

# Índice

<b>1. Procesos estocásticos y datos de series temporales</b>	<b>2</b>
1.1. Datos de sección cruzada vs datos de series temporales . . . . .	3
1.2. El desafío . . . . .	3
<b>2. Estacionariedad</b>	<b>5</b>
2.1. Estacionariedad en sentido débil . . . . .	5
2.2. Función de autocovarianzas y función de autocorrelación . . . . .	6
<b>3. Transformaciones de realizaciones de procesos estocásticos NO estacionarios</b>	<b>7</b>
3.1. Internat. airline passengers: monthly totals in thousands. Jan 49 – Dec 60 . . . . .	7
3.1.1. Trasformación logarítmica de los datos . . . . .	8
3.1.2. Primera diferencia del logaritmo de los datos . . . . .	10
3.1.3. Diferencia estacional de la primera diferencia del logaritmo de los datos . . . . .	10
3.2. Tasa logarítmica de crecimiento . . . . .	11
3.2.1. Comentarios y/o interpretaciones de los datos transformados . . . . .	12

# Lección 1. Transformación de datos

Marcos Bujosa

12 de noviembre de 2025

## Resumen

En esta lección veremos algunas transformaciones de los datos para *"hacerlos estacionarios"*; y daremos interpretación a los datos transformados.

- ([slides](#)) — ([html](#)) — ([pdf](#)) — ([mybinder](#))

## Carga de algunos módulos de python y creación de directorios auxiliares

---

```
# Para trabajar con los datos y dibujarlos necesitamos cargar algunos módulos de python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib as mpl
# definimos parámetros para mejorar los gráficos
mpl.rcParams['text', usetex=False]
import matplotlib.pyplot as plt # data visualization
```

---

- Creación del directorio auxiliar para albergar las figuras de la lección. Para publicar la lección como pdf o página web, necesito los gráficos como ficheros .png alojados algún directorio específico:

---

```
imagenes_leccion = "./img/lecc01" # directorio para las imágenes de la lección
import os
os.makedirs(imagenes_leccion, exist_ok=True) # crea el directorio si no existe
```

---

## 1. Procesos estocásticos y datos de series temporales

**Proceso estocástico** es una secuencia de variables aleatorias,  $X_t$  donde el índice  $t$  recorre el conjunto de números enteros ( $\mathbb{Z}$ ).

$$\mathbf{X} = (\dots, X_{-2}, X_{-1}, X_0, X_1, \dots) = (X_t \mid t \in \mathbb{Z});$$

**Muestra** es una secuencia *finita* de datos (valores).

$$\mathbf{x} = (x_1, x_2, \dots, x_n)$$

- Consideraremos cada dato  $x_t$  como una *realización* de  $X_t$ .

- Consecuentemente, consideraremos que una *muestra* es una *realización de un tramo finito* de un proceso estocástico:

$(x_1, x_2, \dots, x_n)$  es una realización de  $(X_t \mid t = 1 : n)$ .

Nótese que en el **proceso estocástico** el índice  $t$  recorre los infinitos números enteros mientras que en la **muestra** solo recorre los naturales entre 1 y  $n$ .

### 1.1. Datos de sección cruzada vs datos de series temporales

Consideremos dos tipos de muestras  $\mathbf{x} = (x_1, x_2, \dots, x_n)$ :

**Sección cruzada** el índice NO es cronológico. La numeración (la indexación) de cada dato es solo una *asignación arbitraria de etiquetas* que identifican a cada individuo, empresa, objeto, etc. que ha sido medido. Consecuentemente:

- el orden en el que aparecen los datos de la muestra es irrelevante.
- conocer el índice de un dato no permite deducir nada respecto de cualquier otro dato de la muestra.

**Serie temporal** Corresponden a mediciones de un mismo objeto a lo largo del tiempo. El índice indica el instante de cada medición. *Es habitual que el orden cronológico de los datos sea importante* para explicar cada uno de ellos.

