

INTRODUCCIÓN SERIES TEMPORALES

IRONHACK

Intro

Una serie temporal se define como una colección de observaciones de una variable recogidas secuencialmente en el tiempo. Estas observaciones se suelen recoger en instantes de tiempo equiespaciados. Si los datos se recogen en instantes temporales de forma continua, se debe o bien digitalizar la serie, es decir, recoger sólo los valores en instantes de tiempo equiespaciados, o bien acumular los valores sobre intervalos de tiempo.



Intro

Aparecen en numerosos campos. Ejemplos:

Economía y Marketing

- Precio del alquiler de pisos durante una serie de meses.
- Evolución del índice del precio del trigo con mediciones anuales.
- Beneficios netos mensuales de cierta entidad bancaria.
- Índices del precio del petróleo.



Intro

Demografía

- Número de habitantes en cierto país por año.
- Tasa de mortalidad infantil por año.

Medioambiente

- Evolución horaria de niveles de óxido de azufre y de niveles de óxido de nitrógeno en una ciudad durante una serie de años.
- Lluvia recogida diariamente en una localidad.
- Temperatura media mensual.
- Medición diaria del contenido en residuos tóxicos en un río.

Intro

La característica fundamental de las series temporales es que las observaciones sucesivas no son independientes entre sí, y el análisis debe llevarse a cabo teniendo en cuenta el orden temporal de las observaciones. Los métodos estadísticos basados en la independencia de las observaciones no son válidos para el análisis de series temporales porque las observaciones en un instante de tiempo dependen de los valores de la serie en el pasado.



Intro

Una serie temporal puede ser discreta o continua dependiendo de cómo sean las observaciones. Si se pueden predecir exactamente los valores, se dice que las series son determinísticas. Si el futuro sólo se puede determinar de modo parcial por las observaciones pasadas y no se pueden determinar exactamente, se considera que los futuros valores tienen una distribución de probabilidad que está condicionada a los valores pasados. Las series son así estocásticas.



Análisis básico

Cuando se estudia una serie temporal, lo primero que se tiene que hacer es dibujarla y considerar las medidas descriptivas básicas. Así, se tiene que considerar:

- Si los datos presentan forma creciente (tendencia).
- Si existe influencia de ciertos periodos de cualquier unidad de tiempo (estacionalidad).
- Si aparecen outliers (observaciones extrañas o discordantes).



Análisis básico

Vamos a verlo con un ejemplo, con el dataset AirPassengers. Este es un dataset sencillo que relaciona el día con el número de pasajeros totales de una aerolínea random para estudiar la evolución del número de pasajeros en el tiempo. Trataremos al número de pasajeros como variable continua cuantitativa.




```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

#Graficos interactivos

```
import chart_studio.plotly as py
import plotly.io as pio
import plotly.express as px
```

#Cosas de Series temporales

```
import statsmodels.api as sm
```

#test de estacionariedad

```
from statsmodels.tsa.stattools import adfuller, kpss
```

```
airpassengers = pd.read_csv('./Data/AirPassengers.csv')
airpassengers.head()
```

	TravelDate	Passengers
0	1/1/1949	112
1	2/1/1949	118
2	3/1/1949	132
3	4/1/1949	129
4	5/1/1949	121

Como siempre, chequeamos nuestro dataset para ver si hay algún valor vacío.

```
airpassengers.isnull().sum()
```

```
TravelDate    0  
Passengers    0  
dtype: int64
```

Como podemos ver, no tenemos ningún valor vacío en nuestro conjunto de datos, por lo que somos libres de continuar nuestro análisis. Ahora, lo que haremos es confirmar que la columna `TravelDate` está en formato de fecha y hora y no en string. El atributo `.dtypes` del `DataFrame` hace esto posible.

```
airpassengers.dtypes
```

```
TravelDate    object  
Passengers    int64  
dtype: object
```

Podemos ver que la columna TravelDate es de un tipo de objeto genérico que podría ser una string. Dado que queremos realizar acciones relacionadas con el tiempo en estos datos, debemos convertirlo a un formato de fecha y hora antes de que pueda sernos útil. Para ello está la función `to_datetime()`:

```
#Esto es crear un Timestamp
airpassengers['TravelDate'] = pd.to_datetime(airpassengers['TravelDate'])
airpassengers.head()
```

	TravelDate	Passengers
0	1949-01-01	112
1	1949-02-01	118
2	1949-03-01	132
3	1949-04-01	129
4	1949-05-01	121

