STAT 8020 R Lab 10: Adavnced Topics I

Whitney

September 23, 2020

Contents

Nonlinear Regression

U.S. Population Example

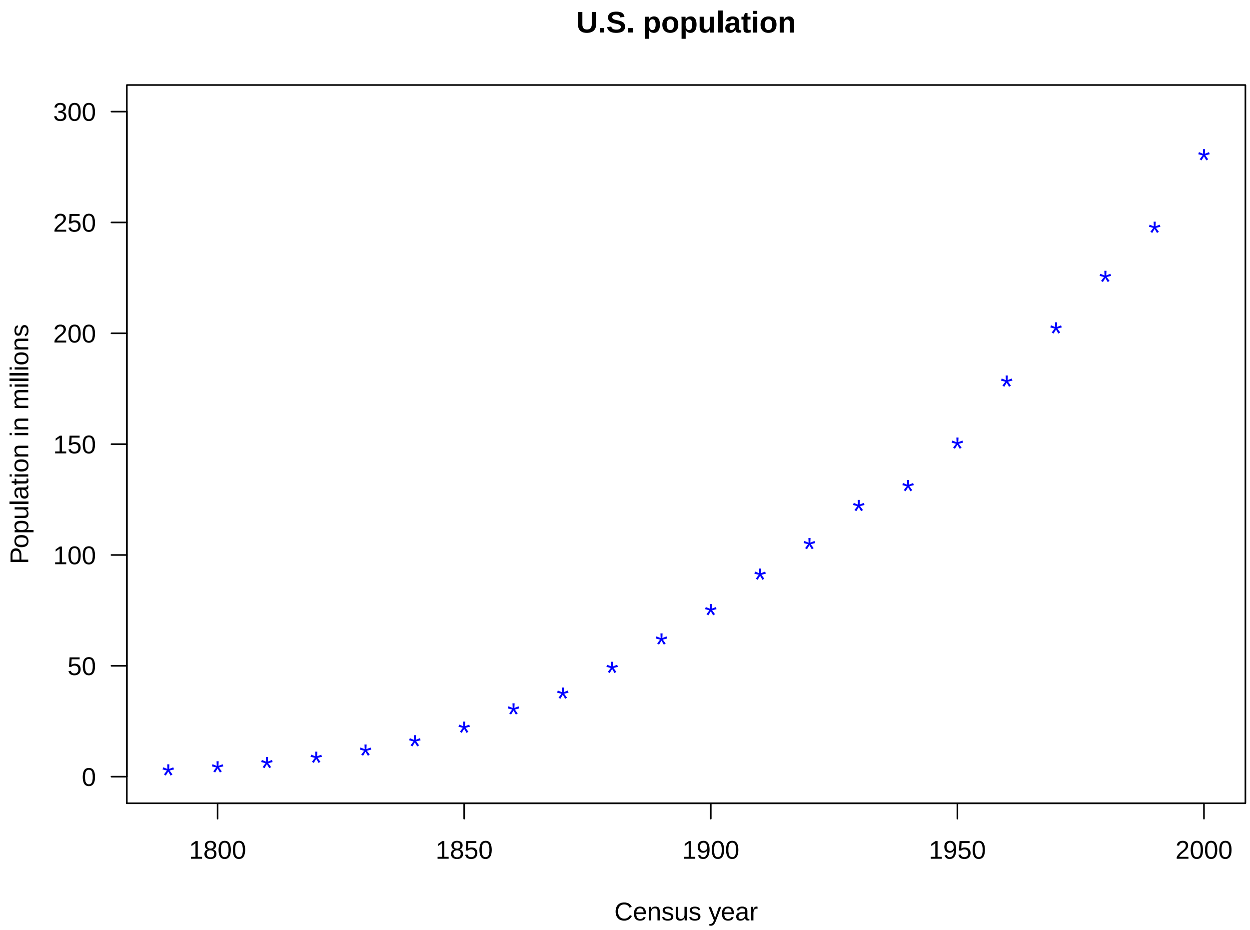
```r
library(car)
```

```
## Warning: package 'car' was built under R version 3.6.2
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 3.6.2
```

```r
plot(population ~ year, data = USPop,
     main = "U.S. population",
     ylim = c(0, 300),pch = "*",
     xlab = "Census year",
     ylab = "Population in millions",
     cex = 1.25, las = 1, col = "blue")
```



Logistic growth curve

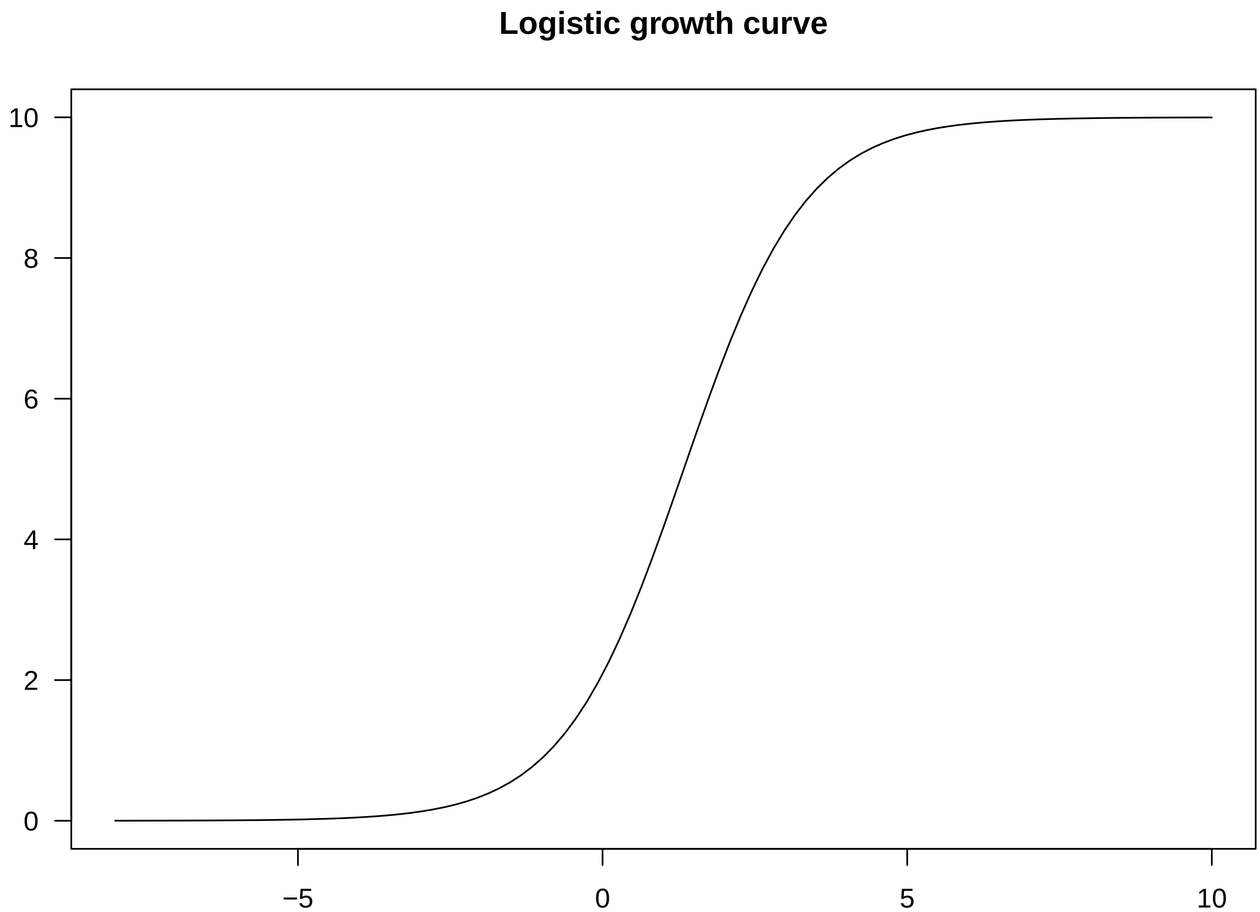
A logistic function is a symmetric S shape curve with equation:

$$f(x) = \frac{\phi_1}{1 + \exp(-(x - \phi_2)/\phi_3)}$$

where ϕ_1 is the curve's maximum value; ϕ_2 is the curve's midpoint in x; and ϕ_3 is the "range" (or the inverse growth rate) of the curve.

One typical application of the logistic equation is to model population growth.

```
# phi_1 = 10; phi_2 = 4/3, phi_3 = 1
curve(10 / (1 + exp(-(x - 4/3))), from = -8, to = 10, main = "Logistic growth curve", las = 1, xlab = "
```



Fit a logistic growth curve to the U.S. population data set

```
pop.ss <- nls(population ~ SSlogis(year, phi1, phi2, phi3), data = USPop)
summary(pop.ss)
```

```
##
## Formula: population ~ SSlogis(year, phi1, phi2, phi3)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## phi1  440.833     35.000   12.60 1.14e-10 ***
## phi2 1976.634      7.556  261.61  < 2e-16 ***
## phi3   46.284      2.157   21.45 8.87e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.909 on 19 degrees of freedom
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 6.818e-07
```

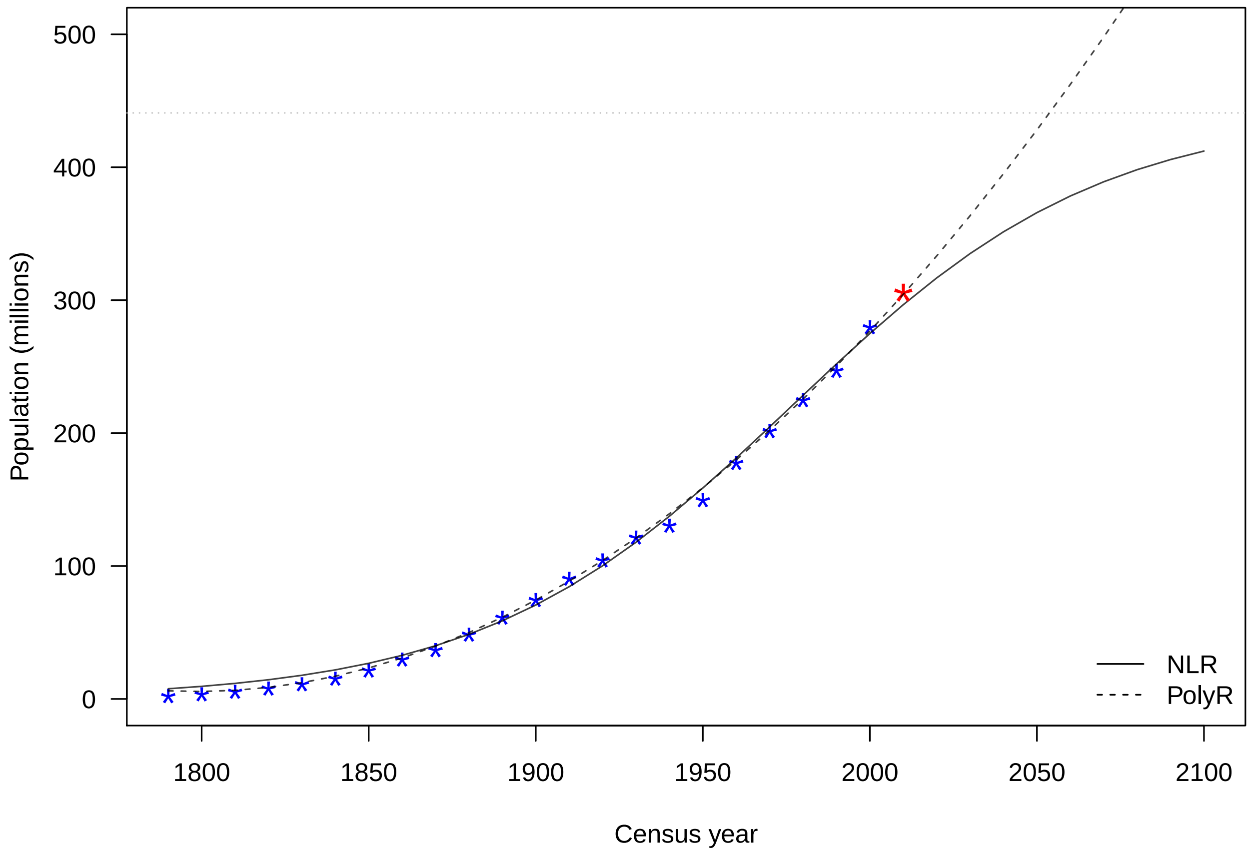
Alternative model: fit a quadratic polynomial

```r
pop.qm <- lm(population ~ year + I(year^2),
         USPop)
summary(pop.qm)
```

```
##
## Call:
## lm(formula = population ~ year + I(year^2), data = USPop)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5557 -0.4308  0.6051  1.4230  4.6486
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.162e+04  6.389e+02   33.83   <2e-16 ***
## year        -2.403e+01  6.749e-01  -35.61   <2e-16 ***
## I(year^2)    6.681e-03  1.780e-04   37.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.997 on 19 degrees of freedom
## Multiple R-squared:  0.9989, Adjusted R-squared:  0.9988
## F-statistic:  8892 on 2 and 19 DF,  p-value: < 2.2e-16
```

Comparing the fits

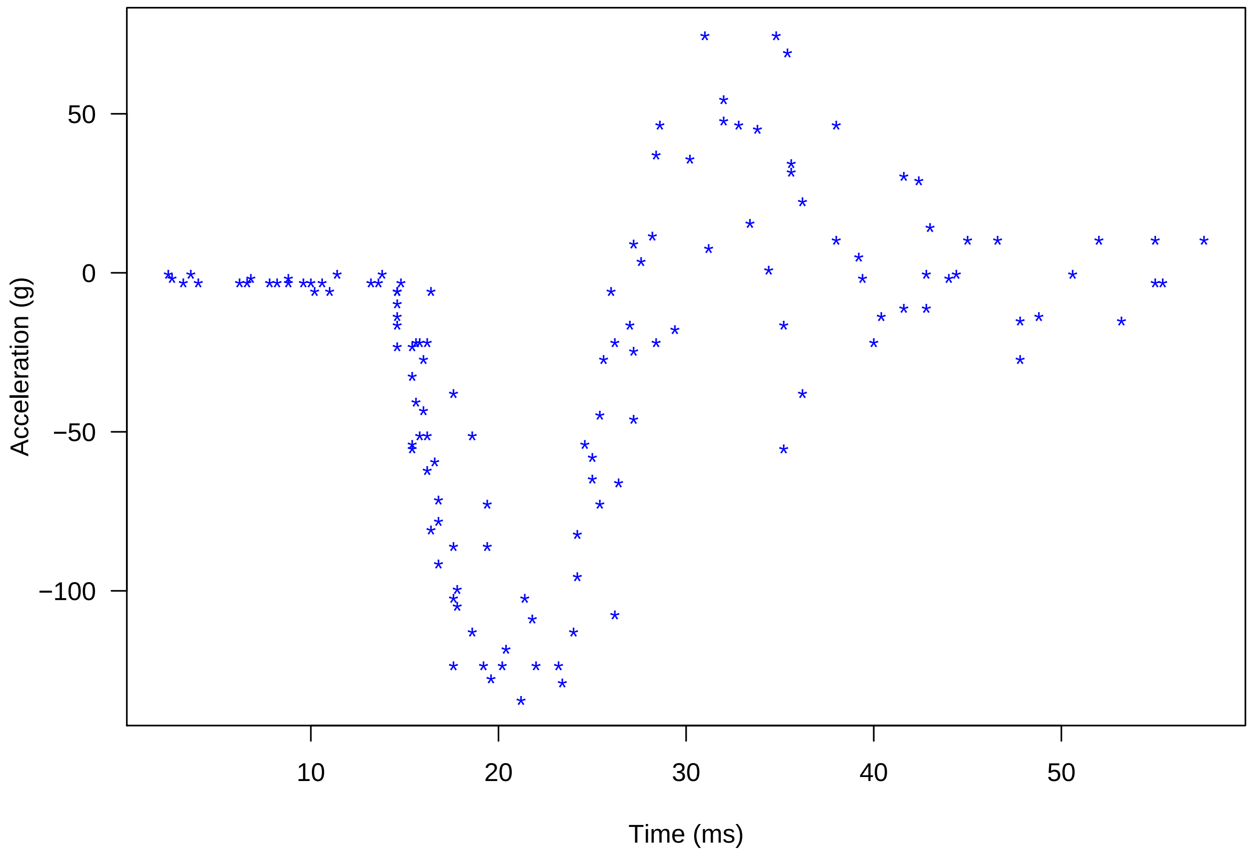
```r
library(scales)
plot(population ~ year, USPop,
     xlim = c(1790, 2100),
     ylim = c(0, 500),
     las = 1, pch = "*", col = "blue",
     xlab = "Census year", ylab = "Population (millions)", cex = 1.6)
with(USPop, lines(seq(1790, 2100, by = 10),
                  predict(pop.ss, data.frame(year = seq(1790, 2100, by = 10))), lwd = 1, col = alpha("b
points(2010, 308, pch = "*", cex = 2,
       col = "red")
abline(h = coef(pop.ss)[1], lty = 3,
       col = "gray", lwd = 0.95)
with(USPop, lines(seq(1790, 2100, by = 10),
                  predict(pop.qm, data.frame(year = seq(1790, 2100, by = 10))), lwd = 1, lty = 2, col =
legend("bottomright",
       legend = c("NLR", "PolyR"),
       lty = c(1, 2),
       bty = "n")
```



Non-parametric Regression

Data

```
library(MASS)
data("mcycle")
attach(mcycle)
plot(times, accel, pch = "*", cex = 1,
     col = "blue", las = 1,
     xlab = "Time (ms)", ylab = "Acceleration (g)")
```



Regression spline

```
library(splines)
# Select the knots
knots <- quantile(times, p = seq(0.1, 0.9, 0.1))
RSFit <- lm (accel ~ bs(times, knots = knots), data = mcycle)
# Make predictions
xg <- seq(0, 58, 0.1)
RSg <- predict(RSFit, data.frame(times = xg))
```

```
## Warning in bs(times, degree = 3L, knots = c(`10%` = 10.04, `20%` = 14.68, : some
## 'x' values beyond boundary knots may cause ill-conditioned bases
```

GAM

```
library(mgcv)
```

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

```
## This is mgcv 1.8-28. For overview type 'help("mgcv-package")'.
```

```r
GAMFit <- gam(accel ~ s(times), data = mcycle)
GAMg <- predict(GAMFit, data.frame(times = xg))
```

Smoothing Spline

```r
library(fields)
```

```
## Loading required package: spam

## Loading required package: dotCall64

## Loading required package: grid

## Spam version 2.4-0 (2019-11-01) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## Loading required package: maps

## See https://github.com/NCAR/Fields for
##  an extensive vignette, other supplements and source code
```

```r
SpFit <- sreg(times, accel)
Spg <- predict(SpFit, xg)
```

Local Regression

```r
library(locfit)
```

```
## locfit 1.5-9.1    2013-03-22
```

```r
locFit <- locfit(accel ~ times,
                 data = mcycle)
locg <- predict(locFit, xg)
```

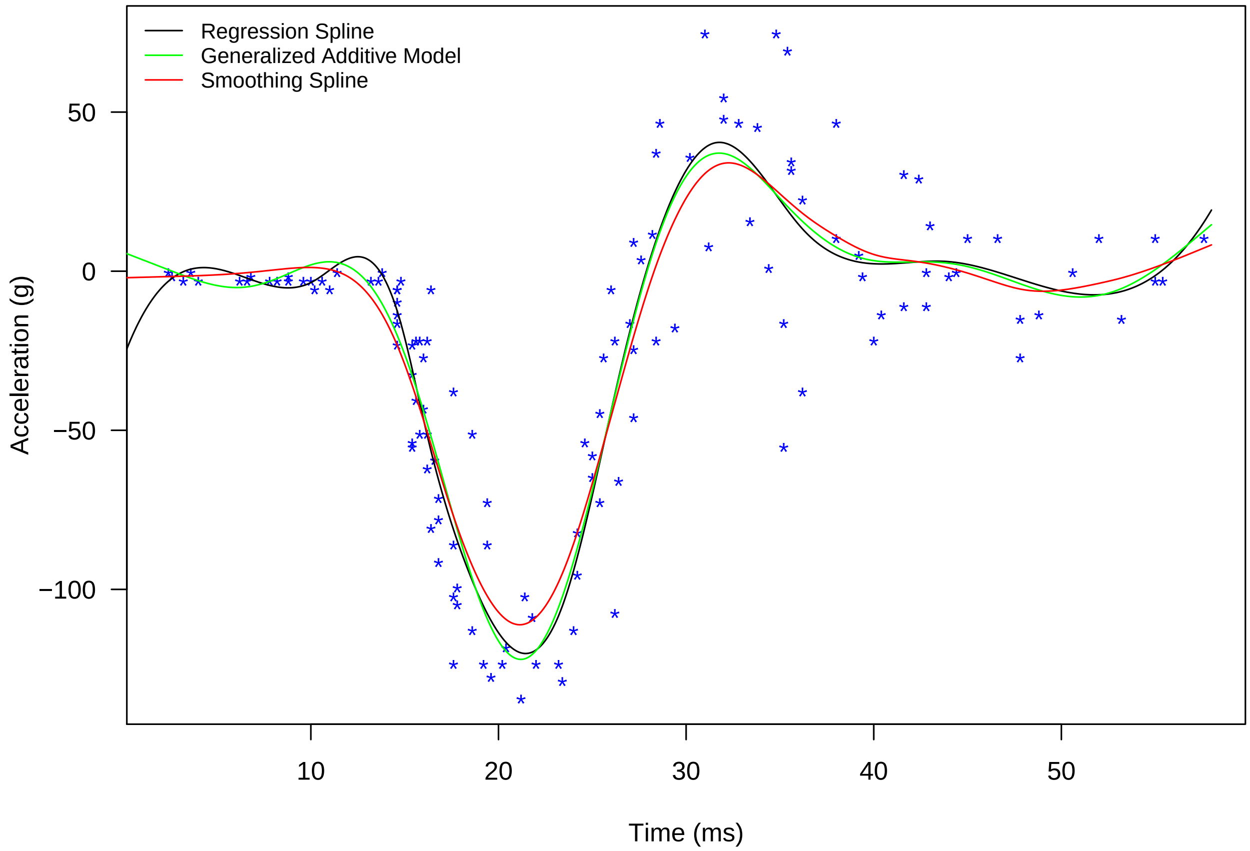
```r
xg <- seq(0, 58, 0.1)
library(MASS)
summary(mcycle)
```

```
##      times           accel
##  Min.   : 2.40   Min.   :-134.00
##  1st Qu.:15.60   1st Qu.: -54.90
##  Median :23.40   Median : -13.30
##  Mean   :25.18   Mean   : -25.55
##  3rd Qu.:34.80   3rd Qu.:   0.00
##  Max.   :57.60   Max.   :  75.00
```

```r
attach(mcycle)
```

```
## The following objects are masked from mcycle (pos = 12):
##
##     accel, times
```

```
plot(times, accel, pch = "*", cex = 1,
     col = "blue", las = 1,
     xlab = "Time (ms)", ylab = "Acceleration (g)")
lines(xg, RSg)
lines(xg, GAMg, col = "green")
lines(xg, Spg, col = "red")
legend("topleft", legend = c("Regression Spline", "Generalized Additive Model",
                             "Smoothing Spline"), lty = 1, bty = "n", cex = 0.8,
       col = c("black", "green", "red"))
```



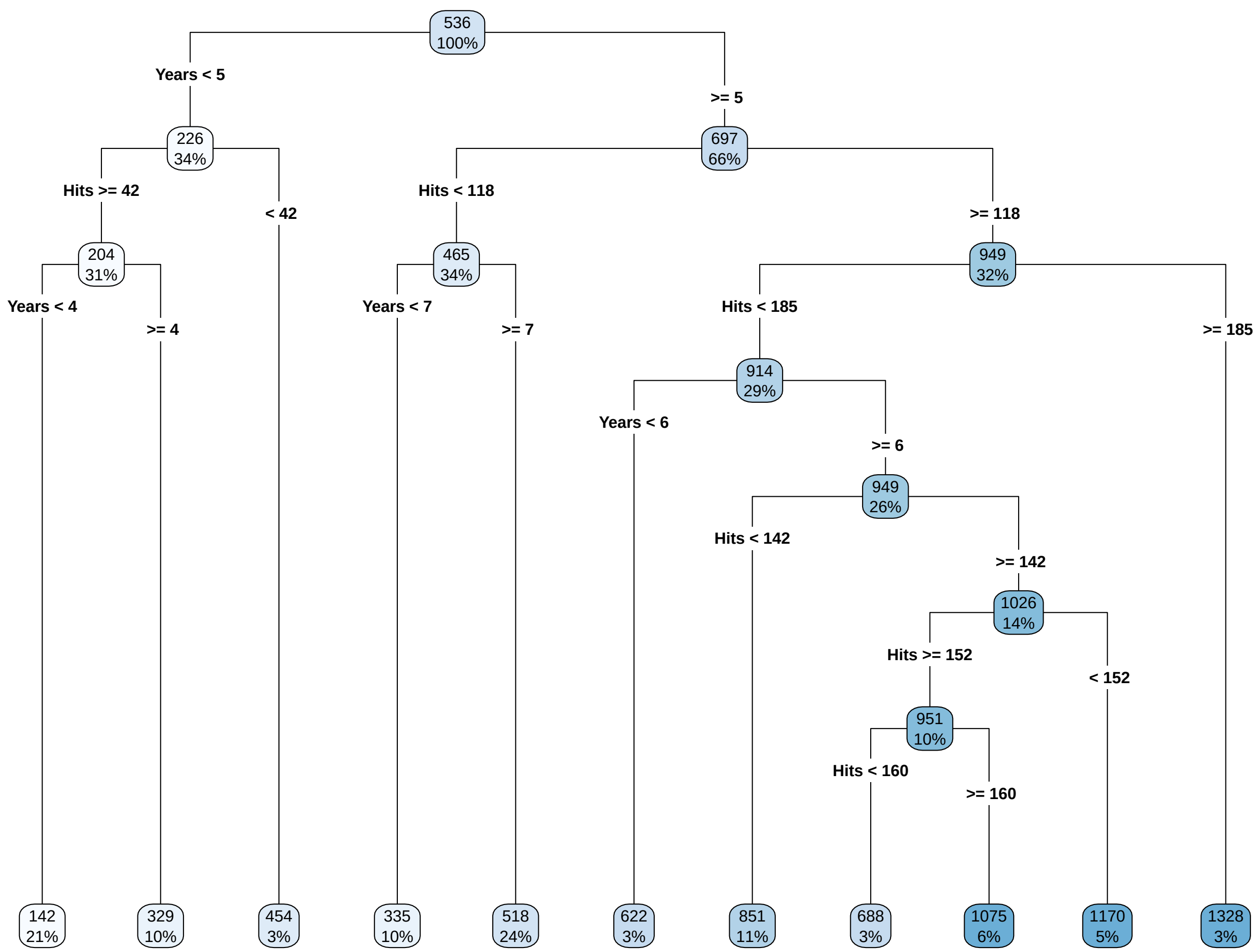
Regression Tree

```
library(rpart)
library(rpart.plot)
hitters <- read.csv('./Hitters.csv')
head(hitters)
```

```
##                     X AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun
## 1     -Andy Allanson   293   66     1   30  29    14     1    293    66      1
## 2        -Alan Ashby   315   81     7   24  38    39    14   3449   835     69
## 3       -Alvin Davis   479  130    18   66  72    76     3   1624   457     63
## 4      -Andre Dawson   496  141    20   65  78    37    11   5628  1575    225
## 5 -Andres Galarraga   321   87    10   39  42    30     2    396   101     12
```

```
## 6  -Alfredo Griffin   594  169     4   74  51    35    11   4408  1133     19
##   CRuns CRBI CWalks League Division PutOuts Assists Errors Salary NewLeague
## 1    30   29     14      A        E     446      33     20     NA         A
## 2   321  414    375      N        W     632      43     10  475.0         N
## 3   224  266    263      A        W     880      82     14  480.0         A
## 4   828  838    354      N        E     200      11      3  500.0         N
## 5    48   46     33      N        E     805      40      4   91.5         N
## 6   501  336    194      A        W     282     421     25  750.0         A
```

```r
reg.tree <- rpart(Salary ~ Years + Hits, data = hitters)
rpart.plot(reg.tree, type = 4)
```



Ridge regression

```r
library (car)
library (ridge)
```

```
## Warning: package 'ridge' was built under R version 3.6.2
```

```r
data(longley, package="datasets")
head (longley)
```

```
##      GNP.deflator     GNP Unemployed Armed.Forces Population Year Employed
## 1947         83.0 234.289      235.6        159.0    107.608 1947   60.323
## 1948         88.5 259.426      232.5        145.6    108.632 1948   61.122
```

```
## 1949         88.2 258.054      368.2        161.6    109.773 1949   60.171
## 1950         89.5 284.599      335.1        165.0    110.929 1950   61.187
## 1951         96.2 328.975      209.9        309.9    112.075 1951   63.221
## 1952         98.1 346.999      193.2        359.4    113.270 1952   63.639
```

```r
inputData <- data.frame (longley)
colnames(inputData)[1] <- "response"
XVars <- inputData[, -1]
round(cor(XVars), 2)
```

```
##               GNP Unemployed Armed.Forces Population Year Employed
## GNP          1.00       0.60         0.45       0.99 1.00     0.98
## Unemployed   0.60       1.00        -0.18       0.69 0.67     0.50
## Armed.Forces 0.45      -0.18         1.00       0.36 0.42     0.46
## Population   0.99       0.69         0.36       1.00 0.99     0.96
## Year         1.00       0.67         0.42       0.99 1.00     0.97
## Employed     0.98       0.50         0.46       0.96 0.97     1.00
```

```r
set.seed(800) # set seed to replicate results
trainingIndex <- sample(1:nrow(inputData), 0.8 * nrow(inputData)) # indices for 80% training data
trainingData <- inputData[trainingIndex,] # training data
testData <- inputData[-trainingIndex,] # test data
```

```r
lmMod <- lm(response ~ ., trainingData)  # the linear reg model
summary (lmMod) # get summary
```

```
##
## Call:
## lm(formula = response ~ ., data = trainingData)
##
## Residuals:
##    1949    1957    1952    1948    1959    1950    1962    1955    1954    1951
## -0.3409  0.8610  0.9575  1.1766 -0.3208 -0.9498 -0.3727 -2.0092  0.3286 -0.6466
##    1958    1960
##  1.1773  0.1389
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7944.87592 9446.02718   0.841    0.439
## GNP             0.30149    0.16006   1.884    0.118
## Unemployed      0.06564    0.05183   1.267    0.261
## Armed.Forces    0.02292    0.02597   0.882    0.418
## Population     -1.59239    1.11379  -1.430    0.212
## Year           -4.05306    4.95035  -0.819    0.450
## Employed        1.86480    2.64045   0.706    0.512
##
## Residual standard error: 1.433 on 5 degrees of freedom
## Multiple R-squared:  0.9913, Adjusted R-squared:  0.9809
## F-statistic: 95.23 on 6 and 5 DF,  p-value: 5.451e-05
```

```r
vif(lmMod) # get VIF
```

```
##          GNP   Unemployed Armed.Forces   Population         Year     Employed
##   1323.56015    102.48146     17.34559    310.12964   2826.31744    465.89465
```

```r
predicted <- predict(lmMod, testData)  # predict on test data
compare <- cbind (actual=testData$response, predicted)  # combine actual and predicted
```

```r
mean((compare[,1] -compare[,2])^2)
```

```
## [1] 1.48457
```

```r
linRidgeMod <- linearRidge(response ~ ., data = trainingData)
summary(linRidgeMod)
```

```
##
## Call:
## linearRidge(formula = response ~ ., data = trainingData)
##
##
## Coefficients:
##               Estimate Scaled estimate Std. Error (scaled) t value (scaled)
## (Intercept)  -1.021e+03              NA                  NA               NA
## GNP           3.096e-02       1.009e+01           2.208e+00            4.567
## Unemployed    1.031e-02       2.885e+00           2.091e+00            1.380
## Armed.Forces  1.231e-02       2.829e+00           1.819e+00            1.556
## Population    6.647e-02       1.506e+00           4.032e+00            0.374
## Year          5.279e-01       8.125e+00           1.487e+00            5.465
## Employed      9.916e-01       1.162e+01           3.793e+00            3.063
##              Pr(>|t|)
## (Intercept)        NA
## GNP          4.94e-06 ***
## Unemployed    0.16766
## Armed.Forces  0.11977
## Population    0.70869
## Year         4.63e-08 ***
## Employed      0.00219 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Ridge parameter: 0.01755985, chosen automatically, computed using 2 PCs
##
## Degrees of freedom: model 3.382 , variance 3.012 , residual 3.751
```

```r
predicted <- predict(linRidgeMod, testData)  # predict on test data
compare <- cbind (actual=testData$response, predicted)  # combine
mean((compare[,1] -compare[,2])^2)
```

```
## [1] 2.562397
```