STAT 797: Non-parametric Density Estimation

Tobias Kuhlmann, Rui Zhang

12/16/2018

Research aim

Compare asymptotic MISE behaviour for kernel and logspline density estimators as $n \to \infty$

Models

Univariate kernel density estimator

We use a univariate kernel density function following Wand and Jones (1995). A density function can be estimated by $\hat{f}(x;h) = (nh)^{-1}\sum_{i=1}^{n} K\{(x - X_i)/h\}$, where K is a kernel function satisfying $\int K(x)dx = 1$ and h is the bandwidth. where K is a kernel function satisfying $\int K(x)dx = 1$ and h is the bandwidth.

Univariate density estimation with logspline

$K \geq 3$, $L < t_1 < \cdots < t_K < U$. Let B be a set of basis functions. β be a collection of feasible column vectors. A column vector β is said to be feasible if $\int_L^U exp(\beta_1 B_1(x) + \cdots + \beta_J B_J(x))dx < \infty$. Given $\beta \in B$, set $f(x;\beta) = exp(\beta_1 B_1(x) + \cdots + \beta_J B_J(x) - C(\beta)), L < x < U$ where $C(\beta) = log(\int_L^U exp(\beta_1 B_1(x) + \cdots + \beta_J B_J(x))dx)$. Then $f(y;\beta)$ is a positive density function on (L,U), and $\int_R f(x;\beta)dx = 1$.

There are several advantages when using logsplines to estimate densities. As one of the penalized approaches, logspline uses a maximum likelihood approach. It adds knots in those parts of the density where they are most needed. It has a natural way to estimate densities with bounded support. It avoids spurious bumps and gives smooth estimates in the tail of the distribution. And it can estimate the density even when some observations are censored.

Asymptotic MISE

We know that the best obtainable rate of convergence of the MISE of the kernel estimator is $\inf \text{MISE}_{h>0}\{\hat{f}(;h)\} \sim \frac{5}{4}\{\mu_2(K)^2 R(K)^4 R(f'')\}^{\frac{1}{5}} n^{-\frac{4}{5}} = C_2 n^{-\frac{4}{5}}$

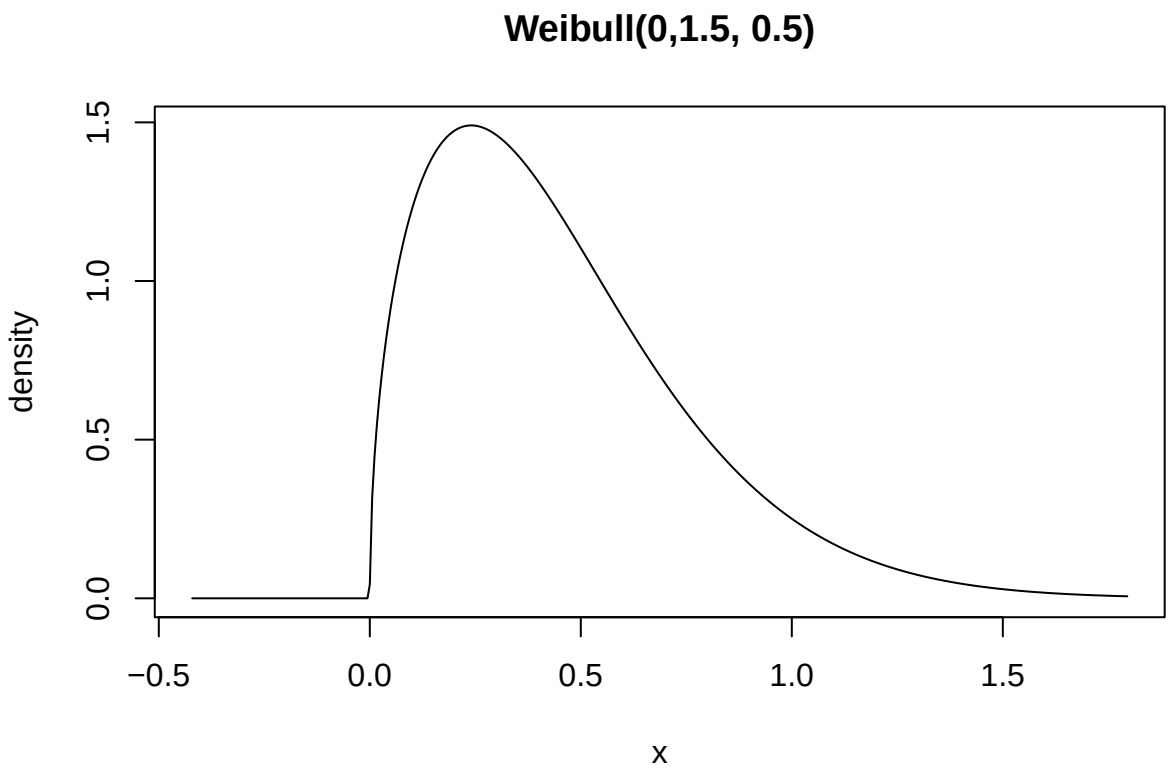
Asymptotic MISE of logspline density estimations have not been studied in literature yet.

