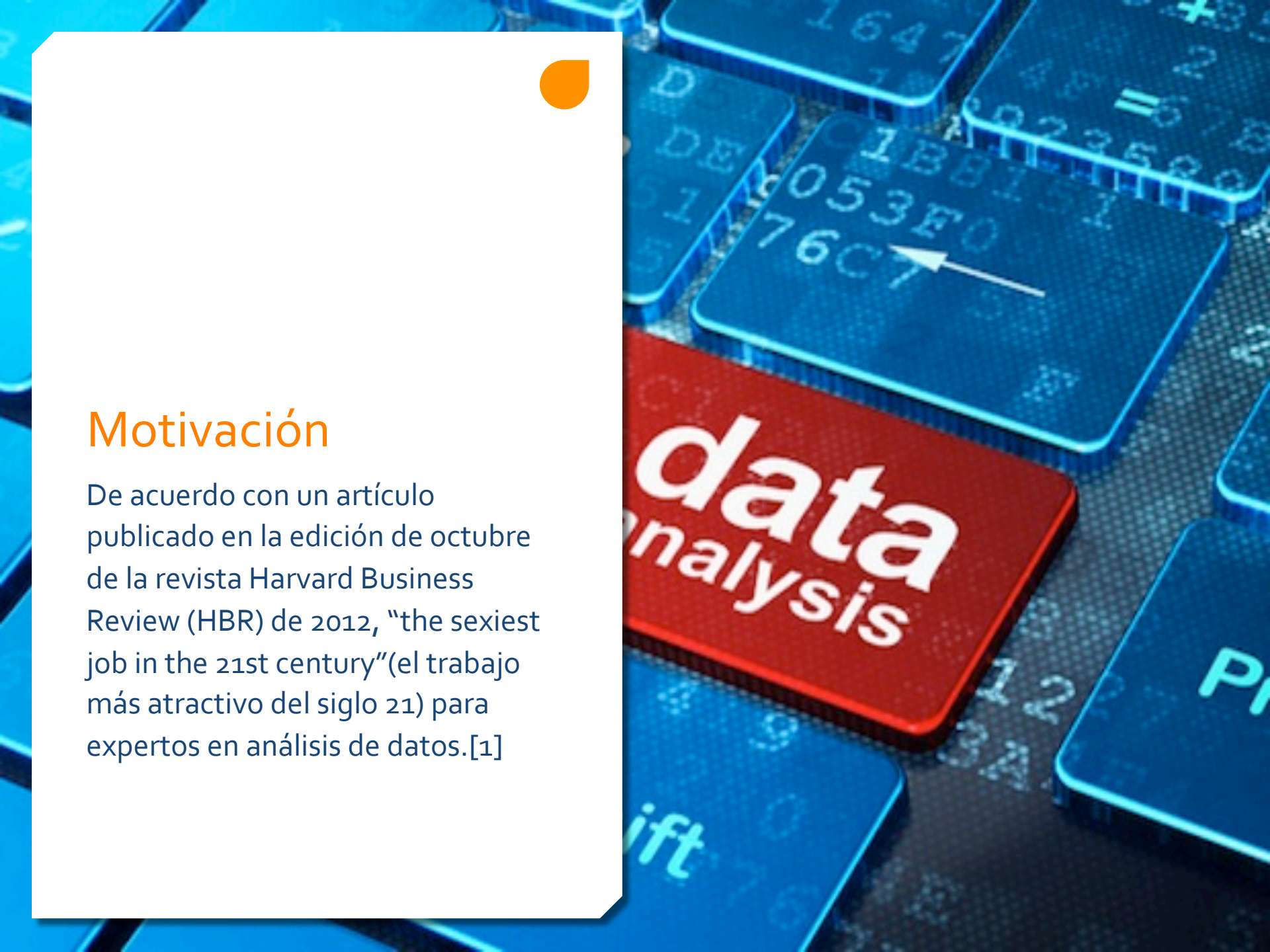
# Análisis de Datos

#PyDay Loja-Ecuador
@marlonsvl

# Motivación

De acuerdo con un artículo publicado en la edición de octubre de la revista Harvard Business Review (HBR) de 2012, "the sexiest job in the 21st century"(el trabajo más atractivo del siglo 21) para expertos en análisis de datos.[1]



