differences (DiD) around the 1974 TSP threshold to analyze the capitalization of pollution changes. Because RD takes a relatively small group of observations around the threshold, it depends on having enough data to estimate the effects. Because the regulation is not only dependend on the 1974 threshold for the mean TSPs (it also depends on the 2nd highest daily concentration), this would be a fuzzy RD. DiD on the other hand could use the entire dataset above and below the threshold and would almost surely have more smaller standard errors on the treatment effect estimates of the regulation.

Assumptions of the RD:

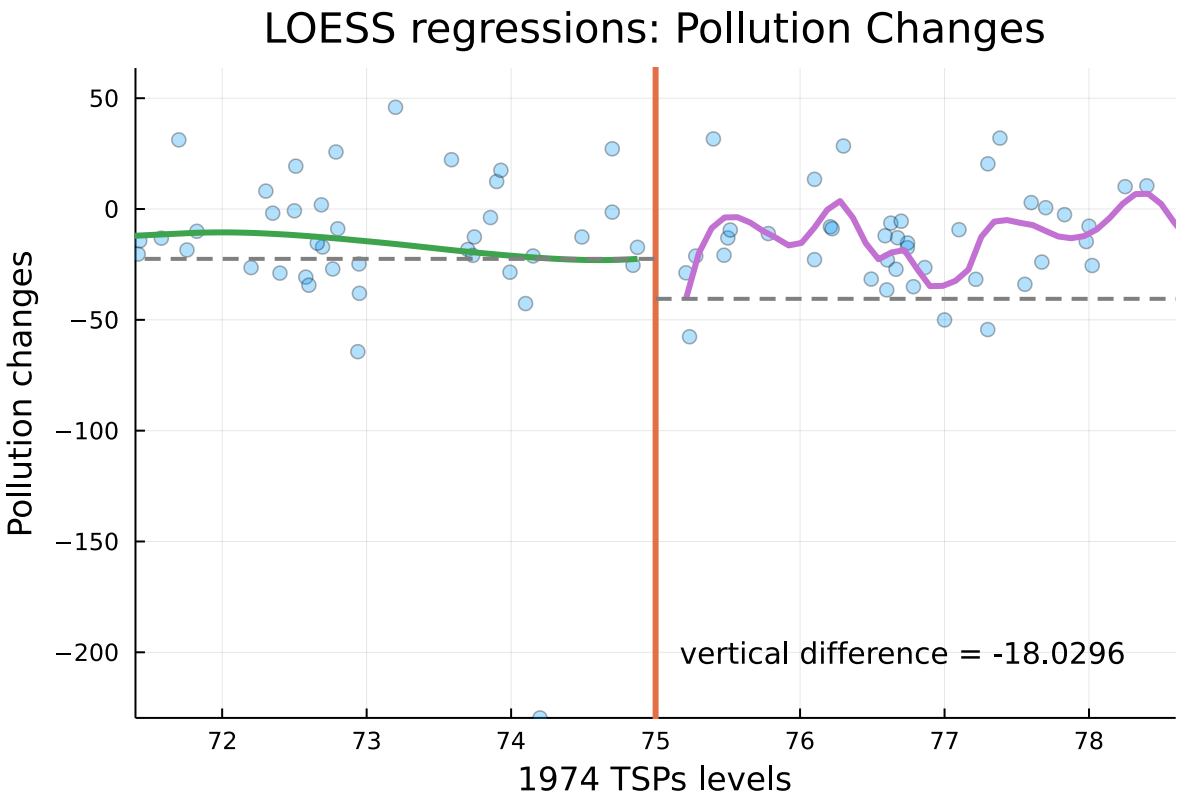
- No manipulation: there shouldn't be signs of units manipulating their values very close to the cutoff to avoid being regulated.
- Units (counties) just above and just below the threshold should be very similar along other observed and unobserved characteristics that are important in determining the outcomes.

Bandwidth =

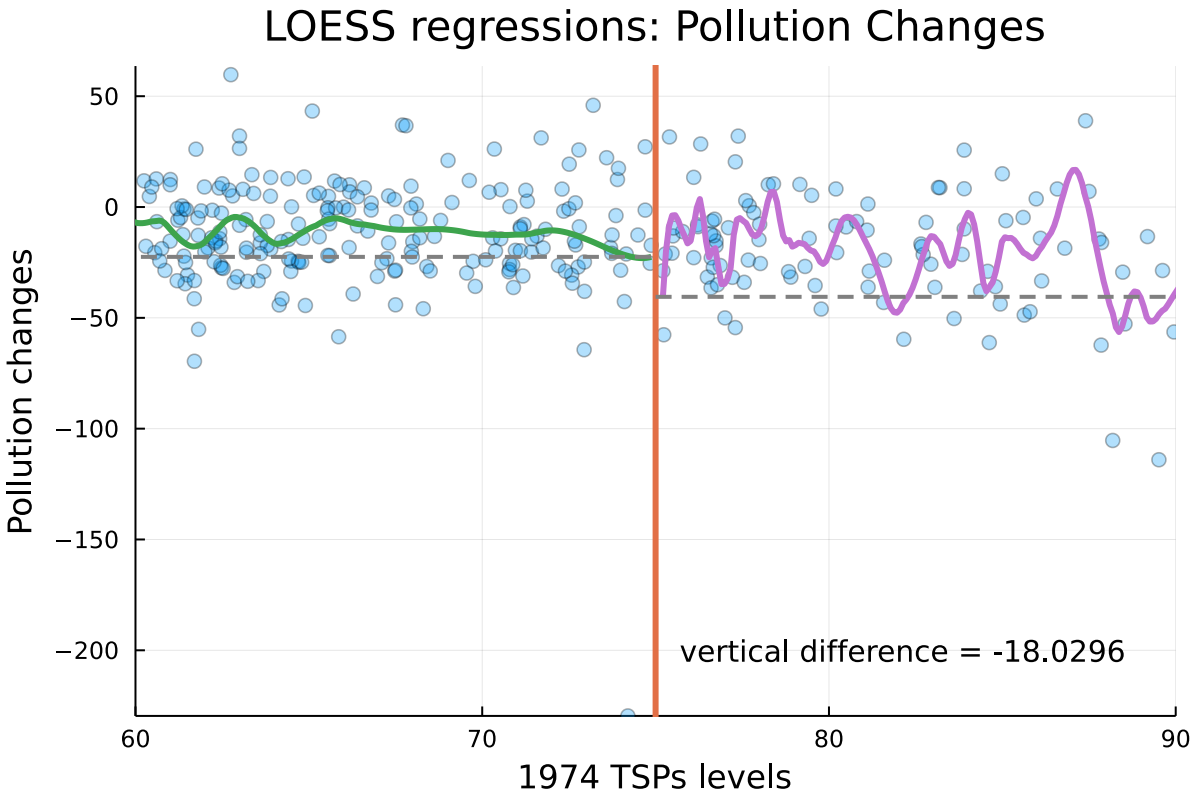
PredictionPoints = 1000

PolynomialDegree = 3

PlotWidth =



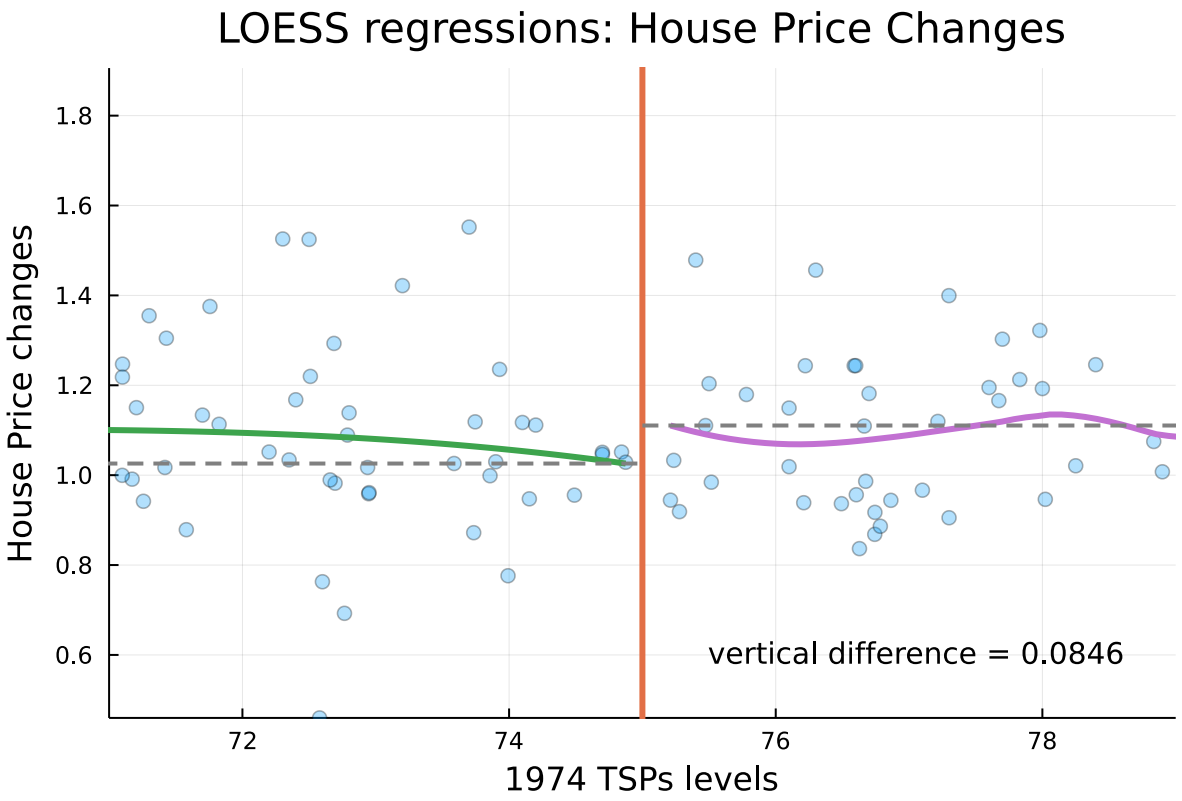
But when we zoom out, the nonparametric estimation further away from the threshold is even more noisey. So how do we know the difference in the estimates near the threshold aren't just due to sampling variation? One way to examine the variance due to sampling this is to use permutation inference: hold the bandwidth constant, and resample the data many times, plotting the mean of the LOESS estimate and the 2.5^{th} and 97.5^{th} percentiles of the prediction at each point.



