Time Series Analysis lab

Игорь Рухович. 22МАГИАД

Topic: Christmas

Imports:

In [ ]:
```
%matplotlib inline
from matplotlib import pyplot as plt
import warnings
import pandas as pd
from scipy import stats
import statsmodels.api as sm
from itertools import product
from tqdm import tqdm
from dateutil.relativedelta import relativedelta

warnings.filterwarnings('ignore')
plt.rcParams['figure.figsize'] = 12, 10
```

In [ ]:
```
data_folder = "../data/"
data_fname = "data.csv"

my_name = "Рухович Игорь Владимирович"
my_topic = "Christmas"
year_period = 365
week_period = 7
forecast_period = 14
```

Data lookup

In [ ]:
```
df = pd.read_csv(data_folder + data_fname)
df
```

Out[ ]:

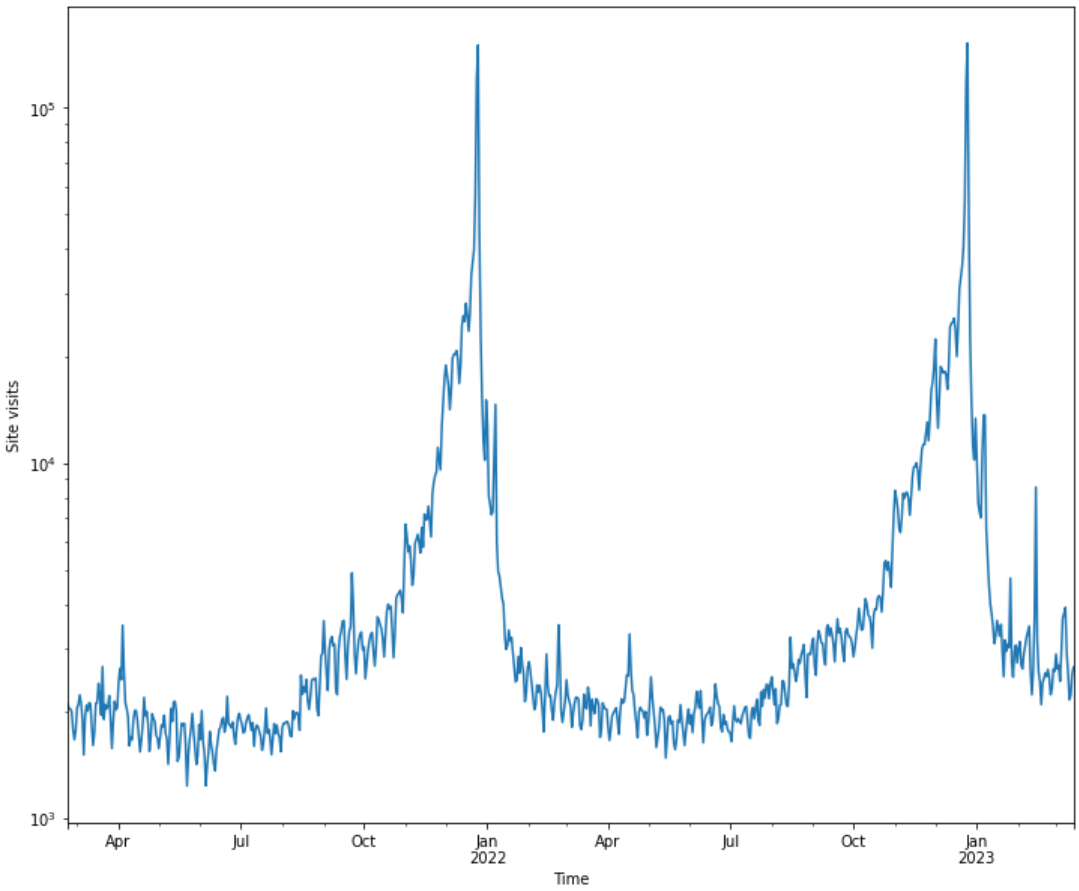
	ФИО	Title	2021-02-22	2021-02-23	2021-02-24	2021-02-25	2021-02-26	2021-02-27	2021-02-28	2021-03-01	...	2023-03-05	2023-03-06	202 03-
0	Тронин Дмитрий Валерьевич	Тригонометрические функции	945	983	1178	1236	1008	790	1113	1296	...	846	1030	7
1	Воробьев Дмитрий Максимович	Логарифм	714	704	886	830	786	712	825	865	...	613	734	5
2	Пьянзин Артём Алексеевич	Экспонента	382	392	499	532	446	421	442	525	...	339	479	4
3	Абрамова Полина Александровна	Гиперболические функции	457	471	573	616	522	426	571	593	...	395	462	3
4	NaN	Обратные тригонометрические функции	361	442	492	436	350	371	413	502	...	361	412	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
71	NaN	March 11	285	329	346	434	338	311	421	733	...	567	845	8
72	NaN	March 10	258	268	300	352	304	266	312	646	...	768	1154	14
73	Поляков Валерий Игоревич	March 15	258	314	320	339	301	247	362	643	...	513	699	6
74	NaN	October 1	92	92	73	50	98	89	85	87	...	66	81	
75	NaN	September 21	92	77	102	74	83	112	70	97	...	57	94	

