PROJECT #3 for Applied Nonparametric Econometrics

Jia Ru

2016-4-18

1. Do Monte Carlo simulations to compare the performances of the local linear and local constant estimations

The nonparametric regression function is

$$Y_t = m(X_t) + u_t, \quad 1 \leq t \leq T$$

where X_t **is stationary**.Choose different sample sizes, different kernels, different bandwidths, and different bandwidth selection methods; Use MADE to measure the performance of estimation. Boxplot the distribution of MADE for different settings.

ANS:

Here I set DGP as: X_t is IID follows Uniform distributon on $[0, 3]$, $m(x) = e^x$, and u_t is IID follows Standard Normal Distribution. I refer to *4.7 Computer Code in Lecture note, page 88* to define functions in order to get the local-constant and local-linear estimator, with a few modifications, that is: 1. set some default arugments: grid points is exactly sample point x, and bandwidth is 0.02 by default, kernel is normal by default for convenience; 2. modify the local.linear function by directly return the fitted value, ie. Intercept instead of the n-by-2 matrix beta. I also define functions to generate data under different settings and to calculate MADE.

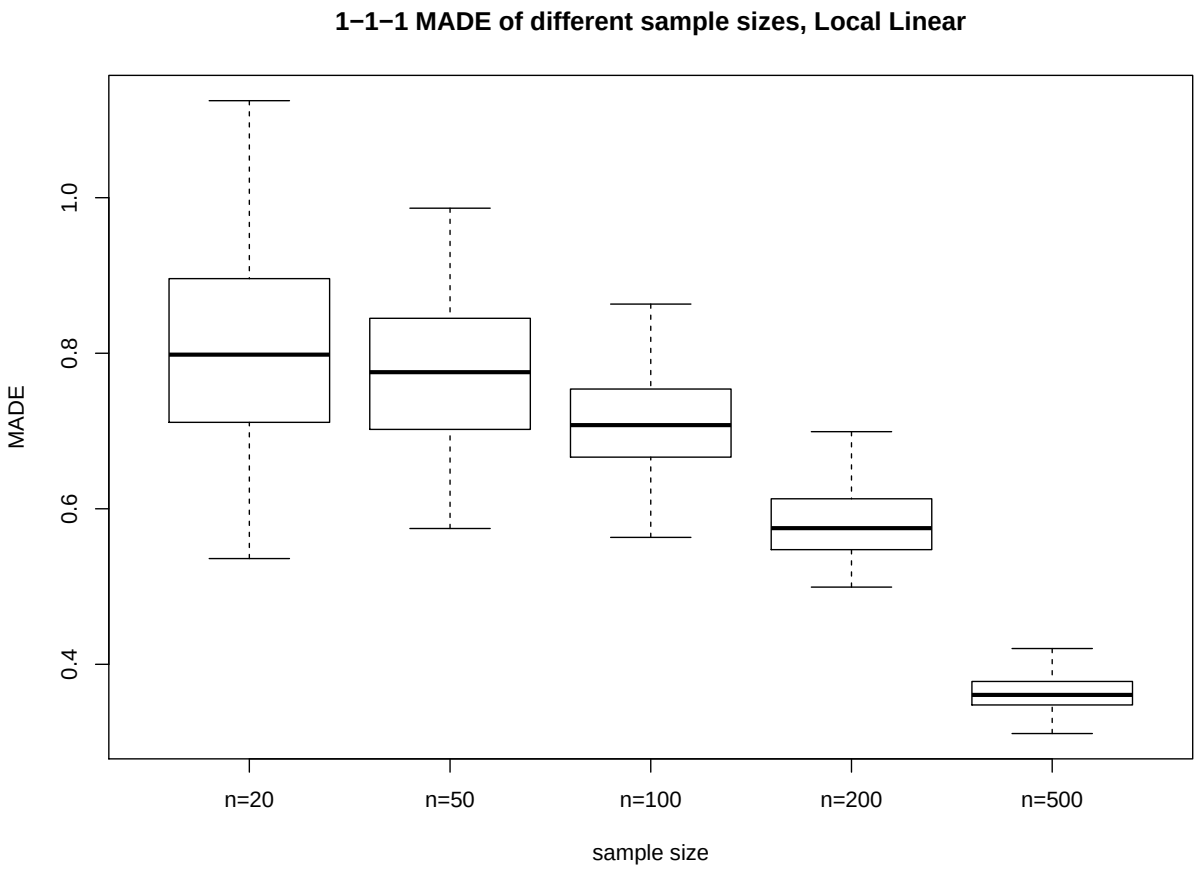
```
rm(list=ls())
EpanKer<-function(x){0.75*(1-x^2)*(abs(x)<=1)}

# ######################################################
# Define function of local constant estimator
# ######################################################

local.constant=function(y,x,z=x,h=0.02,ker="norm"){
  nz <- length(z)
  nx <- length(x)
  x0 = rep(1,nx*nz)
  dim(x0) = c(nx,nz)
  x1 = t(x0)
  x0 = x*x0
  x1 = z*x1
  x0 = x0-t(x1)
  if(ker=="Epan"){x1=EpanKer(x0/h)}
  if(ker=="norm"){x1=dnorm(x0/h)}
  x2 = y*x1
  f1 = apply(x1,2,mean)
  f2 = apply(x2,2,mean)
  f3 = f2/f1

  return(f3)  # return num[1:n]
}
```

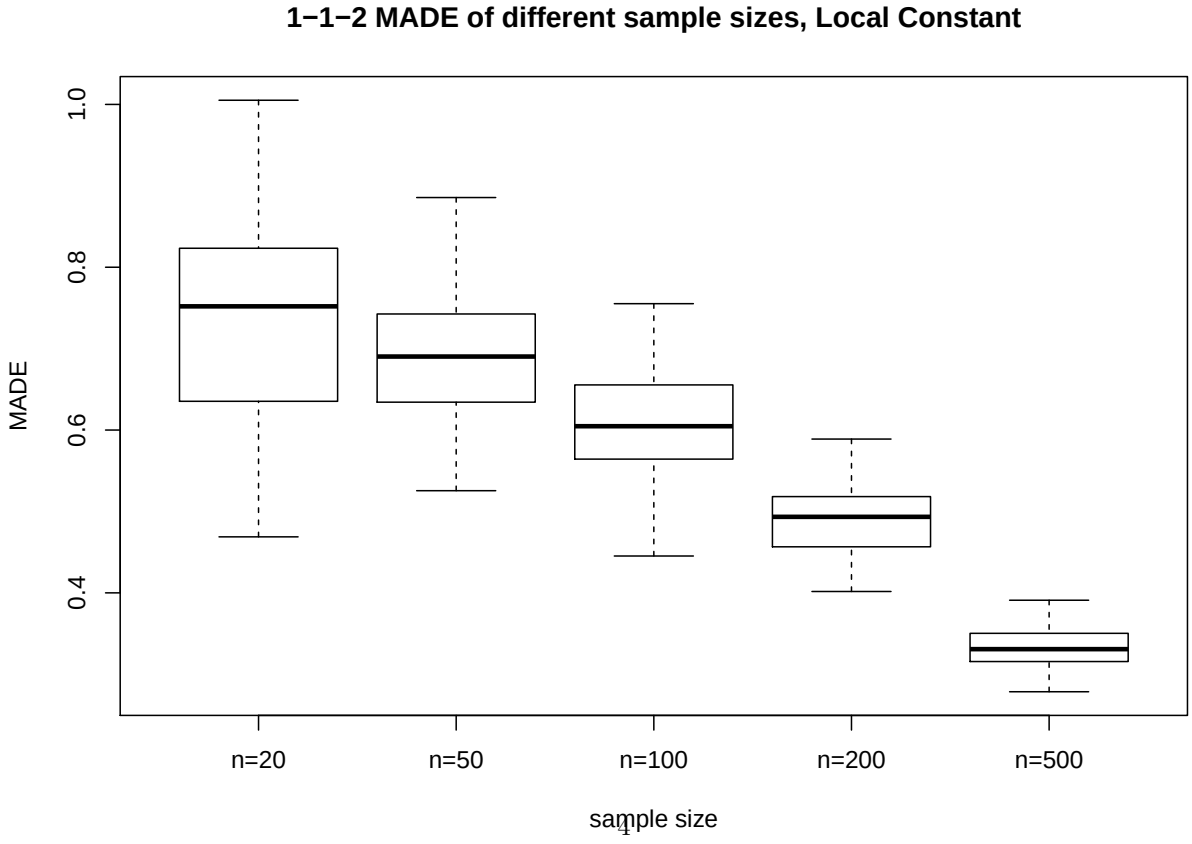
```r
# ######################################################
# Define the local linear estimator
# ######################################################

local.linear<-function(y,x,z=x,h=0.02,ker="norm"){
  nz <- length(z)
  ny <- length(y)
  beta <- rep(0,nz*2)
  dim(beta) <- c(nz,2)
  for(k in 1:nz){
    x0 = x-z[k]
    if(ker=="Epan"){w0 <- EpanKer(x0/h)}
    if(ker=="norm"){w0 <- dnorm(x0/h)}
    beta[k,] <- glm(y~x0,weight=w0)$coeff
  }
  return(beta[,1])
}

# ######################################################
# Define function DGP: Data Generating Process
# ######################################################

DGP <- function(n,stationary=TRUE){
  if (stationary==TRUE) {
    x <- runif(n, min=0, max=3)
    mx <- exp(x)
    y <- mx + rnorm(n)
  } else if (stationary==FALSE){
    x <- runif(1, min = 0, max=3)
    for (i in 2:n) {x[i] <- 0.1*x[i-1]+rnorm(1)}
    mx <- x^2
    y <- mx
  }
  return(rbind(x,y,mx))
}

# ######################################################
# Define function MADE, MADE_set
# ######################################################

# Generate One MADE
MADE <- function(n,ker,h=0.02,stationary=T,method) {
  sample <- DGP(n,stationary)
  y <- sample["y",]
  x <- sample["x",]
  m <- sample["mx",]
  m_hat <-
    if (method=="ll") {
      local.linear(y,x,ker=ker,h=h)
    } else if (method=="lc") {
      local.constant(y,x,ker=ker,h=h)
    }
  MADE <- 1/n*sum(abs(m-m_hat))
```

```
  return(MADE)
}

# Generate a set of n_sim MADEs
library(plyr)
MADE_set <- function(n_sim,
                     n,ker,h=0.02,stationary=T,method)
{
  MADE_set <- laply(
    .data = as.list(1:n_sim),
    .fun = function(foo) MADE(n,ker,h,stationary,method),
    .progress = "text"
  )
  return(MADE_set)
}

# test:
# MADE_set(n_sim=10,
#          n=20,ker="Epan",h=0.02,stationary = T,method="ll")
```

