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

Thomas E. Love, Ph.D.

Built 2021-01-29 22:34:27

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 via Piazza.

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 appropriate link at the 432 course website.

R Packages used in these notes

Here, we'll load in some packages used in these notes. The list of R Packages we will use in 432 is more extensive, and is available on our course website.

```
library(here)
library(janitor)
library(magrittr)
library(conflicted)

library(tableone)

library(broom)
library(haven)
library(janitor)
library(patchwork)
library(Hmisc)
library(rms)

library(MASS)
library(visdat)
library(naniar)
library(caret)
library(simputation)
library(car)
library(mice)
library(leaps)
library(lars)
library(Epi)
library(pROC)
library(ROCR)
library(VGAM)
library(ggridges)
library(pander)
library(arm)
library(survival)
library(survminer)
library(kableExtra)

## and of course, we conclude with...

library(tidymodels)
library(tidyverse)
```

Dealing with Conflicts

I'm loading a lot of packages here, and sometimes individual functions are in conflict. R's default conflict resolution system gives precedence to the most recently loaded package. This can make it hard to detect conflicts, particularly when introduced by an update to an existing package.

Using the code below helps the entire book run properly. You may or may not need to look into the `conflicted` package for your work.

```
conflict_prefer("filter", "dplyr")
```

```
[conflicted] Will prefer dplyr::filter over any other package
```

```
conflict_prefer("select", "dplyr")
```

```
[conflicted] Will prefer dplyr::select over any other package
```

```
conflict_prefer("Predict", "rms")
```

```
[conflicted] Will prefer rms::Predict over any other package
```

```
conflict_prefer("impute_median", "simputation")
```

```
[conflicted] Will prefer simputation::impute_median over any other package
```

```
conflict_prefer("summarize", "dplyr")
```

```
[conflicted] Will prefer dplyr::summarize over any other package
```

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

General Theme for `ggplot` work

```
theme_set(theme_bw())
```

Data used in these notes

All data sets used in these notes are available on our Data and Code website.

Dr. Love is in the process of moving all of the data loads below to their individual chapters.

```
prost <- read_csv("data/prost.csv")
pollution <- read_csv("data/pollution.csv")

bonding <- read_csv("data/bonding.csv")
cortisol <- read_csv("data/cortisol.csv")
emphysema <- read_csv("data/emphysema.csv")
resect <- read_csv("data/resect.csv")
colscr <- read_csv("data/screening.csv")
colscr2 <- read_csv("data/screening2.csv")
authorship <- read_csv("data/authorship.csv")
hem <- read_csv("data/hem.csv")
leukem <- read_csv("data/leukem.csv")
```

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 Data load

Let's load two data sets for this Chapter. All data sets used in these notes are available on our Data and Code website.

```
fakestroke <- read_csv("data/fakestroke.csv")
```

```
-- Column specification --------------------------------------------------------
cols(
  studyid = col_character(),
  trt = col_character(),
  age = col_double(),
  sex = col_character(),
  nihss = col_double(),
  location = col_character(),
  hx.isch = col_character(),
  afib = col_double(),
  dm = col_double(),
  mrankin = col_character(),
  sbp = col_double(),
  iv.altep = col_character(),
  time.iv = col_double(),
  aspects = col_double(),
  ia.occlus = col_character(),
  extra.ica = col_double(),
  time.rand = col_double(),
  time.punc = col_double()
)
```

```r
bloodbrain <- read_csv("data/bloodbrain.csv")
```

```
-- Column specification --------------------------------------------------------
cols(
  case = col_double(),
  brain = col_double(),
  liver = col_double(),
  tlratio = col_double(),
  solution = col_character(),
  sactime = col_double(),
  postin = col_double(),
  sex = col_character(),
  wt.init = col_double(),
  wt.loss = col_double(),
  wt.tumor = col_double()
)
```

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

1.2.1 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).

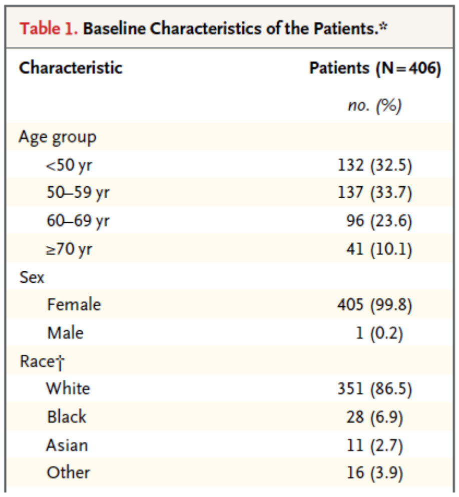


Table 1. Baseline Characteristics of the Patients.*

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.

1.2.2 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].

[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.

