Basic R Materials for 500

Thomas E. Love

2018-01-06

Contents

> It is an odd feeling when you love what you do and everyone else seems to hate it. I get to peer into lists of numbers and tease out knowledge that can help people live longer, healthier lives. But if I tell friends I get a kick out of statistics, they inch away as if I have a communicable disease.
>
> – Andrew Vickers *What is a P Value, Anyway?*

1 Some Opening Thoughts

My goals in this document are to help catalyze your efforts towards . . .

1. Applying statistical methods in evaluating clinical or public health interventions without the use of a , emphasizing activities that might be plausible in a real research project
2. Using the R statistical programming language (free at cran.case.edu) and the R Studio interface (free at rstudio.com) and R Markdown to obtain statistical results for comparison and simple modeling given some data.

This material provides some insight into. . .

- Gathering, managing and describing data
- How to think about collecting some data
- How to get data into the infernal machine
- How to get some useful graphs/other stuff out of it
- How to fit multiple regression and logistic regression models in R

In fact, though, statistical thinking is about a lot more than this. At the very least, it's about

- planning the study,
- collecting then cleaning the data,
- analyzing the results,
- interpreting the analyses and
- presenting the study.

Statistics is far too important to be left to statisticians!

2 Getting R and R Studio onto your computer

See the Notes section of our web site for some detailed instructions on getting R and R Studio onto your computer, as well as a separate document with some tips on using R Studio, and in particular, R Markdown.

3 Getting Data into R from Excel or another Software Package: The Fundamentals

The easiest way to get data from another software package into R is to save the file (from within the other software package) in a form that R can read. What you want is to end up with an Excel file that looks like this. . .

The variable names are in the first row, and the data are in the remaining rows (2-10 in this small example). Categorical variables are most easily indicated by letters (drug A or B, for

	A	B	C	D
1	Patient	Drug	Gender	Response
2	MW	A	M	23
3	TT	B	F	15
4	KH	B	M	18
5	GC	A	M	29
6	DS	B	F	34
7	HJ	B	F	15
8	KM	A	M	7
9	RS	A	M	19
10	DG	A	F	22

Figure 1: sheet1.png

	A	B	C	D	E	F	G	H	I	J	K	L
1	pt.id	insurance	a1c	ldl	sbp	eyexm	pnvax	age	bmi	raceeth	female	smoking
2	1	Medicaid	6.1	124	160	no	no	66	46.9	Black	female	nonsmoker
3	2	Commercial	6.9	187	162	yes	yes	57	43	White	female	nonsmoker
4	3	Uninsured	8.9	113	158	no	no	54	37.3	White	female	nonsmoker
5	4	Uninsured	7.7	64	140	no	no	49	40.9	Black	female	nonsmoker
6	5	Uninsured	11	133	153	no	yes	52	32.2	Black	female	nonsmoker
7	6	Uninsured	9.6	156	100	no	no	39	39.8	White	female	nonsmoker
8	7	Uninsured	6.2	162	114	no	yes	51	36	Black	female	smoker
9	8	Uninsured	7.2	112	150	yes	no	51	40.2	Black	female	nonsmoker
10	9	Commercial	7.1	88	124	no	yes	68	28.3	White	female	smoker
11	10	Commercial	5.2	142	132	no	no	62	28.3	Black	female	nonsmoker

Figure 2: dm401-first10.png

instance) while continuous variables, like response, are indicated by numbers. Leave missing cells blank or use the symbol `NA`, rather than indicating them with, say, -99.

Within Excel, this file can be saved as a `.csv` (comma-separated text file) or just as an Excel .XLS file, and then imported directly into R, via RStudio by clicking Import Dataset under the Workspace tab, then selecting From Text File. If you've saved the file in Excel as a `.csv` file, RStudio will generally make correct guesses about how to import the file. Once imported, you just need to save the workspace when you quit RStudio and you'll avoid the need to re-import.

4 Describing a Diabetes Pilot Study

Consider the `dm401` data set, which provides (hypothetical) pilot demographic and clinical information for 146 continuity diabetic patients in a large metropolitan health system. The `dm401.csv` file's first ten observations are shown below.

```r
dm401 <- read.csv("dm401.csv")
head(dm401, 10) ## shows first ten observations
```

```
   pt.id  insurance  a1c ldl sbp eyexm pnvax age   bmi raceeth female
1      1   Medicaid  6.1 124 160    no    no  66 46.92   Black female
2      2 Commercial  6.9 187 162   yes   yes  57 43.00   White female
3      3  Uninsured  8.9 113 158    no    no  54 37.30   White female
4      4  Uninsured  7.7  64 140    no    no  49 40.90   Black female
5      5  Uninsured 11.3 133 153    no   yes  52 32.19   Black female
6      6  Uninsured  9.6 156 100    no    no  39 39.76   White female
7      7  Uninsured  6.2 162 114    no   yes  51 35.99   Black female
8      8  Uninsured  7.2 112 150   yes    no  51 40.16   Black female
9      9 Commercial  7.1  88 124    no   yes  68 28.32   White female
10    10 Commercial  5.2 142 132    no    no  62 28.26   Black female
     smoking
1  nonsmoker
2  nonsmoker
3  nonsmoker
4  nonsmoker
5  nonsmoker
6  nonsmoker
7     smoker
8  nonsmoker
9     smoker
10 nonsmoker
```

```r
summary(dm401)
```

```
     pt.id             insurance       a1c              ldl
 Min.   :  1.00   Commercial:39   Min.   : 4.700   Min.   : 16
 1st Qu.: 37.25   Medicaid  :25   1st Qu.: 6.125   1st Qu.: 91
 Median : 73.50   Medicare  :52   Median : 7.100   Median :113
 Mean   : 73.50   Uninsured :30   Mean   : 7.677   Mean   :118
 3rd Qu.:109.75                   3rd Qu.: 8.400   3rd Qu.:141
 Max.   :146.00                   Max.   :15.400   Max.   :218
      sbp          eyexm      pnvax          age             bmi
 Min.   : 84.0   no :105   no :56    Min.   :23.0   Min.   :16.62
 1st Qu.:120.0   yes: 41   yes:90    1st Qu.:48.0   1st Qu.:28.48
 Median :131.0                       Median :57.0   Median :33.44
 Mean   :135.4                       Mean   :57.4   Mean   :34.13
 3rd Qu.:149.5                       3rd Qu.:67.0   3rd Qu.:38.71
 Max.   :213.0                       Max.   :93.0   Max.   :65.77
     raceeth       female          smoking
 Black   :72   female:82   nonsmoker:106
 Hispanic:10   male  :64   smoker   : 40
```

```
 White    :64
```

4.0.1 A Bare Bones Data Dictionary

All measures are as of the date of study entry. We have:

- insurance `payer` in four categories
- level of hemoglobin `a1c`
- `ldl` cholesterol
- `sbp` is systolic blood pressure
- `pnvax` indicates a recorded pneumococcal vaccine at any time prior to study entry
- `age` is in years
- `bmi` is body mass index
- `raceeth` is race/ethnicity in three categories
- `female` indicates gender
- `smoking` status (self-report of non-smoker or current smoker at study entry)
- `eyexm` indicates whether an eye examination is recorded in the past 12 months.

```r
str(dm401)
```

```
'data.frame':   146 obs. of  12 variables:
 $ pt.id    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ insurance: Factor w/ 4 levels "Commercial","Medicaid",..: 2 1 4 4 4 4 4 4 1 1 ...
 $ a1c      : num  6.1 6.9 8.9 7.7 11.3 9.6 6.2 7.2 7.1 5.2 ...
 $ ldl      : int  124 187 113 64 133 156 162 112 88 142 ...
 $ sbp      : int  160 162 158 140 153 100 114 150 124 132 ...
 $ eyexm    : Factor w/ 2 levels "no","yes": 1 2 1 1 1 1 1 2 1 1 ...
 $ pnvax    : Factor w/ 2 levels "no","yes": 1 2 1 1 2 1 2 1 2 1 ...
 $ age      : int  66 57 54 49 52 39 51 51 68 62 ...
 $ bmi      : num  46.9 43 37.3 40.9 32.2 ...
 $ raceeth  : Factor w/ 3 levels "Black","Hispanic",..: 1 3 3 1 1 3 1 1 3 1 ...
 $ female   : Factor w/ 2 levels "female","male": 1 1 1 1 1 1 1 1 1 1 ...
 $ smoking  : Factor w/ 2 levels "nonsmoker","smoker": 1 1 1 1 1 1 2 1 2 1 ...
```

4.1 Task 1: Cleaning the Data

We'll begin with some elementary cleaning. Is there any missingness in the data? Do we have any unrealistic values in the data elements? Do range checks pan out?

```r
library(Hmisc)
describe(dm401)
```

```
dm401

 12  Variables      146  Observations
---------------------------------------------------------------------------
pt.id
       n  missing distinct     Info     Mean      Gmd      .05      .10
     146        0      146        1     73.5       49     8.25    15.50
     .25      .50      .75      .90      .95
   37.25    73.50   109.75   131.50   138.75

lowest :   1   2   3   4   5, highest: 142 143 144 145 146
---------------------------------------------------------------------------
insurance
       n  missing distinct
     146        0        4

Value      Commercial   Medicaid   Medicare  Uninsured
Frequency          39         25         52         30
Proportion      0.267      0.171      0.356      0.205
---------------------------------------------------------------------------
a1c
       n  missing distinct     Info     Mean      Gmd      .05      .10
     146        0       62    0.999    7.677     2.28    5.400    5.550
     .25      .50      .75      .90      .95
   6.125    7.100    8.400   11.000   11.900

lowest :  4.7  4.8  5.2  5.3  5.4, highest: 13.0 13.7 14.0 14.4 15.4
---------------------------------------------------------------------------
ldl
       n  missing distinct     Info     Mean      Gmd      .05      .10
     146        0       83        1      118     42.5     64.5     77.0
     .25      .50      .75      .90      .95
    91.0    113.0    141.0    170.0    186.5

lowest :  16  32  39  51  58, highest: 192 210 211 215 218
---------------------------------------------------------------------------
sbp
       n  missing distinct     Info     Mean      Gmd      .05      .10
     146        0       64    0.999    135.4     25.4    102.0    110.0
     .25      .50      .75      .90      .95
   120.0    131.0    149.5    163.5    174.2

lowest :  84  88  98 100 102, highest: 184 185 186 202 213
---------------------------------------------------------------------------
eyexm
```

```
       n  missing distinct 
     146        0        2 

Value         no   yes
Frequency    105    41
Proportion 0.719 0.281
---------------------------------------------------------------------------
pnvax 
       n  missing distinct 
     146        0        2 

Value         no   yes
Frequency     56    90
Proportion 0.384 0.616
---------------------------------------------------------------------------
age 
       n  missing distinct     Info     Mean      Gmd      .05      .10 
     146        0       54    0.999     57.4    16.72    33.00    36.00 
     .25      .50      .75      .90      .95 
   48.00    57.00    67.00    76.00    81.75 

lowest : 23 24 26 27 28, highest: 82 83 84 88 93
---------------------------------------------------------------------------
bmi 
       n  missing distinct     Info     Mean      Gmd      .05      .10 
     146        0      141        1    34.13    8.557    22.96    25.65 
     .25      .50      .75      .90      .95 
   28.48    33.44    38.71    43.53    47.16 

lowest : 16.62 16.92 20.55 21.32 21.33, highest: 50.13 51.06 53.24 58.14 65.77
---------------------------------------------------------------------------
raceeth 
       n  missing distinct 
     146        0        3 

Value         Black Hispanic    White
Frequency        72       10       64
Proportion    0.493    0.068    0.438
---------------------------------------------------------------------------
female 
       n  missing distinct 
     146        0        2 

Value      female   male
Frequency      82     64
```

