# 疫情时期航空业股市波动性分析

致理-数理 1　刘苏青* 2021013371

2024 / 06 / 22

**摘　要**

本文通过分析新冠疫情时期全球主要航空公司股票的市场波动性，探讨疫情对航空业的影响。我们采用了 VAR、RNN 和 LSTM 模型对航空公司股票的时间序列进行了详细分析与预测，并使用 CAPM 和 Fama-French 三因子模型评估了股票的系统风险与收益。此外，通过 Markowitz 均值-方差优化方法优化了投资组合。研究发现，疫情初期航空公司股票经历了显著波动，随后在政府干预和市场逐步恢复的背景下有所回升，但整体波动性仍然较高。本研究为投资者和政策制定者提供了关于疫情期间航空公司股票表现的综合分析。

**关键词**：**新冠疫情**，**航空股票**，VAR，LSTM，CAPM

*liu-sq21@mails.tsinghua.edu.cn

# 目录

# 1 引言

## 1.1 研究背景及意义

COVID-19 疫情自 2019 年底爆发以来，对全球经济各个领域产生了深远的影响。航空业作为最受冲击的行业之一，因旅行限制和需求急剧下降，经历了剧烈的市场动荡。航空公司股票价格在疫情初期大幅下跌，随后在各国政府的支持和疫苗推广的背景下逐步恢复。了解和分析疫情期间航空公司股票的波动性和风险特征，对投资者和政策制定者具有重要的参考价值。本研究旨在通过多种金融统计方法，全面剖析主要航空公司股票在疫情期间的市场表现，评估其系统风险，优化投资组合，并探讨影响股票收益的主要因素。

## 1.2 研究现状与评价

目前关于 COVID-19 疫情对航空公司股票影响的研究较多，但大多集中于短期市场反应的分析，缺乏对长期波动性和风险的深入探讨。同时，现有研究中对多种金融统计模型的综合应用较少，尤其是深度学习模型在股票预测中的应用尚不充分。本研究将通过 VAR、RNN 以及 LSTM 模型，对航空公司股票进行多维度的时间序列分析与预测，并结合 CAPM 模型，系统评估股票的风险和收益特征，以期填补现有研究的空白，提供更全面的分析视角。

## 1.3 本文贡献与创新点

- 多模型综合分析：本文综合应用了多种金融统计模型，包括 VAR、LSTM 和 RNN 模型，对航空公司股票的时间序列进行详细分析与预测，提供了更全面的市场行为理解。
- 系统风险与收益评估：通过 CAPM 模型系统评估了疫情期间航空公司股票的系统风险和预期收益，揭示了市场对航空业前景的预期。
- 投资组合优化：应用 Markowitz 均值-方差优化方法，在疫情背景下进行投资组合优化，为投资者提供科学的投资决策支持。

## 1.4 文章基本思路与结构

本文结构如下：

- 导言：介绍研究背景及意义，阐述研究现状与评价，明确本文的贡献与创新点。
- 数据预处理：详细描述数据来源与清洗过程，并进行初步的探索性数据分析。
- 时间序列分析与预测：应用 VAR、RNN 和 LSTM 模型，对航空公司股票进行时间序列分析与预测。
- CAPM 分析：使用 CAPM 模型，评估航空公司股票的系统风险和市场回报率的关系，并计算预期收益率。
- 投资组合优化：应用 Markowitz 均值-方差优化方法，优化投资组合，绘制有效前沿，并选择最优投资组合。
- 结论与局限性：总结主要发现和结论，讨论各个方法的优缺点和适用性，并提出未来研究的建议。

# 2 数据预处理

## 2.1 数据来源与描述

本研究的数据主要来源于 Yahoo Finance 和 Federal Reserve Economic Data 数据库。我们选择了四家主要航空公司（AAL, DAL, BA, UAL）的股票价格数据以及市场指数数据（如 S&P 500），用于股票价格分析和预测；并从 Federal Reserve Economic Data 数据库下载了三个月期国库券数据，用来当作无风险利率。

## 2.2 数据清洗与处理

数据清洗步骤包括：

- 去除缺失值和异常值：删除数据中的 NaN 值和无穷大值，确保数据的完整性和有效性。
- 使用对数收益率公式 $r_t = \log(S_t/S_{t-1})$ 计算每日收益率，以捕捉价格变动的相对变化。
- 数据集切分：将数据集按照 8:2 的比例切分为训练集和测试集，以便模型在训练集上进行训练，并在测试集上进行验证。

## 2.3 探索性数据分析

通过绘制股票价格和回报关于时间的波动情况，我们可以直观地观察到 COVID-19 疫情期间股票价格和回报的变化趋势。我们不难发现，航空公司股价和 S&P500 指数在疫情初期经历了剧烈波动，尤其是在 2020 年 3 月，股票价格经历了大幅下跌。随后在政府干预、市场恢复以及疫情平稳的影响下波动逐步稳定，并且股票价格有所回升，在 2021 年末和 2022 年初回到了较高水平。此外，对比各航空公司的股票以及 S&P500 指数，我们可以发现与各航空公司的股票情况相比，S&P500 指数受市场的冲击影响较小、整体波动性较小，并且在疫情开始后一段时间能迅速回到并超过疫情前的水平，体现了市场的稳健性。