- con frecuencia la medición en un instante de tiempo está relacionada con otras mediciones próximas en el tiempo. En tal caso...
- no deberemos asumir que las variables aleatorias del proceso estocástico subyacente,  $\mathbf{X} = (X_t \mid t \in \mathbb{Z})$ , sean independientes.

### 1.2. El desafío

El análisis de *series temporales* trata sobre la inferencia estadística de muestras que **frecuentemente NO podemos asumir que sean realizaciones** de variables aleatorias *i.i.d. (independientes e idénticamente distribuidas)*.

Así pues, aunque

- el marco ideal para el análisis es que la serie temporal **"sea estacionaria"** (!!),  
*(!! abuso del lenguaje que expresa que podemos asumir que la serie es una realización de un proceso estocástico estacionario, es decir, cuyos momentos no dependen del índice  $t$ . Veremos una definición formal en lecciones posteriores).*
- lo habitual es que, por distintos motivos, **NO lo sea**.

---

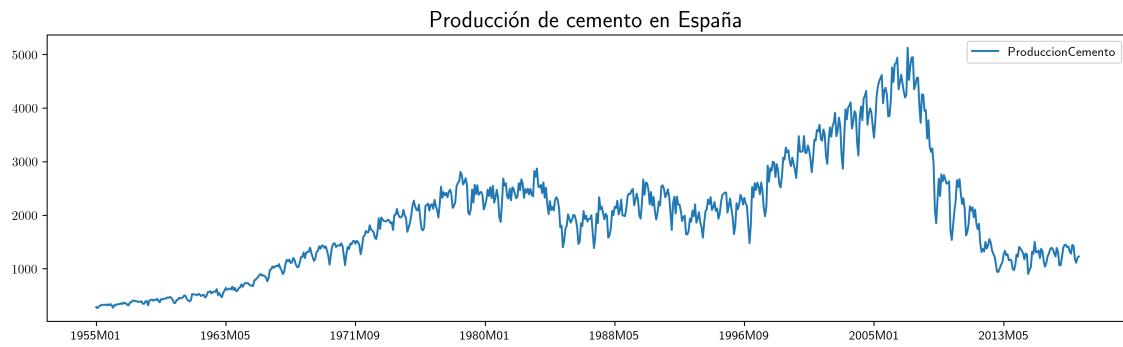
```
path = './datos/'
df1 = pd.read_csv(path+'PIB_UEM.csv')
df2 = pd.read_csv(path+'ProduccionCemento.csv')
df3 = pd.read_csv(path+'IBEX35.csv')
df4 = pd.read_csv(path+'ExportacionDeAcero.csv')
```

---

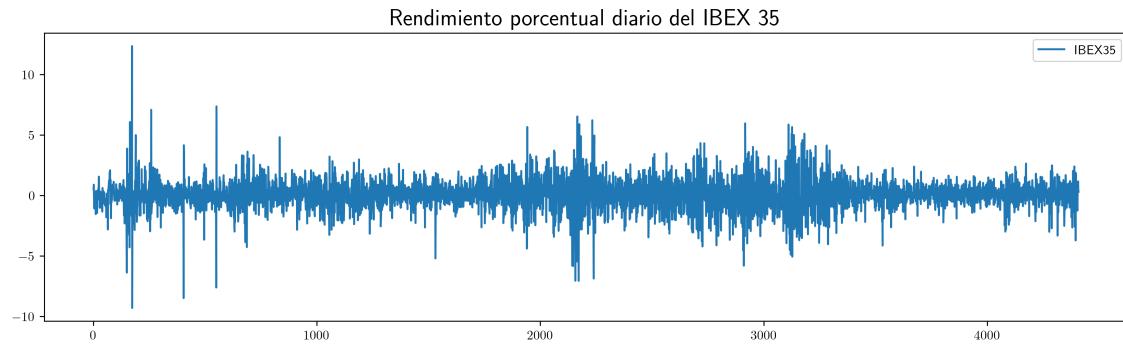
```
df1.plot(x='obs', xlabel='', figsize=(15,4)).set_title('PIB zona euro', fontsize=18)
plt.savefig('./img/lecc01/PIB_UEM.png', dpi=300, bbox_inches='tight')
```



```
df2.plot(x='obs', xlabel='', figsize=(15,4)).set_title('Producción de cemento en España', fontsize=18)
plt.savefig('./img/lecc01/ProduccionCemento.png', dpi=300, bbox_inches='tight')
```



```
df3.plot(x='obs', xlabel='', figsize=(15,4)).set_title('Rendimiento porcentual diario del IBEX 35', fontsize=18)
plt.savefig('./img/lecc01/IBEX35.png', dpi=300, bbox_inches='tight')
```



```
df4.plot(x='obs', xlabel='', figsize=(15,4)).set_title('Exportaciones españolas de acero', fontsize=18)
plt.savefig('./img/lecc01/ExportacionDeAcero.png', dpi=300, bbox_inches='tight')
```



