Лабораторная работа № 6

Изучение особенностей работы с временными рядами

Цель работы: Изучить особенности обработки временных рядов с помощью библиотек Pandas и SciPy. Визуализация с помощью библиотекиSeaborn.

Содержание работы:

1 Постановка задачи

Временные ряды – это важная форма структурированных данных, собираемых в самых разных областях науки таких, как финансы, экономика, экология, нейросети и физика. Любые измеряемые или наблюдаемые величины в произвольные последовательные интервалы времени могут быть представлены в виде временных рядов. Временные ряды могут быть как регулярными (данные собираются через определенные фиксированные интервалы времени) или нерегулярными (интервалы времени могут быть случайными).

То, как вы отмечаете и ссылаетесь на данные временных рядов, зависит от конкретной области применения и от одной из следующих позиций:

– Отметки времени, конкретные моменты времени;

– Фиксированный период, например, такой как месяц Январь 2017 года или целый 2018 год;

– Интервалы времени, отмеченные начало и концом отсчета. Периоды можно рассматривать как особые случаи интервалов;

– Эксперимент или прошедшее время; каждая временная метка является мерой времени относительно определенного времени начала. Например, диаметр печенья, выпекаемого каждую секунду с момента помещения в духовку.

Библиотеки для работы с временными рядами

– Pandas: https://pandas.pydata.org/pandas-docs/stable/timeseries.html

– SciPy http://www.statsmodels.org/devel/tsa.html

Стационарные временные ряды

Временной ряд считается *стационарным*, если его статистические свойства, такие как среднее значение, дисперсия, остаются постоянными во времени:

– Постоянное среднее;

– Постоянная дисперсия;

– Автоковариация, не зависящая от времени.

Что делает временной ряд нестационарным? Существуют две основные причины нестационарности временного ряда:

1. Тренд – изменение среднего значения величины с течением времени. Например, на рисунке 1 показано, что в среднем количество пассажиров со временем растет.

2. Сезонность – это колебания, которые происходят в определенные периоды времени. Например, люди могут иметь тенденцию покупать автомобили в конкретный месяц из-за повышения заработной платы или наличия скидок.

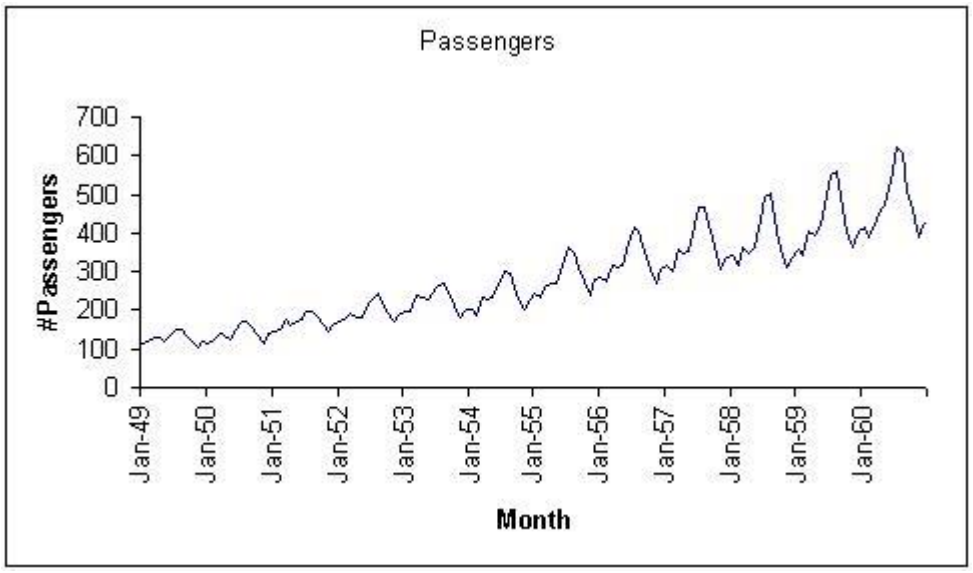

Рисунок 1 – Количество пассажиров, перевезенных за период с 1949 по 1960 годы

2 Анализ временных рядов Google Trends

Источник: https://www.datacamp.com/community/tutorials/time-series-analysis-tutorial

Скачайте данные Google Trends по ключевым словам, таким как «диета» и «спортзал», и посмотрите, как количество запросов меняется во времени, чтобы узнать о тенденциях и сезонности в данных временных рядов.

В данной лабораторной работе вы шаг за шагом ознакомитесь со способами работы с временными рядами.

Лабораторная работа не предполагает выполнения большого количества математических расчетов. Вы будете делать следующее:

– читать данные;
– перекодируете данные;
– проведете анализ данных.

Чтение данных

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Plot appears on its own windows
%matplotlib inline
# Tools / Preferences / Ipython / Console / Graphics / Backend: “automatic”
# Interactive Matplotlib Jupyter Notebook
# %matplotlib inline
try:
    url = "https://raw.githubusercontent.com/datacamp/datacamp_facebook_live_ny_resolution/master/datasets/multiTimeline.csv"
    df = pd.read_csv(url, skiprows=2)
except:
    df = pd.read_csv("../datasets/multiTimeline.csv", skiprows=2)
print(df.head())
# Rename columns
df.columns = ['month', 'diet', 'gym', 'finance']
# Describe
print(df.describe())
```

Запись данных

Теперь превратите столбец «месяц» в тип данных DateTime и сделаете его индексом DataFrame. Обратите внимание, что вы делаете это, потому что в результате работы метода .info() столбец «месяц» на самом деле является объектом типа Данные. Этот общий тип Данные инкапсулирует все, от строк до целых чисел и т.д. Это не совсем то, что нам нужно, когда нам понадобится просматривать данные временных рядов. Вот почему мы используем .to_datetime() для преобразования столбца «месяц» из нашего DataFrame в DateTime.

Будьте осторожны! Обязательно включайте аргумент inplace, когда вы устанавливаете индекс DataFrame df, чтобы фактически изменить исходный индекс и установить его в столбце «месяц».

