



# Statistique descriptive et visualisation

---

`titanic.describe()`

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
<b>count</b>	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
<b>mean</b>	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
<b>std</b>	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
<b>min</b>	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
<b>25%</b>	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
<b>50%</b>	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
<b>75%</b>	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
<b>max</b>	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
titanic[ "Age" ].mean()
```

```
titanic[ "Age" ].mean()
29.69911764705882
```

```
titanic[ [ "Age" , "Fare" ] ].median()
```

```
titanic[ [ "Age" , "Fare" ] ].median()
Age      28.0000
Fare     14.4542
dtype: float64
```

```
titanic.groupby( "Sex" ).mean()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
Sex							
female	431.028662	0.742038	2.159236	27.915709	0.694268	0.649682	44.479818
male	454.147314	0.188908	2.389948	30.726645	0.429809	0.235702	25.523893

```
titanic[ [ "Sex" , "Age" ] ].groupby( "Sex" ).mean()
titanic.groupby( "Sex" )[ [ "Sex" , "Age" ] ].mean()
```

	Age
Sex	
female	27.915709
male	30.726645

```
titanic[["Survived", "Pclass"]].groupby("Pclass").describe()
```

		Survived							
		count	mean	std	min	25%	50%	75%	max
Pclass									
1	216.0	0.629630	0.484026	0.0	0.0	1.0	1.0	1.0	
2	184.0	0.472826	0.500623	0.0	0.0	0.0	1.0	1.0	
3	491.0	0.242363	0.428949	0.0	0.0	0.0	0.0	1.0	

```
titanic.groupby("Pclass").sum()
```

	PassengerId	Survived	Age	SibSp	Parch	Fare
Pclass						
1	99705	136	7111.42	90	77	18177.4125
2	82056	87	5168.83	74	70	3801.8417
3	215625	119	8924.92	302	193	6714.6951

```
titanic.groupby("Sex").sum()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
Sex							
female	135343	233	678	7286.00	218	204	13966.6628
male	262043	109	1379	13919.17	248	136	14727.2865



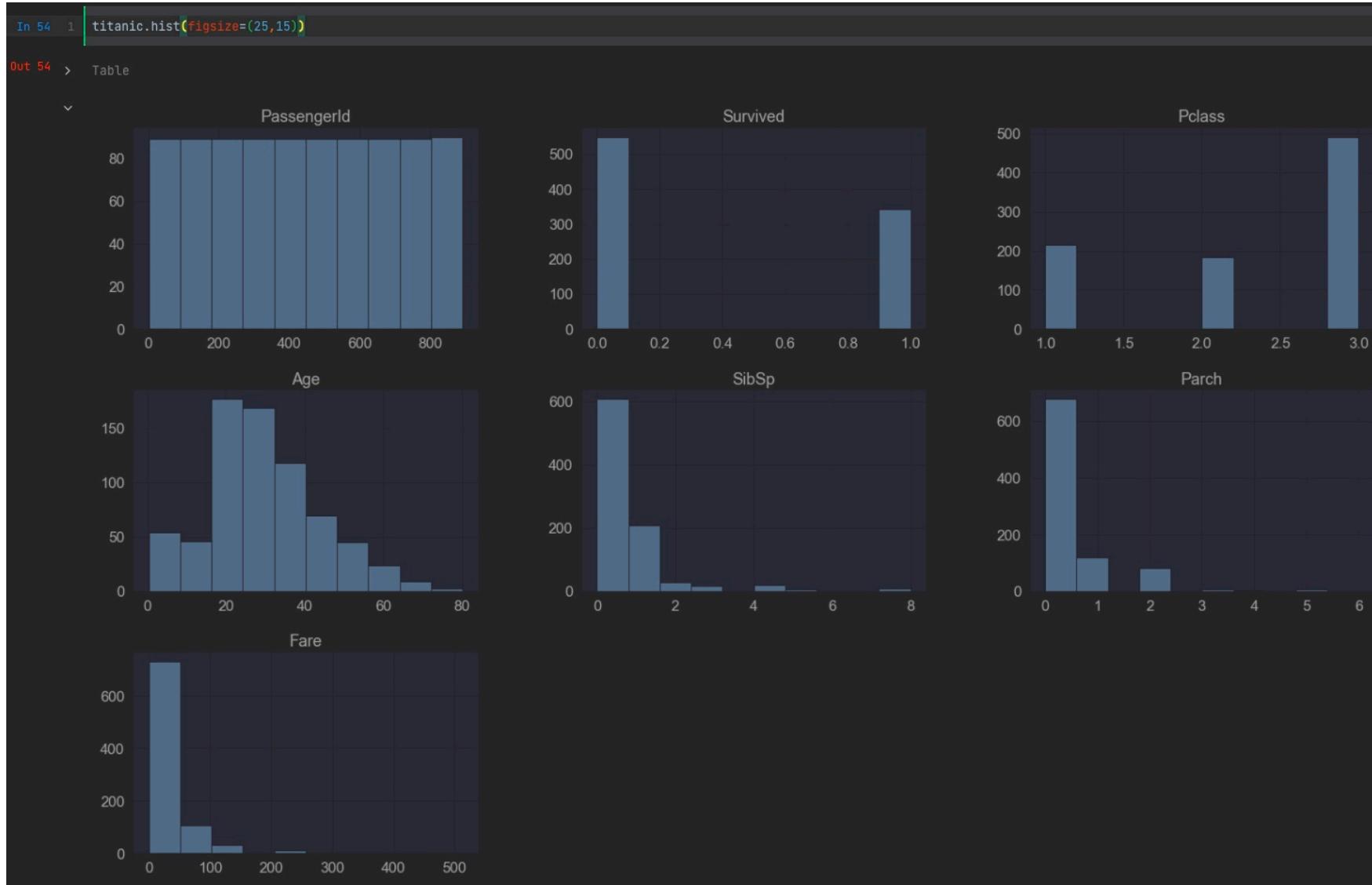
What is the mean ticket fare price for each of the sex and cabin class combinations?

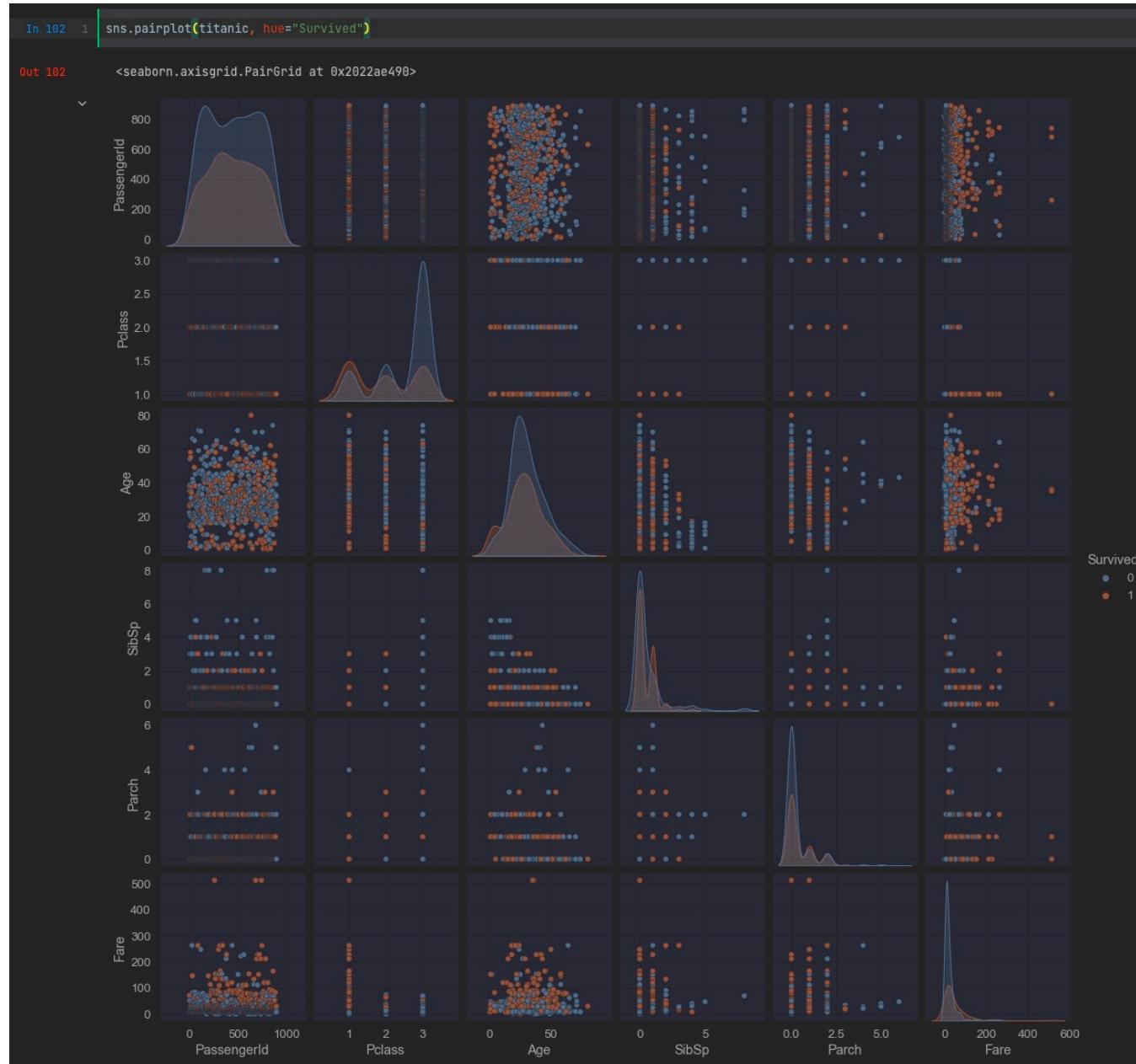
```
titanic.groupby(["Sex", "Pclass"])[ "Fare" ].mean()
```

Sex	Pclass	
female	1	106.125798
	2	21.970121
	3	16.118810
male	1	67.226127
	2	19.741782
	3	12.661633

```
titanic.groupby(["Pclass", "Sex"])[ "Fare" ].mean()
```

Pclass	Sex	
1	female	106.125798
	male	67.226127
2	female	21.970121
	male	19.741782
3	female	16.118810
	male	12.661633

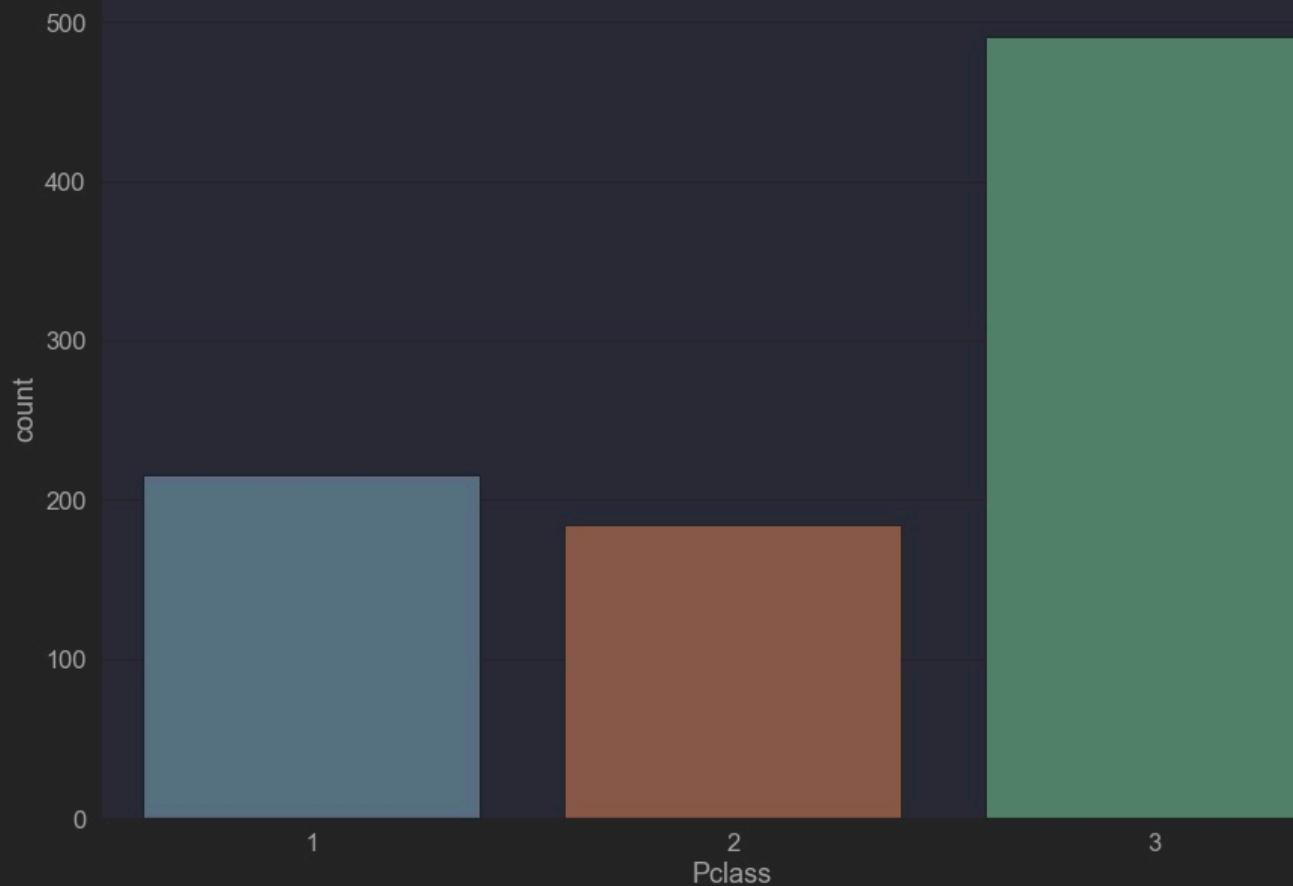




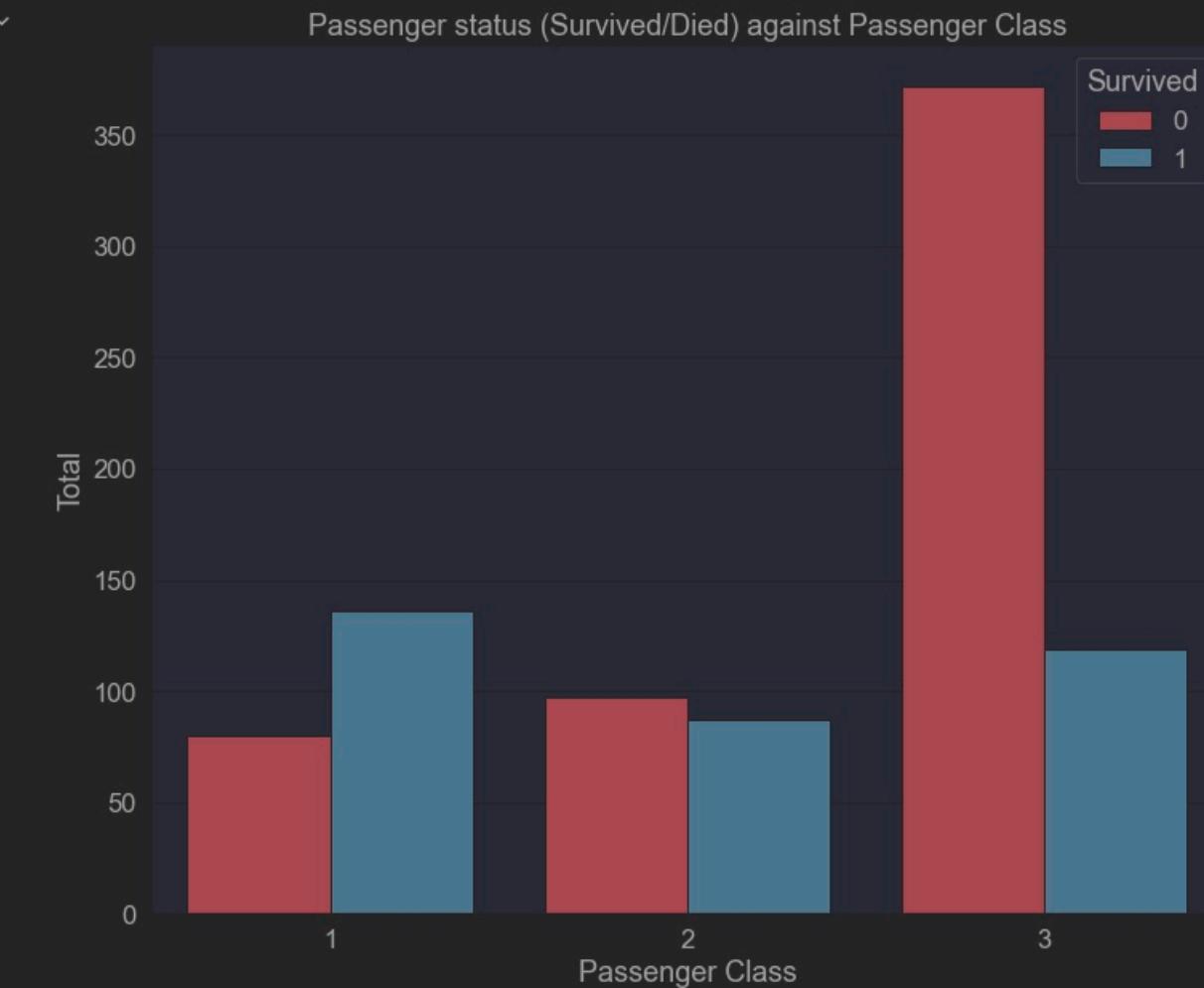
```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In 61 1 plt.figure(figsize=(15,10))
2 sns.countplot(x = 'Pclass', data = titanic)
```

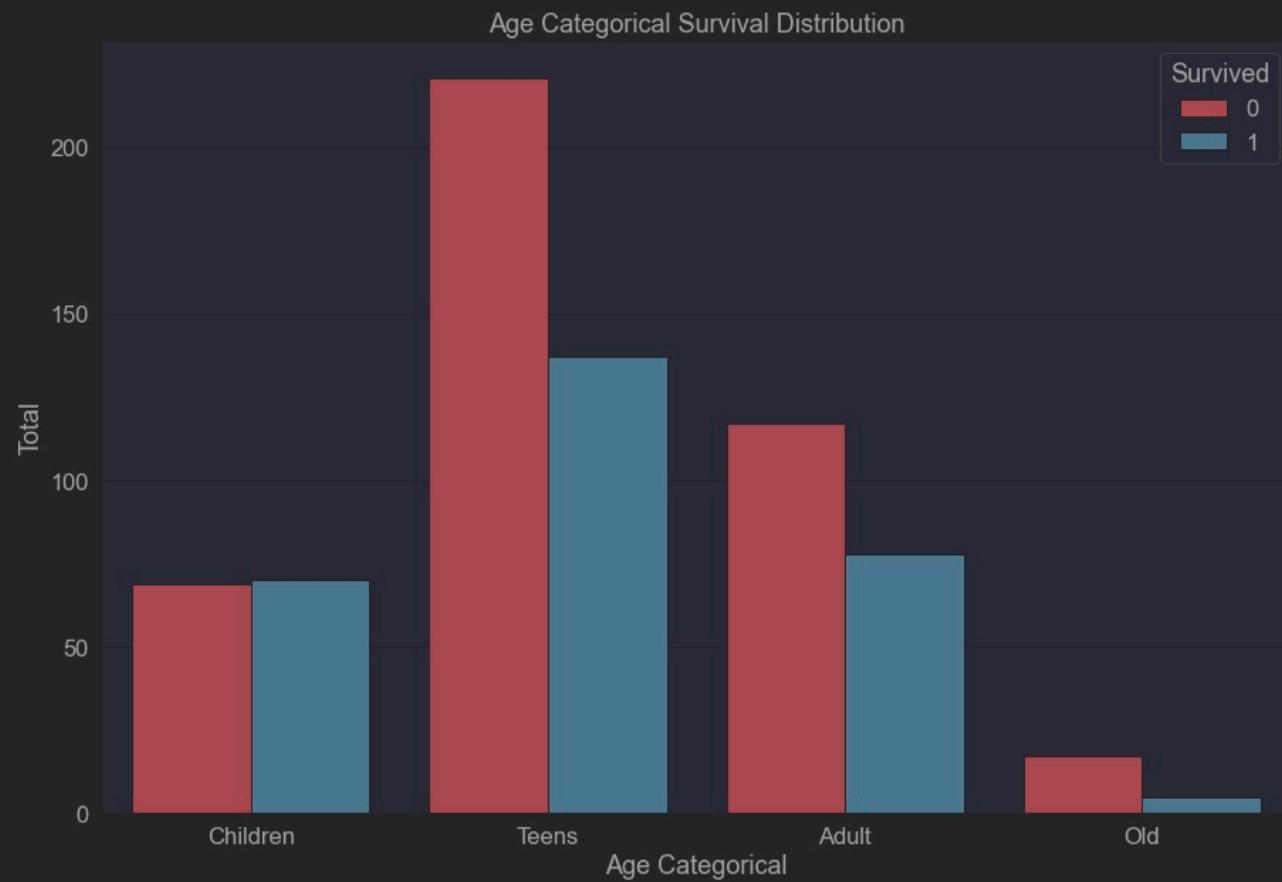
```
Out 61 <AxesSubplot:xlabel='Pclass', ylabel='count'>
```

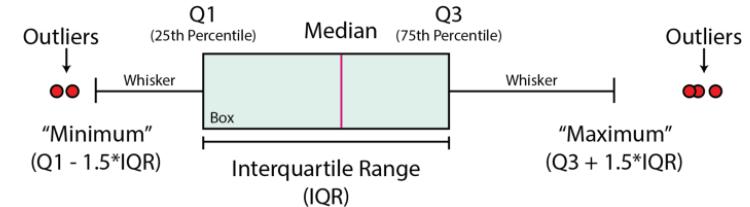


```
In 57 1 fig, ax = plt.subplots(1,1, figsize = (12,10))
2 ax = sns.countplot(x = 'Pclass', hue = 'Survived', palette = 'Set1', data = titanic)
3 ax.set(title = 'Passenger status (Survived/Died) against Passenger Class',
4        xlabel = 'Passenger Class', ylabel = 'Total')
5 plt.show()
```

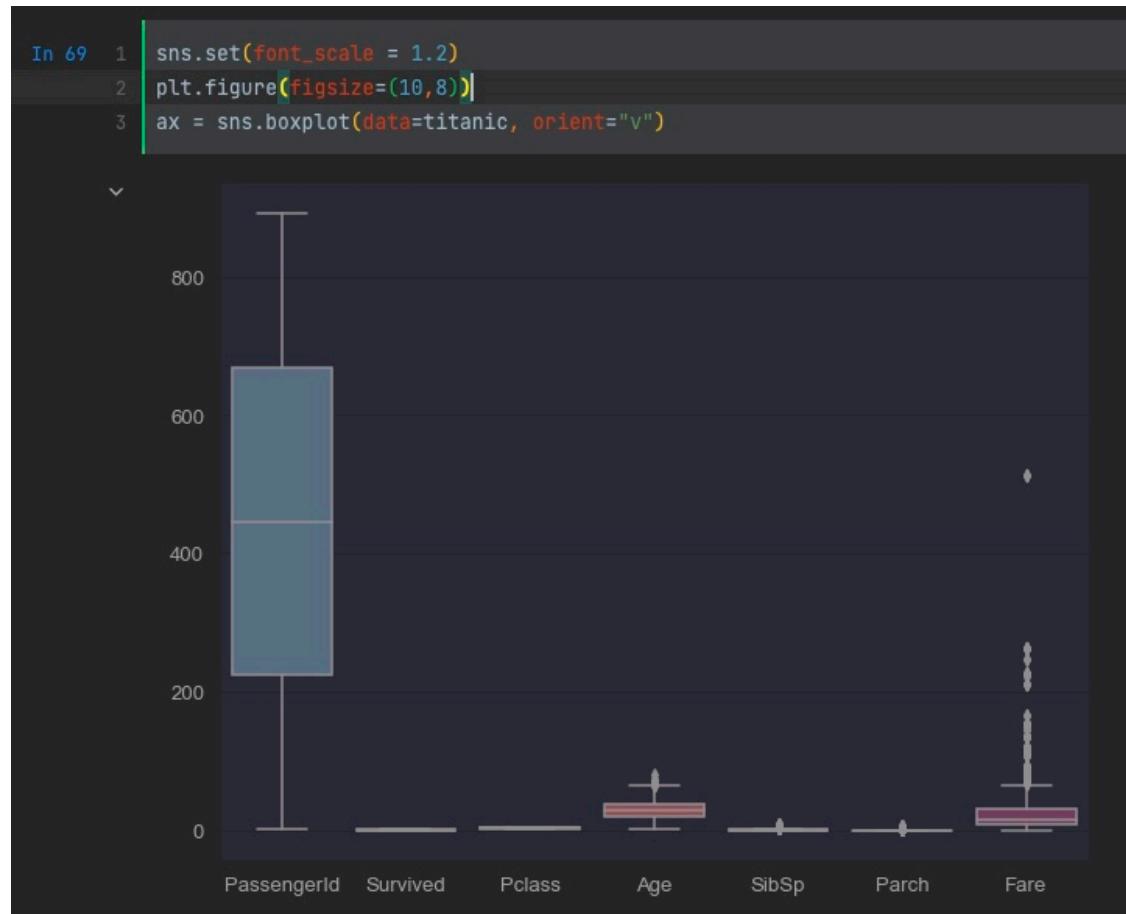


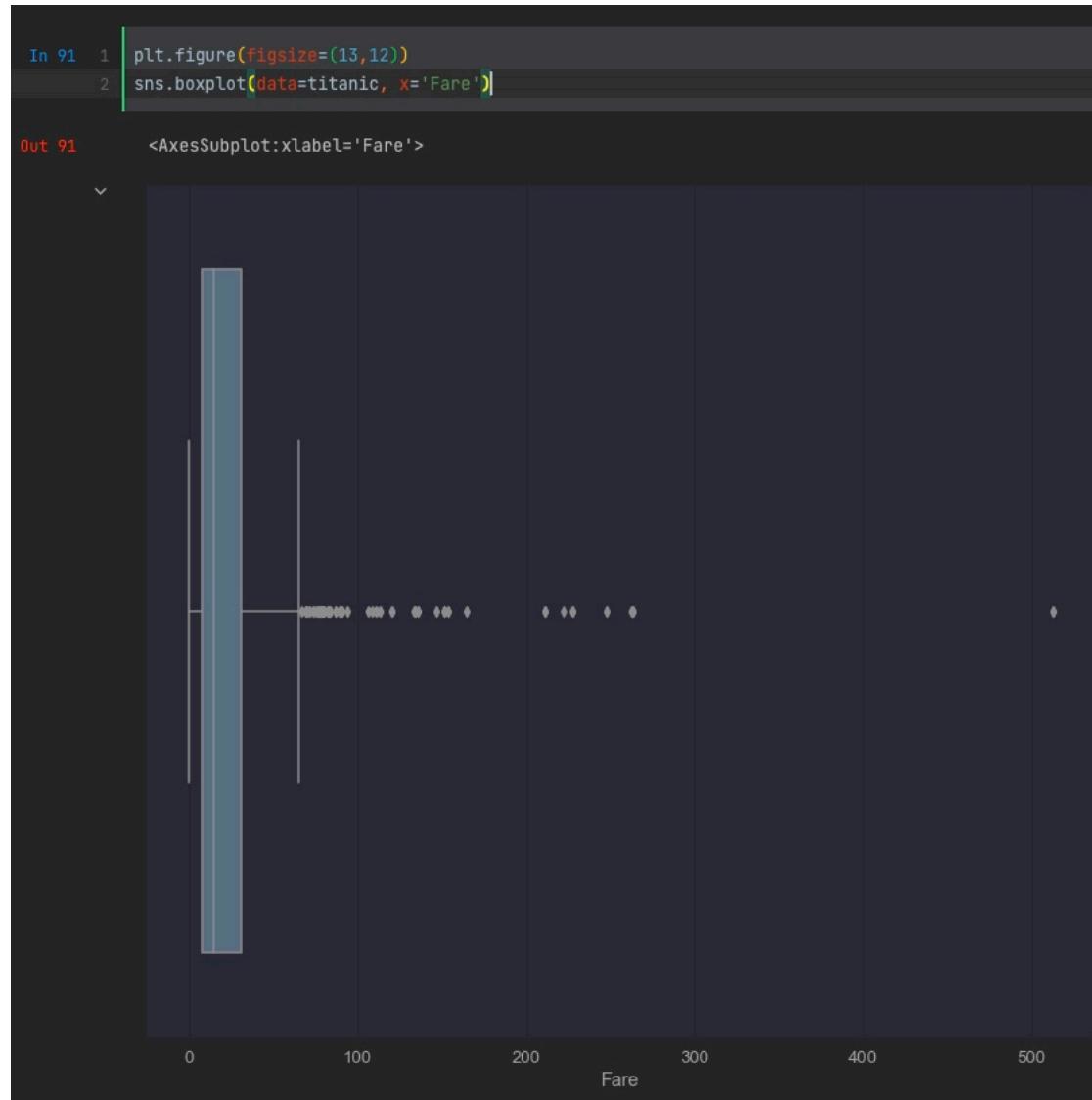
```
In 66 1 ⚡# We look at Age column and set Intevals on the ages and the map them to their categories as
2 ⚡# (Children, Teen, Adult, Old)
3 plt.figure(figsize=(15,10))
4 interval = (0,18,35,60,120)
5 categories = ['Children', 'Teens', 'Adult', 'Old']
6 titanic['Age_cats'] = pd.cut(titanic.Age, interval, labels = categories)
7 ax = sns.countplot(x = 'Age_cats', data = titanic, hue = 'Survived', palette = 'Set1')
8 ax.set(xlabel='Age Categorical', ylabel='Total',
9        title="Age Categorical Survival Distribution")
10 plt.show()
```





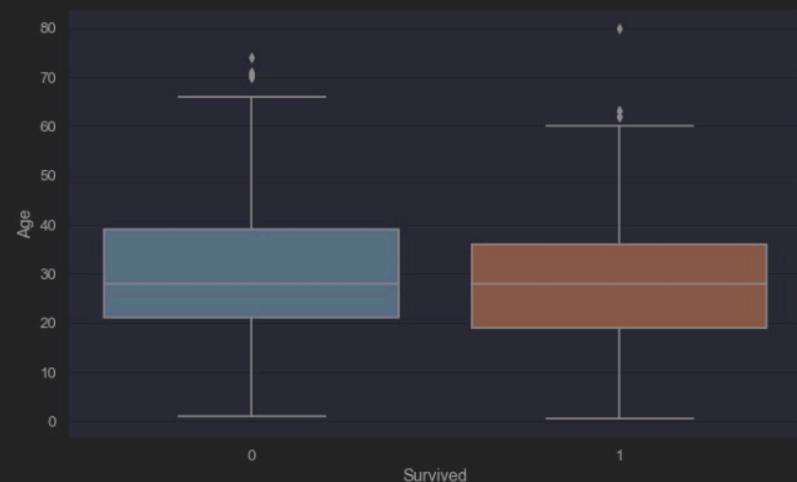
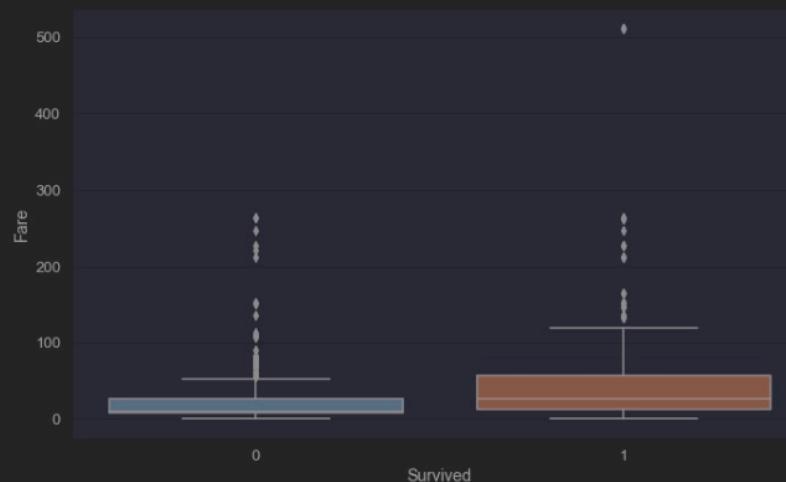
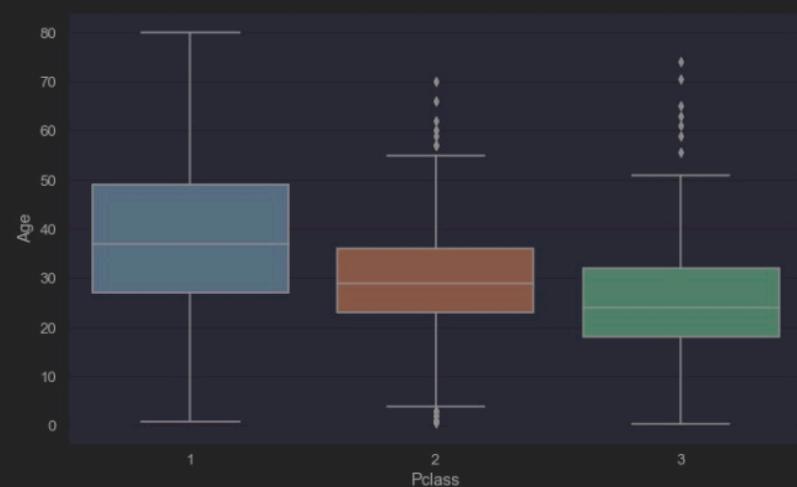
<https://towardsdatascience.com/creating-boxplots-of-well-log-data-using-matplotlib-in-python-34c3816e73f4>



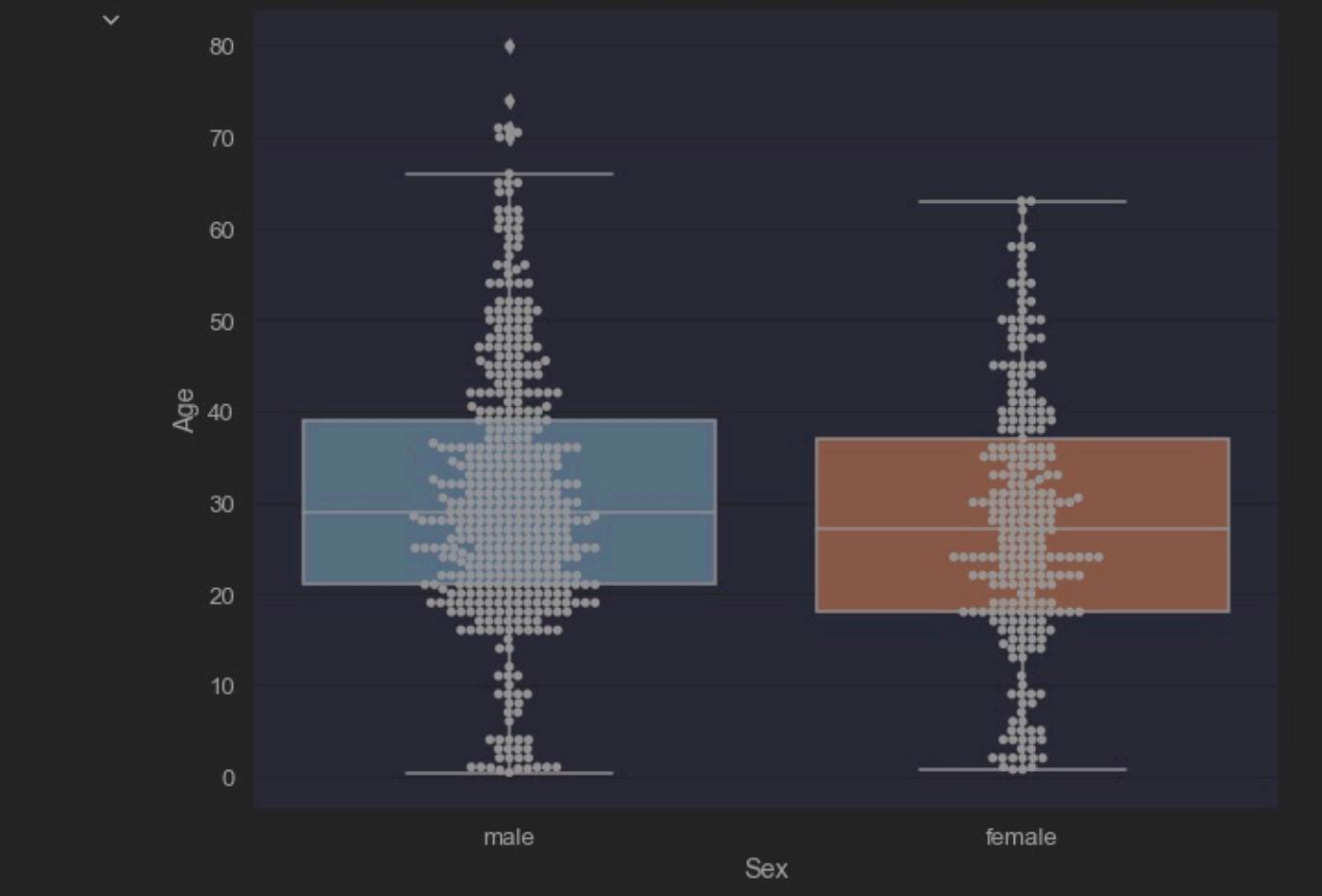