76 rows × 753 columns

In [ ]:
```
christmas_df = pd.Series(df.loc[df["ФИО"] == my_name, "2021-02-22":].values.squeeze(),
                         index = pd.to_datetime(df.columns[2:]),
```

```
                            name = my_topic)
christmas_df.plot()
plt.yscale("log")
plt.ylabel("Site visits")
plt.xlabel("Time")
```

Out[ ]: Text(0.5, 0, 'Time')



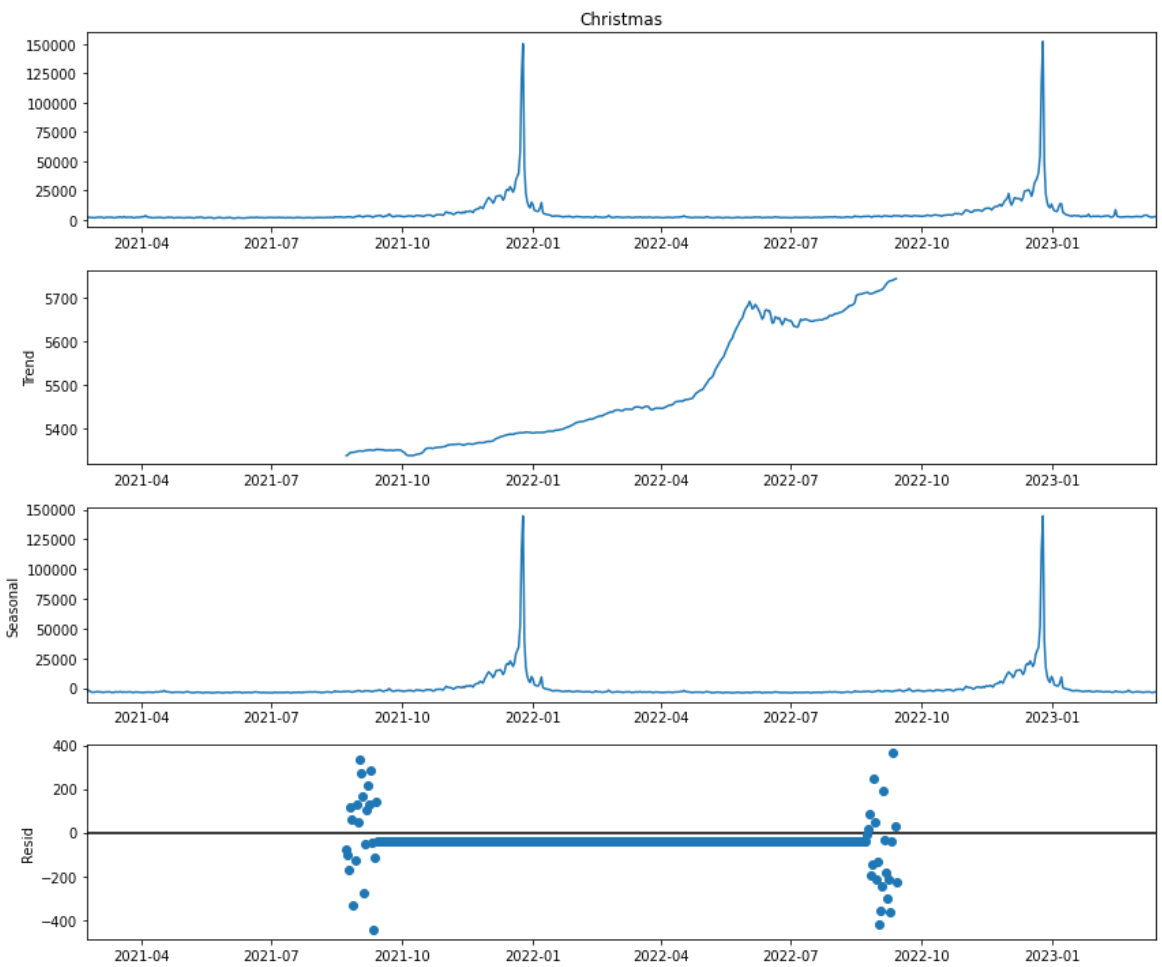
Видим явную сезонность: тема "рождество" интересует пользователей Википедии во время самого праздника и менее интересна летом.