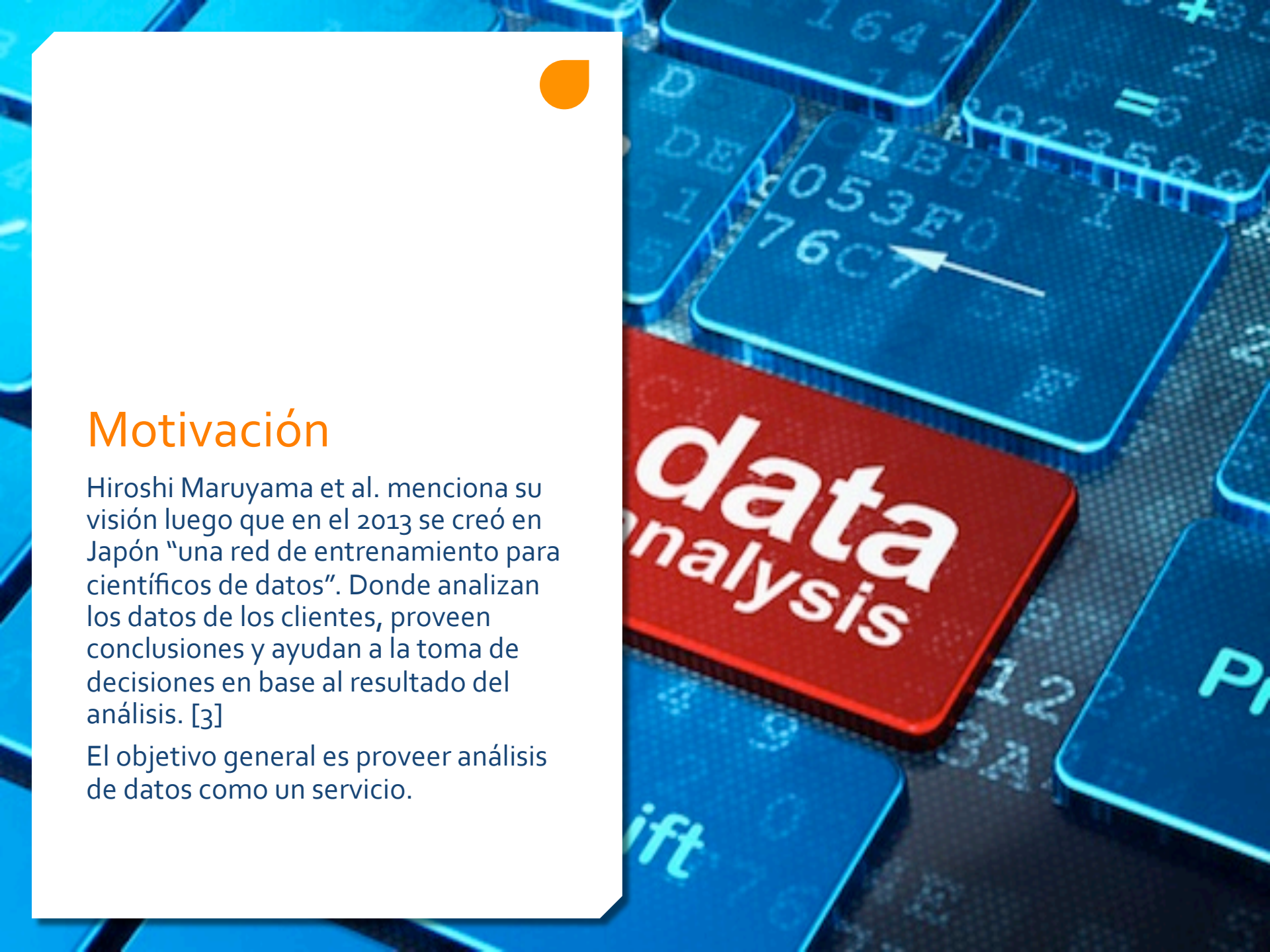
# Motivación

En un informe de McKinsey Global Institute predice que para el año 2018 habrá una escasez de talento en análisis de datos de entre 140,000 y 190,000 en los EEUU. [2]

