3.0 - Regression modelling

Regression analysis

Linear regression is a central tool of statistical analysis. It is used extensively and often is a key component within a more complex analytical procedure.

Regression methods, including generalized linear modelling (GLM) and nonlinear regression, apply to a wide range of problems, including:

- trend estimation and prediction
- diagnostic tests for model selection
- benchmarking and (methodological) explorative analysis

In this section we review elementary notions related to the implementation of regression analysis.

3.1 - Linear regression

Linear regression

In linear regression models, observations depend linearly on the parameters of interest:

$$Y = \theta_0 + \theta_1 X + \varepsilon$$

where ε traditionally represents a collection of realizations of a Gaussian r.v. with mean 0

- Ordinary least squares are most often applied for estimation of $\theta = (\theta_0, \theta_1)$
- When the error term ε is not assumed Gaussian, maximum likelihood estimation is the more generic framework
- MLE is equivalent to OLS when $\varepsilon \sim N(0, \sigma^2)$
- There are many other estimation criteria

M-estimation

M-estimation: an **objective function** ρ is applied to the **residuals**

$$e(\tilde{\theta}_0, \tilde{\theta}_1) = Y - \tilde{Y} = Y - \tilde{\theta}_0 - \tilde{\theta}_1 X$$

to form a criterion that gets optimised with respect to (θ_0, θ_1):

$$(\hat{\theta}_0, \hat{\theta}_1) = \arg\min_{\theta_0, \theta_1} \sum_{i=1}^{N} \rho\left(e_i(\theta_0, \theta_1)\right)$$

For example $\rho(u) = u^2$ and we optimise the sum of squares

$$\sum_{i=1}^{N} \Big(e_i(\tilde{\theta}_0, \tilde{\theta}_1)\Big)^2$$

Any choice of a function ρ may be implemented and optimised.

Linear regression diagnostics

Note that if $(\tilde{\theta}_0, \tilde{\theta}_1) \equiv (\theta_0, \theta_1)$ then the residuals are identical to the error term ε.

Recall the main assumptions on the (traditional) linear model:

- the error term has zero mean and constant variance
- the errors are uncorrelated
- the errors are Normally distributed or sample size is large enough for large sample theory to be applied

This means that if the model fit (estimation) is good, the residuals should look like they satisfy these properties – although residuals are **not** independent r.v.'s.

3.2 - Nonlinear regression

Polynomial regression

There is a lot of flexibility (choice) in how one may tackle nonlinear relationships. Nonlinearity may be expressed with respect to the parameter of interest, the model covariate, or both.

Polynomial regression uses a model that is linear in the parameter of interest but nonlinear in the regression covariate:

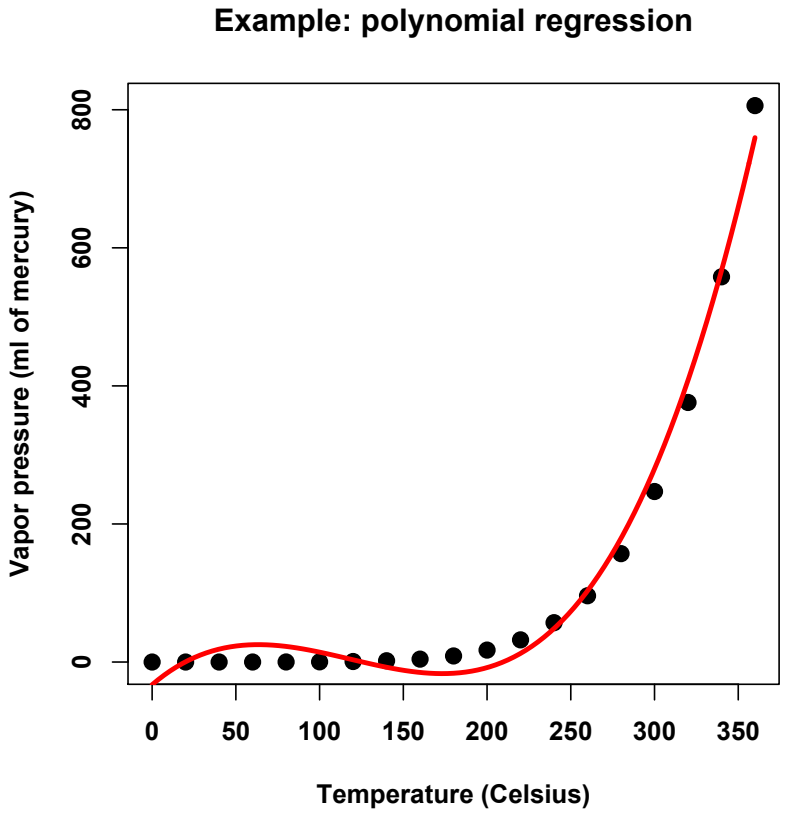
$$Y = \theta_0 + \theta_1 X + \theta_2 X^2 + \cdots + \theta_p X^p + \varepsilon$$

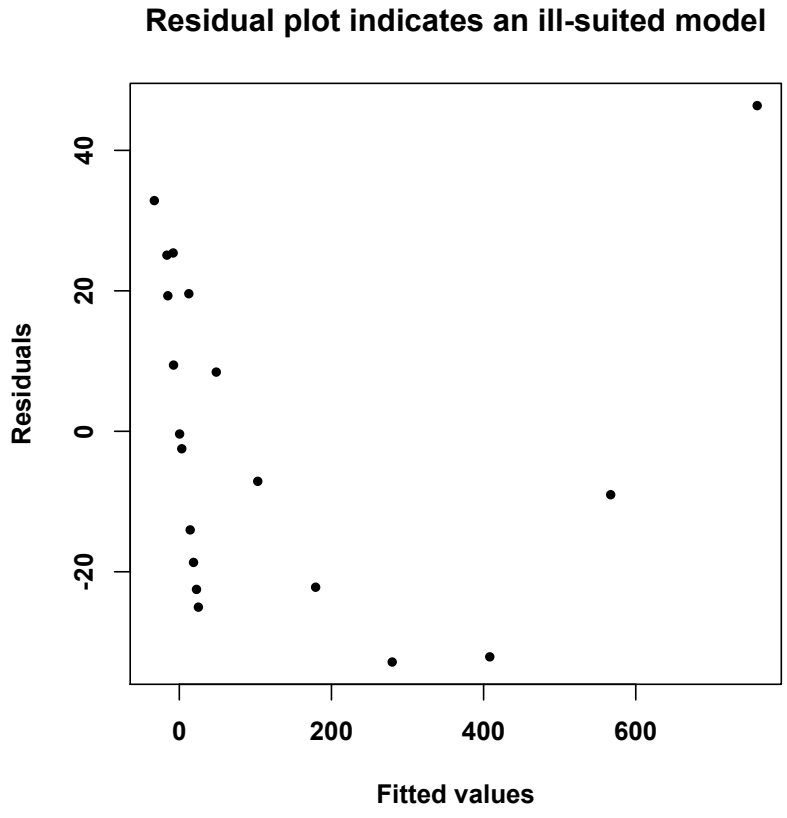
Goodness of fit depends on the order p of the polynomial (ideally, $p = N - 1$), but polynomial regression is hard to interpret.

NB: $Y = \theta^2 X + \varepsilon$ is actually $Y = \beta X + \varepsilon$ with $\beta = \theta^2 \in \mathbb{R}^+$...

Polynomial regression

Exercise: `attach(pressure); x=temperature; y=pressure`
Implement the linear regression to obtain the following plot:





Nonlinear regression (using least squares)

Nonlinear regression allows the model structure (i.e. conditional expectation $E(Y \mid X)$ of the observations Y) to depend nonlinearly on the parameters of interest.

Examples:

$$Y = \theta_0 + \theta_1 X_1 + \theta_2^2 X_2 + \varepsilon$$
$$Y = e^{\theta X} + \varepsilon$$

Nonlinear regression via Least Squares is performed in R using `nls`.

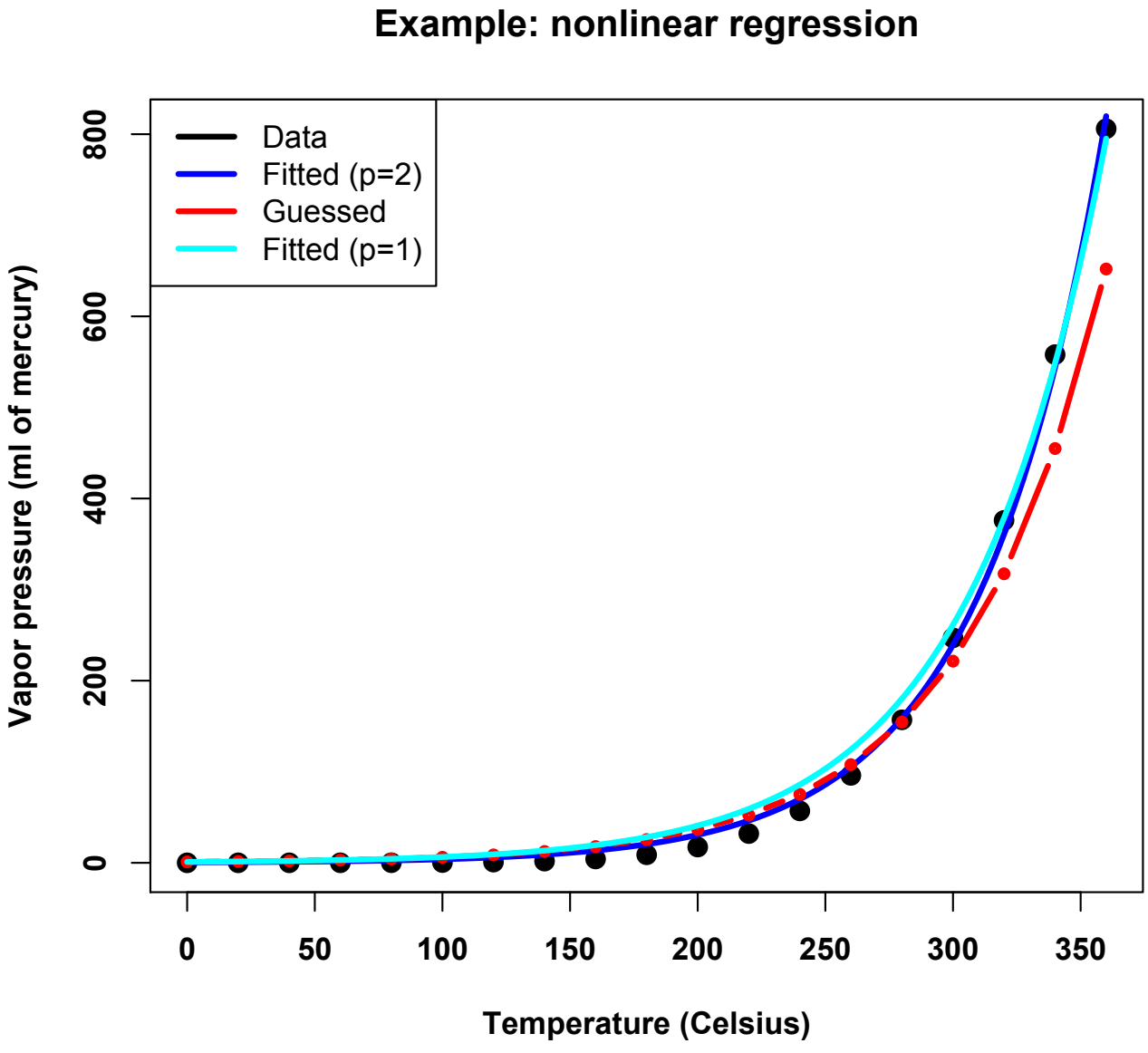
NB: nonlinear methods are usually (very) sensitive to initialisation.

Example: 2-parameter exponential model (LearnR page)

```
par(font.lab= 2, font.axis=2)
plot(x, y, pch=20, cex=2,
     xlab="Temperature (Celsius)",
     ylab="Vapor pressure (ml of mercury)",
     main="Example: polynomial regression")

nlreg <- nls(y~exp(a+b*x), start=list(a=0,b=.5))
     # error... need for better initialisation
nlreg <- nls(y~exp(a+b*x), start=list(a=0,b=.02))
summary(nlreg)
cc = coef(nlreg)
curve(exp(cc[1]+cc[2]*x), col='blue', add=T, lwd=3)
points(x, exp(0+0.018*x), col='red', t='b', lwd=3, cex=.5)

nlreg2 <- nls(y~exp(a*x), start=list(a=.02))
cc = coef(nlreg2)
curve(exp(cc[1]*x), col='cyan', add=T, lwd=3)
legend("topleft", col=c('black','blue','red','cyan'), lwd=3,
     legend=c("Data","Fitted (p=2)","Guessed","Fitted (p=1)"))
```


Example: nonlinear regression
Data
Fitted (p=2)
Guessed
Fitted (p=1)
Vapor pressure (ml of mercury)
0
200
400
600
800
0
50
100
150
200
250
300
350
Temperature (Celsius)

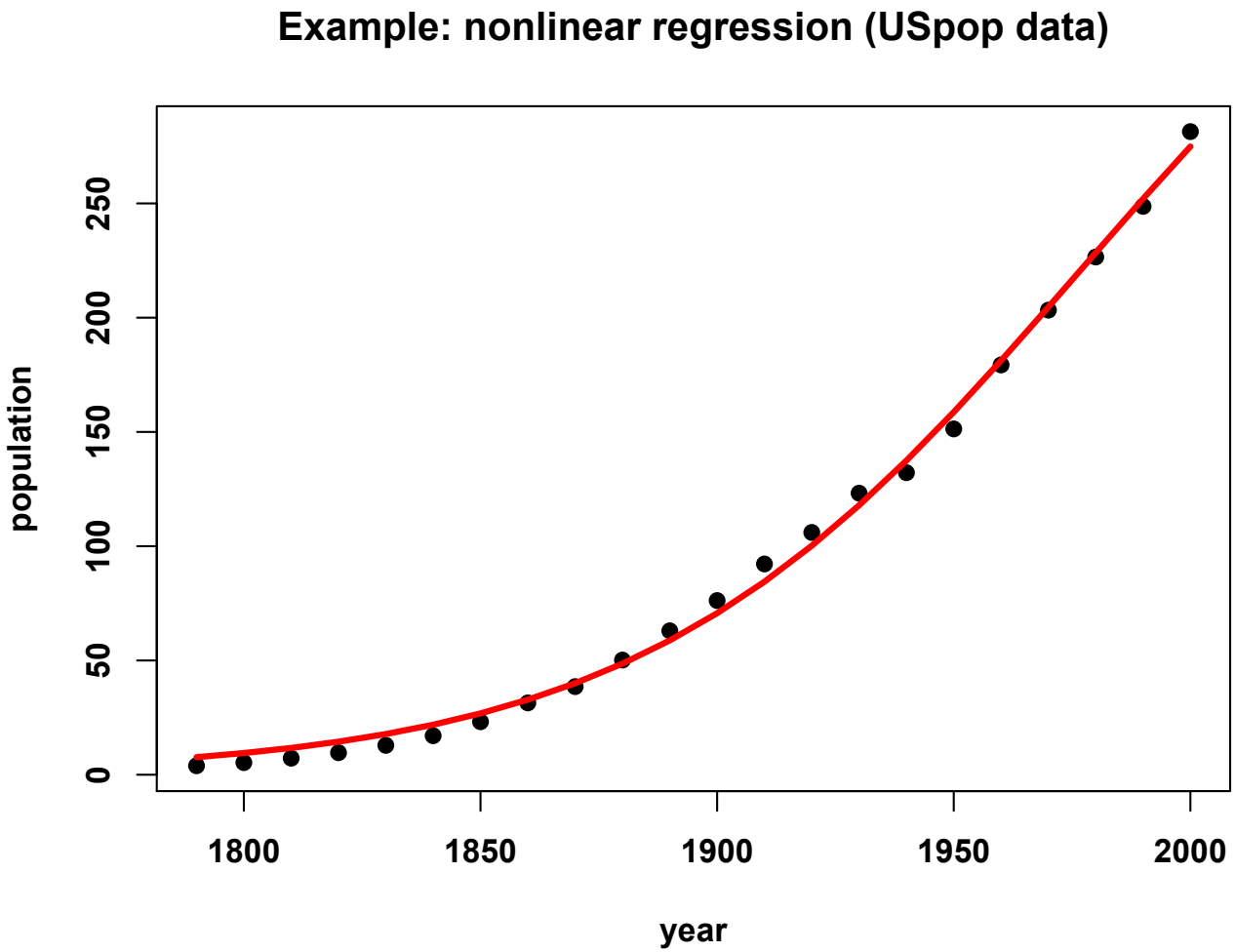
Example: US population (LearnR page)

```
library(car)
data(USPop)
attach(USPop)
par(font.lab= 2, font.axis=2)

plot(year, population, cex=1.5, pch=20,
     main="Example: nonlinear regression (USpop data)")
time <- 0:21

pop.mod <- nls(population ~ beta1/(1 + exp(beta2 + beta3*time)),
     start=list(beta1 = 350, beta2 = 4.5, beta3 = -0.3),
     trace=TRUE)

summary(pop.mod)
lines(year, fitted.values(pop.mod), lwd=3, col='red')
```


Example: nonlinear regression (USpop data)
population
0
50
100
150
200
250
1800
1850
1900
1950
2000
year

More details on NLS may be found e.g. at:
http://cran.r-project.org/doc/contrib/Fox-Companion/appendix-nonlinear-regression.pdf

optim and optimx

If one wants to implement another criterion, `optim` and `optimx` become useful for (general purpose) optimization. (The `optimx` package extends former `optim`.)

`optim` implements Nelder-Mead, quasi-Newton and conjugate-gradient methods, and includes the possibility of box constraints.

It takes definitions of the cost function (estimation criterion) and (optionally) its gradient as arguments, along with starting values. This allows users to define their own estimating functions.

Example: optim (LearnR page)

```
par(mfrow=c(1,2),font.lab= 2, font.axis=2)
model <- function(x, theta){
     return(theta*x)
}
crit <- function(theta,x,y){
# must have theta as 1st parameter and return a single value...
     return( sum( y-model(x,theta) ) )
}

thbar = 1.8
x = rep(c(1,3,7,8), len=100)
y = model(x,thbar) + rnorm(x)

plot(x, y, pch=20, cex=1.5, main="optim example 1: data")
points(x,model(x,thbar),col='green',pch=20,cex=2)

(optim.out <- optim(par=c(1), fn=crit, x=x, y=y,
     method="L-BFGS-B", lower=c(0.01), upper=c(3.05)))
```

Exercise: Why does it not "work"?????

Example: optim (continued)

```
crit <- function(theta,x,y){
# must have theta as 1st parameter and return a single value...
     return( sum( (y-model(x,theta))^2 ) )
}
thbar = 1.8
x = rep(c(1,3,7,8), len=100)
y = model(x,thbar) + rnorm(x)
(optim.out <- optim(par=c(1), fn=crit, x=x, y=y, method="L-BFGS-B",
     lower=c(0.01), upper=c(3.05)))

# Visualize objective shape and estimated optimum:
plot(x, y, pch=20, cex=1.5, main="optim example 2: data")
points(x, model(x,thbar), col='green', pch=20, cex=2)
ths = seq(0,5,len=50)
obj = 0*ths
for(i in 1:length(ths)){ obj[i] = crit(ths[i],x,y) }

plot(ths, obj, main="Criterion...", xlab="theta",
     ylab="Objective value")
points(optim.out$par, crit(optim.out$par,x,y),col='red',pch=20,cex=3)
```

Example: optim (continued again)

```
model.f <- function(x, theta){
     return(theta*x)
}
model.g <- function(x, theta){
gradient of model.f...
     return(x)
}
crit.f <- function(theta,x,y){
     return( sum( (y-model.f(x,theta))^2 ) )
}
crit.g <- function(theta,x,y){
# gradient of crit.f...
     return( sum( -2*model.g(x,theta)*(y-model.f(x,theta)) ) )
}

thbar = 1.8
x = rep(c(1,3,7,8), len=100)
y = model(x,thbar) + rnorm(x)
(optim.out <- optim(par=c(1), fn=crit.f, gr=crit.g, x=x, y=y,
     method="L-BFGS-B", lower=c(0.01), upper=c(3.05)))
...
```

Exercise: implement a simulation of the nonlinear model

$$Y = \exp(-\theta X) + \varepsilon$$

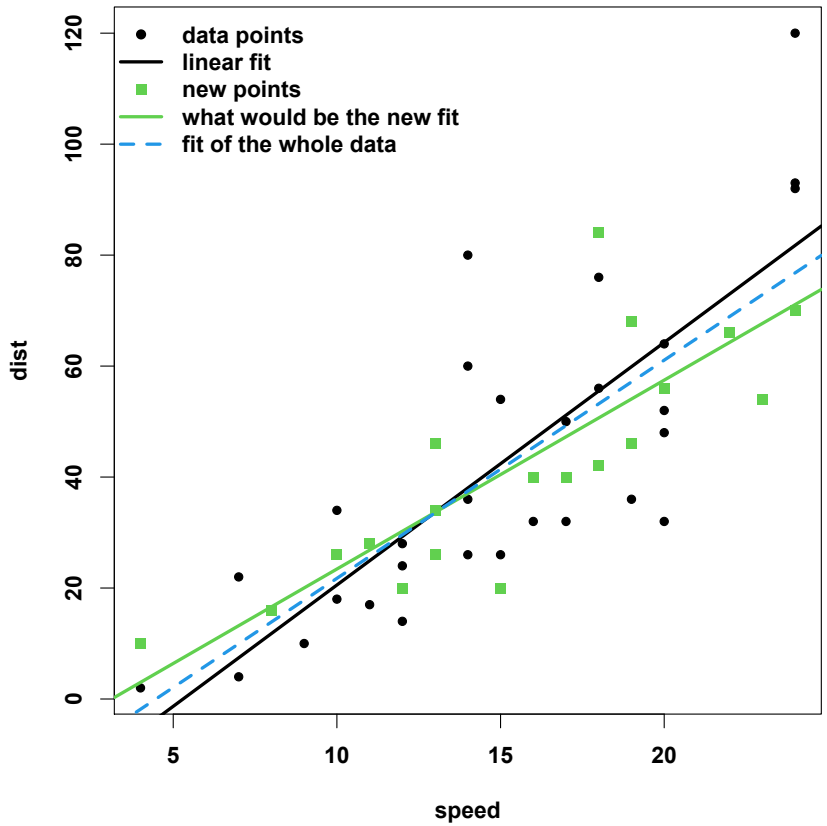
and minimize the sum of least squares for this model using `optim`.

Exercise: implement the same simulation and minimize the sum of least squares for this model using `optimx`, this time.

3.3 - Regularization

Regularization (shrinkage)

- Linear regression models easily **overfit** 'training' data



- Adding irrelevant covariates X artificially increases model fit
- *Shrinking* the less important covariates reduces overfitting
- Shrinking can also help reduce their variance

Ridge regression (shrinkage)

- Add an L_2 penalty on to the regression cost function
- *Shrinks* the effect of some variables
- Output coefficient estimates depend on the choice of tuning parameter λ and minimise the criterion

$$\sum_{i=1}^{N}\left(Y_i - \beta_0 - \sum_{j=1}^{p}\beta_j X_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

- λ is usually selected by cross-validation (tuning)
- Shrinkage is not applied to the intercept parameter
- The final model includes all p covariates

The LASSO

- Add an L_1 penalty on to the regression cost function
- Allows to effectively "get rid of" some variables
- Output coefficient estimates depend on the choice of tuning parameter λ and minimise the criterion

$$\sum_{i=1}^{N}\left(Y_i - \beta_0 - \sum_{j=1}^{p}\beta_j X_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

- λ is usually selected by cross-validation (tuning)
- Shrinkage is not applied to the intercept parameter
- The final model may have less than p covariates
- "Least Absolute Selection and Shrinkage Operator"

Tuning the regularization parameter

Output coefficient estimates depend on the choice of tuning parameter λ used in regularization (for some cost function $\rho(\beta)$)

$$\sum_{i=1}^{N}\left(Y_i-\beta_0-\sum_{j=1}^{p}\beta_j X_{ij}\right)^2+\lambda\sum_{j=1}^{p}\rho\left(\beta_j\right)=RSS+\lambda\sum_{j=1}^{p}\rho\left(\beta_j\right)$$

Cross-validation:

- Try several λ values on CV training sets
- Select λ that minimises CV MSE

```
library(glmnet, ISLR)
dat = na.omit(Hitters)
x = model.matrix(Salary~.,
data=dat)[,-1]
y = dat$Salary
plot( cv.glmnet(x, y) )
```