Chapter 7.

Nonparmatric Regression Models

In Chapter 3, linear regression models were studied in detail. In practice, however, a greatest danger is that the linear relation between the conditional mean response and the covariates does not hold. Natually one would consider nonlinear relations in the model, and this becomes increasingly complicated. We begin with the nonlinear regression for a single covariate and extend to multiple covaraites.

7.1. Regression with a single input.

With data (x_i, y_i), $i = 1, ..., n$, as the input-output pairs with the inputs x_i being just one dimension, the simple linear regression model is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \qquad i = 1, ..., n,$$

where $E(\epsilon_i|x_i) = 0$. The model can also be stated as $E(y_i|x_i) = \beta_0 + \beta_1 x_i$. This is a special case of a much larger model

$$y_i = f(x_i) + \epsilon_i, \qquad f \in \mathcal{F},$$

or $E(y_i|x_i) = f(x_i)$, where $\mathcal{F}$ is certain class of functions. Linear model is essentially requiring $\mathcal{F}$ be all linear functions.

Polynomial regression A natural but crude extension from linear linear model is to consider $\mathcal{F}$ as collections of polynomial functions up to certain order p. The linear model is $p = 1$. If $p = 0$, the output y is not related with input x, and we estimate β_0 by $\bar{y}$. In general, the model becomes

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_p x_i^p + \epsilon_i.$$

This is a multiple linear regression model with p covariates $(x_i, x_i^2, ..., x_i^p)$. Then, the methods and theory developed for linear regression models can be applied with out any problem. A particular problem is the selection of p. A large p corresponds to a large model, which may lead to large variance and overfit, with small training error but large prediction error. Here the model selection methods such as AIC and BIC can be applied. In particular, the general purpose cross-validation approach is useful.

Despite the simplicity, a drawback of the polynomial regression model is that even a moderately large d can often result in extremely large or small values of x_i^d, creating very unstable covariates. Consequently the regression results may be easily distorted by the outliers. Moreover, the polynomial functions are not sensitive to capture local variations of functions, especially those that are highly varying in one part but quite smooth on the other part of the real line.

Basis function regression In polynomial regression, $\mathcal{F}$ is a linear space of functions that are linear combination of $1, x, x^2, ..., x^p$, totally of $p + 1$ dimension. These power functions can be viewed as basis functions that span the space $\mathcal{F}$. In this view, we could could have more alternative choices of the basis functions. For examples, consider a partition of the real line into

$$(-\infty, t_0],\ (t_1, t_2],\ (t_2, t_3],\ ...,\ (t_{K-1}, t_K],\ (t_K, \infty).$$

Then, one can use the indicator functions, or, slightly more generally, use the polynomial functions of degree p on each interval (t_k, t_{k+1}), as the basis functions to form $\mathcal{F}$. The cut points $t_1, ..., t_K$ will be called knots.

The advantage is one can keep the order p to be low, and the number of intervals to be large, so as to capture local variability of the functions. A drawback is that the basis functions are not smooth. Actually they are not even continuous.

There are various sophisticated choices of the basis functions that can overcome the underisable discontinuity, such as wavelets, or Fouriour functions. The regression splines are more straightforward and easier to be understood.

Regression splines Consider truncated power function

$$b_j(x;\tau) = (x-\tau)_+^j$$

where $x_+ = xI(x >= 0)$ denotes the positive part, and $x_- = -xI(x < 0)$ denotes the negative part. The trancated power functions are continuous functions at least. It is then natural to consider the collection of $b_j(x, t_k)$, for $k = 1, ..., K$ and $j \leq d$ as basis function. Here d refers to the degree of the power. Human eyes cannot identify the discontinuity beyond second derivative. Consequnently, it is believed that using functions that are smooth at the second derivative would be generally smooth enough. Cubic splines, with knots $t_1, ..., t_K$, are the linear combinations of truncated power functions $b_j(x; t_k) : j = 0, 1, 2, 3$ and $k = 1, ..., K$, that are smooth up to the second derivatives. There are totally $K + 1$ intervals and on each interval there are four functions, along with 3 restrictions at each of the K knots. Therefore the dimension of the basis functions for cubic splines are $4(K + 1) - 3K = K + 4$.

