Codebook

This guide can be sued to reproduced the data analyses reported in "Fielding Before and After Baseball's Great Transformation," *Journal of Sports Analytics*.

The AL/NL analyses consist of two sets: those reported in the body of the article; and the Defensive Regression Analysis models reported in the Supplemental Information. The data for the analyses are contained in two separate files, jsa_fielding_after_great_transformation_data.csv, and jsa_dra_negro_league_data.csv, respectively. For each set, this guide first sets for a table describing the variables in the relevant data file, and then reproduces the data scripts used to generate the reported analyses.

The data analysis scripts are Stata .do files. They are sufficiently annotated, however, to enable translation into alternative formats.

A. Article analyses

1. Variables

Note that the table identifies and describes the variables contained in the data file. The analyses use these variables to form additional composite variables, the identity and nature of which are described fully in the analysis script annotations.

Variable name	Values	Description	Derivation
yearID	1900-2024	AL/NL seasons	Lahman data base
lgID	AL=American League NL=National League	League	Lahman data base
teamID	string team identifiers	AL/NL teams	Lahman data base
FIP	FIP scores	team fielding-independent pitching score	Baseball Reference
rapg	runs allowed per game	total team runs divided by games played	Formed by dividing total runs allowed for season by games played. Source: Lahman data base, Baseball Refernce
rfield_br	fielding runs	"rfield" measure of individual team "fielding runs	Baseball Reference
uzr	fielding runs	"UZR" measure of individual team "fielding runs	FanGraphs
sc_fld	fielding runs	Statcast measure of individual team "fielding runs	baseballsavant.mlb.com
tfr	fielding runs	Smith DER measure of individual team "fielding runs	baseballprojection.com
fb_drs	fielding runs	Fielding bible "defensive runs saved" measure of individual team "fielding runs	fieldingbible.com
def	fielding runs	FanGraphs "defensive runs saved" measure of individual team "fielding runs post 2002	FanGraphs

fgfield	fielding runs	FanGraphs "defensive runs saved" measure of individual team "fielding runs" pre 2003	FanGraphs
RA	season runs allowed	total team runs allowed for season	Lahman data base; Baseball Refernce

2. Analsis script

```
clear
clear matrix
set matsize 800
set mem 500m
cd [your partition]
use [jsa_fielding_after_great_transformation_data.csv]

*****************************************************
*standardized multi-season variables
*****************************************************



// standardize rfield
	bysort yearID: egen mean_rfield = mean(rfield)
	bysort yearID: egen sd_rfield  = sd(rfield)
	gen z_rfield = (rfield - mean_rfield) / sd_rfield


	// standardize drs and fgfield uzr tfr dwar statcast def

	bysort yearID: egen mean_fgfield = mean(fgfield)
	bysort yearID: egen sd_fgfield  = sd(fgfield)
	gen z_fgfield = (fgfield - mean_fgfield) / sd_fgfield

	bysort yearID: egen mean_fb_drs = mean(fb_drs)
	bysort yearID: egen sd_fb_drs  = sd(fb_drs)
	gen z_fb_drs = (fb_drs - mean_fb_drs) / sd_fb_drs


	bysort yearID: egen mean_def = mean(def)
	bysort yearID: egen sd_def  = sd(def)
	gen z_def = (def - mean_def) / sd_def

		bysort yearID: egen mean_uzr = mean(uzr)
	bysort yearID: egen sd_uzr  = sd(uzr)
	gen z_uzr = (uzr - mean_uzr) / sd_uzr

	bysort yearID: egen mean_tfr = mean(tfr)
	bysort yearID: egen sd_tfr  = sd(tfr)
	gen z_tfr = (tfr - mean_tfr) / sd_tfr

	bysort yearID: egen mean_sc_fld = mean(sc_fld)
	bysort yearID: egen sd_sc_fld  = sd(sc_fld)
	gen z_sc_fld = (sc_fld - mean_sc_fld) / sd_sc_fld


// standardize FIP
	bysort yearID: egen mean_FIP = mean(FIP)
	bysort yearID: egen sd_FIP  = sd(FIP)
	gen z_FIP = (FIP - mean_FIP) / sd_FIP
// standardize runs against per game and runs scored per game
```

```
bysort yearID: egen mean_rapg = mean(rapg)
bysort yearID: egen sd_rapg  = sd(rapg)
gen z_rapg = (rapg - mean_rapg) / sd_rapg
bysort yearID: egen mean_rpg = mean(rpg)
bysort yearID: egen sd_rpg  = sd(rpg)
gen z_rpg = (rpg - mean_rpg) / sd_rpg


**************************************************
* Single-season regression models with rfield
**************************************************

///single season regression model with each measure

      gen season=year

      * Create variables to store results

gen b_FAM= .
