```
*******************************************************************************
*     Individual Discount Rates: A Meta-Analysis of Experimental Evidence      *
*                              November 17, 2020                                *
*******************************************************************************
log using discrate.log, replace
import excel using discrate.xlsx, sheet("data") firstrow
xtset idstudy
set more off
label variable discrate "Discount rate"
destring standard_error, replace ignore("na")
save discrate.dta, replace

*******************************************************************************
* Data preparation - bootstrapping standard errors
*******************************************************************************
use discrate.dta, replace
generate discrate_boot = discrate
save discrate_boot.dta, replace

summarize idstudy
local boots = r(max)
local reps = 1000

clear
set obs `boots'
generate idstudy = .
generate store_means = .
generate original_means = .
generate store_stdevs = .

quietly{
        forvalues i = 1(1) `boots' {
                preserve
                use discrate_boot idstudy using discrate_boot.dta, clear
                if floor((`i'-1)/1) == (`i'-1)/1 {
                        noisily display "Worknig on `i' out of `boots' at $S_TIME"
                }
                keep if idstudy==`i'
                summarize discrate_boot
                local mean_discrate_orig = r(mean)
                tempfile boot_study_`i'

                bootstrap mean=r(mean), reps(`reps') strata(idstudy) seed(1234)
saving(boot_study_`i', replace): summarize discrate_boot, detail
                use boot_study_`i', replace
                summarize mean

                local mean_discrate = r(mean)
                local mean_discrate_sd = r(sd)
                restore
                replace idstudy = `i' in `i'

                display "replacing store_means = `mean_discrate' in `i'"
                replace store_means = `mean_discrate' in `i'
                display "replacing original_means = `mean_discrate_orig' in `i'"
                replace original_means = `mean_discrate_orig' in `i'
                display "replacing store_stdevs = `mean_discrate_sd' in `i'"
                replace store_stdevs = `mean_discrate_sd' in `i'
        }
}
save boot_results.dta, replace
        forvalues i = 1(1) `boots' {
        use boot_study_`i'.dta
        erase boot_study_`i'.dta
```

```
}
use discrate_boot.dta, replace
merge m:1 idstudy using boot_results.dta
drop _merge store_means discrate_boot
rename original_means study_mean
rename store_stdevs study_se

local p = 0.05
winsor discrate, gen(discrate_win) p(`p')
winsor standard_error, gen(standard_error_win) p(`p')
generate precision = .
replace precision = 1/standard_error  if standard_error != .
winsor precision, gen(precision_win) p(`p')

generate standard_error_comb = .
replace standard_error_comb = standard_error if standard_error != .
replace standard_error_comb = study_se if standard_error == .
winsor standard_error_comb, gen(standard_error_comb_win) p(`p')
label variable standard_error_comb_win "SE (publication bias)"

generate students_lab_experiment = 0
replace students_lab_experiment = 1 if students==1 & lab_experiment==1

generate discrate_negative = 0
replace discrate_negative = 1 if discrate<0

generate discrate_na = .
replace discrate_na = discrate if discounting != "na"
generate standard_error_comb_na = .
replace standard_error_comb_na = standard_error_comb if discounting != "na"

erase boot_results.dta
save discrate_boot.dta, replace
*******************************************************************************
* Summary statistics
*******************************************************************************
use discrate_boot.dta, replace

summarize discrate standard_error standard_error_comb
summarize discrate standard_error if standard_error != .
