Extensions of Linear Regression

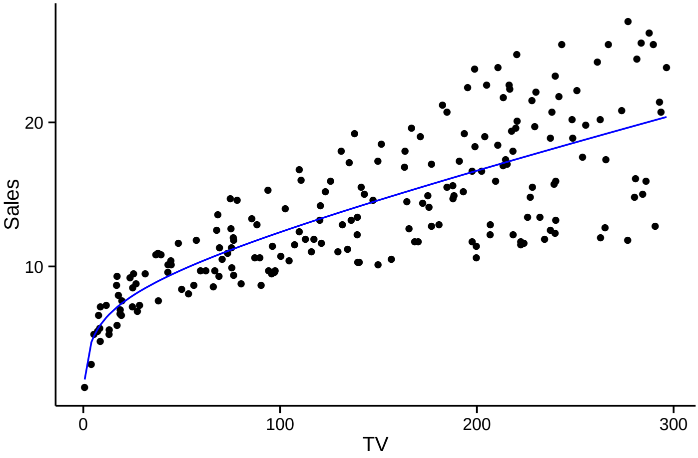
$$\hat{Y} = \beta_0 + \beta_1 Z_1 + \cdot + \beta_k Z_k$$

where Z_i are functions of $X_1, \ldots, X_p$.

```
> mod2 <- lm(Sales ~ TV + log(TV), data=Advertising)
>summary(mod2)
    *   *   *
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.629151   1.606838   1.636  0.10339
TV          0.032968   0.005748   5.736 3.61e-08 ***
log(TV)     1.401023   0.490802   2.855  0.00477 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.201 on 197 degrees of freedom
Multiple R-squared:  0.6273,    Adjusted R-squared:  0.6235
F-statistic: 165.8 on 2 and 197 DF,  p-value: < 2.2e-16
```

```
ggplot(mod2) + geom_point(aes(x=TV, y=Sales)) +
    geom_line(aes(x=TV, y=.fitted), color="blue" )
```



Polynomial Regression

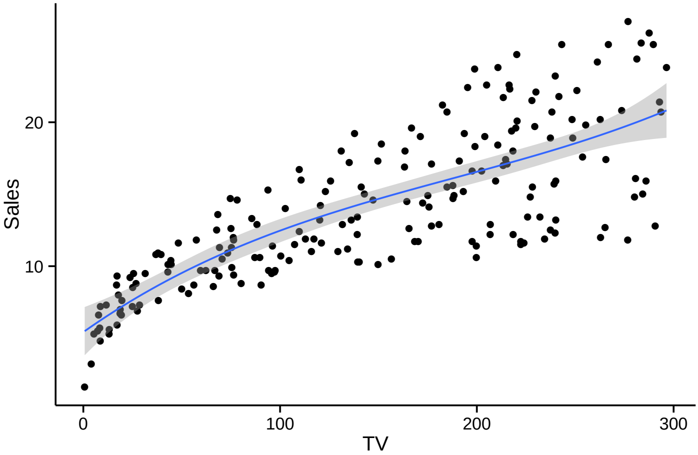
$$\hat{Y} = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdot + \beta_k X^k$$

```
> summary(lm(Sales ~ poly(TV,3), data=Advertising))
    *   *   *
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   14.0225     0.2286  61.353   <2e-16 ***
poly(TV, 3)1  57.5727     3.2322  17.812   <2e-16 ***
poly(TV, 3)2  -6.2288     3.2322  -1.927   0.0554 .
poly(TV, 3)3   4.0074     3.2322   1.240   0.2165
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.232 on 196 degrees of freedom
Multiple R-squared:  0.622,     Adjusted R-squared:  0.6162
F-statistic: 107.5 on 3 and 196 DF,  p-value: < 2.2e-16
```

Polynomial Regression

```
qplot(TV, Sales, data=Advertising) +
    geom_smooth(method="lm", formula=y~poly(x,3))
```



