Master Thesis

Machine Learning Methods for Approximating Nonstandard Distribution Functions

Submitted to
Faculty of Economics and Business Administration
at the
University of Duisburg-Essen
by

Name: Mehdi Belayet Lincon
Matriculation No.: 2283198
E-mail address: mehdi.lincon@stud.uni-due.de
Postal address: Bürgerstr. 2, 40219 Düsseldorf
Thesis advisor: Prof. Dr. Christoph Hanck, Chair of Econometrics
Second referee: Prof. Dr. Andreas Behr, Chair of Statistics
Course of study: Economics, M. Sc.
Semester: Summer term 2019
Due date: 20th September, 2019

Abstract

Machine learning methods offer a modern and viable alternative to more common numerical and analytical approach for approximating non-standard distributions. It is possible to create models that can accurately learn and predict values from such distributions. The application of machine learning methods is illustrated using the Dickey-Fuller distribution, which is widely-used for unit root tests. The training and test data are generated using Monte Carlo simulation. In total six different approaches have been implemented including GAM and neural network. A new `R`-package `pvurt` is created using the best performing models.

Contents

List of Figures

List of Tables

List of Abbreviations

ADF	Augmented Dickey-Fuller
CDF	Cumulative density function
DF	Dickey-Fuller
GAM	Generalized additive model
MAE	Mean absolute error
MARS	Multivariate adaptive regression spline
MLP	Multilayer perceptron
NN	Neural network
PDF	Probability density function
RMSE	Root mean squared error
RSS	Residual sum of squares

1 Introduction

Statistical distributions play a fundamental role in almost any form of statistical analysis or application. However, a convenient functional form for a given distribution is not always available and hence the applications of such a distribution function are based on some kind of approximation. For example, the probability density function (PDF) and the cumulative distribution function (CDF) of the standard normal distribution $x \sim \mathcal{N}(0,1)$ are defined as

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) \text{ and}$$
$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{1}{2}t^2\right) dt = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$$

respectively, and $\operatorname{erf}(\cdot)$ is the so-called error function. The difficulty here lies in expressing the integral for the CDF in terms of elementary functions. Hence the CDF is said to have no closed-form solution and it needs to be approximated. It is possible to find many approximations for the CDF of normal distribution. Such approximations can take a numerical approach involving, for instance Taylor series expansion as undertaken by Marsaglia (2004). Alternatively, others like Raab and Green (1961), Waissi and Rossin (1996), Bryc (2002), and Bowling et al. (2009) define the distribution functions in terms of carefully selected equations and parameters. More recent attempts from Yerukala et al. (2011) and Sokouti et al. (2017) involve the use of feedforward neural networks. In fact, Sokouti et al. (2017, p. 265) present a non-exhaustive list of more than 20 different approximations.

While the normal distribution has and continue to gather interest among researchers that allowed for so many different approximation attempts, the same cannot be said for many other distribution functions, including for the non-standard distributions. Hence, this thesis explores the application of machine learning approach for approximating the non-standard distribution functions. By taking a more data centric approach, machine learning methods offer a modern and viable alternative to more common numerical and analytical approach for approximating non-standard distributions. Based on the data, it is possible to experiment different forms of algorithms starting from the simplest to more complex models and compare them to determine the best suited model or models. The models are capable of accurately learning and predicting from the distributions. It is important to try out different approaches as there is no single method that will always provide the best results for different tasks or problems. In this thesis, I implement several supervised learning algorithms, starting with parametric approaches like polynomial regressions to more non-parametric and complex methods like GAM and neural networks. The models are implemented on simulated test dataset and then compared to derive an accurate approximation. Since, it is not possible within the scope of this thesis to consider all non-standard distribution functions, the application of the machine

learning methods will be illustrated using the Dickey-Fuller (DF) distribution, which is widely used in unit root tests. The Dickey-Fuller distribution, in simple terms, arises from the asymptotic distribution of the test statistics from unit root tests, like (augmented) Dickey-Fuller test. In this thesis, the machine learning methods are used to approximate the empirical CDF. Since the Dickey-Fuller unit root test is a left-tailed test, the cumulative probability or percentile rank can be used as a p-value in hypothesis testing.

In the literature and implementation of unit root tests in statistical software, two approaches to approximate the DF distribution can be found. Along with a brief introduction to unit root and unit root testing, both of these approaches are included in Section 2. The following section explains the Monte Carlo simulation process to generate the data. Note that a limitation of such simulation process is that it is only possible to simulate data from finite samples. Hence, it is important to carry out comprehensive simulations by increasing sample sizes to allow sound conjecture regarding the asymptotic distribution. Section 4 focuses on the various machine learning methods implemented to approximate the distribution function and the approaches are then compared in Section 5 based on their RMSE and MAE scores, simulated share of rejection of null hypothesis, model fit and p-value plots. The best performing models are then selected to create a `R`-package, that allows other researches to use the pre-trained models in their analysis. The design principles along with examples to illustrate the use of the package is included in Section 6. Section 7 summarizes and concludes the thesis. The final section is a synopsis of some ideas regarding how the thesis could be further expanded.

2 Literature Review

This section starts with a brief overview on the theory of unit root and unit root testing. The simplest unit root test is the Dickey-Fuller test (DF-test). The DF-test is a statistical hypothesis testing used to determine whether unit root exists or not. However, the test statistic does not have a normal distribution and the critical values for the hypothesis testing must be derived from simulated distribution of the test statistic. For time series involving multiple lags and other characteristics like autocorrelation, the Augmented Dickey-Fuller test (ADF-test) is used. However, since in accordance with the thesis topic, the primary focus is on the distribution of the test statistics and not the tests itself, it is sufficient to only introduce the simple DF-test. Also note that, in fact, a ADF-test would be similar to the the DF-test in a AR(1) model (Stock and Watson, 2011, p. 553). The section is concluded with an overview of related works which deal with approximating the distribution of the test statistics and few related literatures.

2.1 Unit Root and Unit Root Testing

Consider a simple AR(1) model

$$Y_t = \rho Y_{t-1} + \varepsilon_t, \tag{1}$$

where ε_i is white noise. If $\rho = 1$, then the corresponding polynomial characteristics equation $1 - \rho z = 0$ has a unit root and the model is a random walk process; the series Y_t is non-stationary. As a consequence of the presence of unit root, the distribution of the test or t-statistics is non-normal, even asymptotically; depending on the type of the tests, there are even different asymptotic distributions (Stock and Watson, 2011, p. 549; Davidson and MacKinnon, 2009, p. 614). Other issues that arise in the presence of unit root are biased regression coefficients and possible spurious regressions (Stock and Watson, 2011, p. 549–550).

Based on equation 1, subtracting Y_{t-1} from both sides yield

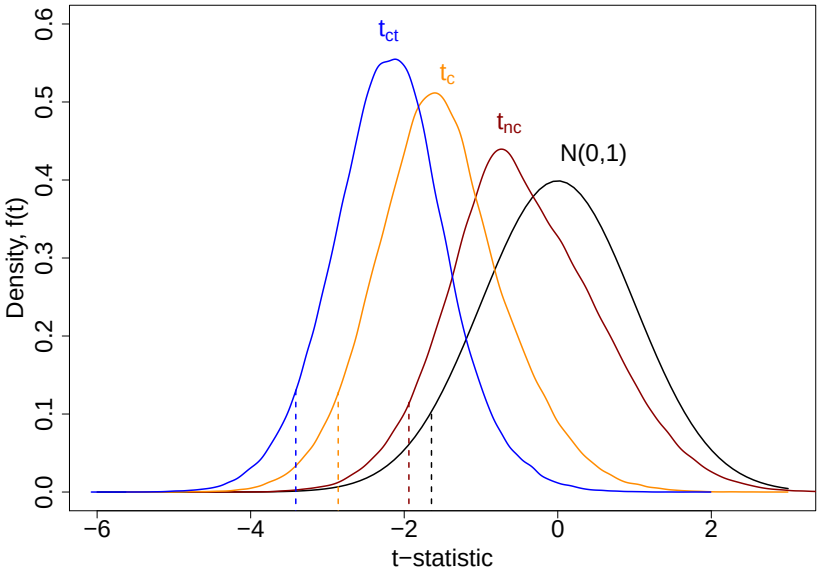
$$\Delta Y_t = (\rho - 1)Y_{t-1} + \varepsilon_t = \gamma Y_{t-1} + \varepsilon_t. \tag{2}$$

where $\gamma = \rho - 1$. If $\gamma < 0$, then the process is stationary and has no unit root. Alternatively if $\gamma = 0$, then there is unit root and it is non-stationary. Therefore testing for $\gamma = 0$ is equivalent to testing that $\rho = 1$. Hence, in the DF-test, the null hypothesis H_0:$\gamma = 0$ is tested against H_0:$\gamma < 0$. Dickey and Fuller (1979, p. 427–428) also suggest two further regressions to test for the presence of unit root:

$$Y_t = \mu + \rho Y_{t-1} + \varepsilon_t \Rightarrow \Delta Y_t = \mu + \gamma Y_{t-1} + \varepsilon_t, \tag{3}$$

$$Y_t = \mu + \beta t + \rho Y_{t-1} + \varepsilon_t \Rightarrow \Delta Y_t = \mu + \beta t + \gamma Y_{t-1} + \varepsilon_t \tag{4}$$

Figure 1: Asymptotic density distribution of Dickey-Fuller test statistics. The dotted lines indicate the significance level $\alpha = 0.05$. Source: Own illustration based on Davidson and MacKinnon (2009, p. 607).



The above two specifications differ from regression 2 through the presence of additional deterministic terms (Dickey and Fuller, 1979, p. 427–428; Enders, 2014, p. 206). While regression 2 represents a pure random walk model, the second one adds a drift term by including an intercept, while the third includes an additional linear time trend (Enders, 2014, p. 206). Let t_{nc}, t_c and t_{ct} represent the test statistics from each of the three regressions respectively.

As already mentioned above, the distribution of the t-statistic is not normally distributed, even asymptotically, and it is also different based on the test type or specification, i.e. inclusion of drift and/or trend terms. This can be easily observed in Figure 1. The figure includes the asymptotic density distributions of the three test statistics along with the PDF of normal distribution and marks the critical values at significance level $\alpha = 0.05$ (dotted vertical lines). In comparison to the normal distributions, each of the three distributions are situated to the left of it and hence the critical values are also more negative (Davidson and MacKinnon, 2009, p. 607). This illustrates clearly why using the normal distribution for testing will lead to the wrong conclusions. In the following, the asymptotic definitions of the distributions for the three test specifications are stated in terms of standard Wiener or Brownian motion processes $W(t), t \in [0, 1]$. The derivation of these distributions is beyond the scope of this thesis but can be found in, for example, Hayashi (2000, p. 567 – 582) or Hamilton (1994, p. 477 – 500). The asymptotic distributions are then given as (Hayashi, 2000, p. 575–582)

$$t_{nc} \xrightarrow[\text{d}]{} \frac{W(1)^2 - 1}{2\left\{\int_0^1 W(r)^2 dr\right\}^{1/2}}, \tag{5}$$

$$t_c \xrightarrow[\text{d}]{} \frac{W^c(1)^2 - W^c(0)^2 - 1}{2\left\{\int_0^1 W^c(r)^2 dr\right\}^{1/2}}, \text{ and} \tag{6}$$

$$t_{ct} \xrightarrow[\text{d}]{} \frac{W^{ct}(1)^2 - W^{ct}(0)^2 - 1}{2\left\{\int_0^1 W^{ct}(r)^2 dr\right\}^{1/2}}, \tag{7}$$

where $W^c(\cdot)$ is a demeaned standard Weiner process defined as $W^c(r) = W(r) - \int_0^1 W(s)ds$ and $W^{ct}(\cdot)$ is a detrended standard Wiener process defined as $W^{ct}(r) = W(r) - \int_0^1 (4-6s)W(s)ds - r\int_0^1 (-6+12s)W(s)ds$. Although Abadir (1995) and Dietrich (2002) have proposed closed form expressions, in general, it is still considered that the distributions have no simple analytical solution (Enders, 2014, p. 202; Davidson and MacKinnon, 2009, p. 618). Hence, the critical and p-values are determined by approximations through simulation studies. So far, the most widely used approximations have been provided by Dickey (1976), Fuller (1996), and MacKinnon (1994, 1996, 2010), which are introduced in the following section.

2.2 Approximation

Dickey (1976, p. 53) is one of the first to provide critical values for the unit root test statistics in the form of empirical percentile values tables.[1] The critical values are determined by simulation studies, i.e. generating sufficiently large number of t-statistics and then estimating the corresponding quantities at the required levels. Depending on the test type, the t-statistics are computed by using either one of the three regression equations 2 to 4. The p-value can then be approximated via simple two-fold linear interpolation. This is illustrated in the following example:

- Assume that for a given time series of size $n = 70$ with no drift and trend, the test statistics is $t = -1.8$.
- The corresponding table from Fuller (1996, p. 642) with the empirical percentiles is shown below. The rows and columns of interest for the given the n and t values have been shaded.

n	0.01	0.025	0.05	0.10	0.50	0.90	0.95	0.975	0.99
25	−2.65	−2.26	−1.95	−1.60	−0.47	0.92	1.33	1.70	2.15
50	−2.62	−2.25	−1.95	−1.61	−0.49	0.91	1.31	1.66	2.08
100	−2.60	−2.24	−1.95	−1.61	−0.50	0.90	1.29	1.64	2.04
250	−2.58	−2.24	−1.95	−1.62	−0.50	0.89	1.28	1.63	2.02
500	−2.58	−2.23	−1.95	−1.62	−0.50	0.89	1.28	1.62	2.01
$\infty^\dagger$	−2.58	−2.23	−1.95	−1.62	−0.51	0.89	1.28	1.62	2.01

† During interpolation, a large sample size like $n = 100,000$ is used instead.

- Since $n = 70$, the rows corresponding to $n = 50$ and $n = 100$ are chosen and then linearly interpolated to derive the approximated quantiles:

n	0.01	0.025	0.05	0.10	0.50	0.90	0.95	0.975	0.99
70	−2.612	−2.246	−1.950	−1.610	−0.494	0.906	1.302	1.652	2.064

[1] The tables have been subsequently republished in further publications. Appendix A includes the tables from Fuller (1996, p. 642).

- The second interpolation finally computes the p-value. The two nearest quantiles to the given test statistics are -1.950 and -1.610 at p-value 0.05 and 0.10 respectively. Hence, the p-value to $t = -1.8$ is 0.072.

In comparison, MacKinnon (1994, 1996, 2010) uses response surface regressions to derive the quantiles.[2] The response surface regressions has the form[3]

$$q^p(n_i) = \theta^p_\infty + \theta^P_1 n_i^{-1} + \theta^P_2 n_i^{-2} + \theta^P_3 n_i^{-3} + \varepsilon_i \tag{8}$$

where $q^p(n_i)$ is the estimated quantile at p and n is the sample size (MacKinnon, 1996, p. 604). Observe that as n tends to infinity, n^{-1} and n^{-2} tends to zero. Thus, θ^p_∞ is the estimate of p quantile of the asymptotic distribution. The rest of the parameters determines the shape of the response surface and allow the finite-sample distribution to be different from the asymptotic distribution (MacKinnon, 1996, p. 604, 2010, p. 6). Due to presence of heteroskedasticity, the response surface regressions are estimated using GMM (MacKinnon, 1994, p. 169, 1996, p. 9) or flexible GLS (MacKinnon, 2010, p. 7, 12), both of which nevertheless provided similar results (MacKinnon, 2010, p. 12). Since Dickey-Fuller test is primarily a left-tailed test, MacKinnon (2010, p. 5) concentrated on the critical values corresponding to the quantiles at levels $0.01, 0.05$ and 0.10[4] and provided the values of the estimated coefficients of the response surface regression at each of these levels in tabular form (MacKinnon, 2010, p. 13–15). The resulting critical values are similar to the simulated values from Dickey (1976, p. 53) and Fuller (1996, p. 642) but are stated to be more accurate by Davidson and MacKinnon (2009, p. 618).

Additionally, MacKinnon (1994, p. 172 – 175, 1996, p. 610 – 612) proposes an additional estimation for approximating p-value for a given test statistic t

$$\hat{p} = \Phi(\hat{\gamma_0} + \hat{\gamma_1} t + \hat{\gamma_2} t^2 + \cdots + \hat{\gamma_k} t^k), \tag{9}$$

where $\Phi(\cdot)$ is the standard normal distribution and $\hat{\gamma_k}, k = \{1, 2, \ldots, k\}$ are parameters estimated by regressing $\Phi^{-1}(p)$ on t. However, MacKinnon (1994, p. 173) also notes that the resulting estimated equations are not "satisfactory from a statistical point of view." Additionally, it is also observed that for $t < t_{min}$ and $t > t_{max}$, where t_{min} and t_{max} represent a large negative and positive value respectively, the approximations "attain spurious minima" and "spurious maxima" respectively and hence the use of the approximations beyond the t_{min} and t_{max} value is not recommended (MacKinnon, 1994, p. 173 - 174).

[2] Note that MacKinnon (1994, 1996, 2010) uses response surface regressions to approximate the (asymptotic) distribution functions for unit root as well as co-integration tests. Co-integration tests are not covered in this thesis.

[3] MacKinnon (1994) did not include at first the cubic term in the regression. In later publications, MacKinnon (1996, 2010) included it to account for additional data points being used.

[4] MacKinnon (1996, p. 604) also used more quantiles, 221 to be precise, starting from 0.0001 upto 0.9999. The use of larger number of quartile provide more information regarding the shape of the CDF of the test statistics.

Hansen (1997) builds on equation 9 and proposes a generalization:

$$p(t) = \Psi(\alpha_k(t|\gamma)|\eta) \tag{10}$$

where $\alpha_k(t|\gamma) = \gamma_0 + \gamma_1 t + \cdots + \gamma_d t^d$ is a polynomial of degreee d and $\Psi(\cdot|\eta)$ is a distribution function of interest that depends on an unknown parameter η. An example of $\Psi(\cdot|\eta)$ is $\chi^2(\eta)$ with η degrees of freedom, as used by Hansen (1997, p. 62) for structural-change test. Another interesting approach is taken by Bai and Carrion-I-Silvestre (2009) to analyze unit root along with structural changes and stochastic trends in the context of panel data, where the test of interest is the so-called modified Sargan–Bhargava test. In the absence of structural breaks, Bai and Carrion-I-Silvestre (2009, p. 485) implement a logit model, with the p-value being a function of the associated quantile values $q(p)$ and sample size n, i.e.

$$\ln\left(\frac{p_i}{1-p_i}\right) = g(q(p), n), \tag{11}$$

where $g(q(p), n)$ is specified as

$$g(q(p), n) = \sum_{j=0}^{1} \left(\zeta_{0j} + \zeta_{1j} q(p_i) + \zeta_{2j} q(p_i)^{-\frac{1}{2}} + \zeta_{3j} q(p_i)^{-\frac{1}{3}} + \zeta_{4j} q(p_i)^{-\frac{1}{4}}\right)\left(\frac{1}{T_i}\right)^j + \varepsilon_i,$$

where T is sample size and $q(p)$ are quantiles of test statistics for chosen p values. The coefficients are estimated using OLS and the approximate of p is derived by a simple transformation

$$\hat{p} = \frac{\exp[\hat{g}(q(p), n)]}{1 + \exp[\hat{g}(q(p), n)]}. \tag{12}$$

In the presence of structural breaks, Bai and Carrion-I-Silvestre (2009, p. 485) resorted to taking a similar approach as MacKinnon (1994, 1996) due to high non-linearity along the response surface.

3 Data Simulation

This section covers the details on the Monte Carlo simulations carried out to generate train and test dataset for training and evaluating the machine learning methods. The steps taken for the simulation follow the example by Enders (2014, pp. 202 – 206). In total three different train and test datasets have been generated, one for each of the three test types, i.e. without drift and trend, with only drift and with both drift and trend. The simulation consists mainly of three steps:

- Generate at first a random walk series of size n
- Depending on the test specification, use one of the regression from equations 2 to 4 to derive the OLS test or t-statistics under the null hypothesis H_0:$\gamma = 0$.
- Repeat the first two steps for $k = 200,000$ times

Hence, n determines the sample size or length of time series and for each n there are k t-statistic values. To ensure accuracy and proper conjecture regarding the limiting distribution, it is important to have sufficient large number of repetitions and different samples with increasing sizes (Dietrich, 2002, p. 5; Davidson and MacKinnon, 2009, p. 618). A total of 34 different values for n are used, with 20 being the smallest and 10,000 being the largest sample size. The rest has been chosen to uniformly cover the ranges $[20, 100]$, $(100, 500]$, $(500, 1000]$ and $(1000, 10000]$ as shown in Table 1. With 200,000 observations for each sample size, in total, each of the three training datasets contain 6.8 million data points. Since simulation at this scale is highly computationally intensive, they are carried out in AWS EC2 instances.

Table 1: Overview of training sample sizes.

Number range	sample size, n
$[20, 100]$	20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100
$(100, 500]$	150, 200, 250, 300, 350, 400, 450, 500
$(500, 1000]$	600, 700, 800, 900, 1000
$(1000, 10000]$	2500, 5000, 7000, 10000

For an impression of the shape of the distribution, it can be plotted for selected n values as in Figure 2. Similar to Figure 1, the figures also include the standard normal distribution as reference. From the figures, note that as n increases, the plot-lines start overlapping each other, implying a convergence of the distributions.

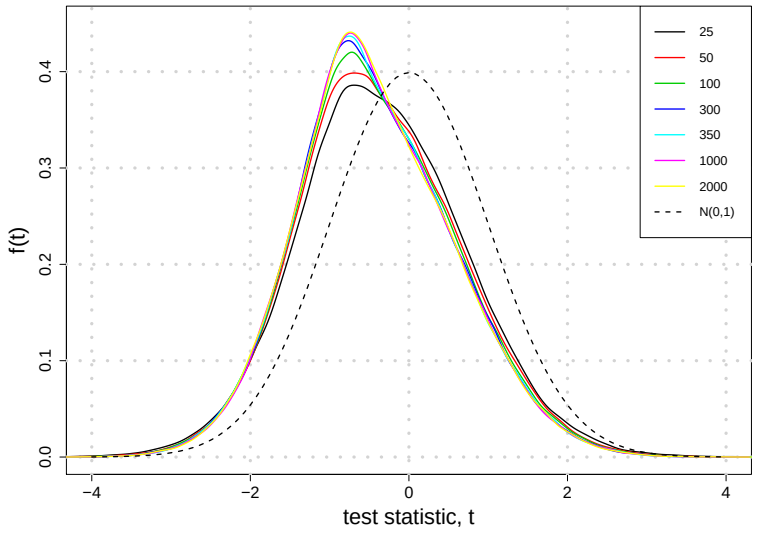
Finally, given the sequence of simulated test statistics $(t_1, \ldots, t_k)$, the empirical CDF for each sample size n is constructed using

$$\hat{F}_n(t) = \frac{1}{k} \sum_{i=1}^{k} \mathbf{1}\{t_i \leq t\},$$

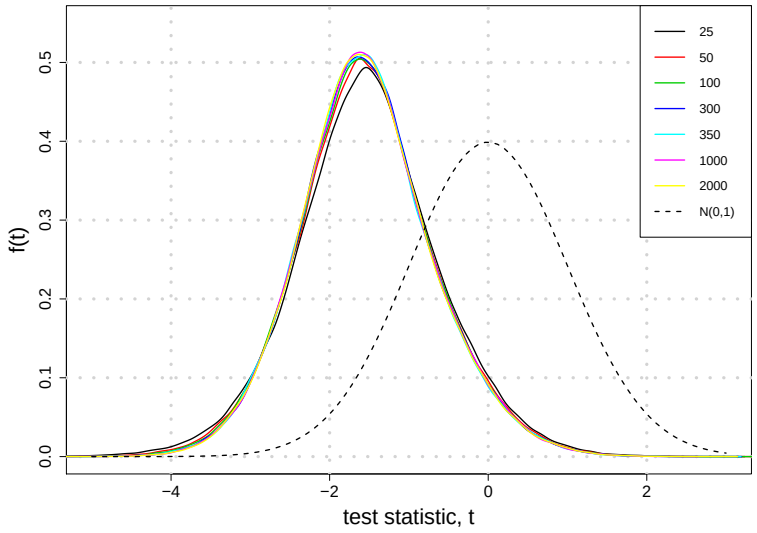
where $\mathbf{1}$ is an indicator function, i.e. $\mathbf{1}\{t_i \leq t\} = 1$ if $t_i < t$. It appears that by strong law of large numbers, as $k \to \infty$, the $\hat{F}_n(t)$ converges almost surely to the

Figure 2: Distributions of Dickey-Fuller test statistics for selected n.

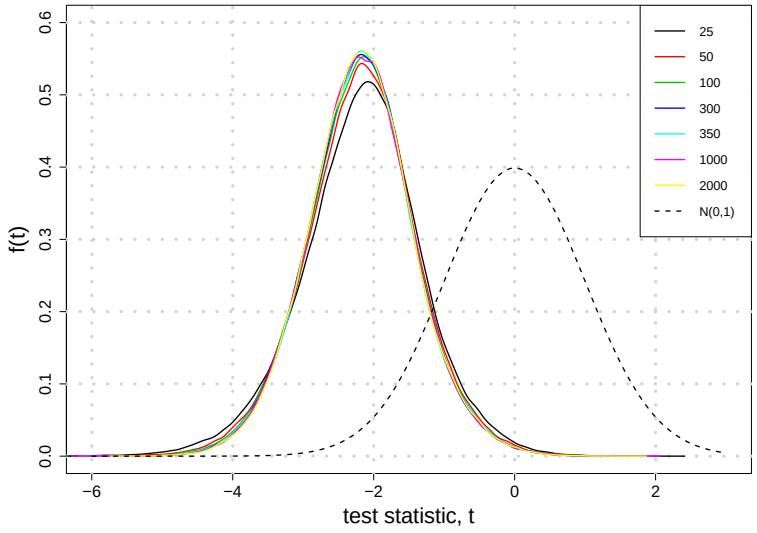
(a) Without drift and trend



(b) With drift



(c) With drift and trend



true cumulative distribution function $F_n(t)$ for every t, i.e. $\hat{F}_n(t) \overset{\text{a.s.}}{\to} F_n(t)\ \forall t$ (van der Vaart, 2000, p. 265). Hence $\hat{F}_n(t)$ is consistent and the empirical distribution function can be used to estimate the CDF of Dickey-Fuller distribution (van der Vaart, 2000, p. 265).

Although all the training datasets have been simulated using the above explained steps, for large n, an alternatively approach can be to use the limiting or asymptotic distribution definitions from expressions 5 to 7 from Section 2. However, there is no clear indication of when n is large enough. Based on the literatures, source code of various software implementations and further own simulations, it appears that the asymptotic definitions can be used for $n > 500$. Hanck (2016) provides examples that translates the expressions into `R`-code.

Finally, besides the training data, the simulation steps are repeated to generate the test datasets as well. Each of the test dataset consists of the same 15 new randomly chosen sample sizes n between 20 and 500, which do not appear in the training set. The test sample sizes are listed in Table 2.

Table 2: Overview of test sample sizes.

Number range	sample size, n
$[20, 500]$	27, 37, 44, 56, 58, 59, 64, 82, 88, 91, 187, 293, 329, 415, 452

4 Approach

In this section, the use of various machine learning methods for approximating the empirical CDF of Dickey-Fuller distribution as a function of both sample size, n, and test statistic, t, are explored. In other words, provided n and t, the approximated function will output the corresponding cumulative probability or percentile rank, which can be also used as a p-value to test for H_0 in unit root test. In total, three different empirical distributions will be approximated, one for each of the three test types or specifications: without drift and trend, with only drift and lastly, with both drift and trend.

Based on the common knowledge that a CDF has a "S" shaped curve and is bounded between 0 and 1, it is easy to note that a CDF's shape is closely resembled by a logistic function. In fact the use of logistic function for approximating distributions functions, like that of standard normal distribution, is not new and can be found in multiple literatures like Lin (1990), Bowling et al. (2009), and Bai and Carrion-I-Silvestre (2009).

