Summer 7-25-2019

One and Two-Step Estimation of Time Variant Parameters and Nonparametric Quantiles

Bogdan Gadidov

One and Two-Step Estimation of Time Variant Parameters and Nonparametric Quantiles

A Dissertation

In partial fulfillment of the requirements for the

degree of Doctor of Philosophy in Analytics and Data Science

The Graduate College

Kennesaw State University

Bogdan Gadidov

Kennesaw State UNIVERSITY
Graduate College

Thesis/Dissertation Defense Outcome

Name	Bogdan Gadidov	KSU ID	000515651
Email	bgadidov@students.kennesaw.edu	Phone Number	678-768-0318
Program	Ph.D. in Analytics and Data Science		

Title
One and Two-Step Estimation of Time Variant Parameters and Nonparametric Quantiles

Thesis/Dissertation Defense: Date 7/12/2019

[✓ Passed] Failed Passed With Revisions (attach revisions)

Signatures

[illegible]	7/12/2019
Thesis/Dissertation Chair	Date
Lewis VanBrackle	7/12/2019
Committee Member	Date
Joe DeMaio	7/12/19
Committee Member	Date
Xuan Huang	7/12/2019
Committee Member	Date
Committee Member	Date
[illegible]	7/12/2019
Program Director	Date
[illegible]	7/12/19
Department Chair	Date
Graduate Dean	Date

Last Modified 05/26/16

Abstract

This dissertation develops and discusses several one-step and two-step smoothing methods of time variant nonparametric quantiles and time variant parameters from probability models. First, we investigate and develop nonparametric techniques for measuring extreme quantiles. The method involves aggregating data by an explanatory variable such as time and smoothing the resulting data with a nonparametric method like kernel, local polynomial or spline smoothing. We demonstrate both in application and simulation that this two-step procedure of quantile estimation is superior to the parametric quantile regression. We then develop a one-step method which combines the strength of maximum likelihood estimation with a local kernel function. This local maximum likelihood estimation is applied in both a discrete and continuous case of distribution, and we consider polynomial expansions of the unknown parameter in each case. In the continuous case, we choose a distribution with two parameters and iteratively solve for each to smooth the data. Results indicate that the one-step procedure can yield improvement over the corresponding two-step methods mentioned previously in both application cases and simulation exercises. We also explore nonparametric techniques for estimating volatility of financial data. We develop a residual based method for estimating the conditional variance function using local composite quantile regression, and compare this to using local least squares regression. These methods are applied on the asset returns for many individual firms, with promising results in favor of local composite quantile regression. Comparisons of these nonparametric techniques in forecasting also indicate some improvement over using a traditional autoregressive model for heteroscedastic data.

Dedication

To my parents, Anda and Radu, and grandmothers Solange and Boba.

Acknowledgments

I would like to express my deepest appreciation to my committee members. Firstly my chair, Dr. Mohammed Chowdhury, who has advised me over the past 2 years. I am very thankful for the research we have done together and for his continued input and knowledge in all our work together. Next, for Dr. VanBrackle, whose constant wisdom and advice over the years has always been greatly appreciated. I would also like to thank Dr. Joe DeMaio for his input and advice on this research in addition to guidance on other projects together. Lastly, thank you to Dr. Xiao Huang, who greatly helped me in the research of the final chapter of this dissertation. Thank you for the constant advisement and research ideas.

I would also like to thank two of my colleagues, Linh Le and Yan Wang, who shared in contributions to the works of the first chapter of this dissertation.

Finally, thank you to Dr. Jennifer Priestley, who put together this Ph.D. program and made this all possible. Her constant willingness to always look out for my well-being, advise and collaborate on research, and assist in job searches is most appreciated.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Nonparametric regression differs from its parametric counterparts in that the relationship between the response and predictor variables are not predetermined. Instead, nonparametric techniques allow for the data to dictate the form of the regression, adapting to the shape of the data, without making assumptions about the form of the true regression function. Many of the still popular techniques used today began over 50 years ago with the development of local estimation methods. In this paper, we study some of these methods and apply them in new scenarios.

Kernel smoothing, local polynomial smoothing and spline smoothing are very popular techniques for smoothing a smaller to moderate sized data under nonparametric regression settings. Nadaraya-Watson (1964) first developed and used the Kernel smoothing estimation and since then it has been used in many applications, such as kernel density estimation (Silverman, 1986 and Scott, 1992), kernel smoothing estimation of unknown functions (Hart and Wehrly, 1986), kernel smoothing estimation of distribution functions (Chowdhury, Wu and Modarres, 2017, 2018), estimation of time-varying coefficient models by kernel estimator (Hoover, Rice, Wu and Yang, 1998) etc. Local polynomial smoothing was first studied by Stone (1977, 1980, 1982) and Cleveland (1979) and then by Fan (1992, 1993), Fan and Gijbels (1992, 1996) and Ruppert and Wand (1994) and others. Fan and Gijbels (1991) show that the local linear smoother performs equally well at the boundary points as at interior points, improving upon previous kernel regression methods. Smoothing splines are studied by many authors, such as Schoenberg (1964), Reinsch (1967), Wahba and Wold (1975), and Silverman (1985), to name a few. Eubank (1988) gives a good review of spline methods and Wahba (1990) provides a complete theoretical treaty. These estimators are widely used when the parametric form of the response variable is unknown, size of the data is not big, and data does not have significant variation by time points.

These nonparametric techniques entail performing local estimation on the data, resulting in the need for the selection of a bandwidth. The bandwidth serves as the parameter which dictates the size of the interval in which the local estimation takes place. Hence, it also affects the smoothness of the fitted regression model; small choices for bandwidth result in spiky, rugged estimates while a large bandwidth can give undersmoothed, almost linear estimates. There are many methods for selecting the bandwidth in the various nonparametric techniques. The most intuitive method for selecting the bandwidth is quite possibly using cross-validation, where a subset of the data is iteratively left out from model fitting and the error produced on the excluded data is then measured (see Stone 1974 and Geisser 1975). The resultant bandwidth choice is then the value which produces the least mean error on the out of sample data. Many variations exist for cross-validation, such as when dealing with dependent data (Hart and Vieu 1990) or outliers (Leung 2005). Xia and Li (2002) propose a method where a weighting function is added to the cross-validation score, to help improve estimation in the boundary regions of the interval of data. See Arlot and Celisse (2010) for an overview of cross-validation procedures. As an alternative to cross-validation, Ruppert and Wand (1995) give some plug-in bandwidth choices. The idea behind these methods are to find the bandwidth to minimize the mean integrated square error, which require estimating the mean and variance functions by dividing the data into blocks and fitting quartic regression functions. Building off of Ruppert, Francisco-Fernández (2004) proposes a similar bandwidth selection process when dealing with nonparametric regression where the error terms may be autocorrelated. In cases where more variables are used, Li and Racine (2004) present a bandwidth selection method for a multivariate model, so each predictor has its own bandwidth. This method is tested for both discrete and continuous data types. In some cases, a constant bandwidth for the entire interval of the data does not capture the true nature of the data, so a variable bandwidth might be desirable in these scenarios. Fan and Gijbels (1992) present asymptotic properties for the optimal variable bandwidth. Schucany (2004) discusses a method of recursive partitioning for selecting a variable bandwidth. Regression trees can be used to determine the regions of the predictor variable where the bandwidth should change. Giordano and Parrella (2014) proposed a hybrid bandwidth selection method, Global Adaptive Smoothing, which combines both local and global bandwidth selection. The optimal bandwidth is created by estimating a global bandwidth on a small interval surrounding a local point in a rolling window procedure. Unlike other plug-in methods, Giordano uses a feedforward neural network to minimize the mean integrated square error. In this paper, we employ leave one out cross-validation and propose a leave one time period out cross-validation for our estimation purposes. We also explore one of the bandwidth selection methods from Ruppert and Wand in the last part of this paper.

The paper is organized as follows. Chapter 2 begins with the development and implementation of the two-step nonparametric techniques on various quantiles. To implement the nonparametric methods on smoothing cross-sectional data with many observations at each time point, we must first take an empirical approach to create an estimate at each time point, which we will then smooth over time. We refer to the first step in this procedure as creating the raw estimate of our unknown value, and the second step is then applying the smoothing estimator such as kernel, local polynomial or spline. For example, in a case where one is interested in estimating a specific quantile, one could first empirically take the original data and estimate this quantile at each time period and then smooth the resulting estimates. In the case of smoothing the mean of the data, we can similarly create the raw estimates to be the mean of each time period, and smooth the resulting estimates. The two-step smoothing procedure, adopted by Fan and Zhang (2000), Wu and Tian (2013a, 2013b), Chowdhury, Wu and Modarres (2017) and Chowdhury (2017), is computationally simple and easy to implement in practice.

One clear downfall to this two-step approach is that much of the original data is lost when creating the raw estimates. As a solution, we also propose a one-step technique which uses all of the original data and creates smoothing estimates without first calculating the raw estimates. If data at each distinct time period can be approximated to a parametric structure, the proposed one-step procedures can be implemented. Local log-likelihood estimation is discussed by Fan and Gijbels (1998) with an application to logistic regression. Our one-step procedure relies on a combination of maximum likelihood estimation and a kernel weighting function. For the one-step procedures, we solve for the parameter of interest by also considering a polynomial expansion of the parameter. We implement and compare the two-step to one-step techniques on both discrete and continuous data and in simulation studies under two different frameworks. In the discrete case, we choose a dataset on fertility rates which tracks the number of months after marriage until first conception. To apply the one-step procedure to this data, we model the number of months until conception as a geometric distribution where we try to estimate the probability of conception for women aged 13 to 31. For the case of a continuous distribution, we model a Londonair dataset measuring the oxides of nitrogen in the air of a London borough. We model these with a normal distribution to estimate the weekly mean level of oxides of nitrogen for a span of one year. Since the normal distribution has two parameters (the mean and standard deviation), we implement the one-step procedure under two scenarios: first where the standard deviation is known and fixed at each time point and second where it is estimated iteratively with the mean. In the latter case, a polynomial expansion of both parameters is considered. In Chapter 3, we develop the first of the one-step procedures, with an application to a discrete distribution. Chapter 4 then continues with the second of the one-step procedures in a continuous case.

Lastly, in Chapter 5, we explore several nonparametric techniques for estimating the conditional variance function. Nonparametric estimation for the mean function is well studied. Fan and Yao (1998) proposed a method for modeling the conditional variance (or volatility) even when the mean function is not given. Their residual based estimator is a two step procedure where the mean is first estimated with local least squares regression, and the squared residuals are then used to model the volatility through another round of local least squares. We propose to replacing local least squares regression with local composite quantile regression (proposed by Kai and Li in 2010) in the above process to estimate volatility. Results indicate that the proposed method can lead to improvements in forecasting and outperform GARCH models for volatility estimation in some circumstances.

Chapter 2

Nonparametric Quantile Estimation

2.1 Quantiles and Quantile Regression

Let $F_{Y_{t_j i}}(\cdot)(j = 1, 2, \ldots, J, i = 1, 2, \ldots, n_j)$ be a time-variant distribution function. $Y_{t_j i}$ stands for i^{th} observation of the t_j^{th} year. More specifically, if $i = 79$ and $j = 11$, then $Y_{t_{11}79}$ stands for the 79^{th} observation of the 11^{th} year. A value for extreme quantile η for each t is estimated as

$$\tilde{\xi}_\eta(t_j) = inf\{y_{t_j i} : F(y_{t_j i}) \geq \eta\} = F^{-1}(\eta)$$

where infimum is running over i and $\eta \in (0, 1)$. When $F(\cdot)$ does not belong to any parametric family, we can use empirical version of $F(\cdot)$ to compute $\xi_\eta(t_j)$. We consider $\eta = 0.95$ and $\eta = 0.05$ in our application to US temperature data.

The quantile regression estimator for quantile η minimizes the objective function:

$$Q(\beta_q) = \sum_{i:y_i \geq x_i' \beta} \eta |y_i - x_i' \beta_q| + \sum_{i:y_i < x_i' \beta} (1 - \eta)|y_i - x_i' \beta_q| \qquad (1)$$

This nondifferentiable function is minimized via the simplex method, which is guaranteed to yield a solution in a finite number of iterations.

2.2 Two-Step Estimation

The basic principle of two-step estimation is that the data set can be split by some variable such as time or age, which is regarded as the explanatory variable. For each split data, unknown statistical constants of interest are estimated (Method of Moments, MLE, Bayesian Methods) or computed empirically, which are called the raw estimates and in the second step, we smoothed them. We derive here two-step smoothing methods by first computing the raw estimates of $\xi_\eta(t_j)$ for all $j = 1, \ldots, J$, and then derive the smoothing estimates of $\xi_n(t)$ for any $t_j \in \mathbf{t}$ by applying the smoothing procedure over the corresponding raw estimates. This two-step smoothing approach is computationally simple and can be used for both longitudinal data and time-variant cross sectional data. For cross-sectional data, this procedure does not need correlation assumptions across different time points and for longitudinal data the correlation between time points would be negligent if the repeated measurements appear in a manner of random long distant time points.

We derive the estimators $\tilde{\xi}_n(t_j)$ of $\xi_n(t_j)$ using observations at time $t_j \in \mathbf{t}$. Suppose that we have enough observations n_j at t_j, so that $\xi_n(t_j)$ can be estimated by some statistical methods using the subjects in $\mathcal{S}_j$. The number of observations n_j at t_j should be sufficiently large to be computed numerically. In the event where the local sample size n_j is not sufficiently large, adjacent time points can be binned and the raw estimates are then computed for each bin. This binning approach has been used by Fan and Zhang (2000), Chowdhury (2017), Chowdhury, Wu and Modarres (2017, 2018). For convenience of notation, $\xi_n(t)$ will be denoted by $\xi(t)$ for the rest of the paper.

We first apply these two-step techniques to estimate extreme quantiles with an application to US weather data for several cities. These three procedures are then compared to the parametric model of quantile regression. The two-step estimation procedures consists of empirically obtaining raw estimates of unknown quantiles from the original data in first step, and applying the nonparametric smoothing in the second step. The three two-step procedures which will be used are detailed in the following sections.

2.2.1 Kernel Smoothing

The mean of the unknown regression function $m(t)$ can be written as the conditional expectation of the response variable Y relative to the predictor T as $m(t) = E(Y|T = t)$. At some specific point, say t_0, the best estimate of $m(t_0)$ would be the value of the dependent variable at that point. In actuality, what we may do to create a smooth fit for the data is that at the point t_0, we take some small neighborhood around t_0 and find the mean of the response data in that local neighborhood, where the data closer to t_0 receives larger weight. This will be the idea behind the first method we investigate: kernel smoothing.

Suppose $f(t, \xi(t))$ is the joint pdf of the random bivariate data $(t_1, \xi(t_1)) \dots (t_J, \xi(t_J))$. Let $m(t)$ be an unknown regression function. Then the nonparametric regression model is

$$\xi(t_j) = m(t_j) + \epsilon_j; j = 1, \dots, J \tag{2}$$

The errors (ϵ_j) satisfy $E(\epsilon_j) = 0, Var(\epsilon_j) = \sigma^2_\epsilon$ and $Cov(\epsilon_j, \epsilon_k) = 0$ for j $\neq$ k. The unknown regression function $m(t)$ will be derived as follows:

$$m(t) = E[\xi(t)|T = t] = \int \xi(t) f\big[\xi(t)|t\big] d\xi(t) = \frac{\int \xi(t) f\big[t, \xi(t)\big] d\xi(t)}{\int f\big[t, \xi(t)\big] d\xi(t)}$$

$m(t)$ is a ratio of two correlated random quantities. The product kernel density estimator technique can be used to estimate the numerator and denominator separately. i.e;

$$\begin{aligned}
\hat{f}\big[t, \xi(t)\big] &= \frac{1}{J h_t h_\xi} \sum_{j=1}^{J} K\big(\frac{t_j - t}{h_t}\big) K\big(\frac{\xi(t_j) - \xi(t)}{h_\xi}\big) \\
&= \frac{1}{J} \sum_{j=1}^{J} K_{h_t}\big(t_j - t\big) K_{h_\xi}\big(\xi(t_j) - \xi(t)\big),
\end{aligned}$$

where $K_h(t_j - t) = K[(t_j - t)/h]/h$, $K(\cdot)$ is a non-negative kernel function, and $h > 0$ is a bandwidth. By using the symmetry of the kernel and transformation of variables, we have

$$\begin{aligned}
\int \xi \hat{f}\big[t, \xi(t)\big] d\xi &= \frac{1}{J} \int \xi \sum_{j=1}^{J} K_{h_t}\big(t_j - t\big) K_{h_\xi}\big(\xi(t_j) - \xi(t)\big) \\
&= \frac{1}{J} \sum_{j=1}^{J} K_{h_t}\big(t_j - t\big) \xi(t_j)
\end{aligned}$$

For the denominator, we have

$$\begin{aligned}
\int \hat{f}\big[t, \xi(t)\big] d\xi &= \frac{1}{J} \sum_{j=1}^{J} K_{h_t}\big(t_j - t\big) \int K_{h_\xi}\big(\xi(t_j) - \xi(t)\big) d\xi \\
&= \frac{1}{J} \sum_{j=1}^{J} K_{h_t}\big(t_j - t\big) \\
&= \hat{f}(t)
\end{aligned}$$

Therefore,

$$\hat{m}(t) = \sum_{j=1}^{J} W_{h_t}\big(t_j - t\big) \xi(t_j) \tag{3}$$

Where $W_{h_t}\big(t_j - t\big) = \frac{K_{h_t}\big(t_j - t\big)}{\sum_{j=1}^{J} K_{h_t}\big(t_j - t\big)}$ and $\sum_{j=1}^{J} W_{h_t}\big(t_j - t\big) = 1$. Estimator (3) is widely known as the Nadaraya-Watson type kernel estimator. In kernel smoothing, the estimate of $m(t)$ is a weighted local average of the response variable $\xi(t)$. Weights are assigned to neighboring points, provided by the kernel function K_h, with data closer to the point of interest receiving greater weight in the calculation.

2.2.2 Local Polynomial Smoothing

Extending the local average which is the result of kernel smoothing above, a higher order function can be used for estimation. Improving from the local constant estimate of kernel smoothing, a local linear (or higher order polynomial) can be used instead. In the case of a local linear estimator, this is the result from fitting a degree one polynomial in the local neighborhoods, rather than performing a simple local average. Suppose that $\hat{\xi}(t)$ is $p+1$ times continuously differentiable with respect to t. Let $\xi^{(q)}(t)$ be the qth derivative of $\xi(t), 1 \leq q \leq p$ and $\beta_q(t) = \xi^{(q)}(t)/q!$. The Taylor series approximation of $\xi(t)$ up to order p can be written as

$$\xi(t) \approx \sum_{q=0}^{p} \beta_q(a_0)(t-a_0)^q$$

for t in the neighborhood of a_0. The raw estimates $\widetilde{\xi}(t_j)$ can be treated as the observations of $\xi(t_j)$ at t_j, $j = 1, \ldots, J$, and obtain the pth local polynomial estimators by minimizing

$$\sum_{j=1}^{J} \left\{ \widetilde{\xi}(t_j) - \sum_{q=0}^{p} \beta_q(t_0)(t_j - t_0)^q \right\}^2 K_h(t_j - t_0),$$

where again $K_h(t_j - t)$ is a non-negative kernel function, and $h > 0$ is the bandwidth. A smooth curve can be created by choosing a grid of t_0's and minimizing this expression in each local neighborhood of t_0. Using the matrix formulation, we define $\widetilde{\xi}(\mathbf{t}) = (\widetilde{\xi}(t_1), \ldots, \widetilde{\xi}(t_J))^T$, $\beta(t) = (\beta_0(t), \ldots, \beta_p(t))^T$, weighting matrix $W(t;h) = \text{diag}\{K_h(t_j - t)\}$ where its jth column $W_j(t;h) = (0, \ldots, K_h(t_j - t), \ldots, 0)^T$, and $T_p(t)$ the $J \times (p+1)$ matrix with its jth row given by $T_{j,p}(t) = (1, t_j - t, \ldots, (t_j - t)^p)$. The local polynomial estimators $\widehat{\beta}_q(t)$ minimize

$$Q_G\big[\beta(t)\big] = \big[\widetilde{\xi}(\mathbf{t}) - T_p(t)\beta(t)\big]^T W(t;h)\big[\widetilde{\xi}(\mathbf{t}) - T_p(t)\beta(t)\big].$$

The pth order local polynomial estimator of $\xi^{(q)}(t)$ based on $\widetilde{\xi}(t_j)$, which minimizes $Q_G\big[\beta(t)\big]$, is

$$\widehat{\xi}^{(q)}(t) = \sum_{j=1}^{J} \Big\{ W_{q,p+1}(t_j, t; h)\, \widetilde{\xi}(t_j) \Big\} \tag{4}$$

where $W_{q,p+1}(t_j, t; h) = q! e_{q+1,p+1} \big[T_p^T(t) W(t; h) T_p(t)\big]^{-1} \big[T_{j,p}^T(t) W_j(t; h)\big]$ is the "equivalent kernel function" (Fan and Zhang, 2000) and $e_{q+1,p+1}$ is the row vector of length $p+1$ with 1 at its $(q+1)$th place and 0 elsewhere. By the definition of $\beta(t)$, we have $\widehat{\beta}(t) = \big(\widehat{\beta}_0(t), \ldots, \widehat{\beta}_p(t)\big)^T$ and $\widehat{\xi}^{(q)}(t) = \widehat{\beta}_q(t)\, q!$ is an estimator for $\xi^{(q)}(t)$, $q = 0, 1, \ldots, p$. For local polynomial fitting $p - q$ should be taken to be odd as shown in Ruppert and Wand (1994) and Fan and Gijbels (1996). When $p = 1$, we get the local linear estimator $\widehat{\xi}_L(t_j) = \widehat{\beta}_0(t_j)$ of $\xi(t)$ based on (4) and the equivalent kernel function $W_{0,2}(t_j, t; h)$. So, the local linear estimator is

$$\widehat{\xi}_L(t) = \widehat{\xi^{(0)}}(t|x) \tag{5}$$

2.2.3 Spline Smoothing

Let us consider the data points $(t_1, \xi(t_1)), (t_2, \xi(t_2)), \ldots, (t_J, \xi(t_J))$. We want to find a function $\hat{m}(t)$, which is a good approximation to the true conditional expectation or regression function $m(t)$ i.e; $m(t) = E(\xi(t)|T = t)$. A natural way to do this in one dimension is to minimize the spline objective function

$$O(m, \lambda) = \sum_{j=1}^{J} \big(\xi(t_j) - m(t_j)\big)^2 + \lambda \int \big(m''(t)\big)^2 dt \tag{6}$$

where λ is a smoothing parameter, chosen by cross-validation approach. The first term is just the mean squared error (MSE) using the curve $m(t)$ to predict $\xi(t)$. The second term penalizes curvature in the function. m'' is the second derivative of m with respect to t. This would be zero for linear m, so it measures the curvature of m at t. The sign of m'' determines whether m is concave or convex but squaring it makes it immaterial. We then integrate this over all t to say how curved m is, on average. Finally, we multiply by λ and add that to the MSE. This is known adding a penalty to the MSE. Given two functions with the same MSE, we choose the one with less average curvature. It can be shown (Green and Silverman, 1994; V. Solo, 2000) that (6) has an explicit, finite-dimensional, unique minimizer which is a natural cubic spline with knots at the unique values of the t_j, j=1,2,. . . , J. It seems that the family is still over-parametrized, since there are as many as J knots, which implies J degrees freedom. However, the penalty term translates to a penalty on the spline coefficients, which are shrunk some of the way toward the linear fit (Hastie, Tibshirani and Friedman, 2009). Since the solution is a natural spline, we can write it as $m(t) = \sum_{j=1}^{J} N_j(t)\theta_j$ where the N_j are an J-dimensional set of basis functions for representing this family of natural splines. After above reparametrization, the optimization problem (6) turns out as

$$O(\theta, h) = \sum_{j=1}^{J} \left(\xi(t_j) - \sum_{j=1}^{J} N_j(t)\theta_j\right)^2 + \lambda \int \left(\sum_{j=1}^{J} N_j^{''}(t)\theta_j\right)^2 dt \tag{7}$$

By defining the basis matrix and penalty matrices $N, \Omega \in \Re$ by
$N_{ij} = N_j(t_j)$ and $\Omega_{ij} = \int N_i^{''}(t) N_j^{''}(t)dt$ for $i, j = 1, 2, \ldots, J$, the problem (7) becomes

$$O(\theta, \lambda) = (\xi - N\theta)^T(\xi - N\theta) + \lambda\theta^T\Omega\theta \tag{8}$$

The solution is easily seen to be $\tilde{\theta} = (N^TN + h\Omega)^{-1}N^T\xi$. The fitted smoothing spline is given by

$$\hat{m}(t) = \sum_{j=1}^{J} N_j(t)\tilde{\theta}_j \tag{9}$$

2.3 Simulation of Nonparametric Quantile Estimation

To assess the performances of the smoothing curves obtained from the three two-step smoothing estimators against quantile regression line, we first conduct a simulation study.

We also compare the relative performance of these three two-step smoothing estimators among themselves. We compare their performances by computing bias, MSE, and coverage. Data is simulated with increasing variance over 50 time points (TP) with the standard deviation s_t being $s_t = 0.1 + 0.05 * t$, $t \in \{1, 2, ..., 50\}$. The heterocedastic model for data simulation is $y = b_0 + b_1 * t + e$, where $b_0 = 3$, $b_1 = 0.1$ and $e \sim N(0, s_t)$. At each time point, 365 data points are generated from this function. Then 500 simulations are created to evaluate these four methods on the 5^{th} and 95^{th} percentile values. For quantile regression, the model is fit on the entire data and to compare against the two-step methods, raw estimates are again computed for each TP. The coverage probability can be computed by creating confidence intervals for each simulation using the variance at each time point. The proportion of simulations which then contain the true value of y is the coverage probability. The Epanechnikov kernel and the optimal bandwidth from cross-validation are used for the smoothing estimators. Leave one out cross-validation (LOOCV) is used to select the optimal bandwidth in local polynomial and kernel smoothing and, in the case of spline smoothing, the smoothing parameter. The parameter which minimizes error in LOOCV is chosen. Local polynomial, kernel and spline smoothing are implemented in R with the *locpoly*, *ksmooth*, and *smooth.spline* functions, respectively, using the KernSmooth and ibr packages.

Table 2.1 contains the simulation results for 95^{th} percentile values. From Table 2.1, we see that two-step local polynomial smoothing (LP) and two-step spline smoothing (SS) have less bias than the quantile regression line (QR) for all 50 time points. For two-step kernel smoothing (KS), QR has less bias only at the first four time points. From comparison of MSE for these four methods, we conclude that SS estimator has less MSE than QR in all 50 time points. We also see that at the first three time points, KS estimator has higher MSE than QR and only in the last time point, LP smoothing estimator has higher MSE than the QR, and in all other time points, QR has higher MSE than the LP estimator and KS estimator. For coverage probability, we see that only at time point 49, QR has higher coverage than the LP estimator and in all other time points, QR has lower coverage than all three two-step smoothing estimators. The striking result is that the three two-step methods outperform QR. In comparing the two-step methods to one another, Table 2.1 also shows that out of 50 time points, LP has less biases in 17 time points whereas KS and SS has less biases 18 and 15 time points. At the boundary points (i.e. TP 1-4 and TP 47-50), LP and SS appear superior to KS, while KS has a slight advantage in interior points. In terms of MSE, KS has less MSE in 41 time points than SS and LP. Again though, LP performs better at the first 5 boundary points when looking at the coverage probability. In interior points, all these three smoothing estimators have consistent results. Similar observations are made for the 5^{th} percentiles, which can be found in Table 2.2. SS is superior in boundary points, and a mixed combination of the three perform best at interior point. Over all of the 50 time points, SS is best at 27, LP best at 14, and KS best at 9 for bias. When comparing MSE, LP is again superior at the first 8 boundary points, and 3 of the last 4 boundary points. KS is superior at almost every interior point, with 36 of the 50 points better estimated by KS in simulations.

Table 2.1: Bias, MSE, Coverage and Best Estimator (BE) corresponding to 50 Time Points (TP) for the Quantile Regression (QR), Local Polynomial Smoothing (LP), Kernel Smoothing (KS) and Spline Smoothing (SS) estimators for the 95th percentile values from the Simulation Design.