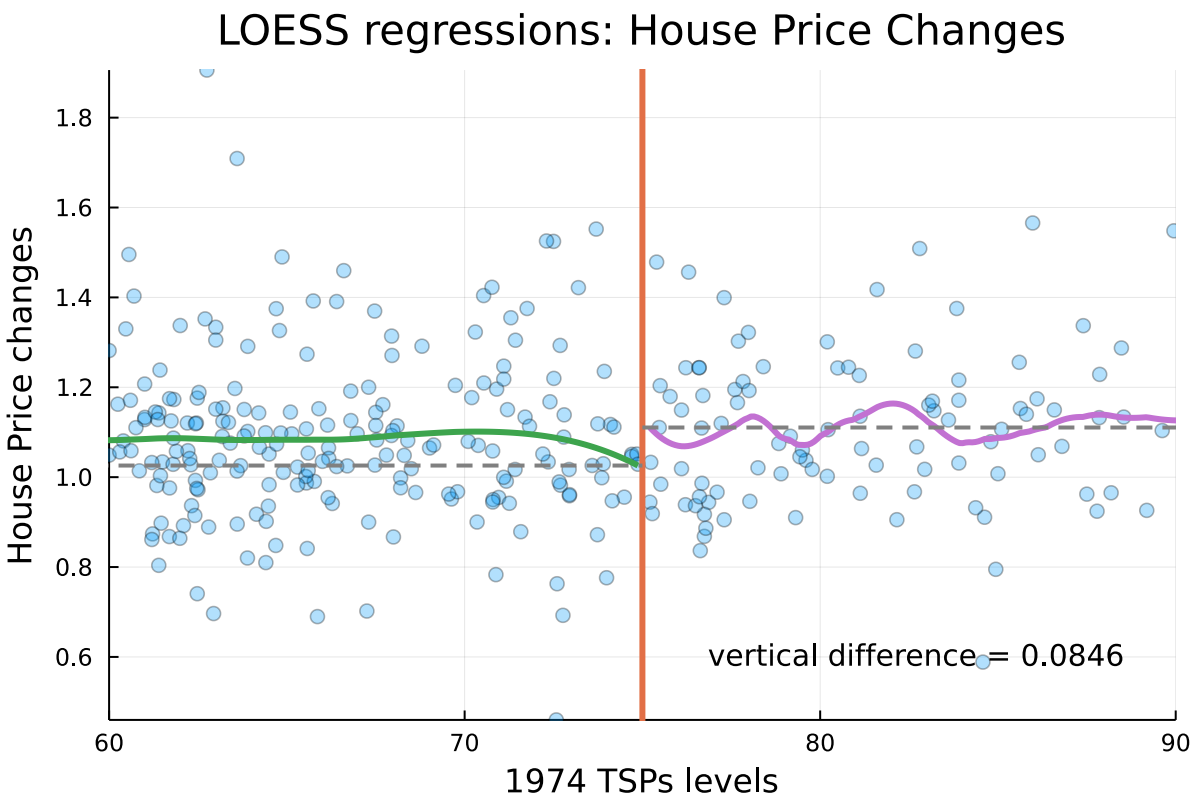
When we look below at changes in house prices based on the 1974 TSP levels, and using a wider bandwidth, it appears that the house price changes were smaller just below the threshold.

`BandwidthHousePrice` =

`PlotWidthHousePrice` =



Same bandwidth, just zoomed out to see more of the data.



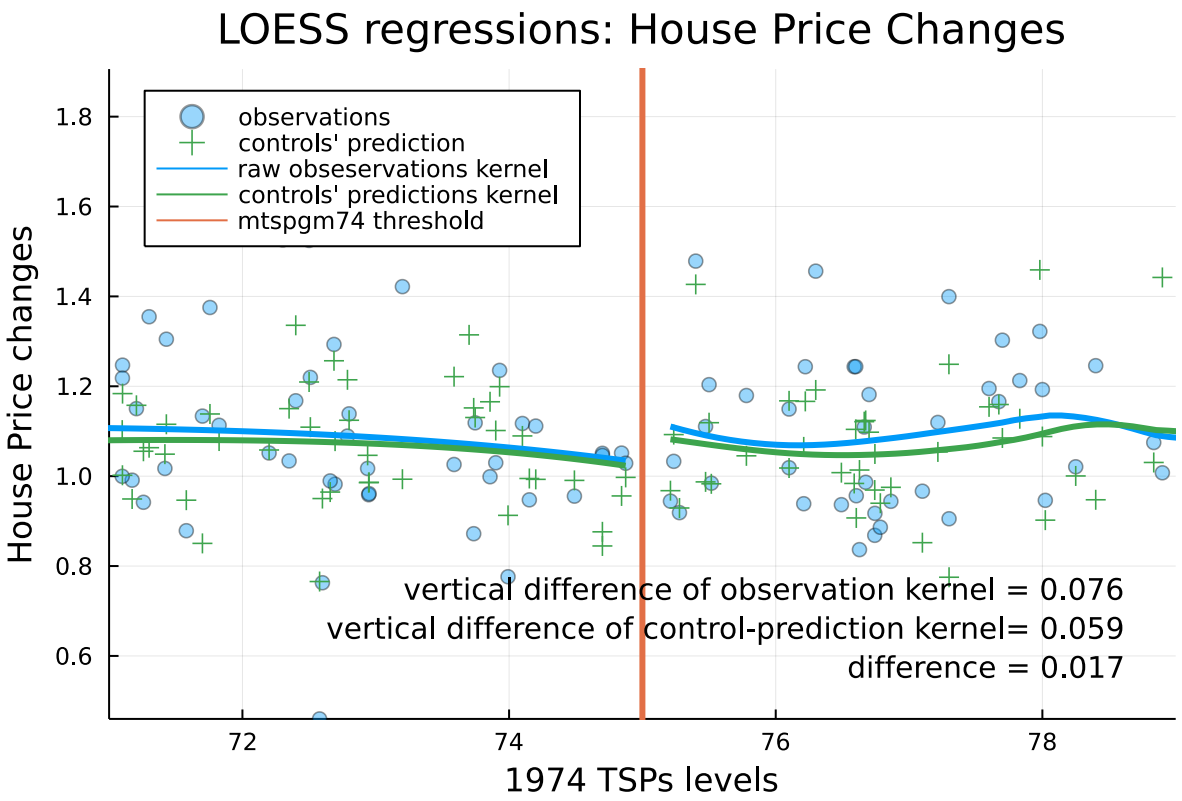
In part (4), we saw that less improvement in pollution levels, as instrumented by being below this threshold, results in smaller increases in housing prices. If we were to take the plot above at face value, we can see that counties just below the threshold experienced a smaller increase in housing prices compared to counties just above the threshold. From the first two plots above of the pollution changes, we saw that counties just below the threshold (presumably less regulated) saw less of a pollution improvement (the decrease in pollution was smaller in magnitude, on average, just below the threshold).