Logistic models are often used in machine learning for classification tasks or to model binary dependent variables (see e.g. James et al., 2013, pp. 131–138; Kuhn, 2013, pp. 282–287; Bishop, 2006, pp. 203–206), but it can also be used when the dependent variable is continuous and bounded within an interval (Manning, 1996). For example, if

$$Y_i = \frac{1}{1 + \exp(-\boldsymbol{X}_i\boldsymbol{\beta} + \varepsilon_i)},$$

where $0 < Y_i < 1$, then simple transformations lead to the following regression

$$\ln\left(\frac{Y_i}{1 - Y_i}\right) = \boldsymbol{X}_i\boldsymbol{\beta} + \varepsilon_i.$$

Thus, if p is the percentile rank from the empirical distribution given n and t, simply approximate the regression[5]

$$\ln\left(\frac{p_i}{1 - p_i}\right) = f(n_i, t_i). \tag{13}$$

The problem hence boils down to finding a suitable functional form for $f(n_i, t_i)$. After the approximation, p can then be recalculated again through the simple transformation

$$\hat{p}_i = \frac{1}{1 + \exp(-\hat{f}(n_i, t_i))} = \frac{\exp(\hat{f}(n_i, t_i))}{1 + \exp(\hat{f}(t, n))}. \tag{14}$$

[5]This approach is similar to logistic model used by Bai and Carrion-I-Silvestre (2009) as stated in Section 2.

4.1 Polynomial Regression

Pearl and Reed (1922, p. 368) and Richards (1959, p. 290) note that the standard logistic function, which is defined as

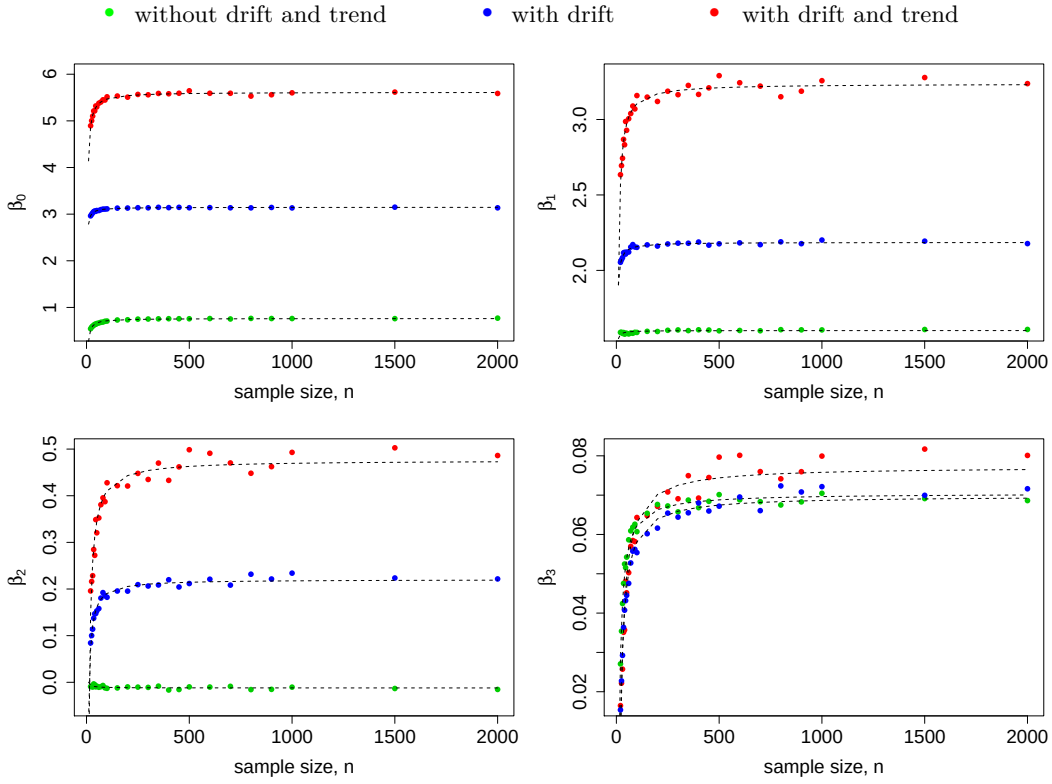
$$f(x) = \frac{L}{1 + \exp(-r(x - x_0))},$$

where L is the maximum value of the curve, r is a dimensionless variable that determines the curve steepness and x_0 is the inflection point, is restrictive in the sense that it requires the logistic curve to be symmetrical round the point of inflection. If L represents the upper limit or maximum value of the curve, then the the point of inflection always corresponds to $L/2$. Based on the shape of the distributions as seen in Figure 2, it is more likely that the CDF of Dickey-Fuller distribution is not symmetric, especially for finite sample sizes. One approach to account for asymmetry as proposed by Pearl and Reed (1922, p.368, 1925, p. 16) is to use a cubic function, such that

$$f_n(t) = \beta_0 + \beta_1 t_i + \beta_2 t_i^2 + \beta_3 t_i^3 + \varepsilon_i \tag{15}$$

for a fixed n. In order to incorporate n in the approximation, first note the magnitude of the parameters change as n increases. This is illustrated in Figure 3. Hence, it is possible to make the parameters $\beta_j, j = \{0, 1, 2, 3\}$ dependent on n. It appears that

Figure 3: Changes in value of parameters $\beta_j, j = \{0, 1, 2, 3\}$ as n increases. Source: Own illustration.



it is possible to fit a curve of the form

$$\beta_j = \gamma_{0,j} + \gamma_{1,j}\left(\frac{1}{n}\right) \tag{16}$$

for each parameter β_j, $j = \{0, 1, 2, 3\}$. Thus, equation 16 is substituted in 15 to yield

$$\begin{aligned} f(n_i, t_i) &= \left(\gamma_{0,0} + \gamma_{1,0}\frac{1}{n}\right) + \left(\gamma_{0,1} + \gamma_{1,1}\frac{1}{n}\right) t_i \\ &\quad + \left(\gamma_{0,2} + \gamma_{1,2}\frac{1}{n}\right) t_i^2 + \left(\gamma_{0,3} + \gamma_{1,3}\frac{1}{n}\right) t_i^3 + \varepsilon_i \\ \Rightarrow f(n_i, t_i) &= \gamma_{0,0} + \gamma_{0,1} t_i + \gamma_{0,2} t_i^2 + \gamma_{0,3} t_i^3 \\ &\quad + \gamma_{1,0}\left(\frac{1}{n}\right) + \gamma_{1,1}\left(\frac{t_i}{n}\right) + \gamma_{1,2}\left(\frac{t_i^2}{n}\right) + \gamma_{1,3}\left(\frac{t_i^3}{n}\right) + \varepsilon_i, \end{aligned} \tag{17}$$

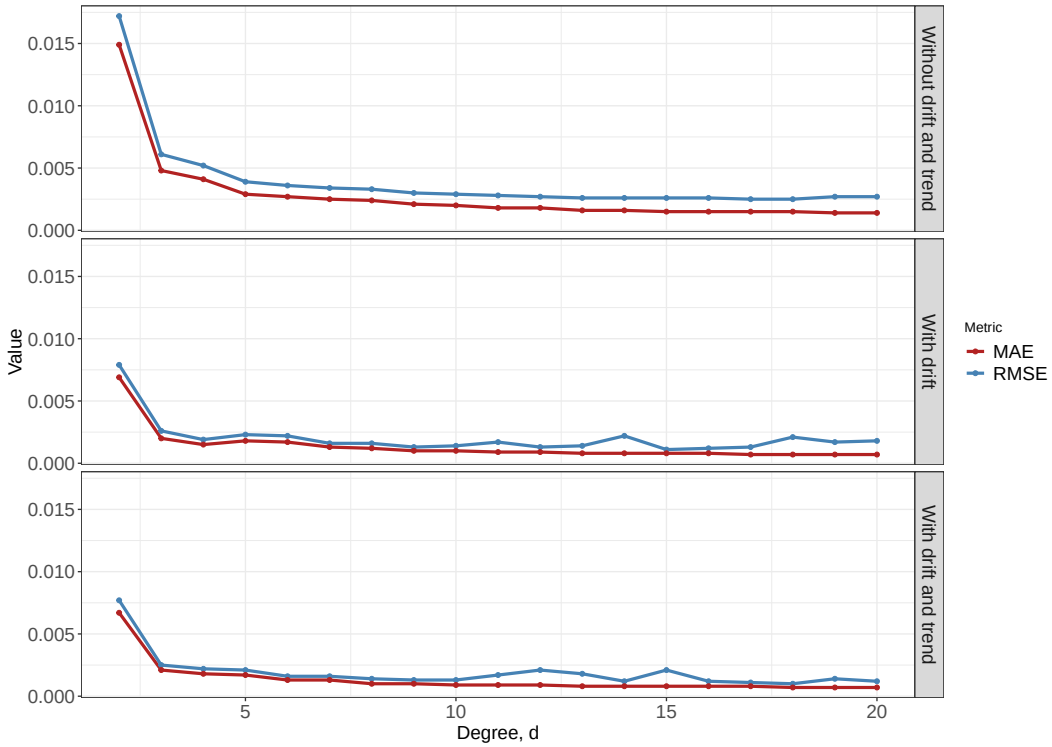
The parameters are then estimated using OLS. Based on equation 17, it is possible to derive more functional forms by using polynomials of higher degrees (d), removing interaction terms and using different expressions for n like natural logarithm or square root. Table A4 in Appendix B lists 36 such regression models.

The models are compared using five-fold cross-validated mean absolute error (MAE) and root mean square error (RMSE) values for the whole distribution as well as only for the lower or left tail (p-value < 0.2) of the distribution. The left tail of the distribution is additionally considered, since Dickey-Fuller is a left-tail test and in fact only the lower tail is of interest for statistical tests. The RMSE and MAE scores provide an initial indication of the accuracy of the models. Thus the smaller the scores, the more accurate is a model. The RMSE and MAE values for all different regression models are tabulated in Appendix B, Table A5 to A7. Based on the values, it can be deduced that increasing the polynomial degree in equation 17 leads to a reduction in both RMSE and MAE, while removing the interactions or using alternative expressions for n increases RMSE and MAE.

4.2 Lasso Regression

As noted above, for the polynomial regressions, increasing the polynomial degree d leads to a decrease in cross-validated RMSE and MAE. Based on Table A5 to A7 from Appendix B, where the maximum d is six, polynomial of degree six leads to the lowest RMSE and MAE in the absence of both drift and trend, a fourth degree polynomial has the lowest RMSE and MAE when drift is present and, finally, polynomial of degree six leads to the lowest scores in the presence of both drift and trend. Increasing the polynomial degree d further will likely decrease RMSE and MAE as well, as can be observed in Figure 4. However, the decrease for each additional d is small; furthermore, each additional d also adds two new parameters in the model and therefore also increases model complexity. For example, if $d = 18$, including intercept there are 38 parameters in the model, while for $d = 4$, the model

Figure 4: Cross-validated RMSE and MAE of polynomial regression with increasing polynomial degree. Source: Own illustration.



has only 10 parameters for slightly larger RMSE and MAE. At this point, it would be interesting to explore whether starting from a model with high polynomial degree and then using lasso regression to select only the terms with non-zero coefficients leads to a model with comparable or better accuracy. In other words, the variable selection property of lasso is utilized to create a penalized polynomial regression that is sparser than the full specification.

From James et al. (2013, p. 219–222), the lasso regression coefficients minimize

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

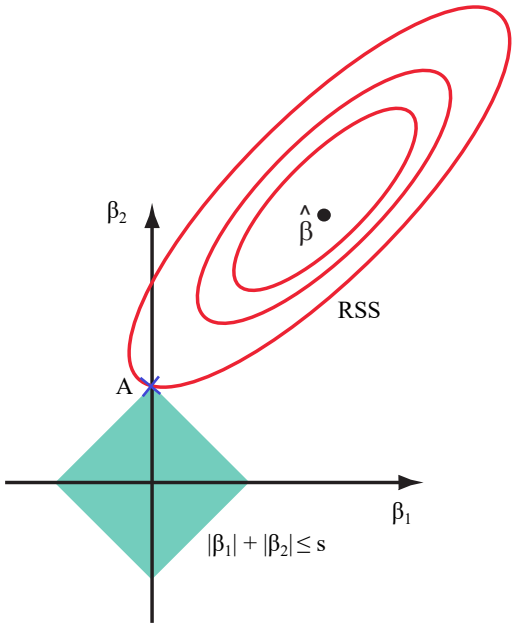
where the first part of the expression is the residual sum of square (RSS), $\sum|\beta_j|$ is the lasso or L1 penalty and λ is a tuning parameter. When $\lambda = 0$, the lasso simplifies to a standard least squares regression and when λ is sufficiently large, the lasso is equal to a null model with only an intercept. The above penalized least square problem can also be expressed as the optimization problem

$$\min_{\beta}\left\{\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2\right\} \text{ s.t. } \sum_{j=1}^{p}|\beta_j| \leq s$$

for some $s > 0$.

In lasso regression, the coefficients, except the intercept, are shrank towards zero and due to the L1 penalty, some of the coefficients are exactly zero. This can be

Figure 5: Variable selection using lasso for $p = 2$. Source: Adapted from James et al. (2013, p. 222).



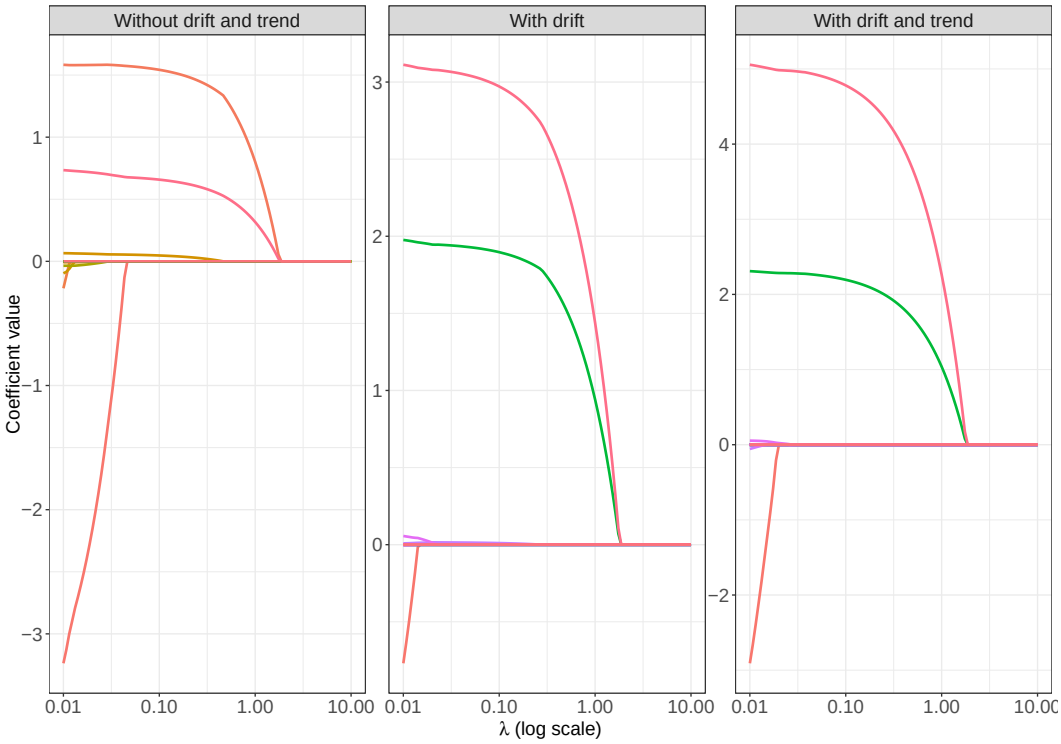
illustrated using Figure 5 where $p = 2$, i.e. there are only two parameters and hence two coefficients β_1 and β_2. The lasso regression coefficients are given when the RSS contour touches or intersects the constraint, as shown by point A in the figure. This intersection occurs in this example at β_2 axis due to the "shrap corners" of the lasso constraint. At this intersection, $\beta_1 = 0$ and thus the model only includes the variable corresponding to the coefficient β_2. In general, for higher dimensions with $p > 2$, the sharp corners of the lasso regression will likely lead to the intersection occurring at a point where multiple coefficients are simultaneously zero. Hence, through regularization or penalization, the lasso regression performs variable selection, resulting in sparser models than an unpenalized model.

The models for each of the three unit root test types are at first initialized by selecting $d = 20$, thus there are in total 41 coefficients excluding the intercept. Figure 6 shows the coefficients from the lasso regression as function of λ. When λ becomes sufficiently large all the coefficients are equal to zero. Thus, the appropriate value of λ can be found using cross-validation. The value of λ that results in the minimum MAE is chosen to perform the lasso regressions. The final penalized regression consist of only the terms with non-zero coefficients.

4.3 Generalized Additive Model (GAM)

One difficulty in using polynomial regression or most parametric approaches is determining the appropriate functional form. This can be avoided by taking a more non-parametric approach via introducing smooth terms like splines. In the following, generalized additive model (GAM) is introduced at first.

Figure 6: Lasso coefficients for the three types of unit root test. Source: Own illustration.



Given variables x_j and a linear model, say

$$Y_i = \beta_0 + \sum_{j=1}^{q} \beta_j x_j,$$

where β_0 is the intercept, GAM replaces the linear component $\sum_{j=1}^{q} \beta_j x_j$ with additive component $\sum_{j=1}^{q} f_j(x_j)$, i.e.

$$g(\mu_i) = \beta_0 + \sum_{j=1}^{q} f_j(x_j)$$

where $\mu_i \equiv \mathrm{E}(Y_i)$, $g(\cdot)$ is a known link function[6] and $f_j(\cdot)$ are so-called smooth functions (Hastie and Tibshirani, 1987, p. 371, 1995, p. 187; Wood, 2017, p. 161). A smooth function f_j itself can be expressed in terms more fundamental *basis* functions, or simply basis, $b_m(x)$, such that

$$f_j(x) = \sum_{m=1}^{M} b_m(x)\beta_m \tag{18}$$

is also linear (Wood, 2017, p. 162). In other words, each smooth function f_j is a linear combination of a number of basis functions $b(x)$ and the associated coefficients β. A simple example of use of basis functions can be to express a polynomial of degree

[6]For example, an identity link function will keep the values of the variable Y as it is.

three in terms of basis functions, $b_j, j = \{1, 2, 3\}$, such that

$$f(x) = \beta_1 + \sum_{m=1}^{3} \beta_{m+1} b_m(x) = \beta_1 + \beta_2 x + \beta_3 x^2 + \beta_4 x^3,$$

where $b_1(x) = x$, $b_2(x) = x^2$ and $b_3(x) = x^3$.

It is relatively trivial to incorporate GAM into our approximation (see for example, Hastie and Tibshirani, 1987, p. 372, 1995, p. 191–193). Define

$$f(n_i, t_i) = \beta_0 + s_t(t_i) + s_n(n_i) + s_{t,n}(n_i, t_i) + \varepsilon_i \tag{19}$$

where β_0 is intercept of the model, $s_t(t_i)$ and $s_n(n_i)$ are smooth functions of the variables t and n respectively and $s_{t,n}(n_i, t_i)$ is a *tensor* product smooth function that accounts for the interaction between t and n. Tensor product smooth, as an alternative to *isotropic* smooth, has the advantages that it allows to model interactions of variables with different scales or units by allowing different smooth parameters for each variable; it is scale invariant, i.e. it is not sensitive to linear scaling of a single variable, and it is independent of the order in which variables are considered for constructing the smooth function (Wood, 2017, pp. 227–229, 237–238).[7] The resulting generalized additive logistic model is thus

$$\ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + s_t(t_i) + s_n(n_i) + s_{t,n}(n_i, t_i) + \varepsilon_i. \tag{20}$$

Popular class of smooth functions are smoothing splines such as cubic spline, which determines the degree of smoothness of a curve not through the functional form but via smoothing parameters in combination with penalization terms and/or number of basis functions or knots, i.e. locations where the basis functions are joined.[8] The cubic smoothing spline minimizes the penalized residual sum of squares

$$\sum_{i=1}^{n} \{Y_i - f(x_i)\}^2 + \lambda \int_a^b f''(x)^2 dx, \tag{21}$$

where λ is the smoothing parameter and $a < x_1 < \cdots < x_n < b$ (Hastie and Tibshirani, 1990, p. 27; Wood, 2017, p. 198). The resulting curve is continuous at the knot positions up to and including the second derivative (James et al., 2013, p. 271–273). The second term in the above expression is the penalty term. As $\lambda \to 0$, the penalty term will have little effect and the curve will be extremely "wiggly", i.e. vary rapidly. If $\lambda \to \infty$, it will be essentially same as a linear line (James et al., 2013, p. 278).

It is possible to control the smoothness of the fit of the GAM model by varying the smoothing parameter and/or the number of knots and hence basis functions. Appropriate value for the smoothing parameter is selected using restricted maximum

[7] Details regarding tensor product bases can be found in Wood (2017, pp. 227–231).

[8] An introduction to splines can be found in, for example, James et al. (2013, pp. 270–280).

likelihood (REML) (Wood, 2017, p. 182). The number of knots is determined via a grid search approach with cross-validated RMSE and MAE as metrics. Lastly, the knots are placed evenly along the range of the variables t and n.

Although computational efficiency can be increased by using penalized regression splines or eigen-decomposition (Wood, 2017, p. 199–201, 217–218), training GAM models using the full training datasets with each containing 6.8 million data points is highly computational memory intensive. One alternative as developed by Wood, Goude, et al. (2015) and Wood, Li, et al. (2017) provides methods to efficiently apply GAM to large datasets. Another alternative is to reduce the size of the datasets by keeping only data points that correspond to selected quantiles, for example: $0.0001, 0.0002 \ldots, 0.009, 0.01, 0.02, 0.03, \ldots, 0.97, 0.98, 0.99, 0.9901, \ldots, 0.9999$. This is somewhat similar to the steps taken by MacKinnon (1994, p. 169, 1996, p. 604), where the authors estimated and stored data for only selected quantiles. Doing so reduces the data size from several million to only several thousands data points. I decide for the second option, solely due to the fact that each of the models trained using the reduced size datasets occupy about or slightly more than 1.5 MB of computer disk space, making them portable, in the sense, that the models can be incorporated into packages or libraries.

4.4 Multivariate Adaptive Regression Splines (MARS)

Similar to GAM, another form of model that utilizes knots and basis functions is the multivariate adaptive regression splines (MARS). As detailed in Faraway (2016, p. 337) and Hastie, Tibshirani, and Friedman (2009, p.321–3222), MARS builds a model of the form

$$\hat{f}(x) = \sum_{j=1}^{k} c_j B_j(x) \tag{22}$$

where $B_j(x)$ are products of linear piecewise basis functions such that for a given knot t

$$(x-t)_+ = \begin{cases} x-t, & \text{if } x > t, \\ 0, & \text{otherwise,} \end{cases} \qquad \text{and} \qquad (t-x)_+ = \begin{cases} t-x, & \text{if } x < t, \\ 0, & \text{otherwise.} \end{cases}$$

In comparison to GAMS, MARS takes an iterative approach by repeatedly searching for suitable knots. The smoothness of the curve is similarly determined by the number of basis functions, where the optimal number of basis functions and hence also knots can be determined using cross-validation. More details on placement and selection of the number of knots can be found in Hastie, Tibshirani, and Friedman (2009, p. 321–329) and Faraway (2016, p. 337–339). MARS allows to build interaction between the multiple variables in the data and thus is more suitable for high-dimensional data (Hastie, Tibshirani, and Friedman, 2009, p. 321), which is

clearly not the case in this thesis. Nevertheless, it is still interesting to see how well it performs in approximating distribution functions. Incorporating MARS in the approximation simply accounts to fitting $f(n_i, t_i)$ with the MARS algorithm.

4.5 Richards' Curve

Richards' curve or also known as generalized logistic curve is, as the name implies, a generalization of logistic function. Recall that, given a standard logistic function

$$f(x) = \frac{L}{1 + \exp(-r(x - x_0))}$$

it is possible to account for asymmetry by using cubic polynomials (Pearl and Reed, 1922, p.368, 1925, p. 16). Richards' curve accounts for asymmetry instead by introducing an additional new variable v (Szparaga and Kocira, 2018, p. 3; Archontoulis and Miguez, 2014, p. 788; Yin et al., 2003, p. 362; Cavallini, 1993, p. 249), such that

$$f(x) = \frac{L}{[1 + v \cdot \exp(-r(x - x_0))]^{1/v}}. \tag{23}$$

Note the relation to standard logistic function by observing that setting $L = 1, x_0 = 0, r = 1$ and $v = 1$ gives back the standard logistic function.

Richards' curve is more commonly used to study, for example, population growth. But its use can be adapted to approximate the CDF for a fixed sample size n using

$$p_{i,n} = \frac{1}{[1 + v \cdot \exp(-r(t_i - t_0))]^{1/v}} + \varepsilon_i. \tag{24}$$

Similar to the parameters $\beta_j, j = \{0, 1, 2, 3\}$ in polynomial regression 17, it appears that the value of v and t_0 also depend on n and change in a similar fashion. Thus incorporating n in regression 24 gives us

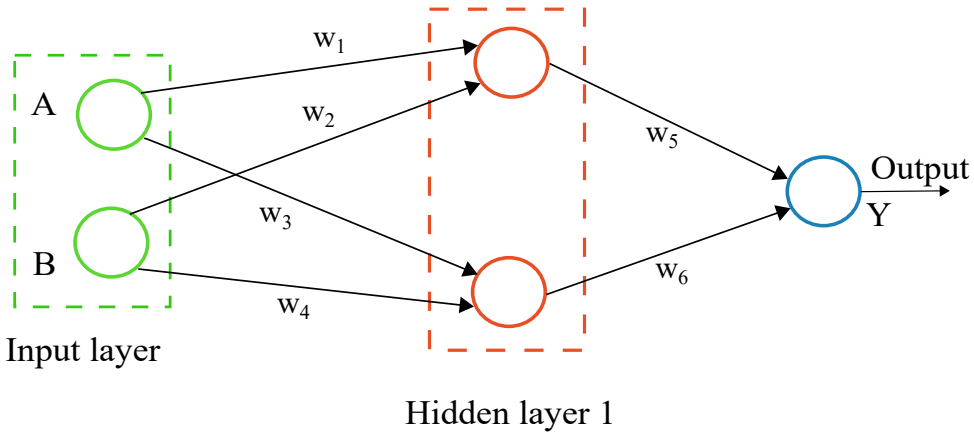
$$p_i = \frac{1}{\left[1 + (\gamma_0 + \gamma_1 \cdot \frac{1}{n}) \cdot \exp(-r(t_i - (\gamma_2 + \gamma_2 \cdot \frac{1}{n})))\right]^{1/\left(\gamma_0 + \gamma_1 \cdot \frac{1}{n}\right)}} + \varepsilon_i. \tag{25}$$

Regression 25 is, however, nonlinear in parameters unlike previous approaches, and the parameters are estimated using nonlinear least-square algorithm. Also note that unlike the above mentioned approaches, the dependent variable is directly p and not its logit transformation.

4.6 Feedforward Neural Networks (NN)

The use of NN for approximating the empirical distribution is motivated by the *universal approximation theorem*, based on Cybenko (1989), Hornik et al. (1989), and Hornik (1991). Before detailing the model specifications, at first, the various terms used in association with neural networks (NN) are introduced. For a detailed explanation of the concepts, see for example Goodfellow et al. (2016).

Figure 7: A simplified illustration of a feedforward neural network with two hidden layers. Source: Own illustration.