Cubic regression splines are known to be highly unstable near the boundary. A remedy to solve the problem is to force the functions outside the left boundary t_1 and the right boudnary t_K to be linear. This reduces the number of parameters by four, 2 on the leftmost interval and 2 on the right most interval. The resulting splines are called cubic natural splines and the total dimension is $K + 4 - 4 = K$. In fact, the basis functions of cubic natural spline can be expressed as:

$$1,\ x,\ g_1(x) - g_{K-1}(x),\ g_2(x) - g_{K-1}(x), ..., g_{K-2}(x) - g_{K-1}(x)$$

where $g_k(x) = [(x - t_k)_+^3 - (x - t_K)_+^3]/(t_K - t_k)$ for $k = 1, ..., K - 1$.

There are many different ways to express the basis functions, as long as they span the same linear space of functions. B-spline is computationally more efficient, especially when K is large.

Smoothing splines A difficult yet critical problem for regression spline is one needs to fix the number and the locations of the knots. Smoothing splines takes the "Loss + Penalty" formula and can avoid this problem. Define

$$RSS(f, \lambda) = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int \{\ddot{f}(x)\}^2 dx.$$

The first term measuares the fit and the second term penalizes the non-smooth or highly varying functions. Here $\ddot{f}$ is the second derivative of f, and λ is a tuning parameter, which can be determined by cross-validation. For fixed λ, the smoothing spline estimator of f is

$$\hat{f}_\lambda = \text{argmax}_f \text{RSS}(f, \lambda).$$

Surprisingly, the solution is natural cubic spline with knots at $x_1, ..., x_n$. It appears that smoothing splines would overfit as the dimension of basis functions is too large. In fact, λ plays the role of constraining the model to a smaller subset. The larger the λ, the smaller the variance and the larger the bias. In the extreme case, when λ approaches infinity (infinite penalty for non-smoothness), the resulting estimator $\hat{f}_\lambda(x)$ approaches the least squares estimator $\hat{\beta}_0 + \hat{\beta}_1 x$. On the other hand, when λ approaches 0 (no penalty for non-smoothness), $\hat{f}_\lambda$ approaches the perfect fit that $\hat{f}_\lambda(x_i) = y_i$ for $i = 1, ..., n$.

Local regression A local regression takes one single point, say x_0, at a time, and tries to predict $f(x_0)$ based on the data. For simplicity, let $f(x_0) = \beta_0$ and $\dot{f}(x_0) = \beta_1$. A key idea is that

the function $f(x)$ in the neighborhood of x_0 can be approxiamted by a polynomial function. For illustration, we only consider linear approximation. Write

$$f(x) \approx f(x_0) + \dot{f}(x_0)(x - x_0) = \beta_0 + \beta_1(x - x_0) \quad \text{for } x \text{ in the neighborhood of } x_0$$

If the neighborhood is well defined, then a least squares procedure can be carried out based on the data (y_i, x_i) with x_i in this neighborhood to obtain the estimator of β_0. A better and more systematic way is to consider a weight function that places higher weight on observations with x_i close to x_0 and lower weight on observations with x_i far away from x_0. This is usually accomplished through a kernal function k. Typical kernel functions are

(a) Uniform kernel: $k(t) = 1/2I(|t| \leq 1)$.
(b) Triangle kernel: $k(t) = (1 - |t|)I(1t| \leq 1)$.
(c) Gaussian kernel: $k(t) = e^{-t^2/2}/\sqrt{2\pi}$
(d) Epanecknikov kernel: $k(t) = 3/4(1 - t^2)_+$
(e) Logistic kernel: $k(t) = 1/(e^t + e^{-t} + 2)$.
(f) Sigmoid kernel: $k(t) = 2/(\pi(e^t + e^{-t}))$.

