Python pandas Cheat Sheet

by julenx via cheatography.com/168626/cs/35356/

Fijar el estilo gráfico de seaborn por defecto

sns.set()

Operaciones básicas con series

s = pd.Series(list, index = list)

Crear serie. Si en vez de una serie se introduce un escalar, todas las filas tendrán ese valor.

s = pd.Series(d)

Crear serie a partir de un diccionario

s[n]

Extraer elemento con el índice implícito de la serie. Si los índices asignados son números entero, el índice implícito queda desactivado

s["name"]

Extraer elemento con el nombre del índice

s.nombre

Extraer elemento con el nombre del índice

s.dtype

Tipo de datos en la serie

s.index

Consultar índice de la serie

s.axes

Nos da acceso "a los ejes" (a los índices)

s.values

Consultar valores de la serie

s.name

Consultar o cambiar nombre de la serie. Por defecto, está vacío

s.index.name

Consultar o cambiar nombre del índice de la serie. Por defecto, está vacío

Operaciones básicas con series (cont)

s.shape

nos devuelve el tamaño de la serie

s = pd.Series(d, index = list)

Crear serie a partir de un diccionario, pero especificando nosotros el índice. Si hay valores que no pertenezcan diccionario, se añaden con un valor NaN:

Aplicación de funciones estadísticas

data.mean()

.podemos calcular el valor medio de los datos

data.mean(level = "Year")

si especificamos el nivel al que queremos aplicarlo, el DataFrame se agrega según los valores de dicho nivel antes de realizar la operación

Selección de datos en dataframes

Ventas["B"]["feb"]

podemos utilizar la sintaxis de los diccionarios para seleccionar la columna "B":. Esto significa que podemos realizar una selección en dicho resultado para, por ejemplo, extraer el valor correspondiente a febrero:

Ventas["B"] = [-1, -2, -3, -4, -5]

Si, una vez seleccionada una columna, le asignamos una lista o array (o serie) de valores de la misma longitud, estamos modificando dicha columna del dataframe:

Ventas["C"] = 0

Si asignamos un único valor escalar, este se propaga por toda la columna:

Selección de datos en dataframes (cont)

Ventas["D"] = pd.Series(list, index = list)

Si asignamos a una columna una serie pandas se consideran los índices del dataframe y de la serie, haciendo coincidir los valores cuyos índices sean los mismos en ambas estructuras (si dicha columna no existe, se crea). En el caso de que haya valores en la serie con índices que no se encuentren en el dataframe, se descartan. Y en el caso de que haya índices en el dataframe que no se encuentren en la serie, se asigna un valor NaN.

del(Ventas["A"])

Borrar columna. Con la notación Ventas.A no es posible crear nuevas columnas ni eliminarlas

Ventas[2:4]

El uso de un rango numérico entre los corchetes realiza una selección de filas

Ventas["feb":"may"]

vemos en el resultado anterior que se devuelven las filas entre el primer valor del rango (incluido) y el último (sin incluir). aunque en este caso la selección incluye tanto la fila correspondiente a la primera etiqueta como la fila correspondiente a la segunda.

tips[["tip", "day"]]

Si situamos entre los corchetes una lista de etiquetas, estaremos seleccionando columnas en el orden en el que aparecen en la lista y con formato dataframe

df.get("name")

extrae la columna indicada devolviendo un valor alternativo (por defecto None) si dicha columna no existe:

By **julenx**
cheatography.com/julenx/

Published 13th November, 2022.
Last updated 13th November, 2022.

Cheatography

Python pandas Cheat Sheet

by julenx via cheatography.com/168626/cs/35356/

Selección de datos en dataframes (cont)

Ventas.loc["may"]

Seleccionar filas siempre por etiqueta. Acepta listas.

Ventas.loc[["feb"]]

Selecciona fila y la devuelve en formato dataframe. Si hay varias en la lista, se devuelve un dataframe con esa selección. Admite rangos.