Now use `MADE_set()` function to do Monte Carlo Simulations: (Code hide)

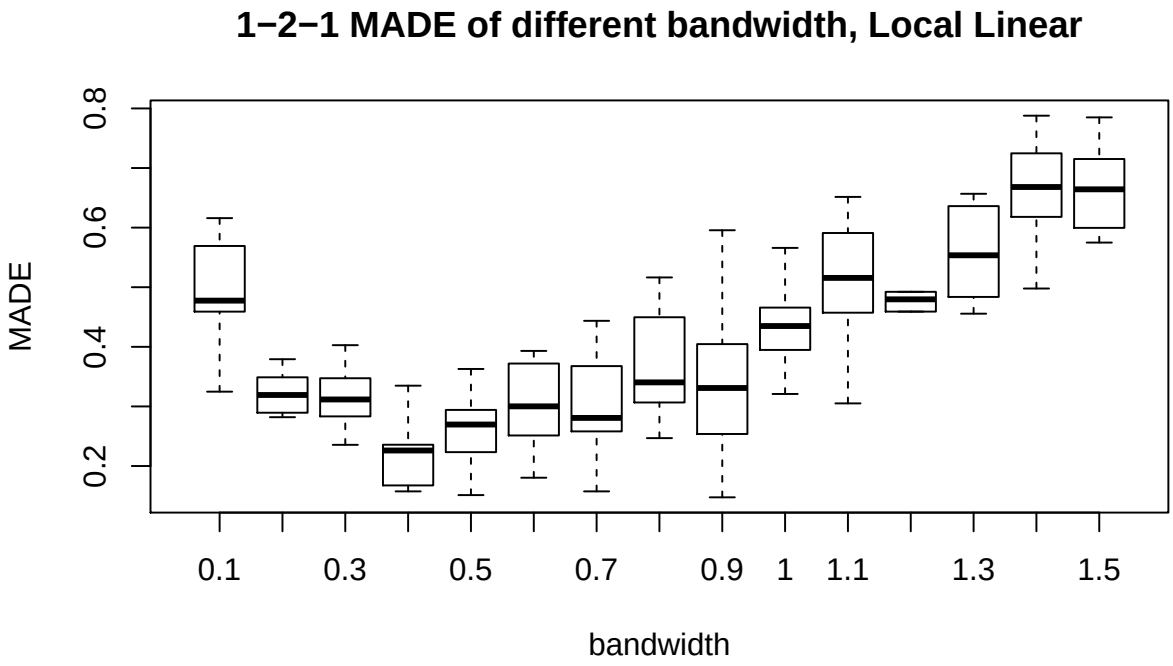
1-1 different sample sizes

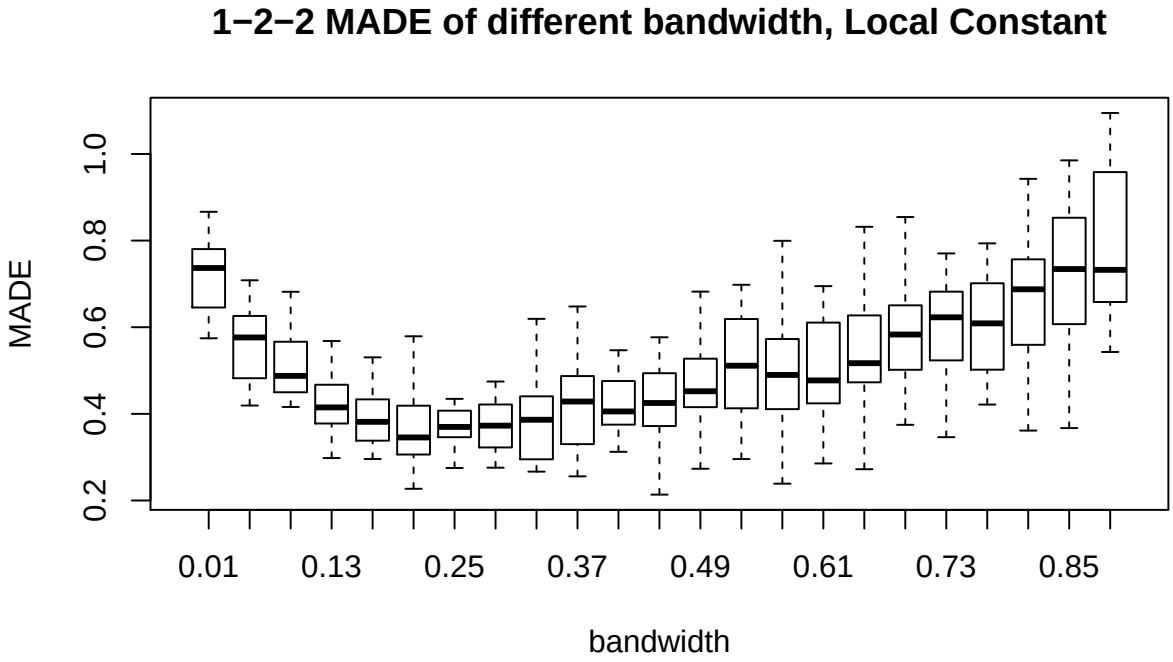

1−1−1 MADE of different sample sizes, Local Linear
MADE
0.4
0.6
0.8
1.0
n=20
n=50
n=100
n=200
n=500
sample size


1−1−2 MADE of different sample sizes, Local Constant
MADE
0.4
0.6
0.8
1.0
n=20
n=50
n=100
n=200
n=500
sample size

Here I try 5 different sample sizes: 20,50,100,200,500. From the figure we can see, as sample size increase, both mean and standard variance of MADE decrease, it is the same for local constant method and local linear method.