	Bias					MSE					Coverage				
TP	QR	LP	KS	SS	BE	QR	LP	KS	SS	BE	QR	LP	KS	SS	BE
1	0.06425	0.00463	0.39915	0.00669	lp	0.01423	0.00166	0.18366	0.00941	lp	0.92000	0.94500	0.24500	0.96000	ss
2	0.06118	-0.00538	0.26246	-0.00319	ss	0.01654	0.00371	0.08819	0.01092	lp	0.93000	0.99500	0.50000	0.99000	lp
3	0.07129	-0.00791	0.16525	-0.00543	ss	0.02093	0.00752	0.04464	0.01355	lp	0.93500	0.99500	0.73000	0.98000	lp
4	0.08926	0.00739	0.11077	0.01033	lp	0.02857	0.01001	0.02627	0.01272	lp	0.89500	0.98500	0.84500	0.97500	lp
5	0.10044	0.01234	0.07110	0.01493	lp	0.03554	0.01530	0.02187	0.01981	lp	0.93000	0.98500	0.91000	0.97500	lp
6	0.08192	-0.00008	0.03112	0.00078	lp	0.03798	0.02053	0.02149	0.02352	lp	0.96000	0.99500	0.99500	0.97500	lp
7	0.07986	-0.00725	0.00992	-0.00823	lp	0.04177	0.02215	0.02171	0.03407	ks	0.95000	0.99500	0.98000	0.99500	lp, ss
8	0.09606	0.00426	0.01423	0.00154	ss	0.05611	0.02946	0.02856	0.03657	ks	0.96000	0.99500	0.99000	0.99000	lp
9	0.10221	-0.01241	-0.00884	-0.01601	ks	0.05471	0.03035	0.02898	0.05234	ks	0.95000	0.98000	0.97500	0.98500	ss
10	0.11524	-0.00287	-0.00389	-0.00682	lp	0.06538	0.03327	0.03173	0.05633	ks	0.92000	0.99000	0.99000	0.99000	lp, ks, ss
11	0.13342	0.01399	0.01329	0.00981	ss	0.08780	0.04618	0.04445	0.05324	ks	0.93000	0.98500	0.98000	0.99000	ss
12	0.11755	-0.00624	-0.00498	-0.01068	ks	0.09408	0.05336	0.05144	0.06182	ks	0.97000	0.98000	0.98000	0.99000	ss
13	0.12793	-0.01482	-0.01708	-0.01874	lp	0.10173	0.05883	0.05739	0.08269	ks	0.94000	0.99000	0.98500	0.99500	ss
14	0.15221	0.00383	0.00703	0.00122	ss	0.11369	0.06822	0.06613	0.09600	ks	0.94000	1.00000	0.99500	1.00000	lp, ss
15	0.17567	0.02365	0.01960	0.02377	ks	0.14488	0.08265	0.07963	0.12156	ks	0.95000	0.99000	0.99000	0.98500	lp, ks
16	0.16839	0.00292	0.00274	0.00584	ks	0.14788	0.09233	0.08765	0.11506	ks	0.96000	0.99000	0.99000	0.99500	ss
17	0.12675	-0.03880	-0.03980	-0.03525	ss	0.12554	0.08892	0.08653	0.13331	ks	0.96000	0.99500	0.99500	0.99500	lp, ks, ss
18	0.19263	0.02017	0.02042	0.02512	lp	0.17832	0.10130	0.09518	0.13619	ks	0.93000	0.98500	0.98000	0.98500	lp, ss
19	0.18431	-0.00435	-0.00379	0.00274	ss	0.20863	0.12756	0.12267	0.14266	ks	0.96500	0.98500	0.98500	0.99500	ss
20	0.19142	0.01231	0.01166	0.01788	ks	0.21168	0.13018	0.12431	0.14824	ks	0.95500	0.99000	0.98500	0.99500	ss
21	0.18547	-0.00595	-0.00489	-0.00495	ks	0.30145	0.18934	0.18163	0.17652	ss	0.97500	0.99500	0.99500	0.99500	lp, ks, ss
22	0.19690	0.00105	0.00362	-0.00004	ss	0.23681	0.16001	0.15440	0.22822	ks	0.96500	0.99500	0.99500	0.99500	lp, ks, ss
23	0.19782	0.00085	-0.00342	0.00094	lp	0.27254	0.17030	0.16408	0.19883	ks	0.96000	0.99000	0.98500	0.99500	ss
24	0.22626	0.00278	0.00769	0.00435	lp	0.28884	0.17887	0.16910	0.27431	ks	0.96500	0.98500	0.98500	1.00000	ss
25	0.22251	0.00181	0.00049	-0.00034	ss	0.27959	0.16815	0.16304	0.23127	ks	0.96000	0.99500	0.99500	0.99500	lp, ks, ss
26	0.23225	-0.00249	-0.00077	-0.01138	ks	0.31266	0.20060	0.18993	0.27053	ks	0.94000	0.99000	0.99000	0.99500	ss
27	0.22757	-0.01118	-0.01484	-0.02007	lp	0.38842	0.24513	0.23673	0.30000	ks	0.97000	0.99500	0.99500	1.00000	ss
28	0.27096	0.02097	0.01547	0.01948	ks	0.40951	0.23771	0.23256	0.35996	ks	0.93000	0.99000	0.99000	0.99500	ss
29	0.25905	0.00601	0.00972	0.01003	lp	0.41258	0.26056	0.25214	0.35465	ks	0.94000	0.99500	0.99500	0.98500	lp, ks
30	0.24640	-0.01457	-0.00428	-0.01101	ks	0.45918	0.29859	0.27881	0.43137	ks	0.96000	0.98500	0.98500	0.98500	lp, ks, ss
31	0.25585	-0.00439	-0.00322	-0.00416	ks	0.53705	0.34162	0.32336	0.39522	ks	0.97500	0.99500	0.99500	1.00000	ss
32	0.31117	0.04842	0.04269	0.04396	ks	0.45838	0.27180	0.26432	0.37854	ks	0.94000	0.97500	0.97500	0.98500	ss
33	0.17132	-0.08865	-0.09348	-0.09957	lp	0.55813	0.37777	0.36595	0.42712	ks	0.98500	0.99500	0.99500	1.00000	ss
34	0.32832	0.07617	0.07505	0.06025	ss	0.54438	0.32752	0.31562	0.44487	ks	0.94500	0.99000	0.98500	0.99000	lp, ss
35	0.21172	-0.04810	-0.04456	-0.06164	ks	0.56733	0.37944	0.36900	0.52047	ks	0.97000	0.98500	0.98500	0.99000	ss
36	0.26538	-0.01243	-0.00836	-0.01607	ks	0.52441	0.35580	0.34631	0.45928	ks	0.96500	0.98500	0.98500	0.98500	lp, ks, ss
37	0.33627	0.03886	0.03886	0.04457	ks	0.66087	0.37524	0.35800	0.49119	ks	0.98000	0.99000	0.99000	0.98500	lp, ks
38	0.32462	0.00868	0.00811	0.01696	ks	0.72391	0.47046	0.45372	0.62837	ks	0.98500	0.98500	0.98500	0.99500	ss
39	0.28420	-0.03108	-0.03720	-0.02279	ss	0.74661	0.47226	0.44182	0.64682	ks	0.97500	0.99000	1.00000	0.99000	ks
40	0.37469	0.04680	0.04211	0.05966	ks	0.78038	0.48931	0.46829	0.66490	ks	0.93500	0.99000	0.99000	0.99500	ss
41	0.33426	0.00881	0.00429	0.02640	ks	0.76065	0.47213	0.45459	0.63980	ks	0.95500	0.98500	0.98500	0.98000	lp, ks
42	0.30612	-0.02824	-0.02377	-0.01419	ss	0.76704	0.54803	0.51463	0.65794	ks	0.99000	0.99500	0.99500	0.99500	lp, ks, ss
43	0.33058	0.00418	-0.00881	0.01246	lp	0.79922	0.54640	0.53184	0.78488	ks	0.94500	0.98500	0.98500	0.98500	lp, ks, ss
44	0.23798	-0.08651	-0.09265	-0.07747	ss	0.87177	0.62392	0.59179	0.70685	ks	0.98500	0.99500	0.99500	0.99500	lp, ks, ss
45	0.38479	0.05659	0.02710	0.06449	ks	0.92005	0.58782	0.55179	0.68489	ks	0.94000	0.98500	0.99000	0.98500	ks
46	0.32514	-0.01238	-0.07962	-0.01648	lp	0.88125	0.59229	0.57485	0.72694	ks	0.96500	0.97500	0.98500	0.98500	ks, ss
47	0.31918	-0.02565	-0.12681	-0.04449	lp	0.81539	0.57140	0.55109	0.65286	ks	0.95000	0.97500	0.98000	0.98000	ks, ss
48	0.39923	0.04661	-0.13383	0.02065	ss	1.29477	0.78562	0.87090	0.71095	ss	0.98000	0.98000	0.99000	0.99500	ss
49	0.34117	-0.01670	-0.29960	-0.04420	lp	1.30480	0.34284	0.86344	0.75018	lp	0.98000	0.97500	1.00000	1.00000	ks, ss
50	0.43877	0.08283	-0.32700	0.04840	ss	0.98880	2.06927	0.62301	0.73852	ks	0.93500	0.97500	1.00000	0.98000	ks

Table 2.2: Bias, MSE, Coverage and Best Estimator (BE) corresponding to 50 Time Points (TP) for the Quantile Regression (QR), Local Polynomial Smoothing (LP), Kernel Smoothing (KS) and Spline Smoothing (SS) estimators for the 5th percentile values of the Simulation Design).

	Bias					MSE					Coverage				
TP	QR	LP	KS	SS	BE	QR	LP	KS	SS	BE	QR	LP	KS	SS	BE
1	-0.03367	-0.00154	-0.18687	-0.00002	ss	0.00711	0.00340	0.04444	0.00941	lp	0.93500	0.93500	0.48000	0.94500	ss
2	-0.03156	0.00754	-0.15130	0.00851	lp	0.00986	0.00503	0.03392	0.01092	lp	0.94000	0.96500	0.66000	0.94000	lp
3	-0.04970	-0.00311	-0.13766	-0.00214	ss	0.01463	0.00677	0.03079	0.01355	lp	0.91000	0.94500	0.71500	0.94500	lp, ss
4	-0.06766	-0.01007	-0.12292	-0.00958	ss	0.01754	0.01043	0.03044	0.01272	lp	0.90500	0.95000	0.81000	0.95500	ss
5	-0.07163	-0.00774	-0.10091	-0.00686	ss	0.02347	0.01441	0.02819	0.01981	lp	0.92000	0.95500	0.88000	0.95500	lp, ss
6	-0.07629	-0.00735	-0.08531	-0.00610	ss	0.02927	0.01888	0.02851	0.02352	lp	0.93000	0.96000	0.91500	0.95500	lp
7	-0.05337	0.01958	-0.04556	0.02241	lp	0.03741	0.02948	0.03163	0.03407	lp	0.95500	0.94000	0.96000	0.95000	ks
8	-0.08811	-0.00604	-0.05839	-0.00221	ss	0.04419	0.03279	0.03662	0.03657	lp	0.92500	0.96000	0.94000	0.96000	lp, ss
9	-0.06736	0.02024	-0.02617	0.02367	lp	0.05582	0.04895	0.04794	0.05234	ks	0.95000	0.93500	0.94000	0.94000	qr
10	-0.08518	0.00886	-0.02479	0.01244	lp	0.06534	0.05437	0.05374	0.05633	ks	0.94500	0.95000	0.96000	0.96000	ks, ss
11	-0.10988	-0.01322	-0.03843	-0.00612	ss	0.06590	0.05051	0.05208	0.05324	lp	0.93000	0.95500	0.96000	0.96000	ks, ss
12	-0.12129	-0.01606	-0.03510	-0.00874	ss	0.07907	0.05659	0.05872	0.06182	lp	0.93000	0.96000	0.96500	0.95500	ks
13	-0.10267	0.01114	0.00041	0.01893	ks	0.09701	0.07597	0.07286	0.08269	ks	0.94000	0.94000	0.94500	0.95000	ss
14	-0.13173	-0.00915	-0.01940	-0.00273	ss	0.11768	0.09062	0.08752	0.09600	ks	0.94000	0.94500	0.95000	0.95000	ks, ss
15	-0.07883	0.05330	0.03909	0.05814	ks	0.12869	0.11445	0.10820	0.12156	ks	0.95000	0.93000	0.93500	0.94000	qr
16	-0.16068	-0.01883	-0.02257	-0.01486	ss	0.14551	0.10746	0.10606	0.11506	ks	0.93000	0.96000	0.96000	0.95500	lp, ks
17	-0.18585	-0.03310	-0.03528	-0.03228	ss	0.17210	0.12772	0.12675	0.13331	ks	0.94500	0.95500	0.95500	0.94500	lp, ks
18	-0.12649	0.03492	0.03625	0.03453	ss	0.15919	0.12695	0.12489	0.13619	ks	0.95000	0.95500	0.95500	0.95000	lp, ks
19	-0.16738	-0.00104	-0.00028	-0.00361	ks	0.17550	0.13461	0.13279	0.14266	ks	0.93500	0.96500	0.96500	0.96500	lp, ks, ss
20	-0.21336	-0.03918	-0.04067	-0.04132	lp	0.19460	0.14296	0.14167	0.14824	ks	0.92000	0.96000	0.96000	0.95000	lp, ks
21	-0.20826	-0.02572	-0.02558	-0.03111	ks	0.22748	0.16617	0.16042	0.17652	ks	0.92500	0.95500	0.95000	0.95500	lp, ss
22	-0.19034	0.00009	-0.00879	-0.00655	lp	0.28100	0.20655	0.19972	0.22822	ks	0.93000	0.96000	0.96000	0.96000	lp, ks, ss
23	-0.17207	0.02716	0.02503	0.01944	ss	0.24029	0.17815	0.17281	0.19883	ks	0.94000	0.95500	0.95500	0.95000	lp, ks
24	-0.22689	-0.01876	-0.01560	-0.02887	ks	0.33813	0.24754	0.24441	0.27431	ks	0.94000	0.96000	0.96000	0.96000	lp, ks, ss
25	-0.21840	-0.00286	-0.00450	-0.01533	lp	0.29043	0.20856	0.20404	0.23127	ks	0.95500	0.97000	0.97000	0.96000	lp, ks
26	-0.18801	0.03584	0.04005	0.02543	ss	0.32143	0.24374	0.24132	0.27053	ks	0.95500	0.96500	0.96500	0.96000	lp, ks
27	-0.20926	0.02129	0.01828	0.01249	ss	0.37558	0.28466	0.27794	0.30000	ks	0.93500	0.95000	0.94500	0.95000	lp, ss
28	-0.28659	-0.05918	-0.06019	-0.06687	lp	0.46518	0.32570	0.31887	0.35996	ks	0.93000	0.95000	0.95000	0.94500	lp, ks
29	-0.23812	-0.01065	-0.00510	-0.01503	ks	0.43735	0.33931	0.33347	0.35465	ks	0.97000	0.97500	0.97500	0.98000	ss
30	-0.21945	0.00901	0.01132	0.00943	lp	0.52576	0.38830	0.38133	0.43137	ks	0.94000	0.95500	0.95500	0.96000	ss
31	-0.27046	-0.03559	-0.03387	-0.03670	ks	0.49602	0.35533	0.34617	0.39522	ks	0.95000	0.94500	0.94000	0.95500	ss
32	-0.27583	-0.03276	-0.03847	-0.03809	lp	0.50473	0.35621	0.34076	0.37854	ks	0.94500	0.95500	0.95500	0.96500	ss
33	-0.21467	0.03659	0.03576	0.03213	ss	0.48765	0.40485	0.40175	0.42712	ks	0.93500	0.95500	0.95500	0.96500	ss
34	-0.25568	0.00547	0.01376	0.00490	ss	0.52470	0.41291	0.40770	0.44487	ks	0.96000	0.95000	0.95000	0.95000	qr
35	-0.22957	0.04015	0.05124	0.03987	ss	0.61595	0.48754	0.47833	0.52047	ks	0.94500	0.95000	0.94500	0.95000	lp, ss
36	-0.29829	-0.02284	-0.02647	-0.02687	lp	0.56104	0.42090	0.40715	0.45928	ks	0.93500	0.95000	0.95500	0.95500	ks, ss
37	-0.29726	-0.01795	-0.00889	-0.02040	ks	0.59651	0.46615	0.46167	0.49119	ks	0.93500	0.95500	0.95500	0.95000	lp, ks
38	-0.20112	0.08033	0.09916	0.07913	ss	0.72335	0.58460	0.57747	0.62837	ks	0.95500	0.94500	0.94000	0.96000	ss
39	-0.28814	-0.00302	0.02325	-0.00839	lp	0.79041	0.60280	0.58407	0.64682	ks	0.94000	0.94500	0.94000	0.95000	ss
40	-0.21661	0.07773	0.08803	0.06660	ss	0.76766	0.62887	0.61174	0.66490	ks	0.95500	0.94000	0.92000	0.93000	qr
41	-0.24084	0.07248	0.09705	0.06346	ss	0.75335	0.60492	0.59711	0.63980	ks	0.95000	0.95500	0.95500	0.95000	lp, ks
42	-0.32561	0.00985	0.03726	-0.00312	ss	0.80273	0.60952	0.56609	0.65794	ks	0.95000	0.96000	0.96000	0.96500	ss
43	-0.38827	-0.03510	0.00627	-0.05414	ks	0.97316	0.73678	0.72060	0.78488	ks	0.95500	0.95000	0.94000	0.96500	ss
44	-0.39009	-0.02991	0.03540	-0.04309	lp	0.90156	0.67517	0.65564	0.70685	ks	0.95000	0.96000	0.96500	0.96000	ks
45	-0.34202	0.01589	0.08624	0.01297	ss	0.89833	0.62719	0.62699	0.68489	ks	0.95000	0.94500	0.94500	0.95000	qr, ss
46	-0.36146	-0.00944	0.06440	-0.01074	lp	0.96700	0.69363	0.67328	0.72694	ks	0.95500	0.94500	0.93500	0.95000	qr
47	-0.38094	-0.02725	0.06702	-0.02164	ss	0.91349	0.54209	0.58494	0.65286	lp	0.94500	0.95000	0.94000	0.93500	lp
48	-0.41814	-0.05362	0.06109	-0.05010	ss	0.97118	0.58349	0.59732	0.71095	lp	0.93500	0.93500	0.94000	0.95000	ss
49	-0.34433	0.04502	0.18337	0.03188	ss	0.96101	0.51763	0.76124	0.75018	lp	0.95500	0.95000	0.94000	0.95000	qr
50	-0.34673	0.07728	0.20169	0.03728	ss	0.99687	1.35717	0.76560	0.73852	ss	0.94000	0.95000	0.94000	0.94500	lp

2.4 Application to Temperature Data in Seven U.S. Cities

In addition to the simulations, the two-step methods are applied to temperature data, measured in degree Celsius, from 7 US cities (Dallas, Kansas City, Miami, Minneapolis, Portland, San Diego, and Seattle). These data were recorded as the minimum and maximum daily temperatures by the US Meteorological Department from 1990 to 2016. From this data, we computed the 5% (on daily minimum temperatures) and 95% (on daily maximum temperatures) quantile of temperature for each of the 27 years for these 7 cities. Throughout this section, 95% quantile smoothings will refer to the daily maximum temperatures, and 5% quantile smoothings will refer to the daily minimum temperatures of the cities. There are $J = 27$ distinct time design points $\{t_1, t_2, \ldots, t_{27}\} = \{1990, 1991, \ldots, 2016\}$. Thus, for a given $1 \leq j \leq J = 27$, we denote $T_{0.05}(t_j)$ and $T_{0.95}(t_j)$ as the 5% and 95% quantile of temperatures at year t_j. Applying the two-step kernel smoothing (KS) estimator, local polynomial smoothing (LPS) estimator, spline smoothing (SS) estimator and the fitted quantile regression (QR) line to the observed data $\{T_{0.95}(t_j), t_j; 1 \leq j \leq J, 1 \leq i \leq n\}$ and $\{T_{0.05}(t_j), t_j; 1 \leq j \leq J, 1 \leq i \leq n\}$, we obtain the 5% and 95% smoothing quantile curves/lines on temperature data for the entire time design points $\{t_1, t_2, \ldots, t_{27}\} = \{1990, 1991, \ldots, 2016\}$.

Table 2.3 shows the raw estimates and resulting smoothing values from the four different techniques at both the 5^{th} and 95^{th} percentiles for the city of Dallas. For the 95^{th} percentile, a combination of KS and LP perform best at the boundary points, while for the 5^{th} percentile KS is best. Figures 2.1 and 2.2 depict the smoothing results shown in Table 2.3. In Figure 2.1, the 95^{th} percentile smoothing results are shown for all 7 cities in the application of the methods. From left to right for each city, KS, LP, SS and QR are displayed. Similarly, Figure 2.2 shows the same for the 5^{th} percentile. The first row of each of these figures has the results for Dallas. It can be seen that LP and SS give a smoother estimate for the 95^{th} percentile of temperatures than KS, while for the 5^{th} percentile LP

Table 2.3: Raw estimates($T_{.95}(t_j)$, $T_{.05}(t_j)$), Kernel smoothing estimates ($KS_{.95}(t_j)$, $KS_{.05}(t_j)$), local polynomial smoothing estimates ($LP_{.95}(t_j)$, $LP_{.05}(t_j)$), spline smoothing estimate ($SS_{.95}(t_j)$, $SS_{.05}(t_j)$), and quantile regression estimate ($QR_{.95}(t_j)$, $QR_{.05}(t_j)$) of 95th and 5th percentile of temperatures from Dallas between 1990 and 2016.

t_j	$T_{.95}(t_j)$	$KS_{.95}(t_j)$	$LP_{.95}(t_j)$	$SS_{.95}(t_j)$	$QR_{.95}(t_j)$	$T_{.05}(t_j)$	$KS_{.05}(t_j)$	$LP_{.05}(t_j)$	$SS_{.05}(t_j)$	$QR_{.05}(t_j)$
1990	37.20	37.17	37.21	37.22	36.93	-0.48	-0.52	-0.77	-0.63	-1.10
1991	36.70	36.67	36.66	36.51	36.99	-1.10	-0.95	-0.81	-0.47	-1.10
1992	35.60	35.79	36.56	36.08	37.06	0.73	0.51	-0.85	-0.11	-1.10
1993	37.80	37.61	36.89	37.28	37.12	-1.00	-0.90	-0.88	-0.54	-1.10
1994	36.70	36.78	37.00	37.02	37.18	-1.10	-1.09	-0.91	-1.14	-1.10
1995	37.10	37.05	36.89	36.96	37.25	-1.10	-1.30	-0.94	-1.99	-1.10
1996	36.70	36.69	36.81	36.55	37.31	-4.28	-3.88	-0.97	-2.71	-1.10
1997	36.10	36.32	37.24	36.67	37.37	-1.10	-1.22	-0.99	-1.54	-1.10
1998	39.40	39.15	38.13	38.75	37.44	0.12	0.08	-1.02	-0.11	-1.10
1999	38.30	38.40	38.48	38.76	37.50	0.70	0.55	-1.05	0.19	-1.10
2000	38.90	38.74	38.07	38.50	37.56	-1.10	-1.02	-1.07	-0.69	-1.10
2001	36.70	36.79	37.24	36.90	37.62	-1.58	-1.56	-1.10	-1.47	-1.10
2002	36.10	36.22	36.74	36.42	37.69	-1.70	-1.69	-1.13	-1.78	-1.10
2003	37.68	37.44	36.58	36.93	37.75	-1.70	-1.69	-1.16	-1.73	-1.10
2004	35.00	35.28	36.50	35.69	37.81	-1.55	-1.50	-1.20	-1.31	-1.10
2005	37.20	37.20	37.16	37.22	37.88	-0.60	-0.58	-1.23	-0.53	-1.10
2006	39.40	39.09	37.74	38.62	37.94	0.60	0.42	-1.27	-0.07	-1.10
2007	36.10	36.40	37.56	36.98	38.00	-1.10	-0.96	-1.31	-0.50	-1.10
2008	38.18	38.04	37.66	37.68	38.07	-0.60	-0.70	-1.35	-0.95	-1.10
2009	37.80	37.85	38.08	37.89	38.13	-1.70	-1.63	-1.40	-1.53	-1.10
2010	38.30	38.40	38.75	38.59	38.19	-1.70	-1.76	-1.44	-1.88	-1.10
2011	40.48	40.30	39.39	40.09	38.25	-2.68	-2.46	-1.50	-1.75	-1.10
2012	39.40	39.40	39.19	39.53	38.32	0.00	-0.26	-1.55	-1.18	-1.10
2013	38.30	38.30	38.35	38.27	38.38	-1.60	-1.64	-1.61	-1.83	-1.10
2014	37.20	37.26	37.54	37.30	38.44	-3.80	-3.49	-1.68	-2.55	-1.10
2015	37.20	37.20	37.11	37.15	38.51	-1.00	-1.11	-1.75	-1.55	-1.10
2016	37.20	37.20	37.02	37.20	38.57	0.00	-0.07	-1.82	-0.03	-1.10

estimates a fairly linear trend. QR captures the slight increase in the 95th percentiles of daily maximum temperature but cannot capture the fluctuations throughout the years. For the 5th percentile of daily minimum temperature, QR gives a flat prediction, similar to LP. SS and KS are able to capture some of the variability of daily minimum temperatures in Figure 2.2. In the case of LP in this figure, cross-validation produced a larger bandwidth than that for KS, giving the undersmoothed estimate.

In Table 2.4, the smoothing results for Kansas City are shown. In this instance, KS performs better at the boundary and interior points. As evidenced in the second row of plots in Figure 2.1 and 2.2, KS is superior to the other three methods. In cases where KS is superior to LP, this can again be attributed to the selection of the bandwidth through cross-validation. A large choice of bandwidth for LP will lead to a very flat estimate as in Figure 2.2 for Kansas City. Again, QR produces a flat estimate for the 95th percentile of temperatures, as there is no change in temperatures when looking at the endpoints of the time range, but some variability in the interior years. For the 5th percentile, QR produces a similar estimate to LP.

Table 2.4: Raw estimates($T_{.95}(t_j)$, $T_{.05}(t_j)$), Kernel smoothing estimates ($KS_{.95}(t_j)$, $KS_{.05}(t_j)$), local polynomial smoothing estimates ($LP_{.95}(t_j)$, $LP_{.05}(t_j)$), spline smoothing estimate ($SS_{.95}(t_j)$, $SS_{.05}(t_j)$), and quantile regression estimate ($QR_{.95}(t_j)$, $QR_{.05}(t_j)$) of 95th and 5th percentile temperature from Kansas City between 1990 and 2016.

t_j	$T_{.95}(t_j)$	$KS_{.95}(t_j)$	$LP_{.95}(t_j)$	$SS_{.95}(t_j)$	$QR_{.95}(t_j)$	$T_{.05}(t_j)$	$KS_{.05}(t_j)$	$LP_{.05}(t_j)$	$SS_{.05}(t_j)$	$QR_{.05}(t_j)$
1990	34.40	34.42	34.73	34.61	33.90	-7.80	-8.12	-9.64	-8.27	-10.50
1991	35.00	34.86	33.76	34.28	33.90	-11.58	-10.85	-9.72	-10.13	-10.53
1992	30.60	30.79	32.59	31.45	33.90	-6.10	-6.89	-9.80	-7.76	-10.57
1993	32.80	32.75	32.54	32.42	33.90	-10.60	-10.33	-9.87	-9.91	-10.60
1994	33.30	33.29	32.95	33.36	33.90	-11.70	-11.47	-9.94	-11.20	-10.63
1995	33.30	33.25	32.84	33.09	33.90	-9.88	-10.46	-10.01	-11.33	-10.66
1996	31.70	31.78	32.55	32.03	33.90	-15.45	-14.60	-10.07	-13.86	-10.70
1997	32.80	32.78	32.75	32.63	33.90	-10.26	-10.47	-10.13	-10.78	-10.73
1998	33.30	33.30	33.30	33.33	33.90	-7.68	-7.92	-10.19	-7.85	-10.76
1999	33.90	33.90	33.81	33.92	33.90	-8.20	-8.52	-10.26	-8.80	-10.79
2000	34.40	34.37	34.16	34.22	33.90	-12.80	-12.21	-10.32	-11.71	-10.83
2001	33.90	33.96	34.51	34.23	33.90	-9.88	-9.95	-10.39	-10.06	-10.86
2002	35.60	35.55	34.86	35.52	33.90	-7.80	-8.22	-10.46	-8.52	-10.89
2003	35.60	35.47	34.34	34.95	33.90	-11.10	-10.92	-10.53	-10.75	-10.92
2004	31.10	31.32	33.50	32.05	33.90	-12.08	-11.87	-10.61	-11.93	-10.96
2005	34.30	34.25	33.94	33.92	33.90	-10.48	-10.26	-10.69	-9.92	-10.99
2006	35.60	35.55	34.66	35.59	33.90	-6.10	-6.80	-10.77	-7.29	-11.02
2007	35.00	34.92	34.12	34.70	33.90	-10.60	-10.52	-10.87	-10.39	-11.05
2008	31.70	31.81	32.96	32.08	33.90	-14.13	-13.57	-10.96	-13.17	-11.09
2009	32.10	32.14	32.81	32.08	33.90	-10.60	-10.91	-11.07	-11.40	-11.12
2010	33.90	33.88	33.83	33.72	33.90	-11.00	-11.02	-11.18	-11.05	-11.15
2011	35.00	35.05	35.15	35.46	33.90	-11.70	-11.37	-11.30	-10.85	-11.18
2012	37.80	37.61	35.60	37.02	33.90	-8.18	-8.72	-11.43	-9.15	-11.22
2013	33.78	33.86	34.48	34.26	33.90	-11.60	-11.63	-11.56	-11.70	-11.25
2014	32.68	32.71	33.17	32.65	33.90	-15.38	-14.77	-11.71	-14.47	-11.28
2015	32.80	32.83	32.66	32.80	33.90	-11.48	-11.53	-11.86	-11.94	-11.31
2016	33.81	33.78	32.69	33.78	33.90	-8.20	-8.48	-12.02	-8.19	-11.35

The results for the third city, Miami, are shown in Table 2.5. For Miami, all methods capture a fairly flat trend in the 95th percentiles of temperature (third row of Figure 2.1). QR is best at the boundary years in this case, but slightly worse at interior points. For 5th percentiles of temperature, KS and LP are able to capture some trend in the yearly 5th percentiles, and SS is slightly undersmoothed. QR captures no trend over the 27 year time period.

Table 2.6 contains the results for Minneapolis. Similarly, refer to the fourth row of Figure 2.1 and 2.2. SS gives similar results to KS, while LP performs slightly worse. For the 95th percentile smoothing, LP gives a smoother curve while LP fails to capture a trend for the 5th percentile. QR captures an overall linear increase over the years.