summarize discrate_win, detail

correlate standard_error_comb_win hyperbolic_discounting exponential_discounting delay frontend_delay
lab_experiment real_reward matching_task health_domain other_domain negative_framing neutral_framing
stakes sample_size students students_lab_experiment males_only females_only north_america asia africa
citations publication_year
summarize discrate_win standard_error_win standard_error_comb_win hyperbolic_discounting
exponential_discounting delay frontend_delay lab_experiment real_reward matching_task health_domain
other_domain negative_framing neutral_framing stakes sample_size students students_lab_experiment
males_only females_only north_america asia africa citations publication_year
summarize discrate_win standard_error_comb_win hyperbolic_discounting exponential_discounting delay
frontend_delay lab_experiment real_reward matching_task health_domain other_domain negative_framing
neutral_framing stakes sample_size students students_lab_experiment males_only females_only
north_america asia africa citations publication_year [aweight=invperst]
summarize standard_error_win [aweight=invperst_origse]

summarize nobs
local p = 0.05
winsor nobs, gen(nobs_win) p(`p')
summarize nobs_win

mean discrate_win
mean discrate_win if hyperbolic_discounting==1
mean discrate_win if exponential_discounting==1
mean discrate_win if hyperbolic_discounting==0 & exponential_discounting==0
```

```
mean discrate_win if frontend_delay==1
mean discrate_win if frontend_delay==0
mean discrate_win if lab_experiment==1
mean discrate_win if lab_experiment==0
mean discrate_win if real_reward==1
mean discrate_win if real_reward==0
mean discrate_win if matching_task==1
mean discrate_win if matching_task==0
mean discrate_win if health_domain==1
mean discrate_win if other_domain==1
mean discrate_win if money_domain==1
mean discrate_win if negative_framing==1
mean discrate_win if neutral_framing==1
mean discrate_win if neutral_framing==0 & negative_framing==0
mean discrate_win if students==1
mean discrate_win if students==0
mean discrate_win if males_only==1
mean discrate_win if females_only==1
mean discrate_win if males_only==0 & females_only==0
mean discrate_win if north_america==1
mean discrate_win if asia==1
mean discrate_win if africa==1
mean discrate_win if north_america==0 & asia==0 & africa==0
mean discrate_win [aweight=invperst]
mean discrate_win [aweight=invperst] if hyperbolic_discounting==1
mean discrate_win [aweight=invperst] if exponential_discounting==1
mean discrate_win [aweight=invperst] if hyperbolic_discounting==0 & exponential_discounting==0
mean discrate_win [aweight=invperst] if frontend_delay==1
mean discrate_win [aweight=invperst] if frontend_delay==0
mean discrate_win [aweight=invperst] if lab_experiment==1
mean discrate_win [aweight=invperst] if lab_experiment==0
mean discrate_win [aweight=invperst] if real_reward==1
mean discrate_win [aweight=invperst] if real_reward==0
mean discrate_win [aweight=invperst] if matching_task==1
mean discrate_win [aweight=invperst] if matching_task==0
mean discrate_win [aweight=invperst] if health_domain==1
mean discrate_win [aweight=invperst] if other_domain==1
mean discrate_win [aweight=invperst] if money_domain==1
mean discrate_win [aweight=invperst] if negative_framing==1
mean discrate_win [aweight=invperst] if neutral_framing==1
mean discrate_win [aweight=invperst] if neutral_framing==0 & negative_framing==0
mean discrate_win [aweight=invperst] if students==1
mean discrate_win [aweight=invperst] if students==0
mean discrate_win [aweight=invperst] if males_only==1
mean discrate_win [aweight=invperst] if females_only==1
mean discrate_win [aweight=invperst] if males_only==0 & females_only==0
mean discrate_win [aweight=invperst] if north_america==1
mean discrate_win [aweight=invperst] if asia==1
mean discrate_win [aweight=invperst] if africa==1
mean discrate_win [aweight=invperst] if north_america==0 & asia==0 & africa==0

summarize discrate, detail
local m = r(mean)
local med = r(p50)
graph twoway histogram discrate if discrate < 5 & discrate > -0.5, bin(50) frequency xline(`m',
lpattern(solid) lcolor(red)) xline(`med', lpattern(dash) lcolor(red)) saving(_histogram, replace)

summarize discrate, detail
local m = r(mean)
local med = r(p50)
graph hbox discrate if discrate < 5 & discrate > -0.5, over(study) xsize(7) ysize(8) scale(0.5)
aspectratio(2) yline(`m', lpattern(solid)) saving(_studies, replace)
*************************************************************************
* Publication bias - Funnel plot (Egger et al., 1997)
*************************************************************************
```

```
generate precision_comb = .
replace precision_comb = (1 / standard_error_comb)
generate precision_comb_log = .