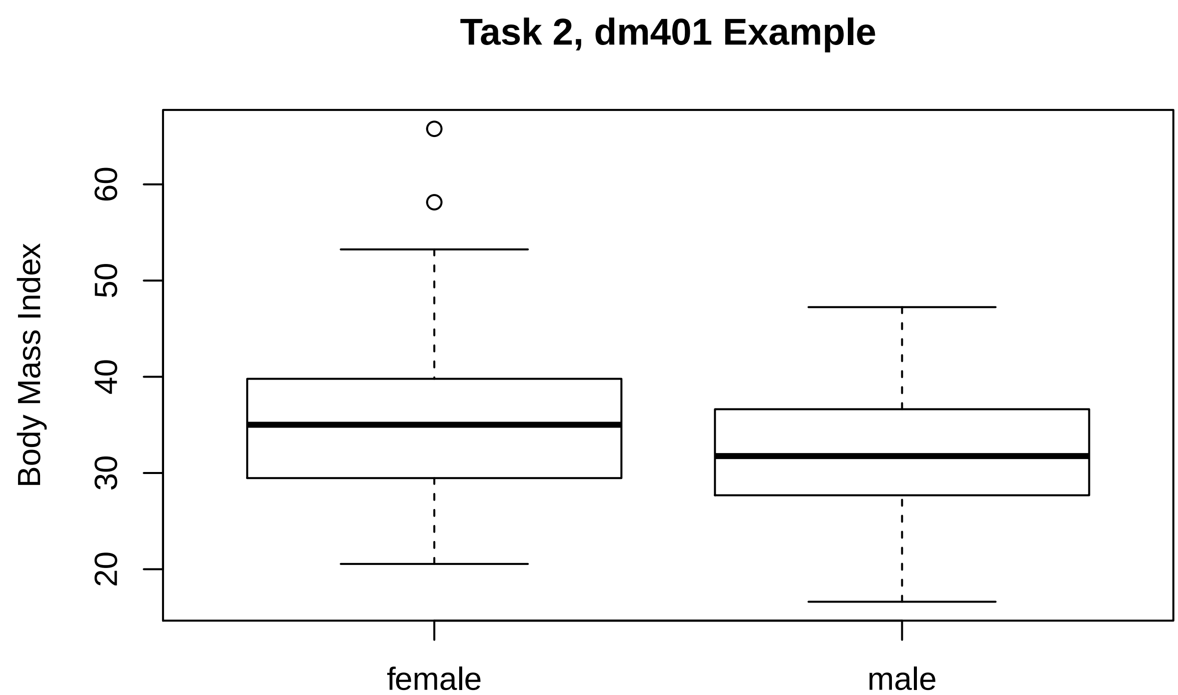
```
Proportion  0.562  0.438
---------------------------------------------------------------------------
smoking
       n  missing distinct
     146        0        2

Value      nonsmoker    smoker
Frequency        106        40
Proportion     0.726     0.274
---------------------------------------------------------------------------
```

4.2 Task 2: Is there an important difference in BMI by gender?

I'll start here by re-creating the `bootdif` function, useful for building bootstrap confidence intervals for the population mean difference using independent samples.

```
`bootdif` <-
  function(y, g, conf.level=0.95, B.reps = 2000) {
    require(Hmisc)
    lowq = (1 - conf.level)/2
    g <- as.factor(g)
    a <- attr(smean.cl.boot(y[g==levels(g)[1]], B=B.reps, reps=TRUE),'reps')
    b <- attr(smean.cl.boot(y[g==levels(g)[2]], B=B.reps, reps=TRUE),'reps')
    meandif <- diff(tapply(y, g, mean, na.rm=TRUE))
    a.b <- quantile(b-a, c(lowq,1-lowq))
    res <- c(meandif, a.b)
    names(res) <- c('Mean Difference',lowq, 1-lowq)
    res
  }
```

```
attach(dm401)
boxplot(bmi ~ female, ylab="Body Mass Index", main="Task 2, dm401 Example")
```




```r
by(bmi, female, summary)
```

```
female: female
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  20.55   29.58   35.02   35.57   39.78   65.77 
--------------------------------------------------------
female: male
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  16.62   27.73   31.76   32.30   36.56   47.24 
```

```r
t.test(bmi ~ female)
```

```

    Welch Two Sample t-test

data:  bmi by female
t = 2.6506, df = 143.96, p-value = 0.008935
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.8316821 5.7096975
sample estimates:
mean in group female   mean in group male 
            35.56866             32.29797 
```

```r
bootdif(bmi, female)
```

```
Mean Difference           0.025           0.975
     -3.2706898      -5.6378521      -0.9978348
```

4.3 Task 3: Are the compliance measures (smoking status and eye exam) strongly correlated?

I'll start by re-creating the `twobytwo` function for performing detailed analyses of 2x2 tables.

```r
`twobytwo` <-
  function(a,b,c,d, namer1 = "Row1", namer2 = "Row2", namec1 = "Col1", namec2 = "Col2")
    # build 2 by 2 table and run Epi library's twoby2 command to summarize
    # from the row-by-row counts in a cross-tab
    # upper left cell is a, upper right is b, lower left is c, lower right is d
    # names are then given in order down the rows then across the columns
    # use standard epidemiological format - outcomes in columns, treatments in rows
  {
    require(Epi)
    .Table <- matrix(c(a, b, c, d), 2, 2, byrow=T,
                     dimnames=list(c(namer1, namer2), c(namec1, namec2)))
    twoby2(.Table)
  }
```

```r
table(smoking, eyexm)
```

```
           eyexm
smoking     no yes
  nonsmoker 73  33
  smoker    32   8
```

```r
twobytwo(33, 73, 8, 32, "Non-Smoker", "Smoker", "Eye Exam", "No Eye Exam")
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : Eye Exam
Comparing : Non-Smoker vs. Smoker

           Eye Exam No Eye Exam    P(Eye Exam) 95% conf. interval
Non-Smoker       33          73         0.3113    0.2306   0.4054
Smoker            8          32         0.2000    0.1033   0.3517

                                   95% conf. interval
             Relative Risk: 1.5566    0.7875   3.0769
         Sample Odds Ratio: 1.8082    0.7522   4.3467
```

```
Conditional MLE Odds Ratio: 1.8013    0.7122   5.0258
    Probability difference: 0.1113   -0.0567   0.2446

             Exact P-value: 0.2187
        Asymptotic P-value: 0.1856
------------------------------------------------------
```

4.4 Task 4: Is insurance status related to pneumovax?

```r
table(insurance, pnvax)
```

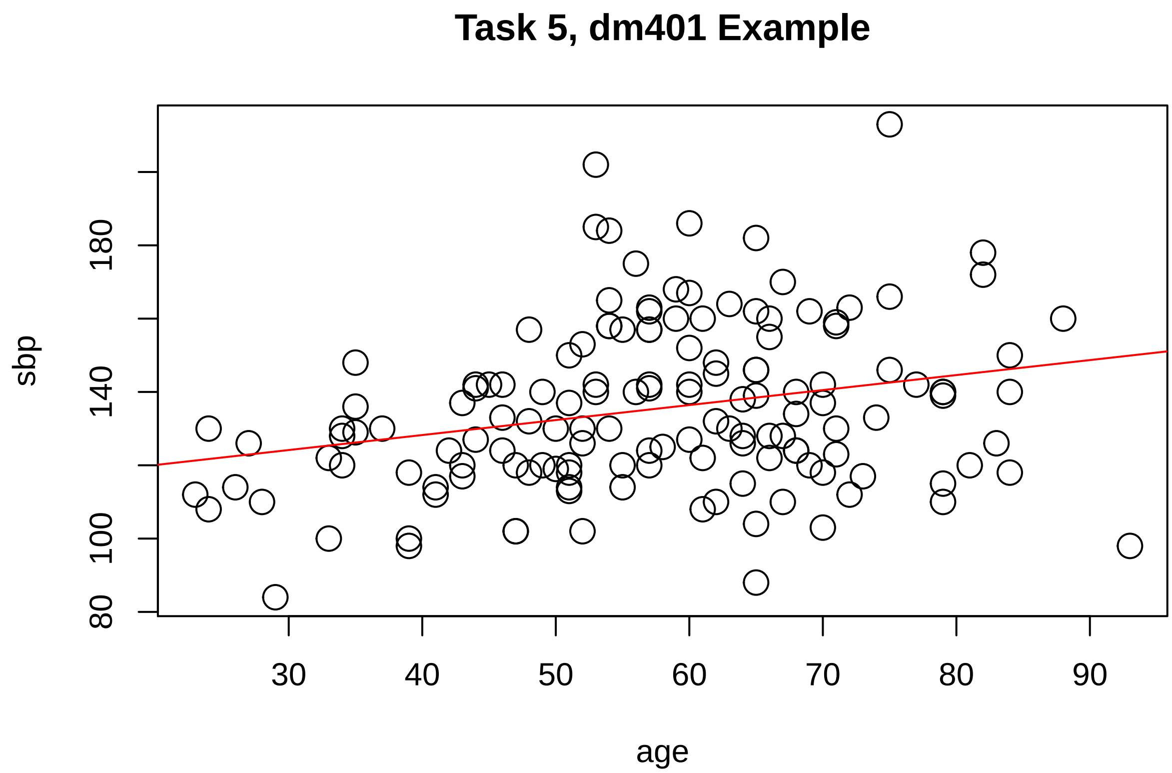
```
            pnvax
insurance    no yes
  Commercial 14  25
  Medicaid   11  14
  Medicare   13  39
  Uninsured  18  12
```

```r
chisq.test(table(insurance, pnvax))
```

```

    Pearson's Chi-squared test

data:  table(insurance, pnvax)
X-squared = 10.304, df = 3, p-value = 0.01615
```

4.5 Task 5: Is systolic blood pressure related to age? Is this a linear relationship?

```r
plot(sbp~age, cex=1.7, main="Task 5, dm401 Example")
abline(lm(sbp ~ age), col="red")
```

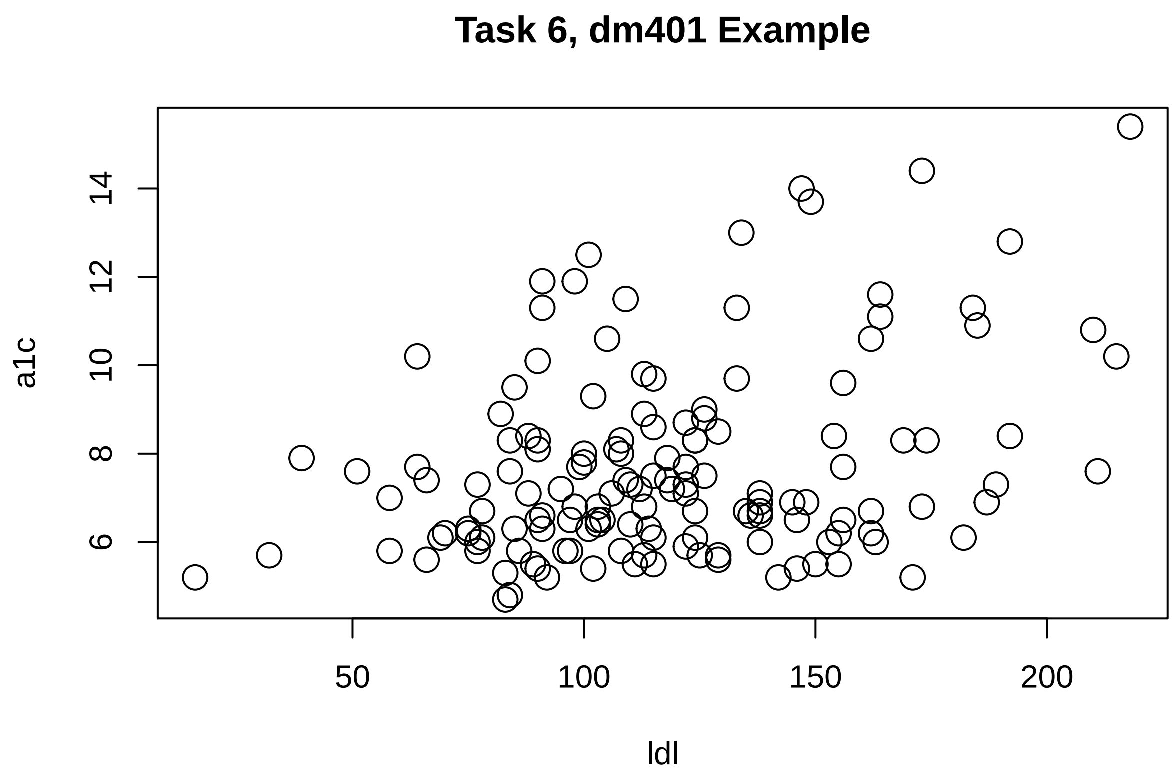



```r
summary(lm(sbp ~ age))
```

```
Call:
lm(formula = sbp ~ age)

Residuals:
    Min      1Q  Median      3Q     Max
-51.919 -14.205  -3.012  12.125  70.444

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 111.8782     7.3519  15.218  < 2e-16 ***
age           0.4090     0.1241   3.296  0.00123 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 22 on 144 degrees of freedom
Multiple R-squared:  0.07015,   Adjusted R-squared:  0.0637
F-statistic: 10.86 on 1 and 144 DF,  p-value: 0.001235
```