Decomposition

Попробуем провести декомпозицию:

In [ ]:
```
sm.tsa.seasonal_decompose(christmas_df, period=year_period).plot()
print(f"Dickey-Fuller criterion: p={sm.tsa.stattools.adfuller(christmas_df)[1]:.6f}")
```

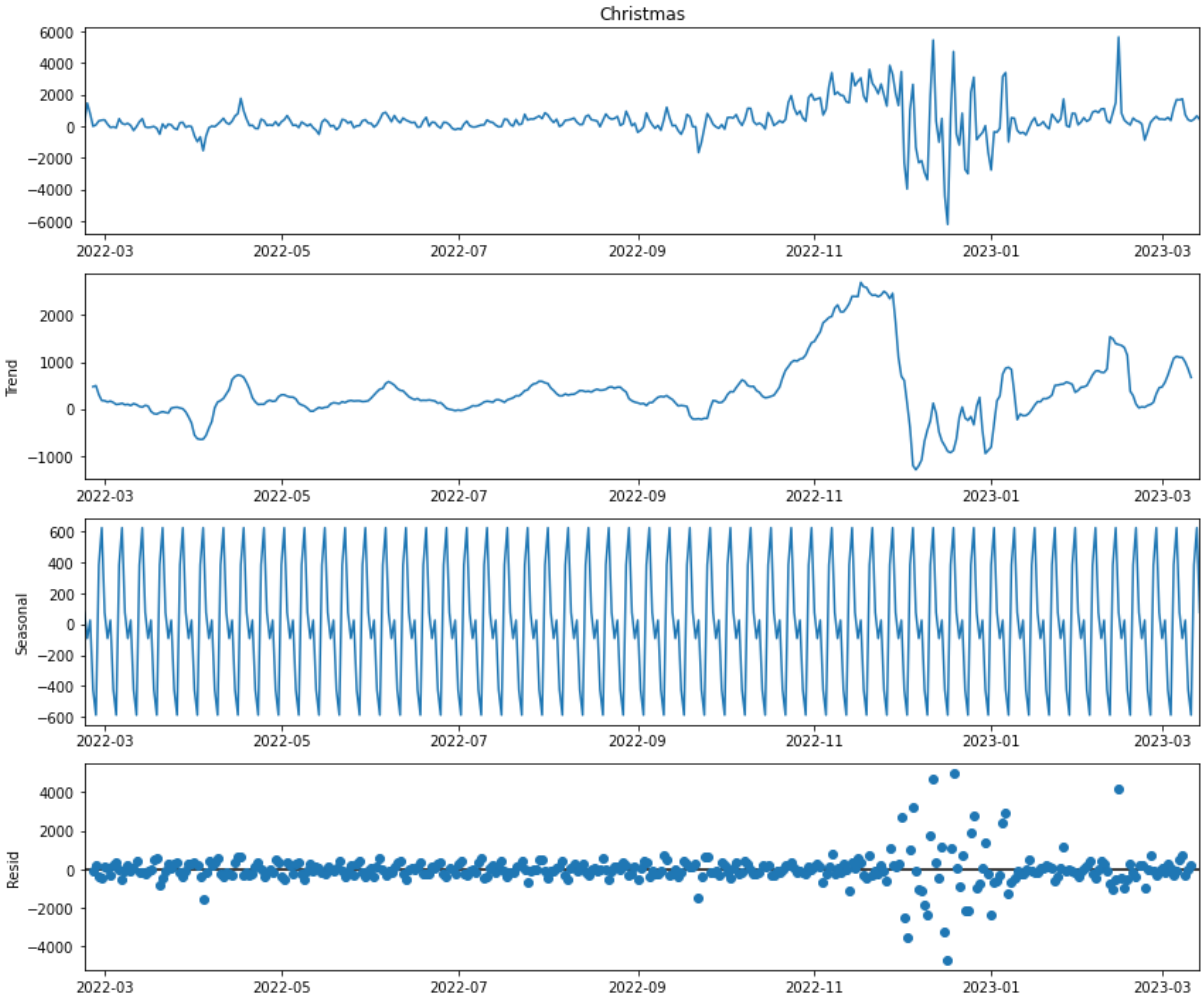
```
Dickey-Fuller criterion: p=0.000000
```