replace precision_comb_log = ln(precision_comb)

summarize discrate, detail
local m = r(mean)
twoway scatter precision_comb_log discrate if precision_comb_log < 8 & precision_comb_log > 0  &
discrate < 6 & discrate > -0.5, ytitle("Logarithm of discount rate precision [ln(1/SE)]")
xtitle("Discount rate") /*xline(`m', lpattern(solid) lcolor (black))*/ xline(0, lpattern(dash) lcolor
(black)) saving(_funnel, replace)
*************************************************************************
* Publication bias - FAT-PET (Stanley, 2005)
*************************************************************************
local p = 0.05
winsor instrument, gen(instrument_win) p(`p')

eststo: ivreg2 discrate_win standard_error_comb_win, cluster(idstudy)
eststo: xtreg discrate_win standard_error_comb_win, fe cluster(idstudy)
eststo: xi, noomit: ivreg2 discrate_win (standard_error_comb_win=instrument_win), cluster(idstudy)
eststo: ivreg2 discrate_win standard_error_comb_win [pweight = 1/standard_error_comb_win],
cluster(idstudy)
esttab using _pet.tex, se booktabs replace compress title(Funnel asymmetry tests - PET
\label{tab:fatpetcheck2}) mtitles( "OLS" "FE" "IV" "Precision") nonumber nogaps width(1\\hsize)
star(\sym{*} 0.10 \sym{**} 0.05 \sym{***} 0.01)
eststo clear

eststo: ivreg2 discrate_win standard_error_win if !missing(standard_error), cluster(idstudy)
eststo: xtreg discrate_win standard_error_win if !missing(standard_error), fe cluster(idstudy)
eststo: xi, noomit: ivreg2 discrate_win (standard_error_win=instrument_win) if
!missing(standard_error), cluster(idstudy)
eststo: ivreg2 discrate_win standard_error_win if !missing(standard_error) [pweight =
1/standard_error_win], cluster(idstudy)
esttab using _pet_check_origse.tex, se booktabs replace compress title(Funnel asymmetry tests - PET
robust check \label{tab:fatpetcheck}) mtitles("OLS" "FE" "IV" "Precision") nonumber nogaps
width(1\\hsize) star(\sym{*} 0.10 \sym{**} 0.05 \sym{***} 0.01)
eststo clear

eststo: ivreg2 discrate_win standard_error_comb_win, cluster(idauthor)
eststo: xtreg discrate_win standard_error_comb_win, fe cluster(idauthor)
eststo: xi, noomit: ivreg2 discrate_win (standard_error_comb_win=instrument_win), cluster(idauthor)
eststo: ivreg2 discrate_win standard_error_comb_win [pweight = 1/standard_error_comb_win],
cluster(idauthor)
esttab using _pet_check_idauthors.tex, se booktabs replace compress title(Funnel asymmetry tests -
PET robust check (idauthors) \label{tab:fatpetcheck2}) mtitles( "OLS" "FE" "IV" "Precision") nonumber
nogaps width(1\\hsize) star(\sym{*} 0.10 \sym{**} 0.05 \sym{***} 0.01)
eststo clear

eststo: ivreg2 discrate_win standard_error_comb_win if discounting!="na", cluster(idstudy)
eststo: xtreg discrate_win standard_error_comb_win if discounting!="na", fe cluster(idstudy)
eststo: xi, noomit: ivreg2 discrate_win (standard_error_comb_win=instrument_win) if
discounting!="na", cluster(idstudy)
eststo: ivreg2 discrate_win standard_error_comb_win if discounting!="na" [pweight =
1/standard_error_comb_win], cluster(idstudy)
esttab using _pet_check_nadisc.tex, se booktabs replace compress title(Funnel asymmetry tests - NA
discounting omitted \label{tab:fatpetcheck2}) mtitles( "OLS" "FE" "IV" "Precision") nonumber nogaps
width(1\\hsize) star(\sym{*} 0.10 \sym{**} 0.05 \sym{***} 0.01)
eststo clear

generate se_discratenegative =  standard_error_comb_win*discrate_negative
eststo: ivreg2 discrate_win standard_error_comb_win se_discratenegative, cluster(idstudy)
eststo: xtreg discrate_win standard_error_comb_win se_discratenegative, fe cluster(idstudy)
eststo: xi,noomit: ivreg2 discrate_win standard_error_comb_win se_discratenegative
(standard_error_comb_win=instrument_win), cluster(idstudy)
eststo: ivreg2 discrate_win standard_error_comb_win se_discratenegative [pweight =
```

```
1/standard_error_comb_win], cluster(idstudy)
esttab using _pet_check_negative.tex, se booktabs replace compress title(Funnel asymmetry tests -
negative discounting \label{tab:fatpetcheck2}) mtitles( "OLS" "FE" "IV" "Precision") nonumber nogaps
width(1\\hsize) star(\sym{*} 0.10 \sym{**} 0.05 \sym{***} 0.01)
eststo clear

generate se_median =  standard_error_comb_win*median