```
airpassengers.dtypes
```

```
TravelDate    datetime64[ns]
Passengers      int64
dtype: object
```

Ahora, necesitamos establecer el objeto de fecha y hora como el índice del DataFrame para permitirnos realmente explorar nuestros datos. Para ello, usamos el método `.set_index()`:

```
airpassengers = airpassengers.set_index('TravelDate')  
airpassengers.head()
```

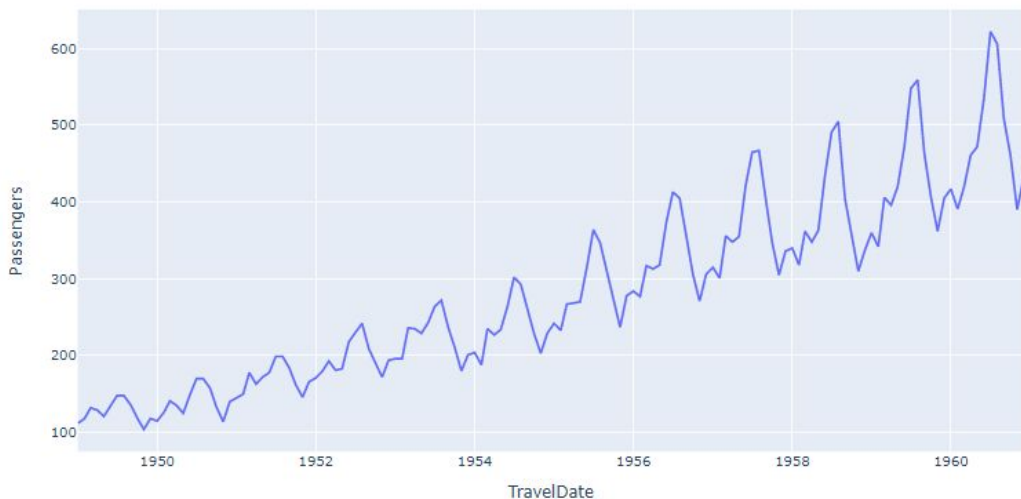
Passengers

TravelDate

1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

Podemos ver que ya es el índice nuestra columna 'TravelDate'. Vamos a plotear la serie a ver qué podemos inferir.

```
#le paso reset index porque plotly express necesita las fechas en una columna  
airpassengerspx = airpassengers.reset_index()  
px.line(airpassengerspx,x="TravelDate", y="Passengers")
```



Vemos una tendencia al alza que indica que la aerolínea tendría más pasajeros con el tiempo. Aunque hay altibajos en cada momento, generalmente podemos observar que la tendencia aumenta. También podemos notar cómo los altibajos parecen ser un poco regulares, lo que significa que también podríamos estar observando un patrón estacional. Echemos un vistazo más de cerca al observar los datos de algunos años:

```
px.line(airpassengerspx[(airpassengerspx.TravelDate.dt.year >= 1959)], x="TravelDate", y="Passengers")
```



En el plot vemos que, generalmente, hay un pico entre julio y septiembre que comienza a caer en octubre, lo que implica que más personas viajan entre julio y septiembre y probablemente viajan menos a partir de octubre.

Descomposición

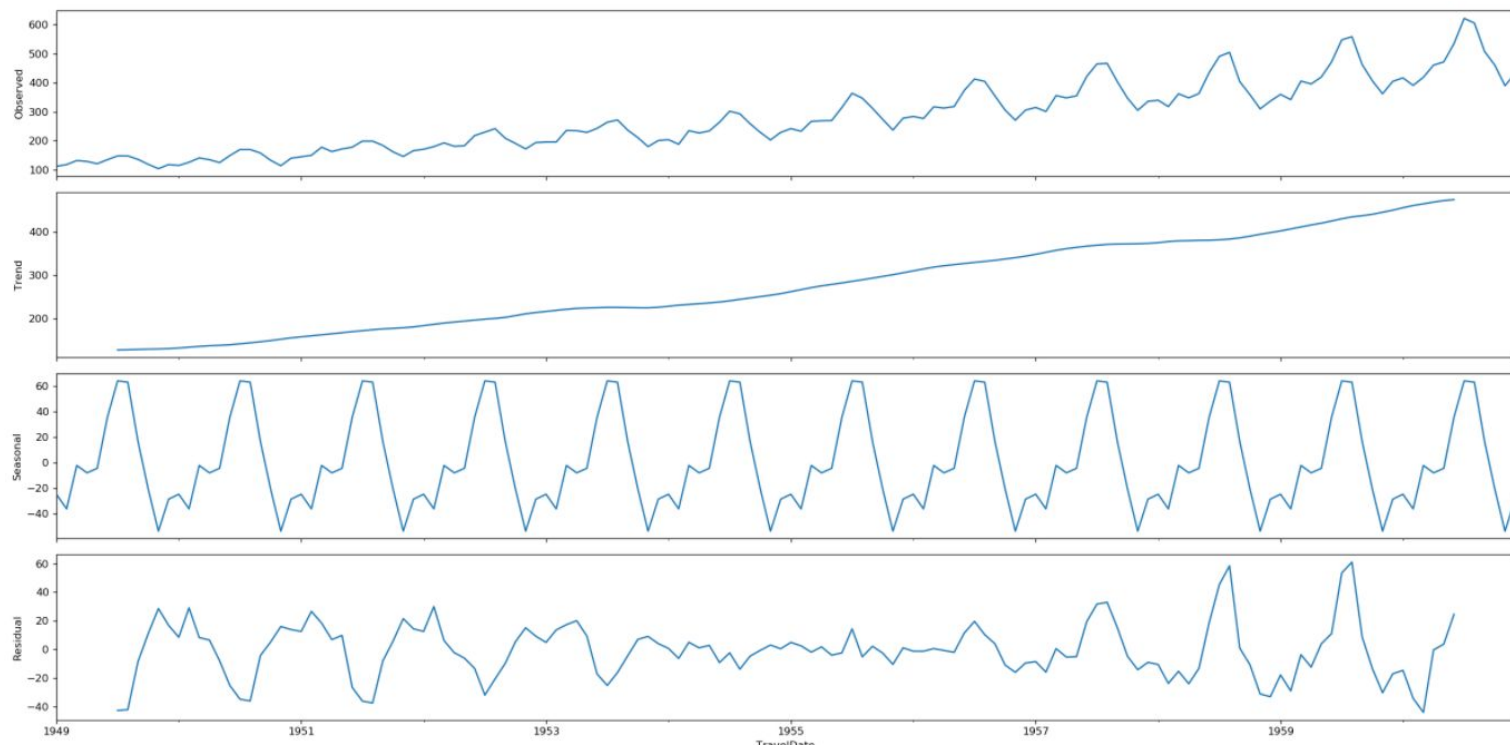
Podemos ir un poco más allá y seguir analizando el comportamiento de nuestra serie usando técnicas de descomposición para deconstruir nuestra observación en varios componentes, cada uno de los cuales representa una de las categorías subyacentes de patrones.

Para la descomposición de nuestra serie vamos a utilizar statsmodels donde se encuentra la función `seasonal_decompose` que necesitamos.



```
plt.rcParams["figure.figsize"] = (20,10)

