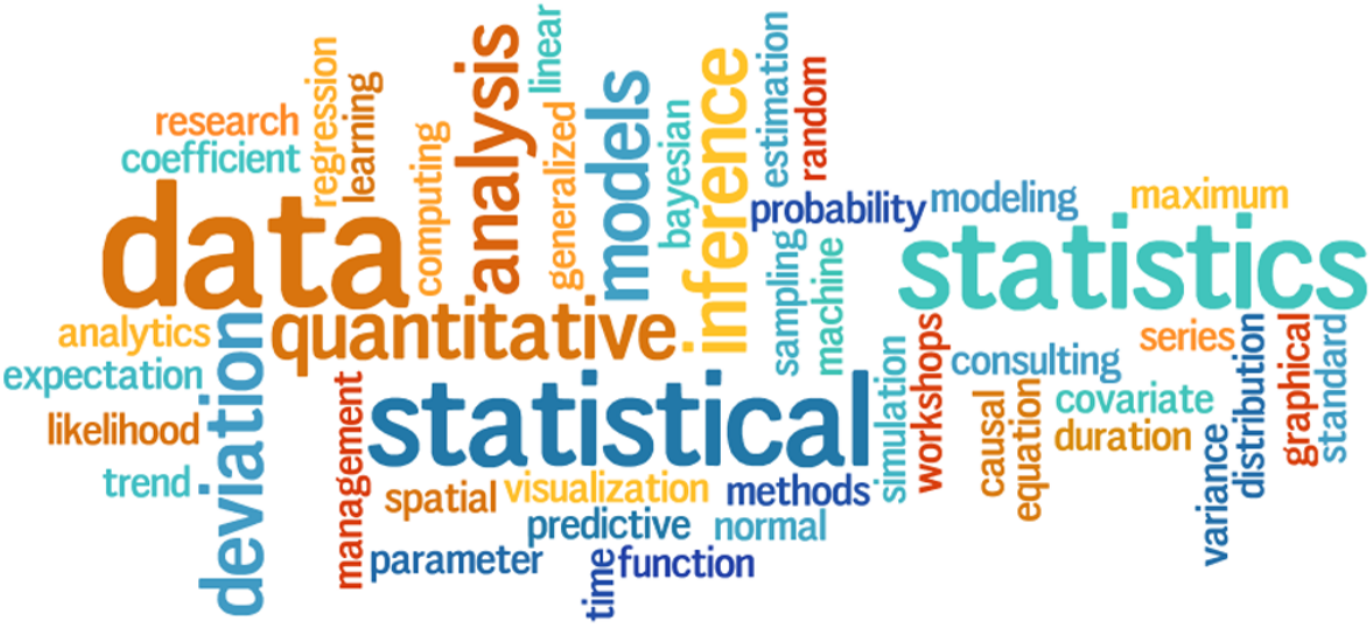
Probabilidad y Estadística con Python

Raul E. Lopez Briega



Actualmente con el boom de la Big Data, tener nociones de *probabilidad* y *estadística* se ha hecho fundamental. En los últimos años ha habido un resurgimiento de todo lo relacionado con *estadística* , *data mining* y *machine learning* empujados principalmente por la explosión de datos con que contamos, estos conceptos combinados forman la base de lo que actualmente se conoce como la *Ciencia de Datos*. Dentro de este contexto, Python es uno de los lenguajes que más nos facilita trabajar con datos. Realizar complejos análisis estadísticos nunca fue tan fácil como con Python!

¿Qué es la Estadística?¶

La *estadística* suele ser definida como la ciencia de aprender de los *datos* o como la ciencia de obtener conclusiones en la presencia de incertidumbre. Se relaciona principalmente con la recolección, análisis e interpretación de *datos*, así como también con la efectiva comunicación y presentación de los resultados basados en esos *datos*. Como por *datos* entendemos a cualquier clase de información grabada, la *estadística* juego un rol importante en muchas disciplinas científicas.

La *estadística* puede ser muy importante para una efectiva toma de decisiones. Existe una gran cantidad de valiosa información escondida entre los *datos*, pero esta información no suele ser fácilmente accesible, la *estadística* nos brinda los principios fundamentales que nos permiten extraer y entender esa información; tambien nos proporciona las herramientas necesarias para verificar la calidad de nuestros *datos* y nuestra información.

La *estadística* suele ser dividida en dos grandes ramas:

1. La *estadística descriptiva*: La cual se dedica a recolectar, ordenar, analizar y representar a un conjunto de datos, con el fin de describir apropiadamente las características de este. Calcula los parámetros estadísticos que describen el conjunto estudiado. Algunas de las herramientas que utiliza son gráficos, medidas de frecuencias, medidas de centralización, medidas de posición, medidas de dispersión, entre otras.

2. La *estadistica inferencial*: La cual estudia cómo sacar conclusiones generales para toda la población a partir del estudio de una muestra, y el grado de fiabilidad o significación de los resultados obtenidos. Sus principales herramientas son el muestreo, la estimación de parámetros y el contraste de hipótesis.

¿Qué es la Probabilidad?¶

La *probabilidad* mide la mayor o menor posibilidad de que se dé un determinado resultado (suceso o evento) cuando se realiza un experimento aleatorio. Para calcular la probabilidad de un evento se toma en cuenta todos los casos posibles de ocurrencia del mismo; es decir, de cuántas formas puede ocurrir determinada situación.Los casos favorables de ocurrencia de un evento serán los que cumplan con la condición que estamos buscando. La *probabilidad* toma valores entre 0 y 1 (o expresados en tanto por ciento, entre 0% y 100%).

La *probabilidad* es a la vez el inverso y complemento para la *estadística*. Dónde la *estadística* nos ayuda a ir desde los *datos* observados hasta hacer generalizaciones sobre como funcionan las cosas; la *probabilidad* funciona en la dirección inversa: si asumimos que sabemos como las cosas funcionan, entonces podemos averiguar la clase de *datos* que vamos a ver y cuan probable es que los veamos.

La *probabilidad* también funciona como complemento de la *estadística* cuando nos proporciona una sólida base para la *estadistica inferencial*. Cuando hay incertidumbre, no sabemos que puede pasar y hay alguna posibilidad de errores, utilizando *probabilidades* podemos aprender formas de controlar la tasa de errores para reducirlos.