For the 95th percentile for Portland (Table 2.7), KS performs the best. LP gives a less

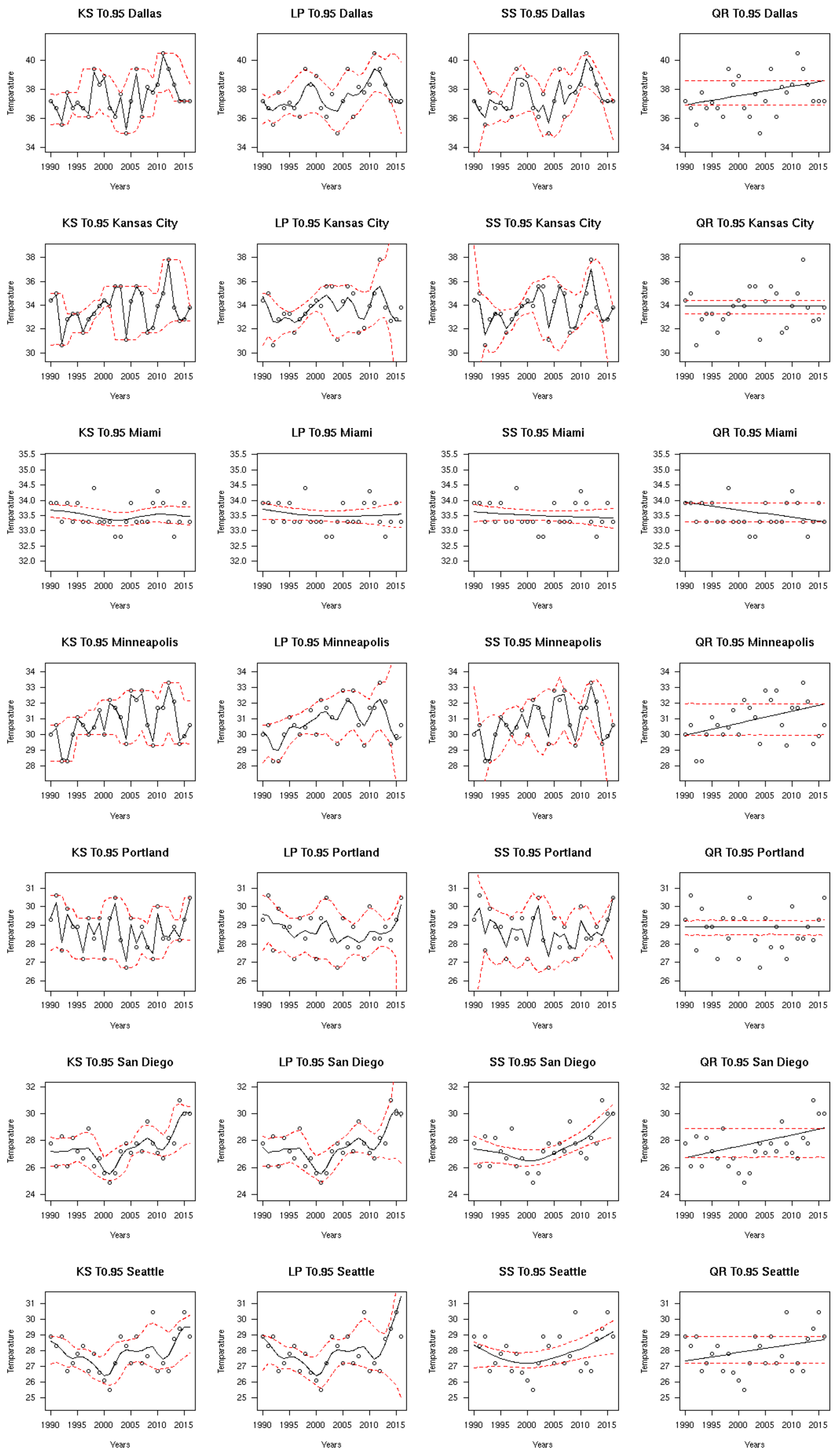


Figure 2.1: Smoothing Results for All Cities - 95th Percentile

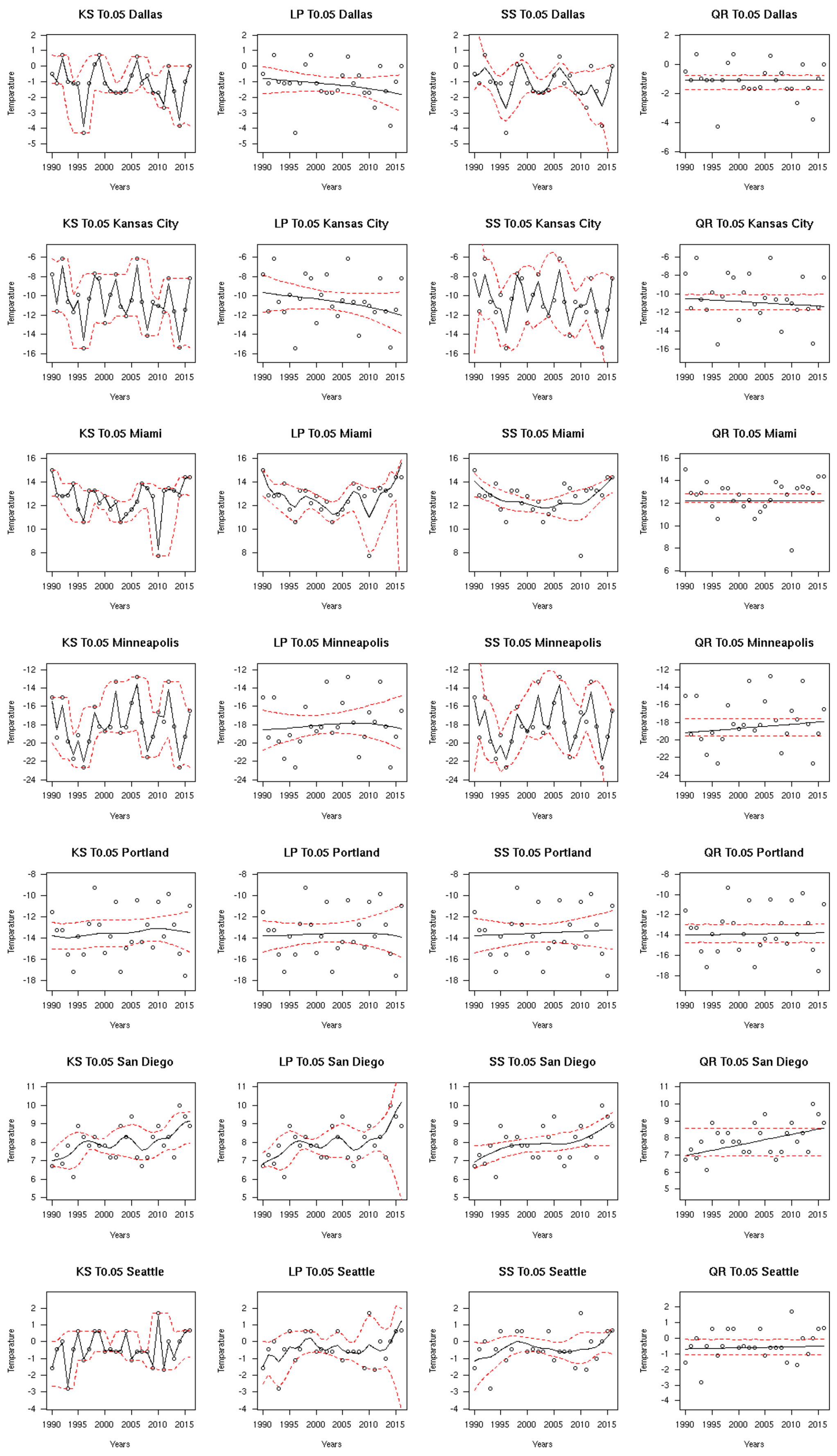


Figure 2.2: Smoothing Results for All Cities - 5th Percentile

Table 2.5: Raw estimates($T_{.95}(t_j)$, $T_{.05}(t_j)$), Kernel smoothing estimates ($KS_{.95}(t_j)$, $KS_{.05}(t_j)$), local polynomial smoothing estimates ($LP_{.95}(t_j)$, $LP_{.05}(t_j)$), spline smoothing estimate ($SS_{.95}(t_j)$, $SS_{.05}(t_j)$), and quantile regression estimate ($QR_{.95}(t_j)$, $QR_{.05}(t_j)$) of 95^{th} and 5^{th} percentile temperature from Miami between 1990 and 2016.

t_j	$T_{.95}(t_j)$	$KS_{.95}(t_j)$	$LP_{.95}(t_j)$	$SS_{.95}(t_j)$	$QR_{.95}(t_j)$	$T_{.05}(t_j)$	$KS_{.05}(t_j)$	$LP_{.05}(t_j)$	$SS_{.05}(t_j)$	$QR_{.05}(t_j)$
1990	33.90	33.68	33.70	33.61	33.93	15.00	14.90	14.83	14.09	12.20
1991	33.90	33.66	33.67	33.60	33.91	12.90	12.99	13.53	13.64	12.20
1992	33.30	33.64	33.64	33.59	33.88	12.80	12.81	13.03	13.25	12.20
1993	33.90	33.62	33.62	33.58	33.86	12.90	12.94	13.05	12.93	12.20
1994	33.30	33.60	33.59	33.57	33.83	13.90	13.75	12.87	12.68	12.20
1995	33.90	33.57	33.57	33.56	33.81	11.70	11.75	12.14	12.48	12.20
1996	33.30	33.54	33.55	33.55	33.78	10.60	10.78	11.87	12.38	12.20
1997	33.30	33.51	33.54	33.54	33.76	13.30	13.17	12.50	12.36	12.20
1998	34.40	33.47	33.52	33.53	33.74	13.30	13.25	12.84	12.35	12.20
1999	33.30	33.43	33.51	33.52	33.71	12.20	12.28	12.64	12.28	12.20
2000	33.30	33.39	33.50	33.51	33.69	12.80	12.72	12.38	12.17	12.20
2001	33.30	33.36	33.49	33.50	33.66	11.70	11.78	12.09	12.01	12.20
2002	32.80	33.35	33.48	33.50	33.64	12.32	12.21	11.72	11.88	12.20
2003	32.80	33.35	33.48	33.49	33.61	10.60	10.71	11.32	11.81	12.20
2004	33.30	33.36	33.48	33.48	33.59	11.25	11.24	11.34	11.84	12.20
2005	33.90	33.39	33.47	33.48	33.56	11.70	11.71	11.81	11.98	12.20
2006	33.30	33.43	33.47	33.47	33.54	12.32	12.37	12.55	12.14	12.20
2007	33.30	33.47	33.48	33.46	33.51	13.90	13.80	13.18	12.24	12.20
2008	33.30	33.51	33.48	33.46	33.49	13.45	13.44	13.02	12.24	12.20
2009	33.90	33.53	33.48	33.45	33.46	12.80	12.59	11.84	12.16	12.20
2010	34.30	33.54	33.49	33.45	33.44	7.80	8.30	11.03	12.16	12.20
2011	33.90	33.54	33.50	33.44	33.41	13.30	13.05	11.98	12.36	12.20
2012	33.30	33.53	33.51	33.44	33.39	13.45	13.44	13.03	12.69	12.20
2013	32.80	33.51	33.51	33.43	33.36	13.30	13.29	13.28	13.07	12.20
2014	33.30	33.49	33.52	33.43	33.34	12.90	12.99	13.47	13.48	12.20
2015	33.90	33.48	33.53	33.42	33.31	14.40	14.33	14.24	13.93	12.20
2016	33.30	33.46	33.54	33.42	33.29	14.40	14.40	15.58	14.38	12.20

pronounced, more smooth, estimate to the trend shown in the plot of Figure 2.1 (fifth row of plots). For the 5th percentile in Figure 2.2, all of the two-step procedures give similar smoothing results, with an almost linear fit. All nonparametric techniques are undersmoothed and thusly give similar results to QR.

Results for San Diego are in Table 2.8 with plots in the sixth row of Figures 2.1 and 2.2. The 95th percentiles show a general increase in temperatures from 1990 to 2016. All methods capture this overall trend, while KS and LP both detect a small decline around the year 2000. Similarly, the estimates for the 5th percentiles are consistent among all methods. There is again a very slight but steady in minimum temperatures over the 27 year span. All methods capture this, with the nonparametric methods giving slightly more robust estimates at interior years.

Lastly, the smoothing results for the final city (Seattle) are in Table 2.9, and the correspondingly in the last rows of Figures 2.1 and 2.2. Plots for 95th percentiles are similar to

Table 2.6: Raw estimates($T_{.95}(t_j)$, $T_{.05}(t_j)$), Kernel smoothing estimates ($KS_{.95}(t_j)$, $KS_{.05}(t_j)$), local polynomial smoothing estimates ($LP_{.95}(t_j)$, $LP_{.05}(t_j)$), spline smoothing estimate ($SS_{.95}(t_j)$, $SS_{.05}(t_j)$), and quantile regression estimate ($QR_{.95}(t_j)$, $QR_{.05}(t_j)$) of 95th and 5th percentile temperature from Minneapolis between 1990 and 2016.

t_j	$T_{.95}(t_j)$	$KS_{.95}(t_j)$	$LP_{.95}(t_j)$	$SS_{.95}(t_j)$	$QR_{.95}(t_j)$	$T_{.05}(t_j)$	$KS_{.05}(t_j)$	$LP_{.05}(t_j)$	$SS_{.05}(t_j)$	$QR_{.05}(t_j)$
1990	30.00	30.04	30.21	30.08	29.97	-15.00	-15.47	-18.54	-15.41	-19.18
1991	30.60	30.41	29.82	30.38	30.04	-19.40	-18.55	-18.53	-18.13	-19.13
1992	28.30	28.45	29.07	28.47	30.12	-15.00	-15.90	-18.51	-16.45	-19.08
1993	28.30	28.41	29.01	28.34	30.20	-19.88	-19.58	-18.47	-19.37	-19.04
1994	30.00	29.96	29.80	29.97	30.27	-21.70	-21.28	-18.43	-21.12	-18.99
1995	31.10	31.00	30.48	31.04	30.35	-19.18	-19.76	-18.38	-20.27	-18.94
1996	30.60	30.59	30.53	30.61	30.43	-22.65	-22.05	-18.33	-21.82	-18.89
1997	30.00	30.07	30.41	30.04	30.50	-19.88	-19.78	-18.27	-19.76	-18.84
1998	30.48	30.52	30.62	30.57	30.58	-16.10	-16.67	-18.21	-16.82	-18.80
1999	31.58	31.41	30.87	31.29	30.66	-18.20	-18.05	-18.15	-17.87	-18.75
2000	30.00	30.24	31.04	30.38	30.73	-18.75	-18.65	-18.09	-18.82	-18.70
2001	32.20	32.03	31.43	31.93	30.81	-18.30	-17.86	-18.04	-17.48	-18.65
2002	31.70	31.69	31.46	31.82	30.88	-13.30	-14.33	-17.99	-14.81	-18.60
2003	31.10	31.03	30.99	30.87	30.96	-18.90	-18.30	-17.95	-17.97	-18.56
2004	29.40	29.73	30.95	29.85	31.04	-18.30	-18.10	-17.91	-18.27	-18.51
2005	32.80	32.54	31.72	32.37	31.11	-15.60	-15.59	-17.88	-15.38	-18.46
2006	32.20	32.28	32.23	32.49	31.19	-12.80	-13.55	-17.86	-13.64	-18.41
2007	32.80	32.62	31.91	32.58	31.27	-17.80	-17.68	-17.86	-17.60	-18.36
2008	30.60	30.66	30.98	30.62	31.34	-21.55	-20.97	-17.86	-20.98	-18.31
2009	29.30	29.54	30.56	29.55	31.42	-19.30	-19.27	-17.88	-19.45	-18.27
2010	31.70	31.55	31.15	31.39	31.50	-16.70	-17.05	-17.91	-17.28	-18.22
2011	31.70	31.80	31.96	31.96	31.57	-17.68	-17.16	-17.95	-16.57	-18.17
2012	33.30	33.12	32.28	33.14	31.65	-13.30	-14.20	-18.01	-14.46	-18.12
2013	32.10	32.00	31.57	32.00	31.73	-18.20	-18.16	-18.09	-18.13	-18.07
2014	29.40	29.60	30.30	29.62	31.80	-22.70	-21.94	-18.18	-21.88	-18.03
2015	29.88	29.90	29.63	29.79	31.88	-19.30	-19.36	-18.29	-19.75	-17.98
2016	30.60	30.55	29.84	30.60	31.95	-16.51	-16.81	-18.42	-16.49	-17.93

Table 2.7: Raw estimates($T_{.95}(t_j)$, $T_{.05}(t_j)$), Kernel smoothing estimates ($KS_{.95}(t_j)$, $KS_{.05}(t_j)$), local polynomial smoothing estimates ($LP_{.95}(t_j)$, $LP_{.05}(t_j)$), spline smoothing estimate ($SS_{.95}(t_j)$, $SS_{.05}(t_j)$), and quantile regression estimate ($QR_{.95}(t_j)$, $QR_{.05}(t_j)$) of 95th and 5th percentile temperature from Portland between 1990 and 2016.

t_j	$T_{.95}(t_j)$	$KS_{.95}(t_j)$	$LP_{.95}(t_j)$	$SS_{.95}(t_j)$	$QR_{.95}(t_j)$	$T_{.05}(t_j)$	$KS_{.05}(t_j)$	$LP_{.05}(t_j)$	$SS_{.05}(t_j)$	$QR_{.05}(t_j)$
1990	29.30	29.42	29.59	29.51	28.90	-11.58	-13.79	-13.81	-13.81	-14.01
1991	30.60	30.24	29.52	29.92	28.90	-13.30	-13.89	-13.80	-13.79	-14.00
1992	27.65	28.09	29.08	28.53	28.90	-13.30	-13.96	-13.80	-13.77	-13.99
1993	29.88	29.61	29.07	29.31	28.90	-15.60	-13.99	-13.79	-13.75	-13.98
1994	28.90	28.98	28.98	29.15	28.90	-17.20	-13.97	-13.77	-13.73	-13.98
1995	28.90	28.76	28.56	28.56	28.90	-13.90	-13.90	-13.75	-13.70	-13.97
1996	27.20	27.53	28.32	27.81	28.90	-15.60	-13.81	-13.73	-13.68	-13.96
1997	29.40	29.12	28.56	28.82	28.90	-12.68	-13.71	-13.71	-13.66	-13.95
1998	28.30	28.48	28.72	28.77	28.90	-9.30	-13.63	-13.69	-13.64	-13.94
1999	29.40	29.12	28.60	28.78	28.90	-12.80	-13.58	-13.67	-13.62	-13.93
2000	27.20	27.57	28.52	27.88	28.90	-15.45	-13.57	-13.64	-13.60	-13.92
2001	29.40	29.31	29.04	29.26	28.90	-13.90	-13.58	-13.62	-13.58	-13.91
2002	30.48	30.20	29.26	30.06	28.90	-10.60	-13.60	-13.60	-13.56	-13.91
2003	28.20	28.27	28.52	28.30	28.90	-17.20	-13.61	-13.58	-13.54	-13.90
2004	26.70	27.05	28.01	27.30	28.90	-15.00	-13.58	-13.57	-13.52	-13.89
2005	29.40	29.04	28.26	28.63	28.90	-14.40	-13.53	-13.56	-13.50	-13.88
2006	27.80	28.03	28.39	28.36	28.90	-10.48	-13.44	-13.55	-13.48	-13.87
2007	28.90	28.72	28.30	28.53	28.90	-14.40	-13.34	-13.55	-13.46	-13.86
2008	27.80	27.84	28.05	27.81	28.90	-12.80	-13.24	-13.55	-13.44	-13.85
2009	27.20	27.49	28.19	27.75	28.90	-14.88	-13.17	-13.56	-13.42	-13.84
2010	30.00	29.62	28.68	29.28	28.90	-10.60	-13.14	-13.58	-13.40	-13.84
2011	28.30	28.44	28.68	28.65	28.90	-13.90	-13.14	-13.61	-13.38	-13.83
2012	28.30	28.35	28.53	28.37	28.90	-9.85	-13.19	-13.64	-13.36	-13.82
2013	28.90	28.79	28.58	28.63	28.90	-12.80	-13.26	-13.69	-13.34	-13.81
2014	28.20	28.35	28.70	28.44	28.90	-15.50	-13.35	-13.75	-13.32	-13.80
2015	29.30	29.31	29.17	29.23	28.90	-17.58	-13.44	-13.83	-13.30	-13.79
2016	30.48	30.37	30.11	30.47	28.90	-11.00	-13.53	-13.92	-13.27	-13.78

that of San Diego above, with all methods detecting an increasing trend in yearly maximum temperatures. In the plot for the 5th percentiles, there is a less distinct pattern among yearly temperatures, reflected by the results of LP, SS and QR. KS creates a very rugged smoothing to the yearly temperatures and QR shows no trend at all.

Table 2.8: Raw estimates($T_{.95}(t_j)$, $T_{.05}(t_j)$), Kernel smoothing estimates ($KS_{.95}(t_j)$, $KS_{.05}(t_j)$), local polynomial smoothing estimates ($LP_{.95}(t_j)$, $LP_{.05}(t_j)$), spline smoothing estimate ($SS_{.95}(t_j)$, $SS_{.05}(t_j)$), and quantile regression estimate ($QR_{.95}(t_j)$, $QR_{.05}(t_j)$) of 95th and 5th percentile temperature from San Diego between 1990 and 2016.

t_j	$T_{.95}(t_j)$	$KS_{.95}(t_j)$	$LP_{.95}(t_j)$	$SS_{.95}(t_j)$	$QR_{.95}(t_j)$	$T_{.05}(t_j)$	$KS_{.05}(t_j)$	$LP_{.05}(t_j)$	$SS_{.05}(t_j)$	$QR_{.05}(t_j)$
1990	27.80	27.24	27.51	27.40	26.74	6.70	6.99	6.83	6.94	6.96
1991	26.10	27.12	27.12	27.34	26.82	7.32	7.06	7.01	7.10	7.02
1992	28.30	27.19	27.19	27.29	26.91	6.83	7.15	7.13	7.25	7.08
1993	26.10	27.22	27.22	27.22	26.99	7.80	7.27	7.27	7.39	7.15
1994	28.20	27.36	27.36	27.15	27.08	6.10	7.51	7.51	7.53	7.21
1995	27.20	27.36	27.36	27.06	27.16	8.90	7.81	7.81	7.65	7.27
1996	26.70	27.40	27.40	26.95	27.25	8.30	8.03	8.01	7.75	7.33
1997	28.90	27.43	27.43	26.83	27.33	7.80	8.07	8.06	7.82	7.39
1998	26.10	26.93	26.93	26.69	27.41	8.30	7.99	7.98	7.87	7.46
1999	26.70	26.31	26.31	26.58	27.50	7.80	7.85	7.85	7.89	7.52
2000	25.60	25.75	25.75	26.50	27.58	7.80	7.71	7.72	7.91	7.58
2001	24.88	25.50	25.50	26.50	27.67	7.20	7.68	7.70	7.91	7.64
2002	25.60	25.96	25.96	26.57	27.75	7.20	7.86	7.87	7.92	7.71
2003	27.20	26.81	26.81	26.70	27.83	8.90	8.15	8.14	7.93	7.77
2004	27.80	27.33	27.33	26.88	27.92	8.30	8.31	8.29	7.93	7.83
2005	27.10	27.45	27.45	27.06	28.00	9.40	8.16	8.14	7.92	7.89
2006	27.80	27.58	27.58	27.24	28.09	7.20	7.80	7.81	7.91	7.96
2007	27.20	27.91	27.91	27.42	28.17	6.70	7.55	7.57	7.91	8.02
2008	29.40	28.22	28.22	27.60	28.26	7.20	7.63	7.65	7.95	8.08
2009	27.80	27.92	27.92	27.77	28.34	8.30	7.93	7.92	8.02	8.14
2010	27.10	27.38	27.38	27.96	28.42	8.90	8.14	8.13	8.12	8.21
2011	26.70	27.33	27.33	28.20	28.51	7.80	8.19	8.19	8.24	8.27
2012	28.18	27.84	27.84	28.50	28.59	8.30	8.25	8.26	8.38	8.33
2013	27.80	28.74	28.75	28.86	28.68	7.20	8.48	8.53	8.55	8.39
2014	31.00	29.72	29.85	29.26	28.76	10.00	8.83	9.07	8.73	8.46
2015	30.00	30.11	30.33	29.66	28.85	9.40	9.08	9.70	8.92	8.52
2016	30.00	30.07	29.76	30.06	28.93	8.90	9.15	10.15	9.10	8.58

In most cases, when comparing the smoothing results for the two-step methods, there is usually agreement between at least two of the three methods. In some instances, either LP or SS produces an undersmoothed result wile the other will agree with KS and give a more rugged fit to the yearly data. Unlike in the simulations, KS does perform strongly at boundary points in the application of the methods on the yearly temperatures, and even better than LP in some cases. LP gives smoother estimates than KS, as a direct consequence of the bandwidth selection process. Even when KS gives a more rugged smoothing estimate, the 95% bootstrap confidence bands are almost identical to those from LP. SS typically gives the most undersmoothed results, as in the plots fo San Diego and Seattle. Consistently, QR performs the worst, with the largest deviations from the raw estimates of

Table 2.9: Raw estimates($T_{.95}(t_j)$, $T_{.05}(t_j)$), Kernel smoothing estimates ($KS_{.95}(t_j)$, $KS_{.05}(t_j)$), local polynomial smoothing estimates ($LP_{.95}(t_j)$, $LP_{.05}(t_j)$), spline smoothing estimate ($SS_{.95}(t_j)$, $SS_{.05}(t_j)$), and quantile regression estimate ($QR_{.95}(t_j)$, $QR_{.05}(t_j)$) of 95th and 5th percentile temperature from Seattle between 1990 and 2016.

t_j	$T_{.95}(t_j)$	$KS_{.95}(t_j)$	$LP_{.95}(t_j)$	$SS_{.95}(t_j)$	$QR_{.95}(t_j)$	$T_{.05}(t_j)$	$KS_{.05}(t_j)$	$LP_{.05}(t_j)$	$SS_{.05}(t_j)$	$QR_{.05}(t_j)$
1990	28.90	28.62	28.85	28.34	27.36	-1.58	-1.55	-1.48	-1.18	-0.68
1991	28.30	28.43	28.54	28.16	27.41	-0.48	-0.50	-0.81	-1.03	-0.67
1992	28.90	28.06	28.09	27.99	27.46	0.00	-0.09	-0.90	-0.96	-0.67
1993	26.70	27.64	27.64	27.82	27.52	-2.80	-2.66	-1.23	-0.91	-0.66
1994	27.20	27.49	27.49	27.67	27.57	-0.48	-0.51	-0.79	-0.72	-0.65
1995	27.80	27.60	27.60	27.53	27.62	0.60	0.53	-0.32	-0.49	-0.65
1996	28.30	27.61	27.61	27.41	27.68	-1.10	-1.04	-0.41	-0.32	-0.64
1997	26.70	27.41	27.41	27.32	27.73	-0.48	-0.47	-0.25	-0.14	-0.63
1998	27.80	27.11	27.11	27.24	27.78	0.60	0.57	0.17	-0.01	-0.62
1999	26.58	26.71	26.71	27.20	27.84	0.60	0.57	0.18	-0.01	-0.62
2000	26.10	26.39	26.39	27.20	27.89	-0.60	-0.56	-0.21	-0.14	-0.61
2001	25.48	26.53	26.53	27.23	27.94	-0.48	-0.49	-0.47	-0.28	-0.60
2002	27.20	27.19	27.19	27.29	27.99	-0.60	-0.60	-0.50	-0.36	-0.60
2003	28.90	27.85	27.85	27.38	28.05	-0.60	-0.57	-0.33	-0.39	-0.59
2004	28.30	28.06	28.06	27.47	28.10	0.60	0.52	-0.25	-0.43	-0.58
2005	27.20	28.01	28.01	27.57	28.15	-1.10	-1.04	-0.51	-0.52	-0.58
2006	28.90	27.97	27.97	27.68	28.21	-0.60	-0.61	-0.66	-0.59	-0.57
2007	27.20	28.01	28.01	27.78	28.26	-0.60	-0.60	-0.67	-0.62	-0.56
2008	27.65	28.21	28.21	27.89	28.31	-0.60	-0.63	-0.72	-0.61	-0.56
2009	30.48	28.26	28.25	28.01	28.36	-1.58	-1.47	-0.49	-0.54	-0.55
2010	26.70	27.83	27.83	28.13	28.42	1.70	1.52	-0.16	-0.45	-0.54
2011	27.20	27.46	27.46	28.28	28.47	-1.70	-1.56	-0.41	-0.47	-0.53
2012	26.70	27.68	27.69	28.44	28.52	0.00	-0.07	-0.56	-0.43	-0.53
2013	28.78	28.42	28.47	28.63	28.58	-1.00	-0.95	-0.45	-0.30	-0.52
2014	29.40	29.17	29.47	28.83	28.63	0.00	-0.01	-0.04	-0.04	-0.51
2015	30.48	29.53	30.47	29.03	28.68	0.60	0.59	0.61	0.30	-0.51
2016	28.90	29.51	31.46	29.23	28.73	0.68	0.67	1.25	0.66	-0.50

desired yearly quantiles for both minimum and maximum temperatures. Many times QR gives a flat regression line, since there is very little variability in the starting and ending points in the data when simply comparing the temperatures from 1990 to 2016. Even so, however, there is some cyclical variability within the middle years which is then not captured by QR. The nonparametric methods give more robust estimates for some of these patterns in the interior points of the data. Similarly, for some cities there is a slight but clear increase in temperatures over the years, and QR correctly detects this, but again does not pick up some of the nuanced trends within the years. Overall, quantile regression does not make good approximations of the extreme quantiles when time-variant quantiles vary. To overcome these estimation problems, two-step estimation procedures should be adopted.

To improve upon the two-step procedures described above, the next chapter will introduce the first of the one-step procedures in this paper. The purpose is again to estimate a time variant parameter and smooth it over time. The one-step procedure will be implemented by assuming a parametric family in each time interval. This will be shown in the next chapter with the development of the kernel log likelihood one-step estimator for a discrete distribution.

Chapter 3

One-Step Estimation with Discrete Distribution

After developing these three two-step techniques, we now propose a one-step smoothing technique which does not require computing a raw estimate to then be smoothed. The previous chapter showed that the two-step techniques perform better than a traditional parametric technique of quantile regression for the 5% and 95% quantiles of temperature. Here we will rely on data which can be approximated to a parametric distribution to create a smoothed estimate through a combination of maximum likelihood estimation and a kernel function. First, this will be demonstrated with a discrete distribution.

3.1 Maximum Likelihood Estimation

In maximum likelihood estimation (MLE), an unknown parameter θ can be estimated from some sample data. Assume there are n independent samples $(x_1, x_2, ..., x_n)$ from a distribution $f(x|\theta)$. For a discrete random variable, $f(x|\theta)$ is a probability mass function, and $f(x|\theta)$ is a probability density function for a continuous random variable. In cases with a distribution which has more than one parameters, $\boldsymbol{\theta}$ can be used to denote a vector of unknown parameters. The likelihood function becomes the joint probability (mass or density) function of the observed data. Since the observed data is independent, we can multiply each of their individual likelihoods to create the likelihood function $L(\theta)$.

$$L(\theta) = f(x_1, x_2, ..., x_n|\theta) = \prod_{i=1}^{n} f(x_i, \theta)$$

We will refer to $L(\theta)$ as simply L. To find the value of θ which maximizes L, the logarithm of the likelihood function can be taken since the logarithm is a strictly increasing function, so the likelihood function achieves a maximum at the same point as the logarithm of the likelihood function (Hogg and Tanis, 2008). This also simplifies the calculation of the value of θ, as the product becomes a summation after taking the logarithm.

$$log(L) = \sum_{i=1}^{n} f(x_i, \theta)$$

Adding in a kernel function, we now have the proposed one-step smoothing solution:

$$log(L) = \sum_{i=1}^{n} f(x_i, \theta) K_h(t_i - t)$$

The one-step smoothing technique will also be referred to as the kernel log-likelihood method. We will explore an example of kernel log-likelihood with the geometric distribution in the next section.

3.2 Kernel Log-Likelihood with Geometric Distribution

Kernel log-likelihood differs by the probability models since there are a number of discrete and continuous probability models. We will consider one discrete (Geometric Model) and one continuous (Gaussian Model) probability model for one-step Kernel log-likelihood smoothing estimation of the time-variant parameter. The geometric distribution models the number of Bernoulli trials until the first success is achieved. We will first show the derivation of the log-likelihood function for the geometric distribution. A geometric distribution with x failures and probability of success θ can be formulated as:

$$P(X = x) = \theta(1 - \theta)^x \quad \text{for x = 0,1,2,...}$$

The probability mass function above will also be referred to as $f(x|\theta)$. If there are n independent samples $(x_1, x_2, ..., x_n)$ from the geometric distribution with parameter θ, we can take their product to create the likelihood function.

$$L = f_n(\boldsymbol{x}|\theta) = \textstyle\prod_{i=1}^{n} f(x_i, \theta)$$

The product of the individual geometric samples can be simplified as such, given that there are n total successes and by summing the total number of failures.

$$L = \theta^n (1-\theta)^{\sum_{i=1}^{n} x_i}$$

By taking the logarithm and with some simplification, we have:

$$\begin{gathered} \log(L) = n\log(\theta) + (\textstyle\sum_{i=1}^{n} x_i)\log(1-\theta) \\ \log(L) = \textstyle\sum_{i=1}^{n} \log(\theta) + \sum_{i=1}^{n} x_i \log(1-\theta) \\ \log(L) = \textstyle\sum_{i=1}^{n} \{\log(\theta) + x_i \log(1-\theta)\} \end{gathered}$$

Adding in the kernel function to the log-likelihood function for the geometric distribution gives the following equation for the one-step procedure:

$$\sum_{i=1}^{n} \{\log(\theta) + x_i \log(1-\theta)\} K_h(t_i - t) \tag{10}$$

The summation is over all data points x_i, $i = 1, 2, \ldots, n$. However, the data used in the examples of upcoming sections is split by time, so several x_i will have the same corresponding t_i. This means that all x_i with the same t_i will be given the same weight. In turn, the summation over all the data can also be thought of as a summation across the discrete time points in the data. For each unique t_i, the number of failures (sum of x_i) and the number of successes (number of x_i's at the unique t_i) is all that will be needed for Equation 10, making this procedure a local MLE at each time point.

