Моделирование временных рядов

Моделирование временных рядов. Детерминистические модели. Основные типы трендов. Модели сезонности. Регулярные и нерегулярные события. Стохастические модели временных рядов. Понятие белый гауссов шум. Нестационарные шумы. Модель временного ряда со случайным блужданием.

Импорт данных

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdates
# Use seaborn style defaults and set the default figure size
sns.set(rc={'figure.figsize':(16, 4)})
```

Детерминированные модели

Модель временного ряда

Простейшим случаем детерминированного временного ряда является одномерная (одномерная) зависимость значения от времени, представленная в следующей форме

$$y(t) = a_0 + trend(t) + cyclic(t) + seasonal(t)$$

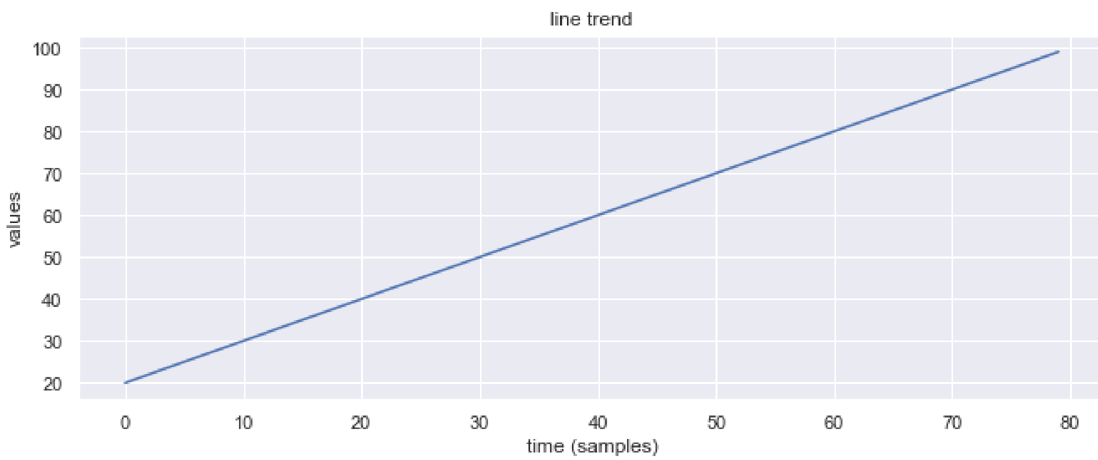
где

- y(t) - это временной ряд - набор выборок, проиндексированных некоторой переменной t, обычно t – это временные отметки, если временной шаг дискретный, он также может быть обозначен как n (номер выборки), в этом случае в реальном времени - значение шага будет соответствовать $t = n \cdot T_s$, где T_s - период шага n (период дискретизации, с которым берутся отсчеты).
- a_0 - некоторый начальный постоянный уровень,
- $trend$ - это наличие некоторого тренда, который является частью зависимости с медленным изменением.
- $seasonal$ - это сезонность или некоторые «относительно быстро изменяющиеся» периодические составляющие - это относительно быстро меняющаяся часть взаимосвязи.
- $cyclic$ - это некоторые периодические компоненты с "относительно медленным изменением" с нерегулярным периодом и относительно высокой интенсивностью.
- Часто в тренд включаются циклическая и a_0 части, в этом случае модель может быть задана как
$$y(t) = trend(t) + seasonal(t).$$

Trend investigation

Сначала промоделируем временной ряд как имеющий только линейный тренд, взятый с единым периодом выборки.

```
ts = np.arange(20,100)
fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='time (samples)', ylabel='values',
       title='line trend')
plt.show()
```



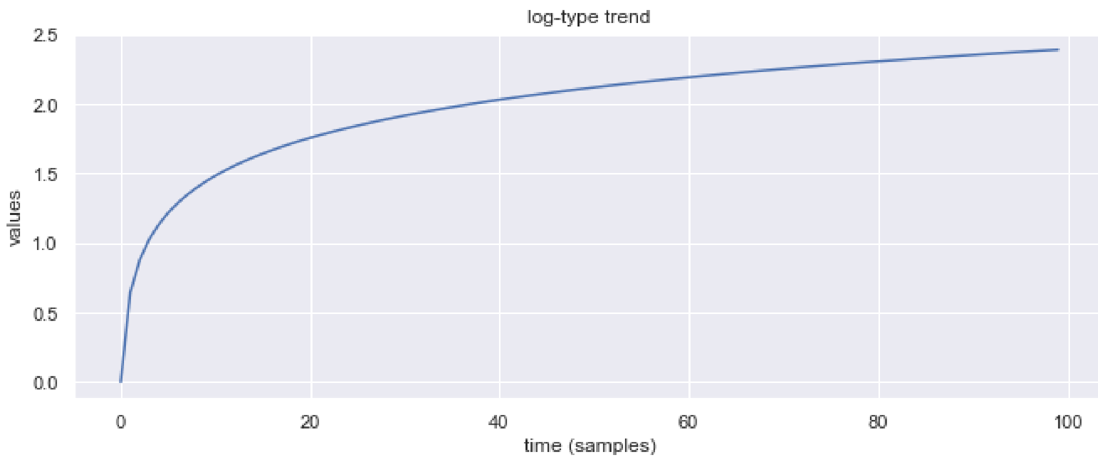
Есть несколько простейших типов трендов, которые могут быть представлены во временных рядах:

- Линейный тренд $y(t) = a \cdot t + b$
- параболический тренд $y(t) = a \cdot t^2 + b \cdot x + c$
- полиномиальный тренд $y(t) = a \cdot t^b + c$
- гиперболический тренд $y(t) = \frac{a}{t^b + c} + d$
- экспоненциальный тренд $y(t) = \exp(a \cdot t + b)$
- насыщение (логистический) тренд $y(t) = \frac{c}{1 + exp(-k(t-m))}$
- логарифмический тренд $y(t) = c\log_b(a \cdot t)$
- многие другие функции, которые, как правило, сглажены, очень медленно меняются или даже монотонны.

Теперь мы можем попробовать логарифмический тренд с основанием e (Число Эйлера, натуральный логарифм) и $a = 4$.

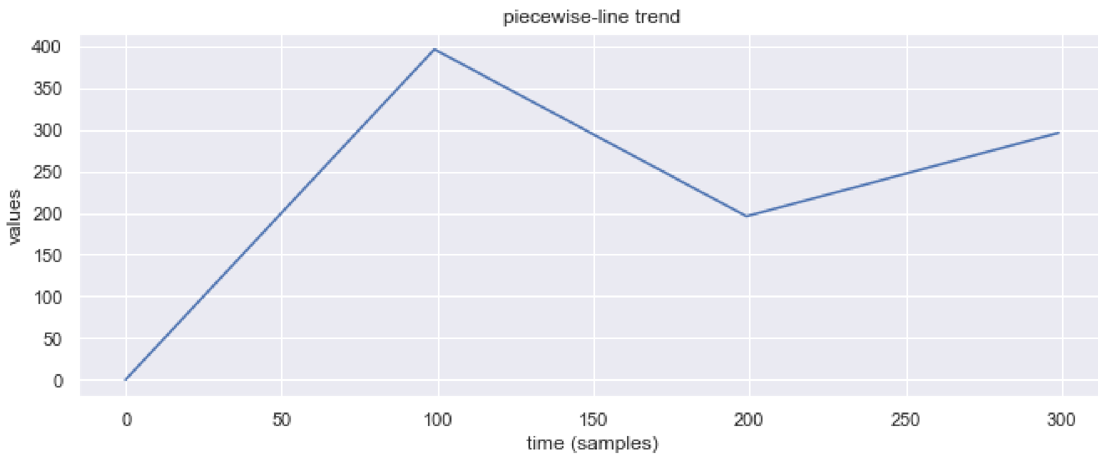
```
N_OF_SAMPLES=100 # Number of samples
a  = 4#const
c  = 0.4
n  = np.arange(N_OF_SAMPLES)
ts = c*np.log(1+a*(n))

