Lecture 8 - 2x2 Tables, ANOVA, and Linear Regression

94-842

February 9, 2017

Contents

Agenda

- 2x2 tables, j x k tables
- ANOVA
- Linear regression
  - Fitting linear regression models in R
  - Diagnostic plots
  - Interpreting regression coefficients
  - Testing significance of factor variables

Let's begin by loading the packages we'll need to get started

```
library(MASS)
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(knitr)
#library(reshape2)
```

```
# Rename the columns to have more descriptive names
colnames(birthwt) <- c("birthwt.below.2500", "mother.age", "mother.weight",
    "race", "mother.smokes", "previous.prem.labor", "hypertension", "uterine.irr",
    "physician.visits", "birthwt.grams")

# Transform variables to factors with descriptive levels
birthwt <- mutate(birthwt,
                  race = as.factor(mapvalues(race, c(1, 2, 3),
                                             c("white","black", "other"))),
                  mother.smokes = as.factor(mapvalues(mother.smokes,
                                                      c(0,1), c("no", "yes"))),
                  hypertension = as.factor(mapvalues(hypertension,
                                                     c(0,1), c("no", "yes"))),
                  uterine.irr = as.factor(mapvalues(uterine.irr,
                                                    c(0,1), c("no", "yes")))
)
```

Tests for 2x2 tables

Here's an example of a 2 x 2 table that we might want to run a test on. This one looks at low birthweight broken down by mother's smoking status. You can think of it as another approach to the t-test problem, this time looking at indicators of low birth weight instead of the actual weights.

First, let's build our table using the `table()` function (we did this back in Lecture 5)

```
weight.smoke.tbl <- with(birthwt, table(birthwt.below.2500, mother.smokes))
weight.smoke.tbl
```

```
##                    mother.smokes
## birthwt.below.2500 no yes
##                  0 86  44
##                  1 29  30
```

We also previously calculated the odds ratio for this table, finding that it was approximately 2. This indicated that the odds of low birthweight double when the mother smokes.

To test for significance, we just need to pass our 2 x 2 table into the appropriate function. Here's the result of using fisher's exact test by calling `fisher.test`

```
birthwt.fisher.test <- fisher.test(weight.smoke.tbl)
birthwt.fisher.test
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  weight.smoke.tbl
```

```
## p-value = 0.03618
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.028780 3.964904
## sample estimates:
## odds ratio
##   2.014137
```

```
attributes(birthwt.fisher.test)
```

```
## $names
## [1] "p.value"     "conf.int"    "estimate"    "null.value"  "alternative"
## [6] "method"      "data.name"
##
## $class
## [1] "htest"
```

As when using the t-test, we find that there is a significant association between smoking an low birth weight.

Interpretation: The odds of low birth weight are 2.01 times greater when the mother smokes than when the mother does not smoke.

You can also use the chi-squared test via the `chisq.test` function. This is the test that you may be more familiar with from your statistics class.

```
chisq.test(weight.smoke.tbl)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  weight.smoke.tbl
## X-squared = 4.2359, df = 1, p-value = 0.03958
```

You get essentially the same answer by running the chi-squared test, but the output isn't as useful. In particular, you're not getting an estimate or confidence interval for the odds ratio. This is why I prefer `fisher.exact()` for testing 2 x 2 tables.

Tests for j x k tables

Here's a small data set on party affiliation broken down by gender.

```
# Manually enter the data
politics <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(politics) <- list(gender = c("F", "M"),
                     party = c("Democrat","Independent", "Republican"))

politics # display the data
```

```
##       party
## gender Democrat Independent Republican
##      F      762         327        468
##      M      484         239        477
```

We may be interested in asking whether men and women have different party affiliations.

The answer will be easier to guess at if we convert the rows to show proportions instead of counts. Here's one way of doing this.

```
politics.prop <- prop.table(politics, 1)
politics.prop
```

```
##       party
## gender  Democrat Independent Republican
##      F 0.4894027   0.2100193  0.3005780
##      M 0.4033333   0.1991667  0.3975000
```

By looking at the table we see that Female are more likely to be Democrats and less likely to be Republicans.

We still want to know if this difference is significant. To assess this we can use the chi-squared test (on the counts table, not the proportions table!).

```
chisq.test(politics)
```

```
##
##  Pearson's Chi-squared test
##
## data:  politics
## X-squared = 30.07, df = 2, p-value = 2.954e-07
```

There isn't really a good one-number summary for general j x k tables the way there is for 2 x 2 tables. One thing that we may want to do at this stage is to ignore the Independent category and just look at the 2 x 2 table showing the counts for the Democrat and Republican categories.

```
politics.dem.rep <- politics[,c(1,3)]
politics.dem.rep
```

```
##       party
## gender Democrat Republican
##      F      762        468
##      M      484        477
```

```
# Run Fisher's exact test
fisher.test(politics.dem.rep)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  politics.dem.rep
## p-value = 6.806e-08
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.347341 1.910944
## sample estimates:
## odds ratio
##   1.604345
```

We see that women have significantly higher odds of being Democrat compared to men.

Plotting the table values with confidence

It may be useful to represent the data graphically. Here's one way of doing so with the `ggplot2` package. Note that we plot the **proportions** not the counts.

1. Convert the table into something ggplot2 can process by using `melt()` from the `reshape` package.

```
library(reshape2)
politics.prop
```

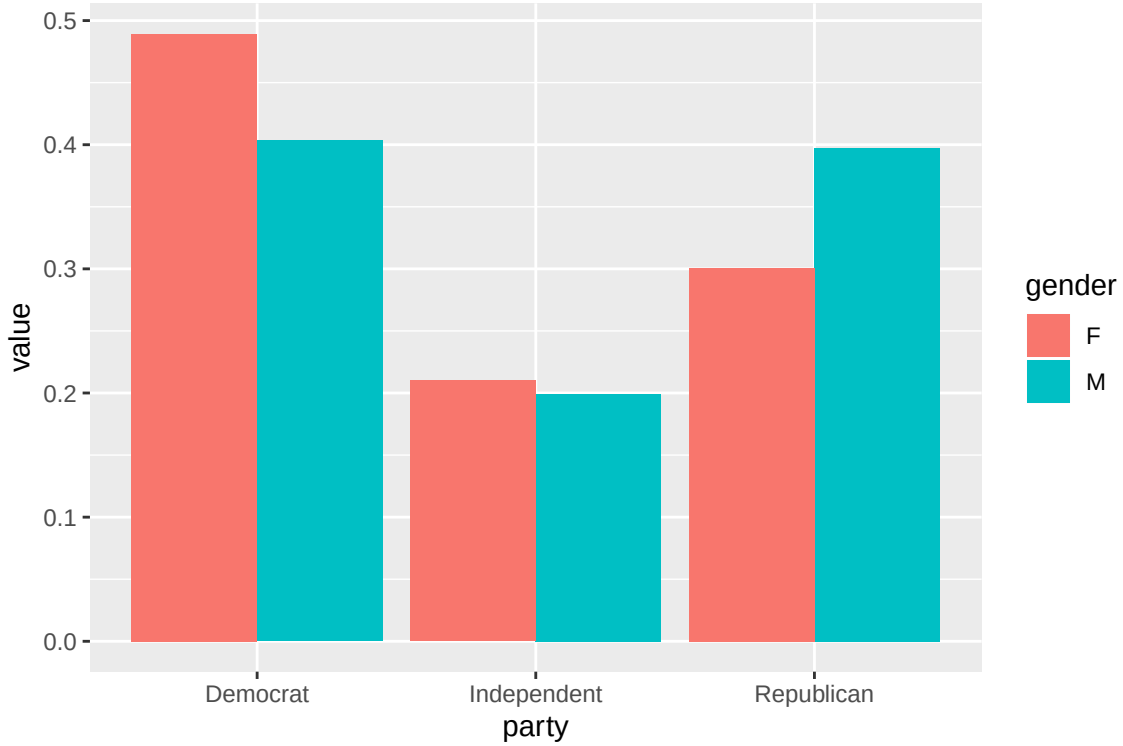
```
##       party
## gender  Democrat Independent Republican
##      F 0.4894027   0.2100193  0.3005780
##      M 0.4033333   0.1991667  0.3975000
```

```
politics.melt <- melt(politics.prop, id=c("gender","party"))
politics.melt
```

```
##   gender       party     value
## 1      F    Democrat 0.4894027
## 2      M    Democrat 0.4033333
## 3      F Independent 0.2100193
## 4      M Independent 0.1991667
## 5      F  Republican 0.3005780
## 6      M  Republican 0.3975000
```

2. Create a ggplot2 object, and plot with `geom_barplot()`

```
ggplot(politics.melt, aes(x=party, y=value, fill=gender)) + geom_bar(position="dodge", stat="identity")
```



This figure is a nice alternative to displaying a table. One thing we might want to add is a way of gauging the statistical significance of the differences in height. We'll do so by adding error bars.

Adding error bars to bar plots

Remember, ggplot wants everything you plot to be sitting nicely in a data frame. Here's some code that will calculate the relevant values and do the plotting.

1. Get the data into a form that's easy to work with.

```
# Form into a long data frame
politics.count.melt <- melt(politics, id=c("gender", "party"))
```

```
# print
politics.count.melt
```

```
##   gender       party value
## 1      F    Democrat   762
## 2      M    Democrat   484
## 3      F Independent   327
## 4      M Independent   239
## 5      F  Republican   468
## 6      M  Republican   477
```

```
# Add a column of marginal counts
politics.count.melt <- mutate(politics.count.melt,
                              totals = rowSums(politics)[gender])
# print
politics.count.melt
```

```
##   gender       party value totals
## 1      F    Democrat   762   1557
## 2      M    Democrat   484   1200
## 3      F Independent   327   1557
## 4      M Independent   239   1200
## 5      F  Republican   468   1557
## 6      M  Republican   477   1200
```

2. Calculate confidence intervals.

To calculate confidence intervals for the proportions, we can use `prop.test` or `binom.test`. Essentially you call these functions the numerator and denominator for the proportion.

We'll do this with a ddply call. First, let's look at an example of how the `prop.test` works.

```
# Suppose that we had 80 coin flips, 27 of which came up heads.
# How can we get a 95% CI for the true probability that this coin comes up H?
coin.test <- prop.test(27, 80)
coin.test
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  27 out of 80, null probability 0.5
## X-squared = 7.8125, df = 1, p-value = 0.005189
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.2379394 0.4528266
## sample estimates:
##      p
## 0.3375
```

```
coin.test$conf.int
```

```
## [1] 0.2379394 0.4528266
## attr(,"conf.level")
## [1] 0.95
```

```
politics.toplot <- ddply(politics.count.melt, ~ gender + party, transform,
                         prop = value / totals,
                         lower = prop.test(value, totals)$conf.int[1],
```

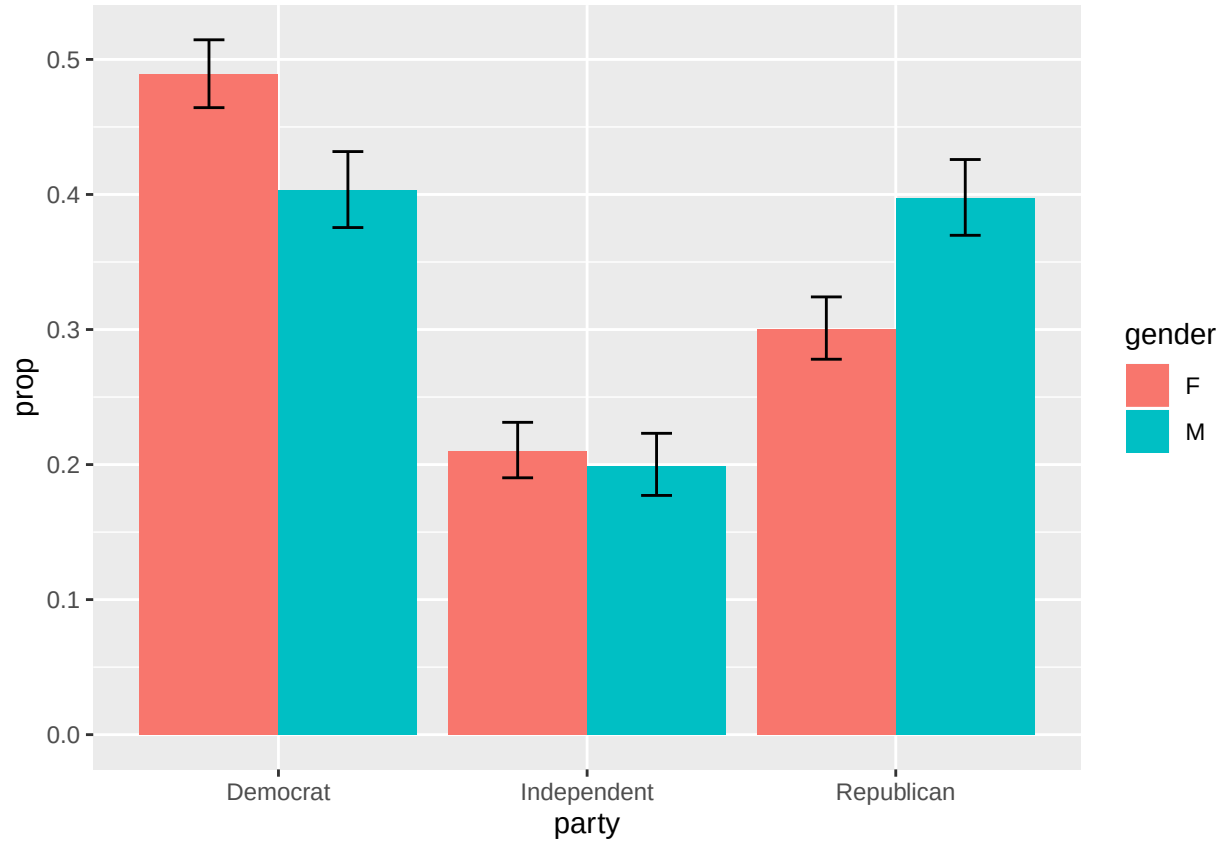
```
                        upper = prop.test(value, totals)$conf.int[2])

politics.toplot
```

```
##   gender       party value totals      prop     lower     upper
## 1      F    Democrat   762   1557 0.4894027 0.4643094 0.5145489
## 2      F Independent   327   1557 0.2100193 0.1902041 0.2312844
## 3      F  Republican   468   1557 0.3005780 0.2780035 0.3241480
## 4      M    Democrat   484   1200 0.4033333 0.3755178 0.4317750
## 5      M Independent   239   1200 0.1991667 0.1771484 0.2231402
## 6      M  Republican   477   1200 0.3975000 0.3697701 0.4258939
```

3. Combine the confidence intervals into the data frame

4. Use `ggplot()`, `geom_bar()` and `geom_errorbar()` to construct the plots

```
ggplot(politics.toplot, aes(x=party, y=prop, fill=gender)) +
  geom_bar(position="dodge", stat="identity") +
  geom_errorbar(aes(ymin=lower, ymax=upper),
                width=.2,                    # Width of the error bars
                position=position_dodge(0.9))
```



ANOVA

You can think of ANOVA (analysis of variance) as a more general version of the t-test, or a special case of linear regression in which all covariates are factors.

Let's go back to our favourite birthwt data set from the MASS library.

One-way ANOVA example

Question: Is there a significant association between race and birthweight?

Here's a table showing the mean and standard error of birthweight by race.

```
ddply(birthwt, ~ race, summarize,
      mean.bwt = mean(birthwt.grams),
      sd.bwt = sd(birthwt.grams) / sqrt(length(birthwt.grams)))
```

```
##    race mean.bwt    sd.bwt
## 1 black 2719.692 125.25621
## 2 other 2805.284  88.23008
## 3 white 3102.719  74.28957
```

It looks like there's some association, but we don't yet know if it's statistically significant. Note that if we had just two racial categories in our data, we could run a t-test. Since we have more than 2, we need to run a 1-way analysis of variance (ANOVA).

Terminology: a k-way ANOVA is used to assess whether the mean of an outcome variable is constant across all combinations of k factors. The most common examples are 1-way ANOVA (looking at a single factor), and 2-way ANOVA (looking at two factors).

We'll use the `aov()` function. For convenience, `aov()` allows you to specify a formula.

```
summary(aov(birthwt.grams ~ race, data = birthwt))
```

```
##              Df   Sum Sq Mean Sq F value  Pr(>F)
## race          2  5015725 2507863   4.913 0.00834 **
## Residuals   186 94953931  510505
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value is significant at the 0.05 level, so the data suggests that there is an association between birthweight and race. In other words, average birthweight varies across the three racial groups considered in the data.

Linear regression

Linear regression is just a more general form of ANOVA. It allows you to model effects of continuous variables.

> **linear regression** is used to model linear relationship between an outcome variable, y, and a set of *covariates* or *predictor variables* $x_1, x_2, \ldots, x_p$.

For our first example we'll look at a small data set in which we're interested in predicting the crime rate per million population based on socio-economic and demographic information at the state level.

Let's first import the data set and see what we're working with.

```
# Import data set
crime <- read.table("http://www.andrew.cmu.edu/user/achoulde/94842/data/crime_simple.txt", sep = "\t", h
```

The variable names that this data set comes with are very confusing, and even misleading.

R: Crime rate: # of offenses reported to police per million population

Age: The number of males of age 14-24 per 1000 population

S: Indicator variable for Southern states (0 = No, 1 = Yes)

Ed: Mean # of years of schooling x 10 for persons of age 25 or older

Ex0: 1960 per capita expenditure on police by state and local government

Ex1: 1959 per capita expenditure on police by state and local government

LF: Labor force participation rate per 1000 civilian urban males age 14-24

M: The number of males per 1000 females

N: State population size in hundred thousands

NW: The number of non-whites per 1000 population

U1: Unemployment rate of urban males per 1000 of age 14-24

U2: Unemployment rate of urban males per 1000 of age 35-39

W: Median value of transferable goods and assets or family income in tens of $

X: The number of families per 1000 earning below 1/2 the median income

We really need to give these variables better names

```
# Assign more meaningful variable names
colnames(crime) <- c("crime.per.million", "young.males", "is.south", "average.ed",
                     "exp.per.cap.1960", "exp.per.cap.1959", "labour.part",
                     "male.per.fem", "population", "nonwhite",
                     "unemp.youth", "unemp.adult", "median.assets", "num.low.salary")

# Convert is.south to a factor
# Divide average.ed by 10 so that the variable is actually average education
# Convert median assets to 1000's of dollars instead of 10's
crime <- mutate(crime, is.south = as.factor(is.south),
                average.ed = average.ed / 10,
                median.assets = median.assets / 100)

# print summary of the data
summary(crime)
```

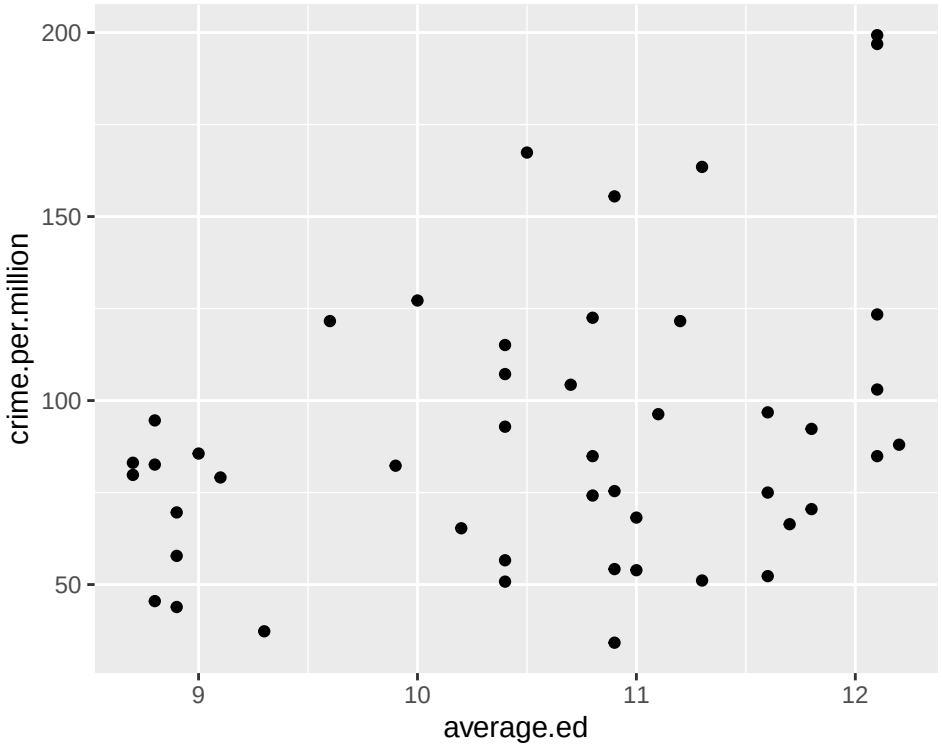
```
##  crime.per.million  young.males    is.south   average.ed
##  Min.   : 34.20    Min.   :119.0   0:31     Min.   : 8.70
##  1st Qu.: 65.85    1st Qu.:130.0   1:16     1st Qu.: 9.75
##  Median : 83.10    Median :136.0            Median :10.80
##  Mean   : 90.51    Mean   :138.6            Mean   :10.56
##  3rd Qu.:105.75    3rd Qu.:146.0            3rd Qu.:11.45
##  Max.   :199.30    Max.   :177.0            Max.   :12.20
##  exp.per.cap.1960 exp.per.cap.1959  labour.part     male.per.fem
##  Min.   : 45.0    Min.   : 41.00   Min.   :480.0   Min.   : 934.0
##  1st Qu.: 62.5    1st Qu.: 58.50   1st Qu.:530.5   1st Qu.: 964.5
##  Median : 78.0    Median : 73.00   Median :560.0   Median : 977.0
##  Mean   : 85.0    Mean   : 80.23   Mean   :561.2   Mean   : 983.0
##  3rd Qu.:104.5    3rd Qu.: 97.00   3rd Qu.:593.0   3rd Qu.: 992.0
##  Max.   :166.0    Max.   :157.00   Max.   :641.0   Max.   :1071.0
##    population        nonwhite      unemp.youth      unemp.adult
##  Min.   :  3.00   Min.   :  2.0   Min.   : 70.00   Min.   :20.00
##  1st Qu.: 10.00   1st Qu.: 24.0   1st Qu.: 80.50   1st Qu.:27.50
##  Median : 25.00   Median : 76.0   Median : 92.00   Median :34.00
##  Mean   : 36.62   Mean   :101.1   Mean   : 95.47   Mean   :33.98
##  3rd Qu.: 41.50   3rd Qu.:132.5   3rd Qu.:104.00   3rd Qu.:38.50
##  Max.   :168.00   Max.   :423.0   Max.   :142.00   Max.   :58.00
##  median.assets   num.low.salary
##  Min.   :2.880   Min.   :126.0
```

```
##  1st Qu.:4.595   1st Qu.:165.5
##  Median :5.370   Median :176.0
##  Mean   :5.254   Mean   :194.0
##  3rd Qu.:5.915   3rd Qu.:227.5
##  Max.   :6.890   Max.   :276.0
```

First step: some plotting and summary statistics

You can start by feeding everything into a regression, but it's often a better idea to construct some simple plots (e.g., scatterplots and boxplots) and summary statistics to get some sense of how the data behaves.

```
# Scatter plot of outcome (crime.per.million) against average.ed
qplot(average.ed, crime.per.million, data = crime)
```

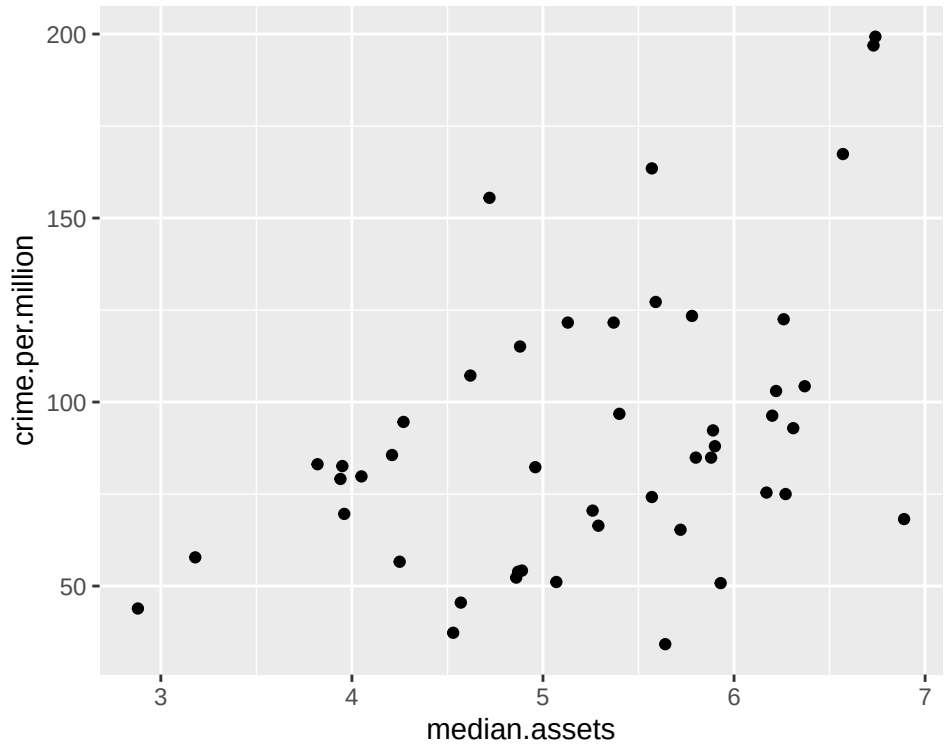



```
# correlation between education and crime
with(crime, cor(average.ed, crime.per.million))
```

```
## [1] 0.3228349
```

This seems to suggest that higher levels of average education are associated with higher crime rates. *Can you come up with an explanation for this phenomenon?*

```
# Scatter plot of outcome (crime.per.million) against median.assets
qplot(median.assets, crime.per.million, data = crime)
```

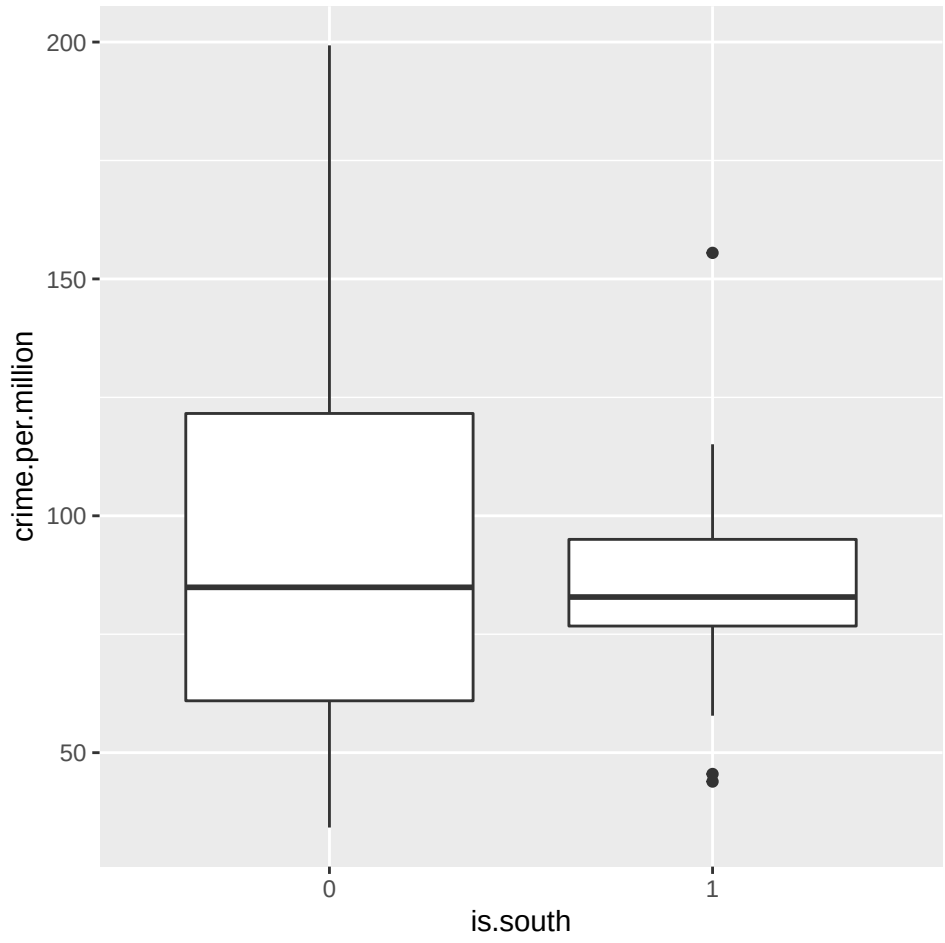



```
# correlation between education and crime
with(crime, cor(median.assets, crime.per.million))
```

```
## [1] 0.4413199
```

There also appears to be a positive association between median assets and crime rates.

```
# Boxplots showing crime rate broken down by southern vs non-southern state
qplot(is.south, crime.per.million, geom = "boxplot", data = crime)
```



Constructing a regression model

To construct a linear regression model in R, we use the `lm()` function. You can specify the regression model in various ways. The simplest is often to use the formula specification.

The first model we fit is a regression of the outcome (`crimes.per.million`) against all the other variables in the data set. You can either white out all the variable names. or use the shorthand `y ~ .` to specify that you want to include all the variables in your regression.

```
crime.lm <- lm(crime.per.million ~ ., data = crime)
# Summary of the linear regression model
crime.lm
```

```
##
## Call:
## lm(formula = crime.per.million ~ ., data = crime)
##
## Coefficients:
##      (Intercept)       young.males         is.south1       average.ed
##       -6.918e+02         1.040e+00        -8.308e+00         1.802e+01
## exp.per.cap.1960  exp.per.cap.1959       labour.part      male.per.fem
##        1.608e+00        -6.673e-01        -4.103e-02         1.648e-01
##       population          nonwhite       unemp.youth       unemp.adult
##       -4.128e-02         7.175e-03        -6.017e-01         1.792e+00
##    median.assets    num.low.salary
```

```
##        1.374e+01         7.929e-01
```

```
summary(crime.lm)
```

```
##
## Call:
## lm(formula = crime.per.million ~ ., data = crime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.884 -11.923  -1.135  13.495  50.560
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -6.918e+02  1.559e+02  -4.438 9.56e-05 ***
## young.males       1.040e+00  4.227e-01   2.460  0.01931 *
## is.south1        -8.308e+00  1.491e+01  -0.557  0.58117
## average.ed        1.802e+01  6.497e+00   2.773  0.00906 **
## exp.per.cap.1960  1.608e+00  1.059e+00   1.519  0.13836
## exp.per.cap.1959 -6.673e-01  1.149e+00  -0.581  0.56529
## labour.part      -4.103e-02  1.535e-01  -0.267  0.79087
## male.per.fem      1.648e-01  2.099e-01   0.785  0.43806
## population       -4.128e-02  1.295e-01  -0.319  0.75196
## nonwhite          7.175e-03  6.387e-02   0.112  0.91124
## unemp.youth      -6.017e-01  4.372e-01  -1.376  0.17798
## unemp.adult       1.792e+00  8.561e-01   2.093  0.04407 *
## median.assets     1.374e+01  1.058e+01   1.298  0.20332
## num.low.salary    7.929e-01  2.351e-01   3.373  0.00191 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.94 on 33 degrees of freedom
## Multiple R-squared:  0.7692, Adjusted R-squared:  0.6783
## F-statistic: 8.462 on 13 and 33 DF,  p-value: 3.686e-07
```

R's default is to output values in scientific notation. This can make it hard to interpret the numbers. Here's some code that can be used to force full printout of numbers.

```
options(scipen=4)  # Set scipen = 0 to get back to default
```

```
summary(crime.lm)
```

```
##
## Call:
## lm(formula = crime.per.million ~ ., data = crime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.884 -11.923  -1.135  13.495  50.560
##
## Coefficients:
##                     Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)      -691.837588  155.887918  -4.438 0.0000956 ***
## young.males         1.039810    0.422708   2.460   0.01931 *
## is.south1          -8.308313   14.911588  -0.557   0.58117
## average.ed         18.016011    6.496504   2.773   0.00906 **
```

```
## exp.per.cap.1960    1.607818    1.058667   1.519   0.13836
## exp.per.cap.1959   -0.667258    1.148773  -0.581   0.56529
## labour.part        -0.041031    0.153477  -0.267   0.79087
## male.per.fem        0.164795    0.209932   0.785   0.43806
## population         -0.041277    0.129516  -0.319   0.75196
## nonwhite            0.007175    0.063867   0.112   0.91124
## unemp.youth        -0.601675    0.437154  -1.376   0.17798
## unemp.adult         1.792263    0.856111   2.093   0.04407 *
## median.assets      13.735847   10.583028   1.298   0.20332
## num.low.salary      0.792933    0.235085   3.373   0.00191 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.94 on 33 degrees of freedom
## Multiple R-squared:  0.7692, Adjusted R-squared:  0.6783
## F-statistic: 8.462 on 13 and 33 DF,  p-value: 0.0000003686
```

```
# Nicer print-out
kable(summary(crime.lm)$coef, digits = c(3, 3, 3, 4))
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-691.838	155.888	-4.438	0.0001
young.males	1.040	0.423	2.460	0.0193
is.south1	-8.308	14.912	-0.557	0.5812
average.ed	18.016	6.497	2.773	0.0091
exp.per.cap.1960	1.608	1.059	1.519	0.1384
exp.per.cap.1959	-0.667	1.149	-0.581	0.5653
labour.part	-0.041	0.153	-0.267	0.7909
male.per.fem	0.165	0.210	0.785	0.4381
population	-0.041	0.130	-0.319	0.7520
nonwhite	0.007	0.064	0.112	0.9112
unemp.youth	-0.602	0.437	-1.376	0.1780
unemp.adult	1.792	0.856	2.093	0.0441
median.assets	13.736	10.583	1.298	0.2033
num.low.salary	0.793	0.235	3.373	0.0019

Looking at the p-values, it looks like `num.low.salary` (number of families per 1000 earning below 1/2 the median income), `unemp.adult` (Unemployment rate of urban males per 1000 of age 35-39), `average.ed` (Mean # of years of schooling 25 or older), and `young.males` (number of males of age 14-24 per 1000 population) are all statistically significant predictors of crime rate.

The coefficients for these predictors are all positive, so crime rates are positively associated with wealth distribution, adult unemployment rates, average education levels, and high rates of young males in the population.

Exploring the lm object

What kind of output do we get when we run a linear model (`lm`) in R?

```
# List all attributes of the linear model
attributes(crime.lm)
```

```
## $names
##  [1] "coefficients"  "residuals"     "effects"       "rank"
```

```
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "contrasts"     "xlevels"       "call"          "terms"
## [13] "model"
##
## $class
## [1] "lm"
```

```
# coefficients
crime.lm$coef
```

```
##      (Intercept)      young.males        is.south1       average.ed
##   -691.837587905      1.039809653     -8.308312889     18.016010601
## exp.per.cap.1960 exp.per.cap.1959      labour.part     male.per.fem
##      1.607818377     -0.667258285     -0.041031047      0.164794968
##       population         nonwhite      unemp.youth      unemp.adult
##     -0.041276887      0.007174688     -0.601675298      1.792262901
##    median.assets   num.low.salary
##     13.735847285      0.792932786
```

None of the attributes seem to give you p-values. Here's what you can do to get a table that allows you to extract p-values.

```
# Pull coefficients element from summary(lm) object
round(summary(crime.lm)$coef, 3)
```

```
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -691.838    155.888  -4.438    0.000
## young.males         1.040      0.423   2.460    0.019
## is.south1          -8.308     14.912  -0.557    0.581
## average.ed         18.016      6.497   2.773    0.009
## exp.per.cap.1960    1.608      1.059   1.519    0.138
## exp.per.cap.1959   -0.667      1.149  -0.581    0.565
## labour.part        -0.041      0.153  -0.267    0.791
## male.per.fem        0.165      0.210   0.785    0.438
## population         -0.041      0.130  -0.319    0.752
## nonwhite            0.007      0.064   0.112    0.911
## unemp.youth        -0.602      0.437  -1.376    0.178
## unemp.adult         1.792      0.856   2.093    0.044
## median.assets      13.736     10.583   1.298    0.203
## num.low.salary      0.793      0.235   3.373    0.002
```

If you want a particular p-value, you can get it by doing the following

```
# Pull the coefficients table from summary(lm)
crime.lm.coef <- round(summary(crime.lm)$coef, 3)
# See what this gives
class(crime.lm.coef)
```

```
## [1] "matrix"
```

```
attributes(crime.lm.coef)
```

```
## $dim
## [1] 14  4
##
## $dimnames
## $dimnames[[1]]
##  [1] "(Intercept)"      "young.males"      "is.south1"
```

```
##  [4] "average.ed"       "exp.per.cap.1960" "exp.per.cap.1959"
##  [7] "labour.part"      "male.per.fem"     "population"
## [10] "nonwhite"         "unemp.youth"      "unemp.adult"
## [13] "median.assets"    "num.low.salary"
##
## $dimnames[[2]]
## [1] "Estimate"   "Std. Error" "t value"    "Pr(>|t|)"
```

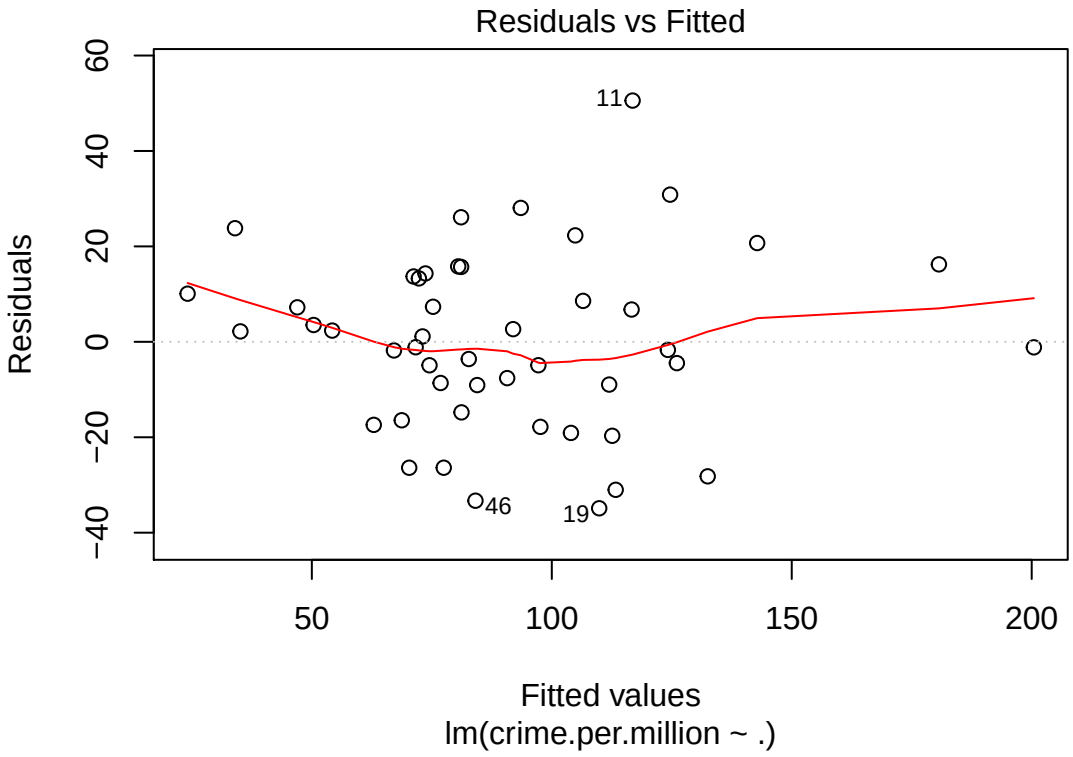
```
crime.lm.coef["average.ed","Pr(>|t|)"]
```

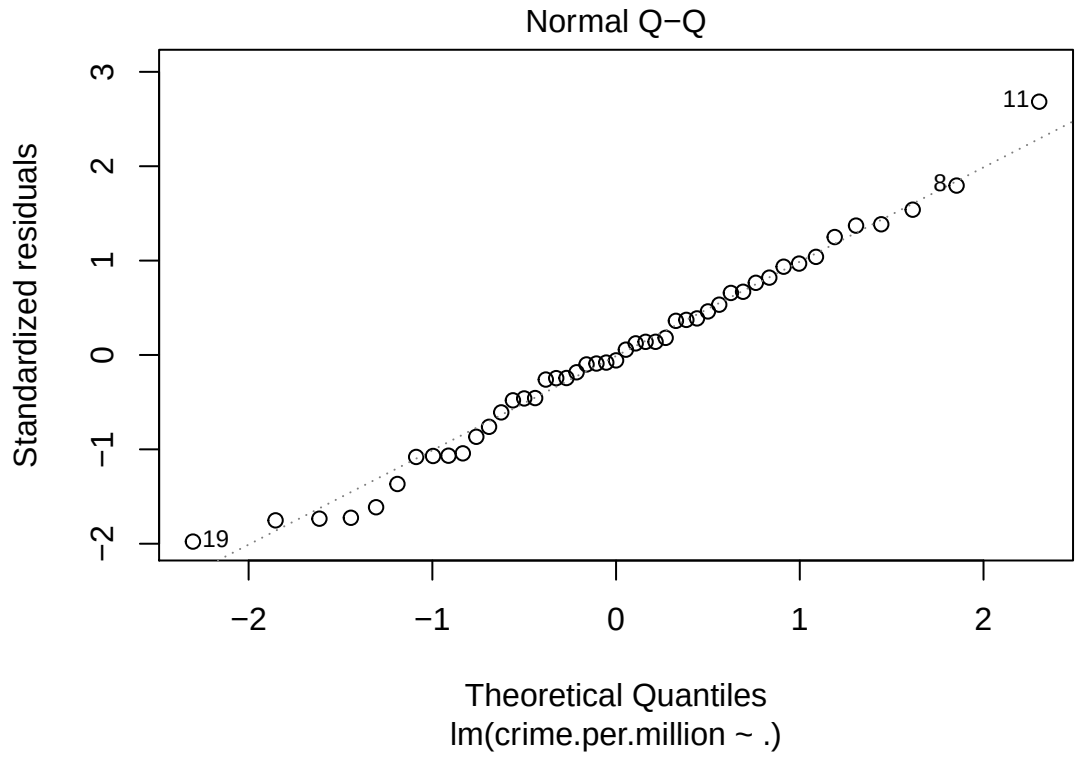
```
## [1] 0.009
```

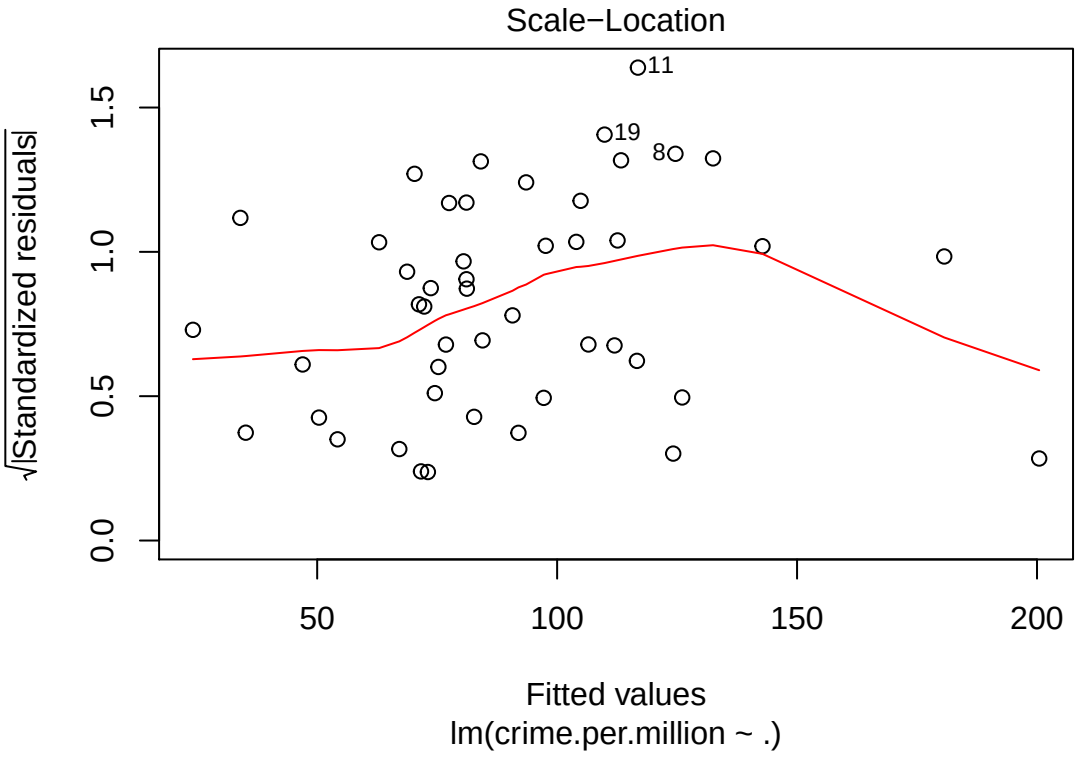
The coefficients table is a matrix with named rows and columns. You can therefore access particular cells either by numeric index, or by name (as in the example above).

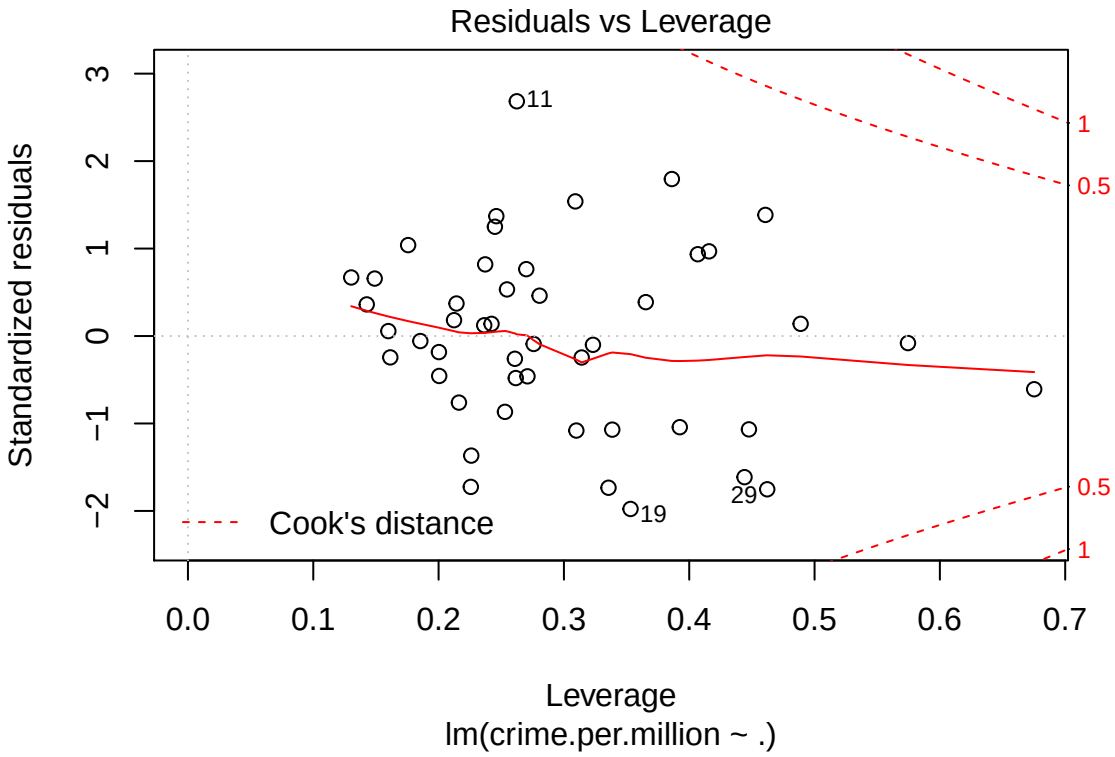
Plotting the lm object

```
plot(crime.lm)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(crime.per.million ~ .)
11
8
19


Scale−Location
√|Standardized residuals|
Fitted values
lm(crime.per.million ~ .)
11
19
8



These four plots are important diagnostic tools in assessing whether the linear model is appropriate. The first two plots are the most important, but the last two can also help with identifying outliers and non-linearities.

Residuals vs. Fitted When a linear model is appropriate, we expect

1. the residuals will have constant variance when plotted against fitted values; and
2. the residuals and fitted values will be uncorrelated.

If there are clear trends in the residual plot, or the plot looks like a funnel, these are clear indicators that the given linear model is inappropriate.

Normal QQ plot You can use a linear model for prediction even if the underlying normality assumptions don't hold. However, in order for the p-values to be believable, the residuals from the regression must look approximately normally distributed.

Scale-location plot This is another version of the residuals vs fitted plot. There should be no discernible trends in this plot.

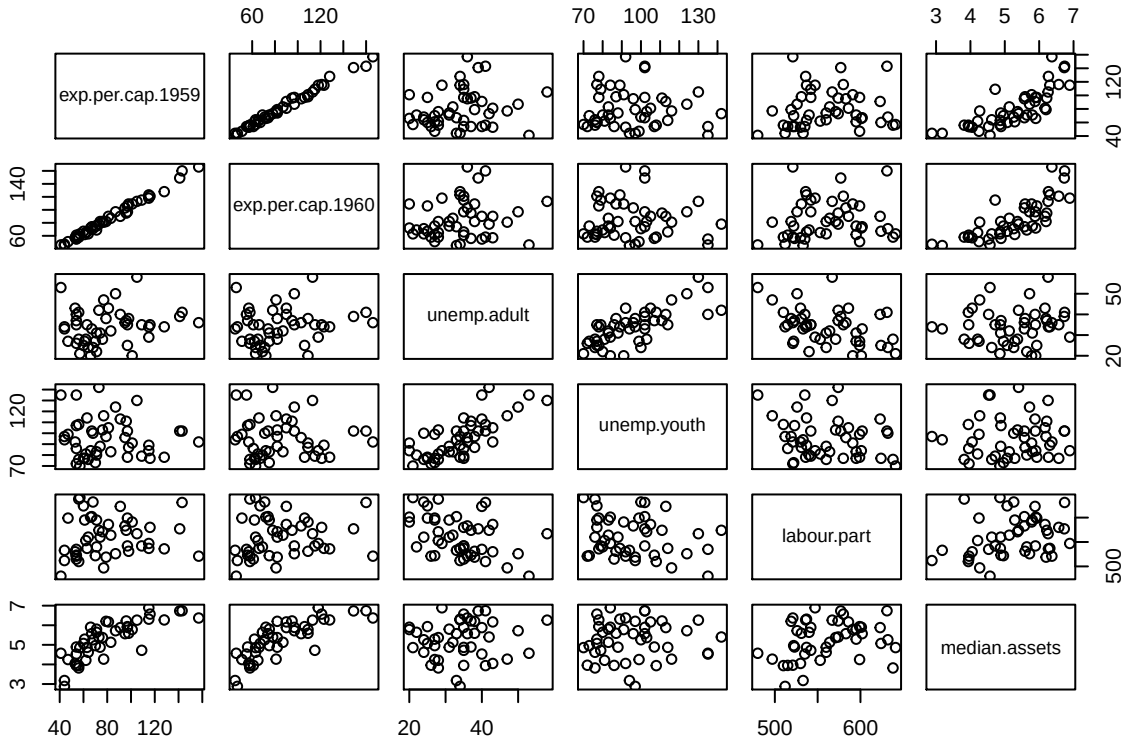
Residuals vs Leverage. Leverage is a measure of how much an observation influenced the model fit. It's a one-number summary of how different the model fit would be if the given observation was excluded, compared to the model fit where the observation is included. Points with *high residual* (poorly described by the model) and *high leverage* (high influence on model fit) are outliers. They're skewing the model fit away from the rest of the data, and don't really seem to fit with the rest of the data.

Collinearity and pairs plots

In your regression class you probably learned that **collinearity** can throw off the coefficient estimates. To diagnose collinearity, we can do a plot matrix. In base graphics, this can be accomplished via the `pairs` function.

As a demo, let's look at some of the economic indicators in our data set.

```
economic.var.names <- c("exp.per.cap.1959", "exp.per.cap.1960", "unemp.adult", "unemp.youth", "labour.pa
pairs(crime[,economic.var.names])
```



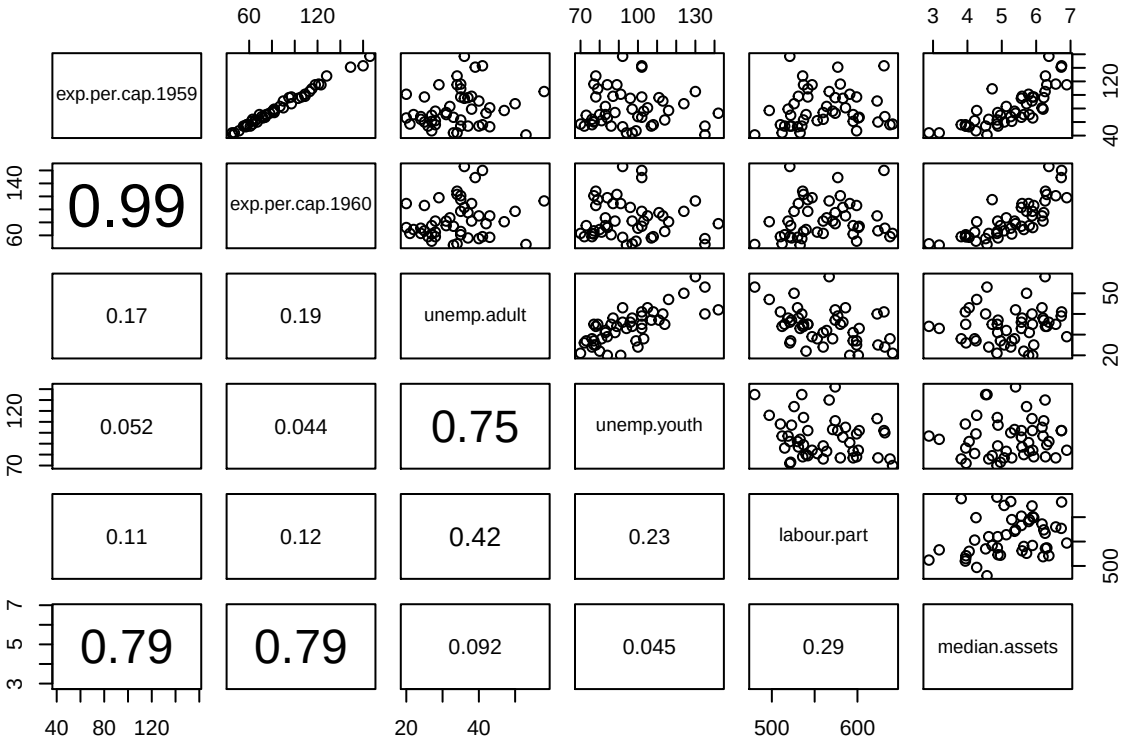

```
round(cor(crime[,economic.var.names]), 3)
```

```
##                  exp.per.cap.1959 exp.per.cap.1960 unemp.adult unemp.youth
## exp.per.cap.1959            1.000            0.994       0.169      -0.052
## exp.per.cap.1960            0.994            1.000       0.185      -0.044
## unemp.adult                 0.169            0.185       1.000       0.746
## unemp.youth                -0.052           -0.044       0.746       1.000
## labour.part                 0.106            0.121      -0.421      -0.229
## median.assets               0.794            0.787       0.092       0.045
##                  labour.part median.assets
## exp.per.cap.1959       0.106         0.794
## exp.per.cap.1960       0.121         0.787
## unemp.adult           -0.421         0.092
## unemp.youth           -0.229         0.045
## labour.part            1.000         0.295
## median.assets          0.295         1.000
```

Since the above-diagonal and below-diagonal plots contain essentially the same information, it's often more useful to display some other values in one of the spaces. In the example below, we use the panel.cor function from the `pairs()` documentation to add text below the diagonal.

