Bias-Variance Trade-off

YEGRUG 2023-03-30

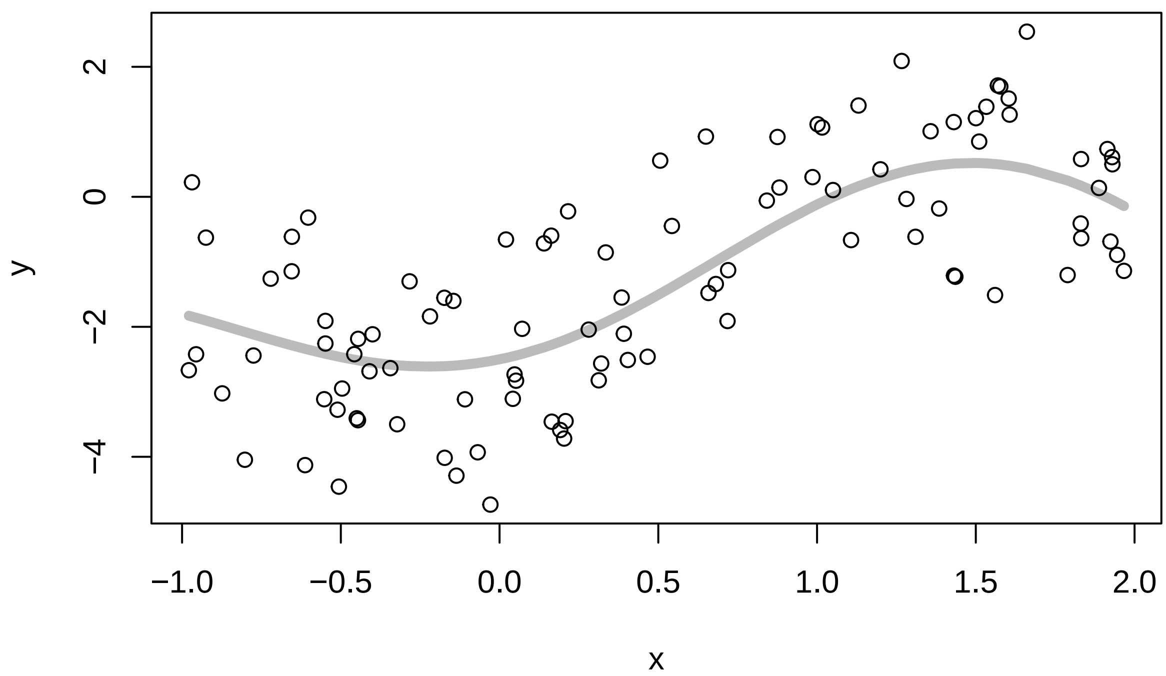
Install dependencies

```
install.packages("deps")
deps::install(ask=FALSE)
```

Linear regression

The data generating model

```
set.seed(2)
sig2 <- 1
n <- 100
x <- sort(runif(n, -1, 2))
mu <- sin(x) - 0.5*sin(x)^2 + 0.5*cos(x) - 3*cos(x)^2

eps <- rnorm(n, 0, sqrt(sig2))
y <- mu + eps

plot(y ~ x)
lines(mu ~ x, col="#00000044", lwd=5)
```



Regression

```
d <- data.frame(x=x, x2=x^2, x3=x^3, x4=x^4, x5=x^5, x6=x^6)

m0 <- lm(y ~ 1, d)
m1 <- lm(y ~ x, d)
m2 <- lm(y ~ x + x2, d)
m3 <- lm(y ~ x + x2 + x3, d)

summary(m0)
```

```
##
## Call:
## lm(formula = y ~ 1, data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4943 -1.4033  0.0195  1.3788  3.7813
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.2406     0.1744  -7.114  1.8e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.744 on 99 degrees of freedom
```

```
coef(m0)
```

```
## (Intercept)
```

```
##   -1.240554
```

```
summary(m0)$sigma
```

```
## [1] 1.743882
```

```
summary(m3)
```

```
##
## Call:
## lm(formula = y ~ x + x2 + x3, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.16661 -0.82031  0.00137  0.89402  2.32937
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.5610     0.1850 -13.840  < 2e-16 ***
## x             1.8395     0.2576   7.140 1.79e-10 ***
## x2            2.0649     0.3650   5.658 1.58e-07 ***
## x3           -1.2322     0.2148  -5.736 1.13e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.126 on 96 degrees of freedom
## Multiple R-squared:  0.5955, Adjusted R-squared:  0.5829
## F-statistic: 47.12 on 3 and 96 DF,  p-value: < 2.2e-16
```

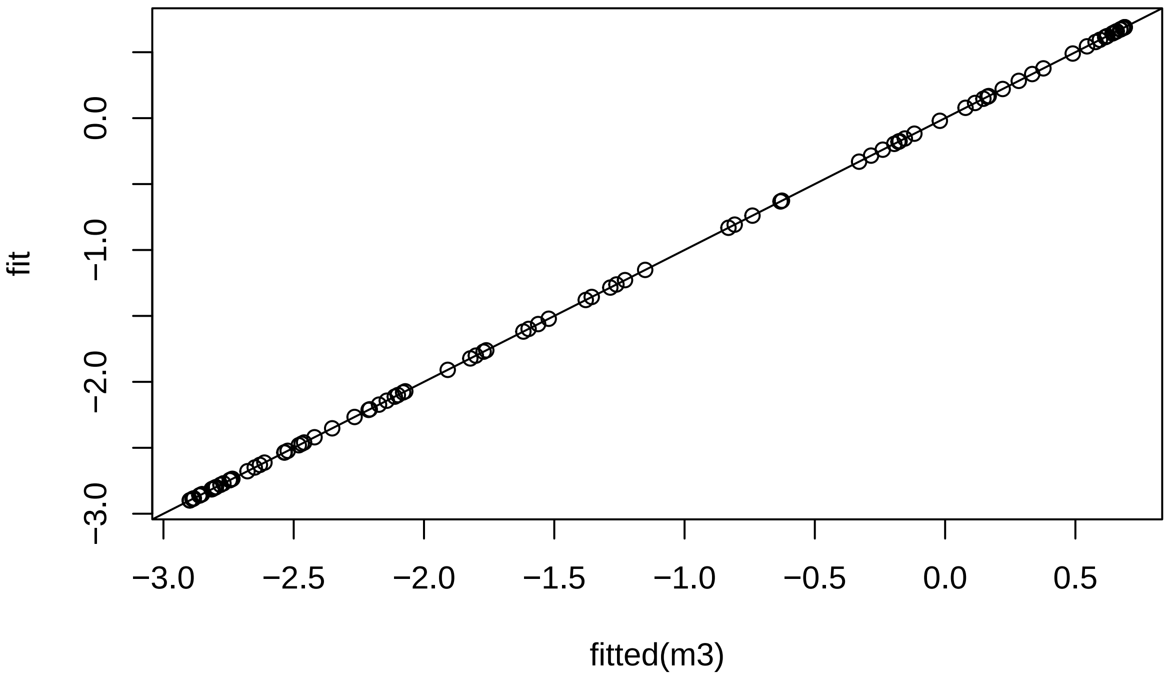
```
coef(m3)
```

```
## (Intercept)           x          x2          x3
##   -2.561035    1.839502    2.064854   -1.232220
```

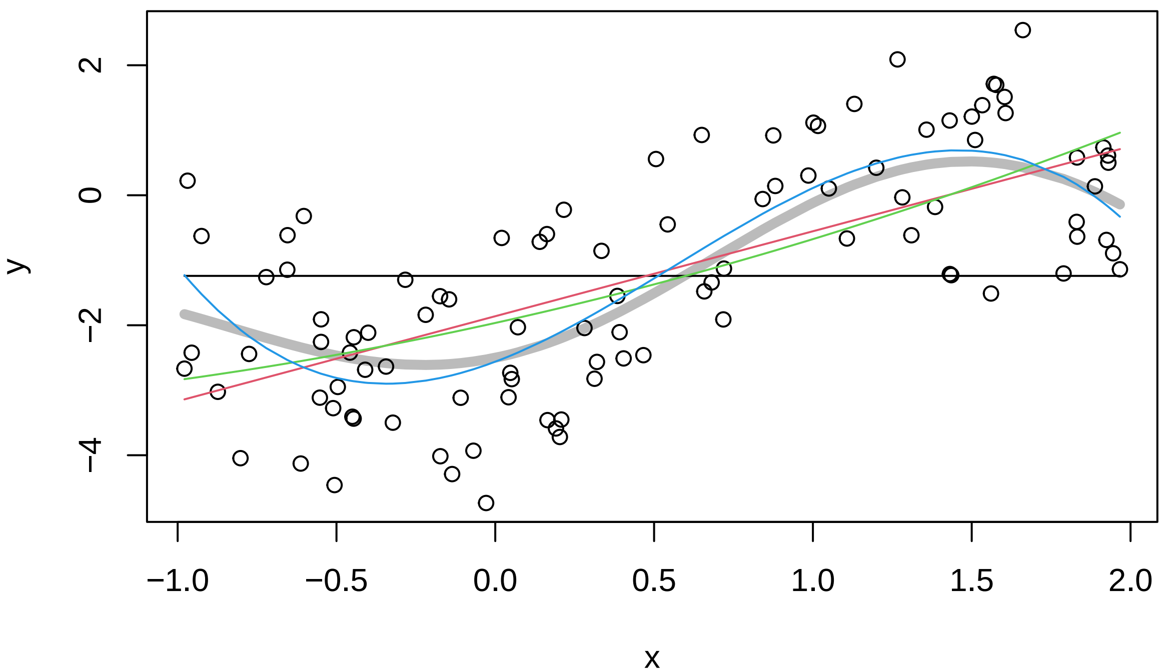
```
summary(m3)$sigma
```

```
## [1] 1.126274
```

```
fit <- coef(m3)[1] + coef(m3)[2]*x + coef(m3)[3]*x^2 + coef(m3)[4]*x^3
plot(fit ~ fitted(m3))
abline(0,1)
```




```
plot(y ~ x)
lines(mu ~ x, col="#00000044", lwd=5)
lines(fitted(m0) ~ x, col=1)
lines(fitted(m1) ~ x, col=2)
lines(fitted(m2) ~ x, col=3)
lines(fitted(m3) ~ x, col=4)
```



Uncertainty quantification

```
library(intrval)

predict_sim <-
function(object, newdata=NULL,
interval = c("none", "confidence", "prediction"),
type=c("asymp", "pboot", "npboot"),
level=0.95, B=99, ...) {
    interval <- match.arg(interval)
    type <- match.arg(type)
    if (is.null(newdata)) {
        x <- model.frame(object)
        X <- model.matrix(object)
    } else {
        x <- model.frame(delete.response(terms(object)), newdata)
        X <- model.matrix(attr(x, "terms"), x)
    }
    n <- nrow(x)
    fun <- switch(family(object)$family,
        "gaussian"=function(x) rnorm(length(x), x, summary(object)$sigma),
        "poisson"= function(x) rpois(length(x), x),
        "binomial"=function(x) rbinom(length(x), 1, x),
        stop("model family not recognized"))
    if (interval=="none")
        return(predict(object, newdata, ...))
    if (B < 2)
        stop("Are you kidding? B must be > 1")
    if (type == "asymp") {
        cm <- rbind(coef(object),
            MASS::mvrnorm(B, coef(object), vcov(object)))
        #fm <- apply(cm, 1, function(z) X %*% z)
    }
    if (type == "boot") {
        cm <- matrix(0, B+1, length(coef(object)))
        cm[1,] <- coef(object)
        xx <- model.frame(object)
        for (i in 2:B) {
            j <- sample.int(n, n, replace=TRUE)
            cm[i,] <- coef(update(object, data=xx[j,]))
        }
    }
    if (type == "npboot") {
        cm <- matrix(0, B+1, length(coef(object)))
        cm[1,] <- coef(object)
        xx <- model.frame(object)
        j <- attr(attr(xx, "terms"), "response")
```

```
        f <- fitted(object)
        for (i in 2:B) {
            xx[,j] <- fun(f)
            cm[i,] <- coef(update(object, data=xx))
        }
    }
    fm <- X %*% t(cm)
    fm <- family(object)$linkinv(fm)
    y <- if (interval == "prediction")
        matrix(fun(fm), n, B+1) else fm
    rownames(y) <- rownames(x)
    p <- c(0.5, (1-level) / 2, 1 - (1-level) / 2)
    stat_fun <- function(x)
        c(mean(x), sd(x), quantile(x, p))
    out <- cbind(fm[,1], t(apply(y, 1, stat_fun)))
    colnames(out) <- c("fit", "mean", "se", "median", "lwr", "upr")
    data.frame(out[,c("fit", "lwr", "upr", "mean", "median", "se")])
}

vcov(m0)
```

```
##             (Intercept)
## (Intercept)  0.03041123
```

```
coef(summary(m0))[,1:2,drop=FALSE]
```

```
##              Estimate Std. Error
## (Intercept) -1.240554  0.1743882
```

```
cbind(coef(m0), sqrt(vcov(m0)))
```

```
##                       (Intercept)
## (Intercept) -1.240554   0.1743882
```

```
vcov(m3)
```

```
##              (Intercept)            x          x2          x3
## (Intercept)  0.034240512 -0.005902869 -0.04787356  0.02233555
## x           -0.005902869  0.066380424  0.01221239 -0.02828937
## x2          -0.047873556  0.012212394  0.13320062 -0.06965248
## x3           0.022335546 -0.028289369 -0.06965248  0.04615640
```

```
coef(summary(m3))[,1:2]
```

```
##              Estimate Std. Error
## (Intercept) -2.561035  0.1850419
## x            1.839502  0.2576440
## x2           2.064854  0.3649666
## x3          -1.232220  0.2148404
```

```
cbind(coef(m3), sqrt(diag(vcov(m3))))
```

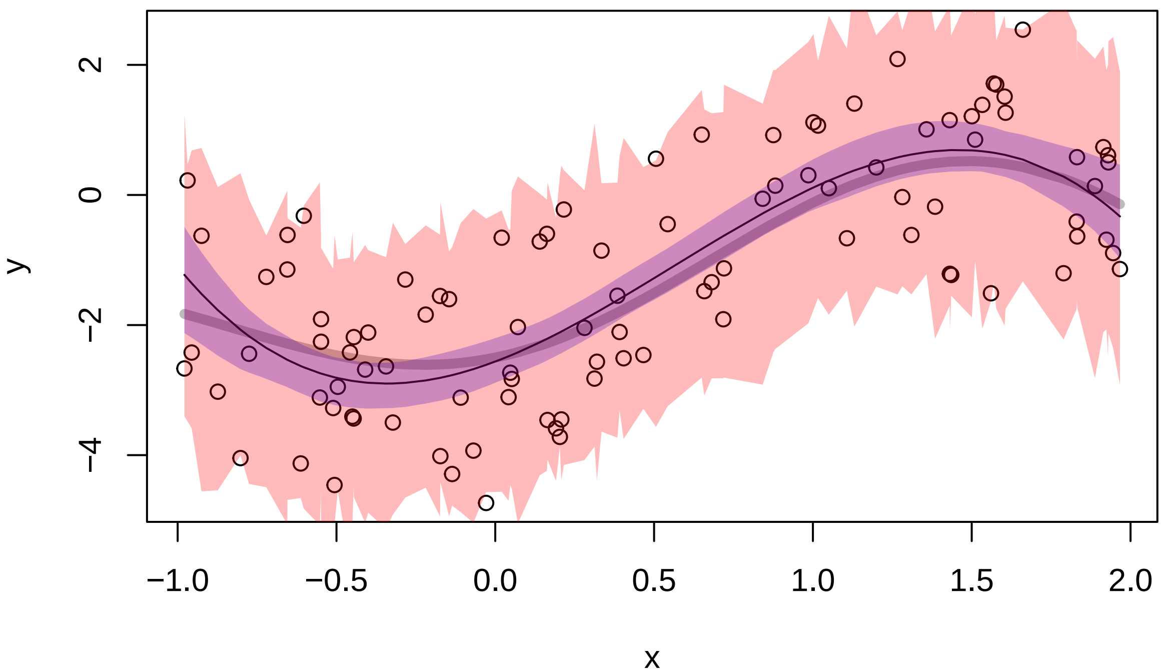
```
##                  [,1]      [,2]
## (Intercept) -2.561035 0.1850419
## x            1.839502 0.2576440
## x2           2.064854 0.3649666
## x3          -1.232220 0.2148404
```

Try this with different models and type as "asymp" or "npboot"

```
mod <- m3
type <- "asymp"
CI <- predict_sim(mod, newdata=d, interval="confidence", type=type)
PI <- predict_sim(mod, newdata=d, interval="prediction", type=type)

table(y %[]% PI[,c("lwr", "upr")])/n
```

```
##
## FALSE  TRUE
##  0.03  0.97
```

```
plot(y ~ x)
lines(mu ~ x, col="#00000044", lwd=5)
lines(fit ~ x, CI)
polygon(c(x, rev(x)), c(CI$lwr, rev(CI$upr)), border=NA, col="#0000ff44")
polygon(c(x, rev(x)), c(PI$lwr, rev(PI$upr)), border=NA, col="#ff000044")
```



The trade-off

```r
sim_fun <- function() {
    eps <- rnorm(n, 0, sqrt(sig2))
    y <- mu + eps
    y
}
pred_fun <- function(y, x) {
    m1 <- lm(y ~ x, d)
    m2 <- lm(y ~ x + x2, d)
    m3 <- lm(y ~ x + x2 + x3, d)
    m4 <- lm(y ~ x + x2 + x3 + x4, d)
    m5 <- lm(y ~ x + x2 + x3 + x4 + x5, d)
    m6 <- lm(y ~ x + x2 + x3 + x4 + x5 + x6, d)
    dnew <- data.frame(x=x, x2=x^2, x3=x^3, x4=x^4, x5=x^5, x6=x^6)
    sapply(list(m1=m1, m2=m2, m3=m3, m4=m4, m5=m5, m6=m6), function(z)
        predict(z, newdata=dnew))
}
vb_fun <- function(fit, i) {
    mu0 <- mu[i]
    Bias <- mean(fit) - unname(mu0)
    Var <- mean((fit - mean(fit))^2)
    c(Bias=Bias, Var=Var)
}

yobs <- sim_fun()
head(yobs)
```

```
## [1] -2.4324477 -0.3720748 -2.0240891 -0.9111539 -2.4725695 -2.0809067
```

```r
pr <- pred_fun(y = yobs, x = x[1])
pr
```

```
##       m1.1       m2.1       m3.1       m4.1       m5.1       m6.1
## -2.8014122 -2.3114848 -0.9604333 -1.2995357 -1.6563554 -1.5848237
```

```r
vb_fun(pr, i=1)
```

```
##       Bias        Var
## 0.06044438 0.38015855
```

```r
## using only a single observation
i <- 1
yall <- replicate(200,sim_fun())
fitall <- apply(yall, 2, function(z) pred_fun(z, x=x[i]))
vb <- apply(fitall, 1, vb_fun, i=1)
vb
```

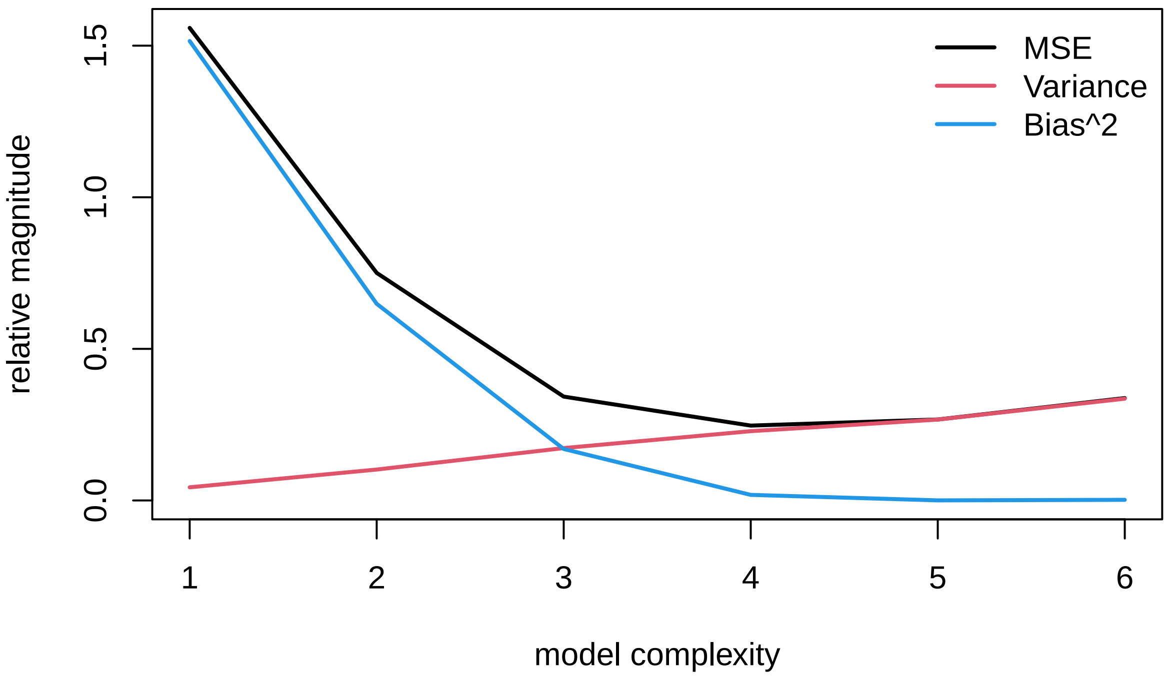
```
##             m1.1       m2.1      m3.1      m4.1       m5.1       m6.1
## Bias -1.23097061 -0.8053779 0.4120720 0.1353255 0.01663199 0.04485581
```

```
## Var   0.04330356  0.1019453 0.1727111 0.2285187 0.26668534 0.33561013
```

```
vb["Bias",]^2
```

```
##        m1.1        m2.1        m3.1        m4.1        m5.1        m6.1
## 1.515288633 0.648633482 0.169803299 0.018313003 0.000276623 0.002012044
```

```
plot(1:6, (vb["Bias",]^2 + vb["Var",]), type="l", lwd=2,
    ylim=c(0, max(vb["Bias",]^2 + vb["Var",])),
    xlab="model complexity", ylab="relative magnitude")
lines(1:6, vb["Var",], type="l", lwd=2, col=2)
lines(1:6, vb["Bias",]^2, type="l", lwd=2, col=4)
legend("topright", bty="n", col=c(1,2,4), lty=1, lwd=2,
    legend=c("MSE", "Variance", "Bias^2"))
```



A single realization

```
set.seed(3)
y <- sim_fun()
```

lm

```
m1 <- lm(y ~ x, d)
m2 <- lm(y ~ x + x2, d)
m3 <- lm(y ~ x + x2 + x3, d)
m4 <- lm(y ~ x + x2 + x3 + x4, d)
m5 <- lm(y ~ x + x2 + x3 + x4 + x5, d)
m6 <- lm(y ~ x + x2 + x3 + x4 + x5 + x6, d)
```

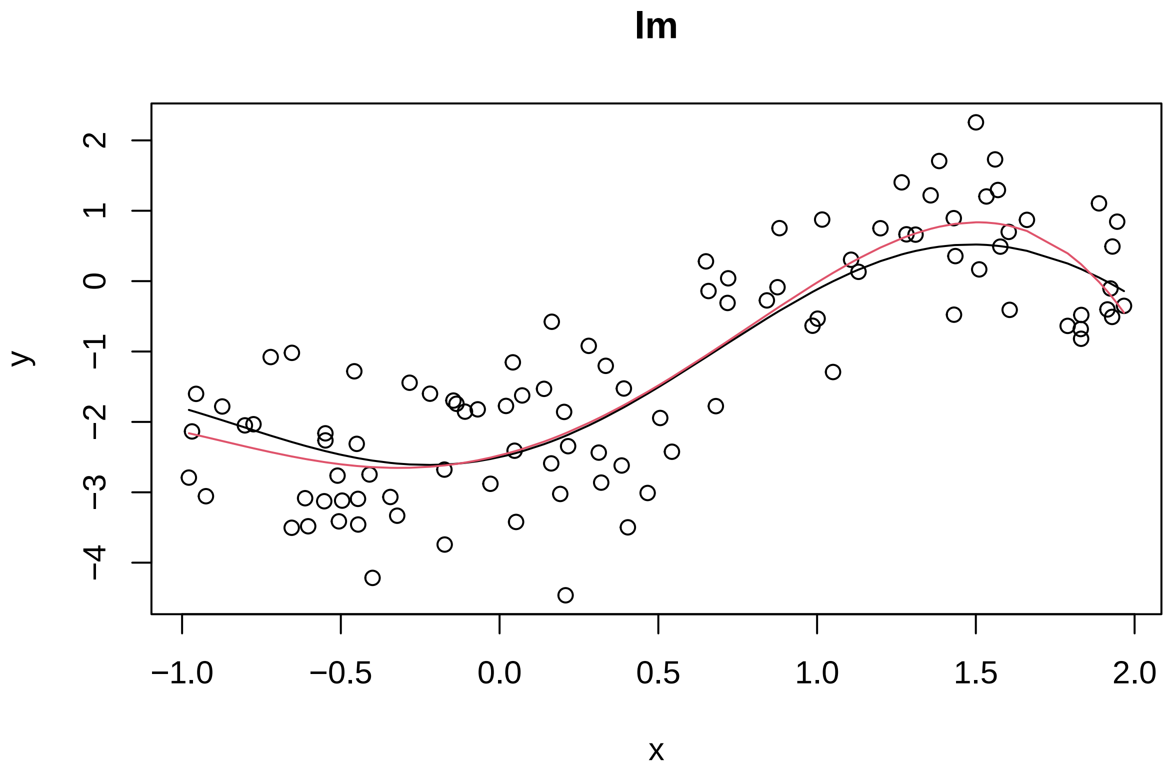
```r
aic <- AIC(m1, m2, m3, m4, m5, m6)
aic$dAIC <- aic$AIC - min(aic$AIC)
aic
```

```
##    df      AIC      dAIC
## m1  3 291.6551 31.184485
## m2  4 292.6840 32.213415
## m3  5 263.2378  2.767198
## m4  6 260.4706  0.000000
## m5  7 262.1736  1.702987
## m6  8 262.0424  1.571799
```

```r
best <- list(m1, m2, m3, m4, m5, m6)[[which.min(aic$AIC)]]
summary(best)
```

```
##
## Call:
## lm(formula = y ~ x + x2 + x3 + x4, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.30278 -0.63987  0.03388  0.72359  1.66186
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.47572    0.14176 -17.465  < 2e-16 ***
## x            1.13984    0.36415   3.130  0.00232 **
## x2           1.84706    0.29334   6.297 9.38e-09 ***
## x3          -0.07993    0.45557  -0.175  0.86110
## x4          -0.45124    0.20952  -2.154  0.03379 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8599 on 95 degrees of freedom
## Multiple R-squared:  0.7152, Adjusted R-squared:  0.7032
## F-statistic: 59.63 on 4 and 95 DF,  p-value: < 2.2e-16
```

```r
plot(y ~ x, main="lm")
lines(x, mu)
lines(x, fitted(best), col=2)
```



gam

```
library(mgcv)
```

```
## Loading required package: nlme
```

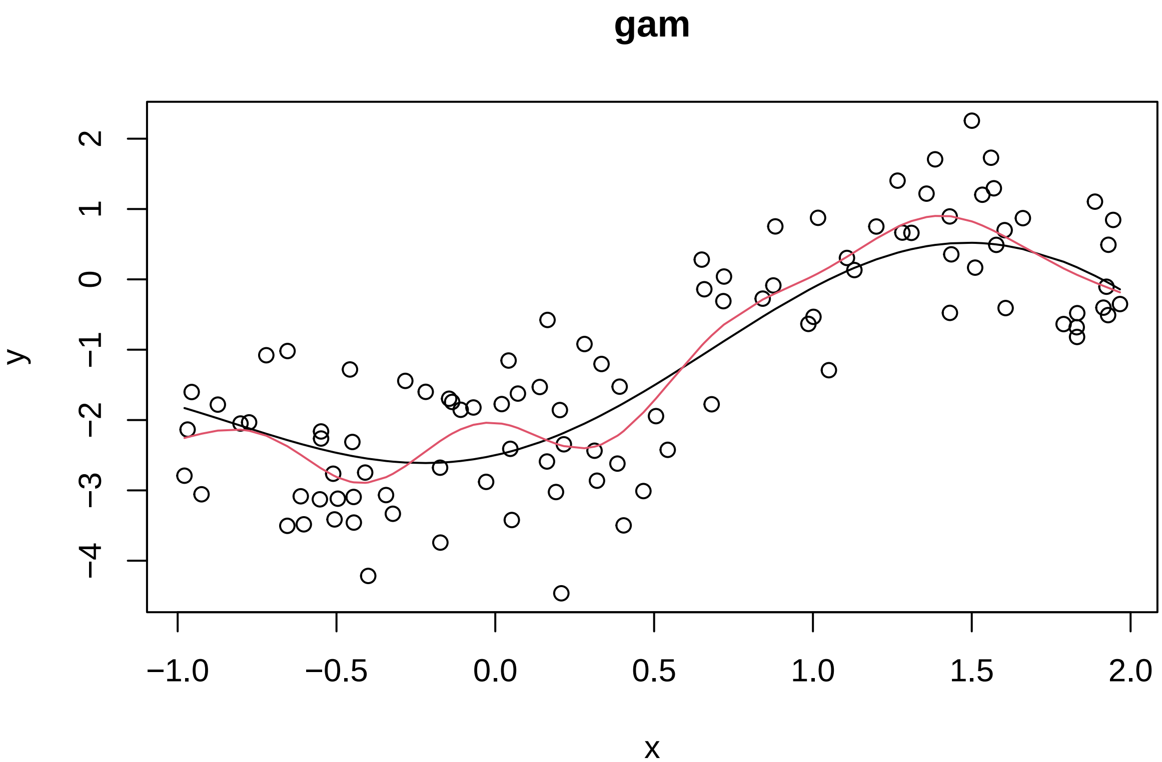
```
## This is mgcv 1.8-40. For overview type 'help("mgcv-package")'.
```

```
mod <- mgcv::gam(y ~ s(x), family=gaussian)
summary(mod)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(x)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.25353    0.08235  -15.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##        edf Ref.df     F p-value
```

```
## s(x) 8.041  8.748 30.6  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.728   Deviance explained =   75%
## GCV = 0.74557  Scale est. = 0.67816   n = 100
```

```r
plot(y ~ x, main="gam")
lines(x, mu)
lines(x, fitted(mod), col=2)
```



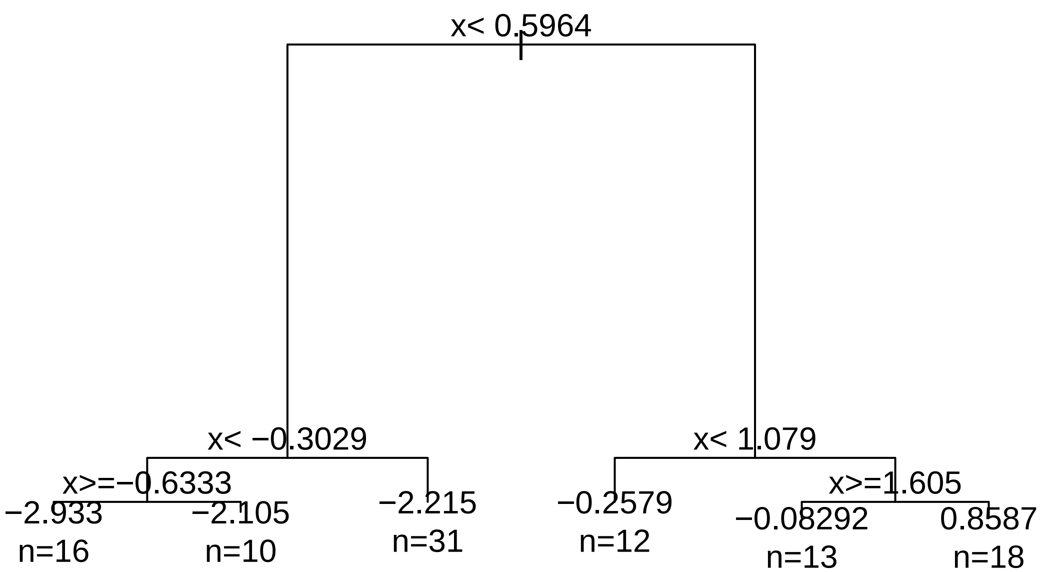
rpart/ctree

```r
library(rpart)
fit <- rpart(y ~ x, method="anova")
fit
```

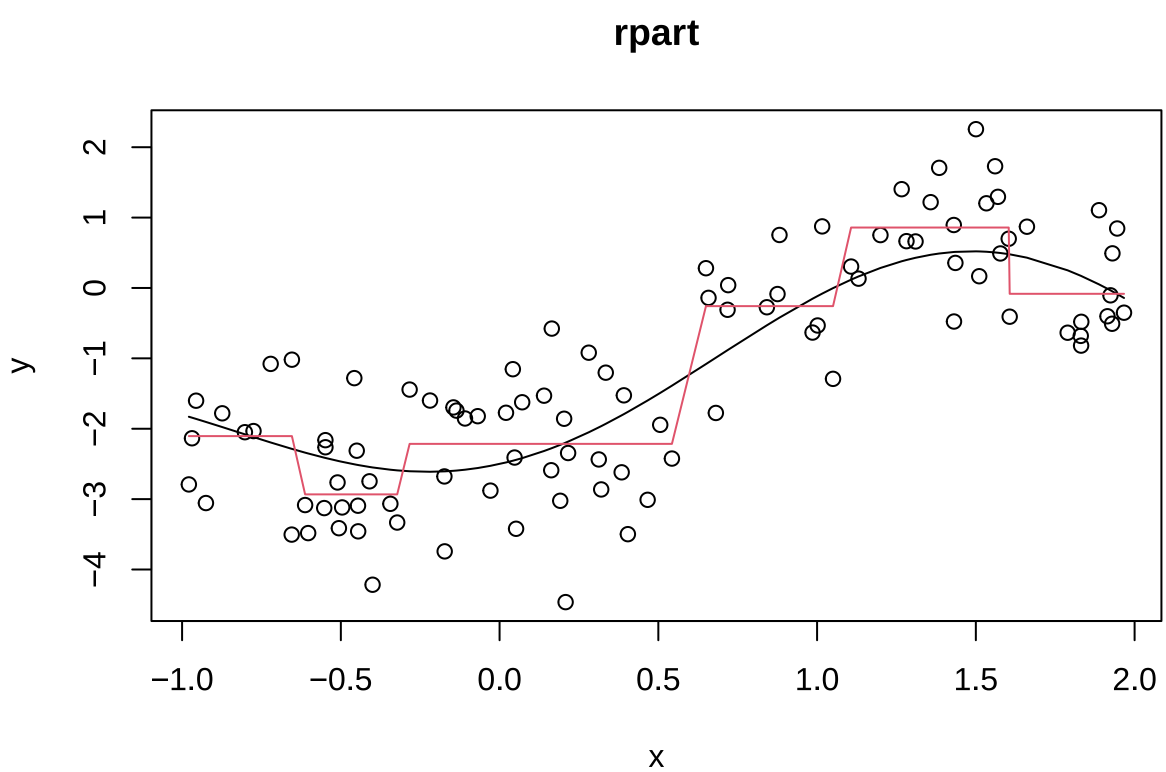
```
## n= 100
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 100 246.607800 -1.25353200
##    2) x< 0.5963983 57  42.625820 -2.39713800
##      4) x< -0.3029197 26  17.123580 -2.61420000
##        8) x>=-0.6332978 16   6.973814 -2.93256000 *
```

```
##         9) x< -0.6332978 10   5.933475 -2.10482400 *
##       5) x>=-0.3029197 31  23.249790 -2.21508500 *
##     3) x>=0.5963983 43  30.617870  0.26241080
##       6) x< 1.078666 12   6.318963 -0.25791030 *
##       7) x>=1.078666 31  19.792500  0.46382540
##        14) x>=1.605059 13   5.335466 -0.08292458 *
##        15) x< 1.605059 18   7.764199  0.85870040 *
```

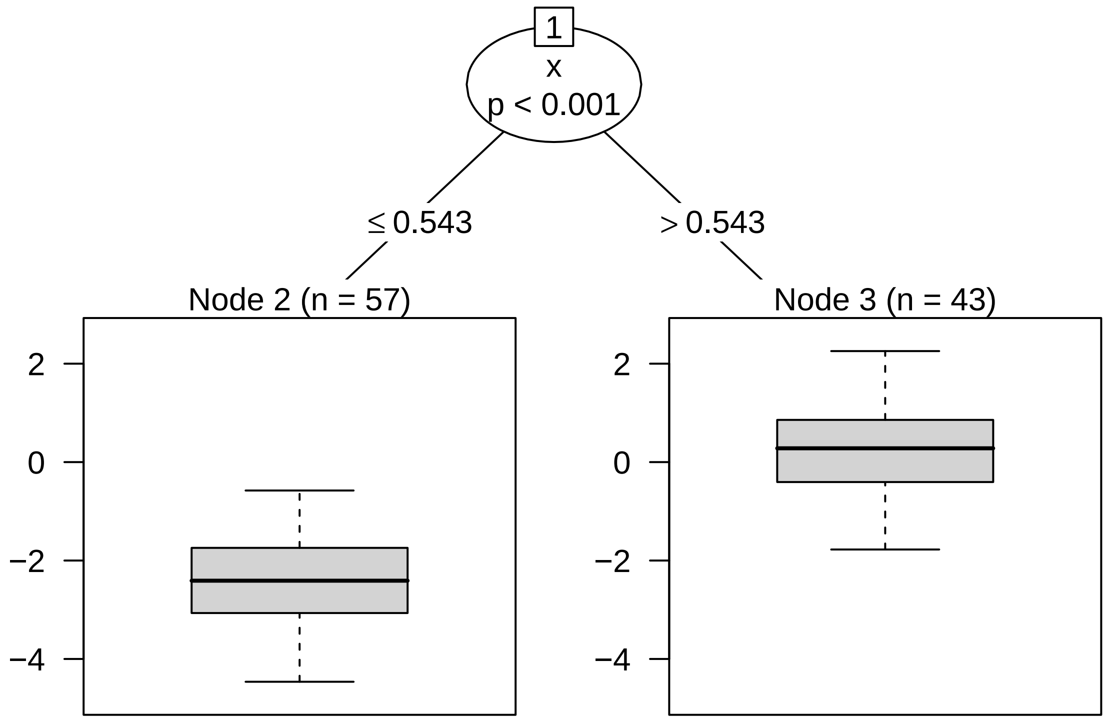
```r
pr <- predict(fit, newdata=d)

par(xpd = TRUE)
plot(fit, compress = TRUE)
text(fit, use.n = TRUE)
```



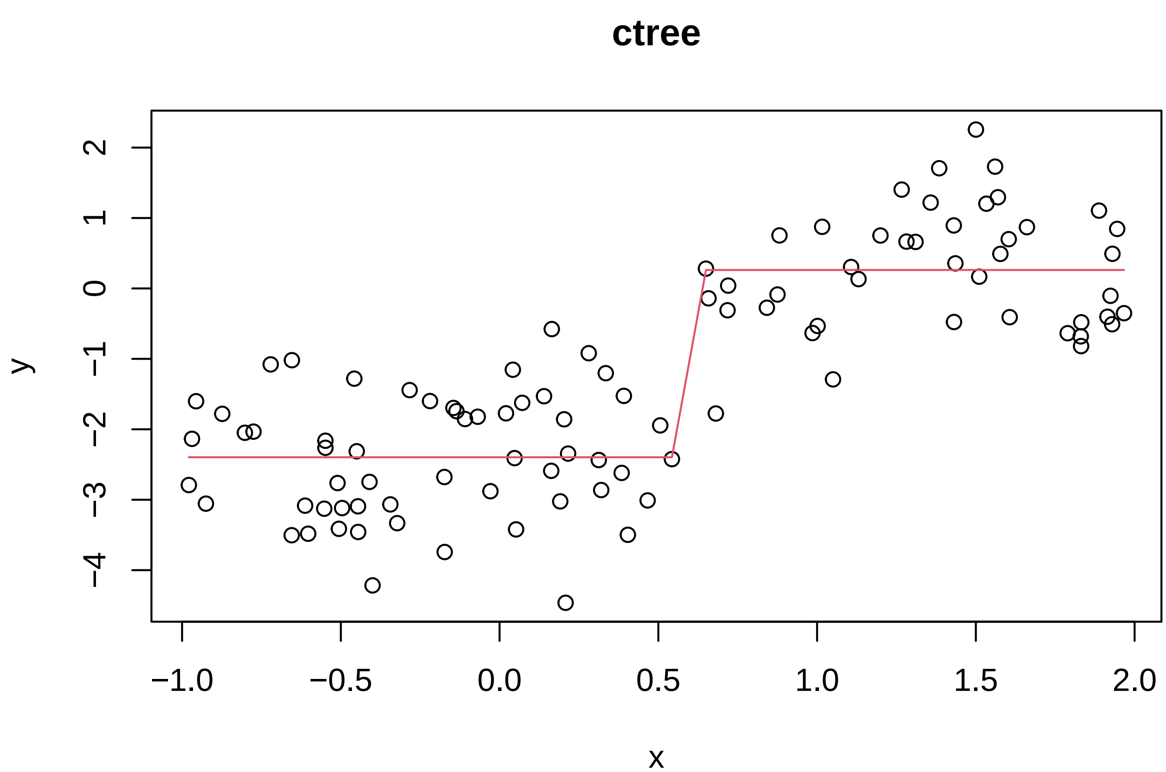

```r
plot(y ~ x, main="rpart")
lines(x, mu)
lines(x, pr, col=2)
```




```
library(partykit)

## Loading required package: grid

## Loading required package: libcoin

## Loading required package: mvtnorm

fit2 <- ctree(y ~ x)
fit2

##
## Model formula:
## y ~ x
##
## Fitted party:
## [1] root
## |   [2] x <= 0.54285: -2.397 (n = 57, err = 42.6)
## |   [3] x > 0.54285: 0.262 (n = 43, err = 30.6)
##
## Number of inner nodes:    1
## Number of terminal nodes: 2

pr2 <- predict(fit2, newdata=d)

plot(fit2)
```




```
plot(y ~ x, main="ctree")
#lines(x, mu)
lines(x, pr2, col=2)
```



gbm

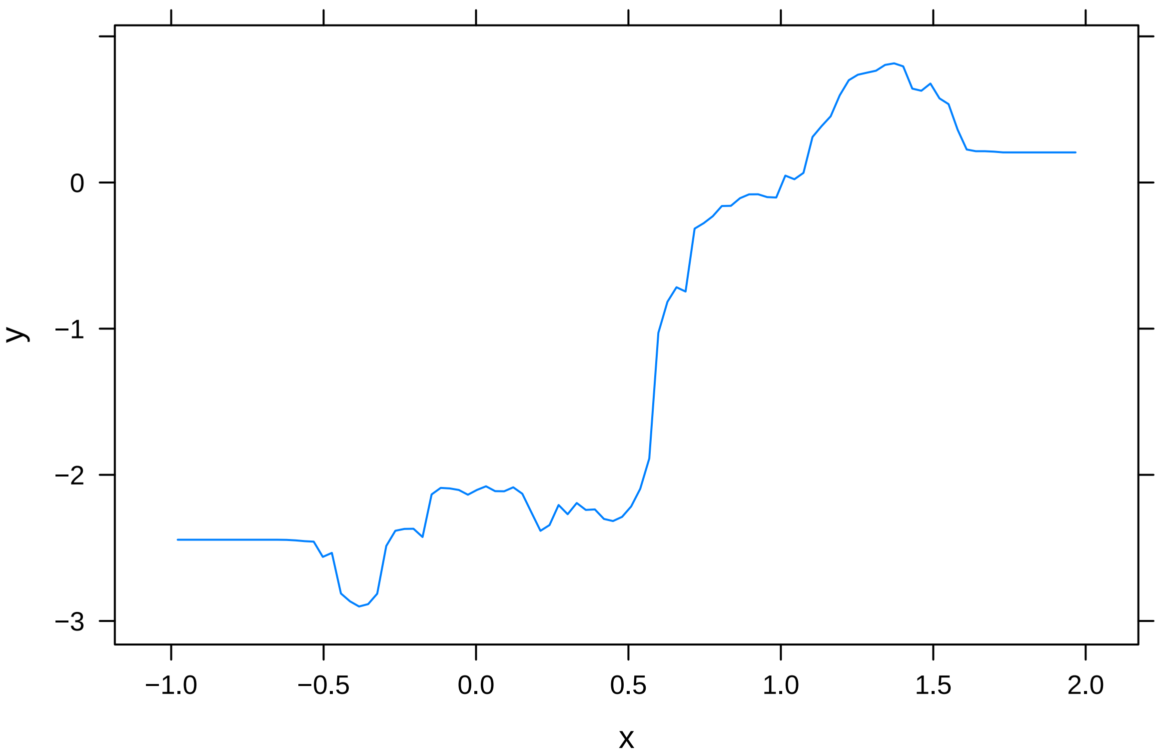
```r
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

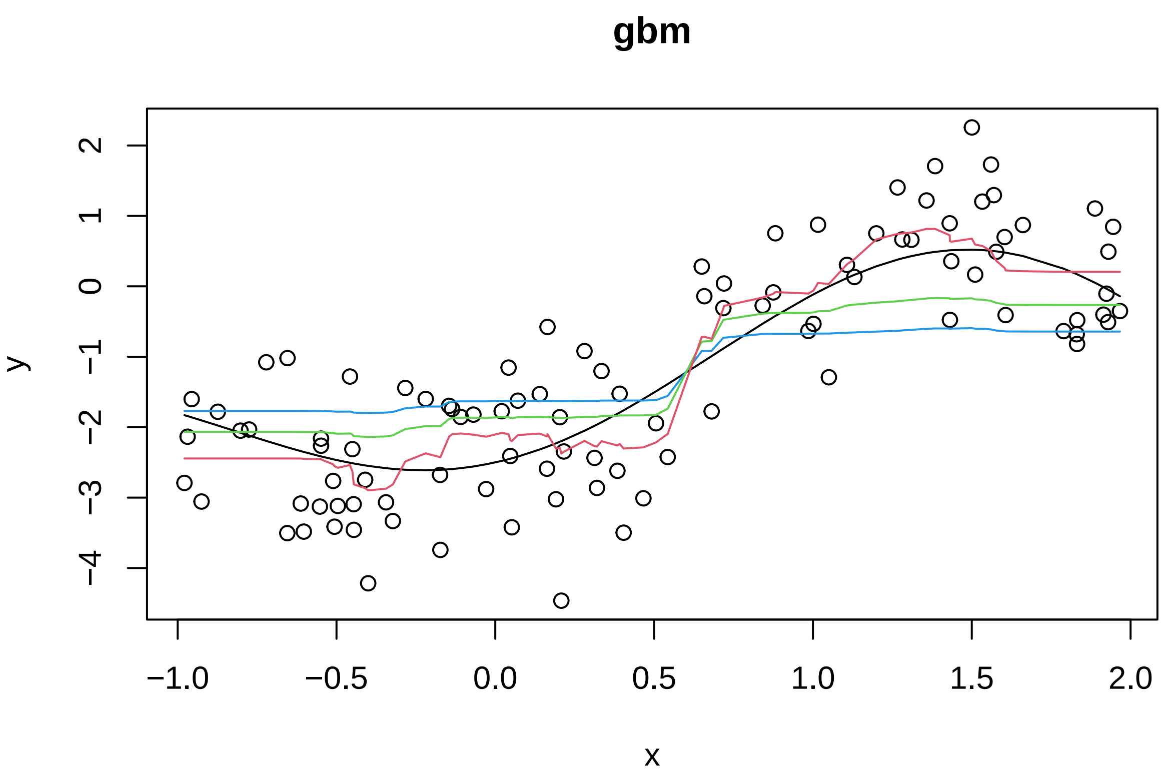
```r
brt <- gbm(y ~ x, distribution="gaussian",
    interaction.depth=5, shrinkage = 0.0001, n.trees = 50000)
brt
```

```
## gbm(formula = y ~ x, distribution = "gaussian", n.trees = 50000,
##     interaction.depth = 5, shrinkage = 1e-04)
## A gradient boosted model with gaussian loss function.
## 50000 iterations were performed.
## There were 1 predictors of which 1 had non-zero influence.
```

```r
pr3 <- predict(brt, newdata=d, n.trees=c(5000, 10000, 50000), type="response")

plot(brt)
```




```r
plot(y ~ x, main="gbm")
lines(x, mu)
matlines(x, pr3, col=c(4,3,2), lty=1)
```



glmnet

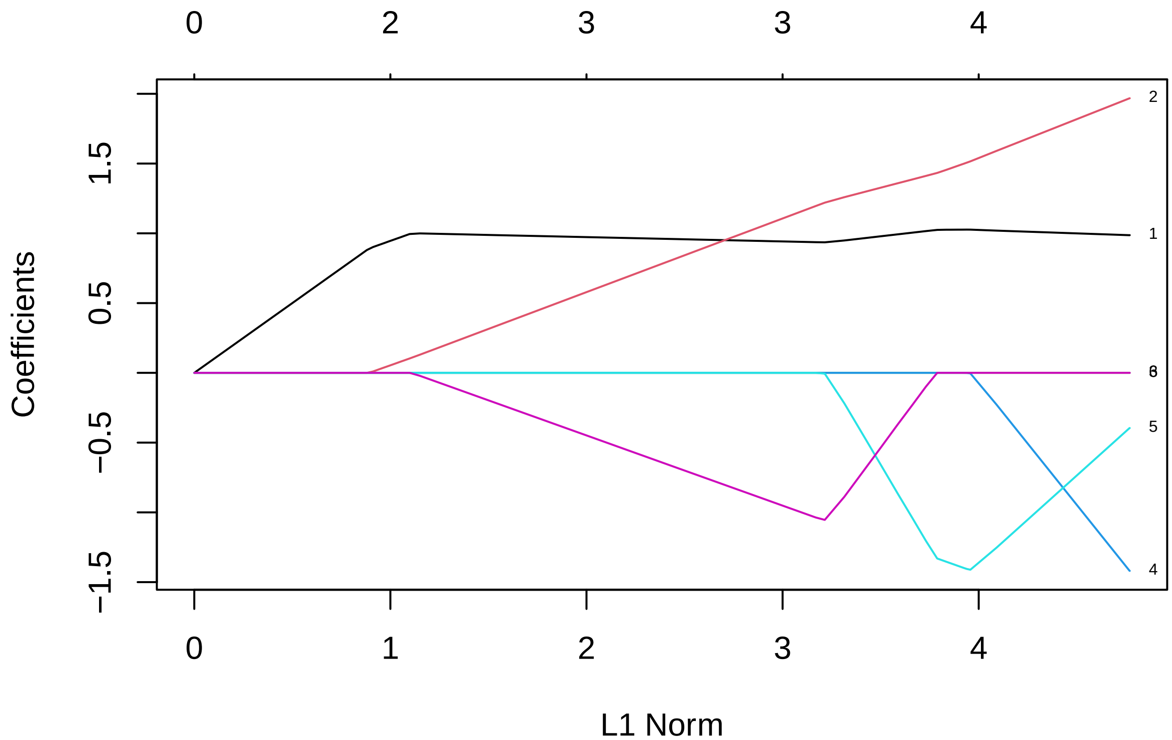
https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

```
library(glmnet)
```

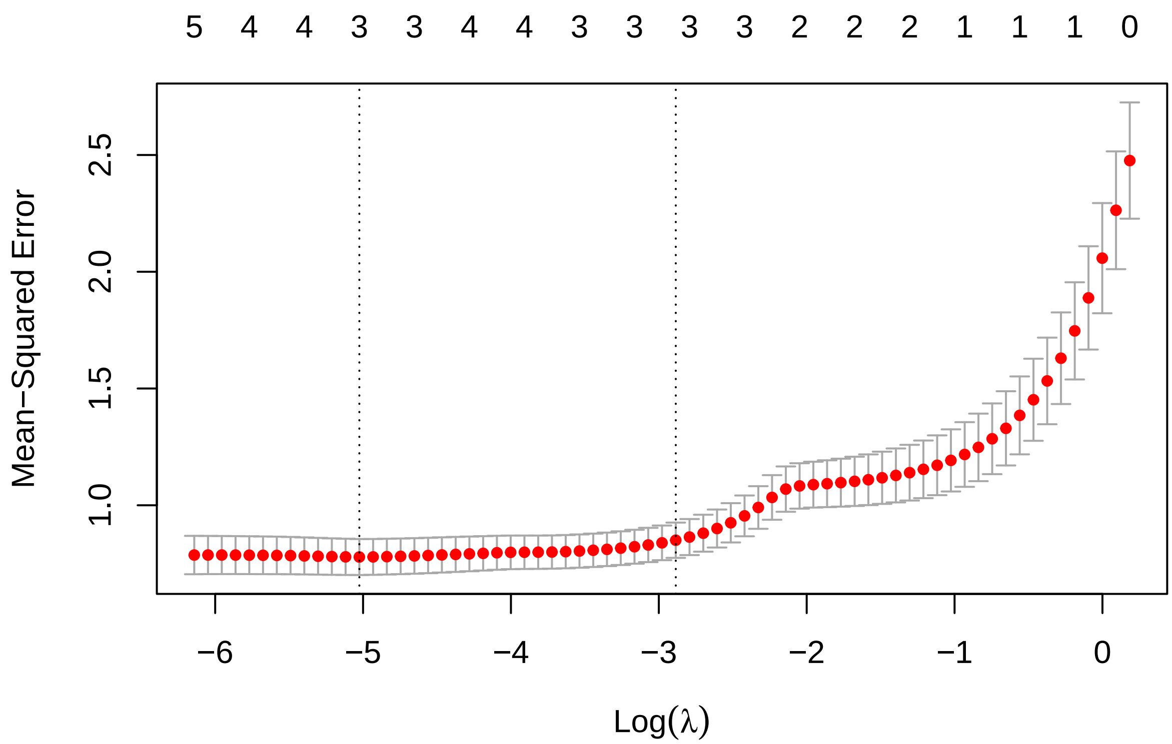
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-7
```

```
xx <- scale(as.matrix(d))
gn <- glmnet(xx, y, alpha=1) # alpha=1 is LASSO
plot(gn, label=TRUE)
```

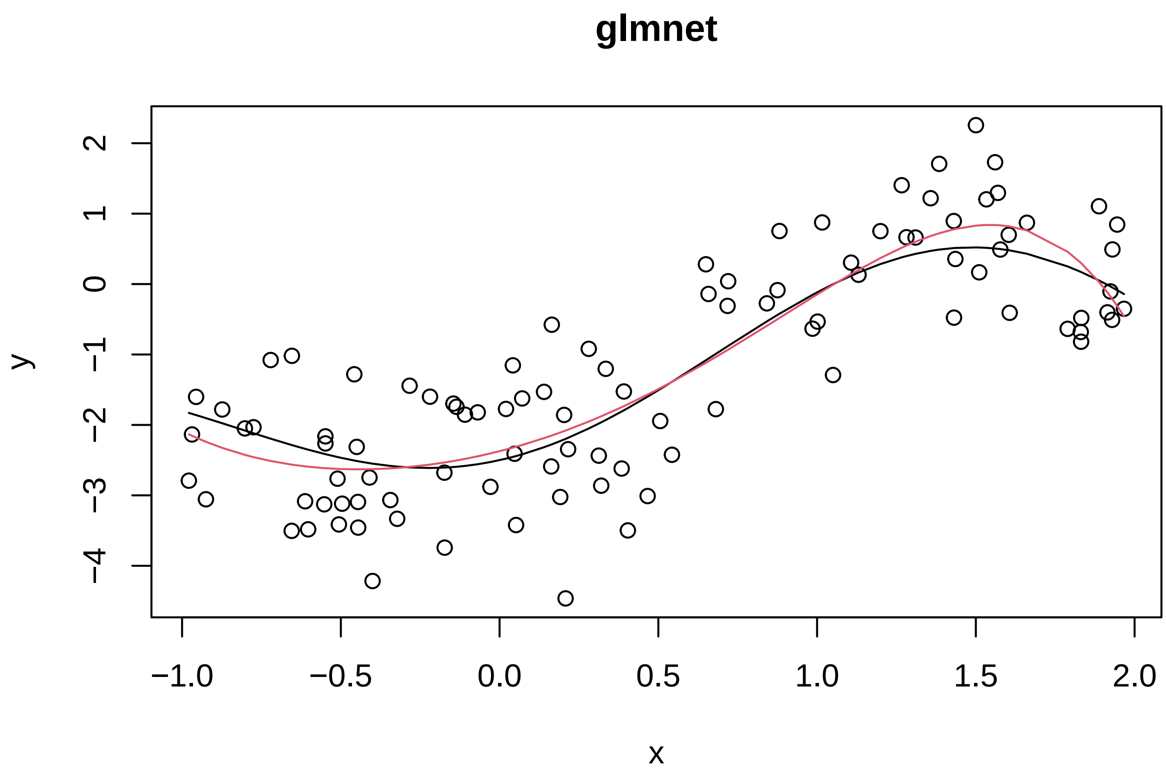



```
cv <- cv.glmnet(xx, y)
plot(cv)
```




```
pr4 <- predict(cv, newx=xx, type="response", s=cv$lambda.min)

plot(y ~ x, main="glmnet")
```

```
lines(x, mu)
lines(x, pr4, col=2)
```



Bootstrap

```
set.seed(123)
B <- 200
BB <- cbind(1:n, replicate(B-1, sample.int(n, n, replace=TRUE)))
dim(BB)
```

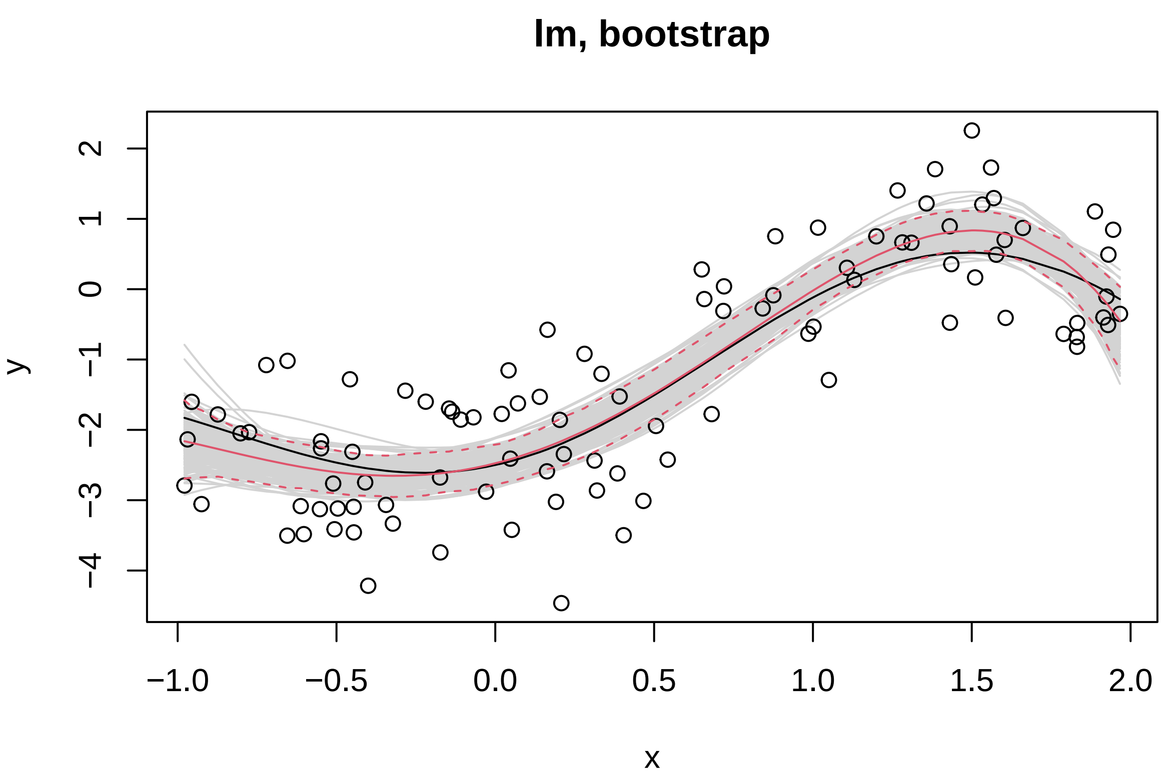
```
## [1] 100 200
```

```
m1 <- lm(y ~ x, d)
m2 <- lm(y ~ x + x2, d)
m3 <- lm(y ~ x + x2 + x3, d)
m4 <- lm(y ~ x + x2 + x3 + x4, d)
m5 <- lm(y ~ x + x2 + x3 + x4 + x5, d)
m6 <- lm(y ~ x + x2 + x3 + x4 + x5 + x6, d)
aic <- AIC(m1, m2, m3, m4, m5, m6)
aic$dAIC <- aic$AIC - min(aic$AIC)
aic
```

```
##    df      AIC      dAIC
## m1  3 291.6551 31.184485
## m2  4 292.6840 32.213415
## m3  5 263.2378  2.767198
```

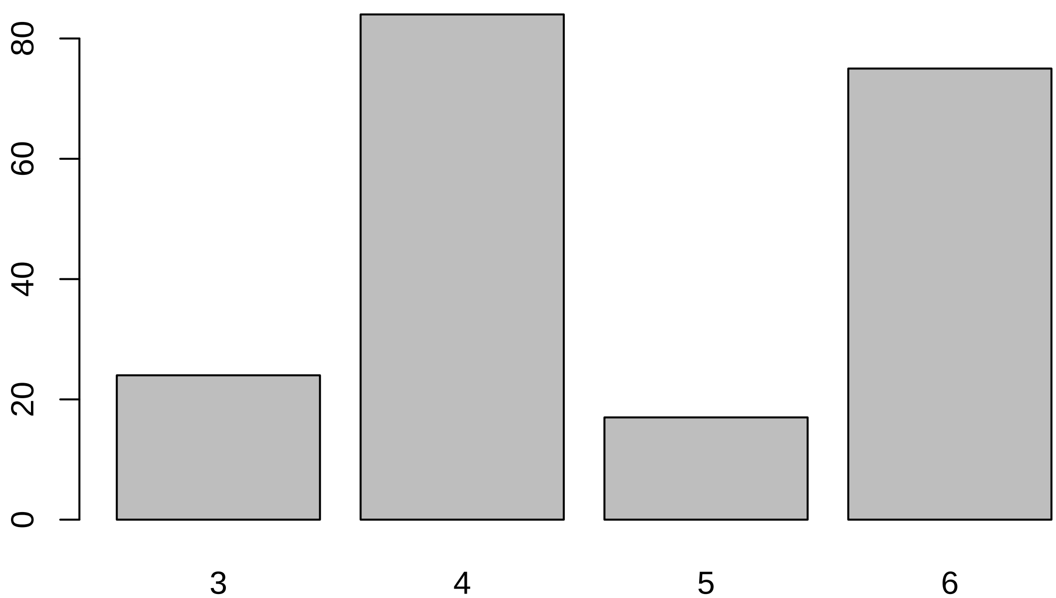
```
## m4   6 260.4706  0.000000
## m5   7 262.1736  1.702987
## m6   8 262.0424  1.571799
```

```r
best <- list(m1, m2, m3, m4, m5, m6)[[which.min(aic$AIC)]]

dd <- model.frame(best)

pr_mat <- matrix(0, 100, B)
for (i in 1:B) {
    mb <- lm(y ~ x + x2 + x3 + x4, dd[BB[,i],])
    pr_mat[,i] <- predict(mb, newdata=d)
}
pr_int <- apply(pr_mat, 1, quantile, probs=c(0.025, 0.975))

plot(y ~ x, main="lm, bootstrap", type="n")
matlines(x, pr_mat, lty=1, col="lightgrey")
points(x, y)
lines(x, mu)
lines(x, pr_mat[,1], col=2)
lines(x, pr_int[1,], col=2, lty=2)
lines(x, pr_int[2,], col=2, lty=2)
```



Bagging

```
ddd <- model.frame(m6)

bag_fun <- function(i) {
    mb1 <- lm(y ~ x, ddd[BB[,i],])
    mb2 <- lm(y ~ x + x2, ddd[BB[,i],])
    mb3 <- lm(y ~ x + x2 + x3, ddd[BB[,i],])
    mb4 <- lm(y ~ x + x2 + x3 + x4, ddd[BB[,i],])
    mb5 <- lm(y ~ x + x2 + x3 + x4 + x5, ddd[BB[,i],])
    mb6 <- lm(y ~ x + x2 + x3 + x4 + x5 + x6, ddd[BB[,i],])
    aic <- AIC(mb1, mb2, mb3, mb4, mb5, mb6)
    j <- which.min(aic$AIC)
    best <- list(mb1, mb2, mb3, mb4, mb5, mb6)[[j]]
    pr <- predict(best, newdata=d)
    attr(pr, "mid") <- j
    pr
}
library(pbapply)
res <- pblapply(1:B, bag_fun)

mid <- sapply(res, attr, "mid")
barplot(table(mid))
```




```
pr_mat <- do.call(cbind, res)
pr_int <- apply(pr_mat, 1, quantile, probs=c(0.025, 0.975))

plot(y ~ x, main="lm, bagging", type="n")
matlines(x, pr_mat, lty=1, col="lightgrey")
points(x, y)
lines(x, mu)
```

```
lines(x, rowMeans(pr_mat), col=2)
lines(x, pr_int[1,], col=2, lty=2)
lines(x, pr_int[2,], col=2, lty=2)
```