# Additional File 1: analysis code

The following sections contain Stata 15 do files that analyse the data sets used in the study “Comparison of six statistical methods for interrupted time series studies: empirical evaluation of 190 published series”

Simon L Turner1, Amalia Karahalios1, Andrew B Forbes1, Monica Taljaard2,3, Jeremy M Grimshaw2,3,4, Joanne E McKenzie1\*

1School of Public Health and Preventive Medicine, Monash University, Melbourne, Victoria, Australia.

2Clinical Epidemiology Program, Ottawa Hospital Research Institute, Ottawa, Ontario, Canada. 1053 Carling Ave, Ottawa.

3School of Epidemiology and Public Health, University of Ottawa, Ottawa, Ontario, Canada. 600 Peter Morand Crescent, Ottawa, Ontario K1G 5Z3.

4Department of Medicine, University of Ottawa, Ottawa, Ontario, Canada. Roger Guindon Hall, 451 Smyth Rd.

The following do files are required:

|  |  |
| --- | --- |
| Name of file | Purpose |
| 001\_master\_empirical\_evaluation.do | Contains instructions on the running of the analysis and calls the subsequent do files |
| empirical\_all\_methods\_published.do | Analyses each data series using a range of statistical methods and saves the results of each method in a separate .dta file |
| combine\_methods\_published.do | Combines the results from each statistical method into a single .dta file |
| get\_wanted\_empirical\_estimates\_published.do | The final data required does not include segments such as those used for transition periods, this do file selects the effect estimates used for the final summaries |

* 1. Data file 1: 001\_master\_empirical\_evaluation.do

////////////////////////////////////////////////////////////////////////////////

// Master Empirical Evaluation final file

// Simon Turner

// For the study:

// "Comparison of six statistical methods for interrupted time series studies: empirical evaluation of 190 published series"

// Turner SL, Karahalios A, Forbes AB, Taljaard M, Grimshaw JM, McKenzie JE.

//

// This do file calls multiple other do files to:

//

// analyse each dataset using multiple statistical methods (empirical\_all\_methods\_published.do)

// save level changes, slope changes, SEs, p-values and autocorrelation estimates

// combine all the results from the different methods (combine\_methods\_published.do)

// and choose the estimate wanted for analysis (get\_wanted\_empirical\_estimates\_published.do)

// the final file used for the manuscript (including graphs) is empirical\_estimates.dta

//

// required data are found in the two excel files:

// STurner\_Empirical\_study\_information.xls

// which contains information about the studies

// also

// STurner\_Empirical\_time\_series.xls

// which contains the actual time series data

//

// further details about the variables can be found in:

// STurner\_Empirical\_Data\_Dictionary.xls

////////////////////////////////////////////////////////////////////////////////

// firstly set the working directory...

global dir "<set directory>"

cd "$dir"

version 15

////////////////////////////////////////////////////////////////////////////////

// analyse the datasets

// now use different methods and capture the outputs

do ${dir}empirical\_all\_methods\_published

// combine the results from the different methods

do ${dir}combine\_methods\_published

// for the empirical study we only want to use a subset of the total datasets

// this next section obtains the wanted datasets

do ${dir}get\_wanted\_empirical\_estimates\_published

////////////////////////////////////////////////////////////////////////////////

// The final file we can use for graphing, tables etc. is called:

// empirical\_estimates\_published.dta

////////////////////////////////////////////////////////////////////////////////

* 1. Data file 2: empirical\_all\_methods\_published.do

////////////////////////////////////////////////////////////////////////////////

// This is the analysis file for the empirical study:

// "Comparison of six statistical methods for interrupted time series studies: empirical evaluation of 190 published series"

// loads the datasets,

// analyses each using a variety of statistical methods

// finally it saves the analysed data in "estimates\_published\_`model\_type'"

// where `model\_type' is one of the statistical methods used

// load the data that is ready for analysis

import excel "STurner\_Empirical\_time\_series.xls", sheet("Sheet1") firstrow clear

// ensure that there is a proper study id

levelsof series\_id, local(series\_ids)