# Motivación

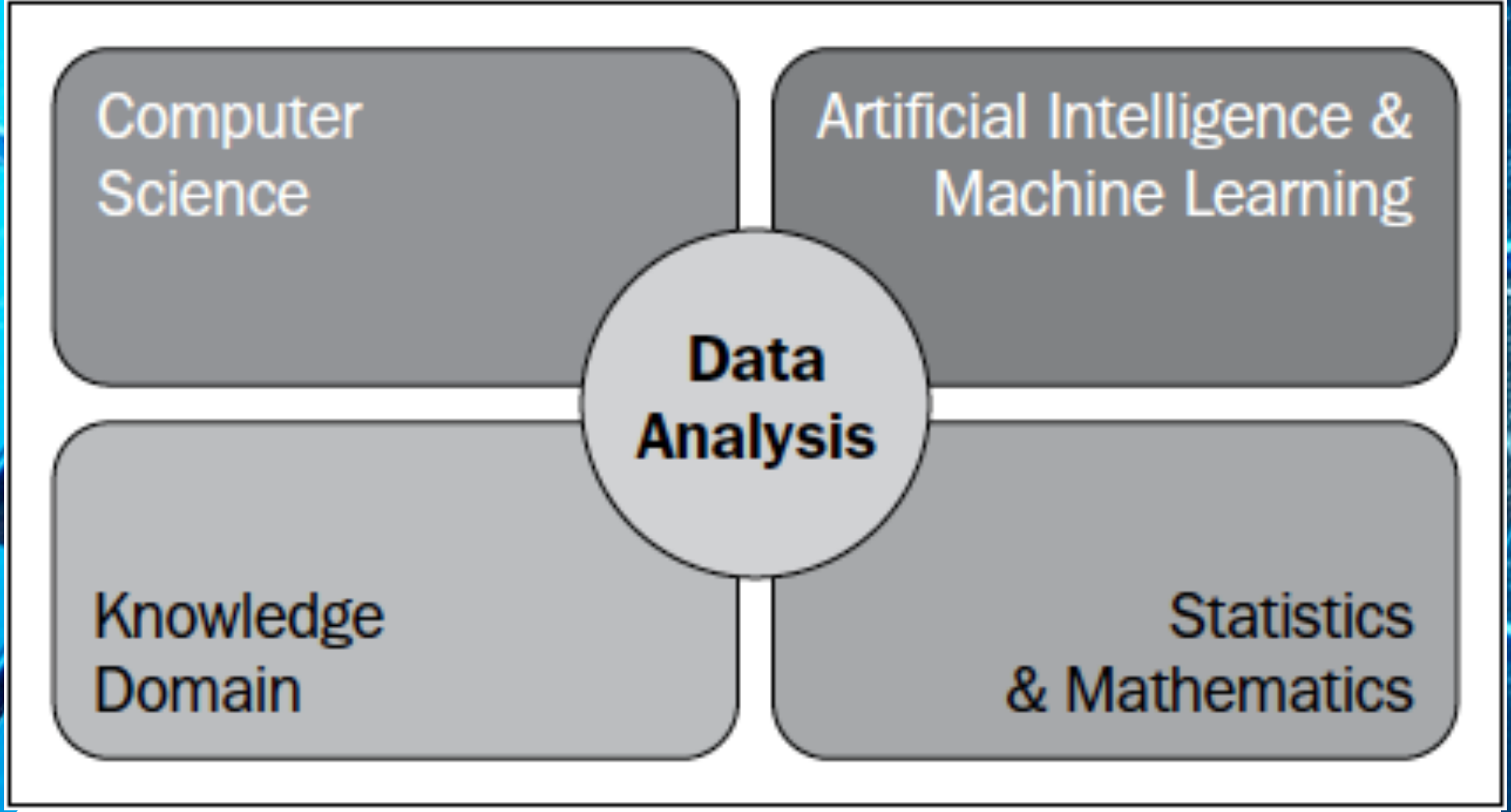
Hiroshi Maruyama et al. menciona su visión luego que en el 2013 se creó en Japón "una red de entrenamiento para científicos de datos". Donde analizan los datos de los clientes, proveen conclusiones y ayudan a la toma de decisiones en base al resultado del análisis. [3]

El objetivo general es proveer análisis de datos como un servicio.

