Data Science for Biological, Medical and Health Research: Notes for 432

Thomas E. Love, Ph.D.

Built 2018-02-04 16:36:01

Contents

Introduction

These Notes provide a series of examples using R to work through issues that are likely to come up in PQHS/CRSP/MPHP 432.

While these Notes share some of the features of a textbook, they are neither comprehensive nor completely original. The main purpose is to give students in 432 a set of common materials on which to draw during the course. In class, we will sometimes:

- reiterate points made in this document,
- amplify what is here,
- simplify the presentation of things done here,
- use new examples to show some of the same techniques,
- refer to issues not mentioned in this document,

but what we don't (always) do is follow these notes very precisely. We assume instead that you will read the materials and try to learn from them, just as you will attend classes and try to learn from them. We welcome feedback of all kinds on this document or anything else. Just email us at `431-help at case dot edu`, or submit a pull request. Note that we still use `431-help` even though we're now in 432.

What you will mostly find are brief explanations of a key idea or summary, accompanied (most of the time) by R code and a demonstration of the results of applying that code.

Everything you see here is available to you as HTML or PDF. You will also have access to the R Markdown files, which contain the code which generates everything in the document, including all of the R results. We will demonstrate the use of R Markdown (this document is generated with the additional help of an R package called bookdown) and R Studio (the "program" which we use to interface with the R language) in class.

To download the data and R code related to these notes, visit the Data and Code section of the 432 course website.

R Packages used in these notes

Here, we'll load in the packages used in these notes.

```
library(tableone)
library(skimr)
library(ggridges)
library(magrittr)
library(arm)
library(rms)
library(leaps)
library(lars)
library(Epi)
library(pROC)
library(ROCR)
library(simputation)
library(modelr)
library(broom)
library(tidyverse)
```

Data used in these notes

Here, we'll load in the data sets used in these notes.

```
fakestroke <- read.csv("data/fakestroke.csv") %>% tbl_df
bloodbrain <- read.csv("data/bloodbrain.csv") %>% tbl_df
smartcle1 <- read.csv("data/smartcle1.csv") %>% tbl_df
bonding <- read.csv("data/bonding.csv") %>% tbl_df
cortisol <- read.csv("data/cortisol.csv") %>% tbl_df
emphysema <- read.csv("data/emphysema.csv") %>% tbl_df
prost <- read.csv("data/prost.csv") %>% tbl_df
pollution <- read.csv("data/pollution.csv") %>% tbl_df
resect <- read.csv("data/resect.csv") %>% tbl_df
```

Special Functions used in these notes

```
specify_decimal <- function(x, k) format(round(x, k), nsmall=k)
skim_with(numeric = list(hist = NULL),
          integer = list(hist = NULL),
          ts = list(line_graph = NULL))
```

Chapter 1

Building Table 1

Many scientific articles involve direct comparison of results from various exposures, perhaps treatments. In 431, we studied numerous methods, including various sorts of hypothesis tests, confidence intervals, and descriptive summaries, which can help us to understand and compare outcomes in such a setting. One common approach is to present what's often called Table 1. Table 1 provides a summary of the characteristics of a sample, or of groups of samples, which is most commonly used to help understand the nature of the data being compared.

1.1 Two examples from the *New England Journal of Medicine*

1.1.1 A simple Table 1

Table 1 is especially common in the context of clinical research. Consider the excerpt below, from a January 2015 article in the *New England Journal of Medicine* (Tolaney et al., 2015).

Table 1. Baseline Characteristics of the Patients.*

Characteristic	Patients (N=406)
	no. (%)
Age group	
<50 yr	132 (32.5)
50–59 yr	137 (33.7)
60–69 yr	96 (23.6)
≥70 yr	41 (10.1)
Sex	
Female	405 (99.8)
Male	1 (0.2)
Race†	
White	351 (86.5)
Black	28 (6.9)
Asian	11 (2.7)
Other	16 (3.9)

This (partial) table reports baseline characteristics on age group, sex and race, describing 406 patients with

HER2-positive[1] invasive breast cancer that began the protocol therapy. Age, sex and race (along with severity of illness) are the most commonly identified characteristics in a Table 1.

In addition to the measures shown in this excerpt, the full Table also includes detailed information on the primary tumor for each patient, including its size, nodal status and histologic grade. Footnotes tell us that the percentages shown are subject to rounding, and may not total 100, and that the race information was self-reported.

1.1.2 A group comparison

A more typical Table 1 involves a group comparison, for example in this excerpt from Roy et al. (2008). This Table 1 describes a multi-center randomized clinical trial comparing two different approaches to caring for patients with heart failure and atrial fibrillation[2].

Table 1. Baseline Characteristics of the Patients.*

Variable	Rhythm-Control Group (N=682)	Rate-Control Group (N=694)
Male sex (%)	78	85
Age (yr)	66±11	67±11
Body-mass index†	27.8±5.4	28.0±5.1
Nonwhite race (%)‡	16	13
NYHA class III or IV (%)		
At baseline	32	31
During previous 6 mo	76	76
Predominant cardiac diagnosis (%)§		
Coronary artery disease	48	48
Valvular heart disease	5	5
Nonischemic cardiomyopathy	36	39
Congenital heart disease	1	1
Hypertensive heart disease	10	7

The article provides percentages, means and standard deviations across groups, but note that it does not provide p values for the comparison of baseline characteristics. This is a common feature of NEJM reports on randomized clinical trials, where we anticipate that the two groups will be well matched at baseline. Note that the patients in this study were *randomly* assigned to either the rhythm-control group or to the rate-control group, using blocked randomizations stratified by study center.

1.2 The MR CLEAN trial

Berkhemer et al. (2015) reported on the MR CLEAN trial, involving 500 patients with acute ischemic stroke caused by a proximal intracranial arterial occlusion. The trial was conducted at 16 medical centers in the Netherlands, where 233 were randomly assigned to the intervention (intraarterial treatment plus usual care) and 267 to control (usual care alone.) The primary outcome was the modified Rankin scale score at 90 days; this categorical scale measures functional outcome, with scores ranging from 0 (no symptoms) to 6 (death). The fundamental conclusion of Berkhemer et al. (2015) was that in patients with acute ischemic stroke

[1] HER2 = human epidermal growth factor receptor type 2. Over-expression of this occurs in 15-20% of invasive breast cancers, and has been associated with poor outcomes.

[2] The complete Table 1 appears on pages 2668-2669 of Roy et al. (2008), but I have only reproduced the first page and the footnote in this excerpt.

caused by a proximal intracranial occlusion of the anterior circulation, intraarterial treatment administered within 6 hours after stroke onset was effective and safe.

Here's the Table 1 from Berkhemer et al. (2015).

Table 1. Baseline Characteristics of the 500 Patients.*

Characteristic	Intervention (N = 233)	Control (N = 267)
Age — yr		
Median	65.8	65.7
Interquartile range	54.5–76.0	55.5–76.4
Male sex — no. (%)	135 (57.9)	157 (58.8)
NIHSS score†		
Median (interquartile range)	17 (14–21)	18 (14–22)
Range	3–30	4–38
Location of stroke in left hemisphere — no. (%)	116 (49.8)	153 (57.3)
History of ischemic stroke — no. (%)	29 (12.4)	25 (9.4)
Atrial fibrillation — no. (%)	66 (28.3)	69 (25.8)
Diabetes mellitus — no. (%)	34 (14.6)	34 (12.7)
Prestroke modified Rankin scale score — no. (%)‡		
0	190 (81.5)	214 (80.1)
1	21 (9.0)	29 (10.9)
2	12 (5.2)	13 (4.9)
>2	10 (4.3)	11 (4.1)
Systolic blood pressure — mm Hg§	146±26.0	145±24.4
Treatment with IV alteplase — no. (%)	203 (87.1)	242 (90.6)
Time from stroke onset to start of IV alteplase — min		
Median	85	87
Interquartile range	67–110	65–116
ASPECTS — median (interquartile range)¶	9 (7–10)	9 (8–10)
Intracranial arterial occlusion — no./total no. (%)‖		
Intracranial ICA	1/233 (0.4)	3/266 (1.1)
ICA with involvement of the M1 middle cerebral artery segment	59/233 (25.3)	75/266 (28.2)
M1 middle cerebral artery segment	154/233 (66.1)	165/266 (62.0)
M2 middle cerebral artery segment	18/233 (7.7)	21/266 (7.9)
A1 or A2 anterior cerebral artery segment	1/233 (0.4)	2/266 (0.8)
Extracranial ICA occlusion — no./total no. (%)‖**	75/233 (32.2)	70/266 (26.3)
Time from stroke onset to randomization — min††		
Median	204	196
Interquartile range	152–251	149–266
Time from stroke onset to groin puncture — min		
Median	260	NA
Interquartile range	210–313	

The Table was accompanied by the following notes.

* The intervention group was assigned to intraarterial treatment plus usual care, and the control group was assigned to usual care alone. Plus–minus values are means ±SD. ICA denotes internal carotid artery, IV intravenous, and NA not applicable.
† Scores on the National Institutes of Health Stroke Scale (NIHSS) range from 0 to 42, with higher scores indicating more severe neurologic deficits. The NIHSS is a 15-item scale, and values for 30 of the 7500 items were missing (0.4%). The highest number of missing items for a single patient was 6.
‡ Scores on the modified Rankin scale of functional disability range from 0 (no symptoms) to 6 (death). A score of 2 or less indicates functional independence.
§ Data on systolic blood pressure at baseline were missing for one patient assigned to the control group.
¶ The Alberta Stroke Program Early Computed Tomography Score (ASPECTS) is a measure of the extent of stroke. Scores ranges from 0 to 10, with higher scores indicating fewer early ischemic changes. Scores were not available for four patients assigned to the control group: noncontrast computed tomography was not performed in one patient, and three patients had strokes in the territory of the anterior cerebral artery.
‖ Vessel imaging was not performed in one patient in the control group, so the level of occlusion was not known.
** Extracranial ICA occlusions were reported by local investigators.
†† Data were missing for two patients in the intervention group.

1.3 Simulated `fakestroke` data

Consider the simulated data, available on the Data and Code page of our course website in the `fakestroke.csv` file, which I built to let us mirror the Table 1 for MR CLEAN (Berkhemer et al., 2015). The `fakestroke.csv` file contains the following 18 variables for 500 patients.

Variable	Description
`studyid`	Study ID # (z001 through z500)
`trt`	Treatment group (Intervention or Control)
`age`	Age in years
`sex`	Male or Female
`nihss`	NIH Stroke Scale Score (can range from 0-42; higher scores indicate more severe neurological deficits)
`location`	Stroke Location - Left or Right Hemisphere
`hx.isch`	History of Ischemic Stroke (Yes/No)
`afib`	Atrial Fibrillation (1 = Yes, 0 = No)
`dm`	Diabetes Mellitus (1 = Yes, 0 = No)
`mrankin`	Pre-stroke modified Rankin scale score (0, 1, 2 or > 2) indicating functional disability - complete range is 0 (no symptoms) to 6 (death)
`sbp`	Systolic blood pressure, in mm Hg
`iv.altep`	Treatment with IV alteplase (Yes/No)
`time.iv`	Time from stroke onset to start of IV alteplase (minutes) if iv.altep=Yes
`aspects`	Alberta Stroke Program Early Computed Tomography score, which measures extent of stroke from 0 - 10; higher scores indicate fewer early ischemic changes
`ia.occlus`	Intracranial arterial occlusion, based on vessel imaging - five categories[3]
`extra.ica`	Extracranial ICA occlusion (1 = Yes, 0 = No)
`time.rand`	Time from stroke onset to study randomization, in minutes
`time.punc`	Time from stroke onset to groin puncture, in minutes (only if Intervention)

Here's a quick look at the simulated data in `fakestroke`.

[3]The five categories are Intracranial ICA, ICA with involvement of the M1 middle cerebral artery segment, M1 middle cerebral artery segment, M2 middle cerebral artery segment, A1 or A2 anterior cerebral artery segment

```
fakestroke
```

```
# A tibble: 500 x 18
   studyid trt        age sex   nihss location hx.isch  afib    dm mrankin
   <fct>   <fct>    <dbl> <fct> <int> <fct>    <fct>   <int> <int> <fct>
 1 z001    Control   53.0 Male     21 Right    No          0     0 2
 2 z002    Interve~  51.0 Male     23 Left     No          1     0 0
 3 z003    Control   68.0 Fema~    11 Right    No          0     0 0
 4 z004    Control   28.0 Male     22 Left     No          0     0 0
 5 z005    Control   91.0 Male     24 Right    No          0     0 0
 6 z006    Control   34.0 Fema~    18 Left     No          0     0 2
 7 z007    Interve~  75.0 Male     25 Right    No          0     0 0
 8 z008    Control   89.0 Fema~    18 Right    No          0     0 0
 9 z009    Control   75.0 Male     25 Left     No          1     0 2
10 z010    Interve~  26.0 Fema~    27 Right    No          0     0 0
# ... with 490 more rows, and 8 more variables: sbp <int>, iv.altep <fct>,
#   time.iv <int>, aspects <int>, ia.occlus <fct>, extra.ica <int>,
#   time.rand <int>, time.punc <int>
```

1.4 Building Table 1 for fakestroke: Attempt 1

Our goal, then, is to take the data in `fakestroke.csv` and use it to generate a Table 1 for the study that compares the 233 patients in the Intervention group to the 267 patients in the Control group, on all of the other variables (except study ID #) available. I'll use the `tableone` package of functions available in R to help me complete this task. We'll make a first attempt, using the `CreateTableOne` function in the `tableone` package. To use the function, we'll need to specify:

- the `vars` or variables we want to place in the rows of our Table 1 (which will include just about everything in the `fakestroke` data except the `studyid` code and the `trt` variable for which we have other plans, and the `time.punc` which applies only to subjects in the Intervention group.)
  - A useful trick here is to use the `dput` function, specifically something like `dput(names(fakestroke))` can be used to generate a list of all of the variables included in the `fakestroke` tibble, and then this can be copied and pasted into the `vars` specification, saving some typing.
- the `strata` which indicates the levels want to use in the columns of our Table 1 (for us, that's `trt`)

```
fs.vars <- c("age", "sex", "nihss", "location",
          "hx.isch", "afib", "dm", "mrankin", "sbp",
          "iv.altep", "time.iv", "aspects",
          "ia.occlus", "extra.ica", "time.rand")

fs.trt <- c("trt")

att1 <- CreateTableOne(data = fakestroke,
                       vars = fs.vars,
                       strata = fs.trt)
print(att1)
```

```
                       Stratified by trt
                        Control        Intervention   p      test
  n                        267            233
  age (mean (sd))        65.38 (16.10)  63.93 (18.09)  0.343
  sex = Male (%)           157 (58.8)     135 (57.9)   0.917
  nihss (mean (sd))      18.08 (4.32)   17.97 (5.04)   0.787
  location = Right (%)     114 (42.7)     117 (50.2)   0.111
```

```
hx.isch = Yes (%)          25 ( 9.4)      29 (12.4)   0.335
afib (mean (sd))         0.26 (0.44)    0.28 (0.45)   0.534
dm (mean (sd))           0.13 (0.33)    0.12 (0.33)   0.923
mrankin (%)                                           0.922
   > 2                     11 ( 4.1)      10 ( 4.3)
   0                      214 (80.1)     190 (81.5)
   1                       29 (10.9)      21 ( 9.0)
   2                       13 ( 4.9)      12 ( 5.2)
sbp (mean (sd))        145.00 (24.40) 146.03 (26.00)  0.647
iv.altep = Yes (%)        242 (90.6)     203 (87.1)   0.267
time.iv (mean (sd))     87.96 (26.01)  98.22 (45.48)  0.003
aspects (mean (sd))      8.65 (1.47)    8.35 (1.64)   0.033
ia.occlus (%)                                         0.795
   A1 or A2                 2 ( 0.8)       1 ( 0.4)
   ICA with M1             75 (28.2)      59 (25.3)
   Intracranial ICA         3 ( 1.1)       1 ( 0.4)
   M1                     165 (62.0)     154 (66.1)
   M2                      21 ( 7.9)      18 ( 7.7)
extra.ica (mean (sd))    0.26 (0.44)    0.32 (0.47)   0.150
time.rand (mean (sd)) 213.88 (70.29) 202.51 (57.33)   0.051
```

1.4.1 Some of this is very useful, and other parts need to be fixed.

1. The 1/0 variables (`afib`, `dm`, `extra.ica`) might be better if they were treated as the factors they are, and reported as the Yes/No variables are reported, with counts and percentages rather than with means and standard deviations.
2. In some cases, we may prefer to re-order the levels of the categorical (factor) variables, particularly the `mrankin` variable, but also the `ia.occlus` variable. It would also be more typical to put the Intervention group to the left and the Control group to the right, so we may need to adjust our `trt` variable's levels accordingly.
3. For each of the quantitative variables (`age`, `nihss`, `sbp`, `time.iv`, `aspects`, `extra.ica`, `time.rand` and `time.punc`) we should make a decision whether a summary with mean and standard deviation is appropriate, or whether we should instead summarize with, say, the median and quartiles. A mean and standard deviation really only yields an appropriate summary when the data are least approximately Normally distributed. This will make the *p* values a bit more reasonable, too. The `test` column in the first attempt will soon have something useful to tell us.
4. If we'd left in the `time.punc` variable, we'd get some warnings, having to do with the fact that `time.punc` is only relevant to patients in the Intervention group.

1.4.2 `fakestroke` Cleaning Up Categorical Variables

Let's specify each of the categorical variables as categorical explicitly. This helps the `CreateTableOne` function treat them appropriately, and display them with counts and percentages. This includes all of the 1/0, Yes/No and multi-categorical variables.

```
fs.factorvars <- c("sex", "location", "hx.isch", "afib", "dm",
                   "mrankin", "iv.altep", "ia.occlus", "extra.ica")
```

Then we simply add a `factorVars = fs.factorvars` call to the `CreateTableOne` function.

We also want to re-order some of those categorical variables, so that the levels are more useful to us. Specifically, we want to:

- place Intervention before Control in the `trt` variable,
- reorder the `mrankin` scale as 0, 1, 2, > 2, and

- rearrange the `ia.occlus` variable to the order[4] presented in Berkhemer et al. (2015).

To accomplish this, we'll use the `fct_relevel` function from the `forcats` package (loaded with the rest of the core `tidyverse` packages) to reorder our levels manually.

```
fakestroke <- fakestroke %>%
    mutate(trt = fct_relevel(trt, "Intervention", "Control"),
           mrankin = fct_relevel(mrankin, "0", "1", "2", "> 2"),
           ia.occlus = fct_relevel(ia.occlus, "Intracranial ICA",
                                   "ICA with M1", "M1", "M2",
                                   "A1 or A2")
           )
```

1.5 `fakestroke` Table 1: Attempt 2

```
att2 <- CreateTableOne(data = fakestroke,
                       vars = fs.vars,
                       factorVars = fs.factorvars,
                       strata = fs.trt)
print(att2)
```

```
                       Stratified by trt
                        Intervention   Control        p      test
  n                        233            267
  age (mean (sd))         63.93 (18.09)  65.38 (16.10)  0.343
  sex = Male (%)           135 (57.9)     157 (58.8)    0.917
  nihss (mean (sd))       17.97 (5.04)   18.08 (4.32)   0.787
  location = Right (%)     117 (50.2)     114 (42.7)    0.111
  hx.isch = Yes (%)         29 (12.4)      25 ( 9.4)    0.335
  afib = 1 (%)              66 (28.3)      69 (25.8)    0.601
  dm = 1 (%)                29 (12.4)      34 (12.7)    1.000
  mrankin (%)                                           0.922
     0                     190 (81.5)     214 (80.1)
     1                      21 ( 9.0)      29 (10.9)
     2                      12 ( 5.2)      13 ( 4.9)
     > 2                    10 ( 4.3)      11 ( 4.1)
  sbp (mean (sd))        146.03 (26.00) 145.00 (24.40)  0.647
  iv.altep = Yes (%)       203 (87.1)     242 (90.6)    0.267
  time.iv (mean (sd))     98.22 (45.48)  87.96 (26.01)  0.003
  aspects (mean (sd))      8.35 (1.64)    8.65 (1.47)   0.033
  ia.occlus (%)                                         0.795
     Intracranial ICA        1 ( 0.4)       3 ( 1.1)
     ICA with M1            59 (25.3)      75 (28.2)
     M1                     154 (66.1)     165 (62.0)
     M2                      18 ( 7.7)      21 ( 7.9)
     A1 or A2                1 ( 0.4)       2 ( 0.8)
  extra.ica = 1 (%)         75 (32.2)      70 (26.3)    0.179
  time.rand (mean (sd)) 202.51 (57.33) 213.88 (70.29)  0.051
```

The categorical data presentation looks much improved.

[4] We might also have considered reordering the `ia.occlus` factor by its frequency, using the `fct_infreq` function

1.5.1 What summaries should we show?

Now, we'll move on to the issue of making a decision about what type of summary to show for the quantitative variables. Since the `fakestroke` data are just simulated and only match the summary statistics of the original results, not the details, we'll adopt the decisions made by Berkhemer et al. (2015), which were to use medians and interquartile ranges to summarize the distributions of all of the continuous variables **except** systolic blood pressure.

- Specifying certain quantitative variables as *non-normal* causes R to show them with medians and the 25th and 75th percentiles, rather than means and standard deviations, and also causes those variables to be tested using non-parametric tests, like the Wilcoxon signed rank test, rather than the t test. The `test` column indicates this with the word `nonnorm`.
  - In real data situations, what should we do? The answer is to look at the data. I would not make the decision as to which approach to take without first plotting (perhaps in a histogram or a Normal Q-Q plot) the observed distributions in each of the two samples, so that I could make a sound decision about whether Normality was a reasonable assumption. If the means and medians are meaningfully different from each other, this is especially important.
  - To be honest, though, if the variable in question is a relatively unimportant covariate and the p values for the two approaches are nearly the same, I'd say that further investigation is rarely important,
- Specifying *exact* tests for certain categorical variables (we'll try this for the `location` and `mrankin` variables) can be done, and these changes will be noted in the `test` column, as well.
  - In real data situations, I would rarely be concerned about this issue, and often choose Pearson (approximate) options across the board. This is reasonable so long as the number of subjects falling in each category is reasonably large, say above 10. If not, then an exact test may be a tiny improvement.
  - Paraphrasing Rosenbaum (2017), having an exact rather than an approximate test result is about as valuable as having a nice crease in your trousers.

To finish our Table 1, then, we need to specify which variables should be treated as non-Normal in the `print` statement - notice that we don't need to redo the `CreateTableOne` for this change.

```
print(att2,
      nonnormal = c("age", "nihss", "time.iv", "aspects", "time.rand"),
      exact = c("location", "mrankin"))
```

```
                         Stratified by trt
                          Intervention            Control
  n                          233                     267
  age (median [IQR])        65.80 [54.50, 76.00]    65.70 [55.75, 76.20]
  sex = Male (%)             135 (57.9)              157 (58.8)
  nihss (median [IQR])      17.00 [14.00, 21.00]    18.00 [14.00, 22.00]
  location = Right (%)       117 (50.2)              114 (42.7)
  hx.isch = Yes (%)           29 (12.4)               25 ( 9.4)
  afib = 1 (%)                66 (28.3)               69 (25.8)
  dm = 1 (%)                  29 (12.4)               34 (12.7)
  mrankin (%)
     0                       190 (81.5)              214 (80.1)
     1                        21 ( 9.0)               29 (10.9)
     2                        12 ( 5.2)               13 ( 4.9)
     > 2                      10 ( 4.3)               11 ( 4.1)
  sbp (mean (sd))           146.03 (26.00)          145.00 (24.40)
  iv.altep = Yes (%)         203 (87.1)              242 (90.6)
  time.iv (median [IQR])     85.00 [67.00, 110.00]   87.00 [65.00, 116.00]
  aspects (median [IQR])      9.00 [7.00, 10.00]      9.00 [8.00, 10.00]
  ia.occlus (%)
```

```
   Intracranial ICA           1 ( 0.4)               3 ( 1.1)
   ICA with M1               59 (25.3)              75 (28.2)
   M1                       154 (66.1)             165 (62.0)
   M2                        18 ( 7.7)              21 ( 7.9)
   A1 or A2                   1 ( 0.4)               2 ( 0.8)
extra.ica = 1 (%)            75 (32.2)              70 (26.3)
time.rand (median [IQR]) 204.00 [152.00, 249.50] 196.00 [149.00, 266.00]
                        Stratified by trt
                         p      test
n
age (median [IQR])        0.579 nonnorm
sex = Male (%)            0.917
nihss (median [IQR])      0.453 nonnorm
location = Right (%)      0.106 exact
hx.isch = Yes (%)         0.335
afib = 1 (%)              0.601
dm = 1 (%)                1.000
mrankin (%)               0.917 exact
   0
   1
   2
   > 2
sbp (mean (sd))           0.647
iv.altep = Yes (%)        0.267
time.iv (median [IQR])    0.596 nonnorm
aspects (median [IQR])    0.075 nonnorm
ia.occlus (%)             0.795
   Intracranial ICA
   ICA with M1
   M1
   M2
   A1 or A2
extra.ica = 1 (%)         0.179
time.rand (median [IQR])  0.251 nonnorm
```

1.6 Obtaining a more detailed Summary

If this was a real data set, we'd want to get a more detailed description of the data to make decisions about things like potentially collapsing categories of a variable, or whether or not a normal distribution was useful for a particular continuous variable, etc. You can do this with the `summary` command applied to a created Table 1, which shows, among other things, the effect of changing from normal to non-normal p values for continuous variables, and from approximate to "exact" p values for categorical factors.

Again, as noted above, in a real data situation, we'd want to plot the quantitative variables (within each group) to make a smart decision about whether a t test or Wilcoxon approach is more appropriate.

Note in the summary below that we have some missing values here. Often, we'll present this information within the Table 1, as well.

```
summary(att2)
```

```
### Summary of continuous variables ###
```

```
trt: Intervention
            n miss p.miss mean sd median p25 p75 min max  skew  kurt
age       233    0    0.0   64 18     66  54  76  23  96 -0.34 -0.52
nihss     233    0    0.0   18  5     17  14  21  10  28  0.48 -0.74
sbp       233    0    0.0  146 26    146 129 164  78 214 -0.07 -0.22
time.iv   233   30   12.9   98 45     85  67 110  42 218  1.03  0.08
aspects   233    0    0.0    8  2      9   7  10   5  10 -0.56 -0.98
time.rand 233    2    0.9  203 57    204 152 250 100 300  0.01 -1.16
------------------------------------------------------------
trt: Control
            n miss p.miss mean sd median p25 p75 min max   skew  kurt
age       267    0    0.0   65 16     66  56  76  24  94 -0.296 -0.28
nihss     267    0    0.0   18  4     18  14  22  11  25  0.017 -1.24
sbp       267    1    0.4  145 24    145 128 161  82 231  0.156  0.08
time.iv   267   25    9.4   88 26     87  65 116  44 130  0.001 -1.32
aspects   267    4    1.5    9  1      9   8  10   5  10 -1.071  0.36
time.rand 267    0    0.0  214 70    196 149 266 120 360  0.508 -0.93

p-values
              pNormal pNonNormal
age       0.342813660 0.57856976
nihss     0.787487252 0.45311695
sbp       0.647157646 0.51346132
time.iv   0.003073372 0.59641104
aspects   0.032662901 0.07464683
time.rand 0.050803672 0.25134327

Standardize mean differences
              1 vs 2
age       0.08478764
nihss     0.02405390
sbp       0.04100833
time.iv   0.27691223
aspects   0.19210662
time.rand 0.17720957

================================================================================

     ### Summary of categorical variables ###

trt: Intervention
       var   n miss p.miss           level freq percent cum.percent
       sex 233    0    0.0          Female   98    42.1        42.1
                                     Male  135    57.9       100.0

  location 233    0    0.0            Left  116    49.8        49.8
                                    Right  117    50.2       100.0

   hx.isch 233    0    0.0              No  204    87.6        87.6
                                       Yes   29    12.4       100.0

      afib 233    0    0.0               0  167    71.7        71.7
                                         1   66    28.3       100.0
```

```
       dm 233    0    0.0               0  204    87.6        87.6
                                        1   29    12.4       100.0

   mrankin 233    0    0.0               0  190    81.5        81.5
                                        1   21     9.0        90.6
                                        2   12     5.2        95.7
                                      > 2   10     4.3       100.0

  iv.altep 233    0    0.0             No   30    12.9        12.9
                                      Yes  203    87.1       100.0

 ia.occlus 233    0    0.0 Intracranial ICA    1     0.4         0.4
                           ICA with M1   59    25.3        25.8
                                   M1  154    66.1        91.8
                                   M2   18     7.7        99.6
                              A1 or A2    1     0.4       100.0

 extra.ica 233    0    0.0               0  158    67.8        67.8
                                        1   75    32.2       100.0

------------------------------------------------------------
trt: Control
       var   n miss p.miss           level freq percent cum.percent
       sex 267    0    0.0          Female  110    41.2        41.2
                                     Male  157    58.8       100.0

  location 267    0    0.0            Left  153    57.3        57.3
                                      Right  114    42.7       100.0

   hx.isch 267    0    0.0              No  242    90.6        90.6
                                       Yes   25     9.4       100.0

      afib 267    0    0.0               0  198    74.2        74.2
                                        1   69    25.8       100.0

       dm 267    0    0.0               0  233    87.3        87.3
                                        1   34    12.7       100.0

   mrankin 267    0    0.0               0  214    80.1        80.1
                                        1   29    10.9        91.0
                                        2   13     4.9        95.9
                                      > 2   11     4.1       100.0

  iv.altep 267    0    0.0              No   25     9.4         9.4
                                       Yes  242    90.6       100.0

 ia.occlus 267    1    0.4 Intracranial ICA    3     1.1         1.1
                           ICA with M1   75    28.2        29.3
                                   M1  165    62.0        91.4
                                   M2   21     7.9        99.2
                              A1 or A2    2     0.8       100.0

 extra.ica 267    1    0.4               0  196    73.7        73.7
                                        1   70    26.3       100.0
```

```
p-values
            pApprox    pExact
sex       0.9171387 0.8561188
location  0.1113553 0.1056020
hx.isch   0.3352617 0.3124683
afib      0.6009691 0.5460206
dm        1.0000000 1.0000000
mrankin   0.9224798 0.9173657
iv.altep  0.2674968 0.2518374
ia.occlus 0.7945580 0.8189090
extra.ica 0.1793385 0.1667574

Standardize mean differences
               1 vs 2
sex       0.017479025
location  0.151168444
hx.isch   0.099032275
afib      0.055906317
dm        0.008673478
mrankin   0.062543164
iv.altep  0.111897009
ia.occlus 0.117394890
extra.ica 0.129370206
```

In this case, I have simulated the data to mirror the results in the published Table 1 for this study. In no way have I captured the full range of the real data, or any of the relationships in that data, so it's more important here to see what's available in the analysis, rather than to interpret it closely in the clinical context.

1.7 Exporting the Completed Table 1 from R to Excel or Word

Once you've built the table and are generally satisfied with it, you'll probably want to be able to drop it into Excel or Word for final cleanup.

1.7.1 Approach A: Save and open in Excel

One option is to **save the Table 1** to a `.csv` file within our `data` subfolder (note that the `data` folder must already exist), which you can then open directly in Excel. This is the approach I generally use. Note the addition of some `quote`, `noSpaces` and `printToggle` selections here.

```
fs.table1save <- print(att2,
      nonnormal = c("age", "nihss", "time.iv", "aspects", "time.rand"),
      exact = c("location", "mrankin"),
      quote = FALSE, noSpaces = TRUE, printToggle = FALSE)

write.csv(fs.table1save, file = "data/fs-table1.csv")
```

When I then open the `fs-table1.csv` file in Excel, it looks like this:

	A	B	C	D	E
1		Intervention	Control	p	test
2	n	233	267		
3	age (median [IQR])	65.80 [54.50, 76.00]	65.70 [55.75, 76.20]	0.579	nonnorm
4	sex = Male (%)	135 (57.9)	157 (58.8)	0.917	
5	nihss (median [IQR])	17.00 [14.00, 21.00]	18.00 [14.00, 22.00]	0.453	nonnorm
6	location = Right (%)	117 (50.2)	114 (42.7)	0.111	
7	hx.isch = Yes (%)	29 (12.4)	25 (9.4)	0.335	
8	afib = 1 (%)	66 (28.3)	69 (25.8)	0.601	
9	dm = 1 (%)	29 (12.4)	34 (12.7)	1	
10	mrankin (%)			0.922	
11	0	190 (81.5)	214 (80.1)		
12	1	21 (9.0)	29 (10.9)		
13	2	12 (5.2)	13 (4.9)		
14	> 2	10 (4.3)	11 (4.1)		
15	sbp (mean (sd))	146.03 (26.00)	145.00 (24.40)	0.647	
16	iv.altep = Yes (%)	203 (87.1)	242 (90.6)	0.267	
17	time.iv (median [IQR])	85.00 [67.00, 110.00]	87.00 [65.00, 116.00]	0.596	nonnorm
18	aspects (median [IQR])	9.00 [7.00, 10.00]	9.00 [8.00, 10.00]	0.075	nonnorm
19	ia.occlus (%)			0.795	
20	Intracranial ICA	1 (0.4)	3 (1.1)		
21	ICA with M1	59 (25.3)	75 (28.2)		
22	M1	154 (66.1)	165 (62.0)		
23	M2	18 (7.7)	21 (7.9)		
24	A1 or A2	1 (0.4)	2 (0.8)		
25	extra.ica = 1 (%)	75 (32.2)	70 (26.3)	0.179	
26	time.rand (median [IQR])	204.00 [152.00, 249.50]	196.00 [149.00, 266.00]	0.251	nonnorm
27	time.punc (median [IQR])	260.00 [212.00, 313.00]	NA [NA, NA]	NA	nonnorm
28					

And from here, I can either drop it directly into Word, or present it as is, or start tweaking it to meet formatting needs.

1.7.2 Approach B: Produce the Table so you can cut and paste it

```
print(att2,
      nonnormal = c("age", "nihss", "time.iv", "aspects", "time.rand"),
      exact = c("location", "mrankin"),
      quote = TRUE, noSpaces = TRUE)
```

This will look like a mess by itself, but if you:

1. copy and paste that mess into Excel
2. select Text to Columns from the Data menu
3. select Delimited, then Space and select Treat consecutive delimiters as one

you should get something usable again.

Or, in Word,

1. insert the text

2. select the text with your mouse
3. select Insert … Table … Convert Text to Table
4. place a quotation mark in the "Other" area under Separate text at …

After dropping blank columns, the result looks pretty good.

1.8 A Controlled Biological Experiment - The Blood-Brain Barrier

My source for the data and the following explanatory paragraph is page 307 from Ramsey and Schafer (2002). The original data come from Barnett et al. (1995).

> The human brain (and that of rats, coincidentally) is protected from the bacteria and toxins that course through the bloodstream by something called the blood-brain barrier. After a method of disrupting the barrier was developed, researchers tested this new mechanism, as follows. A series of 34 rats were inoculated with human lung cancer cells to induce brain tumors. After 9-11 days they were infused with either the barrier disruption (BD) solution or, as a control, a normal saline (NS) solution. Fifteen minutes later, the rats received a standard dose of a particular therapeutic antibody (L6-F(ab')2. The key measure of the effectiveness of transmission across the brain-blood barrier is the ratio of the antibody concentration in the brain tumor to the antibody concentration in normal tissue outside the brain. The rats were then sacrificed, and the amounts of antibody in the brain tumor and in normal tissue from the liver were measured. The study's primary objective is to determine whether the antibody concentration in the tumor increased when the blood-barrier disruption infusion was given, and if so, by how much?

1.9 The `bloodbrain.csv` file

Consider the data, available on the Data and Code page of our course website in the `bloodbrain.csv` file, which includes the following variables:

Variable	Description
`case`	identification number for the rat (1 - 34)
`brain`	an outcome: Brain tumor antibody count (per gram)
`liver`	an outcome: Liver antibody count (per gram)
`tlratio`	an outcome: tumor / liver concentration ratio
`solution`	the treatment: BD (barrier disruption) or NS (normal saline)
`sactime`	a design variable: Sacrifice time (hours; either 0.5, 3, 24 or 72)
`postin`	covariate: Days post-inoculation of lung cancer cells (9, 10 or 11)
`sex`	covariate: M or F
`wt.init`	covariate: Initial weight (grams)
`wt.loss`	covariate: Weight loss (grams)
`wt.tumor`	covariate: Tumor weight (10^{-4} grams)

And here's what the data look like in R.

```
bloodbrain
```

```
# A tibble: 34 x 11
    case  brain   liver tlratio solution sactime postin sex   wt.init
   <int>  <int>   <int>   <dbl> <fct>      <dbl>  <int> <fct>   <int>
 1     1  41081 1456164  0.0282 BD         0.500     10 F         239
```

```
 2     2  44286 1602171  0.0276 BD         0.500     10 F         225
 3     3 102926 1601936  0.0642 BD         0.500     10 F         224
 4     4  25927 1776411  0.0146 BD         0.500     10 F         184
 5     5  42643 1351184  0.0316 BD         0.500     10 F         250
 6     6  31342 1790863  0.0175 NS         0.500     10 F         196
 7     7  22815 1633386  0.0140 NS         0.500     10 F         200
 8     8  16629 1618757  0.0103 NS         0.500     10 F         273
 9     9  22315 1567602  0.0142 NS         0.500     10 F         216
10    10  77961 1060057  0.0735 BD         3.00      10 F         267
# ... with 24 more rows, and 2 more variables: wt.loss <dbl>,
#   wt.tumor <int>
```

1.10 A Table 1 for bloodbrain

Barnett et al. (1995) did not provide a Table 1 for these data, so let's build one to compare the two solutions (BD vs. NS) on the covariates and outcomes, plus the natural logarithm of the tumor/liver concentration ratio (tlratio). We'll opt to treat the sacrifice time (sactime) and the days post-inoculation of lung cancer cells (postin) as categorical rather than quantitative variables.

```
bloodbrain <- bloodbrain %>%
    mutate(logTL = log(tlratio))

dput(names(bloodbrain))
```

```
c("case", "brain", "liver", "tlratio", "solution", "sactime",
"postin", "sex", "wt.init", "wt.loss", "wt.tumor", "logTL")
```

OK - there's the list of variables we'll need. I'll put the outcomes at the bottom of the table.

```
bb.vars <- c("sactime", "postin", "sex", "wt.init", "wt.loss",
             "wt.tumor", "brain", "liver", "tlratio", "logTL")

bb.factors <- c("sactime", "sex", "postin")

bb.att1 <- CreateTableOne(data = bloodbrain,
                          vars = bb.vars,
                          factorVars = bb.factors,
                          strata = c("solution"))
summary(bb.att1)
```

```
     ### Summary of continuous variables ###

solution: BD
          n miss p.miss   mean    sd median    p25   p75    min   max
wt.init  17    0      0    243 3e+01  2e+02  2e+02 3e+02  2e+02 3e+02
wt.loss  17    0      0      3 5e+00  4e+00  1e+00 6e+00 -5e+00 1e+01
wt.tumor 17    0      0    157 8e+01  2e+02  1e+02 2e+02  2e+01 4e+02
brain    17    0      0  56043 3e+04  5e+04  4e+04 8e+04  6e+03 1e+05
liver    17    0      0 672577 7e+05  6e+05  2e+04 1e+06  2e+03 2e+06
tlratio  17    0      0      2 3e+00  1e-01  6e-02 3e+00  1e-02 9e+00
logTL    17    0      0     -1 2e+00 -2e+00 -3e+00 1e+00 -4e+00 2e+00
          skew kurt
wt.init  -0.39  0.7
wt.loss  -0.10  0.2
```

```
wt.tumor  0.53  1.0
brain     0.29 -0.6
liver     0.35 -1.7
tlratio   1.58  1.7
logTL     0.08 -1.7
------------------------------------------------------------
solution: NS
          n miss p.miss   mean    sd median    p25    p75    min   max
wt.init  17    0      0    240 3e+01  2e+02  2e+02  3e+02  2e+02 3e+02
wt.loss  17    0      0      4 4e+00  3e+00  2e+00  7e+00 -4e+00 1e+01
wt.tumor 17    0      0    209 1e+02  2e+02  2e+02  3e+02  3e+01 5e+02
brain    17    0      0  23887 1e+04  2e+04  1e+04  3e+04  1e+03 5e+04
liver    17    0      0 664975 7e+05  7e+05  2e+04  1e+06  9e+02 2e+06
tlratio  17    0      0      1 2e+00  5e-02  3e-02  9e-01  1e-02 7e+00
logTL    17    0      0     -2 2e+00 -3e+00 -3e+00 -7e-02 -5e+00 2e+00
          skew  kurt
wt.init   0.33 -0.48
wt.loss  -0.09  0.08
wt.tumor  0.63  0.77
brain     0.30 -0.35
liver     0.40 -1.56
tlratio   2.27  4.84
logTL     0.27 -1.61

p-values
             pNormal  pNonNormal
wt.init  0.807308940 0.641940278
wt.loss  0.683756156 0.876749808
wt.tumor 0.151510151 0.190482094
brain    0.001027678 0.002579901
liver    0.974853609 0.904045603
tlratio  0.320501715 0.221425879
logTL    0.351633525 0.221425879

Standardize mean differences
             1 vs 2
wt.init  0.08435244
wt.loss  0.14099823
wt.tumor 0.50397184
brain    1.23884159
liver    0.01089667
tlratio  0.34611465
logTL    0.32420504

================================================================================

     ### Summary of categorical variables ###

solution: BD
     var  n miss p.miss level freq percent cum.percent
 sactime 17    0    0.0   0.5    5    29.4        29.4
                          3    4    23.5        52.9
                         24    4    23.5        76.5
                         72    4    23.5       100.0
```

```
  postin 17    0    0.0     9    1     5.9         5.9
                           10   14    82.4        88.2
                           11    2    11.8       100.0

     sex 17    0    0.0     F   13    76.5        76.5
                            M    4    23.5       100.0

-------------------------------------------------------
solution: NS
     var  n miss p.miss level freq percent cum.percent
 sactime 17    0    0.0   0.5    4    23.5        23.5
                            3    5    29.4        52.9
                           24    4    23.5        76.5
                           72    4    23.5       100.0

  postin 17    0    0.0     9    2    11.8        11.8
                           10   13    76.5        88.2
                           11    2    11.8       100.0

     sex 17    0    0.0     F   13    76.5        76.5
                            M    4    23.5       100.0


p-values
          pApprox pExact
sactime 0.9739246      1
postin  0.8309504      1
sex     1.0000000      1

Standardize mean differences
           1 vs 2
sactime 0.1622214
postin  0.2098877
sex     0.0000000
```

Note that, in this particular case, the decisions we make about normality vs. non-normality (for quantitative variables) and the decisions we make about approximate vs. exact testing (for categorical variables) won't actually change the implications of the p values. Each approach gives similar results for each variable. Of course, that's not always true.

1.10.1 Generate final Table 1 for `bloodbrain`

I'll choose to treat `tlratio` and its logarithm as non-Normal, but otherwise, use t tests, but admittedly, that's an arbitrary decision, really.

```
print(bb.att1, nonnormal = c("tlratio", "logTL"))
```

```
                      Stratified by solution
                       BD                      NS
  n                           17                      17
  sactime (%)
     0.5                       5 (29.4)                4 (23.5)
     3                         4 (23.5)                5 (29.4)
     24                        4 (23.5)                4 (23.5)
```

```
     72                          4 (23.5)               4 (23.5)
  postin (%)
     9                           1 ( 5.9)               2 (11.8)
     10                         14 (82.4)              13 (76.5)
     11                          2 (11.8)               2 (11.8)
  sex = M (%)                    4 (23.5)               4 (23.5)
  wt.init (mean (sd))       242.82 (27.23)         240.47 (28.54)
  wt.loss (mean (sd))         3.34 (4.68)            3.94 (3.88)
  wt.tumor (mean (sd))      157.29 (84.00)         208.53 (116.68)
  brain (mean (sd))       56043.41 (33675.40)    23887.18 (14610.53)
  liver (mean (sd))      672577.35 (694479.58)  664975.47 (700773.13)
  tlratio (median [IQR])      0.12 [0.06, 2.84]      0.05 [0.03, 0.94]
  logTL (median [IQR])       -2.10 [-2.74, 1.04]    -2.95 [-3.41, -0.07]
                        Stratified by solution
                         p      test
  n
  sactime (%)             0.974
     0.5
     3
     24
     72
  postin (%)              0.831
     9
     10
     11
  sex = M (%)             1.000
  wt.init (mean (sd))     0.807
  wt.loss (mean (sd))     0.684
  wt.tumor (mean (sd))    0.152
  brain (mean (sd))       0.001
  liver (mean (sd))       0.975
  tlratio (median [IQR])  0.221 nonnorm
  logTL (median [IQR])    0.221 nonnorm
```

Or, we can get an Excel-readable version placed in a `data` subfolder, using

```
bb.t1 <- print(bb.att1, nonnormal = c("tlratio", "logTL"), quote = FALSE,
               noSpaces = TRUE, printToggle = FALSE)

write.csv(bb.t1, file = "data/bb-table1.csv")
```

which, when dropped into Excel, will look like this:

	A	B	C	D	E
1		BD	NS	p	test
2	n	17	17		
3	sex = M (%)	4 (23.5)	4 (23.5)	1	
4	sactime (%)			0.974	
5	0.5	5 (29.4)	4 (23.5)		
6	3	4 (23.5)	5 (29.4)		
7	24	4 (23.5)	4 (23.5)		
8	72	4 (23.5)	4 (23.5)		
9	postin (%)			0.831	
10	9	1 (5.9)	2 (11.8)		
11	10	14 (82.4)	13 (76.5)		
12	11	2 (11.8)	2 (11.8)		
13	wt.init (mean (sd))	242.82 (27.23)	240.47 (28.54)	0.807	
14	wt.loss (mean (sd))	3.34 (4.68)	3.94 (3.88)	0.684	
15	wt.tumor (mean (sd))	157.29 (84.00)	208.53 (116.68)	0.152	
16	brain (mean (sd))	56043.41 (33675.40)	23887.18 (14610.53)	0.001	
17	liver (mean (sd))	672577.35 (694479.58)	664975.47 (700773.13)	0.975	
18	tlratio (median [IQR])	0.12 [0.06, 2.84]	0.05 [0.03, 0.94]	0.221	nonnorm
19	logTL (median [IQR])	-2.10 [-2.74, 1.04]	-2.95 [-3.41, -0.07]	0.221	nonnorm
20					

One thing I would definitely clean up here, in practice, is to change the presentation of the p value for `sex` from 1 to > 0.99, or just omit it altogether. I'd also drop the `computer-ese` where possible, add units for the measures, round **a lot**, identify the outcomes carefully, and use notes to indicate deviations from the main approach.

1.10.2 A More Finished Version (after Cleanup in Word)

Table 1. Comparing Rats Receiving BD to those Receiving NS on Available Covariates and Design Variables, and Key Outcomes

	Barrier Disruption (BD: treatment)	Normal Saline (NS: control)	p
# of Rats	17	17	
Sex = Male	4 (23.5)	4 (23.5)	-
Sacrifice Time (hours)			0.97
0.5	5 (29.4)	4 (23.5)	
3	4 (23.5)	5 (29.4)	
24	4 (23.5)	4 (23.5)	
72	4 (23.5)	4 (23.5)	
Days post-inoculation of lung cancer cells			0.83
9	1 (5.9)	2 (11.8)	
10	14 (82.4)	13 (76.5)	
11	2 (11.8)	2 (11.8)	
Initial Weight (g)	243 (27)	240 (29)	0.81
Weight Loss (g)	3.3 (4.7)	3.9 (3.9)	0.68
Tumor Weight (10^{-4} g)	157.3 (84.0)	208.5 (116.7)	0.15
Key Outcomes: mean (sd) unless otherwise indicated			
Brain Tumor Antibody Count (per g)	56,043 (33,675)	23,887 (14,611)	0.001
Liver Antibody Count (per g)	672,577 (694,480)	664,975 (700,773)	0.98
Tumor/Liver Ratio (median [Q25, Q75])	0.12 [0.06, 2.84]	0.05 [0.03, 0.94]	0.22
Natural Log of Tumor/Liver Ratio (median [Q25, Q75])	-2.10 [-2.74, 1.04]	-2.95 [-3.41, -0.07]	0.22

Table 1 Notes:

- Categorical variables are summarized with counts, percentages and p values based on approximate chi-square tests.
- Continuous variables, unless otherwise indicated, are summarized with means, standard deviations and p values based on t tests.
- The Tumor / Liver ratio and its natural logarithm are summarized with the median and quartiles and a p value from a non-parametric (Wilcoxon signed rank) test.

Chapter 2

Linear Regression on a small SMART data set

2.1 BRFSS and SMART

The Centers for Disease Control analyzes Behavioral Risk Factor Surveillance System (BRFSS) survey data for specific metropolitan and micropolitan statistical areas (MMSAs) in a program called the Selected Metropolitan/Micropolitan Area Risk Trends of BRFSS (SMART BRFSS.)

In this work, we will focus on data from the 2016 SMART, and in particular on data from the Cleveland-Elyria, OH, Metropolitan Statistical Area. The purpose of this survey is to provide localized health information that can help public health practitioners identify local emerging health problems, plan and evaluate local responses, and efficiently allocate resources to specific needs.

2.1.1 Key resources

- the full data are available in the form of the 2016 SMART BRFSS MMSA Data, found in a zipped SAS Transport Format file. The data were released in August 2017.
- the MMSA Variable Layout PDF which simply lists the variables included in the data file
- the Calculated Variables PDF which describes the risk factors by data variable names - there is also an online summary matrix of these calculated variables, as well.
- the lengthy 2016 Survey Questions PDF which lists all questions asked as part of the BRFSS in 2016
- the enormous Codebook for the 2016 BRFSS Survey PDF which identifies the variables by name for us.

Later this term, we'll use all of those resources to help construct a more complete data set than we'll study today. I'll also demonstrate how I built the `smartcle1` data set that we'll use in this Chapter.

2.2 The `smartcle1` data: Cookbook

The `smartcle1.csv` data file available on the Data and Code page of our website describes information on 11 variables for 1036 respondents to the BRFSS 2016, who live in the Cleveland-Elyria, OH, Metropolitan Statistical Area. The variables in the `smartcle1.csv` file are listed below, along with (in some cases) the BRFSS items that generate these responses.

Variable	Description
`SEQNO`	respondent identification number (all begin with 2016)

Variable	Description
physhealth	Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 days was your physical health not good?
menthealth	Now thinking about your mental health, which includes stress, depression, and problems with emotions, for how many days during the past 30 days was your mental health not good?
poorhealth	During the past 30 days, for about how many days did poor physical or mental health keep you from doing your usual activities, such as self-care, work, or recreation?
genhealth	Would you say that in general, your health is … (five categories: Excellent, Very Good, Good, Fair or Poor)
bmi	Body mass index, in kg/m^2
female	Sex, 1 = female, 0 = male
internet30	Have you used the internet in the past 30 days? (1 = yes, 0 = no)
exerany	During the past month, other than your regular job, did you participate in any physical activities or exercises such as running, calisthenics, golf, gardening, or walking for exercise? (1 = yes, 0 = no)
sleephrs	On average, how many hours of sleep do you get in a 24-hour period?
alcdays	How many days during the past 30 days did you have at least one drink of any alcoholic beverage such as beer, wine, a malt beverage or liquor?

```
str(smartcle1)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame':   1036 obs. of  11 variables:
 $ SEQNO     : num  2.02e+09 2.02e+09 2.02e+09 2.02e+09 2.02e+09 ...
 $ physhealth: int  0 0 1 0 5 4 2 2 0 0 ...
 $ menthealth: int  0 0 5 0 0 18 0 3 0 0 ...
 $ poorhealth: int  NA NA 0 NA 0 6 0 0 NA NA ...
 $ genhealth : Factor w/ 5 levels "1_Excellent",..: 2 1 2 3 1 2 3 3 2 3 ...
 $ bmi       : num  26.7 23.7 26.9 21.7 24.1 ...
 $ female    : int  1 0 0 1 0 0 1 1 0 0 ...
 $ internet30: int  1 1 1 1 1 1 1 1 1 1 ...
 $ exerany   : int  1 1 0 1 1 1 1 1 1 0 ...
 $ sleephrs  : int  6 6 8 9 7 5 9 7 7 7 ...
 $ alcdays   : int  1 4 4 3 2 28 4 2 4 25 ...
```

2.3 smartcle2: Omitting Missing Observations: Complete-Case Analyses

For the purpose of fitting our first few models, we will eliminate the missingness problem, and look only at the *complete cases* in our smartcle1 data. We will discuss methods for imputing missing data later in these Notes.

To inspect the missingness in our data, we might consider using the skim function from the skimr package. We'll exclude the respondent identifier code (SEQNO) from this summary as uninteresting.

```
skim_with(numeric = list(hist = NULL), integer = list(hist = NULL))
## above line eliminates the sparkline histograms
## it can be commented out when working in the console,
## but I need it to produce the Notes without errors right now
```

```
smartcle1 %>%
    skim(-SEQNO)
```

```
Skim summary statistics
 n obs: 1036
 n variables: 11

Variable type: factor
  variable missing complete    n n_unique
 genhealth       3     1033 1036        5
                           top_counts ordered
 2_V: 350, 3_G: 344, 1_E: 173, 4_F: 122   FALSE

Variable type: integer
   variable missing complete    n mean   sd p0 p25 median p75 p100
    alcdays      46      990 1036 4.65 8.05  0   0      1   4   30
    exerany       3     1033 1036 0.76 0.43  0   1      1   1    1
     female       0     1036 1036 0.6  0.49  0   0      1   1    1
 internet30       6     1030 1036 0.81 0.39  0   1      1   1    1
 menthealth      11     1025 1036 2.72 6.82  0   0      0   2   30
 physhealth      17     1019 1036 3.97 8.67  0   0      0   2   30
 poorhealth     543      493 1036 4.07 8.09  0   0      0   3   30
   sleephrs       8     1028 1036 7.02 1.53  1   6      7   8   20

Variable type: numeric
 variable missing complete    n  mean   sd    p0  p25 median   p75  p100
      bmi      84      952 1036 27.89 6.47 12.71 23.7  26.68 30.53 66.06
```

Now, we'll create a new tibble called `smartcle2` which contains every variable except `poorhealth`, and which includes all respondents with complete data on the variables (other than `poorhealth`). We'll store those observations with complete data in the `smartcle2` tibble.

```
smartcle2 <- smartcle1 %>%
    select(-poorhealth) %>%
    filter(complete.cases(.))

smartcle2
```

```
# A tibble: 896 x 10
     SEQNO physhealth menthealth genhealth   bmi female internet30 exerany
     <dbl>      <int>      <int> <fct>     <dbl>  <int>      <int>   <int>
 1  2.02e9          0          0 2_VeryGo~  26.7      1          1       1
 2  2.02e9          0          0 1_Excell~  23.7      0          1       1
 3  2.02e9          1          5 2_VeryGo~  26.9      0          1       0
 4  2.02e9          0          0 3_Good     21.7      1          1       1
 5  2.02e9          5          0 1_Excell~  24.1      0          1       1
 6  2.02e9          4         18 2_VeryGo~  27.6      0          1       1
 7  2.02e9          2          0 3_Good     25.7      1          1       1
 8  2.02e9          2          3 3_Good     28.5      1          1       1
 9  2.02e9          0          0 2_VeryGo~  28.6      0          1       1
10  2.02e9          0          0 3_Good     23.1      0          1       0
# ... with 886 more rows, and 2 more variables: sleephrs <int>,
#   alcdays <int>
```

Note that there are only 896 respondents with **complete** data on the 10 variables (excluding `poorhealth`) in the `smartcle2` tibble, as compared to our original `smartcle1` data which described 1036 respondents and

11 variables, but with lots of missing data.

2.4 Summarizing the `smartcle2` data numerically

2.4.1 The New Toy: The `skim` function

```
skim(smartcle2, -SEQNO)
```

```
Skim summary statistics
 n obs: 896
 n variables: 10

Variable type: factor
  variable missing complete   n n_unique
 genhealth       0      896 896        5
                           top_counts ordered
 2_V: 306, 3_G: 295, 1_E: 155, 4_F: 102   FALSE

Variable type: integer
   variable missing complete   n mean   sd p0 p25 median p75 p100
    alcdays       0      896 896 4.83 8.14  0   0      1   5   30
    exerany       0      896 896 0.77 0.42  0   1      1   1    1
     female       0      896 896 0.58 0.49  0   0      1   1    1
 internet30       0      896 896 0.81 0.39  0   1      1   1    1
 menthealth       0      896 896 2.69 6.72  0   0      0   2   30
 physhealth       0      896 896 3.99 8.64  0   0      0   2   30
   sleephrs       0      896 896 7.02 1.48  1   6      7   8   20

Variable type: numeric
 variable missing complete   n  mean   sd    p0  p25 median   p75  p100
      bmi       0      896 896 27.87 6.33 12.71 23.7   26.8 30.53 66.06
```

2.4.2 The usual `summary` for a data frame

Of course, we can use the usual `summary` to get some basic information about the data.

```
summary(smartcle2)
```

```
     SEQNO             physhealth      menthealth          genhealth
 Min.   :2.016e+09   Min.   : 0.00   Min.   : 0.000   1_Excellent:155
 1st Qu.:2.016e+09   1st Qu.: 0.00   1st Qu.: 0.000   2_VeryGood :306
 Median :2.016e+09   Median : 0.00   Median : 0.000   3_Good     :295
 Mean   :2.016e+09   Mean   : 3.99   Mean   : 2.693   4_Fair     :102
 3rd Qu.:2.016e+09   3rd Qu.: 2.00   3rd Qu.: 2.000   5_Poor     : 38
 Max.   :2.016e+09   Max.   :30.00   Max.   :30.000
      bmi            female         internet30        exerany
 Min.   :12.71   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:23.70   1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:1.0000
 Median :26.80   Median :1.0000   Median :1.0000   Median :1.0000
 Mean   :27.87   Mean   :0.5848   Mean   :0.8147   Mean   :0.7667
 3rd Qu.:30.53   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :66.06   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```
    sleephrs         alcdays
 Min.   : 1.000   Min.   : 0.000
 1st Qu.: 6.000   1st Qu.: 0.000
 Median : 7.000   Median : 1.000
 Mean   : 7.022   Mean   : 4.834
 3rd Qu.: 8.000   3rd Qu.: 5.000
 Max.   :20.000   Max.   :30.000
```

2.4.3 The `describe` function in `Hmisc`

Or we can use the `describe` function from the `Hmisc` package.

```
Hmisc::describe(select(smartcle2, bmi, genhealth, female))
```

```
select(smartcle2, bmi, genhealth, female)

 3  Variables      896  Observations
--------------------------------------------------------------------------------
bmi
       n  missing distinct     Info     Mean      Gmd      .05      .10
     896        0      467        1    27.87    6.572    20.06    21.23
     .25      .50      .75      .90      .95
   23.70    26.80    30.53    35.36    39.30

lowest : 12.71 13.34 14.72 16.22 17.30, highest: 56.89 57.04 60.95 61.84 66.06
--------------------------------------------------------------------------------
genhealth
       n  missing distinct
     896        0        5

Value      1_Excellent  2_VeryGood      3_Good      4_Fair      5_Poor
Frequency          155         306         295         102          38
Proportion       0.173       0.342       0.329       0.114       0.042
--------------------------------------------------------------------------------
female
       n  missing distinct     Info      Sum     Mean      Gmd
     896        0        2    0.728      524   0.5848   0.4862

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

2.5 Counting as exploratory data analysis

Counting things can be amazingly useful.

2.5.1 How many respondents had exercised in the past 30 days? Did this vary by sex?

```
smartcle2 %>% count(female, exerany) %>% mutate(percent = 100*n / sum(n))
```

```
# A tibble: 4 x 4
  female exerany     n percent
   <int>   <int> <int>   <dbl>
```

```
1      0       0    64    7.14
2      0       1   308   34.4
3      1       0   145   16.2
4      1       1   379   42.3
```

so we know now that 42.3% of the subjects in our data were women who exercised. Suppose that instead we want to find the percentage of exercisers within each sex...

```
smartcle2 %>%
    count(female, exerany) %>%
    group_by(female) %>%
    mutate(prob = 100*n / sum(n))
```

```
# A tibble: 4 x 4
# Groups:   female [2]
  female exerany     n  prob
   <int>   <int> <int> <dbl>
1      0       0    64  17.2
2      0       1   308  82.8
3      1       0   145  27.7
4      1       1   379  72.3
```

and now we know that 82.8% of the males exercised at least once in the last 30 days, as compared to 72.3% of the females.

2.5.2 What's the distribution of `sleephrs`?

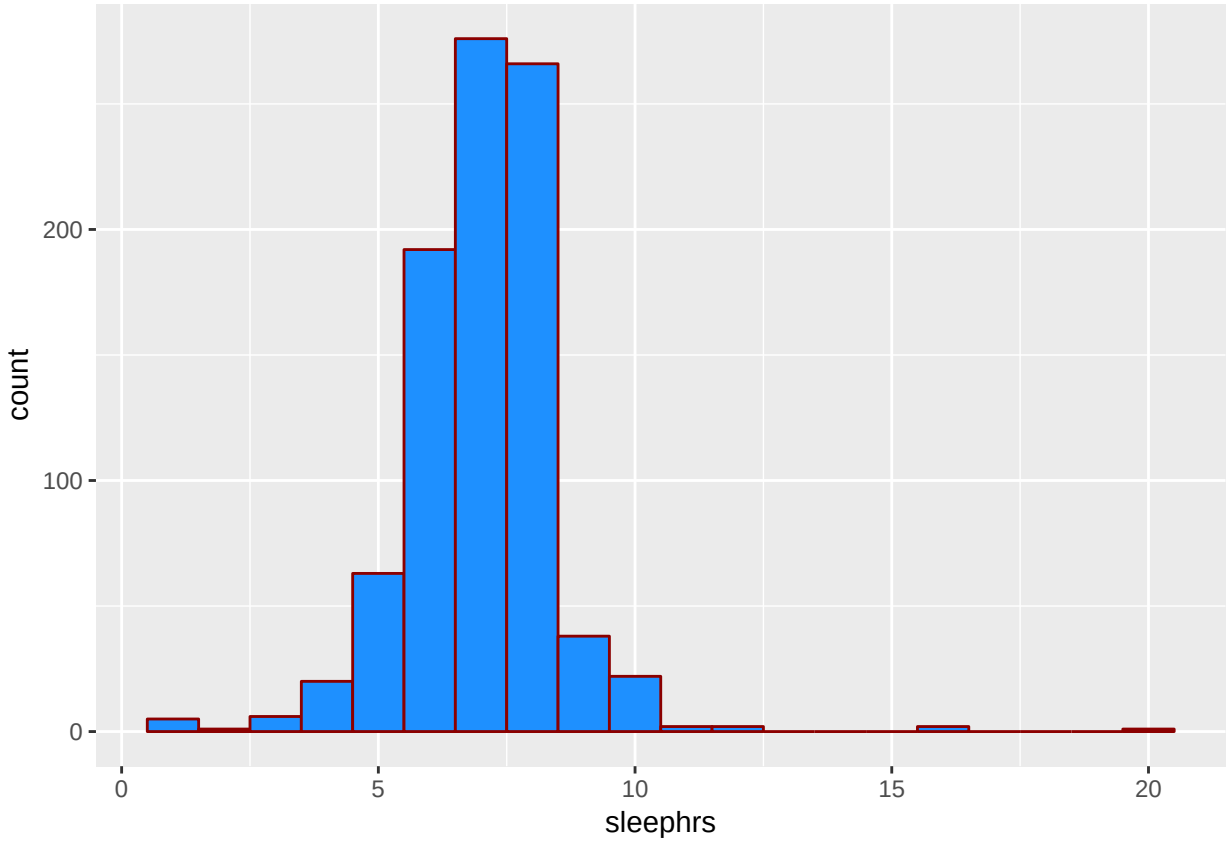
We can count quantitative variables with discrete sets of possible values, like `sleephrs`, which is captured as an integer (that must fall between 0 and 24.)

```
smartcle2 %>% count(sleephrs)
```

```
# A tibble: 14 x 2
   sleephrs     n
      <int> <int>
 1        1     5
 2        2     1
 3        3     6
 4        4    20
 5        5    63
 6        6   192
 7        7   276
 8        8   266
 9        9    38
10       10    22
11       11     2
12       12     2
13       16     2
14       20     1
```

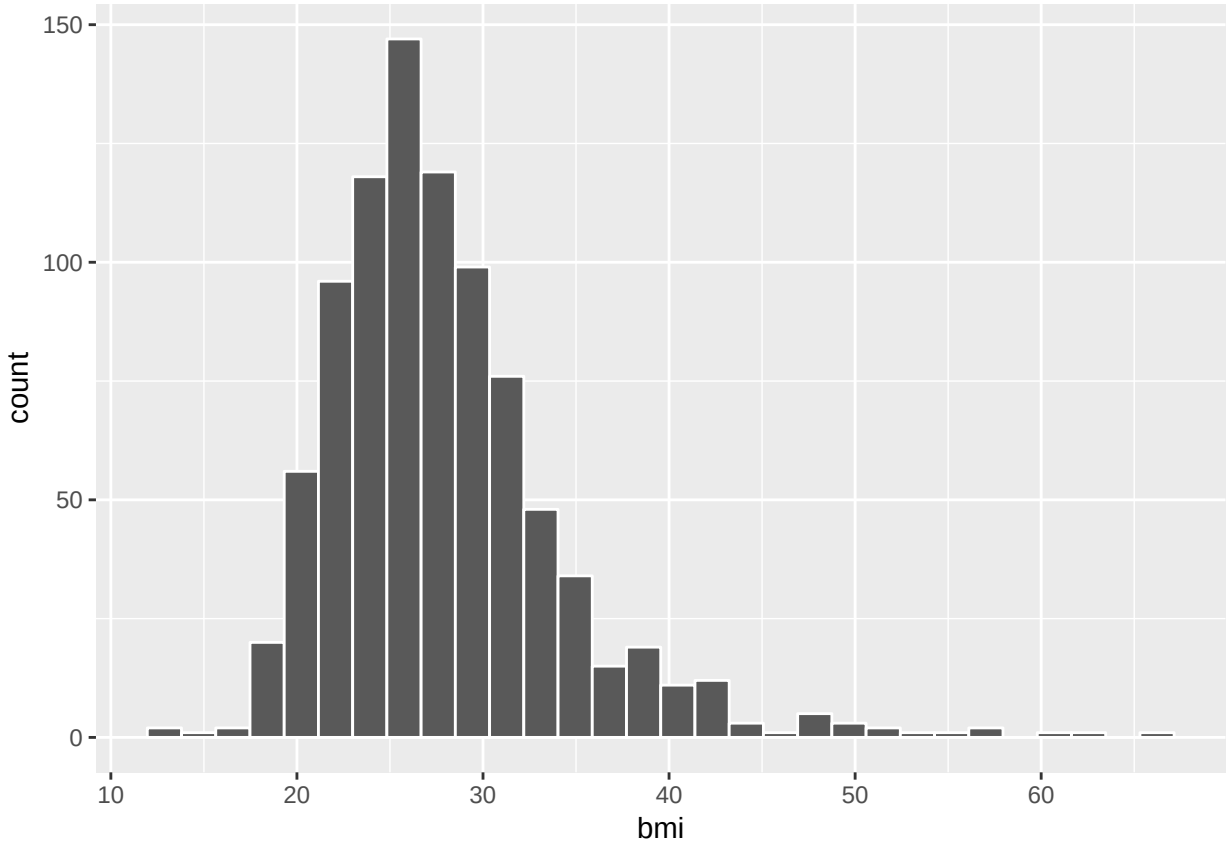
Of course, a natural summary of a quantitative variable like this would be graphical.

```
ggplot(smartcle2, aes(sleephrs)) +
    geom_histogram(binwidth = 1, fill = "dodgerblue", col = "darkred")
```



2.5.3 What's the distribution of `BMI`?

```
ggplot(smartcle2, aes(bmi)) +
    geom_histogram(bins = 30, col = "white")
```



2.5.4 How many of the respondents have a BMI below 30?

```
smartcle2 %>% count(bmi < 30) %>% mutate(proportion = n / sum(n))
```

```
# A tibble: 2 x 3
  `bmi < 30`     n proportion
  <lgl>      <int>      <dbl>
1 F            253      0.282
2 T            643      0.718
```

2.5.5 How many of the respondents who have a BMI $<$ 30 exercised?

```
smartcle2 %>% count(exerany, bmi < 30) %>%
    group_by(exerany) %>%
    mutate(percent = 100*n/sum(n))
```

```
# A tibble: 4 x 4
# Groups:   exerany [2]
  exerany `bmi < 30`     n percent
    <int> <lgl>      <int>   <dbl>
1       0 F             88    42.1
2       0 T            121    57.9
3       1 F            165    24.0
4       1 T            522    76.0
```

2.5.6 Is obesity associated with sex, in these data?

```
smartcle2 %>% count(female, bmi < 30) %>%
    group_by(female) %>%
    mutate(percent = 100*n/sum(n))
```

```
# A tibble: 4 x 4
# Groups:   female [2]
  female `bmi < 30`     n percent
   <int> <lgl>      <int>   <dbl>
1      0 F            105    28.2
2      0 T            267    71.8
3      1 F            148    28.2
4      1 T            376    71.8
```

2.5.7 Comparing `sleephrs` summaries by obesity status

Can we compare the `sleephrs` means, medians and 75th percentiles for respondents whose BMI is below 30 to the respondents whose BMI is not?

```
smartcle2 %>%
    group_by(bmi < 30) %>%
    summarize(mean(sleephrs), median(sleephrs),
              q75 = quantile(sleephrs, 0.75))
```

```
# A tibble: 2 x 4
  `bmi < 30` `mean(sleephrs)` `median(sleephrs)`   q75
  <lgl>                 <dbl>              <int> <dbl>
1 F                      6.93                  7  8.00
2 T                      7.06                  7  8.00
```

2.5.8 The `skim` function within a pipe

The **skim** function works within pipes and with the other `tidyverse` functions.

```
smartcle2 %>%
    group_by(exerany) %>%
    skim(bmi, sleephrs)
```

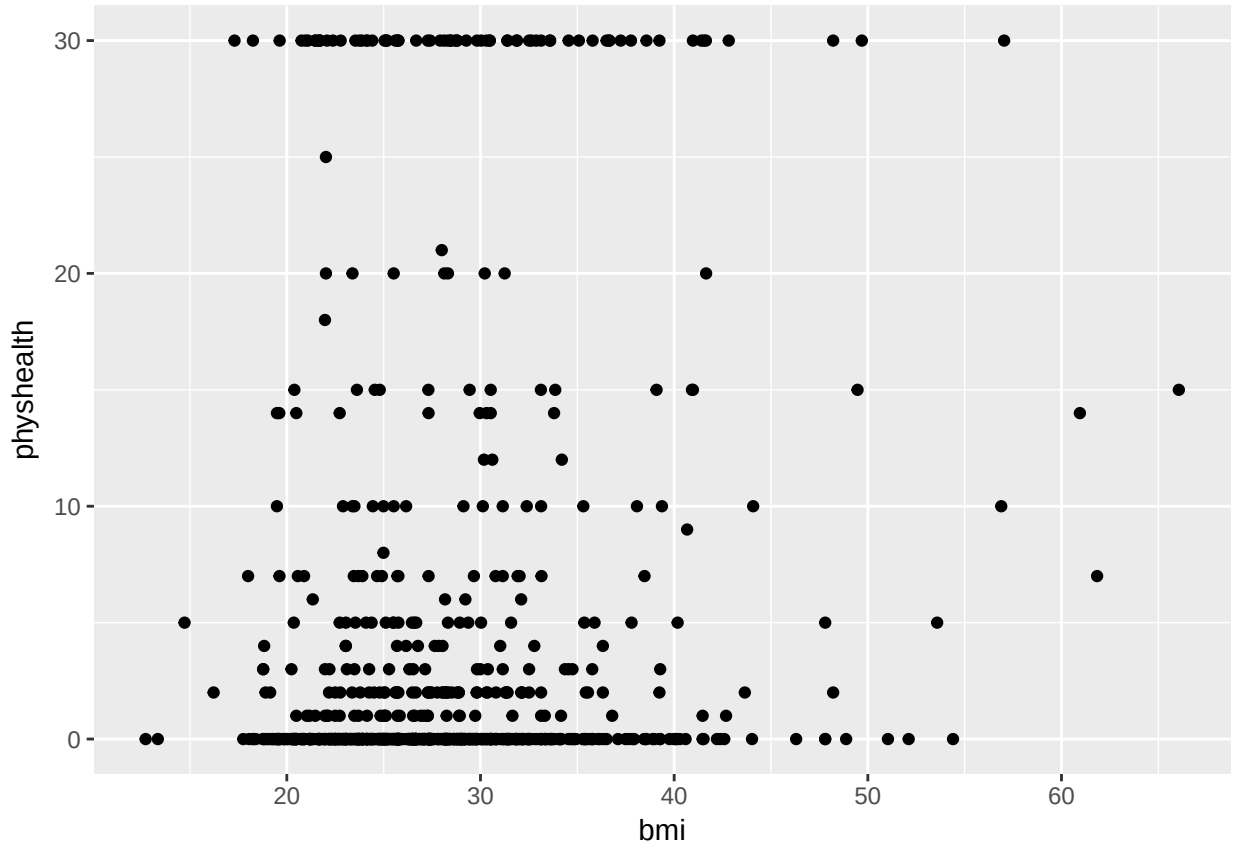
```
Skim summary statistics
 n obs: 896
 n variables: 10
 group variables: exerany

Variable type: integer
 exerany variable missing complete   n mean   sd p0 p25 median p75 p100
       0 sleephrs       0      209 209 7    1.85  1   6      7   8   20
       1 sleephrs       0      687 687 7.03 1.34  1   6      7   8   16

Variable type: numeric
 exerany variable missing complete   n  mean   sd    p0   p25 median   p75
       0      bmi       0      209 209 29.57 7.46 18    24.11  28.49 33.13
       1      bmi       0      687 687 27.35 5.84 12.71 23.7   26.52 29.81
  p100
```

```
66.06
60.95
```

2.6 First Modeling Attempt: Can `bmi` predict `physhealth`?

We'll start with an effort to predict `physhealth` using `bmi`. A natural graph would be a scatterplot.

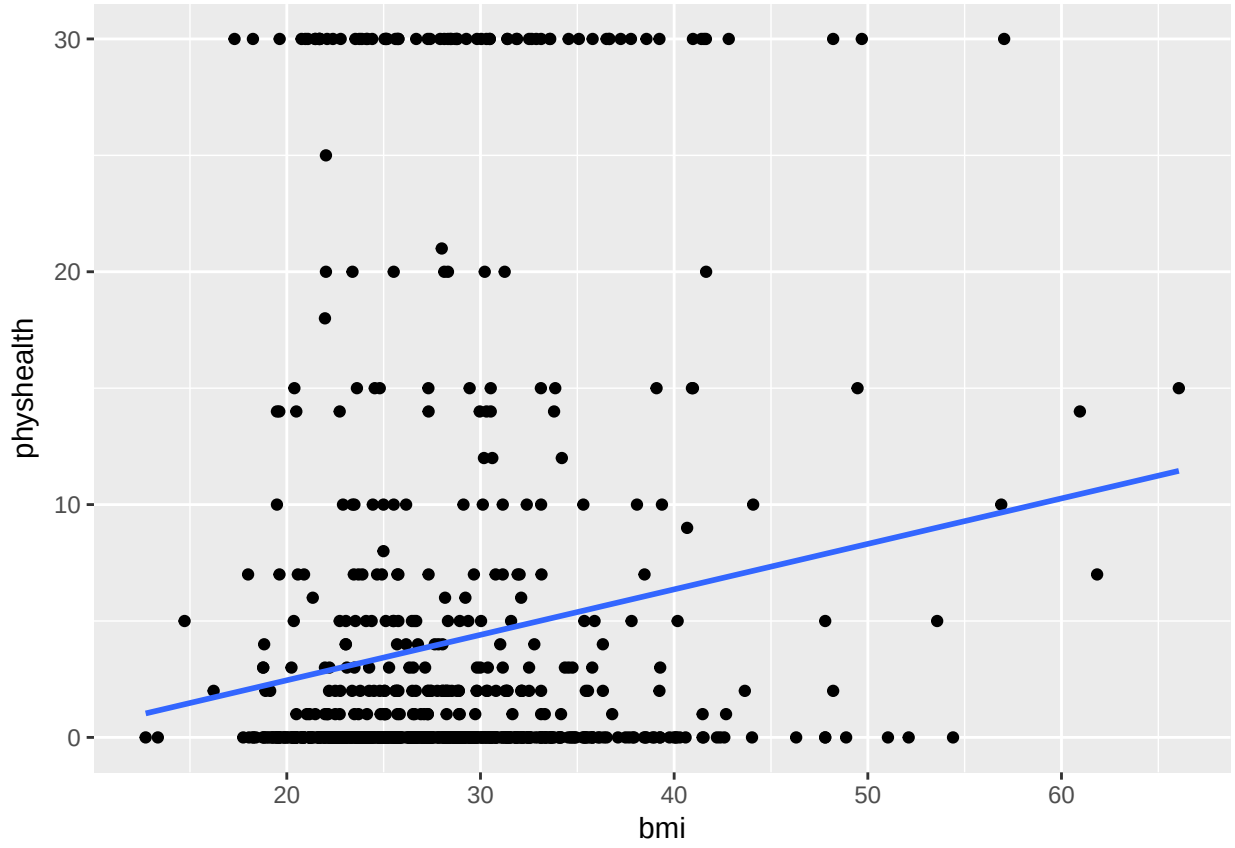
```
ggplot(data = smartcle2, aes(x = bmi, y = physhealth)) +
    geom_point()
```



A good question to ask ourselves here might be: "In what BMI range can we make a reasonable prediction of `physhealth`?"

Now, we might take the plot above and add a simple linear model ...

