Nonparametric Regression and Generalized Additive Models (GAM)

Bin Li

IIT Lecture Series

Nonparametric regression

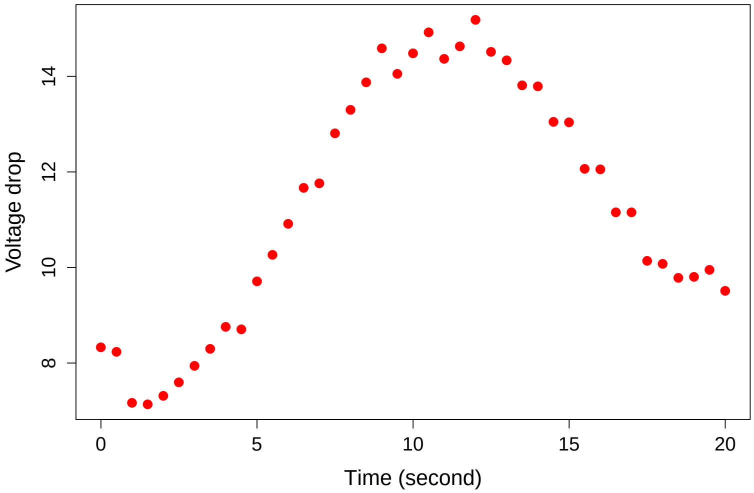
- In the traditional regression analysis, the form of the regression function has been specified. For example, we might use linear model:

  $$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$$

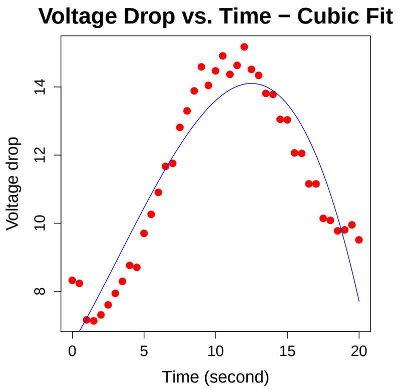
  a cubic polynomial.
- However in some situations we don't have enough information to make an assumption like this, or we don't want to. Instead we might want to only assume: $y = f(x) + \epsilon$
  with some smoothness assumptions on $f(x)$, such as
  - the continuity of the regression function $f(x)$ and its derivatives $f\prime(x)$, $f\prime\prime(x)$.
- So we would like to find a *nonparametric* estimate of $f(x)$ based on the data.

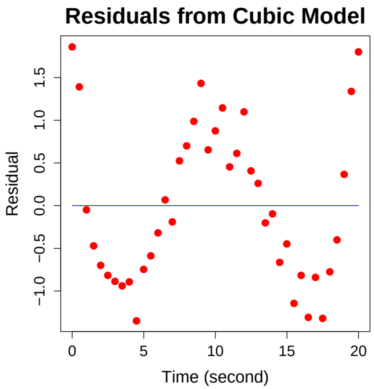
Battery voltage drop example

The battery voltage drop in a guided missile motor was observed over the first 20 seconds of the launch. The data was collected to develop a digital-analog simulation model of the missile.



Cubic fit of battery data





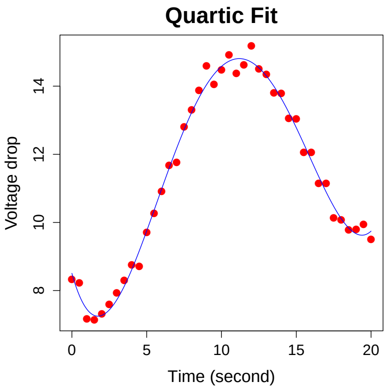
Both the plots of the fits and the residuals from the cubic model suggests that this is not a good model.

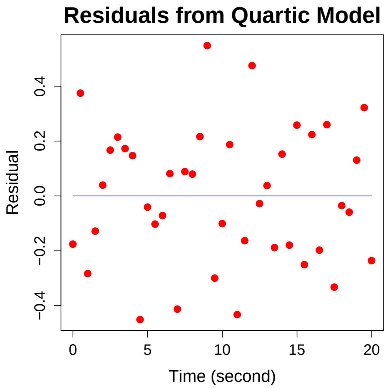
Quartic fit of battery data

However the quartic model

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_3 x^4 + \epsilon$$

seems reasonable based on the the following two plots.





Piecewise polynomials

- Instead of using the quartic model, we can use piecewise polynomial function $f(x)$ by dividing the domain of X into contiguous intervals, and representing f by a separate polynomial in each interval.
- The top right panel on next slides shows a piecewise linear fit which needs six parameters.
- Usually we would prefer the lower left panel, which is also piecewise linear, but restricted to be continuous at the two *knots*. In this case, since there are two restrictions, we only needs four parameters.
- A more direct way to proceed in this case is to use a basis that incorporates the constraints:

$$h_1(X) = 1,\ h_2(X) = X,\ h_3(X) = (X - \xi_1)_+,\ h_4(X) = (X - \xi_2)_+$$

where subscript $+$ denotes the positive part.

Piecewise polynomials (cont.)

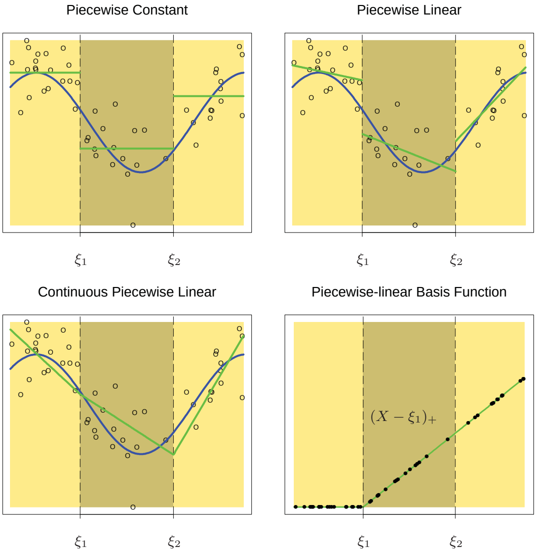


Figure from HTF 2009.