fig, ax = plt.subplots()
ax.plot(ts)
```

```
ax.set(xlabel='time (samples)', ylabel='values',
       title='log-type trend')
plt.show()
```



Для многих реальных временных рядов кусочно-монотонное поведение является естественным, поэтому часто необходимо моделировать кусочно-монотонный тренд с одной или несколькими точками перегиба.

```
N_OF_SAMPLES=100 # Number of samples

a   = 4#const
n   = np.arange(N_OF_SAMPLES)
ts1 =a*n

a   = 2#const
n   = np.arange(1,N_OF_SAMPLES+1)
ts2 = ts1[-1]-a*n

a   = 1#const
n   = np.arange(1,N_OF_SAMPLES+1)
ts3 = ts2[-1]+a*n

ts = np.concatenate((ts1,ts2,ts3))

fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='time (samples)',
       ylabel='values',
       title='piecewise-line trend')
plt.show()
```



Давайте теперь смоделируем поведение кусочно-линейного тренда, предложенное моделью Facebook Prophet,

$$y(t) = (k + a(t)^T\delta)t + m + a(t)^T\gamma,$$

где

- $a(t)$ матрица изменения роста, описывающая точки перегиба t_j (матрица с единицами),
- k постоянная скорости роста,
- m смещение,
- δ вектор изменения скорости роста,
- γ коэффициент изменения роста $\gamma_j = t_j\delta_j$,
- s_j точки перегиба.

Notes:

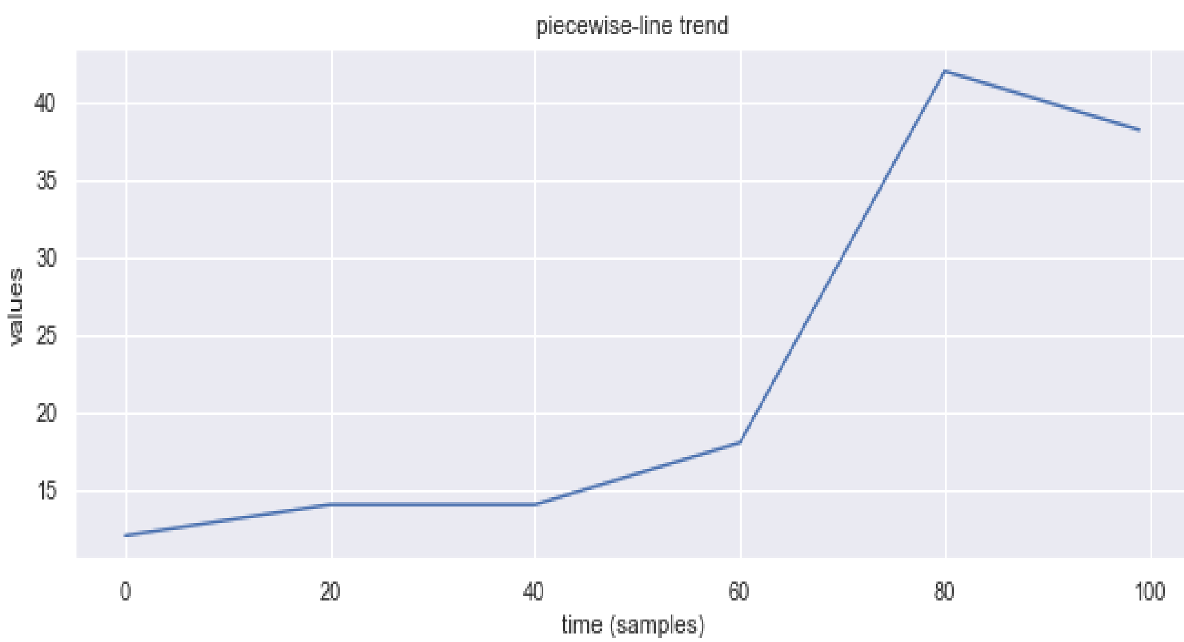
В простейшем случае модель сводится к $y(t) = kt + m$ для временного ряда без точек перегиба t_j.

В модели Facebook Prophet предложили рассматривать логистическую модель как альтернативу линейной, в этом случае тренд можно представить в виде

$$y(t) = \frac{c(t)}{1+exp(-(k+a(t)^T\delta)(t-m-a^T\gamma))}.$$

```
N_OF_SAMPLES=100 # Number of samples

k = 0.1
m = 12

n = np.arange(N_OF_SAMPLES)

inflection_points = np.array([20, 40, 60, 80])#change points
```

```python
a = np.zeros(shape=(inflection_points.size, N_OF_SAMPLES)) #  the matrix of growth changing