```
ggplot(data = smartcle2, aes(x = bmi, y = physhealth)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE)
```



which shows the same least squares regression model that we can fit with the `lm` command.

2.6.1 Fitting a Simple Regression Model

```
model_A <- lm(physhealth ~ bmi, data = smartcle2)

model_A
```

```

Call:
lm(formula = physhealth ~ bmi, data = smartcle2)

Coefficients:
(Intercept)          bmi
    -1.4514       0.1953
```

```
summary(model_A)
```

```

Call:
lm(formula = physhealth ~ bmi, data = smartcle2)

Residuals:
   Min     1Q Median     3Q    Max
-9.171 -4.057 -3.193 -1.576 28.073
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.45143    1.29185  -1.124    0.262
bmi          0.19527    0.04521   4.319 1.74e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.556 on 894 degrees of freedom
Multiple R-squared:  0.02044,   Adjusted R-squared:  0.01934
F-statistic: 18.65 on 1 and 894 DF,  p-value: 1.742e-05
```

```
confint(model_A, level = 0.95)
```

```
                2.5 %    97.5 %
(Intercept) -3.9868457 1.0839862
bmi          0.1065409 0.2840068
```

The model coefficients can be obtained by printing the model object, and the `summary` function provides several useful descriptions of the model's residuals, its statistical significance, and quality of fit.

2.6.2 Model Summary for a Simple (One-Predictor) Regression

The fitted model predicts `physhealth` with the equation -1.45 + 0.195*`bmi`, as we can read off from the model coefficients.

Each of the 896 respondents included in the `smartcle2` data makes a contribution to this model.

2.6.2.1 Residuals

Suppose Harry is one of the people in that group, and Harry's data is `bmi` = 20, and `physhealth` = 3.

- Harry's *observed* value of `physhealth` is just the value we have in the data for them, in this case, observed `physhealth` = 3 for Harry.
- Harry's *fitted* or *predicted* `physhealth` value is the result of calculating -1.45 + 0.195*`bmi` for Harry. So, if Harry's BMI was 20, then Harry's predicted `physhealth` value is -1.45 + (0.195)(20) = 2.45.
- The *residual* for Harry is then his *observed* outcome minus his *fitted* outcome, so Harry has a residual of 3 - 2.45 = 0.55.
- Graphically, a residual represents vertical distance between the observed point and the fitted regression line.
- Points above the regression line will have positive residuals, and points below the regression line will have negative residuals. Points on the line have zero residuals.

The residuals are summarized at the top of the `summary` output for linear model.

- The mean residual will always be zero in an ordinary least squares model, but a five number summary of the residuals is provided by the summary, as is an estimated standard deviation of the residuals (called here the Residual standard error.)
- In the `smartcle2` data, the minimum residual was -9.17, so for one subject, the observed value was 9.17 days smaller than the predicted value. This means that the prediction was 9.17 days too large for that subject.
- Similarly, the maximum residual was 28.07 days, so for one subject the prediction was 28.07 days too small. Not a strong performance.
- In a least squares model, the residuals are assumed to follow a Normal distribution, with mean zero, and standard deviation (for the `smartcle2` data) of about 8.6 days. Thus, by the definition of a Normal distribution, we'd expect
- about 68% of the residuals to be between -8.6 and +8.6 days,

- about 95% of the residuals to be between -17.2 and +17.2 days,
- about all (99.7%) of the residuals to be between -25.8 and +25.8 days.

2.6.2.2 Coefficients section

The `summary` for a linear model shows Estimates, Standard Errors, t values and p values for each coefficient fit.

- The Estimates are the point estimates of the intercept and slope of `bmi` in our model.
- In this case, our estimated slope is 0.195, which implies that if Harry's BMI is 20 and Sally's BMI is 21, we predict that Sally's `physhealth` will be 0.195 days larger than Harry's.
- The Standard Errors are also provided for each estimate. We can create rough 95% confidence intervals by adding and subtracting two standard errors from each coefficient, or we can get a slightly more accurate answer with the `confint` function.
- Here, the 95% confidence interval for the slope of `bmi` is estimated to be (0.11, 0.28). This is a good measure of the uncertainty in the slope that is captured by our model. We are 95% confident in the process of building this interval, but this doesn't mean we're 95% sure that the true slope is actually in that interval.

Also available are a t value (just the Estimate divided by the Standard Error) and the appropriate p value for testing the null hypothesis that the true value of the coefficient is 0 against a two-tailed alternative.

- If a slope coefficient is statistically significantly different from 0, this implies that 0 will not be part of the uncertainty interval obtained through `confint`.
- If the slope was zero, it would suggest that `bmi` would add no predictive value to the model. But that's unlikely here.

If the `bmi` slope coefficient is associated with a small p value, as in the case of our `model_A`, it suggests that the model including `bmi` is statistically significantly better at predicting `physhealth` than the model without `bmi`.

- Without `bmi` our `model_A` would become an *intercept-only* model, in this case, which would predict the mean `physhealth` for everyone, regardless of any other information.

2.6.2.3 Model Fit Summaries

The `summary` of a linear model also displays:

- The residual standard error and associated degrees of freedom for the residuals.
- For a simple (one-predictor) least regression like this, the residual degrees of freedom will be the sample size minus 2.
- The multiple R-squared (or coefficient of determination)
- This is interpreted as the proportion of variation in the outcome (`physhealth`) accounted for by the model, and will always fall between 0 and 1 as a result.
- Our model_A accounts for a mere 2% of the variation in `physhealth`.
- The Adjusted R-squared value "adjusts" for the size of our model in terms of the number of coefficients included in the model.
- The adjusted R-squared will always be less than the Multiple R-squared.
- We still hope to find models with relatively large adjusted R^2 values.
- In particular, we hope to find models where the adjusted R^2 isn't substantially less than the Multiple R-squared.
- The adjusted R-squared is usually a better estimate of likely performance of our model in new data than is the Multiple R-squared.
- The adjusted R-squared result is no longer interpretable as a proportion of anything - in fact, it can fall below 0.

- We can obtain the adjusted R^2 from the raw R^2, the number of observations N and the number of predictors p included in the model, as follows:

$$R^2_{adj} = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1},$$

- The F statistic and p value from a global ANOVA test of the model.
  - Obtaining a statistically significant result here is usually pretty straightforward, since the comparison is between our model, and a model which simply predicts the mean value of the outcome for everyone.
  - In a simple (one-predictor) linear regression like this, the t statistic for the slope is just the square root of the F statistic, and the resulting p values for the slope's t test and for the global F test will be identical.
- To see the complete ANOVA F test for this model, we can run `anova(model_A)`.

```
anova(model_A)
```

```
Analysis of Variance Table

Response: physhealth
           Df Sum Sq Mean Sq F value    Pr(>F)
bmi         1   1366  1365.5  18.655 1.742e-05 ***
Residuals 894  65441    73.2
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.6.3 Using the `broom` package

The `broom` package has three functions of particular use in a linear regression model:

2.6.3.1 The `tidy` function

`tidy` builds a data frame/tibble containing information about the coefficients in the model, their standard errors, t statistics and p values.

```
tidy(model_A)
```

```
         term   estimate  std.error statistic      p.value
1 (Intercept) -1.4514298 1.29185199 -1.123526 2.615156e-01
2         bmi  0.1952739 0.04521145  4.319125 1.741859e-05
```

2.6.3.2 The `glance` function

glance‘ builds a data frame/tibble containing summary statistics about the model, including

- the (raw) multiple R^2 and adjusted R^2
- `sigma` which is the residual standard error
- the F `statistic`, `p.value` model `df` and `df.residual` associated with the global ANOVA test, plus
- several statistics that will be useful in comparing models down the line:
- the model's log likelihood function value, `logLik`
- the model's Akaike's Information Criterion value, `AIC`
- the model's Bayesian Information Criterion value, `BIC`
- and the model's `deviance` statistic

```
glance(model_A)
```

```
  r.squared adj.r.squared    sigma statistic      p.value df    logLik
1 0.02044019    0.01934449 8.555737  18.65484 1.741859e-05  2 -3193.723
       AIC     BIC deviance df.residual
1 6393.446 6407.84 65441.36         894
```

2.6.3.3 The `augment` function

`augment` builds a data frame/tibble which adds fitted values, residuals and other diagnostic summaries that describe each observation to the original data used to fit the model, and this includes

- `.fitted` and `.resid`, the fitted and residual values, in addition to
- `.hat`, the leverage value for this observation
- `.cooksd`, the Cook's distance measure of *influence* for this observation
- `.stdresid`, the standardized residual (think of this as a z-score - a measure of the residual divided by its associated standard deviation `.sigma`)
- and `se.fit` which will help us generate prediction intervals for the model downstream

Note that each of the new columns begins with `.` to avoid overwriting any data.

```
head(augment(model_A))
```

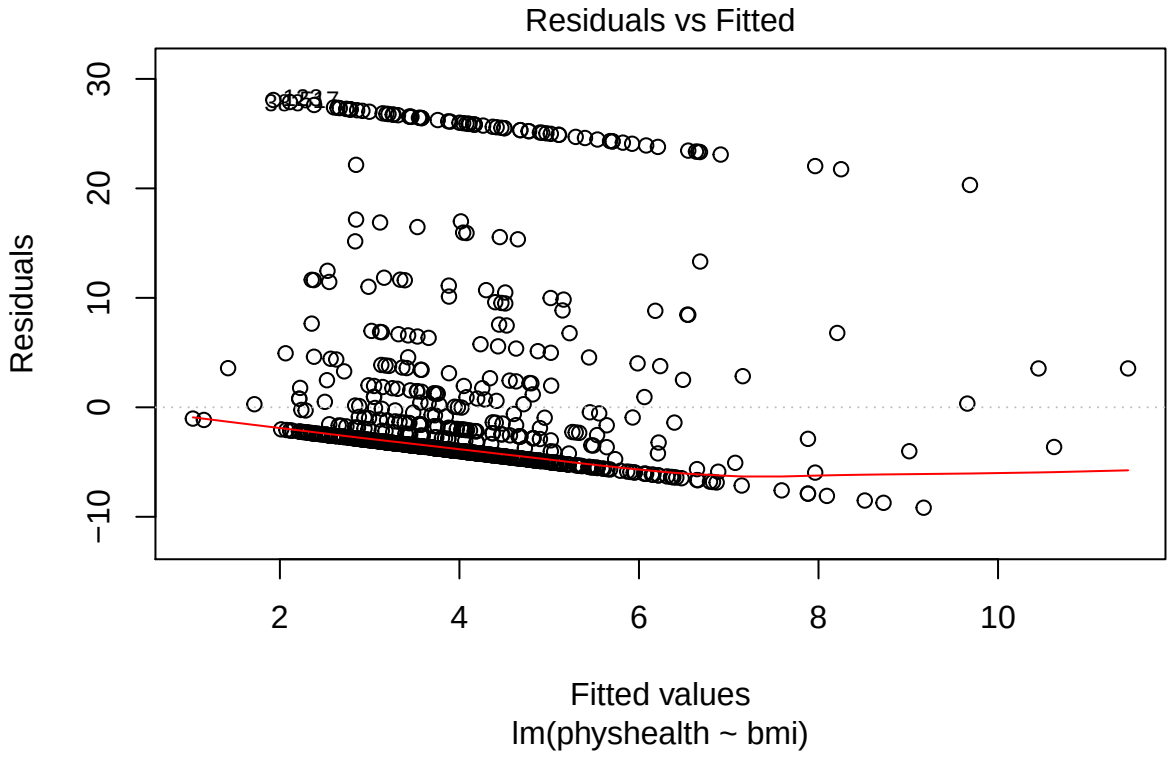
```
  physhealth   bmi  .fitted   .se.fit      .resid        .hat   .sigma
1          0 26.69 3.760430 0.2907252 -3.76043009 0.001154651 8.559600
2          0 23.70 3.176561 0.3422908 -3.17656119 0.001600574 8.559865
3          1 26.92 3.805343 0.2890054 -2.80534308 0.001141030 8.560010
4          0 21.66 2.778202 0.4005101 -2.77820248 0.002191352 8.560020
5          5 24.09 3.252718 0.3329154  1.74728200 0.001514095 8.560326
6          4 27.64 3.945940 0.2860087  0.05405972 0.001117490 8.560526
       .cooksd   .std.resid
1 1.117852e-04 -0.439775451
2 1.106717e-04 -0.371575999
3 6.147744e-05 -0.328077528
4 1.160381e-04 -0.325074461
5 3.167016e-05  0.204378225
6 2.235722e-08  0.006322069
```

For more on the `broom` package, you may want to look at this vignette.

2.6.4 How does the model do? (Residuals vs. Fitted Values)

- Remember that the R^2 value was about 2%.

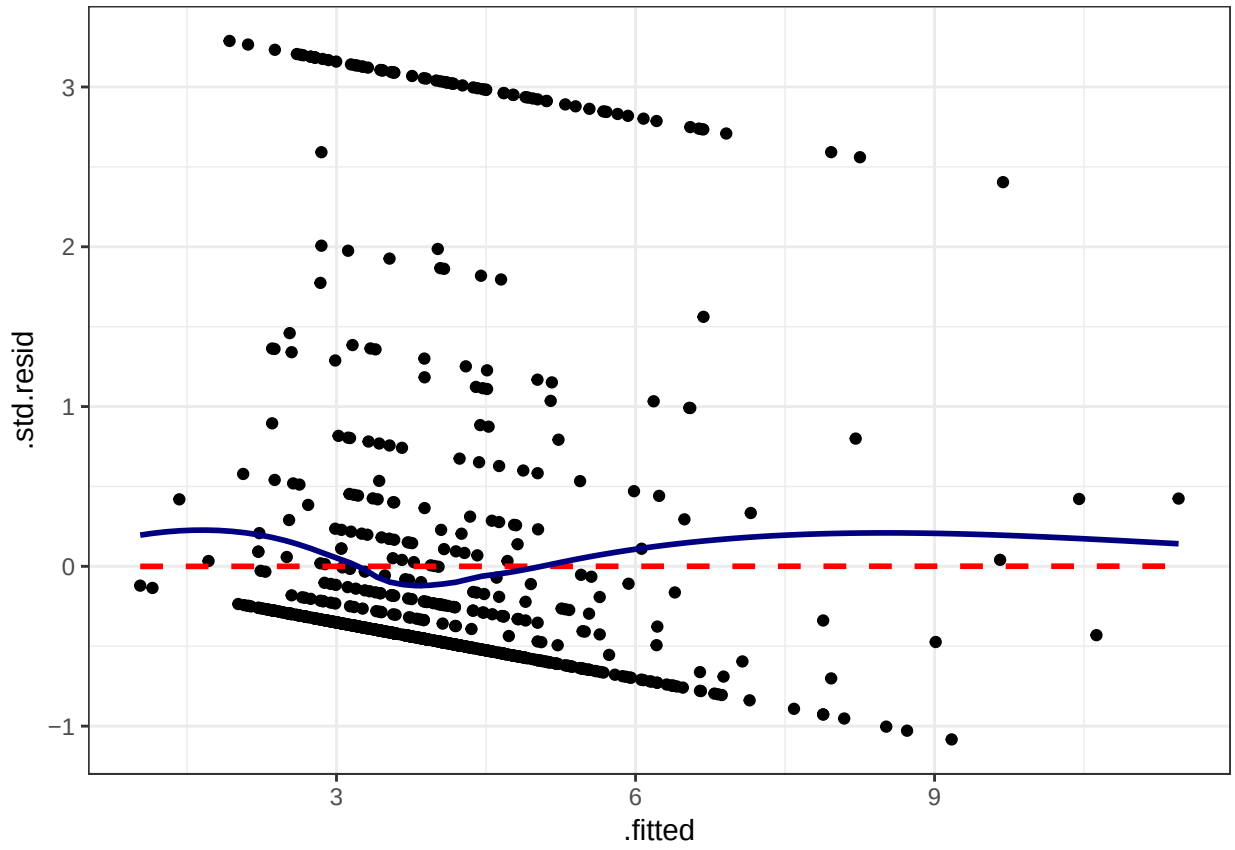
```
plot(model_A, which = 1)
```



This is a plot of residuals vs. fitted values. The goal here is for this plot to look like a random scatter of points, perhaps like a "fuzzy football", and that's **not** what we have. Why?

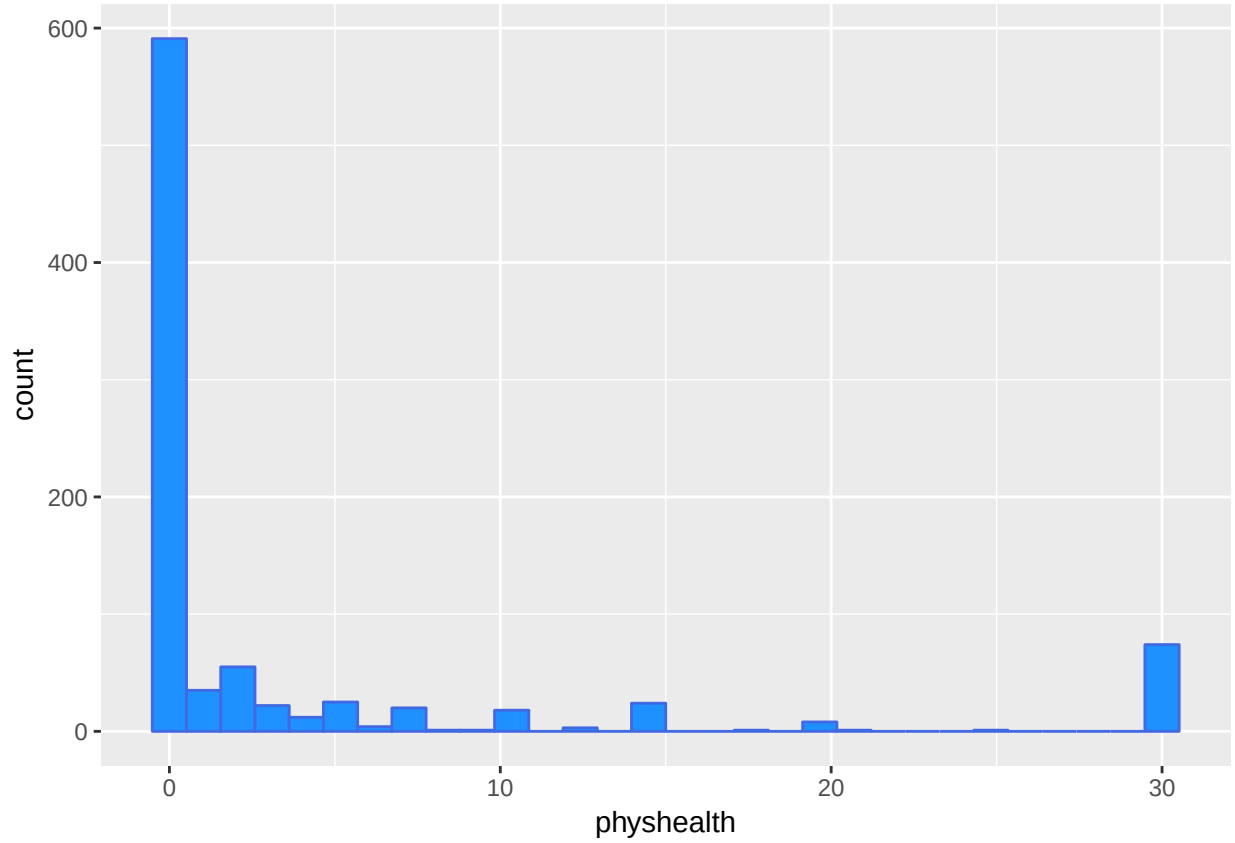
If you prefer, here's a `ggplot2` version of a similar plot, now looking at standardized residuals instead of raw residuals, and adding a loess smooth and a linear fit to the result.

```
ggplot(augment(model_A), aes(x = .fitted, y = .std.resid)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE, col = "red", linetype = "dashed") +
    geom_smooth(method = "loess", se = FALSE, col = "navy") +
    theme_bw()
```



The problem we're having here becomes, I think, a little more obvious if we look at what we're predicting. Does `physhealth` look like a good candidate for a linear model?

```
ggplot(smartcle2, aes(x = physhealth)) +
geom_histogram(bins = 30, fill = "dodgerblue", color = "royalblue")
```



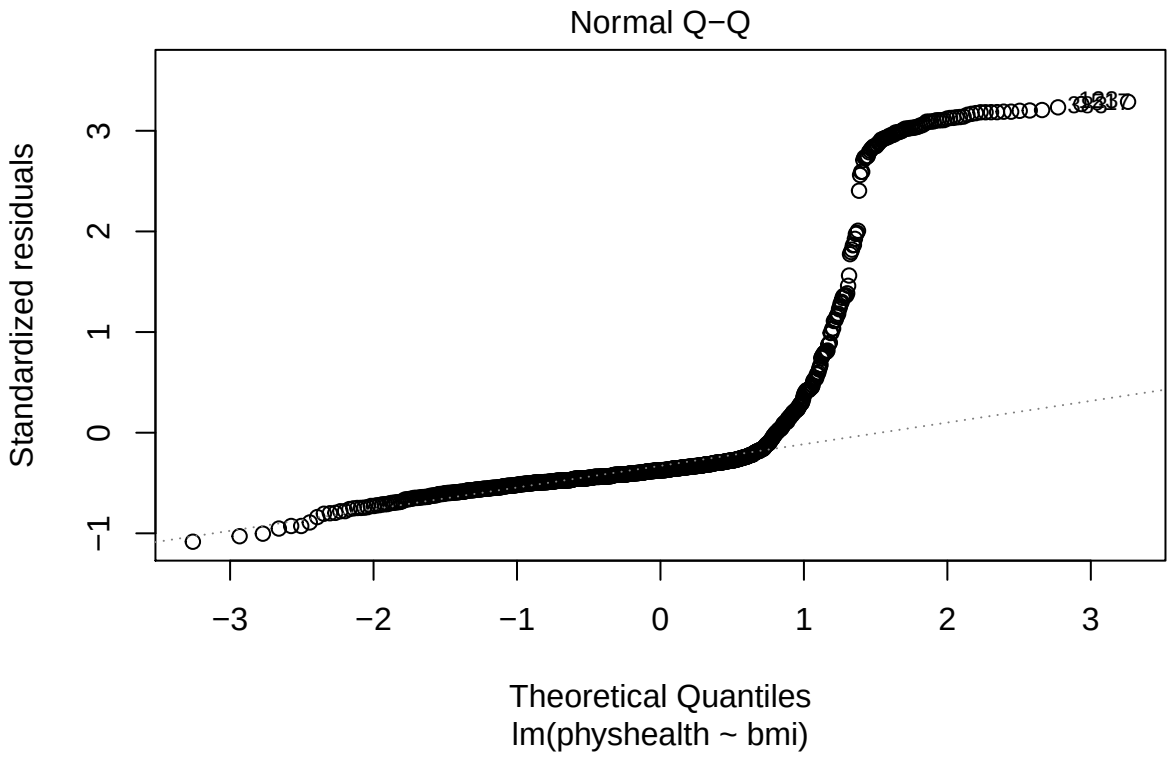

```
smartcle2 %>% count(physhealth == 0, physhealth == 30)
```

```
# A tibble: 3 x 3
  `physhealth == 0` `physhealth == 30`     n
  <lgl>             <lgl>              <int>
1 F                 F                    231
2 F                 T                     74
3 T                 F                    591
```

No matter what model we fit, if we are predicting `physhealth`, and most of the data are values of 0 and 30, we have limited variation in our outcome, and so our linear model will be somewhat questionable just on that basis.

A normal Q-Q plot of the standardized residuals for our `model_A` shows this problem, too.

```
plot(model_A, which = 2)
```



We're going to need a method to deal with this sort of outcome, that has both a floor and a ceiling. We'll get there eventually, but linear regression alone doesn't look promising.

All right, so that didn't go anywhere great. Let's try again, with a new outcome.

2.7 A New Small Study: Predicting BMI

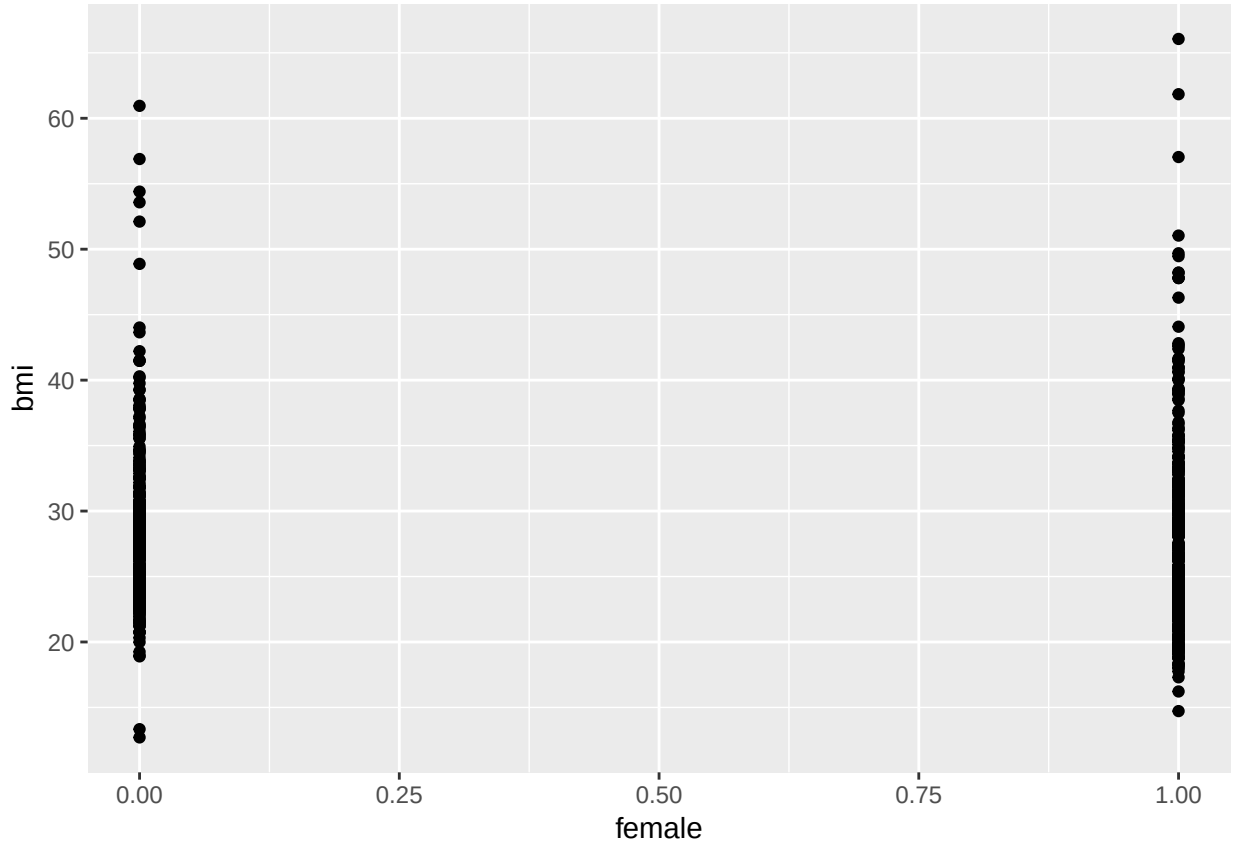
We'll begin by investigating the problem of predicting `bmi`, at first with just three regression inputs: `sex`, `exerany` and `sleephrs`, in our new `smartcle2` data set.

- The outcome of interest is `bmi`.
- Inputs to the regression model are:
  - `female` = 1 if the subject is female, and 0 if they are male
  - `exerany` = 1 if the subject exercised in the past 30 days, and 0 if they didn't
  - `sleephrs` = hours slept in a typical 24-hour period (treated as quantitative)

2.7.1 Does `female` predict `bmi` well?

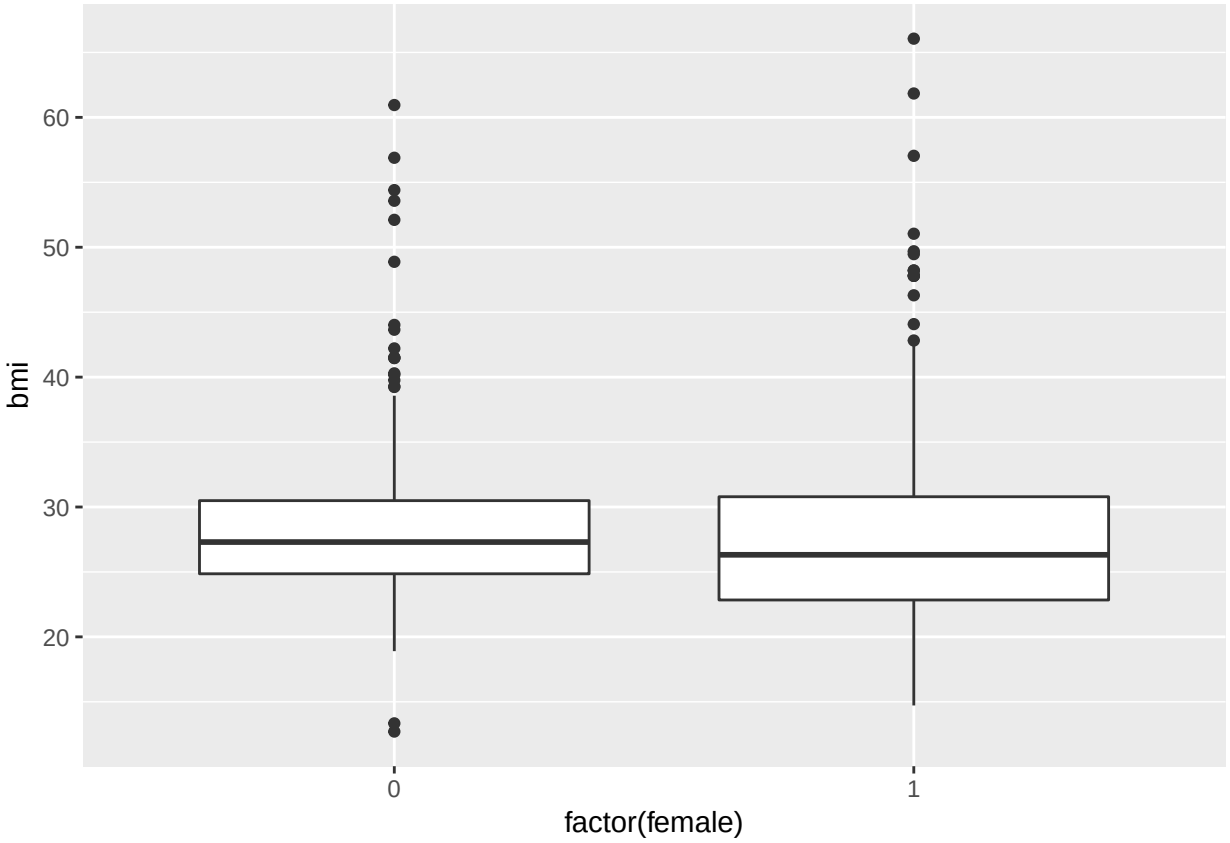
2.7.1.1 Graphical Assessment

```
ggplot(smartcle2, aes(x = female, y = bmi)) +
    geom_point()
```



Not so helpful. We should probably specify that `female` is a factor, and try another plotting approach.

```
ggplot(smartcle2, aes(x = factor(female), y = bmi)) +
    geom_boxplot()
```



The median BMI looks a little higher for males. Let's see if a model reflects that.

2.8 c2_m1: A simple t-test model

```
c2_m1 <- lm(bmi ~ female, data = smartcle2)
c2_m1
```

```

Call:
lm(formula = bmi ~ female, data = smartcle2)

Coefficients:
(Intercept)       female
    28.3600      -0.8457
```

```
summary(c2_m1)
```

```

Call:
lm(formula = bmi ~ female, data = smartcle2)

Residuals:
    Min      1Q  Median      3Q     Max
-15.650  -4.129  -1.080   2.727  38.546

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  28.3600     0.3274  86.613   <2e-16 ***
female       -0.8457     0.4282  -1.975   0.0485 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.315 on 894 degrees of freedom
Multiple R-squared:  0.004345,  Adjusted R-squared:  0.003231
F-statistic: 3.902 on 1 and 894 DF,  p-value: 0.04855
```

```
confint(c2_m1)
```

```
                2.5 %      97.5 %
(Intercept) 27.717372 29.00262801
female      -1.686052 -0.00539878
```

The model suggests, based on these 896 subjects, that

- our best prediction for males is BMI = 28.36 kg/m^2, and
- our best prediction for females is BMI = 28.36 - 0.85 = 27.51 kg/m^2.
- the mean difference between females and males is -0.85 kg/m^2 in BMI
- a 95% confidence (uncertainty) interval for that mean female - male difference in BMI ranges from -1.69 to -0.01
- the model accounts for 0.4% of the variation in BMI, so that knowing the respondent's sex does very little to reduce the size of the prediction errors as compared to an intercept only model that would predict the overall mean (regardless of sex) for all subjects.
- the model makes some enormous errors, with one subject being predicted to have a BMI 38 points lower than his/her actual BMI.

Note that this simple regression model just gives us the t-test.

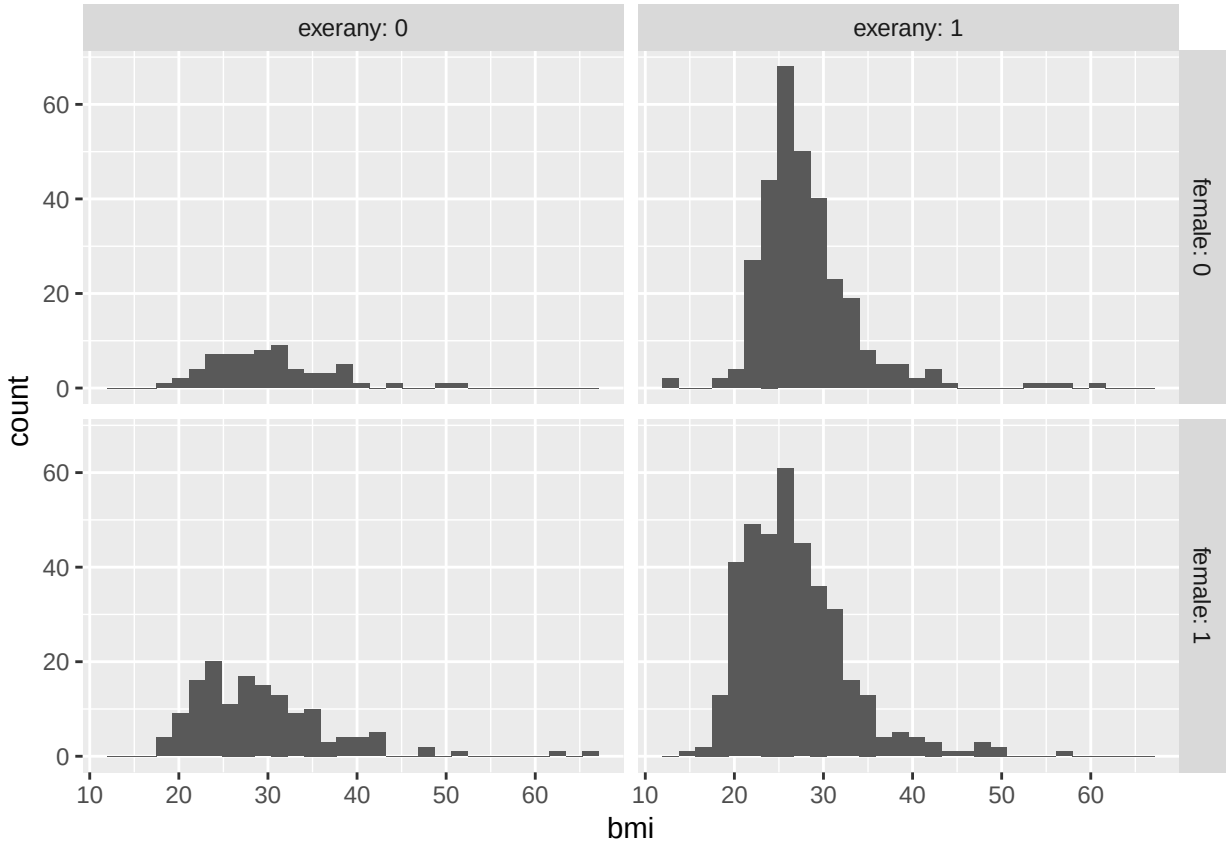
```
t.test(bmi ~ female, var.equal = TRUE, data = smartcle2)
```

```
    Two Sample t-test

data:  bmi by female
t = 1.9752, df = 894, p-value = 0.04855
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.00539878 1.68605160
sample estimates:
mean in group 0 mean in group 1
       28.36000        27.51427
```

2.9 c2_m2: Adding another predictor (two-way ANOVA without interaction)

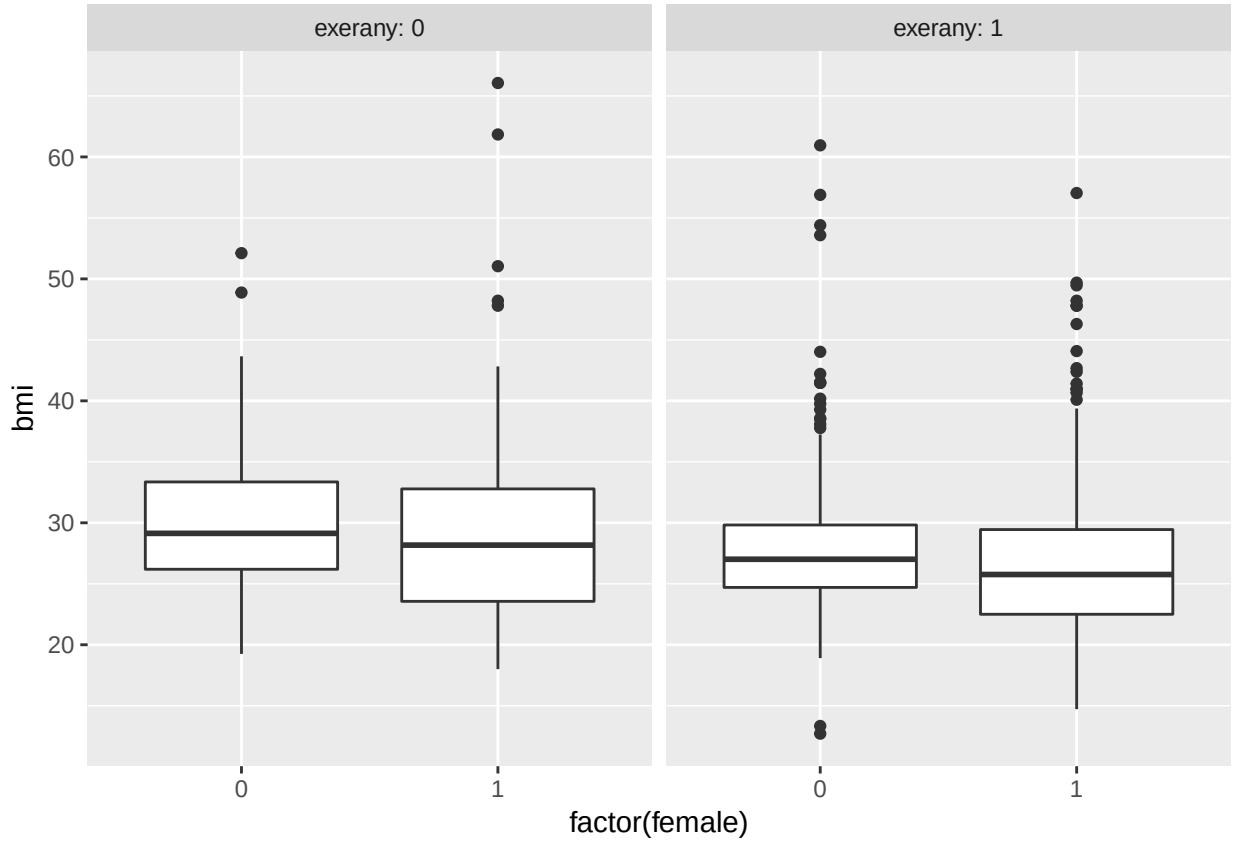
When we add in the information about `exerany` to our original model, we might first picture the data. We could look at separate histograms,

```
ggplot(smartcle2, aes(x = bmi)) +
    geom_histogram(bins = 30) +
    facet_grid(female ~ exerany, labeller = label_both)
```

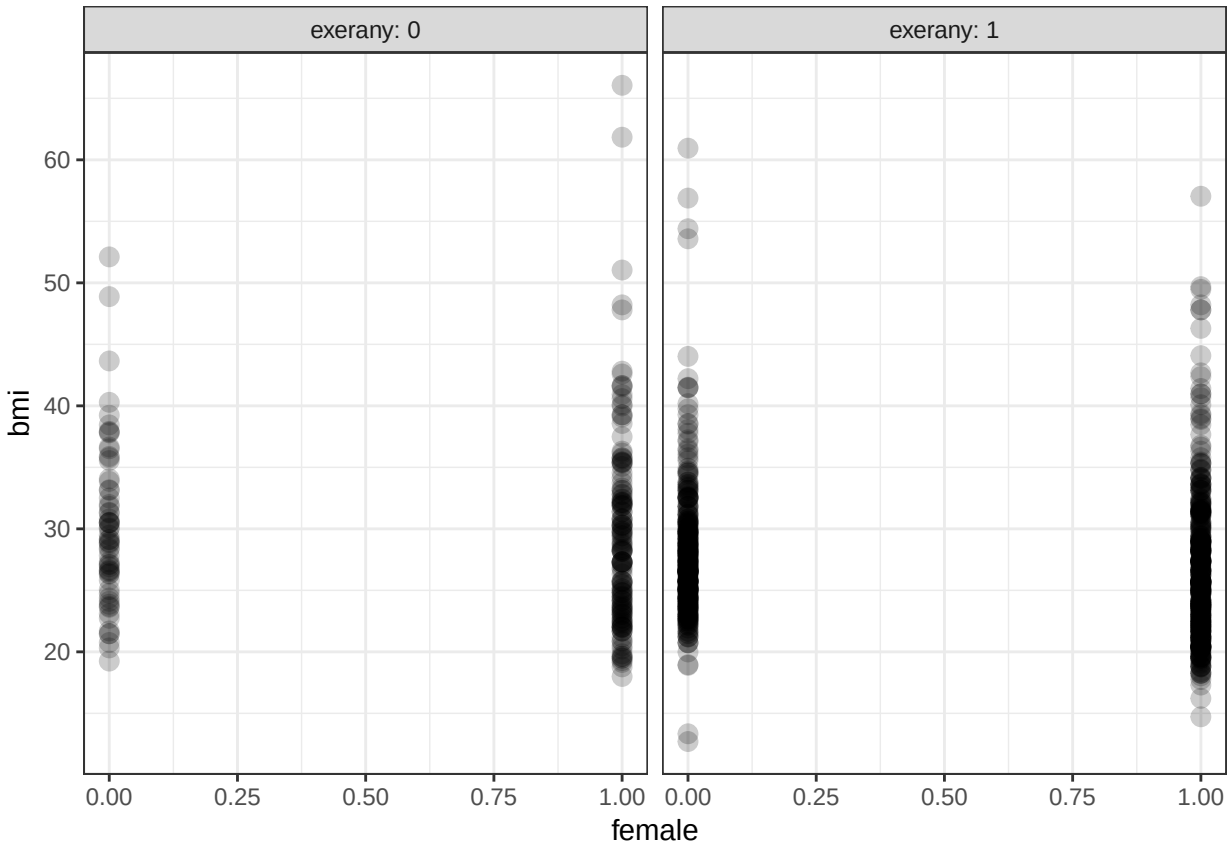


or maybe boxplots?

```
ggplot(smartcle2, aes(x = factor(female), y = bmi)) +
    geom_boxplot() +
    facet_wrap(~ exerany, labeller = label_both)
```




```
ggplot(smartcle2, aes(x = female, y = bmi))+
    geom_point(size = 3, alpha = 0.2) +
    theme_bw() +
    facet_wrap(~ exerany, labeller = label_both)
```



OK. Let's try fitting a model.

```
c2_m2 <- lm(bmi ~ female + exerany, data = smartcle2)
c2_m2
```

```
Call:
lm(formula = bmi ~ female + exerany, data = smartcle2)

Coefficients:
(Intercept)       female      exerany
     30.334       -1.095       -2.384
```

This new model predicts only four predicted values:

- `bmi` = 30.334 if the subject is male and did not exercise (so `female` = 0 and `exerany` = 0)
- `bmi` = 30.334 - 1.095 = 29.239 if the subject is female and did not exercise (`female` = 1 and `exerany` = 0)
- `bmi` = 30.334 - 2.384 = 27.950 if the subject is male and exercised (so `female` = 0 and `exerany` = 1), and, finally
- `bmi` = 30.334 - 1.095 - 2.384 = 26.855 if the subject is female and exercised (so both `female` and `exerany` = 1).

For those who did not exercise, the model is:

- `bmi` = 30.334 - 1.095 `female`

and for those who did exercise, the model is:

- `bmi` = 27.95 - 1.095 `female`

Only the intercept of the `bmi-female` model changes depending on `exerany`.

```
summary(c2_m2)
```

```
Call:
lm(formula = bmi ~ female + exerany, data = smartcle2)

Residuals:
    Min      1Q  Median      3Q     Max
-15.240  -4.091  -1.095   2.602  36.822

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  30.3335     0.5231   57.99  < 2e-16 ***
female       -1.0952     0.4262   -2.57   0.0103 *
exerany      -2.3836     0.4965   -4.80 1.86e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.239 on 893 degrees of freedom
Multiple R-squared:  0.02939,   Adjusted R-squared:  0.02722
F-statistic: 13.52 on 2 and 893 DF,  p-value: 1.641e-06
```

```
confint(c2_m2)
```

```
                2.5 %     97.5 %
(Intercept) 29.306846 31.3602182
female      -1.931629 -0.2588299
exerany     -3.358156 -1.4090777
```

The slopes of both `female` and `exerany` have confidence intervals that are completely below zero, indicating that both `female` sex and `exerany` appear to be associated with reductions in `bmi`.

The R^2 value suggests that just under 3% of the variation in `bmi` is accounted for by this ANOVA model.

In fact, this regression (on two binary indicator variables) is simply a two-way ANOVA model without an interaction term.

```
anova(c2_m2)
```

```
Analysis of Variance Table

Response: bmi
           Df Sum Sq Mean Sq F value    Pr(>F)
female      1    156  155.61  3.9977   0.04586 *
exerany     1    897  896.93 23.0435 1.856e-06 ***
Residuals 893  34759   38.92
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.10 c2_m3: Adding the interaction term (Two-way ANOVA with interaction)

Suppose we want to let the effect of `female` vary depending on the `exerany` status. Then we need to incorporate an interaction term in our model.

```
c2_m3 <- lm(bmi ~ female * exerany, data = smartcle2)
c2_m3
```

```

Call:
lm(formula = bmi ~ female * exerany, data = smartcle2)

Coefficients:
   (Intercept)          female         exerany  female:exerany
       30.1359         -0.8104         -2.1450         -0.3592
```

So, for example, for a male who exercises, this model predicts

- `bmi` = 30.136 - 0.810 (0) - 2.145 (1) - 0.359 (0)(1) = 30.136 - 2.145 = 27.991

And for a female who exercises, the model predicts

- `bmi` = 30.136 - 0.810 (1) - 2.145 (1) - 0.359 (1)(1) = 30.136 - 0.810 - 2.145 - 0.359 = 26.822

For those who did not exercise, the model is:

- `bmi` = 30.136 - 0.81 `female`

But for those who did exercise, the model is:

- `bmi` = (30.136 - 2.145) + (-0.810 + (-0.359)) `female`, or „,
- `bmi` = 27.991 - 1.169 `female`

Now, both the slope and the intercept of the `bmi-female` model change depending on `exerany`.

```
summary(c2_m3)
```

```

Call:
lm(formula = bmi ~ female * exerany, data = smartcle2)

Residuals:
    Min      1Q  Median      3Q     Max
-15.281  -4.101  -1.061   2.566  36.734

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     30.1359     0.7802  38.624   <2e-16 ***
female          -0.8104     0.9367  -0.865   0.3872
exerany         -2.1450     0.8575  -2.501   0.0125 *
female:exerany  -0.3592     1.0520  -0.341   0.7328
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.242 on 892 degrees of freedom
Multiple R-squared:  0.02952,   Adjusted R-squared:  0.02625
F-statistic: 9.044 on 3 and 892 DF,  p-value: 6.669e-06
```

```
confint(c2_m3)
```

```
                    2.5 %     97.5 %
(Intercept)    28.604610 31.6672650
female         -2.648893  1.0280526
exerany        -3.827886 -0.4620407
female:exerany -2.423994  1.7055248
```

In fact, this regression (on two binary indicator variables and a product term) is simply a two-way ANOVA model with an interaction term.

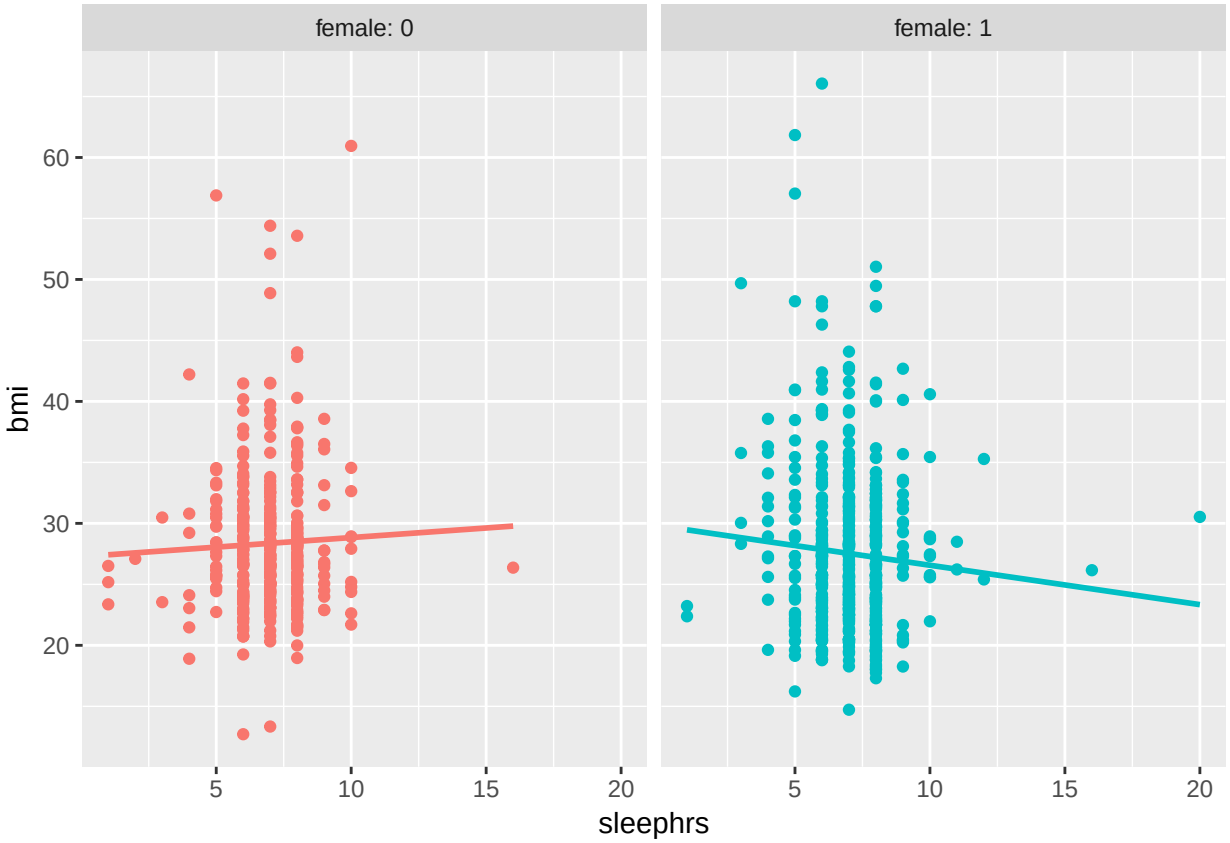
```
anova(c2_m3)
```

```
Analysis of Variance Table

Response: bmi
                Df Sum Sq Mean Sq F value    Pr(>F)
female           1    156  155.61  3.9938   0.04597 *
exerany          1    897  896.93 23.0207 1.878e-06 ***
female:exerany   1      5    4.54  0.1166   0.73283
Residuals      892  34754   38.96
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interaction term doesn't change very much here. Its uncertainty interval includes zero, and the overall model still accounts for just under 3% of the variation in `bmi`.

2.11 c2_m4: Using female and sleephrs in a model for bmi

```
ggplot(smartcle2, aes(x = sleephrs, y = bmi, color = factor(female))) +
    geom_point() +
    guides(col = FALSE) +
    geom_smooth(method = "lm", se = FALSE) +
    facet_wrap(~ female, labeller = label_both)
```



Does the difference in slopes of bmi and sleephrs for males and females appear to be substantial and important?

```
c2_m4 <- lm(bmi ~ female * sleephrs, data = smartcle2)

summary(c2_m4)
```

```

Call:
lm(formula = bmi ~ female * sleephrs, data = smartcle2)

Residuals:
    Min      1Q  Median      3Q     Max
-15.498  -4.179  -1.035   2.830  38.204

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     27.2661     1.6320  16.707   <2e-16 ***
female           2.5263     2.0975   1.204    0.229
sleephrs         0.1569     0.2294   0.684    0.494
female:sleephrs -0.4797     0.2931  -1.636    0.102
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.31 on 892 degrees of freedom
Multiple R-squared:  0.008341,  Adjusted R-squared:  0.005006
F-statistic: 2.501 on 3 and 892 DF,  p-value: 0.05818
```

Does it seem as though the addition of sleephrs has improved our model substantially over a model with female alone (which, you recall, was c2_m1)?

Since the c2_m4 model contains the c2_m1 model's predictors as a subset and the outcome is the same for each model, we consider the models *nested* and have some extra tools available to compare them.

- I might start by looking at the basic summaries for each model.

```
glance(c2_m4)
```

```
   r.squared adj.r.squared    sigma statistic    p.value df    logLik
1 0.008341404   0.005006229 6.309685   2.50104 0.05818038  4 -2919.873
       AIC      BIC deviance df.residual
1 5849.747 5873.736 35512.42         892
```

```
glance(c2_m1)
```

```
   r.squared adj.r.squared   sigma statistic    p.value df    logLik
1 0.004345169   0.003231461 6.31531  3.901534 0.04854928  2 -2921.675
      AIC      BIC deviance df.residual
1 5849.35 5863.744 35655.53         894
```

- The R^2 is twice as large for the model with sleephrs, but still very tiny.
- The p value for the global ANOVA test is actually less significant in c2_m4 than in c2_m1.
- Smaller AIC and smaller BIC statistics are more desirable. Here, there's little to choose from, but c2_m1 is a little better on each standard.
- We might also consider a significance test by looking at an ANOVA model comparison. This is only appropriate because c2_m1 is nested in c2_m4.

```
anova(c2_m4, c2_m1)
```

```
Analysis of Variance Table
```

```
Model 1: bmi ~ female * sleephrs
Model 2: bmi ~ female
  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1    892 35512
2    894 35656 -2   -143.11 1.7973 0.1663
```

The addition of the `sleephrs` term picked up 143 in the sum of squares column, at a cost of two degrees of freedom, yielding a p value of 0.166, suggesting that this isn't a significant improvement over the model that just did a t-test on `female`.

2.12 Making Predictions with a Linear Regression Model

Recall model 4, which yields predictions for body mass index on the basis of the main effects of sex (`female`) and hours of sleep (`sleephrs`) and their interaction.

```
c2_m4
```

```
Call:
lm(formula = bmi ~ female * sleephrs, data = smartcle2)

Coefficients:
    (Intercept)           female         sleephrs  female:sleephrs
        27.2661           2.5263           0.1569          -0.4797
```

2.12.1 Fitting an Individual Prediction and 95% Prediction Interval

What do we predict for the `bmi` of a subject who is `female` and gets 8 hours of sleep per night?

```
c2_new1 <- data_frame(female = 1, sleephrs = 8)
predict(c2_m4, newdata = c2_new1, interval = "prediction", level = 0.95)
```

```
       fit     lwr     upr
1 27.21065 14.8107 39.6106
```

The predicted `bmi` for this new subject is 27.61. The prediction interval shows the bounds of a 95% uncertainty interval for a predicted `bmi` for an individual female subject who gets 8 hours of sleep on average per evening. From the `predict` function applied to a linear model, we can get the prediction intervals for any new data points in this manner.

2.12.2 Confidence Interval for an Average Prediction

- What do we predict for the **average body mass index of a population of subjects** who are female and sleep for 8 hours?

```
predict(c2_m4, newdata = c2_new1, interval = "confidence", level = 0.95)
```

```
       fit      lwr      upr
1 27.21065 26.57328 27.84801
```

- How does this result compare to the prediction interval?

2.12.3 Fitting Multiple Individual Predictions to New Data

- How does our prediction change for a respondent if they instead get 7, or 9 hours of sleep? What if they are male, instead of female?

```
c2_new2 <- data_frame(subjectid = 1001:1006, female = c(1, 1, 1, 0, 0, 0), sleephrs = c(7, 8, 9, 7, 8, 9
pred2 <- predict(c2_m4, newdata = c2_new2, interval = "prediction", level = 0.95) %>% tbl_df

result2 <- bind_cols(c2_new2, pred2)
result2
```

```
# A tibble: 6 x 6
  subjectid female sleephrs   fit   lwr   upr
      <int>  <dbl>    <dbl> <dbl> <dbl> <dbl>
1      1001   1.00     7.00  27.5  15.1  39.9
2      1002   1.00     8.00  27.2  14.8  39.6
3      1003   1.00     9.00  26.9  14.5  39.3
4      1004   0        7.00  28.4  16.0  40.8
5      1005   0        8.00  28.5  16.1  40.9
6      1006   0        9.00  28.7  16.2  41.1
```

The `result2` tibble contains predictions for each scenario.

- Which has a bigger impact on these predictions and prediction intervals? A one category change in `female` or a one hour change in `sleephrs`?

2.12.4 Simulation to represent predictive uncertainty in Model 4

Suppose we want to predict the `bmi` of a female subject who sleeps for eight hours per night. As we have seen, we can do this automatically for a linear model like this one, using the `predict` function applied to the linear model, but a simulation prediction can also be done. Recall the detail of `c2_m4`:

```
c2_m4
```

```
Call:
lm(formula = bmi ~ female * sleephrs, data = smartcle2)

Coefficients:
    (Intercept)           female         sleephrs  female:sleephrs
        27.2661           2.5263           0.1569          -0.4797
```

```
glance(c2_m4)
```

```
    r.squared adj.r.squared    sigma statistic    p.value df    logLik
1 0.008341404   0.005006229 6.309685   2.50104 0.05818038  4 -2919.873
       AIC      BIC deviance df.residual
1 5849.747 5873.736 35512.42         892
```

We see that the residual standard error for our `bmi` predictions with this model is 6.31.

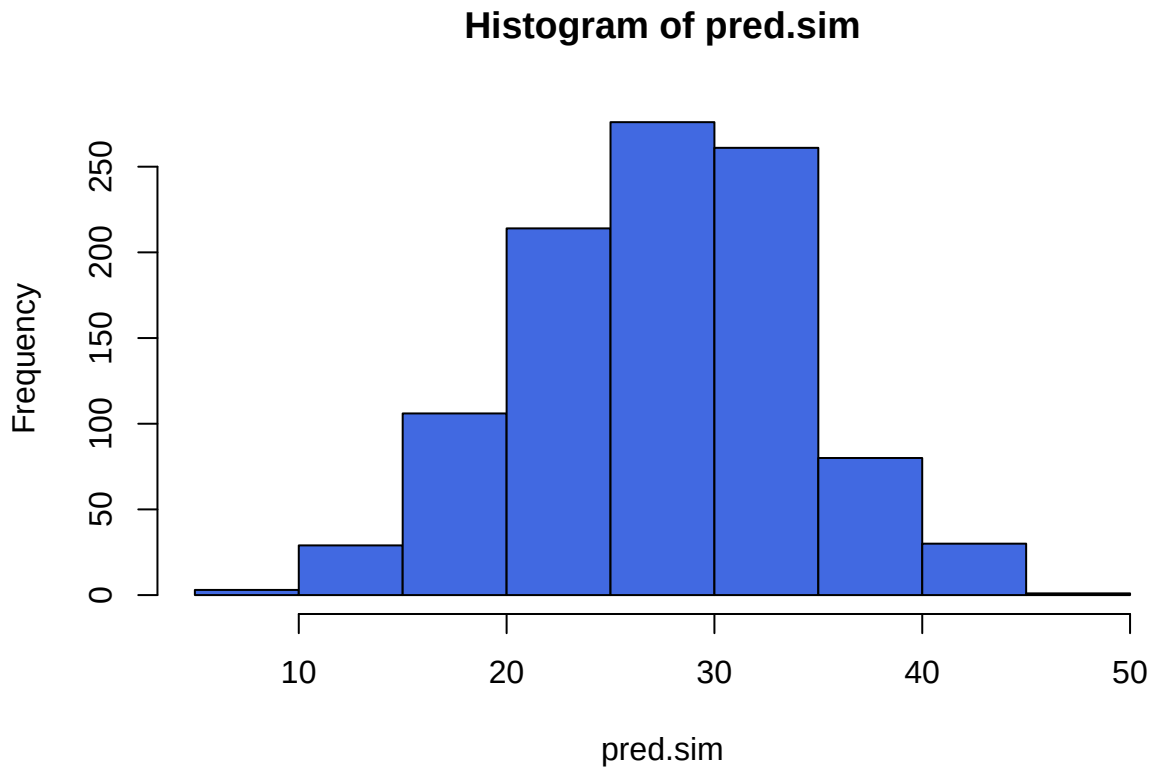
For a female respondent sleeping eight hours, recall that our point estimate (predicted value) of `bmi` is 27.21

```
predict(c2_m4, newdata = c2_new1, interval = "prediction", level = 0.95)
```

```
       fit     lwr     upr
1 27.21065 14.8107 39.6106
```

The standard deviation is 6.31, so we could summarize the predictive distribution with a command that tells R to draw 1000 random numbers from a normal distribution with mean 27.21 and standard deviation 6.31. Let's summarize that and get a quick picture.

```
set.seed(432094)
pred.sim <- rnorm(1000, 27.21, 6.31)
hist(pred.sim, col = "royalblue")
```




```
mean(pred.sim)
```

```
[1] 27.41856
```

```
quantile(pred.sim, c(0.025, 0.975))
```

```
    2.5%    97.5%
14.48487 40.16778
```

How do these results compare to the prediction interval of (14.81, 39.61) that we generated earlier?

2.13 Centering the model

Our model `c2_m4` has four predictors (the constant, `sleephrs`, `female` and their interaction) but just two inputs (`female` and `sleephrs`.) If we **center** the quantitative input `sleephrs` before building the model, we get a more interpretable interaction term.

```
smartcle2_c <- smartcle2 %>%
    mutate(sleephrs_c = sleephrs - mean(sleephrs))
```

```
c2_m4_c <- lm(bmi ~ female * sleephrs_c, data = smartcle2_c)

summary(c2_m4_c)
```

```
Call:
lm(formula = bmi ~ female * sleephrs_c, data = smartcle2_c)

Residuals:
    Min      1Q  Median      3Q     Max
-15.498  -4.179  -1.035   2.830  38.204

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       28.3681     0.3274  86.658   <2e-16 ***
female            -0.8420     0.4280  -1.967   0.0495 *
sleephrs_c         0.1569     0.2294   0.684   0.4940
female:sleephrs_c -0.4797     0.2931  -1.636   0.1021
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.31 on 892 degrees of freedom
Multiple R-squared:  0.008341,  Adjusted R-squared:  0.005006
F-statistic: 2.501 on 3 and 892 DF,  p-value: 0.05818
```

What has changed as compared to the original `c2_m4`?

- Our original model was `bmi` = 27.26 + 2.53 `female` + 0.16 `sleephrs` - 0.48 `female` x `sleephrs`
- Our new model is `bmi` = 28.37 - 0.84 `female` + 0.16 centered `sleephrs` - 0.48 `female` x centered `sleephrs`.

So our new model on centered data is:

- 28.37 + 0.16 centered `sleephrs_c` for male subjects, and
- (28.37 - 0.84) + (0.16 - 0.48) centered `sleephrs_c`, or 27.53 - 0.32 centered `sleephrs_c` for female subjects.

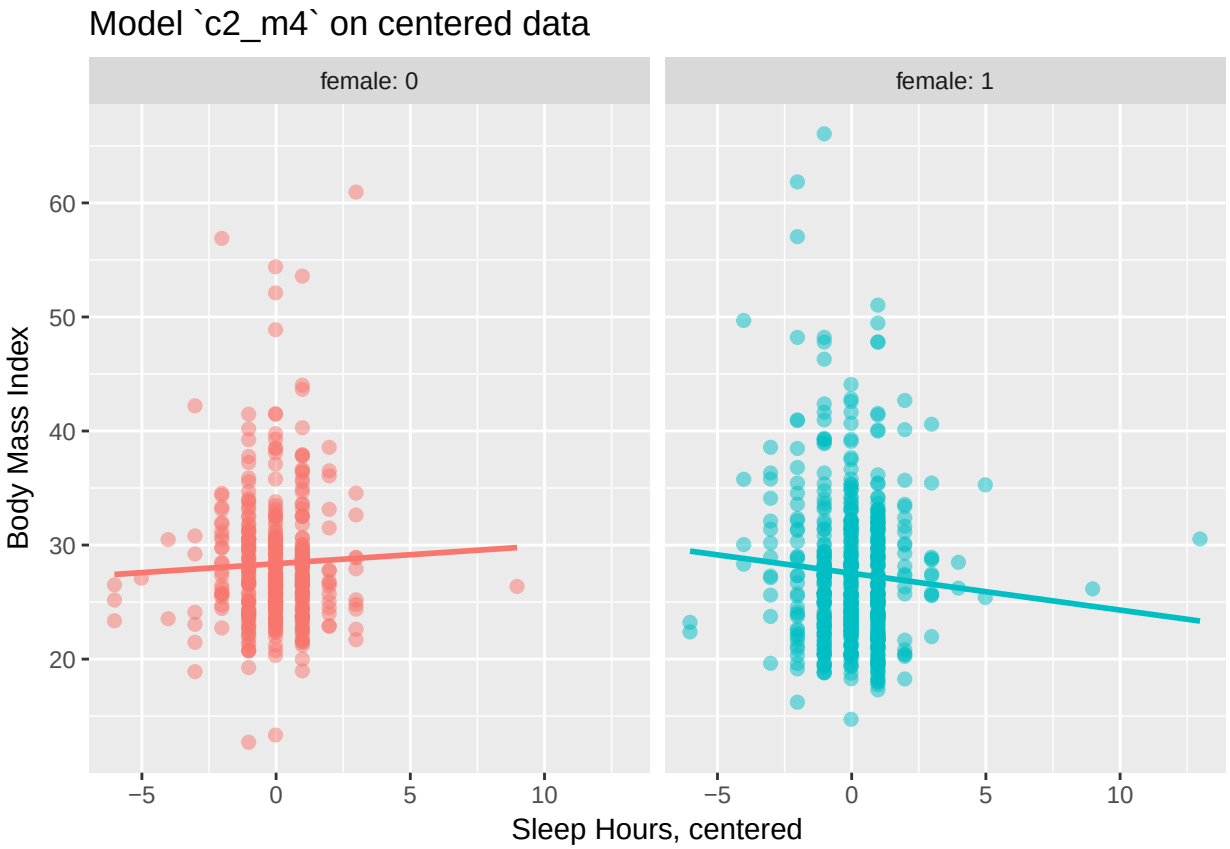
In our new (centered `sleephrs_c`) model,

- the main effect of `female` now corresponds to a predictive difference (female - male) in `bmi` with `sleephrs` at its mean value, 7.02 hours,
- the intercept term is now the predicted `bmi` for a male respondent who sleeps an average number of hours, and
- the product term corresponds to the change in the slope of centered `sleephrs_c` on `bmi` for a female rather than a male subject, while
- the residual standard deviation and the R-squared values remain unchanged from the model before centering.

2.13.1 Plot of Model 4 on Centered `sleephrs`: `c2_m4_c`

```
ggplot(smartcle2_c, aes(x = sleephrs_c, y = bmi, group = female, col = factor(female))) +
    geom_point(alpha = 0.5, size = 2) +
    geom_smooth(method = "lm", se = FALSE) +
    guides(color = FALSE) +
    labs(x = "Sleep Hours, centered", y = "Body Mass Index",
```

```
        title = "Model `c2_m4` on centered data") +
    facet_wrap(~ female, labeller = label_both)
```



2.14 Rescaling an input by subtracting the mean and dividing by 2 standard deviations

Centering helped us interpret the main effects in the regression, but it still leaves a scaling problem.

- The `female` coefficient estimate is much larger than that of `sleephrs`, but this is misleading, considering that we are comparing the complete change in one variable (sex = female or not) to a 1-hour change in average sleep.
- Gelman and Hill (2007) recommend all continuous predictors be scaled by dividing by 2 standard deviations, so that:
  - a 1-unit change in the rescaled predictor corresponds to a change from 1 standard deviation below the mean, to 1 standard deviation above.
  - an unscaled binary (1/0) predictor with 50% probability of occurring will be exactly comparable to a rescaled continuous predictor done in this way.

```
smartcle2_rescale <- smartcle2 %>%
    mutate(sleephrs_z = (sleephrs - mean(sleephrs))/(2*sd(sleephrs)))
```

2.14.1 Refitting model c2_m4 to the rescaled data

```
c2_m4_z <- lm(bmi ~ female * sleephrs_z, data = smartcle2_rescale)

summary(c2_m4_z)
```

```
Call:
lm(formula = bmi ~ female * sleephrs_z, data = smartcle2_rescale)

Residuals:
    Min      1Q  Median      3Q     Max
-15.498  -4.179  -1.035   2.830  38.204

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        28.3681     0.3274  86.658   <2e-16 ***
female             -0.8420     0.4280  -1.967   0.0495 *
sleephrs_z          0.4637     0.6778   0.684   0.4940
female:sleephrs_z  -1.4173     0.8661  -1.636   0.1021
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.31 on 892 degrees of freedom
Multiple R-squared:  0.008341,  Adjusted R-squared:  0.005006
F-statistic: 2.501 on 3 and 892 DF,  p-value: 0.05818
```

2.14.2 Interpreting the model on rescaled data

What has changed as compared to the original c2_m4?

- Our original model was bmi = 27.26 + 2.53 female + 0.16 sleephrs - 0.48 female x sleephrs
- Our model on centered sleephrs was bmi = 28.37 - 0.84 female + 0.16 centered sleephrs_c - 0.48 female x centered sleephrs_c.
- Our new model on rescaled sleephrs is bmi = 28.37 - 0.84 female + 0.46 rescaled sleephrs_z - 1.42 female x rescaled sleephrs_z.

So our rescaled model is:

- 28.37 + 0.46 rescaled sleephrs_z for male subjects, and
- (28.37 - 0.84) + (0.46 - 1.42) rescaled sleephrs_z, or 27.53 - 0.96 rescaled sleephrs_z for female subjects.