```
In 88 1 fig, axes = plt.subplots(2, 2, figsize=(25, 15))
2 sns.boxplot(ax=axes[0, 0], data=titanic, x='Pclass', y='Fare')
3 sns.boxplot(ax=axes[0, 1], data=titanic, x='Pclass', y='Age')
4 sns.boxplot(ax=axes[1, 0], data=titanic, x='Survived', y='Fare')
5 sns.boxplot(ax=axes[1, 1], data=titanic, x='Survived', y='Age')
```

```
Out 88 <AxesSubplot:xlabel='Survived', ylabel='Age'>
```



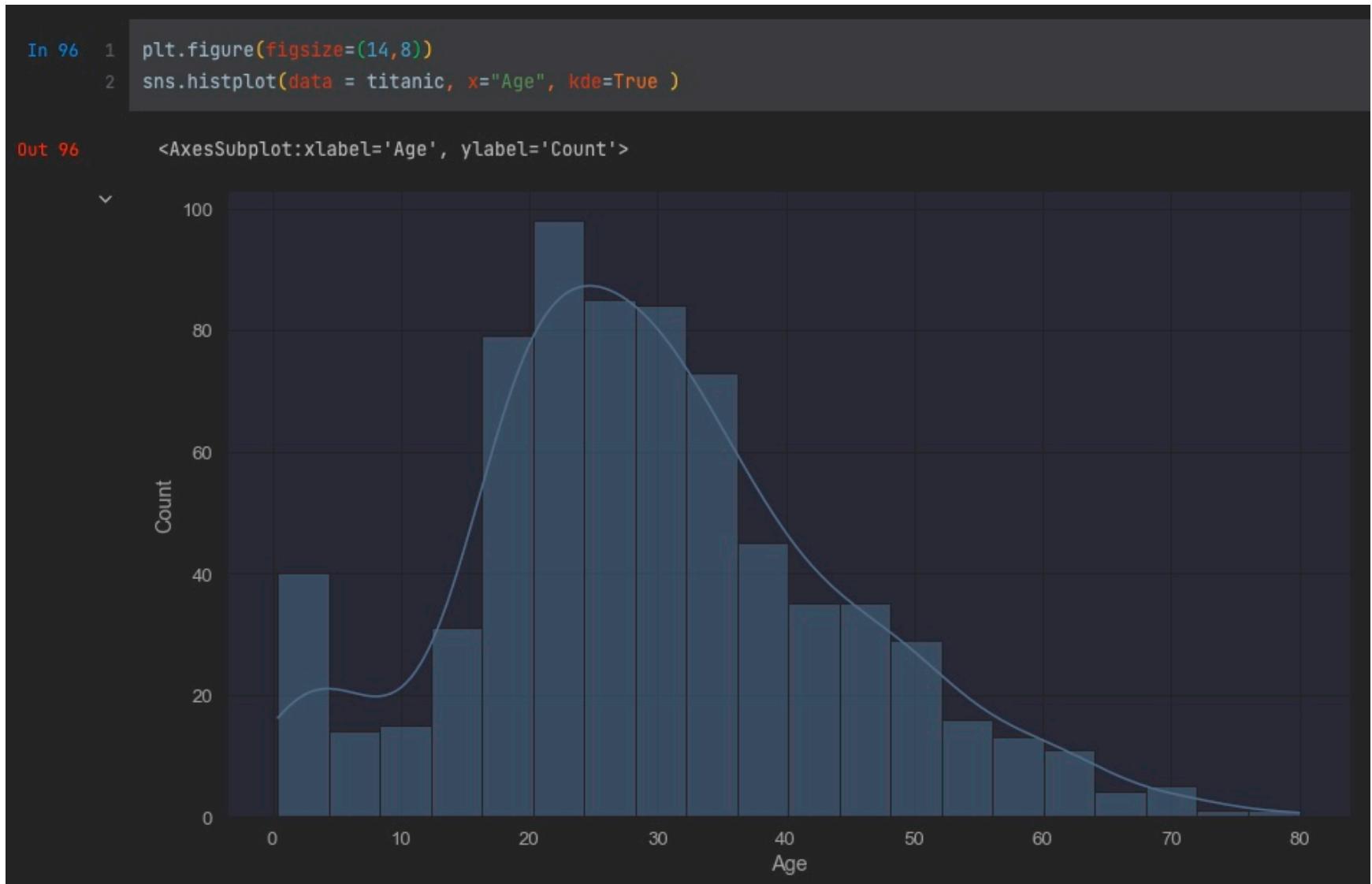
```
In 71 1 sns.set(font_scale = 1.2)
2 plt.figure(figsize=(10,8))
3 ax = sns.boxplot(x='Sex', y="Age", data=titanic, orient="v")
4 ax = sns.swarmplot(x='Sex', y="Age", data=titanic, color=".25")
```



```
In 74 1 sns.set(font_scale = 1.2)
2 plt.figure(figsize=(13,12))
3 ax = sns.catplot(x='Sex', y="Age", hue="Survived", col="Pclass", data=titanic, orient="v")
```

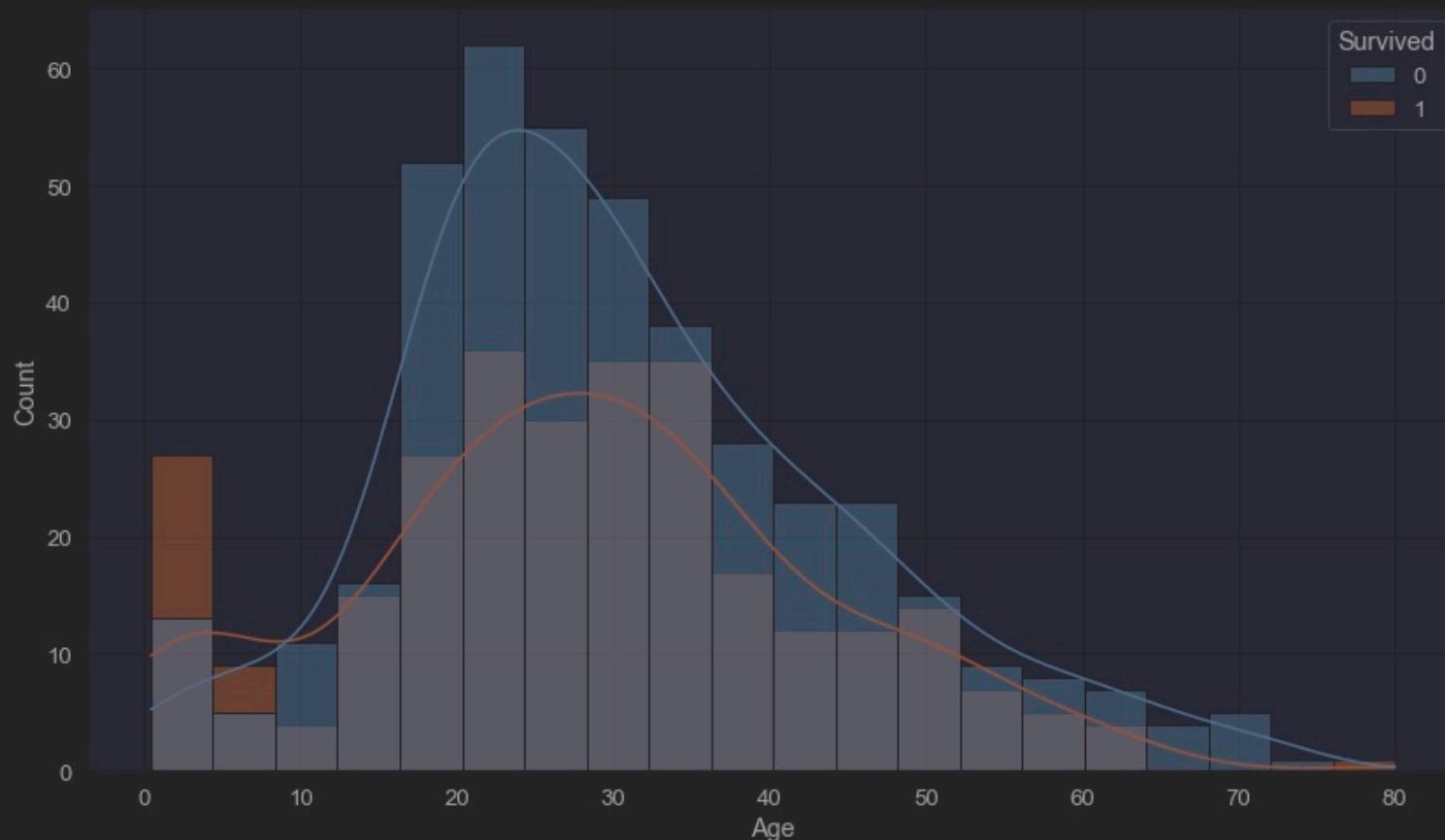
<Figure size 936x864 with 0 Axes>





```
In 97 1 plt.figure(figsize=(14,8))  
2 sns.histplot(data = titanic, hue="Survived", x="Age", kde=True )
```

```
Out 97 <AxesSubplot:xlabel='Age', ylabel='Count'>
```



# Camemberts

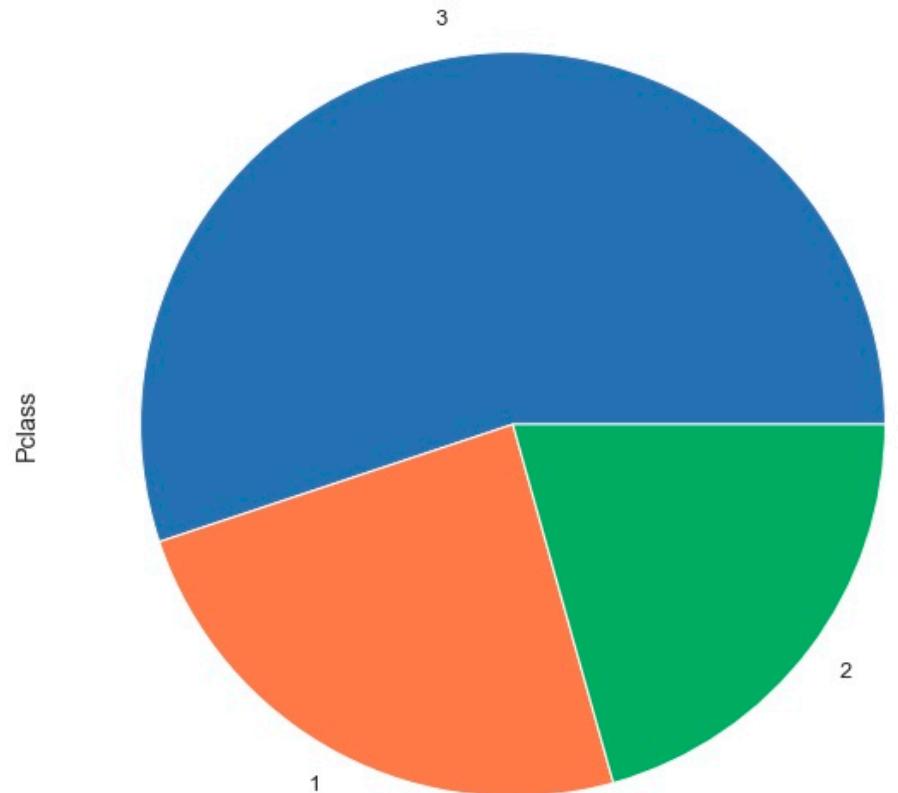
In 177 1 titanic.Pclass.value\_counts()

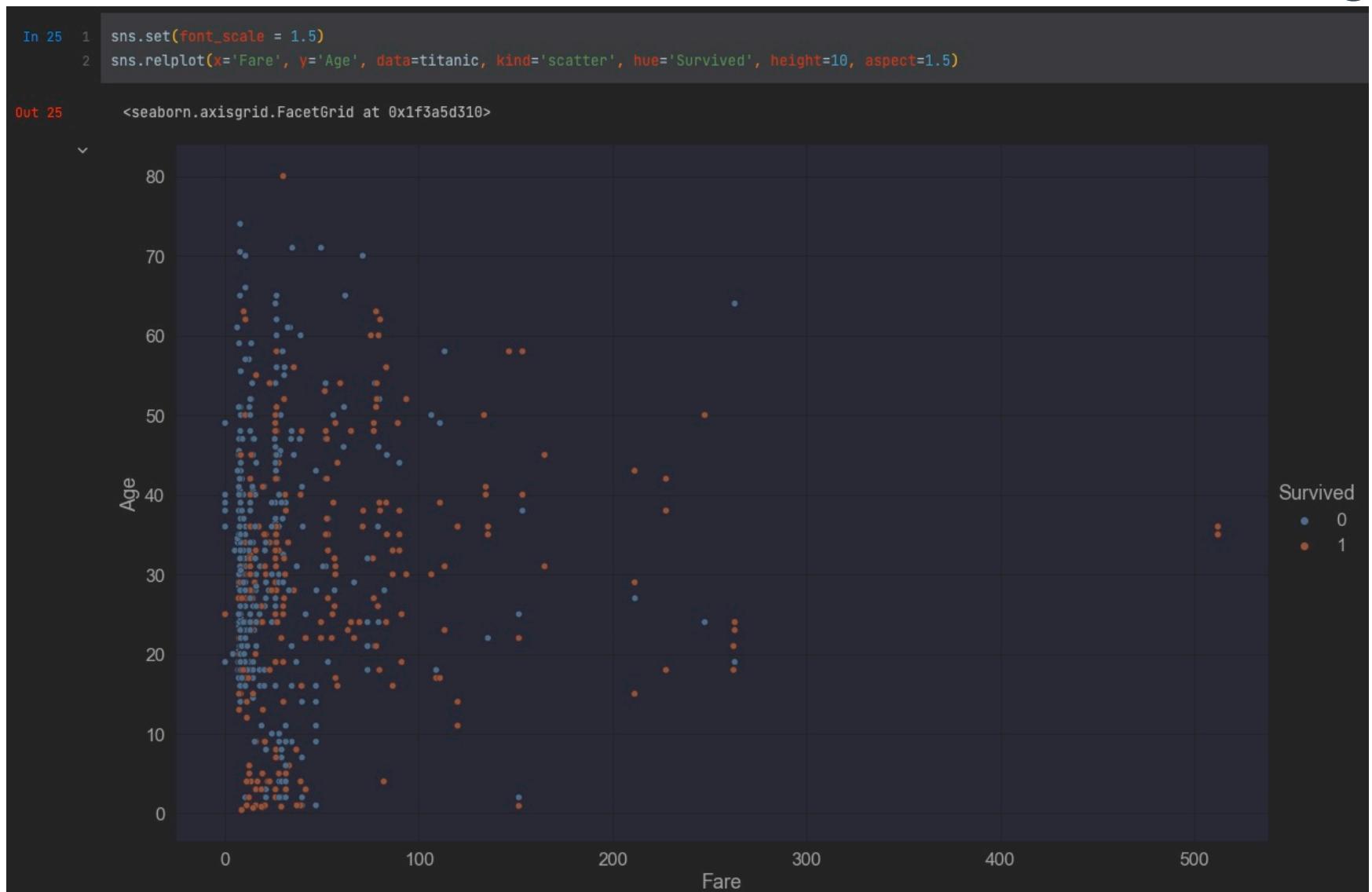
Out 177 ▾

Pclass	
3	491
1	216
2	184

Length: 3, dtype: int64 [Open in new tab](#)

```
plt.figure(figsize=(15,10))
titanic.Pclass.value_counts().plot(kind='pie')
<AxesSubplot:ylabel='Pclass'>
```





# Corrélation entre variables

```
corr = titanic.corr()
```

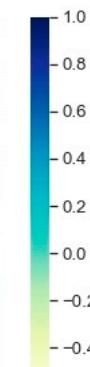
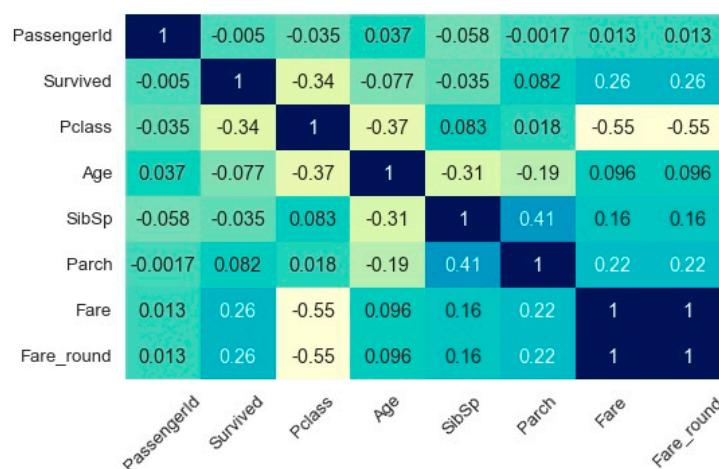
```
corr
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Fare_round
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658	0.012671
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307	0.257322
Pclass	-0.035144	-0.338481	1.000000	-0.369226	0.083081	0.018443	-0.549500	-0.549542
Age	0.036847	-0.077221	-0.369226	1.000000	-0.308247	-0.189119	0.096067	0.096076
SibSp	-0.057527	-0.035322	0.083081	-0.308247	1.000000	0.414838	0.159651	0.159736
Parch	-0.001652	0.081629	0.018443	-0.189119	0.414838	1.000000	0.216225	0.216290
Fare	0.012658	0.257307	-0.549500	0.096067	0.159651	0.216225	1.000000	1.000000
Fare_round	0.012671	0.257322	-0.549542	0.096076	0.159736	0.216290	1.000000	1.000000

```
plt.xticks(rotation=45)
```

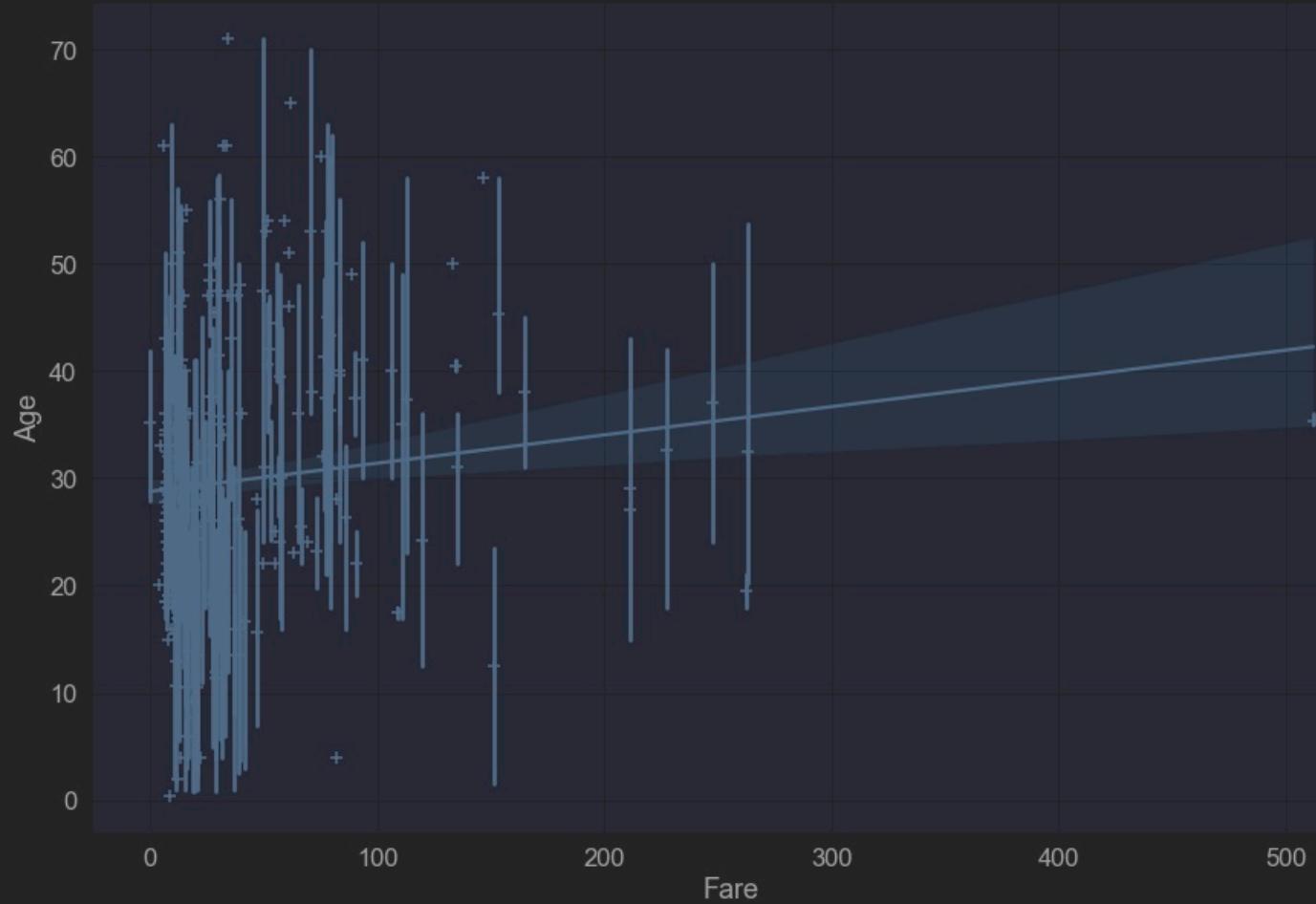
```
sns.heatmap(corr,cmap="YlGnBu",annot = True)
```

<AxesSubplot:>



```
In 33 1 plt.figure(figsize=(15,10))  
2 sns.regplot(x='Fare', y='Age', data=titanic, x_estimator=np.mean, marker='+')
```

```
Out 33 <AxesSubplot:xlabel='Fare', ylabel='Age'>
```

