Some Notes from the Book: Statistical Learning from a Regression Perspective by Richard A. Berk

John L. Weatherwax*

September 12, 2001

Introduction

Here you'll find some notes that I wrote up as I worked through this excellent book. I've worked hard to make these notes as good as I can, but I have no illusions that they are perfect. If you feel that that there is a better way to accomplish or explain an exercise or derivation presented in these notes; or that one or more of the explanations is unclear, incomplete, or misleading, please tell me. If you find an error of any kind – technical, grammatical, typographical, whatever – please tell me that, too. I'll gladly add to the acknowledgments in later printings the name of the first person to bring each problem to my attention.

All comments (no matter how small) are much appreciated. In fact, if you find these notes useful I would appreciate a contribution in the form of a solution to a problem that is not yet worked in these notes. Sort of a "take a penny, leave a penny" type of approach. Remember: pay it forward.

*wax@alum.mit.edu

Statistical Learning as a Regression Problem

Problem Solutions

Problem 1 (the `airquality` dataset)

See the `R` script `chap_1_prob_1.R`.

Part (1): When we use the `pairs` command we get the plot shown in Figure 1. In reading a plot like this it is helpful to note that the y axis scale in each plot is determined by the variable denoted in the same horizontal row. The x axis variable is the variable in the same vertical row. Thus the scatter plot presented in the $(1, 2)$ location of the grid is a plot of `Ozone` considered as a function of `Solar.R`. The scatter plot presented in the $(3, 4)$ location is a plot of `Wind` as a function of `Temp`. Thus plots like this enable one to quickly view how two variable change in relationship to each other. The red curve is a non-parametric "smoothing" of the data that can given a quick understanding of how the two variables depend on each other. For example from the output of the `pairs` function we can see that from the $(1, 3)$ plot that `Ozone` decreases as `Wind` increases. From the $(1, 4)$ plot we see that `Ozone` increases as `Temp` increases. Comparing the "transpose" plots i.e. $(1, 3)$ and $(3, 1)$ can give an argument as to which variable should be the response and which variable should be the explanatory variable. For example in the $(3, 1)$ plot it looks like `Wind` is almost a linear function of `Ozone` while from $(1, 3)$ it does not look like `Ozone` is a linear function of `Wind`.

Part (3): Using `boxplot` to plot `Ozone` as a function of the categorical variable `Month` we get the plot show in Figure 2 (left). Plotting `Ozone` as a function of `Day` we get the plot show in Figure 2 (right). There is a clear pattern in that `Ozone` concentration seems to peak during the months of July and August. There is also a much larger range of possible values during these two months. There does not seem to be much of a pattern in the behaviour of `Ozone` as a function of `Day`. To use these variables in the scatterplots from Part (1) earlier we would have to specify the set of months or days to study in the scatterplots.

Part (5): When we use the `cloud` command we get the plot shown in Figure 3. We can see that `Ozone` increases as `Temp` increases and `Wind` decreases.

Part (6): When we use the `coplot` command we get the plot shown in Figure 4. In that plot it looks like the way that `Wind` is kept constant is to break it up into ordered bins and consider the samples that fall in each bin. From the given plot it looks like that when `Wind` is held constant the general trend is for `Ozone` to be an increasing as `Temp`.

Problem 2 (complexity of the fitting function)

See the `R` script `chap_1_prob_2.R`. When that script is run we get the result show in Figure 5.

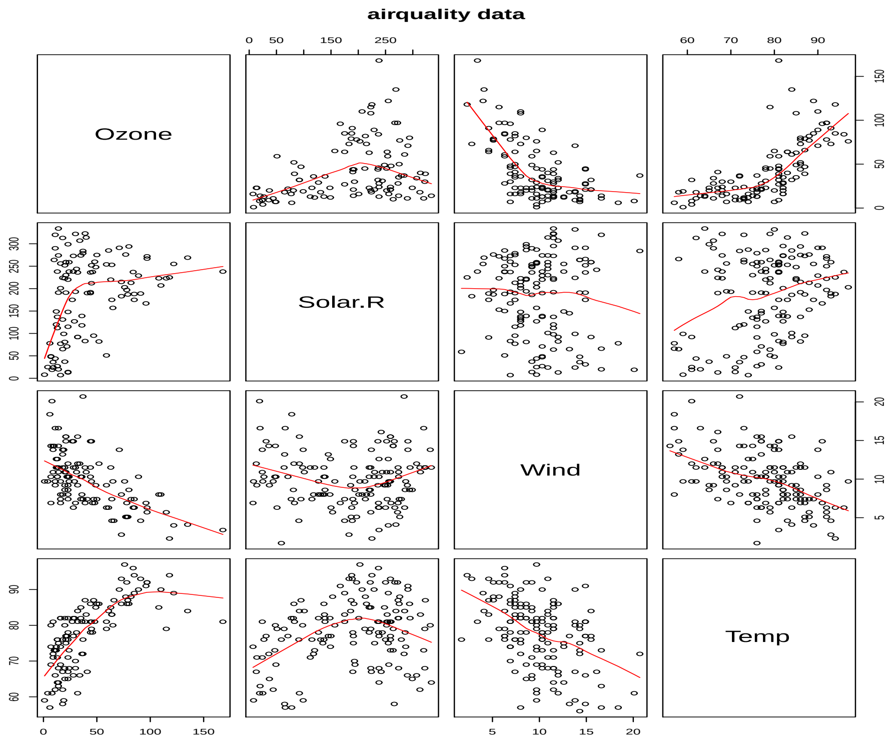

Figure 1: A "pairs" plot of the data in the `airquality` dataset.

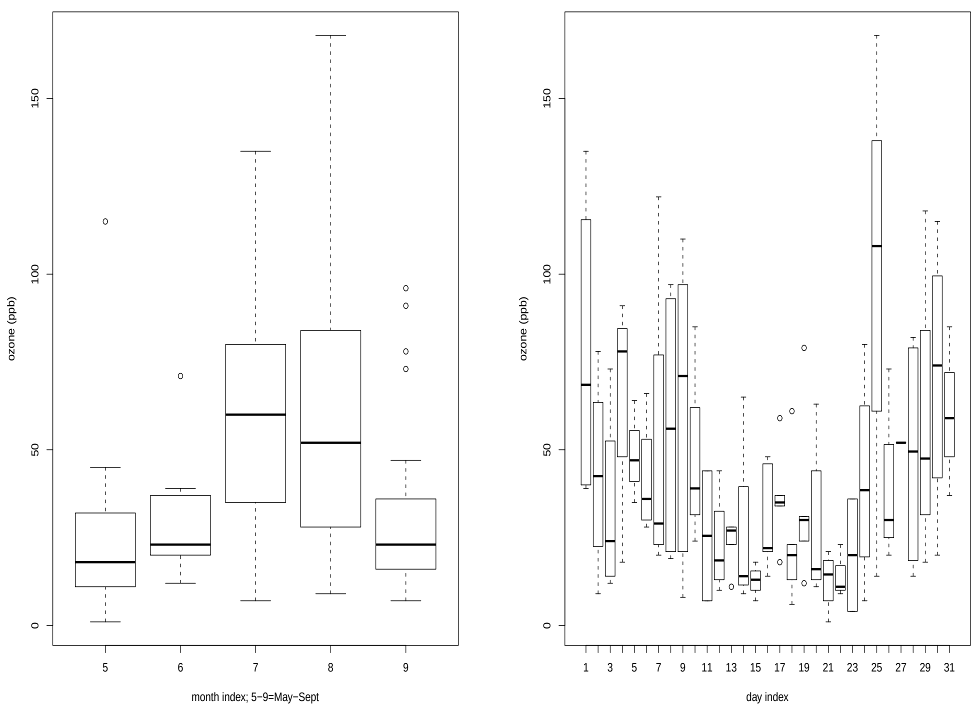

Figure 2: **Left:** Using the `boxplot` command to plot `Ozone` as a function of the month. **Right:** Using the `boxplot` command to plot `Ozone` as a function of the day in the month.

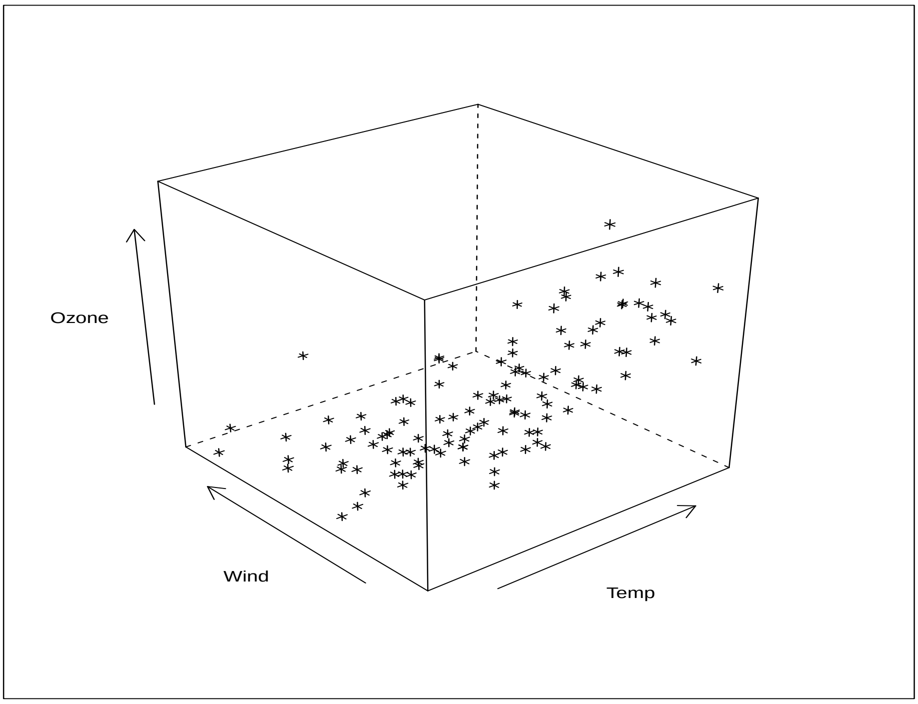

Figure 3: A “cloud” plot of `Ozone` as a function of `Wind` and `Temp`.

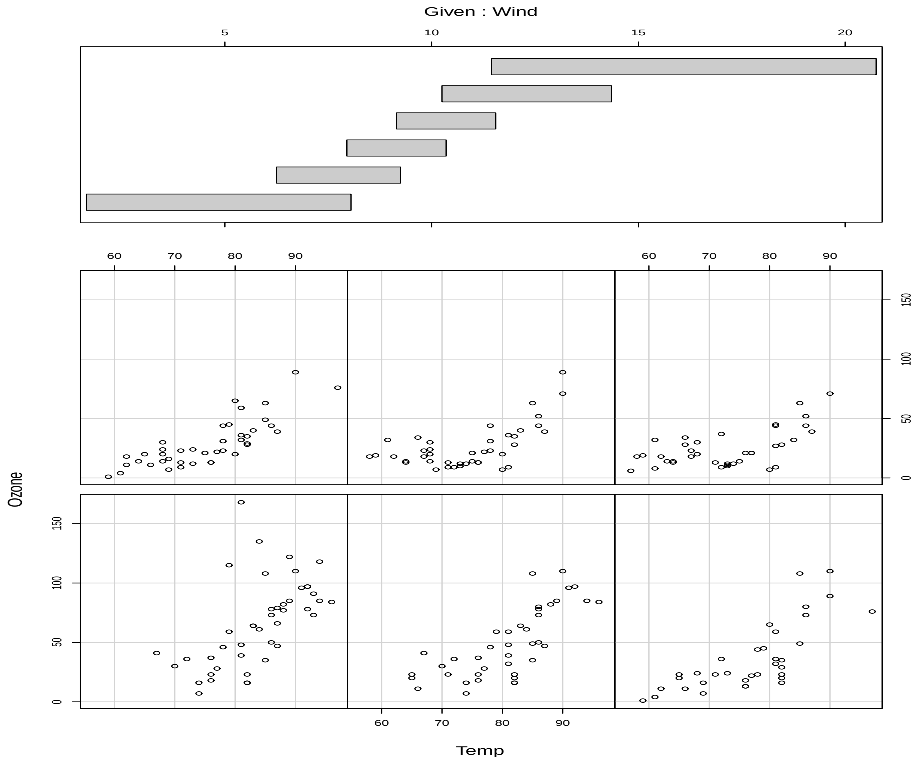

Figure 4: A “coplot” plot of `Ozone` as a function of `Temp` given `Wind`.

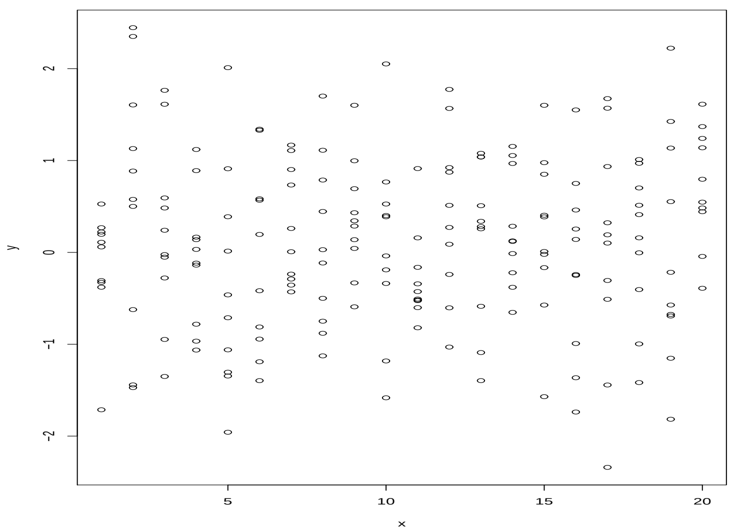

Figure 5: A plot of the point y vs. x generated from the data suggested in Problem 2.

The output from the `lm` command for this data gives

```
Call: lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-2.57442 -0.73968  0.02918  0.72450  2.53713

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.107822   0.153830   0.701    0.484
x           -0.003817   0.012841  -0.297    0.767

Residual standard error: 1.047 on 198 degrees of freedom
Multiple R-squared: 0.000446,   Adjusted R-squared: -0.004602
F-statistic: 0.08834 on 1 and 198 DF,  p-value: 0.7666
```

while the summary from the `glm` command gives

```
Call: glm(formula = y ~ x)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.57442  -0.73968   0.02918   0.72450   2.53713

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.107822   0.153830   0.701    0.484
x           -0.003817   0.012841  -0.297    0.767

(Dispersion parameter for gaussian family taken to be 1.096605)

    Null deviance: 217.22  on 199  degrees of freedom
Residual deviance: 217.13  on 198  degrees of freedom
AIC: 590.01

Number of Fisher Scoring iterations: 2
```

As claimed the fit is the same but the remaining outputs are different. The fact that the p-value (for the full model) is so large 0.7666 for the `lm` fit indicates that the linear model is not very good. The p-values for the individual coefficients (the intercept and slope in this case) are also relatively large. This again indicates that there is not much certainty in the coefficient estimate. In the use of `glm` the fact that the null and the residual deviance are so similar again indicates that the model fit is poor.

Problem 3 (overfitting the data)

Part (1): In this problem we generate 100 random vectors with 50 components each. As we have 99 coefficients to vary (to find fitted coefficients for) and there are only 50 vectors total we expect that there will be a great number of redundant (equally good) coefficient solutions. Thus we are in the case where we should be able to *exactly* fit the given data. If we look at the `lm` output we see that the first 50 coefficients are nonzero and there are NA's for most of the other diagnostic variables. This number of NA's indicate that there is perhaps a singularity in the fitting process.

For my version of `R` the function `stepAIC` gave the warning

```
attempting model selection on an essentially perfect fit is nonsense
```

and produces no output. This indicates that the model has been overfit. When the model is greatly overfit the model selection problem is not well defined since removing different predictors can result in the same change (perhaps no change) in fitting objective.

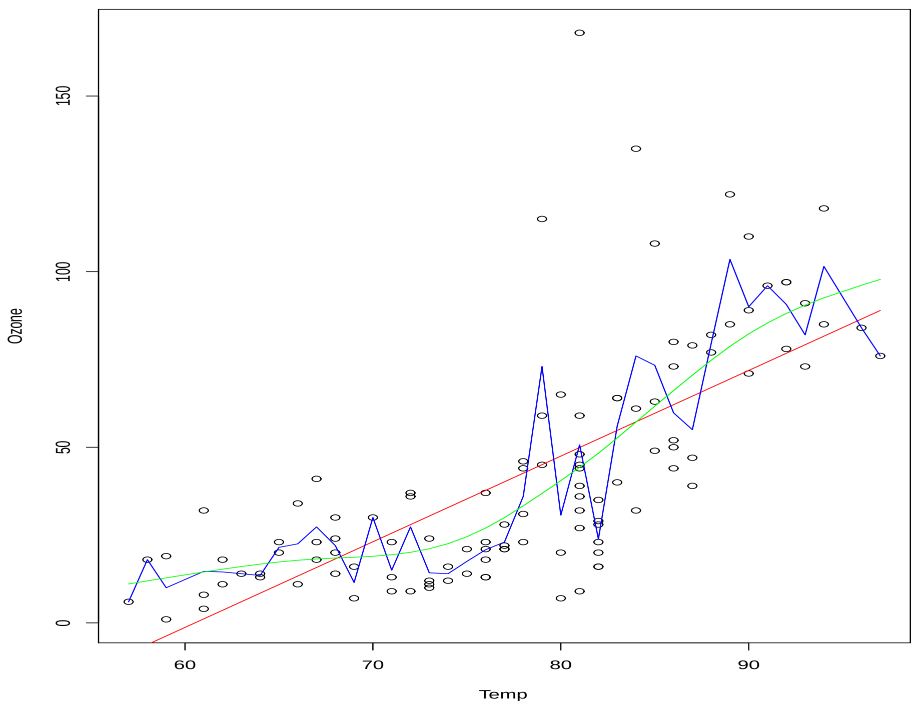

Figure 6: Three different models from the `gam` command for regressing `Ozone` as a function of `Temp` in the `airquality` dataset. The blue curve represents

Regression Splines and Regression Smoothers

Problem Solutions

Problem 1 (smoothers with one predictor)

See the `R` script `chap_2_prob_1.R`.

Part (1): When we run the above script we get the plot shown in Figure 6. The three residual deviances have values for the three models given by

```
[1] 62367.44 36979.84 52525.50
```

Selecting the residual deviance that is smallest would suggest using the second (the roughest) model. The three AIC values for the three models given by

```
[1] 1023.775 1039.759 1010.711
```

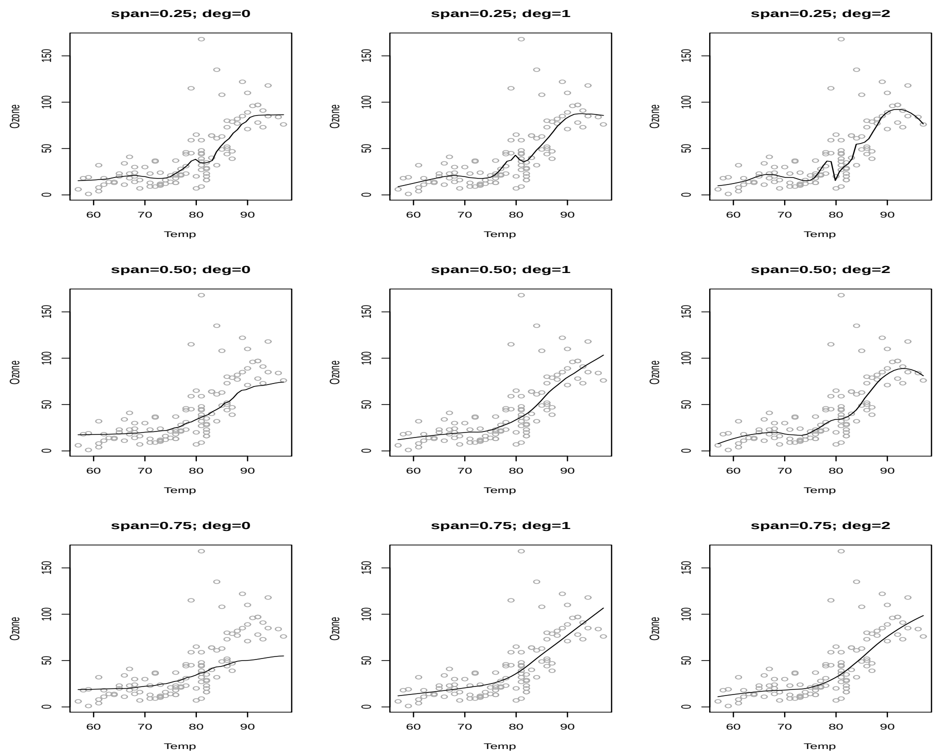

Figure 7: `Ozone` as a function of `Temp` and loess smoothing curves for varied `span` and `degree` values.

The smallest AIC in this case corresponds to the third model. As the AIC criterion penalizes models that have many terms it penalizes the second model due to its roughness. From plots of the `Ozone` as a function of `Temp` it looks like the third model seems to be the best.

Part (2): We can compute a scatter plot (with a loess smooth overlayed) by using the `R` command `scatter.smooth`. If needed we can just extract the fitted values from a loess smooth by using the `R` command `loess.smooth`. Since we are given several values to try of the parameters `span` and `degree`.

When we run the above script we get the plot shown in Figure 7.

Part (3): From the plots in Figure 7 it looks like to make the curve monotonically increasing taking `span` at 0.5 and `deg` at 1 seem to be good compromise from the options.

Problem 2 (smoothers with two predictors)

See the `R` script `chap_2_prob_2.R`.

Part (1): When we run the above script we get the plot shown in Figure 8.

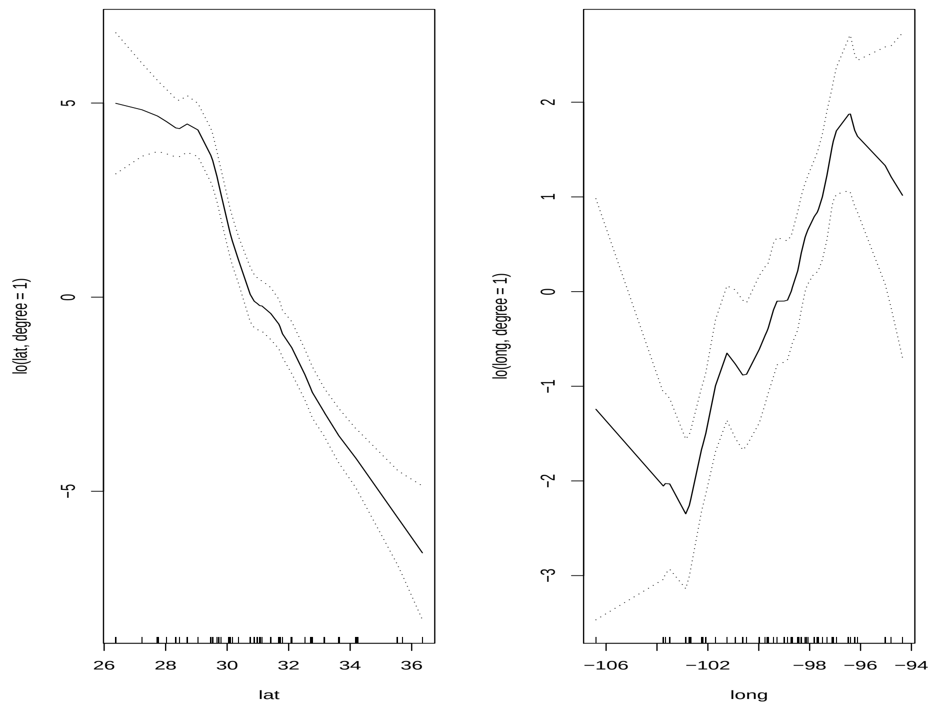

Figure 8: The two plots produced by the routine `plot.gam`.

Problem 3 (smoothers with more than two predictors)

See the `R` script `chap_2_prob_3.R`.

Problem 4 (smoothers with a binary response variable)

See the `R` script `chap_2_prob_4.R`.

Classification and Regression Trees (CART)

Problem Solutions

Problem 1 (CART vs. linear models)

See the R script chap_3_prob_1.R, where this problem is worked.

Part (1): When we use the lm command we get summary results given by

```
Call: lm(formula = y1 ~ x1 + x2)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.9741     0.1128   8.636 3.62e-16 ***
x1            2.2295     0.1167  19.101  < 2e-16 ***
x2            2.9377     0.1121  26.199  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1.952 on 297 degrees of freedom
Multiple R-squared: 0.783,      Adjusted R-squared: 0.7816
F-statistic: 535.9 on 2 and 297 DF,  p-value: < 2.2e-16
```

If we look at the estimated coefficients (and compare them to the known truth for this model) we see that the linear model is quite good at extracting the true underlying linear function from the given data. Next we use the rpart function to fit a CART tree to this data. When we use the text command to plot the resulting tree we get the plot shown in Figure 9 (left). From the given plot it is difficult to observe exactly how the data is generated under this modeling procedure. In this case it would seem that the linear model is better. The best way to tell the difference between models to to generate a second set of data (an out-of-sample) set and compare performance of the two algorithms on this new data set. If we do that in the R code and then compare the mean square error of the true response with the predicted response we get

```
[1] "IS: linear model MSE=   3.770304"
[1] "IS: regression tree MSE=   3.971424"
[1] "OOS: linear model MSE=   4.731974"
[1] "OOS: regression tree MSE=   7.096532"
```

Note that on the in sample data the two techniques are very similar in their performance. When tested out of sample however we see that the linear model outperforms the regression tree.

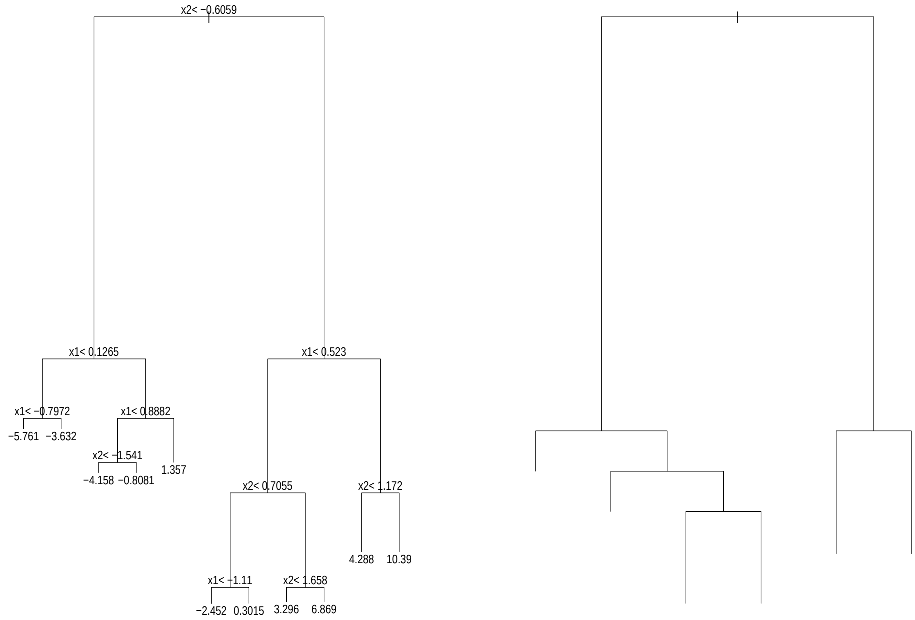

Figure 9: **Left:** The resulting CART tree for the real valued predictors given in Problem 1. This tree seems difficult to interpret. **Right:** The resulting CART tree for the boolean predictors given in Problem 1. This tree is relatively easy to interpret.

Part (2): In this case we expect the regression tree to outperform linear regression. The linear model has summary statistic given by

```
Call: lm(formula = y ~ x11 + x22)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.9879     0.1923   5.138 5.04e-07 ***
x11TRUE       2.2675     0.2262  10.025  < 2e-16 ***
x22TRUE       2.7165     0.2262  12.011  < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 1.955 on 297 degrees of freedom
Multiple R-squared: 0.4417,     Adjusted R-squared: 0.438
F-statistic: 117.5 on 2 and 297 DF,  p-value: < 2.2e-16
```

From this it looks like the linear regression has been fit as well as before. All of the coefficients are significant and the p-value of the entire fit is significant. What is a bit worry some is that the R-squared value is much lower for this fit. The `rpart` plot for this data looks is given in Figure 9 (right). Note how much simpler this tree structure is. Lets now compare the out of sample performance by generating new data as was done in Part (1). We find

```
[1] "IS: linear model MSE=   3.783262"
[1] "IS: regression tree MSE=   3.780213"
[1] "OOS: linear model MSE=   8.810891"
[1] "OOS: regression tree MSE=   8.857592"
```

Surprisingly the predictions made by these two techniques are not very different indicating that the linear model performs well in this situation also. Here we see that the CART technique performs as well as linear regression in that its MSE between the in and the out of sample result is not that different. In hindsight this is not that surprising since indicator functions (i.e. factors) are often used in linear regression to model various affects.

Part (3): In the above two examples in the case where the model was purely linear a linear model did quite well. CART did poorly in the case where the inputs are real valued due to it finding a biased estimate of $f(\cdot)$ (a function that is not strictly linear but in fact involves summing step functions). We therefore expect that CART regression trees to perform better than a linear model when the true function being modeled has its response function change at "steps" *or* the function we are truly modeling is *nonlinear*. If $f(\cdot)$ has either of these two properties then the CART result should outperform a linear model.

Problem 2 (classification with CART)

See the R script chap_3_prob_2.R, where this problem is worked.

Part (1): When we fit a linear model using `glm` and all predictors the summary of this gives

```
Call: glm(formula = pres.abs ~ ., family = binomial(), data = frogs)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.105e+02  1.388e+02   0.796  0.42587
altitude    -3.086e-02  4.076e-02  -0.757  0.44901
distance    -4.800e-04  2.055e-04  -2.336  0.01949 *
NoOfPools    2.986e-02  9.276e-03   3.219  0.00129 **
NoOfSites    4.364e-02  1.061e-01   0.411  0.68077
avrain      -1.140e-02  5.995e-02  -0.190  0.84920
meanmin      4.899e+00  1.564e+00   3.133  0.00173 **
meanmax     -5.660e+00  5.049e+00  -1.121  0.26224
---
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 279.99  on 211  degrees of freedom
Residual deviance: 198.74  on 204  degrees of freedom
```

From the above we see that the logistic regression is not fully sure of the values of many of the coefficients for the predictors. The variables `NoOfPools` and `meanmin` seem to be the best estimated coefficients. The variable `NoOfPools` is the number of potential breeding pools and has a positive coefficient thus as there are more pools there is a great chance of finding frogs. The variable `meanmin` is the mean minimum spring temperature and is also positive meaning that as the minimum temperature increases we are more likely to find frogs. Both of these findings seem to be reasonable results. The numbers above indicate that perhaps the linear model is not able to extract reliable estimate of the coefficients given the amount of data.

Part (2): When we run the `stepAIC` command we fit linear models by sequentially removing predictors one at a time looking for the model that minimizes the AIC. The output from this command shows what the AIC would be for models with various predictors removed. The predictor that is removed is the one that (when removed) has the model that has the smallest AIC. This process is repeated until removing a predictor results in the AIC of the model increasing.

```
> stepAIC( linear_model )
Start:  AIC=214.74
pres.abs ~ altitude + distance + NoOfPools + NoOfSites + avrain +
```

```
    meanmin + meanmax

            Df Deviance    AIC
- avrain     1   198.78 212.78
- NoOfSites  1   198.91 212.91
- altitude   1   199.30 213.30
- meanmax    1   199.97 213.97
<none>           198.74 214.74
- distance   1   206.39 220.39
- meanmin    1   209.60 223.60
- NoOfPools  1   210.84 224.84

... output omitted ...

            Df Deviance    AIC
<none>           199.63 209.63
- distance   1   209.73 217.73
- NoOfPools  1   211.43 219.43
- meanmax    1   216.10 224.10
- meanmin    1   226.94 234.94

Call:
glm(pres.abs ~ distance + NoOfPools + meanmin + meanmax, family = binomial() )

Coefficients:
(Intercept)     distance    NoOfPools      meanmin      meanmax
 14.0074032   -0.0005138    0.0285643    5.6230647   -2.3717579

Degrees of Freedom: 211 Total (i.e. Null);  207 Residual
Null Deviance:      280
Residual Deviance: 199.6        AIC: 209.6
```

The final result indicates what the "most important variables" are. The signs of the estimated coefficients should indicate if our estimation is reasonable. We find that the model states

- Both `NoOfPools` and `meanmin` have positive coefficients indicating that as the number of breeding pools and the average of the lowest temperature increase we expect the probability of finding frogs to *increase*.
- Both `distance` and `meanmax` have negative coefficients indicating that as the distance to the nearest extant (still in existence or surviving) pool and the average of the largest temperature increases we expect the probability of finding frogs to *decrease*.

The sign of the coefficients of the two variables `NoOfPools` and `meanmin` seems to be reasonable. The sign of the coefficients for `distance` also seems to be reasonable in that when we move away from existing frog populations it might become harder to find frogs. The sign of the coefficient of `meanmax` could be argued in that if the spring temperature is too hot then it might be less likely to find frogs.

Part (3): Rebuilding our linear model with the predictors found to be most important via `stepAIC` we next compute a confusion table. When we do this we get the table

```
     0   1
0  110  23
1   20  59
```

Thus we find $\frac{23}{110+23} = 0.1729$ for the false positive rate and $\frac{20}{20+59} = 0.2531$ for the false negative rate. We have $\frac{23+20}{110+23+20+59} = 0.20$ classified incorrectly. The classifier is more accurate in the true presence of frogs.

Part (4): We next use the `gam` with spline smoothing of the four predictors above to develop a model of the probability of `pres.abs`.

Part (5): When we use the `gam` computed above to classify our samples we get a confusion table given by

```
  yhat
     0   1
0  116  17
1   22  57
```

This classification seems to be more accurate for the true absence of frogs. These results are about as good as the results from using the `glm` code.

Part (6): When we fit all predictors in the `frogs` dataset using the `R` command `rpart` we get a plot like that shown in Figure 10. Notice that the predictors selected in defining the tree were selected when we used logistic regression in the `glm` framework. The direction of the splits in the tree correspond to the signs in the `glm` output. Thus the larger the variable `distance` and the lower the value of `meanmin` the less likely we are to find frogs.

Part (7): An in-sample confusion matrix of the CART algorithm gives

```
  yhat
     0   1
0  114  19
1   19  60
```

This is a slightly better result than that obtained from the `glm` logistic regression and slightly better than the `gam` model in the case of the true presence of frogs.

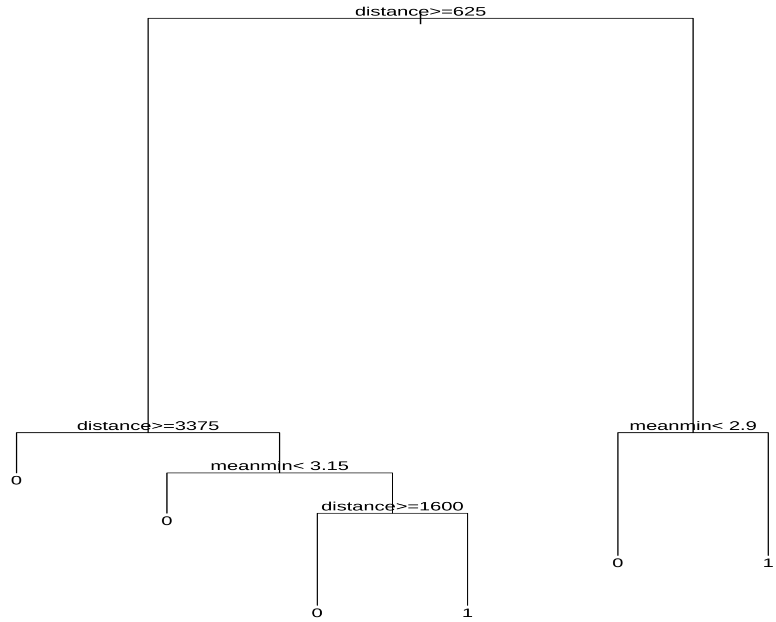

Figure 10: The resulting CART tree using all the predictors given in Problem 2.

Part (8): We next fit two models using `rpart` each with a different distribution of prior information. When we do this and then print the classification errors and the ratio of the false negative to false positives we get

```
[1] "Default CART classificaion error   0.179245; FN/FP =   1.000000"
[1] "CART with 50-50 split classificaion error   0.207547; FN/FP =   0.571429"
[1] "CART with 70-30 split classificaion error   0.160377; FN/FP =   0.619048"
```

Thus we see that by changing the priors we can change the false negative to false positive ratio.

Part (9): By using the formula in the book to scale the prior probabilities we can skew the ratio of false negatives to false positives. We find a confusion table that now looks like

```
   yhat
     0  1
  0 70 63
  1  0 79
```

Thus we see that the number of false negatives is now *zero* due to our choice of priors.

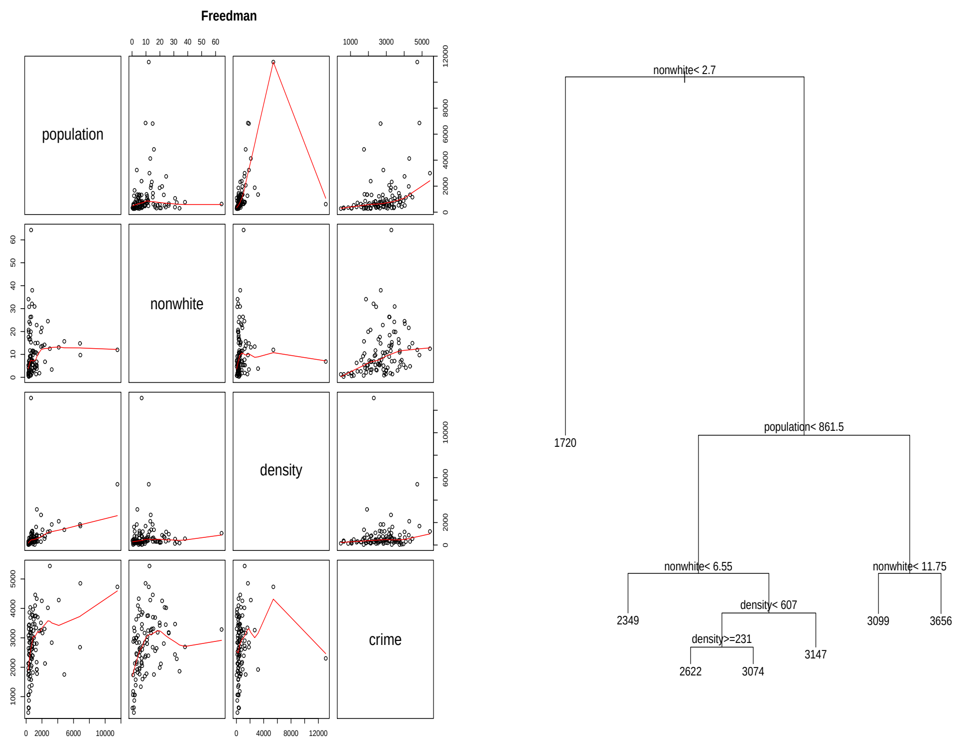

Figure 11: **Left**: Scatter plots (with loess smooths) of all the variables in the `Freedman` dataset. **Right**: The CART regression tree for the `Freedman` dataset.

Part (10): With repeated samples from the data we will get very different in-sample trees.

Part (11): When we enforce the different trees to have a node size of 50 the trees should be much more stable. Thus the CART algorithm in this case has much more bias but will have a lower variance. Repeated draws from the data set will produce much the same algorithm.

Problem 3 (quantitative prediction with CART)

See the `R` script `chap_3_prob_3.R`, where this problem is worked.

Part (1): We first use the command `pairs` command get get a general feel for how `crime` responds to the various other inputs. When we run the above `R` command the resulting `pairs` plot is shown in Figure 11 (left). From the given scatter plots (the bottom row of plots) it looks like `crime` increases with `population`, `crime` seems to increase (to a point) with `nonwhite`, and `crime` seems to increase (to a point) with `density`.

Part (2): Next we use `rpart` to fit a CART tree to the given data. The resulting graph is shown in Figure 11 (right). From the given CART tree it seems that the largest values of `crime` are located for large values of `nonwhite`, `density`, and `population`. These comments are qualitative and are based on looking at the given CART tree and finding the largest

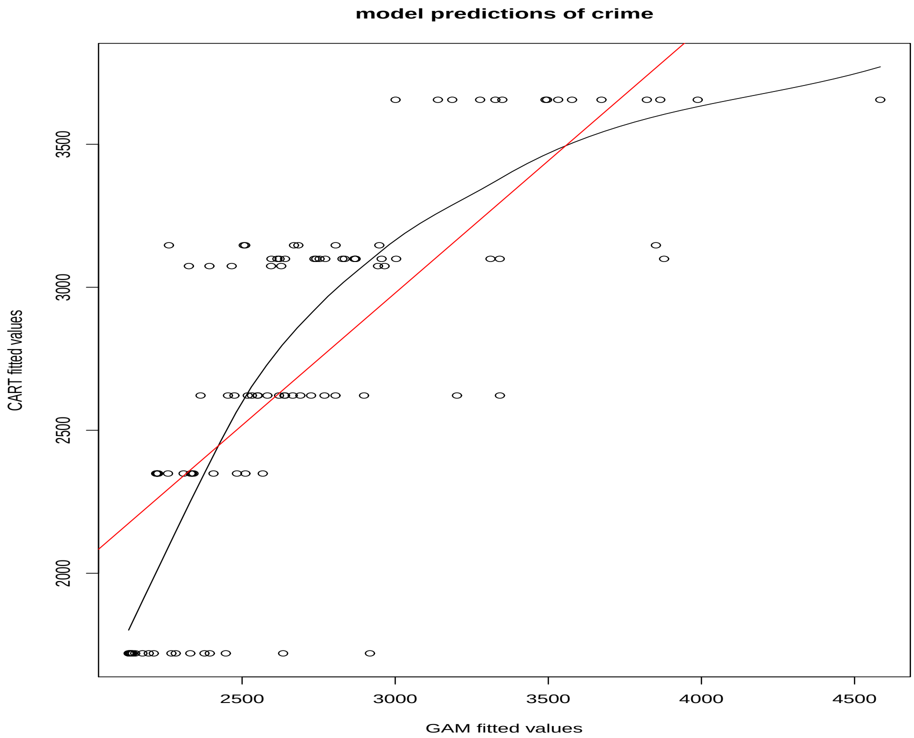

Figure 12: **Left**: Scatter plots (with loess smooths) of all the variables in the `Freedman` dataset.

predicted values for the leafs. The conclusions as how the inputs variable seem to affect `crime` are basically the same.

Part (3): In the Figure 12 we present the GAM fitted values as a function of the predicted CART tree values. If the two sets of points corresponded perfectly one would expect that this scatter plot would have many of the points on a line. Since the CART predictions are constant for all predictions that fall in the same leaf node we see that our scatter plot has

Part (4-6): In Figure 12 we plot the least squares linear fit of the two fitted values in red. This least squares line has an estimated slope of 0.92483 with an intercept of 205.4. When we compute the two correlations of the fitted values of each model with the true crime rate we find

```
> cor( m1$fitted.values, Freedman$crime )
[1] 0.5540136
> cor( tree_predictions, Freedman$crime )
[1] 0.6322788
```

Thus it looks like the CART tree is producing an output that is more correlated with the true output.

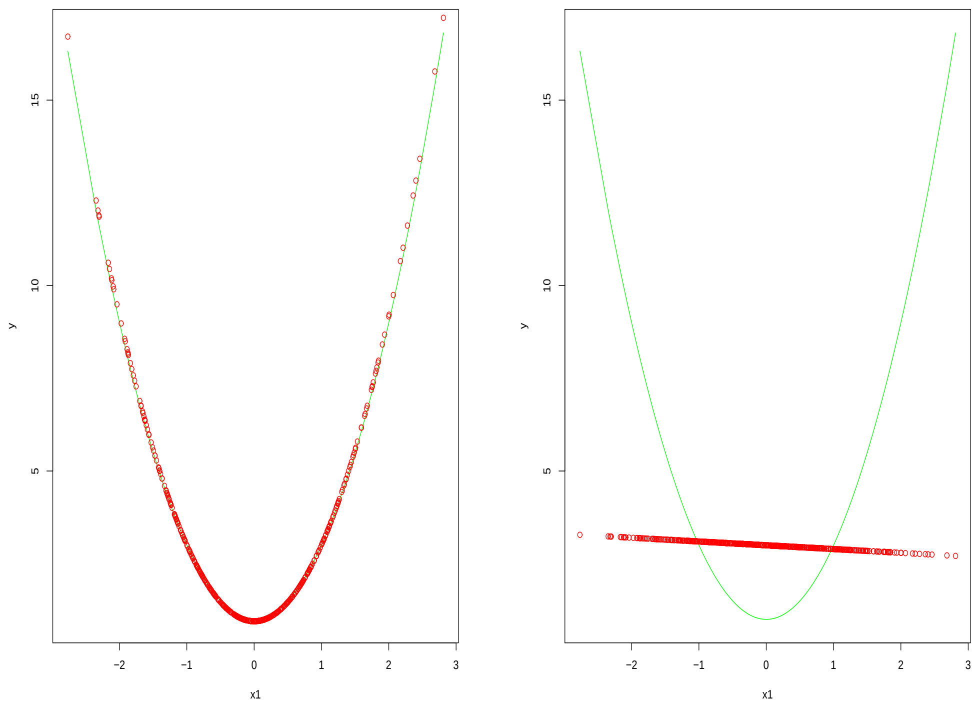

Figure 13: **Left:** The quadratic curve from Problem 1 (in green) and the linear model fit to the response `x12` (in red). **Right:** The quadratic curve from Problem 1 (in green) and the linear model fit to the response `x1` (in red).

Bagging

Problem Solutions

Problem 1 (linear regression, CART, and bagging)

See the R script `chap_4_prob_1.R`, where this problem is worked.

Part (1): When we plot the given relationship we get the plot shown in Figure 13 (left). When we fit a linear model of `y` as a function of `x12` we get back the expected result (edited slightly)

```
Call: lm(formula = y ~ x12)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.01842    0.11192    9.10   <2e-16 ***
x12          1.91678    0.06802   28.18   <2e-16 ***
```

```
---

Residual standard error: 2.013 on 498 degrees of freedom
Multiple R-squared: 0.6146,     Adjusted R-squared: 0.6138
F-statistic: 794.1 on 1 and 498 DF,  p-value: < 2.2e-16
```

From this we see that the constant and the coefficient of `x12` are estimated well. The linear model fits shown in Figure 13 (left) confirm this statement also.

Part (3): If we *don't* know that the data is generated with a quadratic term (i.e. from `x12`) but instead try to fit a linear model to explain `y` from `x1` we expect that the fit will be very poor. Using the `R` command `lm` we see that this is so

```
Call: lm(formula = y ~ x1)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.9975     0.1546  19.386   <2e-16 ***
x1           -0.1019     0.1548  -0.658    0.511
---

Residual standard error: 3.453 on 498 degrees of freedom
Multiple R-squared: 0.0008695,  Adjusted R-squared: -0.001137
F-statistic: 0.4334 on 1 and 498 DF,  p-value: 0.5106
```

The p-value for the entire fit is quite poor 0.5106 and indicates that it is not clear that the model fits the data much better than no model (i.e. using the mean of the values of `y` as the predictor). The linear model fits shown in Figure 13 (right) also confirm this statement. This is a case where we don't know the functional form (or the mapping from the input variable `x1` to the output variable `y`).

Part (4): If we *don't* know a valid model for how the data is generated we can use a nonparametric model like CART to try and learn a function for the relationship between `x1` and `y`. When we do that and then plot the CART fitted values against the true values for $f(X)$ we get the plot in Figure 14 (left). We see that these fitted values look much closer to the true values of $f(X)$ than the linear model with `x1` as the predictor.

Part (5): We now apply bagging to this dataset. The plot of the estimated $f(X)$ is shown in Figure 14 (right). This seems to be a better model than either the linear model (using `x1` as a predictor) and the direct CART model.

Problem 2 (the Freedman dataset)

See the `R` script `chap_4_prob_2.R`, where this problem is worked.

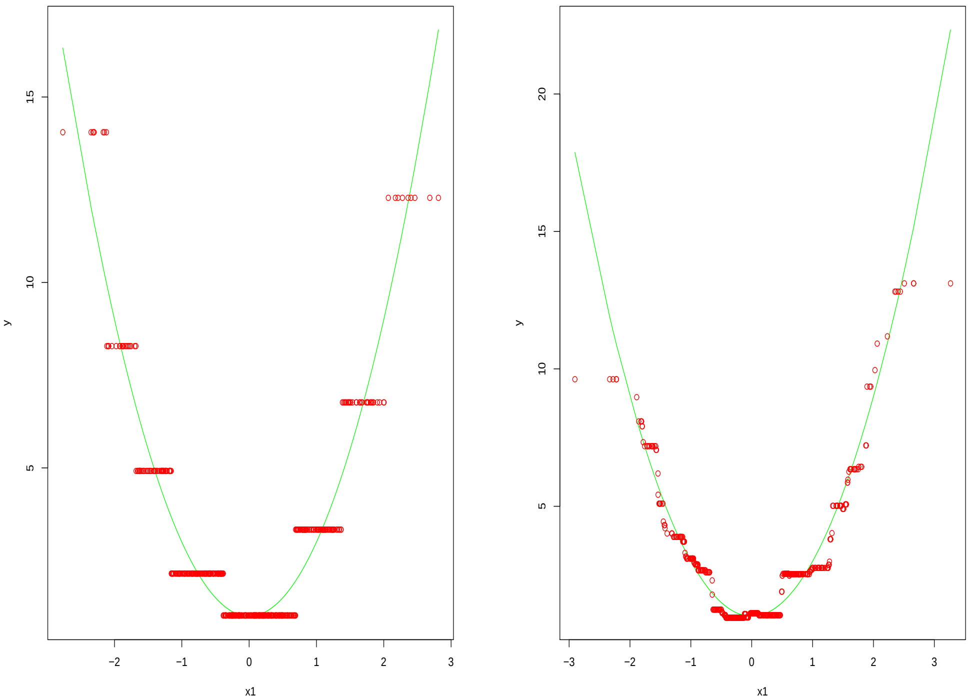

Figure 14: **Left:** The quadratic curve from Problem 1 (in green) and the CART fitted values (in red). **Right:** The quadratic curve from Problem 1 (in green) and the bagged model fit to the response `x1` (in red).

Part (1): When we compare the two model fits two each other it seems that the single CART tree is producing a smaller root-mean-square

```
[1] "CART RMS= 739.476385"
Out-of-bag estimate of root mean squared error:  802.4587
```

Problem 3 (predicting frogs)

See the `R` script `chap_4_prob_3.R`, where this problem is worked.

Part (1): The confusion matrix for the CART tree looks like

```
   y_hat_CART
      0   1
  0  19 114
  1  63  16
```

while the confusion matrix for the bagging results looks like

```
   y_hat_BAGGING
      0   1
  0 133   0
  1   0  79
```

Notice that the bagging result is making perfect predictions in sample.

Part (2): Cross-tabulating the fitted classes from CART and the bagged CART gives

```
          y_hat_BAGGING
y_hat_CART   0   1
         0  19  63
         1 114  16
```

The errors in the positions $(1, 1)$ and $(2, 2)$ of 19 and 16 respectively are approximately the same as claimed in the text.

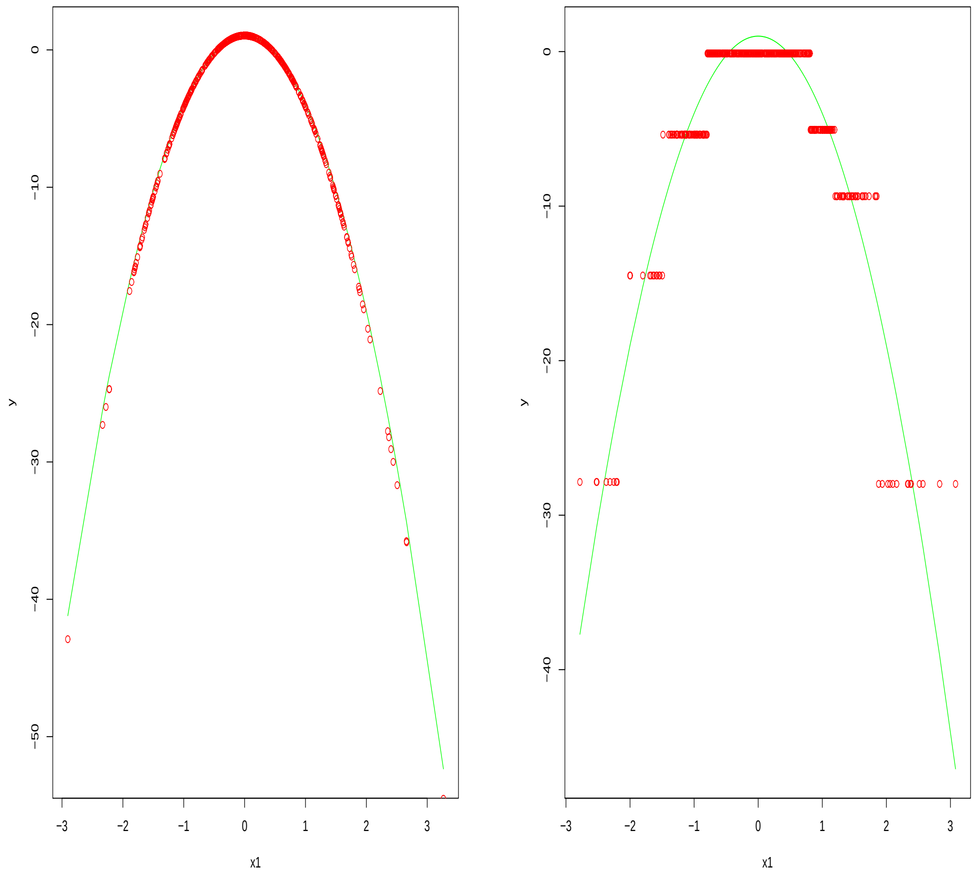

Figure 15: **Left:** The quadratic curve from Problem 1 (in green) and the linear model fit to the response `x12` (in red). **Right:** The quadratic curve from Problem 1 (in green) and the CART model fit to the response `x1` (in red).

Random Forests

Problem Solutions

Problem 1 (nonlinear curve fitting with a random forest)

See the `R` script `chap_5_prob_1.R`, where this problem is worked.

Part (1-2): When we plot the given relationship we get the plot shown in Figure 15 (left). When we fit a linear model of `y` as a function of `x12` we get back the expected result for the coefficients of the model (edited slightly)

```
Call: lm(formula = y ~ x12)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.7371     0.2609   2.825  0.00492 **
x12          -4.9758     0.1667 -29.843  < 2e-16 ***
---

Residual standard error: 4.812 on 498 degrees of freedom
Multiple R-squared: 0.6414,     Adjusted R-squared: 0.6406
F-statistic: 890.6 on 1 and 498 DF,  p-value: < 2.2e-16
```

Notice that the intercept and the coefficient of `x12` are "reasonably" well estimated.

Part (3): When we fit a CART tree to the output `y` with inputs `x12` and then make predictions given this model we get the plot shown in Figure 15 (right). We see that CART is fitting constants in the domain of `x1`.

Part (4-5): Next we use the `R` package `randomForest` to fit a model of `y` to the input of `x1`. The result of this fit is shown in Figure 16 (left). The random forest result seems to fit around the green curve

Part (6): The result of using the `R` command `partialPlot` on the random forest from this problem is shown in Figure 16 (right).

Problem 2 (various options to the `randomForest` code)

See the `R` script `chap_5_prob_2.R`, where this problem is worked. For each of the suggested arguments to the command `randomForest` we produce a plot of the actual value of `wages` as a function of the predicted value of `wages`. These plots are shown in Figure 17. See the caption there for comments. We can see the over fitting in the mean square error (MSE) of the various methods

```
[1] "MSE of the default fit  39.383563"
[1] "MSE with mtry=4  20.370278"
[1] "MSE with ntrees=100  40.125222"
[1] "MSE with ntrees=1000  39.818237"
```

Notice that the MSE of the forest with `mtry` 4 is much lower than the others.

Problem 3 (classifying diabetes with random forests)

See the `R` script `chap_5_prob_3.R`, where this problem is worked.

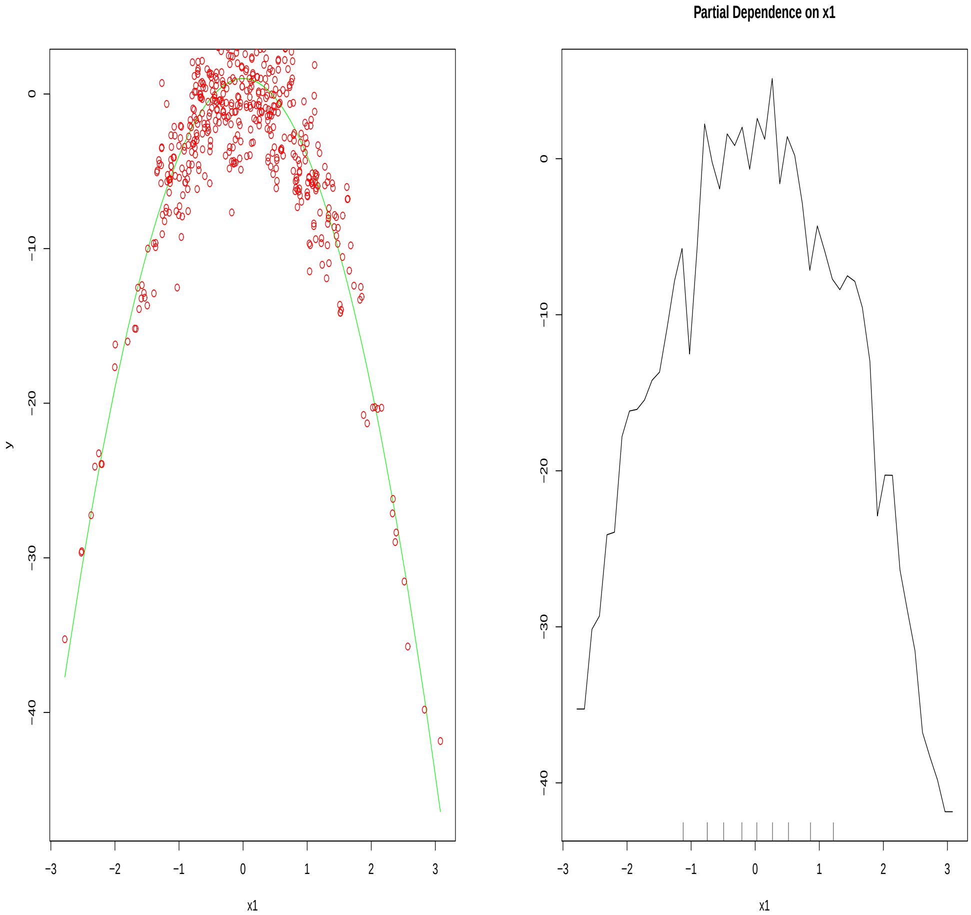

Figure 16: **Left:** The quadratic curve from Problem 1 (in green) and the random forest model fit of `y` to predictor `x1` (in red). **Right:** The partial dependence plot of the random forest (against the variable `x1`) developed to predict $y = 1 - 5x^2$.