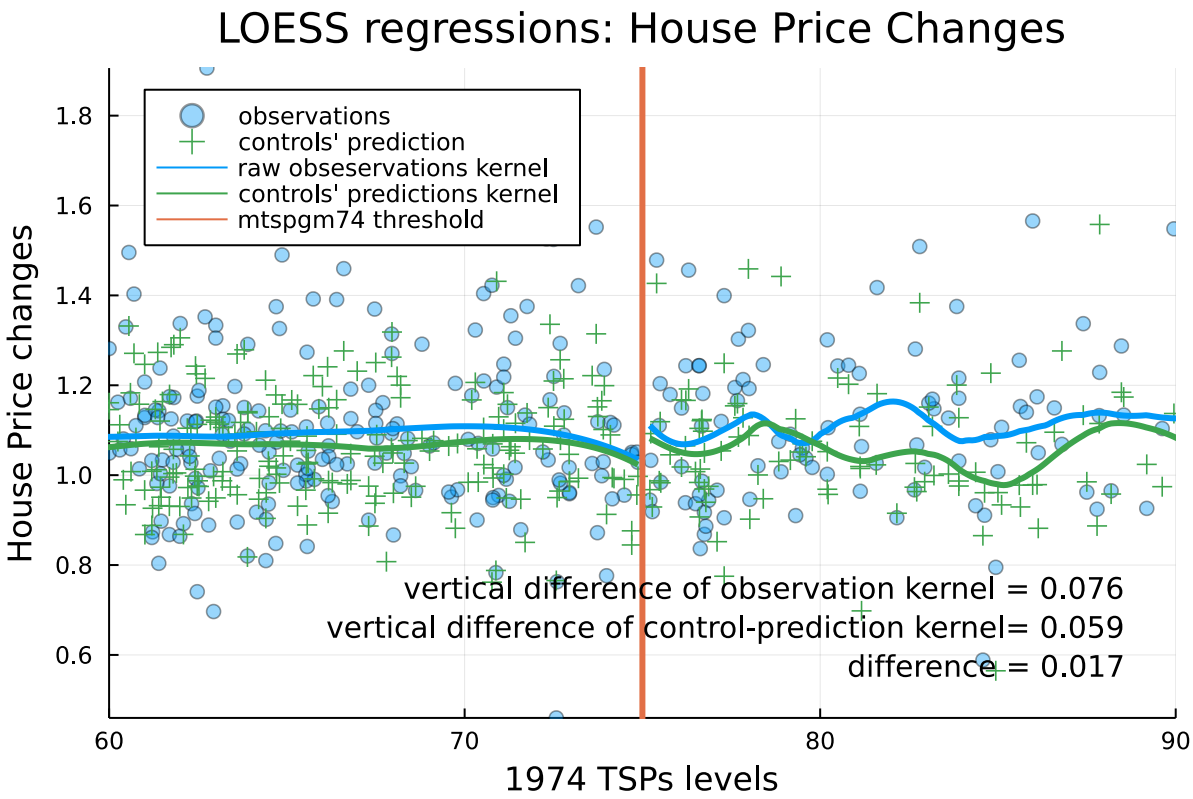
So, here we see that less improvement in pollution levels (just below the threshold) is correlated with a smaller increase in housing prices (though this result is highly dependent on the choice of bandwidth).

2.2.6 The Smoothness Criteria of RD

> Now use linear regression to estimate the predicted change in housing prices based on the control variables (excluding the change in TSPs). This provides a single-index measure of the housing price changes predicted to occur due to other variables changing. Estimate the nonparametric bivariate relation between this index and 1974 TSPs levels and plot it against the smoothed housing price changes from (5). Explain how this provides an indirect test of the smoothness condition required by the regression discontinuity design and interpret your findings.

The second assumption about RD from part (5) describes covariate balance between observations just above and just below the cutoff. This is roughly equivalent to a smoothness condition across the cutoff: the predicted change in housing prices due to all other variables should be smooth across the cutoff, so that we can attribute any observed discontinuity in the change in housing prices across the cutoff to the treatment (regulation).





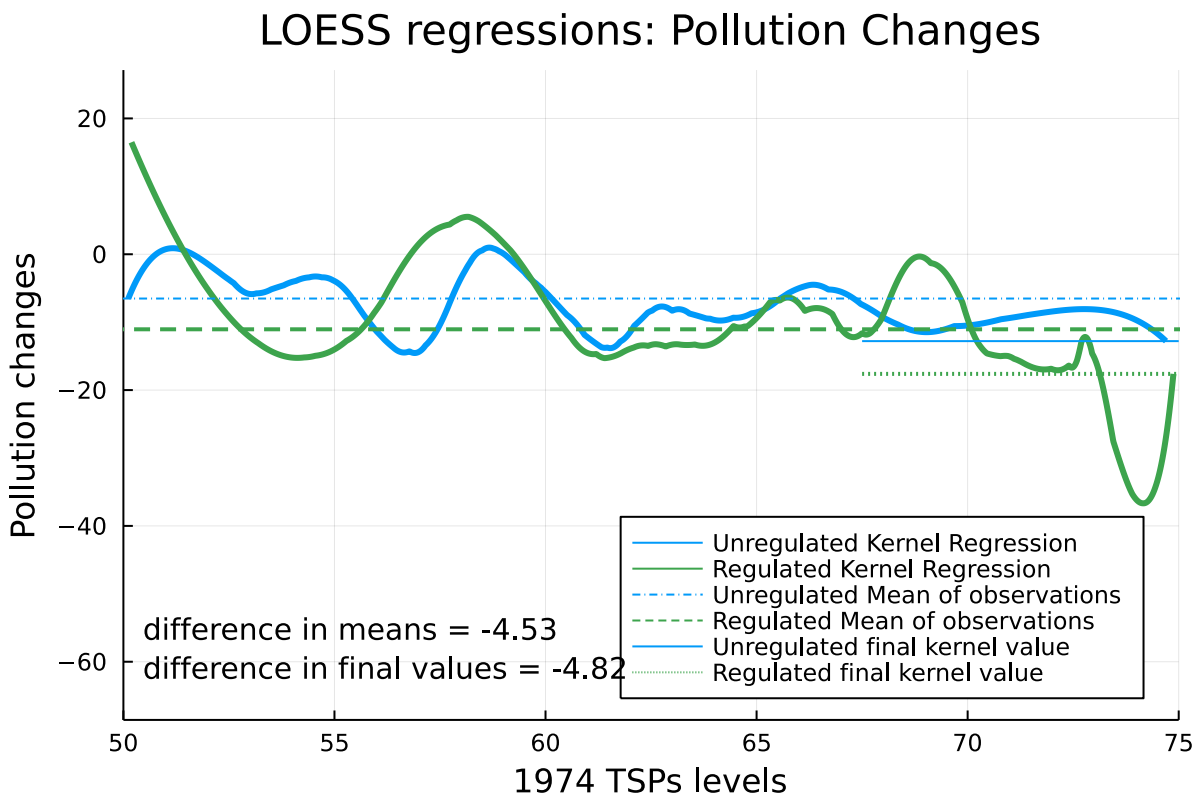
Ideally, to visually test the smoothness condition of the RD design, we would want to see a noticable discontinuity at the threshold for the kernel regression of observed changes in housing price on TSP levels, while also seeing a smooth transition across the threshold for kernel regression of predicted changes in housing price on TSP levels. Since the predicted house price changes are based on control only (and not pollution changes), if those changes show a smooth relationship across the TSP threshold, that is evidence of good covariate balance across the threshold (ie, it's less likely that there is a large difference in the covariates across the threshold that could also create a discontinuity in house price changes).

In the plots above, we see some evidence of this, but it is not conclusive. With this bandwidth, the blue kernel regression (observed data) shows what could be a noticable discontinuity across the threshold. Due to the lack of more data very close to the threshold, it is hard to tell what could happen in the middle. But the green kernel regression (predicted data) looks a little more like it could meet at the threshold. However, even the green lines show some difference on either side of the threshold. But this might provide some indirect evidence that there is some discontinuity happening at the threshold that is not accounted for by the control variables.

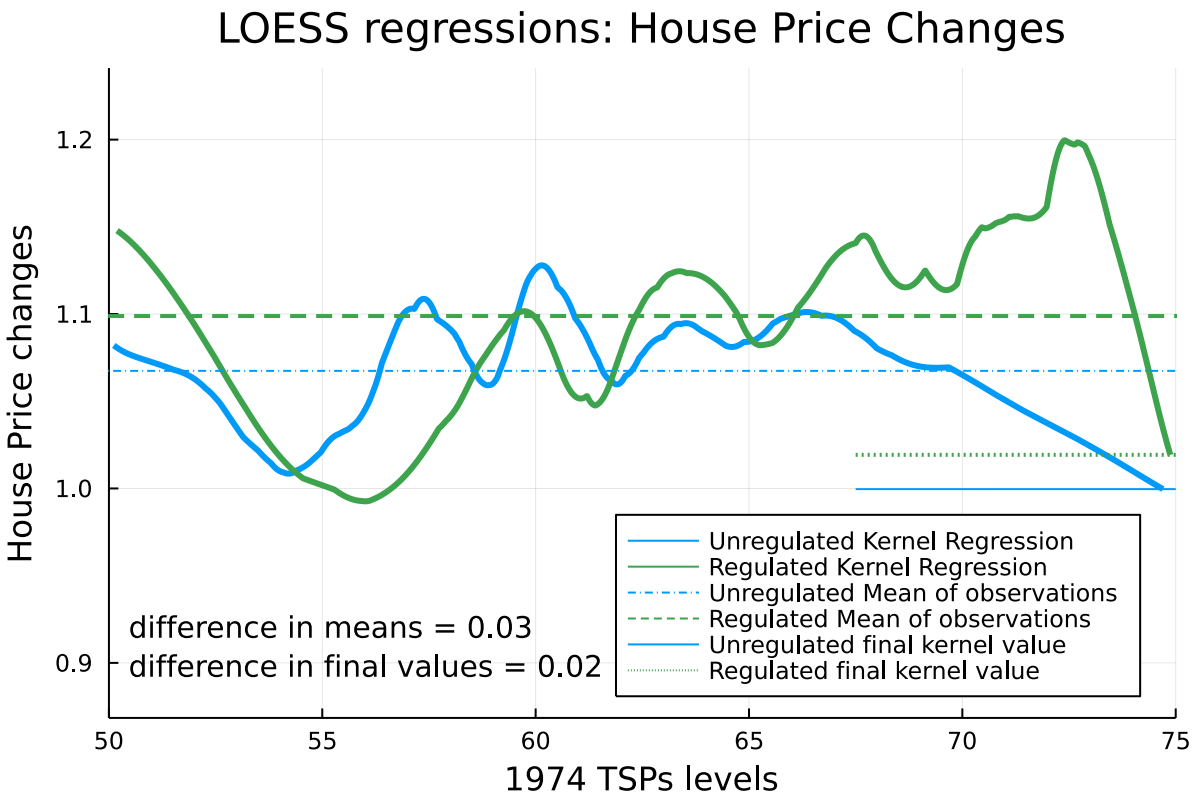
2.2.7 Regulated vs Unregulated

A number of counties with annual geometric mean TSPs below 75 units in 1974 were regulated due to having as few as 2 bad days. This implies that we can compare regulated and unregulated counties with identical average TSPs levels in the regulation selection year (below 75 units). For those counties with 1974 mean TSPs between 50 and 75 units, estimate the bivariate relation between TSPs changes and 1974 TSPs levels separately for regulated and unregulated counties and plot the results. Now do the same for the relation between log-housing price changes and 1974 TSPs levels. Interpret your findings. Since there are fewer observations, you may need to use bigger bandwidths than those in part (5) (e.g., bandwidths between 6-9).