Piecewise polynomials (cont.)

- We often prefer smoother functions, and these can be achieved by increasing the order of the local polynomial.
- Figure on the next slide shows a series of piecewise-cubic polynomials fit to the same data with increasing orders of continuity at the knots.
- The function in the lower right panel is continuous, and has continuous first and second derivatives at the knots. It is known as a *cubic spline*.
- Note cubic spline is the lowest order of splines with continuous first and second derivatives. It is claimed that cubic splines are the lowest-order spline for which the knot-discontinuity is not visible to the human eye.
- We can show that the function in lower right panel on next slides has the following six basis:

$$\begin{aligned} h_1(X) &= 1, & h_3(X) &= X^2, & h_5(X) &= (X-\xi_1)^3_+ \\ h_2(X) &= X, & h_4(X) &= X^3, & h_6(X) &= (X-\xi_2)^3_+ \end{aligned}$$

Piecewise cubic polynomials

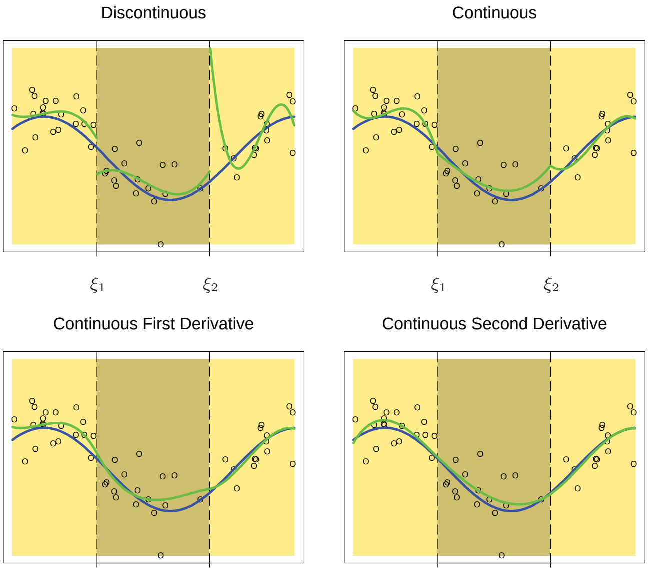
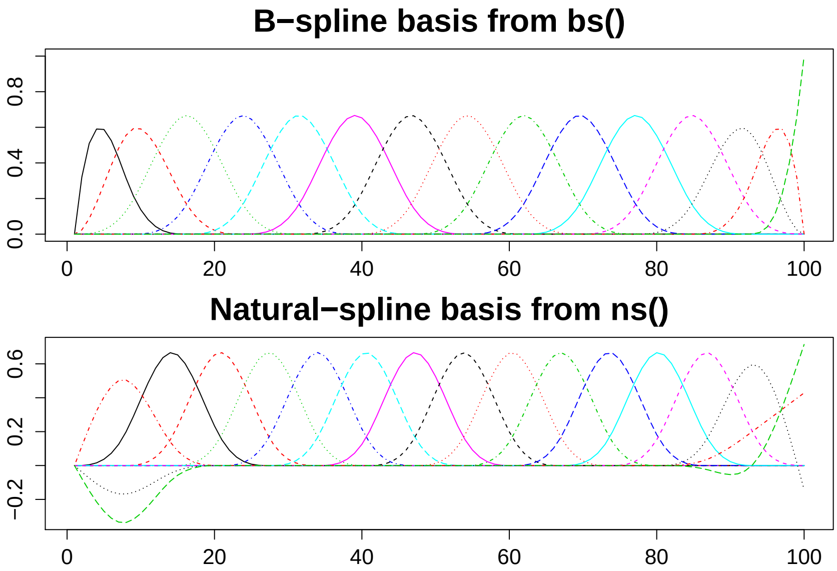


Figure from HTF 2009.

B-spline and natural spline basis



B-splines

- These fixed-knot splines are also known as *regression splines*.
- In practice, one needs to select the order of the spline, the number of knots and their placement.
- One simple approach is to parameterize a family of splines by the number of basis functions or degrees of freedom, and have the observations x_i determine the positions of the knots.
- For example: `bs(x,df=7)` in `R` generates a basis matrix of cubic-spline functions evaluated at the N observations in X, with the $7 - 3 = 4$ interior knots (`ns()` function omits by default the constant term in the basis) at the appropriate percentiles of X (20, 40, 60 and 80th.)
- One can be more explicit, `bs(x, degree=1, knots = c(0.2, 0.4, 0.6))` generates a basis for linear splines, with three interior knots.
- The B-spline basis allows for efficient computations even when the number of knots is large.

Natural splines

- The behavior of polynomials fit to data tends to be erratic near the boundaries, and extrapolation can be dangerous.
- Extrapolation problem is even worse with splines. The polynomials fit beyond the boundary knots behave more wildly than the corresponding global polynomials in that region (see the figure in next slide).
- A *natural cubic spline* adds additional constraints, namely that the function is linear beyond the boundary knots. This frees up four d.f. (two constraints each in both boundary regions).
- Natural splines can spent more profitably by sprinkling more knots in the interior region.
- The price paid in bias near the boundaries, but assuming the function is linear near the boundaries (where we have less information anyway) is often considered reasonable.
- A natural cubic spline with K knots is represented by K basis functions.

