Spring 5-2024

A Novel Correction for the Multivariate Ljung-Box Test

Minhao Huang
Chapman University, minhuang@chapman.edu

A Novel Correction for the Multivariate Ljung-Box Test

A Thesis by

Minhao Huang

Chapman University

Orange, CA

Schmid College of Science and Technology

Submitted in partial fulfillment of the requirements for the degree of

Master of Science in Computational and Data Sciences

May 2024

Committee in charge:

Cyril Rakovski, Ph.D., Chair

Adrian Vajiac, Ph.D.

Sidy Danioko, Ph.D.

CHAPMAN UNIVERSITY
SCHMID COLLEGE OF SCIENCE AND TECHNOLOGY
Computational and Data Sciences

The thesis of Minhao Huang is approved.

Cyril Rakovski

Cyril Rakovski, Ph.D., Chair

Adrian Vajiac

Adrian Vajiac, Ph.D.

Danioko Sidy

Sidy Danioko, Ph.D.

April 2024

A Novel Correction for the Multivariate Ljung-Box Test

by Minhao Huang

ABSTRACT

A Novel Correction for the Multivariate Ljung-Box Test
by Minhao Huang

This research introduces an analytical improvement to the Multivariate Ljung-Box test that addresses significant deviations of the original test from the nominal Type I error rates under almost all scenarios. Prior attempts to mitigate this issue have been directed at modification of the test statistics or correction of the test distribution to achieve precise results in finite samples. In previous studies, focused on designing corrections to the univariate Ljung-Box, a method that specifically adjusts the test rejection region has been the most successful of attaining the best Type I error rates. We adopt the same approach for the more complex, multidimensional time series scenarios. We use large sample simulation data for a range of values of sample sizes, lags, and number of time series to obtain an empirical estimation of the correct rejection regions for the particular combination of values of these variables. Furthermore, we use a regression modeling with interactions and covariate power combinations to parametrically extend these precise rejection regions to all combination of values of sample sizes, lags, and number of time series. Our results show that we attain almost perfect Type I error rates across all scenarios. These findings will improve the goodness-of-fit diagnostics for multivariate time series analysis.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Meaning
ARMA	Auto Regressive Moving Average
CCM	Cross-Covariance Matrix
ToC	Table of Contents

LIST OF SYMBOLS

Symbol	Meaning
μ	Mean Vector
Γ	Cross-Covariance Matrix
Σ	Covariance Matrix
tr	Trace Operation for Matrices
Y_t	Univariate Time Series
Z_t	Multivariate Time Series
Cov	Covariance Operation for Random Variables and Random Vectors
e_t	White Noise
E	Expectation
$Q_k(m)$	Multivariate Ljung-Box Test Statistic

1 Introduction

A time series is a realization of a discrete time stochastic process, where each data point represents a single random variable realization at a particular time point. This type of time series analysis is focused on understanding and modeling the behavior of that single variable over time, often to forecast future values based on historical patterns. For example, in univariate time series analysis, we follow the classical notation to let $\{Y_t\}$ be an observed time series and $\{e_t\}$ be a white noise, that is a sequence of identically distributed, zero mean, random variable.

$$Y_t = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \cdots + \varphi_p Y_{t-p} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \cdots - \theta_q e_{t-q} \quad (1)$$

A multivariate time series is a collection of observations on two or more related processes at consistent time intervals. This type of time series analysis extends beyond univariate analysis, which focuses on a single variable over time. Multivariate time series can capture complex interdependencies and interactions among several variables, mirroring real-world situations where decisions are often based on multiple interconnected factors. Grasping these interrelationships and making precise forecasts of these variables is crucial for informed decision-making. For example:

$$Z_t = \mu + \sum_{i=0}^{\infty} \psi_i a_{t-i} \quad (2)$$

where μ is a k-dimensional constant vector, $\psi_0 = I_k$, a $k \times k$ identity matrix, $\psi_i (i > 0)$ are $k \times k$ constant matrices, and $\{a_t\}$ is a sequence of independent and identically distributed random vectors with mean zero and a positive-definite covariance matrix Σ_a. In this research, we define

$\{Z_{it}\}$ as the i-th component of the multivariate time series $\{Z_t\}$ to explore the dynamic interplay among the series' elements and to enhance the forecasting of $\{Z_{it}\}$by leveraging the collective data from all components of the series. Formula (3) shows how each component of $\{Z_t\}$ is structured in a matrix form.

$$Z_t = \begin{pmatrix} Z_{1t} \\ Z_{2t} \\ \vdots \\ Z_{kt} \end{pmatrix} \tag{3}$$

Stationarity plays a pivotal role in both univariate and multivariate time series analysis because most modeling techniques assume that the statistical properties of the process generating the data do not change over time. Stationarity implies that the model capturing the relationships between variables will remain valid for forecasts. When stationarity is not present, transformations such as differencing, detrending, or even more complex methods like cointegration analysis are often applied to stabilize the statistical properties of the series over time.

Specifically, in a univariate case, the process $\{Y_t\}$ is strictly stationary if the joint distribution of $\{Y_{t_1}, Y_{t_2}, Y_{t_3} \ldots\}$ is identical comparing the joint distribution of $\{Y_{t_1-p}, Y_{t_2-p}, Y_{t_3-p} \ldots\}$ for all choices of time periods and all choices of time lag *p*. For practical reasons, these conditions are often too strict to achieve; therefore, in practice, we mainly focus on weakly stationary processes. A weakly stationary time series is realized when it satisfies the three conditions: the mean is constant over time, the variance is constant over time, and the

autocovariance between two time points only depends on time lag *k* and not the actual time *t* at which the covariance is computed.

Furthermore, for a k-dimensional time series to be strictly stationary, the statistical properties of the series are invariant of any time shifts, regardless of the length of any time shifts. Precisely, joint probability distribution of any set of observations must be the same as the distribution of the same set shifted in time by any number of periods. This applies to every collection of time points and their distributions, not just the individual series within the multivariate time series framework. In this research, we will use a weaker or second-order stationarity, which follows (a) $E(Z_t) = \mu$, a k-dimensional constant vector, and (b) $Cov(Z_t) = E[(Z_t - \mu)(Z_t - u)^T] = \Sigma_Z$, a constant $k \times k$ positive-definite matrix for practical reasons due to the nature of strictly stationary processes. Here $E(Z_t)$ and $Cov(Z_t)$ denote the expectation and the covariance matrices of the $\{Z_t\}$ respectively. Therefore, the mean and the covariance matrices are independent of time. In order to measure the linear dynamic of a stationary time series $\{Z_t\}$, here we define the lag l cross-covariance matrix (CCM) as:

$$\begin{aligned} \Gamma_l &= Cov(Z_t, Z_{t-l}) = E[(Z_t - \mu)(Z_{t-l} - \mu)^T] \\ &= \begin{bmatrix} E(\tilde{Z}_{1t}\tilde{Z}_{1,t-l}) & E(\tilde{Z}_{1t}\tilde{Z}_{2,t-l}) & \cdots & E(\tilde{Z}_{1t}\tilde{Z}_{k,t-l}) \\ \vdots & \vdots & \ddots & \cdots \\ E(\tilde{Z}_{kt}\tilde{Z}_{1,t-l}) & E(\tilde{Z}_{kt}\tilde{Z}_{2,t-l}) & \cdots & E(\tilde{Z}_{kt}\tilde{Z}_{k,t-l}) \end{bmatrix} \end{aligned} \quad (4)$$

where $\mu = E(Z_t)$ is the mean vector of $\{Z_t\}$ and $\tilde{Z}_t = (\tilde{Z}_{1t}, \ldots, \tilde{Z}_{kt}) \equiv Z_t - \mu$ is the mean-adjusted time series. Since the time series is stationary, the CCM is function of lag l instead

of time *t*. When given the sample $\{Z_t\}_{t=1}^{n}$, we can obtain the sample mean vector (4) and sample variance (5):

$$\hat{\mu}_Z = \frac{1}{n}\sum_{t=1}^{n} Z_t \quad (5)$$

$$\hat{\Gamma}_0 = \frac{1}{n-1}\sum_{t=1}^{n} (Z_t - \hat{\mu}_Z)(Z_t - \hat{\mu}_Z)^T \quad (6)$$

Similarly, the lag l sample of the CCM (6) is defined as:

$$\hat{\Gamma}_l = \frac{1}{n-1}\sum_{t=1}^{n} \left(Z_t - \hat{\mu}_Z\right)\left(Z_{t-l} - \hat{\mu}_Z\right)^T \quad (7)$$

The original Ljung-Box test statistic is a tool used extensively in time series analysis to check for the presence of autocorrelation at multiple lag lengths in the residuals (error terms) of a model. Essentially, it tests whether there are significant correlations between lagged values of the time series or residuals, indicating that the model may not have fully captured the data's underlying structure. A model that fits well generates residuals lacking any correlation. Therefore, traditional methods for assessing the goodness of fit essentially serve as comprehensive tests for detecting the absence of correlation within the estimated residuals. Consequently, numerous statistical methods have been developed specifically to evaluate whether there is no correlation present among the residuals of a time series model. The Portmanteau test of univariate time series has been extended to the multivariate case by several authors. For instance, Hosking (1980, 1981), Li and McLeod (1981), and Li (2004). Particularly, the multivariate Ljung–Box test statistic is defined as:

$$Q_k(m) = n^2 \sum_{l=1}^{m} \frac{1}{n-l} tr\left(\hat{\Gamma}_l^T \hat{\Gamma}_0^{-1} \hat{\Gamma}_l \hat{\Gamma}_0^{-1}\right) \quad (8)$$

where *tr(A)* is the trace of matrix *A*, *T* represents the transpose of a matrix, and *n* is the sample size.

The test mentioned previously demonstrates departures from the nominal Type I error rates, which affect its effectiveness. In conducting a simulation study to determine the multivariate Ljung-Box test's optimal lag value, it was discovered that the best number of lags is influenced not only by the time series length but also by the test's significance level. Consequently, we adopt the same approach that was proven to be the most precise in previous studies in univariate cases to the multidimensional time series scenarios. This approach focuses on adjusting the rejection region, rather than altering the test statistic itself to achieve the enhancement of the goodness-of-fit analytics for the multivariate time series studies.

2 Methods

A correction to the rejection region of the multivariate Ljung-Box test was developed based on a linear regression model. Subsequently, an extensive simulation study was carried out. This study aimed to calculate the correction and evaluate the improved performance of the corrected approach, specifically in terms of its consistency with the nominal Type I error rates across almost all scenarios.

2.1 Simulation Design

Let *k*, *n*, and *m* denote the multivariate time series dimensions, the sample size, and the number of lagged residuals that we use in the computation of the Ljung-Box statistic respectively. We simulate 1,000,000 datasets for each combination. We consider the following range of values for *k* (*k = 2,3,5,10*), *n* (*n = 40, 50, …, 300*), and *m* (*m = 2, 3, …, n-1*):

1. For each combination of values of *k*, *n*, and *m*, we simulate 1,000,000 independent lists of m standard normal vectors of size *k*.
2. To each of these 1,000,000 datasets, we apply the multivariate Ljung-Box test and extract the corresponding p values.
3. Calculate the Type I error, based on the combination datasets, for each *k*, *n*, and *m* by computing the proportion of p values that are less than 0.05.
4. Arrange the results in rectangular dataset of dimension 18,144 by 4 (the number of possible combinations of *k*, *n*, and *m* over the range of values that we defined above) where the rows denote the values of *k*, *n*, and *m* and the Type I error rates calculated in step 3.