eststo: ivreg2 discrate_win standard_error_comb_win se_median , cluster(idstudy)
eststo: xtreg discrate_win standard_error_comb_win se_median , fe cluster(idstudy)
eststo: xi,noomit: ivreg2 discrate_win standard_error_comb_win se_median
(standard_error_comb_win=instrument_win), cluster(idstudy)
eststo: ivreg2 discrate_win standard_error_comb_win se_median  [pweight = 1/standard_error_comb_win],
cluster(idstudy)
esttab using _pet_check_median.tex, se booktabs replace compress title(Funnel asymmetry tests -
negative discounting \label{tab:fatpetcheck2}) mtitles( "OLS" "FE" "IV" "Precision") nonumber nogaps
width(1\\hsize) star(\sym{*} 0.10 \sym{**} 0.05 \sym{***} 0.01)
eststo clear

* caliper test -0.5 & 0.5
eststo: ivreg2 discrate_win standard_error_comb_win if discrate_win>=-0.5 & discrate_win<=0.5,
cluster(idstudy)
eststo: xtreg discrate_win standard_error_comb_win if discrate_win>=-0.5 & discrate_win<=0.5, fe
cluster(idstudy)
eststo: xi, noomit: ivreg2 discrate_win (standard_error_comb_win=instrument_win) if
discrate_win>=-0.5 & discrate_win<=0.5, cluster(idstudy)
eststo: ivreg2 discrate_win standard_error_comb_win if discrate_win>=-0.5 & discrate_win<=0.5
[pweight = 1/standard_error_comb_win], cluster(idstudy)
esttab using _pet_caliper05.tex, se booktabs replace compress title(Funnel asymmetry tests - PET
\label{tab:fatpet}) mtitles( "OLS" "FE" "IV" "Precision") nonumber nogaps width(1\\hsize)
star(\sym{*} 0.10 \sym{**} 0.05 \sym{***} 0.01)
eststo clear

* caliper test -1.0 & 1.0
eststo: ivreg2 discrate_win standard_error_comb_win if discrate_win>=-1.0 & discrate_win<=1.0,
cluster(idstudy)
eststo: xtreg discrate_win standard_error_comb_win if discrate_win>=-1.0 & discrate_win<=1.0, fe
cluster(idstudy)
eststo: xi, noomit: ivreg2 discrate_win (standard_error_comb_win=instrument_win) if
discrate_win>=-1.0 & discrate_win<=1.0, cluster(idstudy)
eststo: ivreg2 discrate_win standard_error_comb_win if discrate_win>=-1.0 & discrate_win<=1.0
[pweight = 1/standard_error_comb_win], cluster(idstudy)
esttab using _pet_caliper10.tex, se booktabs replace compress title(Funnel asymmetry tests - PET
\label{tab:fatpet}) mtitles( "OLS" "FE" "IV" "Precision") nonumber nogaps width(1\\hsize)
star(\sym{*} 0.10 \sym{**} 0.05 \sym{***} 0.01)
eststo clear

* caliper test 0.5 & 1.5
eststo: ivreg2 discrate_win standard_error_comb_win if discrate_win>=0.5 & discrate_win<=1.5,
cluster(idstudy)
eststo: xtreg discrate_win standard_error_comb_win if discrate_win>=0.5 & discrate_win<=1.5, fe
cluster(idstudy)
eststo: xi, noomit: ivreg2 discrate_win (standard_error_comb_win=instrument_win) if discrate_win>=0.5
& discrate_win<=1.5, cluster(idstudy)
eststo: ivreg2 discrate_win standard_error_comb_win if discrate_win>=0.5 & discrate_win<=1.5 [pweight
= 1/standard_error_comb_win], cluster(idstudy)
esttab using _pet_caliper15.tex, se booktabs replace compress title(Funnel asymmetry tests - PET
\label{tab:fatpet}) mtitles( "OLS" "FE" "IV" "Precision") nonumber nogaps width(1\\hsize)
star(\sym{*} 0.10 \sym{**} 0.05 \sym{***} 0.01)
eststo clear

* caliper test 0.25 & 0.75
eststo: ivreg2 discrate_win standard_error_comb_win if discrate_win>=0.25 & discrate_win<=0.75,
cluster(idstudy)
eststo: xtreg discrate_win standard_error_comb_win if discrate_win>=0.25 & discrate_win<=0.75, fe
cluster(idstudy)
```

```
eststo: xi, noomit: ivreg2 discrate_win (standard_error_comb_win=instrument_win) if
discrate_win>=0.25 & discrate_win<=0.75, cluster(idstudy)
eststo: ivreg2 discrate_win standard_error_comb_win if discrate_win>=0.25 & discrate_win<=0.75
[pweight = 1/standard_error_comb_win], cluster(idstudy)
esttab using _pet_caliper75.tex, se booktabs replace compress title(Funnel asymmetry tests - PET
\label{tab:fatpet}) mtitles( "OLS" "FE" "IV" "Precision") nonumber nogaps width(1\\hsize)
star(\sym{*} 0.10 \sym{**} 0.05 \sym{***} 0.01)
eststo clear
*******************************************************************************
* Publication bias - Top10 method (Stanley et al., 2010)
*******************************************************************************
summarize precision if precision != ., detail
local top10bound1 = r(p90)
summarize discrate if precision > `top10bound1' & precision != .