Simulation

We conducted a monte carlo experiment with the normal distribution, weibull distribution, and chi2 distribution. The simulation experiment for the normal distribution is the only valid one, as we realized. Non-parametric density estimation with kernels and logsplines have the assumption of a smooth density, which is violated by the weibull and chi2 distribution in our experiment, as can be seen below. Both are only defined for $x >= 0$, however the r random function simulates invalid values below zeros, which introduces a bias to our density estimations. To fit kernel density and logspline density estimators, we used the packages corresponding to their original papers, respectively. KernSmooth and logspline.

```
# Weibull
y <- rweibull(n=50, shape=1.5, scale=0.5)
h <- dpik(y) # select optimal bandwidth
kernel <- bkde(x=y, bandwidth=h) # kde following Wand (1995)
```

```r
density <- dweibull(kernel$x, shape=1.5, scale=0.5)
plot(kernel$x, density, type='l', xlab='x')
title("Weibull(0,1.5, 0.5)")
```



Below is the code for our monte carlo experiment. We use 20 sample sizes, equally spaced on log scale, which 10 random samples each. Asymptotic behaviour can be seen in the plots below.

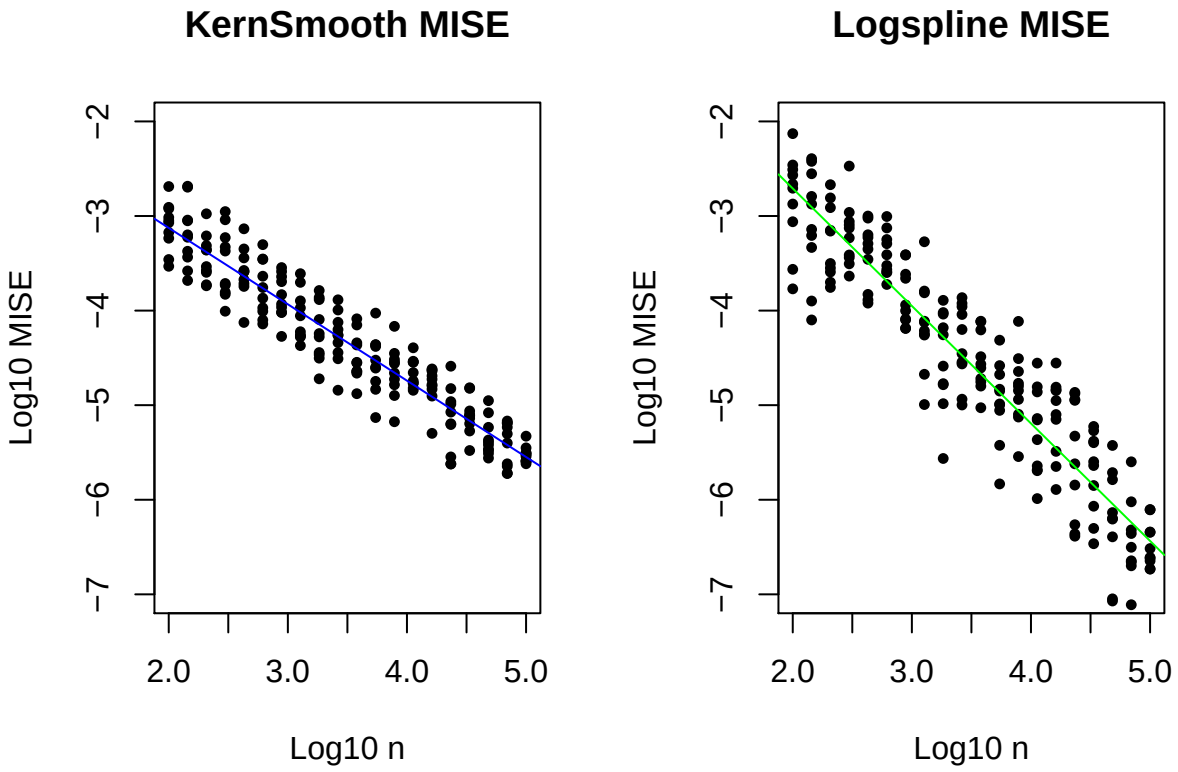
```r
# Normal Distribution
# ------------------------------------------------------------
# Monte Carlo
# ------------------------------------------------------------
reps.per.n <- 10
log10.ns <- seq(from=2,to=5,length=20) # equally space n's on log scale
ns <- round(10^log10.ns)
log10.ns <- log10(ns)

# make storage for what we want to keep
keep.kernel.wand <- data.frame(log.n = rep(log10.ns,each=reps.per.n),log.mise=NA)
keep.logspline <- data.frame(log.n = rep(log10.ns,each=reps.per.n),log.mise=NA)

# let's keep the fits too (it's always useful to look at estimates)
keep.kernel.wand.fits <- matrix(NA,length(keep.kernel.wand$log.n),401)
keep.logspline.fits <- matrix(NA,length(keep.logspline$log.n),401)


counter <- 1
for (n.i in ns)
{
  for (mc.i in 1:reps.per.n)
  {
```

```
    # generate data
    # TODO1: add some noise to distributions?
    y <- rnorm(n.i, 0, 1)

    # Univariate kernel density estimator from KernSmooth package (Wand (1995))
    h <- dpik(y) # select optimal bandwidth
    fit <- bkde(x=y, bandwidth=h, gridsize = 401) # kde