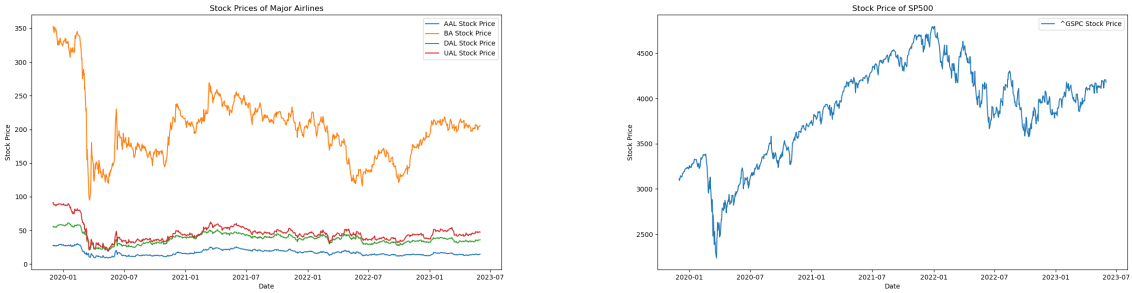


Figure 1: Stock Prices

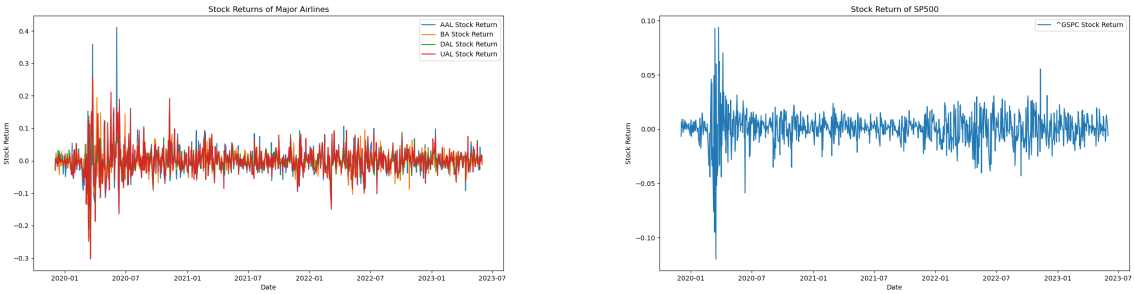


Figure 2: Stock Returns

通过以上的可视化分析，我们可以得出以下几点结论：

- 疫情对市场的冲击显著：无论是 S&P 500 指数还是主要航空公司股票，均在 2020 年初因疫情爆发而经历了剧烈的波动和显著的下跌。这一现象反映了疫情对全球金融市场的巨大冲击。

- 市场已经逐步恢复：尽管疫情初期市场大幅下跌，但在政府干预和经济逐步恢复的背景下，市场价格和收益率逐步回升，S&P 500 指数甚至在 2021 年和 2022 年达到了疫情前的水平。
- 航空业受影响尤为严重：与整体市场相比，航空公司股票的波动性更大，恢复速度更慢。这表明疫情对航空业的影响深远且持久，反映了航空业在疫情期间面临的巨大挑战。
- 未来预测的重要性：通过对历史数据的分析，我们可以更加准确地理解疫情对航空公司股票的影响。这为我们后续使用 VAR 模型和其他时间序列模型进行预测提供了坚实的基础，帮助我们更好地应对未来可能的市场波动。

# 3 时间序列分析与预测

在本部分中，我们将基于前面的探索性数据分析结果，进一步应用 VAR、RNN 以及 LSTM 模型对航空公司股票价格进行时间序列分析与预测。VAR 模型适用于多变量时间序列分析，能够捕捉不同变量之间的动态关系及其相互影响。RNN 和 LSTM 模型则属于深度学习范畴，擅长处理非线性和长时间依赖关系。

## 3.1 ADF 检验

ADF 检验（Augmented Dickey-Fuller Test）用于检验时间序列数据是否具有单位根，即数据是否为平稳的。零假设是时间序列存在单位根（非平稳），备择假设是时间序列不存在单位根（平稳）。

Table 1: ADF Test Results for Airline Stocks

| Stock | ADF Statistic | p-value |
|---|---|---|
| AAL | -10.12 | $9.72\times10^{-18}$ |
| BA | -9.54 | $2.72\times10^{-16}$ |
| DAL | -11.79 | $9.96\times10^{-22}$ |
| UAL | -11.88 | $6.31\times10^{-22}$ |

基于 ADF 检验结果，我们可以得出结论：AAL、BA、DAL 和 UAL 股票的回报序列是平稳的。这意味着这些时间序列数据不需要进一步差分就可以进行时间序列模型的构建和预测。接下来，我们将基于这些平稳的时间序列数据，应用 VAR、LSTM 和 RNN 模型对航空公司股票价格进行时间序列分析与预测。

## 3.2 VAR 模型

VAR 模型是一种多变量时间序列模型，适用于平稳的多变量时间序列数据，它通过捕捉多个变量之间的动态关系来进行建模和预测。在我们的研究中，涉及到四家航空公司的股价数据，这些数据不仅在时间上具有自身的动态特性，还可能存在相互之间的关联性。VAR 模型的优点在于能够同时处理多个时间序列，并且可以捕捉它们之间的相互影响。同时，与一些复杂的时间序列模型（如多变量 GARCH、LSTM 等）相比，VAR 模型具有相对简单的建模过程且可解释能力强，因此我们优先考虑这种高效且易于实现的模型。其基本形式为：

$$\boldsymbol{r}_t = \phi_0 + \boldsymbol{\Phi}_1 \boldsymbol{r}_{t-1} + \cdots + \boldsymbol{\Phi}_p \boldsymbol{r}_{t-p} + \boldsymbol{a}_t, \quad p > 0 \tag{1}$$

其中 $\boldsymbol{r}_t$ 是一个包含多个时间序列变量的向量，$\phi_0$ 是一个常数向量，$\boldsymbol{\Phi}_i$ 是滞后项的系数矩阵，$\boldsymbol{a}_t$ 是一个误差项向量。为了选择 VAR 模型的最优滞后阶数，我们使用了 AIC（Akaike Information Criterion）准则。AIC 准则通过平衡模型的拟合优度和复杂度来选择最佳模型。其计算公式为：

$$AIC = -2 \log \mathcal{L} + 2k, \tag{2}$$

其中，$k$ 是模型的参数数量，$\mathcal{L}$ 是模型的似然函数值，我们选择 AIC 值最小的模型作为最佳模型。在本研究中，我们通过 AIC 准则选取的最优滞后阶数 $p = 17$。通过在训练集上拟合一个 VAR(17) 模型，并在测试集上预测股票回报，我们可以绘制结果如下：

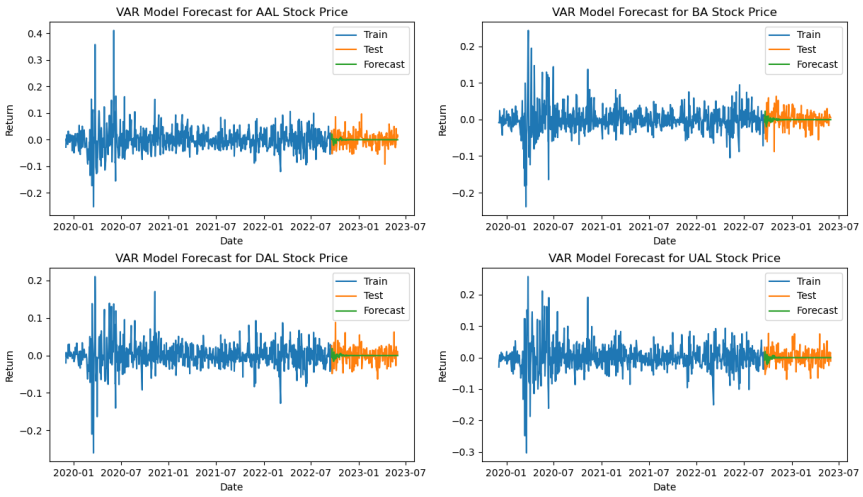


Figure 3: VAR

由图像可知，通过对比测试集上的预测结果和测试集上的真实数据，我们可以发现 VAR 模型在捕捉短期波动性方面表现稍好，但在长期趋势的预测方面相对平稳。这可能是因为 VAR 模型擅长处理多个时间序列变量之间的短期动态关系，而对于数据中的非线性关系和长期依赖性，VAR 模型的预测精度可能有所不足。

## 3.3 RNN 模型

在时间序列分析中，传统模型如 VAR 和 ARIMA 虽然在短期内表现较好，但对长期趋势和非线性关系的捕捉能力较弱。为了解决这一问题，我们引入了循环神经网络（Recurrent Neural Network, RNN）模型。这种模型作为一种经典的深度学习模型，可以通过非线性激活函数和深层网络结构有效地捕捉时间序列中的非线性依赖关系。此外，RNN 模型不严格要求时间序列数据的平稳性，因此我们也可以直接对股价进行建模并预测。在本研究中，我们在股价和回报的训练集上分别构建了一个两层 RNN 网络，并在最后一层加入全连接层以输出测试集上的股价和回报的预测结果，我们可以绘制结果如下：

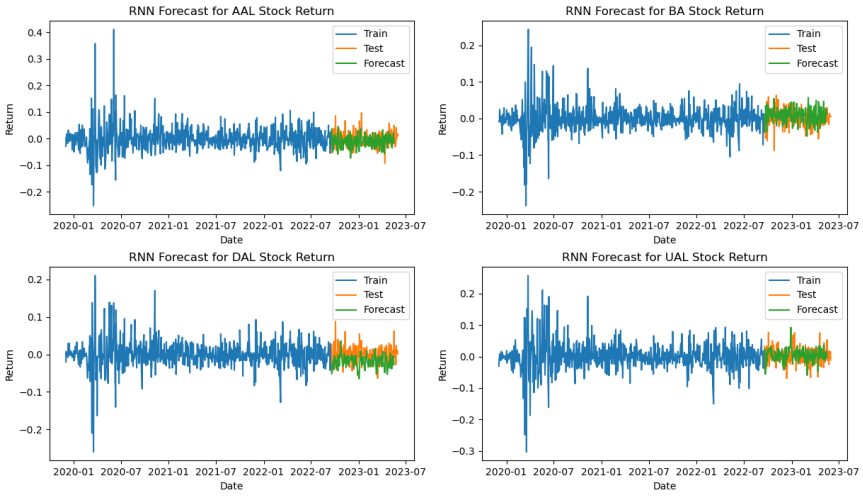


Figure 4: RNN

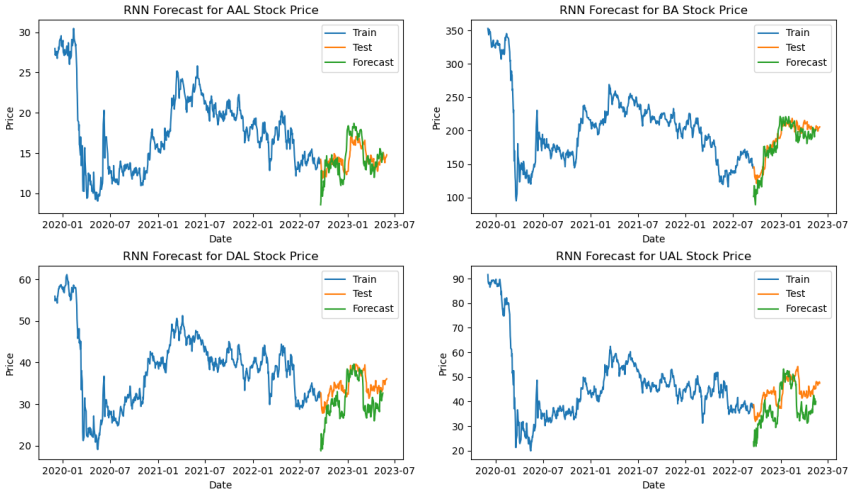


Figure 5: RNN

由图像可知，在股票回报预测问题上，相较于前文的 VAR 模型，RNN 模型能够较好地拟合训练集数据，并在测试集上同时保持良好的短期和长期层面的预测能力；在股票价格预测问题上，RNN 模型对股价的整体趋势进行了有效的捕捉，尽管在某些时刻的预测值与实际值有一定偏差，但总体趋势保持一致。

## 3.4 LSTM 模型

长短期记忆网络（Long Short-Term Memory, LSTM）作为一种特殊的循环神经网络（RNN），旨在解决传统 RNN 在长时间依赖问题中的梯度消失问题。LSTM 通过引入记忆单元和门控机制，能够更有效地捕捉和保留时间序列中的长期依赖关系。在本研究中，我们同样在股价和回报的训练集上分别构建了一个两层 LSTM 网络，并在最后一层加入全连接层以输出测试集上的股价和回报的预测结果，我们可以绘制结果如下：

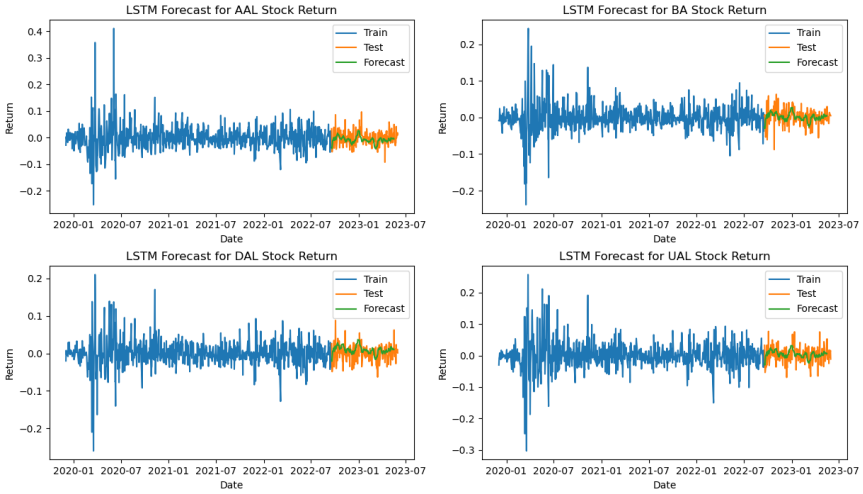


Figure 6: LSTM

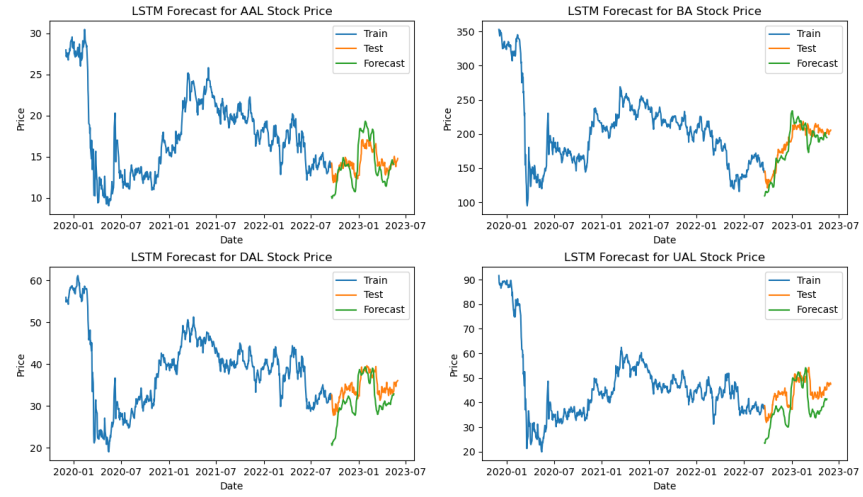


Figure 7: LSTM

由图像可知，在股票回报预测问题上，相较于前文的 VAR 模型，LSTM 模型也能够较好地拟合训练集数据，并在测试集上同时保持良好的短期和长期层面的预测能力，但是与 RNN 模型相比，LSTM 模型预测结果的波动程度较小，证明此模型倾向于给出更保守的估计（与真实数据的趋势接近，但是幅度较弱）；在股票价格预测问题上，RNN 模型对股价的整体趋势进行了有效的捕捉，尽管在某些时刻的预测值与实际值有一定偏差，但总体趋势保持一致。

为了更精准地衡量 VAR、RNN 以及 LSTM 模型对股票回报的预测效果，我们选取了 MSE、MAE 以及 RMSE 三种指标（RMSE 和 MSE 指标的本质相同，仅用来提供一个和 MAE 相同尺度的指标作为参考）来评估测试集上的预测结果与测试集上的真实数

据之间的误差。其定义式分别为：

$$MSE = \frac{1}{|S|}\sum_{i\in S}(\hat{r}_i - r_i)^2, \tag{3}$$

$$MAE = \frac{1}{|S|}\sum_{i\in S}|\hat{r}_i - r_i|, \tag{4}$$

$$RMSE = \sqrt{\frac{1}{|S|}\sum_{i\in S}(\hat{r}_i - r_i)^2}, \tag{5}$$

其中 $S$ 是测试集的所有日期的集合，$\hat{r}_i$ 是 $i$ 时刻股票回报的预测值，$r_i$ 是 $i$ 时刻股票回报的真实值。统计结果如下：

Table 2: Prediction Errors for Airline Stocks

| Model | AAL | BA | DAL | UAL |
|---|---|---|---|---|
| Mean Squared Error (MSE) | | | | |
| VAR | 0.0008 | 0.0005 | 0.0005 | 0.0007 |
| RNN | 0.0012 | 0.0012 | 0.0010 | 0.0009 |
| LSTM | 0.0007 | 0.0005 | 0.0005 | 0.0006 |
| Mean Absolute Error (MAE) | | | | |
| VAR | 0.0209 | 0.0162 | 0.0172 | 0.0202 |
| RNN | 0.0270 | 0.0261 | 0.0255 | 0.0232 |
| LSTM | 0.0211 | 0.0161 | 0.0180 | 0.0199 |
| Root Mean Squared Error (RMSE) | | | | |
| VAR | 0.0275 | 0.0219 | 0.0226 | 0.0267 |
| RNN | 0.0342 | 0.0340 | 0.0321 | 0.0298 |
| LSTM | 0.0273 | 0.0217 | 0.0234 | 0.0252 |

由表格可知，对于所有股票的大部分指标，我们均有 LSTM 模型的结果略优于 VAR 模型，且二者的结果严格优于 RNN 模型。但是结合前文的图像结果可知，VAR 模型对于长期趋势的预测效果较差，证明以上指标并不能完备地度量各模型预测结果的好坏，我们仍需结合具体情况进行分析。综上所述，LSTM 模型兼顾了同趋势以及低误差，因此在本任务中的预测效果最优。

# 4 CAPM 分析

## 4.1 CAPM 模型

在前面的分析中，我们通过 VAR、RNN 和 LSTM 模型对航空公司股票的未来走势进行了预测。为了进一步了解这些股票的系统性风险和市场回报率的关系，我们引入资本资产定价模型（CAPM）进行分析。CAPM 模型可以帮助我们评估每只股票的系统风险（即相对于市场的风险）和预期收益。资本资产定价模型（CAPM）是金融领域中广泛应用的模型，用于描述单个资产的预期收益和市场风险之间的关系。该模型的公式如下：

$$r_i - r_f = \alpha_i + \beta_i(r_m - r_f) + \varepsilon, \tag{6}$$

其中 $r_i$ 是股票 $i$ 的回报，$r_m$ 是市场组合的回报，$r_f$ 是无风险利率，$\beta_i$ 是股票 $i$ 的贝塔值，表示该股票相对于市场的系统性风险。CAPM 模型假设市场是有效的，即投资者通过承担系统性风险来获得额外的预期收益。通过对各航空公司股票的 CAPM 回归分析，我们可以估算每只股票的贝塔值，并了解其系统性风险。取 S&P500 的回报作为市场组合的回报，三个月期国库券收益率作为无风险利率，可得结果如下：

Table 3: CAPM Regression Results for Airline Stocks

| | AAL | BA | DAL | UAL |
|---|---|---|---|---|
| const ($\alpha$) | 0.0018 (0.001) | 0.0020 (0.001) | 0.0011 (0.001) | 0.0022 (0.001) |
| t-value | 1.199 | 1.816 | 1.014 | 1.579 |
| p-value | 0.231 | 0.070 | 0.311 | 0.115 |
| Market Factor ($\beta$) | 1.1624 (0.057) | 1.1923 (0.042) | 1.1170 (0.041) | 1.2010 (0.054) |
| t-value | 20.532 | 28.674 | 26.931 | 22.182 |
| p-value | 0.000 | 0.000 | 0.000 | 0.000 |
| R-squared | 0.325 | 0.484 | 0.453 | 0.359 |
| Adj. R-squared | 0.324 | 0.483 | 0.452 | 0.359 |
| F-statistic | 421.6 | 822.2 | 725.3 | 492.0 |
| Prob(F-statistic) | $8.41\times10^{-77}$ | $4.27\times10^{-128}$ | $6.77\times10^{-117}$ | $6.88\times10^{-87}$ |

由表格可知，在 0.05 的显著性水平下，各航空公司的股票的 $\alpha$ 均不显著，而各航空公司的股票的 $\beta$ 均显著且一致大于 1。这意味着市场模型是有效的，并且各航空公司的股票对市场变化的敏感度均较高，导致这类股票对市场变动的反应更剧烈。这恰恰呼应了前文探索性数据分析中观察到的现象：在新冠疫情初期，市场经历了剧烈的波动，因此高贝塔值的航空公司股票也经历了更大幅度的波动。

基于 CAPM 模型的结果显示了航空公司股票的高敏感度和高风险特性，所以在未来的投资决策中，投资者应考虑这些股票的高贝塔值所带来的潜在波动，并结合市场预期和自身的风险承受能力进行合理的资产配置。

# 5 投资组合优化

在了解了各航空公司股票的系统性风险和市场敏感度后，下一步我们将进行投资组合优化。通过 Markowitz 均值-方差优化模型，我们可以找到这几只航空业股票的最优投资组合，以实现风险和收益的最佳平衡。

## 5.1 Markowitz 均值-方差优化

Markowitz 均值-方差优化模型是一种经典的投资组合优化方法，通过计算不同资产组合的预期收益和风险，帮助投资者找到最优的资产配置。该模型的优化问题如下：

$$\max_{\alpha} \qquad \alpha^{\top}\xi - \frac{A}{2}\alpha^{\top}\Sigma\alpha \tag{7}$$

其中 $A$ 是投资者的风险厌恶系数，衡量了投资者对风险的态度，$\alpha$ 是无风险利率外的投资组合的权重向量，$\xi$ 是各股票的预期回报向量，$\Sigma$ 是各股票的协方差矩阵。在本研究中，假设无风险资产的投资比例为 20%，且投资者的风险厌恶系数 $A = 1$，可得结果如下：

Table 4: Optimized Portfolio Weights (excluding risk-free asset)

| Stock | Weight |
|---|---|
| AAL | 0.1186 |
| BA | 0.2479 |
| DAL | 0.6678 |
| UAL | -0.2342 |

由表格可知，投资组合中 AAL、BA 以及 DAL 股票的权重为正，而 UAL 股票的权重为负，这意味着如果我们想要达到最优投资组合，我们应当分别投资 11.86%、24.79% 以及 66.78% 的资产于 AAL、BA 以及 DAL 股票，同时约 23.42% 的资金应通过做空 UAL 股票来实现。

在前面的 CAPM 分析中，我们计算了各个股票的 $\beta$ 值，了解了它们与市场的敏感度。在投资组合优化中，我们更进一步地优化了投资组合的权重配置。CAPM 提供了单

个股票的风险和预期收益的衡量，而 Markowitz 均值-方差优化则通过这些个体股票的风险和收益，优化整个投资组合的配置，以实现更优的风险收益比。

结合使用 CAPM 以及 Markowitz 均值-方差优化方法，我们可以更全面地理解和优化投资决策，在有效管理风险的同时，实现投资组合的收益最大化。

## 5.2 绘制有效前沿

通过随机生成大量的投资组合，我们可以计算每个组合的预期收益、波动率以及夏普比率并绘制有效前沿如下：

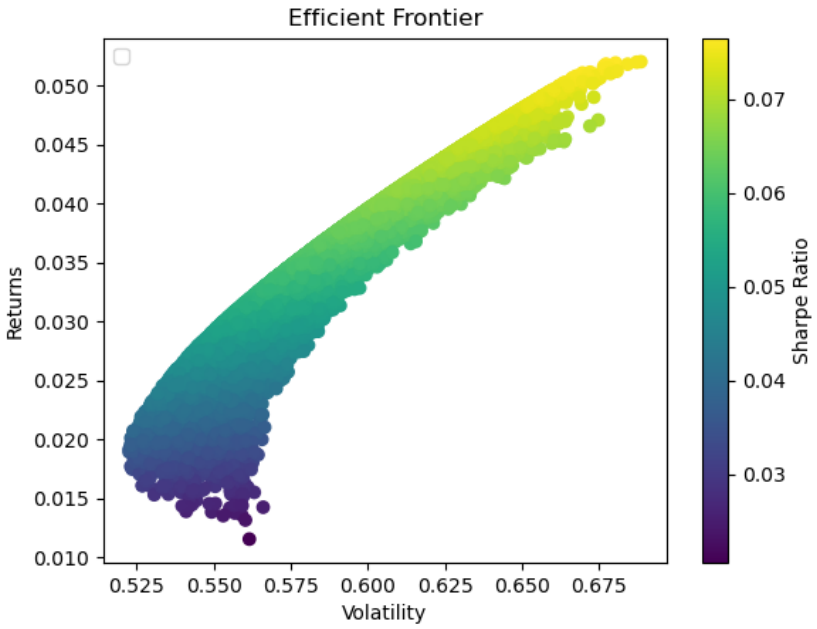


Figure 8: Efficient Frontier

由图像可知，图中的每个点代表一个随机生成的投资组合，横轴为组合的波动率（Volatility），纵轴为组合的预期收益率（Returns）。颜色表示夏普比率（Sharpe Ratio），即每单位风险所获得的超额收益。它展示了在给定风险水平下，能够实现的最高预期收益的组合，或在给定收益水平下，能够实现的最低风险的组合。通过有效前沿，投资者可以找到在不同风险偏好下的最佳投资组合。

# 6 结论与局限性

## 6.1 结论

本文通过多种金融统计模型对 COVID-19 疫情下全球主要航空公司股票的市场波动性进行了分析及预测，主要结论如下：

1. 疫情初期波动剧烈：疫情初期，航空公司股票经历了显著的市场波动，这与旅行限制和需求急剧下降直接相关。

2. 逐步恢复但波动性依旧较高：在各国政府的支持和市场逐步恢复的背景下，航空公司股票价格有所回升，但整体波动性仍然较高，体现出航空业前景的不确定性。

3. 时间序列预测性能：在时间序列预测中，LSTM 模型在捕捉股票价格的长期趋势和短期波动方面表现最佳，RNN 模型在拟合时间序列波动趋势方面不逊于 LSTM 模型，但是局限于某一点处的预测精度较低，VAR 模型在短期预测中表现较好，但在长期趋势预测方面有所不足。

4. 系统风险评估：通过 CAPM 分析发现，航空公司股票具有较高的系统风险，表明这些股票对市场变化的敏感度较高，这也印证了前文的探索性数据分析中的发现。

5. 投资组合优化：通过 Markowitz 均值-方差优化方法，本研究给出了一种合理配置资产的方法，可以在一定程度上降低风险并实现收益最大化。

通过对疫情期间航空公司股票的全面分析，本研究为投资者和政策制定者提供了以下两点重要建议：

- 投资者：在投资航空公司股票时，应考虑其高波动性和高风险特性，合理配置资产以平衡风险和收益。

- 政策制定者：需关注航空业的恢复情况，提供必要的政策支持以稳定市场，并促进航空业的长期发展。

## 6.2 局限性

我认为本研究仍存在以下局限性：

1. 数据时间跨度有限：本研究的数据主要集中在 COVID-19 疫情期间，缺乏更长时间范围内的数据来验证模型的稳定性和普适性，因此本文的结论可能不能够泛化到其他时间段。

2. 外部因素影响：研究未考虑其他可能影响航空公司股票表现的外部因素，如油价波动、地缘政治风险等，这些因素可能对研究结果产生影响。

3. 数据依赖性：研究的数据来源于 Yahoo Finance 和 Federal Reserve Economic Data 数据库，数据质量和准确性可能会对研究结果产生影响。

因此，未来研究可以在更长时间跨度和更多外部因素的考虑下，进一步验证和扩展本文的结论。此外，可以结合更多的市场数据和宏观经济因素，提供更全面的分析视角。

# 7 附录

## 7.1 Python 代码

```python
import pandas as pd
import yfinance as yf
import pandas_datareader.data as web
from statsmodels.tsa.api import VAR
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller, grangercausalitytests
from statsmodels.tsa.vector_ar.vecm import coint_johansen
from statsmodels.tsa.arima.model import ARIMA
from arch import arch_model
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, SimpleRNN
from sklearn.preprocessing import MinMaxScaler
import statsmodels.api as sm
import scipy.optimize as sco
from pmdarima import auto_arima
from statsmodels.tsa.api import VARMAX
from sklearn.metrics import mean_squared_error, mean_absolute_error
import datetime as dt

# 下载航空公司股票数据和市场指数数据
tickers = ['AAL', 'DAL', 'BA', 'UAL', '^GSPC']
stock_data = yf.download(tickers, start='2019-12-01', end='2023-06-01')['Adj Close']
stock_data.to_csv('airline_stock_prices.csv')

# 下载Fama-French三因子数据
#ff_factors = web.DataReader('F-F_Research_Data_Factors_daily', 'famafrench')[0]
#ff_factors = ff_factors / 100

# 计算每日股票收益率和市场收益率
returns = stock_data.pct_change().dropna()
market_returns = returns['^GSPC']
returns = returns.drop(columns=['^GSPC'])
```

```
34
35 # 保存收益率数据
36 returns.to_csv('airline_stock_returns.csv')
37 market_returns.to_csv('market_returns.csv')
38 #ff_factors.to_csv('ff_factors.csv')
39
40 # 设置时间范围
41 start = dt.datetime(2019, 12, 1)
42 end = dt.datetime(2023, 6, 1)
43
44 # 获取3个月国库券收益率数据
45 risk_free_rate = web.DataReader('TB3MS', 'fred', start, end)
46 risk_free_rate.columns = ['Risk_Free_Rate']
47 risk_free_rate.to_csv('risk_free_rate.csv')
48
49 # 加载数据
50 prices = pd.read_csv('airline_stock_prices.csv', index_col='Date', parse_
      dates=True)
51 prices = prices.loc['2019-12-01':]
52 prices = prices.iloc[:, :4]
53
54 plt.figure(figsize=(14, 7))
55 for ticker in prices:
56     plt.plot(prices.index, prices[ticker], label=f'{ticker} Stock Price')
57 plt.title('Stock Prices of Major Airlines')
58 plt.xlabel('Date')
59 plt.ylabel('Stock Price')
60 plt.legend()
61 plt.show()
62
63 # 加载数据
64 prices = pd.read_csv('airline_stock_prices.csv', index_col='Date', parse_
      dates=True)
65 prices = prices.loc['2019-12-01':]
66 prices = prices.iloc[:, 4]
67
68 plt.figure(figsize=(14, 7))
69 plt.plot(prices.index, prices, label=f'{prices.name} Stock Price')
```

```
plt.title('Stock Price of SP500')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.show()

# 加载数据
returns = pd.read_csv('airline_stock_returns.csv', index_col='Date', parse_dates=True)
returns = returns.loc['2019-12-01':]
returns = returns.iloc[:, :4]

plt.figure(figsize=(14, 7))
for ticker in returns:
    plt.plot(returns.index, returns[ticker], label=f'{ticker} Stock Return')
plt.title('Stock Returns of Major Airlines')
plt.xlabel('Date')
plt.ylabel('Stock Return')
plt.legend()
plt.show()

# 加载数据
returns = pd.read_csv('market_returns.csv', index_col='Date', parse_dates=True)
returns = returns.loc['2019-12-01':]

plt.figure(figsize=(14, 7))
plt.plot(returns.index, returns, label=f'^GSPC Stock Return')
plt.title('Stock Return of SP500')
plt.xlabel('Date')
plt.ylabel('Stock Return')
plt.legend()
plt.show()

# 加载数据
returns = pd.read_csv('airline_stock_returns.csv', index_col='Date', parse_dates=True)
```

```
104
105 for column in returns.columns:
106     result = adfuller(returns[column].dropna())
107     print(f'ADF Statistic for {column}: {result[0]}')
108     print(f'p-value: {result[1]}')
109
110 # 切分训练集和测试集
111 train_size = int(len(returns) * 0.8)
112 train, test = returns[:train_size], returns[train_size:]
113
114 # 加载数据
115 returns = pd.read_csv('airline_stock_returns.csv', index_col='Date', parse
        _dates=True)
116
117 # 切分训练集和测试集
118 train_size = int(len(returns) * 0.8)
119 train, test = returns[:train_size], returns[train_size:]
120
121 # VAR模型
122 model = VAR(train)
123 model_fit = model.fit(ic='aic')
124
125 # 查询VAR模型阶数
126 print(f'The order of the fitted VAR model is: {model_fit.k_ar}')
127 print(model_fit.summary())
128
129 # 预测
130 lag_order = model_fit.k_ar
131 forecast_input = train.values[-lag_order:]
132 forecast_steps = len(test)
133 forecast = model_fit.forecast(y=forecast_input, steps=forecast_steps)
134 forecast_df = pd.DataFrame(forecast, index=test.index, columns=train.
        columns)
135
136 fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10), dpi=100)
137 fig.tight_layout(pad=5.0)
138
139 for i, column in enumerate(train.columns):
```

```
        ax = axes[i//2, i%2]
        ax.plot(train.index, train[column], label='Train')
        ax.plot(test.index, test[column], label='Test')
        ax.plot(forecast_df.index, forecast_df[column], label='Forecast')
        ax.set_title(f'VAR Model Forecast for {column} Stock Return')
        ax.set_xlabel('Date')
        ax.set_ylabel('Return')
        ax.legend()

    plt.show()

    # 数据预处理
    def preprocess_data(data, look_back=10):
        scaler = MinMaxScaler(feature_range=(0, 1))
        scaled_data = scaler.fit_transform(data)

        X, y = [], []
        for i in range(len(scaled_data) - look_back):
            X.append(scaled_data[i:i + look_back])
            y.append(scaled_data[i + look_back])
        return np.array(X), np.array(y), scaler

    look_back = 10
    X_train, y_train, scaler = preprocess_data(train, look_back)
    X_test, y_test, _ = preprocess_data(test, look_back)

    # LSTM模型
    def build_lstm_model(input_shape):
        model = Sequential()
        model.add(LSTM(50, return_sequences=True, input_shape=input_shape))
        model.add(LSTM(50))
        model.add(Dense(input_shape[1]))
        model.compile(optimizer='adam', loss='mean_squared_error')
        return model

    lstm_model = build_lstm_model((look_back, X_train.shape[2]))
    lstm_model.fit(X_train, y_train, epochs=20, batch_size=32, verbose=1)
```

```
# RNN模型
def build_rnn_model(input_shape):
    model = Sequential()
    model.add(SimpleRNN(50, return_sequences=True, input_shape=input_shape))
    model.add(SimpleRNN(50))
    model.add(Dense(input_shape[1]))
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

rnn_model = build_rnn_model((look_back, X_train.shape[2]))
rnn_model.fit(X_train, y_train, epochs=20, batch_size=32, verbose=1)

def plot_predictions(train, test, predictions, title, columns):
    fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10), dpi=100)
    fig.tight_layout(pad=5.0)

    for i, column in enumerate(columns):
        ax = axes[i//2, i%2]
        ax.plot(train.index, train[column], label='Train')
        ax.plot(test.index, test[column], label='Test')
        ax.plot(test.index[:len(predictions)], predictions[:, i], label='Forecast')
        ax.set_title(f'{title} for {column} Stock Return')
        ax.set_xlabel('Date')
        ax.set_ylabel('Return')
        ax.legend()

    plt.show()

# LSTM预测
lstm_predictions = lstm_model.predict(X_test)
lstm_predictions = scaler.inverse_transform(lstm_predictions)
plot_predictions(train, test, lstm_predictions, 'LSTM Forecast', returns.columns)

# RNN预测
rnn_predictions = rnn_model.predict(X_test)
```

```
rnn_predictions = scaler.inverse_transform(rnn_predictions)
plot_predictions(train, test, rnn_predictions, 'RNN Forecast', returns.columns)

# 计算误差
def calculate_errors(true_values, predictions):
    mse = mean_squared_error(true_values, predictions)
    mae = mean_absolute_error(true_values, predictions)
    rmse = np.sqrt(mse)
    return mse, mae, rmse

errors = {}
for column in returns.columns:
    true_values = test[column].values
    errors[column] = {
        'VAR': calculate_errors(true_values, forecast_df[column].values),
        'LSTM': calculate_errors(true_values[:len(lstm_predictions)], lstm_predictions[:, returns.columns.get_loc(column)]),
        'RNN': calculate_errors(true_values[:len(rnn_predictions)], rnn_predictions[:, returns.columns.get_loc(column)])
    }

# 输出误差
for column in returns.columns:
    print(f"Errors for {column}:")
    print(f"  VAR - MSE: {errors[column]['VAR'][0]}, MAE: {errors[column]['VAR'][1]}, RMSE: {errors[column]['VAR'][2]}")
    print(f"  LSTM - MSE: {errors[column]['LSTM'][0]}, MAE: {errors[column]['LSTM'][1]}, RMSE: {errors[column]['LSTM'][2]}")
    print(f"  RNN - MSE: {errors[column]['RNN'][0]}, MAE: {errors[column]['RNN'][1]}, RMSE: {errors[column]['RNN'][2]}")

# 加载数据
returns = pd.read_csv('airline_stock_returns.csv', index_col='Date', parse_dates=True)
risk_free_rate = pd.read_csv('risk_free_rate.csv', index_col='DATE', parse_dates=True)
risk_free_rate = risk_free_rate.resample('D').ffill() / 100
```

```
243
244 # 数据对齐
245 returns, risk_free_rate = returns.align(risk_free_rate, join='inner', axis
    =0)
246 daily_risk_free_rate = risk_free_rate['Risk_Free_Rate'].values
247
248 # 假设无风险资产的投资比例为20%
249 risk_free_ratio = 0.2
250 risk_aversion = 1.0  # 风险厌恶系数
251
252 def portfolio_optimization(returns, risk_free_rate, risk_free_ratio, risk_
    aversion):
253     def portfolio_statistics(weights):
254         portfolio_return = np.sum(returns.mean() * weights) * 252
255         portfolio_stddev = np.sqrt(np.dot(weights.T, np.dot(returns.cov(),
              weights))) * np.sqrt(252)
256         sharpe_ratio = (portfolio_return - risk_free_rate.mean() * 252) /
             portfolio_stddev
257         utility = portfolio_return - (risk_aversion / 2) * (portfolio_
            stddev ** 2)
258         return np.array([portfolio_return, portfolio_stddev, sharpe_ratio,
              utility])
259
260     def min_func_utility(weights):
261         return -portfolio_statistics(weights)[3]
262
263     num_assets = len(returns.columns)
264     cons = ({'type': 'eq', 'fun': lambda x: np.sum(x) - (1 - risk_free_
        ratio)})
265     bounds = tuple((-1, 1) for _ in range(num_assets))
266     result = sco.minimize(min_func_utility, num_assets * [1. / num_assets
        ], bounds=bounds, constraints=cons)
267     return result, portfolio_statistics(result.x)
268
269 opt_result, opt_stats = portfolio_optimization(returns, daily_risk_free_
    rate, risk_free_ratio, risk_aversion)
270 print("Optimized portfolio weights (excluding risk-free asset):", opt_
    result.x)
```

```
271
272 opt_weights = np.append(opt_result.x * (1 - risk_free_ratio), risk_free_
        ratio)
273 asset_labels = list(returns.columns) + ['Risk-Free Asset']
274
275 # 输出优化结果
276 opt_return, opt_stddev, opt_sharpe, opt_utility = opt_stats
277
278 print(f"Optimized Portfolio Return: {opt_return:.2f}")
279 print(f"Optimized Portfolio Standard Deviation: {opt_stddev:.2f}")
280 print(f"Optimized Portfolio Sharpe Ratio: {opt_sharpe:.2f}")
281 print(f"Optimized Portfolio Utility: {opt_utility:.2f}")
282
283 # 绘制有效前沿，包括无风险资产
284 def plot_efficient_frontier(returns, risk_free_rate):
285     port_returns = []
286     port_volatility = []
287     port_weights = []
288     num_assets = len(returns.columns)
289     num_portfolios = 5000
290
291     for _ in range(num_portfolios):
292         weights = np.random.random(num_assets)
293         weights /= np.sum(weights)
294         returns_portfolio = np.dot(weights, returns.mean()) * 252
295         volatility_portfolio = np.sqrt(np.dot(weights.T, np.dot(returns.
                cov() * 252, weights)))
296         port_returns.append(returns_portfolio)
297         port_volatility.append(volatility_portfolio)
298         port_weights.append(weights)
299
300     portfolio_data = {'Returns': port_returns, 'Volatility': port_
            volatility}
301     for counter, symbol in enumerate(returns.columns.tolist()):
302         portfolio_data[symbol + ' Weight'] = [weight[counter] for weight
                in port_weights]
303
304     df = pd.DataFrame(portfolio_data)
```

```
305
306         plt.scatter(df['Volatility'], df['Returns'], c=df['Returns'] / df['
                Volatility'], marker='o')
307         plt.xlabel('Volatility')
308         plt.ylabel('Returns')
309         plt.title('Efficient Frontier')
310         plt.colorbar(label='Sharpe Ratio')
311
312         #optimal_weights = opt_result.x
313         #optimal_return = np.dot(optimal_weights, returns.mean()) * 252
314         #optimal_volatility = np.sqrt(np.dot(optimal_weights.T, np.dot(returns
                .cov() * 252, optimal_weights)))
315         #plt.scatter(optimal_volatility, optimal_return, marker='*', color='r
                ', s=200, label='Optimal Portfolio')
316
317         #rf = risk_free_rate.mean() * 252
318         #slope = (optimal_return - rf) / optimal_volatility
319         #x = np.linspace(0, max(df['Volatility']), 100)
320         #y = rf + slope * x
321         #plt.plot(x, y, linestyle='--', color='g', label='Capital Market Line
                ')
322
323         plt.legend()
324         plt.show()
325
326 plot_efficient_frontier(returns, daily_risk_free_rate)
327
328 # 加载数据
329 #returns = pd.read_csv('airline_stock_returns.csv', index_col='Date',
        parse_dates=True)
330 #ff_factors = pd.read_csv('ff_factors.csv', index_col='Date', parse_dates=
        True)
331
332 #merged_data = pd.merge(returns, ff_factors, left_index=True, right_index=
        True)
333
334 # Fama-French回归
335 #def fama_french_analysis(stock_returns, factors):
```

```
#     X = sm.add_constant(factors[['Mkt-RF', 'SMB', 'HML']])
#     ff_results = {}
#     for ticker in stock_returns.columns:
#         y = stock_returns[ticker] - factors['RF']
#         model = sm.OLS(y, X).fit()
#         ff_results[ticker] = model.summary()
#     return ff_results

#ff_results = fama_french_analysis(returns, merged_data)
#for ticker, result in ff_results.items():
#     print(f"Fama-French results for {ticker}:")
#     print(result)
```

## 7.2 参考文献


[1] 陈泽庆.(2023). 突发公共卫生事件对 A 股交通运输业股票价格的影响 (硕士学位论文, 吉林财经大学).

[2] 刘宇.(2023). 新冠疫情下航运上市公司股价与 BDI 关联性分析 (硕士学位论文, 大连海事大学).

[3] Nguyen, Huan & Phan, Truc & Ngo, Vu. (2022). The Impact of the COVID-19 Pandemic on the Performance and Risk Profile of Airline Stocks. Contemporary Economics. 16. 257-275. 10.5709/ce.1897-9254.481.