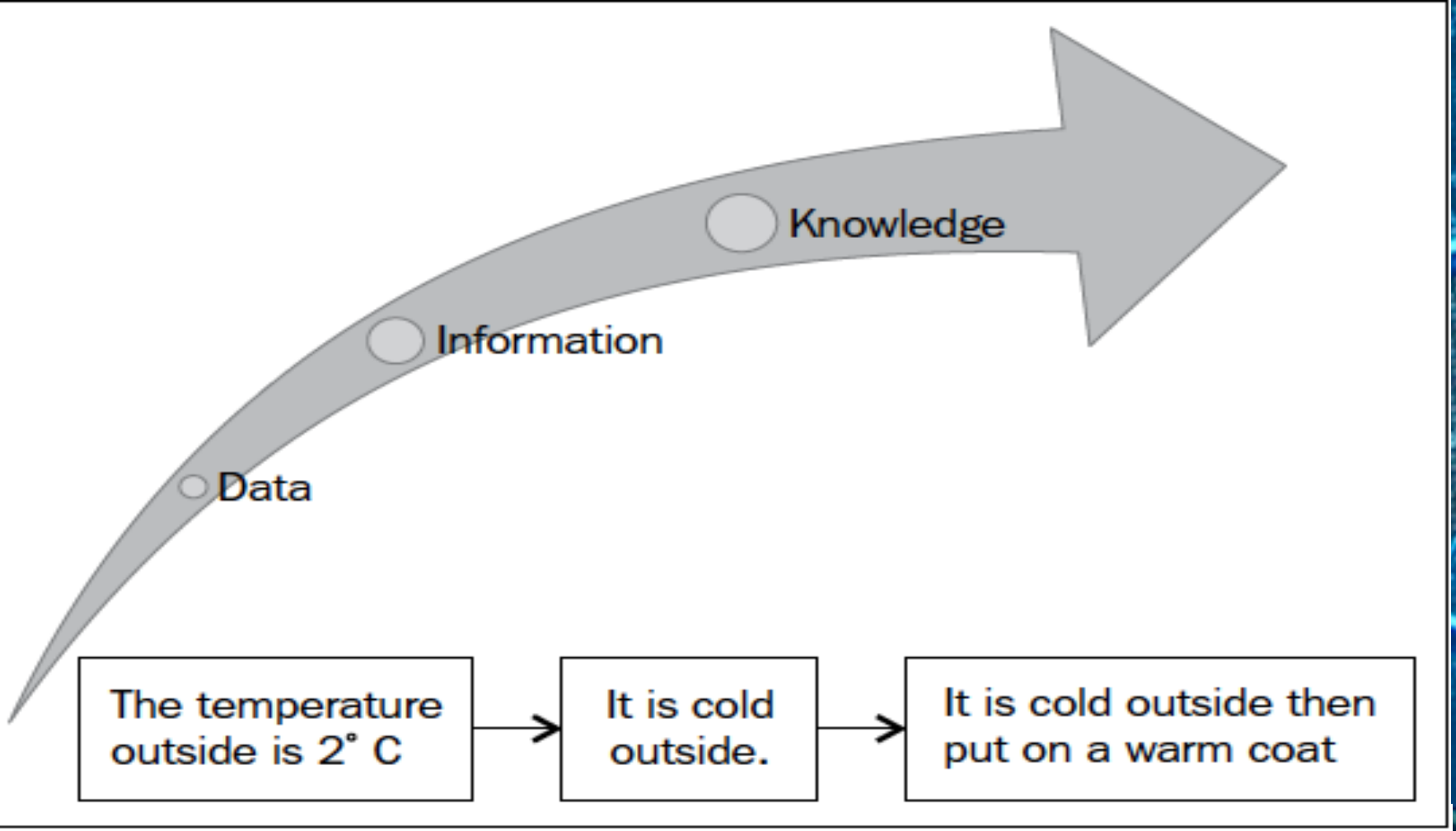



# Análisis de Datos

Es el proceso de inspección, limpieza, transformación y modelado de datos con el objetivo de descubrir información útil de tal manera que permita sacar conclusiones y da soporte a la toma de decisiones.



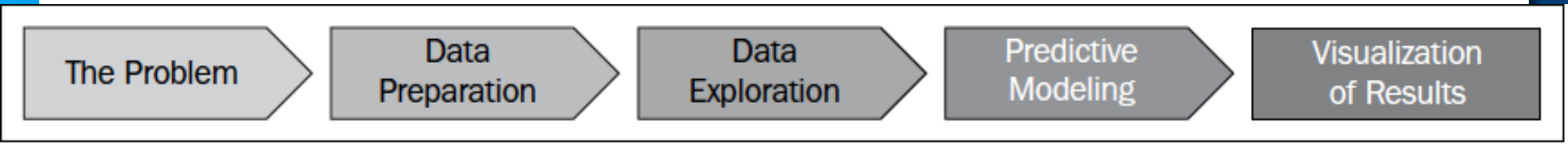

Análisis de datos es un campo multidisciplinario




# Datos, información y conocimiento

# El proceso de análisis de datos

- El problema
- Extracción de datos
- Limpieza de datos
- Normalización de datos
- Transformación de datos
- Análisis estadístico
- Visualización
- Modelo predictivo
- Visualización e interpretación de datos



# Proceso

- ***Interactuando con el mundo exterior:*** Leyendo y escribiendo con una variedad de formatos de archivos y bases de datos.
- ***Preparación:*** Cleaning, munging, combining, normalizing, reshaping, slicing and dicing.
- ***Transformación:*** Aplicando operaciones matemáticas y estadísticas para agrupar conjuntos de datos para derivar en nuevos data sets.
- ***Modelización y computación:*** Conectando tus datos a modelos estadísticos, machine learning algorithms u otras herramientas
- ***Presentación:*** Creación de visualizaciones estáticas o interactivas y resúmenes textuales.


# Porque python?

# NumPy (Numerical Python)

- Funciones matemáticas estándar para operaciones rápidas en matrices enteras de datos sin tener que escribir bucles.
- Herramientas para la lectura / escritura de datos en disco de matrices y trabajar con archivos asignados a la memoria.
- Álgebra lineal, la generación de números aleatorios, y la transformada de Fourier.
- Herramientas para la integración de código escrito en C, C++ y Fortran.
- Sin embargo, para datos estructurados o tabulares es mejor el uso de Panda Library.

# pandas

- El nombre `pandas` está derivado de **panel data** un término de la econometría para los conjuntos de datos estructurados multidimensionales.
- Munge/Munging/Wrangling: Describe el proceso general de la manipulación de los datos no estructurados y / o desordenados en una forma estructurada o limpio. La palabra se ha colado su camino en la jerga de muchos hackers de datos de hoy en día. Munge rima con (lunge) ->"embestida".

