VT2024: IL2233 Lab 1
Time Series Visualization and Feature Extraction

Zhonghai Lu

March 25, 2024

1 Introduction

In statistics, Exploratory Data Analysis (EDA) is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods.

To analyze time-series data, the first step is usually a routine work of performing exploratory data analysis. This includes a number of visualization and statistics generation tasks such as visualizing the data sequence and exposing its statistical characteristics. Besides visualizing data structure and statistics of time-series sequences, one important aspect in exploratory data analysis is to identify and extract features from the time-series data.

To exercise the exploratory data analysis, we begin by looking into two special time series, namely, Gaussian white noise and Random walk, followed by realistic time series measurements.

One important pre-processing step in statistical time-series analysis is to test the stationary of the given time series. To this end, we can do visual inspection of the graphics generated from the time series. More rigorously, we can use hypothesis testing. One common statistical method used for stationary test is the Augmented Dickey-Fuller (ADF) test. If a time series is not stationary, it is often necessary to transform the series into a stationary one by the differencing operation in order to apply stationary time series modeling techniques.

Time series decomposition is a way giving insights into the structure of the time series data at hand, because time series data can exhibit a variety of patterns. It is often helpful to split a time series into several components, each representing an underlying pattern category, which can be visualized together. We typically consider a time series comprising three components: a trend component, a seasonal component, and a remainder component (containing anything else in the time series). For some time series (e.g., those that are observed at least daily), there can be more than one seasonal component, corresponding to the different seasonal periods. A season is a consistent period.

After investigating graphics-based exploratory data analysis, we experiment feature extraction from time series. This includes statistical summarative features, temporal correlation features, and spectral features, all of which are interesting to get a full understanding the structure and properties of the time-series data.

2 Purpose and Setup

The laboration intends to train students in performing exploratory data analysis and feature extraction. It has the following objectives:

- Understand the properties of special time-series, such as white noise and random walk.
- Perform necessary preprocessing of time series data such as checking stationary and randomness. In particular, use the simple yet effective differencing operation to turn a non-stationary series into a stationary series.
- Exploratory data analysis by visualizing time series data in different forms.
- Try time series decomposition using two different ways. One is classical seasonal decomposition based on moving average of the neighboring data points. The other is STL (Seasonal Trend decomposition using Loess).
- Basic statistical, time-domain, and frequency-domain feature extraction.

This lab uses the following experimental setup. The programming language is Python. The libraries to be used include pandas, matplotlib, statsmodels.

This lab is a group work. Each group consists of 2 students.

3 Tasks

To achieve the objectives, this lab comprises the following sequential tasks.

3.1 Task 1. Exploratory Data Analysis

3.1.1 Task 1.1 White noise series

A time series is white noise if the variables are independent and identically distributed (iid) with a mean of zero $\mu = 0$. This means that all variables have the same variance (σ^2) and there is no correlation across all values in the series. If the variables in the series are drawn from a Gaussian distribution, the series is called Gaussian white noise.

White noise is an important concept in time series analysis due to two reasons. (1) Predictability: If a time series is white noise, it is a sequence of random numbers and thus possesses no structure. Hence, it cannot be predicted. (2) Model Diagnostics or Validity: The remainder series (forecast errors) from a prediction model should ideally be white noise. Otherwise, it suggests still possibility to improve the predictive model.

Your task 1.1 is as follows.

1. Generate a white noise series with N data points (e.g. N can be 100, 1000, 5000, or 10000). Then find its actual mean, standard deviation, and draw its line plot, histogram, density plot, box plot, lag-1 plot, ACF and PACF graphs (lags up to 40).
2. Generate 100 random series with length 1000 data points, then use the average values at each time to produce an average value series. Then repeat the same process above.

3. Perform randomness test on the white noise series using the Ljung-Box test.

4. Perform stationarity test on the white noise series using the Augmented Dickey-Fuller (ADF) test.

The following hints may help you with the task.