4.6 Task 6: Is hemoglobin A1c linearly related to LDL cholesterol (treating A1c as the outcome?)

```
plot(a1c ~ ldl, cex=1.7, main="Task 6, dm401 Example")
```




```
summary(lm(a1c ~ ldl))
```

```
Call:
lm(formula = a1c ~ ldl)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.4580 -1.4640 -0.5418  0.9777  5.8719 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 5.492431   0.555008   9.896  < 2e-16 ***
ldl         0.018512   0.004479   4.134 6.04e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 2.044 on 144 degrees of freedom
Multiple R-squared:  0.1061,    Adjusted R-squared:  0.09986
F-statistic: 17.09 on 1 and 144 DF,  p-value: 6.038e-05
```

4.7 Task 7: What can we say about the relationships of insurance and race (separately and together) on A1c? Should we consider collapsing the smallest "race/ethnicity" category?

```r
summary(lm(a1c ~ insurance))
```

```
Call:
lm(formula = a1c ~ insurance)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0865 -1.4606 -0.5865  0.8144  8.2205

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         7.17949    0.33876  21.194   <2e-16 ***
insuranceMedicaid  -0.07549    0.54201  -0.139   0.8894
insuranceMedicare   0.70705    0.44814   1.578   0.1168
insuranceUninsured  1.26051    0.51375   2.454   0.0154 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.116 on 142 degrees of freedom
Multiple R-squared:  0.05587,   Adjusted R-squared:  0.03593
F-statistic: 2.801 on 3 and 142 DF,  p-value: 0.04219
```

```r
summary(lm(a1c ~ raceeth))
```

```
Call:
lm(formula = a1c ~ raceeth)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0653 -1.5078 -0.6153  0.7672  7.5347

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)       7.8653     0.2536  31.017   <2e-16 ***
raceethHispanic  -1.0953     0.7261  -1.508    0.134
raceethWhite     -0.2575     0.3696  -0.697    0.487
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.152 on 143 degrees of freedom
Multiple R-squared:  0.01647,   Adjusted R-squared:  0.002712
F-statistic: 1.197 on 2 and 143 DF,  p-value: 0.3051
```

```r
summary(lm(a1c ~ insurance + raceeth))
```

```
Call:
lm(formula = a1c ~ insurance + raceeth)

Residuals:
    Min      1Q  Median      3Q     Max
-3.1699 -1.4375 -0.5674  0.8287  8.0575

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)          7.3425     0.4451  16.498   <2e-16 ***
insuranceMedicaid   -0.1408     0.5691  -0.247   0.8050
insuranceMedicare    0.6274     0.4604   1.363   0.1751
insuranceUninsured   1.1859     0.5401   2.196   0.0298 *
raceethHispanic     -0.9070     0.7243  -1.252   0.2125
raceethWhite        -0.1050     0.3887  -0.270   0.7874
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.119 on 140 degrees of freedom
Multiple R-squared:  0.06636,   Adjusted R-squared:  0.03302
F-statistic:  1.99 on 5 and 140 DF,  p-value: 0.08371
```

```r
table(raceeth)
```

```
raceeth
   Black Hispanic    White
      72       10       64
```

```r
summary(lm(a1c ~ raceeth=="White"))
```

```
Call:
lm(formula = a1c ~ raceeth == "White")
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-2.9317 -1.5078 -0.5817  0.7493  7.6683 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)              7.7317     0.2387  32.396   <2e-16 ***
raceeth == "White"TRUE  -0.1239     0.3605  -0.344    0.732    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.161 on 144 degrees of freedom
Multiple R-squared:  0.0008197, Adjusted R-squared:  -0.006119 
F-statistic: 0.1181 on 1 and 144 DF,  p-value: 0.7316
```

```r
summary(lm(a1c ~ insurance + (raceeth=="White")))
```

```
Call:
lm(formula = a1c ~ insurance + (raceeth == "White"))

Residuals:
    Min      1Q  Median      3Q     Max 
-3.0748 -1.4464 -0.5929  0.8032  8.2374 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)             7.16257    0.42206  16.971   <2e-16 ***
insuranceMedicaid      -0.06466    0.56699  -0.114   0.9094    
insuranceMedicare       0.71226    0.45625   1.561   0.1207    
insuranceUninsured      1.27066    0.53696   2.366   0.0193 *  
raceeth == "White"TRUE  0.02537    0.37520   0.068   0.9462    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.123 on 141 degrees of freedom
Multiple R-squared:  0.0559,	Adjusted R-squared:  0.02912 
F-statistic: 2.087 on 4 and 141 DF,  p-value: 0.08561
```

```r
summary(lm(a1c ~ insurance * (raceeth=="White")))
```

```
Call:
lm(formula = a1c ~ insurance * (raceeth == "White"))

Residuals:
```

```
    Min      1Q  Median      3Q     Max 
-2.9875 -1.3792 -0.5643  0.8377  7.7692 

Coefficients:
                                          Estimate Std. Error t value
(Intercept)                                7.63077    0.59136  12.904
insuranceMedicaid                         -0.44656    0.76745  -0.582
insuranceMedicare                          0.03352    0.71559   0.047
insuranceUninsured                         0.71923    0.74589   0.964
raceeth == "White"TRUE                    -0.67692    0.72427  -0.935
insuranceMedicaid:raceeth == "White"TRUE   0.34271    1.23351   0.278
insuranceMedicare:raceeth == "White"TRUE   1.15847    0.93614   1.237
insuranceUninsured:raceeth == "White"TRUE  1.01442    1.13995   0.890
                                          Pr(>|t|)    
(Intercept)                                 <2e-16 ***
insuranceMedicaid                            0.562    
insuranceMedicare                            0.963    
insuranceUninsured                           0.337    
raceeth == "White"TRUE                       0.352    
insuranceMedicaid:raceeth == "White"TRUE     0.782    
insuranceMedicare:raceeth == "White"TRUE     0.218    
insuranceUninsured:raceeth == "White"TRUE    0.375    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.132 on 138 degrees of freedom
Multiple R-squared:  0.06797,	Adjusted R-squared:  0.0207 
F-statistic: 1.438 on 7 and 138 DF,  p-value: 0.195
```

4.8 Task 8: How does the impact of insurance (ignoring race/ethnicity) on A1c change if we adjust A1c for the effect of LDL?

```r
summary(lm(a1c ~ insurance))
```

```
Call:
lm(formula = a1c ~ insurance)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.0865 -1.4606 -0.5865  0.8144  8.2205 
```

```
Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         7.17949    0.33876  21.194   <2e-16 ***
insuranceMedicaid  -0.07549    0.54201  -0.139   0.8894
insuranceMedicare   0.70705    0.44814   1.578   0.1168
insuranceUninsured  1.26051    0.51375   2.454   0.0154 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.116 on 142 degrees of freedom
Multiple R-squared:  0.05587,	Adjusted R-squared:  0.03593
F-statistic: 2.801 on 3 and 142 DF,  p-value: 0.04219
```

```r
summary(lm(a1c ~ insurance + ldl))
```

```
Call:
lm(formula = a1c ~ insurance + ldl)

Residuals:
    Min      1Q  Median      3Q     Max
-3.6628 -1.3276 -0.5175  1.0413  6.4717

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        4.853649   0.638552   7.601 3.74e-12 ***
insuranceMedicaid  0.265361   0.519004   0.511  0.60995
insuranceMedicare  0.812967   0.424606   1.915  0.05756 .
insuranceUninsured 1.375822   0.486694   2.827  0.00538 **
ldl                0.018691   0.004439   4.211 4.51e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.001 on 141 degrees of freedom
Multiple R-squared:  0.1613,	Adjusted R-squared:  0.1375
F-statistic: 6.781 on 4 and 141 DF,  p-value: 5.075e-05
```

4.9 Task 9: Build a kitchen sink model to predict A1c using main effects of the other ten variables as predictors. Then use the step function to identify a subset model for further analysis.

```r
summary(lm(a1c ~ ldl + sbp + insurance + eyexm + pnvax + age + bmi + raceeth + female +
```

```
Call:
lm(formula = a1c ~ ldl + sbp + insurance + eyexm + pnvax + age +
    bmi + raceeth + female + smoking)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4919 -1.3185 -0.3683  0.9667  6.1744

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         5.231381   1.588309   3.294  0.00127 **
ldl                 0.019038   0.004600   4.139 6.18e-05 ***
sbp                 0.002440   0.008128   0.300  0.76445
insuranceMedicaid   0.242506   0.550909   0.440  0.66052
insuranceMedicare   0.853014   0.529003   1.612  0.10924
insuranceUninsured  1.307311   0.523294   2.498  0.01371 *
eyexmyes           -0.734565   0.381482  -1.926  0.05631 .
pnvaxyes           -0.067423   0.369029  -0.183  0.85531
age                -0.005002   0.015826  -0.316  0.75246
bmi                -0.002321   0.023531  -0.099  0.92159
raceethHispanic    -1.220652   0.713643  -1.710  0.08953 .
raceethWhite       -0.170904   0.381686  -0.448  0.65506
femalemale         -0.060606   0.357401  -0.170  0.86561
smokingsmoker       0.196833   0.394818   0.499  0.61893
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.01 on 132 degrees of freedom
Multiple R-squared:  0.2074,    Adjusted R-squared:  0.1293
F-statistic: 2.657 on 13 and 132 DF,  p-value: 0.002467
```

```r
step(lm(a1c ~ ldl + sbp + insurance + eyexm + pnvax + age + bmi + raceeth + female + smo
```

```
Start:  AIC=217.2
a1c ~ ldl + sbp + insurance + eyexm + pnvax + age + bmi + raceeth +
    female + smoking

            Df Sum of Sq    RSS    AIC
- bmi        1     0.039 533.57 215.21
- female     1     0.116 533.65 215.24
- pnvax      1     0.135 533.67 215.24
- sbp        1     0.364 533.90 215.30
- age        1     0.404 533.94 215.31
- smoking    1     1.005 534.54 215.48
- raceeth    2    11.832 545.36 216.41
```

```
<none>                    533.53 217.20
- eyexm      1    14.986 548.52 219.25
- insurance  3    31.254 564.79 219.51
- ldl        1    69.236 602.77 233.02

Step:  AIC=215.21
a1c ~ ldl + sbp + insurance + eyexm + pnvax + age + raceeth +
    female + smoking

            Df Sum of Sq    RSS    AIC
- female     1     0.097 533.67 213.24
- pnvax      1     0.143 533.72 213.25
- sbp        1     0.336 533.91 213.31
- age        1     0.379 533.95 213.32
- smoking    1     1.029 534.60 213.50
- raceeth    2    11.891 545.46 214.43
<none>                    533.57 215.21
- eyexm      1    15.312 548.88 217.34
- insurance  3    31.338 564.91 217.55
- ldl        1    69.206 602.78 231.02

Step:  AIC=213.24
a1c ~ ldl + sbp + insurance + eyexm + pnvax + age + raceeth +
    smoking

            Df Sum of Sq    RSS    AIC
- pnvax      1     0.137 533.81 211.28
- age        1     0.358 534.03 211.34
- sbp        1     0.369 534.04 211.34
- smoking    1     0.994 534.66 211.51
- raceeth    2    12.758 546.43 212.69
<none>                    533.67 213.24
- eyexm      1    15.271 548.94 215.36
- insurance  3    31.272 564.94 215.56
- ldl        1    70.685 604.35 229.40

Step:  AIC=211.28
a1c ~ ldl + sbp + insurance + eyexm + age + raceeth + smoking

            Df Sum of Sq    RSS    AIC
- sbp        1     0.428 534.23 209.40
- age        1     0.428 534.23 209.40
- smoking    1     0.899 534.71 209.52
- raceeth    2    12.920 546.73 210.77
<none>                    533.81 211.28
```

```
- eyexm      1    15.641 549.45 213.50
- insurance  3    32.521 566.33 213.91
- ldl        1    70.549 604.35 227.40

Step:  AIC=209.4
a1c ~ ldl + insurance + eyexm + age + raceeth + smoking

            Df Sum of Sq    RSS    AIC
- age        1     0.272 534.51 207.47
- smoking    1     0.713 534.95 207.59
- raceeth    2    13.473 547.71 209.03
<none>                   534.23 209.40
- eyexm      1    15.451 549.69 211.56
- insurance  3    32.678 566.91 212.06
- ldl        1    72.346 606.58 225.94

Step:  AIC=207.47
a1c ~ ldl + insurance + eyexm + raceeth + smoking

            Df Sum of Sq    RSS    AIC
- smoking    1     0.803 535.31 205.69
- raceeth    2    13.370 547.88 207.08
<none>                   534.51 207.47
- eyexm      1    15.396 549.90 209.62
- insurance  3    32.688 567.19 210.14
- ldl        1    72.277 606.78 223.99

Step:  AIC=205.69
a1c ~ ldl + insurance + eyexm + raceeth

            Df Sum of Sq    RSS    AIC
- raceeth    2    14.241 549.55 205.52
<none>                   535.31 205.69
- eyexm      1    16.537 551.85 208.13
- insurance  3    32.352 567.66 208.26
- ldl        1    71.488 606.80 221.99

Step:  AIC=205.52
a1c ~ ldl + insurance + eyexm

            Df Sum of Sq    RSS    AIC
<none>                   549.55 205.52
- eyexm      1    14.985 564.53 207.45
- insurance  3    40.240 589.79 209.84
- ldl        1    65.882 615.43 220.05
```

```
Call:
lm(formula = a1c ~ ldl + insurance + eyexm)

Coefficients:
       (Intercept)                  ldl   insuranceMedicaid
           5.08018              0.01806             0.33631
 insuranceMedicare   insuranceUninsured            eyexmyes
           0.88307              1.44008            -0.71862
```

