Basic R Materials for 432 and 500

Thomas E. Love

2022-01-06

Contents

0.1 Load R packages

```
library(Hmisc)
library(Epi)
library(car)
library(naniar)
library(simputation)
library(broom)
library(magrittr); library(janitor); library(here)
library(tidyverse)

theme_set(theme_bw()) # I like the theme_bw() theme for my plots
```

0.2 Read in Three Data Sets

```
dm401 <- read_csv(here("data", "dm401.csv")) %>%
    type.convert() # converts characters to factors
```

```
Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE
```

```
Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE
```

```
demodata <- read_csv(here("data", "demodata.csv")) %>%
    type.convert() # converts characters to factors
```

```
Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE
```

```
Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE
```

```r
sep <- read_csv(here("data", "sep.csv")) %>%
    type.convert() # converts characters to factors
```

```
Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE

Warning in type.convert.default(x[[i]], ...): 'as.is' should be specified by the
caller; using TRUE
```

> 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 software page on our web site for some detailed instructions on getting R and R Studio onto your computer. Some tips for using RStudio and, in particular, R Markdown are in the last section of this document.

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...

	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

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 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.

	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

```
dm401
```

```
# A tibble: 146 x 12
   pt.id insurance    a1c   ldl   sbp eyexm pnvax   age   bmi raceeth female
   <int> <chr>      <dbl> <int> <int> <chr> <chr> <int> <dbl> <chr>   <chr>
 1     1 Medicaid     6.1   124   160 no    no       66  46.9 Black   female
 2     2 Commercial   6.9   187   162 yes   yes      57  43   White   female
 3     3 Uninsured    8.9   113   158 no    no       54  37.3 White   female
 4     4 Uninsured    7.7    64   140 no    no       49  40.9 Black   female
 5     5 Uninsured   11.3   133   153 no    yes      52  32.2 Black   female
 6     6 Uninsured    9.6   156   100 no    no       39  39.8 White   female
 7     7 Uninsured    6.2   162   114 no    yes      51  36.0 Black   female
 8     8 Uninsured    7.2   112   150 yes   no       51  40.2 Black   female
 9     9 Commercial   7.1    88   124 no    yes      68  28.3 White   female
10    10 Commercial   5.2   142   132 no    no       62  28.3 Black   female
# ... with 136 more rows, and 1 more variable: smoking <chr>
```

```
summary(dm401)
```

```
     pt.id          insurance              a1c              ldl
 Min.   :  1.00   Length:146         Min.   : 4.700   Min.   : 16
 1st Qu.: 37.25   Class :character   1st Qu.: 6.125   1st Qu.: 91
 Median : 73.50   Mode  :character   Median : 7.100   Median :113
 Mean   : 73.50                      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
 Min.   : 84.0   Length:146         Length:146         Min.   :23.0
 1st Qu.:120.0   Class :character   Class :character   1st Qu.:48.0
 Median :131.0   Mode  :character   Mode  :character   Median :57.0
 Mean   :135.4                                         Mean   :57.4
 3rd Qu.:149.5                                         3rd Qu.:67.0
 Max.   :213.0                                         Max.   :93.0
```

```
      bmi          raceeth             female            smoking
 Min.   :16.62   Length:146         Length:146         Length:146
 1st Qu.:28.48   Class :character   Class :character   Class :character
 Median :33.44   Mode  :character   Mode  :character   Mode  :character
 Mean   :34.13
 3rd Qu.:38.71
 Max.   :65.77
```

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.

```
str(dm401)
```

```
spec_tbl_df [146 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ pt.id    : int [1:146] 1 2 3 4 5 6 7 8 9 10 ...
 $ insurance: chr [1:146] "Medicaid" "Commercial" "Uninsured" "Uninsured" ...
 $ a1c      : num [1:146] 6.1 6.9 8.9 7.7 11.3 9.6 6.2 7.2 7.1 5.2 ...
 $ ldl      : int [1:146] 124 187 113 64 133 156 162 112 88 142 ...
 $ sbp      : int [1:146] 160 162 158 140 153 100 114 150 124 132 ...
 $ eyexm    : chr [1:146] "no" "yes" "no" "no" ...
 $ pnvax    : chr [1:146] "no" "yes" "no" "no" ...
 $ age      : int [1:146] 66 57 54 49 52 39 51 51 68 62 ...
 $ bmi      : num [1:146] 46.9 43 37.3 40.9 32.2 ...
 $ raceeth  : chr [1:146] "Black" "White" "White" "Black" ...
 $ female   : chr [1:146] "female" "female" "female" "female" ...
 $ smoking  : chr [1:146] "nonsmoker" "nonsmoker" "nonsmoker" "nonsmoker" ...
 - attr(*, "spec")=
  .. cols(
  ..   pt.id = col_double(),
  ..   insurance = col_character(),
  ..   a1c = col_double(),
  ..   ldl = col_double(),
```

```
  ..   sbp = col_double(),
  ..   eyexm = col_character(),
  ..   pnvax = col_character(),
  ..   age = col_double(),
  ..   bmi = col_double(),
  ..   raceeth = col_character(),
  ..   female = col_character(),
  ..   smoking = col_character()
  .. )
 - attr(*, "problems")=<externalptr>
```

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?

```
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?

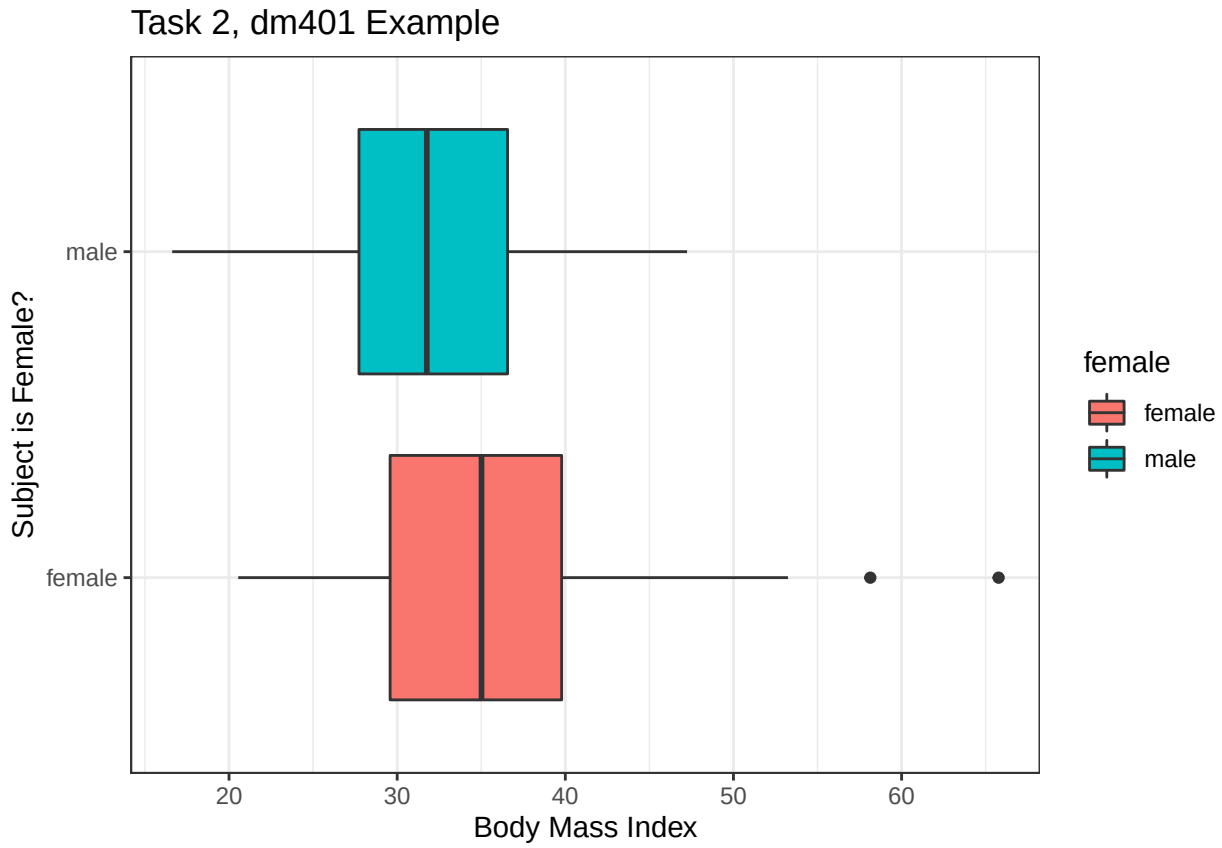
I'll start here by re-creating the `bootdif` function, useful for building bootstrap confidence intervals for the population mean difference using independent samples.

```
`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
  }
```

Let's consider a boxplot of the data.

```
ggplot(dm401, aes(x = female, y = bmi, fill = female)) +
    geom_boxplot() +
    coord_flip() +
    labs(x = "Subject is Female?",
         y = "Body Mass Index",
         title = "Task 2, dm401 Example")
```



Now, we'll run some numerical comparisons, to obtain uncertainty intervals for the difference in average BMI across the two groups...