# pandas

- Combina las características del  paquete Numpy con la flexible manipulación de datos de hojas de cálculos y bases de datos relacionales (como SQL).
- Provee funcionalidades de indexación sofisticadas para remodelar, dividir, agrupar y seleccionar subconjuntos de datos.
- Fue creado principalmente para aplicaciones de análisis de datos financieros.
- La principal estructura de datos de esta librería es el *DataFrame* que representa una tabla o una hoja de cálculo(excel ó csv).

# Matplotlib

- 2D Data visualizations
- Plots interactive

# Instalación (Apple OS X)

- Tener instalado Xcode
- Instalar EPDFree (Scientific python distribution). Al instalarse se agrega automáticamente la ruta del ejecutable de EPDFree al .bash_profile.

```
# Setting PATH for EPD_free-7.3-1
PATH="/Library/Frameworks/Python.framework/Versions/Current/bin:${PATH}"
export PATH
```

- Ejecutar: sudo easy_install pandas
- Ipython --pylab

# Instalación GNU/Linux (Debian)

- Descargar el instalador de EPDFree (32-bit o 64-bit) y ejecutar: bash epd_free-7.3-1-rh5-x86_64.sh
- Es necesario agregar el bin de EPDFree a $PATH variable o ejecutar: export PATH=/home/username/epd/bin:$PATH
- Ejecutar: source ~/.bashrc
- Ejecutar: sudo apt-get install gcc
- Ejecutar: easy_install pandas


# Ejemplos

Análisis de datos de series de tiempo y financieros

# Data Aligment – Time Series

```
In [16]: prices
Out[16]:
              AAPL    JNJ      SPX    XOM
2011-09-06  379.74  64.64  1165.24  71.15
2011-09-07  383.93  65.43  1198.62  73.65
2011-09-08  384.14  64.95  1185.90  72.82
2011-09-09  377.48  63.64  1154.23  71.01
2011-09-12  379.94  63.59  1162.27  71.84
2011-09-13  384.62  63.61  1172.87  71.65
2011-09-14  389.30  63.73  1188.68  72.64

In [17]: volume
Out[17]:
                AAPL       JNJ       XOM
2011-09-06  18173500  15848300  25416300
2011-09-07  12492000  10759700  23108400
2011-09-08  14839800  15551500  22434800
2011-09-09  20171900  17008200  27969100
2011-09-12  16697300  13448200  26205800
```

# Data Aligment – Time Series

```
In [18]: prices * volume
Out[18]:
                 AAPL         JNJ  SPX         XOM
2011-09-06  6901204890  1024434112  NaN  1808369745
2011-09-07  4796053560   704007171  NaN  1701933660
2011-09-08  5700560772  1010069925  NaN  1633702136
2011-09-09  7614488812  1082401848  NaN  1986085791
2011-09-12  6343972162   855171038  NaN  1882624672
2011-09-13         NaN         NaN  NaN         NaN
2011-09-14         NaN         NaN  NaN         NaN

In [19]: vwap = (prices * volume).sum() / volume.sum()
```

```
In [20]: vwap
Out[20]:
AAPL    380.655181
JNJ      64.394769
SPX            NaN
XOM      72.024288
```

```
In [21]: vwap.dropna()
Out[21]:
AAPL    380.655181
JNJ      64.394769
XOM      72.024288
```

# Data Aligment –Time Series

```
In [22]: prices.align(volume, join='inner')
Out[22]:
(                AAPL    JNJ    XOM
2011-09-06  379.74  64.64  71.15
2011-09-07  383.93  65.43  73.65
2011-09-08  384.14  64.95  72.82
2011-09-09  377.48  63.64  71.01
2011-09-12  379.94  63.59  71.84,
                 AAPL        JNJ       XOM
2011-09-06  18173500  15848300  25416300
2011-09-07  12492000  10759700  23108400
2011-09-08  14839800  15551500  22434800
2011-09-09  20171900  17008200  27969100
2011-09-12  16697300  13448200  26205800)
```

# Operaciones sobre series de tiempo

- La técnica más importante para hacer inferencias sobre el futuro con base en lo ocurrido en el pasado, es el análisis de series de tiempo.
- Ejemplos:
  - Precios de un artículos
  - Tasas de desempleo
  - Tasa de inflación
  - Índice de precios

# Series de Tiempo

```
In [28]: ts1 = Series(np.random.randn(3),
   ....:              index=pd.date_range('2012-6-13', periods=3, freq='W-WED'))

In [29]: ts1
Out[29]:
2012-06-13   -1.124801
2012-06-20    0.469004
2012-06-27   -0.117439
Freq: W-WED
```

# Series temporales

```
In [30]: ts1.resample('B')
Out[30]:
2012-06-13   -1.124801
2012-06-14          NaN
2012-06-15          NaN
2012-06-18          NaN
2012-06-19          NaN
2012-06-20    0.469004
2012-06-21          NaN
2012-06-22          NaN
2012-06-25          NaN
2012-06-26          NaN
2012-06-27   -0.117439
Freq: B
```