    # keep fits
    keep.kernel.wand.fits[counter,] <- fit$y
    # calc mse
    mise <- mean((fit$y-dnorm(fit$x, 0, 1))^2)
    # log mise
    keep.kernel.wand$log.mise[counter]  <- log10(mise)

    # store kde x values for log spline evaluation
    x = fit$x

    # Logspline density estimator
    fit <- logspline(y) # fit logspline
    dens <- dlogspline(q=x, fit=fit) # get density values
    # keep fits
    keep.logspline.fits[counter,] <- dens
    # calc mise
    mise <- mean((dens-dnorm(x, 0, 1))^2)
    # log mise
    keep.logspline$log.mise[counter]    <- log10(mise)

    counter <- counter + 1
  }
}

#

#
par(mfrow=c(1,2))
# kde Wand (1995) plot
plot(keep.kernel.wand$log.n,keep.kernel.wand$log.mise,xlab="Log10 n",ylab="Log10 MISE",type="n", ylim=c
points(keep.kernel.wand$log.n,keep.kernel.wand$log.mise,pch=16,cex=.75)
lmfit_kernel.wand <- lm(log.mise~log.n,data=keep.kernel.wand)
summary(lmfit_kernel.wand)
```

```
##
## Call:
## lm(formula = log.mise ~ log.n, data = keep.kernel.wand)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.60271 -0.18708  0.02044  0.19407  0.56846
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.51117    0.07239  -20.88   <2e-16 ***
## log.n       -0.80738    0.02002  -40.34   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2577 on 198 degrees of freedom
## Multiple R-squared:  0.8915, Adjusted R-squared:  0.891
## F-statistic:  1627 on 1 and 198 DF,  p-value: < 2.2e-16
```

```
abline(lmfit_kernel.wand,col="blue")
title("KernSmooth MISE")
# logspline plot
plot(keep.logspline$log.n,keep.logspline$log.mise,xlab="Log10 n",ylab="Log10 MISE",type="n", ylim=c(-7,
points(keep.logspline$log.n,keep.logspline$log.mise,pch=16,cex=.75)
lmfit_logspline <- lm(log.mise~log.n,data=keep.logspline)
abline(lmfit_logspline,col="green")
title("Logspline MISE")
```




```
# look at kernel coefficient of interest
print(summary(lmfit_kernel.wand)$coefficients)
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -1.5111681 0.07238759 -20.87606 6.446251e-52
## log.n       -0.8073822 0.02001579 -40.33727 1.906826e-97
```

```
print(confint(lmfit_kernel.wand))
```

```
##                 2.5 %     97.5 %
## (Intercept) -1.6539177 -1.3684185
## log.n       -0.8468537 -0.7679108
```

```
# look at coefficient of interest
print(summary(lmfit_logspline)$coefficients)
```

