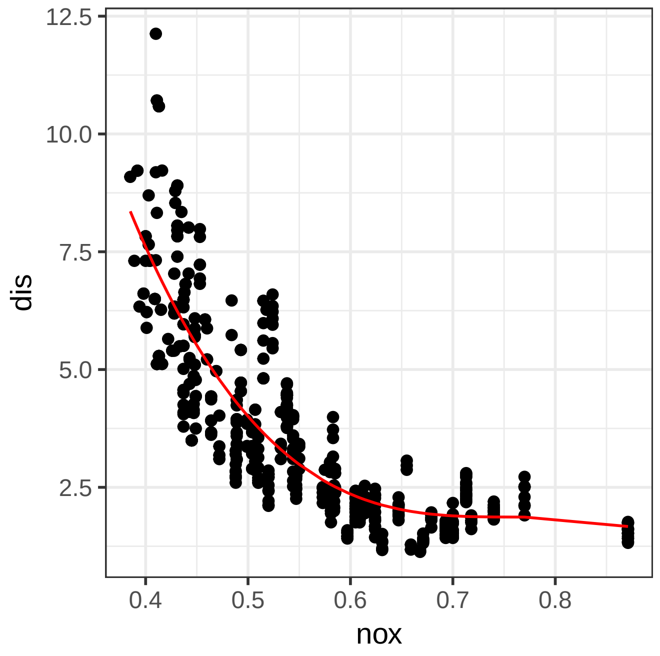
Assigment 4

1. For the Boston data available in package MASS we wish to relate `dis` (weighted mean of distances to five Boston employment centres) to `nox` (nitrogen oxides concentration in parts per 10 million).

(a) Fit a cubic polynomial to the data. Plot the data and the fit. Comment on the fit. Calculate the MSE.

```
library(tidyverse)
library(MASS)

f1 <- lm(dis ~ poly(nox,3), data = Boston)

ggplot(data=Boston, aes(x=nox, y=dis)) +
  geom_point() +
  geom_line(aes(y = fitted(f1)), color ="red") +
  theme_bw()
```



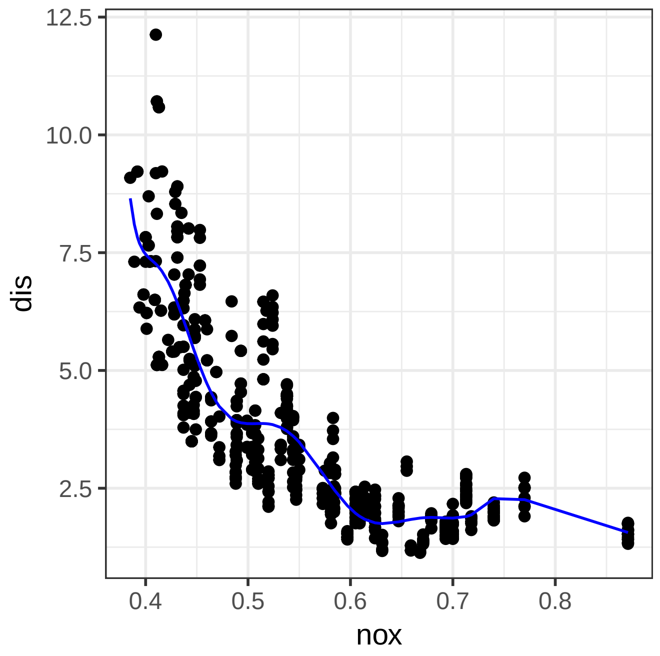
```
mean(residuals(f1)^2)
```

```
## [1] 1.094805
```

A reasonable fit. Residual plots show increasing variance. Very mild curvature.

(b) Repeat (a), this time using a 10th degree polynomial. Compare the fits and the MSE. Use anova to compare the two fits and comment on your findings.

```
f2 <- lm(dis ~ poly(nox, 10), data = Boston)

ggplot(data=Boston, aes(x=nox, y=dis)) +
  geom_point() +
  geom_line(aes(y = fitted(f2)), color = "blue") +
  theme_bw()
```



```r
mean(residuals(f2)^2)
```

```
## [1] 1.011485
```

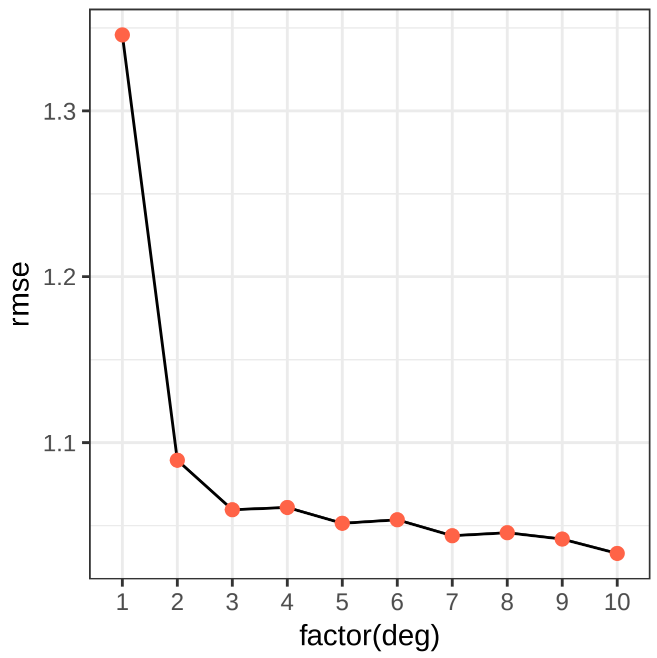
Some of the terms in f2 are significant (not all = 0). From the graph it overfits the data, but picks up the increase in dis with increasing nox past 0.65.

(c) Describe how you might use cross-validation to select the optimal degree (say between 1 and 10).

Split the data into 5 groups of approximately equal size. for each degree (j) between 1 and 10, - for each hold out sample, fit the model on the rest and calculate the average test error on the hold out sample, this gives mse1… mse5. the cv error for degree j is the (weighted) average of the mse1…. mse5. Pick the degree with the smallest cv error.

(d) Carry out the cross-validation procedure. What is the optimal degree?

```r
library(tidymodels)
set.seed(2019)

set.seed(2018)
cv_splits <- vfold_cv(
  data = Boston,
  v = 5
)


spec_lm <- linear_reg() %>% set_engine("lm")

geo_form <- dis ~ poly(nox, 3)

fit_model <- function(split, spec) {
  fit(
    object = spec,
    formula = geo_form,
    data = analysis(split)
  )
}

compute_pred <- function(split, model) {
  # Extract the assessment set
  assess <- assessment(split)
  # Compute predictions (a df is returned)
  pred <- predict(model, new_data = assess)
  bind_cols(assess, pred)
}

compute_perf <- function(pred_df) {
  numeric_metrics <- metric_set(rmse, rsq)

  numeric_metrics(
    pred_df,
    truth = dis,
    estimate = .pred
  )
}


mse_cv <- function(degree){

  geo_form <- dis ~ poly(nox, degree)

  fit_model <- function(split, spec) {
    fit(
      object = spec,
      formula = geo_form,
      data = analysis(split)
    )
  }


  cv_splits <- cv_splits %>%
    mutate(models_lm = map(splits, fit_model, spec_lm),
           pred_lm = map2(splits, models_lm, compute_pred),
           perf = map(pred_lm, compute_perf))

  cv_splits %>%
    unnest(perf) %>%
    filter(.metric == 'rmse') %>%
    summarise(m = mean(.estimate)) %>%
    pull(m)

}

mses <- data.frame(rmse = 1:10 %>% map_dbl(mse_cv),
                   deg = 1:10)

mses %>%
  ggplot(aes(y = rmse, factor(deg), group = 1)) +
  geom_line() +
  geom_point(colour = "tomato", size = 2) +
  theme_bw()
```



```
which.min(mses$rmse)
```

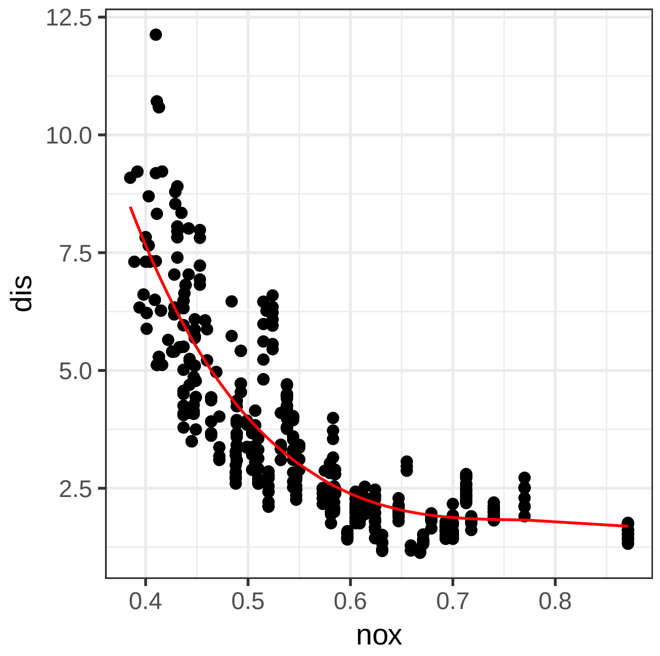
```
## [1] 10
```

(e) Use bs() to fit a regression spline with 4 degrees of freedom. What are the knots used? Plot the data and the fit. Comment on the fit. Calculate the MSE.

```
library(splines)
f3 <- lm(dis ~ bs(nox, df = 4), data = Boston)
attr(bs(Boston$nox, df = 4), "knots")
```

```
##   50%
## 0.538
```

```
Boston %>%
ggplot(aes(x = nox, y = dis)) +
  geom_point() +
  geom_line(aes(y = fitted(f3)), color = "red") +
  theme_bw()
```



```
sqrt(mean(residuals(f3)^2))
```

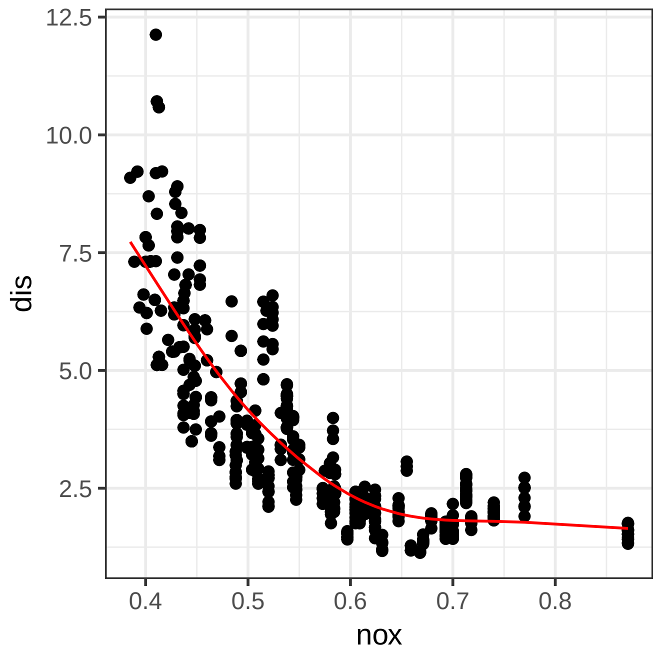
```
## [1] 1.046016
```

(f) Fit a curve using a smoothing spline with the automatically chosen amount of smoothing. Display the fit. Does the automatic λ give a good result?

```
f4 <- smooth.spline(Boston$nox, Boston$dis, cv = TRUE)
f4
```

```
## Call:
## smooth.spline(x = Boston$nox, y = Boston$dis, cv = TRUE)
##
## Smoothing Parameter  spar= 1.050661  lambda= 0.07031691 (16 iterations)
## Equivalent Degrees of Freedom (Df): 4.128915
## Penalized Criterion (RSS): 407.277
## PRESS(l.o.o. CV): 1.119355
```

```
Boston %>%
ggplot(aes(x = nox, y = dis)) +
  geom_point() +
  geom_line(aes(y = fitted(f4)), color = "red") +
  theme_bw()
```



```
sqrt(mean(residuals(f4)^2))
```

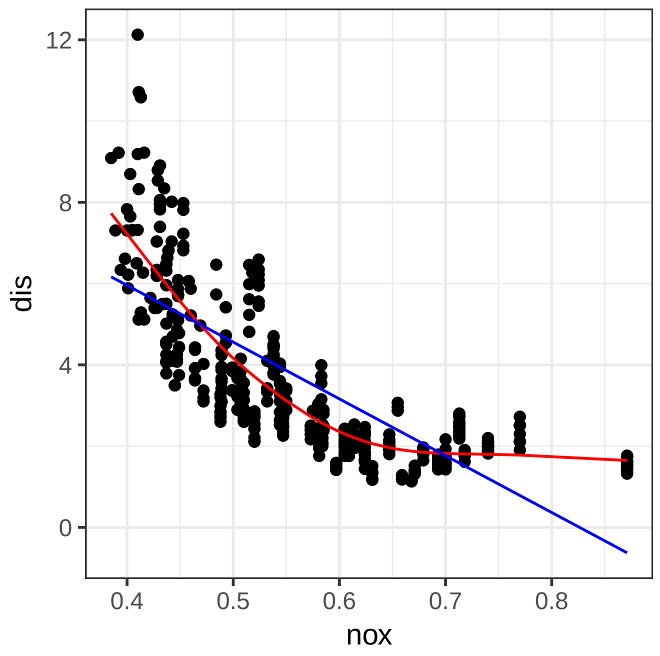
```
## [1] 1.050964
```

(g) Now use smoothing spline with a larger value of spar. Overlay both smoothing spline fits on the plot. Which looks better?

```
f5 <- smooth.spline(Boston$nox,Boston$dis, spar = 2)
f5
```

```
## Call:
## smooth.spline(x = Boston$nox, y = Boston$dis, spar = 2)
##
## Smoothing Parameter  spar= 2  lambda= 508171.7
## Equivalent Degrees of Freedom (Df): 1.999947
## Penalized Criterion (RSS): 762.6087
## GCV: 1.82113
```

```
Boston %>%
  ggplot(aes(x = nox, y = dis))+ geom_point() +
  geom_line(aes(y = fitted(f4)), color = "red")+
  geom_line(aes(y = fitted(f5)), color = "blue") +
  theme_bw()
```



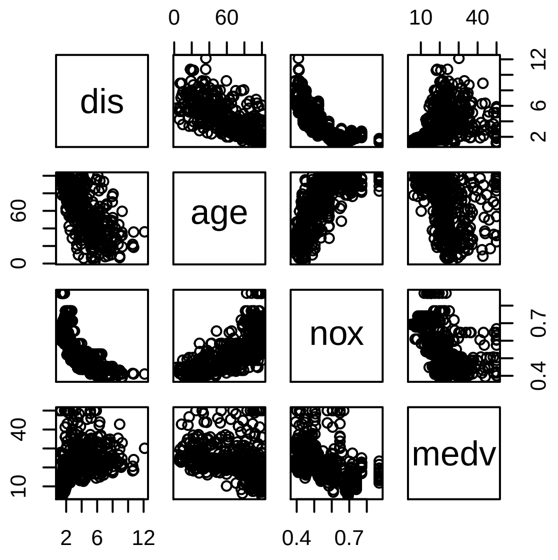
2. Using the Boston data, with `dis` as the response and predictors `medv`, `age` and `nox`.

(a) Split the data into training 60% and test 40%. Using the training data, fit a generalised additive model (GAM). Use ns with 4 degrees of freedom for each predictor.

```
library(splines)

Boston <-  Boston %>%
  mutate(part = ifelse(runif(nrow(.)) > 0.6, "test", "train"))

Boston %>%
  janitor::tabyl(part)
```

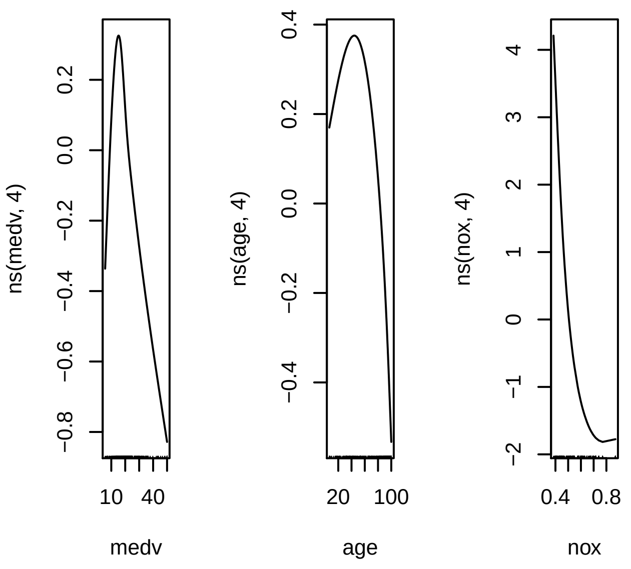
```
##   part   n   percent
##   test 189 0.3735178
##  train 317 0.6264822
```

```
train <- Boston %>% filter(part == "train") %>% dplyr::select(-part)
test <- Boston %>% filter(part == "test") %>% dplyr::select(-part)

gfit1 <-  lm(dis ~ ns(medv, 4)+ ns(age, 4) + ns(nox, 4), data = train)

pairs(Boston[, c(8, 7, 5, 14)])
```



(b) Use plot.gam to display the results. Does it appear if a linear term is appropriate for any of the predictors?

```r
par(mfrow = c(1, 3))
gam::plot.Gam(gfit1)
```



The linear model does not seem appropriate.

(c) Simplify the model fit in part (a). Refit the model. Use anova to compare the two fits and comment on your results.

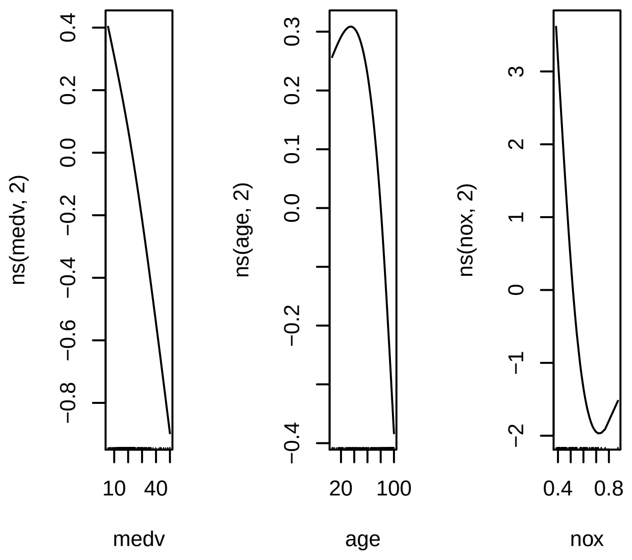
```r
attr(ns(train$age, 4), "knots")
```

```
##  25%  50%  75%
## 45.4 79.7 94.5
```

```r
attr(ns(train$age, 2), "knots")
```

```
##  50%
## 79.7
```

```r
gfit2 <-  lm(dis ~ ns(medv, 2) + ns(age, 2) + ns(nox, 2), data = train)

par(mfrow=c(1,3))
gam::plot.Gam(gfit2)
```



```r
anova(gfit1, gfit2)
```

```
## Analysis of Variance Table
##
## Model 1: dis ~ ns(medv, 4) + ns(age, 4) + ns(nox, 4)
## Model 2: dis ~ ns(medv, 2) + ns(age, 2) + ns(nox, 2)
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1    304 260.44
## 2    310 274.64 -6   -14.203 2.7631 0.01248 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

the model with fewer df is not appropriate.

4.

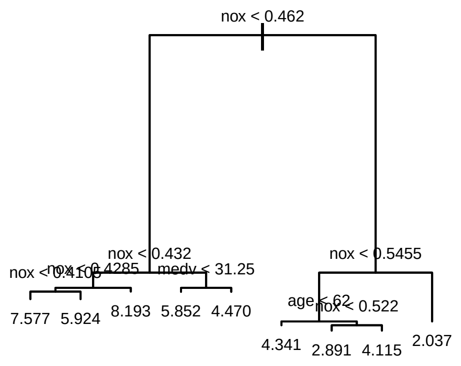
(a) For the training data in question 2, fit a tree model. Use dis as response, and predictors `medv`, `age` and `nox`. Draw the tree. Calculate the training and test MSE.

```r
library(tree)
tree <- tree(dis ~ medv + age + nox, data = train)
summary(tree)
```

```
##
## Regression tree:
## tree(formula = dis ~ medv + age + nox, data = train)
## Number of terminal nodes:  9
## Residual mean deviance:  0.5356 = 165 / 308
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.10400 -0.42980 -0.06625  0.00000  0.38060  3.29900
```

The fitted tree has 5 leaf nodes.

```r
plot(tree)
text(tree, cex = 0.5, pretty = 0)
```



```r
sqrt(mean(residuals(tree)^2))
```

```
## [1] 0.7213952
```

```r
pred <- predict(tree, test)
sqrt(mean((test$dis - pred)^2))
```

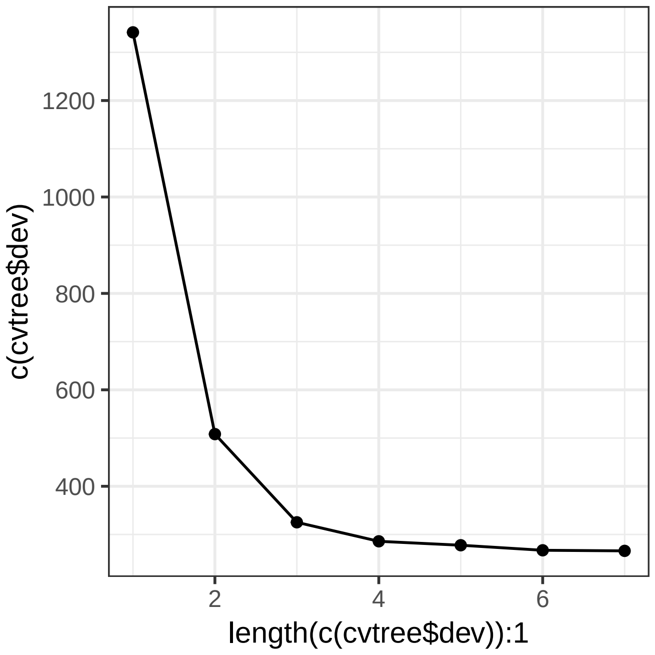
```
## [1] 1.199322
```

(b) Use `cv.tree` to select a pruned tree. If pruning is required, fit and draw the pruned tree. Calculate the training and test MSE. Compare the results to those in (a).

```r
cvtree <- cv.tree(tree)
cvtree
```

```
## $size
## [1] 9 8 6 4 3 2 1
##
## $dev
## [1]  266.0019  267.2256  277.7006  285.8440  325.2640  508.1814 1341.4652
##
## $k
## [1]      -Inf  14.31800  16.52661  20.31122  54.70596 175.73910 854.63983
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
ggplot() +
  geom_point(aes(x = length(c(cvtree$dev)):1, y = c(cvtree$dev))) +
  geom_line(aes(x = length(c(cvtree$dev)):1, y = c(cvtree$dev))) +
  theme_bw()
```



(c) Which fit is better, the (optionally pruned) tree or the GAM? Compare their performance on the test data.

```
pred <- predict(tree, test)
sqrt(mean((test$dis - pred)^2))
```

```
## [1] 1.199322
```

```
pred <- predict(gfit2, test)
sqrt(mean((test$dis - pred)^2))
```

```
## [1] 1.169403
```

The GAM has a slightly lower test MSE, but this is seed dependant.

5. For the data generated in question 6, Assignment 3:

```
set.seed(1)
x <- rnorm(100)
y <- 1 + .2*x+3*x^2+.6*x^3 + rnorm(100)
d <- data.frame(x = x, y = y)
```

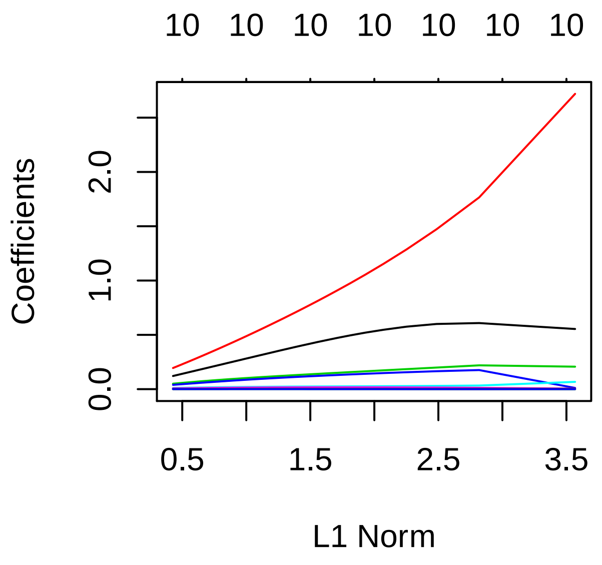
(a) Fit a regression model containing predictors $X, X^2, \ldots X^10$. Based on the output in `summary()` which terms are needed in the model?

```
lm(y ~ poly(x, 10, raw = TRUE), data = d) %>% summary()
```

```
##
## Call:
## lm(formula = y ~ poly(x, 10, raw = TRUE), data = d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9774 -0.5895 -0.1238  0.4923  2.1505
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 1.17283    0.19971   5.873 7.28e-08 ***
## poly(x, 10, raw = TRUE)1    0.71409    0.59009   1.210    0.229
## poly(x, 10, raw = TRUE)2    1.86854    1.29174   1.447    0.152
## poly(x, 10, raw = TRUE)3   -0.33114    1.68567  -0.196    0.845
## poly(x, 10, raw = TRUE)4    1.90383    2.14977   0.886    0.378
## poly(x, 10, raw = TRUE)5    0.55110    1.35654   0.406    0.686
## poly(x, 10, raw = TRUE)6   -1.26499    1.31956  -0.959    0.340
## poly(x, 10, raw = TRUE)7   -0.15569    0.39731  -0.392    0.696
## poly(x, 10, raw = TRUE)8    0.31987    0.32511   0.984    0.328
## poly(x, 10, raw = TRUE)9    0.01628    0.03817   0.426    0.671
## poly(x, 10, raw = TRUE)10  -0.02690    0.02749  -0.979    0.330
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9719 on 89 degrees of freedom
## Multiple R-squared:  0.951,  Adjusted R-squared:  0.9455
## F-statistic: 172.7 on 10 and 89 DF,  p-value: < 2.2e-16
```

(b) Fit a ridge regression model using the glmnet function over a grid of values for λ ranging from 0.001 to 50. Plot coefficients vs penalty using the default plot method. Use the inbuilt function `cv.glmnet` to choose the tuning parameter λ . How do the coefficients at the optimal value of λ compare to the linear regression ones in (a)?

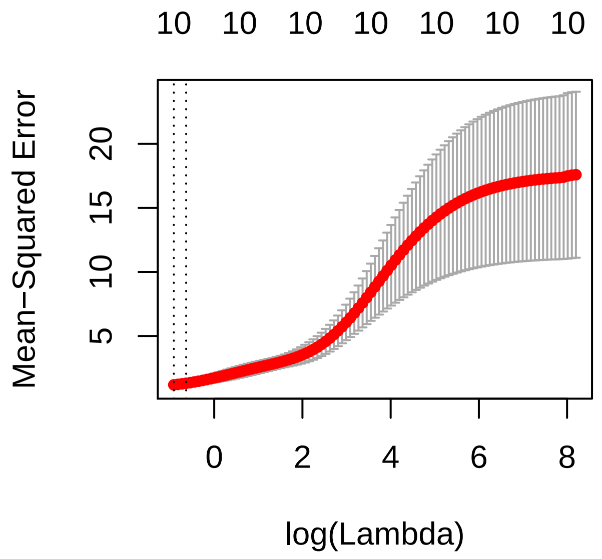
```
library(glmnet)
X <- model.matrix(y ~ poly(x, 10, raw = TRUE), data = d)
grid <- seq(0.001, 50, length = 100)
ridge.fit <- glmnet(X, y, alpha=0, lambda = grid)
plot(ridge.fit)
```



```r
cv.out <- cv.glmnet(X,y,alpha=0)
cv.out$lambda.min
```

```
## [1] 0.4000507
```

```r
plot(cv.out)
```

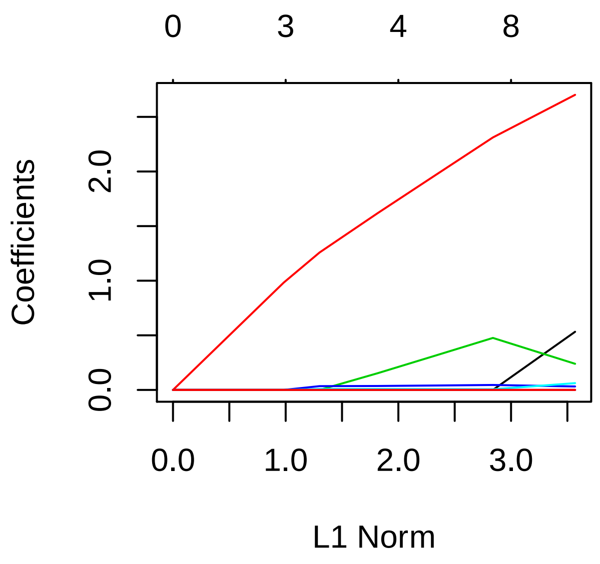


```r
ridge.fit <- glmnet(X, y, alpha = 0,
                    lambda = cv.out$lambda.min)

coef(ridge.fit)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                   s0
## (Intercept)             1.493577e+00
## (Intercept)             .
## poly(x, 10, raw = TRUE)1   6.119103e-01
## poly(x, 10, raw = TRUE)2   1.859173e+00
## poly(x, 10, raw = TRUE)3   2.221955e-01
## poly(x, 10, raw = TRUE)4   1.736758e-01
## poly(x, 10, raw = TRUE)5   3.293830e-02
## poly(x, 10, raw = TRUE)6   1.120493e-02
## poly(x, 10, raw = TRUE)7   3.820191e-03
## poly(x, 10, raw = TRUE)8  -7.804433e-05
## poly(x, 10, raw = TRUE)9   2.623561e-04
## poly(x, 10, raw = TRUE)10 -2.576105e-04
```

(c) Repeat (b) for lasso regression instead of ridge.

```r
lasso.fit <- glmnet(X, y, alpha=1, lambda = grid)
plot(lasso.fit)
```



```r
cv.out <- cv.glmnet(X, y, alpha=1)
cv.out$lambda.min
```

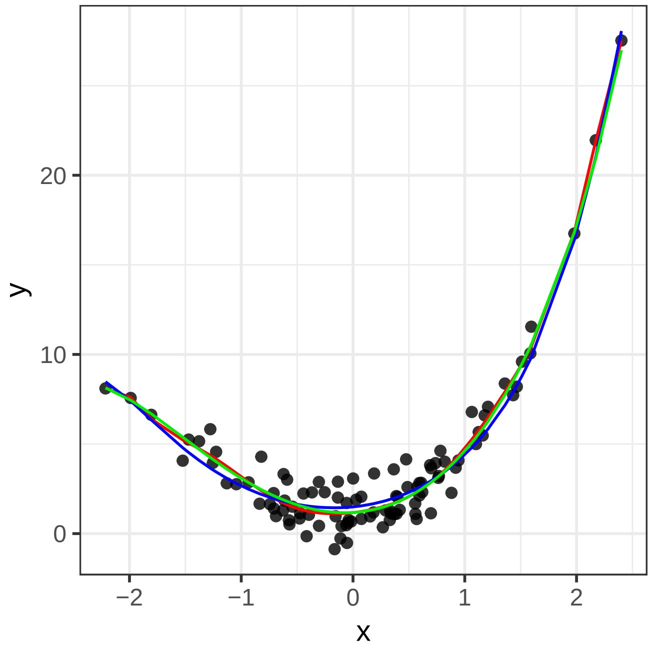
```
## [1] 0.0803795
```

```r
lasso.fit <- glmnet(X,y,alpha=1, lambda = cv.out$lambda.min)

coef(lasso.fit)
```

```
## 12 x 1 sparse Matrix of class "dgCMatrix"
##                                    s0
## (Intercept)               1.180097887
## (Intercept)               .
## poly(x, 10, raw = TRUE)1  0.380165945
## poly(x, 10, raw = TRUE)2  2.633676986
## poly(x, 10, raw = TRUE)3  0.348599739
## poly(x, 10, raw = TRUE)4  0.039359314
## poly(x, 10, raw = TRUE)5  0.032879765
## poly(x, 10, raw = TRUE)6  .
## poly(x, 10, raw = TRUE)7  0.002025108
## poly(x, 10, raw = TRUE)8  .
## poly(x, 10, raw = TRUE)9  .
## poly(x, 10, raw = TRUE)10 .
```

(d) Plot the data y vs x and superimpose the fitted models from linear regression, ridge and lasso with optimal values of lambda as chosen by cross-validation.

```r
Xord <- X[order(X[,2]),]

pred1 <- predict(lm(y~poly(x, 10, raw = TRUE), data = d),
              newdata = data.frame(x = sort(x)))

pred2 <- predict(ridge.fit, newx = Xord)

pred3 <- predict(lasso.fit, newx = Xord)

ggplot() +
  geom_point(aes(x, y), alpha = 0.8) +
  geom_line(aes(sort(x), pred1), color = "red") +
  geom_line(aes(sort(x), pred2), color = "blue") +
  geom_line(aes(sort(x), pred3), color = "green2") +
  theme_bw()
```



6. Titanic data from Assignment 3:

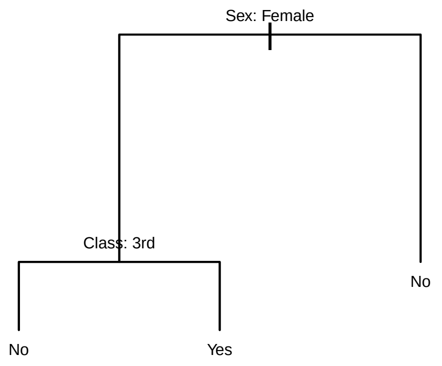
```r
ttrain <- read.csv("data/ttrain.csv", header = TRUE, row.names = 1)
ttest <- read.csv("data/ttest.csv", header = TRUE, row.names = 1)
```

(a) For the training data, fit a tree model using all three predictors. Draw the tree. Interpret the model. For the training and test data what proportion of survivors are missclassified? What proportion of those who died are missclassified? What proportion of the predicted survivors actually survived? What is the overall error rate for the training data?

```r
library(tree)
tree <- tree(Survived ~ ., data = ttrain)
summary(tree)
```

```
##
## Classification tree:
## tree(formula = Survived ~ ., data = ttrain)
## Variables actually used in tree construction:
## [1] "Sex"   "Class"
## Number of terminal nodes:  3
## Residual mean deviance:  0.9852 = 1732 / 1758
## Misclassification error rate: 0.2107 = 371 / 1761
```

```r
plot(tree)
text(tree, cex = 0.5, pretty = 0)
```



Fitted model: males have no chance of survival. For females, those in 3rd class have no chance of survival. The rest are predicted as survived. Age is not in the model.

```r
prob <- predict(tree, ttrain)[,2]
ttrain$pred <- factor(ifelse(prob < .5, "No", "Yes"))

ttrain %>%
  group_by(Survived, pred) %>%
  count() %>%
  group_by(Survived) %>%
  mutate(perc = scales::percent(n/sum(n)))
```

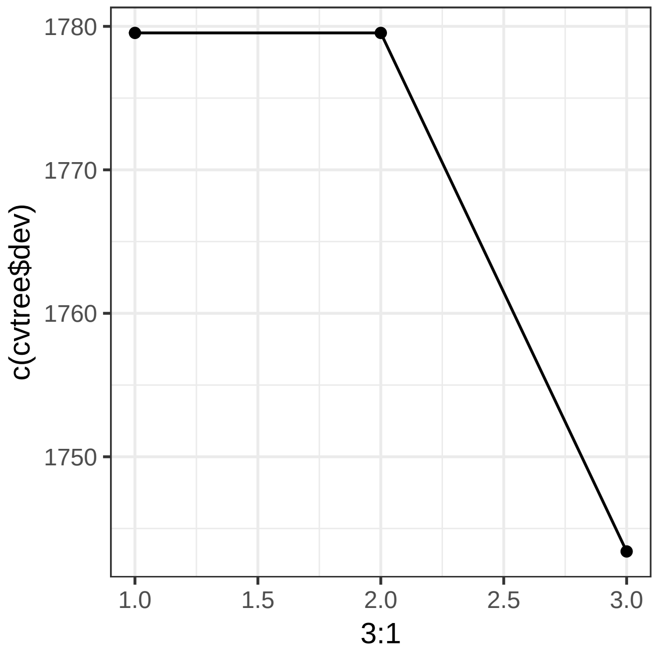
```
## # A tibble: 4 x 4
## # Groups:   Survived [2]
##   Survived pred      n perc
##   <fct>    <fct> <int> <chr>
## 1 No       No     1178 98.7%
## 2 No       Yes      15 1.3%
## 3 Yes      No      356 62.7%
## 4 Yes      Yes     212 37.3%
```

```
prob <- predict(tree, ttest)[,2]
ttest$pred <- factor(ifelse(prob < .5, "No", "Yes"))

ttest %>%
  group_by(Survived, pred) %>%
  count() %>%
  group_by(Survived) %>%
  mutate(perc = scales::percent(n/sum(n)))
```

```
## # A tibble: 4 x 4
## # Groups:   Survived [2]
##   Survived pred      n perc
##   <fct>    <fct> <int> <chr>
## 1 No       No      292 98.3%
## 2 No       Yes       5 1.7%
## 3 Yes      No      101 70.6%
## 4 Yes      Yes      42 29.4%
```

(b) Use `cv.tree` to select a pruned tree. If pruning is required, fit and draw the pruned tree.

```
cvtree <- cv.tree(tree)

ggplot() +
  geom_point(aes(x = 3:1, y = c(cvtree$dev))) +
  geom_line(aes(x = 3:1, y = c(cvtree$dev))) +
  theme_bw()
```

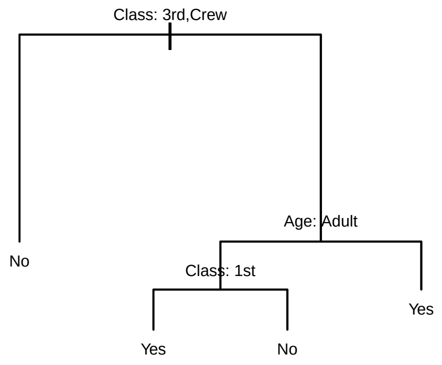


(c) Fit a tree model using only Age and Class as predictors. Draw the tree. Interpret the model. Compare the test set results to (a).

```r
tree <- tree(Survived ~ Age + Class, data = ttrain)
summary(tree)
```

```
##
## Classification tree:
## tree(formula = Survived ~ Age + Class, data = ttrain)
## Number of terminal nodes:  4
## Residual mean deviance:  1.148 = 2017 / 1757
## Misclassification error rate: 0.2731 = 481 / 1761
```

```r
plot(tree)
text(tree, cex = 0.5, pretty = 0)
```



```r
prob <- predict(tree, ttrain)[,2]
ttrain$pred <- factor(ifelse(prob < .5, "No", "Yes"))

ttrain %>%
  dplyr::select(1:3, pred) %>%
  group_by_all() %>%
  count() %>%
  arrange(pred, n)
```

```
## # A tibble: 14 x 5
## # Groups:   Class, Sex, Age, pred [14]
##    Class Sex    Age   pred      n
##    <fct> <fct>  <fct> <fct> <int>
##  1 Crew  Female Adult No       17
##  2 3rd   Female Child No       19
##  3 3rd   Male   Child No       37
##  4 2nd   Female Adult No       80
##  5 3rd   Female Adult No      130
##  6 2nd   Male   Adult No      132
##  7 3rd   Male   Adult No      381
##  8 Crew  Male   Adult No      686
##  9 1st   Female Child Yes       1
## 10 1st   Male   Child Yes       4
## 11 2nd   Male   Child Yes       7
## 12 2nd   Female Child Yes      13
## 13 1st   Female Adult Yes     116
## 14 1st   Male   Adult Yes     138
```

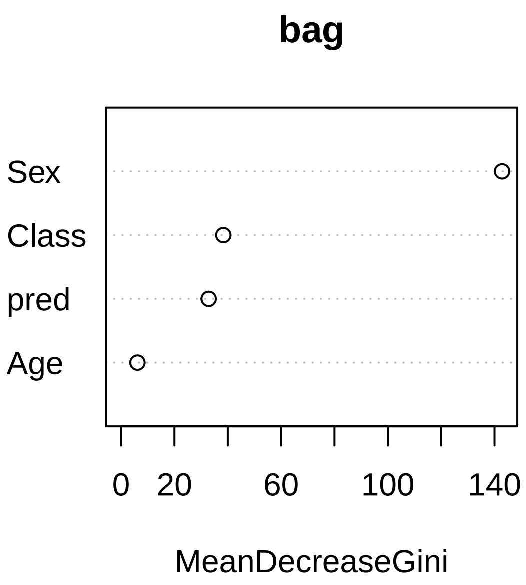
Fitted model: 3rd class and crew have no chance of survival. For those in 1st and 2nd class children are predicted to have survived. Adults in 1st class are predicted to have survived, but those in the 2nd did not.

```r
prob <- predict(tree, ttest)[,2]
ttest$pred <- factor(ifelse(prob < .5, "No", "Yes"))

ttest %>%
  group_by(Survived, pred) %>%
  count() %>%
  group_by(Survived) %>%
  mutate(perc = scales::percent(n/sum(n)))
```

```
## # A tibble: 4 x 4
## # Groups:   Survived [2]
##   Survived pred       n perc
##   <fct>    <fct> <int> <chr>
## 1 No       No      271 91.2%
## 2 No       Yes      26 8.8%
## 3 Yes      No       99 69.2%
## 4 Yes      Yes      44 30.8%
```

(d) Fit a random forest model (using `randomForest`) using all three predictors and compare the test set results to (a) and (c). Which variables are important?

```r
library(randomForest)
bag <- randomForest(Survived ~ ., data = ttrain)
varImpPlot(bag)
```



```r
ttest$pred <- predict(bag, newdata = ttest)

ttest %>%
  group_by(Survived, pred) %>%
  count() %>%
  group_by(Survived) %>%
  mutate(perc = scales::percent(n/sum(n)))
```

```
## # A tibble: 4 x 4
## # Groups:   Survived [2]
##   Survived pred      n perc
##   <fct>    <fct> <int> <chr>
## 1 No       No      292 98.3%
## 2 No       Yes       5 1.7%
## 3 Yes      No       96 67.1%
## 4 Yes      Yes      47 32.9%
```

7. Heart data: binary outcome AHD for 303 patients who presented with chest pain. An outcome value of Yes indicates the presence of heart disease, while No means no heart disease.

There are 13 predictors including Age, Sex, Chol (a cholesterol measurement), and other heart and lung function measurements.

Fit a support vector machine with a radial kernel to this data. Use cross validation to tune the λ and cost parameters (see function `tune()` in e1071 library). How does your result (test error) compare to the test error in the notes (obtained using trees and random forrests)?

```
set.seed(2019)
heart <- read.csv("data/heart.csv", row.names=1) %>% na.omit()

heart <-  heart %>%
  mutate(part = ifelse(runif(nrow(.)) > 0.66, "test", "train"))

heart %>%
  janitor::tabyl(part)
```

```
##   part   n   percent
##   test  92 0.3097643
##  train 205 0.6902357
```

```
train <- heart %>% filter(part == "train") %>% dplyr::select(-part)
test <- heart %>% filter(part == "test") %>% dplyr::select(-part)


library(e1071)

fit.svm <- svm(AHD ~ ., data = train, kernel = "radial")
summary(fit.svm)
```

```
##
## Call:
## svm(formula = AHD ~ ., data = train, kernel = "radial")
##
##
## Parameters:
```

```
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.05882353
##
## Number of Support Vectors:  117
##
##  ( 57 60 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
test$pred <- predict(fit.svm, newdata = test)

test %>%
  group_by(AHD, pred) %>%
  count()
```

```
## # A tibble: 4 x 3
## # Groups:   AHD, pred [4]
##   AHD   pred      n
##   <fct> <fct> <int>
## 1 No    No       42
## 2 No    Yes       3
## 3 Yes   No       12
## 4 Yes   Yes      35
```

```
tune.out <- tune(svm,AHD~., data = train, kernel = "radial",
                 ranges = list(cost = 10^seq(-1, 6, by = 1),
                               gamma = 10^seq(-6, 1, by = 1)))
tune.out
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost gamma
##  1e+05 1e-04
##
## - best performance: 0.1554762
```

```
fit.svm <- svm(AHD~., data = train, kernel = "radial",
               cost = 10000, gamma = 0.00001)
summary(fit.svm)
```

```
##
## Call:
```

```
## svm(formula = AHD ~ ., data = train, kernel = "radial", cost = 10000,
##     gamma = 1e-05)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10000
##       gamma:  1e-05
##
## Number of Support Vectors:  82
##
##  ( 40 42 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
test$pred <- predict(fit.svm, newdata = test)


test %>%
  group_by(AHD, pred) %>%
  count()
```

```
## # A tibble: 4 x 3
## # Groups:   AHD, pred [4]
##   AHD   pred      n
##   <fct> <fct> <int>
## 1 No    No       42
## 2 No    Yes       3
## 3 Yes   No       11
## 4 Yes   Yes      36
```

```
bag <- randomForest(AHD ~ ., data = train)
test$pred <- predict(bag, test)

test %>%
  group_by(AHD, pred) %>%
  count()
```

```
## # A tibble: 4 x 3
## # Groups:   AHD, pred [4]
##   AHD   pred      n
##   <fct> <fct> <int>
## 1 No    No       40
## 2 No    Yes       5
## 3 Yes   No       11
## 4 Yes   Yes      36
```