Let h be the bandwidth and define $w_i = (1/h)k((x_i - x_0)/h)$. A weighted least squares estimator of (β_0, β_1) is

$$(\hat{\beta}_0, \hat{\beta}_1) = \text{argmax}_{\beta_0,\beta_1} \sum_{i=1}^{n} w_i(y_i - \beta_0 - \beta_1(x_i - x_0))^2.$$

Then, the estimator of $f(x_0)$ is $\hat{\beta}_0$. In fact,

$$\hat{\beta}_0 = \bar{y}_w - \hat{\beta}_1(\bar{x}_w - x_0) = \bar{y}_w - \frac{\sum_{i=1}^n w_i(y_i - \bar{y}_w)(x_i - \bar{x}_w)}{\sum_{i=1}^n w_i(x_i - \bar{x}_w)^2}(\bar{x}_w - x_0),$$

where $\bar{y}_w = \sum_{i=1}^n w_i y_i / \sum_{i=1}^n w_i$ and $\bar{x}_w = \sum_{i=1}^n w_i x_i / \sum_{i=1}^n w_i$ are respectively weighted average of y_i and x_i.

The bandwidth h plays a key role in local estimation, just as the role of number of knots in regression spline or the smoothing parameter λ in smoothing spline. A small h means a large model with small bias but large variance. As this estimator has closed form expression, its mean and variance is easily available. The bias is at the order of h^2 and the variance is at the order of $1/nh$, making the optimal order of h at $n^{-1/5}$. In practice, however, h is usually determined by cross-validation.

Linear smoothers and the degree of freedom All the regression method discussed above, can be called linear methods in the sense that

$$\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}.$$

where $\hat{\mathbf{y}}$ in the n-vector of fitted values and $\mathbf{S}$ is $n \times n$ matrix which only depends on the inputs $x_1, ..., x_n$. For model $y_i = f(x_i) + \epsilon_i$ with $E(\epsilon_i) = 0$ and $\text{var}(\epsilon_i) = \sigma^2$. The degree of freedom of a model fitting is defined as

$$\text{d.f} = \sum_{i=1}^{n} \text{cov}(\hat{y}_i, y_i)/\sigma^2.$$

Then, for linear methods, the degree of freedom is

$$\text{d.f} = \frac{1}{\sigma^2} trace(\text{cov}(\hat{\mathbf{y}}, \mathbf{y})) = trace(\mathbf{S}).$$

There are several special cases. Consider the least squares estimation in linear regresion models with p inputs addressed in Chapter 3. Then, $\mathbf{S}$ is the hat matrix, i.e.,

$$\mathbf{S} = \mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}$$

And the degree of freedom is $trace(\mathbf{S}) = trace(\mathbf{H}) = p + 1$. If $\hat{y}_i$ is a constant irrelevant with data, the degree of freedom is 0; and $\hat{y}_i = \bar{y}$ for all i, then degree of freedom is 1. On the other hand if $\hat{y}_i = y_i$ for all i, then the degree of freedom n. Degree of freedom is one proper measurement of how large the model is.

7.2. Regression with multiple inputs: the GAM and the MARS

Though the above discussed smoothing methods can be formally extended to high dimension or multiple inputs, their actual implementation is generally difficult if not impossible, mainly due to the curse of dimensionality. Consider

$$y_i = f(x_{i1}, x_{i2}, ..., x_{ip}) + \epsilon_i.$$

Even if p is moderately high, such as 20, the search for functions of 20 variables becomes formidable. For example, consider local linear regression. Even if the commonly used kernel functions can be extended to high dimension by using the Euclidean norm, number of observations in a reasonably small neighborhood becomes increasely sparse. Suppose the sample size is 1000 and a one-dimension input follow unifomr distributoin on $[0, 1]$, the number of observations in an interval of length 0.1 would be about 100. But, if there are 20 inputs, each following uniform distribution on $[0, 1]$, the number of observations in a 20 dimension cube with side length 0.1 would be only about $1000 \times 0.1^{20} \approx 0$.

