NumPy (Numerical Python)	Indices, Subsets, Metodos de Arrays	Operaciones estadísticas y matemáticas	Funciones de conjuntos	Estadística
Crear arrays	<pre>Indices de arrays array[i] devuelve la indice i; las indices de los</pre>	Operaciones estadísticas y matemáticas El parametro axis en arrays bidimensionales:	<pre>np.unique(array) devuelve un array con los valores únicos del array ordenados np.unique(array, return_index=True) devuelve un array con</pre>	Tablas de frecuencias Frecuencias absolutas
Crear arrays de listas array = np.array(lista, dtype= tipo) crea un array	<pre>arrays unidimensionales funcionan igual que las listas array[i, j] o array[i][j] devuelve el elemento de la columna j de la fila i</pre>	<pre>axis = 0 columnas axis = 1 filas</pre>	los valores únicos del array ordenados y un array con la posición de la primera instancia de cada valor	el número de veces que se repite un número en un conjunto de datos df = df.groupby('columna').count().reset index()
<pre>unidimensional de una lista array = np.array([lista1, lista2]) bidimensional de dos listas</pre>	array[:,:n] seleccionar todas las filas y las columnas hasta n-1	 si especificamos el axis, la operación devuelve el resultado por cada fila o columna. Por ejemplo: 	<pre>np.unique(array, return_inverse=True) devuelve un array con los valores únicos del array ordenados y un array con las posiciones de cada elemento de cada valor</pre>	Frecuencias relativas las veces que se repite un número o categoría en un
<pre>array = np.array([listadelistas1, listadelistas2]) un array bidimensional de dos listas</pre>	<pre>array[h, i, j] o array[h][i][j] devuelve el elemento de la columna j de la fila i del array h array[h][i][j] = n cambiar el valor del elemento en</pre>	<pre>np.sum(array, axis = 0) suma de cada fila</pre>	<pre>np.unique(array, return_counts=True) devuelve un array con los valores únicos del array ordenados y un array con el número de veces que aparece cada valor</pre>	<pre>conjunto de datos respecto al total, en porcentajes df_group_sin_str = df_group.drop('columna_str',</pre>
<pre>Crear otros tipos de arrays array = np.arange(valor_inicio, valor_final, saltos) un array usando el formato [start:stop:step]</pre>	esta posicion al valor n	El parametro axis en arrays multidimensionales: axis = 0 dimensión axis = 1 columnas	np.unique(array, axis = b) devuelve un array con los valores únicos ordenados de las filas o columnas	<pre>axis=1) frecuencia_relativa = df_group_sin_str / df.shape[0] * 100</pre>
<pre>array = np.ones(z,y,x) crea un array de todo unos de la forma especificada</pre>	Subsets array > n devuelve la forma del array con True o False	<pre>axis = 2 filas - si especificamos el axis, la operación devuelve el resultado por cada dimensión, fila o columna.</pre>	Funciones para arrays unidimensionales np.intersect1d(array1, array2) devuelve un array con los	<pre>columnas = df_group_sin_strings.columns df_group[columnas] = frecuencia_relativa</pre>
<pre>array2 = np.ones_like(array1) de la forma basada en otra array array = np.zeros(z,y,x)</pre>	según si el elemento cumple con la condición o no array[array > n] devuelve un subset: todos los valores que cumplen la condición en una lista dentro de un	Por ejemplo: np.sum(array_3D, axis = 0) devuelve un array de una matriz con la suma de todas las matrices	valores únicos de los elementos en común de dos arrays np.intersect1d(array1, array2, return_indices=True) devuelve un array con los valores únicos de los elementos	Tablas de contingencia tabla de frecuencias que cuenta todas las
forma especificada array2 = np.zeros_like(array1) crea un array de todo zeros de la forma basada en otra array	array array[(array > n) & (array < m)] devuelve un subset: todos los valores que cumplen las condiciones en una	np.sum(array_3D, axis = 1) devuelve un array donde las filas contienen las sumas de las columnas de cada matriz	en común de dos arrays y arrays con los índices de cada valor, por array np.union1d(array1, array2) devuelve un array ordenado con	combinaciones posibles de cada pareja de valores de las columnas que estamos intentando comparar df_crosstab = pd.crosstab(df['columna1'],