2.2 Linear Model

The idea of this study was to use a linear regression model to apply the correction to the rejection region of the Multivariate Ljung-Box test, in order to improve the Type I error rates for all dimensions (*k=2, 3, 5, and 10*), sample sizes (*n=40, 50, 60,…,300*), and lags (*m=2, 3, … , n-1*). In this study, there were six linear regressions models being created for each value of *k* and trained on the simulated data described in Section 2.1, then we split the time series into subsets of different intervals [0, 50], [51, 70], [71, 90], [91, 120], [121, 200], and [201, 300] for dimensions of 2, 3, and 5. For the 10-dimensional time series, we split the data into intervals of [0, 50], [51, 70], [71, 90], [91, 110], [111, 200], and [201, 300]. Sample sizes selected are shown in Table 1. The variation in patterns of Type I error rates (shown in ***Figures 1 to 4***) across different sample sizes required the adoption of individual models to achieve the targeted level of accuracy. These linear models are intricate, incorporating various degrees of *n* and *m*, as well as their two-way interactions. The general model that we implemented in the study is shown in formula (8).

$$\begin{aligned} Z_t - 0.05 = {} & \alpha_1 n^s + \alpha_2 m^p + \alpha_3 (n^s \times m^p) + \alpha_4 \left(n^{2s} \times m^{2p}\right) + \\ & \alpha_5 n^{2s} + \alpha_6 \left(n^{3s} \times m^{2p}\right) + \alpha_7 \left(n^{3s} \times m^{3p}\right) + \alpha_8 m^{4p} + \alpha_9 m^{5p} \end{aligned} \tag{9}$$

Additionally, a comprehensive grid search was conducted within the general formula (8) to identify the optimal values for the power transformation parameters *s* and *p*. The summary statistics of each best-fit model are presented in the tables within Chapter 3 of this research paper, along with the specific values for *s* and *p*.

Table 1 Summary of the Subsets for Different Dimensions

Dimensions of the Time Series (k)	Subsets of Sample Sizes (n)					
2	0-50	51-70	71-90	91-120	121-200	201-300
3	0-50	51-70	71-90	91-120	121-200	201-300
5	0-50	51-70	71-90	91-120	121-200	201-300
10	0-50	51-70	71-90	91-110	111-200	201-300

Figure 1 Patterns of Type I Error Rates for k = 2

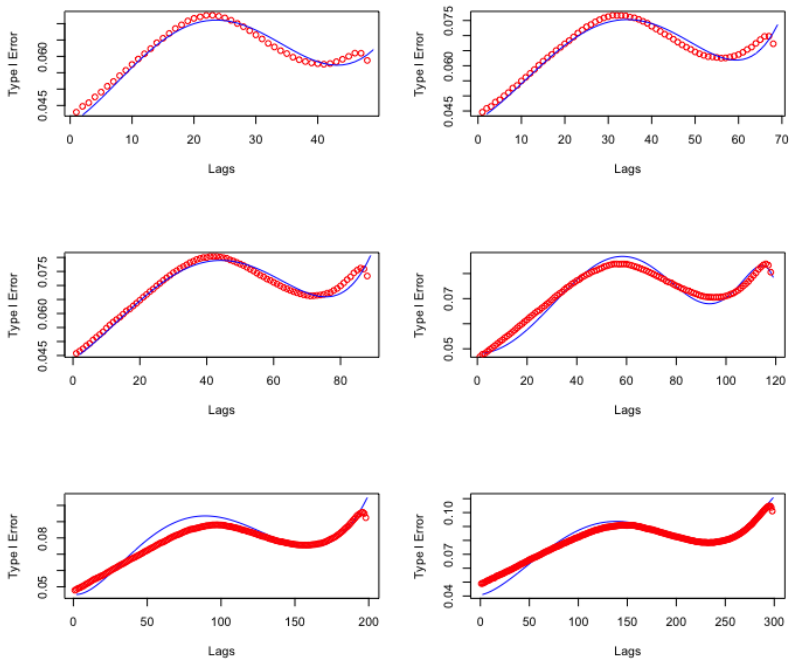


Figure 2 Patterns of Type I Error Rates for k = 3

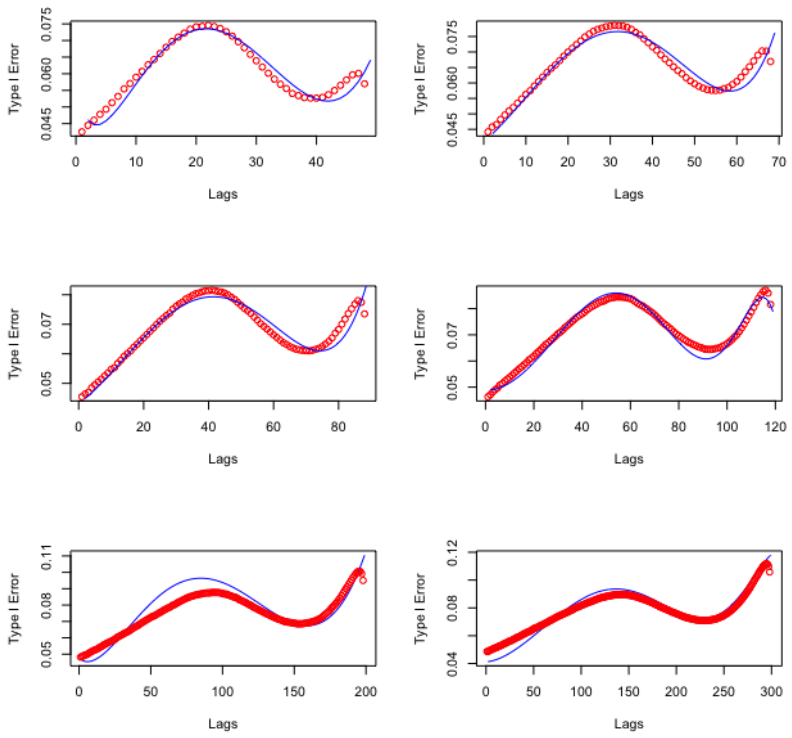


Figure 3 Patterns of Type I Error Rates for k = 5

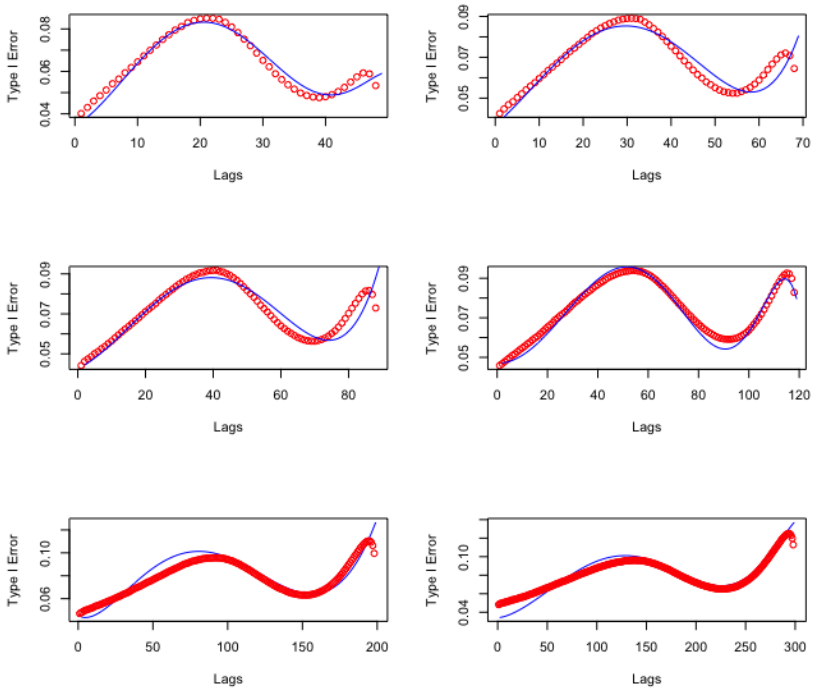
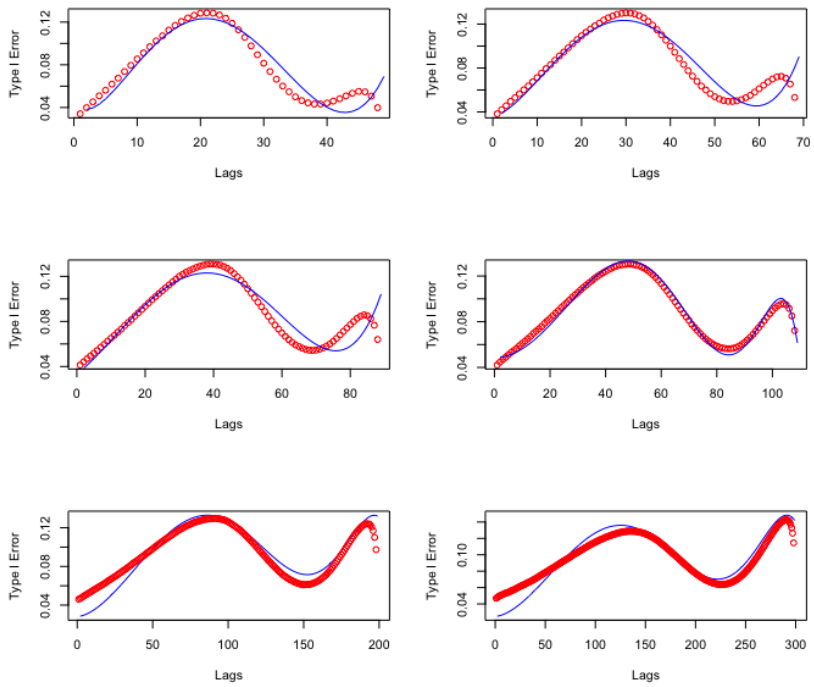


Figure 4 Patterns of Type I Error Rates for k = 10



For Figures 1-4, the blue lines represent the general model described in formula (8), where the red lines are the data obtained from the simulation. Specific *s* and *p* values for each individual model will be presented in the next chapter of this research paper.

3 Results

While conducting our study, it was evident that there are apparent differences in the patterns of the Type I error rates. Consequently, individual models were built according to subsets of the sample sizes shown in Table 1. For each model, the optimal *s* and *p* values in the general models were obtained by doing a grid search and presented in Tables 2, 9, 16, and 23.