The space of functions of p variables is too large in general. One can limit the space to additive functions. This gives rise to the generalized additive model (GAM):

$$y_i = f_1(x_{i1}) + f_2(x_{i2}) + ... + f_p(x_{ip}) + \epsilon_i.$$

A back-fitting algorithm can be used to obtain the estimates of f_i. The algorithm takes advantage of the estimation methods for single input smoothers. It updates each function f_k by holding the rest as known. It is specifically described as follows.

(a) Initialize the estimator of f_1, ..., f_p, denoted as $\hat{f}_1, ..., \hat{f}_p$.

(b) Given estimates $\hat{f}_1, .., \hat{f}_{k-1}$, $\hat{f}_{k+1}$, ..., $\hat{f}_p$, compute

$$\tilde{y}_i = y_i - \hat{f}_1(x_{i1}) - \hat{f}_{k-1}(x_{i,k-1}) - \hat{f}_{k+1}(x_{i,k+1}) - ... - f_p(x_{ip})$$

(c) Run nonlinear regression with response $\tilde{y}_i$ and single input x_{ik}, to obtain the estimate of f_k. Update $\hat{f}_k$ by this estimate.

(d) Continue with the update of f_{k+1}. (If $k = p$ continue the update of f_1.)

(e) Repeat till convergence.

The MARS (multivariate adaptive regresison splines) is nonlinear regression model which is effective in dealing with high dimensional inputs. It very much resembles forward stepwise regression. Forward stepwise regression works with the original inputs, and continue to add the inputs to the current model that has the largest decrease in RSS. MARS works with linear splines and continue to add the splines whose interaction with the current model produces largest decrease in RSS. The specific procedures is described in the following.

Define $c^+_{js}(\mathbf{x}) = (x_j - x_{sj})_+$ and $c^-_{js}(\mathbf{x}) = (x_j - x_{sj})_-$, for $j = 1, ..., p$ and $s = 1, ..., n$. Here, with slight abuse of notation, $\mathbf{x}$ is p-vector with components $(x_1, .., x_p)$, so x_j is the j-th compontent of $\mathbf{x}$, and x_{sj} is the j-th component of $\mathbf{x}_s$, observation s. K is a pre-specified number, related with the size of the model. Note that c^+_{js} and c^-_{js} and h_k in the following are viewed as function of $\mathbf{x}$.

(a) Step 0. Initialize the estimator $\beta_0 = \bar{y}$. Let $h_0(\mathbf{x}) = 1$. Set $\mathcal{M}_0 = \{h_0\}$, and $RSS_0 = \sum_{i=1}^n (y_i - \beta_0 h_0(\mathbf{x}_i))^2$.

(b) Step $k = 1, ..., K$. For all $h_0, h_1, ..., h_{2k-2}$. For any $l = 0, 1, ..., 2k-2$ and any $j = 1, ..., p$, $s = 1, ..., n$. Run least squares procedure for linear regression model

$$y_i = \beta_0 h_0(\mathbf{x}_i) + \beta_p h_1(\mathbf{x}_i) + ...\beta_{2k-2} h_{2k-2}(\mathbf{x}_i) + \beta_{2k-1} h_l(\mathbf{x}_i) c_{js}^+(\mathbf{x}_i) + \beta_{2k} h_l(\mathbf{x}_i) c_{js}^-(\mathbf{x}_i) + \epsilon_i$$

Compute the RSS. Find the one with the smallest RSS, say l^*, j^*, s^*.

(c) Define $h_{2k-1}(\cdot) = h_{l^*}(\cdot) c_{j^* s^*}^+(\cdot)$ and $h_{2k}(\cdot) = h_{l^*}(\cdot) c_{j^* s^*}^-(\cdot)$ and $M_k = \{h_0, h_1, ..., h_{2k}\}$.