BandwidthPollChange =



BandwidthHousePrice2 =



With a fairly large bandwidth (compared to earlier plots): For changes in pollution, we can see that as we get closer to the 75-unit threshold, the regulated counties show pollution improvements (larger negative pollution changes), while the unregulated counties seem to have fairly uniform average pollution improvements.

For changes in house prices, we also see that house prices for regulated counties that are higher in average TSPs in 1974 (closer to the 75-unit threshold) saw larger house price increases than unregulated counties around the same 1974 TSP levels. This provides more evidence that regulation, conditional on average TSPs, seems to have increased house prices (thus, less pollution is valuable to house buyers).

2.2.8 2SLS and the ATE

> Under what assumptions will two-stage least squares identify the average treatment effect (ATE)? If ATE is not identified, describe what may be identifiable with two-stage least squares estimation. Under what conditions is this effect identified? Give some intuition on what this effect may represent when one uses EPA regulation as an instrument variable.

Angrist, Imbens, and Rubin (1996) explain that there are five necessary assumptions to estimate causal treatment effects using 2SLS:

1. SUTVA: The Stable Unit Treatment Value Assumption implies that potential outcomes for each person `i` are unrelated to the treatment status of other individuals,
2. Conditionally Random Assignment: Assignment of values of the instrument are as good as random after conditioning on exogenous covariates.
3. The Exclusion Restriction: any effect of the instrument on the outcome of interest must be via an effect of the instrument on the endogenous variable.
4. Relevance: the instrument must have a nonzero average causal effect on the endogenous variable.
5. Monotonicity: there are no defiers (no one does the opposite of their treatment assignment, like pollute more when they are assigned to non-attainment status).

If we add to this the assumption of constant treatment effects (that every county would have the same increase in avereage house price due to being regulated by non-attainment status), then 2SLS would estimate the ATE. Without the constant treatment effects assumption, we would be estimating LATE (the local average treatment effect, or the treatment effect for the compliers). In this case, we would be estimating the effect of regulating pollution on counties who would have reduced their pollution under regulation, but not reduced pollution as much if they were not regulated (the compliers).

Further, if weaken the exclusion restriction assumption, Angrist, Imbens, and Rubin tells us that 2SLS is estimating the LATE plus a bias term equal to "the average direct effect of Z on Y for noncompliers multiplied by the odds of being a noncomplier."

2.2.9 Summary of TSP regulation effects on House Price Changes

> Provide a concise synthesis/summary of your results. Discuss the credibility of the various research designs underlying the results.

Estimated Impacts of a one-unit increase in pollution changes between 1970 and 1980

	Without Controls	With Controls
Naive Regression	0.003*** (0.001)	0.001 (0.000)
Attainment status IV	-0.006 (0.005)	-0.007* (0.004)
1974 TSP as IV	-0.005 (0.005)	-0.006* (0.003)

The naive regression estimated that more pollution in 1980 resulted on average in higher house prices (or at least unchanged when adding controls). But after applying the IV's, the estimated effect of higher pollution changes sign so that an increase in pollution would result in lower house prices (or, smaller reductions in pollution result in larger increases in house prices). Both IV's resulted in fairly similar estimates – that increasing (making more positive) the change in pollution between 1970 and 1980 by one unit resulted in a smaller increase in house prices by about 0.6 percentage points on average.

The IV however is estimating the local average treatment effect for the regulation compliers – so this is the estimated average effect for counties that would reduce their pollution more when classified as nonattainment (regulated) than if they were classified as attainment (unregulated).

Further, inspecting the visual representation of the regression discontinuity design, we see from part 2.2.6 that the vertical difference near the 75-unit regulation threshold is about 1.7 percentage points after subtracting the difference predicted by the other control covariates. This assumes zero slope however, so is an overestimate. It would be interpreted as: having a mean TSP in 1974 just above the regulatory cutoff (resulting in non-attainment status) resulted in a 1.7-percentage-point higher changes in housing values compared to counties just below the 1974 regulatory cutoff. In a fuzzy regression discontinuity, however, we would account for different slopes on either side of the threshold, likely resulting in a smaller estimate (closer to zero). However, this does seem to corroborate the sign of the IV results above.

Lastly, we can see from section 2.2.7 in our direct comparison of regulated and unregulated counties, that regulated counties with 1974 mean TSPs between 50 and 75 units saw larger house price increases – on average, 3 percentage points larger house price increases. This is based on the mean across the 50-75 unit domain. However, the difference seems even larger if looking to the upper end of the domain, but the gap shrinks rapidly to 2 percentage points difference. This also seems to corroborate our earlier evidence that regulating pollution in the 70's seems to have increased housing values in counties that were regulated and reduced their pollution.

Code Appendix: Functions

File Functions

`extract_file_from_zip`

Extracts file*name from zip folder zip*name in directory root_dir.

```
"""Extracts file_name from zip folder zip_name in directory root_dir."""
function extract_file_from_zip(root_dir, zip_name, file_name)
    zip_path = joinpath(root_dir, zip_name)
    save_path = joinpath(root_dir, file_name)
    # If the file is already extracted, done.
    if isfile(save_path)
        return
    end

    println("Extracting $file_name from $root_dir")
    # Open zip file
    zarchive = ZipFile.Reader(zip_path)
    # Find file_name in zip archive
    files = [f.name for f in zarchive.files]
    idx = findfirst(x -> x == file_name, files)
    # Save file_name to disk
    write(save_path, read(zarchive.files[idx]))
end
```

`df_from_zip`

Return dataframe, after unzipping and saving the file.

```
"""Return dataframe, after unzipping and saving the file."""
function df_from_zip(root_dir, zip_name, filename)
    extract_file_from_zip(root_dir, zip_name, filename)
    println("Loading $filename from file.")
    return DataFrame(load(joinpath(root_dir, filename)))
end
```

`df_from_url`

Read file at save_path to dataframe, if not present, download from url.

```
"""Read file at save_path to dataframe, if not present, download from url."""
function df_from_url(url, save_path)
    if isfile(save_path)
        println("Loading $save_path from file.")
        return DataFrame(load(joinpath(root, example_fn)))
    else
        println("Downloading $save_path from url.")
        download(url, joinpath(root, example_fn))
        println("Loading $save_path from file.")
        return DataFrame(load(joinpath(root, example_fn)))
    end
end
```

`convert_float32`

Return a dataframe, with types Float64 and no missing for given columns.

```
"""Return a dataframe, with types Float64 and no missing for given columns."""
function convert_float32(df, cols)
    df1 = dropmissing(df, cols=cols; disallowmissing=true)
    df1 = transform!(df1, cols .=> ByRow(Float64), renamecols=false)
    return df1
end
```

Temperature Functions

`create_temperature_vars`

Save temperature variables 1.1.1 Construct 4 temperature response variables 1.1.2 Aggregate to year-county - sum each new temperaature variable in each grid point over each year - average tMin, tMax, tAvg, perc in each gridpoint over each year - take a simple average over all grid points in the county to get county-year level

```
"""Save temperature variables
    1.1.1 Construct 4 temperature response variables
    1.1.2 Aggregate to year-county
    - sum each new temperaature variable in each grid point over each year
    - average tMin, tMax, tAvg, perc in each gridpoint over each year
    - take a simple average over all grid points in the county to get county-year
level
"""
function create_temperature_vars()
    # Extract single-county fips file from zip if not present, and read
    extract_file_from_zip(root, zip_fn, county_fn)

    # Load data and create average daily temp
    df_temp1 = DataFrame(load(joinpath(root, county_fn)))
    df_temp1[!, :tAvg] = (df_temp1[!, :tMax] .+ df_temp1[!, :tMin]) ./ 2

    # Add degree days for each day-gridpoint
    thresholds = [30, 32, 34]
    df_temp1 = add_degree_days(df_temp1, thresholds)

    # Add binned temperature variables
    bin_edges = [0, 4, 8, 12, 16, 20, 24, 28, 32]
    df_temp1 = add_bin_indicator_days(df_temp1, bin_edges)
    df_temp1 = add_bin_portion_days(df_temp1, bin_edges)

    # Add cubic spline basis variables for each day-gridpoint
    knots = [0 8 16 24 32]
    df_temp1 = add_cubic_spline_vars(df_temp1, :tAvg, knots)

    # Add piecewise linear basis variables for each day-gridpoint
    knots = [28, 32]
    df_temp1 = add_piecewise_linear_vars(df_temp1, :tAvg, knots)

    # sum over year, then average over all grid points in county
    column_tags = ["dday", "temp", "splineC", "piece"]
    df_temp = aggregate_df(df_temp1, column_tags)

    # Save the new file
    CSV.write(joinpath(root, "CountyAnnualTemperature_aaron.csv"), df_temp)
end
```