Table 2 Model Summary for k=2

Sample Sizes (n)	s	p
45	1.8	1.3
65	2.0	1.1
85	2.0	1.1
100	2.0	2.0
160	0.3	0.9
280	1.5	1.5

Table 3 Summary Statistics for Selected Variables, n < 50 and k = 2

Variables	Estimate	Std. Error	t-value	p-value
n^s	-3.71E-05	2.77E-06	-13.37	< 2e-16***
m^p	2.51E-03	1.71E-04	14.672	< 2e-16***
$n^s \times m^p$	-1.37E-06	1.85E-07	-7.403	1.42E-10***
$n^{2s} \times m^{2p}$	-8.71E-11	3.71E-12	-23.46	< 2e-16***
n^{2s}	2.46E-08	2.70E-09	9.134	6.61E-14***
$n^{3s} \times m^{2p}$	7.45E-14	3.70E-15	20.116	< 2e-16***
$n^{3s} \times m^{3p}$	-8.68E-17	7.17E-18	-12.103	< 2e-16***
m^{4p}	1.07E-09	6.59E-11	16.256	< 2e-16***
m^{5p}	-2.21E-12	2.50E-13	-8.865	2.18E-13***

Table 4 Summary Statistics for Selected Variables, 50 < n < 70 and k = 2

Variables	Estimate	Std. Error	t-value	p-value
n^s	-8.99E-06	9.09E-07	-9.888	< 2e-16***
m^p	4.35E-03	3.27E-04	13.317	< 2e-16***
$n^s \times m^p$	-7.12E-07	8.05E-08	-8.843	1.14E-14***
$n^{2s} \times m^{2p}$	-6.84E-12	4.22E-13	-16.213	< 2e-16***
n^{2s}	1.50E-09	2.05E-10	7.343	3.02E-11***
$n^{3s} \times m^{2p}$	1.49E-15	9.72E-17	15.364	< 2e-16***
$n^{3s} \times m^{3p}$	-3.37E-18	2.67E-19	-12.629	< 2e-16***
m^{4p}	1.28E-09	3.13E-10	4.09	7.95E-05***
m^{5p}	9.21E-12	1.98E-12	4.64	9.14E-06***

Table 5 Summary Statistics for Selected Variables, 70 < n < 90 and k = 2

Variables	Estimate	Std. Error	t-value	p-value
n^s	-4.63E-06	5.52E-07	-8.392	2.62E-14***
m^p	3.38E-03	2.57E-04	13.163	< 2e-16***
$n^s \times m^p$	-3.28E-07	3.70E-08	-8.862	1.61E-15***
$n^{2s} \times m^{2p}$	-1.29E-12	7.76E-14	-16.655	< 2e-16***
n^{2s}	4.77E-10	7.44E-11	6.419	1.55E-09***
$n^{3s} \times m^{2p}$	1.70E-16	1.09E-17	15.596	< 2e-16***
$n^{3s} \times m^{3p}$	-3.12E-19	2.26E-20	-13.791	< 2e-16***
m^{4p}	3.52E-10	9.16E-11	3.837	0.00018***
m^{5p}	2.69E-12	4.25E-13	6.327	2.49E-09***

Table 6 Summary Statistics for Selected Variables, 90 < n < 120 and k = 2

Variables	Estimate	Std. Error	t-value	p-value
n^s	2.82E-06	1.71E-07	16.505	<2.00E-16***
m^p	-1.33E-05	1.09E-06	-12.253	<2.00E-16***
$n^s \times m^p$	2.71E-09	1.10E-10	24.544	<2.00E-16***
$n^{2s} \times m^{2p}$	-3.98E-17	1.63E-18	-24.479	<2.00E-16***
n^{2s}	-2.05E-10	1.35E-11	-15.204	<2.00E-16***
$n^{3s} \times m^{2p}$	1.13E-21	1.28E-22	8.863	<2.00E-16***
$n^{3s} \times m^{3p}$	3.89E-26	4.95E-27	7.866	1.50E-13***
m^{4p}	2.76E-17	1.13E-18	24.559	<2.00E-16***
m^{5p}	-1.40E-21	6.32E-23	-22.157	<2.00E-16***

Table 7 Summary Statistics for Selected Variables, 120< n < 200 and k = 2

Variables	Estimate	Std. Error	t-value	p-value
n^s	4.57E-03	1.31E-03	3.492	0.000495***
m^p	7.11E-03	4.15E-04	17.117	< 2e-16***
$n^s \times m^p$	-1.45E-03	9.54E-05	-15.155	< 2e-16***
$n^{2s} \times m^{2p}$	-9.22E-06	3.51E-07	-26.259	< 2e-16***
n^{2s}	-1.16E-03	2.85E-04	-4.058	5.25E-05***
$n^{3s} \times m^{2p}$	2.38E-06	8.84E-08	26.971	< 2e-16***
$n^{3s} \times m^{3p}$	-9.59E-09	3.03E-10	-31.645	< 2e-16***
m^{4p}	5.94E-09	2.67E-10	22.285	< 2e-16***
m^{5p}	-2.70E-12	9.80E-13	-2.756	0.005928**

Table 8 Summary Statistics for Selected Variables, n > 200 and k = 2

Variables	Estimate	Std. Error	t-value	p-value
n^s	1.00E-05	2.00E-07	50.307	<2e-16***
m^p	-2.22E-05	9.53E-07	-23.321	<2e-16***
$n^s \times m^p$	1.82E-08	3.28E-10	55.568	<2e-16***
$n^{2s} \times m^{2p}$	-2.60E-15	3.58E-17	-72.599	<2e-16***
n^{2s}	-2.27E-09	4.67E-11	-48.582	<2e-16***
$n^{3s} \times m^{2p}$	3.09E-19	7.29E-21	42.368	<2e-16***
$n^{3s} \times m^{3p}$	-3.09E-24	9.32E-25	-3.314	0.000931
m^{4p}	1.34E-15	2.77E-17	48.205	<2e-16***
m^{5p}	-1.31E-19	4.68E-21	-27.957	<2e-16***

Table 9 Model Summary for k=3

Sample Sizes (n)	s	p
45	1.3	0.7
65	2.0	1.1
85	2.0	1.1
100	0.4	2.0
160	0.3	0.9
280	2.0	1.7

Table 10 Summary Statistics for Selected Variables, n < 50 and k = 3

Variables	Estimate	Std. Error	t-value	p-value
n^s	-9.19E-04	1.22E-04	-7.549	7.47E-11***
m^p	6.39E-02	7.78E-03	8.218	3.86E-12***
$n^s \times m^p$	-5.04E-04	5.86E-05	-8.586	7.53E-13***
$n^{2s} \times m^{2p}$	-1.63E-07	5.08E-08	-3.204	0.00197**
n^{2s}	6.19E-06	8.39E-07	7.378	1.59E-10***
$n^{3s} \times m^{2p}$	2.13E-09	3.54E-10	6.008	5.85E-08***
$n^{3s} \times m^{3p}$	-6.15E-11	6.14E-12	-10.018	1.34E-15***
m^{4p}	-1.23E-05	1.66E-06	-7.423	1.31E-10***
m^{5p}	9.14E-07	6.99E-08	13.077	< 2e-16***

Table 11 Summary Statistics for Selected Variables, 50 < n < 70 and k = 3

Variables	Estimate	Std. Error	t-value	p-value
n^s	-1.09E-05	1.36E-06	-8.059	7.34E-13***
m^p	5.46E-03	4.88E-04	11.189	< 2e-16***
$n^s \times m^p$	-9.01E-07	1.20E-07	-7.494	1.39E-11***
$n^{2s} \times m^{2p}$	-9.38E-12	6.30E-13	-14.88	< 2e-16***
n^{2s}	1.86E-09	3.06E-10	6.098	1.42E-08***
$n^{3s} \times m^{2p}$	2.02E-15	1.45E-16	13.92	< 2e-16***
$n^{3s} \times m^{3p}$	-4.58E-18	3.99E-19	-11.478	< 2e-16***
m^{4p}	2.51E-09	4.67E-10	5.379	3.89E-07***
m^{5p}	8.08E-12	2.96E-12	2.726	0.0074**

Table 12 Summary Statistics for Selected Variables, 70 < n < 90 and k = 3

Variables	Estimate	Std. Error	t-value	p-value
n^s	-6.41E-06	8.67E-07	-7.393	8.06E-12***
m^p	4.57E-03	4.04E-04	11.328	< 2e-16***
$n^s \times m^p$	-4.73E-07	5.81E-08	-8.134	1.18E-13***
$n^{2s} \times m^{2p}$	-1.79E-12	1.22E-13	-14.701	< 2e-16***
n^{2s}	6.89E-10	1.17E-10	5.891	2.26E-08***
$n^{3s} \times m^{2p}$	2.38E-16	1.72E-17	13.875	< 2e-16***
$n^{3s} \times m^{3p}$	-4.49E-19	3.56E-20	-12.617	< 2e-16***
m^{4p}	6.63E-10	1.44E-10	4.604	8.51E-06***
m^{5p}	3.06E-12	6.68E-13	4.58	9.43E-06***

Table 13 Summary Statistics for Selected Variables, 90 < n < 120 and k = 3

Variables	Estimate	Std. Error	t-value	p-value
n^s	1.34E-02	1.81E-03	7.414	1.14E-12***
m^p	-1.02E-04	1.06E-05	-9.612	< 2e-16***
$n^s \times m^p$	1.93E-05	1.68E-06	11.479	< 2e-16***
$n^{2s} \times m^{2p}$	1.32E-10	4.24E-11	3.117	0.001994**
n^{2s}	-2.00E-03	2.77E-04	-7.231	3.65E-12***
$n^{3s} \times m^{2p}$	-4.14E-11	7.42E-12	-5.577	5.26E-08***
$n^{3s} \times m^{3p}$	1.20E-15	2.09E-16	5.741	2.21E-08***
m^{4p}	1.63E-17	4.77E-18	3.422	0.000703***
m^{5p}	-1.28E-21	1.34E-22	-9.561	< 2e-16***

Table 14 Summary Statistics for Selected Variables, 120< n < 200 and k = 3

Variables	Estimate	Std. Error	t-value	p-value
n^s	-7.18E-04	2.04E-03	-0.352	0.725
m^p	1.33E-02	6.73E-04	19.81	< 2e-16***
$n^s \times m^p$	-2.92E-03	1.55E-04	-18.831	< 2e-16***
$n^{2s} \times m^{2p}$	-1.70E-05	5.90E-07	-28.847	< 2e-16***
n^{2s}	5.32E-05	4.47E-04	0.119	0.905
$n^{3s} \times m^{2p}$	4.48E-06	1.49E-07	30.02	< 2e-16***
$n^{3s} \times m^{3p}$	-1.86E-08	5.11E-10	-36.318	< 2e-16***
m^{4p}	1.25E-08	4.54E-10	27.577	< 2e-16***
m^{5p}	-1.34E-11	1.72E-12	-7.821	1.22e-14 ***

Table 15 Summary Statistics for Selected Variables, n > 200 and k = 3

Variables	Estimate	Std. Error	t-value	p-value
n^s	6.81E-07	1.31E-08	51.965	<2e-16***
m^p	-1.04E-05	3.29E-07	-31.602	<2e-16***
$n^s \times m^p$	4.17E-10	7.51E-12	55.453	<2e-16***
$n^{2s} \times m^{2p}$	-1.19E-18	1.63E-20	-72.703	<2e-16***
n^{2s}	-8.64E-12	1.83E-13	-47.165	<2e-16***
$n^{3s} \times m^{2p}$	7.96E-24	1.74E-25	45.865	<2e-16***
$n^{3s} \times m^{3p}$	1.59E-30	7.71E-30	0.207	0.836
m^{4p}	2.19E-17	4.08E-19	53.727	<2e-16***
m^{5p}	-7.84E-22	2.28E-23	-34.387	<2e-16***

Table 16 Model Summary for k=5

Sample Sizes (n)	s	p
45	1.5	1.4
65	2.0	1.1
85	2.0	1.0
100	0.4	2.0
175	0.3	0.9
280	2.0	1.6

Table 17 Summary Statistics for Selected Variables, n < 50 and k = 5

Variables	Estimate	Std. Error	t-value	p-value
n^s	-1.38E-04	2.19E-05	-6.309	1.64E-08 ***
m^p	3.11E-03	3.20E-04	9.705	5.30E-15 ***
$n^s \times m^p$	-4.98E-06	1.09E-06	-4.552	1.96E-05 ***
$n^{2s} \times m^{2p}$	-8.97E-10	4.42E-11	-20.293	< 2e-16 ***
n^{2s}	2.70E-07	6.85E-08	3.94	0.000178 ***
$n^{3s} \times m^{2p}$	2.40E-12	1.46E-13	16.461	< 2e-16 ***
$n^{3s} \times m^{3p}$	-2.09E-15	2.09E-16	-10.008	1.40E-15 ***
m^{4p}	7.26E-10	3.76E-11	19.321	< 2e-16 ***
m^{5p}	-1.36E-12	8.73E-14	-15.578	< 2e-16 ***

