Taller Data Analysis con Python - Didáctica de la Sesión

El presente documento es una guía que presenta el principal flujo de análisis para el Taller de Data Analysis con Python. A grandes rasgos el ejercicio busca implementar un análisis exploratorio básico, considerando preprocesamiento y recodificación de datos. Posteriormente el asistente podrá generar gráficos básicos mediante `matplotlib` y `seaborn`, concluyendo con correlaciones y regresiones lineales implementadas con `statsmodels`.

La base de datos a utilizar proviene del libro *Hamilton, L. 2013. Statistics with Stata. Ch4*. Esta es una selección de diferentes atributos de desarrollo humano en 194 países, recolectados por las Naciones Unidas. La definición de las variable son:

```
. use C:\data\Nations2.dta, clear
. describe

Contains data from C:\data\Nations2.dta
  obs:           194                          UN Human Development Indicators
 vars:            13                          1 Apr 2012 11:30
 size:        12,804
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
country         str21  %21s                   Country
region          byte   %8.0g       region     Region
gdp             float  %9.0g                  Gross domestic product per cap
                                                2005$, 2006/2009
school          float  %9.0g                  Mean years schooling (adults)
                                                2005/2010
adfert          float  %8.0g                  Adolescent fertility: births/1000
                                                fem 15-19, 2010
chldmort        float  %9.0g                  Prob dying before age 5/1000 live
                                                births 2005/2009
life            float  %9.0g                  Life expectancy at birth
                                                2005/2010
pop             float  %9.0g                  Population 2005/2010
urban           float  %9.0g                  Percent population urban
                                                2005/2010
femlab          float  %9.0g                  Female/male ratio in labor force
                                                2005/2009
literacy        float  %9.0g                  Adult literacy rate 2005/2009
co2             float  %9.0g                  Tons of CO2 emitted per cap
                                                2005/2006
gini            float  %9.0g                  Gini coef income inequality
                                                2005/2009
-------------------------------------------------------------------------------
Sorted by:  region  country
```

Al presentarse el flujo de trabajo para el taller, el profesor está en la potestad de modificar el ejemplo y/o realizar variaciones de los ejercicios. Si ese es el caso, se recomienda seguir a grandes rasgos el modelo de las dinámicas.

La guía presenta algunos tips sobre preguntas frecuentes realizadas por los alumnos en talleres previos.

Primera dinámica: Explorando el flujo de trabajo con Python

- Posterior a la exposición de los elementos de trabajo (Python, Anaconda y Jupyter), es recomendable que el asistente suelte los dedos con algunos ejercicios simples. Se recomiendan los siguientes:

1. Partir con el clásico `Hello World`, explicar la diferencia entre Python 2.x y Python 3.x. Enseñar el modo de ejecución `Shift + Enter`.
2. Generar ejemplos de operaciones aritméticas.
3. Mostrar los modos de código y markdown, incluyendo las principales reglas de Markdown.

4. Guardar el nombre del asistente en una variable y llamarlo posteriormente.

In [1]:
```
print("Hola mundo!")
print(2+ 4)
nombre = "Ignacio"
print("Hola, mi nombre es " + nombre)
```

```
Hola mundo!
6
Hola, mi nombre es Ignacio
```

Segunda dinámica: Preparación del ambiente de trabajo

- Posterior a la primera interacción con Python, se debe destacar algunos elementos sobre el trabajo en ciencia de datos:
- Librerías: Una de las ventajas de Python son las librerías existentes que sistematizan buenas prácticas y procesos. De esta forma, nos abstraemos de los problemas asociados al código y nos centramos en lo importante como analistas: **los datos**. El alumno implementará las librerías `pandas` y `numpy`

In [2]:
```
import pandas as pd
import numpy as np
```

- Nomenclatura y formalidades: Uno de los errores más comunes de los alumnos neófitos es la falta de rigurosidad en la forma que ingresan y declaran objetos en Python. Se debe poner énfasis en la idea de seguir normas y convenciones. Por ejemplo, es necesario destacar la forma en que se incorporan las librerías como una buena práctica entre la comunidad de científicos de datos. Un ejemplo clásico es el siguiente:

In [3]:
```
esta_es_una_cadena = "Lorem ipsum dolor sit amet"
print(Esta_Es_Una_Cadena)
```

```
---------------------------------------------------------------------
-------
NameError                                 Traceback (most recent cal
l last)
<ipython-input-3-90ae542decad> in <module>()
      1 esta_es_una_cadena = "Lorem ipsum dolor sit amet"
----> 2 print(Esta_Es_Una_Cadena)

NameError: name 'Esta_Es_Una_Cadena' is not defined
```

Desafío: Ingresar base de datos

- Para esto se debe incorporar la base de datos con `pd.read_csv()`.
- En una primera instancia ejecutarlo **sin guardar el resultado de la expresión en una variable**. Explicar que ésta tabla no vive más allá de la ejecución.
- Se debe presentar la documentación asociada al método, para lo cual deben hacer uso de `Shift + Tab`. El docente debe señalar que existen parámetros de entrada y un objeto de retorno. Recalcar la importancia de la documentación.

In [4]:
```
pd.read_csv('nations.csv')
```

Out[4]:

	Unnamed: 0	country	region	gdp	school	adfert	chldmort
0	1	Algeria	Africa	7300.399902	6.716667	7.300000	34.75
1	2	Benin	Africa	1338.800049	3.100000	111.699997	122.75
2	3	Botswana	Africa	12307.400391	8.600000	52.099998	60.25
3	4	Burkina Faso	Africa	1063.400024	1.300000	124.800003	170.50
4	5	Burundi	Africa	349.200012	2.483333	18.600000	168.50
5	6	Cameroon	Africa	1986.800049	5.650000	127.800003	155.00
6	7	Cape Verde	Africa	3052.199951	3.500000	81.599998	30.00
7	8	Central African Rep	Africa	677.000000	3.383333	106.599998	173.50
8	9	Chad	Africa	1266.199951	1.500000	164.500000	209.00
9	10	Comoros	Africa	1099.000000	2.800000	58.000000	105.75
10	11	Congo	Africa	3628.000000	5.850000	118.699997	125.75
11	12	Congo (Dem Rep)	Africa	279.799988	3.416667	201.399994	199.00
12	13	C�te d'Ivoire	Africa	1539.199951	3.200000	129.399994	123.25
13	14	Djibouti	Africa	1972.199951	3.800000	22.900000	96.00
14	15	Egypt	Africa	4754.399902	5.950000	46.599998	24.75
		Equatorial					