// there are several alternatives for scaling

// no scaling (just leave this blank)

// scaling by the rmse for the first segment only, use "rmse\_on"

// or scaling by the rmse of the whole series, use "rmse\_full"

// some of the datasets are very short (three points in pre-series) so first segment scaling does not work

// scaling by the full rmse was used for the analysis

local rmse\_full = "rmse\_full"

////////////////////////////////////////////////////////////////////////////////

// this short program estimates autocorrelation from the residuals after a

// simple linear regression

// this was not used in the analysis, but served as an interesting reference

cap program drop find\_rho

program find\_rho

syntax varlist [if]

marksample touse

quietly regress `varlist' l.`varlist' if `touse' , nocons

local rho=\_b[L.`varlist']

scalar rho = `rho'

end

cap program drop find\_rho\_wrap

program find\_rho\_wrap

syntax varlist [if]

 marksample touse

 tempvar resid

 regress `varlist' if `touse'

 predict double `resid', resid

 find\_rho `resid' if `touse'

end

////////////////////////////////////////////////////////////////////////////////

// we investigated a range of statistical methods

// this string links to the sections below to identify which methods

// are going to be used here

// regress - OLS

// newey - OLS with newey-west standard errors

// prais - prais-winsten with an iterative search

// prais\_raw - simple prais-winsten

// corc - cochrane-orcutt

// mixed - REML

// mixed\_satt - REML with the Satterthwaite small series adjustment

// arima - ARIMA with lag-1

// for the final study we restricted these to the following...

local model\_types "regress newey prais\_raw mixed mixed\_satt arima"

// loop over each statistical method

foreach model\_type of local model\_types {

 // set the end file name

 local save\_name "estimates\_published\_`model\_type'.dta"

 // set up the temporary file used to store the data

 // we are saving data per segment

 // the level change and slope change with CIs, SEs and p-values

 // autocorrelation estimates, degrees of freedom etc.

 tempname post\_values\_`model\_type'

 postfile `post\_values\_`model\_type'' series\_id ///

 str20 model\_type ///

 segment analysis\_autocorr analysis\_effects segment\_num\_points total\_num\_points rmse ///

 level level\_ll level\_ul level\_se level\_p ///

 slope slope\_ll slope\_ul slope\_se slope\_p ///

 rho\_est rho\_cil rho\_ciu ///

 num\_iterations error\_code converged ///

 lincom\_level\_dof lincom\_slope\_dof ///

 using "`save\_name'" , replace

 // now for each of the data sets...

 foreach series\_id of local series\_ids {

 // going to just use one at a time

 preserve

 keep if series\_id == `series\_id'

 local series\_id = series\_id[1]

 display "working through `series\_id'"

 ////////////////////////////////////////////////////////////////////////////

 // set for program

 \*keep outcome time segment segment\_in\_analysis

 sort time

 drop if time == .

 ////////////////////////////////////////////////////////////////////////////////

 // find times programatically

 // this goes through and works out the timing of each segment

 summ segment

 local num\_segments = r(max)

 local min\_seg\_num = r(min)

 if `min\_seg\_num' != 0 {

 replace segment = segment - `min\_seg\_num'

 }

 summ segment

 local num\_segments = r(max)

 local min\_seg\_num = r(min)

 forvalues segment = 0/`num\_segments' {

 summ time if segment == `segment'

 local time\_`segment'\_start = r(min)

 local time\_`segment'\_end = r(max)

 display "segment `segment' goes from `time\_`segment'\_start' to `time\_`segment'\_end'"

 }

 ////////////////////////////////////////////////////////////////////////////////

 // extra variables for analysis

 // generate variables to indicate time of intervention

 forvalues segment = 0/`num\_segments' {

 gen intervention\_`segment' = 0

 replace intervention\_`segment' = 1 if segment >= `segment'

 gen level\_change\_`segment' = intervention\_`segment'

 gen slope\_change\_`segment' = (time-`time\_`segment'\_start')\*level\_change\_`segment'

 }

 ////////////////////////////////////////////////////////////////////////////////

 // model

 ////////////////////////////////////////////////////////////////////////////////

 // create the variables to use (the various level and slope changes)