<pre>array = np.empty((z,y,x), tipo) datos por defecto tipo float array2 = np.empty_like(array1) crea un array vacia con la</pre>	lista dentro de un array; se puede usar para "or"	Operaciones con parámetro del axis:	los elementos resultantes de unir dos arrays (valores únicos) np.in1d(array1, array2) devuelve un array con True o False	<pre>df['columna2'], normalize = True, margins = True) normalize muestra los valores en porcentajes (por uno) margins muestra los totales y subtotales</pre>
<pre>forma basada en otra array array = np.eye(z,y,x, k = n) diagonal empezando en la posicion k</pre>	Metodos de arrays nuevo_array = array.copy() crea un a copia del array	<pre>np.sum(array_3D) devuelve la suma de todos los elementos de los matrices np.mean(array) devuelve la media de todo el array</pre>	por cada elemento de array1 según si aparece el mismo valor en array2 np.setdiff1d(array1, array2) devuelve un array ordenado con	Coeficiente de correlación de Pearson
<pre>array = np.identity(x) crea una matriz de identidad con ceros en filas y unos en la diagonal, de forma cuadrada</pre>	<pre>np.transpose(array_bidimensional) cambia los filas del array a columnas y las columnas a filas np.transpose(array_multidimensional) cambia el número</pre>	<pre>np.std(array) devuelve la desviación estándar de todo np.var(array) devuelve la varianza de valores de</pre>	los valores únicos que están en array1 pero no en array2 np.setxor1d(array1, array2) devuelve un array ordenado con los valores únicos que NO están en común de los dos arrays	- nos permite conocer la intensidad y dirección de la relación entre las dos variables - coeficiente > 0: correlación positiva
NumPy Random	de columnas al número de arrays y viceversa; el número de filas no cambia np.transpose(array multidimensional, (z,y,x)) hace la	todo <pre>np.min(array) devuelve el valor mínimo del array np.max(array) devuelve el valor máximo del array</pre>	Estadística	- coeficiente < 0: correlación negativa - coeficiente = 1 o -1: correlación total - coeficiente = 0: no existe relación lineal
<pre>np.random.seed(x) generador de números aleatorios, para que las funciones</pre>	transposicion segun lo que especificemos usando las posiciones de la tupla (0,1,2) de la forma original array = np.arange(n).reshape((y,x)) crea un array	<pre>np.sum(array) devuelve la suma de los elementos del array np.cumsum(array) devuelve un array con la suma</pre>	Medidas de dispersión Desviación respecto a la media	<pre>df['columna1'].corr(df['columna2'] correlacion entre dos variables</pre>
random que van después siempre cogerán los mismos valores "aleatorios"	usando reshape para definir la forma array = np.reshape(array, (z,y,x)) crea un array con	acumulada de los elementos a lo largo del array np.cumprod(array) devuelve un array con la multiplicación acumulada de los elementos a lo largo	la diferencia en valor absoluto entre cada valor de los datos y su media aritmética	<pre>matriz_correlacion = df.corr() crea una matriz mostrando las correlaciones entre todos los variables sns.heatmap(df.corr()[['column1', 'column2']], cmap =</pre>
Crear arrays con valores aleatorios array = np.random.randint(inicio, final, forma_matriz) crea un array de números aleatorios entre dos valores;	los valores de otro array usando reshape para definir la forma array = np.swapaxes(array, posicion, posicion)	del array Operaciones sin parámetro del axis:	<pre>diferencias = df['columna'] - df['columna'].mean() desviación_media = np.abs(diferencias) Varianza</pre>	<pre>'color_palette', annot = True, vmin = -1, vmax = 1) crea una grafica heatmap de la matriz de correlaciones</pre>
forma_matriz: (z,y,x) z: número de arrays y: número de filas	intercambia dos ejes de una matriz usando las posiciones (z=0,y=1,x=2) de la forma original	<pre>np.sqrt(array) devuelve un array con la raíz cuadrada no negativa de cada elemento del array np.exp(array) devuelve un array con el exponencial</pre>	medida de dispersión; la variabilidad respecto a la media df['columna'].var()	Sesgos (skewness) medida de la asimetría de la distribución de los valores de una variable alrededor de su valor medio
x: número de columnas array = np.random.randint(inicio, final) número aleatorio en el rango	Otras operaciones np.sort(array) devuelve un array con los valores de	de cada elemento del array <pre>np.mod(array1, array2)</pre> devuelve un array con el resto de la división entre dos arrays	Desviación estándar o desviación típica la raíz cuadrada de la varianza; cuanto mayor sea, mayor será la dispersión o variabilidad en nuestros datos	- valor de sesgo positivo: sesgado a la derecha - valor de sesgo negativo: sesgado a la izquierda - valor de sesgo igual a 0: valores simetricos
<pre>array = np.random.rand(z,y,x) aleatorias con la forma que le especificemos; por defecto genera números aleatorios entre 0-1</pre>	cada fila ordenados en orden ascendente por defecto np.sort(array, axis = 0) devuelve un array con los valores de cada columna ordenados en orden ascendente	<pre>np.mod(array1, n) devuelve un array con el resto de la división entre el array y el valor de n np.cos(array) devuelve un array con el coseno de</pre>	df['columna'].std() Robustez	sns.displot(df['columna'], kde = True) crea un histograma que muestra la distribution de los valores import scipy.stats import skew
<pre>array = np.random.random_sample((z,y,x)) crea un array de floats aleatorias con la forma que le especificemos; por defecto genera números aleatorios entre 0-0.9999999</pre>	<pre>np.sort(-array) devuelve un array con los valores de cada fila ordenados en orden descendente</pre>	cada elemento del array <pre>np.sin(array)</pre> devuelve un array con el seno de cada elemento del array	- cuanto más cantidad de datos, más robustos 1/n donde n es el numero de registros Coeficiente de variación	skew(df['columna'] muestra el valor del sesgo de una variable
<pre>array = np.random.z,y,x=None) aleatorio en 0 y 0.999999999999999999999999999999999</pre>	<pre>np.round(array, decimals = x) devuelve un array con los valores del array redondeados a x decimales np.round(array, decimals = x) devuelve un array con los</pre>	<pre>np.sin(array) devuelve un array con la tangente de cada elemento del array</pre>	el cociente entre la desviación típica y la media; cuanto mayor sea, mayor será la dispersión en nuestros datos	Intervalos de confianza describe la variabilidad entre la medida obtenida en un estudio y la medida real de la población (el valor
de n decimales np.random.uniform(n,m, size = (z,y,x)) aleatorias de una distribución uniforme en el intervalo	valores del array redondeados a x decimales np.where(array > x) devuelve los indices de los valores que cumplan la condición, por fila y columna	Operaciones de comparación en arrays bidimensionales np.any(array > n) devuelve True o False segun si	df['columna'].std() / df['columna'].mean() Percentiles	real) import scipy.stats as st
entre n y m np.random.binomial(n,m, size = (z,y,x)) con una distribución binomial; n es el numero total de	Operaciones con arrays	cualquier valor del array cumpla con la condicion np.any(array > n, axis = b) True o False por cada columna o fila según si algún	<pre>divide datos ordenados de menor a mayor en cien partes; muestra la proporción de datos por debajo de su valor percentil_n = np.percentile(df['columna'], n) saca el valor</pre>	st.t.interval(alpha = n, df = len(df['columna']-1, loc = np.mean(df['columna']), scale = st.sem(df['columna']))
<pre>pruebas; m es la probabilidad de éxito np.random.normal(loc = n, scale = m, size = (z,y,x))</pre>	<pre>np.add(array1, array2) suma dos arrays np.subtract(array1, array2) resta el array2 del array1</pre>	valor de la fila o columna cumpla con la condición np.all(array > n) devuelve True o False segun si todos los valores del array cumpla con la condicion	en el percentil n Rangos intercuartílicos	devuelve el rango de valores para lo cual hay un n% de probabilidad que un valor real cae en ese rango alpha: porcentaje de confianza (p.ej. 90%, 95%, o 99%)
genera números aleatorios de una distribución normal (curva de campana); loc es la media; scale es la desviación estándar	<pre>np.multiply(array1, array2) np.divide(array1, array2) array2</pre>	<pre>np.all(array > n, axis = b) devuelve un array con True o False por cada columna o fila según si todos</pre>	<pre>medida de dispersión: diferencia entre cuartiles 75 y 25 q3, q1 = np.percentile(df["columna"], [75, 25]) saca los tercer y primer cuartiles</pre>	df: los datos loc: la media
<pre>np.random.permutation(array) mismos valores mezclados aleatoriamente</pre>	array + n, n * array, etc operadores algebraicos	los valores de la fila o columna cumplan con la condición	rango_intercuartílico = q3 - q1	scale: la desviación estándar