El desafío para el analista es

**primero** transformar los datos para lograr que sean "*estacionarios*"

**y después** transformar los datos estacionarios en "*ruido blanco*" (!!)

(!! *nuevo abuso del lenguaje que expresa que podemos asumir dichos datos transformados son realizaciones de un proceso de ruido blanco, i.e. de media cero e incorrelado.*)

## 2. Estacionariedad

El primer objetivo del *análisis de series temporales* es inferir la distribución de  $\mathbf{X} = (X_t \mid t \in \mathbb{Z})$  usando una muestra finita (serie temporal)  $\mathbf{x} = (x_t \mid t = 1 : n)$ .

Así podremos intentar

**Predecir** datos futuros

**Controlar** datos futuros

Pero esto es inabordable si la evolución de los datos es inestable en el tiempo.

Por tanto, algún tipo de estabilidad (o estacionariedad) es necesaria.

### 2.1. Estacionariedad en sentido débil

Un proceso estocástico  $\mathbf{X}$  se dice **estacionario** (*en sentido débil*) si para todo  $t, k \in \mathbb{Z}$

$$E(X_t) = \mu \tag{1}$$

$$\text{Cov}(X_t, X_{t-k}) = \gamma_k \tag{2}$$

- (1) sugiere que las realizaciones de  $\mathbf{X}$  aparecerán entorno al valor  $\mu$ .
- (2) entre otras cosas, sugiere que la variabilidad de las realizaciones de  $\mathbf{X}$  entorno a  $\mu$  es constante, ya que para el caso particular  $k = 0$

$$\text{Cov}(X_t, X_{t-0}) = \text{Var}(X_t) = \gamma_0 \quad \text{para todo } t,$$

donde  $\gamma_0$  es la varianza común a todas las variables aleatorias del proceso.

Es más, la desigualdad de Chebyshev

$$P(|X_t - \mu| \geq c\sigma) \leq \frac{1}{c^2}, \quad \text{donde } \sigma = \sqrt{\gamma_0}$$