Pointwise variance curves for four different models

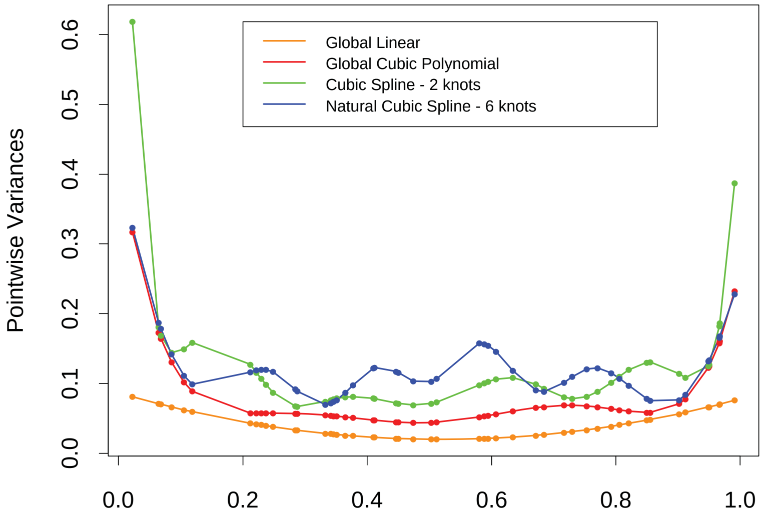
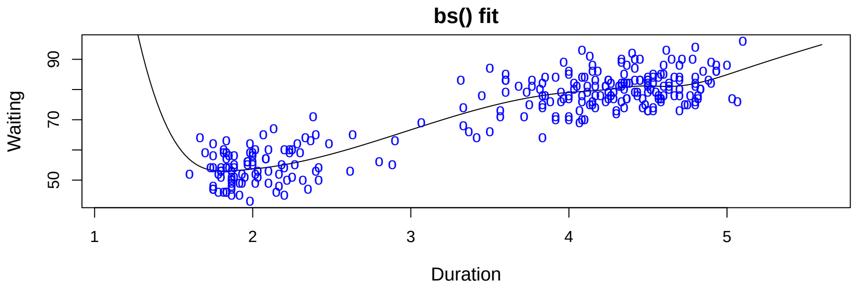
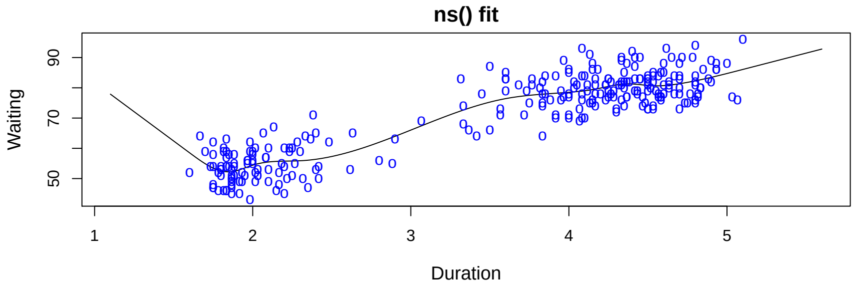


Figure from HTF 2009.

Old Faithful geyser example





R code

```
> fit1 <- lm(waiting~ns(eruptions,df=10),data=faithful)
> pred1 <- predict(fit1,newdata=data)
> plot(x,pred1,type="l",xlab="Duration",ylab="Waiting",
+      xlim=range(x),ylim=range(c(pred1,waiting)))
> points(eruptions,waiting,col="blue",pch="o")
> fit2 <- lm(waiting~bs(eruptions,df=10),data=faithful)
> pred2 <- predict(fit2,newdata=data)
>
> par(mfrow=c(2,1))
> plot(x,pred2,type="l",xlab="Duration",ylab="Waiting",
+      xlim=range(x),ylim=range(c(pred1,waiting)))
> points(eruptions,waiting,col="blue",pch="o")
> title("bs() fit")
> plot(x,pred1,type="l",xlab="Duration",ylab="Waiting",
+      xlim=range(x),ylim=range(c(pred1,waiting)))
> points(eruptions,waiting,col="blue",pch="o")
> title("ns() fit")
```

Smoothing splines

- To avoid knot selection problem in regression splines, *smoothing splines* uses a maximal set of knots.
- The complexity of the fit is controlled by regularization.
- Considering the following problem: among all functions $f(x)$ with second continuous derivatives minimize the penalized residual sum of squares

$$RSS(f, \lambda) = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int_a^b (f''(x))^2 dt$$

  where λ is a fixed smoothing parameter.
- The first term measures the closeness of the data where the second term penalizes the curvature of the function, and λ establishes the tradeoff between the two.
- If $\lambda = 0$: f can be any function that interpolates the data.
- If $\lambda = \infty$: the least squares straight line fit, since no second derivative can be tolerated.

Smoothing splines (cont.)

- Even though the criterion is defined on an infinite-dimensional function space, it ends up it has an explicit, finite-dimensional unique minimizer!
- The optimizer is a natural cubic spline with knots at the unique values of x_i, $i = 1, \ldots, n$. This is a piecewise cubic polynomial.
- It seems the family is still over-parametrized, since there are as many as N knots, which implies N d.f. However, the penalty term shrinks some of the way toward the linear fit.
- Since the solution is a natural spline, we can write it as

$$f(x) = \sum_{j=1}^{n} N_j(x)\theta_j$$

where the $N_j(x)$ are an n-dimensional set of basis functions.

Smoothing splines (cont.)

- It can be shown the criterion reduces to

$$RSS(\theta, \lambda) = (\mathbf{y} - \mathbf{N}\theta)^T(\mathbf{y} - \mathbf{N}\theta) + \lambda\theta^T\mathbf{\Omega}_N\theta$$

  where $\{N\}_{ij} = N_j(x_i)$ and $\{\mathbf{\Omega}_N\}_{ij} = \int N_j''(t)N_k''(t)dt$. The solution is easily seen to be

$$\hat{\theta} = \left(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{\Omega}_N\right)^{-1}\mathbf{N}^T\mathbf{y}$$

  a generalized ridge regression.
- The N-vector of fitted values $\hat{f}(x_i)$ at x_i is:

$$\hat{\mathbf{f}} = \mathbf{N}\left(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{\Omega}_N\right)^{-1}\mathbf{N}^T\mathbf{y} = \mathbf{S}_\lambda\mathbf{y}$$

  where $\mathbf{S}_\lambda$ is known as the *smoother matrix*, which only depends on x and λ.
- The *effective d.f.* of a smoothing spline is $df_\lambda = \text{trace}(\mathbf{S}_\lambda)$. The choice of λ can be chosen by CV or Generalized CV.