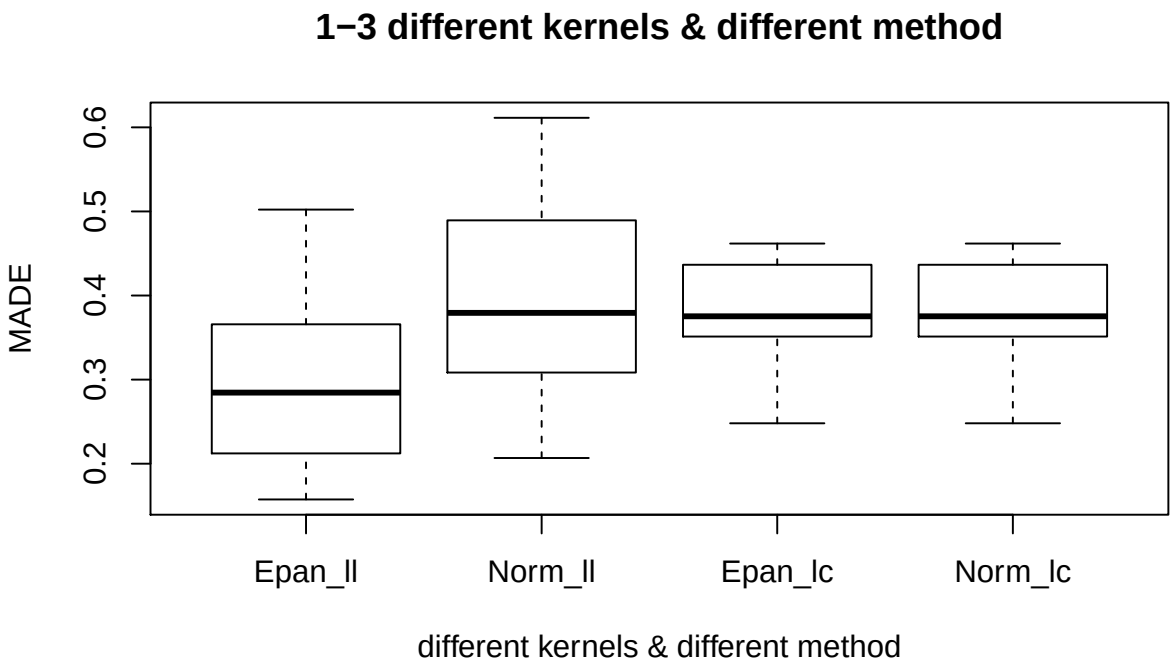
1-2 different bandwidths





from the two figure we can see, the best bandwidth for local constant method is approximately 0.2, the best bandwidth for local linear method is approximately 0.4.

1-3 different kernels



from the figure we can see, Epan kernel with local linear method performs best. But local constant method has smaller variance. Here I use different bandwidth for the two method which is "best" in terms of MADE, obtained in last question.

2. Re-consider Question 1, where X_t is nonstationary

All the operation is the same as in Question 1, except that the data generating process is:

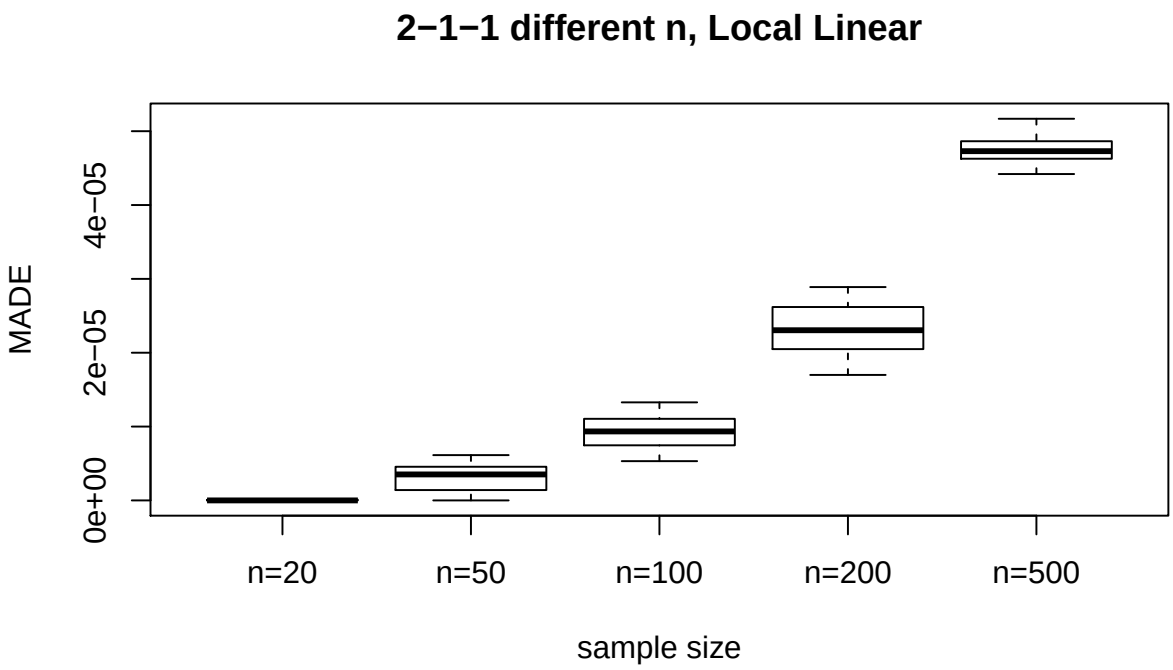
$$X_t = 0.1X_{t-1} + v_t$$

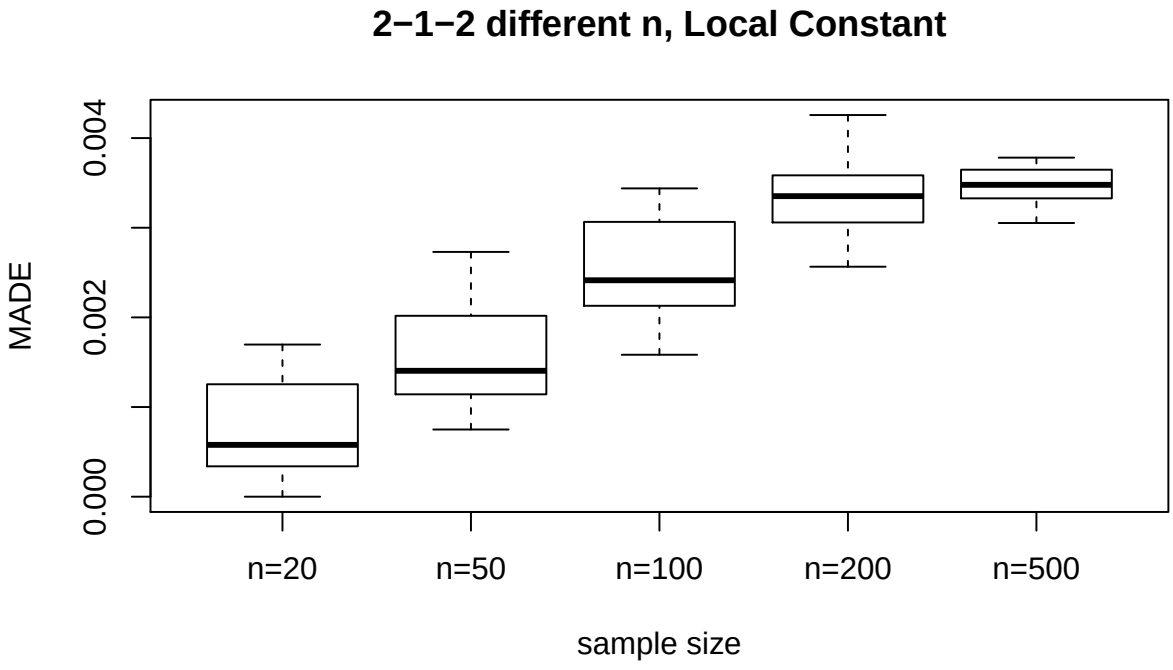
where v_t is iid standard normal. And Y_t is

$$Y_t = X_t^2 + u_t$$

(Codes hide)