4.10 Task 10: Does the smaller model produced by the stepwise analysis above look like a useful partition of the original set of predictors? Evaluate this by looking at significance tests, but also model summary statistics (R^2, RMSE, etc.) for each model.

```r
summary(lm(a1c ~ ldl + insurance + eyexm))
```

```
Call:
lm(formula = a1c ~ ldl + insurance + eyexm)

Residuals:
    Min      1Q  Median      3Q     Max
-3.8507 -1.3427 -0.3116  0.9039  6.3838

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         5.080177   0.642809   7.903 7.28e-13 ***
ldl                 0.018055   0.004407   4.097 7.06e-05 ***
insuranceMedicaid   0.336315   0.515177   0.653  0.51495
insuranceMedicare   0.883068   0.421954   2.093  0.03817 *
insuranceUninsured  1.440076   0.483024   2.981  0.00339 **
eyexmyes           -0.718616   0.367802  -1.954  0.05272 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.981 on 140 degrees of freedom
Multiple R-squared:  0.1836,	Adjusted R-squared:  0.1544
F-statistic: 6.297 on 5 and 140 DF,  p-value: 2.659e-05
```

```r
summary(lm(a1c ~ ldl + sbp + insurance + eyexm + pnvax + age + bmi + raceeth + female +
```

```
Call:
lm(formula = a1c ~ ldl + sbp + insurance + eyexm + pnvax + age +
    bmi + raceeth + female + smoking)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4919 -1.3185 -0.3683  0.9667  6.1744

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         5.231381   1.588309   3.294  0.00127 **
ldl                 0.019038   0.004600   4.139 6.18e-05 ***
sbp                 0.002440   0.008128   0.300  0.76445
insuranceMedicaid   0.242506   0.550909   0.440  0.66052
insuranceMedicare   0.853014   0.529003   1.612  0.10924
insuranceUninsured  1.307311   0.523294   2.498  0.01371 *
eyexmyes           -0.734565   0.381482  -1.926  0.05631 .
pnvaxyes           -0.067423   0.369029  -0.183  0.85531
age                -0.005002   0.015826  -0.316  0.75246
bmi                -0.002321   0.023531  -0.099  0.92159
raceethHispanic    -1.220652   0.713643  -1.710  0.08953 .
raceethWhite       -0.170904   0.381686  -0.448  0.65506
femalemale         -0.060606   0.357401  -0.170  0.86561
smokingsmoker       0.196833   0.394818   0.499  0.61893
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.01 on 132 degrees of freedom
Multiple R-squared:  0.2074,    Adjusted R-squared:  0.1293
F-statistic: 2.657 on 13 and 132 DF,  p-value: 0.002467
```

```r
detach(dm401)
```

5 The SEPSIS and Ibuprofen Study: A Logistic Regression Example

This example is drawn from Dupont WD Statistical Modeling for Biomedical Researchers, Cambridge University Press, 2002: 1st Edition, Exercise 4.25. The original study was Bernard GR et al. (1997) The effects of ibuprofen on the physiology and survival of patients with sepsis. The Ibuprofen in Sepsis Study Group. N Engl J Med 336: 912-918.

5.1 The Data Set

We're going to look now at 30-day mortality in a sample of 350 septic patients as a function of

- receiving either ibuprofen or placebo treatment,
- their race (white or African-American),
- and their baseline APACHE (Acute Physiology and Chronic Health Evaluation) score.

APACHE score is a composite measure of the patient's degree of morbidity collected just prior to recruitment into the study, and is highly correlated with survival.

```
sep <- read.csv("sep.csv")
attach(sep)
summary(sep)
```

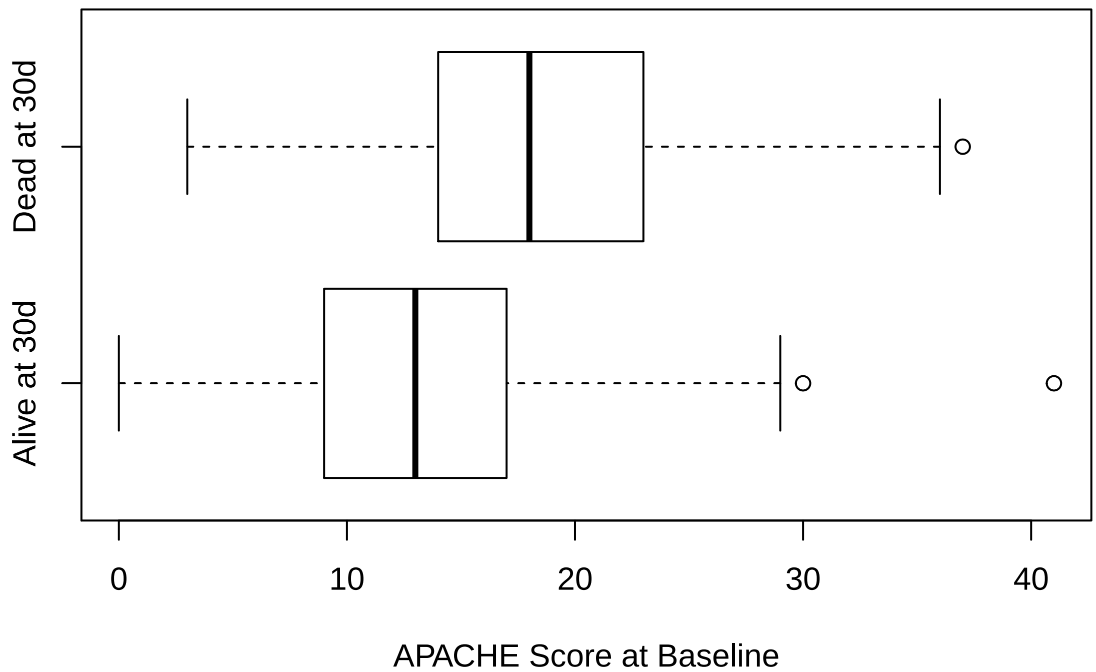
```
     pt.id              treatment    race         apache     
 Min.   :  1.00   ibuprofen:174   AA:109   Min.   : 0.00  
 1st Qu.: 88.25   placebo  :176   W :241   1st Qu.:11.00  
 Median :175.50                            Median :15.00  
 Mean   :175.50                            Mean   :15.74  
 3rd Qu.:262.75                            3rd Qu.:21.00  
 Max.   :350.00                            Max.   :41.00  
    death30d     
 Min.   :0.0000  
 1st Qu.:0.0000  
 Median :0.0000  
 Mean   :0.3914  
 3rd Qu.:1.0000  
 Max.   :1.0000  
```

Note that `death30d` = 0 if patient was alive 30 days after study entry, 1 if patient was dead 30 days after study entry.

We will estimate a **logistic regression model** to predict the probability of death at 30 days on the basis of these predictors. Overall, 39.14% were dead 30 days after study entry.

5.2 Is Death Rate related to APACHE scores?

```
boxplot(apache ~ death30d, horizontal=T,
        names=c("Alive at 30d", "Dead at 30d"), xlab="APACHE Score at Baseline")
```




```
tapply(apache, death30d, summary)
```

```
$`0`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    9.00   13.00   13.64   17.00   41.00

$`1`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      3      14      18      19      23      37
```

It looks like higher APACHE scores (on average) are associated with 30-day mortality. Is this significant? Well, we could do a t test, or the regression equivalent, using APACHE as the outcome variable . . .

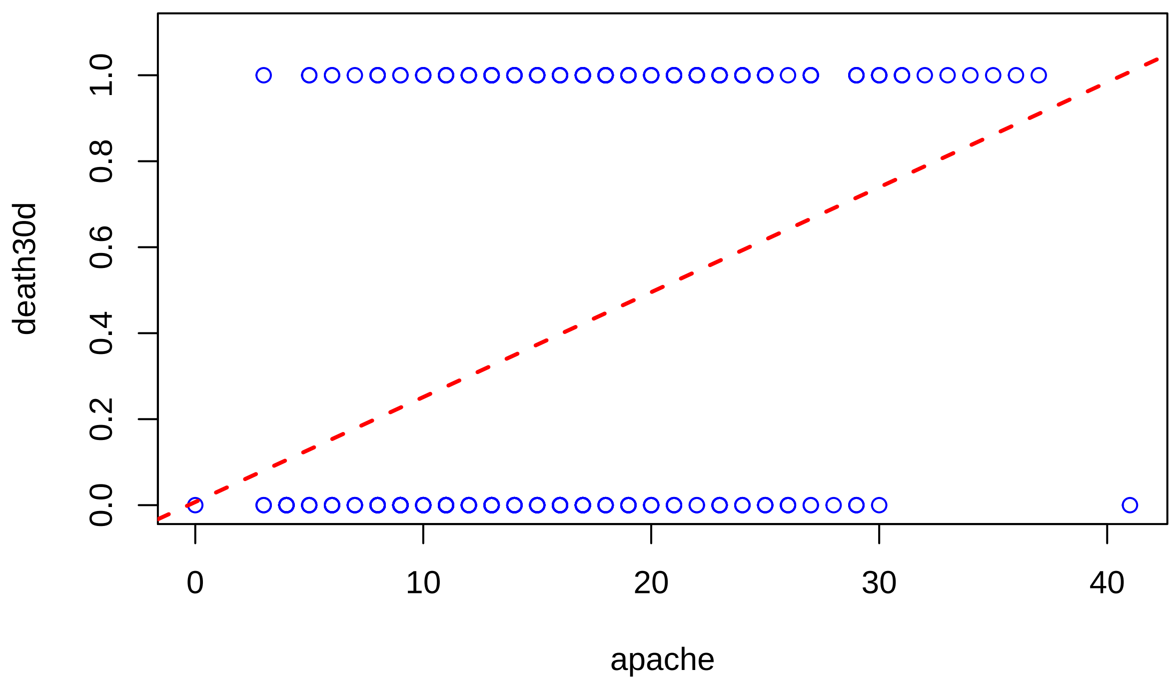
```
summary(lm(apache ~ death30d))
```

```

Call:
lm(formula = apache ~ death30d)

Residuals:
     Min       1Q   Median       3Q      Max
-16.0000  -4.6432  -0.6432   4.0000  27.3568
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  13.6432     0.4632  29.451   <2e-16 ***
death30d      5.3568     0.7404   7.235    3e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.761 on 348 degrees of freedom
Multiple R-squared:  0.1307,    Adjusted R-squared:  0.1282
F-statistic: 52.34 on 1 and 348 DF,  p-value: 2.998e-12
```