Actividades básicas del analisis estadístico¶

Las técnicas *estadísticas* deberían ser vistas como una parte importante de cualquier proceso de toma de dicisiones, permitiendo tomar decisiones estratégicamente informadas que combinen intuición con experiencia y un entendimiento *estadístico* de los *datos* que tenemos disponibles.

Un análisis estadístico suele contener 5 actividades básicas:

1. **Diseño del análisis**: Esta actividad involucra el planeamiento de los detalles para obtener los datos que necesitamos y la generación de la hipótesis a ser evaluada.

2. **Exploración de datos**: En esta actividad nos dedicamos a jugar con nuestros datos, los describimos, los resumimos, realizamos gráficos para mirarlos desde distintos ángulos. Esta exploración nos ayuda a asegurarnos que los datos que obtuvimos son completos y que la etapa de diseño fue correcta.

3. **Armado del modelo**: En esta actividad intentamos armar un modelo que explique el comportamiento de nuestros datos y pueda llegar a hacer predicciones sobre los mismos. La idea es que el modelo pueda describir las propiedades fundamentales de nuestros datos.

4. **Realizar estimaciones**: Aquí vamos a intentar realizar estimaciones basadas en el modelo que armamos anteriormente. También vamos a intentar estimar el tamaño del error que nuestro modelo puede tener en sus predicciones.

5. **Contraste de la hipótesis**: Esta actividad es la que va a producir la decisión final sobre si las predicciones del modelo son correctas y ayudarnos a concluir si los datos que poseemos confirman o rechazan la hipótesis que generamos en la actividad 1.

Conceptos básicos de la estadística descriptiva¶

En *estadística descriptiva* se utilizan distintas medidas para intentar describir las propiedades de nuestros datos, algunos de los conceptos básicos, son:

- **Media aritmética**: La media aritmética es el valor obtenido al sumar todos los *datos* y dividir el resultado entre el número total elementos. Se suele representar con la letra griega μ. Si tenemos una muestra de n valores, x_i, la *media aritmética*, μ, es la suma de los valores divididos por el numero de elementos; en otras palabras:

$$\mu = \frac{1}{n}\sum_i x_i$$

- **Desviación respecto a la media**: La desviación respecto a la media es la diferencia en valor absoluto entre cada valor de la variable estadística y la media aritmética.

$$D_i = |x_i - \mu|$$

- **Varianza**: La varianza es la media aritmética del cuadrado de las desviaciones respecto a la media de una distribución estadística. La varianza intenta describir la dispersión de los *datos*. Se representa como σ^2.

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}$$

- **Desviación típica**: La desviación típica es la raíz cuadrada de la varianza. Se representa con la letra griega σ.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$$

- **Moda**: La moda es el valor que tiene mayor frecuencia absoluta. Se representa con M_0

- **Mediana**: La mediana es el valor que ocupa el lugar central de todos los datos cuando éstos están ordenados de menor a mayor. Se representa con $\tilde{x}$.

- **Correlación**: La correlación trata de establecer la relación o dependencia que existe entre las dos variables que intervienen en una distribución bidimensional. Es decir, determinar si los cambios en una de las variables influyen en los cambios de la otra. En caso de que suceda, diremos que las variables están correlacionadas o que hay correlación entre ellas. La correlación es positiva cuando los valores de las variables aumenta juntos; y es negativa cuando un valor de una variable se reduce cuando el valor de la otra variable aumenta.

- **Covarianza**: La covarianza es el equivalente de la varianza aplicado a una variable bidimensional. Es la media aritmética de los productos de las desviaciones de cada una de las variables respecto a sus medias respectivas.La covarianza indica el sentido de la correlación entre las variables; Si $\sigma_{xy} > 0$ la correlación es directa; Si $\sigma_{xy} < 0$ la correlación es inversa.

$$\sigma_{xy} = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{n}$$

- **Valor atípico**: Un valor atípico es una observación que se aleja demasiado de la moda; esta muy lejos de la tendencia principal del resto de los *datos*. Pueden ser causados por errores

en la recolección de *datos* o medidas inusuales. Generalmente se recomienda eliminarlos del conjunto de datos.

Librerías de Python para probabilidad y estadística¶

Como ya les vengo mostrando en mis anteriores artículos, Python se lleva muy bien con las matemáticas. Además, la comunidad python es tan amplia que solemos encontrar una librería para cualquier problema al que nos enfrentemos. En este caso, los principales módulos que Python nos ofrece para trabajar con *probabilidad* y *estadística*, son:

- **numpy**: El popular paquete matemático de Python, se utiliza tanto que mucha gente ya lo considera parte integral del lenguaje. Nos proporciona algunas funciones estadísticas que podemos aplicar fácilmente sobre los *arrays* de Numpy.
- **scipy.stats**: Este submodulo del paquete científico Scipy es el complemento perfecto para Numpy, las funciones estadisticas que no encontremos en uno, las podemos encontrar en el otro.
- **statsmodels**: Esta librería nos brinda un gran número de herramientas para explorar *datos*, estimar modelos estadísticos, realizar pruebas estadísticas y muchas cosas más.
- **matplotlib**: Es la librería más popular en Python para visualizaciones y gráficos. Ella nos va a permitir realizar los gráficos de las distintas distribuciones de datos.
- **seaborn**: Esta librería es un complemento ideal de matplotlib para realizar gráficos estadísticos.
- **pandas**: Esta es la librería más popular para análisis de *datos* y financieros. Posee algunas funciones muy útiles para realizar *estadística descriptiva* sobre nuestros datos y nos facilita sobremanera el trabajar con series de tiempo.
- **pyMC**: pyMC es un módulo de Python que implementa modelos estadísticos bayesianos, incluyendo la cadena de Markov Monte Carlo(MCMC). pyMC ofrece funcionalidades para hacer el análisis bayesiano lo mas simple posible.