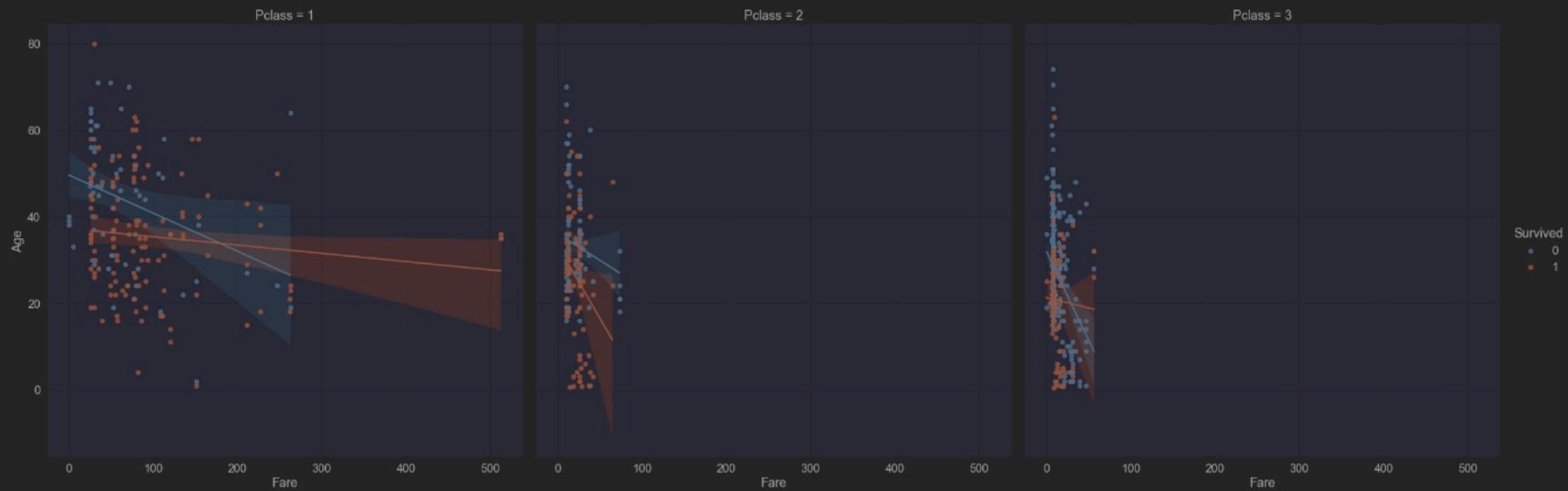




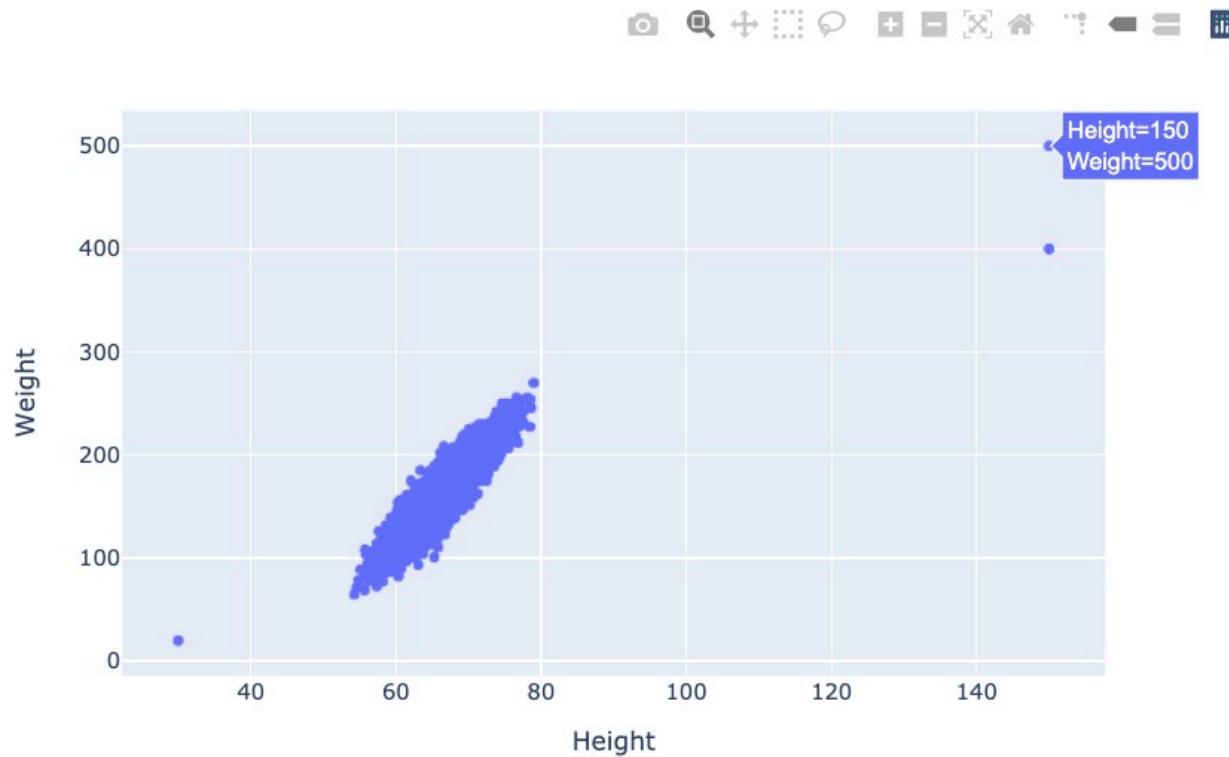
avec  
facettes

```
In 48 1 sns.set(font_scale = 1.5)
2 sns.lmplot(x='Fare', y='Age', hue="Survived", col='Pclass', data=titanic, height=10)
```

```
Out 48 <seaborn.axisgrid.FacetGrid at 0x1f46e7580>
```



```
import plotly.express as px
fig = px.scatter(wh, x='Height', y='Weight')
fig.show()
```



<https://medium.com/spatial-data-science/4-tools-to-speed-up-exploratory-data-analysis-eda-in-python-e240ebcd18de>

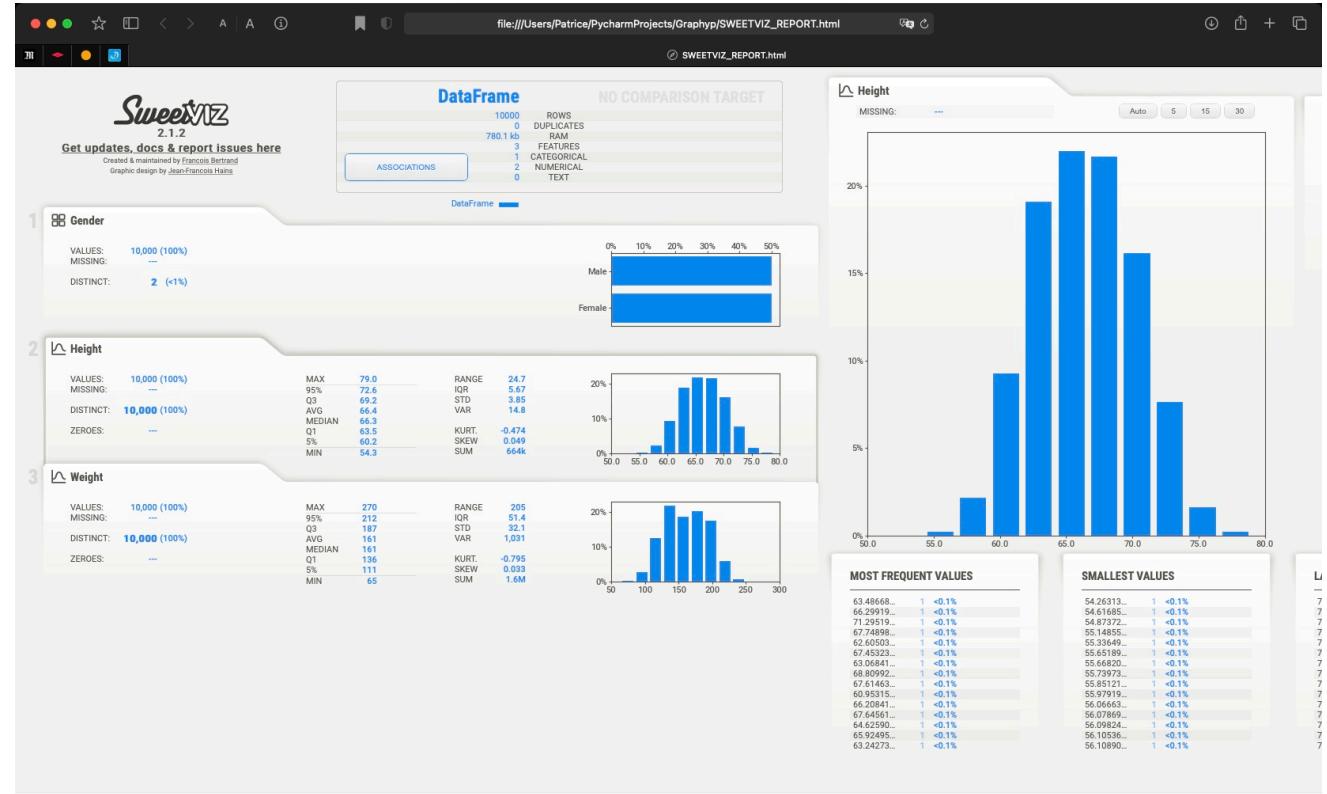
# Des modules « tout prêts »

<https://pypi.org/project/sweetviz/>

SweetViz

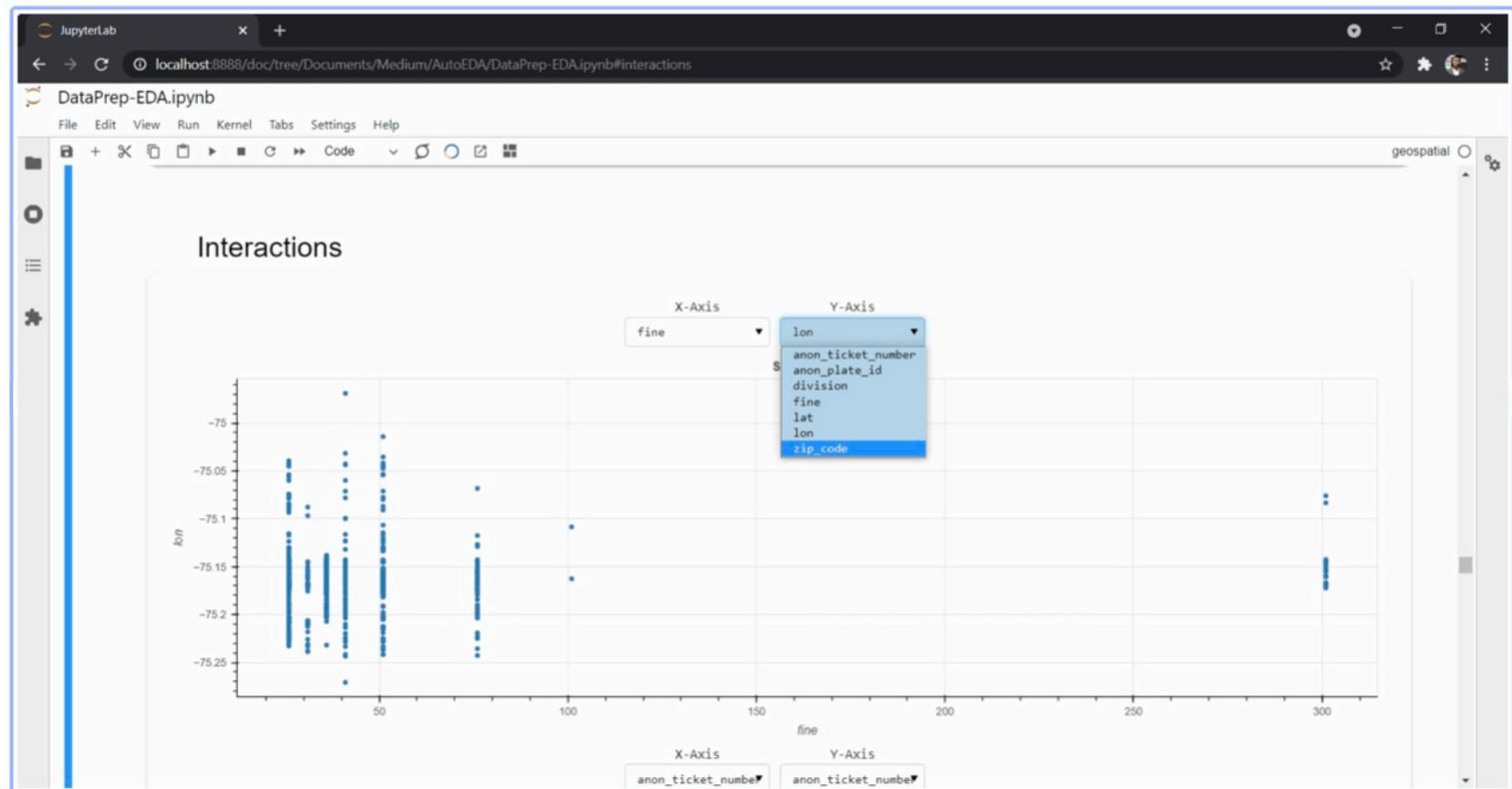
```
import sweetviz as sv

my_report = sv.analyze(df)
my_report.show_html()
```



```
from dataprep.eda import create_report  
create_report(df)
```

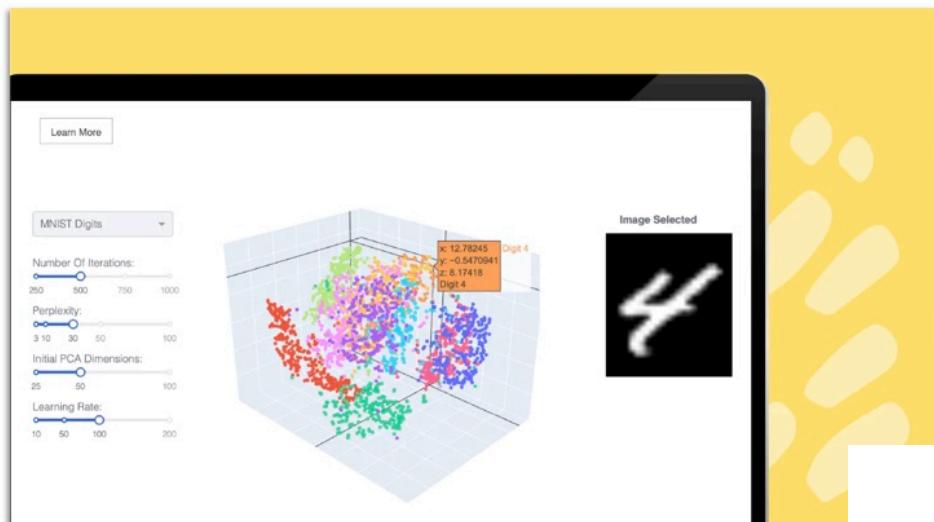
dataprep



DataPrep EDS report.

<https://plotly.com>

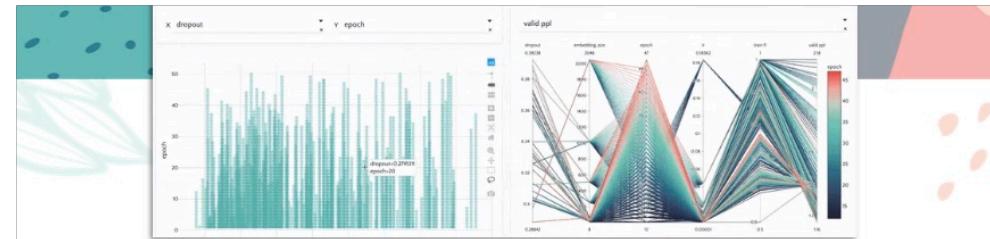
ML & AI apps made with Dash



## Clustering

Dash is the fastest way to deploy apps for dimensionality reduction.

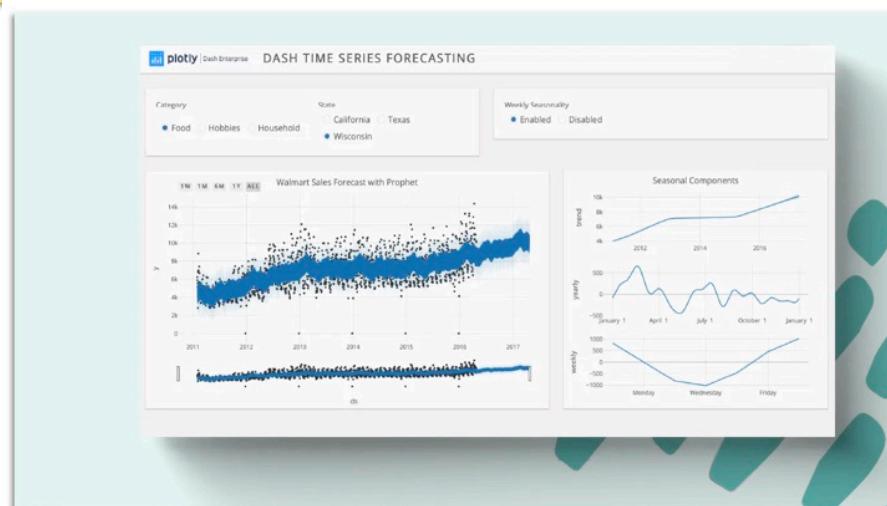
This Dash app demos TSNE in ~20 lines of Python code



## Dash Enterprise is the low-code platform for ML & data science apps.

With Dash Enterprise, full-stack AI applications that used to require a team of front-end, back-end, and DevOps engineers can now be built, deployed, and hyperscaled by a single data scientist within hours.

## ML & AI apps made with Dash



## Predictive Analytics

Dash is the fastest way to deploy Python-based apps for predictive analytics and forecasting.

Use predictive analytics + Dash to put neural networks, nonlinear regressions, decision trees, SVMs, and other forecasting methods in the hands of business users.

This Dash app demos Facebook's Prophet library in ~200 lines of Python code

• • ○ • •



# Données temporelles

---

# Le module de base *datetime*

```
import datetime
```

```
x = datetime.datetime.now()  
print(x)
```

2021-06-02 11:03:29.693092

```
print(x.year)          2021  
print(x.month)         6  
print(x.day)           2  
print(x.strftime("%B")) June
```

```
x = datetime.datetime(2020, 5, 17)
```

Directive	Description
%a	Weekday, short version
%A	Weekday, full version
%w	Weekday as a number 0-6, 0 is Sunday
%d	Day of month 01-31
%b	Month name, short version
%B	Month name, full version
%m	Month as a number 01-12
%y	Year, short version, without century
%Y	Year, full version
%H	Hour 00-23
%I	Hour 00-12
%p	AM/PM
%M	Minute 00-59
%S	Second 00-59
%f	Microsecond 000000-999999
%z	UTC offset
%Z	Timezone
%j	Day number of year 001-366
%U	Week number of year, Sunday as the first day of week, 00-53
%W	Week number of year, Monday as the first day of week, 00-53

%c	Local version of date and time
%x	Local version of date
%X	Local version of time
%%	A % character
%G	ISO 8601 year
%u	ISO 8601 weekday (1-7)
%V	ISO 8601 weeknumber (01-53)

# Les modules NumPy *datetime64* et *timedelta*

```
import numpy as np
np.datetime64('2005-02-25')
```

## Example

All the dates for one month:

```
>>> np.arange('2005-02', '2005-03', dtype='datetime64[D]')
array(['2005-02-01', '2005-02-02', '2005-02-03', '2005-02-04',
       '2005-02-05', '2005-02-06', '2005-02-07', '2005-02-08',
       '2005-02-09', '2005-02-10', '2005-02-11', '2005-02-12',
       '2005-02-13', '2005-02-14', '2005-02-15', '2005-02-16',
       '2005-02-17', '2005-02-18', '2005-02-19', '2005-02-20',
       '2005-02-21', '2005-02-22', '2005-02-23', '2005-02-24',
       '2005-02-25', '2005-02-26', '2005-02-27', '2005-02-28'],
      dtype='datetime64[D]')
```

## Example

```
>>> np.datetime64('2009-01-01') - np.datetime64('2008-01-01')
numpy.timedelta64(366,'D')
```

```
>>> np.datetime64('2009') + np.timedelta64(20, 'D')
numpy.datetime64('2009-01-21')
```

```
>>> np.datetime64('2011-06-15T00:00') + np.timedelta64(12, 'h')
numpy.datetime64('2011-06-15T12:00')
```

# Pandas et *datetime*

```
dti = pd.to_datetime(["1/1/2018", np.datetime64("2018-01-01"),
                      datetime.datetime(2018, 1, 1)])
```

## pandas.to\_datetime

`pandas.to_datetime(arg, errors='raise', dayfirst=False, yearfirst=False, utc=None, format=None, exact=True, unit=None, infer_datetime_format=False, origin='unix', cache=True)` [source]

Convert argument to datetime.

**Parameters:** `arg : int, float, str, datetime, list, tuple, 1-d array, Series, DataFrame/dict-like`

The object to convert to a datetime.

`errors : {'ignore', 'raise', 'coerce}', default 'raise'`

- If 'raise', then invalid parsing will raise an exception.
- If 'coerce', then invalid parsing will be set as NaT.
- If 'ignore', then invalid parsing will return the input.

`dayfirst : bool, default False`

Specify a date parse order if `arg` is str or its list-likes. If True, parses dates with the day first, eg 10/11/12 is parsed as 2012-11-10. Warning: dayfirst=True is not strict, but will prefer to parse with day first (this is a known bug, based on dateutil behavior).

`yearfirst : bool, default False`

Specify a date parse order if `arg` is str or its list-likes.

- If True parses dates with the year first, eg 10/11/12 is parsed as 2010-11-12.
- If both dayfirst and yearfirst are True, yearfirst is preceded (same as dateutil).

Warning: yearfirst=True is not strict, but will prefer to parse with year first (this is a known bug, based on dateutil behavior).

`utc : bool, default None`

Return UTC DatetimeIndex if True (converting any tz-aware datetime.datetime objects as well).

`format : str, default None`

The strftime to parse time, eg "%d/%m/%Y", note that "%f" will parse all the way up to nanoseconds. See strftime documentation for more information on choices:  
<https://docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior>.

`exact : bool, True by default`

Behaves as: - If True, require an exact format match. - If False, allow the format to match anywhere in the target string.

`unit : str, default 'ns'`

The unit of the arg (D,s,ms,us,ns) denote the unit, which is an integer or float number. This will be based off the origin. Example, with unit='ms' and origin='unix' (the default), this would calculate the number of milliseconds to the unix epoch start.

`infer_datetime_format : bool, default False`

If True and no `format` is given, attempt to infer the format of the datetime strings based on the first non-NaN element, and if it can be inferred, switch to a faster method of parsing them. In some cases this can increase the parsing speed by ~5-10x.

`origin : scalar, default 'unix'`

Define the reference date. The numeric values would be parsed as number of units (defined by `unit`) since this reference date.

- If 'unix' (or POSIX) time; origin is set to 1970-01-01.
- If 'julian', unit must be 'D', and origin is set to beginning of Julian Calendar. Julian day number 0 is assigned to the day starting at noon on January 1, 4713 BC.
- If Timestamp convertible, origin is set to Timestamp identified by origin.

[https://pandas.pydata.org/pandas-docs/stable/user\\_guide/timeseries.html](https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html)

# Lorsque la date est sur plusieurs colonnes

	year	month	day
0	2015	2	4
1	2016	3	5

```
pd.to_datetime(df)
```

0	2015-02-04
1	2016-03-05

	year	month	day	temp
0	2015	2	4	15
1	2016	3	5	20

```
pd.to_datetime(df[['year', 'month', 'day']])
```

```
df['date'] =  
pd.to_datetime(df[['year', 'month', 'day']])
```

```
df.set_index('date')
```

	year	month	day	temp	date
0	2015	2	4	15	2015-02-04
1	2016	3	5	20	2016-03-05

```
df.set_index('date', inplace=True)
```

	year	month	day	temp
date				
2015-02-04	2015	2	4	15
2016-03-05	2016	3	5	20

```
df.drop(['year', 'month', 'day'], axis=1, inplace=True)
```

	temp
date	
2015-02-04	15
2016-03-05	20

## Manipulating and converting date times with timezone information

```
In [6]: dti = dti.tz_localize("UTC")  
  
In [7]: dti  
Out[7]:  
DatetimeIndex(['2018-01-01 00:00:00+00:00', '2018-01-01 01:00:00+00:00',  
               '2018-01-01 02:00:00+00:00'],  
               dtype='datetime64[ns, UTC]', freq='H')  
  
In [8]: dti.tz_convert("US/Pacific")  
Out[8]:  
DatetimeIndex(['2017-12-31 16:00:00-08:00', '2017-12-31 17:00:00-08:00',  
               '2017-12-31 18:00:00-08:00'],  
               dtype='datetime64[ns, US/Pacific]', freq='H')
```

## Resampling or converting a time series to a particular frequency

```
In [9]: idx = pd.date_range("2018-01-01", periods=5, freq="H")  
  
In [10]: ts = pd.Series(range(len(idx)), index=idx)  
  
In [11]: ts  
Out[11]:  
2018-01-01 00:00:00    0  
2018-01-01 01:00:00    1  
2018-01-01 02:00:00    2  
2018-01-01 03:00:00    3  
2018-01-01 04:00:00    4  
Freq: H, dtype: int64  
  
In [12]: ts.resample("2H").mean()  
Out[12]:  
2018-01-01 00:00:00    0.5  
2018-01-01 02:00:00    2.5  
2018-01-01 04:00:00    4.0  
Freq: 2H, dtype: float64
```

If you use dates which start with the day first (i.e. European style), you can pass the `dayfirst` flag:

```
In [45]: pd.to_datetime(["04-01-2012 10:00"], dayfirst=True)
Out[45]: DatetimeIndex(['2012-01-04 10:00:00'], dtype='datetime64[ns]', freq=None)

In [46]: pd.to_datetime(["14-01-2012", "01-14-2012"], dayfirst=True)
Out[46]: DatetimeIndex(['2012-01-14', '2012-01-14'], dtype='datetime64[ns]', freq=None)
```

### ⚠ Warning

You see in the above example that `dayfirst` isn't strict, so if a date can't be parsed with the day being first it will be parsed as if `dayfirst` were False.

## Invalid data

The default behavior, `errors='raise'`, is to raise when unparsable:

```
In [2]: pd.to_datetime(['2009/07/31', 'asd'], errors='raise')
ValueError: Unknown string format
```

Pass `errors='ignore'` to return the original input when unparsable:

```
In [56]: pd.to_datetime(["2009/07/31", "asd"], errors="ignore")
Out[56]: Index(['2009/07/31', 'asd'], dtype='object')
```

Pass `errors='coerce'` to convert unparsable data to `NaT` (not a time):

```
In [57]: pd.to_datetime(["2009/07/31", "asd"], errors="coerce")
Out[57]: DatetimeIndex(['2009-07-31', 'NaT'], dtype='datetime64[ns]', freq=None)
```

# Des formats de dates spécifiques

```
df = pd.DataFrame({'date': ['2016-6-10 20:30:0',
                            '2016-7-1 19:45:30',
                            '2013-10-12 4:5:1'],
                   'value': [2, 3, 4]})

df['date'] = pd.to_datetime(df['date'], format="%Y-%d-%m %H:%M:%S")
df
```

	date	value
0	2016-10-06 20:30:00	2
1	2016-01-07 19:45:30	3
2	2013-12-10 04:05:01	4

# Ajouter une colonne « âge »

## 5. Get the age from the date of birth

The simplest solution to get age is by subtracting year:

```
today = pd.to_datetime('today')
df['age'] = today.year - df['DoB'].dt.year

df
```

	name	DoB	year	month	day	week_of_year	day_of_week	is_leap_year	day_of_week_name	age
0	Tom	1997-08-05	1997	8	5	32	1	False	Tuesday	23
1	Andy	1996-04-28	1996	4	28	17	6	True	Sunday	24
2	Lucas	1995-12-16	1995	12	16	50	5	False	Saturday	25

<https://towardsdatascience.com/working-with-datetime-in-pandas-dataframe-663f7af6c587>

However, this is not accurate as people might haven't had their birthday this year. A more accurate solution would be to consider the birthday

```
# Year difference
today = pd.to_datetime('today')
diff_y = today.year - df['DoB'].dt.year
# Haven't had birthday
b_md = df['DoB'].apply(lambda x: (x.month,x.day) )
no_birthday = b_md > (today.month,today.day)

df['age'] = diff_y - no_birthday
df
```

	<b>name</b>	<b>DoB</b>	<b>year</b>	<b>month</b>	<b>day</b>	<b>week_of_year</b>	<b>day_of_week</b>	<b>is_leap_year</b>	<b>day_of_week_name</b>	<b>age</b>
<b>0</b>	Tom	1997-08-05	1997	8	5	32	1	False	Tuesday	23
<b>1</b>	Andy	1996-04-28	1996	4	28	17	6	True	Sunday	24
<b>2</b>	Lucas	1995-12-16	1995	12	16	50	5	False	Saturday	24

<https://towardsdatascience.com/working-with-datetime-in-pandas-dataframe-663f7af6c587>

# Sélectionner selon l'année

```
df.loc['2018']
```

	num	city
date		
2018-01-01 09:00:00	2	London
2018-01-01 09:01:00	1	London
2018-01-01 09:02:00	3	London
2018-01-01 09:03:00	3	London
2018-01-01 09:04:00	3	London
...	...	...
2018-12-31 15:56:00	4	Cambridge
2018-12-31 15:57:00	2	Cambridge
2018-12-31 15:58:00	3	Cambridge
2018-12-31 15:59:00	3	Cambridge
2018-12-31 16:00:00	2	Cambridge

439524 rows × 2 columns

Get the total num in 2018

```
df.loc['2018','num'].sum()
```

1231190

Get the total num for each city in 2018

```
df['2018'].groupby('city').sum()
```

	num
city	
Cambridge	308428
Durham	307965
London	307431
Oxford	307366

# Sélectionner une période

Select data between 2016 and 2018

```
df.loc['2016' : '2018']
```

Select data between 10 and 11 o'clock on the 2nd May 2018

```
df.loc['2018-5-2 10' : '2018-5-2 11' ]
```

Select data between 10:30 and 10:45 on the 2nd May 2018

```
df.loc['2018-5-2 10:30' : '2018-5-2 10:45' ]
```

And to select data between time, we should use `between_time()`, for example, 10:30 and 10:45

```
df.between_time('10:30','10:45')
```



# Séries temporelles

---

# DateOffsets

Alias	Description
B	business day frequency
C	custom business day frequency
D	calendar day frequency
W	weekly frequency
M	month end frequency
SM	semi-month end frequency (15th and end of month)
BM	business month end frequency
CBM	custom business month end frequency
MS	month start frequency
SMS	semi-month start frequency (1st and 15th)
BMS	business month start frequency

BQS	business quarter start frequency
A, Y	year end frequency
BA, BY	business year end frequency
AS, YS	year start frequency
BAS, BYS	business year start frequency
BH	business hour frequency
H	hourly frequency
T, min	minutely frequency
S	secondly frequency
L, ms	milliseconds
U, us	microseconds
N	nanoseconds

[https://pandas.pydata.org/pandas-docs/stable/user\\_guide/timeseries.html#offset-aliases](https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#offset-aliases)

```
In [238]: pd.date_range(start, periods=5, freq="B")
Out[238]:
DatetimeIndex(['2011-01-03', '2011-01-04', '2011-01-05', '2011-01-06',
               '2011-01-07'],
              dtype='datetime64[ns]', freq='B')

In [239]: pd.date_range(start, periods=5, freq=pd.offsets.BDay())
Out[239]:
DatetimeIndex(['2011-01-03', '2011-01-04', '2011-01-05', '2011-01-06',
               '2011-01-07'],
              dtype='datetime64[ns]', freq='B')
```

You can combine together day and intraday offsets:

```
In [240]: pd.date_range(start, periods=10, freq="2h20min")
Out[240]:
DatetimeIndex(['2011-01-01 00:00:00', '2011-01-01 02:20:00',
               '2011-01-01 04:40:00', '2011-01-01 07:00:00',
               '2011-01-01 09:20:00', '2011-01-01 11:40:00',
               '2011-01-01 14:00:00', '2011-01-01 16:20:00',
               '2011-01-01 18:40:00', '2011-01-01 21:00:00'],
              dtype='datetime64[ns]', freq='140T')

In [241]: pd.date_range(start, periods=10, freq="1D10U")
Out[241]:
DatetimeIndex(['2011-01-01 00:00:00', '2011-01-02 00:00:00.000010',
               '2011-01-03 00:00:00.000020', '2011-01-04 00:00:00.000030',
               '2011-01-05 00:00:00.000040', '2011-01-06 00:00:00.000050',
               '2011-01-07 00:00:00.000060', '2011-01-08 00:00:00.000070',
               '2011-01-09 00:00:00.000080', '2011-01-10 00:00:00.000090'],
              dtype='datetime64[ns]', freq='86400000010U')
```

# Sélectionner les lignes pour un jour (n°) donné

dataframe['colonne'].day n'est pas possible

AttributeError: 'Series' object has no attribute 'days'

pandas.Series.dt

`Series.dt()`

Accessor object for datetimelike properties of the Series values.