15	16	Guinea	Africa	27645.800781	5.400000	122.900002	149.50
16	17	Eritrea	Africa	579.599976	3.400000	66.599998	60.50
17	18	Ethiopia	Africa	741.400024	1.500000	72.400002	112.00
18	19	Gabon	Africa	13183.200195	7.183333	89.900002	72.50
19	20	Gambia	Africa	1218.400024	2.550000	76.599998	108.25
20	21	Ghana	Africa	1303.000000	6.783333	71.099998	75.25
21	22	Guinea	Africa	957.599976	1.600000	157.399994	149.50
22	23	Guinea-Bissau	Africa	967.599976	2.300000	111.099998	197.50
23	24	Kenya	Africa	1405.400024	6.750000	100.199997	87.75
24	25	Lesotho	Africa	1284.400024	5.583333	73.500000	98.25
25	26	Liberia	Africa	345.000000	3.650000	142.600006	125.50
26	27	Libya	Africa	14497.799805	6.816667	3.200000	19.75
27	28	Madagascar	Africa	919.799988	5.200000	134.300003	64.50
28	29	Malawi	Africa	660.400024	3.816667	119.199997	120.25
29	30	Mali	Africa	1032.800049	1.833333	186.300003	196.00
...	...	...	...	...	...	...	...
164	165	Netherlands	Europe	36646.398438	11.300000	5.100000	4.75
165	166	Norway	Europe	48169.398438	12.666667	9.000000	3.75
166	167	Poland	Europe	15446.400391	9.816667	14.800000	7.25
167	168	Portugal	Europe	21628.400391	7.450000	16.799999	4.25
168	169	Romania	Europe	10560.400391	10.283333	32.000000	14.50
169	170	Russian Federation	Europe	13424.799805	9.783334	30.000000	14.25
170	171	San Marino	Europe	NaN	NaN	2.500000	2.25
171	172	Serbia	Europe	9473.200195	9.950000	22.100000	8.00
172	173	Slovakia	Europe	18552.199219	11.600000	20.200001	7.50
173	174	Slovenia	Europe	25321.199219	10.850000	5.000000	3.50
174	175	Spain	Europe	27862.199219	10.033333	12.700000	4.25

175	176	Sweden	Europe	33621.398438	11.700000	6.000000	3.25
176	177	Switzerland	Europe	37105.800781	10.866667	4.600000	4.75
177	178	Ukraine	Europe	6124.000000	11.200000	30.799999	16.00
178	179	United Kingdom	Europe	33295.800781	9.066667	29.600000	6.00
179	180	Angola	Oceania	4662.000000	4.400000	171.100006	170.00
180	181	Australia	Oceania	33707.199219	11.983334	16.500000	5.25
181	182	Fiji	Oceania	4259.600098	10.300000	45.200001	18.00
182	183	Kiribati	Oceania	2294.199951	7.800000	22.200001	49.25
183	184	Marshall Is	Oceania	NaN	9.800000	53.500000	36.00
184	185	Micronesia	Oceania	2906.800049	8.800000	25.400000	40.00
185	186	Nauru	Oceania	NaN	NaN	31.200001	45.50
186	187	New Zealand	Oceania	25199.599609	12.366667	30.900000	6.25
187	188	Palau	Oceania	NaN	12.100000	13.800000	15.00
188	189	Papua New Guinea	Oceania	1953.800049	4.116667	66.900002	69.75
189	190	Samoa	Oceania	4012.600098	10.300000	28.299999	26.75
190	191	Solomon Islands	Oceania	2249.199951	4.500000	70.300003	36.00
191	192	Tonga	Oceania	4072.199951	10.133333	22.299999	19.25
192	193	Tuvalu	Oceania	NaN	NaN	23.299999	36.50
193	194	Vanuatu	Oceania	3809.800049	6.700000	54.000000	17.75

194 rows × 14 columns

Desafío (y moraleja): Entender que las variables son asignaciones de expresión

- Explicar que trabajaremos constantemente con esta tabla, por lo que preferimos guardarla en una variable.
- Explicar que una variable permite asignar el resultado de una expresión en el ambiente de trabajo.

In [5]:
```
df = pd.read_csv('nations.csv')
```

Desafío: Imprimir las primeras 5 observaciones con `df.head()`

- El profesor debe comentar sobre los datos `NaN` (not a number) presentes en la columna `gini`.
- El profesor también debe comentar sobre la variable `Unnamed: 0` que hace referencia al índice en el `csv`.
- **Pregunta frecuente:** Muchas veces preguntan sobre cómo leer más de 5 datos por defecto. Para ello se ingresa la cantidad a implementar dentro de los paréntesis. Se puede ocupar `tail` para obtener las últimas observaciones.

In [6]:
```
df.head()
```

Out[6]:

	Unnamed: 0	country	region	gdp	school	adfert	chldmort	
0	1	Algeria	Africa	7300.399902	6.716667	7.300000	34.75	72.316
1	2	Benin	Africa	1338.800049	3.100000	111.699997	122.75	54.733
2	3	Botswana	Africa	12307.400391	8.600000	52.099998	60.25	52.250
3	4	Burkina Faso	Africa	1063.400024	1.300000	124.800003	170.50	53.783
4	5	Burundi	Africa	349.200012	2.483333	18.600000	168.50	48.866

Desafío: Eliminar una columna en la base de datos

- Resulta que `Unnamed: 0` estorba. Para eliminarla generamos lo siguiente

In [7]:
```
df = df.drop(columns = 'Unnamed: 0')
```

- Con esto explicamos sobre la actualización del objeto y sobreescritura

Tercera dinámica: Estadística descriptiva y recodificación

Desafío: Diferenciar entre atributos contínuos y categóricos

- Ahora desarrollaremos nuestras primeras estadísticas descriptivas sobre las variables. Una buena práctica es identificar la naturaleza de las variables con las que estamos trabajando. El docente debe separar entre categóricas y numéricas.

```
In [8]: df.columns
```

```
Out[8]: Index(['country', 'region', 'gdp', 'school', 'adfert', 'chldmort', '
        life',
               'pop', 'urban', 'femlab', 'literacy', 'co2', 'gini'],
              dtype='object')
```

Desafío: Aplicar los métodos `value_counts` y `describe` en la variable `region`.

