# Exploratory Data Analysis

## Tabular Data Analysis and Smoothing

David B King, Ph.D.

October 18, 2015

# Smoothing

- Let $\{(t_i, y_i)\}_{i=1}^{n}$ denote the $n$ ordered pairs of data points
- Want to fit the regression model

$$y_i = \mu(t_i) + \epsilon_i, \quad i = 1, \ldots, n,$$

with $\epsilon_i$ zero mean, uncorrelated random variables (the noise) and $\mu(\cdot)$ some **unknown function** of $t$.

1. Want $\mu(\cdot)$ to be a reasonably smooth function of $t$.
2. Want $\mu(\cdot)$ to conform to the local behavior of the data.
3. Don't necessarily have a shape predefined in advance like a line or parabola.
4. Smoothing is a very deep and advanced topic which deserves a semester course.

# Smoothing References

- Non=parametric Regression and Spline Smoothing *Eubank, Randal L.*
- Exploratory Data Analysis *Tukey, John W.*
- Spline Models for Observational Data *Wahba, Grace*
- Theoretical Foundations of Functional Data Analysis, with an Introduction to Linear Operators *Hsing, Tailen & Eubank, Randall*

# Two Flavors of Smoothing

Broadly speaking, there are two different flavors of smoothing:

1. **Local linear or kernel based estimators** of $\mu(\cdot)$ are based upon the estimator
$$\hat{\mu}_h(t) = \sum_{i=1}^{n} K(t, t_i, h) y_i$$
where $\{K(t, t_i, h)\}_{i=1}^{n}$ are a collection of kernel weight functions that determine the weights to use when fitting locally around the point $t_i$.
2. **The smoothing spline approach** seeks to find a function $\mu_\lambda(\cdot)$ which minimizes
$$n^{-1} \sum_{i=1}^{n} (y_i - f(t_i))^2 + \lambda \int_0^1 [f^{(m)}(t)]^2 dt$$
for some $\lambda > 0$. The first term in the expression above is a measure of fidelity to the data and the second is one which penalizes functions which are too "curvy".

# Kernel based smoother example

Let $\lambda$ be some positive integer and partition the interval $[0, 1]$ into $\lambda$ subintervals of the form $P_j = [\frac{j-1}{\lambda}, \frac{j}{\lambda}]$ for $j = 1, \ldots, \lambda$. Then if we use the "Boxcar" kernel we set

$$K(t, t_i, \lambda) = \frac{\sum_{r=1}^{\lambda} I_{P_r}(t) I_{P_r}(t_i)}{\sum_{j=1}^{n} \sum_{r=1}^{\lambda} I_{P_r}(t) I_{P_r}(t_i)}$$

with $I_{P_r}(\cdot)$ the indicator function for the interval $P_r$. The local linear regression estimator would be given by

$$\mu_\lambda(t) = \sum_{i=1}^{n} K(t, t_i, \lambda) y_i = \sum_{i=1}^{n} w_i y_i$$

# Risk or MSE

The performance of the estimator $\hat{\mu}_\lambda$ is measured by the average Risk or average MSE

$$R(\hat{\mu}) = n^{-1} \sum_{i=1}^{n} \mathrm{E}[\hat{\mu}(t_i) - \mu(t_i)]^2.$$

and if $R(\hat{\mu}) \to 0$ as $n \to \infty$ then $\hat{\mu}$ is MSE consistent with $\mu$.
Let $\lambda$ be some positive integer and partition the interval $[0, 1]$ into $\lambda$ subintervals of the form $P_j = [\frac{j-1}{\lambda}, \frac{j}{\lambda}]$ for $j = 1, \ldots, \lambda$. Then if we use the "Boxcar" kernel we set
Now if we take points evenly distributed on $[0, 1]$ with $t_i = (2i - 1)/2n$ then for any point $t \in P_j$

$$\mathrm{Var}(\hat{\mu}_\lambda(t)) = \frac{\sigma^2}{n_j} \text{ and,}$$

$$\mathrm{E}(\hat{\mu}_\lambda(t)) = \sum_{t_i \in P_j} \frac{\mu(t_i)}{n_j}$$

with $n_j = \sum_{i=1}^{n} I_{P_j}(t_i)$ the number of design points falling in the $j^{th}$ partition.

# Risk or MSE

The mean value theorem gives that $\mu(t_i) = \mu(t) + \mu'(\xi_{ij})(t_i - t)$ for some $\xi_{ij} \in P_j$. Thus for $t \in P_j$, $|\mathrm{E}\hat{\mu}_\lambda(t) - \mu(t)| \leq \lambda^{-1} \sup_{s\in[0,1]} |\mu'(s)|$ because $|t - t_i| \leq \lambda^{-1}$ for $t, t_i \in P_j$. Consequently the average risk for the regressogram is

$$\begin{aligned}
R(\lambda) &= n^{-1} \sum_{i=1}^{n} \mathrm{E}[\hat{\mu}_\lambda(t_i) - \mu(t_i)]^2 \\
&= n^{-1} \sum_{j=1}^{\lambda} \sum_{i:t_i \in P_j} [\mathrm{Var}(\hat{\mu}_\lambda(t_i)) + (\mathrm{E}\hat{\mu}_\lambda(t_i) - \mu(t_i))^2] \\
&= \frac{\lambda\sigma^2}{n} + \lambda^{-2}(\sup_{s\in[0,1]} |\mu'(s)|)^2
\end{aligned}$$

which converges to zero provided that $\lambda, n \to \infty$ with $\lambda/n \to 0$.
and if $R(\hat{\mu}) \to 0$ as $n \to \infty$ then $\hat{\mu}$ is MSE consistent with $\mu$.

# Optimal Interval Size

We can find the optimal interval size by minimizing $R(\lambda)$, hence

$$\frac{dR(\lambda)}{d\lambda} = 0 \Longrightarrow \frac{\sigma^2}{n} - 2\lambda^{-3}(\sup_{s\in[0,1]} |\mu'(s)|)^2 = 0$$

which implies that

$$\boxed{\lambda \propto n^{1/3} \text{ and } R(\lambda) \propto n^{-2/3}}$$

Say I think I see some similarities with non-parametric kernel density estimation!!