```
df.month = pd.to_datetime(df.month)
df.set_index('month', inplace=True)
print(df.head())
```

Исследовательский анализ данных

Можно использовать встроенный метод визуализации Pandas .plot(), чтобы визуализировать данные в виде 3 линейных графиков на одном графике (по одному для каждого столбца, а именно: «диета», «спортзал» и «финансы»).

```
df.plot()
plt.xlabel('Year');
# change figure parameters
# df.plot(figsize=(20,10), linewidth=5, fontsize=20)
# Plot single column
# df[['diet']].plot(figsize=(20,10), linewidth=5, fontsize=20)
# plt.xlabel('Year', fontsize=20);
```

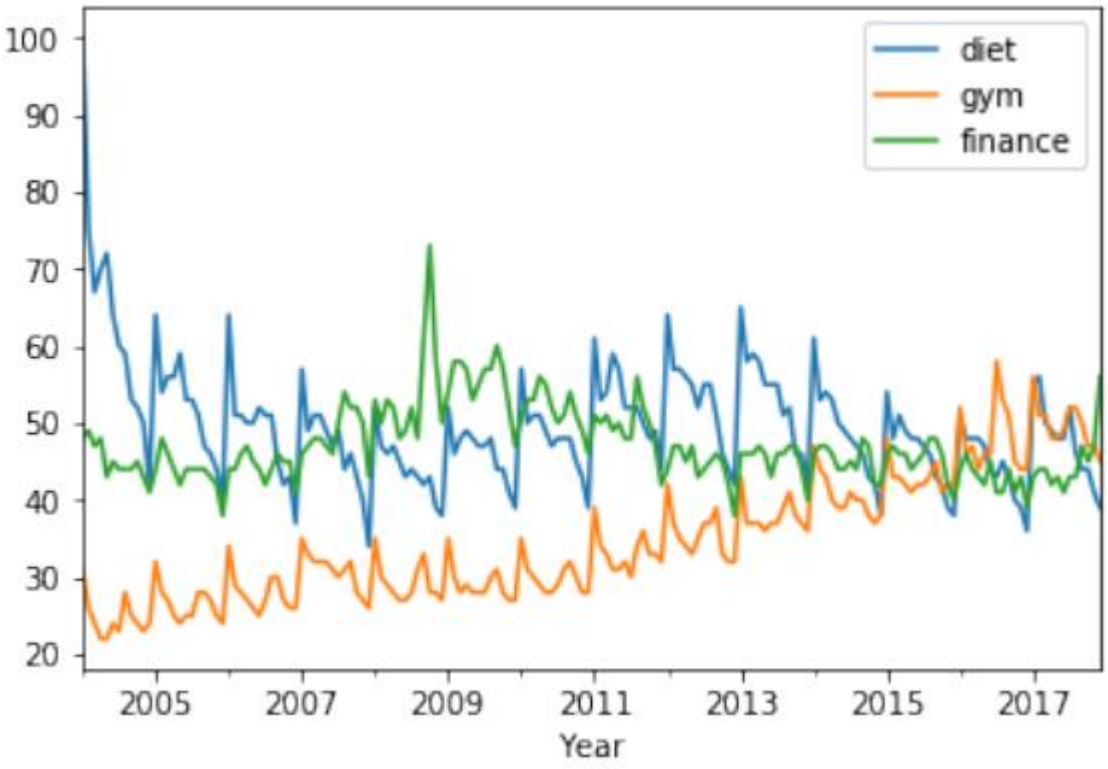

Рисунок 2 – Результаты визуализации

Обратите внимание, что эти данные являются относительными. Как вы можете прочитать о Google Trends: числа представляют поисковый интерес относительно самой высокой точки на графике для данного региона и времени. Значение 100 является пиковой популярностью для данного термина. Значение 50 означает, что термин в два раза менее

популярен. Аналогичным образом, оценка 0 означает, что термином интересовались менее 1% по сравнению с пиковой характеристикой.

Передискретизация, сглаживание, управление окнами, скользящее среднее: Тренды

Для расчета скользящего среднего, для каждой временной точки возьмите среднее от точек по обе стороны от рассматриваемой точки. Обратите внимание, что количество точек определяется размером окна.

Удалим Сезонность с помощью Pandas Series.

Дополнительные материалы можно найти по ссылке: http://pandas.pydata.org/pandas-docs/stable/timeseries.html

