Matlab for Finance Course: Session 4

Dr. Peter A. Bebbington

Brainpool AI

peter@brainpool.ai

bebbington @brainpoolai @brainpoolai

November 30, 2024

OBJECTIVES

- Linear Regression Fundamentals
  - Understanding the mathematical framework
  - Implementation in MATLAB
  - Model validation techniques
- Advanced Regression Topics
  - Polynomial basis functions
  - Regularization methods
  - Overfitting and underfitting
- Statistical Analysis
  - Correlation matrices and stability
  - Coefficient of determination (R^2)
  - Significance testing
- Session Outcomes
  - Ability to implement regression models
  - Understanding of model selection criteria
  - Skills in model validation and testing

FUNCTION HANDLES - BASICS

- Function Handle Definition:
  - A variable that contains a reference to a function
  - Can be passed as arguments to other functions
  - Enables dynamic function calls
- Basic Syntax Examples:

```matlab
% Anonymous function handle
fun1 =
    @(x)((2*pi).^(-0.5))*exp((-0.5)*x.^2);

% Handle to existing function
fun2 = @sin; % Reference to sine function
```

- Visualization Example:

```matlab
fplot(fun1) % Plot Gaussian function
```

FUNCTION HANDLES - APPLICATIONS

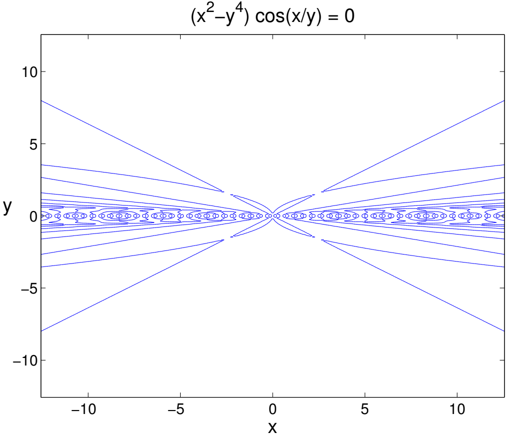
- Common Applications:
  - Numerical Integration
  - Function Optimization
  - Callback Functions
- Function String Alternative:

```
% For symbolic math and plotting
fplot('(x^2-y^4)*cos(x/y)',[-4*pi,4*pi])
```

- Key Differences:
  - Function Handles: Better for numerical computations
  - Function Strings: Better for symbolic manipulation
  - Both useful for visualization

FUNCTION STRING

- Given $f(x,y) = (x^2 - y^4)\cos(x/y)$
- Plot the implicit function $(x^2 - y^4)\cos(x/y) = 0$ by
  `fplot('(x^2-y^4).*cos(x./y)',[-4*pi,4*pi])`



PORTFOLIO OPTIMISATION

- Given a portfolio which is defined as follows:

$$\pi_i \begin{cases} > 0, & \text{long position (buying assets)} \\ = 0, & \text{no position} \\ < 0, & \text{short position (selling asset)} \end{cases}$$

- We can define the portfolios variance as

$$\sigma_p^2 = \boldsymbol{\pi}'\boldsymbol{\Sigma}\boldsymbol{\pi}$$

- Our goal in portfolio optimisation is to

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}}\{\sigma_p^2\}$$

with the following constraints

$$\boldsymbol{\pi}'\mathbf{1} = 1$$

$$\boldsymbol{\pi}'\boldsymbol{\mu} = \mu_p.$$

OPTIMAL SOLUTION

- The solution to $\boldsymbol{\pi}^*$ is found by FOC giving

$$\pi_i^* = \frac{\lambda_1}{2}\sum_{j=1}^{N}\Sigma_{ij}^{-1} + \frac{\lambda_2}{2}\sum_{j=1}^{N}\Sigma_{ij}^{-1}\mu_j$$

