


# Пропуски во временных рядах

[Материалы](#) > [Анализ и обработка данных](#)

На прошлом занятии мы поговорили про работу с пропущенными значениями в перекрестных данных. Отдельный подход требуется для заполнения пропусков во временных рядах.

Откроем блокнот к этому занятию 

## Подготовка данных

Продолжим работать с данными об авиаперевозках и рождаемости. Скачаем и подгрузим датасеты.

passengers.csv

[Скачать](#)

births.csv

[Скачать](#)

```
1 # импортируем данные
2 passengers = pd.read_csv('/content/passengers.csv')
3 births = pd.read_csv('/content/births.csv')
```

Почему именно эти два датасета?

- во-первых, у них разный шаг временного ряда, пассажирские перевозки — это ежемесячные данные, рождаемость — посуточные; это имеет значение для заполнения пропусков;
- во-вторых, первый временной ряд нестационарен, то есть в нем есть тренд и сезонность, второй — стационарен, как мы увидим, это также влияет на выбор метода заполнения пропусков.

Добавим пропуски в данные о пассажирских перевозках.

```
1 import random
2
3 random.seed(1)
4
5 # переименуем столбец #Passengers в reference
6 passengers.rename(columns = {'#Passengers' : 'reference'}, inplace = True)
```

```
7 |
8 | # сделаем две копии этого столбца с названиями target и missing
9 | passengers['target'] = passengers.reference
10 | passengers['missing'] = passengers.reference
11 |
12 | # посчитаем количество наблюдений
13 | n_samples = len(passengers)
14 | # вычислим 20 процентов от этого числа,
15 | # это будет количество пропусков
16 | how_many = int(0.20 * n_samples)
17 |
18 | # случайным образом выберем 20 процентов значений индекса
19 | mask_target = random.sample(list(passengers.index), how_many)
20 | # и заполним их значением NaN в столбце target
21 | passengers.iloc[mask_target, 2] = np.nan
22 |
23 | # найдем оставшиеся значения индекса
24 | mask_missing = list(set(passengers.index) - set(mask_target))
25 | # сделаем их NaN и поместим в столбец missing
26 | passengers.iloc[mask_missing, 3] = np.nan
27 |
28 | # переведем столбец Month в формат datetime
29 | passengers.index = pd.to_datetime(passengers.Month)
30 | passengers.drop(columns = ['Month'], inplace = True)
```

Посмотрим на результат.

```
1 | # посчитаем количество пропусков в каждом столбце
2 | passengers.isnull().sum()
```

```
1 | reference      0
2 | target        28
3 | missing       116
4 | dtype: int64
```

```
1 | passengers.head(3)
```

	reference	target	missing
Month			
1949-01-01	112	NaN	112.0
1949-02-01	118	118.0	NaN
1949-03-01	132	NaN	132.0

Обсудим,

- в столбце reference сохранились все данные без пропусков;
- в столбце target мы случайным образом создали 20 процентов пропущенных значений, именно этот столбец мы будем использовать в процессе заполнения пропусков;
- в столбце missing, наоборот, пропусками являются те значения, которые заполнены в target, этот столбец мы будем использовать исключительно для построения визуализаций.

Прделаем то же самое с датасетом о рождаемости.

```
1 | random.seed(1)
2 |
3 | births.rename(columns = {'Births' : 'reference'}, inplace = True)
4 | births['target'] = births.reference
```

```

5 | births['missing'] = births.reference
6 |
7 | n_samples = len(births)
8 | how_many = int(0.15 * n_samples)
9 |
10 | mask_target = random.sample(list(births.index), how_many)
11 | births.iloc[mask_target, 2] = np.nan
12 |
13 | mask_missing = list(set(births.index) - set(mask_target))
14 | births.iloc[mask_missing, 3] = np.nan
15 |
16 | births.index = pd.to_datetime(births.Date)
17 | births.drop(columns = ['Date'], inplace = True)

```

```
1 | births.isnull().sum()
```

```

1 | reference      0
2 | target        54
3 | missing       311
4 | dtype: int64

```

```
1 | births.head(3)
```

	reference	target	missing
Date			
1959-01-01	35	35.0	NaN
1959-01-02	32	NaN	32.0
1959-01-03	30	30.0	NaN

## Визуализация

Выведем данные на графиках. Перед этим в учебных целях сократим временные интервалы:

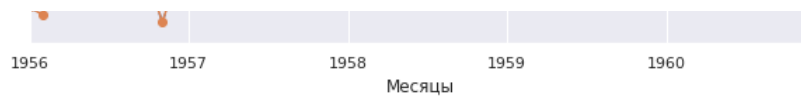
- это упростит применение методов заполнения пропусков, поскольку на полных временных рядах некоторые методы оставляют отдельные пропуски незаполненными;
- кроме того, иллюстрации станут более наглядными.

```

1 | passengers = passengers['1956-01':'1960-12']
2 |
3 | ax = passengers.plot(style=['--', 'o-', 'o'])
4 | ax.set(title = 'Перевозки пассажиров с 1956 по 1960 год',
5 |       xlabel = 'Месяцы',
6 |       ylabel = 'Количество пассажиров');

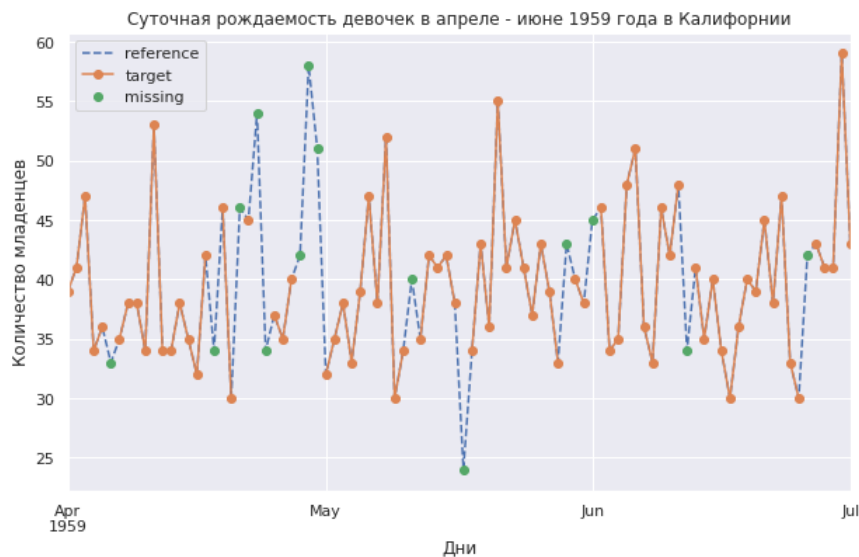
```





Сделаем то же самое с данными о рождаемости.

```
1 births = births['1959-04-01':'1959-07-01']
2
3 ax = births.plot(style=['--', 'o-', 'o'])
4 ax.set(title = 'Суточная рождаемость девочек в апреле - июне 1959 года в Калифорнии',
5         xlabel = 'Дни',
6         ylabel = 'Количество младенцев');
```



Итак, говоря проще, нам нужно заполнить значения зеленых точек на основе значений оранжевых точек.

## Заполнение средним и медианой

Самый простой подход к заполнению пропусков — использовать среднее арифметическое или медиану:

- используем **метод .assign()** для создания новых столбцов, в которые будем помещать результат очередного метода;
- для заполнения пропусков будем использовать **метод .fillna()**, которому передадим одно или несколько значений для заполнения пропусков.

```
1 # передадим в метод .fillna() среднее арифметическое и медиану
2 passengers = passengers.assign(FillMean = passengers.target.fillna(passengers.target.me
3 passengers = passengers.assign(FillMedian = passengers.target.fillna(passengers.target.
```

```
1 # сделаем то же самое для данных о рождаемости
2 births = births.assign(FillMean = births.target.fillna(births.target.mean()))
3 births = births.assign(FillMedian = births.target.fillna(births.target.median()))
```

## Заполнение предыдущим и последующим значениями

Во временных рядах может присутствовать автокорреляция между значениями, и есть смысл использовать это для заполнения пропусков:

- подход **last observation carried forward** (LOCF) предполагает, что мы берем предыдущее от пропущенного значение и заполняем им пропуск;
- подход **next observation carried backward** (NOCB), наоборот, заполняет пропуск последующим значением.

В библиотеке Pandas для реализации этих подходов есть соответственно **методы .ffill()** и **.bfill()**.

```
1 # заполним пропуски предыдущим значением
2 passengers = passengers.assign(FFill = passengers.target.ffill())
3 births = births.assign(FFill = births.target.ffill())
```

```
1 # заполним пропуски последующим значением
2 passengers = passengers.assign(BFill = passengers.target.bfill())
3 births = births.assign(BFill = births.target.bfill())
```

Это простые и понятные методы, которые, тем не менее, могут внести систематическую ошибку, особенно если заполнять таким образом несколько пропусков подряд. Например, предположим, что в данных есть тренд, а мы шесть из двенадцати месяцев заполнили одним значением (константой). Как следствие, тренд был удален из данных.