In this new rescaled (sleephrs_z) model, then,

- the main effect of female, -0.84, still corresponds to a predictive difference (female - male) in bmi with sleephrs at its mean value, 7.02 hours,
- the intercept term is still the predicted bmi for a male respondent who sleeps an average number of hours, and
- the residual standard deviation and the R-squared values remain unchanged,

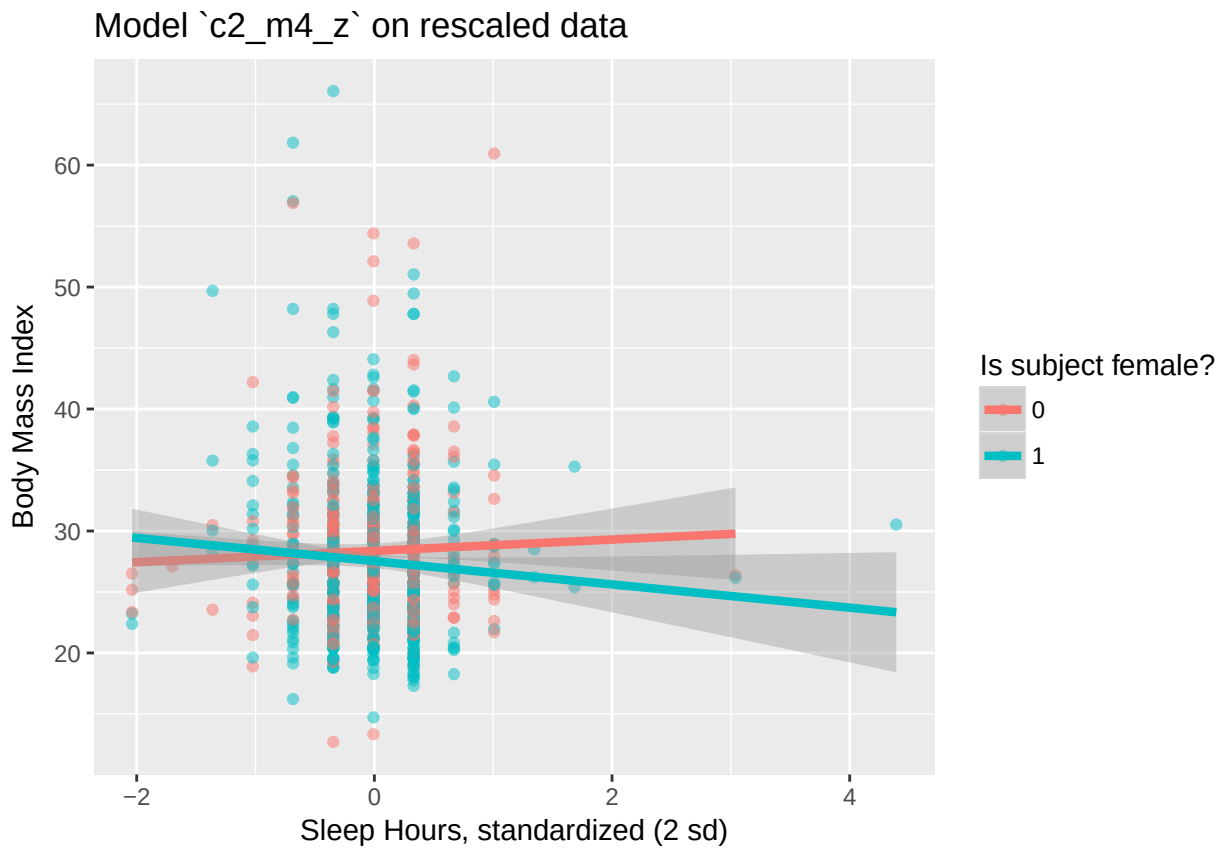
as before, but now we also have that:

- the coefficient of sleephrs_z indicates the predictive difference in bmi associated with a change in sleephrs of 2 standard deviations (from one standard deviation below the mean of 7.02 to one standard deviation above 7.02.)

  - Since the standard deviation of `sleephrs` is 1.48, this corresponds to a change from 5.54 hours per night to 8.50 hours per night.
- the coefficient of the product term (-1.42) corresponds to the change in the coefficient of `sleephrs_z` for females as compared to males.

2.14.3 Plot of model on rescaled data

```
ggplot(smartcle2_rescale, aes(x = sleephrs_z, y = bmi,
                              group = female, col = factor(female))) +
    geom_point(alpha = 0.5) +
    geom_smooth(method = "lm", size = 1.5) +
    scale_color_discrete(name = "Is subject female?") +
    labs(x = "Sleep Hours, standardized (2 sd)", y = "Body Mass Index",
         title = "Model `c2_m4_z` on rescaled data")
```



2.15 c2_m5: What if we add more variables?

We can boost our R^2 a bit, to over 5%, by adding in two new variables, related to whether or not the subject (in the past 30 days) used the internet, and on how many days the subject drank alcoholic beverages.

```
c2_m5 <- lm(bmi ~ female + exerany + sleephrs + internet30 + alcdays,
         data = smartcle2)
summary(c2_m5)
```

```
Call:
lm(formula = bmi ~ female + exerany + sleephrs + internet30 +
    alcdays, data = smartcle2)

Residuals:
    Min      1Q  Median      3Q     Max
-16.147  -3.997  -0.856   2.487  35.965

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.84066    1.18458  26.035  < 2e-16 ***
female      -1.28801    0.42805  -3.009   0.0027 **
exerany     -2.42161    0.49853  -4.858 1.40e-06 ***
sleephrs    -0.14118    0.13988  -1.009   0.3131
internet30   1.38916    0.54252   2.561   0.0106 *
alcdays     -0.10460    0.02595  -4.030 6.04e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.174 on 890 degrees of freedom
Multiple R-squared:  0.05258,   Adjusted R-squared:  0.04726
F-statistic: 9.879 on 5 and 890 DF,  p-value: 3.304e-09
```

1. Here's the ANOVA for this model. What can we study with this?

```
anova(c2_m5)
```

```
Analysis of Variance Table

Response: bmi
            Df Sum Sq Mean Sq F value    Pr(>F)
female       1    156  155.61  4.0818   0.04365 *
exerany      1    897  896.93 23.5283 1.453e-06 ***
sleephrs     1     33   32.90  0.8631   0.35313
internet30   1    178  178.33  4.6779   0.03082 *
alcdays      1    619  619.26 16.2443 6.044e-05 ***
Residuals  890  33928   38.12
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2. Consider the revised output below. Now what can we study?

```
anova(lm(bmi ~ exerany + internet30 + alcdays + female + sleephrs,
         data = smartcle2))
```

```
Analysis of Variance Table

Response: bmi
            Df Sum Sq Mean Sq F value    Pr(>F)
exerany      1    795  795.46 20.8664 5.618e-06 ***
internet30   1    212  211.95  5.5599 0.0185925 *
alcdays      1    486  486.03 12.7496 0.0003752 ***
female       1    351  350.75  9.2010 0.0024891 **
sleephrs     1     39   38.83  1.0186 0.3131176
Residuals  890  33928   38.12
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3. What does the output below let us conclude?

```
anova(lm(bmi ~ exerany + internet30 + alcdays + female + sleephrs,
         data = smartcle2),
      lm(bmi ~ exerany + female + alcdays,
         data = smartcle2))
```

```
Analysis of Variance Table

Model 1: bmi ~ exerany + internet30 + alcdays + female + sleephrs
Model 2: bmi ~ exerany + female + alcdays
  Res.Df   RSS Df Sum of Sq      F  Pr(>F)
1    890 33928
2    892 34221 -2    -293.2 3.8456 0.02173 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4. What does it mean for the models to be "nested"?

2.16 c2_m6: Would adding self-reported health help?

And we can do even a bit better than that by adding in a multi-categorical measure: self-reported general health.

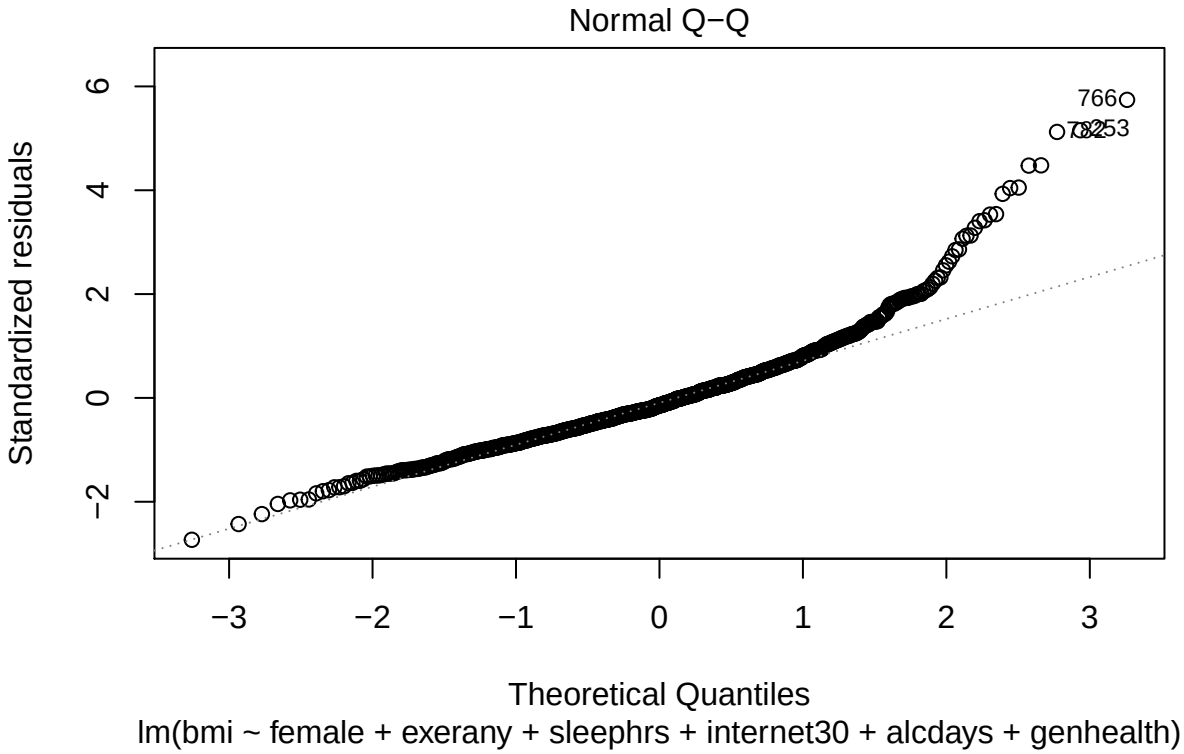
```
c2_m6 <- lm(bmi ~ female + exerany + sleephrs + internet30 + alcdays + genhealth,
         data = smartcle2)
summary(c2_m6)
```

```
Call:
lm(formula = bmi ~ female + exerany + sleephrs + internet30 +
    alcdays + genhealth, data = smartcle2)

Residuals:
    Min      1Q  Median      3Q     Max
-16.331  -3.813  -0.838   2.679  34.166

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         26.49498    1.31121  20.206  < 2e-16 ***
female              -0.85520    0.41969  -2.038 0.041879 *
exerany             -1.61968    0.50541  -3.205 0.001400 **
sleephrs            -0.12719    0.13613  -0.934 0.350368
internet30           2.02498    0.53898   3.757 0.000183 ***
alcdays             -0.08431    0.02537  -3.324 0.000925 ***
genhealth2_VeryGood  2.10537    0.59408   3.544 0.000415 ***
genhealth3_Good      4.08245    0.60739   6.721 3.22e-11 ***
genhealth4_Fair      4.99213    0.80178   6.226 7.37e-10 ***
genhealth5_Poor      3.11025    1.12614   2.762 0.005866 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.993 on 886 degrees of freedom
```

```
Multiple R-squared:  0.1115,    Adjusted R-squared:  0.1024
F-statistic: 12.35 on 9 and 886 DF,  p-value: < 2.2e-16
```

1. If Harry and Marty have the same values of `female`, `exerany`, `sleephrs`, `internet30` and `alcdays`, but Harry rates his health as Good, and Marty rates his as Fair, then what is the difference in the predictions? Who is predicted to have a larger BMI, and by how much?
2. What does this normal probability plot of the residuals suggest?

```
plot(c2_m6, which = 2)
```



2.17 c2_m7: What if we added the menthealth variable?

```
c2_m7 <- lm(bmi ~ female + exerany + sleephrs + internet30 + alcdays +
              genhealth + physhealth + menthealth,
            data = smartcle2)

summary(c2_m7)
```

```
Call:
lm(formula = bmi ~ female + exerany + sleephrs + internet30 +
    alcdays + genhealth + physhealth + menthealth, data = smartcle2)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-16.060  -3.804  -0.890   2.794  33.972

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         25.88208    1.31854  19.629  < 2e-16 ***
female              -0.96435    0.41908  -2.301 0.021616 *
exerany             -1.43171    0.50635  -2.828 0.004797 **
sleephrs            -0.08033    0.13624  -0.590 0.555583
internet30           2.00267    0.53759   3.725 0.000207 ***
alcdays             -0.07997    0.02528  -3.163 0.001614 **
genhealth2_VeryGood  2.09533    0.59238   3.537 0.000425 ***
genhealth3_Good      3.90949    0.60788   6.431 2.07e-10 ***
genhealth4_Fair      4.27152    0.83986   5.086 4.47e-07 ***
genhealth5_Poor      1.26021    1.31556   0.958 0.338361
physhealth           0.06088    0.03005   2.026 0.043064 *
menthealth           0.06636    0.03177   2.089 0.037021 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.964 on 884 degrees of freedom
Multiple R-squared:  0.1219,    Adjusted R-squared:  0.111
F-statistic: 11.16 on 11 and 884 DF,  p-value: < 2.2e-16
```

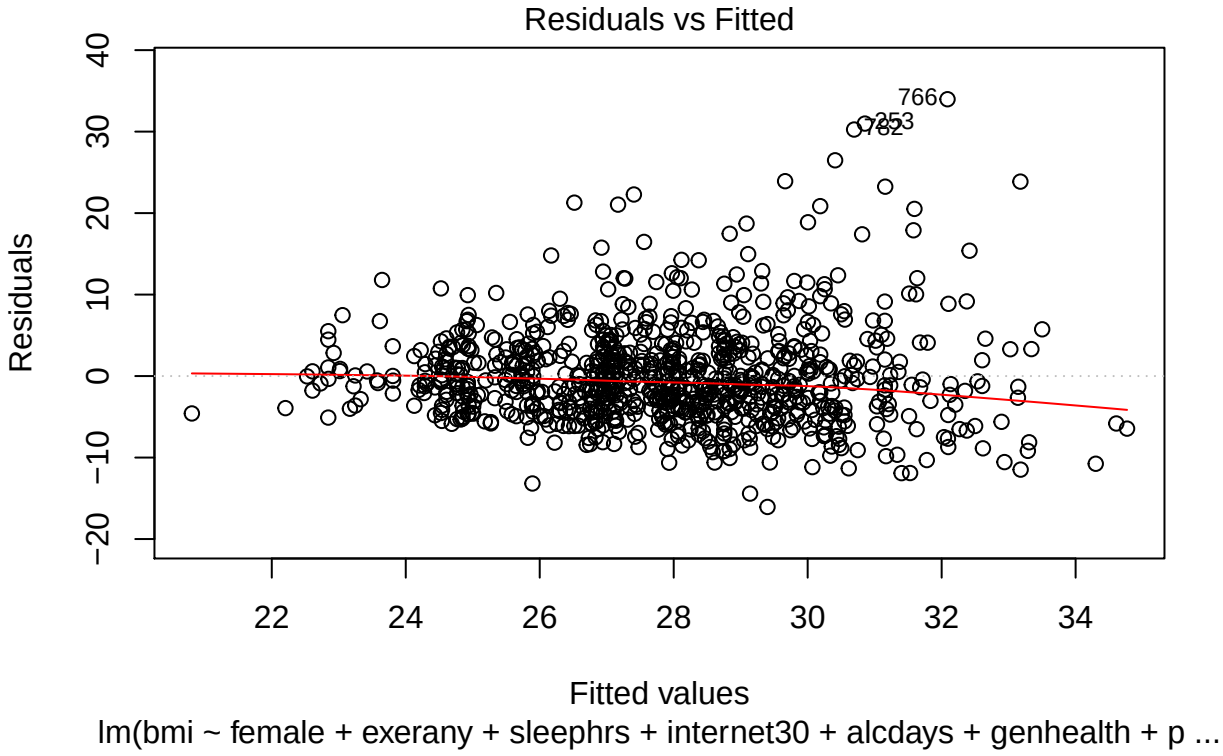
2.18 Key Regression Assumptions for Building Effective Prediction Models

1. Validity - the data you are analyzing should map to the research question you are trying to answer.
    - The outcome should accurately reflect the phenomenon of interest.
    - The model should include all relevant predictors. (It can be difficult to decide which predictors are necessary, and what to do with predictors that have large standard errors.)
    - The model should generalize to all of the cases to which it will be applied.
    - Can the available data answer our question reliably?
2. Additivity and linearity - most important assumption of a regression model is that its deterministic component is a linear function of the predictors. We often think about transformations in this setting.
3. Independence of errors - errors from the prediction line are independent of each other
4. Equal variance of errors - if this is violated, we can more efficiently estimate paramaters using *weighted least squares* approaches, where each point is weighted inversely proportional to its variance, but this doesn't affect the coefficients much, if at all.
5. Normality of errors - not generally important for estimating the regression line

2.18.1 Checking Assumptions in model `c2_m7`

1. How does the assumption of linearity behind this model look?

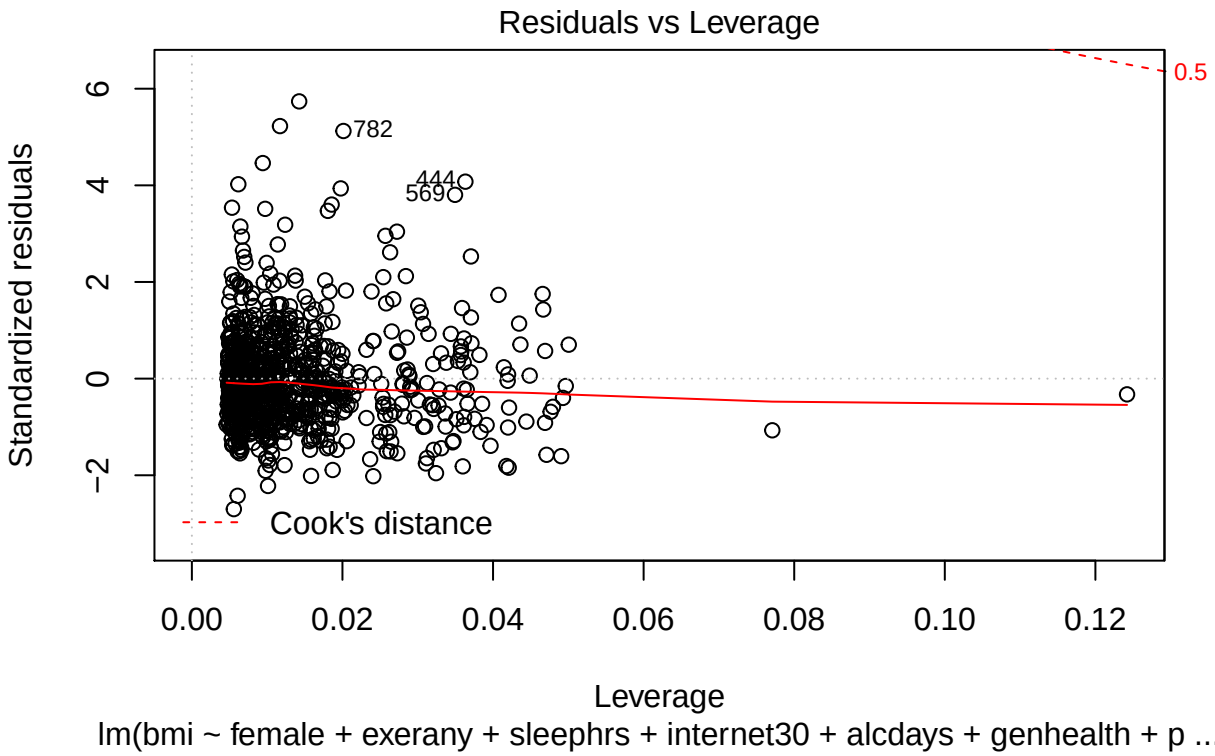
```
plot(c2_m7, which = 1)
```



We see no strong signs of serious non-linearity here. There's no obvious curve in the plot, for example.

2. What can we conclude from the plot below?

```
plot(c2_m7, which = 5)
```



This plot can help us identify points with large standardized residuals, large leverage values, and large influence on the model (as indicated by large values of Cook's distance.) In this case, I see no signs of any points used in the model with especially large influence, although there are some poorly fitted points (with especially large standardized residuals.)

Chapter 3

Analysis of Variance

3.1 The bonding data: A Designed Dental Experiment

The `bonding` data describe a designed experiment into the properties of four different resin types (`resin` = A, B, C, D) and two different curing light sources (`light` = Halogen, LED) as they relate to the resulting bonding strength (measured in MPa[1]) on the surface of teeth. The source is Kim (2014).

The experiment involved making measurements of bonding strength under a total of 80 experimental setups, or runs, with 10 runs completed at each of the eight combinations of a light source and a resin type. The data are gathered in the `bonding.csv` file.

```
bonding
```

```
# A tibble: 80 x 4
   run_ID light   resin strength
   <fct>  <fct>   <fct>    <dbl>
 1 R101   LED     B         12.8
 2 R102   Halogen B         22.2
 3 R103   Halogen B         24.6
 4 R104   LED     A         17.0
 5 R105   LED     C         32.2
 6 R106   Halogen B         27.1
 7 R107   LED     A         23.4
 8 R108   Halogen A         23.5
 9 R109   Halogen D         37.3
10 R110   Halogen A         19.7
# ... with 70 more rows
```

3.2 A One-Factor Analysis of Variance

Suppose we are interested in the distribution of the `strength` values for the four different types of `resin`.

```
bonding %>% group_by(resin) %>% summarize(n = n(), mean(strength), median(strength))
```

```
# A tibble: 4 x 4
  resin     n `mean(strength)` `median(strength)`
  <fct> <int>            <dbl>              <dbl>
1 A        20             18.4               18.0
```

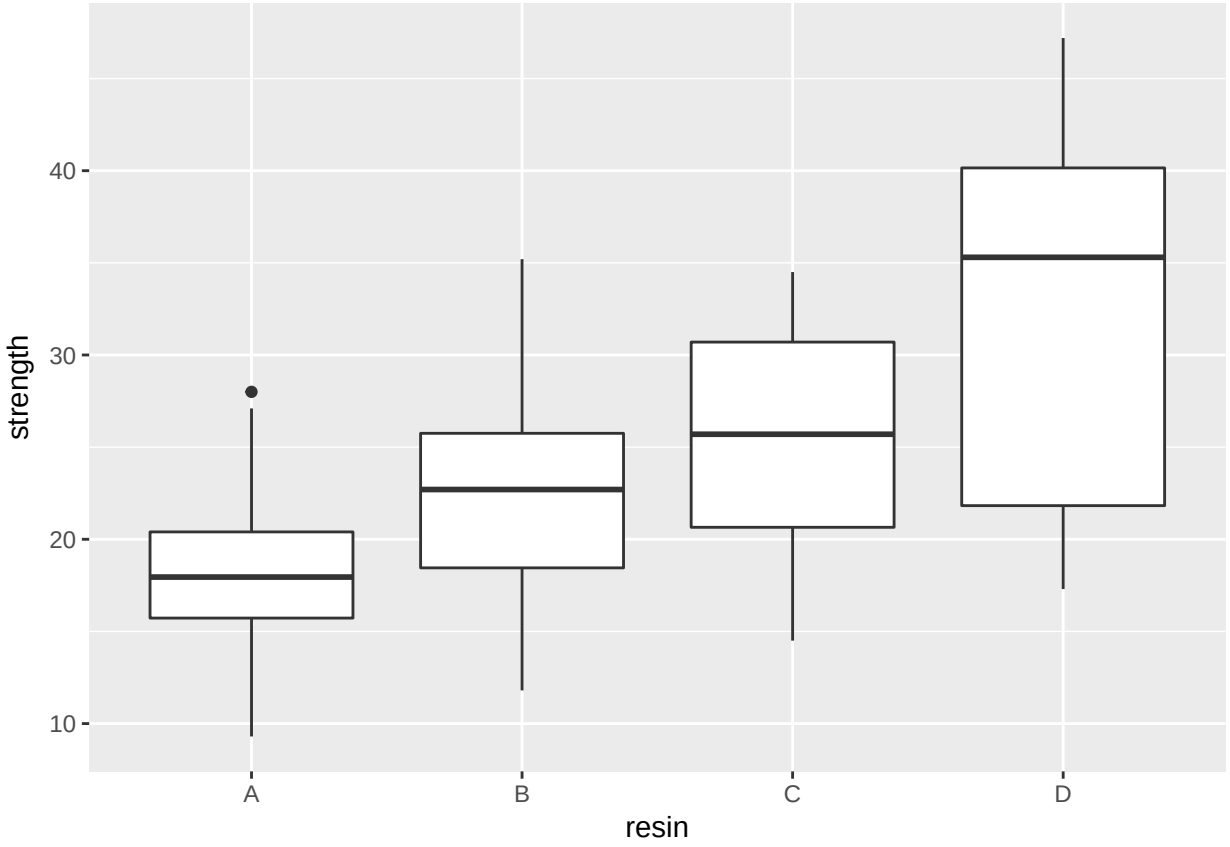
[1]The MPa is defined as the failure load (in Newtons) divided by the entire bonded area, in mm^2.

```
2 B        20              22.2              22.7
3 C        20              25.2              25.7
4 D        20              32.1              35.3
```

I'd begin serious work with a plot.

3.2.1 Look at the Data!

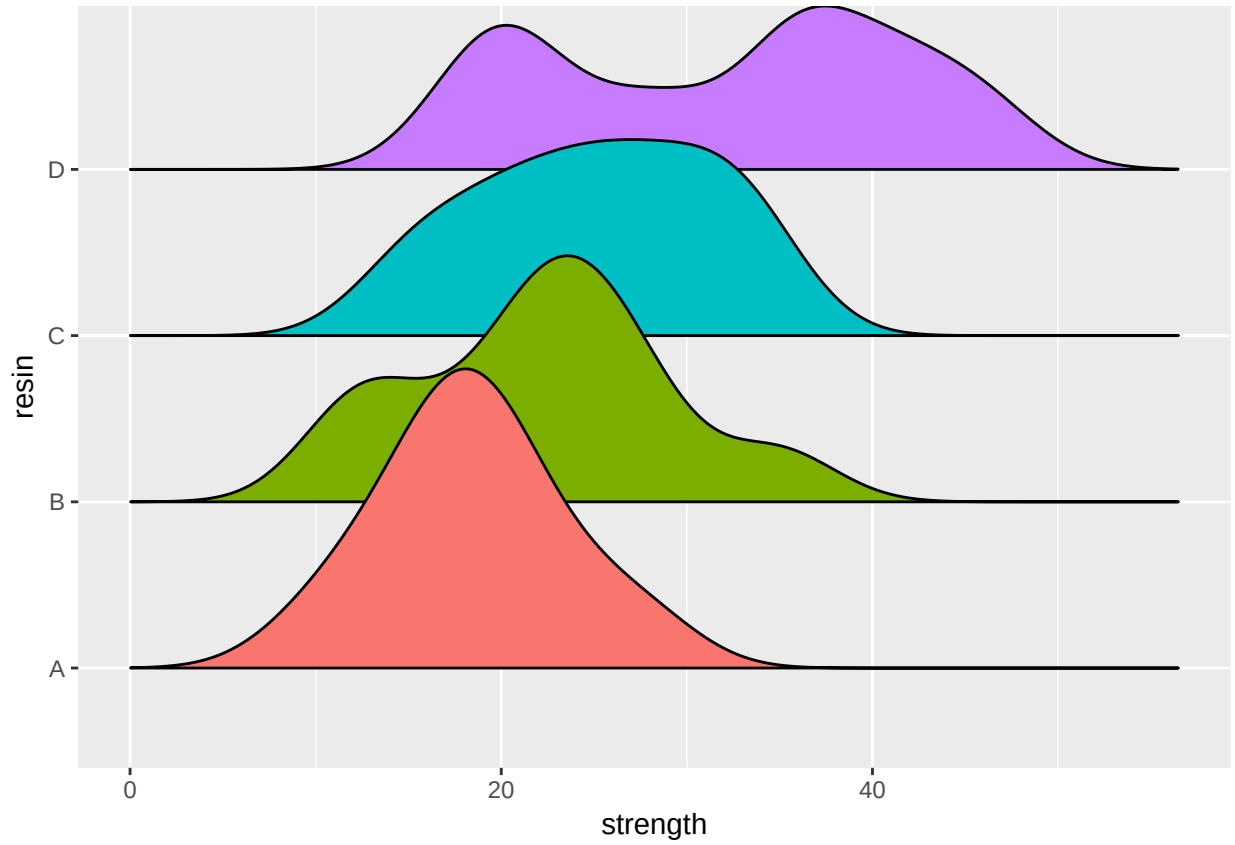
```
ggplot(bonding, aes(x = resin, y = strength)) +
    geom_boxplot()
```



Another good plot for this purpose is a ridgeline plot.

```
ggplot(bonding, aes(x = strength, y = resin, fill = resin)) +
    geom_density_ridges2() +
    guides(fill = FALSE)
```

```
Picking joint bandwidth of 3.09
```



3.2.2 Table of Summary Statistics

With the small size of this experiment ($n = 20$ for each `resin` type), graphical summaries may not perform as well as they often do. We'll also produce a quick table of summary statistics for `strength` within each `resin` type, with the `skim()` function.

```
bonding %>% group_by(resin) %>% skim(strength)
```

```
Skim summary statistics
 n obs: 80
 n variables: 4
 group variables: resin

Variable type: numeric
 resin variable missing complete  n  mean   sd   p0   p25 median   p75
     A strength       0       20 20 18.41 4.81  9.3 15.73  17.95 20.4
     B strength       0       20 20 22.23 6.75 11.8 18.45  22.7  25.75
     C strength       0       20 20 25.16 6.33 14.5 20.65  25.7  30.7
     D strength       0       20 20 32.08 9.74 17.3 21.82  35.3  40.15
 p100
 28
 35.2
 34.5
 47.2
```

Since the means and medians within each group are fairly close, and the distributions (with the possible exception of `resin` D) are reasonably well approximated by the Normal, I'll fit an ANOVA model[2].

```
anova(lm(strength ~ resin, data = bonding))
```

```
Analysis of Variance Table

Response: strength
          Df Sum Sq Mean Sq F value   Pr(>F)
resin      3 1999.7  666.57  13.107 5.52e-07 ***
Residuals 76 3865.2   50.86
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It appears that the `resin` types have a significant association with mean `strength` of the bonds. Can we identify which `resin` types have generally higher or lower `strength`?

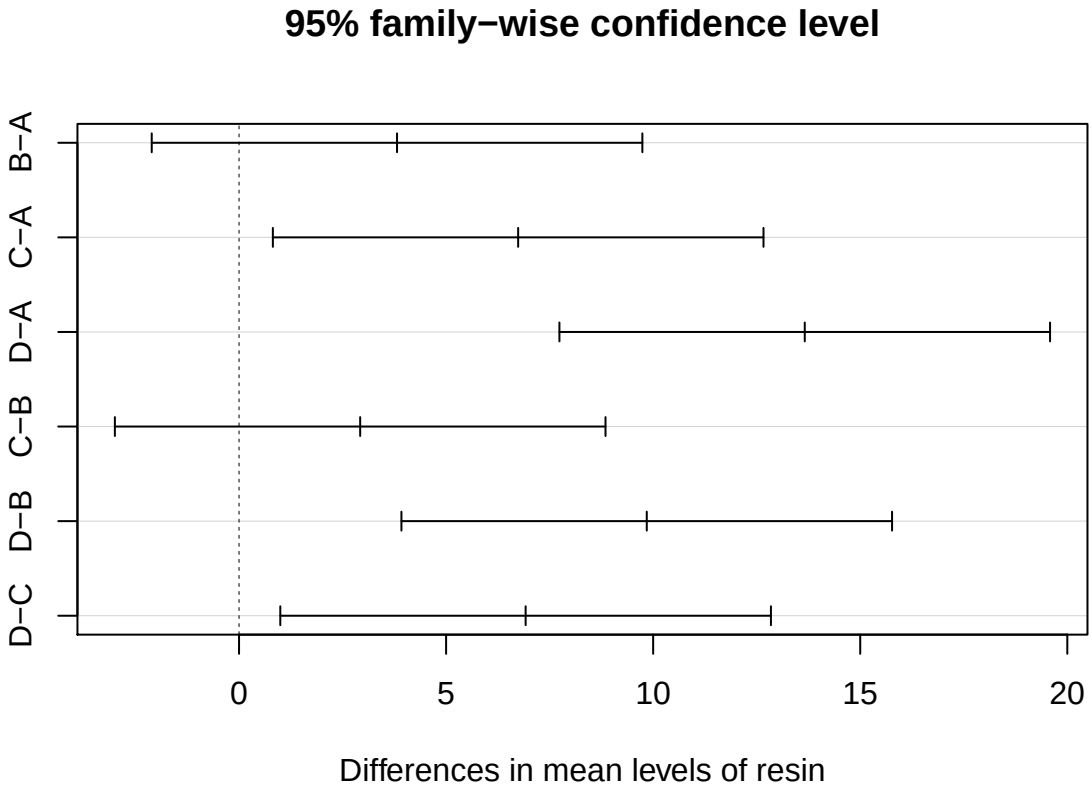
```
TukeyHSD(aov(lm(strength ~ resin, data = bonding)))
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = lm(strength ~ resin, data = bonding))

$resin
      diff        lwr       upr     p adj
B-A  3.815 -2.1088676  9.738868 0.3351635
C-A  6.740  0.8161324 12.663868 0.0193344
D-A 13.660  7.7361324 19.583868 0.0000003
C-B  2.925 -2.9988676  8.848868 0.5676635
D-B  9.845  3.9211324 15.768868 0.0002276
D-C  6.920  0.9961324 12.843868 0.0154615
```

Based on these confidence intervals (which have a family-wise 95% confidence level), we see that D is associated with significantly larger mean `strength` than A or B or C, and that C is also associated with significantly larger mean `strength` than A. This may be easier to see in a plot of these confidence intervals.

```
plot(TukeyHSD(aov(lm(strength ~ resin, data = bonding))))
```

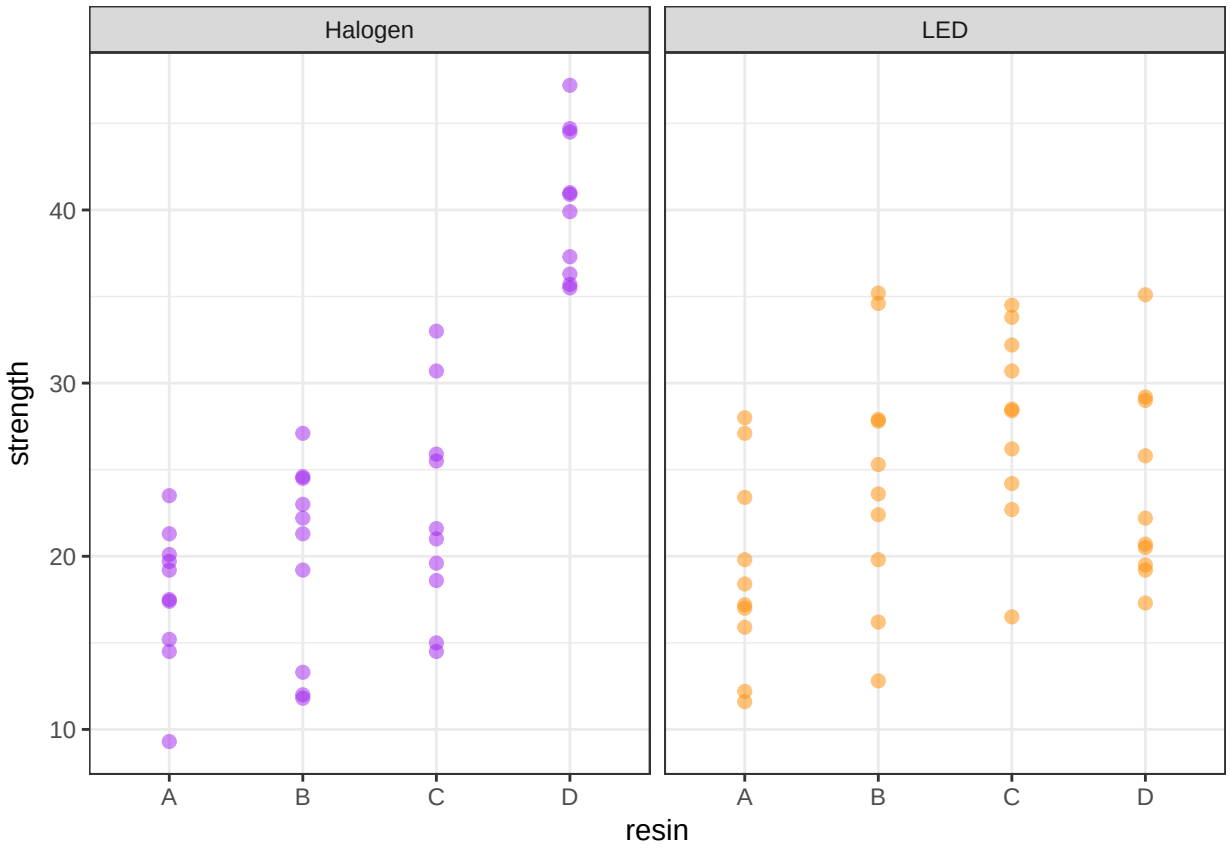
[2] If the data weren't approximately Normally distributed, we might instead consider a rank-based alternative to ANOVA, like the Kruskal-Wallis test.



3.3 A Two-Way ANOVA: Looking at Two Factors

Now, we'll now add consideration of the `light` source into our study. We can look at the distribution of the `strength` values at the combinations of both `light` and `resin`, with a plot like this one…

```
ggplot(bonding, aes(x = resin, y = strength, color = light)) +
    geom_point(size = 2, alpha = 0.5) +
    facet_wrap(~ light) +
    guides(color = FALSE) +
    scale_color_manual(values = c("purple", "darkorange")) +
    theme_bw()
```



3.4 A Means Plot (with standard deviations) to check for interaction

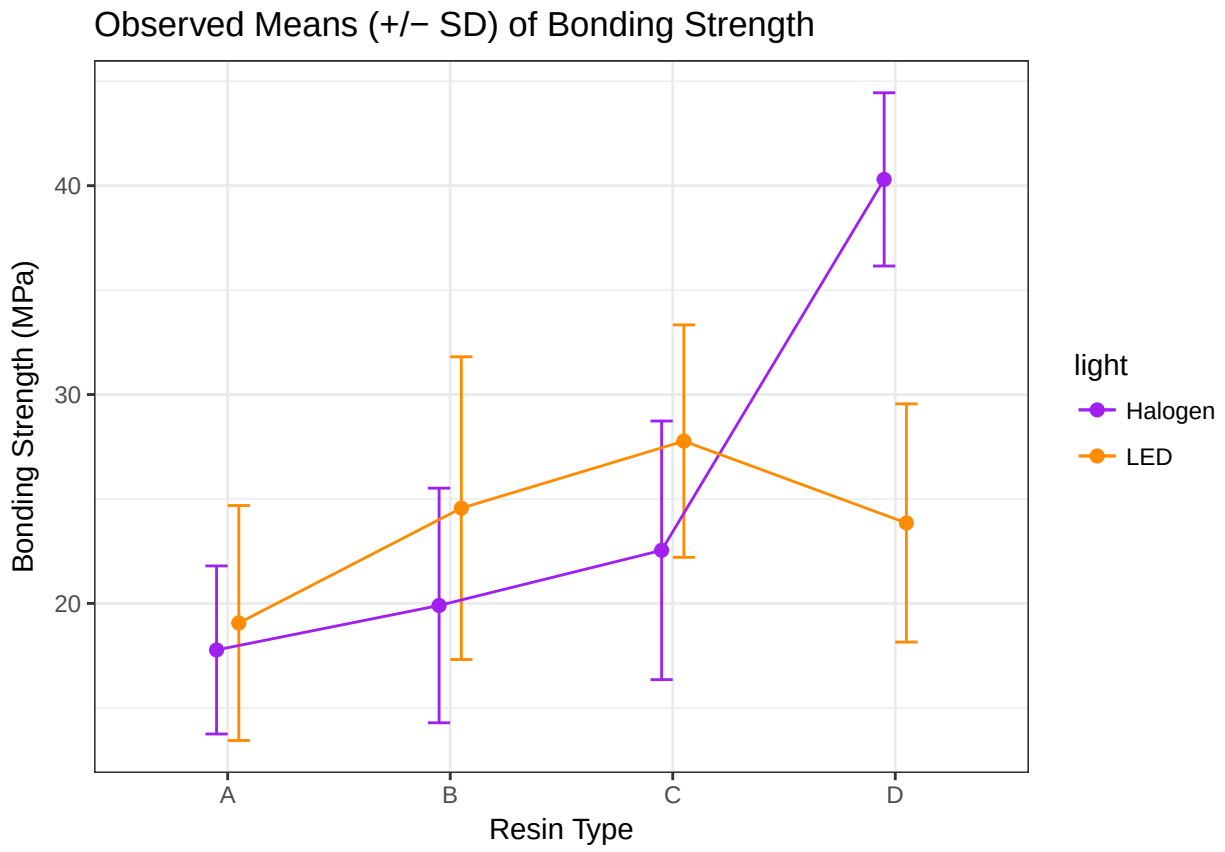
Sometimes, we'll instead look at a plot simply of the means (and, often, the standard deviations) of `strength` at each combination of `light` and `resin`. We'll start by building up a data set with the summaries we want to plot.

```
bond.sum <- bonding %>%
    group_by(resin, light) %>%
    summarize(mean.str = mean(strength), sd.str = sd(strength))

bond.sum
```

```
# A tibble: 8 x 4
# Groups:   resin [?]
  resin light   mean.str sd.str
  <fct> <fct>      <dbl>  <dbl>
1 A     Halogen     17.8   4.02
2 A     LED         19.1   5.63
3 B     Halogen     19.9   5.62
4 B     LED         24.6   7.25
5 C     Halogen     22.5   6.19
6 C     LED         27.8   5.56
7 D     Halogen     40.3   4.15
8 D     LED         23.8   5.70
```

Now, we'll use this new data set to plot the means and standard deviations of `strength` at each combination of `resin` and `light`.

```
## The error bars will overlap unless we adjust the position.
pd <- position_dodge(0.2) # move them .1 to the left and right

ggplot(bond.sum, aes(x = resin, y = mean.str, col = light)) +
    geom_errorbar(aes(ymin = mean.str - sd.str,
                      ymax = mean.str + sd.str),
                  width = 0.2, position = pd) +
    geom_point(size = 2, position = pd) +
    geom_line(aes(group = light), position = pd) +
    scale_color_manual(values = c("purple", "darkorange")) +
    theme_bw() +
    labs(y = "Bonding Strength (MPa)", x = "Resin Type",
         title = "Observed Means (+/- SD) of Bonding Strength")
```



Is there evidence of a meaningful interaction between the resin type and the `light` source on the bonding strength in this plot?

- Sure. A meaningful interaction just means that the strength associated with different `resin` types depends on the `light` source.
  - With LED `light`, it appears that `resin` C leads to the strongest bonding strength.
  - With Halogen `light`, though, it seems that `resin` D is substantially stronger.
- Note that the lines we see here connecting the `light` sources aren't in parallel (as they would be if we had zero interaction between `resin` and `light`), but rather, they cross.

3.4.1 Skimming the data after grouping by `resin` and `light`

We might want to look at a numerical summary of the `strengths` within these groups, too.

```
bonding %>%
    group_by(resin, light) %>%
    skim(strength)
```

```
Skim summary statistics
 n obs: 80
 n variables: 4
 group variables: resin, light

Variable type: numeric
 resin   light variable missing complete  n  mean   sd   p0   p25 median
     A Halogen strength       0       10 10 17.77 4.02  9.3 15.75  18.35
     A     LED strength       0       10 10 19.06 5.63 11.6 16.18  17.8
     B Halogen strength       0       10 10 19.9  5.62 11.8 14.78  21.75
     B     LED strength       0       10 10 24.56 7.25 12.8 20.45  24.45
     C Halogen strength       0       10 10 22.54 6.19 14.5 18.85  21.3
     C     LED strength       0       10 10 27.77 5.56 16.5 24.7   28.45
     D Halogen strength       0       10 10 40.3  4.15 35.5 36.55  40.4
     D     LED strength       0       10 10 23.85 5.7  17.3 19.75  21.45
   p75 p100
 20    23.5
 22.5  28
 24.12 27.1
 27.87 35.2
 25.8  33
 31.83 34.5
 43.62 47.2
 28.2  35.1
```

3.5 Fitting the Two-Way ANOVA model with Interaction

```
c3_m1 <- lm(strength ~ resin * light, data = bonding)

summary(c3_m1)
```

```
Call:
lm(formula = strength ~ resin * light, data = bonding)

Residuals:
    Min      1Q  Median      3Q     Max
-11.760  -3.663  -0.320   3.697  11.250

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)      17.770      1.771  10.033 2.57e-15 ***
resinB            2.130      2.505   0.850   0.3979
resinC            4.770      2.505   1.904   0.0609 .
resinD           22.530      2.505   8.995 2.13e-13 ***
```

```
lightLED           1.290      2.505   0.515   0.6081
resinB:lightLED    3.370      3.542   0.951   0.3446
resinC:lightLED    3.940      3.542   1.112   0.2697
resinD:lightLED  -17.740      3.542  -5.008 3.78e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.601 on 72 degrees of freedom
Multiple R-squared:  0.6149,    Adjusted R-squared:  0.5775
F-statistic: 16.42 on 7 and 72 DF,  p-value: 9.801e-13
```

3.5.1 The ANOVA table for our model

In a two-way ANOVA model, we begin by assessing the interaction term. If it's important, then our best model is the model including the interaction. If it's not important, we will often move on to consider a new model, fit without an interaction.

The ANOVA table is especially helpful in this case, because it lets us look specifically at the interaction effect.

```
anova(c3_m1)
```

```
Analysis of Variance Table

Response: strength
            Df  Sum Sq Mean Sq F value    Pr(>F)
resin        3 1999.72  666.57 21.2499 5.792e-10 ***
light        1   34.72   34.72  1.1067    0.2963
resin:light  3 1571.96  523.99 16.7043 2.457e-08 ***
Residuals   72 2258.52   31.37
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.5.2 Is the interaction important?

In this case, the interaction:

- is evident in the means plot, and
- is highly statistically significant, and
- accounts for a sizeable fraction (27%) of the overall variation

$$\eta^2_{interaction} = \frac{\text{SS(resin:light)}}{SS(Total)} = \frac{1571.96}{1999.72 + 34.72 + 1571.96 + 2258.52} = 0.268$$

If the interaction were *either* large or significant we would be inclined to keep it in the model. In this case, it's both, so there's no real reason to remove it.

3.5.3 Interpreting the Interaction

Recall the model equation, which is:

```
c3_m1
```

```
Call:
lm(formula = strength ~ resin * light, data = bonding)

Coefficients:
    (Intercept)           resinB           resinC           resinD
          17.77             2.13             4.77            22.53
       lightLED  resinB:lightLED  resinC:lightLED  resinD:lightLED
           1.29             3.37             3.94           -17.74
```

so we have:

$$strength = 17.77 + 2.13 resinB + 4.77 resinC + 22.53 resinD + 1.29 lightLED + 3.37 resinB * lightLED + 3.94 resinC * lightLED -$$

So, if `light` = Halogen, our equation is:

$$strength = 17.77 + 2.13 resinB + 4.77 resinC + 22.53 resinD$$

And if `light` = LED, our equation is:

$$strength = 19.06 + 5.50 resinB + 8.71 resinC + 4.79 resinD$$

Note that both the intercept and the slopes change as a result of the interaction. The model yields a different prediction for every possible combination of a `resin` type and a `light` source.

3.6 Comparing Individual Combinations of `resin` and `light`

To make comparisons between individual combinations of a `resin` type and a `light` source, using something like Tukey's HSD approach for multiple comparisons, we first refit the model using the `aov` structure, rather than `lm`.

```
c3m1_aov <- aov(strength ~ resin * light, data = bonding)

summary(c3m1_aov)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
resin        3 1999.7   666.6  21.250 5.79e-10 ***
light        1   34.7    34.7   1.107    0.296
resin:light  3 1572.0   524.0  16.704 2.46e-08 ***
Residuals   72 2258.5    31.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And now, we can obtain Tukey HSD comparisons (which will maintain an overall 95% family-wise confidence level) across the `resin` types, the `light` sources, and the combinations, with the TukeyHSD command. This approach is only completely appropriate if these comparisons are pre-planned, and if the design is balanced (as this is, with the same sample size for each combination of a `light` source and `resin` type.)

```
TukeyHSD(c3m1_aov)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level
```

```
Fit: aov(formula = strength ~ resin * light, data = bonding)

$resin
      diff       lwr       upr     p adj
B-A  3.815 -0.843129  8.473129 0.1461960
C-A  6.740  2.081871 11.398129 0.0016436
D-A 13.660  9.001871 18.318129 0.0000000
C-B  2.925 -1.733129  7.583129 0.3568373
D-B  9.845  5.186871 14.503129 0.0000026
D-C  6.920  2.261871 11.578129 0.0011731

$light
               diff       lwr      upr     p adj
LED-Halogen -1.3175 -3.814042 1.179042 0.2963128

$`resin:light`
                        diff          lwr        upr     p adj
B:Halogen-A:Halogen     2.13  -5.68928258   9.949283 0.9893515
C:Halogen-A:Halogen     4.77  -3.04928258  12.589283 0.5525230
D:Halogen-A:Halogen    22.53  14.71071742  30.349283 0.0000000
A:LED-A:Halogen         1.29  -6.52928258   9.109283 0.9995485
B:LED-A:Halogen         6.79  -1.02928258  14.609283 0.1361092
C:LED-A:Halogen        10.00   2.18071742  17.819283 0.0037074
D:LED-A:Halogen         6.08  -1.73928258  13.899283 0.2443200
C:Halogen-B:Halogen     2.64  -5.17928258  10.459283 0.9640100
D:Halogen-B:Halogen    20.40  12.58071742  28.219283 0.0000000
A:LED-B:Halogen        -0.84  -8.65928258   6.979283 0.9999747
B:LED-B:Halogen         4.66  -3.15928258  12.479283 0.5818695
C:LED-B:Halogen         7.87   0.05071742  15.689283 0.0473914
D:LED-B:Halogen         3.95  -3.86928258  11.769283 0.7621860
D:Halogen-C:Halogen    17.76   9.94071742  25.579283 0.0000000
A:LED-C:Halogen        -3.48 -11.29928258   4.339283 0.8591455
B:LED-C:Halogen         2.02  -5.79928258   9.839283 0.9922412
C:LED-C:Halogen         5.23  -2.58928258  13.049283 0.4323859
D:LED-C:Halogen         1.31  -6.50928258   9.129283 0.9995004
A:LED-D:Halogen       -21.24 -29.05928258 -13.420717 0.0000000
B:LED-D:Halogen       -15.74 -23.55928258  -7.920717 0.0000006
C:LED-D:Halogen       -12.53 -20.34928258  -4.710717 0.0001014
D:LED-D:Halogen       -16.45 -24.26928258  -8.630717 0.0000002
B:LED-A:LED             5.50  -2.31928258  13.319283 0.3665620
C:LED-A:LED             8.71   0.89071742  16.529283 0.0185285
D:LED-A:LED             4.79  -3.02928258  12.609283 0.5471915
C:LED-B:LED             3.21  -4.60928258  11.029283 0.9027236
D:LED-B:LED            -0.71  -8.52928258   7.109283 0.9999920
D:LED-C:LED            -3.92 -11.73928258   3.899283 0.7690762
```

One conclusion from this is that the combination of D and Halogen is significantly stronger than each of the other seven combinations.

3.7 The bonding model without Interaction

It seems incorrect in this situation to fit a model without the interaction term, but we'll do so just so you can see what's involved.

```
c3_m2 <- lm(strength ~ resin + light, data = bonding)

summary(c3_m2)
```

```
Call:
lm(formula = strength ~ resin + light, data = bonding)

Residuals:
     Min       1Q   Median       3Q      Max
-14.1163  -4.9531   0.1187   4.4613  14.4663

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   19.074      1.787  10.676  < 2e-16 ***
resinB         3.815      2.260   1.688  0.09555 .
resinC         6.740      2.260   2.982  0.00386 **
resinD        13.660      2.260   6.044 5.39e-08 ***
lightLED      -1.317      1.598  -0.824  0.41229
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.147 on 75 degrees of freedom
Multiple R-squared:  0.3469,    Adjusted R-squared:  0.312
F-statistic: 9.958 on 4 and 75 DF,  p-value: 1.616e-06
```

In the no-interaction model, if `light` = Halogen, our equation is:

$$strength = 19.07 + 3.82 resinB + 6.74 resinC + 13.66 resinD$$

And if `light` = LED, our equation is:

$$strength = 17.75 + 3.82 resinB + 6.74 resinC + 13.66 resinD$$

So, in the no-interaction model, only the intercept changes.

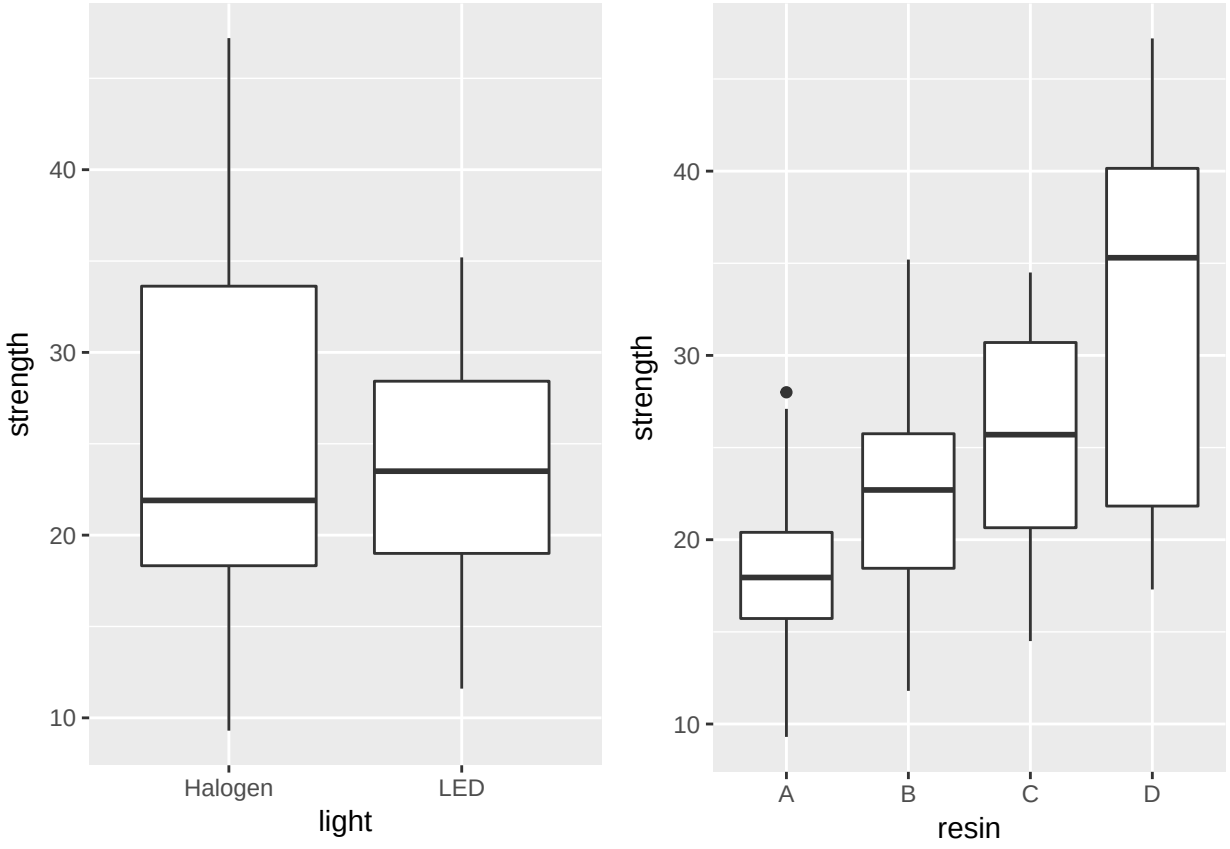
```
anova(c3_m2)
```

```
Analysis of Variance Table

Response: strength
          Df Sum Sq Mean Sq F value    Pr(>F)
resin      3 1999.7  666.57 13.0514 6.036e-07 ***
light      1   34.7   34.72  0.6797    0.4123
Residuals 75 3830.5   51.07
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And, it appears, if we ignore the interaction, then `resin` type has a significant impact on `strength` but `light` source doesn't. This is clearer when we look at boxplots of the separated `light` and `resin` groups.

```
p1 <- ggplot(bonding, aes(x = light, y = strength)) +
    geom_boxplot()
p2 <- ggplot(bonding, aes(x = resin, y = strength)) +
    geom_boxplot()
```

```
gridExtra::grid.arrange(p1, p2, nrow = 1)
```



3.8 `cortisol`: A Hypothetical Clinical Trial

156 adults who complained of problems with a high-stress lifestyle were enrolled in a hypothetical clinical trial of the effectiveness of a behavioral intervention designed to help reduce stress levels, as measured by salivary cortisol.

The subjects were randomly assigned to one of three intervention groups (usual care, low dose, and high dose.) The "low dose" subjects received a one-week intervention with a follow-up at week 5. The "high dose" subjects received a more intensive three-week intervention, with follow up at week 5.

Since cortisol levels rise and fall with circadian rhythms, the cortisol measurements were taken just after rising for all subjects. These measurements were taken at baseline, and again at five weeks. The difference (baseline - week 5) in cortisol level (in micrograms / l) serves as the primary outcome.

3.8.1 Codebook and Raw Data for `cortisol`

The data are gathered in the `cortisol` data set. Included are:

Variable	Description
`subject`	subject identification code
`interv`	intervention group (UC = usual care, Low, High)
`waist`	waist circumference at baseline (in inches)

Variable	Description
sex	male or female
cort.1	salivary cortisol level (microg/l) week 1
cort.5	salivary cortisol level (microg/l) week 5

```
cortisol
```

```
# A tibble: 156 x 6
   subject interv waist sex   cort.1 cort.5
     <int> <fct>  <dbl> <fct>  <dbl>  <dbl>
 1    1001 UC      48.3 M      13.4   13.3
 2    1002 Low     58.3 M      17.8   16.6
 3    1003 High    43.0 M      14.4   12.7
 4    1004 Low     44.9 M       9.00   9.80
 5    1005 High    46.1 M      14.2   14.2
 6    1006 UC      41.3 M      14.8   15.1
 7    1007 Low     51.0 F      13.7   16.0
 8    1008 UC      42.0 F      17.3   18.7
 9    1009 Low     24.7 F      15.3   15.8
10    1010 Low     59.4 M      12.4   11.7
# ... with 146 more rows
```

3.9 Creating a factor combining sex and waist

Next, we'll put the `waist` and `sex` data in the `cortisol` example together. We want to build a second categorical variable (called `fat_est`) combining this information, to indicate "healthy" vs. "unhealthy" levels of fat around the waist.

- Male subjects whose waist circumference is 40 inches or more, and
- Female subjects whose waist circumference is 35 inches or more, will fall in the "unhealthy" group.

```
cortisol <- cortisol %>%
    mutate(
        fat_est = factor(case_when(
            sex == "M" & waist >= 40 ~ "unhealthy",
            sex == "F" & waist >= 35 ~ "unhealthy",
            TRUE                     ~ "healthy")),
        cort_diff = cort.1 - cort.5)

summary(cortisol)
```

```
    subject      interv       waist       sex        cort.1
 Min.   :1001   High:53   Min.   :20.80   F:83   Min.   : 6.000
 1st Qu.:1040   Low :52   1st Qu.:33.27   M:73   1st Qu.: 9.675
 Median :1078   UC  :51   Median :40.35          Median :12.400
 Mean   :1078             Mean   :40.42          Mean   :12.686
 3rd Qu.:1117             3rd Qu.:47.77          3rd Qu.:16.025
 Max.   :1156             Max.   :59.90          Max.   :19.000
     cort.5          fat_est      cort_diff
 Min.   : 4.2   healthy  : 56   Min.   :-2.3000
 1st Qu.: 9.6   unhealthy:100   1st Qu.:-0.5000
 Median :12.6                   Median : 0.2000
 Mean   :12.4                   Mean   : 0.2821
```

```
 3rd Qu.:15.7                    3rd Qu.: 1.2000
 Max.   :19.7                    Max.   : 2.0000
```

3.10 A Means Plot for the cortisol trial (with standard errors)

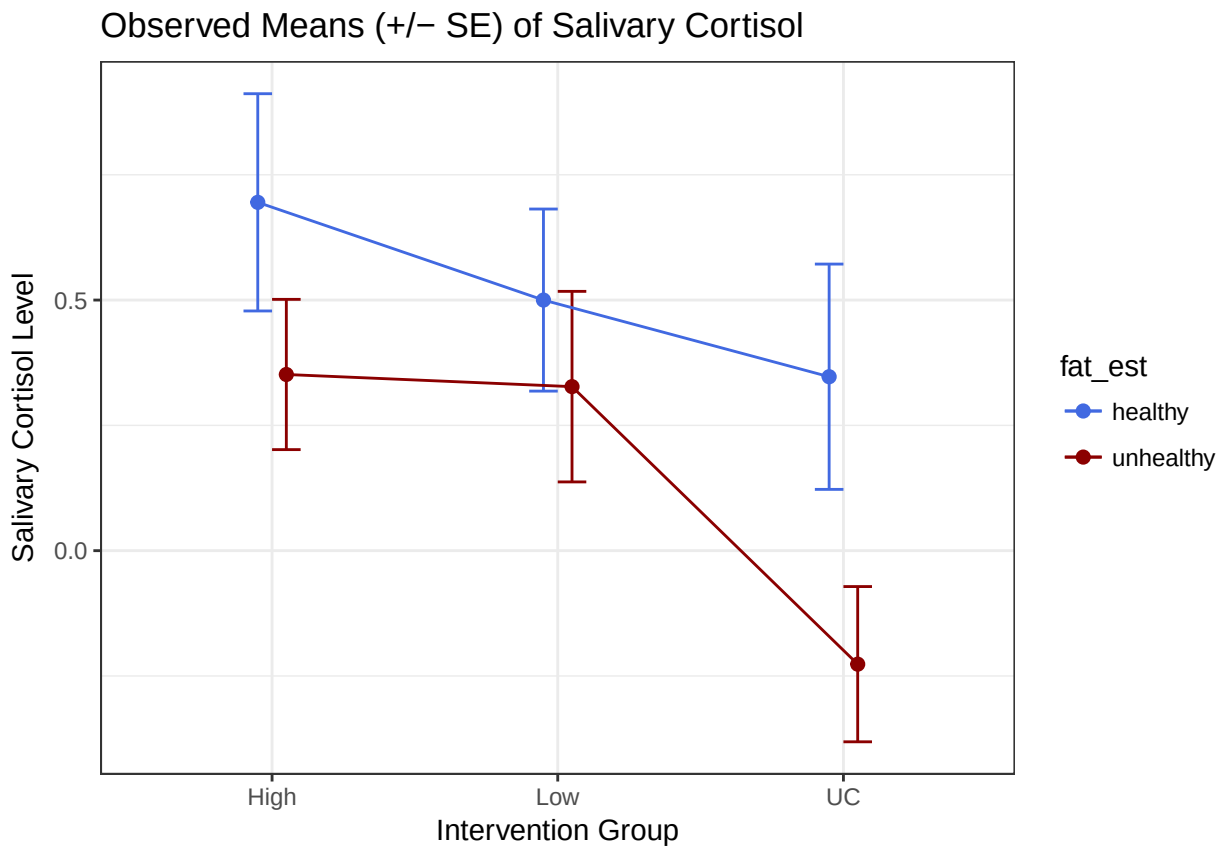
Again, we'll start by building up a data set with the summaries we want to plot.

```
cort.sum <- cortisol %>%
    group_by(interv, fat_est) %>%
    summarize(mean.cort = mean(cort_diff),
              se.cort = sd(cort_diff)/sqrt(n()))

cort.sum
```

```
# A tibble: 6 x 4
# Groups:   interv [?]
  interv fat_est   mean.cort se.cort
  <fct>  <fct>         <dbl>   <dbl>
1 High   healthy       0.695   0.217
2 High   unhealthy     0.352   0.150
3 Low    healthy       0.500   0.182
4 Low    unhealthy     0.327   0.190
5 UC     healthy       0.347   0.225
6 UC     unhealthy    -0.226   0.155
```

Now, we'll use this new data set to plot the means and standard errors.

```
## The error bars will overlap unless we adjust the position.
pd <- position_dodge(0.2) # move them .1 to the left and right

ggplot(cort.sum, aes(x = interv, y = mean.cort, col = fat_est)) +
    geom_errorbar(aes(ymin = mean.cort - se.cort,
                      ymax = mean.cort + se.cort),
                  width = 0.2, position = pd) +
    geom_point(size = 2, position = pd) +
    geom_line(aes(group = fat_est), position = pd) +
    scale_color_manual(values = c("royalblue", "darkred")) +
    theme_bw() +
    labs(y = "Salivary Cortisol Level", x = "Intervention Group",
         title = "Observed Means (+/- SE) of Salivary Cortisol")
```



3.11 A Two-Way ANOVA model for `cortisol` with Interaction

```
c3_m3 <- lm(cort_diff ~ interv * fat_est, data = cortisol)

anova(c3_m3)
```

```
Analysis of Variance Table

Response: cort_diff
               Df  Sum Sq Mean Sq F value  Pr(>F)
interv          2   7.847  3.9235  4.4698 0.01301 *
fat_est         1   4.614  4.6139  5.2564 0.02326 *
interv:fat_est  2   0.943  0.4715  0.5371 0.58554
Residuals     150 131.666  0.8778
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Does it seem like we need the interaction term in this case?

```
summary(c3_m3)
```

```
Call:
lm(formula = cort_diff ~ interv * fat_est, data = cortisol)

Residuals:
```

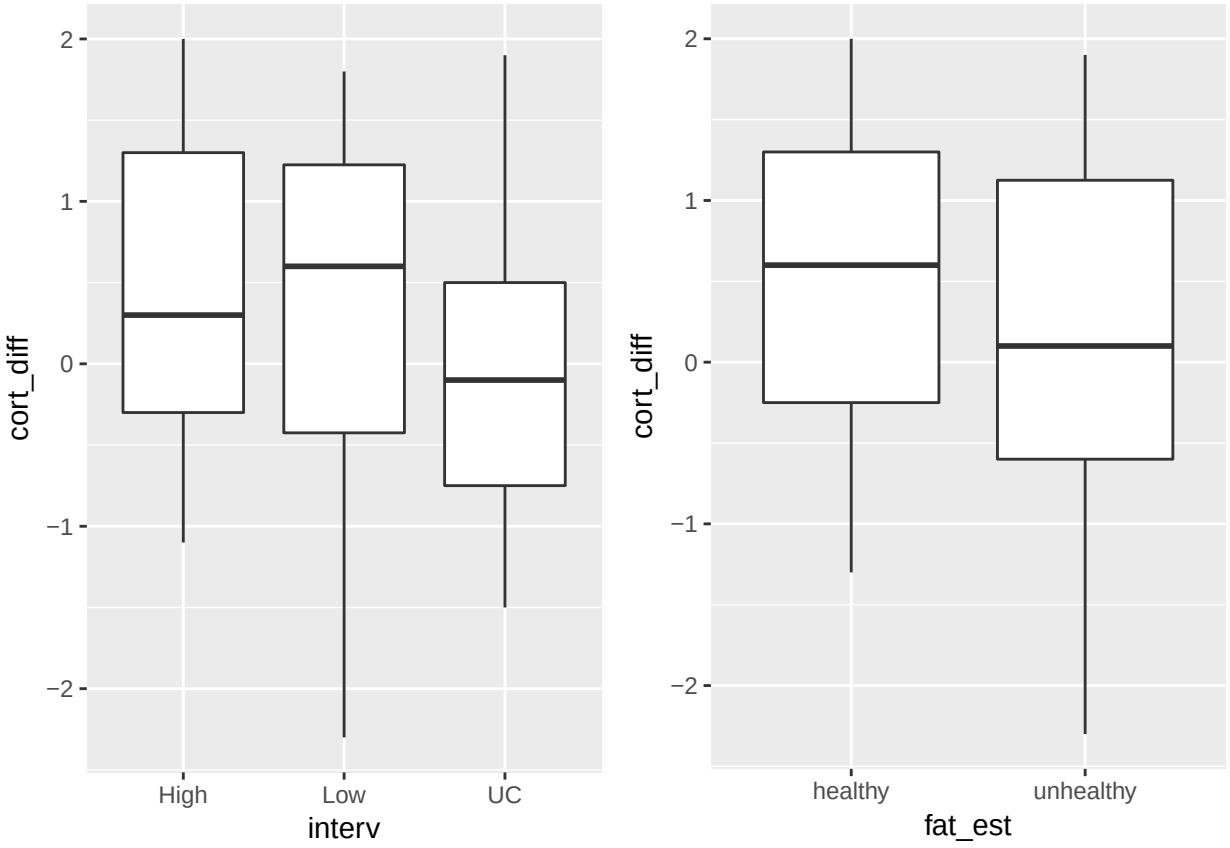
```
     Min       1Q   Median       3Q      Max
-2.62727 -0.75702  0.08636  0.84848  2.12647

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)                  0.6950     0.2095   3.317  0.00114 **
intervLow                   -0.1950     0.3001  -0.650  0.51689
intervUC                    -0.3479     0.3091  -1.126  0.26206
fat_estunhealthy            -0.3435     0.2655  -1.294  0.19774
intervLow:fat_estunhealthy   0.1708     0.3785   0.451  0.65256
intervUC:fat_estunhealthy   -0.2300     0.3846  -0.598  0.55068
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9369 on 150 degrees of freedom
Multiple R-squared:  0.0924,    Adjusted R-squared:  0.06214
F-statistic: 3.054 on 5 and 150 DF,  p-value: 0.01179
```

How do you reconcile the apparent difference in significance levels between this regression summary and the ANOVA table above?

3.12 A Two-Way ANOVA model for `cortisol` without Interaction

3.12.1 The Graph

```
p1 <- ggplot(cortisol, aes(x = interv, y = cort_diff)) +
    geom_boxplot()
p2 <- ggplot(cortisol, aes(x = fat_est, y = cort_diff)) +
    geom_boxplot()

gridExtra::grid.arrange(p1, p2, nrow = 1)
```



3.12.2 The ANOVA Model

```
c3_m4 <- lm(cort_diff ~ interv + fat_est, data = cortisol)

anova(c3_m4)
```

```
Analysis of Variance Table

Response: cort_diff
           Df  Sum Sq Mean Sq F value  Pr(>F)
interv      2   7.847  3.9235  4.4972 0.01266 *
fat_est     1   4.614  4.6139  5.2886 0.02283 *
Residuals 152 132.609  0.8724
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

How do these results compare to those we saw in the model with interaction?

3.12.3 The Regression Summary

```
summary(c3_m4)
```

```
Call:
lm(formula = cort_diff ~ interv + fat_est, data = cortisol)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-2.55929 -0.74527  0.05457  0.86456  2.05489

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       0.70452    0.16093   4.378 2.22e-05 ***
intervLow        -0.08645    0.18232  -0.474  0.63606
intervUC         -0.50063    0.18334  -2.731  0.00707 **
fat_estunhealthy -0.35878    0.15601  -2.300  0.02283 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.934 on 152 degrees of freedom
Multiple R-squared:  0.0859,    Adjusted R-squared:  0.06785
F-statistic: 4.761 on 3 and 152 DF,  p-value: 0.00335
```

3.12.4 Tukey HSD Comparisons

Without the interaction term, we can make direct comparisons between levels of the intervention, and between levels of the `fat_est` variable. This is probably best done here in a Tukey HSD comparison.

```
TukeyHSD(aov(cort_diff ~ interv + fat_est, data = cortisol))
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = cort_diff ~ interv + fat_est, data = cortisol)

$interv
                diff        lwr         upr     p adj
Low-High -0.09074746 -0.5222655  0.34077063 0.8724916
UC-High  -0.51642619 -0.9500745 -0.08277793 0.0150150
UC-Low   -0.42567873 -0.8613670  0.01000948 0.0570728

$fat_est
                        diff        lwr         upr     p adj
unhealthy-healthy -0.3582443 -0.6662455 -0.05024305 0.0229266
```

What conclusions can we draw, at a 5% significance level?

Chapter 4

Analysis of Covariance

4.1 An Emphysema Study

My source for this example is Riffenburgh (2006), section 18.4. Serum theophylline levels (in mg/dl) were measured in 16 patients with emphysema at baseline, then 5 days later (at the end of a course of antibiotics) and then at 10 days after baseline. Clinicians anticipate that the antibiotic will increase the theophylline level. The data are stored in the `emphysema.csv` data file, and note that the age for patient 5 is not available.

4.1.1 Codebook

Variable	Description
patient	ID code
age	patient's age in years
sex	patient's sex (F or M)
st_base	patient's serum theophylline at baseline (mg/dl)
st_day5	patient's serum theophylline at day 5 (mg/dl)
st_day10	patient's serum theophylline at day 10 (mg/dl)

We're going to look at the change from baseline to day 5 as our outcome of interest, since the clinical expectation is that the antibiotic (azithromycin) will increase theophylline levels.