- This problem is written linear as

$$\underbrace{\begin{pmatrix} \tilde{\Sigma}_{11} & \tilde{\Sigma}_{12} & \cdots & \tilde{\Sigma}_{1N} & 1 & \tilde{\mu}_1 \\ \tilde{\Sigma}_{21} & \tilde{\Sigma}_{22} & & \vdots & 1 & \tilde{\mu}_2 \\ \vdots & & \ddots & & \vdots & \vdots \\ \tilde{\Sigma}_{N1} & \cdots & & \tilde{\Sigma}_{NN} & 1 & \tilde{\mu}_N \\ 1 & 1 & \dots & 1 & 0 & 0 \\ \tilde{\mu}_1 & \tilde{\mu}_2 & \dots & \tilde{\mu}_N & 0 & 0 \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} \pi_1^* \\ \pi_2^* \\ \vdots \\ \pi_N^* \\ -\lambda_1/2 \\ -\lambda_2/2 \end{pmatrix}}_{\mathbf{a}} = \underbrace{\begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ \mu_p \end{pmatrix}}_{\mathbf{b}}$$

where the the mutual fund strategy is found by

$$\mathbf{a} = \mathbf{A}^{-1}\mathbf{b}.$$

PORTFOLIO OPTIMIZATION - IMPLEMENTATION

- Generate synthetic data using GARCH models:

```
model = garch('Constant', 0.01,...
              'GARCH', 0.1,...
              'ARCH',0.1);
[x,returns] = simulate(model,nmax);
```

- Calculate optimal portfolio weights:

```
A = [cov_mat  e'    xav';
     e        0     0;
     xav      0     0];
w = A\b; % Solve system for weights
```

COEFFICIENT OF DETERMINATION

- In the session script, we see that the solution to $\mathbf{a}$ is found using $\mathbf{a} = \mathbf{A}\backslash\mathbf{b}$ (equivalent to a linear regression), which is quicker and more accurate than using `inv(A)` or $\mathbf{A}^{-1}$.
- Assuming linear correlations $X_j = a + bX_i$, we can measure how well the model fits with:

$$\varepsilon_{i,j} = X_j - (a + bX_i)$$

  which is known as a residual.
- We can define the "Coefficient of Determination" as the square of the elements of the correlation matrix:

$$\rho_{i,j}^2 = 1 - \frac{\mathbb{V}[\varepsilon_{i,j}]}{\mathbb{V}[X_j]}$$

  where:
  - $\rho_{i,j}^2 = 1 \ \forall i, j$ means the linear model fits perfectly
  - Large $\mathbb{V}[\varepsilon_{i,j}]$ indicates a poor linear fit

CORRELATION MATRIX VISUALIZATION

- Visualize correlation matrix using heatmap:

```
imagesc(cor_mat)
colorbar
colormap('jet')
axis square
```

- Key insights:
  - Diagonal elements are always 1 (self-correlation)
  - Symmetric matrix: $\rho_{i,j} = \rho_{j,i}$
  - Color intensity shows correlation strength

STABILITY OF CORRELATIONS - WINDOW ANALYSIS

- After calculating correlations, we analyze their stability using:
  - Rolling windows of 250 days
  - Mean correlation over windows:

$$\bar{\rho} = \frac{1}{w} \sum_{t=1}^{w} \mathrm{corr}(X_{t:t+250})$$

  - Standard deviation:

$$\sigma_\rho = \sqrt{\frac{1}{w} \sum_{t=1}^{w} \mathrm{corr}(X_{t:t+250})^2 - \bar{\rho}^2}$$

CORRELATION SIGNIFICANCE: STUDENT'S T-TEST

- Parametric Test Characteristics:
  - Assumes normal distribution
  - Tests null hypothesis of zero correlation
  - Computationally efficient

```
% Returns correlation matrix and p-values
[cor_mat, P_ttest] = corrcoef(X);
% Interpret results
significant = P_ttest < 0.05; % 5%
    significance
```

- Interpretation:
  - `cor_mat`: Pearson correlation coefficients
  - `P_ttest`: Corresponding *p*-values
  - Small *p*-values indicate significant correlation