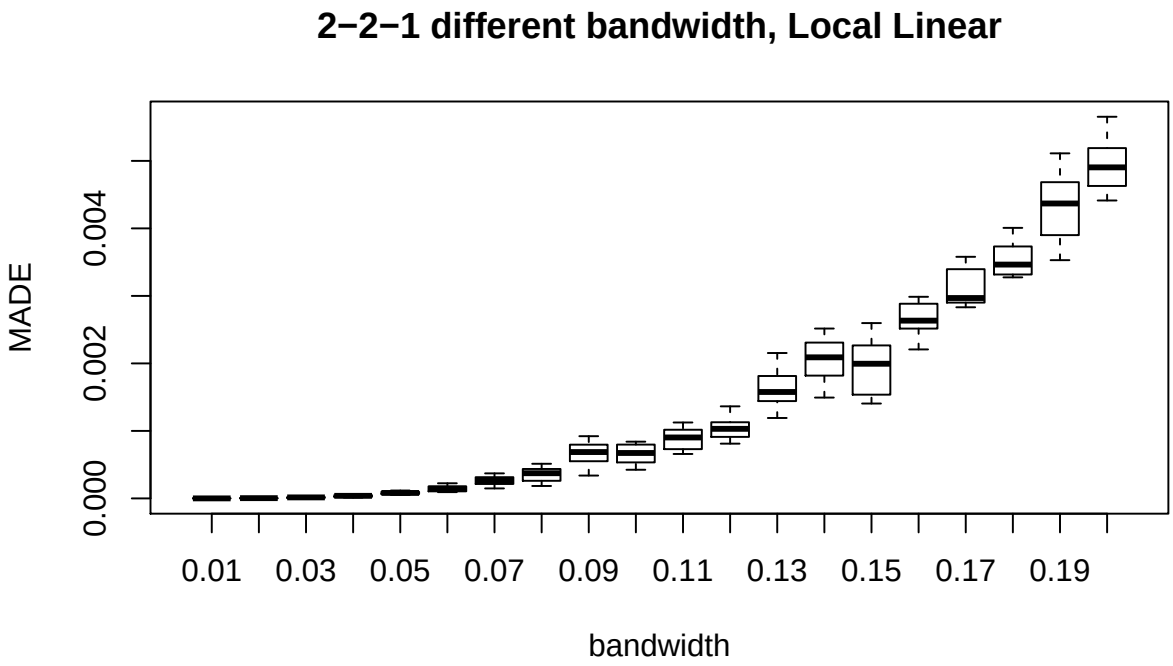
2-1 different sample sizes

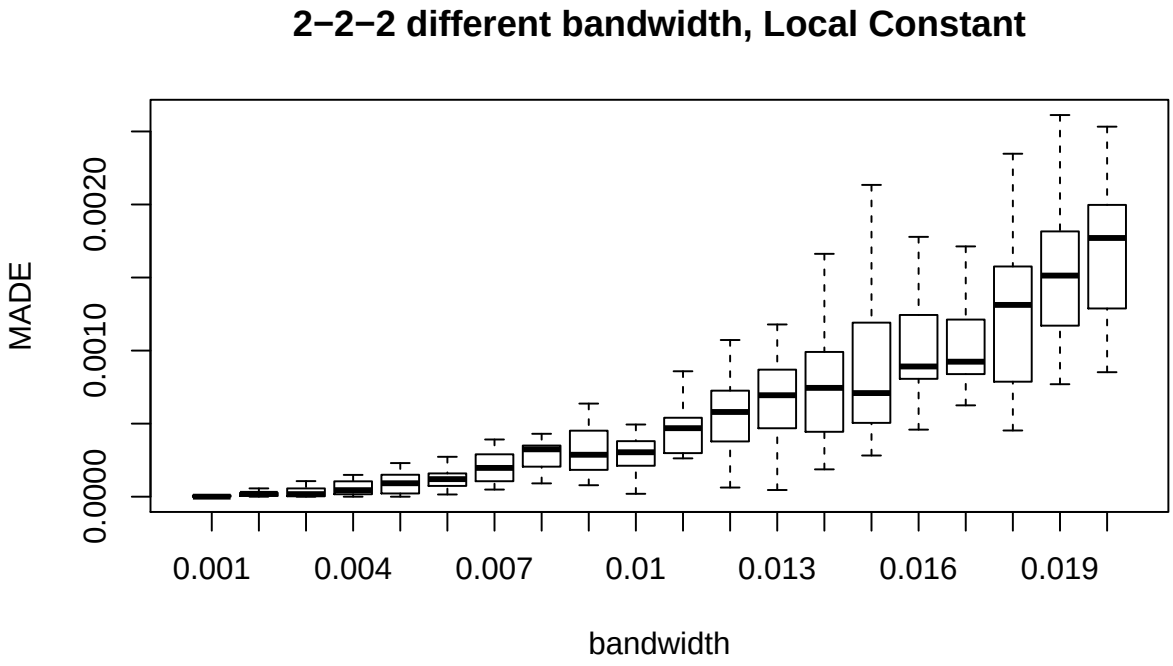




It is totally different from the stationary case that as sample size increase, the MADE also increase. This is because of serial correlation of X_t. The local linear method seems to have smaller MADE for all sample sizes.

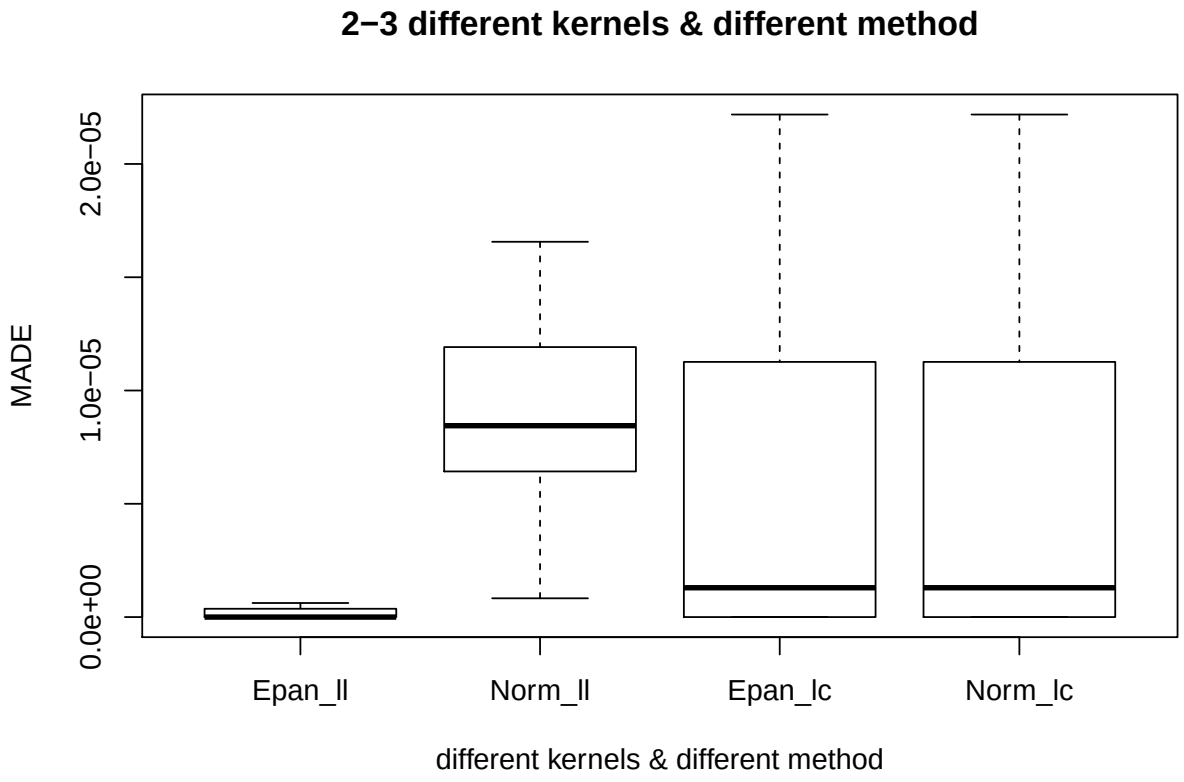
2-2 different bandwidths





from the two figure we can see, the optimal bandwidth seem to be as small as possible.(I have tried even smaller bandwidth but it's almost the same pattern).