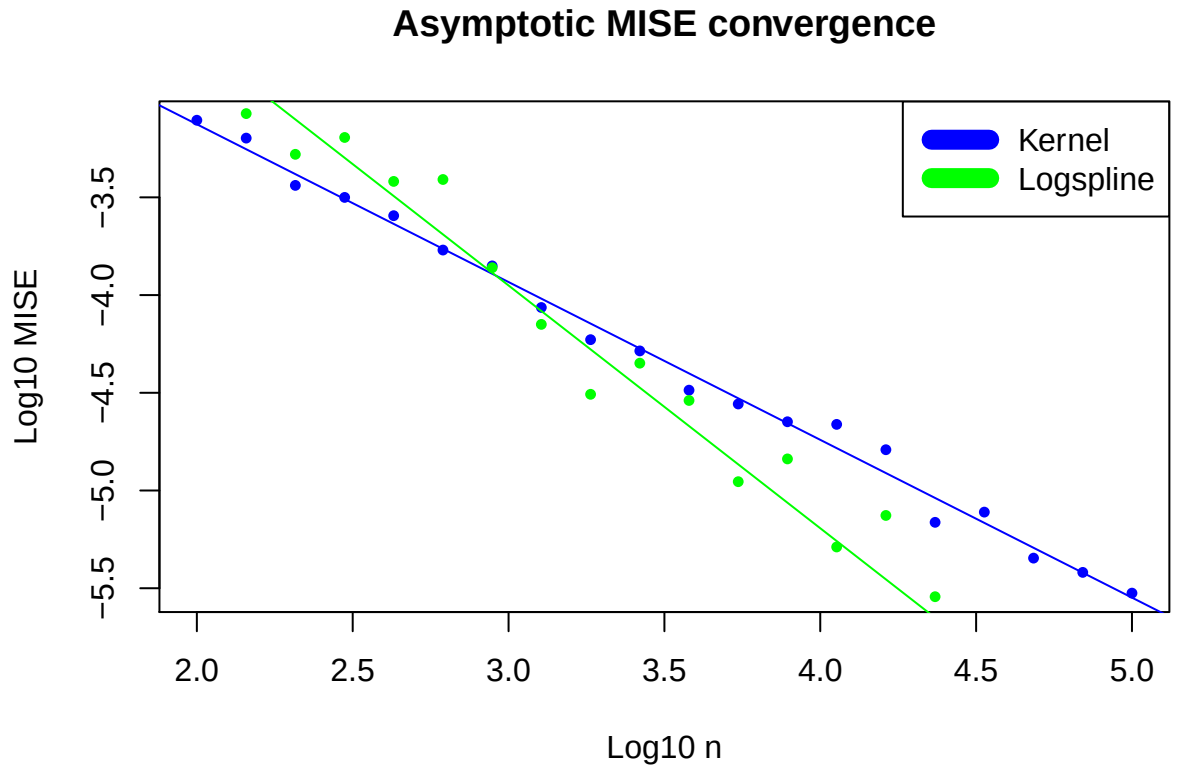
```
##               Estimate Std. Error    t value     Pr(>|t|)
```

```
## (Intercept) -0.2241916 0.13024043  -1.721367 8.674616e-02
## log.n       -1.2422290 0.03601259 -34.494299 1.158404e-85
```

```
print(confint(lmfit_logspline))
```