CORRELATION SIGNIFICANCE: PERMUTATION TEST

- Non-parametric Test Characteristics:
  - No distribution assumptions
  - More robust for non-normal data
  - Computationally intensive

```
pp = zeros(size(cor_mat));
for t = 1:1000 % Random perm of time series
    ct = corrcoef(X(randperm(T),:));
    pp = pp + (abs(ct) >= abs(cor_mat));
end % Count stronger correlations
P_perm = pp/1000; % Convert to p-values
```

- Interpretation:
  - `P_perm`: Ratio of random correlations exceeding observed
  - Lower values indicate stronger evidence against null
  - More reliable for non-normal distributions

INTERPRETING CORRELATION STABILITY

Three key aspects to consider:

1. Statistical Significance
   - p-value < 0.05 suggests correlation is significant
   - Both t-test and permutation test should agree
   - Consider multiple testing corrections (e.g., Bonferroni)
   - Higher sample size increases statistical power
2. Temporal Stability
   - High σ_ρ indicates unstable correlations
   - Market regimes can affect stability
   - Consider using rolling windows of different sizes
   - Test for structural breaks in correlation patterns
3. Economic Significance
   - Strong correlations may not imply causation
   - Consider fundamental relationships (Supply/demand dynamics, Interest rate sensitivity, Business cycle effects)
   - Evaluate impact of market microstructure
   - Account for trading costs and liquidity

SUPERVISED LEARNING METHOD

- One Can define a training set as

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$$

- The goal is to infer a function

$$\mathcal{D} \mapsto f_{\mathcal{D}}(\mathbf{x}_i) \approx y_i$$

then apply $f_{\mathcal{D}}$ help predict future data set

$$\mathcal{D}' = \{(\mathbf{x}_{m+1}, y_{m+1}), (\mathbf{x}_{m+2}, y_{m+2}), \ldots\}$$

- examples:
  - Classification $y \in \{-1, +1\}$
  - Regression $y \in \mathbb{R}$

LINEAR REGRESSION

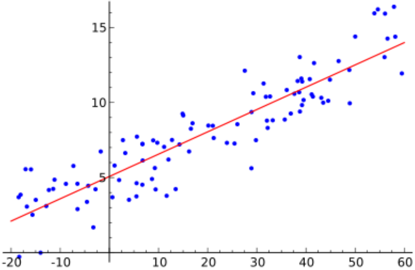
- This approach tries to fit the linear line

$$y_i \approx w_1 x_{i1} + w_2 x_{i2} + ... + w_m x_{im} + b_i$$

where $i = 1, 2, ..., n$ which can be written in matrix notation

$$y_i \approx \mathbf{x}_i' \mathbf{w} + b_i = \mathbf{X}'\mathbf{w} + \bar{b}$$

where $\bar{b} \in \mathbb{R}^n$ is a constant and represents the error/residuals, $\mathbf{w} \in \mathbb{R}^m$, $\mathbf{x}_i \in \mathbb{R}^n$ are column vectors and $\mathbf{X} \in \mathbb{R}^{n \times m}$ is a rectangular matrix.



MEAN SQUARE ERROR (MSE)

- MSE is defined as

$$\mathrm{MSE}(\mathcal{D}, \mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

where $\hat{y}_i$ is our linear predictor $\hat{y}_i = \mathbf{w}'\mathbf{x} = \sum_{j=1}^{n} \mathrm{w}_j x_{ij}$

- Implementation in MATLAB (`mse_cost.m`):

```
function mse = mse_cost(X, y, w)
    mse = mean((X * w - y).^2);
end
```

LEAST SQUARE REGRESSION (LSR)

- This model looks for the weight vector that minimises the mean square errors on all **training samples** and is defined as

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} (\mathbf{x}_i' \mathbf{w} - y_i)^2$$

- To solve the LSR equation we use matrix notation

$$\mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} (\mathbf{x}_i' \mathbf{w} - y_i)^2 = \frac{1}{m} (\mathbf{X}' \mathbf{w} - \mathbf{y})' (\mathbf{X}' \mathbf{w} - \mathbf{y})$$

- Then apply FOC ($\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = 0$), we find

$$\mathbf{w}^* = (\mathbf{X}' \mathbf{X})^{-1} \mathbf{X}' \mathbf{y}$$

- Solution exists if $\mathbf{X}$ is non-singular.

