Construction of a Novel Omnibus Normality Test and Power Comparison with the Shapiro-Wilk Test

Aaron Scherf, Dr. Mark Inlow

May 13, 2024

1 Introduction

Normality testing is a vital part of empirical statistics, as it determines the applicability of further analyses; to date, however, the most commonly employed normality test–the Shapiro-Wilk–has a relatively low degree of power in the face of certain departures from normality. This limitation motivated this research to develop new omnibus normality tests with power comparable to the Shapiro-Wilk Test. The intended theoretical contribution is to construct a flexible method of normality testing, by examining the behavior of an empirical process using a technique analogous to smoothing splines. This paper aims to create a foundation for empirically feasible normality testing with more opportunities for future development than traditional tests. We present the theoretical foundations for the new test, a discussion of the methodology, and preliminary empirical results from simulated power tests.

The approach is based on an empirical process which standardizes and orders a given sample $\vec{y}$ from a continuous distribution to convert the sample into order statistics $s(i)$. The process then applies the probability integral transform to convert the order statistics to random variables with a standard uniform distribution $s(i)$. Finally, the expected value $E[s(i)]$ is subtracted from the sample to yield a zero-mean vector $s^*(i)$. If the transformed sample displays large departures from the zero vector, our test rejects the null hypothesis that the sample was drawn from a normal distribution. A full description of the empirical process and hypothesis test is provided below.

The result of the empirical process is summarized in Figures 1 through 4. For samples drawn from a normal distribution, $s^*(i)$ displays random fluctuations around the zero vector, represented by the horizontal black line. For non-normal distributions, there are clear and systematic departures from zero. In some distributions, such as the chi-square or gamma distribution, these departures clearly lead to a different expected value, while for others such as the t-distribution, the departures are symmetric. The observable pattern in these distributions is what motivated the use of ordered samples, to incorporate not only the magnitude but the direction of departures from zero after application of the empirical process.

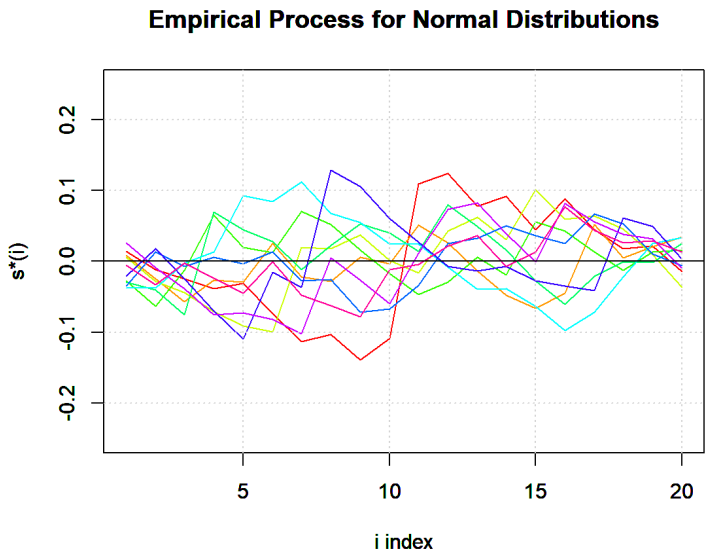
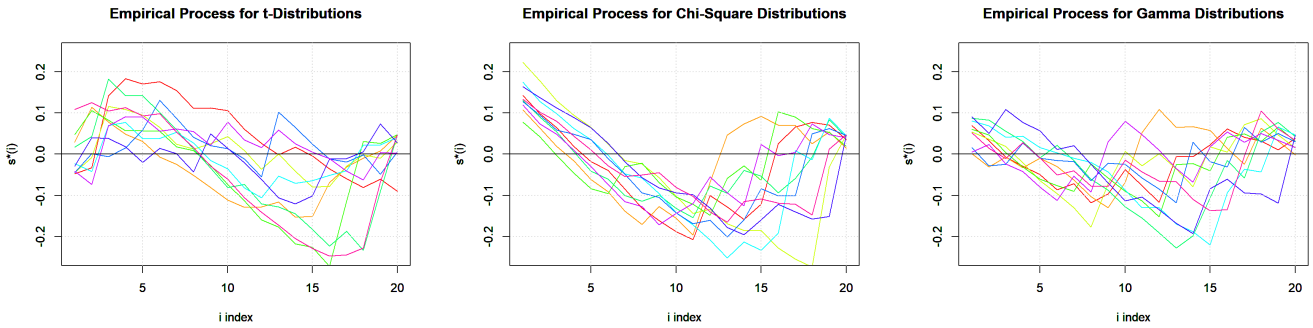


Fig. 1: Samples from the empirical process drawn from a normal distribution



(a) t-Distribution (b) Chi-Square Distribution (c) Gamma Distribution

Fig. 2: Samples after Empirical Process Transform

The goal of the hypothesis test proposed below is to smooth the randomness displayed by these samples, in order to detect this systematic departure from zero, particularly for samples which are nearly normal and thus difficult to detect via the Shapiro-Wilk test.

The kernel density distributions of various distributions after passing through the empirical process are superimposed in Figure 5 below. This plot rotates Figures 1-4 so that the expected value of s*(i) is on the x-axis and the density of observed values on the y-axis. Again, we see the normal distribution clearly has a tight symmetric distribution centered at zero, while non-normal distributions display various degrees of departure. A test which can capture the magnitude of these departures would provide an omnibus normality test.

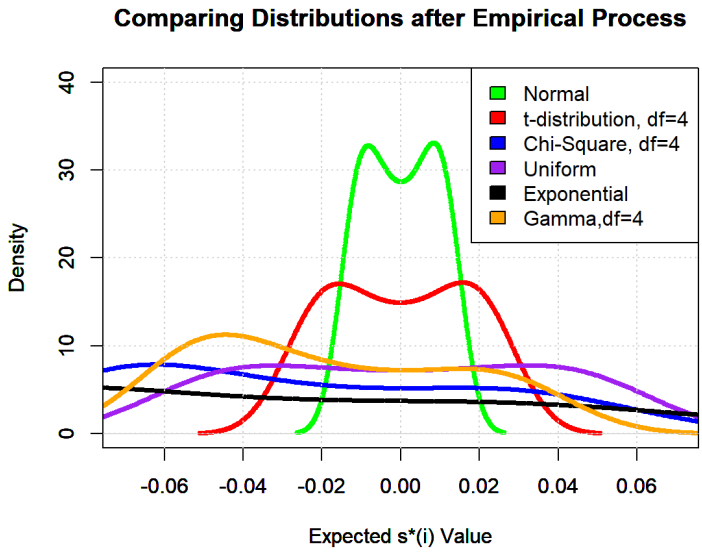


Fig. 3: Comparison of density after application of the empirical process

2 Theoretical Foundations

The following section describes some of the basic ideas necessary for the methodology of the normality test. Readers who are already familiar with the concepts may skip to Section 3.

2.1 The Shapiro-Wilk Test

The benchmark for normality testing in most applications is the Shapiro-Wilk test [Shapiro, S. S., and Wilk, M. B., 1965], due to its wide applicability and relatively high likelihood of detecting that a distribution is non-normal, also known as the power of the test. The null hypothesis of the test is that the sample being tested was sampled from a normally distributed population. This is achieved by calculating the test statistic:

$$W = \frac{(\sum_{i=1}^{n} a_i x_{(i)})^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{1}$$

where $x(i)$ is the ith order statistic of the sample and x is the sample mean. The coefficients a_i are calculated as $a_i = \frac{m^T V^{-1}}{C}$, where$m = (m_1, m_2, ..., m_n)^T$ is the vector of expected values of the order statistics of an independently and identically distributed random variable sampled from a standard normal distribution, V is the covariance matrix of these normal order statistics, and

$$C = ||V^{-1}m|| = (m^T V^{-1} m)^{1/2} \tag{2}$$

Note that since the expected values of the order statistic and its covariance do not change, the coefficients a_i are usually given in a table rather than recalculated each time. The test statistic W is used to accept or reject the null hypothesis for a given sample based on the

significance level chosen via Monte Carlo simulations.

The Shapiro-Wilk test has been broadly recognized as the normality test with the highest power for many practical applications. However, the test relies on certain assumptions about the skew and kurtosis properties of the target sample being tested. Therefore, the SW-test is only approximately omnibus, so alternative tests are required for samples which are near to normal. Thus there is an obvious motive to devise a truly omnibus test, which can be applied without the need to confirm the properties of the underlying distribution, with a similar power specification for various non-normal distributions.

2.2 Generalized Least Squares Regression

Given that our empirical process described above orders the sample, its errors are necessarily correlated. As a result, hypothesis testing by ordinary least squares (OLS) is no longer appropriate, since the conditions of the model require IID errors. Generalized least squares regression (GLS) is an extension of the OLS linear regression model, to account for correlation between random errors in the estimated values. GLS was first introduced by Alexander Aitkin [Aitken, A. C. (1935)] and requires specification of the covariance matrix for the $\vec{\varepsilon}$ values, which is only possible if their distribution is known; without a known covariance matrix, the method of feasible generalized least squares is a viable alternative, but with less certainty of statistical efficiency.

