Stats 206 - Homework 4

Clark Fitzgerald - 3013

5 November 2014

1a

```r
setwd('~/junkyard/STA206/hw4')
property = read.table('property.txt')
names(property) = c('Y', 'X1', 'X2', 'X3', 'X4')
plot(property)
```

The scatterplot is on the next page. We look at the correlation matrix:

```r
cor(property)
```

```
##              Y         X1         X2          X3         X4
## Y   1.00000000 -0.2502846  0.4137872  0.06652647 0.53526237
## X1 -0.25028456  1.0000000  0.3888264 -0.25266347 0.28858350
## X2  0.41378716  0.3888264  1.0000000 -0.37976174 0.44069713
## X3  0.06652647 -0.2526635 -0.3797617  1.00000000 0.08061073
## X4  0.53526237  0.2885835  0.4406971  0.08061073 1.00000000
```

There are no obviously strong linear relationships here. X_2 and X_4 exhibit the largest correlations.

1b

The least squares estimates, R^2, and R_a^2 can be read from the model summary output:

```r
fit1 = lm(Y ~ ., data = property)
summary(fit1)
```

```
##
## Call:
## lm(formula = Y ~ ., data = property)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1872 -0.5911 -0.0910  0.5579  2.9441
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.220e+01  5.780e-01  21.110  < 2e-16 ***
## X1          -1.420e-01  2.134e-02  -6.655 3.89e-09 ***
```

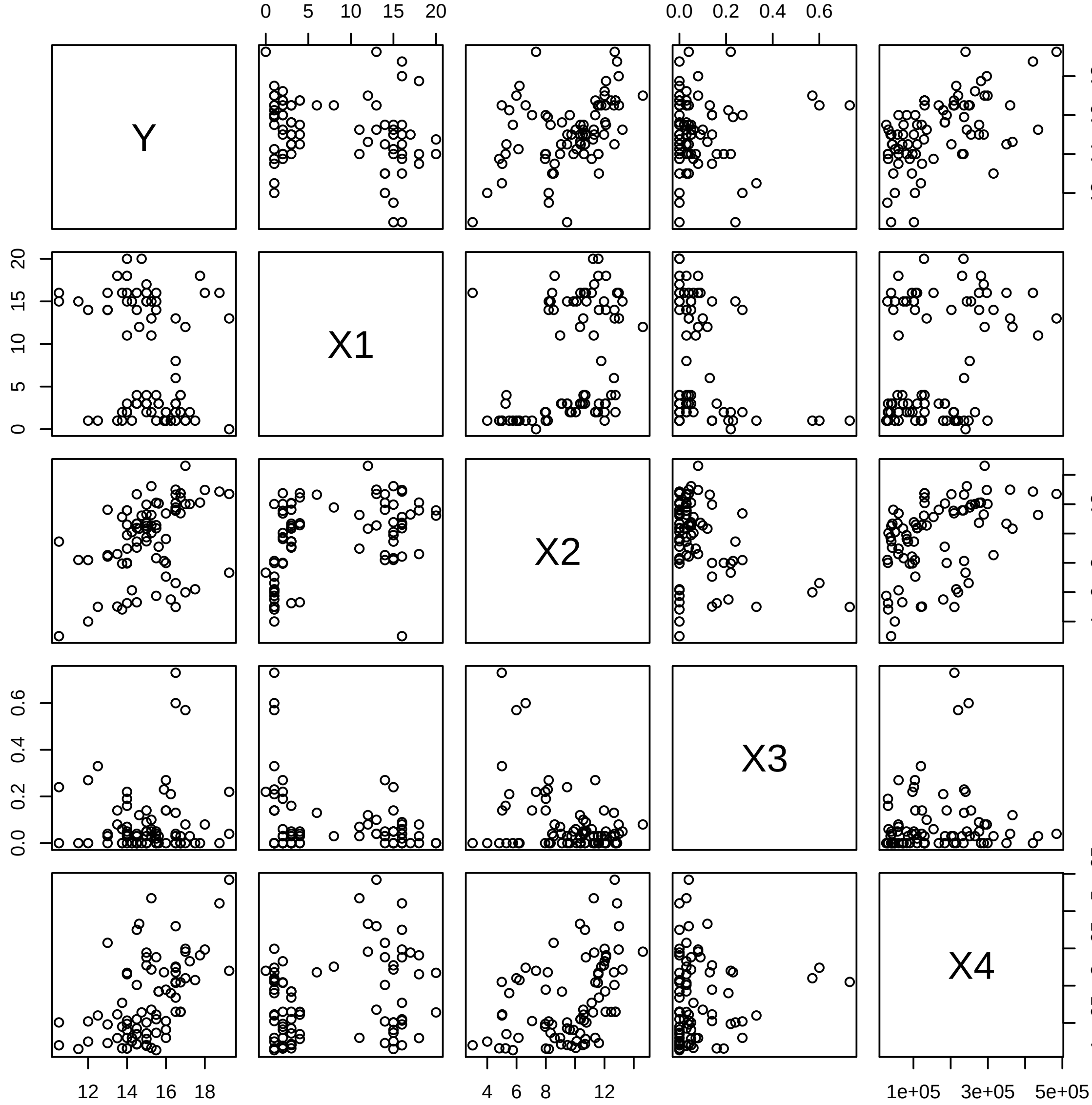

Figure 1: plot of chunk scatterplot

```
## X2           2.820e-01  6.317e-02   4.464 2.75e-05 ***
## X3           6.193e-01  1.087e+00   0.570     0.57
## X4           7.924e-06  1.385e-06   5.722 1.98e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.137 on 76 degrees of freedom
## Multiple R-squared:  0.5847, Adjusted R-squared:  0.5629
## F-statistic: 26.76 on 4 and 76 DF,  p-value: 7.272e-14
```

The fitted regression equation is:

$$Y = 12.2 - 0.142X_1 + 0.282X_2 + 0.619X_3 + 7.92 \times 10^{-6}X_4$$

MSE is 1.293 from the ANOVA table.

```
anova(fit1)
```

```
## Analysis of Variance Table
##
## Response: Y
##           Df Sum Sq Mean Sq F value    Pr(>F)
## X1         1 14.819  14.819 11.4649  0.001125 **
## X2         1 72.802  72.802 56.3262 9.699e-11 ***
## X3         1  8.381   8.381  6.4846  0.012904 *
## X4         1 42.325  42.325 32.7464 1.976e-07 ***
## Residuals 76 98.231   1.293
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1c

```
par(mfrow=c(1, 3))

plot.lm(fit1, which=c(1, 2))
boxplot(fit1$residuals)
```

The residuals versus fitted graph shows no clear patterns, which is good. We can see from the QQ plot that the tails of the distribution are a little bit heavy.

1d

```
par(mfrow=c(2, 2))

plotone = function(x) {
    plot(fit1$residuals, property[, x], xlab='residuals', ylab=x)
}

sapply(c('X1', 'X2', 'X3', 'X4'), plotone)
```

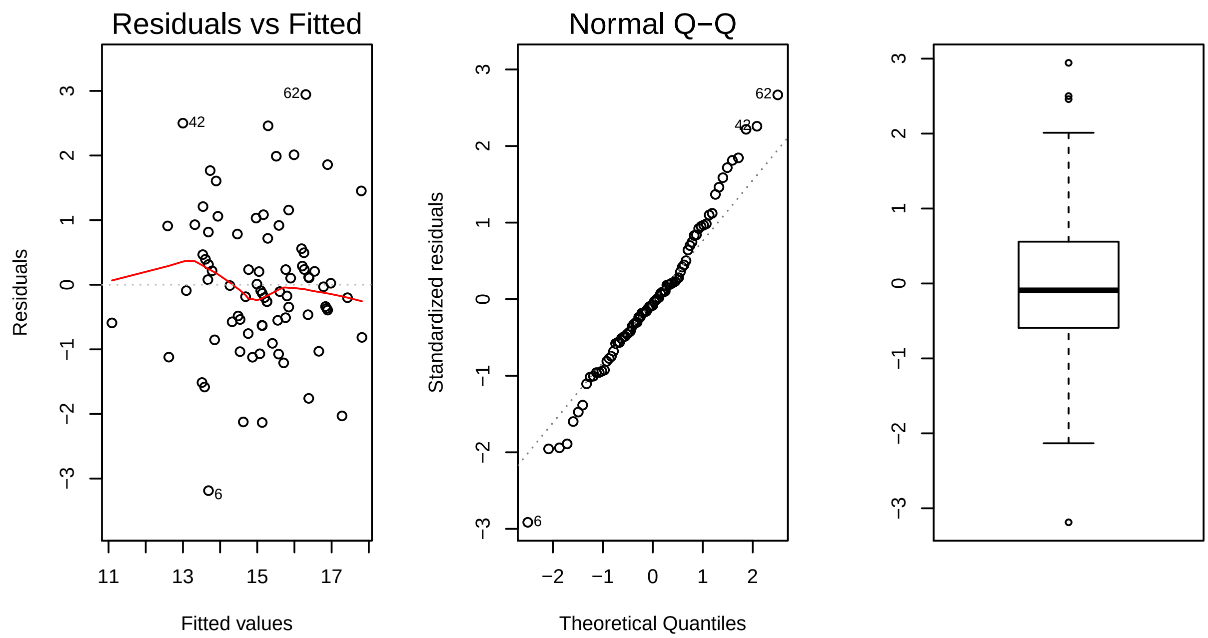

Figure 2: plot of chunk Residuals

```
par(mfrow=c(3, 2))

plot.interact = function(xx) {
    # xx is a vector of the columns names to select
    d = property[, xx]
    interaction = d[, 1] * d[, 2]
    plot(fit1$residuals, interaction, xlab='residuals',
         ylab=paste(xx, collapse = ' * '))
}

apply(combn(c('X1', 'X2', 'X3', 'X4'), 2), 2, plot.interact)
```

These plots show no obvious linear relationship between the residuals and other terms, which suggests that we don't need the interaction terms

1e

Testing whether each regression coefficient is 0 at level 0.01 with p predictors (including the intercept) is similar to testing with a single predictor, we just use a t distribution with $n - p$ degrees of freedom. For $\hat{\beta}_i, i = 0, 1, 2, 3, 4$ the null hypothesis H_0 is $\beta_i = 0$, and the alternative hypothesis H_1 is $\beta_i \neq 0$. The test statistic is given by

$$T_i^* = \frac{\hat{\beta}_i - 0}{se(\hat{\beta}_i)} \sim t(n-5)$$

under the null hypothesis. $se(\hat{\beta}_i)$ is the $i+1$ diagonal element of $MSE(X'X)^{-1}$. The decision rule is to reject H_0 if $|T_i^*| > t(0.995, n-5) \approx 2.64$. All of this information is available in the summary output:

```
qt(0.995, 81 - 5)
```

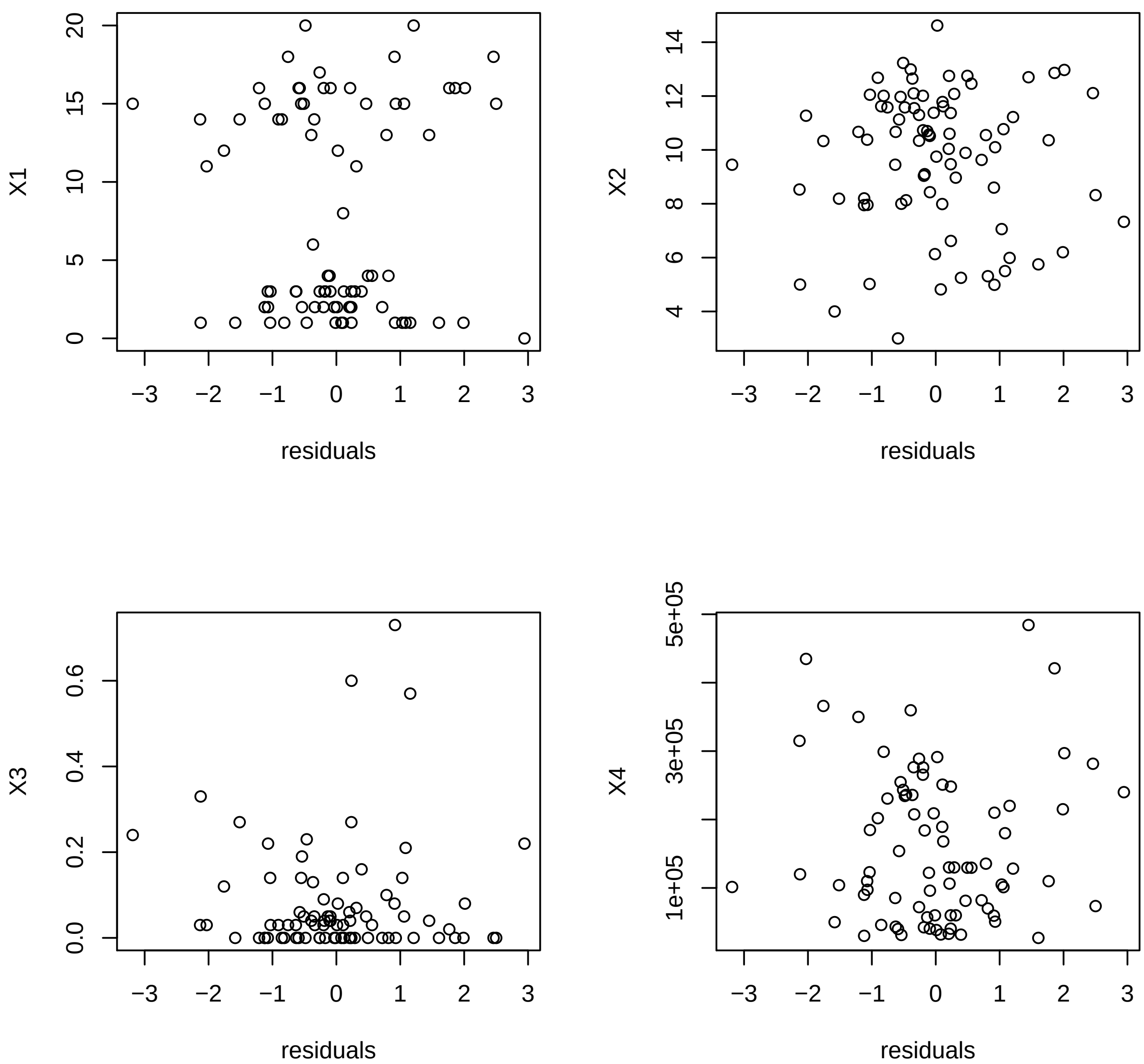

Figure 3: plot of chunk Effects

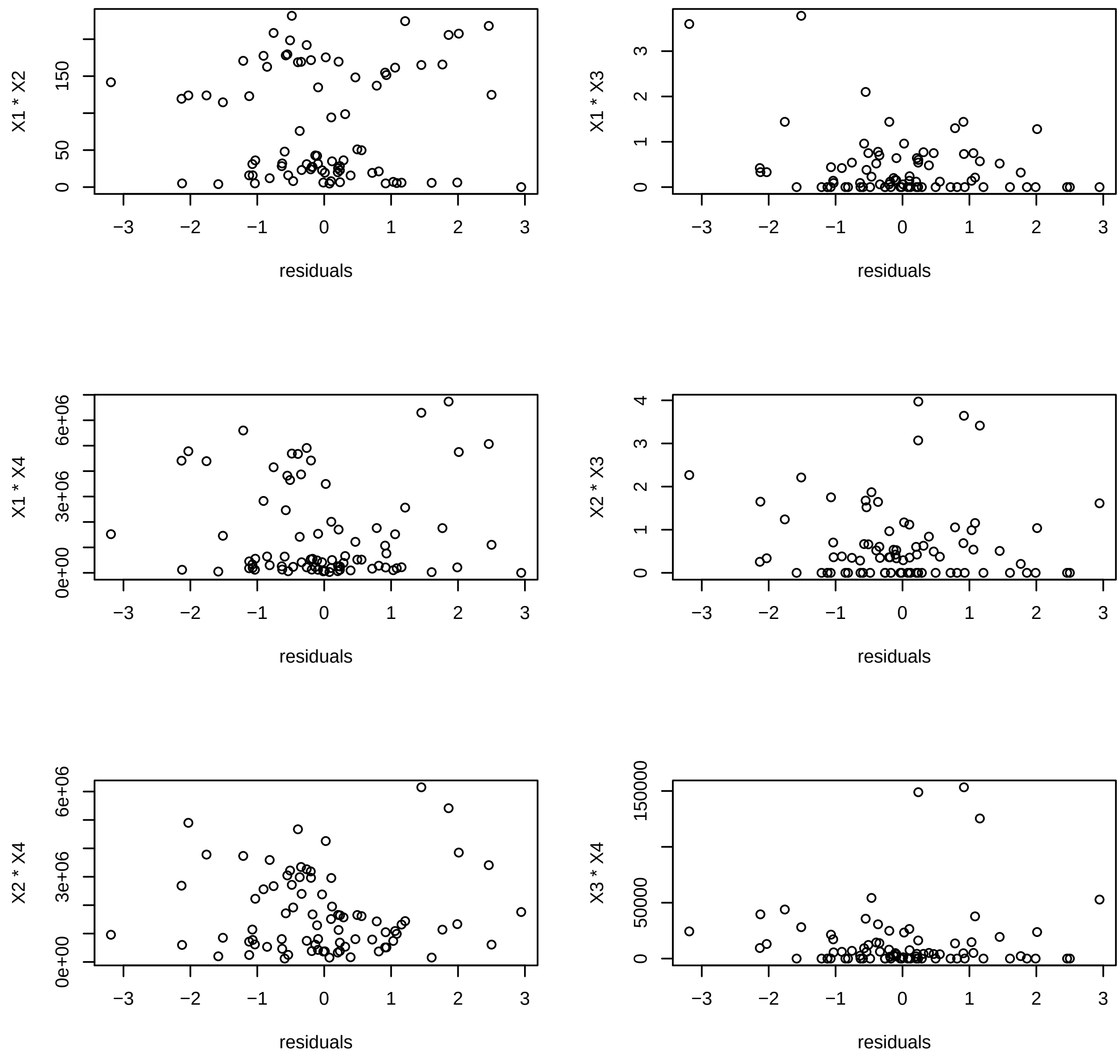

Figure 4: plot of chunk Interactions

```
## [1] 2.642078
```

```
summary(fit1)
```

```
##
## Call:
## lm(formula = Y ~ ., data = property)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1872 -0.5911 -0.0910  0.5579  2.9441
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.220e+01  5.780e-01  21.110  < 2e-16 ***
## X1          -1.420e-01  2.134e-02  -6.655 3.89e-09 ***
## X2           2.820e-01  6.317e-02   4.464 2.75e-05 ***
## X3           6.193e-01  1.087e+00   0.570     0.57
## X4           7.924e-06  1.385e-06   5.722 1.98e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.137 on 76 degrees of freedom
## Multiple R-squared:  0.5847, Adjusted R-squared:  0.5629
## F-statistic: 26.76 on 4 and 76 DF,  p-value: 7.272e-14
```

We see that in the multiple regression model, only X_3 is not significant. This implies that we could drop X_3 from the model.

1f

The ANOVA table below shows SSTO, SSR, and SSE.

```
anova.table = function(model){

    # Makes an ANOVA table for a model resulting from lm output

    a = anova(model)

    total = data.frame('SS' = sum(a[, 'Sum Sq']), 'DF' = sum(a$Df))
    error = data.frame('SS' = a['Residuals', 'Sum Sq'],
                       'DF' = a['Residuals', 'Df'])
    regression = data.frame(total - error)

    out = rbind(regression, error, total)
    out$MS = out$SS / out$DF
    row.names(out) = c('regression', 'error', 'total')
    return(out)
}
anova1 = anova.table(fit1)
anova1
```

```
##                    SS DF        MS
## regression 138.32691  4 34.581727
## error       98.23059 76  1.292508
## total      236.55750 80  2.956969
```

To test whether there is a regression relation at $\alpha = 0.01$ we use an F test. The null hypothesis H_0 is that $\beta_i = 0$ for $i = 1, 2, 3, 4$. H_1 is that not all such $\beta_i = 0$. The test statistic is

$$F^* = \frac{MSR}{MSE} \sim_{H_0} F(4, 76)$$

The decision rule is to reject H_0 if $F^* > F(0.99; 4, 76)$.

```
qf99 = qf(0.99, 4, 76)
qf99
```

```
## [1] 3.57652
```

```
Fstar = anova1['regression', 'MS'] / anova1['error', 'MS']
Fstar
```

```
## [1] 26.75553
```

```
Fstar > qf99
```

```
## [1] TRUE
```

Hence we reject H_0 and conclude that there is a significant regression relation at $\alpha = 0.01$. Note that this information is available in the last line of the `summary` output as well.

1g

Now we exclude X_3 from the model, because it failed the significance test in part e) above.

```
fit2 = lm(Y ~ X1 + X2 + X4, data = property)
summary(fit2)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X4, data = property)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0620 -0.6437 -0.1013  0.5672  2.9583
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.237e+01  4.928e-01  25.100  < 2e-16 ***
## X1          -1.442e-01  2.092e-02  -6.891 1.33e-09 ***
## X2           2.672e-01  5.729e-02   4.663 1.29e-05 ***
## X4           8.178e-06  1.305e-06   6.265 1.97e-08 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.132 on 77 degrees of freedom
## Multiple R-squared:  0.583,  Adjusted R-squared:  0.5667
## F-statistic: 35.88 on 3 and 77 DF,  p-value: 1.295e-14
```

```
anova(fit2)
```

```
## Analysis of Variance Table
##
## Response: Y
##           Df Sum Sq Mean Sq F value    Pr(>F)
## X1         1 14.819  14.819  11.566  0.001067 **
## X2         1 72.802  72.802  56.825 7.841e-11 ***
## X4         1 50.287  50.287  39.251 1.973e-08 ***
## Residuals 77 98.650   1.281
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We compare MSE, R^2 and R_a^2 with those of model 1.

```
getr2 = function(modelname){
    model = get(modelname)
    r2 = summary(model)[c('r.squared', 'adj.r.squared')]

    data.frame(R2 = r2[1], R2a = r2[2],
               MSE = anova(model)['Residuals', 'Mean Sq'],
               row.names = modelname)
}
sapply(c('fit1', 'fit2'), getr2)
```

```
##                fit1      fit2
## r.squared     0.5847496 0.5829752
## adj.r.squared 0.5628943 0.5667275
## MSE           1.292508  1.281173
```

The second fit has better adjusted R_a^2, and smaller MSE.

1h

We compare standard errors of the regression coefficient estimates for X_1, X_2, X_4 with those of model 1.

```
getse = function(modelname){
    model = get(modelname)
    coef(summary(model))[c('X1', 'X2', 'X4'), 'Std. Error']
}
sapply(c('fit1', 'fit2'), getse)
```

```
##            fit1         fit2
## X1 2.134261e-02 2.092012e-02
## X2 6.317235e-02 5.729487e-02
## X4 1.384775e-06 1.305377e-06
```

As expected, Model 2 has smaller standard errors for each coefficient. Here are 95% confidence intervals for the regression coefficients:

```r
confint(fit2, level=0.95)
```

```
##                    2.5 %       97.5 %
## (Intercept)  1.138920e+01  1.335197e+01
## X1          -1.858219e-01 -1.025074e-01
## X2           1.530784e-01  3.812557e-01
## X4           5.578873e-06  1.077755e-05
```

The confidence intervals for Model 1 are larger, since Model 1 has larger standard error around all regression coefficients.

```r
confint(fit1, level=0.95)
```

```
##                    2.5 %       97.5 %
## (Intercept)  1.104949e+01  1.335169e+01
## X1          -1.845411e-01 -9.952615e-02
## X2           1.561979e-01  4.078352e-01
## X3          -1.545232e+00  2.783919e+00
## X4           5.166283e-06  1.068232e-05
```

1i

We predict a new property under both models:

```r
x.new = data.frame(X1 = 4, X2 = 10, X3 = 0.1, X4 = 8e4)

predict(fit1, x.new, interval = 'prediction', level = 0.99)
```

```
##       fit     lwr      upr
## 1 15.1485 12.1027 18.19429
```

```r
predict(fit2, x.new, interval = 'prediction', level = 0.99)
```

```
##        fit      lwr      upr
## 1 15.11985 12.09134 18.14836
```

The fitted values and intervals are similar with both models. The intervals for Model 2 are marginally smaller.

1j

Model 2 is preferable. All of the exercises above demonstrate that including the $X3$ variable does not help the fit in any significant way. Therefore we choose the simpler model.

1k

We calculate the coefficient of partial determination

$$R^2_{Y3|124} = \frac{SSE(X_1, X_2, X_4) - SSE(X_1, X_2, X_3, X_4)}{SSE(X_1, X_2, X_4)}.$$

This measures the relative change in SSE when the X_3 term is included.

```
# Writing it in this order lets us see the effect of X3 after X4 is in the
# model.

a = anova(lm(Y ~ X1 + X2 + X4 + X3, data = property))
coefpd = a['X3', 'Sum Sq'] / a['Residuals', 'Sum Sq']
coefpd
```

```
## [1] 0.004273071
```

The coefficient of partial correlation $r_{Y3|214}$ is:

```
cpc = sqrt(coefpd) * sign(coef(fit1)['X4'])
names(cpc) = ''
cpc
```

```
##
## 0.06536873
```

The correlation coefficient between the two sets of residuals is:

```
cor(fit1$residuals, fit2$residuals)
```

```
## [1] 0.9978703
```

2a

```
par(mfrow = c(2, 5))
plothelper = function(varname, plotfunc, data=property, ...){
    # Plots individual plots with the variable name
    # ... are additional arguments to `plotfunc`
    plotfunc(data[, varname], main = varname, ...)
}
varnames = names(property)
sapply(varnames, plothelper, boxplot)
sapply(varnames, plothelper, hist, xlab='')
```

```
summary(property)
```

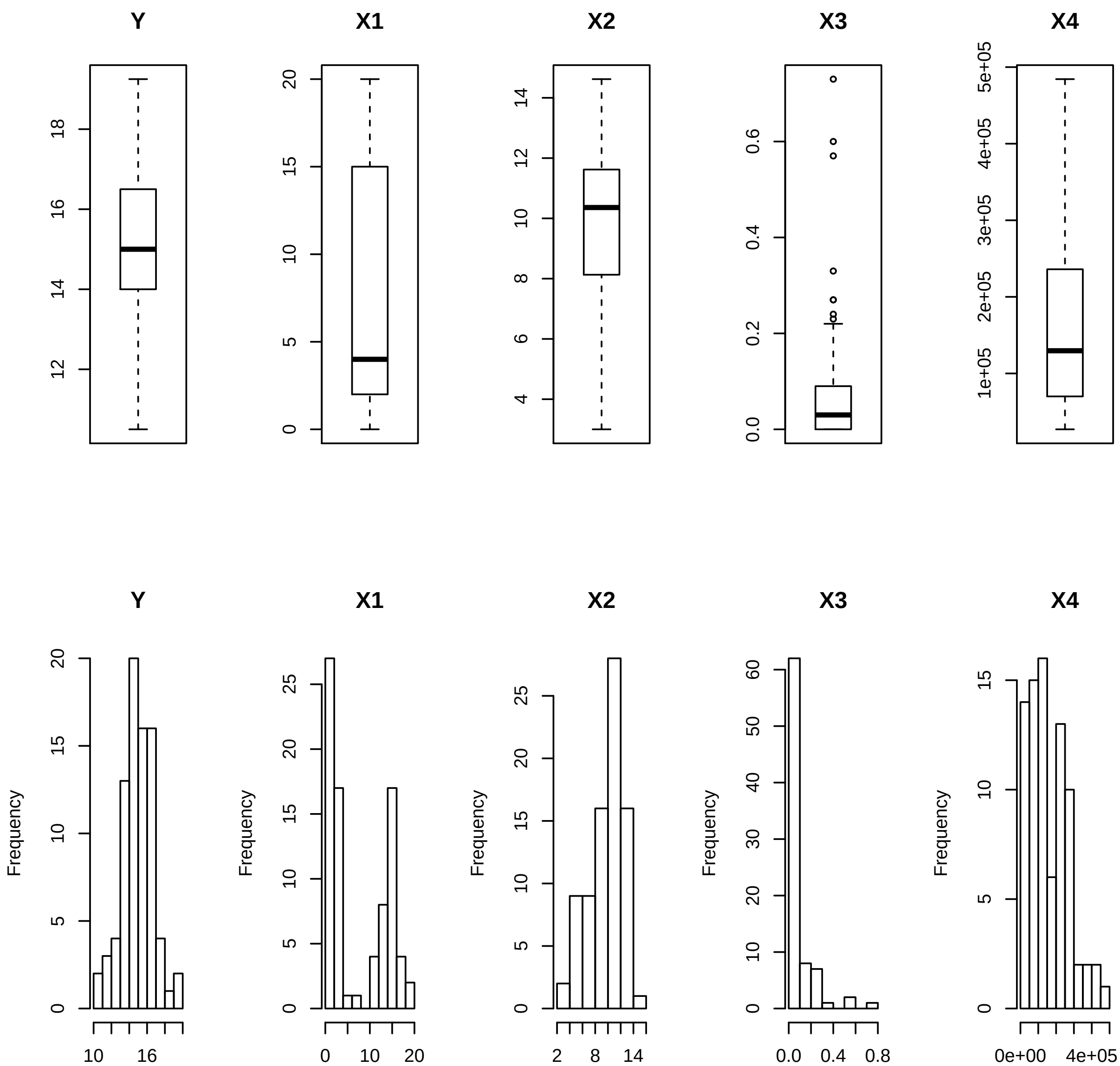

Figure 5: plot of chunk boxplots

```
##        Y               X1               X2               X3         
##  Min.   :10.50   Min.   : 0.000   Min.   : 3.000   Min.   :0.00000  
##  1st Qu.:14.00   1st Qu.: 2.000   1st Qu.: 8.130   1st Qu.:0.00000  
##  Median :15.00   Median : 4.000   Median :10.360   Median :0.03000  
##  Mean   :15.14   Mean   : 7.864   Mean   : 9.688   Mean   :0.08099  
##  3rd Qu.:16.50   3rd Qu.:15.000   3rd Qu.:11.620   3rd Qu.:0.09000  
##  Max.   :19.25   Max.   :20.000   Max.   :14.620   Max.   :0.73000  
##        X4        
##  Min.   : 27000  
##  1st Qu.: 70000  
##  Median :129614  
##  Mean   :160633  
##  3rd Qu.:236000  
##  Max.   :484290  
```

The scale of the X_4 variable is much larger than all other variables.

2b

The sample means are available from the `summary` output above. The sample standard deviations are:

```r
sapply(property, sd)
```

```
##            Y           X1           X2           X3           X4 
## 1.719584e+00 6.632784e+00 2.583169e+00 1.345512e-01 1.090990e+05 
```

The sample means and sample standard deviations for the transformed variables are:

```r
sapply(property.t, mean)
```

```
##             Y            X1            X2            X3            X4 
## -2.475792e-17 -4.830639e-18  6.347937e-18 -1.403105e-18  1.481986e-17 
```

```r
sapply(property.t, sd)
```

```
##         Y        X1        X2        X3        X4 
## 0.1118034 0.1118034 0.1118034 0.1118034 0.1118034 
```

The means are numerically 0.

Here are some functions to perform the correlation transformation, as well as the inverse transformation back to the original scale.

```r
getscale = function(X){
    # Returns scaling information
    # X is the original source of data
    # To be used together with `cor.transform`
    list(multiplier = 1 / sqrt(nrow(X) - 1),
         sd = sapply(X, sd),
         mean = sapply(X, mean))
}
```

```r
cor.transform = function(X, scaleinfo, inverse=FALSE){
    # Perform a correlation transformation on X
    # X         : matrix
    # scaleinfo : output from getscale() on original data
    # inverse   : TRUE means to transform from standardized version
    #             standardized -> original

    # Create vectors from scaleinfo to work with R's recycling rules
    # Can't call nrow on vector X
    n = max(nrow(X), 1)
    s = lapply(scaleinfo, rep, each=n)

    if (inverse){
        Xnew = X * s$sd / s$multiplier + s$mean
    }
    else{
        Xnew = s$multiplier * (X - s$mean) / s$sd
    }
    return(Xnew)
}

scl = getscale(property)
property.t = cor.transform(property, scl)

# testing correctness
property.t2 = scale(property) / sqrt(nrow(property) - 1)
pback = cor.transform(property.t, scl, inverse=TRUE)

# testing works with vectors
a = as.vector(property[1, ])
at = cor.transform(a, scl)
cor.transform(at, scl, inverse=TRUE)
```

```
##      Y X1   X2   X3     X4
## 1 13.5  1 5.02 0.14 123000
```

2c

The model equation for the standardized first-order regression model is

.

If we fit the model on the standardized data including the intercept we get:

```r
fit3 = lm(Y ~ ., data = property.t)
fit3
```

```
##
## Call:
## lm(formula = Y ~ ., data = property.t)
##
```

```
## Coefficients:
## (Intercept)           X1           X2           X3           X4
##  -4.652e-17   -5.479e-01    4.236e-01    4.846e-02    5.028e-01
```

The intercept is numerically 0, as expected.

Now we fit the standardized data excluding the intercept.

```
fit4 = lm(Y ~ . -1, data = property.t)
fit4
```

```
##
## Call:
## lm(formula = Y ~ . - 1, data = property.t)
##
## Coefficients:
##       X1        X2        X3        X4
## -0.54785   0.42365   0.04846   0.50276
```

Transforming the standardized regression coefficients back to the ones for the original model and compare with the original coefficients produces:

```
std = coef(fit3) * sd(property$Y) / scl$sd
std[-1]
```

```
##            X1            X2            X3            X4
## -1.420336e-01  2.820165e-01  6.193435e-01  7.924302e-06
```

```
coef(fit1)
```

```
##   (Intercept)            X1            X2            X3            X4
##  1.220059e+01 -1.420336e-01  2.820165e-01  6.193435e-01  7.924302e-06
```

With the appropriate transformation we can recover the original coefficients.

2d

Compare SSTO, SSE, and SSR under the standardized model with the original model. The standardized model has one more degree of freedom when the intercept is removed.

```
# standardized
anova.table(fit4)
```

```
##                   SS DF          MS
## regression 0.5847496  4 0.146187403
## error      0.4152504 77 0.005392862
## total      1.0000000 81 0.012345679
```

2e

R^2 and R_a^2 are available from the `summary` output.

```
summary(fit4)
```

```
##
## Call:
## lm(formula = Y ~ . - 1, data = property.t)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.207223 -0.038429 -0.005914  0.036276  0.191422
##
## Coefficients:
##    Estimate Std. Error t value Pr(>|t|)
## X1 -0.54785    0.08179  -6.699 3.08e-09 ***
## X2  0.42365    0.09428   4.494 2.43e-05 ***
## X3  0.04846    0.08449   0.574    0.568
## X4  0.50276    0.08728   5.760 1.64e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07344 on 77 degrees of freedom
## Multiple R-squared:  0.5847, Adjusted R-squared:  0.5632
## F-statistic: 27.11 on 4 and 77 DF,  p-value: 4.745e-14
```

R^2 is the same for the standardized model and the original. R_a^2 is marginally better for the standardized model excluding the intercept, because R considers it as using one less degree of freedom. This is not true however, because the correlation transformation centered the data, projecting it into the subspace orthogonal to the vector of 1's.

3a

We look at the correlation matrices. First we calculate through matrix multiplication using the standardized variables.

```
Xs = as.matrix(property.t[, 2:5])
Ys = as.matrix(property.t[, 1])
rxx = t(Xs) %*% Xs
rxy = t(Xs) %*% Ys
rxx
```

```
##            X1         X2          X3         X4
## X1  1.0000000  0.3888264 -0.25266347 0.28858350
## X2  0.3888264  1.0000000 -0.37976174 0.44069713
## X3 -0.2526635 -0.3797617  1.00000000 0.08061073
## X4  0.2885835  0.4406971  0.08061073 1.00000000
```

```
rxy
```

```
##             [,1]
## X1 -0.25028456
## X2  0.41378716
## X3  0.06652647
## X4  0.53526237
```

Here is the correlation between the original variables.

```
cor(property)
```

```
##               Y         X1         X2          X3         X4
## Y   1.00000000 -0.2502846  0.4137872  0.06652647 0.53526237
## X1 -0.25028456  1.0000000  0.3888264 -0.25266347 0.28858350
## X2  0.41378716  0.3888264  1.0000000 -0.37976174 0.44069713
## X3  0.06652647 -0.2526635 -0.3797617  1.00000000 0.08061073
## X4  0.53526237  0.2885835  0.4406971  0.08061073 1.00000000
```

They match the matrix calculations.

3b

The variance inflator factors are:

```
rxxinv = solve(rxx)
diag(rxxinv)
```

```
##       X1       X2       X3       X4
## 1.240348 1.648225 1.323552 1.412722
```

We confirm that $VIF_k = \frac{1}{1-R_k^2}$ by regressing each X_k on the other $X_j \neq X_k$.

```
mods = list()
mods$X1 = lm(X1 ~ X2 + X3 + X4, data=property)
mods$X2 = lm(X2 ~ X1 + X3 + X4, data=property)
mods$X3 = lm(X3 ~ X1 + X2 + X4, data=property)
mods$X4 = lm(X4 ~ X1 + X2 + X3, data=property)
sapply(mods, function(x) 1 / (1 - summary(x)$r.squared))
```

```
##       X1       X2       X3       X4
## 1.240348 1.648225 1.323552 1.412722
```

The rule of thumb is that if max $VIF_k > 10$ then multicollinearity is a cause for concern. We don't observe that here.

3c

```
mod.X4 = lm(Y ~ X4, data=property)
mod.X3X4 = lm(Y ~ X3 + X4, data=property)
mod.X4
```

```
##
## Call:
## lm(formula = Y ~ X4, data = property)
##
## Coefficients:
## (Intercept)           X4
##   1.378e+01    8.437e-06
```

```
mod.X3X4
```

```
##
## Call:
## lm(formula = Y ~ X3 + X4, data = property)
##
## Coefficients:
## (Intercept)           X3           X4
##   1.376e+01    3.007e-01    8.407e-06
```

The regression coefficients for X_4 are similar if X_3 is included or excluded. This does not surprise us, since X_3 and X_4 are not highly correlated.

```
anova(mod.X4)
```

```
## Analysis of Variance Table
##
## Response: Y
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## X4         1  67.775  67.775  31.723 2.628e-07 ***
## Residuals 79 168.782   2.136
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(mod.X3X4)
```

```
## Analysis of Variance Table
##
## Response: Y
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## X3         1   1.047   1.047  0.4842    0.4886
## X4         1  66.858  66.858 30.9213 3.626e-07 ***
## Residuals 78 168.652   2.162
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the ANOVA output we can see that X_3 makes almost no difference in SSR.

3d

```
mod.X2 = lm(Y ~ X2, data=property)
mod.X2X4 = lm(Y ~ X4 + X2, data=property)
mod.X2
```

```
##
## Call:
## lm(formula = Y ~ X2, data = property)
##
## Coefficients:
## (Intercept)           X2
##     12.4703       0.2755
```

```
mod.X2X4
```

```
##
## Call:
## lm(formula = Y ~ X4 + X2, data = property)
##
## Coefficients:
## (Intercept)           X4           X2
##   1.261e+01    6.903e-06    1.470e-01
```

X_2 and X_4 have sample correlation 0.44, and we can see that including X_4 in the model makes a large change in the regression coefficient for X_2.

```
anova(mod.X2)
```

```
## Analysis of Variance Table
##
## Response: Y
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## X2         1  40.503  40.503  16.321 0.0001231 ***
## Residuals 79 196.054   2.482
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(mod.X2X4)
```

```
## Analysis of Variance Table
##
## Response: Y
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## X4         1  67.775  67.775 33.1457 1.611e-07 ***
## X2         1   9.291   9.291  4.5438   0.03619 *
## Residuals 78 159.491   2.045
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the ANOVA table we see that $SSR(X_2|X_4)$ is small, 9.3 compared to 40.5 when it's the only term in the model. This is due to the collinearity of X_2 and X_4.