

Trazado de tablas

La función `table()` de Matplotlib recibe como parámetros los textos a colocar en las celdas de la tabla y puedes usarlo en cualquier gráfico, independientemente de la plataforma de trazado empleada. Continuaremos usando el `dataframe` `languages`, que almacena el contenido del archivo **Lenguajes.csv** (ubicado en el apartado "Archivos adjuntos" de la plataforma), para mostrar cómo usar dicha función.

Filtremos el `dataframe` `languages` para mantener la información de los idiomas español, inglés, chino, francés, ruso, italiano y alemán:

```
some_languages = languages.loc[languages['Language'].isin(['Spanish','English','Chinese','French','Russian','Italian','German'])]
```

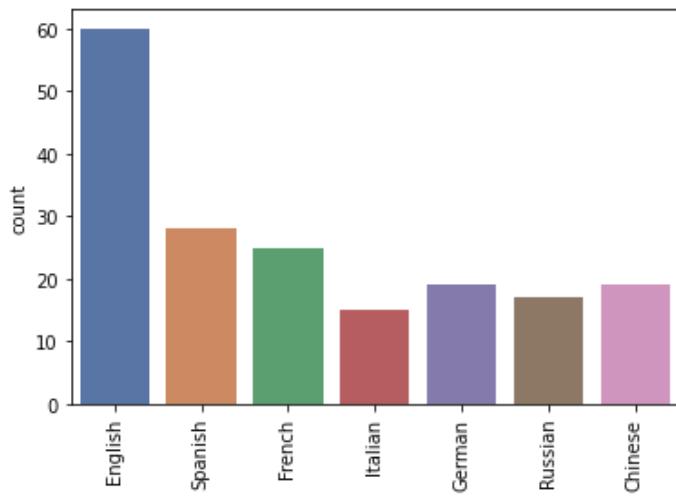
Lo que resulta en un `dataframe` de 183 registros, como se muestra en la siguiente figura:

	CountryCode	Language	IsOfficial	Percentage
1	ABW	English	F	9.5
3	ABW	Spanish	F	7.4
18	AIA	English	T	0.0
23	AND	French	F	6.2
...
949	VUT	French	T	14.2
952	WSM	English	T	0.6
964	ZAF	English	T	8.5
980	ZWE	English	T	2.2

183 rows × 4 columns

Usemos una gráfica de barras para determinar el conteo o frecuencia de cada idioma. Si recuerdas, anteriormente revisamos que la única plataforma que ofrece una función directa de trazado con este fin es Seaborn: `countplot()`. En Matplotlib y Pandas usamos `value_counts()` como auxiliar en las funciones `bar()` y `plot.bar()`, respectivamente, para obtener los mismos resultados.

```
sns.countplot(x='Language',data=some_languages)  
plt.xticks(rotation=90)
```



En la figura resultante se observa que el idioma predominante es el inglés (casi 60 países lo incluyen como parte de sus lenguas), seguido por el español y el francés. Ahora, si quisieras incluir los valores numéricos en el gráfico, podrías utilizar la función `text()` o una tabla de datos en el eje horizontal, como veremos a continuación. Pero antes, es necesario generar el *dataframe* con los conteos por idioma. Para ello, hemos utilizado ya dos formas: con la función `value_counts()` y con `groupby()`. Recordemos cómo:

Con value_counts()

```
counts =  
pd.DataFrame(some_languages['La  
nguage'].value_counts())
```

Language	
English	60
Spanish	28
French	25
German	19
Chinese	19
Russian	17
Italian	15

Con groupby()

```
counts =  
some_languages.groupby(['Language'])  
.count()[['CountryCode']]
```

CountryCode	
Language	CountryCode
Chinese	19
English	60
French	25
German	19
Italian	15
Russian	17
Spanish	28

Aunque en ambas tablas obtienes los mismos valores, difieren en los nombres de las columnas y en el orden de los registros. Puedes tomar cualquiera como base y renombrar sus elementos para que el *dataframe* resulte más legible:



```
counts.index.name = 'Language'  
counts.columns = ['Count']
```

Count	
Language	Count
English	60
Spanish	28
French	25
German	19
Chinese	19
Russian	17
Italian	15

Sin embargo, para incluir la tabla con esta información en el gráfico, el formato del *dataframe* debe coincidir con la forma que adquirirá la tabla. Viendo el gráfico de barras antes generado, lo más conveniente es que los idiomas se presenten de manera horizontal con los conteos abajo.