3.2.1 Choice of Kernel Function

Before detailing the implementation of Equation 10, we will first discuss the role of the kernel function in more detail, and how the bandwidth h is selected. Gaussian and Epanechnikov kernels are commonly used kernel functions. A key difference between these kernels is that the Gaussian kernel gives non-zero probability to all data, as there is still density in the tails of the normal distribution. The Epanechnikov kernel is a quadratic function in which data within an interval determined by the bandwidth choice has non-zero probability, but all other data is given zero weight. The choice of kernel function is not as important as the choice of the bandwidth, which will be discussed in the next section. The Epanechnikov kernel has been shown to yield the lowest mean integrated square error (Wand and Jones, 1994), and is shown below.

$$K(u) = \begin{cases} \frac{3}{4}(1-u^2) & |u| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

3.2.2 Choice of Bandwidth

Within the kernel function, a bandwidth h is also required. The selection of the bandwidth dictates the size of the window which is used in the local estimation of the unknown regression function. A large choice for the bandwidth lead to smoother estimates, while a smaller choice for the bandwidth results in spiky, rough estimates. This is demonstrated in Figure 3.1. For a very small bandwidth (h=0.03), the estimated regression function goes through each data point. Relaxing the bandwidth slightly to h=0.1, the overall trend is still captured very will, but a smoother estimate is created. Choosing a bandwidth which is the size of domain of the sample data leads to a linear fit with no curvature.

In each time period, the bandwidth dictates the window of time to use in calculation of the estimate. For example, for a bandwidth of $h = 2$, data within $-2 < t_j - t < 2$ has non-zero weight. Here, t_j is the current time period which is being estimated. Any data which falls within 2 time periods of t_j will be used in estimating the value at time t_j (e.g. if $t_j = 8$, then data from time periods 6 through 10 will be used). The weights for an Epanechnikov

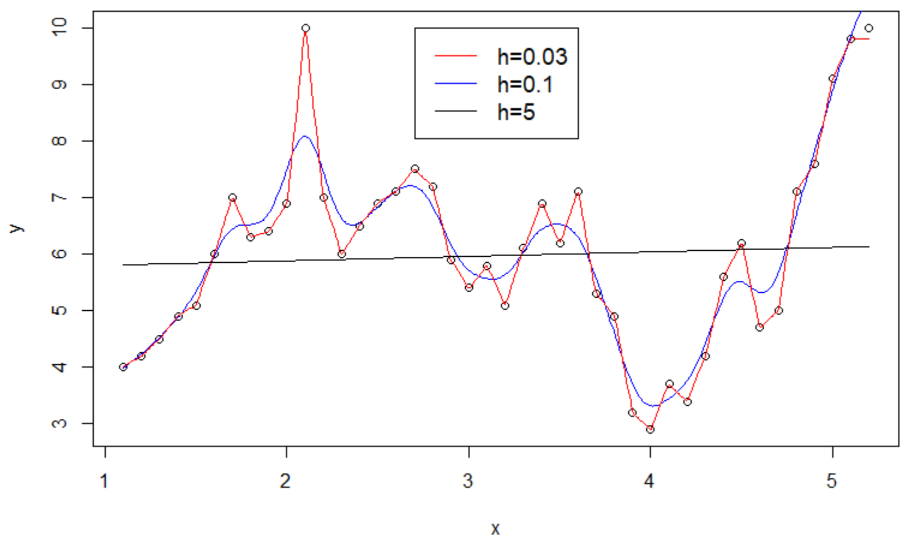


Figure 3.1: Effect of Bandwidth on Smoothing of Regression Function

kernel can be computed as such:

$$K\Big(\frac{t_j - t}{h}\Big) = \frac{3}{4}\Big(1 - \Big(\frac{t_j - t}{h}\Big)^2\Big)$$

Using the above function, weights can be computed when solving the kernel log likelihood function in equation 10. For example, for a bandwidth of $h = 2$, the following matrix shows the weights which are assigned to data in computation of the kernel log likelihood function.

$$\text{k.weights.matrix} = \begin{array}{c} \\ t1 \\ t2 \\ t3 \\ t4 \\ t5 \end{array} \begin{array}{c} \begin{array}{ccccc} t1 & t2 & t3 & t4 & t5 \end{array} \\ \begin{pmatrix} 0.75 & 0.5625 & 0 & 0 & 0 \\ 0.5625 & 0.75 & 0.5625 & 0 & 0 \\ 0 & 0.5625 & 0.75 & 0.5625 & 0 \\ 0 & 0 & 0.5625 & 0.75 & 0.5625 \\ 0 & 0 & 0 & 0.5625 & 0.75 \end{pmatrix} \end{array}$$

In this example above, assume there are 5 time periods t_1 to t_5. Here we see that for estimating θ at time period t_1, data from time periods 1 and 2 are used, with corresponding weights shown in row 1 of the matrix. This results in a local MLE estimate for time period t_1. In estimating θ for time period t_2, data from time periods 1, 2 and 3 are used with the corresponding weights shown in row 2, resulting in a local MLE estimate for time period t_2. In computation of θ in Equation 10, the respective row from this weights matrix must be supplemented.

3.2.3 Selection of Optimal Bandwidth

To assess how well the one-step approach compares to the two-step approach, estimations are compared to the original raw estimates. In the two-step approach, the raw estimates are directly smoothed so it is intuitive to compare the results of the smoothed values to the raw estimates. In the one-step approach, no raw estimate is computed as all the original data is used to create the estimates. However, we know that from regular maximum likelihood estimation, the estimate at each time period would be the same as the raw estimate used in the two-step technique. In the case of a geometric distribution, the raw estimate is equal to $\hat{\theta} = \frac{1}{\bar{x}}$. Even with the one-step approach where a kernel is introduced, a bandwidth of $h = 1$ would yield a local MLE which is equal to the MLE of the geometric distribution. This can be checked by inspecting the kernel weights matrix for when $h = 1$, shown below.

$$\text{k.weights.matrix} = \begin{array}{c} \\ t1 \\ t2 \\ t3 \\ t4 \\ t5 \end{array} \begin{array}{c} \begin{array}{ccccc} t1 & t2 & t3 & t4 & t5 \end{array} \\ \begin{pmatrix} 0.75 & 0 & 0 & 0 & 0 \\ 0 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 0.75 & 0 & 0 \\ 0 & 0 & 0 & 0.75 & 0 \\ 0 & 0 & 0 & 0 & 0.75 \end{pmatrix} \end{array}$$

At each time period, only the data from that time period has non-zero weight, so the local

MLE with kernel is the same as regular MLE. For consistency and through properties of MLE, the results of the one-step smoothing approach should be compared to the same raw estimates as the two-step approach.

The choice of bandwidth has a direct effect on the smoothness of the fitted curves. When $h = 1$, the results from the one-step procedure perfectly fits the raw estimates, and this produces a very unsmooth curve. For very large bandwidths, the fitted curve becomes very linear and has no curvature. We must choose a bandwidth in such a way which balances this tradeoff.

To select the optimal bandwidth which minimizes the error between the smoothed points and raw estimate, we can perform leave-one-out cross validation (LOOCV). In the two-step estimation, this would entail iteratively removing a time point t_j and smoothing the remaining time points and computing the error of the estimate at t_j. This is performed for each bandwidth h, and the bandwidth which results in the minimum cross validation score is chosen. For time series data where the series may be around 30 time points, this is not computationally expensive. The equation for this process is shown below.

$$CV(h) = \textstyle\sum_{j=1}^{J} \frac{1}{N}[Y_j - \xi_j^{(-j)}]^2$$

In one-step estimation, we can still perform cross validation in this fashion, but instead of leaving one observation out, we can leave the entire time period of observations. Since the one-step approach uses all the data at each time point, this is equivalent to the two-step approach where an entire time point is removed (there is only one data point at each time, the raw estimate, in the two-step approach).

3.3 Algorithm

To implement the one-step procedure for the discrete case, the original data should be summarized by time period. The original data should contain the number of failures until the first success for each observation and be split by the time period. Based on Equation 10, we just need to know the number of successes (which is equal to the sample size) and total number of failures at each time period. The raw estimates should also be computed, which are just the probability (i.e. $\frac{1}{\bar{x}}$) at each time period based on the sample data. The kernel weights matrix (referred to as k.weights.matrix) is computed within each iteration for each specific bandwidth to check. The algorithm checks all bandwidths within some interval (predefined to check for h in the range (1,8) in intervals of 0.05) and returns the bandwidth with minimum cross validation error. This algorithm is detailed below in Algorithm 1. Algorithm 1 requires a function *Geom.theta* to be optimized in each iteration. This function

Algorithm 1 Bandwidth Selection for Geometric Kernel Log Likelihood

```
1: procedure BANDWIDTH SELECTION - GEOMETRIC
2:     $n$ is a list of sample size at each time period
3:     $x$ is a list of number of failures at each time period
4:     $tp$ is a list of time points
5:     $y$ is a list of raw estimates (probability at each tp)

6:     $bw \leftarrow seq(1, 10, by = 0.05)$          ▷ Assign list of bandwidths to be checked
7:     $opt.h \leftarrow 0$                          ▷ Initialize list to track error from LOOCV

8:     for h in bw do
9:         $tot \leftarrow 0$                        ▷ Initialize error to 0 each iteration
10:        Compute $k.weights.matrix$ based on current h

11:        for k in tp do
12:            $n[k] \leftarrow 0$                   ▷ Remove current tp for LOOCV
13:            $x[k] \leftarrow 0$                   ▷ Remove current tp for LOOCV
14:            $t \leftarrow$ optimize Geom.theta function
                                        ▷ Geom.theta(theta,n,x,tp,k.weights.matrix,k,h)
                                        ▷ Geom.theta function returns theta based on MLE
15:            $tot \leftarrow tot + (y[k] - t)^2$   ▷ Compute MSE from estimated theta
16:        Append $tot$ to $opt.h$
17:    return $h$ with minimum error
```

computes the local MLE calculations and returns the optimal θ based on geometric MLE. This function is detailed in Algorithm 2.

Algorithm 2 Cross Validation Function for Geometric Kernel Log Likelihood

```
1: function GEOM.THETA(theta,n,x,tp,k.weights.matrix,k,h)
2:     for j in tp do                                          ▷ Initialization
3:         $logs[j] \leftarrow 0$
4:     for j in tp do
5:         $logs[j] \leftarrow n * \log\theta + x * \log(1-\theta)$          ▷ geometric MLE
6:     return $logs \cdot k.weights.matrix$[:,cv]   ▷ Multiply by column $cv$ of weights matrix
```

3.3.1 Alternate Solution

The kernel log-likelihood function from Section 3.2 has a closed form solution for the local constant (degree 0 polynomial) expansion of θ. The expression in Equation 10 (also shown below) can be rewritten from a summation of logarithms the logarithm of a power:

$$\begin{gathered}\textstyle\sum_{i=1}^{n}\{\log(\theta)+x_i\log(1-\theta)\}K_h(t_i-t)\\ =\log(\theta^{\sum_{i=1}^{n}K_{h_i}})+\log[(1-\theta)^{\sum_{i=1}^{n}x_iK_{h_i}}]\\ =\log[\theta^{\sum_{i=1}^{n}K_{h_i}}(1-\theta)^{\sum_{i=1}^{n}x_iK_{h_i}}],\end{gathered}$$

where K_{h_i} is used to denote the kernel $K_h(t_i-t)$. Being a strictly increasing function, maximizing the previous expression is equivalent to maximizing

$$\theta^{\sum_{i=1}^{n}K_{h_i}}(1-\theta)^{\sum_{i=1}^{n}x_iK_{h_i}}$$

The solution to the above expression is straightforward. The problem of trying to maximize $x^a(1-x)^b$, provided that $x\in[0,1]$ is shown below

$$\underset{x}{\text{maximize}}\ x^a(1-x)^b.$$

Differentiating, and solving for x gives:

$$ax^{a-1}(1-x)^b-bx^a(1-x)^{b-1}=0$$

$$\begin{gathered}ax^{a-1}(1-x)^b=bx^a(1-x)^{b-1}\\ a(1-x)=bx\\ a-ax=bx\\ a=ax+bx\end{gathered}$$

$$a = x(a+b)$$

$$x = \frac{a}{a+b}$$

Going back to the original problem, letting $a = \sum_{i=1}^{n} K_{h_i}$ and $b = \sum_{i=1}^{n} x_i K_{h_i}$, the solution to θ is then a ratio of the exponents, which are the kernel weights.

$$\hat{\theta} = \frac{\sum K_{h_i}}{\sum K_{h_i} + \sum x_i K_{h_i}}$$

3.4 Polynomial Expansion in Kernel Log Likelihood

We will next consider higher order polynomial expansions of $\theta(t)$. This would entail choosing the parameters of a local polynomial to maximize

$$\sum_{i=1}^{n} \left\{ log\left(\sum_{q=0}^{p} \beta_q(t)(t_i - t)^q \right) + (y_i) log\left(1 - \sum_{q=0}^{p} \beta_q(t)(t_i - t)^q \right) \right\} K_h(t_i - t) \qquad (11)$$

Here $\beta_q(t) = \frac{\theta^{(q)}(t)}{q!}; 0 \leq q \leq p$. The form of the equation is very similar to Equation 10. In fact, Equation 10 is identical to Equation 11 when the order of polynomial is taken to be $p = 0$. In the polynomial expansion of $\theta(t)$, we try up to a degree 5 expansion in both the application and simulation exercises. The method remains similar to Algorithms 1 and 2, with the inclusion of additional parameters in the optimization. These additional parameters represent the terms of the polynomial expansion of $\theta(t)$, with β_0 representing the estimate of $\theta(t)$. Then β_1 can be treated as the estimate of $\theta'(t)$, and in a similar fashion, β_2 is the estimate of $\theta''(t)/2!$, and so on. To accommodate for the extra parameters, the algorithms in Section 3.3 are augmented with a multivariate objective function. Additionally, several constraints are added in the optimization, which are detailed below. Matlab version 2018a is used to perform the constrained optimization with the *fmincon* function.

The objective function of the polynomial expansion problem is the one written in Equation 11. Note that this is a maximization problem, and some solvers by default are for minimization problems. Any maximization problem can be converted to a minimization problem by taking the negative of the objective function (i.e. maximizing a quantity is equivalent to minimizing the negative of that quantity). For completeness, constraints are added to the multivariate optimization in Equation 13. The constraints ensure that the quantity in the expansion of $\theta(t) \in [0,1]$. There is one additional constraint to ensure that $\beta_0 \in [0,1]$, as this is the coefficient used as the final estimate of $\theta(t)$.

$$\begin{aligned}
\underset{\beta_q}{\text{maximize}} \quad & \sum_{i=1}^{n} \left\{ log\left(\sum_{q=0}^{p} \beta_q(t)(t_i - t)^q \right) + (y_i) log\left(1 - \sum_{q=0}^{p} \beta_q(t)(t_i - t)^q \right) \right\} K_h(t_i - t) \\
\text{subject to} \quad & \sum_{q=0}^{p} \beta_q(t)(t_i - t)^q \leq 1, \ i \in 1, \ldots, n. \\
& \beta_0 \in [0,1]
\end{aligned} \tag{12}$$

While the optimization above shows n constraints, practically this should be done only for any data which receives non-zero weight from the kernel function $K_h(t_i - t)$. With the Epanechnikov kernel, any data not within $[t-h, t+h]$ receives zero weight (discussed in Section 3.2.1). As a result, the number of constraints is equal to the number of time points included in the estimation of t, determined by the size of the bandwidth. For example, with a degree 2 polynomial expansion of $\theta(t)$ and estimation performed around $t = 10$ with bandwidth $h = 3.5$ yields the following objective function and constraints in Equation 13. With estimation performed at $t = 10$, data within $t \in [6.5, 13.5]$ receives non-zero weight when using the Epanechnikov kernel. Therefore with this example, data at the discrete time points $t \in \{t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}\}$ is shown in the objective function and constraints. The first 7 constraints ensure the expansion of $\theta(t) \in [0,1]$ with the 8^{th} being to assure β_0 is in the interval [0,1] as the estimate for the unknown probability at time t which is being smoothed.

$$
\begin{aligned}
\underset{\beta_0,\beta_1,\beta_2}{\text{maximize}} \quad & \textstyle\sum_{i=7}^{13} \left\{ log[\beta_0 + \beta_1(t_i - t) + \beta_2(t_i - t)^2] + (y_i) log[1 - \beta_0 - \beta_1(t_i - t) - \beta_2(t_i - t)^2] \right\} K_h(t_i - t) \\
\text{subject to} \quad & \beta_0 + \beta_1(t_7 - t) + \beta_2(t_7 - t)^2 \leq 1 \\
& \beta_0 + \beta_1(t_8 - t) + \beta_2(t_8 - t)^2 \leq 1 \\
& \beta_0 + \beta_1(t_9 - t) + \beta_2(t_9 - t)^2 \leq 1 \\
& \beta_0 + \beta_1(t_{10} - t) + \beta_2(t_{10} - t)^2 \leq 1 \\
& \beta_0 + \beta_1(t_{11} - t) + \beta_2(t_{11} - t)^2 \leq 1 \\
& \beta_0 + \beta_1(t_{12} - t) + \beta_2(t_{12} - t)^2 \leq 1 \\
& \beta_0 + \beta_1(t_{13} - t) + \beta_2(t_{13} - t)^2 \leq 1 \\
& \beta_0 \in [0, 1]
\end{aligned} \tag{13}
$$

The constraints represent the polynomial expansion of $\theta(t)$ to degree p. Using the same example, the constraints are shown below, rewritten in matrix formulation. The left-hand matrix is also known as a Vandermonde matrix. While performing cross validation to select the bandwidth, note that the middle constraint (where $t_i = t$ and the row contains 0's in the higher order expansions) is removed.

$$
\begin{bmatrix} 1 & -3 & 9 \\ 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \times \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}
$$

As stated previously, Algorithm 2 can be updated to include the multivariate optimization function. Shown in Algorithm 3, θ can be expanded to its polynomial approximation with the addition of up to p terms. The result of the optimization will be values for each of the coefficients β_q, but again, the one which will be used to estimate θ is β_0.

Algorithm 3 Cross Validation Function for Polynomial Expansion in Geometric Kernel Log Likelihood

```
1: function GEOM.THETA(theta,n,x,tp,k.weights.matrix,k,h)
2:     for j in tp do                                            ▷ Initialization
3:         $logs[j] \leftarrow 0$
4:     for j in tp do
5:         $logs[j] \leftarrow n * \log\{\beta_0 + \beta_1(t_j - k) + ... + \beta_p(t_j - k)^p\}+$
6:         $x * \log\{\beta_0 + \beta_1(t_j - k) + ... + \beta_p(t_j - k)^p\}$          ▷ geometric MLE
7:     return $logs \cdot k.weights.matrix$[:,cv]    ▷ Multiply by column cv of weights matrix
```

A note on the bandwidth should also be addressed here. When considering a polynomial expansion of degree p, bandwidths smaller than p lead to perfect estimation in the boundary points, as not enough time periods lie in the interval. For example, if a polynomial expansion of degree 3 is considered with a bandwidth of 2, there will be more parameters to solve for than there are data within the bandwidth for the first and last time periods. Depending on the size of the bandwidth and the degree of the polynomial, this may be true for several interior points also. Careful consideration should be made to the selection of the bandwidth when increasing the degree of the polynomial.

3.5 Application

The data for the application of one-step procedure in discrete case is the Bangladesh Demographic and Health Survey (BDHS, 2011) data which is a part of the worldwide Demographic and Health Survey Program. In this survey, 17,842 woman were surveyed to collect information about fertility preferences, household size, family planning, maternal and child health problems, infant and child mortality, etc. The variable of interest is the number of months after marriage until first birth, which is treated as time variant cross sectional discrete data. The data contains information for women aged from 13 to 31.

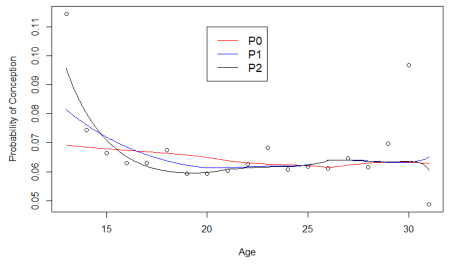
Table 3.1 shows the ratio of mean absolute error by time point for the two-step procedures as compared to the one-step procedures. The two-step procedures (smoothed on the raw estimates) are denoted with parentheses (2). The degree of the polynomial used in the expansion is reflected in the number following "P" in the header of the table (e.g. P1 (2) is the degree 1 expansion of the local polynomial smoothing on the raw estimates of the data). The other two-step procedure, spline smoothing, is denoted by SS. The one-step procedures using the local kernel log-likelihood are denoted by the degree of the polynomial (e.g. P1 is the degree 1 expansion of $\theta(t)$ using kernel log-likelihood approach). All ratios are compared to the degree 0 polynomial expansion of the two-step procedure (P0 (2) has been left out of this table). So the value of 0.54 as the first entry in the table indicates that the error produced by the smoothed value of the degree 1 two-step function (P1 (2)) is 46% less than that produced by the smoothed value of the degree 0 two-step value. Values greater than 1 indicate that more error is produced by the corresponding technique than the degree 0 two-step function. The last row shows the overall mean absolute error across all time points, and is to be interpreted similarly. A clear trend can be detected from this table; higher order degrees of polynomials (regardless of technique) tend to produce less error. The more complex the function (addition of parameters in the expansion of $\theta(t)$), the better the fit. Spline smoothing overall performs the worst across all techniques, but is comparable to the degree 0 expansions using either the one or two-step techniques.

To further show the effect of increasing the degree of the polynomial in the one and two-step procedures, Figure 3.2 shows several plots with smoothed values. To better display the six one-step smoothings, they have been split into two separate figures. Figure 3.2a and 3.2b show the degree 0 to 2 and degree 3 to 5, respectively, expansions using one-step smoothing. It is clear to see that increasing the order of the polynomial expansion has a huge effect on capturing the trends at the boundary points, or tails of the distributions. One detriment to increasing the degree of the polynomial, however, is that the smoothed function is very sensitive to these boundary points. The smoothing function may change drastically when fit on new data, so the tradeoff between increased precision should be understood by the increase in variability of the estimate.

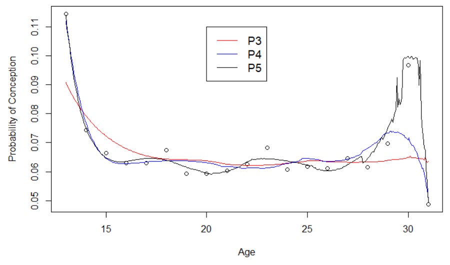
Figure 3.2c compares the one and two-step procedures when using a degree 2 polynomial expansion. Figure 3.2d similarly shows the same comparison for a degree 3 polynomial expansion. Since the degree of the polynomial has such a substantial impact on the resulting smoothing, Table 3.2 shows side by side comparisons of one and two-step procedures of equal degree. This presents a logical way to compare how the one and two-step approaches perform. Similarly to before, the mean absolute ratio is shown, with all ratios compared to the baseline of the two-step method. It can be seen that the one-step procedure produces less error in the degree 0, 1, 2 and 5 cases (error ratios less than 1). For polynomials of degree 3 and 4, the two-step method outperforms the one-step method. Additionally in this table, the best estimator column indicates the number of the 19 time points (ages 13 to 31) which are best smoothed by the corresponding method. For degree 0 polynomial, it can be seen that the two-step method performs best at a majority (11) of the time points, while overall producing slightly more mean absolute error. For other polynomial degrees, the method which produced lesser mean absolute error also performed best at a majority of the time points.

Table 3.1: Mean Absolute Error for Two-Step and One-Step Procedures on Application Dataset

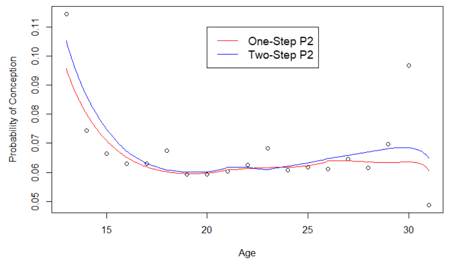
Age	P1 (2)	P2 (2)	P3 (2)	P4 (2)	P5 (2)	SS	P0	P1	P2	P3	P4	P5
13	0.54	0.42	0.23	0.03	0.03	15434.35	1.37	0.73	0.57	1.25	0.11	0.12
14	5.16	1.19	0.59	0.23	0.60	3.92	1.55	0.31	3.22	0.84	0.49	0.21
15	2.71	0.70	0.17	1.66	0.88	4.16	0.17	3.73	0.80	1.31	0.31	0.83
16	1.65	0.39	0.25	0.21	3.87	10.87	0.47	1.27	0.39	2.27	0.04	3.00
17	1.41	0.01	67.34	0.38	1.35	3.56	0.57	0.75	0.33	2.60	0.19	3.40
18	0.92	14.94	1.34	0.38	1.01	0.10	3.03	3.36	1.95	0.42	1.20	0.97
19	1.04	0.11	0.16	16.85	1.29	2.56	0.96	0.46	0.10	16.33	0.90	0.52
20	1.00	0.17	1.00	0.78	2.48	3.37	1.02	0.38	0.24	9.00	0.87	0.05
21	1.00	0.55	1.00	0.37	2.22	2.98	0.98	0.34	0.50	3.83	0.65	0.07
22	1.00	4.68	1.00	0.91	0.07	14.83	0.57	2.25	1.33	0.33	3.25	0.62
23	1.00	1.53	1.00	0.54	1.17	1.14	1.08	1.12	1.03	0.90	1.17	0.57
24	1.00	0.42	1.00	2.09	1.23	0.66	0.75	0.71	0.75	2.44	0.95	1.43
25	1.05	0.53	1.08	0.56	0.79	2.05	0.15	3.00	0.67	4.75	1.42	0.15
26	1.11	0.88	1.12	0.48	1.72	0.76	0.12	9.00	1.00	0.78	1.14	0.38
27	1.95	0.87	2.18	1.50	1.60	0.05	7.19	0.22	1.20	2.17	0.15	10.00
28	1.27	1.01	1.68	0.59	0.19	3.13	0.43	1.43	0.95	0.89	3.82	0.45
29	0.52	0.84	3.21	2.12	0.64	0.63	1.55	1.03	0.97	0.90	0.66	1.46
30	0.92	1.01	0.82	0.22	0.38	15.97	1.10	1.00	0.99	0.96	0.81	0.11
31	1.17	0.75	0.50	0.03	0.17	405.07	0.77	1.16	0.72	1.26	0.12	0.39
Total	1.04	0.77	0.62	0.33	0.30	1.02	1.00	0.90	0.74	0.86	0.52	0.23



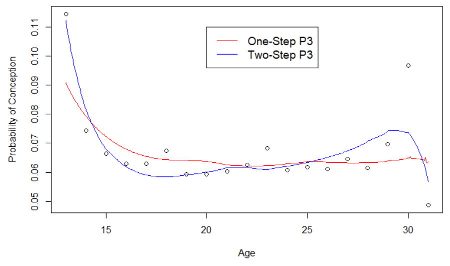
(a) Smoothed Values with Degree 0 to 2 Polynomials



(b) Smoothed Values with Degree 3 to 5 Polynomials



(c) Comparison of One to Two-Step on Degree 2 Fit



(d) Comparison of One to Two-Step on Degree 3 Fit

Figure 3.2: Smoothing Comparisons on Application Dataset

Table 3.2: Mean Absolute Error Ratios between Two-Step and One-Step Procedures

Degree of Polynomial	Method	Mean Error Ratio	Best Estimator
Degree 0	Two-Step	1	11
	One-Step	0.998	8
Degree 1	Two-Step	1	6
	One-Step	0.867	13
Degree 2	Two-Step	1	6
	One-Step	0.964	13
Degree 3	Two-Step	1	11
	One-Step	1.389	8
Degree 4	Two-Step	1	13
	One-Step	1.569	6
Degree 5	Two-Step	1	6
	One-Step	0.782	13

3.6 Simulations

Two simulations are performed to compare the performance of the smoothing techniques. The first is of a time variant variable parameter (TVVP), where each time period's data comes from a different simulated mean. The second simulation exercise is of a time variant fixed parameter (TVFP), where each time period's data comes from a constant simulated mean. Details and results of each simulation follow in the next two sections.

3.6.1 Simulation Under Time Variant Variable Parameter

In this simulation, 40 time periods are generated to have a mean between 0.05 and 0.1 to give some trend to the series (shown in Figure 3.3). Then, with the fixed means, 30 values from a geometric distribution are simulated at each time period. The smoothing techniques are applied to the resulting data and 200 replications of this simulation are done and the minimum absolute deviation (MAD) ratio between the one and two-step procedures for each of the 40 time periods is reported. MAD can be calculated as the mean average error between the raw estimate and the smoothing result. Results are shown in Table 3.3. Results are again split by degree of polynomial. In addition to the MAD ratio, the number of time points where the mean absolute error is lower for each method is shown in the Best Estimator column. For polynomials of degree 0, 1, 2 and 4, the mean MAD is lower for the one-step method. Even for degree 3 and 5 polynomials where the two-step method produces less error, the one-step method produces less absolute error at a majority of the time points.

Similar to Table 3.3, Table 3.4 shows the MSE ratio between the one and two-step procedures. While the MAD ratio between these procedures suggested improvements in the one-step methods, the MSE ratios in Table 3.4 show that the one-step procedures produce larger square error across all time points. However, when looking at the best estimator column, it can be seen that there is almost an even split in the number of time points in which the MSE ratio is smaller.

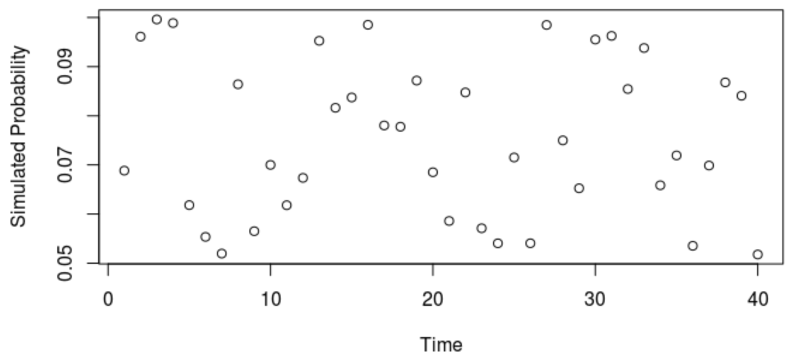


Figure 3.3: Trend for TVVP Simulation

Table 3.3: MAD Ratio in Geometric TVVP Simulation

Degree of Polynomial	Method	MAD Ratio	Best Estimator
Degree 0	Two-Step	1	19
	One-Step	0.975	21
Degree 1	Two-Step	1	18
	One-Step	0.986	22
Degree 2	Two-Step	1	17
	One-Step	0.974	23
Degree 3	Two-Step	1	18
	One-Step	1.048	22
Degree 4	Two-Step	1	17
	One-Step	0.969	23
Degree 5	Two-Step	1	20
	One-Step	1.054	20

Table 3.4: MSE Ratio in Geometric TVVP Simulation

Degree of Polynomial	Method	MSE Ratio	Best Estimator
Degree 0	Two-Step	1	20
	One-Step	1.043	20
Degree 1	Two-Step	1	22
	One-Step	1.058	18
Degree 2	Two-Step	1	18
	One-Step	1.08	22
Degree 3	Two-Step	1	21
	One-Step	1.191	19
Degree 4	Two-Step	1	18
	One-Step	1.047	22
Degree 5	Two-Step	1	21
	One-Step	1.209	19

3.6.2 Simulation Under Time Variant Fixed Parameter

The time variant fixed parameter simulation is similar to the TVVP framework with the exception being that each time period's data comes from the same mean. Otherwise, the setup is the same with the same number of time points (40) and number of data simulated at each time (30). Results are shown in Table 3.5. Under the TVFP simulation framework, the one-step method performs better again for degree 0, 1, 2 and 4 polynomials, as evidenced with MAD ratios less than 1 when compared to the respective two-step method of equal polynomial degree. The best estimator column shows that for these polynomial degrees (0,1,2,4), the majority of the time points had lower MAD ratios with the one-step procedure. The same is true for the two cases where the two-step method produced better results; the best estimator column also agrees with the method of lower MAD ratios.

