kinemathic Documentation

Release 0.2.0

Francois Dion

Apr 25, 2018

CONTENTS:

1 kinematic **1**

2 Tweening two lines **3**

3 Kinematic Modules **7**
- 3.1 kinematic 7
- 3.2 `animation` 7
- 3.3 `resample` 9

4 Indices and tables **11**

Python Module Index **13**

Index **15**

CHAPTER

ONE

KINEMATHIC

Tool to generate Kinematic Displays

```
In [1]: import pandas as pd
        from kinematic.animation import ianimate, tweening

In [2]: df = pd.read_excel('../datasets/WPP2015_FERT_F01_BIRTHS_BOTH_SEXES.XLS', skiprows=16, index_c

In [3]: df.rename(columns={df.columns[2]:'Description'}, inplace=True)

In [4]: df.drop(df.columns[[0, 1, 3]], axis=1, inplace=True)

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

Out[5]:

Description	1950-1955	\
Country code		
900	WORLD	487364.363
901	More developed regions	93850.232
902	Less developed regions	393514.131
941	Least developed countries	49463.116
934	Less developed regions, excluding least develo...	344051.015

	1955-1960	1960-1965	1965-1970	1970-1975	1975-1980	\
Country code						
900	512937.177	561132.795	597172.105	611175.699	607055.970	
901	95448.675	92629.195	85134.137	82271.418	78923.492	
902	417488.502	468503.600	512037.968	528904.281	528132.478	
941	54921.227	61422.115	68881.235	76829.959	85551.485	
934	362567.275	407081.485	443156.733	452074.322	442580.993	

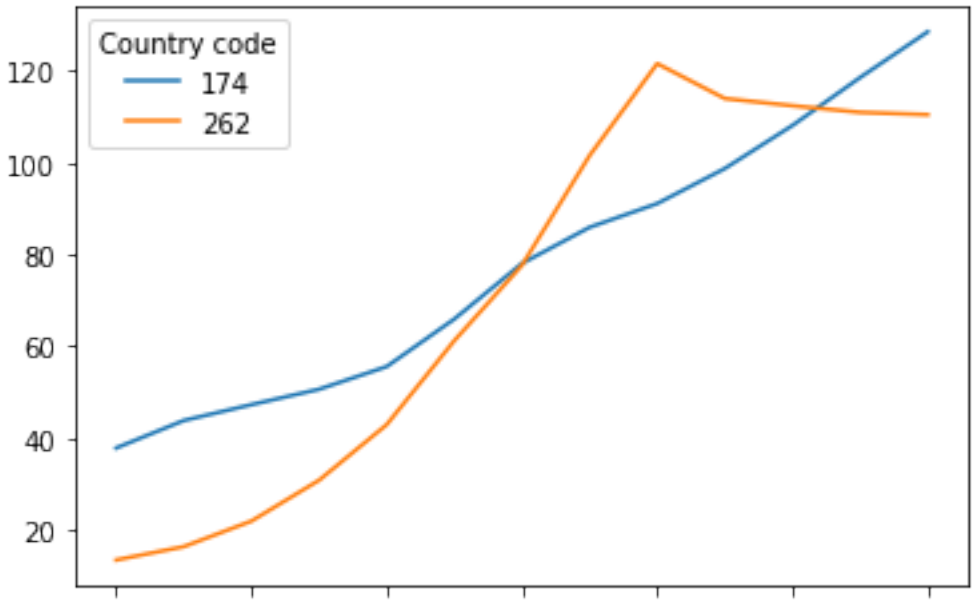
	1980-1985	1985-1990	1990-1995	1995-2000	2000-2005	\
Country code						
900	646458.598	698316.802	676437.337	650312.457	658430.353	
901	79161.433	78297.163	71535.899	66248.331	66076.568	
902	567297.165	620019.639	604901.438	584064.126	592353.785	
941	95364.057	104201.246	113396.831	123647.646	133323.975	
934	471933.108	515818.393	491504.607	460416.480	459029.810	

	2005-2010	2010-2015
Country code		
900	680566.922	699213.776
901	69551.907	68687.122
902	611015.015	630526.654
941	141890.152	149996.813
934	469124.863	480529.841

```
In [6]: test = df[df.index <900][1:3]

In [7]: test.T[1:].plot()
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7f47f8cc7e10>



In [8]: test

Out[8]:

Description	1950-1955	1955-1960	1960-1965	1965-1970	\
Country code					
174	Comoros	37.869	43.858	47.306	50.659
262	Djibouti	13.491	16.392	21.958	30.850

	1970-1975	1975-1980	1980-1985	1985-1990	1990-1995	\
Country code						
174	55.594	65.953	78.042	85.920	91.089	
262	42.948	61.240	77.648	101.534	121.562	