```
dm401 %>% group_by(female) %>%
    summarize(n = n(), mean(bmi), median(bmi), sd(bmi))
```

```
# A tibble: 2 x 5
  female     n `mean(bmi)` `median(bmi)` `sd(bmi)`
  <chr>  <int>       <dbl>         <dbl>     <dbl>
1 female    82        35.6          35.0      8.32
```

```
2 male      64        32.3          31.8      6.59
```

```
dm401 %$%
  mosaic::favstats(bmi ~ female)
```

```
Registered S3 method overwritten by 'mosaic':
  method                           from
  fortify.SpatialPolygonsDataFrame ggplot2
```

```
  female   min      Q1 median      Q3   max     mean       sd  n missing
1 female 20.55 29.5800 35.015 39.7825 65.77 35.56866 8.319085 82       0
2   male 16.62 27.7275 31.760 36.5650 47.24 32.29797 6.590426 64       0
```

```
dm401 %$% 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 between group female and group male is
95 percent confidence interval:
 0.8316821 5.7096975
sample estimates:
mean in group female   mean in group male
            35.56866             32.29797
```

```
dm401 %$% bootdif(bmi, female)
```

```
Mean Difference           0.025           0.975
     -3.2706898      -5.7610328      -0.9808506
```

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.

```
`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)
  }
```

```
dm401 %>%
    tabyl(smoking, eyexm)
```

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

I'd rather change the ordering of the levels of those factors so that the table yielded Non-Smokers with Eye Exams in the top left.

```
dm401 <- dm401 %>%
    mutate(eyexm = fct_relevel(eyexm, "yes", "no"))

dm401 %>%
    tabyl(smoking, eyexm)
```

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

That's better. Now we can use our `twobytwo` function if we like to obtain uncertainty intervals...

```
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
------------------------------------------------------
```

Another option would be:

```
dm401 %$% table(smoking, eyexm) %>% Epi::twoby2()
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : yes
Comparing : nonsmoker vs. smoker

          yes no    P(yes) 95% conf. interval
nonsmoker  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?

```
dm401 %$% table(insurance, pnvax)
```

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

```
dm401 %$% table(insurance, pnvax) %>% chisq.test()
```

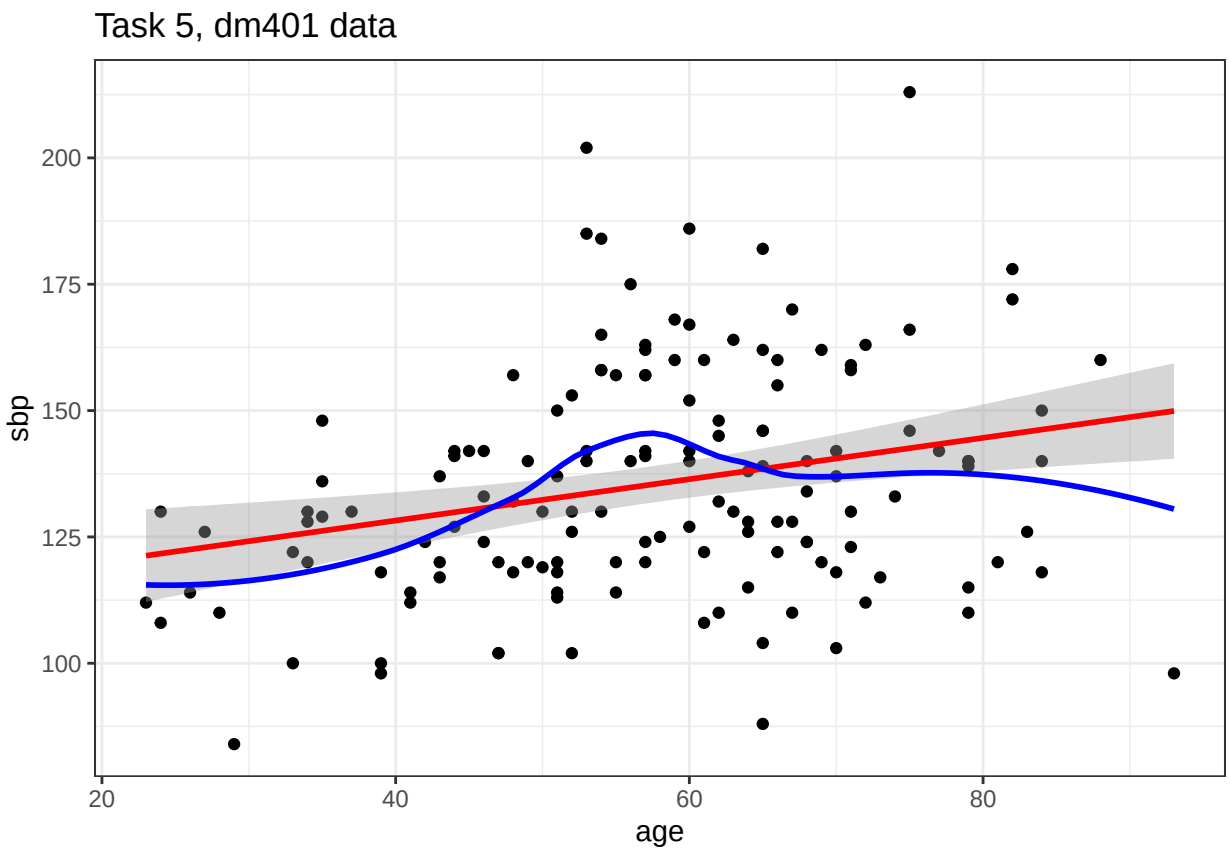
```
    Pearson's Chi-squared test

data:  .
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?

```
ggplot(dm401, aes(x = age, y = sbp)) +
    geom_point() +
    geom_smooth(method = "lm", col = "red") +
    geom_smooth(method = "loess", se = FALSE, col = "blue") +
    labs(title = "Task 5, dm401 data")
```

```
`geom_smooth()` using formula 'y ~ x'
`geom_smooth()` using formula 'y ~ x'
```




```
m5 <- lm(sbp ~ age, data = dm401)
summary(m5)
```

```
Call:
lm(formula = sbp ~ age, data = dm401)

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

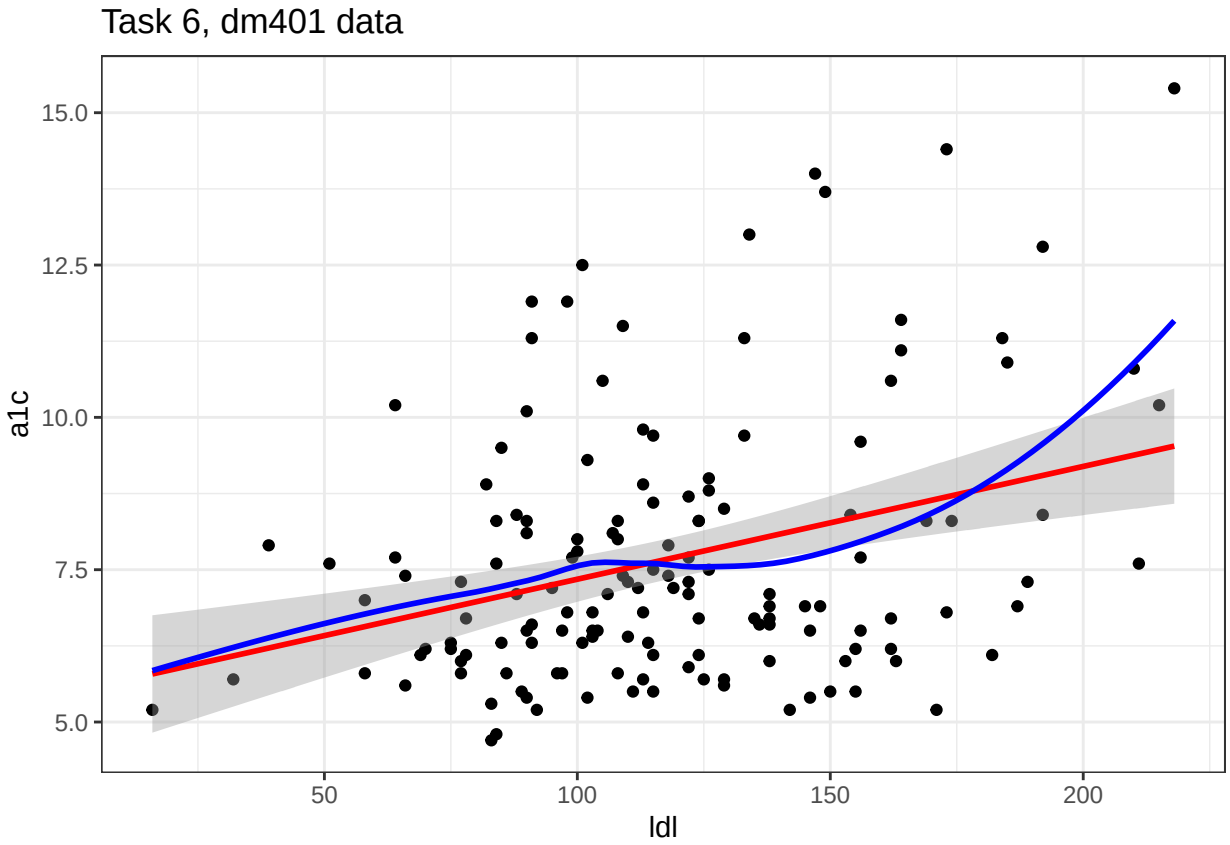
```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 111.8782     7.3519  15.218  < 2e-16 ***
age           0.4090     0.1241   3.296  0.00123 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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?)