Table 3.5: MAD Ratio in Geometric TVFP Simulation

Degree of Polynomial	Method	MAD Ratio	Best Estimator
Degree 0	Two-Step	1	13
	One-Step	0.989	27
Degree 1	Two-Step	1	10
	One-Step	0.988	30
Degree 2	Two-Step	1	14
	One-Step	0.993	26
Degree 3	Two-Step	1	32
	One-Step	1.058	8
Degree 4	Two-Step	1	14
	One-Step	0.990	26
Degree 5	Two-Step	1	38
	One-Step	1.093	2

The MSE ratios for the 40 time points are also shown in Table 3.6. Here, it can be seen that the one-step procedures perform worse when measuring the MSE. All two-step methods perform better in terms of lower MSE ratios across all time points, and at a majority of time points in the last column of the table. Results are consistent across both simulations (TVVP and TFVP) that the MAD ratio is more favorable to look at when comparing the one and two-step methods. However, in the TVVP simulation framework, the one-step procedures are still approximately even or superior at a majority of the time points when inspecting both the MAD and MSE ratios.

Table 3.6: MSE Ratio in Geometric TVFP Simulation

Degree of Polynomial	Method	MSE Ratio	Best Estimator
Degree 0	Two-Step	1	31
	One-Step	1.034	9
Degree 1	Two-Step	1	30
	One-Step	1.032	10
Degree 2	Two-Step	1	31
	One-Step	1.052	9
Degree 3	Two-Step	1	39
	One-Step	1.167	1
Degree 4	Two-Step	1	28
	One-Step	1.044	12
Degree 5	Two-Step	1	39
	One-Step	1.202	1

Chapter 4

One-Step Estimation with Continuous Distribution

4.1 Continuous Distribution - Normal Distribution

For the continuous case, we choose the normal distribution which has the following probability density function:

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}; -\infty < x < \infty$$

Similar to the example with the geometric distribution, we will show the derivation for the log-likelihood function. Consider a sample of size n $(x_1, x_2, ..., x_n)$ from the normal distribution with parameters μ and σ. Taking their product then creates the joint distribution and the likelihood function.

$$L = f_n(x|\mu,\sigma^2) = \prod_{i=1}^{n} f(x_i|\mu,\sigma^2)$$

$$L = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

$$L = (2\pi\sigma^2)^{-n/2} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_j-\mu)^2}$$

Again taking the logarithm of the likelihood function simplifies the maximization problem, and with some simplification we reach the result below.

$$\log(L) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2$$

4.2 Kernel Log-Likelihood with Normal Distribution

The one-step solution using kernel log-likelihood would entail choosing the parameter μ to maximize the following equation at each time point:

$$\sum_{j=1}^{tp}[-\frac{1}{2\sigma_j^2}\sum_{x_i \in tp_j}(x_i-\mu_j)^2]K_h(t_j-t) \tag{14}$$

where tp is the number of time points and the kernel function K_h with bandwidth h determines the window of time periods used in estimating each mean μ_j. We use the notation with a double summation to show that the computation takes place for each time period, and for all data points x_i which lie in that time period. The equation above assumes a fixed standard deviation σ_j at each time point, which can be calculated from the original data. Parts of the log-likelihood function which do not depend on μ can be left off in the maximization, as they are constant. This estimation should again occur at each time period, using all data which follows within the bandwidth h which is selected through cross-validation. We also create a method for using the kernel log likelihood where σ is not assumed to be known and fixed. In this method, we begin with the same initial standard deviation, and simultaneously optimize both equations 14 and 15. Equation 15 solves for σ at each time point based on a previously found μ from Equation 14. This method is detailed in Section 4.4.

$$\sum_{j=1}^{tp}[-\frac{n}{2}\log(\sigma_j^2)-\frac{1}{2\sigma_j^2}\sum_{x_i \in tp_j}(x_i-\mu_j)^2]K_h(t_j-t) \tag{15}$$

4.3 Algorithm - Fixed Sigmas

With the normal distribution, we have two parameters: μ and σ. For one-step estimation, we can at first assume a fixed σ at each time period and use these to estimate the mean μ based on MLE for normal distribution and the kernel function described in Equation 14. The procedure for finding the optimal bandwidth is similar to Algorithm 1 for the geometric distribution. We search a list of bandwidths and compute the corresponding kernel weights matrix at each bandwidth. The $Norm.mu.cv$ function shown in Algorithm 5 computes the individual contributions of each time point to the kernel based MLE function in Equation 14. At each time point, we perform leave-one-out cross validation to use neighboring time periods, as determined by the current bandwidth and k.weights.matrix, to estimate the mean of the left out time period. We then compare the estimated mean to the raw estimate. The total error is retained across all time periods for each bandwidth, with the bandwidth corresponding to the minimum error being chosen in the end. The procedure is detailed in Algorithm 4.

Algorithm 4 Bandwidth Selection for Normal KLL with Fixed Sigma

```
1: procedure BANDWIDTH SELECTION - FIXED SIGMA
2:     $raw$ are the raw estimates from the original data
3:     $y$ is original data, split by time points
4:     $tp$ is a list of time points
5:     $sigmas$ is a list of standard deviations for each time period

6:     $bw \leftarrow seq(1, 10, by = 0.05)$          ▷ Assign list of bandwidths to be checked
7:     $opt.h \leftarrow 0$          ▷ Initialize list to track error from LOOCV

8:     for h in bw do
9:         $tot \leftarrow 0$          ▷ Initialize error to 0 each iteration
10:        Compute $k.weights.matrix$ based on current h
11:        for cv in tp do
12:            $mu \leftarrow$ optimize Norm.mu.cv function
                   ▷ Norm.mu.cv(mu,y,tp,sigmas,k.weights.matrix,cv,h)
                   ▷ cv is the current time period from inner for-loop
                   ▷ h is current bw from outer for-loop
                   ▷ Norm.mu.cv function returns mu based on MLE
13:            $tot \leftarrow tot + (raw[cv] - mu)^2$          ▷ Compute MSE from estimated mu
14:        Append $tot$ to $opt.h$          ▷ Error from all time points for current bw
15:    return $h$ with minimum error
```

Algorithm 5 Cross Validation Function for Normal Kernel Log Likelihood with Fixed Sigmas

```
1: function NORM.MU.CV(mu,y,tp,sigmas,k.weights.matrix,cv,h)
2:     for j in tp do          ▷ Initialization
3:         $b[j] \leftarrow 0$
4:     for j in tp between cv-h and cv+h do          ▷ Any time period within bandwidth
5:         if $j == cv$ then
6:             next          ▷ Leave time period cv out
7:         $b[j] \leftarrow -\frac{1}{2sigmas[j]^2} \sum_{y_i \in t_j} (y_i - mu)^2$          ▷ From normal MLE
8:     return $b \cdot k.weights.matrix[:,cv]$          ▷ Multiply by column $cv$ of the weights matrix
```

4.4 Algorithm - Variable Sigmas

The previous algorithm assumed that σ was known for each time period when solving for μ. In this section we discuss an approach to simultaneously estimate $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Let $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ represent a vector of means and standard deviations, for all the time periods. The procedure begins in the same manner as the previous one with fixed sigmas. As input, we require an initial list of standard deviations and means for each time period. We begin by first estimating $\boldsymbol{\mu}$ based on the initial $\boldsymbol{\sigma}$ (which were previously assumed to be fixed). In this part, we assume the standard deviations are again fixed. Then, using the newly computed $\boldsymbol{\mu}$ (we can call it $\boldsymbol{\mu_{next}}$), we assume these to be fixed and solve Equation 15 to obtain a new $\boldsymbol{\sigma}$ (or $\boldsymbol{\sigma_{next}}$). In this iterative process of solving for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, we continue this for either a maximum number of iterations or until convergence has been met between successive $\boldsymbol{\mu}$ solutions, as defined by some tolerance level. After $\boldsymbol{\mu}$ has converged, we use the most recent solution $\boldsymbol{\sigma}$ to solve for each μ in the LOOCV procedure. The functions for simultaneous estimation of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are shown in Algorithms 6 and 7, and the entire procedure is detailed in Algorithm 8.

Algorithm 6 Normal KLL Function for Finding Optimal Mu

```
1: function NORM.MU(mu,y,tp,sigmas,k.weights.matrix,k,h)
2:     for j in tp do                                          ▷ Initialization
3:         $b[j] \leftarrow 0$
4:     for j in tp between k-h and k+h do         ▷ Any time period within bandwidth
5:         $b[j] \leftarrow -\frac{1}{2sigmas[j]^2} \sum_{y_i \in t_j} (y_i - mu)^2$           ▷ From normal MLE
6:     return $b \cdot k.weights.matrix$[:,k]          ▷ Multiply by the weights matrix
```

Algorithm 7 Normal KLL Function for Finding Optimal Sigma

```
1: function NORM.SIGMA(sigma,y,tp,mus,k.weights.matrix,k,h)
2:     for j in tp do                                          ▷ Initialization
3:         $b[j] \leftarrow 0$
4:     for j in tp between k-h and k+h do         ▷ Any time period within bandwidth
5:         $n \leftarrow length(y_j)$                 ▷ Number of observations at each time
6:         $b[j] \leftarrow -\frac{n}{2}\log(sigma^2) - \frac{1}{2sigma^2} \sum_{y_i \in t_j} (y_i - mus[j])^2$         ▷ From MLE
7:     return $b \cdot k.weights.matrix$[:,k]          ▷ Multiply by the weights matrix
```

Algorithm 8 Bandwidth Selection for Normal KLL with Variable Sigma

```
procedure BANDWIDTH SELECTION - VARIABLE SIGMA
    $raw$ are the raw estimates from the original data
    $y$ is original data, split by time points
    $tp$ is a list of time points
    $sigmas$ is the initial list of standard deviations for each time period
    $mus$ is the initial list of means for each time period

    $bw \leftarrow seq(1, 10, by = 0.05)$        ▷ Assign list of bandwidths to be checked
    $opt.h \leftarrow 0$        ▷ Initialize list to track error from LOOCV
    $max.iter \leftarrow 25$        ▷ Maximum number of iterations if convergence not met
    $tolerance \leftarrow 1$        ▷ Tolerance level for convergence

    for h in bw do
        $tot \leftarrow 0$        ▷ Initialize error to 0 each iteration
        $mus.prev \leftarrow mus$        ▷ Keep track of previous list of mus
        $iter \leftarrow 1$        ▷ Count number of iterations
        Compute $k.weights.matrix$ based on current h

        while iter $\leq$ max.iter do
            for k in tp do
                $mu \leftarrow$ optimize Norm.mu function
                        ▷ Norm.mu(mu,y,tp,sigmas,k.weights.matrix,k,h)
                        ▷ Norm.mu function returns mu based on MLE
                        ▷ Uses the most recently solved $sigmas$ list ($\boldsymbol{\sigma}$)
                $mus[k] \leftarrow mu$        ▷ Update mus with new mu
            for k in tp do
                $s \leftarrow$ optimize Norm.sigma function
                        ▷ Norm.sigma(sigma,y,tp,mus,k.weights.matrix,k,h)
                        ▷ Norm.sigma function returns sigma based on MLE
                        ▷ Uses the most recently solved $mus$ list ($\boldsymbol{\mu}$)
                $sigmas[k] \leftarrow s$        ▷ Update sigma with new s
            if Norm(mus-mus.prev) $\leq$ tolerance then
                break while loop        ▷ If convergence is met, break loop
            $iter \leftarrow iter + 1$
        for cv in tp do
            $mu \leftarrow$ optimize Norm.mu.cv function
                        ▷ Norm.mu.cv(mu,y,tp,sigmas,k.weights.matrix,cv,h)
                        ▷ Norm.mu.cv function returns mu based on MLE
            $tot \leftarrow tot + (raw[cv] - mu)^2$        ▷ Compute MSE from estimated mu
        Append $tot$ to $opt.h$        ▷ Error from all time points for current bw
    return $h$ with minimum error
```

4.5 Polynomial Expansion in Fixed and Variable Sigmas Algorithms

Similar to Section 3.3, the previous algorithms can be adjusted to solve for higher degree polynomial expansions. In Chapter 3, there was simply one parameter θ for the probability in a geometric distribution which was expanded to higher degrees. In this chapter, this can be done for both parameters in the normal probability model, in both instances of considering σ to be known or allowing it to be variable. Local kernel log-likelihood estimation of the time-variant parameter $\mu(t)$ (when $\sigma(t)$ is known/fixed) requires to maximize

$$-\frac{1}{2\sigma^2(t)}\sum_{i=1}^{n}\left\{y_i-\sum_{q=0}^{p}\beta_q(t)(t_i-t)^q\right\}^2 K_h(t_i-t), \tag{16}$$

where $\beta_q(t)=\frac{\mu^{(q)}(t)}{q!}; 0\le q\le p$. Similarly, when $\mu(t)$ is known, estimation of the standard deviation $(\sigma(t))$ can be obtained by maximizing

$$\sum_{i=1}^{n}-\frac{1}{2}\log(\sum_{q=0}^{p}\gamma_q(t)(t_i-t)^q)+\sum_{i=1}^{n}-\frac{1}{2\left\{\sum_{q=0}^{p}\gamma_q(t)(t_i-t)^q\right\}^2}\left\{y_i-\mu(t)\right\}^2 K_h(t_i-t) \tag{17}$$

where $\gamma_q(t)=\frac{\sigma^{(q)}(t)}{q!}; 0\le q\le p$. When both parameters $\Theta(t)=(\mu(t),\sigma(t))$ are unknown, we can simultaneously estimate equations (16) and (17) for the one-step smoothing of both parameters by the proposed local kernel log-likelihood method.

4.5.1 Closed Form Solution

The problem of local least squares regression can be expressed in matrix notation. First recall that in least squares regression, the sum of squares expression which is minimized can be expressed as

$$(y-X\beta)^T(y-X\beta),$$

where y is an $n\times 1$ response vector, X is an $n\times p$ design matrix and β is $p\times 1$ coefficients matrix. Further expanding this results in

$$y^Ty-y^TX\beta-\beta^TX^Ty+\beta^TX^TX\beta.$$

Using the fact that $(\beta^TX^Ty)^T=y^TX\beta$, the above reduces to

$$\begin{aligned}&y^T y - y^T X\beta - \beta^T X^T y + \beta^T X^T X\beta \\ &= y^T y - 2y^T X\beta + \beta^T X^T X\beta.\end{aligned}$$

To minimize this, taking the derivative and setting the result to 0 gives a solution for $\hat{\beta}$

$$\begin{gathered}\tfrac{d}{d\beta}[y^T y - 2y^T X\beta + \beta^T X^T X\beta] \\ = -2y^T X + 2X^T X\beta = 0 \\ X^T X\beta = X^T y \\ \hat{\beta} = (X^T X)^{-1} X^T y,\end{gathered}$$

provided that $X^T X$ is invertible. In cases where the matrix $X^T X$ is not invertible, a generalized inverse can be used instead. In the weighted case there are very similar results. Beginning with the weighted least squares function

$$\sum\nolimits_{i=1}^{n} w_i (y_i - x_i\beta)^2$$

In matrix form, this is equivalent to:

$$(y - X\beta)^T W (y - X\beta)$$

where W is the diagonal weights matrix (of size $n \times n$ with entries w_i on the diagonal). Similarly expanding this gives

$$y^T W y - y^T w X\beta - \beta^T X^T w y + \beta^T X^T W X\beta.$$

Lastly, differentiating yields a solution for $\hat{\beta}$

$$\begin{gathered}\tfrac{d}{d\beta}[y^T W y - y^T w X\beta - \beta^T X^T w y + \beta^T X^T W X\beta] \\ = -2X^T W y + 2X^T W X\beta = 0\end{gathered}$$

$$\hat{\beta} = (X^T W X)^{-1} X^T W y.$$

This can be used to solve for $\mu(t)$ in both the fixed and variable sigmas methods. In the scenario with fixed sigmas, the additional coefficient resulting from the standard deviation can be included in the weight matrix W as a constant scale factor. In the procedure with variable sigmas, the iteration when solving for $\mu(t)$ can be solved in this fashion, before alternating to the step where $\sigma(t)$ is expanded by a polynomial approximation, which can be solved with similar optimization techniques as is the case for a polynomial expansion of $\theta(t)$ in Section 3.4.

4.6 Application Data

The application dataset comes from Londonair which is the website of the London Air Quality Network (LAQN). The selected data is a measurement of the oxides of nitrogen in the air of Camden, a borough of London. Oxides of nitrogen refer to various gases that are composed of nitrogen and oxygen, including NO and NO2 (measured in micrograms/cubic meter). The collected data measures the levels of oxides of nitrogen in 15 minute intervals from January 1, 2017 to Dec 31, 2017. We take samples of roughly 30 readings per week (measurements made approximately every 5 hours) to gather data which meets requirements of a normal distribution. Each time period undergoes and passes the Shapiro-Wilk test of normality at 0.01 level of significance.

Results of the one and two-step procedures are shown in Table 4.1. Table 4.1 compares methods of the same degree for the polynomial expansion of μ, stratified further by the expansion of σ. The first column indicates the degree of polynomial used in the expansion of μ, varying from 0 to 3. The second column then indicates the degree of expansion of σ. For example, the S0 row indicates a polynomial expansion of degree 0 for σ in the kernel log-likelihood for normal distribution. S1 then indicates a degree 1 expansion, and so on. The row with "One-Step" (no specification of S) represents the one-step method with fixed sigmas. The mean error column is shown as a ratio of mean error for the one-step procedures to the two-step procedure, stratified by the degree of the polynomial. The best estimator column again shows the number of time points in which each method attains the closest predictions. For degree 0 expansion of μ, we see that the two-step method performs the best, with all the mean error ratios greater than 1. However, for all methods with a degree 0 expansion of μ, the one-step method (with fixed sigmas) yields the most points with closest prediction with 19. For the degree 1 expansion of μ, the one-step method with degree 0 polynomial expansion of σ yields the lowest mean error ratio. All one-step procedures with variable sigmas (again, denoted with S0, S1, S2 or S3) yield improvements in the mean error. Similar results are seen with the degree 2 expansion of μ methods, with all one-step procedures with variable sigmas resulting in lesser mean error. The degree 3 expansion of μ suggests the same pattern, with increasing degree of σ expansion yielding small improvements in error reduction.

Table 4.1: Mean Absolute Error Ratios on Application Data

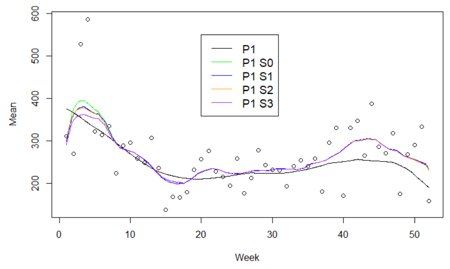
Degree of Polynomial (μ)	Method (Degree of σ)	Mean Error	Best Estimator
Degree 0	Two-Step	1	13
	One-Step	1.027	19
	One-Step S0	1.001	3
	One-Step S1	1.007	1
	One-Step S2	1.014	5
	One-Step S3	1.015	11
Degree 1	Two-Step	1	12
	One-Step	1.145	13
	One-Step S0	0.965	11
	One-Step S1	0.972	3
	One-Step S2	0.977	9
	One-Step S3	0.987	4
Degree 2	Two-Step	1	11
	One-Step	1.055	16
	One-Step S0	0.939	7
	One-Step S1	0.936	4
	One-Step S2	0.934	6
	One-Step S3	0.937	8
Degree 3	Two-Step	1	7
	One-Step	1.036	20
	One-Step S0	0.987	4
	One-Step S1	0.979	2
	One-Step S2	0.967	8
	One-Step S3	0.964	11

What is evident from Table 4.1 is that increasing the degree of the polynomial expansion of σ yields marginal improvements, at best, in the smoothing results. To better compare the two-step to the one-step procedure, Table 4.2 shows side by side comparisons of which method produces more accurate predictions at the majority of the 52 time points in this application data. Here we highlight the two-step method and the one-step method with a degree 0 expansion of σ, again since higher order expansion of σ doesn't necessarily yield significant improvement. For degree 0 expansion of μ, each method is the best estimator at half the points. For degree 1 expansion of μ, the one-step method is the best estimator at 31 of the 52 time points. At its best, the degree 2 expansion of μ yields better estimates at roughly two-thirds of the 52 time points for the one-step method over the two-step method.

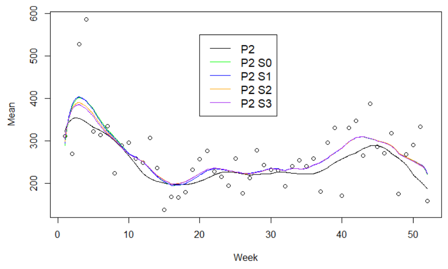
Table 4.2: Comparison of One to Two-Step Methods on Application Data

Degree of Polynomial (μ)	Method (Degree 0 for σ)	Best Estimator
Degree 0	Two-Step	26
	One-Step S0	26
Degree 1	Two-Step	21
	One-Step S0	31
Degree 2	Two-Step	17
	One-Step S0	35
Degree 3	Two-Step	20
	One-Step S0	32

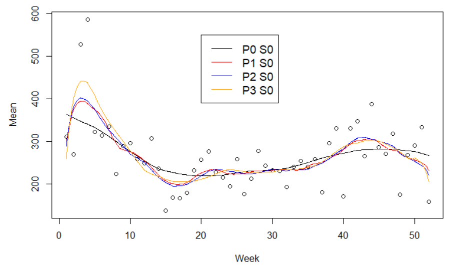
Figure 4.1 visualizes what is seen in Table 4.1. Figure 4.1a depicts degree 1 polynomial expansions of μ with varying expansions of σ. Besides the one-step method using fixed sigmas (denoted by P1 in the figure), the remaining methods produce very similar smoothing results. The labels for other methods remain consistent from the tables above; P1 S0 representing a polynomial of degree 1 for μ with degree 0 polynomial for σ, P1 S1 representing the same degree for the mean with a degree 1 polynomial for the standard deviation, and so on. For polynomial expansion of degree 2 of μ in Figure 4.1b, a similar phenomenon is observed. Between Figure 4.1a and 4.1b, it again suggests that higher order polynomials for μ perform better at the boundary points. This is further confirmed in Figure 4.1c. Figure 4.1c shows the results of varying the polynomial expansion of μ (from 0 to 3) with constant degree 0 for σ. Many of the interior points have similar smoothing values, with clear benefit at the boundary points with increased polynomial degree for μ. Figure 4.1d depicts the same varying degrees of μ with constant degree 3 for σ. Smoothing results are almost identical between Figure 4.1c and 4.1d, further confirming that a degree 0 expansion of σ yields the largest incremental benefit in smoothing (from the fixed sigmas approach).



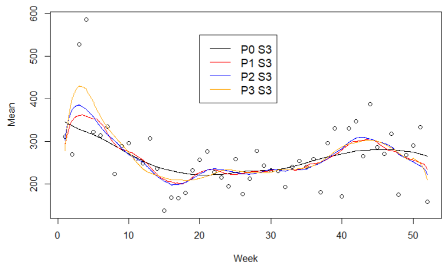
(a) Comparison of Degree 1 Polynomials



(b) Comparison of Degree 2 Polynomials



(c) Comparison of Increasing Degree of Polynomial



(d) Comparison of Increasing Degree of Polynomial with Higher Degree Sigma Expansion

Figure 4.1: Smoothing Comparisons on Application Dataset

4.7 Simulations

4.7.1 Simulation Under Time Variant Variable Parameter

Again two simulation exercises are performed to now test the different smoothing techniques. In the time variant variable parameter framework, we simulate a trend consisting of 50 time points from a normal distribution with mean 250 and standard deviation 20. This trend is shown in Figure 4.2. With these fixed means, 50 replications of simulating 30 observations at each point (with its respective mean) are performed and the mean MAD and MSE is measured at each time point after smoothing the series with each technique. Table 4.3 shows the results of the ratios of mean absolute deviation in comparing the one-step to the two-step methods. Results are generally mixed, with no consistent trend between methods for degree 0 and 1 polynomial expansions of μ. For the degree 2 polynomial expansion of μ, the one-step methods consistently produce less mean absolute error than the two-step method. Again, increasing degree of σ shows minimal gain in reducing the mean error in this table. When comparing the two-step method to the one-step method with degree 0 expansion of σ in Table 4.4, similar to before, we see that the one-step procedure generally performs just as well as the two-step procedure in a majority of the smoothed time points in the simulation exercise, despite having slightly higher absolute error in some instances in Table 4.3.

Similar tables are created for the MSE ratios between two-step and one-step methods (Tables A.1 and A.2). While the one-step procedure performed comparably when comparing the mean absolute error, the ratio of MSE indicates that the two-step method is slightly better with the exception of the case with degree 2 polynomial expansion of μ. However, when comparing the raw number of points in which each method is the ”best estimator” among the 50 time points in simulation, the one-step method performs just as well as the two-step method for degree 0, 1 and 2 degree expansions of μ (Table A.2).

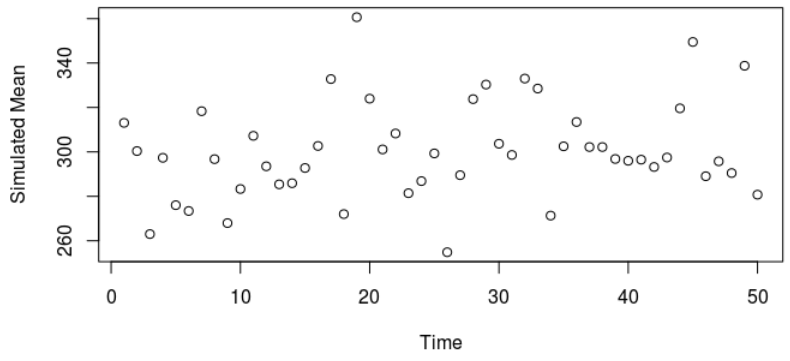


Figure 4.2: Trend for TVVP Simulation

Table 4.3: MAD Ratio in TVVP Simulation

Degree of Polynomial (μ)	Method (Degree of σ)	MAD Ratio	Best Estimator
Degree 0	Two-Step	1	13
	One-Step	1.007	5
	One-Step S0	1.002	5
	One-Step S1	1.001	4
	One-Step S2	0.992	16
	One-Step S3	1.021	7
Degree 1	Two-Step	1	14
	One-Step	1.001	3
	One-Step S0	0.987	6
	One-Step S1	1.007	6
	One-Step S2	0.994	8
	One-Step S3	1.022	13
Degree 2	Two-Step	1	14
	One-Step	1.001	3
	One-Step S0	0.986	1
	One-Step S1	0.977	9
	One-Step S2	0.973	12
	One-Step S3	0.984	11
Degree 3	Two-Step	1	21
	One-Step	1.009	4
	One-Step S0	1.008	5
	One-Step S1	1.008	1
	One-Step S2	0.998	6
	One-Step S3	0.996	13

Table 4.4: Comparison of Methods - TVVP MAD Simulation

Degree of Polynomial (μ)	Method (Degree 0 of σ)	Best Estimator
Degree 0	Two-Step	25
	One-Step S0	25
Degree 1	Two-Step	23
	One-Step S0	27
Degree 2	Two-Step	25
	One-Step S0	25
Degree 3	Two-Step	29
	One-Step S0	21

4.7.2 Simulation Under Time Variant Fixed Parameter

The second simulation is of a time variant fixed parameter. Data is simulated from a constant mean with standard deviation 5 to generate these results. Unlike results from the previous section where the one-step methods generally showed improvements in smoothing accuracy, the results in Table 4.5 show that the two-step method performs much better in this simulation exercise. In all instances, the two-step method (of same polynomial degree for μ) is superior to any of the one-step methods. At best, the one-step method of same degree performs approximately 8% worse in MAD ratios. Results when looking at the MSE ratios in Table A.3 are consistent to those of before, that the one-step methods perform better when comparing the MAD rather than MSE.

Table 4.5: MAD Ratio in TVFP Simulation

Degree of Polynomial (μ)	Method (Degree of σ)	MAD Ratio	Best Estimator
Degree 0	Two-Step	1	46
	One-Step	1.082	2
	One-Step S0	1.084	0
	One-Step S1	1.084	0
	One-Step S2	1.085	1
	One-Step S3	1.086	1
Degree 1	Two-Step	1	44
	One-Step	1.098	1
	One-Step S0	1.098	1
	One-Step S1	1.098	0
	One-Step S2	1.098	0
	One-Step S3	1.099	3
Degree 2	Two-Step	1	49
	One-Step	1.206	1
	One-Step S0	1.202	0
	One-Step S1	1.201	0
	One-Step S2	1.200	0
	One-Step S3	1.201	0
Degree 3	Two-Step	1	49
	One-Step	1.216	1
	One-Step S0	1.213	0
	One-Step S1	1.213	0
	One-Step S2	1.212	0
	One-Step S3	1.212	0

Chapter 5

Nonparametric Volatility Estimation

5.1 Introduction

In economics, it is often of interest to model the variability, or volatility, of asset returns. While traditional time series models assume a constant variance, in economic and financial time series, one usually looks at stock returns which go through periods of high volatility followed by periods of steady returns. The autoregressive conditional heteroscedastic (ARCH) model, first introduced by Engle in 1982, allowed for the conditional variance to vary over time as a function of its lagged errors. Since then, there have been many evolutions of the ARCH model and nonparametric counterparts. Specifically, in this chapter we focus on two local regression techniques for volatility estimation: local least squares and local composite quantile regression. We will compare our nonparametric method to the GARCH method (discussed below), so we first discuss the evolution of autoregressive methods.

