```
* Regression discontinuity data task
* There are 2 ways to run this. One is described in lines 5-10, another one in lines 11-13
* The file contains a lot of line breaks (///), so do NOT run it through the Command window
* Use .do file instead
* This .do file should be put in the folder with the data files
* Run the following 2 lines ALONE; a pop-up will appear, then browse this .do file.
ssc install project, replace
project, setup
* Line to set the current directory to the one with the .do file and data files
project OlegTeleginDiscontinuityCodeSample, cd
* Alternatively, you can ignore lines 4-9 and just change line 12 manually to
* set cd to the directory containing data files, and run this line alone
cd "C:\Users\Username\Downloads\Data Task Files"
* Lines 14-287 can be run altogether.
* Clear memory, install and set the scheme for the graphs, and install
* the command for the binned scatter plots and RD
clear
net install scheme-modern, from("https://raw.githubusercontent.com/mdroste/stata-scheme-modern/master/")
set scheme modern, perm
ssc install binscatter, replace
net install rdrobust, from(https://raw.githubusercontent.com/rdpackages/rdrobust/master/stata) replace
ssc install estout, replace
set autotabgraphs on
* Import .txt file as a string and delimit it into a table
infix str Imptxt 1-16 using Revenue.txt
gen logrev = real(substr(Imptxt, 1, 10))
gen rest_id = real(substr(Imptxt, 11, 4))
gen time = real(substr(Imptxt, 15, 2))
drop Imptxt
frame create additional
gen Dopvar1=.
gen Dopvar2=.
* Import .csv files into the additional frame, delimit and merge them with our data
forvalues i = 1(1)10 {
    frame change additional
    if `i' == 10 {
        local filename "month`i'.csv"
    }
    else {
        local filename "month0`i'.csv"
    }
    import delimited "`filename'", varnames(1) clear
    save temp_data, replace
    frame change default
    merge 1:1 rest_id time using temp_data
    replace Dopvar1 = stars if _merge == 3
    replace Dopvar2 = score if _merge == 3
    drop _merge stars score
}
rename Dopvar1 stars
rename Dopvar2 score
sort rest_id time
gen ndate = _n
* Delete an additional frame
frame drop additional
* Calculate fractional parts of scores and scores around the threshold
gen frac_score = score - trunc(score)
gen half_frac_score = (2*frac_score-trunc(2*frac_score))/2
* For each number of displayed stars, tabulate the number of restaurants,
* the mean, s.d., min, and max revenue. Store to .tex
estpost tabstat logrev , by(stars) statistics(count mean sd min max)
esttab using "table1.tex", replace ///
cells("count mean(fmt(%12.2f)) sd(fmt(%12.2f)) min(fmt(%12.2f)) max(fmt(%12.2f))") ///
nonumber nomtitle nonote noobs label booktabs ///
collabels("N" "Mean" "SD" "Min" "Max") varlabels(`e(labels)')
*Check if any of the stars are miscalculated
gen mistake = stars - round(2*stars)/2
summarize mistake
* Plot cumulative distribution function for logrev
cumul logrev, gen(cumul_logrev)
sort cumul_logrev
line cumul_logrev logrev, scale(1.5) name(a)
sort rest_id time
* Calculate massive consecutive score changes in the opposite directions and create an index
* for the restaurants with huge fluctuations (likely to have a small number of reviews)
gen diffscore = score-score[_n-1]
gen incons_score = 0
replace incons_score = 1 if diffscore > 1.5 & diffscore[_n+1] < -1.5
replace incons_score = 1 if diffscore < -1.5 & diffscore[_n+1] > 1.5
replace incons_score = . if time == 1 | time == 10
egen no_reviews_index = max(incons_score), by(rest_id)
drop diffscore incons_score
summarize no_reviews_index
* Find restaurants with potentially deleted reviews and create an index for the obs to ignore
gen wrong_scores_index = 0
quietly summarize ndate
forvalues i = 1(1)`r(max)' {
    if score[`i']==0 {
        local m = `i'-time[`i']+1
        quietly summarize score in `m'/`i'