Указанную проблему можно попытаться решить через *параметр limit*, который ограничивает количество последовательно заполняемых одним и тем же значением пропусков. Другими словами, если пропущено четыре значения подряд, а *limit* установлен на уровне двух, предыдущим либо последующим наблюдением будут заполнены только два из четырех значений.

## Заполнение скользящим средним и медианой

Для заполнения пропусков можно использовать не одно предыдущее или последующее значение, а несколько. Вспомним про скользящее среднее, которое рассчитывается с помощью **методов .rolling()** и **.mean()**. Помимо среднего арифметического ничто не мешает нам рассчитать медиану окна и заполнить ею пропущенное значение.

Здесь нужно уточнить настройки:

- размер окна (параметр *window*);
- минимальное количество периодов для расчета среднего и медианы (*min\_periods*); установим этот параметр на уровне *min\_periods = 1*, чтобы избежать сохранения незаполненных значений.

```
1 # рассчитаем скользящее среднее и медиану для данных о пассажирах
2 passengers = passengers.assign(RollingMean =
3                               passengers.target.fillna(
4                                   passengers.target.rolling(window = 5,
5                                                           min_periods = 1).mean()))
6
7 passengers = passengers.assign(RollingMedian =
8                               passengers.target.fillna(
9                                   passengers.target.rolling(window = 5,
10                                                            min_periods = 1).median())
```

```
1 # рассчитаем скользящее среднее и медиану для данных о рождаемости
2 births = births.assign(RollingMean =
3                         births.target.fillna(
4                             births.target.rolling(window = 5,
5                                                     min_periods = 1).mean()))
6
7 births = births.assign(RollingMedian =
8                         births.target.fillna(
9                             births.target.rolling(window = 5,
10                                                    min_periods = 1).median()))
```

## Интерполяция

### Понятие интерполяции

Для временных рядов, в которых есть тренд, подойдут различные методы **интерполяции** (interpolation). По большому счету, мы берем имеющиеся у нас наблюдения и пытаемся найти (вычислить) проходящую через них функцию.

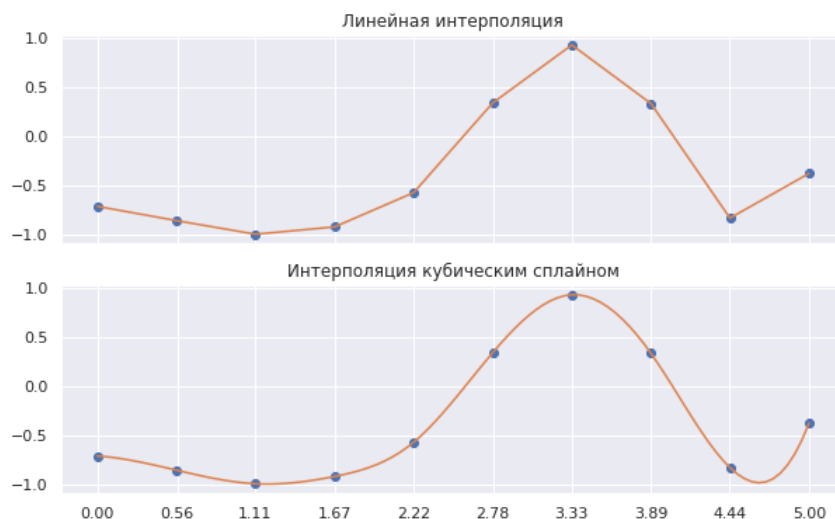
Что важно знать про интерполяцию:

- функция, найденная в процессе интерполяции, *проходит строго через известные точки* (этим она отличается от **аппроксимации**, которая стремится с помощью простой функции описать более сложную, и **регрессии**, которая стремится найти минимизирующую ошибку функцию);
- найденная функция описывает только *заданный известными точками интервал и не выходит за его пределы* (этим она отличается от **экстраполяции**).

Введем некоторые термины:

- значения  $x$  каждой из известных точек, по которым строится интерполирующая функция, называются **узлами** (knots);
- совокупность точек — интерполяционной **сеткой** (grid);
- сама функция называется **интерполянт** (interpolant).

