

Obtención y Preparación de Datos

Clase 04

Breve Introducción a Machine Learning

Dr. Ramón Caraballo
SECIU Red Académica Uruguaya
UDELAR



UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Objetivos

- Importar datos desde archivos de texto, excel, json, etc
- Limpieza básica de Datos, reemplazo, imputacion de datos etc.
- Análisis Exploratorio de Datos
- Estadística Básica

Importando Datos con Numpy

texto plano (CSV, TSV, etc)

```
In [1]: import numpy as np
```

```
In [2]: filename = 'MNIST_header.txt'
```

```
In [3]: data = np.loadtxt(filename, delimiter=',',  
skiprows=1)
```

```
In [4]: print(data)  
[[ 0.  0.  0.  0.  0.]  
[ 86. 250. 254. 254. 254.]  
[ 0.  0.  0.  9.  254.]  
...,  
[ 0.  0.  0.  0.  0.]  
[ 0.  0.  0.  0.  0.]  
[ 0.  0.  0.  0.  0.]]
```

```
numpy.loadtxt(fname, dtype=<class 'float'>, comments='#',  
delimiter=None, converters=None, skiprows=0, usecols=None,  
unpack=False, ndmin=0, encoding=None, max_rows=None, *,  
quotechar=None, like=None)
```

<https://numpy.org/doc/stable/user/index.html>

Los datos deben ser del mismo tipo!!

Importando Datos con Pandas

Datos de diferente tipo

```
In [1]: import pandas as pd
```

```
In [2]: filename = 'winequality-red.csv'
```

```
In [3]: data = pd.read_csv(filename)      https://pandas.pydata.org/docs/reference/api/pandas.read\_csv.html
```

```
In [4]: data.head()
```

```
Out[4]:
```

	volatile acidity	citric acid	residual sugar
0	0.70	0.00	1.9
1	0.88	0.00	2.6
2	0.76	0.04	2.3
3	0.28	0.56	1.9
4	0.70	0.00	1.9

```
In [5]: data_array = data.values
```

Lo que necesita un científico de datos

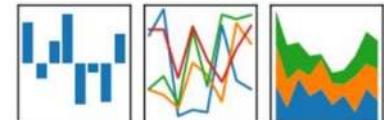
- ❖ Estructuras de datos bidimensionales etiquetadas
- ❖ Columnas de tipos potencialmente diferentes
- ❖ Manipular, segmentar, remodelar, agrupar, unir, fusionar
- ❖ Realizar estadísticas
- ❖ Trabajar con datos de series temporales
- ❖ Importar archivos planos de forma sencilla
- ❖ Importar archivos planos con problemas como comentarios y valores faltantes



Wes McKinney

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Python DataFrame = R Dataframe!!

- Estándares y mejores prácticas para usar pandas



Hadley Wickham
@hadleywickham



Following

A matrix has rows and columns. A data frame has observations and variables. #rstats #tidydata

RETWEETS
128

LIKES
233



Otros tipos de Archivos

- Hojas de cálculo de Excel
- Archivos MATLAB
- Archivos SAS
- Archivos Stata
- Archivos HDF5
- Bases de Datos NoSQL
- Páginas web

Importing Excel spreadsheets

```
In [1]: import pandas as pd
```

```
In [2]: file = 'urbanpop.xlsx'
```

```
In [3]: data = pd.ExcelFile(file)
```

```
In [4]: print(data.sheet_names)
['1960-1966', '1967-1974', '1975-2011']
```

```
In [5]: df1 = data.parse('1960-1966')
```

← sheet name, as a string

```
In [6]: df2 = data.parse(0)
```

← sheet index, as a float

Leyendo archivos JSON

- **JSON = JavaScript Object Notation.**
- Se originó a partir de JavaScript pero ha trascendido sus orígenes para convertirse en un idioma independiente y ahora se reconoce como el estándar para el intercambio de datos.
- La popularidad de JSON se puede atribuir a la compatibilidad nativa del lenguaje JavaScript, lo que da como resultado un excelente rendimiento de análisis en los navegadores web.
- Además de eso, la sintaxis sencilla de JSON permite que tanto humanos como computadoras lean y escriban datos JSON sin esfuerzo.

Leyendo archivos JSON

JSON

hello_frieda.json

```
1 {  
2     "name": "Frieda",  
3     "isDog": true,  
4     "hobbies": ["eating", "sleeping", "barking"],  
5     "age": 8,  
6     "address": {  
7         "work": null,  
8         "home": ["Berlin", "Germany"]  
9     },  
10    "friends": [  
11        {  
12            "name": "Philipp",  
13            "hobbies": ["eating", "sleeping", "reading"]  
14        },  
15        {  
16            "name": "Mitch",  
17            "hobbies": ["running", "snacking"]  
18        }  
19    ]  
20 }
```

- Un par clave-valor en un objeto JSON está separado por dos puntos (:).
- En el lado izquierdo de los dos puntos, se define una clave. Una clave es una cadena que debe encerrarse entre comillas dobles (").
- **A diferencia de Python, los strings en JSON no admiten comillas simples (').**

Leyendo archivos JSON

`json.loads(...)` Analiza una cadena de datos JSON. Loads = abreviatura de load as string.

Convierte: Objetos a diccionarios, matrices a listas, booleanos, enteros, pto flotantes y los strings se reconocen por lo que son y se convertirán en los tipos correctos en Python .Cualquier valor nulo se convertirá en el tipo *None* de Python

```
>>> import json

>>> jsonstring = '{"name": "erik", "age": 38, "married": true}'
>>> person = json.loads(jsonstring)
>>> print(person['name'], 'is', person['age'], 'years old')
erik is 38 years old
>>> print(person)
{'name': 'erik', 'age': 38, 'married': True}
```

The output might look like a string, but it's actually a dictionary that you can use in your code as explained on our page about [Python dictionaries](#). You can check for yourself:

```
>>> type(person)
<class 'dict'>
```

JSON OBJECT	PYTHON OBJECT
object	dict
array	list
string	str
null	None
number (int)	int
number (real)	float
true	True
false	False

Creando archivos JSON

json.dumps() Codifica datos a formato JSON en Python.