The other assumptions of GLS are that the conditional mean of the target sample $\vec{y}$ given the design matrix X is a linear function of X and that the aforementioned covariance matrix of the random errors is nonsingular. Thus we have that $\vec{y} = X\vec{\beta} + \vec{\varepsilon}$ (the OLS regression model), with $E[\vec{\varepsilon} \mid X] = 0$, and $Cov[\vec{\varepsilon} \mid X] = \sigma$. The objective is to optimally estimate the coefficient vector $\vec{\beta}$ for the model by minimizing the residual error $\vec{\varepsilon}$.

Starting with the solution to the objective function for OLS, which is:

$$\hat{\beta}_{OLS} = argmin_{\vec{\beta}}(\vec{y} - X\vec{\beta})^T(\vec{y} - X\vec{\beta}) \tag{3}$$

To specify a GLS model, we incorporate the inverse of the covariance matrix explicitly, such that the solution becomes:

$$\hat{\beta}_{GLS} = argmin_{\vec{\beta}}(\vec{y} - X\vec{\beta})^T\sigma^{-1}(\vec{y} - X\vec{\beta}) \tag{4}$$

It is well known that the solution of equation (4) yields the formula:

$$\hat{\beta}_{GLS} = (X^T\sigma^{-1}X)^{-1}X^T\sigma^{-1}\vec{y} \tag{5}$$

We see that the main difference between OLS and GLS is the inclusion of the inverse covariance matrix σ^{-1} of the random errors . The use of σ^{-1} explicitly incorporates the covariance of the random errors into the regression estimates, thus accounting for correlation between random errors.

The GLS estimate $\hat{\beta_{GLS}}$ has the useful properties of being unbiased, consistent, efficient, and asymptotically normal. More importantly, it is able to account for autocorrelation between random errors and thus use samples which are not IID, by explicitly incorporating a known covariance matrix. The applicability to non-IID samples motivated the use of GLS for this project, since the empirical process above clearly has correlated errors.

2.3 Penalized Regression

Regression penalization, while originally used to ameliorate multicollinearity, has more recently been used to manage the bias-variance tradeoff; in most use cases, introducing a bias term to control the variance of a model, which is essentially smoothing the predictor function. Penalization also has applications in controlling the number of independent variables in a model to reduce overfitting and improve interpretability. The two most popular forms of penalization in regression models are lasso and ridge regression, where the former uses the L1 absolute value distance metric and the latter the L2 Euclidean distance metric [Green, P. J.; Silverman, B.W., 1994].

The initial motivation behind penalization were cases of multicollinearity, where the matrix $X^T X$ is near singular, such that the determinant $det(X^T X)$ is near zero and the matrix $X^T X$ is non-invertible. Ridge regression solves this by adding positive values to the diagonal, controlled by a parameter λ, which becomes the penalization factor [Hoerl, Arthur E., and Kennard, Robert W., 1970].

Starting again with the solution to the objective function for OLS, given by Equation (3) above, to obtain ridge regression we simply add a regularization term $\lambda(\beta^T \beta)$:

$$\hat{\beta}_R = argmin_{\vec{\beta}}(\vec{y} - X\vec{\beta})^T(\vec{y} - X\vec{\beta} + \lambda(\beta^T \beta) \tag{6}$$

Note that ridge regression simplifies to OLS when $\lambda = 0$, since ridge regression is simply OLS augmented with an L2 penalty on the beta-vector.

The solution of the ridge regression objective function yields a closed form solution for $\hat{\beta_R}$, given by:

$$\hat{\beta}_R = (X^T X + \lambda I)^{-1} X^T \vec{y} \tag{7}$$

This formula demonstrates that we are adding the λI term to the matrix $X^T X$, hence increasing the values on the diagonal of $X^T X$ when is positive, and thus ameliorating the scenario where $X^T X$ is near singular and thus non-invertible. The λ parameter is multiplied by the identity matrix I in order to apply the penalization factor to the entire diagonal of $X^T X$.

The extension of this method to smoothing variance and reducing overfitting makes use of the same λ parameter and objective function structure, but rather than the identity matrix I used in equation (7), various weighting matrices are used to incorporate different types of penalties based on the desired properties of the model. While this method increases the

bias of the model, by forcing the regression to weigh some columns of X more than others, it results in more stable and more interpretable models.

2.4 Smoothing Splines

Smoothing splines are a class of estimated functions $\hat{f}(x_i)$ for a given target function $f(x_i) = y_i$, where y_i are observed values which include some statistical noise and x_i are observed input data. The estimated functions smooth the noisy x_i, y_i data into lower variance estimations [Craven, P., and Wahba, G. (1979)].

The most typical example is that of the cubic smoothing spline. Given a set of observations x_i, y_i for $i = 1, ..., n$, we define the target function $f(x_i) = y_i - \varepsilon_i$, where the ε_i random errors are independent random variables with expectation of zero. The smoothing spline estimate $f(x_i)$ minimizes the value of:

$$\sigma_{i=1}^{n}(y_i - \hat{f}(x_i))^2 + \lambda \int \hat{f}''(x_i)^2 dx \tag{8}$$

where λ is a smoothing parameter and $\int \hat{f}''(x_i)^2 dx$ a roughness penalty term. The second derivative of the estimated function provides a measure of the rate of change in the slope of the estimation function between adjacent points in x_i. Squaring this term ensures that we capture the overall sum of this rate of change as the magnitude of the vector $\hat{f}''(x_i)$ in a strictly positive value. Finally, taking its integral allows us to measure this magnitude over the entire distribution of the continuous function. Thereby, $\int \hat{f}''(x_i)^2 dx$ provides a measure of the variability in the estimation function over its entire domain.

Clearly, if $\lambda = 0$, the minimization is simply over the sum of squared residuals. As λ increases towards infinity, the roughness penalty term $\int \hat{f}''(x_i)^2 dx$ becomes more relevant to the minimization than the residual term, such that the estimate converges to OLS regression.

Smoothing splines as a general category of function approximation have many applications and have inspired similar approaches to smoothing predictions. The method employed in this paper does not technically use smoothing splines, strictly defined, since in a smoothing spline the predictor functions are defined across their entire domain. The methodology defined below uses basis vectors, which are only defined at specifically observed values, rather than complete functions as in true smoothing splines.

This difference means that the $\lambda \int \hat{f}''(x_i)^2 dx$ penalty term cannot be utilized, as smoothing splines depend on a complete integral of the function across its entire domain, while a basis vector is only observed as a point estimate. Thus, we must convert our roughness term to a discrete case.

The basis vector approach was selected over a true smoothing spline since the normality test methodology below is concerned with point-to-point variation in the empirical process. Complete knowledge of the residual function, and thus its underlying distribution, would

depend on asymptotically infinite sampling iterations, making it computationally intractable and impractical for an applied normality test.

2.5 Basis Vectors and Eigenvectors

A set of vectors B in a finite-dimensional vector space V forms a basis if the entire vector space can be formed from a finite linear combination of elements in B, which are then referred to as basis vectors. Basis vectors are linearly independent, such that no vector $b \in B$ can be formed as a linear combination of the remaining vectors. The basis B thus forms a spanning set for the vector space V.

In our methodology below, we use a design matrix X composed of the eigenvectors of a covariance matrix σ of a random vector $\vec{y}$. Critically, the eigenvectors of the $n \times n$ matrix σ form a basis, with n distinct eigenvalues. The eigenvalue corresponding to an eigenvector measures the variance of $\vec{y}$ along the direction of that eigenvector. The largest eigenvalue λ_1 thus corresponds to the eigenvector $\vec{e_1}$ describing the direction of maximum variance of the random vector $\vec{y}$, with each subsequent eigenvalue $\lambda_2, \lambda_3, ..., \lambda_n$ providing an ordering of descending variance for the corresponding eigenvectors.. Since the set of eigenvectors form a basis, they provide a linearly independent set which spans the vector space, which in this case is $\mathbb{R}^n$, where n is the number of elements of $\vec{y}$.

2.6 Generalized Lambda and Generalized Pareto Distributions

The generalized lambda distribution (GLD) was first described by John Tukey [1960] to create a flexible distribution which can approximate many other probability distributions by manipulation of a set of parameters λ. Note: this λ is not related to the penalization parameter λ above, but is maintained as general notation. The exact combination of λ parameters required to approximate a distribution of interest, however, is known to be computationally expensive; to create a range of distributions with specific moments, such as a set mean, variance, skew, and kurtosis, it is necessary to reference expansive tables of pre-calculated GLD parameter values.

