

## Variables aleatorias

<b>Variables aleatorias</b>	<b>1</b>
¿Qué aprenderás?	2
Introducción	2
Analizando el Índice de Desarrollo Humano	4
Sobre iterrows y otros métodos	7
Histograma: Visualizando el comportamiento de una variable	9
Sobre loc e iloc	10
Preguntas de proceso	14
¿Qué he aprendido hasta ahora?	14
¿Hay algo que me está dificultando? Si es así, retrocede un poco y revísalo nuevamente antes de seguir avanzando	14
La distribución normal: Caracterizando nuestras variables	15
¿Y cómo comprobamos la normalidad de una variable?	15
¿Qué podemos esperar de los parámetros de una distribución normal?	18
Una variable no-normal y una posible transformación	19
Una variante importante: La distribución normal estandarizada	21
Puntajes Z	22
¿Cuál es el IDH de Chile?	23
Variables Discretas: ¿Qué pasa cuando nuestras variables miden atributos?	28
Ensayo de Bernoulli	29
Distribución Binomial: Analizando una serie de ensayos de Bernoulli	31
Sobre enumerate.	33
¿Qué pasa cuando mantenemos nuestra probabilidad fija condicionada al tamaño muestral?	34
Reflexiona	35
Referencias	35



**¡Comencemos!**

## ¿Qué aprenderás?

- Hacer uso de métodos de pandas para segmentar columnas y filas.
- Hacer uso de los métodos `iterrows` e `iteritems` para implementar loops en pandas.
- Implementar `enumerate` en loops.
- Conocer las principales convenciones en la visualización de resultados en histogramas, gráficos de punto y barras.
- Generar simulaciones de la distribución normal.
- Conocer las principales aplicaciones de las distribuciones.
- Calcular e interpretar puntajes z.

## Introducción

En la presente unidad, aprenderemos sobre cómo se comportan las variables y las razones que subyacen a dicho accionar. Para ello, más adelante haremos uso de gráficos para observar la forma en que se distribuyen de forma empírica, y lograr definir su comportamiento mediante las variables aleatorias.

También utilizaremos la base de datos *Quality of Government*, base gestionada de forma periódica por la Universidad de Gotemburgo. Las unidades de medición en esta base corresponden a 194 países, recolectando los últimos datos de enero del 2018. Este tipo de base de datos se conoce como *transversal*, a diferencia de las bases *longitudinales*, que son las unidades de medición seguidas por un periodo finito.

La base de datos es un compendio de alrededor de 1900 variables que miden las causas y consecuencias de un buen gobierno en materias económicas, salud pública, medio ambiente, salud institucional, corrupción, entre otras.

**¡Vamos con todo!**



## Analizando el Índice de Desarrollo Humano

En esta primera parte analizaremos el Índice de Desarrollo Humano de diversos países. Dicho índice fue creado por el Programa de las Naciones Unidas para el Desarrollo buscando generar una medición alternativa para el desarrollo de un país, en oposición a las mediciones clásicas, como el crecimiento económico.

Este índice es un resumen de las facilidades con las que cuenta un país para entregarle a sus habitantes una vida larga y saludable, una educación de calidad que les permita comprender su entorno y tener una vida digna, en términos generales. Para ello, la dimensión de la salud se mide por la esperanza de vida al nacer, la educación, mediante el promedio de años de escolaridad en la población sobre los 25 años y la cantidad esperada de años de escolaridad en la población infantil. La dimensión del nivel de vida se mide con el producto nacional bruto per cápita. Todas estas, posteriormente, son normalizadas y promediadas.

El índice va entre 0 (nivel de desarrollo humano precario) a 1 (nivel de desarrollo humano perfecto).

Para poder hacer este cálculo, primero importamos las librerías clásicas que necesitamos para el análisis.

```
%matplotlib inline
# importamos la triada de Data Science
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# importamos scipy.stats que ayudará a generar distribuciones
import scipy.stats as stats
# archivos con funciones de visualización
import lec3_graphs as gfx

plt.style.use('seaborn-whitegrid') # Gráficos estilo seaborn
plt.rcParams["figure.figsize"] = (10, 6) # Tamaño gráficos
```

Ahora, ingresamos la base de datos que se encuentra bajo el nombre `qog_std_cs_jan18.csv`.

Mediante `shape` observamos que la tabla de datos tiene 194 filas y 1883 columnas.

```
df = pd.read_csv('qog_std_cs_jan18.csv')  
df.shape
```

```
(194, 1882)
```

```
df['undp_hdi'].describe().round(3)
```

```
count    186.000  
mean      0.696  
std       0.155  
min       0.347  
25%      0.567  
50%      0.726  
75%      0.805  
max       0.948  
Name: undp_hdi, dtype: float64
```

Al ejecutar `describe` en la `Serie`, se obtiene una media de .69, por lo que podemos decir que a nivel global el nivel de desarrollo está por sobre el punto de desarrollo medio (0.5), por lo que el nivel de desarrollo a nivel mundial se acerca más a ser 'bueno'. El valor mínimo registrado es de .347, sustancialmente inferior a la media global, mientras que el valor máximo es de .948. Hay que considerar que `describe` informa que hay 186 observaciones con valores registrados, que no concuerdan con las 194 observaciones originales.

Para obtener una lista de los países que no registran niveles de desarrollo humano, podemos hacer lo siguiente:

```
# Generamos una nueva columna que devuelve un booleano si es que la
observación no está presente
df['flagnull'] = df['undp_hdi'].isnull()

# dentro de cada fila de nuestra base
for i, r in df.iterrows():
    # si flagnull es verdadero
    if r['flagnull'] is True:
        # imprime el nombre
        print(r['cname'])

# Obs: Hay una forma menos 'memory hungry' de hacer esto con el metodo
'loc' que veremos mas adelante:
df.loc[df['undp_hdi'].isnull()]['cname'].unique()
```

```
Taiwan
Korea, North
Monaco
Nauru
Marshall Islands
San Marino
Somalia
Tuvalu

array(['Taiwan', 'Korea, North', 'Monaco', 'Nauru', 'Marshall Islands',
      'San Marino', 'Somalia', 'Tuvalu'], dtype=object)
```

La mayoría de los países corresponden a protectorados o países con situaciones de gobiernos anómalos (Dictaduras, guerras, etc), además de muchas veces no tener los suficientes recursos como para invertir en institutos de estadísticas gubernamentales.

Para excluirlos de los datos, pandas ofrece el método `dropna`. Funciona tanto en los `DataFrame` como en las `Series`. Hay que destacar que utilizar `dropna` en `DataFrame` requiere un poco más de cautela en el mecanismo de omisión de datos.

## Sobre iterrows y otros métodos

Si buscamos iterar un DataFrame con los métodos clásicos, obtendremos resultados no muy satisfactorios. Hagamos un print de los elementos contenidos en el DataFrame.

```
for i in df:
    print(i)
ccode
cname
ccodealp
ccodecow
...output truncado
```

Por defecto cuando ejecutamos un loop en un DataFrame, este infiere que recorreremos por el nombre de cada columna. Sin embargo, los DataFrames tienen métodos para especificar el tipo de loop a ejecutar.

- **iteritems()**: Instruye al loop iterar sobre un par de llaves (nombre de la columna) y por los valores (elementos contenidos dentro de cada columna).

```
demo = pd.DataFrame(np.random.randn(3, 4), columns=['col1', 'col2',
'col3', 'col4'])
for llave, valor in demo.iteritems():
    print(llave, valor)
col1 0    -0.302614
1     0.582780
2    -1.310241
Name: col1, dtype: float64
col2 0    -1.298668
1    -0.664782
2     0.858832
Name: col2, dtype: float64
col3 0     0.051574
1    -0.577034
2     0.902375
Name: col3, dtype: float64
col4 0     1.393319
1    -0.150681
2    -1.635101
Name: col4, dtype: float64
```

- **iterrows()**: Instruye al loop iterar por el índice (posición de la fila en el DataFrame) en conjunto a los valores (elementos contenidos dentro de cada fila).

```
for indice, valor in demo.iterrows():  
    print(indice, valor)  
0 col1    -0.302614  
  col2    -1.298668  
  col3     0.051574  
  col4     1.393319  
Name: 0, dtype: float64  
1 col1     0.582780  
  col2    -0.664782  
  col3    -0.577034  
  col4    -0.150681  
Name: 1, dtype: float64  
2 col1    -1.310241  
  col2     0.858832  
  col3     0.902375  
  col4    -1.635101  
Name: 2, dtype: float64
```

## Histograma: Visualizando el comportamiento de una variable

Cuando buscamos visualizar cómo se comporta una variable continua, una de las formas exploratorias más comunes es mediante un **histograma**, que es una representación de la *distribución empírica* de la variable.

Para ello haremos uso de `matplotlib`, una librería orientada a "*hacer fácil las cosas fáciles y posibles las cosas imposibles*" mediante una sintaxis clara y concisa respecto a la forma en que se pasan argumentos y se pueden modificar gráficos. Está bien integrado con `pandas` y `numpy`, facilitando la interacción entre estos elementos.

Para generar un histograma, los valores continuos se agrupan en *casillas* (bins) y posteriormente se cuentan la cantidad de observaciones dentro de cada casilla. El comportamiento de `matplotlib` por defecto es calcular de forma automática las casillas.

Si observamos el histograma *índice de desarrollo humano*, en el eje X encontramos el *rango de valores de la variable*, y en el eje Y encontramos la *frecuencia* (la cantidad de repeticiones).

Existe una gran cantidad de casos con un índice de desarrollo humano igual o superior a .70, pero también encontramos otro grupo de observaciones con valores cercanos a .50. Este gráfico revela algo que ya sabíamos: existe el mundo desarrollado y el mundo subdesarrollado.

```
# vamos a eliminar los datos perdidos en la columna undp_hdi con  
dropna()  
hdi_dropna = df['undp_hdi'].dropna();  
plt.hist(hdi_dropna, color='grey', alpha=.4)  
plt.title("Distribución empírica del índice de la variable undp_hdi");
```



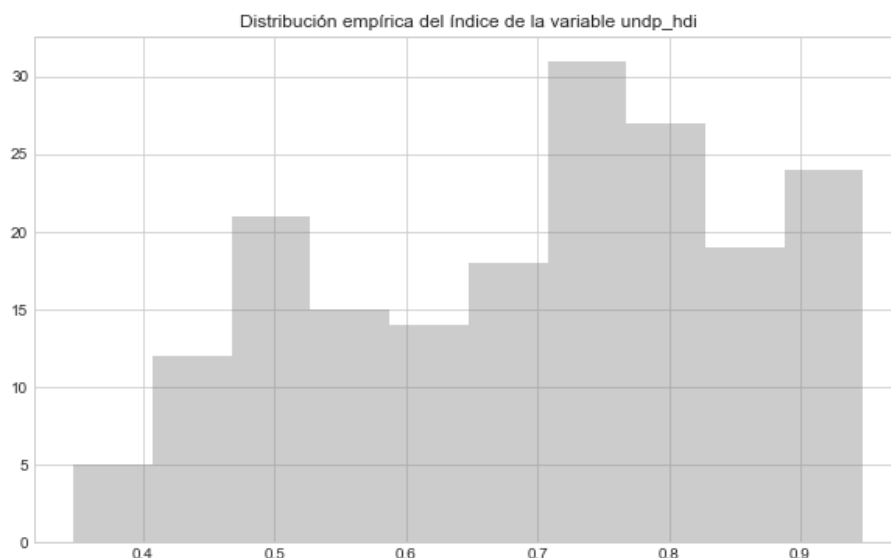


Imagen 1. Distribución empírica del índice de la variable undp\_hdi.  
Fuente: Desafío Latam.

Ya caracterizando la media del índice de desarrollo humano, estamos interesados en ver los niveles de desarrollo en distintas regiones. Para ello necesitamos calcular las medias a nivel de grupo y posteriormente graficarlas.

Nuestro primer punto es separar las columnas a procesar en un nuevo objeto. Ésto lo logramos mediante al método `loc` que tiene un `DataFrame`.

Para consultar por filas y columnas específicas, `DataFrame` dispone de los métodos `loc` (para especificar etiquetas de los ejes) e `iloc` (para especificar los números integrales asociados a los ejes).

## Sobre `loc` e `iloc`

- La forma canónica para `loc` e `iloc` es la siguiente:

```
<pandas.DataFrame>.(loc|iloc)[<filas>, <columnas>]
```

- Para las consultas con dos o más en una de las posiciones, se pueden pasar listas como argumentos:

```
<pandas.DataFrame>.(loc|iloc)[['elementos', 'de', 'fila'],  
                                ['elementos', 'de', 'columna']]
```

- Para este caso, nuestro segmentador especifica que preservaremos todas las filas (mediante el operador ":" ). Posteriormente pasamos una lista con el nombre de todas las columnas a segmentar.

```
hdi_group = df.loc[:, ['cname', 'ccode', 'undp_hdi', 'ht_region' ] ]  
hdi_group.shape
```

```
(194, 4)
```

Nuestro nuevo objeto posee 194 observaciones y 4 columnas.

Resulta que nuestra variable indicadora sobre la región de cada país necesita renombrar cada dígito. Sabiendo que el libro de códigos nos entrega las etiquetas, podemos re codificarlas con `replace`.

```
hdi_group['region_recod'] = hdi_group['ht_region'].replace([1, 2, 3, 4,  
5, 6, 7, 8, 9, 10],  
['EastEurope', 'LatAm', 'NorthAfrica',  
'SubSaharian', 'WesternDem', 'EastAsia',  
'SouthEastAsia', 'SouthAsia', 'Pacific', 'Caribbean' ])
```

```
print("La media del índice de desarrollo humano para nuestra muestra es  
de : ",round(hdi_group['undp_hdi'].mean(),2))
```

```
La media del índice de desarrollo humano para nuestra muestra es de :  
0.7
```

Sabiendo que si nuestro índice de desarrollo humano va de 0 a 1, donde menores valores implican un nivel más precario de desarrollo humano, la media se posiciona cercana al .70. Esto sugiere que a nivel global no deberíamos estar tan mal.

Ahora, el contrapunto es que la media no es ponderada y asume iguales pesos de las observaciones.

Siendo así, lo que necesitamos es calcular puntajes a nivel regional, lo que se logra mediante `groupby`. Esta función mapea todos los valores únicos de una columna en grupos

(en este caso, las 10 posibles regiones existentes), y permite pasar funciones a estos grupos.

Para este caso, y posterior a la declaración de nuestra variable aglutinadora `'region_recod'`, vamos a solicitar las medias agrupadas del nivel de desarrollo humano.

```
group_mean =  
round(hdi_group.groupby('region_recod')['undp_hdi'].mean(),2)  
group_mean
```

```
region_recod  
Caribbean      0.74  
EastAsia        0.82  
EastEurope      0.78  
LatAm           0.72  
NorthAfrica     0.75  
Pacific         0.64  
SouthAsia       0.61  
SouthEastAsia   0.70  
SubSaharian     0.51  
WesternDem      0.90  
Name: undp_hdi, dtype: float64
```

Al llamar el objeto creado, el resultado es una serie con floats que entrega el porcentaje regional.

Si presentamos esta información de esta manera, probablemente pasaremos más tiempo buscando diferenciar cada región entre sí. Por tanto, es preferible presentar estos datos mediante un **dotplot**.

En nuestra primera línea realizaremos un gráfico de puntos, donde el eje x serán los valores del objeto, e "y" serán las llaves (o etiquetas) asociadas a nuestros valores.

```
plt.title('Posición de distintas zonas geográficas en cuanto a HDI')  
plt.plot(group_mean.values, group_mean.index, 'o', color = 'grey')  
plt.axvline(hdi_group['undp_hdi'].mean(), color = 'tomato', linestyle =  
'--');
```

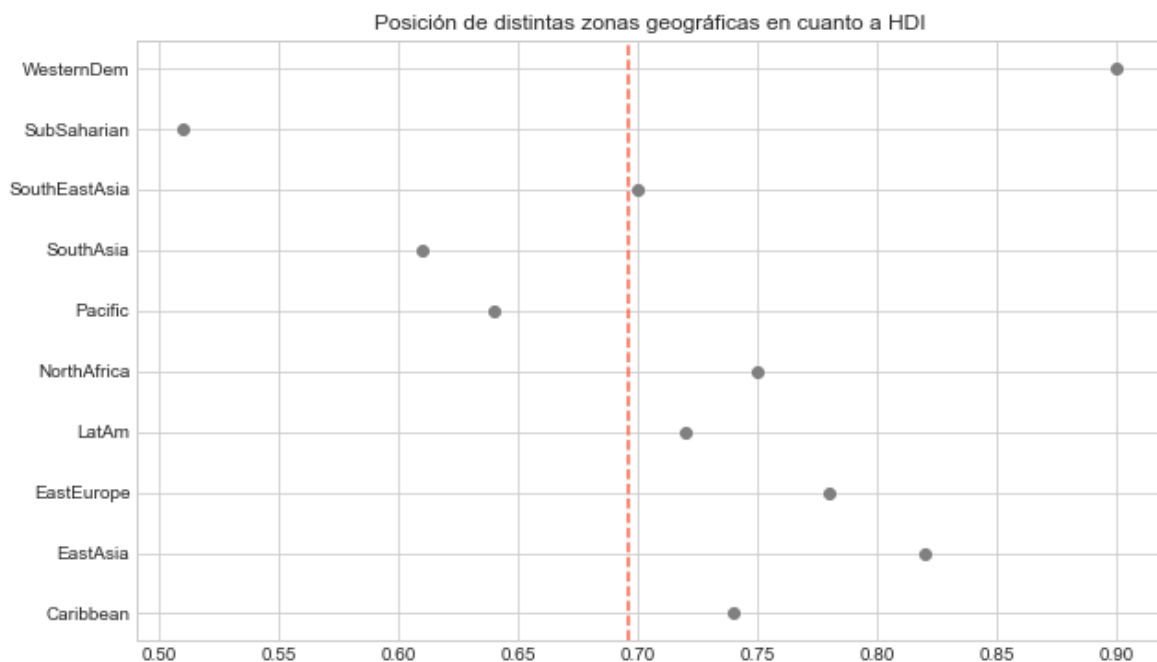


Imagen 2. Posición de distintas zonas geográficas en cuanto a HDI.

Fuente: Desafío Latam.

Nuestro gráfico aclara el hecho que, en promedio, las democracias avanzadas (Europa Occidental y Estados Unidos, Canadá y Australia) alcanzan niveles de desarrollo humano sobre el .90. A diferencia, las regiones como África subsahariana y las Islas del pacífico tienen condiciones de vida más precarias e inestables.

Para clarificar aún más este punto, agregamos la media de la variable a nuestro histograma. Para ello utilizaremos el método `axvline` de `matplotlib`, que inserta una línea vertical. También agregaremos una línea vertical que señale la mitad de la escala de la variable, no su mediana. Así observaremos cuántos países alcanzan una situación deficiente.

```
#guardemos la media en un objeto
hdi_mean = hdi_dropna.mean()

plt.hist(hdi_dropna, color='grey', alpha=.4)
plt.axvline(hdi_mean, color='dodgerblue', linestyle='--', lw=2)
plt.axvline(0.5, color='tomato', linestyle='--', lw=2)
plt.title('Histograma de la variable HDI');
```

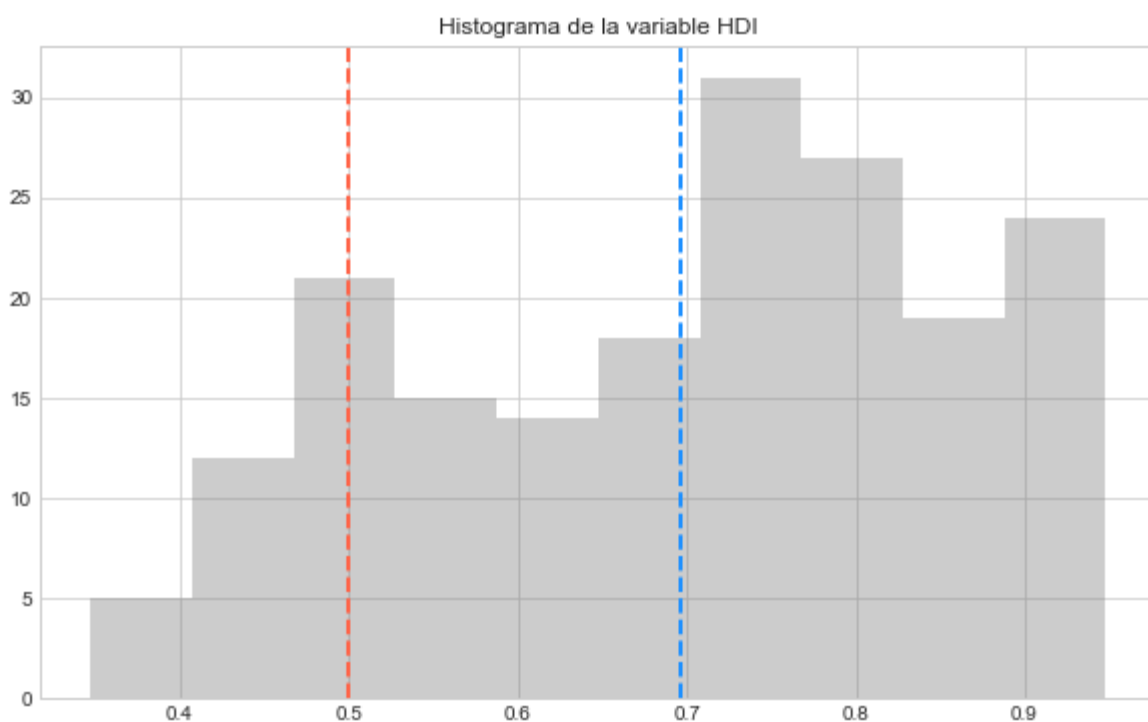


Imagen 3. Histograma de la variable HDI.  
Fuente: Desafío Latam.

Nuestro argumento se mantiene sólido. Con base en la evidencia generada, existen dos grandes grupos de países: Los desarrollados y los no desarrollados.

Cabe destacar que entre .50 y .70 encontramos a los países en vías de desarrollo.



### Preguntas de proceso

Tomémonos un momento para reflexionar sobre lo revisado:

- ¿Qué he aprendido hasta ahora?
- ¿Hay algo que me está dificultando? Si es así, retrocede un poco y revísalo nuevamente antes de seguir avanzando

## La distribución normal: Caracterizando nuestras variables

Resulta que el índice de desarrollo humano (y toda medición empírica), tiende a responder a ciertas leyes respecto a su comportamiento. Estos se resumen en *variables aleatorias*, que definen los posibles valores asociados a una distribución probabilística.

Para nuestro ejemplo, el índice de desarrollo humano tiene una distribución aproximadamente normal.

### ¿Y cómo comprobamos la normalidad de una variable?

Podemos inspeccionar qué tanto se acopla el histograma a la distribución normal al incluir la media y varianza de `hdi_dropna`.

La distribución normal (o Gaussiana) es una de las distribuciones más utilizadas para modelar eventos continuos. Fenómenos como la altura y peso de una población, así como los puntajes PSU o las notas de una clase se comportan de manera *aproximadamente normal*. Para el caso de una sola variable, depende de dos parámetros: la media (representada como  $\mu$ ) y varianza (representada como  $\sigma^2$ ).

- El parámetro media gobierna el punto donde se concentra la mayoría de los datos. Ésta sigue la definición de las medidas de tendencia central.
- El parámetro varianza representa qué tan concentrados están los datos **en torno a la media**.

```
print("La media es: ", round(hdi_dropna.mean(), 3))  
print("La varianza es: ", round(np.var(hdi_dropna), 3))
```

```
La media es:  0.696  
La varianza es:  0.024
```

Si la media es de .696 y la varianza es de 0.024, podemos analizar si la distribución empírica del índice de desarrollo humano sigue la distribución normal  $X_i \sim N(0.696, 0.024)$ .

Para superponer la curva gaussiana a nuestro histograma, utilizamos las siguientes líneas de código.

```
# Con plt.subplot vamos a dividir el espacio del en dos partes
plt.subplot(2,1,1)

# Graficamos el mismo histograma, especificando Normed para que el
# histograma representa densidades y no frecuencias
plt.hist(hdi_dropna, color='grey', alpha=.4, normed=True)

# extraemos los límites del histograma
x_min, x_max = plt.xlim()

# utilizando los límites del histograma para crear un array
x_axis = np.linspace(x_min, x_max, 100)

# graficamos la curva de densidad empirica (permite comparar
# directamente con la curva de densidad teorica)
gauss_kde = stats.gaussian_kde(hdi_dropna) # Kernel gaussiano
Z = np.reshape(gauss_kde(x_axis).T, x_axis.shape)
plt.plot(x_axis, Z, color='tomato', lw=3)

# agregamos la línea vertical para identificar la media
plt.axvline(hdi_dropna.mean(), color='dodgerblue', linestyle='--', lw=3)
plt.title("Histograma del IDH")

# El segundo gráfico
plt.subplot(2, 1, 2)

# Simulamos cómo debería comportarse un histograma normal.
plt.hist(np.random.normal(0.696, np.sqrt(0.024), 1000), color='grey',
alpha=.4, normed=True)

# Fijamos los límites del eje x
plt.xlim(0.3, 1)

# Generamos una lista entre
x_axis = np.linspace(.3, 1, 1000)
plt.plot(x_axis, stats.norm.pdf(x_axis,
                                0.696, np.sqrt(0.024)),
         color='tomato', lw=3)
plt.axvline(0.696, color='dodgerblue', linestyle='--', lw=3)
plt.title("Distribución simulada del IDH")
plt.subplots_adjust(hspace=0.6)
```

```
/Users/veterok/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6521: MatplotlibDeprecationWarning:  
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed  
in 3.1. Use 'density' instead.  
    alternative="'density'", removal="3.1")
```

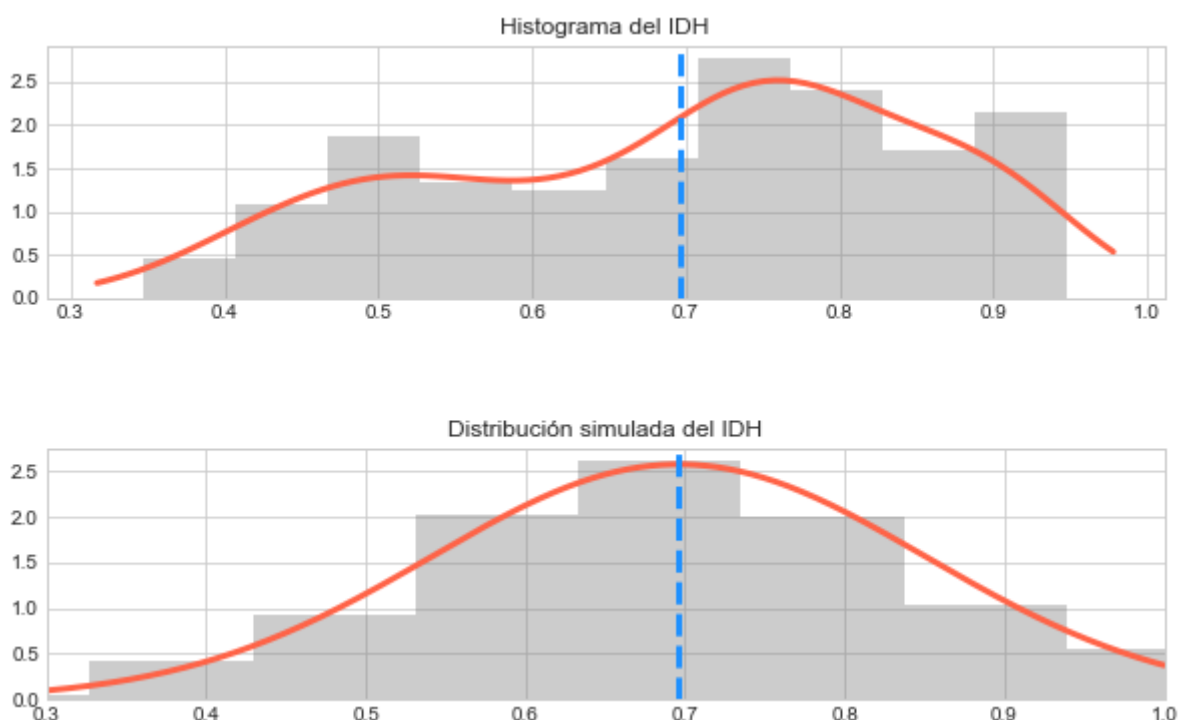


Imagen 4. Histograma y Distribución simulada.  
Fuente: Desafío Latam.

La figura de arriba presenta el histograma del índice de desarrollo humano, y la figura de abajo representa la simulación de una variable distribuida normal con la media y varianza de la variable. La recta azul indica la media de .696, y la curva roja representa la distribución normal.

Mientras que en el gráfico superior los puntajes no se acoplan de forma exacta a la curva y la media, la figura de la distribución simulada presenta el comportamiento *esperado*.

Cabe destacar que el comportamiento descrito en el histograma de la variable no descarta la utilización de la distribución normal como una ley para resumir el comportamiento.



## ¿Qué podemos esperar de los parámetros de una distribución normal?

Ya teniendo una primera aproximación a la distribución normal, generamos varias situaciones para comparar su comportamiento. En la figura creada con `gfx.normal_distributions()` se presentan tres casos donde, tanto media como varianza, cambian de valor:

- La distribución normal estandarizada, que representa el caso donde la media es 0 y la varianza es de 1. Se utiliza como benchmark para realizar comparaciones.
- Una distribución donde la media es de 0 y su varianza es de .5.
- Una distribución donde la media es de -1 y su varianza es 1.

Consideremos el caso de comparación. Dado que la media es 0, la cúspide de la distribución se concentrará en torno a este valor. Comparado a la curva con menor varianza, que si bien presenta la cúspide en 0, tiende a presentar una mayor densidad dado que la menor varianza conlleva a concentrar una mayor cantidad de información alrededor de la media. Si bien la curva verde presenta forma idéntica a la normal estandarizada, su cúspide se centra en -1 a diferencia de la normal estandarizada.

En el lado derecho se visualizan las densidades acumuladas de las simulaciones. La función acumulada expresa el porcentaje de datos *hasta* cierto punto del rango. Observamos que cuando  $x=0$ , la función acumulada de la distribución normal estandarizada es cercana al 50%, mientras que para la distribución con menor media es cercana al 90%. De similar manera, el 50% de la información en la simulación con una menor media se alcanza en -1.

```
gfx.normal_distributions()
```

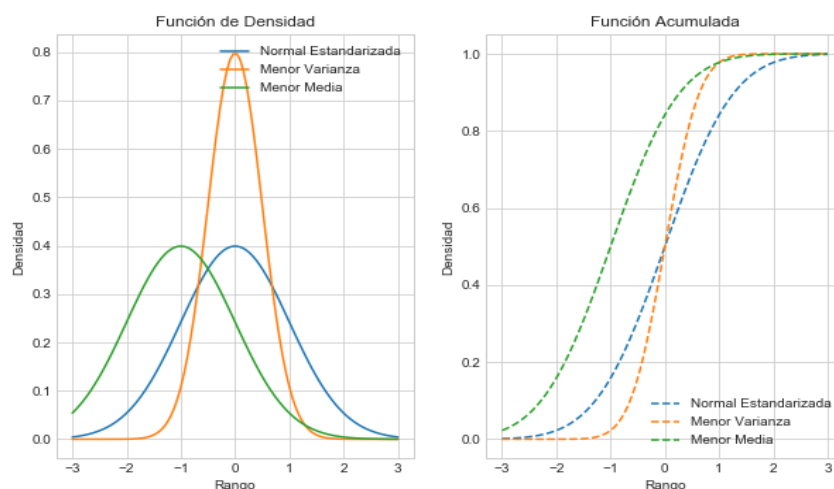


Imagen 5. Función de Densidad y Función Acumulada.

Fuente: Desafío Latam.

## Una variable no-normal y una posible transformación

Ahora generamos el histograma del Producto Interno Bruto (PIB) per cápita de los países. Para graficarlo, procedemos de manera similar a cómo lo hicimos con el Índice de Desarrollo Humano (IDH).

Primero excluimos todas las observaciones perdidas y guardemosla en una variable nueva.

La posible transformación candidata es el logaritmo:

```
# Generamos la normalización del gdp mediante el logaritmo.
gdp_dropna = df['gle_cgdpc'].dropna()
log_gdp = np.log(gdp_dropna)
print(round(log_gdp.mean(), 2))
```

8.85

Posteriormente grafiquemos el histograma y su curva.

```
plt.subplot(1, 2, 1)
plt.hist(gdp_dropna, color='grey', alpha=.4, normed=True)
x_min, x_max = plt.xlim()
x_axis = np.linspace(x_min, x_max, 100)
plt.plot(x_axis, stats.norm.pdf(x_axis,
                                gdp_dropna.mean(),
                                gdp_dropna.std()),
         color='tomato', lw=3)
plt.axvline(gdp_dropna.mean(),
            color='dodgerblue',
            linestyle='--', lw=3)
plt.title("Histograma PIB")

plt.subplot(1, 2, 2)
plt.hist(log_gdp, color='grey', alpha=.4, normed=True)
x_min, x_max = plt.xlim()
x_axis = np.linspace(x_min, x_max, 100)
plt.plot(x_axis, stats.norm.pdf(x_axis,
                                log_gdp.mean(),
                                log_gdp.std()),
         color='tomato', lw=3)
```

```
plt.axvline(log_gdp.mean(),  
            color='dodgerblue',  
            linestyle='--', lw=3)  
plt.title("Histograma log(PIB)");
```

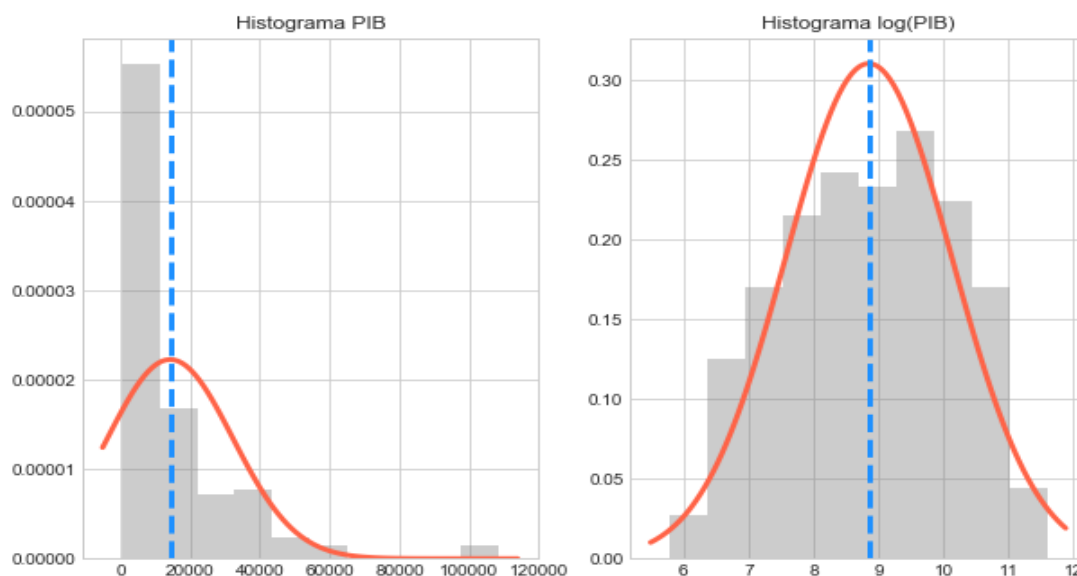


Imagen 6. Histogramas de PIB y log(PIB)  
Fuente: Desafío Latam.

En el gráfico de la izquierda se presenta el histograma del PIB sin transformar, y observamos que la distribución presenta un fuerte sesgo en valores bajos. Hay pocos países que tienen ingresos promedio de USD 100.000 o más que escapan de la media, mientras que hay una amplia mayoría de observaciones con ingresos bajo la media. Esta distribución altamente sesgada, donde algunos valores se escapan de forma substancial de la muestra, se puede corregir mediante la distribución *lognormal*.

Podemos aplicar un logaritmo al vector `gdp` para reescalar sus valores. En la figura de la derecha se presenta el histograma del logaritmo del PIB.

El principal contrapunto de ocupar logaritmos para transformar variables, es el hecho que sus resultados son difíciles de implementar. Por lo general sirven para transformar data simétrica en una distribución más ajustada a lo normal o para mejorar la linealidad entre variable dependiente e independiente en un modelo de regresión. Sin embargo, se debe implementar con precaución, especialmente al momento de interpretar los resultados.

Efectivamente, al reescalar la variable, los nuevos puntajes generan un histograma aproximadamente normal.

## Una variante importante: La distribución normal estandarizada

Cuando la distribución normal tiene una media de 0 y una varianza de 1, se conoce como la *distribución normal estandarizada*. Dado que su media representa cero, permite reflejar diversos eventos tales como la ausencia de efecto de un estimador (distribución conocida como *distribución bajo la hipótesis nula*), o también cuando buscamos centrar observaciones de una variable respecto a la media (procedimiento conocido como puntajes *z*).

Los ejemplos anteriores sugieren la omnipresencia de la distribución normal. Dado su afabilidad para ser interpretada, cabe resaltar sus características más importantes:

- Es **unimodal**, lo que implica que tiene sólo un punto que es el más alto, donde se concentran la mayoría de los datos.
- Es **simétrica** dado que la mayoría de los datos gravitan alrededor de la media.
- Es **positiva**, por tanto todos los valores (positivos o negativos) tienen una probabilidad  $Pr \geq 0$  de suceder.

Mediante los componentes paramétricos de la distribución normal podemos calcular el porcentaje de datos que estará bajo la curva. Esto se visualiza abajo con `gfx.normal_distribution_sigma()`:

- El 68% de los datos se encontrarán entre  $\pm 1\sigma \rightsquigarrow Pr(\mu - \sigma \leq X \leq \mu + \sigma)$ . Esto corresponde al área de azul intenso.
- El 95% de los datos se encontrarán entre  $\pm 2\sigma \rightsquigarrow Pr(\mu - 2\sigma \leq X \leq \mu + 2\sigma)$ .
- El 99% de los datos se encontrarán entre  $\pm 3\sigma \rightsquigarrow Pr(\mu - 3\sigma \leq X \leq \mu + 3\sigma)$ . Esto corresponde a las colas de la distribución normal, con un azul pálido.

```
gfx.normal_distribution_sigma()
```

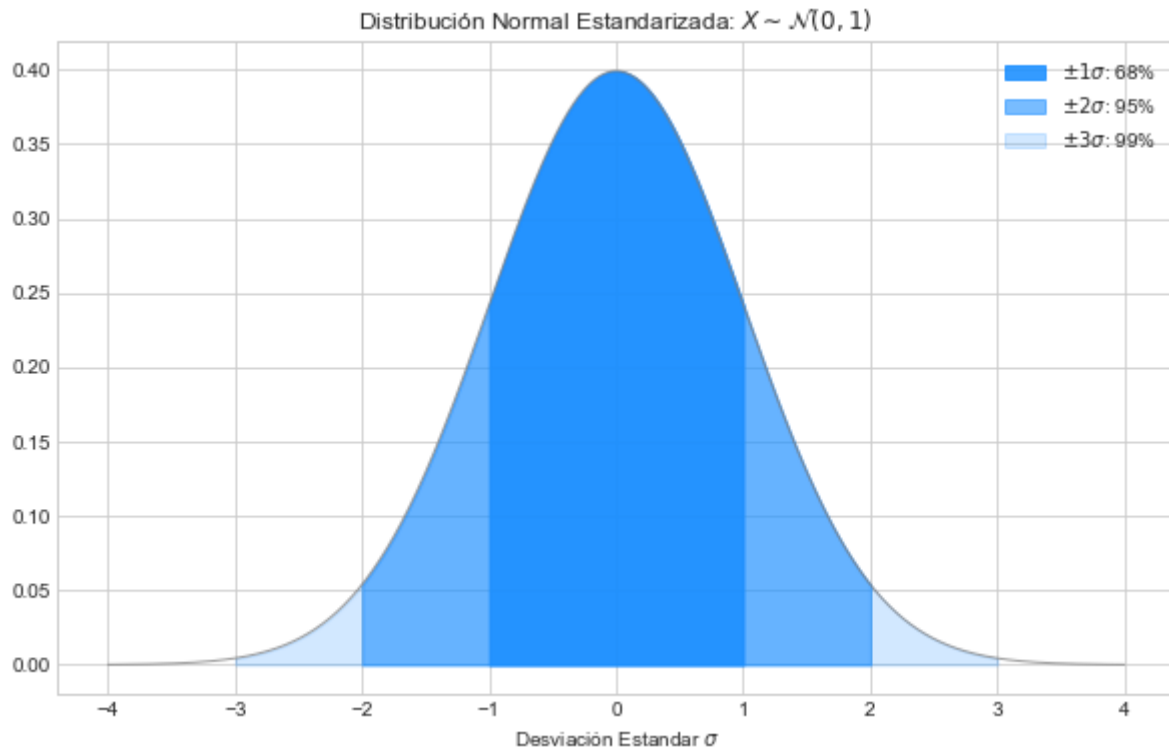


Imagen 7. Distribución Normal Estandarizada.  
Fuente: Desafío Latam.

## Puntajes Z

La información entregada por la distribución normal nos permite aproximar nuestro conocimiento respecto al comportamiento de la variable.

También podemos utilizar esta información para analizar cómo una observación en específico se comporta respecto a la media. Para ello utilizamos los *puntajes z*.

La fórmula se detalla a continuación.

$$Puntaje\ Z = \frac{x_i - \bar{x}}{\sigma}$$

El procedimiento para obtener el puntaje z de una observación implica:

1. Obtener la media del vector donde proviene la observación.
2. Restar la observación con la media.
3. Dividir el resultado por la desviación estándar.

El resultado de la operación indica a cuántas desviaciones estándares se encuentra una observación respecto a la media.

## ¿Cuál es el IDH de Chile?

Para ello debemos acceder a la celda específica.

Comencemos por separar la fila de Chile. Esto lo logramos mediante:

```
chile = df[df['cname'] == 'Chile']
```

Instruimos al `DataFrame` para seleccionar sólo la fila que satisfaga la condición `df['cname'] == 'Chile'`. Posteriormente podemos acceder a la columna específica:

```
chile['undp_hdi']
```

```
35    0.845  
Name: undp_hdi, dtype: float64
```

Resulta más conveniente concatenar todas estas acciones de manera tal de escribir código más sucinto (a expensas de hacerlo un poco más inentendible).

```
hdi_chile = df[df['cname']=='Chile']['undp_hdi']  
print('hdi Chile: %.3f' %hdi_chile)  
print('hdi medio a nivel mundial: %.3f' %round(hdi_dropna.mean(),3))
```

```
hdi Chile: 0.845  
hdi medio a nivel mundial: 0.696
```

Una forma rudimentaria de calcular el puntaje-z de Chile es la siguiente, donde aplicamos los componentes detallados en la fórmula.

Con un valor final de 0.96, observamos que Chile se posiciona casi en una desviación estándar respecto al promedio mundial del índice de desarrollo humano.

```
print('Puntaje Z para Chile: %.2f' %((hdi_chile - hdi_dropna.mean()) /  
np.sqrt(hdi_dropna.var())))
```

Puntaje Z para Chile: 0.96

Ya sabemos cómo desarrollar funciones, y podemos generalizar todo este procedimiento para calcular el puntaje z de cualquier país respecto a cualquier variable.

Para este ejemplo, vamos a seguir analizando el índice de desarrollo humano.

```
def get_zscore(col, case):  
    get_point = df[df['cname']==case][col]  
    preproc=df[col].dropna()  
    get_mean = preproc.mean()  
    get_std = np.sqrt(preproc.var())  
    return (get_point - get_mean) / get_std
```



Como ya sabemos, la prueba de fuego de una función es que replique el mismo resultado que la versión rudimentaria. Observamos que nuestra función `get_zscore` devuelve el mismo resultado.

```
get_zscore('undp_hdi', 'Chile')
```

```
35    0.960783  
Name: undp_hdi, dtype: float64
```

Ahora calculemos los puntajes “z” de países como Namibia, Uruguay y China.

```
get_zscore('undp_hdi', 'Namibia')
```

```
118   -0.378656  
Name: undp_hdi, dtype: float64
```

La evidencia sugiere que el nivel de desarrollo humano de Namibia, comparado al promedio mundial, es menor, situándose -.38 desviaciones estándar por debajo de la media.

```
get_zscore('undp_hdi', 'Uruguay')
```

```
188    0.632363  
Name: undp_hdi, dtype: float64
```

Para el caso de Uruguay, siguiendo las mediciones del IDH, observamos que se sitúa sobre la media mundial, pero relativamente más abajo de Chile.

```
get_zscore('undp_hdi', 'China')
```

```
36      0.245986  
Name: undp_hdi, dtype: float64
```

Independiente de su poder económico, el IDH para China es sustancialmente bajo, situándose .2 desviaciones estándares sobre la media.

Cabe destacar que el criterio de comparación desarrollado con los puntajes z asume que nuestra distribución es normal estandarizada ( $X \sim N(0, 1)$ ).

Para visualizar de mejor manera la posición de las observaciones, vamos a refactorizar la función `get_zscore` para que haga lo siguiente:

1. Calcule un puntaje z asociado al país y a la medición.
2. Grafique su posición en referencia a una distribución normal estandarizada.

```
# definamos la función  
def point_zscore(col, case):  
    get_point = df[df['cname']==case][col]  
    preproc=df[col].dropna()  
    get_mean = preproc.mean()  
    get_std = np.sqrt(hdi_dropna.var())  
    get_zscore = float((get_point - get_mean) / get_std)  
    plt.axvline(get_zscore, ymin = 0, ymax = .1, color='tomato', lw=2)  
    plt.annotate(case, xy=(get_zscore + .05, 0.06),  
                 color='tomato', rotation=60, ha='center', size = 7)
```



Ahora graficamos una curva gaussiana que siga  $X \sim N(0,1)$ . También demarcaremos la media en 0.

```
# generamos un array lineal
x_axis = np.linspace(-3, 3, 200)
# simulamos la función de densidad
plt.plot(x_axis, stats.norm.pdf(x_axis, 0, 1))
# marcamos la media en 0
plt.axvline(0, color='dodgerblue', linestyle='--')

# Pasamos un array de casos a visualizar en un loop.
# donde para cada elemento de éste array
for i in ['Chile', 'Burkina Faso', 'Haiti', 'Myanmar',
          'Germany', 'Norway', 'Finland', 'Yemen', 'Bolivia',
          'Algeria', 'Namibia', 'Liberia']:
    # aplicaremos la función point_zscore
    point_zscore('undp_hdi', i)
```

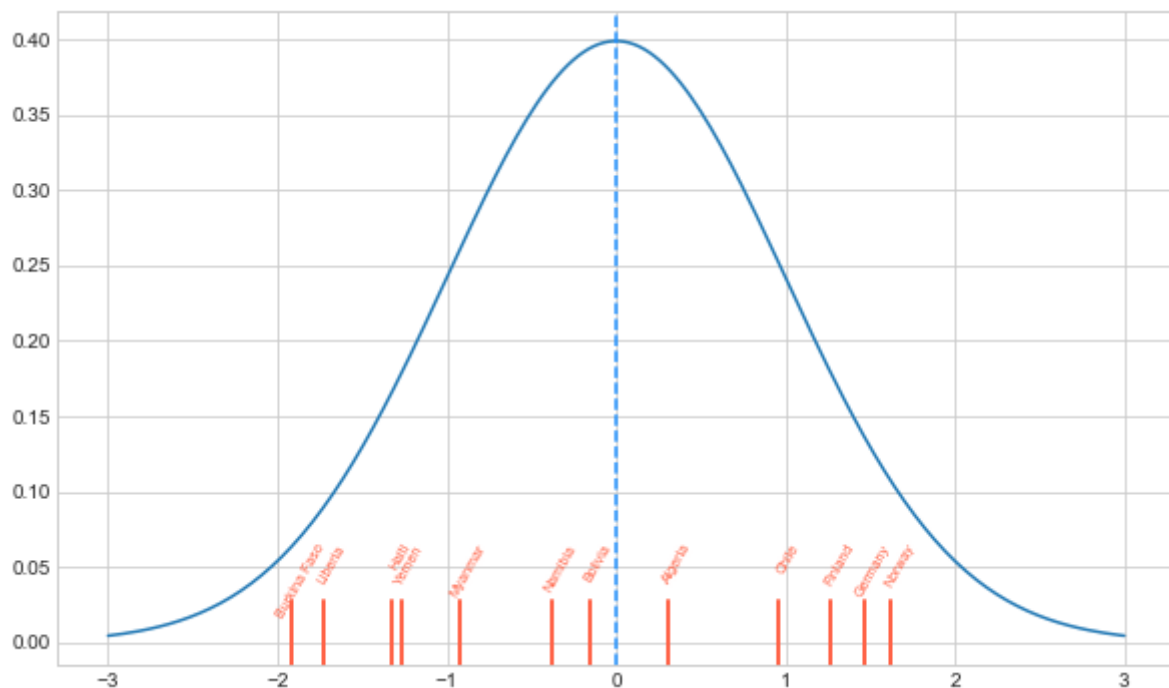


Imagen 8. Curva Gaussiana.  
Fuente: Desafío Latam.

Posterior a graficar nuestra distribución normal estandarizada, pasaremos una lista donde posicionamos una serie de países en cuanto a sus desviaciones estándares. Intentamos agregar una cantidad variada de países, para ver los correlatos desarrollados en nuestro análisis. En el rango inferior de nuestra distribución encontramos países de la zona subsahariana de África, como Burkina Faso, Liberia y Namibia, países caracterizados por su fragilidad institucional. En el extremo opuesto, encontramos a países asociados al Estado de Bienestar escandinavo como Noruega, Finlandia e incluso, Alemania.

## Variables Discretas: ¿Qué pasa cuando nuestras variables miden atributos?

Hasta ahora sabemos trabajar cuando las variables representan mediciones continuas. Resulta que muchas veces nuestros objetivos a analizar presentan *categorías* y sus números son representaciones conceptuales.

En aquellas situaciones, nos debemos valer de las variables aleatorias discretas. las cuales permiten resumir el comportamiento de un fenómeno mediante el conteo (a diferencia de las variables aleatorias continuas que permiten resumir el comportamiento de un fenómeno mediante la medición).

La distribución de una variable aleatoria  $X$  generalmente se especifica mediante un listado de todos los posibles valores y una función de probabilidad de masa  $p(x)$ .

$$p(x) = px(x) = Pr[X = x]$$

## Ensayo de Bernoulli

Tomemos la siguiente situación: Como investigadores nos preguntan por la estabilidad democrática a nivel mundial. Para esto emplearemos la variable `gol_inst`, que mide el tipo de régimen en cada una de las observaciones.

Nuestro objetivo es desarrollar una variable binaria, donde asignaremos la etiqueta 'Democracia' para todos aquellos casos que presenten alguna variante de democracia (régimen parlamentario, presidencial o semipresidencial), y 'No Democracia' para todos los casos que no presenten dicha gobernanza con participación y división de los poderes del Estado.

`numpy` ofrece el operador ternario `np.where()`, el cual nos facilita la reconversión de valores en base a la satisfacción de un criterio.

La sintaxis de `np.where()` consta de tres partes:

```
np.where(<condición>, <valor_positivo>, <valor_negativo>)
```

- `<condición>` hace referencia a un criterio formalizado en alguna expresión booleana que evalúe en verdaderos o falsos.
- `<valor_positivo>` es el valor que tomarán aquellos elementos que satisfagan la condición.
- `<valor_negativo>` es el valor que tomarán aquellos elementos que no satisfagan la condición.

Sabemos que en la variable las democracias tienen un número asignado entre el 0 y el 2, y que las dictaduras tienen números entre el 3 y el 5.

Para utilizar `np.where`, desarrollamos la condición booleana `df['gol_inst'] <= 2` para evaluar como 'Democracia' todos aquellos casos menores o iguales a 2 y 'No democracia' los demás.

```
# Proporciones de cada tipo de gobierno en el dataset
df['democracies'] = np.where(df['gol_inst'] <= 2, 'Democracia', 'No
Democracia')
# pidamos los porcentajes de democracias
democount = df['democracies'].value_counts('%').round(2)
democount
```

```
Democracia      0.65  
No Democracia   0.35  
Name: democracies, dtype: float64
```

```
# Conteo de tipos de gobierno en el dataset  
df['democracies'].value_counts()
```

```
Democracia      127  
No Democracia    67  
Name: democracies, dtype: int64
```

Acorde a nuestro criterio de recodificación, 127 observaciones se clasificaron como regímenes con tendencias democráticas y 67 como regímenes con tendencias autoritarias. El objeto `democount` que creamos nos puede servir para generar un gráfico de barras.

```
democount.plot(kind = 'bar')  
plt.title("Democracias vs Dictaduras en el mundo")  
plt.xticks(rotation = 'horizontal');
```

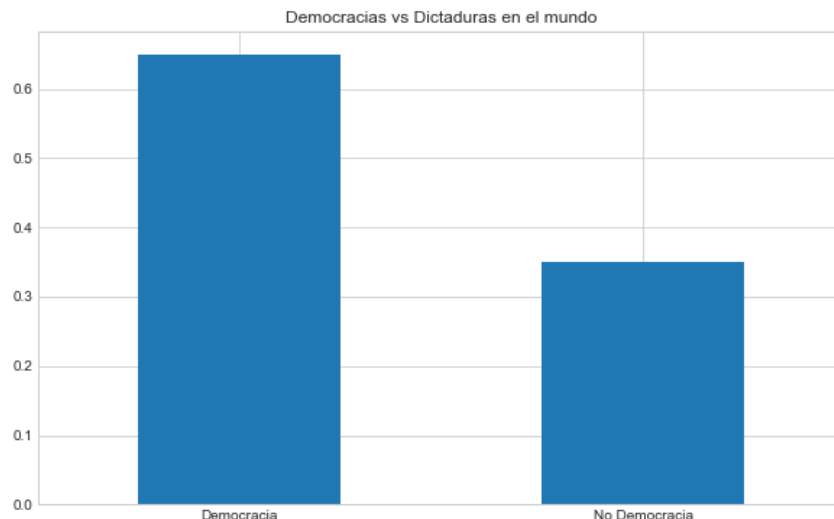


Imagen 9. Democracias vs Dictaduras en el mundo.  
Fuente: Desafío Latam.

Observamos que el 65.4% de las observaciones son clasificadas como democracias. Si tomamos un país al azar de los registrados en el dataset, hay un 65% de probabilidad de que el país sea una variante del régimen democrático.

Para sustentar este enunciado, es necesario respaldarnos en las leyes estadísticas. **¿Cómo podemos hacerlo?**

Acorde a esta variable, un país sólo puede ser *democracia o dictadura*.

Sabemos que la probabilidad de suceso de un evento es  $P$ . Por tanto, su tasa de ausencia está definida como  $Q = (1 - P)$ .

La distribución Bernoulli (también llamada ensayo de Bernoulli) representa un caso individual donde se realiza un ensayo con  $x \in [0, 1]$ . Está gobernada por un parámetro continuo  $\mu \in [0, 1]$  que representa la probabilidad de  $x = 1$ .

Podemos solicitar los primeros momentos de un ensayo de Bernoulli en el módulo `scipy.stats` mediante la función `bernoulli.stats`, donde ingresando nuestra probabilidad obtenemos nuestra media y varianza.

```
mu, sigma = stats.bernoulli.stats(.65)
print("La media del ensayo de Bernoulli es de: ", mu.round(2))
print("La varianza del ensayo de Bernoulli es de: ", sigma.round(2))
print("La probabilidad de ocurrencia asumiendo una desviación estandar  
se encuentra entre :",
      round(mu - sigma, 2), round(mu + sigma, 2))
```

```
La media del ensayo de Bernoulli es de: 0.65
La varianza del ensayo de Bernoulli es de: 0.23
La probabilidad de ocurrencia asumiendo una desviación estandar se  
encuentra entre : 0.42 0.88
```

## Distribución Binomial: Analizando una serie de ensayos de Bernoulli

La información generada por el ensayo de Bernoulli es útil cuando buscamos caracterizar un caso cualquiera en nuestra muestra. Resulta que es de más interés ver cómo se distribuiría la tasa de éxito o fracaso en una muestra finita. Para ello utilizamos la distribución binomial.

La distribución binomial da la probabilidad de observar  $m$  ocurrencias de  $x = 1$  en un conjunto de  $N$  muestras que surgen de una distribución Bernoulli, donde la probabilidad de ocurrencia de  $x = 1$  es  $\mu \in [0, 1]$ .

Solicitemos los primeros momentos de la distribución binomial para la muestra de 194 países, con la probabilidad de .654 de ser democráticos. Mediante la función `binom` podemos generar un objeto que tendrá el método `stats` que devuelve la cantidad de países que deberían ser democracias (`mu`) y la varianza (`sigma`)

```
prob_dem = stats.binom(len(df), .654)
mu, sigma = prob_dem.stats()
print("Con", len(df), "países y una probabilidad de .654, esperamos  
que", mu, " países tengan regímenes democráticos." )
```

Con 194 países y una probabilidad de .654, esperamos que 126.876 países tengan regímenes democráticos.

La distribución Binomial depende de dos parámetros: el tamaño muestral y la proporción estimada. Para entender mejor cómo la proporción estimada afecta a la distribución, consideremos el siguiente ejemplo.

Vamos a graficar las distribuciones empíricas de cuatro niveles de probabilidad (.3, .6, .9, .95), manteniendo constante el tamaño muestral.

Observamos que en la medida que la proporción de casos va aumentando, nuestra curva de densidad se desplaza a lo largo del eje x, indicando mayores tasas de casos presentes en la muestra. Otro elemento a destacar es que, a medida que aumentamos la probabilidad, la curva tiende a disminuir su varianza.

```
# generamos una lista con cuatro colores
colors = ["tomato", "darkgoldenrod", "limegreen", "dodgerblue"]

# iniciamos un loop para índices y valores
for i, prob in enumerate([.3, .6, .9, .95]):
    # generamos un arreglo con numpy de 175
    x_axis= np.arange(175)
    # para cada valor de nuestro array graficamos la probabilidad
    plt.vlines(x_axis, 0, stats.binom(150, prob).pmf(x_axis),
              # y asignamos un color en específico
              colors=colors[i], label="n=150, p={}".format(prob))
plt.legend(loc=2, fontsize = 15)
plt.title("Densidad de la distribución Binomial en distintos p", size = 20)
plt.xlabel('Cantidad de casos donde p está presente', size = 15)
plt.ylabel('Densidad', size = 15);
```

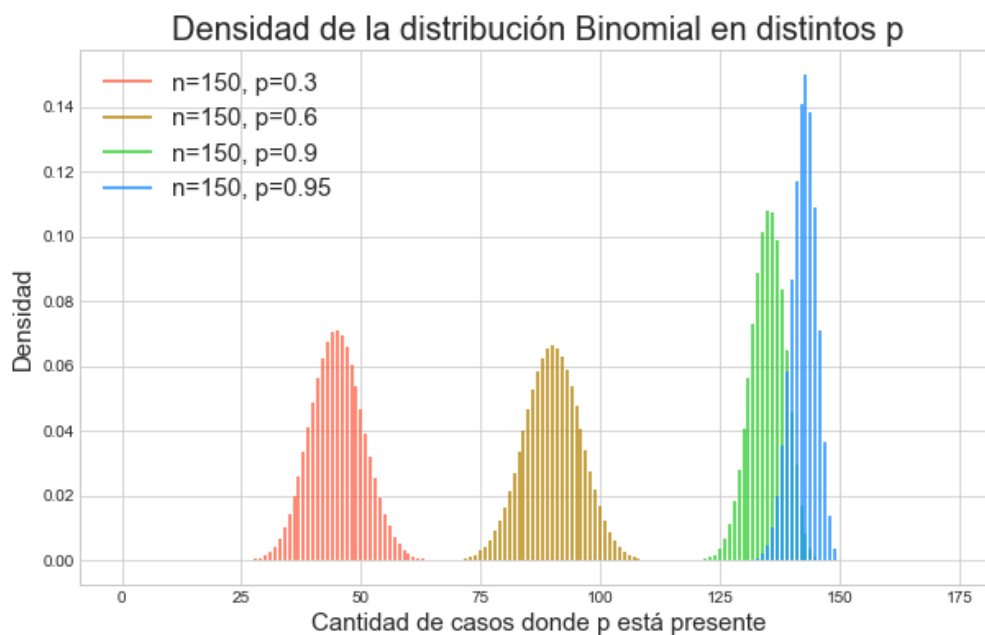


Imagen 10. Densidad de la distribución Binomial en distintos p.  
Fuente: Desafío Latam.

## Sobre `enumerate`.

- Hay casos dónde estamos interesados en pasar por todos los elementos de un arreglo, pero deseamos pasar tanto su posición y elemento como parámetros.
- `enumerate` permite contar la posición e identificar cada elemento de una lista en específico. Así podemos pasar dos argumentos cuando ejecutamos el loop, separados por coma.
- En el código del gráfico establecemos que vamos a recorrer todos los valores `i` (que corresponden a la posición del elemento en la lista) y `prob` (que corresponde al valor del elemento en la lista).
- `i` se pasa como un localizador de cada elemento en `colors[i]`, mientras que `prob` se pasa como argumento en `stats.binom(150, prob)`.



## ¿Qué pasa cuando mantenemos nuestra probabilidad fija condicionada al tamaño muestral?

Grafiquemos el caso donde mantenemos nuestra probabilidad en .65, y vamos aumentando nuestra muestra de 50 hasta 200. Mientras que en valores muestrales pequeños la curva de densidad se muestra con una menor varianza, la curva se aproxima a la campana gaussiana en la medida que aumentamos su tamaño.

```
# generamos una lista de colores
colors = ["tomato", "darkgoldenrod", "limegreen", "dodgerblue"]

# iniciamos un loop para índices y valores
for i, n_size in enumerate([50, 100, 150, 200]):
    # generamos un arreglo con numpy de 200
    x_axis= np.arange(200)
    # para cada valor de nuestro array graficamos la probabilidad
    # condicional al tamaño
    plt.vlines(x_axis, 0, stats.binom(n_size, .65).pmf(x_axis),
              colors=colors[i], label=r'n:{0}, p=.65'.format(n_size))
plt.legend()
plt.title("Densidad de la distribución Binomial en distintos n")
plt.xlabel("Cantidad de casos donde p está presente")
plt.ylabel('Densidad');
```

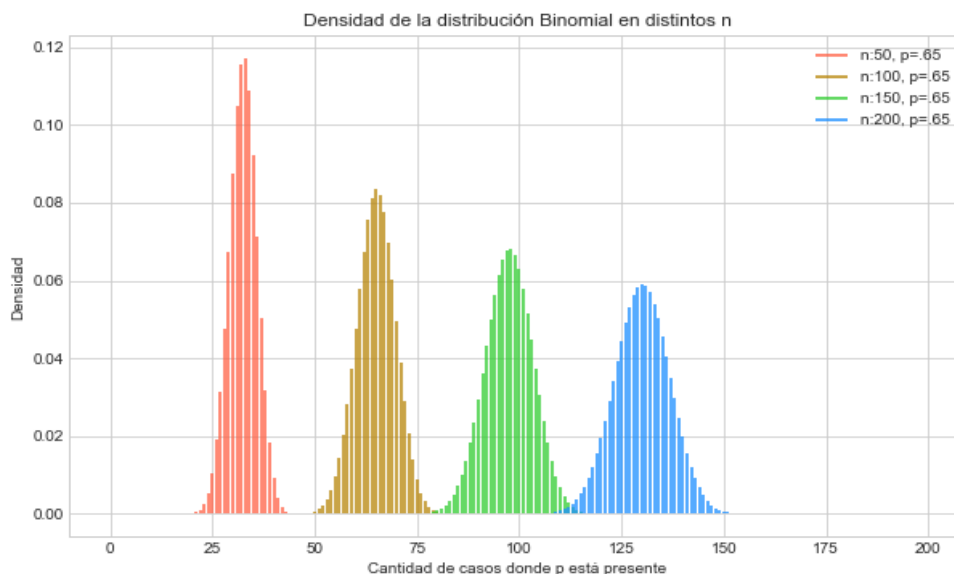


Imagen 11. Densidad de la distribución Binomial en distintos  $n$ .

Fuente: Desafío Latam.

## Reflexiona

- ¿Cuáles son los métodos que tienen los DataFrames para especificar el tipo de loop a ejecutar? E qué se diferencian
- ¿Qué es un histograma y por qué se relaciona con el comportamiento de una variable?
- ¿Cómo se comprueba que una variable es normal o no-normal?
- ¿Qué es el puntaje Z? ¿Cuál es el procedimiento para obtener el puntaje z de una observación?
- ¿Para qué utilizamos el Ensayo de Bernoulli? ¿En qué se relaciona con la distribución binomial?



## Referencias

- Para profundizar sobre las variables aleatorias y sus atributos, pueden referirse a:
  - Caffo, B. 2015. Statistical Inference for Data Science:
    - Ch. 6: Some common distributions.
    - Ch. 7: Asymptopia
  - Dodge, Y. 2006. The Concise Encyclopedia of Statistic:
    - Random Variable (página 446)
- Si desean tener una exposición rigurosa a los aspectos formales y asintóticos de las variables aleatorias, pueden referirse a:
  - Casella, G; Berger, R. 2002. Statistical Inference:
    - Ch 3: Common families of distributions