Table 18 Summary Statistics for Selected Variables, 50 < n < 70 and k = 5

Variables	Estimate	Std. Error	t-value	p-value
n^s	-1.59E-05	2.48E-06	-6.415	3.11E-09 ***
m^p	8.31E-03	8.92E-04	9.317	8.85E-16 ***
$n^s \times m^p$	-1.36E-06	2.20E-07	-6.172	9.99E-09 ***
$n^{2s} \times m^{2p}$	-1.54E-11	1.15E-12	-13.328	< 2e-16 ***
n^{2s}	2.72E-09	5.59E-10	4.858	3.72E-06 ***
$n^{3s} \times m^{2p}$	3.24E-15	2.66E-16	12.209	< 2e-16 ***
$n^{3s} \times m^{3p}$	-6.98E-18	7.30E-19	-9.567	2.28E-16 ***
m^{4p}	5.18E-09	8.54E-10	6.06	1.71E-08 ***
m^{5p}	3.84E-12	5.42E-12	0.708	0.48

Table 19 Summary Statistics for Selected Variables, 70 < n < 90 and k = 5

Variables	Estimate	Std. Error	t-value	p-value
n^s	-1.05E-05	1.80E-06	-5.848	2.80E-08***
m^p	1.12E-02	1.24E-03	9.053	5.12E-16***
$n^s \times m^p$	-1.29E-06	1.79E-07	-7.208	2.24E-11***
$n^{2s} \times m^{2p}$	-5.86E-12	5.70E-13	-10.277	< 2e-16***
n^{2s}	1.19E-09	2.43E-10	4.892	2.45E-06***
$n^{3s} \times m^{2p}$	8.35E-16	7.97E-17	10.478	< 2e-16***
$n^{3s} \times m^{3p}$	-2.75E-18	2.51E-19	-10.951	< 2e-16***
m^{4p}	1.41E-09	1.62E-09	0.866	0.388
m^{5p}	8.05E-11	1.18E-11	6.809	1.97E-10***

Table 20 Summary Statistics for Selected Variables, 90 < n < 120 and k = 5

Variables	Estimate	Std. Error	t-value	p-value
n^s	2.26E-02	2.79E-03	8.103	1.20E-14***
m^p	-1.79E-04	1.63E-05	-11	< 2e-16***
$n^s \times m^p$	3.26E-05	2.59E-06	12.601	< 2e-16***
$n^{2s} \times m^{2p}$	3.33E-10	6.54E-11	5.088	6.23E-07***
n^{2s}	-3.39E-03	4.27E-04	-7.955	3.26E-14***
$n^{3s} \times m^{2p}$	-8.39E-11	1.14E-11	-7.343	1.80E-12***
$n^{3s} \times m^{3p}$	2.33E-15	3.22E-16	7.234	3.59E-12***
m^{4p}	1.26E-17	7.35E-18	1.716	0.0871
m^{5p}	-1.66E-21	2.07E-22	-8.026	2.01E-14***

Table 21 Summary Statistics for Selected Variables, 120< n < 200 and k = 5

Variables	Estimate	Std. Error	t-value	p-value
n^s	2.93E-03	2.78E-03	1.055	0.291
m^p	1.60E-02	8.80E-04	18.22	<2e-16***
$n^s \times m^p$	-3.47E-03	2.02E-04	-17.174	<2e-16***
$n^{2s} \times m^{2p}$	-2.14E-05	7.44E-07	-28.727	<2e-16***
n^{2s}	-7.86E-04	6.04E-04	-1.303	0.193
$n^{3s} \times m^{2p}$	5.54E-06	1.87E-07	29.559	<2e-16***
$n^{3s} \times m^{3p}$	-2.28E-08	6.42E-10	-35.552	<2e-16***
m^{4p}	1.62E-08	5.65E-10	28.755	<2e-16***
m^{5p}	-1.99E-11	2.08E-12	-9.567	<2e-16***

Table 22 Summary Statistics for Selected Variables, n > 200 and k = 5

Variables	Estimate	Std. Error	t-value	p-value
n^s	8.62E-07	1.90E-08	45.304	<2e-16***
m^p	-2.80E-05	8.19E-07	-34.21	<2e-16***
$n^s \times m^p$	1.00E-09	1.89E-11	53.08	<2e-16***
$n^{2s} \times m^{2p}$	-4.90E-18	6.92E-20	-70.838	<2e-16***
n^{2s}	-1.15E-11	2.67E-13	-43.244	<2e-16***
$n^{3s} \times m^{2p}$	3.30E-23	7.18E-25	46.017	<2e-16***
$n^{3s} \times m^{3p}$	-4.62E-29	5.76E-29	-0.803	0.422
m^{4p}	2.81E-16	5.41E-18	52.068	<2e-16***
m^{5p}	-1.72E-20	5.39E-22	-31.999	<2e-16***

Table 23 Model Summary for k=10

Sample Sizes (n)	s	p
45	1.4	0.8
65	2.0	0.9
85	2.0	1.0
100	2.0	2.0
185	2.0	1.7
280	2.0	1.7

Table 24 Summary Statistics for Selected Variables, n < 50 and k = 10

Variables	Estimate	Std. Error	t-value	p-value
n^s	-1.72E-03	2.56E-04	-6.729	2.70E-09***
m^p	1.36E-01	1.76E-02	7.759	2.95E-11***
$n^s \times m^p$	-6.27E-04	9.00E-05	-6.967	9.58E-10***
$n^{2s} \times m^{2p}$	-2.53E-07	3.96E-08	-6.39	1.16E-08***
n^{2s}	7.19E-06	1.19E-06	6.03	5.33E-08***
$n^{3s} \times m^{2p}$	1.39E-09	1.88E-10	7.426	1.28E-10***
$n^{3s} \times m^{3p}$	-1.88E-11	2.28E-12	-8.242	3.48E-12***
m^{4p}	-2.07E-06	1.34E-06	-1.539	0.128
m^{5p}	2.54E-07	3.84E-08	6.604	4.62E-09***

Table 25 Summary Statistics for Selected Variables, 50 < n < 70 and k = 10

Variables	Estimate	Std. Error	t-value	p-value
n^s	-4.52E-05	7.66E-06	-5.903	3.57E-08***
m^p	4.37E-02	5.64E-03	7.759	3.54E-12***
$n^s \times m^p$	-8.76E-06	1.39E-06	-6.289	5.71E-09***
$n^{2s} \times m^{2p}$	-1.16E-10	1.59E-11	-7.279	4.19E-11***
n^{2s}	8.62E-09	1.73E-09	4.989	2.13E-06***
$n^{3s} \times m^{2p}$	2.86E-14	3.59E-15	7.969	1.18E-12***
$n^{3s} \times m^{3p}$	-1.81E-16	2.14E-17	-8.447	9.44E-14***
m^{4p}	-9.48E-08	6.18E-08	-1.534	0.128
m^{5p}	6.19E-09	9.25E-10	6.692	8.00E-10***

Table 26 Summary Statistics for Selected Variables, 70 < n < 90 and k = 10

Variables	Estimate	Std. Error	t-value	p-value
n^s	-2.03E-05	4.04E-06	-5.017	1.41E-06***
m^p	2.13E-02	2.78E-03	7.658	1.82E-12***
$n^s \times m^p$	-2.33E-06	4.00E-07	-5.807	3.43E-08***
$n^{2s} \times m^{2p}$	-1.19E-11	1.28E-12	-9.348	< 2e-16***
n^{2s}	2.24E-09	5.44E-10	4.122	6.06E-05***
$n^{3s} \times m^{2p}$	1.62E-15	1.79E-16	9.082	4.32E-16***
$n^{3s} \times m^{3p}$	-4.75E-18	5.62E-19	-8.451	1.85E-14***
m^{4p}	6.13E-09	3.63E-09	1.688	0.0934
m^{5p}	1.12E-10	2.65E-11	4.22	4.12E-05***

Table 27 Summary Statistics for Selected Variables, 90 < n < 110 and k = 10

Variables	Estimate	Std. Error	t-value	p-value
n^s	6.56E-06	1.08E-06	6.077	6.21E-09***
m^p	-2.82E-05	1.30E-05	-2.174	0.0309*
$n^s \times m^p$	8.96E-09	1.22E-09	7.351	5.15E-12***
$n^{2s} \times m^{2p}$	-2.89E-16	2.25E-17	-12.839	< 2e-16***
n^{2s}	-5.52E-10	9.61E-11	-5.746	3.43E-08***
$n^{3s} \times m^{2p}$	1.20E-20	2.19E-21	5.469	1.36E-07***
$n^{3s} \times m^{3p}$	3.05E-25	6.31E-26	4.827	2.77E-06***
m^{4p}	2.12E-16	9.17E-18	23.158	< 2e-16***
m^{5p}	-1.31E-20	4.35E-22	-30.107	< 2e-16***

Table 28 Summary Statistics for Selected Variables, 110 < n < 200 and k = 10

Variables	Estimate	Std. Error	t-value	p-value
n^s	3.95E-06	1.18E-07	33.57	<2e-16***
m^p	-5.70E-05	2.26E-06	-25.29	<2e-16***
$n^s \times m^p$	4.37E-09	1.33E-10	32.84	<2e-16***
$n^{2s} \times m^{2p}$	-5.82E-17	1.59E-18	-36.548	<2e-16***
n^{2s}	-1.13E-10	3.79E-12	-29.718	<2e-16***
$n^{3s} \times m^{2p}$	8.91E-22	3.47E-23	25.687	<2e-16***
$n^{3s} \times m^{3p}$	2.76E-27	2.95E-27	0.936	0.349
m^{4p}	8.60E-16	3.05E-17	28.161	<2e-16***
m^{5p}	-6.55E-20	3.28E-21	-19.955	<2e-16***

Table 29 Summary Statistics for Selected Variables, n > 200 and k = 10

Variables	Estimate	Std. Error	t-value	p-value
n^s	1.59E-06	3.46E-08	45.959	<2e-16***
m^p	-3.61E-05	8.69E-07	-41.532	<2e-16***
$n^s \times m^p$	1.14E-09	1.98E-11	57.66	<2e-16***
$n^{2s} \times m^{2p}$	-2.93E-18	4.32E-20	-67.956	<2e-16***
n^{2s}	-2.08E-11	4.84E-13	-42.929	<2e-16***
$n^{3s} \times m^{2p}$	1.73E-23	4.58E-25	37.74	<2e-16***
$n^{3s} \times m^{3p}$	1.78E-28	2.04E-29	8.763	<2e-16***
m^{4p}	6.16E-17	1.08E-18	57.1	<2e-16***
m^{5p}	-2.54E-21	6.02E-23	-42.184	<2e-16***

Each model summary across varying sample sizes and dimensions indicates exceptionally low p-values for the covariates. Symbols such as *, **, *** indicate how significantly each variable contributes to the respective models, which span different lengths and dimensions of time series data. These symbols consistently suggest strong significance for most variables. The visual comparisons (provided in the previous chapter of this paper) also suggest that models based on linear regression are superior in terms of Type I error rates. From the empirical data, it appears the models that fit the best have been successfully identified.