The GLD quartile function is given by:

$$Q_x(p) = \lambda_1 + \frac{p^{\lambda_3} - (1-p)^{\lambda_4}}{\lambda_2} \tag{9}$$

Van Staden and Loots [2009] extend this popular distribution through a reparametrization which they call the generalized pareto distribution (GPD). The key advantage of GPD is simplifying the process of approximating a given distribution using closed-form L-moment estimators, which are provided in their paper "Method of L-moment estimation for the generalized lambda distribution". The quartile function for GPD is defined as:

$$Q_x(p) = \alpha + \beta[(1-\delta)(\frac{p^\lambda - 1}{\lambda}) - \delta(\frac{(1-p)^\lambda - 1}{\lambda})] \tag{10}$$

The L-moments below allow for simple comparison of the properties of the GPD. L-moments summarize the shape of a given probability distribution, providing an analog to the conventional moments: the mean, standard deviation, skewness and kurtosis. The L-1 moment, often referred to as the L-mean or L-location of the distribution, is given in van Staden and Loot's paper for the GPD as:

$$L_1 = \alpha - \frac{\beta(1-2\delta)}{\lambda+1} \tag{11}$$

The L-2 moment, the L-scale, provides an analog to the standard deviation of the distribution, a measure of spread, and is given for the GPD by:

$$L_2 = \frac{\beta}{(\lambda+1)(\lambda+1)} \tag{12}$$

The L-3 and L-4 moments provide ratios to measure the L-skewness and L-kurtosis, respectively, and are given for the GPD by:

$$L_3 = \frac{(\lambda-1)(1-2\delta)}{\lambda+3} \tag{13}$$

$$L_4 = \frac{(\lambda-1)(\lambda-2)}{(\lambda+3)(\lambda+4)} \tag{14}$$

In order to approximate a desired distribution using GPD, it is a simple process to identify the desired L-moments of the distribution and work backwards to calculate the necessary parameter values in $\alpha, \beta, \lambda, \delta$.

The standard normal distribution, for example, has an L-1 moment of 0, L-2 moment of 1, L-3 moment of 0, and L-4 moment of approximately 0.1226. Thus, we can clearly define parameter values to satisfy these L-moments by taking any combination in which:

$$\alpha = \frac{\beta(1-2\delta)}{\lambda+1}$$

$$\beta = (\lambda+1)(\lambda+2)$$

$$0 = (\lambda-1)(1-2\delta)$$

$$0.1226 \approx \frac{(\lambda-1)(\lambda-2)}{(\lambda+3)(\lambda+4)}$$

Choosing $\delta = 0.5$ and letting $\alpha = \frac{\beta(1-2\delta)}{\lambda+1}, \beta = (\lambda+1)(\lambda+2)$, we can adjust the equations so they can all be stated solely in terms of λ, since we now have:

$$\alpha = \frac{(\lambda+1)(\lambda+2)(0)}{\lambda+1} = 0$$

$$0 = (\lambda-1)(1-2\delta)) = (\lambda-1)(0) = 0$$

$$0.1226 \approx \frac{(\lambda-1)(\lambda-2)}{(\lambda+3)(\lambda+4)}$$

It can be verified empirically that the appropriate λ parameter value is approximately 4.256, as follows:

$$0.1226 \approx \tfrac{(\lambda-1)(\lambda-2)}{(\lambda+3)(\lambda+4)} \text{ is satisfied by } \tfrac{(3.256)(2.256)}{(7.256)(8.256)} = \tfrac{7.345536}{59.905536} = 0.122618$$

The R code provided in the appendix provides an implementation of the GPD quartile function and application for a range of parameter values, to study the effect of gradual increases in either the L-skewness or L-kurtosis ratios compared to the normal distribution.

3 Methods

This section includes a discussion of the methodology behind the sampling of the empirical process, smoothing predicted values of the sample via a penalized GLS regression, and the construction of the test statistic for the final testing of the null hypothesis that the sample is normal.

3.1 Sampling from Empirical Process and Estimation of Covariance Matrix

Let $(y_1, y_2, ..., y_n)$ be an IID sample drawn from the distribution of interest. Using the probability integral transform we construct an empirical process as follows.

We first compute the order statistics $y(1), y(2), ..., y(n)$ where $y(1)$ is the smallest of the sample values y_i, $y(2)$ is the next smallest, and so on. We then standardize the order statistics by subtracting the sample mean $\bar{y}$ from each order statistic and dividing the result by the sample standard deviation S_y, i.e.

$$y^*(i) = \frac{y(i) - \bar{y}}{S_y} \tag{15}$$

We then apply the probability integral transform to the standardized order statistics, to create $s(1), s(2), ..., s(n)$ where $s(i) = \phi(y^*(i))$, with ϕ being the standard normal cumulative distribution function. Finally, we convert the $s(i)$ to a zero-mean empirical process by subtracting the expected value $E[s(i)]$ from $s(i)$ where $E[s(i)] \approx \frac{i}{(n+1)}$, i.e., we compute

$$s^*(i) = s(i) - \frac{i}{n+1} \tag{16}$$

The zero-mean process $s^*(i)$ for a sample drawn from a normally distributed population should have values which are approximately zero. Therefore, measuring systematic deviations from zero across the values of $s^*(i)$ should provide a useful test of normality; however, the large variance of $s^*(i)$ makes detection of systematic deviations problematic and consistent testing difficult. Thus, we measure the smoothed magnitude of these deviations through an approach analogous to smoothing splines, employing a penalized generalized least squares (GLS) regression estimation, with $s^*(i)$ as the response variable vector $\vec{y}$.

$$\vec{y} = (s_1^*, s_2^*, ..., s_n^*)^T \tag{17}$$

Sampling $s^*(i)$ from a normal distribution allows us to observe the covariance matrix of the standardized probability integral transform σ, an $n \times n$ matrix.

3.2 Construction of the Penalized Generalized Least Squares Objective Function

The motivation for the GLS approach is simple: the elements of the empirical process $s^*(i)$ are not independently and identically distributed (IID), being ordered and with unequal variance, and thus violate the assumptions of ordinary least squares regression. Thus, in order to estimate a reliable test statistic from this empirical process, we need to account explicitly for the covariance between the sample observations. Therefore, we analyze $s^*(i)$ using GLS rather than OLS.

As noted above, for GLS, we have the solution to the objective function given by:

$$\hat{\beta}_{GLS} = argmin_\beta (\vec{y} - X\vec{\beta})^T \sigma^{-1} (\vec{y} - X\vec{\beta}) \tag{18}$$

with σ as the covariance matrix of the response variable vector $\vec{y} = s^*(i)$ for an index $i = 1, 2, ..., n$.

In order to control the relative roughness of the predicted values of $\hat{y} = X\hat{\beta}$, we introduce a term which measures deviations from the zero vector, which would be the expected value of the vector $\vec{y}$ if it were a normal sample. To construct such a roughness term, we began with a weighted first differences matrix D multiplied by the fitted vector $\hat{y} = X\hat{\beta}$; the weighted first difference provides a discrete analog of the first derivative, allowing for a measure of the slope between adjacent points in the order statistic. This approach is motivated by the fact that a smooth function estimating each point in the order statistic would have a slope of small magnitude, with little to no change between adjacent values. This transformation is represented by the matrix:

Where σ is the covariance matrix of $\vec{y}$, such that each ϕ_i term provides a measure of the average standard deviations between neighboring points in the order statistic. Multiplying D by $\hat{y} = X\hat{\beta}$ thus creates a weighted difference of adjacent predicted values, where larger values of $DX\hat{\beta}$ correspond to rapid changes in adjacent predicted values $\hat{y}$, and therefore a more rough prediction function. Summing the square of these differences via $(\beta^T X^T D^T)(DX\beta)$ provides a nonnegative measure of roughness which can be added to the GLS objective function.

$$D = \begin{bmatrix} \varphi_1 & -\varphi_1 & 0 & \dots & 0 \\ 0 & \varphi_2 & -\varphi_2 & \dots & 0 \\ 0 & 0 & \varphi_3 & -\varphi_3 & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \varphi_{n-1} & -\varphi_{n-1} \end{bmatrix}, \ \varphi_i = \sqrt{\frac{\left(\Sigma_{i,i} + \Sigma_{i+1,i+1}\right)}{2}}$$

Fig. 4: Differences Matrix D and entries ϕ_i

$$\hat{\beta}_{PGLS} = argmin_{\beta}(\vec{y} - X\vec{\beta})^T \sigma^{-1}(\vec{y} - X\vec{\beta}) + \lambda(\beta^T X^T D^T)(DX\beta) \tag{19}$$

Controlling the relative weighting of the pure GLS term and roughness term in the solution function via the regularization parameter λ provides a means to specify the tradeoff between the fidelity and smoothness of the estimates. The use of the ϕ_i term in the weighted difference matrix D enforces smoothing according to the relative contribution of each element of $\vec{y}$ to the overall variance, rather than smoothing each one equally. The regularization factor λ then allows us to optimize the degree of smoothing of the fitted value vector $\hat{y}$.