`rythme[rythme['creationDate'].dt.day==27].head()`

`rythme.query('creationDate.dt.day==27').head()`

	creationDate	startDate	value
24016	2020-05-27 17:59:38+01:00	2020-05-27 17:56:03+01:00	98
24017	2020-05-27 18:05:26+01:00	2020-05-27 18:02:36+01:00	85
24018	2020-05-27 18:07:07+01:00	2020-05-27 18:07:06+01:00	85
24019	2020-05-27 18:07:07+01:00	2020-05-27 18:07:07+01:00	83
24020	2020-05-27 18:07:09+01:00	2020-05-27 18:07:08+01:00	83

# Sélectionner les lignes pour un jour précis

```
mask = rythme['creationDate'].between('2022-02-27 0:00', '2022-02-27 23:59')
rythme.loc[mask].head()
```

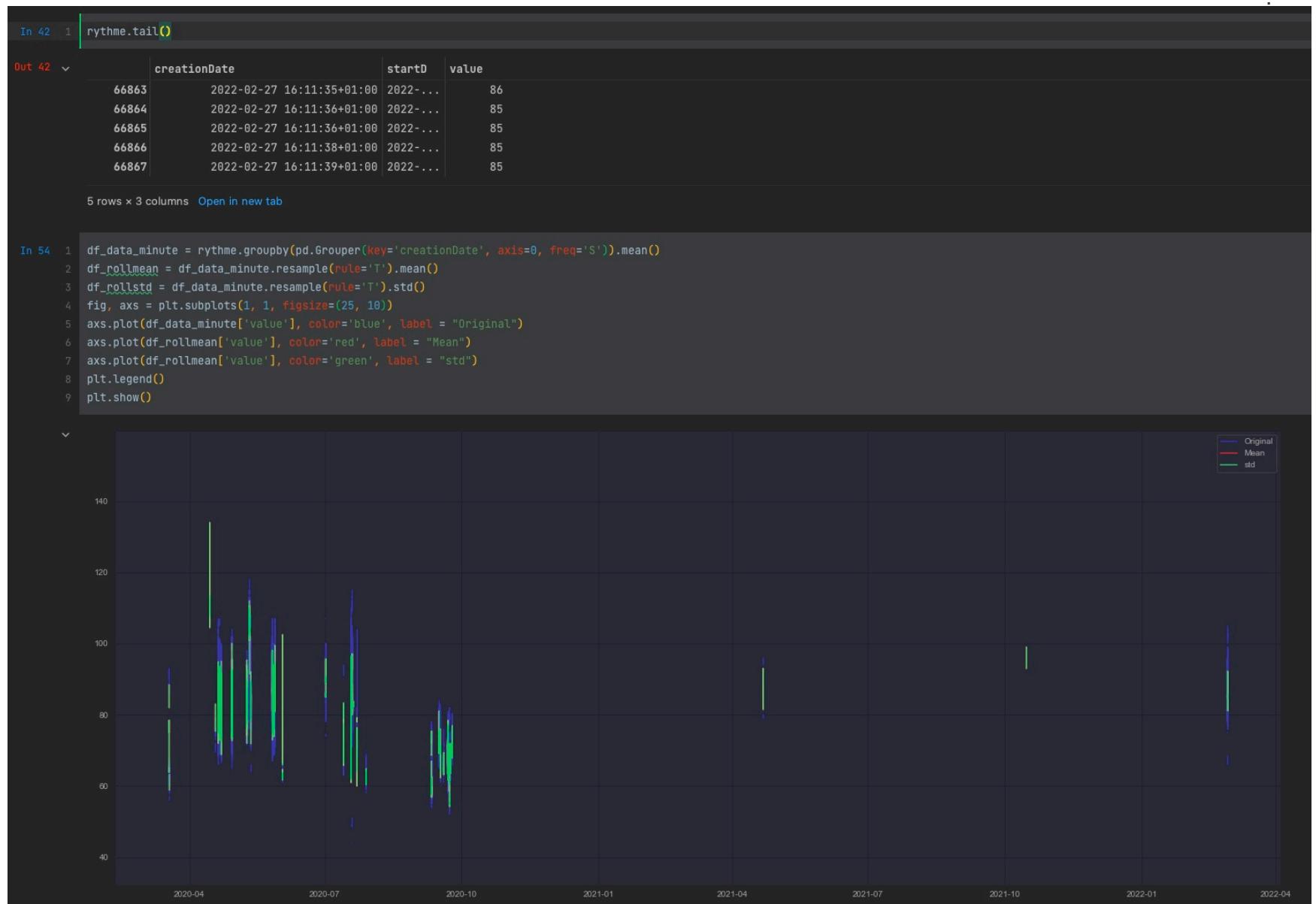
	creationDate	startDate	value
65957	2022-02-27 15:49:40+01:00	2021-04-21 15:11:51+01:00	87
65958	2022-02-27 15:52:32+01:00	2022-02-27 15:52:31+01:00	91
65959	2022-02-27 15:52:32+01:00	2022-02-27 15:52:32+01:00	90
65960	2022-02-27 15:52:34+01:00	2022-02-27 15:52:33+01:00	88
65961	2022-02-27 15:52:34+01:00	2022-02-27 15:52:34+01:00	88

mieux :

```
mask = rythme['creationDate'].between('2022-02-27', '2022-02-28', inclusive='left')
```

et si date dans index :

```
df_data_minute[df_data_minute.index.get_loc("2022-02-27")] df_data_minute.loc["2022-02-27"]
```



# Ré-échantillonnage

```
In 1  1 import pandas as pd
      2 from datetime import datetime
      3 import numpy as np

In 2  1 # création d'un intervalle temporel
      2 intervalle = pd.date_range(start='2022-05-01', end='2022-05-10', freq='H')

In 5  1 intervalle

Out 5  ▾ DatetimeIndex(['2022-05-01 00:00:00', '2022-05-01 01:00:00',
                         '2022-05-01 02:00:00', '2022-05-01 03:00:00',
                         '2022-05-01 04:00:00', '2022-05-01 05:00:00',
                         '2022-05-01 06:00:00', '2022-05-01 07:00:00',
                         '2022-05-01 08:00:00', '2022-05-01 09:00:00',
                         ...
                         '2022-05-09 15:00:00', '2022-05-09 16:00:00',
                         '2022-05-09 17:00:00', '2022-05-09 18:00:00',
                         '2022-05-09 19:00:00', '2022-05-09 20:00:00',
                         '2022-05-09 21:00:00', '2022-05-09 22:00:00',
                         '2022-05-09 23:00:00', '2022-05-10 00:00:00'],
                         dtype='datetime64[ns]', length=217, freq='H')

In 6  1 type(intervalle[0])

Out 6    pandas._libs.tslibs.timestamps.Timestamp
```

```
In 9  1 #création d'un dataframe avec des valeurs aléatoires
2 df = pd.DataFrame(np.random.randint(0,100,size=(len(intervalle))), index = intervalle, columns=["valeur"])

In 12 1 display(df)
```

	valeur
2022-05-01 00:00:00	7
2022-05-01 01:00:00	90
2022-05-01 02:00:00	54
2022-05-01 03:00:00	75
2022-05-01 04:00:00	22
...	...
2022-05-09 20:00:00	50
2022-05-09 21:00:00	47
2022-05-09 22:00:00	7
2022-05-09 23:00:00	7
2022-05-10 00:00:00	90

217 rows × 1 columns [Open in new tab](#)

```
In 13 1 df.hist()
```

```
Out 13 array([[<AxesSubplot:title={'center':'valeur'}>]], dtype=object)
```

The histogram displays the frequency distribution of values in the 'valeur' column. The x-axis represents the value range from 0 to 100, and the y-axis represents the frequency count from 0 to 25. The distribution is relatively flat, with no single dominant peak, suggesting a uniform or random distribution of values.



```
In 25  1 df.resample('D').mean()
```

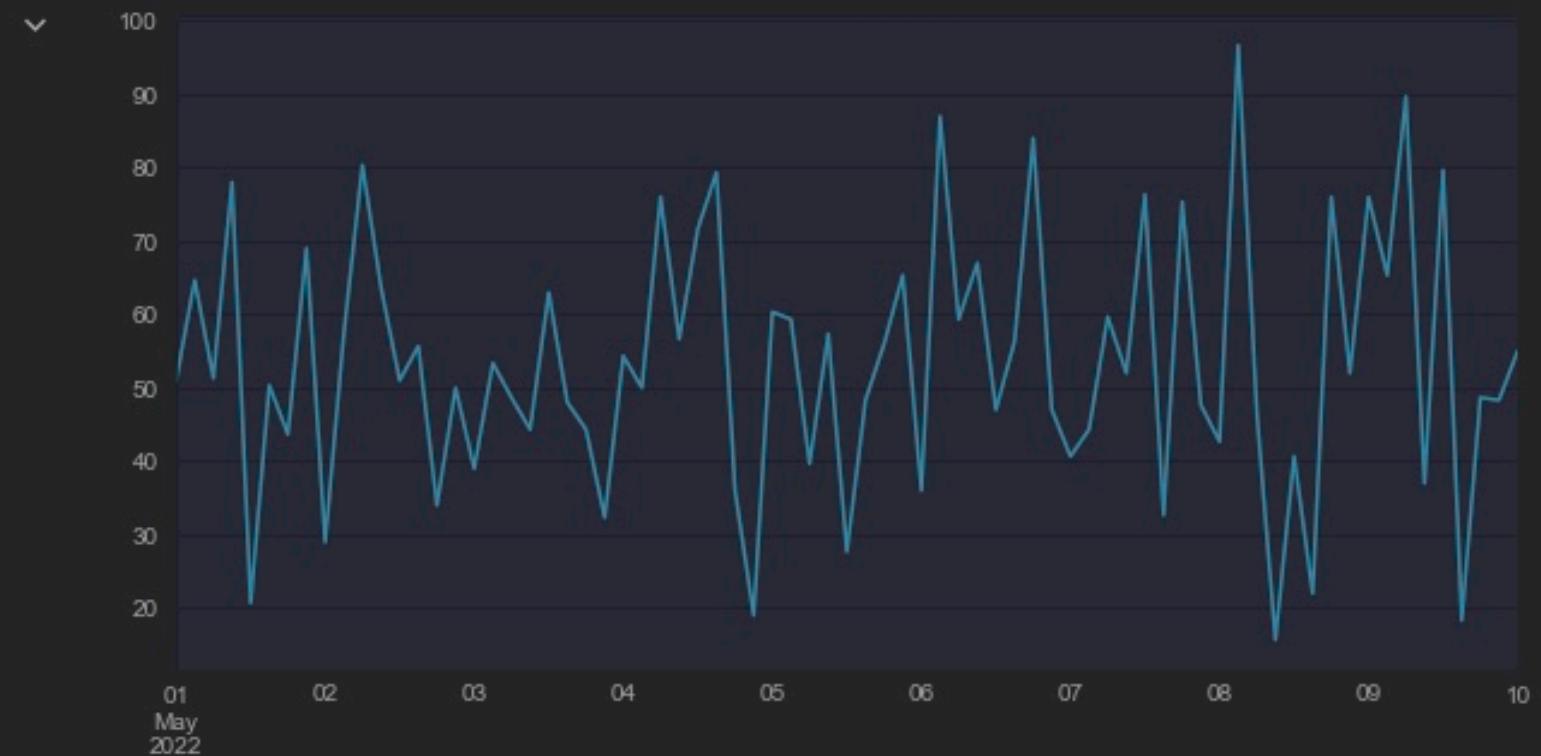
Out 25 ✓

	valeur
2022-05-01	50.625000
2022-05-02	53.375000
2022-05-03	51.875000
2022-05-04	52.708333
2022-05-05	51.208333
2022-05-06	52.541667
2022-05-07	43.375000
2022-05-08	55.125000
2022-05-09	37.000000
2022-05-10	90.000000

10 rows × 1 columns [Open in new tab](#)

```
In 52  1 df['valeur'].resample('3H').mean().plot(figsize=(10,5))
```

```
Out 52 <AxesSubplot:>
```



# Somme par jour : groupby sur index

2022-05-09 20:00:00	49	146.0
2022-05-09 21:00:00	1	60.0
2022-05-09 22:00:00	77	127.0
2022-05-09 23:00:00	67	145.0
2022-05-10 00:00:00	55	199.0

```
In 48  1 df.groupby(df.index.day)['valeur'].sum()

Out 48  ✓      valeur
          1    1286
          2    1263
          3    1119
          4    1329
          5    1242
          6    1451
          7    1286
          8    1177
          9    1389
         10    55
```

# Synthèse par fenêtre glissante : rolling

df.head()	
	valeur
2022-05-01 00:00:00	77
2022-05-01 01:00:00	60
2022-05-01 02:00:00	16
2022-05-01 03:00:00	88
2022-05-01 04:00:00	82

In 44	1	df['Somme par...'] = df.valeur.rolling(3).sum()
In 45	1	df
Out 45	v	valeur Somme par...
		2022-05-01 00:00:00 77 NaN
		2022-05-01 01:00:00 60 NaN
		2022-05-01 02:00:00 16 153.0
		2022-05-01 03:00:00 88 164.0
		2022-05-01 04:00:00 82 186.0
		2022-05-01 05:00:00 24 194.0
		2022-05-01 06:00:00 1 107.0
		2022-05-01 07:00:00 99 124.0
		2022-05-01 08:00:00 54 154.0

<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.rolling.html>

## Moyenne glissante sur 24 heures

```
In 57 1 df.valeur.rolling(24).mean().plot(figsize=(10,5))
```

```
Out 57 <AxesSubplot:>
```



# Exemple de série avec tendance, cycles et saisonnalité

```
In 59 1 df = pd.read_csv("https://raw.githubusercontent.com/AileenNielsen/TimeSeriesAnalysisWithPython/master/data/AirPassengers.csv")
2 df['Month'] = pd.to_datetime(df['Month'],infer_datetime_format=True)
3 df = df.set_index(['Month'])
4 df.head(10)
```

```
Out 59 #Passe
Month
1949-01-01    112
1949-02-01    118
1949-03-01    132
1949-04-01    129
1949-05-01    121
1949-06-01    135
1949-07-01    148
1949-08-01    148
1949-09-01    136
1949-10-01    119
```

In 62 1 df.info()

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 144 entries, 1949-01-01 to 1960-12-01
Data columns (total 1 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   #Passengers  144 non-null    int64  
dtypes: int64(1)
memory usage: 2.2 KB
```

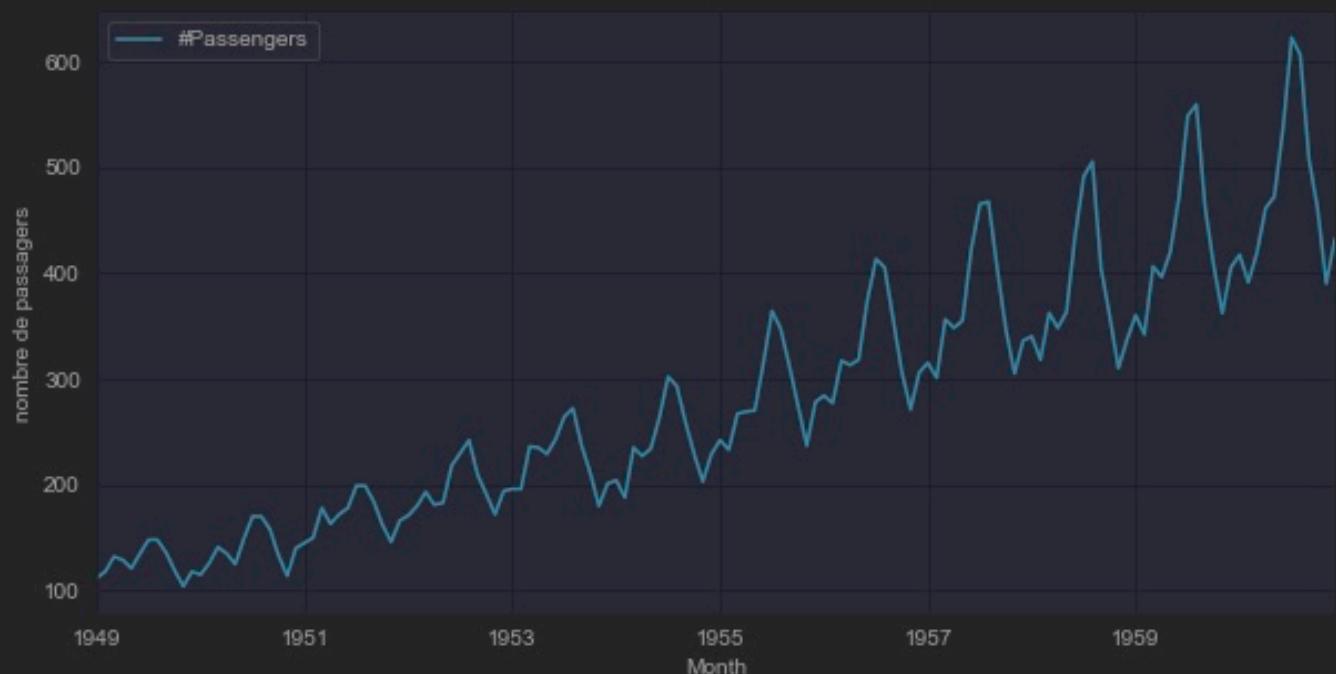
In 63 1 df.describe()

```
Out[63] 
#Passenger
count    144.0...
mean     280.2...
std      119.9...
min     104.0...
25%    180.0...
50%    265.5...
75%    360.5...
max     622.0...
```

8 rows x 1 columns [Open in new tab](#)

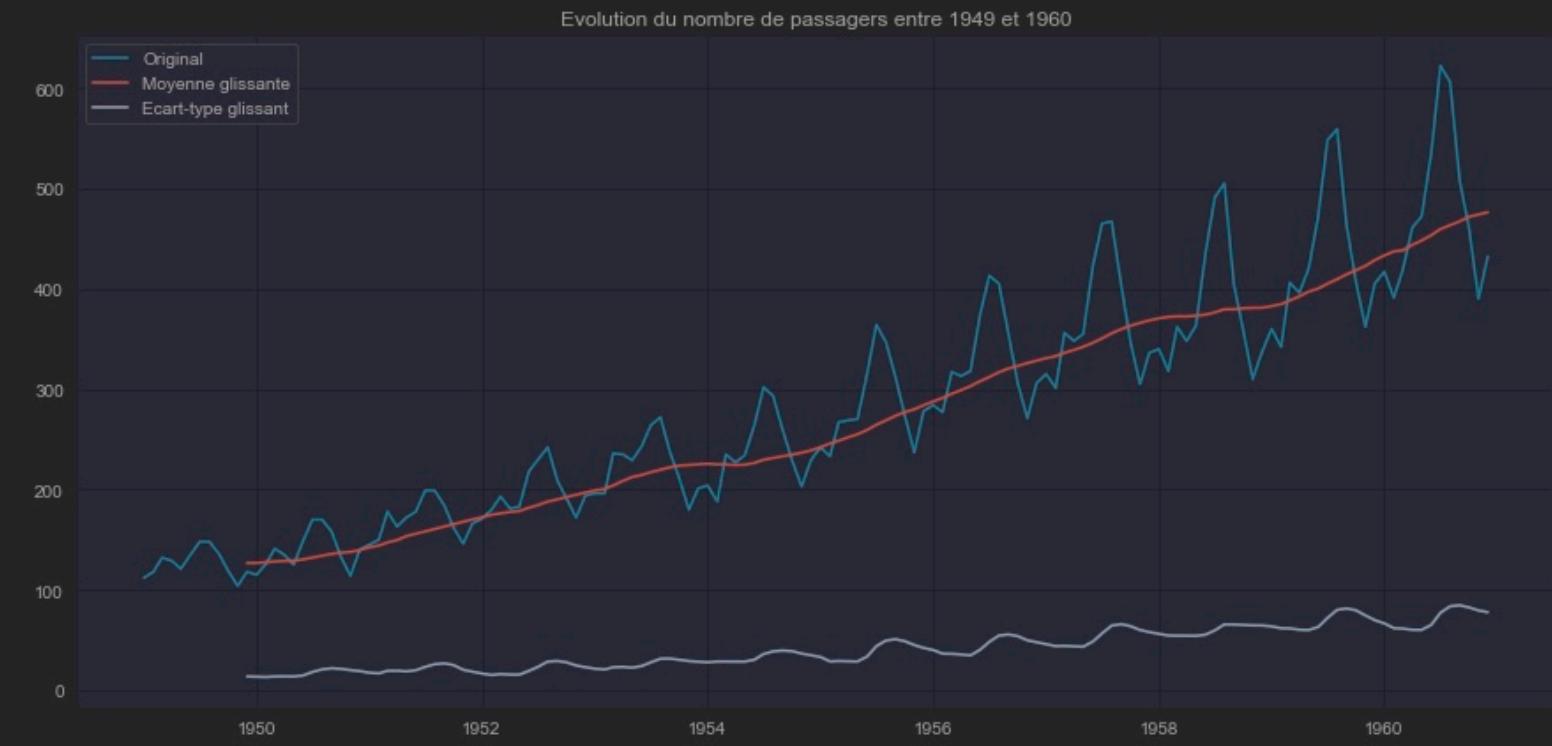
```
df.plot(figsize=(10,5), ylabel='nombre de passagers')
```

```
<AxesSubplot:xlabel='Month', ylabel='nombre de passagers'>
```



```
df["Moyenne glissante"] = df['#Passengers'].rolling(window=12).mean()
df["Ecart-type glissant"] = df['#Passengers'].rolling(window=12).std()

import matplotlib.pyplot as plt
plt.figure(figsize=(15,7))
plt.plot(df["#Passengers"], color="#379BDB", label='Original')
plt.plot(df["Moyenne glissante"], color="#D22A0D", label='Moyenne glissante')
plt.plot(df["Ecart-type glissant"], color="#142039", label='Ecart-type glissant')
plt.legend(loc='best')
plt.title('Evolution du nombre de passagers entre 1949 et 1960')
plt.show(block=False)
```



# Stationnarité ? Non !

Augmented Dickey-Fuller test

```
from statsmodels.tsa.stattools import adfuller
import pmdarima as pm
#Augmented Dickey-Fuller test:
print('Results of Dickey Fuller Test:')
dftest = adfuller(df['#Passengers'], autolag='AIC')

dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value

print(dfoutput)

Results of Dickey Fuller Test:
Test Statistic          0.815369
p-value                  0.991880
#Lags Used              13.000000
Number of Observations Used 130.000000
```

Risque de se tromper en affirmant qu'elle est stationnaire > 99,1 %

## Autoregressive integrated moving average (ARIMA)

# Essai de modélisation avec ARIMA

Prise en compte des valeurs antérieures (auto-régressif)  
+ des moyennes mobiles (lissage) avec suppression de la tendance générale

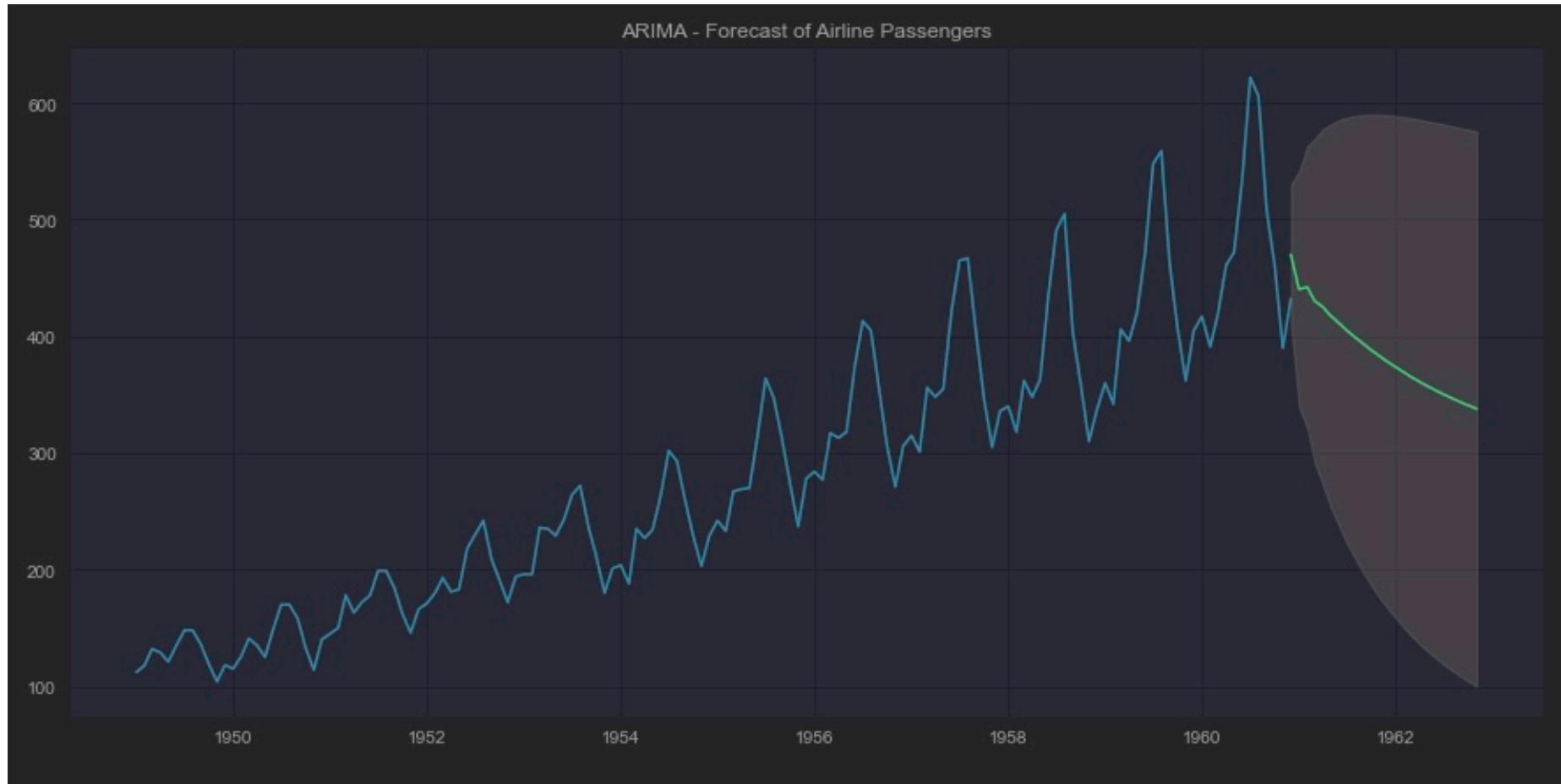
```
def forecast(ARIMA_model, periods=24):
    # Forecast
    n_periods = periods
    fitted, confint = ARIMA_model.predict(n_periods=n_periods, return_conf_int=True)
    index_of_fc = pd.date_range(df.index[-1], periods=n_periods, freq='MS')

    # make series for plotting purpose
    fitted_series = pd.Series(fitted, index=index_of_fc)
    lower_series = pd.Series(confint[:, 0], index=index_of_fc)
    upper_series = pd.Series(confint[:, 1], index=index_of_fc)