Esto lo puedes lograr usando la propiedad `T` para obtener la transpuesta:

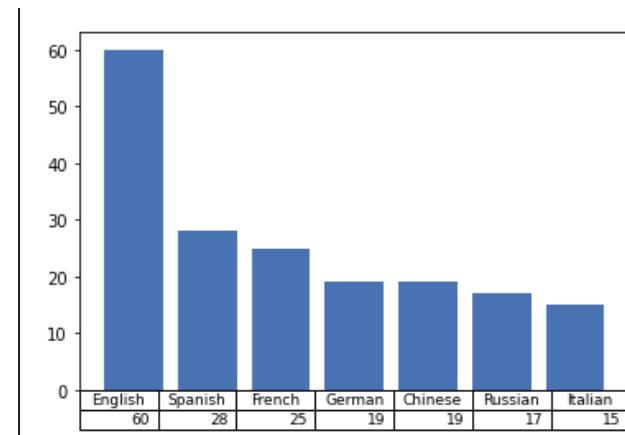
```
counts_T = counts.T
```

Language	English	Spanish	French	German	Chinese	Russian	Italian
Count	60	28	25	19	19	17	15

Ahora ya es posible incluir la tabla en gráficos de barras con las tres plataformas:

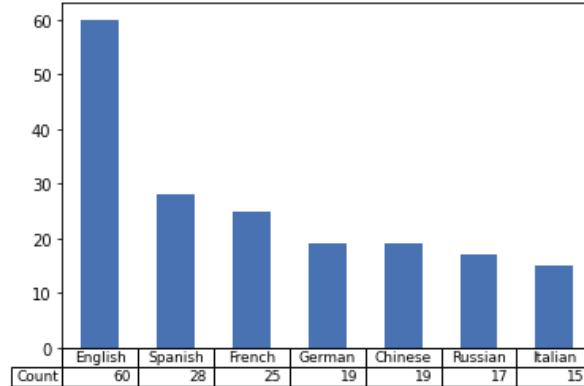
Con Matplotlib

```
plt.bar(counts.index, 'Count', data=counts)  
counts_T = counts.T  
plt.table(cellText=counts_T.values, colLabels=  
counts_T.columns)  
plt.xticks([])
```



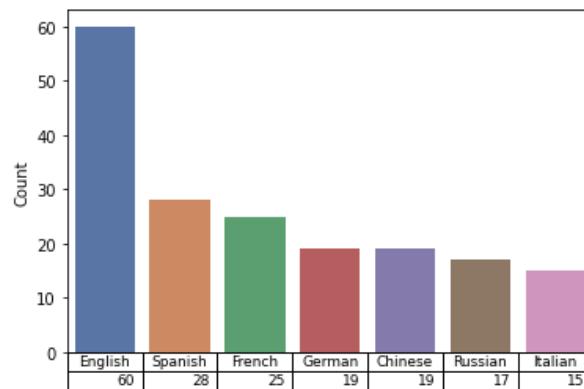
Con Pandas

```
counts.plot(kind='bar',table=True,xticks=[],x  
label='',legend='')
```



Con Seaborn

```
sns.barplot(x=counts.index,y='Count',data=counts)  
counts_T = counts.T  
plt.table(cellText=counts_T.values,colLabels=counts_T.colu  
mns)  
plt.xticks([])  
plt.xlabel('')
```

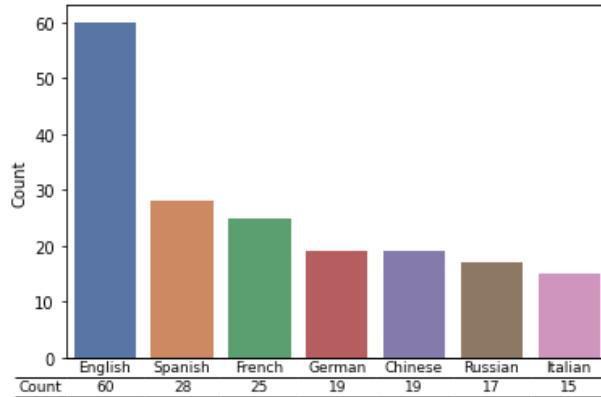


Observa de los códigos anteriores que:

- En Matplotlib y Seaborn ocupamos el **dataframe** `counts_T` para construir la tabla y pasamos una lista vacía a la función `xticks()` para omitir las marcas del eje x.
- La función `plot()` de Pandas posee el parámetro `table` para incluir la tabla de datos directamente, sin necesidad de la transpuesta, y los parámetros `xticks` y `xlabel` para ocultar los valores del eje x. Sin embargo, si deseas personalizar su estructura o formato, deberás usar `table()` de Matplotlib.
- En Seaborn ya no es necesario ocupar `countplot()`, como inicialmente lo hicimos, porque se tiene el conteo en el **dataframe** `counts` y es éste el que se grafica.

Enfoquémonos a continuación en las propiedades de la tabla. Si quieres incluir encabezados para los registros o filas, utiliza la propiedad `rowLabels`:

```
plt.table(cellText=counts_T.values,colLabels=counts_T.columns,  
rowLabels=counts_T.index,cellLoc='center',edges='horizontal')
```



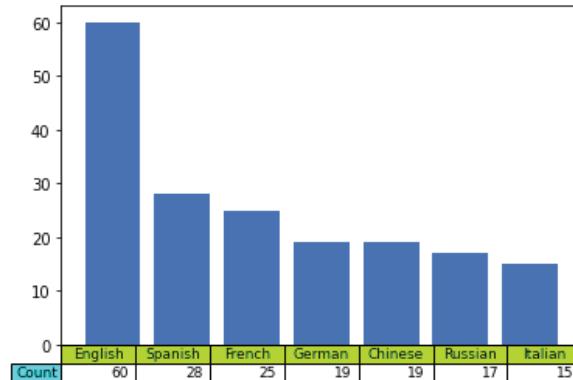
También puedes especificar la alineación de las celdas con `cellLoc` y podrías hacer lo mismo con los encabezados de filas y columnas, utilizando `rowLoc` y `colLoc`, respectivamente.

El parámetro `edges` te permite indicar qué líneas de la tabla incluir. En este ejemplo, únicamente mostramos las horizontales, pero podrías usar:

```
'open' sin líneas, 'closed' todas las líneas, 'vertical' líneas verticales
```

Cuando la tabla posee todas las líneas (`closed` como valor por defecto), es posible colorear las celdas de los encabezados, a través de los parámetros `rowColours` y `colColours`:

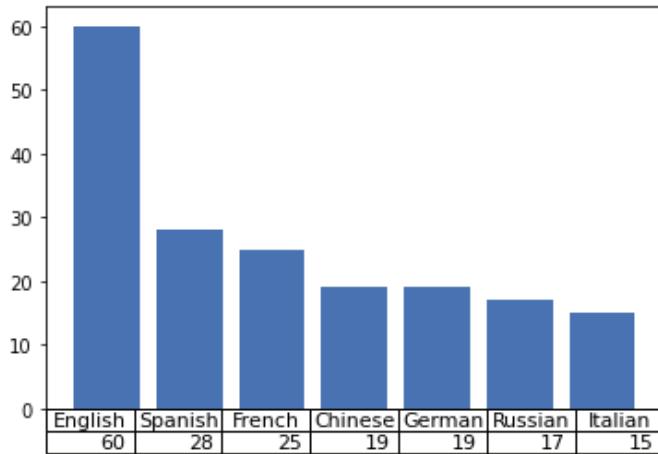
```
plt.table(cellText=counts_T.values, colLabels=counts_T.columns,
rowLabels=counts_T.index, rowColours=['cyan'],
colColours=['greenyellow']*7)
```



Aquí se ha usado el mismo color para todas las columnas (indicado por el 7 en `colColours`), pero podrías asignar un color particular a cada una, incluyéndolos en la lista.