Критерий Dickey-Fuller отвергает гипотезу о нестационарности выборки, но глазами мы видим выраженную сезонность. Проведём дифференцирование:

In [ ]:
```
df_shifted = christmas_df - christmas_df.shift(year_period)
sm.tsa.seasonal_decompose(df_shifted[year_period:], period=week_period).plot()
print(f"Dickey-Fuller criterion: p={sm.tsa.stattools.adfuller(df_shifted[year_period:])[1]:.6f}")
```

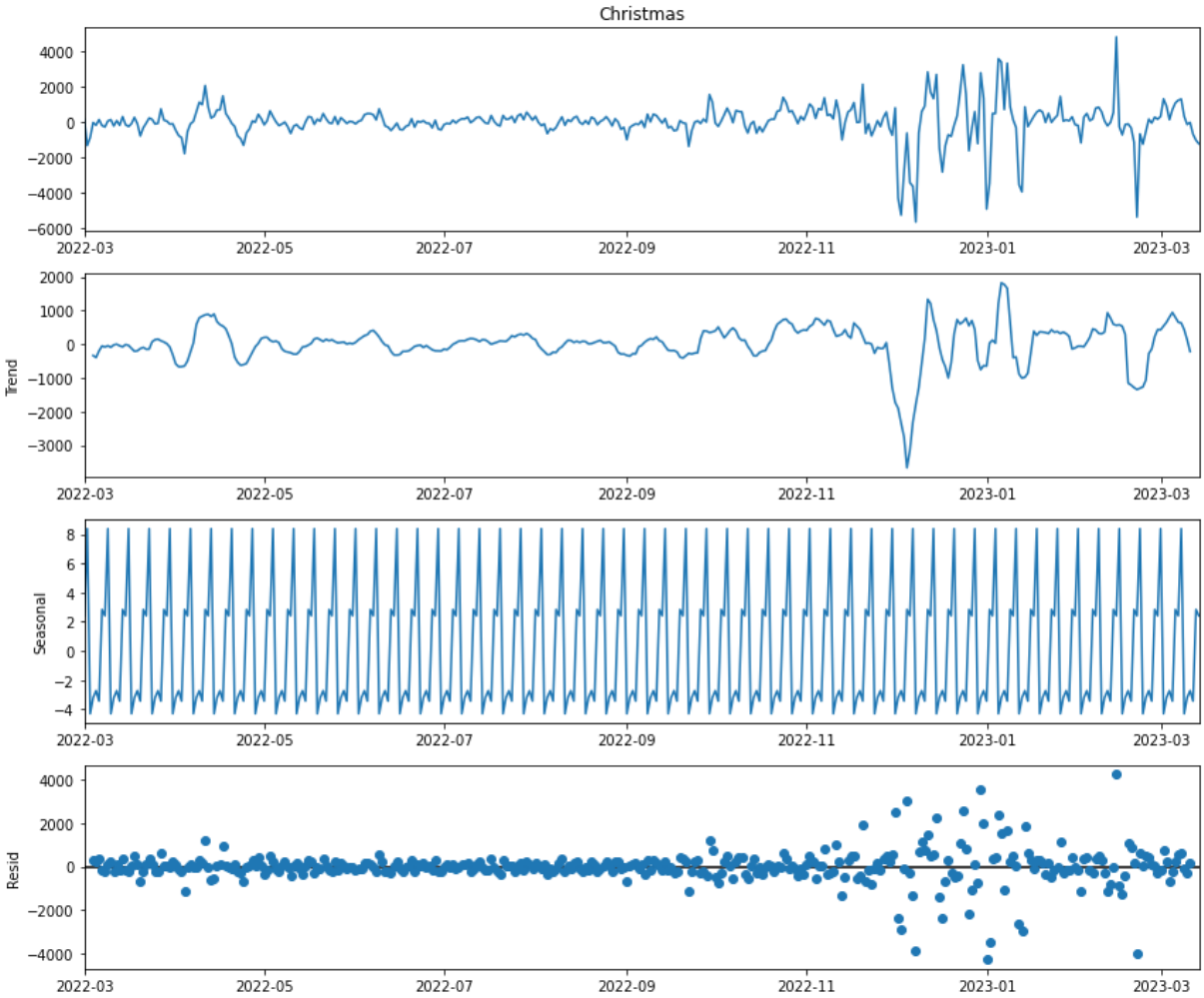
```
Dickey-Fuller criterion: p=0.051731
```