        local m1 = r(sum)
        quietly replace wrong_scores_index = `m1' in `m'/`i'
        quietly replace wrong_scores_index = 0 in `i'
    }
}
replace wrong_scores_index = 1 if wrong_scores_index!=0
summarize wrong_scores_index
* Calculate smoothed revenue if rev faces huge consecutive changes in different directions
gen rev = exp(logrev)
gen rev_change = rev[_n]/rev[_n-1]
replace rev_change = . if time == 1
gen rev_incr = 0
replace rev_incr = 1 if rev_change > 10
replace rev_incr = . if time == 1
gen rev_drop = 0
replace rev_drop = 1 if rev_change < 0.1
replace rev_drop = . if time == 1
gen rev_big_chan = 0
replace rev_big_chan = 1 if rev_drop == 1 & rev_incr[_n+1] == 1
replace rev_big_chan = 1 if rev_incr == 1 & rev_drop[_n+1] == 1
replace rev_big_chan = . if time == 1 | time == 10
egen rest_w_big_chan = max( rev_big_chan ), by( rest_id )
summarize rev_big_chan rest_w_big_chan
gen rev_smooth = rev
replace rev_smooth = (rev[_n-1]+rev+rev[_n+1])/3 if rest_w_big_chan == 1
gen logrev_smooth = log(rev_smooth)
gen rev_smooth_index = 1
replace rev_smooth_index = 0 if (time == 1 | time == 10) & rest_w_big_chan == 1
* Draw histograms of + and - spikes around the threshold compared with the whole sample
gen rev_big_chan_incr = 0
replace rev_big_chan_incr = 1 if rev_incr == 1 & rev_drop[_n+1] == 1
replace rev_big_chan_incr = . if time == 1 | time == 10
gen rev_big_chan_drop = 0
replace rev_big_chan_drop = 1 if rev_drop == 1 & rev_incr[_n+1] == 1
replace rev_big_chan_drop = . if time == 1 | time == 10
twoway (histogram half_frac_score if rev_big_chan_incr== 1 & half_frac_score >0.05 & ///
half_frac_score <0.45, width(0.1) color(sand%80)) (histogram half_frac_score if ///
half_frac_score >0.05 & half_frac_score < 0.45, width(0.1) color(edkblue%60)), ///
legend(on order(1 "Upward spikes  " 2 "Whole sample") row(1) position(7)) ///
xtitle("Score around the threshold") xlabel(0.05 "-0.2" 0.15 "-0.1" ///
0.25 "0" 0.35 "+0.1" 0.45 "+0.2") ylabel(0 1 2 3) ysc(r(0 3.5)) scale(1.5) ///
xline(0.25) fxsize(73) fysize(63) name(b)
twoway (histogram half_frac_score if rev_big_chan_drop == 1 & half_frac_score > 0.05 ///
& half_frac_score <0.45, width(0.1) color(sand%80)) (histogram half_frac_score if ///
half_frac_score >0.05 & half_frac_score <0.45, width(0.1) color(edkblue%60)), ///
legend(on order(1 "Downward spikes  " 2 "Whole sample") row(1) position(7)) ///
xtitle("Score around the threshold") xlabel(0.05 "-0.2" 0.15 "-0.1" 0.25 "0" ///
0.35 "+0.1" 0.45 "+0.2") ylabel(0 1 2 3) ysc(r(0 3.5)) scale(1.5) xline(0.25) ///
fxsize(73) fysize(63)  name(c)
graph combine c b, name(comb1)
drop rev_change rev_incr rev_drop
* Check if we still have a lot of restaurants with fluctuating revenue (then drop vars)
gen rev_change_smooth = rev_smooth[_n]/rev_smooth[_n-1]
replace rev_change_smooth = . if time == 1
gen rev_incr_smooth = 0
replace rev_incr_smooth = 1 if rev_change_smooth > 10
replace rev_incr_smooth = . if time == 1
gen rev_drop_smooth = 0
replace rev_drop_smooth = 1 if rev_change_smooth < 0.1
replace rev_drop_smooth = . if time == 1
gen rev_big_chan_smooth = 0
replace rev_big_chan_smooth = 1 if rev_drop_smooth == 1 & rev_incr_smooth[_n+1] == 1
replace rev_big_chan_smooth = 1 if rev_incr_smooth == 1 & rev_drop_smooth[_n+1] == 1
replace rev_big_chan_smooth = . if time == 1 | time == 10 | time == 2 | time == 9
egen rest_w_big_chan_smooth = max( rev_big_chan_smooth ), by( rest_id )
replace rest_w_big_chan_smooth = . if rev_smooth_index == 0
summarize rev_big_chan_smooth rest_w_big_chan_smooth
drop rev_change_smooth rev_incr_smooth rev_drop_smooth ///
rev_big_chan_smooth rest_w_big_chan_smooth
* Summary statistics for the cleaned data
summarize logrev if rev_smooth_index == 1 & no_reviews_index == 0 & wrong_scores_index == 0
tabstat logrev if rev_smooth_index == 1 & no_reviews_index == 0 & ///
