Descripción estadística del corpus

Instalación de librerías

Se usarán las siguientes librerías de la sesión pasada:

- **PANDAS**: Análisis de datos.
- **NUMPY**: Creación de vectores y matrices. Procesamiento matemático.
- **SEABORN**: Librería para visualización de datos.
- **MATPLOTLIB** Librería para visualización de datos.

```
!pip install pandas
!pip install numpy
!pip install seaborn
!pip install matplotlib
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.1.4)
Requirement already satisfied: numpy<2,>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (2.1.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seabor
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotli
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seab
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->sea
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.26.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (10.
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotli
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7-
```

Importar librerías

Comando de la sesión anterior.

- **import** LIBRARY.

Recordar que, si no es necesaria toda la librería, se puede importar solo una función de ella. Esto hace que el coste computacional y temporal sean más eficientes. La forma de hacerlo es la siguiente:

- **from** LIBRARY **import** PACKAGE.

- **from** OFFICE **import** WORD

```
import pandas as pd #es común que se usen abreviaturas para hacer más ligero el código. En este caso, se especific
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os
```

Cargar e instanciar corpus

```
corpus = pd.read_csv('CBspectral-measures.csv',  encoding='utf-8')
#corpus = pd.read_csv('Cuestionario.csv',  encoding='utf-8')
```

Exploración y visualización del corpus

Se usarán los comandos de la sesión anterior para tener información sobre el corpus.

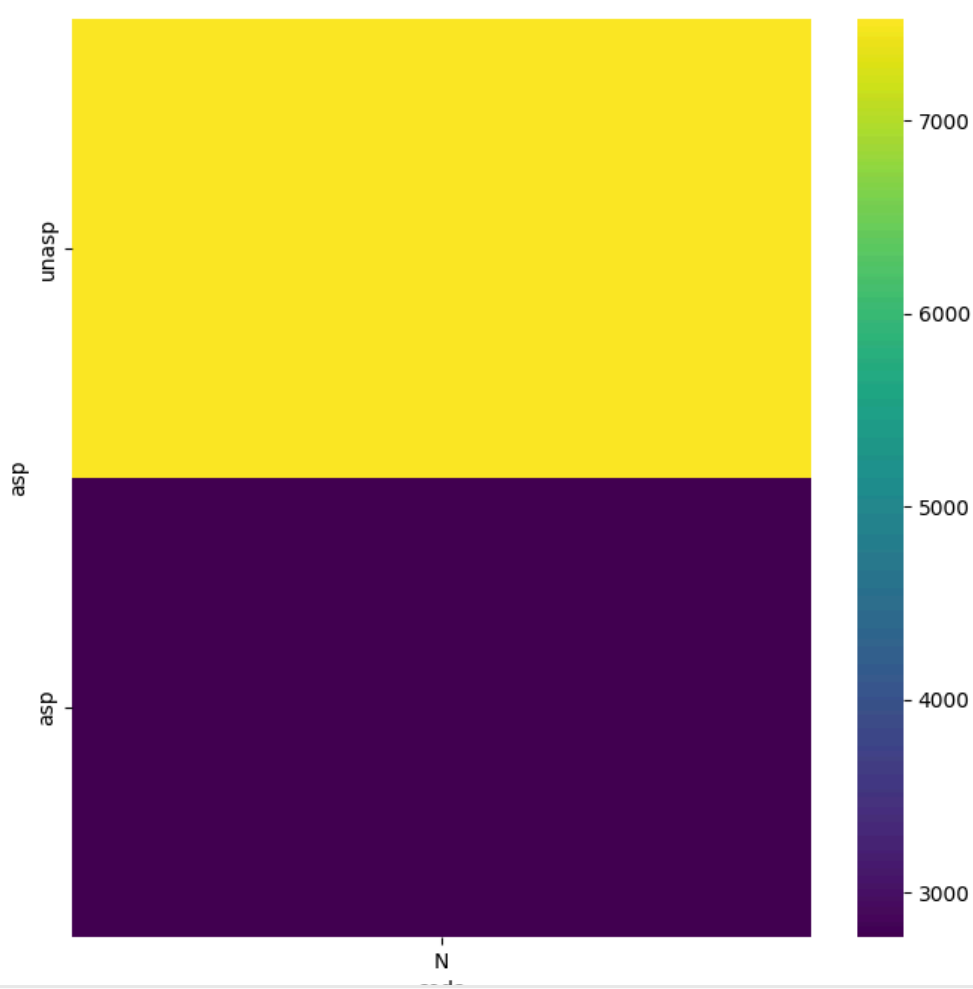
Recordar que la mayoría de estos comandos son funciones que se activan al crear variables y que ello hace que se activen herramientas específicas que leen la información de las variables, por ejemplo, .head, .info, .CABECERA.value_counts, etc.

```
corpus #primeras y últimas líneas del corpus
```