Для годового периода теперь недостаточно данных: необходимо 2 полных цикла, зато теперь видим недельную сезонность. Тест теперь едва ли отвергает гипотезу о нестационарности выборки, проведём повторное дифференцирование с периодом 7 дней:

In [ ]:
```
df_shifted_twice = df_shifted - df_shifted.shift(week_period)
sm.tsa.seasonal_decompose(df_shifted_twice[year_period + week_period:], period=week_period).plot()
print(f"Dickey-Fuller criterion: p={sm.tsa.stattools.adfuller(df_shifted_twice[year_period +\
                                                                week_period:])[1]:.6f}")
```

```
Dickey-Fuller criterion: p=0.000010
```



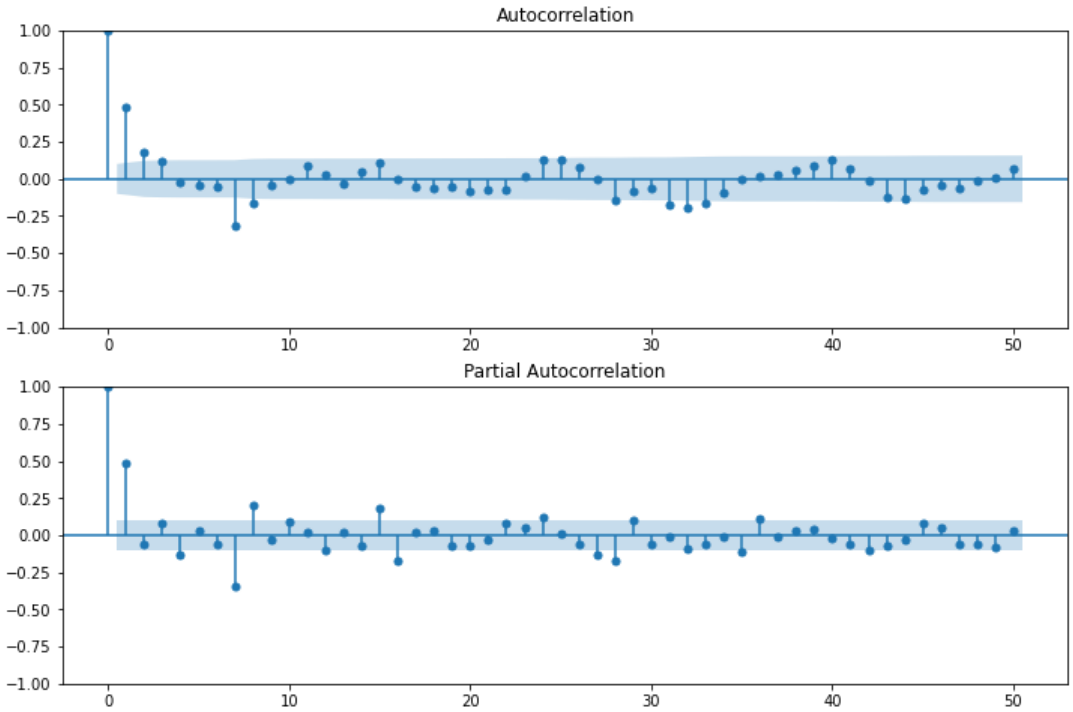
Стало намного лучше: не видим выраженной сезонности, но остаётся тренд - проведём в модели дополнительно одно дифференцирование по соседним значениям.

Дополнительно замечу, что поскольку в задании требуется предсказать поведение за 14 дней, где само рождество не будет встречаться, то мы можем (ввиду недостатка данных) не обращать внимания на такую сезонность, а работать только с недельной. 7x52=364, практически 365. То есть годовой сдвиг включает в себя множитель 7, а значит, если мы тренируем модель с недельным периодом, необходимо будет дифференцировать по сезонности дважды (за "год" и за "неделю").

Посмотрим также на автокорреляцию остатков:

In [ ]:
```
df_diff = df_shifted_twice[year_period + week_period:]

plt.figure(figsize=(12, 8))
ax = plt.subplot(211)
sm.graphics.tsa.plot_acf(df_diff.values.squeeze(),
                         lags=50, ax=ax)
ax = plt.subplot(212)
sm.graphics.tsa.plot_pacf(df_diff.values.squeeze(),
                          lags=50, ax=ax);
```



Видим значимую корреляцию с соседним значением. От сезонности мы практически избавились.