# Series temporales

```
In [133]: ts1.resample('B', fill_method='ffill')
Out[133]:
2012-06-13    1.247252
2012-06-14    1.247252
2012-06-15    1.247252
2012-06-18    1.247252
2012-06-19    1.247252
2012-06-20   -1.062814
2012-06-21   -1.062814
2012-06-22   -1.062814
2012-06-25   -1.062814
2012-06-26   -1.062814
2012-06-27   -0.165290
Freq: B, dtype: float64
```

# Series temporales (reindex)

```
In [136]: dates = pd.DatetimeIndex(['2012-6-12', '2012-6-17', '2012-6-18','2012-6-21', '2012-6-22', '2012-6-29'])

In [137]: ts2 = Series(np.random.randn(6), index=dates)

In [138]: ts1
Out[138]:
2012-06-13    1.247252
2012-06-20   -1.062814
2012-06-27   -0.165290
Freq: W-WED, dtype: float64

In [139]: ts1.reindex(ts2.index, method='ffill')
Out[139]:
2012-06-12         NaN
2012-06-17    1.247252
2012-06-18    1.247252
2012-06-21   -1.062814
2012-06-22   -1.062814
2012-06-29   -0.165290
dtype: float64

In [140]:
```

# Splicing o empalme de dos DataSources

- En un contexto económico o financiero existen los siguientes casos de uso:
  1. Cambiar de una fuente de datos(una serie temporal o una colección de series de tiempo) a otro punto específico en el tiempo.
  2. "Patching" (parchar) valores faltantes en una serie de tiempo al comienzo, la mitad o al final usando otras series de tiempo.
  3. Sustituir completamente los datos por un subconjunto de símbolos (países, tickets de acciones, etc)

# Primer Caso: Saltando de un DatSource a otro en un punto en el tiempo

```
In [149]: data1 = DataFrame(np.ones((6, 3), dtype=float),
   .....: ....: columns=['a', 'b', 'c'],
   .....: ....: index=pd.date_range('6/12/2012', periods=6))

In [150]: data2 = DataFrame(np.ones((6, 3), dtype=float) * 2, columns=['a', 'b', 'c'],
   .....: index=pd.date_range('6/13/2012', periods=6))

In [151]: spliced = pd.concat([data1.ix[:'2012-06-14'], data2.ix['2012-06-15':]])

In [152]:

In [152]: spliced
Out[152]:
            a  b  c
2012-06-12  1  1  1
2012-06-13  1  1  1
2012-06-14  1  1  1
2012-06-15  2  2  2
2012-06-16  2  2  2
2012-06-17  2  2  2
2012-06-18  2  2  2

In [153]:
```

# Segundo Caso: Datos faltantes

```
In [157]: data2 = DataFrame(np.ones((6, 4), dtype=float) * 2, columns=['a', 'b', 'c', 'd'],
index=pd.date_range('6/13/2012', periods=6))

In [158]: spliced = pd.concat([data1.ix[:'2012-06-14'], data2.ix['2012-06-15':]])

In [159]: spliced
Out[159]:
            a  b  c   d
2012-06-12  1  1  1 NaN
2012-06-13  1  1  1 NaN
2012-06-14  1  1  1 NaN
2012-06-15  2  2  2   2
2012-06-16  2  2  2   2
2012-06-17  2  2  2   2
2012-06-18  2  2  2   2

In [160]: spliced_filled = spliced.combine_first(data2)

In [161]: spliced_filled
Out[161]:
            a  b  c   d
2012-06-12  1  1  1 NaN
2012-06-13  1  1  1   2
2012-06-14  1  1  1   2
2012-06-15  2  2  2   2
2012-06-16  2  2  2   2
2012-06-17  2  2  2   2
2012-06-18  2  2  2   2
```

# Tercer caso: Sustitución de Datos

```
In [163]: cp_spliced = spliced.copy()

In [164]: cp_spliced[['a', 'c']] = data1[['a', 'c']]

In [165]: cp_spliced
Out[165]:
            a  b   c   d
2012-06-12  1  1   1 NaN
2012-06-13  1  1   1 NaN
2012-06-14  1  1   1 NaN
2012-06-15  1  2   1   2
2012-06-16  1  2   1   2
2012-06-17  1  2   1   2
2012-06-18 NaN  2 NaN   2

In [166]:
```