- La primera tarea es llamar a la columna `df['region']` para ver qué contiene. Explicar que és un `pd.Series` y por tanto tiene datos y funciones asociadas. Se debe checkear a los alumnos para ver quiénes no ocupan corchetes.

```
In [9]: df['region']
```

```
Out[9]: 0       Africa
        1       Africa
        2       Africa
        3       Africa
        4       Africa
        5       Africa
        6       Africa
        7       Africa
        8       Africa
        9       Africa
        10      Africa
        11      Africa
        12      Africa
        13      Africa
        14      Africa
        15      Africa
        16      Africa
        17      Africa
        18      Africa
```

```
19      Africa
20      Africa
21      Africa
22      Africa
23      Africa
24      Africa
25      Africa
26      Africa
27      Africa
28      Africa
29      Africa
        ...
164     Europe
165     Europe
166     Europe
167     Europe
168     Europe
169     Europe
170     Europe
171     Europe
172     Europe
173     Europe
174     Europe
175     Europe
176     Europe
177     Europe
178     Europe
179    Oceania
180    Oceania
181    Oceania
182    Oceania
183    Oceania
184    Oceania
185    Oceania
186    Oceania
187    Oceania
188    Oceania
189    Oceania
190    Oceania
191    Oceania
192    Oceania
193    Oceania
Name: region, Length: 194, dtype: object
```

- Concatenar los métodos `value_counts` y `describe` en la variable region. Como `df['region']` es string, demostrar que Python realiza operaciones dependiendo del tipo de dato que infiere.

Desafío: Implementar `describe` en las variables contínuas

- **Tips:** El profesor mostrar el uso de loops para sistematizar el flujo. Enfatizar la idea de la economía del lenguaje. Resulta que

```
for i in df.columns:
    print(df[i].describe()
```

Es más eficiente que estar repitiendo la expresión por cada variable de la siguiente forma:

```
In [10]: df['gini'].describe()
```

```
Out[10]: count    81.000000
         mean     40.477778
         std       8.487874
         min      19.000000
         25%      34.000000
         50%      39.700001
         75%      46.200001
         max      58.500000
         Name: gini, dtype: float64
```

- Centrarse en 1 o 2 variables del describe y explicar el output.

Desafío: Crear una nueva variable poblada con ceros

- Genere una variable llamada `df['Americas']` donde se asigne 0 (sin numpy)
- Imprimir el dataframe actualizado. Explicar que estamos agregando más información de forma bruta.

```
In [11]: df['Americas'] = 0
         df['Americas'].head()
```

```
Out[11]: 0    0
         1    0
         2    0
         3    0
         4    0
         Name: Americas, dtype: int64
```

Desafío: Generar una recodificación binaria para cada continente

- De esta forma se generan 5 nuevas columnas, una marcando como 1 si la observación pertenece al continente, 0 de lo contrario. Por defecto mostrar la estrategia lenta en un caso:
- Explicar que `replace` toma dos listas con elementos pareados.

In [12]:
```
# estrategia lenta
df['americas'] = df['region'].replace(['Africa', 'Asia', 'Europe', 'Am
ericas', 'Oceania'],
                                      [0, 0, 0, 1, 0])
```

- Posteriormente, el profesor puede hacer uso de un loop con `np.where` para demostrar la facilidad de Python.

In [13]:
```
# forma más sucinta
for i in df['region'].unique():
    df[i] = np.where(df['region'] == i, 1, 0)
```

- Mostrar el `df` actualizado.

Desafío: Generar una recodificación de una variable contínua

- Implementando `np.where`, clasificar como 1 todas las observaciones emisiones de `co2` sobre la media, 0 de lo contrario. Asignar esta operación a otra varable

In [14]:
```
df['co2_dummy'] = np.where(df['co2'] > df['co2'].mean(), 1, 0)
```

Cuarta dinámica: Gráficos

- En esta sección el alumno interactuará con `matplotlib` para generar gráficos.
- El primer gráfico se realizará directamente sobre la columna. De esta manera se destaca la interconexión entre las librerías `pandas`, `numpy` y `matplotlib`.

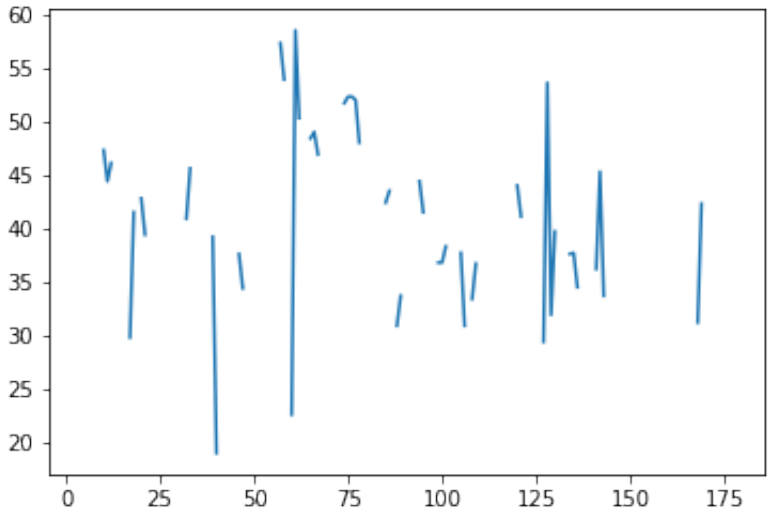
In [15]:
```
import matplotlib.pyplot as plt
```

Desafío: Graficar un histograma

- Si generamos nuestro primer gráfico con la siguiente línea, el resultado será horrible

```
In [16]: df['gini'].plot()
```