Ventas.loc["may", "C"]

Extraer un único valor por fila y columna. Admite rangos como Ventas.loc[:, "A"]

Ventas.loc[["may", "ene"], "B"]

seleccionar la intersección de las filas e y c (en este orden) y la columna B

df.iloc[n]

Selección por posición de fila. Mismas funcionalidades que iloc.

Ventas.columns.get_loc("B")

Obtiene el índice de la columna con esa etiqueta.

Ventas.columns.get_indexer(["A", "C"])

Obtiene el índice de columnas con esas etiquetas. Devuelve un array

Ventas.index.get_loc("feb")

Obtiene el índice de la fila con esa etiqueta.

Ventas.index.get_indexer(["feb", "abr"])

Obtiene el índice de filas con esas etiquetas. Devuelve un array

Ventas.iloc[Ventas.index.get_loc("feb"), 2]

extraer del anterior dataframe el dato que ocupa la fila "feb" y la columna de índice 2

Selección de datos en dataframes (cont)

df.iloc[[5, 3], df.columns.get_indexer(["C", "A"])]

obtener de las filas 5 y 3 (en este orden) los valores correspondientes a las columnas C y A (en este orden)

df[df.col > n]

Selección con booleanos. También funcionaría Ventas.loc[df.col > n] o con Ventas.iloc[(df.col > n).values] (explicación en la sección de series)

df.sample(n, random_state = ..., axis = ..., frac=...)

Al igual que ocurre con las series, también los dataframes tienen un método que permite extraer elementos del mismo de forma aleatoria. Este método permite especificar el número de elementos a extraer (o el porcentaje respecto del total, parámetros n y frac, respectivamente), si la extracción se realiza con reemplazo o no (parámetro replace), los pesos a aplicar a los elementos para realizar una extracción aleatoria ponderada (parámetro weights) y una semilla para el generador de números aleatorios que asegure la reproducibilidad de la extracción (parámetro random_state). También es posible indicar el eje a lo largo del cual se desea realizar la extracción (por defecto se extraen filas, correspondiente al eje 0)

s = df.pop("col")

extrae y elimina una columna de un dataframe

Multi-índices por producto cartesiano de arrays

```
MultiIndex([(2018,    'Spain'),
            (2018, 'Portugal'),
            (2018,   'France'),
            (2019,    'Spain'),
            (2019, 'Portugal'),
            (2019,   'France')],
           names=['Year', 'Country'])
```

		Sales
Year	Country	
2018	Spain	18
	Portugal	20
	France	10
2019	Spain	15
	Portugal	12
	France	18

```
`index = pd.MultiIndex.from_product(´
[
[2018, 2019],
["Sp ain ", " Por tug al", " - Fra nce "]
],
names = ["Ye ar", " Cou ntr y"]
)

data = pd.Dat aFr ame (data = [18, 20, 10, 15, 12, 18], index = index, columns = ["Sa les "])
```

Multi-indexación

```
MultiIndex([(2018,    'Spain'),
            (2018, 'Portugal'),
            (2018,   'France'),
            (2019,    'Spain'),
            (2019, 'Portugal'),
            (2019,   'France')],
           names=['Year', 'Country'])
```

		Sales
Year	Country	
2018	Spain	18
	Portugal	20
	France	10
2019	Spain	15
	Portugal	12
	France	18

```
index = pd.Mul tiI nde x.f rom - _ar rays(
[
 [2018, 2018, 2018, 2019, 2019,
```

```
2019],
 ["Sp ain ", " UK", " Fra -
nce ", " Spa in", " UK", " Fra -
nce "]
],
names = ["Ye ar", " Cou ntr y"]
)

data = pd.Dat aFrame(
 data = [18, 20, 10, 15, 12,
18],
 index = index, columns = ["Sa -
les "])
```

Operaciones entre dataframes y series

df + s

> Se suman los valores de una serie a todas las columnas (en todas las filas) cuyos valores coindicen con las etiquetas de la serie. Si la serie tiene índices cuyas columnas no existen en el df, se añaden con el valor NaN en todas las filas.

df.add(s, axis = 0)

Aplicación de funciones y mapeado

s.apply(fun)