add_degree_days

Return dataframe with degree day columns, degree days over each threshold in t.

```
"""Return dataframe with degree day columns, degree days over each threshold in t."""
function add_degree_days(df::DataFrame, t::Vector)
    println("Adding degree days above thresholds in t.")
    for k ∈ 1:length(t)
        df[!, "dday$(t[k])C"] = degree_day_single_day.(df[!, :tMax], df[!, :tMin],
t[k])
    end
    return df
end
```

degree_day_single_day

Return degree days above threshold temperature for single day based on max and min temperatures.

```
"""Return degree days above threshold temperature for single day based on max and min
temperatures."""
function degree_day_single_day(Tmax, Tmin, Threshold)
    if Threshold ≥ Tmax
        return 0
    elseif Threshold ≤ Tmin
        return (Tmax + Tmin) / 2 - Threshold
    else
        return degree_day_calc(Tmax, Tmin, Threshold)
    end

end
```

degree_day_calc

Degree day calculation from Snyder 1985, using integral of sine.

```
"""Degree day calculation from Snyder 1985, using integral of sine."""
function degree_day_calc(MAX, MIN, THR)
    M = (MAX + MIN) / 2  # average temp
    W = (MAX - MIN) / 2  # half temp range
    θ = asin((THR - M) / W)  # radian between -π/2 and π/2 where sine crosses THR
    DD = ((M - THR)*(π / 2 - θ) + W * cos(θ)) / π
    return DD
end
```

add_bin_indicator_days

Return dataframe with binned temperature variables columns (indicator for tAvg in bin).

```
"""Return dataframe with binned temperature variables columns (indicator for tAvg in bin)."""
function add_bin_indicator_days(df::DataFrame, edges::Vector)
    println("Adding inidicator for average daily temperature in each bin")
    for k ∈ 1:length(edges)
        if k == 1
            # tAvg below lowest threshold
            df[!, "tempB$(edges[k])"] = (df[!, :tAvg] .< edges[k])
        else
            #  tAvg between this threshold and previous threshold
            df[!, "temp$(edges[k-1])to$(edges[k])"] = (edges[k-1] .≤ df[!, :tAvg] .< edges[k])

            if k == length(edges)
                # tAvg above threshold
                df[!, "tempA$(edges[k])"] = (df[!, :tAvg] .≥ edges[k])
            end
        end
    end
    return df
end
```

add_bin_portion_days

Return dataframe with binned temperature variables columns (portion of day in bin).

```
"""Return dataframe with binned temperature variables columns (portion of day in bin)."""
function add_bin_portion_days(df::DataFrame, bins::Vector)
    println("Adding portion of days between bin edges in bins.")
    for k ∈ 1:length(bins)
        if k == 1
            # Portion of day below lowest threshold
            df[!, "tempPropB$(bins[k])"] = 1 .- fraction_above_threshold.(df[!, :tMax], df[!, :tMin], bins[k])
        else
            # Portion of day between this threshold and previous threshold
            df[!, "tempProp$(bins[k-1])to$(bins[k])"] = fraction_between_thresholds.(df[!, :tMax], df[!, :tMin], bins[k-1], bins[k])

            if k == length(bins)
                # Portion of day above threshold
                df[!, "tempPropA$(bins[k])"] = fraction_above_threshold.(df[!, :tMax], df[!, :tMin], bins[k])
            end
        end
    end
    return df
end
```

fraction_above_threshold

Return portion (0 to 1) of day's temperature above temperature Threshold, given max and min temperature, based on sinusoidal curve to model diurnal temperature cycle.

```
"""Return portion (0 to 1) of day's temperature above temperature Threshold,
    given max and min temperature,
    based on sinusoidal curve to model diurnal temperature cycle.
"""
function fraction_above_threshold(Tmax, Tmin, Threshold)
    α = convert_temp_threshold(Tmax, Tmin, Threshold)
    if α ≥ 1
        return 0
    elseif α ≤ -1
        return 1
    else
        return (π - 2 * asin(α)) / (2π)
    end
end
```

fraction_between_thresholds

Return portion (0 to 1) of day's temperature between temperature Thresholds, given max and min temperature, based on sinusoidal curve to model diurnal temperature cycle.

```
"""Return portion (0 to 1) of day's temperature between temperature Thresholds,
    given max and min temperature,
    based on sinusoidal curve to model diurnal temperature cycle.
"""
function fraction_between_thresholds(Tmax, Tmin, Threshold_lower, Threshold_upper)
    # Fraction between = (fraction above lower) - (fraction above upper)
    lower = fraction_above_threshold(Tmax, Tmin, Threshold_lower)
    upper = fraction_above_threshold(Tmax, Tmin, Threshold_upper)
    return lower - upper
end
```

convert_temp_threshold

Return temperature threshold after converting to -1,1 interval (for sine function)

```
"""Return temperature threshold after converting to -1,1 interval (for sine
function)"""
function convert_temp_threshold(Tmax, Tmin, Thresh)
    proportion = (Thresh - Tmin) / (Tmax - Tmin)
    return (2 * proportion - 1)
end
```

add_cubic_spline_vars

Add restricted cubic spline basis variables to dataframe df. Using continuous variable varname from dataframe and list of knots. After trying both the stata and Harrell formulas, neither matched the example file given. After comparing the formulas, I recognized the only difference was a scaling factor of `julia 1/(knots[last] - knots[first])^2` so I added that scaling factor to the Harrell formula and it produces results that match the example file. So I probably just messed up the stata formula somehow. I assume the example file was created in stata with mksplines.

```
"""Add restricted cubic spline basis variables to dataframe df.
    Using continuous variable varname from dataframe and list of knots.
    After trying both the stata and Harrell formulas, neither matched the example file given. After comparing the formulas, I recognized the only difference was a scaling factor of
        `julia 1/(knots[last] - knots[first])^2`
    so I added that scaling factor to the Harrell formula and it produces results that match the example file. So I probably just messed up the stata formula somehow. I assume the example file was created in stata with mksplines.
"""
function add_cubic_spline_vars(df, varname, knots; newbasename="splineC")
    for i in 1:(length(knots)-1)
        newname = "$newbasename$i"
        println("Making $newname")
        df[!, newname] = cubic_spline_var_harrell(df[!, varname], i, knots)
    end
    return df
end
```

cubic_spline_var_harrell

Return jth restricted cubic spline variable using the list of knots t. From Frank Harrell's Stats Textbook: Regression Modeling Strategies, 2001, sect. 2.4.4

```
"""Return jth restricted cubic spline variable using the list of knots t.
    From Frank Harrell's Stats Textbook: Regression Modeling Strategies, 2001, sect.
2.4.4
"""
function cubic_spline_var_harrell(x, j, t)
    k = length(t)
    if (j < 1) | (j > k - 1)
        println("Out of Bounds (j=$j): j is < 1 or > k-1. Not a valid variable.")
    elseif j == 1
        return x
    else
        xⱼ = m.(x .- t[j-1]) .^ 3 .-
             m.(x .- t[k-1]) .^ 3 * (t[k] - t[j-1]) / (t[k] - t[k-1]) +
             m.(x .- t[k]) .^ 3 * (t[k-1] - t[j-1]) / (t[k] - t[k-1])
        # Divide by the scale factor from the stata formula (range of the knots)
        return xⱼ / (t[k] - t[1])^2
    end
end
```

cubic_spline_var_stata

Return ith restricted cubic spline variable using the list of knots k. From stata's mkspline documentation

```
"""Return ith restricted cubic spline variable using the list of knots k.
    From stata's mkspline documentation
"""
function cubic_spline_var_stata(V, i, k)
    n = length(k)
    if (i < 1) | (i > n - 1)
        println("Out of Bounds (i=$i, n=$n): i < 1 or i > n-1. Not a valid variable.")
    elseif i == 1
        return V
    else
        Vᵢ = (m.(V .- k[i-1]) .^ 3 .- (k[n] - k[n-1])^(-1) *
                                       (m.(V .- k[n-1]) .^ 3 * (k[n] - k[i-1]) .- (V
.- k[n]) .^ 3 * (k[n-1] - k[i-1]))) ./
             (k[n] - k[1])^2
        return Vᵢ
    end
end
```

min_vecs

Return vector of element-wise minimum between a vector and constant.

```
"""Return vector of element-wise minimum between a vector and constant."""
min_vecs(vec, c) = minimum.(zip(vec, repeat([c], length(vec))))
```

max_vecs

Return vector of element-wise maximum between a vector and constant.

```
"""Return vector of element-wise maximum between a vector and constant."""
max_vecs(vec, c) = maximum.(zip(vec, repeat([c], length(vec))))
```

m

$(\cdot)_+$ function from Harrell 2001

```
"""(•)₊ function from Harrell 2001"""
m(a) = maximum([0, a])
```