A neural network like multilayer perceptron (MLP) consists of three types of layer, namely input, hidden and output layer, and the information flows within the network only in one direction, i.e. from the input to output layer via the one or more hidden layers. Figure 7 shows a diagram of a simplified neural network with a input layer consisting of two nodes for the two variables A and B, one hidden layer with two hidden units or nodes and finally the output layer with a single node. To each of the edges in the network or lines connecting an input to a hidden layer node, a weight w is attached. The weight signifies how strongly a input node affects a connecting hidden node. In NN, the weights are the parameters of the model and the NN model is fitted, by adjusting the weights. Finally, in each of the nodes in hidden layer, an *activation function*, like logistic or rectified linear unit (ReLU), transforms the input before passing it to the next node. A detailed overview of different activation functions can be found in Goodfellow et al. (2016, p. 191–197).

Mathematically, the output of a MLP with a single hidden layer and a single output node like the one in Figure 7 can be formulated as (Günther and Fritsch, 2010, p. 31)

$$o(\boldsymbol{x}) = f\left(w_0 + \sum_{j=1}^{J} w_j \cdot f\left(w_{0j} + \boldsymbol{w_j}^T \boldsymbol{x}\right)\right)$$

where w_0 and w_{0j} are the intercept of the output node and the j-th hidden node respectively and f is an activation function. The parameter vector $\boldsymbol{w_j}$ corresponds the weights at the edges connecting each node and $\boldsymbol{x}$ is the input vector. Through (resilient) backpropagation mechanism, the weights can be adjusted via gradient descent, in order to minimize a loss function like MSE for regression or cross-entropy for binary classification (see e.g. Günther and Fritsch, 2010, p. 32–33; Hastie, Tibshirani, and Friedman, 2009, pp. 395–397).

Returning to the universal approximation theorem, in essence, the theorem states that standard MLP with as few as one hidden layer and sufficient finite number of hidden units or node can approximate any continuous function arbitrarily well (Goodfellow et al., 2016, p. 198; Günther and Fritsch, 2010, p. 31). The theorem has been proved for various classes of activation functions (Goodfellow et al., 2016, p.198). However, Goodfellow et al. (2016, p. 199) also note that the theorem does not state how large or how many hidden nodes each units each layer should have. It is possible that a deeper network may be more feasible than a network with a single layer consisting of large number of hidden nodes.

Fitting a neural network involves additionally determining so-called hyperparameters besides the weights. A non-exhaustive list of such parameters include:

- the number of (hidden) layers in the network
- number of nodes in each layer
- activation functions in the hidden layers
- activation function in the output layer
- optimizers
- batch size
- learning rates of the optimizers

While optimizers, batch size and learning rate can affect convergence, the model capacity can be influenced by varying number of layers and nodes and the model output range via the activation functions. The higher the number of nodes and layer, the larger is the model capacity.

The NNs are fitted using the Keras framework with TensorFlow as backend. For each of the test types, the hyperparameters number of layers, number of nodes and activation function for the hidden and output layer have been tuned separately. The combination of parameters that lead to the lowest MAE on validation set is chosen for training the NN. It appears that using multiple hidden layers and sigmoid activation function in the hidden and output layers lead to better performance. An advantage of sigmoid function is that it bounds the output between 0 and 1. Hence, it is possible to directly approximate p without transforming it at first as in equation 13.

Depending of the size of the network and training sample, training NNs can be highly computationally intensive. To speed up the parameter tuning and training process, GPUs are utilized via CUDA from NVIDIA.

5 Model Evaluation

In the previous section, six different approaches have been introduced. In this section, the performance of these six approaches are compared against each other as well as against the MacKinnon's p-value approximation and linear interpolation based on the empirical percentile values tables from Dickey and Fuller (see Appendix A).

Test RMSE and MAE

First the RMSE and MAE are compared based on the test dataset. However, the linear interpolation approach cannot interpolate when the sample size n and/or t-statistics lie outside the minimum and maximum n and/or t values in the empirical percentile values tables. Therefore, in order to ensure that the six models along with the linear interpolation and MacKinnon's approximation are evaluated using the same data, a subset of the original test datasets have been used to compute the RMSE and MAE values. The results have been tabulated in Table 3.

The RMSE and MAE values provide a first indication of the accuracy of the models; thus the lower the values, the higher is the accuracy of the models. Both of the two metrics have been calculated in total four times, as indicated by the four column groups in the table. As the header label 'Full distribution' suggests, the first column group covers the whole distribution. For the unit root test, accuracy at the lower or left tail is more important; thus this is accounted for separately by considering only the p-values less than 0.2, i.e. $p < 0.2$. The corresponding RMSE and MAE values are presented in the second column group with the label '$p < 0.2$'. Similarly, the distribution is further divided to separately consider the middle part ($p \in [0.20, 0.80]$) and upper tail ($p > 0.8$) of the distribution. The corresponding RMSE and MAE values are presented in the last two column groups respectively. Finally, the shaded cells indicate the lowest RMSE and MAE among the six models, test type and column wise.

Based on the values in Table 3, I discard the neural network and MARS models for further assessment, due to their lower accuracy than the rest. The lasso regression models will also be dropped in favor of the unpenalized polynomial regressions with $d = \{3, 4, 5, 6\}$, as the lasso models have relatively lower accuracy and often still more parameters than the polynomial regressions considered in the table. Among the remaining, GAM models have in general the lowest RMSE and MAE values when the test is specified both without and with drift and trend. When the test is specified with only drift, there is no single model that consistently has the lowest scores across the whole distribution or each of the three regions. Interesting is also the comparison of the RMSE and MAE values against the linear interpolation approach and MacKinnon's p-value approximation. Although the linear interpolation is simple and easy to implement, it performs only better than the neural network models; in other words, it has the second lowest accuracy among all. MacKinnon's approximation of the p-values are on a par with the GAM models and even attain

slightly lower RMSE and MAE values than GAM models in the presence of drift and/or trend. Nevertheless, the differences in the RMSE and MAE values among all the models, except the three discarded ones, and MacKinnon's approximation are small.

Table 3: RMSE and MAE based on test dataset. The shaded cells indicate the lowest RMSE and MAE among the six models, test type and column-wise.

	Full distribution		$p < 0.2$		$p \in [0.20, 0.80]$		$p > 0.8$	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Without drift and trend								
Poly ($d = 3$)	0.00577	0.00473	0.00268	0.00243	0.00681	0.00586	0.00422	0.00343
Poly ($d = 4$)	0.00508	0.00410	0.00157	0.00136	0.00605	0.00530	0.00383	0.00302
Poly ($d = 5$)	0.00330	0.00263	0.00206	0.00177	0.00389	0.00327	0.00201	0.00147
Poly ($d = 6$)	0.00310	0.00244	0.00158	0.00134	0.00372	0.00313	0.00184	0.00134
Lasso	0.00660	0.00519	0.00418	0.00366	0.00769	0.00624	0.00449	0.00341
GAM	0.00296	0.00236	0.00084	0.00063	0.00356	0.00311	0.00210	0.00174
MARS	0.01229	0.00958	0.00984	0.00767	0.01391	0.01131	0.00840	0.00601
Richards	0.00548	0.00427	0.00222	0.00194	0.00494	0.00413	0.00857	0.00706
NN	0.05789	0.04837	0.07225	0.06247	0.06058	0.05336	0.02141	0.01832
LInt	0.02992	0.02331	0.01278	0.00915	0.03589	0.03099	0.01957	0.01314
McK	0.01201	0.00817	0.00092	0.00069	0.01460	0.01109	0.00832	0.00640
With drift								
Poly ($d = 3$)	0.00276	0.00214	0.00099	0.00077	0.00325	0.00269	0.00221	0.00179
Poly ($d = 4$)	0.00203	0.00160	0.00071	0.00056	0.00243	0.00207	0.00150	0.00117
Poly ($d = 5$)	0.00244	0.00196	0.00092	0.00073	0.00297	0.00263	0.00142	0.00110
Poly ($d = 6$)	0.00236	0.00190	0.00090	0.00074	0.00290	0.00258	0.00123	0.00093
Lasso	0.00307	0.00250	0.00133	0.00099	0.00281	0.00241	0.00471	0.00428
GAM	0.00183	0.00144	0.00147	0.00124	0.00199	0.00159	0.00160	0.00118
MARS	0.00592	0.00475	0.00398	0.00325	0.00581	0.00475	0.00761	0.00628
Richards	0.00158	0.00122	0.00124	0.00100	0.00131	0.00107	0.00241	0.00193
NN	0.05166	0.03780	0.07017	0.04433	0.05107	0.04107	0.02449	0.02094
LInt	0.03604	0.02881	0.01837	0.01334	0.04152	0.03605	0.03032	0.02146
McK	0.00099	0.00077	0.00055	0.00041	0.00115	0.00095	0.00074	0.00059
With drift and trend								
Poly ($d = 3$)	0.00233	0.00195	0.00190	0.00165	0.00255	0.00216	0.00196	0.00160
Poly ($d = 4$)	0.00206	0.00172	0.00173	0.00147	0.00222	0.00188	0.00183	0.00145
Poly ($d = 5$)	0.00204	0.00169	0.00132	0.00112	0.00216	0.00183	0.00224	0.00182
Poly ($d = 6$)	0.00153	0.00123	0.00096	0.00076	0.00158	0.00132	0.00180	0.00142
Lasso	0.00431	0.00373	0.00422	0.00354	0.00478	0.00433	0.00238	0.00202
GAM	0.00105	0.00079	0.00069	0.00055	0.00118	0.00090	0.00093	0.00070
MARS	0.00818	0.00633	0.00826	0.00675	0.00880	0.00688	0.00571	0.00415
Richards	0.08012	0.06894	0.09985	0.09368	0.06390	0.05348	0.10118	0.09294
NN	0.10321	0.08540	0.16524	0.14674	0.08992	0.08231	0.04593	0.03386
LInt	0.03099	0.02572	0.02337	0.01632	0.03494	0.03136	0.02353	0.01730
McK	0.00086	0.00067	0.00062	0.00049	0.00094	0.00074	0.00080	0.00065

Keys: Poly: Polynomial regression, Lasso: Lasso regression, Richards: Richards' curve, NN: Neural network, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Proportion of Rejections of Null Hypothesis

Next, the proportion of rejection of the H_0 by the models are compared via a simulation. MacKinnon's approximation and the linear interpolation approach are included as well, but, as already mentioned, neural networks, MARS and lasso will be excluded due to their relatively high RMSE and MAE values. The share of rejection is compared at different combinations of sample size n and significance level α from the set $n \in \{27, 57, 130, 265, 1150\}$[9] and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. The simulation steps are outlined below:

1. Select a combination of n and α
2. Generate a series of size n based on an AR(1) process

$$Y_t = \rho Y_{t-1} + \varepsilon_t$$

   for a selected value of ρ
3. Test for unit root and note whether the p-value from the test is less than α or not
4. Repeat step 2 and 3 for 10,000 times and calculate the proportion of rejections. For example, if the H_0 that there is unit root, is rejected only 7,600 times out of 10,000 repetitions, the proportion of rejection is 0.76 or 76%
5. Vary the value of ρ from 0 to 1 and repeat steps 2 to 4, in order to obtain the share of rejection at a new ρ value
6. Repeat the above steps for a new combination of (n, α)

The simulation steps are carried out for each of the three test specifications.

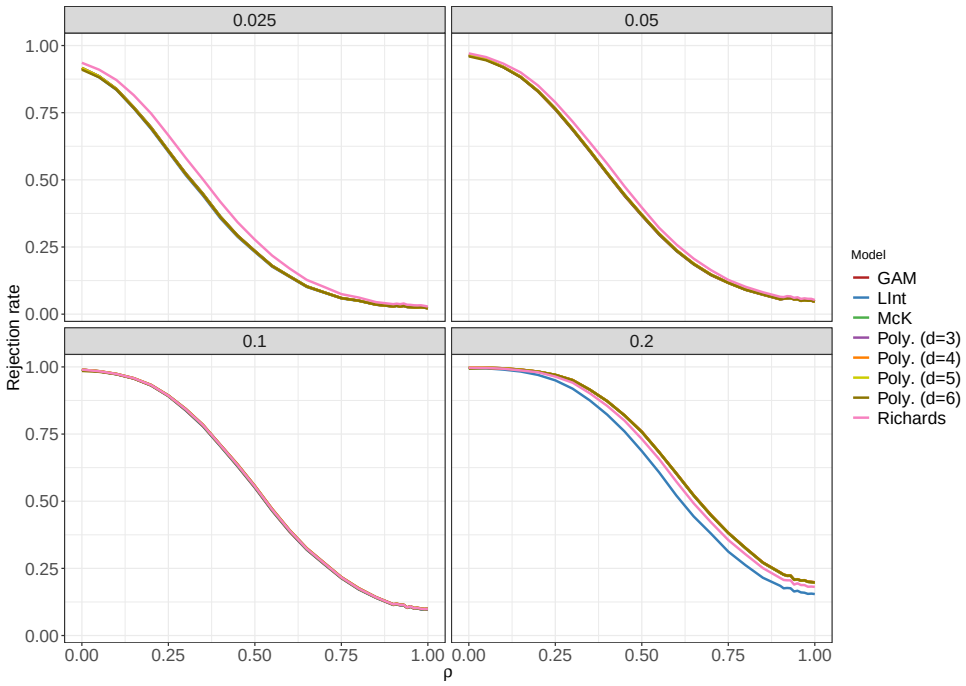
The results for unit root test with drift and trend are shown in Figure 8 to 12. Starting off with Figure 8, where the sample size is $n = 27$, except Richards' curve and the linear interpolation, the models have similar rejection proportion. Model based on Richards' curve has initially a higher share of rejection at $\alpha = 0.025$, similar at $\alpha = 0.1$ and than smaller at $\alpha = 0.2$. However, as seen from Figure 9 to 12, the Richards' curve systematically under-rejects and in the presence of unit roots, i.e. $\rho = 1$, the share of rejections is smaller than the significance level. On the other hand, the linear interpolation approach has a smaller share of rejection than the other models only at $\alpha = 0.2$. This behavior of the linear interpolation is, however, not unexpected as it is based on empirical percentile values tables, which includes the percentiles for only selected sample sizes and significance level. In the tables more emphasis is given to the lower ($p \leq 0.10$) and upper ($p \geq 0.90$) tail of the distribution in comparison to the middle part; hence the approximation of the p-values is better at the tails. Based on Figure 9 to 12, this deviation of the linear interpolation from the rest of the models becomes less prominent or diminishes as the sample size increases. When n is still relatively small as in Figure 9, the polynomial model of degree six considerably starts to deviate from the rest and shows smaller

[9] This set of n values do not appear in the training datasets as well.

proportion of rejection for $\rho < 0.45$. In fact, this deviation becomes larger as n increases. For example, when $n = 1150$, as in Figure 12, the model barely rejects any H_0 for even $\rho < 0.8$. This characteristics of wrongly failing to reject the H_0 is also shared by the polynomial model of degree five, as evident in Figure 10, where $n = 130$.

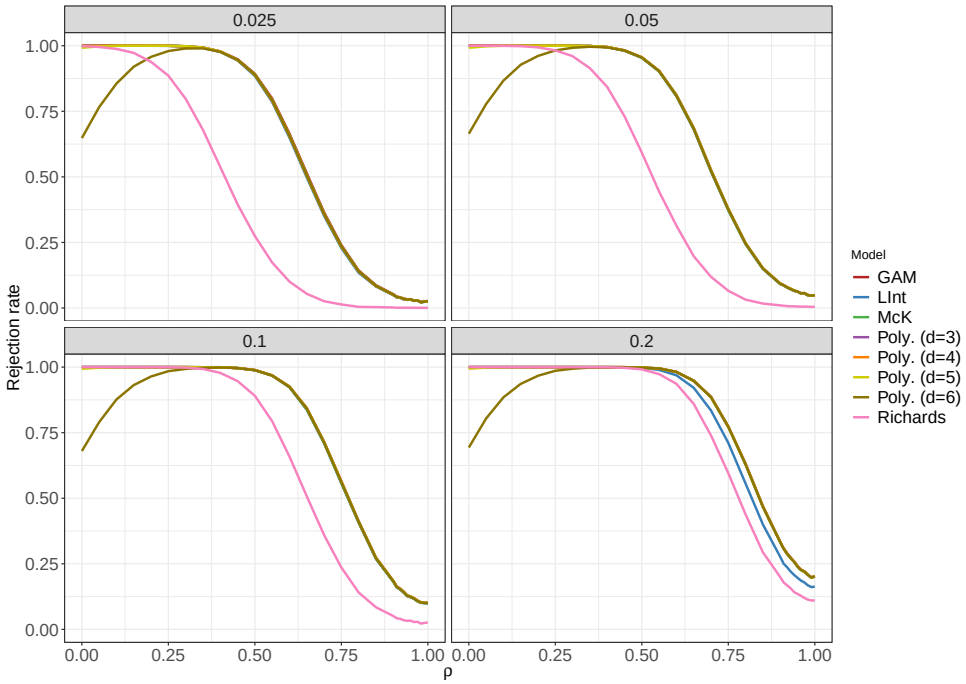
The results for the other two types of test specifications have been included in Appendix C. Similar behavior of the polynomial models and linear interpolation approach are observed. Richards' curve surprisingly has better performance in the absence of linear trends and behaves similarly as to other models.

Figure 8: Share of rejection from 10,000 repetition of unit root test with drift and trend, $n = 27$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



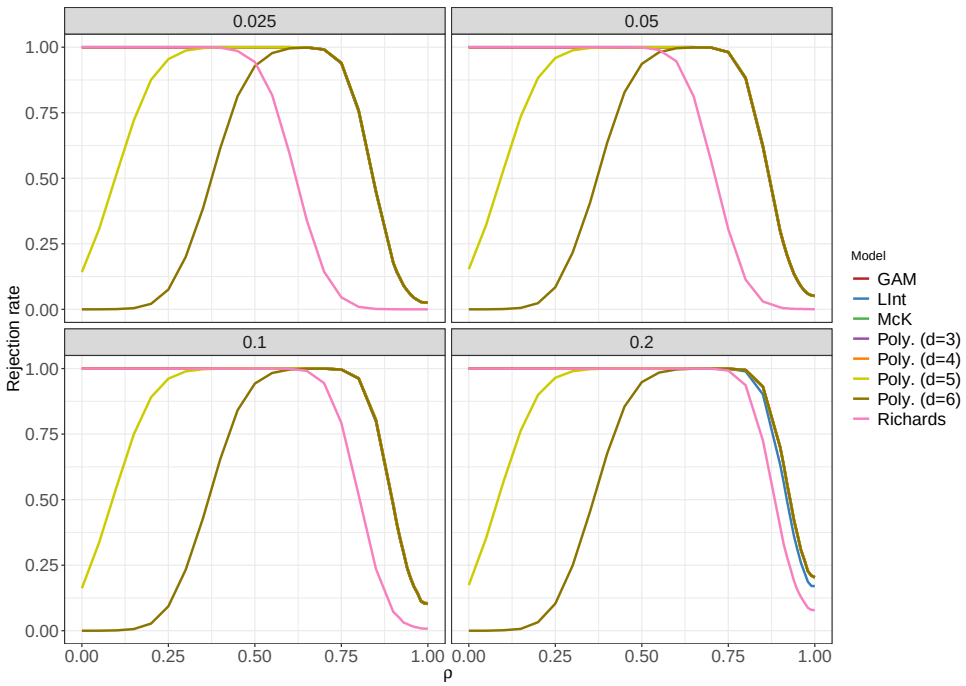
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure 9: Share of rejection from 10,000 repetition of unit root test with drift and trend, $n = 57$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



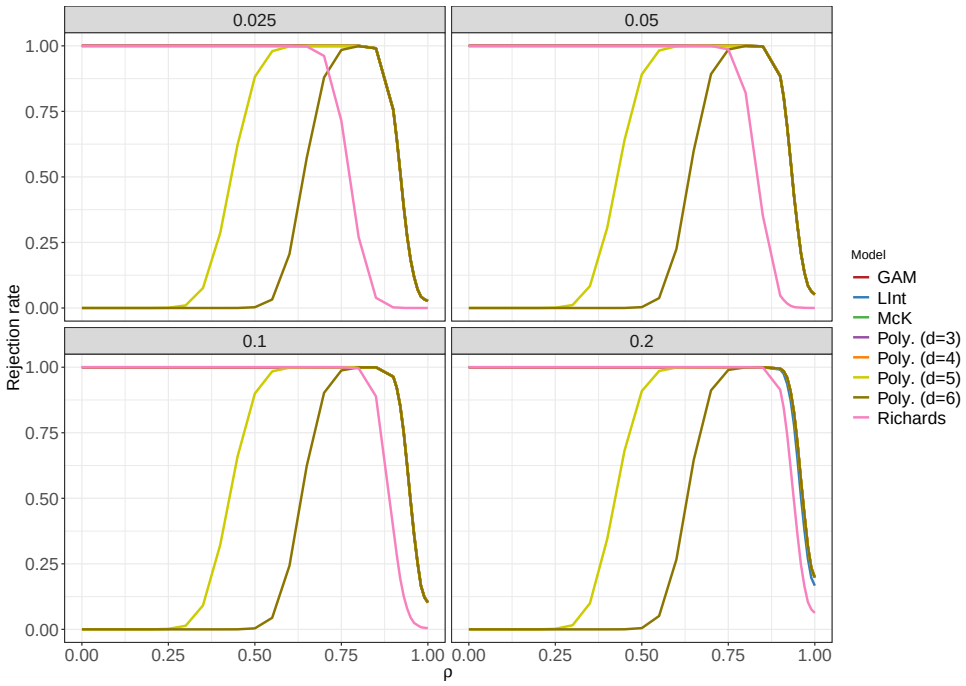
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure 10: Share of rejection from 10,000 repetition of unit root test with drift and trend, $n = 130$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



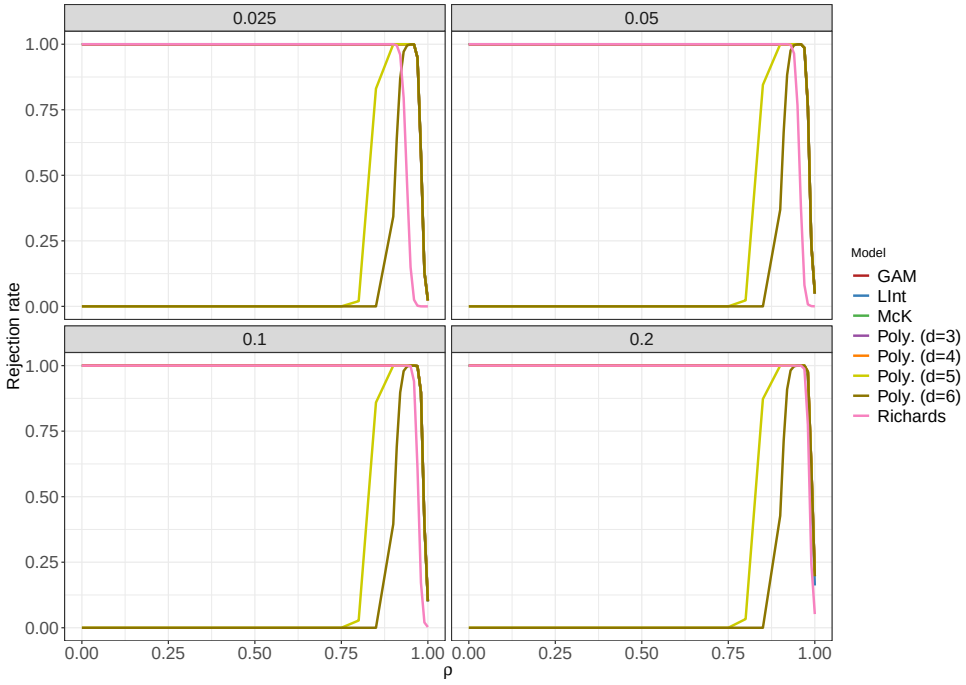
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure 11: Share of rejection from 10,000 repetition of unit root test with drift and trend, $n = 265$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, NN: Neural network, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure 12: Share of rejection from 10,000 repetition of unit root test with drift and trend, $n = 1150$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



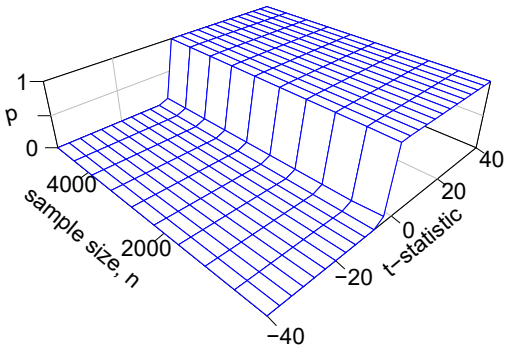
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Plot of Model Fit

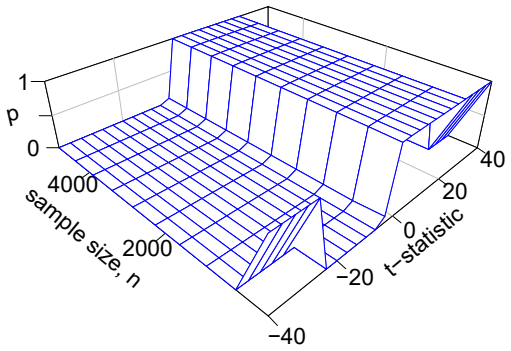
The fact that some of the polynomial based models reject too few even when the simulated time series data is stationary indicates that the approximated p-values from the models are larger than it should be. While this is not reflected in the RMSE and MAE values in Table 3, plotting the model fit can help to understand it better. The plots in Figure 13 show the model fit for the unit root test specified with drift and trend.[10] It appears that, attempting to fit globally flexible polynomial models has resulted in model that are non-decreasing only within a restricted range but not across the entire range of t or n. This is undesirable, as $F_n(t) \nrightarrow 0$ as $t \to -\infty$ or $F_n(t) \nrightarrow 1$ as $t \to \infty$. As a consequence, outside the aforementioned

Figure 13: Model fit for unit test with drift and trend. Source: Own illustration.

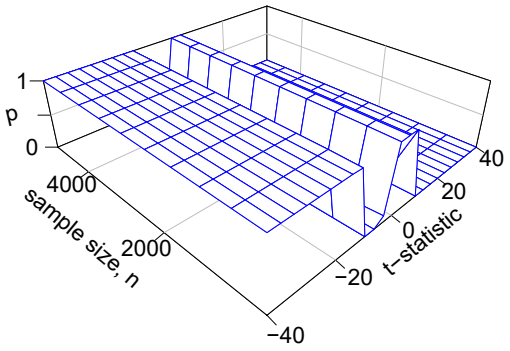
(a) Polynomial ($d = 3$)



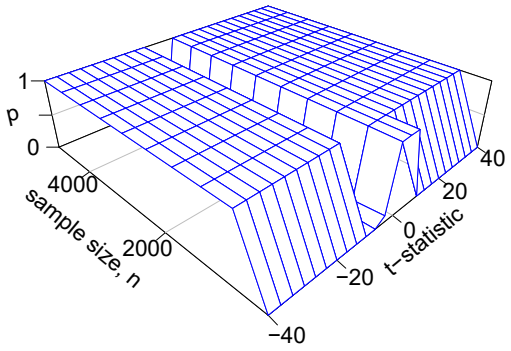
(b) Polynomial ($d = 4$)



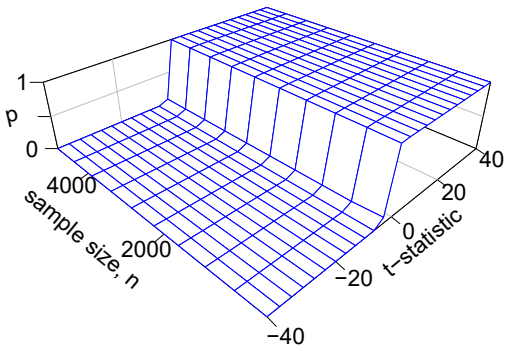
(c) Polynomial ($d = 5$)



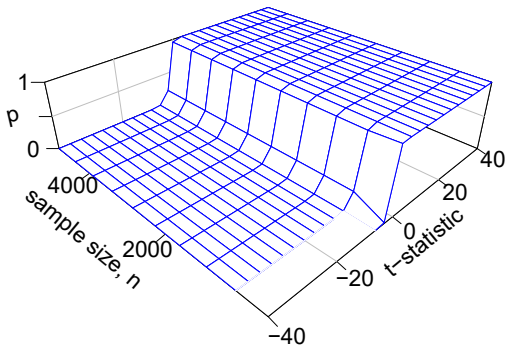
(d) Polynomial ($d = 6$)



(e) GAM



(f) Richard's curve



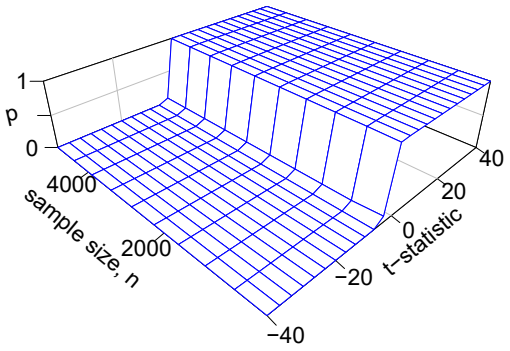
[10] For the sake of clarity, the plots have been created with low grid "resolution", hence the sharp turns and straight lines instead of curves.

range, the model will approximate too high or low p-values and when used within context of hypothesis testing, it will lead to wrong decision, as already observed. In comparison, the GAM model does not show this behavior. While the figures show the fit for only $t \in [-40, 40]$ and and $n \in [20, 5000]$, the same is observed for larger $|t|$ and n. The model fit for other two types of unit root test have been included in Appendix D (Figure A13 and A14), where again similar observations are made.