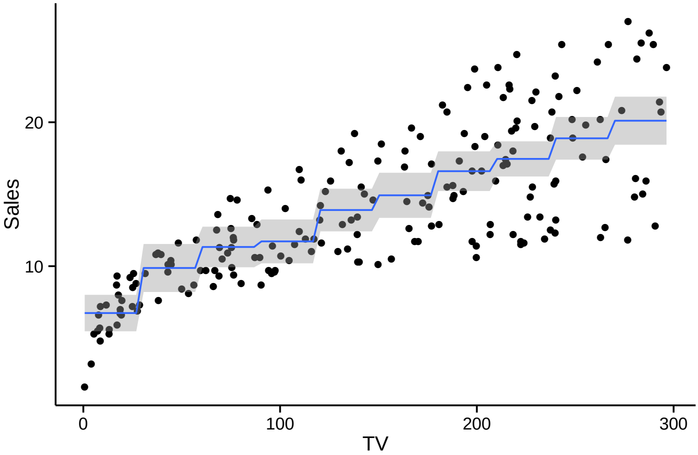
Step Function Regression

$$\hat{Y} = \beta_0 + \beta_1 I(X < c_1) + \beta_2 I(c_1 \leq X < c_2) + \cdot + \beta_k I(c_k \leq X)$$

```
> summary(lm(Sales ~ cut(TV,10), data=Advertising))
    *   *   *
Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)              6.7423     0.6425  10.493  < 2e-16 ***
cut(TV, 10)(30,59.7]     3.1310     1.0623   2.947  0.00361 **
cut(TV, 10)(59.7,89.3]   4.5910     0.9613   4.776 3.57e-06 ***
cut(TV, 10)(89.3,119]    4.9799     1.0046   4.957 1.58e-06 ***
cut(TV, 10)(119,149]     7.1577     0.9889   7.238 1.09e-11 ***
    *   *   *
Residual standard error: 3.276 on 190 degrees of freedom
Multiple R-squared:  0.6235,    Adjusted R-squared:  0.6057
F-statistic: 34.96 on 9 and 190 DF,  p-value: < 2.2e-16
```

Step Function Regression

```
qplot(TV, Sales, data=Advertising) +
    geom_smooth(method="lm", formula=y~cut(x,10))
```



Regression Splines

In general

$$\hat{Y} = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \cdot + \beta_k b_k(X)$$

where b_i are *basis functions*.

- Splines are piece-wise polynomials that connect smoothly at the points where the pieces meet (*knots*).
- Partition interval using k knots:

$$a = c_0 < c_1 < \cdots < c_k < c_{k+1} = b$$

- For cubic splines we have a cubic polynomial in each interval:

$$s(x) = A_0 + A_1 x + A_2 x^2 + A_3 x^3$$

- At each knot we have 3 conditions: value, first derivative and second derivative must match on both sides

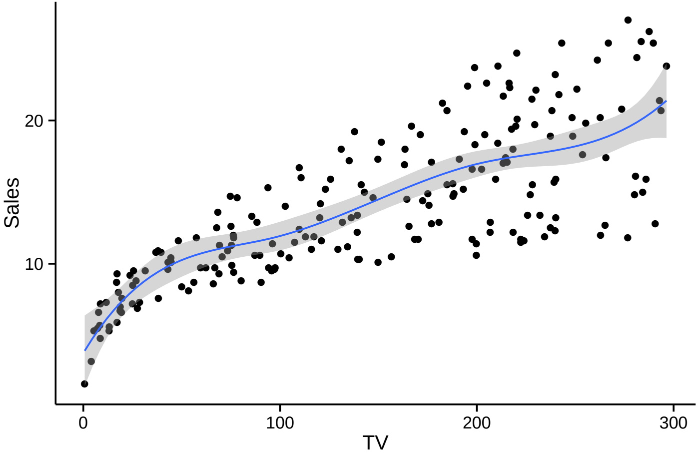
Regression Splines

- Total number of coefficients: $4(k+1)$.
- Total number of constraints: $3k$
- Degrees of freedom left to fit the model: $k+4$

- Additional boundary constraints may be imposed, e.g. *natural* spline is linear at the boundaries.
- Degrees of freedom left: $k+2$.
- Natural splines are more stable at the boundary.