Para cambiar el tamaño de la letra, debes hacerlo con las propiedades del objeto tabla creado con la función `table()` e identificado en el código como `t`:

```
t = plt.table(cellText=counts_T.values, colLabels=counts_T.columns)
t.auto_set_font_size(False)
t.set_fontsize(11)
```

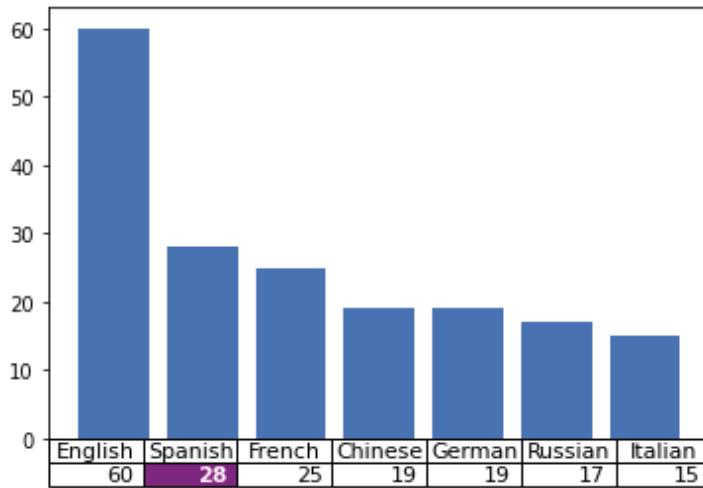


Antes de especificar el tamaño de la letra, deshabilita el autoajuste, que reduce el tamaño de la fuente hasta que el texto se ajusta al ancho de la celda.

También puedes indexar directamente el objeto tabla para acceder a celdas individuales, como lo haces con las matrices: `cell = table[row, col]`. Una vez referenciada la celda, podrías darle el formato que deseas.

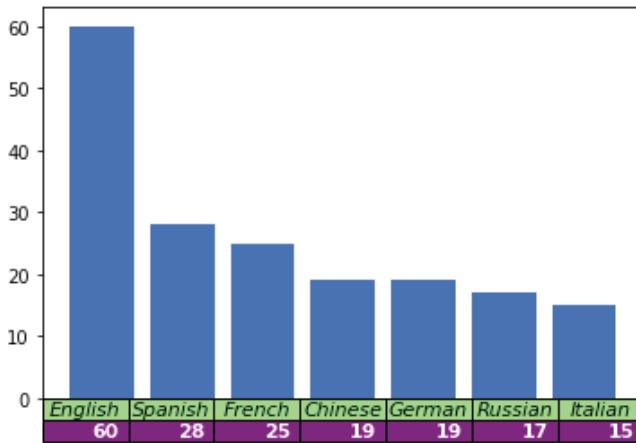
Por ejemplo, para hacer énfasis en el español, tendrías que indicar `row = 1` y `col = 1` (recuerda que los índices siempre inician con 0):

```
cell = t[1,1]
cell.set_text_props(weight='bold', color='white')
cell.set_facecolor('purple')
```



Para dar formato a varias celdas, tendrías que hacer un recorrido por la tabla, usando un ciclo y ocupando la función `get_cellld()`, que devuelve un diccionario de celdas con el mapeo de la tabla (fila, columna) a celdas:

```
for (row,col), cell in t.get_cellld().items():
    if (row == 0):
        cell.set_text_props(weight='light',style='italic')
        cell.set_facecolor('palegreen')
    else:
        cell.set_text_props(weight='bold',color='white')
        cell.set_facecolor('purple')
```



El cambio de color también puede resultar conveniente cuando se están representando varias series de datos, porque puede ocuparse en sustitución de la leyenda. Para exemplificarlo, divide el conteo anterior, en países, donde el idioma se habla como lengua oficial o no:

```
counts_isofficial = some_languages.groupby(['IsOfficial','Language']).count() [['CountryCode']]
counts_isofficial.columns=['Count']
```

Count		
	IsOfficial	Language
F	Chinese	17
	English	16
	French	7
	German	13
	Italian	11
	Russian	14
	Spanish	8
T	Chinese	2
	English	44
	French	18
	German	6
	Italian	4
	Russian	3
	Spanish	20