summarize precision_comb, detail
local top10bound2 = r(p90)
summarize discrate if precision_comb > `top10bound2'
summarize precision_comb if discounting!="na", detail
local top10bound3 = r(p90)
summarize discrate if precision_comb > `top10bound3' & discounting!="na"
*******************************************************************************
* Publication bias - WAAP (Ioannidis et al., 2017)
*******************************************************************************
summarize discrate [aweight=1/(standard_error*standard_error)]
gen waapbound1 = abs(r(mean))/2.8
summarize discrate if standard_error < waapbound1

summarize discrate [aweight=1/(standard_error_comb*standard_error_comb)]
gen waapbound2 = abs(r(mean))/2.8
summarize discrate if standard_error_comb < waapbound2

summarize discrate if discounting!="na" [aweight=1/(standard_error_comb*standard_error_comb)]
gen waapbound3 = abs(r(mean))/2.8
summarize discrate if standard_error_comb < waapbound3 & discounting!="na"
*******************************************************************************
* Publication bias - Endogenous kink (Bom & Rachinger, 2020)
*      code downloaded from https://sites.google.com/site/heikorachinger/codes
*******************************************************************************
quietly{
clear
use discrate_boot.dta, replace
rename discrate bs
rename standard_error_comb sebs
gen ones=1
sum
local M=r(N)
sum sebs
local sebs_min=r(min)
local sebs_max=r(max)
gen sebs2=sebs^2
gen wis=ones/sebs2
gen bs_sebs=bs/sebs
gen ones_sebs=ones/sebs
gen bswis=bs*wis
sum wis
local wis_sum=r(sum)

regress bs_sebs ones_sebs ones,noc
local pet=_b[ones_sebs]
local t1_linreg = (_b[ones_sebs]/_se[ones_sebs])
local b_lin=_b[ones_sebs]
local Q1_lin = e(rss)
di `t1_linreg'
local abs_t1_linreg = abs(`t1_linreg')
di `abs_t1_linreg'
```

```
regress bs_sebs ones_sebs sebs,noc
local peese=_b[ones_sebs]
local b_sq=_b[ones_sebs]
local Q1_sq = e(rss)
di `Q1_sq'

if `abs_t1_linreg' > invt(`M-2', 0.975) {
    local combreg=`b_sq'
        local Q1=`Q1_sq'
        }
else {
    local combreg=`b_lin'
        local Q1=`Q1_lin'
}

local sigh2hat=max(0,`M'*((`Q1'/(`M'-e(df_m)-1))-1)/`wis_sum')
local sighhat=sqrt(`sigh2hat')

if `combreg'>1.96*`sighhat' {
    local a1=(`combreg'-1.96*`sighhat')*(`combreg'+1.96*`sighhat')/(2*1.96*`combreg')
}
else {
        local a1=0
        }
        rename bs bs_original
        rename bs_sebs bs
        rename ones_sebs constant
        rename ones pub_bias
    noisily: display "EK regression: "
if `a1'>`sebs_min' & `a1'<`sebs_max' {
    gen sebs_a1=sebs-`a1' if sebs>`a1'
        replace sebs_a1=0 if sebs<=`a1'
        gen pubbias=sebs_a1/sebs
        noisily regress bs constant pubbias, noc
        local b0_ek=_b[constant]
        local b1_ek=_b[pubbias]
        local sd0_ek=_se[constant]
        local sd1_ek=_se[pubbias]
}
else if `a1'<`sebs_min' {
    noisily regress bs constant pub_bias, noc
        local b0_ek=_b[constant]
        local b1_ek=_b[pub_bias]
        local sd0_ek=_se[constant]
        local sd1_ek=_se[pub_bias]
}
else if `a1'>`sebs_max' {
    noisily regress bs constant, noc
        local b0_ek=_b[constant]
        local sd0_ek=_se[constant]
}
noisily: display "EK's mean effect estimate (alpha1) and standard error:"
noisily: di `b0_ek'
noisily: di `sd0_ek'
noisily: display "EK's publication bias estimate (delta) and standard error:"
noisily: di `b1_ek'
noisily: di `sd1_ek'
}
rename bs discrate
rename sebs standard_error_comb

quietly{
clear
use discrate_boot.dta, replace
```

```
rename discrate bs
rename standard_error sebs
gen ones=1
sum
local M=r(N)
sum sebs
local sebs_min=r(min)
local sebs_max=r(max)
gen sebs2=sebs^2
gen wis=ones/sebs2
gen bs_sebs=bs/sebs
gen ones_sebs=ones/sebs
gen bswis=bs*wis
sum wis
local wis_sum=r(sum)
regress bs_sebs ones_sebs ones,noc
local pet=_b[ones_sebs]
local t1_linreg = (_b[ones_sebs]/_se[ones_sebs])
local b_lin=_b[ones_sebs]
local Q1_lin = e(rss)
di `t1_linreg'
local abs_t1_linreg = abs(`t1_linreg')
di `abs_t1_linreg'
regress bs_sebs ones_sebs sebs,noc
local peese=_b[ones_sebs]
local b_sq=_b[ones_sebs]