add_piecewise_linear_vars

Return dataframe with piecewise linear basis variable columns using given knots.

```
"""Return dataframe with piecewise linear basis variable columns using given knots."""
function add_piecewise_linear_vars(df, varname, knots; newbasename="piece")
    println("Adding piecewise linear basis variable columns with breakpoints.")
    n = length(knots)+1
    for i in 1:n
        e = i == 1 ? "B$(knots[i])" :
            i == n ? "A$(knots[n-1])" :
            #= o.w =# "$(knots[i-1])to$(knots[i])"
        newname = "$newbasename$e"
        println("Making $newname")
        df[!, newname] = piecewise_linear_var_stata(df[!, varname], i, knots)
    end
    return df
end
```

piecewise_linear_var_stata

Return ith linear piecewise variable using the list of knots k. From stata's mkspline documentation

```
"""Return ith linear piecewise variable using the list of knots k.
    From stata's mkspline documentation
"""
function piecewise_linear_var_stata(V, i, k)
    n = length(k) + 1
    if i == 1
        return min_vecs(V, k[i])
    elseif i > 1 && i < n
        return max_vecs(min_vecs(V, k[i]), k[i-1]) .- k[i-1]
    elseif i == n
        return max_vecs(V, k[n-1]) .- k[n-1]
    else
        println("Out of Bounds (i=$i, n=$n): i < 1 or i > n-1. Not a valid variable.")
    end
end
```

aggregate_df

Return grid-day data aggregated to county-year level

```
"""Return grid-day data aggregated to county-year level"""
function aggregate_df(df, tags)
    # List of columns to average over
    avg_list = [:tMin, :tMax, :tAvg, :prec]
    # List of columns to sum over the year (all cols with a substring from tags)
    sum_list = [n for n in names(df) if any(occursin.(tags, n))]
    # sum_list = [:splineC1, :splineC2, :splineC3, :splineC4]

    # Sum over all days in each year, for each grid point
    df[!, "date"] = Date(1960, 1, 1) + Day.(df.dateNum)
    df[!, "year"] = Year.(df.date)
    gd_gy = groupby(df, [:gridNumber, :year])
    df_gy = combine(gd_gy,
                    avg_list .=> mean .=> avg_list,
                    sum_list .=> sum .=> sum_list)
    # Take the average over all grid points in the county
    gd_y = groupby(df_gy, :year)
    df_y = combine(gd_y, valuecols(gd_y) .=> mean .=> valuecols(gd_y))

    return df_y
end
```

Analysis Functions

1.2.2 Functions

plotdf_init

Return DF of avg temp, cubic spline, and predicted outcome variables needed for regressions and plotting.

```
  @param: reg_output: output from a regression on 4 cubic spline variables

  - create an x-axis for plotting: define an array of avg temps from 0 to 40°C,
   0.25 steps
  - define the spline knots/breakpoints
  - Create spline basis variables. After regressing, the absolute level of the o
  utcome (y-axis)
      is not identified because the county & year fixed effects shift the interc
  ept around.
      So these basis will be compared to the spline basis variables created for
   X°C.
  - Create spline basis variables for X°C
  - Create relative basis variables (base - X°C variables) which is equivalent t
  o shifting
      the base prediction by the X°C prediction, but allows for adjusted standar
  d errors
      in the delta method calculation.
```

```
"""Return DF of avg temp, cubic spline, and predicted outcome variables needed for regressions and plotting.

    @param: reg_output: output from a regression on 4 cubic spline variables

    - create an x-axis for plotting: define an array of avg temps from 0 to 40°C, 0.25 steps
    - define the spline knots/breakpoints
    - Create spline basis variables. After regressing, the absolute level of the outcome (y-axis)
        is not identified because the county & year fixed effects shift the intercept around.
        So these basis will be compared to the spline basis variables created for X°C.
    - Create spline basis variables for X°C
    - Create relative basis variables (base - X°C variables) which is equivalent to shifting
        the base prediction by the X°C prediction, but allows for adjusted standard errors
        in the delta method calculation.
"""
function plotdf_init(reg_output; X=20)
    df = DataFrame(tAvg=0:0.25:40)      # Initialize x-axis
    knots = [0 8 16 24 32]              # breakpoints in spline
    # Create base basis spline variables
    df = add_cubic_spline_vars(df, :tAvg, knots, newbasename="splinebaseC")
    # Create X°C basis spline variables
    df[!, "XC"] = repeat([X], nrow(df))
    df = add_cubic_spline_vars(df, "XC", knots, newbasename="XC")
    # Create relative basis spline variables (relative to X°C)
    for k in 1:4
        df[!, "splinerelativeC$k"] = df[!, "splinebaseC$k"] - df[!, "XC$k"]
```

```
    end
    # Construct the predicted spline outcome
    df[!, :yhat] = Matrix(df[!,r"splinerelativeC"]) * reg_output.coef
    # Return the x-axis, predicted outcome, splinerelCnd relative spline variables
(for delta method SE estimation)
    return df[!, [["tAvg", "yhat"]; ["splinerelativeC$k" for k ∈ 1:4]]]
end
```

get_diagonal

Return vector of diagonal elements of matrix

```
"""Return vector of diagonal elements of matrix"""
function get_diagonal(M::Matrix)
    n,_ = size(M)
    return [M[i,i] for i∈1:n]
end
```

deltamethod_CIbounds!

Return DF with upper and lower bounds created from delta method.

```
  Delta method: SE =  √(∂f(β)/∂β • VCOV • ∂f(β)/∂β)
  Calculate the β-derivative of the function of parameters
  f(β) = Σxₖ•βₖ ⇒ ∂f∂βₖ = xₖ
  ∂fβ_∂β = Matrix(df_plot[!,r"splineC"])
  SEs are the diagonal of .√(∂fβ_∂β * VCOV * ∂fβ_∂β'))
  where VCOV is the variance-covariance matrix of parameter estimates
  from the regression results
```

```
"""Return DF with upper and lower bounds created from delta method.

    Delta method: SE =  √(∂f(β)/∂β • VCOV • ∂f(β)/∂β)
    Calculate the β-derivative of the function of parameters
    f(β) = Σxₖ•βₖ ⇒ ∂f∂βₖ = xₖ
    ∂fβ_∂β = Matrix(df_plot[!,r"splineC"])
    SEs are the diagonal of .√(∂fβ_∂β * VCOV * ∂fβ_∂β'))
    where VCOV is the variance-covariance matrix of parameter estimates
    from the regression results
"""
function deltamethod_CIbounds!(df, reg_output; column_stem="splinerelativeC", α=0.05)
    VCOV = reg_output.vcov
    ∂fβ_∂β = Matrix(df[!, Regex(column_stem)])
    n = reg_output.nobs
    df[!, :SE] = .√(get_diagonal(∂fβ_∂β * VCOV * ∂fβ_∂β'))
    # Use a t-distribution just in case the sample is small
    tcrit = quantile(TDist(reg_output.dof_residual), 1-α/2)
    # Use √(1 + 1/n) correction because both the mean (predicted value) and the
variance are unknown estimated parameters
    width = tcrit * df[!, :SE] * √(1 + 1/n)
    df[!, :y_lb] = df[!, :yhat] - width
    df[!, :y_ub] = df[!, :yhat] + width
    #! If the sample size is small, consider using the second-order delta method
    return df
end
```

prediction_sample

Given a dataframe sample and dataframe with tAvg values to predict on, estimate the predicted values vector (for bootstrapping).

```
# Let's try bootstrapping!
"""Given a dataframe sample and dataframe with tAvg values to predict on,
    estimate the predicted values vector (for bootstrapping)."""
function prediction_sample(df, df_plot)
    reg_ = reg(df, formula122, Vcov.cluster(:fips))
    df2 = DataFrame(tAvg=0:0.25:40)
    df2[!, :yhat] = Matrix(df_plot[!,r"splinerelativeC"]) * reg_.coef
    return df2[!, [:tAvg, :yhat]]
end
```

confidence_interval

Return DF of confidence intervals for each temperature for bootstraped simulation results

```
"""Return DF of confidence intervals for each temperature for bootstraped simulation
results"""
function confidence_interval(df, α)
    df_bs = @chain df begin
        groupby(:tAvg)
        combine(:yhat => mean => :y_mean,
                :yhat => (x -> quantile!(x, α/2)) => :y_lb,
                :yhat => (x -> quantile!(x, 1-α/2)) => :y_ub,
                :yhat => minimum => :y_min, :yhat => maximum => :y_max)
    end
    return df_bs
end
```

bootstrap_predicted_outcome

Return dataframe of bootstrapped predictions with α confidence intervals.

```
"""Return dataframe of bootstrapped predictions with α confidence intervals."""
function bootstrap_predicted_outcome(df::DataFrame,
                                     prediction_function::Function;
                                     screenwidth=50,
                                     nsimulations=10000,
                                     α=0.05
                                     )
    println("\nStarting simple bootstrap")
    cols = [:year, :fips, :tAvg, :inc_farm_prop_inc_lpc, :splineC1, :splineC2,
:splineC3, :splineC4]
    preds = []
    t = @timed for k ∈ 1:nsimulations
        k%ceil(nsimulations/screenwidth) == 0 ? print("•") : nothing
        sample_rows = sample(1:nrow(df), nrow(df), replace=true)
        df_sample = df[sample_rows, cols]
        append!(preds, [prediction_function(df_sample, df_plot)])
    end
    # Vertically concatenate the bootstrap predictions
    preds_comb = reduce(vcat, preds)
    # Get mean and 95% CI for each value of tAvg
    df_bs = confidence_interval(preds_comb, α)
    print("$(t.time) seconds\n")
    return df_bs
end
```

bootstrap_predicted_outcome

Return dataframe of bootstrapped predictions with α confidence intervals.