# fill matrix
# n[:,None] -mean add new dimention,
#(n[:,None] > inflection_points) is the logic operation to fill matrix with false, true
#(n[:,None] > inflection_points)*1 prodece 1 for true and 0 for false
a = ((n[:,None] > inflection_points) * 1).T


delta = np.array([-0.1, 0.2, 1,  -1.4])#vector with growth rate adjustments

growth = (k + np.dot(a.T,delta))

gamma = -inflection_points * delta
offset = m + np.dot(a.T,gamma)

ts = growth* n + offset


fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='time (samples)',
       ylabel='values',
       title='piecewise-line trend')
plt.show()
```



Упражнение 1

- Реализуйте модель логистического тренда Facebook Prophet

$$y(t) = \frac{c(t)}{1+exp(-(k+a(t)^T\delta)(t-m-a^T\gamma))}$$

Простейшую сезонную часть временного ряда можно представить в виде

$$s(t) = a \cdot sin(\frac{2\pi t}{T} + \theta_0) = a \cdot sin(\frac{2\pi n T_s}{T} + \theta_0) = a \cdot sin(\frac{2\pi f n}{f_s} + \theta_0),$$

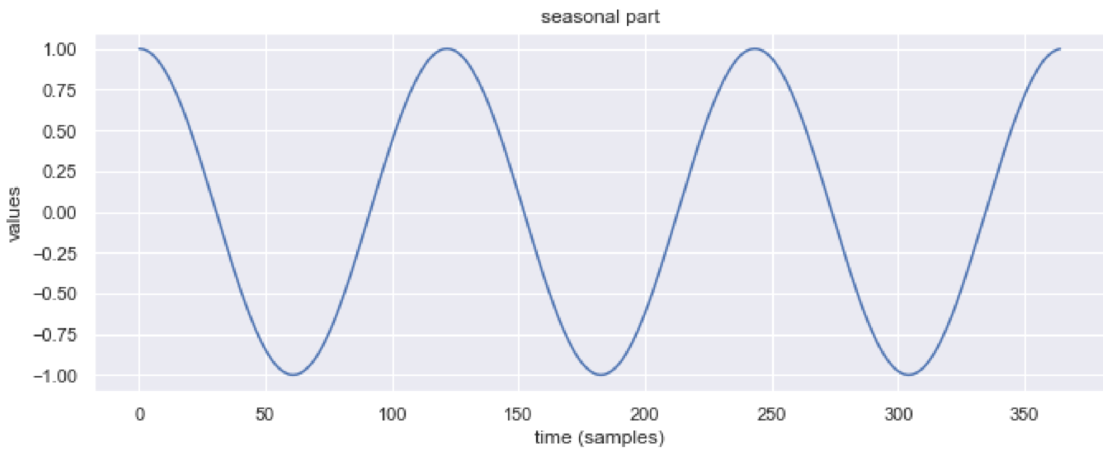
где:

- a интенсивность сезонной компоненты;
- T период сезонности (месяц, день, неделя и т.д.);
- θ_0 начальный сдвиг (начальная фаза) сезонности;
- T_s период дискретизации;
- f и f_s частота сезонности ($f = 1/T$) и частота дискретизации $f_s = 1/T_s$.

Отметим, что в соответствии с теоремой Шеннона-Найквиста-Котельникова минимальное значение f_s должно быть $f_s \geq 2f \rightarrow T \geq 2T_s$. Для оценки приведенного количества периодов используйте $N_{periods} = N \cdot T\_s/T$

```
N_OF_SAMPLES=365 # Number of samples
n  = np.arange(N_OF_SAMPLES)
a = 1
Ts = 1/365
T =1/3
theta = np.pi/2
print('number of periods = ',N_OF_SAMPLES*Ts/T)
ts = a*np.sin(2*np.pi*n*Ts/T+theta)
fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='time (samples)',
       ylabel='values',
       title='seasonal part')
plt.show()
```

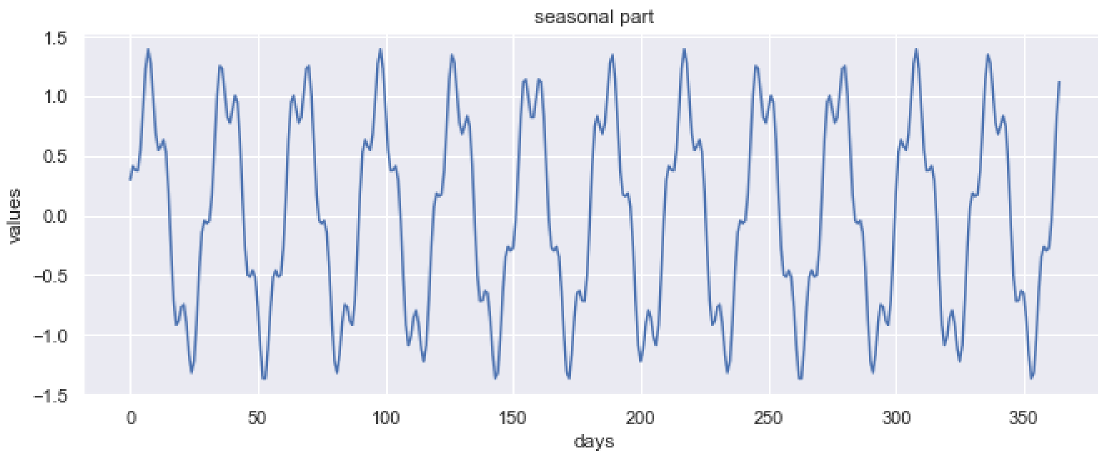
```
number of periods =  3.0
```



Теперь давайте смоделируем более сложную сезонность в году, например, месяц и неделю, которые мы сделаем аддитивно, так что:

$$seasonality = \sum_{i=0}^{M} a_i \cdot sin(2\pi n T_s/T_i + \theta_i)$$

```
N_OF_DAYS=365# Number of samples

days   = np.arange(N_OF_DAYS)

a_w = 0.3 #weak influence

a_m = 1.1 #month influence

T_w = 7/365

T_m = 30/365

Ts = 1/365

theta_w = np.pi/2

theta_m = 0

ts = a_w*np.sin(2*np.pi*days*Ts/T_w + theta_w)+a_m*np.sin(2*np.pi*days*Ts/
T_m + theta_m)

fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='days',
       ylabel='values',
       title='seasonal part')
plt.show()
```



Упражнение 2

- Для предыдущей модели добавьте квартальную сезонность.

Теперь мы можем промоделировать аддитивные и мультипликативные временные ряды.

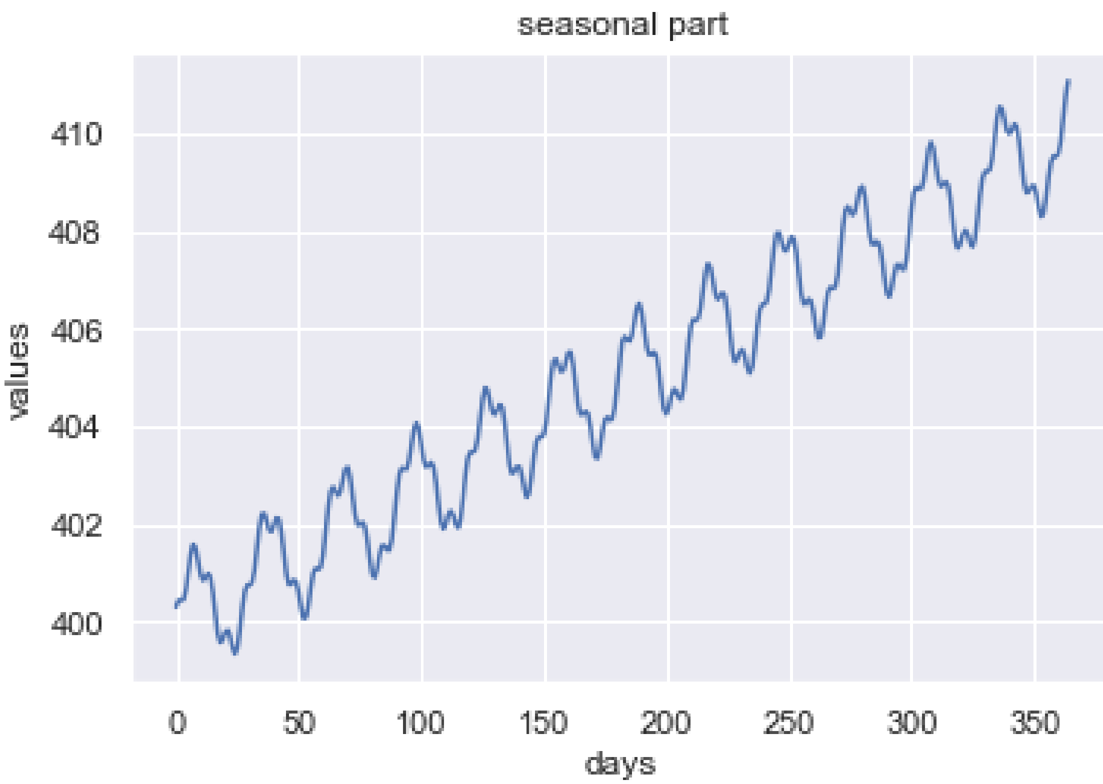
$$y(t) = seasonality(t) + trend(t)$$

В нашем случае смоделируем их так:

$$y(t) = bias_{trend} + a_{trend} nT_s + a_m \cdot sin(2\pi nT_s/T_m) + a_w \cdot sin(2\pi nT_s/T_w)$$

```
YEAR = 365

WEEK = 7

MONTH = 30

N_OF_DAYS=YEAR# Number of samples

days  = np.arange(N_OF_DAYS)

a_w = 0.3 #weak influence

a_m = 1.1 #month influence

T_w = WEEK/YEAR

T_m = MONTH/YEAR

Ts = 1/YEAR

theta_w = np.pi/2

theta_m = 0
```

```
a_trend = 10 #slope

bias_trend = 400

trend = a_trend*days*Ts+bias_trend

seasonality = a_w*np.sin(2*np.pi*days*Ts/T_w + theta_w)+a_m*np.sin(2*np.pi
*days*Ts/T_m + theta_m)

ts =trend + seasonality

fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='days',
       ylabel='values',
       title='seasonal part')
plt.show()
```



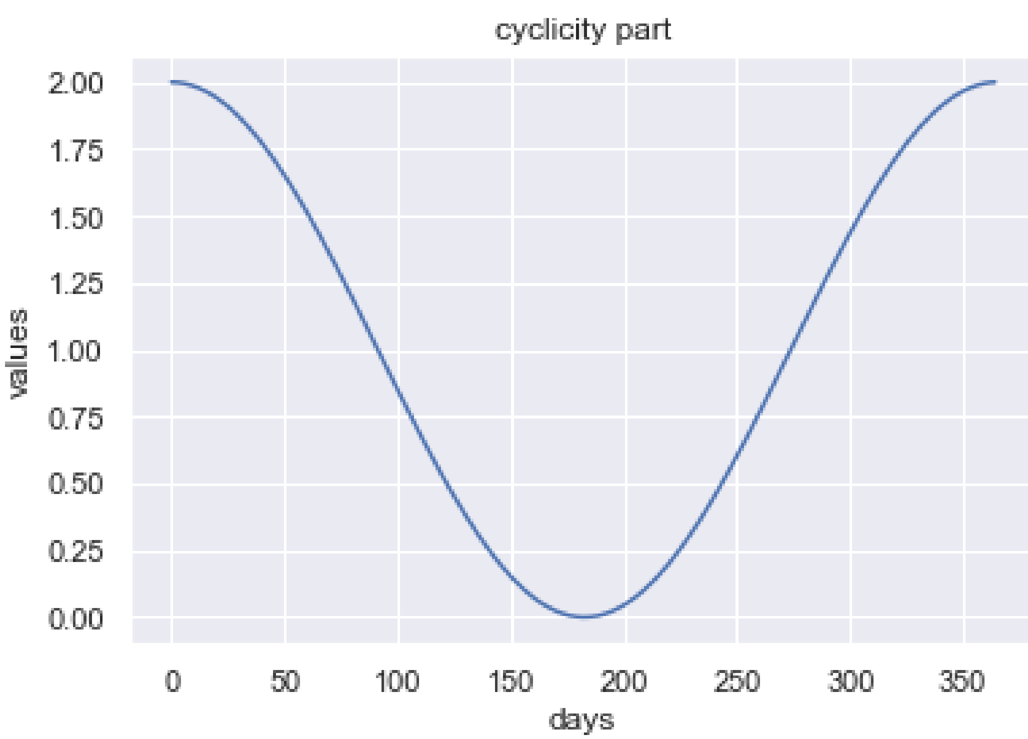
Упражнение 3

- Реализовать мультипликативную детерминированную модель временных рядов с сезонной частью и частью тренда.
- Реализовать модель логистического тренда с аддитивной сезонностью.

Циклическая часть

Помимо тренда и сезонности, мы можем добавить некоторую цикличность (в качестве альтернативы можно рассматривать как медленное изменение тренда). Давайте смоделируем цикличность как некоторую зависимость год-сезон, например, в приведенном ниже примере мы моделируем падение продаж в середине года (летом).

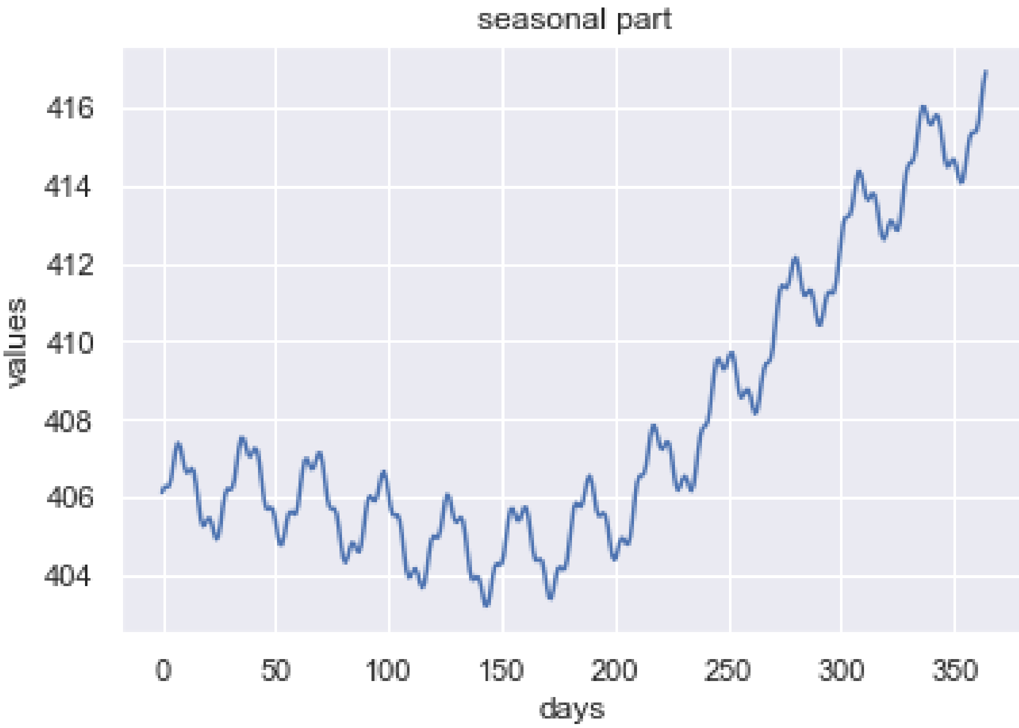
```
a_cycl = 1
T_cycl = 1
cyclicity = a_cycl +a_cycl *np.sin(2*np.pi*days*Ts/T_cycl + np.pi/2)