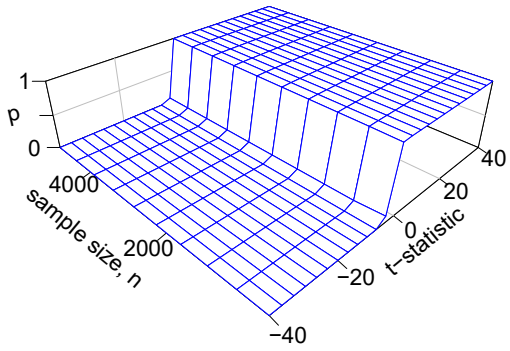
One simply way to remedy this issue can be to determine a $t_{\min}$ and $t_{\max}$, such that the polynomial model is non-decreasing in the range $[t_{\min}, t_{\max}]$. Then, for any t-statistic that is $t < t_{\min}$ or $t > t_{\max}$, the corresponding $t_{\min}$ or $t_{\max}$ will be used instead as a proxy to approximate the p-value. The use of such limits is not unwarranted. Every CDF is non-decreasing (Park, 2018, pp. 78–81) and the value of $t_{\min}$ and $t_{\max}$ can be derived from simulated training dataset. As an example, consider again the unit root test without drift and trend. The average minimum and maximum value of t-statistic, rounded to 2 decimal places, in the training dataset across all sample size is -6.30 and 1.88. These values can be set as $t_{\min}$ and $t_{\max}$, i.e. if $t < -6.30$, then instead of the original t-statistic, $t_{\min} = -6.30$ will be passed onto the polynomial model to calculate the p-value. Similar steps will be undertaken if $t > t_{\max} = 1.88$. But, if $-6.30 < t < 1.88$, then the original t-value will be used. Using the limits, thus ensures that the polynomial models are non-decreasing across the whole range of t. The limits do not change the RMSE and MAE values from

Figure 14: Model fit of polynomial models for unit test with drift and trend after setting minimum and maximum limits. Source: Own illustration.

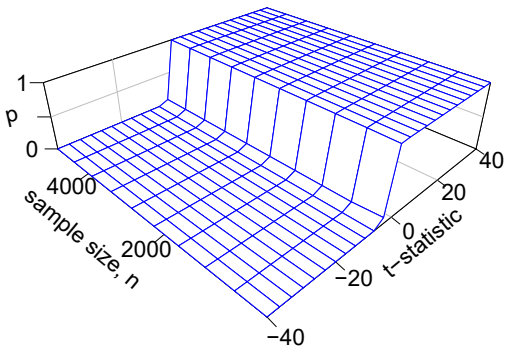
(a) Polynomial ($d = 3$)



(b) Polynomial ($d = 4$)



(c) Polynomial ($d = 5$)



(d) Polynomial ($d = 6$)

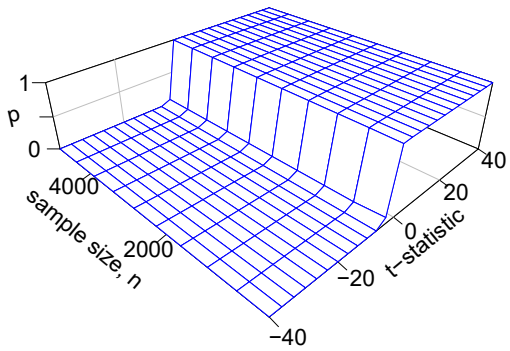


Figure 15: Share of rejection of polynomial models for unit root test with drift and trend after setting minimum and maximum limits, $n = 1150$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.

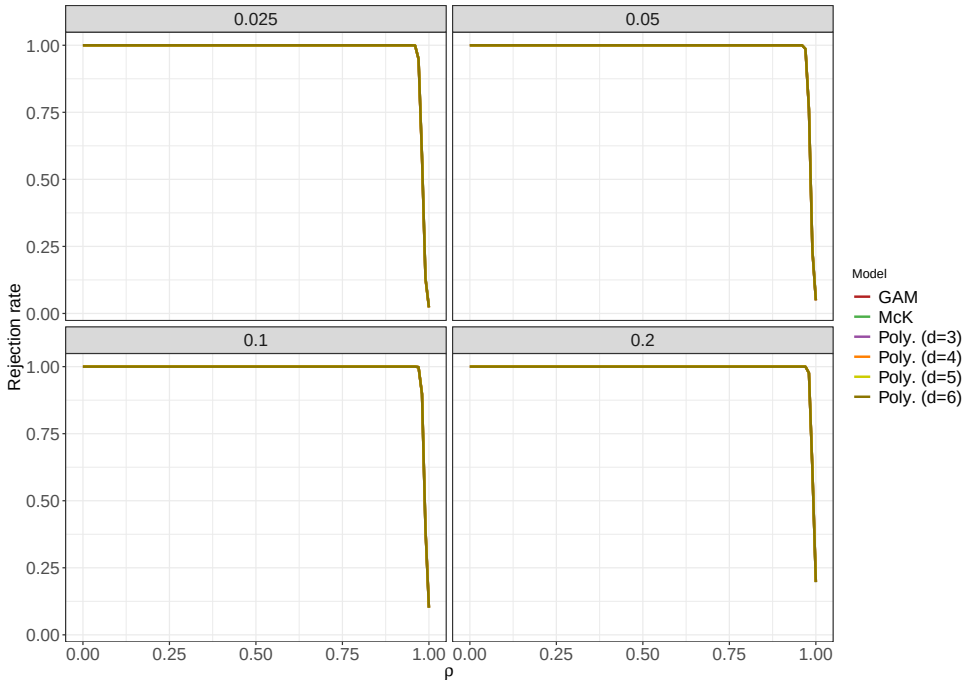


Table 3, but they do improve the model fit (see Figure 14) and the rejection share are now similar to that of the GAM model and MacKinnon's approximation. As an example, Figure 15 shows the share of rejections when $n = 1150$. Previously, the largest discrepancy was observed at this sample size (see Figure 12). Taking similar steps for the other two test types also improves the model fit and the resulting rejection share are consistent with other approaches.[11]

P-Value Plot

One drawback faced during the simulation of the rejection share is that it is restricted to only the selected significance levels α. In order to compare the models at another α, the whole simulation will need to be repeated. Therefore, the above analyses will be complemented with p value plots, a graphical method as suggested by Davidson and MacKinnon (1998). The plots are based on the empirical distribution function (EDF) of the t-statistics. For $x_j \in (0, 1)$, the EDF $\hat{F}$ is defined as

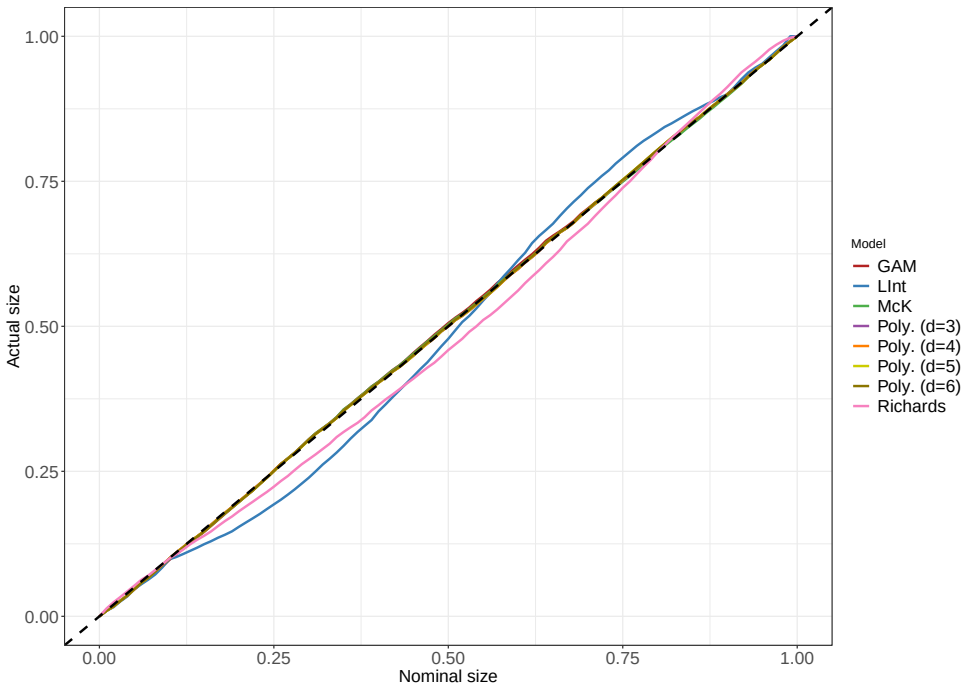
$$\hat{F}(x_j) = \frac{1}{N} \sum_{j=1}^{N} \mathbf{1}\{p_j \leq x_j\} \tag{26}$$

where $\mathbf{1}$ is an indicator function (Davidson and MacKinnon, 1998, p. 2–3). The p-value plot plots $\hat{F}(x_j)$ against x_j, as shown in Figure 16. As described by Davidson and MacKinnon (1998, p. 3), the resulting line graph from plotting $\hat{F}(x_j)$ (Actual size) against x_j (Nominal size) should lie close to the line through origin or the 45°

[11] See Figure A11 and A15 for unit root test with drift and trend and Figure A12 and A16 for unit root test with only drift.

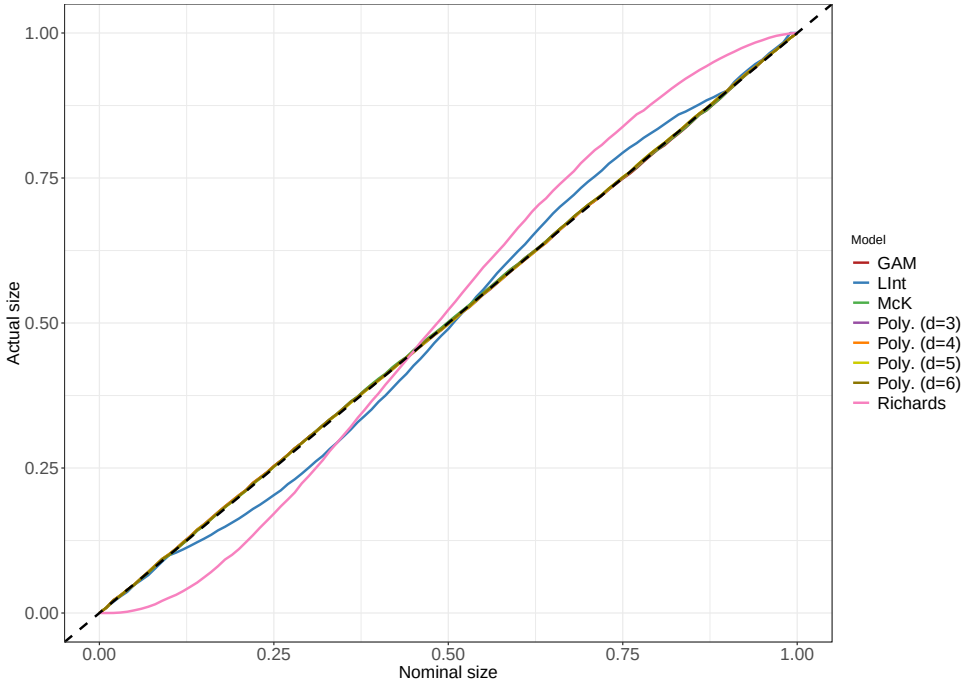
line (the dashed line in the figure). The 45° line will be henceforth referred to as the reference line. Deviations from the reference line are indication of systematic over-rejections or under-rejections and thus the further the graph lie from the line, the more a test over or under-rejects. When sample size is small, e.g. $n = 27$ as in Figure 16, the linear interpolation approach and Richards' curve deviate considerably from the reference line after $x_j > 0.10$. The linear interpolation at first under-rejects and then over-rejects, while the model based on Richards' curve under-rejects for the most part. Again, the behavior of the linear interpolation can be explained in terms of the way it is constructed, i.e. the empirical percentile values tables from Dickey and Fuller, allow better approximation at the lower and upper tails of the distribution than the middle part. Except the two, all of the other models along with the MacKinnon's approximation lie quite close to the reference line and does not appear to considerably either over or under-reject. When the sample size is gradually increased to $n = 57$ and onwards, as in Figure 17 to 20, there is no noticeable change in the linear interpolation, but the deviation of Richards' curve becomes more prominent. The rest of the models do not show any noticeable change as well.

Figure 16: P-value plots for unit root tests with drift and trend, $n = 27$. Source: Own illustration.



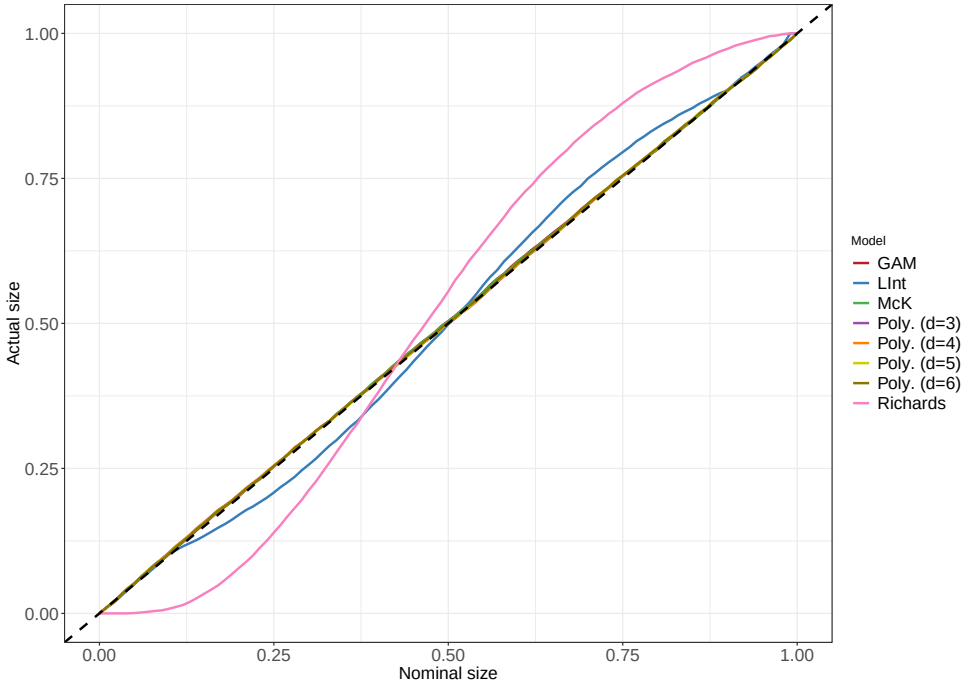
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure 17: P-value plots for unit root tests with drift and trend, $n = 57$. Source: Own illustration.



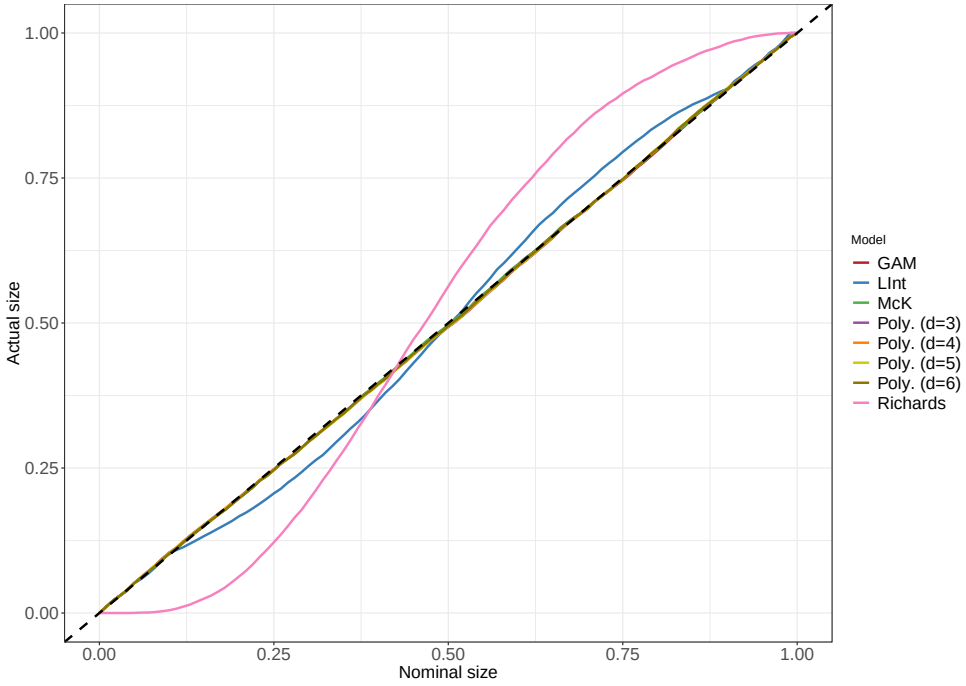
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure 18: P-value plots for unit root tests with drift and trend, $n = 130$. Source: Own illustration.



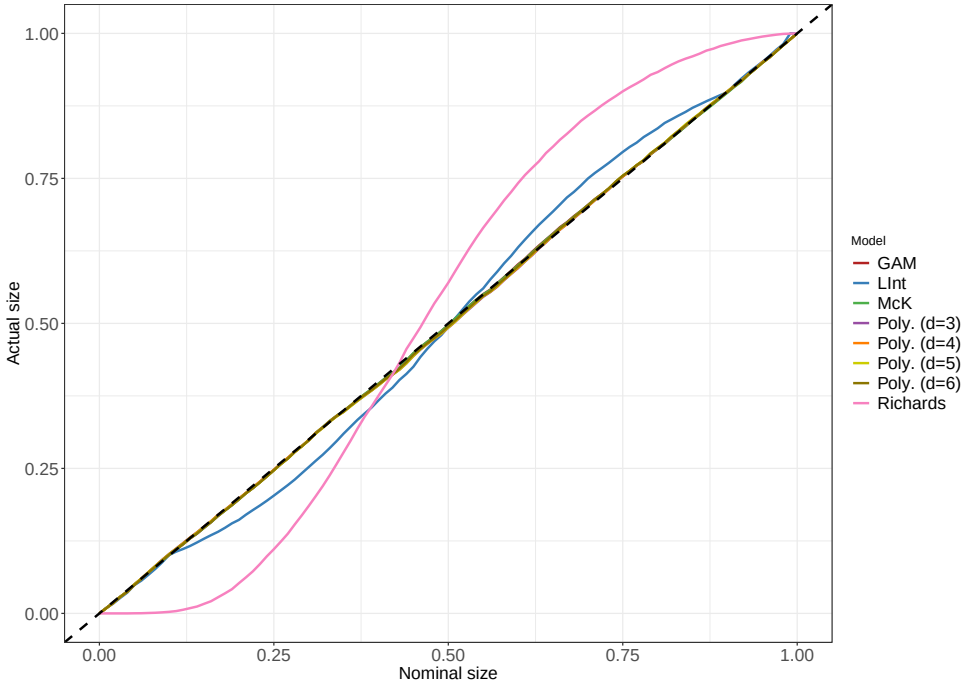
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure 19: P-value plots for unit root tests with drift and trend, $n = 265$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure 20: P-value plots for unit root tests with drift and trend, $n = 1150$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

The p-value plots for the other two types of unit root test have been included in Appendix E. The linear interpolation approach shows similar behavior, i.e. the graph lies close to the reference line in the left and right tail of the distribution, but then under-rejects followed by over-rejection. In the absence of linear trends, as already observed, the Richards' curve performs similar to the rest of the models. Only in the absence of both drift and trend and when the sample size is small ($n = 27$, Figure A17), it appears that all the models, except the MacKinnon's approximation, deviates from the reference and start to over-reject in the middle part of the distribution, with the polynomial and Richards' curve models having larger deviation than GAM. The deviations, however, diminish with increasing sample sizes.

6 Software Implementation

Based on the model evaluation results from Section 5, the corrected polynomial regression and GAM models have good accuracy and desirable properties. The models can be bundled together in a `R`-package, so that it becomes possible for other users to use and benefit from these pre-trained models. With this in mind, the package `pvurt`, as in short for ***p*-v**alue for **u**nit **r**oot **t**est, has been created.[12] Currently, the package has not been published in CRAN package repository but is only available in github. In this section, the package is introduced and its use is illustrated with examples.

The package `pvurt` has been primarily designed to act as an extension for other popular functions/packages that do DF or ADF tests. Currently, it supports the following four functions from the three packages: `adfTest()` and `unitrootTest()` from `fUnitRoots` by Wuertz et al. (2017), `adf.test()` from `tseries` by Trapletti and Hornik (2018) and, lastly, `ur.df()` from `urca` by Pfaff (2008). This has been achieved by either utilizing existing `R` `S4`[13] object classes from the respective packages or creating new `S4` classes that inherits and imitates the functionality of the parent classes. There is only one user facing function in `pvurt`, the `computePValue()` function, that takes the output from each of the aforementioned functions, approximates the p-value using either the polynomial regression or GAM model and then appends it to the original output. The `computePValue()` function itself is just a `R` generic function[14] that calls the appropriate methods for each packages. This approach has advantages both on technical side and for the end user. From technical point-of-view, the approach allows to easily extend the support for additional functions and/or packages, without the need of considerably changing existing functionalities, while the end user has to learn only one function instead of multiple ones. Thus, incorporating `pvurt` in existing script is quite straightforward and amounts to simply passing the output from the other functions to `computePValue()` or simply chaining it using the pipe (`%>%`) operator. Since the package is currently hosted only in github, it must be installed from github directly.

```
devtools::install_github("mlincon/pvurt")
```

The following examples illustrates the use of the package and `computePValue()` function. Based on the example by Pfaff (2008, p. 91–92), a dataset containing the real consumption expenditure in 1985 prices for the United Kingdom from 1966:Q4 to 1991:Q2 is used for the examples. This data is available as `Raotbl3` in the `urca` package.

[12] The manual to the package has been included in Appendix F.
[13] For details regarding the `S4` object system, see Wickham (2019, p. 341–362).
[14] For details on generic functions and `S3` object system, see Wickham (2019, p. 297–324).

```
library(urca)
data("Raotbl3")
head(Raotbl3)
```

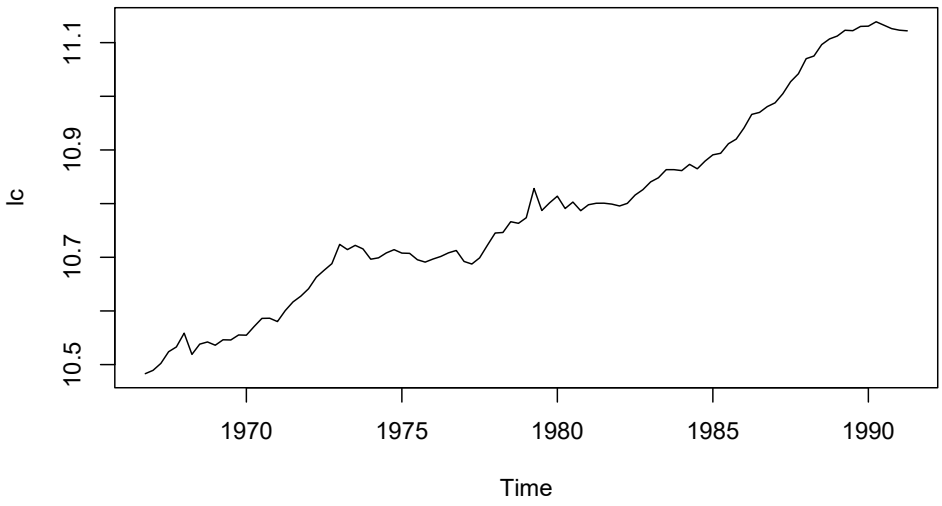
Table 4: The first five rows from the `Raotbl3` data. Source: `Raotbl3` data from `urca` package.

Quarter	lc	li	lw	dd682	dd792	dd883
1966.4	10.4831	10.5821	12.9481	.	.	.
1967.1	10.4893	10.5800	12.9895	0	0	0
1967.2	10.5022	10.5990	13.0115	0	0	0
1967.3	10.5240	10.6262	13.0411	0	0	0
1967.4	10.5329	10.6145	13.0357	0	0	0

The column of interest is `lc` which contains the real consumption expenditure series. We will like to test whether the real consumption contains unit root using the ADF test.

```
lc <- ts(Raotbl3$lc, start = c(1966, 4), end = c(1991, 2),
         frequency = 4)
plot(lc)
```

Figure 21: Plot of the `lc` series. Source: Own illustration.