We simplify the calculations of the roughness parameter by naming the internal component P:

$$P = X^T D^T DX \tag{20}$$

Thus, the complete roughness penalty term is given by $\lambda(\beta^T X^T D^T)(DX\beta) = \lambda\beta^T P\beta$, yielding the complete penalized GLS solution to the objective function:

$$\hat{\beta}_{PGLS} = argmin_{\beta}(\vec{y} - X\vec{\beta})^T \sigma^{-1}(\vec{y} - X\vec{\beta}) + \lambda\beta^T P\beta \tag{21}$$

The Penalized GLS (PGLS) objective function is:

$$f_{PGLS}(\vec{\beta}; \vec{y}; \lambda) = (\vec{y} - X\vec{\beta})^T \sigma^{-1}(\vec{y} - X\vec{\beta}) + \lambda\vec{\beta}^T P\beta$$

Taking the partial derivatives of the quadratic forms yields:

$$\frac{\delta f_{PGLS}}{\delta\beta} = 2(\vec{y} - X\vec{\beta})^T \sigma^{-1}(-X) + 2\lambda\vec{\beta}^T P$$

Setting the partial derivatives to zero and solving gives:

$$\begin{aligned} 2(\vec{y} - X\vec{\beta})^T\sigma^{-1}(-X) + 2\lambda\vec{\beta}^T P &= 0 \\ (\vec{y} - X\vec{\beta})^T\sigma^{-1}(-X) + \lambda\vec{\beta}^T P &= 0 \\ -\vec{y}^T\sigma^{-1}X + \vec{\beta}^T X^T\sigma^{-1}X + \lambda\vec{\beta}^T P &= 0 \\ \vec{\beta}^T X^T\sigma^{-1}X + \lambda\vec{\beta}^T P &= \vec{y}^T\sigma^{-1}X \end{aligned}$$

Transpose both sides and using the fact that σ^{-1} and P are symmetric gives:

$$\begin{aligned} (\vec{\beta}^T X^T\sigma^{-1}X + \lambda\vec{\beta}^T P)^T &= (\vec{y}^T\sigma^{-1}X)^T \\ X^T\sigma^{-1}X\vec{\beta} + \lambda P\vec{\beta} &= X^T\sigma^{-1}\vec{y} \\ (X^T\sigma^{-1}X + \lambda P)\vec{\beta} &= X^T\sigma^{-1}\vec{y} \end{aligned}$$

Multiplying both sides by the inverse of $(X^T\sigma^{-1}X + \lambda P)$ yields the solution

$$\begin{aligned} (X^T\sigma^{-1}X + \lambda P)(X^T\sigma^{-1}X + \lambda P)\vec{\beta} &= (X^T\sigma^{-1}X + \lambda P)(X^T\sigma^{-1}\vec{y}) \\ \vec{\beta} &= (X^T\sigma^{-1}X + \lambda P)^{-1}X^T\sigma^{-1}\vec{y} \end{aligned}$$

Thus the estimate $\hat{\beta}_{PGLS}$ which minimizes the objective function is :

$$\hat{\beta}_{PGLS} = (X^T\sigma^{-1}X + \lambda P)^{-1}X^T\sigma^{-1}\vec{y} \tag{22}$$

The coefficient vector estimate $\hat{\beta}$ will provide the optimal weights to estimate the smoothed values of $\hat{y}$ when multiplied by the basis vectors X, for the typical prediction equation $\hat{y} = \hat{\beta}X$.

In fact, we can use the prediction equation to define a useful shorthand for our smoothing process. Starting with the predicted values and expanding our $\hat{\beta}$ using equation 12 above:

$$\hat{y} = X\hat{\beta}_{PGLS} = X(X^T\sigma^{-1}X + \lambda P)^-1X^T\sigma^{-1}\vec{y}$$

If we define $X(X^T\sigma^{-1}X + \lambda P)^{-1}X^T\sigma^{-1} = H$, we produce a smoothing matrix H which directly transforms the observed $\vec{y}$ into the smoothed predictions $\hat{y}$, since $\hat{y} = H\vec{y}$. The trace of H provides the equivalent degrees of freedom of the smoothing process, where higher degrees of freedom are associated with less smoothing of the predicted values, which provides an important metric to optimize our choice of lambda, described further in section 3.6 below.

3.3 Choice of the Design Matrix X

Clearly, the design matrix X must have n rows in order to produce n predictions of the empirical process. Since the number of columns in X is equal to n, some penalization of the GLS model is required in order to avoid a simple interpolation of the input sample $\vec{y}$. This would be a model with full fidelity, by simply tracing the points between each order statistic in the sample y, but with no smoothness, thus resulting in overly high variance.

The penalization term allows the model to enforce smoothing on the estimates, introducing a controlled amount of bias to reduce the variance. The need to introduce the smoothing penalty therefore makes the model analogous to smoothing spline estimates, in that it attempts to approximate a smooth function to predict the sample order statistics from high-variance data.

For the design matrix X used in the PGLS estimation in the tests below, we chose to use the eigenvectors of the covariance matrix of the empirical process, σ. This selection is based on the property that eigenvectors capture the direction of the departure from zero in order of highest to lowest variance, as described in section 2.5 above. The first eigenvector, as ranked by the scale of its corresponding eigenvalue, will be the vector capturing the deviation from zero in the empirical process $\vec{y}$ in the direction of greatest variance. Each subsequent eigenvector captures the next highest variance, in a direction orthogonal to all other eigenvectors, making the set of eigenvectors a complete orthonormal basis.

For $\vec{y}$ samples drawn from non-normal distributions which are close to normal, the empirical process values should deviate from the zero vector in approximately the same direction as those for samples from normal distributions. This is what makes them particularly difficult to measure and prevents tests like the Shapiro-Wilk from being completely omnibus; the power of the test is too low when the distribution in question is too similar to the normal. By using the eigenvectors of the covariance matrix σ as the design matrix X, the penalized GLS model is able to measure deviations in the direction in $\mathbb{R}^n$ corresponding to a normal distribution, thereby simplifying the problem to simply detecting the magnitude of deviation. This magnitude is then captured in the $\hat{\beta}_{PGLS}$ coefficients, hence motivating their use in the test statistic.

3.4 Construction of the Test Statistic

The expected value of $\hat{\beta}_{PGLS}$ is the zero-vector under the null hypothesis that the sample is from a normally distributed population. Therefore, we know that a significant deviation of $\hat{\beta}_{PGLS}$ from the zero vector would indicate that the sample was drawn from a non-normal distribution. To measure the significance of this deviation, we calculate a distance metric g, which is the Mahalanobis distance of $\hat{\beta}_{PGLS}$ from the zero vector, as follows:

$$g = \hat{\beta}_{PGLS}^T C^{-1} \hat{\beta}_{PGLS} \tag{23}$$

Where C is the covariance matrix of $\hat{\beta}_{PGLS}$ when the null hypothesis is true and thus the samples are drawn from a known normal distribution.

To use the distance metric g as a test statistic for our hypothesis test of normality, we must define a cutoff threshold g_α based on the significance level of the test, α. Using simulation, we determined the value of g_α by estimating the $100(1-\alpha)$ percentile of g from a sample of g values computed from normal pseudo-random samples.

If a given sample $\vec{y}$ has a g test statistic derived from $\hat{\beta}_{PGLS}$ which equals or exceeds the threshold g_α, then the null hypothesis that it is drawn from a normal distribution is rejected at significance level α.

3.5 Simulation for Known Non-Normal Distributions

To test the penalized GLS method against Shapiro-Wilk for a range of non-normal distributions, we ran 10,000 simulations per distribution, constructing empirical process $\vec{y}$ vectors for samples from either the t-, chi-square, gamma, or generalized pareto distributions. *vecy* was then passed into the PGLS equation, using the same design matrix X, covariance inverse σ^{-1}, and penalty matrix P from above, to estimate a new vector of coefficients, $\hat{\beta}_{PGLS}$. The Mahalanobis distance between this vector and the zero vector was then calculated yielding the test statistic $g = \hat{\beta}_{PGLS}^T C^{-1} \hat{\beta}_{PGLS}$. If the resulting distance g met or exceeded the threshold g_α, the null hypothesis of normality was rejected. The resulting power estimate of the PGLS test approach was then compared to that of the Shapiro-Wilk test for each distribution.

4 Empirical Results

The calculation of the empirical process, penalized GLS model, test statistic g, and application to various distributions are implemented via simulation using the R statistical software. The full code for the implementation in R is provided in the appendix below.

In order to study the behavior of the test in small samples, we used $n = 20$ for our simulations. The λ value of 2 million was used, in order to provide four effective degrees of freedom for the model, as given by the trace of the H smoothing matrix. The penalized GLS normality test was compared against the Shapiro-Wilk (SW) test for three well known distributions using a range of degrees of freedom: the t-distribution, the chi-square distribution, and the gamma distribution. In order to test the performance of the test in the presence of a range of skew and kurtosis values for a distribution, the generalized pareto distribution was also tested.

The penalized GLS test performed similarly to the SW test for the t-distribution across all degrees of freedom, but was slightly less powerful for the gamma distribution (which includes the exponential distribution) and the chi-square distribution. The penalized GLS test also performed similarly to SW for the generalized pareto distribution for various degrees of skew and kurtosis, though the low performance of SW for L-skew and L-kurtosis values near zero demonstrate the limitations of SW, and thus the opportunity for an optimized version of the PGLS model to develop a truly omnibus normality test.

4.1 Comparison of t-, chi-square, and gamma distribution results

The results of our simulated power tests for 10,000 iterations are provided in the figures below. The power of the test is plotted on the y-axis and degrees of freedom on the x-axis; the power of the test indicates the proportion of simulations which the test was able to correctly reject the null hypothesis that the sample was drawn from a non-normal distribution.