ts =cyclicity

fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='days',
       ylabel='values',
       title='cyclicity part')
plt.show()
```



Теперь можем добавить это к тренду

```
YEAR = 365

WEEK = 7

MONTH = 30

N_OF_DAYS=YEAR# Number of samples
```

```
days  = np.arange(N_OF_DAYS)

a_w = 0.3 #weak influence

a_m = 1.1 #month influence

T_w = WEEK/YEAR

T_m = MONTH/YEAR

Ts = 1/YEAR

theta_w = np.pi/2

theta_m = 0

a_trend = 10 #slope

bias_trend = 400

trend = a_trend*days*Ts+bias_trend

seasonality = a_w*np.sin(2*np.pi*days*Ts/T_w + theta_w)+a_m*np.sin(2*np.pi
*days*Ts/T_m + theta_m)

a_cycl = 2.91
T_cycl = 1
cyclicity = a_cycl+a_cycl *np.sin(2*np.pi*days*Ts/T_cycl + np.pi/2)

ts =trend + seasonality + cyclicity

fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='days',
       ylabel='values',
       title='seasonal part')
plt.show()
```



Упражнение 4

- Реализуйте мультипликативную детерминированную модель временных рядов с сезонной, циклической и трендовой частями.

Моделирование редких и регулярных явлений

Помимо тренда и регулярной сезонности, в модель временных рядов могут быть внесены конкретные события. Например, если мы моделируем временные ряды продаж, будет интересно добавить изменение спроса в будние и выходные дни. В простейшем случае это можно сделать так:

$$\mathrm{week\_days}(day) = \sum_{i=1}^{7} a_i\, \delta(\lfloor (day-1)/7 \rfloor + 1 - i),$$

где

- $\lfloor day/7 \rfloor$ - остаток деления;
- δ дельта функция Кронокера,

$$\delta(i,j) = \delta(i-j) = \begin{cases} 1 & \mathrm{if} i = j \\ 0 & \mathrm{otherwise} \end{cases}$$

- i номер дня недели ($i = 1,2,3,4,5,6,7$).

Отметим, что если необходимо считать не с 1, то можно описать их влияние следующим образом, с использованием переменной shift:

$$\text{week\_days}(day) = \sum_{i=1}^{7} a_i\,\delta(\lfloor(day - 1 + shift - 1)/7\rfloor + 1 - i),$$

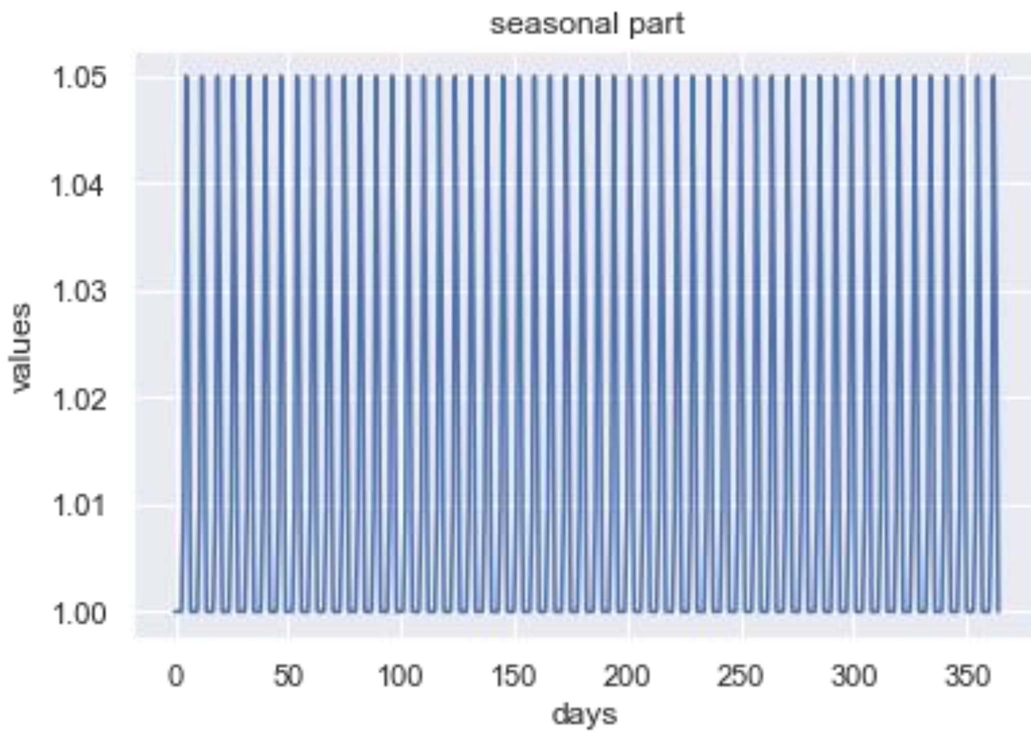
Давайте проверим результат