	1995-2000	2000-2005	2005-2010	2010-2015
Country code				
174	98.722	108.099	118.535	128.497
262	113.961	112.392	110.948	110.445

CHAPTER

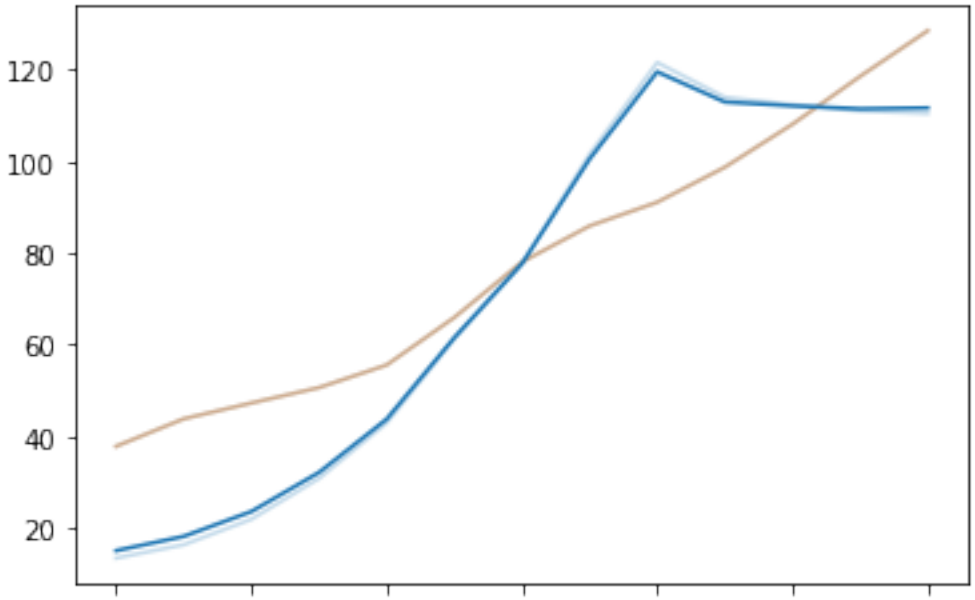
TWO

TWEENING TWO LINES

```
In [9]: tweening(test, 'test_30.mp4', fps=10, transitions=30)
```

```
dropping Description
```

```
Out[9]: <IPython.core.display.HTML object>
```




```
In [10]: import statsmodels.api as sm
```

```
/home/fdion/anaconda3_5/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarning: T
  from pandas.core import datetools
```

```
In [11]: data = sm.datasets.co2.load_pandas()
```

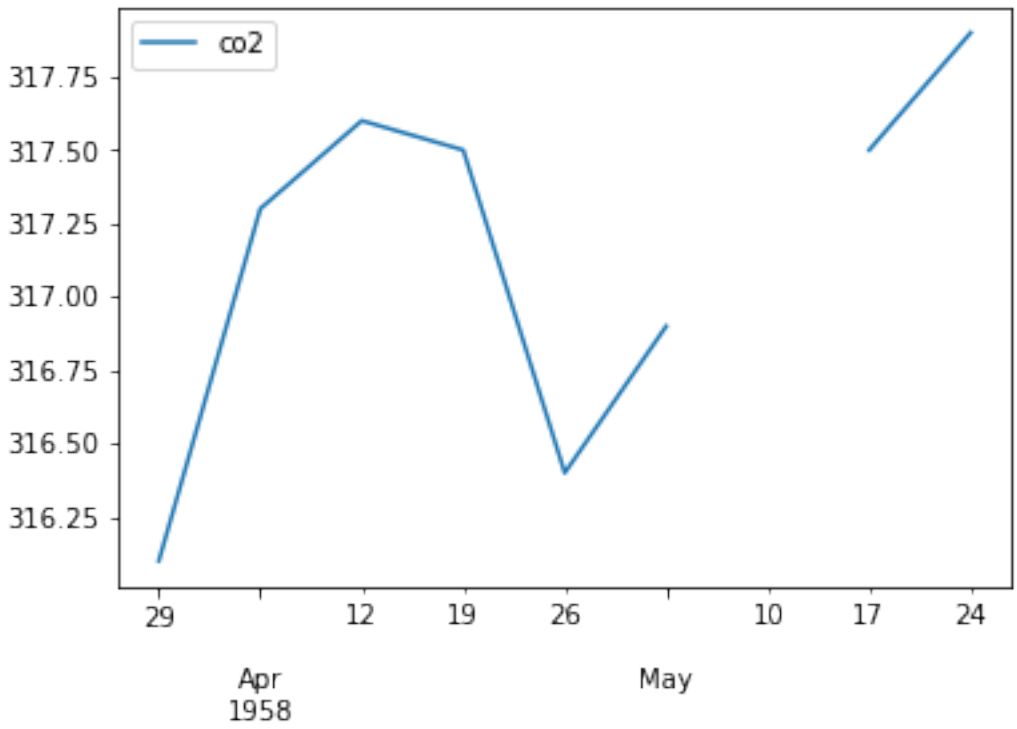
```
In [12]: co2 = data.data
```

```
In [13]: co2.head()
```

```
Out[13]: co2
         1958-03-29  316.1
         1958-04-05  317.3
         1958-04-12  317.6
         1958-04-19  317.5
         1958-04-26  316.4
```

```
In [14]: co2[0:11].plot()
```