 // starting with level\_0 and slope\_0, then incrementing for each segment

 // e.g. regular segmented regression will have

 // level\_0 (intercept) slope\_0 (pre-interruption slope) level\_1 (level change at interruption) slope\_1 (slope change post interruption)

 local variables = ""

 forvalues segment = 0/`num\_segments' {

 local variables = "`variables'" + " level\_change\_`segment' slope\_change\_`segment'"

 }

 display "variables: `variables'"

 display "study: `year' number: `number\_in\_year' multiple: `multiple' data\_type: `data\_type'"

 // first find rmse from all segments (for scaling)

 regress outcome `variables', nocons

 local rmse = e(rmse)

 local total\_num\_points = e(N)

 ///////////////////////////////////////////

 // now apply the correct statistical method according to model\_type...

 tsset time

 local rho\_est = .

 local rho\_est\_cil = .

 local rho\_est\_ciu = .

 local model\_error = 0

 if "`model\_type'" == "regress" { // basic OLS regression

 regress outcome `variables', nocons

 matrix local\_results = r(table)

 local rho\_est = 0

 local num\_iterations = 1

 local error\_code = \_rc

 local converged = 1

 } // end regress check

 else if "`model\_type'" == "newey" { // OLS regression with newey-west standard errors

 newey outcome `variables', nocons lag(1) force // need force option to ignore missing values otherwise get time not equally spaced errors

 matrix local\_results = r(table)

 local varlist = "outcome `variables'"

 quietly find\_rho\_wrap `varlist'

 local rho\_est = rho

 local num\_iterations = 1

 local error\_code = \_rc

 local converged = 1

 newey outcome `variables', nocons lag(1) force // need force option to ignore missing values otherwise get time not equally spaced errors

 } // end newey check

 else if "`model\_type'" == "prais" { // Prais-Winsten with iterative search

 cap prais outcome `variables', nocons ssesearch

 if \_rc != 0 {

 local model\_error = \_rc

 matrix local\_results = J(6,5,.)

 local rho\_est = .

 local converged = .

 local num\_iterations = e(ic)

 local error\_code = \_rc

 }

 else {

 matrix local\_results = r(table)

 local rho\_est = e(rho)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 local converged = 1

 }

 } // end prais check

 else if "`model\_type'" == "prais\_raw" { // standard Prais-Winsten

 cap prais outcome `variables', nocons

 if \_rc != 0 {

 local model\_error = \_rc

 matrix local\_results = J(6,5,.)

 local rho\_est = .

 local converged = .

 local num\_iterations = e(ic)

 local error\_code = \_rc

 }

 else {

 matrix local\_results = r(table)

 local rho\_est = e(rho)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 local converged = 1

 }

 } // end prais\_raw check

 else if "`model\_type'" == "corc" { // Cochrane-Orcutt

 cap prais outcome `variables', nocons corc ssesearch

 if \_rc != 0 {

 local model\_error = \_rc

 matrix local\_results = J(6,5,.)

 local rho\_est = .

 local converged = .

 local num\_iterations = e(ic)

 local error\_code = \_rc

 }

 else {

 matrix local\_results = r(table)

 local rho\_est = e(rho)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 local converged = 1

 }

 } // end corc check

 else if "`model\_type'" == "mixed" { // REML with maximum iterations set to 1000 to stop really long non-convergence

 cap mixed outcome `variables', nocons res(ar 1, t(time)) var reml iter(1000)

 if \_rc != 0 {

 local model\_error = \_rc

 matrix local\_results = J(6,5,.)

 local rho\_est = .

 local converged = e(converged)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 }

 else {

 matrix local\_results = r(table)

 local num\_cols = colsof(local\_results)

 local rho\_est = tanh(local\_results[1,`num\_cols'])

 local rho\_est\_cil = tanh(local\_results[5,`num\_cols'])

 local rho\_est\_ciu = tanh(local\_results[6,`num\_cols'])

 local converged = e(converged)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 }