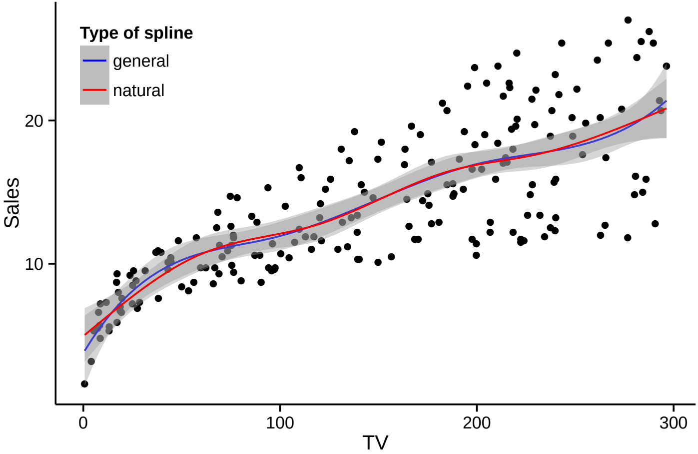
Regression Splines

```
library(splines)
qplot(TV, Sales, data=Advertising) +
    geom_smooth(method="lm", formula=y~bs(x,df=5))
```



Regression Splines

```
qplot(TV, Sales, data=Advertising) +
    geom_smooth(method="lm", formula=y~bs(x,df=5)) +
    geom_smooth(method="lm", formula=y~ns(x,df=5), color="red")
```



Model Selection

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

The goal is to find a model that is a good balance between complexity and fit.

- Variable selection.
- Ridge regression:

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \longrightarrow \min$$

- LASSO:

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \longrightarrow \min$$

λ is called *penalty parameter* and is determined separately.

Model Selection

- Variable selection will select a model with only some of the variables.
- Ridge regression always produces a model with all predictors, however:
  - computationally less expensive;
  - good choice of λ would produce better fit.
- LASSO:
  - forces some coefficients to be exactly zero, i.e. performs variable selection;
  - computationally between the other two.

Penalty Parameter

The *bias-variance trade-off*:

$$MSE = \left(\text{BIAS of } \hat{f}\right)^2 + \left(\text{Variance of } \hat{f}\right)$$

- Many degrees of freedom allow a closer fit to the data, resulting in low bias and high variance.
- Few degrees of freedom result in stiffer fit with low variance and high bias.

- As $\lambda \to 0$ the penalty term is weak and allows more flexibility.
- As $\lambda \to \infty$ the bias of the estimator increases while the variance decreases.
- There is a value of λ that minimizes *MSE*.

Ridge/LASSO Regression

```
x <- model.matrix(Sales ~ ., data=Advertising)[,-1]
y <- Advertising$Sales
tf <- sample(c(FALSE,TRUE), size=length(y),
     prob=c(0.1,0.9),repl=TRUE)
library(glmnet)

> # For ridge regression set alpha=0
> rr <- glmnet(x[tf,], y[tf], alpha=0, lambda=seq(0.01,0.1,by=0.01))
> y.pred <- predict(rr, newx=x, type="response")
> MSE <- colMeans( (y.pred - y[!tf,)^2 )
> rbind(lambda=rr$lambda, MSE)
          s0    s1    s2   s3    s4    s5    s6    s7    s8    s9
lambda 0.100 0.090 0.080 0.07 0.060 0.050 0.040 0.030 0.020 0.010
MSE    1.804 1.799 1.795 1.79 1.785 1.781 1.777 1.773 1.769 1.765
> # For lasso set alpha=1
lasso <- glmnet(x[tf,], y[tf], alpha=1, lambda=seq(0.01,0.1,by=0.01))
y.pred <- predict(lasso, newx=x[!tf,], type="response")
MSE <- colMeans( (y.pred - y[!tf])^2 )
rbind(lambda=lasso$lambda, MSE, DF=lasso$df)
          s0    s1    s2    s3    s4    s5    s6    s7    s8    s9
lambda 0.100 0.090 0.080 0.070 0.060 0.050 0.040 0.030 0.020 0.010
MSE    1.778 1.766 1.755 1.746 1.742 1.738 1.735 1.737 1.744 1.753
DF     2.000 2.000 2.000 3.000 3.000 3.000 3.000 4.000 4.000 4.000
```

Automatic selection of λ

In *k-fold Cross Validation* the training set is split into k subset. The i-th subset is left out and the model is fit. The *MSE* over the i-th set is computed, denote MSE_i. The *cross-validation statistic* is

$$CV_k = \frac{1}{k} \sum_{i=1}^{k} MSE_i.$$

- LOOCV - leave one out cross-validation when $k = n$.

```
> # 20-fold cross-validation of lasso
> cv.out <- cv.glmnet(x, y, nfolds=20, alpha=1)
> cv.out$lambda.min
[1] 0.07452947
> plot(cv.out)
```