We start off, by using the `adfTest()` and `unitrootTest()`[15] functions, both from the `fUnitRoots` package. The first of the two uses the empirical percentile or critical values tables from Dickey and Fuller, while the latter uses MacKinnon's approximation.

```
library(fUnitRoots)
fadfTest <- adfTest(lc, lags = 3, type = "ct")
funitrootTest <- unitrootTest(lc, lags = 3, type = "ct")
```

[15]The `unitrootTest` function actually provides an interface to the Fortran routines authored and made publicly available by MacKinnon. The source code can be found in `http://qed.econ.queensu.ca/jae/1996-v11.6/mackinnon/` (Accessed: 01/09/2019).

```
print(fadfTest)

##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 3
##   STATISTIC:
##     Dickey-Fuller: -2.2389
##   P VALUE:
##     0.4777
##
## Description:
##  Thu Sep 12 14:40:36 2019 by user: User

print(funitrootTest)

##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 3
##   STATISTIC:
##     DF: -2.2389
##   P VALUE:
##     t: 0.4627
##     n: 0.9622
##
## Description:
##  Thu Sep 12 14:40:36 2019 by user: User
```

From the output above, the p-value is quite large and hence the null hypothesis H_0 cannot be rejected at conventional significance levels. Thus the series does have a unit root. Note the slight difference in the p-values between the two functions. The difference between the values arises due to difference in approach for approximating p-value as already stated above. Since the results have been assigned to objects, simply pass the saved object to `computePValue()` or chain it, as below in the code block below:

```
library(pvurt)
library(magrittr)  # for %>% operator

computePValue(fadfTest)

##
## Title:
##  Augmented Dickey-Fuller Test
```

```
##
## Test Results:
##   PARAMETER:
##     Lag Order: 3
##   STATISTIC:
##     Dickey-Fuller: -2.2389
##   P VALUE:
##     t: 0.4777
##     pvurt: 0.4627 <-----
##
## Description:
##  Thu Sep 12 14:41:34 2019 by user: User
```

```
funitrootTest %>%
  computePValue()
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 3
##   STATISTIC:
##     DF: -2.2389
##   P VALUE:
##     t: 0.4627
##     n: 0.9622
##     pvurt: 0.4627 <-----
##
## Description:
##  Thu Sep 12 14:41:34 2019 by user: User
```

In both cases, the output still looks the same after applying `computePValue()`, except that a new entry has been appended under the `P VALUE` section with the label `pvurt` (as indicated with the arrow sign[16]). The corresponding value at this label is the approximation done by the `computePValue()` function. By default, unless specified, the GAM model is used. Alternatively, the polynomial regression models can be used as well. In this case, a polynomial degree between 3 and 6 must be provided as well, as show below:

```
funitrootTest %>%
  computePValue("poly", d = 4)
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
```

[16]The arrow sign has only been included here to highlight the appended output. It does not appear while running the code in R.

```
##   PARAMETER:
##     Lag Order: 3
##   STATISTIC:
##     DF: -2.2389
##   P VALUE:
##     t: 0.4627
##     n: 0.9622
##     pvurt: 0.466
##
## Description:
##  Thu Sep 12 14:47:31 2019 by user: User
```

Unlike the two functions from `fUnitRoots`, `ur.df()` does not provide any p-values but only the test statistics as seen below:

```
library(urca)

# Without computePValue
ur.df(lc, lags = 3, type = "trend")

##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -2.2389 3.7382 2.5972
```

The first of the three values is the unit root t-statistics, the last two are F type statistics (Pfaff, 2008, p. 92). Using `computePValue()`, it is possible to get the p-value for the t-statistic in the same output:

```
# With computePValue
ur.df(lc, lags = 3, type = "trend") %>% computePValue()

##
## ###############################################################
## # Augmented Dickey-Fuller Test Unit Root / Cointegration Test #
## ###############################################################
##
## The value of the test statistic is: -2.2389 3.7382 2.5972
## Approximated P value is (pvurt):     0.4627 NA NA         <-----
```

As seen in the function output, `computePValue` appends the approximated p-value directly below the t-statistic. Since no p-value for the F-statisitcs are computed, the place below the two values are filled with `NA`. The package `urca` also provides a class to summarize the output from `ur.df()`. The package `pvurt` ensures that the summary also includes the p-value when `computePValue()` is used, as shown in the following code block.

```r
summary(computePValue(ur.df(lc, lags = 3, type = "trend")))

# or alternative with %>%
ur.df(lc, lags = 3, type = "trend") %>%
  computePValue() %>%
  summary()

##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.044714 -0.006525  0.000129  0.006225  0.045353
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.7976591  0.3547775   2.248   0.0270 *
## z.lag.1     -0.0758706  0.0338880  -2.239   0.0277 *
## tt           0.0004915  0.0002159   2.277   0.0252 *
## z.diff.lag1 -0.1063957  0.1006744  -1.057   0.2934
## z.diff.lag2  0.2011373  0.1012373   1.987   0.0500 .
## z.diff.lag3  0.2998586  0.1020548   2.938   0.0042 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01307 on 89 degrees of freedom
## Multiple R-squared:  0.1472,Adjusted R-squared:  0.09924
## F-statistic: 3.071 on 5 and 89 DF,  p-value: 0.01325
##
##
## Value of test-statistic is:     -2.2389 3.7382 2.5972
## Approximated P value is (pvurt): 0.4627 NA NA         <-----
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

The `adf.test()` from `tseries` has the most limited functionality among the four functions, as it does not allow to specify the test type but always consider that both drift and trend are included. It also outputs p-value for the t-statistic based on the values from Dickey and Fuller. The following code block shows the output without and with `computePValue()`. The `computePValue()` appends the approximated p-

value along with the label `pvurt` in the same output.

```
library(tseries)

# Without computePValue
adf.test(lc, alternative = "stationary", k = 3)

##
##  Augmented Dickey-Fuller Test
##
## data:  lc
## Dickey-Fuller = -2.2389, Lag order = 3, p-value = 0.4777
## alternative hypothesis: stationary

# With computePValue
adf.test(lc, alternative = "stationary", k = 3) %>%
   computePValue()

##
##  Augmented Dickey-Fuller Test
##
## data:  lc
## Dickey-Fuller = -2.239, Lag order = 3, p-value = 0.4777, pvurt =
## 0.4627 <-----
## alternative hypothesis: stationary
```

We observe a slight difference in the p-value. Again, this difference is due to the underlying different approximation approach.

Finally, it is also possible to approximate the p-value from a given t-statistic and sample size simply using only `computePValue()`, as shown in the example below. This provides flexibility and allows to use `computePValue()` without begin absolutely dependent of the above three packages.

```
tStat <- -2.239
sampleSize <- 100
computePValue(tStat, n = sampleSize, model = "gam", type = "ct")