4 Data Example

An application of the correction to the multivariate Ljung-Box test was done using ten Fortune 500 stock data obtained from Yahoo finance. Here, we provide an example of a false positive and false negative results using the classical multivariate Ljung-Box test on these ten sets of stock data. For a sample size of 50, numerous false positive and false negative points were discovered at different lags.

An instance of false negative is found at lag 35 (red dot), where the p values were obtained by the standard multivariate Ljung-Box test. By the proposed correction to the original test, the false positive p value of 0.0614 is inflated to 0.0646. By using another timeframe from the same stocks, an instance of a false positive is also found at lag 39 (blue dot) by the standard multivariate Ljung-Box test. With our proposed correction to the original test, the false negative p value of 0.0472 is shrunk down to 0.0439. Both of the instances are shown in Figure 5 and Figure 6 respectively.

Figure 5 False Negative Correction for Sample Size 50

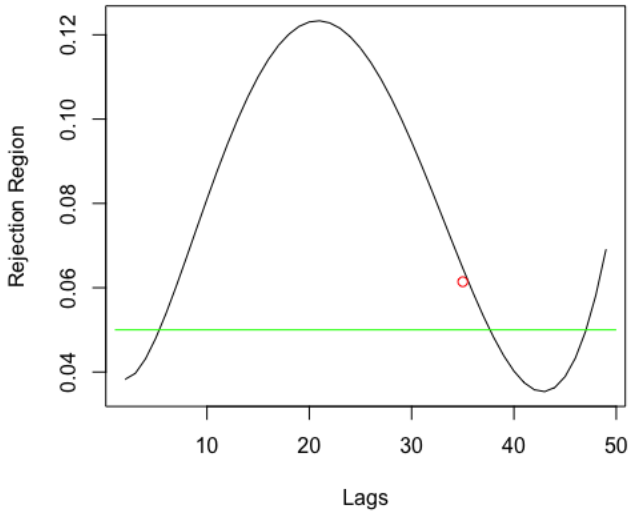
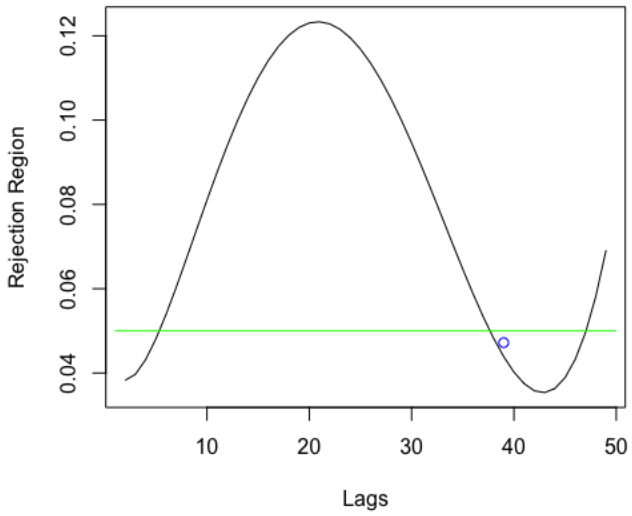


Figure 6 False Positive Correction for Sample Size 50



References

Cryer, J. D., & Chan, K.-s. (2008). *Time series analysis : with applications in R* (2nd ed.). Springer. Table of contents http://www.loc.gov/catdir/toc/fy0804/2008923058.html

Danioko, S., Zheng, J. W., Anderson, K., Barrett, A., & Rakovski, C. S. (2022). A Novel Correction for the Adjusted Box-Pierce Test. *Frontiers in Applied Mathematics and Statistics*, *8*. https://doi.org/ARTN 873746

10.3389/fams.2022.873746

Hosking, J. R. M. (1980). The Multivariate Portmanteau Statistic. *Journal of the American Statistical Association*, *75*(371), 602-608. https://doi.org/Doi 10.2307/2287656

Hosking, J. R. M. (1981). Lagrange-Multiplier Tests of Multivariate Time-Series Models. *Journal of the Royal Statistical Society Series B-Methodological*, *43*(2), 219-230. <Go to ISI>://WOS:A1981LZ09700014

Li, W. K. (2004). *Diagnostic checks in time series*. Chapman & Hall/CRC. http://ed-primo.hosted.exlibrisgroup.com/openurl/44UOE/44UOE_services_page?u.ignore_date_coverage=true&rft.mms_id=9924225507002466

http://whel-primo.hosted.exlibrisgroup.com/openurl/44WHELF_NLW/44WHELF_NLW_services_page?u.ignore_date_coverage=true&rft.mms_id=99934658702419

http://www.crcnetbase.com/isbn/9780203485606

https://eu04.alma.exlibrisgroup.com/view/uresolver/44LIV_INST/openurl?u.ignore_date_coverage=true&portfolio_pid=53276274930007351&Force_direct=true

https://eu04.alma.exlibrisgroup.com/view/uresolver/44LIV_INST/openurl?u.ignore_date_coverage=true&portfolio_pid=53371214200007351&Force_direct=true

https://nls.ldls.org.uk/welcome.html?ark:/81055/vdc_100047262315.0x000001

https://tcdlibrary.ldls.org.uk/vdc_100047262315.0x000001

https://www.taylorfrancis.com/start-session?idp=https%3A%2F%2Fidp.ed.ac.uk%2Fshibboleth&redirectUri=https%3A%2F%2Fwww.taylorfrancis.com%2Fbooks%2F9780429204869

Li, W. K., & Mcleod, A. I. (1981). Distribution of the Residual Autocorrelations in Multivariate Arma Time-Series Models. *Journal of the Royal Statistical Society Series B-Methodological*, *43*(2), 231-239. <Go to ISI>://WOS:A1981LZ09700015

Tsay, R. S., & ProQuest. (2013). *Multivariate Time Series Analysis : With R and Financial Applications* (1st ed.). John Wiley & Sons, Incorporated. http://ebookcentral.proquest.com/lib/abdn/detail.action?docID=1562422

http://ebookcentral.proquest.com/lib/reading/detail.action?docID=7103878

http://ebookcentral.proquest.com/lib/soas-ebooks/detail.action?docID=1562422

http://ebookcentral.proquest.com/lib/warw/detail.action?docID=1562422

http://eu.alma.exlibrisgroup.com/view/uresolver/44SAL_INST/openurl?u.ignore_date_coverage=true&rft.mms_id=9913235819201611

http://eu.alma.exlibrisgroup.com/view/uresolver/44SAL_INST/openurl?u.ignore_date_coverage=true&rft.mms_id=9913692365401611

http://eu.alma.exlibrisgroup.com/view/uresolver/44WHELF_SWA/openurl?u.ignore_date_coverage=true&rft.mms_id=998635452802417
http://imp-primo.hosted.exlibrisgroup.com/openurl/44IMP/44IMP_services_page?u.ignore_date_coverage=true&rft.mms_id=9955886600301591
http://surrey-primotc.hosted.exlibrisgroup.com/openurl/44SUR/44SUR_services_page?u.ignore_date_coverage=true&rft.mms_id=99695034602346
https://birmingham-primo.hosted.exlibrisgroup.com/openurl/44BIR/44BIR_VU1?u.ignore_date_coverage=true&rft.mms_id=9934051655504871
https://discover.durham.ac.uk/openurl/44DUR_INST/44DUR_INST:VU1?u.ignore_date_coverage=true&rft.mms_id=991010379482007366
https://discover.durham.ac.uk/openurl/44DUR_INST/44DUR_INST:VU1?u.ignore_date_coverage=true&rft.mms_id=991010423487907366
https://discover.gcu.ac.uk/discovery/openurl?institution=44GLCU_INST&vid=44GLCU_INST:44GLCU_VU2&?u.ignore_date_coverage=true&rft.mms_id=991002824176703836
https://ebookcentral.proquest.com/lib/aston/detail.action?docID=1562422
https://ebookcentral.proquest.com/lib/bham/detail.action?docID=7103878
https://ebookcentral.proquest.com/lib/edgehill/detail.action?docID=1562422
https://ebookcentral.proquest.com/lib/gmul-ebooks/detail.action?docID=7103878
https://ebookcentral.proquest.com/lib/lancaster/detail.action?docID=1562422
https://ebookcentral.proquest.com/lib/manchester/detail.action?docID=1562422
https://ebookcentral.proquest.com/lib/manchester/detail.action?docID=7103878
https://ebookcentral.proquest.com/lib/portsmouth-ebooks/detail.action?docID=7103878
https://ebookcentral.proquest.com/lib/qmu/detail.action?docID=7103878
https://ebookcentral.proquest.com/lib/ucreative-ebooks/detail.action?docID=7103878
https://ebookcentral.proquest.com/lib/warw/detail.action?docID=1562422
https://eu.alma.exlibrisgroup.com/openurl/44PLY_INST/VU_PLY?u.ignore_date_coverage=true&rft.mms_id=9915216639401281
https://eu04.alma.exlibrisgroup.com/view/uresolver/44LIV_INST/openurl?u.ignore_date_coverage=true&portfolio_pid=53398366730007351&Force_direct=true
https://hud.alma.exlibrisgroup.com/openurl/44HUD_INST/44HUD_INST:Services?u.ignore_date_coverage=true&rft.mms_id=991002437682504221
https://le.userservices.exlibrisgroup.com/view/uresolver/44UOLE_INST/openurl?u.ignore_date_coverage=true&rft.mms_id=991010103400702746
https://library-collections-search.westminster.ac.uk/openurl/44WST_INST/44WST_INST:WST_VUA?u.ignore_date_coverage=true&rft.mms_id=997393449303711
https://librarysearch.cardiff.ac.uk/openurl/44WHELF_CAR/44WHELF_CAR:44WHELF_CAR_VU1? ?u.ignore_date_coverage=true&rft.mms_id=9912122538502420
https://ljmu-primo.hosted.exlibrisgroup.com/openurl/44JMU/44JMU_services_page?u.ignore_date_coverage=true&rft.mms_id=9911681633203826
https://locate.coventry.ac.uk/openurl/COV/COV_VU1?u.ignore_date_coverage=true&rft.mms_id=996999900802011

https://manchester.primo.exlibrisgroup.com/discovery/openurl?institution=44MAN&vid=44MAN_INST:MU_NUI?u.ignore_date_coverage=true&rft.mms_id=992976582325401631
https://manchester.primo.exlibrisgroup.com/discovery/openurl?institution=44MAN&vid=44MAN_INST:MU_NUI?u.ignore_date_coverage=true&rft.mms_id=992984702754701631
https://metsearch.cardiffmet.ac.uk/openurl/44WHELF_CMU/44WHELF_CMU:44WHELF_CMU_NU1?u.ignore_date_coverage=true&rft.mms_id=99513383502425
https://metsearch.cardiffmet.ac.uk/openurl/44WHELF_CMU/44WHELF_CMU:44WHELF_CMU_NU1?u.ignore_date_coverage=true&rft.mms_id=99644613602425
https://onesearch.lancaster-university.uk/openurl/44LAN/LUL_VU1? ?u.ignore_date_coverage=true&rft.mms_id=9930643411401221
https://onesearch.lancaster-university.uk/openurl/44LAN/LUL_VU1? ?u.ignore_date_coverage=true&rft.mms_id=9930979150401221
https://primo.aber.ac.uk/openurl/44WHELF_ABW/44WHELF_ABW:44WHELF_ABW_VU1?u.ignore_date_coverage=true&rft.mms_id=9912429627702418
https://shu.primo.exlibrisgroup.com/discovery/openurl?institution=44SHU_INST&vid=44SHU_INST:44SHU_VU1&?u.ignore_date_coverage=true&rft.mms_id=99696202502501
https://www.open.ac.uk/library/openurl?u.ignore_date_coverage=true&rft.mms_id=9952951767702316

Appendix A. R Code

The following sections include all of the R codes that we used in the research. The first section of R codes has details and comments about the structures of the simulation process, including the creation of our data frame to use in the research. The second section of the R code provides detailed information of how the best model of each dimension is obtained.