> permite aplicar a cada uno de los elementos de la serie una función. Ésta deberá aceptar un único valor como argumento y devolver también un único valor

s.map(obj)

> Cambia los valores de la serie por los indicados en la función. Puede ser un diccionario u otra serie, en cuyo caso se cambian por los valores de la otra serie siguiendo buscand en el índice los valores que coinciden. También acepta funciones.

df.apply(fun)

> Aplica una función al df (por columnas eje 0 por defecto. Devuelve una serie donde en el caso por defecto los índices son los nombres de las columnas.

df,applymap(fun)

> aplica una función que acepta y devuelve un único escalar. Va celda por celda y devuelve otro df.

Agrupaciones

s.groupby(by = fun).mean()

> A groupby operation involves some combination of splitting the object, applying a function, and combining the results. This can be used to group large amounts of data and compute operations on these groups

s.groupby(by = d).mean()

s.groupby(level = 0).mean()

df.groupby(by = "col").mean()

df.groupby(by = ["Categoría", "Producto"]).mean()

ventas.groupby(level = 0).mean()

> Según las etiquetas del índice.

df.pivot_table(index = "foo", columns = "bar", values = "baz")

> Take a dataframe and create a new one where the index is one of the columns, columns are the values of other column and the values are taken from one more different column. Los valores que toma la variable incluida en el parámetro values van a la intersección de filas y columnas, aplicándoseles una cierta función de agregación que, por defecto, es np.mean (cálculo del valor medio).

aggfunc = "count"

> En vez de agrega los valores, agrega el número de registros presentado en cada intersección.

df.pivot_table(index = "foo", columns = "bar", values = "baz", aggfunc = ["mean", "count"])

> Es posible aplicar más de una función de agregación a los datos. Se devuelve un df con las columnas repetidas según cada parámetro.

Métodos de agregación estadística

df.describe()

> información estadística sobre los valores contenidos

df.mean()

> Devuelve la media aritmética de los valores del dataframe a lo largo de un determinado eje (eje 0 -vertical- por defecto)

pandas.DataFrame.median

pandas.DataFrame.mode

pandas.DataFrame.std

> Devuelve la desviación estándar de los valores del dataframe a lo largo de un determinado eje

pandas.DataFrame.var

df.pct_change()

> devuelve el porcentaje de cambio de una fila con respecto a la anterior (también puede aplicarse a columnas usando el parámetro axis). Podemos ver que los valores de la primera fila, al no existir una anterior con respecto a la que realizar el cálculo, reciben un valor NaN por defecto. En todo caso, es posible regular el comportamiento del método al respecto de los valores NaN con el parámetro fill_method.

df.nunique()

> devuelve el número de valores diferentes a lo largo del eje indicado

Ordenación y clasificación

s.sort_index()

Ordenación de series por índice. De forma descendente con ascending = False. Si los índices fuesen cadenas de texto, se ordenarían de la "a" a la "z", dando a las mayúsculas mayor prioridad.

df.sort_index()

Se puede hacer por filas o columnas con el parámetro axis. Acepta el parámetro ascending

df.sort_index().sort_index(axis = 1)

Ordenación a lo largo de ambos ejes

s.sort_values()

Ordenación de series por valor. De forma descendente con ascending = False.

df.sort_values(by = "A")

Supongamos que queremos ordenar esta estructura según la columna A. El eje por defecto son las columnas (axis=0).

df.sort_values(by = ["A", "C"])

En el caso de que dos filas tengan el mismo valor durante la ordenación, se recurre al valor de la segunda columna indicada.

s.rank()

Devuelve una serie conteniendo la posición de cada valor de la serie original si fuesen ordenados de menor a mayor. En el caso de valores repetidos, se calcula la posición media. Con el método method = "min" se les adjudica la posición más baja.

df.rank()

Por defecto en el eje vertical (0)

Selección con multi-índices

data.index.get_level_values(n)

Trabajando con un DataFrame o una Serie pandas con multi-índice, es posible extraer los valores de un nivel del índice (columna). El parámetro que deberemos pasar a este método será o el número del nivel o su nombre

data.loc[2018]

data.loc[(2018, "Spain")]

Es posible extraer los valores de un índice que está dentro de otro índice.