But that's backwards: death at 30 days is the *outcome* here, not a predictor. We need a regression model that predicts the probability of death! But, as we can see in the plot below, a straight line regression model won't predict `death30d` from `apache` well at all.

```r
plot(death30d ~ apache, ylim=c(0,1.1), col="blue")
abline(lm(death30d ~ apache), col="red", lwd=2, lty=2)
```



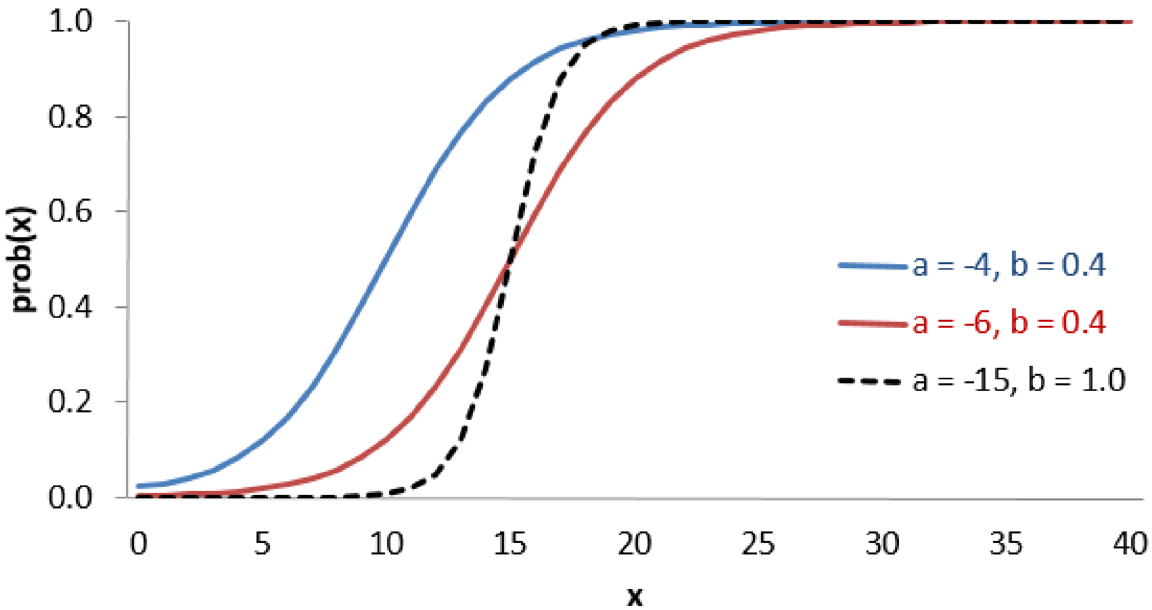


Figure 3: logistic1.png

5.3 The Logistic Regression Model

We will develop a logistic regression model to predict prob(x) = the probability that a patient with apache score x will die. In logistic regression, we fit probability functions of the form $prob(x) = exp[a + bx]/(1 + exp[a + bx])$, where a and b are unknown parameters (regression coefficients) that we will estimate from the data. So we have the logistic probability function

$$prob(x) = \frac{exp[a + bx]}{1 + exp[a + bx]}$$

This describes a family of curves appropriate for estimating probabilities on a 0-1 scale…

- The two solid curves (in blue and red) have the same value of the b parameter, which gives identical slopes.
- The different values of the a parameter shift the red curve to the right of the blue curve.
- The slopes of these curves increase as b gets larger.
- The magnitude of b determined how quickly prob(x) rises from 0 to 1.
- For a given b, a controls where the 50% survival point is located.
- Specifically, when $x = -a/b$, it turns out that prob(x) = 0.5, so, for instance, in our blue curve, prob(x) = 0.5 when x = 4/.4 = 10.

We can represent the probabilities in terms of their log odds, using the **logit function**:

$$logit(prob(x)) = log\frac{(prob(x))}{(1 - prob(x))} = a + bx$$

which works from any prob(x) between 0 and 1, where a and b are the regression coefficients for R to estimate, and the right-hand side is called the **linear predictor**.

5.4 Fitting a Logistic Regression Model

We wish to choose the best curve to fit our data. To do this, we inform R about our binary response variable (`death30d`, which is 1 for dead, 0 for alive), our predictor variable (`apache` score) and our desired regression function (the `logit`), as follows:

```
summary(glm(death30d ~ apache, family=binomial(logit)))
```

```

Call:
glm(formula = death30d ~ apache, family = binomial(logit))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2153  -0.9029  -0.6745   1.0867   2.0324

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.26864    0.31569  -7.186 6.66e-13 ***
apache       0.11299    0.01784   6.334 2.39e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 468.57  on 349  degrees of freedom
Residual deviance: 420.90  on 348  degrees of freedom
AIC: 424.9

Number of Fisher Scoring iterations: 4
```

The logistic regression procedure estimates the two key parameters of the logistic probability function.

- Our intercept a is estimated to be -2.27, and
- Our slope b for APACHE score is estimated to be 0.113, as can be seen in the coefficient estimates.

So the fitted prediction model for the probability of death by 30 days based on APACHE score is…

$$prob(x) = \frac{exp(a + bx)}{1 + exp(a + bx)} = \frac{exp(-2.27 + 0.113apache)}{1 + exp(-2.27 + 0.113apache)}$$

and we also know that the linear predictor is:

$$logit(prob(x)) = log\frac{prob(x)}{1 - prob(x)} = a + bx = -2.27 + 0.113apache.$$

5.5 Using the Fitted Logistic Regression Model To Make Predictions

We have 350 observations in the `sep` data, and five variables.

```
dim(sep)
```

```
[1] 350   5
```

The first patient in the data set, shown below, had an APACHE score of 27.

```
sep[1,]
```

```
  pt.id treatment race apache death30d
1     1   placebo    W     27        1
```

While we know that this patient died, based on their APACHE score and our model, what was their estimated probability of 30-day mortality?

- The linear predictor for patient 1 must be $-2.27 + 0.113(27)$, or 0.781.
- To get to a predicted probability, we'll need to exponentiate that result:

$exp(-2.27 + 0.113(27)) = exp(.781)$ or 2.184

- And the logistic probabilty function yields:

$$prob(x) = \frac{exp(-2.27 + 0.113apache)}{1 + exp(-2.27 + 0.113apache)} = \frac{2.184}{1 + 2.184}$$

$= 0.69$

Similarly, the second patient has an APACHE score of 14. We can calculate their estimated 30-day mortality risk as follows:

- Linear predictor is -2.27 + 0.113 (14) = -0.688
- Exponentiating, we get exp(-0.688) = 0.5026
- And so the probability of death by 30 days is 0.5026/(1 + 0.5026) = 0.33

The good news is that R will calculate these probabilities for you.

```
model1 <- glm(death30d ~ apache, family=binomial(logit))
fitted(model1)[c(1:2)]
```

```
        1         2 
0.6861055 0.3347359 
```

5.6 Interpreting the Logistic Regression Model Summary

Returning to our fitted model, we are left to interpret the remaining logistic regression output.

```
summary(model1)
```

```

Call:
glm(formula = death30d ~ apache, family = binomial(logit))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2153  -0.9029  -0.6745   1.0867   2.0324

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.26864    0.31569  -7.186 6.66e-13 ***
apache       0.11299    0.01784   6.334 2.39e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 468.57  on 349  degrees of freedom
Residual deviance: 420.90  on 348  degrees of freedom
AIC: 424.9

Number of Fisher Scoring iterations: 4
```

We interpret the coefficients in terms of log odds, or (after exponentiating) as odds ratios.

- For instance, an increase of 1 point in APACHE score is associated with an increase of 0.113 in the log odds of 30-day mortality.
- Or, we can exponentiate the coefficient (i.e. calculate $exp[0.113] = 1.12$) which is interpreted as the odds ratio comparing the odds of death for a patient with APACHE score = x + 1 to the odds of death for a patient with APACHE score = x.
- In general, $exp(x)$ is the odds ratio for the outcome (here, death) associated with a one-unit increase in x.
- A property of logistic regression is that this ratio remains constant for all values of x. So in this case, an increase of one point in the APACHE score is associated with an increase by a factor of 1.12 in the odds of death.

Our p value is `2.39e-10` (or 2.39 x 10^{-10}, i.e. a very, very small number) for APACHE, indicating (according, technically, to a Wald test) that the APACHE score has statistically significant predictive value (at usual α levels) for 30-day mortality risk.

- As in simple linear regression, our null hypothesis here is that the predictor is of no help

in predicting the outcome, and our alternative is that the predictor is of statistically significant help.
- Note that, as in simple linear regression, we generally don't interpret the *p* value associated with the intercept term, since we will by default include it in our logistic regression modeling.

5.7 The Analysis of Deviance

We'll skip the rest of the output here. To assess whether the model (overall) has a statistically significant effect, we can run an Analysis of Deviance table as follows (note that Anova must be capitalized here, and is part of the `car` library)...

```
library(car)
model1 <- glm(death30d ~ apache, family=binomial(logit))
Anova(model1, type="II")
```

```
Analysis of Deviance Table (Type II tests)

Response: death30d
       LR Chisq Df Pr(>Chisq)
apache   47.668  1  5.048e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This table provides a *p* value for the improvement in the deviance statistic due to the inclusion of apache score in the model, and is in that sense somewhat comparable to an overall ANOVA F test in linear regression. Here, again, the impact is statistically significant.

6 Logistic Regression with Multiple Predictors

Now, suppose we consider including additional information beyond the APACHE score, starting by including the treatment received by the patient. Does adding the treatment statistically significantly improve the quality of the predictions we make?

```
summary(glm(death30d ~ apache + treatment, family=binomial(logit)))
```

```
Call:
glm(formula = death30d ~ apache + treatment, family = binomial(logit))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2869  -0.9085  -0.6627   1.1151   2.0036
```

```
Coefficients:
                 Estimate Std. Error z value Pr(>|z|)    
(Intercept)      -2.43628    0.35187  -6.924 4.39e-12 ***
apache            0.11467    0.01798   6.379 1.78e-10 ***
treatmentplacebo  0.27386    0.23693   1.156    0.248    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 468.57  on 349  degrees of freedom
Residual deviance: 419.56  on 347  degrees of freedom
AIC: 425.56

Number of Fisher Scoring iterations: 4
```

It looks like the main effect of `treatment` doesn't add statistically significant predictive value (Wald test $p = 0.248$) to the model with APACHE score. What is we add `race` as well?

```r
model3 <- glm(death30d ~ apache + treatment + race, family=binomial(logit))
summary(model3)
```

```
Call:
glm(formula = death30d ~ apache + treatment + race, family = binomial(logit))

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.3213  -0.9067  -0.6471   1.1045   2.0220  

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)    
(Intercept)      -2.25697    0.41278  -5.468 4.56e-08 ***
apache            0.11207    0.01826   6.138 8.36e-10 ***
treatmentplacebo  0.28622    0.23773   1.204    0.229    
raceW            -0.20888    0.25740  -0.812    0.417    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 468.57  on 349  degrees of freedom
Residual deviance: 418.90  on 346  degrees of freedom
AIC: 426.9

Number of Fisher Scoring iterations: 4
```

6.1 Making Predictions

We can calculate the fitted probabilities for the first two patients, using this model, as follows.

```
model3$fitted.values[1:2]
```

```
        1         2 
0.6998077 0.3344952 
```

We can also calculate the linear predictors associated with the first two patients, using the following.

```
model3$linear.predictors[1:2]
```

```
         1          2 
 0.8463821 -0.6879231 
```

```
detach(sep)
```

7 The demodata Example: A Data Management Primer

I built a small data set (100 rows, and 18 columns) contained in the `demodata.csv` file in the **Data and Code** page of the course website. The purpose is to demonstrate ways of importing data of varying types into R in ways that are useful for doing the sorts of analyses you'll do in your projects.

