2.14

2.14.1

The *sample* function in R will select a random sample for you. The *set.seed* function below provides a starting point for the random number generator, and if you use the same seed you will get exactly the same results shown here.

```
set.seed(54321)
n <- dim(Heights)[1]
sel <- sample(1:n, floor(2*n/3))
m0 <- lm(dheight ~ mheight, Heights)
m1 <- lm(dheight ~ mheight, Heights, subset=sel)
compareCoefs(m0, m1)
```

```
Call:
1: lm(formula = dheight ~ mheight, data = Heights)
2: lm(formula = dheight ~ mheight, data = Heights, subset = sel)
             Est. 1    SE 1  Est. 2    SE 2
(Intercept) 29.9174  1.6225 29.3089  1.9850
mheight      0.5417  0.0260  0.5511  0.0318
```

The fit *m0* is to all the cases, and *m1* is to the construction set only. The estimates are quite similar but as should be expected the standard errors of the estimates are larger in m1 because the sample size is smaller.

2.14.2

First, obtain predictions for the cases not used in computing the estimates.

```
preds <- predict(m1, newdata=Heights[-sel, ])
```

Next, compute the prediction errors and square them.

```
sqPredErrors <- (Heights$dheight[-sel] - preds)^2
```

Compute and print summaries

```
meanError <- mean(sqPredErrors)
round(c("Ave. sq. pred error"= meanError,
"SD of pred" = sqrt(meanError)), 2)
```

```
Ave. sq. pred error          SD of pred
               5.27                2.30
```

Thus the SD for prediction is about 2.3 inches for a future value sampled from a population like the population from which this sample was drawn.

2.14.3

The R function predict can be used to get standard errors of fitted values,

```
se.fit <- predict(m1, new.data=Heights[-sel, ], se.fit=TRUE)$se.fit
```

The squared standard errors of prediction are $\hat{\sigma}^2 + SE_{fit}^2$, and so the average prediction variance is

```
predvar <- mean(sigmaHat(m1)^2 + se.fit^2)
round(c("Ave. sq. pred error"= predvar,
```

"SD of pred" = sqrt(predvar)), 2)

```
Ave. sq. pred error          SD of pred
               5.08                2.25
```

The linear regression model appears to match these data quite closely and so it is no surprise that the parametric approach of this subproblem matches the approach of the last subproblem. If the simple regression model were inadequate, results could have been quite different.

2.16

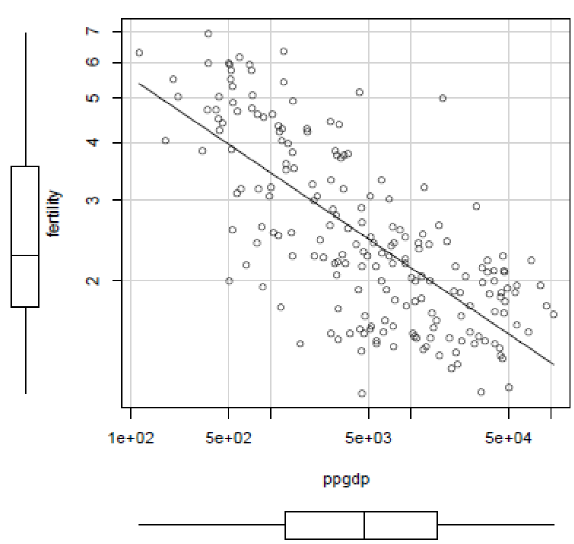
2.16.1

m1 <- lm(log(fertility) ~ log(ppgdp), UN11)
summary(m1)

```
Call:
lm(formula = log(fertility) ~ log(ppgdp), data = UN11)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7983 -0.2164  0.0267  0.2342  0.9560

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.666      0.121    22.1   <2e-16
log(ppgdp)    -0.207      0.014   -14.8   <2e-16

Residual standard error: 0.307 on 197 degrees of freedom
Multiple R-squared:  0.526,       Adjusted R-squared:  0.524
F-statistic:  219 on 1 and 197 DF,  p-value: <2e-16
```

2.16.2

The *scatterplot* function in the car makes this very easy:
scatterplot(fertility ~ ppgdp, data=UN11, log="xy", smooth=FALSE)



The *scatterplot* function always draws the fitted line unless you suppress it using the argument *reg.line=FALSE*. You could suppress the boxplots with the argument *boxplots=FALSE*. Alternatively, you can get the same graph, but with the ticks labeled in log-units, using scatterplot(log(fertility) ~ log(ppgdp), UN11, smooth=FALSE)

2.16.3

The t-test can be used, t = −14.79 with 197 df. The p-value is essentially 0, so the one-sided p-value will also be near 0. We have strong evidence that $\beta_1 < 0$, suggesting that countries with higher log(*ppgdp*) have on average lower log(*fertility*).

2.16.4

R^2= 0.526, so about 52.6% of the variability in log(*fertility*) can be explained by conditioning on log(*ppgdp*).

2.16.5

If *ppgdp* = 1000, then log(*ppgdp*) = 3. The prediction and its standard error can be obtained using the formulas in the chapter. To do the computation in R, we can use the *predict* function as follows.

new.data <- data.frame(ppgdp=1000)
(pred1 <- predict(m1, new.data, interval="prediction"))

```
     fit    lwr   upr
 1 1.235 0.6259 1.843
```

This may require a bit of explanation. The first argument to the *predict* function is the name of a regression object. If no other arguments are given, then predictions are returned for each of the original data points. To get predictions for a different point, its values must be supplied as the second argument. The function expects an object called a data frame to contain the values of the predictors for the new prediction. The variable *new.data* above is a data frame with just 1 value, *ppgdp*=1000. We do not need to take logarithms here because of the way that *m1* was defined, with the log in the de_nition of the mean function, so m1 will take the log for us. If we wanted predictions at, say *ppgdp* = 1000, 2000, 5000, we would have defined *new.data* to be data.frame(ppgdp=c(1000, 2000, 5000)).

The predict function was then used with the additional argument *interval="prediction"* to give the 95% prediction interval in log scale. Exponentiating the end points

```
exp(pred1)
     fit  lwr   upr
 1 3.437 1.87 6.317
```

gives a surprisingly wide interval for the predicted *fertility*.

2.16.6

This problem should be solved using an interactive program. Although R is weak in general on interactive graphics, the *identify* function will do the trick:

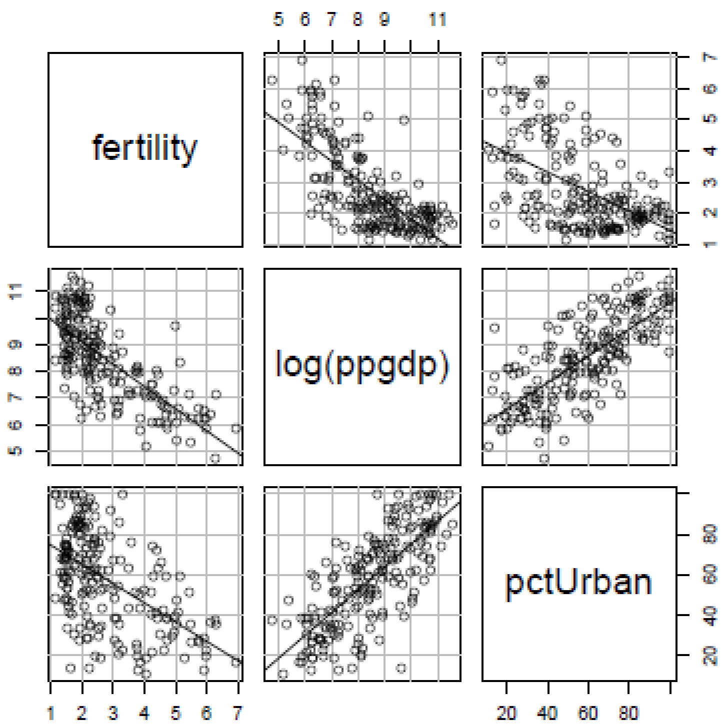
plot(log10(fertility) ~ log10(ppgdp), UN11)
abline(m1)
with(UN11, identify(log10(ppgdp), log10(fertility),
row.names(UN11)))

Niger, followed by Somalia and Zambia, have the largest fertility rates, while Bosnia-Herzegovina, Macao and Hong Kong have the lowest. To find the residuals, it is convenient to plot the residuals versus either the _tted values or the predictor. You can use the *residualPlot* function in the *car* package for this purpose

residualPlot(m1, id.n=4)
Equatorial Guinea and Angola have the largest positive residuals, and are therefore the 2 countries with fertility rates that are much larger than expected after conditioning on *ppgdp*. Moldova, and Bosnia-Herzegovina have negative residuals, and so have low fertility rates given their *ppgdp*.

3.2

3.2.1



All the variables appear to be strongly linearly related. Thus both of the predictors appear to be marginally related to *fertility*.

3.2.2

summary(m1 <- lm(fertility ~ log(ppgdp), UN11))$coef

```
             Estimate Std. Error t value  Pr(>|t|)
(Intercept)   8.0097    0.36529   21.93 9.338e-55
log(ppgdp)   -0.6201    0.04245  -14.61 3.165e-33
```

summary(m2 <- lm(fertility ~ pctUrban, UN11))$coef

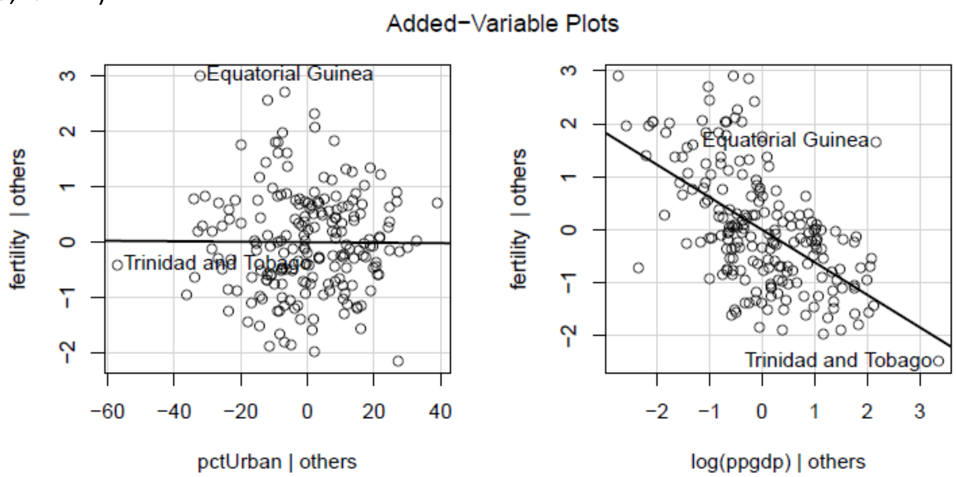
```
             Estimate Std. Error t value  Pr(>|t|)
(Intercept)  4.55982   0.213681  21.339 4.060e-53
pctUrban    -0.03105   0.003421  -9.076 1.178e-16
```

3.2.3

Although the outline of Section 3.1 could be followed, if using Rth added-variable plots can be obtained directly:
m3 <- update(m2, ~ . + log(ppgdp))
summary(m3)

```
Call:
lm(formula = fertility ~ pctUrban + log(ppgdp), data = UN11)

Residuals:
   Min     1Q Median     3Q    Max 
-2.151 -0.649 -0.066  0.632  2.991 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.993270   0.399337   20.02   <2e-16
pctUrban    -0.000439   0.004266   -0.10     0.92
log(ppgdp)  -0.615142   0.064156   -9.59   <2e-16

Residual standard error: 0.933 on 196 degrees of freedom
Multiple R-squared:  0.52,	Adjusted R-squared:  0.515 
F-statistic:  106 on 2 and 196 DF,  p-value: <2e-16
```

avPlots(m3, id.n=1)



The plot for log(*ppgdp*) suggests that this is an important variable adjusting for *pctUrban*, but the added-variable plot for *pctUrban* shows essentially no linear trend and it is quite likely that the variability explained by this variable is a subset of the variability explained by log(*ppgdp*).

3.2.4

m4 <- lm(log(ppgdp) ~ pctUrban, UN11)
m5 <- lm(residuals(m2) ~ residuals(m4))
summary(m5)$coef

```
                 Estimate Std. Error     t value  Pr(>|t|)
(Intercept)    -1.986e-16    0.06596 -3.010e-15 1.000e+00
residuals(m4)  -6.151e-01    0.06399 -9.613e+00 3.504e-18
```

The coefficients for log(*ppgdp*) are identical in *m3* and *m5*, although one is printed in scientific notation and the other is standard notation and not very many digits are shown.

3.2.5

The residuals can be shown to be the same by either plotting one set against the other or by subtracting them.

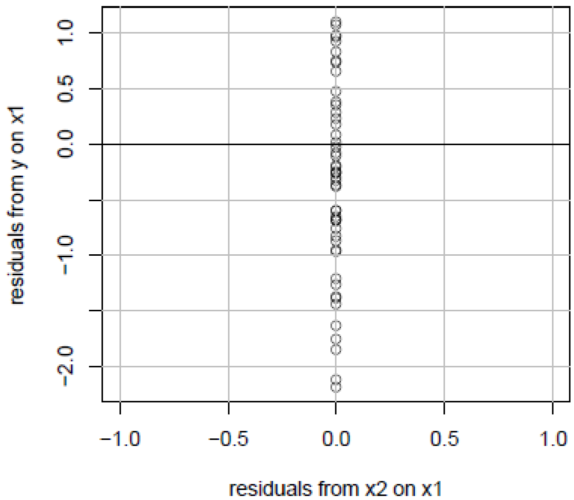
3.2.6

The added-variable plot computation has the *df* wrong, with 1 extra *df*. After correcting the *df*, the computations are identical.

3.4

3.4.1

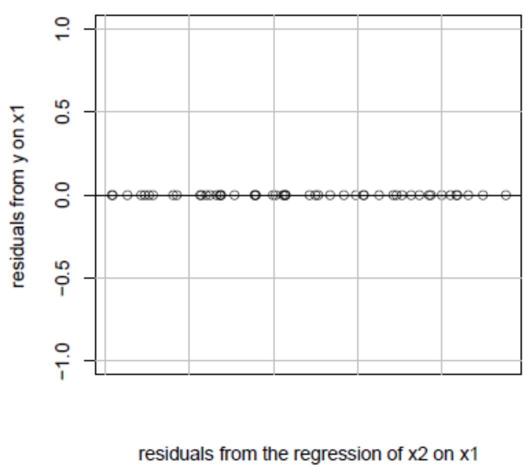
Since X_2 is an exact linear function of X_1, the residuals from the regression of X_2 on X_1 will all be 0, and so the plot will look like this:



Since X_1 and X_2 are the same apart from a constant multiplier, X_2 explains no extra variation after X_1 and a model that includes X_1 cannot provide an estimate for the effect of X_2 adjusted for X_1. In general, if X_1 and X_2 are highly correlated, the variability on the horizontal axis of an added-variable plot will be very small compared to the variability of the original variable. The coefficient for such a variable will be very poorly estimated.

3.4.2

Since $Y = 3X_1$ the residuals from the regression of Y on X_1 will all be 0, and so the plot will look like



In general, if Y and X_1 are highly correlated, the variability on the vertical axis of an added-variable plot will be very small compared to the variability of the original variable, and we will get an approximately null plot.

3.4.3

If X_1 is uncorrelated with both X_2 and Y, then these two plots will be the same.

3.4.4

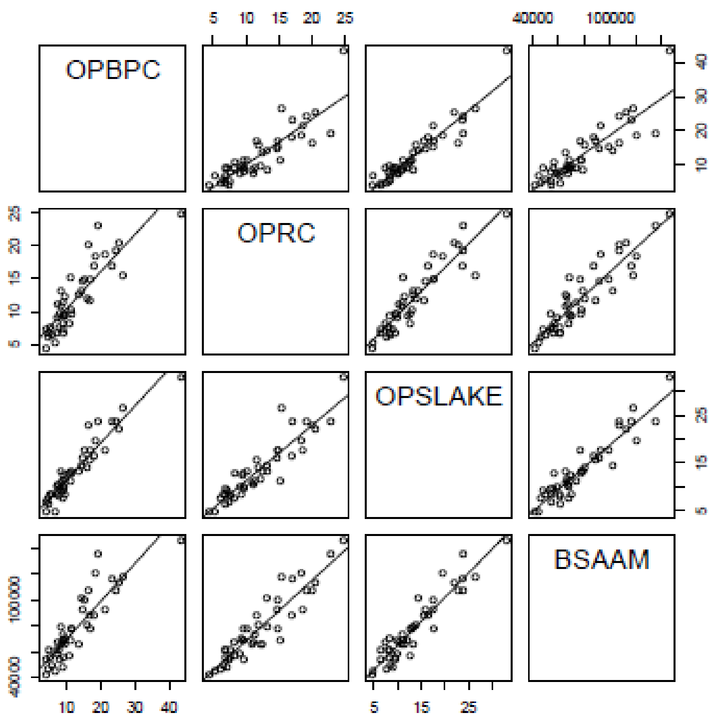
Since the vertical variable is the residuals from the regression of Y on X_1, the vertical variation in the added-variable plot is never larger than the vertical variation in the plot of Y versus X_2.

3.6

3.6.1

The scatterplot matrix is

```
scatterplotMatrix(~ OPBPC + OPRC + OPSLAKE + BSAAM, water,
smooth=FALSE, spread=FALSE, diagonal="none")
```



All the variables are strongly and positively related, which can lead to problems in understanding coefficients, since each of the 3 predictors is nearly the same variable. The correlation matrix and regression output are

```
cor(water[, c("OPBPC", "OPRC","OPSLAKE","BSAAM")])
```

	OPBPC	OPRC	OPSLAKE	BSAAM
OPBPC	1.0000	0.8647	0.9433	0.8857
OPRC	0.8647	1.0000	0.9191	0.9196
OPSLAKE	0.9433	0.9191	1.0000	0.9384
BSAAM	0.8857	0.9196	0.9384	1.0000

3.6.2

The regression summary is

```
summary(m1 <- lm(BSAAM ~ OPBPC + OPRC + OPSLAKE, data=water))
```

```
Call:
lm(formula = BSAAM ~ OPBPC + OPRC + OPSLAKE, data = water)

Residuals:
   Min     1Q Median     3Q    Max
-15964  -6492   -404   4742  19921


Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  22991.9     3545.3    6.49  1.1e-07
OPBPC           40.6      502.4    0.08   0.9360
OPRC          1867.5      647.0    2.89   0.0063
OPSLAKE       2354.0      771.7    3.05   0.0041

Residual standard error: 8300 on 39 degrees of freedom
Multiple R-squared:  0.902,        Adjusted R-squared:  0.894
F-statistic:  119 on 3 and 39 DF,  p-value: <2e-16
```

The variable *OPBPC* is unimportant after the others because of its tiny p-value, in spite of its high correlation with the response of more than 0.86. This could be verified using the added-variable plot for *OPBPC*. The value of R^2 = 0.902 suggests that most of the variation in *BSAAM* is explained by these 3 variables.