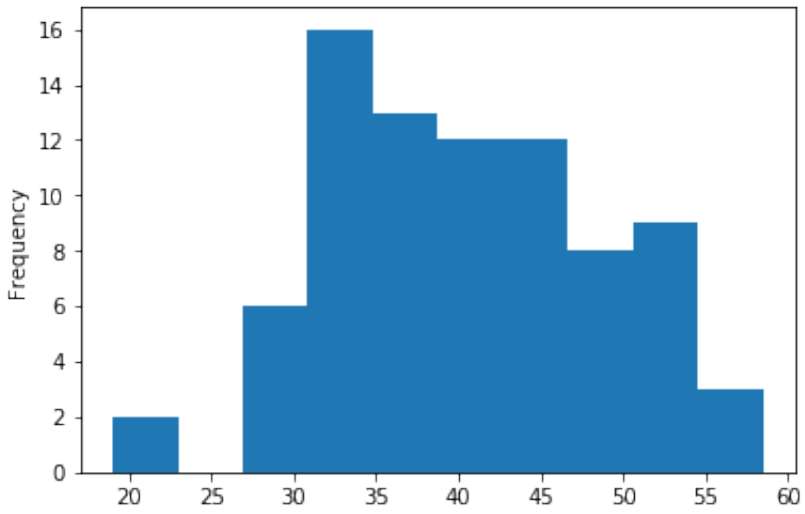
```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x10effa550>
```



- Resulta que dentro de `plot` se puede especificar qué tipo de gráfico se genera. Se recomienda ver la documentación asociada. Necesitamos un histograma que resuma la frecuencia de casos entre intervalos.