In 1982, Engle proposed an autoregressive model for heteroscedastic data, allowing the conditional variance to vary over time as a function of its lagged errors. Let r_t be the log return of an asset at time t. Since the goal is to model the conditional variance of this series, define $a_t = r_t - \mu_t$ be the residuals of the series, after subtracting the mean of the series μ_t. Usually, μ_t can simply be as the sample mean of the return series, but some other model (autoregressive or moving average combination) can be used to remove the mean of the series. This series a_t is commonly referred to as the shock or innovation of an asset. The squared series of a_t^2 can then be used to test for conditional heteroscedasticity. The ARCH model attempts to model the dependence of a_t as a function of its squared lag values. The ARCH model is defined as

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i a_{t-i}^2,$$

where $a_t = \sigma_t \epsilon_t$, also referred to as shocks of the return series, with ϵ_t being an iid random variable with mean 0 and standard deviation 1 and σ_t is the conditional standard deviation of the series at time t (so σ_t^2 is the volatility), and constants α_0 and α_i non-negative. Early use of these ARCH models gave rise to a generalized (GARCH) autoregressive conditional heteroscedastic model. Situations arose where many lagged periods would be required in the ARCH model, so Bollerslev (1986) introduced the GARCH model as both a function of lagged errors (like ARCH) and additionally of an autoregressive term on the conditional variance, defined below:

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i a_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2,$$

where again there are non-negativity constraints on the coefficients of the model. In practice, a GARCH(1,1) model is usually sufficient for modeling the conditional variance (Hansen and Lunde 2001, Teräsvirta 2006, Füss 2007). Further building on the GARCH model, in 1991 Nelson proposed an exponential GARCH model, defined as

$$ln(\sigma_t^2) = \alpha_0 + \sum_{i=1}^{q} \alpha_i \frac{|a_{t-i}| + \delta_i a_{t-i}}{\sigma_{t-i}} + \sum_{j=1}^{p} \beta_j ln(\sigma_{t-j}^2).$$

The exponential GARCH (EGARCH) model presents several advantages. Previously with ARCH and GARCH, there were non-negativity constraints to ensure that the conditional variance was positive. With the EGARCH model, this is no longer necessary with the logged conditional variance being modeled, so positive shocks a_{t-i} can be treated differently than negative shocks (a positive shock would have coefficient $1 + \delta_i$ while a negative shock would have coefficient $1 - \delta_i$). If δ_i is negative, then negative shocks have a larger impact on estimated volatility than positive shocks, which is expected to occur and desirable to test (Tsay 2010). As an alternative and extension of EGARCH, the GJR (named for the authors in Glosten, Jagannathan and Runkle in 1993) model expresses the variance as

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i a_{t-i}^2 + \sum_{i=1}^{q} \delta S_{t-i}^- a_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2,$$

where S_{t-i}^- is an indicator variable for a negative asset shock. Bad news of a negative shock in this model will have an impact of $(\alpha + \delta)$, which is greater than the impact of a positive shock (simply an effect of α). When $\delta > 0$, a leverage effect for negative shocks exists (Dutta 2014).

Many other variations of GARCH exist, such as IGARCH, GARCH-M and TARCH, to name a few (see Teräsvirta 2006). Dufays (2015) describes a class of Markov switching (MS-GARCH) models as alternatives to GARCH models since they may struggle to detect sharp increases in volatility. These MS-GARCH models are of the form $\sigma_{t-1}^2 = \alpha_0 + \alpha_{s_t} a_{t-1}^2 + \beta_{s_t} \sigma_{t-1}^2(s_{1:t-1})$ where the lagged conditional variance term depends on an additional state variable s_t. This additional variable indicates which parameters of the GARCH process are active at time t, allowing for a more robust process to capture irregularites. Dufays develops a method of Bayesian inference for MS-GARCH models with path dependence.

Nonparametric and semiparametric integrations with the GARCH model also exist. Wang et. al. (2012) introduced a semiparametric GARCH approach with an additive autoregressive structure (coined GARCH-ADD). The different component functions in the additive model are linked by a parametric coefficient which can be estimated with polynomial splines. While a GARCH(1,1) model only requires solving for two parameters, Giordano and Parrella (2016) point out some computational issues and propose a nonparametric N ARCH(1) model which does not need the secondary covariate of a lagged variance term. Giordano rewrites the variance function as a nonlinear representation of the variance function from a GARCH(1,1) process which only depends on one lagged shock term a_{t-1}, and can be solved for with local polynomials. To estimate this new representation, an updated bandwidth selection from Giordano and Parrella (2014) is implemented for heteroscedastic and autoregressive models.

Nonparametric regression techniques have also been developed to estimate the conditional variance function. Fan and Yao (1998) propose a residual based estimator to estimate the conditional variance $\sigma^2(x)$ which requires two rounds of local least squares estimation. The first round of least squares regression models the mean and the second then takes the squared residuals from the first round. Yu and Jones (2004) consider an expansion for $\log[\sigma^2(x)]$ instead of $\sigma^2(x)$, to ensure positivity of the variance estimates. Chen (2009) propose a slightly different estimator for the variance for the case of distributions with heavy tail, which is commonly seen in financial data. Xu and Phillips (2011) proposed bypassing the second round of local least squares from Fan (1998) by simply considering a reweighted local constant estimator of the squared residuals. Jin et. al. (2014) present a method of jointly estimating the mean and variance with a multi-parameter local likelihood model. Their method is able to adapt for an asymmetric error distribution. Local composite quantile regression was introduced in 2010 by Kai and Li as an alternative to local least squares regression. These local regression techniques require the additional selection of a bandwidth.

With their rise in popularity, other machine learning methods exist for modeling volatility. Luong and Dokuchaev (2018) use a combination of classification and regression trees to estimate volatility. They do so in two steps. First, they model the direction of change for the period (i.e. up or down) using lagged predictors. Then, using this as an additional predictor, they perform a regression tree to compute estimates for realized volatility. Feedforward artificial neural networks can also be used for volatility estimation (see Johnsson 2018 and Arnerić et. al. 2018).

Our plan is to combine the ideas of Fan and Yao (1998) with those of Kai and Li (2010) to create a residual based local composite quantile regression method for estimation of volatility. More details on these will follow in Section 5.2 and 5.3. Then we discuss the method for choosing the bandwidth for the nonparametric methods, after which we compare the methods in training and forecasting on daily stock closing prices for 990 firms.

5.2 Residual Based Estimator

Fan and Yao (1998) proposed a residual based estimator to model the conditional variance. In this paper, they compare this residual based estimator to a direct estimator and the "benchmark" estimator. Details of these three estimators are given below.

Let $\{(Y_i, X_i)\}$ be a stationary process where $m(x) = E(Y|X = x)$ and $\sigma^2(x) = var(Y|X = x)$. The regression model can be written as $Y_i = m(X_i) + \sigma(X_i)\epsilon_i$. The direct estimator arises from the decomposition of the variance as $\sigma^2(x) = E(Y^2|X = x) - m^2(x)$. The direct estimator $\hat{\sigma}^2_d(x)$ can be found as:

$$\hat{\sigma}^2_d(x) = \hat{v}(x) - \hat{m}^2(x)$$

where $\hat{m}(x)$ and $\hat{v}(x)$ are estimators for $m(x)$ and $v(x) \equiv E(Y^2|X = x)$, respectively. Some of the issues with this estimator are that $\hat{\sigma}^2_d(x)$ is not necessarily guaranteed to be nonnegative, depending of choice in smoothing parameters when calculating $m(x)$ and $v(x)$, and the direct method can lead to high bias.

The benchmark estimator is defined by Fan and Yao to be the estimator resulting from a regression function where $m(x)$ is given. Here, $\sigma^2(x)$ can be estimated as a nonparametric regression function as such:

$$(\hat{\alpha}, \hat{\beta}) = arg \min_{\alpha,\beta} \sum_{i=1}^{n} \{r_i - \alpha - \beta(X_i - x)\}^2 K(\frac{X_i - x}{h_1}) \tag{18}$$

where $r_i = \{Y_i - m(X_i)\}^2$. The benchmark local linear estimator of $\sigma^2(x)$ is $\hat{\sigma}^2_b(x) = \hat{\alpha}$. When $m(x)$ is unknown, which is the likely scenario, the residuals are computed after a nonparametric regression estimator is calculated for $m(x)$. The first step results in $\hat{m}(x) = \hat{a}$, which is the local linear estimator from the following least squares problem:

$$(\hat{a}, \hat{b}) = arg \min_{a,b} \sum_{i=1}^{n} \{Y_i - a - b(X_i - x)\}^2 K(\frac{X_i - x}{h_1}) \tag{19}$$

where $K(.)$ is a nonnegative kernel function and $h_1 > 0$ is a bandwidth. The squared residuals from the above equation are denoted as $\hat{r}_i = \{Y_i - \hat{m}(X_i)\}^2$. With these residuals, the estimate for variance can be computed with the same least squares problem in Equation 18, using the squared residual $\hat{r}_i$ from Equation 19:

$$(\hat{\alpha}, \hat{\beta}) = arg \min_{\alpha,\beta} \sum_{i=1}^{n} \{\hat{r}_i - \alpha - \beta(X_i - x)\}^2 K(\frac{X_i - x}{h_2}) \tag{20}$$

where h_2 is another bandwidth. The residual based estimator for variance is $\hat{\sigma}^2 = \hat{\alpha}$. Fan and Yao showed that this residual based estimator is more efficient than the direct estimator (with efficiency gains of up to 70%), and that asymptotically it performs as well as the benchmark estimator, so it is an efficient estimator for the case of an unknown regression function $m(x)$.

5.3 Local Composite Quantile Regression

Local polynomial regression has been extensively covered by Fan and Gijbels since 1996. While the least squares method is popular and relatively easy to implement, other local methods may perform better, such as in the presence of outliers. In 2008, Zou and Yuan first introduced composite quantile regression as an alternative to least squares regression. Kai and Li (2010) then introduced local composite quantile regression (CQR) as a nonparametric alternative for estimating a regression function. Let

$$\rho_{\tau_k} = \begin{cases} \tau_k r & r \geq 0 \\ r(1-\tau_k) & r < 0 \end{cases}$$

for $k = 1, 2, \ldots, q$ be q check loss functions for q equally spaced quantile positions $\tau_k = \frac{k}{q+1}$. CQR fits q quantile regressions with the same slope. The locally weighted CQR model requires minimizing:

$$(\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_q, \hat{b}) = arg\min_{a,b} \sum_{k=1}^{q} \sum_{i=1}^{n} \rho_{\tau_k}\{y_i - a_k - b(x_i - x)\} K(\frac{x_i - x}{h}) \tag{21}$$

The average of $\hat{a}_q$ are then taken to create the nonparametric estimate of $\hat{m}(x)$, and, if interested in the derivative of our nonparametric function, use $\hat{b}$ from Equation 21.

$$\begin{aligned} \hat{m}(x) &= \frac{1}{q} \sum_{k=1}^{q} \hat{a}_k \\ \hat{m}'(x) &= \hat{b} \end{aligned} \tag{22}$$

In their paper, Kai and Li compare local CQR to local least squares (for estimating the mean m(x) rather than the variance) leads to efficiency gains in relative MSE ratios. In their simulations, they find that for a normally distributed error distribution, local least squares is best. Under several non-normally distributed error distributions, the local CQR method outperforms local least squares. Our plan is to combine the residual based estimator of Section 5.2 with the local CQR method of this section to create a new estimator for the conditional variance function (residual based local CQR method).

5.3.1 Asymptotic Properties

Kai and Li also develop some asymptotic results to demonstrate that local CQR is an efficient alternative to local least squares regression. First, for local least squares regression (denoted LLS):

$$bias\{\hat{m}(x)_{LLS}|X\} = \frac{1}{2}m''(x)\mu_2 h^2 + o_p(h^2) \tag{23}$$

$$var\{\hat{m}(x)_{LLS}|X\} = \frac{1}{nh}\frac{\nu_0\,\sigma^2(x)}{f_X(x)} + o_p(\frac{1}{nh}) \tag{24}$$

where μ_i and ν_j are defined as the moments of the kernel density function K and the squared kernel density function, respectively, $\mu_i = \int u^i K(u)du$ and $\nu_j = \int u^j K^2(u)du$, for $i, j = 0, 1, 2, \ldots$. The bias and variance of the CQR method is:

$$bias\{\hat{m}(x)_{CQR}|X\} = \frac{1}{2}m''(x)\mu_2 h^2 + o_p(h^2) \tag{25}$$

$$var\{\hat{m}(x)_{CQR}|X\} = \frac{1}{nh}\frac{\nu_0\,\sigma^2(x)}{f_X(x)}R_1(q) + o_p(\frac{1}{nh}) \tag{26}$$

The term $R_1(q)$ arises in the variance for CQR, which is defined below as

$$R_1(q) = \frac{1}{q^2}\sum_{k=1}^{q}\sum_{k'=1}^{q}\frac{\tau_{kk'}}{f(c_k)f(c_{k'})} \tag{27}$$

where τ_k are the q quantiles, $c_k = F^{-1}(\tau_k)$ and $\tau_{kk'} = min(\tau_k, \tau_{k'}) - \tau_k\tau_{k'}$. The denominator terms $f(c_k)$ and $f(c_{k'})$ denote the density function of the error distribution and can be estimated through nonparametric density estimation of the residuals, discussed later. Combining both the bias and variance into MSE:

$$MSE\{\hat{m}_{LLS}(x)\} = \Big\{\frac{1}{2}m''(x)\mu_2\Big\}^2 h^4 + \frac{1}{nh}\frac{\nu_0\,\sigma^2(x)}{f_X(x)} + o_p\Big(h^4 + \frac{1}{nh}\Big) \tag{28}$$

$$MSE\{\hat{m}_{CQR}(x)\} = \Big\{\frac{1}{2}m''(x)\mu_2\Big\}^2 h^4 + \frac{1}{nh}\frac{\nu_0\,\sigma^2(x)}{f_X(x)}R_1(q) + o_p\Big(h^4 + \frac{1}{nh}\Big) \tag{29}$$

which is the sum of the squared bias and the variance. The optimal bandwidths for each of LLS and CQR can be computed by minimizing the MSE, and by taking a ratio of the optimal MSE's, we get the result that

$$h^{opt}_{CQR} = R_1(q)^{1/5}\, h^{opt}_{LLS} \tag{30}$$

so the optimal bandwidth for the CQR method can be computed by finding the optimal bandwidth for LLS, and scaling it by a factor of $R_1(q)^{1/5}$. Therefore, when $R_1(q)^{1/5} < 1$, the bandwidth for CQR will be smaller than that for LLS, and the opposite is true when $R_1(q)^{1/5} > 1$.

With this optimal bandwidth, the optimal MSE for CQR and LLS can be compared, and as n approaches ∞, the ratio of MSE's is connected by the term $R_1(q)$

$$\frac{MSE_{opt}\{\hat{m}_{LLS}(x)\}}{MSE_{opt}\{\hat{m}_{CQR}(x)\}} \to R_1(q)^{-4/5}. \tag{31}$$

5.3.2 Asymptotic Properties for Residual Based CQR

Using the previous sections, we attempt to develop the asymptotic results for the two step residual based CQR method we propose. Begin again with the nonparametric regression in the first step:

$$Y = m(X) + \sigma(X)\epsilon \tag{32}$$

with $m(x) = E(Y|X = x)$, $\sigma^2(x) = E(Y^2|X = x) - m^2(x)$ and $\epsilon = \frac{Y-m(X)}{\sigma(X)}$ having mean 0 and variance 1. Let $\tilde{m}(x)$ be the estimate of $m(x)$. In the second step of the two-step residual approach, we now model the squared residuals $(Y - \tilde{m}(X))^2$ as

$$(Y - \tilde{m}(X))^2 = r(X) + \tilde{\sigma}(X)\tilde{\epsilon} \tag{33}$$

where $r(x)$ is the conditional mean of the squared residuals and error term $\tilde{\sigma}(X)\tilde{\epsilon}$ similar to the error term from Equation 32, with a distinction in notation to show that it is for the second step. Like the definition of $m(x)$ in Equation 32, $r(x)$ can be expressed as the conditional expectation of the squared residual:

$$\begin{aligned} r(x) &= E((Y - \tilde{m}(X))^2|X = x) \\ &= E(Y^2|X = x) - 2E(Y\tilde{m}(X)|X = x) + E(\tilde{m}^2(X)|X = x) \\ &= \sigma^2(x) + m^2(x) - 2\tilde{m}(x)m(x) + \tilde{m}^2(x) \\ &= \sigma^2(x) + (m(x) - \tilde{m}(x))^2. \end{aligned} \tag{34}$$

This representation above shows that $r(x)$ is the sum of the variance and squared bias of the original response variable Y. Similarly, $\tilde{\sigma}^2(x)$ can be written as the conditional expectation below:

$$\tilde{\sigma}^2(x) = E((Y - \tilde{m}(X))^4 | X = x) - r^2(x) \tag{35}$$

Let $\tilde{r}$ be an approximation of $r(x)$, the bias of $\tilde{r}(x)$ (coming from Equation 25)

$$Bias[\tilde{r}(x)|X] = \frac{1}{2} r''(x) \mu_2 h^2 + o_p(h^2) \tag{36}$$

and variance (from Equation 26)

$$Var[\tilde{r}(x)|X] = \frac{1}{nh} \frac{\nu_0 \, \tilde{\sigma}^2(x)}{\tilde{f}_X(x)} \tilde{R}_1(q) + o_p\Big(\frac{1}{nh}\Big) \tag{37}$$

where $\tilde{R}_1(q)$ is calculated similarly to $R_q(q)$, but for the second step.

5.4 Bandwidth Selection

To implement the residual based local least squares and local CQR methods, bandwidths must first be selected for each nonparametric regression. Ruppert, Sheather and Wand (1995) give three methods for bandwidth selection in local least squares polynomial regression. These methods are referred to as plug-in methods, as they estimate functionals in the asymptotically optimal bandwidth in kernel density estimation. The MISE-optimal bandwidth has the following asymptotic approximation:

$$h_{MISE} \approx \left(\frac{(p+1)(p!)^2 R(K_p) \int_S v(x)\, dx}{2\mu_{p+1}(K_p)^2 \int_S m^{(p+1)}(x)^2 f(x)\, dx \; n} \right)^{1/(2p+3)} \tag{38}$$

Methods for selecting the bandwidth rely on approximations for the integrals in equation 38. When errors are homoscedastic, the numerator will be replaced with a local approximation for variance and the denominator will be replaced with an approximation for the mean regression function. Ruppert gives a solution for a plug-in (referred to also as the rule of thumb bandwidth, or ROT) bandwidth with the MISE-optimal bandwidth for the local linear kernel estimator:

$$h_{ROT} = C_1(K) \Big(\frac{\sigma^2 (b-a)}{\theta_{22} \; n} \Big)^{1/5} \tag{39}$$

where $C_1(K) = \left[\frac{R(K)}{\mu_2(K)^2}\right]^{1/5}$, which is a kernel-dependent constant. The term in denominator is defined as:

$$\theta_{rs} = \int m^{(r)}(x)m^{(s)}(x)f(x)\,dx \tag{40}$$

where $r, s \geq 0$ and $r + s$ is even. Detailed by Ruppert, to create an estimate of θ_{rs} (here r, s=2 will be used), a least squares blocked quartic approach can be used. Begin by partitioning the data into N subgroups, or blocks (where the independent "X" variable is sorted in increasing order). Each block is to contain the same amount of data points, instead of dividing the range into equal widths. With N as the number of subgroups, define χ_j be the j^{th} subsample of the ordered $x_i's$. If N divides the total number of observations n, then each subgroup χ_j will have equal sample size; otherwise, the observations can be rounded off in a way to have each subgroup of roughly the same size. In the blocked quartic regression, a degree 4 polynomial is fit on each of the N subgroups of data. Applying Equation 40 to each of the χ_j subgroups yields Equation 41. The first summation is for the blocked quartic regression across each of the j groups and the second takes the $x_i's$ from the j^{th} subgroup, indicated by the bolded **1** with subscript in the equation.

$$\hat{\theta}^Q_{rs}(N) = \frac{1}{n}\sum_{j=1}^{N}\sum_{i=1}^{n}(\hat{m}^Q_j)^{(r)}(x_i)(\hat{m}^Q_j)^{(s)}(x_i)\mathbf{1}_{(X_i \in \chi_j)} \tag{41}$$

A blocked quartic estimate of the variance is also used, written in Equation 42

$$\hat{\sigma}^2_Q(N) = \frac{1}{n - 5N}\sum_{j=1}^{N}\sum_{i=1}^{n}\{Y_i - \hat{m}^Q_j(X_i)\}^2\mathbf{1}_{(x_i \in \chi_j)} \tag{42}$$

To choose the number of subgroups N, Mallow's C_p (1973), used to assess fit in regression models, gives a good solution, with some modification to the blocked regression design which is used. The N (which ranges from $1, 2, \ldots, N_{max}$) to be used should minimize:

$$\frac{RSS(N)}{RSS(N_{max})/(n - 5N_{max})} - (n - 10N), \tag{43}$$

where $RSS(N)$ is the residual sum of squares based on the blocked quartic regression with N subgroups. Equation 44 gives a way to choose N_{max}:

$$N_{max} = max(min(\lfloor n/20 \rfloor, N^*), 1), \tag{44}$$

with N^* being an integer chosen to be 5. Higher choices of N^* would allow for more subgroups being chosen, resulting in finer estimates on smaller blocks of data (each χ_j subsample would have fewer observations). Additional details for Equations 41 and 42 follow.

In implementing Equation 41, the form of the mean regression function $m(x)$ on each subgroup χ_j is a 4th degree polynomial resembling:

$$\hat{m}_j^Q(x) = \hat{\beta}_{0j} + \hat{\beta}_{1j}\, x_i + \hat{\beta}_{2j}\, x_i^2 + \hat{\beta}_{3j}\, x_i^3 + \hat{\beta}_{4j}\, x_i^4.$$

Equation 41 can then be written as shown below:

$$\begin{aligned}\hat{\theta}_{22}^Q(N) &= \frac{1}{n}\sum_{j=1}^{N}\sum_{i=1}^{n}(\hat{m}_j^Q)^{(2)}(x_i)(\hat{m}_j^Q)^{(2)}(x_i)\mathbf{1}_{(X_i \in \chi_j)} \\ &= \frac{1}{n}\sum_{j=1}^{N}\sum_{i=(j-1)n/N+1}^{jn/N}\left(2\hat{\beta}_{2j} + 6\hat{\beta}_{3j}\, x_i + 12\hat{\beta}_{4j}\, x_i^2\right)^2.\end{aligned}$$

The coefficients of 2, 6 and 12 on the terms $\hat{\beta}_{2j}$, $\hat{\beta}_{3j}$ and $\hat{\beta}_{4j}$, respectively come from the 2nd derivative of the function, while the $\hat{\beta}_{0j}$ and $\hat{\beta}_{1j}$ terms vanish. These coefficients are computed for each of the subgroups (denoted with the additional j subscript) with their respective subset of the data. Similarly, for σ_Q^2:

$$\begin{aligned}\hat{\sigma}_Q^2(N) &= \frac{1}{n-5N}\sum_{j=1}^{N}\sum_{i=1}^{n}\{Y_i - \hat{m}_j^Q(X_i)\}^2\mathbf{1}_{(x_i \in \chi_j)} \\ &= \frac{1}{n-5N}\sum_{j=1}^{N}\sum_{i=(j-1)n/N+1}^{jn/N}\left(y_i - \hat{\beta}_{0j} - \hat{\beta}_{1j}\, x_i - \hat{\beta}_{2j}\, x_i^2 - \hat{\beta}_{3j}\, x_i^3 - \hat{\beta}_{4j}\, x_i^4\right)^2.\end{aligned}$$

Again, each N in the range $(1, 2, \ldots, N_{max})$ is checked to see which results in minimum $RSS(N)$. With both $\hat{\theta}_{22}^Q(N)$ and $\hat{\sigma}_Q^2(N)$, the rule of thumb bandwidth in Equation 39 can be found (Schindler 2011).

The steps for the residual based CQR (or LLS) are summarized below:

1. Compute bandwidth h_{ROT} (Equation 39 in Section 5.4) for LLS.

2. Fit LLS with h_{ROT} and compute estimates for the density function of the error distribution

for $R_1(q)$ (Equation 27 in Section 5.3.1).

3. For LLS, the bandwidth to be used is h_{ROT} from Step 1. For CQR, scale the bandwidth for LLS by a factor of $R_1(q)^{1/5}$.

4. Compute squared residuals $(Y - \tilde{m}(x))^2$ from first non-parametric regression, either with LLS or CQR.

5. Treat the squared residuals from Step 4 as the new response variable for the second step. Recompute the bandwidth h_{ROT} (Step 1) and $R_1(q)$ (Step 2), keeping the same covariate in the second step.

6. Perform another round of LLS or CQR on the squared residuals, resulting in the estimated volatility.

5.5 Calculating Volatility for Model Assessment

We will be modeling the monthly volatility with the proposed method of residual based local CQR and compare it to both a local least squares and GARCH(1,1) model. The data for application of the methods are daily returns for 990 firms. The monthly volatility is what will be considered for modeling. To aggregate the daily return data to a monthly return, we take the natural logarithm of the price of the first trading day in the month subtracted from the natural logarithm of the price of the last trading day in the month. Logarithmic (log) returns are commonly used (Tsay 2010, Miskolczi 2017) due to additivity when dealing with multi-period returns. A multi-period return can be written as a product of the individual period returns. Log returns are instead time additive (i.e. the log of a product becomes the sum of the logs).

As stated before, the log returns will be used to model the monthly volatility. However, the actual volatility is not observable, making it difficult to quantify. We use a method by Tsay (2010) to measure the volatility of the returns data. Monthly volatility can be estimated by

$$Var(r_t) = nVar(r_{t,1}) + 2(n-1)Cov(r_{t,i}, r_{t,j}),$$

where r_t is the t$^{\text{th}}$ month log return, $r_{t,i}$ is the i$^{\text{th}}$ daily return in month t. This can be estimated by

$$\hat{\sigma}_m^2 = \tfrac{n}{n-1} \textstyle\sum_{i=1}^{n} (r_{t,i} - \bar{r}_t)^2 + 2 \sum_{i=1}^{n-1} (r_{t,i} - \bar{r}_t)(r_{t,i+1} - \bar{r}_t),$$

which accounts for the variability of daily returns within each month. Predictions made for volatility using the models described in previous sections will be compared to these estimates of the actual volatility.

5.6 Results

The residual based local CQR and local least squares (LLS) are tested on monthly returns of 990 firms. These will be referred to as CQR and LLS, respectively, in tables of the results section. Data comes from the Center for Research in Security Prices (CRSP). In total there are 192 monthly returns for each firm, from January of 2000 to December of 2015. We first apply local CQR and LLS to each firms' series using 4 different predictors as our independent (X) variable: lagged monthly return, de-meaned lagged monthly return, S&P index returns, and a volatility index series. With each of these predictors, both methods are applied and compared to see how predictions perform on both a training and one-step forecasting periods. Data is split to have roughly 2/3 of the 192 months for training and the remaining for forecasting. Predictions are compared to the realized volatility which is estimated using the method described previously. Error is compared using both mean squared and mean absolute error on the training and forecasting periods with each method. Let $\hat{m}(x_m)$ be the estimated volatility for month m from the second round of the residual based estimation and σ_m^2 be the realized volatility from month m. The mean squared error (MSE) is defined as

$$\frac{1}{n}\sum_{m=1}^{n}(\sigma_m^2 - \hat{m}(x_m))^2$$

Similarly, the mean absolute deviation (MAD) is simply the mean of the absolute errors

$$\frac{1}{n}\sum_{m=1}^{n}|\sigma_m^2 - \hat{m}(x_m)|$$

To summarize the results, Table 5.1 aggregates which method (CQR or LLS) performs better across the 4 models (one model for each predictor used) shown in Table 5.2. For example, when comparing the MSE in training, CQR is outperformed by LLS in both raw number of firms and the overall mean error (last 2 columns of Table 5.1) across all 4 models. Table 5.1 depicts this with a 0/4, favoring LLS in all 4 cases. In terms of CQR MAD in training, CQR is best in a majority of firms for all models except the VIX series (so 3/4 for raw number of firms), and produces lesser MAD in all 4 models (4/4). The same explanation goes for the forecasting periods.

Table 5.1: Summary of MSE and MAD Results with Epanechnikov Kernel

Train/Forecast	Metric	Raw Number of Firms	Error for Metric
Training	CQR MSE	0/4	0/4
	CQR MAD	3/4	4/4
Forecasting	CQR MSE	3/4	1/4
	CQR MAD	4/4	4/4

We next investigate each of the 4 models more closely. The first series used is a lagged (1) monthly return series of the individual firms. In Table 5.2, the results for both MSE and MAD are shown on each of the training and forecasting data. For example, the MSE on the training data for LLS method using the lagged (1) series is shown by the 0.00151 in the "Error for Metric" column. This is the MSE across predictions on the training data for all 990 firms. Similarly, the MSE for the 990 firms' training data using CQR method is 0.00158. In addition to the MSE, the Best (of 990) column shows the number of the 990 firms in which LLS performs better than CQR. We can see that when comparing MSE on the training data using the lagged (1) series, the volatility of 883 of the 990 firms is more closely estimated by LLS than CQR. Conversely, 107 of the firms are more closely estimated when using CQR (the pairs of numbers will always add to 990). In the forecasting period, we can see that CQR performs better than LLS (in 653 to 337 firms, again when comparing MSE). When looking at the MAD, we can see that CQR performs better both in training (676 to 314) and in forecasting (854 to 136), with lower MAD error in each. Error in both training and forecasting is lower for CQR when comparing the MAD. Intuitively, it makes sense that looking at MAD in addition to MSE would benefit the CQR method as the penalty in CQR is also the least absolute deviation.

In the second series, instead of applying the two-step methods, the mean of each firms' series is first subtracted from the monthly returns (referred to as the de-meaned series). Then, these values are squared and used as the squared residuals for the second step in which another round of CQR or LLS is performed (in effect, the first step of estimating the mean has been reduced by subtracting

Table 5.2: MSE and MAD Results with Epanechnikov Kernel

Model	Metric	Train/Forecast	Method	Best (of 990)	Error for Metric
Lagged (1) Series	MSE	Training	LLS	883	0.00151
			CQR	107	0.00158
		Forecasting	LLS	337	0.00047
			CQR	653	0.00100
	MAD	Training	LLS	314	0.01417
			CQR	676	0.01324
		Forecasting	LLS	136	0.00984
			CQR	854	0.00777
De-meaned Series	MSE	Training	LLS	838	0.00149
			CQR	152	0.00153
		Forecasting	LLS	289	0.00179
			CQR	701	0.00377
	MAD	Training	LLS	203	0.01435
			CQR	787	0.01313
		Forecasting	LLS	131	0.01110
			CQR	859	0.01028
S&P Series	MSE	Training	LLS	819	0.00149
			CQR	171	0.00155
		Forecasting	LLS	470	0.00069
			CQR	520	0.00069
	MAD	Training	LLS	409	0.01399
			CQR	581	0.01344
		Forecasting	LLS	223	0.00980
			CQR	767	0.00805
VIX Series	MSE	Training	LLS	810	0.00153
			CQR	180	0.00160
		Forecasting	LLS	499	0.00127
			CQR	491	0.00191
	MAD	Training	LLS	502	0.01386
			CQR	488	0.01360
		Forecasting	LLS	359	0.00957
			CQR	631	0.00940

the overall mean first). The lagged de-meaned log return series is then again used as the predictor in LLS and CQR. Results are similar to the lagged (1) series, with LLS performing better in the training set when considering the MSE (838 to 152), but performing worse in forecasting (701 to 289 in favor of CQR). When comparing the MAD, CQR is superior in predicting volatility for a vast majority of the firms in both training (787 to 203) and forecasting (859 to 131). If wanting to compare these two methods, the two-step residual based method generally produces less error in forecasting than when removing the mean and running one round of LLS or CQR. For instance when comparing MAD, the lagged (1) series produces approximately 11% and 24% less error for LLS and CQR, respectively, than the corresponding de-meaned series. Otherwise in training, the de-meaned series produces slightly less error than the lagged (1) series when comparing MSE (0.00149 vs. 0.00151 for LLS and 0.00153 vs. 0.00158 for CQR) or the MAD for CQR (0.01313 vs. 0.01324).