Generalized cross-validation

- *Generalized cross-validation* provides a convenient approximation to leave-one-out cross-validation (LOOCV), for linear fitting under squared-error loss.
- A linear fitting method is one for which we can write $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$.
- For many linear fitting methods, the average LOOCV error is:

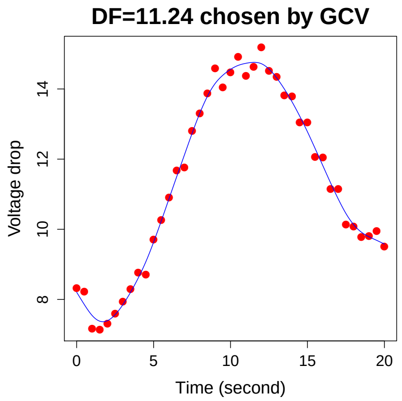
$$\frac{1}{N}\sum_{i=1}^{N}[y_i - \hat{f}^{-i}(x_i)]^2 = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right]^2$$

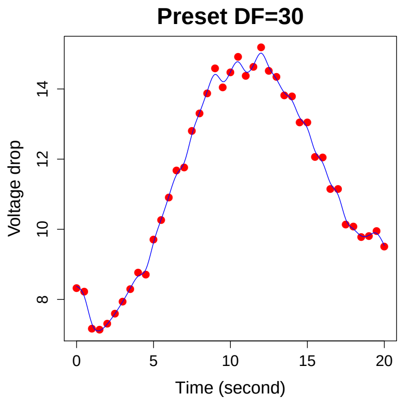
  where $\mathbf{S}_{ii}$ is the ith diagonal element of $\mathbf{S}$.
- The GCV approximation is

$$GCV(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - \hat{f}(x_i)}{1 - \text{trace}(\mathbf{S})/N}\right]^2$$

- GCV can have a computational advantage in some settings, where the trace of $\mathbf{S}$ can be computed more easily than the individual elements S_{ii}.

Revisit battery voltage drop example





```
library(splines)
fit3 <- smooth.spline(data$Time,data$Voltage,cv=F)  #GCV chosen
fit4 <- smooth.spline(data$Time,data$Voltage,df=30) #preset df
```

Other smoothers: kernel smoother

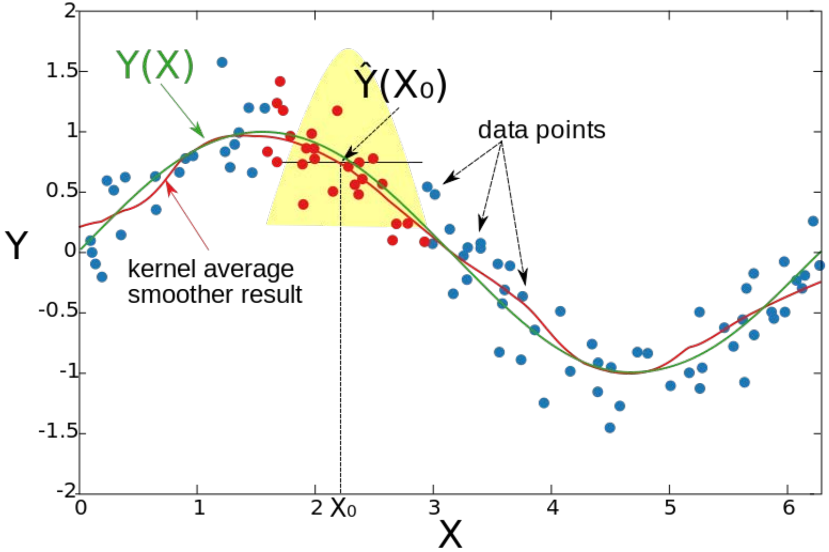
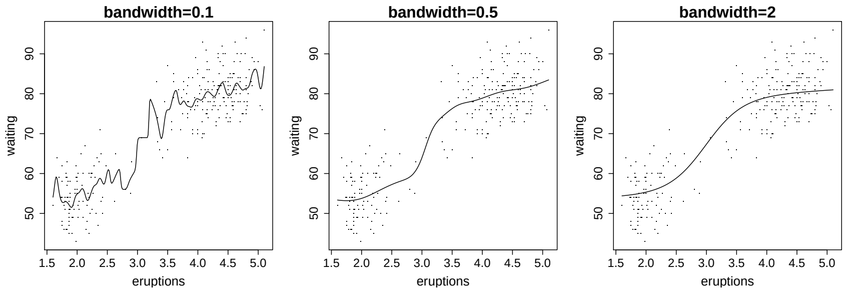


Figure from http://en.wikipedia.org/wiki/Kernel_smoother

Revisit old faithful example using kernel smoothering

```
plot(waiting~eruptions,faithful,main="bandwidth=0.1",pch=".")
lines(ksmooth(eruptions,waiting,"normal",0.1))
plot(waiting~eruptions,faithful,main="bandwidth=0.5",pch=".")
lines(ksmooth(eruptions,waiting,"normal",0.5))
plot(waiting~eruptions,faithful,main="bandwidth=2",pch=".")
lines(ksmooth(eruptions,waiting,"normal",2))
```



`KernSmooth` is a comprehensive kernel smoothing package in `R`.