```
diet = df['diet']
diet_resamp_yr = diet.resample('A').mean()
diet_roll_yr = diet.rolling(12).mean()
ax = diet.plot(alpha=0.5, style='-') # store axis (ax) for latter plots
diet_resamp_yr.plot(style=':', label='Resample at year frequency', ax=ax)
diet_roll_yr.plot(style='--', label='Rolling average (smooth), window size=12', ax=ax)
ax.legend()
```

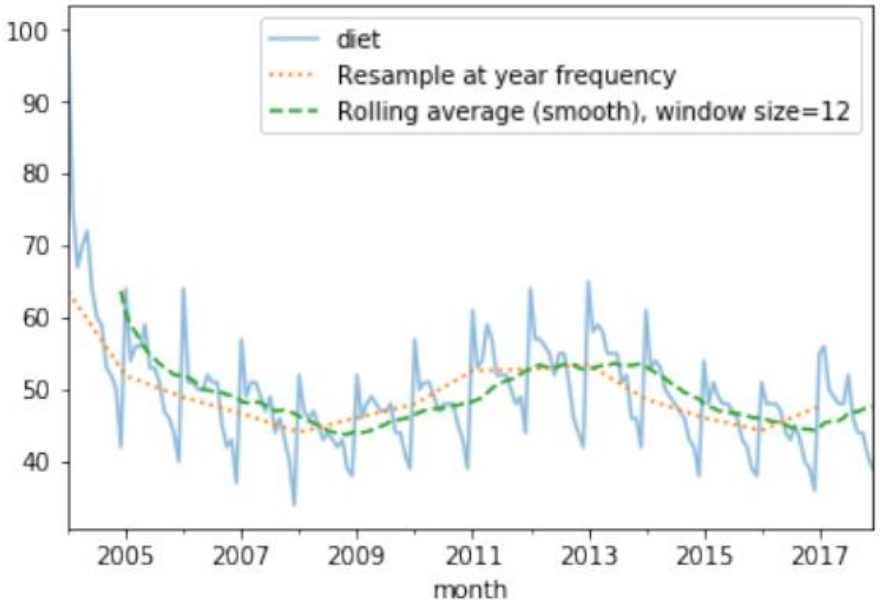

Рисунок 3 – Сравнение исходного ряда с результатами передискретизации и выполнения сглаживания.

Скользящее средне (сглаживание) с использованием Numpy

```
x = np.asarray(df[['diet']])
win = 12
win_half = int(win / 2)
# print([[(idx-win_half), (idx+win_half)) for idx in np.arange(win_half, len(x))]])
```

```
diet_smooth = np.array([x[(idx-win_half):(idx+win_half)].mean() for idx in np.arange(win_half, len(x))])
plt.plot(diet_smooth)
```

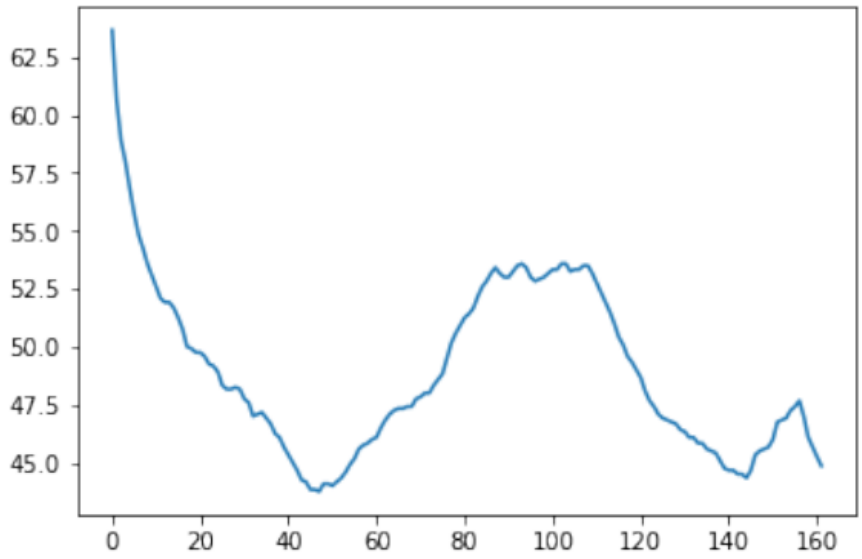

Рисунок 4 – Выполнение сглаживания с использованием NumPy

Изображение линий тренда для запросов “diet” и “gym”

Создайте новый DataFrame, который представляет собой объединение данных по “diet” и сглаженных данных по “gym”.

```
gym = df['gym']
df_avg = pd.concat([diet.rolling(12).mean(), gym.rolling(12).mean()], axis=1)
df_avg.plot()
plt.xlabel('Year')
```

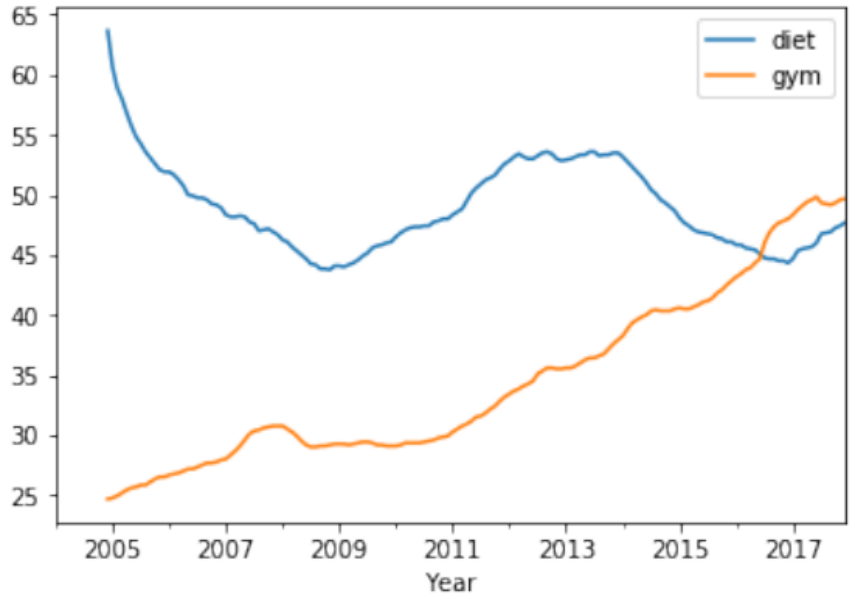

Рисунок 5

Вычитание тренда