Посмотрим на график ниже (код для построения графика можно найти в ноутбук, он нам сейчас не важен).



Если провести между точками прямые, то получится линейная интерполяция, если попытаться описать точки одним многочленом — полиномиальная, кусочно-заданной функцией, состоящей из нескольких полиномов — сплайн.

Рассмотрим эти способы интерполяции более подробно.

## Способы интерполяции

### Линейная интерполяция

Самый простой способ — построить линейную функцию, проходящую через две соседние точки. Например, предположим, что соседними наблюдениями  $(x_0, y_0)$  и  $(x_1, y_1)$  являются точки с координатами  $(1, 3)$  и  $(3, 5)$ . Найдем значение в точке  $x = 2$ .

Так как все три точки лежат на одной прямой, наклон этой прямой между любыми двумя точками одинаков. Из уравнения наклона,

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$
$$y = \frac{y_0(x_1 - x) + y_1(x - x_0)}{x_1 - x_0}$$

Тогда в точке  $x = 2$

$$y = \frac{3(3 - 2) + 5(2 - 1)}{3 - 1} = 4$$

Если мы хотим найти функцию прямой линии, проходящей через две точки, то можем построить систему линейных уравнений,

$$\begin{cases} y_0 = a_0 + a_1 \cdot x_0 \\ y_1 = a_0 + a_1 \cdot x_1 \end{cases}$$

Подставив известные точки,

$$\begin{cases} 3 = a_0 + a_1 \cdot 1 \\ 5 = a_0 + a_1 \cdot 3 \end{cases}$$

Решив систему линейных уравнений, получим

$$y = x + 2$$

Значение функции в точке  $x = 2$  равно четырем.

### Полиномиальная интерполяция

Не все процессы могут быть точно описаны прямыми линиями. Одним из вариантов решения может быть нахождение полинома второй или большей степени, график которого проходил бы через все известные точки.

Полиномиальная интерполяция имеет ряд недостатков. В частности, с ростом числа точек у

кривой могут возникнуть колебания (осцилляции), которые негативно скажутся на точности заполнения пропусков. Эта особенность называется **феноменом Рунге** (Runge's phenomenon).

## Сплайн

При линейной интерполяции мы строили отдельную линейную функцию между каждой парой известных точек. **Сплайн** (от англ. spline, «чертежное лекало») также состоит из нескольких функций (по одной на каждый отрезок), но не линейных, а полиномиальных.

В частности, это позволяет использовать меньшую (чем при полиномиальной интерполяции) степень каждого из полиномов и, таким образом, преодолеть последствия феномена Рунге.

Перейдем к практике.

## Реализация на Питоне

В Питоне интерполяцию можно выполнить либо с помощью модуля interpolate библиотеки Scipy, либо с помощью во многом опирающегося на этот модуль метода .interpolate() библиотеки Pandas.

Создадим список из названий методов интерполяции, которые передадим в `.interpolate()`

```
1 methods = ['linear', 'polynomial', 'quadratic', 'cubic', 'spline']
```

Обратите внимание, как видно из документации Scipy, значения параметра `method = 'quadratic'` и `method = 'cubic'` метода `.interpolate()` в Pandas рассчитывают квадратичный и кубический сплайны, а не полиномы второй и третьей степени.

В цикле `for` найдем интерполирующую функцию с помощью каждого из методов и используем ее для заполнения пропусков (создания нового столбца с полными значениями).

```
1 # применим каждый из методов к данным о пассажирах
2 for m in methods:
3     if m == 'polynomial':
4         # для полиномиальной интерполяции нужно указать степень полинома
5         # (пока поддерживаются только нечетные степени)
6         passengers[m] = passengers.target.interpolate(method = m, order = 3)
7     elif m == 'spline':
8         # для сплайна порядок должен быть 1 <= k <= 5
9         passengers[m] = passengers.target.interpolate(method = m, order = 5)
10    else:
11        passengers[m] = passengers.target.interpolate(method = m)
```

```
1 # сделаем то же самое с данными о рождаемости
2 for m in methods:
3     if m == 'polynomial':
4         births[m] = births.target.interpolate(method = m, order = 3)
5     elif m == 'spline':
6         births[m] = births.target.interpolate(method = m, order = 5)
7     else:
8         births[m] = births.target.interpolate(method = m)
```



## Сравнение методов

Так как нам предстоит оценивать отклонение одного количественного показателя от другого, в качестве метрики мы можем выбрать корень среднеквадратической ошибки (RMSE).

```
1 # импортируем функцию для расчета RMSE
2 from sklearn.metrics import root_mean_squared_error
```