LSR FUNCTION

- Implementation of Least Square Regression in MATLAB (`linreg.m`):

```
function w = linreg(X, y)
    w = (X' * X) \ (X' * y);
end
```

- Example usage:

```
% Create example data
X = randn(3,3); % Random 3x3 matrix
y = randn(3,1); % Random output vector
% Compute weights and MSE
w = linreg(X,y);
mse = mse_cost(X,y,w);
```

- Note: Uses backslash operator for numerical stability

fitlm

- Alternative, one can use the function `fitlm()` which has the benefit of calculating various statistics

```
mdl =
Linear regression model:
    y ~ 1 + x1 + x2 + x3

Estimated Coefficients:
                   Estimate       SE         tStat       pValue
                   _________    ________    ________    __________

    (Intercept)    -0.028804    0.031438    -0.91619       0.35979
    x1               0.31826    0.032101      9.9144    3.6812e-22
    x2                1.0032    0.030797      32.575   4.7086e-159
    x3              -0.30363    0.031046     -9.7798    1.2461e-21

Number of observations: 1000, Error degrees of freedom: 996
Root Mean Squared Error: 0.992
R-squared: 0.559,  Adjusted R-Squared 0.557
F-statistic vs. constant model: 420, p-value = 2.61e-176
>>
```

BASIS FUNCTIONS

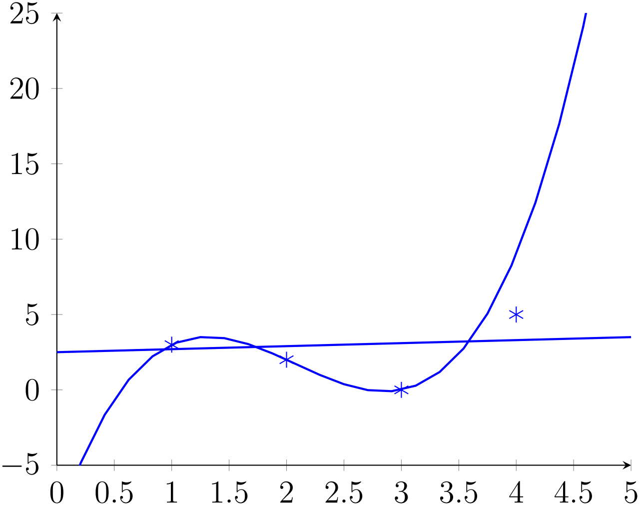
- If the function $f_{\mathcal{D}}$ is non-linear try introducing a polynomial vector as your basis function $\boldsymbol{\phi}(\mathbf{x}_i) = \phi_j(\mathbf{x}_i) = (1, x_i, x_i^2, \ldots, x_i^k)$ where $i = 1, 2, ..., m$ and $j = 1, 2, ..., k$. We now make the following change of basis $\mathbf{x}_i'\mathbf{w} \rightarrow \boldsymbol{\phi}(\mathbf{x}_i)'\mathbf{w}$.
- The LSR problem know becomes

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} (y_i - \sum_{j=1}^{k} \phi_j(\mathbf{x}_i) w_j)^2 = (\boldsymbol{\Phi}'\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}'\mathbf{y}$$

where the matrix

$$\boldsymbol{\Phi} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & \cdots & x_2^k \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_m & x_m^2 & \cdots & x_m^k \end{pmatrix}$$

POLYFIT EXAMPLE



POLYFIT EXAMPLE - SIMPLE CASE

- Basic polynomial fitting with different degrees:

```
x = [1,2,3,4]';
y = [3,2,0,5]';
for k = 1:4
    xx = basis(x,k); % Create basis functions
    w = linreg(xx,y); % Perform linear
        regression
    c = mse_cost(xx,y,w);
    fprintf('Bases dim: %g, MSE: %.2f\n', k,
        c);
end
```

- Key insights:
  - Higher degree polynomials reduce training error
  - Risk of overfitting increases with polynomial degree