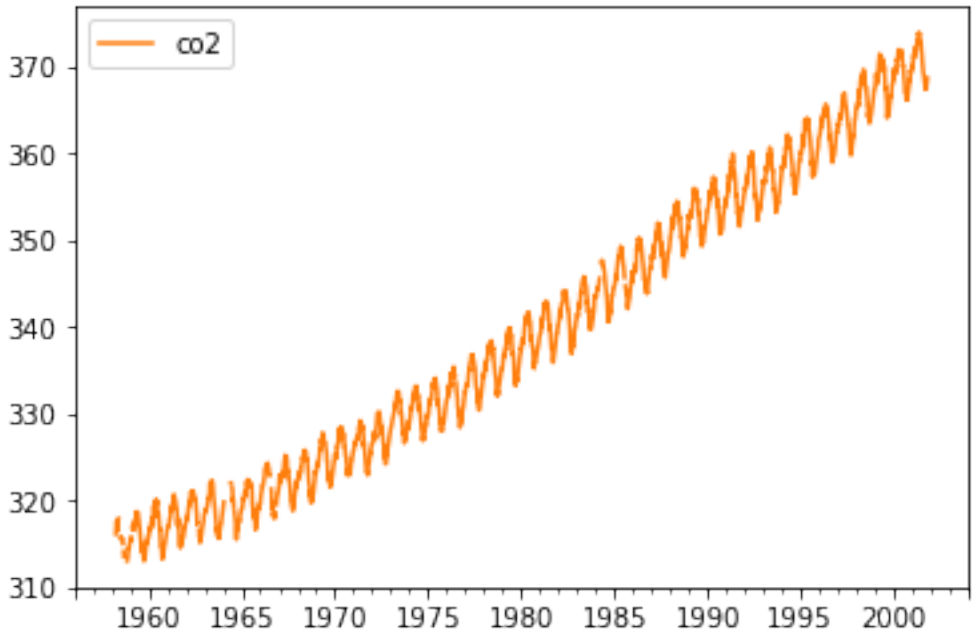
```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f47e64e5470>
```




```
In [15]: ## Time series animation
```

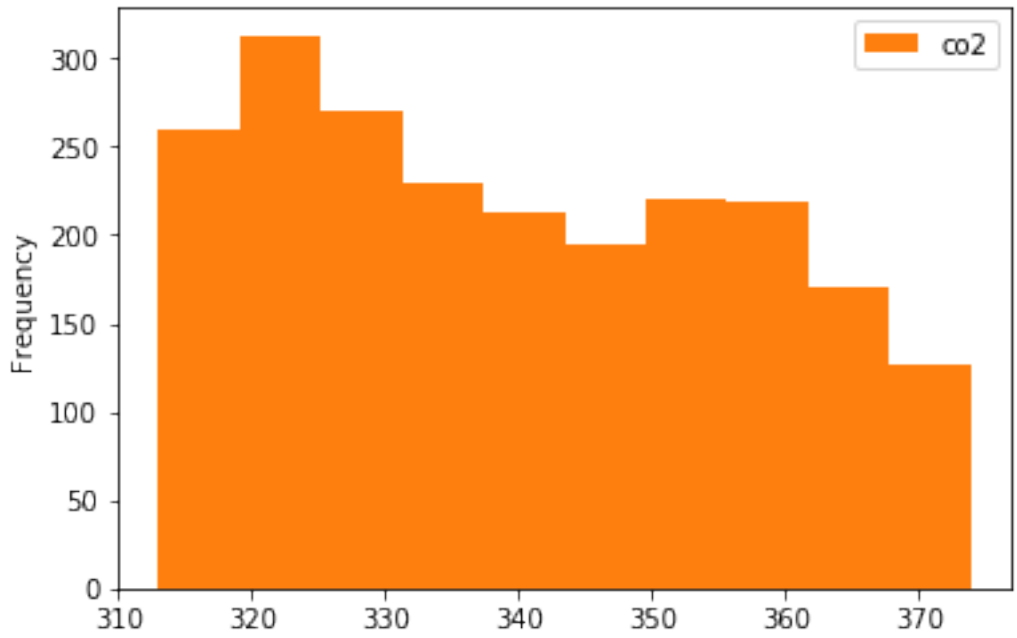
```
In [16]: ianimate(co2, n=len(co2), axis=1, step=25, autoscale=True)
```

```
Out[16]: <IPython.core.display.HTML object>
```




```
In [17]: ianimate(co2, filename='hist.mp4', n=len(co2), axis=1, step=25, autoscale=True, kind='hist')
```

```
Out[17]: <IPython.core.display.HTML object>
```