```
ggplot(dm401, aes(x = ldl, y = a1c)) +
    geom_point() +
    geom_smooth(method = "lm", col = "red") +
    geom_smooth(method = "loess", se = FALSE, col = "blue") +
    labs(title = "Task 6, dm401 data")
```

```
`geom_smooth()` using formula 'y ~ x'
`geom_smooth()` using formula 'y ~ x'
```




```
m6 <- lm(a1c ~ ldl, data = dm401)
summary(m6)
```

```
Call:
lm(formula = a1c ~ ldl, data = dm401)

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?

```
summary(lm(a1c ~ insurance, data = dm401))
```

```
Call:
lm(formula = a1c ~ insurance, data = dm401)

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
```

```
summary(lm(a1c ~ raceeth, data = dm401))
```

```
Call:
lm(formula = a1c ~ raceeth, data = dm401)

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
```

```
summary(lm(a1c ~ insurance + raceeth, data = dm401))
```

```
Call:
lm(formula = a1c ~ insurance + raceeth, data = dm401)

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
```

```
dm401 %>% tabyl(raceeth)
```

```
  raceeth  n    percent
    Black 72 0.49315068
 Hispanic 10 0.06849315
    White 64 0.43835616
```

```
summary(lm(a1c ~ raceeth=="White", data = dm401))
```

```
Call:
lm(formula = a1c ~ raceeth == "White", data = dm401)

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
```

```
summary(lm(a1c ~ insurance + (raceeth=="White"), data = dm401))
```

```
Call:
lm(formula = a1c ~ insurance + (raceeth == "White"), data = dm401)

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
```

```
summary(lm(a1c ~ insurance * (raceeth=="White"), data = dm401))
```

```
Call:
lm(formula = a1c ~ insurance * (raceeth == "White"), data = dm401)

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

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                                7.63077    0.59136  12.904   <2e-16
insuranceMedicaid                         -0.44656    0.76745  -0.582    0.562
insuranceMedicare                          0.03352    0.71559   0.047    0.963
insuranceUninsured                         0.71923    0.74589   0.964    0.337
raceeth == "White"TRUE                    -0.67692    0.72427  -0.935    0.352
insuranceMedicaid:raceeth == "White"TRUE   0.34271    1.23351   0.278    0.782
```

```
insuranceMedicare:raceeth == "White"TRUE   1.15847    0.93614   1.237    0.218
insuranceUninsured:raceeth == "White"TRUE  1.01442    1.13995   0.890    0.375

(Intercept)                                    ***
insuranceMedicaid
insuranceMedicare
insuranceUninsured
raceeth == "White"TRUE
insuranceMedicaid:raceeth == "White"TRUE
insuranceMedicare:raceeth == "White"TRUE
insuranceUninsured:raceeth == "White"TRUE
---
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?

```
summary(lm(a1c ~ insurance, data = dm401))

Call:
lm(formula = a1c ~ insurance, data = dm401)

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
```

```
summary(lm(a1c ~ insurance + ldl, data = dm401))
```

```
Call:
lm(formula = a1c ~ insurance + ldl, data = dm401)

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.

```
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, data = dm401)

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)         4.496817   1.624234   2.769  0.00644 **
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 *
eyexmno             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
```

```
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, data = dm401)

Coefficients:
       (Intercept)                ldl   insuranceMedicaid   insuranceMedicare
           4.36156            0.01806             0.33631             0.88307
insuranceUninsured            eyexmno
           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.

```
summary(lm(a1c ~ ldl + insurance + eyexm, data = dm401))
```

```
Call:
lm(formula = a1c ~ ldl + insurance + eyexm, data = dm401)

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)        4.361561   0.680584   6.409 2.09e-09 ***
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 **
eyexmno            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
```

```
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, data = dm401)

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)        4.496817   1.624234   2.769  0.00644 **
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 *
eyexmno             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
```

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.

```
summary(sep)
```

```
     pt.id         treatment             race               apache
 Min.   :  1.00   Length:350         Length:350         Min.   : 0.00
```

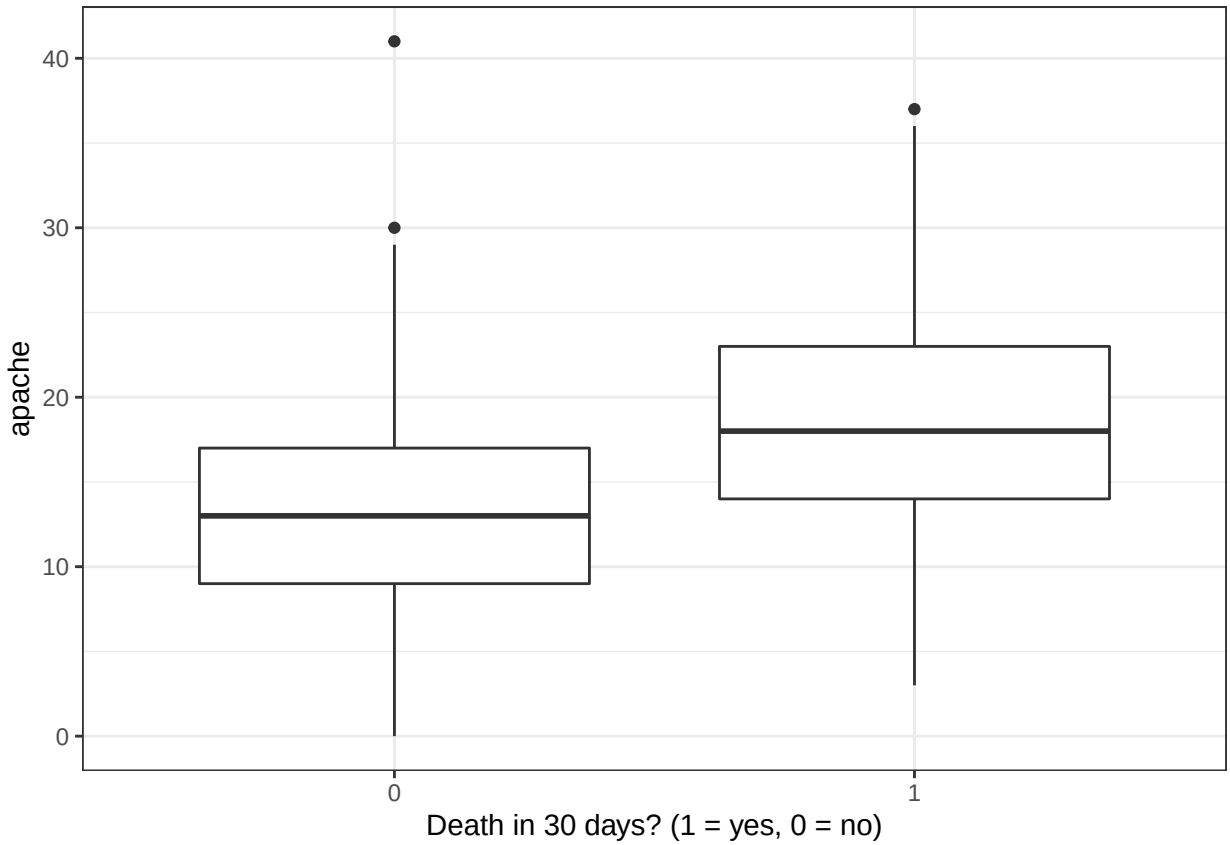
```
 1st Qu.: 88.25   Class :character   Class :character   1st Qu.:11.00
 Median :175.50   Mode  :character   Mode  :character   Median :15.00
 Mean   :175.50                                        Mean   :15.74
 3rd Qu.:262.75                                        3rd Qu.:21.00
 Max.   :350.00                                        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?

```
ggplot(sep, aes(x = factor(death30d), y = apache)) +
  geom_boxplot() +
  labs(x = "Death in 30 days? (1 = yes, 0 = no)")
```



plot1-1.pdf

```
sep %$%
  mosaic::favstats(apache ~ death30d)
```

```
  death30d min Q1 median Q3 max     mean       sd   n missing
1        0   0  9     13 17  41 13.64319 6.492696 213       0
2        1   3 14     18 23  37 19.00000 7.158911 137       0
```

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, data = sep))
```

```
Call:
lm(formula = apache ~ death30d, data = sep)