decomposition = sm.tsa.seasonal_decompose(airpassengers)
decomposition.plot();
```



Componentes de una serie temporal

Nuestra serie es no estacionaria, tiene una tendencia a la alza y una estacionalidad que parece muy marcada, pero, ¿Qué significa cada cosa?

El estudio descriptivo de series temporales se basa en la idea de descomponer la variación de una serie en varias componentes básicas. Este enfoque descriptivo consiste en encontrar componentes que correspondan a una tendencia a largo plazo, un comportamiento estacional y una parte aleatoria.



Componentes

Las componentes o fuentes de variación que se consideran habitualmente son las siguientes:

- **Tendencia:** Se puede definir como un cambio a largo plazo que se produce en relación al nivel medio, o el cambio a largo plazo de la media. La tendencia se identifica con un movimiento suave de la serie a largo plazo.
- **Efecto Estacional:** Muchas series temporales presentan cierta periodicidad o dicho de otro modo, variación de cierto periodo (anual, mensual ...). Por ejemplo, el paro laboral aumenta en general en invierno y disminuye en verano. Estos tipos de efectos son fáciles de entender y se pueden medir explícitamente o incluso se pueden eliminar del conjunto de los datos, desestacionalizando la serie original.



Componentes

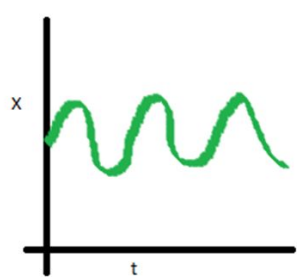
- **Componente Aleatoria o Residuos:** Una vez identificados los componentes anteriores y después de haberlos eliminado, persisten unos valores que son aleatorios. Se pretende estudiar qué tipo de comportamiento aleatorio presentan estos residuos, utilizando algún tipo de modelo probabilístico que los describa. De las tres componentes reseñadas, las dos primeras son componentes determinísticas, mientras que la última es aleatoria. Así, se puede denotar que $X_t = T_t + E_t + I_t$; donde T_t es la tendencia, E_t es la componente estacional, que constituyen la señal o parte determinística, e I_t es el ruido o parte aleatoria.



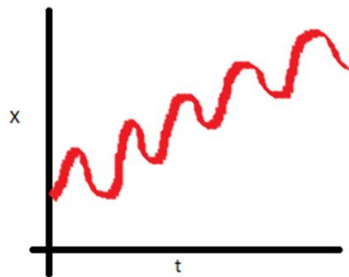
Clasificación de la serie temporal

Las series temporales se pueden clasificar en:

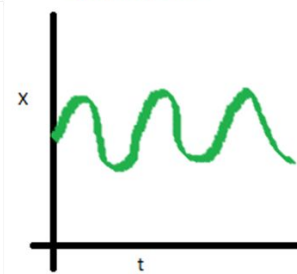
- **Estacionarias:** Una serie es estacionaria cuando es estable, es decir, cuando la media y la variabilidad son constantes a lo largo del tiempo. Esto se refleja gráficamente en que los valores de la serie tienden a oscilar alrededor de una media constante y la variabilidad con respecto a esa media también permanece constante en el tiempo. Es una serie básicamente estable a lo largo del tiempo, sin que se aprecien aumentos o disminuciones sistemáticos de sus valores. Para este tipo de series tiene sentido conceptos como la media y la varianza. Sin embargo, también es posible aplicar los mismos métodos a series no estacionarias si se transforman previamente en estacionarias.
- **No Estacionarias:** Son series en las cuales la media y/o variabilidad cambian en el tiempo. Los cambios en la media determinan una tendencia a crecer o decrecer a largo plazo, por lo que la serie no oscila alrededor de un valor constante.



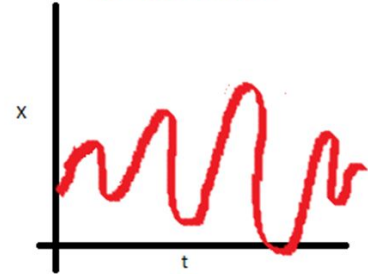
Stationary series



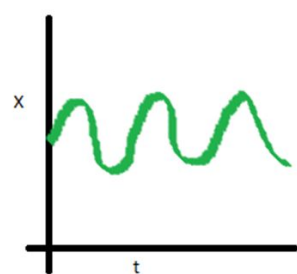
Non-Stationary series



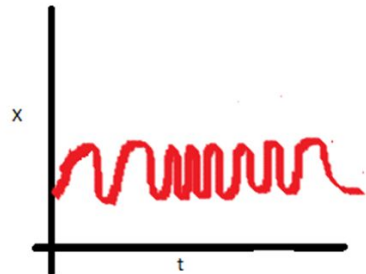
Stationary series



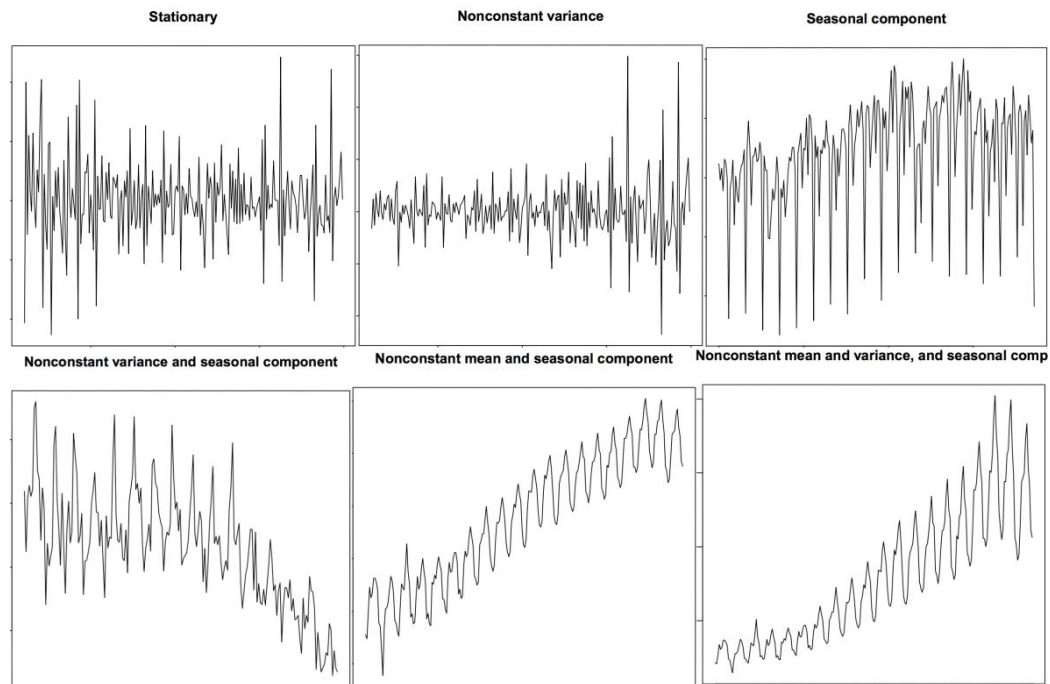
Non-Stationary series



Stationary series



Non-Stationary series



Clasificación de la serie temporal

La estacionariedad de una serie se puede establecer observando los plots de la serie como lo hicimos anteriormente.

Otro método es dividir la serie en 2 o más partes contiguas y calcular las estadísticas de resumen como la media, la varianza y la autocorrelación. Si las estadísticas son bastante diferentes, entonces es probable que la serie no sea estacionaria.

Sin embargo, para estar totalmente seguros, necesitamos un método para determinar cuantitativamente si una serie dada es estacionaria o no. Esto se puede hacer mediante pruebas estadísticas llamadas "Pruebas de raíz unitaria".




Clasificación de la serie temporal

Las dos pruebas de raíz unitaria más usadas son:

- Prueba Dickey Fuller aumentada (prueba ADF)
- Kwiatkowski-Phillips-Schmidt-Shin - Prueba KPSS (tendencia estacionaria)

La más utilizada es la prueba ADF, donde la hipótesis nula es que la serie temporal posee una raíz unitaria y no es estacionaria. Entonces, si el valor P en la prueba ADH es menor que el nivel de significancia (0.05), rechaza la hipótesis nula.

La prueba KPSS, por otro lado, se usa para probar la estacionariedad de tendencia. La hipótesis nula y la interpretación del valor P es justo lo contrario de la prueba ADH. El siguiente código implementa estas dos pruebas usando el paquete statsmodels en python.