POLYFIT EXAMPLE - COMPLEX CASE

- Three polynomial models tested:
  - Underfit: degree 2 (linear + quadratic terms)
  - Close fit: degree 3 (adds cubic term)
  - Overfit: degree 9 (high-order polynomial)
- Analysis includes:
  - Training error vs Test error
  - Effect of increasing data points
  - RMS error comparison across models
- Key findings:
  - Degree 3 polynomial typically provides best balance
  - Higher degrees show lower training error but higher test error
  - More data points help reduce overfitting

REGULARIZATION EFFECTS

- Regularization parameter $\lambda = e^{-10}$ helps control overfitting
- Effects on different models:
  - Underfitting model: minimal impact
  - Close fitting model: slight smoothing
  - Overfitting model: significant reduction in oscillations
- Trade-offs:
  - Higher λ: smoother fits, potentially underfitting
  - Lower λ: closer fits, risk of overfitting
  - Optimal λ depends on noise level and data quantity

Ridge Regression (RR)

- A rule of thumb if $m << n$ LSR may find a function $f_{\mathcal{D}}$ that will overfit your data, that is you are just fitting noise.
- A way avoid this is RR defined as

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left\{ \lambda \mathbf{w}'\mathbf{w} + \frac{1}{m} \sum_{i=1}^{m} (\mathbf{x}_i'\mathbf{w} - y_i)^2 \right\}$$

solve as before impose the FOC and we find

$$\begin{aligned} \mathbf{X}'\mathbf{X}\mathbf{w}^* + \lambda m \mathbf{w}^* &= \mathbf{X}'\mathbf{y} \\ \Rightarrow \mathbf{w}^* &= (\mathbf{X}'\mathbf{X} + \lambda m \mathbb{I}_n)^{-1}\mathbf{X}'\mathbf{y} \end{aligned}$$

where $\mathbb{I}_n$ is the $n \times n$ identity matrix.

RR & POLYNOMIAL BASIS

- Same Logic as before, replace $\mathbf{x}_i'\mathbf{w} \to \phi(\mathbf{x}_i)'\mathbf{w}$ picking a polynomial basis of the form $\phi(\mathbf{x}_i) = \phi_j(\mathbf{x}_i) = (1, x_i, x_i^2, \ldots, x_i^k)$ where $i = 1, 2, ..., m$ and $j = 1, 2, ..., k$.
- The RR problem is now defined as

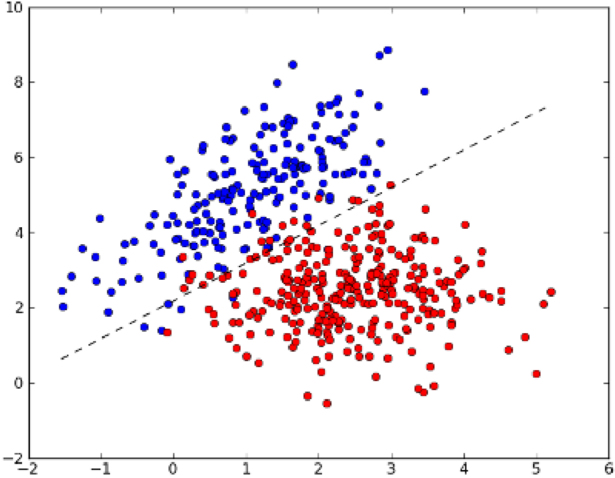
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \left\{ \lambda \mathbf{w}'\mathbf{w} + \frac{1}{m} \sum_{i=1}^{m} (y_i - \sum_{j=1}^{k} \phi_j(\mathbf{x}_i) w_j)^2 \right\}$$

solve as before impose the FOC and we find

$$\begin{aligned} \mathbf{\Phi}'\mathbf{\Phi}\mathbf{w}^* + \lambda k \mathbf{w}^* &= \mathbf{\Phi}'\mathbf{y} \\ \Rightarrow \mathbf{w}^* &= (\mathbf{\Phi}'\mathbf{\Phi} + \lambda k \mathbb{I}_k)^{-1} \mathbf{\Phi}'\mathbf{y} \end{aligned}$$

where $\mathbb{I}_k$ is the $k \times k$ identity matrix.