Non-parametric regressions

Most generally we have the model

$$\hat{Y} = f(X)$$

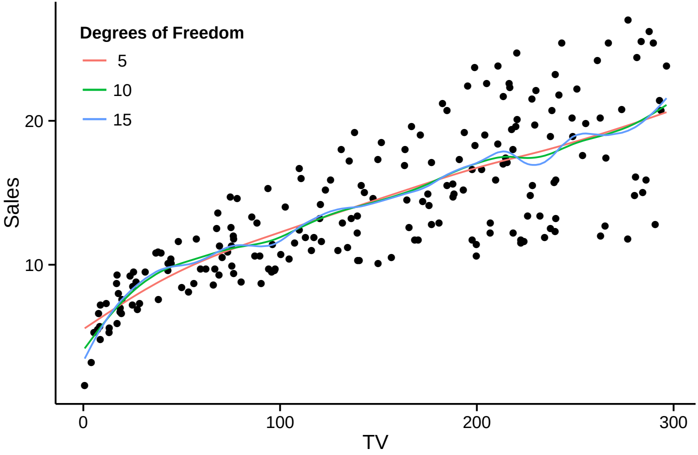
- Smoothing splines: set f to be a cubic spline with given knots and solve

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int \left(f''(x)dx\right)^2 \longrightarrow \text{min}\,.$$

- The *smoothing parameter*, λ can be chosen by cross-validation.

```
> sp1 <- smooth.spline(Advertising$TV, Advertising$Sales, df=5)
> sp1.p <- predict(sp1, Advertising$TV)
> qplot(TV, Sales, data=Advertising) +
    geom_line(aes(x=TV, y=sp1.p$y))
```

Smoothing splines



Local regression

- Local regression: set $f = f(x;\beta)$ to be some parametric regression model and allow the parameters, β to change with x
- For given x, find β that solves the following weighted minimization

$$\sum_{i=1}^{n} K(x, x_i)\,(y_i - f(x_i;\beta))^2 \longrightarrow \min .$$

- Set $f(x) = f(x, \beta)$ for the value of β found above.

Local regression

- The weights should add-up (or integrate) to 1 and give more weight to points close to x.
- Some choices of the weights, K, include
- $1/k$ fraction of all points nearest x

$$K(x, x_i) = \begin{cases} k/n, & x_i \text{ is among the } n/k \text{ points nearest } x; \\ 0, & \text{otherwise.} \end{cases}$$

- Uniform over window of width h:

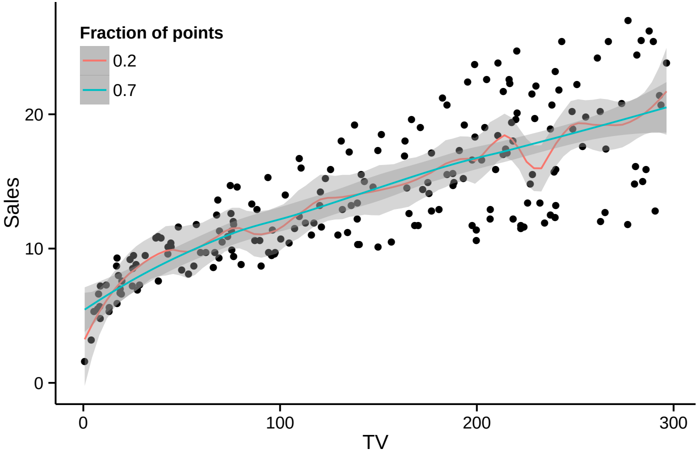
$$K(x, x_i) = \begin{cases} 1/h, & |x - x_i| < h; \\ 0, & \text{otherwise.} \end{cases}$$

- Gaussian with standard deviation h

$$K(x, x_i) = \varphi(x_i; \mu = x, \sigma = h).$$

Local regression

```
qplot(TV, Sales, data=Advertising) +
    geom_smooth(method="loess", degree=2, span=0.2, aes(color="0.2")) +
    geom_smooth(method="loess", degree=2, span=0.7, aes(color="0.7")) +
    theme_classic() + labs(color="Fraction of points") +
    theme(legend.justification=c(0,1), legend.position=c(0,1))
```



IBM lectures

Reminder:

- February 1 - Cognos Workspace in HP5345
- February 22 - SPSS Modeler in HP5345
- March 7 - Watson Analytics in HP5345