gen t_b_FAM= .
gen R2_FAM = .
gen cons_FAM = .
gen t_cons_FAM = .
gen R2_FPFM = .
gen R2i_FM = .
gen b1_FPFM = .
gen t_b1_FPFM = .
gen b2_FPFM = .
gen t_b2_FPFM = .
gen cons_FPFM = .
gen t_cons_FPFM = .

foreach y of numlist 1900/2024 {
      di "Processing year `y' - rfield"
      * Model 1: RA = b1 * FIP
      quietly regress RA FIP if year == `y'
      replace b_FAM= _b[FIP] if year ==`y'
      replace t_b_FAM=  _b[FIP] / _se[FIP] if year ==`y'
      replace R2_FAM = e(r2) if year == `y'
      replace cons_FAM  = _b[_cons] if year == `y'
      replace t_cons_FAM  = _b[_cons] / _se[_cons] if year == `y'
      * Model 2: RA = b1' * FIP + b2_FPFM * rfield
      quietly regress RA FIP rfield if year == `y'
      replace R2_FPFM = e(r2) if year == `y'
      replace R2i_FM   = e(r2) - R2_FAM if year == `y'
      replace b1_FPFM  = _b[FIP] if year == `y'
      replace t_b1_FPFM = _b[FIP] / _se[FIP] if year == `y'
      replace b2_FPFM        = _b[rfield] if year == `y'
      replace t_b2_FPFM      = _b[rfield] / _se[rfield] if year == `y'
      replace cons_FPFM  = _b[_cons] if year == `y'
      replace t_cons_FPFM  = _b[_cons] / _se[_cons] if year == `y'
}

      preserve
      collapse b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM b2_FPFM
t_b2_FPFM cons_FPFM t_cons_FPFM R2_FPFM R2i_FM , by(season)
      foreach var in b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM
b2_FPFM t_b2_FPFM cons_FPFM t_cons_FPFM R2_FPFM R2i_FM {
      format `var' %9.2f
}

      export excel using "[your partition/file name]", firstrow(varlabels)   replace
```

```
	restore

	*****************************************************
	* Single-season regression models with OAA (tfr)
	*****************************************************

	foreach y of numlist 1912/2024 {
		di "Processing year `y' - tfr"

		* Model 2: RA = b1' * FIP + b2_FPFM * tfr
		quietly regress RA FIP tfr if year == `y'
		replace R2_FPFM = e(r2) if year == `y'
		replace R2i_FM   = e(r2) - R2_FAM if year == `y'
		replace b1_FPFM  = _b[FIP] if year == `y'
		replace t_b1_FPFM = _b[FIP] / _se[FIP] if year == `y'
		replace b2_FPFM        = _b[tfr] if year == `y'
		replace t_b2_FPFM      = _b[tfr] / _se[tfr] if year == `y'
		replace cons_FPFM  = _b[_cons] if year == `y'
		replace t_cons_FPFM  = _b[_cons] / _se[_cons] if year == `y'
	}

preserve
	collapse b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM b2_FPFM
t_b2_FPFM cons_FPFM t_cons_FPFM R2_FPFM R2i_FM , by(season)
	foreach var in b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM
b2_FPFM t_b2_FPFM cons_FPFM t_cons_FPFM R2_FPFM R2i_FM {
		format `var' %9.2f
	}

	export excel using "[your partition/file name]", firstrow(varlabels)  nolabel
replace

restore

	*****************************************************
	* Single-season regression models with DEF
	*****************************************************
	* Loop over each year
	foreach y of numlist 1900/2002 {
		di "Processing year `y'"

		* Model 2: RA = b1' * FIP + b2_FPFM * fgfield
		quietly: regress RA FIP fgfield if year == `y'
		replace R2_FPFM = e(r2) if year == `y'
		replace R2i_FM   = e(r2) - R2_FAM if year == `y'
		replace b1_FPFM  = _b[FIP] if year == `y'
		replace b2_FPFM        = _b[fgfield] if year == `y'
		replace t_b1_FPFM = _b[FIP] / _se[FIP] if year == `y'
		replace t_b2_FPFM      = _b[fgfield] / _se[fgfield] if year == `y'
		replace cons_FPFM   = _b[_cons] if year == `y'
		replace t_cons_FPFM   = _b[_cons] / _se[_cons] if year == `y'
	}

	foreach y of numlist 2003/2024 {
		di "Processing year `y'"

		* Model 2: RA = b1' * FIP + b2_FPFM * def
		quietly: regress RA FIP def if year == `y'
		replace R2_FPFM = e(r2) if year == `y'
		replace R2i_FM   = e(r2) - R2_FAM if year == `y'
		replace b1_FPFM  = _b[FIP] if year == `y'
		replace b2_FPFM        = _b[def] if year == `y'
		replace t_b1_FPFM = _b[FIP] / _se[FIP] if year == `y'
```

```
            replace t_b2_FPFM       = _b[def] / _se[def] if year == `y'
            replace cons_FPFM   = _b[_cons] if year == `y'