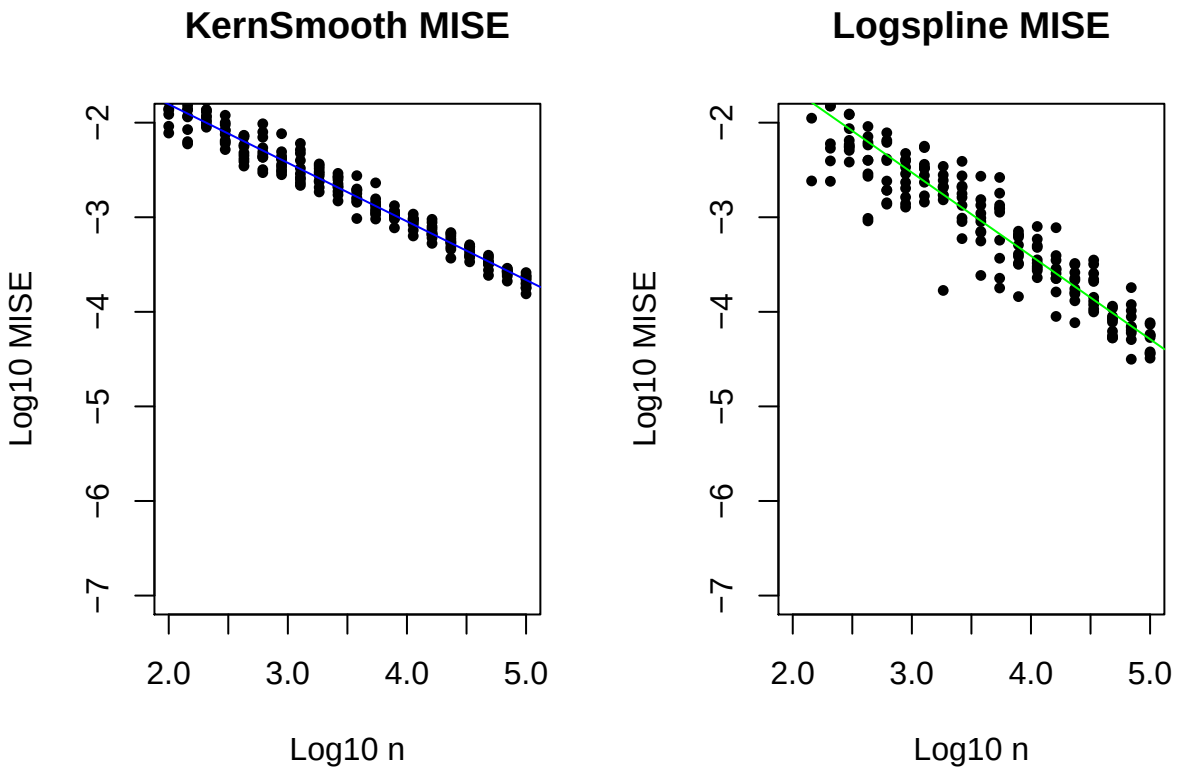
```
##                 2.5 %      97.5 %
## (Intercept) -0.481028  0.03264482
## log.n       -1.313246 -1.17121157
```

```
# combined mean plots
par(mfrow=c(1,1))
kernel_mean_mise = aggregate(keep.kernel.wand, by=list(keep.kernel.wand$log.n), mean)
logspline_mean_mise = aggregate(keep.logspline, by=list(keep.logspline$log.n), mean)
# kernel
plot(kernel_mean_mise$log.n,kernel_mean_mise$log.mise,xlab="Log10 n",ylab="Log10 MISE",type="n")
points(kernel_mean_mise$log.n,kernel_mean_mise$log.mise,pch=16,cex=.75, col='blue')
lmfit <- lm(log.mise~log.n,data=keep.kernel.wand)
abline(lmfit,col="blue")
# logspline
points(logspline_mean_mise$log.n,logspline_mean_mise$log.mise,pch=16,cex=.75, col='green')
lmfit <- lm(log.mise~log.n,data=keep.logspline)
abline(lmfit,col="green")
title("Asymptotic MISE convergence")
legend("topright", c("Kernel", "Logspline"), col=c("blue", "green"), lwd=10)
```