```
demodata <- read.csv("demodata.csv")
str(demodata)
```

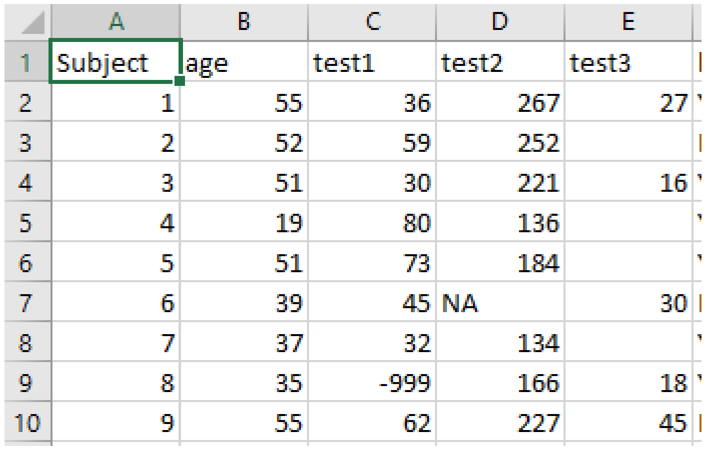
```
'data.frame':	100 obs. of  18 variables:
 $ Subject : int  1 2 3 4 5 6 7 8 9 10 ...
 $ age     : int  55 52 51 19 51 39 37 35 55 32 ...
 $ test1   : int  36 59 30 80 73 45 32 -999 62 40 ...
 $ test2   : int  267 252 221 136 184 NA 134 166 227 154 ...
 $ test3   : int  27 NA 16 NA NA 30 NA 18 45 NA ...
 $ histA   : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 1 2 2 1 2 ...
 $ histB   : int  2 2 2 1 1 1 2 2 2 2 ...
 $ histC   : int  0 0 1 0 0 1 1 1 1 0 ...
 $ histD   : int  1 0 0 0 0 0 NA 1 0 1 ...
 $ histE   : int  1 0 NA NA NA 1 1 1 1 1 ...
 $ histF   : int  1 0 1 99 0 77 1 0 0 0 ...
 $ race    : int  4 4 1 3 1 2 3 4 3 2 ...
 $ rating  : Factor w/ 5 levels "Exc","Fair","Good",..: 1 5 NA 3 4 5 2 5 3 1 ...
 $ return  : Factor w/ 5 levels "","A","B","C",..: 3 4 5 2 2 5 4 2 1 2 ...
 $ rotation: Factor w/ 4 levels "Unknown","X",..: 2 3 3 4 1 4 2 3 3 4 ...
 $ reason  : Factor w/ 12 levels "anxiety","costly",..: 3 10 5 2 6 10 6 3 5 4 ...
```

```
$ date1    : Factor w/ 95 levels "1/12/2011","1/13/2012",..: 73 70 61 41 35 13 13 51 52
$ date2    : int  40734 41430 41421 40999 40948 41210 41210 41369 41040 40722 ...
```

7.1 A Quick Summary of the Data, as Initially Imported

```r
summary(demodata) ## basic numerical summaries of the eighteen variables
```

```
   Subject            age             test1               test2      
 Min.   :  1.00   Min.   :19.00   Min.   :-999.00   Min.   :102.0  
 1st Qu.: 25.75   1st Qu.:33.75   1st Qu.:  32.75   1st Qu.:162.0  
 Median : 50.50   Median :50.50   Median :  48.00   Median :189.0  
 Mean   : 50.50   Mean   :48.23   Mean   :  18.25   Mean   :198.5  
 3rd Qu.: 75.25   3rd Qu.:60.25   3rd Qu.:  65.25   3rd Qu.:243.0  
 Max.   :100.00   Max.   :75.00   Max.   :  80.00   Max.   :300.0  
                                                    NA's   :5      
     test3        histA        histB          histC          histD       
 Min.   : 2.00   No :54   Min.   :1.00   Min.   :0.00   Min.   :0.0000  
 1st Qu.:10.00   Yes:46   1st Qu.:1.00   1st Qu.:0.00   1st Qu.:0.0000  
 Median :25.00            Median :2.00   Median :0.00   Median :1.0000  
 Mean   :24.26            Mean   :1.52   Mean   :0.46   Mean   :0.5532  
 3rd Qu.:38.00            3rd Qu.:2.00   3rd Qu.:1.00   3rd Qu.:1.0000  
 Max.   :48.00            Max.   :2.00   Max.   :1.00   Max.   :1.0000  
 NA's   :57                                             NA's   :6       
     histE            histF            race          rating    return
 Min.   :0.0000   Min.   : 0.00   Min.   :1.00   Exc    : 7    :26  
 1st Qu.:0.0000   1st Qu.: 0.00   1st Qu.:1.75   Fair   : 9   A:14  
 Median :0.0000   Median : 1.00   Median :3.00   Good   :54   B:13  
 Mean   :0.4932   Mean   : 7.93   Mean   :2.57   Poor   : 5   C:30  
 3rd Qu.:1.0000   3rd Qu.: 1.00   3rd Qu.:4.00   V Good:21   D:17  
 Max.   :1.0000   Max.   :99.00   Max.   :4.00   NA's   : 4         
 NA's   :27                                                          
   rotation          reason             date1            date2      
 Unknown: 4   expensive:22   10/17/2012: 2   Min.   :40541  
 X      :23   fear     :15   10/28/2012: 2   1st Qu.:40806  
 Y      :47   no time  :13   5/11/2012 : 2   Median :41040  
 Z      :26   tied up  : 8   5/26/2013 : 2   Mean   :41055  
              costly   : 7   6/5/2013  : 2   3rd Qu.:41247  
              too busy : 7   1/12/2011 : 1   Max.   :41617  
              (Other)  :28   (Other)   :89                  
```



	A	B	C	D	E
1	Subject	age	test1	test2	test3
2	1	55	36	267	27
3	2	52	59	252	
4	3	51	30	221	16
5	4	19	80	136	
6	5	51	73	184	
7	6	39	45	NA	30
8	7	37	32	134	
9	8	35	-999	166	18
10	9	55	62	227	45

Figure 4: first5.png

8 Recoding Continuous Variables, including Time-to-Event and Count Variables

Here are the first 10 rows of the first five variables in the `demodata.csv` file, as they appear in Excel.

Continuous variables are relatively easy to import into R.

- The `age` variable has no missing values, while `test1`, `test2` and `test3` each contain various ways of representing missing values, indicated by `-999` for `test1`, by `NA` for `test2` and by blank cells (which R converts to NAs) for `test3`.

When we import the demodata.csv file into R, we'll see from a summary of the first five columns in the data (those are the continuous variables here) that two of these approaches to coding missing data (`NA` and blanks) each work properly, while the use of `-999` causes problems.

After initial import into R, here's what the same part of the `demodata` data frame looks like…

```
demodata[1:10, 1:5] ## shows first ten rows of the first five variables
```

```
  Subject age test1 test2 test3
1       1  55    36   267    27
2       2  52    59   252    NA
3       3  51    30   221    16
4       4  19    80   136    NA
5       5  51    73   184    NA
6       6  39    45    NA    30
7       7  37    32   134    NA
8       8  35  -999   166    18
9       9  55    62   227    45
```

```
10       10  32    40    154    NA
```

```r
summary(demodata[1:5]) ## summarizes the first five variables
```

```
    Subject            age            test1            test2      
 Min.   :  1.00   Min.   :19.00   Min.   :-999.00   Min.   :102.0  
 1st Qu.: 25.75   1st Qu.:33.75   1st Qu.:   32.75   1st Qu.:162.0  
 Median : 50.50   Median :50.50   Median :   48.00   Median :189.0  
 Mean   : 50.50   Mean   :48.23   Mean   :   18.25   Mean   :198.5  
 3rd Qu.: 75.25   3rd Qu.:60.25   3rd Qu.:   65.25   3rd Qu.:243.0  
 Max.   :100.00   Max.   :75.00   Max.   :   80.00   Max.   :300.0  
                                                     NA's   :5      
     test3      
 Min.   : 2.00  
 1st Qu.:10.00  
 Median :25.00  
 Mean   :24.26  
 3rd Qu.:38.00  
 Max.   :48.00  
 NA's   :57     
```

In the `test2` and `test3` cases, we see that R correctly identifies the values `NA` (in the case of `test2`) and ‘`blank'' (in the case of`test3‘) as indicating missingness.

But, for `test1`, we have a problem, in that R thinks that the code value `-999` is in fact a legitimate value, rather than a placeholder indicating missingness, and includes those values of `-999` when calculating the minimum and other summary statistics.

So, we need to fix test1 so that it treats the three -999s as missing values. To do this, try the following…

```r
is.na(demodata$test1) <- demodata$test1==-999
summary(demodata$test1)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  20.00   35.00   48.00   49.71   66.00   80.00       3 
```

8.1 Imputing Values for the Missing Observations in Continuous Variables

Here is one potential approach for imputing values for the missing observations in `test1`, `test2` and `test3`.

```r
library(Hmisc)
na.pattern(demodata[c("test1", "test2")])
```

```
pattern
```

```
00 01 10
92  5  3
```

For test1 and test2, we have only 3 and 5 missing values, respectively, which is less than 10% of the data, and less than 20 observations that are missing in each column. Confronted with relatively modest missingness like this, under certain circumstances, like in your class project, I might recommend a simple imputation before including these as covariates in a propensity model.

```r
demodata$test1.i <- impute(demodata$test1, fun="random")
set.seed(500001); summary(demodata[c("test1", "test1.i")])
```

```

Imputed Values:

[1] 20 36 41

     test1           test1.i     
 Min.   :20.00   Min.   :20.00  
 1st Qu.:35.00   1st Qu.:35.00  
 Median :48.00   Median :48.00  
 Mean   :49.71   Mean   :49.19  
 3rd Qu.:66.00   3rd Qu.:65.25  
 Max.   :80.00   Max.   :80.00  
 NA's   :3                      
```

```r
demodata$test2.i <- impute(demodata$test2, fun="random")
set.seed(500002); summary(demodata[c("test2", "test2.i")])
```

```

Imputed Values:

[1] 252 247 184 291 295

     test2           test2.i     
 Min.   :102.0   Min.   :102.0  
 1st Qu.:162.0   1st Qu.:162.8  
 Median :189.0   Median :190.5  
 Mean   :198.5   Mean   :201.3  
 3rd Qu.:243.0   3rd Qu.:247.0  
 Max.   :300.0   Max.   :300.0  
 NA's   :5                      
```

Note that I'm using the `set.seed()` function here just to guarantee that if I rerun this Markdown file, I'll get the same imputed values.

On the other hand, for `test3`, we have 57 missing out of 100 values in total. Since this is both more than 20 missing values, and more than 10% of our data set, my project-specific advice indicates that we should create two new variables:

- one to indicate missingness in `test3`, which I will call `test3.NA` and
- another where we impute the same (I'll use the median) value for each missing observation in `test3`, which I'll call `test3.i`

```
demodata$test3.NA <- as.numeric(is.na(demodata$test3))
demodata$test3.i <- impute(demodata$test3, fun=median)

summary(demodata[c("test3", "test3.i", "test3.NA")])
```

```
 57 values imputed to 25

     test3           test3.i          test3.NA   
 Min.   : 2.00   Min.   : 2.00   Min.   :0.00  
 1st Qu.:10.00   1st Qu.:25.00   1st Qu.:0.00  
 Median :25.00   Median :25.00   Median :1.00  
 Mean   :24.26   Mean   :24.68   Mean   :0.57  
 3rd Qu.:38.00   3rd Qu.:25.00   3rd Qu.:1.00  
 Max.   :48.00   Max.   :48.00   Max.   :1.00  
 NA's   :57                                     
```

And we'd include `test1.i`, `test2.i` and both `test3.NA` and `test3.i` in our propensity model to represent the information, while leaving the original variables `test1`, `test2` and `test3` out of the model.

8.2 Creating a Binary Variable from a Continuous one

One more type of recoding is creating a binary or multi-categorical variable from a continuous one. For instance, we might create a binary variable that divides our patients into two groups, based on whether they were above or below the age of, say, 50. Here, I'll make the arbitrary choice to put those with ages equal to 50 into the "above" group.

```
demodata$age.50plus <- as.numeric(demodata$age >= 50)
by(demodata$age, demodata$age.50plus, summary) ## sanity check on recoding
```

```
demodata$age.50plus: 0
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  19.00   28.00   33.00   34.59   42.00   49.00 
--------------------------------------------------------
demodata$age.50plus: 1
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  50.00   55.50   60.00   61.33   66.00   75.00 
```

And we could create a factor, as well, from this new variable.