(d) Return the final model:

$$y_i = \sum_{j=0}^{2K} \beta_j h_j(\mathbf{x}_i) + \epsilon_i$$

The resulting model usually is large and may over-fit the data, and backward deletion may be needed to produce the desired model. The backward deletion procedure deletes the term whose removal has the smallest increase in RSS, as in backward stepwise regression. Alternatively, generalized cross validation can be applied to find the best model.

7.3. Examples.

Examples 1

In this example, we use pot price of gold (Au) to predict the stock price of 600547 (ShanDong Gold). Specifically, we consider the close-to-close price change ratios (c2c) for these two securities. In Chapter 2, we studied their relationship by linear regression. In this example, we use the regression spline method.

The time range of the dataset is from Aug 28, 2013 to July 20, 2016, totally 3132 trading days. The first 2500 trading days will be treated as training set, and the remaining 632 trading treated as test set.

Here we use quantiles $(10\%, 20\%, \ldots 90\%)$ as knots in the model setting. They are listed in the following table.

10%	20%	30%	40%	50%	60%	70%	80%	90%
-0.014	-0.008	-0.004	-0.001	0.001	0.003	0.006	0.009	0.014

Table 1: Quantiles of C2C(Au) in the training set

And we will consider the commonly-used cubic splines. The fitting result is shown in Table 2. The plot of the fitting spline is shown in Figure 1.

The results show that only t_9 and t_{10} are significant, implying that the stock price may not be very sensitive to gold price change, except for violent increase. The spline plot also shows that this model is highly curved near the boundary, but for the interval which contains most data points (roughly -0.04 to 0.04), the spline is very close to a straight line.

The fitted model is used to predict $c2c$ in the test set. The mean and standard deviation of the absolute error (i.e. $|c2c_{predict} - c2c_{true}|$) are 0.02194523 and 0.0234526 respectively, which is moderate compared with the range of $c2c$ itself (from -10% to 10%). (Note: for the training set, the mean and standard deviation of the absolute error are 0.02177864 and 0.02125378 respectively)

Examples 2

In the real case, since the trading hours are different for A stock (9:30am to 3pm, HK time) and gold (e.g. London Gold, 7:30am to 3:30am, HK time), if gold price rises high during the evening,

	Dependent variable:
	c2c
t_0	-0.046
	(0.031)
t_1	0.017
	(0.023)
t_2	0.018
	(0.024)
t_3	0.022
	(0.024)
t_4	0.025
	(0.024)
t_5	0.023
	(0.024)
t_6	0.030
	(0.024)
t_7	0.033
	(0.024)
t_8	0.032
	(0.024)
t_9	0.108^{***}
	(0.027)
t_{10}	0.104^{***}
	(0.037)
Constant	-0.023
	(0.024)
Observations	2,500
R^2	0.138
Adjusted R^2	0.134
Residual Std. Error	0.031 (df = 2488)
F Statistic	36.074^{***} (df = 11; 2488)
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 2: Results of regression splines for training set

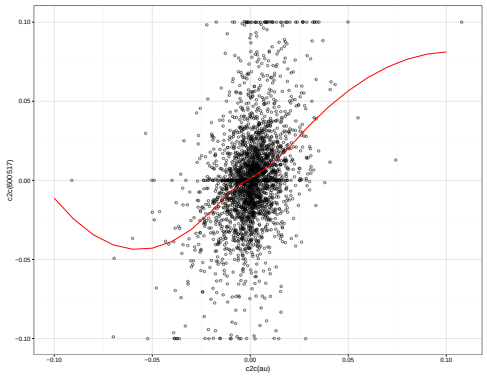

Figure 1: Plot of regression spline for training set

the stock price will usually gap higher when market opens. In this circumstance, c2c may not be so useful.