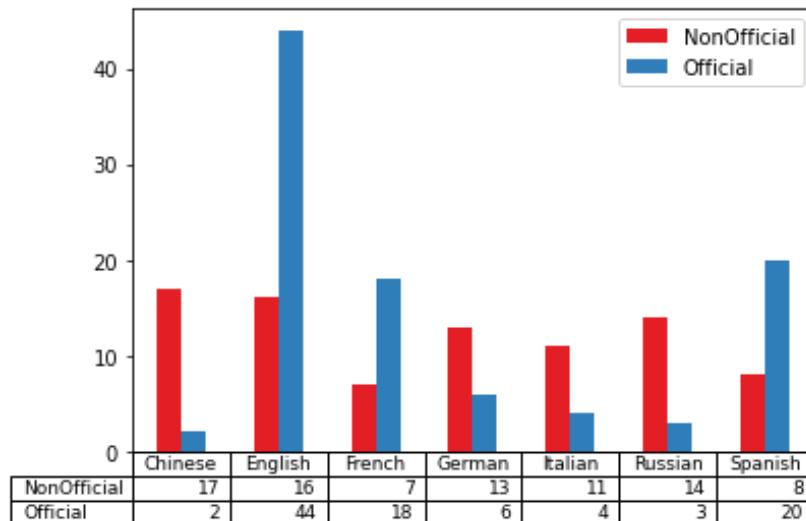
Acomoda la información de forma tal que los conteos queden en columnas independientes haciendo diferenciación por el valor de la columna `IsOfficial`:

```
counts_isofficial =  
counts_isofficial.iloc[:7,:].merge(counts_isofficial.iloc[7:,:],  
on='Language',suffixes=('_F', '_T'))[['Count_F','Count_T']]  
counts_isofficial.columns=['NonOfficial','Official']
```

Language	NonOfficial	Official
Chinese	17	2
English	16	44
French	7	18
German	13	6
Italian	11	4
Russian	14	3
Spanish	8	20

Si graficas el `dataframe` resultante usando Pandas tendrías:

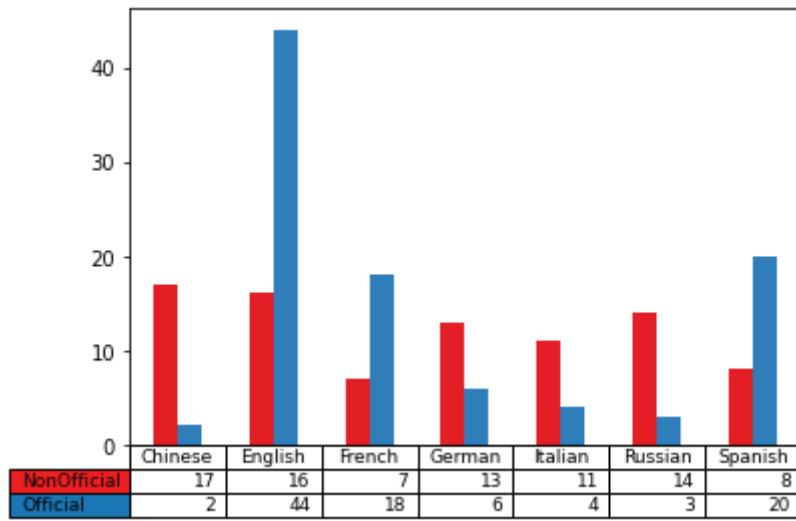
```
counts_isofficial.plot.bar(table=True,xticks=[],xlabel='')
```



*Se utilizó la paleta Set1.