```
emphysema <- emphysema %>%
    mutate(st_delta = st_day5 - st_base)

emphysema
```

```
# A tibble: 16 x 7
   patient   age sex   st_base st_day5 st_day10 st_delta
     <int> <int> <fct>   <dbl>   <dbl>    <dbl>    <dbl>
 1       1    61 F       14.1     2.30    10.3   -11.8
 2       2    70 F        7.20    5.40     7.30  - 1.80
 3       3    65 M       14.2    11.9     11.3   - 2.30
 4       4    65 M       10.3    10.7     13.8     0.400
 5       5    NA M        9.90   10.7     11.7     0.800
 6       6    76 M        5.20    6.80     4.20    1.60
 7       7    72 M       10.4    14.6     14.1     4.20
 8       8    69 F       10.5     7.20     5.40  - 3.30
```

```
 9       9    66 M        5.00    5.00     5.10    0
10      10    62 M        8.60    8.10     7.40  - 0.500
11      11    65 F       16.6    14.9     13.0   - 1.70
12      12    71 M       16.4    18.6     17.1     2.20
13      13    51 F       12.2    11.0     12.3   - 1.20
14      14    71 M        6.60    3.70     4.50  - 2.90
15      15    64 F       15.4    15.2     13.6   - 0.200
16      16    50 M       10.2    10.8     11.2     0.600
```

4.2 Does sex affect the mean change in theophylline?

```
emphysema %>% skim(st_delta)
```

```
Skim summary statistics
 n obs: 16
 n variables: 7

Variable type: numeric
 variable missing complete  n  mean   sd    p0   p25 median  p75 p100
 st_delta       0       16 16 -0.99 3.48 -11.8 -1.92  -0.35 0.65  4.2
```

```
emphysema %>% group_by(sex) %>% skim(st_delta)
```

```
Skim summary statistics
 n obs: 16
 n variables: 7
 group variables: sex

Variable type: numeric
 sex variable missing complete  n  mean   sd    p0   p25 median   p75 p100
   F st_delta       0        6  6 -3.33 4.27 -11.8 -2.92  -1.75 -1.32 -0.2
   M st_delta       0       10 10  0.41 2.07  -2.9 -0.38   0.5   1.4   4.2
```

Overall, the mean change in theophylline during the course of the antibiotic is -0.99, but this is -3.33 for female patients and 0.41 for male patients.

A one-way ANOVA model looks like this:

```
anova(lm(st_delta ~ sex, data = emphysema))
```

```
Analysis of Variance Table

Response: st_delta
          Df  Sum Sq Mean Sq F value  Pr(>F)
sex        1  52.547  52.547  5.6789 0.03189 *
Residuals 14 129.542   9.253
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA F test finds a statistically significant difference between the mean `st_delta` among males and the mean `st_delta` among females. But is there more to the story?

4.3 Is there an association between age and sex in this study?

```
emphysema %>% group_by(sex) %>% skim(age)
```

```
Skim summary statistics
 n obs: 16
 n variables: 7
 group variables: sex

Variable type: integer
 sex variable missing complete  n  mean   sd p0   p25 median p75 p100
   F      age       0        6  6 63.33 6.89 51 61.75   64.5  68   70
   M      age       1        9 10 66.44 7.57 50 65      66    71   76
```

But we note that the male patients are also older than the female patients, on average (mean age for males is 66.4, for females 63.3)

- Does the fact that male patients are older affect change in theophylline level?
- And how should we deal with the one missing age value (in a male patient)?

4.4 Adding a quantitative covariate, age, to the model

We could fit an ANOVA model to predict st_delta using sex and age directly, but only if we categorized age into two or more groups. Because age is not categorical, we cannot include it in an ANOVA. But if age is an influence, and we don't adjust for it, it may well bias the outcome of our initial ANOVA. With a quantitative variable like age, we will need a method called ANCOVA, for **analysis of covariance**.

4.4.1 The ANCOVA model

ANCOVA in this case is just an ANOVA model with our outcome (st_delta) adjusted for a continuous covariate, called age. For the moment, we'll ignore the one subject with missing age and simply fit the regression model with sex and age.

```
summary(lm(st_delta ~ sex + age, data = emphysema))
```

```

Call:
lm(formula = st_delta ~ sex + age, data = emphysema)

Residuals:
    Min      1Q  Median      3Q     Max
-8.3352 -0.4789  0.6948  1.5580  3.5202

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.90266    7.92948  -0.871   0.4011
sexM         3.52466    1.75815   2.005   0.0681 .
age          0.05636    0.12343   0.457   0.6561
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.255 on 12 degrees of freedom
  (1 observation deleted due to missingness)
```

```
Multiple R-squared:  0.2882,    Adjusted R-squared:  0.1696
F-statistic:  2.43 on 2 and 12 DF,  p-value: 0.13
```

This model assumes that the slope of the regression line between `st_delta` and `age` is the same for both sexes.

Note that the model yields `st_delta` = -6.9 + 3.52 (`sex` = male) + 0.056 `age`, or

- `st_delta` = -6.9 + 0.056 `age` for female patients, and
- `st_delta` = (-6.9 + 3.52) + 0.056 `age` = -3.38 + 0.056 `age` for male patients.

Note that we can test this assumption of equal slopes by fitting an alternative model (with a product term between `sex` and `age`) that doesn't require the assumption, and we'll do that later.

4.4.2 The ANCOVA Table

First, though, we'll look at the ANCOVA table.

```
anova(lm(st_delta ~ sex + age, data = emphysema))
```

```
Analysis of Variance Table

Response: st_delta
          Df  Sum Sq Mean Sq F value  Pr(>F)
sex        1  49.284  49.284  4.6507 0.05203 .
age        1   2.209   2.209  0.2085 0.65612
Residuals 12 127.164  10.597
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

When we tested `sex` without accounting for `age`, we found a p value of 0.032, which is less than our usual cutpoint of 0.05. But when we adjusted for `age`, we find that `sex` loses significance, even though `age` is not a significant influence on `st_delta` by itself, according to the ANCOVA table.

4.5 Rerunning the ANCOVA model after simple imputation

We could have *imputed* the missing `age` value for patient 5, rather than just deleting that patient. Suppose we do the simplest potentially reasonable thing to do: insert the mean `age` in where the NA value currently exists.

```
emph_imp <- replace_na(emphysema, list(age = mean(emphysema$age, na.rm = TRUE)))

emph_imp
```

```
# A tibble: 16 x 7
   patient   age sex   st_base st_day5 st_day10 st_delta
     <int> <dbl> <fct>   <dbl>   <dbl>    <dbl>    <dbl>
 1       1  61.0 F        14.1     2.30    10.3   -11.8
 2       2  70.0 F         7.20    5.40     7.30  - 1.80
 3       3  65.0 M        14.2    11.9     11.3   - 2.30
 4       4  65.0 M        10.3    10.7     13.8     0.400
 5       5  65.2 M         9.90   10.7     11.7     0.800
 6       6  76.0 M         5.20    6.80     4.20    1.60
 7       7  72.0 M        10.4    14.6     14.1     4.20
 8       8  69.0 F        10.5     7.20     5.40  - 3.30
 9       9  66.0 M         5.00    5.00     5.10    0
```

```
10      10  62.0 M        8.60    8.10     7.40  - 0.500
11      11  65.0 F       16.6    14.9     13.0   - 1.70
12      12  71.0 M       16.4    18.6     17.1     2.20
13      13  51.0 F       12.2    11.0     12.3   - 1.20
14      14  71.0 M        6.60    3.70     4.50  - 2.90
15      15  64.0 F       15.4    15.2     13.6   - 0.200
16      16  50.0 M       10.2    10.8     11.2     0.600
```

More on simple imputation and missing data is coming soon.

For now, we can rerun the ANCOVA model on this new data set, after imputation…

```
anova(lm(st_delta ~ sex + age, data = emph_imp))
```

```
Analysis of Variance Table

Response: st_delta
          Df  Sum Sq Mean Sq F value  Pr(>F)
sex        1  52.547  52.547  5.3623 0.03755 *
age        1   2.151   2.151  0.2195 0.64721
Residuals 13 127.392   9.799
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

When we do this, we see that now the `sex` variable returns to a p value below 0.05. Our complete case analysis (which omitted patient 5) gives us a different result than the ANCOVA based on the data after mean imputation.

4.6 Looking at a factor-covariate interaction

Let's run a model including the interaction (product) term between `age` and `sex`, which implies that the slope of `age` on our outcome (`st_delta`) depends on the patient's sex. We'll use the imputed data again. Here is the new ANCOVA table, which suggests that the interaction of `age` and `sex` is small (because it accounts for only a small amount of the total Sum of Squares) and not significant (p = 0.91).

```
anova(lm(st_delta ~ sex * age, data = emph_imp))
```

```
Analysis of Variance Table

Response: st_delta
          Df  Sum Sq Mean Sq F value  Pr(>F)
sex        1  52.547  52.547  4.9549 0.04594 *
age        1   2.151   2.151  0.2028 0.66051
sex:age    1   0.130   0.130  0.0123 0.91355
Residuals 12 127.261  10.605
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since the interaction term is neither substantial nor significant, we probably don't need it here. But let's look at its interpretation anyway, just to fix ideas. To do that, we'll need the coefficients from the underlying regression model.

```
tidy(lm(st_delta ~ sex * age, data = emph_imp))
```

```
         term    estimate  std.error  statistic   p.value
1 (Intercept) -5.64606742 13.4536974 -0.4196666 0.6821446
2        sexM  1.72031026 16.8389209  0.1021627 0.9203148
```

```
3         age  0.03651685  0.2113871  0.1727488 0.8657284
4    sexM:age  0.02885946  0.2603044  0.1108681 0.9135536
```

Our ANCOVA model for `st_delta` incorporating the `age` x `sex` product term is -5.65 + 1.72 (sex = M) + 0.037 age + 0.029 (sex = M)(age). So that means:

- our model for females is `st_delta` = -5.65 + 0.037 `age`
- our model for males is `st_delta` = (-5.65 + 1.72) + (0.037 + 0.029) `age`, or -3.93 + 0.066 `age`

but, again, our conclusion from the ANCOVA table is that this increase in complexity (letting both the slope and intercept vary by `sex`) doesn't add much in the way of predictive value for our `st_delta` outcome.

4.7 Centering the Covariate to Facilitate ANCOVA Interpretation

When developing an ANCOVA model, we will often **center** or even **center and rescale** the covariate to facilitate interpretation of the product term. In this case, let's center `age` and rescale it by dividing by two standard deviations.

```
emph_imp %>% skim(age)
```

```
Skim summary statistics
 n obs: 16
 n variables: 7

Variable type: numeric
 variable missing complete  n mean   sd p0  p25 median   p75 p100
      age       0       16 16 65.2 6.98 50 63.5   65.1 70.25   76
```

Note that in our imputed data, the mean `age` is 65.2 and the standard deviation of `age` is 7 years.

So we build the rescaled `age` variable that I'll call `age_z`, and then use it to refit our model.

```
emph_imp <- emph_imp %>%
    mutate(age_z = (age - mean(age))/ (2 * sd(age)))

anova(lm(st_delta ~ sex * age_z, data = emph_imp))
```

```
Analysis of Variance Table

Response: st_delta
          Df  Sum Sq Mean Sq F value  Pr(>F)
sex        1  52.547  52.547  4.9549 0.04594 *
age_z      1   2.151   2.151  0.2028 0.66051
sex:age_z  1   0.130   0.130  0.0123 0.91355
Residuals 12 127.261  10.605
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
tidy(lm(st_delta ~ sex * age_z, data = emph_imp))
```

```
         term   estimate std.error  statistic    p.value
1 (Intercept) -3.2651685  1.386802 -2.3544587 0.03641637
2        sexM  3.6019471  1.735706  2.0752055 0.06013138
3       age_z  0.5096337  2.950144  0.1727488 0.86572835
4  sexM:age_z  0.4027661  3.632839  0.1108681 0.91355364
```

Comparing the two models, we have:

- (unscaled): `st_delta` = -5.65 + 1.72 (`sex` = M) + 0.037 `age` + 0.029 (`sex` = M) x (`age`)

- (rescaled): `st_delta` = -3.27 + 3.60 (`sex` = M) + 0.510 rescaled `age_z` + 0.402 (`sex` = M) x (rescaled `age_z`)

In essence, the rescaled model on `age_z` is:

- `st_delta` = -3.27 + 0.510 `age_z` for female subjects, and
- `st_delta` = (-3.27 + 3.60) + (0.510 + 0.402) `age_z` = 0.33 + 0.912 `age_z` for male subjects

Interpreting the centered, rescaled model, we have:

- no change in the ANOVA results or R-squared or residual standard deviation compared to the uncentered, unscaled model, but
- the intercept (-3.27) now represents the `st_delta` for a female of average age,
- the `sex` slope (3.60) represents the (male - female) difference in predicted `st_delta` for a person of average age,
- the `age_z` slope (0.510) represents the difference in predicted `st_delta` for a female one standard deviation older than the mean age as compared to a female one standard deviation younger than the mean age, and
- the product term's slope (0.402) represents the male - female difference in the slope of `age_z`, so that if you add the `age_z` slope (0.510) and the interaction slope (0.402) you see the difference in predicted `st_delta` for a male one standard deviation older than the mean age as compared to a male one standard deviation younger than the mean age.

Chapter 5

Missing Data Mechanisms and Single Imputation

Almost all serious statistical analyses have to deal with missing data. Data values that are missing are indicated in R, and to R, by the symbol `NA`.

5.1 A Toy Example

In the following tiny data set called `sbp_example`, we have four variables for a set of 15 subjects. In addition to a subject id, we have:

- the treatment this subject received (A, B or C are the treatments),
- an indicator (1 = yes, 0 = no) of whether the subject has diabetes,
- the subject's systolic blood pressure at baseline
- the subject's systolic blood pressure after the application of the treatment

```
# create some temporary variables

subject <- 101:115
x1 <- c("A", "B", "C", "A", "C", "A", "A", NA, "B", "C", "A", "B", "C", "A", "B")
x2 <- c(1, 0, 0, 1, NA, 1, 0, 1, NA, 1, 0, 0, 1, 1, NA)
x3 <- c(120, 145, 150, NA, 155, NA, 135, NA, 115, 170, 150, 145, 140, 160, 135)
x4 <- c(105, 135, 150, 120, 135, 115, 160, 150, 130, 155, 140, 140, 150, 135, 120)

sbp_example <-
  data.frame(subject, treat = x1, diabetes = x2,
             sbp.before = x3, sbp.after = x4) %>%
  tbl_df

rm(subject, x1, x2, x3, x4) # just cleaning up

sbp_example
```

```
# A tibble: 15 x 5
   subject treat diabetes sbp.before sbp.after
     <int> <fct>    <dbl>      <dbl>     <dbl>
 1     101 A         1.00        120       105
 2     102 B         0           145       135
```

```
 3     103 C         0           150       150
 4     104 A         1.00         NA       120
 5     105 C        NA           155       135
 6     106 A         1.00         NA       115
 7     107 A         0           135       160
 8     108 <NA>      1.00         NA       150
 9     109 B        NA           115       130
10     110 C         1.00        170       155
11     111 A         0           150       140
12     112 B         0           145       140
13     113 C         1.00        140       150
14     114 A         1.00        160       135
15     115 B        NA           135       120
```

5.1.1 How many missing values do we have in each column?

```
colSums(is.na(sbp_example))
```

```
   subject      treat   diabetes sbp.before  sbp.after
         0          1          3          3          0
```

We are missing one `treat`, 3 `diabetes` and 3 `sbp.before` values.

5.1.2 What is the pattern of missing data?

```
mice::md.pattern(sbp_example)
```

```
  subject sbp.after treat diabetes sbp.before
9       1         1     1        1          1 0
3       1         1     1        0          1 1
2       1         1     1        1          0 1
1       1         1     0        1          0 2
        0         0     1        3          3 7
```

We have nine subjects with complete data, three subjects with missing `diabetes` (only), two subjects with missing `sbp.before` (only), and 1 subject with missing `treat` and `sbp.before`.

5.1.3 How can we identify the subjects with missing data?

```
sbp_example %>% filter(!complete.cases(.))
```

```
# A tibble: 6 x 5
  subject treat diabetes sbp.before sbp.after
    <int> <fct>    <dbl>      <dbl>     <dbl>
1     104 A         1.00         NA       120
2     105 C        NA           155       135
3     106 A         1.00         NA       115
4     108 <NA>      1.00         NA       150
5     109 B        NA           115       130
6     115 B        NA           135       120
```

5.2 Missing-data mechanisms

My source for this description of mechanisms is Chapter 25 of Gelman and Hill (2007), and that chapter is available at this link.

1. **MCAR = Missingness completely at random**. A variable is missing completely at random if the probability of missingness is the same for all units, for example, if for each subject, we decide whether to collect the `diabetes` status by rolling a die and refusing to answer if a "6" shows up. If data are missing completely at random, then throwing out cases with missing data does not bias your inferences.
2. **Missingness that depends only on observed predictors**. A more general assumption, called **missing at random** or **MAR**, is that the probability a variable is missing depends only on available information. Here, we would have to be willing to assume that the probability of nonresponse to `diabetes` depends only on the other, fully recorded variables in the data. It is often reasonable to model this process as a logistic regression, where the outcome variable equals 1 for observed cases and 0 for missing. When an outcome variable is missing at random, it is acceptable to exclude the missing cases (that is, to treat them as NA), as long as the regression controls for all the variables that affect the probability of missingness.
3. **Missingness that depends on unobserved predictors**. Missingness is no longer "at random" if it depends on information that has not been recorded and this information also predicts the missing values. If a particular treatment causes discomfort, a patient is more likely to drop out of the study. This missingness is not at random (unless "discomfort" is measured and observed for all patients). If missingness is not at random, it must be explicitly modeled, or else you must accept some bias in your inferences.
4. **Missingness that depends on the missing value itself.** Finally, a particularly difficult situation arises when the probability of missingness depends on the (potentially missing) variable itself. For example, suppose that people with higher earnings are less likely to reveal them.

Essentially, situations 3 and 4 are referred to collectively as **non-random missingness**, and cause more trouble for us than 1 and 2.

5.3 Options for Dealing with Missingness

There are several available methods for dealing with missing data that are MCAR or MAR, but they basically boil down to:

- Complete Case (or Available Case) analyses
- Single Imputation
- Multiple Imputation

5.4 Complete Case (and Available Case) analyses

In **Complete Case** analyses, rows containing NA values are omitted from the data before analyses commence. This is the default approach for many statistical software packages, and may introduce unpredictable bias and fail to include some useful, often hard-won information.

- A complete case analysis can be appropriate when the number of missing observations is not large, and the missing pattern is either MCAR (missing completely at random) or MAR (missing at random.)
- Two problems arise with complete-case analysis:
    1. If the units with missing values differ systematically from the completely observed cases, this could bias the complete-case analysis.
    2. If many variables are included in a model, there may be very few complete cases, so that most of the data would be discarded for the sake of a straightforward analysis.

- A related approach is *available-case* analysis where different aspects of a problem are studied with different subsets of the data, perhaps identified on the basis of what is missing in them.

5.5 Single Imputation

In **single imputation** analyses, NA values are estimated/replaced *one time* with *one particular data value* for the purpose of obtaining more complete samples, at the expense of creating some potential bias in the eventual conclusions or obtaining slightly *less* accurate estimates than would be available if there were no missing values in the data.

- A single imputation can be just a replacement with the mean or median (for a quantity) or the mode (for a categorical variable.) However, such an approach, though easy to understand, underestimates variance and ignores the relationship of missing values to other variables.
- Single imputation can also be done using a variety of models to try to capture information about the NA values that are available in other variables within the data set.
- The `simputation` package can help us execute single imputations using a wide variety of techniques, within the pipe approach used by the `tidyverse`. Another approach I have used in the past is the `mice` package, which can also perform single imputations.

5.6 Multiple Imputation

Multiple imputation, where NA values are repeatedly estimated/replaced with multiple data values, for the purpose of obtaining mode complete samples *and* capturing details of the variation inherent in the fact that the data have missingness, so as to obtain *more* accurate estimates than are possible with single imputation.

- We'll postpone the discussion of multiple imputation for a while.

5.7 Building a Complete Case Analysis

We can drop all of the missing values from a data set with `drop_na` or with `na.omit` or by filtering for `complete.cases`. Any of these approaches produces the same result - a new data set with 9 rows (after dropping the six subjects with any NA values) and 5 columns.

```
cc.1 <- na.omit(sbp_example)
cc.2 <- sbp_example %>% drop_na
cc.3 <- sbp_example %>% filter(complete.cases(.))
```

5.8 Single Imputation with the Mean or Mode

The most straightforward approach to single imputation is to impute a single summary of the variable, such as the mean, median or mode.

```
skim(sbp_example)
```

```
Skim summary statistics
 n obs: 15
 n variables: 5

Variable type: factor
 variable missing complete  n n_unique             top_counts ordered
```

```
    treat       1       14 15        3 A: 6, B: 4, C: 4, NA: 1   FALSE

Variable type: integer
 variable missing complete  n mean   sd  p0   p25 median   p75 p100
  subject       0       15 15  108 4.47 101 104.5    108 111.5  115

Variable type: numeric
   variable missing complete  n   mean    sd  p0 p25 median    p75 p100
   diabetes       3       12 15   0.58  0.51   0   0      1   1       1
  sbp.after       0       15 15 136    15.83 105 125    135 150     160
 sbp.before       3       12 15 143.33 15.72 115 135    145 151.25  170
```

Here, suppose we decide to impute

- `sbp.before` with the mean (143.33) among non-missing values,
- `diabetes` with its median (1) among non-missing values, and
- `treat` with its most common value, or mode (A)

```
si.1 <- sbp_example %>%
    replace_na(list(sbp.before = 143.33,
                    diabetes = 1,
                    treat = "A"))
si.1
```

```
# A tibble: 15 x 5
   subject treat diabetes sbp.before sbp.after
     <int> <fct>    <dbl>      <dbl>     <dbl>
 1     101 A         1.00        120       105
 2     102 B         0           145       135
 3     103 C         0           150       150
 4     104 A         1.00        143       120
 5     105 C         1.00        155       135
 6     106 A         1.00        143       115
 7     107 A         0           135       160
 8     108 A         1.00        143       150
 9     109 B         1.00        115       130
10     110 C         1.00        170       155
11     111 A         0           150       140
12     112 B         0           145       140
13     113 C         1.00        140       150
14     114 A         1.00        160       135
15     115 B         1.00        135       120
```

We could accomplish the same thing with, for example:

```
si.2 <- sbp_example %>%
    replace_na(list(sbp.before = mean(sbp_example$sbp.before, na.rm = TRUE),
                    diabetes = median(sbp_example$diabetes, na.rm = TRUE),
                    treat = "A"))
```

5.9 Doing Single Imputation with `simputation`

Single imputation is a potentially appropriate method when missingness can be assumed to be either completely at random (MCAR) or dependent only on observed predictors (MAR). We'll use the `simputation` package to accomplish it.

- The `simputation` vignette is available at https://cran.r-project.org/web/packages/simputation/vignettes/intro.html
- The `simputation` reference manual is available at https://cran.r-project.org/web/packages/simputation/simputation.pdf

5.9.1 Mirroring Our Prior Approach (imputing means/medians/modes)

Suppose we want to mirror what we did above, simply impute the mean for `sbp.before` and the median for `diabetes` again.

```
si.3 <- sbp_example %>%
    impute_lm(sbp.before ~ 1) %>%
    impute_median(diabetes ~ 1) %>%
    replace_na(list(treat = "A"))

si.3
```

```
# A tibble: 15 x 5
   subject treat diabetes sbp.before sbp.after
 *   <int> <fct>    <dbl>      <dbl>     <dbl>
 1     101 A         1.00        120       105
 2     102 B         0           145       135
 3     103 C         0           150       150
 4     104 A         1.00        143       120
 5     105 C         1.00        155       135
 6     106 A         1.00        143       115
 7     107 A         0           135       160
 8     108 A         1.00        143       150
 9     109 B         1.00        115       130
10     110 C         1.00        170       155
11     111 A         0           150       140
12     112 B         0           145       140
13     113 C         1.00        140       150
14     114 A         1.00        160       135
15     115 B         1.00        135       120
```

5.9.2 Using a model to impute `sbp.before` and `diabetes`

Suppose we wanted to use:

- a robust linear model to predict `sbp.before` missing values, on the basis of `sbp.after` and `diabetes` status, and
- a predictive mean matching approach to predict `diabetes` status, on the basis of `sbp.after`, and
- a decision tree approach to predict `treat` status, using all other variables in the data

```
set.seed(50001)

imp.4 <- sbp_example %>%
    impute_rlm(sbp.before ~ sbp.after + diabetes) %>%
    impute_pmm(diabetes ~ sbp.after) %>%
    impute_cart(treat ~ .)

imp.4
```

```
# A tibble: 15 x 5
```

```
   subject treat diabetes sbp.before sbp.after
 *   <int> <fct>    <dbl>      <dbl>     <dbl>
 1     101 A         1.00        120       105
 2     102 B         0           145       135
 3     103 C         0           150       150
 4     104 A         1.00        139       120
 5     105 C         1.00        155       135
 6     106 A         1.00        136       115
 7     107 A         0           135       160
 8     108 A         1.00        155       150
 9     109 B         1.00        115       130
10     110 C         1.00        170       155
11     111 A         0           150       140
12     112 B         0           145       140
13     113 C         1.00        140       150
14     114 A         1.00        160       135
15     115 B         1.00        135       120
```

Details on the many available methods in `simputation` are provided in its manual. These include:

- `impute_cart` uses a Classification and Regression Tree approach for numerical or categorical data. There is also an `impute_rf` command which uses Random Forests for imputation.
- `impute_pmm` is one of several "hot deck" options for imputation, this one is predictive mean matching, which can be used with numeric data (only). Missing values are first imputed using a predictive model. Next, these predictions are replaced with the observed values which are nearest to the prediction. Other imputation options in this group include random hot deck, sequential hot deck and k-nearest neighbor imputation.
- `impute_rlm` is one of several regression imputation methods, including linear models, robust linear models (which use what is called M-estimation to impute numerical variables) and lasso/elastic net/ridge regression models.

The `simputation` package can also do EM-based multivariate imputation, and multivariate random forest imputation, and several other approaches.

Chapter 6

A Study of Prostate Cancer

6.1 Data Load and Background

The data in `prost.csv` is derived from Stamey et al. (1989) who examined the relationship between the level of prostate-specific antigen and a number of clinical measures in 97 men who were about to receive a radical prostatectomy. The `prost` data, as I'll name it in R, contains 97 rows and 11 columns.

```
prost
```

```
# A tibble: 97 x 10
   subject   lpsa lcavol lweight   age bph      svi   lcp gleason pgg45
     <int>  <dbl>  <dbl>   <dbl> <int> <fct>  <int> <dbl> <fct>   <int>
 1       1 -0.431 -0.580    2.77    50 Low        0 -1.39 6           0
 2       2 -0.163 -0.994    3.32    58 Low        0 -1.39 6           0
 3       3 -0.163 -0.511    2.69    74 Low        0 -1.39 7          20
 4       4 -0.163 -1.20     3.28    58 Low        0 -1.39 6           0
 5       5  0.372  0.751    3.43    62 Low        0 -1.39 6           0
 6       6  0.765 -1.05     3.23    50 Low        0 -1.39 6           0
 7       7  0.765  0.737    3.47    64 Medium     0 -1.39 6           0
 8       8  0.854  0.693    3.54    58 High       0 -1.39 6           0
 9       9  1.05  -0.777    3.54    47 Low        0 -1.39 6           0
10      10  1.05   0.223    3.24    63 Low        0 -1.39 6           0
# ... with 87 more rows
```

Note that a related `prost` data frame is also available as part of several R packages, including the `faraway` package, but there is an error in the `lweight` data for subject 32 in those presentations. The value of `lweight` for subject 32 should not be 6.1, corresponding to a prostate that is 449 grams in size, but instead the `lweight` value should be 3.804438, corresponding to a 44.9 gram prostate[1].

I've also changed the `gleason` and `bph` variables from their presentation in other settings, to let me teach some additional details.

6.2 Code Book

Variable	Description
`subject`	subject number (1 to 97)

[1] https://statweb.stanford.edu/~tibs/ElemStatLearn/ attributes the correction to Professor Stephen W. Link.

Variable	Description
`lpsa`	log(prostate specific antigen in ng/ml), our **outcome**
`lcavol`	log(cancer volume in cm^3)
`lweight`	log(prostate weight, in g)
`age`	age
`bph`	benign prostatic hyperplasia amount (Low, Medium, or High)
`svi`	seminal vesicle invasion (1 = yes, 0 = no)
`lcp`	log(capsular penetration, in cm)
`gleason`	combined Gleason score (6, 7, or > 7 here)
`pgg45`	percentage Gleason scores 4 or 5

Notes:

- in general, higher levels of PSA are stronger indicators of prostate cancer. An old standard (established almost exclusively with testing in white males, and definitely flawed) suggested that values below 4 were normal, and above 4 needed further testing. A PSA of 4 corresponds to an `lpsa` of 1.39.
- all logarithms are natural (base e) logarithms, obtained in R with the function `log()`
- all variables other than `subject` and `lpsa` are candidate predictors
- the `gleason` variable captures the highest combined Gleason score[^Scores range (in these data) from 6 (a well-differentiated, or low-grade cancer) to 9 (a high-grade cancer), although the maximum possible score is 10. 6 is the lowest score used for cancerous prostates. As this combination value increases, the rate at which the cancer grows and spreads should increase. This score refers to the combined Gleason grade, which is based on the sum of two areas (each scored 1-5) that make up most of the cancer.] in a biopsy, and higher scores indicate more aggressive cancer cells. It's stored here as 6, 7, or > 7.
- the `pgg45` variable captures the percentage of individual Gleason scores[^The 1-5 scale for individual biopsies are defined so that 1 indicates something that looks like normal prostate tissue, and 5 indicates that the cells and their growth patterns look very abnormal. In this study, the percentage of 4s and 5s shown in the data appears to be based on 5-20 individual scores in most subjects.] that are 4 or 5, on a 1-5 scale, where higher scores indicate more abnormal cells.

6.3 Additions for Later Use

The code below adds to the `prost` tibble:

- a factor version of the `svi` variable, called `svi_f`, with levels No and Yes,
- a factor version of `gleason` called `gleason_f`, with the levels ordered > 7, 7, and finally 6,
- a factor version of `bph` called `bph_f`, with levels ordered Low, Medium, High,
- a centered version of `lcavol` called `lcavol_c`,
- exponentiated `cavol` and `psa` results derived from the natural logarithms `lcavol` and `lpsa`.

```
prost <- prost %>%
    mutate(svi_f = fct_recode(factor(svi), "No" = "0", "Yes" = "1"),
           gleason_f = fct_relevel(gleason, c("> 7", "7", "6")),
           bph_f = fct_relevel(bph, c("Low", "Medium", "High")),
           lcavol_c = lcavol - mean(lcavol),
           cavol = exp(lcavol),
           psa = exp(lpsa))

glimpse(prost)
```

```
Observations: 97
Variables: 16
$ subject   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1...
```

```
$ lpsa      <dbl> -0.4307829, -0.1625189, -0.1625189, -0.1625189, 0.37...
$ lcavol    <dbl> -0.5798185, -0.9942523, -0.5108256, -1.2039728, 0.75...
$ lweight   <dbl> 2.769459, 3.319626, 2.691243, 3.282789, 3.432373, 3....
$ age       <int> 50, 58, 74, 58, 62, 50, 64, 58, 47, 63, 65, 63, 63, ...
$ bph       <fct> Low, Low, Low, Low, Low, Low, Medium, High, Low, Low...
$ svi       <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
$ lcp       <dbl> -1.3862944, -1.3862944, -1.3862944, -1.3862944, -1.3...
$ gleason   <fct> 6, 6, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 6, 7, 6...
$ pgg45     <int> 0, 0, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 30, 5, 5, 0, 30...
$ svi_f     <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, ...
$ gleason_f <fct> 6, 6, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 6, 7, 6...
$ bph_f     <fct> Low, Low, Low, Low, Low, Low, Medium, High, Low, Low...
$ lcavol_c  <dbl> -1.9298281, -2.3442619, -1.8608352, -2.5539824, -0.5...
$ cavol     <dbl> 0.56, 0.37, 0.60, 0.30, 2.12, 0.35, 2.09, 2.00, 0.46...
$ psa       <dbl> 0.65, 0.85, 0.85, 0.85, 1.45, 2.15, 2.15, 2.35, 2.85...
```

6.4 Fitting and Evaluating a Two-Predictor Model

To begin, let's use two predictors (`lcavol` and `svi`) and their interaction in a linear regression model that predicts `lpsa`. I'll call this model `c5_prost_A`

Earlier, we centered the `lcavol` values to facilitate interpretation of the terms. I'll use that centered version (called `lcavol_c`) of the quantitative predictor, and the 1/0 version of the `svi` variable[^We could certainly use the factor version of `svi` here, but it won't change the model in any meaningful way. There's no distinction in model *fitting* via `lm` between a 0/1 numeric variable and a No/Yes factor variable. The factor version of this information will be useful elsewhere, for instance in plotting the model.].

```
c5_prost_A <- lm(lpsa ~ lcavol_c * svi, data = prost)
summary(c5_prost_A)
```

```

Call:
lm(formula = lpsa ~ lcavol_c * svi, data = prost)

Residuals:
    Min      1Q  Median      3Q     Max
-1.6305 -0.5007  0.1266  0.4886  1.6847

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.33134    0.09128  25.540  < 2e-16 ***
lcavol_c      0.58640    0.08207   7.145 1.98e-10 ***
svi           0.60132    0.35833   1.678   0.0967 .
lcavol_c:svi  0.06479    0.26614   0.243   0.8082
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7595 on 93 degrees of freedom
Multiple R-squared:  0.5806,    Adjusted R-squared:  0.5671
F-statistic: 42.92 on 3 and 93 DF,  p-value: < 2.2e-16
```

6.4.1 Using `tidy`

It can be very useful to build a data frame of the model's results. We can use the `tidy` function in the `broom` package to do so.

```
tidy(c5_prost_A)
```

```
          term   estimate  std.error  statistic      p.value
1  (Intercept) 2.33134409 0.09128253 25.5398727 8.246849e-44
2    lcavol_c 0.58639599 0.08206929  7.1451331 1.981492e-10
3          svi 0.60131973 0.35832695  1.6781314 9.667899e-02
4 lcavol_c:svi 0.06479298 0.26614194  0.2434527 8.081909e-01
```

This makes it much easier to pull out individual elements of the model fit.

For example, to specify the coefficient for `svi`, rounded to three decimal places, I could use `tidy(c5_prost_A) %>% filter(term == "svi") %>% select(estimate) %>% round(., 3)`

- The result is 0.601.
- If you look at the Markdown file, you'll see that the number shown in the bullet point above this one was generated using inline R code, and the function specified above.

6.4.2 Interpretation

1. The intercept, 2.33, for the model is the predicted value of `lpsa` when `lcavol` is at its average and there is no seminal vesicle invasion (e.g. `svi` = 0).
2. The coefficient for `lcavol_c`, 0.59, is the predicted change in `lpsa` associated with a one unit increase in `lcavol` (or `lcavol_c`) when there is no seminal vesicle invasion.
3. The coefficient for `svi`, 0.60, is the predicted change in `lpsa` associated with having no `svi` to having an `svi` while the `lcavol` remains at its average.
4. The coefficient for `lcavol_c:svi`, the product term, which is 0.06, is the difference in the slope of `lcavol_c` for a subject with `svi` as compared to one with no `svi`.

Note: If you look at the R Markdown, you'll notice that in bullet point 3, I didn't use `round` to round off the estimate (as I did in the other three bullets), but instead a special function I specified at the start of the R Markdown file called `specify_decimal()` which uses the `format` function. This forces, in this case, the trailing zero in the two decimal representation of the `svi` coefficient to be shown. The special function, again, is:

```
specify_decimal <- function(x, k) format(round(x, k), nsmall=k)
```

6.5 Exploring Model `c5_prost_A`

The `glance` function from the `broom` package builds a nice one-row summary for the model.

```
glance(c5_prost_A)
```

```
  r.squared adj.r.squared     sigma statistic      p.value df    logLik
1 0.5806435     0.5671158 0.7594785  42.92278 1.678836e-17  4 -108.9077
       AIC      BIC deviance df.residual
1 227.8153 240.6889 53.64311          93
```

This summary includes, in order,

- the model R^2, adjusted R^2 and $\hat{\sigma}$, the residual standard deviation,
- the ANOVA F statistic and associated p value,
- the number of degrees of freedom used by the model, and its log-likelihood ratio
- the model's AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion)

- the model's deviance statistic and residual degrees of freedom

6.5.1 summary for Model c5_prost_A

If necessary, we can also run `summary` on this `c5_prost_A` object to pick up some additional summaries. Since the `svi` variable is binary, the interaction term is, too, so the t test here and the F test in the ANOVA yield the same result.

```
summary(c5_prost_A)
```

```
Call:
lm(formula = lpsa ~ lcavol_c * svi, data = prost)

Residuals:
    Min      1Q  Median      3Q     Max
-1.6305 -0.5007  0.1266  0.4886  1.6847

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.33134    0.09128  25.540  < 2e-16 ***
lcavol_c      0.58640    0.08207   7.145 1.98e-10 ***
svi           0.60132    0.35833   1.678   0.0967 .
lcavol_c:svi  0.06479    0.26614   0.243   0.8082
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7595 on 93 degrees of freedom
Multiple R-squared:  0.5806,    Adjusted R-squared:  0.5671
F-statistic: 42.92 on 3 and 93 DF,  p-value: < 2.2e-16
```

If you've forgotten the details of the pieces of this summary, review the Part C Notes from 431.

6.5.2 Adjusted R^2

R^2 is greedy.

- R^2 will always suggest that we make our models as big as possible, often including variables of dubious predictive value.
- As a result, there are various methods for penalizing R^2 so that we wind up with smaller models.
- The **adjusted R^2** is often a useful way to compare multiple models for the same response.
  - $R^2_{adj} = 1 - \frac{(1-R^2)(n-1)}{n-k}$, where n = the number of observations and k is the number of coefficients estimated by the regression (including the intercept and any slopes).
  - So, in this case, $R^2_{adj} = 1 - \frac{(1-0.5806)(97-1)}{97-4} = 0.5671$
  - The adjusted R^2 value is not, technically, a proportion of anything, but it is comparable across models for the same outcome.
  - The adjusted R^2 will always be less than the (unadjusted) R^2.

6.5.3 Coefficient Confidence Intervals

Here are the 90% confidence intervals for the coefficients in Model A. Adjust the `level` to get different intervals.

```
confint(c5_prost_A, level = 0.90)
```

```
                    5 %      95 %
(Intercept)   2.17968697 2.4830012
lcavol_c      0.45004577 0.7227462
svi           0.00599401 1.1966454
lcavol_c:svi -0.37737623 0.5069622
```

What can we conclude from this about the utility of the interaction term?

6.5.4 ANOVA for Model c5_prost_A

The interaction term appears unnecessary. We might wind up fitting the model without it. A complete ANOVA test is available, including a p value, if you want it.

```
anova(c5_prost_A)
```

```
Analysis of Variance Table

Response: lpsa
             Df Sum Sq Mean Sq  F value    Pr(>F)
lcavol_c      1 69.003  69.003 119.6289 < 2.2e-16 ***
svi           1  5.237   5.237   9.0801  0.003329 **
lcavol_c:svi  1  0.034   0.034   0.0593  0.808191
Residuals    93 53.643   0.577
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the `anova` approach for a `lm` object is sequential. The first row shows the impact of `lcavol_c` as compared to a model with no predictors (just an intercept). The second row shows the impact of adding `svi` to a model that already contains `lcavol_c`. The third row shows the impact of adding the interaction (product) term to the model with the two main effects. So the order in which the variables are added to the regression model matters for this ANOVA. The F tests here describe the incremental impact of each covariate in turn.

6.5.5 Residuals, Fitted Values and Standard Errors with augment

The `augment` function in the `broom` package builds a data frame including the data used in the model, along with predictions (fitted values), residuals and other useful information.

```
c5_prost_A_frame <- augment(c5_prost_A) %>% tbl_df
skim(c5_prost_A_frame)
```

```
Skim summary statistics
 n obs: 97
 n variables: 10

Variable type: integer
 variable missing complete  n mean   sd p0 p25 median p75 p100
      svi       0       97 97 0.22 0.41  0   0      0   0    1

Variable type: numeric
   variable missing complete  n     mean     sd       p0      p25 median
    .cooksd       0       97 97  0.011   0.02    6.9e-06  0.00078 0.0035
    .fitted       0       97 97  2.48    0.88    0.75     1.84    2.4
```

```
       .hat       0       97 97  0.041   0.041   0.013    0.016   0.025
     .resid       0       97 97 -6.9e-17 0.75   -1.63    -0.5     0.13
    .se.fit       0       97 97  0.14    0.061   0.087    0.095   0.12
     .sigma       0       97 97  0.76    0.0052  0.74     0.76    0.76
 .std.resid       0       97 97  0.0012  1.01   -2.19    -0.69    0.17
   lcavol_c       0       97 97  5.4e-17 1.18   -2.7     -0.84    0.097
       lpsa       0       97 97  2.48    1.15   -0.43     1.73    2.59
   p75 p100
 0.01  0.13
 3.07  4.54
 0.049 0.25
 0.49  1.68
 0.17  0.38
 0.76  0.76
 0.65  2.26
 0.78  2.47
 3.06  5.58
```

Elements shown here include:

- `.fitted` Fitted values of model (or predicted values)
- `.se.fit` Standard errors of fitted values
- `.resid` Residuals (observed - fitted values)
- `.hat` Diagonal of the hat matrix (these indicate *leverage* - points with high leverage indicate unusual combinations of predictors - values more than 2-3 times the mean leverage are worth some study - leverage is always between 0 and 1, and measures the amount by which the predicted value would change if the observation's y value was increased by one unit - a point with leverage 1 would cause the line to follow that point perfectly)
- `.sigma` Estimate of residual standard deviation when corresponding observation is dropped from model
- `.cooksd` Cook's distance, which helps identify influential points (values of Cook's d > 0.5 may be influential, values > 1.0 almost certainly are - an influential point changes the fit substantially when it is removed from the data)
- `.std.resid` Standardized residuals (values above 2 in absolute value are worth some study - treat these as normal deviates [Z scores], essentially)

See `?augment.lm` in R for more details.

6.5.6 Making Predictions with `c5_prost_A`

Suppose we want to predict the `lpsa` for a patient with cancer volume equal to this group's mean, for both a patient with and without seminal vesicle invasion, and in each case, we want to use a 90% prediction interval?

```
newdata <- data.frame(lcavol_c = c(0,0), svi = c(0,1))
predict(c5_prost_A, newdata, interval = "prediction", level = 0.90)
```

```
       fit      lwr      upr
1 2.331344 1.060462 3.602226
2 2.932664 1.545742 4.319586
```

Since the predicted value in `fit` refers to the natural logarithm of PSA, to make the predictions in terms of PSA, we would need to exponentiate. The code below will accomplish that task.

```
pred <- predict(c5_prost_A, newdata, interval = "prediction", level = 0.90)
exp(pred)
```

```
       fit      lwr      upr
```

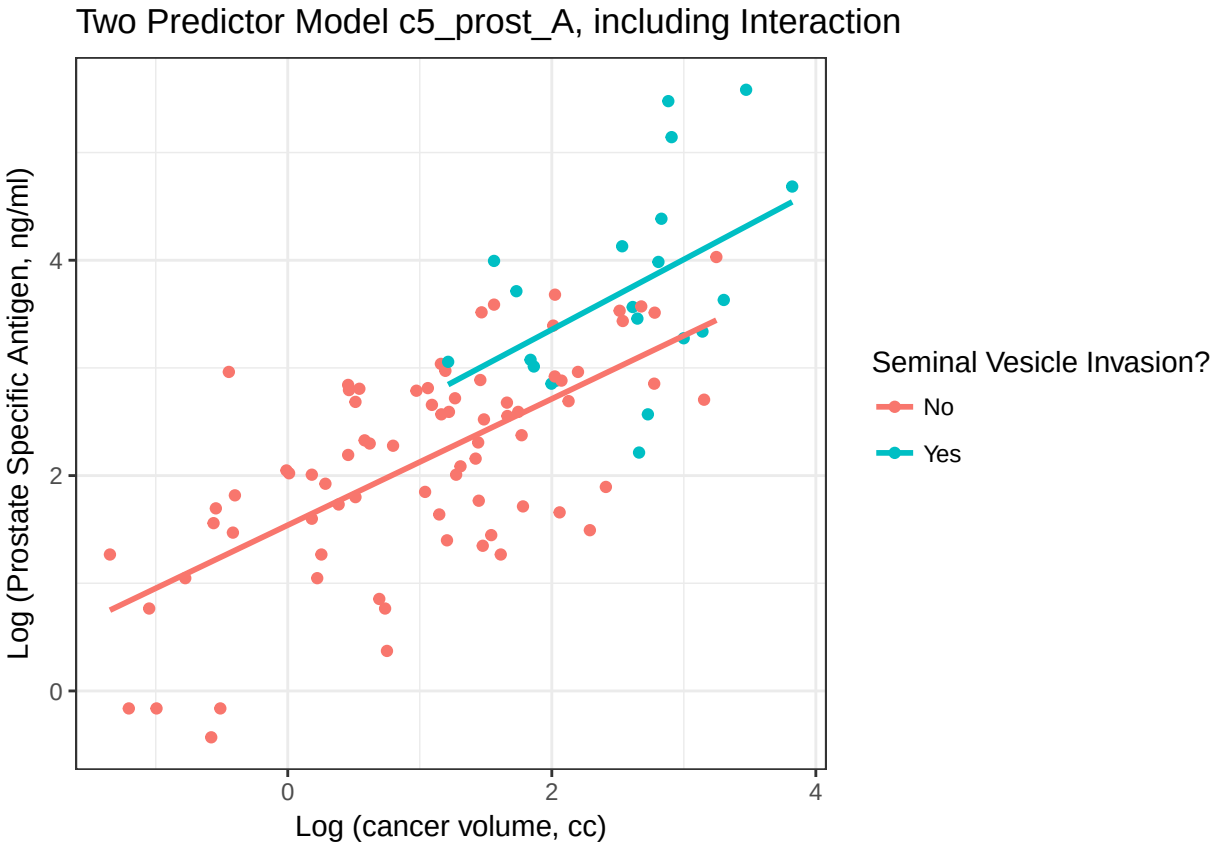
```
1 10.29177 2.887706 36.67978
2 18.77758 4.691450 75.15750
```

6.6 Plotting Model c5_prost_A

6.6.0.1 Plot logs conventionally

Here, we'll use ggplot2 to plot the logarithms of the variables as they came to us, on a conventional coordinate scale. Note that the lines are nearly parallel. What does this suggest about our Model A?

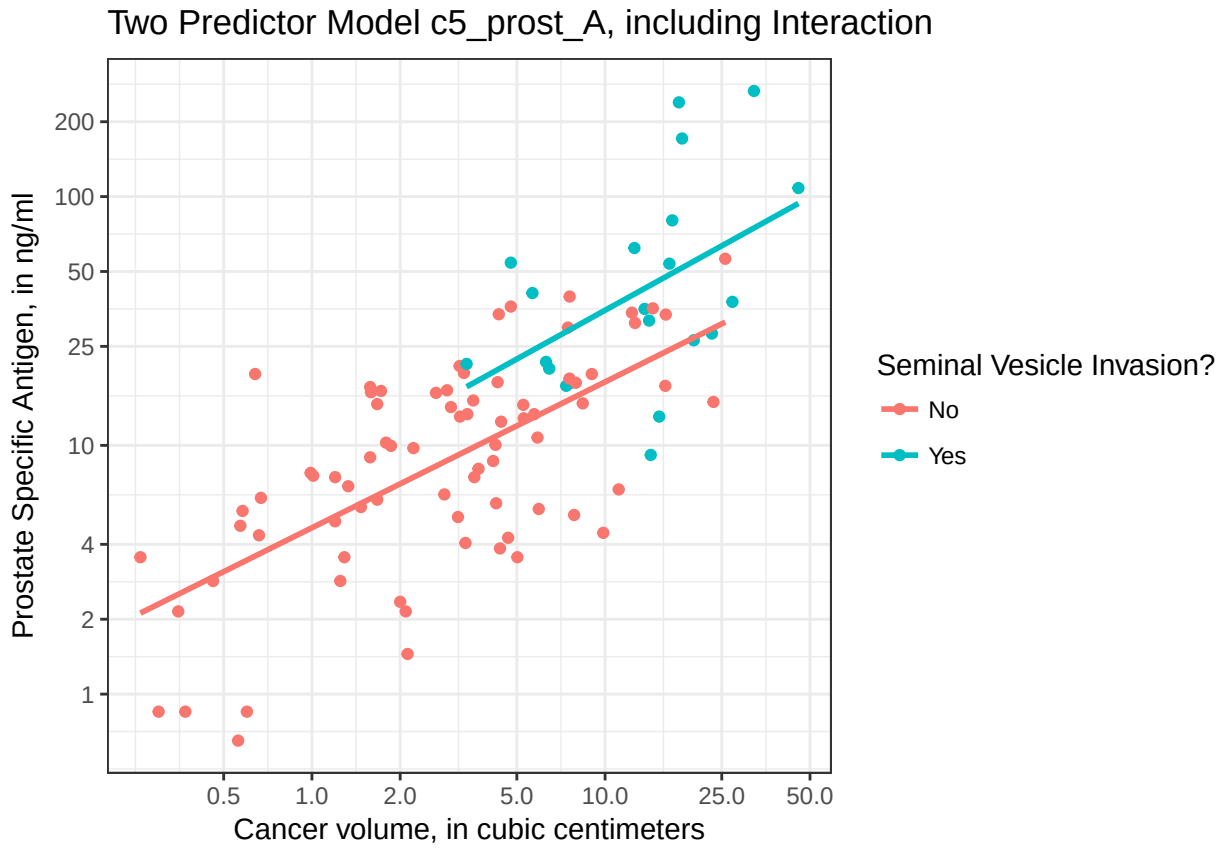
```
ggplot(prost, aes(x = lcavol, y = lpsa, group = svi_f, color = svi_f)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
    scale_color_discrete(name = "Seminal Vesicle Invasion?") +
    theme_bw() +
    labs(x = "Log (cancer volume, cc)",
         y = "Log (Prostate Specific Antigen, ng/ml)",
         title = "Two Predictor Model c5_prost_A, including Interaction")
```



6.6.0.2 Plot on log-log scale

Another approach (which might be easier in some settings) would be to plot the raw values of Cancer Volume and PSA, but use logarithmic axes, again using the natural (base e) logarithm, as follows. If we use the default choice with 'trans = "log", we'll find a need to select some useful break points for the grid, as I've done in what follows.

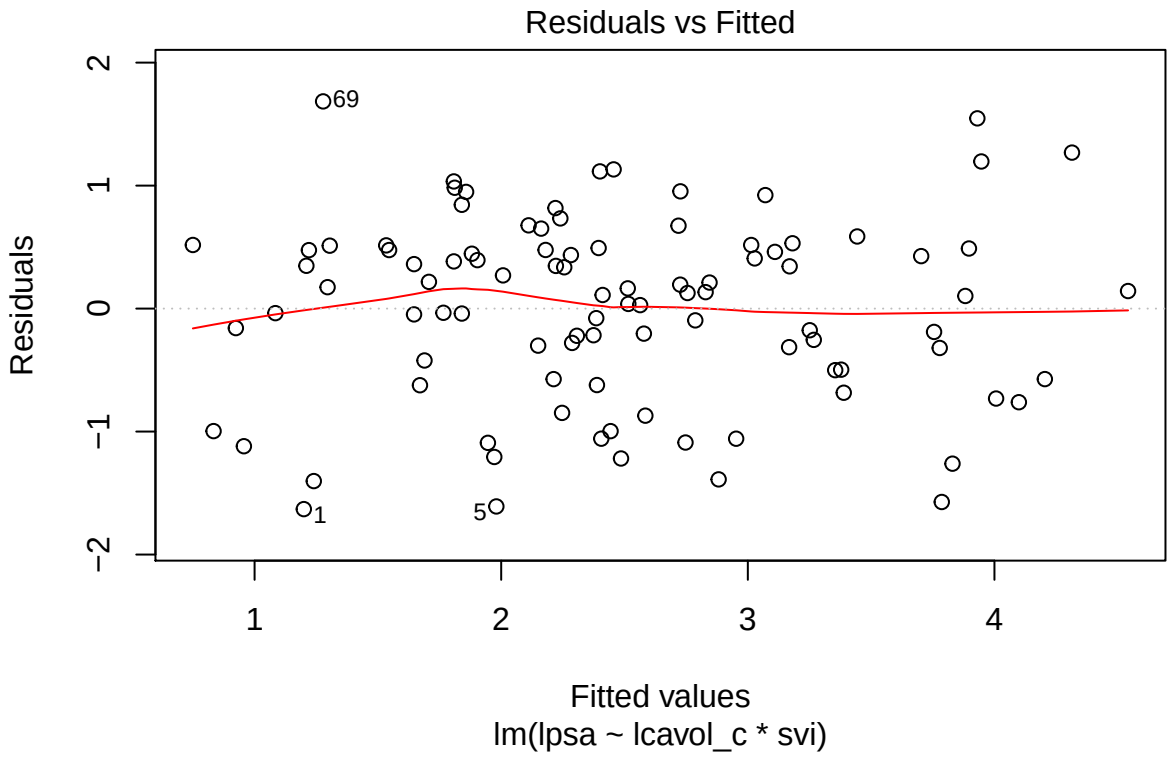
```
ggplot(prost, aes(x = cavol, y = psa, group = svi_f, color = svi_f)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
    scale_color_discrete(name = "Seminal Vesicle Invasion?") +
    scale_x_continuous(trans = "log",
                       breaks = c(0.5, 1, 2, 5, 10, 25, 50)) +
    scale_y_continuous(trans = "log",
                       breaks = c(1, 2, 4, 10, 25, 50, 100, 200)) +
    theme_bw() +
    labs(x = "Cancer volume, in cubic centimeters",
         y = "Prostate Specific Antigen, in ng/ml",
         title = "Two Predictor Model c5_prost_A, including Interaction")
```



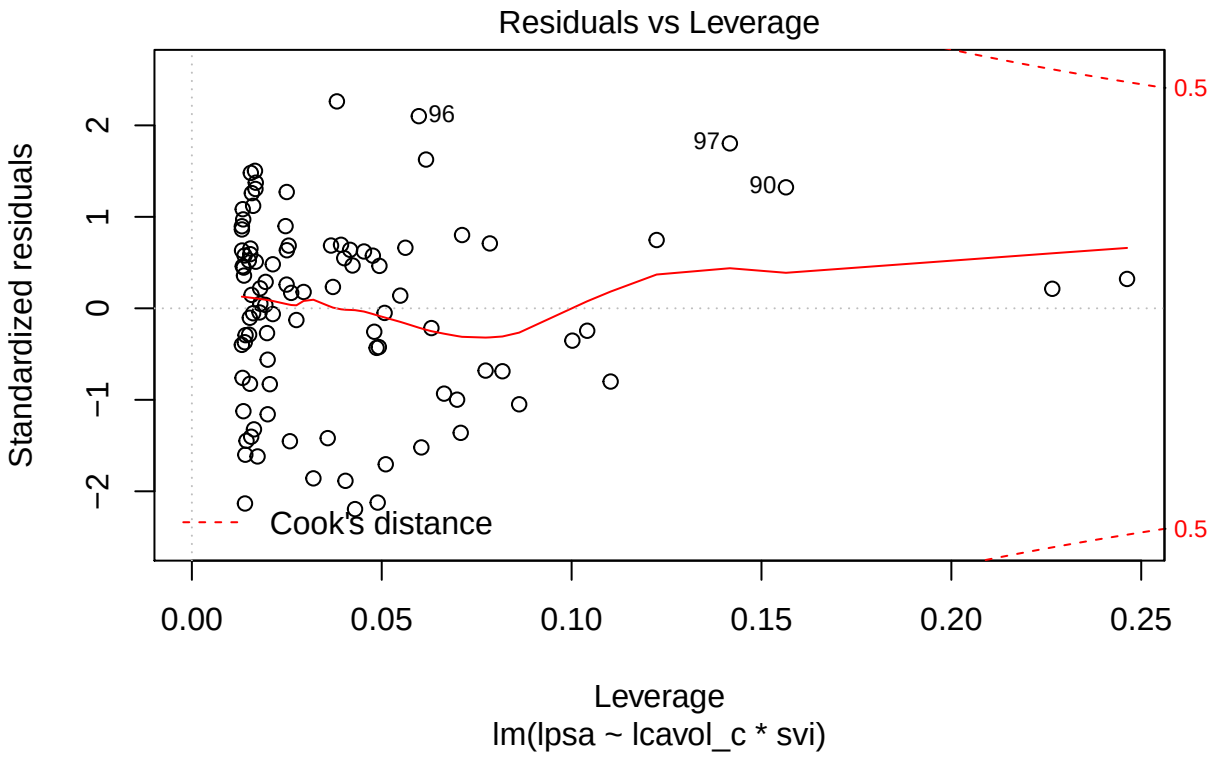
I've used the break point of 4 on the Y axis because of the old rule suggesting further testing for asymptomatic men with PSA of 4 or higher, but the other break points are arbitrary - they seemed to work for me, and used round numbers.

6.6.1 Residual Plots of c5_prost_A

```
plot(c5_prost_A, which = 1)
```




```
plot(c5_prost_A, which = 5)
```



6.7 Cross-Validation of Model `c5_prost_A`

Suppose we want to evaluate whether our model `c5_prost_A` predicts effectively in new data.

One approach (used, for instance, in 431) would be to split our sample into a separate training (perhaps 70% of the data) and test (perhaps 30% of the data) samples, and then:

- 1. fit the model in the training sample,
- 2. use the resulting model to make predictions for `lpsa` in the test sample, and
- 3. evaluate the quality of those predictions, perhaps by comparing the results to what we'd get using a different model.

One problem with this approach is that with a small data set like this, we may be reluctant to cut our sample size for the training or the testing down because we're afraid that our model building and testing will be hampered by a small sample size. A potential solution is the idea of **cross-validation**, which involves partitioning our data into a series of training-test subsets, multiple times, and then combining the results.

The rest of this section is built on some material by David Robinson at https://rpubs.com/dgrtwo/cv-modelr.

Suppose that we want to perform what is called *10-crossfold separation*. In words, this approach splits the 97 observations in our `prost` data frame into 10 exclusive partitions of about 90% (so about 87-88 observations) into a training sample, and the remaining 10% (9-10 observations) in a test sample[2]. We then refit a model of interest using the training data, and fit the resulting model on the test data using the `broom` package's `augment` function. This process is then repeated (a total of 10 times) so that each observation is used 9 times in the training sample, and once in the test sample.

[2]If we did 5-crossfold validation, we'd have 5 partitions into samples of 80% training and 20% test samples.

To code this in R, we'll make use of a few new ideas. Our goal will be to cross-validate model `c5_prost_A`, which, you'll recall, uses `lcavol_c`, `svi` and their interaction, to predict `lpsa` in the `prost` data.

1. First, we set a seed for the validation algorithm, so we can replicate our results down the line.
2. Then we use the `crossv_kfold` function from the `modelr` package to split the `prost` data into ten different partitions, and then use each partition for a split into training and test samples, which the machine indexes with `train` and `test`.
3. Then we use some magic and the `map` function from the `purrr` package (part of the core `tidyverse`) to fit a new `lm(lpsa ~ lcavol_c * svi)` model to each of the training samples generated by `crossv_kfold`.
4. Finally, some additional magic with the `unnest` and `map2` functions applies each of these new models to the appropriate test sample, and generate predictions (`.fitted`) and standard errors for each prediction (`.se.fit`).

```
set.seed(4320308)

prost_models <- prost %>%
    crossv_kfold(k = 10) %>%
    mutate(model = map(train, ~ lm(lpsa ~ lcavol_c * svi, data = .)))

prost_predictions <- prost_models %>%
    unnest(map2(model, test, ~ augment(.x, newdata = .y)))

head(prost_predictions)
```

```
# A tibble: 6 x 19
  .id   subject   lpsa   lcavol lweight   age bph      svi    lcp gleason
  <chr>   <int>  <dbl>    <dbl>   <dbl> <int> <fct>  <int>  <dbl> <fct>
1 01          3 -0.163 -0.511      2.69    74 Low        0 -1.39  7
2 01         12  1.27  -1.35       3.60    63 Medium     0 -1.39  6
3 01         16  1.45   1.54       3.06    66 Low        0 -1.39  6
4 01         18  1.49   2.29       3.65    66 Low        0  0.372 6
5 01         30  1.89   2.41       3.38    65 Low        0  1.62  6
6 01         34  2.02   0.00995    3.27    54 Low        0 -1.39  6
# ... with 9 more variables: pgg45 <int>, svi_f <fct>, gleason_f <fct>,
#   bph_f <fct>, lcavol_c <dbl>, cavol <dbl>, psa <dbl>, .fitted <dbl>,
#   .se.fit <dbl>
```

The results are a set of predictions based on the splits into training and test groups (remember there are 10 such splits, indexed by `.id`) that describe the complete set of 97 subjects again.

6.7.1 Cross-Validated Summaries of Prediction Quality

Now, we can calculate the root Mean Squared Prediction Error (RMSE) and Mean Absolute Prediction Error (MAE) for this modeling approach (using `lcavol_c` and `svi` to predict `lpsa`) across these observations.

```
prost_predictions %>%
    summarize(RMSE_ourmodel = sqrt(mean((lpsa - .fitted) ^2)),
              MAE_ourmodel = mean(abs(lpsa - .fitted)))
```

```
# A tibble: 1 x 2
  RMSE_ourmodel MAE_ourmodel
          <dbl>        <dbl>
1         0.783        0.638
```

For now, we'll compare our model to the "intercept only" model that simply predicts the mean `lpsa` across all patients.

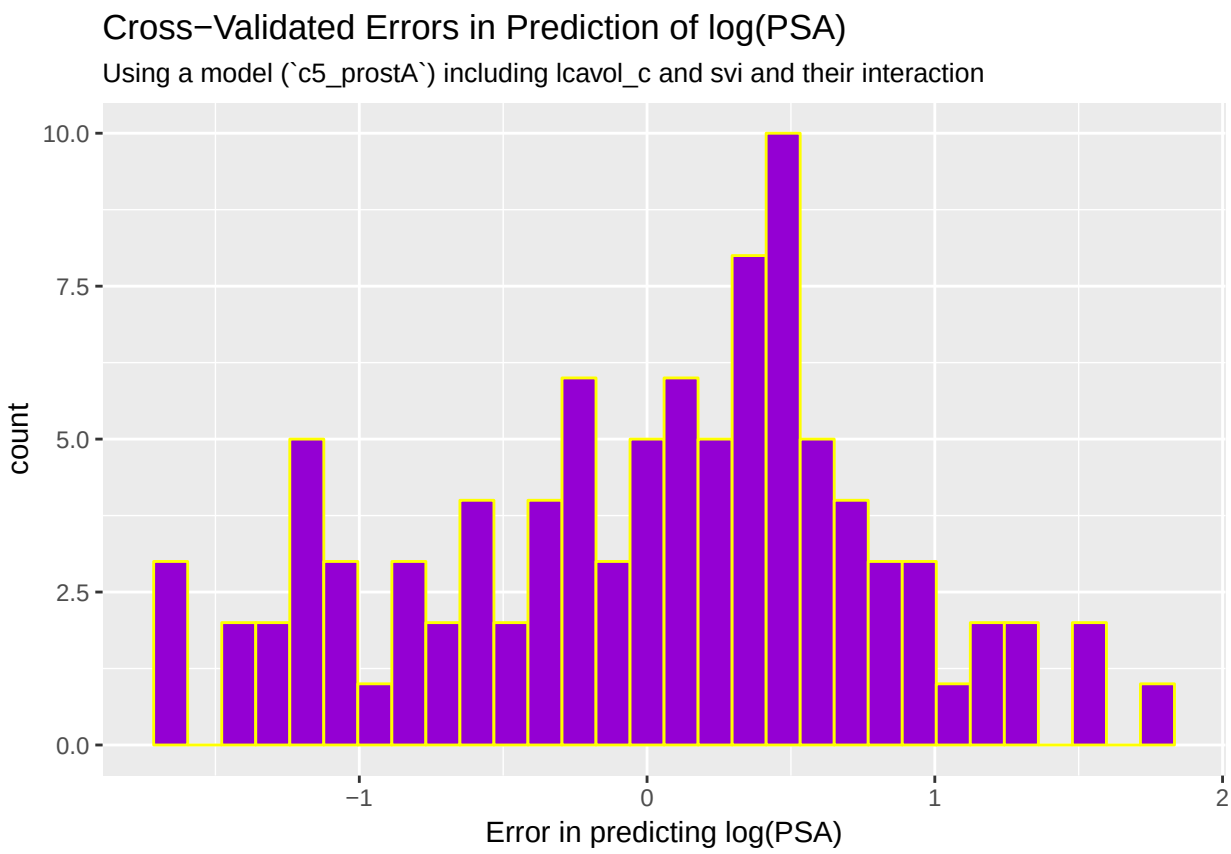
```
prost_predictions %>%
    summarize(RMSE_intercept = sqrt(mean((lpsa - mean(lpsa)) ^2)),
              MAE_intercept = mean(abs(lpsa - mean(lpsa))))
```

```
# A tibble: 1 x 2
  RMSE_intercept MAE_intercept
           <dbl>         <dbl>
1           1.15         0.891
```

So our model looks meaningfully better than the "intercept only" model, in that both the RMSE and MAE are much lower (better) with our model.

Another thing we could do with this tibble of predictions we have created is to graph the size of the prediction errors (observed `lpsa` minus predicted values in `.fitted`) that our modeling approach makes.

```
prost_predictions %>%
    mutate(errors = lpsa - .fitted) %>%
    ggplot(., aes(x = errors)) +
    geom_histogram(bins = 30, fill = "darkviolet", col = "yellow") +
    labs(title = "Cross-Validated Errors in Prediction of log(PSA)",
         subtitle = "Using a model (`c5_prostA`) including lcavol_c and svi and their interaction",
         x = "Error in predicting log(PSA)")
```



This suggests that some of our results are off by quite a bit, on the log(PSA) scale, which is summarized for the original data below.

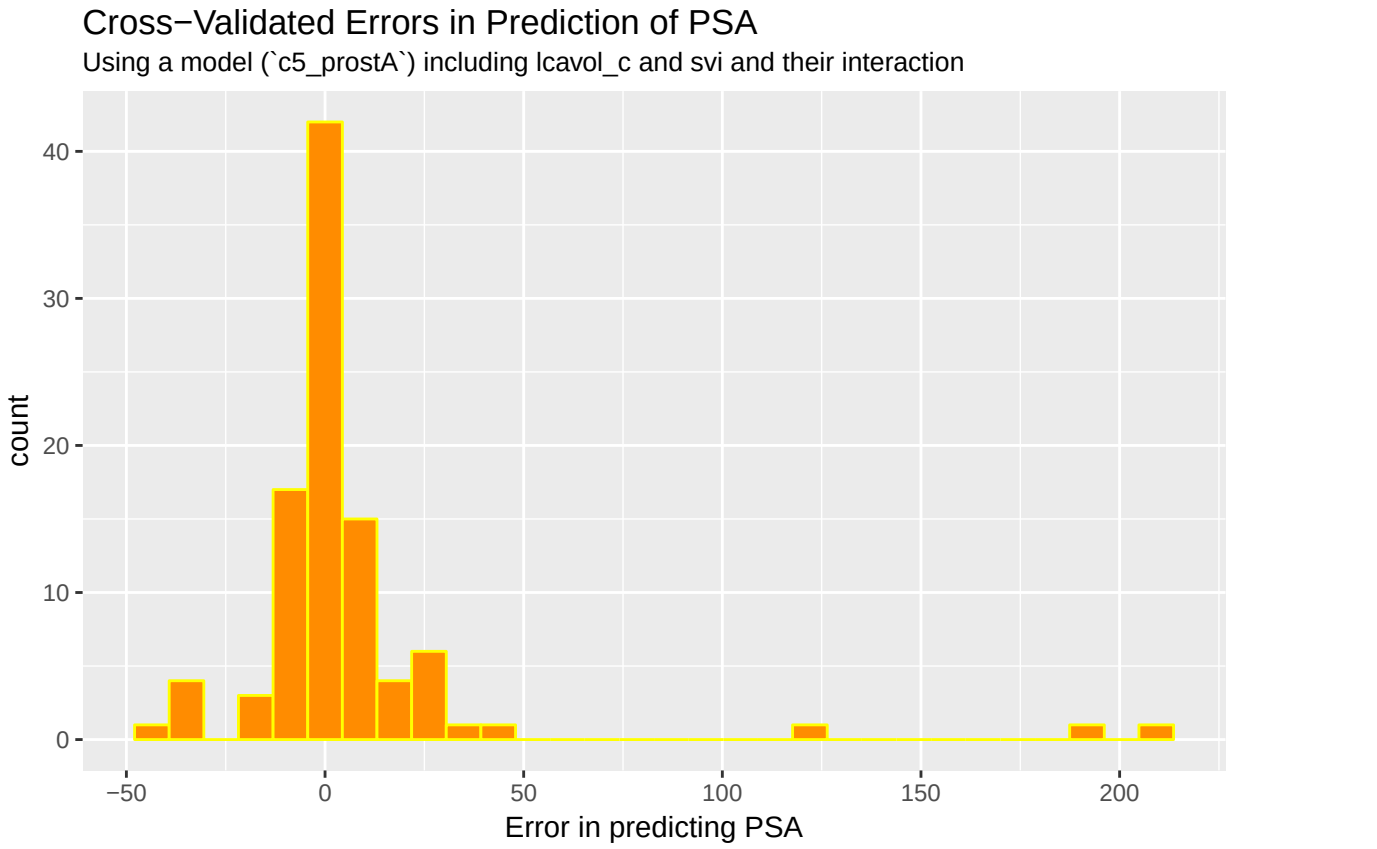
```
prost %>% skim(lpsa)
```

```
Skim summary statistics
```

```
 n obs: 97
 n variables: 16

Variable type: numeric
 variable missing complete  n mean   sd    p0  p25 median  p75 p100
     lpsa       0       97 97 2.48 1.15 -0.43 1.73   2.59 3.06 5.58
```

If we like, we could transform the predictions and observed values back to the scale of PSA (unlogged) and then calculate and display errors, as follows:

```
prost_predictions %>%
    mutate(err.psa = exp(lpsa) - exp(.fitted)) %>%
    ggplot(., aes(x = err.psa)) +
    geom_histogram(bins = 30, fill = "darkorange", col = "yellow") +
    labs(title = "Cross-Validated Errors in Prediction of PSA",
         subtitle = "Using a model (`c5_prostA`) including lcavol_c and svi and their interaction",
         x = "Error in predicting PSA")
```



This suggests that some of our results are off by quite a bit, on the original scale of PSA, which is summarized below.

```
prost %>% mutate(psa = exp(lpsa)) %>% skim(psa)
```

```
Skim summary statistics
 n obs: 97
 n variables: 16

Variable type: numeric
 variable missing complete  n  mean    sd   p0  p25 median   p75   p100
```

```
psa       0       97 97 23.74 40.83 0.65 5.65  13.35 21.25 265.85
```

We'll return to the notion of cross-validation again, but for now, let's consider the problem of considering adding more predictors to our model, and then making sensible selections as to which predictors actually should be incorporated.

Chapter 7

Stepwise Variable Selection

7.1 Strategy for Model Selection

Ramsey and Schafer (2002) suggest a strategy for dealing with many potential explanatory variables should include the following elements:

1. Identify the key objectives.
2. Screen the available variables, deciding on a list that is sensitive to the objectives and excludes obvious redundancies.
3. Perform exploratory analysis, examining graphical displays and correlation coefficients.
4. Perform transformations, as necessary.
5. Examine a residual plot after fitting a rich model, performing further transformations and considering outliers.
6. Find a suitable subset of the predictors, exerting enough control over any semi-automated selection procedure to be sensitive to the questions of interest.
7. Proceed with the analysis, using the selected explanatory variables.

The Two Key Aspects of Model Selection are:

1. Evaluating each potential subset of predictor variables
2. Deciding on the collection of potential subsets

7.1.1 How Do We Choose Potential Subsets of Predictors?

Choosing potential subsets of predictor variables usually involves either:

1. Stepwise approaches
2. All possible subset (or best possible subset) searches

Note that the use of any variable selection procedure changes the properties of ...

- the estimated coefficients, which are biased, and
- the associated tests and confidence intervals, which are overly optimistic.

Leeb and Potscher (2005) summarize the key issues:

1. Regardless of sample size, the model selection step typically has a dramatic effect on the sampling properties of the estimators that cannot be ignored. In particular, the sampling properties of post-model-selection estimators are typically significantly different from the nominal distributions that arise if a fixed model is supposed.

2. As a consequence, use of inference procedures that do not take into account the model selection step (e.g. using standard t-intervals as if the selected model has been given prior to the statistical analysis) can be highly misleading.

7.2 A "Kitchen Sink" Model (Model `c5_prost_ks`)

Suppose that we now consider a model for the `prost` data we have been working with, which includes main effects (and, in this case, only the main effects) of all eight candidate predictors for `lpsa`, as follows.

```
c5_prost_ks <- lm(lpsa ~ lcavol + lweight + age + bph_f + svi_f +
                  lcp + gleason_f + pgg45, data = prost)

tidy(c5_prost_ks)
```

```
          term     estimate   std.error  statistic      p.value
1  (Intercept)  0.169937821 0.931332512  0.1824674 8.556454e-01
2       lcavol  0.544313829 0.087979210  6.1868461 2.010505e-08
3      lweight  0.702237531 0.203013089  3.4590751 8.455164e-04
4          age -0.023857982 0.011081414 -2.1529727 3.412099e-02
5  bph_fMedium  0.364036274 0.182575941  1.9938896 4.933267e-02
6    bph_fHigh  0.248789989 0.195975792  1.2694935 2.076898e-01
7     svi_fYes  0.710949408 0.241990241  2.9379259 4.240326e-03
8          lcp -0.119311781 0.089458946 -1.3337043 1.858223e-01
9   gleason_f7  0.220746268 0.343065609  0.6434520 5.216430e-01
10  gleason_f6 -0.053096704 0.430098039 -0.1234526 9.020368e-01
11       pgg45  0.003984574 0.004146495  0.9609499 3.392714e-01
```

```
glance(c5_prost_ks)
```

```
  r.squared adj.r.squared     sigma statistic      p.value df    logLik
1 0.6790343     0.6417127 0.6909479  18.19414 2.373796e-17 11 -95.93939
       AIC      BIC deviance df.residual
1 215.8788 246.7753 41.05718          86
```

We'll often refer to this (all predictors on board) approach as a "kitchen sink" model[This refers to the English idiom "… everything but the kitchen sink" which describes, essentially, everything imaginable. A "kitchen sink regression" is often used as a pejorative term, since no special skill or insight is required to identify it, given a list of potential predictors. For more, yes, there is a Wikipedia page.].