Other smoothers: local linear/polynomial regression

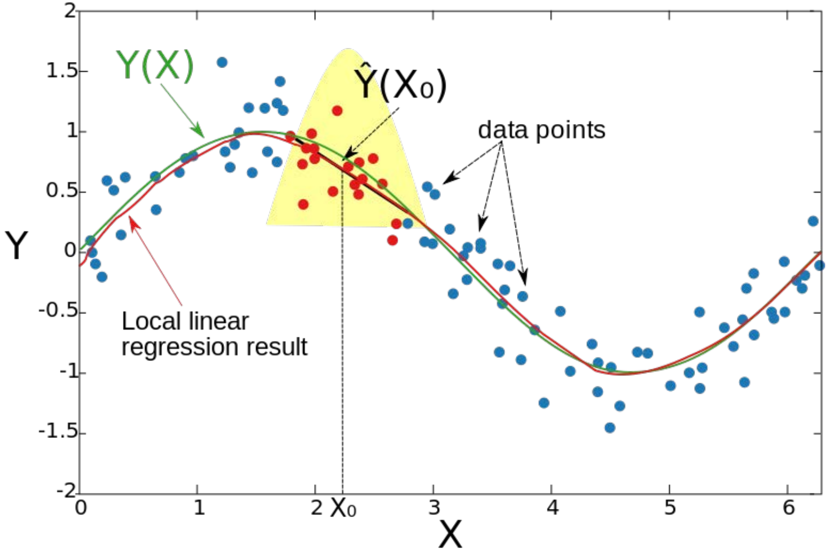


Figure from http://en.wikipedia.org/wiki/Kernel_smoother

Loess smoother

- For each point $\mathbf{x}_i = (z_i, y_i)$, the $\alpha \times n$ nearest points are identified based on the distance $|z_i - z|$. We call this neighborhood of $\alpha \times n$ points "$\mathcal{N}(z)$".
  - With $\alpha = 0.30$ and $n = 164$, the algorithm puts 49 points into $\mathcal{N}(z)$.
- A weighted least-squares linear regression

$$\hat{r}_z(Z) = \hat{\beta}_{z,0} + \hat{\beta}_{z,1} Z$$

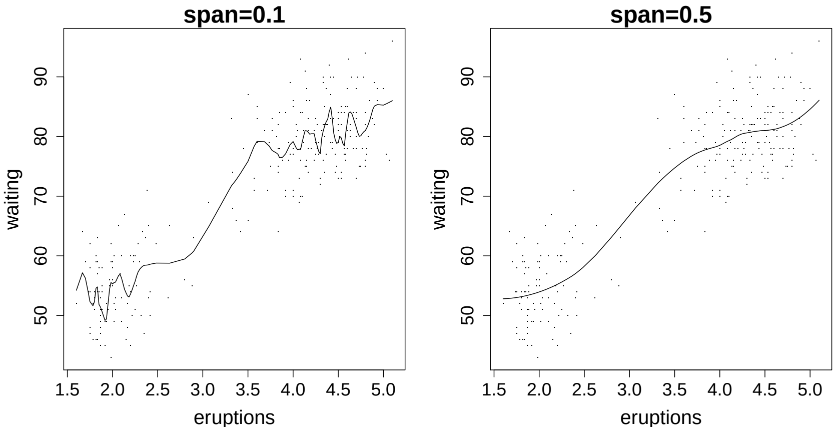
  is fit to the $\alpha \times n$ points in $\mathcal{N}(z)$, where the *weights* $w_{z,j}$ are positive numbers which depend on $|z_j - z|$. Let

$$u_j = \frac{|z_j - z|}{\max_{\mathcal{N}(z)} |z_k - z|},$$

  the weights w_j equal $(1 - u_j^3)^3$.
- Finally, the loess estimate $\hat{r}_{loess}(z)$ is set equal to the value of $\hat{r}_z(Z)$ at $Z = z$.

Revisit old faithful example using loess smoother

```
f <- loess(waiting~eruptions,span=0.1)
i <- order(eruptions)
plot(waiting~eruptions,faithful,main="span=0.1",pch=".")
lines(f$x[i],f$fitted[i])
```



Motivation of generalized additive models

- Traditional linear model: $Y = \beta_0 + \sum_j \beta_j X_j + \epsilon$
  - Pros: simple and easy to interpret.
  - Ideal for linear effect and small sample size data.
  - Cons: In real life, the effects are often not linear
- Multiple nonparametric models: $Y = f(X_1, X_2, \cdots, X_p) + \epsilon$
  - Pros: allow nonlinearity and all possible interactions.
  - Ideal for large n small p data with nonlinear effects and interactions.
  - Cons: Estimates is hard to interpret and highly unstable due to the "curse of dimensionality".
    - In high-dimensional, data is sparse. Neighborhood with fixed proportion of pts become less local as $p \uparrow$ (see next slides).
    - For high-dimensional problem, we need a huge sample size (e.g. to get the same density as $N = 100$ and $p = 1$ in $p = 10$, we need to have $N = 100^{10}$.
- GAM is more flexible than linear model, but still interpretable. Since each nonlinear term is estimated in low dimension, it avoids the curse of dimensionality.

Curse of dimensionality

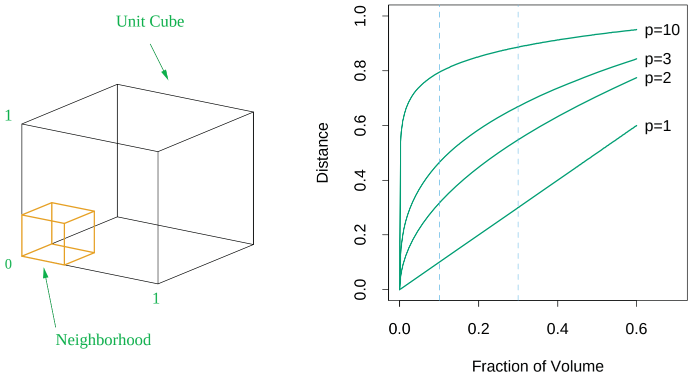


FIGURE 2.6. *The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover* 80% *of the range of each coordinate to capture* 10% *of the data.*

Figure from HTF 2009.

Generalized additive models in regression

- Model assumes that the mean response is a sum of terms each depending on (usually) a single predictor:

$$Y = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) + \epsilon$$

  where f_j are arbitrary functions estimated from the data; the errors ϵ are assumed to have constant variance and a mean of zero.
- if the f_j's are linear terms, this is just traditional regression
- if they are step functions main effect of a factor term
- Additive regression models essentially apply local regression to low dimensional projections of the data
  – That is, they estimate the regression surface by a combination of a collection of one- (or two-) dimensional functions.