In [1]:

```
# Ejemplos de estadistica descriptiva con python

import numpy as np # importando numpy
from scipy import stats # importando scipy.stats
import pandas as pd # importando pandas

np.random.seed(2131982) # para poder replicar el random
```

In [2]:

```
datos = np.random.randn(5, 4) # datos normalmente distribuidos
datos
```

Out[2]:

```
array([[ 0.46038022, -1.08942528, -0.62681496, -0.63329028],
       [-0.1074033 , -0.88138082, -0.34466623, -0.28320214],
       [ 0.94051171,  0.86693793,  1.20947882, -0.16894118],
       [-0.12790177, -0.58099931, -0.46188426, -0.18148302],
       [-0.76959435, -1.37414587,  1.37696874, -0.18040537]])
```

In [3]:

```
# media arítmetica
datos.mean() # Calcula la media aritmetica de
```

In [4]:

```
np.mean(datos) # Mismo resultado desde la funcion de numpy
```

In [5]:

```
datos.mean(axis=1) # media aritmetica de cada fila
```

Out[5]:

```
array([-0.47228757, -0.40416312,  0.71199682, -0.33806709, -0.23679421])
```

In [6]:

```
datos.mean(axis=0) # media aritmetica de cada columna
```

Out[6]:

```
array([ 0.0791985 , -0.61180267,  0.23061642, -0.2894644 ])
```

In [7]:

```
# mediana
np.median(datos)
```

In [8]:

```
np.median(datos, 0) # media aritmetica de cada columna
```

Out[8]:

```
array([-0.1074033 , -0.88138082, -0.34466623, -0.18148302])
```

In [9]:

```
 # Desviación típica
np.std(datos)
```

In [10]:

```
np.std(datos, 0) # Desviación típica de cada columna
```

Out[10]:

```
array([ 0.58057213,  0.78352862,  0.87384108,  0.17682485])
```

In [12]:

```
np.var(datos, 0) # varianza de cada columna
```

Out[12]:

```
array([ 0.337064  ,  0.6139171 ,  0.76359823,  0.03126703])
```

In [13]:

```
# moda
stats.mode(datos) # Calcula la moda de cada columna
# el 2do array devuelve la frecuencia.
```

Out[13]:

```
(array([[-0.76959435, -1.37414587, -0.62681496, -0.63329028]]),
 array([[ 1.,  1.,  1.,  1.]]))
```

In [14]:

```
datos2 = np.array([1, 2, 3, 6, 6, 1, 2, 4, 2, 2, 6, 6, 8, 10, 6])
stats.mode(datos2) # aqui la moda es el 6 porque aparece 5 veces en el vector.
```

Out[14]:

```
(array([6]), array([ 5.]))
```

In [15]:

```
# correlacion
np.corrcoef(datos) # Crea matriz de correlación.
```

Out[15]:

```
array([[ 1.        ,  0.82333743,  0.15257202,  0.78798675, -0.02292073],
       [ 0.82333743,  1.        , -0.13709662,  0.86873632,  0.41234875],
       [ 0.15257202, -0.13709662,  1.        , -0.47691376,  0.21216856],
       [ 0.78798675,  0.86873632, -0.47691376,  1.        , -0.03445705],
       [-0.02292073,  0.41234875,  0.21216856, -0.03445705,  1.        ]])
```

In [16]:

```
# calculando la correlación entre dos vectores.
np.corrcoef(datos[0], datos[1])
```

Out[16]:

```
array([[ 1.        ,  0.82333743],
       [ 0.82333743,  1.        ]])
```

In [17]:

```
# covarianza
np.cov(datos) # calcula matriz de covarianza
```

Out[17]:

```
array([[ 0.43350958,  0.18087281,  0.06082243,  0.11328658, -0.01782409],
       [ 0.18087281,  0.11132485, -0.0276957 ,  0.06329134,  0.16249513],
       [ 0.06082243, -0.0276957 ,  0.36658864, -0.06305065,  0.15172255],
       [ 0.11328658,  0.06329134, -0.06305065,  0.04767826, -0.00888624],
       [-0.01782409,  0.16249513,  0.15172255, -0.00888624,  1.39495179]])
```

In [18]:

```
# covarianza de dos vectores
np.cov(datos[0], datos[1])
```

Out[18]:

```
array([[ 0.43350958,  0.18087281],
       [ 0.18087281,  0.11132485]])
```

In [19]:

```
# usando pandas
dataframe = pd.DataFrame(datos, index=['a', 'b', 'c', 'd', 'e'],
                         columns=['col1', 'col2', 'col3', 'col4'])
dataframe
```

Out[19]:

	col1	col2	col3	col4
a	0.460380	-1.089425	-0.626815	-0.633290
b	-0.107403	-0.881381	-0.344666	-0.283202
c	0.940512	0.866938	1.209479	-0.168941
d	-0.127902	-0.580999	-0.461884	-0.181483
e	-0.769594	-1.374146	1.376969	-0.180405

In [20]:

```
# resumen estadistadistico con pandas
dataframe.describe()
```

Out[20]:

	col1	col2	col3	col4
count	5.000000	5.000000	5.000000	5.000000
mean	0.079199	-0.611803	0.230616	-0.289464
std	0.649099	0.876012	0.976984	0.197696
min	-0.769594	-1.374146	-0.626815	-0.633290
25%	-0.127902	-1.089425	-0.461884	-0.283202
50%	-0.107403	-0.881381	-0.344666	-0.181483
75%	0.460380	-0.580999	1.209479	-0.180405
max	0.940512	0.866938	1.376969	-0.168941

In [21]:

```
# sumando las columnas
dataframe.sum()
```

Out[21]:

```
col1    0.395993
col2   -3.059013
col3    1.153082
col4   -1.447322
dtype: float64
```

In [22]:

```
# sumando filas
dataframe.sum(axis=1)
```

Out[22]:

```
a   -1.889150
b   -1.616652
c    2.847987
d   -1.352268
e   -0.947177
dtype: float64
```

In [23]:

```
dataframe.cumsum() # acumulados
```

Out[23]:

	col1	col2	col3	col4

	col1	col2	col3	col4
a	0.460380	-1.089425	-0.626815	-0.633290
b	0.352977	-1.970806	-0.971481	-0.916492
c	1.293489	-1.103868	0.237998	-1.085434
d	1.165587	-1.684867	-0.223887	-1.266917
e	0.395993	-3.059013	1.153082	-1.447322

In [24]:

```
# media aritmetica de cada columna con pandas
dataframe.mean()
```

Out[24]:

```
col1    0.079199
col2   -0.611803
col3    0.230616
col4   -0.289464
dtype: float64
```

In [25]:

```
# media aritmetica de cada fila con pandas
dataframe.mean(axis=1)
```

Out[25]:

```
a   -0.472288
b   -0.404163
c    0.711997
d   -0.338067
e   -0.236794
dtype: float64
```

Histogramas y Distribuciones¶

Muchas veces los indicadores de la *estadística descriptiva* no nos proporcionan una imagen clara de nuestros *datos*. Por esta razón, siempre es útil complementarlos con gráficos de las distribuciones de los *datos*, que describan con qué frecuencia aparece cada valor. La representación más común de una distribución es un histograma, que es un gráfico que muestra la frecuencia o probabilidad de cada valor. El histograma muestra las frecuencias como un gráfico de barras que indica cuan frecuente un determinado valor ocurre en el conjunto de datos. El eje horizontal representa los valores del conjunto de datos y el eje vertical representa la frecuencia con que esos valores ocurren.