```
demodata$age.50plus.f <-
  factor(demodata$age.50plus, levels=c(1,0), labels=c("50 plus", "Less than 50"))
```

```
table(demodata$age.50plus.f, demodata$age.50plus) ## another sanity check
```

```
              0  1
50 plus       0 51
Less than 50 49  0
```

8.3 Creating A 4-Category Variable from a Continuous one

Now, what if we wanted to create a four-category factor by age? One approach would be to use the `cut2` function from the `Hmisc` library to select four groups of roughly equal size (these would be quartiles)...

```
## assumes library(Hmisc) has already been run
demodata$age.4groups <- cut2(demodata$age, g=4)
by(demodata$age, demodata$age.4groups, summary) ## sanity check
```

```
demodata$age.4groups: [19,34)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  19.00   24.00   28.00   27.28   31.00   33.00 
-------------------------------------------------------- 
demodata$age.4groups: [34,51)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  34.00   40.00   45.00   42.52   46.00   50.00 
-------------------------------------------------------- 
demodata$age.4groups: [51,61)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  51.00   54.00   56.00   55.64   58.00   60.00 
-------------------------------------------------------- 
demodata$age.4groups: [61,75]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  61.00   64.00   67.00   67.48   72.00   75.00 
```

Or, we could pre-specify that we want groups at Up to age 35, then 35 up to 50, and 50 up to 64 and finally 65 or older...

```
demodata$age.groups4 <- cut2(demodata$age, cuts=c(35,50,65))
by(demodata$age, demodata$age.groups4, summary)
```

```
demodata$age.groups4: [19,35)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  19.00   24.75   29.00   28.00   32.00   34.00 
-------------------------------------------------------- 
demodata$age.groups4: [35,50)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
```

	F	G	H	I	J	K
1	histA	histB	histC	histD	histE	histF
2	Yes	2	0	1	1	1
3	No	2	0	0	0	0
4	Yes	2	1	0		1
5	Yes	1	0	0		99
6	Yes	1	0	0		0
7	No	1	1	0	1	77
8	Yes	2	1	NA	1	1
9	Yes	2	1	1	1	0
10	No	2	1	0	1	0

Figure 5: sheet2.png

```
  35.00   41.00   45.00   43.38   46.00   49.00
--------------------------------------------------------
demodata$age.groups4: [50,65)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  50.00   54.00   58.00   57.39   61.00   64.00
--------------------------------------------------------
demodata$age.groups4: [65,75]
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   65.0    68.5    71.0    70.8    74.0    75.0
```

By default, the results of applying the `cut2` function is a single factor that divides the subjects into groups.

9 Recoding Binary Categorical Variables

Binary variables can come in many different forms. The easiest thing to deal with is a simple 1-0 numeric variable, where 1 indicates the presence of the characteristic and 0 its absence. But we can see lots of different options.

- The `histA` variable has Yes and No values, `histB` has 1 for Yes and 2 for No, while `histC` is set up as we'd usually prefer.
- Then variables `histD` and `histE` have missing values represented by `NA`s and blanks, respectively (which will work smoothly)
- Yet `histF` has three kinds of missing values: `99` for missing, `88` for no response and `77` for "don't know." We'll assume that all three possibilities should be treated as missing.

When we import the `demodata.csv` file into R, the `NA` and blanks approaches to coding missingness each work properly, but we still have work ahead.

```r
summary(demodata[c("histA", "histB", "histC", "histD", "histE", "histF")])
```

```
 histA        histB           histC           histD            histE
 No :54   Min.   :1.00   Min.   :0.00   Min.   :0.0000   Min.   :0.0000
 Yes:46   1st Qu.:1.00   1st Qu.:0.00   1st Qu.:0.0000   1st Qu.:0.0000
          Median :2.00   Median :0.00   Median :1.0000   Median :0.0000
          Mean   :1.52   Mean   :0.46   Mean   :0.5532   Mean   :0.4932
          3rd Qu.:2.00   3rd Qu.:1.00   3rd Qu.:1.0000   3rd Qu.:1.0000
          Max.   :2.00   Max.   :1.00   Max.   :1.0000   Max.   :1.0000
                                        NA's   :6        NA's   :27
     histF
 Min.   : 0.00
 1st Qu.: 0.00
 Median : 1.00
 Mean   : 7.93
 3rd Qu.: 1.00
 Max.   :99.00
```

9.1 Creating Factors and 1-0 variables

Most of the time, we're going to want to create both a 1-0 (in standard epidemiological format) and a factor version of a binary variable. The 1-0 version is generally more useful for outcomes, exposures and covariates, but there are times when the factor version is also helpful. So, here's how I might do that.

9.1.1 Converting `histA`

```r
table(demodata$histA)
```

```

 No Yes
 54  46
```

For `histA`, we already have a factor variable (Yes/No), but we need to get that into standard epidemiological format (with presence [i.e. Yes] first, and absence [No] second) and I'll label that histA.f, and then we'll also want a 1-0 numeric version, which I'll call `histA`, after I copy the original data to `histA.original`.

```r
demodata$histA.original <- demodata$histA
demodata$histA.f <- factor(demodata$histA, levels=c("Yes","No"))
demodata$histA <- as.numeric(demodata$histA.f == "Yes")

table(demodata$histA, demodata$histA.f)
```

```
    Yes No
  0   0 54
  1  46  0
```

```
table(demodata$histA.original, demodata$histA.f)
```

```
      Yes No
  No    0 54
  Yes  46  0
```

```
summary(demodata[c("histA.original", "histA", "histA.f")])
```

```
 histA.original     histA        histA.f
 No :54         Min.   :0.00   Yes:46
 Yes:46         1st Qu.:0.00   No :54
                Median :0.00
                Mean   :0.46
                3rd Qu.:1.00
                Max.   :1.00
```

9.1.2 Converting `histB`

```
table(demodata$histB)
```

```
 1  2
48 52
```

For `histB`, we already have a numeric variable, where 1 = Yes, and 2 = No, but we need to get that into 1-0 form, and also build a factor to describe the results in standard epidemiological format. To do so, use the following:

```
demodata$histB.original <- demodata$histB
demodata$histB <- as.numeric(demodata$histB == 1)
demodata$histB.f <- factor(demodata$histB, levels=c(1,0), labels=c("Yes", "No"))

table(demodata$histB, demodata$histB.original)
```

```
     1  2
  0  0 52
  1 48  0
```

```r
table(demodata$histB, demodata$histB.f)
```

```
   
    Yes No
  0   0 52
  1  48  0
```

```r
summary(demodata[c("histB.original", "histB", "histB.f")])
```

```
 histB.original     histB       histB.f
 Min.   :1.00   Min.   :0.00   Yes:48
 1st Qu.:1.00   1st Qu.:0.00   No :52
 Median :2.00   Median :0.00
 Mean   :1.52   Mean   :0.48
 3rd Qu.:2.00   3rd Qu.:1.00
 Max.   :2.00   Max.   :1.00
```

9.1.3 Converting histC

```r
table(demodata$histC)
```

```

 0  1
54 46
```

For histC, we already have a numeric variable, where 1 = Yes, and 0 = No, so that's great, and all we need is to also build a factor to describe the results in standard epidemiological format. To do so, use the following:

```r
demodata$histC.f <- factor(demodata$histC, levels=c(1,0), labels=c("Yes", "No"))
table(demodata$histC, demodata$histC.f)
```

```
   
    Yes No
  0   0 54
  1  46  0
```

OK, this looks great.

9.2 Dealing with Missingness in Binary Data

Now, we'll deal with missingness, in binary data, as shown in histD, histE and histF.

9.2.1 Imputation for `histD`

```
table(demodata$histD, useNA="ifany")
```

```
   0    1 <NA> 
  42   52    6 
```

In `histD`, we have a 1-0 numeric variable, and have successfully gotten R to recognize 6 missing values. To use this as a covariate, we'll first impute (simply) the 6 missing values, since we have less than 20 missing values (and less than 10% of our data missing, for that matter.)

```
## set seed to ensure that imputations are the same if we rerun the file
set.seed(500003); demodata$histD.i <- impute(demodata$histD, fun="random")
demodata$histD.f <- factor(demodata$histD.i, levels=c(1,0), labels=c("Yes","No"))
## And we'll do some sanity checks, as usual.
summary(demodata[c("histD", "histD.i", "histD.f")])
```

```
Imputed Values:

[1] 0 0 0 0 1 0

     histD            histD.i      histD.f 
 Min.   :0.0000   Min.   :0.00   Yes:53  
 1st Qu.:0.0000   1st Qu.:0.00   No :47  
 Median :1.0000   Median :1.00           
 Mean   :0.5532   Mean   :0.53           
 3rd Qu.:1.0000   3rd Qu.:1.00           
 Max.   :1.0000   Max.   :1.00           
 NA's   :6                               
```

9.2.2 Working with `histE`

```
table(demodata$histE, useNA="ifany")
```

```
   0    1 <NA> 
  37   36   27 
```

In `histE`, we again have a 1-0 numeric variable, and R has recognized 27 missing values. To use this as a covariate, we'll create both an indicator of missingness (called `histE.NA`) and then do a simple imputation of the same value for each of the 27 missing values, putting the result in `histE.i`. Then, we'll create a factor called `histE.f` with three levels: Yes, No and Missing.

```r
demodata$histE.NA <- as.numeric(is.na(demodata$histE))
demodata$histE.i <- impute(demodata$histE, fun=median)
demodata$histE.f <- factor(demodata$histE, levels=c(1,0), labels=c("Yes","No"), exclude=
## The exclude=NULL part keeps in the NAs.
## And we move on to sanity checking ...
summary(demodata[c("histE", "histE.i", "histE.NA", "histE.f")])
```

```
 27 values imputed to 0

     histE             histE.i          histE.NA        histE.f
 Min.   :0.0000   Min.   :0.00   Min.   :0.00   Yes :36
 1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:0.00   No  :37
 Median :0.0000   Median :0.00   Median :0.00   NA's:27
 Mean   :0.4932   Mean   :0.36   Mean   :0.27
 3rd Qu.:1.0000   3rd Qu.:1.00   3rd Qu.:1.00
 Max.   :1.0000   Max.   :1.00   Max.   :1.00
 NA's   :27
```

9.2.3 Working with histF

```r
table(demodata$histF, useNA="ifany")
```

```
 0  1 77 88 99
47 45  1  2  5
```

In histF, we again have a 1-0 numeric variable, but now we have codes 77, 88 and 99, all of which we'll take to mean missing values. So, we'll get R to recognize these values as missing in a new version of histF. Then, to use this as a covariate, we'll do a simple imputation (since the missingness rate < 10% and there are less than 20 missing values) into a variable called histF.i. Then, we'll create a factor called histF.f with two levels: Yes and No, based on the imputed values in histF.i.

```r
demodata$histF.original <- demodata$histF
is.na(demodata$histF) <- demodata$histF > 1
table(demodata$histF, useNA="ifany")
```

```
   0    1 <NA>
  47   45    8
```

```r
set.seed(500004); demodata$histF.i <- impute(demodata$histF, fun="random")
demodata$histF.f <- factor(demodata$histF.i, levels=c(1,0), labels=c("Yes","No"))
summary(demodata[c("histF.original", "histF", "histF.i", "histF.f")])
```

	L	M	N	O	P
1	race	rating	return	rotation	reason
2	4	Exc	B	X	expensive
3	4	V Good	C	Y	too busy
4	1	NA	D	Y	high priced
5	3	Good	A	Z	costly
6	1	Poor	A	Unknown	no time
7	2	V Good	D	Z	too busy
8	3	Fair	C	X	no time
9	4	V Good	A	Y	expensive
10	3	Good		Y	high priced

Figure 6: sheet3.png

```
Imputed Values:

[1] 1 1 0 0 0 1 1 0
```

```
 histF.original       histF            histF.i       histF.f
 Min.    : 0.00   Min.    :0.0000   Min.    :0.00   Yes:49
 1st Qu.: 0.00   1st Qu.:0.0000   1st Qu.:0.00   No :51
 Median : 1.00   Median :0.0000   Median :0.00
 Mean    : 7.93   Mean    :0.4891   Mean    :0.49
 3rd Qu.: 1.00   3rd Qu.:1.0000   3rd Qu.:1.00
 Max.    :99.00   Max.    :1.0000   Max.    :1.00
                  NA's    :8
```

10 Recoding Categorical Variables with More Than Two Categories

There are lots of things we might want to do with a multi-categorical variable, including rearranging its levels, create factors which are labeled properly and appear in a sensible order, create binary 1/0 variables for individual categories, deal with missingness sensibly, and collapse categories. In addition, a multi-categorical variable can be coded originally in several different forms.

We have five such variables here.

- `race` is coded as 1 = White, 2 = Black, 3 = Asian and 4 = All Other, with no missing values
- `rating` is either Exc, V Good, Good, Fair or Poor. There are 4 missing values, coded by `NA`.

- `return` is either A, B, C, or D. There are 26 missing values, coded in the .csv file by blanks.
- `rotation` is either X, Y or Z. There are 4 missing values, coded in the .csv as "Unknown".
- `reason` can take on 12 different values for primary reason why the subject did not go to the doctor. The `reason` variable has no missing values, but we might want to collapse the reasons into three groups, perhaps combining the several reasons pertaining to fear into one category, the reasons related to cost into another category, and reasons related to time into a third category.