```
from statsmodels.tsa.stattools import adfuller, kpss
# ADF Test
result = adfuller(airpassengers.Passengers, autolag='AIC')
print(f'ADF Statistic: {result[0]}')
print(f'p-value: {result[1]}')

# KPSS Test
result = kpss(airpassengers.Passengers, regression='c')
print(f'\nKPSS Statistic: %f' % result[0])
print(f'p-value: %f' % result[1])
```

ADF Statistic: 0.8153688792060423
p-value: 0.9918802434376409

KPSS Statistic: 1.052175
p-value: 0.010000

En ambos tests comprobamos que nuestra serie no es estacionaria, por lo que hemos de transformarla para aplicar modelos sobre ella.

Transformación en estacionaria

Para poder hacer pronósticos de nuestra serie temporal nos vamos a encontrar con que la mayoría de los modelos están contruidos en torno a unos supuestos que la serie ha de cumplir. El principal es que la serie sea estacionaria. Cuando una serie es estacionaria la tendencia y la estacionalidad no están afectando al componente aleatorio, que es el componente que guarda la información más crucial y más 'única' de nuestra serie. Entonces, para convertir una serie en estacionaria (una que no ha pasado los tests de raíz unitaria) simplemente tenemos que reducir o eliminar los componentes de tendencia y estacionalidad hasta que nuestra serie consiga pasar los tests de raíz unitaria.



Transformación en estacionaria

Hay muchos métodos para transformar una serie en estacionaria (transformación log, transformación de BoxCox, suavizado con medias móviles, trabajando con un ajuste de la estacionalidad...), pero aquí expongo el quizá más utilizado que nos va a servir también para la estimación de parámetros de nuestro modelo.

- Método de diferenciación: consiste en suponer que la tendencia evoluciona lentamente en el tiempo, de manera que en el instante t la tendencia debe estar próxima a la tendencia en el instante $t - 1$. De esta forma, si restamos a cada valor de la serie el valor anterior, la serie resultante estará aproximadamente libre de tendencia. Esta operación se denomina diferenciación de la serie y consiste en pasar de la serie original x_t a la serie y_t mediante: $y_t = x_t - x_{t-1}$



Podemos utilizar funciones como `ndiffs` o `nsdiffs` que nos dicen directamente el número de diferenciaciones que necesita nuestra serie para pasar cada test de raíz unitaria.