            replace t_cons_FPFM   = _b[_cons] / _se[_cons] if year == `y'
      }
      * Display the updated dataset
      preserve

      collapse b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM b2_FPFM
t_b2_FPFM cons_FPFM t_cons_FPFM R2_FPFM R2i_FM , by (season)
      foreach var in b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM
b2_FPFM t_b2_FPFM cons_FPFM t_cons_FPFM R2_FPFM R2i_FM {
            format `var' %9.2f
      }

      export excel using "[your partition/file name]", firstrow(varlabels)  nolabel
replace
      restore

      ****************************************************
      * Single-season regression models with UZR
      ****************************************************


      * Loop over each year
      foreach y of numlist 2003/2024 {
            di "Processing year `y'"

            * Model 2: RA = b1' * FIP + b2_FPFM * uzr
            quietly: regress RA FIP uzr if year == `y'
            replace R2_FPFM = e(r2) if year == `y'
            replace R2i_FM   = e(r2) - R2_FAM if year == `y'
            replace b1_FPFM  = _b[FIP] if year == `y'
            replace b2_FPFM       = _b[uzr] if year == `y'
            replace t_b1_FPFM = _b[FIP] / _se[FIP] if year == `y'
            replace t_b2_FPFM       = _b[uzr] / _se[uzr] if year == `y'
            replace cons_FPFM   = _b[_cons] if year == `y'
            replace t_cons_FPFM   = _b[_cons] / _se[_cons] if year == `y'
      }

* Display the updated dataset

preserve

drop if year <2003

collapse b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM b2_FPFM t_b2_FPFM
cons_FPFM t_cons_FPFM R2_FPFM R2i_FM, by (season)
      foreach var in b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM
b2_FPFM t_b2_FPFM cons_FPFM t_cons_FPFM R2_FPFM R2i_FM {
            format `var' %9.2f
      }

      export excel using "[your partition/file name]", firstrow(varlabels)  nolabel
replace
      restore

      ****************************************************
      * Single-season regression models with DRS
      ****************************************************
      * Loop over each year
      foreach y of numlist 2003/2024 {
            di "Processing year `y'"
            * Model 2: RA = b1' * FIP + b2_FPFM * fb_drs
```

```
            quietly: regress RA FIP fb_drs if year == `y'
            replace R2_FPFM = e(r2) if year == `y'
            replace R2i_FM   = e(r2) - R2_FAM if year == `y'
            replace b1_FPFM  = _b[FIP] if year == `y'
            replace b2_FPFM        = _b[fb_drs] if year == `y'
            replace t_b1_FPFM = _b[FIP] / _se[FIP] if year == `y'
            replace t_b2_FPFM       = _b[fb_drs] / _se[fb_drs] if year == `y'
            replace cons_FPFM   = _b[_cons] if year == `y'
            replace t_cons_FPFM   = _b[_cons] / _se[_cons] if year == `y'
      }

* Display the updated dataset

preserve

drop if year <2003

collapse b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM b2_FPFM t_b2_FPFM
cons_FPFM t_cons_FPFM R2_FPFM R2i_FM, by (season)
      foreach var in b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM
b2_FPFM t_b2_FPFM cons_FPFM t_cons_FPFM R2_FPFM R2i_FM {
            format `var' %9.2f
      }

      export excel using "[your partition/file name]", firstrow(varlabels)  nolabel
replace

      restore

      ****************************************************
      * Single-season regression models with sc_fld
      ****************************************************

foreach y of numlist 2016/2024 {
            di "Processing year `y' - sc_fld"

            * Model 2: RA = b1' * FIP + b2_FPFM * sc_fld
            quietly regress RA FIP sc_fld if year == `y'
            replace R2_FPFM = e(r2) if year == `y'
            replace R2i_FM   = e(r2) - R2_FAM if year == `y'
            replace b1_FPFM  = _b[FIP] if year == `y'
            replace t_b1_FPFM = _b[FIP] / _se[FIP] if year == `y'
            replace b2_FPFM        = _b[sc_fld] if year == `y'
            replace t_b2_FPFM      = _b[sc_fld] / _se[sc_fld] if year == `y'