Напишем функцию для сравнения методов.

```
1 def compare_methods(df):
2     # в цикле list comprehension будем брать по одному столбцу
3     # (итерируя по названиям столбцов)
4     # и рассчитывать корень среднеквадратической ошибки
5     results = [(method, root_mean_squared_error(df.reference, df[method])).round(2)) for
6     # преобразуем получившийся список вначале в массив Numpy, затем в датафрейм
7     results = pd.DataFrame(np.array(results), columns = ['Method', 'RMSE'])
8     # отсортируем по размеру ошибки в возрастающем (по умолчанию) порядке
9     results.sort_values(by = 'RMSE', inplace = True)
10    # сбросим индекс
11    results.reset_index(drop = True, inplace = True)
12    return results
```

Сравним методы заполнения пропусков.

```
1 # сравним методы для данных о пассажирах
2 passengers_results = compare_methods(passengers)
3 passengers_results
```

	Method	RMSE
0	spline	12.29
1	polynomial	12.47
2	cubic	12.47
3	quadratic	12.72
4	linear	19.26
5	BFill	23.32
6	FFill	28.96
7	RollingMean	40.44
8	RollingMedian	43.35
9	FillMedian	49.79
10	FillMean	50.47

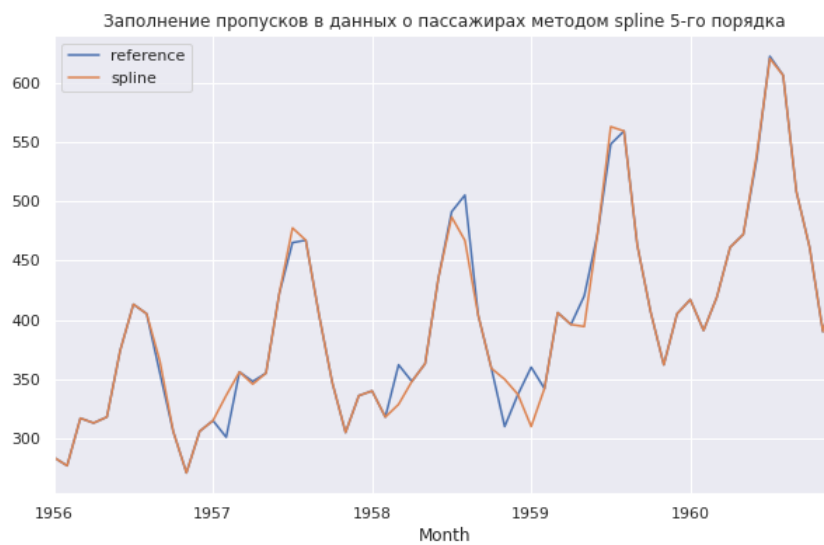
```
1 # и рождаемости
2 births_results = compare_methods(births)
3 births_results
```

	Method	RMSE
0	FillMean	3.55
1	FillMedian	3.65
2	RollingMedian	3.81
3	RollingMean	3.89

4	linear	4.04
5	polynomial	4.22
6	quadratic	4.22
7	cubic	4.22
8	FFill	4.3
9	BFill	4.39
10	spline	4.77

Выведем «лидеров» на графиках и сравним с соответствующими референтными значениями.

```
1 # выведем лидера по точности заполнения пропусков в данных о пассажирах
2 passengers[['reference', 'spline']].plot()
3 plt.title('Заполнение пропусков в данных о пассажирах методом spline 5-го порядка');
```



```
1 # сделаем то же самое для данных о рождаемости
2 births[['reference', 'FillMean']].plot()
3 plt.title('Заполнение пропусков в данных о рождаемости средним арифметическим');
```



В целом, как мы видим, более сложные методы интерполяции хорошо работают на нестационарных данных с относительно большим шагом временного ряда. При этом в

стационарных временных рядах с меньшим шагом между периодами их эффективность заметно ниже.

## Подведем итог

Сегодня мы рассмотрели несколько способов заполнения пропусков во временных рядах. В частности, мы изучили заполнение пропущенных значений с помощью среднего арифметического, медианы, предыдущего и последующего значений, скользящего среднего и медианы. Кроме того, был рассмотрен метод линейной, полиномиальной и сплайн интерполяции.

Выбор конкретного способа интерполяции во многом зависит от стационарности и шага временного ряда.

---