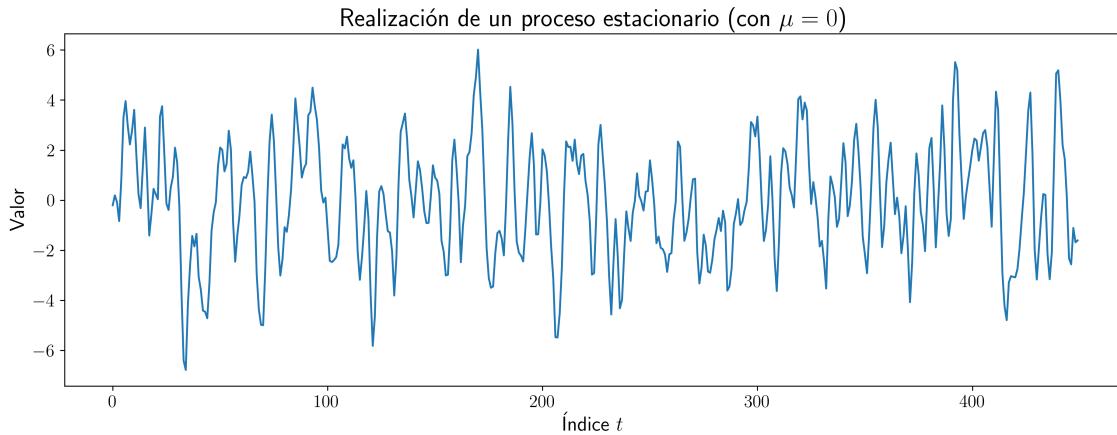
sugiere que para cualquier proceso estacionario (y un  $c$  grande), al pintar una realización, tan solo un pequeño porcentaje de los datos caerán fuera de la franja  $(\mu - c\sigma, \mu + c\sigma)$ .

---

```
# simulamos un proceso ARMA(p,q)
import statsmodels.api as sm
np.random.seed(12345)
arparams = np.array([.75, -.25])
maparams = np.array([.65, .35])
ar = np.r_[1, -arparams] # add zero-lag and negate
ma = np.r_[1, maparams] # add zero-lag
y = sm.tsa.arma_generate_sample(ar, ma, 450)

# creamos el gráfico de la serie simulada
plt.figure(figsize=(15,5))
plt.title("Realización de un proceso estacionario (con $\mu=0$)", fontsize=20)
plt.xlabel("Índice $t$", fontsize=16)
plt.ylabel("Valor", fontsize=16)
plt.tick_params(axis='both', labelsize=14)
plt.plot(y)
plt.savefig("./img/lecc01/stationaryTimeSeriesExample.png", dpi=300, bbox_inches='tight')
```

---



## 2.2. Función de autocovarianzas y función de autocorrelación

Cuando  $\mathbf{X}$  es un proceso estocástico (débilmente) **estacionario**:

- La secuencia  $(\gamma_k \mid k \in \mathbb{Z})$ , donde  $\gamma_k = Cov(X_t, X_{t-k})$  se denomina *función de autocovarianzas*.

Debido a la estacionariedad, la correlación entre  $X_t$  y  $X_{t+k}$  no depende de  $t$ ; tan solo depende de la distancia  $k$  entre los índices de ambas variables.

- La secuencia  $(\rho_k \mid k \in \mathbb{Z})$ , donde  $\rho_k = \frac{Cov(X_t, X_{t-k})}{\sqrt{Var(X_t)Var(X_{t-k})}} = \frac{\gamma_k}{\gamma_0}$  se denomina *función de autocorrelación (ACF)*.

(Estas secuencias serán fundamentales en el análisis de ciertos procesos estocásticos en futuras lecciones).

### 3. Transformaciones de realizaciones de procesos estocásticos NO estacionarios

Un proceso estocástico  $\mathbf{X} = (X_t \mid t \in \mathbb{Z})$  puede ser:

**NO estacionario en media** porque  $E(X_t)$  depende de  $t$ .

**NO estacionario en covarianza** porque  $Cov(X_t, X_{t-k})$  depende de  $t$ .

Separar o distinguir ambos tipos de no estacionariedad no es sencillo.

Veamos un ejemplo de serie temporal para la que

- no podemos asumir que sea realización de un proceso estocástico *estacionario*;
- y algunos intentos de transformación para obtener datos "**estacionarios**"(!!).  
(!!) recuerde que esta expresión, aunque extendida, es un abuso del lenguaje).