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 ***
```

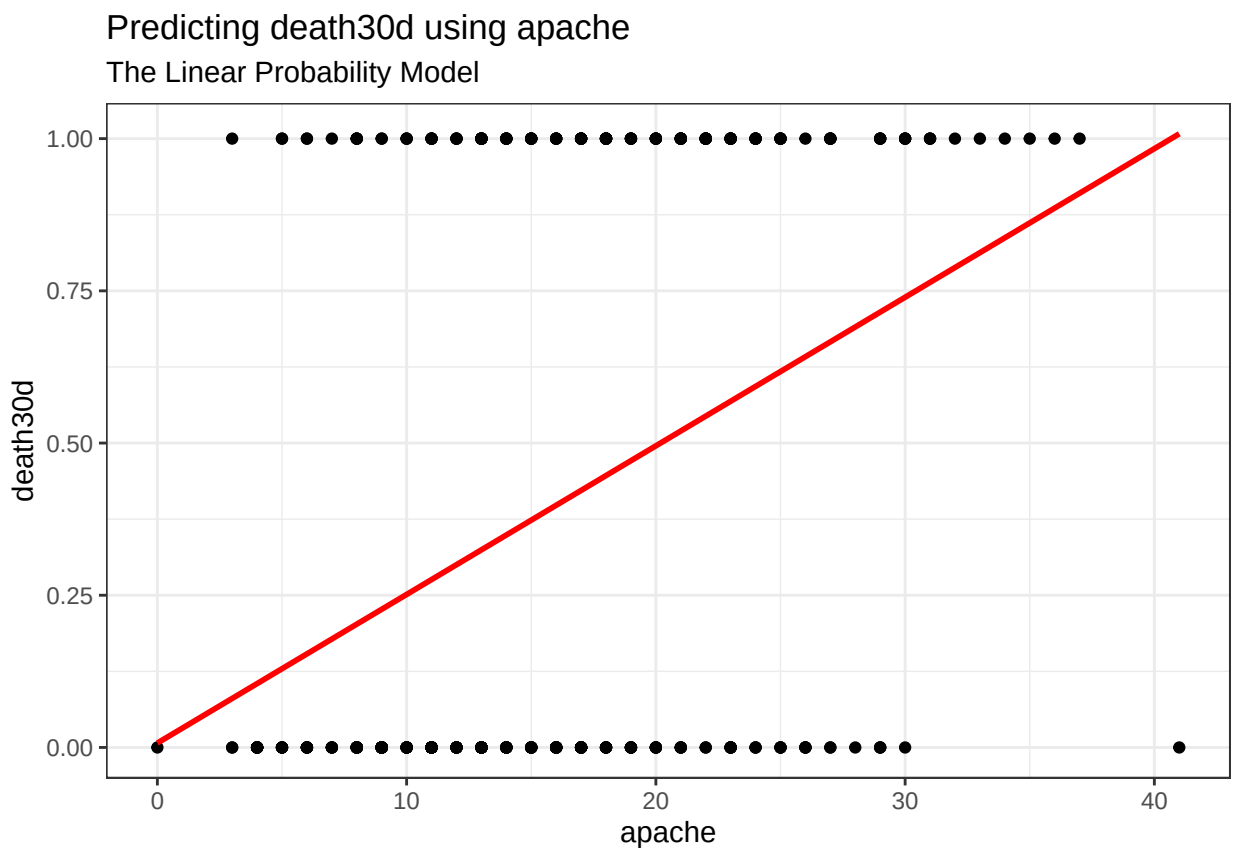
```
death30d      5.3568     0.7404   7.235    3e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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.

```
ggplot(sep, aes(x = apache, y = death30d)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red", se = FALSE) +
  labs(title = "Predicting death30d using apache",
       subtitle = "The Linear Probability Model")
```

```
`geom_smooth()` using formula 'y ~ x'
```

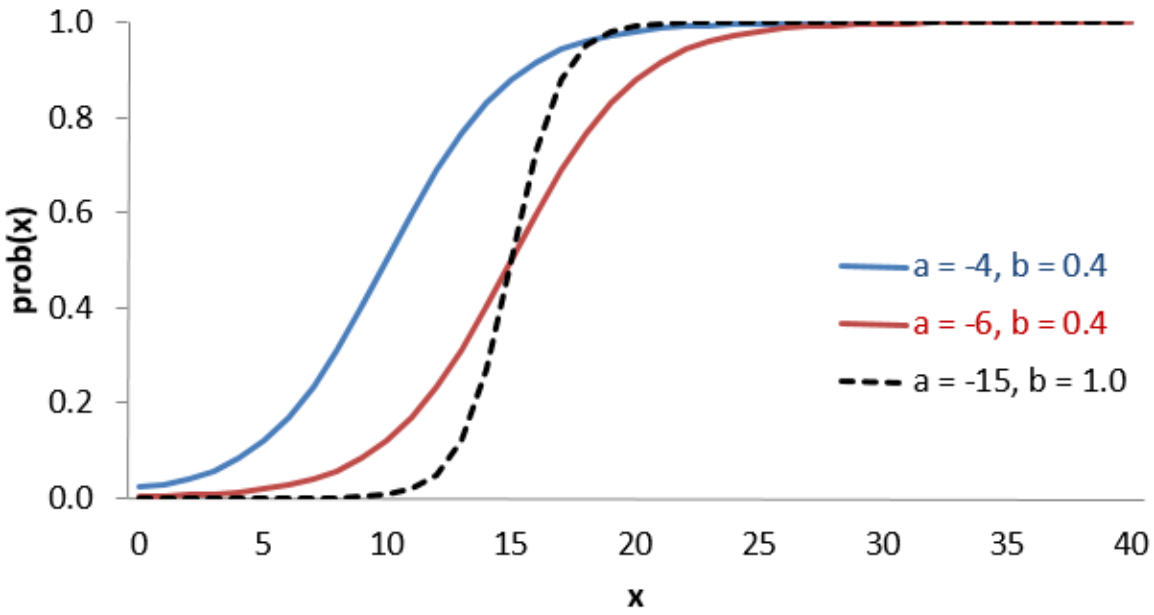


5.3 The Logistic Regression Model

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:

```
sep_m1 <- glm(death30d ~ apache,
              family = binomial(), data = sep)

summary(sep_m1)
```

```
Call:
glm(formula = death30d ~ apache, family = binomial(), data = sep)

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, and the second has a score of 14.

```
sep %>% slice(1:2)
```

```
# A tibble: 2 x 5
  pt.id treatment race  apache death30d
  <int> <chr>     <chr>  <int>    <int>
1     1 placebo   W         27        1
2     2 ibuprofen AA        14        0
```

While we know that patient 1 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.

```
fitted(sep_m1)[1:2]
```

```
        1         2
0.6861055 0.3347359
```

Or, we can use the `augment` function from the `broom` package to get even more information...

```
augment(sep_m1, type.predict = "response") %>% slice(1:2)
```

```
# A tibble: 2 x 8
  death30d apache .fitted .resid .std.resid    .hat .sigma  .cooksd
     <int>  <int>   <dbl>  <dbl>      <dbl>   <dbl>  <dbl>    <dbl>
1        1     27   0.686  0.868      0.873 0.0106    1.10 0.00249
2        0     14   0.335 -0.903     -0.904 0.00349   1.10 0.000885
```

To get the linear predictions, we can use either:

```
augment(sep_m1) %>% slice(1:2) # on linear scale
```

```
# A tibble: 2 x 8
  death30d apache .fitted .resid .std.resid    .hat .sigma  .cooksd
     <int>  <int>   <dbl>  <dbl>      <dbl>   <dbl>  <dbl>    <dbl>
1        1     27   0.782  0.868      0.873 0.0106    1.10 0.00249
2        0     14  -0.687 -0.903     -0.904 0.00349   1.10 0.000885
```

or

```
sep_m1$linear.predictors[1:2]
```

```
         1          2
 0.7819742 -0.6868423
```

5.6 Interpreting the Logistic Regression Model Summary

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

```
summary(sep_m1)
```

```
Call:
glm(formula = death30d ~ apache, family = binomial(), data = sep)

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)...

```
sep_m1 %$% car::Anova(., 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?

```
sep_m2 <- glm(death30d ~ apache + treatment,
              family=binomial(), data = sep)

summary(sep_m2)
```

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

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?

```
sep_m3 <- glm(death30d ~ apache + treatment + race,
              family=binomial(), data = sep)

summary(sep_m3)
```

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

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 or the linear predictors for the first two patients, using this model, as follows.

```
augment(sep_m3, type.predict = "response") %>% slice(1:2)
```

```
# A tibble: 2 x 10
  death30d apache treatment race  .fitted .resid .std.resid   .hat .sigma
     <int>  <int> <chr>     <chr>   <dbl>  <dbl>      <dbl>  <dbl>  <dbl>
1        1     27 placebo   W       0.700  0.845      0.852 0.0166   1.10
2        0     14 ibuprofen AA      0.334 -0.902     -0.909 0.0139   1.10
# ... with 1 more variable: .cooksd <dbl>
```

```
augment(sep_m3) %>% slice(1:2) # on linear scale
```

```
# A tibble: 2 x 10
  death30d apache treatment race  .fitted .resid .std.resid   .hat .sigma
     <int>  <int> <chr>     <chr>   <dbl>  <dbl>      <dbl>  <dbl>  <dbl>
1        1     27 placebo   W       0.846  0.845      0.852 0.0166   1.10
2        0     14 ibuprofen AA     -0.688 -0.902     -0.909 0.0139   1.10
# ... with 1 more variable: .cooksd <dbl>
```

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

```
sep_m3$fitted[1:2]
```

```
        1         2
0.6998077 0.3344952
```

```
sep_m3$linear.predictors[1:2]
```

```
         1          2
 0.8463821 -0.6879231
```

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. 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.