In Python, you can use the "random" module to generate Gaussian white noise. You can create a list of N random Gaussian variables using the gauss() function from the random module (`https://docs.python.org/3/library/random.html`). Each variable has a Gaussian distribution with a mean (μ) of 0.0 and a standard deviation (σ) of 1.0. Once generated, you can wrap the list in a Pandas Series for convenience. An example of generating a white-noise series is shown below.

```
from random import gauss
from random import seed
from pandas import Series
from pandas.plotting import autocorrelation_plot

# seed the random number generator
seed(10)
# create white noise series
series = [gauss(0.0, 1.0) for i in range(1000)]
series = Series(series)
```

3.1.2 Task 1.2 Random-walk series

A random-walk series is a series of dependent values where its current value y_t equals to its previous value y_{t-1} plus a random variable x_t. If x_t is a binary process (with only two possible values, e.g, +1, -1), then y_t is called a simple symmetric random walk. Because of the dependence relationship, a random-walk sequence is different from a whit-noise series.

Your task 1.2 is as follows.

1. Realize a simple random walk series with N data points (e.g. N can be 100, 1000, 5000, or 10000) starting from an initial value of 0 ($y_0 = 0$), and $x_t = +1, -1$.

2. Then find its actual mean, standard deviation, and draw its line plot, histogram, density plot, box plot, lag-1 plot, ACF and PACF graphs (lags up to 40).

3. Perform randomness test on the random walk series using the Ljung-Box test.

4. Perform stationarity test on the random walk series using the Augmented Dickey-Fuller (ADF) test. Is the series stationary? If not, which operation can be applied to make it stationary?

Answer the following questions:

- What methods can be used to check if a series is random? Describe both visualization and statistic test methods.

- What methods can be used to check if a series is stationary? Describe both visualization and statistic test methods.

- Why is white noise important for time-series prediction?
- What is the difference between a white noise series and a random walk series?
- Is it possible to change a random walk series into a series without correlation across its values ? If so, how? Explain also why it can.

3.1.3 Task 1.3 Global land temperature anomalies series

In this task, you use a real data set, namely, the global land temperature anomalies data from year 1880 to 2020 (download from the canvas course room), to do exploratory data analysis. The data set contains 141 data points (one averaged value per year), and is provided in an excel format. The dataset is from `https://www.statista.com/statistics/1048518/average-land-sea-temperature-anomaly-since-1850/`

1. Use the global land temperature anomalies data to draw line plot, histogram, density plot, box-plot, heatmap, lag-1 plot, auto-correlation function (acf) and partial acf (pacf) graphs (lags up to 40).
2. Take the first order difference of the temperature anomaly dataset. Draw line plot, histogram, density plot, box-plot, heatmap, lag-1 plot, acf and pacf graphs (lags up to 40).
3. Test if the original and the differenced temperature anomaly series are random or not.
4. Test if the original and the differenced temperature anomaly series are stationary or not.
5. Perform the classical decomposition and STL decomposition on the dataset.

Answer the following questions:

- What is a stationary time series?
- If a series is not stationary, is it possible to transform it into a stationary one? If so, give one technique to do it?
- Is the global land temperature anomaly series stationary? Why or why not?
- Is the data set after the first-order difference stationary?
- Why is it useful to decompose a time series into a few components? What are the typical components in a time-series decomposition?

3.2 Task 2. Feature Extraction

In this task, you will extract a variety of features from a synthetic signal and a given electroencephalogram (EEG) time-series data. Typically this includes statistical features, time-domain and frequency-domain features.

3.2.1 Task 2.1. Frequency components of a synthetic time-series signal

Data: Generate a series of five sequential sine wave signals for five seconds, each sine wave lasting 1 second. The nth sine wave signal $x_n = sin(2\pi \cdot n \cdot f)$, where $f = 10$, and $n = 1, 2, 3, 4, 5$, i.e., frequency 10Hz, 20Hz, 30Hz, 40Hz and 50Hz. The series is digitalized with a sampling rate is 200 Hz.

Task:

1. Draw a line plot of the series.
2. Draw power spectrum (power density graph) of the series.
3. Draw the spectrogram of the series.
4. Draw and compare the ACF and PACF graphs of the first one-second (frequency 10Hz) and the second one-second series (frequency 20Hz), with lags up to 50.

3.2.2 Task 2.2. Statistical features and discovery of event-related potential

Dataset: 1 second of scalp EEG data sampled at 500 Hz during 1,000 trials in two conditions. The data (File name: 02_EEG-1.mat from the canvas course website) and its description below are from `https://mark-kramer.github.io/Case-Studies-Python/02.html`

Voltage recordings from the scalp surface - the electroencephalogram or EEG - provide a powerful window into brain voltage activity.