local Q1_sq = e(rss)
di `Q1_sq'
if `abs_t1_linreg' > invt(`M-2', 0.975) {
    local combreg=`b_sq'
        local Q1=`Q1_sq'
        }
else {
    local combreg=`b_lin'
        local Q1=`Q1_lin'
}
local sigh2hat=max(0,`M'*((`Q1'/(`M'-e(df_m)-1))-1)/`wis_sum')
local sighhat=sqrt(`sigh2hat')
if `combreg'>1.96*`sighhat' {
    local a1=(`combreg'-1.96*`sighhat')*(`combreg'+1.96*`sighhat')/(2*1.96*`combreg')
}
else {
        local a1=0
        }
        rename bs bs_original
        rename bs_sebs bs
        rename ones_sebs constant
        rename ones pub_bias
    noisily: display "EK regression: "

if `a1'>`sebs_min' & `a1'<`sebs_max' {
    gen sebs_a1=sebs-`a1' if sebs>`a1'
        replace sebs_a1=0 if sebs<=`a1'
        gen pubbias=sebs_a1/sebs
        noisily regress bs constant pubbias, noc
        local b0_ek=_b[constant]
        local b1_ek=_b[pubbias]
        local sd0_ek=_se[constant]
        local sd1_ek=_se[pubbias]
}
else if `a1'<`sebs_min' {
    noisily regress bs constant pub_bias, noc
        local b0_ek=_b[constant]
        local b1_ek=_b[pub_bias]
        local sd0_ek=_se[constant]
```

```
        local sd1_ek=_se[pub_bias]
}
else if `a1'>`sebs_max' {
    noisily regress bs constant, noc
        local b0_ek=_b[constant]
        local sd0_ek=_se[constant]
}
noisily: display "EK's mean effect estimate (alpha1) and standard error:"
noisily: di `b0_ek'
noisily: di `sd0_ek'
noisily: display "EK's publication bias estimate (delta) and standard error:"
noisily: di `b1_ek'
noisily: di `sd1_ek'
}
rename bs discrate
rename sebs standard_error


quietly{
clear
use discrate_boot.dta, replace
rename discrate_na bs
rename standard_error_comb_na sebs
gen ones=1
sum
local M=r(N)
sum sebs
local sebs_min=r(min)
local sebs_max=r(max)
gen sebs2=sebs^2
gen wis=ones/sebs2
gen bs_sebs=bs/sebs
gen ones_sebs=ones/sebs
gen bswis=bs*wis
sum wis
local wis_sum=r(sum)

regress bs_sebs ones_sebs ones,noc
local pet=_b[ones_sebs]
local t1_linreg = (_b[ones_sebs]/_se[ones_sebs])
local b_lin=_b[ones_sebs]
local Q1_lin = e(rss)
di `t1_linreg'
local abs_t1_linreg = abs(`t1_linreg')
di `abs_t1_linreg'

regress bs_sebs ones_sebs sebs,noc
local peese=_b[ones_sebs]
local b_sq=_b[ones_sebs]
local Q1_sq = e(rss)
di `Q1_sq'

if `abs_t1_linreg' > invt(`M-2', 0.975) {
    local combreg=`b_sq'
        local Q1=`Q1_sq'
        }
else {
    local combreg=`b_lin'
        local Q1=`Q1_lin'
}

local sigh2hat=max(0,`M'*((`Q1'/(`M'-e(df_m)-1))-1)/`wis_sum')
local sighhat=sqrt(`sigh2hat')

if `combreg'>1.96*`sighhat' {
```

```
    local a1=(`combreg'-1.96*`sighhat')*(`combreg'+1.96*`sighhat')/(2*1.96*`combreg')
}
else {
        local a1=0
        }
        rename bs bs_original
        rename bs_sebs bs
        rename ones_sebs constant
        rename ones pub_bias
    noisily: display "EK regression: "
if `a1'>`sebs_min' & `a1'<`sebs_max' {
    gen sebs_a1=sebs-`a1' if sebs>`a1'
        replace sebs_a1=0 if sebs<=`a1'
        gen pubbias=sebs_a1/sebs
        noisily regress bs constant pubbias, noc
        local b0_ek=_b[constant]
        local b1_ek=_b[pubbias]
        local sd0_ek=_se[constant]
        local sd1_ek=_se[pubbias]
}
else if `a1'<`sebs_min' {
    noisily regress bs constant pub_bias, noc
        local b0_ek=_b[constant]
        local b1_ek=_b[pub_bias]
        local sd0_ek=_se[constant]
        local sd1_ek=_se[pub_bias]
}
else if `a1'>`sebs_max' {
    noisily regress bs constant, noc
        local b0_ek=_b[constant]
        local sd0_ek=_se[constant]
}
noisily: display "EK's mean effect estimate (alpha1) and standard error:"
noisily: di `b0_ek'
noisily: di `sd0_ek'
noisily: display "EK's publication bias estimate (delta) and standard error:"
noisily: di `b1_ek'
noisily: di `sd1_ek'
}
rename bs discrate_na
rename sebs standard_error_comb_na
*************************************************************************
* Publication bias - Stem-based method in R (Furukawa, 2020)
*************************************************************************
/*
source("stem_method.R") #github.com/Chishio318/stem-based_method
datadiscrate = read.table("clipboard-512", sep="\t", header=TRUE)