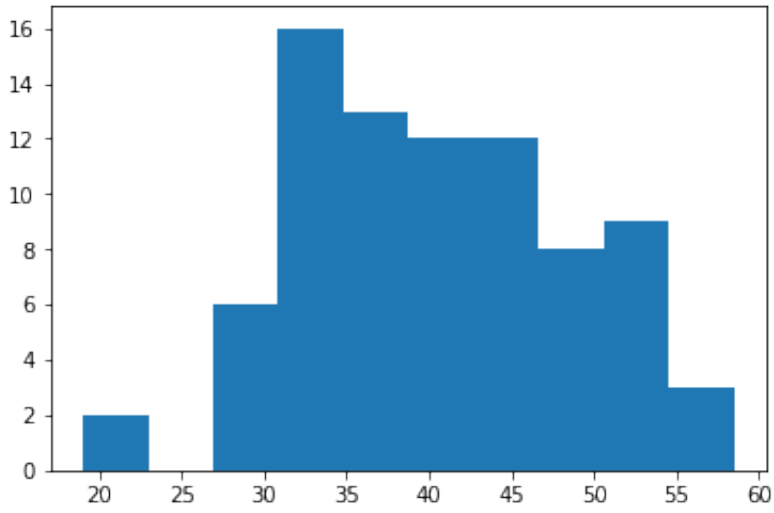
In [17]:
```
df['gini'].plot(kind='hist')
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x116e3b8d0>



- Hay que destacar que el método `plot` dentro de una serie es una buena forma de salir del paso, pero que la mejor manera de implementar gráficos es mediante matplotlib directo. El equivalente en matplotlib puro es `plt.hist(df['gdp'])`.
- Un punto a considerar es que arrojará un error, esto dado a que no soporta datos perdidos. Para ello hay que botar los datos perdidos de la serie con

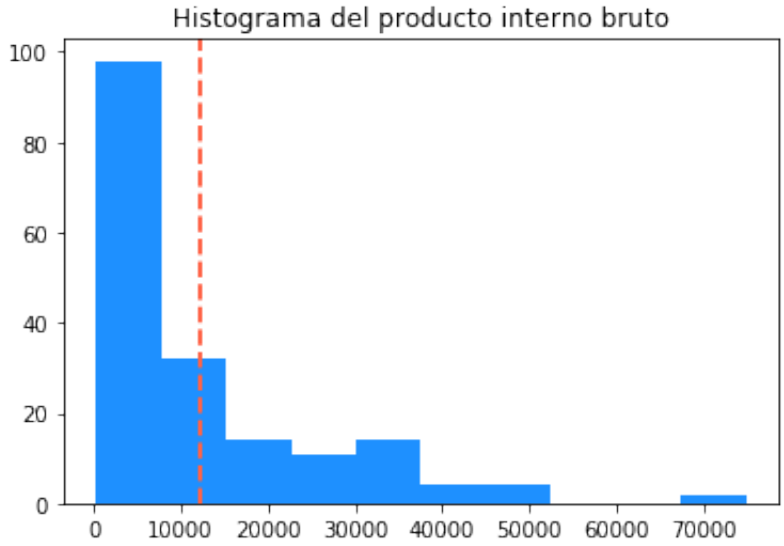
In [18]:
```
plt.hist(df['gini'].dropna());
```



- Podemos agregar más detalles sobre los elementos a operar en el gráfico. Para este ejemplo vamos a realizar los siguientes pasos:

In [19]:
```
# generamos una variable con la variable sin observaciones perdidas
gdp_na = df['gdp'].dropna()
# la ingresamos en plt.hist y cambiamos el color
plt.hist(gdp_na, color='dodgerblue');
# agregamos una línea roja que señale la media de la variable
plt.axvline(gdp_na.mean(), color='tomato', linewidth=2, linestyle='--'
)
# agregamos el título del gráfico
plt.title('Histograma del producto interno bruto')
```

Out[19]:
```
Text(0.5,1,'Histograma del producto interno bruto')
```



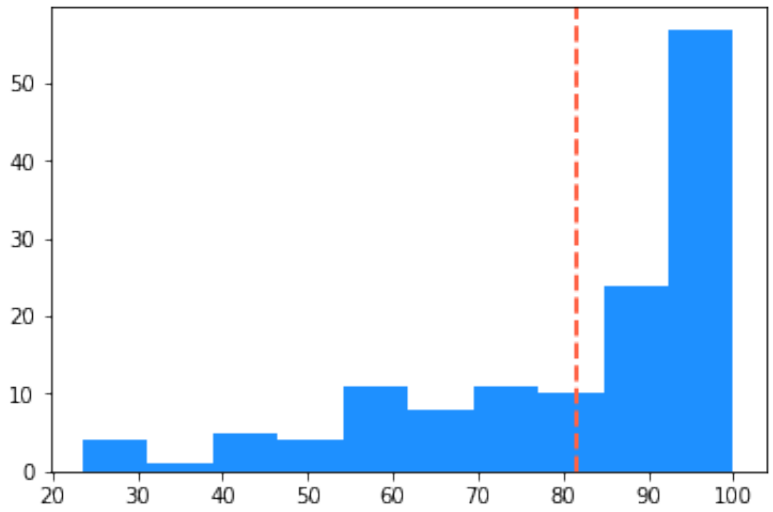
Desafío: Implementación de funciones

- Acá vale la pena destacar otra de las máximas de la programacíon: **Don't Repeat Yourself**. Envolveremos el código de arriba en una función para reutilizarla con distintas variables.
- Es necesario señalar que una función toma un parámetro de ingreso, en este caso nuestra variable

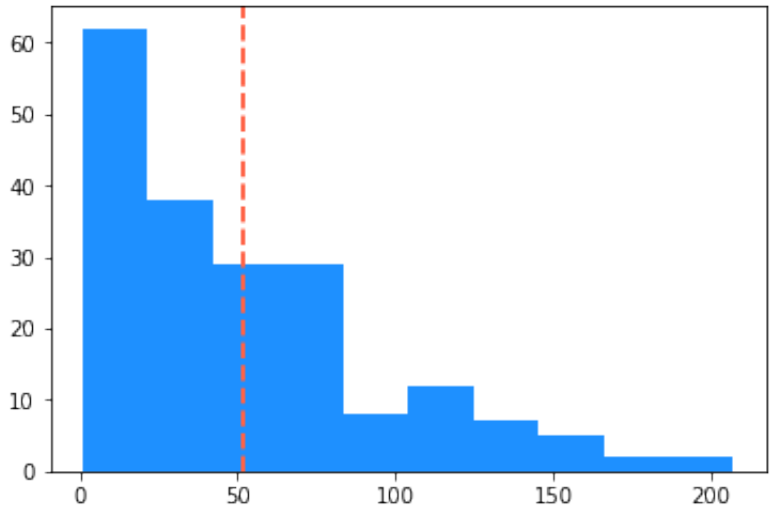
In [20]:
```
def plot_hist(var):
    tmp = var.dropna()
    plt.hist(tmp, color='dodgerblue')
    plt.axvline(tmp.mean(), color='tomato', linewidth=2, linestyle='--
')
```

- El alumno podrá implementar esta función en otras variables.

In [21]:
```
plot_hist(df['literacy'])
```



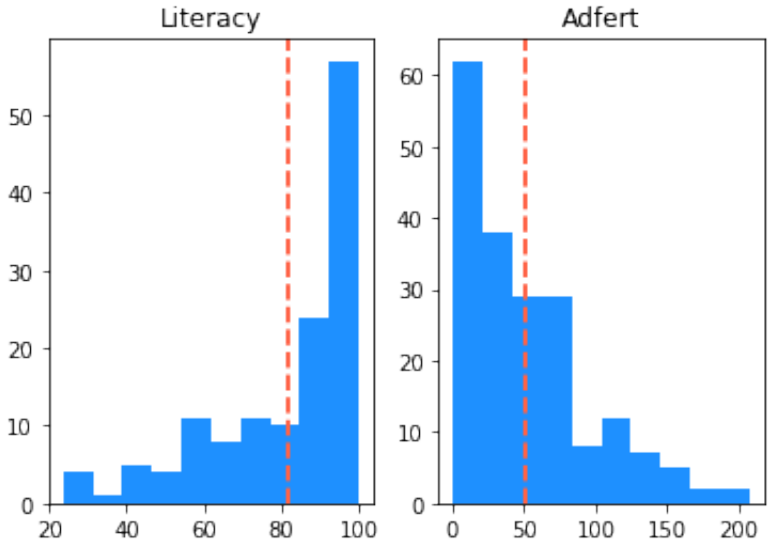
In [22]:
```
plot_hist(df['adfert'])
```



- **Pregunta frecuente** una pregunta frecuente es cómo hacer gráficos pareados:

In [23]:
```
plt.subplot(1, 2, 1)
plot_hist(df['literacy'])
plt.title('Literacy')
plt.subplot(1, 2,2)
plot_hist(df['adfert'])
plt.title('Adfert')
```

Out[23]: Text(0.5,1,'Adfert')

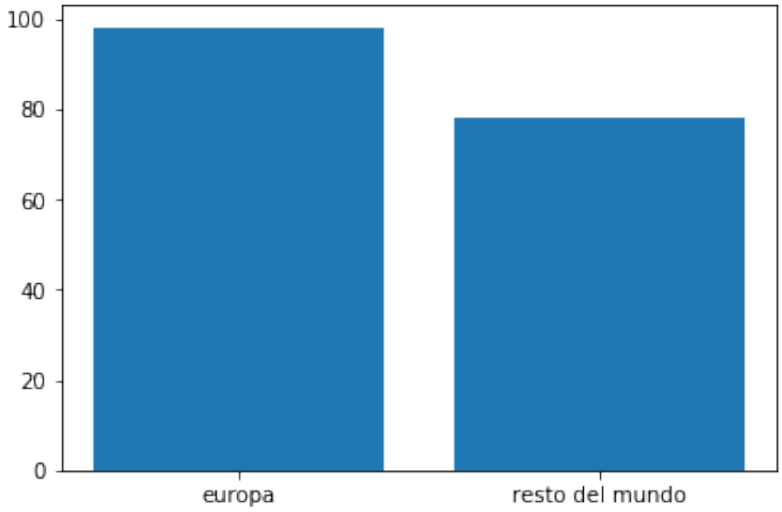


Desafío: Gráfico de barras

- Para realizar un gráfico de barras comparando medias entre dos grupos, realizamos lo siguiente:

In [24]:
```
# guardamos la media de europa
media_europa = df[df['Europe'] == 1]['literacy'].dropna().mean()
# guardamos la media en el resto del mundo
media_resto = df[df['Europe'] == 0]['literacy'].dropna().mean()
# Mediante dos listas, guardamos las etiquetas del eje x, y los datos
a graficar en el eje y.
plt.bar(['europa', 'resto del mundo'],[media_europa, media_resto])
```

Out[24]: <BarContainer object of 2 artists>



Desafío: Diagrama de dispersión y correlación

- Antes de iniciar el desafío, se generar una nueva copia de la base de datos sin valores perdidos. Explicar que por defecto la política de `dropna` es eliminar todas aquellas observaciones que tengan por lo menos un valor perdido.
- Comparar la cantidad de países en cada región antes y después de eliminar valores. Todos los países de oceanía presentan por lo menos 1 dato perdido:

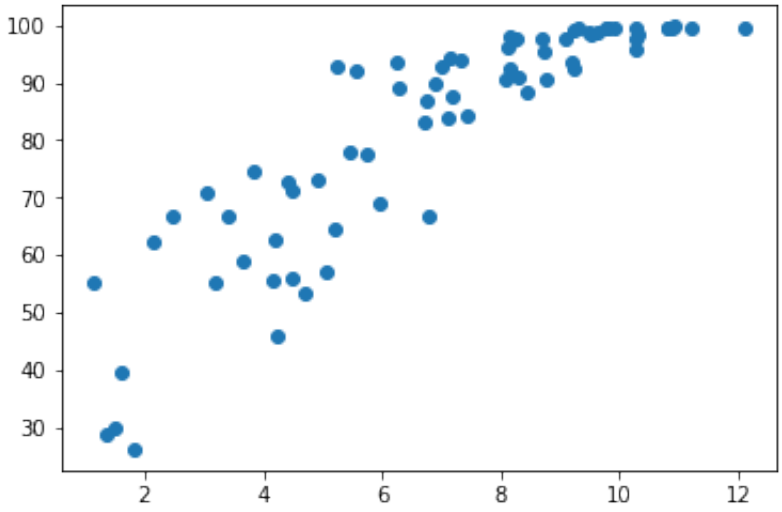
In [25]:
```
df_clean = df.dropna().drop(columns='Oceania')
print(df['region'].value_counts())
print(df_clean['region'].value_counts())
```