In this example, we try to predict o2c (the change ratio of today's open price to today's close price) by using four covariates: c2c of gold as example 1 showed, c2o of the stock (the change ratio of yesterday's close price to today's open price) and c2c of U.S. Dollar Index and c2c of Shanghai composite index (yesterday's value). Here we consider the GAM model which we learned in 7.2. The results of significance shows that the smooth term f(sh) is not significant. A further reguarlization adds an additional penalty for each smooth, which attempts to remove the entire smoothing term if if it is not significant. This approach is more expensive computationally, but can be implemented automatically to any smooth term in `R`. Table 4 and Figure 2-4 show the results of GAM with selection. We also use the same training set and test set in Example 1, and calculate the test error is 0.000803.

Table 3: Approximate significance of smooth terms without selection.

	Estimated df	Ref.df	F	p-value
f(au)	1.000	1.000	5.845	0.0157
f(c2o)	2.760	3.526	7.753	1.13e-05
f(us)	1.000	1.000	4.077	0.0436
f(sh)	2.989	3.857	2.258	0.0526

Table 4: Approximate significance of smooth terms with selection.

	Estimated df	Ref.df	F	p-value
f(au)	0.8903	9	0.505	0.00874
f(c2o)	4.4297	9	3.181	1.44e-06
f(us)	0.6573	9	0.332	0.01857
f(sh)	2.8179	9	0.969	0.01687

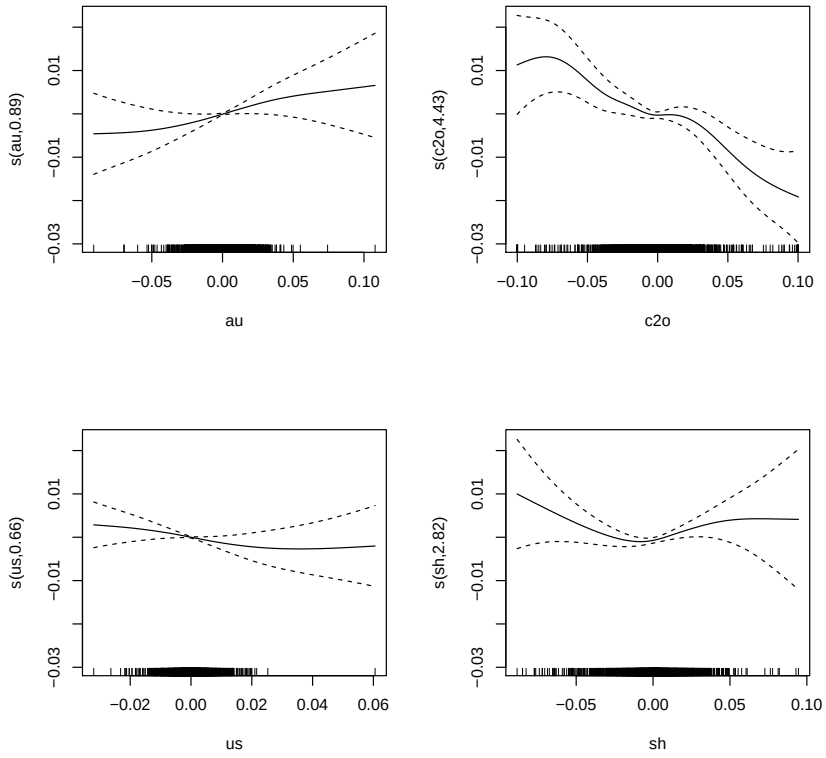

Figure 2: The fitted spline functions.