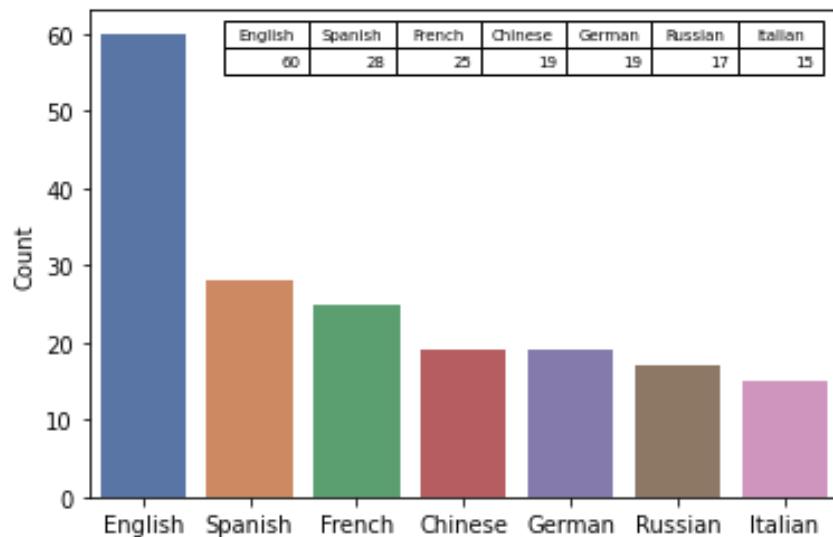
Para indicar los colores en la tabla, es necesario que uses la función `table()` de Matplotlib porque `plot()` de Pandas no permite tal nivel de personalización. No olvides que, en este caso, tendrás que ocupar la propiedad `T` para obtener la transpuesta:

```
counts_isofficial.plot.bar(xticks=[],xlabel='',legend='')  
counts_isofficial_T = counts_isofficial.T  
plt.table(cellText=counts_isofficial_T.values,colLabels=counts_isofficial_T.columns,rowLabels=counts_isofficial_T.index,rowColours=['tab:red','tab:blue'])
```



Aunque en todos los ejemplos previos ubicamos la tabla en el eje horizontal, para utilizar los encabezados de columna como marcas del eje, la posición de la tabla puedes cambiarla a través del parámetro `loc`:

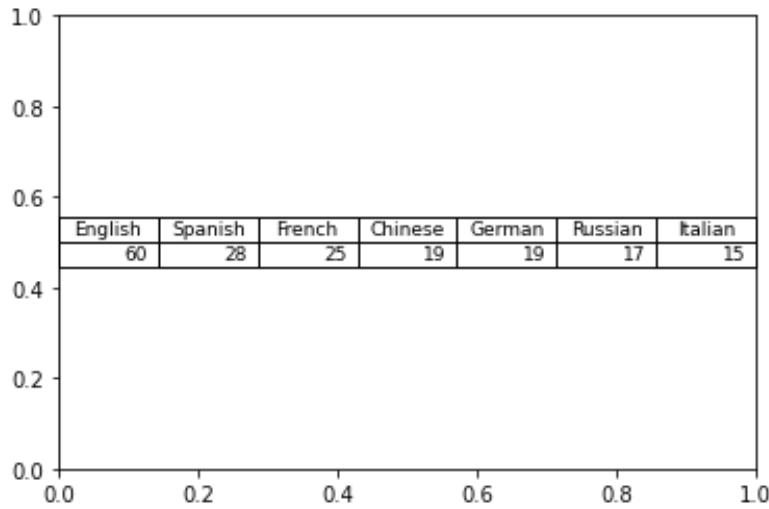
```
t = plt.table(cellText=counts_T.values, colLabels=counts_T.columns,
loc='upper right')
t.scale(0.8,1)
```



También escalamos la tabla con la función `scale(xscale,yscale)`. Para el ancho utiliza un factor de 0.8 que la reduce horizontalmente. El alto se ha mantenido con el tamaño original.

Incluso puedes crear una figura sin gráfico con la función `subplot()`, en la que únicamente incluyas la información contenida en la tabla:

```
fig,ax = plt.subplots(1,1)
ax.axis('off')
ax.table(cellText=counts_T.values,colLabels=counts_T.columns,loc='center')
```



Para que no se muestren los ejes, ni sus marcas, utiliza `axis('off')`:

```
fig,ax = plt.subplots(1,1)
ax.axis('off')
ax.table(cellText=counts_T.values,colLabels=counts_T.columns,loc='center')
```

English	Spanish	French	Chinese	German	Russian	Italian
60	28	25	19	19	17	15