```
#pip install pmdarima  
from pmdarima.arima.utils import ndiffs
```

```
n_adf = ndiffs(airpassengers, test='adf')  
n_adf
```

0

```
n_kpss = ndiffs(airpassengers, test='kpss')  
n_kpss
```

1

```
from pmdarima.arima.utils import nsdiffs
```

```
#nsdiffs nos dice el número de diferenciaciones estacionales que necesitamos para pasar el test ch.  
n_ch = nsdiffs(airpassengers, m = 12, test='ch')  
n_ch
```

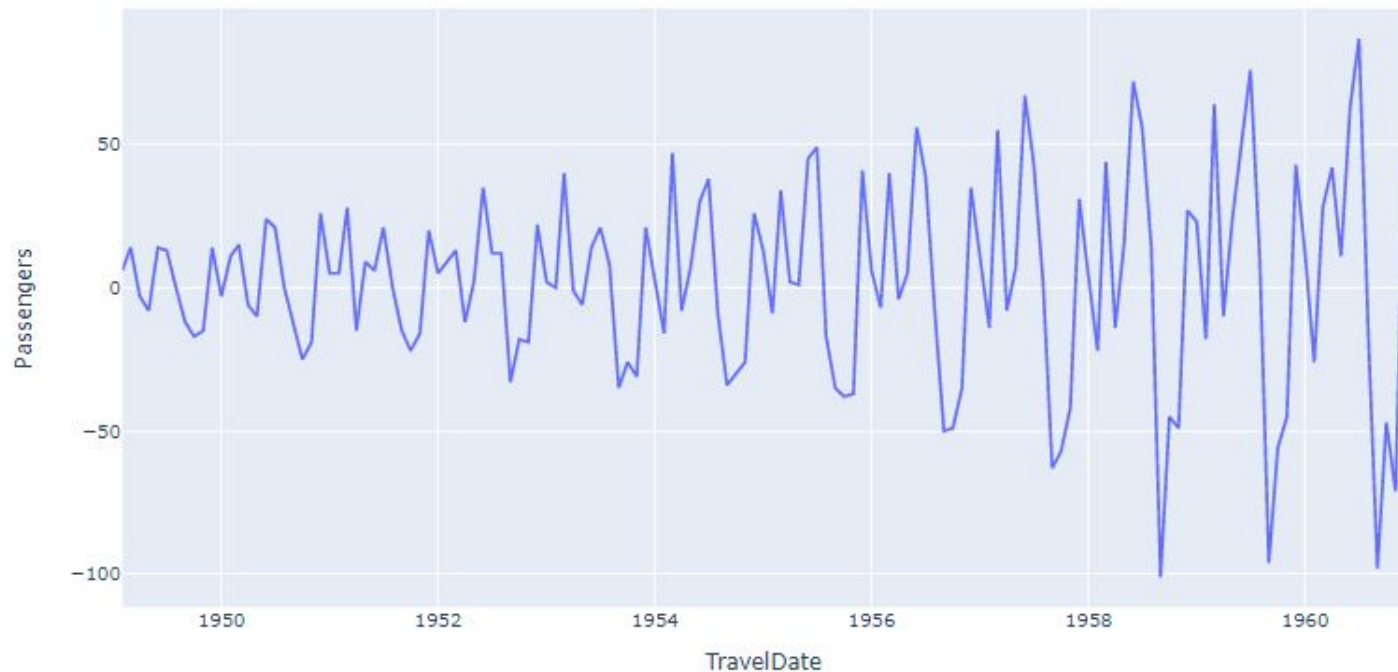
0

Haremos diferenciación de lag 1 para no pillarnos los dedos.

```
#Aplicamos el método de diferenciación con la función diff() de pandas.  
#Como solo necesitamos diff de factor 1, no ponemos nada en sus parámetros, ya que es la que viene por default  
diff1 = airpassengers.Passengers.diff()  
diff1.head()
```

```
TravelDate  
1949-01-01    NaN  
1949-02-01     6.0  
1949-03-01    14.0  
1949-04-01    -3.0  
1949-05-01    -8.0  
Name: Passengers, dtype: float64
```

```
#creo el df px para pyplot express  
diff1px = diff1.reset_index()  
px.line(diff1px.dropna(),x="TravelDate", y="Passengers")
```



Aquí vemos cómo cambia la serie, suavizando la tendencia y la varianza.

Vamos a volver a pasar los tests de raíz unitaria a esta nueva serie para ver si la transformación ha surtido efecto y es estacionaria. Retiramos los NaN para que no nos dé error.

