FACULTAD DE CIENCIAS SOCIALES
ESPECIALIDAD DE ECONOMÍA
Laboratorio de R y Python
SEMESTRE
Prof. Roberto Mendoza Matos

Trabajo Final - Grupo 7

---

1. Regresiones lineales - Python y R

Para este trabajo se importaron las librerías haven, texreg, xtable (para visualizar data, crear matrices y tablas), tidyverse (para crear gráficos) y; fastDummies, stargazer, sandwich, estimatr, caret (para generar regresiones). Se descargó la base **mss-repdata.dta** con el nombre de **repdata**.

1.1. Tabla estadística

Se produce la tabla de estadísticas de la Tasa de variación del índice de vegetación, términos de intercambio, porcentaje de las exportaciones respecto al PBI, densidad poblacional rural, porcentaje de tierra cultivable en uso, valor agregado del sector agricultaura respecto PBI y del sector manufacturero respecto PBI. Para ello, se incluyen el número de observaciones, la media, la desv. estándar, el mínimo y el máximo.

Cuadro 1: Descriptive Statistics

Statistic	N	Mean	St. Dev.	Min	Max
Tasa de variación del índice de vegetación	646	0.01	0.09	−0.47	0.66
Términos de intercambio	668	109.88	34.68	45.75	348.28
Porcentaje de las exportaciones respecto al PBI	698	64.25	34.29	6.32	180.96
Densidad poblacional rural	720	324.82	193.09	69.12	986.60
Porcentaje de tierra cultivable en uso	701	1.98	3.37	0.002	13.84
Valor agregado del sector agricultura respecto PBI	702	32.18	15.17	3.20	69.33
Valor agregado del sector manufacturero respecto PBI	669	11.12	6.26	1.87	37.16

Note.—The source of most characteristics is the World Bank's World Development Indicators (WDI).

TRABAJO

Cuadro 2:

Statistic	N	Mean	St. Dev.	Min	Max
american	627	0.903	0.297	0	1
instate	627	0.222	0.416	0	1
freshman	627	0.864	0.343	0	1
ACumGPA	627	3.432	0.435	1.200	4.000
greek	501	0.643	0.480	0	1
econ_hs	501	0.579	0.494	0	1
varsity	500	0.072	0.259	0	1

Cuadro 3:

	Dependent variable:	
	took_year	tookanother
	(1)	(2)
treat2016	0.115**	0.158**
	(0.055)	(0.066)
yr_2016	−0.040	−0.062
	(0.037)	(0.045)
treatment_class	−0.038	−0.048
	(0.038)	(0.046)
Constant	0.147***	0.237***
	(0.027)	(0.033)
Observations	627	627
R^2	0.008	0.010
Adjusted R^2	0.003	0.006
Residual Std. Error (df = 623)	0.340	0.411
F Statistic (df = 3; 623)	1.679	2.160*
Note:	*p<0.1; **p<0.05; ***p<0.01	

Cuadro 4:

	Dependent variable:	
	numeconclass	econmajor
	(1)	(2)
treat2016	0.692	0.098**
	(0.431)	(0.048)
yr_2016	−0.173	−0.023
	(0.294)	(0.033)
treatment_class	−0.129	−0.023
	(0.300)	(0.034)
Constant	1.026***	0.103***
	(0.214)	(0.024)
Observations	627	627
R^2	0.006	0.009
Adjusted R^2	0.001	0.005
Residual Std. Error (df = 623)	2.678	0.300
F Statistic (df = 3; 623)	1.290	1.944
Note:	*p<0.1; **p<0.05; ***p<0.01	

Cuadro 5:

	Dependent variable:			
	Major_STEM	Major_Business	Major_Finance	Major_Marketing
	(1)	(2)	(3)	(4)
treat2016	0.008	0.023	0.070	−0.047
	(0.051)	(0.066)	(0.057)	(0.055)
yr_2016	0.003	0.002	−0.049	0.039
	(0.035)	(0.045)	(0.039)	(0.037)
treatment_class	−0.012	−0.041	−0.033	0.044
	(0.036)	(0.046)	(0.040)	(0.038)
Constant	0.115***	0.224***	0.173***	0.103***
	(0.025)	(0.033)	(0.029)	(0.027)
Observations	627	627	627	627
R^2	0.0003	0.002	0.003	0.003
Adjusted R^2	−0.004	−0.003	−0.002	−0.002
Residual Std. Error (df = 623)	0.318	0.409	0.356	0.340
F Statistic (df = 3; 623)	0.071	0.396	0.608	0.548
Note:				*p<0.1; **p<0.05; ***p<0.01