```
df_dtrend = df[["diet", "gym"]] - df_avg
df_dtrend.plot()
```

```
plt.xlabel('Year')
```

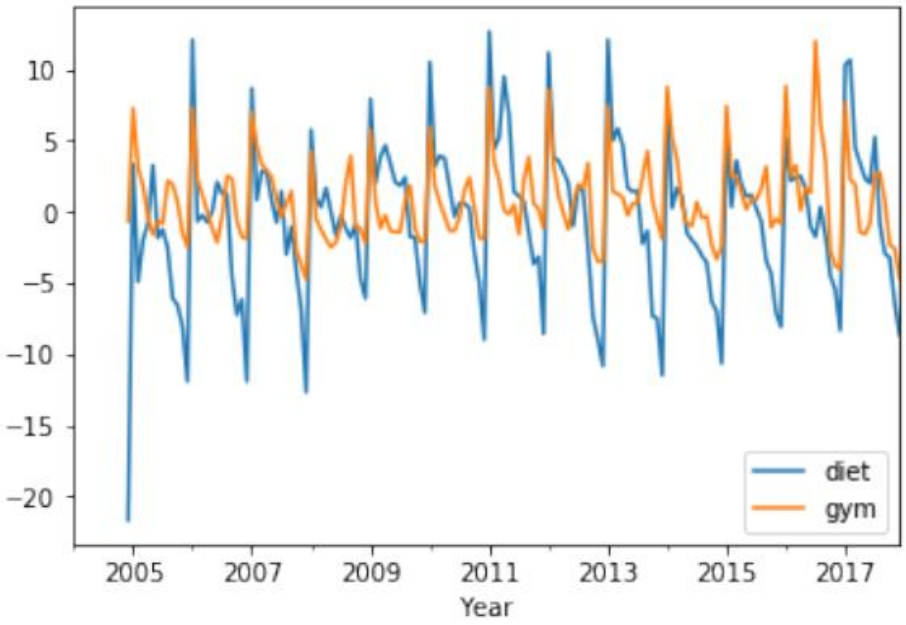

Рисунок 6

Разница первого порядка: сезонные модели

```
# diff = original - shiftted data
# (exclude first term for some implementation details)
assert np.all((diet.diff() == diet - diet.shift())[1:])
df.diff().plot()
plt.xlabel('Year')
```

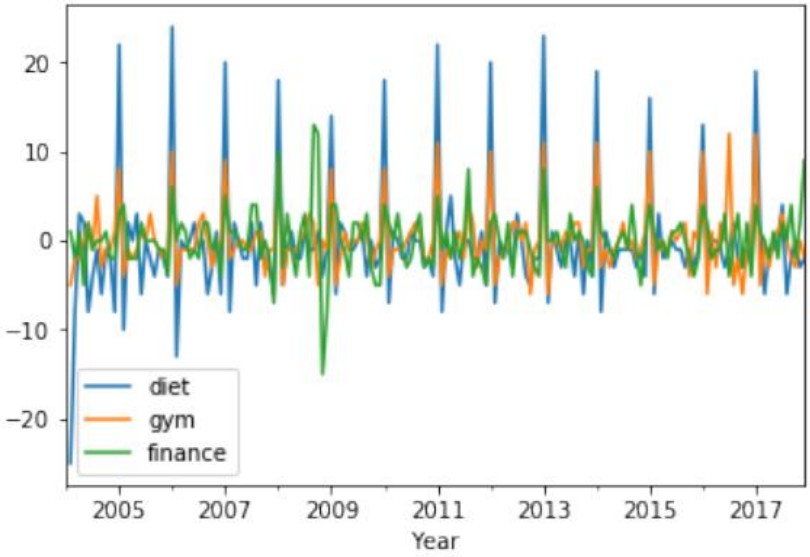

Рисунок 7

Периодичность и корреляция

```
df.plot()
plt.xlabel('Year');
print(df.corr())
```

```
             diet       gym   finance
diet     1.000000 -0.100764 -0.034639
gym     -0.100764  1.000000 -0.284279
finance -0.034639 -0.284279  1.000000
```