```
N_OF_DAYS =14
shift = 1
days = np.arange(1,N_OF_DAYS+1)
print((days-1+(shift-1))%7+1)
```

```
[1 2 3 4 5 6 7 1 2 3 4 5 6 7]
```

Теперь можно ввести коэффициенты от дня a_{week}

```
N_OF_DAYS=365

days  = np.arange(N_OF_SAMPLES)

# week days coefficients
a_week = np.array([1, 1, 1, 1, 1.01, 1.05, 1.03])

#for the number of days multiples of the week
week_days = list(a_week)*int(N_OF_DAYS/7)

# add rest of the days
week_days = np.array([*week_days,*a_week[:N_OF_DAYS%7]])

#check that week_days size equal to N_OF_DAYS
assert week_days.size==N_OF_DAYS

ts = week_days

fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='days',
       ylabel='values',
       title='seasonal part')
plt.show()
```

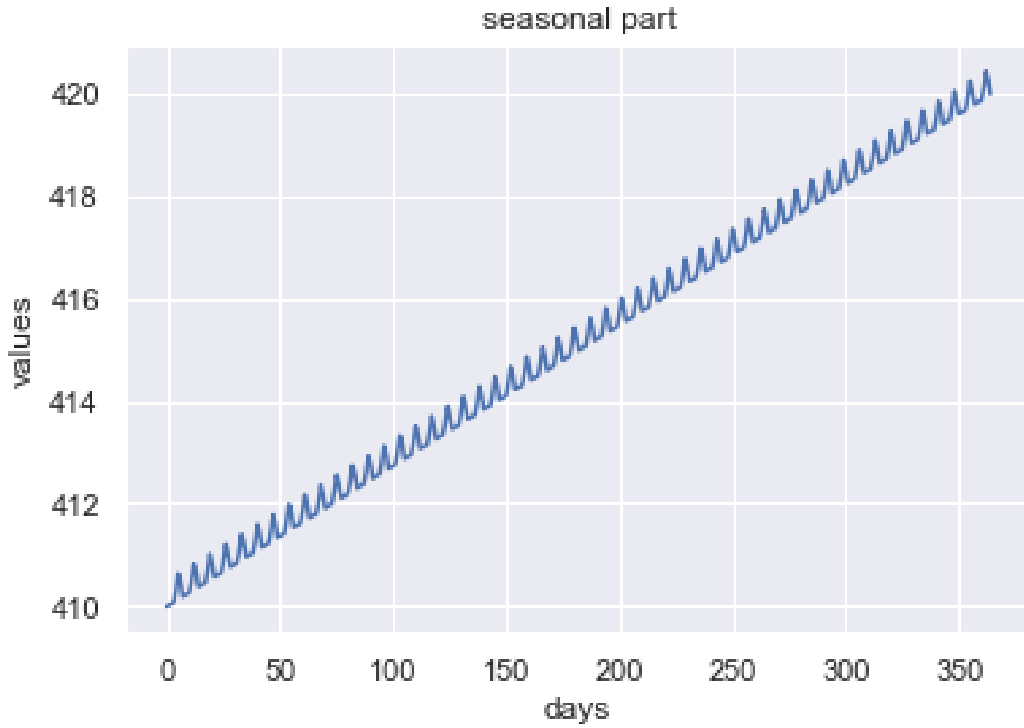


Давайте проверим как дневные изменения выглядят на фоне линейного тренда

$$y(day) = bias_{trend} + a_{trend} day + a_{trend}\, week\_days,$$

где $week\_days$ номер дня.

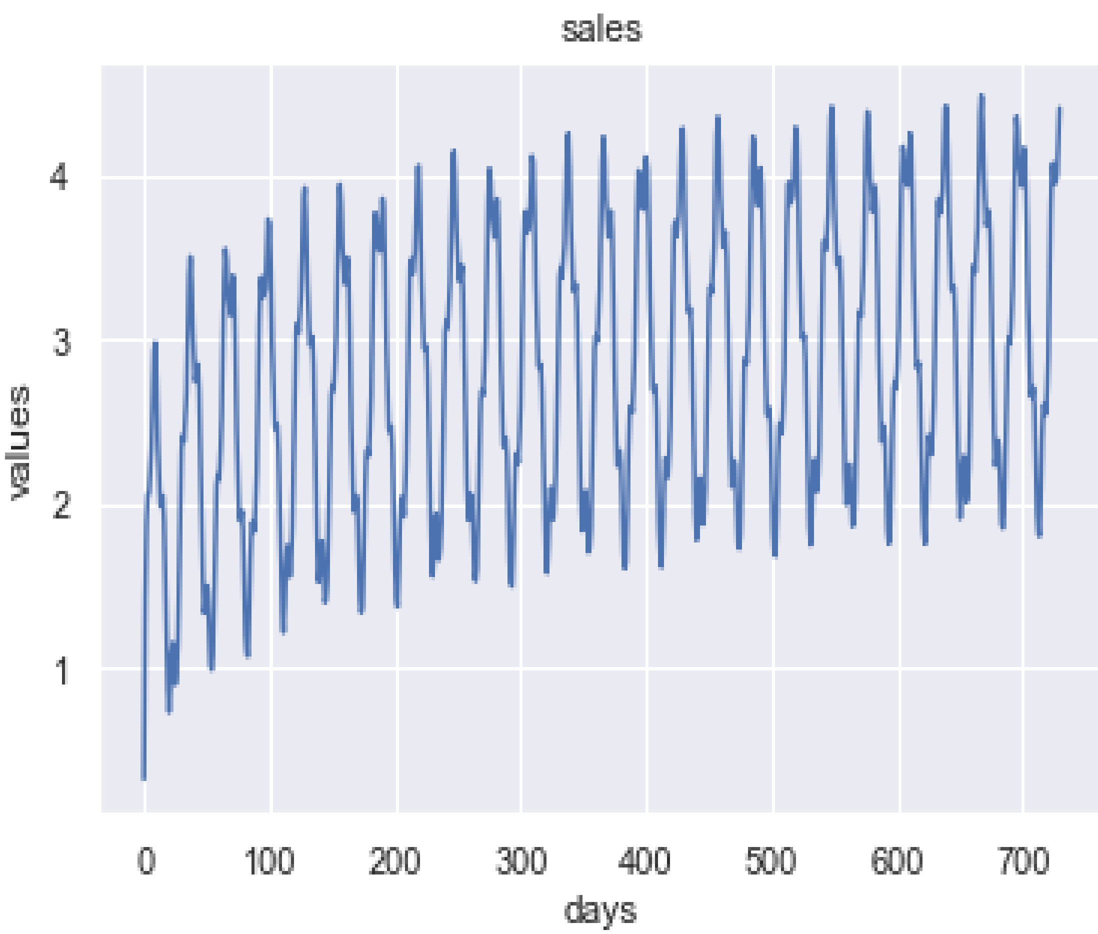
```
N_OF_DAYS = 365
days      = np.arange(N_OF_SAMPLES)

a_trend    = 10 #slope
bias_trend = 400
week_coefficients = np.array([1, 1, 1, 1, 1.02, 1.05, 1.03])

a_week = week_coefficients*a_trend

week_days = np.array([*list(a_week)*int(N_OF_DAYS/7), *a_week[:N_OF_DAYS%7
]])

trend = a_trend*n*Ts+bias_trend

ts =week_days + trend

fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='days',
       ylabel='values',
       title='seasonal part')
plt.show()
```



Теперь добавим сезонную часть.