```
str(demodata)
```

```
spec_tbl_df [100 x 18] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ Subject : int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
 $ age     : int [1:100] 55 52 51 19 51 39 37 35 55 32 ...
 $ test1   : int [1:100] 36 59 30 80 73 45 32 -999 62 40 ...
 $ test2   : int [1:100] 267 252 221 136 184 NA 134 166 227 154 ...
 $ test3   : int [1:100] 27 NA 16 NA NA 30 NA 18 45 NA ...
 $ histA   : chr [1:100] "Yes" "No" "Yes" "Yes" ...
 $ histB   : int [1:100] 2 2 2 1 1 1 2 2 2 2 ...
 $ histC   : int [1:100] 0 0 1 0 0 1 1 1 1 0 ...
 $ histD   : int [1:100] 1 0 0 0 0 0 NA 1 0 1 ...
 $ histE   : int [1:100] 1 0 NA NA NA 1 1 1 1 1 ...
 $ histF   : int [1:100] 1 0 1 99 0 77 1 0 0 0 ...
 $ race    : int [1:100] 4 4 1 3 1 2 3 4 3 2 ...
 $ rating  : chr [1:100] "Exc" "V Good" NA "Good" ...
 $ return  : chr [1:100] "B" "C" "D" "A" ...
 $ rotation: chr [1:100] "X" "Y" "Y" "Z" ...
```

```
$ reason  : chr [1:100] "expensive" "too busy" "high priced" "costly" ...
$ date1   : chr [1:100] "7/10/2011" "6/5/2013" "5/27/2013" "3/31/2012" ...
$ date2   : int [1:100] 40734 41430 41421 40999 40948 41210 41210 41369 41040 40722 ...
- attr(*, "spec")=
 .. cols(
 ..   Subject = col_double(),
 ..   age = col_double(),
 ..   test1 = col_double(),
 ..   test2 = col_double(),
 ..   test3 = col_double(),
 ..   histA = col_character(),
 ..   histB = col_double(),
 ..   histC = col_double(),
 ..   histD = col_double(),
 ..   histE = col_double(),
 ..   histF = col_double(),
 ..   race = col_double(),
 ..   rating = col_character(),
 ..   return = col_character(),
 ..   rotation = col_character(),
 ..   reason = col_character(),
 ..   date1 = col_character(),
 ..   date2 = col_double()
 .. )
- attr(*, "problems")=<externalptr>
```

7.1 A Quick Summary of the Data, as Initially Imported

```
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      
Min.   : 2.00   Length:100         Min.   :1.00   Min.   :0.00  
1st Qu.:10.00   Class :character   1st Qu.:1.00   1st Qu.:0.00  
Median :25.00   Mode  :character   Median :2.00   Median :0.00  
Mean   :24.26                      Mean   :1.52   Mean   :0.46  
```

```
3rd Qu.:38.00                    3rd Qu.:2.00   3rd Qu.:1.00
Max.   :48.00                    Max.   :2.00   Max.   :1.00
NA's   :57
    histD            histE            histF            race
Min.   :0.0000   Min.   :0.0000   Min.   : 0.00   Min.   :1.00
1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.00   1st Qu.:1.75
Median :1.0000   Median :0.0000   Median : 1.00   Median :3.00
Mean   :0.5532   Mean   :0.4932   Mean   : 7.93   Mean   :2.57
3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 1.00   3rd Qu.:4.00
Max.   :1.0000   Max.   :1.0000   Max.   :99.00   Max.   :4.00
NA's   :6        NA's   :27
   rating            return           rotation           reason
Length:100         Length:100         Length:100         Length:100
Class :character   Class :character   Class :character   Class :character
Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
   date1              date2
Length:100         Min.   :40541
Class :character   1st Qu.:40806
Mode  :character   Median :41040
                   Mean   :41055
                   3rd Qu.:41247
                   Max.   :41617
```

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.

	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

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
```

```
# A tibble: 100 x 18
   Subject   age test1 test2 test3 histA histB histC histD histE histF  race
     <int> <int> <int> <int> <int> <chr> <int> <int> <int> <int> <int> <int>
 1       1    55    36   267    27 Yes       2     0     1     1     1     4
 2       2    52    59   252    NA No        2     0     0     0     0     4
 3       3    51    30   221    16 Yes       2     1     0    NA     1     1
 4       4    19    80   136    NA Yes       1     0     0    NA    99     3
 5       5    51    73   184    NA Yes       1     0     0    NA     0     1
 6       6    39    45    NA    30 No        1     1     0     1    77     2
 7       7    37    32   134    NA Yes       2     1    NA     1     1     3
 8       8    35  -999   166    18 Yes       2     1     1     1     0     4
 9       9    55    62   227    45 No        2     1     0     1     0     3
10      10    32    40   154    NA Yes       2     0     1     1     0     2
# ... with 90 more rows, and 6 more variables: rating <chr>, return <chr>,
#   rotation <chr>, reason <chr>, date1 <chr>, date2 <int>
```

```
demodata %>% select(1:5) %>% summary()
```

```
    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
```

```
## summarizes the first five variables
```

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. . .

```
demodata <- demodata %>%
  mutate(test1 = na_if(test1, -999))

demodata %$% summary(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

This advice is strictly meant for building a propensity score model, and thus applies to 500 students. For those of you taking 500, here is one potential approach for imputing values for the missing observations in `test1`, `test2` and `test3`. We'll attack this in a different way in 432. The `na.pattern` function is part of the `Hmisc` package.

```
demodata %>% select(test1, test2, test3) %>% na.pattern()
```

```
pattern
000 001 010 011 100 101
```

```
 40  52   2   3   1   2
```

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.

One option would be to use the random hot deck imputation strategy as implemented in the `simputation` package to accomplish these simple imputations. 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.

```
set.seed(500001)

demodata_imp01 <- data.frame(demodata) %>%
    mutate(test1_imp = test1) %>%
    mutate(test2_imp = test2) %>%
   impute_rhd(., test1_imp ~ 1, pool="univariate") %>%
   impute_rhd(., test2_imp ~ 1, pool="univariate") %>%
  tbl_df()
```

```
Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
Please use `tibble::as_tibble()` instead.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
demodata_imp01 %>%
  select(test1, test1_imp, test2, test2_imp) %>% summary()
```

```
     test1          test1_imp         test2          test2_imp     
 Min.   :20.00   Min.   :20.00   Min.   :102.0   Min.   :102.0  
 1st Qu.:35.00   1st Qu.:34.50   1st Qu.:162.0   1st Qu.:162.0  
 Median :48.00   Median :48.00   Median :189.0   Median :189.0  
 Mean   :49.71   Mean   :49.25   Mean   :198.5   Mean   :198.1  
 3rd Qu.:66.00   3rd Qu.:65.25   3rd Qu.:243.0   3rd Qu.:242.5  
 Max.   :80.00   Max.   :80.00   Max.   :300.0   Max.   :300.0  
 NA's   :3                       NA's   :5                      
```

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`

```
set.seed(500001)

demodata_imp01 <- demodata %>%
    mutate(test1_imp = test1,
           test2_imp = test2,
           test3_imp = test3,
           test3_NA =
             as.numeric(is.na(test3)),
           test3_imp =
             replace_na(test3_imp,
                        median(test3, na.rm = T))) %>%
  data.frame() %>%
    impute_rhd(., test1_imp ~ 1, pool="univariate") %>%
    impute_rhd(., test2_imp ~ 1, pool="univariate") %>%
  tbl_df()

demodata_imp01 %>%
  select(test1, test1_imp,
         test2, test2_imp,
         test3, test3_imp, test3_NA) %>% summary()
```

```
     test1          test1_imp         test2          test2_imp
 Min.   :20.00   Min.   :20.00   Min.   :102.0   Min.   :102.0
 1st Qu.:35.00   1st Qu.:34.50   1st Qu.:162.0   1st Qu.:162.0
 Median :48.00   Median :48.00   Median :189.0   Median :189.0
 Mean   :49.71   Mean   :49.25   Mean   :198.5   Mean   :198.1
 3rd Qu.:66.00   3rd Qu.:65.25   3rd Qu.:243.0   3rd Qu.:242.5
 Max.   :80.00   Max.   :80.00   Max.   :300.0   Max.   :300.0
 NA's   :3                       NA's   :5
     test3          test3_imp        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
```

As an alternative, we might consider imputing `test3` more thoroughly, perhaps again with a hot deck...

```
set.seed(500002)

demodata_imp02 <- demodata %>%
    mutate(test1_imp = test1,
```

```
            test2_imp = test2,
            test3_imp = test3,
            test3_NA =
              as.numeric(is.na(test3))) %>%
  data.frame() %>%
   impute_rhd(., test1 ~ 1, pool="univariate") %>%
   impute_rhd(., test2 ~ 1, pool="univariate") %>%
   impute_rhd(., test3 ~ 1, pool="univariate") %>%
  tbl_df()

demodata_imp02 %>%
  select(test1, test1_imp, test2, test2_imp,
         test3, test3_imp, test3_NA) %>% summary()
```