Las distribuciones se pueden clasificar en dos grandes grupos:

1. Las **distribuciones continuas**, que son aquellas que presentan un número infinito de posibles soluciones. Dentro de este grupo vamos a encontrar a las distribuciones:
    - normal,
    - gamma,
    - chi cuadrado,
    - t de Student,
    - pareto,
    - entre otras
2. Las **distribuciones discretas**, que son aquellas en las que la variable puede tomar un

número determinado de valores. Los principales exponentes de este grupo son las distribuciones:

- poisson,
- binomial,
- hipergeométrica,
- bernoulli
- entre otras

Veamos algunos ejemplos graficados con la ayuda de Python.

Distribución normal¶

La distribución normal es una de las principales distribuciones, ya que es la que con más frecuencia aparece aproximada en los fenómenos reales. Tiene una forma acampanada y es simétrica respecto de un determinado parámetro estadístico. Con la ayuda de Python la podemos graficar de la siguiente manera:

In [26]:

```
# Graficos embebidos.
%matplotlib inline
```

In [27]:

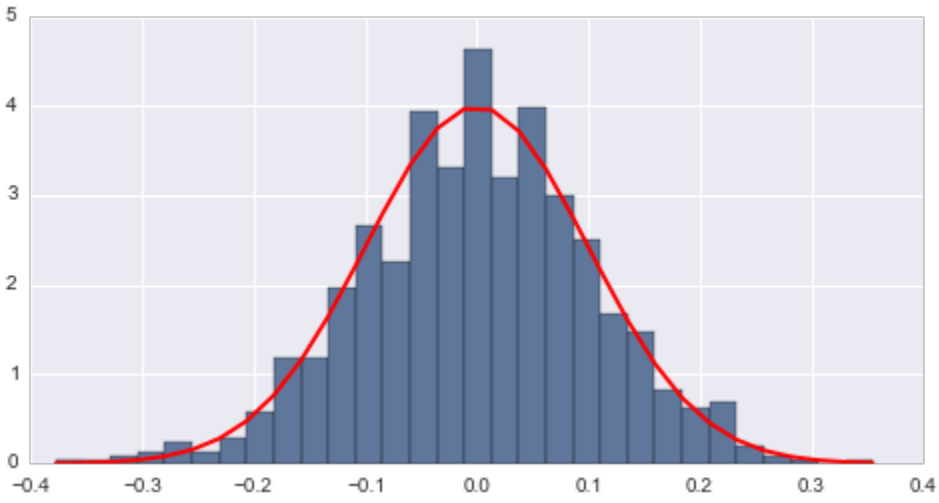
```
import matplotlib.pyplot as plt # importando matplotlib
import seaborn as sns # importando seaborn

# parametros esteticos de seaborn
sns.set_palette("deep", desat=.6)
sns.set_context(rc={"figure.figsize": (8, 4)})
```

In [28]:

```
mu, sigma = 0, 0.1 # media y desvio estandar
s = np.random.normal(mu, sigma, 1000) #creando muestra de datos
```

In [29]:

```
# histograma de distribución normal.
cuenta, cajas, ignorar = plt.hist(s, 30, normed=True)
normal = plt.plot(cajas, 1/(sigma * np.sqrt(2 * np.pi)) *
         np.exp( - (cajas - mu)**2 / (2 * sigma**2) ),
         linewidth=2, color='r')
```



Distribuciones simetricas y asimetricas¶

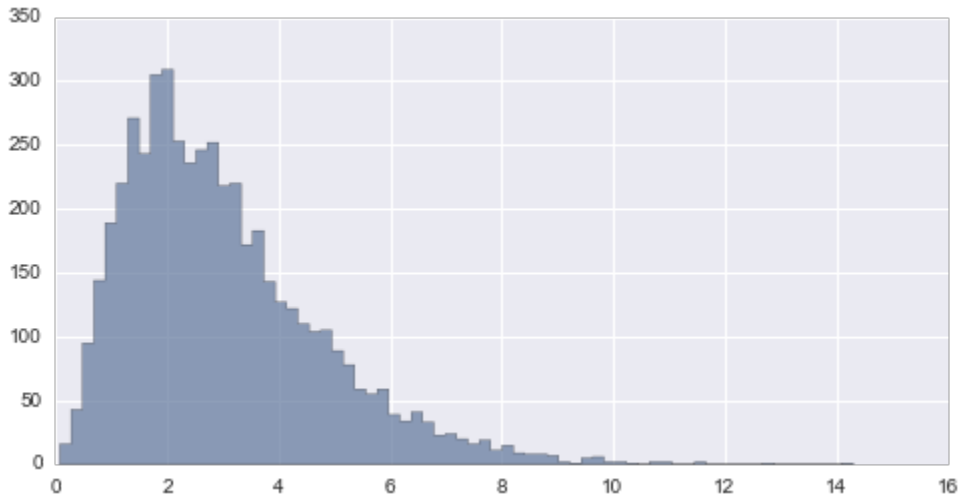
Una distribución es simétrica cuando moda, mediana y media coinciden aproximadamente en sus valores. Si una distribución es simétrica, existe el mismo número de valores a la derecha que a la izquierda de la media, por tanto, el mismo número de desviaciones con signo positivo que con signo negativo.

Una distribución tiene asimetria positiva (o a la derecha) si la "cola" a la derecha de la media es más larga que la de la izquierda, es decir, si hay valores más separados de la media a la derecha. De la misma forma una distribución tiene asimetria negativa (o a la izquierda) si la "cola" a la izquierda de la media es más larga que la de la derecha, es decir, si hay valores más separados de la media a la izquierda.