SARIMA

Попробуем перебрать разные значения гиперпараметров SARIMA:

In [ ]:
```
ps = range(0, 4)
d=1
qs = range(0, 1)
Ps = range(0, 3)
D=2
Qs = range(0, 3)

parameters = product(ps, qs, Ps, Qs)
parameters_list = list(parameters)
len(parameters_list)
```

Out[ ]: 36

In [ ]:
```
%%time
results = []
best_aic = float("inf")

warnings.filterwarnings('ignore')

for param in tqdm(parameters_list):
    #try except is needed because some parameter combinations are not valid
    try:
        model=sm.tsa.statespace.SARIMAX(christmas_df, order=(param[0], d, param[1]),
                                        seasonal_order=(param[2], D,
                                                        param[3], week_period)).fit(disp=-1)
    except ValueError:
        print('wrong parameters:', param)
        continue
    aic = model.aic
    # save best model, it's AIC and params
    if aic < best_aic:
        best_model = model
        best_aic = aic
        best_param = param
    results.append([param, model.aic])

warnings.filterwarnings('default')
```

```
100%|██████████| 36/36 [01:09<00:00,  1.94s/it]
CPU times: user 4min 30s, sys: 2min 9s, total: 6min 39s
Wall time: 1min 9s
```

Верхушка таблицы параметров лучших моделей (по AIC, правдоподобие минус число параметров):

In [ ]:
```
result_table = pd.DataFrame(results)
result_table.columns = ['parameters', 'aic']
print(result_table.sort_values(by = 'aic', ascending=True).head())
```

```
      parameters          aic
20  (2, 0, 0, 2)  15056.789496
23  (2, 0, 1, 2)  15057.936661
29  (3, 0, 0, 2)  15057.940854
32  (3, 0, 1, 2)  15059.246643
26  (2, 0, 2, 2)  15143.999774
```

В целом можно сказать, что число параметров по порядку сильно отличается от лосса, поэтому правильнее в такой метрике было бы брать эти значения с коэффициентами, но не будем спорить с учеными :)

In [ ]:
```
print(best_model.summary())
```

```
                                     SARIMAX Results                                      
==========================================================================================
Dep. Variable:                              Christmas   No. Observations:                  751
Model:             SARIMAX(2, 1, 0)x(0, 2, [1, 2], 7)   Log Likelihood               -7523.395
Date:                                Mon, 27 Mar 2023   AIC                          15056.789
Time:                                        02:40:55   BIC                          15079.796
Sample:                                  02-22-2021   HQIC                         15065.662
                                       - 03-14-2023                                         
Covariance Type:                                 opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.2068      0.014     14.805      0.000       0.179       0.234
ar.L2         -0.3679      0.008    -48.237      0.000      -0.383      -0.353
ma.S.L7       -1.9930      0.015   -131.269      0.000      -2.023      -1.963
ma.S.L14       0.9931      0.016     62.781      0.000       0.962       1.024
sigma2       3.93e+07   7.83e-10   5.02e+16      0.000    3.93e+07    3.93e+07
===================================================================================
Ljung-Box (L1) (Q):                   0.12   Jarque-Bera (JB):            451042.59
Prob(Q):                              0.73   Prob(JB):                         0.00
Heteroskedasticity (H):             680.04   Skew:                            -5.08
Prob(H) (two-sided):                  0.00   Kurtosis:                       123.85
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.17e+31. Standard errors ma
y be unstable.
```

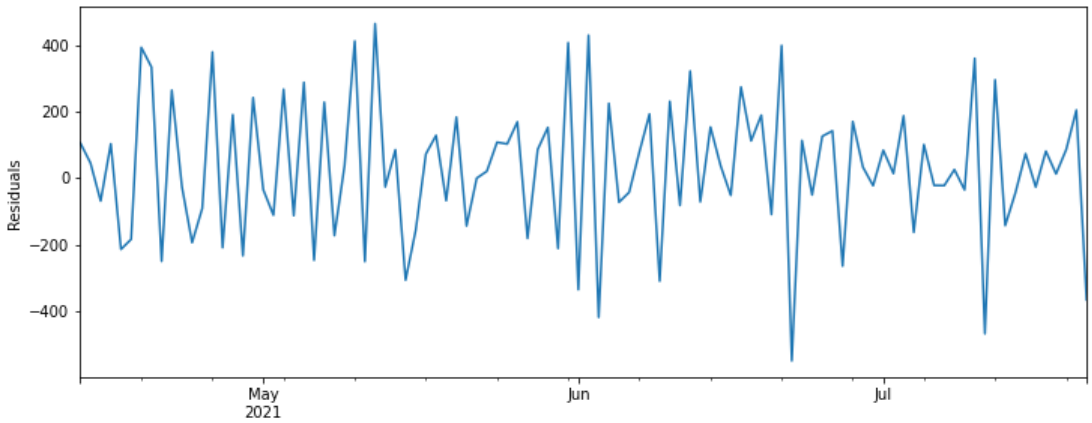
Посмотрим на участок, не включающий рождество:

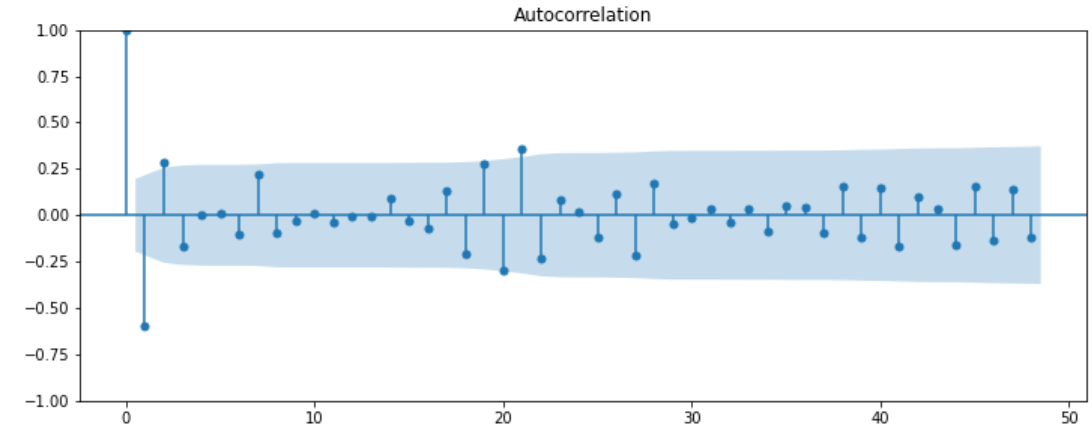
In [ ]:
```
plt.subplot(211)
best_model.resid[50:150].plot()
plt.ylabel(u'Residuals')

ax = plt.subplot(212)
sm.graphics.tsa.plot_acf(best_model.resid[50:150].values.squeeze(), lags=48, ax=ax)

print("Dickey-Fuller criterion: p=%f" % sm.tsa.stattools.adfuller(best_model.resid[15:])[1])
```

```
Dickey-Fuller criterion: p=0.000000
```





Если рассмотрим часть выборки, не включающую рождество, выглядит, что она стационарная. Это подтверждает критерий dickey-fuller

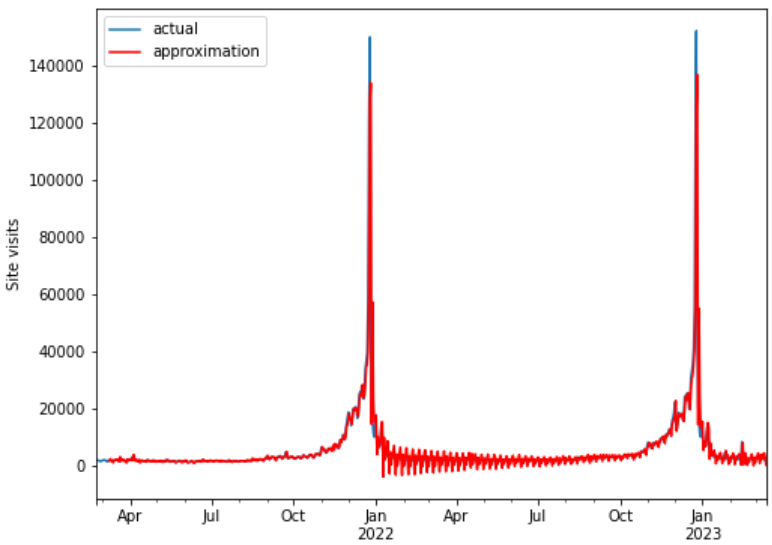
Напоминаю, что из-за недостатка данных по годам, не получается обучить SARIMA на годовых сезонных интервалах. По хорошему, в этой задаче следует взять больше данных (например, за 10 лет) и более мощную модель, позволяющую находить несколько сезонностей

Forecast & Outcome

Посмотрим на качество аппроксимации:

In [ ]:
```
plt.figure(figsize=(8, 6))
approx_vals = best_model.fittedvalues
christmas_df.plot(label='actual')
approx_vals[15:].plot(color='r', label='approximation')
plt.ylabel("Site visits")
plt.legend()
```