The PGLS method performed similarly to the SW test for the t-distribution, with only a slightly lower power level for all degrees of freedom ranging from 1 to 10. Both tests rejected the null hypothesis in over 80 percent of iterations with a t-distribution of 1 degree of freedom, with the power of each test dropping sharply as the degrees of freedom of the distribution increased. This drop is expected, since a t-distribution with more degrees of freedom is increasingly difficult to distinguish from a normal distribution.

The PGLS method performed slightly worse for the gamma distribution, trailing the SW test in power by 10 to 20 percentage points depending on the degrees of freedom; the exponential distribution, provided by the case when gamma has a single degree of freedom, had a gap of about 10 percentage points between the PGLS method and SW test.

The chi-square distribution had the largest gap in power between the two methods, with PGLS significantly underperforming compared to SW at all degrees of freedom greater than 1.

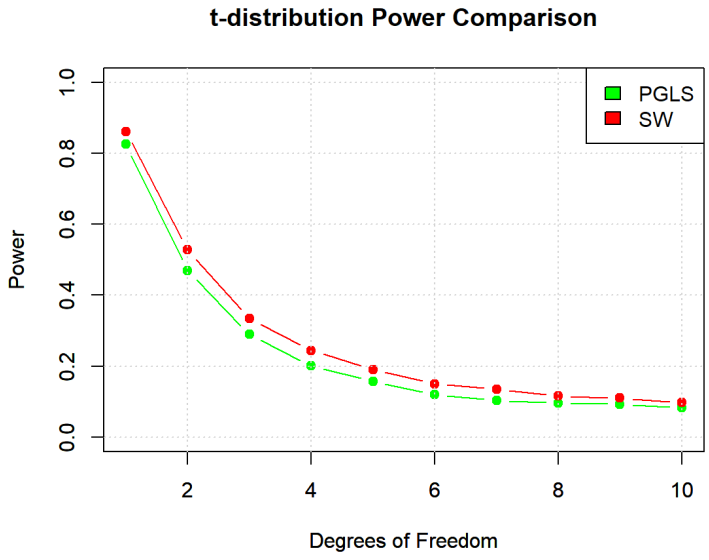


Fig. 5: Power Test Comparison for t-distribution

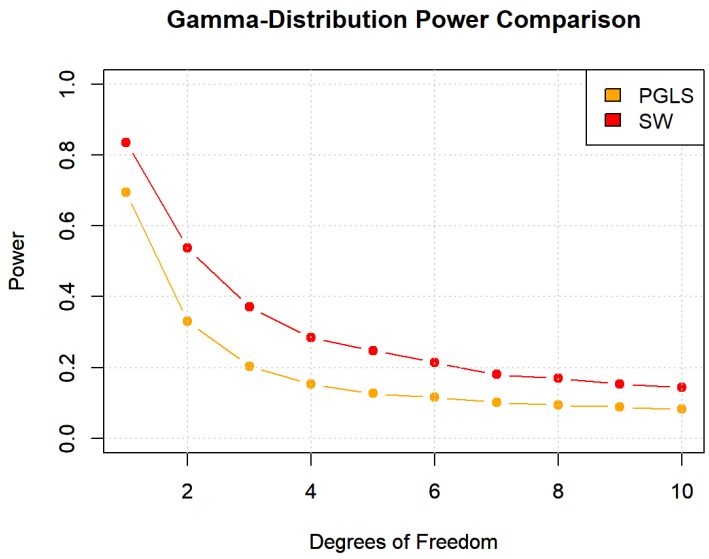


Fig. 6: Power Test Comparison for Gamma Distribution

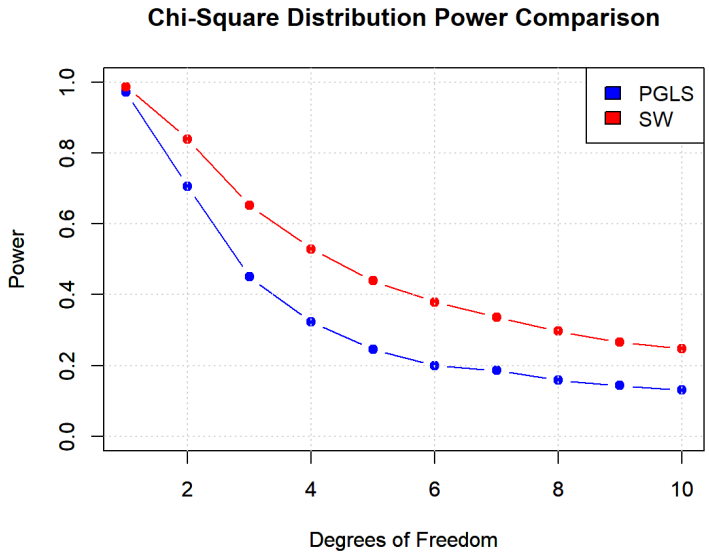


Fig. 7: Power Test Comparison for Chi-Square Distribution

While the lower power level of the PGLS test does not yet indicate that it can be applied as an omnibus normality test, its relatively similar performance across all degrees of freedom demonstrates that it is, in principle, a viable approach.

4.2 Comparison for Generalized Pareto Distribution

The results for the generalized pareto distribution are more interesting, since the SW test performed particularly poorly for distributions which were very nearly normal. GPD samples with near zero L-skew or near zero L-kurtosis both had power levels approaching zero; this aligns with the intuition that it is increasingly difficult for any test to distinguish between samples which are truly normal and those which are normal with a small amount of skew or kurtosis introduced.

The PGLS test performed similarly to the SW test again, with slightly lower power in the presence of moderate amounts of L-skew, but the similar shape of the curves for both tests again indicates that PGLS is a valid normality test.

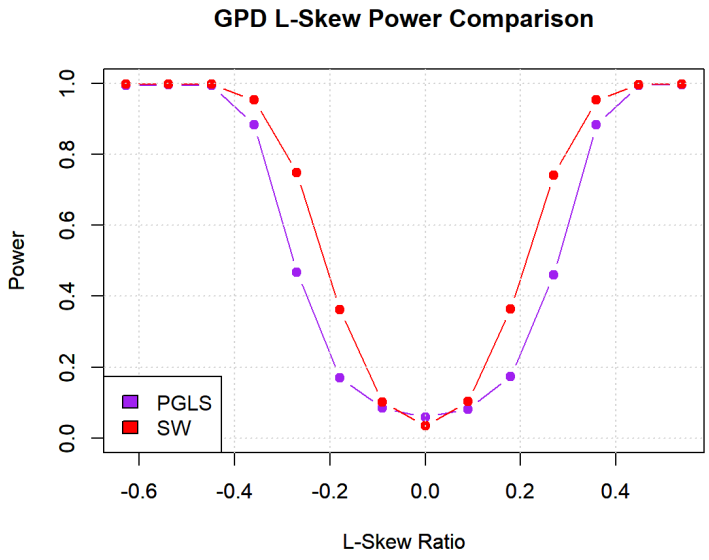


Fig. 8: Power Test Comparison for GPD Distribution across L-Skew ratios

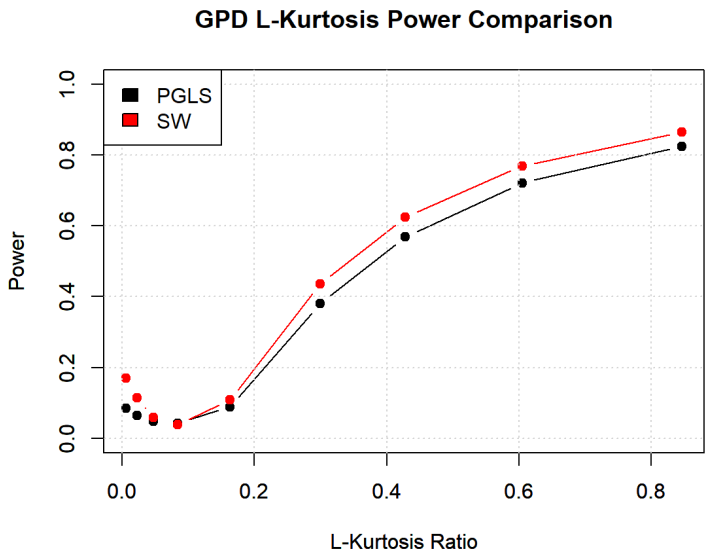


Fig. 9: Power Test Comparison for GPD Distribution across L-Kurtosis Ratios

5 Conclusions and Further Research

While the tests above do not demonstrate that the penalized GLS model outperforms the SW in terms of power, the similarity of results to SW is nonetheless promising. The penalized GLS approach provides a flexible method for normality testing by measuring the deviations in the empirical process $s^*(i)$ from zero. The approach offers several avenues for further exploration: the choice of the penalty weight λ, the construction of the penalty matrix P, the choice of the basis vectors in the design matrix X, and the construction of the test statistic g.