Las distribuciones asimétricas suelen ser problemáticas, ya que la mayoría de los métodos estadísticos suelen estar desarrollados para distribuciones del tipo normal. Para salvar estos problemas se suelen realizar transformaciones a los datos para hacer a estas distribuciones más simétricas y acercarse a la distribución normal.

In [30]:

```
# Dibujando la distribucion Gamma
x = stats.gamma(3).rvs(5000)
gamma = plt.hist(x, 70, histtype="stepfilled", alpha=.7)
```



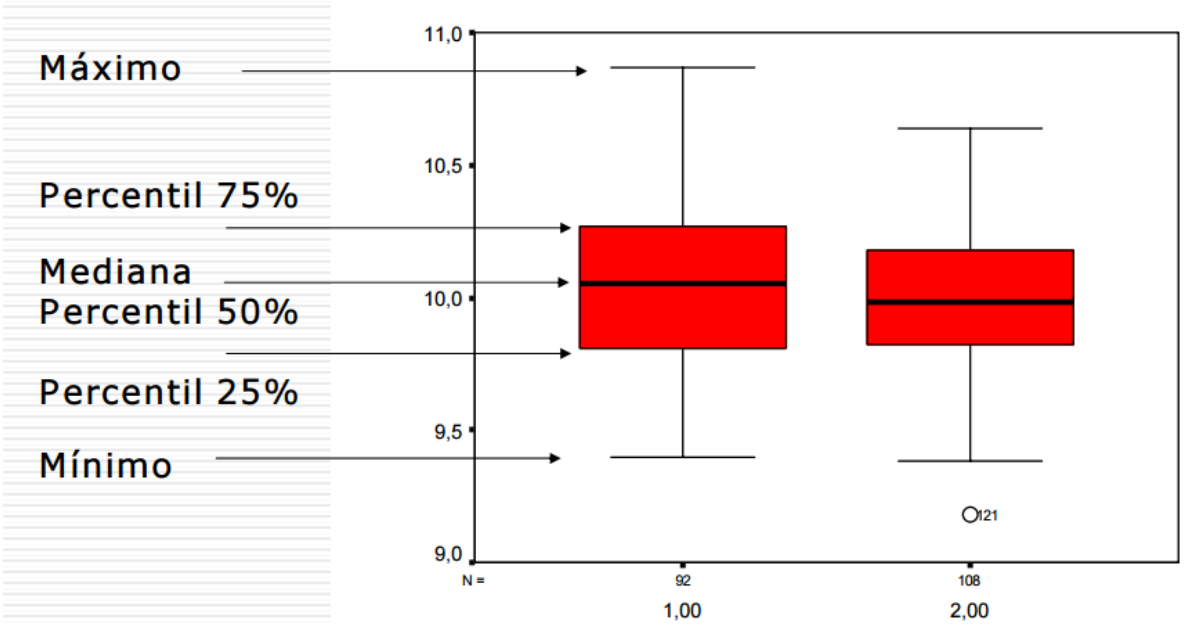
In [31]:

```
# Calculando la simetria con scipy
stats.skew(x)
```

Cuartiles y diagramas de cajas¶

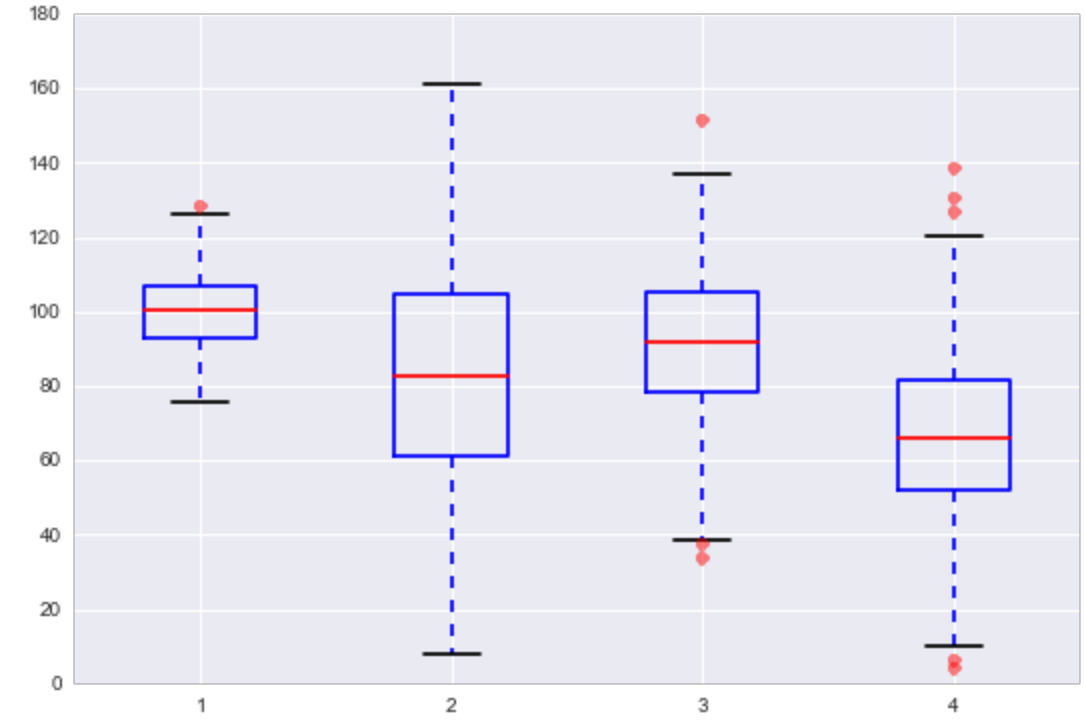
Los **cuartiles** son los tres valores de la variable estadística que dividen a un conjunto de datos ordenados en cuatro partes iguales. Q1, Q2 y Q3 determinan los valores correspondientes al 25%, al 50% y al 75% de los datos. Q2 coincide con la mediana.

Los diagramas de cajas son una presentación visual que describe varias características importantes al mismo tiempo, tales como la dispersión y simetría. Para su realización se representan los tres cuartiles y los valores mínimo y máximo de los datos, sobre un rectángulo, alineado horizontal o verticalmente. Estos gráficos nos proporcionan abundante información y son sumamente útiles para encontrar valores atípicos y comparar dos conjunto de datos.