A.1 P-Value Simulation Code

```
library(MASS)
library(MTS)

#########################This is what mq() doing in the
background################################
MVLB <- function(x,lag=24,adj=0){
 # Compute multivariate Ljung-Box test statistics
 #
 # adj: adjustment for the degrees of freedomm in the chi-square distribution.
 # adj is the number of coefficient parameters used in the fitted model, if any.
 #
 if(!is.matrix(x))x=as.matrix(x)
 nr=nrow(x)
 nc=ncol(x)
 g0=var(x)
 ginv=solve(g0)
 qm=0.0
 QM=NULL
 df = 0
 for (i in 1:lag){
  x1=x[(i+1):nr,]
  x2=x[1:(nr-i),]
  g = cov(x1,x2)
```

```
   g = g*(nr-i-1)/(nr-1)
   h=t(g)%*%ginv%*%g%*%ginv
   qm=qm+nr*nr*sum(diag(h))/(nr-i)
   df=df+nc^2
   dff= df-adj
   mindeg=nc^2-1
   pv = 1
   if(dff > mindeg)pv=1-pchisq(qm,dff)
   QM=rbind(QM,c(i,qm,dff,pv))
  }
  pvs=QM[,4]
  dimnames(QM) = list(names(pvs),c("  m  ","    Q(m) ","   df  "," p-value"))
  # cat("Ljung-Box Statistics: ","\n")
  return(QM)
  # # below is the graph
  # par(mfcol=c(1,1))
  # plot(pvs,ylim=c(0,1),xlab="m",ylab="prob",main="p-values of Ljung-Box statistics")
  # abline(h=c(0))
  # lines(rep(0.05,lag),lty=2,col='blue')
}
## I altered the function so it saves the results.########################
## I commented out the graphing ability########################

k_values <- 2 # this value can be changed to 3, 5, or 10 for other dimensions
n_values <- seq(40,300,10)
df <- data.frame(k = integer(), n = integer(), m = integer())
# for loop to create a dataframe for the study
for (k in k_values) {
  # Then iterate over n_values for each k
  for (n in n_values) {
    m_values <- 2:(n - 1)  # Generate m values for the current n

    # Create a temporary dataframe for the current k, n and all its m values
    temp_df <- data.frame(k = rep(k, length(m_values)), n = rep(n, length(m_values)), m =
m_values)

    # Append the temporary dataframe to the main dataframe
    df <- rbind(df, temp_df)
  }
}

checkpoint_path <- "~/Desktop/Grad School/Spring
2024/Thesis/simulation_checkpoint2.rds"
# Check if a checkpoint exists
```

```
if (file.exists(checkpoint_path)) {
  # Load the checkpoint
  load(checkpoint_path)
  cat("Resuming from checkpoint...\n")
} else {
  # Initialize variables if starting fresh
  all_pval <- NULL
  start_i_index <- 1
}

alpha <- 0.05
num_simulation <- 10^6

#### This is for all k's, the results gives 10^6 p-values
start_time <- Sys.time() # check how long the simulation lasts
pval_dim <- matrix(nrow = 4536, ncol = num_simulation)
  for (i in start_i_index:num_simulation){
    pval_list <- NULL
    for (n in n_values){
      r <- mvrnorm((n+1), rep(0,k), diag(k))
      pval_list <- rbind(pval_list,matrix(MVLB(r, lag = n-1)[2:(n-1),4]))
    }
    pval_dim[1:length(pval_list), i] <- pval_list
    # Save checkpoint periodically, e.g., every 10000 iterations of the inner loop
    if (i %% 10000 == 0) {
      saveRDS(pval_dim,file = checkpoint_path)
      cat("Checkpoint saved at iteration", i, "of simulation for k value", k, "\n")
    }
  }


end_time <- Sys.time()
duration <- end_time - start_time
print(duration)
```