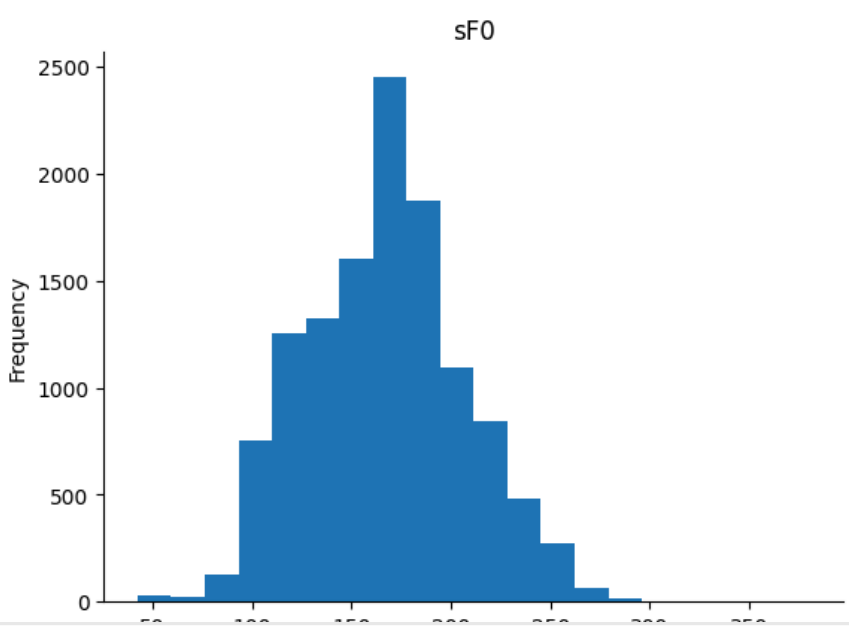
	subject	item	repetition	window	duration	sF0	strF0	pF0	CPP	H1H2c	...	nuc
0	M2	waa2	3	1	127.282	115.281143	115.728357	117.494571	24.968429	-4.857143	...	aa
1	M2	waa2	3	2	127.282	119.400333	119.014778	122.719667	22.630778	-4.437333	...	aa
2	M2	waa2	3	3	127.282	116.618111	116.471333	120.430111	22.010889	-4.654333	...	aa
3	M2	waa2	3	4	127.282	113.650556	113.756111	117.844222	22.498000	-4.820000	...	aa
4	M2	waa2	3	5	127.282	111.229000	111.078444	114.904333	23.665000	-5.294444	...	aa
...	...	...	...	...	...	...	...	...	...	...	...	...
12420	F1	paw1	1	7	369.473	257.627813	257.592938	266.801312	19.061906	10.254781	...	a
12421	F1	paw1	1	8	369.473	233.443219	233.334219	243.518313	18.732250	10.389812	...	a
12422	F1	paw1	1	9	369.473	209.254188	209.296469	218.284687	17.322406	7.271875	...	a
12423	F1	paw1	1	10	369.473	194.228500	193.977969	199.294187	18.685844	6.761875	...	a
12424	F1	paw1	1	11	369.473	188.398778	187.114741	189.745741	17.816407	5.335630	...	a