2-3 different kernels

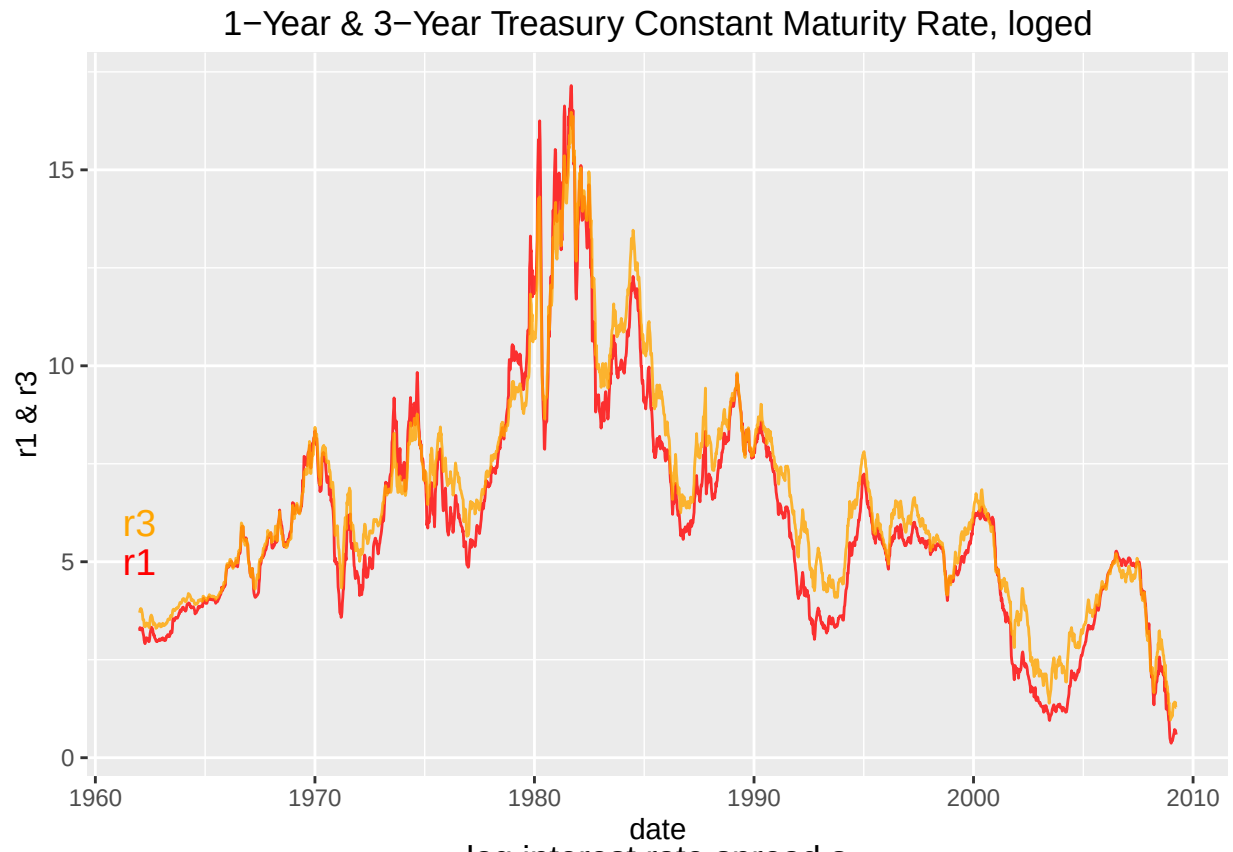


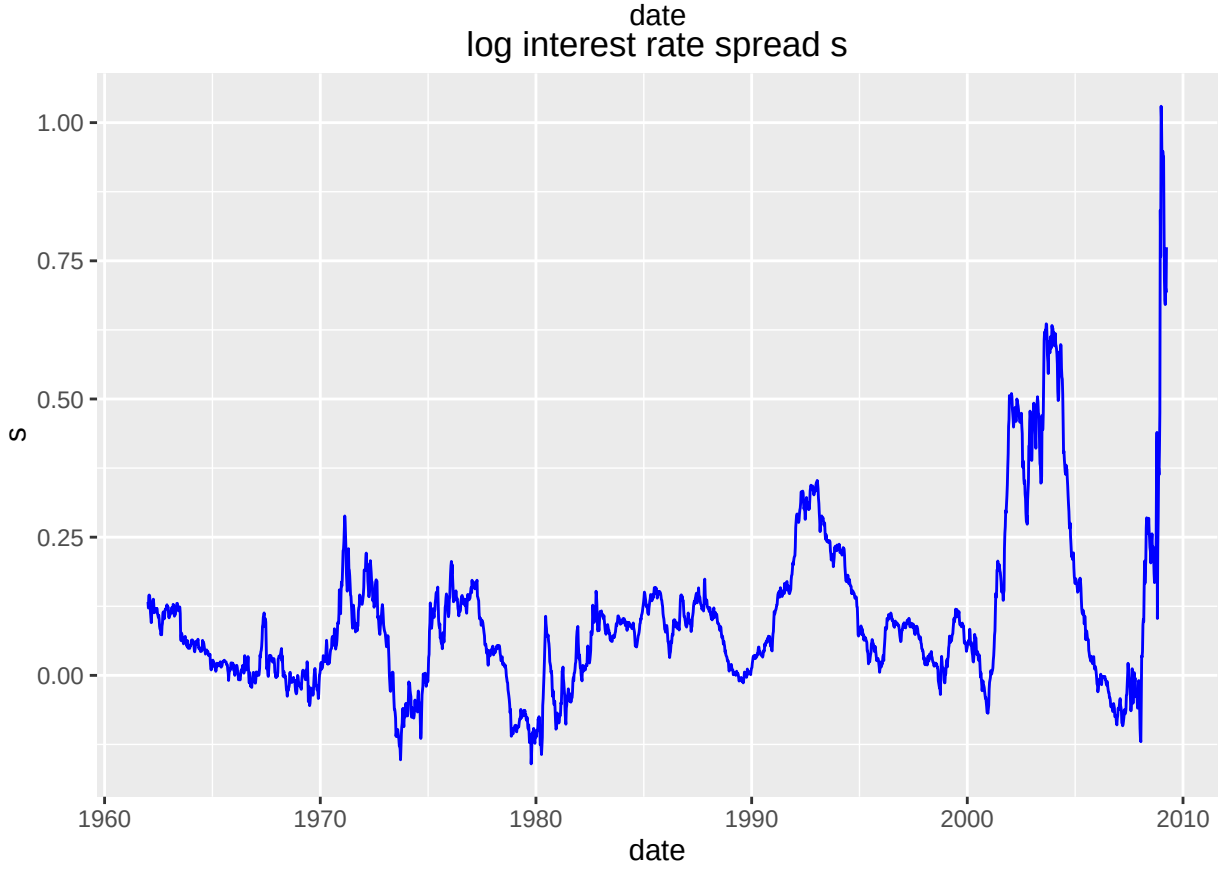
from the figure we can see, Epan kernel with local linear method has smallest MADE.

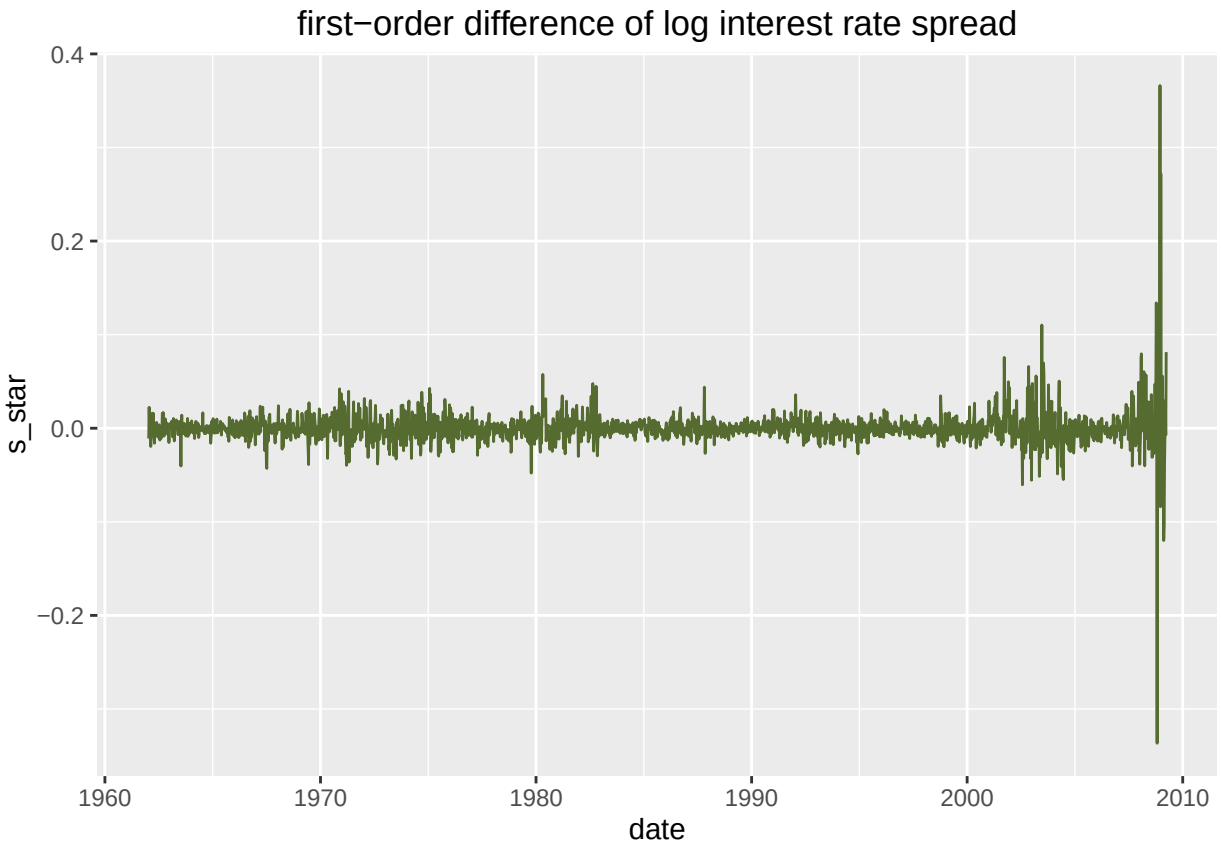
3. Empirical Analysis: The nonlinearity of spread of interest rates

I download 1-Year Treasury Constant Maturity Rate from https://research.stlouisfed.org/fred2/series/WGS1YR# and 3-Year Treasury Constant Maturity Rate from https://research.stlouisfed.org/fred2/series/WGS3YR#.