 } // end mixed check

 else if "`model\_type'" == "mixed\_kr" { // REML with KR adjustment

 cap mixed outcome `variables', nocons res(ar 1, t(time)) var reml iter(1000) dfmethod(kr)

 if \_rc != 0 {

 local model\_error = \_rc

 matrix local\_results = J(6,5,.)

 local rho\_est = .

 local converged = e(converged)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 }

 else {

 matrix local\_results = r(table)

 local num\_cols = colsof(local\_results)

 local rho\_est = tanh(local\_results[1,`num\_cols'])

 local rho\_est\_cil = tanh(local\_results[5,`num\_cols'])

 local rho\_est\_ciu = tanh(local\_results[6,`num\_cols'])

 local converged = e(converged)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 mat dfs = e(df)

 }

 } // end mixed\_kr check

 else if "`model\_type'" == "mixed\_satt" { // REML with Satt adjustment

 cap mixed outcome `variables', nocons res(ar 1, t(time)) var reml iter(1000) dfmethod(satt)

 if \_rc != 0 {

 local model\_error = \_rc

 matrix local\_results = J(6,5,.)

 local rho\_est = .

 local converged = e(converged)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 }

 else {

 display "`model\_type' model ran"

 matrix local\_results = r(table)

 local num\_cols = colsof(local\_results)

 local rho\_est = tanh(local\_results[1,`num\_cols'])

 local rho\_est\_cil = tanh(local\_results[5,`num\_cols'])

 local rho\_est\_ciu = tanh(local\_results[6,`num\_cols'])

 local converged = e(converged)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 mat dfs = e(df)

 }

 } // end mixed\_satt check

 else if "`model\_type'" == "arima" {

 cap arima outcome `variables', nocons collinear ar(1) iter(1000)

 if \_rc != 0 {

 // new section for error catch includes a "slow down" as for some reason

 // Stata sometimes crashes if it went too quickly here (Stata 15.0)

 local model\_error = \_rc

 local iteration = 1

 while `iteration' < 10 & \_rc != 0 {

 cap matrix local\_results = J(6,5,.)

 local rho\_est = .

 local converged = e(converged)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 sleep 10

 local iteration = `iteration' + 1

 }

 }

 else {

 matrix local\_results = r(table)

 local num\_cols = colsof(local\_results)

 local rho\_col = `num\_cols' - 1

 local rho\_est = (local\_results[1,`rho\_col'])

 local rho\_est\_cil = (local\_results[5,`rho\_col'])

 local rho\_est\_ciu = (local\_results[6,`rho\_col'])

 local converged = e(converged)

 local num\_iterations = e(ic)

 local error\_code = \_rc

 }

 } // end arima check

 else {

 display "Model type `model\_type' is not recognised"

 stop

 }

 display "study: `year' number: `number\_in\_year' multiple: `multiple' data\_type: `data\_type' model `model\_type' ran with error code `model\_error'"

 // if there was an error just set the output to missing values

 if `model\_error' != 0 {

 forvalues segment = 0/`num\_segments' {

 local level\_change\_counter\_`segment' = .

 local level\_change\_counter\_`segment'\_ll = .

 local level\_change\_counter\_`segment'\_ul = .

 local level\_change\_counter\_`segment'\_se = .

 local level\_change\_counter\_`segment'\_p = .

 local slope\_change\_counter\_`segment' = .

 local slope\_change\_counter\_`segment'\_ll = .

 local slope\_change\_counter\_`segment'\_ul = .

 local slope\_change\_counter\_`segment'\_se = .

 local slope\_change\_counter\_`segment'\_p = .

 }

 }

 else {

 // use lincom to find the CIs that go with the various level and slope changes

 // predict the estimates to find the counterfactual, level change etc.