12425 rows × 32 columns

```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
plt.subplots(figsize=(8, 8))
df_2dhist = pd.DataFrame({
    x_label: grp['asp'].value_counts()
    for x_label, grp in corpus.groupby('coda')
})
sns.heatmap(df_2dhist, cmap='viridis')
plt.xlabel('coda')
_ = plt.ylabel('asp')
```




```
from matplotlib import pyplot as plt
corpus['sF0'].plot(kind='hist', bins=20, title='sF0')
plt.gca().spines[['top', 'right',]].set_visible(False)
```




```
corpus.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12425 entries, 0 to 12424
Data columns (total 32 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   subject     12425 non-null  object
 1   item        12425 non-null  object
 2   repetition  12425 non-null  int64
 3   window      12425 non-null  int64
 4   duration    12425 non-null  float64
 5   sF0         12228 non-null  float64
 6   strF0       12425 non-null  float64
 7   pF0         12250 non-null  float64
 8   CPP         12425 non-null  float64
 9   H1H2c       12425 non-null  float64
 10  H2H4c       12425 non-null  float64
 11  H1A1c       12425 non-null  float64
 12  H1A2c       12425 non-null  float64
 13  H1A3c       12425 non-null  float64
 14  H42Kc       12425 non-null  float64
 15  H2KH5Kc     12425 non-null  float64
 16  Energy      12403 non-null  float64
 17  HNR05       12425 non-null  float64
 18  HNR15       12425 non-null  float64
 19  HNR25       12425 non-null  float64
 20  HNR35       12425 non-null  float64
 21  onset       12425 non-null  object
 22  nuc         12425 non-null  object
 23  coda        10300 non-null  object
 24  asp         12425 non-null  object
 25  tone        12425 non-null  int64
 26  voicing     12425 non-null  object
 27  semitones   12425 non-null  float64
 28  register    12425 non-null  object
 29  contour     12425 non-null  object
 30  manner      12425 non-null  object
 31  pct         12425 non-null  float64
dtypes: float64(19), int64(3), object(10)
memory usage: 3.0+ MB
```

```
corpus.Energy.value_counts()
```

Energy	count
0.001000	233
0.003000	127
0.002000	115
0.004000	94
0.005000	68
...	...
0.052571	1
0.061143	1
0.137786	1
0.124357	1
0.012844	1

9098 rows × 1 columns

```
corpus.voicing.unique()
```

```
array(['voiced', 'voiceless'], dtype=object)
```

```
corpus.voicing.value_counts()
```

voicing	count
voiced	9098
voiceless	3327

```
corpus
```

	subject	item	repetition	window	duration	sF0	strF0	pF0	CPP	H1H2c	...	nuc
0	M2	waa2	3	1	127.282	115.281143	115.728357	117.494571	24.968429	-4.857143	...	aa
1	M2	waa2	3	2	127.282	119.400333	119.014778	122.719667	22.630778	-4.437333	...	aa
2	M2	waa2	3	3	127.282	116.618111	116.471333	120.430111	22.010889	-4.654333	...	aa
3	M2	waa2	3	4	127.282	113.650556	113.756111	117.844222	22.498000	-4.820000	...	aa
4	M2	waa2	3	5	127.282	111.229000	111.078444	114.904333	23.665000	-5.294444	...	aa
...	...	...	...	...	...	...	...	...	...	...	...	...
12420	F1	paw1	1	7	369.473	257.627813	257.592938	266.801312	19.061906	10.254781	...	a
12421	F1	paw1	1	8	369.473	233.443219	233.334219	243.518313	18.732250	10.389812	...	a
12422	F1	paw1	1	9	369.473	209.254188	209.296469	218.284687	17.322406	7.271875	...	a
12423	F1	paw1	1	10	369.473	194.228500	193.977969	199.294187	18.685844	6.761875	...	a
12424	F1	paw1	1	11	369.473	188.398778	187.114741	189.745741	17.816407	5.335630	...	a

12425 rows × 32 columns

```
corpus.rmul(2)    #método para multiplicar valores
```