```
YEAR = 365

WEEK = 7

MONTH = 30

N_OF_DAYS=YEAR*2# Number of samples

days  = np.arange(N_OF_DAYS)

a_w = 0.3 #weak influence

a_m = 1.1 #month influence

T_w = WEEK/YEAR

T_m = MONTH/YEAR

Ts = 1/YEAR

a_trend = 5
c_trend = 0.34

week_coefficients = np.array([0.95, 1, 1, 1, 1, 1.25, 1.03])

a_week = week_coefficients*c_trend
```

```
trend = c_trend*np.log(1+a_trend*days)

seasonality = a_w*np.sin(2*np.pi*days*Ts/T_w )+a_m*np.sin(2*np.pi*days*Ts/T_m
)

week_days = np.array([*list(a_week)*int(N_OF_DAYS/7), *a_week[:N_OF_DAYS%7]])


ts     = week_days + trend  + seasonality

fig, ax = plt.subplots()
ax.plot(ts)
ax.set(xlabel='days',
       ylabel='values',
       title='sales')
plt.show()
```



Упражнение 5

- добавить падение спроса в праздничные дни в начале года к временным рядам, реализованным выше примере.

Симуляция случайного поведения

Белый гауссов шум

Помимо детерминированной части временного ряда, полезно смоделировать его стохастическое поведение. Стохастическое поведение временного ряда, в первую очередь связанное с влиянием шума. Самая простая и наиболее распространенная модель шума - это белый гауссовский шум (WGN) (почти что тоже самое, что и модели независимого и одинаково распределенного (i.i.d) шума). WGN имеет нормальное распределение с нулевым средним значением и дисперсией σ^2:

$$noise(t) \sim N(0,\sigma);$$

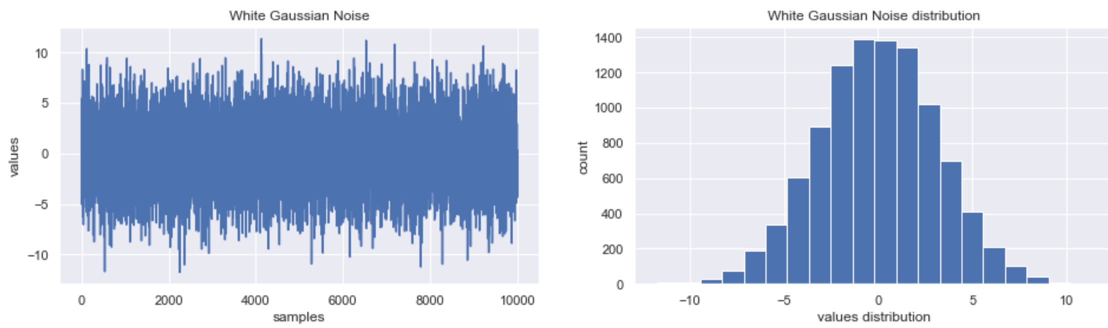
Нормальное распределение имеет вид:

$$P(t) = N(0,\sigma) = \frac{1}{\sigma\sqrt{2\pi}exp\left(-\frac{t^2}{2\sigma^2}\right)}$$

где σ средний разброс шумов или их стандартное отклонение (корень дисперсии).

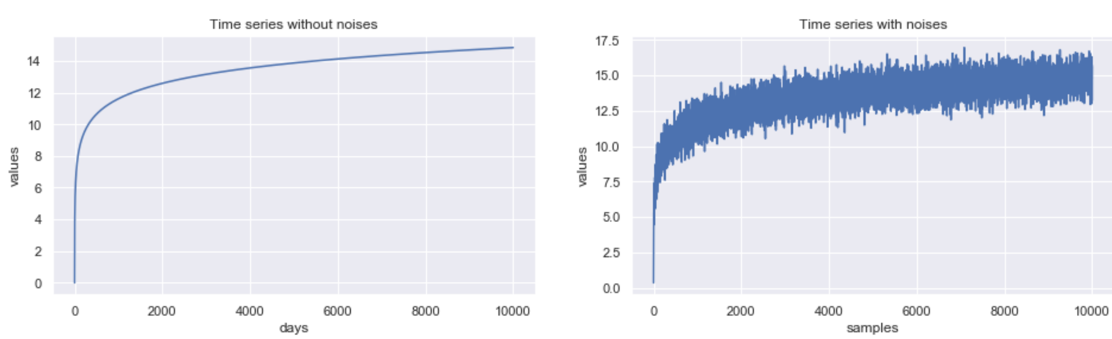
Построим модель такого шума.

```
N_OF_SAMPLES = 10000

noise_power  = 10

wgn = np.sqrt(noise_power)*np.random.normal(size = N_OF_SAMPLES)

ts = wgn

fig, ax = plt.subplots(1,2)

ax[0].plot(ts)
ax[0].set(xlabel='samples',
       ylabel='values',
       title='White Gaussian Noise')