stem_results = stem(datadiscrate$armel, datadiscrate$se, param)
view(stem_results$estimates)
*/
*************************************************************************
* Publication bias - p-uniform* (van Aert & van Assen, 2019) - code for Stata & R
*************************************************************************
*generate input data
gen tstatistics = discrate_win/standard_error_comb_win
gen variance_comb_win = standard_error_comb_win*standard_error_comb_win
local p = 0.05
winsor nobs, gen(nobs_win) p(`p')
bysort idstudy: egen discrate_med = median(discrate_win)
bysort idstudy: egen variance_med = median(variance_comb_win)
bysort idstudy: egen tstat_med = median(tstatistics)

preserve
collapse  (lastnm) discrate_med variance_med tstat_med (p50) nobs_med=nobs_win, by(idstudy)
save "pcurve_med.dta", replace
```

```
restore

/* code for R
library(puniform)
library(metafor)
library(dmetar)
library(haven)
library(tidyverse)
library(meta)

data <- read_dta("pcurve.dta")
#puni_star(yi = data$discrate_med, vi = data$variance_med, side="right", method="ML",alpha = 0.05,
control=list( max.iter=1000,tol=0.1,reps=10000, int=c(0,2), verbose=TRUE))
# alternative
puni_star(tobs = data$tstat_med, ni = data$nobs_med, alpha = 0.05, side = "right", method = "ML",
control=list(stval.tau=0.5, max.iter=10000,tol=0.1,reps=10000, int=c(0,2), verbose=TRUE))
*/
*************************************************************************
* Heterogeneity - Frequentist check (OLS)
*************************************************************************
use discrate_boot.dta, replace

correlate standard_error_comb_win hyperbolic_discounting exponential_discounting delay frontend_delay
lab_experiment real_reward matching_task health_domain other_domain negative_framing neutral_framing
sample_size students students_lab_experiment males_only females_only north_america asia africa
citations publication_year stakes
collin standard_error_comb_win hyperbolic_discounting exponential_discounting delay frontend_delay
lab_experiment real_reward matching_task health_domain other_domain negative_framing neutral_framing
sample_size students students_lab_experiment males_only females_only north_america asia africa
citations publication_year stakes
correlate standard_error_comb_win hyperbolic_discounting exponential_discounting delay frontend_delay
lab_experiment real_reward matching_task health_domain other_domain negative_framing neutral_framing
sample_size students students_lab_experiment males_only females_only north_america asia africa
citations publication_year
collin standard_error_comb_win hyperbolic_discounting exponential_discounting delay frontend_delay
lab_experiment real_reward matching_task health_domain other_domain negative_framing neutral_framing
sample_size students students_lab_experiment males_only females_only north_america asia africa
citations publication_year

correlate discrate_win standard_error_comb_win hyperbolic_discounting exponential_discounting delay
frontend_delay lab_experiment real_reward matching_task health_domain other_domain negative_framing
neutral_framing sample_size students students_lab_experiment males_only females_only north_america
asia africa citations publication_year if exact_delay==1
collin discrate_win standard_error_comb_win hyperbolic_discounting exponential_discounting delay
frontend_delay lab_experiment real_reward matching_task health_domain other_domain negative_framing
neutral_framing sample_size students students_lab_experiment males_only females_only north_america
asia africa citations publication_year if exact_delay==1

reg discrate_win standard_error_comb_win hyperbolic_discounting exponential_discounting delay
frontend_delay lab_experiment real_reward matching_task health_domain other_domain negative_framing
neutral_framing sample_size students students_lab_experiment males_only females_only north_america
asia africa citations publication_year if exact_delay==1, cluster(idstudy)
reg discrate_win standard_error_comb_win hyperbolic_discounting exponential_discounting delay