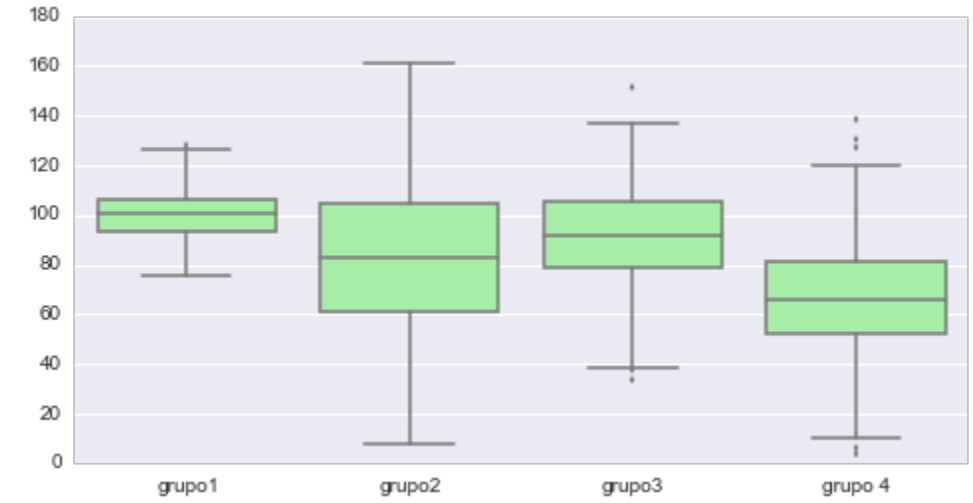
In [32]:

```
# Ejemplo de grafico de cajas en python

datos_1 = np.random.normal(100, 10, 200)
datos_2 = np.random.normal(80, 30, 200)
datos_3 = np.random.normal(90, 20, 200)
datos_4 = np.random.normal(70, 25, 200)

datos_graf = [datos_1, datos_2, datos_3, datos_4]

# Creando el objeto figura
fig = plt.figure(1, figsize=(9, 6))

# Creando el subgrafico
ax = fig.add_subplot(111)

# creando el grafico de cajas
bp = ax.boxplot(datos_graf)

# visualizar mas facile los atípicos
for flier in bp['fliers']:
    flier.set(marker='o', color='red', alpha=0.5)
# los puntos aislados son valores atípicos
```



In [33]:

```
# usando seaborn
sns.boxplot(datos_graf, names=["grupo1", "grupo2", "grupo3", "grupo 4"],
            color="PaleGreen");
```



Regresiones¶

Las **regresiones** es una de las herramientas principales de la *estadistica inferencial*. El objetivo del análisis de regresión es describir la relación entre un conjunto de variables, llamadas variables dependientes, y otro conjunto de variables, llamadas variables independientes o explicativas. Más específicamente, el análisis de regresión ayuda a entender cómo el valor típico de la variable dependiente cambia cuando cualquiera de las variables independientes es cambiada, mientras que se mantienen las otras variables independientes fijas. El producto final del análisis de regresión es la estimación de una función de las variables independientes llamada la **función de regresión**. La idea es que en base a esta función de regresión podamos hacer estimaciones sobre eventos futuros.

La regresión lineal es una de las técnicas más simples y mayormente utilizadas en los análisis de regresiones. Hace suposiciones muy rígidas sobre la relación entre la variable dependiente y y

variable independiente x. Asume que la relación va a tomar la forma:

$$y = \beta_0 + \beta_1 * x$$

Uno de los métodos más populares para realizar regresiones lineales es el de mínimos cuadrados ordinarios (OLS, por sus siglas en inglés), este método es el estimador más simple y común en la que los dos βs se eligen para minimizar el cuadrado de la distancia entre los valores estimados y los valores reales.

Realizar análisis de regresiones en Python es sumamente fácil gracias a statsmodels.

Veamos un pequeño ejemplo utilizando el dataset longley, el cual es ideal para realizar regresiones:

In [34]:

```
# importanto la api de statsmodels
import statsmodels.formula.api as smf
import statsmodels.api as sm

# Creando un DataFrame de pandas.
df = pd.read_csv('https://vincentarelbundock.github.io/Rdatasets/csv/datasets/longley.csv', index_col=0)
df.head() # longley dataset
```

Out[34]:

	GNP.deflator	GNP	Unemployed	Armed.Forces	Population	Year	Employed
1947	83.0	234.289	235.6	159.0	107.608	1947	60.323
1948	88.5	259.426	232.5	145.6	108.632	1948	61.122
1949	88.2	258.054	368.2	161.6	109.773	1949	60.171
1950	89.5	284.599	335.1	165.0	110.929	1950	61.187
1951	96.2	328.975	209.9	309.9	112.075	1951	63.221

In [35]:

```
# utilizando la api de formula de statsmodels
est = smf.ols(formula='Employed ~ GNP', data=df).fit()
est.summary() # Employed se estima en base a GNP.
```

Out[35]:

OLS Regression Results

Dep. Variable:	Employed	**R-squared:**	0.967
Model:	OLS	**Adj. R-squared:**	0.965
Method:	Least Squares	**F-statistic:**	415.1
Date:	Sat, 27 Jun 2015	**Prob (F-statistic):**	8.36e-12
Time:	15:30:24	**Log-Likelihood:**	-14.904
No. Observations:	16	**AIC:**	33.81
Df Residuals:	14	**BIC:**	35.35
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf.	Int.]
Intercept	51.8436	0.681	76.087	0.000	50.382	53.305
GNP	0.0348	0.002	20.374	0.000	0.031	0.038