Return dataframe of cluster-bootstrapped predictions with α confidence intervals.

```
  Clustered Bootstrap (over each fips county code)
  Need to sample 3055 counties of 3055 counties (with replacement)
  Requires repeating some counties in the data and keeping all years
  of their data
```

```
"""Return dataframe of cluster-bootstrapped predictions with α confidence intervals.

    Clustered Bootstrap (over each fips county code)
    Need to sample 3055 counties of 3055 counties (with replacement)
    Requires repeating some counties in the data and keeping all years
    of their data
"""
function bootstrap_predicted_outcome(df::DataFrame,
                                     prediction_function::Function,
                                     cluster::Union{Symbol, String};
                                     screenwidth=50,
                                     nsimulations=10000,
                                     α=0.05
                                     )
    println("\nStarting clustered bootstrap")
    cols = [:year, :fips, :tAvg, :inc_farm_prop_inc_lpc, :splineC1, :splineC2,
:splineC3, :splineC4]
    preds = []
    df_grouped = groupby(df[!, cols], cluster)
    ngroups = length(df_grouped)
    t = @timed for k ∈ 1:nsimulations
        k%ceil(nsimulations/screenwidth) == 0 ? print("•") : nothing
        sample_idx = sample(1:ngroups, ngroups; replace = true, ordered = false)
        df_sample = reduce(vcat, [df_grouped[i] for i in sample_idx])
        append!(preds, [prediction_function(df_sample, df_plot)])
    end
    # Vertically concatenate the bootstrap predictions
    preds_comb = reduce(vcat, preds)
    # Get mean and 95% CI for each value of tAvg
    df_bs = confidence_interval(preds_comb, α)
    print("$(t.time) seconds\n")
    return df_bs
end
```

run_bootstrap

Bootstrapping confidence intervals for Problem 1

```
begin
"""Bootstrapping confidence intervals for Problem 1"""
function run_bootstrap(df, df_plot, prediction_sample)
    # Get mean and 95% CI for each value of tAvg using bootstraps
    nsimulations = 100
    # Simple bootstrap (over all individuals)
    df_bs = bootstrap_predicted_outcome(df, prediction_sample,
    nsimulations=nsimulations)
    # Clustered bootstrap
    df_bs_cluster = bootstrap_predicted_outcome(df, prediction_sample, :fips;
    nsimulations=nsimulations)


    # Plot Delta method and bootstraps confidence intervals and predicted outcome
    s = 0; dpi=400;
    @df df_plot plot(:tAvg, :y_lb, w=0, msw = 0, ms = s, c=1,
        fillrange=:y_ub, fillalpha=0.35,
        label="95% CI Delta Method")
    @df df_bs plot!(:tAvg, :y_lb, w=0, msw = 0, ms = s, c=2,
        fillrange=:y_ub, fillalpha=0.35,
        label="95% CI BootStrap ($nsimulations)")
    @df df_bs_cluster plot!(:tAvg, :y_lb, w=0, msw = 0, ms = s, c=3,
        fillrange=:y_ub, fillalpha=0.35,
        label="95% CI BootStrap-clustered ($nsimulations, FIPS)")
    p8 = @df df_plot plot!(:tAvg, :yhat, label="Predicted Response",
    legend=:bottomleft)
    savefig(p8, "plot122-delta-boot-bootcluster.svg")
end
end
```

1.2.3 Functions

add_moving_avgs!

Return dataframe with backward-looking `windowsize` moving averages of all columns

```
"""Return dataframe with backward-looking `windowsize` moving averages of all columns"""
function add_moving_avgs!(df; windowsize=10)
    # group each county
    cols = [["fips", "Date"]; names(df, r"temp[^P]")]
    gdf = groupby(df[!, cols], :fips)
    dfs = []
    # for each county, calculate a 10-year backward-looking moving average
    for group in gdf
        # Create TimeArray
        temp = TimeArray(group, timestamp = :Date)
        # create moving average of last 10 years
        temp = moving(mean, temp, windowsize)
        # convert back to dataframe
        temp = DataFrame(temp)
        append!(dfs, [temp])
    end
    # concatenate county dataframes and leftjoin onto df using fips and Date
    df_avg = reduce(vcat, dfs)
    renamingdict = [names(df, r"temp[^P]") .=> names(df, r"temp[^P]") .* "_avg"; ["timestamp" => "Date"]]
    df_avg = DataFrames.rename(df_avg, renamingdict)
    df = leftjoin(df, df_avg, on=[:fips, :Date])
    return df
end
```

reg123

Regress the outcomes on the deviance from the running average bins

```
"""Regress the outcomes on the deviance from the running average bins"""
function reg123(df, y; cluster=:fips, regex=r"temp(?!20to24).*_dev")
    formula_ = term(y) ~ sum([term.(names(df, regex)); [term(fe(:fips)), term(fe(:year))]])
    reg_ = @time reg(df, formula_, Vcov.cluster(cluster))
    se = .√[reg_.vcov[i,i] for i in 1:length(reg_.coef)]
    ub = reg_.coef + 1.96*se
    lb = reg_.coef - 1.96*se
    df2 = DataFrame(tAvg=[-2,2,6,10,14,18,26,30,34],
                    yhat=reg_.coef,
                    yerror=1.96*se,
                    y_ub=ub,
                    y_lb=lb)
    df2 = reduce(vcat, [[df2]; [DataFrame(tAvg=22,yhat=0,yerror=0,y_lb=0,y_ub=0)]])
    df2 = sort!(df2, :tAvg)
    return df2
end
```

plot123

Plot deviance bins: above avg # of days this year that were in this bin (days above the 10-year average # of days)

```
""" Plot deviance bins: above avg # of days this year that were in this bin (days above the 10-year average # of days)"""
function plot123(y, df)
    labels = Dict(:emp_farm_ln => "Farm Employment", :inc_farm_prop_inc_lpc => "Farm Income")
    ylabels = Dict(:emp_farm_ln => "log(Farm Employment)", :inc_farm_prop_inc_lpc => "log(Farm Income per capita)")
    temp = reg123(df, y)
    plot(temp[!,:tAvg], repeat([0], nrow(temp)), c="gray", s=:dash, label="")
    @df temp plot!(:tAvg, :y_lb, w=0, msw = 0, ms = 0, c=1,
    fillrange=:y_ub, fillalpha=0.35,
    label="95% CI")
    xlabel!("Average Temperature Bins")
    ylabel!(ylabels[y])
    title!("Deviance from 10-yr Avg Binned Days: $(labels[y])          ")
    @df temp plot!(:tAvg, :yhat,
        label="Predicted Response",
        legend=:bottomright, c=2, shape=:circle, markerstrokewidth=0)
end
```

plot123b

Plot basic bins: # of days in a county-year that day's avg temp was in this bin

```
"""Plot basic bins: # of days in a county-year that day's avg temp was in this bin"""
function plot123b(y, df)
    labels = Dict(:emp_farm_ln => "Farm Employment", :inc_farm_prop_inc_lpc => "Farm Income")
    ylabels = Dict(:emp_farm_ln => "log(Farm Employment)", :inc_farm_prop_inc_lpc => "log(Farm Income per capita)")
    temp = reg123(df, y; regex=r"^temp(?!20to24)[^P].*[^vg]$")
    plot(temp[!,:tAvg], repeat([0], nrow(temp)), c="gray", s=:dash, label="")
    @df temp plot!(:tAvg, :y_lb, w=0, msw = 0, ms = 0, c=1,
        fillrange=:y_ub, fillalpha=0.35,
        label="95% CI")
    xlabel!("Average Temperature Bins")
    ylabel!(ylabels[y])
    title!("Binned Temperature Response: $(labels[y])          ")
    @df temp plot!(:tAvg, :yhat,
        label="Predicted Response",
        legend=:bottomright, c=2, shape=:circle, markerstrokewidth=0)
end
```

plot124

Plot histogram of >32°C bin.

```
"""Plot histogram of >32°C bin."""
function plot124(y, df)
    labels = Dict(:emp_farm_ln => "Farm Employment", :inc_farm_prop_inc_lpc => "Farm
    Income")
    ylabels = Dict(:emp_farm_ln => "log(Farm Employment)", :inc_farm_prop_inc_lpc =>
    "log(Farm Income per capita)")
    temp = reg123(df, y; regex=r"^temp(?!20to24)[^P].*[^vg]$")
    plot(temp[!,:tAvg], repeat([0], nrow(temp)), c="gray", s=:dash, label="")
    @df temp plot!(:tAvg, :y_lb, w=0, msw = 0, ms = 0, c=1,
    fillrange=:y_ub, fillalpha=0.35,
    label="95% CI")
    xlabel!("Average Temperature Bins")
    ylabel!(ylabels[y])
    title!("Binned Temperature Response: $(labels[y])          ")
    @df temp plot!(:tAvg, :yhat,
        label="Predicted Response",
        legend=:bottomright, c=2, shape=:circle, markerstrokewidth=0)
end
```