#### 3.1. Internat. airline passengers: monthly totals in thousands. Jan 49 – Dec 60

---

```
# Leemos los datos de un fichero csv y generamos un dataframe de pandas.
OrigData = pd.read_csv('../datos/airline-passengers.csv')
#OrigData = pd.read_csv('../database/Datasets-master/airline-passengers.csv')
OrigData['Month']=pd.to_datetime(OrigData['Month'])
OrigData = OrigData.set_index(['Month'])
# print(OrigData.head())
```

---



---

```
plt.figure(figsize=(15,5))
plt.subplot(1, 2, 1)
plt.plot(OrigData['Passengers'])
plt.xlabel("Month", fontsize=16)
plt.ylabel(r"Number of Air Passengers, ($\boldsymbol{x}$)", fontsize=16)
plt.tick_params(axis='both', labelsize=14)
plt.subplot(1, 2, 2)
plt.hist(OrigData['Passengers'], edgecolor='white', bins=11)
plt.xlabel("Number of Air Passengers in thousands", fontsize=16)
plt.tick_params(axis='both', labelsize=14)
plt.tight_layout()
plt.savefig('../img/lecc01/airlinepass+hist.png', dpi=300, bbox_inches='tight')
```

---



$$\mathbf{x} = (x_1, \dots x_{114})$$

Serie "no estacionaria" (!!):

- El nivel de la serie crece de año en año.
- La variabilidad estacional crece con el nivel (creciente diferencia entre el verano y el otoño).

### 3.1.1. Trasformación logarítmica de los datos

- Al aplicar la función logarítmica transformamos **monótonamente** los datos estabilizando la varianza cuando los valores son mayores que 0.567 (aprox.).
- Pero ocurre lo contrario cuando los valores son pequeños (aumenta el valor absoluto de aquellos entre 0 y 0.567 aprox.). De hecho,  $\lim_{x \rightarrow 0} \ln(x) = -\infty$ .

Recuerde que *el logaritmo no está definido para valores negativos*.

---

```
# Definir el rango de valores para x (empezando desde un número positivo ya que log(0) no está definido)
x = np.linspace(0.01, 7, 400) # Valores de 0.1 a 10
# Calcular y = log(x)
y = np.log(x)

# Crear el gráfico
plt.figure(figsize=(16, 5))
plt.plot(x, y, label='y = ln(x)')
# Añadir etiquetas y título
plt.xlabel('x', fontsize=16)
plt.ylabel('ln(x)', fontsize=16)
plt.tick_params(axis='both', labelsize=14)
plt.title('Gráfico de la función logarítmica y = ln(x)', fontsize=20)
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.grid(color = 'gray', linestyle = '--', linewidth = 0.5)
plt.legend(fontsize=20)
plt.savefig("./img/lecc01/funcion_logaritmica.png", dpi=300, bbox_inches='tight')
```

---




---

```
# Creamos un nuevo dataframe con los datos originales y varias transformaciones de los mismos
TransformedData = OrigData.copy()
TransformedData['dataLog'] = np.log(OrigData['Passengers'])
TransformedData['dataLogDiff'] = TransformedData['dataLog'].diff(1)
TransformedData['dataLogDiffDiff12'] = TransformedData['dataLogDiff'].diff(12)
```