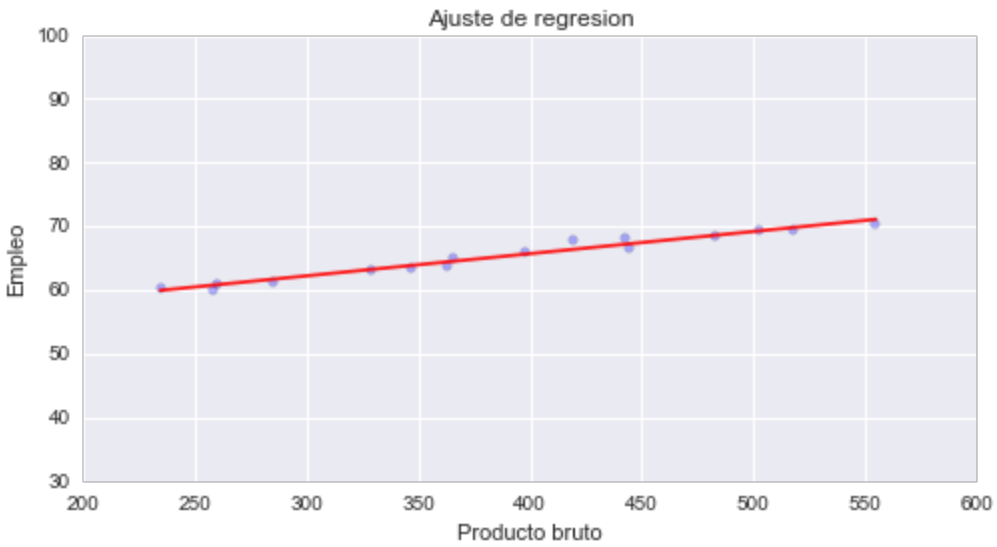
Omnibus:	1.925	**Durbin-Watson:**	1.619
Prob(Omnibus):	0.382	**Jarque-Bera (JB):**	1.215
Skew:	0.664	**Prob(JB):**	0.545
Kurtosis:	2.759	**Cond. No.**	1.66e+03

Como podemos ver, el resumen que nos brinda statsmodels sobre nuestro modelo de regresión contiene bastante información sobre como se ajuste el modelo a los datos. Pasemos a explicar algunos de estos valores:

- **Dep. Variable:** es la variable que estamos estimasdo.
- **Model:** es el modelo que estamos utilizando.
- **R-squared:** es el coeficiente de determinación, el cual mide cuan bien nuestra recta de regresion se aproxima a los datos reales.
- **Adj. R-squared:** es el coeficiente anterior ajustado según el número de observaciones.
- **[95.0% Conf. Int.]:** Los valores inferior y superior del intervalo de confianza del 95%.
- **coef:** el valor estimado del coeficiente.
- **std err:** el error estándar de la estimación del coeficiente.
- **Skew:** una medida de la asimetria de los datos sobre la media.
- **Kurtosis:** Una medida de la forma de la distribución. La curtosis compara la cantidad de datos cerca de la media con los que están más lejos de la media(en las colas).

In [36]:

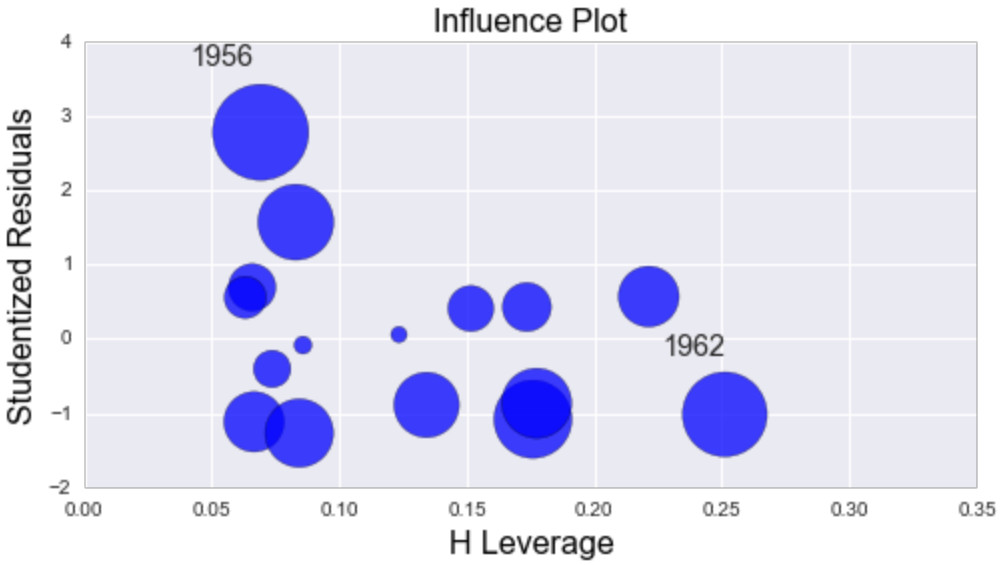
```
# grafico de regresion. que tanto se ajusta el modelo a los datos.
y = df.Employed  # Respuesta
X = df.GNP  # Predictor
X = sm.add_constant(X) # agrega constante

X_1 = pd.DataFrame({'GNP': np.linspace(X.GNP.min(), X.GNP.max(), 100)})
X_1 = sm.add_constant(X_1)

y_reg = est.predict(X_1) # estimacion

plt.scatter(X.GNP, y, alpha=0.3)  # grafica los puntos de datos
plt.ylim(30, 100)  # limite de eje y
plt.xlabel("Producto bruto") # leyenda eje x
plt.ylabel("Empleo") # leyenda eje y
plt.title("Ajuste de regresion") # titulo del grafico
reg = plt.plot(X_1.GNP, y_reg, 'r', alpha=0.9)  # linea de regresion
```



In [37]:

```
# grafico de influencia
```

```
from statsmodels.graphics.regressionplots import influence_plot

inf =influence_plot(est)
```



Este último gráfico nos muestra el apalancamiento y la influencia de cada caso.

La estadística bayesiana¶

La *estadística bayesiana* es un subconjunto del campo de la *estadística* en la que la evidencia sobre el verdadero estado de las cosas se expresa en términos de grados de creencia. Esta filosofía de tratar a las creencias como probabilidad es algo natural para los seres humanos. Nosotros la utilizamos constantemente a medida que interactuamos con el mundo y sólo vemos verdades parciales; necesitando reunir pruebas para formar nuestras creencias.

La diferencia fundamental entre la *estadística clásica* (frecuentista) y la *bayesiana* es el concepto de *probabilidad*. Para la *estadística clásica* es un concepto objetivo, que se encuentra en la naturaleza, mientras que para la *estadística bayesiana* se encuentra en el observador, siendo así un concepto subjetivo. De este modo, en *estadística clásica* solo se toma como fuente de información las muestras obtenidas. En el caso *bayesiano*, sin embargo, además de la muestra también juega un papel fundamental la información previa o externa que se posee en relación a los fenómenos que se tratan de modelar.