Problem 2 Functions (in order of the problems)

left_right_plot

2.2.5: Plot separate kernel regressions on either side of the 75-unit threshold.

```
"""2.2.5: Plot separate kernel regressions on either side of the 75-unit threshold."""
function left_right_plot(df_in, xcol, ycol; bandwidth, plotwidth, threshold=75, npoints=200)
    # Plot all observations as circles
    df = dropmissing(df_in, [xcol, ycol])
    ylabels = Dict(:dlhouse => "House Price", :dgtsp => "Pollution")
    plot(df[!, xcol], df[!, ycol],
        seriestype=:scatter,
        markeralpha=0.3,
        label="observations",
        xlabel="1974 TSPs levels",
        ylabel="$(ylabels[ycol]) changes",
        title="LOESS regressions: $(ylabels[ycol]) Changes",
        xlims = (75-plotwidth, 75+plotwidth), #75±width
        ylims = extrema(df[!, ycol]),
        legend=:bottomleft
    )
    thickness = 3
    vline!([threshold], label="$xcol threshold", w=thickness)
    # Plot Loess below threshold
    df1 = subset(df, xcol => ByRow(<(threshold)))
    xs1, ys1 = get_loess_prediction(df1, xcol, ycol; bandwidth=bandwidth, npoints=npoints)
    plot!(xs1, ys1, label="loess regression (below)", w=thickness)
    # Plot Loess above threshold
    df2 = subset(df, xcol => ByRow(≥(threshold)))
    xs2, ys2 = get_loess_prediction(df2, xcol, ycol; bandwidth=bandwidth, npoints=npoints)
    plot!(xs2, ys2, label="loess regression (above)", w=thickness)
    # Plot dashed lines and text of height difference
    plot!([first(xs1), 75], [last(ys1), last(ys1)], lw=2, lc=:gray, legend=false, style=:dash)
    p = plot!([75, last(xs2)], [first(ys2), first(ys2)], lw=2, lc=:gray, legend=false, style=:dash)
    xpos = (xlims(p)[2] - xlims(p)[1])*0.95 + xlims(p)[1]
    ypos = (ylims(p)[2] - ylims(p)[1])*0.1 + ylims(p)[1]
    annotate!(xpos, ypos,
        ("vertical difference = $(round(first(ys2) - last(ys1), digits=4))",
        :right, 10))
end
```

controls_smoothed_plot

2.2.6: Plot unconditional kernal regression and control-conditioned-prediction kernal regression on either side of the 75-unit threshold.

```
"""2.2.6: Plot unconditional kernal regression and control-conditioned-prediction kernal regression on either side of the 75-unit threshold."""
function controls_smoothed_plot(df, xcol, ycol1, ycol2; bandwidth, plotwidth, threshold=75, npoints=200)
    thickness = 3
    alpha = 0.4
    c3 = 3
    # Plot all observations as circles
    ylabels = Dict(:dlhouse => "House Price", :dgtsp => "Pollution")
    plot(df[!, xcol], df[!, ycol1], seriestype=:scatter, markeralpha=alpha, label="observations", c=1)
    plot!(df[!, xcol], df[!, ycol2], seriestype=:scatter, label="controls' prediction", c=c3, shape=:+, markerstrokewidth=5, markersize=5)
    # Plot Loess below threshold
    df1 = subset(df, xcol => ByRow(<(threshold)))
    x1, y1 = get_loess_prediction(df1, xcol, ycol1; bandwidth=bandwidth, npoints=npoints)
    x2, y2 = get_loess_prediction(df1, xcol, ycol2; bandwidth=bandwidth, npoints=npoints)
    plot!(x1, y1,
        label="raw obseservations kernel",
        xlabel="1974 TSPs levels",
        ylabel="$(ylabels[ycol1]) changes",
        title="LOESS regressions: $(ylabels[ycol1]) Changes",
        xlims = (75-plotwidth, 75+plotwidth), #75±width
        ylims = extrema([df[!, ycol1]; df[!, ycol2]]),
        legend=:topleft,
        w=thickness, c=1)
    plot!(x2, y2, label="controls' predictions kernel", w=thickness, c=c3)
    # Plot Loess above threshold
    df2 = subset(df, xcol => ByRow(≥(threshold)))
    x3, y3 = get_loess_prediction(df2, xcol, ycol1; bandwidth=bandwidth, npoints=npoints)
    x4, y4 = get_loess_prediction(df2, xcol, ycol2; bandwidth=bandwidth, npoints=npoints)
    plot!(x3, y3, label="", w=thickness, c=1)
    plot!(x4, y4, label="", w=thickness, c=c3)
    # Vertical line at the threshold
    p = vline!([threshold], label="$xcol threshold", w=thickness, c=2)
    # Plot dashed lines and text of height difference
    # 1,2 = left raw, predicted    3,4 = right raw, predicted
    # plot!([first(xs1), 75], [last(ys1), last(ys1)], lw=2, lc=:gray, legend=false, style=:dash)
    # p = plot!([75, last(xs2)], [first(ys2), first(ys2)], lw=2, lc=:gray, legend=false, style=:dash)
    xpos = (xlims(p)[2] - xlims(p)[1])*0.95 + xlims(p)[1]
    ypos1 = (ylims(p)[2] - ylims(p)[1])*0.2 + ylims(p)[1]
    ypos2 = (ylims(p)[2] - ylims(p)[1])*0.14 + ylims(p)[1]
```

```
    ypos3 = (ylims(p)[2] - ylims(p)[1])*0.08 + ylims(p)[1]
    annotate!(xpos, ypos1,
        ("vertical difference of observation kernel = $(round(first(y3) - last(y1),
        digits=3))",
        :right, 10))
    annotate!(xpos, ypos2,
        ("vertical difference of control-prediction kernel= $(round(first(y4) -
        last(y2), digits=3))",
        :right, 10))
    annotate!(xpos, ypos3,
        ("difference = $(round((first(y3) - last(y1)) - (first(y4) - last(y2)),
        digits=3))",
        :right, 10))
end
```

```
"""2.2.7: Plot Regulated vs Unregulated kernel regressions for 50-75 1974 TSP."""
function un_regulated_plot(df, xcol, ycol; bandwidth, npoints=2000)
    thickness = 3
    alpha = 0
    c3=3
    # Plot all observations as circles
    df1 = @subset(df, :tsp7576 .== 0)
    df2 = @subset(df, :tsp7576 .== 1)
    ylabels = Dict(:dlhouse => "House Price", :dgtsp => "Pollution")
    plot(df1[!, xcol], df1[!, ycol],
        xlabel="1974 TSPs levels",
        ylabel="$(ylabels[ycol]) changes",
        title="LOESS regressions: $(ylabels[ycol]) Changes",
        xlims = (50, 75),
        ylims = extrema(df[!, ycol]),
        seriestype=:scatter,
        markeralpha=alpha,
        label="",
        c=1)
    plot!(df2[!, xcol], df2[!, ycol], seriestype=:scatter, alpha=0,
        label="", c=c3, shape=:+, markerstrokewidth=5, markersize=5)
    # Plot unregulated
    x1, y1 = get_loess_prediction(df1, xcol, ycol; bandwidth=bandwidth,
    npoints=npoints)
    plot!(x1, y1,
        label="Unregulated Kernel Regression",
        legend=:bottomright,
        w=thickness, c=1)
    # Plot regulated
    x2, y2 = get_loess_prediction(df2, xcol, ycol; bandwidth=bandwidth,
    npoints=npoints)
    plot!(x2, y2, label="Regulated Kernel Regression", w=thickness, c=c3)
    ymax = maximum([y1; y2]) + (maximum([y1; y2]) - minimum([y1; y2]))*0.2
    ymin = minimum([y1; y2]) - (maximum([y1; y2]) - minimum([y1; y2]))*0.6
    p = ylims!(ymin, ymax)
    # Plot averages
    xpos = (xlims(p)[2] - xlims(p)[1])*0.02 + xlims(p)[1]
    ypos1 = (ylims(p)[2] - ylims(p)[1])*0.14 + ylims(p)[1]
    annotate!(xpos, ypos1,
        ("difference in means = $(round(mean(df2[!, ycol]) - mean(df1[!, ycol]),
        digits=2))",
        10, :left))
    hline!([mean(df1[!, ycol])]; label="Unregulated Mean of observations", c=1, lw=1,
    style=:dashdot)
    hline!([mean(df2[!, ycol])]; label="Regulated Mean of observations", c=c3, lw=2,
    style=:dash)
    # Plot last values
    plot!([percentile_range(xlims(p), 0.7), xlims(p)[2]], [last(y1), last(y1)],
        c=1, lw=1, style=:solid, label="Unregulated final kernel value")
    plot!([percentile_range(xlims(p), 0.7), xlims(p)[2]], [last(y2), last(y2)],
        c=c3, lw=2, style=:dot, label="Regulated final kernel value")
    ypos2 = (ylims(p)[2] - ylims(p)[1])*0.08 + ylims(p)[1]
    annotate!(xpos, ypos2,
        ("difference in final values = $(round(last(y2) - last(y1), digits=2))",
```

```
        10, :left))
end;
```

```
percentile_range (generic function with 1 method)
```

```
function percentile_range(vec, percentile)
    min, max = extrema(vec)
    return min + percentile*(max-min)
end
```

Notebook Settings

Packages

File Name variables and notebook printing macros

Table of contents sidebar