7.3 Sequential Variable Selection: Stepwise Approaches

- Forward Selection
  - We begin with a constant mean and then add potential predictors one at a time according to some criterion (R defaults to minimizing the Akaike Information Criterion) until no further addition significantly improves the fit.
  - Each categorical factor variable is represented in the regression model as a set of indicator variables. In the absence of a good reason to do something else, the set is added to the model as a single unit, and R does this automatically.
- Backwards Elimination
  - Start with the "kitchen sink" model and then delete potential predictors one at a time.
  - Backwards Elimination is less likely than Forward Selection, to omit negatively confounded sets of variables, though all stepwise procedures have problems.
- Stepwise Regression can also be done by combining these methods.

7.3.1 The Big Problems with Stepwise Regression

There is no reason to assume that a single best model can be found.

- The use of forward selection, or backwards elimination, or stepwise regression including both procedures, will NOT always find the same model.
- It also appears to be essentially useless to try different stepwise methods to look for agreement.

Users of stepwise regression frequently place all of their attention on the particular explanatory variables included in the resulting model, when there's **no reason** (in most cases) to assume that model is in any way optimal.

Despite all of its problems, let's use stepwise regression to help predict `lpsa` given a subset of the eight predictors in `c5_prost_ks`.

7.4 Forward Selection with the `step` function

1. Specify the null model (intercept only)
2. Specify the variables R should consider as predictors (in the scope element of the step function)
3. Specify forward selection only
4. R defaults to using AIC as its stepwise criterion

```
with(prost,
     step(lm(lpsa ~ 1),
     scope=(~ lcavol + lweight + age + bph_f + svi_f +
                lcp + gleason_f + pgg45),
     direction="forward"))
```

```
Start:  AIC=28.84
lpsa ~ 1

            Df Sum of Sq     RSS     AIC
+ lcavol     1    69.003  58.915 -44.366
+ svi_f      1    41.011  86.907  -6.658
+ lcp        1    38.528  89.389  -3.926
+ gleason_f  2    30.121  97.796   6.793
+ lweight    1    24.019 103.899  10.665
+ pgg45      1    22.814 105.103  11.783
+ age        1     3.679 124.239  28.007
<none>                   127.918  28.838
+ bph_f      2     4.681 123.237  29.221

Step:  AIC=-44.37
lpsa ~ lcavol

            Df Sum of Sq    RSS     AIC
+ lweight    1    7.1726 51.742 -54.958
+ svi_f      1    5.2375 53.677 -51.397
+ bph_f      2    3.2994 55.615 -45.956
+ pgg45      1    1.6980 57.217 -45.203
+ gleason_f  2    2.7834 56.131 -45.061
<none>                   58.915 -44.366
+ lcp        1    0.6562 58.259 -43.452
+ age        1    0.0025 58.912 -42.370
```

```
Step:  AIC=-54.96
lpsa ~ lcavol + lweight

            Df Sum of Sq    RSS     AIC
+ svi_f      1    5.1737 46.568 -63.177
+ pgg45      1    1.8158 49.926 -56.424
+ gleason_f  2    2.6770 49.065 -56.111
<none>                   51.742 -54.958
+ lcp        1    0.8187 50.923 -54.506
+ age        1    0.6456 51.097 -54.176
+ bph_f      2    1.4583 50.284 -53.731

Step:  AIC=-63.18
lpsa ~ lcavol + lweight + svi_f

            Df Sum of Sq    RSS     AIC
<none>                   46.568 -63.177
+ gleason_f  2   1.60467 44.964 -62.579
+ age        1   0.62301 45.945 -62.484
+ bph_f      2   1.50046 45.068 -62.354
+ pgg45      1   0.50069 46.068 -62.226
+ lcp        1   0.06937 46.499 -61.322


Call:
lm(formula = lpsa ~ lcavol + lweight + svi_f)

Coefficients:
(Intercept)       lcavol      lweight     svi_fYes
    -0.7772       0.5259       0.6618       0.6657
```

The resulting model, arrived at after three forward selection steps, includes `lcavol`, `lweight` and `svi_f`.

```
model.fs <- lm(lpsa ~ lcavol + lweight + svi_f,
               data=prost)
summary(model.fs)$adj.r.squared
```

```
[1] 0.6242063
```

```
extractAIC(model.fs)
```

```
[1]   4.00000 -63.17744
```

The adjusted R^2 value for this model is 0.624, and the AIC value used by the stepwise procedure is -63.18, on 4 effective degrees of freedom.

7.5 Backward Elimination using the `step` function

In this case, the backward elimination approach, using reduction in AIC for a criterion, comes to the same conclusion about the "best" model.

```
with(prost,
     step(lm(lpsa ~ lcavol + lweight + age + bph_f +
                 svi_f + lcp + gleason_f + pgg45),
          direction="backward"))
```

```
Start:  AIC=-61.4
```

```
lpsa ~ lcavol + lweight + age + bph_f + svi_f + lcp + gleason_f +
    pgg45

            Df Sum of Sq    RSS     AIC
- gleason_f  2    1.1832 42.240 -62.639
- pgg45      1    0.4409 41.498 -62.359
- lcp        1    0.8492 41.906 -61.409
<none>                   41.057 -61.395
- bph_f      2    2.0299 43.087 -60.714
- age        1    2.2129 43.270 -58.303
- svi_f      1    4.1207 45.178 -54.118
- lweight    1    5.7123 46.769 -50.760
- lcavol     1   18.2738 59.331 -27.683

Step:  AIC=-62.64
lpsa ~ lcavol + lweight + age + bph_f + svi_f + lcp + pgg45

          Df Sum of Sq    RSS     AIC
- lcp      1    0.8470 43.087 -62.713
<none>                 42.240 -62.639
- pgg45    1    1.2029 43.443 -61.916
- bph_f    2    2.2515 44.492 -61.602
- age      1    2.0730 44.313 -59.992
- svi_f    1    4.6431 46.884 -54.523
- lweight  1    5.5988 47.839 -52.566
- lcavol   1   21.4956 63.736 -24.736

Step:  AIC=-62.71
lpsa ~ lcavol + lweight + age + bph_f + svi_f + pgg45

          Df Sum of Sq    RSS     AIC
- pgg45    1    0.5860 43.673 -63.403
<none>                 43.087 -62.713
- bph_f    2    2.0214 45.109 -62.266
- age      1    1.7101 44.798 -60.938
- svi_f    1    3.7964 46.884 -56.523
- lweight  1    5.6462 48.734 -52.769
- lcavol   1   22.5152 65.603 -23.936

Step:  AIC=-63.4
lpsa ~ lcavol + lweight + age + bph_f + svi_f

          Df Sum of Sq    RSS     AIC
<none>                 43.673 -63.403
- bph_f    2    2.2720 45.945 -62.484
- age      1    1.3945 45.068 -62.354
- svi_f    1    5.2747 48.948 -54.343
- lweight  1    5.3319 49.005 -54.230
- lcavol   1   25.5538 69.227 -20.720


Call:
lm(formula = lpsa ~ lcavol + lweight + age + bph_f + svi_f)
```

```
Coefficients:
(Intercept)       lcavol      lweight          age  bph_fMedium
    0.14329      0.54022      0.67283     -0.01819      0.37607
  bph_fHigh     svi_fYes
    0.27216      0.68174
```

The backwards elimination approach in this case lands on a model with five inputs (one of which includes two `bph` indicators,) eliminating only `gleason_f`, `pgg45` and `lcp`.

7.6 Allen-Cady Modified Backward Elimination

Ranking candidate predictors by importance in advance of backwards elimination can help avoid false-positives, while reducing model size. See Vittinghoff et al. (2012), Section 10.3 for more details.

1. First, force into the model any predictors of primary interest, and any confounders necessary for face validity of the final model.
   - "Some variables in the hypothesized causal model may be such well-established causal antecedents of the outcome that it makes sense to include them, essentially to establish the face validity of the model and without regard to the strength or statistical significance of their associations with the primary predictor and outcome …"
2. Rank the remaining candidate predictors in order of importance.
3. Starting from an initial model with all candidate predictors included, delete predictors in order of ascending importance until the first variable meeting a criterion to stay in the model hits. Then stop.

Only the remaining variable hypothesized to be least important is eligible for removal at each step. When we are willing to do this sorting before collecting (or analyzing) the data, then we can do Allen-Cady backwards elimination using the `drop1` command in R.

7.6.1 Demonstration of the Allen-Cady approach

Suppose, for the moment that we decided to fit a model for the log of `psa` and we decided (before we saw the data) that we would:

lcavol + lweight + svi_f + age + bph_f + gleason_f + lcp + pgg45

- force the `gleason_f` variable to be in the model, due to prior information about its importance,
- and then rated the importance of the other variables as `lcavol` (most important), then `svi_f` then `age`, and then `bph_f`, then `lweight` and `lcp` followed by `pgg45` (least important)

When we are willing to do this sorting before collecting (or analyzing) the data, then we can do Allen-Cady backwards elimination using the `drop1` command in R.

Step 1. Fit the full model, then see if removing `pgg45` improves AIC…

```
with(prost, drop1(lm(lpsa ~ gleason_f + lcavol + svi_f +
              age + bph_f + lweight + lcp + pgg45),
              scope = (~ pgg45)))
```

```
Single term deletions

Model:
lpsa ~ gleason_f + lcavol + svi_f + age + bph_f + lweight + lcp +
    pgg45
       Df Sum of Sq    RSS     AIC
<none>              41.057 -61.395
pgg45   1   0.44085 41.498 -62.359
```

Since -62.3 is smaller (i.e. more negative) than -61.4, we delete `pgg45` and move on to assess whether we can remove the variable we deemed next least important (`lcp`)

Step 2. Let's see if removing `lcp` from this model improves AIC...

```
with(prost, drop1(lm(lpsa ~ gleason_f + lcavol + svi_f +
              age + bph_f + lweight  + lcp),
              scope = (~ lcp)))
```

```
Single term deletions

Model:
lpsa ~ gleason_f + lcavol + svi_f + age + bph_f + lweight + lcp
       Df Sum of Sq    RSS     AIC
<none>              41.498 -62.359
lcp     1   0.56767 42.066 -63.041
```

Again, since -63.0 is smaller than -62.4, we delete `lcp` and next assess whether we should delete `lweight`.

Step 3. Does removing `lweight` from this model improves AIC...

```
with(prost, drop1(lm(lpsa ~ gleason_f + lcavol + svi_f +
              age + bph_f + lweight),
              scope = (~ lweight)))
```

```
Single term deletions

Model:
lpsa ~ gleason_f + lcavol + svi_f + age + bph_f + lweight
        Df Sum of Sq    RSS     AIC
<none>               42.066 -63.041
lweight  1     5.678 47.744 -52.760
```

Since the AIC for the model after the removal of `lweight` is larger (i.e. less negative), we stop, and declare our final model by the Allen-Cady approach to include `gleason_f`, `lcavol`, `svi_f`, `age`, `bph_f` and `lweight`.

7.7 Summarizing the Results

Method	Suggested Predictors
Forward selection	`lcavol`, `lweight`, `svi_f`
Backwards elimination	`lcavol`, `lweight`, `svi_f`, `age`, `bph_f`
Allen-Cady approach	`lcavol`, `lweight`, `svi_f`, `age`, `bph_f`, `gleason_f`

7.7.1 In-Sample Testing and Summaries

Since these models are nested in each other, let's look at the summary statistics (like R^2, and AIC) and also run an ANOVA-based comparison of these nested models to each other and to the model with the intercept alone, and the kitchen sink model with all available predictors.

```
prost_m_int <- lm(lpsa ~ 1, data = prost)
prost_m_fw <- lm(lpsa ~ lcavol + lweight + svi_f, data = prost)
prost_m_bw <- lm(lpsa ~ lcavol + lweight + svi_f +
              age + bph_f + gleason_f, data = prost)
prost_m_ac <- lm(lpsa ~ lcavol + lweight + svi_f +
              age + bph_f + gleason_f + lcp, data = prost)
```

```
prost_m_ks <- lm(lpsa ~ lcavol + lweight + svi_f +
              age + bph_f + gleason_f + lcp + pgg45, data = prost)
```

7.7.1.1 Model Fit Summaries (in-sample) from glance

Here are the models, at a glance from the broom package.

```
prost_sum <- bind_rows(glance(prost_m_int), glance(prost_m_fw),
                       glance(prost_m_bw), glance(prost_m_ac),
                       glance(prost_m_ks)) %>% round(., 3)
prost_sum$names <- c("intercept", "lcavol + lweight + svi",
                     "... + age + bhp + gleason", "... + lcp", "... + pgg45")
prost_sum <- prost_sum %>%
    select(names, r.squared, adj.r.squared, AIC, BIC, sigma, df, df.residual)

prost_sum
```

```
                    names r.squared adj.r.squared     AIC     BIC sigma
1                 intercept     0.000         0.000 306.112 311.261 1.154
2    lcavol + lweight + svi     0.636         0.624 214.097 226.970 0.708
3 ... + age + bhp + gleason     0.671         0.641 214.233 239.980 0.691
4                 ... + lcp     0.676         0.642 214.915 243.237 0.691
5               ... + pgg45     0.679         0.642 215.879 246.775 0.691
  df df.residual
1  1          96
2  4          93
3  9          88
4 10          87
5 11          86
```

From these summaries, it looks like:

- the adjusted R^2 is essentially indistinguishable between the three largest models, but a bit less strong with the three-predictor (4 df) model, and
- the AIC and BIC are (slightly) better (lower) with the three-predictor model (4 df) than any other.

So we might be motivated by these summaries to select any of the three models we're studying closely here.

7.7.1.2 Model Testing via ANOVA (in-sample)

To obtain ANOVA-based test results, we'll run...

```
anova(prost_m_int, prost_m_fw, prost_m_bw, prost_m_ac, prost_m_ks)
```

```
Analysis of Variance Table

Model 1: lpsa ~ 1
Model 2: lpsa ~ lcavol + lweight + svi_f
Model 3: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f
Model 4: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f + lcp
Model 5: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f + lcp +
    pgg45
  Res.Df     RSS Df Sum of Sq       F Pr(>F)
1     96 127.918
2     93  46.568  3    81.349 56.7991 <2e-16 ***
```

```
3     88  42.066  5     4.503  1.8863 0.1050
4     87  41.498  1     0.568  1.1891 0.2786
5     86  41.057  1     0.441  0.9234 0.3393
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What conclusions can we draw on the basis of these ANOVA tests?

- There is a statistically significant improvement in predictive value for Model 2 (the forward selection approach) as compared to Model 1 (the intercept only.)
- The ANOVA test comparing Model 5 (kitchen sink) to Model 4 (Allen-Cady result) shows no statistically significant improvement in predictive value.
- Neither does the ANOVA test comparing Model 3 to Model 2 or Model 4 to Model 3.

This suggests that, **if we are willing to let the ANOVA test decide our best model** than that would be the model produced by forward selection, with predictors `lcavol`, `lweight` and `svi_f`. But we haven't validated the models.

1. If the purpose of the model is to predict new data, some sort of out-of-sample or cross-validation approach will be necessary, and
2. Even if our goal isn't prediction but merely description of the current data, we would still want to build diagnostic plots to regression assumptions in each model, and
3. There is no reason to assume in advance that any of these models is in fact correct, or that any one of these stepwise approaches is necessarily better than any other, and
4. The mere act of running a stepwise regression model, as we'll see, can increase the bias in our findings if we accept the results at face value.

So we'll need some ways to validate the results once we complete the selection process.

7.7.2 Validating the Results of the Various Models

We can use a 5-fold cross-validation approach to assess the predictions made by our potential models and then compare them. Let's compare our three models:

- the three predictor model obtained by forward selection, including `lcavol`, `lweight`, and `svi_f`
- the five predictor model obtained by backwards elimination, including `lcavol`, `lweight`, `svi_f`, and also `age`, and `bph_f`
- the six predictor model obtained by the Allen-Cady approach, adding `gleason_f` to the previous model.

Here's the 5-fold validation work (and resulting RMSE and MAE estimates) for the three-predictor model.

```
set.seed(43201012)

prost3_models <- prost %>%
    crossv_kfold(k = 5) %>%
    mutate(model = map(train, ~ lm(lpsa ~ lcavol + lweight +
                                       svi_f, data = .)))

prost3_preds <- prost3_models %>%
    unnest(map2(model, test, ~ augment(.x, newdata = .y)))

prost3_preds %>%
    summarize(RMSE_prost3 = sqrt(mean((lpsa - .fitted) ^2)),
              MAE_prost3 = mean(abs(lpsa - .fitted)))
```

```
# A tibble: 1 x 2
  RMSE_prost3 MAE_prost3
        <dbl>      <dbl>
```

```
1       0.745      0.587
```

Now, we'll generate the RMSE and MAE estimates for the five-predictor model.

```
set.seed(43206879)

prost5_models <- prost %>%
    crossv_kfold(k = 5) %>%
    mutate(model = map(train, ~ lm(lpsa ~ lcavol + lweight +
                                       svi_f + age + bph_f, data = .)))

prost5_preds <- prost5_models %>%
    unnest(map2(model, test, ~ augment(.x, newdata = .y)))

prost5_preds %>%
    summarize(RMSE_prost5 = sqrt(mean((lpsa - .fitted) ^2)),
              MAE_prost5 = mean(abs(lpsa - .fitted)))
```

```
# A tibble: 1 x 2
  RMSE_prost5 MAE_prost5
        <dbl>      <dbl>
1       0.750      0.581
```

And at last, we'll generate the RMSE and MAE estimates for the six-predictor model.

```
set.seed(43236198)

prost6_models <- prost %>%
    crossv_kfold(k = 5) %>%
    mutate(model = map(train, ~ lm(lpsa ~ lcavol + lweight +
                                       svi_f + age + bph_f + gleason_f, data = .)))

prost6_preds <- prost6_models %>%
    unnest(map2(model, test, ~ augment(.x, newdata = .y)))

prost6_preds %>%
    summarize(RMSE_prost6 = sqrt(mean((lpsa - .fitted) ^2)),
              MAE_prost6 = mean(abs(lpsa - .fitted)))
```

```
# A tibble: 1 x 2
  RMSE_prost6 MAE_prost6
        <dbl>      <dbl>
1       0.720      0.551
```

It appears that the six-predictor model does better than either of the other two approaches, with smaller RMSE and MAE. The three-predictor model does slightly better in terms of root mean square prediction error and slightly worse in terms of mean absolute prediction error than the five-predictor model.

OK. A mixed bag, with different conclusions depending on which summary we want to look at. But of course, stepwise regression isn't the only way to do variable selection. Let's consider a broader range of potential predictor sets.

Chapter 8

"Best Subsets" Variable Selection in our Prostate Cancer Study

A second approach to model selection involved fitting all possible subset models and identifying the ones that look best according to some meaningful criterion and ideally one that includes enough variables to model the response appropriately without including lots of redundant or unnecessary terms.

8.1 Four Key Summaries We'll Use to Evaluate Potential Models

1. Adjusted R^2, which we try to maximize.
2. Akaike's Information Criterion (AIC), which we try to minimize, and a Bias-Corrected version of AIC due to Hurvich and Tsai (1989), which we use when the sample size is small, specifically when the sample size n and the number of predictors being studied k are such that $n/k \leq 40$. We also try to minimize this bias-corrected AIC.
3. Bayesian Information Criterion (BIC), which we also try to minimize.
4. Mallows' C_p statistic, which we (essentially) try to minimize.

Choosing between AIC and BIC can be challenging.

> For model selection purposes, there is no clear choice between AIC and BIC. Given a family of models, including the true model, the probability that BIC will select the correct model approaches one as the sample size n approaches infinity - thus BIC is asymptotically consistent, which AIC is not. [But, for practical purposes,] BIC often chooses models that are too simple [relative to AIC] because of its heavy penalty on complexity.

- Source: Hastie et al. (2001), page 208.

Several useful tools for running "all subsets" or "best subsets" regression comparisons are developed in R's `leaps` package.

8.2 Using `regsubsets` in the `leaps` package

We can use the `leaps` package to obtain results in the `prost` study from looking at all possible subsets of the candidate predictors. The `leaps` package isn't particularly friendly to the tidyverse. In particular, we **cannot have any character variables** in our predictor set. We specify our "kitchen sink" model, and apply the `regsubsets` function from `leaps`, which identifies the set of models.

To start, we'll ask R to find the one best subset (with 1 predictor variable [in addition to the intercept], then with 2 predictors, and then with each of 3, 4, … 8 predictor variables) according to an exhaustive search without forcing any of the variables to be in or out.

- Use the `nvmax` command within the `regsubsets` function to limit the number of regression inputs to a maximum.
- Use the `nbest` command to identify how many subsets you want to identify for each predictor count.
- If all of your predictors are **quantitative** or **binary** then you can skip the `preds` step, and simply place your kitchen sink model into `regsubsets`.
- But if you have multi-categorical variables (like `gleason_f` or `svi_f` in our case) then you must create a `preds` group, as follows.

```
preds <- with(prost, cbind(lcavol, lweight, age, bph_f,
                           svi_f, lcp, gleason_f, pgg45))

rs.ks <- regsubsets(preds, y = prost$lpsa,
                    nvmax = 8, nbest = 1)
rs.summ <- summary(rs.ks)
rs.summ
```

```
Subset selection object
8 Variables  (and intercept)
          Forced in Forced out
lcavol        FALSE      FALSE
lweight       FALSE      FALSE
age           FALSE      FALSE
bph_f         FALSE      FALSE
svi_f         FALSE      FALSE
lcp           FALSE      FALSE
gleason_f     FALSE      FALSE
pgg45         FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
         lcavol lweight age bph_f svi_f lcp gleason_f pgg45
1  ( 1 ) "*"    " "     " " " "   " "   " " " "       " "
2  ( 1 ) "*"    "*"     " " " "   " "   " " " "       " "
3  ( 1 ) "*"    "*"     " " " "   "*"   " " " "       " "
4  ( 1 ) "*"    "*"     " " "*"   "*"   " " " "       " "
5  ( 1 ) "*"    "*"     "*" "*"   "*"   " " " "       " "
6  ( 1 ) "*"    "*"     "*" "*"   "*"   " " "*"       " "
7  ( 1 ) "*"    "*"     "*" "*"   "*"   "*" "*"       " "
8  ( 1 ) "*"    "*"     "*" "*"   "*"   "*" "*"       "*"
```

So…

- the best one-predictor model used `lcavol`
- the best two-predictor model used `lcavol` and `lweight`
- the best three-predictor model used `lcavol`, `lweight` and `svi_f`
- the best four-predictor model added `bph_f`, and
- the best five-predictor model added `age`
- the best six-input model added `gleason_f`,
- the best seven-input model added `lcp`,
- and the eight-input model adds `pgg45`.

All of these "best subsets" are hierarchical, in that each model is a subset of the one below it. This isn't inevitably true.

- To determine which model is best, we can plot key summaries of model fit (adjusted R^2, Mallows' C_p,

bias-corrected AIC, and BIC) using either base R plotting techniques (what I've done in the past) or `ggplot2` (what I use now.) I'll show both types of plotting approaches in the next two sections.

8.2.1 Identifying the models with `which` and `outmat`

To see the models selected by the system, we use:

```
rs.summ$which
```

```
  (Intercept) lcavol lweight   age bph_f svi_f   lcp gleason_f pgg45
1        TRUE   TRUE   FALSE FALSE FALSE FALSE FALSE     FALSE FALSE
2        TRUE   TRUE    TRUE FALSE FALSE FALSE FALSE     FALSE FALSE
3        TRUE   TRUE    TRUE FALSE FALSE  TRUE FALSE     FALSE FALSE
4        TRUE   TRUE    TRUE FALSE  TRUE  TRUE FALSE     FALSE FALSE
5        TRUE   TRUE    TRUE  TRUE  TRUE  TRUE FALSE     FALSE FALSE
6        TRUE   TRUE    TRUE  TRUE  TRUE  TRUE FALSE      TRUE FALSE
7        TRUE   TRUE    TRUE  TRUE  TRUE  TRUE  TRUE      TRUE FALSE
8        TRUE   TRUE    TRUE  TRUE  TRUE  TRUE  TRUE      TRUE  TRUE
```

Another version of this formatted for printing is:

```
rs.summ$outmat
```

```
         lcavol lweight age bph_f svi_f lcp gleason_f pgg45
1  ( 1 ) "*"    " "     " " " "   " "   " " " " "       " "
2  ( 1 ) "*"    "*"     " " " "   " "   " " " " "       " "
3  ( 1 ) "*"    "*"     " " " "   "*"   " " " " "       " "
4  ( 1 ) "*"    "*"     " " "*"   "*"   " " " " "       " "
5  ( 1 ) "*"    "*"     "*" "*"   "*"   " " " " "       " "
6  ( 1 ) "*"    "*"     "*" "*"   "*"   " " "*" "       " "
7  ( 1 ) "*"    "*"     "*" "*"   "*"   "*" "*" "       " "
8  ( 1 ) "*"    "*"     "*" "*"   "*"   "*" "*" "       "*"
```

We built one subset of each size up to eight predictors, and if we add the intercept term, this means we have models of size k = 2, 3, 4, 5, 6, 7, 8 and 9.

The models are:

Size k	Predictors included (besides intercept)
2	`lcavol`
3	`lcavol` and `lweight`
4	add `svi_f`
5	add `bph_f`
6	add `age`
7	add `gleason_f`
8	add `lcp`
9	add `pgg45`

8.3 Calculating bias-corrected AIC

The bias-corrected AIC formula developed in Hurvich and Tsai (1989) requires three inputs:

- the residual sum of squares for a model
- the sample size (n) or number of observations used to fit the model
- the number of regression inputs, k, including the intercept, used in the model

So, for a particular model fit to n observations, on k predictors (including the intercept) and a residual sum of squares equal to RSS, we have:

$$AIC_c = nlog(\frac{RSS}{n}) + 2k + \frac{2k(k+1)}{n-k-1}$$

Note that the corrected AIC_c can be related to the original AIC via:

$$AIC_c = AIC + \frac{2k(k+1)}{n-k-1}$$

8.3.1 Calculation of aic.c in our setting

In our case, we have $n = 97$ observations, and built a series of models with $k =$ `2:9` predictors (including the intercept in each case), so we will insert those values into the general formula for bias-corrected AIC which is:

```
aic.c <- n * log( rs.summ$rss / n) + 2 * k +
                      (2 * k * (k + 1) / (n - k - 1))
```

We can obtain the residual sum of squares explained by each model by pulling `rss` from the `regsubsets` summary contained here in `rs.summ`.

```
data_frame(k = 2:9, RSS = rs.summ$rss)
```

```
# A tibble: 8 x 2
      k   RSS
  <int> <dbl>
1     2  58.9
2     3  51.7
3     4  46.6
4     5  45.7
5     6  44.6
6     7  43.7
7     8  43.0
8     9  42.8
```

In this case, we have:

```
rs.summ$aic.c <- 97*log(rs.summ$rss / 97) + 2*(2:9) +
                (2 * (2:9) * ((2:9)+1) / (97 - (2:9) - 1))

round(rs.summ$aic.c,2) # bias-corrected
```

```
[1] -44.24 -54.70 -62.74 -62.29 -62.34 -62.11 -61.17 -59.36
```

The impact of this bias correction can be modest but important. Here's a little table looking closely at the results in this problem. The uncorrected AIC are obtained using `extractAIC`, as described in the next section.

Size	2	3	4	5	6	7	8	9
Bias-corrected AIC	-44.2	-54.7	-62.7	-62.3	-62.3	-62.1	-61.2	-59.4
Uncorrected AIC	-44.4	-55.0	-63.2	-62.4	-63.4	-63.0	-62.4	-61.4

8.3.2 The Uncorrected AIC provides no more useful information here

We could, if necessary, also calculate the *uncorrected* `aic` value for each model, but we won't make any direct use of that, because that will not provide any new information not already gathered by the C_p statistic for a linear regression model. If you wanted to find the uncorrected AIC for a given model, you can use the `extractAIC` function.

```
extractAIC(lm(lpsa ~ lcavol, data = prost))
```

```
[1]   2.00000 -44.36603
```

```
extractAIC(lm(lpsa ~ lcavol + lweight, data = prost))
```

```
[1]   3.00000 -54.95846
```

Note that:

- these results are fairly comparable to the bias-corrected AIC we built above, and
- the `extractAIC` and `AIC` functions look like they give very different results, but they really don't.

```
AIC(lm(lpsa ~ lcavol, data = prost))
```

```
[1] 232.908
```

```
AIC(lm(lpsa ~ lcavol + lweight, data = prost))
```

```
[1] 222.3156
```

But notice that the differences in AIC are the same, either way, comparing these two models:

```
extractAIC(lm(lpsa ~ lcavol, data = prost)) - extractAIC(lm(lpsa ~ lcavol + lweight, data = prost))
```

```
[1] -1.00000 10.59243
```

```
AIC(lm(lpsa ~ lcavol, data = prost)) - AIC(lm(lpsa ~ lcavol + lweight, data = prost))
```

```
[1] 10.59243
```

- AIC is only defined up to an additive constant.
- Since the difference between two models using either `AIC` or `extractAIC` is the same, this doesn't actually matter which one we use, so long as we use the same one consistently.

8.3.3 Building a Tibble containing the necessary information

Again, note the use of 2:9 for the values of k, because we're fitting one model for each size from 2 through 9.

```
best_mods_1 <- data_frame(
    k = 2:9,
    r2 = rs.summ$rsq,
    adjr2 = rs.summ$adjr2,
    cp = rs.summ$cp,
    aic.c = rs.summ$aic.c,
    bic = rs.summ$bic
)

best_mods <- cbind(best_mods_1, rs.summ$which)

best_mods
```

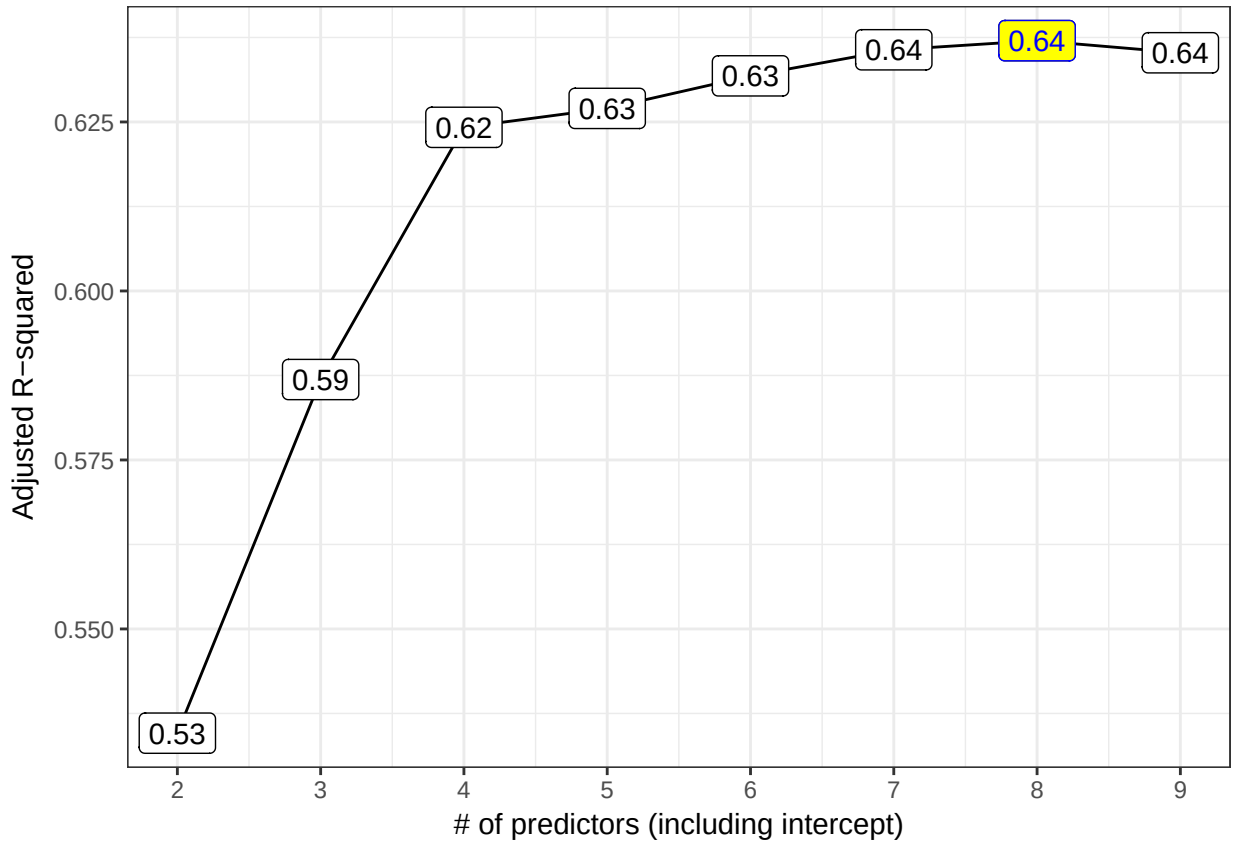
```
  k        r2     adjr2        cp     aic.c       bic (Intercept) lcavol
1 2 0.5394320 0.5345839 28.213914 -44.23838 -66.05416        TRUE   TRUE
```

```
2 3 0.5955040 0.5868977 15.456669 -54.70040 -74.07188        TRUE   TRUE
3 4 0.6359499 0.6242063  6.811986 -62.74265 -79.71614        TRUE   TRUE
4 5 0.6425479 0.6270065  7.075509 -62.29223 -76.91557        TRUE   TRUE
5 6 0.6509970 0.6318211  6.851826 -62.33858 -74.66120        TRUE   TRUE
6 7 0.6584484 0.6356783  6.890739 -62.10692 -72.17992        TRUE   TRUE
7 8 0.6634967 0.6370302  7.562119 -61.17338 -69.04961        TRUE   TRUE
8 9 0.6656326 0.6352355  9.000000 -59.35841 -65.09253        TRUE   TRUE
  lweight   age bph_f svi_f   lcp gleason_f pgg45
1   FALSE FALSE FALSE FALSE FALSE     FALSE FALSE
2    TRUE FALSE FALSE FALSE FALSE     FALSE FALSE
3    TRUE FALSE FALSE  TRUE FALSE     FALSE FALSE
4    TRUE FALSE  TRUE  TRUE FALSE     FALSE FALSE
5    TRUE  TRUE  TRUE  TRUE FALSE     FALSE FALSE
6    TRUE  TRUE  TRUE  TRUE FALSE      TRUE FALSE
7    TRUE  TRUE  TRUE  TRUE  TRUE      TRUE FALSE
8    TRUE  TRUE  TRUE  TRUE  TRUE      TRUE  TRUE
```

8.4 Plotting the Best Subsets Results using `ggplot2`

8.4.1 The Adjusted R^2 Plot

```
p1 <- ggplot(best_mods, aes(x = k, y = adjr2,
                            label = round(adjr2,2))) +
    geom_line() +
    geom_label() +
    geom_label(data = subset(best_mods,
                             adjr2 == max(adjr2)),
               aes(x = k, y = adjr2, label = round(adjr2,2)),
               fill = "yellow", col = "blue") +
    theme_bw() +
    scale_x_continuous(breaks = 2:9) +
    labs(x = "# of predictors (including intercept)",
         y = "Adjusted R-squared")

p1
```



Models 4-9 all look like reasonable choices here. The maximum adjusted R^2 is seen in the model of size 8.

8.4.2 Mallows' C_p

The C_p statistic focuses directly on the tradeoff between **bias** (due to excluding important predictors from the model) and extra **variance** (due to including too many unimportant predictors in the model.)

If N is the sample size, and we select p regression predictors from a set of K (where $p < K$), then the C_p statistic is

$C_p = \frac{SSE_p}{MSE_K} - N + 2p$

where:

- SSE_p is the sum of squares for error (residual) in the model with p predictors
- MSE_K is the residual mean square after regression in the model with all K predictors

As it turns out, this is just measuring the particular model's lack of fit, and then adding a penalty for the number of terms in the model (specifically $2p - N$ is the penalty since the lack of fit is measured as $(N - p)\frac{SSE_p}{MSE_K}$.

- If a model has no meaningful lack of fit (i.e. no substantial bias) then the expected value of C_p is roughly p.
- Otherwise, the expectation is p plus a positive bias term.
- In general, we want to see *smaller* values of C_p.
- We usually select a "winning model" by choosing a subset of predictors that have C_p near the value of p.

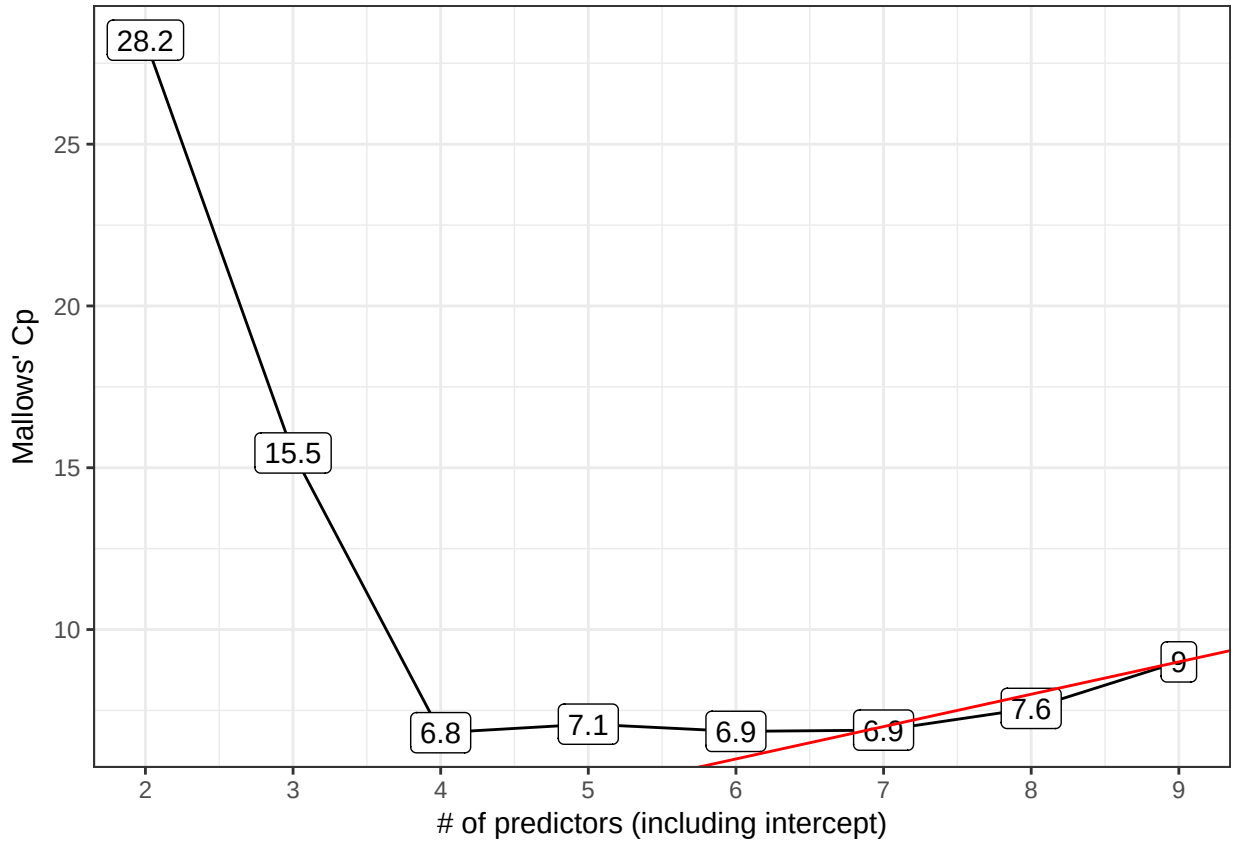
8.4.3 The C_p Plot

The C_p plot is just a scatterplot of C_p on the Y-axis, and the size of the model (coefficients plus intercept) $p = k$ on the X-axis.

Each of the various predictor subsets we will study is represented in a single point. A model without bias should have C_p roughly equal to p, so we'll frequently draw a line at $C_p = p$ to make that clear. We then select our model from among all models with small C_p statistics.

- My typical approach is to identify the models where $C_p - p \geq 0$, then select from among those models the model where $C_p - p$ is minimized, and if there is a tie, select the model where p is minimized.
- Another good candidate might be a slightly overfit model (where $C_p - p < 0$ but just barely.)

```
p2 <- ggplot(best_mods, aes(x = k, y = cp,
                            label = round(cp,1))) +
    geom_line() +
    geom_label() +
    geom_abline(intercept = 0, slope = 1,
                col = "red") +
    theme_bw() +
    scale_x_continuous(breaks = 2:9) +
    labs(x = "# of predictors (including intercept)",
         y = "Mallows' Cp")

p2
```



- Model 6 is a possibility here, with the difference $C_p - p$ minimized among all models with $C_p >= p$.
- Model 7 also looks pretty good, with C_p just barely smaller than the size (p = 7) of the model.

8.4.4 "All Subsets" Regression and Information Criteria

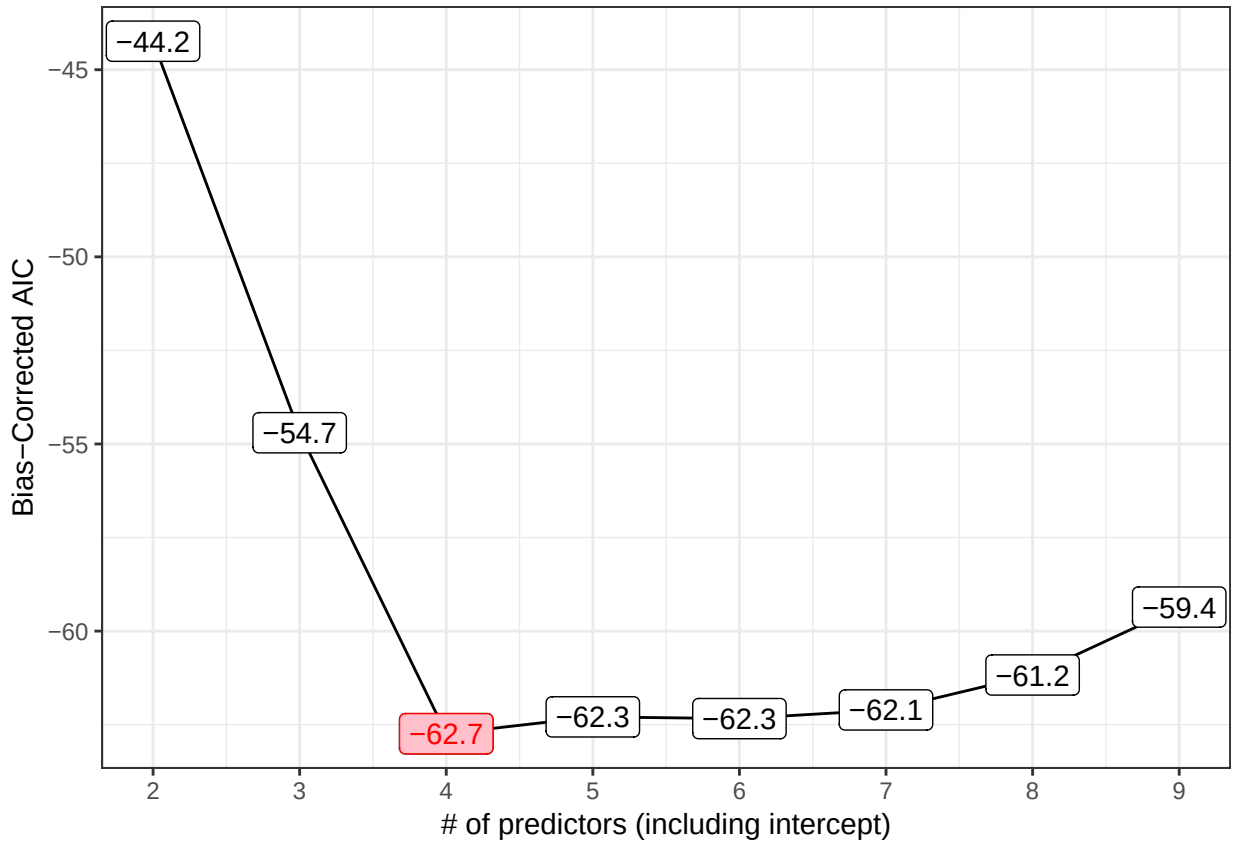
We might consider any of three main information criteria:

- the Bayesian Information Criterion, called BIC
- the Akaike Information Criterion (used by R's default stepwise approaches,) called AIC
- a corrected version of AIC due to Hurvich and Tsai (1989), called AIC_c or `aic.c`

Each of these indicates better models by getting smaller. Since the C_p and AIC results will lead to the same model, I'll focus on plotting the bias-corrected AIC and on BIC.

8.4.5 The bias-corrected AIC plot

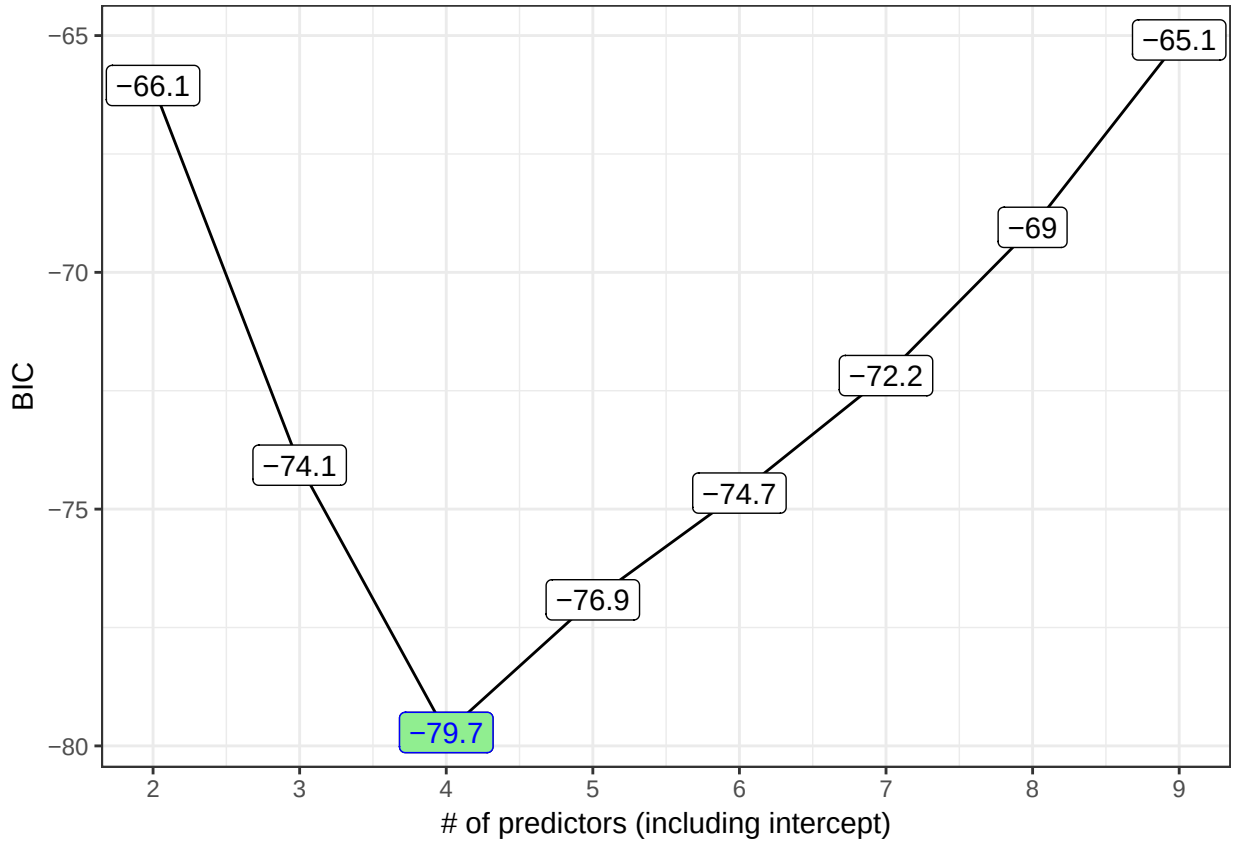
```
p3 <- ggplot(best_mods, aes(x = k, y = aic.c,
                            label = round(aic.c,1))) +
    geom_line() +
    geom_label() +
    geom_label(data = subset(best_mods, aic.c == min(aic.c)),
               aes(x = k, y = aic.c), fill = "pink",
               col = "red") +
    theme_bw() +
    scale_x_continuous(breaks = 2:9) +
    labs(x = "# of predictors (including intercept)",
         y = "Bias-Corrected AIC")

p3
```



The smallest AIC_c values occur in models 4 and later, especially model 4 itself.

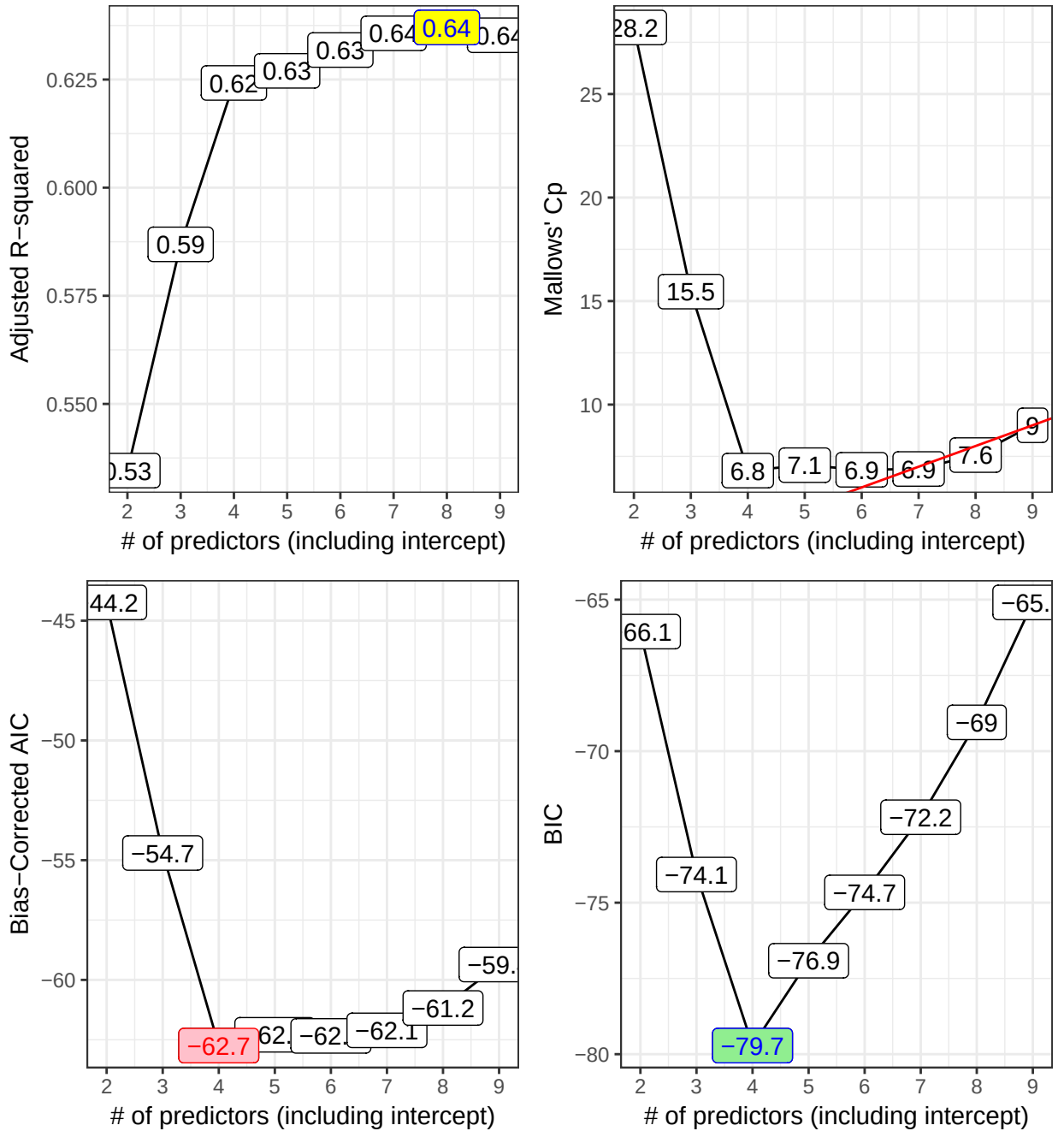
8.4.6 The BIC plot

```
p4 <- ggplot(best_mods, aes(x = k, y = bic,
                            label = round(bic,1))) +
    geom_line() +
    geom_label() +
    geom_label(data = subset(best_mods, bic == min(bic)),
               aes(x = k, y = bic),
               fill = "lightgreen", col = "blue") +
    theme_bw() +
    scale_x_continuous(breaks = 2:9) +
    labs(x = "# of predictors (including intercept)",
         y = "BIC")

p4
```



8.4.7 All Four Plots in One Figure (via ggplot2)

```
gridExtra::grid.arrange(p1, p2, p3, p4, nrow = 2)
```



8.5 Table of Key Results

We can build a big table, like this:

```
best_mods
```

```
  k        r2     adjr2        cp     aic.c       bic (Intercept) lcavol
1 2 0.5394320 0.5345839 28.213914 -44.23838 -66.05416        TRUE   TRUE
2 3 0.5955040 0.5868977 15.456669 -54.70040 -74.07188        TRUE   TRUE
3 4 0.6359499 0.6242063  6.811986 -62.74265 -79.71614        TRUE   TRUE
```

```
4 5 0.6425479 0.6270065  7.075509 -62.29223 -76.91557        TRUE   TRUE
5 6 0.6509970 0.6318211  6.851826 -62.33858 -74.66120        TRUE   TRUE
6 7 0.6584484 0.6356783  6.890739 -62.10692 -72.17992        TRUE   TRUE
7 8 0.6634967 0.6370302  7.562119 -61.17338 -69.04961        TRUE   TRUE
8 9 0.6656326 0.6352355  9.000000 -59.35841 -65.09253        TRUE   TRUE
  lweight   age bph_f svi_f   lcp gleason_f pgg45
1   FALSE FALSE FALSE FALSE FALSE     FALSE FALSE
2    TRUE FALSE FALSE FALSE FALSE     FALSE FALSE
3    TRUE FALSE FALSE  TRUE FALSE     FALSE FALSE
4    TRUE FALSE  TRUE  TRUE FALSE     FALSE FALSE
5    TRUE  TRUE  TRUE  TRUE FALSE     FALSE FALSE
6    TRUE  TRUE  TRUE  TRUE FALSE      TRUE FALSE
7    TRUE  TRUE  TRUE  TRUE  TRUE      TRUE FALSE
8    TRUE  TRUE  TRUE  TRUE  TRUE      TRUE  TRUE
```

8.6 Models Worth Considering?

k	Predictors	Reason
4	`lcavol lweight svi_f`	minimizes BIC, AIC_c
7	`+ age bph_f gleason_f`	C_p near p
8	`+ lcp`	max R^2_{adj}

8.7 Compare these candidate models in-sample?

8.7.1 Using `anova` to compare nested models

Let's run an ANOVA-based comparison of these nested models to each other and to the model with the intercept alone.

- The models are **nested** because `m04` is a subset of the predictors in `m07`, which includes a subset of the predictors in `m08`.

```
m.int <- lm(lpsa ~ 1, data = prost)
m04 <- lm(lpsa ~ lcavol + lweight + svi_f, data = prost)
m07 <- lm(lpsa ~ lcavol + lweight + svi_f +
              age + bph_f + gleason_f, data = prost)
m08 <- lm(lpsa ~ lcavol + lweight + svi_f +
              age + bph_f + gleason_f + lcp, data = prost)
m.full <- lm(lpsa ~ lcavol + lweight + svi_f +
              age + bph_f + gleason_f + lcp + pgg45, data = prost)
```

Next, we'll run...

```
anova(m.full, m08, m07, m04, m.int)
```

```
Analysis of Variance Table

Model 1: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f + lcp +
    pgg45
Model 2: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f + lcp
Model 3: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f
Model 4: lpsa ~ lcavol + lweight + svi_f
```

```
Model 5: lpsa ~ 1
  Res.Df     RSS Df Sum of Sq       F Pr(>F)
1     86  41.057
2     87  41.498 -1    -0.441  0.9234 0.3393
3     88  42.066 -1    -0.568  1.1891 0.2786
4     93  46.568 -5    -4.503  1.8863 0.1050
5     96 127.918 -3   -81.349 56.7991 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What conclusions can we draw here, on the basis of these ANOVA tests?

- The first p value, of 0.3393, compares what the `anova` called Model 1, and what we call `m.full` to what the `anova` called Model 2, and what we call `m08`. So there's no significant decline in predictive value observed when we drop from the `m.full` model to the `m08` model. This suggests that the `m08` model may be a better choice.
- The second p value, of 0.2786, compares `m08` to `m07`, and suggests that we lose no significant predictive value by dropping down to `m07`.
- The third p value, of 0.1050, compares `m07` to `m04`, and suggests that we lose no significant predictive value by dropping down to `m04`.
- But the fourth p value, of 2e-16 (or, functionally, zero), compares `m04` to `m.int` and suggests that we do gain significant predictive value by including the predictors in `m04` as compared to a model with an intercept alone.
- So, by the significance tests, the model we'd select would be `m04`, but, of course, in-sample statistical significance alone isn't a good enough reason to select a model if we want to do prediction well.

8.8 AIC and BIC comparisons, within the training sample

Next, we'll compare the three candidate models (ignoring the intercept-only and kitchen sink models) in terms of their AIC values and BIC values, again using the same sample we used to fit the models in the first place.

```
AIC(m04, m07, m08)
```

```
    df      AIC
m04  5 214.0966
m07 10 214.2327
m08 11 214.9148
```

```
BIC(m04, m07, m08)
```

```
    df      BIC
m04  5 226.9702
m07 10 239.9798
m08 11 243.2366
```

- The model with the smallest AIC value shows the best performance within the sample on that measure.
- Similarly, smaller BIC values are associated with predictor sets that perform better in sample on that criterion.
- BIC often suggests smaller models (with fewer regression inputs) than does AIC. Does that happen in this case?
- Note that `AIC` and `BIC` can be calculated in a few different ways, so we may see some variation if we don't compare apples to apples with regard to the R functions involved.

8.9 Cross-Validation of Candidate Models out of Sample

8.9.1 20-fold Cross-Validation of model m04

Model m04 uses lcavol, lweight and svi_f to predict the lpsa outcome. Let's do 20-fold cross-validation of this modeling approach, and calculate the root mean squared prediction error and the mean absolute prediction error for that modeling scheme.

```
set.seed(43201)

cv_m04 <- prost %>%
    crossv_kfold(k = 20) %>%
    mutate(model = map(train,
                       ~ lm(lpsa ~ lcavol + lweight + svi_f,
                                   data = .)))

cv_m04_pred <- cv_m04 %>%
    unnest(map2(model, test, ~ augment(.x, newdata = .y)))

cv_m04_results <- cv_m04_pred %>%
    summarize(Model = "m04",
              RMSE = sqrt(mean((lpsa - .fitted) ^2)),
              MAE = mean(abs(lpsa - .fitted)))

cv_m04_results
```

```
# A tibble: 1 x 3
  Model  RMSE   MAE
  <chr> <dbl> <dbl>
1 m04   0.725 0.574
```

8.9.2 20-fold Cross-Validation of model m07

Model m07 uses lcavol, lweight, svi_f, age, bph_f, and gleason_f to predict the lpsa outcome. Let's now do 20-fold cross-validation of this modeling approach, and calculate the root mean squared prediction error and the mean absolute prediction error for that modeling scheme. Note the small changes required, as compared to our cross-validation of model m04 a moment ago.

```
set.seed(43202)

cv_m07 <- prost %>%
    crossv_kfold(k = 20) %>%
    mutate(model = map(train,
                       ~ lm(lpsa ~ lcavol + lweight +
                                svi_f + age + bph_f +
                                gleason_f,
                                   data = .)))

cv_m07_pred <- cv_m07 %>%
    unnest(map2(model, test, ~ augment(.x, newdata = .y)))

cv_m07_results <- cv_m07_pred %>%
    summarize(Model = "m07",
              RMSE = sqrt(mean((lpsa - .fitted) ^2)),
```

```
              MAE = mean(abs(lpsa - .fitted)))

cv_m07_results
```

```
# A tibble: 1 x 3
  Model  RMSE   MAE
  <chr> <dbl> <dbl>
1 m07   0.730 0.556
```

8.9.3 20-fold Cross-Validation of model m08

Model m08 uses lcavol, lweight, svi_f, age, bph_f, gleason_f and lcp to predict the lpsa outcome. Let's now do 20-fold cross-validation of this modeling approach.

```
set.seed(43202)

cv_m08 <- prost %>%
    crossv_kfold(k = 20) %>%
    mutate(model = map(train,
                       ~ lm(lpsa ~ lcavol + lweight +
                                svi_f + age + bph_f +
                                gleason_f + lcp,
                                   data = .)))

cv_m08_pred <- cv_m08 %>%
    unnest(map2(model, test, ~ augment(.x, newdata = .y)))

cv_m08_results <- cv_m08_pred %>%
    summarize(Model = "m08",
              RMSE = sqrt(mean((lpsa - .fitted) ^2)),
              MAE = mean(abs(lpsa - .fitted)))

cv_m08_results
```

```
# A tibble: 1 x 3
  Model  RMSE   MAE
  <chr> <dbl> <dbl>
1 m08   0.729 0.557
```

8.9.4 Comparing the Results of the Cross-Validations

```
bind_rows(cv_m04_results, cv_m07_results, cv_m08_results)
```

```
# A tibble: 3 x 3
  Model  RMSE   MAE
  <chr> <dbl> <dbl>
1 m04   0.725 0.574
2 m07   0.730 0.556
3 m08   0.729 0.557
```

It appears that model m04 has the smallest RMSE and MAE in this case. So, that's the model with the strongest cross-validated predictive accuracy, by these two standards.

8.10 What about Interaction Terms?

Suppose we consider for a moment a much smaller and less realistic problem. We want to use best subsets to identify a model out of a set of three predictors for `lpsa`: specifically `lcavol`, `age` and `svi_f`, but now we also want to consider the interaction of `svi_f` with `lcavol` as a potential addition. Remember that `svi` is the 1/0 numeric version of `svi_f`. We could simply add a numerical product term to our model, as follows.

```
pred2 <- with(prost, cbind(lcavol, age, svi_f, svixlcavol = svi*lcavol))

rs.ks2 <- regsubsets(pred2, y = prost$lpsa,
                     nvmax = NULL, nbest = 1)
rs.summ2 <- summary(rs.ks2)
rs.summ2
```

```
Subset selection object
4 Variables  (and intercept)
           Forced in Forced out
lcavol         FALSE      FALSE
age            FALSE      FALSE
svi_f          FALSE      FALSE
svixlcavol     FALSE      FALSE
1 subsets of each size up to 4
Selection Algorithm: exhaustive
         lcavol age svi_f svixlcavol
1  ( 1 ) "*"    " " " "   " "
2  ( 1 ) "*"    " " "*"   " "
3  ( 1 ) "*"    " " "*"   "*"
4  ( 1 ) "*"    "*" "*"   "*"
```

In this case, best subsets doesn't identify the interaction term as an attractive predictor until it has already included the main effects that go into it. So that's fine. But if that isn't the case, we would have a problem.

To resolve this, we could:

1. Consider interactions beforehand, and force them in if desired.
2. Consider interaction terms outside of best subsets, and only after the selection of main effects.
3. Use another approach to deal with variable selection for interaction terms.

Chapter 9

Adding Non-linear Terms to a Linear Regression Model

9.1 The `pollution` data

Consider the `pollution` data set, which contain 15 independent variables and a measure of mortality, describing 60 US metropolitan areas in 1959-1961. The data come from McDonald and Schwing (1973), and are available at http://www4.stat.ncsu.edu/~boos/var.select/pollution.html and our web site.

```
pollution
```

```
# A tibble: 60 x 16
      x1    x2    x3    x4    x5    x6    x7    x8     x9   x10   x11
   <int> <int> <int> <dbl> <dbl> <dbl> <dbl> <int>  <dbl> <dbl> <dbl>
 1    36    27    71  8.10  3.34 11.4   81.5  3243  8.80   42.6  11.7
 2    35    23    72 11.1   3.14 11.0   78.8  4281  3.50   50.7  14.4
 3    44    29    74 10.4   3.21  9.80  81.6  4260  0.800  39.4  12.4
 4    47    45    79  6.50  3.41 11.1   77.5  3125 27.1    50.2  20.6
 5    43    35    77  7.60  3.44  9.60  84.6  6441 24.4    43.7  14.3
 6    53    45    80  7.70  3.45 10.2   66.8  3325 38.5    43.1  25.5
 7    43    30    74 10.9   3.23 12.1   83.9  4679  3.50   49.2  11.3
 8    45    30    73  9.30  3.29 10.6   86.0  2140  5.30   40.4  10.5
 9    36    24    70  9.00  3.31 10.5   83.2  6582  8.10   42.5  12.6
10    36    27    72  9.50  3.36 10.7   79.3  4213  6.70   41.0  13.2
# ... with 50 more rows, and 5 more variables: x12 <int>, x13 <int>,
#   x14 <int>, x15 <int>, y <dbl>
```

Here's a codebook:

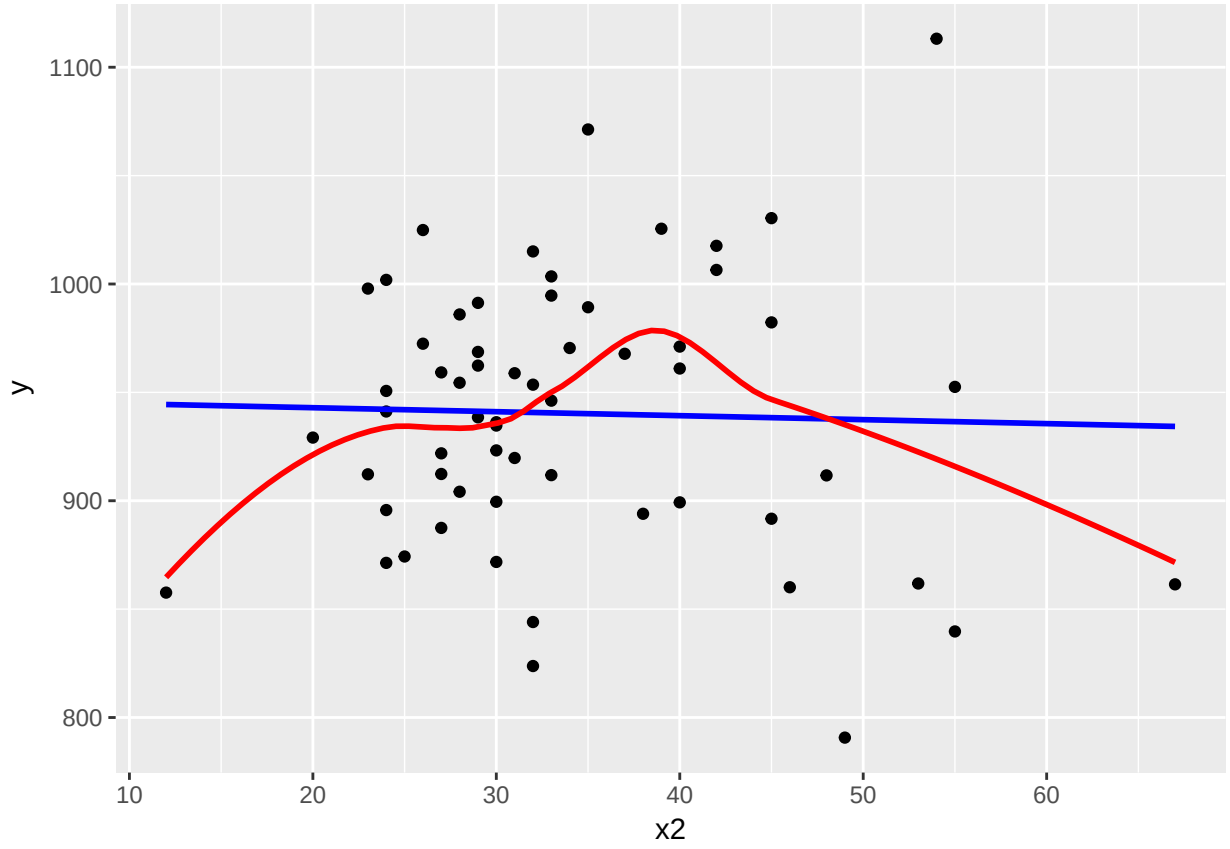
Variable	Description
`y`	Total Age Adjusted Mortality Rate
`x1`	Mean annual precipitation in inches
`x2`	Mean January temperature in degrees Fahrenheit
`x3`	Mean July temperature in degrees Fahrenheit
`x4`	Percent of 1960 SMSA population that is 65 years of age or over
`x5`	Population per household, 1960 SMSA
`x6`	Median school years completed for those over 25 in 1960 SMSA
`x7`	Percent of housing units that are found with facilities
`x8`	Population per square mile in urbanized area in 1960

Variable	Description
x9	Percent of 1960 urbanized area population that is non-white
x10	Percent employment in white-collar occupations in 1960 urbanized area
x11	Percent of families with income under 3; 000 in 1960 urbanized area
x12	Relative population potential of hydrocarbons, HC
x13	Relative pollution potential of oxides of nitrogen, NOx
x14	Relative pollution potential of sulfur dioxide, SO2
x15	Percent relative humidity, annual average at 1 p.m.

9.2 Fitting a straight line model to predict y from x2

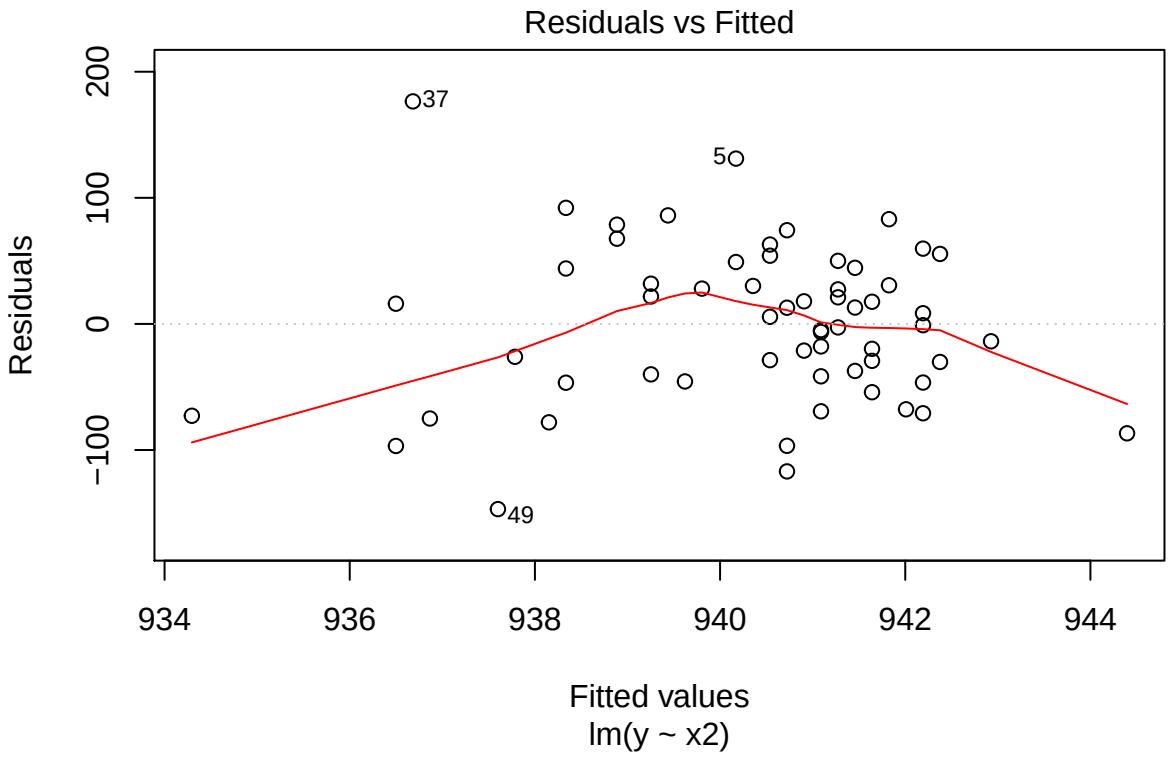
Consider the relationship between y, the age-adjusted mortality rate, and x2, the mean January temperature, across these 60 areas. I'll include both a linear model (in blue) and a loess smooth (in red.) Does the relationship appear to be linear?

```
ggplot(pollution, aes(x = x2, y = y)) +
    geom_point() +
    geom_smooth(method = "lm", col = "blue", se = F) +
    geom_smooth(method = "loess", col = "red", se = F)
```



Suppose we plot the residuals that emerge from the linear model shown in blue, above. Do we see a curve in a plot of residuals against fitted values?

```
plot(lm(y ~ x2, data = pollution), which = 1)
```



9.3 Quadratic polynomial model to predict y using x2

A polynomial in the variable `x` of degree D is a linear combination of the powers of `x` up to D.

For example:

- Linear: $y = \beta_0 + \beta_1 x$
- Quadratic: $y = \beta_0 + \beta_1 x + \beta_2 x^2$
- Cubic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$
- Quartic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$
- Quintic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5$

Fitting such a model creates a **polynomial regression*.

9.3.1 The raw quadratic model

Let's look at a **quadratic model** which predicts `y` using `x2` and the square of `x2`, so that our model is of the form:

$$y = \beta_0 + \beta_1 x_2 + \beta_2 x_2^2 + error$$

There are several ways to fit this exact model.

- One approach is to calculate the square of `x2` within our `pollution` data set, and then feed both `x2` and `x2squared` to `lm`.
- Another approach uses the I function within our `lm` to specify the use of both `x2` and its square.