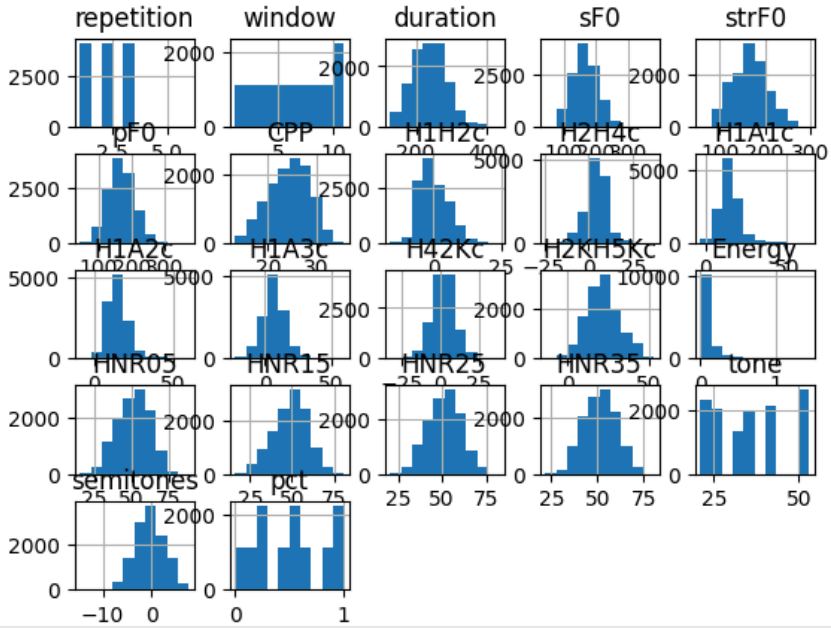
	subject	item	repetition	window	duration	sF0	strF0	pF0	CPP	H1H2c	...
0	M2M2	waa2waa2	6	2	254.564	230.562286	231.456714	234.989143	49.936857	-9.714286	...
1	M2M2	waa2waa2	6	4	254.564	238.800667	238.029556	245.439333	45.261556	-8.874667	...
2	M2M2	waa2waa2	6	6	254.564	233.236222	232.942667	240.860222	44.021778	-9.308667	...
3	M2M2	waa2waa2	6	8	254.564	227.301111	227.512222	235.688444	44.996000	-9.640000	...
4	M2M2	waa2waa2	6	10	254.564	222.458000	222.156889	229.808667	47.330000	-10.588889	...
...	...	...	...	...	...	...	...	...	...	...	...
12420	F1F1	paw1paw1	2	14	738.946	515.255625	515.185875	533.602625	38.123812	20.509563	...
12421	F1F1	paw1paw1	2	16	738.946	466.886437	466.668437	487.036625	37.464500	20.779625	...
12422	F1F1	paw1paw1	2	18	738.946	418.508375	418.592938	436.569375	34.644813	14.543750	...
12423	F1F1	paw1paw1	2	20	738.946	388.457000	387.955938	398.588375	37.371688	13.523750	...
12424	F1F1	paw1paw1	2	22	738.946	376.797556	374.229481	379.491481	35.632815	10.671259	...

12425 rows × 32 columns

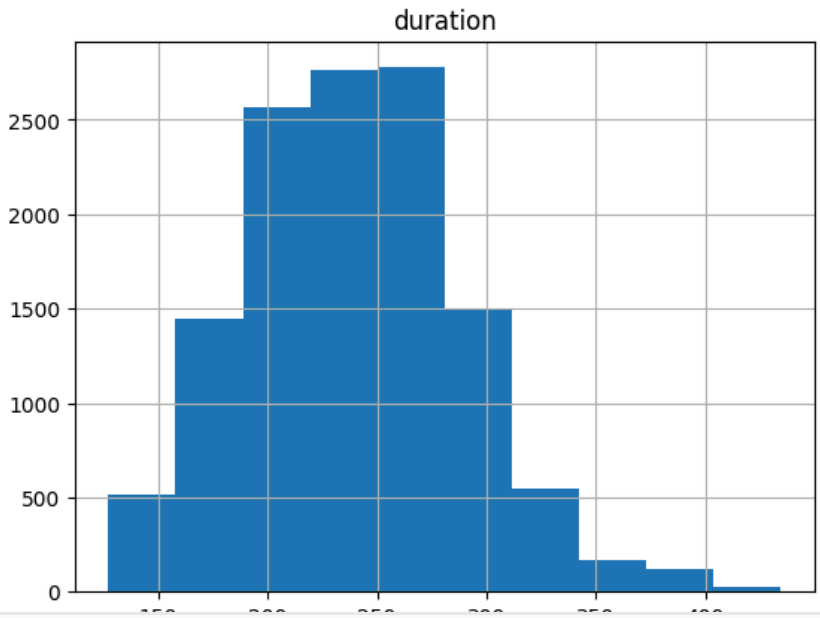
Otras funciones para visualizar los datos: histogramas

```
corpus.hist() #genera histogramas a partir de los datos del corpus
```