```
sns.heatmap(df.corr(), cmap="coolwarm")
```

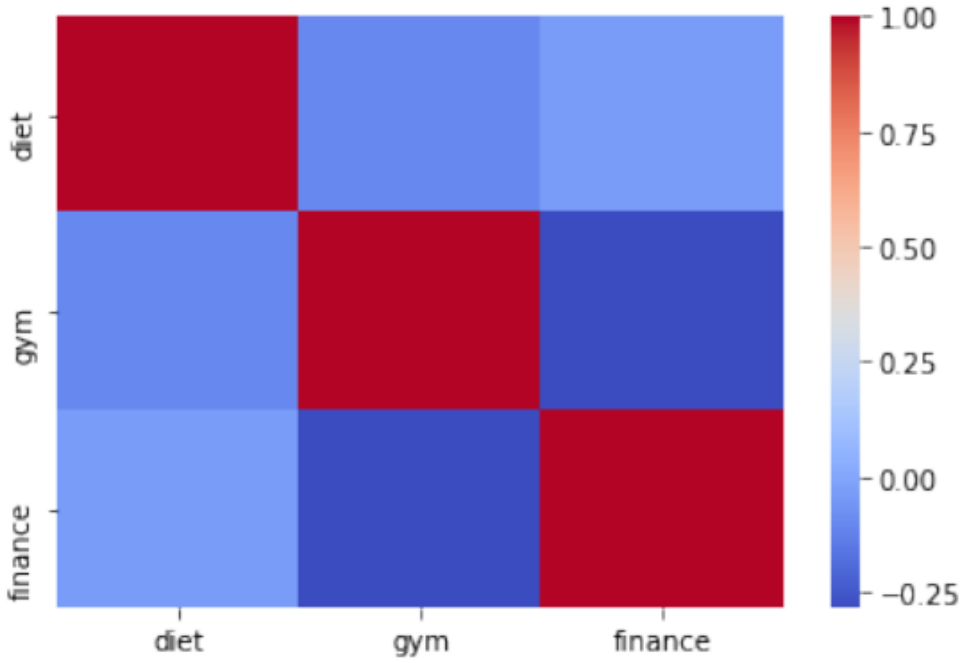

Рисунок 8 – Корреляционная матрица

«diet» и «gym» имеют отрицательную корреляцию! Следует помнить, что у вас есть сезонный и трендовый компонент. Отрицательный коэффициент корреляции «diet» и «gym» означает:

– трендовые компоненты имеют отрицательную корреляцию.

– сезонные компоненты положительно коррелируют

Фактический коэффициент корреляции захватывает оба термина.

Сезонная корреляция: корреляция разностей первого порядка этих временных рядов:

```
df.diff().plot()
plt.xlabel('Year');
print(df.diff().corr())
```

```
             diet       gym   finance
diet     1.000000  0.758707  0.373828
gym      0.758707  1.000000  0.301111
finance  0.373828  0.301111  1.000000
```

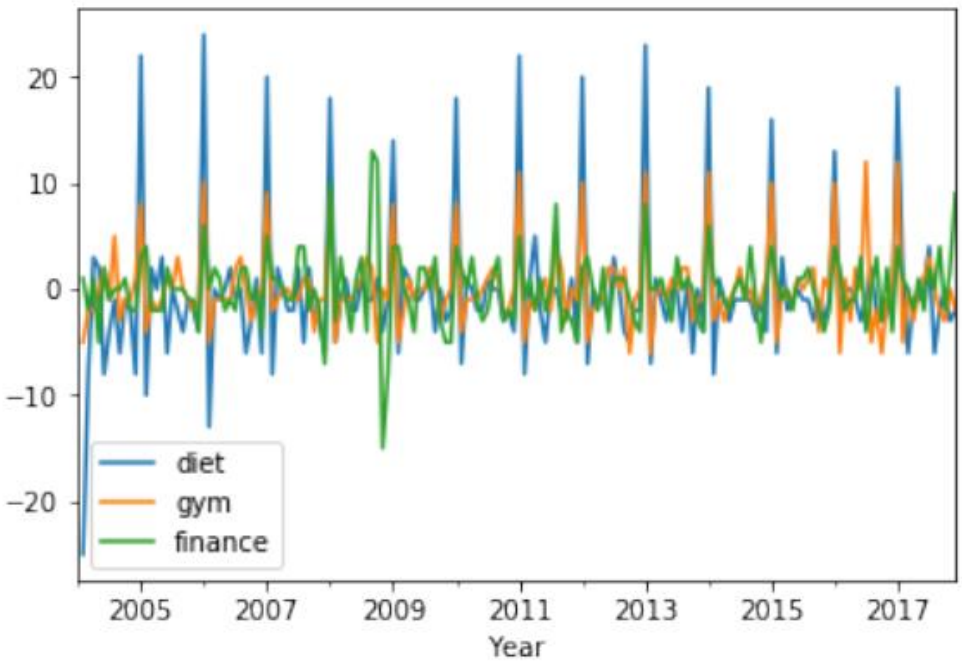

Рисунок 9

Построим матрицу корреляции

```
sns.heatmap(df.diff().corr(), cmap="coolwarm")
```

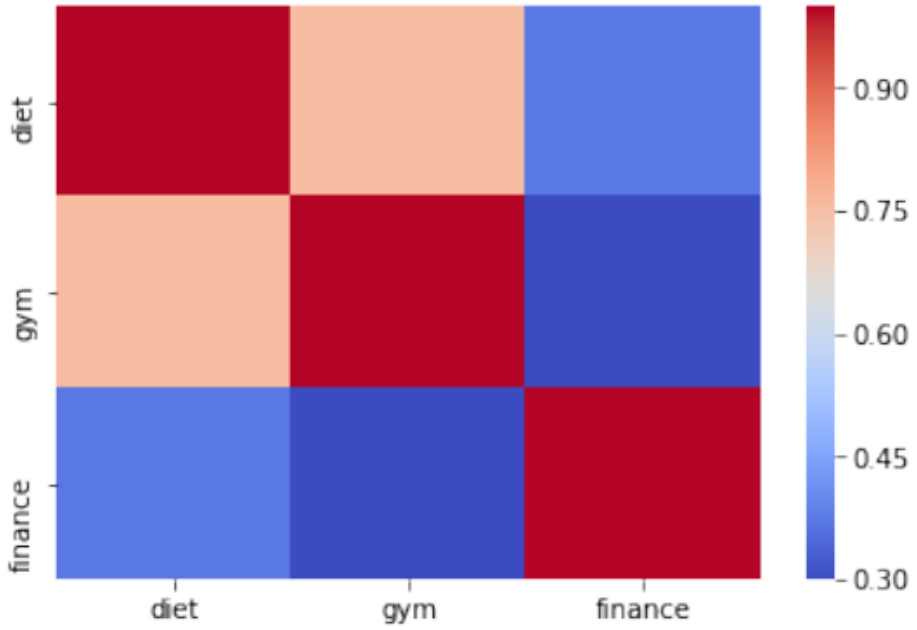

Рисунок 10

Разложение временных рядов по тренду, сезонности и остаткам