wrong_scores_index == 0, by( stars ) statistics(count mean sd min max)
* Draw binned scatter plots
binscatter logrev score if no_reviews_index==0 & wrong_scores_index==0, ///
rd(0.25 0.75 1.25 1.75 2.25 2.75 3.25 3.75 4.25 4.75) line(none) xtitle("Score") ///
ytitle("Revenue, log($)") xlabel(0.25 "0.25" 0.75 "0.75" 1.25 1.75 2.25 2.75 ///
3.25 3.75 4.25 4.75, nogrid labsize(vsmall)) ylabel(, nogrid) xticks(1 2 3 4 5) ///
scale(1.5) fxsize(75) fysize(55) name(d)
binscatter logrev_smooth score if no_reviews_index==0 & wrong_scores_index==0 & ///
rev_smooth_index==1, rd(0.25 0.75 1.25 1.75 2.25 2.75 3.25 3.75 4.25 4.75) ///
line(none) xtitle("Score") ytitle("Smoothed Revenue, log($)") xlabel(0.25 ///
"0.25" 0.75 "0.75" 1.25 1.75 2.25 2.75 3.25 3.75 4.25 4.75, nogrid labsize(vsmall)) ///
ylabel(, nogrid) xticks(1 2 3 4 5) scale(1.5) fxsize(75) fysize(55) name(e)
graph combine d e, name(comb2)
* Draw histograms of scores and fractional parts of scores
xtile logrev_quar = logrev if no_reviews_index==0 & wrong_scores_index==0 & ///
rev_smooth_index==1, nq(4)
histogram score, width(0.125) xlabel(0 1 2 3 4 5, nogrid) xtitle("Score") ///
ylabel(, nogrid) scale(1.5) name(f)
histogram half_frac_score if half_frac_score>0.05 & half_frac_score<0.45 & ///
no_reviews_index==0 & wrong_scores_index==0 & rev_smooth_index==1, width(0.025) ///
xlabel(0.05 "-0.2" 0.15 "-0.1" 0.25 "0" 0.35 "+0.1" 0.45 "+0.2", nogrid) ///
ylabel(, nogrid) xline(0.25) xtitle("Score around the threshold") scale(1.5) ///
lwidth(vvthin) lcolor(black%70) name(g)
* Now for quartiles
histogram half_frac_score if half_frac_score>0.05 & half_frac_score<0.45 ///
& logrev_quar == 1 & no_reviews_index==0 & wrong_scores_index==0 & ///
rev_smooth_index==1, width(0.025) xlabel(0.05 "-0.2" 0.15 "-0.1" 0.25 ///
"0" 0.35 "+0.1" 0.45 "+0.2", nogrid) ylabel(, nogrid) xline(0.25) ///
xtitle("Score around the threshold") scale(1.5) name(h)
histogram half_frac_score if half_frac_score>0.05 & half_frac_score<0.45 ///
& logrev_quar == 2 & no_reviews_index==0 & wrong_scores_index==0 & ///
rev_smooth_index==1, width(0.025) xlabel(0.05 "-0.2" 0.15 "-0.1" 0.25 ///
"0" 0.35 "+0.1" 0.45 "+0.2", nogrid) ylabel(, nogrid) xline(0.25) ///
xtitle("Score around the threshold") scale(1.5) name(i)
histogram half_frac_score if half_frac_score>0.05 & half_frac_score<0.45 ///
& logrev_quar == 3 & no_reviews_index==0 & wrong_scores_index==0 & ///
rev_smooth_index==1, width(0.025) xlabel(0.05 "-0.2" 0.15 "-0.1" 0.25 "0" ///
0.35 "+0.1" 0.45 "+0.2", nogrid) ylabel(, nogrid) xline(0.25) ///
xtitle("Score around the threshold") scale(1.5) name(j)
histogram half_frac_score if half_frac_score>0.05 & half_frac_score<0.45 ///
& logrev_quar == 4 & no_reviews_index==0 & wrong_scores_index==0 & ///
rev_smooth_index==1, width(0.025) xlabel(0.05 "-0.2" 0.15 "-0.1" 0.25 ///
"0" 0.35 "+0.1" 0.45 "+0.2", nogrid) ylabel(, nogrid) xline(0.25) ///
xtitle("Score around the threshold") scale(1.5) name(k)
histogram logrev, name(l)
* Generate scores at T+1 and bins of equal width for scores and scores at T+1 (101)
gen score_change = score[_n+1] if time < 10
gen bin = trunc(20*score)+1
gen bin_new = trunc(20*score_change)+1
* New dummy equals 1 if the bin number increases at t+1, 0 if not, N/A if time = 10
gen bin_incr = 0
replace bin_incr = 1 if bin_new > bin & time < 10
replace bin_incr = . if time == 10
* Calculate 20 bins for int_scores (0.05 each) and create bins around the threshold
gen frac_bin = trunc(20*frac_score)+1
gen half_frac_bin = trunc(20*(frac_score - trunc(2*frac_score)/2))+1
* Calculate the increase frequency for each bin and store them as a variable
mean bin_incr if no_reviews_index==0 & wrong_scores_index==0 & rev_smooth_index==1, ///
over(half_frac_bin)
matrix meansdop = r(table)
local n = rowsof(meansdop)+1
gen increase_frequencydop = .
