# Weather Prediction

## Based on ARIMA, RNN and Prophet

Xiangqing Wang

WISE

2024-05-31

# Outline

# Background

- Weather prediction is an essential tool that impacts almost every aspect of modern life.
    - It enhances safety, supports economic activities, aids in disaster management, and contributes to public health and welfare.
    - As technology advances, the accuracy and reliability of weather forecasts continue to improve, providing even greater benefits to society.
- This study: focusing on city of Shenyang in China as a sample to establish a model to analyze the **model composition** and **predict the weather condition**.

# Outline

# Data Introduction

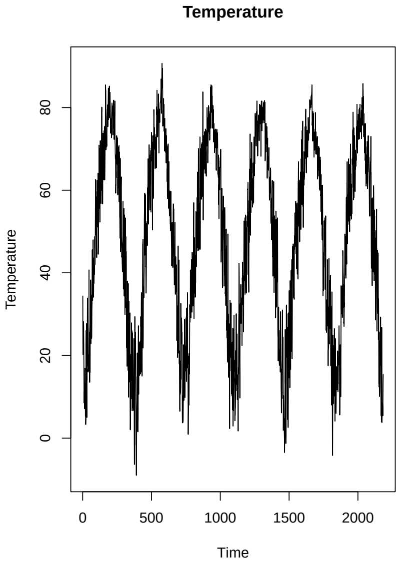
- Data Source: NOAA's global sites from 1929 to 2024, published by the National Oceanic and Atmospheric Administration's (NOAA) National Center for Environmental Information (NCEI)
- Here in this study: Shenyang site from 1st January 2017 to 30th December 2022, together 2184 days

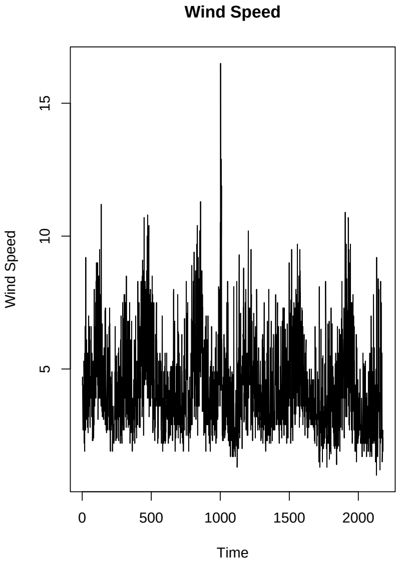
# Data Introduction

Weather Information from 1st January 2017 to 30th December 2022 in the city of Shenyang, China.

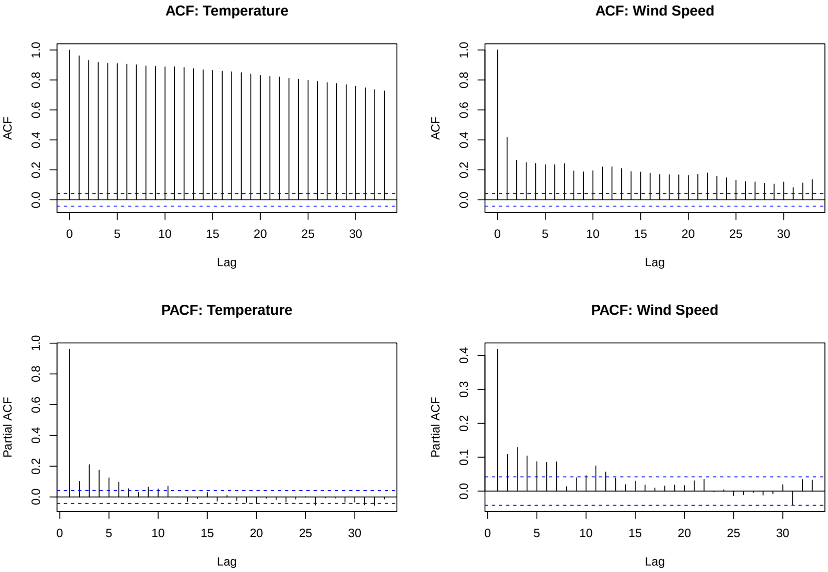
```
##      DATE                TEMP             WDSP
##  Length:2184        Min.   :-9.00    Min.   :  1.000
##  Class :character   1st Qu.:29.05    1st Qu.:  3.100
##  Mode  :character   Median :52.20    Median :  3.950
##                     Mean   :48.64    Mean   :  6.216
##                     3rd Qu.:69.62    3rd Qu.:  5.300
##                     Max.   :90.70    Max.   :999.900
```

# Visualization



# ACF & PACF Plot

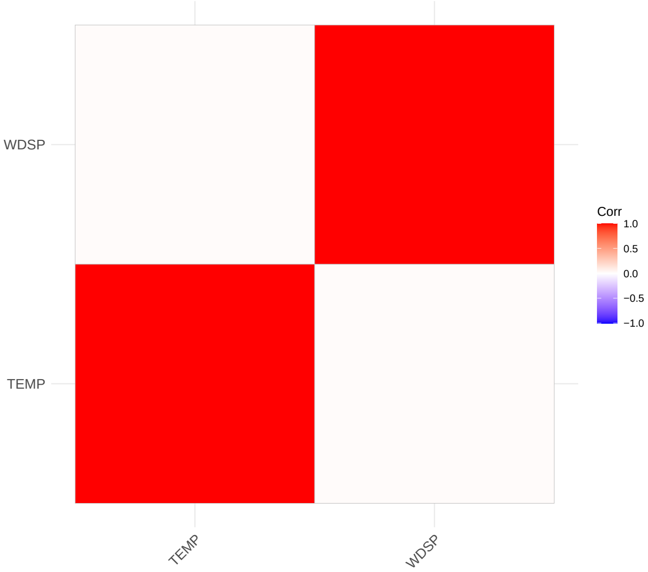


# Model Introduction

- ARIMA Model ($ARIMA(p, d, q)$)
  - p: AR coefficient
  - d: step of differencing
  - q: MA coefficient
- Simple RNN
  - A class of neural networks designed for processing sequential data.
- Prophet Model
  - Prophet, or "Facebook Prophet," is an open-source library for univariate (one variable) time series forecasting developed by Facebook.
  - It is particularly suitable for forecasting time series with **obvious seasonal periodicity** (such as temperature, commodity sales, traffic flow, etc.).

# Outline

# Correlation



# Wind Speed Fitting

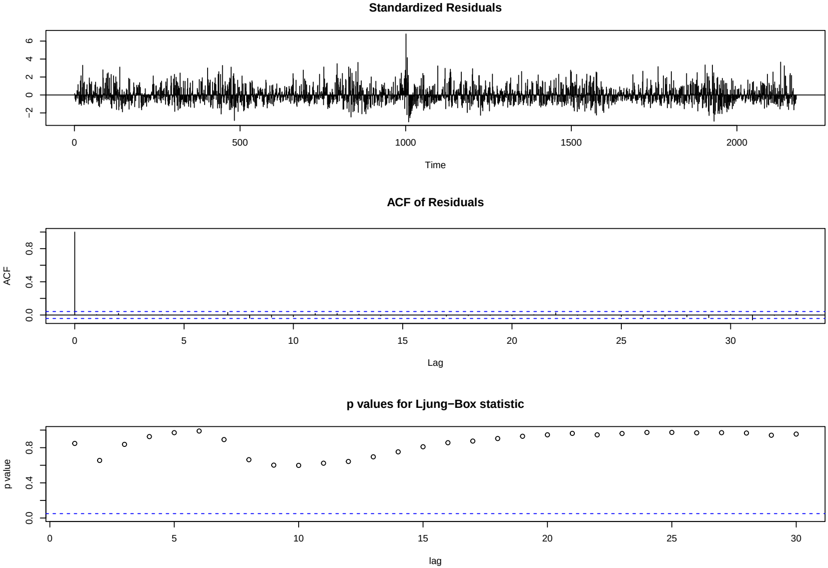
```
## Series: (d.wind)
## ARIMA(0,0,2) with zero mean
##
## Coefficients:
##           ma1      ma2
##       -0.6883  -0.2053
## s.e.   0.0206   0.0209
##
## sigma^2 = 2.459:  log likelihood = -4071.79
## AIC=8149.58   AICc=8149.59   BIC=8166.64
```

Therefore, the ARIMA fitted model is

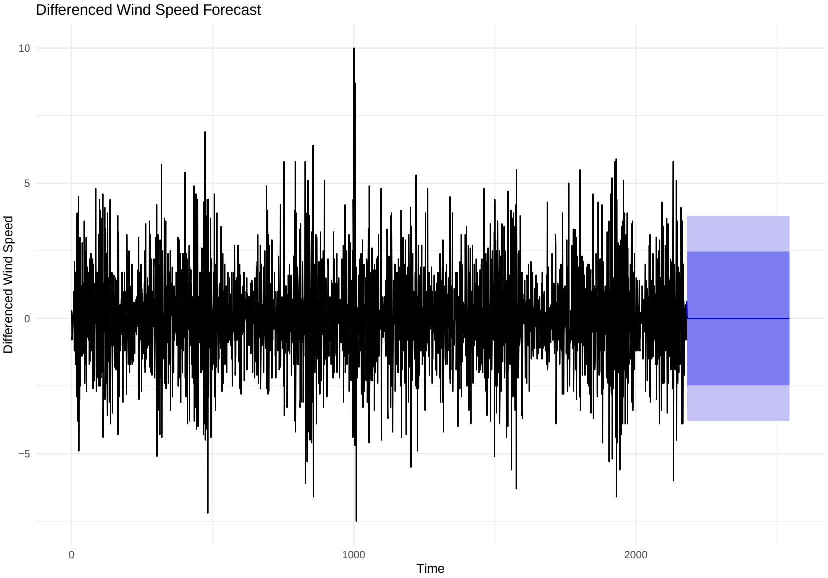
$$(1-L) \times y_t = \epsilon_t - 0.69 \times \epsilon_{t-1} - 0.21 \times \epsilon_{t-2}$$

$$\epsilon_t \sim WN(0, 2.46)$$

# Wind Speed: Residuals Checking



# Wind Speed: Forecasting



# Temperature: Fitting

```
## Series: d.temp
## ARIMA(3,0,1) with zero mean
##
## Coefficients:
##           ar1      ar2      ar3      ma1
##        0.5108  -0.1500  -0.0323  -0.7680
## s.e.   0.0345   0.0244   0.0255   0.0274
##
## sigma^2 = 36.31:  log likelihood = -7016.52
## AIC=14043.05   AICc=14043.07   BIC=14071.49
```

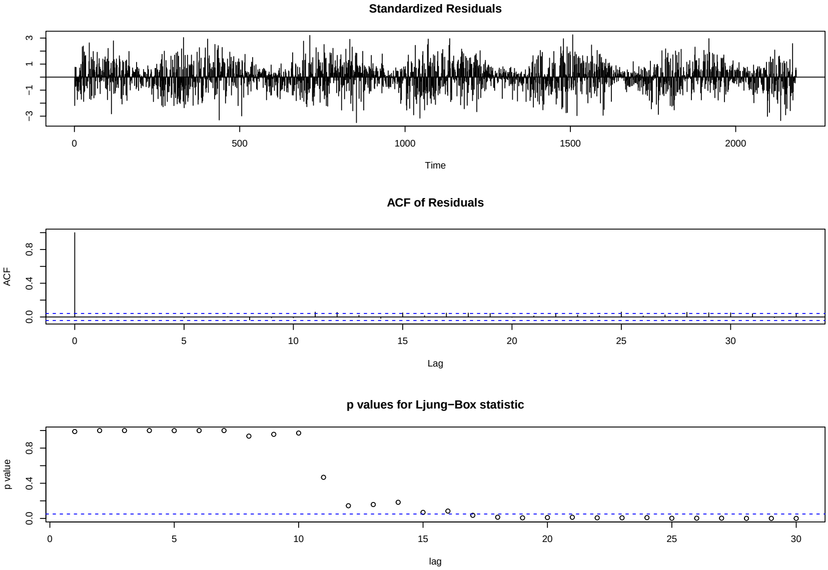
Therefore, the estimated model is

$$(1-L) \times y_t = 0.51(1-L) \times y_{t-1} - 0.15(1-L) \times y_{t-2}$$

$$-0.03(1-L) \times y_{t-3} + \epsilon_t - 0.77\epsilon_{t-1}$$

$$\epsilon_t \sim WN(0, 36.31)$$

# Temperature: Residuals Checking



# Key Features of Simple RNN

1. **Sequential Data Processing:** the order of the data points matters
2. **Hidden States:** RNNs maintain a hidden state that is updated at each time step. This hidden state acts as a memory that captures information from previous time steps, allowing the network to learn temporal dependencies.
3. **Recurrent Connections:** RNN loop back from the hidden state to itself. This feedback loop enables the network to retain information over time and leverage past data to make predictions about future events.
4. **Parameter Sharing:** RNNs share parameters across different time steps, which reduces the number of parameters to be learned and makes the model more efficient.

# Simple Structure of RNN

- **Input Sequence**: $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$
- **Hidden State**: $\mathbf{h}_t$ at time step $t$
- **Output Sequence**: $\mathbf{y} = \{y_1, y_2, \ldots, y_T\}$

The hidden state $\mathbf{h}_t$ is updated using the input at the current time step $x_t$ and the hidden state from the previous time step $\mathbf{h}_{t-1}$:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{xh} x_t + \mathbf{W}_{hh} \mathbf{h}_{t-1} + \mathbf{b}_h)$$

- $\sigma$: a non-linear activation function (e.g., tanh or ReLU).
- $\mathbf{W}_{xh}$: the weight matrix for the input to hidden state connection.
- $\mathbf{W}_{hh}$: the weight matrix for the hidden to hidden state connection.
- $\mathbf{b}_h$: the bias term for the hidden state.

# Simple Structure of RNN

The output at each time step $y_t$ can be computed as:

$$y_t = \mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y$$

- $\mathbf{W}_{hy}$: the weight matrix for the hidden state to output connection.
- $\mathbf{b}_y$: the bias term for the output.

# RNN Fitting Process in Practice

1. Scale the Data `MinMaxScaler`
2. Convert Time Series to Supervised Learning Problem
3. Reshape Data for RNN Input: [samples, time steps, features].
4. Build the RNN Model: `tensorflow.keras`
    - Use a Sequential model.
    - Add a SimpleRNN layer with 50 units.
    - Add a Dense layer with 1 unit for the output.
    - Compile the model with the **Adam optimizer** and **Mean Squared Error (MSE)** loss function.
5. Train the Model:
    - 100 epochs with a batch size of 32
6. Make Predictions
7. Model Evaluation `sklearn.metrics`

# Model Evaluation

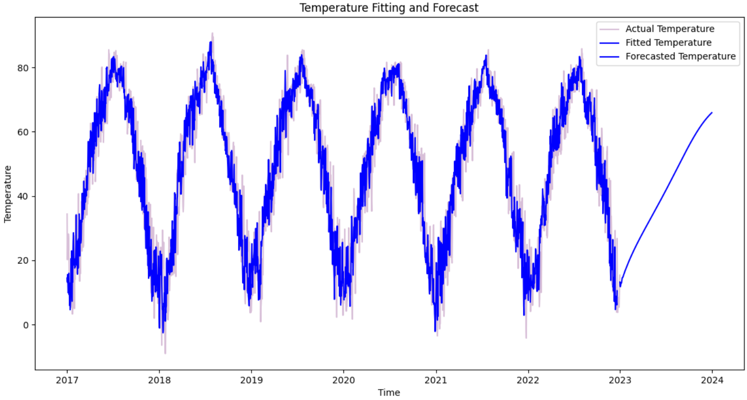
- MSE: 78.27741187023945
- MAE: 6.98986138816O2625
- $R^2$: 0.8550623276781453
- AIC: -6770.927499234256

$$AIC = n * log(mse) + 2 * k$$

- BIC: 8296.995912425538

$$BIC = n * log(mse) + k * log(n)$$

# Forecast

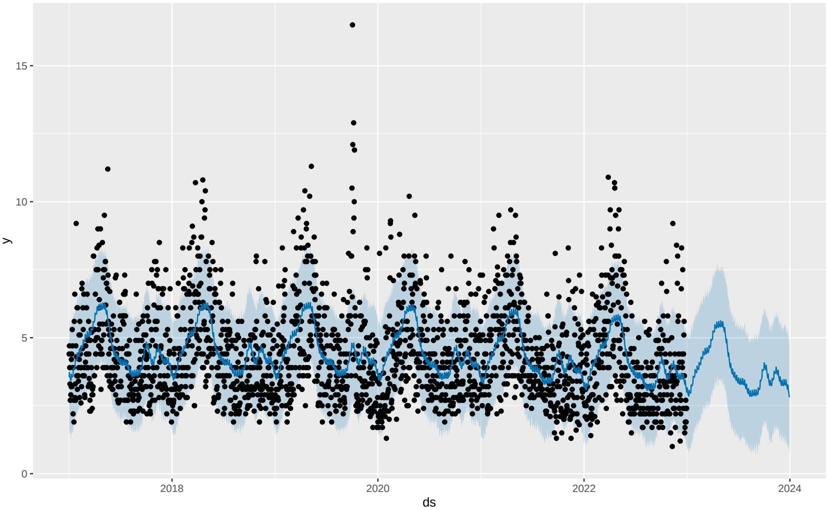

Temperature Fitting and Forecast
Actual Temperature
Fitted Temperature
Forecasted Temperature
Temperature
80
60
40
20
0
2017
2018
2019
2020
2021
2022
2023
2024
Time

# Prophet Model

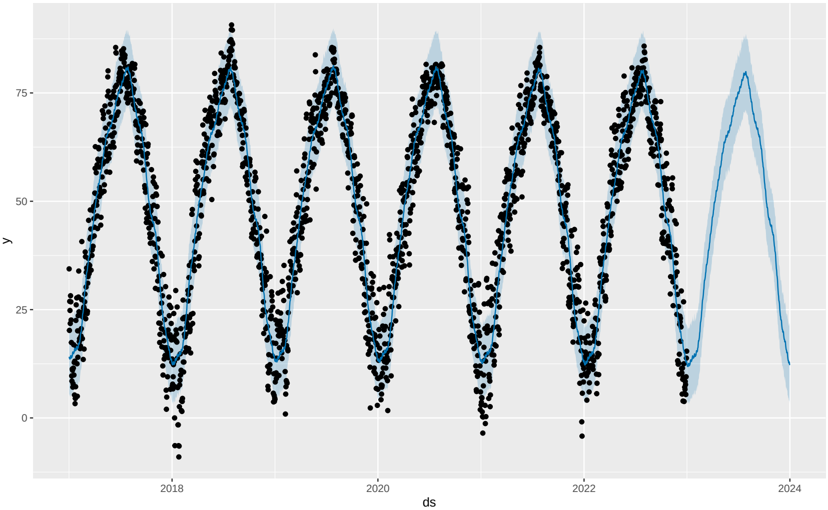
**Usage**: The input data requires columns of `ds` and `y`, containing the *date* and *numeric value* respectively.

- The ds column should be YYYY-MM-DD for a date, or YYYY-MM-DD HH:MM:SS for a timestamp.
- Model fitting: `prophet()` function
- Predict:
  - `future <- make_future_dataframe(model, periods = h)`
  - `prediction <- predict(model future)`

# Prophet Model Fitting: Wind Speed



# Prophet Model Fitting: Temperature



# Outline

# Conclusion & Discussions

- **ARIMA** is limited at fitting the daily weather data
  - For monthly specific data (in this case, temperature in summer is always higher than winter), the forecasting is not that good at the level of day.
- **Recurrent Neural Network** gives an satisfactory model to fit the data but the forecasts for the future need to be developed.
- **Prophet Model** gives an satisfactory model to implement forecasting and explore the seasonal pattern behind it, which could serve as a strong tool to help predict.

# Acknowledgements

Thank You!