```
from statsmodels.tsa.seasonal import seasonal_decompose
x = gym
x = x.astype(float) # force float
decomposition = seasonal_decompose(x)
trend = decomposition.trend
```

```
seasonal = decomposition.seasonal
residual = decomposition.resid
plt.subplot(411)
plt.plot(x, label='Original')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(trend, label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(seasonal,label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(residual, label='Residuals')
plt.legend(loc='best')
plt.tight_layout()
```

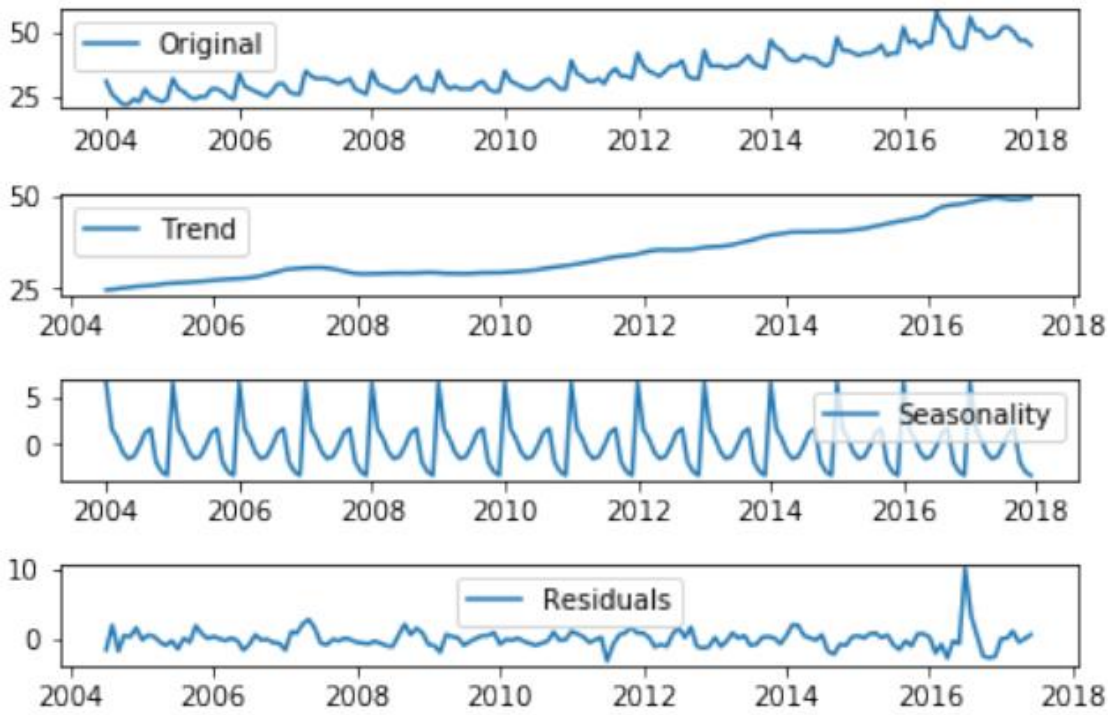

Рисунок 11

Автокорреляция

Временной ряд является периодическим, если он повторяется через равные интервалы, скажем, каждые 12 месяцев.

Функция автокорреляции (ACF): это мера корреляции между временным рядом и его запаздывающей компонентой. Например, при лаге 5 ACF сравнивает ряды в момент времени t1. … t2 с рядом в моменты t1-5, ..., t2-5, в котором t1-5 и t2-5 являются конечными точками.

```
# from pandas.plotting import autocorrelation_plot
from pandas.plotting import autocorrelation_plot
x = df["diet"].astype(float)
autocorrelation_plot(x)
```

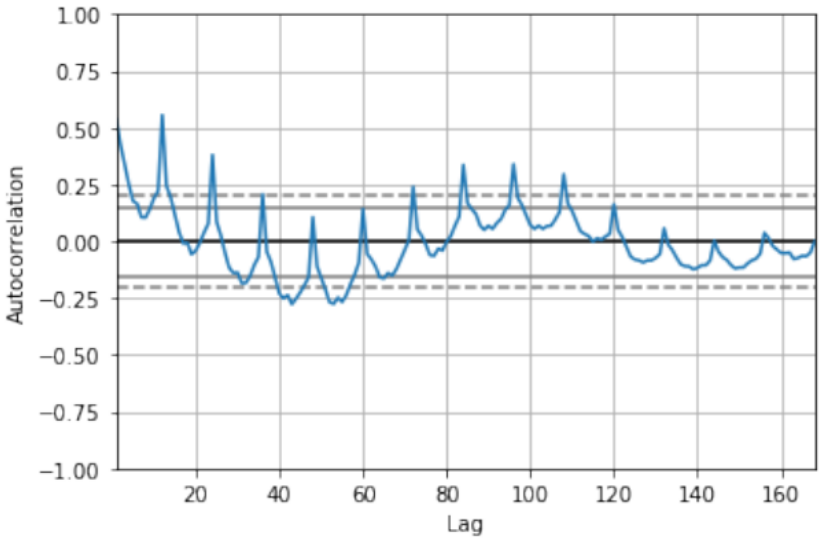

Рисунок 12

Вычислим функцию автокорреляции (ACF)