# Example

```
t=seq(0,1,length.out=100)
f=function(t){ t^3-3*t^2+3*t +1}
curve(f(x),0,1)
y=f(t)+rnorm(100,0,0.2)

plot(t,y)
curve(f(x),0,1,add=TRUE,col="red")

# Construct the Kernel Matrix
# Let's say we have 10 intervals where
# [0,0.1], [0.1,0.2], ..., [0.9,1]
lambda = 10
partition = seq(0,1,length.out=lambda)
K = function(s,t){ abs(s-t)<= 1/lambda}
Kmat = outer(t,t,FUN=K)
rsum= apply(Kmat,1,sum)
one=rep(1,100)

dmat=rsum %o% one
Kmat=Kmat/dmat

yhat=Kmat %*% y
plot(t,yhat)
plot(t,yhat,type="l",col="blue")
curve(f(x),0,1,add=TRUE,col="red")
```

# Performance Criteria

## Loss

Define the **loss** in estimating $\mu$ by

$$L(\lambda) = n^{-1} \sum_{i=1}^{n} (\mu(t_i) - \hat{\mu}_\lambda(t_i))^2.$$

## Risk

Define the **risk** as the expected loss

$$R(\lambda) = \mathrm{E}L(\lambda) = n^{-1} \sum_{i=1}^{n} \mathrm{E}(\mu(t_i) - \hat{\mu}_\lambda(t_i))^2.$$

The prediction for a new observation is $\mathbf{y}^* = \mathbf{y} + \epsilon^*$ hence

## Predictive Risk

define the **predictive risk** as the expected loss for new prediction

$$P(\lambda) = n^{-1} \sum_{i=1}^{n} \mathrm{E}(y_i^* - \hat{\mu}_\lambda(t_i))^2 = \sigma^2 + R(\lambda).$$

# Performance Criteria

## Integrated Loss

Define the **integrated loss** in estimating $\mu$ by

$$IL(\lambda) = \int_0^1 (\mu(t) - \hat{\mu}_\lambda(t))^2 dt.$$

## Integrated Risk

Define the **integrated risk** as the expected integrated loss

$$R(\lambda) = \mathrm{E} IL(\lambda) = \int_0^1 \mathrm{E}(\mu(t) - \hat{\mu}_\lambda(t))^2 dt.$$

# Performance Criteria

As we saw in our smoothing example the prediction at $(t_1, \ldots, t_n)$ could be written by

$$\hat{\mathbf{y}} = \mathbf{K}_\lambda \mathbf{y}$$

for the weight matrix $\mathbf{K}$. Hence the residual sum of squares (RSS) is given by

$$RSS(\lambda) = (\mathbf{y} - \hat{\boldsymbol{\mu}})^T(\mathbf{y} - \hat{\boldsymbol{\mu}}) = \mathbf{y}^T(\mathbf{I} - \mathbf{K}_\lambda)^2\mathbf{y}.$$

Hence one might estimate the average risk as

$$\begin{aligned} \mathrm{E}RSS(\lambda) &= \boldsymbol{\mu}^T(\mathbf{I} - \mathbf{K})^2\boldsymbol{\mu} + \sigma^2\mathrm{tr}[(\mathbf{I} - \mathbf{K}_\lambda)^2] \\ &= \boldsymbol{\mu}^T(\mathbf{I} - \mathbf{K})^2\boldsymbol{\mu} + n\sigma^2 + \sigma^2\mathrm{tr}[\mathbf{K}_\lambda^2] - 2\sigma^2\mathrm{tr}[\mathbf{K}_\lambda]. \end{aligned}$$

# Performance Criteria

However, in contrast

$$
\begin{aligned}
P(\lambda) &= \sigma^2 + R(\lambda) \\
&= \sigma^2 + n^{-1} \sum_{i=1}^{n} \mathrm{E}(mu(t_i) - \hat{mu}_\lambda(t_i))^2 \\
&= \sigma^2 + n^{-1} \mathrm{E}(\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}_\lambda)^T (\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}_\lambda) \\
&= \sigma^2 + n^{-1} \boldsymbol{\mu}^T (\mathbf{I} - \mathbf{K}_\lambda)^2 \boldsymbol{\mu} + n^{-1} \sigma^2 \mathrm{tr}[\mathbf{K}_\lambda^2].
\end{aligned}
$$

# Cross Validation Performance Criteria

One measure of performance is to strip away one data point at a time and measure how closely the model which is trained without data point $i$ predicts the value $y_i$ observed at $i$.

$$CV(\lambda) = n^{-1}\sum_{i=1}^{n}(y_i - \hat{\mu}_{\lambda(i)}(t_i))^2.$$

However it can be shown that

$$\hat{\mu}_{\lambda(i)}(t_i) = \hat{\mu}_{\lambda}(t_i) - k_{ii}(y_i - \hat{\mu}_{\lambda}(t_i))/(1 - k_{ii})$$

where $k_{ii}$ is the $i^{th}$ diagonal element of $\mathbf{K}_\lambda$. Another computationally nice method is by measuring the **generalized cross validation** given by

$$\boxed{GCV(\lambda) = n^{-1}RSS(\lambda)/(n^{-1}\mathrm{tr}[\mathbf{I} - \mathbf{K}_\lambda])^2.}$$

# Some Functional Space Theory

If $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ are two complex vectors the Euclidean (or $\ell^2$) inner product is given by

$$\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^* \mathbf{b} = \bar{\mathbf{a}}^T \mathbf{b} = \sum_{i=1}^{n} \bar{a}_i b_i$$

where $\bar{a}_i$ denotes the complex conjugate of $a_i$, and $\mathbf{a}^*$ denotes the adjoint or complex conjugate transpose of the vector $\mathbf{a}$.

If $f(t)$ and $g(t)$ are two square integrable complex functions on $[0, 1]$ then the $L^2[0, 1]$ inner product between $f(\cdot)$ and $g(\cdot)$ is given by

$$\langle f, g \rangle = \int_0^1 \bar{f}(t) g(t) dt$$

and $L^2[0, 1]$ consists of all functions where

$$\langle f, f \rangle = \|f\|^2 < \infty.$$

# Some Functional Space Theory

## Orthogonality

Two functions $\mu_1(\cdot), \mu_2(\cdot) \in L^2[0,1]$ are said to be orthogonal if $\langle \mu_1, \mu_2 \rangle = 0$. We denote this by $\mu_1 \perp \mu_2$.

## Orthonormality

A sequence of functions $\{\phi_i\}_{i=1}^{\infty}$ is said to be orthonormal if the $\phi_j$ are pairwise orthogonal and $\|\phi_i\| = 1$ for all $i$.

## Complete Orthonormal Sequence (CONS)

A sequence of functions $\{\phi_i\}_{i=1}^{\infty}$ is said to be a complete orthonormal sequence (CONS) if $f \perp \phi_i$ for all $i$ implies that $f = 0$ almost everywhere (a.e.).

# Fourier Basis Functions

There are three ways to construct a CONS for $L^2[0,1]$ using trigonometric functions

$$
\begin{aligned}
\phi_1(t) &= 1 \\
\phi_{2j}(t) &= \sqrt{2}\cos(2j\pi t) \text{ and,} \\
\phi_{2j+1}(t) &= \sqrt{2}\sin(2j\pi t)
\end{aligned}
$$

for $j = 1, 2 \ldots$, or

$$
\begin{aligned}
\phi_1(t) &= 1 \\
\phi_j(t) &= \sqrt{2}\cos((j-1)\pi t) \text{ for } j = 2, 3, \ldots
\end{aligned}
$$

or

$$
\begin{aligned}
\phi_1(t) &= 1 \\
\phi_j(t) &= \sqrt{2}\sin(j\pi t) \text{ for } j = 2, 3, \ldots.
\end{aligned}
$$

# Legendre Polynomials

Another CONS for $L^2[0,1]$ can be derived by applying the Gramm-Schmidt orthonormalization process to the basis of polynomial functions $q_i(t) = t^{j-1}, j = 1, 2, \ldots$. To construct this CONS one proceeds as follows. First take

$$\phi_i(t) = q_1(t)/\|q_1(t)\| \equiv 1.$$

Now define basis functions recursively via the formula

$$\phi_j(t) = \frac{\left[q_j(t) - \sum_{k=1}^{j-1} \langle q_j, \phi_k \rangle \phi_k(t)\right]}{\|q_j - \sum_{k=1}^{j-1} \langle q_j, \phi_k \rangle \phi_k\|}$$

# Best Approximate Function

## Proposition

Let $\{\phi_j\}_{j=1}^{\infty}$ be any CONS for $L^2[0,1]$ and for any $\mu \in L^2[0,1]$ define

$$\beta_j = \langle \mu, \phi_j \rangle, j = 1, 2, \ldots$$

Then $\sum_{j=1}^{\lambda} \beta_j \phi_j$ is the best approximation to $\mu$ in the sense that

$$\|\mu - \sum_{j=1}^{\lambda} \beta_j \phi_j\| \leq \|\mu - \sum_{j=1}^{\lambda} b_j \phi_j\|$$

for all $\mathbf{b} = (b_1, \ldots, b_\lambda) \in \mathbb{R}^\lambda$. Moreover as $\lambda \to \infty$

$$\|\mu - \sum_{j=1}^{\lambda} \beta_j \phi_j\|^2 \to 0$$

# Taylor's Theorem

## Taylor's Theorem

If $\mu \in W_2^m[0,1]$, then there exist coefficients $\theta_1, \ldots, \theta_m$ such that

$$\mu(t) = \sum_{j=1}^{m} \theta_j t^{j-1} + \int_0^1 \frac{(t-u)_+^{m-1}}{(m-1)!} u^{(m)}(u) du,$$

where

$$(x)_+^r = \begin{cases} x^r, & x \geq 0 \\ 0, & x < 0. \end{cases}$$

Proof: Write $\mu(t) = \int_0^1 (t-u)_+^0 \mu'(u) du + \mu(0)$ and integrate by parts. $\diamondsuit$

# Taylor's Theorem

Taylor's Theorem suggests that if, for some positive integer $\lambda$, the remainder term

$$Rem_\lambda(t) = [(\lambda - 1)!]^{-1} \int_0^1 (t-u)_+^{\lambda-1} \mu^{(\lambda)}(u)du$$

is uniformly small then we could write

$$y_i = \sum_{j=1}^{\lambda} \theta_j t^{j-1} + \epsilon_i \quad , i = 1, \ldots, n.$$

with $\epsilon_i$ uniformly small errors.

# Polynomial Regression

Let $\mathbf{Q}_\lambda$ denote the $(n\mathbf{y}\lambda)$ Vandermonde matrix

$$\mathbf{Q}_\lambda \equiv \{q_j(t_i)\}_{i=1,\ldots,n:j=1,\ldots,\lambda} = \begin{bmatrix} t_1^0 & t_1^1 & \cdots & t_1^{\lambda-1} \\ t_2^0 & t_2^1 & \cdots & t_2^{\lambda-1} \\ \vdots & \vdots & \cdots & \vdots \\ t_n^0 & t_n^1 & \cdots & t_n^{\lambda-1} \end{bmatrix}$$

Then the polynomial regression estimator of $\mu(t)$ is given by

$$\mu_\lambda(t) = (1, t, \ldots, t^{\lambda-1})(\mathbf{Q}_\lambda^T \mathbf{Q}_\lambda)^{-1} \mathbf{Q}_\lambda^T \mathbf{y}$$

However, $(\mathbf{Q}_\lambda^T \mathbf{Q}_\lambda)$ get's non-singular fast

# Polynomial Regression

So better to work with poly() basis functions in R which are defined by

$$x_{jn}(t) = \sum_{k=1}^{\lambda} a_{k\lambda} t^{k-1}$$

with $\mathbf{A}_\lambda = \{a_{jr}\}$ a nonsingular satisfying

$$\mathbf{A}_\lambda^T \mathbf{Q}_\lambda^T \mathbf{Q}_\lambda \mathbf{A}_\lambda = n\mathbf{I}$$

so that

$$n^{-1} \sum_{i=1}^{n} x_{jn}(t_i) x_{jn}(t_i) = \delta_{jk}$$

# Kernel Methods

Kernel functions often have the form

$$K(t, t_i, \lambda) = \frac{1}{\lambda} K\left(\frac{t - t_i}{\lambda}\right)$$

with the following moment conditions

$$\int_{-1}^{1} K(u)du = 1, \int_{-1}^{1} uK(u)du = 0$$
$$\int_{-1}^{1} u^2 K(u)du = M_2, \int_{-1}^{1} K^2(u)du = R < \infty.$$

| Kernel | $K$ | $R$ | $M_2$ |
|---|---|---|---|
| Uniform | $K(u) = \frac{1}{2} I_{[-1,1]}(u)$ | $\frac{1}{2}$ | $\frac{1}{3}$ |
| Quadratic | $K(u) = \frac{3}{4}(1 - u^2) I_{[-1,1]}(u)$ | $\frac{3}{5}$ | $\frac{1}{5}$ |
| Biweight | $K(u) = \frac{15}{16}(1 - u^2)^2 I_{[-1,1]}(u)$ | $\frac{5}{7}$ | $\frac{1}{7}$ |

# Local Linear Methods

Locally Linear estimator for $\mu(t)$ attempts to estimate through minimization of

$$\sum_{i=1}^{n} K\left(\frac{t-t_i}{\lambda}\right)(y_i-\theta_1-\theta_2(t_i-t))^2$$

The explicit form of the estimator is given by first defining

$$M_{jn}(t)=(\lambda n)^{-1}\sum_{i=1}^{n} K\left(\frac{t-t_i}{\lambda}\right)(t-t_j)^j, j=0,1,2.$$

Then

$$u_\lambda(t)=\sum_{i=1}^{n} y_i w(t,t_i,\lambda)$$

with

$$w(t,t_i,\lambda)=\frac{1}{n\lambda}K\left(\frac{t-t_i}{\lambda}\right)\frac{M_{2n}(t)-(\frac{t-t_i}{\lambda})M_{1n}(t)}{M_{2n}(t)M_{0n}(t)-M_{1n}^2(t)}.$$

# LOWESS Smoother

```
Scatter plot smoothing

lowess(x, y = NULL, f = 2/3, iter = 3,
              delta = 0.01 * diff(range(x)))
```

This function performs the computations for the LOWESS smoother which uses locally-weighted polynomial regression.

# Linear smoother vs Nonlinear smoothers

Problems with linear smoothers:

- Smooth over sharp features
- Strongly affected by outliers

Nonlinear smoothers:

- cannot be expressed as $\sum w_i y_i$
- flexible (no linear constraints)
- usually involve medians instead of means
- catch depths of troughs, heights of peaks
- reduce influence of outliers
- easy to do by hand

# Tukey's Running median Smoothing

$$smooth(x, kind = c(\text{"3RS3R"}, \text{"3RSS"}, \text{"3RSR"}, \text{"3R"}, \text{"3"}, \text{"S"}),\\ twiceit = FALSE, endrule = \text{"Tukey"}, do.ends = FALSE)$$

```
xx <- rnorm(20); xx

0.98 0.54 -0.75 0.34 0.88 0.48 0.08 -0.43 -0.57 0.68
1.56 -0.58 0.22 0.71 0.71 -0.15 1.29 0.64 1.17 1.85

#Smooth by 3:
xx3 <- smooth(xx, kind="3")
3 Tukey smoother resulting from smooth(x=xx, kind="3")
used 1 iterations

0.94 0.54 0.34 0.34 0.48 0.48 0.08 -0.43 -0.43 0.68
0.68 0.22 0.22 0.71 0.71 0.71 0.64 1.17 1.17 1.17
```

# Tukey's Running median Smoothing

Some specific problems with $3R$:

- Plateaus (hence "splitting")

  An artifact of $3R$ is the presence of two adjacent smoothed $y$'s with the same value. "Split" between them and apply end-value rule to each one so the values will differ.

# Tukey's Running median Smoothing

Some specific problems with $3R$:

- "End value rules": Construct $y_0, y_{n+1}$, Tukey EDA, p221
    - "the change from the end smoothed value to the next-to-end smoothed value is between 0 and $+2$ times the change from the next-to-end smoothed-value to the next-to-end-but-one smoothed value."
    - "subject to this being true, the end smoothed-value is as close to the end input-value as possible."

"This means that we can look at two differences:

end input-value MINUS next-to-end smoothed value

and

next-to-end smoothed value MINUS next-but-one-to-end smoothed value

# Tukey's Running median Smoothing

Some specific problems with $3R$:

- "End value rules": Construct $y_0, y_{n+1}$, Tukey EDA, p221
    - 

  and if the first is between 0 and $+2$ times the second, we can copy on. Otherwise, we can make

  end smoothed value MINUS next-to-end smoothed value

  either zero or two times

  next-to-end smoothed value MINUS next-but-one-to-end smoothed value."

  $$\tilde{y}_1 = \text{median}\{y_1, \tilde{y}_2, y_0\}$$

  where $y_0 = 3\tilde{y}_2 - 2\tilde{y}_3$ is a linear extrapolation of $y_3$ and $y_2$ to $x_0$ as if $x_0, x_1, x_2, x_3$ are equally spaced.

# Tukey's Running median Smoothing

Some specific problems with $3R$:

- "Twicing": smooth the residuals and add back to the original smooth

# Tukey's Running median Smoothing

3RS3R, 3RSS, 3RSR (3RSSHT)

- 3 = smooth by medians of length 3
- R = repeat the previous smooth until no change
- S = split 2-plateaus
- H = hanning (1/4, 1/2, 1/4)
- T = twice (repeat on residuals)

*smoothEnds(y, k=3)*: apply end-value rule to ends only

*runmed*: apply end-value rule also:

*runmed(x, k, endrule = c("median", "keep", "constant"),*
*algorithm = NULL, print.level = 0)*

# Extensions:

Extending to several variables: *Backfitting*

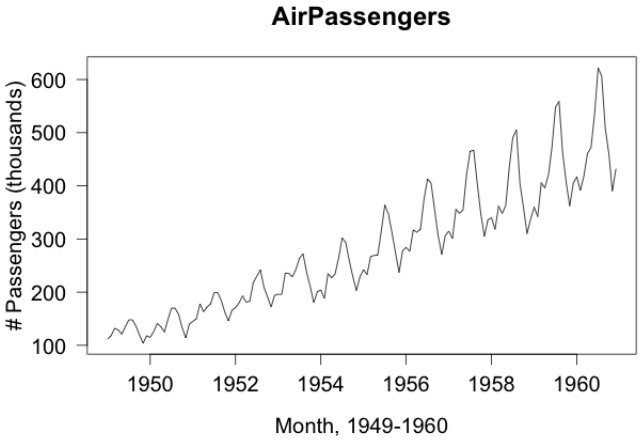
$$\tilde{y} = f(x_1) + f(x_2)$$

1. Initial smooth: fit $y$ as a linear function of $x_1$, $x_2$
2. Iterate: smooth residuals as a function of $x_i$
3. Repeat step (2) for each $x_i$, in turn, until residuals are "flat"

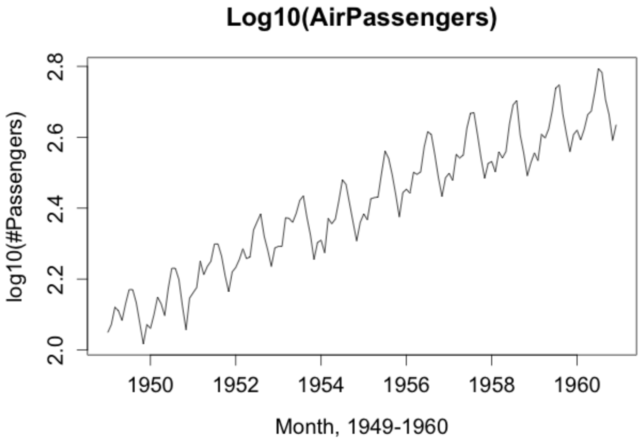
See Hastie and Tibshirani (1993), *Generalized Additive Models*.

# Example

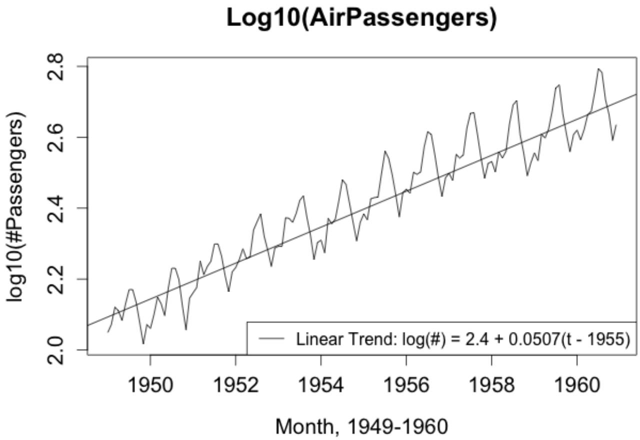
The classic Box & Jenkins airline data. Monthly totals of international airline passengers, 1949 to 1960.



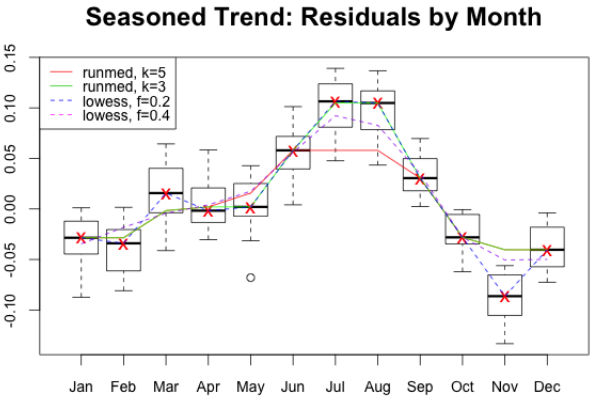
# Example


Log10(AirPassengers)
log10(#Passengers)
2.0
2.2
2.4
2.6
2.8
1950
1952
1954
1956
1958
1960
Month, 1949-1960

# Example


Log10(AirPassengers)
log10(#Passengers)
2.0
2.2
2.4
2.6
2.8
1950
1952
1954
1956
1958
1960
Month, 1949-1960
Linear Trend: log(#) = 2.4 + 0.0507(t - 1955)

# Example


Seasoned Trend: Residuals by Month
runmed, k=5
runmed, k=3
lowess, f=0.2
lowess, f=0.4
-0.10
-0.05
0.00
0.05
0.10
0.15
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

# Summary

R functions for smoothing: help.search("smooth")

1. Basic library:
   - ksmooth: Kernel regression smoother
   - lowess: Scatter plot smoothing
   - smooth.spline: Fits a cubic smoothing spline to the supplied data.
   - predict.smooth.spline: predict from spline smooth
   - runmed: running medians
   - scatter.smooth: Plot and add a smooth curve computed by loess to a scatter plot.
   - smooth: Tukey's running median smoothing
   - smoothEnds: end-value smoothing for running medians
   - supsmu: Friedman's SuperSmoother
   - pspline: Smoothing splines using a pspline basis

# Summary

R functions for smoothing: help.search("smooth")

2. library("graphics"):
   - panel.smooth: Simple Panel Plot
   - smoothScatter: Scatterplots with Smoothed Densities Color Representation
   - smooth: Tukey's running median smoothing
   - smoothEnds: end-value smoothing for running medians
   - runmed: running medians

# Introduction to Splines

The origin of splines began as an attempt to solve the interpolation problem by fitting a smooth curve through data.

## Lagrange Interpolation

For a given set of data points $\{(t_i, y_i)\}_{i=1}^{n}$ the **Lagrange** basis functions for the set of polynomials of order $n-1$, $\mathbb{P}_{n-1}$, is given by

$$\ell_j(t) = \frac{\prod_{k=1,k\neq j}^{n}(t - t_k)}{\prod_{k=1,k\neq j}^{n}(t_j - t_k)} \quad j = 1, \ldots, n.$$

For the definition, we see that $\ell_j(t_i)$ is a polynomial of degree $n-1$ and

$$\ell_j(t_i) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, i,j = 1, \ldots, n.$$

# Lagrange Interpolation

The design matrix $\mathbf{X}$ for the **Lagrange** polynomial basis functions is:

$$\mathbf{X} = \begin{bmatrix} \ell_1(t_1) & \ell_2(t_1) & \cdots & \ell_n(t_1) \\ \ell_1(t_2) & \ell_2(t_2) & \cdots & \ell_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \ell_1(t_n) & \ell_2(t_n) & \cdots & \ell_n(t_n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Hence, the polynomial which interpolates through the points $\{(t_i, y_i)\}_{i=1}^{n}$ satisfies

$$\mathbf{X}\beta = \mathbf{y} \Longrightarrow \mathbf{I}\beta = \mathbf{y} \Longrightarrow \beta = \mathbf{y}$$

and so the interpolating polynomial is given by

$$p_{n-1}(t)y_1\ell_1(t) + y_2\ell_2(t) + \cdots + y_n\ell_n(t).$$

# Lagrange Interpolation Example

Suppose we know the values of three data points without error $\{(t_1, y_1), (t_2, y_2), (t_3, y_3)\}$. The Lagrange interpolating polynomial will be

$$p_2(t) = y_1 \frac{(t-t_2)(t-t_3)}{(t_1-t_2)(t_1-t_3)} + y_2 \frac{(t-t_1)(t-t_3)}{(t_2-t_1)(t_2-t_3)} + y_3 \frac{(t-t_1)(t-t_2)}{(t_3-t_1)(t_3-t_2)}.$$

- Notice that $p(t_i) = y_i$ so the function is continuous and agrees with data.
- In general, $p_{n-1}(t)$ is the unique $(n-1)^{th}$ order polynomial that agrees with $y_i$ at $t_i$.
- However we want to construct a basis whose derivatives at the the points $\{t_1, t_2, t_3\}$ will be continuous as well.

# The Divided Difference

## The Divided Difference

The $q^{th}$ order divided difference of a function $g$ at $[t_i, \ldots, t_{i+q}]$ is

$$[t_i, \ldots, t_{i+q}]g = \frac{[t_{i+1}, \ldots t_{i+q}]g - [t_i, \ldots, t_{i+q-1}]g}{t_{i+q} - t_i}$$

with $[t_i]g = g(t_i)$ being used to initiate the recursion.

For example, the divided difference

$$[t_i, t_{i+1}]g = \frac{g(t_{i+1}) - g(t_i)}{t_{i+1} - t_i} \text{ or}$$

$$[t_{i-1}, t_{i+1}]g = \frac{g(t_{i+1}) - g(t_{i-1})}{t_{i+1} - t_{i-1}}.$$

can be used to approximate $g'(t_i)$.

# Lagrange Interpolation Theorem

## Lagrange Interpolation Theorem

Let $p_q(t) = \sum_{i=1}^{q} g(t_i)\ell_i(t)$ for

$$\ell_j(t) = \prod_{\substack{i=1 \\ i \neq j}}^{q} \frac{(t - t_i)}{(t_j - t_i)}.$$

Then,

1. $p_q$ is the unique $(q-1)^{th}$ order polynomial that agrees with $g$ at $t_i$ for $i = 1, \ldots, q$ and
2. for each $q = 1, 2, \ldots$ the coefficient $t^q$ corresponding to $g(t_i)$ in $p_{q+1}$ is $[t_i, \ldots, t_{i+q}]g$.

# Lagrange Interpolation Theorem Proof

Proof: The function $\ell_j(t)$ is a polynomial of order $q$ and vanishes at all the $t_i$ except for $t_j$ where it takes the value 1. So property (1) holds.
To verify the second property we proceed by induction. For $q = 1$, the coefficient corresponding to $g(t_i)$ in $p_2$ is either $\ell_i(t) = \frac{(t-t_1)}{(t_2-t_1)}$ for $i = 1$ or $\ell_2(t) = \frac{(t-t_2)}{(t_1-t_2)}$ for $i = 2$ so the coefficient of $t^1$ is $[t_i]g = g(t_i)$ for $i = 1, 2$. For the induction step, let $p_q(t)$ be the polynomial of order $(q-1)$ that agrees with $g$ at $t_i, \ldots, t_{i+q-1}$ and take $\tilde{p}_q(t)$ to be the polynomial of order $(q-1)$ that agrees with $g$ at $t_{i+1}, \ldots, t_{i+q}$ Then,

$$p(t) = \frac{(t - t_i)}{(t_{i+q} - t_i)}\tilde{p}_q(t) + \frac{(t_{i+q} - t)}{(t_{i+q} - t_i)}p_q(t)$$

is a polynomial of order $q$ and since $p(t_i) = p_q(t_i)$ and $p(t_{i+q}) = \tilde{p}_q(t_{i+q})$, it agrees with $g$ at $t_i, \ldots t_{i+q}$. By uniqueness of Lagrange interpolating polynomials, we must have $p_{q+1}(t) = p(t)$ and the coefficient of $t^q$ corresponding to $g(t_i)$ is

$$\frac{[t_{i+1}, \ldots, t_{i+q}]g - [t_i, \ldots, t_{i+q-1}]g}{(t_{i+q} - t_i)} \equiv [t_i, \ldots, t_{i+q}]g.$$

$\diamondsuit$

# Lagrange Corollary

## Corollary

If $g$ is a polynomial of order $q-1$ on $[t_i, t_{i+q}]$ then $[t_i, \ldots t_{i+q}]g = 0$.

Proof: Let $p_k(t) = \sum_{i=1}^{k} g(t_i)\ell_i(t)$ and note that $p_k = g$ for all $k \geq q-1$. Then, as $[t_i, \ldots, t_{i+q}]g$ is the lead coefficient of $t^q$ in $p_{q+1}$ in front of $g(t_i)$, and $g$ has degree one less than $q$ that coefficient must be zero, hence $[t_i, \ldots t_{i+q}]g = 0$.

# Smoothing Splines

## Splines

A spline of order $q \geq 1$, with knots at $0 < t_1 < t_2 < \cdots t_J < 1$ is any function of the form

$$g(t) = \sum_{i=0}^{q-1} \theta_i t^i + \sum_{j=0}^{J} \delta_j (t - t_j)_+^{q-1}$$

for constants $\theta_0, \ldots \theta_{q-1}, \delta_1, \ldots, \delta_J \in \mathbb{R}$.