Cuadro 6:

	Dependent variable:			
	Major_SocSc	Major_Arts	Major_Comm	Major_Hum
	(1)	(2)	(3)	(4)
treat2016	−0.007	−0.018	−0.008	−0.125***
	(0.050)	(0.036)	(0.037)	(0.044)
yr_2016	−0.034	0.012	−0.013	0.057*
	(0.034)	(0.025)	(0.026)	(0.030)
treatment_class	−0.031	0.029	0.003	0.077**
	(0.035)	(0.025)	(0.026)	(0.031)
Constant	0.141***	0.038**	0.064***	0.045**
	(0.025)	(0.018)	(0.019)	(0.022)
Observations	627	627	627	627
R^2	0.006	0.002	0.001	0.014
Adjusted R^2	0.001	−0.002	−0.003	0.009
Residual Std. Error (df = 623)	0.311	0.227	0.233	0.275
F Statistic (df = 3; 623)	1.261	0.484	0.284	2.861**
Note:				*p<0.1; **p<0.05; ***p<0.01

1.2. Regresión

- **Se empezó produciendo las dummy's para hallar efectos fijos a nivel país**
- **Se crea la variable temporal obteniendo la diferencia de los años con 1978**
- **Se crea ccode como una variable tipo factor**
- **Se usa la función lm (fitting linear models) y coef test para errores estandar robustas y clusterizadas para el modelo 1 y 2, los cuales se diferencian al tener la variable dependiente diferente. En la primera sería any prio; y la segunda, war prio**

$$any_Prio_t = \beta_1 GrowthInRainfall_t + \beta_2 GrowthInRainfall_{t-1} + \beta_3 CcodeFactor + \beta_4 CountryFixedEffects + \beta_5 CountrySpecificTrends + ... + \epsilon_t$$

$$war_Prio_t = \beta_1 GrowthInRainfall_t + \beta_2 GrowthInRainfall_{t-1} + \beta_3 CcodeFactor + \beta_4 CountryFixedEffects + \beta_5 CountrySpecificTrends + ... + \epsilon_t$$

Cuadro 7: Rainfall and Civil Conflict (Reduced-Form)

	Civil Conflict $\geq$ 25 Death (OLS)	Civil Conflict $\geq$ 1,000 Death (OLS)
	(1)	(2)
Growth in rainfall, t	-0.024	-0.062**
	(0.043)	(0.030)
Growth in rainfall, $t-1$	-0.122**	-0.069**
	(0.052)	(0.032)
Country fixed effects	yes	yes
Country-specific time trends	yes	yes
Root mean square error	0.24	0.2
R2	0.71	0.7
N	743	743
Notes:	Huber robust standard errors are in parentheses	

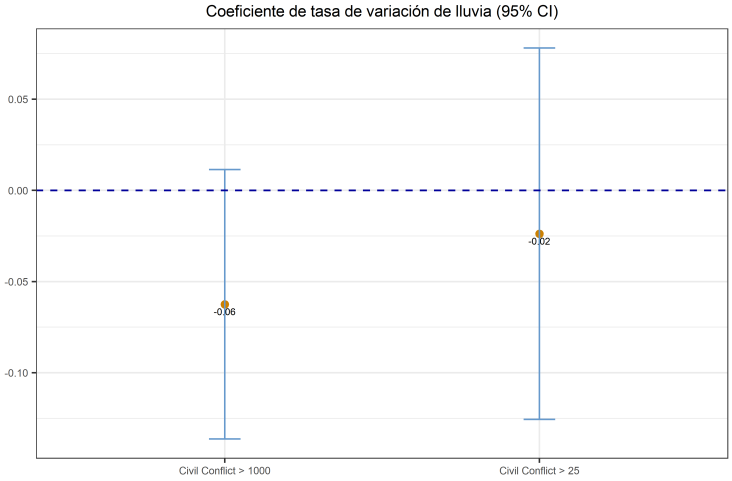
1.3. Coefplot

Se pide el **coefplot de la variable GPCP-g** para ambos modelos Civil Conflict > 25 y > 1000. Los pasos para gráficar fueron:

- **Extraer coeficientes (coef), desviación estándar (std) e intervalos de confianza (ic) de las tablas de ambas regresiones.** En el caso de los ic. se usa el comando **coefci** aplicaándolo a la tabla de estimación 'm1' y 'm2'.
- **Construir tabla para coef, std e ic.** La tabla se crea con el comando **matrix()**. Se coloca nombre a las columnas con **colnames**, los índices con **rownames**.