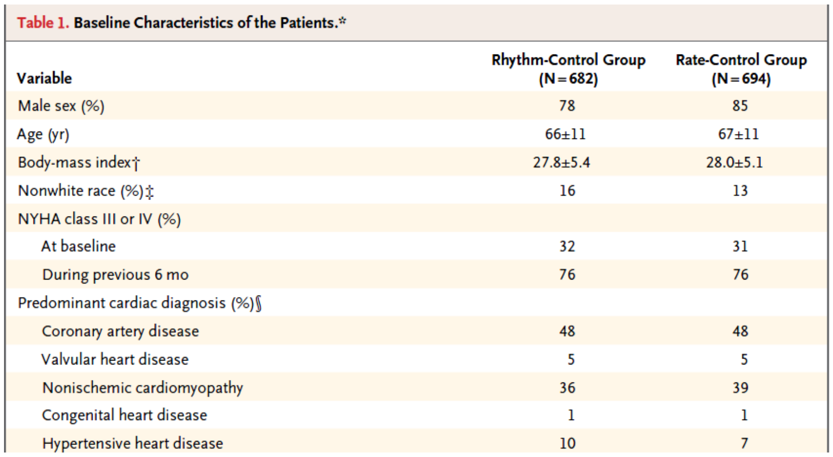


Table 1. Baseline Characteristics of the Patients.*

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 randomization stratified by study center.

1.3 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 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).

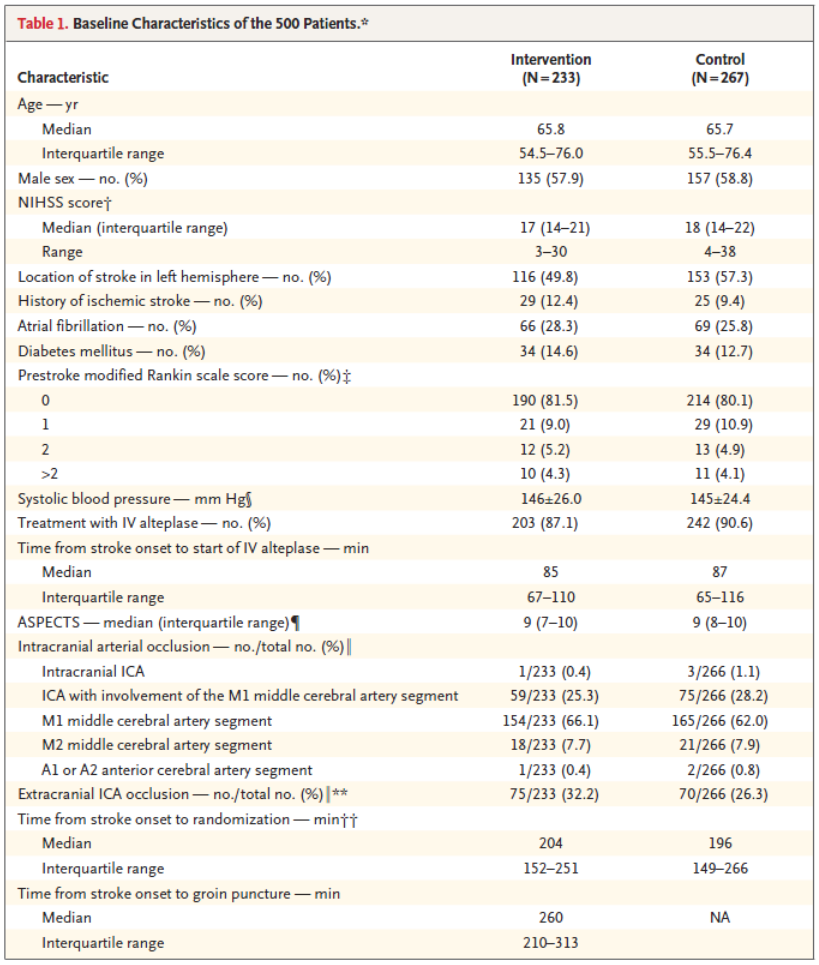
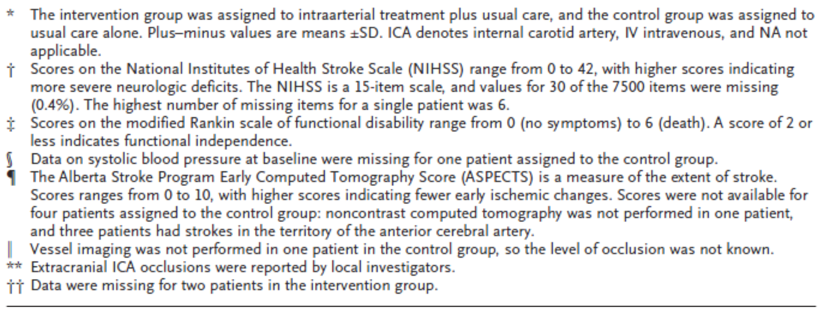


Table 1. Baseline Characteristics of the 500 Patients.*

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.

1.4 Simulated `fakestroke` data

Consider the simulated data, available on our Data and Code 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`.

```
fakestroke
```

```
# A tibble: 500 x 18
   studyid trt     age sex   nihss location hx.isch  afib    dm mrankin   sbp
   <chr>   <chr> <dbl> <chr> <dbl> <chr>    <chr>   <dbl> <dbl> <chr>   <dbl>
```

[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

```
 1 z001     Cont~    53 Male     21 Right    No           0      0 2           127
 2 z002     Inte~    51 Male     23 Left     No           1      0 0           137
 3 z003     Cont~    68 Fema~    11 Right    No           0      0 0           138
 4 z004     Cont~    28 Male     22 Left     No           0      0 0           122
 5 z005     Cont~    91 Male     24 Right    No           0      0 0           162
 6 z006     Cont~    34 Fema~    18 Left     No           0      0 2           166
 7 z007     Inte~    75 Male     25 Right    No           0      0 0           140
 8 z008     Cont~    89 Fema~    18 Right    No           0      0 0           157
 9 z009     Cont~    75 Male     25 Left     No           1      0 2           129
10 z010     Inte~    26 Fema~    27 Right    No           0      0 0           143
# ... with 490 more rows, and 7 more variables: iv.altep <chr>, time.iv <dbl>,
#   aspects <dbl>, ia.occlus <chr>, extra.ica <dbl>, time.rand <dbl>,
#   time.punc <dbl>
```

1.5 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
```

1.5.1 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.

1.5.2 `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")
           )
```

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

1.6 fakestroke Table 1: Attempt 2

```r
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.

1.6.1 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.

```r
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
```

1.7 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.

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

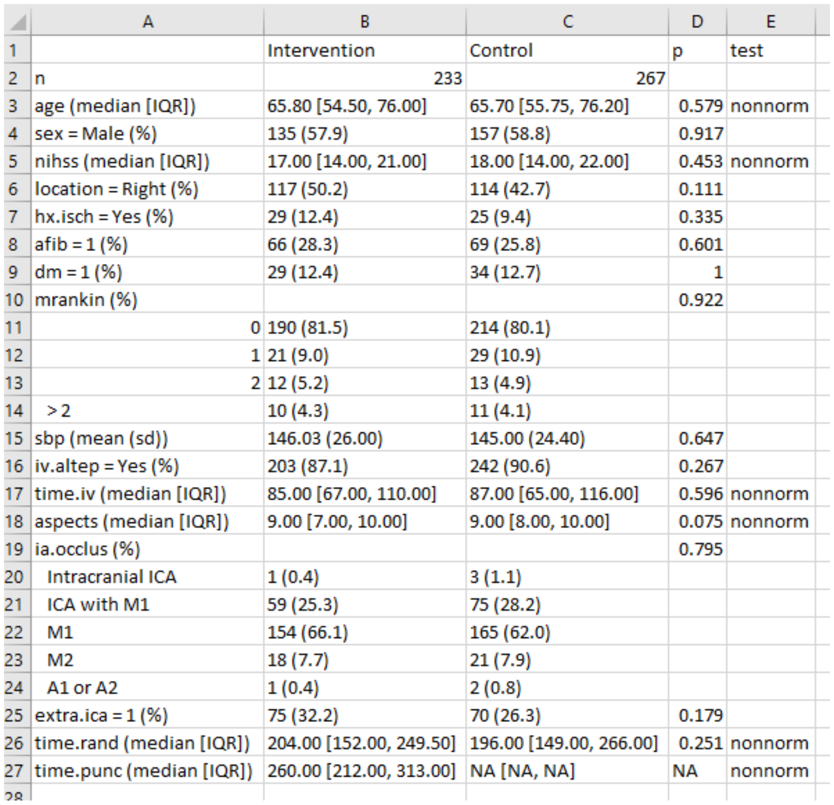
Once you've built the table and are generally satisfied with it, you'll probably want to be able to drop it into Excel or Word for final cleanup.

1.8.1 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.

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

```r
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.

1.9 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?

1.10 The `bloodbrain.csv` file

Consider the data, available on our Data and Code 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

Variable	Description
`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 wt.loss
   <dbl>  <dbl>  <dbl>   <dbl> <chr>      <dbl>  <dbl> <chr>   <dbl>   <dbl>
 1     1  41081 1.46e6  0.0282 BD           0.5     10 F         239     5.9
 2     2  44286 1.60e6  0.0276 BD           0.5     10 F         225     4
 3     3 102926 1.60e6  0.0642 BD           0.5     10 F         224    -4.9
 4     4  25927 1.78e6  0.0146 BD           0.5     10 F         184     9.8
 5     5  42643 1.35e6  0.0316 BD           0.5     10 F         250     6
 6     6  31342 1.79e6  0.0175 NS           0.5     10 F         196     7.7
 7     7  22815 1.63e6  0.0140 NS           0.5     10 F         200     0.5
 8     8  16629 1.62e6  0.0103 NS           0.5     10 F         273     4
 9     9  22315 1.57e6  0.0142 NS           0.5     10 F         216     2.8
10    10  77961 1.06e6  0.0735 BD           3       10 F         267     2.6
# ... with 24 more rows, and 1 more variable: wt.tumor <dbl>
```

1.11 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  skew
wt.init  17    0      0    243 3e+01  2e+02  2e+02 3e+02  2e+02 3e+02 -0.39
wt.loss  17    0      0      3 5e+00  4e+00  1e+00 6e+00 -5e+00 1e+01 -0.10
wt.tumor 17    0      0    157 8e+01  2e+02  1e+02 2e+02  2e+01 4e+02  0.53
brain    17    0      0  56043 3e+04  5e+04  4e+04 8e+04  6e+03 1e+05  0.29
liver    17    0      0 672577 7e+05  6e+05  2e+04 1e+06  2e+03 2e+06  0.35
tlratio  17    0      0      2 3e+00  1e-01  6e-02 3e+00  1e-02 9e+00  1.58
logTL    17    0      0     -1 2e+00 -2e+00 -3e+00 1e+00 -4e+00 2e+00  0.08
         kurt
wt.init   0.7
wt.loss   0.2
wt.tumor  1.0
brain    -0.6
liver    -1.7
tlratio   1.7
logTL    -1.7
------------------------------------------------------------
solution: NS
          n miss p.miss   mean    sd median    p25    p75    min   max  skew
wt.init  17    0      0    240 3e+01  2e+02  2e+02  3e+02  2e+02 3e+02  0.33
wt.loss  17    0      0      4 4e+00  3e+00  2e+00  7e+00 -4e+00 1e+01 -0.09
wt.tumor 17    0      0    209 1e+02  2e+02  2e+02  3e+02  3e+01 5e+02  0.63
brain    17    0      0  23887 1e+04  2e+04  1e+04  3e+04  1e+03 5e+04  0.30
liver    17    0      0 664975 7e+05  7e+05  2e+04  1e+06  9e+02 2e+06  0.40
tlratio  17    0      0      1 2e+00  5e-02  3e-02  9e-01  1e-02 7e+00  2.27
logTL    17    0      0     -2 2e+00 -3e+00 -3e+00 -7e-02 -5e+00 2e+00  0.27
          kurt
wt.init  -0.48
wt.loss   0.08
wt.tumor  0.77
brain    -0.35
liver    -1.56
tlratio   4.84
logTL    -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.

1.11.1 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.

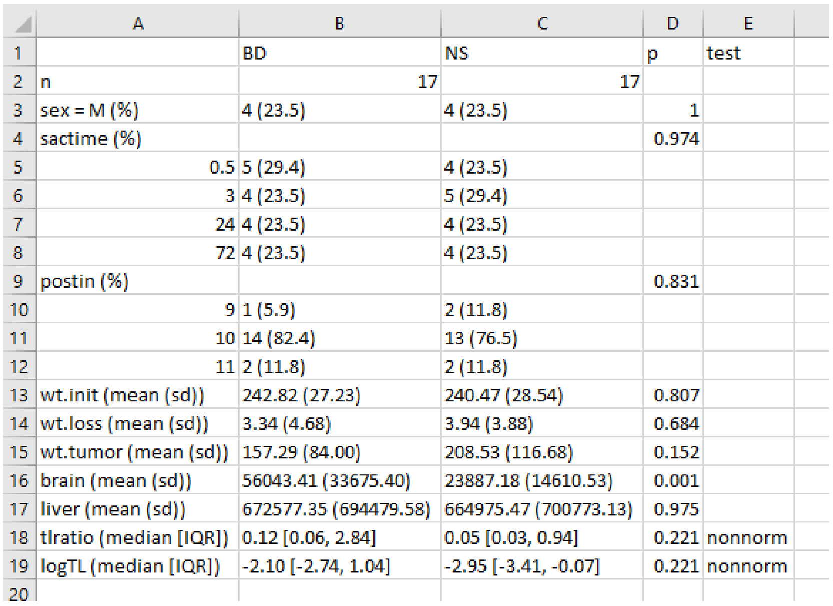
```
print(bb.att1, nonnormal = c("tlratio", "logTL"))
```

```
                        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

```r
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.

1.11.2 A More Finished Version (after Cleanup in Word)

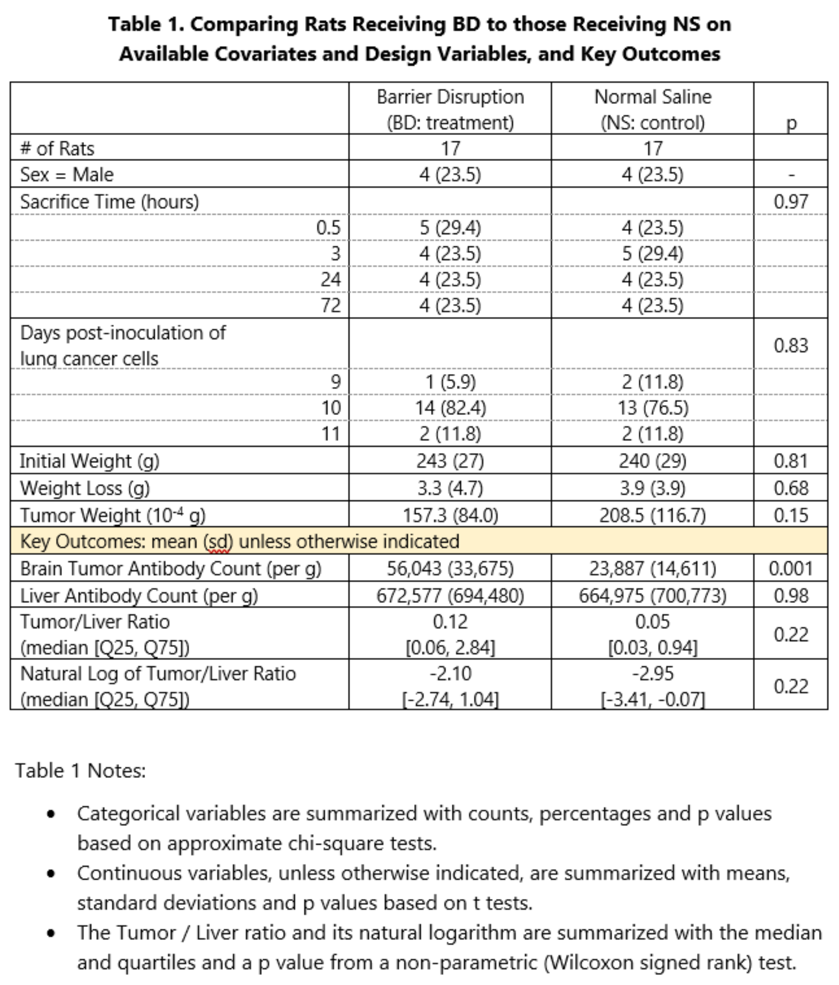


Table 1. Comparing Rats Receiving BD to those Receiving NS on Available Covariates and Design Variables, and Key Outcomes

	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

BRFSS SMART Data Building

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 2017 SMART, and in particular on data from the state of Ohio, and 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.

In this chapter, I describe some cleaning of the BRFSS SMART data, and break it out into national, statewide, and local samples.

The data files produced by this chapter include:

- `smart_ohio.Rds` which includes data on approximately 100 variables for over 7000 subjects in six MMSAs that are at least partially located in the state of Ohio.
- `smart_cle.Rds` which includes data on those same variables for a little over 1000 subjects in the Cleveland-Elyria-Lorain OH MMSA.

2.1 Key resources

- the "raw" data, in the form of the 2017 SMART BRFSS MMSA Data, found in a zipped SAS Transport Format file. The data were released in

October 2018.
- the MMSA Variable Layout 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.
- the lengthy 2017 Survey Questions PDF which lists all questions asked as part of the BRFSS in 2017
- the enormous Codebook for the 2017 BRFSS Survey PDF which identifies the variables by name for us.

Also, for each subject, we are also provided with a sampling weight, in `_MMSAWT`, which will help us incorporate the sampling design later. These weights are at the MMSA level, and are used for generating MMSA-level estimates for variables in the data set. Details on the weighting methodology are available at this PDF.

2.2 Ingesting the Raw Data

To create the data files we'll use, I used the `read_xpt` function from the `haven` package to bring in the SAS XPT data file that is provided by CDC. The codes I used (but won't use in these Notes) were:

```
smart_raw <- read_xpt("MMSA2017/MMSA2017.xpt")
```

This gives the nationwide data, which has 230,875 rows and 177 columns.

But for the purposes of putting these Notes online, I needed to crank down the sample size enormously. To that end, I created a new data file, which I developed by

- importing the MMSA2017.xpt file as above
- filtering away all observations except those from MMSAs which include Ohio in their name, and
- saving the result, which now has 7,412 rows and 177 columns.

The code (again, not run here) that I used to filter to the OH-based MMSAs was:

```
smart_ohio_raw <- smart_raw %>%
    filter(str_detect(MMSANAME, "OH"))

write_csv(smart_ohio_raw, "data/smart_ohio_raw.csv")
```

So, for purposes of these notes, our complete data set is actually coming from `smart_ohio_raw.csv` and consists only of the 7,412 observations associated with the six MMSAs that include Ohio in their names.

2.3 Ingesting from our CSV file

Note that the `smart_ohio_raw.csv` and other data files we're developing in this Chapter are available on our Data and Code website

```
smart_ohio_raw <- read_csv("data/smart_ohio_raw.csv")

dim(smart_ohio_raw)
```

```
[1] 7412  177
```

2.4 What does the raw data look like?

Here is a list of all variable names included in this file. We're not going to use all of those variables, but this will give you a sense of what is available.

```
names(smart_ohio_raw)
```

```
  [1] "DISPCODE" "STATERE1" "SAFETIME" "HHADULT"  "GENHLTH"  "PHYSHLTH"
  [7] "MENTHLTH" "POORHLTH" "HLTHPLN1" "PERSDOC2" "MEDCOST"  "CHECKUP1"
 [13] "BPHIGH4"  "BPMEDS"   "CHOLCHK1" "TOLDHI2"  "CHOLMED1" "CVDINFR4"
 [19] "CVDCRHD4" "CVDSTRK3" "ASTHMA3"  "ASTHNOW"  "CHCSCNCR" "CHCOCNCR"
 [25] "CHCCOPD1" "HAVARTH3" "ADDEPEV2" "CHCKIDNY" "DIABETE3" "DIABAGE2"
 [31] "LMTJOIN3" "ARTHDIS2" "ARTHSOCL" "JOINPAI1" "SEX"      "MARITAL"
 [37] "EDUCA"    "RENTHOM1" "NUMHHOL2" "NUMPHON2" "CPDEMO1A" "VETERAN3"
 [43] "EMPLOY1"  "CHILDREN" "INCOME2"  "INTERNET" "WEIGHT2"  "HEIGHT3"
 [49] "PREGNANT" "DEAF"     "BLIND"    "DECIDE"   "DIFFWALK" "DIFFDRES"
 [55] "DIFFALON" "SMOKE100" "SMOKDAY2" "STOPSMK2" "LASTSMK2" "USENOW3"
 [61] "ECIGARET" "ECIGNOW"  "ALCDAY5"  "AVEDRNK2" "DRNK3GE5" "MAXDRNKS"
 [67] "FRUIT2"   "FRUITJU2" "FVGREEN1" "FRENCHF1" "POTATOE1" "VEGETAB2"
 [73] "EXERANY2" "EXRACT11" "EXEROFT1" "EXERHMM1" "EXRACT21" "EXEROFT2"
 [79] "EXERHMM2" "STRENGTH" "SEATBELT" "FLUSHOT6" "FLSHTMY2" "PNEUVAC3"
 [85] "SHINGLE2" "HIVTST6"  "HIVTSTD3" "HIVRISK5" "CASTHDX2" "CASTHNO2"
 [91] "CALLBCKZ" "WDUSENOW" "WDINFTRK" "WDHOWOFT" "WDSHARE"  "NAMTRIBE"
 [97] "NAMOTHR"  "_URBNRRL" "_STSTR"   "_IMPSEX"  "_RFHLTH"  "_PHYS14D"
[103] "_MENT14D" "_HCVU651" "_RFHYPE5" "_CHOLCH1" "_RFCHOL1" "_MICHD"
[109] "_LTASTH1" "_CASTHM1" "_ASTHMS1" "_DRDXAR1" "_LMTACT1" "_LMTWRK1"
[115] "_LMTSCL1" "_PRACE1"  "_MRACE1"  "_HISPANC" "_RACE"    "_RACEG21"
[121] "_RACEGR3" "_AGEG5YR" "_AGE65YR" "_AGE80"   "_AGE_G"   "WTKG3"
[127] "_BMI5"    "_BMI5CAT" "_RFBMI5"  "_EDUCAG"  "_INCOMG"  "_SMOKER3"
[133] "_RFSMOK3" "_ECIGSTS" "_CURECIG" "DRNKANY5" "_RFBING5" "_DRNKWEK"
[139] "_RFDRHV5" "FTJUDA2_" "FRUTDA2_" "GRENDA1_" "FRNCHDA_" "POTADA1_"
[145] "VEGEDA2_" "_MISFRT1" "_MISVEG1" "_FRTRES1" "_VEGRES1" "_FRUTSU1"
[151] "_VEGESU1" "_FRTLT1A" "_VEGLT1A" "_FRT16A"  "_VEG23A"  "_FRUITE1"
[157] "_VEGETE1" "_TOTINDA" "_MINAC11" "_MINAC21" "_PACAT1"  "_PAINDX1"
```

```
[163] "_PA150R2" "_PA300R2" "_PA30021" "_PASTRNG" "_PAREC1"  "_PASTAE1"
[169] "_RFSEAT2" "_RFSEAT3" "_FLSHOT6" "_PNEUMO2" "_AIDTST3" "_MMSA"
[175] "_MMSAWT"  "SEQNO"    "MMSANAME"
```

2.5 Cleaning the BRFSS Data

2.5.1 Identifying Information

The identifying variables for each subject are gathered in `SEQNO`, which I'll leave alone.

- Each statistical (geographic) area is identified by a `_MMSA` variable, which I'll rename `mmsa_code`, and by an `MMSANAME` which I'll rename as `mmsa_name`
- For each subject, we are also provided with a sampling weight, in `_MMSAWT`, which will help us incorporate the sampling design later in the semester. We'll rename this as `mmsa_wt`. Details on the weighting methodology are available at https://www.cdc.gov/brfss/annual_data/2017/pdf/2017_SMART_BRFSS_MMSA_Methodology-508.pdf

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(mmsa_code = `_MMSA`,
           mmsa_name = `MMSANAME`,
           mmsa_wt = `_MMSAWT`)

smart_ohio_raw %>% count(mmsa_code, mmsa_name)
```

```
# A tibble: 6 x 3
  mmsa_code mmsa_name                                                      n
      <dbl> <chr>                                                      <int>
1     17140 Cincinnati, OH-KY-IN, Metropolitan Statistical Area         1737
2     17460 Cleveland-Elyria, OH, Metropolitan Statistical Area         1133
3     18140 Columbus, OH, Metropolitan Statistical Area                 2033
4     19380 Dayton, OH, Metropolitan Statistical Area                    587
5     26580 Huntington-Ashland, WV-KY-OH, Metropolitan Statistical Area  1156
6     45780 Toledo, OH, Metropolitan Statistical Area                    766
```

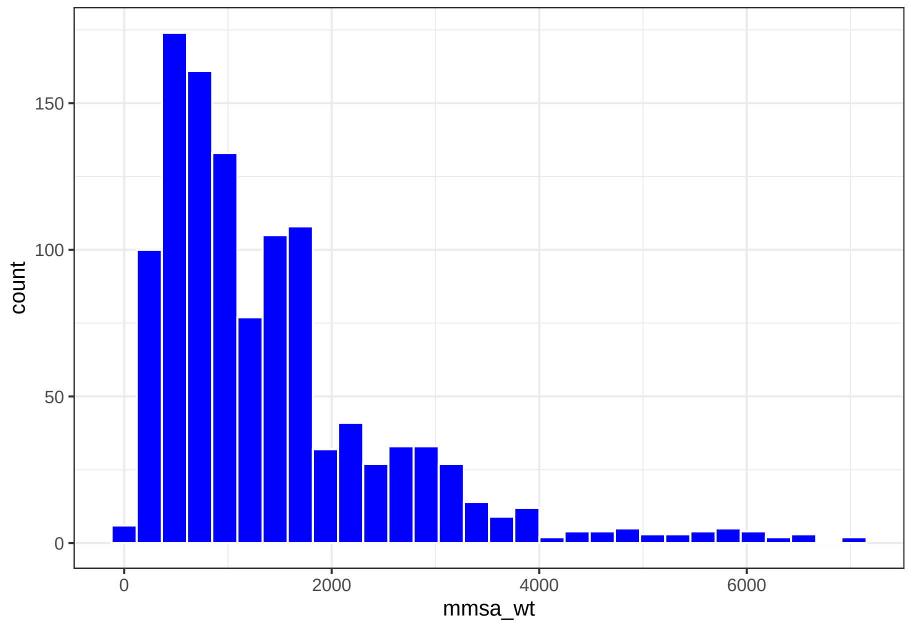
Those names are very long. I'll build some shorter ones, by dropping everything after the comma.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(mmsa = str_replace_all(string = mmsa_name, pattern="\\,.*$",replacement=" "

smart_ohio_raw %>% count(mmsa, mmsa_name)
```

```
# A tibble: 6 x 3
  mmsa                mmsa_name                                                      n
  <chr>                <chr>                                                   <int>
1 "Cincinnati "        Cincinnati, OH-KY-IN, Metropolitan Statistical Area      1737
2 "Cleveland-Elyria "  Cleveland-Elyria, OH, Metropolitan Statistical Area      1133
3 "Columbus "          Columbus, OH, Metropolitan Statistical Area              2033
4 "Dayton "            Dayton, OH, Metropolitan Statistical Area                 587
5 "Huntington-Ashlan~  Huntington-Ashland, WV-KY-OH, Metropolitan Statisti~     1156
6 "Toledo "            Toledo, OH, Metropolitan Statistical Area                 766
```

And here are the sampling weights for the subjects in the Cleveland-Elyria MSA.