First of all, take an overlook at the data. Plot r_{1t}, r_{3t}, s_t and $s_t^\star$. The data looks like this:


1−Year & 3−Year Treasury Constant Maturity Rate, loged
r1 & r3
15
10
5
0
r3
r1
1960
1970
1980
1990
2000
2010
date


log interest rate spread s
s
1.00
0.75
0.50
0.25
0.00
1960
1970
1980
1990
2000
2010
date

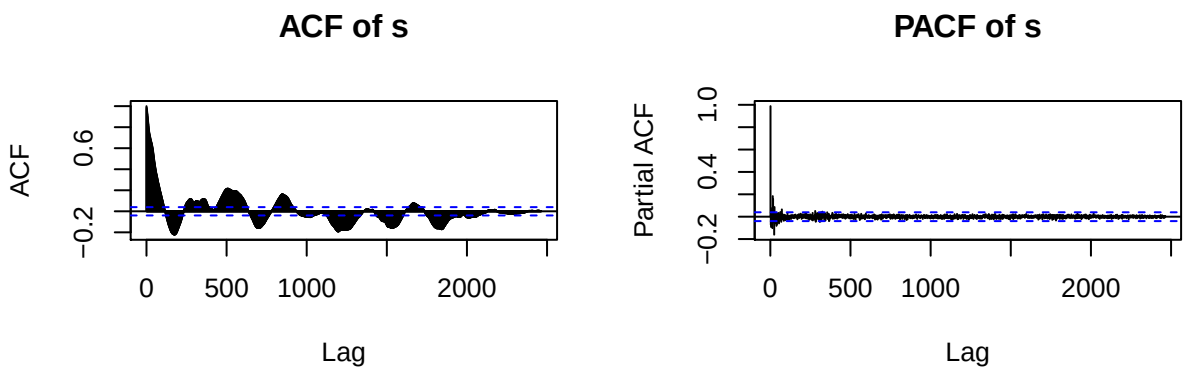


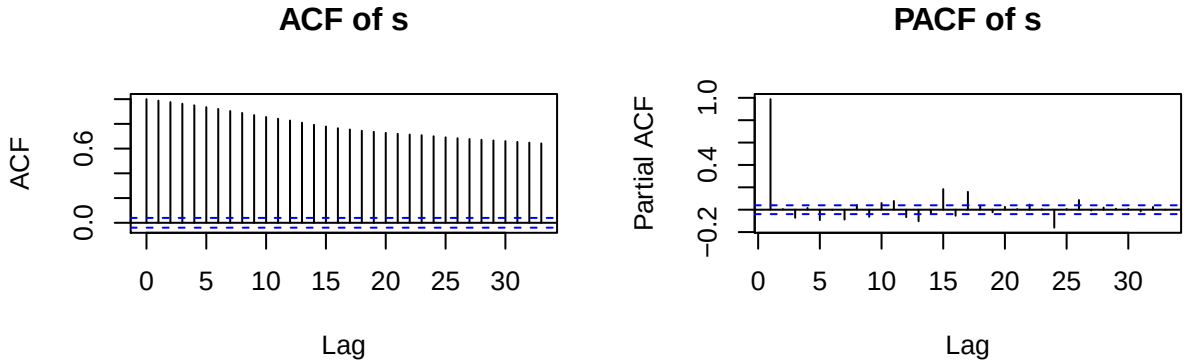
To study if $\{s_t\}$ is nonlinear, I try different nonparametric regression model and do analysis of variance to test whether local linear model is suitable. Since the $\{s_t\}$ is time series, I first see the ACF and PACF plot of the series. From the ACF plot we can see, the auto-correlation is significant positive and very large for all lag less than about 100 while the 1st-order lag is almost 0.9, so I construct the regression model using only 1st-order lag as explainatory variable:

$$s_t = m(s_{t-1}) + \epsilon_t$$

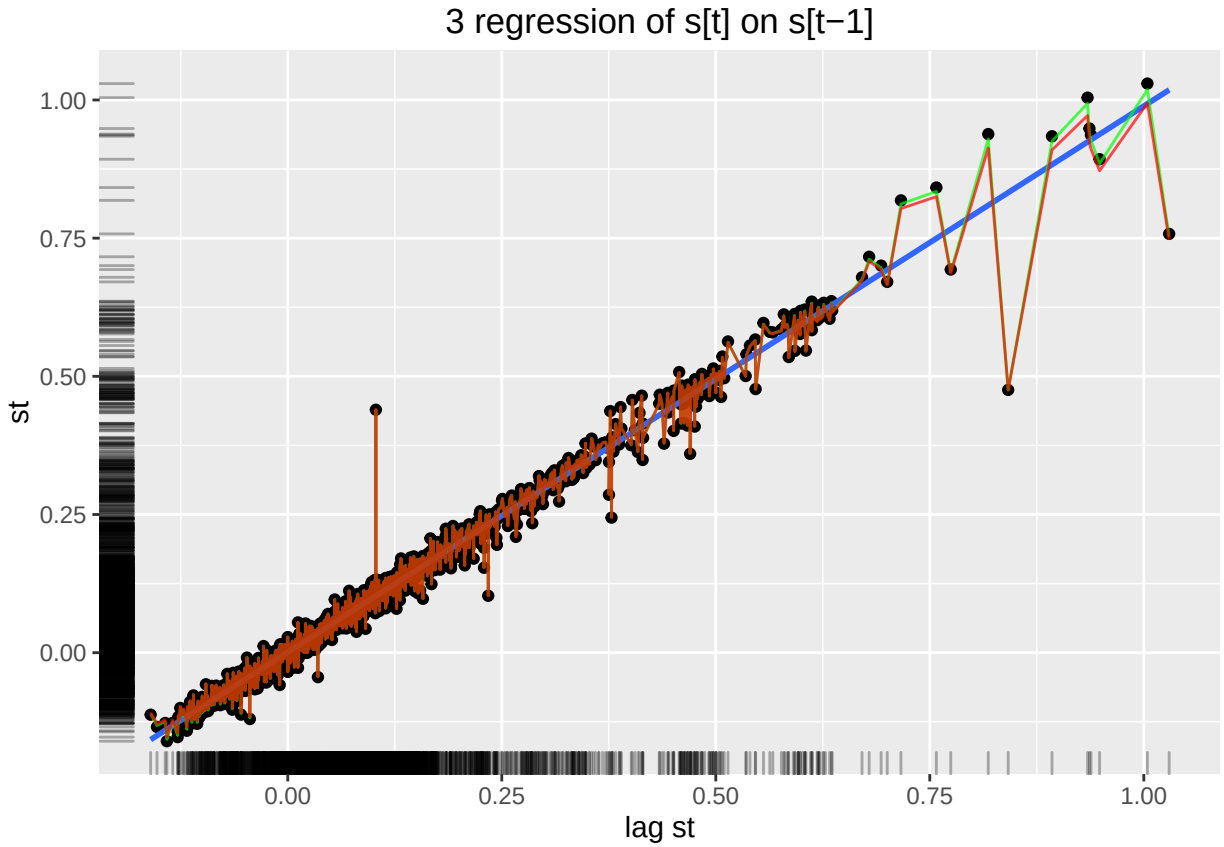
Here I use `dyn` and `loess` package in R to do the regression. Use `anova()` to test Hypothesis.

```
require(np)
require(loess)
require(locfit)
library(dyn)

#####################  1. s   ######################
st <- ts(WGS$s)

# ACF and PACF
par(mfrow=c(2,2))
acf(st, lag.max = nrow(WGS), main="ACF of s")
pacf(st, lag.max = nrow(WGS), main="PACF of s")
acf(st, main="ACF of s")
pacf(st, main="PACF of s")
```






```r
# noneparametric regression
np_l <- dyn$loess(st ~ lag(st, -1),degree=1)  # local linear
np_nl <- dyn$loess(st ~ lag(st, -1),degree=2)  # local polynomial
fitted_l <- fitted(np_l)
fitted_nl <- fitted(np_nl)

# visialization
ggplot(data=NULL, aes(x=st[-1],y=st[-length(st)])) +
  geom_rug(alpha=0.3) +
  geom_point() +
  geom_smooth(method="lm") +
  geom_line(aes(y=fitted_l), col="green",alpha=0.7) +
  geom_line(aes(y=fitted_nl), col="red", alpha=0.7) +
  xlab("lag st") +
  ylab("st") +
  ggtitle("3 regression of s[t] on s[t-1]")
```




```
# Hypothesis Testing
anova(np_l,np_nl)[-1]
```