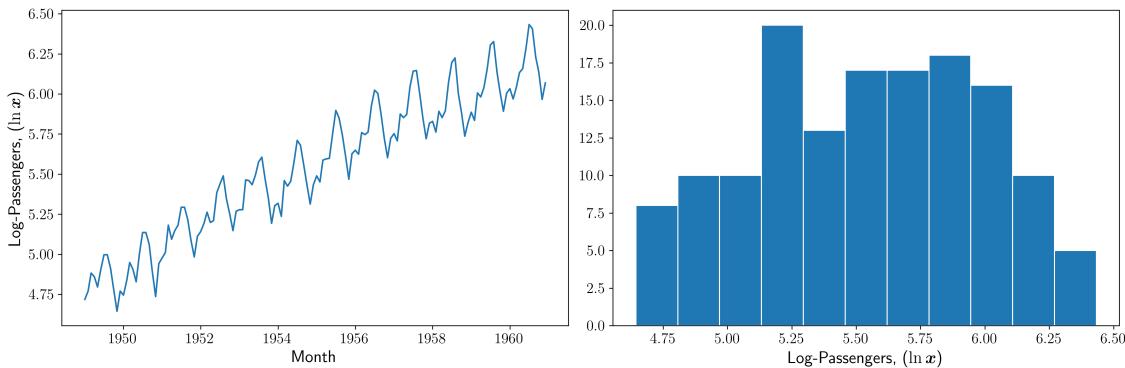
---

## Transformación logarítmica de los datos

---

```
plt.figure(figsize=(15,5))
plt.subplot(1, 2, 1)
plt.plot(TransformedData['dataLog'])
plt.xlabel("Month", fontsize=16)
plt.ylabel(r"Log-Passenger, ($\ln x$)", fontsize=16)
plt.tick_params(axis='both', labelsize=14)
plt.subplot(1, 2, 2)
plt.hist(TransformedData['dataLog'], edgecolor='white', bins=11)
plt.xlabel(r"Log-Passenger, ($\ln x$)", fontsize=16)
plt.tick_params(axis='both', labelsize=14)
plt.tight_layout()
plt.savefig('./img/lecc01/airlinepass_log+hist.png', dpi=300, bbox_inches='tight')
```

---



$$\ln \mathbf{x} = (\ln(x_1), \dots, \ln(x_{114}))$$

Ésta tampoco parece la realización de un proceso estocástico *estacionario*:

- Aunque la variabilidad estacional parece mantenerse de año en año,

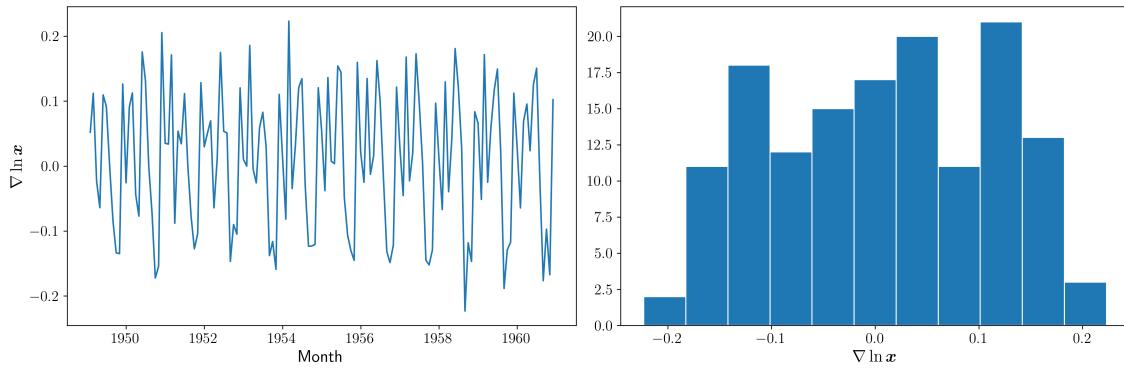
- el nivel sigue creciendo de año en año.

### 3.1.2. Primera diferencia del logaritmo de los datos

---

```
plt.figure(figsize=(15,5))
plt.subplot(1, 2, 1)
plt.plot(TransformedData['dataLogDiff'])
plt.xlabel("Month", fontsize=16)
plt.ylabel(r"\nabla \ln \boldsymbol{x}", fontsize=16)
plt.tick_params(axis='both', labelsize=14)
plt.subplot(1, 2, 2)
plt.hist(TransformedData['dataLogDiff'], edgecolor='white', bins=11)
plt.xlabel(r"\nabla \ln \boldsymbol{x}", fontsize=16)
plt.tick_params(axis='both', labelsize=14)
plt.tight_layout()
plt.savefig('./img/lecc01/airlinepass_logDiff+hist.png', dpi=300, bbox_inches='tight')
```