- **Graficar**. Ajustar tamaño del gráfico con **options(repr.plot.width = 8, repr.plot.height =10)**. Graficar los coeficientes con **ggplot** usando en el eje 'x' los índices y en el eje 'y' la columna 'Estimate' de la tabla anterior. Graficar los ic con **geom-errorbar** usando las columnas 'Upper bound' y 'Lower bound' de la tabla anterior. Graficar linea en y=0 con **geom-hline**. Agregar titulo con **ggtittle**.
- **Guardar con comando ggsave** en formato png.

El resultado es:



2. Modelos lineales (Python)

Para este trabajo se importaron las librerías pandas y numpy (para visualizar data y crear matrices), matplotlib y seaborn (para crear gráficos)y; stasmodels y pystout (para generar regresiones). Se descargó la base **mss-repdata.dta** con el nombre de **repdata**.

2.1. Tabla estadística

Se pide construir una tabla estadística donde se muestre la media, desv. estándar y total de observaciones de las siguientes variables:
NDVI g ⇒ Tasa variación índice de vegetación
tot 100 ⇒ Términos de intercambio
trade pGDP ⇒ Porcentaje de las exportaciones recpecto al PBI
pop den rur ⇒ Densidad poblacional rural
land crop ⇒ Porcentaje de tierra cultivable en uso
va agr ⇒ Valor agregado del sector agricultura respecto PBI
va ind manf ⇒ Valor agregado del sector manufacturero respecto PBI
Los pasos fueron:

- Crear una tabla (**table1**) donde solo estén las variables mencionadas. Se usó el comando loc: repdata.loc[: , "nombre de columna"] para ello.
- Usar **'describe'** para conseguir las estadísticas descriptivas de estas variables.
- Usar comando **loc** para quedarse solo con las filas que contienen la media, desviación y total. El formato fue: table1.describe().loc[["mean","std", çount"]] Guardar ello en Dataframe llamado **summary-table**.

- El nombre de las variables sale de forma abreviada (es decir, NDVI g, va agr, ...) por lo que se etiqueta ello con su descripción. Por ejemplo:

  pop den rur (nombre inicial) ⇒ Densidad poblacional rural (nuevo nombre)
  Para ello se crea una lista con los nuevos nombres y se renombra las filas del Dataframe **summary-table**.

- Por último exportar a latex el Dataframe **summary-table** con el comando **.to-latex**.

El resultado se muestra en el siguiente cuadro:

Cuadro 8: Estadisticas descriptivas

Statistic	Mean	Standard Deviation	Minimum	Maximum	Observations
American student	0.90	0.30	0.0	1.0	627
In-state student	0.22	0.42	0.0	1.0	627
Freshman	0.86	0.34	0.0	1.0	627
Cumulative GPA	3.43	0.44	1.2	4.0	627
Belongs to sorority	0.64	0.48	0.0	1.0	501
Took econ in high school	0.58	0.49	0.0	1.0	501
Athlete	0.07	0.26	0.0	1.0	500

2.2. Regresión

Se pide replicar la Tabla 3 del paper Economic Shocks and Civil Conflict. La regresión estimada es:

$$CiviConflict_t = \beta_1 GrowthR_t + \beta_2 GrowthR_{t-1} + ... + \epsilon_t \quad (1)$$

A esta ecuación base se le agrega variables que controlan los **efectos fijos** por país y los **efectos específicos temporales** por país. Se trabaja con dos variables endógenas para caracterizar a 'Civil War'. La primera 'anyprio' es una dicotómica que es 1 si un conflicto tiene más de 25 fallecidos y 0 en otro caso. La segunda es 'warprio', una dicotómica que es 1 si un conflicto tiene más de 1000 fallecidos y 0 otro caso. Los pasos para la estimación son:

- **Obtener dummies por país que representan los efectos fijos.** Utilizar comando **.get-dummies** usando como información la columna 'ccode' con los códigos por país. Se consiguen 41 columnas (pues son 41 países). Concatenar dummies usando comando **.concat** a la data principal **repdata**.

- **Obtener dummies para country-trend.** Esto significa multiplicar las dummies de país por una variable temporal. Se utiliza **DatetimeIndex** para crear la variable temporal en base a información de columna *year*. Luego se usa el loop 'while' para iterar las columnas dummies de país con la columna de tiempo.