# What is Special About the Function $(t-s)_+^p$?

Consider the inverse to the $p^{th}$ derivative operator

$$\frac{\partial^{p+1}}{\partial t^{p+1}} f(t) = (Lf)(t) = g(t).$$

subject to boundary conditions
$f^{(p)}(0) = f^{(p-1)}(0) = \cdots f^{(1)}(0) = f(0) = 0$. What to find inverse operator $L^{-1}$ such that

$$f(t) = (L^{-1}g)(t).$$

Consider the equation

$$\int_0^1 (t-s)_+^p f^{(p+1)}(s)ds = \int_0^t (t-s)^p f^{(p+1)}(s)ds$$

# What is Special About the Function $(t-s)_+^p$?

If we integrate by parts tabularly we find that

| Derivative | Integral | Sign |
|---|---|---|
| $(t-s)^p$ | $f^{(p+1)}(s)$ | |
| $(-1)^1 p(t-s)^{p-1}$ | $f^{(p)}(s)$ | $+$ |
| $(-1)^2 p(p-1)(t-s)^{p-2}$ | $f^{(p-1)}(s)$ | $-$ |
| $\vdots$ | $\vdots$ | $\vdots$ |
| $(-1)^{p-1}[p(p-1)\cdots(2)](t-s)^1$ | $f^{(2)}(s)$ | $(-1)^{p-2}$ |
| $(-1)^p p!$ | $f^{(1)}(s)$ | $(-1)^{p-1}$ |
| $0$ | $f(s)$ | $(-1)^p$ |

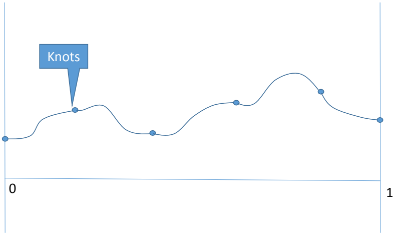
$$\begin{aligned}\int_0^1 (t-s)_+^p f^{(p+1)}(s)ds &= \int_0^t (t-s)^p f^{(p+1)}(s)ds \\ &= (t-s)^p f^{(p)}(s)|_0^t + p(t-s)^{p-1} f^{(p-1)}(s)|_0^t + \cdots + p! f(s)|_0^t \\ &= 0 + 0 + \cdots + p! f(t).\end{aligned}$$

Hence,

$$\frac{\partial^{p+1}}{\partial t^{p+1}} f(t) = (Lf)(t) = g(t) \Longrightarrow f(t) = (L^{-1}g)(t) = \frac{1}{p!}\int_0^1 (t-s)_+^p g(s)ds.$$

# Splines

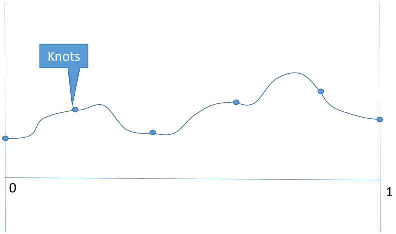
$$s(t) = \sum_{i=0}^{q-1} \theta_i t^i + \sum_{j=0}^{J} \delta_j (t - t_j)_+^{q-1}$$




1. $s(t)$ is a piecewise polynomial of order $q-1$ on any subinterval $[t_i, t_{i+1}[$
2. $s(t)$ has $q-2$ continuous derivatives and
3. $s(\cdot)$ has a discontinuous $(r-1)^{st}$ derivative with jumps at $t_1, t_2, \ldots, t_J$.

# Splines

$$s(t) = \sum_{i=0}^{q-1} \theta_i t^i + \sum_{j=0}^{J} \delta_j (t - t_j)_+^{q-1}$$




1. $s(t)$ is a piecewise polynomial of order $q-1$ on any subinterval $[t_i, t_{i+1}[$
2. $s(t)$ has $q-2$ continuous derivatives and
3. $s(\cdot)$ has a discontinuous $(r-1)^{st}$ derivative with jumps at $t_1, t_2, \ldots, t_J$.

# Spline Subspace

Let $\zeta_1, \ldots \zeta_k$ denote the interior knots inside the interval $[0, 1]$ and let $S^q(\zeta_1, \ldots \zeta_k)$ denote the set of functions of the form

$$s(t) = \sum_{i=0}^{q-1} \theta_i t^i + \sum_{j=1}^{k} \eta_j (t - \zeta_j)_+^{q-1}$$

Then $S^q(\zeta_1, \ldots, \zeta_k)$ is a vector space in the sense that the functions

$$1, t, \ldots, t^{q-1}, (t - \zeta_1)_+^{q-1}, \ldots, (t - \zeta_k)_+^{q-1}$$

are linearly independent and linear combinations of these functions remain in this set. It follows that $S^q(\zeta_1, \ldots \zeta_k)$ has dimension $k + r$.

# Spline Subspace

The terms $\sum_{j=0}^{J} \delta_j (t - \zeta_j)_+^{q-1}$ are important because they enable us to put continuity constraints on the $\{(1), (2), \ldots, (q-2)\}$ derivatives at the knots $t = \zeta_1, \zeta_2, \ldots, \zeta_k$ and ensure that

$$\lim_{t \to \zeta_i^+} g^{(q-2)}(t) = g^{(q-2)}(\zeta_i^+) = g^{(q-2)}(\zeta_i^-) = \lim_{t \to \zeta_i^+} g^{(q-2)}(t)$$

$$\lim_{t \to \zeta_i^+} g^{(q-3)}(t) = g^{(q-3)}(\zeta_i^+) = g^{(q-3)}(\zeta_i^-) = \lim_{t \to \zeta_i^+} g^{(q-3)}(t)$$

$$\vdots = \vdots = \vdots = \vdots$$

$$\lim_{t \to \zeta_i^+} g^{(1)}(t) = g^{(1)}(\zeta_i^+) = g^{(1)}(\zeta_i^-) = \lim_{t \to \zeta_i^+} g^{(1)}(t)$$

$$\lim_{t \to \zeta_i^+} g(t) = g(\zeta_i^+) = g(\zeta_i^-) = \lim_{t \to \zeta_i^+} g(t)$$

# Natural Smoothing Spline Subspace

Of particular importance is the set of natural splines of order $r = 2m$ with $k = n$ knots at the design points.

## Natural Splines

A spline function is a *natural spline* of order $2m$ with knots at $\zeta_1, \ldots, \zeta_k$ if in addition to properties (1-3) above it satisfies

4. $s(t)$ is a polynomial of order $m$ outside of the interior knots $[\zeta_1, \zeta_n]$

- Let $NS^{2m}(\zeta_1, \ldots, \zeta_n)$ denote the collection of all natural splines or order $2m$ with knots at $\zeta_1, \ldots, \zeta_n$.
- $NS^{2m}(\zeta_1, \ldots, \zeta_n)$ is a subspace of $S^{2m}(\zeta_1, \ldots, \zeta_n)$ obtained by placing $2m$ restrictions on the coefficients.
- One can show that $\theta_m = \cdots \theta_{2m-1} = 0$ and $NS^{2m}(\zeta_1, \ldots, \zeta_n)$ has dimension $m$.
- The argument is that $S^{2m}(\zeta_1, \ldots, \zeta_n)$ has dimension $2m + n$ and since we have $2m$ constraints $NS^{2m}(\zeta_1, \ldots, \zeta_n)$ has dimension $2m + n - 2m = n$.

# B-Splines

Let $0 < \zeta_1 < \zeta_2 < \cdots < \zeta_k < 1$ be a sequence of interior knots. We define the B-spline basis functions iteratively by first letting the $1^{st}$ order (or $0^{th}$ degree) B-spline basis to be

$$B_{i1}(t) \equiv \begin{cases} 1 & \text{if } \zeta_i < t < \zeta_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

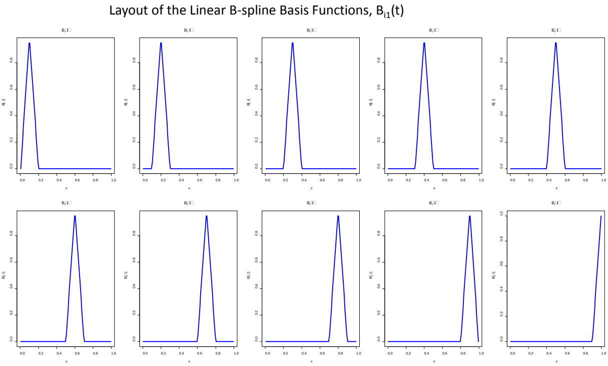
Next, for $k > 1$ we let

$$B_{ik}(t) \equiv \frac{(t - \zeta_i)}{(\zeta_{i+k-1} - \zeta_i)} B_{i,k-1}(t) + \frac{(\zeta_{i+k} - t)}{(\zeta_{i+k} - \zeta_{i+1})} B_{i+1,k-1}(t).$$

# B-Splines

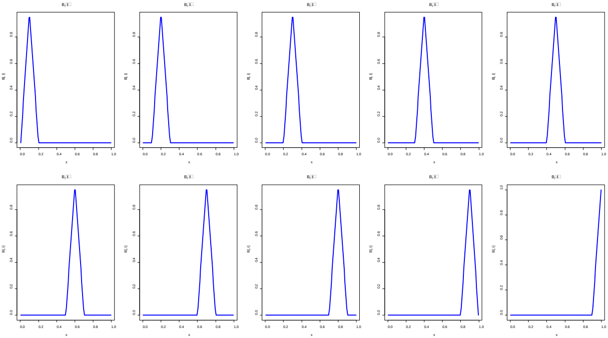
```
library("splines")
library("fda")
?bs
?smooth.splines
?ns
k=100
x=(c(1:k)-0.5)/k
print(x)
knots=c(1:9)/10
print(knots)
B=bs(x,knots=knots,degree=1)
plotfun = function(x,B){
p = ncol(B)
par(mfrow=c(2,ceiling(p/2)))
for(i in 1:p){
plot(x,B[,i],type="l",lwd=3,col="blue",main=expression(B[i,1](t)) )
}
}
# Plot the Order degree=1 B-spline basis
plotfun(x,B)
```

# Linear B-Spline Basis



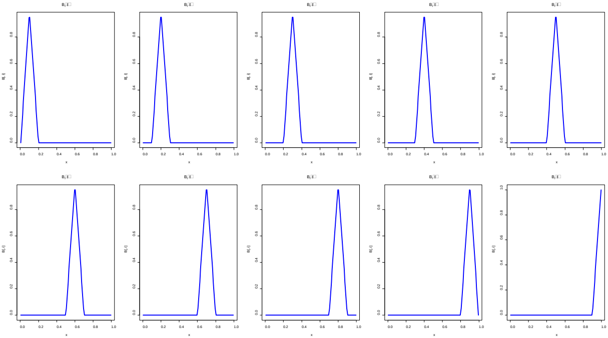
# Quadratic B-Spline Basis

Layout of the Linear B-spline Basis Functions, $B_{i1}(t)$



# Cubic B-Spline Basis

Layout of the Linear B-spline Basis Functions, $B_{i1}(t)$



# B-Spline Basis Theorem

- The knots $0 < \zeta_1 < \cdots < \zeta_k < 1$ are called the *interior knots*.
- When we construct the B-spline basis functions it is often useful to take *boundary knots* $\zeta_0 = 0$ and $\zeta_{k+1} = 1$.
- According to the Ansolone–Laurent–Reinsch procedure we then define an additional $2(q-1)$ *phantom knots*

$$\zeta_{-(q-1)} < \zeta_{-(q-1)} < \cdots < t_{-1} \leq 0 \text{ and,}$$
$$1 \leq \zeta_{k+2} < \zeta_{k+3} < \cdots < \zeta_{k+q}.$$

## B-Spline Basis Theorem

The collection of functions $\{B_{iq}(\cdot)\}_{i=-(q-1)}^{k+q}$ satisfy

1. $B_{iq}(\cdot)$ is a polynomial of order $q$ on each interval $(\zeta_i, \zeta i + 1)$.
2. $B_{iq}(\cdot) = 0$ for $t \notin [\zeta_i, \zeta_{i+q}]$ and,
3. $B_{iq}(t) = (\zeta_{i+q} - \zeta_i)[\zeta_i, \ldots, \zeta_{i+q}](\cdot - t)_+^{q-1} = \sum_{r=0}^{q} \alpha_{rq}^{[i]}(t_{i+r} - t)_+^{q-1}$ with $\alpha_{rq}^{[i]}$ such that

$$(\zeta_{i+q} - \zeta_i)[\zeta_i, \ldots, \zeta_{i+q}]g = \sum_{r=0}^{q} \alpha_{rq}^{[i]} g(t_{i+r})$$