frontend_delay  real_reward matching_task health_domain other_domain negative_framing neutral_framing
sample_size students students_lab_experiment males_only females_only north_america asia africa
citations publication_year if exact_delay==1, cluster(idstudy)

eststo: reg discrate_win standard_error_comb_win lab_experiment health_domain other_domain
negative_framing sample_size students students_lab_experiment asia africa publication_year,
cluster(idstudy)
*************************************************************************
* Heterogeneity - Bayesian model averaging in R
*************************************************************************
/*
library(BMS)
```

```
library(corrplot)
    *ctrl+C from discrate.xlsx, sheet("data.r")
datadiscrate = read.table("clipboard-512", sep="\t", header=TRUE)
discrate1 = bms(datadiscrate, burn=1e6,iter=2e6, g="UIP", mprior="dilut", nmodel=5000, mcmc="bd",
user.int=FALSE)
discrate2 = bms(datadiscrate, burn=1e6,iter=2e6, g="BRIC", mprior="random", nmodel=5000, mcmc="bd",
user.int=FALSE)
coef(discrate1, order.by.pip = F, exact=T, include.constant=T)
image(discrate1, yprop2pip=FALSE, order.by.pip=TRUE, do.par=TRUE, do.grid=TRUE, do.axis=TRUE,
cex.axis = 0.7)
summary(discrate1)
plot(discrate1)
print(discrate1$topmod[1])

col<- colorRampPalette(c("red", "white", "blue"))
M <- round(cor(datadiscrate, use="complete.obs", method="pearson"), 2)
corrplot (M, method="color", col=col(200), type="upper", addCoef.col = "black", number.cex=0.5,
tl.pos = c("lt"), tl.col="black", tl.srt=45, sig.level = 0.01, insig = "blank")
*/
*************************************************************************
* HETEROGENEITY - Frequentist model averaging (Hanson) code for R
*************************************************************************
/*
library(foreign)
library(xtable)
library(LowRankQP)
datadiscrate=read.table("clipboard-512", sep="\t", header=TRUE)
datadiscrate <-na.omit(datadiscrate)
x.data <- datadiscrate[,-1]
const_<-c(1)
x.data <-cbind(const_,x.data)

x <- sapply(1:ncol(x.data),function(i){x.data[,i]/max(x.data[,i])})
scale.vector <- as.matrix(sapply(1:ncol(x.data),function(i){max(x.data[,i])}))
Y <- as.matrix(datadiscrate[,1])
output.colnames <- colnames(x.data)
full.fit <- lm(Y~x-1)
beta.full <- as.matrix(coef(full.fit))
M <- k <- ncol(x)
n <- nrow(x)
beta <- matrix(0,k,M)
e <- matrix(0,n,M)
K_vector <- matrix(c(1:M))
var.matrix <- matrix(0,k,M)
bias.sq <- matrix(0,k,M)

for(i in 1:M)
{
  X <- as.matrix(x[,1:i])
  ortho <- eigen(t(X)%*%X)
  Q <- ortho$vectors ; lambda <- ortho$values
  x.tilda <- X%*%Q%*%(diag(lambda^-0.5,i,i))
  beta.star <- t(x.tilda)%*%Y
  beta.hat <- Q%*%diag(lambda^-0.5,i,i)%*%beta.star
  beta[1:i,i] <- beta.hat
  e[,i] <- Y-x.tilda%*%as.matrix(beta.star)
  bias.sq[,i] <- (beta[,i]-beta.full)^2
  var.matrix.star <- diag(as.numeric(((t(e[,i])%*%e[,i])/(n-i))),i,i)
  var.matrix.hat <- var.matrix.star%*%(Q%*%diag(lambda^-1,i,i)%*%t(Q))
  var.matrix[1:i,i] <- diag(var.matrix.hat)
  var.matrix[,i] <- var.matrix[,i]+ bias.sq[,i]
}

e_k <- e[,M]
sigma_hat <- as.numeric((t(e_k)%*%e_k)/(n-M))
```

```
G <- t(e)%*%e
a <- ((sigma_hat)^2)*K_vector
A <- matrix(1,1,M)
b <- matrix(1,1,1)
u <- matrix(1,M,1)
optim <- LowRankQP(Vmat=G,dvec=a,Amat=A,bvec=b,uvec=u,method="LU",verbose=FALSE)
weights <- as.matrix(optim$alpha)
beta.scaled <- beta%*%weights
final.beta <- beta.scaled/scale.vector
std.scaled <- sqrt(var.matrix)%*%weights
final.std <- std.scaled/scale.vector
results.reduced <- as.matrix(cbind(final.beta,final.std))
rownames(results.reduced) <- output.colnames; colnames(results.reduced) <- c("Coefficient", "Sd.
Err")
MMA.fls <- round(results.reduced,4)
MMA.fls <- data.frame(MMA.fls)
t <- as.data.frame(MMA.fls$Coefficient/MMA.fls$Sd..Err)
MMA.fls$pv <-round( (1-apply(as.data.frame(apply(t,1,abs)), 1, pnorm))*2,3)
MMA.fls$names <- rownames(MMA.fls)
names <- c(colnames(datadiscrate))
names <- c(names,"const_")
MMA.fls <- MMA.fls[match(names, MMA.fls$names),]
MMA.fls$names <- NULL
MMA.fls
*/
*********************************************************************************
window manage close graph
log close
exit, clear
```