```
     test1          test1_imp         test2          test2_imp
 Min.   :20.00   Min.   :20.00   Min.   :102.0   Min.   :102.0
 1st Qu.:35.00   1st Qu.:35.00   1st Qu.:161.8   1st Qu.:162.0
 Median :49.00   Median :48.00   Median :189.0   Median :189.0
 Mean   :49.71   Mean   :49.71   Mean   :198.2   Mean   :198.5
 3rd Qu.:66.25   3rd Qu.:66.00   3rd Qu.:244.5   3rd Qu.:243.0
 Max.   :80.00   Max.   :80.00   Max.   :300.0   Max.   :300.0
                 NA's   :3                       NA's   :5
     test3          test3_imp        test3_NA
 Min.   : 2.00   Min.   : 2.00   Min.   :0.00
 1st Qu.: 7.00   1st Qu.:10.00   1st Qu.:0.00
 Median :25.00   Median :25.00   Median :1.00
 Mean   :24.44   Mean   :24.26   Mean   :0.57
 3rd Qu.:39.00   3rd Qu.:38.00   3rd Qu.:1.00
 Max.   :48.00   Max.   :48.00   Max.   :1.00
                 NA's   :57
```

In a propensity score setting, we'd use the post-imputation values `test1_imp`, `test2_imp`, and `test3_imp` along with `test3_NA` in our propensity score 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 <- demodata %>%
  mutate(age_50plus = as.numeric(age >= 50))
```

```
demodata %$%
  mosaic::favstats(age ~ age_50plus) # sanity check on recoding
```

```
  age_50plus min   Q1 median Q3 max     mean       sd  n missing
1          0  19 28.0     33 42  49 34.59184 8.746233 49       0
2          1  50 55.5     60 66  75 61.33333 7.325754 51       0
```

Instead of a 1/0 variable for `age_50plus` we could instead create a factor called `age_cat`.

```
demodata <- demodata %>%
  mutate(age_cat = fct_recode(factor(age >= 50),
                              Older = "TRUE",
                              Below50 = "FALSE"))

demodata %$%
  mosaic::favstats(age ~ age_50plus) # sanity check on recoding
```

```
  age_50plus min   Q1 median Q3 max     mean       sd  n missing
1          0  19 28.0     33 42  49 34.59184 8.746233 49       0
2          1  50 55.5     60 66  75 61.33333 7.325754 51       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)...

```
demodata <- demodata %>%
  mutate(age_group = cut2(age, g = 4))

demodata %$%
  mosaic::favstats(age ~ age_group)
```

```
  age_group min Q1 median Q3 max  mean       sd  n missing
1   [19,34)  19 24     28 31  33 27.28 4.286801 25       0
2   [34,51)  34 40     45 46  50 42.52 4.831494 25       0
3   [51,61)  51 54     56 58  60 55.64 2.782086 25       0
4   [61,75]  61 64     67 72  75 67.48 5.058985 25       0
```

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 <- demodata %>%
  mutate(age_grp = cut2(age, cuts = c(35, 50, 65)))
```

```
demodata %$%
  mosaic::favstats(age ~ age_grp)
```

```
  age_grp min    Q1 median Q3 max     mean       sd  n missing
1 [19,35)  19 24.75     29 32  34 28.00000 4.562326 28       0
2 [35,50)  35 41.00     45 46  49 43.38095 3.680709 21       0
3 [50,65)  50 54.00     58 61  64 57.38889 4.121565 36       0
4 [65,75]  65 68.50     71 74  75 70.80000 3.629246 15       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.

	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

- 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.

```
demodata %>%
  select(histA:histF) %>%
  summary()
```

```
    histA               histB          histC          histD
```

```
 Length:100         Min.   :1.00   Min.   :0.00   Min.   :0.0000
 Class :character   1st Qu.:1.00   1st Qu.:0.00   1st Qu.:0.0000
 Mode  :character   Median :2.00   Median :0.00   Median :1.0000
                    Mean   :1.52   Mean   :0.46   Mean   :0.5532
                    3rd Qu.:2.00   3rd Qu.:1.00   3rd Qu.:1.0000
                    Max.   :2.00   Max.   :1.00   Max.   :1.0000
                                                  NA's   :6
    histE            histF
 Min.   :0.0000   Min.   : 0.00
 1st Qu.:0.0000   1st Qu.: 0.00
 Median :0.0000   Median : 1.00
 Mean   :0.4932   Mean   : 7.93
 3rd Qu.:1.0000   3rd Qu.: 1.00
 Max.   :1.0000   Max.   :99.00
 NA's   :27
```

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`

```
demodata %>% tabyl(histA)
```

```
 histA  n percent
    No 54    0.54
   Yes 46    0.46
```

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`.

```
demodata <- demodata %>%
  mutate(histA_num = as.numeric(histA == "Yes"))

demodata %>% count(histA, histA_num)
```

```
# A tibble: 2 x 3
  histA histA_num     n
  <chr>     <dbl> <int>
```

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

9.1.2 Converting histB

```
demodata %>% tabyl(histB)
```

```
 histB  n percent
     1 48    0.48
     2 52    0.52
```

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 <- demodata %>%
  mutate(histB_original = histB,
         histB_num = 2 - histB,
         histB_fac = fct_recode(factor(histB == "1"),
                                Yes = "TRUE",
                                No = "FALSE"))

demodata %>% count(histB_original, histB_num, histB_fac)
```

```
# A tibble: 2 x 4
  histB_original histB_num histB_fac     n
           <int>     <dbl> <fct>     <int>
1              1         1 Yes          48
2              2         0 No           52
```

9.1.3 Converting histC

```
demodata %>% tabyl(histC)
```

```
 histC  n percent
     0 54    0.54
     1 46    0.46
```

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:

```
demodata <- demodata %>%
  mutate(histC_fac = fct_recode(factor(histC == "1"),
                                Yes = "TRUE",
```

```
                                  No = "FALSE"))

demodata %>% count(histC_fac, histC)
```

```
# A tibble: 2 x 3
  histC_fac histC     n
  <fct>     <int> <int>
1 No            0    54
2 Yes           1    46
```

If for some reason we wanted to rearrange the levels of the factor to (Yes, No) instead of (No, Yes) we would use `fct_relevel` to do so.

```
demodata <- demodata %>%
  mutate(histC_fac = fct_recode(factor(histC == "1"),
                                Yes = "TRUE",
                                No = "FALSE"),
         histC_fac2 = fct_relevel(histC_fac, "Yes"))

demodata %>% tabyl(histC_fac)
```

```
 histC_fac  n percent
        No 54    0.54
       Yes 46    0.46
```

```
demodata %>% tabyl(histC_fac2)
```

```
 histC_fac2  n percent
        Yes 46    0.46
         No 54    0.54
```

9.2 Dealing with Missingness in Binary Data

Now, we'll deal with missingness, in binary data, as shown in `histD`, `histE` and `histF`. Again, this advice is strictly meant for building a propensity score model, and thus applies only to students working on project or assignments for 500.

```
demodata %>% select(histD, histE, histF) %>% na.pattern()
```

```
pattern
000 010 100 110
 70  24   3   3
```

9.2.1 Imputation for `histD` for building a propensity model.

```
demodata %>% tabyl(histD)
```

```
 histD  n percent valid_percent
     0 42    0.42     0.4468085
     1 52    0.52     0.5531915
    NA  6    0.06            NA
```

In `histD`, we have a 1-0 numeric variable, and R recognizes 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(500003)

demodata_imp03 <- demodata %>%
  mutate(histD_imp = histD) %>%
  data.frame() %>%
   impute_rhd(., histD_imp ~ 1, pool="univariate") %>%
  tbl_df()

demodata_imp03 %>% count(histD, histD_imp)
```

```
# A tibble: 4 x 3
  histD histD_imp     n
  <int>     <int> <int>
1     0         0    42
2     1         1    52
3    NA         0     3
4    NA         1     3
```

9.2.2 Working with `histE`

```
demodata %>% tabyl(histE)
```

```
 histE  n percent valid_percent
     0 37    0.37     0.5068493
     1 36    0.36     0.4931507
    NA 27    0.27            NA
```

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`). Then, we'll create a factor called `histE.f` with three levels: Yes, No and Missing. Finally, we'll do a simple imputation of the same value for each of the 27 missing values.

```
set.seed(500004)

demodata_imp04 <- demodata %>%
    mutate(histE_orig = histE,
           histE_imp = histE,
           histE_na =
             as.numeric(is.na(histE)),
           histE_fac =
             fct_recode(factor(histE),
                        Yes = "1", No = "0"),
           histE_fac =
             fct_explicit_na(histE_fac,
                             na_level = "Missing")) %>%
  data.frame() %>%
   impute_rhd(., histE_imp ~ 1, pool="univariate") %>%
  tbl_df()
```

```
demodata_imp04 %>% count(histE_orig, histE_na, histE_fac, histE_imp)
```

```
# A tibble: 4 x 5
  histE_orig histE_na histE_fac histE_imp     n
       <int>    <dbl> <fct>         <int> <int>
1          0        0 No                0    37
2          1        0 Yes               1    36
3         NA        1 Missing           0    18
4         NA        1 Missing           1     9
```

9.2.3 Working with `histF`

```
demodata %>% tabyl(histF)
```

```
 histF  n percent
     0 47    0.47
     1 45    0.45
    77  1    0.01
    88  2    0.02
    99  5    0.05
```

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 variable called `histF_fix`. 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_imp`. Then, we'll create a factor called `histF_fac` with two levels: Yes and No, based on the imputed values in `histF_imp`.

```
set.seed(500005)

demodata_imp05 <- demodata %>%
    mutate(histF_orig = histF,
           histF_fix = na_if(histF, 77),
           histF_fix = na_if(histF_fix, 88),
           histF_fix = na_if(histF_fix, 99),
           histF_imp = histF_fix) %>%
  data.frame() %>%
    impute_rhd(., histF_imp ~ 1, pool="univariate") %>%
  tbl_df() %>%
    mutate(histF_fac = factor(histF_imp),
           histF_fac = fct_recode(histF_fac,
                                  Yes = "1",
                                  No = "0"))
```

```
demodata_imp05 %>% count(histF_orig, histF_fix, histF_imp, histF_fac)
```

```
# A tibble: 7 x 5
  histF_orig histF_fix histF_imp histF_fac     n
       <int>     <int>     <int> <fct>     <int>
1          0         0         0 No           47
2          1         1         1 Yes          45
3         77        NA         1 Yes           1
4         88        NA         0 No            1
5         88        NA         1 Yes           1
6         99        NA         0 No            2
7         99        NA         1 Yes           3
```

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.

	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

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.