- Yet another approach uses the `poly` function within our `lm`, which can be used to specify raw models including `x2` and `x2squared`.

```
pollution <- pollution %>%
    mutate(x2squared = x2^2)

mod2a <- lm(y ~ x2 + x2squared, data = pollution)
mod2b <- lm(y ~ x2 + I(x2^2), data = pollution)
mod2c <- lm(y ~ poly(x2, degree = 2, raw = TRUE), data = pollution)
```

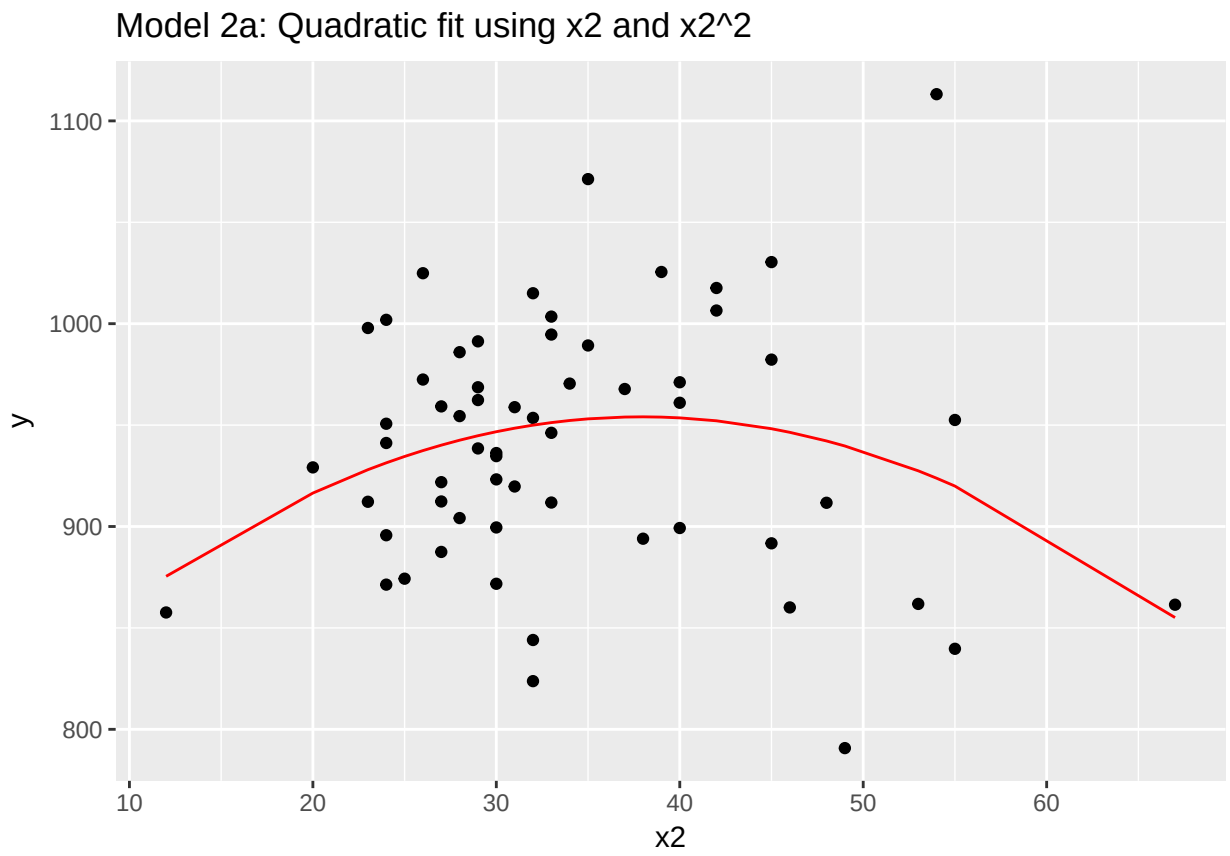
Each of these approaches produces the same model, as they are just different ways of expressing the same idea.

```
summary(mod2a)
```

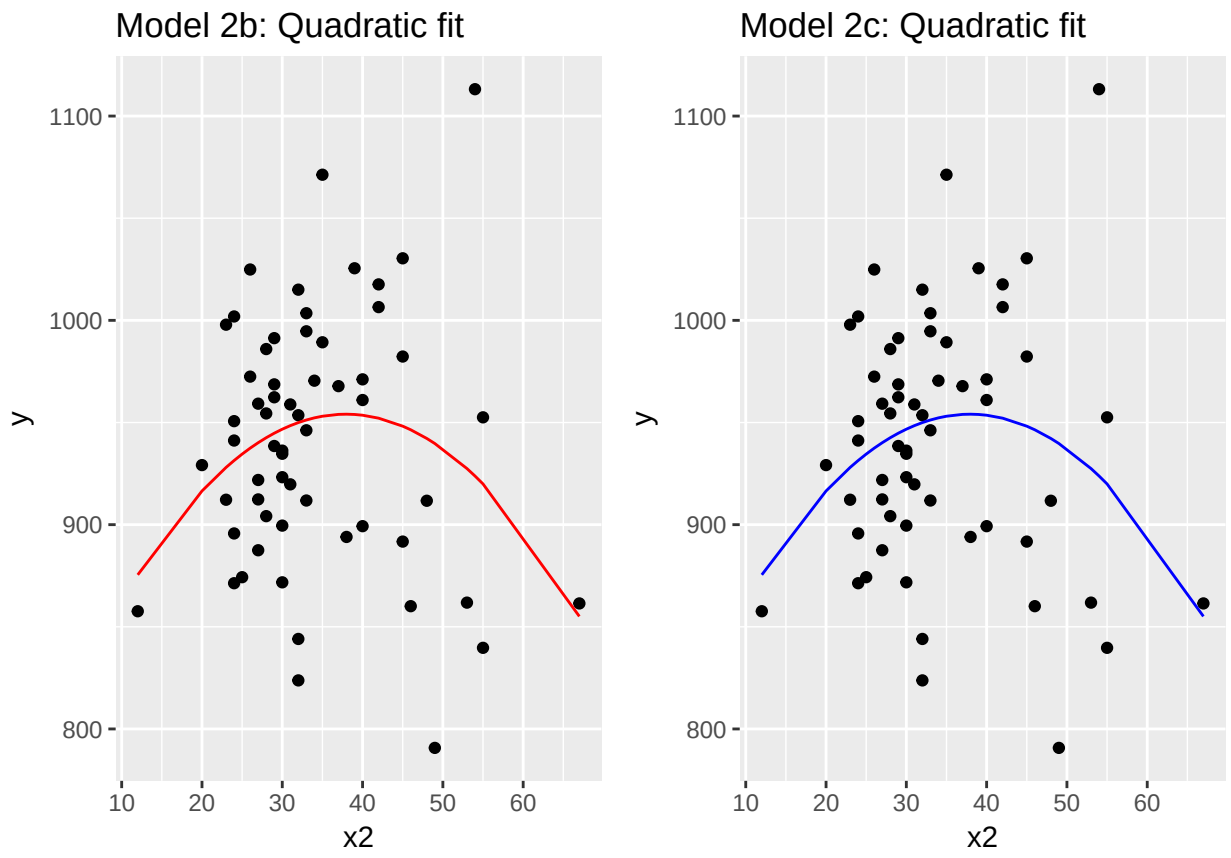
```
Call:
lm(formula = y ~ x2 + x2squared, data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-148.977  -38.651    6.889   35.312  189.346

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 785.77449   79.54086   9.879 5.87e-14 ***
x2            8.87640    4.27394   2.077   0.0423 *
x2squared    -0.11704    0.05429  -2.156   0.0353 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.83 on 57 degrees of freedom
Multiple R-squared:  0.07623,   Adjusted R-squared:  0.04382
F-statistic: 2.352 on 2 and 57 DF,  p-value: 0.1044
```

And if we plot the fitted values for this `mod2` using whatever approach you like, we get exactly the same result.

```
mod2a.aug <- augment(mod2a)
mod2a.aug$x2 <- pollution$x2

ggplot(pollution, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(data = mod2a.aug, aes(x = x2, y = .fitted),
              col = "red") +
    labs(title = "Model 2a: Quadratic fit using x2 and x2^2")
```




```
mod2b.aug <- augment(mod2b)
mod2b.aug$x2 <- pollution$x2

mod2c.aug <- augment(mod2c)
mod2c.aug$x2 <- pollution$x2

p1 <- ggplot(pollution, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(data = mod2b.aug, aes(x = x2, y = .fitted),
              col = "red") +
    labs(title = "Model 2b: Quadratic fit")

p2 <- ggplot(pollution, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(data = mod2c.aug, aes(x = x2, y = .fitted),
              col = "blue") +
    labs(title = "Model 2c: Quadratic fit")

gridExtra::grid.arrange(p1, p2, nrow = 1)
```



9.3.2 Raw quadratic fit after centering x2

Sometimes, we'll center (and perhaps rescale, too) the x2 variable before including it in a quadratic fit like this.

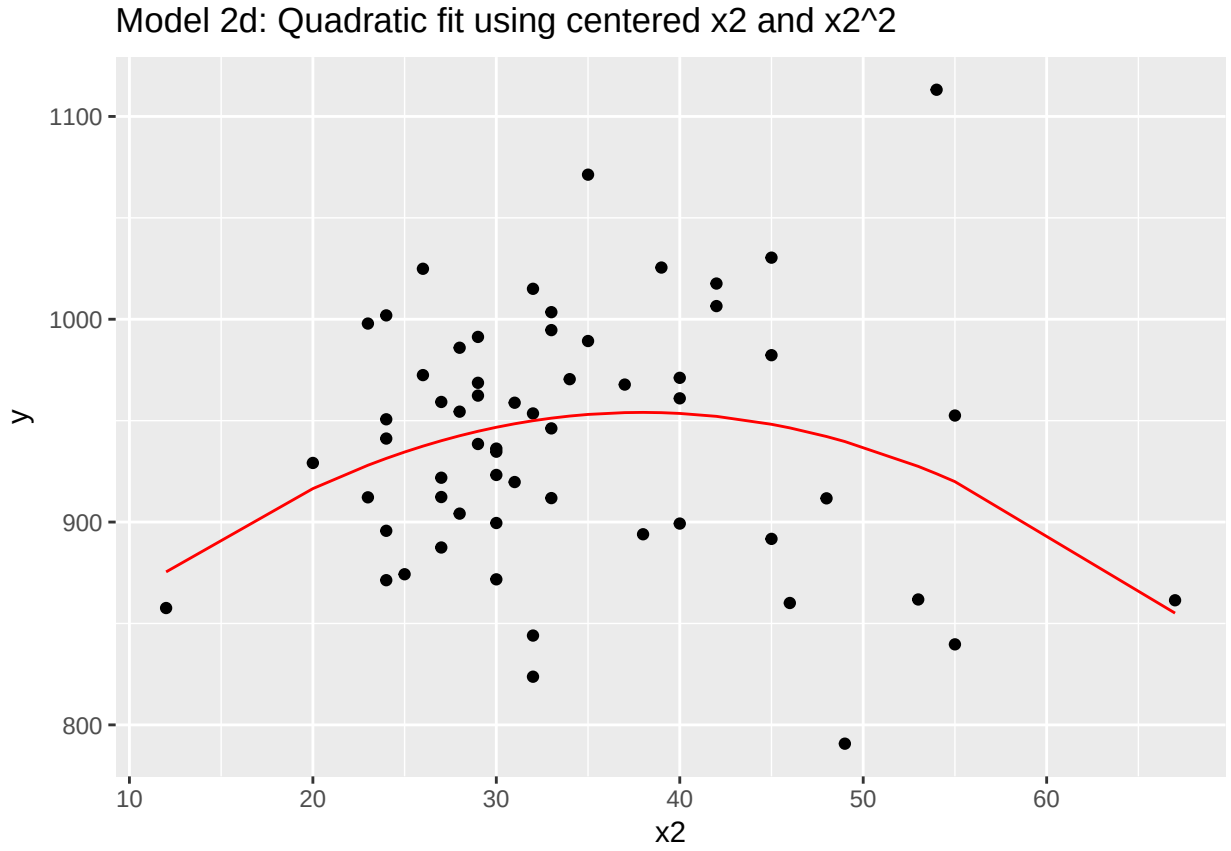
```
pollution <- pollution %>%
    mutate(x2_c = x2 - mean(x2))

mod2d <- lm(y ~ x2_c + I(x2_c^2), data = pollution)

summary(mod2d)
```

```
Call:
lm(formula = y ~ x2_c + I(x2_c^2), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-148.977  -38.651    6.889   35.312  189.346

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 952.25941    9.59896  99.204   <2e-16 ***
x2_c          0.92163    0.93237   0.988   0.3271
I(x2_c^2)    -0.11704    0.05429  -2.156   0.0353 *
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.83 on 57 degrees of freedom
Multiple R-squared:  0.07623,   Adjusted R-squared:  0.04382
F-statistic: 2.352 on 2 and 57 DF,  p-value: 0.1044
```

Note that this model looks very different, with the exception of the second order quadratic term. But, it produces the same fitted values as the models we fit previously.

```
mod2d.aug <- augment(mod2d)
mod2d.aug$x2 <- pollution$x2

ggplot(pollution, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(data = mod2d.aug, aes(x = x2, y = .fitted),
              col = "red") +
    labs(title = "Model 2d: Quadratic fit using centered x2 and x2^2")
```



Or, if you don't believe me yet, look at the four sets of fitted values another way.

```
mod2a.aug %>% skim(.fitted)
```

```
Skim summary statistics
 n obs: 60
 n variables: 10

Variable type: numeric
 variable missing complete  n   mean    sd    p0    p25 median    p75
```

```
  .fitted       0       60 60 940.36 17.18 855.1 936.72  945.6 950.29
   p100
 954.07
```

```
mod2b.aug %>% skim(.fitted)
```

```
Skim summary statistics
 n obs: 60
 n variables: 10

Variable type: numeric
 variable missing complete  n   mean    sd    p0    p25 median    p75
  .fitted       0       60 60 940.36 17.18 855.1 936.72  945.6 950.29
   p100
 954.07
```

```
mod2c.aug %>% skim(.fitted)
```

```
Skim summary statistics
 n obs: 60
 n variables: 10

Variable type: numeric
 variable missing complete  n   mean    sd    p0    p25 median    p75
  .fitted       0       60 60 940.36 17.18 855.1 936.72  945.6 950.29
   p100
 954.07
```

```
mod2d.aug %>% skim(.fitted)
```

```
Skim summary statistics
 n obs: 60
 n variables: 11

Variable type: numeric
 variable missing complete  n   mean    sd    p0    p25 median    p75
  .fitted       0       60 60 940.36 17.18 855.1 936.72  945.6 950.29
   p100
 954.07
```

9.4 Orthogonal Polynomials

Now, let's fit an orthogonal polynomial of degree 2 to predict `y` using `x2`.

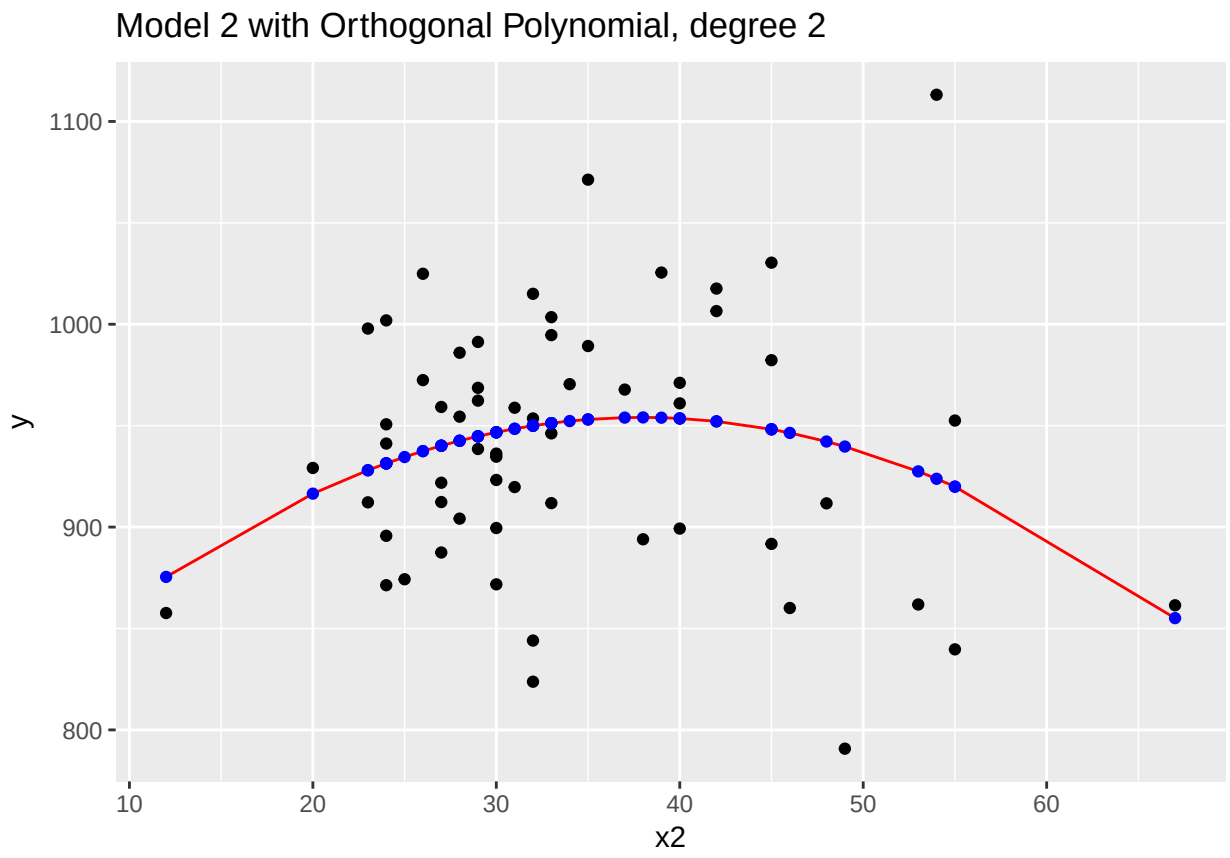
```
mod2_orth <- lm(y ~ poly(x2, 2), data = pollution)

summary(mod2_orth)
```

```

Call:
lm(formula = y ~ poly(x2, 2), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-148.977  -38.651    6.889   35.312  189.346
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   940.358      7.853 119.746   <2e-16 ***
poly(x2, 2)1  -14.345     60.829  -0.236   0.8144
poly(x2, 2)2 -131.142     60.829  -2.156   0.0353 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.83 on 57 degrees of freedom
Multiple R-squared:  0.07623,   Adjusted R-squared:  0.04382
F-statistic: 2.352 on 2 and 57 DF,  p-value: 0.1044
```

Now this looks very different in the equation, but, again, we can see that this produces exactly the same fitted values as our previous models, and the same model fit summaries. Is it, in fact, the same model? Here, we'll plot the fitted Model 2a in a red line, and this new Model 2 with Orthogonal Polynomials as blue points.

```
mod2orth.aug <- augment(mod2_orth)
mod2orth.aug$x2 <- pollution$x2

ggplot(pollution, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(data = mod2a.aug, aes(x = x2, y = .fitted),
              col = "red") +
    geom_point(data = mod2orth.aug, aes(x = x2, y = .fitted),
              col = "blue") +
    labs(title = "Model 2 with Orthogonal Polynomial, degree 2")
```



Yes, it is again the same model in terms of the predictions it makes for `y`.

By default, with `raw = FALSE`, the `poly()` function within a linear model computes what is called an **orthogonal polynomial**. An orthogonal polynomial sets up a model design matrix using the coding we've seen previously: `x2` and `x2^2` in our case, and then scales those columns so that each column is **orthogonal** to the previous ones. This eliminates the collinearity (correlation between predictors) and lets our t tests tell us whether the addition of any particular polynomial term improves the fit of the model over the lower orders.

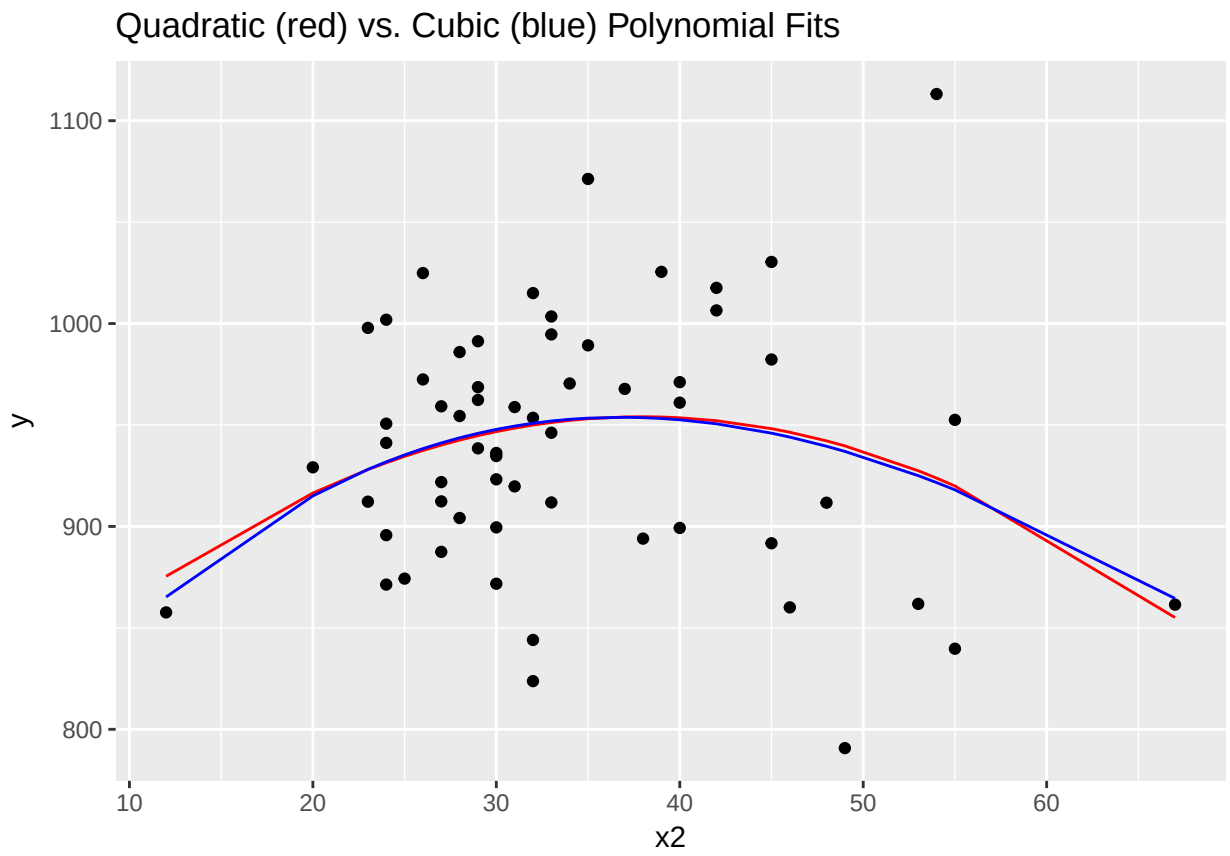
Would the addition of a cubic term help us much in predicting `y` from `x2`?

```
mod3 <- lm(y ~ poly(x2, 3), data = pollution)
summary(mod3)
```

```
Call:
lm(formula = y ~ poly(x2, 3), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-146.262  -39.679    5.569   35.984  191.536

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   940.358      7.917 118.772   <2e-16 ***
poly(x2, 3)1  -14.345     61.328  -0.234   0.8159
poly(x2, 3)2 -131.142     61.328  -2.138   0.0369 *
poly(x2, 3)3   16.918     61.328   0.276   0.7837
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 61.33 on 56 degrees of freedom
Multiple R-squared:  0.07748,   Adjusted R-squared:  0.02806
F-statistic: 1.568 on 3 and 56 DF,  p-value: 0.2073
```

It doesn't appear that the cubic term adds much here, if anything. The p value is not significant for the third degree polynomial, the summaries of fit quality aren't much improved, and as we can see from the plot below, the predictions don't actually change all that much.

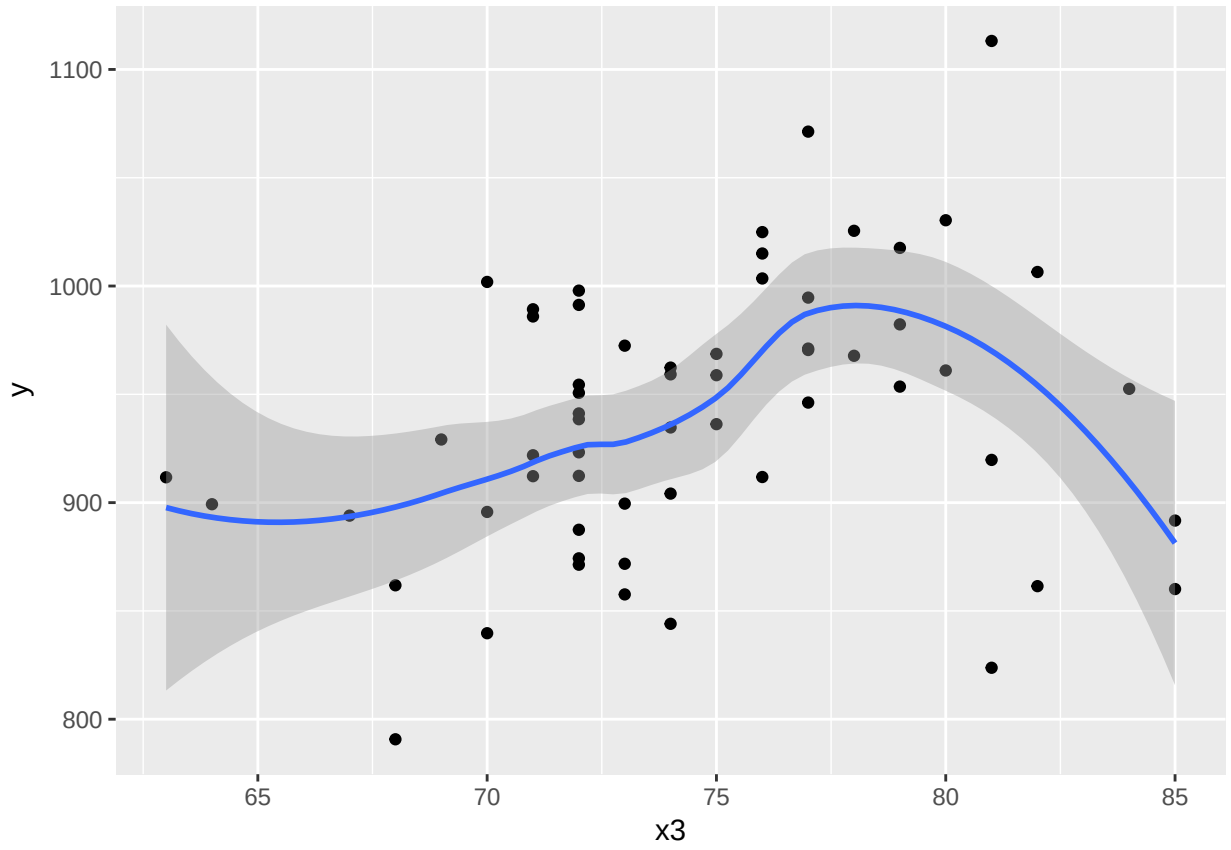
```
mod3.aug <- augment(mod3)
mod3.aug$x2 <- pollution$x2

ggplot(pollution, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(data = mod2orth.aug, aes(x = x2, y = .fitted),
              col = "red") +
    geom_line(data = mod3.aug, aes(x = x2, y = .fitted),
              col = "blue") +
    labs(title = "Quadratic (red) vs. Cubic (blue) Polynomial Fits")
```



9.5 Fit a cubic polynomial to predict y from x3

What if we consider another predictor instead? Let's look at x3, the Mean July temperature in degrees Fahrenheit. Here is the loess smooth.

```
ggplot(pollution, aes(x = x3, y = y)) +
    geom_point() +
    geom_smooth(method = "loess")
```



That looks pretty curvy - perhaps we need a more complex polynomial. We'll consider a linear model (`mod4_L`), a quadratic fit (`mod4_Q`) and a polynomial of degree 3: a **cubic** fit (`mod_4C`)

```
mod4_L <- lm(y ~ x3, data = pollution)
summary(mod4_L)
```

```
Call:
lm(formula = y ~ x3, data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-139.813  -34.341    4.271   38.197  149.587

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  670.529    123.140   5.445  1.1e-06 ***
x3             3.618      1.648   2.196   0.0321 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.29 on 58 degrees of freedom
Multiple R-squared:  0.07674,   Adjusted R-squared:  0.06082
F-statistic: 4.821 on 1 and 58 DF,  p-value: 0.03213
```

```
mod4_Q <- lm(y ~ poly(x3, 2), data = pollution)
summary(mod4_Q)
```

```
Call:
lm(formula = y ~ poly(x3, 2), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-132.004  -42.184    4.069   47.126  157.396

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   940.358      7.553 124.503   <2e-16 ***
poly(x3, 2)1  132.364     58.504   2.262   0.0275 *
poly(x3, 2)2 -125.270     58.504  -2.141   0.0365 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 58.5 on 57 degrees of freedom
Multiple R-squared:  0.1455,    Adjusted R-squared:  0.1155
F-statistic: 4.852 on 2 and 57 DF,  p-value: 0.01133
```

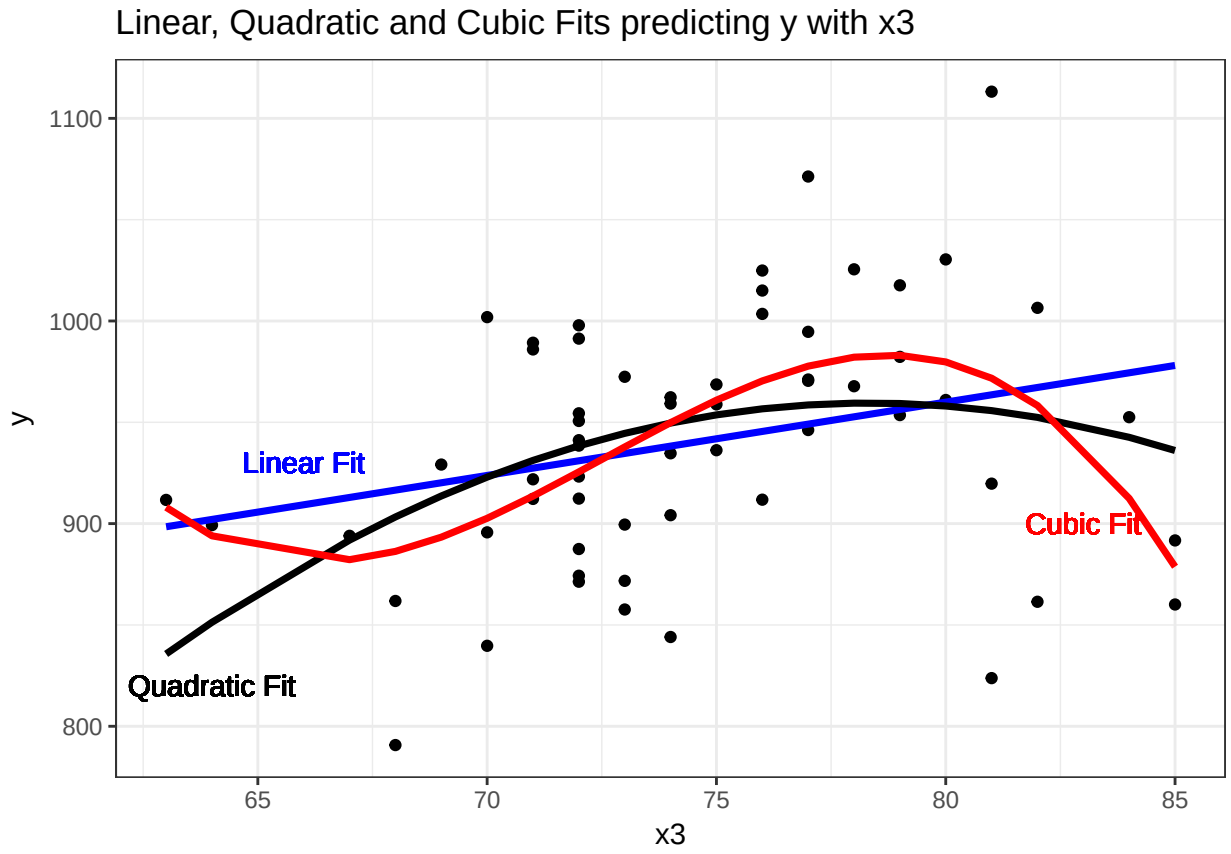
```
mod4_C <- lm(y ~ poly(x3, 3), data = pollution)
summary(mod4_C)
```

```
Call:
lm(formula = y ~ poly(x3, 3), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-148.004  -29.998    1.441   34.579  141.396

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   940.358      7.065 133.095  < 2e-16 ***
poly(x3, 3)1  132.364     54.728   2.419  0.01886 *
poly(x3, 3)2 -125.270     54.728  -2.289  0.02588 *
poly(x3, 3)3 -165.439     54.728  -3.023  0.00377 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 54.73 on 56 degrees of freedom
Multiple R-squared:  0.2654,    Adjusted R-squared:  0.226
F-statistic: 6.742 on 3 and 56 DF,  p-value: 0.0005799
```

It looks like the cubic polynomial term is of some real importance here. Do the linear, quadratic and cubic model fitted values look different?

```
mod4_L.aug <- augment(mod4_L)
mod4_L.aug$x3 <- pollution$x3

mod4_Q.aug <- augment(mod4_Q)
mod4_Q.aug$x3 <- pollution$x3

mod4_C.aug <- augment(mod4_C)
mod4_C.aug$x3 <- pollution$x3
```

```
ggplot(pollution, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(data = mod4_L.aug, aes(x = x3, y = .fitted),
              col = "blue", size = 1.25) +
    geom_line(data = mod4_Q.aug, aes(x = x3, y = .fitted),
              col = "black", size = 1.25) +
    geom_line(data = mod4_C.aug, aes(x = x3, y = .fitted),
              col = "red", size = 1.25) +
    geom_text(x = 66, y = 930, label = "Linear Fit", col = "blue") +
    geom_text(x = 64, y = 820, label = "Quadratic Fit", col = "black") +
    geom_text(x = 83, y = 900, label = "Cubic Fit", col = "red") +
    labs(title = "Linear, Quadratic and Cubic Fits predicting y with x3") +
    theme_bw()
```



9.6 Fitting a restricted cubic spline in a linear regression

- A **linear spline** is a continuous function formed by connecting points (called **knots** of the spline) by line segments.
- A **restricted cubic spline** is a way to build highly complicated curves into a regression equation in a fairly easily structured way.
- A restricted cubic spline is a series of polynomial functions joined together at the knots.
  - Such a spline gives us a way to flexibly account for non-linearity without over-fitting the model.
  - Restricted cubic splines can fit many different types of non-linearities.
  - Specifying the number of knots is all you need to do in R to get a reasonable result from a

restricted cubic spline.

The most common choices are 3, 4, or 5 knots. Each additional knot adds to the non-linearity, and spends an additional degree of freedom:

- 3 Knots, 2 degrees of freedom, allows the curve to "bend" once.
- 4 Knots, 3 degrees of freedom, lets the curve "bend" twice.
- 5 Knots, 4 degrees of freedom, lets the curve "bend" three times.

For most applications, three to five knots strike a nice balance between complicating the model needlessly and fitting data pleasingly. Let's consider a restricted cubic spline model for our `y` based on `x3` again, but now with:

- in `mod5a`, 3 knots,
- in `mod5b`, 4 knots, and
- in `mod5c`, 5 knots

```
mod5a_rcs <- lm(y ~ rcs(x3, 3), data = pollution)
mod5b_rcs <- lm(y ~ rcs(x3, 4), data = pollution)
mod5c_rcs <- lm(y ~ rcs(x3, 5), data = pollution)
```

Here, for instance, is the summary of the 5-knot model:

```
summary(mod5c_rcs)
```

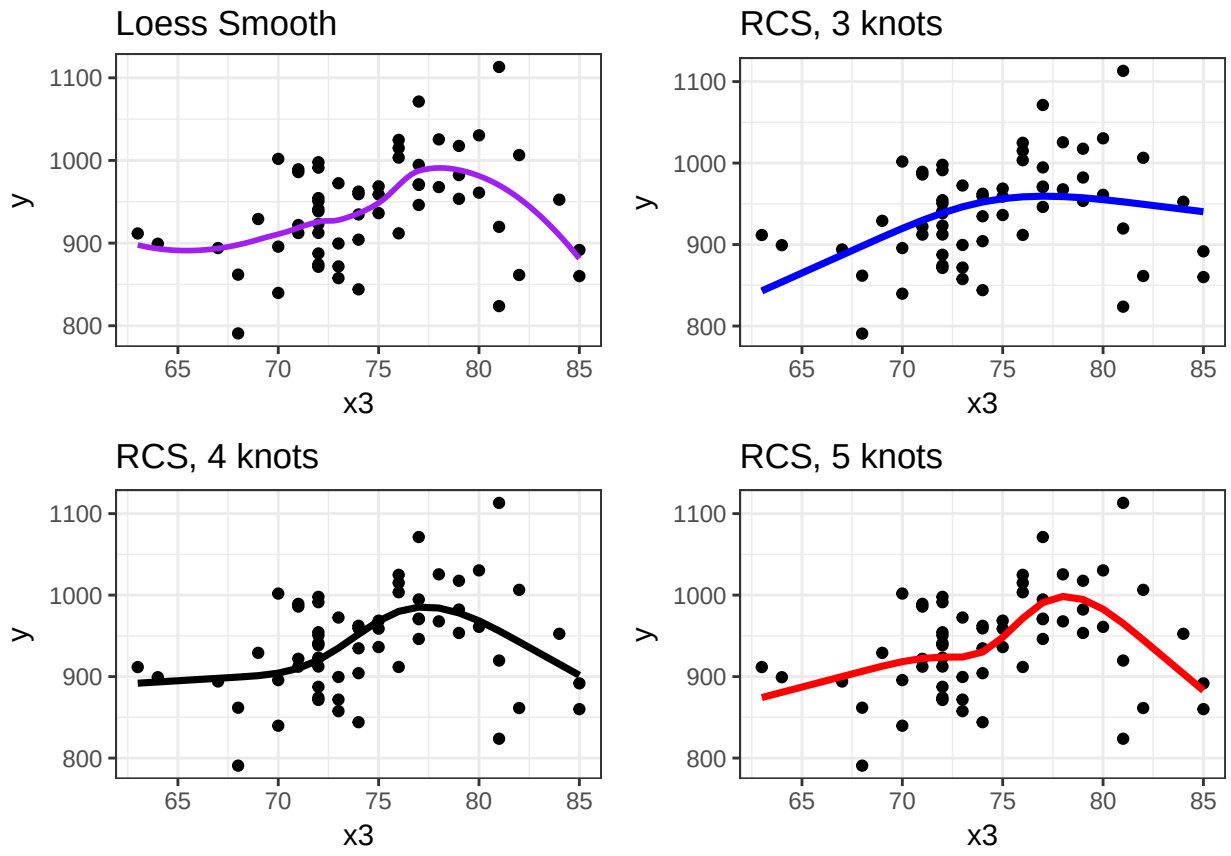
```

Call:
lm(formula = y ~ rcs(x3, 5), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max 
-141.522  -32.009    1.674   31.971  147.878 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      468.113    396.319   1.181    0.243
rcs(x3, 5)x3       6.447      5.749   1.121    0.267
rcs(x3, 5)x3'    -25.633     46.810  -0.548    0.586
rcs(x3, 5)x3''   323.137    293.065   1.103    0.275
rcs(x3, 5)x3''' -612.578    396.270  -1.546    0.128

Residual standard error: 54.35 on 55 degrees of freedom
Multiple R-squared:  0.2883,    Adjusted R-squared:  0.2366 
F-statistic: 5.571 on 4 and 55 DF,  p-value: 0.0007734
```

We'll begin by storing the fitted values from these three models and other summaries, for plotting.

```
mod5a.aug <- augment(mod5a_rcs)
mod5a.aug$x3 <- pollution$x3

mod5b.aug <- augment(mod5b_rcs)
mod5b.aug$x3 <- pollution$x3

mod5c.aug <- augment(mod5c_rcs)
mod5c.aug$x3 <- pollution$x3
```

```
p2 <- ggplot(pollution, aes(x = x3, y = y)) +
    geom_point() +
    geom_smooth(method = "loess", col = "purple", se = F) +
```

```
    labs(title = "Loess Smooth") +
    theme_bw()

p3 <- ggplot(pollution, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(data = mod5a.aug, aes(x = x3, y = .fitted),
              col = "blue", size = 1.25) +
    labs(title = "RCS, 3 knots") +
    theme_bw()

p4 <- ggplot(pollution, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(data = mod5b.aug, aes(x = x3, y = .fitted),
              col = "black", size = 1.25) +
    labs(title = "RCS, 4 knots") +
    theme_bw()

p5 <- ggplot(pollution, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(data = mod5c.aug, aes(x = x3, y = .fitted),
              col = "red", size = 1.25) +
    labs(title = "RCS, 5 knots") +
    theme_bw()

gridExtra::grid.arrange(p2, p3, p4, p5, nrow = 2)
```



Does it seem like the fit improves markedly (perhaps approaching the loess smooth result) as we increase the number of knots?

```
anova(mod5a_rcs, mod5b_rcs, mod5c_rcs)
```

```
Analysis of Variance Table

Model 1: y ~ rcs(x3, 3)
Model 2: y ~ rcs(x3, 4)
Model 3: y ~ rcs(x3, 5)
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1     57 194935
2     56 171448  1   23486.9 7.9503 0.006672 **
3     55 162481  1    8967.2 3.0354 0.087057 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on an ANOVA comparison, the fourth knot adds significant predictive value (p = 0.0067), but the fifth knot is borderline (p = 0.0871). From the `glance` function in the `broom` package, we can also look at some key summaries.

```
glance(mod5a_rcs)
```

```
  r.squared adj.r.squared    sigma statistic    p.value df    logLik
1  0.146184     0.1162256 58.48006  4.879558 0.01106323  3 -327.7187
       AIC      BIC deviance df.residual
1 663.4373 671.8147 194935.3          57
```

```
glance(mod5b_rcs)
```

```
  r.squared adj.r.squared    sigma statistic   p.value df    logLik
1 0.2490566     0.2088274 55.33153  6.190953 0.0010423  4 -323.8671
       AIC      BIC deviance df.residual
1 657.7342 668.2059 171448.4          56
```

```
glance(mod5c_rcs)
```

```
  r.squared adj.r.squared    sigma statistic      p.value df    logLik
1 0.2883327     0.2365751 54.35259  5.570826 0.0007734418  5 -322.2555
      AIC      BIC deviance df.residual
1 656.511 669.0771 162481.2          55
```

Model	Knots	R^2	Adj. R^2	AIC	BIC
5a	3	0.146	0.116	663.4	671.8
5b	4	0.249	0.209	657.7	**668.2**
5c	5	0.288	**0.237**	**656.5**	669.1

Within our sample, the five-knot RCS outperforms the 3- and 4-knot versions on adjusted R^2 and AIC (barely) and does a little worse than the 4-knot RCS on BIC.

Of course, we could also use the cross-validation methods we've developed for other linear regressions to assess predictive capacity of these models. I'll skip that for now.

To see the values of `x3` where the splines place their knots, we can use the `attributes` function.

```
attributes(rcs(pollution$x3, 5))
```

```
$dim
[1] 60  4
```

```
$dimnames
$dimnames[[1]]
NULL

$dimnames[[2]]
[1] "pollution"    "pollution'"   "pollution''"  "pollution'''"


$class
[1] "rms"

$name
[1] "pollution"

$label
[1] "pollution"

$assume
[1] "rcspline"

$assume.code
[1] 4

$parms
[1] 68 72 74 77 82

$nonlinear
[1] FALSE  TRUE  TRUE  TRUE

$colnames
[1] "pollution"    "pollution'"   "pollution''"  "pollution'''"
```

The knots in this particular 5-knot spline are placed by the computer at 68, 72, 74, 77 and 82, it seems.

There are two kinds of Multivariate Regression Models

1. [Prediction] Those that are built so that we can make accurate predictions.
2. [Explanatory] Those that are built to help understand underlying phenomena.

While those two notions overlap considerably, they do imply different things about how we strategize about model-building and model assessment. Harrell's primary concern is effective use of the available data for **prediction** - this implies some things that will be different from what we've seen in the past.

Harrell refers to multivariable regression modeling strategy as the process of **spending degrees of freedom**. The main job in strategizing about multivariate modeling is to

1. Decide the number of degrees of freedom that can be spent
2. Decide where to spend them
3. Spend them, wisely.

What this means is essentially linked to making decisions about predictor complexity, both in terms of how many predictors will be included in the regression model, and about how we'll include those predictors.

9.7 "Spending" Degrees of Freedom

- "Spending" df includes

  - fitting parameter estimates in models, or
  - examining figures built using the outcome variable Y that tell you how to model the predictors.

If you use a scatterplot of Y vs. X or the residuals of the Y-X regression model vs. X to decide whether a linear model is appropriate, then how many degrees of freedom have you actually spent?

Grambsch and O'Brien conclude that if you wish to preserve the key statistical properties of the various estimation and fitting procedures used in building a model, you can't retrieve these degrees of freedom once they have been spent.

9.7.1 Overfitting and Limits on the # of Predictors

Suppose you have a total sample size of n observations, then you really shouldn't be thinking about estimating more than $n/15$ regression coefficients, at the most.

- If k is the number of parameters in a full model containing all candidate predictors for a stepwise analysis, then k should be no greater than $n/15$.
- k should include all variables screened for association with the response, including interaction terms.

So if you have 97 observations in your data, then you can probably just barely justify the use of a stepwise analysis using the main effects alone of 5 candidate variables (with one additional DF for the intercept term.)

Harrell (2001) also mentions that if you have a **narrowly distributed** predictor, without a lot of variation to work with, then an even larger sample size n should be required. See Vittinghoff et al. (2012), Section 10.3 for more details.

9.7.2 The Importance of Collinearity

> Collinearity denotes correlation between predictors high enough to degrade the precision of the regression coefficient estimates substantially for some or all of the correlated predictors

- Vittinghoff et al. (2012), section 10.4.1
- Can one predictor in a model be predicted well using the other predictors in the model?
  - Strong correlations (for instance, $r \geq 0.8$) are especially troublesome.
- Effects of collinearity
  - decreases precision, in the sense of increasing the standard errors of the parameter estimates
  - decreases power
  - increases the difficulty of interpreting individual predictor effects
  - overall F test is significant, but individual t tests may not be

Suppose we want to assess whether variable X_j is collinear with the other predictors in a model. We run a regression predicting X_j using the other predictors, and obtain the R^2. The VIF is defined as 1 / (1 - this R^2), and we usually interpret VIFs above 5 as indicating a serious multicollinearity problem (i.e. R^2 values for this predictor of 0.8 and above would thus concern us.)

```
vif(lm(y ~ x1 + x2 + x3 + x4 + x5 + x6, data = pollution))
```

```
      x1       x2       x3       x4       x5       x6
2.238862 2.058731 2.153044 4.174448 3.447399 1.792996
```

Occasionally, you'll see the inverse of VIF reported, and this is called *tolerance*.

- tolerance = 1 / VIF

9.7.3 Collinearity in an Explanatory Model

- When we are attempting to **identify multiple independent predictors** (the explanatory model approach), then we will need to choose between collinear variables
  - options suggested by Vittinghoff et al. (2012), p. 422, include choosing on the basis of plausibility as a causal factor,
  - choosing the variable that has higher data quality (is measured more accurately or has fewer missing values.)
  - Often, we choose to include a variable that is statistically significant as a predictor, and drop others, should we be so lucky.
- Larger effects, especially if they are associated with predictors that have minimal correlation with the other predictors under study, cause less trouble in terms of potential violation of the $n/15$ rule for what constitutes a reasonable number of predictors.

9.7.4 Collinearity in a Prediction Model

- If we are primarily building a **prediction model** for which inference on the individual predictors is not of interest, then it is totally reasonable to use both predictors in the model, if doing so reduces prediction error.
  - Collinearity doesn't affect predictions in our model development sample.
  - Collinearity doesn't affect predictions on new data so long as the new data have similar relationships between predictors.
  - If our key predictor is correlated strongly with a confounder, then if the predictor remains significant after adjustment for the confounder, then this suggests a meaningful independent effect.
    * If the effects of the predictor are clearly confounded by the adjustment variable, we again have a clear result.
    * If neither is statistically significant after adjustment, the data may be inadequate.
  - If the collinearity is between adjustment variables, but doesn't involve the key predictor, then inclusion of the collinear variables is unlikely to cause substantial problems.

9.8 Spending DF on Non-Linearity: The Spearman ρ^2 Plot

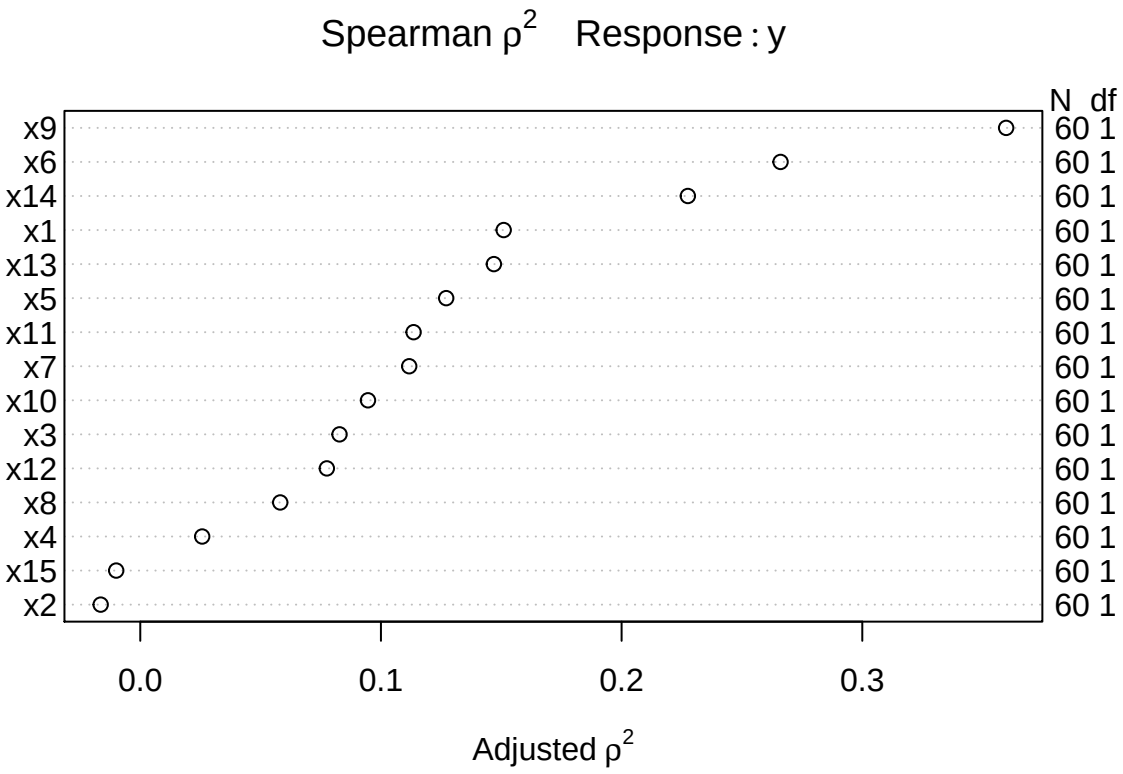
We need a flexible approach to assessing non-linearity and fitting models with non-linear predictors. This will lead us to a measure of what Harrell (2001) calls **potential predictive punch** which hides the true form of the regression from the analyst so as to preserve statistical properties, but that lets us make sensible decisions about whether a predictor should be included in a model, and the number of parameters (degrees of freedom, essentially) we are willing to devote to it.

What if we want to consider where best to spend our degrees of freedom on non-linear predictor terms, like interactions, polynomial functions or curved splines to represent our input data? The approach we'll find useful in the largest variety of settings is a combination of

1. a rank correlation assessment of potential predictive punch (using a Spearman ρ^2 plot, available in the `Hmisc` package), followed by
2. the application of restricted cubic splines to fit and assess models.

Suppose, for instance, that we want to create a model for `y` using some combination of linear and non-linear terms drawn from the complete set of 15 predictors available in the `pollution` data. I'd begin by running a Spearman ρ^2 plot:

```
plot(Hmisc::spearman2(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 +
                          x8 + x9 + x10 + x11 + x12 + x13 +
                          x14 + x15, data = pollution))
```



The variable with the largest adjusted squared Spearman ρ statistic in this setting is `x9`, followed by `x6` and `x14`. With only 60 observations, we might well want to restrict ourselves to a very small model. What the Spearman plot suggests is that we focus any non-linear terms on `x9` first, and then perhaps `x6` and `x14` as they have some potential predictive power. It may or may not work out that the non-linear terms are productive.

9.8.1 Fitting a Big Model to the `pollution` data

So, one possible model built in reaction this plot might be to fit:

- a restricted cubic spline with 5 knots on x9
- a restricted cubic spline with 3 knots on x6
- and a quadratic polynomial on x14
- plus a linear fit to x1 and x13

That's way more degrees of freedom (4 for `x9`, 2 for `x6`, 2 for `x14` and 1 each for x1 and x13 makes a total of 10 without the intercept term) than we can really justify with a sample of 60 observations. But let's see what happens.

```
mod_big <- lm(y ~ rcs(x9, 5) + rcs(x6, 3) + poly(x14, 2) + x1 + x13, data = pollution)

anova(mod_big)
```

```
Analysis of Variance Table

Response: y
             Df Sum Sq Mean Sq F value    Pr(>F)
rcs(x9, 5)    4 100164 25040.9 17.8482 4.229e-09 ***
```

```
rcs(x6, 3)    2  38306 19152.8 13.6513 1.939e-05 ***
poly(x14, 2)  2  15595  7797.7  5.5579  0.006677 **
x1            1   4787  4787.3  3.4122  0.070759 .
x13           1    712   711.9  0.5074  0.479635
Residuals    49  68747  1403.0
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This `anova` suggests that we have at least some predictive value in each spline (`x9` and `x6`) and some additional value in `x14`, although it's not as clear that the linear terms (`x1` and `x13`) did much good.

9.8.2 Limitations of `lm` for fitting complex linear regression models

We can certainly assess this big, complex model using `lm` in comparison to other models:

- with in-sample summary statistics like adjusted R^2, AIC and BIC,
- we can assess its assumptions with residual plots, and
- we can also compare out-of-sample predictive quality through cross-validation,

But to really delve into the details of how well this complex model works, and to help plot what is actually being fit, we'll probably want to fit the model using an alternative method for fitting linear models, called `ols`, from the `rms` package developed by Frank Harrell and colleagues. That will be the focus of our next chapter.

Chapter 10

Using `ols` from the `rms` package to fit linear models

At the end of the previous chapter, we had fit a model to the `pollution` data that predicted our outcome `y` = Age-Adjusted Mortality Rate, using:

- a restricted cubic spline with 5 knots on `x9`
- a restricted cubic spline with 3 knots on `x6`
- a polynomial in 2 degrees on `x14`
- linear terms for `x1` and `x13`

but this model was hard to evaluate in some ways. Now, instead of using `lm` to fit this model, we'll use a new function called `ols` from the `rms` package developed by Frank Harrell and colleagues, in part to support ideas developed in Harrell (2001) for clinical prediction models.

10.1 Fitting a model with `ols`

We will use the `datadist` approach when fitting a linear model with `ols` from the `rms` package, so as to store additional important elements of the model fit.

```
library(rms)

d <- datadist(pollution)
options(datadist = "d")
```

Next, we'll fit the model using `ols` and place its results in `newmod`.

```
newmod <- ols(y ~ rcs(x9, 5) + rcs(x6, 3) + pol(x14, 2) +
                  x1 + x13,
              data = pollution, x = TRUE, y = TRUE)
newmod
```

```
Linear Regression Model

 ols(formula = y ~ rcs(x9, 5) + rcs(x6, 3) + pol(x14, 2) + x1 +
     x13, data = pollution, x = TRUE, y = TRUE)

                 Model Likelihood     Discrimination
                    Ratio Test           Indexes
 Obs       60    LR chi2     72.02    R2       0.699
```

```
sigma37.4566    d.f.           10    R2 adj   0.637
d.f.      49    Pr(> chi2) 0.0000    g       58.961

Residuals

    Min      1Q  Median      3Q     Max
-86.189 -18.554  -1.799  18.645 104.307


          Coef      S.E.     t     Pr(>|t|)
Intercept  796.2658 162.3269  4.91 <0.0001
x9          -2.6328   6.3504 -0.41 0.6803
x9'        121.4651 124.4827  0.98 0.3340
x9''      -219.8025 227.6775 -0.97 0.3391
x9'''      151.5700 171.3867  0.88 0.3808
x6           7.6817  15.5230  0.49 0.6229
x6'        -29.4388  18.0531 -1.63 0.1094
x14          0.5652   0.2547  2.22 0.0311
x14^2       -0.0010   0.0010 -0.96 0.3407
x1           1.0717   0.7317  1.46 0.1494
x13         -0.1028   0.1443 -0.71 0.4796
```

Some of the advantages and disadvantages of fitting linear regression models with `ols` or `lm` will reveal themselves over time. For now, one advantage for `ols` is that the entire variance-covariance matrix is saved. Most of the time, there will be some value to considering both `ols` and `lm` approaches.

Most of this output should be familiar, but a few pieces are different.

10.1.1 The Model Likelihood Ratio Test

The **Model Likelihood Ratio Test** compares `newmod` to the null model with only an intercept term. It is a goodness-of-fit test that we'll use in several types of model settings this semester.

- In many settings, the logarithm of the likelihood ratio, multiplied by -2, yields a value which can be compared to a χ^2 distribution. So here, the value 72.02 is -2(log likelihood), and is compared to a χ^2 distribution with 10 degrees of freedom. We reject the null hypothesis that `newmod` is no better than the null model, and conclude instead that at least of these predictors adds statistically significant value.
  - For `ols`, interpret the model likelihood ratio test like the global (ANOVA) F test in `lm`.
  - The likelihood function is the probability of observing our data under the specified model.
  - We can compare two nested models by evaluating the difference in their likelihood ratios and degrees of freedom, then comparing the result to a χ^2 distribution.

10.1.2 The g statistic

The **g statistic** is new and is referred to as the g-index. it's based on Gini's mean difference and is purported to be a robust and highly efficient measure of variation.

- Here, g = 58.9, which implies that if you randomly select two of the 60 areas included in the model, the average difference in predicted `y` (Age-Adjusted Mortality Rate) using this model will be 58.9.
  - Technically, g is Gini's mean difference of the predicted values.

10.2 ANOVA for an ols model

One advantage of the ols approach is that when you apply an anova to it, it separates out the linear and non-linear components of restricted cubic splines and polynomial terms (as well as product terms, if your model includes them.)

```
anova(newmod)
```

```
                Analysis of Variance          Response: y

 Factor          d.f. Partial SS  MS         F     P
 x9               4    35219.7647  8804.9412  6.28 0.0004
  Nonlinear       3     1339.3081   446.4360  0.32 0.8121
 x6               2     9367.6008  4683.8004  3.34 0.0437
  Nonlinear       1     3730.7388  3730.7388  2.66 0.1094
 x14              2    18679.6957  9339.8478  6.66 0.0028
  Nonlinear       1     1298.7625  1298.7625  0.93 0.3407
 x1               1     3009.1829  3009.1829  2.14 0.1494
 x13              1      711.9108   711.9108  0.51 0.4796
 TOTAL NONLINEAR  5     6656.1824  1331.2365  0.95 0.4582
 REGRESSION      10   159563.8285 15956.3829 11.37 <.0001
 ERROR           49    68746.8004  1402.9959
```

Unlike the anova approach in lm, in ols ANOVA, *partial* F tests are presented - each predictor is assessed as "last predictor in" much like the usual t tests in lm. In essence, the partial sums of squares and F tests here describe the marginal impact of removing each covariate from newmod.

We conclude that the non-linear parts of x9 and x6 and x14 combined don't seem to add much value, but that overall, x9, x6 and x14 seem to be valuable. So it must be the linear parts of those variables within our model that are doing the lion's share of the work.

10.3 Effect Estimates

A particularly useful thing to get out of the ols approach that is not as easily available in lm (without recoding or standardizing our predictors) is a summary of the effects of each predictor in an interesting scale.

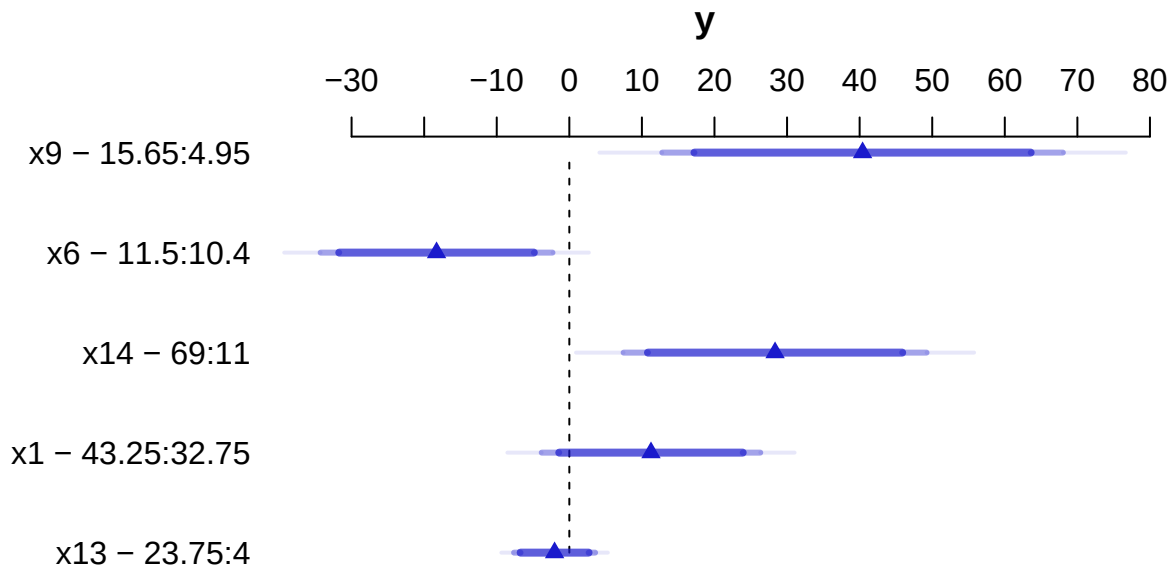
```
summary(newmod)
```

```
             Effects              Response : y

 Factor Low   High  Diff. Effect   S.E.    Lower 0.95 Upper 0.95
 x9      4.95 15.65 10.70  40.4060 14.0790  12.1120   68.6990
 x6     10.40 11.50  1.10 -18.2930  8.1499 -34.6710   -1.9153
 x14    11.00 69.00 58.00  28.3480 10.6480   6.9503   49.7460
 x1     32.75 43.25 10.50  11.2520  7.6833  -4.1878   26.6930
 x13     4.00 23.75 19.75  -2.0303  2.8502  -7.7579    3.6973
```

This "effects summary" shows the effect on y of moving from the 25th to the 75th percentile of each variable (along with a standard error and 95% confidence interval) while holding the other variable at the level specified at the bottom of the output.

The most useful way to look at this sort of analysis is often a plot.

```
plot(summary(newmod))
```



For `x9` note from the `summary` above that the 25th percentile is 4.95 and the 75th is 15.65. Our conclusion is that the estimated effect of moving `x9` from 4.95 to 15.65 is an increase of 40.4 on `y`, with a 95% CI of (12.1, 68.7).

For a categorical variable, the low level is shown first and then the high level.

The plot shows the point estimate (arrow head) and then the 90% (narrowest bar), 95% (middle bar) and 99% (widest bar in lightest color) confidence intervals for each predictor's effect.

- It's easier to distinguish this in the `x9` plot than the one for `x13`.
- Remember that what is being compared is the first value to the second value's impact on the outcome, with other predictors held constant.

10.3.1 Simultaneous Confidence Intervals

These confidence intervals make no effort to deal with the multiple comparisons problem, but just fit individual 95% (or whatever level you choose) confidence intervals for each predictor. The natural alternative is to make an adjustment for multiple comparisons in fitting the confidence intervals, so that the set of (in this case, 2) confidence intervals for effect sizes has a family-wise 95% confidence level. You'll note that the effect estimates and standard errors are unchanged, but the confidence limits are a bit wider.

```
summary(newmod, conf.type=c('simultaneous'))
```

```
             Effects              Response : y

 Factor Low   High  Diff. Effect   S.E.    Lower 0.95 Upper 0.95
 x9      4.95 15.65 10.70  40.4060 14.0790   3.12280  77.6890
 x6     10.40 11.50  1.10 -18.2930  8.1499 -39.87400   3.2882
```

```
 x14    11.00 69.00 58.00  28.3480 10.6480   0.15192  56.5440
 x1     32.75 43.25 10.50  11.2520  7.6833  -9.09340  31.5980
 x13     4.00 23.75 19.75  -2.0303  2.8502  -9.57760   5.5171
```

Remember that if you're looking for the usual `lm` summary for an `ols` object, use `summary.lm`, and that the `display` function from `arm` does not recognize `ols` objects.

10.4 The `Predict` function for an `ols` model

The `Predict` function is very flexible, and can be used to produce individual or simultaneous confidence limits.

```
Predict(newmod, x9 = 12, x6 = 12, x14 = 40, x1 = 40, x13 = 20) # individual limits
```

```
  x9 x6 x14 x1 x13     yhat    lower   upper
1 12 12  40 40  20 923.0982 893.0984 953.098

Response variable (y): y

Limits are 0.95 confidence limits
```

```
Predict(newmod, x9 = 5:15) # individual limits
```

```
   x9    x6 x14 x1 x13     yhat    lower    upper
1   5 11.05  30 38   9 913.7392 889.4802 937.9983
2   6 11.05  30 38   9 916.3490 892.0082 940.6897
3   7 11.05  30 38   9 921.3093 898.9657 943.6529
4   8 11.05  30 38   9 927.6464 907.0355 948.2574
5   9 11.05  30 38   9 934.3853 913.3761 955.3946
6  10 11.05  30 38   9 940.5510 917.8371 963.2648
7  11 11.05  30 38   9 945.2225 921.9971 968.4479
8  12 11.05  30 38   9 948.2885 926.4576 970.1194
9  13 11.05  30 38   9 950.2608 930.3003 970.2213
10 14 11.05  30 38   9 951.6671 932.2370 971.0971
11 15 11.05  30 38   9 953.0342 932.1662 973.9021

Response variable (y): y

Adjust to: x6=11.05 x14=30 x1=38 x13=9

Limits are 0.95 confidence limits
```

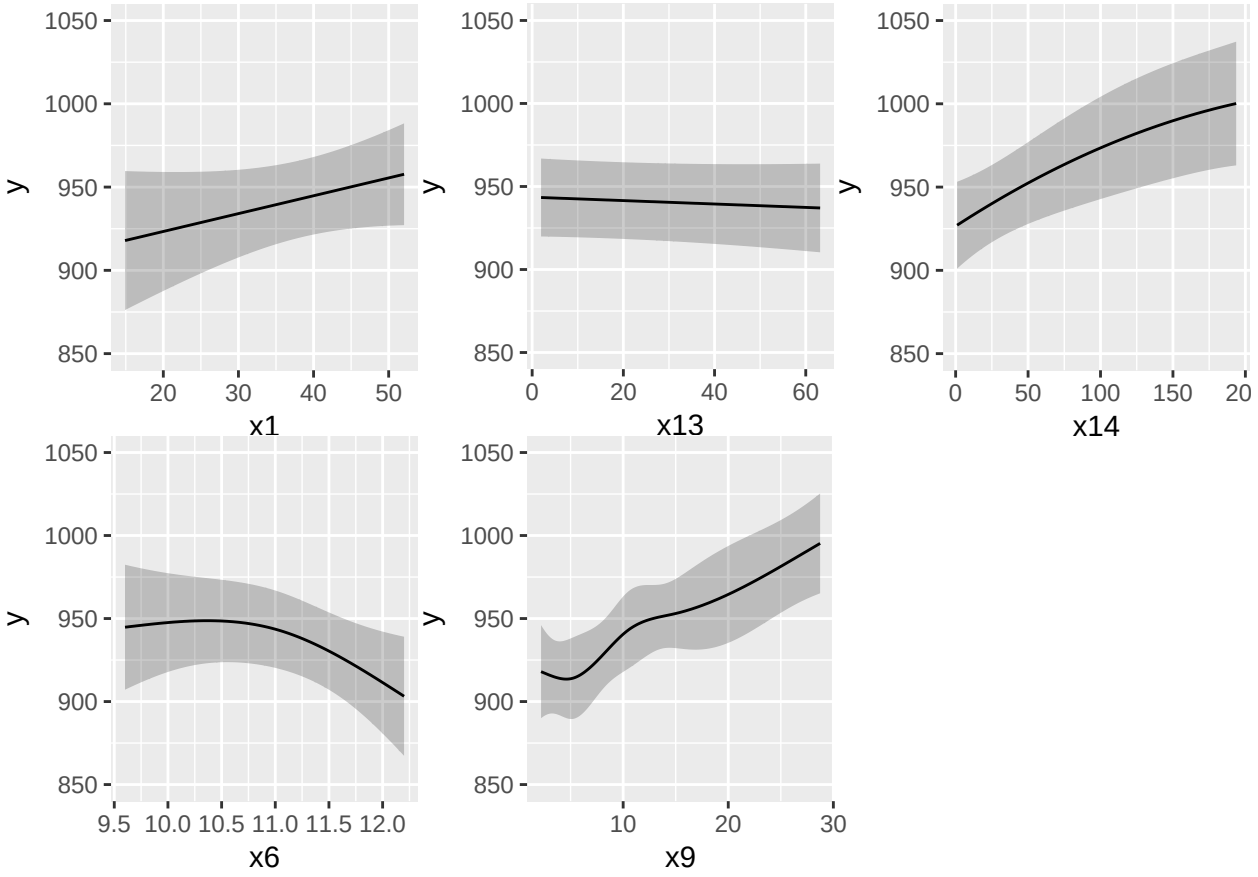
```
Predict(newmod, x9 = 5:15, conf.type = 'simult')
```

```
   x9    x6 x14 x1 x13     yhat    lower    upper
1   5 11.05  30 38   9 913.7392 882.4311 945.0473
2   6 11.05  30 38   9 916.3490 884.9354 947.7625
3   7 11.05  30 38   9 921.3093 892.4733 950.1454
4   8 11.05  30 38   9 927.6464 901.0465 954.2464
5   9 11.05  30 38   9 934.3853 907.2713 961.4993
6  10 11.05  30 38   9 940.5510 911.2371 969.8649
7  11 11.05  30 38   9 945.2225 915.2484 975.1966
8  12 11.05  30 38   9 948.2885 920.1141 976.4629
9  13 11.05  30 38   9 950.2608 924.5003 976.0212
10 14 11.05  30 38   9 951.6671 926.5912 976.7430
11 15 11.05  30 38   9 953.0342 926.1025 979.9658
```

```
Response variable (y): y

Adjust to: x6=11.05 x14=30 x1=38 x13=9

Limits are 0.95 confidence limits
```

The plot below shows the individual effects in `newmod` in five subpanels, using the default approach of displaying the same range of values as are seen in the data. Note that each panel shows point and interval estimates of the effects, and spot the straight lines in `x1` and `x13`, the single bends in `x14` and `x6` and the wiggles in `x9`, corresponding to the amount of non-linearity specified in the model.

```
ggplot(Predict(newmod))
```



10.5 Checking Influence via `dfbeta`

For an `ols` object, we have several tools for looking at residuals. The most interesting to me is `which.influence` which is reliant on the notion of `dfbeta`.

- DFBETA is estimated for each observation in the data, and each coefficient in the model.
- The DFBETA is the difference in the estimated coefficient caused by deleting the observation, scaled by the coefficient's standard error estimated with the observation deleted.
- The `which.influence` command applied to an `ols` model produces a list of all of the predictors estimated by the model, including the intercept.
  - For each predictor, the command lists all observations (by row number) that, if removed from the model, would cause the estimated coefficient (the "beta") for that predictor to change by at least

some particular cutoff.
  - The default is that the DFBETA for that predictor is 0.2 or more.

```
which.influence(newmod)
```

```
$Intercept
[1]  2 11 28 32 37 49 59

$x9
[1]  2  3  6  9 31 35 49 57 58

$x6
[1]  2 11 15 28 32 37 50 56 59

$x14
[1]  2  6  7 12 13 16 32 37

$x1
[1]  7 18 32 37 49 57

$x13
[1] 29 32 37
```

The implication here, for instance, is that if we drop row 3 from our data frame, and refit the model, this will have a meaningful impact on the estimate of `x9` but not on the other coefficients. But if we drop, say, row 37, we will affect the estimates of the intercept, `x6`, `x14`, `x1`, and `x13`.