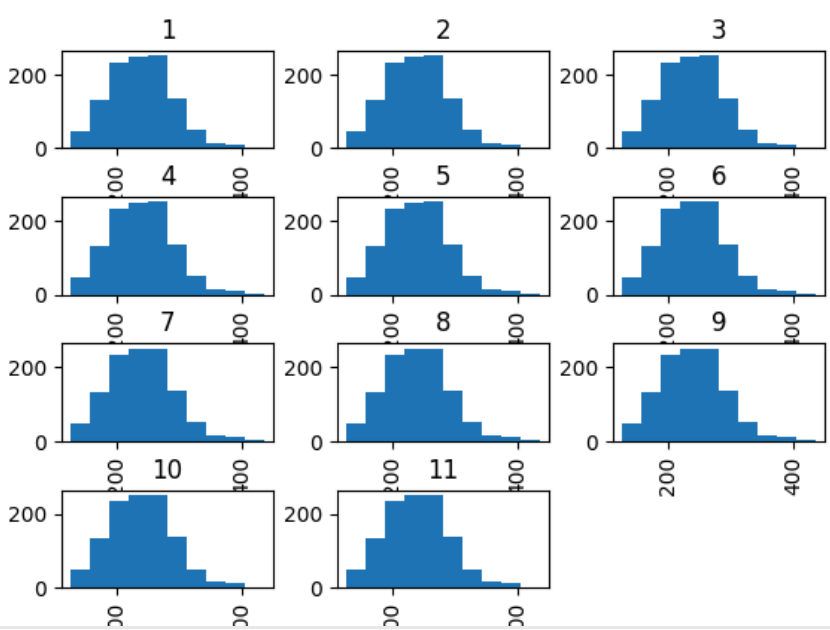
```
array([[<Axes: title={'center': 'repetition'}>,
        <Axes: title={'center': 'window'}>,
        <Axes: title={'center': 'duration'}>,
        <Axes: title={'center': 'sF0'}>,
        <Axes: title={'center': 'strF0'}>],
       [<Axes: title={'center': 'pF0'}>, <Axes: title={'center': 'CPP'}>,
        <Axes: title={'center': 'H1H2c'}>,
        <Axes: title={'center': 'H2H4c'}>,
        <Axes: title={'center': 'H1A1c'}>],
       [<Axes: title={'center': 'H1A2c'}>,
        <Axes: title={'center': 'H1A3c'}>,
        <Axes: title={'center': 'H42Kc'}>,
        <Axes: title={'center': 'H2KH5Kc'}>,
        <Axes: title={'center': 'Energy'}>],
       [<Axes: title={'center': 'HNR05'}>,
        <Axes: title={'center': 'HNR15'}>,
        <Axes: title={'center': 'HNR25'}>,
        <Axes: title={'center': 'HNR35'}>,
        <Axes: title={'center': 'tone'}>],
       [<Axes: title={'center': 'semitones'}>,
        <Axes: title={'center': 'pct'}>, <Axes: >, <Axes: >, <Axes: >]],
      dtype=object)
```




```
corpus.hist('duration') #para hacer el histograma de una columna específica
plt.savefig('duration.png') #guardar el histograma en FILE.png
```




```
corpus.hist('duration', 'window') #comparación de datos
plt.savefig('comparisson.pdf')
```




```
#ejemplo para guardar una muestra aleatoria del corpus
corpus_150 = corpus.sample(150) #guarda 150 líneas en la variable

#si se quiere guardar los datos que están en la variable, se puede usar el siguiente código
file1 = open ('corpus_150.txt', 'w')
file1.write(str((corpus_150)))
file1.close()


corpus_150
```