# Datos Financieros (rendimiento del precio de activos)

```
In [181]: import pandas.io.data as web

In [182]: price = web.get_data_yahoo('AAPL', '2011-01-01')['Adj Close']

In [183]: returns = price.pct_change()

In [184]: ret_index = (1 + returns).cumprod()

In [185]: ret_index[0] = 1

In [186]: ret_index
Out[186]:
Date
2011-01-03    1.000000
2011-01-04    1.005219
2011-01-05    1.013442
2011-01-06    1.012622
2011-01-07    1.019874
2011-01-10    1.039081
2011-01-11    1.036623
2011-01-12    1.045059
2011-01-13    1.048882
2011-01-14    1.057378
2011-01-18    1.033620
2011-01-19    1.028128
2011-01-20    1.009437
2011-01-21    0.991352
2011-01-24    1.023910
2011-01-25    1.035895
2011-01-26    1.043329
```

# Datos financieros: (rendimiento de una frecuencia en particular)

```
In [188]: m_returns = ret_index.resample('BM', how='last').pct_change()

In [189]: m_returns['2012']
Out[189]:
Date
2012-01-31    0.127111
2012-02-29    0.188311
2012-03-30    0.105284
2012-04-30   -0.025970
2012-05-31   -0.010702
2012-06-29    0.010853
2012-07-31    0.045822
2012-08-31    0.093877
2012-09-28    0.002796
2012-10-31   -0.107600
2012-11-30   -0.012375
2012-12-31   -0.090743
Freq: BM, Name: Adj Close, dtype: float64

In [190]:
```

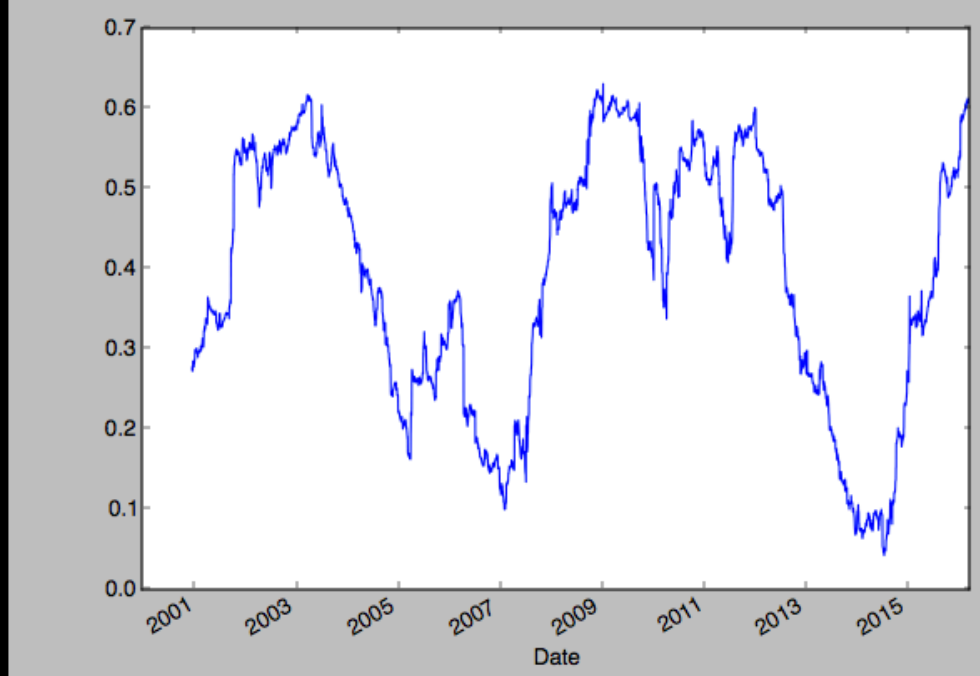
# Correlación entre Apple y Microsoft

1. python

bash | bash | bash | python | bash | bash

```
In [234]: aapl = web.get_data_yahoo('AAPL', '2000-01-01')['Adj Close']

In [235]: msft = web.get_data_yahoo('MSFT', '2000-01-01')['Adj Close']


In [236]:

In [236]: aapl_rets = aapl.pct_change()

In [237]: msft_rets = msft.pct_change()

In [238]: pd.rolling_corr(aapl_rets, msft_rets, 250).plot()
Out[238]: <matplotlib.axes._subplots.AxesSubplot at 0x10eeecc18>

In [239]:
```