    # Plot
    plt.figure(figsize=(15,7))
    plt.plot(df["Passengers"], color="#1f76b4")
    plt.plot(fitted_series, color='darkgreen')
    plt.fill_between(lower_series.index,
                     lower_series,
                     upper_series,
                     color='k', alpha=.15)

    plt.title("ARIMA - Forecast of Airline Passengers")
    plt.show()

forecast(ARIMA_model)
```

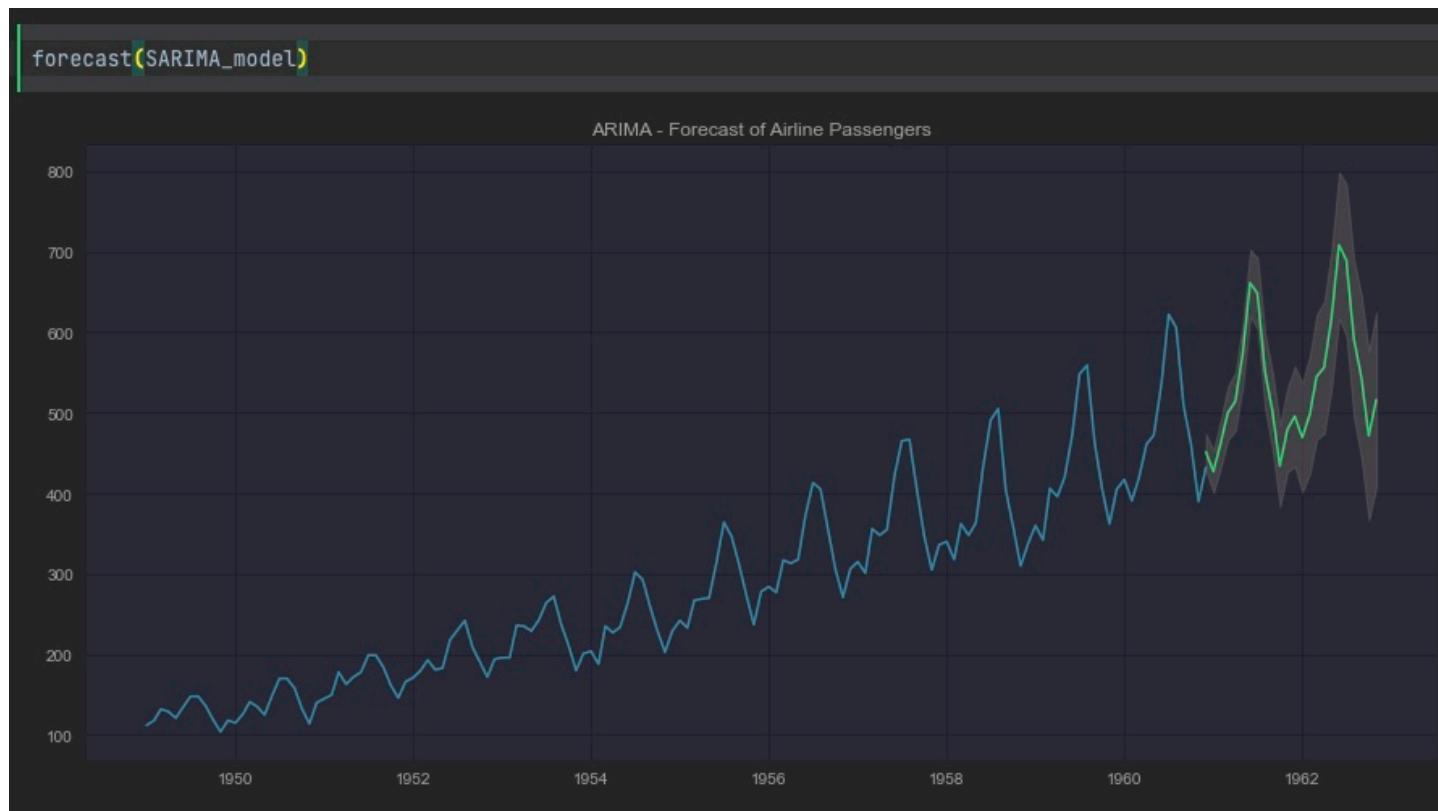


## Seasonal ARIMA

# Essai avec SARIMA

Ajout d'une composante saisonnière

```
# Seasonal - fit stepwise auto-ARIMA
SARIMA_model = pm.auto_arima(df["#Passengers"], start_p=1, start_q=1,
                             test='adf',
                             max_p=3, max_q=3,
                             m=12, #12 is the frequency of the cycle
                             start_P=0,
                             seasonal=True, #set to seasonal
                             d=None,
                             D=1, #order of the seasonal differencing
                             trace=False,
                             error_action='ignore',
                             suppress_warnings=True,
                             stepwise=True)
```





# Les données textuelles

---

Pandas, Représentations vectorielles, NLTK...

# Qu'est-ce qu'un texte ?

Des caractères, des mots, des phrases, des concepts, des documents  
Des références, des structures (syntaxe, discours), des rôles

Analyse lexicale, analyse syntaxique, analyse sémantique, pragmatique

Des arborescences, des sacs de mots, des représentations distribuées, des formes

# L'encodage des caractères

✓ Par défaut	
Occidental (ISO Latin 1)	
Occidental (Mac OS Roman)	
Unicode (UTF-8)	
Japonais (Shift JIS)	
Japonais (ISO 2022-JP)	
Japonais (EUC)	
Japonais (Shift JIS X0213)	
Chinois traditionnel (Big 5)	
Chinois traditionnel (Big 5 HKSCS)	
Chinois traditionnel (Windows, DOS)	
Coréen (ISO 2022-KR)	
Coréen (Mac OS)	
Coréen (Windows, DOS)	
Arabe (ISO 8859-6)	
Arabe (Windows)	
Hébreu (ISO 8859-8)	
Hébreu (Windows)	
Grec (ISO 8859-7)	
Grec (Windows)	
Cyrillique (ISO 8859-5)	
Cyrillique (Mac OS)	
Cyrillique (KOI8-R)	
Cyrillique (Windows)	
Ukrainien (KOI8-U)	
Thailandais (Windows, DOS)	
Chinois simplifié (GB 2312)	
Chinois simplifié (HZ GB 2312)	
Chinois (GB 18030)	
Europe centrale (ISO Latin 2)	
Europe centrale (Mac OS)	
Europe centrale (Windows Latin 2)	
Vietnamien (Windows)	
Turc (ISO Latin 5)	
Turc (Windows Latin 5)	
Europe centrale (ISO Latin 4)	
Balte (Windows)	

Dec	Hex	Char	Dec	Hex	Char
64	40	@	96	60	'
65	41	A	97	61	a
66	42	B	98	62	b
67	43	C	99	63	c
68	44	D	100	64	d
69	45	E	101	65	e
70	46	F	102	66	f
71	47	G	103	67	g
72	48	H	104	68	h
73	49	I	105	69	i
74	4A	J	106	6A	j
75	4B	K	107	6B	k
76	4C	L	108	6C	l
77	4D	M	109	6D	m
78	4E	N	110	6E	n
79	4F	O	111	6F	o
80	50	P	112	70	p
81	51	Q	113	71	q
~	~	D	~	~	~

..	@	a	«	»	—
7	00A8	00A9	00AA	00AB	00AC
	ı	ö	»	¼	¾
7	00B8	00B9	00BA	00BB	00BC
	È	É	Ê	Ë	Ì
7	00C8	00C9	00CA	00CB	00CC
	Ø	Ù	Ú	Û	Ý
7	00D8	00D9	00DA	00DB	00DC
	è	é	ë	ë	í
7	00E8	00E9	00EA	00EB	00EC
	ø	ù	ú	ü	ÿ
7	00F8	00F9	00FA	00FB	00FC
	..	..	..	..	..

ISO-Latin 1 sur 8 bits  
(ISO/CEI 8859)

<https://unicode.org/emoji/>

face-smiling		Browser	App	Goog	FB
940	á	é	í	ú	ó
950	ç	η	θ	ι	χ
960	π	ϙ	ϙ	ϙ	ϙ
970	ȝ	ȝ	ȝ	ȝ	ȝ
980	Ӯ	ӹ	ӹ	ӹ	ӹ
990	ӵ	Ӷ	Ӷ	Ӷ	Ӷ
1000	ӷ	Ӹ	Ӹ	Ӹ	Ӹ
1010	ӹ	ӻ	ӻ	ӻ	ӻ
1020	ӻ	ӻ	ӻ	ӻ	ӻ
1030	Ӵ	Ӵ	Ӵ	Ӵ	Ӵ
1040	ӵ	ӵ	ӵ	ӵ	ӵ
1050	Ӷ	Ӷ	Ӷ	Ӷ	Ӷ
1060	ӷ	ӷ	ӷ	ӷ	ӷ
1070	Ӹ	Ӹ	Ӹ	Ӹ	Ӹ
1080	ӹ	ӹ	ӹ	ӹ	ӹ
1090	ӻ	ӻ	ӻ	ӻ	ӻ
1100	ӻ	ӻ	ӻ	ӻ	ӻ
1110	ӵ	ӵ	ӵ	ӵ	ӵ
1120	ӷ	ӷ	ӷ	ӷ	ӷ

Unicode (1988) dont UTF-8  
plus de 100 000 caractères  
(dont > 3 000 émojis)  
de 1 jusqu'à 6 octets

# L'encodage des documents

Virtual Reality (2006) 10:135–147  
DOI 10.1007/s10055-006-0048-0

**ORIGINAL ARTICLE**

**The design and realization of CoViD: a system for collaborative virtual 3D design**

Wolfgang Stuerzlinger · Loufouz Zaman ·  
Andriy Pavlovych · Ji-Young Oh

Received: 14 March 2006 / Accepted: 12 May 2006 / Published online: 19 September 2006  
© Springer-Verlag London Limited 2006

**Abstract** Many important decisions in the design process are made during fairly early on, after designers have presented initial concepts. In many domains, these concepts are realized as 3D digital models. Then, they are discussed as part of the project get together and evaluate these potential solutions. Frequently, the participants in this meeting want to interactively modify the proposed 3D designs to explore the design space better. Today's systems and tools do not support this, as computer systems typically support only a single user and computer-aided design tools require significant training. This paper presents the design of a new system to facilitate a collaborative 3D design process. First, we discuss a set of guidelines which have been introduced by others and that are relevant to collaborative 3D design systems. Then, we introduce the new system, which consists of two main parts. The first part is an easy-to-use conceptual 3D design tool that can be used productively even by naive users. The tool provides novel interaction techniques that support important properties of conceptual design. The user interface is non-obtrusive, easy-to-learn, and supports rapid creation and modification of 3D models. The second part is a novel infrastructure for collaborative work, which offers an interactive table and several large interactive displays in a semi-immersive setup. It is designed to support multiple users working together. This infrastructure also includes novel pointing devices that work both as a stylus and a remote pointing device. The combination of the (modified) design tool with the collaborative infrastructure forms a new platform for collaborative virtual 3D design. Finally, we present results of a preliminary user study, which asked naive users to collaborate in a 3D design task on the new system.

**Keywords** Collaborative design · 3D design · Collaborative virtual reality

**1 Introduction**

Today's design models are created in various domains such as architecture, urban planning, all kinds of industrial design, the entertainment industry, and many engineering applications. Many of the important decisions surrounding a design are made in the initial phases, after the designer(s) have proposed a first version of the design. There, typically in a meeting, the stakeholders in the project get together and evaluate these potential solutions. Frequently, the participants in this meeting want to interactively modify the proposed designs to explore the design space better. Today's design tools and computer infrastructure do not support such activities well, as computer systems typically support only a single user and computer-aided design tools require significant training.

Traditional tools for 3D design require a large amount of training. Part of this is based on the fact that

W. Stuerzlinger (✉) · L. Zaman · A. Pavlovych  
York University, Toronto, Canada  
e-mail: wstuerz@cs.yorku.ca; wstuerz@wolfgang-stuerzlinger.com  
URL: http://www.cs.yorku.ca/~wstuerz  
URL: http://www.cs.yorku.ca/~andriy

J.-Y. Oh  
University of Arizona, Tucson, AZ, USA  
e-mail: jyoh@optics.arizona.edu

Springer

Evitons les PDF...

```
(base) [ Patrice Mac-Pro-de-Patrice ~ ] more /Users/Patrice/Downloads/ark_67375_VQC-MZ15GK10-Q/PDF/10055_2006_Article_48.pdf
%PDF-1.3<M><80-><84-><88-><90-><94-><98-><9C-><A0-><A4-><A8-><AC-><B0-><B4-><BC-><C0->
<C4-><C8-><CC-><D0-><D4-><DC-><E0-><E4-><E8-><EC-><F0-><F8-><FC-><M1 0 obj><M></CropBox>
<M2 0 R 3 0 R 4 0 R 5 0 R 6 0 R 7 0 R 8 0 R M1 0 obj><M></Rotate 0 /MediaBox><M1 0 0 59
0 R /Contents 0 0 R /Type /Page><M>><Mendobj><M2 0 obj><M></Rect><M1 367.483032 417
><M2 0 0 0 M1 0 /Type /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj><M2 0 obj><M></Rect><M1 435
R18><M></Subtype /Link /Border
<M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj><M4 0 obj><M></Rect><M1 306.142029 255.628006
k /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj><M5 0 obj><M></Rect><M1 430.356018 15
type /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj><M6 0 obj><M></Rect><M1 454
CR9><Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj><M7 0 obj><M></Rect>
<M1 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M8 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M9 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M10 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M11 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M12 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M13 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M14 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M15 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M16 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M17 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M18 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M19 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M20 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M21 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M22 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M23 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M24 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M25 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M26 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M27 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M28 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M29 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M30 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M31 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M32 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M33 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M34 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M35 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M36 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M37 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M38 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M39 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M40 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M41 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M42 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M43 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M44 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M45 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M46 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M47 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M48 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M49 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M50 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M51 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M52 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M53 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M54 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M55 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M56 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M57 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M58 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M59 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M60 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M61 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M62 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M63 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M64 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M65 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M66 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M67 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M68 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M69 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M70 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M71 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M72 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M73 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M74 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M75 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M76 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M77 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M78 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M79 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M80 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M81 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M82 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M83 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M84 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M85 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M86 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M87 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M88 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M89 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M90 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M91 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M92 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M93 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M94 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M95 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M96 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M97 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M98 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M99 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M100 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M101 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M102 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M103 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M104 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M105 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M106 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M107 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M108 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M109 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M110 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M111 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M112 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M113 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M114 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M115 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M116 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M117 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M118 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M119 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M120 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M121 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M122 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M123 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M124 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M125 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M126 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M127 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M128 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M129 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M130 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M131 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M132 0 0 0 M1 0 /Dest /CL10><M></Subtype /Link /Border><M1 0 0 0 M1 0 /Type /Annot><M>><Mendobj>
<M133 0 0 0 M1 0 /Dest /CL10><
```

# Des documents « images »

	Royaume-Uni-United Kingdom
26866	Boss, Medard : A PSYCHIATRIST DISCOVERS INDIA. — Henry A. Frey. — London : Wolff, 1965. 192, 31/6. Dt: <i>Indienfahrt eines Psychiaters</i>
26867	Boutrais, Cyprien Burns & Oates, VII
26868	Buyx, J. : CHRIST, L'Amour, Chapmar du monde
26869	Carrier, Hervé : T — Arthur J. Arrieti, 1965, 335, 30/-, F religieux
26870	Congar, Yves : E Loretz. — London Christ et dialogue
26871	— Jesus Christus — 223, 25/-, Fr: Jésus
26872	— TRADITION ANI Thomas Rainborou 90/-, Fr: La tradition
26873	Cristiani, Léon : 'London : Burns & Oates, 1965.
26874	Daniel-Rops, Henri 1870-1938. — John 48/-, Fr: L'Eglise d
26875	Desbuquois, Gustav Wells : Fowler
26876	Durmes, Jacques : London : G. Chapman, 203, 30/-, Fr: L'Amie des païens
26877	Dréze, A. : LIVING IN CHRIST, liturgy and sacraments. — F.M. Gale & Jennifer Nieh, 209, 8/6, Fr: Jésus Christ
26878	Ebeling, Gerhard : The Lo — James W. Leitch, Tim D. Von Gebet: Predigten
26879	— THE NATURE OF FAITH. — Collins, 191, 7/6. Dt: Glaubens
26880	— THEOLOGY AND PROCLAMATION: don : Collins, 186, 28/-, I
26881	ECUMENICAL DIALOGUE IN EUROPE. — Fletcher Fleet. — London : Lutterworth, 83, 12/6. Dt: Dialogue œcuménique, les rencontres œcuméniques des Dombes
26882	Emery, Pierre Y. : THE COMMUNION OF SAINTS. — D.J. & M. Watson. — London : Faith P. XIII, 256. Fr: L'unité des
26883	Emparando una enseñanza
26884	Ep., La parroquia, o
26885	Gebet, Predigten. — R. W. J. O'Hara. — London : Die Heilige Schrift und die
26886	Härting, Bernhard : THE N
26887	London : Burns & Oates, 1 Auftrag der Sakramente
26888	Héris, Charles V. : SPIRITUS
26889	London : Herder, 1965. Pneuma
26890	Jaeger, Lorenz Erzbischof ECUMENISM : THE COUNCIL
26891	London : G. Chapman, 1 Konzildebret über den Ökum
26892	Jeanne d'Arc, Sœur : V
26893	Martin, Michael : Les religions dans l'Église
26894	Jungmann, Josef A. : Tuis L. Batley. — London : Das Eucharistische Hochgebet
26895	LITURGICAL RENEWAL IN RETROSPECT AND PROSPECT. — Clifford Howell. — London : Burns & Oates, 1965, 45, 4/6. Dt: Liturgische Erneuerung
26901	LITURGY IN DEVELOPMENT. — H.J.J. Vaughan. — London : Sheed & Ward, 1965. IX, 187, 12/6. Ned: Liturgie in ontwikkeling
26902	— LITURGY IN DEVELOPMENT. — H.J.J. Vaughan. — London : Sheed & Ward, 1965. IX, 187, 12/6. Ned: Liturgie in ontwikkeling
26903	— LITURGY IN DEVELOPMENT. — H.J.J. Vaughan. — London : Sheed & Ward, 1965. IX, 187, 12/6. Ned: Liturgie in ontwikkeling
26904	Roman Catholic Church. 2d Vatican Ecumenical Council. DECLARATION ON RELIGIOUS LIBERTY. — Thomas Athill. — London : Catholic Truth Society, 19, 1/-, Lat: De libertate religiosa

# L'encodage des documents : formats ouverts et métadonnées

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0"
      xmlns:s1="http://standoff.proposal"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="https://xml-schema.delivery.istex.fr/formats/tei-istex.xsd">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title level="a" type="main" xml:lang="en">The design and realization of CoViD: a system for
          design</title>
      </titleStmt>
      <publicationStmt>
        <authority>ISTEX</authority>
        <publisher ref="https://scientific-publisher.data.istex.fr/ark:/67375/H02-SWLMH5L1-1">Springer
        <pubPlace>London</pubPlace>
        <availability>
          <licence>Springer-Verlag London Limited</licence>
          <p scheme="https://loaded-corpus.data.istex.fr/ark:/67375/XBH-3XSW68JL-F">springer</p>
        </availability>
        <date type="published" when="2006">2006</date>
      </publicationStmt>
      <notesStmt>
        <note type="content-type"
              subtype="research-article"
              source="OriginalPaper"
              scheme="https://content-type.data.istex.fr/ark:/67375/XTP-1JC4F85T-7">research-article
        <note type="publication-type"
              subtype="journal"
              scheme="https://publication-type.data.istex.fr/ark:/67375/JMC-5WTMB5N-F">journal</note>
      </notesStmt>
      <sourceDesc>
        <biblStruct>
          <analytic>
            <title level="a" type="main" xml:lang="en">The design and realization of CoViD: a syst
              design</title>
            <author role="corresp">
              <persName>
                <forename type="first">Wolfgang</forename>
                <surname>Stuerzlinger</surname>
              </persName>
              <affiliation>
                <orgName type="institution">York University</orgName>
                <address>
                  <settlement>Toronto</settlement>
                  <country key="CA" xml:lang="en">CANADA</country>
                </address>
              </affiliation>
            </author>
          </analytic>
        </biblStruct>
      </sourceDesc>
    </fileDesc>
  </teiHeader>
  <body>
    <!-- Content of the document -->
  </body>
</TEI>
```

<https://tei-c.org/release/doc/tei-p5-doc/en/html/SG.html>

## XML



Abstract Many important decisions in the design process are made during fairly early on, after designers have presented initial concepts. In many domains, these concepts are already realized as 3D digital models. Then, in a meeting, the stakeholders for the project get together and evaluate these potential solutions. Frequently, the participants in this meeting want to interactively modify the proposed 3D designs to explore the design space better. Today's systems and tools do not support this, as computer systems typically support only a single user and computer-aided design tools require significant training. This paper presents the design of a new system to facilitate a collaborative 3D design process. First, we discuss a set of guidelines which have been introduced by others and that are relevant to collaborative 3D design systems. Then, we introduce the new system, which consists of two main parts. The first part is an easy-to-use conceptual 3D design tool that can be used productively even by naive users. The tool provides novel interaction techniques that support important properties of conceptual design. The user interface is non-obtrusive, easy-to-learn, and supports rapid creation and modification of 3D models. The second part is a novel infrastructure for collaborative work, which of a semi-immersive

W. Stuerzlinger (&amp;#038;) L. Zaman A. Pavlovych  
York University, Toronto, Canada  
URL: <http://www.cs.yorku.ca/~wolfgang>  
URL: <http://www.cs.yorku.ca/~zaman>  
URL: <http://www.cs.yorku.ca/~andriy>  
J.-Y. Oh  
University of Arizona, Tucson, AZ, USA  
e-mail: [jyoh@optics.arizona.edu](mailto:jyoh@optics.arizona.edu)

setup. It is designed to support multiple users working together. This infrastructure also includes novel pointing devices that work both as a stylus and a remote pointing device. collaborative infrastructure forms a new platform for collaborative virtual 3D design. Then, we present against the guidelines for collaborative 3D design. Finally, we present res which asked naive users to collaborate in a 3D design task on the new system.  
Keywords Collaborative design 3D design  
Collaborative virtual reality

# D'autres formats ouverts

```
"jour";"nomReg";"numReg";"incid_rea"
2020-03-19;"Auvergne-Rhône-Alpes";84;44
2020-03-19;"Bourgogne-Franche-Comté";27;33
2020-03-19;"Bretagne";53;8
2020-03-19;"Centre-Val de Loire";24;6
2020-03-19;"Corse";94;11
2020-03-19;"Grand-Est";44;69
2020-03-19;"Guadeloupe";1;0
2020-03-19;"Guyane";3;0
2020-03-19;"Hauts-de-France";32;37
2020-03-19;"Île-de-France";11;151
2020-03-19;"La Réunion";4;0
2020-03-19;"Martinique";2;0
2020-03-19;"Mayotte";6;0
2020-03-19;"Normandie";28;7
2020-03-19;"Nouvelle-Aquitaine";75;7
2020-03-19;"Occitanie";76;29
2020-03-19;"Pays de la Loire";52;11
2020-03-19;"Provence-Alpes-Côte d'Azur";93;25
2020-03-20;"Auvergne-Rhône-Alpes";84;16
2020-03-20;"Bourgogne-Franche-Comté";27;9
2020-03-20;"Bretagne";53;2
2020-03-20;"Centre-Val de Loire";24;4
2020-03-20;"Corse";94;0
2020-03-20;"Grand-Est";44;45
```

CSV

[https://en.wikipedia.org/wiki/Comma-separated\\_values](https://en.wikipedia.org/wiki/Comma-separated_values)

```
"header": [
    "title": "The JSON example",
    "descriptionText": "This is some title text."
],
"content": [
    {
        "title": "The content example text",
        "elements": [
            {
                "title": "The first element",
                "mainText": "First element main text",
                "additionalText": "First element additional te
            },
            {
                "title": "The second element",
                "mainText": "Second element main text",
                "additionalText": "Second element additional
            }
        ]
    }
]
```

JSON

<https://en.wikipedia.org/wiki/JSON>

```
---
receipt: Oz-Ware Purchase Invoice
date: 2012-08-06
customer:
    first_name: Dorothy
    family_name: Gale

items:
    - part_no: A4786
        descrip: Water Bucket (Filled)
        price: 1.47
        quantity: 4

    - part_no: E1628
        descrip: High Heeled "Ruby" Slippers
        size: 8
        price: 133.7
        quantity: 1

bill-to: *id001
street: |
    123 Tornado Alley
    Suite 16
city: East Centerville
state: KS

ship-to: *id001
specialDelivery: >
    Follow the Yellow Brick
```

YAML

<https://en.wikipedia.org/wiki/YAML>

# Balises orientées affichage vs. annotations sémantiques

## BACHELARD, Gaston

1975 *La formation de l'esprit scientifique* (Paris, Librairie philosophique J. Vrin)  
[1<sup>re</sup> éd. 1938].

## BARTHES, Roland

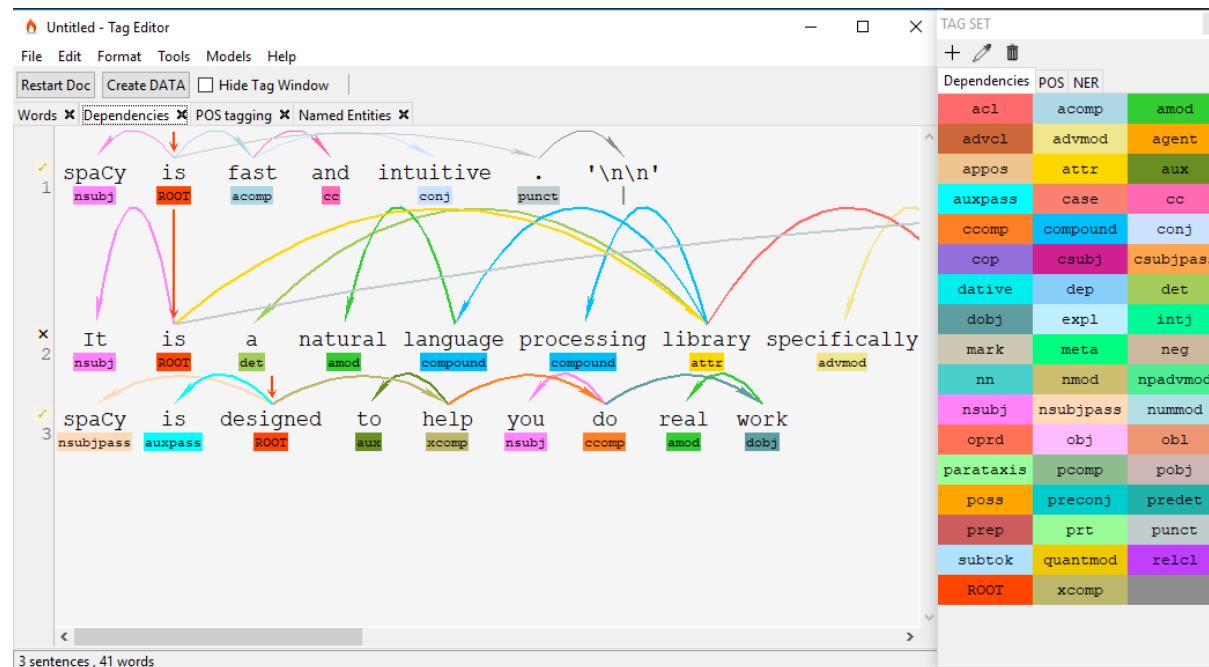
1957 *Mythologies* (Paris, Le Seuil).

## BENVENISTE, Émile

1966 De la subjectivité dans le langage, in É. Benveniste (dir.), *Problèmes de linguistique générale*, I (Paris, Gallimard) : 258-266.

1974 La forme et le sens dans le langage, in É. Benveniste (dir.), *Problèmes de linguistique générale*, II (Paris, Gallimard) : 215-240.

```
<bibl><hi rend="bold">C</hi><hi rend="small-caps bold">oupry</hi><hi rend="bold">, François<lb/></hi>1999 <hi rend="bold">C</hi><hi rend="small-caps bold">rutz</hi><hi rend="bold"> S</hi><hi rend="small-caps bold">escola</hi><hi rend="bold">, Philippe<lb/></hi>1994 Rétri<lb/><hi rend="bold">D</hi><hi rend="small-caps bold">ubois</hi><hi rend="bold">, Jacques<lb/></hi>1992 <hi rend="bold">D</hi><hi rend="small-caps bold">Etnoloska sticiska</hi><hi xml:lang="en" rend="italic bold">Etnoloska sticiska</hi><hi xml:lang="en" rend="italic bold"><lb/></hi><hi xml:lang="en" rend="italic bold">1997 </hi><hi xml:lang="en" rend="italic">Prejudices and stereotypes in the social sciences and humanities</hi><hi xml:lang="en" rend="italic">, 5 et 7 (Ljubljana, Jezernik, Bozidr Ed.).</hi></bibl><bibl><hi rend="bold">F</hi><hi rend="small-caps bold">erney</hi><hi rend="bold">, Alice<lb/></hi>1997 <hi rend="bold">G</hi><hi rend="small-caps bold">eertz</hi><hi rend="bold">, Clifford<lb/></hi>1986 Comme<lb/><hi xml:lang="en" rend="bold">G</hi><hi xml:lang="en" rend="small-caps bold">ilman</hi><hi xml:lang="en" rend="bold">, Sander L.<lb/></hi><hi xml:lang="en" rend="bold">1985 </hi><hi xml:lang="en" rend="italic">Difference and pathology, stereotypes of sexuality, race and madness</hi><hi xml:lang="en" rend="italic"> (Ithaca, Cornell University Press).</hi></bibl>
```



<https://github.com/d5555/TagEditor>

File Edit Format Tools Models Help

Back to Text Create DATA  Hide Tag Window  Assign/unassign all

Words Dependencies POS tagging Categories

1 That was great!  
✗ **POSITIVE: True**

2 This is terrible.  
✗ **POSITIVE: False**

3 He likes to travel to Hawaii.  
✗ **TRAVEL: True** ✗ **POSITIVE: True**

4 The deal was a success, and everyone was happy.  
✗ **BUSINESS: True** ✗ **POSITIVE: True**

**Create DATA**

**DATA options**

- Include named entities
- Include dependencies
- Include POS tags
- Include CATS
- Add words
- Convert to JSON

Score: **True** **False**

	Dependencies	POS	CATS
POSITIVE			TRAVEL
FASHION			FOOD

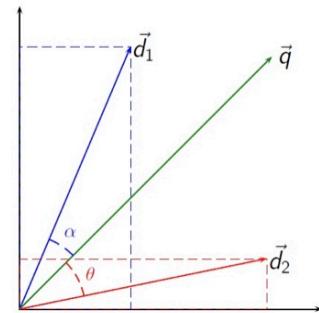
Ok Cancel

# Des formes, des distributions et des vecteurs

cette série  
d'un certain  
is revenir  
nière fois  
se : Per

Des formes rassemblées,  
des sacs (de lettres, de mots)

$$P(\text{Classe}|(w_1, w_2, \dots, w_{T-1}, w_T))$$



Des vecteurs

$$\text{Similarité}(d_1, d_2) \approx \vec{d}_1 \cdot \vec{d}_2$$

$$\text{Similarité}(d_1, d_2) \approx \cos(\vec{d}_1, \vec{d}_2)$$

Mot	Probabilité

Des modèles de langues

$$P(w_1, w_2, \dots, w_{T-1}, w_T) = \prod_{t=1}^T P(w_t|w_{t-1}, w_{t-2}, \dots, w_1)$$

the	cat	sat	on	the	mat	$P(w_1)$
the	cat	sat	on	the	mat	$P(w_2 w_1)$
the	cat	sat	on	the	mat	$P(w_3 w_2, w_1)$
the	cat	sat	on	the	mat	$P(w_4 w_3, w_2, w_1)$
the	cat	sat	on	the	mat	$P(w_5 w_4, w_3, w_2, w_1)$
the	cat	sat	on	the	mat	$P(w_6 w_5, w_4, w_3, w_2, w_1)$

Slide Credit: Piotr Mirowski

# Un exemple de modèle de langue

**Google Books**

Ngram Viewer Exports

The Google Books Ngram Viewer is optimized for quick inquiries into the usage of small sets of phrases. If you're interested prefer to download a portion of the corpora yourself. Or all of it, if you have the bandwidth and space. We're happy to oblige.