10.5.1 Using the `residuals` command for `dfbetas`

To see the `dfbeta` values, standardized according to the approach I used above, you can use the following code (I'll use `head` to just show the first few rows of results) to get a matrix of the results.

```
head(residuals(newmod, type = "dfbetas"))
```

```
           [,1]         [,2]         [,3]        [,4]        [,5]
[1,]  0.03071160 -0.023775487 -0.004055111  0.01205425 -0.03260003
[2,] -0.38276573 -0.048404993 -0.142293606  0.17009666 -0.22350621
[3,]  0.17226780 -0.426153536  0.350913139 -0.32949129  0.25777913
[4,]  0.06175110 -0.006460916  0.024828272 -0.03009337  0.04154812
[5,]  0.16875200  0.039839994 -0.058178534  0.06449504 -0.07772208
[6,]  0.03322073  0.112699877 -0.203543632  0.23987378 -0.35201736
           [,6]        [,7]        [,8]         [,9]        [,10]
[1,] -0.02392315  0.01175375 -0.06494414  0.060929683 -0.011042644
[2,]  0.44737372 -0.48562818  0.19372285 -0.212186731 -0.107830147
[3,] -0.10263448  0.05005284 -0.02049877  0.014059330  0.010793169
[4,] -0.06254145  0.05498432  0.01135031 -0.001877983 -0.005490454
[5,] -0.18058630  0.16151742  0.02723710  0.065483158  0.003326357
[6,] -0.04075617  0.02900006 -0.21508009  0.171627718  0.019241676
          [,11]
[1,]  0.03425156
[2,] -0.01503250
[3,]  0.04924166
[4,] -0.01254111
[5,] -0.05570035
[6,]  0.05775536
```

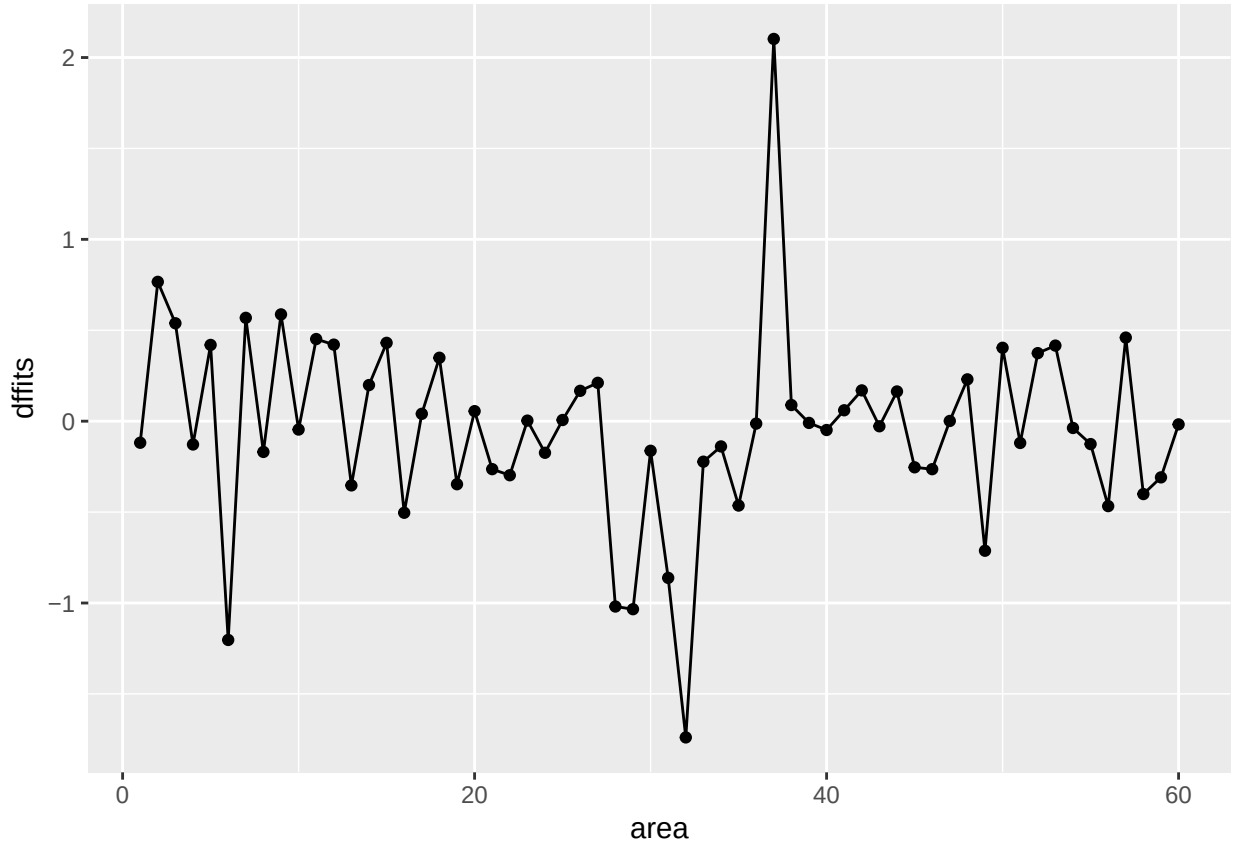
10.5.2 Using the residuals command for other summaries

The residuals command will also let you get ordinary residuals, leverage values and dffits values, which are the normalized differences in predicted values when observations are omitted. See ?residuals.ols for more details.

```
temp <- data.frame(area = 1:60)
temp$residual <- residuals(newmod, type = "ordinary")
temp$leverage <- residuals(newmod, type = "hat")
temp$dffits <- residuals(newmod, type = "dffits")
tbl_df(temp)
```

```
# A tibble: 60 x 4
    area residual leverage  dffits
   <int>    <dbl>    <dbl>   <dbl>
 1     1   -13.3    0.0929 -0.119
 2     2    81.0    0.0941  0.766
 3     3    28.8    0.266   0.539
 4     4   -12.5    0.117  -0.128
 5     5    27.8    0.204   0.419
 6     6   -40.4    0.416  -1.20
 7     7    37.0    0.207   0.568
 8     8   -14.3    0.145  -0.169
 9     9    66.6    0.0863  0.587
10    10   - 4.96   0.0997 -0.0460
# ... with 50 more rows
```

```
ggplot(temp, aes(x = area, y = dffits)) +
    geom_point() +
    geom_line()
```



It appears that point 37 has the largest (positive) `dffits` value. Recall that point 37 seemed influential on several predictors and the intercept term. Point 32 has the smallest (or largest negative) `dffits`, and also appears to have been influential on several predictors and the intercept.

```
which.max(temp$dffits)
```

```
[1] 37
```

```
which.min(temp$dffits)
```

```
[1] 32
```

10.6 Model Validation and Correcting for Optimism

In 431, we learned about splitting our regression models into **training** samples and **test** samples, performing variable selection work on the training sample to identify two or three candidate models (perhaps via a stepwise approach), and then comparing the predictions made by those models in a test sample.

At the final project presentations, I mentioned (to many folks) that there was a way to automate this process a bit in 432, that would provide some ways to get the machine to split the data for you multiple times, and then average over the results, using a bootstrap approach. This is it.

The `validate` function allows us to perform cross-validation of our models for some summary statistics (and then correct those statistics for optimism in describing likely predictive accuracy) in an easy way.

`validate` develops:

- Resampling validation with or without backward elimination of variables
- Estimates of the *optimism* in measures of predictive accuracy

- Estimates of the intercept and slope of a calibration model

$$\text{(observed y)} = \text{Intercept} + \text{Slope (predicted y)}$$

with the following code…

```
set.seed(432002); validate(newmod, method = "boot", B = 40)
```

```
          index.orig training      test  optimism index.corrected  n
R-square      0.6989   0.7500    0.5964    0.1536          0.5452 40
MSE        1145.7800 888.2564 1535.8277 -647.5714       1793.3514 40
g            58.9614  58.9323   55.2085    3.7238         55.2376 40
Intercept     0.0000   0.0000   86.5968  -86.5968         86.5968 40
Slope         1.0000   1.0000    0.9088    0.0912          0.9088 40
```

So, for `R-square` we see that our original estimate was 0.6989

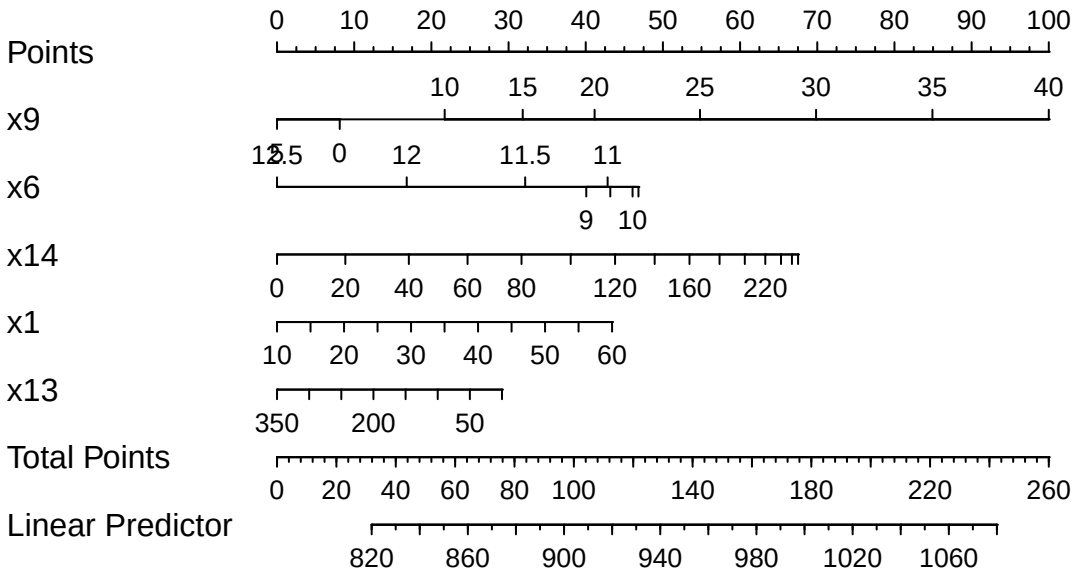
- Our estimated `R-square` across `n` = 40 training samples was 0.7500, but in the resulting tests, the average `R-square` was only 0.5964
- This suggests an optimism of 0.7500 - 0.5964 = 0.1536 (after rounding).
- We then apply that optimism to obtain a new estimate of R^2 corrected for overfitting, at 0.5452, which is probably a better estimate of what our results might look like in new data that were similar to (but not the same as) the data we used in building `newmod` than our initial estimate of 0.6989

We also obtain optimism-corrected estimates of the mean squared error (square of the residual standard deviation), the g index, and the intercept and slope of the calibration model. The "corrected" slope is a shrinkage factor that takes overfitting into account.

10.7 Building a Nomogram for Our Model

Another nice feature of an `ols` model object is that we can picture the model with a **nomogram** easily. Here is model `newmod`.

```
plot(nomogram(newmod))
```



For this model, we can use this plot to predict `y` as follows:

1. find our values of `x9` on the appropriate line
2. draw a vertical line up to the points line to count the points associated with our subject
3. repeat the process to obtain the points associated with `x6`, `x14`, `x1`, and `x13`. Sum the points.
4. draw a vertical line down from that number in the Total Points line to estimate `y` (the Linear Predictor) = Age-Adjusted Mortality Rate.

The impact of the non-linearity is seen in the `x6` results, for example, which turn around from 9-10 to 11-12. We also see non-linearity's effects in the scales of the non-linear terms in terms of points awarded.

An area with a combination of predictor values leading to a total of 100 points, for instance, would lead to a prediction of a Mortality Rate near 905. An area with a total of 140 points would have a predicted Mortality Rate of 955, roughly.

Chapter 11

Other Variable Selection Strategies

11.1 Why not use stepwise procedures?

1. The R^2 for a model selected in a stepwise manner is biased, high.
2. The coefficient estimates and standard errors are biased.
3. The p values for the individual-variable t tests are too small.
4. In stepwise analyses of prediction models, the final model represented noise 20-74% of the time.
5. In stepwise analyses, the final model usually contained less than half of the actual number of real predictors.
6. It is not logical that a population regression coefficient would be exactly zero just because its estimate was not statistically significant.

This last comment applies to things like our "best subsets" approach as well as standard stepwise procedures.

Sander Greenland's comments on parsimony and stepwise approaches to model selection are worth addressing…

- Stepwise variable selection on confounders leaves important confounders uncontrolled.
- Shrinkage approaches (like ridge regression and the lasso) are far superior to variable selection.
- Variable selection does more damage to confidence interval widths than to point estimates.

If we are seriously concerned about **overfitting** - winding up with a model that doesn't perform well on new data - then stepwise approaches generally don't help.

Vittinghoff et al. (2012) suggest four strategies for minimizing the chance of overfitting

1. Pre-specify well-motivated predictors and how to model them.
2. Eliminate predictors without using the outcome.
3. Use the outcome, but cross-validate the target measure of prediction error.
4. Use the outcome, and **shrink** the coefficient estimates.

The best subsets methods we have studied either include a variable or drop it from the model. Often, this choice is based on only a tiny difference in the quality of a fit to data.

- Harrell (2001): not reasonable to assume that a population regression coefficient would be exactly zero just because it failed to meet a criterion for significance.
- Brad Efron has suggested that a stepwise approach is "overly greedy, impulsively eliminating covariates which are correlated with other covariates."

So, what's the alternative?

11.2 Ridge Regression

Ridge regression involves a more smooth transition between useful and not useful predictors which can be obtained by constraining the overall size of the regression coefficients.

Ridge regression assumes that the regression coefficients (after normalization) should not be very large. This is reasonable to assume when you have lots of predictors and you believe *many* of them have some effect on the outcome.

Pros:

1. Some nice statistical properties
2. Can be calculated using only standard least squares approaches, so it's been around for a while.
3. Available in the `MASS` package.

Ridge regression takes the sum of the squared estimated standardized regression coefficients and constrains that sum to only be as large as some value k.

$$\sum \hat{\beta}_j^{\,2} \leq k.$$

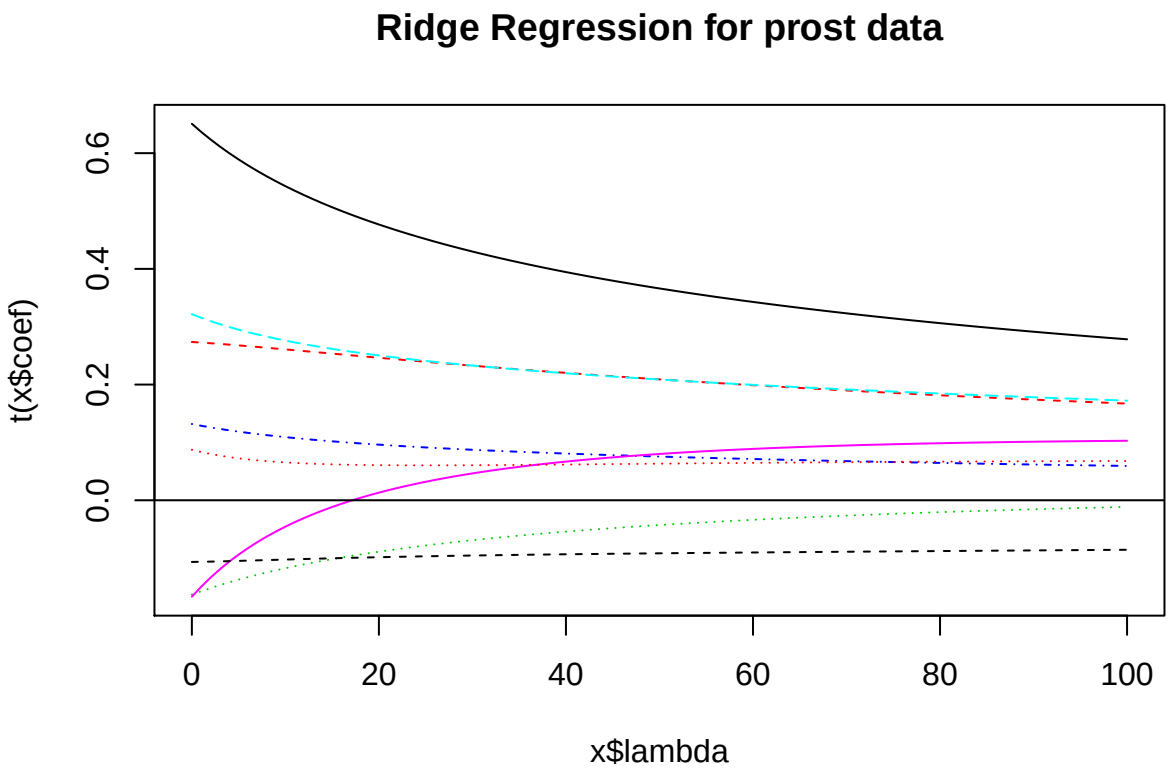
The value k is one of several available measures of the amount of shrinkage, but the main one used in the `MASS` package is a value λ. As λ increases, the amount of shrinkage goes up, and k goes down.

11.2.1 Assessing a Ridge Regression Approach

We'll look at a plot produced by the `lm.ridge` function for a ridge regression for the prostate cancer study we worked on when studying Stepwise Regression and Best Subsets methods earlier.

- Several (here 101) different values for λ, our shrinkage parameter, will be tested.
- Results are plotted so that we see the coefficients across the various (standardized) predictors.
  - Each selection of a λ value implies a different vector of covariate values across the predictors we are studying.
  - The idea is to pick a value of λ for which the coefficients seem relatively stable.

```
preds <- with(prost, cbind(lcavol, lweight, age, bph_f,
                           svi_f, lcp, gleason_f, pgg45))

x <- lm.ridge(prost$lpsa ~ preds, lambda = 0:100)

plot(x)
title("Ridge Regression for prost data")
abline(h = 0)
```

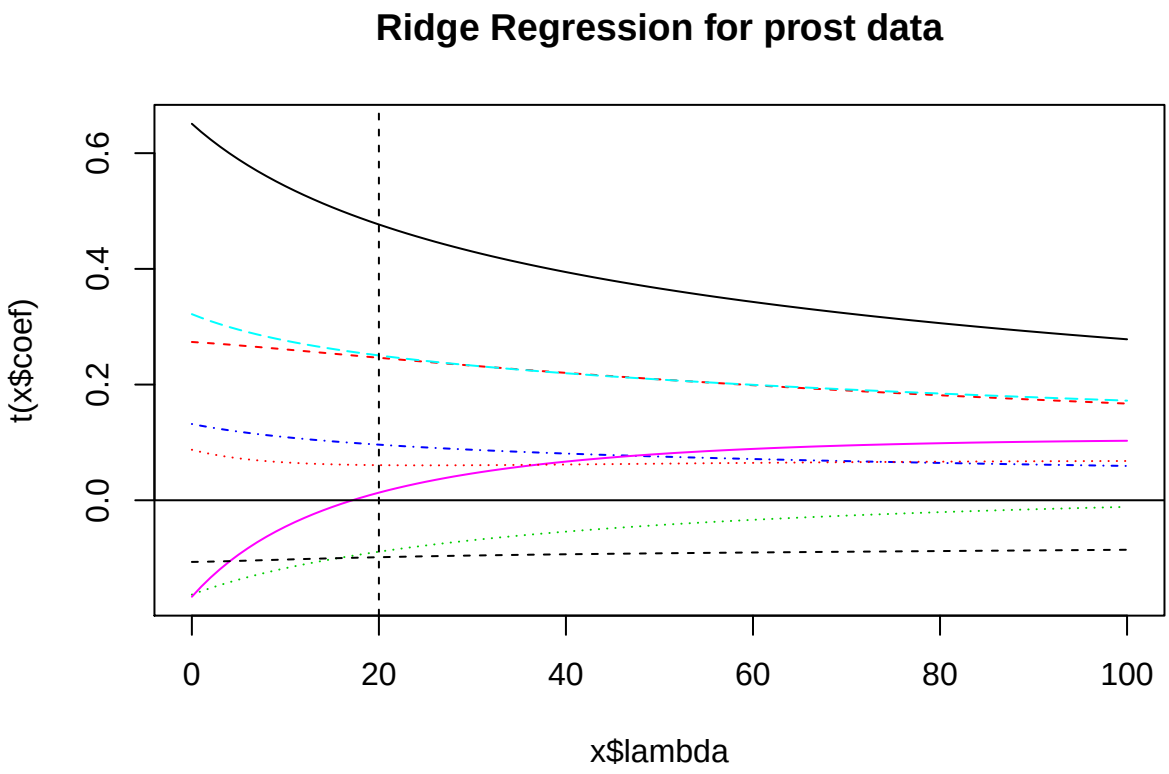


Usually, you need to use trial and error to decide the range of λ to be tested. Here, `0:100` means going from 0 (no shrinkage) to 100 in steps of 1.

11.2.2 The `lm.ridge` plot - where do coefficients stabilize?

Does $\lambda = 20$ seem like a stable spot here?

```
x <- lm.ridge(prost$lpsa ~ preds, lambda = 0:100)
plot(x)
title("Ridge Regression for prost data")
abline(h = 0)
abline(v=20, lty=2, col="black")
```



The coefficients at $\lambda = 20$ can be determined from the `lm.ridge` output. These are fully standardized coefficients. The original predictors are centered by their means and then scaled by their standard deviations and the outcome has also been centered, in these models.

```
round(x$coef[,20],3)
```

```
   predslcavol   predslweight       predsage     predsbph_f     predssvi_f
         0.482          0.248         -0.091          0.097          0.252
     predslcp predsgleason_f     predspgg45
         0.009         -0.099          0.061
```

Was an intercept used?

```
x$Inter
```

```
[1] 1
```

Yes, it was. There is an automated way to pick λ. Use the `select` function in the `MASS` package:

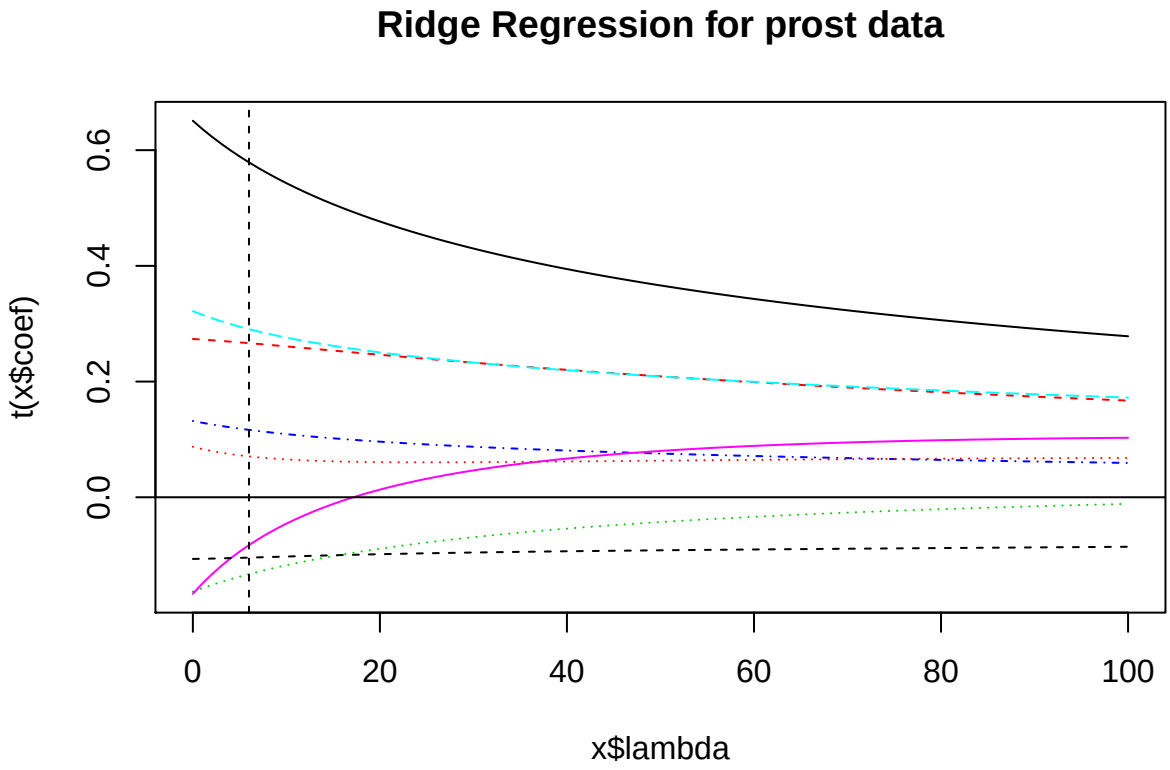
```
MASS::select(x)
```

```
modified HKB estimator is 4.210238
modified L-W estimator is 3.32223
smallest value of GCV  at 6
```

I'll use the GCV = generalized cross-validation to select $\lambda = 6$ instead.

```
x <- lm.ridge(prost$lpsa ~ preds, lambda = 0:100)
plot(x)
title("Ridge Regression for prost data")
abline(h = 0)
```

```
abline(v=6, lty=2, col="black")
```




```
x$coef[,6]
```

```
   predslcavol   predslweight       predsage     predsbph_f     predssvi_f
     0.58911149     0.26773757    -0.13715070     0.11862949     0.29491008
      predslcp predsgleason_f     predspgg45
    -0.09389545    -0.10477578     0.07250609
```

11.2.3 Ridge Regression: The Bottom Line

The main problem with ridge regression is that all it does is shrink the coefficient estimates, but it's not so useful in practical settings because it still includes all variables.

1. It's been easy to do ridge regression for many years, so you see it occasionally in the literature.
2. It leads to the **lasso**, which incorporates the positive features of shrinking regression coefficients with the ability to wisely select some variables to be eliminated from the predictor pool.

11.3 The Lasso

The lasso works by takes the sum of the absolute values of the estimated standardized regression coefficients and constrains it to only be as large as some value k.

$$\sum |\hat{\beta}_j| \leq k.$$

This looks like a minor change, but it's not.

11.3.1 Consequences of the Lasso Approach

1. In ridge regression, while the individual coefficients shrink and sometimes approach zero, they seldom reach zero and are thus excluded from the model. With the lasso, some coefficients do reach zero and thus, those predictors do drop out of the model.
   - So the lasso leads to more parsimonious models than does ridge regression.
   - Ridge regression is a method of shrinkage but not model selection. The lasso accomplishes both tasks.
2. If k is chosen to be too small, then the model may not capture important characteristics of the data. If k is too large, the model may over-fit the data in the sample and thus not represent the population of interest accurately.
3. The lasso is far more difficult computationally than ridge regression (the problem requires an algorithm called least angle regression published in 2004), although R has a library (`lars`) which can do the calculations pretty efficiently.

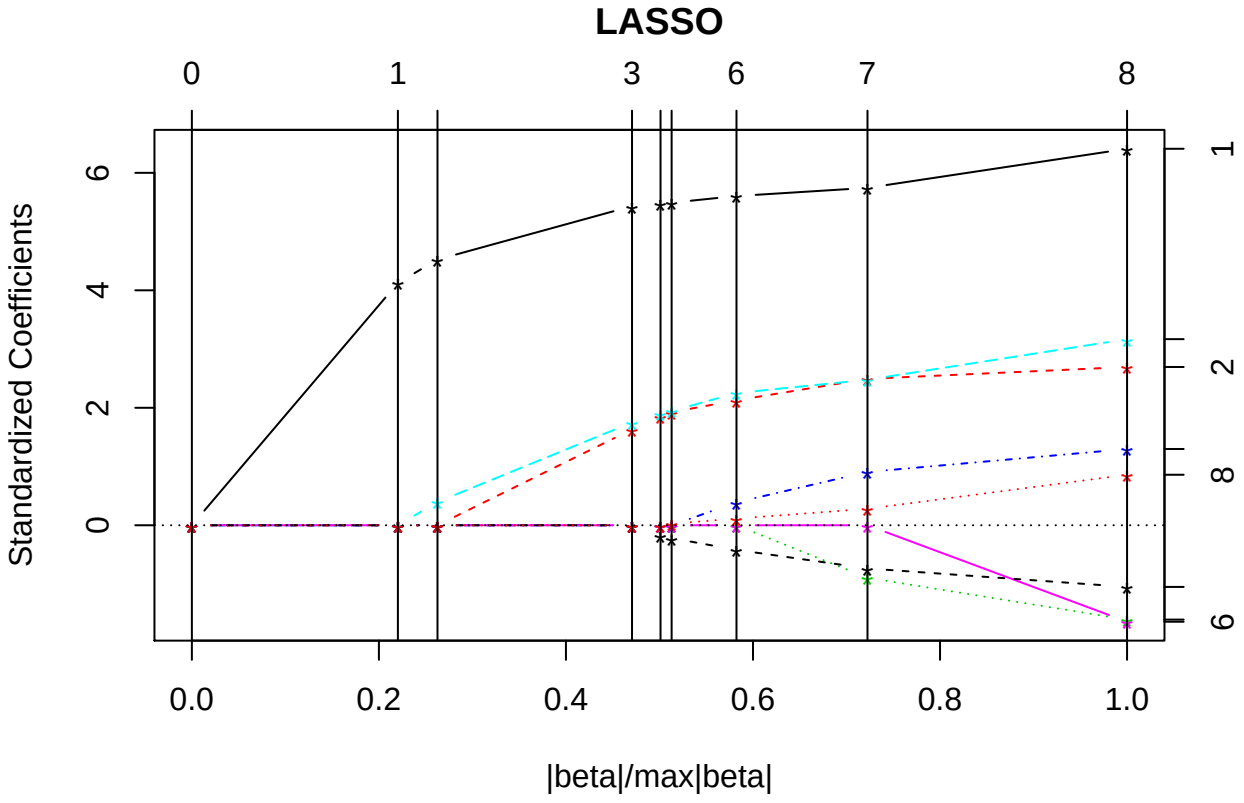
The lasso is not an acronym, but rather refers to cowboys using a rope to pull cattle from the herd, much as we will pull predictors from a model.

11.3.2 How The Lasso Works

The `lars` package lets us compute the lasso coefficient estimates **and** do cross-validation to determine the appropriate amount of shrinkage. The main tool is a pair of graphs.

1. The first plot shows what coefficients get selected as we move from constraining all of the coefficients to zero (complete shrinkage) towards fewer constraints all the way up to ordinary least squares, showing which variables are included in the model at each point.
2. The second plot suggests where on the first plot we should look for a good model choice, according to a cross-validation approach.

```
## requires lars package
lasso1 <- lars(preds, prost$lpsa, type="lasso")
plot(lasso1)
```



- The y axis shows standardized regression coefficients.
  - The `lars` package standardizes all variables so the shrinkage doesn't penalize some coefficients because of their scale.
- The x-axis is labeled `|beta|/max|beta|`.
  - This ranges from 0 to 1.
  - 0 means that the sum of the $|\hat{\beta}_j|$ is zero (completely shrunk)
  - 1 means the ordinary least squares unbiased estimates.

The lasso graph starts at constraining all of the coefficients to zero, and then moves toward ordinary least squares.

Identifiers for the predictors (numbers) are shown to the right of the graph.

The vertical lines in the lasso plot show when a variable has been eliminated from the model, and in fact these are the only points that are actually shown in the default lasso graph. The labels on the top of the graph tell you how many predictors are in the model at that stage.

```
summary(lasso1)
```

```
LARS/LASSO
Call: lars(x = preds, y = prost$lpsa, type = "lasso")
  Df     Rss       Cp
0  1 127.918 168.1835
1  2  76.392  64.1722
2  3  70.247  53.5293
3  4  50.598  15.1017
4  5  49.065  13.9485
5  6  48.550  14.8898
6  7  46.284  12.2276
```

```
7  8  44.002   9.5308
8  9  42.772   9.0000
```

Based on the C_p statistics, it looks like the improvements continue throughout, and don't really finish happening until we get pretty close to the full model with 9 df.

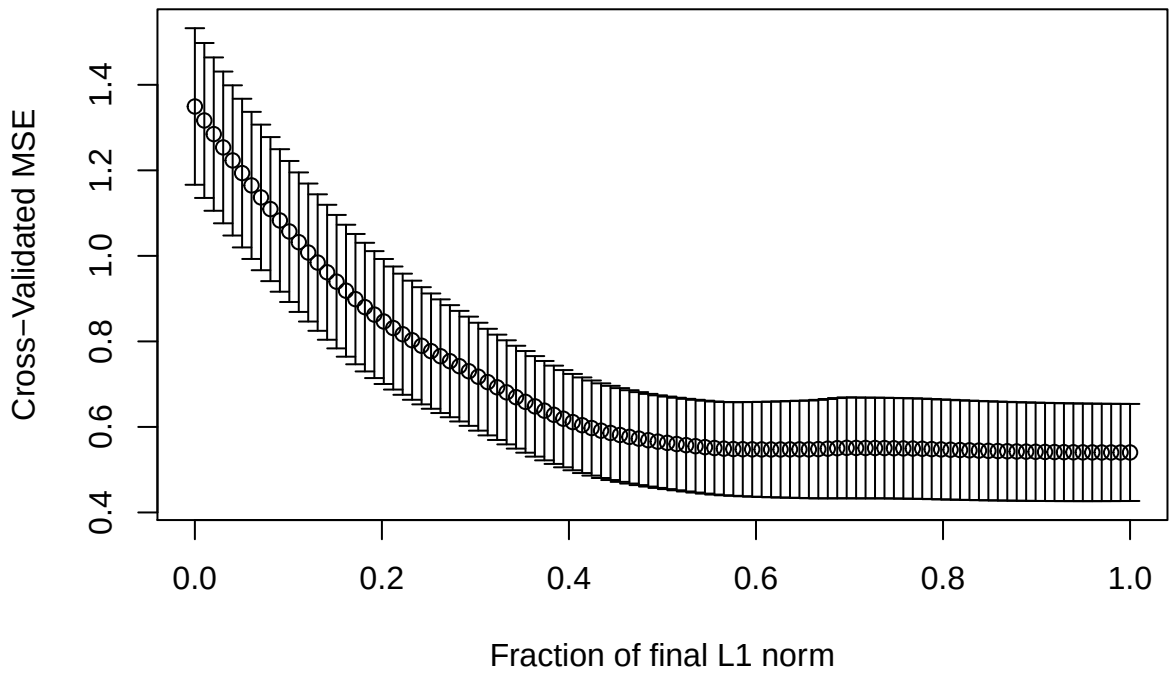
11.3.3 Cross-Validation with the Lasso

Normally, cross-validation methods are used to determine how much shrinkage should be used. We'll use the `cv.lars` function.

- 10-fold (K = 10) cross-validation
  - the data are randomly divided into 10 groups.
  - Nine groups are used to predict the remaining group for each group in turn.
  - Overall prediction performance is computed, and the machine calculates a cross-validation criterion (mean squared error) and standard error for that criterion.

The cross-validation plot is the second lasso plot.

```
set.seed(432)
lassocv <- cv.lars(preds, prost$lpsa, K=10)
```




```
## default cv.lars K is 10
```

We're looking to minimize cross-validated mean squared error in this plot, which doesn't seem to happen until the fraction gets very close to 1.

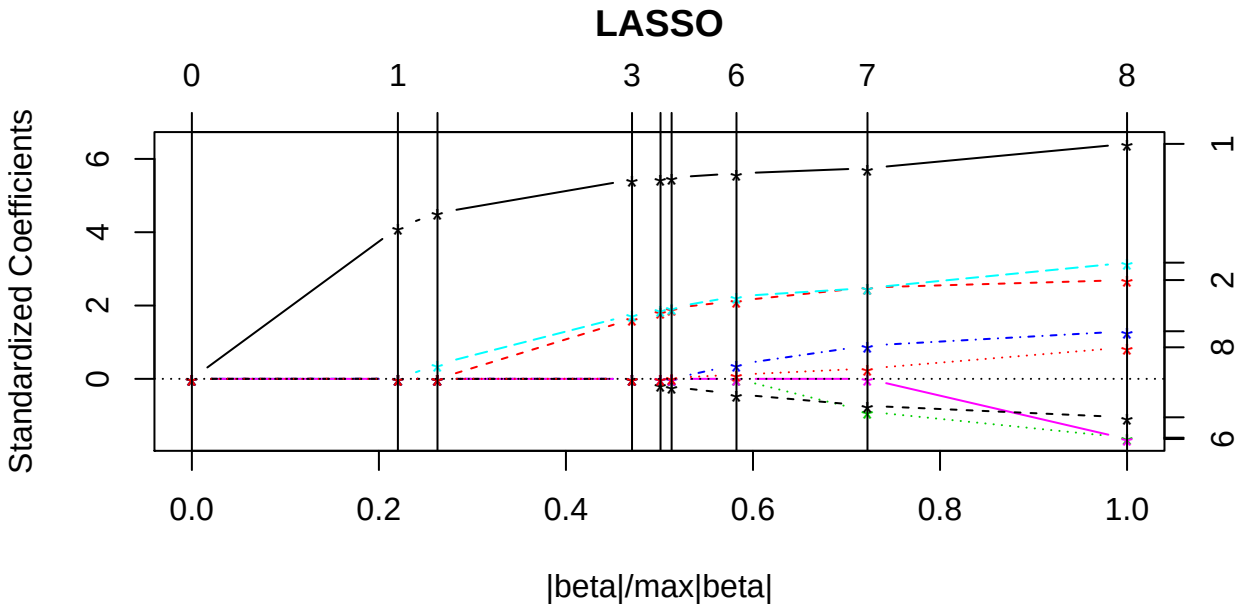
11.3.4 What value of the key fraction minimizes cross-validated MSE?

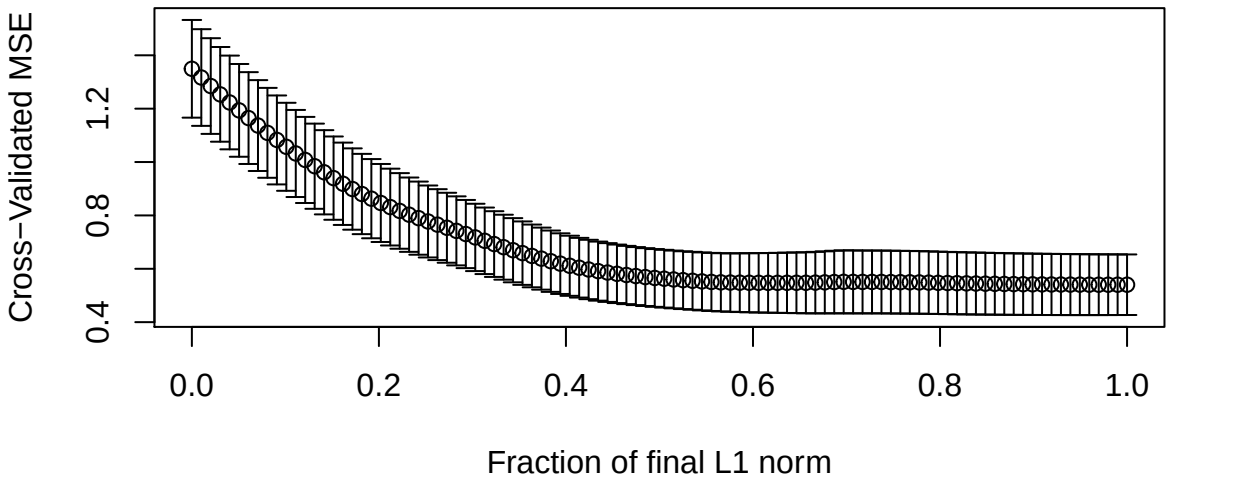
```
frac <- lassocv$index[which.min(lassocv$cv)]
frac
```

```
[1] 0.989899
```

The cross-validation plot suggests we use a fraction of about 0.3, that's suggesting a model with 4-5 predictors, based on the top LASSO plot.

```
par(mfrow=c(2,1))
lasso1 <- lars(preds, prost$lpsa, type="lasso")
plot(lasso1)
set.seed(432)
lassocv <- cv.lars(preds, prost$lpsa, K=10)
```






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

11.3.5 Coefficients for the Model Identified by the Cross-Validation

```
coef.cv <- coef(lasso1, s=frac, mode="fraction")
round(coef.cv,4)
```

```
   lcavol   lweight       age     bph_f     svi_f       lcp gleason_f
   0.5529    0.6402   -0.0217    0.1535    0.7750   -0.1155   -0.1826
    pgg45
```

```
   0.0030
```

So the model suggested by the lasso still includes all sight of these predictors.

11.3.6 Obtaining Fitted Values from Lasso

```
fits.cv <- predict.lars(lasso1, preds, s=frac,
                        type="fit", mode="fraction")
fits.cv
```

```
$s
[1] 0.989899

$fraction
[1] 0.989899

$mode
[1] "fraction"

$fit
 [1] 0.7995838 0.7493971 0.5111634 0.6098520 1.7001847 0.8338020 1.8288518
 [8] 2.1302316 1.2487955 1.2661752 1.4704969 0.7782005 2.0755860 1.9129272
[15] 2.1533975 1.8124981 1.2713610 2.3993624 1.3232566 1.7709029 1.9757841
[22] 2.7451649 1.1658326 2.4825521 1.8036338 1.9112578 2.0144298 1.7829219
[29] 1.9706111 2.1688199 2.0377131 1.8657882 1.6955904 1.3580186 1.0516394
[36] 2.9097450 2.1898622 1.0454123 3.8896481 1.7971270 2.0932871 2.3253395
[43] 2.0809295 2.5303655 2.4451523 2.5827203 4.0692397 2.6845105 2.7034959
[50] 1.9590266 2.4522082 2.9801227 2.1902084 3.0559124 3.3447025 2.9765233
[57] 1.7620182 2.3424646 2.2856404 2.6188548 2.3056410 3.5568662 2.9756755
[64] 3.6764122 2.5097586 2.6579014 2.9482717 3.0892917 1.5113015 3.0282296
[71] 3.2887119 2.1083273 2.8889223 3.4903026 3.6959516 3.6070031 3.2749993
[78] 3.4518575 3.4049180 3.1814731 2.0496216 2.8986175 3.6743113 3.3292860
[85] 2.6965297 3.8339856 2.9892543 3.0555536 4.2903885 3.0986508 3.3784385
[92] 4.0205201 3.8309974 4.7531590 3.6290575 4.1347645 4.0982744
```

11.3.7 Complete Set of Fitted Values from the Lasso

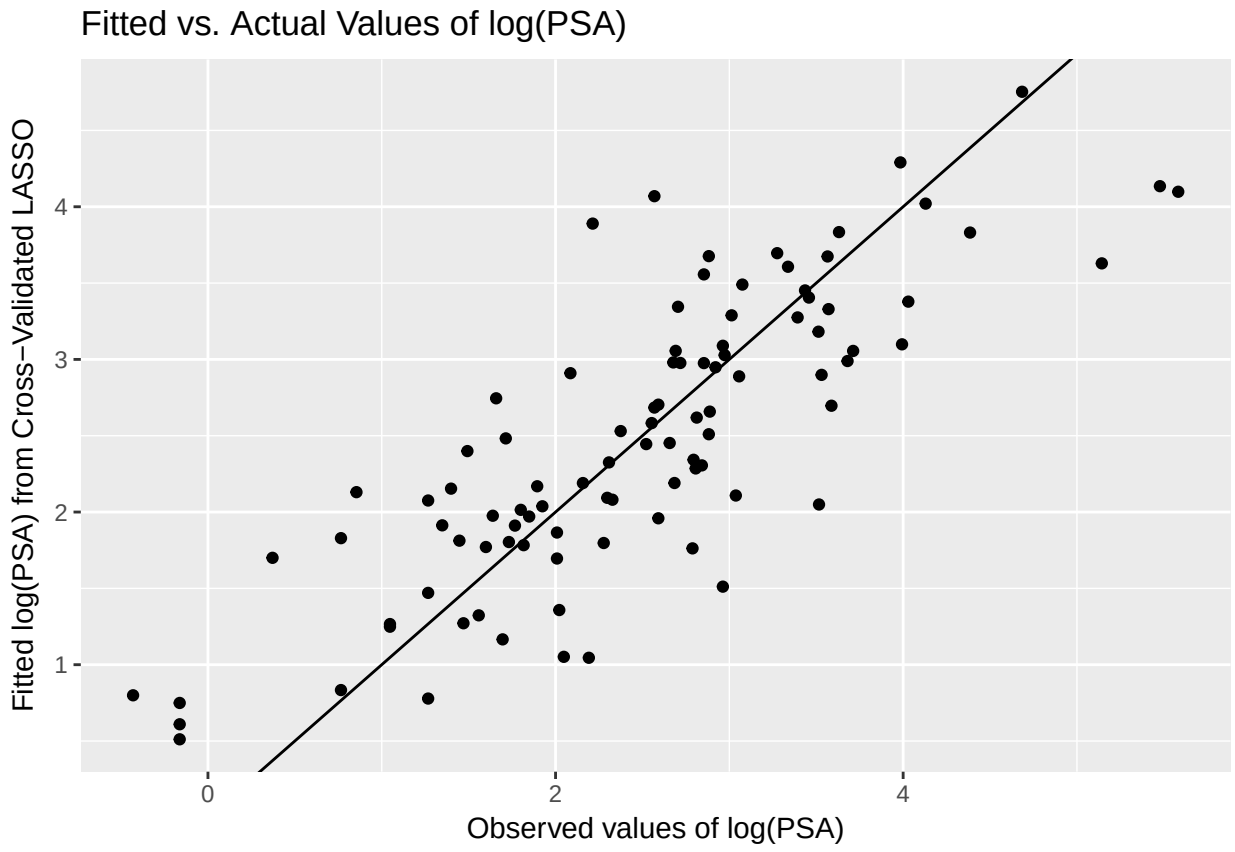
```
round(fits.cv$fit,3)
```

```
 [1] 0.800 0.749 0.511 0.610 1.700 0.834 1.829 2.130 1.249 1.266 1.470
[12] 0.778 2.076 1.913 2.153 1.812 1.271 2.399 1.323 1.771 1.976 2.745
[23] 1.166 2.483 1.804 1.911 2.014 1.783 1.971 2.169 2.038 1.866 1.696
[34] 1.358 1.052 2.910 2.190 1.045 3.890 1.797 2.093 2.325 2.081 2.530
[45] 2.445 2.583 4.069 2.685 2.703 1.959 2.452 2.980 2.190 3.056 3.345
[56] 2.977 1.762 2.342 2.286 2.619 2.306 3.557 2.976 3.676 2.510 2.658
[67] 2.948 3.089 1.511 3.028 3.289 2.108 2.889 3.490 3.696 3.607 3.275
[78] 3.452 3.405 3.181 2.050 2.899 3.674 3.329 2.697 3.834 2.989 3.056
[89] 4.290 3.099 3.378 4.021 3.831 4.753 3.629 4.135 4.098
```

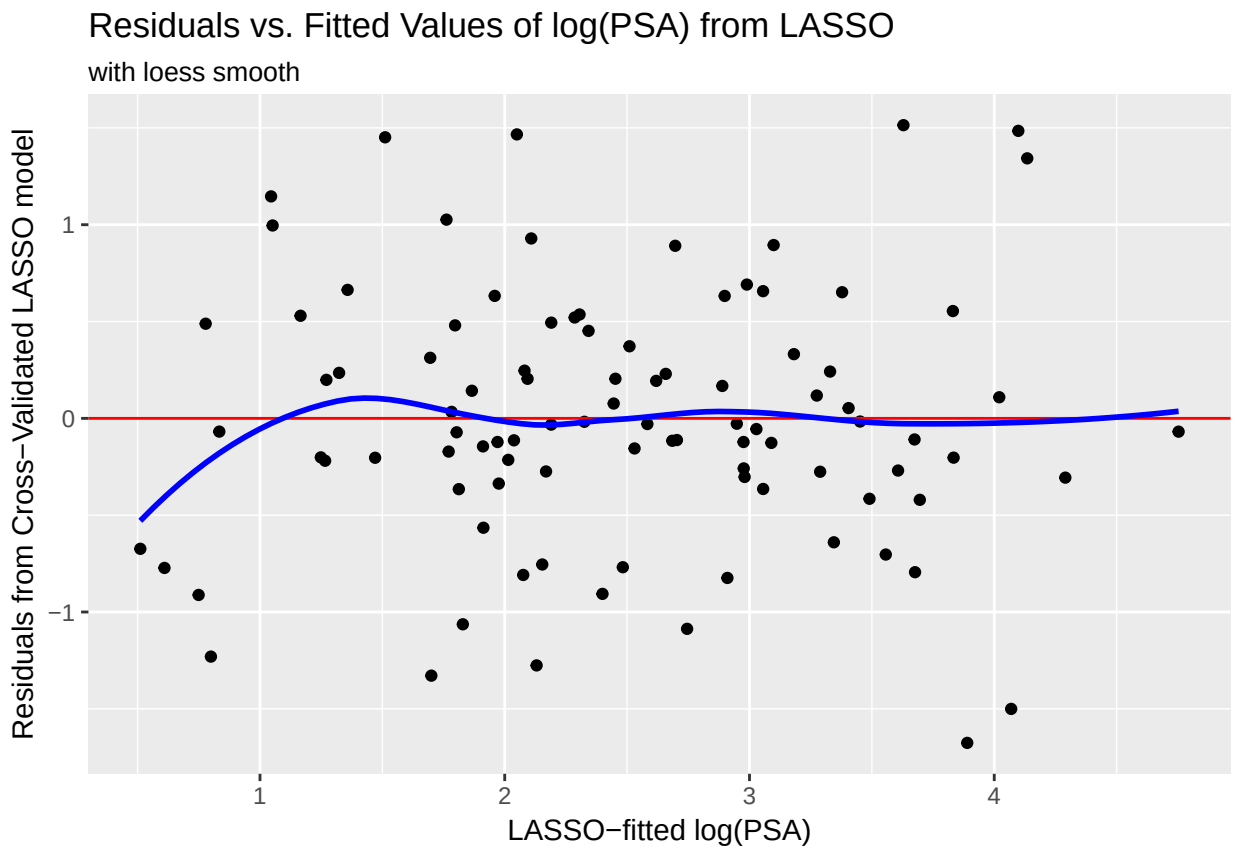
To assess the quality of these predictions, we might plot them against the observed values of our outcome (`lpsa`), or we might look at residuals vs. these fitted values.

```
prost_lasso_res <- data_frame(fitted = fits.cv$fit,
                              actual = prost$lpsa,
```

```
                              resid = actual - fitted)

ggplot(prost_lasso_res, aes(x = actual, y = fitted)) +
    geom_point() +
    geom_abline(slope = 1, intercept = 0) +
    labs(y = "Fitted log(PSA) from Cross-Validated LASSO",
         x = "Observed values of log(PSA)",
         title = "Fitted vs. Actual Values of log(PSA)")
```




```
ggplot(prost_lasso_res, aes(x = fitted, y = resid)) +
    geom_point() +
    geom_hline(yintercept = 0, col = "red") +
    geom_smooth(method = "loess", col = "blue", se = F) +
    labs(x = "LASSO-fitted log(PSA)",
         y = "Residuals from Cross-Validated LASSO model",
         title = "Residuals vs. Fitted Values of log(PSA) from LASSO",
         subtitle = "with loess smooth")
```



11.3.8 When is the Lasso Most Useful?

As Faraway (2015) suggests, the lasso is particularly useful when we believe the effects are sparse, in the sense that we believe that few of the many predictors we are evaluating have a meaningful effect.

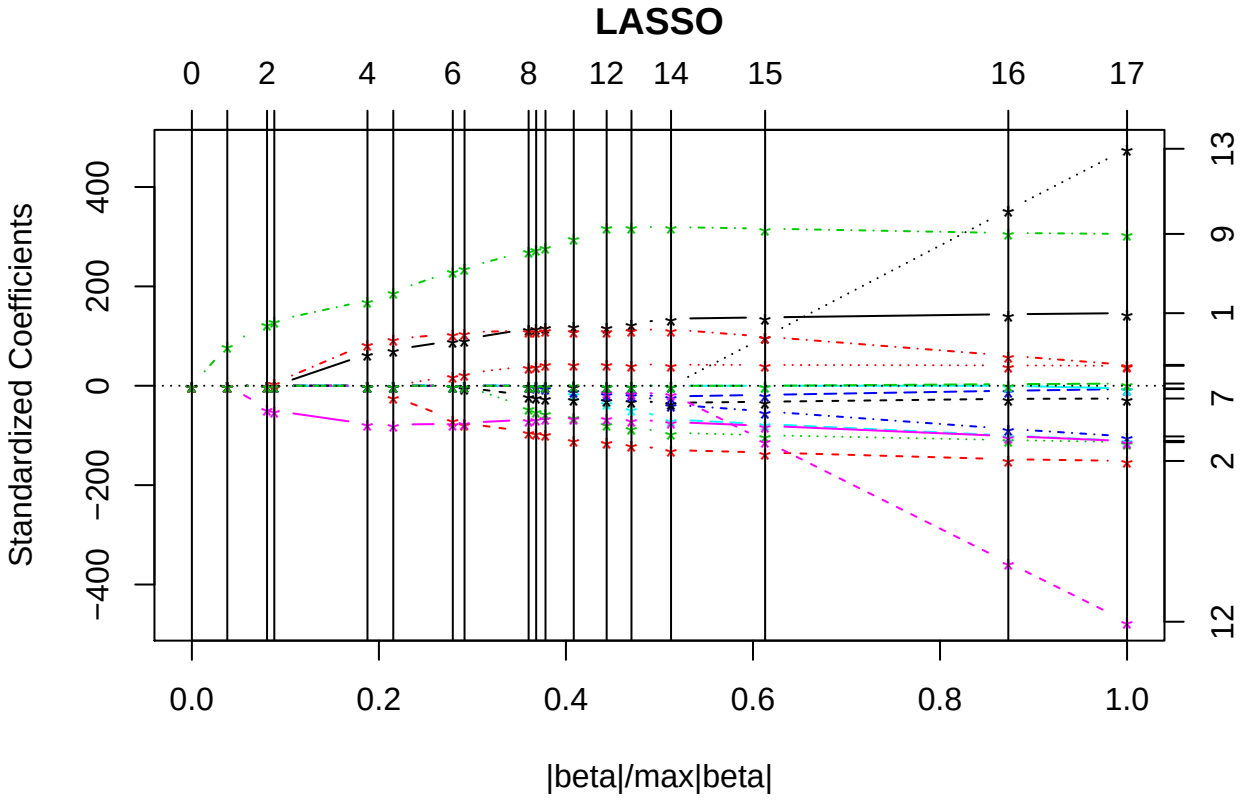
Consider, for instance, the analysis of gene expression data, where we have good reason to believe that only a small number of genes have an influence on our response of interest.

Or, in medical claims data, where we can have thousands of available codes to search through that may apply to some of the people included in a large analysis relating health care costs to outcomes.

11.4 Applying the Lasso to the `pollution` data

Let's consider the lasso approach in application to the `pollution` data we've seen previously. Recall that we have 60 observations on an outcome, `y`, and 15 predictors, labeled x1 through x15.

```
preds <- with(pollution, cbind(x1, x2, x3, x4, x5, x6, x7,
                               x8, x9, x10, x11, x12, x13,
                               x14, x15))

lasso_p1 <- lars(preds, pollution$y, type="lasso")
plot(lasso_p1)
```

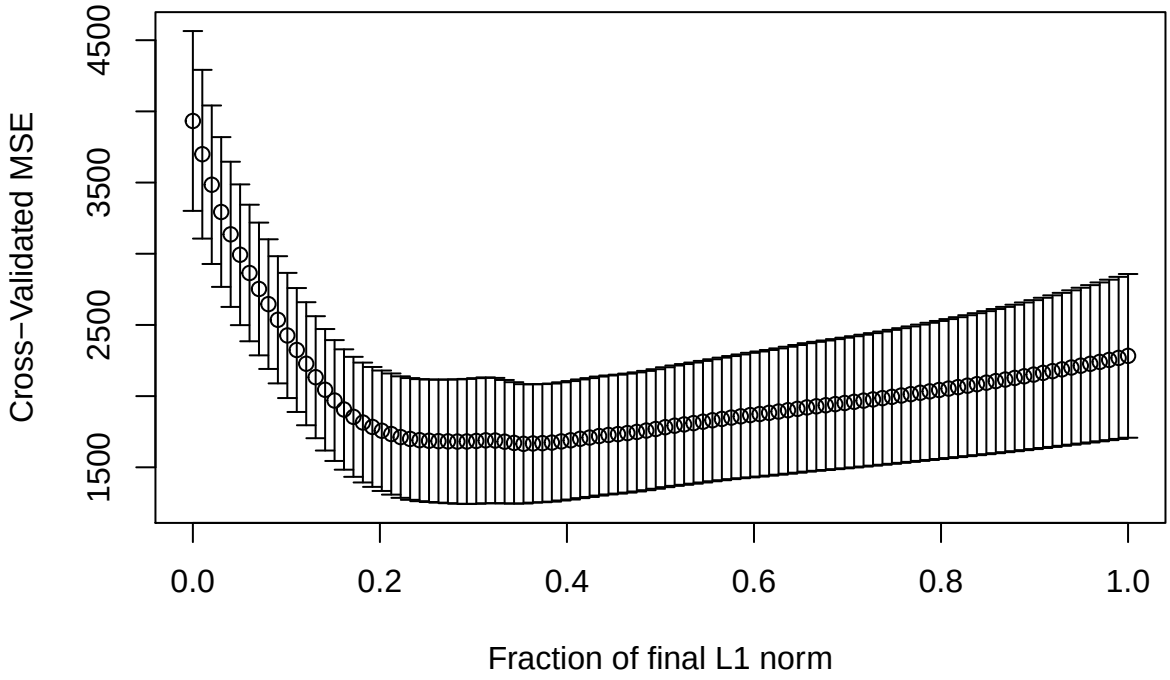



```
summary(lasso_p1)
```

```
LARS/LASSO
Call: lars(x = preds, y = pollution$y, type = "lasso")
   Df    Rss       Cp
0   1 228311 129.1367
1   2 185419  95.9802
2   3 149370  68.4323
3   4 143812  65.8764
4   5  92077  25.4713
5   6  83531  20.4668
6   7  69532  10.9922
7   8  67682  11.4760
8   9  60689   7.7445
9  10  60167   9.3163
10 11  59609  10.8588
11 12  58287  11.7757
12 13  57266  12.9383
13 14  56744  14.5107
14 13  56159  12.0311
15 14  55238  13.2765
16 15  53847  14.1361
17 16  53681  16.0000
```

Based on the C_p statistics, it looks like the big improvements occur somewhere around the move from 6 to 7 df. Let's look at the cross-validation

```
set.seed(432012)
pollution_lassocv <- cv.lars(preds, pollution$y, K=10)
```

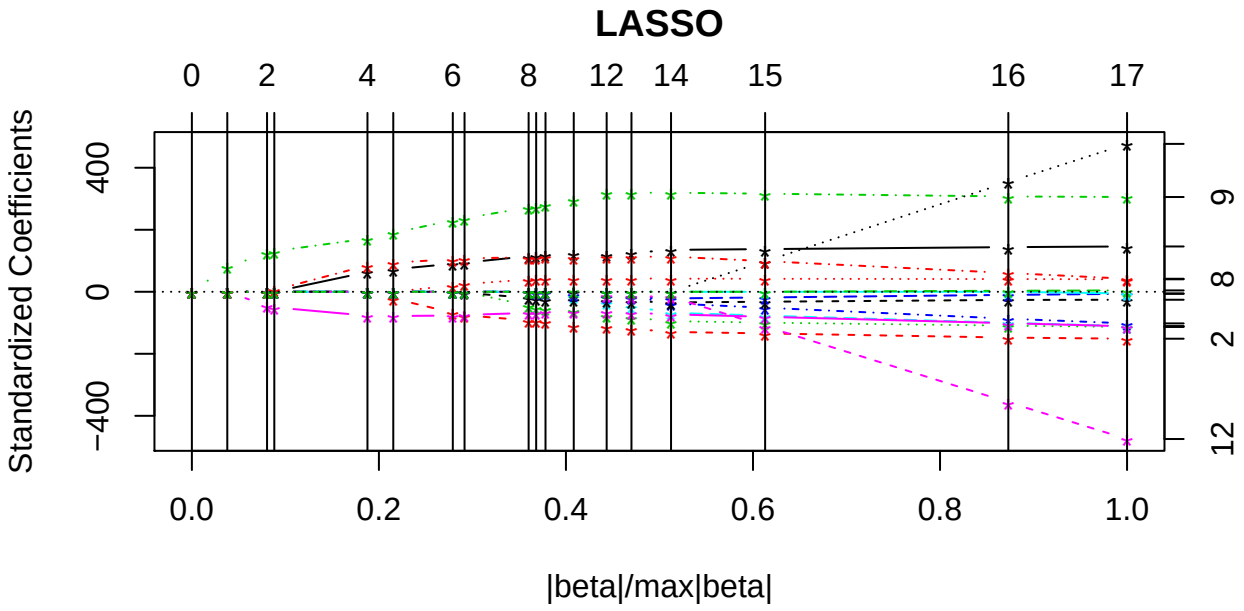


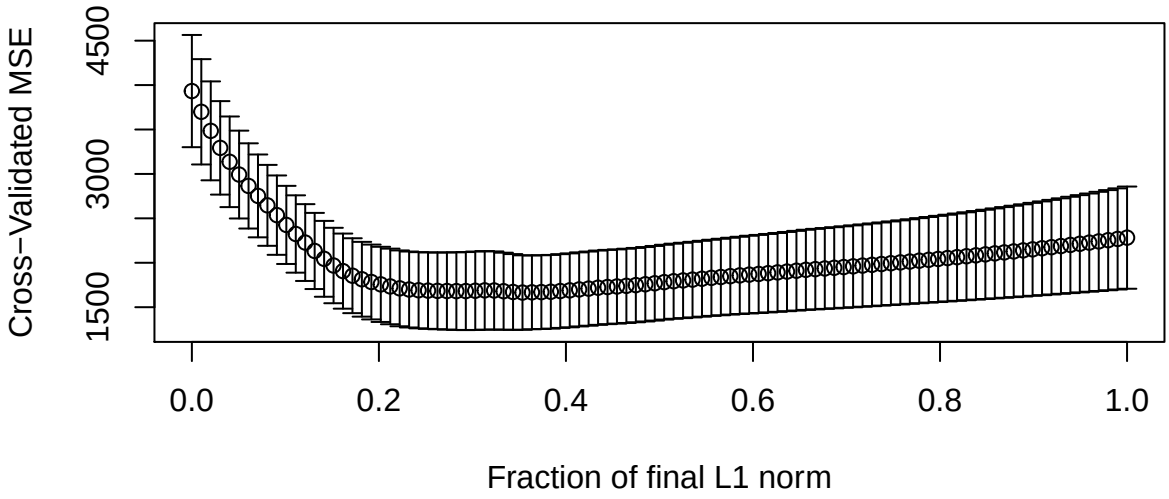
Here it looks like cross-validated MSE happens somewhere between a fraction of 0.2 and 0.4.

```
frac <- pollution_lassocv$index[which.min(pollution_lassocv$cv)]
frac
```

```
[1] 0.3535354
```

```
par(mfrow=c(2,1))
lasso_p1 <- lars(preds, pollution$y, type="lasso")
plot(lasso_p1)
set.seed(432012)
pollution_lassocv <- cv.lars(preds, pollution$y, K=10)
```






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

It looks like a model with 6-8 predictors will be the most useful. The cross-validated coefficients are as follows:

```
poll.cv <- coef(lasso_p1, s=frac, mode="fraction")
round(poll.cv,3)
```

```
    x1     x2     x3     x4     x5     x6     x7     x8     x9
 1.471 -1.164 -1.102  0.000  0.000 -10.610 -0.457  0.003  3.918
   x10    x11    x12    x13    x14    x15
 0.000  0.000  0.000  0.000  0.228  0.000
```

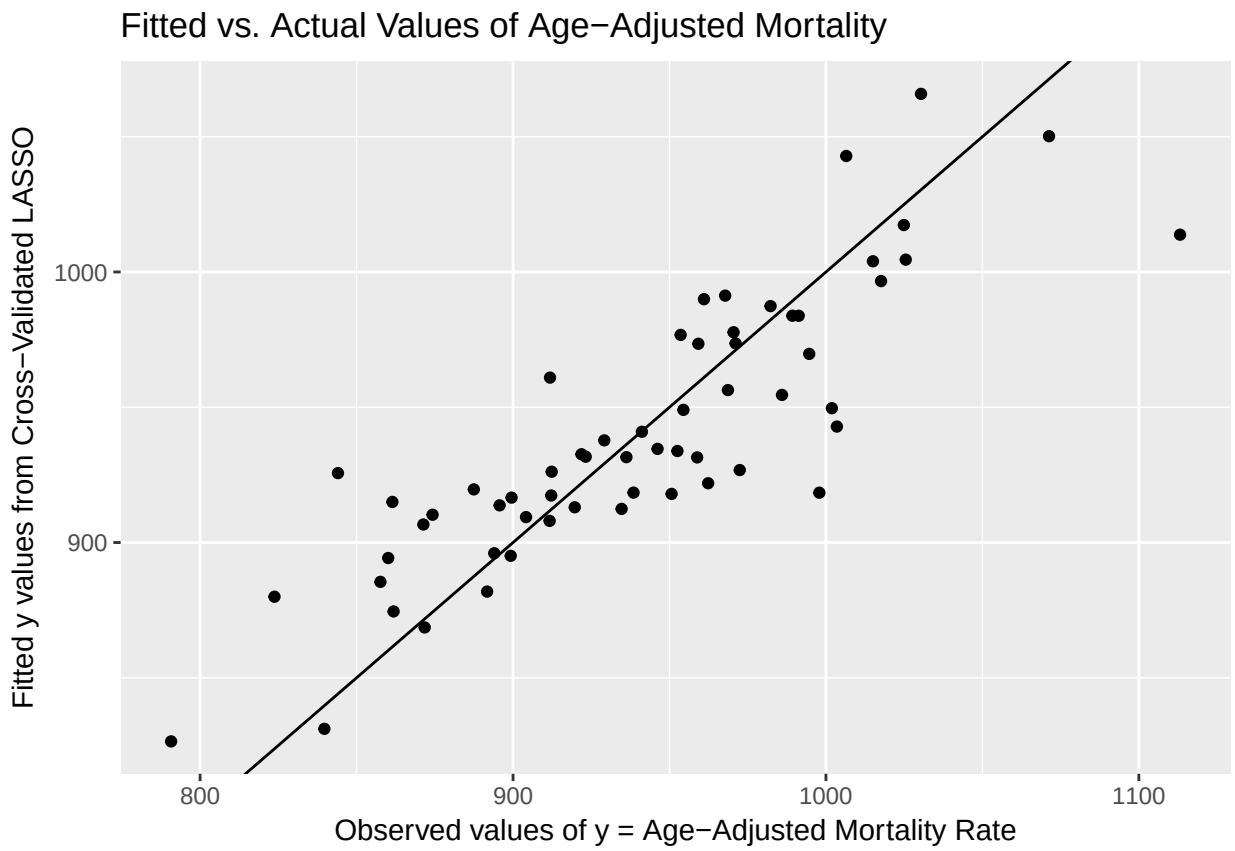
Note that by this cross-validated lasso selection, not only are the coefficients for the 8 variables remaining in the model shrunken, but variables x4, x5, x10, x11, x12, x13 and x15 are all dropped from the model, and model x8 almost is, as well.

```
poll_fits <- predict.lars(lasso_p1, preds, s=frac,
                          type="fit", mode="fraction")
round(poll_fits$fit,3)
```

```
 [1]  932.627  918.415  921.904  987.396 1050.184 1065.837  912.424
 [8]  916.605  949.647  926.168  996.625 1017.362  977.730  954.550
[15]  931.455  894.263  931.551  868.599  973.471  940.937  881.867
[22]  906.666  973.609  919.640  933.821  956.352  913.018  925.650
[29]  874.528  983.829 1042.870  915.002  937.760  885.464  989.947
[36]  931.709 1013.795  969.729 1003.962  983.813  896.042  918.446
[43]  934.609 1004.565  910.273  976.747  831.132  907.996  826.485
[50]  895.082  909.398  917.969  926.777  917.381  991.266  879.972
[57]  942.867  913.737  960.952  949.030
```

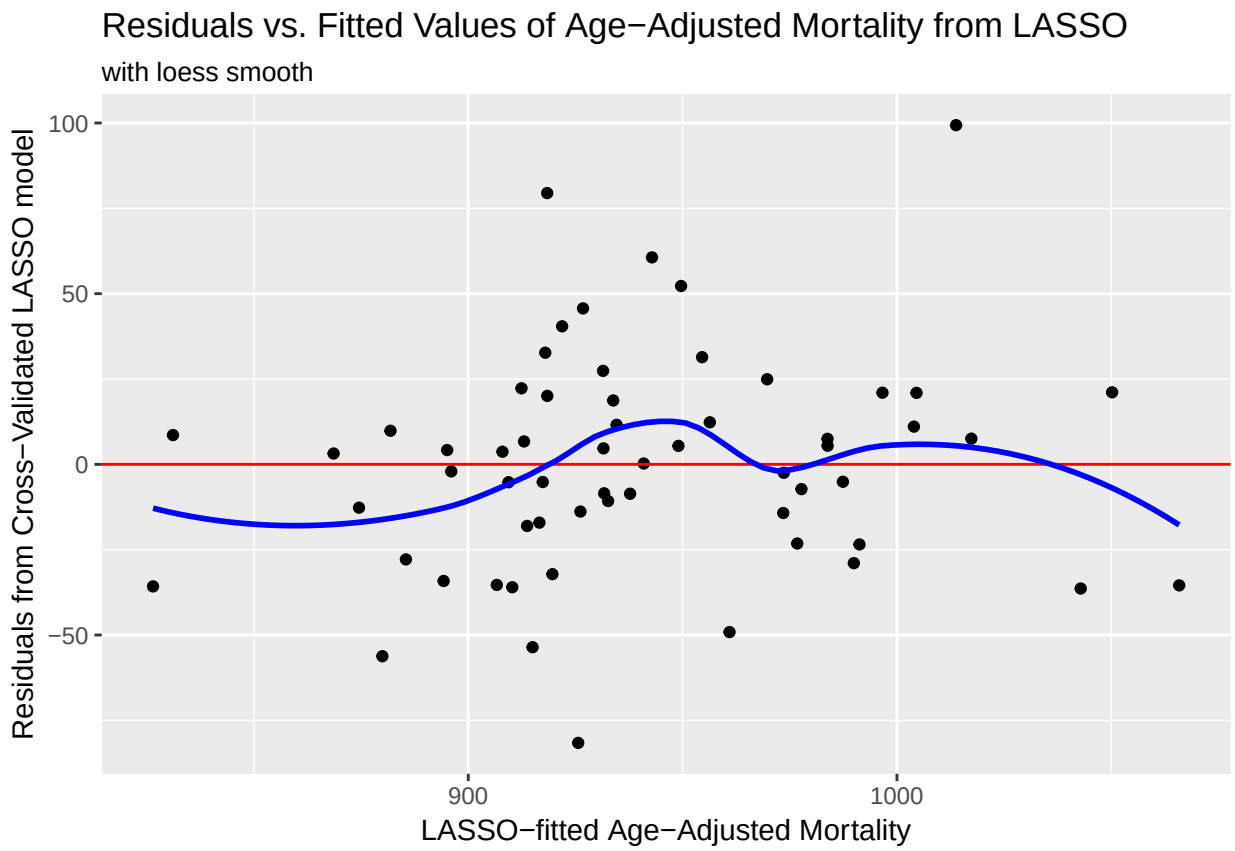
Here's a plot of the actual pollution y values, against these fitted values.

```
poll_lasso_res <- data_frame(fitted = poll_fits$fit,
                             actual = pollution$y,
                             resid = actual - fitted)

ggplot(poll_lasso_res, aes(x = actual, y = fitted)) +
    geom_point() +
    geom_abline(slope = 1, intercept = 0) +
    labs(y = "Fitted y values from Cross-Validated LASSO",
         x = "Observed values of y = Age-Adjusted Mortality Rate",
         title = "Fitted vs. Actual Values of Age-Adjusted Mortality")
```



And now, here's a plot or residuals vs. fitted values.

```
ggplot(poll_lasso_res, aes(x = fitted, y = resid)) +
    geom_point() +
    geom_hline(yintercept = 0, col = "red") +
    geom_smooth(method = "loess", col = "blue", se = F) +
    labs(x = "LASSO-fitted Age-Adjusted Mortality",
         y = "Residuals from Cross-Validated LASSO model",
         title = "Residuals vs. Fitted Values of Age-Adjusted Mortality from LASSO",
         subtitle = "with loess smooth")
```


Residuals vs. Fitted Values of Age−Adjusted Mortality from LASSO
with loess smooth
Residuals from Cross−Validated LASSO model
100
50
0
−50
900
1000
LASSO−fitted Age−Adjusted Mortality

Chapter 12

Logistic Regression and the `resect` data

12.1 The `resect` data

My source for these data was Riffenburgh (2006). The data describe 134 patients who had undergone resection of the tracheal carina (most often this is done to address tumors in the trachea), and the `resect.csv` data file contains the following variables:

- `id` = a patient ID #,
- `age`= the patient's age at surgery,
- `prior` = prior tracheal surgery (1 = yes, 0 = no),
- `resection` = extent of the resection (in cm),
- `intubated` = whether intubation was required at the end of surgery (1 = yes, 0 = no), and
- `died` = the patient's death status (1 = dead, 0 = alive).

```
resect %>% group_by(died) %>% skim(-id)
```

```
Skim summary statistics
 n obs: 134
 n variables: 6
 group variables: died

Variable type: integer
 died  variable missing complete   n   mean    sd p0 p25 median p75 p100
    0       age       0      117 117 48.05  16.01  8  36     51  62   80
    0 intubated       0      117 117  0.068  0.25  0   0      0   0    1
    0     prior       0      117 117  0.24   0.43  0   0      0   0    1
    1       age       0       17  17 46.41  14.46 26  33     46  60   66
    1 intubated       0       17  17  0.65   0.49  0   0      1   1    1
    1     prior       0       17  17  0.35   0.49  0   0      0   1    1

Variable type: numeric
 died  variable missing complete   n mean   sd p0 p25 median p75 p100
    0 resection       0      117 117 2.82 1.21  1 2      2.5 3.5    6
    1 resection       0       17  17 3.97 1     2 3.5    4   4.5    6
```

We have no missing data, and 17 of the 134 patients died. Our goal will be to understand the characteristics of the patients, and how they relate to the binary outcome of interest, death.

12.2 Running A Simple Logistic Regression Model

In the most common scenario, a logistic regression model is used to predict a binary outcome (which can take on the values 0 or 1.) We will eventually fit a logistic regression model in two ways.

1. Through the `glm` function in the base package of R (similar to `lm` for linear regression)
2. Through the `lrm` function available in the `rms` package (similar to `ols` for linear regression)

We'll focus on the `glm` approach in this Chapter, and save the `lrm` ideas for later.

12.2.1 Logistic Regression Can Be Harder than Linear Regression

- Logistic regression models are fitted using the method of maximum likelihood in `glm`, which requires multiple iterations until convergence is reached.
- Logistic regression models are harder to interpret (for most people) than linear regressions.
- Logistic regression models don't have the same set of assumptions as linear models.
- Logistic regression outcomes (yes/no) carry much less information than quantitative outcomes. As a result, fitting a reasonable logistic regression requires more data than a linear model of similar size.
  - The rule I learned in graduate school was that a logistic regression requires 100 observations to fit an intercept plus another 15 observations for each candidate predictor. That's not terrible, but it's a very large sample size.
  - Frank Harrell recommends that 96 observations + a function of the number of candidate predictors (which depends on the amount of variation in the predictors, but 15 x the number of such predictors isn't too bad if the signal to noise ratio is pretty good) are required just to get reasonable confidence intervals around your predictions.
    * In a twitter note, Frank suggests that 96 + 8 times the number of candidate parameters might be reasonable so long as the smallest cell of interest (combination of an outcome and a split of the covariates) is 96 or more observations.
  - Peduzzi et al. (1996) suggest that if we let π be the smaller of the proportions of "yes" or "no" cases in the population of interest, and k be the number of inputs under consideration, then $N = 10k/\pi$ is the minimum number of cases to include, except that if N < 100 by this standard, you should increase it to 100, according to Long (1997).
    * That suggests that if you have an outcome that happens 10% of the time, and you are running a model with 3 predictors, then you could get away with $(10 \times 3)/(0.10) = 300$ observations, but if your outcome happened 40% of the time, you could get away with only $(10 \times 3)/(0.40) = 75$ observations, which you'd round up to 100.