In the next series, the S&P log return index is used as the predictor for volatility. With MSE as the metric for comparison, LLS performs better in training (819 to 171), but CQR is best in forecasting (520 to 470). With respect to the MAD, results in both training (581 to 409) and forecasting (767 to 223) are in favor of CQR. The lower mean error is consistent with which method performs best at a majority of firms.

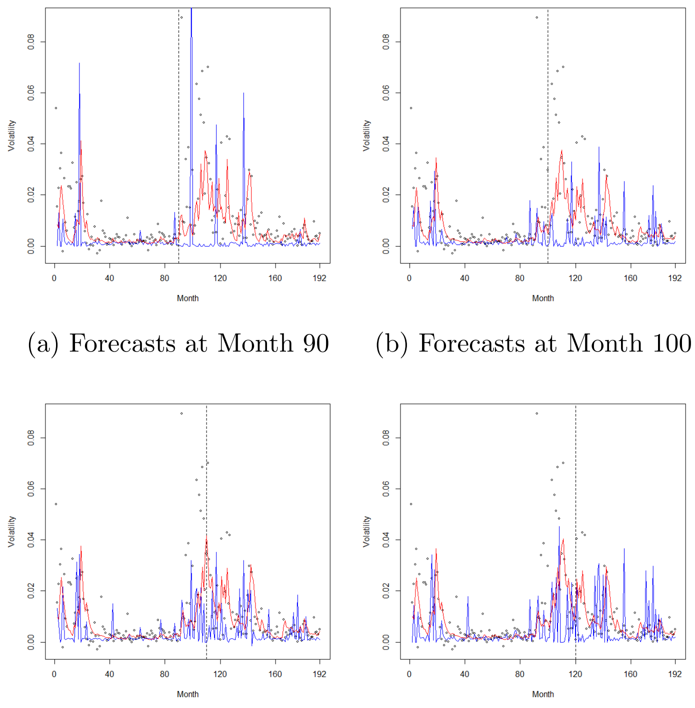
Lastly, a volatility index (VIX) is used as the predictor for volatility. The VIX is an index derived from S&P 500 index options which is designed to measure 30 day market volatility. For MSE, LLS is superior in training (810 to 180) while the mean error is very close (0.00153 to 0.00160). For forecasted points, LLS is best but by a very small margin (499 to 491); however, mean error is substantially smaller for LLS. This is likely attributable to several firms with large mean squared errors, as evidenced by looking at the MAD which points in favor of CQR in the forecast period. With respect to the MAD, LLS performs better in a majority of the firms in training by a slim margin (502 to 488), albeit with 2% more mean error. In forecasting, CQR is again superior (631 to 359), with a small decrease in mean error across all firms.

Table 5.2 shows results for an Epanechnikov kernel. Results for using a Gaussian kernel are shown in Table A.4 of the Appendix. Results for Gaussian kernel are largely similar to that of using an Epanechnikov kernel; there is almost always agreement in which method (CQR or LLS) performs best in a majority of the firms for each combination of model and training/forecast period of the data. The use of a Gaussian kernel favors CQR even more in some instances. For example, the Gaussian kernel for the lagged (1) and Vix series gives even more favorable results for CQR than when using an Epanechnikov kernel. On the other hand, the Epanechnikov kernel yields more favorable results for CQR with the S%P series. The choice of method is more indicative of the performance than the choice of kernel.

5.7 Comparison to GARCH

In addition to comparing the CQR method to LLS, we would also like to compare how CQR performs to a GARCH(1,1) model. The GARCH(1,1) process is first simulated with a rolling forecast, where one-step ahead predictions are made at each point in the forecast period. That is to say, the forecast at period $t+1$ is made with information up to period t, and the forecast at period $t+2$ is then made with returns until period $t+1$, and so on. We test how these two methods perform at different training and forecasting periods. Month 110 on the x-axis corresponds to March of 2009, where the Dow Jones hit its lowest point during the recession of 2008. We try 4 different cutoff points for the end of the training period (beginning of forecasting): month 90, month 100, month 110, and month 120. These are labeled accordingly in the tables and figures, with a vertical line signifying the beginning month of forecasting. In all the figures, the blue line represents predictions from CQR and the red line represents predictions from GARCH. Figure 5.1 shows a firm which exhibits high volatility in the period from months 90 to 130. It can be seen that when forecasting begins in month 90 in Figure 5.1a, both models capture an increase in volatility. The CQR model largely predicts low volatility, with the exception of 3 large (yet short lived) spikes in predicted volatility. When forecasting begins in month 100 (shown in Figure 5.1b), both models again predict spikes during the volatile periods, with CQR continuing to predict periods of high volatility throughout the forecast period. Figures 5.1c and 5.1d show similar results when forecasting begins in months 110 and 120, respectively, with both capturing spikes in volatility around between months 110 to 120. The GARCH predicted volatility then tends to decrease throughout the forecast period, while CQR continues to predict several periods of sustained volatility. In all instances for Figure 5.1, GARCH produces less error than CQR.

Figure 5.2 similarly shows the difference in forecasts depending on which month is chosen for the start of the forecast period. Overall, the GARCH method performs better than CQR in both training and forecasting. However, CQR predicts larger spikes in volatility in the period of month 10 to 20 for this firm. The GARCH model creates a smooth estimate of sustained volatility, while CQR produces a sustained period of volatility with many spikes. For this firm, there is a small spike in



(a) Forecasts at Month 90 (b) Forecasts at Month 100

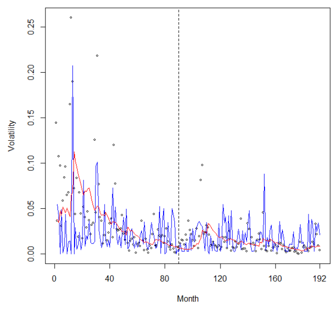
(c) Forecasts at Month 110 (d) Forecasts at Month 120

Figure 5.1: CQR vs. Garch Comparison Firm 647

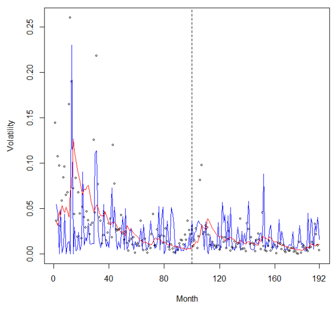
volatility around month 110, followed by a rather calm period with very few jumps in volatility. The CQR forecasts capture some of the spikes in the forecast period, but are generally overestimating this firm's realized volatility, while GARCH predicts a steady period of volatility.

Figure 5.3 shows an asset return series for firm 109, which exhibits high volatility early in the asset return history and also during the noted time of the recession. All plots in Figure 5.3 show similar results from GARCH and CQR. CQR seems to do a better job than GARCH in capturing a short spike around month 110. There are infrequent and short lived jumps in realized volatility for this firm and the GARCH model is unable to capture them. In all 4 instances, CQR produces less error in forecasting, but slightly more error in training.

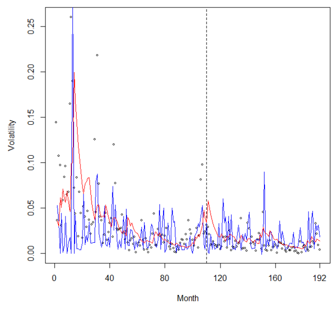
Figure 5.4 shows another asset return series for firm 684. In this instance across the 4 figures, CQR performs better in forecasting while GARCH does better in training. In Figure 5.4a with forecasting beginning at month 90, both methods predict spikes in volatility around month 110, followed by a



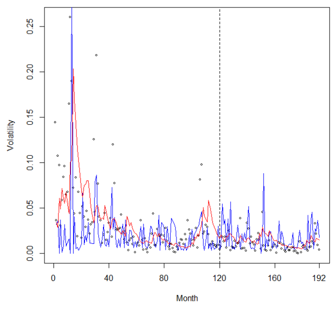
(a) Forecasts at Month 90



(b) Forecasts at Month 100



(c) Forecasts at Month 110

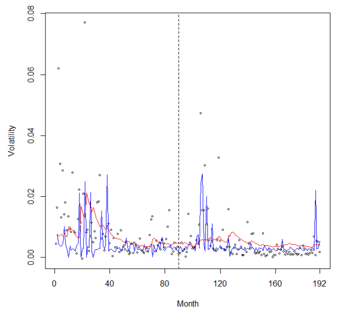


(d) Forecasts at Month 120

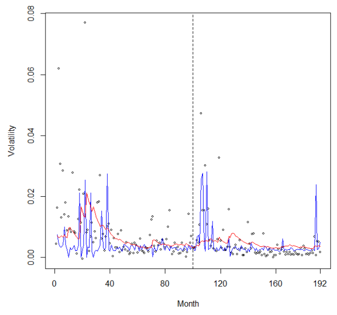
Figure 5.2: CQR vs. Garch Comparison Firm 511

sharp decline in volatility. Figures 5.4b and 5.4c exhibit similar patterns. In all of these subfigures, CQR predicts a spike in volatility at month 80 which GARCH does not capture. In Figure 5.4d where forecasting begins in month 120, the GARCH model looks largely the same in both training and forecasting periods. The CQR model predicts spikes in volatility at the same periods as before, but in much smaller magnitude in Figure 5.4d. This can be attributable to one of several possible reasons. First, the inclusion of additional points in the training portion of the data can introduce new values which dampen the predicted volatility at the points of interest. If the new data lies in the local neighborhood surrounding, for instance, month 80, then these values could greatly impact the prediction which is made at this point. Additionally, the size of the bandwidth which is chosen after the inclusion of more data in the training of the model can dampen the effect of large spikes in realized volatility.

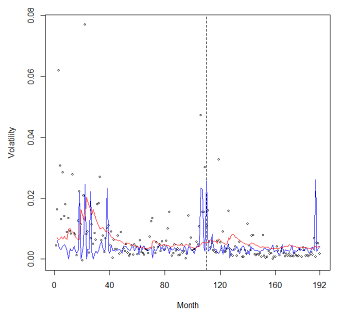
Results across all 990 firms are shown in Table 5.3. The MSE and MAD ratios are shown for both training and forecasting periods for each of the 4 starting points for the forecast period (month 90,



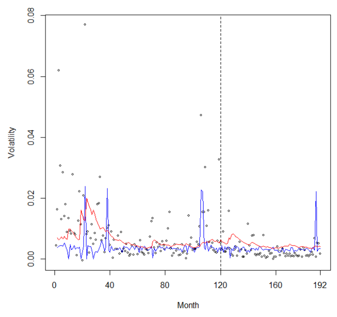
(a) Forecasts at Month 90



(b) Forecasts at Month 100



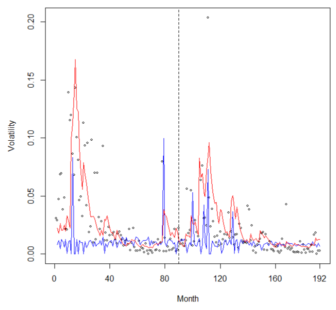
(c) Forecasts at Month 110



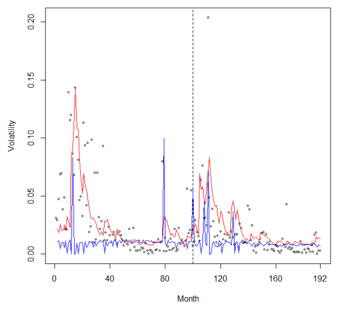
(d) Forecasts at Month 120

Figure 5.3: CQR vs. Garch Comparison Firm 109

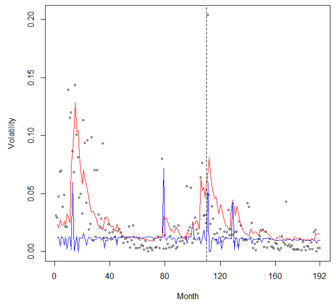
100, 110, 120). Again, the Best column reflects which method (GARCH or CQR) performed better in a majority of the individual firm series, and the last column shows the corresponding MSE or MAD across all firms using the combination of metric/methods. It can be seen that when forecasting begins in month 120, the CQR method performs better in forecasting MAD error in a majority of firms (546 to 444). For forecasts beginning in month 110, CQR performs comparably to GARCH, but not in a majority of firms (433 to 557), although producing less mean error (0.00927 to 0.01001). For forecasting beginning in month 90 or 100, GARCH is superior, with close to 90% majorities in number of firms and superior error in each category. From the 4 figures, we can see that the GARCH model generally does well in training, but in some cases worse in forecasting. CQR is able to capture large spikes in volatility, but only in small periods. Predictions for CQR are generally steady with occasional large spikes while the GARCH model produces a more smooth estimate to the volatility, resulting in overall less error when comparing the estimates to realized volatility. CQR performs local regression on the squared residuals from a previous round of local regression. When there are large residuals, there will also be a large estimated volatility. A GARCH model makes predictions



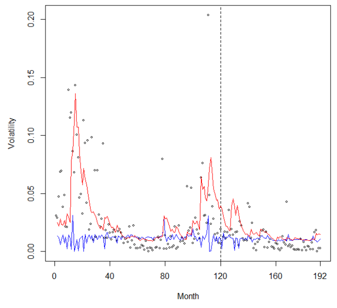
(a) Forecasts at Month 90



(b) Forecasts at Month 100



(c) Forecasts at Month 110



(d) Forecasts at Month 120

Figure 5.4: CQR vs. Garch Comparison Firm 684

based on previous shocks in the log return series, so a previous large value generally leads to another large prediction (and a previous small value similarly generally leads to another small value). With the CQR model, the predictions are not made with the previous value of the series in mind, but with the local region around the current value of the predictor "x" variable. If large log returns tend to have larger residuals, then estimates in those local regions will lead to higher volatility. But a larger log return will not necessarily lead to larger estimated volatility. One advantage of CQR is exhibited in Figure 5.4, near month 80. GARCH is unable to capture a sudden spike in volatility while CQR does. As stated before, GARCH creates smooth estimates, where the next prediction is made off the previous value(s). It would be unlikely to see a large jump in a predicted volatility when the surrounding data is comprised of periods of low volatility. As seen in Figure 5.4, GARCH does not predict a huge increase at the point near month 80. With CQR, however, performing local regression, it is possible to detect a large spike in a small local neighborhood of a point. Something similar was seen in Figure 5.3, where CQR was able to capture several spikes in volatility in between months 110 and 120, but GARCH did not. CQR is more robust to sudden events of high impact than GARCH.

Table 5.3: GARCH vs. CQR - MAD and MSE in Training and Forecasting for Varying Starting Months

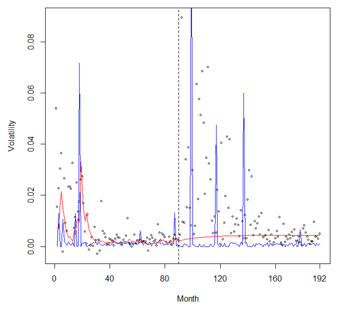
Start of Forecast	Metric	Train/Forecast	Method	Best (of 990)	Error for Metric
Month 90	MSE	Training	GARCH	917	0.00089
			CQR	73	0.00112
		Forecasting	GARCH	944	0.00096
			CQR	46	0.00212
	MAD	Training	GARCH	850	0.00986
			CQR	140	0.01107
		Forecasting	GARCH	902	0.00972
			CQR	88	0.01313
Month 100	MSE	Training	GARCH	925	0.00087
			CQR	65	0.00107
		Forecasting	GARCH	939	0.00095
			CQR	51	0.00237
	MAD	Training	GARCH	828	0.00991
			CQR	162	0.01093
		Forecasting	GARCH	873	0.00976
			CQR	117	0.01276
Month 110	MSE	Training	GARCH	893	0.00130
			CQR	97	0.00156
		Forecasting	GARCH	840	0.00040
			CQR	150	0.00206
	MAD	Training	GARCH	753	0.01208
			CQR	237	0.01294
		Forecasting	GARCH	611	0.00827
			CQR	379	0.00927
Month 120	MSE	Training	GARCH	880	0.00132
			CQR	110	0.00158
		Forecasting	GARCH	721	0.00024
			CQR	269	0.00120
	MAD	Training	GARCH	720	0.01263
			CQR	270	0.01339
		Forecasting	GARCH	472	0.00692
			CQR	518	0.00747

5.7.1 Comparison to N-Step Ahead Forecasts

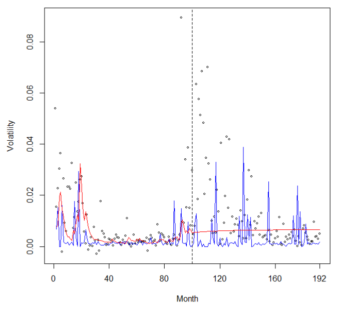
In another exercise to compare GARCH to CQR, we perform GARCH forecasts with a window of the entire forecasting period. That is, there are no longer rolling forecasts, and the entire window is forecasted without updating the previous periods' returns. Figure 5.5 shows n-step forecasts beginning at each of months 90, 100, 110 and 120. This is for the same firm seen in Figure 5.1. It can be seen that when forecasting begins in month 90 in Figure 5.5a, the GARCH model performs very poorly at predicting any spikes in volatility. The CQR model largely predicts low volatility, with the exception of 3 large (yet short lived) spikes in predicted volatility. A similar picture is shown in Figure 5.5b, where the GARCH volatility predictions are fairly constant and the CQR method forecasts several longer sustained spikes in volatility around months 120 and 130. When forecasting begins in month 110 in Figure 5.5c, the CQR method again accurately predicts some periods of sustained volatility, while the GARCH method predicts a huge increase in volatility which doesn't return to normal. Something similar is again seen in Figure 5.5d where the GARCH model predicts a continued increase in volatility from month 120 and on, while CQR predicts several periods of sustained volatility.

Figure 5.6 similarly shows the difference in forecasts depending on which month is chosen for the start of the forecast period. Overall, the GARCH method performs better in Figures 5.6a and 5.6b in both training and forecasting.However, CQR predicts larger periods of volatility around month 110, which corresponds to the worst times of the 2008 recession. Predictions on the training portions of Figures 5.6c and 5.6d are similar, but GARCH performs worse in forecasting periods than CQR. Both methods capture the period of high volatility in the beginning of the series, with CQR seeming to give more versatile results in the different lengths of forecasting periods.

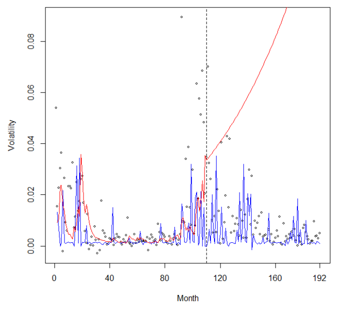
From the 2 figures, a pattern that emerges is that the GARCH models perform relatively well in the training periods, but begin to struggle in forecasting periods of high volatility. The autoregressive model tends to overestimate the future volatility when beginning the forecast in a period of high volatility. In some instances, it even predicts volatilty to increase at what appears to be an exponential rate. The CQR method, on the other hand, creates stable predictions around the periods of high volatilty, even when the forecast begins 10-20 months before the increase. The summary results are captured in Table 5.4, showing that CQR outperforms this autoregressive model



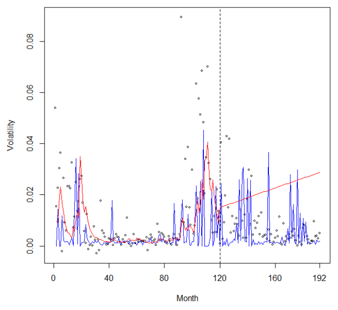
(a) Forecasts at Month 90



(b) Forecasts at Month 100



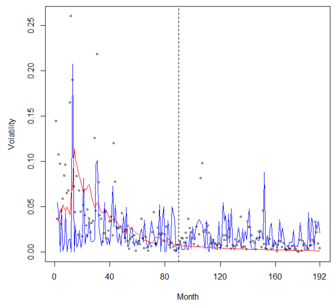
(c) Forecasts at Month 110



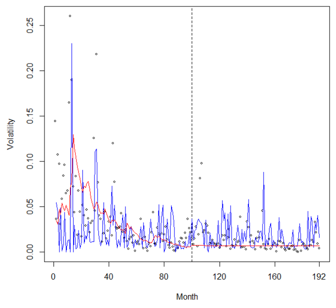
(d) Forecasts at Month 120

Figure 5.5: N-Step Forecasts for CQR and Garch for Firm 647

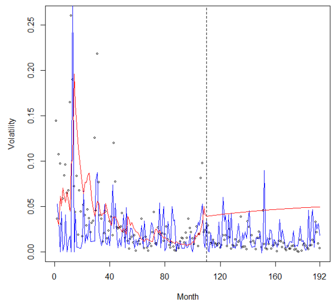
only in forecasting. Taking a quick glance, the findings look poor for CQR. But upon further investigation in Table 5.5, the error produced by CQR is very comparable to GARCH in many of the training instances. The MSE and MAD ratios are shown for both training and forecasting periods for each of the 4 starting points for the forecast period (month 90, 100, 110, 120). Again, the Best column reflects which method (GARCH or CQR) performed better in a majority of the individual firm series, and the last column shows the corresponding MSE or MAD across all firms using the combination of metric/methods. It can be seen that when forecasting begins in months 110 or 120, the CQR method captures a clear advantage in forecasting error, when comparing either MSE or MAD. For forecasting beginning in month 90, although GARCH performs better in forecasting error for a majority of the firms, CQR produces less absolute error. In training, GARCH is superior in raw counts of firms, but CQR does not fall far behind when comparing MAD or MSE.



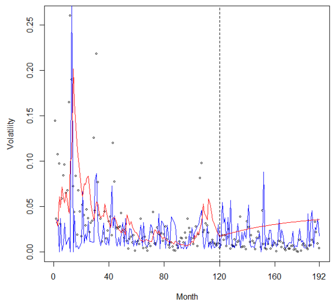
(a) Forecasts at Month 90



(b) Forecasts at Month 100



(c) Forecasts at Month 110



(d) Forecasts at Month 120

Figure 5.6: N-Step Forecasts for CQR and Garch for Firm 511

Table 5.4: GARCH vs. CQR Summary

Train/Forecast	Metric	Raw Number of Firms	Error for Metric
Training	CQR MSE	0/4	0/4
	CQR MAD	0/4	0/4
Forecasting	CQR MSE	2/4	2/4
	CQR MAD	2/4	3/4

Table 5.5: Garch vs. CQR - MAD and MSE in Training and Forecasting for Varying Starting Months Using N-Step Ahead Forecasts

Start of Forecast	Metric	Train/Forecast	Method	Best (of 990)	Error for Metric
Month 90	MSE	Training	GARCH	916	0.00092
			CQR	74	0.00112
		Forecasting	GARCH	739	0.00190
			CQR	251	0.00212
	MAD	Training	GARCH	833	0.01002
			CQR	157	0.01107
		Forecasting	GARCH	663	0.01270
			CQR	327	0.01313
Month 100	MSE	Training	GARCH	927	0.00088
			CQR	63	0.00107
		Forecasting	GARCH	783	0.00258
			CQR	207	0.00237
	MAD	Training	GARCH	817	0.01000
			CQR	173	0.01093
		Forecasting	GARCH	665	0.01454
			CQR	325	0.01276
Month 110	MSE	Training	GARCH	890	0.00146
			CQR	100	0.00156
		Forecasting	GARCH	434	0.01042
			CQR	556	0.00206
	MAD	Training	GARCH	742	0.01234
			CQR	248	0.01294
		Forecasting	GARCH	199	0.03565
			CQR	791	0.00927
Month 120	MSE	Training	GARCH	873	0.00156
			CQR	117	0.00158
		Forecasting	GARCH	347	0.00572
			CQR	643	0.00120
	MAD	Training	GARCH	712	0.01292
			CQR	278	0.01339
		Forecasting	GARCH	134	0.02266
			CQR	856	0.00747

Chapter 6

Discussion

We began this dissertation by introducing several two-step nonparametric approaches for smoothing estimation of extreme quantiles for daily temperatures in U.S. cities. We argue that rather than performing quantile regression to estimate the 5^{th} and 95^{th} percentiles of daily temperatures, respectively, a number of nonparametric regression techniques can be used on summarized data instead. If the data can first be aggregated by some time interval (in our examples this was taking the corresponding 5^{th} or 95^{th} percentiles of the yearly daily temperatures), then these values (denoted as raw estimates) can be smoothed with nonparametric methods such as kernel, local polynomial or spline smoothing. Through both simulation and application, we show that the two-step nonparametric methods outperform quantile regression, as the former methods are able to capture variability between interior points, while quantile regression can only accurately capture the overall trends of the time varying quantiles. The nonparametric smoothing of the raw estimates is computationally simple and easy to implement.

While the two-step method is appealing, we also explored a one-step method for estimating time variant quantiles and time varying parameters. Again, we consider data which was segmented by some independent variable such as time, so that there were many observations which shared the same time period. The proposed one-step solution would seek to utilize maximum likelihood estimation by applying a local log-likelihood solution. This is tested for two scenarios: a discrete and continuous data type. In the discrete case, we apply the one-step technique to both a dataset where the response variable is time to conception for women entering marriage and in simulations. We treat the data as following a geometric distribution where the parameter of interest is the probability of success (or pregnancy in the application data). When solving for the unknown parameter θ, we also consider a polynomial expansion up to degree 5. All results are furthermore compared to the respective two-step smoothing of same polynomial degree. For low order polynomials, the one-step procedure outperforms the two-step procedure of similar polynomial expansion on the application data. For higher order degrees, results sometimes flip in favor of the two-step method, and there are some large disparities in the comparisons of the methods. This seems to indicate that for high order polynomials, results may become unstable in one or both of the methods. In simulation, when comparing the mean absolute deviation, results tend to favor the one-step procedures, both in raw error and by a majority of time points. We also apply the one-step procedure for the scenario with continuous data. Here we model the mean level of oxides of nitrogen as a local log-likelihood with the normal distribution. Estimation is performed under two scenarios: one where the variance is treated as locally constant and the other where the variance is allowed to vary and is estimated locally. On the application data, improvements in estimation are made when considering the variance as locally estimable. The largest decrease in prediction error generally occurs when going from the case where the variance is constant to where the variance is estimated locally with a degree 0 polynomial. Error continues to decrease as the variance is estimated with higher degree polynomials, but generally at a diminishing rate. Similar observations are made from the simulation under the time variant variable parameter framework. In both application and simulation, we can see that there is merit to using the one-step procedures.

Lastly, we investigated a residual based estimator of volatility and proposed a new method using local composite quantile regression (CQR). These methods are compared on 990 different firms' monthly return data. In comparing the local least squares (LLS) to local composite quantile regression approaches, local CQR consistently outperforms the LLS method in forecasting and in training when comparing the mean absolute deviation. This is true when modeling the monthly logarithmic returns against several different predictors: two lagged series, an S&P index series, and a volatility indicator (VIX). In comparison to a GARCH(1,1) model with rolling forecasts, CQR displays some advantages in being able to capture sudden, short lived spikes in volatility, while the autoregressive model generally produces smoother estimates for periods of sustained volatility. In forecasting, CQR outperforms GARCH in scenarios where the forecasts begin after a very volatile period. This distinction is made even clearer in the scenario where n-step ahead forecasts are made instead of rolling forecasts. When the forecasts are created without updating the previous periods' return, the GARCH predictions eventually stabilize on a trajectory and follow it throughout the forecast period. In this scenario, all the forecasts are built on forecasted values, so the process quickly converges on a trend. The local CQR method is robust and sensitive to outliers in volatility estimation and a safe alternative to local least squares regression.

There are many other theoretical and methodological aspects which can be further investigated. First, other smoothing methods such as wavelets, b-splines, p-splines and other basis approximations can be investigated. Second, multiple covariates can be considered in both the one and two-step procedures in addition to the models for the conditional variance. Third, other bandwidth selection methods merit investigation, including non-global options.