---



$$\mathbf{y} = \nabla \ln \mathbf{x} = \left( [\ln(x_2) - \ln(x_1)], \dots, [\ln(x_{114}) - \ln(x_{113})] \right)$$

Esta serie tampoco parece "estacionaria" (!!):

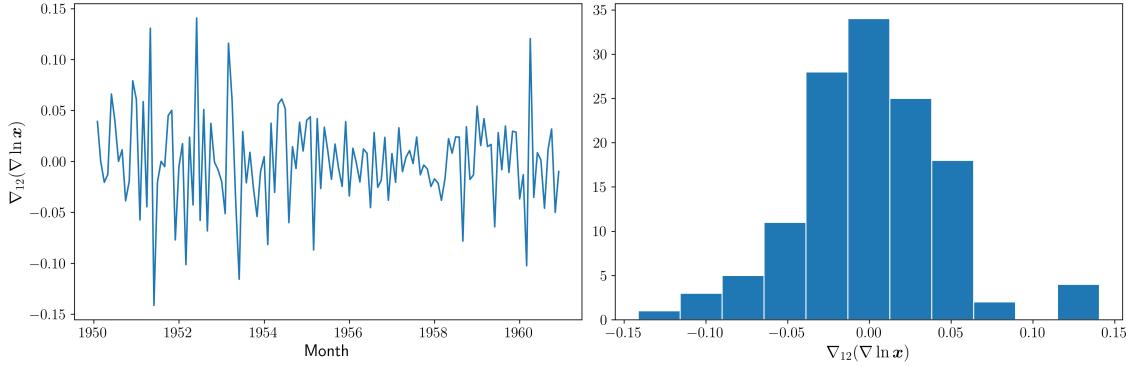
- Hay un *persistente* componente periódico (de naturaleza estacional) debido a que hay pocos viajes en otoño y muchos en Navidad, Semana Santa y verano (i.e., el número esperado de viajeros parece cambiar en función del mes o estación).

### 3.1.3. Diferencia estacional de la primera diferencia del logaritmo de los datos

---

```
plt.figure(figsize=(15,5))
plt.subplot(1, 2, 1)
plt.plot(TransformedData['dataLogDiffDiff12'])
plt.xlabel("Month", fontsize=16)
plt.ylabel(r"\nabla_{12} (\nabla \ln \boldsymbol{x})", fontsize=16)
plt.tick_params(axis='both', labelsize=14)
plt.subplot(1, 2, 2)
plt.hist(TransformedData['dataLogDiffDiff12'], edgecolor='white', bins=11)
plt.xlabel(r"\nabla_{12} (\nabla \ln \boldsymbol{x})", fontsize=16)
plt.tick_params(axis='both', labelsize=14)
plt.tight_layout()
plt.savefig('./img/lecc01/airlinepass_logDiffDiff12+hist.png', dpi=300, bbox_inches='tight')
```

---



$$\mathbf{z} = \nabla_{12}(\nabla \ln \mathbf{x}) = \nabla_{12}(\mathbf{y}) = ((y_{13} - y_1), \dots, (y_{113} - y_{101}))$$

- Esta serie tiene el aspecto de realización de un proceso *estacionario*.
- De propina, el histograma sugiere una distribución aproximadamente Gaussiana.