	subject	item	repetition	window	duration	sF0	strF0	pF0	CPP	H1H2c	...	nuc
3584	M2	waan3	2	10	251.711	139.462524	139.411714	140.562000	21.388238	8.433333	...	aa
3300	M2	bang5	3	1	240.709	109.876550	109.073550	110.398800	22.191300	0.373400	...	a
3565	M2	waan3	1	2	251.232	159.472095	158.185143	160.003381	29.226143	-1.837143	...	aa
9724	F1	baaj3	2	9	234.416	215.920105	218.181263	215.179211	25.825316	-0.081737	...	aa
8698	F1	waa3	2	2	191.978	227.225867	226.528133	224.052133	25.795867	-3.531267	...	aa
...	...	...	...	...	...	...	...	...	...	...	...	...
2340	M2	bhaang6	1	9	211.361	94.367588	94.293588	92.279882	16.902471	-3.605706	...	aa
8996	F1	bhaa6	3	3	207.447	155.412529	155.202235	156.840824	27.228588	-3.848118	...	aa
217	M2	maa1	2	9	149.953	156.654364	156.800727	156.856273	28.136727	1.718182	...	aa
6974	M1	maaj4	1	11	278.605	156.772783	156.799522	158.630435	22.373478	-0.667870	...	aa
11448	F1	bhaan4	1	1	279.757	186.497182	186.960519	193.136222	19.801852	-1.642963	...	aa

150 rows × 32 columns

Estadística general con la función .describe()

```
corpus.shape #columnas por filas
```

```
(12425, 32)
```

```
corpus.describe()
```

	repetition	window	duration	sF0	strF0	pF0	CPP	H1H2c	
count	12425.00000	12425.000000	12425.000000	12228.000000	12425.000000	12250.000000	12425.000000	12425.000000	12425.0
mean	2.00837	5.998551	238.673984	167.017001	167.028633	167.725490	24.306411	0.164720	4.2
std	0.83069	3.162659	49.694000	39.522986	38.900939	39.267786	4.098053	5.239892	5.8
min	1.00000	1.000000	127.282000	42.533667	56.855444	42.644524	13.316625	-15.803154	-24.3
25%	1.00000	3.000000	201.912000	138.095800	138.131538	138.981574	21.369739	-3.802684	1.0
50%	2.00000	6.000000	237.344000	167.834021	167.620600	168.321505	24.609474	-0.832789	4.9
75%	3.00000	9.000000	271.429000	191.470088	191.407700	192.470073	27.433704	3.803824	7.7
max	6.00000	11.000000	434.372000	380.638250	297.963538	375.098150	35.448565	24.197563	36.5

8 rows × 22 columns

```
file2 = open ('corpus_describe.txt', 'w')
file2.write(str((corpus.describe())))
file2.close()
```

Estadística descriptiva

Métricas descriptivas sobre los datos:

- Media
- Mediana
- Moda

- Desviación estándar
- Varianza

```
corpus.sF0.mean()
```

```
167.01700075840418
```

```
corpus.sF0.median()
```

```
167.834021381579
```

```
corpus.sF0.mode()
```

	sF0
0	139.961500
1	145.230800
2	147.328000
3	159.638000
4	162.063750
5	163.609667
6	183.922417
7	250.902000

```
corpus.sF0.std()
```

```
39.52298606758945
```

```
corpus.sF0.var()
```

```
1562.06642769887
```

```
corpus.sF0.min()
```

```
42.5336666666667
```

```
corpus.sF0.max()
```

```
380.63825
```

```
corpus.sF0.quantile()
```

```
167.834021381579
```

```
corpus.sF0.describe()
```

	sF0
count	12228.000000
mean	167.017001
std	39.522986
min	42.533667
25%	138.095800
50%	167.834021
75%	191.470088
max	380.638250

```
corpus.groupby('duration').agg({'Energy': ['mean', 'std', 'var']}).reset_index()
```

	duration	Energy		
		mean	std	var
0	127.282	0.096288	0.024938	0.000622
1	128.025	0.084864	0.017912	0.000321
2	128.556	0.019591	0.007609	0.000058
3	130.445	0.216809	0.054279	0.002946
4	132.005	0.057157	0.009164	0.000084
...	...	...	...	...
1122	394.096	0.115818	0.078858	0.006219
1123	394.656	0.246830	0.273242	0.074661
1124	395.154	0.064929	0.055890	0.003124
1125	416.977	0.087477	0.094223	0.008878
1126	434.372	0.269770	0.289955	0.084074

```
corpus.groupby('duration').agg({'sF0': ['mean']}).reset_index()
```

	duration	sF0 mean
0	127.282	110.143659
1	128.025	110.552080
2	128.556	93.250492
3	130.445	145.290314
4	132.005	123.027203
...	...	...
1122	394.096	168.906993
1123	394.656	176.572284
1124	395.154	169.812398
1125	416.977	142.898018
1126	434.372	185.415411

```
distN_Energy = sns.distplot(corpus.Energy)
```