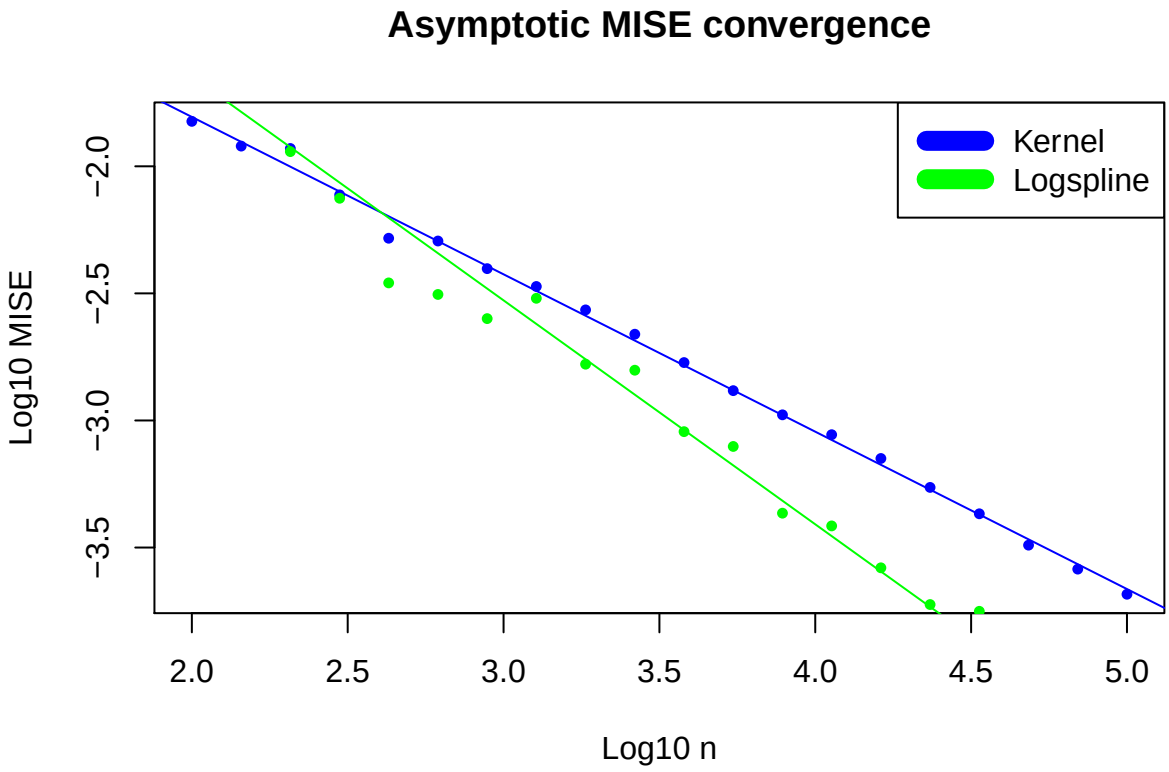



```
# ------------------------------------------------------------
```