An undergraduate student volunteers to participate in a psychology study at his university. In this study, EEG electrodes (sampling rate 500 Hz, i.e., 500 samples per second) are placed on the student's scalp, and he is seated in a comfortable chair in a dark, electrically isolated room. The student is instructed to place headphones over his ears and listen to a series of repeated sounds. The sounds consist of two tones - either a high pitch tone or a low pitch tone. A single tone is presented once every few seconds, and the student responds with a button press to the low pitch tone. The tone presentation is repeated to collect the EEG response to numerous presentations of the two tones, as shown below:

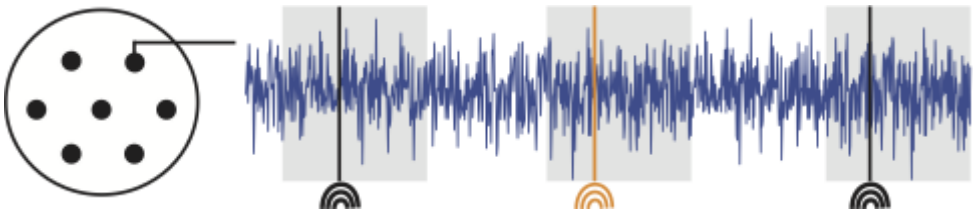

Figure 1: Illustration of EEG recording

In this illustration of the EEG experiment, the EEG electrodes are placed on the scalp surface of a human subject (left). The EEG activity (blue) is recorded as a function of time during presentation of high pitch tones (black) and low pitch tones (orange).

The EEG data were recorded at a single electrode for 1000 presentations (trials) of the high pitch tone, and 1000 presentations of the low pitch tone. Each presentation or "trial" contains 1 second of EEG data, and the tone occurs at 0.25 second into the trial.

Task: We will analyze these data to determine what (if any) activity is evoked (so called Event-related potential, ERP) following two different types of stimuli presented to a human subject. Specifically,

1. Visualize the response, i.e., ERP of the EEG, in the two conditions, A and B.

2. Find the brain activity frequency in the data of condition A (see below for condition A and B).

In the EEG data, there are data from two conditions, EEGa and EEGb. One condition is called Condition A (EEGa), and the other is Condition B (EEGb). Use the following to read the .mat file and read the two-condition data.

```
from scipy.io import loadmat       # Import function to read data.
data = loadmat('matfiles/02_EEG-1.mat')
data.keys()
EEGa = data['EEGa']
EEGb = data['EEGb']
t = data['t'][0]
```

Hint: Read Chapter 2 "The Event-Related Potential" at `https://mark-kramer.github.io/Case-Studies-Python/02.html`

3.2.3 Task 2.3. Features of observed rhythms in EEG

In this task, we use data recorded in the scalp EEG. It is a 2 seconds of scalp EEG data sampled at 1000 Hz. The EEG data set is given in the matlab format. The file is named 03_EEG-1.mat, which is downloadable from the course canvas room.

The EEG provides a measure of brain voltage activity with high temporal resolution (typically on the order of milliseconds) but poor spatial resolution (on the order of 10 cm^2 of cortex). The EEG data set records EEG activity from a single scalp electrode. The data set comprises 2 seconds of scalp EEG data sampled at 1000 Hz (i.e., 2000 data points).

The goal is to characterize the observed rhythms in the data. We are interested in relevant features in the recorded data set. We will analyze these data to determine its statistical features, temporal features and spectral features. From the analysis, we will see if and what rhythmic activity is present. For spectral analysis, we will apply an important technique to characterize rhythmic behavior in data - the Fourier transform and power spectral density or "spectrum" - and many subtleties associated with this technique.

1. For the EEG data set, draw the line plot, histogram, density plot, box plot, lag-1 plot, ACF and PACF graphs (lags up to 50).

2. Show the statistical characteristics of the EEG data, such as mean, variance, standard deviation.

3. Compute the auto-covariance of the EEG data. Draw and save the auto-covariance graph. Examine the auto-covariance plot, Why does the auto-covariance exhibit repeated peaks and troughs approximately every 0.0166 s?

4. Compute and plot the power-spectrum of the EEG data. Show it in both linear scale and log (dB) scale. To emphasize low-amplitude rhythms hidden by large-amplitude oscillations is to change the scale of the spectrum to decibels.