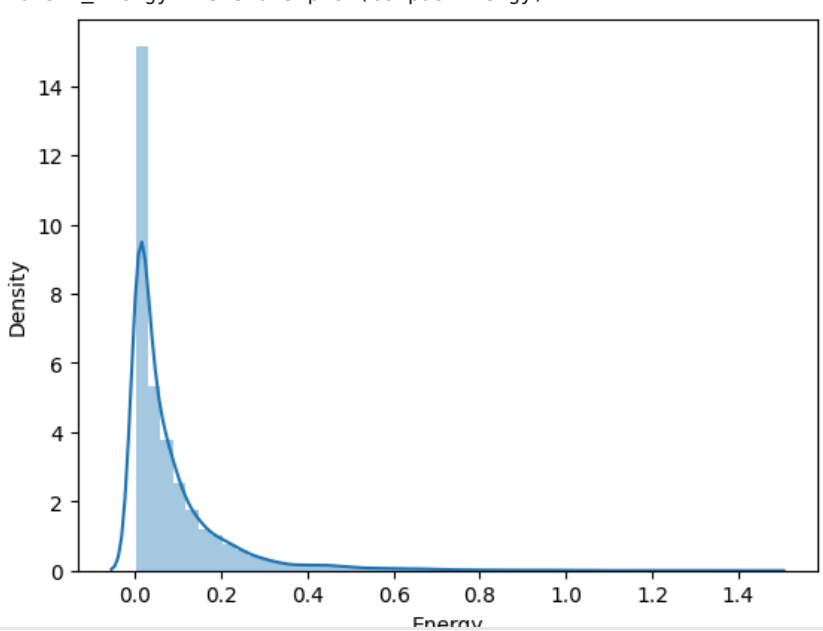
```
<ipython-input-32-5627ef83eee6>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  distN_Energy = sns.distplot(corpus.Energy)
```




```
distN_tone = sns.distplot(corpus.tone)
```

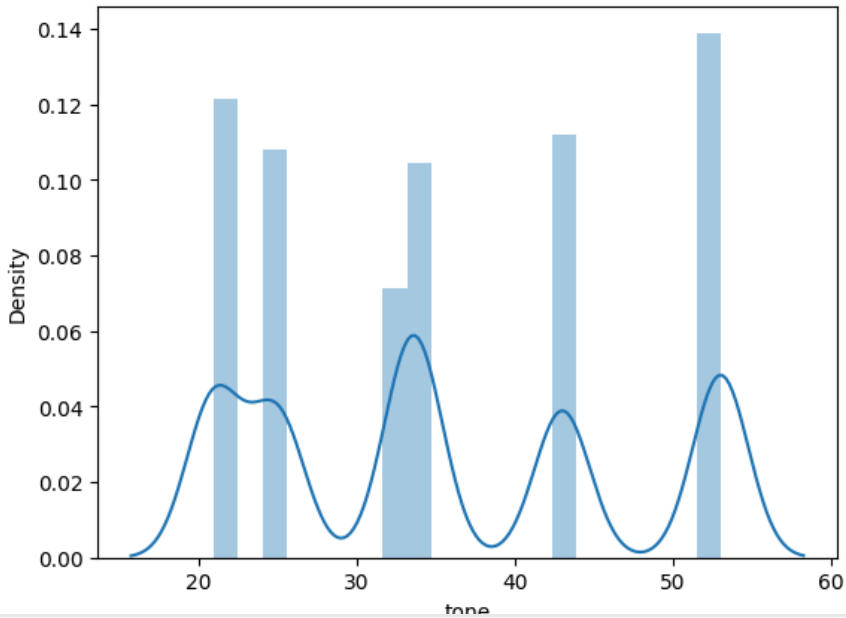
```
<ipython-input-33-6e5bc5bff8bf>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
distN_tone = sns.distplot(corpus.tone)
```




```
corpus.manner.value_counts()
```