Utilice json.dumps (abreviatura de 'dump to string') para convertir un objeto de Python que consta de diccionarios, listas y otros tipos nativos en una cadena:

```
import json

person = {'name': 'erik', 'age': 38, 'married': True}
json_string = json.dumps(person)
print(json_string)

# To make sure, let's print the type too
print(type(json_string))
```



Leyendo archivos JSON

json.dump() & json.load() Permiten Codificar y Decodificar datos JSON en Python.
Cargará datos o escribirá un archivo, pero debe abrirse el archivo primero

```
data = {'name': 'Eric', 'age': 38 }

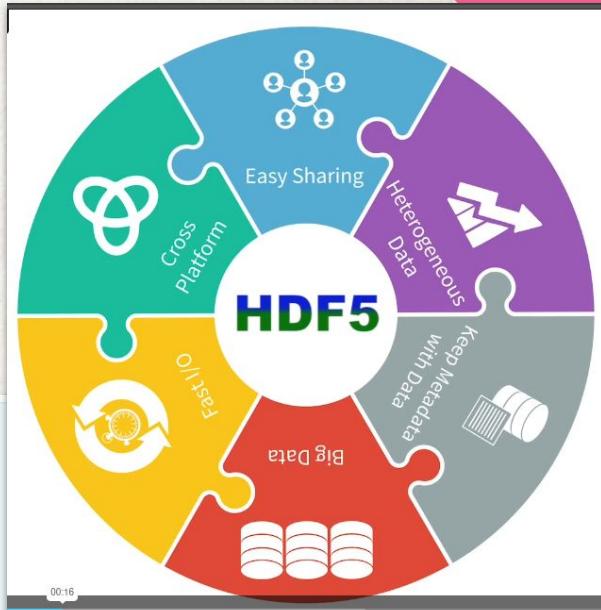
with open('data.json', 'w') as json_file:
    json.dump(data, json_file)
```

```
with open('data.json') as json_file:
    data = json.load(json_file)
    ...
```

Archivos HDF5

- Formato de datos jerárquicos versión 5
- Estándar para almacenar grandes cantidades de datos numéricos
- Los conjuntos de datos pueden tener cientos de gigabytes o terabytes
- HDF5 puede escalar a exabytes

```
In [1]: import h5py  
  
In [2]: filename = 'H-H1_LOSC_4_V1-815411200-4096.hdf5'  
  
In [3]: data = h5py.File(filename, 'r') # 'r' is to read  
  
In [4]: print(type(data))  
<class 'h5py._hl.files.File'>
```



Bases de Datos Relacionales

Modelo relacional de “Ted” Codd

- Orders table

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName	ShipAddress
10248	VINET	5	7/4/1996 12:00:00 AM	8/1/1996 12:00:00 AM	7/16/1996 12:00:00 AM	3	32.38	Vins et alcools Chevalier	59 rue de l'Abbaye
10251	VICTE	3	7/8/1996 12:00:00 AM	8/5/1996 12:00:00 AM	7/15/1996 12:00:00 AM	1	41.34	Victuailles en stock	2, rue du Commerce
10254	CHOPS	5	7/11/1996 12:00:00 AM	8/8/1996 12:00:00 AM	7/23/1996 12:00:00 AM	2	22.98	Chop-suey Chinese	Hauptstr. 31

Conjunto de tablas interrelacionadas

- Customers table

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	None	12209	Germany
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	None	WA1 1DP	UK
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	None	68306	Germany
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	None	13008	France

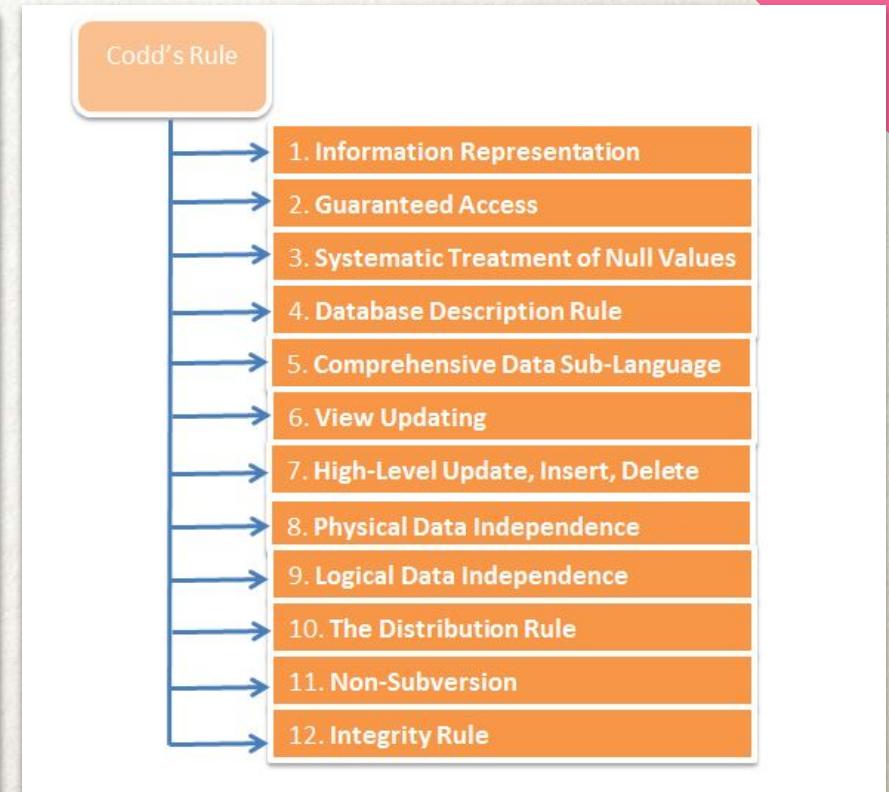
- Employees table

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region
1	Davolio	Nancy	Sales Representative	Ms.	12/8/1948 12:00:00 AM	5/1/1992 12:00:00 AM	507 - 20th Ave. E.\nApt. 2A	Seattle	WA
2	Fuller	Andrew	Vice President, Sales	Dr.	2/19/1952 12:00:00 AM	8/14/1992 12:00:00 AM	908 W. Capital Way	Tacoma	WA
3	Leverling	Janet	Sales Representative	Ms.	8/30/1963 12:00:00 AM	4/1/1992 12:00:00 AM	722 Moss Bay Blvd.	Kirkland	WA

Modelo Relacional

- ❖ Ampliamente aceptado
- ❖ Las 13 Reglas de Codd
- ❖ Describe lo que es una Gestión de Bases de Datos Relacionales
- ❖ Reglas que el sistema debe cumplir para ser considerado relacional
- ❖ 5 Formas Normales

https://en.wikipedia.org/wiki/Codd%27s_12_rules



Sistemas de Bases de Datos

- PostgreSQL
- MySQL
- SQLite
- SQL = Structured Query Language



Python: SQLAlchemy

```
In [1]: from sqlalchemy import create_engine
```

```
In [2]: engine = create_engine('sqlite:///Northwind.sqlite')
```

```
In [3]: table_names = engine.table_names()
```

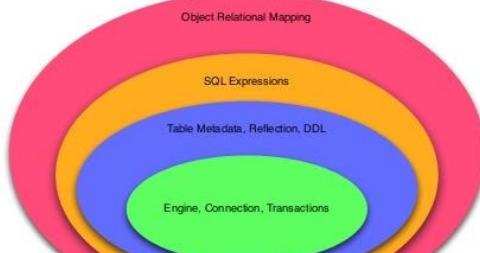
```
In [4]: print(table_names)
['Categories', 'Customers', 'EmployeeTerritories',
'Employees', 'Order Details', 'Orders', 'Products',
'Region', 'Shippers', 'Suppliers', 'Territories']
```

Flujo de trabajo de consultas SQL

1. Importar paquetes y funciones
2. Crear el motor de base de datos
3. Conéctese al motor
4. Consultar la base de datos
5. Guardar los resultados de la consulta en un DataFrame
6. Cierre la conexión

SQLAlchemy is like an Onion

Can be learned from the inside out, or outside in



Consultando una base de datos

Usando el manejador de contexto

```
In [1]: from sqlalchemy import create_engine
```

```
In [2]: import pandas as pd
```

```
In [3]: engine = create_engine('sqlite:///Northwind.sqlite')
```

```
In [4]: with engine.connect() as con:
```

```
....:         rs = con.execute("SELECT OrderID, OrderDate,  
ShipName FROM Orders")
```

```
....:         df = pd.DataFrame(rs.fetchmany(size=5))
```

```
....:         df.columns = rs.keys()
```

nro de filas que quiero

Resultados

```
In [8]: print(df.head())
```

	0	1	2	3	4
0	10248	VINET	5	7/4/1996 12:00:00 AM	8/1/1996 12:00:00 AM
1	10251	VICTE	3	7/8/1996 12:00:00 AM	8/5/1996 12:00:00 AM
2	10254	CHOPS	5	7/11/1996 12:00:00 AM	8/8/1996 12:00:00 AM
3	10256	WELLI	3	7/15/1996 12:00:00 AM	8/12/1996 12:00:00 AM
4	10258	ERNSH	1	7/17/1996 12:00:00 AM	8/14/1996 12:00:00 AM

```
In [5]: rs = con.execute("SELECT * FROM Orders")
```

```
In [6]: df = pd.DataFrame(rs.fetchall())
```

```
In [7]: con.close()
```

```
In [9]: print(df.head())
```

	OrderID	CustomerID	EmployeeID	OrderDate
0	10248	VINET	5	7/4/1996 12:00:00 AM
1	10251	VICTE	3	7/8/1996 12:00:00 AM
2	10254	CHOPS	5	7/11/1996 12:00:00 AM
3	10256	WELLI	3	7/15/1996 12:00:00 AM
4	10258	ERNSH	1	7/17/1996 12:00:00 AM

```
In [6]: df = pd.DataFrame(rs.fetchall())
```

```
In [7]: df.columns = rs.keys()
```

Usando Pandas en la consulta

usando `pandas_read_sql()`

```
In [1]: from sqlalchemy import create_engine  
  
In [2]: import pandas as pd  
  
In [3]: engine = create_engine('sqlite:///Northwind.sqlite')  
  
In [4]: with engine.connect() as con:  
....:     rs = con.execute("SELECT * FROM Orders")  
....:     df = pd.DataFrame(rs.fetchall())  
....:     df.columns = rs.keys()  
  
In [5]: df = pd.read_sql_query("SELECT * FROM Orders", engine)
```

Consultas Avanzadas

Las tablas estan ligadas por relaciones

- Orders table

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName	ShipAddress
10248	VINET	5	7/4/1996 12:00:00 AM	8/1/1996 12:00:00 AM	7/16/1996 12:00:00 AM	3	32.38	Vins et alcools Chevalier	59 rue de l'Abbaye
10251	VICTE	3	7/8/1996 12:00:00 AM	8/5/1996 12:00:00 AM	7/15/1996 12:00:00 AM	1	41.34	Victuailles en stock	2, rue du Commerce
10254	CHOPS	5	7/11/1996 12:00:00 AM	8/8/1996 12:00:00 AM	7/23/1996 12:00:00 AM	2	22.98	Chop-suey Chinese	Hauptstr. 31

- Customers table

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	None	12209	Germany
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	None	WA1 1DP	UK
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim	None	68306	Germany
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille	None	13008	France

- Employees table

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address	City	Region
1	Davolio	Nancy	Sales Representative	Ms.	12/8/1948 12:00:00 AM	5/1/1992 12:00:00 AM	507 - 20th Ave. E.\nApt. 2A	Seattle	WA
2	Fuller	Andrew	Vice President, Sales	Dr.	2/19/1952 12:00:00 AM	8/14/1992 12:00:00 AM	908 W. Capital Way	Tacoma	WA
3	Leverling	Janet	Sales Representative	Ms.	8/30/1963 12:00:00 AM	4/1/1992 12:00:00 AM	722 Moss Bay Blvd.	Kirkland	WA

Consultas Avanzadas

Cruzando tablas: uso de JOIN

- Orders table

OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	Status
10248	VINET	5	7/4/1996 12:00:00 AM	8/1/1996 12:00:00 AM	7/16/1996 12:00:00 AM	Shipped
10251	VICTE	3	7/8/1996 12:00:00 AM	8/5/1996 12:00:00 AM	7/15/1996 12:00:00 AM	Shipped
10254	CHOPS	5	7/11/1996 12:00:00 AM	8/8/1996 12:00:00 AM	7/23/1996 12:00:00 AM	Shipped

- Customers table

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Country	PostalCode	Phone	Fax
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	Germany	12205	(030) 1234567	(030) 12345678
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	UK	EC1R 5NT	(010) 1234567	(010) 12345678
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Munich	Germany	80537	(089) 1234567	(089) 12345678
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Paris	France	75002	(01) 1234567	(01) 12345678

```
In [1]: from sqlalchemy import create_engine
```

```
In [2]: import pandas as pd
```

```
In [3]: engine = create_engine('sqlite:///Northwind.sqlite')
```

```
In [4]: df = pd.read_sql_query("SELECT OrderID, CompanyName FROM Orders INNER JOIN Customers on Orders.CustomerID = Customers.CustomerID", engine)
```

```
In [5]: print(df.head())
```

	OrderID	CompanyName
0	10248	Vins et alcools Chevalier
1	10251	Victuailles en stock
2	10254	Chop-suey Chinese
3	10256	Wellington Importadora
4	10258	Ernst Handel

Importando archivos de la Web

urllib vs requests

```
In [1]: from urllib.request import urlretrieve
```

```
In [2]: url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv'
```

```
In [3]: urlretrieve(url, 'winequality-white.csv')
Out[3]: ('winequality-white.csv', <http.client.HTTPMessage at 0x103cf1128>)
```

GET requests using urllib

```
In [1]: from urllib.request import urlopen, Request
```

```
In [2]: url = "https://www.wikipedia.org/"
```

```
In [3]: request = Request(url)
```

```
In [4]: response = urlopen(request)
```

```
In [5]: html = response.read()
```

```
In [6]: response.close()
```

Importando archivos de la Web

urllib

Para recuperar un recurso a través de una URL y almacenarlo en una ubicación temporal, puede hacerlo a través de `shutil.copyfileobj()` y `tempfile`. Funciones de `NamedTemporaryFile()`:

```
import shutil
import tempfile
import urllib.request

with urllib.request.urlopen('http://python.org/') as response:
    with tempfile.NamedTemporaryFile(delete=False) as tmp_file:
        shutil.copyfileobj(response, tmp_file)

with open(tmp_file.name) as html:
    pass
```

Python requests

requests package

GET requests using requests

- One of the most downloaded Python packages

```
In [1]: import requests
```

```
In [2]: url = "https://www.wikipedia.org/"
```

```
In [3]: r = requests.get(url)
```

```
In [4]: text = r.text
```

Acondicionamiento de Datos

- Puede consumir hasta el 90% del tiempo del proyecto
- Los datos casi nunca vienen limpios
- Como hacer Diagnosis de los Datasets



Problemas comunes

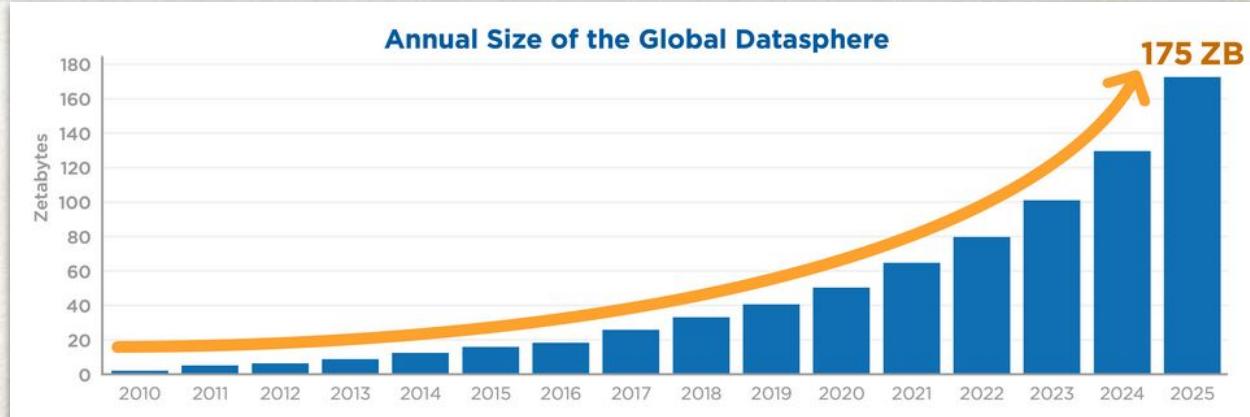
- Nombres de columnas inconsistentes
- Datos faltantes
- Valores atípicos
- Filas duplicadas
- Desordenado
- Necesidad de procesar columnas
- Los tipos de columnas pueden indicar valores de datos inesperados

Datos: Un Recurso Natural

Los datos como recurso natural Al igual que el petróleo, el gas y otros recursos naturales, la extracción de datos requiere habilidad y energía. De hecho, ya consume ~10% de la electricidad a nivel mundial. A medida que se afianza la frontera digital, necesitamos abordar mejor la forma en que tratamos los datos para mitigar el crecimiento de los llamados vertederos de datos.

Antonio Neri

President and Chief Executive Officer, Hewlett Packard Enterprise

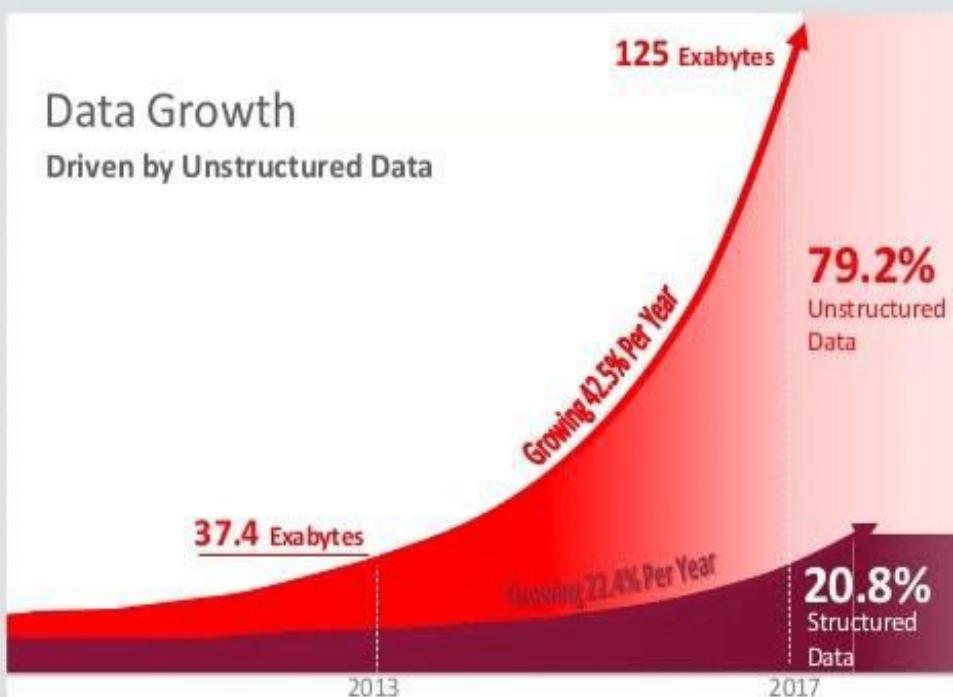


<https://www.weforum.org/stories/2020/03/we-should-treat-data-as-a-natural-resource-heres-why/>

Datos Estructurados vs No Estructurados

Data Growth

Driven by Unstructured Data



Structured Data

Can be displayed in rows, columns and relational databases

Numbers, dates and strings

Estimated 20% of enterprise data (Gartner)

Requires less storage

Easier to manage and protect with legacy solutions



Unstructured Data

Cannot be displayed in rows, columns and relational databases

Images, audio, video, word processing files, e-mails, spreadsheets

Estimated 80% of enterprise data (Gartner)

Requires more storage

More difficult to manage and protect with legacy solutions



Prácticas útiles para acondicionar Datos

- Análisis Exploratorio de Datos
- Reemplazar valores y/o cadenas de caracteres
- Cambio de tipo de datos
- Imputación de datos faltantes
- Pivoteo de tablas

Análisis Exploratorio de Datos

Con Pandas y Numpy

Importé mis datos: Y ahora qué?

Necesito saber como es el estado de estos datos:

- Están en un formato adecuado ?
- Los tipos de datos son correctos o hay que cambiarlos?
- Hay datos faltantes? Cuantos?, Qué hacemos con estos?
- Hay valores extremos/erróneos que debamos corregir?
- Qué hacer con los datos categóricos?
- Como puedo obtener una vista general de los datos?

Datos desordenados

Messy Data



	Continent	Country	female literacy	fertility	population
0	ASI	Chine	90.5	1.769	1.324655e+09
1	ASI	Inde	50.8	2.682	1.139965e+09
2	NAM	USA	99.0	2.077	3.040600e+08
3	ASI	Indonésie	88.8	2.132	2.273451e+08
4	LAT	Brésil	90.2	1.827	NaN

- ❖ Nombres de columna inconsistentes
- ❖ Datos Faltantes
- ❖ Nombres de país en Francés

```
In [1]: import pandas as pd  
In [2]: df = pd.read_csv('literacy_birth_rate.csv')
```

Archivo '*literacy_birth_rate.csv*' en el repo de GitHub dentro de /datasets bajarlo con `urllib` primero!

Inspección Visual

head() & tail()

In [3]: df.head()

Out[3]:

	Continent	Country	female literacy	fertility	population
0	ASI	Chine	90.5	1.769	1.324655e+09
1	ASI	Inde	50.8	2.682	1.139965e+09
2	NAM	USA	99.0	2.077	3.040600e+08
3	ASI	Indonésie	88.8	2.132	2.273451e+08
4	LAT	Brésil	90.2	1.827	NaN

In [4]: df.tail()

Out[4]:

	Continent	Country	female literacy	fertility	population
0	AF	Sao Tomé-et-Principe	90.5	1.769	1.324655e+09
1	LAT	Aruba	50.8	2.682	1.139965e+09
2	ASI	Tonga	99.0	2.077	3.040600e+08
3	OCE	Australia	88.8	2.132	2.273451e+08
4	OCE	Sweden	90.2	1.827	NaN

Inspección Visual

```
In [5]: df.columns
```

```
Out[5]: Index(['Continent', 'Country', 'female literacy',
'fertility', 'population'], dtype='object')
```

```
In [6]: df.shape
```

```
Out[6]: (164, 5)
```

```
In [7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 164 entries, 0 to 163
Data columns (total 5 columns):
Continent           164 non-null object
Country            164 non-null object
female literacy    164 non-null float64
fertility          164 non-null object
population         122 non-null float64
dtypes float64(2), object(3)
memory usage: 6.5+ KB
```

tipo de cada columna

```
In [1]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 164 entries, 0 to 163
Data columns (total 5 columns):
continent          164 non-null object
country            164 non-null object
female literacy    164 non-null float64
fertility          164 non-null object
population         122 non-null float64
dtypes float64(2), object(3)
memory usage: 6.5+ KB
```

Frecuencia de valores

```
In [2]: df.continent.value_counts(dropna=False)  
Out[2]:  
AF      49  
ASI     47  
EUR     36  
LAT     24  
OCE      6  
NAM      2  
Name: continent, dtype: int64
```

df['continent'].value_counts(dropna=False)
da lo mismo

```
In [4]: df.country.value_counts(dropna=False).head()  
Out[4]:  
Sweden    2  
Algerie   1  
Germany   1  
Angola    1  
Indonésie 1  
Name: country, dtype: int64
```

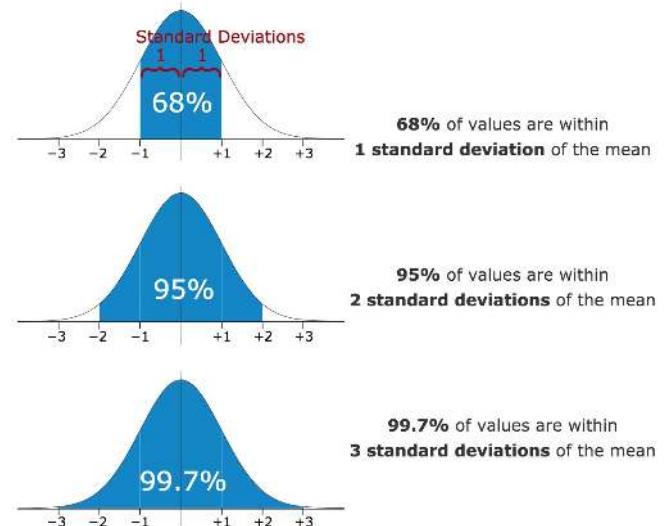
chaining = concatenación de
Métodos

Estadísticas generales

```
In [7]: df.describe()
```

```
Out[7]:
```

	female_literacy	population
count	164.000000	1.220000e+02
mean	80.301220	6.345768e+07
std	22.977265	2.605977e+08
min	12.600000	1.035660e+05
25%	66.675000	3.778175e+06
50%	90.200000	9.995450e+06
75%	98.500000	2.642217e+07
max	100.000000	2.313000e+09



Gráficos Generales

Histogramas: Distribución continua de datos

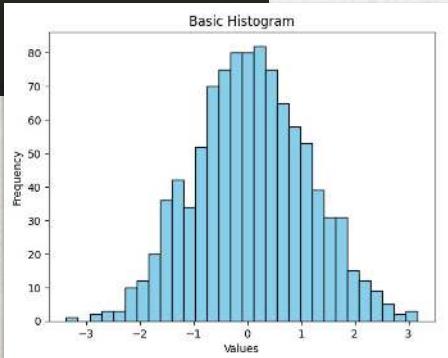
```
import matplotlib.pyplot as plt
import numpy as np

# Generate random data for the histogram
data = np.random.randn(1000)

# Plotting a basic histogram
plt.hist(data, bins=30, color='skyblue', edgecolor='black')

# Adding labels and title
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Basic Histogram')

# Display the plot
plt.show()
```

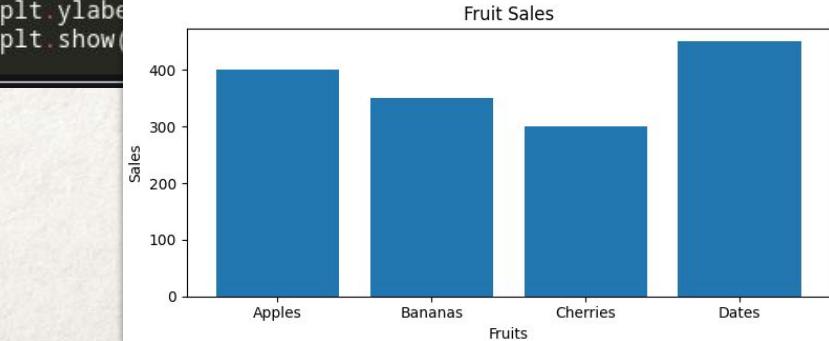


Bar plot: Datos discretos o categóricos

```
import matplotlib.pyplot as plt
import numpy as np

fruits = ['Apples', 'Bananas', 'Cherries', 'Dates']
sales = [400, 350, 300, 450]

plt.bar(fruits, sales)
plt.title('Fruit Sales')
plt.xlabel('Fruits')
plt.ylabel('Sales')
plt.show()
```



Box Plot vs Scatter Plot

```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np

# Creating dataset
np.random.seed(10)

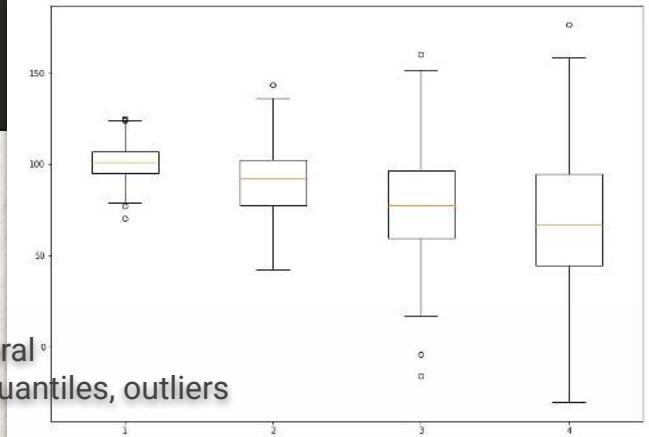
data_1 = np.random.normal(100, 10, 200)
data_2 = np.random.normal(90, 20, 200)
data_3 = np.random.normal(80, 30, 200)
data_4 = np.random.normal(70, 40, 200)
data = [data_1, data_2, data_3, data_4]

fig = plt.figure(figsize =(10, 7))

# Creating axes instance
ax = fig.add_axes([0, 0, 1, 1])

# Creating plot
bp = ax.boxplot(data)

# show plot
plt.show()
```

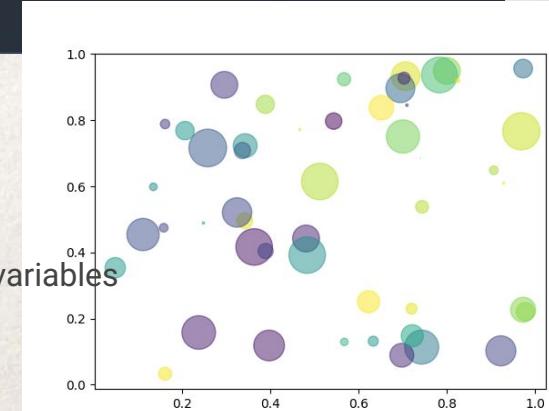


```
import matplotlib.pyplot as plt
import numpy as np

# Fixing random state for reproducibility
np.random.seed(19680801)

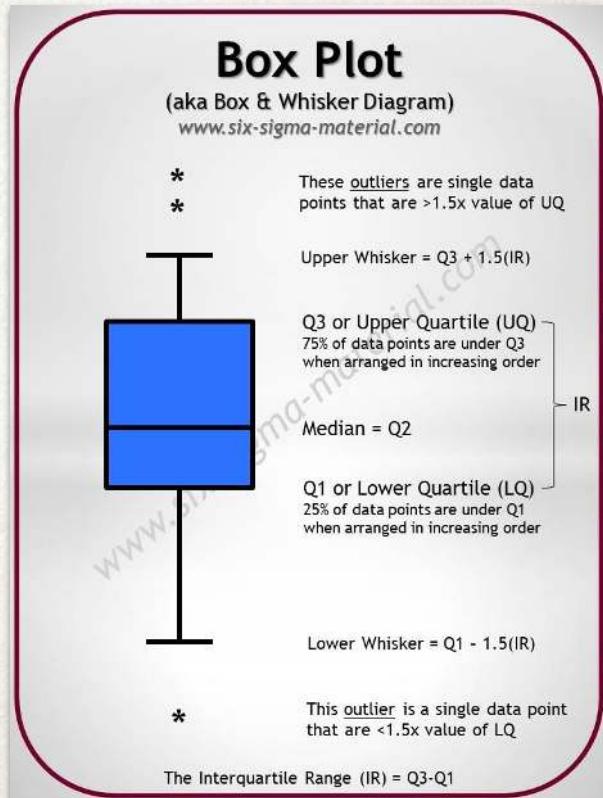
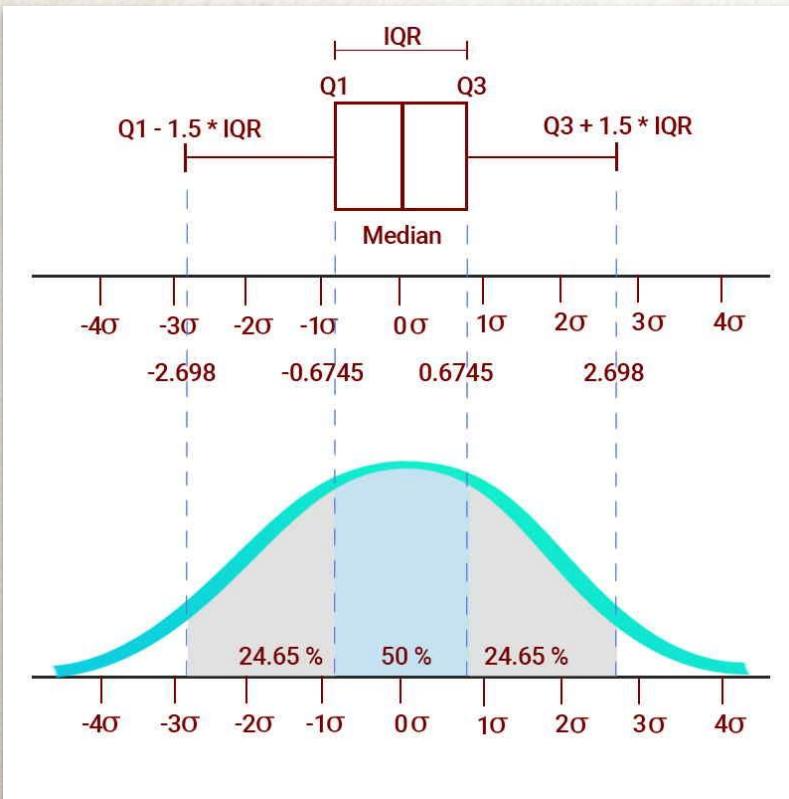
N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2 # 0 to 15 point radii
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```

Relación entre variables



Box-Plots

Distribución e identificación de valores extremos



Inspección Visual

Python missingno

```
# Program to visualize missing values in dataset

# Importing the libraries
import pandas as pd
import missingno as msno

# Loading the dataset
df = pd.read_csv("kamyr-digester.csv")

# Visualize missing values as a matrix
msno.matrix(df)
```



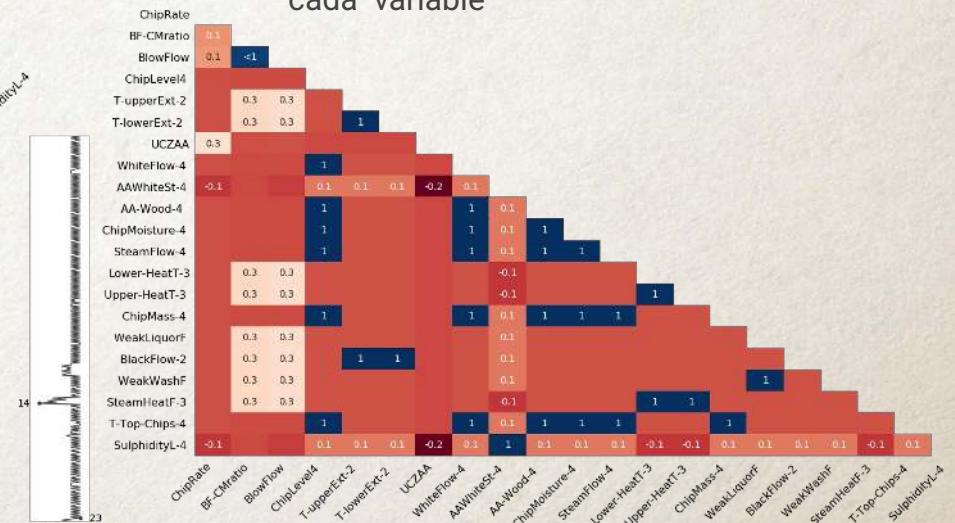
```
# Program to visualize missing values in dataset

# Importing the libraries
import pandas as pd
import missingno as msno

# Loading the dataset
df = pd.read_csv("kamyr-digester.csv")

# Visualize the correlation between the number of
# missing values in different columns as a heatmap
msno.heatmap(df)
```

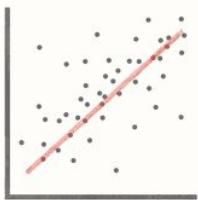
Correlación entre valores perdidos en cada variable



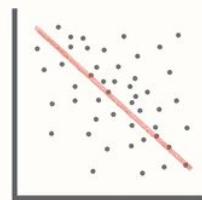
Inspección Visual

Correlación

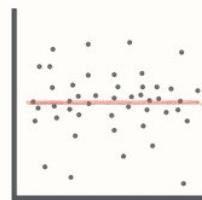
Correlation Coefficient



Positive Correlation



Negative Correlation



No Correlation

$$-1 \leq r \leq 1$$

ThoughtCo.

$$r=0$$

$$r = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

Correlación plot en Missingno:

- $r \sim -1$: \Rightarrow si aparece una variable, es muy probable que falte la otra.
- $r \sim 0$: \Rightarrow no hay dependencia entre la aparición de valores faltantes de dos variables.
- $r \sim 1$: \Rightarrow si aparece una variable, es muy probable que la otra variable esté presente.

Obs: La existencia de correlación no implica necesariamente relación de causa-efecto entre variables.
También pueden haber relaciones fortuitas entre variables: Ej. máximos solares y guerras

Datos Faltantes y Duplicados

Método `drop_duplicates()`

	name	sex	treatment a	treatment b
0	Daniel	male	-	42
1	John	male	12	31
2	Jane	female	24	27
3	Daniel	male	-	42

```
In [1]: df = df.drop_duplicates()
```

```
In [2]: print(df)
```

	name	sex	treatment a	treatment b
0	Daniel	male	-	42
1	John	male	12	31
2	Jane	female	24	27

Datos Faltantes y Duplicados

Estrategias para datos faltantes

¿Que hago con los datos faltantes o erróneos?

- Remoción : método `.dropna()`
- Imputación: método `.fillna()`
 - Reemplazo por: media general
 - media de los vecinos próximos
 - interpolacion
 - valor predefinido
 - etc.

```
tips_dropped = tips_nan.dropna()
```

```
In [6]: tips_nan['sex'] = tips_nan['sex'].fillna('missing')
```

Conversiones de Tipo

```
In [2]: df['treatment b'] = df['treatment b'].astype(str)
```

```
In [3]: df['sex'] = df['sex'].astype('category')
```

```
In [4]: df.dtypes
```

```
Out[4]:
```

```
name          object
sex           category
treatment a   object
treatment b   object
dtype: object
```

```
col_names=['Datetime', 'gic', 'T1', 'T2'];

missing_vals = ["NaN", "NO DATA"]

files = glob.glob("gic_grp*.csv")

df = pd.DataFrame([], columns=col_names)

for file in files:
    raw = pd.read_csv(file,
                      header = None,
                      skiprows = 1,
                      usecols = [0,1,2,3],
                      parse_dates = [0],
                      na_values = missing_vals,
                      names = col_names);
    df = pd.concat([df, raw], ignore_index=True)

convert_dict = {'Datetime': 'datetime64[ns]',
                'gic': float,
                'T1' : float,
                'T2' : float}

df = df.astype(convert_dict);
```

Cadenas de caracteres

Paquete re

- Gran parte de la limpieza de datos implica la manipulación de cadenas de caracteres
- La mayoría de los datos del mundo son texto no estructurado
- Manipulación de cadenas para hacer conjuntos de datos coherentes entre sí

Uso de expresiones regulares

- Compilar el patrón
- Utilice el patrón compilado para hacer coincidir los valores,
- Esto nos permite usar el patron una y otra vez
- Útil ya que queremos hacer coincidir los valores a lo largo de una columna de valores

```
In [1]: import re
```

```
In [2]: pattern = re.compile('^\$\d*\.\d{2}')
```

```
In [3]: result = pattern.match('$17.89')
```

```
In [4]: bool(result)
```

```
True
```

<https://docs.python.org/3/library/re.html#>

Cadenas de caracteres

Método replace()

El método **replace()** devuelve una nueva cadena con algunos caracteres de la cadena original reemplazados. **La cadena original no se ve afectada ni modificada.**

string.replace(old_char, new_char, count)

```
str = "I love JavaScript. I prefer JavaScript to Python because JavaScript looks mo:  
  
new_str = str.replace("JavaScript", "PHP")  
  
print(new_str)  
# I love PHP. I prefer PHP to Python because PHP looks more beautiful
```

old_char: cadena original
new_char: cadena nueva
count: max. nro de cambios, (todos por defecto)

```
s = 'one two one two one'  
# reemplazo todas las ocurrencias  
print(s.replace('one', 'XXX')) (  
# XXX two XXX two XXX  
# reemplazo solo en 2 ocurrencias  
print(s.replace('one', 'XXX', 2))  
# XXX two XXX two one
```

Cadenas de caracteres

Métodos `.translate()`, `re.sub()` & `re.subn()`

Utilice el método `translate()` para reemplazar varios caracteres diferentes. Puedes crear la tabla de traducción necesaria para `translate()` usando `str.maketrans()`.