A.2 Modeling R Code

```
# this file shows how the modeling process of dimension 10.
```

```
# For other dimensions, the process will be similar with different datasets and different
values of k
library(MASS)
#read the data file
df10 <- readRDS("~/Desktop/Grad School/Spring 2024/Thesis/df10.rds")
df10 <- as.data.frame(df10)
#########################This is what mq() doing in the
background###############################
############## MVLB Function ########################
MVLB <- function(x,lag=24,adj=0){
 # Compute multivariate Ljung-Box test statistics
 #
 # adj: adjustment for the degrees of freedomm in the chi-square distribution.
 # adj is the number of coefficient parameters used in the fitted model, if any.
 #
 if(!is.matrix(x))x=as.matrix(x)
 nr=nrow(x)
 nc=ncol(x)
 g0=var(x)
 ginv=solve(g0)
 qm=0.0
 QM=NULL
 df = 0
 for (i in 1:lag){
  x1=x[(i+1):nr,]
  x2=x[1:(nr-i),]
  g = cov(x1,x2)
  g = g*(nr-i-1)/(nr-1)
  h=t(g)%*%ginv%*%g%*%ginv
  qm=qm+nr*nr*sum(diag(h))/(nr-i)
  df=df+nc^2
  dff= df-adj
  mindeg=nc^2-1
  pv = 1
  if(dff > mindeg)pv=1-pchisq(qm,dff)
  QM=rbind(QM,c(i,qm,dff,pv))
 }
 pvs=QM[,4]
 dimnames(QM) = list(names(pvs),c("  m  ","    Q(m) ","   df  "," p-value"))
 # cat("Ljung-Box Statistics: ","\n")
 return(QM)
 # # below is the graph
 # par(mfcol=c(1,1))
 # plot(pvs,ylim=c(0,1),xlab="m",ylab="prob",main="p-values of Ljung-Box statistics")
```

```
 # abline(h=c(0))
 # lines(rep(0.05,lag),lty=2,col='blue')
}
##########################################################################
###################
# Subseting the entire dataframe for k=10
between0and50 = subset(df10,n<=50)
between51and70= subset(df10, (51 < n) & (n <=70))
between71and90= subset(df10, (71 <n) & (n <=90))
between91and110 = subset(df10, (n > 91) &(n <=110))
between111and200 = subset(df10, (n>111)&(n<=200))
between201andup = subset(df10, n >= 201)
##########################################################################
########################
# Let's try for k = 10 just to see

# plotting to the trends of the data
# plot(df10[39:86,4])
# plot(df10[1:38,4])
# plot(df10[87:144,4])

# Function to generate all teh s and p values
allPowers = function(start,end,by){
 nps = seq(start,end,by)
 mps = seq(start,end,by)
 eg = expand.grid(nps,mps)
 return(eg)
}
power_list = allPowers(0.1,2,0.1) # This generates all the s and p values for the model

# I will check for the cases for k=10
generate.data <- function(nPoints){
 simulated =mvrnorm(nPoints, rep(0,10), diag(10))
 return(simulated)
}
data0to50 = generate.data(45) # 45x10 matrix
##########################################################################
###################
# I'm separating the data by columns to make it a little easier to type
nlist1 <- between0and50[,2]
mlist1 <- between0and50[,3]
proplist1 <- between0and50[,4]
```

```
####### Linear Regression model used for the study, similar approach from Sidy's paper
linearModels = function(prop,nL,mL,dframe)
{

 lis.of.coefs = list()

 for(i in 1:nrow(dframe))
 {

  s = dframe[i,1]
  p = dframe[i,2]

  lis.of.coefs[[i]] = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +
I(nL^(2*s)) + I(nL^(3*s)*mL^(2*p))
                + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05,
length(prop)))$coefficients


 }

 lis.of.coefs = as.matrix(lapply(lis.of.coefs, function(y) as.vector(y)))
 mat = do.call(cbind,lis.of.coefs)
 return(mat)
}
howard1 <- linearModels(proplist1,nlist1,mlist1,power_list)

########################
# This is another model from Sidy's code, retrieve the residuals
all.my.res0_50= function(n,m,mat,dframe){
 container = list()
 for(i in 1:nrow(dframe)){
  s = dframe[i,1]
  p = dframe[i,2]
  vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
  container[[i]] = sum(mat[,i]*vec.of.var) + 0.05
 }
 return(container)
}
try1 <- all.my.res0_50(45,2,howard1,power_list)
#####################
# Now, let us take n and
all.myModels0_50 = function(timeseries,mat,dframe){
```

```
 n = nrow(timeseries)
 mlags = 2:(n-1)
 vals = lapply(mlags, function(m) all.my.res0_50(n,m,mat,dframe))
 val_differentLags = do.call(rbind,vals)
 return(val_differentLags)
}
check1 <- all.myModels0_50(data0to50,howard1,power_list)

setwd("~/Desktop/Grad School/Spring 2024/Thesis/MVLjungBox")
# sim45k10 = lapply(1:100000, function(i) generate.data(nPoints=46))
# save(sim45k10,file = "sim45k10.Rda")
load("sim45k10.Rda")
# d45 = sim45

# applying the original Multi. Ljung-Box test to validation data
MLB45 <- matrix(nrow = 43, ncol=100000)
for (i in 1:100000){
 MLB45[,i] <- as.matrix(MVLB(sim45k10[[i]], lag = 44)[2:44,4])
}

MLB45 <- as.data.frame(t(MLB45))
# function to count p value less than 0.05
less_than_0.05 <-function(x,nSimulations=100000){
 return(length(which(x < 0.05))/nSimulations)
}
# retrieve the proportion
howard45 <- lapply(1:nrow(power_list), function(i) mapply(function(x,y)
sum(x<y)/length(x), MLB45, check1[,i]))
myRes45 <- sapply(howard45, function(s) mean(s))
myRes45
new45 = apply(MLB45,2,less_than_0.05)
newMean45 = mean(new45)
newMean45
# locating the best pair of s and p
loch1 = which.min(abs(myRes45-0.05)) # 154
howardmodel1 <- power_list[154,] # s=1.4, p=0.8

#function of the best model
bestModel0to50 = function(n,m,s, p){
 vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
 coef = c(-1.722358e-03,1.364009e-01,-6.268533e-04,-2.528472e-07,7.190455e-
06,1.392121e-09,-1.879418e-11,-2.065772e-06,2.537805e-07)
```

```
  prod = sum(coef*vec.of.var) + 0.05
  return(prod)
}

#plot(between0and50[,4])
# function to test the best model with best pair of s and p
TestmyModels0to50 = function(n){
  mlags = 2:(n-1)
  vals = lapply(mlags, function(m) bestModel0to50(n,m,1.4,0.8))
  return(vals)
}
# plotting the data using the original Ljung-Box and my model
howardtest50 = TestmyModels0to50(50)
this50 = df10[39:86,4]
plot(this50,col="red",xlab = "Lags",ylab = "Type I Error")
points(2:49,howardtest50, type ="l", col="Blue")
legend("topright",legend = c("Multi Ljung-Box", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)

# true model of the subset
TrueModel1= function(prop,nL,mL,s=1.5,p=1.4)
{


  regr = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +  I(nL^(2*s)) +
I(nL^(3*s)*mL^(2*p))
         + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05, length(prop)))

  return(regr)
}
Cyril1 = TrueModel1(proplist1,nlist1,mlist1,1.4,0.8)

# similar process like above for other subsets of the data
##########################################################################
##############################################
generate.data <- function(nPoints){
  simulated =mvrnorm(nPoints, rep(0,10), diag(10))
  return(simulated)
}
data51to70 = generate.data(65) # 65x10 matrix
##################################################
#
nlist2 <- between51and70[,2]
mlist2 <- between51and70[,3]
```

```
proplist2 <- between51and70[,4]


####### This is the model that Sidy created for his project
linearModels = function(prop,nL,mL,dframe)
{

 lis.of.coefs = list()

 for(i in 1:nrow(dframe))
 {

  s = dframe[i,1]
  p = dframe[i,2]

  lis.of.coefs[[i]] = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +
I(nL^(2*s)) + I(nL^(3*s)*mL^(2*p))
                + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05,
length(prop)))$coefficients



 }

 lis.of.coefs = as.matrix(lapply(lis.of.coefs, function(y) as.vector(y)))
 mat = do.call(cbind,lis.of.coefs)
 return(mat)
}
howard2 <- linearModels(proplist2,nlist2,mlist2,power_list)

######################
# This is another model from Sidy's code
all.my.res51_70= function(n,m,mat,dframe){
 container = list()
 for(i in 1:nrow(dframe)){
  s = dframe[i,1]
  p = dframe[i,2]
  vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
  container[[i]] = sum(mat[,i]*vec.of.var) + 0.05
 }
 return(container)
}
try2 <- all.my.res51_70(65,2,howard2,power_list)
```

```
####################
# Now, let us take n and
all.myModels51_70 = function(timeseries,mat,dframe){
 n = nrow(timeseries)
 mlags = 2:(n-1)
 vals = lapply(mlags, function(m) all.my.res51_70(n,m,mat,dframe))
 val_differentLags = do.call(rbind,vals)
 return(val_differentLags)
}
check2 <- all.myModels51_70(data51to70,howard2,power_list)

# setwd("~/Desktop/Grad School/Spring 2024/Thesis/MVLjungBox")
# sim65k10 = lapply(1:100000, function(i) generate.data(nPoints=66))
# save(sim65k10,file = "sim65k10.Rda")
load("sim65k10.Rda")


MLB65 <- matrix(nrow = 63, ncol=100000)
for (i in 1:100000){
 MLB65[,i] <- as.matrix(MVLB(sim65k10[[i]], lag = 64)[2:64,4])
}
MLB65 <- as.data.frame(t(MLB65))

less_than_0.05 <-function(x,nSimulations=100000){
 return(length(which(x < 0.05))/nSimulations)
}

howard65 <- lapply(1:nrow(power_list), function(i) mapply(function(x,y)
sum(x<y)/length(x), MLB65, check2[,i]))
myRes65 <- sapply(howard65, function(s) mean(s))

new65 = apply(MLB65,2,less_than_0.05)
newMean65 = mean(new65)
newMean65 # 0.08193429

loch2 = which.min(abs(myRes65-0.05)) # 180
howardmodel2 <- power_list[180,] # s=2, p=0.9

bestModel51to70 = function(n,m,s, p){
 vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
 coef = c(-4.521172e-05,4.374216e-02,-8.761956e-06,-1.156707e-10,8.618181e-
09,2.863483e-14,-1.805493e-16,-9.482726e-08,6.187954e-09)
```

```
  prod = sum(coef*vec.of.var) + 0.05
  return(prod)
}

plot(between51and70[,4])
TestmyModels51to70 = function(n){
  mlags = 2:(n-1)
  vals = lapply(mlags, function(m) bestModel51to70(n,m,2.0,0.9))
  return(vals)
}
howardtest70 = TestmyModels51to70(70)
this70 = df10[145:212,4]
plot(this70,col="red",xlab = "Lags",ylab = "Type I Error")
points(2:69,howardtest70, type ="l", col="Blue")
legend("topright",legend = c("Multi Ljung-Box", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)

TrueModel2= function(prop,nL,mL,s=2,p=0.9)
{


  regr = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +  I(nL^(2*s)) +
I(nL^(3*s)*mL^(2*p))
         + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05, length(prop)))

  return(regr)
}
Cyril2 = TrueModel2(proplist2,nlist2,mlist2,2,0.9)
##########################################################################
##########
# Now let's try doing 70 < n <= 90 for k=10

# I will check for the cases for k=10
generate.data <- function(nPoints){
  simulated =mvrnorm(nPoints, rep(0,10), diag(10))
  return(simulated)
}
data71to90 = generate.data(85) # 85x10 matrix
##########################################################################
##################
# I'm separating the data by columns to make it a little easier to type
nlist3 <- between71and90[,2]
mlist3 <- between71and90[,3]
proplist3 <- between71and90[,4]
```

```
####### This is the model that Sidy created for his project
linearModels = function(prop,nL,mL,dframe)
{

 lis.of.coefs = list()

 for(i in 1:nrow(dframe))
 {

  s = dframe[i,1]
  p = dframe[i,2]

  lis.of.coefs[[i]] = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +
I(nL^(2*s)) + I(nL^(3*s)*mL^(2*p))
                + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05,
length(prop)))$coefficients


 }

 lis.of.coefs = as.matrix(lapply(lis.of.coefs, function(y) as.vector(y)))
 mat = do.call(cbind,lis.of.coefs)
 return(mat)
}
howard3 <- linearModels(proplist3,nlist3,mlist3,power_list)

########################
# This is another model from Sidy's code
all.my.res71_90= function(n,m,mat,dframe){
 container = list()
 for(i in 1:nrow(dframe)){
  s = dframe[i,1]
  p = dframe[i,2]
  vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
  container[[i]] = sum(mat[,i]*vec.of.var) + 0.05
 }
 return(container)
}
try3 <- all.my.res71_90(85,2,howard3,power_list)
#####################
```

```
# Now, let us take n and
all.myModels71_90 = function(timeseries,mat,dframe){
  n = nrow(timeseries)
  mlags = 2:(n-1)
  vals = lapply(mlags, function(m) all.my.res71_90(n,m,mat,dframe))
  val_differentLags = do.call(rbind,vals)
  return(val_differentLags)
}
check3 <- all.myModels71_90(data71to90,howard3,power_list)

# setwd("~/Desktop/Grad School/Spring 2024/Thesis/MVLjungBox")
# sim85k10 = lapply(1:100000, function(i) generate.data(nPoints=86))
# save(sim85k10,file = "sim85k10.Rda")
load("sim85k10.Rda")


MLB85 <- matrix(nrow = 83, ncol=100000)
for (i in 1:100000){
  MLB85[,i] <- as.matrix(MVLB(sim85k10[[i]], lag = 84)[2:84,4])
}

MLB85 <- as.data.frame(t(MLB85))
less_than_0.05 <-function(x,nSimulations=100000){
  return(length(which(x < 0.05))/nSimulations)
}

howard85 <- lapply(1:nrow(power_list), function(i) mapply(function(x,y)
sum(x<y)/length(x), MLB85, check3[,i]))
myRes85 <- sapply(howard85, function(s) mean(s))

new85 = apply(MLB85,2,less_than_0.05)
newMean85 = mean(new85)
newMean85 #0.08581867

loch3 = which.min(abs(myRes85-0.05)) # 200
howardmodel3 <- power_list[200,] # s=2, p=1.0

bestModel71to90 = function(n,m,s, p){
  vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
  coef = c(-2.025258e-05,2.128571e-02,-2.325250e-06,-1.193464e-11,2.243568e-
09,1.621225e-15,-4.745842e-18,6.133523e-09,1.117398e-10)
  prod = sum(coef*vec.of.var) + 0.05
```

```
 return(prod)
}

plot(between71and90[,4])
TestmyModels71to90 = function(n){
 mlags = 2:(n-1)
 vals = lapply(mlags, function(m) bestModel71to90(n,m,2.0,1.0))
 return(vals)
}
howardtest90 = TestmyModels71to90(90)
this90 = df10[291:378,4]
plot(this90,col="red",xlab = "Lags",ylab = "Type I Error")
points(2:89,howardtest90, type ="l", col="Blue")