Multi-índices a partir de un array de tuplas

```
MultiIndex([(2018,    'Spain'),
            (2018, 'Portugal'),
            (2018,   'France'),
            (2019,    'Spain'),
            (2019, 'Portugal'),
            (2019,   'France')],
           names=['Year', 'Country'])
```

		Sales
Year	Country	
2018	Spain	18
	Portugal	20
	France	10
2019	Spain	15
	Portugal	12
	France	18

```
index = pd.Mul tiI nde x.f rom -
_tu ples(
[
(2018, " Spa in"),
(2018, " Por tug al"),
(2018, " Fra nce "),
(2019, " Spa in"),
(2019, " Por tug al"),
(2019, " Fra nce ")
],
names = ["Ye ar", " Cou ntr y"])

data = pd.Dat aFr ame (data =
[18, 20, 10, 15, 12, 18], index
= index, columns = ["Sa les "])
```

Unión de dataframes

pd.concat([df1, df2])

Unión básica de dataframes. Se pone uno encima del otro, añadiendo todos los índices aunque se repitan. En las columnas se añaden primero las columnas del primer df y luego las del segundo que no se hayan repetido, y si un dataframe no tiene valores para esa columna se añaden valores NaN.

pd.concat([df1, df2], axis = 1)

Se pone un dataframe al lado del otro. Se añaden las columnas comunes y no comunes aunque se repita el nombre. En el eje 0 se ponen primero los índices del primer df y luego los índices del segundo que no se hayan repetido.

join = "inner"

Parámetro de concat para que solo se unan con las etiquetas comunes. El parámetro por defecto es "outer"

ignore_index = True

Se elimina el nombre de las filas para axis=0 en el parámetro concat o el nombre de las columnas para axis=1 y se introduce un índice nuevo empezando por 0. Por defecto, el valor es False

pd.merge(df1, df2)

Unión de dfs ignorando el índice. Se buscan las columnas en común, y después se colocan los valores en de cada fila de ambos dfs cuyo valor en esa columna coincida.

Unión de dataframes (cont)

how="outer"

Parámetro de merge. La función coge todos los valores de las columnas en común, no solo los valores de la columna en común que estén presentes en ambos dfs.

on="column"

Especificar la columna en común. Si hay más de una columna en común y solo se especifica una, se crearán varias copias mostrando todas las posibles combinaciones.

left_on = "Month", right_on = "MonthName"

Especificar qué columna debería ser común cuando tengan diferentes nombres

left_on = "Month", right_index = True

Especificar que en vez de una columna común se tiene un índice

Unión de series

t = pd.concat([s, r])

Unión básica de series. Si las etiquetas coinciden se repiten para cada valor.

pd.concat([a, b], axis = 1)

Unión de series en forma de dataframe. La primera columna será la primera seria, la segunda columna será la segunda serie y el índice de filas serán las etiquetas compartidas y no compartidas. Valores NaN para laos valors de las series cuyas etiquetas no existen en la otra. Admite el parámetro sort.

Edición de series

s[0] = -1

Podemos modificar un valor de una serie usando la notación corchetes, y haciendo referencia a índices o a etiquetas:

s["b"] = -2

Podemos modificar un valor de una serie usando la notación corchetes, y haciendo referencia a índices o a etiquetas:

s[1:3] = 0

s["b":"d"] = -10

s["b":"d"] = [10, 11, 12]

s["f"] = 0

si se trata de una etiqueta (y no existe) se añade:

s["d":"h"] = 0

Si el rango incluye valores que no existen, se ignoran

s[["c", "a"]] = [-1, -2]

Podemos incluir como argumento del operador selección una lista de etiquetas, en cuyo caso los valores se asignan en el orden indicado

s[[1, 0]] = [20, 21]

También podemos usar índices. Pero si coinciden los índices y las etiquetas, éstas tienen preferencia en este tipo de selección, a menos que usemos los métodos loc e iloc

r = s.drop("b")

devuelve una copia de la serie tras eliminar el elemento cuya etiqueta se especifica. Admite listas. El argumento inplace = True realiza la eliminación "-inplace" (modificando directamente la serie).

Edición de series (cont)

s.drop(s.index[[1, 3]])

Uso de drop con índices en vez de etiquetas.

s.pop("b")

devuelve el valor correspondiente a dicha etiqueta, eliminándolo de la serie in-place

s.where(condition, iftrue, iffalse)

permite filtrar los valores de una serie de forma que solo los que cumplan cierta condición se mantengan. Los valores que no la cumplan son sustituidos por un valor (NaN por defecto, u otro valor si se especifica):

Operaciones básicas con dataframes

df = pd.DataFrame(d, index = list, columns=list)

Crear dataframe a partir de un diccionario o un array matriz. Si el diccionario contiene listas, se crean varias filas. Lo mismo ocurre si en vez de un diccionario se usa una lista de diccionarios que tengan las mismas claves. Las claves del diccionario son los nombres de las columnas, a excepción de que se indique algo distinto en el parámetro columns, en cuyo caso deberá coincidir con las claves del diccionario o no se mostrarán los valores.

pandas.DataFrame.from_dict

crea un dataframe a partir de un diccionario de diccionarios o de secuencias tipo array

pandas.DataFrame.from_records

parte de una lista de tuplas o de arrays NumPy con un tipo estructurado

df["col"]

Consultar una columna de dataframe

df.col

Consultar una columna de dataframe

Python pandas Cheat Sheet

by julenx via cheatography.com/168626/cs/35356/

Operaciones básicas con dataframes (cont)

df.dtypes

Consultar tipos de datos en las columnas de un dataframe

df.index

Índice de filas de un dataframe

df.columns

Lista de nombres de columnas del dataframe

df.axes

Ver ejes del dataframe (filas y columnas)

df.index.name

Ver/cambiar el nombre del eje x (conjunto de filas)

df.columns.name

Ver/cambiar el nombre del eje y (conjunto de columnas)

df.values

Ver valores del df en forma de array

df.shape

Dimensiones del df

Lectura y escritura de ficheros

df = pd.read_csv("file.csv")

Lectura de fichero

df.to_csv('out.zip', index=False)

Guardar dataframe o serie

Operaciones con dataframes

df1.add(df2, fill_value = 0)

establecar un valor predeterminado para aquellos valores que no se encuentren en uno de los dataframes.

pandas.DataFrame.add

pandas.DataFrame.sub

pandas.DataFrame.mul

Operaciones con dataframes (cont)

pandas.DataFrame.div

pandas.DataFrame.mod

calcular el módulo de un dataframe y otro dataframe, elemento por elemento

pandas.DataFrame.dot

multiplicación de las dos matrices representadas por los dos dataframes

pandas.DataFrame.abs

copia del dataframe conteniendo el valor absoluto de cada uno de sus valores

Operaciones con series

r.add(s, fill_value = 0)

Sumar series por etiquetas, añadiendo un valor por defecto cuando una serie no tiene las etiquetas de la otra.

pandas.Series.sub

pandas.Series.mul

pandas.Series.div

pandas.Series.round

Reindexación de series

s.reindex(list)

copia reindexada de una serie. El primer argumento siempre es el nuevo índice. Si el nuevo índice es un subconjunto del original, la serie generada no contendrá todos los valores de la serie de la que partimos. si en el nuevo índice se incluyen etiquetas no incluidas en el índice original, la nueva serie incluirá dicha etiqueta pero el valor asignado a ella recibe el valor por defecto NaN. es personalizable usando el parámetro fill_value.

Reindexación de series (cont)

method = "ffill"

los valores existentes rellenan los valores inexistentes que los sigan. Se rellenan los valores inexistentes con el primer valor existente que los precedan. Por orden alfabético.

method = "bfill"

method = "nearest"

asigna a cada valor desconocido el valor más próximo en la serie original. Para ver esta opción en funcionamiento necesitamos partir de una serie cuyo índice sea numérico. Si en reindex un índice es 19 y en la serie original había un índice 20, se usa el valor de ese.