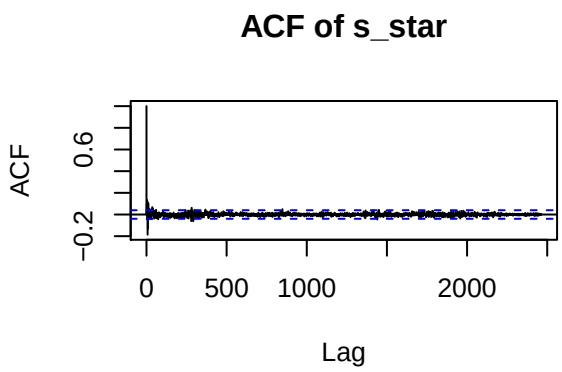
```
##          RSS F-value   Pr(>F)
## [1,] 0.81091
## [2,] 0.80700  6.7889 0.002323 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

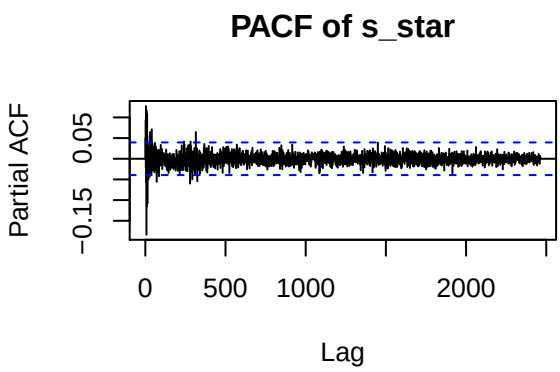
Results:

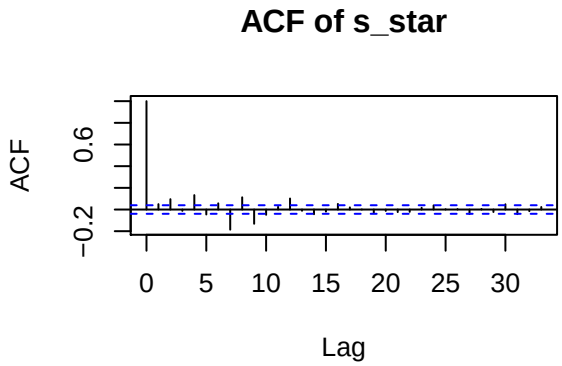
the linear and nonlinear approach gives almost the same result and very close to the sample point except some outliers. The Hypothesis Testing result show that we have to **reject** the null hypothesis that s_t is linear under the significance level 0.05.

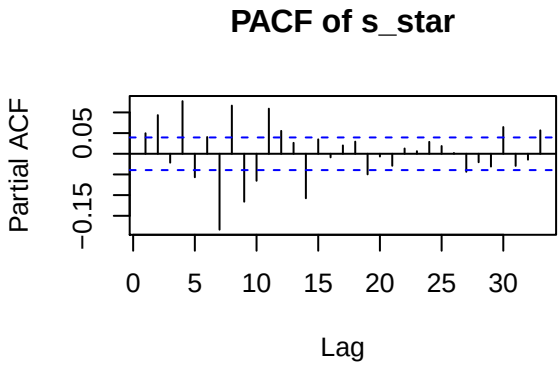
For series $s_t^\star$ do the same thing:

```
######################  2. s_star   ######################
st <- ts(WGS$s_star)[-1]

# ACF and PACF
par(mfrow=c(2,2))
acf(st, lag.max = nrow(WGS), main="ACF of s_star")
pacf(st, lag.max = nrow(WGS), main="PACF of s_star")
acf(st, main="ACF of s_star")
pacf(st, main="PACF of s_star")
```

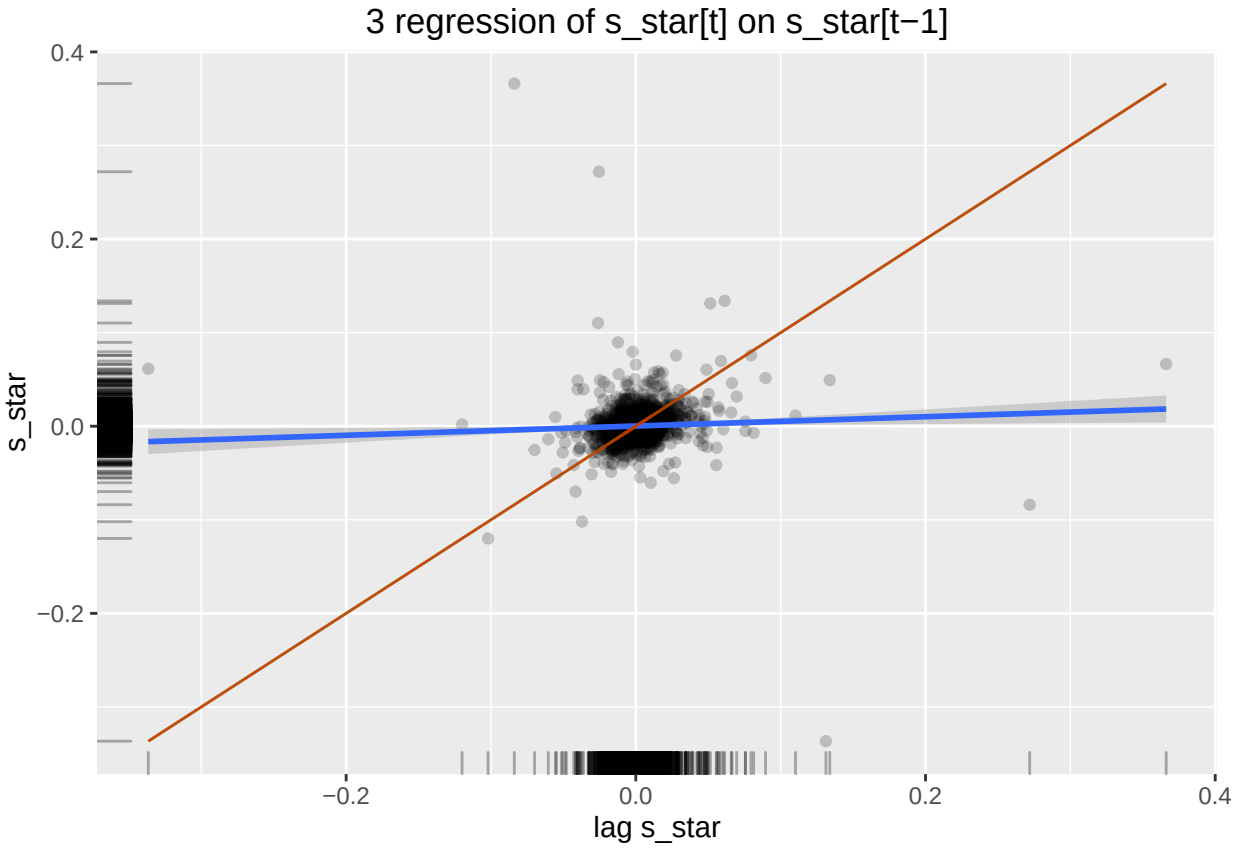









```r
# nonparametrci regression
lm <- lm(st ~ lag(st, -1))
np_l <- dyn$loess(st ~ lag(st, -1),degree=1)
np_nl <- dyn$loess(st ~ lag(st, -1),degree=2)
fitted_l <- fitted(np_l)
fitted_nl <- fitted(np_nl)
fitted_lm <- fitted(lm)

# visalization
ggplot(data=NULL, aes(x=st[-1],y=st[-length(st)])) +
  geom_rug(alpha=0.3) +
  geom_point(alpha=0.2) +
  geom_smooth(method="lm") +
  geom_line(aes(y=fitted_l[-1]), col="green",alpha=0.7) +
  geom_line(aes(y=fitted_nl[-1]), col="red", alpha=0.7) +
  xlab("lag s_star") +
  ylab("s_star") +
  ggtitle("3 regression of s_star[t] on s_star[t-1]")
```