```
Africa      52
Asia        49
Europe      43
Americas    35
Oceania     15
Name: region, dtype: int64
Asia        23
Africa      20
Americas    14
Europe      14
Name: region, dtype: int64
```

- Para graficar una diagrama de dispersión, declaramos `x` e `y`. También señalamos que se debe ocupar puntos `'o'`.

In [26]:
```
plt.plot(df_clean['school'], df_clean['literacy'], 'o')
# Imprimimos la correlación entre ambas variables.
df_clean['school'].corr(df_clean['literacy'])
```

Out[26]: 0.884432491376833

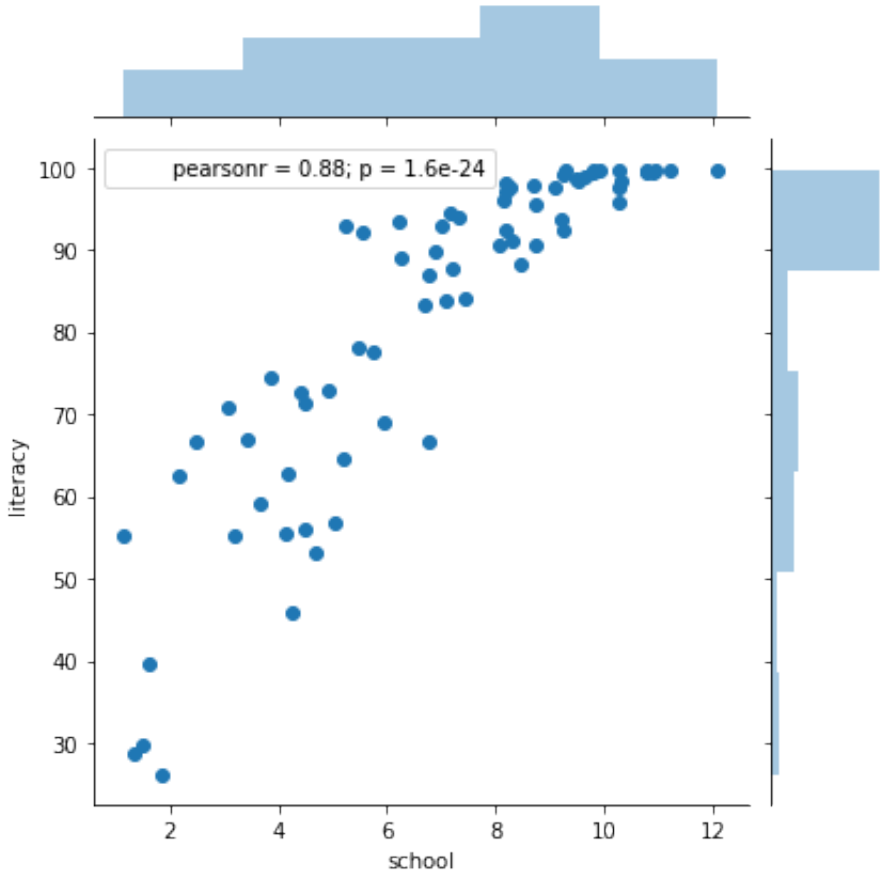


- Desde este punto estamos en pie de hablar de correlación y causalidad.
- Acá también ingresamos la librería `seaborn` que resume buenas prácticas en la visualización de datos.

In [27]:
```
import seaborn as sns
sns.jointplot('school', 'literacy', df_clean)
```

```
/Users/isz/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_ax
es.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has b
een replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/Users/isz/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_ax
es.py:6462: UserWarning: The 'normed' kwarg is deprecated, and has b
een replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