Weibull Distribution

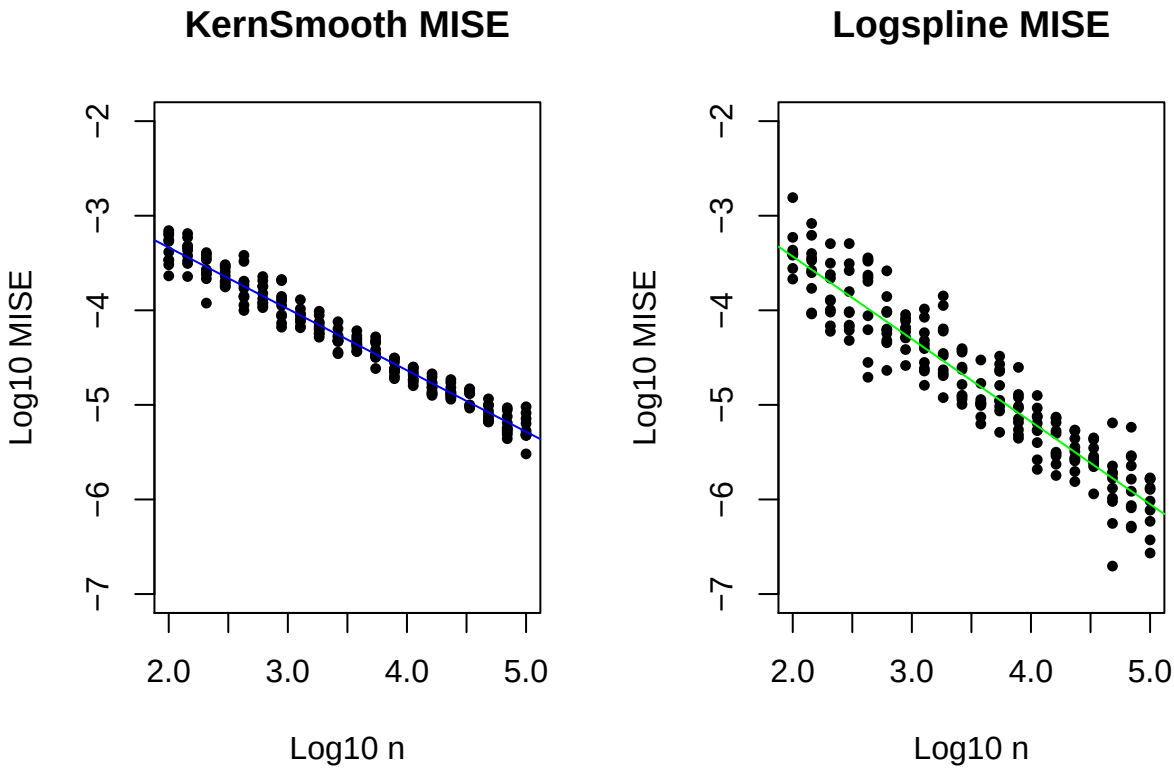



```
## [1] "KDE regression results"

##               Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept) -0.5674673 0.031060832 -18.26955  2.374640e-44
## log.n       -0.6191805 0.008588585 -72.09343 4.572085e-144

##                  2.5 %     97.5 %
## (Intercept) -0.6287198 -0.5062148
## log.n       -0.6361173 -0.6022437

## [1] "Logspline regression results"

##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  0.1157849 0.07738331   1.496252 1.361803e-01
## log.n       -0.8810938 0.02139715 -41.178099 4.946425e-99

##                   2.5 %     97.5 %
## (Intercept) -0.03681632  0.2683862
## log.n       -0.92328933 -0.8388982
```



Chi2 Distribution

```
## Warning in logspline(y): Not all models could be fitted
```



```
## [1] "KDE regression results"
```