Hint: Read Chapter 3 "The Power Spectrum (Part 1)" at `https://mark-kramer.github.io/Case-Studies-Python/03.html`

After completing this task, you know what features to be extracted and how to do that. Hopefully also understand why they are useful and perhaps important. Now it is time to answer the following questions:

- What features do you typically consider useful for analyzing and modeling time-series data?
- What features are specific for time-series, and what are general for both time-series and non-time-series data?
- How are auto-covariance and auto-correlation are defined for a time series? Give mathematical formulas for the definitions.
- Assume a short time-series $\{1, 2, 3, 4, 5, 6, 7, 8, 7, 6, 5, 4, 3, 2, 1\}$. (1) Calculate the auto-covariance and auto-correlations for all valid lags. Do the calculations manually. (2) Write a Python program to validate your calculations. (3) Draw the ACF graph for the time series.

3.3 Task 3. Lab Completion and Documentation

After completing the lab tasks above, you are not complete yet. You as a group need to do the following to finalize:

1. Result approval: ask a lab assistant to check your results and approve your lab completion. The lab assistant will also ask all questions present in this lab manual and possible other questions regarding your implementations. Every member in your group should be able to answer the questions by him/herself.
2. Write report: write a short technical document, where you write down what you have done, how you have done it, what results you have obtained, and discuss the results (often the questions in the lab manual contains discussion points), and present conclusions in your own words. Try to use figures and tables to assist your explanations and discussions.
3. Prepare deliverables. Prepare a zip file for all your source code and the results, and write a a readme file. In the readme file, (1) describe the file folder structure and the role of each file in a folder; (2) give a short instruction about how to execute your code and obtain the results.

   As a group work, your write one report and prepare one deliverable package per group. In the report, you list all group members' names on the first cover page.
4. Submit report and deliverables: After (1) (2) (3), you (the group leader) submit your technical report to the Canvas course page below. The lab assistant will review your report and execute your code (if needed to double check) and, if accepted, your lab is counted as completed. If not, comments will be given to you for revision and re-submission of your technical report/deliverable.

   `https://canvas.kth.se/courses/46239`

4 Appendix

4.1 Brief on ADF test

Augmented Dickey Fuller (ADF) test is one of the most common statistical tests used to test whether a given time series is stationary or not. The ADF test is fundamentally a statistical significance test based on unit root. That means, there is a hypothesis testing involved with a null and alternative hypothesis.

For the ADF test, the null hypothesis is that the series is not stationary. The alternative hypothesis is that the series is stationary. As a result a test statistic is computed and p-values get reported. From the test statistic and the p-value, you can judge whether a given series is stationary or not.

The statsmodel package provides a reliable implementation of the ADF test via the adfuller() function in statsmodels.tsa.stattools. It returns the following outputs:

- The p-value
- The value of the test statistic
- Number of lags considered for the test
- The critical value cutoffs.

When the test statistic is lower than the critical value shown, you reject the null hypothesis that the series is not stationary, and consider that the time series is stationary. Otherwise, you accept the null hypothesis.

Equivalently, you check the p-value. If the p-value is less or equal to the significance level (default value 0.05), then you reject the null hypothesis and consider that the series is stationary. Otherwise, if the p-value is larger than the significance level, you accept the null hypothesis, and consider that the series is not stationary.

Let's run the ADF test on a random number series.

```
from statsmodels.tsa.stattools import adfuller
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# ADF test on random numbers
series = np.random.randn(100)

% Visual inspection
fig, axes = plt.subplots(figsize=(10,7))
plt.plot(series);
plt.title('Random');

% ADF test
result = adfuller(series, autolag='AIC')
print(f'ADF Statistic: {result[0]}')
print(f'p-value: {result[1]}')
```

```
for key, value in result[4].items():
    print('Critial Values:')
    print(f'   {key}, {value}')
```

Suppose that you get the following ADF test result:

```
ADF Statistic: -7.4715740767231456
p-value: 5.0386184272419386e-11
Critial Values:
   1%, -3.4996365338407074
Critial Values:
   5%, -2.8918307730370025
Critial Values:
   10%, -2.5829283377617176
```

The p-value (5.0386184272419386e-11) is much less than the significance level of 0.05 and hence we can reject the null hypothesis and judge that the series is stationary.

As a final note, the KPSS test (KPSS short for Kwiatkowski-Phillips-Schmidt-Shin), is another commom unit-root test that checks the stationarity of a given series around a determinstic trend. It is similar to ADF. However keep in mind that the null and alternative hypothesis for the KPSS test are opposite that of the ADF test. Null Hypothesis: The process is trend stationary. Alternative Hypothesis: The series has a unit root (series is not stationary).