Figure 1

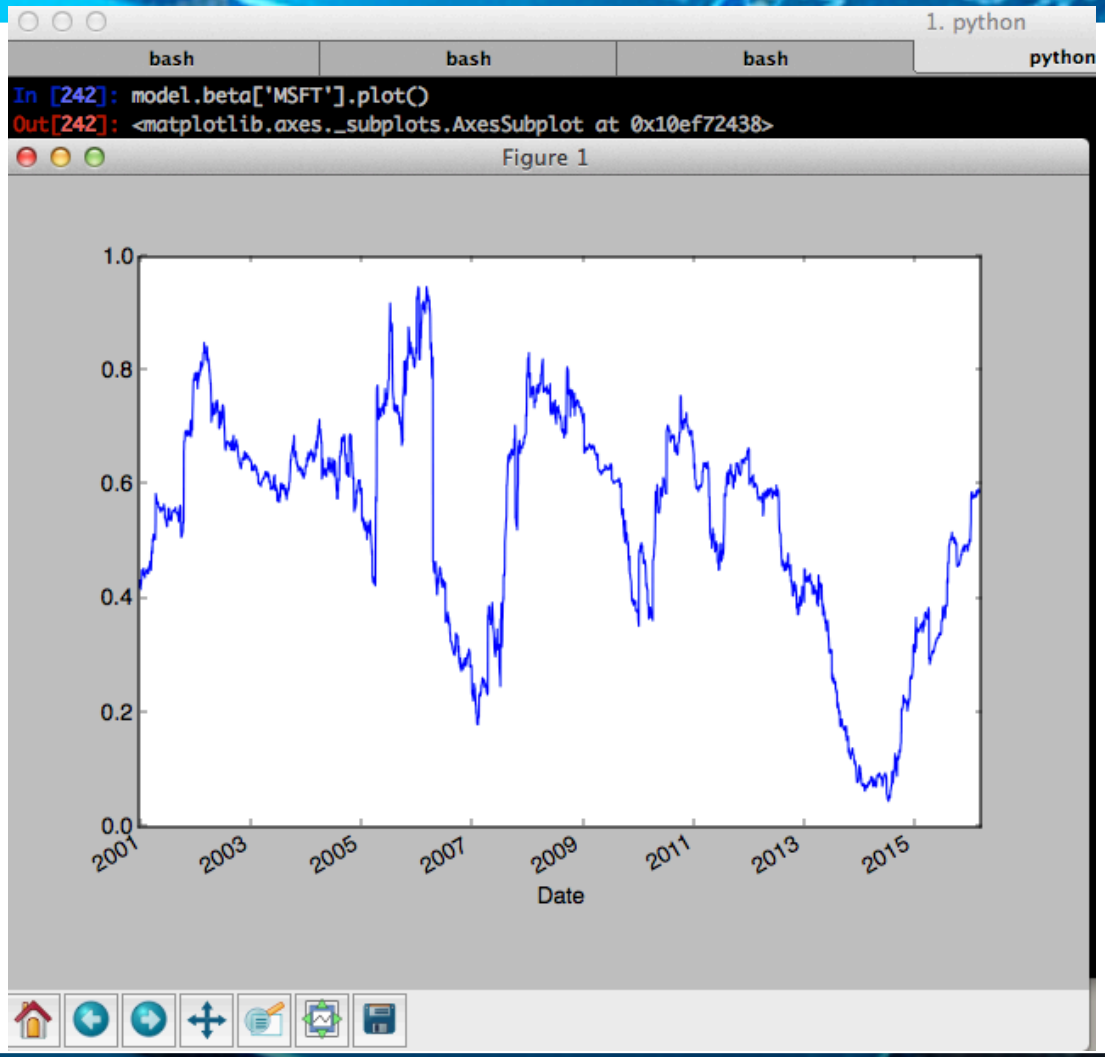



x=2005 y=0.283333

# Regresión lineal por mínimos cuadrados

```
In [240]: model = pd.ols(y=aapl_rets, x={'MSFT': msft_rets}, window=250)

In [241]: model.beta
Out[241]:
                MSFT  intercept
Date
2000-12-28  0.429021  -0.002113
2000-12-29  0.421103  -0.001796
2001-01-02  0.420596  -0.001839
2001-01-03  0.433292  -0.001289
2001-01-04  0.432772  -0.001307
2001-01-05  0.431028  -0.001414
2001-01-08  0.428223  -0.001209
2001-01-09  0.426630  -0.000978
2001-01-10  0.416963  -0.001586
2001-01-11  0.423251  -0.001376
2001-01-12  0.423460  -0.001603
2001-01-16  0.435644  -0.001784
2001-01-17  0.436671  -0.002144
```

# Regresión lineal por mínimos cuadrados



# Donde aplicar?

1. Bioinformática
2. Análisis periodístico.
3. Análisis financiero
4. Análisis Económico
5. Academia
6. Estadística
7. Social media
8. ...

# Referencias


1. Davenport, T. H. and Patil, D. J.: "Data Scientist: The Sexiest Job of the 21st Century." Harv. Bus. Rev., Vol. 90, No. 10, pp. 70–76 (201)

2. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H.: Big Data: The Next Frontier for Innovation, Competition, and Productivity, McKinsey & Company (2011).

3. Hiroshi MARUYAMA†1, Naoki KAMIYA†1, Tomoyuki HIGUCHI†1 and Akimichi TAKEMURA: Developing Data Analytics Skills in Japan: Status and Challenge(2015)

# Referencias

- Python for data analysis, WesMckinney (Book)


Preguntas ?


# Ejemplos (Github)

https://github.com/marlonsvl/analisisDeDatos_pyday2016