LINEAR CLASSIFICATION



BINARY CLASSIFICATION

- Used Everywhere!
- A few projects last year used this (Credit Risk).
- Not a Regression!
- Its Actual Classification Method.
- Think back if we have a training set then

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$$

- Classification $\Rightarrow y_i \in \{0, 1\}$, $y_i \in \{-1, +1\}$ or $y \in \{C_1, C_2\} \rightarrow$ known as binary classification

LOGISTIC REGRESSION

- Our objective is to find some linear relationship (a hyperplane) in our new basis function space that divides the two classes $\{C_1, C_2\}$. The hyperplane is defined as before $\mathbf{w} \in \mathbb{R}^n$ such that

$$p(C_1|\mathbf{w}, \mathbf{x}) = \sigma(\mathbf{w}'\phi(\mathbf{x}))$$

  where is the $\sigma(.)$ is a sigmoid function defined as

$$\sigma(u) = \frac{1}{1 + e^{-u}}$$

- You can see what the function looks by the command `fplot('1/(1+exp(-u))')`.
- **Please note that this is not a sigma-algebra**

MLE SOLUTION

- For the likelihood of observing outputs $\mathbf{y} \in \{C_1, C_2\}_{i=1}^m$ given inputs $\mathbf{X}$ and a hyperplane parametrised by $\mathbf{w}$ will be given by

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}) = \prod_{i=1}^{m} [\sigma(\mathbf{w}'\phi(\mathbf{x}_i))]^{y_i} [1 - \sigma(\mathbf{w}'\phi(\mathbf{x}_i))]^{1-y_i}$$

- Objective $\arg\max_{\mathbf{w}}\{\log(p(\mathbf{t}|\mathbf{w}, \mathbf{X}))\}$ a useful relationship to find this $\sigma'(u) = \sigma(u)(1 - \sigma(u))$.
- As exercise prove the following result

$$\frac{\partial}{\partial \mathbf{w}} \log(p(\mathbf{y}|\mathbf{w}, \mathbf{X})) = \sum_{i=1}^{m} (y_i - \underbrace{\sigma(\mathbf{w}'\phi(\mathbf{x}_i))}_{\hat{y}_i})\phi(\mathbf{x}_i) = 0$$

- We now have something which is of a similar form to a LSR and the original reason why this method was called a regression.

BINARY LOGISTIC REGRESSION

- Consider $\mathbf{w} = (\mathrm{w}_0, \mathrm{w}_1, \mathrm{w}_2, \mathrm{w}_3)$ and $\phi(\mathbf{x}_i) = (1, x_1, x_2, x_3)$, so our

$$\hat{y}_i = \sigma(\mathbf{w}'\phi(\mathbf{x}_i)) = \sigma(\mathrm{w}_0 + \mathrm{w}_1 x_1 + \mathrm{w}_2 x_2 + \mathrm{w}_3 x_3)$$

- Credit Risk

$$p(\mathrm{default}|\mathrm{data}) = p(y = 1|\mathbf{w}, \mathbf{X}) = \sigma(\mathbf{w}'\mathbf{X})$$

- Lets look at an example in the session script

PRACTICAL APPLICATIONS

- Financial Applications:
  - Credit Risk Assessment
  - Trading Signal Generation
  - Market Regime Classification
- Implementation Considerations:
  - Data Preprocessing
  - Feature Engineering
  - Model Selection Criteria
- Common Pitfalls:
  - Class Imbalance
  - Feature Correlation
  - Overfitting to Historical Data

KEY TAKEAWAYS

- Portfolio Optimization
  - Efficient implementation using backslash operator
  - Consider correlation stability in weight calculation
- Statistical Analysis
  - Coefficient of determination measures fit quality
  - Multiple approaches to test correlation significance
  - Window analysis reveals temporal patterns
- Implementation Tips
  - Use vectorized operations when possible
  - Consider computational efficiency
  - Validate results with multiple methods