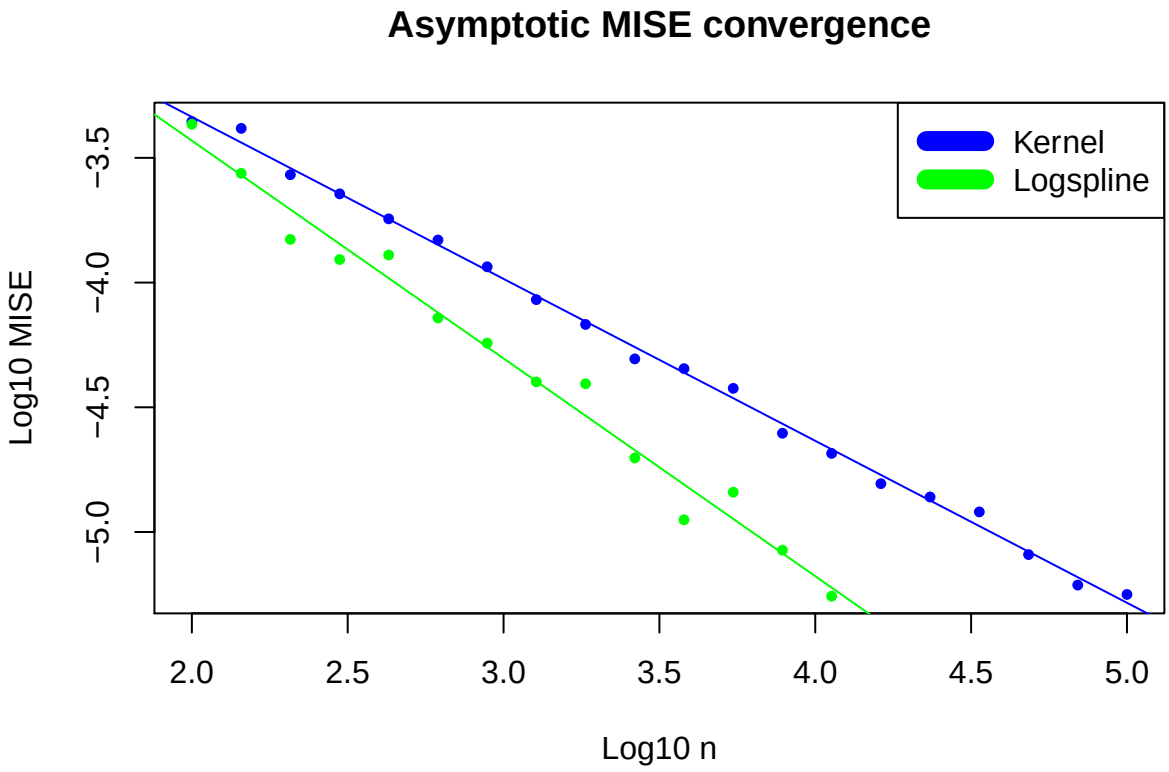
```
##               Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept) -2.0374438 0.032042380 -63.58591 1.028324e-133
## log.n       -0.6492672 0.008859991 -73.28079 2.022532e-145
```

```
##                  2.5 %     97.5 %
## (Intercept) -2.1006320 -1.9742557
## log.n       -0.6667393 -0.6317951
```

```
## [1] "Logspline regression results"
```

```
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -1.6851068 0.07865674 -21.42355 1.832102e-53
## log.n       -0.8730091 0.02174926 -40.13972 4.536501e-97
```

```
##                  2.5 %     97.5 %
## (Intercept) -1.8402192 -1.5299943
## log.n       -0.9158991 -0.8301192
```



Results

We could replicate the asymptotic MISE convergence rate of -4/5 on log scale for kernel density estimations. Our simulation for logsplines with normal distribution suggests a linear convergance rate on log scale of -1.25. This has to be further confirmed in other simulations, especially since confidence intervals are wide (ca. +-0.07). Besides that, Logspline MISE converges faster to zero than kernel density estimation in all three experiments. As two our distributions turned out to violate the assumptions, one should be careful with distribution definition bounds, ensuring density is smooth.

For further research

Further research could theoretically derivate the asymptotic logspline MISE convergance rate. It is also open to confirm, if that convergance rate is linear on log scale or of a different type. One definitely need to conduct simulation experiments with other valid densities, or try fitting the methods with estimation bounds on defined distribution values.

References

- Stone, Hansen, Kooperberg, and Truong, Polynomial Splines and their Tensor Products in Extended Linear Modeling, Annals of Statistics, Volume 25,Issue 4(Aug., 1997), 1371-1425.
- MP. Wand and M.C.Jones, Kernel Smoothing, Chapman& Hall, 1995.