legend("topright",legend = c("Multi Ljung-Box", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)

TrueModel3= function(prop,nL,mL,s=2,p=1)
{


 regr = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +  I(nL^(2*s)) +
I(nL^(3*s)*mL^(2*p))
        + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05, length(prop)))

 return(regr)
}
Cyril3 = TrueModel3(proplist3,nlist3,mlist3,2,1)
##########################################################################
##########
# Now let's try doing 90 < n <= 110 for k=2

# I will check for the cases for k=2
generate.data <- function(nPoints){
 simulated =mvrnorm(nPoints, rep(0,10), diag(10))
 return(simulated)
}
data91to110 = generate.data(100) # 100x10 matrix
##########################################################################
##################
# I'm separating the data by columns to make it a little easier to type
nlist4 <- between91and110[,2]
mlist4 <- between91and110[,3]
proplist4 <- between91and110[,4]
```

```
####### This is the model that Sidy created for his project
linearModels = function(prop,nL,mL,dframe)
{

 lis.of.coefs = list()

 for(i in 1:nrow(dframe))
 {

  s = dframe[i,1]
  p = dframe[i,2]

  lis.of.coefs[[i]] = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +
I(nL^(2*s)) + I(nL^(3*s)*mL^(2*p))
                + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05,
length(prop)))$coefficients


 }

 lis.of.coefs = as.matrix(lapply(lis.of.coefs, function(y) as.vector(y)))
 mat = do.call(cbind,lis.of.coefs)
 return(mat)
}
howard4 <- linearModels(proplist4,nlist4,mlist4,power_list)

########################
# This is another model from Sidy's code
all.my.res91_110= function(n,m,mat,dframe){
 container = list()
 for(i in 1:nrow(dframe)){
  s = dframe[i,1]
  p = dframe[i,2]
  vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
  container[[i]] = sum(mat[,i]*vec.of.var) + 0.05
 }
 return(container)
}
try4 <- all.my.res91_110(100,2,howard4,power_list)
####################
# Now, let us take n and
```

```
all.myModels91_110 = function(timeseries,mat,dframe){
 n = nrow(timeseries)
 mlags = 2:(n-1)
 vals = lapply(mlags, function(m) all.my.res91_110(n,m,mat,dframe))
 val_differentLags = do.call(rbind,vals)
 return(val_differentLags)
}
check4 <- all.myModels91_110(data91to110,howard4,power_list)

# setwd("~/Desktop/Grad School/Spring 2024/Thesis/MVLjungBox")
# sim100k10 = lapply(1:100000, function(i) generate.data(nPoints=101))
# save(sim100k10,file = "sim100k10.Rda")
load("sim100k10.Rda")


MLB100 <- matrix(nrow = 98, ncol=100000)
for (i in 1:100000){
 MLB100[,i] <- as.matrix(MVLB(sim100k10[[i]], lag = 99)[2:99,4])
}

MLB100 <- as.data.frame(t(MLB100))
less_than_0.05 <-function(x,nSimulations=100000){
 return(length(which(x < 0.05))/nSimulations)
}

howard100 <- lapply(1:nrow(power_list), function(i) mapply(function(x,y)
sum(x<y)/length(x), MLB100, check4[,i]))
myRes100 <- sapply(howard100, function(s) mean(s))

new100 = apply(MLB100,2,less_than_0.05)
newMean100 = mean(new100)
newMean100 #0.08809949

loch4 = which.min(abs(myRes100-0.05)) # 400
howardmodel4 <- power_list[400,] # s=2, p=2

bestModel91to110 = function(n,m,s, p){
 vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
 coef = c(6.560008e-06,-2.814827e-05,8.957106e-09,-2.888567e-16,-5.523747e-
10,1.195876e-20,3.044985e-25,2.124154e-16,-1.308718e-20)
 prod = sum(coef*vec.of.var) + 0.05
 return(prod)
```

```
}

plot(between91and110[,4])
TestmyModels91to110 = function(n){
 mlags = 2:(n-1)
 vals = lapply(mlags, function(m) bestModel91to110(n,m,2,2))
 return(vals)
}
howardtest110 = TestmyModels91to110(110)
this110 = df10[477:584,4]
plot(this110,col="red",xlab = "Lags",ylab = "Type I Error")
points(2:109,howardtest110, type ="l", col="Blue")
legend("topright",legend = c("Multi Ljung-Box", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)

TrueModel4= function(prop,nL,mL,s=2,p=2)
{


 regr = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +  I(nL^(2*s)) +
I(nL^(3*s)*mL^(2*p))
        + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05, length(prop)))

 return(regr)
}
Cyril4 = TrueModel4(proplist4,nlist4,mlist4,2,2)
##########################################################################
###################
# Now let's try doing 110 < n <= 200 for k=2

# I will check for the cases for k=2
generate.data <- function(nPoints){
 simulated =mvrnorm(nPoints, rep(0,10), diag(10))
 return(simulated)
}
data111to200 = generate.data(185) # 185x10 matrix
##########################################################################
###################
# I'm separating the data by columns to make it a little easier to type
nlist5 <- between111and200[,2]
mlist5 <- between111and200[,3]
proplist5 <- between111and200[,4]
```

```
####### This is the model that Sidy created for his project
linearModels = function(prop,nL,mL,dframe)
{

 lis.of.coefs = list()

 for(i in 1:nrow(dframe))
 {

  s = dframe[i,1]
  p = dframe[i,2]

  lis.of.coefs[[i]] = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +
I(nL^(2*s)) + I(nL^(3*s)*mL^(2*p))
                + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05,
length(prop)))$coefficients


 }

 lis.of.coefs = as.matrix(lapply(lis.of.coefs, function(y) as.vector(y)))
 mat = do.call(cbind,lis.of.coefs)
 return(mat)
}
howard5 <- linearModels(proplist5,nlist5,mlist5,power_list)

########################
# This is another model from Sidy's code
all.my.res111_200= function(n,m,mat,dframe){
 container = list()
 for(i in 1:nrow(dframe)){
  s = dframe[i,1]
  p = dframe[i,2]
  vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
  container[[i]] = sum(mat[,i]*vec.of.var) + 0.05
 }
 return(container)
}
try5 <- all.my.res111_200(185,2,howard5,power_list)
#####################
# Now, let us take n and
all.myModels111_200 = function(timeseries,mat,dframe){
```

```
 n = nrow(timeseries)
 mlags = 2:(n-1)
 vals = lapply(mlags, function(m) all.my.res111_200(n,m,mat,dframe))
 val_differentLags = do.call(rbind,vals)
 return(val_differentLags)
}
check5 <- all.myModels111_200(data111to200,howard5,power_list)

# setwd("~/Desktop/Grad School/Spring 2024/Thesis/MVLjungBox")
# sim185k10 = lapply(1:100000, function(i) generate.data(nPoints=186))
# save(sim185k10,file = "sim185k10.Rda")
load("sim185k10.Rda")


MLB185 <- matrix(nrow = 183, ncol=100000)
for (i in 1:100000){
 MLB185[,i] <- as.matrix(MVLB(sim185k10[[i]], lag = 184)[2:184,4])
}

MLB185 <- as.data.frame(t(MLB185))
less_than_0.05 <-function(x,nSimulations=100000){
 return(length(which(x < 0.05))/nSimulations)
}

howard185 <- lapply(1:nrow(power_list), function(i) mapply(function(x,y)
sum(x<y)/length(x), MLB185, check5[,i]))
myRes185 <- sapply(howard185, function(s) mean(s))

new185 = apply(MLB185,2,less_than_0.05)
newMean185 = mean(new185)
newMean185 #0.09389754

loch5 = which.min(abs(myRes185-0.05)) #340
howardmodel5 <- power_list[340,] #s=2, p=1.7

bestModel111to200 = function(n,m,s, p){
 vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
 coef = c(3.954236e-06,-5.704072e-05,4.373655e-09,-5.817395e-17,-1.124866e-
10,8.912555e-22,2.764302e-27,8.600325e-16,-6.547621e-20)
 prod = sum(coef*vec.of.var) + 0.05
 return(prod)
}
```

```
plot(between111and200[,4])
TestmyModels111to200 = function(n){
 mlags = 2:(n-1)
 vals = lapply(mlags, function(m) bestModel111to200(n,m,2,1.7))
 return(vals)
}
howardtest200 = TestmyModels111to200(200)
this200 = df10[1809:2006,4]
points(this200,col="red")
plot(2:199,howardtest200, type ="l", col="Blue",xlab = "Lags",ylab = "Type I Error")
legend("bottomright",legend = c("Multi Ljung-Box", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)

TrueModel5= function(prop,nL,mL,s=2,p=1.7)
{


 regr = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +  I(nL^(2*s)) +
I(nL^(3*s)*mL^(2*p))
        + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05, length(prop)))

 return(regr)
}
Cyril5 = TrueModel5(proplist5,nlist5,mlist5,2,1.7)
##########################################################################
###################
# Now let's try doing n>200 for k=10

# I will check for the cases for k=10
generate.data <- function(nPoints){
 simulated =mvrnorm(nPoints, rep(0,10), diag(10))
 return(simulated)
}
data201up = generate.data(280) # 250x10 matrix
##########################################################################
###################
# I'm separating the data by columns to make it a little easier to type
nlist6 <- between201andup[,2]
mlist6 <- between201andup[,3]
proplist6 <- between201andup[,4]


####### This is the model that Sidy created for his project
```

```
linearModels = function(prop,nL,mL,dframe)
{

 lis.of.coefs = list()

 for(i in 1:nrow(dframe))
 {

  s = dframe[i,1]
  p = dframe[i,2]

  lis.of.coefs[[i]] = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +
I(nL^(2*s)) + I(nL^(3*s)*mL^(2*p))
                + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05,
length(prop)))$coefficients


 }

 lis.of.coefs = as.matrix(lapply(lis.of.coefs, function(y) as.vector(y)))
 mat = do.call(cbind,lis.of.coefs)
 return(mat)
}
howard6 <- linearModels(proplist6,nlist6,mlist6,power_list)

########################
# This is another model from Sidy's code
all.my.res201up= function(n,m,mat,dframe){
 container = list()
 for(i in 1:nrow(dframe)){
  s = dframe[i,1]
  p = dframe[i,2]
  vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
  container[[i]] = sum(mat[,i]*vec.of.var) + 0.05
 }
 return(container)
}
try6 <- all.my.res201up(280,2,howard6,power_list)
#####################
# Now, let us take n and
all.myModels201up = function(timeseries,mat,dframe){
 n = nrow(timeseries)
```

```
 mlags = 2:(n-1)
 vals = lapply(mlags, function(m) all.my.res201up(n,m,mat,dframe))
 val_differentLags = do.call(rbind,vals)
 return(val_differentLags)
}
check6 <- all.myModels201up(data201up,howard6,power_list)

# setwd("~/Desktop/Grad School/Spring 2024/Thesis/MVLjungBox")
# sim280k10 = lapply(1:100000, function(i) generate.data(nPoints=281))
# save(sim280k10,file = "sim280k10.Rda")
load("sim280k10.Rda")


MLB280 <- matrix(nrow = 278, ncol=100000)
for (i in 1:100000){
 MLB280[,i] <- as.matrix(MVLB(sim280k10[[i]], lag = 279)[2:279,4])
}

MLB280 <- as.data.frame(t(MLB280))
less_than_0.05 <-function(x,nSimulations=100000){
 return(length(which(x < 0.05))/nSimulations)
}

howard280 <- lapply(1:nrow(power_list), function(i) mapply(function(x,y)
sum(x<y)/length(x), MLB280, check6[,i]))
myRes280 <- sapply(howard280, function(s) mean(s))

new280 = apply(MLB280,2,less_than_0.05)
newMean280 = mean(new280)
newMean280 #0.09522183

loch6 = which.min(abs(myRes280-0.05)) #340
howardmodel6 <- power_list[340,] #s=2, p=1.7

bestModel201up = function(n,m,s, p){
 vec.of.var =
c(n^s,m^p,n^(s)*m^(p),n^(2*s)*m^(2*p),n^(2*s),n^(3*s)*m^(2*p),n^(3*s)*m^(3*p),m^(4*
p),m^(5*p))
 coef = c(1.589487e-06,-3.609943e-05,1.143797e-09,-2.931991e-18,-2.076415e-
11,1.729112e-23,1.783932e-28,6.156433e-17,-2.540740e-21)
 prod = sum(coef*vec.of.var) + 0.05
 return(prod)
}
```

```
plot(between201andup[,4])
TestmyModels201up = function(n){
 mlags = 2:(n-1)
 vals = lapply(mlags, function(m) bestModel201up(n,m,2,1.7))
 return(vals)
}
howardtest300 = TestmyModels201up(300)
this300 = df10[4239:4536,4]
points(this300,col="red")
plot(2:299,howardtest300, type ="l", col="Blue",xlab = "Lags",ylab = "Type I Error")
legend("bottomright",legend = c("Multi Ljung-Box", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)

TrueModel6= function(prop,nL,mL,s=2,p=1.7)
{


 regr = lm(prop ~ 0 + I(nL^s)+ I(mL^p) + I(nL^s*mL^p) + I(nL^(2*s)*mL^(2*p)) +  I(nL^(2*s)) +
I(nL^(3*s)*mL^(2*p))
        + I(nL^(3*s)*mL^(3*p)) + I(mL^(4*p)) + I(mL^(5*p)), offset=rep(0.05, length(prop)))

 return(regr)
}
Cyril6 = TrueModel6(proplist6,nlist6,mlist6,2,1.7)
##########################################################################
########################################
####### ALL GRAPHS IN ONE ############
par(mfrow=c(3,2))

plot(this50,col="red",xlab = "Lags",ylab = "Type I Error")
points(2:49,howardtest50, type ="l", col="Blue")
#legend("topleft",legend = c("Adjusted Box Pierce", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)

plot(this70,col="red",xlab = "Lags",ylab = "Type I Error")
points(2:69,howardtest70, type ="l", col="Blue")
#legend("topleft",legend = c("Adjusted Box Pierce", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)

plot(this90,col="red",xlab = "Lags",ylab = "Type I Error")
points(2:89,howardtest90, type ="l", col="Blue")
#legend("topleft",legend = c("Adjusted Box Pierce", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)
```

```
plot(this110,col="red",xlab = "Lags",ylab = "Type I Error")
points(2:109,howardtest110, type ="l", col="Blue")
#legend("topleft",legend = c("Adjusted Box Pierce", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)


plot(2:199,howardtest200, type ="l", col="Blue",xlab = "Lags",ylab = "Type I Error")
points(this200,col="red")
#legend("topleft",legend = c("Adjusted Box Pierce", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)


plot(2:299,howardtest300, type ="l", col="Blue",xlab = "Lags",ylab = "Type I Error")
points(this300,col="red")
#legend("topleft",legend = c("Adjusted Box Pierce", "Corrected Version"),col=c("red",
"blue"),lty=1:2, cex=0.8)
mtext("Rejection Region Correction", outer = TRUE, cex = 1.5)
```