 // base estimates of level change compared to first segment

 predict estimates

 gen counterfactual = \_b[level\_change\_0] + \_b[slope\_change\_0]\*slope\_change\_0

 // if satterthwaite is used need to add ",small" to the options after lincom

 if "`model\_type'" == "mixed\_satt" {

 local small " , small"

 }

 else {

 local small ""

 }

 ////////////////////////////////////////////////////////////////////////////////

 // level changes from counterfactual

 // for each segment...

 forvalues segment = 0/`num\_segments' {

 // for the first segment just initialise everything

 if `segment' == 0 {

 local level\_change\_counter\_`segment' = .

 local level\_change\_counter\_`segment'\_ll = .

 local level\_change\_counter\_`segment'\_ul = .

 local level\_change\_counter\_`segment'\_se = .

 local level\_change\_counter\_`segment'\_p = .

 }

 else {

 // for each subsequent segment

 // set up a local macro that holds the names of all the lincom variables we want

 // this is going to be

 // level\_change\_1 for the first level change

 // and then adding on the subsequent level and slope change values for any subsequent segments

 forvalues segment\_sub = 1/`segment' {

 if `segment\_sub' == 1 {

 local L`segment' = `" level\_change\_`segment\_sub' "'

 }

 else {

 local segment\_multiplier = slope\_change\_`=`segment\_sub'-1'[`time\_`segment'\_start']

 local L`segment' = `" `L`segment'' "' + `" + slope\_change\_`=`segment\_sub'-1'\*`segment\_multiplier' + level\_change\_`segment\_sub' "'

 }

 \*display "L`segment' `L`segment''"

 }

 // now lincom those variables (adding the small option if using REML-Satt)

 lincom `L`segment'' `small'

 // lincom gives slightly different responses in different situations...

 if "`model\_type'" == "arima" | "`model\_type'" == "mixed" {

 local lincom\_level\_dof = e(N) - e(df\_m)

 }

 else {

 local lincom\_level\_dof = r(df)

 }

 \*return list

 display "level lincom dof = `lincom\_level\_dof'"

 // we are using a cut-off of 2 for the degrees of freedom for the REML-Satt method

 if `lincom\_level\_dof' < 2 & "`model\_type'" == "mixed\_satt" {

 lincom `L`segment'', df(2)

 }

 // now save those values

 local level\_change\_counter\_`segment' = r(estimate)

 local level\_change\_counter\_`segment'\_ll = r(lb)

 local level\_change\_counter\_`segment'\_ul = r(ub)

 local level\_change\_counter\_`segment'\_se = r(se)

 local level\_change\_counter\_`segment'\_p = r(p)

 } // end if loop

 } // end segment loop

 ////////////////////////////////////////////////////////////////////////////////

 // slope changes from counterfactual

 // this runs as the level change above, but for slope change values instead

 forvalues segment = 0/`num\_segments' {

 if `segment' == 0 {

 local slope\_change\_counter\_`segment' = .

 local slope\_change\_counter\_`segment'\_ll = .

 local slope\_change\_counter\_`segment'\_ul = .

 local slope\_change\_counter\_`segment'\_se = .

 local slope\_change\_counter\_`segment'\_p = .

 }

 else {

 forvalues segment\_sub = 1/`segment' {

 if `segment\_sub' == 1 {

 local SC\_`segment' = "slope\_change\_`segment\_sub'"

 }

 else {

 local SC\_`segment' = "`SC\_`segment''" + " + slope\_change\_`segment\_sub'"

 }

 }

 lincom `SC\_`segment'' `small'

 if "`model\_type'" == "arima" | "`model\_type'" == "mixed" {

 local lincom\_slope\_dof = e(N) - e(df\_m)

 }

 else {

 local lincom\_slope\_dof = r(df)

 }

 \*return list

 display "slope lincom dof = `lincom\_slope\_dof'"

 if `lincom\_slope\_dof' < 2 & "`model\_type'" == "mixed\_satt" {

 lincom `SC\_`segment'', df(2)

 }

 local slope\_change\_counter\_`segment' = r(estimate)

 local slope\_change\_counter\_`segment'\_ll = r(lb)

 local slope\_change\_counter\_`segment'\_ul = r(ub)

 local slope\_change\_counter\_`segment'\_se = r(se)

 local slope\_change\_counter\_`segment'\_p = r(p)

 } // end if loop

 } // end segment loop

 } // end model error check.