- **Se regresiona los dos modelos**. Las ecuaciones en python tienen la estructura:
  formula = y ∼ GPCPg + GPCPgl + C(ccode) + ' + ' + ' + '.join(country-trend)
  Donde **y** es any-prio o war-prio dependiendo si son 25 o 1000 fallecidos en la guerra civil. C(ccode) indica que hay efectos fijos y .join(country-trend) que hay efectos por país en el tiempo.

- **Las ecuaciones anteriores se estiman por OLS.** La estructura de la estimación es: smf.ols(formula, data).fit(cov type=cluster, cov kwds=groups: repdata[ccode])
  En el primer paréntesis se incluye la ecuación descrito en el item anterior. En el segundo, se incluye las especificaciones sobre la varianza **cov-type** que es heterocedástico tipo Huber White robust y se especifica que las perturbaciones están agrupadas por niveles de países con el comando con **cov-kwds**. La tabla para el modelo cuya 'y' es Civil War > 25 se guarda con el nombre ols-model1 y el modelo cuya 'y' es Civil War > 1000 se guarda como ols-model2.

- **Se exporta a latex con pystout.**

Los resultados se muestran en el Cuadro 4.

2.3. Coefplot

Se pide el **coefplot de la variable GPCP-g** para ambos modelos Civil Conflict > 25 y > 1000. Los pasos para la gráficar el coefplot fueron:

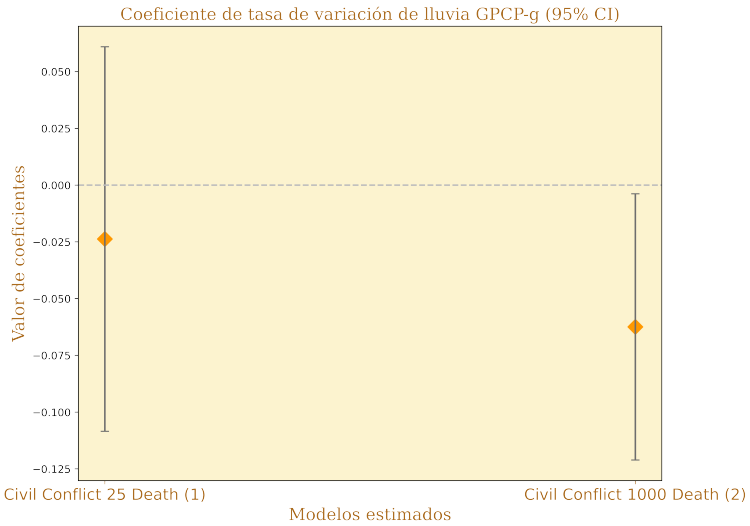
- **Extraer coeficientes (coef), desviación estándar (std) e intervalos de confianza (ic) de las dos tablas de estimaciones.** Para ello usar comando .loc. por ejemplo en el primer modelo usar: model1.loc [GPCP-g, Coef.]. Repetir procedimiento con demás coef., std. e ic.

- **Construir tabla con coef, std e ic.** usando el comando **np.zeros**. Colocar nombre a las columnas con **pd.DataFrame(**table, columns = [Estimate , Std. Error, Lower bound, Upper bound]).

- **Graficar**. Primero, ajustar tamaño de gráfico con **plt.subplots(figsize = (,))**. Segundo, graficar coeficientes con **ax.scatter** usando en el eje 'x' los índices y en el eje 'y' la columna 'Estimate' de la tabla anterior. Tercero, graficar los i.c. con **plt.errorbar** usando las columnas 'Upper bound' y 'Lower bound' de la tabla anterior. Cuarto, graficar linea en y=0 con **plt.axhline** con el objetivo de ver la significancia. Por último, agregar titulo y descripción con **plt.tittle, plt.ylabel y plt.xlabel**.

- **Guardar usando el comando fig.savefig** en formato png.

El resultado se muestra en la siguiente gráfica donde se evidencia que GPC-g es solo significativo para la regresión con más de 1000 fallecidos.



En general, los resultados pudieron ser replicados con ambos programas.

3. Web Scraping

El ejercicio consiste en extraer datos de la página web que contiene el número de votos por centro de votación de acuerdo a sus características (votos nulos, válidos, etc.). Los datos para desde 2000 al 2015. Para realizar este ejercicio se descarga diferentes tipos de librerías como Selenium que se usa para automatizar la interacción del navegador web desde Python, se descarga webdriver que permite generar un controlador automático. También se instala la librería pandas para manejo de bases.