These datasets were generated in February 2020 (version 3), July 2012 (Version 2) and July 2009 (Version 1); we will update versions will have distinct and persistent version identifiers (20200217, 20120701 and 20090715 for the current sets).

Each of the numbered links below will directly download a fragment of the corpus. In Version 2 the ngrams are grouped alpha: Version 1 the ngrams are partitioned into files of equal size. In addition, for each corpus we provide a file named `total_count` that make up the corpus. This file is useful for computing the relative frequencies of ngrams.

A summary of how the corpora were constructed can be found [here](#). We explain it in greater depth [here](#) (Version 2) and [here](#) appear over 40 times across the corpus. That's why the sum of the 1-gram occurrences in any given corpus is smaller than the total number of tokens in the corpus.

**File format:** Each of the files below is compressed *tab*-separated data. In Version 2 each line has the following format:

```
ngram TAB year TAB match_count TAB volume_count NEWLINE
```

**French**

**Version 20200217**

- 1-grams
- 2-grams
- 3-grams
- 4-grams
- 5-grams
- Dependencies

**Version 20120701**

[total\\_counts](#)

Language	#Volumes	#Tokens
English	4,541,627	468,491,999,592
Spanish	854,649	83,967,471,303
French	792,118	102,174,681,393
German	657,991	64,784,628,286
Russian	591,310	67,137,666,353
Italian	305,763	40,288,810,817
Chinese	302,652	26,859,461,025
Hebrew	70,636	8,172,543,728

Table 1: Number of volumes and tokens for each language in our corpus. The total collection contains more than 6% of all books ever published.

<https://books.google.com/ngrams>

[Reading Comprehension](#)

[Visual Question Answering](#)

[Annotate a sentence](#)

[Named Entity Recognition](#)

[Open Information Extraction](#)

[Sentiment Analysis](#)

[Dependency Parsing](#)

[Constituency Parsing](#)

[Semantic Role Labeling](#)

[Annotate a passage](#)

[Coreference Resolution](#)

[Generate a passage](#)

[Language Modeling](#)

[Masked Language Modeling](#)

[Compare two sentences](#)

[Textual Entailment](#)

## Language Modeling

Language modeling is the task of determining the probability of a given sequence of words occurring in a sentence.

Model

### GPT2-based Next Token Language Model

This is the public 345M parameter OpenAI GPT-2 language model for generating sentences. The model embeds some input tokens, contextualizes them, then predicts the next word, computing a loss against known target. If BeamSearch is given, this model will predict a sequence of next tokens.

[TaskDemo](#) [Model Card](#)

#### Example Inputs

The doctor ran to the emergency room to see the

#### Sentence

The doctor ran to the emergency room to see the

[Run Model](#)

## Model Output

[Share](#)

Prediction	Score
The doctor ran to the emergency room to see the <b>patient.</b> ←" ...	99,1 %
The doctor ran to the emergency room to see the <b>girl. She was crying</b> ...	0,6 %
The doctor ran to the emergency room to see the <b>injured victim.</b> ←"	0,2 %

## Approches statistiques, probabilistes Apprentissage automatique

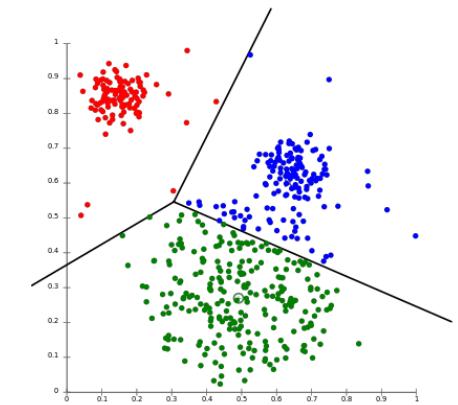
Quels sont les mots caractéristiques d'un groupe de documents ?

$$Z_{\text{score}}(t_{ij}) = \frac{\text{tf}_{rij} - \text{mean}_i}{\text{sdi}}$$

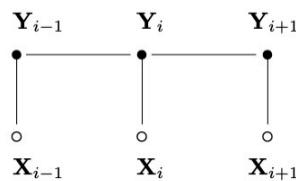
positive	Z_score	negative	Z_score	Neutral	Z_score
Love	14.31	Not	13.99	Htpbit	6.44
Good	14.01	Fuck	12.97	Htpfb	4.56
Happy	12.30	Don't	10.97	Htpbnd	3.78
Great	11.10	Shit	8.99	Intern	3.58
Excite	10.35	Bad	8.40	Nov	3.45
Best	9.24	Hate	8.29	Htpdvr	3.40
Thank	9.21	Sad	8.28	Open	3.30
Hope	8.24	Sorry	8.11	Live	3.28
Cant	8.10	Cancel	7.53	Cloud	3.28
Wait	8.05	stupid	6.83	begin	3.17

Table1. The first ten terms having the highest Z\_score in each class

Similarités, corrélations

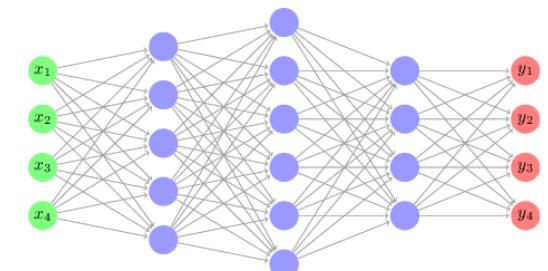
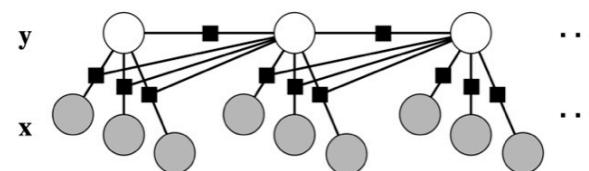


Quelles relations significatives à partir des seules formes observées ?

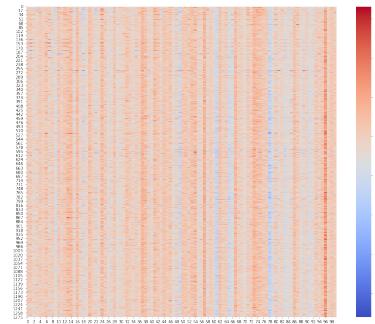


$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x})} \exp \left( \sum_j \lambda_j F_j(\mathbf{y}, \mathbf{x}) \right).$$

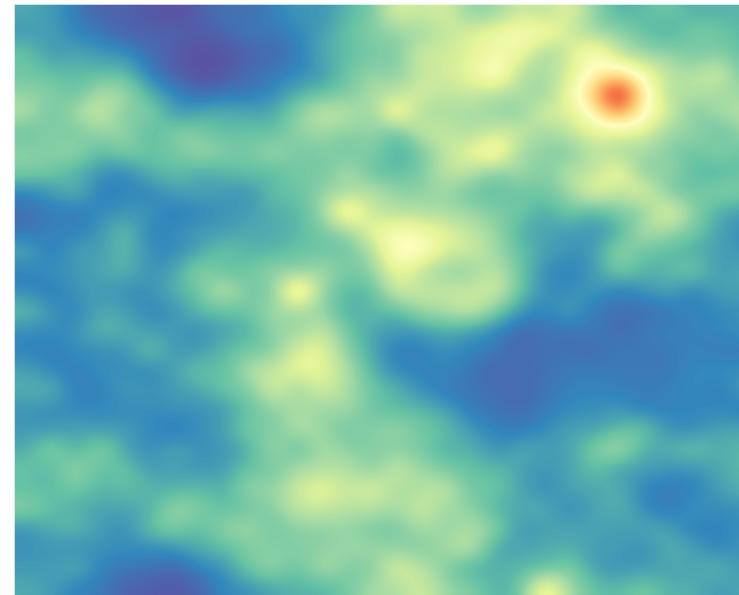
Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data  
Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001)



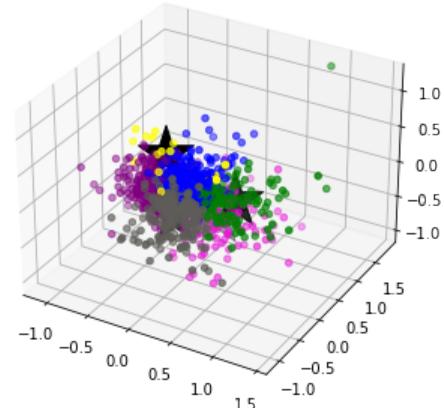
# Représentation de documents par projection



Vectorisation dans des espaces continus  
(Doc2Vec) par plongements lexicaux

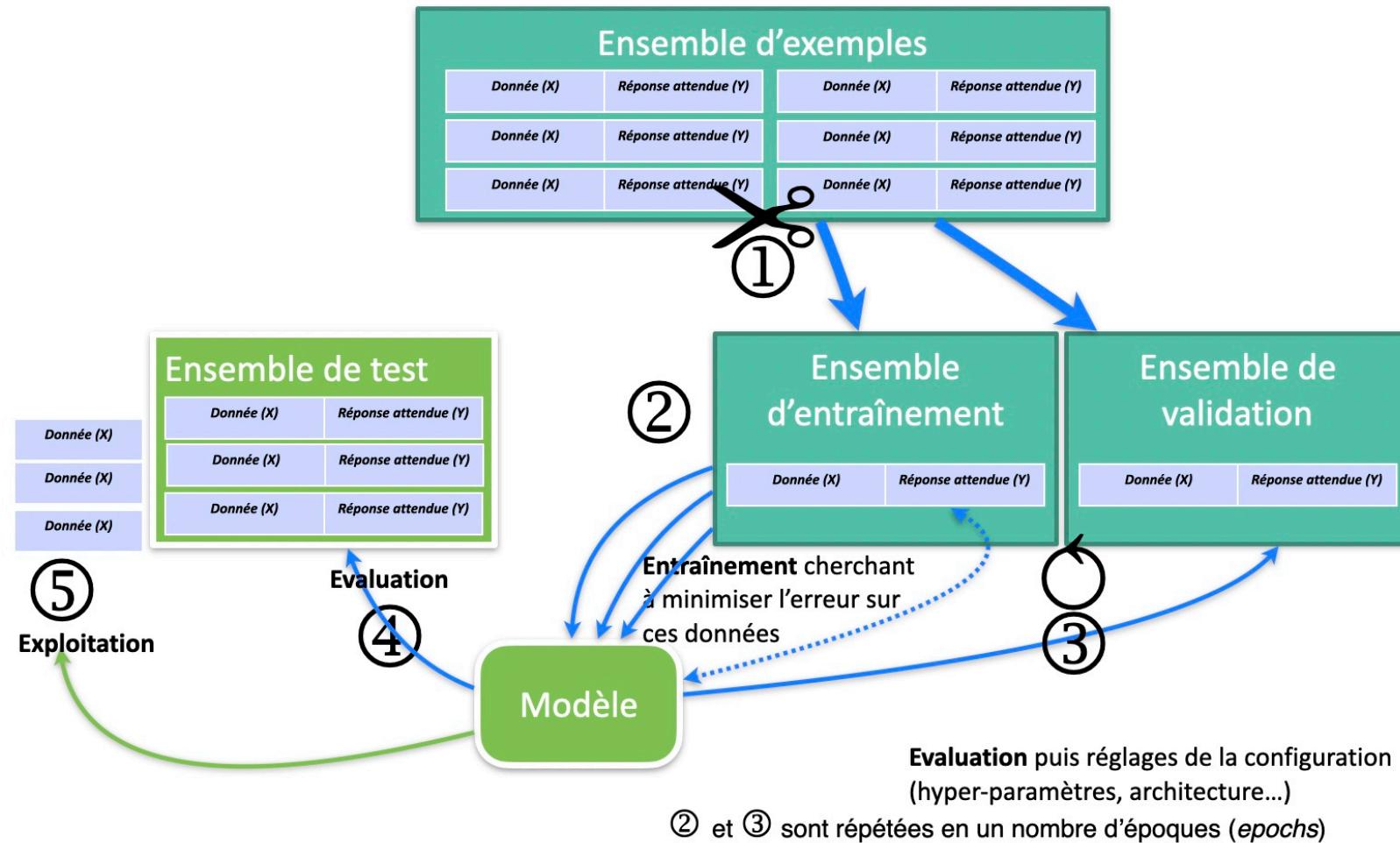


Cartes auto-organisées (SOM)



Analyse en Composantes Principales

# Apprentissage automatique, modèle, évaluation



# Documents, lexique et *dataframes* : des représentations

	méta-donnée 1	méta-donnée 2	...
document 1			
document 2			
document 3			
...			

	mot 1	mot 2	...
document 1			
document 2			
document 3			
...			

sacs de mots

	dim1	dim 2	...
mot 1			
mot 2			
mot 3			
...			

espaces de représentation réduits

# Le module *RegEx*

[https://www.w3schools.com/python/python\\_regex.asp](https://www.w3schools.com/python/python_regex.asp)

## Example

Print the part of the string where there was a match.

The regular expression looks for any words that starts with an upper case "S":

```
import re

txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.group())
```

## Example

Search the string to see if it starts with "The" and ends with "Spain":

```
import re

txt = "The rain in Spain"
x = re.search("^The.*Spain$", txt)
```

## RegEx Functions

The `re` module offers a set of functions that allows us to search a string for a match:

Function	Description
<code>findall</code>	Returns a list containing all matches
<code>search</code>	Returns a <u>Match object</u> if there is a match anywhere in the string
<code>split</code>	Returns a list where the string has been split at each match
<code>sub</code>	Replaces one or many matches with a string

## Example

Replace every white-space character with the number 9:

```
import re

txt = "The rain in Spain"
x = re.sub("\s", "9", txt)
print(x)
```

## Example

Print a list of all matches:

```
import re

txt = "The rain in Spain"
x = re.findall("ai", txt)
print(x)
```

## Metacharacters

Metacharacters are characters with a special meaning:

Character	Description	Example
[]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
^	Starts with	"^hello"
\$	Ends with	"world\$"
*	Zero or more occurrences	"aix**"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
	Either or	"falls stays"
()	Capture and group	

## Special Sequences

A special sequence is a \ followed by one of the characters in the list below, and has a special meaning:

Character	Description	Example
\A	Returns a match if the specified characters are at the beginning of the string	"\AThe"
\b	Returns a match where the specified characters are at the beginning or at the end of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\bain" r"ain\b"
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word (the "r" in the beginning is making sure that the string is being treated as a "raw string")	r"\Bain" r"ain\B"
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"
\Z	Returns a match if the specified characters are at the end of the string	"Spain\Z"

## Sets

A set is a set of characters inside a pair of square brackets `[]` with a special meaning:

Set	Description
<code>[arn]</code>	Returns a match where one of the specified characters ( <code>a</code> , <code>r</code> , or <code>n</code> ) are present
<code>[a-n]</code>	Returns a match for any lower case character, alphabetically between <code>a</code> and <code>n</code>
<code>[^arn]</code>	Returns a match for any character EXCEPT <code>a</code> , <code>r</code> , and <code>n</code>
<code>[0123]</code>	Returns a match where any of the specified digits ( <code>0</code> , <code>1</code> , <code>2</code> , or <code>3</code> ) are present
<code>[0-9]</code>	Returns a match for any digit between <code>0</code> and <code>9</code>
<code>[0-5][0-9]</code>	Returns a match for any two-digit numbers from <code>00</code> and <code>59</code>
<code>[a-zA-Z]</code>	Returns a match for any character alphabetically between <code>a</code> and <code>z</code> , lower case OR upper case
<code>[+]</code>	In sets, <code>+</code> , <code>*</code> , <code>.</code> , <code> </code> , <code>(</code> , <code>)</code> , <code>\$</code> , <code>{</code> has no special meaning, so <code>[+]</code> means: return a match for any <code>+</code> character in the string

## NLTK Corpora

NLTK has built-in support for dozens of corpora and trained models, as listed below. To use these within NLTK we recommend that you consult the README file included with each corpus for further information.

## NLTK 3.5 documentation

[NEXT](#) | [MODULES](#) | [INDEX](#)

## Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](#) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active [discussion forum](#).

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called "a wonderful tool for teaching, and working in, computational linguistics using Python," and "an amazing library to play with natural language."

<http://www.nltk.org>

*icode Version 7.0.0 character properties in Perl* [[download](#) | [source](#)]  
00266; author: ; copyright: ; license: ;

*gner (Sultan et al. 2015) subset of the Paraphrase Database*. [[download](#) | [source](#)]  
11; author: ; copyright: ; license: Creative Commons Attribution 3.0 Unported (CC-BY);  
[download](#) | [source](#)]  
; author: Jan Strunk; copyright: ; license: ;

*or de Sufixos da Lingua Portuguesa* [[download](#) | [source](#)]  
or: Viviane Moreira Orengo (vmoren@inf.ufrgs.br) and Christian Huyck; copyright: ; license: ;  
[download](#) | [source](#)]  
0510; author: ; copyright: ; license: ;

*er (Maximum entropy)* [[download](#) | [source](#)]  
size: 13404747; author: ; copyright: ; license: ;  
[download](#) | [source](#)]  
0961490; author: ; copyright: ; license: ;  
[download](#) | [source](#)]  
ze: 24516205; author: ; copyright: ; license: ;

*oad* [[download](#) | [source](#)]  
e: 49396025; author: ; copyright: ; license: ;  
11. *Evaluation data from WMT15* [[download](#) | [source](#)]  
id: wmt15\_eval; size: 383096; author: ; copyright: ; license: ;  
12. *Grammars for Spanish* [[download](#) | [source](#)]  
id: spanish\_grammars; size: 4047; author: Kepa Sarasola; copyright: ; license: ;  
13. *Sample Grammars* [[download](#) | [source](#)]  
id: sample\_grammars; size: 20293; author: ; copyright: ; license: ;  
14. *Large context-free and feature-based grammars for parser comparison* [[download](#) | [source](#)]  
id: large\_grammars; size: 283747; author: ; copyright: ; license: See the individual grammar files;  
15. *Grammars from NLTK Book* [[download](#) | [source](#)]  
id: book\_grammars; size: 9103; author: Ewan Klein; copyright: ; license: ;  
16. *Grammars for Basque* [[download](#) | [source](#)]  
id: basque\_grammars; size: 4704; author: Kepa Sarasola; copyright: ; license: ;

<http://www.nltk.org>

## Some simple things you can do with NLTK

Tokenize and tag some text:

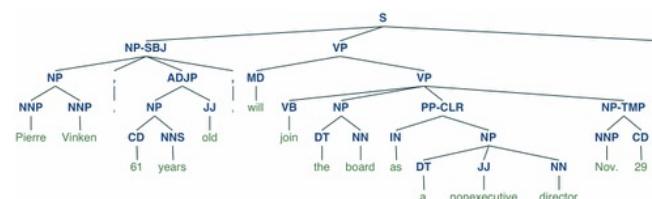
```
>>> import nltk
>>> sentence = """At eight o'clock on Thursday morning
... Arthur didn't feel very good."""
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good', '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Identify named entities:

```
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'),
('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'),
Tree('PERSON', [('Arthur', 'NNP')]),
('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'),
('very', 'RB'), ('good', 'JJ'), ('.', '.')])
```

Display a parse tree:

```
>>> from nltk.corpus import treebank
>>> t = treebank.parsed_sents('wsj_0001.mrg')[0]
>>> t.draw()
```



spaCy

Out now: spaCy v3.0

USAGE MODELS API UNIVERSE

19,524

Search docs

# Industrial-Strength Natural Language Processing

IN PYTHON

## Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive.

GET STARTED

## Blazing fast

spaCy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. If your application needs to process entire web dumps, spaCy is the library you want to be using.

FACTS & FIGURES

## Awesome ecosystem

In the five years since its release, spaCy has become an industry standard with a huge ecosystem. Choose from a variety of plugins, integrate with your machine learning stack and build custom components and workflows.

READ MORE

Edit the code & try spaCy spaCy v3.0 · Python 3 · via Binder

```
# pip install -U spacy
# python -m spacy download en_core_web_sm
import spacy

# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
        "Google in 2007, few people outside of the company took him "
        "seriously. "I can tell you very senior CEOs of major American "
        "car companies would shake my hand and turn away because I wasn't "
        "worth talking to," said Thrun, in an interview with Recode earlier "
        "this week.")
doc = nlp(text)

# Analyze syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])

# Find named entities, phrases and concepts
for entity in doc.ents:
    print(entity.text, entity.label_)

RUN
```

Noun phrases: ['Sebastian Thrun', 'self-driving cars', 'Google', 'few people', 'the company', 'him', 'I', 'you', 'very senior CEOs', 'major American car companies', 'my hand', 'I', 'Thrun', 'an interview', 'Recode']  
Verbs: ['start', 'work', 'drive', 'take', 'tell', 'shake', 'turn', 'be', 'talk', 'say']  
Sebastian Thrun PERSON  
2007 DATE  
American NORP  
Thrun PERSON  
Recode PERSON  
earlier this week DATE

## Features

- ✓ Support for **69+ languages**
- ✓ **58 trained pipelines** for 18 languages
- ✓ Multi-task learning with pretrained **transformers** like BERT
- ✓ Pretrained **word vectors**
- ✓ State-of-the-art speed
- ✓ Production-ready **training system**
- ✓ Linguistically-motivated **tokenization**
- ✓ Components for **named entity** recognition, part-of-speech tagging, dependency parsing, sentence segmentation, **text classification**, lemmatization, morphological analysis, entity linking and more
- ✓ Easily extensible with **custom components** and attributes
- ✓ Support for custom models in **PyTorch**, **TensorFlow** and other frameworks
- ✓ Built in **visualizers** for syntax and NER
- ✓ Easy **model packaging**, deployment and workflow management
- ✓ Robust, rigorously evaluated accuracy

<https://spacy.io/>

# Features

In the documentation, you'll come across mentions of spaCy's features and capabilities. Some of them refer to linguistic concepts, while others are related to more general machine learning functionality.

NAME	DESCRIPTION
<b>Tokenization</b>	Segmenting text into words, punctuations marks etc.
<b>Part-of-speech (POS) Tagging</b>	Assigning word types to tokens, like verb or noun.
<b>Dependency Parsing</b>	Assigning syntactic dependency labels, describing the relations between individual tokens, like subject or object.
<b>Lemmatization</b>	Assigning the base forms of words. For example, the lemma of "was" is "be", and the lemma of "rats" is "rat".
<b>Sentence Boundary Detection (SBD)</b>	Finding and segmenting individual sentences.
<b>Named Entity Recognition (NER)</b>	Labelling named "real-world" objects, like persons, companies or locations.
<b>Entity Linking (EL)</b>	Disambiguating textual entities to unique identifiers in a knowledge base.
<b>Similarity</b>	Comparing words, text spans and documents and how similar they are to each other.
<b>Text Classification</b>	Assigning categories or labels to a whole document, or parts of a document.
<b>Rule-based Matching</b>	Finding sequences of tokens based on their texts and linguistic annotations, similar to regular expressions.
<b>Training</b>	Updating and improving a statistical model's predictions.
<b>Serialization</b>	Saving objects to files or byte strings.



# Annexes : pense-bêtes

---

# Python For Data Science Cheat Sheet

## Python Basics

Learn More Python for Data Science Interactively at [www.datacamp.com](http://www.datacamp.com)



### Variables and Data Types

#### Variable Assignment

```
>>> x=5
>>> x
5
```

#### Calculations With Variables

<code>&gt;&gt;&gt; x+2</code> 7	Sum of two variables
<code>&gt;&gt;&gt; x-2</code> 3	Subtraction of two variables
<code>&gt;&gt;&gt; x*2</code> 10	Multiplication of two variables
<code>&gt;&gt;&gt; x**2</code> 25	Exponentiation of a variable
<code>&gt;&gt;&gt; x%2</code> 1	Remainder of a variable
<code>&gt;&gt;&gt; x/float(2)</code> 2.5	Division of a variable

#### Types and Type Conversion

<code>str()</code>	'5', '3.45', 'True'	Variables to strings
<code>int()</code>	5, 3, 1	Variables to integers
<code>float()</code>	5.0, 1.0	Variables to floats
<code>bool()</code>	True, True, True	Variables to booleans

### Asking For Help

```
>>> help(str)
```

### Strings

```
>>> my_string = 'thisStringIsAwesome'
>>> my_string
'thisStringIsAwesome'
```

#### String Operations

```
>>> my_string * 2
'thisStringIsAwesomethisStringIsAwesome'
>>> my_string + 'Innit'
'thisStringIsAwesomeInnit'
>>> 'm' in my_string
True
```

### Lists

#### Also see NumPy Arrays

```
>>> a = 'is'
>>> b = 'nice'
>>> my_list = ['my', 'list', a, b]
>>> my_list2 = [[4,5,6,7], [3,4,5,6]]
```

### Selecting List Elements

Index starts at 0

#### Subset

```
>>> my_list[1]
>>> my_list[-3]
```

#### Slice

```
>>> my_list[1:3]
```

```
>>> my_list[1:]
>>> my_list[:3]
```

```
>>> my_list[::]
```

#### Subset Lists of Lists

```
>>> my_list2[1][0]
>>> my_list2[1][:2]
```

Select item at index 1  
Select 3rd last item

Select items at index 1 and 2

Select items after index 0

Select items before index 3

Copy my\_list

```
my_list[list][itemOfList]
```

### List Operations

```
>>> my_list + my_list
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list * 2
['my', 'list', 'is', 'nice', 'my', 'list', 'is', 'nice']
>>> my_list2 > 4
True
```

### List Methods

```
>>> my_list.index('a')
>>> my_list.count('a')
>>> my_list.append('!!')
>>> my_list.remove('!!')
>>> del(my_list[0:1])
>>> my_list.reverse()
>>> my_list.extend('!!')
>>> my_list.pop(-1)
>>> my_list.insert(0, '!!')
>>> my_list.sort()
```

Get the index of an item

Count an item

Append an item at a time

Remove an item

Remove an item

Reverse the list

Append an item

Remove an item

Insert an item

Sort the list

### String Operations

Index starts at 0

```
>>> my_string[3]
>>> my_string[4:9]
```

### String Methods

```
>>> my_string.upper()
>>> my_string.lower()
>>> my_string.count('w')
>>> my_string.replace('e', 'i')
>>> my_string.strip()
```

String to uppercase

String to lowercase

Count String elements

Replace String elements

Strip whitespaces

### Libraries

#### Import libraries

```
>>> import numpy
```

```
>>> import numpy as np
```

#### Selective import

```
>>> from math import pi
```



Data analysis



Machine learning



Scientific computing



2D plotting

### Install Python



ANACONDA®

Leading open data science platform  
powered by Python



Free IDE that is included  
with Anaconda



Create and share  
documents with live code,  
visualizations, text, ...

### Numpy Arrays

Also see Lists

```
>>> my_list = [1, 2, 3, 4]
>>> my_array = np.array(my_list)
>>> my_2darray = np.array([[1,2,3],[4,5,6]])
```

### Selecting Numpy Array Elements

Index starts at 0

#### Subset

```
>>> my_array[1]
2
```

Select item at index 1

#### Slice

```
>>> my_array[0:2]
array([1, 2])
```

Select items at index 0 and 1

#### Subset 2D Numpy arrays

```
>>> my_2darray[:,0]
array([1, 4])
```

my\_2darray[rows, columns]

### Numpy Array Operations

```
>>> my_array > 3
array([False, False, False, True], dtype=bool)
>>> my_array * 2
array([2, 4, 6, 8])
>>> my_array + np.array([5, 6, 7, 8])
array([6, 8, 10, 12])
```

### Numpy Array Functions