12.2.2 Obtaining the fitted equation

We'll begin by attempting to predict death based on the extent of the resection.

```
res_modA <- glm(died ~ resection, data=resect,
                family="binomial"(link="logit"))

res_modA
```

```
Call:  glm(formula = died ~ resection, family = binomial(link = "logit"),
    data = resect)

Coefficients:
(Intercept)    resection
    -4.4337       0.7417
```

```
Degrees of Freedom: 133 Total (i.e. Null);  132 Residual
Null Deviance:      101.9
Residual Deviance: 89.49    AIC: 93.49
```

Note that the `logit` link is the default approach with the `binomial` family, so we could also have used:

```
res_modA <- glm(died ~ resection, data = resect,
                family = "binomial")
```

which yields the same model.

12.2.3 Interpreting the Coefficients of a Logistic Regression Model

Our model is:

$$logit(died = 1) = log\left(\frac{Pr(died = 1)}{1 - Pr(died = 1)}\right) = \beta_0 + \beta_1 x = -4.4337 + 0.7417 \times resection$$

The predicted log odds of death for a subject with a resection of 4 cm is:

$$log\left(\frac{Pr(died = 1)}{1 - Pr(died = 1)}\right) = -4.4337 + 0.7417 \times 4 = -1.467$$

The predicted odds of death for a subject with a resection of 4 cm is thus:

$$\frac{Pr(died = 1)}{1 - Pr(died = 1)} = e^{-4.4337+0.7417\times 4} = e^{-1.467} = 0.2306$$

Since the odds are less than 1, we should find that the probability of death is less than 1/2. With a little algebra, we see that the predicted probability of death for a subject with a resection of 4 cm is:

$$Pr(died = 1) = \frac{e^{-4.4337+0.7417\times 4}}{1 + e^{-4.4337+0.7417\times 4}} = \frac{e^{-1.467}}{1 + e^{-1.467}} = \frac{0.2306}{1.2306} = 0.187$$

In general, we can frame the model in terms of a statement about probabilities, like this:

$$Pr(died = 1) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}} = \frac{e^{-4.4337+0.7417\times resection}}{1 + e^{-4.4337+0.7417\times resection}}$$

and so by subtituting in values for `resection`, we can estimate the model's fitted probabilities of death.

12.2.4 Using `predict` to describe the model's fits

To obtain these fitted odds and probabilities in R, we can use the `predict` function.

- The default predictions are on the scale of the log odds. These predictions are also available through the `type = "link"` command within the `predict` function for a generalized linear model like logistic regression.
- Here are the predicted log odds of death for a subject (Sally) with a 4 cm resection and a subject (Harry) who had a 5 cm resection.

```
predict(res_modA, newdata = data_frame(resection = c(4,5)))
```

```
         1          2
-1.4669912 -0.7253027
```

- We can also obtain predictions for each subject on the original response (here, probability) scale, backing out of the logit link.

```
predict(res_modA, newdata = data_frame(resection = c(4, 5)),
        type = "response")
```

```
        1         2
0.1874004 0.3262264
```

So the predicted probability of death is 0.187 for Sally, the subject with a 4 cm resection, and 0.326 for Harry, the subject with a 5 cm resection.

12.2.5 Odds Ratio interpretation of Coefficients

Often, we will exponentiate the estimated slope coefficients of a logistic regression model to help us understand the impact of changing a predictor on the odds of our outcome.

```
exp(coef(res_modA))
```

```
(Intercept)   resection
 0.01186995  2.09947754
```

To interpret this finding, suppose we have two subjects, Harry and Sally. Harry had a resection that was 1 cm larger than Sally. This estimated coefficient suggests that the estimated odds for death associated with Harry is 2.099 times larger than the odds for death associated with Sally. In general, the odds ratio comparing two subjects who differ by 1 cm on the resection length is 2.099.

To illustrate, again let's assume that Harry's resection was 5 cm, and Sally's was 4 cm. Then we have:

$$log\left(\frac{Pr(Harrydied)}{1-Pr(Harrydied)}\right) = -4.4337+0.7417\times5 = -0.7253, log\left(\frac{Pr(Sallydied)}{1-Pr(Sallydied)}\right) = -4.4337+0.7417\times4 = -1.4667.$$

which implies that our estimated odds of death for Harry and Sally are:

$$Odds(Harrydied) = \frac{Pr(Harrydied)}{1-Pr(Harrydied)} = e^{-4.4337+0.7417\times5} = e^{-0.7253} = 0.4842 Odds(Sallydied) = \frac{Pr(Sallydied)}{1-Pr(Sallydied)} = e$$

and so the odds ratio is:

$$OR = \frac{Odds(Harrydied)}{Odds(Sallydied)} = \frac{0.4842}{0.2307} = 2.099$$

- If the odds ratio was 1, that would mean that Harry and Sally had the same estimated odds of death, and thus the same estimated probability of death, despite having different sizes of resections.
- Since the odds ratio is greater than 1, it means that Harry has a higher estimated odds of death than Sally, and thus that Harry has a higher estimated probability of death than Sally.
- If the odds ratio was less than 1, it would mean that Harry had a lower estimated odds of death than Sally, and thus that Harry had a lower estimated probability of death than Sally.

Remember that the odds ratio is a fraction describing two positive numbers (odds can only be non-negative) so that the smallest possible odds ratio is 0.

12.2.6 Interpreting the rest of the model output from `glm`

```
res_modA
```

```
Call:  glm(formula = died ~ resection, family = "binomial", data = resect)

Coefficients:
(Intercept)    resection
    -4.4337       0.7417

Degrees of Freedom: 133 Total (i.e. Null);  132 Residual
Null Deviance:      101.9
Residual Deviance: 89.49    AIC: 93.49
```

In addition to specifying the logistic regression coefficients, we are also presented with information on degrees of freedom, deviance (null and residual) and AIC.

- The degrees of freedom indicate the sample size.
  - Recall that we had $n = 134$ subjects in the data. The "Null" model includes only an intercept term (which uses 1 df) and we thus have n - 1 (here 133) degrees of freedom available for estimation.
  - In our `res_modA` model, a logistic regression is fit including a single slope (resection) and an intercept term. Each uses up one degree of freedom to build an estimate, so we have n - 2 = 134 - 2 = 132 residual df remaining.
- The AIC or Akaike Information Criterion (lower values are better) is also provided. This is helpful if we're comparing multiple models for the same outcome.

12.2.7 Deviance and Comparing Our Model to the Null Model

- The deviance (a measure of the model's *lack of fit*) is available for both the null model (the model with only an intercept) and for our model (`res_modA`) predicting our outcome, mortality.
- The deviance test, though available in R (see below) isn't really a test of whether the model works well. Instead, it assumes the model is true, and then tests to see if the coefficients are detectably different from zero. So it isn't of much practical use.
  - To compare the `deviance` statistics, we can subtract the residual deviance from the null deviance to decribe the impact of our model on fit.
  - Null Deviance - Residual Deviance can be compared to a χ^2 distribution with Null DF - Residual DF degrees of freedom to obtain a global test of the in-sample predictive power of our model.
  - We can see this comparison more directly by running `anova` on our model:

```
anova(res_modA)
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: died

Terms added sequentially (first to last)

          Df Deviance Resid. Df Resid. Dev
NULL                        133    101.943
resection  1    12.45       132     89.493
```

To complete a deviance test and obtain a p value, we can run the following code to estimate the probability that a chi-square distribution with a single degree of freedom would exhibit an improvement in deviance as large as 12.45.

```
pchisq(12.45, 1, lower.tail = FALSE)
```

```
[1] 0.0004179918
```

The p value for the deviance test here is about 0.0004. But, again, this isn't a test of whether the model is any good - it assumes the model is true, and then tests some consequences.

- Specifically, it tests whether (if the model is TRUE) some of the model's coefficients are non-zero.
- That's not so practially useful, so I discourage you from performing global tests of a logistic regression model with a deviance test.

12.2.8 Using glance with a logistic regression model

We can use the `glance` function from the `broom` package to obtain the null and residual deviance and degrees of freedom. Note that the deviance for our model is related to the log likelihood by -2*`logLik`.

```
glance(res_modA)
```

```
  null.deviance df.null    logLik      AIC     BIC deviance df.residual
1      101.9431     133 -44.74646 93.49292 99.2886 89.49292         132
```

The `glance` result also provides the AIC, and the BIC (Bayes Information Criterion), each of which is helpful in understanding comparisons between multiple models for the same outcome (with smaller values of either criterion indicating better models.) The AIC is based on the deviance, but penalizes you for making the model more complicated. The BIC does the same sort of thing but with a different penalty.

Again we see that we have a null deviance of 101.94 on 133 degrees of freedom. Including the `resection` information in the model decreased the deviance to 89.49 points on 132 degrees of freedom, so that's a decrease of 12.45 points while using one degree of freedom, a statistically significant reduction in deviance.

12.3 Interpreting the Model Summary

Let's get a more detailed summary of our `res_modA` model, including 95% confidence intervals for the coefficients:

```
summary(res_modA)
```

```

Call:
glm(formula = died ~ resection, family = "binomial", data = resect)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.1844  -0.5435  -0.3823  -0.2663   2.4501

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -4.4337     0.8799  -5.039 4.67e-07 ***
resection     0.7417     0.2230   3.327 0.000879 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 101.943  on 133  degrees of freedom
Residual deviance:  89.493  on 132  degrees of freedom
```

```
AIC: 93.493

Number of Fisher Scoring iterations: 5
```

```
confint(res_modA, level = 0.95)
```

```
Waiting for profiling to be done...
                2.5 %    97.5 %
(Intercept) -6.344472 -2.855856
resection    0.322898  1.208311
```

Some elements of this summary are very familiar from our work with linear models.

- We still have a five-number summary of residuals, although these are called *deviance* residuals.
- We have a table of coefficients with standard errors, and hypothesis tests, although these are Wald z-tests, rather than the t tests we saw in linear modeling.
- We have a summary of global fit in the comparison of null deviance and residual deviance, but without a formal p value. And we have the AIC, as discussed above.
- We also have some new items related to a *dispersion* parameter and to the number of Fisher Scoring Iterations.

Let's walk through each of these elements.

12.3.1 Wald Z tests for Coefficients in a Logistic Regression

The coefficients output provides the estimated coefficients, and their standard errors, plus a Wald Z statistic, which is just the estimated coefficient divided by its standard error. This is compared to a standard Normal distribution to obtain the two-tailed p values summarized in the `Pr(>|z|)` column.

- The interesting result is `resection`, which has a Wald Z = 3.327, yielding a p value of 0.00088.
- The hypotheses being tested here are H_0_: `resection` does not have an effect on the log odds of `died` vs. H_A_: `resection` does have such an effect.
- Another way of stating this is that the p value assesses whether the estimated coefficient of `resection`, 0.7417, is statistically detectably different from 0. If the coefficient (on the logit scale) for `resection` was truly 0, this would mean that:
  - the log odds of death did not change based on the `resection` size,
  - the odds of death were unchanged based on the `resection` size (the odds ratio would be 1), and
  - the probability of death was unchanged based on the `resection` size.

In our case, we have a statistically detectable change in the log odds of `died` associated with changes in `resection`, according to this p value. We conclude that `resection` size is associated with a positive impact on death rates (death rates are generally higher for people with larger resections.)

12.3.2 Confidence Intervals for the Coefficients

As in linear regression, we can obtain 95% confidence intervals (to get other levels, change the `level` parameter in `confint`) for the intercept and slope coefficients.

Here, we see, for example, that the coefficient of `resection` has a point estimate of 0.7417, and a confidence interval of (0.3229, 1.208). Since this is on the logit scale, it's not that interpretable, but we will often exponentiate the model and its confidence interval to obtain a more interpretable result on the odds ratio scale.

```
exp(coef(res_modA))
```

```
(Intercept)   resection
 0.01186995  2.09947754
```

```
exp(confint(res_modA))
```

```
Waiting for profiling to be done...
```

```
                  2.5 %     97.5 %
(Intercept) 0.001756429 0.05750655
resection   1.381124459 3.34782604
```

From this output, we can estimate the odds ratio for death associated with a 1 cm increase in resection size is 2.099, with a 95% CI of (1.38, 3.35). - If the odds ratio was 1, it would indicate that the odds of death did not change based on the change in resection size. - Here, it's clear that the estimated odds of death will be larger (odds > 1) for subjects with larger resection sizes. Larger odds of death also indicate larger probabilities of death. This confidence interval indicates that with 95% confidence, we conclude that increases in resection size are associated with statistically detectable increases in the odds of death. - If the odds ratio was less than 1 (remember that it cannot be less than 0) that would mean that subjects with larger resection sizes were associated with smaller estimated odds of death.

12.3.3 Deviance Residuals

In logistic regression, it's certainly a good idea to check to see how well the model fits the data. However, there are a few different types of residuals. The residuals presented here by default are called deviance residuals. Other types of residuals are available for generalized linear models, such as Pearson residuals, working residuals, and response residuals. Logistic regression model diagnostics often make use of multiple types of residuals.

The deviance residuals for each individual subject sum up to the deviance statistic for the model, and describe the contribution of each point to the model likelihood function.

The deviance residual, d_i, for the i$^{\text{th}}$ observation in a model predicting y_i (a binary variable), with the estimate being $\hat{\pi}_i$ is:

$$d_i = s_i \sqrt{-2[y_i log\hat{\pi}_i + (1 - y_i) log(1 - \hat{\pi}_i)]},$$

where s_i is 1 if $y_i = 1$ and $s_i = -1$ if $y_i = 0$.

Again, the sum of the deviance residuals is the deviance.

12.3.4 Dispersion Parameter

The dispersion parameter is taken to be 1 for `glm` fit using either the `binomial` or `Poisson` families. For other sorts of generalized linear models, the dispersion parameter will be of some importance in estimating standard errors sensibly.

12.3.5 Fisher Scoring iterations

The solution of a logistic regression model involves maximizing a likelihood function. Fisher's scoring algorithm in our `res_modA` needed five iterations to perform the logistic regression fit. All that this tells you is that the model converged, and didn't require a lot of time to do so.

12.4 Plotting a Simple Logistic Regression Model

Let's plot the logistic regression model `res_modA` for `died` using the extent of the resection in terms of probabilities. We can use either of two different approaches:

- we can plot the fitted values from our specific model against the original data, using the `augment` function from the `broom` package, or
- we can create a smooth function called `binomial_smooth` that plots a simple logistic model in an analogous way to `geom_smooth(method = "lm")` for a simple linear regression.

12.4.1 Using `augment` to capture the fitted probabilities

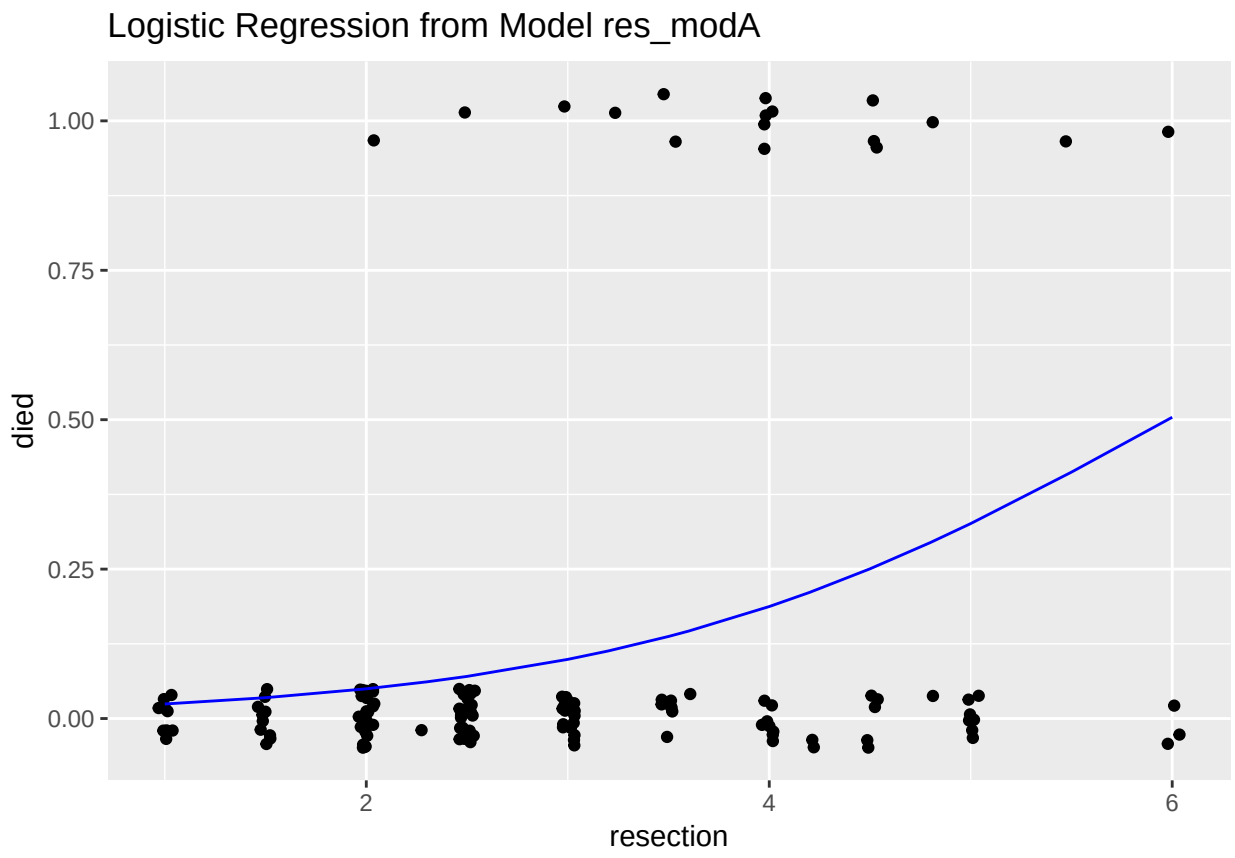
```
res_A_aug <- augment(res_modA, resect,
                     type.predict = "response")
head(res_A_aug)
```

```
  id age prior resection intubated died    .fitted    .se.fit     .resid
1  1  34     1       2.5         0    0 0.07046791 0.02562381 -0.3822929
2  2  57     0       5.0         0    0 0.32622637 0.08605551 -0.8886631
3  3  60     1       4.0         1    1 0.18740037 0.04269795  1.8300317
4  4  62     1       4.2         0    0 0.21104240 0.04871389 -0.6885386
5  5  28     0       6.0         1    1 0.50409637 0.14302982  1.1704596
6  6  52     0       3.0         0    0 0.09897375 0.02867196 -0.4565542
         .hat    .sigma      .cooksd .std.resid
1 0.010024061 0.8258481 0.0003876961 -0.3842235
2 0.033691765 0.8227475 0.0087350915 -0.9040227
3 0.011972088 0.8107264 0.0265893468  1.8410857
4 0.014252277 0.8243062 0.0019617278 -0.6934983
5 0.081835623 0.8196110 0.0477480056  1.2215077
6 0.009218619 0.8255581 0.0005157780 -0.4586733
```

This approach augments the `resect` data set with fitted, residual and other summaries of each observation's impact on the fit, using the "response" type of prediction, which yields the fitted probabilities in the `.fitted` column.

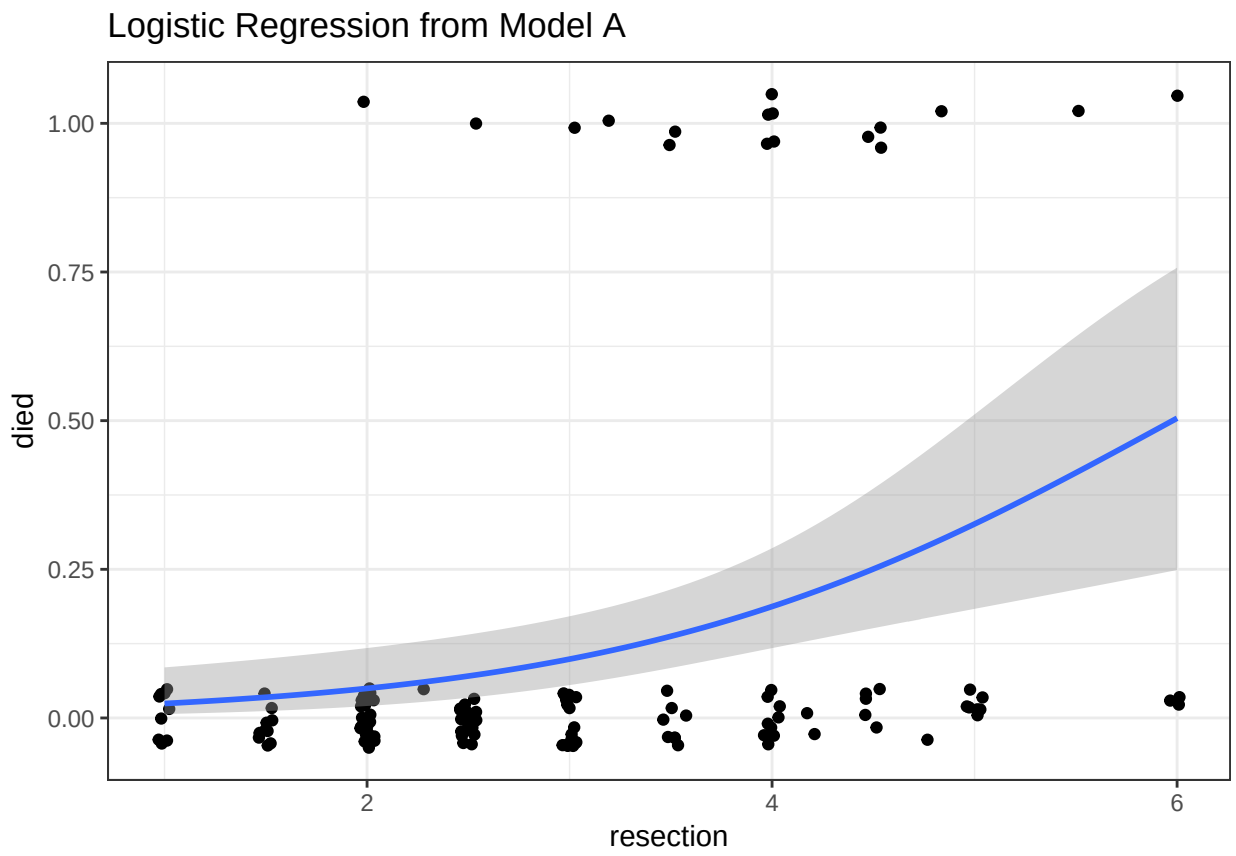
12.4.2 Plotting a Logistic Regression Model's Fitted Values

```
ggplot(res_A_aug, aes(x = resection, y = died)) +
    geom_jitter(height = 0.05) +
    geom_line(aes(x = resection, y = .fitted),
              col = "blue") +
    labs(title = "Logistic Regression from Model res_modA")
```



12.4.3 Plotting a Simple Logistic Model using `binomial_smooth`

```
binomial_smooth <- function(...) {
  geom_smooth(method = "glm",
              method.args = list(family = "binomial"), ...)
}

ggplot(resect, aes(x = resection, y = died)) +
  geom_jitter(height = 0.05) +
  binomial_smooth() + ## ...smooth(se=FALSE) to leave out interval
  labs(title = "Logistic Regression from Model A") +
  theme_bw()
```



As expected, we see an increase in the model probability of death as the extent of the resection grows larger.

12.5 Receiver Operating Characteristic Curve Analysis

One way to assess the predictive accuracy within the model development sample in a logistic regression is to consider an analyses based on the receiver operating characteristic (ROC) curve. ROC curves are commonly used in assessing diagnoses in medical settings, and in signal detection applications.

The accuracy of a "test" can be evaluated by considering two types of errors: false positives and false negatives.

In our `res_modA` model, we use `resection` size to predict whether the patient `died`. Suppopse we established a value R, so that if the resection size was larger than R cm, we would predict that the patient `died`, and otherwise we would predict that the patient did not die.

A good outcome of our model's "test", then, would be when the resection size is larger than R for a patient who actually died. Another good outcome would be when the resection size is smaller than R for a patient who survived.

But we can make errors, too.

- A false positive error in this setting would occur when the resection size is larger than R (so we predict the patient dies) but in fact the patient does not die.
- A false negative error in this case would occur when the resection size is smaller than R (so we predict the patient survives) but in fact the patient dies.

Formally, the true positive fraction (TPF) for a specific resection cutoff R, is the probability of a positive test (a prediction that the patient will die) among the people who have the outcome died = 1 (those who

actually die).

$$TPF(R) = Pr(resection > R|subjectdied)$$

Similarly, the false positive fraction (FPF) for a specific cutoff R is the probability of a positive test (prediction that the patient will die) among the people with died = 0 (those who don't actually die)

$$FPF(R) = Pr(resection > R|subjectdidnotdie)$$

The True Positive Rate is referred to as the sensitivity of a diagnostic test, and the True Negative rate (1 - the False Positive rate) is referred to as the specificity of a diagnostic test.

Since the cutoff R is not fixed in advanced, we can plot the value of TPF (on the y axis) against FPF (on the x axis) for all possible values of R, and this is what the ROC curve is. Others refer to the Sensitivity on the Y axis, and 1-Specificity on the X axis, and this is the same idea.

Before we get too far into the weeds, let me show you some simple situations so you can understand what you might learn from the ROC curve. The web page http://blog.yhat.com/posts/roc-curves.html provides source materials.

12.5.1 Interpreting the Area under the ROC curve

The AUC or Area under the ROC curve is the amount of space underneath the ROC curve. Often referred to as the c statistic, the AUC represents the quality of your TPR and FPR overall in a single number. The C statistic ranges from 0 to 1, with C = 0.5 for a prediction that is no better than random guessing, and C = 1 for a perfect prediction model.

Next, I'll build a simulation to demonstrate several possible ROC curves in the sections that follow.

```
set.seed(432999)
sim.temp <- data_frame(x = rnorm(n = 200),
                       prob = exp(x)/(1 + exp(x)),
                       y = as.numeric(1 * runif(200) < prob))

sim.temp <- sim.temp %>%
    mutate(p_guess = 1,
           p_perfect = y,
           p_bad = exp(-2*x) / (1 + exp(-2*x)),
           p_ok = prob + (1-y)*runif(1, 0, 0.05),
           p_good = prob + y*runif(1, 0, 0.27))
```

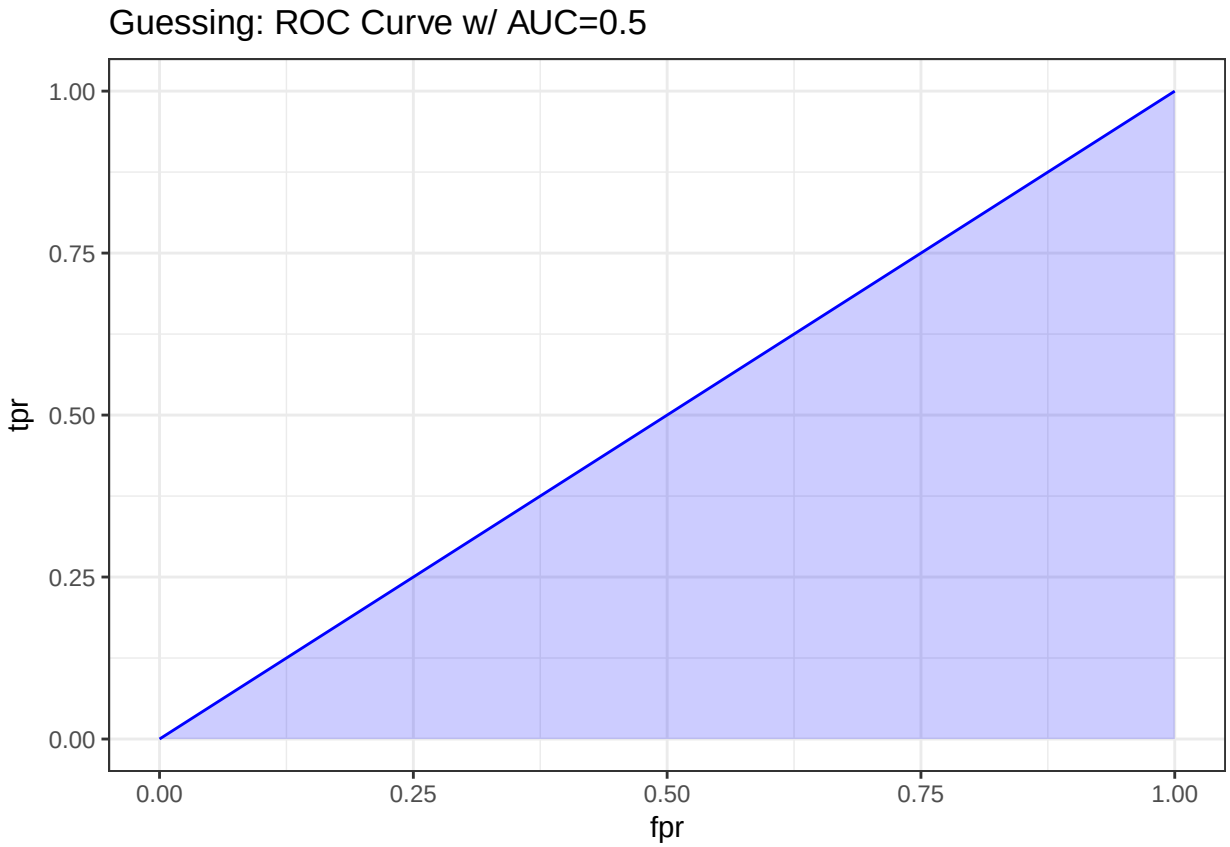
12.5.1.1 What if we are guessing?

If we're guessing completely at random, then the model should correctly classify a subject (as died or not died) about 50% of the time, so the TPR and FPR will be equal. This yields a diagonal line in the ROC curve, and an area under the curve (C statistic) of 0.5.

There are several ways to do this on the web, but I'll show this one, which has some bizarre code, but that's a function of using a package called `ROCR` to do the work. It comes from this link

```
pred_guess <- prediction(sim.temp$p_guess, sim.temp$y)
perf_guess <- performance(pred_guess, measure = "tpr", x.measure = "fpr")
auc_guess <- performance(pred_guess, measure="auc")

auc_guess <- round(auc_guess@y.values[[1]],3)
```

```
roc_guess <- data.frame(fpr=unlist(perf_guess@x.values),
                        tpr=unlist(perf_guess@y.values),
                        model="GLM")

ggplot(roc_guess, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2, fill = "blue") +
    geom_line(aes(y=tpr), col = "blue") +
    labs(title = paste0("Guessing: ROC Curve w/ AUC=", auc_guess)) +
    theme_bw()
```

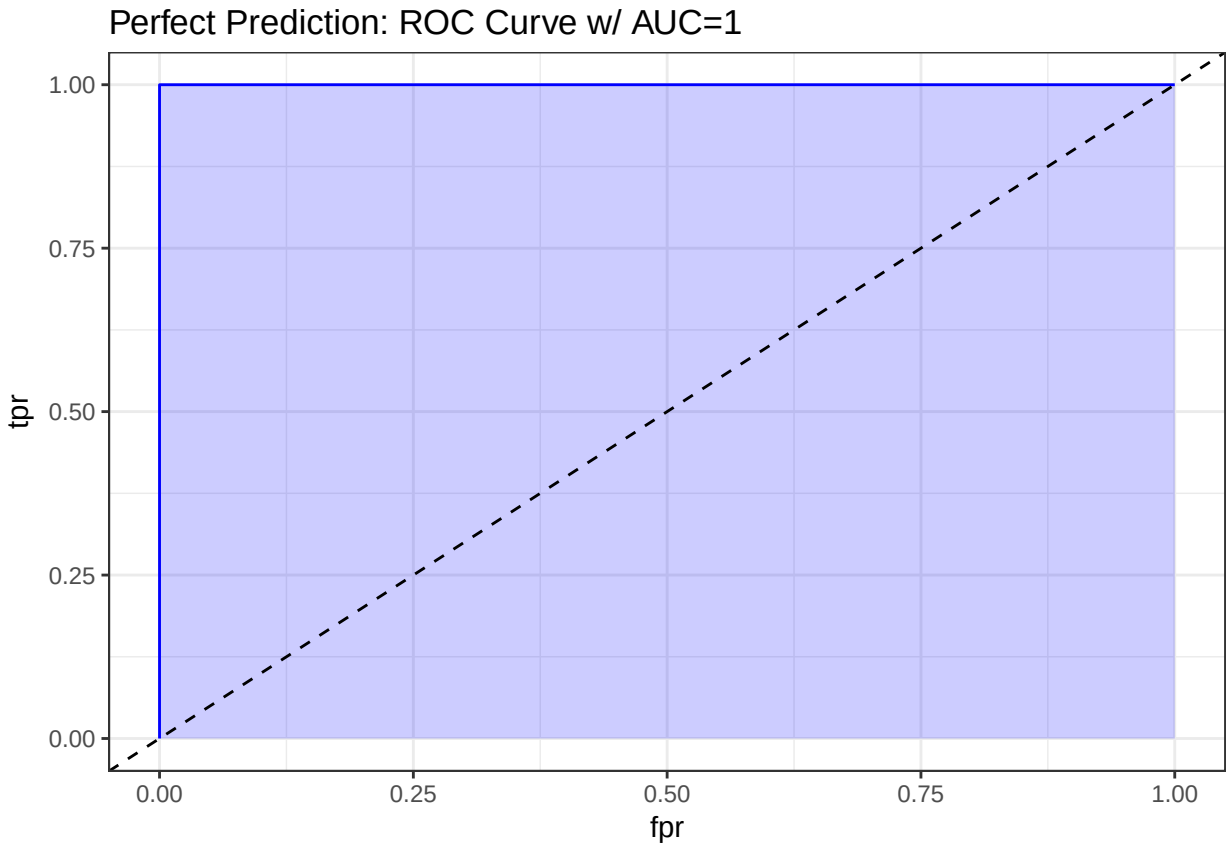


12.5.1.2 What if we classify things perfectly?

If we're classifying subjects perfectly, then we have a TPR of 1 and an FPR of 0. That yields an ROC curve that looks like the upper and left edges of a box. If our model correctly classifies a subject (as died or not died) 100% of the time, the area under the curve (c statistic) will be 1.0. We'll add in the diagonal line here (in a dashed black line) to show how this model compares to random guessing.

```
pred_perf <- prediction(sim.temp$p_perfect, sim.temp$y)
perf_perf <- performance(pred_perf, measure = "tpr", x.measure = "fpr")
auc_perf <- performance(pred_perf, measure="auc")

auc_perf <- round(auc_perf@y.values[[1]],3)
roc_perf <- data.frame(fpr=unlist(perf_perf@x.values),
                        tpr=unlist(perf_perf@y.values),
                        model="GLM")
```

```
ggplot(roc_perf, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2, fill = "blue") +
    geom_line(aes(y=tpr), col = "blue") +
    geom_abline(intercept = 0, slope = 1, lty = "dashed") +
    labs(title = paste0("Perfect Prediction: ROC Curve w/ AUC=", auc_perf)) +
    theme_bw()
```

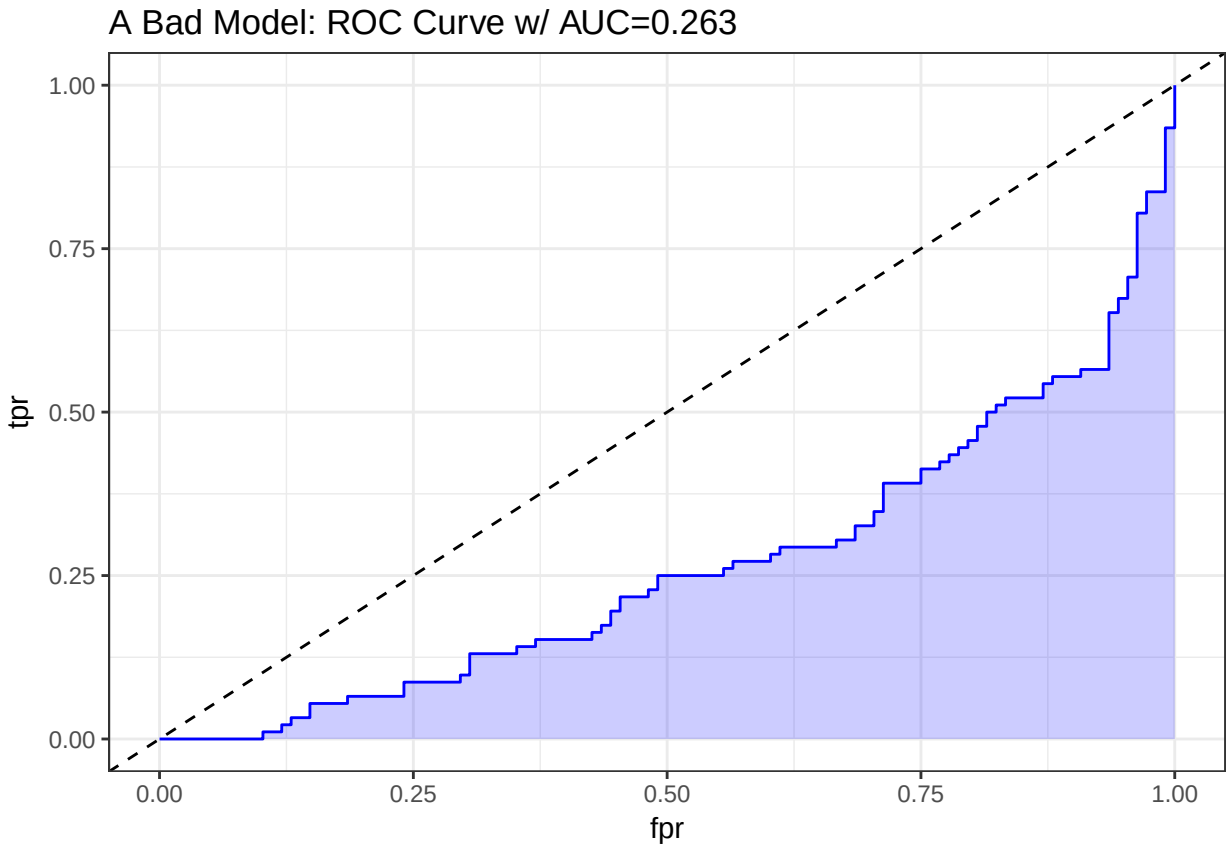


12.5.1.3 What does "worse than guessing" look like?

A bad classifier will appear below and to the right of the diagonal line we'd see if we were completely guessing. Such a model will have a c statistic below 0.5, and will be valueless.

```
pred_bad <- prediction(sim.temp$p_bad, sim.temp$y)
perf_bad <- performance(pred_bad, measure = "tpr", x.measure = "fpr")
auc_bad <- performance(pred_bad, measure="auc")

auc_bad <- round(auc_bad@y.values[[1]],3)
roc_bad <- data.frame(fpr=unlist(perf_bad@x.values),
                      tpr=unlist(perf_bad@y.values),
                      model="GLM")

ggplot(roc_bad, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2, fill = "blue") +
    geom_line(aes(y=tpr), col = "blue") +
    geom_abline(intercept = 0, slope = 1, lty = "dashed") +
```

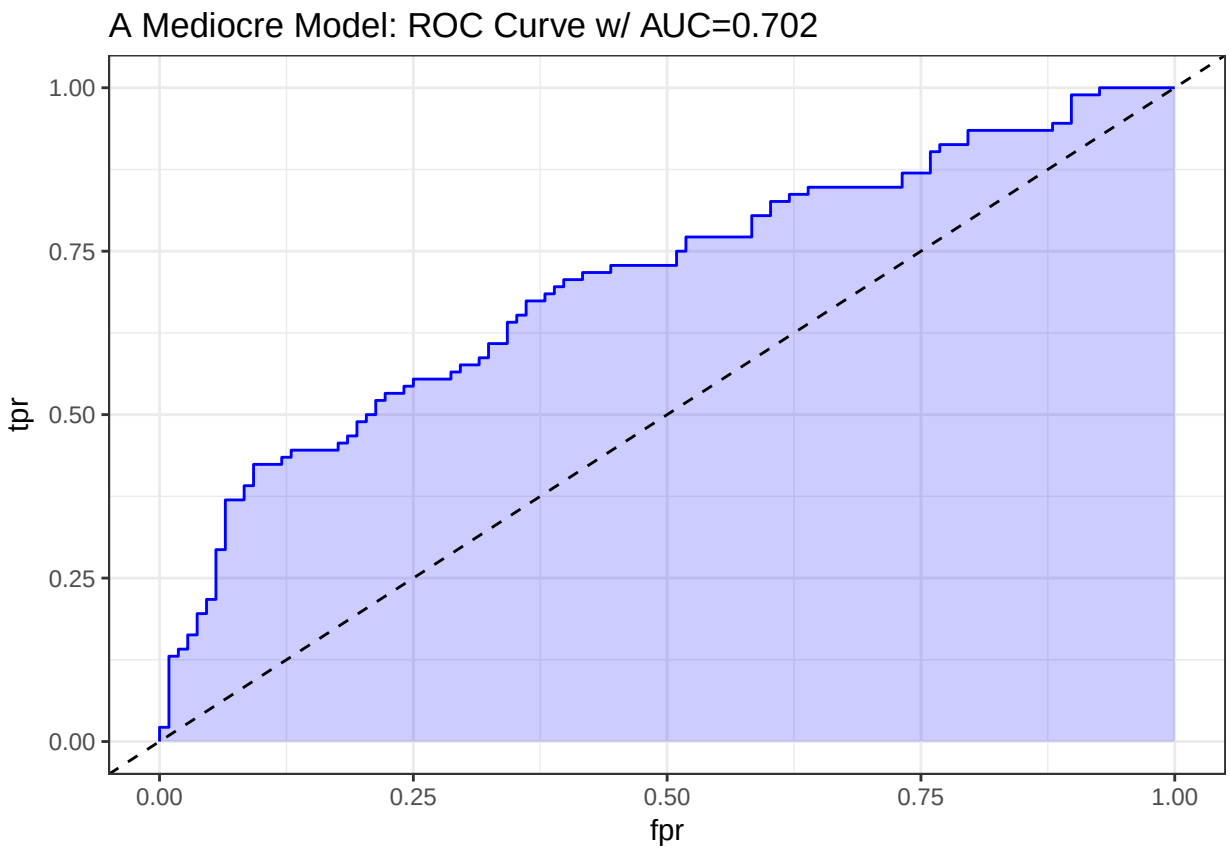
```
    labs(title = paste0("A Bad Model: ROC Curve w/ AUC=", auc_bad)) +
    theme_bw()
```



12.5.1.4 What does "better than guessing" look like?

An "OK" classifier will appear above and to the left of the diagonal line we'd see if we were completely guessing. Such a model will have a c statistic above 0.5, and might have some value. The plot below shows a very fairly poor model, but at least it's better than guessing.

```
pred_ok <- prediction(sim.temp$p_ok, sim.temp$y)
perf_ok <- performance(pred_ok, measure = "tpr", x.measure = "fpr")
auc_ok <- performance(pred_ok, measure="auc")

auc_ok <- round(auc_ok@y.values[[1]],3)
roc_ok <- data.frame(fpr=unlist(perf_ok@x.values),
                     tpr=unlist(perf_ok@y.values),
                     model="GLM")

ggplot(roc_ok, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2, fill = "blue") +
    geom_line(aes(y=tpr), col = "blue") +
    geom_abline(intercept = 0, slope = 1, lty = "dashed") +
    labs(title = paste0("A Mediocre Model: ROC Curve w/ AUC=", auc_ok)) +
    theme_bw()
```



Sometimes people grasp for a rough guide as to the accuracy of a model's predictions based on the area under the ROC curve. A common thought is to assess the C statistic much like you would a class grade.

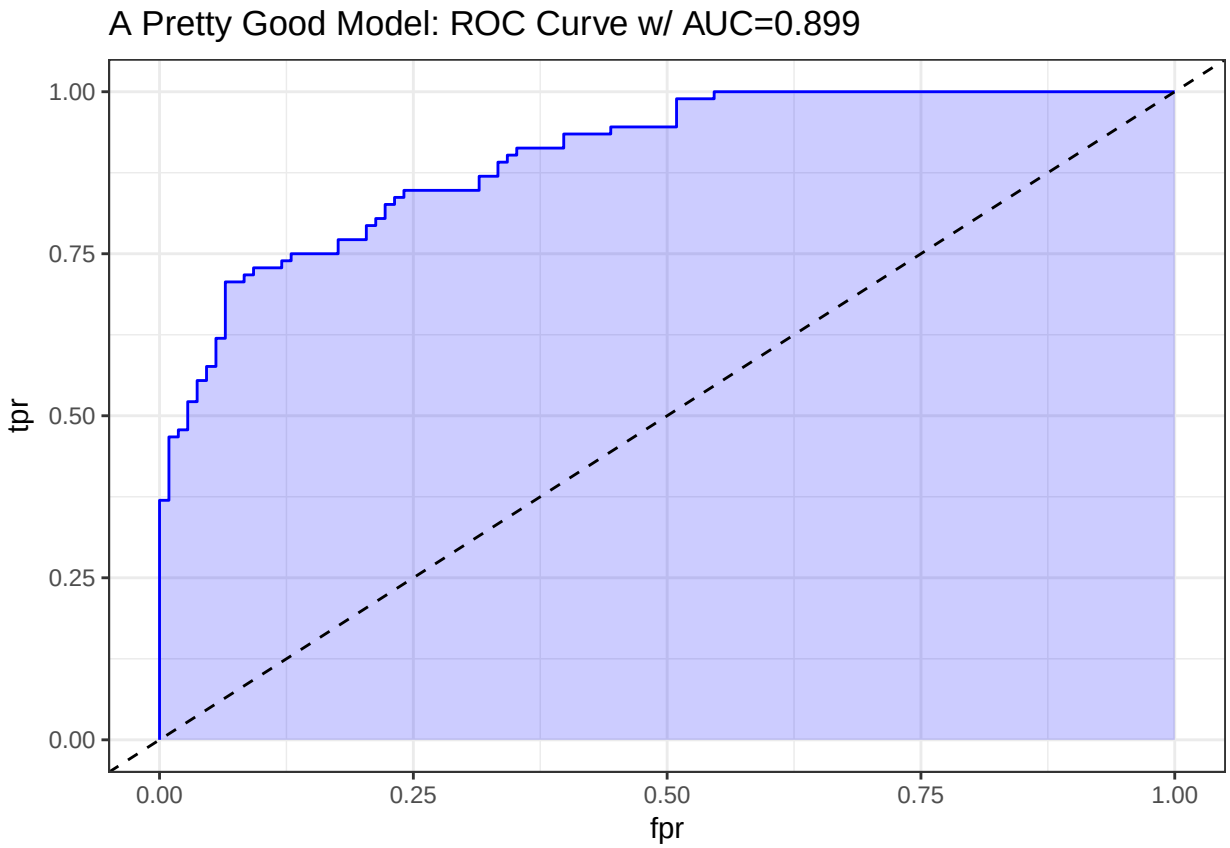
C statistic	Interpretation
0.90 to 1.00	model does an excellent job at discriminating "yes" from "no" (A)
0.80 to 0.90	model does a good job (B)
0.70 to 0.80	model does a fair job (C)
0.60 to 0.70	model does a poor job (D)
0.50 to 0.60	model fails (F)
below 0.50	model is worse than random guessing

12.5.1.5 What does "pretty good" look like?

A strong and good classifier will appear above and to the left of the diagonal line we'd see if we were completely guessing, often with a nice curve that is continually increasing and appears to be pulled up towards the top left. Such a model will have a c statistic well above 0.5, but not as large as 1. The plot below shows a stronger model, which appears substantially better than guessing.

```
pred_good <- prediction(sim.temp$p_good, sim.temp$y)
perf_good <- performance(pred_good, measure = "tpr", x.measure = "fpr")
auc_good <- performance(pred_good, measure="auc")

auc_good <- round(auc_good@y.values[[1]],3)
roc_good <- data.frame(fpr=unlist(perf_good@x.values),
```

```
                        tpr=unlist(perf_good@y.values),
                        model="GLM")

ggplot(roc_good, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2, fill = "blue") +
    geom_line(aes(y=tpr), col = "blue") +
    geom_abline(intercept = 0, slope = 1, lty = "dashed") +
    labs(title = paste0("A Pretty Good Model: ROC Curve w/ AUC=", auc_good)) +
    theme_bw()
```

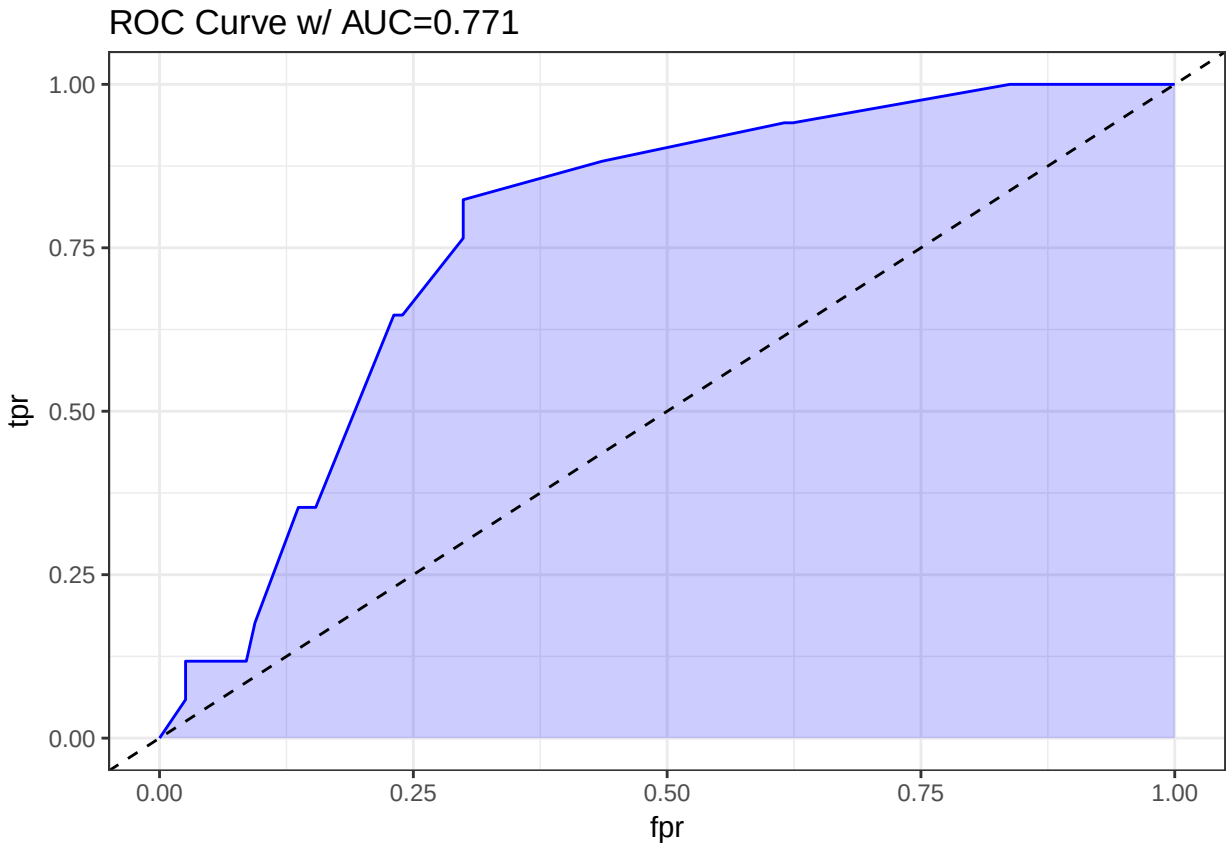


12.6 The ROC Plot for `res_modA`

Let me show you the ROC curve for our `res_modA` model.

```
## requires ROCR package
prob <- predict(res_modA, resect, type="response")
pred <- prediction(prob, resect$died)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
auc <- performance(pred, measure="auc")
## the rest of this code is a little strange
auc <- round(auc@y.values[[1]],3)
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values),
                       model="GLM")
```

```
ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2, fill = "blue") +
    geom_line(aes(y=tpr), col = "blue") +
    geom_abline(intercept = 0, slope = 1, lty = "dashed") +
    labs(title = paste0("ROC Curve w/ AUC=", auc)) +
    theme_bw()
```



Based on the C statistic (AUC = 0.771) this would rank somewhere near the high end of a "fair" predictive model by this standard, not quite to the level of a "good" model.

12.7 Assessing Residual Plots from Model A

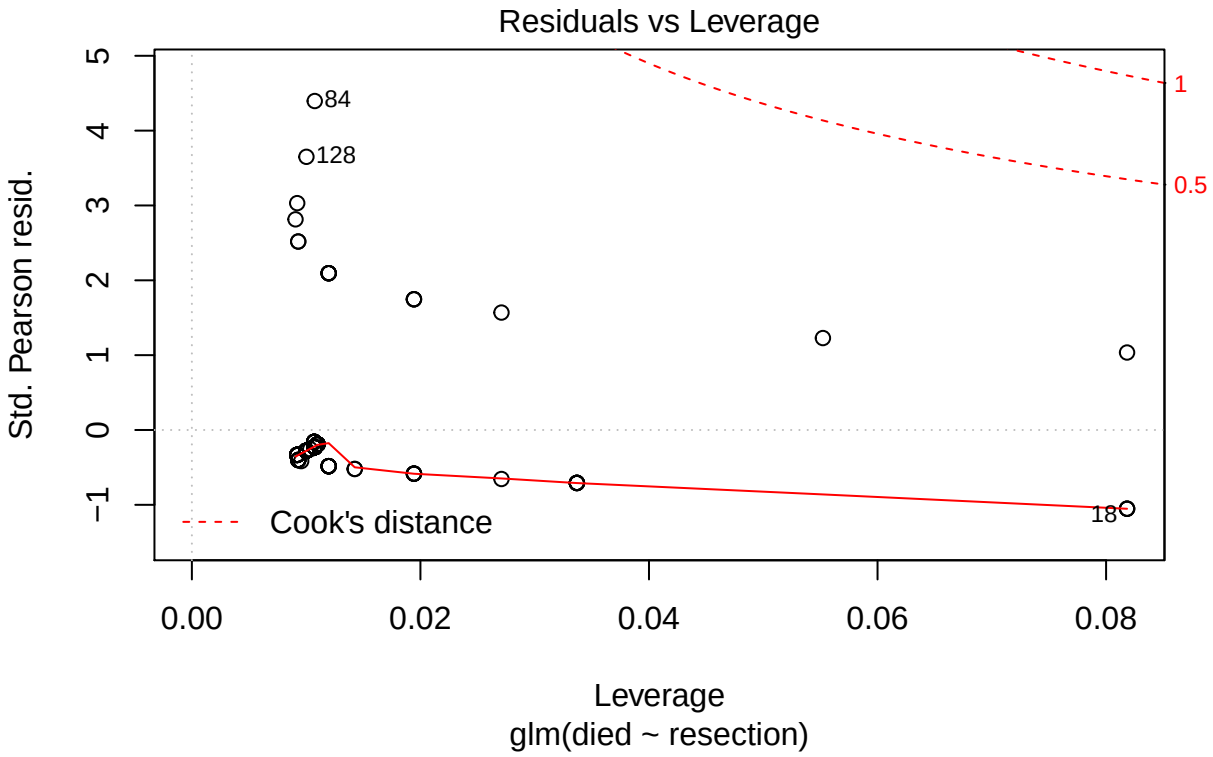
> Residuals are certainly less informative for logistic regression than they are for linear regression: not only do yes/no outcomes inherently contain less information than continuous ones, but the fact that the adjusted response depends on the fit hampers our ability to use residuals as external checks on the model.
>
> This is mitigated to some extent, however, by the fact that we are also making fewer distributional assumptions in logistic regression, so there is no need to inspect residuals for, say, skewness or heteroskedasticity.

- Patrick Breheny, University of Kentucky, Slides on GLM Residuals and Diagnostics

The usual residual plots are available in R for a logistic regression model, but most of them are irrelevant in the logistic regression setting. The residuals shouldn't follow a standard Normal distribution, and they will not show constant variance over the range of the predictor variables, so plots looking into those issues aren't helpful.

The only plot from the standard set that we'll look at in many settings is plot 5, which helps us assess influence (via Cook's distance contours), and a measure related to leverage (how unusual an observation is in terms of the predictors) and standardized Pearson residuals.

```
plot(res_modA, which = 5)
```



In this case, I don't see any highly influential points, as no points fall outside of the Cook's distance (0.5 or 1) contours.

12.8 Model B: A "Kitchen Sink" Logistic Regression Model

```
res_modB <- glm(died ~ resection + age + prior + intubated,
                data = resect, family = binomial)

res_modB
```

```
Call:  glm(formula = died ~ resection + age + prior + intubated, family = binomial,
    data = resect)

Coefficients:
(Intercept)    resection          age        prior    intubated
  -5.152886     0.612211     0.001173     0.814691     2.810797

Degrees of Freedom: 133 Total (i.e. Null);  129 Residual
Null Deviance:      101.9
```

```
Residual Deviance: 67.36    AIC: 77.36
```

12.8.1 Comparing Model A to Model B

```
anova(res_modA, res_modB)
```

```
Analysis of Deviance Table

Model 1: died ~ resection
Model 2: died ~ resection + age + prior + intubated
  Resid. Df Resid. Dev Df Deviance
1       132     89.493
2       129     67.359  3   22.134
```

The addition of `age`, `prior` and `intubated` reduces the lack of fit by 22.134 points, at a cost of 3 degrees of freedom.

```
glance(res_modA)
```

```
  null.deviance df.null    logLik      AIC     BIC deviance df.residual
1      101.9431     133 -44.74646 93.49292 99.2886 89.49292         132
```

```
glance(res_modB)
```

```
  null.deviance df.null   logLik     AIC     BIC deviance df.residual
1      101.9431     133 -33.6793 77.3586 91.8478  67.3586         129
```

By either AIC or BIC, the larger model (`res_modB`) looks more effective.

12.8.2 Interpreting Model B

```
summary(res_modB)
```

```

Call:
glm(formula = died ~ resection + age + prior + intubated, family = binomial,
    data = resect)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7831  -0.3741  -0.2386  -0.2014   2.5228

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.152886   1.469453  -3.507 0.000454 ***
resection    0.612211   0.282807   2.165 0.030406 *
age          0.001173   0.020646   0.057 0.954700
prior        0.814691   0.704785   1.156 0.247705
intubated    2.810797   0.658395   4.269 1.96e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 101.943  on 133  degrees of freedom
```

```
Residual deviance:  67.359  on 129  degrees of freedom
AIC: 77.359

Number of Fisher Scoring iterations: 6
```

It appears that the `intubated` predictor adds significant value to the model, by the Wald test.

Let's focus on the impact of these variables through odds ratios.

```
exp(coef(res_modB))
```

```
 (Intercept)    resection          age        prior    intubated
 0.005782692  1.844504859  1.001173503  2.258476846 16.623153519
```

```
exp(confint(res_modB))
```

```
Waiting for profiling to be done...
                   2.5 %     97.5 %
(Intercept) 0.0002408626  0.0837263
resection   1.0804548590  3.3495636
age         0.9618416869  1.0442885
prior       0.5485116610  9.1679931
intubated   4.7473282453 64.6456919
```

At a 5% significance level, we might conclude that:

- larger sized `resection`s are associated with a meaningful rise (est OR: 1.84, 95% CI 1.08, 3.35) in the odds of death, holding all other predictors constant,
- the need for `intubation` at the end of surgery is associated with a substantial rise (est OR: 16.6, 95% CI 4.7, 64.7) in the odds of death, holding all other predictors constant, but that
- older `age` as well as having a `prior` tracheal surgery appears to be associated with an increase in death risk, but not to an extent that we can declare statistically significant.

12.9 Plotting Model B

Let's think about plotting the fitted values from our model, in terms of probabilities.

12.9.1 Using `augment` to capture the fitted probabilities

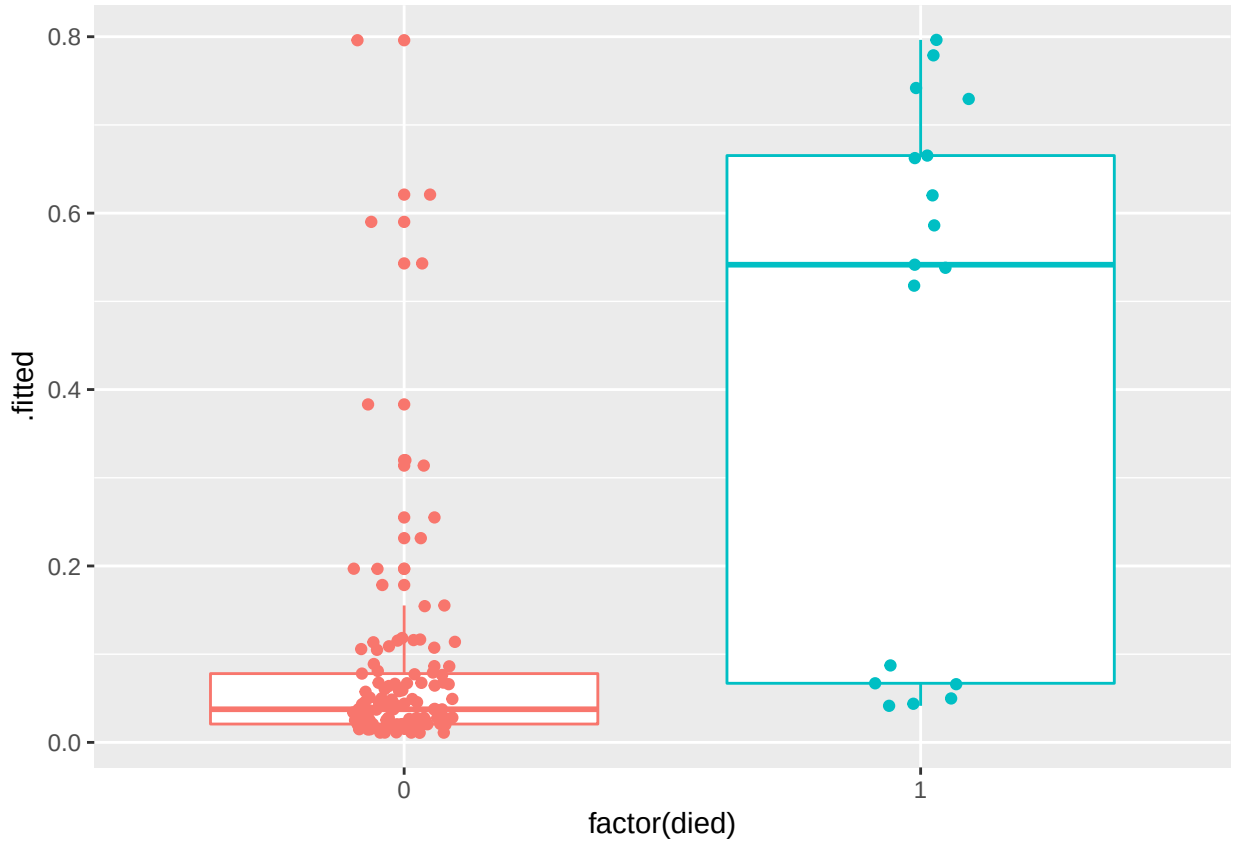
```
res_B_aug <- augment(res_modB, resect,
                     type.predict = "response")
head(res_B_aug)
```

```
  id age prior resection intubated died    .fitted    .se.fit     .resid
1  1  34     1       2.5         0    0 0.05908963 0.03851118 -0.3490198
2  2  57     0       5.0         0    0 0.11660492 0.06253774 -0.4979613
3  3  60     1       4.0         1    1 0.72944600 0.15010423  0.7943172
4  4  62     1       4.2         0    0 0.15522494 0.09607978 -0.5808354
5  5  28     0       6.0         1    1 0.79641141 0.14588554  0.6747435
6  6  52     0       3.0         0    0 0.03713809 0.01933270 -0.2751191
        .hat    .sigma      .cooksd .std.resid
1 0.02667562 0.7247491 0.0003536652 -0.3537702
2 0.03796756 0.7240341 0.0010829917 -0.5076925
3 0.11416656 0.7215778 0.0107925872  0.8439524
4 0.07039819 0.7234665 0.0029937671 -0.6024273
```

```
5 0.13126049 0.7225958 0.0088920280  0.7239256
6 0.01045207 0.7250114 0.0000823406 -0.2765683
```

12.9.2 Plotting Model B Fits by Observed Mortality

```
ggplot(res_B_aug, aes(x = factor(died), y = .fitted, col = factor(died))) +
    geom_boxplot() +
    geom_jitter(width = 0.1) +
    guides(col = FALSE)
```

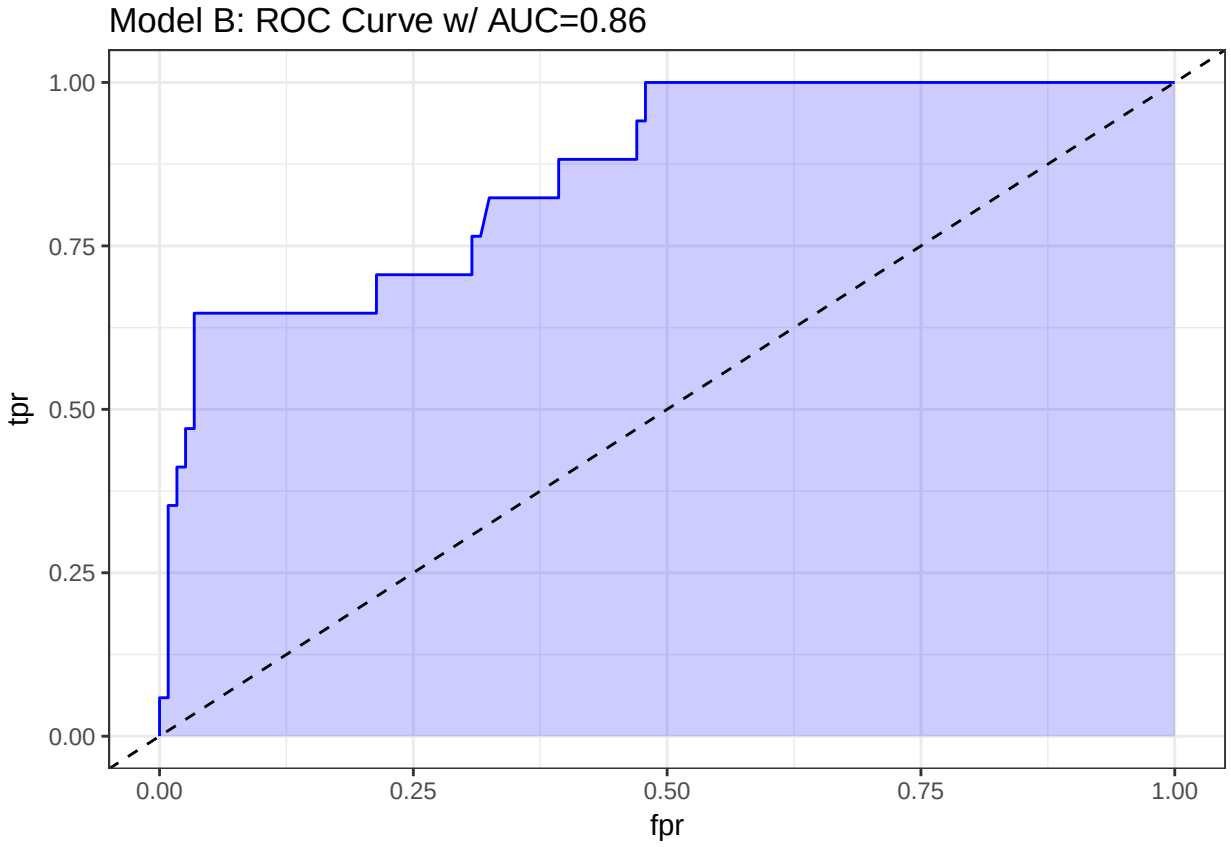


Certainly it appears as though most of our predicted probabilities (of death) for the subjects who actually survived are quite small, but not all of them. We also have at least 6 big "misses" among the 17 subjects who actually died.

12.9.3 The ROC curve for Model B

```
## requires ROCR package
prob <- predict(res_modB, resect, type="response")
pred <- prediction(prob, resect$died)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
auc <- performance(pred, measure="auc")
## the rest of this code is a little strange
auc <- round(auc@y.values[[1]],3)
roc.data <- data.frame(fpr=unlist(perf@x.values),
```

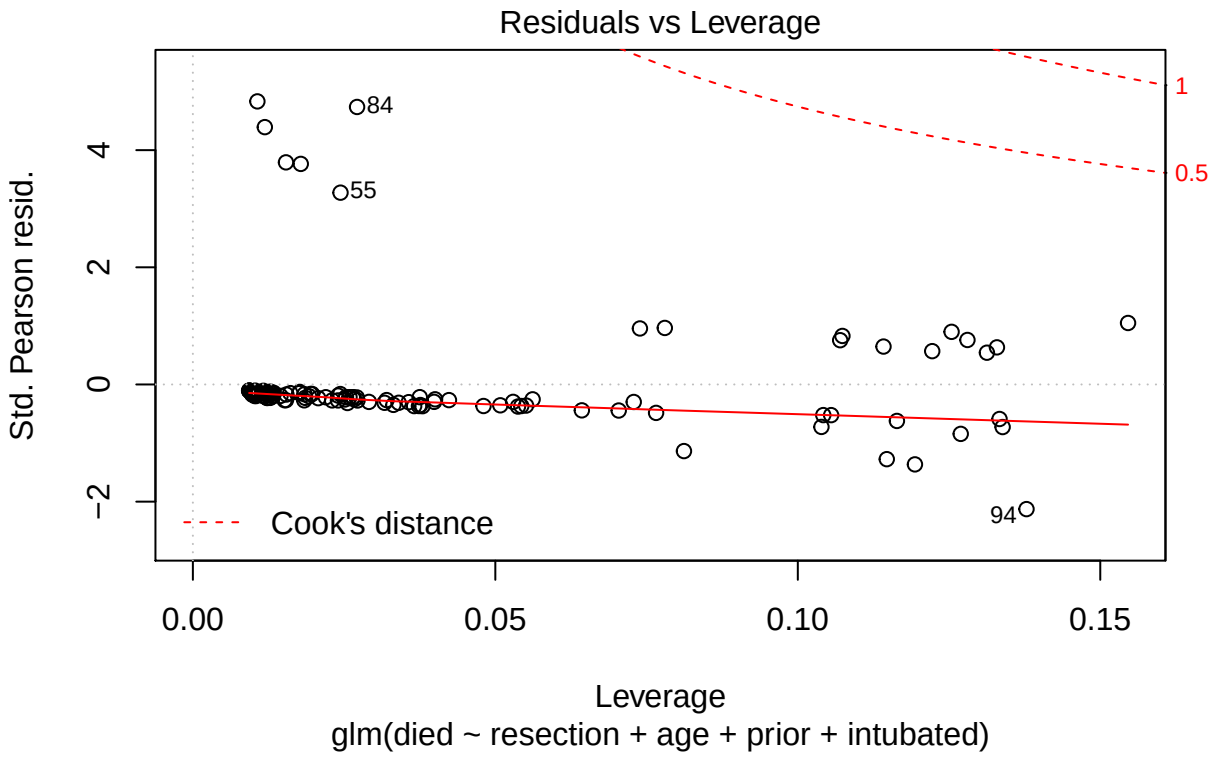
```
                        tpr=unlist(perf@y.values),
                        model="GLM")

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2, fill = "blue") +
    geom_line(aes(y=tpr), col = "blue") +
    geom_abline(intercept = 0, slope = 1, lty = "dashed") +
    labs(title = paste0("Model B: ROC Curve w/ AUC=", auc)) +
    theme_bw()
```



The area under the curve (C-statistic) is 0.86, which certainly looks like a more discriminating fit than model A with resection alone.

12.9.4 Residuals, Leverage and Influence

```
plot(res_modB, which = 5)
```



Again, we see no signs of deeply influential points in this model.

Bibliography

Barnett, P. A., Roman-Golstein, S., Ramsey, F., et al. (1995). Differential permeability and quantitative mr imaging of a human lung carcinoma brain xenograft in the nude rat. *American Journal of Pathology*, 146(2):436–449.

Berkhemer, O. A., Fransen, P. S. S., Buemer, D., et al. (2015). A randomized trial of intraarterial treatment for acute ischemic stroke. *New England Journal of Medicine*, 372:11–20.

Faraway, J. J. (2015). *Linear Models with R*. CRC Press, Boca Raton, FL, second edition.

Gelman, A. and Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, New York.

Harrell, F. E. (2001). *Regression Modeling Strategies*. Springer, New York.

Hastie, T., Tibshriani, R., and Frideman, J. H. (2001). *The Elements of Statistical Learning*. Springer, New York.

Hurvich, C. M. and Tsai, C.-L. (1989). Regression and time series model selection in small samples. *Biometrika*, 76:297–307.

Kim, H.-Y. (2014). Statistical notes for clinical researchers: Two-way analysis of variance (anova) - exploring possible interaction between factors. *Restorative Dentistry & Endodontics*, 39(2):143–147.

Leeb, H. and Potscher, B. M. (2005). Model selection and inference: Facts and fiction. *Econometric Theory*, 21(1):21–59.

Long, J. S. (1997). *Regression Models for categorical and limited dependent variables*. Sage Publications, Thousand Oaks, CA.

McDonald, G. C. and Schwing, R. C. (1973). Instabilities of regression estimates relating air pollution to mortality. *Technometrics*, 15(3):463–481.

Peduzzi, P., Concato, J., Kemper, E., Holford, T. R., and Feinstein, A. R. (1996). A simulation study of the number of events per variable in logistic regression analysis. *Journal of Clinical Epidemiology*, 49(12):1373–1379.

Ramsey, F. L. and Schafer, D. W. (2002). *The Statistical Sleuth: A Course in Methods of Data Analysis*. Duxbury, Pacific Grove, CA, second edition edition.

Riffenburgh, R. H. (2006). *Statistics in Medicine*. Elsevier Academic Press, Burlington, MA, second edition edition.

Rosenbaum, P. R. (2017). *Observation and Experiment: An Introduction to Causal Inference*. Harvard University Press, Cambridge, MA.

Roy, D., Talajic, M., Nattel, S., et al. (2008). Rhythm control versus rate control for atrial fibrillation and heart failure. *New England Journal of Medicine*, 358:2667–2677.

Stamey, T.A., J. K. I. J. et al. (1989). Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate: Ii. radical prostatectomy treated patients. *Journal of Urology*, 141(5):1076–1083.

Tolaney, S. M., Barry, W. T., Chau, T. D., et al. (2015). Adjuvant paclitaxel and trastuzumab for node-negative, her2-positive breast cancer. *New England Journal of Medicine*, 372:134–141.

Vittinghoff, E., Glidden, D. V., Shiboski, S. C., and McCulloch, C. E. (2012). *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. Springer-Verlag, Inc., second edition edition.