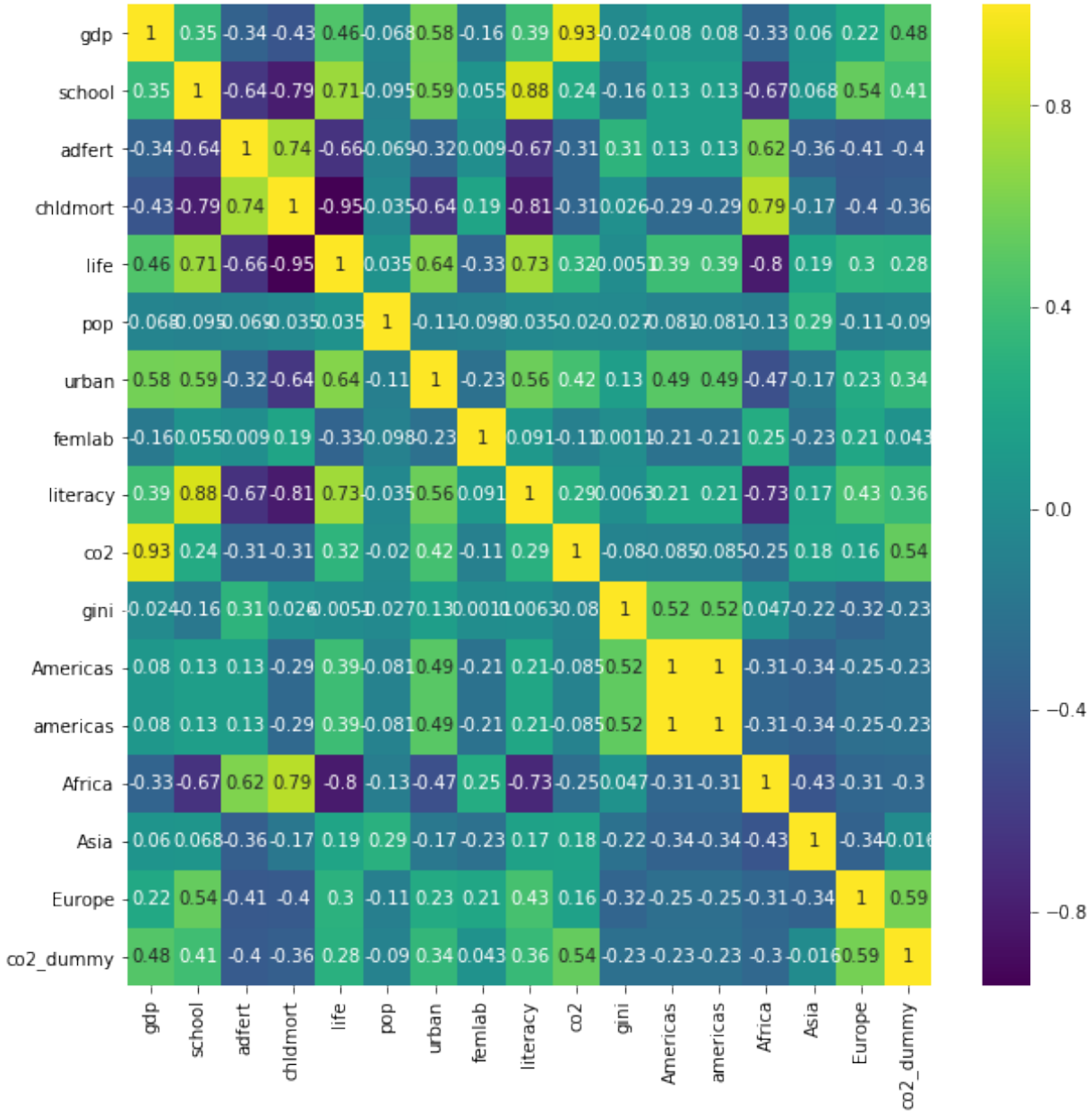
Out[27]: <seaborn.axisgrid.JointGrid at 0x1170f6240>



- Ahora crearemos una matriz de correlaciones entre los datos.

In [28]:
```
corr_mat = df_clean.corr()
plt.figure(figsize=(10, 10))
sns.heatmap(corr_mat, cmap='viridis', annot=True)
```

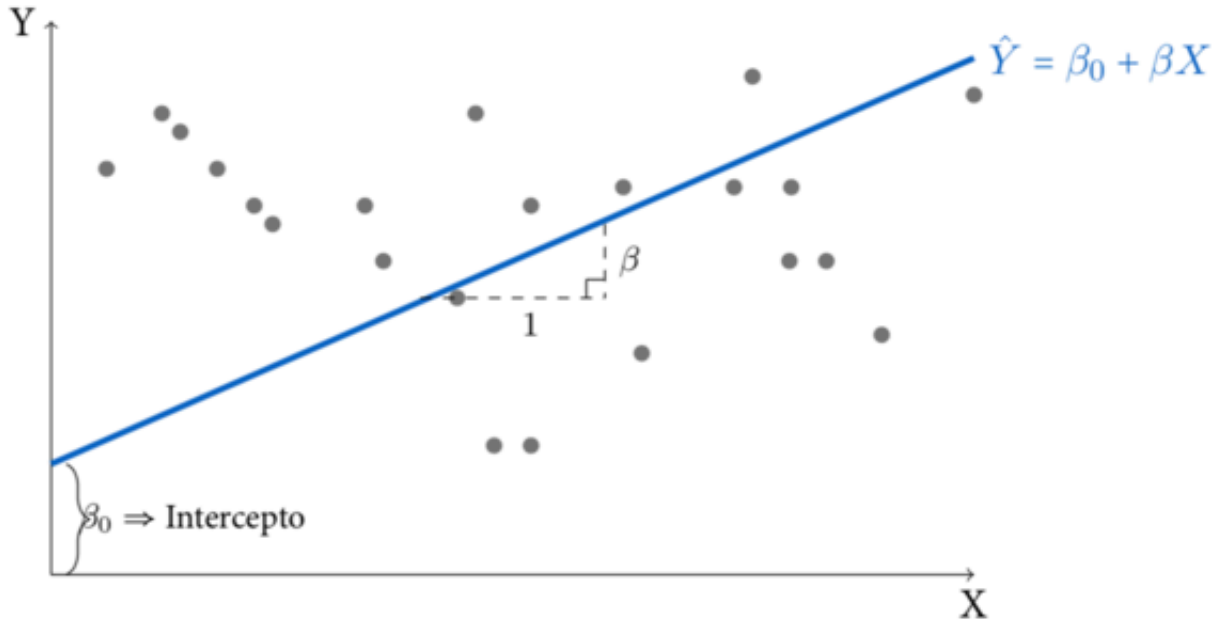
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1cef4f98>