### 3.2. Tasa logarítmica de crecimiento

---

```

START = 100
UnoPorCiento = lambda n0, t: n0 if t<=1 else 1.01 * UnoPorCiento(n0, t-1)
TasaLogCrecimiento = pd.DataFrame({'$y_t$': [UnoPorCiento(START,t+1) for t in range(10)]})
TasaLogCrecimiento['r'$\frac{y_t-y_{t-1}}{y_{t-1}}$'] = TasaLogCrecimiento['$y_t$'].pct_change()
TasaLogCrecimiento['r'$\ln y_t$'] = np.log(TasaLogCrecimiento['$y_t$'])
TasaLogCrecimiento['r'$\backslash(\ln y_t-\ln y_{t-1})$'] = TasaLogCrecimiento['r'$\ln y_t$']-TasaLogCrecimiento['r'$\ln y_{t-1}$'].shift(+1)
TasaLogCrecimiento['r'$\frac{y_t-y_0}{y_0}$'] = TasaLogCrecimiento['$y_t$'].apply(lambda x: ((x/START)-1))
TasaLogCrecimiento['r'$\backslash(\ln y_t-\ln y_0)$'] = TasaLogCrecimiento['r'$\ln y_t$']-TasaLogCrecimiento['r'$\ln y_0$'].iloc[0]

```

---

La tasa logarítmica de variación de  $\mathbf{y}$  se define como  $z_t = \ln y_t - \ln y_{t-1}$ ; es decir

$$\mathbf{z} = \nabla \ln \mathbf{y} = ([\ln(y_2) - \ln(y_1)], \dots, [\ln(y_n) - \ln(y_{n-1})])$$

y se *aproxima* a la tasa de crecimiento (en tanto por uno) si el incremento es pequeño.

	$y_t$	$\frac{y_t-y_{t-1}}{y_{t-1}}$	$\ln y_t$	$(\ln y_t - \ln y_{t-1})$	$\frac{y_t-y_0}{y_0}$	$(\ln y_t - \ln y_0)$
0	100.000000	NaN	4.605170	NaN	0.000000	0.000000
1	101.000000	0.01	4.615121	0.00995	0.010000	0.009950
2	102.010000	0.01	4.625071	0.00995	0.020100	0.019901
3	103.030100	0.01	4.635021	0.00995	0.030301	0.029851
4	104.060401	0.01	4.644972	0.00995	0.040604	0.039801
5	105.101005	0.01	4.654922	0.00995	0.051010	0.049752
6	106.152015	0.01	4.664872	0.00995	0.061520	0.059702
7	107.213535	0.01	4.674823	0.00995	0.072135	0.069652
8	108.285671	0.01	4.684773	0.00995	0.082857	0.079603
9	109.368527	0.01	4.694723	0.00995	0.093685	0.089553

### 3.2.1. Comentarios y/o interpretaciones de los datos transformados

Transformación de la serie temporal $\mathbf{y} = \{y_t\}, t = 1 : n$	Comentario y/o interpretación
$\mathbf{z} = \ln \mathbf{y} = \{\ln y_t\}$	A veces independiza la volatilidad del nivel. A veces induce normalidad.
$\mathbf{z} = \nabla \mathbf{y} = \{y_t - y_{t-1}\}$	Indica al crecimiento absoluto entre periodos consecutivos.
$\begin{aligned} \mathbf{z} &= \nabla \ln \mathbf{y} \\ &= \{\ln y_t - \ln y_{t-1}\} \end{aligned}$	Tasa logarítmica de crecimiento. Aproximación del crecimiento relativo entre periodos consecutivos.
$\mathbf{z} = \nabla \nabla \ln \mathbf{y} = \nabla^2 \ln \mathbf{y}$	Cambio en la tasa log. de crecimiento. Indica la “aceleración” en el crecimiento relativo.
$\begin{aligned} \mathbf{z} &= \nabla_s \ln \mathbf{y} \\ &= \{\ln y_t - \ln y_{t-s}\} \end{aligned}$	Tasa log. de crecimiento acumulada en un ciclo estacional completo ( $s$ períodos). Cuando el período estacional es de un año, se conoce como “tasa anual” o “tasa interanual” de crecimiento.
$\mathbf{z} = \nabla \nabla_s \ln \mathbf{y}$	Cambio en la tasa log. de crecimiento acumulada en un ciclo estacional completo. Es un indicador de aceleración en el crecimiento acumulado.