forvalues i = 1/`n' {
    replace increase_frequencydop = meansdop[1, `i'] in `i'
}
* Plot typical restaurant revenue over time
scatter rev time in 1/10, scale(1.5) xtitle("Time") ytitle("Revenue, $") name(m)
* Create a graph to visualize probabilites for different bins
quietly summarize bin_incr if no_reviews_index==0 & wrong_scores_index==0 & ///
rev_smooth_index==1 & half_frac_bin>1 & half_frac_bin<10
local mean04 = r(mean)
quietly summarize bin_incr if no_reviews_index==0 & wrong_scores_index==0 & ///
rev_smooth_index==1 & half_frac_bin>2 & half_frac_bin<9
local mean03 = r(mean)
quietly summarize bin_incr if no_reviews_index==0 & wrong_scores_index==0 & ///
rev_smooth_index==1 & half_frac_bin>3 & half_frac_bin<8
local mean02 = r(mean)
twoway bar increase_frequencydop ndate if (ndate >= 2 & ndate <= 4) | ///
(ndate >= 6 & ndate <= 9), barw(1) scale(1.4) xtitle("Score around the threshold", ///
size(small)) color(navy%60) ytitle("Prob. of getting to a higher bin at T+1", ///
size(small)) xlabel(1.5 "-0.2" 2.5 "-0.15" 3.5 "-0.1" 4.5 "-0.05" 5.5 "0" 6.5 ///
"+0.05" 7.5 "+0.1" 8.5 "+0.15" 9.5 "+0.2") ylabel(.46 "0.46" .48 "0.48" .5 "0.5" ///
.52 "0.52" .54 "0.54") xline(5.5) || bar increase_frequencydop ndate in 5, barw(1) ///
color(navy%100) || function y = `mean04', lpattern(dash) range(1.5 9.5) lcolor(sand) ///
|| function y = `mean03', lpattern(dash) range(1.5 9.5) lcolor(dkgreen) || ///
function y = `mean02', lpattern(dash) range(1.5 9.5) lcolor(cranberry) ///
legend(order(3 4 5) label(3 "Average probability for the bandwidth = 0.4") ///
label(4 "Average probability for the bandwidth = 0.3") ///
label(5 "Average probability for the bandwidth = 0.2") position(7)) ///
plotregion(margin(0)) name(n)
* Linear regressions and VIF checks
reg logrev score if no_reviews_index==0 & wrong_scores_index==0
reg logrev_smooth score if no_reviews_index==0 & wrong_scores_index==0 & rev_smooth_index==1
reg logrev score stars if no_reviews_index==0 & wrong_scores_index==0
estat vif
reg logrev_smooth score stars if no_reviews_index==0 & wrong_scores_index==0 & ///
rev_smooth_index==1
estat vif
* RD figures for smoothed and non-smoothed data
rdplot logrev_smooth half_frac_score if half_frac_score>0.05 & half_frac_score<0.45 ///
& no_reviews_index==0 & wrong_scores_index==0 & rev_smooth_index==1, c(0.25) ///
graph_options(xtitle("Score around the threshold") ytitle("Smoothed Revenue, log($)") ///
xlabel(0.05 "-0.2" 0.15 "-0.1" 0.25 "0" 0.35 "+0.1" 0.45 "+0.2") ///
xsc(r(0.05 0.45)) scale(1.3) name(o))
rdplot logrev half_frac_score if half_frac_score>0.05 & half_frac_score<0.45 & ///
no_reviews_index==0 & wrong_scores_index==0, c(0.25) ///
graph_options(xtitle("Score around the threshold") ytitle("Revenue, log($)") ///
xlabel(0.05 "-0.2" 0.15 "-0.1" 0.25 "0" 0.35 "+0.1" 0.45 "+0.2") ///
xsc(r(0.05 0.45)) scale(1.3) name(p))
* RD regressions
rdrobust logrev_smooth half_frac_score if half_frac_score>0.05 & half_frac_score<0.45 ///
& no_reviews_index==0 & wrong_scores_index==0 & rev_smooth_index==1, c(0.25)
rdrobust logrev half_frac_score if half_frac_score>0.05 & half_frac_score<0.45 ///
& no_reviews_index==0 & wrong_scores_index==0, c(0.25)
* After the end of the work, delete temporary data file, .tex tables, and graph names
* For Windows
erase temp_data.dta
erase table1.tex
graph drop a b c d e f g h i j k l m n o p comb1 comb2
* For Mac/Linux (one of two should display a mistake)
rm temp_data.dta
rm table1.tex
graph drop a b c d e f g h i j k l m n o p comb1 comb2
* Delete the project used to obtain the current directory
project RDDataTask, pclear
```