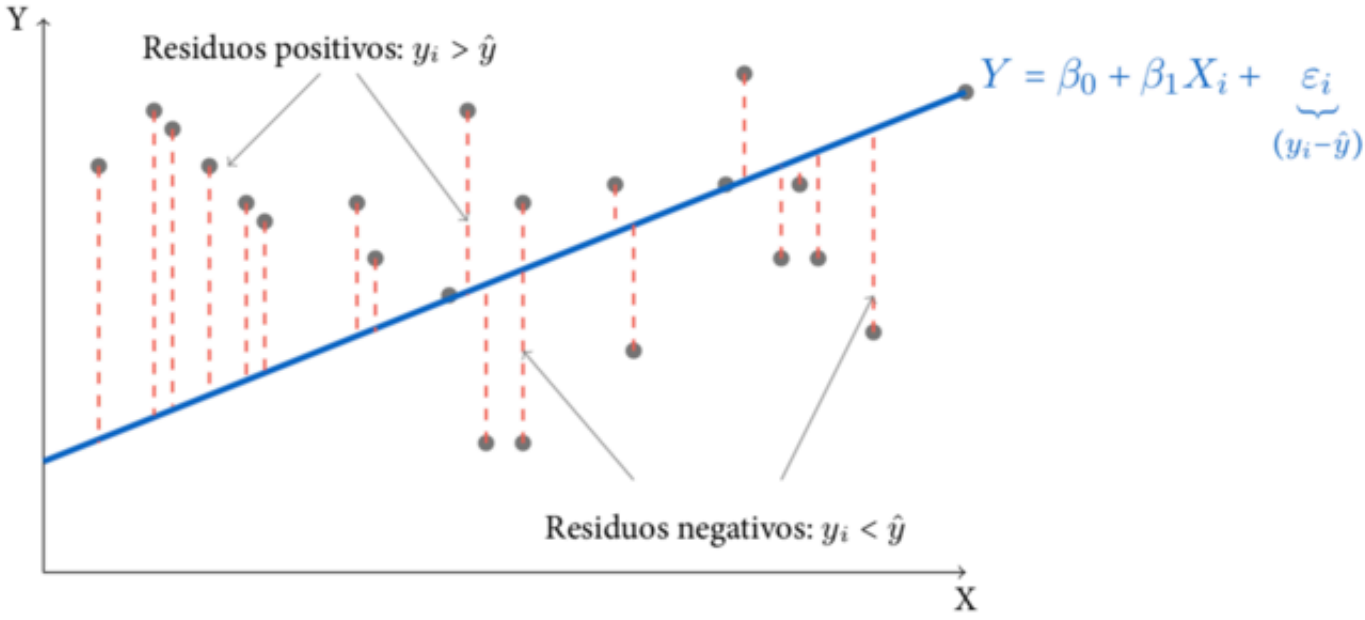
Quinta dinámica: Regresión lineal

- Se debe explicar el objetivo de la regresión lineal, así como sus componentes. Se presentan las siguientes imágenes que se pueden dibujar:

Elementos básicos de la regresión: Intercepto y Pendiente



Método de elección de la recta: Mínimos cuadrados ordinarios



Desafío: Importar `statsmodels`

- En esta parte introducimos a la librería `statsmodels`

In [29]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

Desafío: Generar un objeto de regresión

- Para generar nuestra primera regresión debemos generar un objeto que almacene la siguiente información:
- ¿Qué fórmula entra? En este caso generamos una regresión donde deseamos ver la influencia de `school` (nuestra variable independiente) en los puntajes de `adfert` (nuestra variable dependiente).
- El segundo elemento es dónde se van a buscar estas variables.

In [30]:
```
modelo = smf.ols('adfert ~ school', df_clean)
```

- Posteriormente debemos hacer `fit`, que es la orden de ejecutar el modelo y obtener los parámetros estimados.

In [31]:
```
modelo = modelo.fit()
```

- Para acceder a los elementos del modelo, utilizamos `summary()`. En esta parte cabe destacar los siguientes elementos:
- **R-squared**: El coeficiente de determinación. Responde a la pregunta ¿En qué medida la combinación lineal de parámetros explica la variabilidad de puntajes en mi dependiente?
- **Prob(F-statistic)**: Prueba de hipótesis donde si se rechaza la nula, hay evidencia para la existencia de por lo menos un coeficiente estadísticamente distinto de 0.

- Posteriormente se analizan el intercepto y la pendiente del modelo.

In [32]:
```
modelo.summary()
```

Out[32]:

OLS Regression Results

Dep. Variable:	adfert	**R-squared:**	0.413
Model:	OLS	**Adj. R-squared:**	0.405
Method:	Least Squares	**F-statistic:**	48.59
Date:	Thu, 20 Sep 2018	**Prob (F-statistic):**	1.50e-09
Time:	14:06:25	**Log-Likelihood:**	-356.09
No. Observations:	71	**AIC:**	716.2
Df Residuals:	69	**BIC:**	720.7
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	137.8805	11.588	11.899	0.000	114.764	160.997
school	-10.8342	1.554	-6.970	0.000	-13.935	-7.733

Omnibus:	0.527	**Durbin-Watson:**	1.777
Prob(Omnibus):	0.768	**Jarque-Bera (JB):**	0.144
Skew:	0.067	**Prob(JB):**	0.930
Kurtosis:	3.176	**Cond. No.**	20.0

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

- A partir de este punto se pueden agregar más variables al modelo y observar cambios en los parámetros estimados.

In [33]:
```
modelo_2 = smf.ols('adfert ~ school + Europe', df_clean).fit()
modelo_2.summary()
```

Out[33]:

OLS Regression Results

Dep. Variable:	adfert	**R-squared:**	0.419
Model:	OLS	**Adj. R-squared:**	0.402
Method:	Least Squares	**F-statistic:**	24.53
Date:	Thu, 20 Sep 2018	**Prob (F-statistic):**	9.53e-09
Time:	14:06:27	**Log-Likelihood:**	-355.74
No. Observations:	71	**AIC:**	717.5
Df Residuals:	68	**BIC:**	724.3
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	134.3052	12.384	10.845	0.000	109.594	159.016
school	-10.0037	1.850	-5.406	0.000	-13.696	-6.311
Europe	-10.9254	13.136	-0.832	0.408	-37.139	15.288

Omnibus:	0.809	**Durbin-Watson:**	1.771
Prob(Omnibus):	0.667	**Jarque-Bera (JB):**	0.305
Skew:	0.078	**Prob(JB):**	0.858
Kurtosis:	3.281	**Cond. No.**	25.5

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.