```
>>> my_array.shape
>>> np.append(other_array)
>>> np.insert(my_array, 1, 5)
>>> np.delete(my_array, [1])
>>> np.mean(my_array)
>>> np.median(my_array)
>>> my_array.correlcoef()
>>> np.std(my_array)
```

Get the dimensions of the array

Append items to an array

Insert items in an array

Delete items in an array

Mean of the array

Median of the array

Correlation coefficient

Standard deviation

DataCamp

Learn Python for Data Science Interactively



## Python For Data Science Cheat Sheet

### Jupyter Notebook

Learn More Python for Data Science Interactively at [www.DataCamp.com](http://www.DataCamp.com)



#### Saving/Loading Notebooks

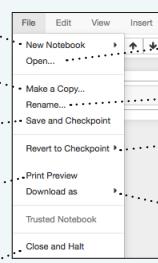
Create new notebook

Make a copy of the current notebook

Save current notebook and record checkpoint

Preview of the printed notebook

Close notebook & stop running any scripts



Open an existing notebook

Rename notebook

Revert notebook to a previous checkpoint

Download notebook as  
- IPython notebook  
- Python  
- HTML  
- Markdown  
- reST  
- LaTeX  
- PDF

#### Writing Code And Text

Code and text are encapsulated by 3 basic cell types: markdown cells, code cells, and raw NBConvert cells.

##### Edit Cells

Cut currently selected cells to clipboard

Paste cells from clipboard above current cell

Paste cells from clipboard on top of current cell

Revert "Delete Cells" invocation

Merge current cell with the one above

Move current cell up

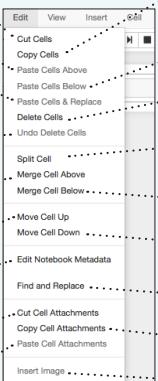
Adjust metadata underlying the current notebook

Remove cell attachments

Paste attachments of current cell

##### Insert Cells

Add new cell above the current one



Copy cells from clipboard to current cursor position

Paste cells from clipboard below current cell

Delete current cells  
Split up a cell from current cursor position

Merge current cell with the one below

Move current cell down

Find and replace in selected cells

Copy attachments of current cell

Insert image in selected cells

Add new cell below the current one

#### Working with Different Programming Languages

Kernels provide computation and communication with front-end interfaces like the notebooks. There are three main kernels:



IPython



IRkernel



IJulia

Installing Jupyter Notebook will automatically install the IPython kernel.

Restart kernel

Restart kernel & run all cells

Restart kernel & run all cells

Change kernel

Interrupt kernel

Interrupt kernel & clear all output

Connect back to a remote notebook

Run other installed kernels

Kernel Widgets Help

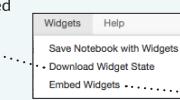
Interrupt  
Restart  
Restart & Run All  
Restart & Run All  
Reconnect  
Shutdown  
Change kernel

#### Widgets

Notebook widgets provide the ability to visualize and control changes in your data, often as a control like a slider, textbox, etc.

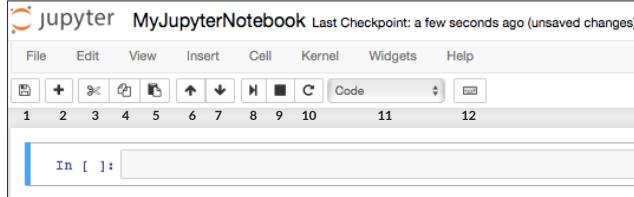
You can use them to build interactive GUIs for your notebooks or to synchronize stateful and stateless information between Python and JavaScript.

Download serialized state of all widget models in use

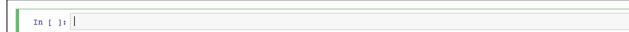


Save notebook with interactive widgets  
Embed current widgets

#### Command Mode:



#### Edit Mode:



#### Executing Cells

Run selected cell(s)

Run current cells down and create a new one above

Run all cells above the current cell

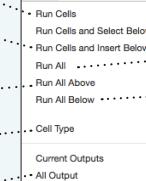
Change the cell type of current cell

toggle, toggle scrolling and clear all output

Run current cells down and create a new one below

Run all cells

Run all cells below the current cell  
toggle, toggle scrolling and clear current outputs



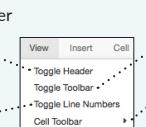
#### View Cells

Toggle display of Jupyter logo and filename

Toggle line numbers in cells

Toggle display of toolbar

Toggle display of cell action icons:  
- None  
- Edit metadata  
- Raw cell format  
- Slideshow  
- Attachments  
- Tags



1. Save and checkpoint
2. Insert cell below
3. Cut cell
4. Copy cell(s)
5. Paste cell(s) below
6. Move cell up
7. Move cell down
8. Run current cell
9. Interrupt kernel
10. Restart kernel
11. Display characteristics
12. Open command palette
13. Current kernel
14. Kernel status
15. Log out from notebook server

#### Asking For Help

Walk through a UI tour

Edit the built-in keyboard shortcuts

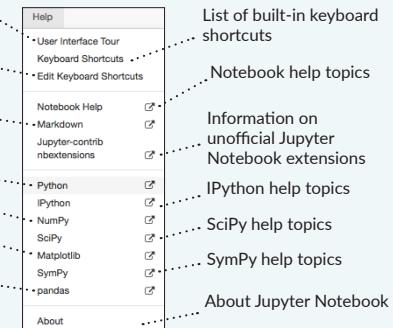
Description of markdown available in notebook

Python help topics

NumPy help topics

Matplotlib help topics

Pandas help topics



List of built-in keyboard shortcuts

Notebook help topics

Information on unofficial Jupyter Notebook extensions

IPython help topics

SciPy help topics

SymPy help topics

About Jupyter Notebook



# Python For Data Science Cheat Sheet

## NumPy Basics

Learn Python for Data Science Interactively at [www.DataCamp.com](http://www.DataCamp.com)



### NumPy

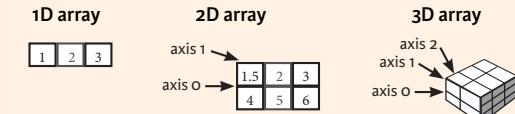
The NumPy library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```



### NumPy Arrays



### Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]),
      dtype = float)
```

### Initial Placeholders

<pre>&gt;&gt;&gt; np.zeros((3,4)) &gt;&gt;&gt; np.ones((2,3),dtype=np.int16) &gt;&gt;&gt; d = np.arange(10,25,5)  &gt;&gt;&gt; np.linspace(0,2,9)  &gt;&gt;&gt; e = np.full((2,2),7) &gt;&gt;&gt; f = np.eye(2) &gt;&gt;&gt; np.random.random((2,2)) &gt;&gt;&gt; np.empty((3,2))</pre>	Create an array of zeros Create an array of ones Create an array of evenly spaced values (step value) Create an array of evenly spaced values (number of samples) Create a constant array Create a 2x2 identity matrix Create an array with random values Create an empty array
---	--

### I/O

#### Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

#### Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

### Data Types

<pre>&gt;&gt;&gt; np.int64 &gt;&gt;&gt; np.float32 &gt;&gt;&gt; np.complex &gt;&gt;&gt; np.bool &gt;&gt;&gt; np.object &gt;&gt;&gt; np.string_ &gt;&gt;&gt; np.unicode_</pre>	Signed 64-bit integer types Standard double-precision floating point Complex numbers represented by 128 floats Boolean type storing TRUE and FALSE values Python object type Fixed-length string type Fixed-length unicode type
---	---

### Inspecting Your Array

<pre>&gt;&gt;&gt; a.shape &gt;&gt;&gt; len(a) &gt;&gt;&gt; b.ndim &gt;&gt;&gt; a.size &gt;&gt;&gt; b.dtype &gt;&gt;&gt; b.dtype.name &gt;&gt;&gt; b.astype(int)</pre>	Array dimensions Length of array Number of array dimensions Number of array elements Data type of array elements Name of data type Convert an array to a different type
---	---

### Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

### Array Mathematics

#### Arithmetic Operations

<pre>&gt;&gt;&gt; g = a - b array([-0.5,  0. ,  0. ], [-3. , -3. , -3. ]) &gt;&gt;&gt; np.subtract(a,b) &gt;&gt;&gt; b + a array([[ 2.5,  4. ,  6. ], [ 5. ,  7. ,  9. ]]) &gt;&gt;&gt; np.add(b,a) &gt;&gt;&gt; a / b array([[ 0.66666667,  1. , [ 0.25,  0.4,  0.5 ]], [ 1.5,  4.,  9. ], [ 4., 10., 18. ]]) &gt;&gt;&gt; np.multiply(a,b) &gt;&gt;&gt; np.exp(b) &gt;&gt;&gt; np.sqrt(b) &gt;&gt;&gt; np.sin(a) &gt;&gt;&gt; np.cos(b) &gt;&gt;&gt; np.log(a) &gt;&gt;&gt; e.dot(f) array([[ 7.,  7.], [ 7.,  7.]])</pre>	Subtraction Subtraction Addition Addition Division Division Multiplication Exponentiation Square root Print sines of an array Element-wise cosine Element-wise natural logarithm Dot product
--	--

#### Comparison

<pre>&gt;&gt;&gt; a == b array([[False,  True,  True],        [False, False, False]], dtype=bool) &gt;&gt;&gt; a &lt; 2 array([True, False, False], dtype=bool) &gt;&gt;&gt; np.array_equal(a, b)</pre>	Element-wise comparison Element-wise comparison Array-wise comparison
---	---

#### Aggregate Functions

<pre>&gt;&gt;&gt; a.sum() &gt;&gt;&gt; a.min() &gt;&gt;&gt; b.max(axis=0) &gt;&gt;&gt; b.cumsum(axis=1) &gt;&gt;&gt; a.mean() &gt;&gt;&gt; b.median() &gt;&gt;&gt; a.corrcoef() &gt;&gt;&gt; np.std(b)</pre>	Array-wise sum Array-wise minimum value Maximum value of an array row Cumulative sum of the elements Mean Median Correlation coefficient Standard deviation
--	--

### Copying Arrays

<pre>&gt;&gt;&gt; h = a.view() &gt;&gt;&gt; np.copy(a) &gt;&gt;&gt; h = a.copy()</pre>	Create a view of the array with the same data Create a copy of the array Create a deep copy of the array
--	--

### Sorting Arrays

<pre>&gt;&gt;&gt; a.sort() &gt;&gt;&gt; c.sort(axis=0)</pre>	Sort an array Sort the elements of an array's axis
--	---

### Subsetting, Slicing, Indexing

Also see [Lists](#)

#### Subsetting

```
>>> a[2]
3
>>> b[1,2]
6.0
```

	1 2 3
	1.5 2 3
	4 5 6

Select the element at the 2nd index  
 Select the element at row 0 column 2 (equivalent to `b[1][2]`)

#### Slicing

```
>>> a[0:2]
array([1, 2])
>>> b[0:2,1]
array([ 2.,  5.])
```

	1 2 3
	1.5 2 3
	4 5 6

Select items at index 0 and 1  
 Select items at rows 0 and 1 in column 1

#### Boolean Indexing

```
>>> a[<2]
array([1])
>>> a[<2]
array([1])
```

	1 2 3
	1.5 2 3
	4 5 6

Reversed array `a`

#### Fancy Indexing

```
>>> b[[1, 0, 1, 0], [0, 1, 2, 0]]
array([ 4.,  2.,  6.,  1.5])
>>> b[[1, 0, 1, 0], [0, 1, 2, 0]]
array([ 1.5,  2.,  6.,  4.],
[ 4.,  5.,  6.,  4.],
[ 1.5,  2.,  3.,  1.5]])
```

	1 2 3
	1.5 2 3
	4 5 6

Select elements `(1,0),(0,1),(1,2)` and `(0,0)`  
 Select a subset of the matrix's rows and columns

### Array Manipulation

#### Transposing Array

```
>>> i = np.transpose(b)
>>> i.T
```

Permute array dimensions  
 Permute array dimensions

#### Changing Array Shape

```
>>> b.ravel()
>>> g.reshape(3,-2)
```

Flatten the array  
 Reshape, but don't change data

#### Adding/Removing Elements

```
>>> h.resize((2,6))
>>> np.append(h,g)
>>> np.insert(a, 1, 5)
>>> np.delete(a, [1])
```

Return a new array with shape (2,6)  
 Append items to an array  
 Insert items in an array  
 Delete items from an array

#### Combining Arrays

```
>>> np.concatenate((a,d),axis=0)
array([ 1,  2,  3, 10, 15, 20])
>>> np.vstack((a,b))
array([[ 1.,  2.,  3.],
[ 1.5,  2.,  3.],
[ 4.,  5.,  6.]])
```

Concatenate arrays  
 Stack arrays vertically (row-wise)

#### Stacking Arrays

```
>>> np.r_[e,f]
>>> np.hstack((e,f))
array([ 7.,  7.,  1.,  0.],
[ 7.,  7.,  0.,  1.])
```

Stack arrays vertically (row-wise)  
 Stack arrays horizontally (column-wise)

#### Create stacked column-wise arrays

```
>>> np.column_stack((a,d))
array([[ 1, 10],
[ 2, 15],
[ 3, 20]])
>>> np.c_[a,d]
```

Create stacked column-wise arrays

#### Splitting Arrays

```
>>> np.hsplit(a,3)
[array([1]),array([2]),array([3])]
>>> np.vsplit(c,2)
[array([[ 1.5,  2.,  3.],
[ 4.,  5.,  6.]]),
array([[ 3.,  2.,  3.],
[ 4.,  5.,  6.]])]
```

Split the array horizontally at the 3rd index  
 Split the array vertically at the 2nd index



# Python For Data Science Cheat Sheet

## SciPy - Linear Algebra

Learn More Python for Data Science Interactively at [www.datacamp.com](http://www.datacamp.com)



### SciPy

The SciPy library is one of the core packages for scientific computing that provides mathematical algorithms and convenience functions built on the NumPy extension of Python.



### Interacting With NumPy

[Also see NumPy](#)

```
>>> import numpy as np
>>> a = np.array([1, 2, 3])
>>> b = np.array([(1+5j, 2j, 3j), (4j, 5j, 6j)])
>>> c = np.array([(1.5, 2, 3), (4, 5, 6), [(3, 2, 1), (4, 5, 6)]])
```

### Index Tricks

>>> np.mgrid[0:5, 0:5]	Create a dense meshgrid
>>> np.ogrid[0:2, 0:2]	Create an open meshgrid
>>> np.r_[3, [0]*5, -1:1:10j]	Stack arrays vertically (row-wise)
>>> np.c_[b, c]	Create stacked column-wise arrays

### Shape Manipulation

>>> np.transpose(b)	Permute array dimensions
>>> b.flatten()	Flatten the array
>>> np.hstack((b, c))	Stack arrays horizontally (column-wise)
>>> np.vstack((a, b))	Stack arrays vertically (row-wise)
>>> np.hsplit(c, 2)	Split the array horizontally at the 2nd index
>>> np.vsplit(d, 2)	Split the array vertically at the 2nd index

### Polynomials

```
>>> from numpy import poly1d
>>> p = poly1d([3, 4, 5])
```

Create a polynomial object

### Vectorizing Functions

```
>>> def myfunc(a):
...     if a < 0:
...         return a**2
...     else:
...         return a/2
>>> np.vectorize(myfunc)
```

Vectorize functions

### Type Handling

>>> np.real(c)	Return the real part of the array elements
>>> np.imag(c)	Return the imaginary part of the array elements
>>> np.real_if_close(c, tol=1000)	Return a real array if complex parts close to 0
>>> np.cast['E'](np.pi)	Cast object to a data type

### Other Useful Functions

>>> np.angle(b, deg=True)	Return the angle of the complex argument
>>> g = np.linspace(0, np.pi, num=5)	Create an array of evenly spaced values (number of samples)
>>> g[3:] += np.pi	Unwrap
>>> np.unwrap(g)	Create an array of evenly spaced values (log scale)
>>> np.logspace(0, 10, 3)	Return values from a list of arrays depending on conditions
>>> np.select([c<4], [c**2])	Factorial
>>> misc.factorial(a)	Combine N things taken at k time
>>> misc.comb(10, 3, exact=True)	Weights for N-point central derivative
>>> misc.central_diff_weights(3)	Find the n-th derivative of a function at a point
>>> misc.derivative(myfunc, 1.0)	

## Linear Algebra

You'll use the `linalg` and `sparse` modules. Note that `scipy.linalg` contains and expands on `numpy.linalg`.

[Also see NumPy](#)

198

### Matrix Functions

#### Addition

```
>>> np.add(A, D)
```

Addition

#### Subtraction

```
>>> np.subtract(A, D)
```

Subtraction

#### Division

```
>>> np.divide(A, D)
```

Division

#### Multiplication

```
>>> np.multiply(D, A)
```

Multiplication

```
>>> np.dot(A, D)
```

Dot product

```
>>> np.vdot(A, D)
```

Vector dot product

```
>>> np.inner(A, D)
```

Inner product

```
>>> np.outer(A, D)
```

Outer product

```
>>> np.tensordot(A, D, 0)
```

Tensor dot product

```
>>> np.kron(A, D)
```

Kronecker product

#### Exponential Functions

```
>>> linalg.expm(A)
```

Matrix exponential

```
>>> linalg.expm2(A)
```

Matrix exponential (Taylor Series)

```
>>> linalg.expm3(D)
```

Matrix exponential (eigenvalue decomposition)

#### Logarithm Function

```
>>> linalg.logm(A)
```

Matrix logarithm

#### Trigonometric Functions

```
>>> linalg.sinm(D)
```

Matrix sine

```
>>> linalg.cosm(D)
```

Matrix cosine

```
>>> linalg.tanm(A)
```

Matrix tangent

#### Hyperbolic Trigonometric Functions

```
>>> linalg.sinh(D)
```

Hyperbolic matrix sine

```
>>> linalg.cosh(D)
```

Hyperbolic matrix cosine

```
>>> linalg.tanh(A)
```

Hyperbolic matrix tangent

#### Matrix Sign Function

```
>>> np.sign(A)
```

Matrix sign function

#### Matrix Square Root

```
>>> linalg.sqrtm(A)
```

Matrix square root

#### Arbitrary Functions

```
>>> linalg.funm(A, lambda x: x*x)
```

Evaluate matrix function

### Decompositions

#### Eigenvalues and Eigenvectors

```
>>> la, v = linalg.eig(A)
```

Solve ordinary or generalized eigenvalue problem for square matrix

```
>>> l1, l2 = la
```

Unpack eigenvalues

```
>>> v[:, 0]
```

First eigenvector

```
>>> v[:, 1]
```

Second eigenvector

```
>>> la
```

Unpack eigenvalues

#### Singular Value Decomposition

```
>>> U, s, Vh = linalg.svd(B)
```

Singular Value Decomposition (SVD)

```
>>> M, N = B.shape
```

Construct sigma matrix in SVD

```
>>> Sig = linalg.diagsvd(s, M, N)
```

LU Decomposition

```
>>> P, L, U = linalg.lu(C)
```

LU Decomposition

#### Sparse Matrix Decompositions

```
>>> la, v = sparse.linalg.eigs(F, 1)
```

Eigenvalues and eigenvectors

```
>>> sparse.linalg.svds(H, 2)
```

SVD

### Asking For Help

```
>>> help(scipy.linalg.diagsvd)
>>> np.info(np.matrix)
```



## Python For Data Science Cheat Sheet

### Pandas Basics

Learn Python for Data Science Interactively at [www.DataCamp.com](http://www.DataCamp.com)



#### Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.



Use the following import convention:

```
>>> import pandas as pd
```

#### Pandas Data Structures

##### Series

A one-dimensional labeled array capable of holding any data type

Index	a	3
	b	-5
	c	7
	d	4

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

##### DataFrame

Columns

	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasilia	207847528

A two-dimensional labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
   >>>          'Capital': ['Brussels', 'New Delhi', 'Brasília'],
   >>>          'Population': [11190846, 1303171035, 207847528]}
>>> df = pd.DataFrame(data,
   >>>                  columns=['Country', 'Capital', 'Population'])
```

#### I/O

##### Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

##### Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
Read multiple sheets from the same file
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

#### Asking For Help

```
>>> help(pd.Series.loc)
```

#### Selection

Also see NumPy Arrays

##### Getting

```
>>> s['b']
```

-5

```
>>> df[1]
   Country    Capital  Population
1  India      New Delhi  1303171035
2  Brazil     Brasilia  207847528
```

Get one element

Get subset of a DataFrame

#### Selecting, Boolean Indexing & Setting

##### By Position

```
>>> df.iloc[0], [0])
'Belgium'
>>> df.iat([0], [0])
'Belgium'
```

Select single value by row & column

##### By Label

```
>>> df.loc([0], ['Country'])
'Belgium'
>>> df.at[0, ['Country']]
'Belgium'
```

Select single value by row & column labels

##### By Label/Position

```
>>> df.ix[2]
   Country    Brazil
   Capital    Brasilia
   Population 207847528
>>> df.ix[:, 'Capital']
0  Brussels
1  New Delhi
2  Brasilia
>>> df.ix[1, 'Capital']
'New Delhi'
```

Select single row of subset of rows

Select a single column of subset of columns

Select rows and columns

##### Boolean Indexing

```
>>> s[(s > 1)]
>>> s[(s < -1) | (s > 2)]
>>> df[df['Population']>1200000000]
```

Series s where value is not >1

s where value is <-1 or >2

Use filter to adjust DataFrame

##### Setting

```
>>> s['a'] = 6
```

Set index a of Series s to 6

#### Dropping

```
>>> s.drop(['a', 'c'])
```

Drop values from rows (axis=0)

```
>>> df.drop('Country', axis=1)
```

Drop values from columns(axis=1)

#### Sort & Rank

```
>>> df.sort_index()
>>> df.sort_values(by='Country')
>>> df.rank()
```

Sort by labels along an axis  
Sort by the values along an axis  
Assign ranks to entries

#### Retrieving Series/DataFrame Information

##### Basic Information

>>> df.shape	(rows,columns)
>>> df.index	Describe index
>>> df.columns	Describe DataFrame columns
>>> df.info()	Info on DataFrame
>>> df.count()	Number of non-NA values

##### Summary

>>> df.sum()	Sum of values
>>> df.cumsum()	Cummulative sum of values
>>> df.min()/df.max()	Minimum/maximum values
>>> df.idxmin()/df.idxmax()	Minimum/Maximum index value
>>> df.describe()	Summary statistics
>>> df.mean()	Mean of values
>>> df.median()	Median of values

#### Applying Functions

```
>>> f = lambda x: x**2
>>> df.apply(f)
>>> df.applymap(f)
```

Apply function  
Apply function element-wise

#### Data Alignment

##### Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
a    10.0
b    NaN
c    5.0
d    7.0
```

#### Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
a    10.0
b    -5.0
c    5.0
d    7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```



# Python For Data Science Cheat Sheet

## Scikit-Learn

Learn Python for data science interactively at [www.DataCamp.com](http://www.DataCamp.com)



### Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

#### A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

### Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10, 5))
>>> y = np.array(['M', 'M', 'F', 'F', 'M', 'F', 'M', 'F', 'F'])
>>> X[X < 0.7] = 0
```

### Training And Test Data

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
...                                                    y,
...                                                    random_state=0)
```

### Preprocessing The Data

#### Standardization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```

#### Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

#### Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

## Create Your Model

### Supervised Learning Estimators

#### Linear Regression

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

#### Support Vector Machines (SVM)

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

#### Naive Bayes

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

#### KNN

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

### Unsupervised Learning Estimators

#### Principal Component Analysis (PCA)

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

#### K Means

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

## Model Fitting

### Supervised learning

```
>>> lr.fit(X, y)
>>> knn.fit(X_train, y_train)
>>> svc.fit(X_train, y_train)
```

### Unsupervised Learning

```
>>> k_means.fit(X_train)
>>> pca_model = pca.fit_transform(X_train)
```

Fit the model to the data

Fit the model to the data

Fit to data, then transform it

## Prediction

### Supervised Estimators

```
>>> y_pred = svc.predict(np.random.random((2,5)))
>>> y_pred = lr.predict(X_test)
>>> y_pred = knn.predict_proba(X_test)
```

### Unsupervised Estimators

```
>>> y_pred = k_means.predict(X_test)
```

Predict labels

Predict labels

Estimate probability of a label

Predict labels in clustering algos

### Encoding Categorical Features

```
>>> from sklearn.preprocessing import LabelEncoder
>>> enc = LabelEncoder()
>>> y = enc.fit_transform(y)
```

### Imputing Missing Values

```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```

### Generating Polynomial Features

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

## Evaluate Your Model's Performance

### Classification Metrics

#### Accuracy Score

```
>>> knn.score(X_test, y_test)
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Estimator score method  
Metric scoring functions

#### Classification Report

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, y_pred))
```

Precision, recall, f1-score and support

#### Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

### Regression Metrics

#### Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

#### Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

#### R<sup>2</sup> Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

### Clustering Metrics

#### Adjusted Rand Index

```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

#### Homogeneity

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

#### V-measure

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

### Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

### Tune Your Model

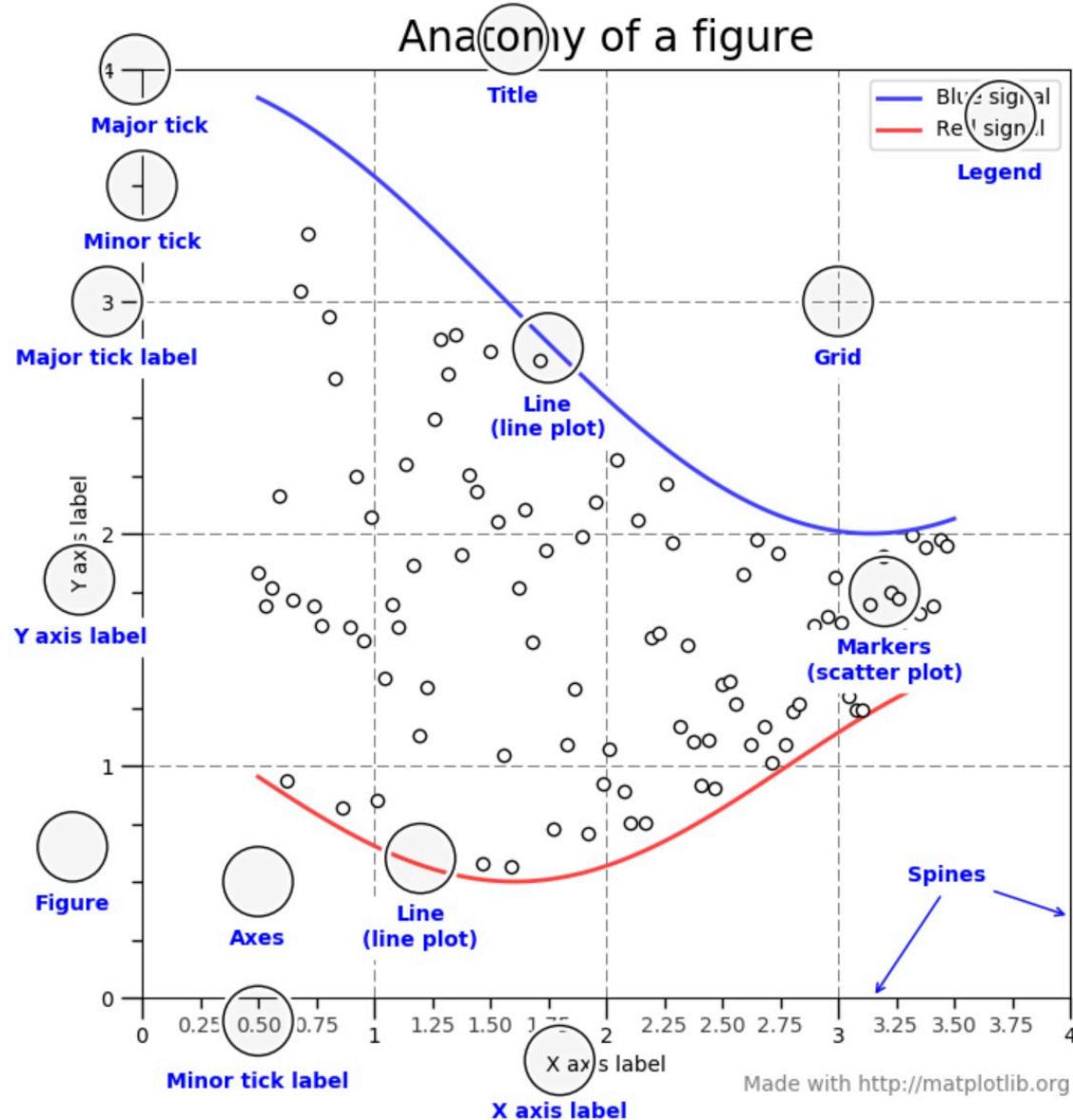
#### Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {'n_neighbors': np.arange(1,3),
...            "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
...                      param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

#### Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {'n_neighbors': range(1,5),
...            "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
...                               param_distributions=params,
...                               cv=4,
...                               n_iter=8,
...                               random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```





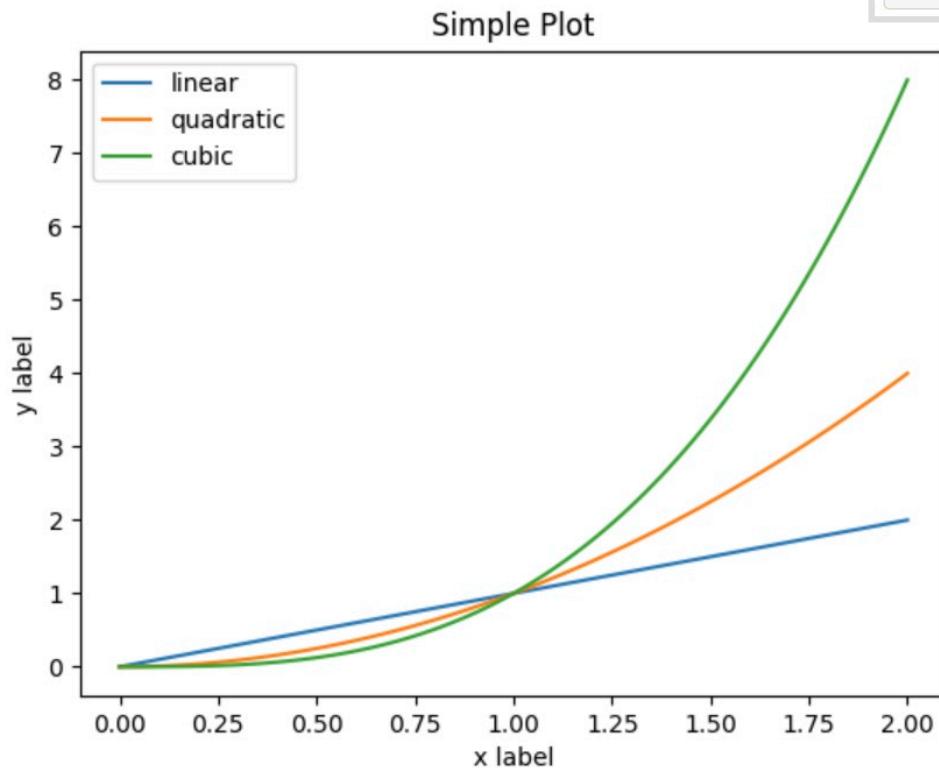
```
x = np.linspace(0, 2, 100)

# Note that even in the OO-style, we use `pyplot.figure` to create the figure.
fig, ax = plt.subplots() # Create a figure and an axes.
ax.plot(x, x, label='linear') # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic') # Plot more data on the axes...
ax.plot(x, x**3, label='cubic') # ... and some more.
ax.set_xlabel('x label') # Add an x-label to the axes.
ax.set_ylabel('y label') # Add a y-label to the axes.
ax.set_title("Simple Plot") # Add a title to the axes.
ax.legend() # Add a legend.
```

or (pyplot-style)

```
x = np.linspace(0, 2, 100)

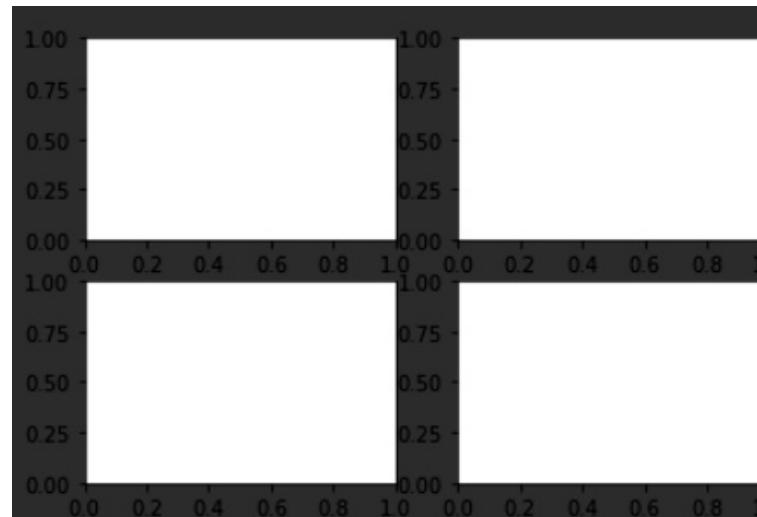
plt.plot(x, x, label='linear') # Plot some data on the (implicit) axes.
plt.plot(x, x**2, label='quadratic') # etc.
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
```



```
fig = plt.figure() # an empty figure with no Axes  
  
fig, ax = plt.subplots() # a figure with a single Axes
```



```
fig, axs = plt.subplots(2, 2) # a figure with a 2x2 grid of Axes
```



## Python For Data Science Cheat Sheet

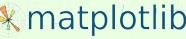
### Matplotlib

Learn Python Interactively at [www.DataCamp.com](http://www.DataCamp.com)



#### Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.



#### 1 Prepare The Data

Also see [Lists & NumPy](#)

##### 1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

##### 2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]
>>> U = -1 - X**2 - Y
>>> V = 1 + X - Y**2
>>> from matplotlib.cbook import get_sample_data
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

#### 2 Create Plot

```
>>> import matplotlib.pyplot as plt
```

##### Figure

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

##### Axes

All plotting is done with respect to an `Axes`. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) # row-col-num
>>> ax3 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows=2, ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

#### 3 Plotting Routines

##### 1D Data

```
>>> fig, ax = plt.subplots()
>>> lines = ax.plot(x,y)
>>> ax.scatter(x,y)
>>> axes[0,0].bar([1,2,3],[3,4,5])
>>> axes[1,0].barh([0.5,1,2.5],[0,1,2])
>>> axes[1,1].axhline(0.45)
>>> axes[0,1].axvline(0.65)
>>> ax.fill(x,y,color='blue')
>>> ax.fill_between(x,y,color='yellow')
```

Draw points with lines or markers connecting them  
Draw unconnected points, scaled or colored  
Plot vertical rectangles (constant width)  
Plot horizontal rectangles (constant height)  
Draw a horizontal line across axes  
Draw a vertical line across axes  
Draw filled polygons  
Fill between y-values and 0

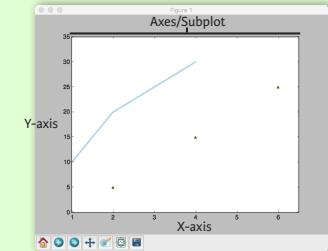
##### 2D Data or Images

```
>>> fig, ax = plt.subplots()
>>> im = ax.imshow(img,
                  cmap='gist_earth',
                  interpolation='nearest',
                  vmin=-2,
                  vmax=2)
```

Colormapped or RGB arrays

## Plot Anatomy & Workflow

### Plot Anatomy



Figure

### Workflow

The basic steps to creating plots with matplotlib are:

- 1 Prepare data
- 2 Create plot
- 3 Plot
- 4 Customize plot
- 5 Save plot
- 6 Show plot

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4] Step 1
>>> y = [10,20,25,30]
>>> fig = plt.figure() Step 2
>>> ax = fig.add_subplot(111) Step 3
>>> ax.plot(x, y, color='lightblue', linewidth=3) Step 3,4
>>> ax.scatter([2,4,6],
              [5,15,25],
              color='darkgreen',
              marker='^')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png') Step 5
>>> plt.show() Step 6
```

#### 4 Customize Plot

##### Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, c='k')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img,
                  cmap='seismic')
```

##### Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker=".") 
>>> ax.plot(x,y,marker="o")
```

##### Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,ls='solid')
>>> plt.plot(x,y,ls='--')
>>> plt.plot(x,y,'-.',x**2,y**2,'-.')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

##### Text & Annotations

```
>>> ax.text(1,-2,1,
           'Example Graph',
           style='italic')
>>> ax.annotate("Sine",
               xy=(8,0),
               xycoords='data',
               xytext=(10.5, 0),
               textcoords='data',
               arrowprops=dict(arrowstyle="->",
                               connectionstyle="arc3"),)
```

##### Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)
>>> axes[1,1].quiver(y,z)
>>> axes[0,1].streamplot(X,Y,U,V)
```

Add an arrow to the axes  
Plot a 2D field of arrows  
Plot a 2D field of arrows

##### Data Distributions

```
>>> ax1.hist(y)
>>> ax3.boxplot(y)
>>> ax3.violinplot(z)
```

Plot a histogram  
Make a box and whisker plot  
Make a violin plot

#### 5 Save Plot

**Save figures**

```
>>> plt.savefig('foo.png')
Save transparent figures
>>> plt.savefig('foo.png', transparent=True)
```

#### 6 Show Plot

```
>>> plt.show()
```

#### Close & Clear

```
>>> plt.clf()
>>> plt.cla()
>>> plt.close()
```

Clear an axis  
Clear the entire figure  
Close a window





0.11.1

Gallery

Tutorial

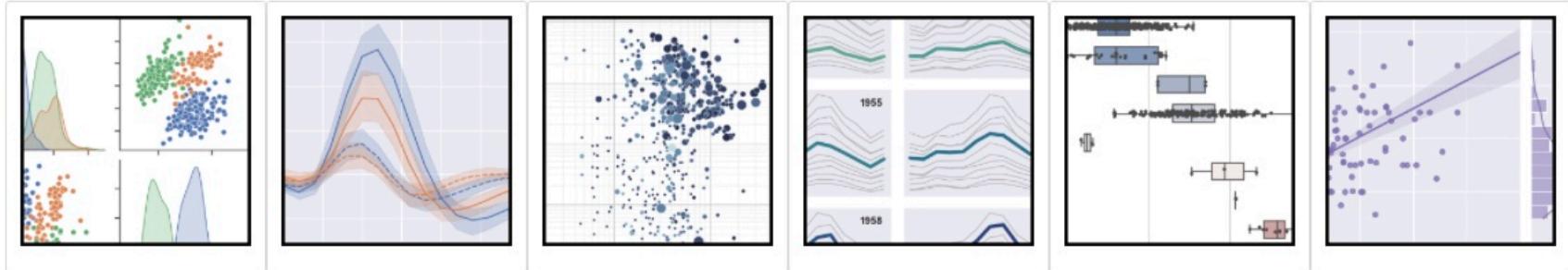
API

Site ▾

Page ▾

Search

# seaborn: statistical data visualization



Seaborn is a Python data visualization library based on [matplotlib](#). It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the [introductory notes](#). Visit the [installation page](#) to see how you can download the package and get started with it. You can browse the [example gallery](#) to see what you can do with seaborn, and then check out the [tutorial](#) and [API reference](#) to find out how.

To see the code or report a bug, please visit the [GitHub repository](#). General support questions are most at home on [stackoverflow](#) or [discourse](#), which have dedicated channels for seaborn.

## Contents

- [Introduction](#)
- [Release notes](#)
- [Installing](#)
- [Example gallery](#)
- [Tutorial](#)
- [API reference](#)

## Features

- Relational: [API](#) | [Tutorial](#)
- Distribution: [API](#) | [Tutorial](#)
- Categorical: [API](#) | [Tutorial](#)
- Regression: [API](#) | [Tutorial](#)
- Multiples: [API](#) | [Tutorial](#)
- Style: [API](#) | [Tutorial](#)
- Color: [API](#) | [Tutorial](#)

## Python For Data Science Cheat Sheet

### Seaborn

Learn Data Science Interactively at [www.DataCamp.com](http://www.DataCamp.com)



#### Statistical Data Visualization With Seaborn

The Python visualization library **Seaborn** is based on matplotlib and provides a high-level interface for drawing attractive statistical graphics.

Make use of the following aliases to import the libraries:

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
```

The basic steps to creating plots with Seaborn are:

1. Prepare some data
2. Control figure aesthetics
3. Plot with Seaborn
4. Further customize your plot

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> tips = sns.load_dataset("tips")          Step 1
>>> sns.set_style("whitegrid")               Step 2
>>> g = sns.lmplot(x="tip",
                   y="total_bill",
                   data=tips,
                   aspect=2)                         Step 3
>>> g = (g.set_axis_labels("Tip", "Total bill(USD)")
        .set(xlim=(0,10), ylim=(0,100))           Step 4
        .set(legend=False))                      Step 5
>>> plt.title("title")
>>> plt.show(g)
```

#### 1) Data

[Also see Lists, NumPy & Pandas](#)

```
>>> import pandas as pd
>>> import numpy as np
>>> uniform_data = np.random.rand(10, 12)
>>> data = pd.DataFrame({ "x":np.arange(1,101),
                           "y":np.random.normal(0,4,100) })
```

Seaborn also offers built-in data sets:

```
>>> titanic = sns.load_dataset("titanic")
>>> iris = sns.load_dataset("iris")
```

#### 2) Figure Aesthetics

```
>>> f, ax = plt.subplots(figsize=(5, 6))
```

Create a figure and one subplot

##### Seaborn styles

```
>>> sns.set()
>>> sns.set_style("whitegrid")
>>> sns.set_style("ticks",
                  {"xtick.major.size":8,
                   "ytick.major.size":8})
>>> sns.axes_style("whitegrid")
```

(Re)set the seaborn default  
Set the matplotlib parameters  
Set the matplotlib parameters

Return a dict of params or use with  
with to temporarily set the style

#### 3) Plotting With Seaborn

##### Axis Grids

```
>>> g = sns.FacetGrid(titanic,
                      col="survived",
                      row="sex")
>>> g = g.map(plt.hist, "age")
>>> sns.factorplot(x="pclass",
                   y="survived",
                   hue="sex",
                   data=titanic)
>>> sns.lmplot(x="sepal_width",
                   y="sepal_length",
                   hue="species",
                   data=iris)
```

Subplot grid for plotting conditional relationships

Draw a categorical plot onto a Facetgrid

Plot data and regression model fits across a FacetGrid

```
>>> h = sns.PairGrid(iris)
>>> h = h.map(plt.scatter)
>>> sns.pairplot(iris)
>>> i = sns.JointGrid(x="x",
                      y="y",
                      data=data)
>>> i = i.plot(sns.regplot,
                  sns.distplot)
>>> sns.jointplot("sepal_length",
                  "sepal_width",
                  data=iris,
                  kind="kde")
```

Subplot grid for plotting pairwise relationships  
Plot pairwise bivariate distributions  
Grid for bivariate plot with marginal univariate plots

Plot bivariate distribution

##### Categorical Plots

```
Scatterplot
>>> sns.stripplot(x="species",
                   y="petal_length",
                   data=iris)
>>> sns.swarmplot(x="species",
                   y="petal_length",
                   data=iris)
```

Scatterplot with one categorical variable

Categorical scatterplot with non-overlapping points

```
Bar Chart
>>> sns.barplot(x="sex",
                 y="survived",
                 hue="class",
                 data=titanic)
```

Show point estimates and confidence intervals with scatterplot glyphs

```
Count Plot
>>> sns.countplot(x="deck",
                  data=titanic,
                  palette="Greens_d")
```

Show count of observations

```
Point Plot
>>> sns.pointplot(x="class",
                   y="survived",
                   hue="sex",
                   data=titanic,
                   palette={"male":"g",
                            "female":"m"},
                   markers=["^", "o"],
                   linestyles=["-", "--"])
```

Show point estimates and confidence intervals as rectangular bars

```
Boxplot
>>> sns.boxplot(x="alive",
                 y="age",
                 hue="adult_male",
                 data=titanic)
```

Boxplot

```
>>> sns.boxplot(data=iris, orient="h")
```

Boxplot with wide-form data

```
Violinplot
>>> sns.violinplot(x="age",
                   y="sex",
                   hue="survived",
                   data=titanic)
```

Violin plot

```
Regression Plots
>>> sns.regplot(x="sepal_width",
                 y="sepal_length",
                 data=iris,
                 ax=ax)
```

Plot data and a linear regression model fit

```
Distribution Plots
>>> plot = sns.distplot(data.y,
                        kde=False,
                        color="b")
```

Plot univariate distribution

```
Matrix Plots
>>> sns.heatmap(uniform_data, vmin=0, vmax=1)
```

Heatmap

#### 4) Further Customizations

[Also see Matplotlib](#)

##### Axisgrid Objects

```
>>> g.despine(left=True)
>>> g.set_ylabels("Survived")
>>> g.set_xticklabels(rotation=45)
>>> g.set_axis_labels("Survived",
                      "Sex")
>>> h.set(xlim=(0, 5),
          ylim=(0, 5),
          xticks=[0, 2.5, 5],
          yticks=[0, 2.5, 5])
```

Remove left spine  
Set the labels of the y-axis  
Set the tick labels for x  
Set the axis labels

Set the limit and ticks of the x-and y-axis

##### Plot

```
>>> plt.title("A Title")
>>> plt.ylabel("Survived")
>>> plt.xlabel("Sex")
>>> plt.ylim(0,100)
>>> plt.xlim(0,10)
>>> plt.setp(ax, yticks=[0,5])
>>> plt.tight_layout()
```

Add plot title  
Adjust the label of the y-axis  
Adjust the label of the x-axis  
Adjust the limits of the y-axis  
Adjust the limits of the x-axis  
Adjust a plot property  
Adjust subplot params

#### 5) Show or Save Plot

[Also see Matplotlib](#)

```
>>> plt.show()
>>> plt.savefig("foo.png")
>>> plt.savefig("foo.png",
                transparent=True)
```

Show the plot  
Save the plot as a figure  
Save transparent figure

#### Close & Clear

[Also see Matplotlib](#)

```
>>> plt.cla()
>>> plt.clf()
>>> plt.close()
```

Clear an axis  
Clear an entire figure  
Close a window



## Matplotlib for beginners

Matplotlib is a library for making 2D plots in Python. It is designed with the philosophy that you should be able to create simple plots with just a few commands:

### 1 Initialize

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

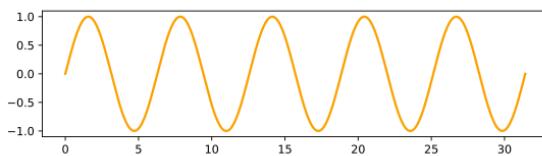
### 2 Prepare

```
X = np.linspace(0, 4*np.pi, 1000)
Y = np.sin(X)
```

### 3 Render

```
fig, ax = plt.subplots()
ax.plot(X, Y)
fig.show()
```

### 4 Observe



## Choose

Matplotlib offers several kind of plots (see Gallery):

```
X = np.random.uniform(0, 1, 100)
Y = np.random.uniform(0, 1, 100)
ax.scatter(X, Y)
```



```
X = np.arange(10)
Y = np.random.uniform(1, 10, 10)
ax.bar(X, Y)
```



```
Z = np.random.uniform(0, 1, (8,8))
ax.imshow(Z)
```



<https://github.com/matplotlib/cheatsheets#cheatsheets>

```
Z = np.random.uniform(0, 1, (8,8))
ax.contourf(Z)
```



```
Z = np.random.uniform(0, 1, 4)
ax.pie(Z)
```



```
Z = np.random.normal(0, 1, 100)
ax.hist(Z)
```



```
X = np.arange(5)
Y = np.random.uniform(0,1,5)
ax.errorbar(X, Y, Y/4)
```



```
Z = np.random.normal(0,1,(100,3))
ax.boxplot(Z)
```



## Organize

You can plot several data on the same figure but you can also split a figure in several subplots (named Axes):

```
X = np.linspace(0,10,100)
Y1, Y2 = np.sin(X), np.cos(X)
ax.plot(X, Y1, color="C1")
ax.plot(X, Y2, color="C0")
```



```
fig, (ax1, ax2) = plt.subplots((2,1))
ax1.plot(Y1, X, color="C1")
ax2.plot(Y2, X, color="C0")
```



```
fig, (ax1, ax2) = plt.subplots((1,2))
ax1.plot(Y1, X, color="C1")
ax2.plot(Y2, X, color="C0")
```



## Label (everything)

```
ax.plot(X, Y)
fig.suptitle(None)
ax.set_title("A Sine wave")
```



```
ax.plot(X, Y)
ax.set_ylabel(None)
ax.set_xlabel("Time")
```



## Tweak

You can modify pretty much anything in a plot, including limits, colors, markers, line width and styles, ticks and ticks labels, titles, etc.

```
X = np.linspace(0,10,100)
Y = np.sin(X)
ax.plot(X, Y, color="black")
```



```
X = np.linspace(0,10,100)
Y = np.sin(X)
ax.plot(X, Y, linestyle="--")
```



```
X = np.linspace(0,10,100)
Y = np.sin(X)
ax.plot(X, Y, linewidth=5)
```



```
X = np.linspace(0,10,100)
Y = np.sin(X)
ax.plot(X, Y, marker="o")
```



## Explore

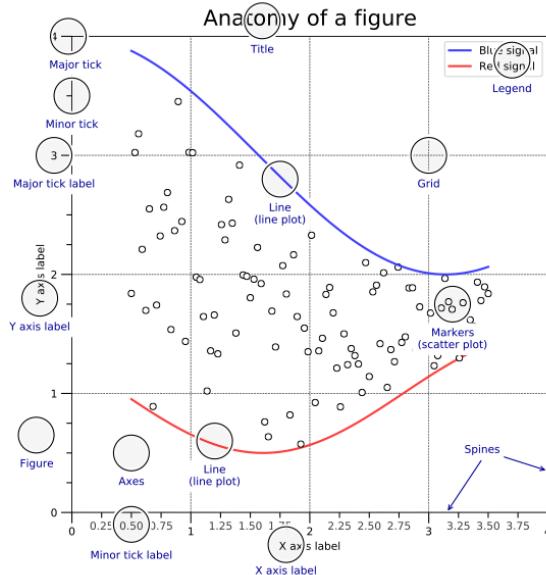
Figures are shown with a graphical user interface that allows to zoom and pan the figure, to navigate between the different views and to show the value under the mouse.

## Save (bitmap or vector format)

```
fig.savefig("my-first-figure.png", dpi=300)
fig.savefig("my-first-figure.pdf")
```

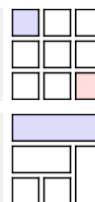
## Matplotlib for intermediate users

A matplotlib figure is composed of a hierarchy of elements that forms the actual figure. Each element can be modified.

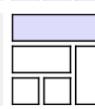


### Figure, axes & spines

```
fig, axs = plt.subplots((3,3))
axs[0,0].set_facecolor("#ddffff")
axs[2,2].set_facecolor("#ffffdd")
```



```
gs = fig.add_gridspec(3, 3)
ax = fig.add_subplot(gs[0, :])
ax.set_facecolor("#ddffff")
```



```
fig, ax = plt.subplots()
ax.spines["top"].set_color("None")
ax.spines["right"].set_color("None")
```



### Ticks & labels

```
from mpl.ticker import MultipleLocator as ML
from mpl.ticker import ScalarFormatter as SF
ax.xaxis.set_minor_locator(ML(0.2))
ax.xaxis.set_minor_formatter(SF())
ax.tick_params(axis='x', which='minor', rotation=90)
```

### Lines & markers

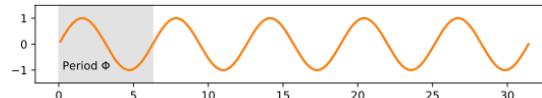
```
X = np.linspace(0.1, 10*np.pi, 1000)
Y = np.sin(X)
ax.plot(X, Y, "C1o-", markevery=25, mec="1.0")
```

### Scales & Projections

```
fig, ax = plt.subplots()
ax.set_xscale("log")
ax.plot(X, Y, "C1o-", markevery=25, mec="1.0")
```

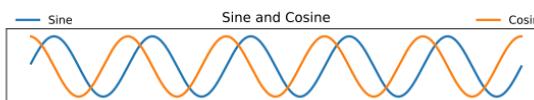
### Text & Ornaments

```
ax.fill_betweenx([-1,1],[0],[2*np.pi])
ax.text(0, -1, r"Period $\Phi$")
```



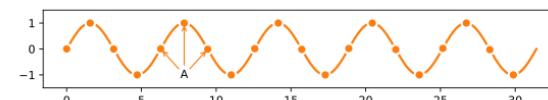
### Legend

```
ax.plot(X, np.sin(X), "C0", label="Sine")
ax.plot(X, np.cos(X), "C1", label="Cosine")
ax.legend(bbox_to_anchor=(0,1,1,.1), ncol=2,
mode="expand", loc="lower left")
```



### Annotation

```
ax.annotate("A", (X[250],Y[250]), (X[250],-1),
ha="center", va="center", arrowprops =
{"arrowstyle": "->", "color": "C1"})
```



### Colors

Any color can be used but Matplotlib offers sets of colors:

C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

### Size & DPI

Consider a square figure to be included in a two-columns A4 paper with 2cm margins on each side and a column separation of 1cm. The width of a figure is  $(21 - 2*2 - 1)/2 = 8\text{cm}$ . One inch being 2.54cm, figure size should be  $3.15 \times 3.15\text{ in}$ .

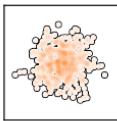
```
fig = plt.figure(figsize=(3.15,3.15), dpi=50)
plt.savefig("figure.pdf", dpi=600)
```

## Matplotlib tips & tricks

### Transparency

Scatter plots can be enhanced by using transparency (alpha) in order to show area with higher density and multiple scatter plots can be used to delineate a frontier.

```
X = np.random.normal(-1, 1, 500)
Y = np.random.normal(-1, 1, 500)
ax.scatter(X, Y, 50, "0.0", lw=2) # optional
ax.scatter(X, Y, 50, "1.0", lw=0) # optional
ax.scatter(X, Y, 40, "C1", lw=0, alpha=0.1)
```



### Rasterization

If your figure is made of a lot graphical elements such as a huge scatter, you can rasterize them to save memory and keep other elements in vector format.

```
X = np.random.normal(-1, 1, 10_000)
Y = np.random.normal(-1, 1, 10_000)
ax.scatter(X, Y, rasterized=True)
fig.savefig("rasterized-figure.pdf", dpi=600)
```

### Offline rendering

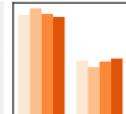
Use the Agg backend to render a figure directly in an array.

```
from matplotlib.backends.backend_agg import FigureCanvas
canvas = FigureCanvas(Figure())
... # draw som stuff
canvas.draw()
Z = np.array(canvas.renderer.buffer_rgba())
```

### Range of continuous colors

You can use colormap to pick a range of continuous colors.

```
X = np.random.randn(1000, 4)
cmap = plt.get_cmap("Blues")
colors = [cmap(i) for i in [.2,.4,.6,.8]]
ax.hist(X, 2, histtype='bar', color=colors)
```



### Text outline

Use text outline to make text more visible.

```
import matplotlib.path_effects as fx
text = ax.text(0.5, 0.1, "Label")
text.set_path_effects([
    fx.Stroke(linewidth=3, foreground='1.0'),
    fx.Normal()])
```



### Multiline plot

You can plot several lines at once using None as separator.

```
X, Y = [], []
for x in np.linspace(0, 10*np.pi, 100):
    X.extend([x, x, None]), Y.extend([0, sin(x), None])
ax.plot(X, Y, "black")
```

### Dotted lines

To have rounded dotted lines, use a custom linestyle and modify dash\_capstyle.

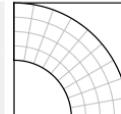
```
ax.plot([0,1], [0,0], "C1",
        linestyle = (0, (0.01, 1)), dash_capstyle="round")
ax.plot([0,1], [1,1], "C1",
        linestyle = (0, (0.01, 2)), dash_capstyle="round")
```



### Combining axes

You can use overlaid axes with different projections.

```
ax1 = fig.add_axes([0,0,1,1],
                   label="cartesian")
ax2 = fig.add_axes([0,0,1,1],
                   label="polar",
                   projection="polar")
```



### Colorbar adjustment

You can adjust colorbar aspect when adding it.

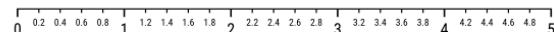
```
im = ax.imshow(Z)
cb = plt.colorbar(im,
                  fraction=0.046, pad=0.04)
cb.set_ticks([])
```



### Taking advantage of typography

You can use a condensed face such as Roboto Condensed to save space on tick labels.

```
for tick in ax.get_xticklabels(which='both'):
    tick.set_fontname("Roboto Condensed")
```



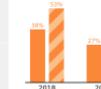
### Getting rid of margins

Once your figure is finished, you can call `tight_layout()` to remove white margins. If there are remaining margins, you can use the `pdfcrop` utility (comes with TeX live).

### Hatching

You can achieve nice visual effect with thick hatch patterns.

```
cmap = plt.get_cmap("Oranges")
plt.rcParams['hatch.color'] = cmap(0.2)
plt.rcParams['hatch.linewidth'] = 8
ax.bar(X, Y, color=cmap(0.6), hatch="/")
```



### Read the documentation

Matplotlib comes with an extensive documentation explaining every details of each command and is generally accompanied by examples with. Together with the huge online gallery, this documentation is a gold-mine.

Matplotlib 3.2 handout for tips & tricks. Copyright (c) 2020 Nicolas P. Rougier. Released under a CC-BY 4.0 License. Supported by NumFocus Grant #12345.

# matplotlib

Cheat sheet Version 3.2

## Quick start

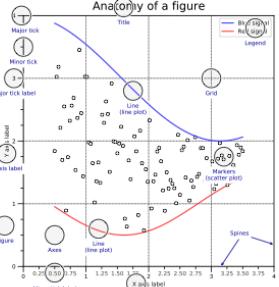
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt

X = np.linspace(0, 2*np.pi, 100)
Y = np.cos(X)

fig, ax = plt.subplots()
ax.plot(X,Y,color='C1')

fig.savefig("figure.pdf")
fig.show()
```

## Anatomy of a figure



## Subplots layout

```
subplots([cols,rows,...])
fig, axs = plt.subplots(3,3)

G = gridspec([cols,rows,...])
ax = G[0,:]

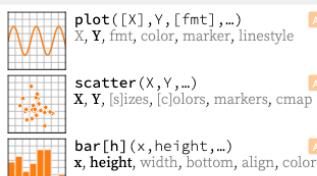
ax.inset_axes(extent)

d=make_axes_locatable(ax)
ax=d.new_horizontal('10%')
```

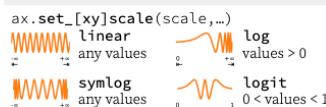
## Getting help

- [matplotlib.org](http://matplotlib.org)
- [github.com/matplotlib/matplotlib/issues](https://github.com/matplotlib/matplotlib/issues)
- [discourse.matplotlib.org](https://discourse.matplotlib.org)
- [stackoverflow.com/matplotlib](https://stackoverflow.com/matplotlib)
- [gitter.im/matplotliblib](https://gitter.im/matplotliblib)
- [twitter.com/matplotlib](https://twitter.com/matplotlib)
- [Matplotlib users mailing list](mailto:Matplotlib users mailing list)

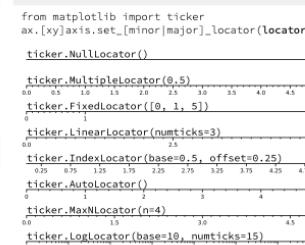
## Basic plots



## Scales



## Tick locators



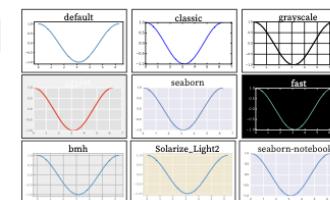
## Animation

```
import matplotlib.animation as plt

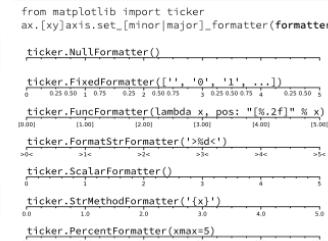
T = np.linspace(0,2*np.pi,100)
S = np.sin(T)
line, = plt.plot(T,S)
def animate(i):
    line.set_ydata(np.sin(T+i/50))
anim = plt.FuncAnimation(
    plt.gcf(), animate, interval=5)
plt.show()
```

## Styles

plt.style.use(style)



## Tick formatters



## Quick reminder

```
ax.grid()
ax.patch.set_alpha(0)
ax.set_[xy]lim(vmin, vmax)
ax.set_[xy]label(label)
ax.set_[xy]ticks(list)
ax.set_[xy]ticklabels(list)
ax.set_[sup]title(title)
ax.tick_params(width=10, ...)
ax.set_axis_[on/off]()
```

```
ax.tight_layout()
plt.gcf(), plt.gca()
mpl.rcParams['axes', linewidth=1, ...]
fig.patch.set_alpha(0)
text=r'$\frac{-e^{\pi i}}{n}$'
```

## Keyboard shortcuts

<b>ctrl + S</b>	Save	<b>ctrl + W</b>	Close plot
<b>r</b>	Reset view	<b>f</b>	Fullscreen 0/1
<b>f</b>	View forward	<b>b</b>	View back
<b>p</b>	Pan view	<b>o</b>	Zoom to rect
<b>x</b>	X pan/zoom	<b>y</b>	Y pan/zoom
<b>g</b>	Minor grid 0/1	<b>G</b>	Major grid 0/1
<b>l</b>	X axis log/linear	<b>L</b>	Y axis log/linear

## Ten Simple Rules

READ

1. Know Your Audience
2. Identify Your Message
3. Adapt the Figure
4. Captions Are Not Optional
5. Do Not Trust the Defaults
6. Use Color Effectively
7. Do Not Mislead the Reader
8. Avoid "Chartjunk"
9. Message Trumps Beauty
10. Get the Right Tool

## Event handling

```
fig, ax = plt.subplots()
def on_click(event):
    print(event)
fig.canvas.mpl_connect(
    'button_press_event', on_click)
```

## Advanced plots



## Colors



## Colormaps

plt.get\_cmap(name)