The choice of the λ parameter clearly introduces flexibility into the model, by determining how much weight should be placed on the penalization matrix P in the PGLS formula. For the tests above, we selected a lambda value of 2 million, in order to approximate 4 effective degrees of freedom, given by the trace of the smoothing matrix H. We note, however, that the choice of λ did not seem to have a significant impact on the power of the subsequent tests. Since λ controls the degree of smoothing for the predictions, which is central to the concept of the approach, the limited impact of variations in λ on the power test results deserves further investigation.

An obvious next step would be many iterations of the tests described above for a range of λ values to test the subsequent impact on the power calculations. Given the number of calculations involved, however, an exhaustive search would be computationally difficult. As such, an improved theoretical grounding for the selection of λ and construction of the penalty matrix P would help determine a potential range of optimal values. It is also possible that the optimal λ depends on the specification of the other parameters, or even on the distribution from which the test sample is being drawn.

In addition, the form of the penalty matrix P offers avenues for further investigation, since a range of penalty matrices are possible. Our choice of $P = X^T D^T DX$, or—after including the $\vec{\beta}$ vectors—the full penalty term $\vec{\beta}^T X^T D^T DX\vec{\beta}$, is motivated by a smoothing spline approach to the prediction function, as described in section 3.2 above. Measuring the slope between each subsequent estimate in the predicted values $\hat{y} = X\hat{\beta}$ is a useful proxy for the overall roughness of the prediction function. Note that this is strictly a localized model, since it only incorporates neighboring prediction values via the weighted first-order differencing matrix D. Penalization matrices which provide more global estimates of roughness are simple to implement, such as by incorporating more off-diagonal terms in D. The choice of ϕ value along the expanded diagonal in D also provides flexibility; the tests below used a proxy for the average standard deviation of the neighboring columns in the design matrix X in order to provide relatively higher weights to high-variance order statistics, but in principle a range of weighting values are possible.

The choice of the design matrix X also provides flexibility for specifying the penalized GLS equation, since it does not depend on the sample $\vec{y}$ to be tested. For the tests above, we chose to use the eigenvectors of the covariance matrix σ. We use the set of eigenvectors as basis vectors for the PGLS model because they provide an orthonormal system in $\mathbb{R}^n$ which provides the direction of variance in n-space in which the magnitude of deviation from zero is maximal. The directions of the eigenvectors are particularly useful for samples which are drawn from distributions which are close to normal. These distributions are the most difficult to distinguish since their direction of the deviation is so similar to normal, as described in section 3.3 above. Thus, the choice of design matrix as the eigenvectors of a covariance matrix which describes departures in the empirical process from the zero vector should theoretically allow the model to perform well even on distributions which are nearly normal.

While the choice of the design matrix X as the eigenvectors of has a strong theoretical grounding, a range of design matrices are possible. As such, exploring other design matrices, such as eigenvectors of alternate covariance matrices, could provide further avenues for optimization of the test.

It would also be possible to construct an alternative test statistic. Our metric $g = \hat{\beta}_{PGLS}^T C^{-1} \hat{\beta}_{PGLS}$ gives equal weights to each $\hat{\beta}_{PGLS}$ coefficient in the distance calculation, but it is possible to specify particular weights for each element in the vector $\hat{\beta}_{PGLS}$. An even simpler approach would be to simply take the sum of the residuals squared, since these also capture the distance from the predicted values to the observed $\vec{y}$.

The tests above serve as a proof of concept rather than thorough review of an optimized model, with many opportunities remaining to explore variations on the PGLS approach. By carefully explaining each component of the empirical process, penalized GLS model, and test statistic, we hope to encourage further research in this novel domain and contribute to the development of a truly omnibus normality test.

6 References

Aitken, A. C. (1935). "On Least Squares and Linear Combinations of Observations". Proceedings of the Royal Society of Edinburgh. 55: 42-48.

Craven, P., and Wahba, G. (1979). "Smoothing noisy data with spline functions". Numerische Mathematik. 31 (4): 377-403.

Green, P. J.; Silverman, B.W. (1994). "Nonparametric Regression and Generalized Linear Models: A roughness penalty approach". Chapman and Hall.

Hoerl, Arthur E., and Kennard, Robert W. (1970). "Ridge Regression: Biased Estimation for Nonorthogonal Problems". Technometrics. 12 (1): 55-67.

Shapiro, S. S., and Wilk, M. B. (1965). "An Analysis of Variance Test for Normality (Complete Samples)". Biometrika, Vol. 52, No. 3/4 (Dec., 1965), pp. 591-61.

Tukey , J. W. (1960). "The Practical Relationship Between the Common Transformations of Percentages or Fractions and of Amounts,". Princeton: Statistical Research Group. Technical Report 36.

Van Staden, P., and Loots, M.T. (2009). "Method of L-moment estimation for the generalized lambda distribution". Third Annual ASEARC Conference. December 7-8, 2009, Newcastle, Australia.

Acknowledgments

Many thanks to my thesis advisor Dr. Mark Inlow for all of his support, great advice, and many hours on video calls helping me complete this paper.

Appendix: R Code to Reproduce Results

6.1 Main PGLS Test Script

```
#### Main Penalized General Least Squares Test ####

len <- 20
sim_iter <- 1000000
y_mean <- 0
y_sd <- 1
lambda <- 20

# application of standardization the probability integral transform

source("calc_sim_mat.R")

y <- rnorm(n=len,mean=y_mean,sd=y_sd)

# calculation of the empirical process via simulation

sim_mat <- calc_sim_mat(len=length(y),mean=mean(y),sd=sd(y),iter=sim_iter)

# covariance matrix of the standardized probability integral transform

cov_mat <- cov(sim_mat)
cov_inv <- solve(cov_mat)

# differences matrix with standard deviations per column vector of sim_mat
   ↪ , through n-1

D_sigma <- cbind(diag(19),numeric(19))
for (row in 1:dim(D_sigma)[1]){
  D_sigma[row,row+1] <- -1
  var_av <- sqrt((cov_mat[row,row] + cov_mat[row+1,row+1])/2)
  D_sigma[row,] <- D_sigma[row,] * var_av
}

# penalty matrix of D_sigˆ{\, T} * D_sig; results in 20x20 square matrix

# calculation of eigenvectors and design matrix X

eigen_cov <- eigen(cov_mat)

X <- eigen_cov$vectors
```

```
# calculation of P matrix

P_mat <- t(X) %*% t(D_sigma) %*% D_sigma %*% X

interior <- t(X) %*% cov_inv %*% X

interior_lambda <- interior + (lambda * P_mat)
int_inv <- solve(interior_lambda)

# Approximating the effective degrees of freedom via trace of H

H <- X %*% int_inv %*% t(X) %*% cov_inv
print(sum(diag(H)))

#### Normal Iterations Test for Expected Value ####

source("pgls.R")

iter <- 10000

beta_mat <- matrix(0,nrow=iter,ncol=len)

for (i in 1:iter) {
  y <- rnorm(n=len,mean=y_mean,sd=y_sd)
  beta_mat[i,] <- pgls(y=y,X=X,cov_inv=cov_inv,int_inv=int_inv)
}

cov_beta <- cov(beta_mat) #C
cov_beta_inv <- solve(cov_beta) #C^-1

#### Calculating 95th percentile of distance metric by simulation for
   ↪ normal input vector ####

g_result <- numeric(iter)

for (i in 1:iter) {
  y <- rnorm(n=len,mean=y_mean,sd=y_sd)
  beta_hat <- pgls(y=y,X=X,cov_inv=cov_inv,int_inv=int_inv)
  g_result[i] <- t(beta_hat) %*% cov_beta_inv %*% beta_hat

}

g_alpha <- quantile(g_result,probs = 0.95)

source("dist_iter.R")
```

```
source("rlambda.R")

#### Exponential Iterations Test for Expected Value ####

dist_iter(X=X,cov_inv=cov_inv,int_inv=int_inv,g_alpha=g_alpha,type="exp",
   ↪ iter=iter,len=len,df=1)

#### Uniform Iterations Test for Expected Value ####

dist_iter(X=X,cov_inv=cov_inv,int_inv=int_inv,g_alpha=g_alpha,type="uni",
   ↪ iter=iter,len=len,df=1)


#### t-distribution Iterations Test for Expected Value ####
df_index <- 1:10

t_results <- matrix(0,nrow=10,ncol=2)
for (i in df_index) {
  print(i)
  t_results[i,] <- dist_iter(X=X,cov_inv=cov_inv,int_inv=int_inv,g_alpha=g_
     ↪ alpha,
                          type="t",iter=iter,len=len,df=i)
}

#### Chi-square Iterations Test for Expected Value ####
chi_results <- matrix(0,nrow=10,ncol=2)
for (i in 1:10) {
  chi_results[i,] <- dist_iter(X=X,cov_inv=cov_inv,int_inv=int_inv,g_alpha=
     ↪ g_alpha,
                            type="chi",iter=iter,len=len,df=i)
}

#### Gamma Iterations Test for Expected Value ####
g_results <- matrix(0,nrow=10,ncol=2)
for (i in 1:10) {
  g_results[i,] <- dist_iter(X=X,cov_inv=cov_inv,int_inv=int_inv,g_alpha=g_
     ↪ alpha,
                           type="gamma",iter=iter,len=len,df=i)
}

### Generalized Pareto Distribution ###
source("rpareto.R")
source("g_pareto_prep.R")
```

```
skew_tab <- g_pareto_prep(g_alpha = 0,g_beta = 1,g_delta=0.5,g_lambda =
    ↪ 4.256,type="skew")

kurt_tab <- g_pareto_prep(g_alpha = 0,g_beta = 1,g_delta=0.5,g_lambda =
    ↪ 4.256,type="kurt")

gp_results_s <- matrix(0,nrow=dim(skew_tab)[1],ncol=2)
for (i in 1:dim(skew_tab)[1]) {
  gp_results_s[i,] <- dist_iter(X=X,cov_inv=cov_inv,int_inv=int_inv,g_alpha
      ↪ =g_alpha,
                                type="gpareto",iter=iter,len=len,df=skew_tab[i
                                    ↪ ,1],
                                l1=skew_tab[i,2],l2=skew_tab[i,3],l3=skew_tab[i
                                    ↪ ,4],l4=skew_tab[i,5])
}

gp_results_k <- matrix(0,nrow=dim(kurt_tab)[1],ncol=2)
for (i in 1:dim(kurt_tab)[1]) {
  gp_results_k[i,] <- dist_iter(X=X,cov_inv=cov_inv,int_inv=int_inv,g_alpha
      ↪ =g_alpha,
                                type="gpareto",iter=iter,len=len,df=kurt_tab[i
                                    ↪ ,1],
                                l1=kurt_tab[i,2],l2=kurt_tab[i,3],l3=kurt_tab[i
                                    ↪ ,4],l4=kurt_tab[i,5])
}


# Plotting all results

all_plots(len=len,t_results=t_results,chi_results=chi_results,g_results=g_
    ↪ results,
                    skew_tab=skew_tab,gp_results_s=gp_results_s,
                    kurt_tab=kurt_tab,gp_results_k=gp_results_k)
```