            replace cons_FPFM  = _b[_cons] if year == `y'
            replace t_cons_FPFM  = _b[_cons] / _se[_cons] if year == `y'
      }

preserve

      drop if year <2016

collapse b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM b2_FPFM t_b2_FPFM
cons_FPFM t_cons_FPFM R2_FPFM R2i_FM , by (season)
      foreach var in b_FAM t_b_FAM cons_FAM t_cons_FAM R2_FAM  b1_FPFM t_b1_FPFM
b2_FPFM t_b2_FPFM cons_FPFM t_cons_FPFM R2_FPFM R2i_FM {
            format `var' %9.2f
      }

      export excel using "[your partition/file name]", firstrow(varlabels)  nolabel
replace
      restore
```

```
	**************************************************
	* Incremental fielding R2s
	**************************************************

	gen gen_frs=.

	// composite tzr/drs/sc_fld fielding runs saved variable

	replace gen_frs=rfield if year <1990 |  year >1999 & year < 2003
		replace gen_frs=tfr if year >1989 & year <2000
		replace gen_frs=rfield if year >1999 & year <2003
		replace gen_frs=fb_drs if year >2002

	gen R21 = .
	gen R22 = .
	foreach yr of numlist 1900/2024 { // Adjust the range to your data
		// Run the regression with z_FIP only for the current year

		di `yr'
		regress RA FIP if year == `yr'
		// Store R2 from the first model in R21 for the current year
		local r2_1 = e(r2)
		replace R21 = `r2_1' if year == `yr'
		// Run the regression with z_FIP and z_rfield for the current year
		regress RA FIP gen_frs if year == `yr'
		// Store R2 from the second model in R22 for the current year
		local r2_2 = e(r2)
		replace R22 = `r2_2' if year == `yr'
	}

preserve

	* Collapse to get mean R2 by year
	collapse (mean) R21 R22, by(year)



	* convert dataset to time series based on year
	sort year
	tsset year
	*
	* Multiply the y-axis variables by 100 for % scaling

	gen R21_scaled=R21*100
	gen R22_scaled=R22*100

	* Plot figure 2

	* Generate smoothed values for R22 using lpoly with specified points
		lpoly R22_scaled year, gen(R22_smooth) bw(2) at(year)

			* Plot with gray-filled area for R22 and dashed line for R21
		twoway (area R22_smooth year, color(gs12%50)) ///  // Gray-filled area
for smoothed R22
		       (lpoly R21_scaled year, lcolor(black) lpattern(longdash)
lwidth(medium) bw(5)), ///  // Black dashed lpoly line for smoothed R21
		       ytitle("") ///
		       xtitle("") ///
		       ylabel(0(10)100, labsize(medium) nogrid) ///
		       xlabel(1900(10)2024, angle(45) labsize(medium) nogrid) ///
		       legend(off) ///
		       graphregion(color(white)) ///
		       plotregion(margin(zero))
```

```
                graph export "R_squared_trends2.pdf", replace

restore

drop R21 R22


       ***************************************************
       * fielding/FIP variance
       ***************************************************

 bysort yearID: egen sd_frs  = sd(gen_frs) // sd composite team fielding variable


 preserve


drop if year ==2020 // COVID-shortened season variances misleadingly inflated

    twoway (scatter sd_frs year, mcolor(ltblue)) (lpoly sd_frs year, bw(3)
lcolor(blue) lwidth(medium)), ///
        xlabel(1900(10)2024, nogrid format(%02.0f) angle(45)) ///
        ylabel(, nogrid) ytitle("SD") xtitle("year") ///
        graphregion(color(white)) plotregion(style(none)) ///
        xscale(r(1900 2024)) legend(off) aspect(0.6)
       graph  save    sd_frs.gph, replace


    twoway (scatter sd_FIP year, mcolor(ltblue)) (lpoly sd_FIP year, bw(3)
lcolor(blue) lwidth(medium)), ///
        xlabel(1900(10)2024, nogrid format(%02.0f) angle(45)) ///
        ylabel(, nogrid) ytitle(" ") xtitle(" ") ///
        graphregion(color(white)) plotregion(style(none)) ///
        xscale(r(1900 2024)) legend(off) aspect(0.6)
             graph  save    sd_fip.gph, replace


       graph combine  sd_frs.gph sd_FIP.gph, col(2) iscale(1) xsize(9) ysize(4)


restore

    ***************************************************
    * multi-season regression models post 2002
    ***************************************************

* calculate post-2002 incremental R^2s

gen Rfip = .