## [1] 0.4626582
```

When `computePValue()` is used this way, the test type should be included as well. In the above example, `ct` stands for test with both drift and trend. Similar `nc` is for test without drift and trend and `c` is for test with only drift. This notation is similar to those from the `fUnitRoots` package. If no option is selected, then `nc` is used as default.

7 Conclusion

This thesis explores the application of machine learning methods for approximating non-standard distribution functions. Determining a closed form analytical expression for a distribution is no trivial task and sometimes such a form does not exist. Hence, in order to use the distributions in statistical analysis, one often relies on crude simulations. One such example of non-standard distribution function is so-called the Dickey-Fuller distribution. The distribution is not normal, even asymptotically. Additionally, the distribution also depends on the test type or inclusion of deterministic elements, i.e. drift or constant and linear time trend. Currently, two approaches can be found in the literature and statistical software to deal with the distribution. The first involves linear interpolation based on empirical percentile values or critical values tables provided by Dickey and Fuller. The tables only consider selected sample sizes and percentiles, with more concentration in the tails (see Appendix A). Thus, the approximation of the linear interpolation is relatively better at the tails than the middle part of the distribution. The second approach uses MacKinnon's approximation based on response surface regressions, which covers more percentiles and sample sizes than Dickey and Fuller.

Machine learning methods take a more data-driven approach to approximate the distributions functions. To illustrate the use of machine learning methods, the Dickey-Fuller distribution has been chosen and the following three common test types are considered: unit root test without drift and trend, with only drift and with both drift and trend. The machine learning approaches are used to approximate the empirical CDF. Since the DF or ADF test is a left-tailed test, the percentile rank or cumulative probability can be used a p-value for hypothesis testing. The percentile rank p is dependent on the test statistic t and sample size n. To train the models, hence, at first train and test data is generated using Monte Carlo simulations.

When presented with such a task, it is important to try and compare several methods, as there is no predetermined single method that is guaranteed to perform well enough. Since the cumulative distribution function is non-linear, a good starting point is polynomial regressions. By varying the polynomial degree d it is possible to increase the flexibility of the models. However, increasing the flexibility comes at the cost of the models becoming increasingly complex and there are chances of overfitting. One way to determine the appropriate d can be to use cross-validation to find the degree that leads to the minimum RMSE and/or MAE. However, it is observed that while increasing d lead to smaller values for RMSE and MAE, the decrease is small. Additionally, increasing d by one adds two new terms in the regression. For example, if the minimum for RMSE ro MAE is at $d = 18$, the corresponding regression has 38 parameters including intercept. In comparison, a model with $d = 4$ has only 10 parameters for a slightly larger error score. To deal with this issue, the variable selection property of lasso regression is utilized, since lasso shrinks the value of coefficients towards zero through regularization or penalization. Suitable value

for the tunning parameter λ in lasso can be determined using cross-validation. Then for the selected λ, the resulting regression models are sparser than the unpenalized model as some of the coefficients are exactly zero and hence the corresponding terms can be left out of the models. However, whether penalized or unpenalized polynomial regressions are used, their parametric nature means that the functional form must be determined beforehand. Non-parametric approaches like GAM allows to build models without pre-specifying the functional form. In GAM, the flexibility of the models are governed by the number of knots or basis functions. Additionally, through tensor products, it is possible to build interactions between multiple variables. Similar to GAM, another approach that utilizes knots and basis functions is MARS. Although MARS is more suitable for high dimensional data, it is still interesting to observe how well it performs compared to the other approaches. Another attempted parametric approach involved adapting the Richards' generalized logistic function to account for both t and n variables. Due to non-linearity in parameters of the resulting model in parameters, the coefficients must be estimated using non-linear least-square algorithms. Finally, multilayer feedforward neural networks are implemented. Although the universal approximation theorem states that a single layer with finite number of nodes can approximate any continuous function arbitrarily well, networks consisting of three to four layers, depending on the test type, are used instead as it leads to the smallest MAE during hyperparameter tunning.

Among the aforementioned methods, the neural network and MARS have the lowest overall accuracy, while lasso provided no benefit over lower degree polynomials based on RMSE and MAE values computed using the test dataset. Taking into consideration of the results from the simulated share of rejection of null hypothesis and p-value plots, the model based on Richards' curve performed poorly when the test included linear time trend. Thus, out of the six, only the GAM and polynomials models, which performed on a par with MacKinnon's approximation and are better than the more widely used linear interpolation approach, are selected. It is important to note at this point, that simply because, for example, GAM model performed well for the Dickey-Fuller distribution and neural network did not, this does not imply that the same would hold true for other non-standard or even standard distribution functions. Yerukala et al. (2011) and Sokouti et al. (2017), for instance, successfully used neural networks to approximate the normal distribution.

The pre-trained GAM and polynomial models have been then bundled into an `R`-package called `pvurt`. The package contains only one user facing function that, for a given test type and t and n value, computes the corresponding cumulative probability or p-value. The function is versatile, since it can be used in combination with four other functions from three different packages that implements ADF unit root test, as well as independently. The versatile nature allows the function to be incorporated at already existing analysis with minimal effort and change in code.

To summarize, machine learning methods offer a modern and viable alternative

to more general numerical or analytical approaches for approximating non-standard distribution functions. By taking a data centric approach, one can utilize the various toolkits and algorithms. It is also important to try out various approaches, starting from the simplest to more complex methods, as there is no single method that works best for all tasks. However, note that the machine learning methods are dependent on the of training data. Complexity of algorithm will not compensate for lack of sufficient data and in such cases, the approximation will fall short of expectation. Therefore, it is important to carry out the data simulation process properly.

8 Further Work

To limit the scope of this, I concentrated only on approximating the Dickey-Fuller distribution function and hence the (Augmented) Dickey-Fuller unit root test. A natural extension would be to include co-integration tests, similar to MacKinnon (1994, 1996, 2010), or consider unit root test in panel data, similar to Bai and Carrion-I-Silvestre (2009). Also the three unit root test types were each considered separately and hence there are separate datasets and models for each of the test types. From modeling perspective, it is indeed possible to consider all the cases in a single model by including an additional dummy or factor variable. Although, this would increase the model complexity, it would be still interesting to observe any difference in model performance. On the software implementation side, it ca be considered to extend support for any further `R`-functions and/or packages and carryout a comprehensive software unit testing. Finally, there is also no reason to bound the use of machine learning to only non-standard distributions. The same principles would also apply when approximating any standard distribution functions.

References

Abadir, K. M. (1995). "The limiting distribution of the t ratio under a unit root." *Econometric Theory*, 11(4), pp. 775–793.

Archontoulis, S. V. and Miguez, F. E. (2014). "Nonlinear regression models and applications in agricultural research." *Agronomy Journal*, 107(2), pp. 786–798.

Bai, J. and Carrion-I-Silvestre, J. L. (2009). "Structural changes, common stochastic trends, and unit roots in panel data." *The Review of Economic Studies*, 76(2), pp. 471–501.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Bowling, S. R., Khasawneh, M. T., Kaewkuekool, S., and Cho, B. R. (2009). "A logistic approximation to the cumulative normal distribution." *Journal of Industrial Engineering and Management*, 2(1), pp. 114–127.

Bryc, W. (2002). "A uniform approximation to the right normal tail integral." *Applied Mathematics and Computation*, 127(2), pp. 365–374.

Cavallini, F. (1993). "Fitting a logistic curve to data." *The College Mathematics Journal*, 24(3), pp. 247–253.

Cybenko, G. (1989). "Approximation by superpositions of a sigmoidal function." *Mathematics of Control, Signals and Systems*, 2(4), pp. 303–314.

Davidson, R. and MacKinnon, J. G. (1998). "Graphical methods for investigating the size and power of hypothesis tests." *The Manchester School*, 66(1), pp. 1–26.

– (2009). *Econometric theory and methods*. International edition. Oxford University Press.

Dickey, D. A. and Fuller, W. A. (1979). "Distribution of the estimators for autoregressive time series with a unit root." *Journal of the American Statistical Association*, 74(366), pp. 427–431.

Dickey, D. A. (1976). "Estimation and hypothesis testing in nonstationary time series." PhD thesis. Iowa State University.

Dietrich, F. K. (2002). "Closed analytical forms and numerical approximation of Dickey-Fuller probability distributions." PhD thesis. Brasenose College, University of Oxford.

Enders, W. (2014). *Applied econometric time series*. 4th ed. Wiley Series in Probability and Statistics. Wiley.

Faraway, J. J. (2016). *Extending the linear model with R : generalized linear, mixed effects and nonparametric regression models*. Second edition. CRC Press.

Fuller, W. A. (1996). *Introduction to statistical time series*. 2nd ed. John Wiley & Sons.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. `http://www.deeplearningbook.org`. MIT Press.

Günther, F. and Fritsch, S. (2010). "neuralnet: Training of neural networks." *The R journal*, 2(1), pp. 30–38.

Hamilton, J. D. (1994). *Time series analysis*. Princeton, NJ: Princeton University Press.

Hanck, C. (10, 2016). "How is the augmented Dickey–Fuller test (ADF) table of critical values calculated?" URL:https://stats.stackexchange.com/q/213589 (version: 2017-04-13).

Hansen, B. E. (1997). "Approximate asymptotic p values for structural-change tests." *Journal of Business and Economic Statistics*, 15(1), pp. 60–67.

Hastie, T. and Tibshirani, R. (1987). "Generalized additive models: Some applications." *Journal of the American Statistical Association*, 82(398), pp. 371–386.

– (1990). *Generalized additive models*. 1. ed. Chapman and Hall.

– (1995). "Generalized additive models for medical research." *Statistical Methods in Medical Research*, 4(3), pp. 187–196.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of statistical learning: Data mining, inference, and prediction*. 2nd ed. 2009. Corr. 12th printing 2017. New York: Springer.

Hayashi, F. (2000). *Econometrics*. Princeton University Press.

Hornik, K. (1991). "Approximation capabilities of multilayer feedforward networks." *Neural Networks*, 4(2), pp. 251–257.

Hornik, K., Stinchcombe, M., and White, H. (1989). "Multilayer feedforward networks are universal approximators." *Neural Networks*, 2(5), pp. 359–366.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R*. 1st ed. 2013. Corr. 4th printing 2014. Springer.

Kuhn, M. (2013). *Applied predictive modeling*. New York: Springer New York.

Lin, J.-T. (1990). "A Simpler Logistic Approximation to the Normal Tail Probability and its Inverse." *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 39(2), pp. 255–257.

MacKinnon, J. G. (1994). "Approximate asymptotic sistribution functions for unit-root and cointegration tests." *Journal of Business & Economic Statistics*, 12(2), pp. 167–176.

– (1996). "Numerical distribution functions for unit root and cointegration tests." *Journal of Applied Econometrics*, 11(6), pp. 601–618.

– (2010). *Critical values for cointegration tests*. Queen's Economics Department Working Paper. Reissued January 2010 with additional results. Department of Economics, Queen's University.

Manning, R. L. (1996). "Logit regressions with continuous dependent variables measured with error." *Applied Economics Letters*, 3(3), pp. 183–184.

Marsaglia, G. (2004). "Evaluating the normal distribution." *Journal of Statistical Software, Articles*, 11(4), pp. 1–11.

Park, K. I. (2018). *Fundamentals of probability and stochastic processes with applications to communications*. Springer International Publishing.

Pearl, R. and Reed, L. J. (1922). "A Further note on the mathematical theory of population growth." *Proceedings of the National Academy of Sciences of the United States of America*, 8 (12), pp. 365–368.

– (1925). "Skew-Growth curves." *Proceedings of the National Academy of Sciences of the United States of America*, 11 (1), pp. 16–22.

Pfaff, B. (2008). *Analysis of integrated and cointegrated time series with R*. Second. ISBN 0-387-27960-1. New York: Springer.

Raab, D. H. and Green, E. H. (1961). "A cosine approximation to the normal distribution." *Psychometrika*, 26(4), pp. 447–450.

Richards, F. J. (1959). "A flexible growth function for empirical use." *Journal of Experimental Botany*, 10(2), pp. 290–301.

Sokouti, M., Sadeghi, R., Pashazadeh, S., Abadi, S. E. H., Ghojazadeh, M., and Sokouti, B. (2017). "Most accurate non-linear approximation of standard normal distribution integral based on artificial neural networks." *Suranaree journal of science and technology*, 24(3), pp. 263–280.

Stock, J. H. and Watson, M. W. (2011). *Introduction to econometrics*. 3rd. Boston: Addison-Wesley.

Szparaga, A. and Kocira, S. (2018). "Generalized logistic functions in modelling emergence of Brassica napus L." *PLOS ONE*, 13(8), pp. 1–14.

Trapletti, A. and Hornik, K. (2018). *tseries: Time series analysis and computational finance*. R package version 0.10-46.

van der Vaart, A. W. (2000). *Asymptotic statistics*. 1st ed. Cambridge [u.a.]: Cambridge University Press.

Waissi, G. R. and Rossin, D. F. (1996). "A sigmoid approximation of the standard normal integral." *Applied Mathematics and Computation*, 77(1), pp. 91–95.

Wickham, H. (1, 2019). *Advanced R, Second Edition*. Chapman & Hall/CRC The R Series. Chapman and Hall/CRC.

Wood, S. N. (2017). *Generalized additive models : an introduction with R*. 2nd ed. Chapman & Hall/CRC.

Wood, S. N., Goude, Y., and Shaw, S. (2015). "Generalized additive models for large data sets." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 64(1), pp. 139–155.

Wood, S. N., Li, Z., Shaddick, G., and Augustin, N. H. (2017). "Generalized additive models for gigadata: Modeling the U.K. black smoke network daily data." *Journal of the American Statistical Association*, 112(519), pp. 1199–1210.

Wuertz, D., Setz, T., and Chalabi, Y. (2017). *fUnitRoots: Rmetrics - Modelling Trends and Unit Roots*. R package version 3042.79.

Yerukala, R., Boiroju, N. K., and Reddy, M. K. (2011). "An approximation to the CDF of standard normal distribution." *International Journal of Mathematical Archive*, 2(7), pp. 1077–1079.

Yin, X., Goudriaan, J., Lantinga, E. A., Vos, J., and Spiertz, H. J. (2003). "A flexible sigmoid function of determinate growth." *Annals of Botany*, 91(3), pp. 361–371.

Appendix

A Critical Values Tables from Dickey and Fuller

Table A1: Empicial percentiles for t_{nc} statistic. Source: Fuller (1996, p. 642)

n	0.01	0.025	0.05	0.10	0.50	0.90	0.95	0.975	0.99
25	−2.65	−2.26	−1.95	−1.60	−0.47	0.92	1.33	1.70	2.15
50	−2.62	−2.25	−1.95	−1.61	−0.49	0.91	1.31	1.66	2.08
100	−2.60	−2.24	−1.95	−1.61	−0.50	0.90	1.29	1.64	2.04
250	−2.58	−2.24	−1.95	−1.62	−0.50	0.89	1.28	1.63	2.02
500	−2.58	−2.23	−1.95	−1.62	−0.50	0.89	1.28	1.62	2.01
∞	−2.58	−2.23	−1.95	−1.62	−0.51	0.89	1.28	1.62	2.01

Table A2: Empicial percentiles for t_c statistic. Source: Fuller (1996, p. 642)

n	0.01	0.025	0.05	0.10	0.50	0.90	0.95	0.975	0.99
25	−3.75	−3.33	−2.99	−2.64	−1.53	−0.37	0.00	0.34	0.71
50	−3.59	−3.23	−2.93	−2.60	−1.55	−0.41	−0.04	0.28	0.66
100	−3.50	−3.17	−2.90	−2.59	−1.56	−0.42	−0.06	0.26	0.63
250	−3.45	−3.14	−2.88	−2.58	−1.56	−0.42	−0.07	0.24	0.62
500	−3.44	−3.13	−2.87	−2.57	−1.57	−0.44	−0.07	0.24	0.61
∞	−3.42	−3.12	−2.86	−2.57	−1.57	−0.44	−0.08	0.23	0.60

Table A3: Empicial percentiles for t_{ct} statistic. Source: Fuller (1996, p. 642)

n	0.01	0.025	0.05	0.10	0.50	0.90	0.95	0.975	0.99
25	−4.38	−3.95	−3.60	−3.24	−2.14	−1.14	−0.81	−0.50	−0.15
50	−4.16	−3.80	−3.50	−3.18	−2.16	−1.19	−0.87	−0.58	−0.24
100	−4.05	−3.73	−3.45	−3.15	−2.17	−1.22	−0.90	−0.62	−0.28
250	−3.98	−3.69	−3.42	−3.13	−2.18	−1.23	−0.92	−0.64	−0.31
500	−3.97	−3.67	−3.42	−3.13	−2.18	−1.24	−0.93	−0.65	−0.32
∞	−3.96	−3.67	−3.41	−3.13	−2.18	−1.25	−0.94	−0.66	−0.32

B RMSE and MAE for Polynomial Regressions

Table A4 lists 36 various polynomial regressions. To highlight the differences among the regressions in a compact and concise form, a short-hand notation is adopted, as illustrated by Equation 27. Thus Equation 17 from Section 4

$$\ln\left(\frac{p_i}{1-p_i}\right) = \gamma_{0,0} + \gamma_{0,1}t_i + \gamma_{0,2}t_i^2 + \gamma_{0,3}t_i^3 + \gamma_{1,0}\left(\frac{1}{n}\right) + \gamma_{1,1}\left(\frac{t_i}{n}\right) + \gamma_{1,2}\left(\frac{t_i^2}{n}\right) + \gamma_{1,3}\left(\frac{t_i^3}{n}\right) + \varepsilon_i,$$

will be represented as

$$y = c + \text{poly}(t,3) + \left(\frac{1}{n}\right) + \text{poly}(t,3)\cdot\left(\frac{1}{n}\right), \tag{27}$$

where $y = \ln\left(\frac{p_i}{1-p_i}\right)$, c is the intercept, $\text{poly}(t,3)$ implies polynomial of degree 3 and $\text{poly}(t,3)\cdot\left(\frac{1}{n}\right)$ represents the interaction of $1/n$ with the polynomial terms.

Table A4: Different specifications of polynomial regressions. The models have been written using a shorthand notation as exemplified in equation 27.

Number	Functional form
1	$y = c + \text{poly}(t,3) + \left(\frac{1}{n}\right) + \text{poly}(t,3) \cdot \left(\frac{1}{n}\right)$
2	$y = c + \text{poly}(t,4) + \left(\frac{1}{n}\right) + \text{poly}(t,4) \cdot \left(\frac{1}{n}\right)$
3	$y = c + \text{poly}(t,5) + \left(\frac{1}{n}\right) + \text{poly}(t,5) \cdot \left(\frac{1}{n}\right)$
4	$y = c + \text{poly}(t,6) + \left(\frac{1}{n}\right) + \text{poly}(t,6) \cdot \left(\frac{1}{n}\right)$
5	$y = c + \text{poly}(t,3) + \sqrt{n} + \text{poly}(t,3) \cdot \sqrt{n}$
6	$y = c + \text{poly}(t,4) + \sqrt{n} + \text{poly}(t,4) \cdot \sqrt{n}$
7	$y = c + \text{poly}(t,5) + \sqrt{n} + \text{poly}(t,5) \cdot \sqrt{n}$
8	$y = c + \text{poly}(t,6) + \sqrt{n} + \text{poly}(t,6) \cdot \sqrt{n}$
9	$y = c + \text{poly}(t,3) + \ln(n) + \text{poly}(t,3) \cdot \ln(n)$
10	$y = c + \text{poly}(t,4) + \ln(n) + \text{poly}(t,4) \cdot \ln(n)$
11	$y = c + \text{poly}(t,5) + \ln(n) + \text{poly}(t,5) \cdot \ln(n)$
12	$y = c + \text{poly}(t,6) + \ln(n) + \text{poly}(t,6) \cdot \ln(n)$
13	$y = c + \text{poly}(t,3) + \left(\frac{1}{n}\right)$
14	$y = c + \text{poly}(t,4) + \left(\frac{1}{n}\right)$
15	$y = c + \text{poly}(t,5) + \left(\frac{1}{n}\right)$
16	$y = c + \text{poly}(t,6) + \left(\frac{1}{n}\right)$
17	$y = c + \text{poly}(t,3) + \sqrt{n}$
18	$y = c + \text{poly}(t,4) + \sqrt{n}$
19	$y = c + \text{poly}(t,5) + \sqrt{n}$
20	$y = c + \text{poly}(t,6) + \sqrt{n}$
21	$y = c + \text{poly}(t,3) + \ln(n)$
22	$y = c + \text{poly}(t,4) + \ln(n)$
23	$y = c + \text{poly}(t,5) + \ln(n)$
24	$y = c + \text{poly}(t,6) + \ln(n)$
25	$y = c + \text{poly}(t,3) + n + \left(\frac{1}{n}\right)$
26	$y = c + \text{poly}(t,4) + n + \left(\frac{1}{n}\right)$
27	$y = c + \text{poly}(t,5) + n + \left(\frac{1}{n}\right)$
28	$y = c + \text{poly}(t,6) + n + \left(\frac{1}{n}\right)$
29	$y = c + \text{poly}(t,3) + n + \sqrt{n}$
30	$y = c + \text{poly}(t,4) + n + \sqrt{n}$
31	$y = c + \text{poly}(t,5) + n + \sqrt{n}$
32	$y = c + \text{poly}(t,6) + n + \sqrt{n}$
33	$y = c + \text{poly}(t,3) + n + \ln(n)$