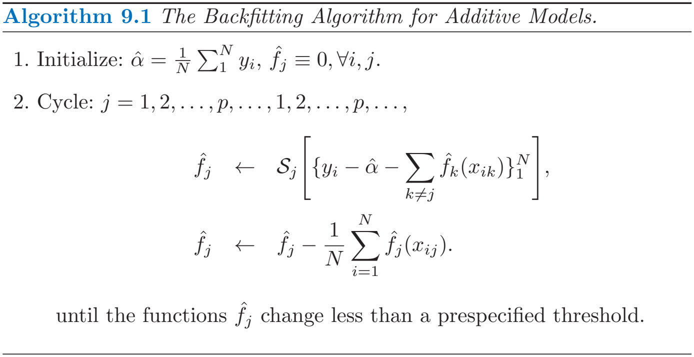
Choices of terms in GAMs

- The f_j represents arbitrary trend that can be estimated by some non-linear smoothers.
- In some cases the additive terms may be known.
- Local regressions such as loess smoother.
- Splines with fixed degrees of freedom.
- Splines with known knots and boundary knot positions.
- Harmonic terms and etc.
- In general the degrees of smoothness of f_j's are chosen by cross validation.

Fitting generalized additive models

- Now comes the question: How do we find these arbitrary trends?
- If the X's are completely independent – which will not be the case – we could simply estimate each functional form using a nonparametric regression of Y on each of the X's separately.
- However, since the X's are related, however, we need to proceed in another way. For a particular j, in order to estimate the trend of $f_j(X_j)$, we need to remove the effects of other predictors.
- We use a procedure called **backfitting** to find each curve, controlling for the effects of the others.

Backfitting algorithm



Algorithm 9.1 *The Backfitting Algorithm for Additive Models.*

1. Initialize: $\hat{\alpha} = \frac{1}{N}\sum_1^N y_i$, $\hat{f}_j \equiv 0, \forall i, j$.
2. Cycle: $j = 1, 2, \ldots, p, \ldots, 1, 2, \ldots, p, \ldots,$

$$\begin{aligned} \hat{f}_j &\leftarrow \mathcal{S}_j\left[\{y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})\}_1^N\right], \\ \hat{f}_j &\leftarrow \hat{f}_j - \frac{1}{N}\sum_{i=1}^N \hat{f}_j(x_{ij}). \end{aligned}$$

until the functions $\hat{f}_j$ change less than a prespecified threshold.

- Backfitting algorithm fits $f_j(x_j)$ using the *partial residuals* $(y - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_k))$ on x_j.
- The algorithm fits the cycle $(f_1, f_2, \ldots, f_p)$ iteratively until the estimates converge.

Figure from HTF 2009.

GAMs for various types of response

- In general, the generalized additive models can be written as:

$$g[\mu(X)] = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p)$$

  where $\mu(X)$ is the conditional mean of the response and g is the **link** function.
- $g(\mu) = \mu$ is the identity link for Gaussian response data.
- $g(\mu) = \text{logit}(\mu) = \log[\mu(X)/(1 - \mu(X))]$ is additive logistic regression for modeling binomial probabilities (note: $\mu(X) = Pr(Y = 1|X)$).
- $g(\mu) = \text{probit}(\mu)$ (inverse Gaussian cumulative distribution) is also for modeling binomial probabilities.
- $g(\mu) = log(\mu)$ for log-linear or log-additive models for Poisson count data.

Various types of nonlinear terms in GAMs

- The nonlinear terms are not restricted to main effects. They can have nonlinear components in two or more variables, or separate curves in X_j for each level of the factor X_k.
- Suppose X, Z, W are quantitative variables and V is a qualitative variables with d levels.
- $g(\mu) = X\beta + \sum_{k=1}^{d} \alpha_k + f(Z)$ is a *semiparametric* model: X is modeled linearly; α_k the effect for the kth level of a qualitative input V; Z is modeled nonparametrically.
- $g(\mu) = f(X) + \sum_{k=1}^{d} g_k(Z)$ includes an interaction term between V and Z (i.e. $g(V, Z) = \sum_{k=1}^{d} g_k(Z)$).
- $g(\mu) = f(X) + g(Z, W)$ includes a nonparametric function g in two predictors.

GAMs in R

- There are two excellent packages for fitting generalized additive models in `R`.
- The `gam` (for generalized additive model) function in the `mgcv` (multiple smoothing parameter estimation with generalized cross validation) fits generalized additive models using smoothing splines.
  - The smoothing parameter can be chosen automatically using cross-validation or manually by specifying the degrees of freedom.
- The `gam` function in the `gam` package allows either loess (`lo(x)`) or smoothing splines (`s(x)`) to be specified. Other smoothers can be added by the users with appropriate interface functions.
- The `anova` function can be used for both functions, allowing different models to be easily compared.

Example: the Iowa wheat yield data

- An example from Draper N R, and Smith H, *Applied regression analysis*, 2nd Ed., John Wiley & Sons, New York, 1981.
  - Response: Yield of wheat in bushels/acre for the state of Iowa for the years 1930-1962
  - Predictors: Year (as surrogate), Rain0, 1, 2, 3, Temp1, 2, 3, 4
- Objective: Build a predictor for Yield from the predictors available.
  - Note: with only 33 observations and 9 possible predictors some care has to be taken in choosing a model.

An initial linear model

```
> Iowa <- read.table("Iowa.csv",header=T,sep=",")
> iowa.lm1 <- lm(Yield ~ ., Iowa)
> iowa.step <- stepAIC(iowa.lm1, scope = list(lower =  ~ Year,
+             upper =  ~ .), k = log(nrow(Iowa)), trace = F)
> dropterm(iowa.step, test = "F", k = log(nrow(Iowa)),
+ sorted = T)

Model:
Yield ~ Year + Rain0 + Rain2 + Temp4
       Df Sum of Sq    RSS    AIC F Value     Pr(F)
<none>               1554.6  144.6
Temp4   1     188.0 1742.6  144.9     3.4   0.07641 .
Rain0   1     196.0 1750.6  145.0     3.5   0.07070 .
Rain2   1     240.2 1794.8  145.9     4.3   0.04680 *
Year    1    1796.2 3350.8  166.5    32.4 4.253e-06 ***
```

Initial reflections

- With the BIC penalty on model complexity, only two of the terms found are borderline significant in the conventional sense – a consequence of the small sample size.
- Nevertheless the terms found are tentatively realistic:
  - `Year`: surrogate for crop improvements
  - `Rain0`: a measure of pre-season sowing conditions
  - `Rain2`: rainfall during the critical growing month
  - `Temp4`: climatic conditions during harvesting
- Are strictly linear terms in these variables reasonable?