```
from statsmodels.tsa.stattools import acf
x_diff = x.diff().dropna() # first item is NA
lag_acf = acf(x_diff, nlags=36)
plt.plot(lag_acf)
plt.title('Autocorrelation Function')
```

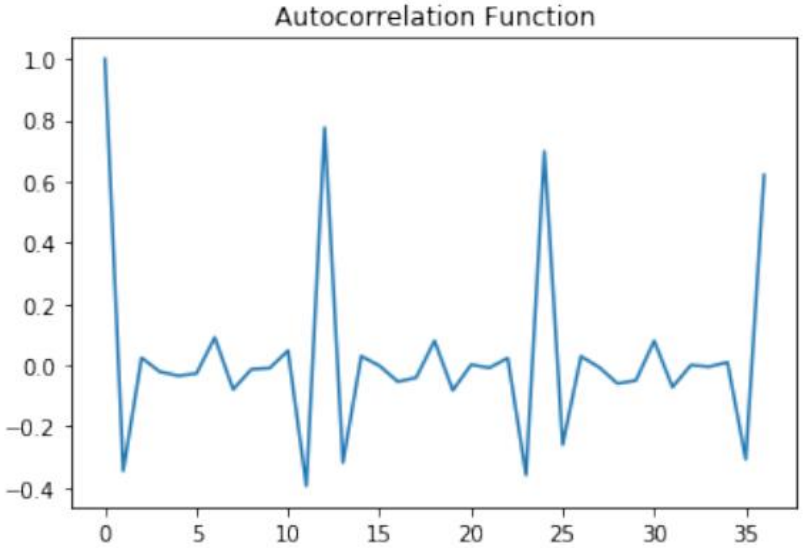

Рисунок 13

ACF достигает максимума каждые 12 месяцев: временной ряд коррелирует с самим собой, смещенным на 12 месяцев.

3 Прогнозирование временных рядов инструментами Python с использованием моделей авторегрессионного скользящего среднего (ARMA)

Источники:

– https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781783553358/7/ch07lvl1sec77/arma-models

– http://en.wikipedia.org/wiki/Autoregressive%E2%80%93moving-average_model

– ARIMA: https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/

Модели ARMA часто используются для прогнозирования временных рядов. Эти модели объединяют модели авторегрессии и скользящего среднего. В моделях скользящего среднего мы предполагаем, что переменная является суммой среднего временного ряда и линейной комбинации компонентов шума.

Модели авторегрессии и скользящего среднего могут иметь разные порядки. В общем, мы можем определить модель ARMA с p слагаемыми авторегрессии и q слагаемыми скользящего среднего следующим образом:

$$x_t = \sum_{i}^{p} a_i x_{t-i} + \sum_{i}^{q} b_i \varepsilon_{t-i} + \varepsilon_t$$

Выбор p и q

Построим частичные автокорреляционные функции для оценки p, а также использовать автокорреляционные функции для оценки q.

Функция частичной автокорреляции (PACF): измеряет корреляцию между временным рядом и своей запаздывающей версией, но после устранения изменений, уже объясненных промежуточными этапами сравнения. Например, при лаге 5 она проверит корреляцию, но удалит эффекты, уже объясненные лагами 1-4.

```
from statsmodels.tsa.stattools import acf, pacf
x = df["gym"].astype(float)
x_diff = x.diff().dropna() # first item is NA
# ACF and PACF plots:
lag_acf = acf(x_diff, nlags=20)
lag_pacf = pacf(x_diff, nlags=20, method='ols')
#Plot ACF:
plt.subplot(121)
plt.plot(lag_acf)
plt.axhline(y=0,linestyle='--',color='gray')
```