34	$y = c + \text{poly}(t,4) + n + \ln(n)$
35	$y = c + \text{poly}(t,5) + n + \ln(n)$
36	$y = c + \text{poly}(t,6) + n + \ln(n)$

The five-fold cross validated RMSE and MAE values for the models in Table A4 are tabulated below in Tables A5 to A7. The RMSE and MAE calculations are based on how accurately p is estimated. Thus the model predictions for the polynomial

models are re-mapped at first between 0 and 1 by

$$p_i = \frac{1}{1+\exp(-y)} = \frac{\exp(y)}{1+\exp(y)}.$$

Table A5: Cross-validated RMSE and MAE of different polynomial regression for unit root test without drift and trend. The shaded cells indicate the minimum RMSE and MAE among all the polynomial regression models.

	Full distribution		Lower tail ($p < 0.2$)	
Functional form	RMSE	MAE	RMSE	MAE
$y = c + \text{poly}(t,3) + \left(\frac{1}{n}\right) + \text{poly}(t,3) \cdot \left(\frac{1}{n}\right)$	0.0061	0.0048	0.0029	0.0026
$y = c + \text{poly}(t,4) + \left(\frac{1}{n}\right) + \text{poly}(t,4) \cdot \left(\frac{1}{n}\right)$	0.0052	0.0040	0.0018	0.0015
$y = c + \text{poly}(t,5) + \left(\frac{1}{n}\right) + \text{poly}(t,5) \cdot \left(\frac{1}{n}\right)$	0.0038	0.0029	0.0022	0.0018
$y = c + \text{poly}(t,6) + \left(\frac{1}{n}\right) + \text{poly}(t,6) \cdot \left(\frac{1}{n}\right)$	0.0035	0.0026	0.0020	0.0014
$y = c + \text{poly}(t,3) + \sqrt{n} + \text{poly}(t,3) \cdot \sqrt{n}$	0.0150	0.0096	0.0045	0.0032
$y = c + \text{poly}(t,4) + \sqrt{n} + \text{poly}(t,4) \cdot \sqrt{n}$	0.0148	0.0091	0.0036	0.0023
$y = c + \text{poly}(t,5) + \sqrt{n} + \text{poly}(t,5) \cdot \sqrt{n}$	0.0141	0.0085	0.0036	0.0025
$y = c + \text{poly}(t,6) + \sqrt{n} + \text{poly}(t,6) \cdot \sqrt{n}$	0.0142	0.0083	0.0031	0.0020
$y = c + \text{poly}(t,3) + \ln(n) + \text{poly}(t,3) \cdot \ln(n)$	0.0089	0.0065	0.0033	0.0027
$y = c + \text{poly}(t,4) + \ln(n) + \text{poly}(t,4) \cdot \ln(n)$	0.0084	0.0060	0.0023	0.0018
$y = c + \text{poly}(t,5) + \ln(n) + \text{poly}(t,5) \cdot \ln(n)$	0.0075	0.0053	0.0027	0.0021
$y = c + \text{poly}(t,6) + \ln(n) + \text{poly}(t,6) \cdot \ln(n)$	0.0074	0.0051	0.0022	0.0017
$y = c + \text{poly}(t,3) + \left(\frac{1}{n}\right)$	0.0071	0.0057	0.0051	0.0043
$y = c + \text{poly}(t,4) + \left(\frac{1}{n}\right)$	0.0063	0.0051	0.0046	0.0037
$y = c + \text{poly}(t,5) + \left(\frac{1}{n}\right)$	0.0048	0.0039	0.0048	0.0039
$y = c + \text{poly}(t,6) + \left(\frac{1}{n}\right)$	0.0046	0.0037	0.0046	0.0036
$y = c + \text{poly}(t,3) + \sqrt{n}$	0.0144	0.0095	0.0081	0.0045
$y = c + \text{poly}(t,4) + \sqrt{n}$	0.0142	0.0092	0.0075	0.0040
$y = c + \text{poly}(t,5) + \sqrt{n}$	0.0135	0.0085	0.0078	0.0040
$y = c + \text{poly}(t,6) + \sqrt{n}$	0.0135	0.0084	0.0075	0.0038
$y = c + \text{poly}(t,3) + \ln(n)$	0.0093	0.0070	0.0052	0.0037
$y = c + \text{poly}(t,4) + \ln(n)$	0.0089	0.0065	0.0046	0.0033
$y = c + \text{poly}(t,5) + \ln(n)$	0.0079	0.0057	0.0049	0.0033
$y = c + \text{poly}(t,6) + \ln(n)$	0.0078	0.0056	0.0047	0.0031
$y = c + \text{poly}(t,3) + n + \left(\frac{1}{n}\right)$	0.0082	0.0062	0.0063	0.0048
$y = c + \text{poly}(t,4) + n + \left(\frac{1}{n}\right)$	0.0076	0.0057	0.0057	0.0042
$y = c + \text{poly}(t,5) + n + \left(\frac{1}{n}\right)$	0.0063	0.0044	0.0060	0.0044
$y = c + \text{poly}(t,6) + n + \left(\frac{1}{n}\right)$	0.0061	0.0043	0.0058	0.0041
$y = c + \text{poly}(t,3) + n + \sqrt{n}$	0.0423	0.0173	0.0144	0.0066
$y = c + \text{poly}(t,4) + n + \sqrt{n}$	0.0420	0.0169	0.0142	0.0063
$y = c + \text{poly}(t,5) + n + \sqrt{n}$	0.0414	0.0159	0.0143	0.0063
$y = c + \text{poly}(t,6) + n + \sqrt{n}$	0.0413	0.0158	0.0142	0.0062
$y = c + \text{poly}(t,3) + n + \ln(n)$	0.0215	0.0108	0.0090	0.0051
$y = c + \text{poly}(t,4) + n + \ln(n)$	0.0211	0.0103	0.0088	0.0048
$y = c + \text{poly}(t,5) + n + \ln(n)$	0.0203	0.0091	0.0089	0.0048
$y = c + \text{poly}(t,6) + n + \ln(n)$	0.0202	0.0090	0.0088	0.0047

Table A6: Cross-validated RMSE and MAE of different polynomial regressions for unit root test with drift. The shaded cells indicate the minimum RMSE and MAE among all the polynomial regression models.

	Full distribution		Lower tail ($p < 0.2$)	
Functional form	RMSE	MAE	RMSE	MAE
$y = c + \text{poly}(t, 3) + \left(\frac{1}{n}\right) + \text{poly}(t, 3) \cdot \left(\frac{1}{n}\right)$	0.0027	0.0021	0.0011	0.0008
$y = c + \text{poly}(t, 4) + \left(\frac{1}{n}\right) + \text{poly}(t, 4) \cdot \left(\frac{1}{n}\right)$	0.0019	0.0015	0.0008	0.0006
$y = c + \text{poly}(t, 5) + \left(\frac{1}{n}\right) + \text{poly}(t, 5) \cdot \left(\frac{1}{n}\right)$	0.0023	0.0018	0.0009	0.0007
$y = c + \text{poly}(t, 6) + \left(\frac{1}{n}\right) + \text{poly}(t, 6) \cdot \left(\frac{1}{n}\right)$	0.0022	0.0017	0.0010	0.0007
$y = c + \text{poly}(t, 3) + \sqrt{n} + \text{poly}(t, 3) \cdot \sqrt{n}$	0.0085	0.0052	0.0050	0.0033
$y = c + \text{poly}(t, 4) + \sqrt{n} + \text{poly}(t, 4) \cdot \sqrt{n}$	0.0072	0.0043	0.0048	0.0029
$y = c + \text{poly}(t, 5) + \sqrt{n} + \text{poly}(t, 5) \cdot \sqrt{n}$	0.0076	0.0046	0.0050	0.0031
$y = c + \text{poly}(t, 6) + \sqrt{n} + \text{poly}(t, 6) \cdot \sqrt{n}$	0.0075	0.0045	0.0051	0.0031
$y = c + \text{poly}(t, 3) + \ln(n) + \text{poly}(t, 3) \cdot \ln(n)$	0.0047	0.0035	0.0029	0.0022
$y = c + \text{poly}(t, 4) + \ln(n) + \text{poly}(t, 4) \cdot \ln(n)$	0.0041	0.0029	0.0028	0.0019
$y = c + \text{poly}(t, 5) + \ln(n) + \text{poly}(t, 5) \cdot \ln(n)$	0.0043	0.0032	0.0029	0.0021
$y = c + \text{poly}(t, 6) + \ln(n) + \text{poly}(t, 6) \cdot \ln(n)$	0.0043	0.0031	0.0029	0.0020
$y = c + \text{poly}(t, 3) + \left(\frac{1}{n}\right)$	0.0049	0.0037	0.0041	0.0032
$y = c + \text{poly}(t, 4) + \left(\frac{1}{n}\right)$	0.0041	0.0029	0.0041	0.0029
$y = c + \text{poly}(t, 5) + \left(\frac{1}{n}\right)$	0.0044	0.0033	0.0041	0.0030
$y = c + \text{poly}(t, 6) + \left(\frac{1}{n}\right)$	0.0044	0.0033	0.0041	0.0030
$y = c + \text{poly}(t, 3) + \sqrt{n}$	0.0052	0.0039	0.0043	0.0033
$y = c + \text{poly}(t, 4) + \sqrt{n}$	0.0047	0.0032	0.0043	0.0031
$y = c + \text{poly}(t, 5) + \sqrt{n}$	0.0049	0.0035	0.0043	0.0031
$y = c + \text{poly}(t, 6) + \sqrt{n}$	0.0049	0.0036	0.0043	0.0031
$y = c + \text{poly}(t, 3) + \ln(n)$	0.0050	0.0038	0.0041	0.0032
$y = c + \text{poly}(t, 4) + \ln(n)$	0.0042	0.0029	0.0041	0.0030
$y = c + \text{poly}(t, 5) + \ln(n)$	0.0045	0.0033	0.0041	0.0030
$y = c + \text{poly}(t, 6) + \ln(n)$	0.0045	0.0034	0.0041	0.0030
$y = c + \text{poly}(t, 3) + n + \left(\frac{1}{n}\right)$	0.0050	0.0038	0.0045	0.0034
$y = c + \text{poly}(t, 4) + n + \left(\frac{1}{n}\right)$	0.0044	0.0030	0.0044	0.0031
$y = c + \text{poly}(t, 5) + n + \left(\frac{1}{n}\right)$	0.0047	0.0034	0.0044	0.0032
$y = c + \text{poly}(t, 6) + n + \left(\frac{1}{n}\right)$	0.0047	0.0035	0.0044	0.0032
$y = c + \text{poly}(t, 3) + n + \sqrt{n}$	0.0110	0.0059	0.0057	0.0036
$y = c + \text{poly}(t, 4) + n + \sqrt{n}$	0.0112	0.0053	0.0062	0.0036
$y = c + \text{poly}(t, 5) + n + \sqrt{n}$	0.0112	0.0056	0.0060	0.0036
$y = c + \text{poly}(t, 6) + n + \sqrt{n}$	0.0113	0.0056	0.0061	0.0036
$y = c + \text{poly}(t, 3) + n + \ln(n)$	0.0065	0.0045	0.0042	0.0031
$y = c + \text{poly}(t, 4) + n + \ln(n)$	0.0062	0.0037	0.0044	0.0030
$y = c + \text{poly}(t, 5) + n + \ln(n)$	0.0064	0.0040	0.0043	0.0030
$y = c + \text{poly}(t, 6) + n + \ln(n)$	0.0063	0.0041	0.0043	0.0030

Table A7: Cross-validated RMSE and MAE of different polynomial regressions for unit root test with drift and trend. The shaded cells indicate the minimum RMSE and MAE among all the polynomial regression models.

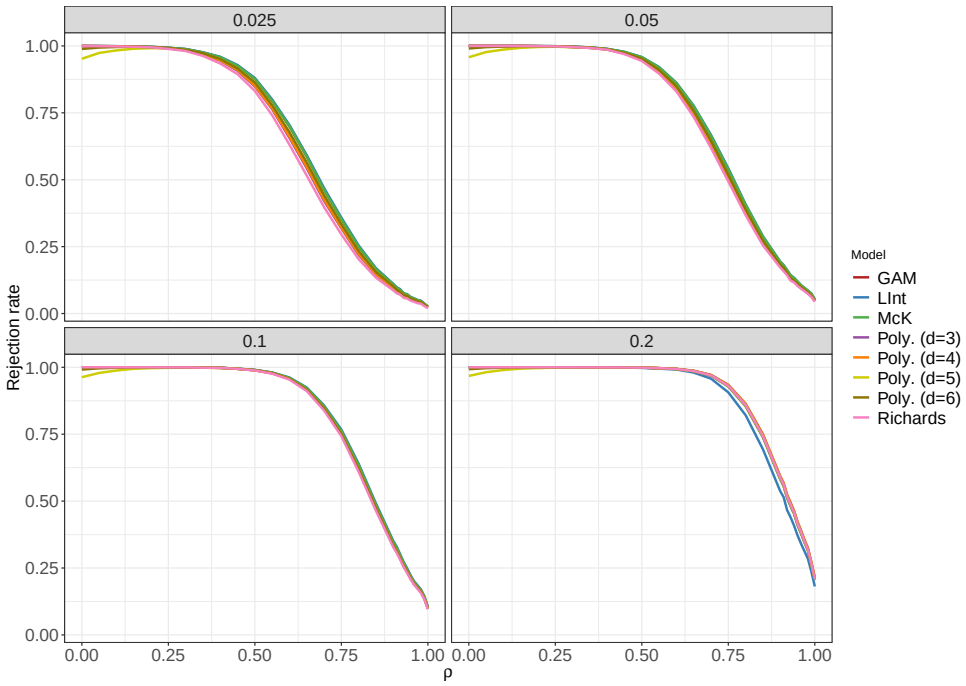
	Full distribution		Lower tail ($p < 0.2$)	
Functional form	RMSE	MAE	RMSE	MAE
$y = c + \text{poly}(t, 3) + \left(\frac{1}{n}\right) + \text{poly}(t, 3) \cdot \left(\frac{1}{n}\right)$	0.0026	0.0021	0.0019	0.0016
$y = c + \text{poly}(t, 4) + \left(\frac{1}{n}\right) + \text{poly}(t, 4) \cdot \left(\frac{1}{n}\right)$	0.0022	0.0018	0.0017	0.0014
$y = c + \text{poly}(t, 5) + \left(\frac{1}{n}\right) + \text{poly}(t, 5) \cdot \left(\frac{1}{n}\right)$	0.0022	0.0018	0.0015	0.0011
$y = c + \text{poly}(t, 6) + \left(\frac{1}{n}\right) + \text{poly}(t, 6) \cdot \left(\frac{1}{n}\right)$	0.0017	0.0013	0.0011	0.0009
$y = c + \text{poly}(t, 3) + \sqrt{n} + \text{poly}(t, 3) \cdot \sqrt{n}$	0.0071	0.0047	0.0053	0.0036
$y = c + \text{poly}(t, 4) + \sqrt{n} + \text{poly}(t, 4) \cdot \sqrt{n}$	0.0067	0.0044	0.0052	0.0035
$y = c + \text{poly}(t, 5) + \sqrt{n} + \text{poly}(t, 5) \cdot \sqrt{n}$	0.0067	0.0045	0.0051	0.0035
$y = c + \text{poly}(t, 6) + \sqrt{n} + \text{poly}(t, 6) \cdot \sqrt{n}$	0.0065	0.0043	0.0049	0.0034
$y = c + \text{poly}(t, 3) + \ln(n) + \text{poly}(t, 3) \cdot \ln(n)$	0.0052	0.0035	0.0038	0.0026
$y = c + \text{poly}(t, 4) + \ln(n) + \text{poly}(t, 4) \cdot \ln(n)$	0.0049	0.0033	0.0037	0.0025
$y = c + \text{poly}(t, 5) + \ln(n) + \text{poly}(t, 5) \cdot \ln(n)$	0.0048	0.0033	0.0036	0.0024
$y = c + \text{poly}(t, 6) + \ln(n) + \text{poly}(t, 6) \cdot \ln(n)$	0.0046	0.0031	0.0034	0.0023
$y = c + \text{poly}(t, 3) + \left(\frac{1}{n}\right)$	0.0071	0.0050	0.0077	0.0055
$y = c + \text{poly}(t, 4) + \left(\frac{1}{n}\right)$	0.0072	0.0049	0.0077	0.0055
$y = c + \text{poly}(t, 5) + \left(\frac{1}{n}\right)$	0.0070	0.0050	0.0075	0.0055
$y = c + \text{poly}(t, 6) + \left(\frac{1}{n}\right)$	0.0070	0.0049	0.0076	0.0055
$y = c + \text{poly}(t, 3) + \sqrt{n}$	0.0076	0.0053	0.0075	0.0056
$y = c + \text{poly}(t, 4) + \sqrt{n}$	0.0077	0.0053	0.0075	0.0055
$y = c + \text{poly}(t, 5) + \sqrt{n}$	0.0075	0.0053	0.0073	0.0055
$y = c + \text{poly}(t, 6) + \sqrt{n}$	0.0075	0.0052	0.0074	0.0055
$y = c + \text{poly}(t, 3) + \ln(n)$	0.0073	0.0051	0.0076	0.0056
$y = c + \text{poly}(t, 4) + \ln(n)$	0.0075	0.0051	0.0076	0.0055
$y = c + \text{poly}(t, 5) + \ln(n)$	0.0073	0.0051	0.0074	0.0055
$y = c + \text{poly}(t, 6) + \ln(n)$	0.0073	0.0051	0.0075	0.0055
$y = c + \text{poly}(t, 3) + n + \left(\frac{1}{n}\right)$	0.0070	0.0049	0.0078	0.0056
$y = c + \text{poly}(t, 4) + n + \left(\frac{1}{n}\right)$	0.0071	0.0049	0.0078	0.0055
$y = c + \text{poly}(t, 5) + n + \left(\frac{1}{n}\right)$	0.0069	0.0050	0.0076	0.0056
$y = c + \text{poly}(t, 6) + n + \left(\frac{1}{n}\right)$	0.0069	0.0049	0.0077	0.0056
$y = c + \text{poly}(t, 3) + n + \sqrt{n}$	0.0078	0.0053	0.0076	0.0055
$y = c + \text{poly}(t, 4) + n + \sqrt{n}$	0.0078	0.0053	0.0076	0.0055
$y = c + \text{poly}(t, 5) + n + \sqrt{n}$	0.0076	0.0053	0.0073	0.0054
$y = c + \text{poly}(t, 6) + n + \sqrt{n}$	0.0076	0.0053	0.0074	0.0054
$y = c + \text{poly}(t, 3) + n + \ln(n)$	0.0073	0.0050	0.0076	0.0055
$y = c + \text{poly}(t, 4) + n + \ln(n)$	0.0074	0.0050	0.0076	0.0054
$y = c + \text{poly}(t, 5) + n + \ln(n)$	0.0072	0.0051	0.0074	0.0055
$y = c + \text{poly}(t, 6) + n + \ln(n)$	0.0071	0.0050	0.0075	0.0055

C Comparison of Share of Rejections

The figures below show the result of the test comparing the share of rejection of the various models including MacKinnon's approximation and the linear interpolation approach. The types of unit root test shown below are the tests without both drift and trend and test with only drift. For test with both drift and trend, the results are shown in Section 5, Figure 8 to12.

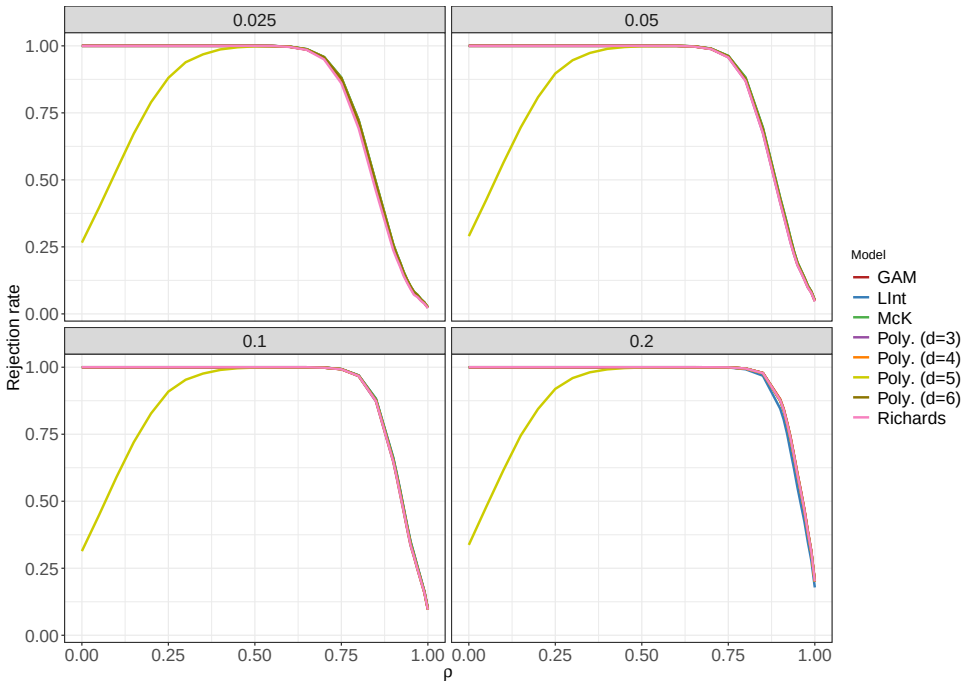
Without drift and trend

Figure A1: Share of rejection from 10,000 repetition of unit root test without drift and trend, $n = 27$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



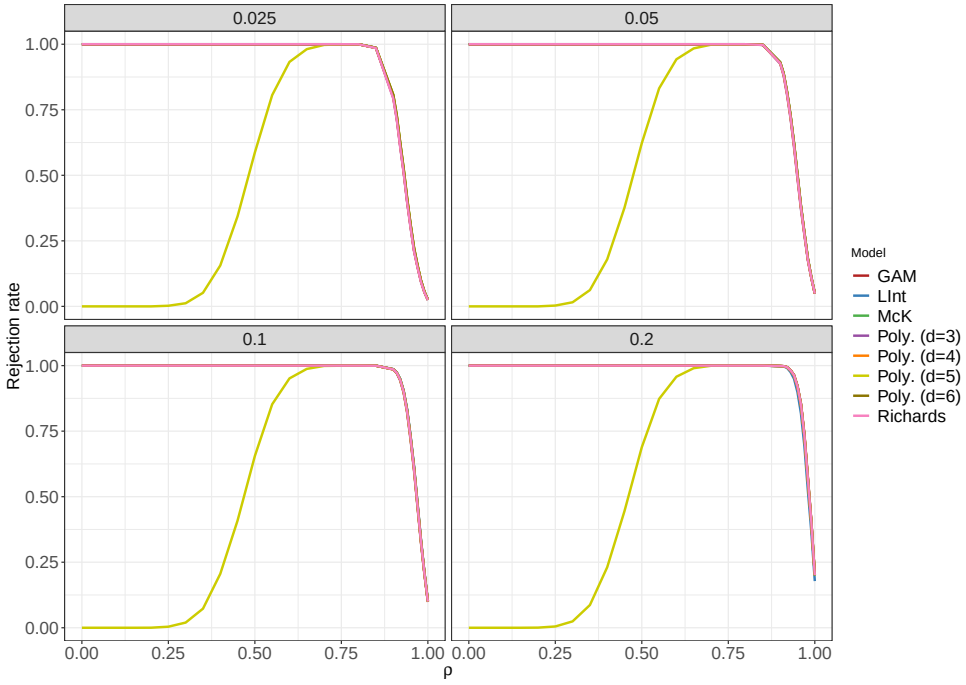
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A2: Share of rejection from 10,000 repetition of unit root test without drift and trend, $n = 57$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



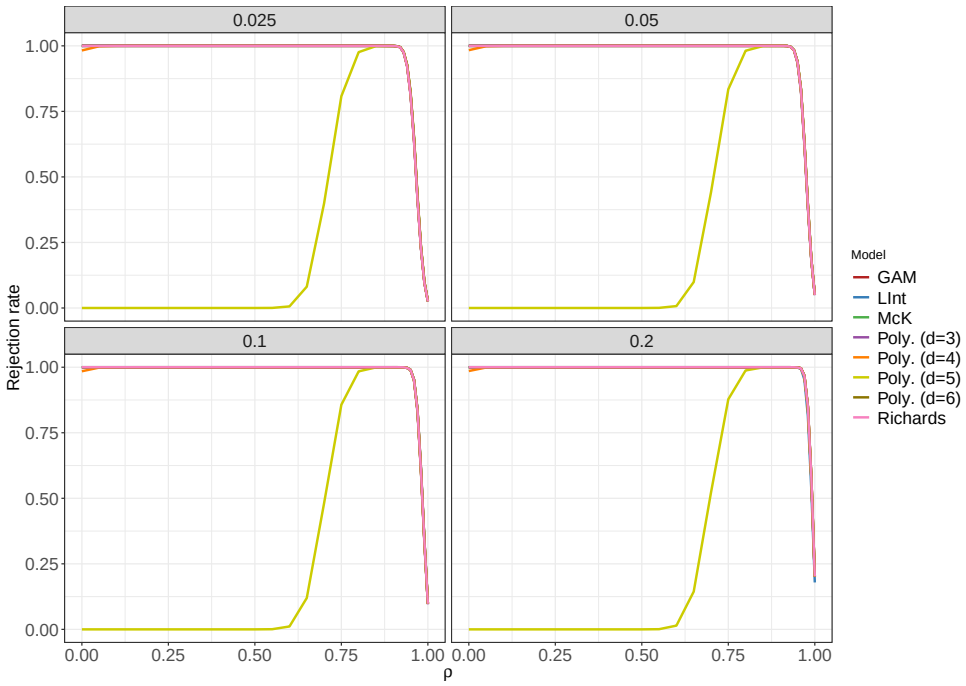
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A3: Share of rejection from 10,000 repetition of unit root test without drift and trend, $n = 130$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



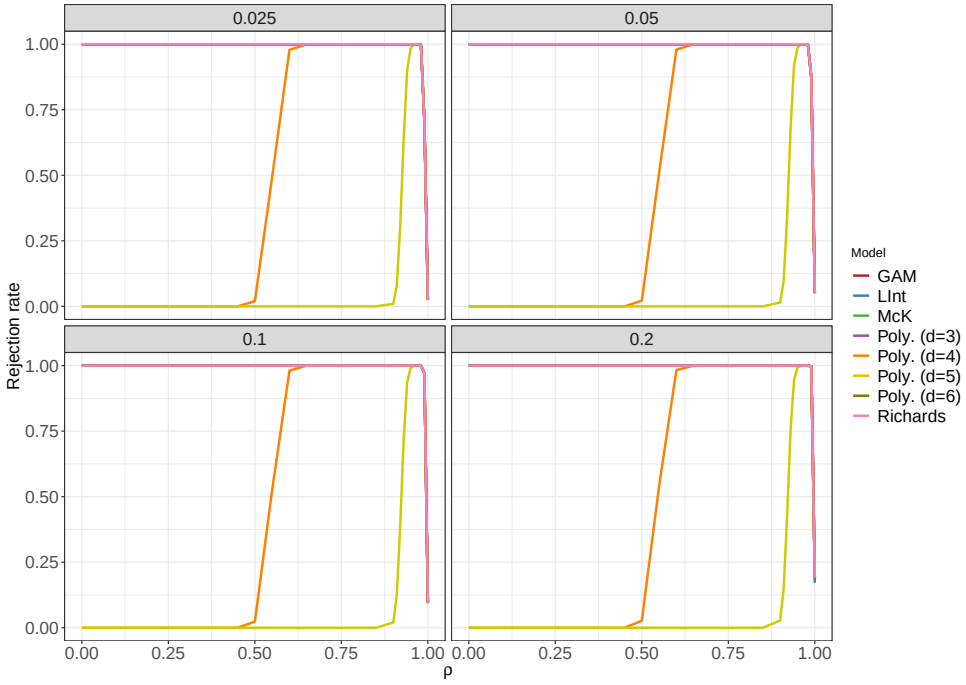
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A4: Share of rejection from 10,000 repetition of unit root test without drift and trend, $n = 265$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

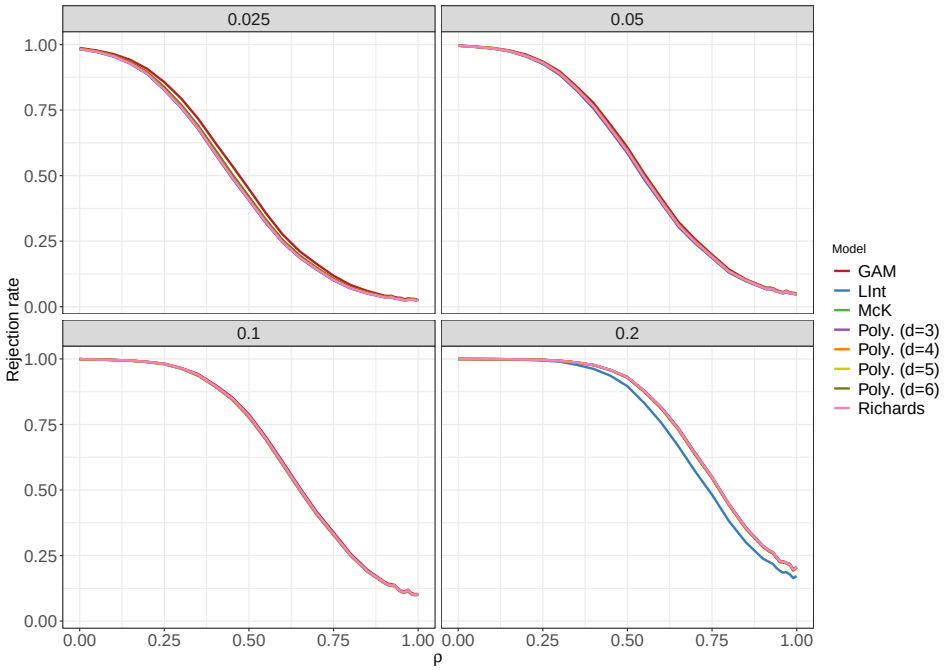
Figure A5: Share of rejection from 10,000 repetition of unit root test without drift and trend, $n = 1150$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

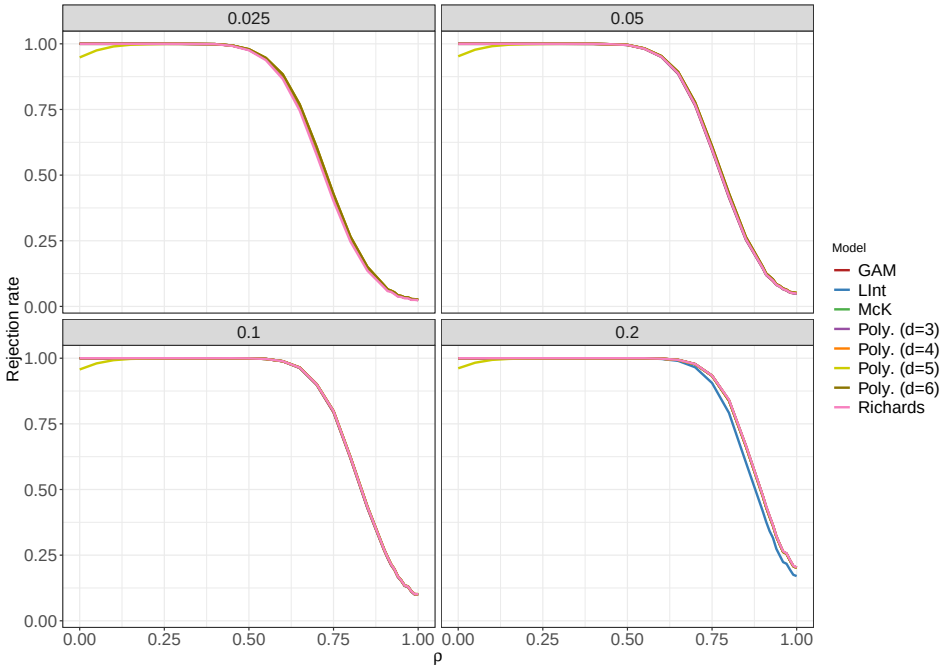
With drift

Figure A6: Share of rejection from 10,000 repetition of unit root test with drift, $n = 27$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



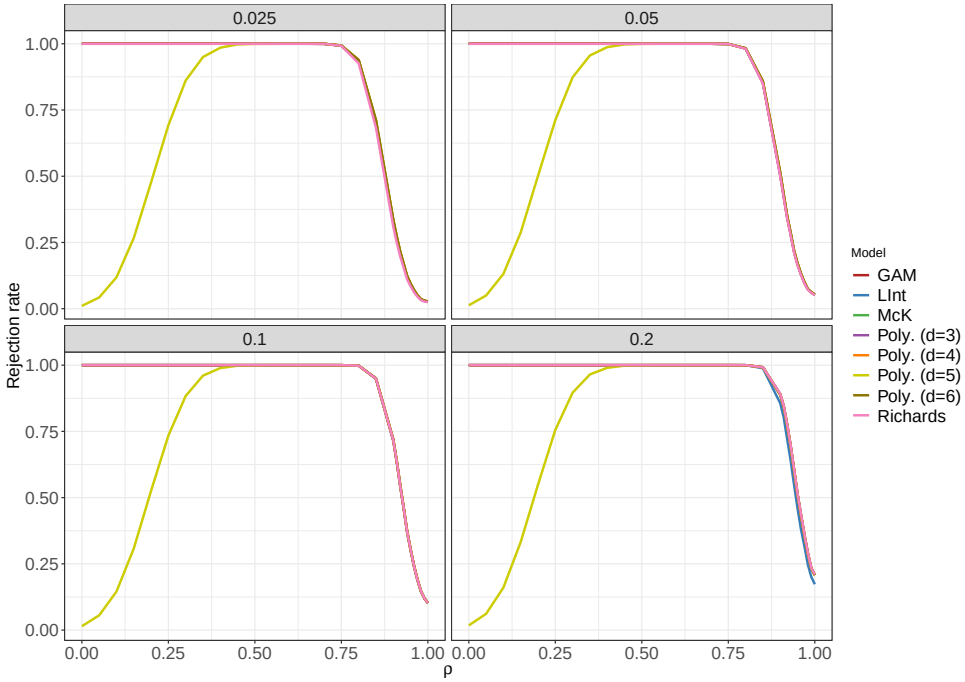
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A7: Share of rejection from 10,000 repetition of unit root test with drift, $n = 57$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



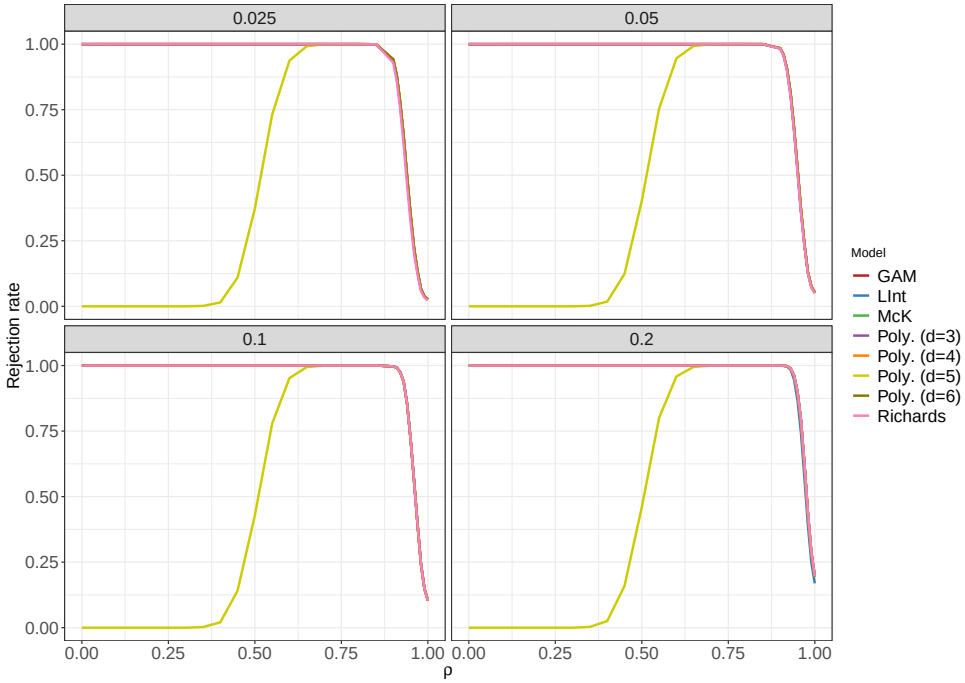
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A8: Share of rejection from 10,000 repetition of unit root test with drift, $n = 130$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



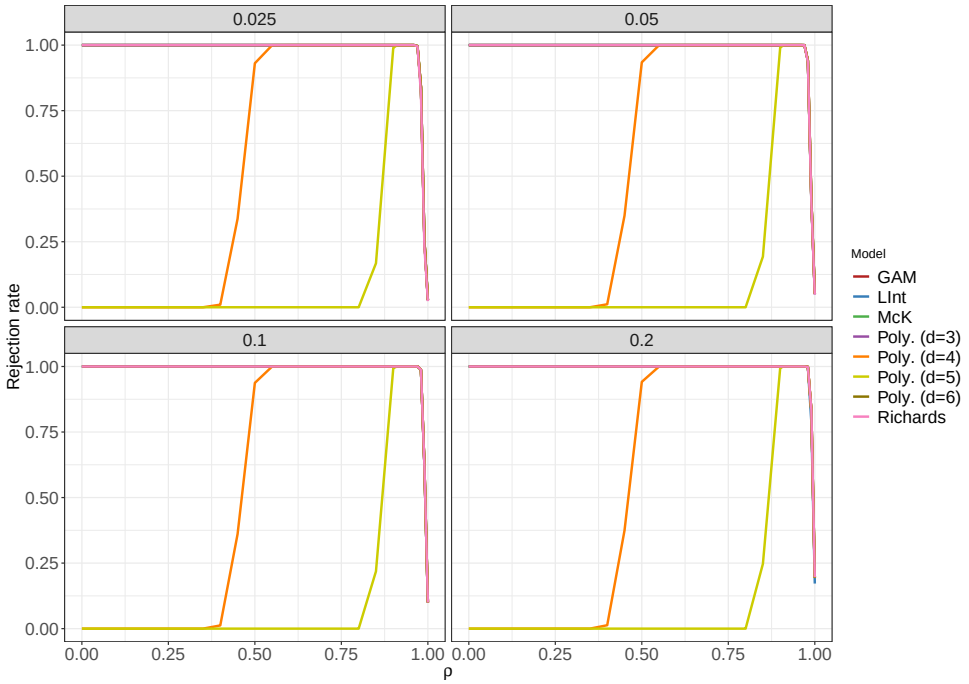
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A9: Share of rejection from 10,000 repetition of unit root test with drift, $n = 265$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A10: Share of rejection from 10,000 repetition of unit root test with drift, $n = 1150$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Polynomial models with limits

Based on the above diagrams, some of the polynomial models reject too few resulting in a discrepancy when compared with other models. This implies that the approximated p-value is larger than it should be. A possible reason is the fact that the model functions are not non-decreasing across the entire of t as seen in the model fit plots in Figure A15 and A16 in the following section (Appendix D). A simple way to remedy this is to set limits so that the model functions are non-decreasing across the whole range of t. The limits are set based on the average minimum and maximum value of t-statistic in the training dataset across all sample size. For test type without drift and trend, the lower and upper limit, rounded to 2 decimal places, is -5.04 and 4.68 respectively. Similarly, the lower and upper limit, rounded to 2 decimal places, for the test with only drift is -5.80 and 3.04. Figure A11 and A12 show the results after applying the limits with $n = 1150$ as an example. Now the rejection shares of the polynomial models are similar to that of the GAM model and MacKinnon's approximation.

Figure A11: Share of rejection of polynomial models for unit root test without drift and trend after setting minimum and maximum limits, $n = 1150$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.

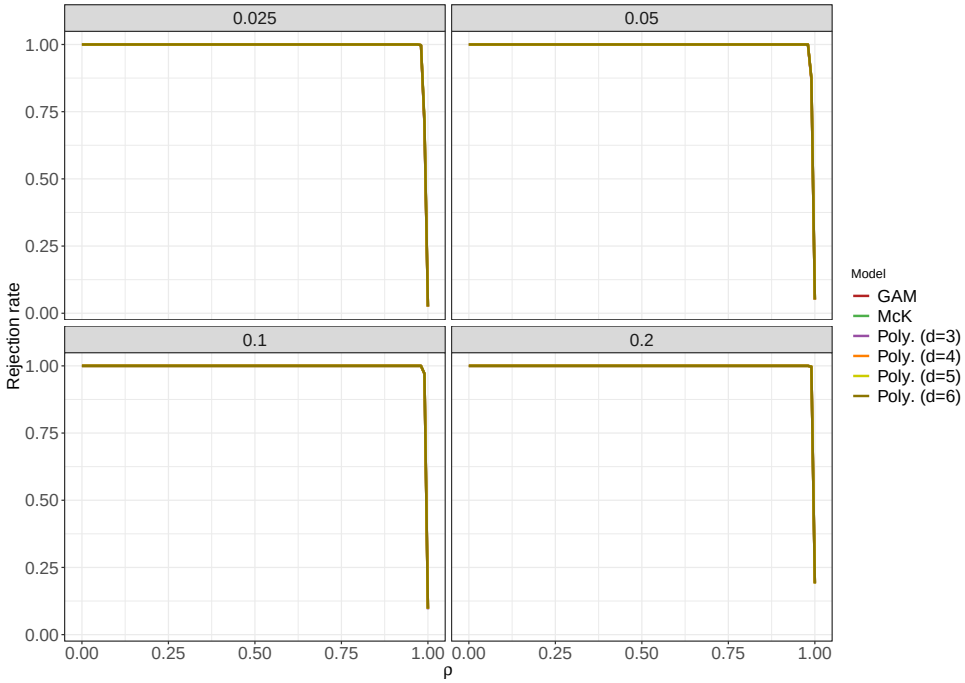
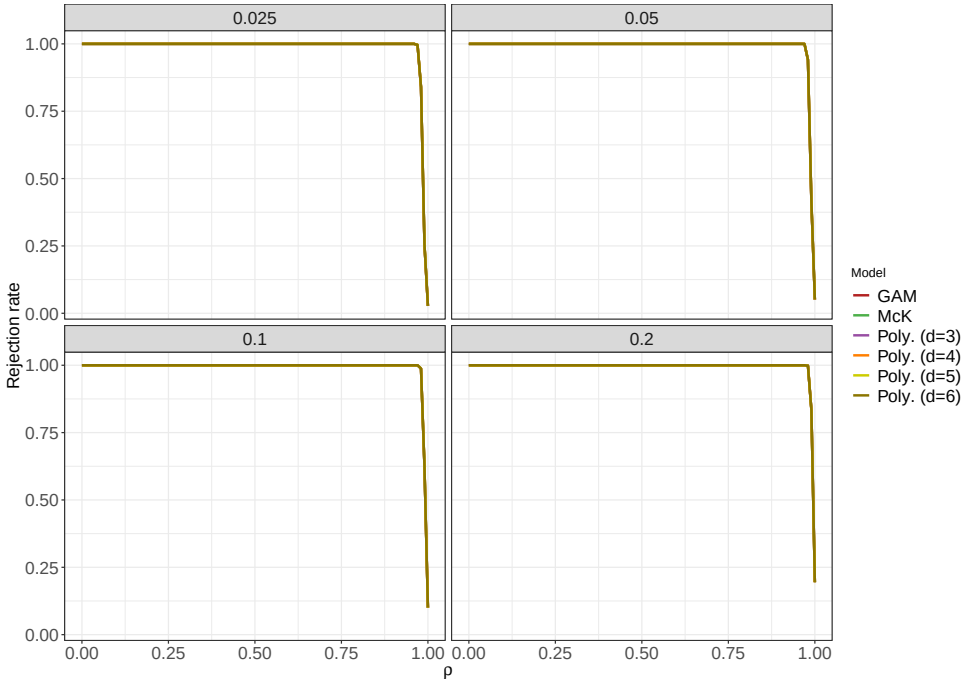


Figure A12: Share of rejection of polynomial models for unit root test with drift after setting minimum and maximum limits, $n = 1150$ and $\alpha \in \{0.025, 0.05, 0.10, 0.20\}$. Source: Own illustration.

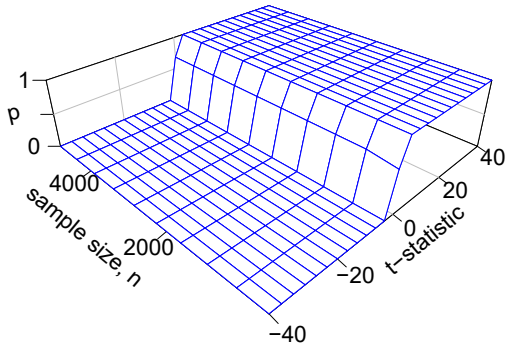


D Plot of Model Fit

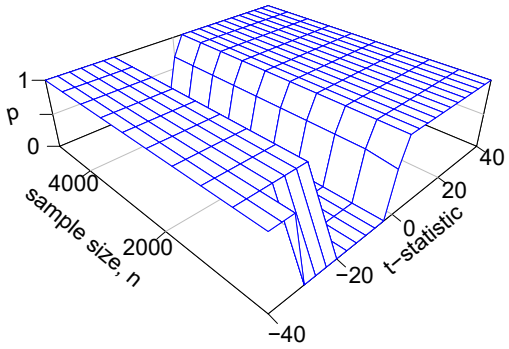
The figures below show the model fits for the unit root test without drift and trend and with only drift. As already noted, attempting to fit globally flexible polynomial models has resulted in model that are non-decreasing only within a restricted range but not across the entire range of t or n. As a result, as $F_n(t) \nrightarrow 0$ as $t \to -\infty$ or $F_n(t) \nrightarrow 1$ as $t \to \infty$.

Figure A13: Model fit for unit test without drift and trend. Source: Own illustration.

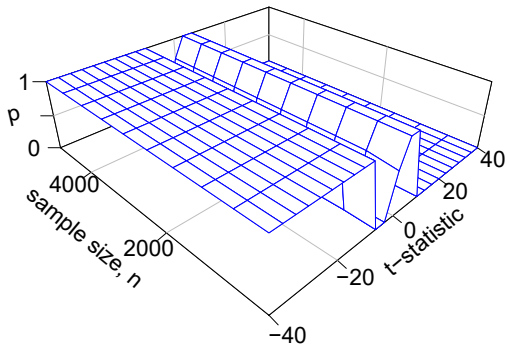
(a) Polynomial ($d = 3$)



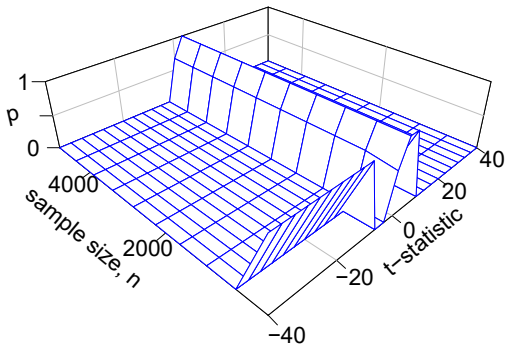
(b) Polynomial ($d = 4$)



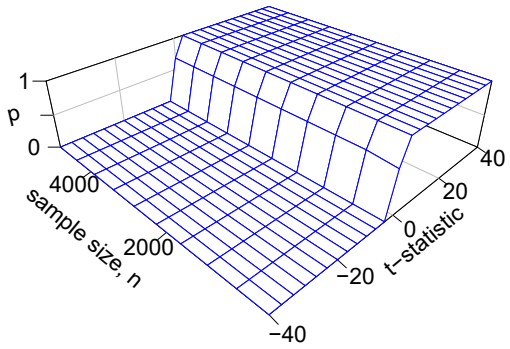
(c) Polynomial ($d = 5$)



(d) Polynomial ($d = 6$)



(e) GAM



(f) Richard's curve

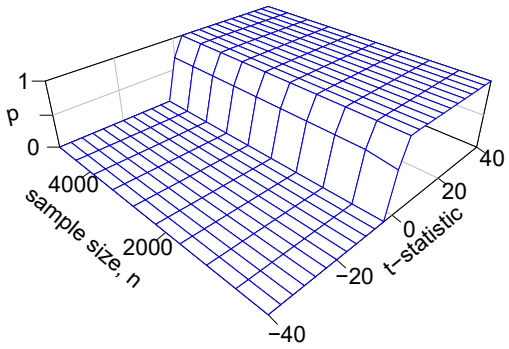
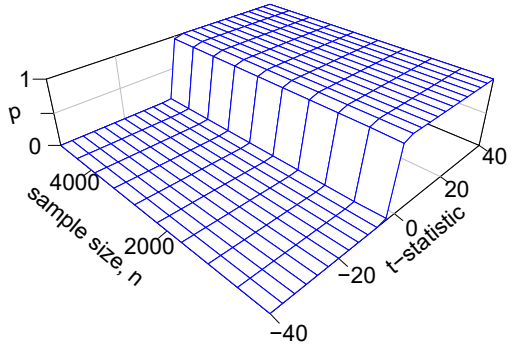
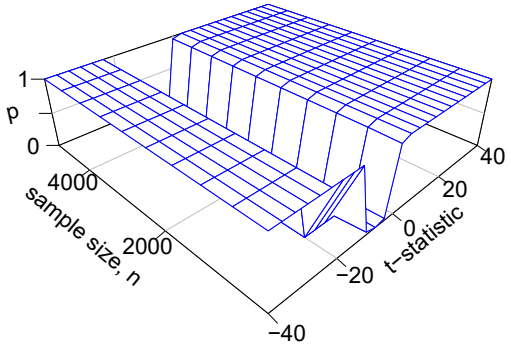


Figure A14: Model fit for unit test with only drift. Source: Own illustration.

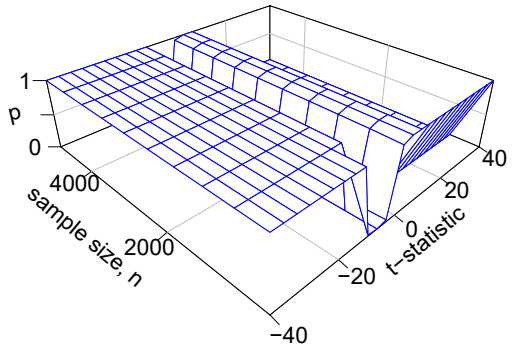
(a) Polynomial ($d = 3$)



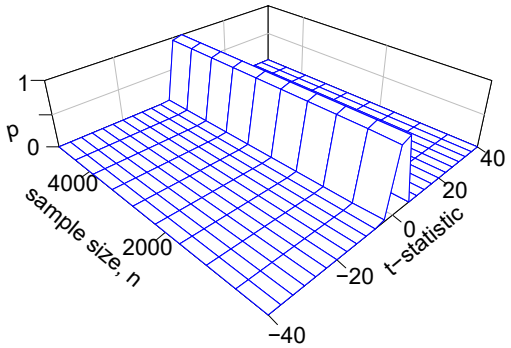
(b) Polynomial ($d = 4$)



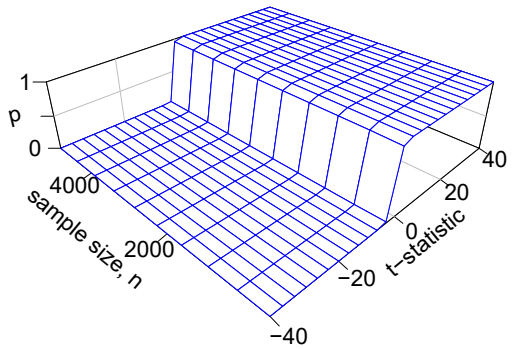
(c) Polynomial ($d = 5$)



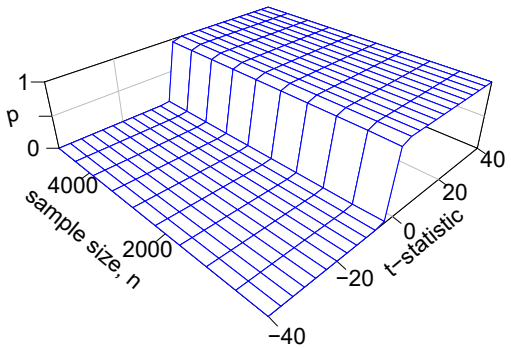
(d) Polynomial ($d = 6$)



(e) GAM



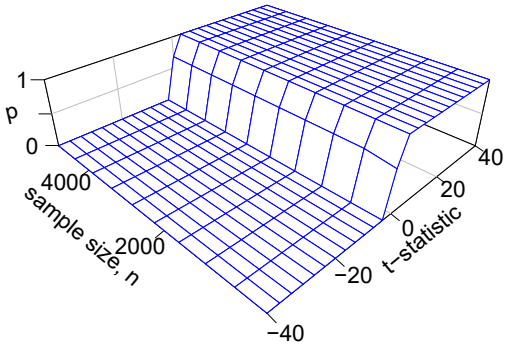
(f) Richard's curve



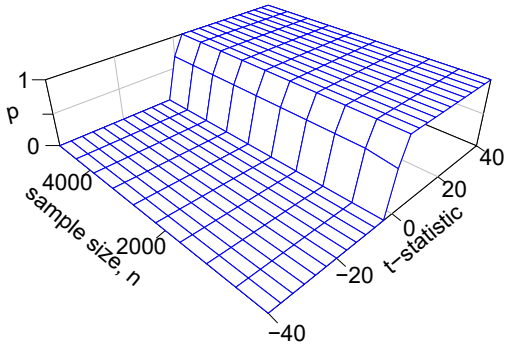
The correction applied to remedy this issue has been to set lower and upper limits on the t-statistics axis based on average minimum and maximum value of t-statistic in the training dataset across all sample size. For test type without drift and trend, the lower and upper limit, rounded to 2 decimal places, is -5.04 and 4.68 respectively. Similarly, the lower and upper limit, rounded to 2 decimal places, for the test with only drift is -5.80 and 3.04. With the limits in place, the polynomial model fits are non-decreasing across the whole range of t, as seen in Figure A15 and A16 below.

Figure A15: Model fit for unit test without drift after setting minimum and maximum limits. Source: Own illustration.

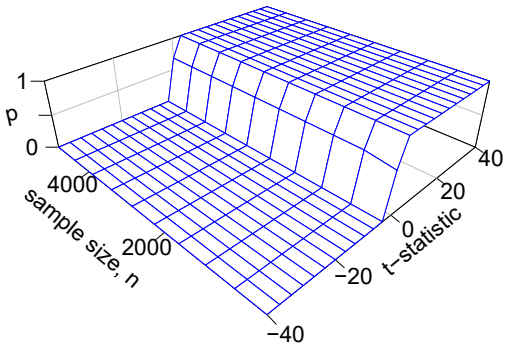
(a) Polynomial ($d = 3$)



(b) Polynomial ($d = 4$)



(c) Polynomial ($d = 5$)



(d) Polynomial ($d = 6$)

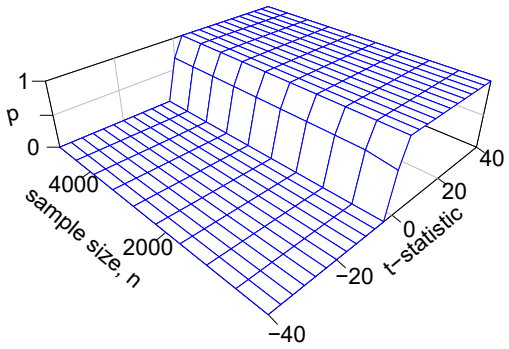
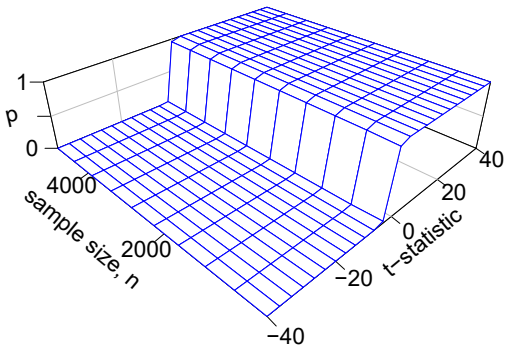
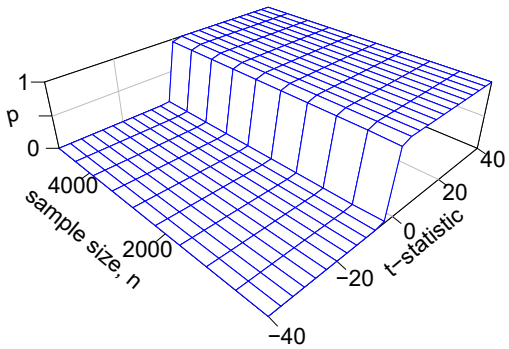


Figure A16: Model fit for unit test with drift after setting minimum and maximum limits. Source: Own illustration.

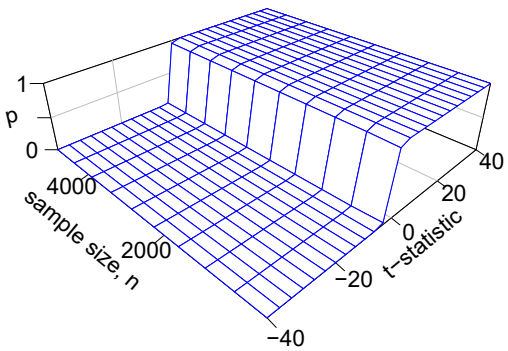
(a) Polynomial ($d = 3$)



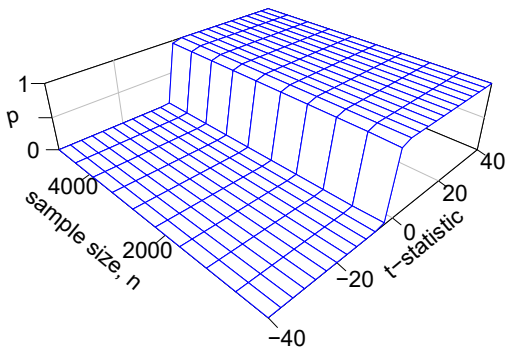
(b) Polynomial ($d = 4$)



(c) Polynomial ($d = 5$)



(d) Polynomial ($d = 6$)

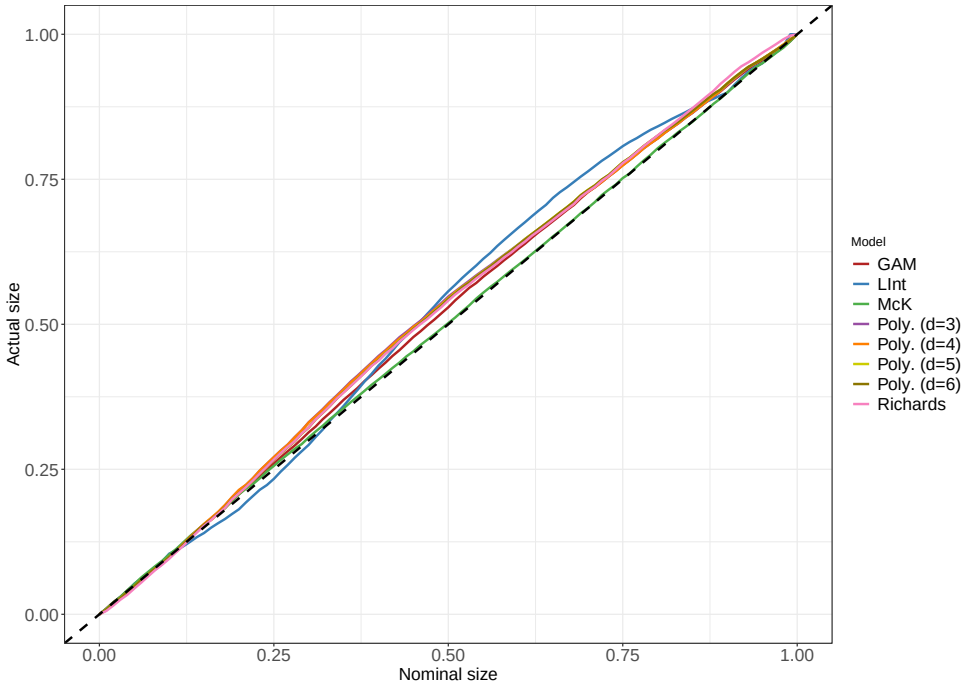


E P-Value Plots

The p-value plots for unit root test without drift and trend and unit root test with drift are shown below. As already described in Section 5, the linear interpolation approach shows behavior in the following plots similar to that in the p-value plots for the test with both drift and trend, i.e. the graph lies close to the reference line in the left and right tail of the distribution, but then under-rejects followed by over-rejection. By comparing with the p-value plots in Section 5, the Richards' curve performs similar to the rest of the models in the absence of linear trends. In the absence of both drift and trend and when the sample size is small ($n = 27$, Figure A17), it appears that all the models, except the MacKinnon's approximation, deviates from the reference and start to over-reject in the middle part of the distribution, with the polynomial and Richards' curve models having larger deviation than GAM. The deviations, however, diminish with increasing sample sizes.

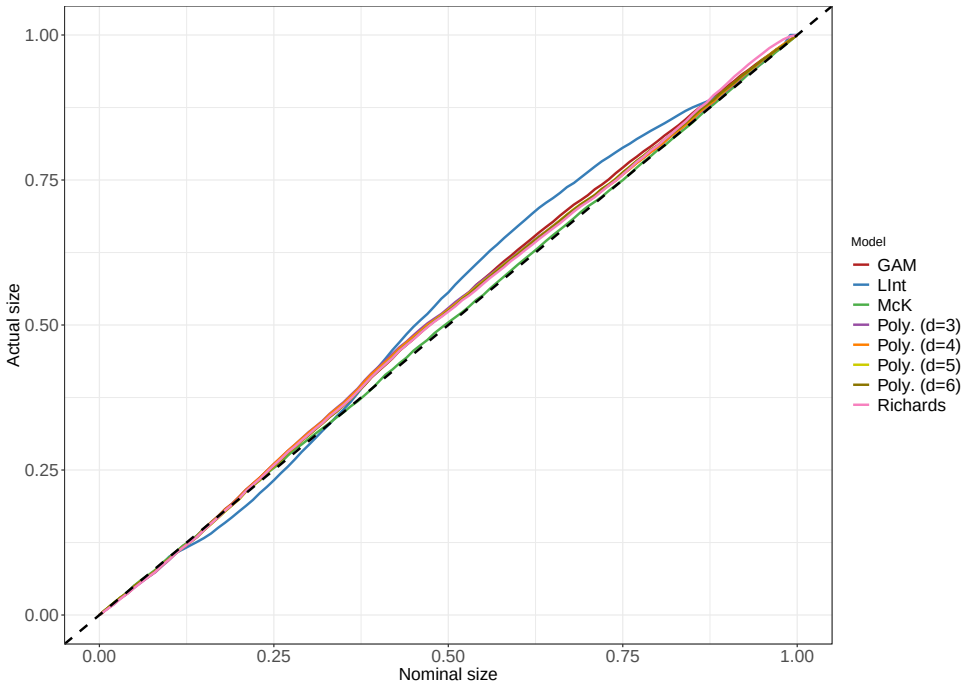
Without drift and trend

Figure A17: P-value plots for unit root tests without drift and trend, $n = 27$. Source: Own illustration.



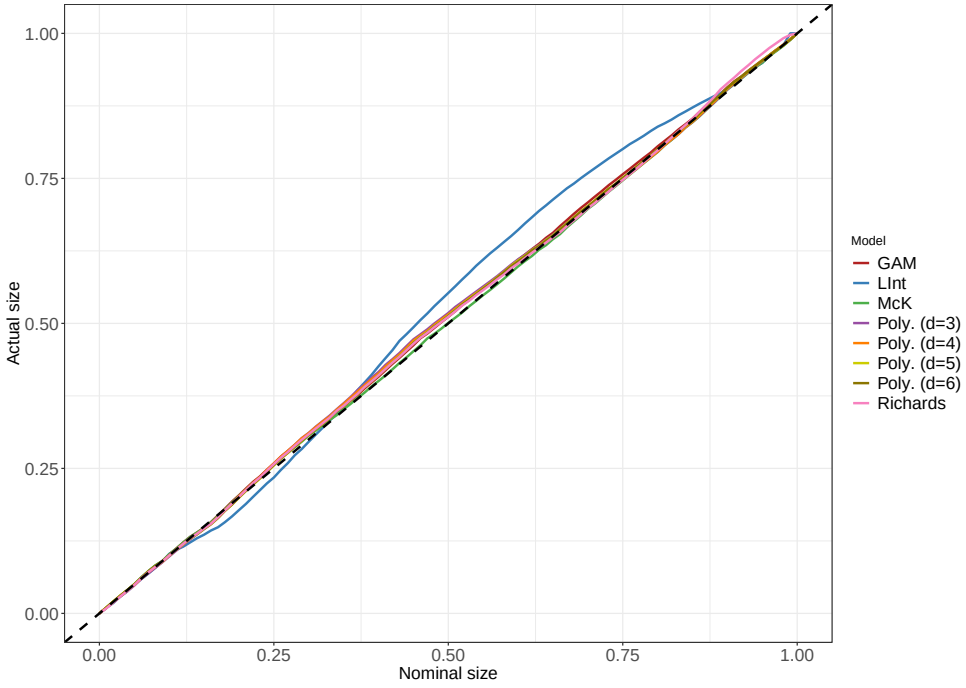
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A18: P-value plots for unit root tests without drift and trend, $n = 57$. Source: Own illustration.



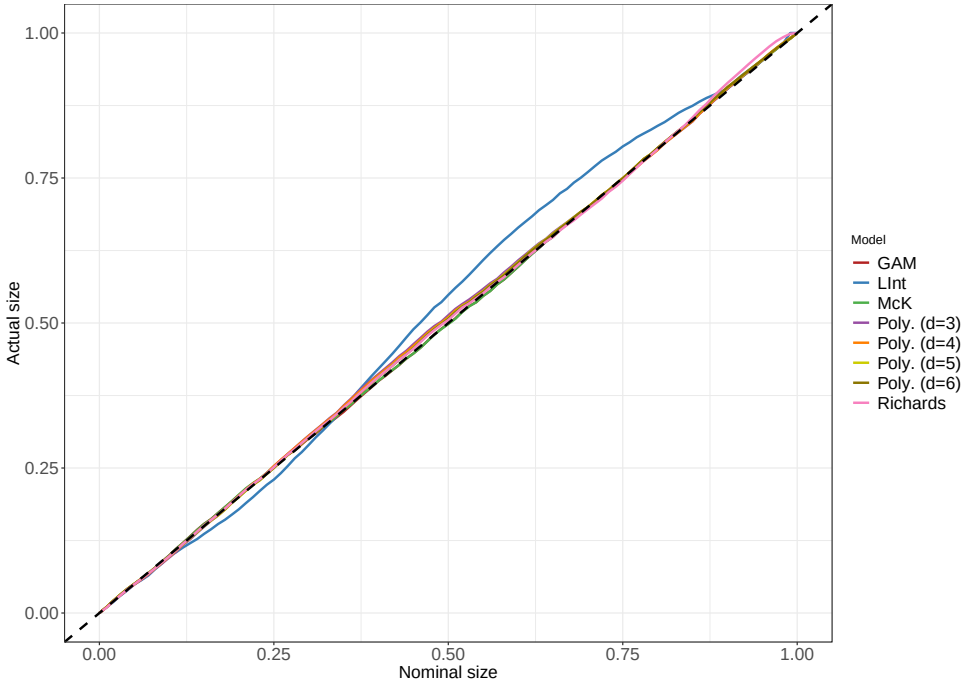
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A19: P-value plots for unit root tests without drift and trend, $n = 130$. Source: Own illustration.



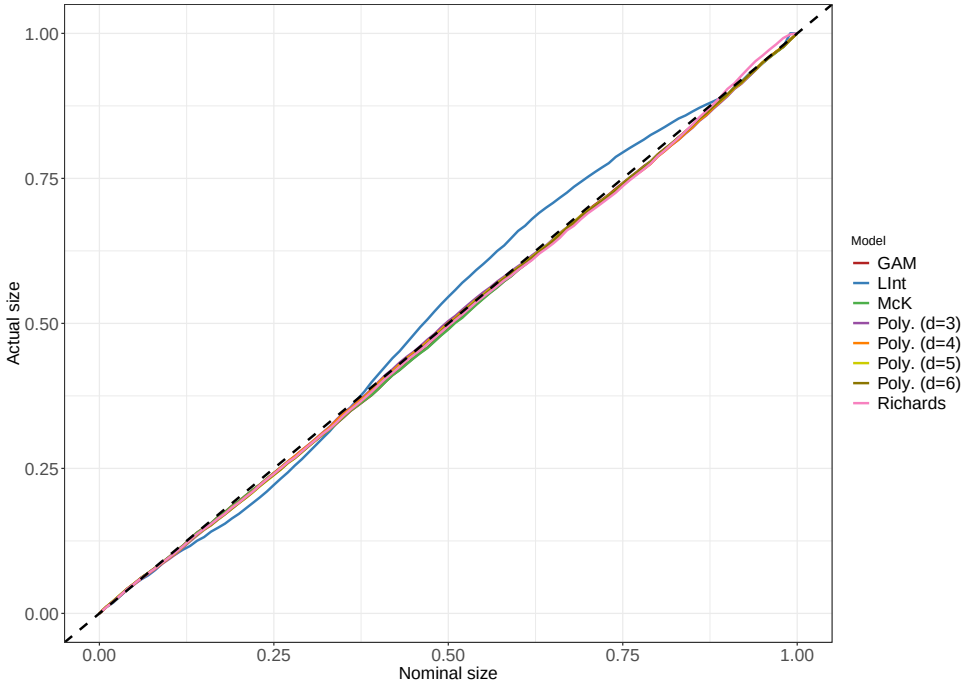
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A20: P-value plots for unit root tests without drift and trend, $n = 265$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

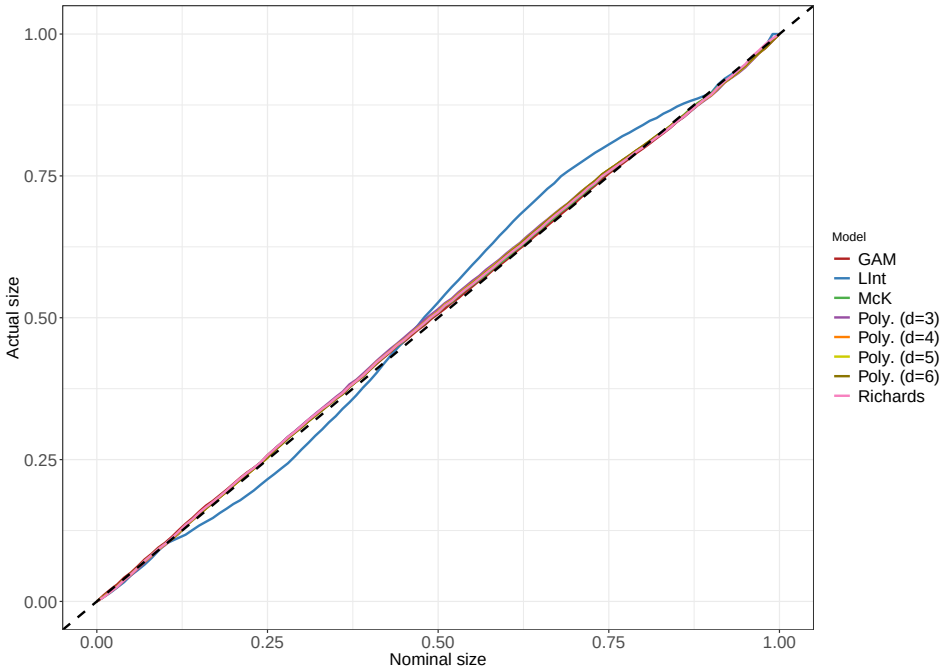
Figure A21: P-value plots for unit root tests without drift and trend, $n = 1150$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

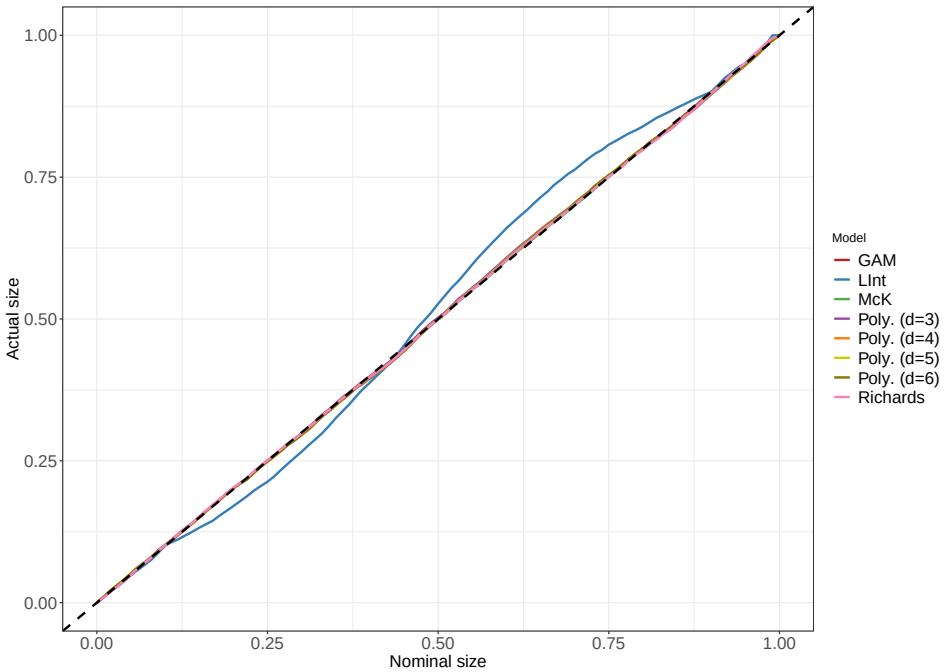
With drift

Figure A22: P-value plots for unit root tests with drift, $n = 27$. Source: Own illustration.



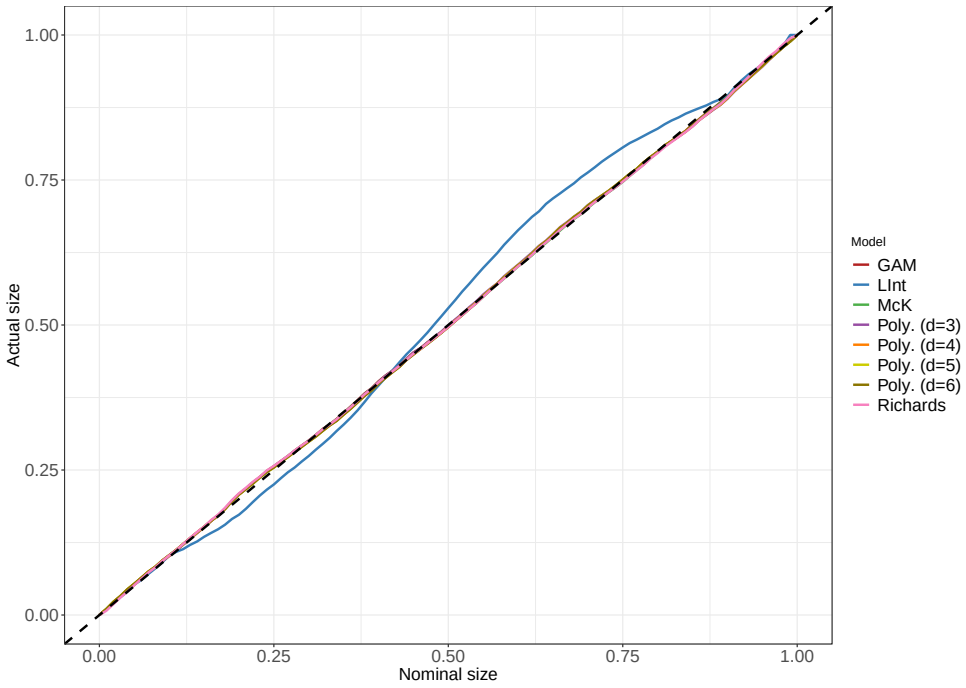
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A23: P-value plots for unit root tests with drift, $n = 57$. Source: Own illustration.



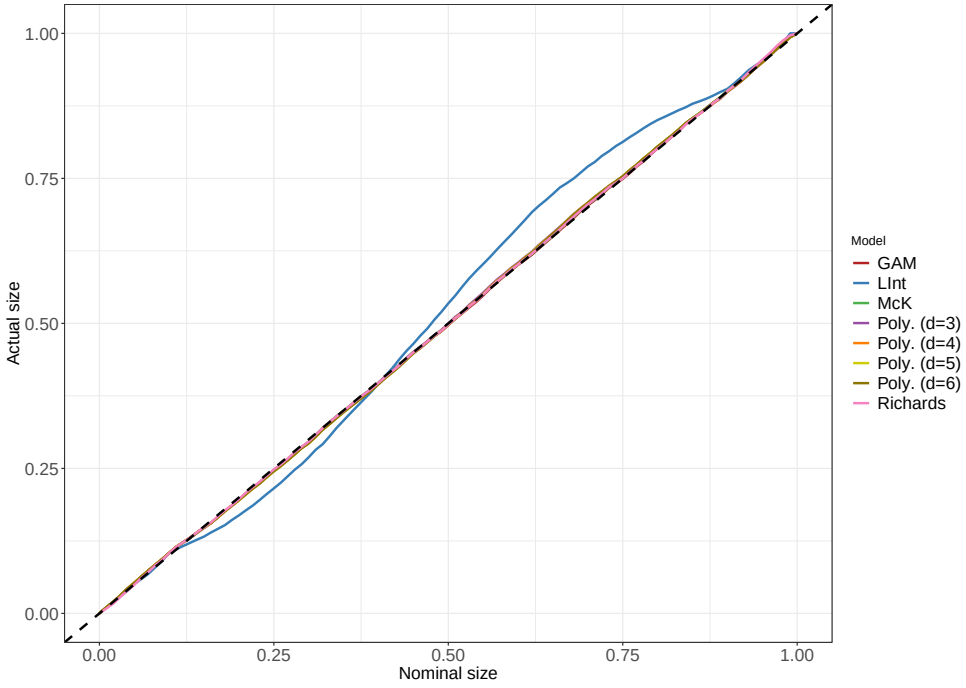
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A24: P-value plots for unit root tests with drift, $n = 130$. Source: Own illustration.



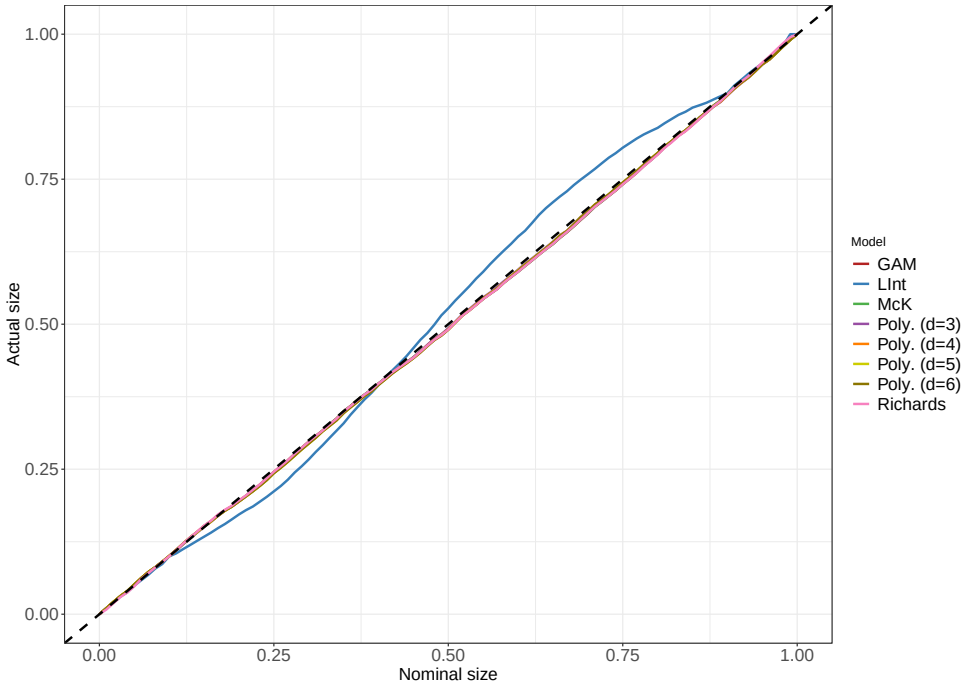
Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A25: P-value plots for unit root tests with drift, $n = 265$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Figure A26: P-value plots for unit root tests with drift, $n = 1150$. Source: Own illustration.



Keys: Poly: Polynomial, Richards: Richards' curve, LInt: Linear interpolation based on critical values table from Dickey (1976, p. 53) and Fuller (1996, p. 642), McK: Approximation based on MacKinnon's unit root p-values (MacKinnon, 1996).

Package 'pvurt'

September 13, 2019

Title P-value for augmented Dickey-Fuller unit root test

Version 1.0.0

Description Approximate p-value for augmented Dickey-Fuller unit root test.
The package has been created as part of the author's master thesis.

Depends R (>= 3.5.0)

License GPL-3

Imports mgcv,
stringr,
urca,
timeSeries

Suggests fUnitRoots,
tseries

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

R topics documented:

`commons` *Common functions*

Description

Common functions used by top level functions.

Usage