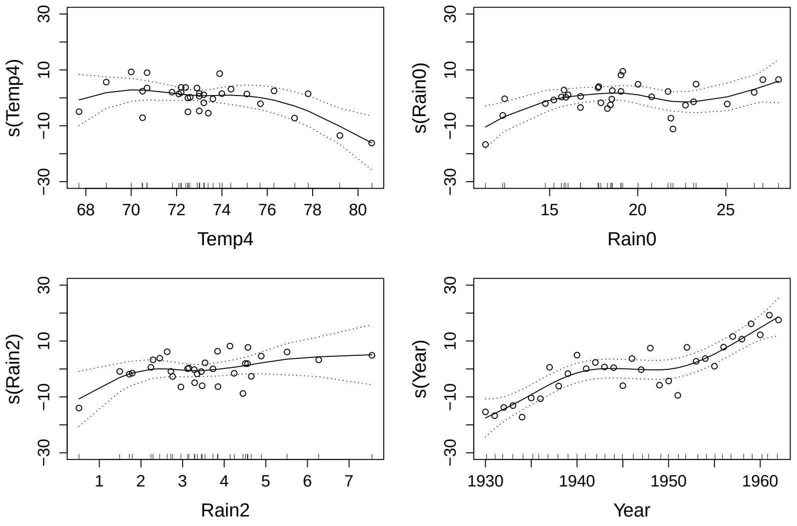
Additive models

- Here we use gam function in gam library.
- Consider a non-parametric smoother in each term:

```
> library(gam)
> iowa.gam <- gam(Yield ~ s(Temp4) + s(Rain0) +
+ s(Rain2) + s(Year), data = Iowa)
> par(mfrow = c(2,2))
> plot(iowa.gam, se = T, ylim = c(-30, 30),
+ resid = T)
```

- `s` specifies a smoothing spline fit in `gam` formula. The default is `s(x, df=4, spar=1)`. Note if `df=1`, it fits a linear model. `spar` is the smoothing parameter between 0 and 1.
- For specifying loess smoother in `gam` formula, use `lo`. The default is `lo(x, span=0.5, degree=1)`. Note `degree=1` specifies the degree of local polynomial to be fit. It is restricted to be 1 or 2 (i.e. local quadratic regression).
- It can be important to keep the *y*-axes of these plots approximately the same to allow comparisons between terms.

Component plot for each term



The univariate histogram or rugplot is displayed along the base of each plot.

Speculative comments

- `Temp4`: Two very hot years had crop damage during harvest.
- `Rain0`: Wide range where little difference, but very dry years may lead to a reduced yield and very wet years to an enhanced one.
- `Rain2`: One very dry growing month led to a reduced yield.
- `Year`: Strongest and most consistent predictor by far. Some evidence of a pause in new varieties during the war and immediately post-war period.

Tentative inference

```
> summary(iowa.gam) # (result edited)

Call: gam(formula = Yield ~ s(Temp4) + s(Rain0) + s(Rain2) +
+ s(Year), data = Iowa)

(Dispersion Parameter for Gaussian family taken to be 31.1175 )

Residual Deviance: 497.8868 on 16.002 degrees of freedom

Number of Local Scoring Iterations: 2

DF for Terms and F-values for Nonparametric Effects

            Df Npar Df   Npar F     Pr(F)
(Intercept)  1
   s(Temp4)  1        3 2.4709    0.09917
   s(Rain0)  1        3 3.0301    0.05993
   s(Rain2)  1        3 1.3746    0.28638
    s(Year)  1        3 3.6841    0.03437
```

Use lm to fit an additive model

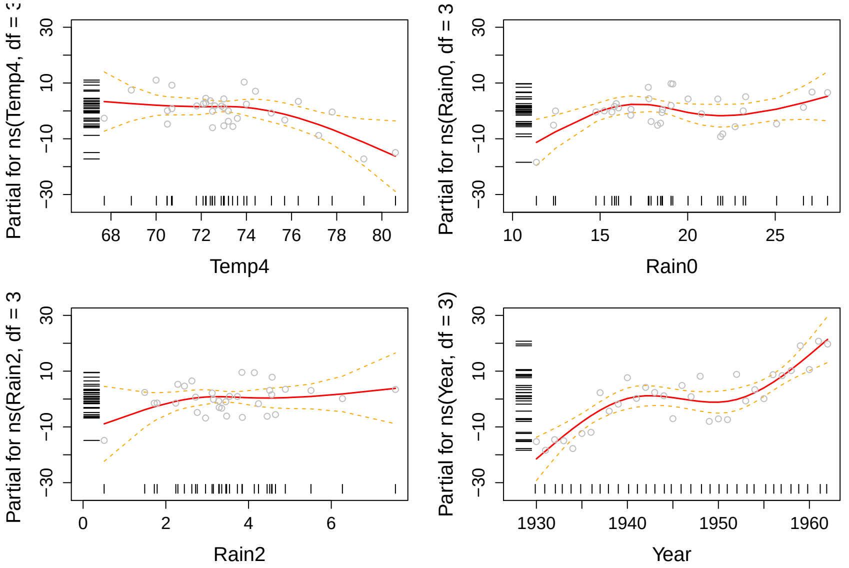
```
> iowa.ns <- lm(Yield ~ ns(Temp4, df=3) + ns(Rain0, df=3)
+            + ns(Rain2, df = 3) + ns(Year, df=3), Iowa)
> termplot(iowa.ns, se = T, rug = T, partial = T)
> dropterm(iowa.ns, test = "F", k = log(nrow(Iowa)))

Single term deletions

Model:
Yield ~ ns(Temp4, df = 3) + ns(Rain0, df = 3) + ns(Rain2, df = 3) +
        ns(Year, df = 3)
                  Df  Sum of Sq     RSS     AIC  F Value     Pr(F)
           <none>                726.26  147.47
ns(Temp4, df = 3)  3     274.60 1000.86  147.56     2.52   0.08706
ns(Rain0, df = 3)  3     332.31 1058.57  149.41     3.05   0.05231
ns(Rain2, df = 3)  3      70.61  796.87  140.04     0.65   0.59327
 ns(Year, df = 3)  3    2022.93 2749.19  180.91    18.57   0.00001
```

Unlike gam package use backfitting to fit GAMs, this model is fitted by simple least squares.

Component plot for each term



Final remarks on Iowa wheat example

- Very similar pattern to the components as for the additive model
- Now clear that the term in `Rain2` is not significant (can you see that on the plot?)
- The `Temp4` and `Rain0` terms will need to be reassessed.
- The term in `Year` stands out as dominant with a clear pattern in the response curve and the partial residuals following it closely.
- Small data sets like this can be very misleading! Extreme caution is needed.

Rock data example

- Response (perm) is the permeability in milli-Darcies
- Three predictors: area (area of pores space), peri (perimeter in pixels) and shape (perimeter/sqrt(area)).
- Problem: build a predictor for log(perm) using the available predictors

```
> rock.lm <- lm(log(perm) ~ area + peri + shape,
+               data = rock)
> summary(rock.lm)

Coefficients:
               Value  Std. Error   t value Pr(>|t|)
(Intercept)   5.3331      0.5487     9.720    0.000
       area   0.0005      0.0001     5.602    0.000
       peri  -0.0015      0.0002    -8.623    0.000
      shape   1.7570      1.7560     1.000    0.323
```

Rock data example (cont.)

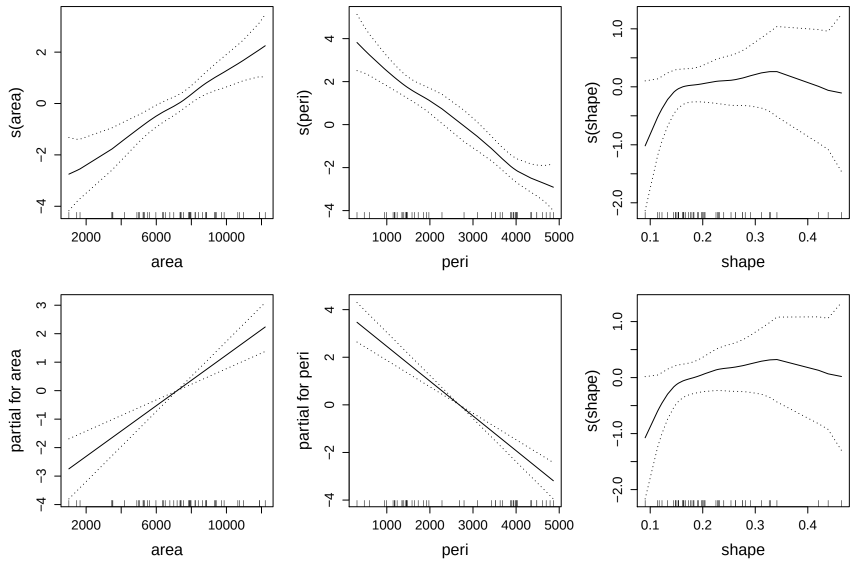
```
> rock.gam <- gam(log(perm) ~ s(area) + s(peri) + s(shape), control =
+              gam.control(maxit = 50, bf.maxit = 50), data = rock)
> summary(rock.gam)
Anova for Parametric Effects
              Df Sum Sq Mean Sq  F value    Pr(>F)
s(area)    1.000 14.577  14.577  19.5788 8.988e-05 ***
s(peri)    1.000 75.670  75.670 101.6321 6.856e-12 ***
s(shape)   1.000  0.413   0.413   0.5548    0.4613
Residuals 35.001 26.060   0.745

Anova for Nonparametric Effects
            Npar Df  Npar F  Pr(F)
(Intercept)
s(area)           3 0.34165 0.7953
s(peri)           3 0.94055 0.4315
s(shape)          3 1.43200 0.2500
>
> anova(rock.lm, rock.gam) # shows no improvement
Model 1: log(perm) ~ area + peri + shape
Model 2: log(perm) ~ s(area) + s(peri) + s(shape)
  Res.Df    RSS     Df Sum of Sq      F Pr(>F)
1 44.000 31.949
2 35.001 26.060 8.9995    5.8891 0.8789 0.5528
```

Rock data example (cont.)

```
> rock.gam1 <- gam(log(perm) ~ area + peri + s(shape), data = rock)
> summary(rock.gam1)
Anova for Nonparametric Effects
            Npar Df Npar F  Pr(F)
area
peri
s(shape)          3 1.3901 0.2594
>
> anova(rock.lm, rock.gam1, rock.gam)
Model 1: log(perm) ~ area + peri + shape
Model 2: log(perm) ~ area + peri + s(shape)
Model 3: log(perm) ~ s(area) + s(peri) + s(shape)
  Res.Df    RSS     Df Sum of Sq      F Pr(>F)
1 44.000 31.949
2 41.000 28.999 3.0000    2.9495 1.3205 0.2833
3 35.001 26.060 5.9995    2.9396 0.6581 0.6835
>
> par(mfrow=c(2,3),mar=c(5,5,1,1), cex.lab=1.5, cex.axis=1.3,
+     cex.main=2, pty = "s")
> plot(rock.gam, se = T)
> plot(rock.gam1, se = T)
```

Component plot for each term



Lessons

- Although suggestive, the curve in shape is not particularly convincing.
- Choosing the degree of smoothness can be tricky. The `gam` function in Simon Wood's `R` implementation (`mgcv` library) offers a way around this.
- In this case, Simon Wood's `gam` function suggests essentially linear terms, at most, in all three variables.

Reference

- Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models by Julian Faraway.
- The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd Ed.) by Trevor Hastie, Robert Tibshirani, and Jerome Friedman.
- Generalized Additive Models by Trevor Hastie and Robert Tibshirani.