6.2 Function to Generate All Plots

```
### Plots for Reproducing Paper
source("std_res.R")

all_plots <- function(len,t_results,chi_results,g_results,
                    skew_tab,gp_results_s,kurt_tab,gp_results_k){

  ### Example Plots Describing std_residual process
```

```
n_plots <- 10
for (i in 1:n_plots){
  y <- rnorm(n=len,mean=0,sd=1)
  s_y <- std_res(y=y)
  plot(1:len,s_y, type = "l", pch=19, main="Empirical Process for Normal
     ↪ Distributions",
       col = palette(rainbow(n_plots))[i] , xlab = "i index", ylab = "s*(i
          ↪ )",ylim = c(-0.25,0.25))
  if (i<n_plots){
    par(new=TRUE)
  } else if (i==n_plots){
    grid()
    abline(h = 0, col = "black")
  }
}


for (i in 1:n_plots){
  y <- rt(n=len, df=2)
  s_y <- std_res(y=y)
  plot(1:len,s_y, type = "l", pch=19, main="Empirical Process for t-
     ↪ Distributions",
       col = palette(rainbow(n_plots))[i] , xlab = "i index", ylab = "s*(i
          ↪ )",ylim = c(-0.25,0.25))
  if (i<n_plots){
    par(new=TRUE)
  } else if (i==n_plots){
    grid()
    abline(h = 0, col = "black")
  }
}

for (i in 1:n_plots){
  y <- rchisq(n=len, df=2)
  s_y <- std_res(y=y)
  plot(1:len,s_y, type = "l", pch=19, main="Empirical Process for Chi-
     ↪ Square Distributions",
       col = palette(rainbow(n_plots))[i] , xlab = "i index", ylab = "s*(i
          ↪ )",ylim = c(-0.25,0.25))
  if (i<n_plots){
    par(new=TRUE)
  } else if (i==n_plots){
    grid()
    abline(h = 0, col = "black")
  }
```

```
}

for (i in 1:n_plots){
  y <- rgamma(n=len, shape=2)
  s_y <- std_res(y=y)
  plot(1:len,s_y, type = "l", pch=19, main="Empirical Process for Gamma
     ↪ Distributions",
       col = palette(rainbow(n_plots))[i] , xlab = "i index", ylab = "s*(i
          ↪ )",ylim = c(-0.25,0.25))
  if (i<n_plots){
    par(new=TRUE)
  } else if (i==n_plots){
    grid()
    abline(h = 0, col = "black")
  }
}

### Rotated kernel density plot describing empirical process
source("ex_plot.R")
plot_iter <- 100000

d_norm <- ex_plot(len=len,type="Normal",iter=plot_iter,color="green")
par(new=TRUE)
d_t <- ex_plot(len=len,type="t",iter=plot_iter,df=2,color="red")
par(new=TRUE)
d_chi <- ex_plot(len=len,type="Chi Square",iter=plot_iter,df=2,color="
   ↪ blue")
par(new=TRUE)
d_uni <- ex_plot(len=len,type="Uniform",iter=plot_iter,color='purple')
par(new=TRUE)
d_exp <- ex_plot(len=len,type="Exponential",iter=plot_iter,color='black')
par(new=TRUE)
d_gamma <- ex_plot(len=len,type="Gamma",iter=plot_iter,df=2,color='orange
   ↪ ')

grid()
legend(x = "topright",legend=c('Normal','t-distribution, df=2','Chi-Square
   ↪ , df=2','Uniform','Exponential','Gamma,df=2'),
      fill=c('green','red','blue','purple','black','orange'))



### Results plots
```

```
plot(1:10, t_results[,1], type = "b", pch = 19, main="t-distribution␣Power
   ↪ ␣Comparison",
     col = "green", xlab = "Degrees␣of␣Freedom", ylab = "Power",ylim = c
        ↪ (0,1))
par(new=TRUE)
plot(1:10, t_results[,2], type = "b", pch = 19,
     col = "red", xlab = "Degrees␣of␣Freedom", ylab = "Power",ylim = c
        ↪ (0,1))
grid()
legend(x = "topright",legend=c('PGLS','SW'), fill=c('green','red'))


plot(1:10, chi_results[,1], type = "b", pch = 19, main="Chi-Square␣
   ↪ Distribution␣Power␣Comparison",
     col = "blue", xlab = "Degrees␣of␣Freedom", ylab = "Power",ylim = c
        ↪ (0,1))
par(new=TRUE)
plot(1:10, chi_results[,2], type = "b", pch = 19,
     col = "red", xlab = "Degrees␣of␣Freedom", ylab = "Power",ylim = c
        ↪ (0,1))
grid()
legend(x = "topright",legend=c('PGLS','SW'), fill=c('blue','red'))

plot(1:10, g_results[,1], type = "b", pch = 19, main="Gamma-Distribution␣
   ↪ Power␣Comparison",
     col = "orange", xlab = "Degrees␣of␣Freedom", ylab = "Power",ylim = c
        ↪ (0,1))
par(new=TRUE)
plot(1:10, g_results[,2], type = "b", pch = 19,
     col = "red", xlab = "Degrees␣of␣Freedom", ylab = "Power",ylim = c
        ↪ (0,1))
grid()
legend(x = "topright",legend=c('PGLS','SW'), fill=c('orange','red'))


plot(skew_tab[,1], gp_results_s[,1], type = "b", pch = 19, main="GPD␣L-
   ↪ Skew␣Power␣Comparison",
     col = "purple", xlab = "L-Skew␣Ratio", ylab = "Power",ylim = c(0,1))
par(new=TRUE)
plot(skew_tab[,1], gp_results_s[,2], type = "b", pch = 19,
     col = "red", xlab = "L-Skew␣Ratio", ylab = "Power",ylim = c(0,1))
grid()
legend(x = "bottomleft",legend=c('PGLS','SW'), fill=c('purple','red'))
```

```
  plot(kurt_tab[,1], gp_results_k[,1], type = "b", pch = 19, main="GPD␣L-
     ↪ Kurtosis␣Power␣Comparison",
       col = "black", xlab = "L-Kurtosis␣Ratio", ylab = "Power",ylim = c
          ↪ (0,1))
  par(new=TRUE)
  plot(kurt_tab[,1], gp_results_k[,2], type = "b", pch = 19,
       col = "red", xlab = "L-Kurtosis␣Ratio", ylab = "Power",ylim = c(0,1))
  grid()
  legend(x = "topleft",legend=c('PGLS','SW'), fill=c('black','red'))

}
```

6.3 Calculation of Simulation Matrix Function

```
#### Residual Simulation Matrix Function

source("std_res.R")

calc_sim_mat <- function(len,mean,sd,iter){
      sim_mat <- matrix(0,nrow=iter,ncol=len)
      for (i in 1:iter) {
        y <- rnorm(n=len,mean=mean,sd=sd)
        sim_mat[i,] <- std_res(y=y) # each sample vector is stored as a row
      }

      return(sim_mat)
}
```

6.4 Standardized Residual Function

```
### Standardized Residual Function for empirical process

std_res <- function(y){
    n <- length(y)
    y_s <- (y - mean(y)) / sd(y)
    y_ss <- sort(y_s, decreasing = FALSE)
    phi_y <- pnorm(q=y_ss,mean=0,sd=1)
    i <- 1:20
    r <- phi_y - (i/(n+1))
    return(r)
}
```