gen Ri_fb = .
gen Ri_fg = .
gen Ri_uzr= .
gen Ri_tfr =.
gen Ri_sc_fld=.



foreach yr of numlist 2003/2024 {
    // Run the regression with FIP only for the current year
    regress RA FIP if year == `yr'
    // Store R2 from the first model in R21 for the current year
```

```
    local r2_1 = e(r2)
    replace Rfip = `r2_1' if year == `yr'
    // Run the regression with FIP and fb for the current year
    regress RA FIP fb_drs if year == `yr'
    // Store incremental R2 from the second model
    local r2_2 = e(r2)
    replace Ri_fb = `r2_2'-Rfip if year == `yr'


       // Run the regression with FIP and fg for the current year
    regress RA FIP fgfield if year == `yr'

        // Store incemental R2 from the third model for the current year
    local r2_3 = e(r2)
    replace Ri_fg = `r2_3'-Rfip if year == `yr'


       // Run the regression with FIP and fg for the current year
    regress RA FIP uzr if year == `yr'

        // Store incemental R2 from the third model for the current year
    local r2_4 = e(r2)
    replace Ri_uzr = `r2_4'-Rfip if year == `yr'


       // Run the regression with FIP and fg for the current year
    regress RA FIP tfr if year == `yr'

        // Store incemental R2 from the third model for the current year
    local r2_5 = e(r2)
    replace Ri_tfr = `r2_5'-Rfip if year == `yr'

}

foreach yr of numlist 2016/2024 {


    // Run the regression with FIP and statcast for the current year
    regress RA FIP sc_fld if year == `yr'
    // Store incremental R2 from the second model
    local r2_2 = e(r2)
    replace Ri_sc_fld = `r2_2'-Rfip if year == `yr'
}

preserve

collapse (mean) Rfip Ri_fb Ri_fg Ri_uzr Ri_tfr Ri_sc_fld,by(yearID)

drop if year <2003

* Create cumulative variables for ribbon placement
gen cum_fip = Rfip*100
gen cum_drs = (cum_fip)+Ri_fb*100
gen cum_fg = (cum_fip) + Ri_fg*100
gen cum_uzr = (cum_fip) + Ri_uzr*100
gen cum_tfr = (cum_fip) + Ri_tfr*100
gen cum_sc_fld = (cum_fip) + Ri_sc_fld*100
* Smooth the data for each cumulative variable
foreach lname in fip drs fg uzr tfr sc_fld  {
    lpoly cum_`lname' year,bw(1.6) gen(smooth_`lname')  at(year)
}

* Create a variable for the lower bound
```

```
gen zero = 0

twoway ///
    rarea smooth_fip zero year, color(gs12%60) alcolor(gs12%01) || /// Smoothed Base
FIP
    rarea smooth_tfr smooth_fip year, color(yellow%80) alcolor(yellow%01)  || ///
Smoothed Increment
    rarea smooth_drs smooth_fip year, color(red%55) alcolor(blue%01) || /// Smoothed
Increment
       rarea smooth_uzr smooth_fip year, color(blue%40) alcolor(red%01) || ///
Smoothed Increment
       rarea smooth_sc_fld smooth_fip year if year >2015, color(magenta%30)
alcolor(magenta%01) || /// Smoothed Increment
    , legend(order(1 "" 2 "" 3 "" 4 "" 5 "")) ///
      title(" ") ///
      xlabel(2003(3)2024, nogrid angle(45)format(%ty)) ///
      ylabel(0(10)100, nogrid angle(0)) ///
      ytitle("R² Value") ///
      xtitle("Year")



restore

/// post 2002 explanatory power


hireg z_rapg (z_FIP) (z_fb_drs)  if year  > 2002
hireg z_rapg (z_FIP) (z_uzr)  if year  > 2002
hireg z_rapg (z_FIP) (z_tfr) if year  > 2002


hireg z_rapg (z_FIP) (z_fb_drs)  if year  > 2014
hireg z_rapg (z_FIP) (z_uzr)  if year  > 2014
hireg z_rapg (z_FIP) (z_tfr) if year  > 2014
hireg z_rapg (z_FIP) (z_sc_fld) if year  > 2015



**************************************************
* run saved inflation
**************************************************

*** save unit value of refield for each year

gen beta_rfield = .