ax[1].hist(ts, bins = 20)
ax[1].set(xlabel='values distribution',
       ylabel='count',
       title='White Gaussian Noise distribution')
plt.show()
```



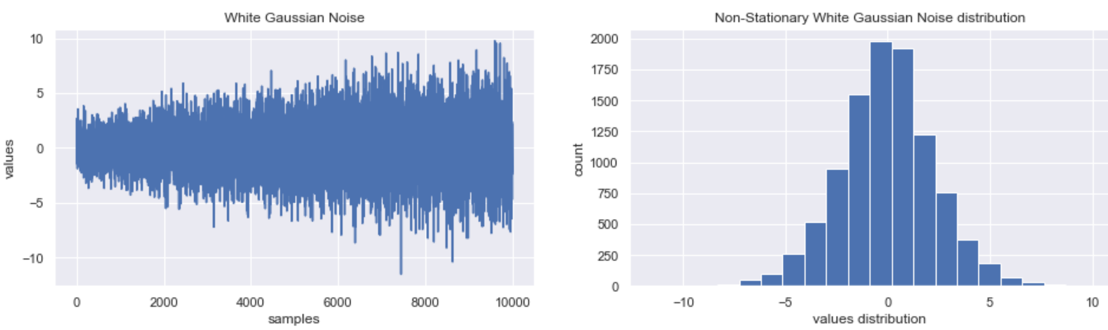
Теперь посмотрим на влияние шумов на временной ряд

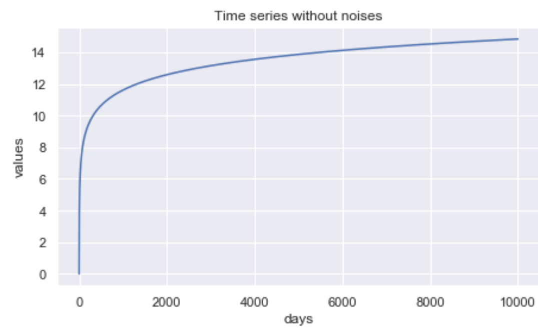
```
N_OF_SAMPLES = 10000

noise_power  = 0.5

wgn = (np.sqrt(noise_power))*(np.random.normal(size = N_OF_SAMPLES))

a  = 4#const
c  = 1.4
n  = np.arange(N_OF_SAMPLES)
ts = c*np.log(1+a*(n))

ts_wn = ts + wgn

fig, ax = plt.subplots(1,2)

ax[0].plot(ts)
ax[0].set(xlabel='days',
       ylabel='values',
       title='Time series without noises')

ax[1].plot(ts_wn)
ax[1].set(xlabel='samples',
       ylabel='values',
       title='Time series with noises')
plt.show()
```

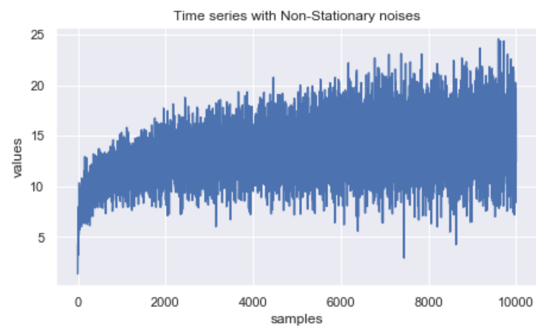


Помимо равномерно распределенного шума, соответствующего стационарной модели шума, важно моделировать нестационарные случаи. Самый простой случай - это линей но возрастающая дисперсия,

```python
N_OF_SAMPLES = 10000
a  = 4#const
c  = 1.4

noise_power  = np.linspace(1,10,N_OF_SAMPLES) #linearly growing noise power

wgn = np.sqrt(noise_power)*np.random.normal(size = N_OF_SAMPLES)

ts = c*np.log(1+a*np.arange(N_OF_SAMPLES))

ts_wn = ts + wgn

fig, ax = plt.subplots(1,2)

ax[0].plot(wgn)
ax[0].set(xlabel='samples',
       ylabel='values',
       title='White Gaussian Noise')

ax[1].hist(wgn, bins = 20)
ax[1].set(xlabel='values distribution',
       ylabel='count',
       title='Non-Stationary White Gaussian Noise distribution')
fig, ax = plt.subplots(1,2)
ax[0].plot(ts)
ax[0].set(xlabel='days',
       ylabel='values',
       title='Time series without noises')

ax[1].plot(ts_wn)
ax[1].set(xlabel='samples',
       ylabel='values',
       title='Time series with Non-Stationary noises')
plt.show()
```







Exercise 6

- Исследовать влияние аддитивного стационарного и нестационарного белого шума на временные ряды с сезонными частями и частями тренда в следующей форме
$$y(t) = a_0 + trend(t) + seasonal(t) + cyclic(t) + \text{rare\_events(t)} + noise(t).$$
- Смоделировать модель временного ряда в следующем виде:
$$y(t) = a_0 + (trend(t) \cdot cyclic(t) + seasonal_1(t)) \cdot seasonal_2(t) + noise(t).$$

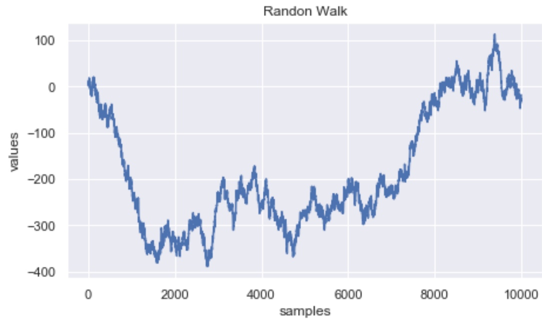
Модель случайного блуждания

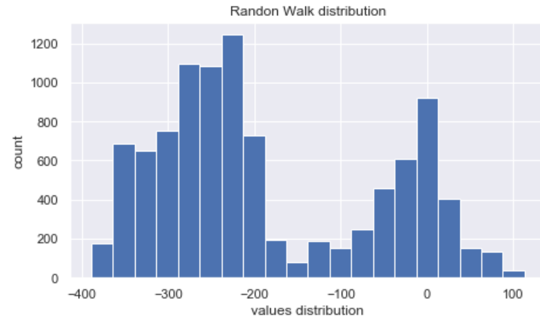
Помимо аддитивного шума, важной моделью шума является случайное блуждание, которое в простейшем случае можно смоделировать как

$$y(t) = y(t-1) + \eta(t),$$

где $\eta(t) \sim N(0, \sigma^2)$. Такая модель широко распространена во многих бизнес процессах и не только.

```
N_OF_SAMPLES = 10000

noise_power  = 10

wgn = np.sqrt(noise_power)*np.random.normal(size = N_OF_SAMPLES)

ts = np.cumsum(wgn )

fig, ax = plt.subplots(1,2)

ax[0].plot(ts)
ax[0].set(xlabel='samples',
       ylabel='values',
       title='Randon Walk')

ax[1].hist(ts, bins = 20)
ax[1].set(xlabel='values distribution',
       ylabel='count',
       title='Randon Walk distribution')
plt.show()
```





Упражнение 7

1. Исследуйте 3 модели случайного блуждания:

- Модель с дрифтом:
  $y(t) = \alpha + y(t-1) + \eta(t)$
- Модель с трендом:
  $y(t) = \alpha + \beta \cdot t + y(t-1) + \eta(t)$
- Модель с изменением интенсивности:
  $y(t) = y(t-1) + \eta(t), \ \eta(t) \sim N\left(0, \sigma^2(t)\right), \qquad \sigma^{2(t)} = \sigma_0 + \gamma \cdot t$