```
demodata %>%
  select(race, rating, return, rotation, reason) %>%
  summary()
```

```
      race          rating             return            rotation
 Min.   :1.00   Length:100         Length:100         Length:100
 1st Qu.:1.75   Class :character   Class :character   Class :character
 Median :3.00   Mode  :character   Mode  :character   Mode  :character
 Mean   :2.57
 3rd Qu.:4.00
 Max.   :4.00
    reason
 Length:100
 Class :character
 Mode  :character
```

10.1 Working with race

race is coded as 1 = White, 2 = Black, 3 = Asian and 4 = Other, with no missing values...

```
demodata %>% tabyl(race)
```

```
 race  n percent
    1 25    0.25
    2 21    0.21
    3 26    0.26
    4 28    0.28
```

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

```
demodata <- demodata %>%
  mutate(race_f = factor(race),
         race_f =
           fct_recode(factor(race),
                      "White" = "1",
                      "Black" = "2",
                      "Asian" = "3",
                      "Other" = "4"))
```

Here's the sanity check...

```
demodata %>% tabyl(race, race_f)
```

```
 race 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 might 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.

```
demodata <- demodata %>%
  mutate(race_W = as.numeric(race_f == "White"),
         race_B = as.numeric(race_f == "Black"),
         race_A = as.numeric(race_f == "Asian"),
         race_O = as.numeric(race_f == "Other"))

demodata %>%
  count(race, race_f, race_W, race_B, race_A, race_O)
```

```
# A tibble: 4 x 7
   race race_f race_W race_B race_A race_O     n
  <int> <fct>   <dbl>  <dbl>  <dbl>  <dbl> <int>
1     1 White       1      0      0      0    25
```

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

10.2 Working with rating

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

```
demodata %>% count(rating)
```

```
# A tibble: 6 x 2
  rating     n
  <chr>  <int>
1 Exc        7
2 Fair       9
3 Good      54
4 Poor       5
5 V Good    21
6 <NA>       4
```

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

```
demodata <- demodata %>%
  mutate(rating_f = rating,
         rating_f =
           fct_relevel(rating_f,
                       "Exc", "V Good", "Good", "Fair", "Poor"))

demodata %>% tabyl(rating_f)
```

```
 rating_f  n percent valid_percent
      Exc  7    0.07    0.07291667
   V Good 21    0.21    0.21875000
     Good 54    0.54    0.56250000
     Fair  9    0.09    0.09375000
     Poor  5    0.05    0.05208333
     <NA>  4    0.04            NA
```

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(500006)
demodata_imp06 <- demodata %>%
  mutate(rating_imp = rating_f) %>%
  data.frame() %>%
```

```
    impute_rhd(., rating_imp ~ 1, pool="univariate") %>%
  tbl_df()

demodata_imp06 %>% count(rating_f, rating_imp)
```

```
# A tibble: 7 x 3
  rating_f rating_imp     n
  <fct>    <fct>      <int>
1 Exc      Exc            7
2 V Good   V Good        21
3 Good     Good          54
4 Fair     Fair           9
5 Poor     Poor           5
6 <NA>     Good           3
7 <NA>     Poor           1
```

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_imp06 <- demodata_imp06 %>%
  mutate(rating_10 = fct_collapse(rating_imp,
                                  High = c("Exc", "V Good", "Good"),
                                  Low = c("Fair", "Poor")))

demodata_imp06 %>% count(rating_10, rating_imp)
```

```
# A tibble: 5 x 3
  rating_10 rating_imp     n
  <fct>     <fct>      <int>
1 High      Exc            7
2 High      V Good        21
3 High      Good          57
4 Low       Fair           9
5 Low       Poor           6
```

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, which R recognizes as missing.

```
demodata %>% tabyl(return)
```

```
 return  n percent valid_percent
```

```
      A 14    0.14     0.1891892
      B 13    0.13     0.1756757
      C 30    0.30     0.4054054
      D 17    0.17     0.2297297
   <NA> 26    0.26            NA
```

Imputing here can work as we've done in the past.

```
set.seed(500007)

demodata_imp07 <- demodata %>%
  mutate(return_imp = return) %>%
  data.frame() %>%
    impute_rhd(., return_imp ~ 1, pool="univariate") %>%
  tbl_df()

demodata_imp07 %>% count(return, return_imp)
```

```
# A tibble: 8 x 3
  return return_imp     n
  <chr>  <chr>      <int>
1 A      A             14
2 B      B             13
3 C      C             30
4 D      D             17
5 <NA>   A              6
6 <NA>   B              6
7 <NA>   C             11
8 <NA>   D              3
```

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"`.

```
demodata %>% tabyl(rotation)
```

```
 rotation  n percent
  Unknown  4    0.04
        X 23    0.23
        Y 47    0.47
        Z 26    0.26
```

First, we convert those to NAs, creating `rotation_fix` and then we impute into `rotation_imp`.

```
demodata_imp08 <- demodata %>%
    mutate(rotation_orig = rotation,
           rotation_fix = na_if(rotation_orig, "Unknown"),
           rotation_imp = rotation_fix) %>%
  data.frame() %>%
    impute_rhd(., rotation_imp ~ 1, pool="univariate") %>%
  tbl_df()
```

```
demodata_imp08 %>% count(rotation_orig, rotation_fix, rotation_imp)
```

```
# A tibble: 6 x 4
  rotation_orig rotation_fix rotation_imp     n
  <chr>         <chr>        <chr>        <int>
1 Unknown       <NA>         X                1
2 Unknown       <NA>         Y                1
3 Unknown       <NA>         Z                2
4 X             X            X               23
5 Y             Y            Y               47
6 Z             Z            Z               26
```

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.

```
demodata %>% tabyl(reason)
```

```
      reason  n percent
     anxiety  5    0.05
      costly  7    0.07
   expensive 22    0.22
        fear 15    0.15
 high priced  4    0.04
     no time 13    0.13
       panic  4    0.04
     swamped  6    0.06
     tied up  8    0.08
    too busy  7    0.07
      unease  4    0.04
       worry  5    0.05
```

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, again using `fct_collapse`...

```
demodata <- demodata %>%
  mutate(reason_3 =
           fct_collapse(reason,
                        fear = c("anxiety", "fear", "panic", "unease", "worry"),
                        cost = c("costly", "expensive", "high priced"),
                        time = c("no time", "swamped", "tied up", "too busy")))

demodata %>% count(reason_3, reason)
```

```
# A tibble: 12 x 3
   reason_3 reason          n
   <fct>    <chr>       <int>
 1 fear     anxiety         5
 2 fear     fear           15
 3 fear     panic           4
 4 fear     unease          4
 5 fear     worry           5
 6 cost     costly          7
 7 cost     expensive      22
 8 cost     high priced     4
 9 time     no time        13
10 time     swamped         6
11 time     tied up         8
12 time     too busy        7
```

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.

	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

Neither import well into R through `read_csv`.

- `date1` produces an unordered factor, and
- `date2` just produces a set of integers.

```
demodata %>% select(date1:date2) %>% str()
```

```
tibble [100 x 2] (S3: tbl_df/tbl/data.frame)
 $ date1: chr [1:100] "7/10/2011" "6/5/2013" "5/27/2013" "3/31/2012" ...
 $ date2: int [1:100] 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.         Max.
"2010-12-29" "2011-09-19" "2012-05-11" "2012-05-26" "2012-12-04" "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.         Max.
"2010-12-29" "2011-09-19" "2012-05-11" "2012-05-26" "2012-12-04" "2013-12-09"
```

12 On Using RStudio and R Markdown

12.1 Use R Studio Projects Whenever You Can

- As the documentation suggests, RStudio Projects make it straightforward to divide your work into multiple contexts, each with their own working directory, workspace, history, and source documents.
- Open R Studio, and either start a new Project (using the File . . . New Project menu) or open an existing Project by clicking on the Open Project button at the top right of R Studio.

12.2 What is R Markdown?

R Markdown is the most useful tool I have. It is. . .

> an authoring format that enables easy creation of dynamic documents, presentations, and reports from R. It combines the core syntax of markdown (an easy-to-write plain text format) with embedded R code chunks that are run so their output can be included in the final document. R Markdown documents are fully reproducible (they can be automatically regenerated whenever underlying R code or data changes).

Moving from simply writing scripts in R to Markdown files is a short step, and well worth the effort. Within R Studio, you can write R Markdown syntax, run it to see your results,

and then produce a final document which looks great, and is completely reproducible. This is a tool you will get a lot of use from in this course, and, I expect, in your future work.