manner	count
obstruent	6318
sonorant	4919
fricative	1188

Anotación POS

En esta sección veremos cómo se puede anotar el corpus con etiquetas de categoría gramatical de forma automática e iterativa. Para ello, usaremos el software TreeTagger.

```
lexicon = corpus['manner'].astype(str)
file2 = open ('lexicon.txt', 'w')
file2.write(str((lexicon)))
file2.close()
```

```
!sh install-tagger.sh
```

```
mkdir: cannot create directory ‘cmd’: File exists
mkdir: cannot create directory ‘lib’: File exists
mkdir: cannot create directory ‘bin’: File exists
mkdir: cannot create directory ‘doc’: File exists
```

```
TreeTagger version for PC-Linux installed.
Tagging scripts installed.
English BNC parameter file installed.
Spanish parameter file installed.
Path variables modified in tagging scripts.

You might want to add /content/cmd and /content/bin to the PATH variable so that you do not need to specify th
```

```
!echo 'this is a test' | cmd/tree-tagger-english-bnc
```

```
         reading parameters ...
         tagging ...
this     DT0     this
is       VBZ     be
a        AT0     a
test     NN1     test
          finished.
```

```
!cat lexicon.txt | cmd/tree-tagger-english-bnc > lexicon_annotated.txt
```

```
         reading parameters ...
         tagging ...
          finished.
```

Refuerzo

Replica los mismos procesos pero con los datos de un nuevo dataset. Usa el archivo *experiencia.xlsx* para probar con la lectura de archivos de Excel.

```
datos = pd.read_excel('experiencia.xlsx')
```

```
datos
```

	ID	Inicio	Final	Email	Matrícula	Edad	Género	Nacionalidad	Nacionalidad2	Pre-escolar	...	ChatG
0	1	2024-04-17 12:41:16	2024-04-17 12:45:33	anonymous	110206452	18	Femenino	Mexicana	NaN	Pública	...	Menos ve
1	2	2024-04-17 12:42:07	2024-04-17 12:47:27	anonymous	110244664	19	Masculino	Mexicana	NaN	Pública	...	Diaria varias
2	3	2024-04-17 12:41:00	2024-04-17 12:51:06	anonymous	110256327	17	Femenino	Mexicana	NaN	Pública	...	[varias la
3	4	2024-04-17 12:42:13	2024-04-17 12:51:26	anonymous	110221592	17	Femenino	Mexicana	NaN	Privada	...	Diaria varias
4	5	2024-04-17 12:42:56	2024-04-17 12:51:29	anonymous	110257507	17	Masculino	Doble nacionalidad	Mexicana y estadounidense	Pública	...	[varias
5	6	2024-04-17 12:41:15	2024-04-17 12:52:37	anonymous	110213560	17	Femenino	Mexicana	NaN	Privada	...	[varias la
6	7	2024-04-17 12:45:06	2024-04-17 12:54:08	anonymous	110221232	17	Masculino	Mexicana	.	Pública	...	[varias la
7	8	2024-04-17 12:41:05	2024-04-17 12:54:28	anonymous	110217632	20	Femenino	Mexicana	NaN	Pública	...	[varias
8	9	2024-04-17 12:45:24	2024-04-17 12:54:37	anonymous	110228933	17	Masculino	Mexicana	NaN	Ambas	...	[varias la
9	10	2024-04-17 12:45:16	2024-04-17 12:54:40	anonymous	110141394	18	Masculino	Mexicana	NaN	Privada	...	Diaria varias
10	11	2024-04-17 12:44:32	2024-04-17 12:55:08	anonymous	110245742	17	Masculino	Mexicana	NaN	Pública	...	[varias la

```
datos.describe()
```

ID	Inicio	Final	Matrícula	Edad	Calificación

```
datos.dtypes
```

	0
ID	int64
Inicio	datetime64[ns]
Final	datetime64[ns]
Email	object
Matrícula	int64
Edad	int64
Género	object