gen beta_drs = .
gen beta_fgrf=.
gen beta_uzr=.
gen beta_frs=.
gen beta_tfr=.
gen beta_sc_fld=.
gen beta_def=.
gen beta_FIP=.

foreach yr of numlist 1900/2024 { //
    // Run the regression with FIP and rfield only for the current year
    quietly regress RA FIP rfield if year == `yr'

    // Store the betas in the current year
    local b_rfield = _b[rfield]
    local b_FIP = _b[FIP]
```

```
    // Replace stored betas with betas for observations in the current year
    replace beta_rfield = `b_rfield' if year == `yr'
    replace beta_FIP = `b_FIP' if year == `yr'


     // Run the regression with FIP and fgfield only for the current year
    quietly regress RA FIP fgfield if year == `yr'


    // Replace beta_ for observations in the current year
    replace beta_fgrf = _b[fgfield] if year == `yr'

}


*** save unit value of tfr for each year

foreach yr of numlist 1920/2024 {



   // Run the regression with FIP and rfield only for the current year
    quietly regress  RA FIP tfr if year == `yr'


    // Replace beta_ for observations in the current year
    replace beta_tfr = _b[tfr] if year == `yr'

}

*** save unit value of uzr for each year

foreach yr of numlist 2003/2024 {




   // Run the regression with FIP and rfield only for the current year
    quietly regress  RA FIP uzr if year == `yr'


    // Replace beta_ for observations in the current year
    replace beta_uzr = _b[uzr] if year == `yr'

}
*** save unit value of drs for each year

foreach yr of numlist 2003/2024 {




   // Run the regression with FIP and rfield only for the current year
    quietly regress  RA FIP fb_drs if year == `yr'


    // Replace beta_ for observations in the current year
    replace beta_drs = _b[fb_drs] if year == `yr'

}

foreach yr of numlist 2016/2024 {
```

```
   // Run the regression with FIP and rfield only for the current year
    regress  RA FIP sc_fld if year == `yr'


    // Replace beta_ for observations in the current year
    replace beta_sc_fld = _b[sc_fld] if year == `yr'

}

*** save unit value of def for each year

foreach yr of numlist 2003/2024 {




   // Run the regression with FIP and rfield only for the current year
    quietly regress  RA FIP  def if year == `yr'


    // Replace beta_ for observations in the current year
    replace beta_def = _b[ def] if year == `yr'

}

*** plot

** change sign of betas  to reflect runs saved
gen beta_rfield_neg= beta_rfield*-1
gen beta_fgrf_neg= beta_fgrf*-1
gen beta_drs_neg= beta_drs*-1
gen beta_uzr_neg= beta_uzr*-1
gen beta_def_neg=beta_def*-1
gen beta_tfr_neg=beta_tfr*-1
gen beta_sc_fld_neg=beta_sc_fld*-1

su beta_rfield if year >2014
su beta_fgrf if year > 2014
su beta_drs if year >2014
su beta_uzr if year > 2014
su beta_tfr if year > 2014
su beta_sc_fld if year > 2015
su beta_def if year >2002

gen beta_fg_comp=beta_fgrf_neg

replace beta_fg_comp=beta_def_neg if year >2002

       *** Figure 5 ****

       twoway ///
                 (lpolyci beta_rfield_neg year if year >1950,  bw(2)fcolor(gs12%36)
alcolor(gs12%10) clstyle(none)) ///
                 (lpoly beta_rfield_neg year if year >1950, bw(2) lcolor(black)
lpattern(dash)), ///
                ylabel(, nogrid) xlabel(1950(10)2024, angle(45) nogrid) ///
                title("") xtitle("") ///
                legend(off)
```

```