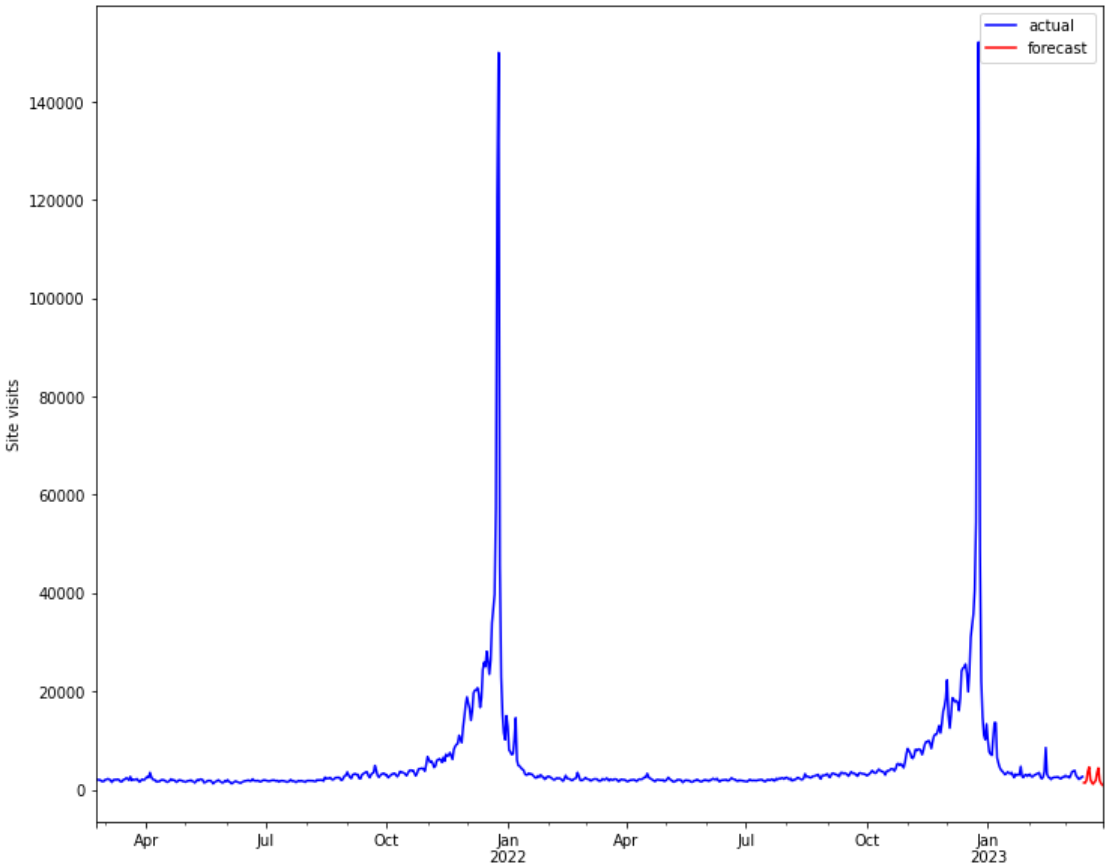
Out[ ]:
```
<matplotlib.legend.Legend at 0x7ff16b490940>
```



Видим, что модель довольно четко описала данные, хотя и имеется значительный разброс с января по май 2022. Попробуем произвести предсказание на 14 следующих дней:

In [ ]:
```
christmas_df.plot(color='b', label='actual')
best_model.predict(christmas_df.shape[0], christmas_df.shape[0] + forecast_period)\
          .plot(color='r', label='forecast')
plt.ylabel("Site visits")
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x7ff1988e7280>
```



Предсказания в сыром виде:

In [ ]:
```
best_model.predict(christmas_df.shape[0], christmas_df.shape[0] + forecast_period)
```

Out[ ]:
```
2023-03-15    1430.748978
2023-03-16    1400.046627
2023-03-17    1912.997049
2023-03-18    3694.400955
2023-03-19    4611.827742
2023-03-20    2052.321312
2023-03-21    1480.804943
2023-03-22    1150.353571
2023-03-23    1562.180912
2023-03-24    1830.194831
2023-03-25    3420.846543
2023-03-26    4407.531510
2023-03-27    1900.250334
2023-03-28    1323.093192
2023-03-29     955.410781
Freq: D, Name: predicted_mean, dtype: float64
```

Предсказания получились немного разбросанными. Полагаю, что это произошло из-за недостаточного объема исходных данных и слабой модели: в данных явно присутствует 2 сезонности, но для годовой мы имеем немногим больше 2 циклов, а после первого дифференцирования фактически теряем половину датасета.

Для более точных прогнозов хочется использовать модель с возможностью нахождения нескольких сезонностей.

In [ ]: