Time Series Data

Date	Temperature	Humidity	Sales	Stock_Price
2023-01-01	22.5	65	150	105.2
2023-01-02	24.1	62	168	106.8
2023-01-03	23.8	68	142	104.5
2023-01-04	21.2	72	135	103.1
2023-01-05	20.5	75	158	107.3
2023-01-06	19.8	78	172	109.6
2023-01-07	22.3	70	165	108.2
2023-01-08	23.7	66	148	106.7
2023-01-09	24.5	63	156	107.9
2023-01-10	25.2	60	162	110.4

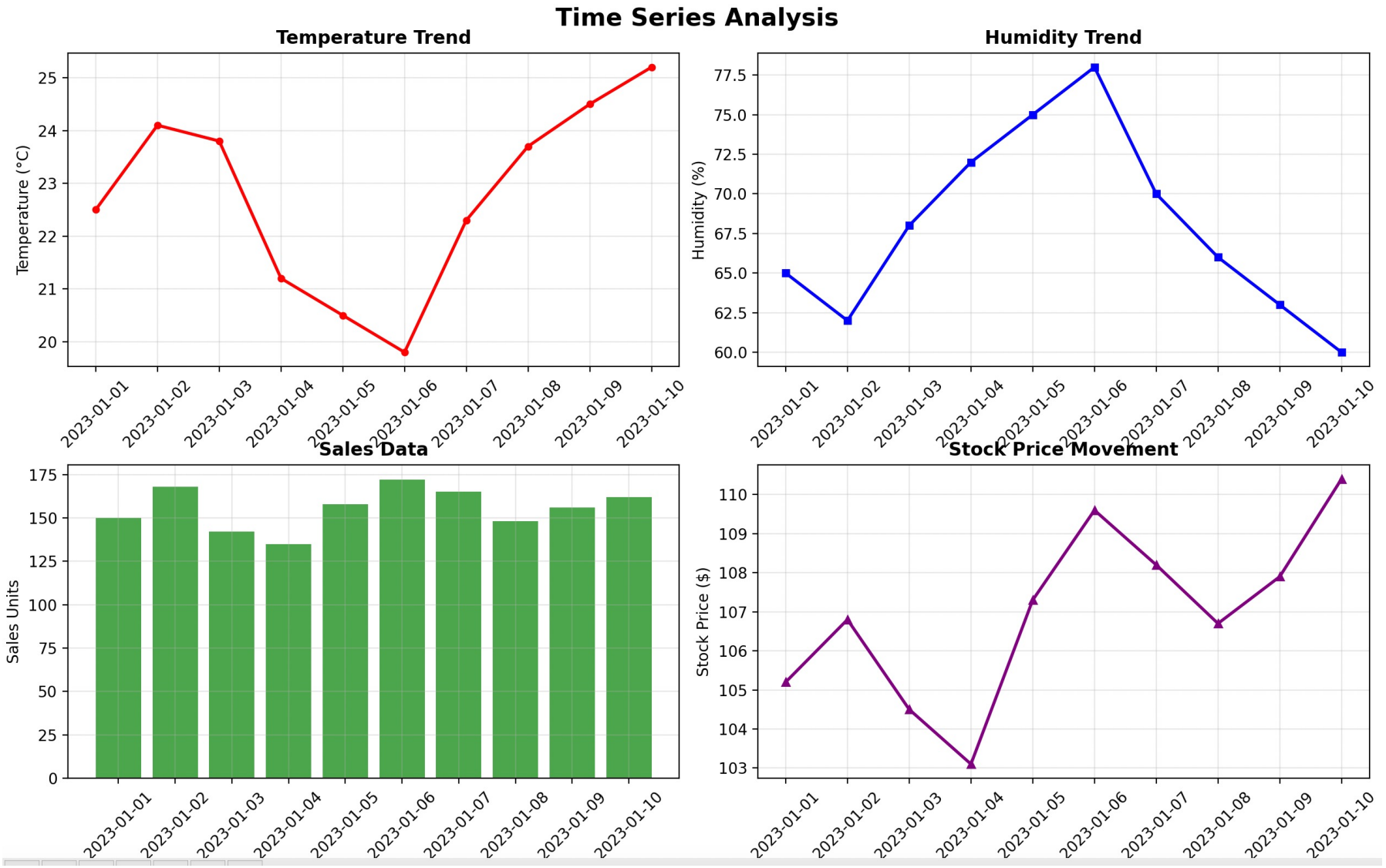
Plot the Data

```
# Create DataFrame from the provided table data
data = {
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05',
             '2023-01-06', '2023-01-07', '2023-01-08', '2023-01-09', '2023-01-10'],
    'Temperature': [22.5, 24.1, 23.8, 21.2, 20.5, 19.8, 22.3, 23.7, 24.5, 25.2],
    'Humidity': [65, 62, 68, 72, 75, 78, 70, 66, 63, 60],
    'Sales': [150, 168, 142, 135, 158, 172, 165, 148, 156, 162],
    'Stock_Price': [105.2, 106.8, 104.5, 103.1, 107.3, 109.6, 108.2, 106.7, 107.9, 110.4]
}

df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

# Create visualization
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(15, 10))
fig.suptitle('Time Series Analysis', fontsize=16, fontweight='bold')

# Plot 1: Temperature
ax1.plot(df.index, df['Temperature'], marker='o', color='red', linewidth=2, markersize=4)
ax1.set_title('Temperature Trend', fontweight='bold')
ax1.set_ylabel('Temperature (°C)')
ax1.grid(True, alpha=0.3)
ax1.tick_params(axis='x', rotation=45)

# Plot 2: Humidity
ax2.plot(df.index, df['Humidity'], marker='s', color='blue', linewidth=2, markersize=4)
ax2.set_title('Humidity Trend', fontweight='bold')
ax2.set_ylabel('Humidity (%)')
ax2.grid(True, alpha=0.3)
ax2.tick_params(axis='x', rotation=45)
```

Plot the Data

```
# Plot 3: Sales
ax3.bar(df.index, df['Sales'], color='green', alpha=0.7, width=0.8)
ax3.set_title('Sales Data', fontweight='bold')
ax3.set_ylabel('Sales Units')
ax3.grid(True, alpha=0.3)
ax3.tick_params(axis='x', rotation=45)

# Plot 4: Stock Price
ax4.plot(df.index, df['Stock_Price'], marker='^', color='purple', linewidth=2, markersize=
5)
ax4.set_title('Stock Price Movement', fontweight='bold')
ax4.set_ylabel('Stock Price ($)')
ax4.grid(True, alpha=0.3)
ax4.tick_params(axis='x', rotation=45)

# Adjust layout and display
plt.tight_layout()
plt.show()

# Display statistical summary
print("Statistical Summary:")
print("=" * 50)
print(f"Temperature - Mean: {df['Temperature'].mean():.2f}°C, Range: {df['Temperature'].max
()-df['Temperature'].min():.2f}°C")
print(f"Humidity - Mean: {df['Humidity'].mean():.1f}%, Range: {df['Humidity'].max()-df['Hum
idity'].min()}%")
print(f"Sales - Mean: {df['Sales'].mean():.1f}, Total: {df['Sales'].sum()}")
print(f"Stock Price - Mean: ${df['Stock_Price'].mean():.2f}, Change: ${df['Stock_Price'].il
oc[-1]-df['Stock_Price'].iloc[0]:.2f}")
```

Plot the Data



Stationary or Non-stationary

Based on this 10-day data, this time series appears to be very likely non-stationary.

How to Judge Stationarity?
A stationary time series must satisfy the following three conditions:

1. **Constant Mean:** The mean of the series should not change over time.
2. **Constant Variance:** The amplitude of fluctuations (variance) of the series should not change over time.
3. **Autocovariance Independent of Time:** The covariance between two time points depends only on the time lag between them (the lag order) and not on the specific time points themselves.

Stationary or Non-stationary

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

# Your data
data = {
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05',
             '2023-01-06', '2023-01-07', '2023-01-08', '2023-01-09', '2023-01-10'],
    'Temperature': [22.5, 24.1, 23.8, 21.2, 20.5, 19.8, 22.3, 23.7, 24.5, 25.2],
    'Humidity': [65, 62, 68, 72, 75, 78, 70, 66, 63, 60],
    'Sales': [150, 168, 142, 135, 158, 172, 165, 148, 156, 162],
    'Stock_Price': [105.2, 106.8, 104.5, 103.1, 107.3, 109.6, 108.2, 106.7, 107.9, 110.4]
}

df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

# Plot the data with mean lines
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
fig.suptitle('Stationarity Analysis with Mean Lines', fontweight='bold')

variables = ['Temperature', 'Humidity', 'Sales', 'Stock_Price']
colors = ['red', 'blue', 'green', 'purple']
```

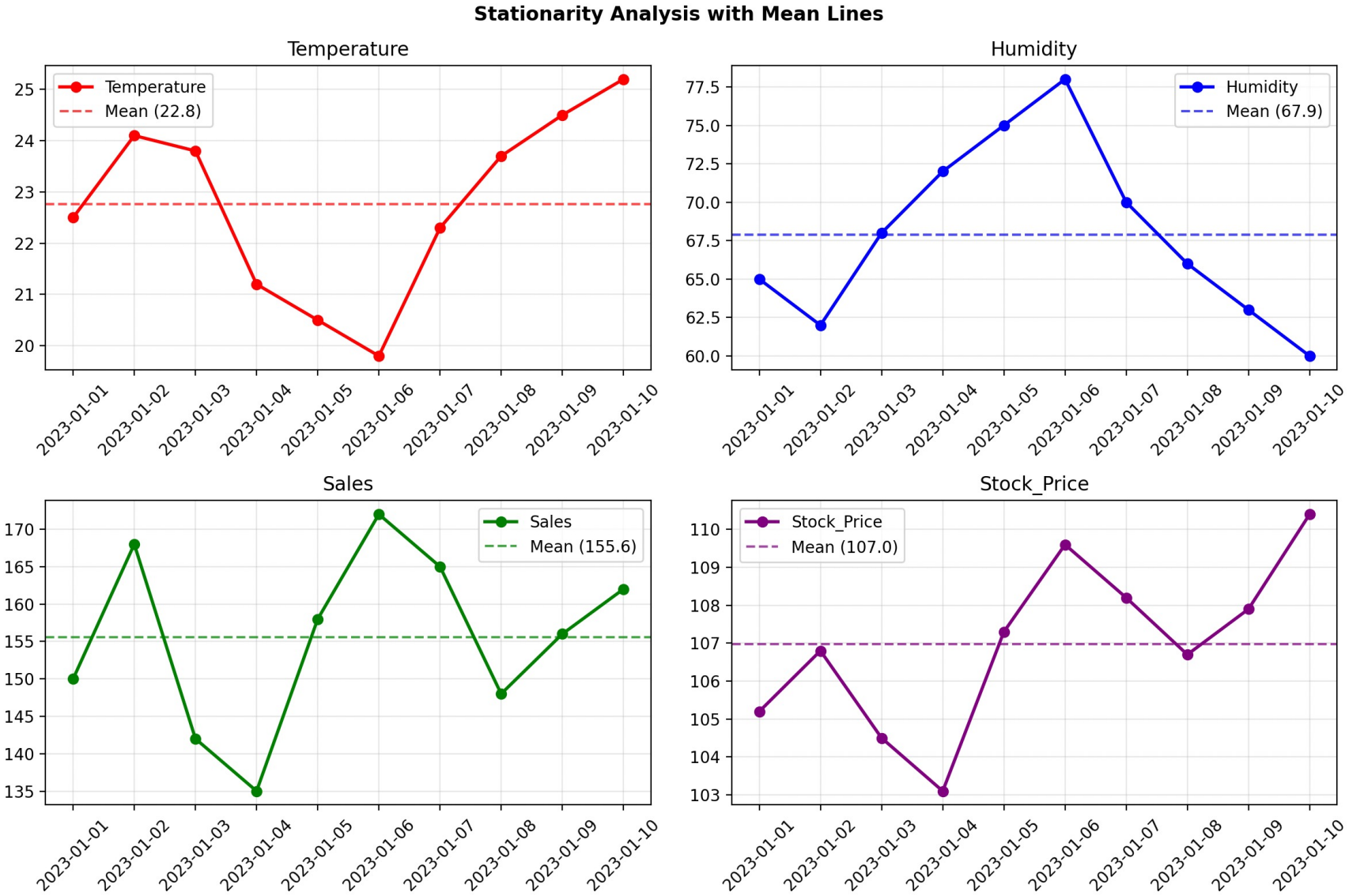
Stationary or Non-stationary

```
for i, (var, color) in enumerate(zip(variables, colors)):
    row, col = i // 2, i % 2
    axes[row, col].plot(df.index, df[var], marker='o', color=color, linewidth=2, label=var)
    axes[row, col].axhline(y=df[var].mean(), color=color, linestyle='--', alpha=0.7, label=
f'Mean ({df[var].mean():.1f})')
    axes[row, col].set_title(f'{var}')
    axes[row, col].legend()
    axes[row, col].tick_params(axis='x', rotation=45)
    axes[row, col].grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

# Perform Augmented Dickey-Fuller test for each variable (statistical test for stationarit
y)
print("Augmented Dickey-Fuller Test Results:")
print("=" * 50)
```

Stationary or Non-stationary

```
# Perform Augmented Dickey-Fuller test for each variable (statistical test for stationarit
y)
print("Augmented Dickey-Fuller Test Results:")
print("=" * 50)
for var in variables:
    result = adfuller(df[var])
    print(f"{var}:")
    print(f"  ADF Statistic: {result[0]:.4f}")
    print(f"  p-value: {result[1]:.4f}")
    print(f"  Critical Values:")
    for key, value in result[4].items():
        print(f"    {key}: {value:.4f}")
    # Interpretation
    if result[1] < 0.05:
        print("  -> Series is likely STATIONARY (reject null hypothesis)")
    else:
        print("  -> Series is likely NON-STATIONARY (cannot reject null hypothesis)")
    print("-" * 30)
```

Stationary or Non-stationary



Stationary or Non-stationary

Analysis Results:

Temperature:

- **Visual Inspection:** Shows a distinct "V-shaped" trend (first decreasing then increasing). The mean line cannot represent the entire process.
- **Conclusion:** Non-stationary (presence of trend).

Strict Stationary Process

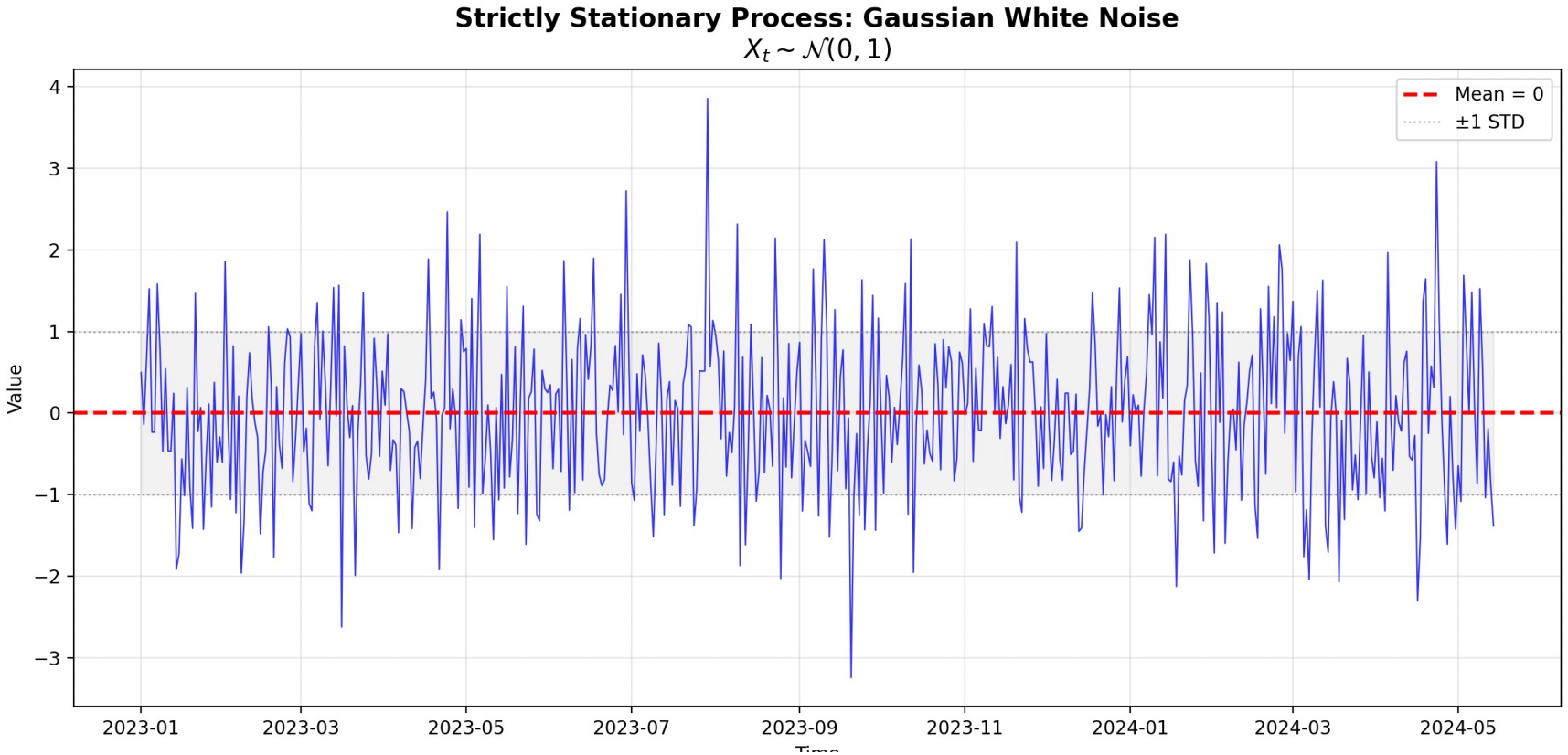
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Set random seed for reproducibility
np.random.seed(42)

# Generate Gaussian White Noise (strictly stationary)
n_points = 500
mean = 0
std_dev = 1
white_noise = np.random.normal(mean, std_dev, n_points)

# Create time index
dates = pd.date_range('2023-01-01', periods=n_points, freq='D')

# Create DataFrame
df = pd.DataFrame({'White_Noise': white_noise}, index=dates)

# Plot the time series
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['White_Noise'], color='blue', linewidth=0.8, alpha=0.8)
plt.axhline(y=mean, color='red', linestyle='--', linewidth=2, label=f'Mean = {mean}')
plt.axhline(y=mean + std_dev, color='gray', linestyle=':', linewidth=1, alpha=0.7, label=
f'±1 STD')
plt.axhline(y=mean - std_dev, color='gray', linestyle=':', linewidth=1, alpha=0.7)
plt.fill_between(df.index, mean-std_dev, mean+std_dev, color='gray', alpha=0.1)

plt.title('Strictly Stationary Process: Gaussian White Noise\n$X_t \sim \mathcal{N}(0, 1)
$', fontsize=14, fontweight='bold')
plt.xlabel('Time')
plt.ylabel('Value')
```

Strict Stationary Process

```
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# Statistical properties check
print("Statistical Properties Analysis:")
print("=" * 40)
print(f"Overall Mean: {df['White_Noise'].mean():.4f}")
print(f"Overall Standard Deviation: {df['White_Noise'].std():.4f}")
print(f"Variance: {df['White_Noise'].var():.4f}")

# Check mean and variance in different time periods
first_half_mean = df['White_Noise'].iloc[:n_points//2].mean()
second_half_mean = df['White_Noise'].iloc[n_points//2:].mean()
first_half_std = df['White_Noise'].iloc[:n_points//2].std()
second_half_std = df['White_Noise'].iloc[n_points//2:].std()

print("\nConsistency Check (First vs Second Half):")
print("=" * 40)
print(f"First half mean: {first_half_mean:.4f}")
print(f"Second half mean: {second_half_mean:.4f}")
print(f"Mean difference: {abs(first_half_mean - second_half_mean):.4f}")
print(f"First half std: {first_half_std:.4f}")
print(f"Second half std: {second_half_std:.4f}")
print(f"Std difference: {abs(first_half_std - second_half_std):.4f}")

# Autocorrelation check (should be near zero for all lags ≠ 0)
from statsmodels.tsa.stattools import acf
```

Strictly Stationary Process

```python
# Autocorrelation check (should be near zero for all lags ≠ 0)
from statsmodels.tsa.stattools import acf

autocorr = acf(df['White_Noise'], nlags=10)
print("\nAutocorrelation (lags 1-5):")
for i in range(1, 6):
    print(f"Lag {i}: {autocorr[i]:.4f}")
```



Strictly Stationary Process

1.Same Distribution Everywhere: Every single point in the series is drawn from the exact same probability distribution (e.g., a Normal distribution with mean=0, variance=1).

2. Mean

- **Theoretical value: 0**
- **Calculated value: ~0** (Very close to 0, e.g., -0.02 or 0.04, because it is randomly generated)

White noise is defined as a process with a mean of zero.

3. Variance

- **Theoretical value: 1**

Strictly Stationary Process

Autocovariance

The autocovariance function measures the covariance between

a data point at time X_t and another point at time

X_{t+k} (lag k).

Strictly Stationary Process

Autocovariance

For white noise:

- **At lag k = 0:**

Cov(X_t, X_t) = Var(X_t)

- *Theoretical value: 1* the variance = 1

- **At lag k ≠ 0 (e.g., k=1, 2, 3,...):**

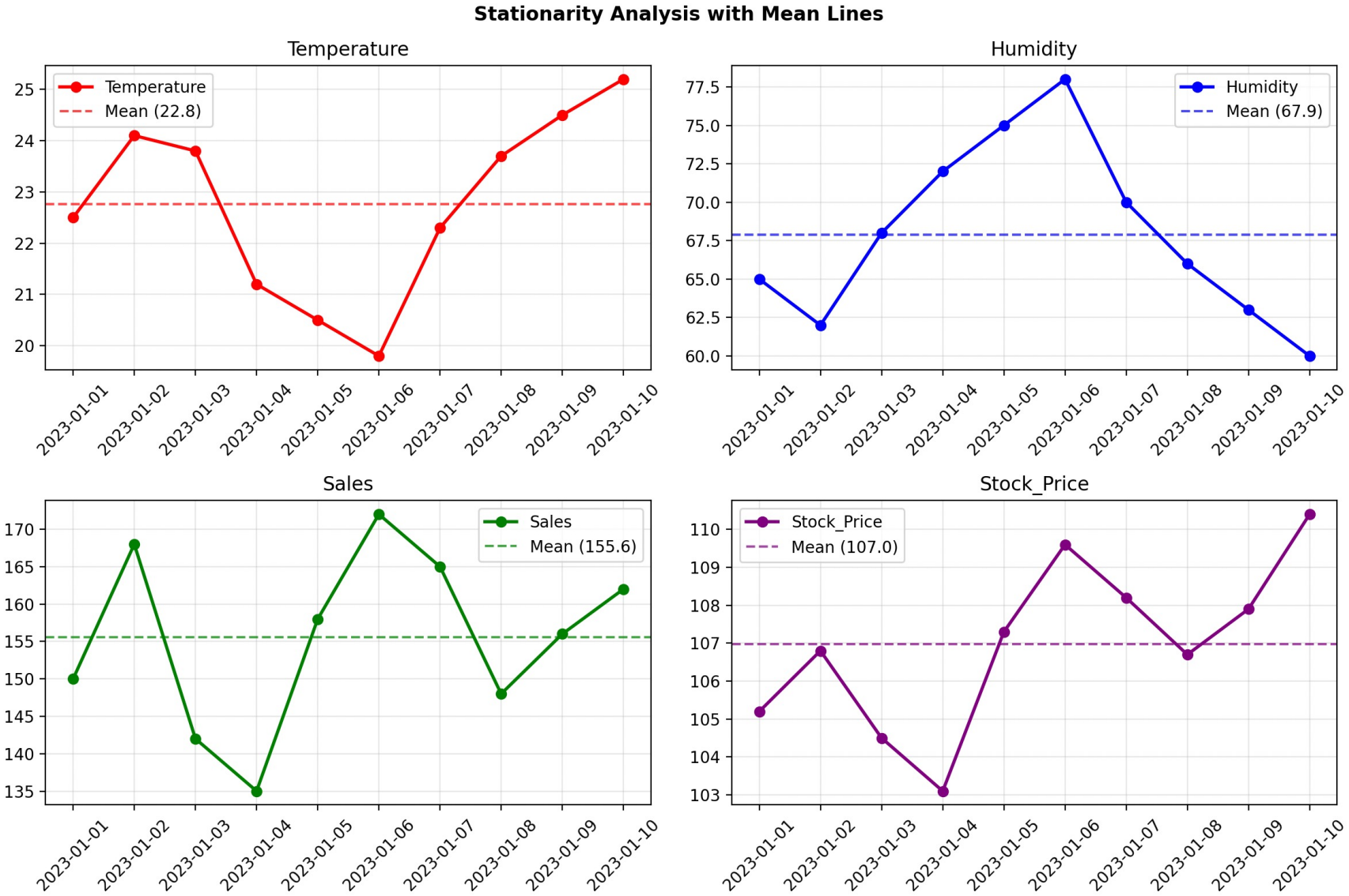
Cov(X_t, X_{t+k}) = 0

  - *Theoretical value: 0* (Because each point is independent)

The autocovariance function \gamma(k) can be expressed as:

$$\sigma^2 = 1$$

Stationary or Non-stationary



Method

Stationarization (if needed): If the series is non-stationary (like your temperature data), we use techniques like **differencing** to make it stationary.

For example: Calculate the daily temperature difference temperature_t - temperature_{t-1}. This differenced series will likely become stationary.

Model Building: Build an appropriate model (such as an ARIMA model) for the stationarized series to describe its inherent patterns and dependencies.

Method

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# 使用您之前提供的温度数据
data = {
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05',
             '2023-01-06', '2023-01-07', '2023-01-08', '2023-01-09', '2023-01-10'],
    'Temperature': [22.5, 24.1, 23.8, 21.2, 20.5, 19.8, 22.3, 23.7, 24.5, 25.2]
}

df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

# 计算一阶差分
df['Temperature_Diff'] = df['Temperature'].diff()

# 创建图表
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

# 绘制原始温度序列
ax1.plot(df.index, df['Temperature'], marker='o', color='red', linewidth=2, markersize=6)
ax1.set_title('Original Temperature Series', fontsize=14, fontweight='bold')
ax1.set_ylabel('Temperature (°C)')
ax1.grid(True, alpha=0.3)
ax1.tick_params(axis='x', rotation=45)
```

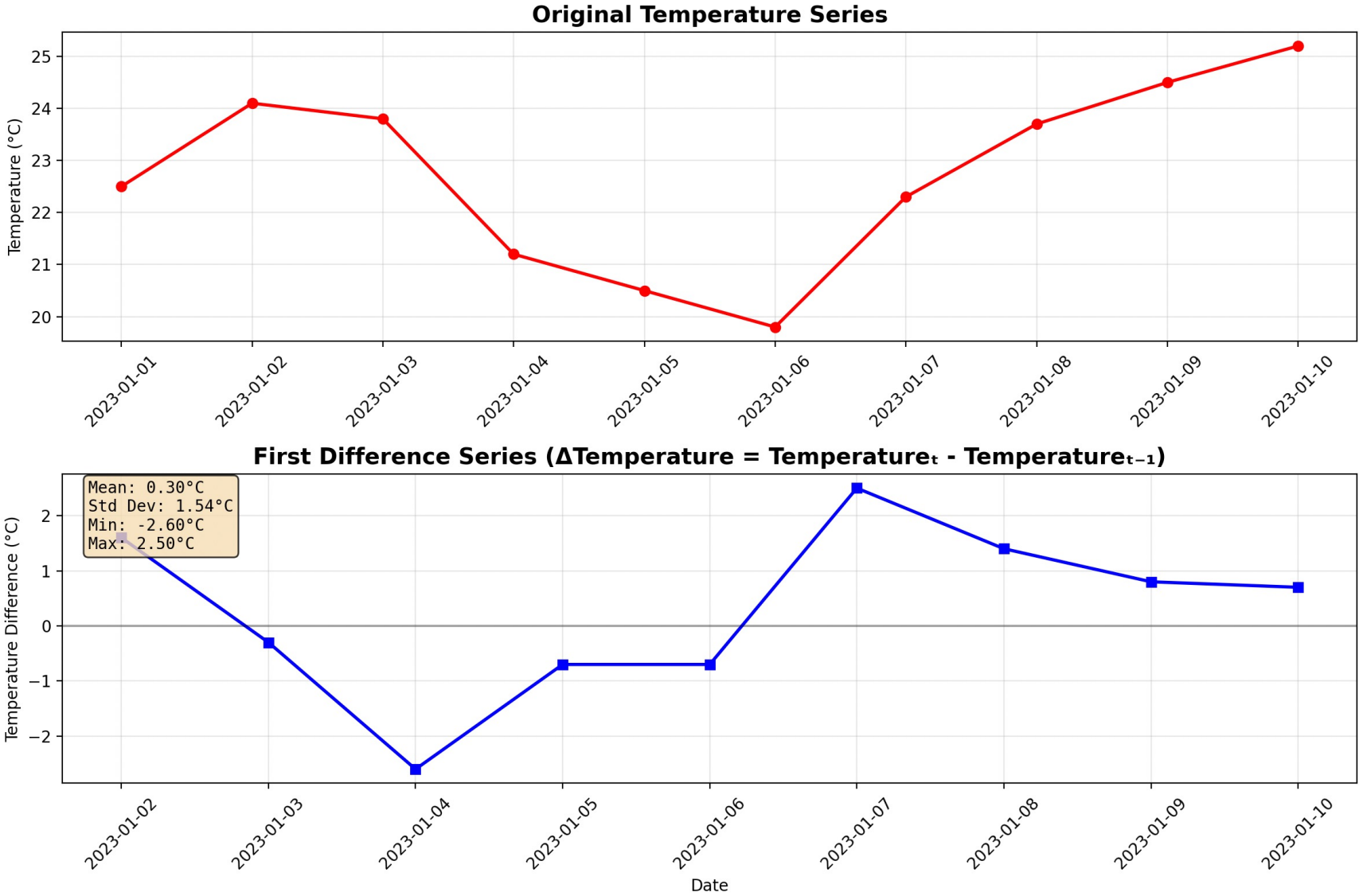
Method

```
# 绘制差分后的序列
ax2.plot(df.index[1:], df['Temperature_Diff'][1:], marker='s', color='blue', linewidth=2, m
arkersize=6)
ax2.axhline(y=0, color='black', linestyle='-', alpha=0.3)
ax2.set_title('First Difference Series (ΔTemperature = Temperature_t - Temperature_t-1)',
              fontsize=14, fontweight='bold')
ax2.set_ylabel('Temperature Difference (°C)')
ax2.set_xlabel('Date')
ax2.grid(True, alpha=0.3)
ax2.tick_params(axis='x', rotation=45)

# 添加统计信息框
diff_stats = f"Mean: {df['Temperature_Diff'].mean():.2f}°C\n"
diff_stats += f"Std Dev: {df['Temperature_Diff'].std():.2f}°C\n"
diff_stats += f"Min: {df['Temperature_Diff'].min():.2f}°C\n"
diff_stats += f"Max: {df['Temperature_Diff'].max():.2f}°C"

ax2.text(0.02, 0.98, diff_stats, transform=ax2.transAxes, verticalalignment='top',
         bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.8),
         fontfamily='monospace')

plt.tight_layout()
plt.show()
```

Method

```
# 打印差分结果
print("Temperature Difference Calculation:")
print("=" * 40)
for i in range(1, len(df)):
    print(f"Δ({df.index[i].strftime('%Y-%m-%d')}) = {df['Temperature'].iloc[i]:.1f} - {df
['Temperature'].iloc[i-1]:.1f} = {df['Temperature_Diff'].iloc[i]:.1f}°C")
```

Method

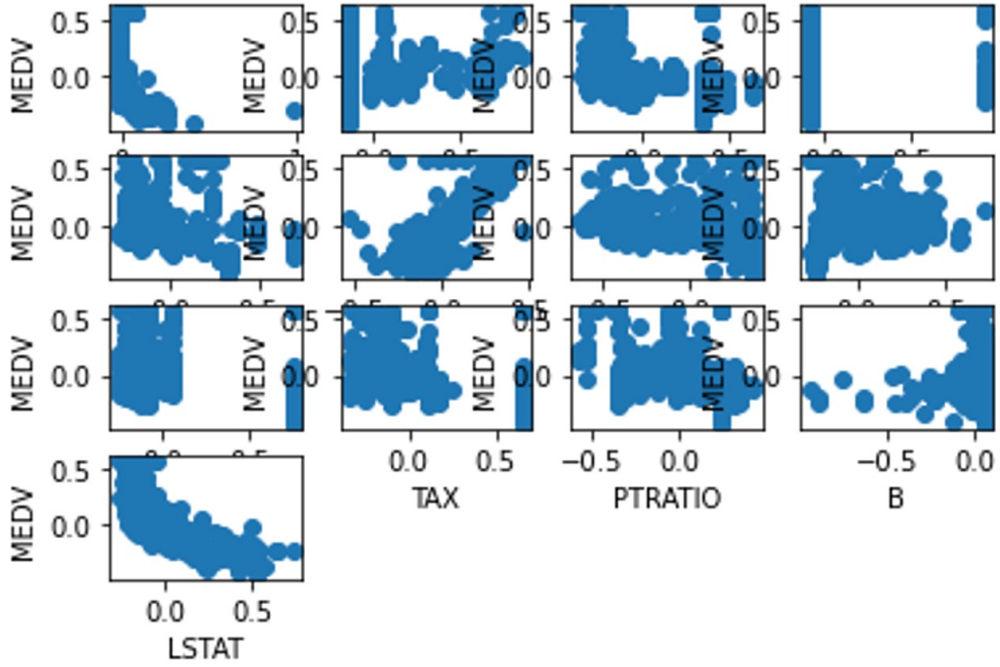

Original Temperature Series
Temperature (°C)
25
24
23
22
21
20
2023-01-01
2023-01-02
2023-01-03
2023-01-04
2023-01-05
2023-01-06
2023-01-07
2023-01-08
2023-01-09
2023-01-10
First Difference Series (ΔTemperature = Temperature$_t$ - Temperature$_{t-1}$)
Mean: 0.30°C
Std Dev: 1.54°C
Min: -2.60°C
Max: 2.50°C
Temperature Difference (°C)
2
1
0
−1
−2
2023-01-02
2023-01-03
2023-01-04
2023-01-05
2023-01-06
2023-01-07
2023-01-08
2023-01-09
2023-01-10
Date

- Data Preprocessing: An Overview 
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Reduction
- Data Transformation and Data Discretization
- Summary

Data Quality: Why Preprocess the Data?

- Measures for data quality: A multidimensional view
  - Accuracy: correct or wrong, accurate or not
  - Completeness: not recorded, unavailable, …
  - Consistency: some modified but some not, dangling, …
  - Timeliness: timely update?
  - Believability: how trustable the data are correct?
  - Interpretability: how easily the data can be understood?

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	target
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88	11.9



Major Tasks in Data Preprocessing

- **Data cleaning**
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- **Data integration**
  - Integration of multiple databases, data cubes, or files
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation

- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Reduction
- Data Transformation and Data Discretization
- Summary

Data Cleaning

- Data in the Real World Is Dirty: Lots of potentially incorrect data, e.g., instrument faulty, human or computer error, transmission error
  - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
    - e.g., *Occupation*=" " (missing data)
  - noisy: containing noise, errors, or outliers
    - e.g., *Salary*="−10" (an error)
  - inconsistent: containing discrepancies in codes or names, e.g.,
    - *Age*="42", *Birthday*="03/07/2010"
    - Was rating "1, 2, 3", now rating "A, B, C"
    - discrepancy between duplicate records
  - Intentional (e.g., *disguised missing* data)
    - Jan. 1 as everyone's birthday?

Incomplete (Missing) Data

- Data is not always available
  - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- Missing data may need to be inferred

Detecting (Missing) Data

0.00632	18.00	2.310	0	0.5380	6.5750	65.20	4.0900	1	296.0	15.30	39
0.02731	0.00	7.070	0	0.4690	6.4210	78.90	4.9671	2	242.0	17.80	39
0.02729	0.00	7.070	0	0.4690	7.1850	61.10	4.9671	2	242.0	17.80	39
0.03237	0.00	2.180	0	0.4580	6.9980	45.80	6.0622	3	222.0	18.70	39
0.06905	0.00	2.180	0	0.4580	7.1470	54.20	6.0622	3	222.0	18.70	39
0.02985	0.00	2.180	0	0.4580	6.4300	58.70	6.0622	3	222.0	18.70	39
0.08829	12.50	7.870	0	0.5240	6.0120	66.60	5.5605	5	311.0	15.20	39
0.14455	12.50	7.870	0	0.5240	6.1720	96.10	5.9505	5	311.0	15.20	39
0.21124	12.50	7.870	0	0.5240	5.6310	100.00	6.0821	5	311.0	15.20	38
0.17004	12.50	7.870	0	0.5240	6.0040	85.90	6.5921	5	311.0	15.20	38
0.22489	12.50	7.870	0	0.5240	6.3770	94.30	6.3467	5	311.0	15.20	39
0.11747	12.50	7.870	0	0.5240	6.0090	82.90	6.2267	5	311.0	15.20	39
0.09378	12.50	7.870	0	0.5240	5.8890	39.00	5.4509	5	311.0	15.20	39
0.62976	0.00	8.140	0	0.5380	5.9490	61.80	4.7075	4	307.0	21.00	39
0.63796	0.00	8.140	0	0.5380	6.0960	84.50	4.4619	4	307.0	21.00	38
0.62739	0.00	8.140	0	0.5380	5.8340	56.50	4.4986	4	307.0	21.00	39
1.05393	0.00	8.140	0	0.5380	5.9350	29.30	4.4986	4	307.0	21.00	38
0.78420	0.00	8.140	0	0.5380	5.9900	81.70	4.2579	4	307.0	21.00	38
0.80271	0.00	8.140	0	0.5380	5.4560	36.60	3.7965	4	307.0	21.00	28
0.72580	0.00	8.140	0	0.5380	5.7270	69.50	3.7965	4	307.0	21.00	39
1.25179	0.00	8.140	0	0.5380	5.5700	98.10	3.7979	4	307.0	21.00	37
0.85204	0.00	8.140	0	0.5380	5.9650	89.20	4.0123	4	307.0	21.00	39
1.23247	0.00	8.140	0	0.5380	6.1420	91.70	3.9769	4	307.0	21.00	39
0.98843	0.00	8.140	0	0.5380	5.8130	100.00	4.0952	4	307.0	21.00	39
0.75026	0.00	8.140	0	0.5380	5.9240	94.10	4.3996	4	307.0	21.00	39
0.84054	0.00	8.140	0	0.5380	5.5990	85.70	4.4546	4	307.0	21.00	30
0.67191	0.00	8.140	0	0.5380	5.8130	90.30	4.6820	4	307.0	21.00	37
0.95577	0.00	8.140	0	0.5380	6.0470	88.80	4.4534	4	307.0	21.00	30
0.77299	0.00	8.140	0	0.5380	6.4950	94.40	4.4547	4	307.0	21.00	38
1.00245	0.00	8.140	0	0.5380	6.6740	87.30	4.2390	4	307.0	21.00	38

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  target   506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
None
```

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing (when doing classification)—not effective when the % of missing values per attribute varies considerably
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant : e.g., "unknown", a new class?!
  - the attribute mean
  - the attribute mean for all samples belonging to the same class: smarter
  - the most probable value: inference-based such as Bayesian formula or decision tree

Noisy Data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may be due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - technology limitation
  - inconsistency in naming convention
- Other data problems which require data cleaning
  - duplicate records
  - incomplete data
  - inconsistent data

How to Handle Noisy Data?

- Binning
  - first sort data and partition into (equal-frequency) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Regression
  - smooth by fitting the data into regression functions
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)

Data Cleaning as a Process

- Data discrepancy detection
  - Use metadata (e.g., domain, range, dependency, distribution)
  - Check field overloading
  - Check uniqueness rule, consecutive rule and null rule
  - Use commercial tools
    - Data scrubbing: use simple domain knowledge (e.g., postal code, spell-check) to detect errors and make corrections
    - Data auditing: by analyzing data to discover rules and relationship to detect violators (e.g., correlation and clustering to find outliers)
- Data migration and integration
  - Data migration tools: allow transformations to be specified
  - ETL (Extraction/Transformation/Loading) tools: allow users to specify transformations through a graphical user interface
- Integration of the two processes
  - Iterative and interactive (e.g., Potter's Wheels)

- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration 
- Data Reduction
- Data Transformation and Data Discretization
- Summary

Data Integration

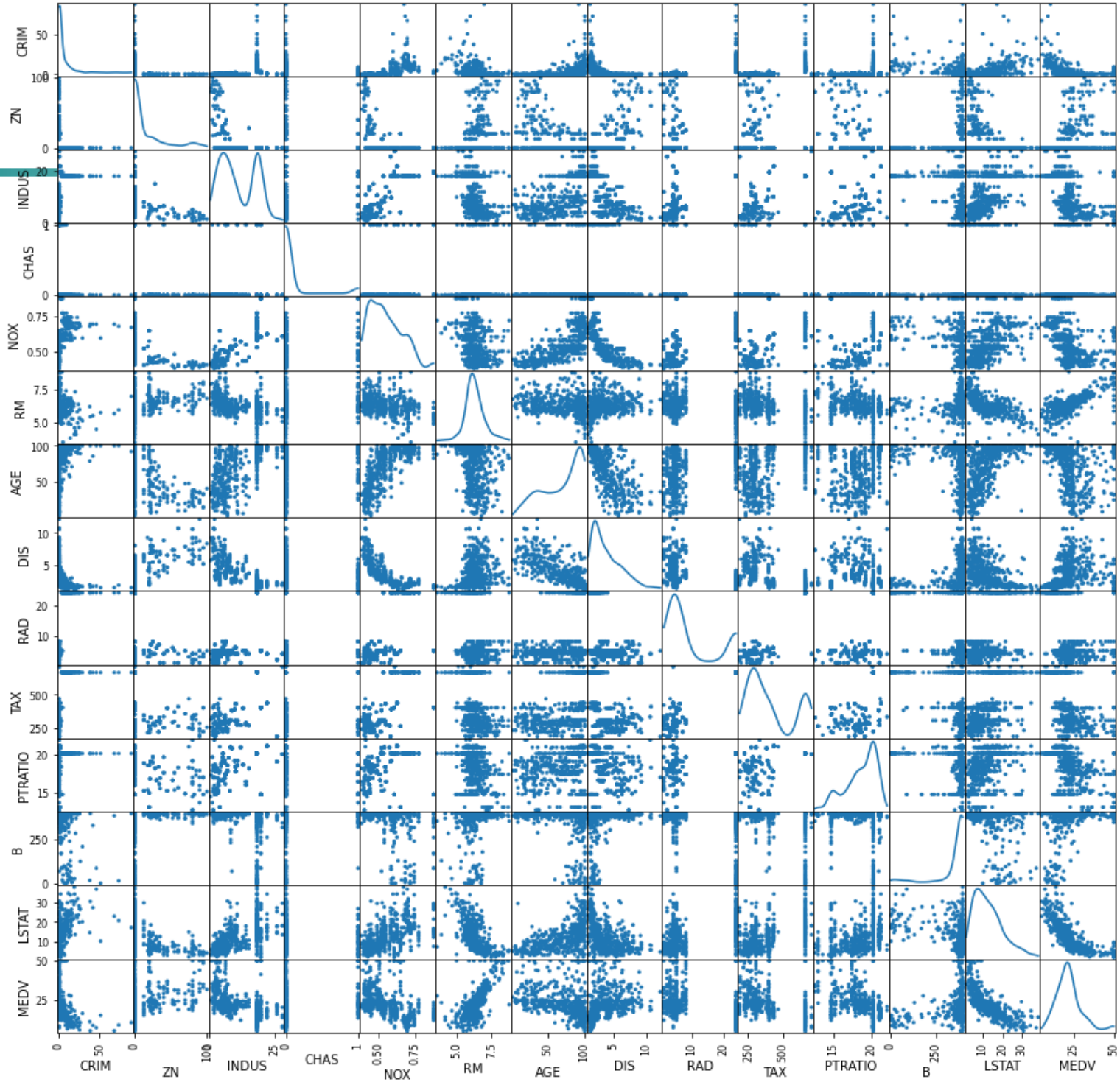
- **Data integration**:
  - Combines data from multiple sources into a coherent store
- Schema integration: e.g., A.cust-id ≡ B.cust-#
  - Integrate metadata from different sources
- Entity identification problem:
  - Identify real world entities from multiple data sources, e.g., Bill Clinton = William Clinton
- Detecting and resolving data value conflicts
  - For the same real world entity, attribute values from different sources are different
  - Possible reasons: different representations, different scales, e.g., metric vs. British units

Handling Redundancy in Data Integration

- Redundant data occur often when integration of multiple databases
  - *Object identification*: The same attribute or object may have different names in different databases
  - *Derivable data:* One attribute may be a "derived" attribute in another table, e.g., annual revenue
- Redundant attributes may be able to be detected by *correlation analysis* and *covariance analysis*
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Plot Correlation Matrix

```
# 读取数据
df = pd.read_csv('D:/poston/housing.csv', delim_whitespace=True, header=None)
# 设置列名
feature_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B',
'LSTAT', 'MEDV']
df.columns = feature_names
# 绘制相关系数矩阵
corr_matrix = df.corr()
plt.figure(figsize=(14, 14))
sns.heatmap(corr_matrix, annot=True, cmap=plt.cm.Reds)
plt.show()
```


CRIM
ZN
INDUS
CHAS
NOX
RM
AGE
DIS
RAD
TAX
PTRATIO
B
LSTAT
MEDV

Correlation Analysis (Nominal Data)

- **X^2 (chi-square) test**

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

- The larger the X^2 value, the more likely the variables are related
- The cells that contribute the most to the X^2 value are those whose actual count is very different from the expected count
- Correlation does not imply causality
  - # of hospitals and # of car-theft in a city are correlated
  - Both are causally linked to the third variable: population

Chi-Square Calculation: An Example

	Play chess	Not play chess	Sum (row)
Like science fiction	250(90)	200(360)	450
Not like science fiction	50(210)	1000(840)	1050
Sum(col.)	300	1200	1500

- X^2 (chi-square) calculation (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840} = 507.93$$

- It shows that like_science_fiction and play_chess are correlated in the group

Correlation Analysis (Numeric Data)

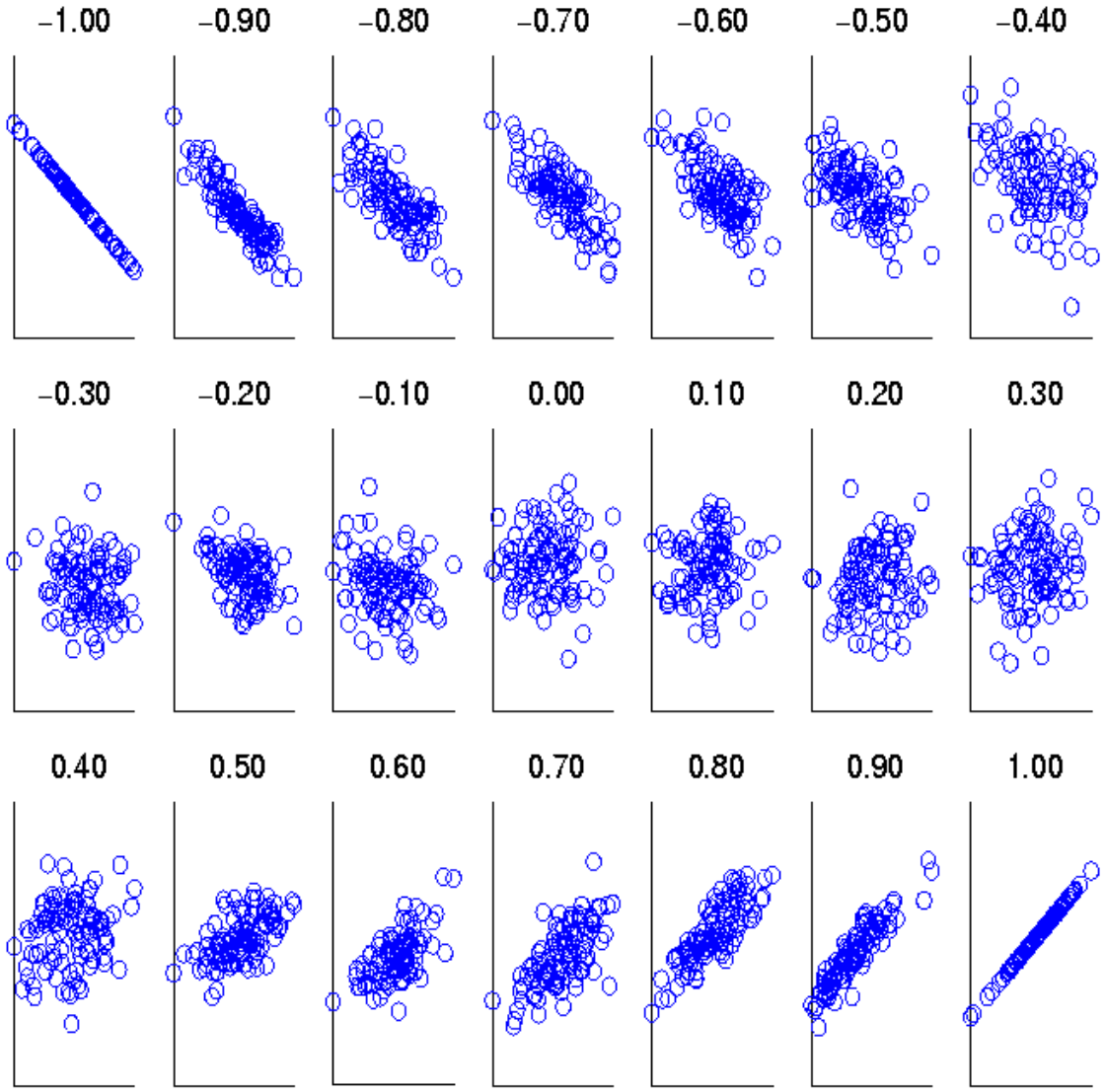
- Correlation coefficient (also called Pearson's product moment coefficient)

$$r_{A,B} = \frac{\sum_{i=1}^{n}(a_i - \overline{A})(b_i - \overline{B})}{(n-1)\sigma_A\sigma_B} = \frac{\sum_{i=1}^{n}(a_i b_i) - n\overline{A}\overline{B}}{(n-1)\sigma_A\sigma_B}$$

where n is the number of tuples, $\overline{A}$ and $\overline{B}$ are the respective means of A and B, σ_A and σ_B are the respective standard deviation of A and B, and $\Sigma(a_i b_i)$ is the sum of the AB cross-product.

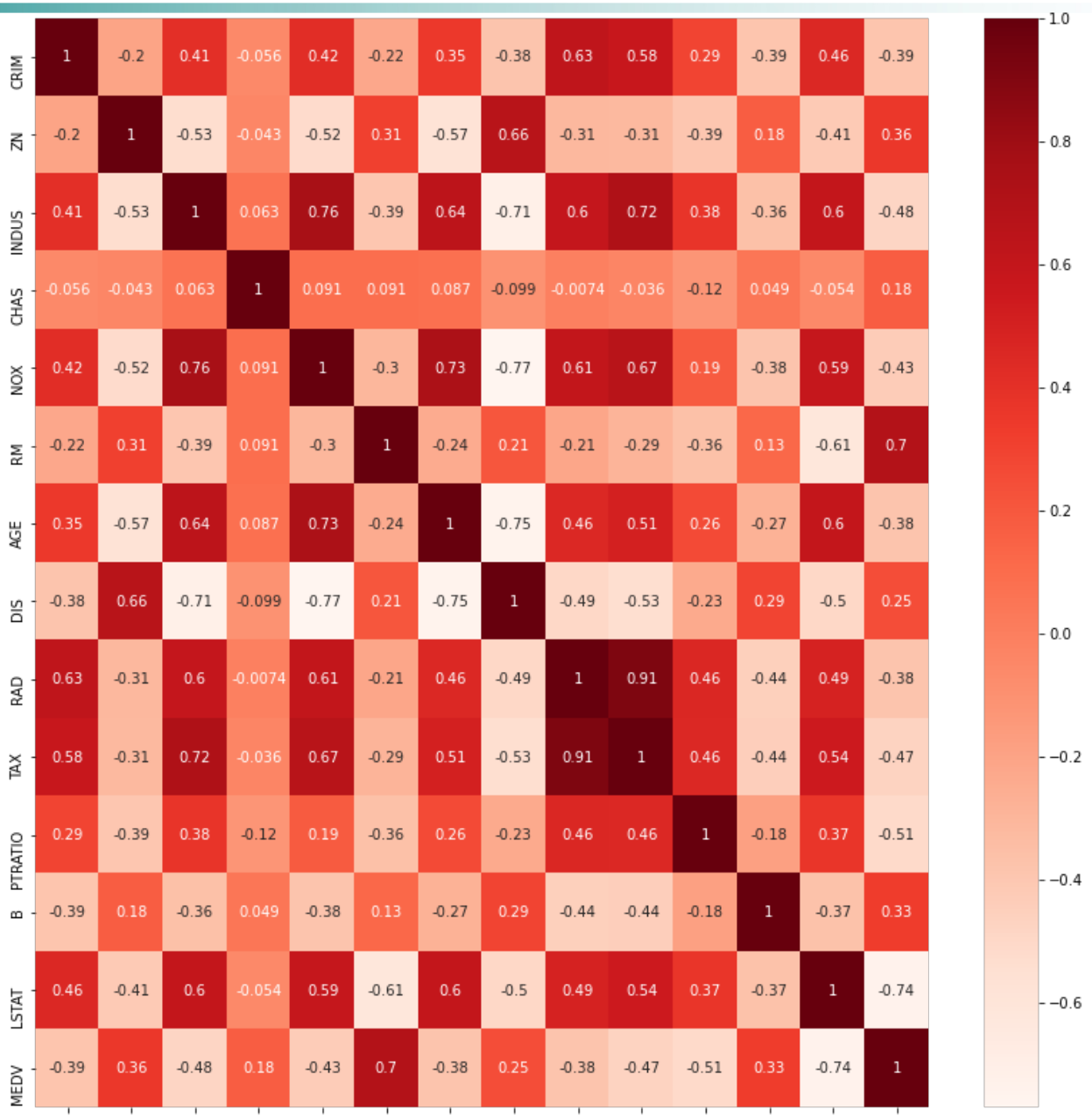
- If $r_{A,B} > 0$, A and B are positively correlated (A's values increase as B's). The higher, the stronger correlation.
- $r_{A,B} = 0$: independent; $r_{AB} < 0$: negatively correlated

Visually Evaluating Correlation



Scatter plots showing the similarity from –1 to 1.

Visually Correlation with Heatmap



Draw Conclusion from Correlations

- **Among the 13 features, the following pairs have a correlation coefficient greater than 0.5**

TAX	RAD	0.910228
INDUS	NOX	0.763651
NOX	AGE	0.731470
TAX	INDUS	0.720760
MEDV	RM	0.695360
TAX	NOX	0.668023
DIS	ZN	0.664408
INDUS	AGE	0.644779
CRIM	RAD	0.625505
NOX	RAD	0.611441
INDUS	LSTAT	0.603800
LSTAT	AGE	0.602339
INDUS	RAD	0.595129
NOX	LSTAT	0.590879
TAX	CRIM	0.582764
LSTAT	TAX	0.543993
TAX	AGE	0.506456

Correlation (viewed as linear relationship)

- Correlation measures the linear relationship between objects
- To compute correlation, we standardize data objects, A and B, and then take their dot product

$$a'_k = (a_k - mean(A)) / std(A)$$

$$b'_k = (b_k - mean(B)) / std(B)$$

$$correlation(A, B) = A' \bullet B'$$

Covariance (Numeric Data)

- Covariance is similar to correlation

$$Cov(A,B) = E((A-\bar{A})(B-\bar{B})) = \frac{\sum_{i=1}^{n}(a_i-\bar{A})(b_i-\bar{B})}{n}$$

Correlation coefficient: $$r_{A,B} = \frac{Cov(A,B)}{\sigma_A \sigma_B}$$

where n is the number of tuples, $\bar{A}$ and $\bar{B}$ are the respective mean or **expected values** of A and B, σ_A and σ_B are the respective standard deviation of A and B.

- **Positive covariance**: If $Cov_{A,B} > 0$, then A and B both tend to be larger than their expected values.
- **Negative covariance**: If $Cov_{A,B} < 0$ then if A is larger than its expected value, B is likely to be smaller than its expected value.
- **Independence**: $Cov_{A,B} = 0$ but the converse is not true:
  - Some pairs of random variables may have a covariance of 0 but are not independent. Only under some additional assumptions (e.g., the data follow multivariate normal distributions) does a covariance of 0 imply independence

Co-Variance: An Example

$$Cov(A, B) = E((A - \bar{A})(B - \bar{B})) = \frac{\sum_{i=1}^{n}(a_i - \bar{A})(b_i - \bar{B})}{n}$$

- It can be simplified in computation as

$$Cov(A, B) = E(A \cdot B) - \bar{A}\bar{B}$$

- Suppose two stocks A and B have the following values in one week: (2, 5), (3, 8), (5, 10), (4, 11), (6, 14).
- Question: If the stocks are affected by the same industry trends, will their prices rise or fall together?
  - E(A) = (2 + 3 + 5 + 4 + 6)/ 5 = 20/5 = 4
  - E(B) = (5 + 8 + 10 + 11 + 14) /5 = 48/5 = 9.6
  - Cov(A,B) = (2×5+3×8+5×10+4×11+6×14)/5 – 4 × 9.6 = 4
- Thus, A and B rise together since Cov(A, B) > 0.

- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Reduction 
- Data Transformation and Data Discretization
- Summary

Feature Creation

Use PolynomialFeatures for feature engineering and filter out the results with strong correlations

```
# 使用 PolynomialFeatures 进行特征构造
poly = PolynomialFeatures(degree=2, include_bias=False)
data_poly = pd.DataFrame(poly.fit_transform(data), columns=poly.get_feature_names(data.columns))

# 合并构造出的特征和原始特征
data_all = pd.concat([data, data_poly], axis=1)

# 计算相关性
corr_matrix = data_all.corr().abs()

# 找到相关性大于 0.9 的特征对
high_corr = np.where(corr_matrix > 0.9)
high_corr = [(corr_matrix.columns[x], corr_matrix.columns[y]) for x, y in zip(*high_corr) if x != y and x < y]
print(high_corr)
```

Feature Creation

Use PolynomialFeatures for feature engineering and filter out the results with strong correlations

```
# 使用 PolynomialFeatures 进行特征构造
poly = PolynomialFeatures(degree=2, include_bias=False)
data_poly = pd.DataFrame(poly.fit_transform(data), columns=poly.get_feature_names(data.columns))

# 合并构造出的特征和原始特征
data_all = pd.concat([data, data_poly], axis=1)

# 计算相关性
corr_matrix = data_all.corr().abs()

# 找到相关性大于 0.9 的特征对
high_corr = np.where(corr_matrix > 0.9)
high_corr = [(corr_matrix.columns[x], corr_matrix.columns[y]) for x, y in zip(*high_corr) if x != y and x < y]
print(high_corr)
```

Feature Creation

('INDUS^2', 'INDUS NOX'), ('INDUS^2', 'INDUS RM'), ('INDUS^2', 'INDUS AGE'), ('INDUS^2', 'INDUS PTRATIO'), ('INDUS CHAS', 'CHAS^2'), ('INDUS CHAS', 'CHAS NOX'), ('INDUS CHAS', 'CHAS AGE'), ('INDUS CHAS', 'CHAS TAX'), ('INDUS NOX', 'INDUS RM'), ('INDUS NOX', 'INDUS AGE'), ('INDUS NOX', 'INDUS TAX'), ('INDUS NOX', 'INDUS PTRATIO'), ('INDUS RM', 'INDUS AGE'), ('INDUS RM', 'INDUS PTRATIO'), ('INDUS AGE', 'INDUS PTRATIO'), ('INDUS RAD', 'NOX RAD'), ('INDUS RAD', 'NOX TAX'), ('INDUS RAD', 'RM RAD'), ('INDUS RAD', 'AGE RAD'), ('INDUS RAD', 'RAD^2'), ('INDUS RAD', 'RAD TAX'), ('INDUS RAD', 'RAD PTRATIO'), ('INDUS RAD', 'TAX^2'), ('INDUS RAD', 'TAX PTRATIO'), ('INDUS TAX', 'INDUS PTRATIO'), ('INDUS TAX', 'NOX TAX'), ('INDUS TAX', 'AGE TAX'), ('INDUS TAX', 'TAX^2'), ('INDUS TAX', 'TAX PTRATIO'), ('INDUS LSTAT', 'NOX LSTAT'), ('INDUS LSTAT', 'AGE LSTAT'), ('INDUS LSTAT', 'TAX LSTAT'), ('CHAS^2', 'CHAS NOX'), ('CHAS^2', 'CHAS RM'), ('CHAS^2', 'CHAS AGE'), ('CHAS^2', 'CHAS DIS'), ('CHAS^2', 'CHAS TAX'), ('CHAS^2', 'CHAS PTRATIO'), ('CHAS^2', 'CHAS B'), ('CHAS NOX', 'CHAS RM'), ('CHAS NOX', 'CHAS AGE'), ('CHAS NOX', 'CHAS TAX'), ('CHAS NOX', 'CHAS PTRATIO'), ('CHAS NOX', 'CHAS B'), ('CHAS RM', 'CHAS AGE'), ('CHAS RM', 'CHAS DIS'), ('CHAS RM', 'CHAS TAX'), ('CHAS RM', 'CHAS PTRATIO'), ('CHAS RM', 'CHAS B'), ('CHAS AGE', 'CHAS TAX'), ('CHAS AGE', 'CHAS PTRATIO'), ('CHAS AGE', 'CHAS B'), ('CHAS DIS', 'CHAS PTRATIO'), ('CHAS DIS', 'CHAS B'), ('CHAS RAD', 'CHAS TAX'), ('CHAS TAX', 'CHAS PTRATIO'), ('CHAS PTRATIO', 'CHAS B'), ('NOX AGE', 'AGE^2'), ('NOX DIS', 'RM DIS'), ('NOX DIS', 'DIS^2'), ('NOX DIS', 'DIS PTRATIO'), ('NOX DIS', 'DIS B'), ('NOX RAD', 'NOX TAX'), ('NOX RAD', 'RM RAD'), ('NOX RAD', 'AGE RAD'), ('NOX RAD', 'RAD^2'), ('NOX RAD', 'RAD TAX'), ('NOX RAD', 'RAD PTRATIO'), ('NOX RAD', 'TAX^2'), ('NOX RAD', 'TAX PTRATIO'), ('NOX TAX', 'RM TAX'), ('NOX TAX', 'AGE RAD'), ('NOX TAX', 'AGE TAX'), ('NOX

Feature Creation

Alternative Approach: Statistical Feature Engineering. This method creates new features by computing statistical measures (e.g., mean, standard deviation, max, min) from original features. It helps capture non-linear relationships and enhances model performance. Code implementation:

```
Index(['RM', 'RM^2', 'RM^3', 'CRIM', 'log(CRIM)', 'B'], dtype='object')
```

Feature Creation

```
# 使用 PolynomialFeatures 进行特征构造
poly = PolynomialFeatures(degree=2, include_bias=False)
data_poly = pd.DataFrame(poly.fit_transform(data), columns=poly.get_feature_names(data.columns))

# 合并构造出的特征和原始特征
data_all = pd.concat([data, data_poly], axis=1)

# 计算相关性
corr_matrix = data_all.corr().abs()

# 找到相关性大于 0.9 的特征对
high_corr = np.where(corr_matrix > 0.9)
high_corr = [(corr_matrix.columns[x], corr_matrix.columns[y]) for x, y in zip(*high_corr) if x != y and x < y]
print(high_corr)
```

```
Index(['RM', 'RM^2', 'RM^3', 'CRIM', 'log(CRIM)', 'B'], dtype='object')
```

Feature Creation

Method 3: Feature Engineering via Arithmetic Operations. This approach generates new features by applying mathematical operations to existing features. We selected several features that demonstrated potential correlations and were suitable for such transformations to construct the new features.

Feature Creation

```
# 构造新特征
data['RM_LSTAT'] = data['RM'] * data['LSTAT']    # 房间数乘低收入人群比例
data['RM_PTRATIO'] = data['RM'] / data['PTRATIO']    # 房间数除以学生与教师比例
data['RM_TAX'] = data['RM'] / data['TAX']    # 房间数除以房产税
data['AGE_PTRATIO'] = data['AGE'] / data['PTRATIO']    # 楼龄除以学生与教师比例
data['INDUS_LSTAT'] = data['INDUS'] * data['LSTAT']    # 行业比例乘低收入人群比例

# 计算每个新特征与房价的相关系数
corrs = data.corr()['MEDV'].abs().sort_values(ascending=False)
selected_features = corrs[corrs >= 0.5].index.tolist()
selected_features.remove('MEDV')    # 移除目标变量

# 打印相关系数较高的特征
print(selected_features)
```

```
['LSTAT', 'RM_LSTAT', 'RM_PTRATIO', 'RM', 'INDUS_LSTAT', 'RM_TAX', 'PTRATIO']
```

Data Reduction Strategies

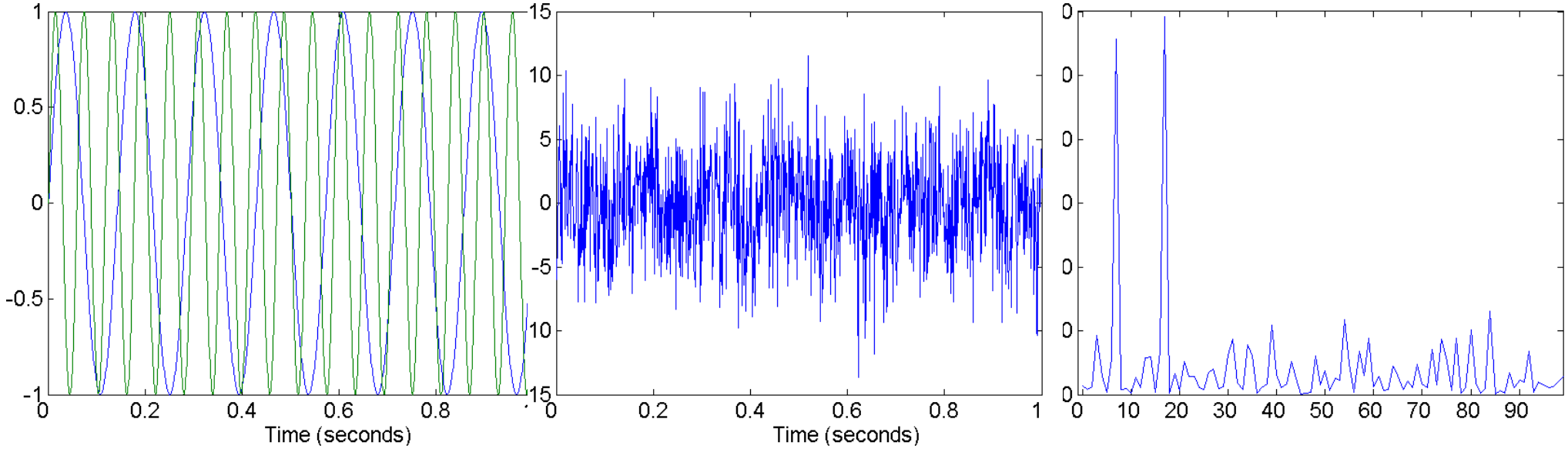
- **Data reduction**: Obtain a reduced representation of the data set that is much smaller in volume but yet produces the same (or almost the same) analytical results
- Why data reduction? — A database/data warehouse may store terabytes of data. Complex data analysis may take a very long time to run on the complete data set.
- Data reduction strategies
  - Dimensionality reduction, e.g., remove unimportant attributes
    - Wavelet transforms
    - Principal Components Analysis (PCA)
    - Feature subset selection, feature creation
  - Numerosity reduction (some simply call it: Data Reduction)
    - Regression and Log-Linear Models
    - Histograms, clustering, sampling
    - Data cube aggregation
  - Data compression

Data Reduction 1: Dimensionality Reduction

- **Curse of dimensionality**
  - When dimensionality increases, data becomes increasingly sparse
  - Density and distance between points, which is critical to clustering, outlier analysis, becomes less meaningful
  - The possible combinations of subspaces will grow exponentially
- **Dimensionality reduction**
  - Avoid the curse of dimensionality
  - Help eliminate irrelevant features and reduce noise
  - Reduce time and space required in data mining
  - Allow easier visualization
- **Dimensionality reduction techniques**
  - Wavelet transforms
  - Principal Component Analysis
  - Supervised and nonlinear techniques (e.g., feature selection)

Mapping Data to a New Space

- Fourier transform
- Wavelet transform



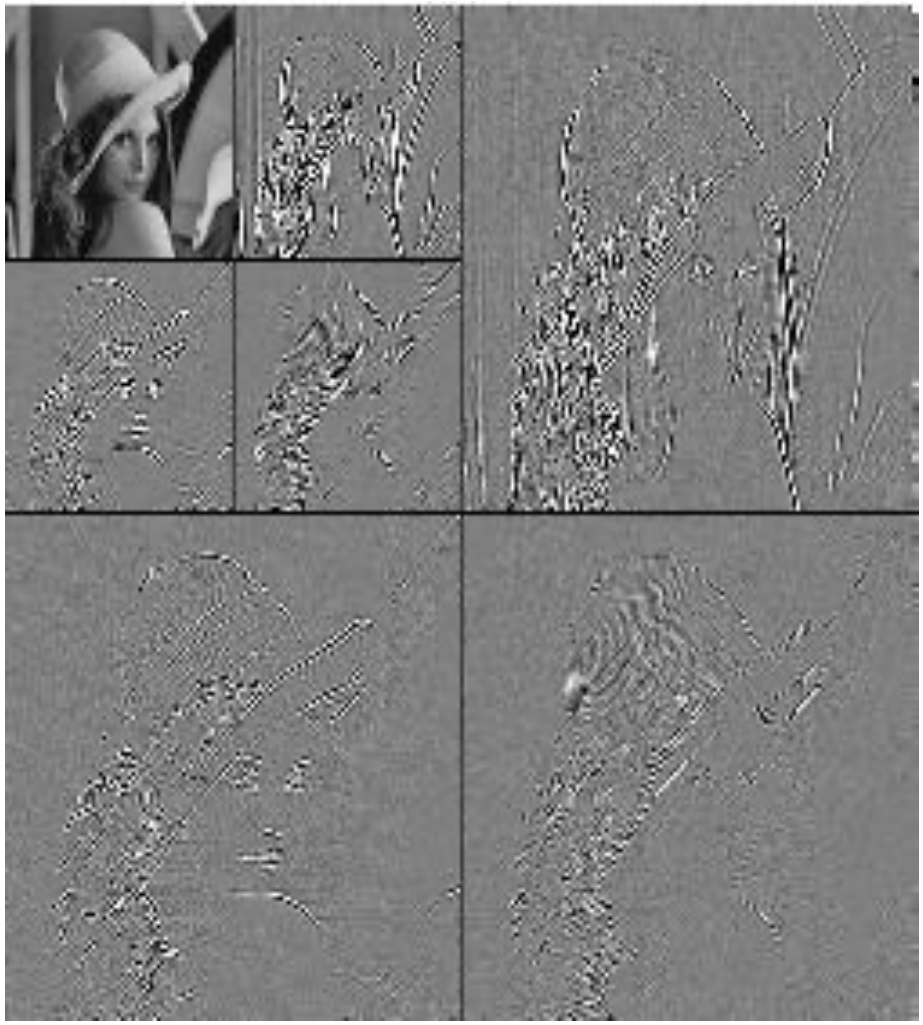
Two Sine Waves

Two Sine Waves + Noise

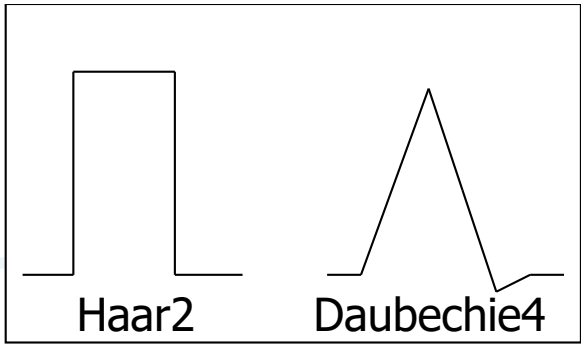
Frequency

What Is Wavelet Transform?

- Decomposes a signal into different frequency subbands
  - Applicable to n-dimensional signals
- Data are transformed to preserve relative distance between objects at different levels of resolution
- Allow natural clusters to become more distinguishable
- Used for image compression



Wavelet Transformation



- Discrete wavelet transform (DWT) for linear signal processing, multi-resolution analysis
- Compressed approximation: store only a small fraction of the strongest of the wavelet coefficients
- Similar to discrete Fourier transform (DFT), but better lossy compression, localized in space
- Method:
  - Length, L, must be an integer power of 2 (padding with 0's, when necessary)
  - Each transform has 2 functions: smoothing, difference
  - Applies to pairs of data, resulting in two set of data of length L/2
  - Applies two functions recursively, until reaches the desired length

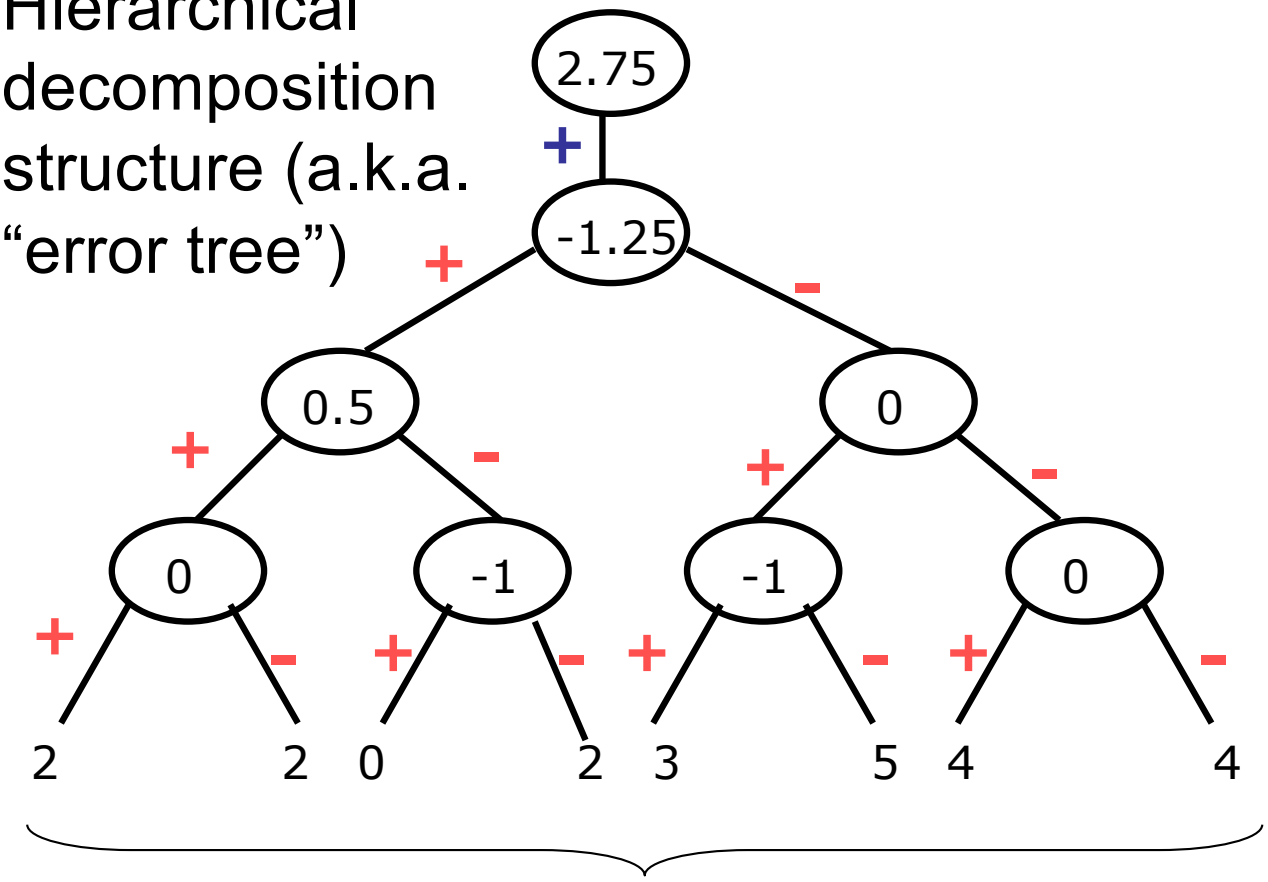
Wavelet Decomposition

- Wavelets: A math tool for space-efficient hierarchical decomposition of functions
- S = [2, 2, 0, 2, 3, 5, 4, 4] can be transformed to $S^\wedge$ = [$2^3/_4$, $-1^1/_4$, $^1/_2$, 0, 0, -1, -1, 0]
- Compression: many small detail coefficients can be replaced by 0's, and only the significant coefficients are retained

Resolution	Averages	Detail Coefficients
8	$[2, 2, 0, 2, 3, 5, 4, 4]$	
4	$[2, 1, 4, 4]$	$[0, -1, -1, 0]$
2	$[1\frac{1}{2}, 4]$	$[\frac{1}{2}, 0]$
1	$[2\frac{3}{4}]$	$[-1\frac{1}{4}]$

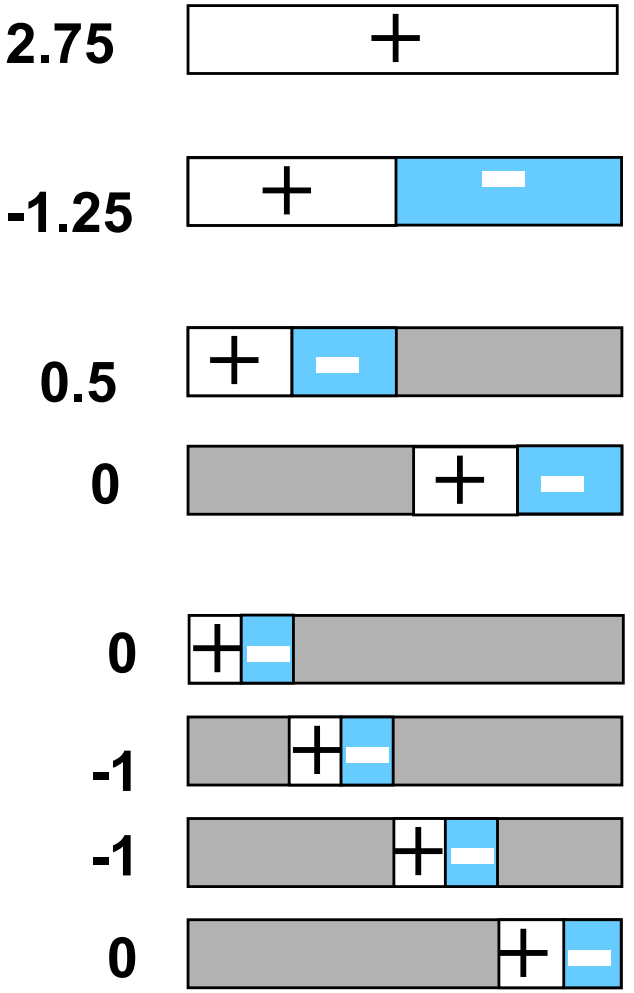
Haar Wavelet Coefficients

Hierarchical decomposition structure (a.k.a. "error tree")



Original frequency distribution

Coefficient "Supports"

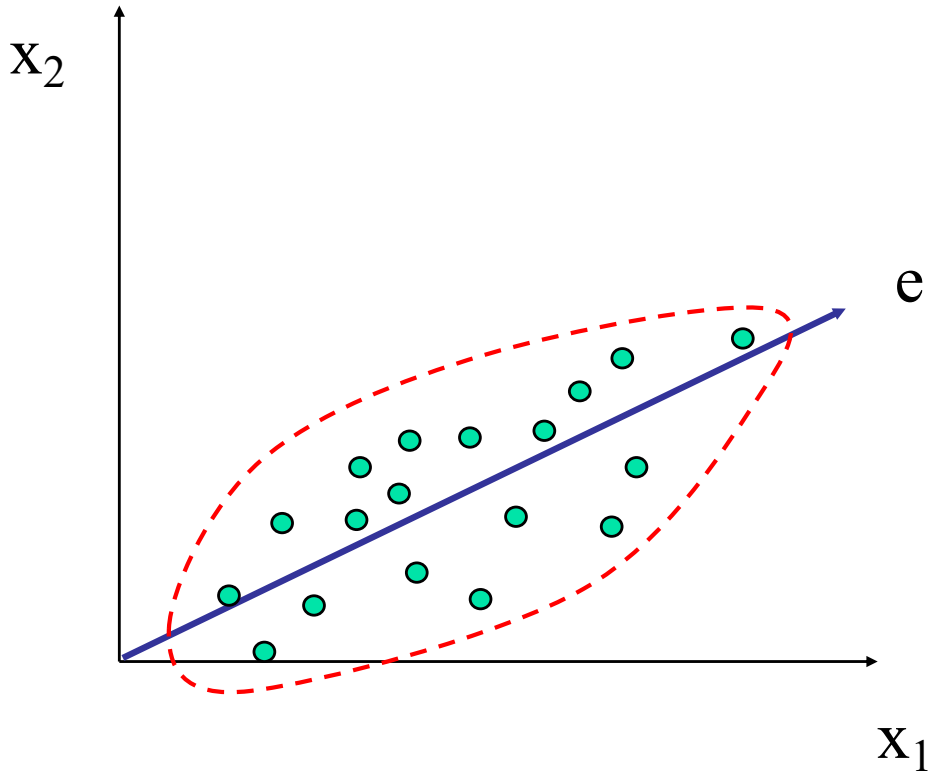


Why Wavelet Transform?

- Use hat-shape filters
  - Emphasize region where points cluster
  - Suppress weaker information in their boundaries
- Effective removal of outliers
  - Insensitive to noise, insensitive to input order
- Multi-resolution
  - Detect arbitrary shaped clusters at different scales
- Efficient
  - Complexity O(N)
- Only applicable to low dimensional data

Principal Component Analysis (PCA)

- Find a projection that captures the largest amount of variation in data
- The original data are projected onto a much smaller space, resulting in dimensionality reduction. We find the eigenvectors of the covariance matrix, and these eigenvectors define the new space



Principal Component Analysis (Steps)

- Given *N* data vectors from *n*-dimensions, find $k \leq n$ orthogonal vectors (*principal components*) that can be best used to represent data
  - Normalize input data: Each attribute falls within the same range
  - Compute *k* orthonormal (unit) vectors, i.e., *principal components*
  - Each input data (vector) is a linear combination of the *k* principal component vectors
  - The principal components are sorted in order of decreasing "significance" or strength
  - Since the components are sorted, the size of the data can be reduced by eliminating the *weak components*, i.e., those with low variance (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)
- Works for numeric data only

Attribute Subset Selection

- Another way to reduce dimensionality of data
- Redundant attributes
  - Duplicate much or all of the information contained in one or more other attributes
  - E.g., purchase price of a product and the amount of sales tax paid
- Irrelevant attributes
  - Contain no information that is useful for the data mining task at hand
  - E.g., students' ID is often irrelevant to the task of predicting students' GPA

Heuristic Search in Attribute Selection

- There are 2^d possible attribute combinations of d attributes
- Typical heuristic attribute selection methods:
  - Best single attribute under the attribute independence assumption: choose by significance tests
  - Best step-wise feature selection:
    - The best single-attribute is picked first
    - Then next best attribute condition to the first, ...
  - Step-wise attribute elimination:
    - Repeatedly eliminate the worst attribute
  - Best combined attribute selection and elimination
  - Optimal branch and bound:
    - Use attribute elimination and backtracking

Attribute Creation (Feature Generation)

- Create new attributes (features) that can capture the important information in a data set more effectively than the original ones
- Three general methodologies
  - Attribute extraction
    - Domain-specific
  - Mapping data to new space (see: data reduction)
    - E.g., Fourier transformation, wavelet transformation, manifold approaches (not covered)
  - Attribute construction
    - Combining features (see: discriminative frequent patterns in Chapter 7)
    - Data discretization

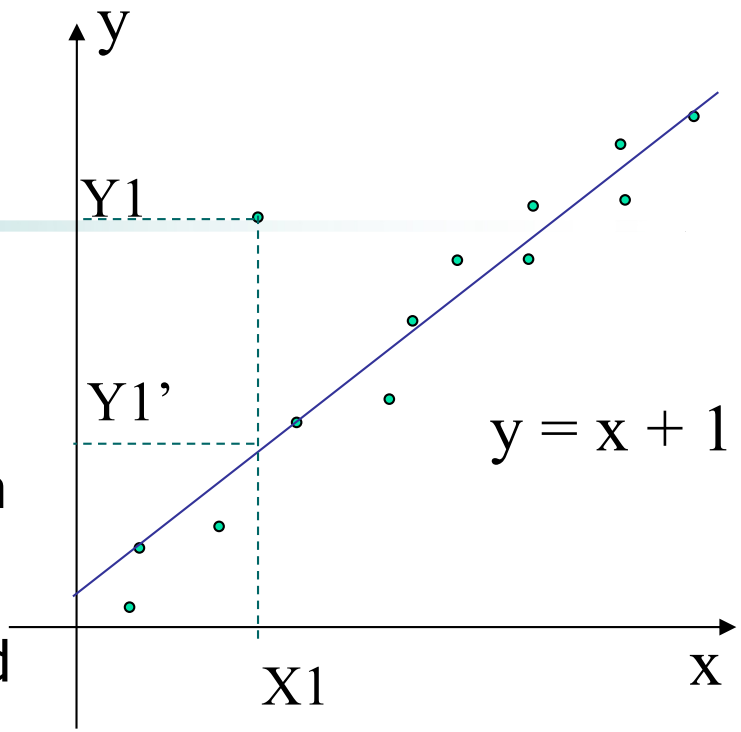
Data Reduction 2: Numerosity Reduction

- Reduce data volume by choosing alternative, *smaller forms* of data representation
- **Parametric methods** (e.g., regression)
  - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
  - Ex.: Log-linear models—obtain value at a point in *m*-D space as the product on appropriate marginal subspaces
- **Non-parametric** methods
  - Do not assume models
  - Major families: histograms, clustering, sampling, …

Parametric Data Reduction: Regression and Log-Linear Models

- **Linear regression**
  - Data modeled to fit a straight line
  - Often uses the least-square method to fit the line
- **Multiple regression**
  - Allows a response variable Y to be modeled as a linear function of multidimensional feature vector
- **Log-linear model**
  - Approximates discrete multidimensional probability distributions

Regression Analysis



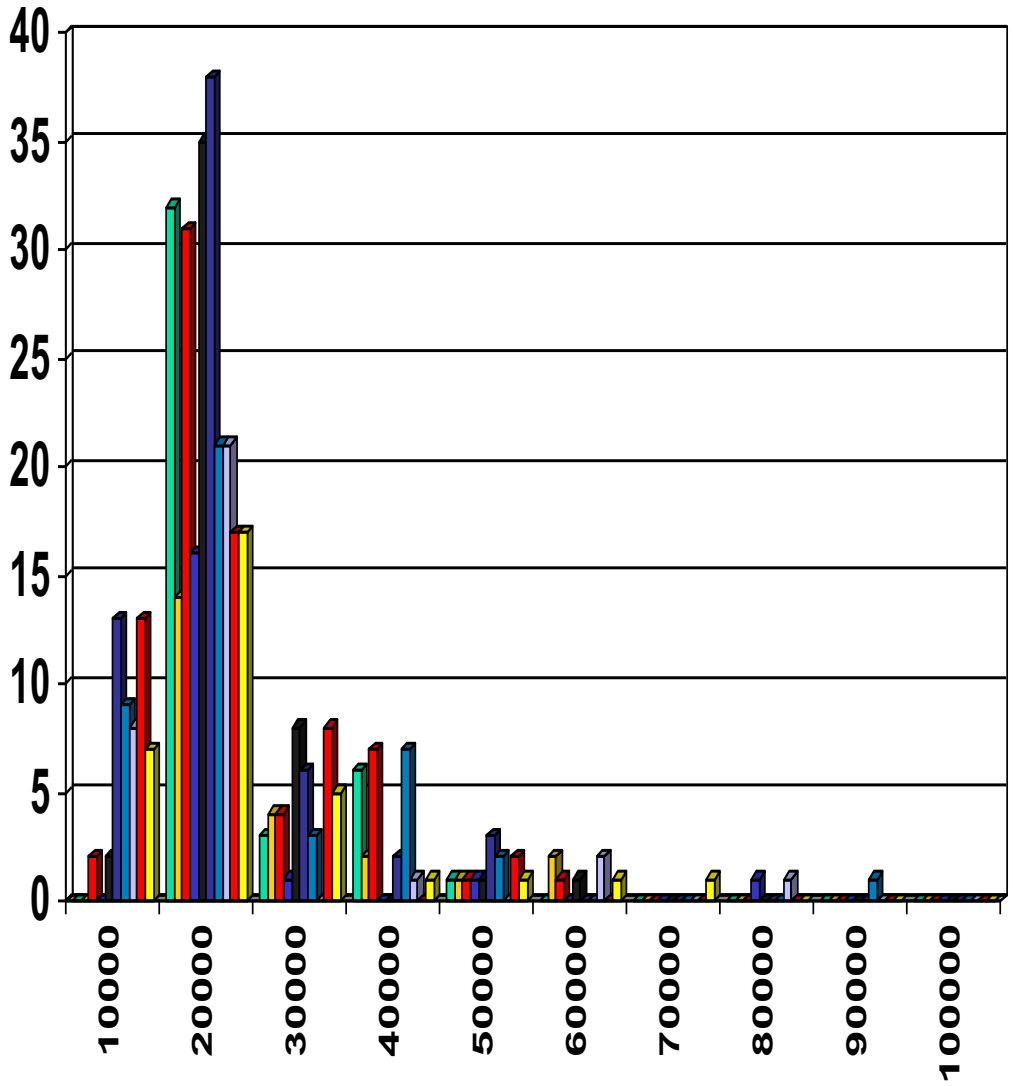
- Regression analysis: A collective name for techniques for the modeling and analysis of numerical data consisting of values of a ***dependent variable*** (also called ***response variable*** or *measurement*) and of one or more *independent variables* (aka. ***explanatory variables*** or ***predictors***)
- The parameters are estimated so as to give a "**best fit**" of the data
- Most commonly the best fit is evaluated by using the ***least squares method***, but other criteria have also been used
- Used for prediction (including forecasting of time-series data), inference, hypothesis testing, and modeling of causal relationships

Regress Analysis and Log-Linear Models

- Linear regression: $Y = w X + b$
  - Two regression coefficients, w and b, specify the line and are to be estimated by using the data at hand
  - Using the least squares criterion to the known values of $Y_1, Y_2, \ldots,$ $X_1, X_2, \ldots.$
- Multiple regression: $Y = b_0 + b_1 X_1 + b_2 X_2$
  - Many nonlinear functions can be transformed into the above
- Log-linear models:
  - Approximate discrete multidimensional probability distributions
  - Estimate the probability of each point (tuple) in a multi-dimensional space for a set of discretized attributes, based on a smaller subset of dimensional combinations
  - Useful for dimensionality reduction and data smoothing

Histogram Analysis

- Divide data into buckets and store average (sum) for each bucket
- Partitioning rules:
  - Equal-width: equal bucket range
  - Equal-frequency (or equal-depth)



Clustering

- Partition data set into clusters based on similarity, and store cluster representation (e.g., centroid and diameter) only
- Can be very effective if data is clustered but not if data is "smeared"
- Can have hierarchical clustering and be stored in multi-dimensional index tree structures
- There are many choices of clustering definitions and clustering algorithms
- Cluster analysis will be studied in depth in Chapter 10

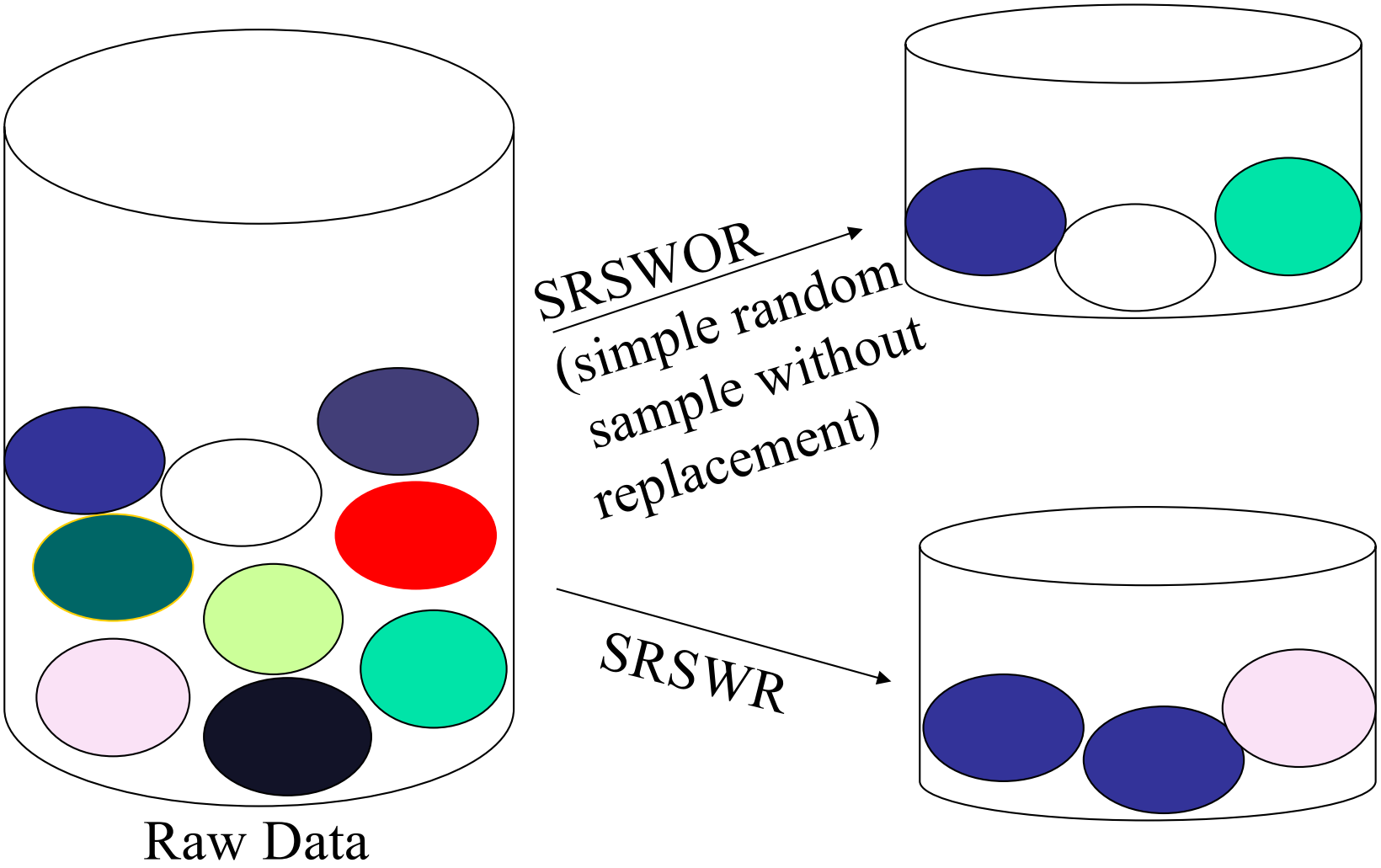
Sampling

- Sampling: obtaining a small sample *s* to represent the whole data set *N*
- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Key principle: Choose a representative subset of the data
  - Simple random sampling may have very poor performance in the presence of skew
  - Develop adaptive sampling methods, e.g., stratified sampling:
- Note: Sampling may not reduce database I/Os (page at a time)

Types of Sampling

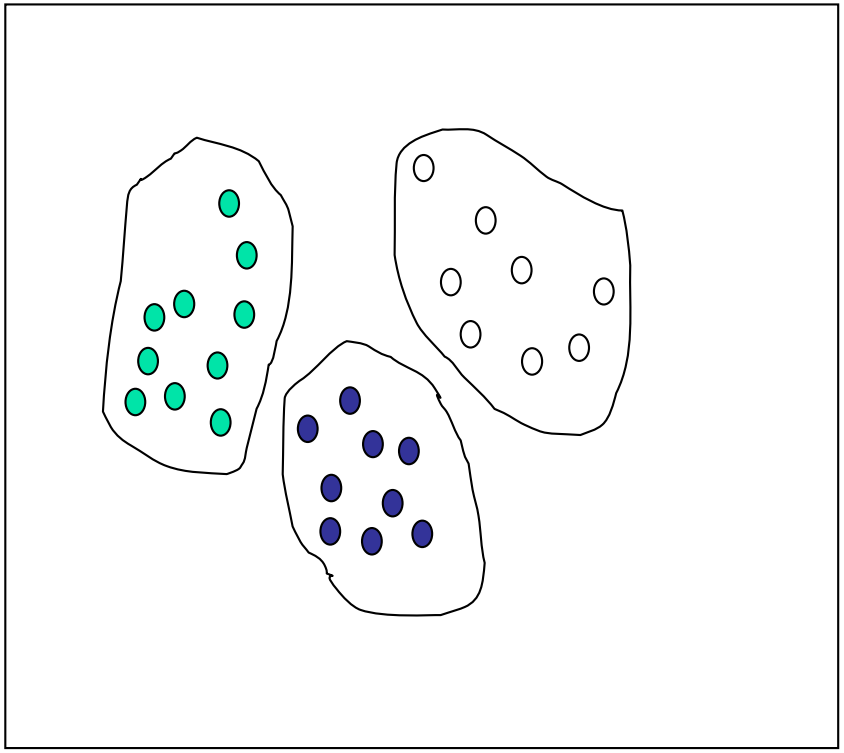
- **Simple random sampling**
  - There is an equal probability of selecting any particular item
- **Sampling without replacement**
  - Once an object is selected, it is removed from the population
- **Sampling with replacement**
  - A selected object is not removed from the population
- **Stratified sampling:**
  - Partition the data set, and draw samples from each partition (proportionally, i.e., approximately the same percentage of the data)
  - Used in conjunction with skewed data

Sampling: With or without Replacement

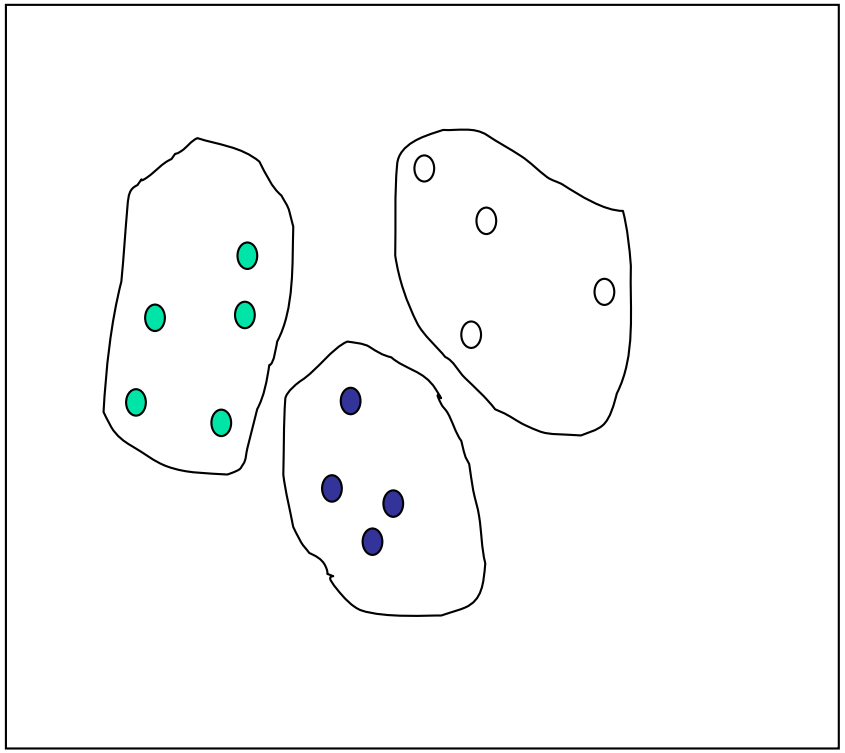


Sampling: Cluster or Stratified Sampling

Raw Data



Cluster/Stratified Sample



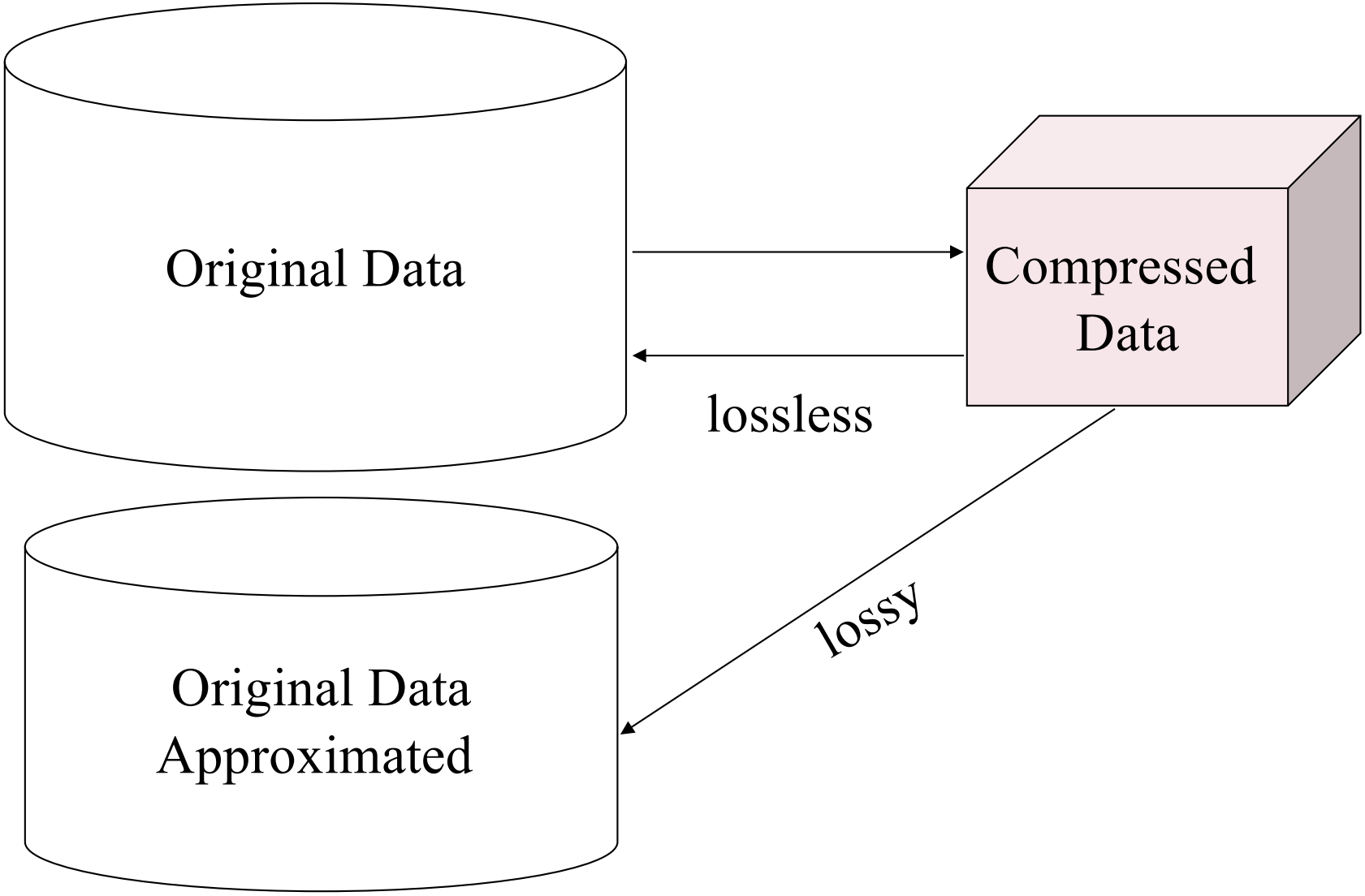
Data Cube Aggregation

- The lowest level of a data cube (base cuboid)
  - The aggregated data for an individual entity of interest
  - E.g., a customer in a phone calling data warehouse
- Multiple levels of aggregation in data cubes
  - Further reduce the size of data to deal with
- Reference appropriate levels
  - Use the smallest representation which is enough to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible

Data Reduction 3: Data Compression

- String compression
  - There are extensive theories and well-tuned algorithms
  - Typically lossless, but only limited manipulation is possible without expansion
- Audio/video compression
  - Typically lossy compression, with progressive refinement
  - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
  - Typically short and vary slowly with time
- Dimensionality and numerosity reduction may also be considered as forms of data compression

Data Compression



- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Reduction
- Data Transformation and Data Discretization 
- Summary

Data Transformation

- A function that maps the entire set of values of a given attribute to a new set of replacement values s.t. each old value can be identified with one of the new values
- Methods
  - Smoothing: Remove noise from data
  - Attribute/feature construction
    - New attributes constructed from the given ones
  - Aggregation: Summarization, data cube construction
  - Normalization: Scaled to fall within a smaller, specified range
    - min-max normalization
    - z-score normalization
    - normalization by decimal scaling
  - Discretization: Concept hierarchy climbing

Normalization

- **Min-max normalization**: to [new_min$_A$, new_max$_A$]

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

  - Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,000 is mapped to $\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$

- **Z-score normalization** (μ: mean, σ: standard deviation):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  - Ex. Let μ = 54,000, σ = 16,000. Then $\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$

- **Normalization by decimal scaling**

$$v' = \frac{v}{10^j} \quad \text{Where } j \text{ is the smallest integer such that Max}(|v'|) < 1$$

Discretization

- Three types of attributes
  - Nominal—values from an unordered set, e.g., color, profession
  - Ordinal—values from an ordered set, e.g., military or academic rank
  - Numeric—real numbers, e.g., integer or real numbers
- Discretization: Divide the range of a continuous attribute into intervals
  - Interval labels can then be used to replace actual data values
  - Reduce data size by discretization
  - Supervised vs. unsupervised
  - Split (top-down) vs. merge (bottom-up)
  - Discretization can be performed recursively on an attribute
  - Prepare for further analysis, e.g., classification

Data Discretization Methods

- Typical methods: All the methods can be applied recursively
  - Binning
    - Top-down split, unsupervised
  - Histogram analysis
    - Top-down split, unsupervised
  - Clustering analysis (unsupervised, top-down split or bottom-up merge)
  - Decision-tree analysis (supervised, top-down split)
  - Correlation (e.g., χ^2) analysis (unsupervised, bottom-up merge)

Simple Discretization: Binning

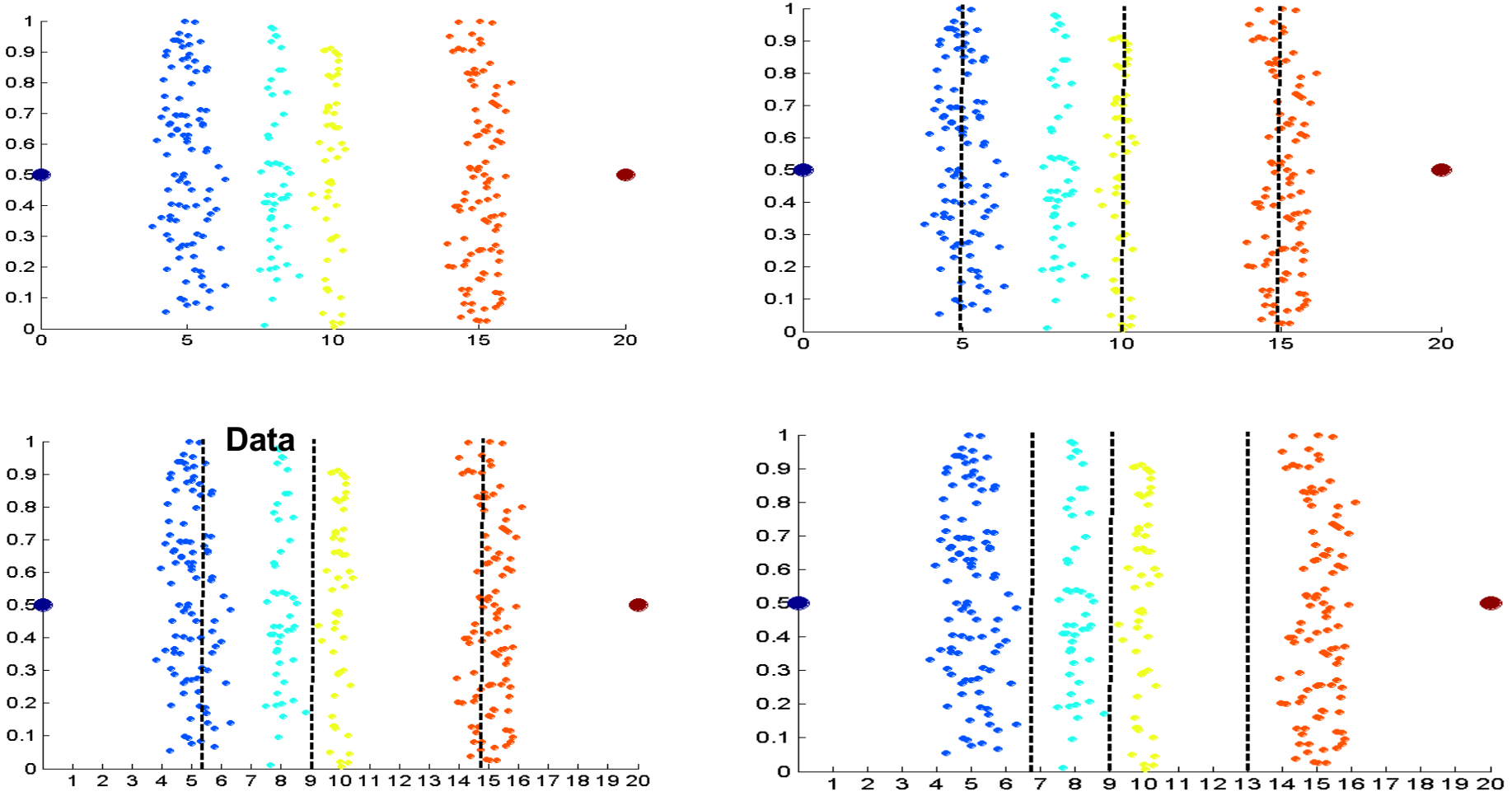
- Equal-width (distance) partitioning
  - Divides the range into *N* intervals of equal size: uniform grid
  - if *A* and *B* are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
  - The most straightforward, but outliers may dominate presentation
  - Skewed data is not handled well
- Equal-depth (frequency) partitioning
  - Divides the range into *N* intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky

Binning Methods for Data Smoothing

- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

* Partition into equal-frequency (**equi-depth**) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
* Smoothing by **bin means**:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
* Smoothing by **bin boundaries**:
  - Bin 1: 4, 4, 4, 15
  - Bin 2: 21, 21, 25, 25
  - Bin 3: 26, 26, 26, 34

Discretization Without Using Class Labels (Binning vs. Clustering)



Equal frequency (binning)

K-means clustering leads to better results

Discretization by Classification & Correlation Analysis

- Classification (e.g., decision tree analysis)
  - Supervised: Given class labels, e.g., cancerous vs. benign
  - Using *entropy* to determine split point (discretization point)
  - Top-down, recursive split
  - Details to be covered in Chapter 7
- Correlation analysis (e.g., Chi-merge: χ^2-based discretization)
  - Supervised: use class information
  - Bottom-up merge: find the best neighboring intervals (those having similar distributions of classes, i.e., low χ^2 values) to merge
  - Merge performed recursively, until a predefined stopping condition

Concept Hierarchy Generation

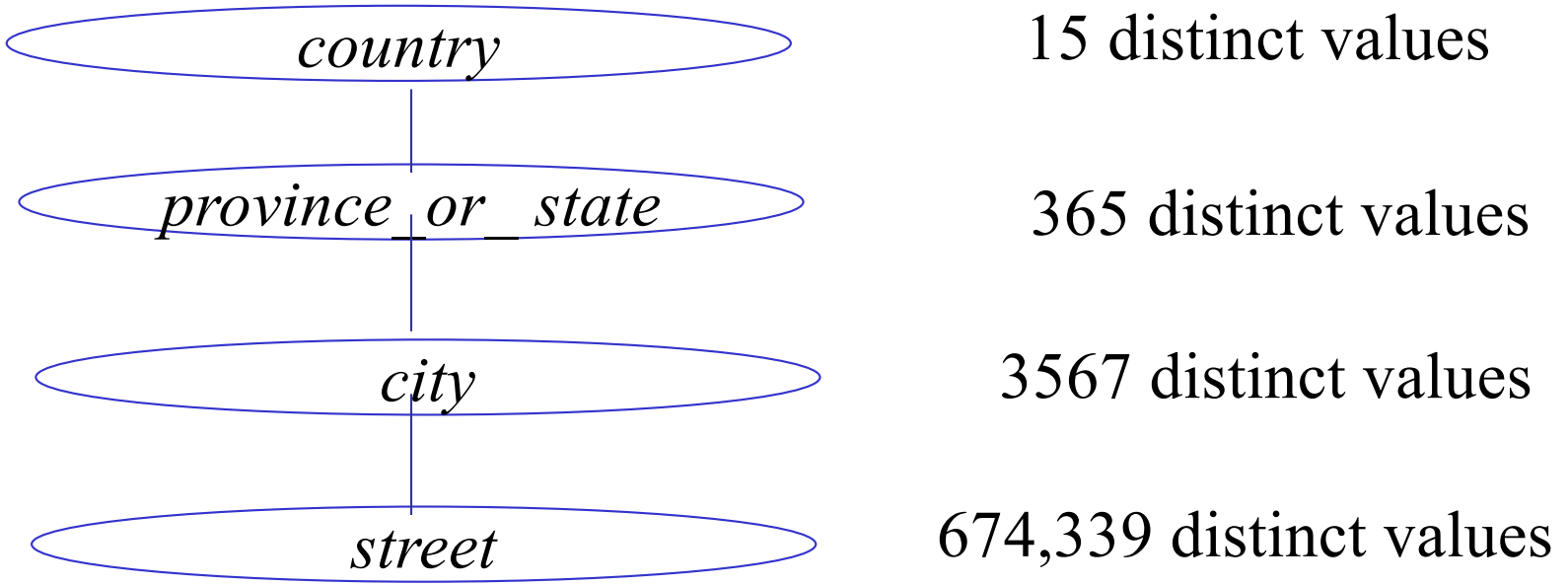
- **Concept hierarchy** organizes concepts (i.e., attribute values) hierarchically and is usually associated with each dimension in a data warehouse
- Concept hierarchies facilitate drilling and rolling in data warehouses to view data in multiple granularity
- Concept hierarchy formation: Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for *age*) by higher level concepts (such as *youth, adult*, or *senior*)
- Concept hierarchies can be explicitly specified by domain experts and/or data warehouse designers
- Concept hierarchy can be automatically formed for both numeric and nominal data. For numeric data, use discretization methods shown.

Concept Hierarchy Generation for Nominal Data

- Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
  - *street* < *city* < *state* < *country*
- Specification of a hierarchy for a set of values by explicit data grouping
  - {Urbana, Champaign, Chicago} < Illinois
- Specification of only a partial set of attributes
  - E.g., only *street* < *city*, not others
- Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
  - E.g., for a set of attributes: {*street, city, state, country*}

Automatic Concept Hierarchy Generation

- Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set
  - The attribute with the most distinct values is placed at the lowest level of the hierarchy
  - Exceptions, e.g., weekday, month, quarter, year



- Data Preprocessing: An Overview
  - Data Quality
  - Major Tasks in Data Preprocessing
- Data Cleaning
- Data Integration
- Data Reduction
- Data Transformation and Data Discretization
- Summary

Summary

- **Data quality**: accuracy, completeness, consistency, timeliness, believability, interpretability
- **Data cleaning**: e.g. missing/noisy values, outliers
- **Data integration** from multiple sources:
  - Entity identification problem
  - Remove redundancies
  - Detect inconsistencies
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation

References

- D. P. Ballou and G. K. Tayi. Enhancing data quality in data warehouse environments. Comm. of ACM, 42:73-78, 1999
- A. Bruce, D. Donoho, and H.-Y. Gao. Wavelet analysis. *IEEE Spectrum*, Oct 1996
- T. Dasu and T. Johnson. Exploratory Data Mining and Data Cleaning. John Wiley, 2003
- J. Devore and R. Peck. *Statistics: The Exploration and Analysis of Data*. Duxbury Press, 1997.
- H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita. Declarative data cleaning: Language, model, and algorithms. *VLDB'01*
- M. Hua and J. Pei. Cleaning disguised missing data: A heuristic approach. *KDD'07*
- H. V. Jagadish, et al., Special Issue on Data Reduction Techniques. Bulletin of the Technical Committee on Data Engineering, 20(4), Dec. 1997
- H. Liu and H. Motoda (eds.). *Feature Extraction, Construction, and Selection: A Data Mining Perspective*. Kluwer Academic, 1998
- J. E. Olson. *Data Quality: The Accuracy Dimension*. Morgan Kaufmann, 2003
- D. Pyle. Data Preparation for Data Mining. Morgan Kaufmann, 1999
- V. Raman and J. Hellerstein. Potters Wheel: An Interactive Framework for Data Cleaning and Transformation, VLDB'2001
- T. Redman. *Data Quality: The Field Guide*. Digital Press (Elsevier), 2001
- R. Wang, V. Storey, and C. Firth. A framework for analysis of data quality research. IEEE Trans. Knowledge and Data Engineering, 7:623-640, 1995