```
computePValueMain(object, n, model = c("gam", "poly"), type = c("nc",
  "c", "ct"), d = NULL)

modelName(type, model, d)
```

Arguments

object	Numeric value or an object (fHTEST, ur.df or htest) for which p-value needs to approximated.
n	Sample size.
model	The model type to be used for approximation. Available is GAM and polynomial regression. If gam is chosen, then d has no effect.
type	The type of unit root test. Currently supports: nc for test without drift and trend, c for test with only drift and ct for test with both drift and trend.
d	The degree for polynomial. d must be ≥ 3 and ≤ 6. If gam is chosen, then d has no effect.

Details

modelName generates the name of function based on the arguments model, type and d (when model = "poly").
computePValueMain is called by the other top level functions to approximate the p-value.

---

computePValue *P-value*

Description

Function to approximate the p-value for augmented Dickey-Fuller test.

Usage

```
computePValue(object, ...)

## S3 method for class 'numeric'
computePValue(object, n, type = c("nc", "c", "ct"),
  model = c("gam", "poly"), d = NULL, ...)

## S3 method for class 'fHTEST'
computePValue(object, model = c("gam", "poly"),
  type = c("nc", "c", "ct"), d = NULL, ...)

## S3 method for class 'ur.df'
computePValue(object, model = c("gam", "poly"),
  d = NULL, ...)

## S3 method for class 'htest'
computePValue(object, model = c("gam", "poly"),
  d = NULL, ...)
```

Arguments

object	Numeric value or an object (fHTEST, ur.df or htest) for which p-value needs to approximated.
...	Further arguments passed to methods.

n	Sample size.
type	The type of unit root test. Currently supports: nc for test without drift and trend, c for test with only drift and ct for test with both drift and trend.
model	The model type to be used for approximation. Available is GAM and polynomial regression. If gam is chosen, then d has no effect.
d	The degree for polynomial. d must be ≥ 3 and ≤ 6. If gam is chosen, then d has no effect.

Details

Based on the chosen model (GAM or polynomial), the function returns the approximated p-value. Default is GAM model.

Examples

```
library(pvurt)
y <- arima.sim(model = list(order = c(0, 1, 0)), n = 100)

# Test type: with drift and trend
# package: fUnitRoots
library(fUnitRoots)
computePValue(adfTest(y, lags = 3, type = "ct"))
computePValue(unitrootTest(y, lags = 3, type = "ct"))

# package: urca
library(urca)
computePValue(ur.df(y, lags = 3, type = "trend"))
# print summary
summary(computePValue(ur.df(y, lags = 3, type = "trend")))

# package: tseries
library(tseries)
computePValue(adf.test(y, alternative = "stationary", k = 3))

# no packages
tStat <- -2.239
sampleSize <- 100
computePValue(tStat, n = sampleSize, model = "gam", type = "ct")
```

---

models *Models*

Description

Models for approximating p-value.

Usage

```
woTD_poly3(t, n)

woTD_poly4(t, n)
```

```
woTD_poly5(t, n)

woTD_poly6(t, n)

woTD_gam(t, n)

wD_poly3(t, n)

wD_poly4(t, n)

wD_poly5(t, n)

wD_poly6(t, n)

wD_gam(t, n)

wTD_poly3(t, n)

wTD_poly4(t, n)

wTD_poly5(t, n)

wTD_poly6(t, n)

wTD_gam(t, n)
```

Arguments

t	Test statistics
n	Sample size

Details

These are the functions based on polynomial regression and GAM to approximate the p-value for ADF unit root tests of three types: without drift and trend (woTD), with drift (wD) and with both drift and trend (wTD).

Value

p-value from the ADF test statistic distributions.

Index

Eidesstattliche Versicherung

Ich versichere an Eides statt durch meine Unterschrift, dass ich die vorstehende Arbeit selbständig und ohne fremde Hilfe angefertigt und alle Stellen, die ich wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen habe, als solche kentlich gemacht habe, mich auch keiner anderen als der angegebenen Literatur oder sonstiger Hilfsmittel bedient habe. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen.

Düsseldorf, September 17, 2019

.....

Mehdi Belayet Lincon