Previa creación del código, es necesario ingresar a la página para ubicar los elementos de interés. En este caso, identificamos con especial importancia el botón que nos permitirá pasar a la siguiente página para obtener la siguiente tabla. Con un simple inspect, podemos obtener el XPATH, que actúa como la dirección del elemento, y la cual incluiremos en nuestro código para programar un click y que así funcione nuestro programa automático.

No solo bastará con extraer los datos de la página web. Como es indicado, también deberemos incluir la información extraída en una única base de datos. Nuestra aproximación a tal tarea será incluir dentro de nuestro código de extracción, la creación de un "dataframe"para el cual sean añadidos, a través de las función append, los datos que se vayan extrayendo de las tablas.

Los pasos son:

1. Descargar las librerias necesarias para la tarea, principalmente pandas, selenium y webdriver.
2. Generar el webdriver para Chrome, e indicar el url de donde se extraerá la información. Indicamos que se abra la página.
3. Generamos el **dataframe**: donde se colocarán los valores extraídos.
4. Definimos la función que indica cómo se deben extraer los elementos. La primera fila de cada tabla debe ser los nombres de las columnas en nuestros dataframe, y los demás valores deben ser añadidos a tabla por medio de un append.
5. Definimos la presencia del botón "next", e indicamos un click y añadimos la indicación de que el programa espere hasta que el elemento aparezca.
6. Para la nueva página, aplicamos la misma función de extracción de tablas. Luego se define el click en el nuevo botón de "next". Esto se define como **loop**:
7. Una vez que el programa ya no encuentra el botón de next, se le pide que de por finalizado el ciclo.
8. Se pide que nos muestre el dataframe generado, y que lo exporte como un archivo .csv.

4. Mapas (Python y R)

El ejercicio consiste en obtener 2 figuras del paper Land Reform and Civil Conflict: Theory and Evidence from Peru de Michael Albertus.

La Figura 1 es un mapa que comprende las zonas agrarias, las zonas agrarias core y los límites distritales.

En Python, los pasos para la elaboración de la figura 1 son:

1. Cargar los archivos shapefile de los distritos, departamentos, el archivo de polígonos del núcleo de la zona agraria
2. Ajustar el tamaño de la figura y añadir los subplots
3. Agregar nombres y números de los departamentos en el mapa de distritos y zonas agrarias después de haber delimitado el tamaño de las lineas, el color, y el tipo de línea, que en este caso es punteada o dashed.
4. Obtener el límite de ejes del mapa de departamentos y zonas agrarias
5. Delimitar la configuración de las líneas y colores en el Mapa de departamentos y zonas agrarias
6. Agregar nombres y números de los departamentos
7. Mostrar la figura.

La figura 2 se logró en Python con los siguientes pasos:

1. Fijamos el directorio para los shapefiles y para la data de zonas
2. Leemos los shapefiles
3. Leemos la data de zonas
4. Graficamos los shapefiles y la data de zonas
5. Graficamos los shapefiles y la data de zonas
6. Imprimimos los datos del shapedata.csv para hacer el merge
7. hacemos merge con los datos del dataframe
8. Creamos el mapa de calor Log percent land areas expropiated by district, 1969-80
9. Creamos el segundo mapa de calor Log total attacks by district, 1980-2000.

En R, los pasos para la elaboración de la figura 1 son:

1. Instalamos los paquetes y librerias necesarias
2. Cargamos las librerías requeridas
3. Establecemos el directorio de trabajo para la carpeta que contiene los shapefiles y archivos de zona
4. Leemos los shapefiles
5. Leemos los archivos de zona
6. En este paso, seleccionamos los colores que queremos que nuestro mapa tenga, así como el tamaño del alfa. Luego, graficamos los shapefiles y archivos de zona para la figura 3.1.A
7. Seleccionamos colores, el tamaño del alfa,y graficamos los shapefiles y archivos de zona para la figura 3.1.B.

La figura 2 comprende los logaritmos de los porcentajes de áreas que fueron expropiadas por distrito. Además, se incluye otro mapa con los logaritmos de los ataques totales por distrito.

En R, los pasos par la elaboración de la figura 2 son:

1. Teniendo en cuenta los pasos anteriores, hacemos merge del shapefile con el marco de datos 'datos'

2. Restablecemos el estado de gráficos

3. Finalmente, creamos el mapa de calor geográfico 3.2.A y el mapa de calor geográfico 3.2.B