References

- Arlot, S. and Celisse, A. A Survey of cross-valdation procedures for model selection. *Statistics Surveys, Institute of Mathematical Statistics (IMS), 4*(1), 40-79, doi: 10.1214/09-SS054
- Arnerić, J., Poklepović, T. and Teai, J. W. (2018). Neural network approach in forecasting realized variance using high-frequency data. *Business Systems Research, 9*(20), 18-34. doi: 10.2478/bsrj-2018-0016
- Avery, M. Literature review for local polynomial regression. Retrieved from https://p-dfs.sem-anticscholar.org/c017/6f034b9730a01914ef48221630cb497610d7.pdf
- Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics, 31*(3), 307-327. doi: https://doi.org/10.1016/0304-4076(86)90063-1
- Chen, L. H., Cheng, M. Y. and Peng, L. (2009). Conditional variance estimation in heterscedastic regression models. *Journal of Statistical Planning and Inference, 139*(2), 236-245. doi: https://doi.org/10.1016/j.jspi.2008.04.020
- Chowdhury, M., Wu, C. and Modarres, R. (2017). Local Box-Cox transformation on time-varying parametric models for smoothing estimation of conditional CDF with longitudinal data. *Journal of Statistical Computation and Simulation, 87*(15), 2900-2914. doi: https://doi.-org/10.1080/00949655.2017.1347656
- Chowdhury, M., Wu, C. and Modarres, R. (2018). Nonparametric estimation of conditional distribution functions with longitudinal data and time-varying parametric models. *Metrika, 81*(1), 61-83. doi: 10.1007/s00184-017-0634-z
- Chowdhury, M. (2017). Nonparametric estimation of time-variant parametric models with application to cross-sectional data. *Journal of Japan Statistical Society, 47*(2), 197-220.
- Cleveland, W. S (1979). Robust locally weighted regression and smoothing scatterplots *Journal of the American Statistical Association, 74*(368), 829-836. doi: 10.2307/2286407
- Dufays, A. (2015). Infinite-state markov-switching for dynamic volatilty. *Journal of Financial Econometrics, 14*(2), 418-460. doi: https://doi.org/10.1093/jjfinec/nbv017
- Dutta, A. (2014). Modelling volatility: symmetric or asymmetric GARCH models? *Journal of Statistics: Advances in Theory and Applications, 12*(2), 99-108. Retrieved from https://pdfs.s-emanticscholar.org/5c18/f7ade84ed77319381a45db4f74bde243eec8.pdf

- Engle, Robert F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica, 50*(4), 987-1008. doi: 10.2307/1912773
- Eubank, R, L. (1988). *Spline smoothing and nonparametric regression.* New York, New York: Marcel Dekker, Inc. pp. 438.
- Fan, J. (1992). Design-adaptive nonparametric regression. *Journal of the American Statistical Association, 87*(420), 998-1004. doi: 10.2307/2290637
- Fan, J. (1993). Local linear regression smoothers and their minimax efficiency. *Annals of Statistics, 21*(1), 196-216. doi: 10.1214/aos/1176349022
- Fan, J., Farmen, M. and Gijbels, I. (1998). Local maximum likelihood estimation and inference. *Journal of the Royal Statistical Society. Series B (Statistical Methodology), 60*(3), 591-608. doi: https://www.jstor.org/stable/2985933
- Fan, J. and Gijbels, I. (1991). Local linear smoothers in regression function estimation. Retrieved from https://pdfs.semanticscholar.org/3d89/0548efbd8e44518db3517349f56c7eb1831-9.pdf
- Fan, J. and Gijbels, I. (1992). Variable bandwidth and local linear regression smoothers *The Annals of Statistics, 20*(4), 2008-2036. doi: 10.1214/aos/1176348900
- Fan, J. and Gijbels, I. (1996). *Local polynomial modelling and its applications.* Boca Raton, Florida: Chapman & Hall.
- Fan, J. and Yao, Q. (1998). Efficient estimation of conditional variance functions in stochastic regression. *Biometrika, 85*(3), 645-660. doi: https://doi.org/10.1093/biom- et/85.3.645
- Fan, J. and Zhang, J.T.(2000). Two-step estimation of functional linear models with applications to longitudinal data. *Journal of the Royal Statistical Society. Series B, 62*(2), 303-322. Retrieved from http://www.jstor.org/stable/3088861
- Francisco-Fernández, M., Opsomer J., and Vilar-Fernández, J. M. (2004). Plug-in bandwidth selector for local polynomial regression estimator with correlated errors. *Journal of Nonparametric Statistics, 16*(1-2), 127-151. doi: https://doi.org/10.1080/10485250310001622848
- Füss, Roland, K., Dieter, G. and Adams, Z. Value at risk, GARCH modelling and the forecasting of hedge fund return volatility. *Journal of Derivatives & Hedge Funds, 13*(1), 2-25. doi: https://doi.org/10.1057/palgrave.jdhf.1850048

- Geisser, S. (1975). The predictive sample reuse method with applications. *Journal of the American Statistical Association, 70*(350), 320-328. doi: 10.2307/2285815
- Giordano, F. and Parrella, M. L. (2014). Global adaptive smoothing regression. *Electronic Journal of Statistics, 8*(2), 2848-2878. doi: https://projecteuclid.org/euclid.ejs/1420726193
- Giordano, F. and Parrella, M. L. (2016). Efficient nonparametric estimation and inference for the volatility function. *A Journal of Theoretical and Applied Statistics, 53*(4), 770-791. doi: https://doi.org/10.1080/02331888.2019.1615066
- Glosten, L. R., Jagannathan, R. and Runkle, D. E. (1993). On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance, 48*(5), 1779-1801. doi: https://doi.org/10.1111/j.1540-6261.1993.tb05128.x
- Hansen, P. R. and Lunde, A. (2001). A comparison of volatility models: does anything beat a GARCH(1,1)? Center for Analytical Finance, University of Aarhus, Working Paper Series No. 84. Retrieved from http://www-stat.wharton.upenn.edu/ steele/Courses/434/434Context/G-ARCH/HansenLunde01.pdf
- Hart, J. D. and Vieu, P. (1990). Data-driven bandwidth choice for density estimation based on dependent data. *The Annals of Statistics, 18*(2), 873-990. Retrieved from https://projecteucli-d.org/euclid.aos/1176347630
- Hart, T.D. and Wehrly, T.E. (1986). Kernel regressionn estimation using repeated measurements data. *Journal of American Statistical Assosciation, 81*(396), 1080-1088. doi: https://www.tandfonline.com/doi/abs/10.1080/01621459.1986.10478377
- Hogg, R. V. and Tanis, Elliot A. (2010). *Probability and statistical inference (Eighth Edition)*. Upper Saddle River, New Jersey: Pearson Prentice Hall. 273-280
- Hoover, D. R., Rice, J. A., Wu, C. O. and Yang, L. P. (1998). Nonparametric smoothing estimates of time-varying coefficient models with longitudinal data. *Biometrika, 85*(4), 809-822.
- Jin, S., Su, L. and Xiao, Z. (2014). Adaptive nonparametric regression with conditonal heteroskedasticity. *Econometric Theory, 31*(6), 1153-1191. doi: 10.1017/S0266466614- 000450
- Johnsson, O. (2018). Predicting stock index volatilty using artificial neural networks (Unpublished master's thesis). Lund University.

- Kai, B., Li, R., and Zou, H. (2010). Local CQR Smoothing: An efficient and safe alternative to local polynomial regression. *Journal of the Royal Statistical Society. Series B, Statistical methodology, 72*(1), 49-69. doi: 10.1111/j.1467-9868.2009.00725.x
- Koenker, R. and Bassett, G. (1978). Regression quantiles. *Econometrica, 46*(1), 33-50. doi: https://www.jstor.org/stable/1913643
- Kownatzki, C. (2016). How good is the VIX as a predictor of market risk? *Journal of Accounting and Finance, 16*(6), 39-60.
- Leung, D. (2005). Cross-validation in nonparametric regression with outliers. *The Annals of Statisitics, 33*(5), 2291-2310. Retrieved from https://www.jstor.org/stable/3448642
- Li, Q. and Racine, J. (2004). Cross-validated local linear nonparametric regression. *Statistica Sinica, 14*(2), 485-512. doi: https://www.jstor.org/stable/24307205
- Luong, C. and Dokuchaev, N. (2018). Forecasting of realised volatility with the random forests algorithm. *Journal of Risk and Financal Management, 11*(4), 1-15. doi: 10.3390/jrfm11040061
- Mallows, C. L. (1973). Some comments on Cp. *Technometrics, 15*(4), 661-675. doi: https://w-ww.jstor.org/stable/i254276
- Miskolczi, P. (2017). Note on simple and logarithmic return. *Applied Studies in Agribusiness and Congress, 11*(2), 127-136. doi: 10.19041/APSTRACT/2017/1-2/16
- Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications, 9*(1), 141-142. doi: https://doi.org/10.1137/1109020
- Nelson, D. B. (1991). Conditional heteroskedasticity in asset returns: a new approach. *Econometrica, 59*(2), 347-370. doi: https://www.jstor.org/stable/2938260
- Reinsch, C. H. (1967). Smoothing by spline functions. *Numerische Mathematik, 10*(3), 177-183. doi: https://doi.org/10.1007/BF02162161
- Ruppert, D., Sheather, S. J., and Wand, M. P. (1995). An effective bandwidth selector for local least squares regression. *Journal of the American Statistical Association, 90*(432), 1257-1270. doi: 10.2307/2291516
- Ruppert, D., and Wand, M.P (1994). Multivariate locally weighted least squares regression. *Annals of Statistics, 22*(3), 1346-1370. doi: 10.1214/aos/1176325632

- Schindler, Anja. (2011). Bandwidth selection in nonparametric kernel estimation (Unpublished doctoral dissertation). Georg-August-Universität Göttingen. Annaberg-Buchholz, Germany.

- Schoenberg, I, J (1964). Spline functions and the problem of graduation. *Proceedings of the National Academy of Sciences, 52*(4), 947-950. doi: 10.1073/pnas.52.4.947

- Schucany, W. (2004). Kernel smoothers: an overview of curve estimators for the first graduate course in nonparametric statistics. *Statistical Science, 19*(4), 663-675. Retrieved from https://projecteuclid.org/euclid.ss/1113832731

- Scott, D.W. (1992). *Multivariate density estimation: theory, practice, and visualization.* New York, New York: John Wiley & Sons, Inc.

- Silverman, B, W. (1985). Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of Royal Statistical Society. Series B (Methodological), 47*(1), 1-52. Retrieved from https://www.jstor.org/stable/2345542

- Stone, C.J (1977). Consistent nonparametric regression. *The Annals of Statistics, 5*(4), 595-620. doi: 10.1214/aos/1176343886

- Stone, C.J (1980). Optimal rates of convergence for nonparametric estimators. *The Annals of Statistics, 8*(6), 1348-1360. Retrieved from https://www.jstor.org/stable/2240947

- Stone, C.J (1982). Optimal global rates of convergence for nonparametric regression. *The Annals of Statistics, 10*(4), 1040-1053. doi: 10.1214/aos/1176345969

- Stone, M. (1974). Cross-validatory choice and assessment of statistical preditions. *Journal of the Royal Statistical Society. Series B (Methodological), 36*(2), 111-147. Retrieved from https://www.jst- or.org/stable/2984809

- Teräsvirta, Timo. (2006). An introduction to univariate GARCH Models. *SSE/EFI Working Papers in Economics and Finance, 646.*

- Tsay, Ruey S. (2010). *Analysis of financial time series.* Hoboken, New Jersey: John Wiley & Sons, Inc.

- Wahba, G., Wold, S. (1975). A completely automatic french curve: fitting spline functions by cross validation. *Communications in Statistics, 4*(1), 1-17. doi: https://doi.org/10.1080/0361-0927508827223

- Wahba, G. (1990) Spline models for observational data. *CBMS-NSF Regional Conference Series in Applied Mathematics.* doi: https://doi.org/10.1137/1.9781611970128.
- Wand, M. P and Jones, M. C. (1994). *Kernel Smoothing.* Boca Raton, Florida: Chapman & Hall. CRC Monographs on Statistics and Applied Probability.
- Wang, L., Feng, C., Song, Q. and Yang, L. (2012). Efficient semiparametric GARCH modeling of financial volatility. *Statistica Sinica, 22*, 249-270. doi: http://dx.doi.org/1- 0.5705/ss.2009.-285
- Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A, 26*(4), 359-372. Retrieved from https://www.jstor.org/stable/25049340
- Wu, C. O. and Tian, X. (2013a). Nonparametric estimation of conditional distribution functions and rank-tracking probabilities with longitudinal data. *Journal of Statistical Theory and Practice, 7*(2), 1-26.
- Wu, C. O. and Tian, X. (2013b). Nonparametric estimation of conditional distribution functions and rank-tracking probabilities with time-varying transformation models in longitudinal studies. *Journal of the American Statistical Association, 108*(503), 971-982.
- Xia, Y. and Li, W. K. (2002). Asymptotic behavior of bandwidth selected by the cross-validation method for local polynomial fitting. *Journal of Multivariate Analysis, 83*(2), 265-287. doi: https://doi.org/10.1006/jmva.2001.2048
- Xu, K. L. and Phillips, P. (2011). Tilted nonparametric estimation of volatility functions with empirical applications. *Journal of Business & Economic Statistics, 29*(4), 518-528. doi: https://doi.org/10.1198/jbes.2011.09012
- Yu, K. and Jones, M. C. (2004). Likelihood-based local linear estimation of the conditional variance function. *Journal of the American Statistical Association, 99*(465), 139-144. doi: https://doi.org/10.1198/016214504000000133
- Zou, H., and Yuan, M. (2008). Composite quantile regression and the oracle model selection theory. *The Annals of Statistics, 36*(3), 1108-1126. doi: 10.1214/07-AOS507

Appendix A

Supplementary Tables

Table A.1: MSE Ratio in TVVP Simulation

Degree of Polynomial	Method	MSE Ratio	Best Estimator
Degree 0	Two-Step	1	13
	One-Step	1.027	5
	One-Step S0	1.025	5
	One-Step S1	1.023	4
	One-Step S2	1.012	16
	One-Step S3	1.088	7
Degree 1	Two-Step	1	13
	One-Step	1.026	2
	One-Step S0	1.002	9
	One-Step S1	1.037	7
	One-Step S2	1.021	6
	One-Step S3	1.089	13
Degree 2	Two-Step	1	16
	One-Step	1.018	3
	One-Step S0	0.983	1
	One-Step S1	0.971	8
	One-Step S2	0.986	12
	One-Step S3	1.011	10
Degree 3	Two-Step	1	24
	One-Step	1.034	2
	One-Step S0	1.025	6
	One-Step S1	1.027	1
	One-Step S2	1.027	5
	One-Step S3	1.026	12

Table A.2: Comparison of Methods - TVVP MSE Simulation

Degree of Polynomial	Method	Best Estimator
Degree 0	Two-Step	26
	One-Step S0	24
Degree 1	Two-Step	24
	One-Step S0	26
Degree 2	Two-Step	26
	One-Step S0	24
Degree 3	Two-Step	29
	One-Step S0	21

Table A.3: MSE Ratio in TVFP Simulation

Degree of Polynomial (μ)	Method (Degree of σ)	MSE Ratio	Best Estimator
Degree 0	Two-Step	1	48
	One-Step	1.169	0
	One-Step S0	1.166	0
	One-Step S1	1.167	0
	One-Step S2	1.170	1
	One-Step S3	1.170	1
Degree 1	Two-Step	1	44
	One-Step	1.197	2
	One-Step S0	1.191	1
	One-Step S1	1.190	0
	One-Step S2	1.191	3
	One-Step S3	1.193	0
Degree 2	Two-Step	1	49
	One-Step	1.423	1
	One-Step S0	1.413	0
	One-Step S1	1.412	0
	One-Step S2	1.409	0
	One-Step S3	1.411	0
Degree 3	Two-Step	1	49
	One-Step	1.438	1
	One-Step S0	1.430	0
	One-Step S1	1.430	0
	One-Step S2	1.427	0
	One-Step S3	1.429	0

Table A.4: MSE and MAD Results with Gaussian Kernel

Model	Metric	Train/Forecast	Method	Best (of 990)	Error for Metric
Lagged (1) Series	MSE	Training	LLS	826	0.00151
			CQR	164	0.00155
		Forecasting	LLS	299	0.00047
			CQR	691	0.00056
	MAD	Training	LLS	299	0.01405
			CQR	691	0.01322
		Forecasting	LLS	122	0.01007
			CQR	868	0.00722
De-meaned Series	MSE	Training	LLS	789	0.00148
			CQR	201	0.00152
		Forecasting	LLS	184	0.00420
			CQR	806	0.00587
	MAD	Training	LLS	171	0.01426
			CQR	819	0.01304
		Forecasting	LLS	59	0.01314
			CQR	931	0.00993
S&P Series	MSE	Training	LLS	688	0.00149
			CQR	302	0.00153
		Forecasting	LLS	492	0.00063
			CQR	498	0.00055
	MAD	Training	LLS	420	0.01379
			CQR	570	0.01332
		Forecasting	LLS	254	0.00943
			CQR	736	0.00784
VIX Series	MSE	Training	LLS	715	0.00151
			CQR	275	0.00155
		Forecasting	LLS	446	0.00103
			CQR	554	0.00137
	MAD	Training	LLS	475	0.01376
			CQR	515	0.01345
		Forecasting	LLS	306	0.00965
			CQR	684	0.00883

Appendix B

Select Code

```r
#------------------------------------------------
# Kernel Smoothers Bootstrap Code
#------------------------------------------------
ksboot <- function(x,y,nreps=1000, b9, confidence = 0.95){

  # Put input data into a data frame, sorted by x, with no missing
      values.
  dat <- na.omit(data.frame(x=x,y=y))
  if(nrow(dat) == 0) {
    print("Error: No data left after dropping NAs")
    print(dat)
    return(NULL)
  }
  ndx <- order(dat$x)
  dat$x <- dat$x[ndx]
  dat$y <- dat$y[ndx]
  # Fit curve to data
  require(KernSmooth)
  len <- length(dat$x)
  f0 <- ksmooth(x, y, kernel = "normal", bandwidth = b9, n.points =
      len)
  y.fit <- f0$y
  # Generate bootstrap replicates
  mat <- matrix(0,NROW(dat), nreps)
  for(i in seq(nreps)){
    ndx <- sample(len,replace=T)
    x.repl <- x[ndx]
    y.repl <- y[ndx]
    f <- ksmooth(x.repl, y.repl, kernel = "normal", bandwidth = b9,
        n.points = len)
    mat[, i] <- f$y
  }
  # calculating confidence intervals
  ci <- t(apply(mat, 1, quantile, probs = c((1-confidence)/2, (1+
      confidence)/2),na.rm = TRUE))
  res <- cbind(as.data.frame(f0), ci)
  colnames(res) <- c('x','y', 'lwr.limit','upr.limit')
  res
}

######################
###Kernel smoothing###
######################
###Kernel Smoothing (calls ks.opt.bw below)####
kernel.smooth <- function(l){
  k <- ncol(l)
  res <- sapply(1:k, function(i){
    d1 <- l[,i]
    op.band1 <- ksm.opt.bw(years,d1,1,20,0.1)
    sm1 <- ksmooth(years,d1,kernel="normal",bandwidth=op.band1[1],x.
        points=1:t)$y
    c(sm1)
  })
  list(res)
}

###Selecting optimal bandwidth with cross validation (calls ksm.lscv
    )####
ksm.opt.bw <- function(x,y,min,max,by){
  step <- (max - min) / by
  MSE <- rep(NA,step)
  for(i in 0:step){
    MSE[i+1] <- ksm.lscv(x,y,min + by*i)
  }
  return(c(min + by*which.min(MSE), min(na.omit(MSE))))
}

###Optimizing kernel smoothing parameter (cross validation function)
    ####
ksm.lscv <- function(x,y,h){
  n <- length(x)
  SE <- rep(NA,n)
  for(i in 1:n){
    y.smt <- ksmooth(x[-i],y[-i], kernel = 'normal', bandwidth = h,
        x.points = x[i])$y
    SE[i] <- (y[i] - y.smt)^2
  }
  return(mean(SE))
}
```

Listing B.1: Code for Kernel Smoothing Cross Validation and Bootstrapping

```
#-----------------------------------------------
# Local Polynomial Smoothers Bootstrap Code
#-----------------------------------------------
lpsboot <- function(x,y,nreps=1000, b9, confidence = 0.95){
  # Put input data into a data frame, sorted by x, with no missing
      values.
  dat <- na.omit(data.frame(x=x,y=y))
  if(nrow(dat) == 0) {
    print("Error:l No data left after dropping NAs")
    print(dat)
    return(NULL)
  }
  ndx <- order(dat$x)
  dat$x <- dat$x[ndx]
  dat$y <- dat$y[ndx]
  # Fit curve to data
  require(KernSmooth)
  len <- length(dat$x)
  f0 <- locpoly(x, y, kernel = "epanechnikov", bandwidth = b9,
      gridsize = len)
  y.fit <- f0$y
  # Generate bootstrap replicates
  mat <- matrix(0,NROW(dat), nreps)
  for(i in seq(nreps)){
    ndx <- sample(len,replace=T)
    x.repl <- x[ndx]
    y.repl <- y[ndx]
    f <- locpoly(x.repl, y.repl, kernel = "normal", bandwidth = b9,
        gridsize = len)
    mat[, i] <- f$y
  }
  # calculating confidence intervals
  ci <- t(apply(mat, 1, quantile, probs = c((1-confidence)/2, (1+
      confidence)/2),na.rm = TRUE))
  res <- cbind(as.data.frame(f0), ci)
  colnames(res) <- c('x','y', 'lwr.limit','upr.limit')
  res
}

######################
#### LP smoothing ####
######################
###Local Polynomial Smoothing (calls lps.opt.bw below)####
lp.smooth <- function(l){
  k <- ncol(l)
  res <- sapply(1:k, function(i){
    d1 <- l[,i]
    opt.b1 <- lps.opt.bw(years,d1,1,20,0.1)
    sm1<-locpoly(years,d1,kernel="normal",bandwidth=opt.b1[1],
        gridsize=t)$y
    c(sm1)
  })
  list(res)
}

###Selecting optimal bandwidth with cross validation (calls lps.lscv
    below)####
lps.opt.bw <- function(x,y,min,max,by){
  step <- (max - min) / by
  MSE <- rep(NA,step)
  for(i in 0:step){
    MSE[i+1] <- lps.lscv(x,y,min + by*i)
  }
  return(c(min + by*which.min(MSE), min(MSE)))
}

###Optimizing local polynormial parameter (cross validation function
  )####
lps.lscv <- function(x,y,h){
  n <- length(x)
  SE <- rep(NA,n)
  for(i in 1:n){
    lps <- npregress(x[-i],y[-i], bandwidth = h)
    SE[i] <- (y[i] - predict.npregress(lps,x[i]))^2
  }
  return(mean(SE))
}
```

Listing B.2: Code for Local Polynomial Smoothing Cross Validation and Bootstrapping

```
#------------------------------------------
# Spline Smoothing Bootstrap Code
#------------------------------------------
splineboot<-function(x,y,nreps=1000,confidence=0.95,spar){
  dat<-na.omit(data.frame(x=x,y=y))
  # original data
  f0<- smooth.spline(x=dat$x, y = dat$y,spar=spar)
  len<-length(dat$x)
  # store replicates
  mat <- matrix(0,NROW(dat), nreps)
  new_x<-seq(min(x),max(x), length.out = NROW(dat))
  y.fit<-predict(f0,new_x)
  # bootstraps
  for (i in 1:nreps){
    ndx<-sample(len,replace=T)
    x.repl<-x[ndx]
    y.repl<-y[ndx]
    f<-smooth.spline(x=x.repl, y = y.repl,spar=spar)
    mat[,i]<-predict(f,new_x)$y
  }
  ##confidence interval
  ci <- t(apply(mat, 1, quantile, probs = c((1-confidence)/2, (1+
      confidence)/2),na.rm = TRUE))
  res <- cbind(as.data.frame(y.fit), ci)
  colnames(res) <- c('x','y', 'lwr.limit','upr.limit')
  res
}

########################
####Spline Smoothing####
########################
###Spline Smoothing (calls smooth.opt.bw below)####
spline.smooth <- function(l){
  k <- ncol(l)
  res <- sapply(1:k, function(i){
    d1 <- l[,i]
    opt.b1 <- smooth.opt.bw(years,d1,0.1,1.5,0.1)
    sm1 <- smooth.spline(years,d1,spar = opt.b1[1])$y
    c(sm1)
  })
  list(res)
}

###Selecting optimal bandwidth with cross validation (calls smooth.
    cv)####
smooth.opt.bw <- function(x,y,min,max,by){
  step <- (max - min) / by
  MSE <- rep(NA,step)
  for(i in 0:step){
    MSE[i+1] <- smooth.cv(x,y,min + by*i)
  }
  return(c(min + by*which.min(MSE), min(MSE)))
}

###Optimizing smooth spline parameter (cross validation)####
smooth.cv<- function(x,y,h){
  n<- length(x)
  SE <- rep(NA,n)
  for(i in 1:n){
    spline <- smooth.spline(x[-i],y[-i],spar=h)
    SE[i] <- (y[i] - predict(spline,x[i])$y)^2
  }
  return(mean(SE))
}
```

Listing B.3: Code for Spline Smoothing Cross Validation and Bootstrapping

```
#--------------------------------------------
# Quantile Regression (QR) Bootstrap Code
#--------------------------------------------
qrboot <- function(x,y,tlbflag=FALSE,tubflag=FALSE,tlb=0.05,tub
    =0.95,nreps=1000,confidence = 0.95){
  require(quantreg)
  # Put input data into a data frame, sorted by x, with no missing
      values.
  dat <- na.omit(data.frame(x=x,y=y))

  if(nrow(dat) == 0) {
    print("Error: No data left after dropping NAs")
    print(dat)
    return(NULL)
  }

  if(tlbflag == TRUE){
    ndx <- order(dat$x)
    dat$x <- dat$x[ndx]
    dat$y <- dat$y[ndx]
    # Fit curve to data
    len <- length(dat$x)
    currdata=data.frame(x=x,y=y)
    f0 <- rq(y ~ x, data=currdata, tau = tlb)
    y.fit=predict(f0,newdata=currdata)
  }

  if(tubflag == TRUE){
    ndx <- order(dat$x)
    dat$x <- dat$x[ndx]
    dat$y <- dat$y[ndx]
    # Fit curve to data
    len <- length(dat$x)
    currdata=data.frame(x=x,y=y)
    f0 <- rq(y ~ x, data=currdata, tau = tub)
    y.fit=predict(f0,newdata=currdata)
  }

  # Generate bootstrap replicates
  if(tlbflag == TRUE){
    mat <- matrix(0,NROW(dat), nreps)
    for(i in seq(nreps)){
      ndx <- sample(len,replace=T)
      x.repl <- x[ndx]
      y.repl <- y[ndx]
      currnewdata=data.frame(x.repl=x.repl, y.repl=y.repl)
      f <- rq(y.repl~x.repl,data=currnewdata,tau=tlb)
      ycurr.fit=predict(f,newdata=currnewdata)
      mat[, i] <- ycurr.fit
    }}

  if(tubflag == TRUE){
    mat <- matrix(0,NROW(dat), nreps)
    for(i in seq(nreps)){
      ndx <- sample(len,replace=T)
      x.repl <- x[ndx]
      y.repl <- y[ndx]
      currnewdata=data.frame(x.repl=x.repl, y.repl=y.repl)
      f <- rq(y.repl~x.repl,data=currnewdata,tau=tub)
      ycurr.fit=predict(f,newdata=currnewdata)
      mat[, i] <- ycurr.fit
    }}
  # calculating confidence intervals
  ci <- t(apply(mat, 1, quantile, probs = c((1-confidence)/2, (1+
     confidence)/2),na.rm = TRUE))
  res <- cbind(x, y.fit, ci)
  colnames(res) <- c('x','y', 'lwr.limit','upr.limit') #y is fitted
     values
  res
}
```

Listing B.4: Code for Quantile Regression Bootstrapping

```
%%
% initializations of bandwidths to test and store results in "
    results"
bw = linspace(2.05,8,120);
results = ones(size(bw,2),2);
results(:,1) = bw;
%%
% for each bandwidth, perform leave one time period out cross-
    validation
for h = bw
    tot = 0;
    for tp = 1:length(d4(:,1))
        d4_tmp = d4(d4(:,1) > (tp-h) & d4(:,1) < (tp+h),:);
        d4_tmp(d4_tmp(:,1)==tp,:) = [];
        w = (3/(4*h))*(1-((d4_tmp(:,1)-tp)/h).^2);
        rows = size(d4_tmp);

        % set up constraints
        cons2 = d4_tmp(:,1)-tp;
        cons3 = (d4_tmp(:,1)-tp).^2;
        A = ones(rows(1),3);
        A(:,2) = cons2;
        A(:,3) = cons3;
        b = ones(rows(1),1);
        lb=[0 -10000 -10000];ub=[1 10000 10000];
        Aeq=[];beq=[];
        f = @(v)three_var(v,rows,w,d4_tmp,tp);
        [x, fval] = fmincon(f,[0.1 0 0],A,b,Aeq,beq,lb,ub);
        %disp(x(1))
        %disp(tp)
        tot = tot + (x(1)-y(tp))^2;
    end
    results(results(:,1)==h,2) = tot;
end
%%
% select best h from cross-validation
[m,ind] = min(results(:,2));
best_h = results(ind,1)
%%
% with optimal h, now score the data for plotting
pts = linspace(1,19,361);
pts = linspace(1,19,19);
smoothed = ones(size(pts,2),2);
smoothed(:,1) = pts;
h=best_h;
%%
% repeat smoothig process, without leave one out CV
for tp = pts
    d4_tmp = d4(d4(:,1) > (tp-h) & d4(:,1) < (tp+h),:);
    w = (3/(4*h))*(1-((d4_tmp(:,1)-tp)/h).^2);
    rows = size(d4_tmp);

    % set up constraints
    cons2 = d4_tmp(:,1)-tp;
    cons3 = (d4_tmp(:,1)-tp).^2;
    A = ones(rows(1),3);
    A(:,2) = cons2;
    A(:,3) = cons3;
    b = ones(rows(1),1);
    lb=[0 -10000 -10000];ub=[1 10000 10000];
    Aeq=[];beq=[];
    f = @(v)three_var(v,rows,w,d4_tmp,tp);
    [x, fval] = fmincon(f,[0.1 0 0],A,b,Aeq,beq,lb,ub);
    %[x, fval] = fmincon(f,[0.1 0 0]);
    smoothed(smoothed(:,1)==tp,2) = x(1);
end
```

Listing B.5: Code for Cross Validation and Smoothing for Degree 2 Polynomial Expansion in Geometric Kernel Log-Likelihood

```
function a = three_var(v,rows,w,d4_tmp,tp)
beta0 = v(1);
beta1 = v(2);
beta2 = v(3);
sum_logs=0;

for j = 1:rows(1)
    sum_logs = sum_logs + (d4_tmp(j,3)*log(beta0+(d4_tmp(j,1)-tp)*
        beta1+(d4_tmp(j,1)-tp)^2*beta2)+d4_tmp(j,2)*log(1-beta0-(d4_
        tmp(j,1)-tp)*beta1-(d4_tmp(j,1)-tp)^2*beta2))*w(j);
end
a = -sum_logs;
end
```

Listing B.6: Cross Validation Function for Degree 2 Polynomial Expansion in Geometric Kernel Log-Likelihood

```
function mus = lp_mus(p,h,df,sigmas)
len = size(sigmas,1);
mus = zeros(1,len);
for tp = sigmas(:,1).'
    lastwarn('');
    % only take data with +- h
    ind1 = df(:,1) > (tp-h) & df(:,1) < (tp+h);
    df_tmp = df(ind1,:);
    % remove current tp
    df_tmp(df_tmp(:,1)==tp,:) = [];
    % same thing for s_tmp
    ind2 = sigmas(:,1) > (tp-h) & sigmas(:,1) < (tp+h);
    s_tmp = sigmas(ind2,:);
    s_tmp(s_tmp(:,1)==tp,:) = [];
    wgts = repelem(s_tmp(:,2),s_tmp(:,3));

    weights = 3/(4*h)*(1-((df_tmp(:,1)-tp)/h).^2);
    k = diag(0.5*weights./wgts);

    X1 = ones(size(df_tmp,1),p+1);
    Y1 = df_tmp(:,2);
    if (p ~= 0)
        for i = 2:(p+1)
            X1(:,i) = (df_tmp(:,1)-tp).^(i-1);
        end
    end
    coef = inv(X1.'*k*X1)*X1.'*k*Y1;
    [warnMsg, warnId] = lastwarn;
    if ~isempty(warnMsg)
        coef = pinv(X1.'*k*X1)*X1.'*k*Y1;
    end
    mus(tp) = coef(1);
end
end
```

Listing B.7: Cross Validation Function for Mean Estimation in Normal Kernel Log-Likelihood

```
function a = lp_sigp1(s,h,tp,df,sigmas,mus)
s0 = s(1);
s1 = s(2);
ind1 = df(:,1) > (tp-h) & df(:,1) < (tp+h);
df_tmp = df(ind1,:);
% remove current tp
df_tmp(df_tmp(:,1)==tp,:) = [];
% same thing for s_tmp
ind2 = sigmas(:,1) > (tp-h) & sigmas(:,1) < (tp+h);
s_tmp = sigmas(ind2,:);
s_tmp(s_tmp(:,1)==tp,:) = [];
weights = s_tmp(:,1:2);
weights(:,2) = 3/(4*h)*(1-((s_tmp(:,1)-tp)/h).^2);
sum_logs = 0;
for tp2 = s_tmp(:,1).'
    % need vector of coefs (results from mu)
    w = weights(weights(:,1)==tp2,2);
    sum_logs = sum_logs - s_tmp(s_tmp(:,1)==tp2,3)/2*log(s0+(tp2-tp)
        *s1)*w - 1/(2*(s0+(tp2-tp)*s1))*sum((df_tmp(df_tmp(:,1)==tp2
        ,2)-mus(tp2)).^2)*w;
end
a = - sum_logs;
end
```

Listing B.8: Cross Validation Function for Variance Estimation Under Degree 1 Polynomial Expansion in Normal Kernel Log-Likelihood