       *** Figure 6 ****


             twoway ///
       (lpolyci beta_tfr_neg year if year >2002, bw(1.3) fcolor(dkorange%50)
alcolor(dkorange%1) clstyle(none)) ///
       (lpoly beta_tfr_neg year if year >2002, bw(1.3)  lcolor(dkorange)
lpattern(dash)) ///
       (lpolyci beta_uzr_neg year if year >2002,bw(1.3)   fcolor(cranberry%50)
alcolor(cranberry%1) clstyle(none)) ///
       (lpoly beta_uzr_neg year if year >2002, bw(1.3)  lcolor(cranberry)
lpattern(dash)) ///
       (lpolyci beta_drs_neg year if year >2002, bw(1.3)  fcolor(green%50)
alcolor(green%1) clstyle(none)) ///
         (lpolyci beta_sc_fld_neg year if year>2015, bw(1.3)  fcolor(magenta%50)
alcolor(magenta%1) clstyle(none)) ///
         (lpoly beta_sc_fld_neg year if year >2015, bw(1.3)  lcolor(magenta)
lpattern(dash)) ///
       (lpoly beta_drs_neg year if year >2002,bw(1.3)   lcolor(green) lpattern(dash)),
///
       ylabel(, nogrid) xlabel(2003(3)2024, angle(45) nogrid) ///
       title("") xtitle("") ///
       legend(off)


**************************************************
* what if analyses
**************************************************



** use clarify MC to compute difference in runs avoided for Red Sox and Twins '67
set seed 19632024

       estsimp regress RA FIP rfield  if  year >1964 & year <1968

       setx mean
       simqi, fd(ev) changex(rfield 37 -28 )
       simqi, fd(ev) changex( FIP 3.5 2.97)
       simqi, fd(ev) changex(rfield 37 -28 FIP 3.5 2.97)

gen deltarf= b2*(37+28)
             sum deltarf,d


gen deltafip=b1*(3.5-2.97)
sum deltafip
gen deltat1=deltarf+deltafip
sum deltat1
drop deltafip deltarf


drop b*

estsimp regress RA FIP rfield  if  year >2020


       setx mean
       simqi, fd(ev) changex(rfield 37 -28 FIP 3.50 2.97)

       gen deltarf= b2*(37+28)
              sum deltarf,d
```

```
gen deltafip=b1*(3.5-2.97)
sum deltafip
gen deltat2=deltarf+deltafip
sum deltat2

       gen total_delta=-1*(deltat2-deltat1)

       sum total_delta,d


       _pctile total_delta, percentiles(2.5 97.5)
di "2.5th percentile: `r(r1)'"
di "97.5th percentile: `r(r2)'"

histogram total_delta, bcolor(gray) lcolor(black) ///
    yscale(off) ylabel(,nogrid) xtitle("") xlabel(-70(10)0, labsize(large)nogrid)  ///
    ytitle("")

drop b*

drop deltat1 deltarf deltafip deltat2 total_delta

** use clarify MC to compute difference in runs avoided for Indians & Red Sox '48

estsimp regress RA FIP rfield  if year >1944 & year <1949
setx mean
simqi, fd(ev) changex(rfield 24 84)
       simqi, fd(ev) changex( FIP  4.13 4.04)
       simqi, fd(ev) changex(rfield 24 84 FIP  4.13 4.04)

gen deltarf= b2*(84-24)
             sum deltarf,d


gen deltafip=b1*(4.04-4.13)
sum deltafip
gen deltat1=deltarf+deltafip
sum deltat1
drop deltafip deltarf


drop b*


estsimp regress RA FIP rfield  if year >2020
setx mean
       simqi, fd(ev) changex(rfield 84 24)
       simqi, fd(ev) changex( FIP  4.04 413)
       simqi, fd(ev) changex(rfield 84 25 FIP  4.04 4.13)

gen deltarf= b2*(84-24)
             sum deltarf,d


gen deltafip=b1*(4.04-4.13)
sum deltafip
gen deltat2=deltarf+deltafip
sum deltat2

       gen total_delta=-1*(deltat2-deltat1)

       sum total_delta
```

```
       histogram total_delta, bcolor(gray) lcolor(black) ///
              yscale(off) ylabel(,nogrid) xtitle("")  xlabel(-80(10)-10, labsize(large)