```
# Function taken from ?pairs Example section.
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789), digits = digits)[1]
```

```
    txt <- paste0(prefix, txt)
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = pmax(1, cex.cor * r))
}

# Use panel.cor to display correlations in lower panel.
pairs(crime[,economic.var.names], lower.panel = panel.cor)
```



Looking at the plot, we see that many of the variables are very strongly correlated. In particular, police expenditures are pretty much identical in 1959 and 1960. This is an extreme case of collinearity. Also, unsurprisingly, youth unemployment and adult unemployment are also highly correlated.

Let's just include the 1960 police expenditure variable, and also drop the young unemployment variable. We'll do this using the `update()` function. Here's what happens.

```
crime.lm.2 <- update(crime.lm, . ~ . - exp.per.cap.1959 - unemp.youth)
summary(crime.lm.2)
```

```
##
## Call:
## lm(formula = crime.per.million ~ young.males + is.south + average.ed +
##     exp.per.cap.1960 + labour.part + male.per.fem + population +
##     nonwhite + unemp.adult + median.assets + num.low.salary,
##     data = crime)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -35.82 -11.57  -1.51  10.63  55.02
##
## Coefficients:
##                     Estimate  Std. Error t value  Pr(>|t|)
```

```
## (Intercept)      -633.438828  145.470340  -4.354  0.000111 ***
## young.males         1.126883    0.418791   2.691  0.010853 *
## is.south1          -0.556600   13.883248  -0.040  0.968248
## average.ed         15.328028    6.202516   2.471  0.018476 *
## exp.per.cap.1960    1.138299    0.226977   5.015 0.0000153 ***
## labour.part         0.068716    0.133540   0.515  0.610087
## male.per.fem        0.003021    0.173041   0.017  0.986172
## population         -0.064477    0.128278  -0.503  0.618367
## nonwhite           -0.013794    0.061901  -0.223  0.824960
## unemp.adult         0.931498    0.541803   1.719  0.094402 .
## median.assets      15.158975   10.524458   1.440  0.158653
## num.low.salary      0.825936    0.234189   3.527  0.001197 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.98 on 35 degrees of freedom
## Multiple R-squared:  0.7543, Adjusted R-squared:  0.6771
## F-statistic: 9.769 on 11 and 35 DF,  p-value: 0.00000009378
```

```
crime.lm.summary.2 <- summary(crime.lm.2)
```

When outputting regression results, it's always good to use the `kable()` function to make things look a little nicer.

```
kable(round(crime.lm.summary.2$coef, 3), format = 'markdown')
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-633.439	145.470	-4.354	0.000
young.males	1.127	0.419	2.691	0.011
is.south1	-0.557	13.883	-0.040	0.968
average.ed	15.328	6.203	2.471	0.018
exp.per.cap.1960	1.138	0.227	5.015	0.000
labour.part	0.069	0.134	0.515	0.610
male.per.fem	0.003	0.173	0.017	0.986
population	-0.064	0.128	-0.503	0.618
nonwhite	-0.014	0.062	-0.223	0.825
unemp.adult	0.931	0.542	1.719	0.094
median.assets	15.159	10.524	1.440	0.159
num.low.salary	0.826	0.234	3.527	0.001

Thinking more critically about the linear model

We see that `exp.per.cap.1960` is now highly significant.

```
crime.lm.summary.2$coef["exp.per.cap.1960",]
```

```
##      Estimate    Std. Error       t value      Pr(>|t|)
## 1.13829907170 0.22697675756 5.01504684417 0.00001532994
```

This is interesting. It's essentially saying that, all else being equal, every dollar per capita increase in police expenditure is on average associated with an increase in crime of 1.13 per million population.

```
crime.lm.summary.2$coef["average.ed",]
```

```
##    Estimate  Std. Error     t value    Pr(>|t|)
## 15.32802778  6.20251646  2.47125951  0.01847635
```

Also, for every unit increase in average education, we find that the number of reported crimes increases by about 15.3 per million.

One of my main reasons for selecting this data set is that it illustrates some of the more common pitfalls in interpreting regression models.

Just because a coefficient is significant, doesn't mean your covariate causes your response

- This is the old adage that correlation does not imply causation. In this example, we have strong evidence that higher police expenditures are positively associated with crime rates. This doesn't mean that decreasing police expenditure will lower crime rate. The relationship is not causal – at least not in that direction. A more reasonable explanation is that higher crime rates promt policy makers to increase police expenditure.

There's a difference between practical significance and statistical significance

- Both `average.ed` and `exp.per.cap.1960` are statistically significant. `exp.per.cap.1960` has a much more significant p-value, but also a much smaller coefficient. When looking at your regression model, you shouldn't just look at the p-value column. The really interesting covariates are the ones that are significant, but also have the largest effect.

Factors in linear regression

Interpreting coefficients of factor variables

For categorical variables, the interpretation is relative to the given baseline. To understand what this means, let's go back to the birthwt data and try regressing birthweight on race, mother's smoking status, and mother's age.

```
# Fit regression model
birthwt.lm <- lm(birthwt.grams ~ race + mother.smokes + mother.age, data = birthwt)

# Regression model summary
summary(birthwt.lm)
```

```
##
## Call:
## lm(formula = birthwt.grams ~ race + mother.smokes + mother.age,
##     data = birthwt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2322.6  -447.3    28.4   502.2  1612.3
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2837.604    257.573  11.017  < 2e-16 ***
## raceother          -3.789    161.115  -0.024 0.981264
## racewhite         444.069    156.194   2.843 0.004973 **
## mother.smokesyes -426.093    109.988  -3.874 0.000149 ***
## mother.age          2.134      9.771   0.218 0.827326
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 690 on 184 degrees of freedom
## Multiple R-squared:  0.1236, Adjusted R-squared:  0.1046
## F-statistic:  6.49 on 4 and 184 DF,  p-value: 0.00006592
```

Observe that while there are 3 levels in the `race` variable, there are only two coefficents estimated: one called `raceother` and the other called `racewhite`.

Why is one of the levels missing in the regression?: These coefficients represent difference from the **baseline** level. The baseline level is the one coded as `1`. By default, it's the level that comes first in the alphabet (here, `black`). The first level is essentially pulled into the intercept term, so we don't see it explicitly.

Interpretation of the coefficients: The baseline level for `race` is `black`. Thus we see that, once we control for mother's smoking status and age, babies of white mothers tend to weigh on average 444.1g more than those of black mothers. Babies whose mothers are non-white and non-black on average weigh 3.8g less than those of black mothers.

Note the numbers in the above paragraph come from inline code chunks. Here's the syntax for grabbing & rounding the coefficients of race.

```
# white
round(coef(birthwt.lm)["racewhite"], 1)
```

```
## racewhite
##     444.1
```

```
# other
round(coef(birthwt.lm)["raceother"], 1)
```

```
## raceother
##      -3.8
```

Assessing significance of factors in regression

When dealing with multi-level factors, significance is assessed by considering dropping the entire factor from the regression. You can therefore wind up in a situation where a factor is significant even when none of the individual estimated coefficients are on their own significant.

To run this kind of test, we'll use the `anova()` function and specify two models, one of which is nested in the other.

Here's how we can test whether `race` is a significant predictor of birth weight even after we control for mother's age and smoking status.

```
# Check if including the race variable significantly improves model fit
anova(update(birthwt.lm, . ~ . - race), birthwt.lm, test = "Chisq")
```

```
## Analysis of Variance Table
##
## Model 1: birthwt.grams ~ mother.smokes + mother.age
## Model 2: birthwt.grams ~ race + mother.smokes + mother.age
##   Res.Df      RSS Df Sum of Sq  Pr(>Chi)
## 1    186 95672288
## 2    184 87608636  2   8063652 0.0002101 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This returns a p-value based on a Chi-square variable with degrees of freedom = (# levels in factor - 1)

The test is highly significant, so race is indeed a significant predictor of birthweight.