```
s = 'one two one two one'

print(s.translate(str.maketrans({'o': '0', 't': 'T'})))
# One Tw0 One Tw0 One

print(s.translate(str.maketrans({'o': 'XXX', 't': None})))
# XXXne wXXX XXXne wXXX XXXne
```

También se puede usar `re.sub()` & `re.subn()`, la url abajo ofrece varios ejemplos

<https://note.nkmk.me/en/python-str-replace-translate-re-sub/>

Formateo & Conversión de Fechas

Paquete `datetime`

Parsing: Convertir un string en un objeto tipo datetime: `strptime()`

```
>>> from datetime import datetime  
>>> datetime.strptime("2020-01-01 14:00", "%Y-%m-%d %H:%M")  
datetime.datetime(2020, 1, 1, 14, 0)
```

Formatting: Convertir un objeto tipo datetime en un string : `strftime()`

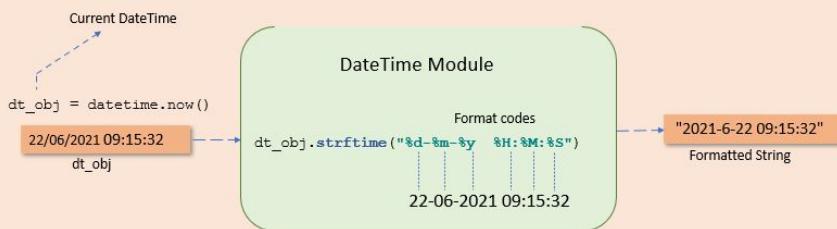
```
>>> from datetime import datetime  
>>> date = datetime.strptime("2020-01-01 14:00", "%Y-%m-%d %H:%M")  
>>> datetime.strftime(date, "%a %B %d, %Y %I:%M %p")  
'Wed January 01, 2020 02:00 PM'
```

Códigos de Conversión de Fechas

```
date_time = now.strftime("%m/%d/%Y, %H:%M:%S")
```

12/24/2018

04:59:31



Directive	Description	Example
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%W	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	5
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	8
%f	Microsecond 000000-999999	548513
%z	UTC offset	100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%C	Local version of date and time	Mon Dec 31 17:41:00 2018
%C	Century	20
%x	Local version of date	12/31/18
%X	Local version of time	17.41.00
%%	A % character	%
%G	ISO 8601 year	2018
%u	ISO 8601 weekday (1-7)	1
%V	ISO 8601 weeknumber (01-53)	1

Operaciones con Fechas

```
1 >>> from datetime import datetime, timedelta
2 >>> date = datetime.now()
3 >>> date
4 datetime(2020, 2, 6, 14, 49, 14, 277747)
5 >>> date - timedelta(minutes=10)
6 datetime(2020, 2, 6, 14, 39, 14, 277747)
7 >>> date - timedelta(hours=10)
8 datetime(2020, 2, 6, 4, 49, 14, 277747)
9 >>> date - timedelta(days=10)
10 datetime(2020, 1, 27, 14, 49, 14, 277747)
```

.timedelta()

Adiciona un corrimiento temporal

En Pandas puedo crear un rango de fechas

```
indx = pd.date_range('2021-01-01 00:00:00', '2021-07-31-23:59:00', freq='min')
```

Tidy Data

Concepto de Datos Ordenados (Hadley Wickham, 2014)

¿Por que los datos tienen que estar ordenados?:

1. Hay una ventaja general al elegir una forma coherente de almacenar datos. Si tiene una estructura de datos coherente, es más fácil aprender las herramientas que funcionan con ella porque tienen una uniformidad subyacente. Algunas herramientas, por ejemplo, el paquete de visualización de datos `seaborn`, están diseñadas teniendo en cuenta los datos ordenados.
2. Hay una ventaja específica en la colocación de variables en columnas, ya que le permite aprovechar las operaciones vectorizadas de `pandas`.

Los datos ordenados no van a ser apropiados siempre, pero son un valor predeterminado muy bueno para los datos tabulares. Una vez que lo use como predeterminado, es más fácil pensar en cómo realizar las operaciones posteriores.

country	year	cases	population
Afghanistan	2000	45	1071
Afghanistan	2009	566	2095060
Brazil	2009	31737	17206362
Brazil	2010	80388	17304898
China	1990	214558	127255272
China	2010	294566	12825583

country	year	cases	population
Afghanistan	2000	45	1071
Afghanistan	2009	566	2095060
Brazil	2009	31737	17206362
Brazil	2010	80388	17304898
China	1990	214558	127255272
China	2010	294566	12825583

formato ancho (messy)

id	bp1	bp2
A	100	120
B	140	115
C	120	125

formato largo, (tidy)

id	measurement	value
A	bp1	100
	bp2	120
B	bp1	140
	bp2	115
C	bp1	120
	bp2	125

Melt vs Pivot

Dos problemas comunes que se encuentran en los datos que se ingieren y que hacen que estén desordenados:

- Una variable puede estar distribuida en varias columnas.
- Una observación puede estar dispersa en varias filas.

Para el primero, debemos "fundir", (**melt**) los datos anchos, con múltiples columnas, en datos largos.

Para el último, necesitamos desapilar o **pivolar** las múltiples filas en columnas (es decir, ir de largo a ancho).

Mejor para reporting vs mejor para análisis

Obs: En algunas operaciones el formato tidy no es adecuado, p. ej. diferenciación en series temporales

	name	treatment a	treatment b
0	Daniel	-	42
1	John	12	31
2	Jane	24	27



	name	treatment	value
0	Daniel	treatment a	-
1	John	treatment a	12
2	Jane	treatment a	24
3	Daniel	treatment b	42
4	John	treatment b	31
5	Jane	treatment b	27

```
In [2]: pd.melt(frame=df, id_vars='name',
...:           value_vars=['treatment a', 'treatment b'],
...:           var_name='treatment', value_name='result')
```

```
Out[2]:
```

	name	treatment	result
0	Daniel	treatment a	-
1	John	treatment a	12
2	Jane	treatment a	24
3	Daniel	treatment b	42
4	John	treatment b	31
5	Jane	treatment b	27

```
In [5]: weather2_tidy = weather.pivot_table(values='value',
...:                                         index='date',
...:                                         columns='element',
...:                                         aggfunc=np.mean)
```

```
Out[5]:
```

element	tmax	tmin
date		
2010-01-30	27.8	14.5
2010-02-02	27.3	15.4

Sumario

- La limpieza de datos involucra combinar una gran cantidad de habilidades
- En general es necesario combinar datos de diversas fuentes
- No es posible implementar un algoritmo exitoso de ML con datos malos
- Python ofrece una gran variedad de paquetes y métodos para resolver la mayoría de los problemas comunes en los datos de trabajo
- Aun así, hay problemas difíciles de abordar y que requieren del expertise del operador
- El Tidy Data es muy útil a la hora de normalizar datos y realizar análisis pero no muy adecuado para realizar reportes y visualización