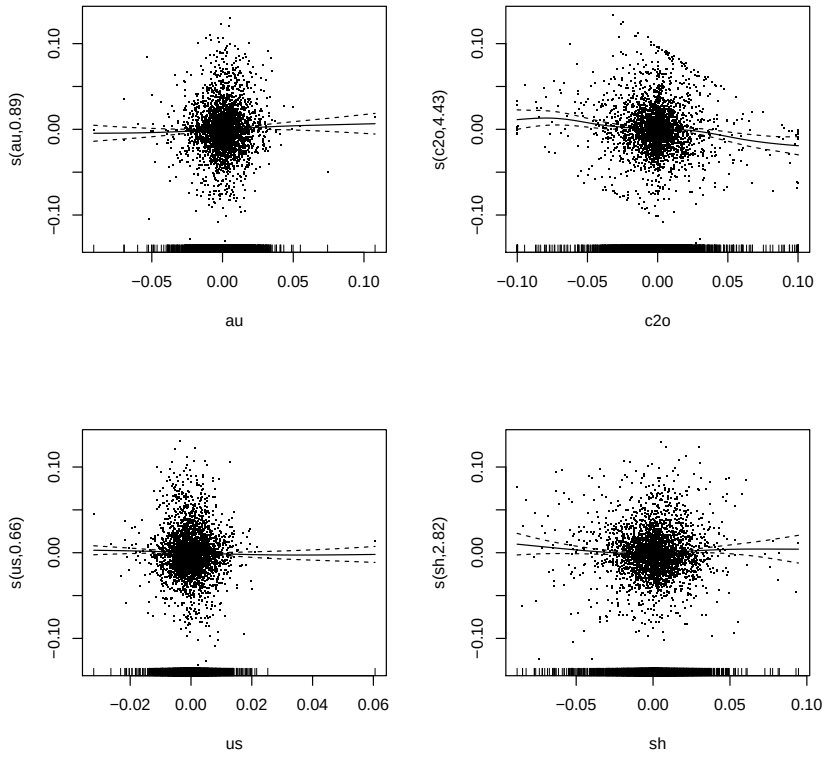

Figure 3: The residual of fitted spline functions.

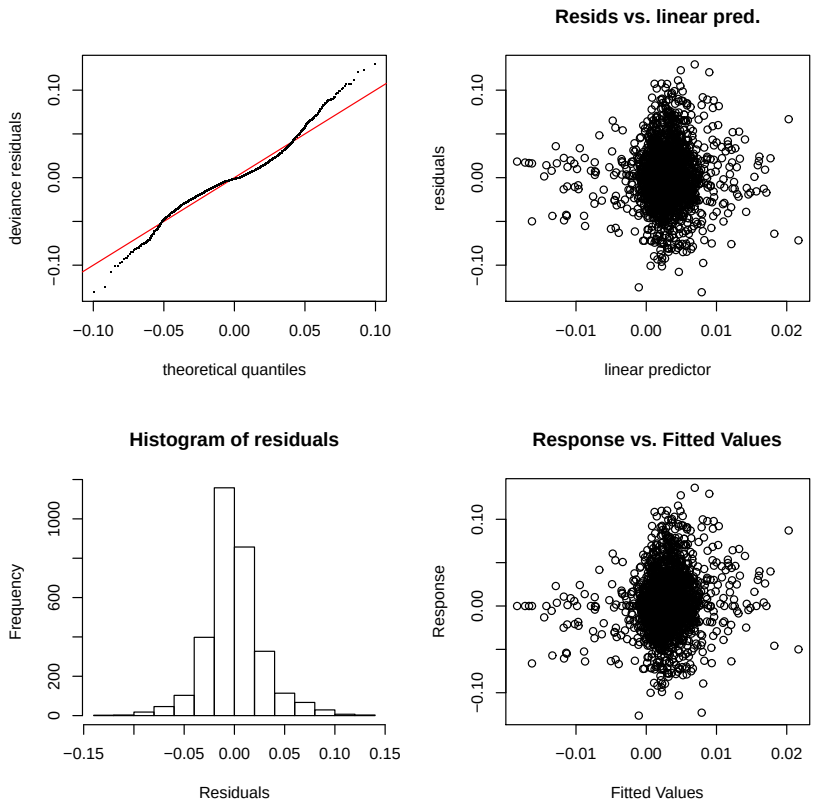

Figure 4: Result of fitness of GAM.