12.2.1 Learning More about Writing Markdown Files

1. I encourage you to visit the help page linked to directly within R Studio, by clicking on the question mark box in R Studio that appears when you open a Markdown file.
2. To get started, try simply writing your report in plain text. Markdown syntax is used to describe how to format text in the final report, and to embed R code.
3. To learn more about writing Markdown files, you can look at the examples I'll provide in class (all of my presentations are built using Markdown).
4. R Studio provides cheat sheets that others have found to be very helpful. They have stuff for R Markdown (including a detailed reference guide), R Studio, Data Wrangling with dplyr and tidyr, Data Visualization, etc.

12.2.2 Creating an HTML, Word or PDF file

1. An R Markdown file is essentially a text document, with interspersed R code that lets you produce reports that combine narration with results, and that can be easily exported as an HTML, PDF or Word file.
    - Open a new R Markdown file (File ... New File ... R Markdown), or an existing one (File ... Open File), and indicate the desired output type.

    - R Markdown files use the `.Rmd` extension.
    - Assuming you have Microsoft Office installed on your computer, you should be immediately able to write a Markdown file and render it in either HTML or as a Word document, by creating a Markdown document and then clicking on the Knit HTML or Knit Word button at the top of R Studio.
    - In order to get Markdown to generate a PDF file directly (rather than Word or HTML) you'll want to download an installation of the TeX (TeX is pronounced "tech") software, which builds those relatively pretty documents. I use MacTeX on my Mac, and MikTeX on my PC. This also lets me include LaTeX and TeX commands directly within a Markdown file, which is something you might eventually want to do, and that I did to build this document.

12.2.3 Some Specific Tips

1. When using R Markdown, you need to be sure that Markdown is looking for your files in the proper directory. The easiest way for me to do this is to build a separate directory for each new analysis, and open up a new Project in that directory before developing your Markdown code.
    - The directory being used is almost inevitably the directory in which the Markdown file is stored by RStudio.

    - The `here` package can be enormously helpful.
2. When writing code in Markdown, you need to name each chunk, and give each chunk a different name.
    - If you name something `chunkname`, then your next chunk of code needs to be named something different than `chunkname`, like `chunk2`, or whatever.
    - Good coding practice suggests the use of a name that describes what the chunk of code does. I encourage you to use the underscore to separate words, rather than spaces.
    - Putting a comma after a chunk name lets you, in addition to naming the chunk, specify commands (after the comma) to R Markdown about what you want it to do with the chunk. Here are a few useful commands, some of which also appear at http://rmarkdown.rstudio.com/authoring_rcodechunks.html:

- **{r chunk01, echo=FALSE}** tells Markdown to execute your code, but simply write the result, rather than the code, into the results file.
- **{r chunk01, eval=FALSE}** tells Markdown to write the code into the results, but not execute the code.
- **{r chunk01, message=FALSE}** tells Markdown to not print any of the messages your code generates.
- **{r chunk01, warning=FALSE}** tells Markdown to not print any of the warnings your code generates.
- **{r chunk01, fig.height=4, fig.width=6}** tells Markdown to keep any figures produced by this chunk to a maximum of 4 inches high, and 6 inches wide. The default values depend on the type of output you are generating. You can use fig.height or fig.width alone if you want to keep the other value at its default.
- **{r chunk01, fig.align=`center`}** tells Markdown to align your figure in the center (other options are left or right.)
- **{r chunk01, tidy=TRUE}** tells Markdown to tidy up your code for presentation, so it will still print, but it will (perhaps) be a bit more organized.

Here's the default header information I use when writing materials for a PDF document with a table of contents...

```
---
title: "Basic R Materials for 432 and 500"
author: "Thomas E. Love"
date: "2022-01-06"
output:
  pdf_document:
    number_sections: yes
    toc: yes
    toc_depth: 3
fontsize: 12pt
geometry: margin=1in
---
```

To set up R Markdown so it doesn't add any comment symbols before the results, I follow that with this code chunk:

```
knitr::opts_chunk$set(comment=NA)
```

Then I load packages, and data. I usually load `magrittr`, `here`, `janitor` and the `tidyverse` at a minimum.

12.3 Session Information

```
xfun::session_info()
```

```
R version 4.1.2 (2021-11-01)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 19043)

Locale:
  LC_COLLATE=English_United States.1252
  LC_CTYPE=English_United States.1252
  LC_MONETARY=English_United States.1252
  LC_NUMERIC=C
  LC_TIME=English_United States.1252

Package version:
  abind_1.4-5             askpass_1.1             assertthat_0.2.1
  backports_1.3.0         base64enc_0.1-3         bit_4.0.4
  bit64_4.0.5             blob_1.2.2              boot_1.3.28
  broom_0.7.10            callr_3.7.0             car_3.0-12
  carData_3.0-4           caret_6.0.90            cellranger_1.1.0
  checkmate_2.0.0         class_7.3.19            cli_3.1.0
  clipr_0.7.1             cluster_2.1.2           cmprsk_2.2-10
  codetools_0.2.18        colorspace_2.0-2        compiler_4.1.2
  conquer_1.2.1           cpp11_0.4.2             crayon_1.4.2
  crosstalk_1.2.0         curl_4.3.2              data.table_1.14.2
  DBI_1.1.1               dbplyr_2.1.1            digest_0.6.28
  dplyr_1.0.7             dtplyr_1.1.0            e1071_1.7.9
  ellipsis_0.3.2          Epi_2.44                etm_1.1.1
  evaluate_0.14           fansi_0.5.0             farver_2.1.0
  fastmap_1.1.0           forcats_0.5.1           foreach_1.5.1
  foreign_0.8-81          Formula_1.2-4           fs_1.5.0
  future_1.23.0           future.apply_1.8.1      gargle_1.2.0
  generics_0.1.1          ggdendro_0.1.22         ggforce_0.3.3
  ggformula_0.10.1        ggplot2_3.3.5           ggrepel_0.9.1
  ggridges_0.5.3          ggstance_0.3.5          globals_0.14.0
```

glue_1.4.2 googledrive_2.0.0 googlesheets4_1.0.0
gower_0.2.2 graphics_4.1.2 grDevices_4.1.2
grid_4.1.2 gridExtra_2.3 gtable_0.3.0
haven_2.4.3 here_1.0.1 highr_0.9
Hmisc_4.6-0 hms_1.1.1 htmlTable_2.3.0
htmltools_0.5.2 htmlwidgets_1.5.4 httr_1.4.2
ids_1.0.1 ipred_0.9.12 isoband_0.2.5
iterators_1.0.13 janitor_2.1.0 jpeg_0.1-9
jquerylib_0.1.4 jsonlite_1.7.2 KernSmooth_2.23.20
knitr_1.36 labeling_0.4.2 labelled_2.9.0
lattice_0.20-45 latticeExtra_0.6-29 lava_1.6.10
lazyeval_0.2.2 leaflet_2.0.4.1 leaflet.providers_1.9.0
lifecycle_1.0.1 listenv_0.8.0 lme4_1.1.27.1
lubridate_1.8.0 magrittr_2.0.1 maptools_1.1.2
markdown_1.1 MASS_7.3-54 Matrix_1.3-4
MatrixModels_0.5.0 matrixStats_0.61.0 methods_4.1.2
mgcv_1.8-38 mime_0.12 minqa_1.2.4
ModelMetrics_1.2.2.2 modelr_0.1.8 mosaic_1.8.3
mosaicCore_0.9.0 mosaicData_0.20.2 munsell_0.5.0
naniar_0.6.1 nlme_3.1-153 nloptr_1.2.2.2
nnet_7.3-16 norm_1.0.9.5 numDeriv_2016.8-1.1
openssl_1.4.5 parallel_4.1.2 parallelly_1.29.0
pbkrtest_0.5.1 pillar_1.6.4 pkgconfig_2.0.3
plyr_1.8.6 png_0.1-7 polyclip_1.10-0
prettyunits_1.1.1 pROC_1.18.0 processx_3.5.2
prodlim_2019.11.13 progress_1.2.2 progressr_0.9.0
proxy_0.4.26 ps_1.6.0 purrr_0.3.4
quantreg_5.86 R6_2.5.1 rappdirs_0.3.3
raster_3.5.2 RColorBrewer_1.1-2 Rcpp_1.0.7
RcppArmadillo_0.10.7.3.0 RcppEigen_0.3.3.9.1 readr_2.1.0
readxl_1.3.1 recipes_0.1.17 rematch_1.0.1
rematch2_2.1.2 reprex_2.0.1 reshape2_1.4.4
rlang_0.4.12 rmarkdown_2.11 rpart_4.1-15
rprojroot_2.0.2 rstudioapi_0.13 rvest_1.0.2
scales_1.1.1 selectr_0.4.2 simputation_0.2.7
snakecase_0.11.0 sp_1.4.6 SparseM_1.81
splines_4.1.2 SQUAREM_2021.1 stats_4.1.2
stats4_4.1.2 stringi_1.7.5 stringr_1.4.0
survival_3.2-13 sys_3.4 terra_1.4.20
tibble_3.1.5 tidyr_1.1.4 tidyselect_1.1.1
tidyverse_1.3.1 timeDate_3043.102 tinytex_0.35
tools_4.1.2 tweenr_1.0.2 tzdb_0.2.0
UpSetR_1.4.0 utf8_1.2.2 utils_4.1.2
uuid_1.0.3 vctrs_0.3.8 viridis_0.6.2
viridisLite_0.4.0 visdat_0.5.3 vroom_1.5.6

withr_2.4.3 xfun_0.27 xml2_1.3.2
yaml_2.2.1 zoo_1.8-9