```
plt.axhline(y=-1.96/np.sqrt(len(x_diff)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(x_diff)),linestyle='--',color='gray')
plt.title('Autocorrelation Function (q=1)')
#Plot PACF:
plt.subplot(122)
plt.plot(lag_pacf)
plt.axhline(y=0,linestyle='--',color='gray')
plt.axhline(y=-1.96/np.sqrt(len(x_diff)),linestyle='--',color='gray')
plt.axhline(y=1.96/np.sqrt(len(x_diff)),linestyle='--',color='gray')
plt.title('Partial Autocorrelation Function (p=1)')
plt.tight_layout()
```

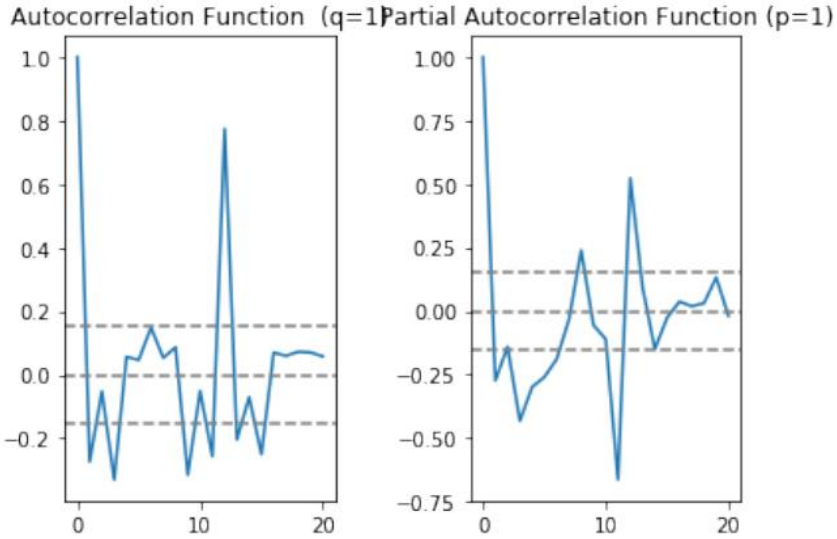

Рисунок 14

На этом графике две пунктирные линии по обе стороны от 0 являются доверительными интервалами. Их можно использовать для определения значений p и q следующим образом:

– p: значение задержки, когда диаграмма PACF впервые пересекает верхний доверительный интервал, в данном случае $p = 1$.

– q: значение задержки, когда диаграмма ACF впервые пересекает верхний доверительный интервал, в данном случае $q = 1$.

Подгонка модели ARMA с помощью стат. моделей

1. Определите модель, вызвав ARMA() и передав параметры p и q.
2. Модель подготавливается на основе тренировочных данных путем вызова функции fit().
3. Предсказания могут быть сделаны путем вызова функции predict() и указания либо индекса времени, либо времени, которое должно быть предсказано.

```
from statsmodels.tsa.arima_model import ARMA
```

```
model = ARMA(x, order=(1, 1)).fit() # fit model
print(model.summary())
plt.plot(x)
plt.plot(model.predict(), color='red')
plt.title('RSS: %.4f'% sum((model.fittedvalues-x)**2))
```

ARMA Model Results

Dep. Variable:	gym	No. Observations:	168
Model:	ARMA(1, 1)	Log Likelihood	-436.852
Method:	css-mle	S.D. of innovations	3.229
Date:	Thu, 16 May 2019	AIC	881.704
Time:	20:15:20	BIC	894.200
Sample:	01-01-2004 - 12-01-2017	HQIC	886.776

	coef	std err	z	P>|z|	[0.025	0.975]
const	36.4315	8.827	4.127	0.000	19.131	53.732
ar.L1.gym	0.9967	0.005	220.566	0.000	0.988	1.006
ma.L1.gym	-0.7494	0.054	-13.931	0.000	-0.855	-0.644

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.0033	+0.0000j	1.0033	0.0000
MA.1	1.3344	+0.0000j	1.3344	0.0000

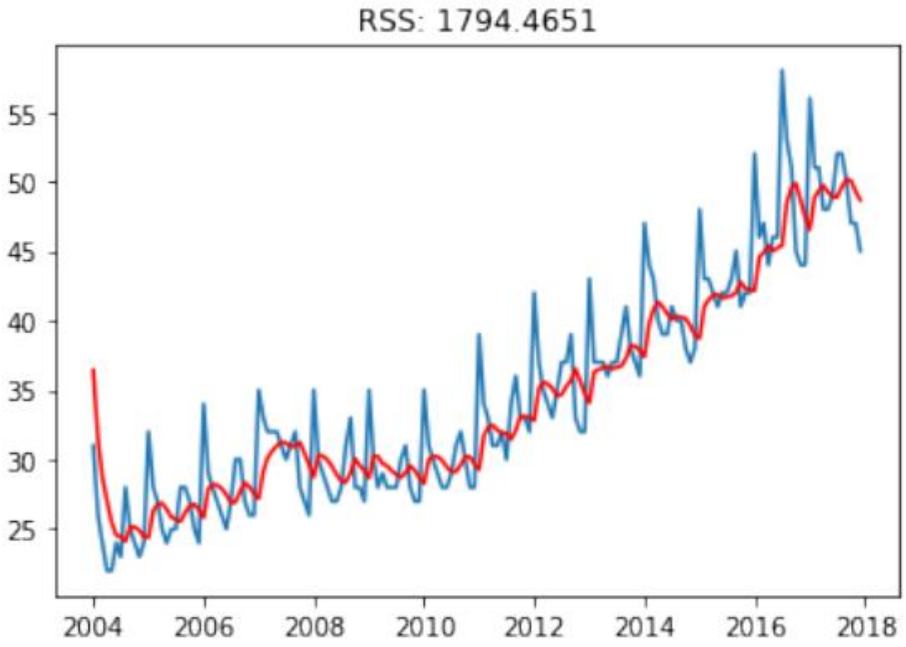

Рисунок 15

4 Ход работы

1. Внимательно прочитайте текст лабораторной работы и составьте конспект.
2. Проделайте примеры, показанные в тексте работы и сохраните результаты.
3. В качестве индивидуального задания выполните анализ временных рядов для результатов измерения температуры https://datahub.io/core/global-temp#data и законспектируйте результаты.
4. Представьте преподавателю скрипты Python и отчет о выполнении работы.