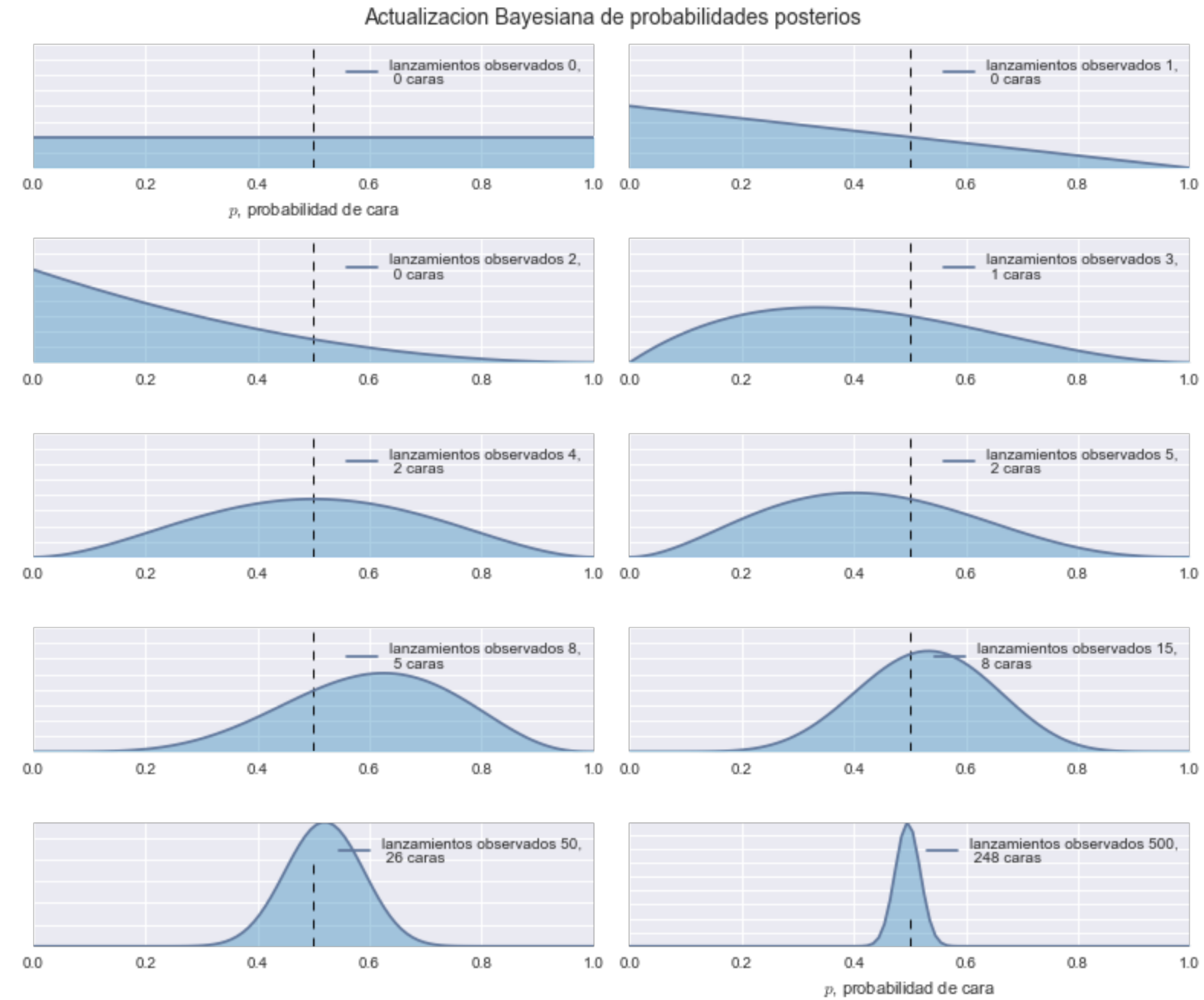
La *estadística bayesiana* está demostrando su utilidad en ciertas estimaciones basadas en el conocimiento subjetivo a priori y el hecho de permitir revisar esas estimaciones en función de la evidencia empírica es lo que está abriendo nuevas formas de hacer conocimiento. Una aplicación de esto son los clasificadores bayesianos que son frecuentemente usados en implementaciones de filtros de correo basura, que se adaptan con el uso. La *estadística bayesiana* es un tema muy interesante que merece un artículo en sí mismo.

Para entender más fácilmente como funciona la *estadística bayesiana* veamos un simple ejemplo del lanzamiento de una moneda. La idea principal de la inferencia bayesiana es que la noción de *probabilidad* cambia mientras más *datos* tengamos.

In [38]:

```
sns.set_context(rc={"figure.figsize": (11, 9)})

dist = stats.beta
n_trials = [0, 1, 2, 3, 4, 5, 8, 15, 50, 500]
data = stats.bernoulli.rvs(0.5, size=n_trials[-1])
x = np.linspace(0, 1, 100)
```

```
for k, N in enumerate(n_trials):
    sx = plt.subplot(len(n_trials) / 2, 2, k + 1)
    plt.xlabel("$p$, probabilidad de cara") \
        if k in [0, len(n_trials) - 1] else None
    plt.setp(sx.get_yticklabels(), visible=False)
    heads = data[:N].sum()
    y = dist.pdf(x, 1 + heads, 1 + N - heads)
    plt.plot(x, y, label="lanzamientos observados %d,\n %d caras" % (N, heads))
    plt.fill_between(x, 0, y, color="#348ABD", alpha=0.4)
    plt.vlines(0.5, 0, 4, color="k", linestyles="--", lw=1)

    leg = plt.legend()
    leg.get_frame().set_alpha(0.4)
    plt.autoscale(tight=True)

plt.suptitle("Actualizacion Bayesiana de probabilidades posterios",
             y=1.02,
             fontsize=14)

plt.tight_layout()
```



Como el gráfico de arriba muestra, cuando empezamos a observar nuevos *datos* nuestras probabilidades posteriores comienzan a cambiar y moverse. Eventualmente, a medida que

observamos más y más datos (lanzamientos de monedas), nuestras probabilidades se acercan más y más hacia el verdadero valor de p = 0.5 (marcado por una línea discontinua).

Aquí termina este tutorial, espero que les haya sido util.

Saludos!

Este post fue escrito utilizando IPython notebook. Pueden descargar este notebook o ver su version estática en nbviewer.

Este articulo fue escrito utilizando Jupyter notebook. Presionar aquí para la versión interactiva:
launch binder