Reindexación de dataframes

df.reindex(list)

Reindexa por filas.

df.reindex(index = list)

Mismo resultado

df.reindex(columns = list)

Reindexa por columnas

df.reindex(index = list1, columns = list2)

Reindexa por filas y columnas

df.set_index("col")

fija una columna del dataframe como índice, descartando el índice existente. Con el parámetro drop = False se mantiene la columna.

Con reindex se asignan valores NaN a los valores de filas o columnas que no existan, lo que se puede cambiar con el parámetro fill_value, o los mismos valores que en las series del parámetro method

Edición de dataframes

df.iloc[1, 2] = -100

Podemos modificar un valor concreto usando los métodos loc o iloc, en función de que queramos usar sus etiquetas o índices.

df["C"] = [-100, -200, -300, -400, -500, -600]

Podemos modificar una columna completa seleccionándola y asignándole, por ejemplo, una lista con los nuevos valores

df = Ventas.copy() df.loc["feb":"mar", "-A":"B"] = [[-100, -200], [-300, -400]]

Si la selección es un bloque de datos de un tamaño arbitrario, nos encontramos en el mismo escenario: o bien insertamos datos con el mismo tamaño que la selección, o insertamos un único valor que se propagará a toda la selección.

df = Ventas.copy() df.loc["feb":"mar", "-A":"B"] = -1

df.where(condition, iftrue, iffalse)

filtra los valores contenidos en el dataframe de forma que solo los que cumplan cierta condición se mantengan. El resto de valores son sustituidos por un valor que, por defecto, es NaN, o por otro valor si se especifica.

Ventas.drop(["feb", "abr"], axis = 0)

elimina las filas o columnas indicadas y devuelve el resultado. lo que se muestra es el resultado de eliminar las filas indicadas del dataframe. Éste no se modifica salvo que utilicemos el argumento inplace = True.

Ventas.drop(index = ["feb", "abr"])

Admite índices

Edición de dataframes (cont)

Ventas.drop(["A", "C"], axis = 1)

Para eliminar columnas, habría que indicar el eje correspondiente o usar el parámetro columns

Ventas.drop(columns = ["A", "C"])

Multi-índices a partir de un DataFrame

	Year	Country
0	2018	Spain
1	2018	Portugal
2	2018	France
3	2019	Spain
4	2019	Portugal
5	2019	France

```
MultiIndex([(2018,    'Spain'),
            (2018, 'Portugal'),
            (2018,   'France'),
            (2019,    'Spain'),
            (2019, 'Portugal'),
            (2019,   'France')],
           names=['Year', 'Country'])
```

Year	Country	
2018	Spain	18
	Portugal	20
	France	10
2019	Spain	15
	Portugal	12
	France	18

```
df = pd.Dat aFr ame({
" Yea r":[ 2018, 2018, 2018,
2019, 2019, 2019],
" Cou ntr y": ["Sp ain ", " -
Por tug al", " Fra nce ", " -
Spa in", " Por tug al", " Fra -
nce "]
})

index = pd.Mul tiI nde x.f rom -
_fr ame(df)

data = pd.Dat aFr ame (data =
[18, 20, 10, 15, 12, 18], index
= index, columns = ["Sa les "])
```

Selección de datos en series

s.get(n)

devuelve el valor que ocupa el índice indicado, y devuelve un valor nulo en caso de que no exista.

Si utilizamos el índice númerico implícito como rango (s[1:3]), se seleccionan los valores desde el primer índice incluido hasta el último sin incluir. Si se utilizan los índices explícitos, se incluyen los valores desde el primer hasta el último índice incluyendo ambos.

Sin embargo, si al utilizar un índice explícito numérico hacemos referencia a los datos con un rango, se sigue cogiendo desde el primer valor incluido hasta el último sin incluir.

s.loc["name"]

seleccionar un grupo de elementos por etiquetas y no por índice implícito. Acepta listas y rangos (devuelve todos los elementos entre los límites indicados, ambos incluidos)

s.iloc[n]

Extrae datos de la serie siempre a partir de los índices implícitos que éstos tienen asignados. si el rango tiene la forma a:b, se incluyen todos los elementos desde aquel cuyo índice es a (incluido) hasta el que tiene el índice b (sin incluir).

s[s > 2]

Ejemplo para extraer valores de una serie con valores booleanos.

s.loc[s > 2]

Mismo comportamiento que el método anterior,

Selección de datos en series (cont)

s.iloc[(s > 2).values]

Mismo comportamiento que el método anterior. Con iloc debe ser así puesto que puede aceptar una lista de valores booleanos on un array NumPy, pero no una serie de pandas que es lo que genera este método.

s.pop(i)

extrae y elimina un elemento de una serie cuyo índice se indica como argumento. Si la serie tiene un índice explícito, el argumento de pop hará referencia a este índice.

Inspección de series y dataframes

df.head()

devuelve los primeros elementos de la estructura. Por defecto, se trata de los 5 primeros elementos, pero podemos especificar el número que deseamos como argumento de la función.

df.tail()

muestran los últimos elementos de la estructura. Si no indicamos otra cosa como argumento, serán los 5 últimos elementos los que se muestren

Inspección de series y dataframes (cont)

s.sample(frac = 0.6, random_state = 18)

ver datos aleatorios de nuestra estructura. el número de elementos devueltos por defecto es uno. permite especificar o bien el número de elementos a extraer o bien la fracción del número total de elementos a extraer (parámetros x y frac, respectivamente), pudiendo especificar si la extracción se realiza con reemplazo o no (parámetro replace), los pesos a aplicar a cada elemento para realizar una extracción aleatoria ponderada (parámetro weights), y una semilla para el generador de números aleatorios que asegure la reproducibilidad de la extracción (parámetro random_state)

df.describe()

devuelve información estadística de los datos del dataframe o de la serie.
acepta el parámetro percentiles conteniendo una lista (o semejante) de los percentiles a mostrar. También acepta los parámetros include y exclude para especificar los tipos de las características a incluir o excluir del resultado.

df.info()

muestra un resumen de un dataframe, no de una serie. Incluyendo información sobre el tipo de los índices de filas y columnas, los valores no nulos y la memoria usada:

Inspección de series y dataframes (cont)

s.value_counts(dropna = False)

devuelve una estructura conteniendo los valores presentes en la serie y el número de ocurrencias de cada uno, Si se trata de una serie numérica, en lugar de devolver los valores distintos y el número de ocurrencias, este método también puede agrupar los datos en "-bins" y devolver una lista de bins (indicando sus márgenes) con el número de valores en cada uno de ellos (bins = 5)

Gestión de valores nulos

pd.isnull(s)

devuelve una estructura con las mismas dimensiones que la que se cede como argumento sustituyendo cada valor por el booleano True si el correspondiente elemento es un valor nulo, y por el booleano False en caso contrario

s.isnull()

pd.isnull(df)

df.isnull()

s.dropna()

filtra los valores para dejar solo aquellos no nulos

df.dropna()

Se aplica por defecto al eje 0, y borra las filas que tengan un valor nulo en cualquiera de las columnas. Con how = "all" se borran solo las filas que tengan todos los valores nulos. Esto por defecto es how="any"

Gestión de valores nulos (cont)

s.fillna(n)

> permite sustituir los valores nulos de una estructura pandas por otro valor según ciertos criterios. Acepta los parámetros method = "ffill" y method="bfill"

df.fillna(n)

> Si usamos los métodos indicados arriba también podemos cambiar el eje.

df.fillna(axis = 1, method = "bfill").fillna(n)

> asegurarnos de que todos los elementos han sido sustituidos adecuadamente

By **julenx**
cheatography.com/julenx/

Published 13th November, 2022.
Last updated 13th November, 2022.