 ////////////////////////////////////////////////////////////////////////////////

 // postvalues

 // finally put the values into the post file

 forvalues segment = 0/`num\_segments' {

 local analysis\_autocorr = analysis\_autocorr[`time\_`segment'\_start']

 // in the excel file the segments begin with 1, in this file we set to zero

 // therefore we need to similarly reduce the analysis effects by 1 so they all match up

 local analysis\_effects = analysis\_effects[`time\_`segment'\_start'] - 1

 summ time if segment == `segment'

 local segment\_num\_points = r(N)

 // scaled adjustment by rmse

 local level\_change\_counter\_`segment' = `level\_change\_counter\_`segment''/`rmse'

 local level\_change\_counter\_`segment'\_ll = `level\_change\_counter\_`segment'\_ll'/`rmse'

 local level\_change\_counter\_`segment'\_ul = `level\_change\_counter\_`segment'\_ul'/`rmse'

 local level\_change\_counter\_`segment'\_se = `level\_change\_counter\_`segment'\_se'/`rmse'

 local slope\_change\_counter\_`segment' = `slope\_change\_counter\_`segment''/`rmse'

 local slope\_change\_counter\_`segment'\_ll = `slope\_change\_counter\_`segment'\_ll'/`rmse'

 local slope\_change\_counter\_`segment'\_ul = `slope\_change\_counter\_`segment'\_ul'/`rmse'

 local slope\_change\_counter\_`segment'\_se = `slope\_change\_counter\_`segment'\_se'/`rmse'

 post `post\_values\_`model\_type'' (`series\_id') ///

 ("`model\_type'") ///

 (`segment') (`analysis\_autocorr') (`analysis\_effects') (`segment\_num\_points') (`total\_num\_points') (`rmse') ///

 (`level\_change\_counter\_`segment'') (`level\_change\_counter\_`segment'\_ll') (`level\_change\_counter\_`segment'\_ul') (`level\_change\_counter\_`segment'\_se') (`level\_change\_counter\_`segment'\_p') /// ///

 (`slope\_change\_counter\_`segment'') (`slope\_change\_counter\_`segment'\_ll') (`slope\_change\_counter\_`segment'\_ul') (`slope\_change\_counter\_`segment'\_se') (`slope\_change\_counter\_`segment'\_p') ///

 (`rho\_est') (`rho\_est\_cil') (`rho\_est\_ciu') ///

 (`num\_iterations') (`error\_code') (`converged') (`lincom\_level\_dof') (`lincom\_slope\_dof')

 } // end segment loop

 restore

 } // end study id loop

 postclose `post\_values\_`model\_type''

} // end of model type loop

* 1. Data file 3: combine\_methods\_published.do

////////////////////////////////////////////////////////////////////////////////

// This is the analysis file for the empirical study:

// "Comparison of six statistical methods for interrupted time series studies: empirical evaluation of 190 published series"

// this do file combines the results for all the different statistical methods

// the combined file is called estimates\_published\_all.dta

local model\_types "regress newey prais\_raw mixed mixed\_satt arima"

local first = 0

foreach model of local model\_types {

 if `first' == 0 {

 use "estimates\_published\_`model'.dta", clear

 local first = 1

 }

 else {

 append using "estimates\_published\_`model'.dta"

 }

}

save estimates\_published\_all.dta, replace

* 1. Data file 4: get\_wanted\_empirical\_estimates\_published.do

////////////////////////////////////////////////////////////////////////////////

// This is the final do file for the empirical study:

// "Comparison of six statistical methods for interrupted time series studies: empirical evaluation of 190 published series"

// It selects the effect estimate wanted for final comparisons

// excluding segments that are not required (e.g. transition periods)

// It requires the data from STurner\_Empirical\_study\_information which contains the desired segment number

// The final file "empirical\_estimates\_published.dta" is the output

// keep only the segments we need

keep if segment == analysis\_effects

// tidy the file

sort series\_id model\_type segment

order series\_id model\_type segment segment\_num\_points total\_num\_points

// save the final data file

save empirical\_estimates\_published.dta, replace