If you want to dig more, see more description with example on the statesmodels website: `https://www.statsmodels.org/dev/examples/notebooks/generated/stationarity_detrending_adf_kpss.html`

4.2 The Ljung-Box test

The Ljung-Box test is a statistical test that checks if auto-correlations are significantly from zero in a time series. It is often used to test if the residual series after prediction is random noise.

It has the following two hypotheses:

- Null hypothesis (H0): The series is independently distributed, i.e, random.
- Alternative hypothesis (Ha): The series is not independently distributed, exhibiting serial correlation. It is not random.

To perform the Ljung-Box test on a data series in Python, we can use the acorr_ljungbox() function from the statsmodels library:

```
acorr_ljungbox(x, lags=None),
```

where x is the series and lags the number of lags to test.

```
import statsmodels.api as sm
data = sm.datasets.sunspots.load_pandas().data
res = sm.tsa.ARMA(data["SUNACTIVITY"], (1,1)).fit(disp=-1)
sm.stats.acorr_ljungbox(res.resid, lags=[10], return_df=True)
       lb_stat     lb_pvalue
10  214.106992  1.827374e-40
```

The lb_stat is the value of the test statistic, and lb_pvalue is the p value of the test.

- If lb_pvalue > 0.05 (default threshold), then accept the Null hypothesis that the series is independent, meaning that the series is random.
- If lb_pvalue < 0.05 (default threshold), then reject the Null hypothesis and accept the alternative hypothesis that the series is dependent, meaning that the series is not random.

In this example, lb_pvalue $<< 0.5$, thus we reject the null hypothesis and accept the alternative hypothesis, i.e., the series is not random. Since it is much less than 0.5, it means the series is not random at all.

You can find more details at `https://www.statsmodels.org/dev/generated/statsmodels.stats.diagnostic.acorr_ljungbox.html`

4.3 Time series decomposition

Decomposition is an analytical approach often useful to uncover insightful structure on the data. There are various means or models for decomposition. One reason is that the notion of seasonal variation is always intrinsically ambiguous: whether the temporal variation should be considered Seasonal, Trend, or Remainder is, to some degree, a matter of opinion and determined by choice of model and model parameters. This is true in Seasonal Trend decomposition using Loess (STL) as well as any seasonal variational approach.

The statesmodels libaray implements both the classical decomposition method (seasonal_decompose()) and the STL method (STL()). The classical decomposition is good for conceptual understanding while the STL method is more sophisticated and preferred for analysis.

`https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal_decompose.html`

`https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.STL.html`

The classical decomposition may be additive or multiplicative.

- The additive model is $Y[t] = T[t] + S[t] + e[t]$
- The multiplicative model is $Y[t] = T[t] \cdot S[t] \cdot e[t]$

The results are obtained by first estimating the trend by applying a moving average to the data. The trend is then removed from the series and the average of this de-trended series for each period is returned as seasonal component.

The STL method assumes an additive model. It was developed by Cleveland *et al.* Journal of Official Statistics 6, No. 1, pp 3-33, 1990. Key to the STL approach is Loess (LOcal regrESSion) smoothing.

```
from pandas import read_csv
from matplotlib import pyplot
from statsmodels.tsa.seasonal import seasonal_decompose

series = read_csv('airline-passengers.csv', header=0, index_col=0, squeeze=True)
```

```
# Show the data set
print(series)

# multiplicative decompose time series.
# Additive decompose time series, use model = 'additive'
result = seasonal_decompose(series, model='multiplicative', period=12)
# one command to plot four graphs
result.plot()
pyplot.show()

# use STL
from statsmodels.tsa.seasonal import STL
res = STL(series, period=12).fit()
res.plot()
pyplot.show()

# If you want to view the original and three split components
print(result.observed)
print(result.trend)
print(result.seasonal)
presnt(result.resid)
```

About STL, you may further refer to the following materials.

[1] Cleveland, R.B., Cleveland W.S., McRae J.E., Terpenning, I. (1990). "STL: Seasonal-Trend Decomposition Procedure Based on Loess", Journal of Official Statistics.

[2] Gardner, Dillon R. (2017) "STL Algorithm Explained: STL Part II". (online).