```
# ADF Test
result = adfuller(diff1.dropna(), autolag='AIC')
print(f'ADF Statistic: {result[0]}')
print(f'p-value: {result[1]}')
```

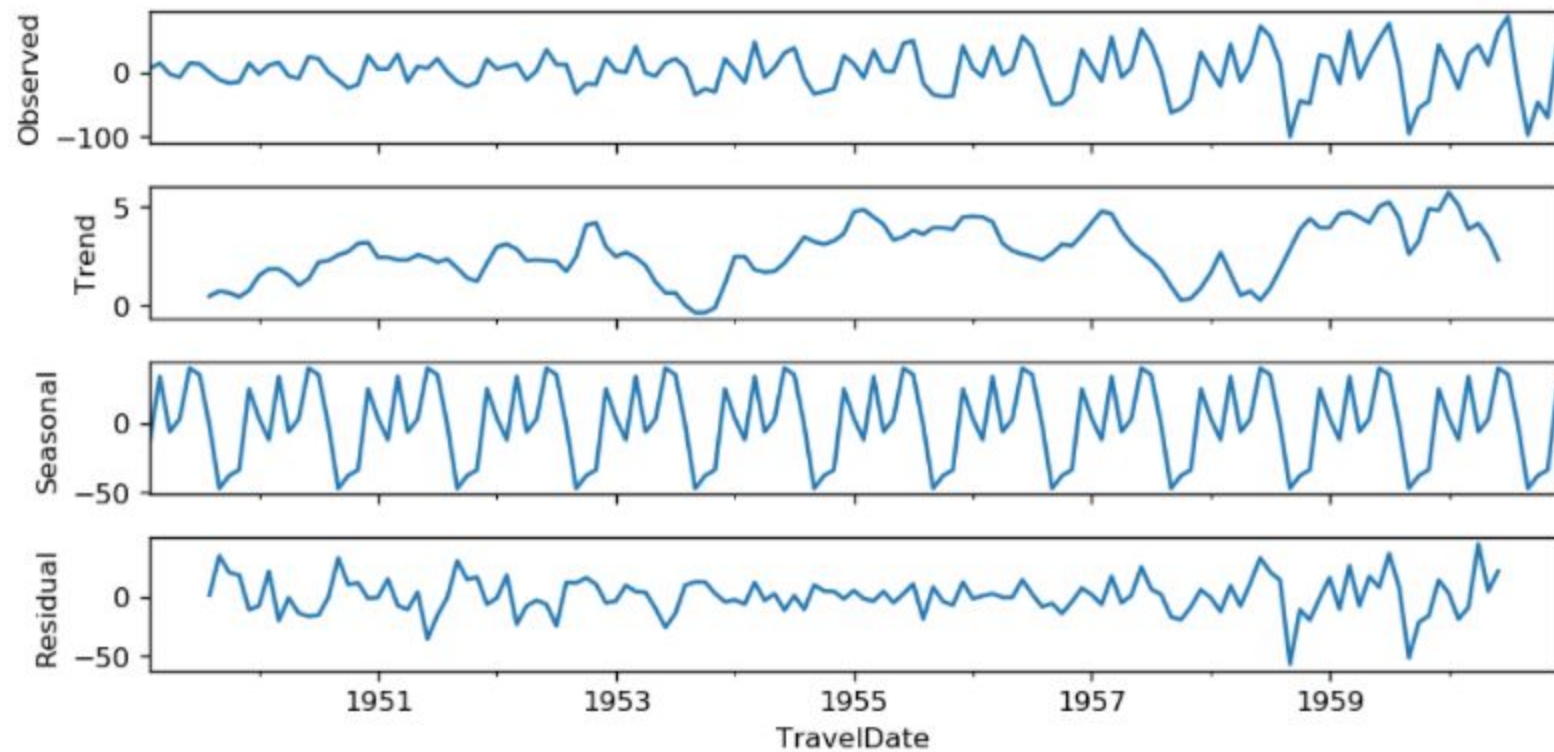
```
# KPSS Test
result = kpss(diff1.dropna(), regression='c')
print('\nKPSS Statistic: %f' % result[0])
print('p-value: %f' % result[1])
```

```
ADF Statistic: -2.8292668241699874
p-value: 0.05421329028382711
```

```
KPSS Statistic: 0.053010
p-value: 0.100000
```

Ha cambiado un poco la cosa. En el test de ADF vemos que podemos rechazar la H_0 (no estacionaria) con cierta confianza, no muchísima pero bueno. En el test KPSS pasa más o menos lo mismo, no podemos rechazar la H_0 (estacionaria) si no queremos arriesgarnos a un error de al menos el 10%