co2
Frequency
300
250
200
150
100
50
0
310
320
330
340
350
360
370

CHAPTER

THREE

KINEMATHIC MODULES

3.1 kinemathic

3.2 animation

kinemathic.animation.**animate**(*df*, *filename*, *autoscale=True*, *ax=None*, *axis=0*, *fps=5*, *kind=None*, *n=None*, *rx=None*, *ry=None*, *step=1*, *title=None*)

animate.

Animate plots and save to mp4 video. :param df: pandas dataframe, required :param filename: file name to save the mp4 file to, required :param autoscale: bool, if True will adjust the scale as data is processed, else will pre-render full scale :param ax: matplotlib ax :param axis: 0 or 1 for horizontal or vertical data :param fps: frames per second - defaults to 5 :param kind: pandas plot kind :param n: frame to render :param rx: remove x axis :param ry: remove y axis :param step: how many steps between frames :param title: optional, title for the plot :return: N/A

kinemathic.animation.**ianimate**(*df*, *autoscale=True*, *ax=None*, *axis=0*, *filename=None*, *fps=5*, *kind=None*, *n=None*, *rx=None*, *ry=None*, *step=1*, *title=None*)

ianimate.

Animate plots, output to jupyter notebook.

Parameters

- **df** – pandas dataframe
- **autoscale** – bool, if True will adjust the scale as data is processed, else will pre-render full scale
- **ax** – matplotlib ax
- **axis** – 0 or 1 for horizontal or vertical data
- **filename** – file name to save the mp4 file to
- **fps** – frames per second - defaults to 5
- **kind** – pandas plot kind
- **n** – frame to render
- **rx** – remove x axis
- **ry** – remove y axis
- **step** – how many steps between frames
- **title** – optional, title for the plot

Returns HTML video control widget for jupyter notebook

kinematic.animation.**tweening**(*df*, *filename*, *autoscale=True*, *axis=0*, *fps=5*, *kind=None*, *rx=None*, *ry=None*, *step=1*, *title=None*, *transitions=10*)

tweening.

Parameters

- **df** – pandas dataframe, required
- **filename** – file name to save the mp4 file to, required
- **autoscale** – bool, if True will adjust the scale as data is processed, else will pre-render full scale
- **axis** – 0 or 1 for horizontal or vertical data
- **fps** – frames per second - defaults to 5
- **kind** – pandas plot kind
- **rx** – remove x axis
- **ry** – remove y axis
- **step** – how many steps between frames
- **title** – optional, title for the plot
- **transitions** – how many interimary values to generate

Returns HTML video control widget for jupyter notebook

kinematic.animation.**update_figure**(*df1*, *n*, *autoscale=False*, *ax=None*, *axis=0*, *grayed=None*, *j=0*, *kind=None*, *rx=None*, *ry=None*, *start=0*, *title=None*)

update_figure.

Recalculates the whole plot for frame n.

Parameters

- **df1** – pandas dataframe, required
- **n** – frame to render, required
- **autoscale** – bool, if True will adjust the scale as data is processed, else will pre-render full scale
- **ax** – matplotlib ax
- **axis** – 0 or 1 for horizontal or vertical data
- **grayed** – grayed out “background” data
- **j** – this is used to calculate if we are dealing with an even or odd data series, for color selection.
- **kind** – pandas plot kind
- **rx** – remove x axis
- **ry** – remove y axis
- **start** – will be used in 0.3 for offset
- **title** – optional, title for the plot

Returns affects ax

3.3 resample

kinematic.resample.**between_rows**(*df*, *n=10*, *skip_objects=True*)
between_rows.

Create new rows between existing rows.

Parameters

- **df** – pandas dataframe
- **n** – number of rows to insert
- **skip_objects** – Bool, if True, columns of type object will be skipped.

Returns transformed dataframe

kinematic.resample.**num_only**(*df*)
num_only.

Remove non numerical columns from a dataframe

Parameters **df** – pandas dataframe

Returns transformed dataframe

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

k

kinematic.__init__, 7
kinematic.animation, 7
kinematic.resample, 9

INDEX

A

animate() (in module kinematic.animation), 7

B

between_rows() (in module kinematic.resample), 9

I

ianimate() (in module kinematic.animation), 7

K

kinematic.__init__ (module), 7
kinematic.animation (module), 7
kinematic.resample (module), 9

N

num_only() (in module kinematic.resample), 9

T

tweening() (in module kinematic.animation), 8

U

update_figure() (in module kinematic.animation), 8