6.5 Distribution and Test Calculation Function

```
# simulation of given distributions and application of test
source("pgls.R")

dist_iter <- function(X,cov_inv,int_inv,g_alpha,type,iter,len,df,l1,l2,l3,
    ↪ l4){
      beta_mat <- matrix(0,nrow=iter,ncol=len)
      D_result <- numeric(iter)
      SW_result <- numeric(iter)
      for (i in 1:iter) {
        if(type=="exp"){
          y <- rexp(n=len, rate = 1)
        } else if (type=="uni") {
          y <- runif(n=len,min=-1,max=1)
        } else if (type=="t"){
          y <- rt(n=len, df=df)
        } else if (type=="chi") {
          y <- rchisq(n=len, df=df)
        } else if (type=="gamma"){
          y <- rgamma(n=len, shape=df)
        } else if (type=="glambda"){
          y <- rlambda(n=len,l1=l1,l2=l2,l3=l3,l4=l4)
        } else if (type=="gpareto"){
          y <- rpareto(n=len,alpha=l1,beta=l2,delta=l3,lambda=l4)
        }

        beta_hat <- pgls(y=y,X=X,cov_inv=cov_inv,int_inv=int_inv)
        beta_mat[i,] <- beta_hat
        g_result[i] <- t(beta_hat) %*% cov_beta_inv %*% beta_hat
        SW_result[i] <- shapiro.test(y)$p.value <= 0.05
      }
      power <- sum(g_result >= g_alpha) / length(g_result)
      print(paste(type,df,"␣power:␣",power))

      SW_power <- mean(SW_result)
      print(paste(type,df,"␣SW␣power:␣",SW_power))

      result_mat <- matrix(0,nrow=1,ncol=2)
      result_mat[1,1] <- power
      result_mat[1,2] <- SW_power

      return(result_mat)
}
```

6.6 Helper Function for Distribution Comparison Plot

```
### Example Plots describing s(i) for various distributions
source("std_res.R")

ex_plot <- function(len,type,iter,df,l1,l2,l3,l4,color){
     plot_mat <- matrix(0,nrow=iter,ncol=len)

     for (i in 1:iter) {
       if(type=="Exponential"){
         y <- rexp(n=len, rate = 1)
       } else if (type=="Uniform") {
         y <- runif(n=len,min=-1,max=1)
       } else if (type=="t"){
         y <- rt(n=len, df=df)
       } else if (type=="Chi Square") {
         y <- rchisq(n=len, df=df)
       } else if (type=="Gamma"){
         y <- rgamma(n=len, shape=df)
       } else if (type=="Generalized Lambda"){
         y <- rlambda(n=len,l1=l1,l2=l2,l3=l3,l4=l4)
       } else if (type=="Generalized Pareto"){
         y <- rpareto(n=len,alpha=l1,beta=l2,delta=l3,lambda=l4)
       } else if (type=="Normal"){
         y <- rnorm(n=len,mean=0,sd=1)
       }

       plot_mat[i,] <- std_res(y=y)
     }
     plot_ex <- colMeans(plot_mat)

     #h<-hist(plot_ex, breaks=len, xlab="s(i) value",
      # main=paste("Empirical Process for",type,"Distribution"))

     ex_dens <- density(plot_ex,na.rm = T) # returns the density data

     plot(ex_dens, type = "b", pch=19,cex=0.5, main="Comparing
        ↪ Distributions after Empirical Process",
         col = color, xlab = "Expected s*(i) Value", ylab = "Density",xlim=
            ↪ c(-0.07,0.07),ylim = c(0,40))
}
```

6.7 Generalized Pareto Distribution Quantile Function

```
# Generalized Pareto Distribution

rpareto <- function(alpha,beta,delta,lambda,n){
     p<-runif(n=n)
     r_p <- alpha + beta*((1-delta)*((p^lambda-1)/lambda)-delta*(((1-p)^
        ↪ lambda-1)/lambda))
     return(r_p)
}
```

6.8 Generalized Pareto Table Calculation Function

```
### Generalized Pareto Distribution preparation calculations

g_pareto_prep <- function(g_alpha,g_beta,g_delta,g_lambda,type){
# delta skewness parameter -1 to 1; lambda held at

  if(type=="skew"){
    # n=len,alpha=l1,beta=l2,delta=l3,lambda=l4
    skew_tab <- matrix(0,nrow=14,ncol=5)
    skew_tab[1,] <- c(0, g_alpha, g_beta, -0.1, g_lambda)
    skew_tab[2,] <- c(0,g_alpha, g_beta, 0, g_lambda)
    skew_tab[3,] <- c(0, g_alpha, g_beta, 0.1, g_lambda)
    skew_tab[4,] <- c(0,g_alpha, g_beta, 0.2, g_lambda)
    skew_tab[5,] <- c(0, g_alpha, g_beta, 0.3, g_lambda)
    skew_tab[6,] <- c(0, g_alpha, g_beta, 0.4, g_lambda)
    skew_tab[7,] <- c(0, g_alpha, g_beta, 0.5, g_lambda) # normal
       ↪ distribution
    skew_tab[8,] <- c(0, g_alpha, g_beta, 0.6, g_lambda)
    skew_tab[9,] <- c(0, g_alpha, g_beta, 0.7, g_lambda)
    skew_tab[10,] <- c(0, g_alpha, g_beta, 0.8, g_lambda)
    skew_tab[11,] <- c(0, g_alpha, g_beta, 0.9, g_lambda)
    skew_tab[12,] <- c(0, g_alpha, g_beta, 1, g_lambda)
    skew_tab[13,] <- c(0, g_alpha, g_beta, 1.1, g_lambda)
    skew_tab[14,] <- c(0, g_alpha, g_beta, 1.2, g_lambda)

    # scaling the beta and alpha parameters to fix L1 at 0 and L2 at 1
    for (i in 1:dim(skew_tab)[1]){
      skew_tab[i,3] <- ((skew_tab[i,5]+1)*(skew_tab[i,5]+2))
      skew_tab[i,2] <- (skew_tab[i,3]*(1-2*skew_tab[i,4])/(skew_tab[i,5]+1))
    }

    l_moments <- matrix(0,nrow=dim(skew_tab)[1],ncol=4)
    for (i in 1:dim(skew_tab)[1]){
```

```
    print(i)
    a<-skew_tab[i,2]
    b<-skew_tab[i,3]
    d<-skew_tab[i,4]
    l<-skew_tab[i,5]
    l_moments[i,1] <- a - (b*(1-2*d)/(l+1))
    l_moments[i,2] <- b / ((l+1)*(l+2))
    l_moments[i,3] <- (((l-1)*(1-2*d))/(l+3))
    l_moments[i,4] <- (((l-1)*(l-2))/((l+3)*(l+4)))
    skew_tab[i,1] <- l_moments[i,3]
  }
  print("L-Moments␣Ratio,␣Skew")

  print(l_moments)
  tab <- skew_tab

} else if (type=="kurt"){
  # lambda kurt parameter -1 to 1; delta held at 0.5
  kurt_tab <- matrix(0,nrow=9,ncol=5)
  kurt_tab[1,] <- c(-0.9, g_alpha, g_beta,g_delta, -0.9)
  kurt_tab[2,] <- c(-0.7, g_alpha, g_beta,g_delta, -0.7)
  kurt_tab[3,] <- c(-0.5, g_alpha, g_beta,g_delta, -0.5)
  kurt_tab[4,] <- c(-0.3, g_alpha, g_beta,g_delta, -0.3)
  kurt_tab[5,] <- c(0, g_alpha, g_beta,g_delta, 0.01)
  kurt_tab[6,] <- c(0.3, g_alpha, g_beta,g_delta, 0.3)
  kurt_tab[7,] <- c(0.5, g_alpha, g_beta,g_delta, 0.5)
  kurt_tab[8,] <- c(0.7, g_alpha, g_beta,g_delta, 0.7)
  kurt_tab[9,] <- c(0.9, g_alpha, g_beta,g_delta, 0.9)

  for (i in 1:dim(kurt_tab)[1]){
    kurt_tab[i,3] <- ((kurt_tab[i,5]+1)*(kurt_tab[i,5]+2))
  }

  l_moments <- matrix(0,nrow=dim(kurt_tab)[1],ncol=4)
  for (i in 1:dim(kurt_tab)[1]){
    print(i)
    a<-kurt_tab[i,2]
    l<-kurt_tab[i,5]
    b<-kurt_tab[i,3]
    d<-kurt_tab[i,4]

    l_moments[i,1] <- a - (b*(1-2*d)/(l+1))
    l_moments[i,2] <- b / ((l+1)*(l+2))
    l_moments[i,3] <- (((l-1)*(1-2*d))/(l+3))
    l_moments[i,4] <- (((l-1)*(l-2))/((l+3)*(l+4)))
```

```
      kurt_tab[i,1] <- l_moments[i,4]
    }
    print("L-Moments␣Ratio,␣Kurtosis")
    print(l_moments)
    tab <- kurt_tab
  }

  return(tab)
}
```