```
sm.tsa.seasonal_decompose(diff1.dropna()).plot();
```



Forecasting con ARIMA

Hay muchos modelos que pueden aplicarse para trabajar con series temporales, siempre depende de las características de nuestra serie temporal y los objetivos que tengamos. Sin embargo, posiblemente el más pulido y utilizado para series temporales sea el modelo ARIMA, que consigue batir casi siempre a modelos mucho más avanzados como las redes neuronales recurrentes.



Forecasting con ARIMA

Entonces, ¿qué es exactamente un modelo ARIMA?

ARIMA, abreviatura de 'Autoregressive integrated moving average' es en realidad una clase de modelos que 'explica' una serie temporal dada en función de sus propios valores pasados, es decir, en función de sus propios 'lags' (AR) y de los errores de pronóstico de sus valores reales (pasados) respecto de los pronosticados (MA), de modo que se puede usar la ecuación para pronosticar valores futuros.

Cualquier serie de tiempo "no estacional" que muestre patrones y no sea un ruido blanco aleatorio (que no tenga autocorrelación) puede modelarse con los modelos ARIMA

Forecasting con ARIMA

¿Cómo se construye? 3 parámetros: p , d , q

El primer paso para construir un modelo ARIMA es hacer estacionarias las series temporales porque el término "Autoregressive" en ARIMA significa que es un modelo de regresión lineal que usa sus propios lags como predictores. Los modelos de regresión lineal funcionan mejor cuando los predictores no están correlacionados y son independientes entre sí.



Forecasting con ARIMA

Hemos visto que para hacer una serie estacionaria el metodo más común es el de diferenciación, que va a dar el valor al parámetro 'd'. Por lo que:

- d: es la cantidad de diferenciación requerida para hacer estacionarias las series de tiempo

Mientras que,

- p: es el orden del término "Auto regresivo" (AR). Se refiere al número de lags de Y que se utilizarán como predictores.
- q: es el orden del término "Promedio móvil" (MA). Se refiere a la cantidad de errores de pronóstico pasados que deberían incluirse en el modelo ARIMA.

El truco del almendruco. La función `auto_arima`

A partir de esta práctica para el cálculo de parámetros algún listo/a crea la función de `auto_arima` que hace todo esto por nosotros y nos devuelve el mejor modelo.



```
import pmdarima as pm

model = pm.auto_arima(airpassengers, start_p=0, start_q=0,
                      max_p=3, max_q=3, m=12,
                      start_P=0, seasonal=True,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

model.summary()
```

El mejor modelo sería un SARIMA (1,1,0)(2,1,1), podemos ir tocando los parámetros de la función para ver si conseguimos uno mejor

```
model.predict(n_periods = 24)
```

```
array([448.86832753, 422.51966258, 463.6420321 , 492.67984489,
       509.75144142, 573.32235415, 663.26429646, 654.75042348,
       549.65503907, 497.19361936, 429.79627663, 474.97360671,
       492.20732303, 462.99499775, 502.09229757, 538.25429744,
       555.50257145, 622.86756051, 718.24171991, 706.54275657,
       595.94305909, 542.50355822, 469.55460095, 516.61556359])
```

Ploteamos los pronósticos sobre la serie original.

Dep. Variable: y No. Observations: 144

Model: SARIMAX(1, 1, 0)x(2, 1, 1, 12) Log Likelihood -502.480

Date: Mon, 12 Aug 2019 AIC 1016.960

Time: 13:20:53 BIC 1034.211

Sample: 0 HQIC 1023.970

- 144

Covariance Type: opg

	coef	std err	z	P> z	[0.025	0.975]
--	------	---------	---	------	--------	--------

intercept	0.0045	0.178	0.025	0.980	-0.345	0.354
-----------	--------	-------	-------	-------	--------	-------

ar.L1	-0.3766	0.077	-4.889	0.000	-0.528	-0.226
-------	---------	-------	--------	-------	--------	--------

ar.S.L12	0.6891	0.140	4.916	0.000	0.414	0.964
----------	--------	-------	-------	-------	-------	-------

ar.S.L24	0.3091	0.107	2.884	0.004	0.099	0.519
----------	--------	-------	-------	-------	-------	-------

ma.S.L12	-0.9742	0.512	-1.904	0.057	-1.977	0.029
----------	---------	-------	--------	-------	--------	-------

sigma2	113.2075	48.900	2.315	0.021	17.364	209.051
--------	----------	--------	-------	-------	--------	---------

Ljung-Box (Q): 58.67 Jarque-Bera (JB): 12.12

Prob(Q):	0.03	Prob(JB):	0.00
----------	------	-----------	------

Heteroskedasticity (H):	2.70	Skew:	0.10
-------------------------	------	-------	------

Prob(H) (two-sided):	0.00	Kurtosis:	4.48
----------------------	------	-----------	------

Gracias!

alopezr@faculty.ie.edu

<https://www.linkedin.com/in/antoniolopezrosell/>

616299397