```
summary(demodata[c("race", "rating", "return", "rotation", "reason")])
```

```
      race          rating    return    rotation           reason
 Min.   :1.00   Exc   : 7    :26    Unknown: 4   expensive:22
 1st Qu.:1.75   Fair  : 9   A:14    X      :23   fear     :15
 Median :3.00   Good  :54   B:13    Y      :47   no time  :13
 Mean   :2.57   Poor  : 5   C:30    Z      :26   tied up  : 8
 3rd Qu.:4.00   V Good:21   D:17                 costly   : 7
 Max.   :4.00   NA's  : 4                        too busy : 7
                                                 (Other)  :28
```

```
table(demodata$reason, useNA="ifany")
```

```

    anxiety      costly   expensive        fear high priced     no time
          5           7          22          15           4          13
      panic     swamped     tied up    too busy      unease       worry
          4           6           8           7           4           5
```

10.1 Working with race

As mentioned, `race` is coded as 1 = White, 2 = Black, 3 = Asian and 4 = All Other, with no missing values

```
table(demodata$race, useNA="ifany")
```

```

 1  2  3  4
25 21 26 28
```

To use race as a covariate, we would want to create a factor, as follows.

```
demodata$race.f <-
  factor(demodata$race, levels=c(1, 2, 3, 4),
         labels=c("White", "Black", "Asian", "Other"))
table(demodata$race, demodata$race.f, useNA="ifany")
```

```
   White Black Asian Other
  1    25     0     0     0
  2     0    21     0     0
  3     0     0    26     0
  4     0     0     0    28
```

Also, we would likely need a series of indicator / dummy 1-0 numeric variables, one for each of the four categories of race, although we might only use three of them in modeling.

```r
demodata$race.White <- as.numeric(demodata$race.f=="White")
demodata$race.Black <- as.numeric(demodata$race.f=="Black")
demodata$race.Asian <- as.numeric(demodata$race.f=="Asian")
demodata$race.Other <- as.numeric(demodata$race.f=="Other")

## Some quick sanity checks
summary(demodata[c("race", "race.f", "race.White")])
```

```
      race         race.f     race.White
 Min.   :1.00   White:25   Min.   :0.00
 1st Qu.:1.75   Black:21   1st Qu.:0.00
 Median :3.00   Asian:26   Median :0.00
 Mean   :2.57   Other:28   Mean   :0.25
 3rd Qu.:4.00              3rd Qu.:0.25
 Max.   :4.00              Max.   :1.00
```

```r
table(demodata$race.f, demodata$race.Asian)
```

```

         0  1
  White 25  0
  Black 21  0
  Asian  0 26
  Other 28  0
```

10.2 Working with rating

`rating` is either Exc, V Good, Good, Fair or Poor. There are 4 missing values, coded by `NA`.

```r
table(demodata$rating, useNA="ifany")
```

```

   Exc   Fair   Good   Poor V Good   <NA>
     7      9     54      5     21      4
```

That is a factor, but an annoyingly poor ordering of the variables. We could adjust that…

```
demodata$rating.f <- factor(demodata$rating, levels=c("Exc", "V Good", "Good", "Fair", "
table(demodata$rating, demodata$rating.f, useNA="ifany")
```

```
         Exc V Good Good Fair Poor <NA>
  Exc      7      0    0    0    0    0
  Fair     0      0    0    9    0    0
  Good     0      0   54    0    0    0
  Poor     0      0    0    0    5    0
  V Good   0     21    0    0    0    0
  <NA>     0      0    0    0    0    4
```

That's a much more meaningful ordering, but we still have four missing values. We could either impute (probably the better choice for your project) or create a new category for Missingness. Given that there are only 4 missing values (much less than 20) I would just impute, simply, as follows…

```
set.seed(500005); demodata$rating.f.i <- impute(demodata$rating.f, fun="random")
table(demodata$rating.f, demodata$rating.f.i, useNA="ifany")
```

```
         Exc V Good Good Fair Poor
  Exc      7      0    0    0    0
  V Good   0     21    0    0    0
  Good     0      0   54    0    0
  Fair     0      0    0    9    0
  Poor     0      0    0    0    5
  <NA>     0      1    2    1    0
```

And, as before, we could then create a series of indicator variables to represent the various categories.

What if we wanted to compare those with Exc, V Good or Good results to those with Fair or Poor results, in a binary variable? To do that, we could use the following approach:

```
demodata$rating.10 <- as.numeric(demodata$rating.f=="Exc" |
                                   demodata$rating.f.i=="V Good" |
                                   demodata$rating.f.i=="Good")
table(demodata$rating.f.i, demodata$rating.10)
```

```
          0  1
  Exc     0  7
  V Good  0 22
  Good    0 56
  Fair    9  0
  Poor    5  0
```

10.3 Working with `return`

`return` is either A, B, C, or D. There are 26 missing values, coded in the `.csv` file by blanks.

```
table(demodata$return, useNA="ifany")
```

```
    A  B  C  D 
26 14 13 30 17 
```

The blanks don't come through here as missing values, but instead look like another category called "blank." While that might work if we want to create a new category to deal with missingness, we probably want to first convert the variable so R recognizes the missingness.

```
demodata$return.original <- demodata$return
is.na(demodata$return) <- demodata$return==""
demodata$return.f <- factor(demodata$return,
                            levels=c("A", "B", "C", "D"), exclude=NULL)
table(demodata$return.f, useNA="ifany")
```

```
   A    B    C    D <NA> 
  14   13   30   17   26 
```

```
demodata$return.f.i <- impute(demodata$return.f, "Missing")
table(demodata$return.f.i)
```

```
      A       B       C       D Missing 
     14      13      30      17      26 
```

Again, we could then create a series of indicator variables to represent the various categories, should we want them.

10.4 Working with `rotation`

`rotation` is either X, Y or Z. There are 4 missing values, coded in the `.csv` as `"Unknown"`.

```
table(demodata$rotation, useNA="ifany")
```

```
Unknown       X       Y       Z 
      4      23      47      26 
```

```
is.na(demodata$rotation) <- demodata$rotation=="Unknown"

demodata$rotation.f <- factor(demodata$rotation, levels=c("X", "Y", "Z"), exclude=NULL)
demodata$rotation.f.i <- impute(demodata$rotation.f, fun="random")
```

```r
table(demodata$rotation.f, demodata$rotation.f.i, useNA="ifany")
```

```
        X  Y  Z
  X    23  0  0
  Y     0 47  0
  Z     0  0 26
  <NA>  2  0  2
```

Once again, we could create indicator variables to represent the various categories, should we want them.

10.5 Working with reason

`reason` can take on 12 different values for primary reason why the subject did not go to the doctor.

```r
table(demodata$reason, useNA="ifany")
```

```
  anxiety      costly   expensive        fear high priced     no time
        5           7          22          15           4          13
    panic     swamped     tied up    too busy      unease       worry
        4           6           8           7           4           5
```

The `reason` variable has no missing values, but we might want to collapse the reasons into three groups, perhaps combining the several reasons pertaining to fear into one category, the reasons related to cost into another category, and reasons related to time into a third category.

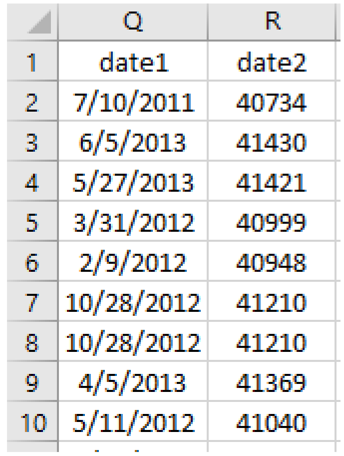
Suppose your desired combination was as follows:

Old Reason (12 categories)	New Reason (3 categories)
anxiety, fear, panic, unease, worry	fear
costly, expensive, high priced	cost
no time, swamped, tied up, too busy	time

So, we'll build a new factor that includes only our three new categories. This is a little tricky: first we create a new variable with no data in it, but only including the three new categories.

```r
demodata$reason3.f <- factor(rep(NA, length(demodata$reason) ), levels=c("fear", "cost"
table(demodata$reason3.f, useNA="ifany")
```

```
fear cost time <NA>
```



	Q	R
1	date1	date2
2	7/10/2011	40734
3	6/5/2013	41430
4	5/27/2013	41421
5	3/31/2012	40999
6	2/9/2012	40948
7	10/28/2012	41210
8	10/28/2012	41210
9	4/5/2013	41369
10	5/11/2012	41040

Figure 7: sheet4.png

```
   0    0    0  100
```

Next, we fill in the `fear` values, followed by the `cost` and then `time` values until our variable is completed, with no remaining `NA`s.

```
demodata$reason3.f[demodata$reason %in% c("anxiety", "fear", "panic", "unease", "worry")
table(demodata$reason3.f, useNA="ifany")
```

```
fear cost time <NA>
  33    0    0   67
```

```
demodata$reason3.f[demodata$reason %in% c("costly", "expensive", "high priced")] <- "cos
demodata$reason3.f[demodata$reason %in% c("no time", "swamped", "tied up", "too busy")]
table(demodata$reason3.f, useNA="ifany")
```

```
fear cost time
  33   33   34
```

11 Date Variables

If you've got a `.csv` file that was built in Excel, there are three likely data formats for dates that you'll see, as demonstrated in the `date1` and `date2` variables.

Neither import well into R. `date1` produces an unordered factor, and `date2` just produces a set of integers.

```
str(demodata[c("date1", "date2")])
```

```
'data.frame':   100 obs. of  2 variables:
```

```
 $ date1: Factor w/ 95 levels "1/12/2011","1/13/2012",..: 73 70 61 41 35 13 13 51 52 68
 $ date2: int  40734 41430 41421 40999 40948 41210 41210 41369 41040 40722 ...
```

11.1 The date format in Excel yields date1

The `date1` approach is obtained using the date format in Excel, and is fine for humans to read, even in R, but R still has no idea how to use it, interpreting it as a factor. The data are provided in month/day/4-digit year format. In order to get R to treat this as a date, we use the following...

```
demodata$date1.fix <- as.Date(demodata$date1, "%m/%d/%Y")
```

The command includes a capital Y since the data include all 4 digits of the year.

```
str(demodata$date1.fix)
```

```
 Date[1:100], format: "2011-07-10" "2013-06-05" "2013-05-27" "2012-03-31" "2012-02-09" .
```

```
summary(demodata$date1.fix)
```

```
        Min.      1st Qu.       Median         Mean      3rd Qu.
"2010-12-29" "2011-09-19" "2012-05-11" "2012-05-26" "2012-12-04"
        Max.
"2013-12-09"
```

11.2 The general format in Excel yields date2

For `date2`, which contains **exactly** the same data as `date1`, but using the general format in Excel, R just sees an integer. But what Excel is actually trying to represent is "days since 12/31/1899" so that 1 = January 1, 1900. This isn't too useful for a computer or a human, although you can at least calculate differences between two dates in terms of number of days with such an approach. Another problem is that Excel's function for doing this believes that 1900 was a leap year. So, to account for this, we use the following approach to build a date.

```
demodata$date2.fix <- as.Date(demodata$date2, origin="1899-12-30")
str(demodata$date2.fix)
```

```
 Date[1:100], format: "2011-07-10" "2013-06-05" "2013-05-27" "2012-03-31" "2012-02-09" .
```

```
summary(demodata$date2.fix)
```

```
        Min.      1st Qu.       Median         Mean      3rd Qu.
"2010-12-29" "2011-09-19" "2012-05-11" "2012-05-26" "2012-12-04"
        Max.
"2013-12-09"
```