nogrid) ///
              ytitle("")


                     _pctile total_delta, percentiles(2.5 97.5)
di "2.5th percentile: `r(r1)'"
di "97.5th percentile: `r(r2)'"


drop b*

drop deltat1 deltarf deltafip deltat2 total_delta

** use clarify MC to compute difference in runs avoided for Giants & Dodgers 62
estsimp regress RA FIP rfield   if year >1958 & year < 1963
      simqi, fd(ev) changex( rfield  30 -14)
       simqi, fd(ev) changex( FIP   3.81 3.39)
       simqi, fd(ev) changex(rfield  30 -14 FIP 3.81 3.39)

gen deltarf= b2*(30+14)
              sum deltarf,d


gen deltafip=b1*(3.81-3.39)
sum deltafip
gen deltat1=deltarf+deltafip
sum deltat1
drop deltafip deltarf


drop b*


estsimp regress RA FIP rfield   if year >2020

setx mean
simqi, fd(ev) changex(rfield 30 -14)
       simqi, fd(ev) changex( FIP   3.81 3.39)
       simqi, fd(ev) changex(rfield  30 -14 FIP 3.81 3.39)

gen deltarf= b2*(30+14)
              sum deltarf,d


gen deltafip=b1*(3.81-3.39)
sum deltafip
gen deltat2=deltarf+deltafip
sum deltat2

       gen total_delta=-1*(deltat2-deltat1)

       sum total_delta
histogram total_delta, bcolor(gray) lcolor(black)  ///
    yscale(off) ylabel(,nogrid) xtitle("") xlabel(-60(10)-10, labsize(large) nogrid)
///
    ytitle("")


                     _pctile total_delta, percentiles(2.5 97.5)
di "2.5th percentile: `r(r1)'"
di "97.5th percentile: `r(r2)'"
```

```
drop b*
drop deltat1 deltarf deltafip deltat2 total_delta
```

B. SI DRA analyses

1. Variabless

Note that the table identifies and describes the variables contained in the data file. The analyses use these variables to form additional composite variables, the identity and nature of which is described fully in the analysis script annotations.

Variable name	Values	Description	Derivation
Season	1920-1948	Negro League seasons	Baseball Reference
Team	string team identifiers	Negro League teams	Baseball Reference
lg	ANL=American Negro League ECL=Eastern Colored Leagu EWL=East/West League NAL=Negro American League I NN2=Negro American League II NNL=Negro National	Negro Leagues	Baseball Reference
rfield	fielding runs	rfield measure of individual team "fielding runs	Baseball Reference
gp	games played	games played in season	Baseball Reference
RA	runs allowed	total runs allowed season	Baseball Reference
FIP	FIP scores	team fielding-independent pitching score	Baseball Reference

2. Analyses script

```
clear
clear matrix
set matsize 800
set mem 500m
cd "[your partition]"
use [jsa_dra_negro_league.csv]

drop if gp<25 /// confine to meaningful sample

// runs allowed per game
gen rapg=RA/gp

//generate z-scores for values of interest

bysort Season lg: egen mean_rfield = mean(rfield)
bysort Season lg: egen sd_rfield = sd(rfield)

bysort Season lg: egen mean_FIP = mean(FIP)
bysort Season lg: egen sd_FIP = sd(FIP)
```

```
bysort Season lg: egen mean_rapg = mean(rapg)
bysort Season lg: egen sd_rapg = sd(rapg)

gen z_rfield = (rfield - mean_rfield) / sd_rfield
gen z_FIP = (FIP - mean_FIP) / sd_FIP
gen z_rapg = (rapg - mean_rapg) / sd_rapg

drop if mean_rfield ==0 // BBREF assigns "0" for missing--not a good idea...

//regression analyses

hireg z_rapg (z_FIP) (z_rfield) if lg =="NNL"
hireg z_rapg (z_FIP) (z_rfield)  if lg =="NN2"
hireg z_rapg (z_FIP) (z_rfield) if lg !="NN2" & lg !="NN2"
```