```
# Hypothesis Testing
anova(np_l,np_nl)[-1]
```

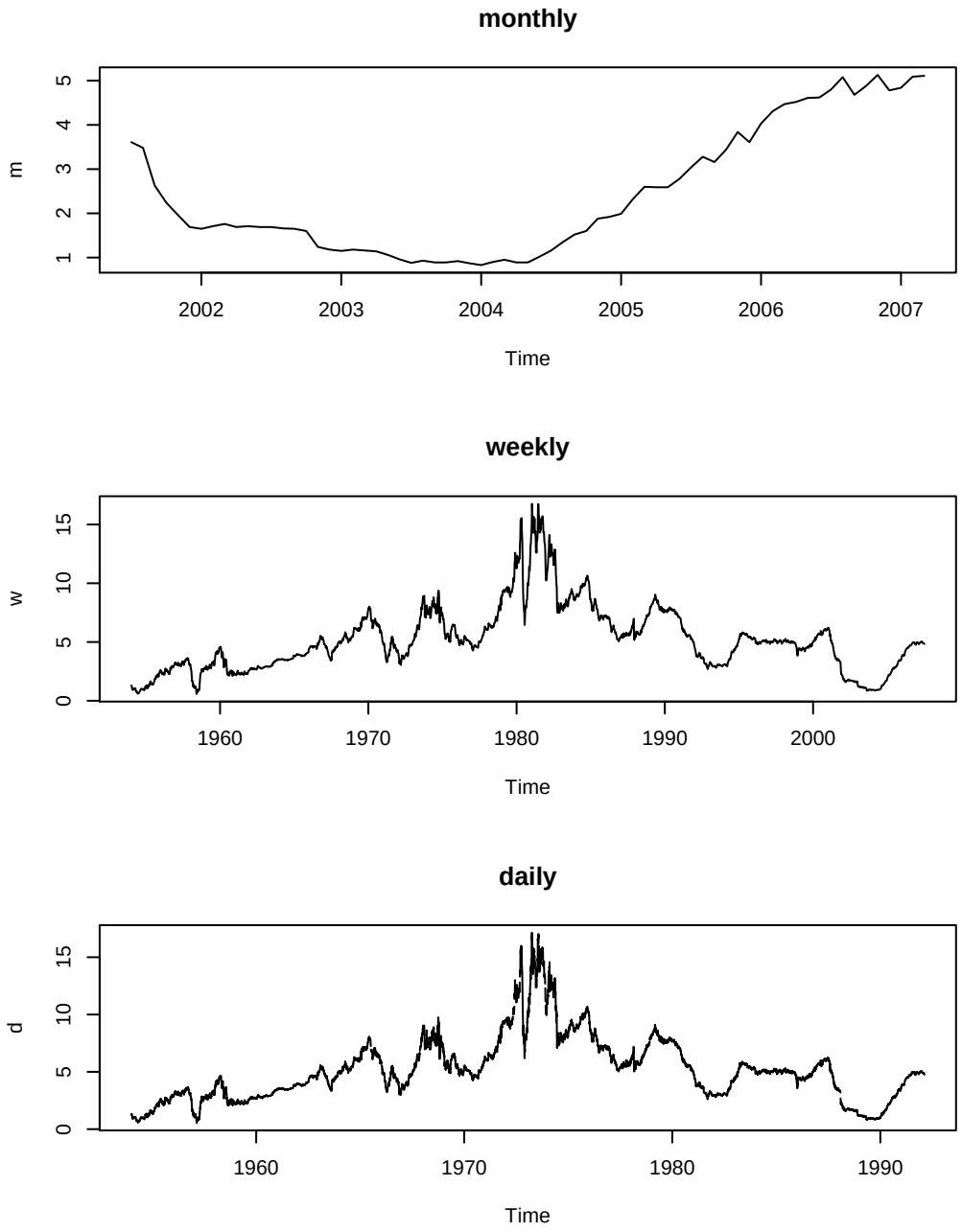
```
##             RSS F-value    Pr(>F)
## [1,] 2.7785e-30
## [2,] 2.9352e-30  76.473 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Results:

the linear and nonlinear approach gives exactly the same result, but from the scatter plot we can see the joint distribution of $s_t^\star$ and $s_{t-1}^\star$ is clustered at point $(0, 0)$, so the both the nonparametric models and paramethric linear model does not fit well. The Hypothesis Testing result show that we **reject** the null hypothesis that $s_t^\star$ is linear under the significance level 0.05.

4. Consider three real data sets for the U.S. Treasury bill (Secondary Market Rate): the daily, weekly and monthly 3-month Treasury bill.

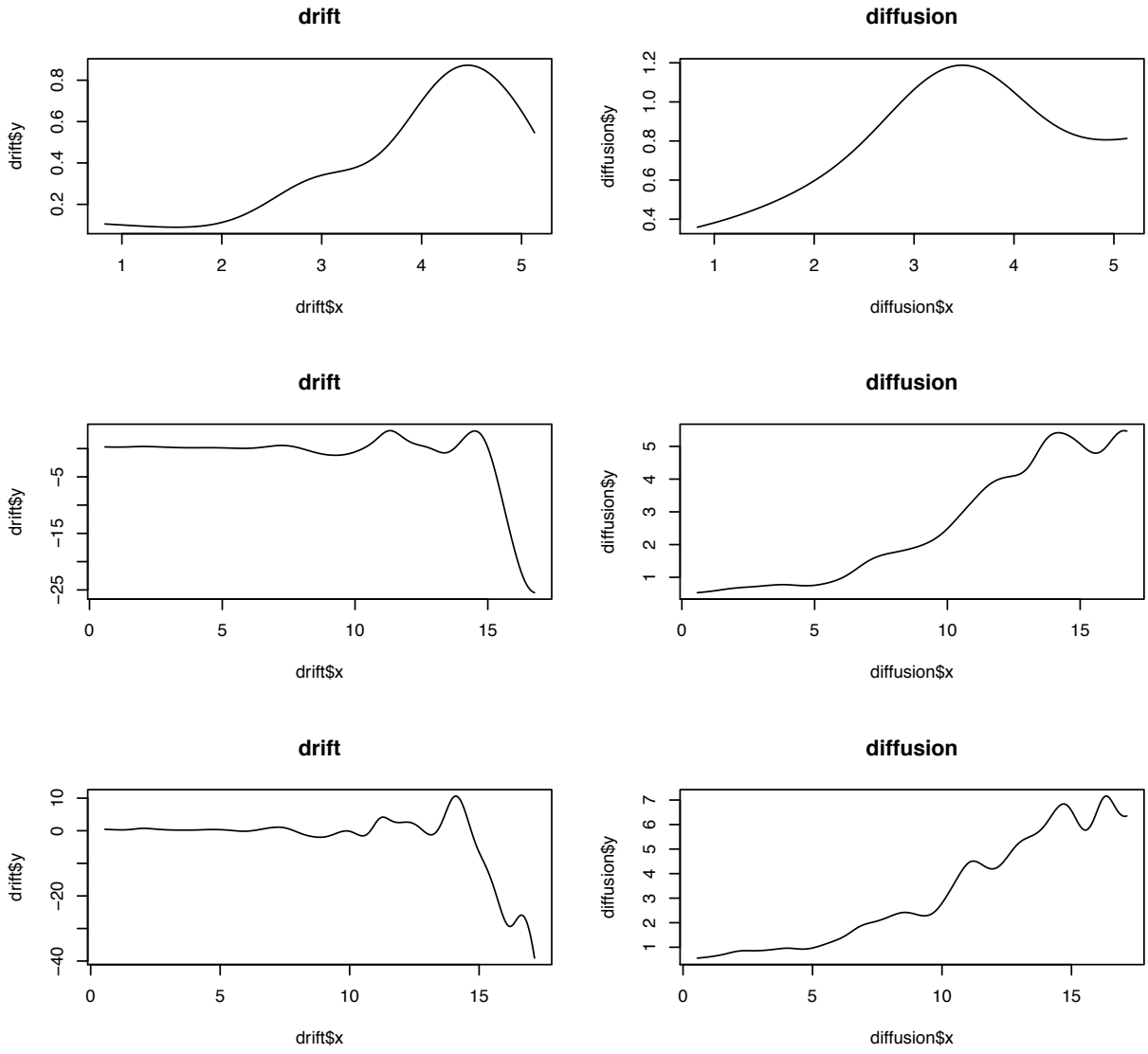
First let's see what the datas look like:



4-1. Apply the nonparametric regression estimation methods to estimate the drift and diffusion functions for each series and higher moments such as skewedness and kurtosis.

I will use the function `ksdrift` and `ksdiff` in R package `sde` , whose description in the document is "Implementation of simple Nadaraya-Watson nonparametric estimation of drift and diffusion coefficient", the default bandwidth is calculated using Scott's rule (i.e., $bandwidth = n^{-1/5} * sd(x)$). The estimation code and result is plot as follows:

(First row is **monthly** data, second is **weekly**, the bottom line is **daily**.)



Next we estimate the skewness and kurtoiss of the 3 data: Here I use `skewness` and `kurtosis` function in `moments` package:

```
require(moments)

skewness(m); kurtosis(m)
```

```
## [1] 0.633276
```

```
## [1] 1.933679
```

```
skewness(w);kurtosis(w)
```

```
## [1] 1.106601
```

```
## [1] 4.841152
```

```
skewness(d, na.rm = T);kurtosis(d, na.rm = T)
```

```
## [1] 1.112128
```

```
## [1] 4.872407
```

4-2. Any conclusions and comments on three drift and diffusion functions? Also, think about your conclusions on whether a jump diffusion model is appropriate for the data; see Johannes (2004).

I use R command `BNSjumptest` in package `highfreqency`to test the presence of jumps. It is Barndorff-Nielsen and Shephard (2006) tests for the presence of jumps in the price series,which examines the presence of jumps in highfrequency price series. It is based on theory of Barndorff- Nielsen and Shephard (BNS). The null hypothesis is no jumps.

The test result is that we **reject** the null hypothesis, i.e we consider there exist jumps in the series of intrest rates.

```
require(highfrequency)

BNSjumptest(m)
```

```
## $ztest
## [1] -33.80844
##
## $critical.value
## [1] -1.959964  1.959964
##
## $pvalue
## [1] 1.48243e-250
```

```
BNSjumptest(w)
```

```
## $ztest
## [1] -1398.895
##
## $critical.value
## [1] -1.959964  1.959964
##
## $pvalue
## [1] 0
```

```
BNSjumptest(d_new)
```

```
## $ztest
## [1] -6984.056
##
## $critical.value
## [1] -1.959964  1.959964
##
## $pvalue
## [1] 0
```