```
smart_ohio_raw %>%
    filter(mmsa_code == 17460) %>%
    ggplot(., aes(x = mmsa_wt)) +
    geom_histogram(bins = 30, fill = "blue", col = "white")
```



2.5.2 Survey Method

2.5.2.1 DISPCODE and its cleanup to completed

DISPCODE which is 1100 if the subject completed the interview, and 1200 if they partially completed the interview. We'll create a variable called `completed` that indicates (1 = complete, 0 = not) whether the subject completed the interview.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(completed = 12 - (DISPCODE/100))
```

```
smart_ohio_raw %>% count(DISPCODE, completed)
```

```
# A tibble: 2 x 3
  DISPCODE completed     n
     <dbl>     <dbl> <int>
1     1100         1  6277
2     1200         0  1135
```

2.5.2.2 `STATERE1` and `SAFETIME` and their reduction to `landline`

BRFSSS is conducted by telephone. The next two variables help us understand whether the subject was contacted via land line or via cellular phone.

- `STATERE1` is 1 if the subject is a resident of the state (only asked of people in the land line version of the survey).
- `SAFETIME` is 1 if this is a safe time to talk (only asked of people in the cell phone version of the survey).
- We'll use `STATERE1` and `SAFETIME` to create an indicator variable `landline` that specifies how the respondent was surveyed (1 = land line, 0 = cell phone), as follows…

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(landline = replace_na(STATERE1, 0))

smart_ohio_raw %>% count(STATERE1, SAFETIME, landline)
```

```
# A tibble: 2 x 4
  STATERE1 SAFETIME landline     n
     <dbl>    <dbl>    <dbl> <int>
1        1       NA        1  3649
2       NA        1        0  3763
```

2.5.2.3 `HHADULT` and its cleanup to `hhadults`

- `HHADULT` is the response to "How many members of your household, including yourself, are 18 years of age or older?"
  - The permitted responses range from 1-76, with special values 77 for Don't Know/Not Sure and 99 for refused, with BLANK for missing or not asked.
  - So we should change all numerical values above 76 to NA for our analyses (the blanks are already regarded as NAs by R in the ingestion process.)

```
smart_ohio_raw %>% tabyl(HHADULT)
```

```
 HHADULT    n      percent valid_percent
       1  274 0.0369670804   0.236206897
       2  603 0.0813545602   0.519827586
       3  170 0.0229357798   0.146551724
       4   73 0.0098488937   0.062931034
       5   28 0.0037776579   0.024137931
       6    4 0.0005396654   0.003448276
       7    3 0.0004047491   0.002586207
       8    1 0.0001349164   0.000862069
      10    1 0.0001349164   0.000862069
      11    1 0.0001349164   0.000862069
      99    2 0.0002698327   0.001724138
      NA 6252 0.8434970318            NA
```

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(hhadults = HHADULT,
           hhadults = replace(hhadults, hhadults > 76, NA))

smart_ohio_raw %>% count(HHADULT, hhadults) %>% tail()
```

```
# A tibble: 6 x 3
  HHADULT hhadults     n
    <dbl>    <dbl> <int>
1       7        7     3
2       8        8     1
3      10       10     1
4      11       11     1
5      99       NA     2
6      NA       NA  6252
```

2.5.3 Health Status (1 item)

The next variable describes relate to the subject's health status.

2.5.3.1 `GENHLTH` and its cleanup to `genhealth`

- `GENHLTH`, the General Health variable, which is the response to "Would you say that in general your health is . . . "
  - 1 = Excellent
  - 2 = Very good
  - 3 = Good
  - 4 = Fair
  - 5 = Poor
  - 7 = Don't know/Not sure
  - 9 = Refused

- BLANK = Not asked or missing

To clean up the `GENHLTH` data into a new variable called `genhealth` we'll need to - convince R that the 7 and 9 values are in fact best interpreted as `NA`, - and perhaps change the variable to a factor and incorporate the names into the levels.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(genhealth = fct_recode(factor(GENHLTH),
                                "1_Excellent" = "1",
                                "2_VeryGood" = "2",
                                "3_Good" = "3",
                                "4_Fair" = "4",
                                "5_Poor" = "5",
                                NULL = "7",
                                NULL = "9"))

smart_ohio_raw %>% count(GENHLTH, genhealth)
```

```
# A tibble: 7 x 3
  GENHLTH genhealth       n
    <dbl> <fct>       <int>
1       1 1_Excellent  1057
2       2 2_VeryGood   2406
3       3 3_Good       2367
4       4 4_Fair       1139
5       5 5_Poor        428
6       7 <NA>           10
7       9 <NA>            5
```

2.5.4 Healthy Days - Health-Related Quality of Life (3 items)

The next three variables describe the subject's health-related quality of life.

2.5.4.1 `PHYSHLTH` and its cleanup to `physhealth`

PHYSHLTH', the Number of Days Physical Health Not Good variable, which is the response to "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?"

- Values of 1-30 are numeric and reasonable.
- A value of 88 indicates "none" and should be recoded to 0.
- 77 is the code for Don't know/Not sure
- 99 is the code for Refused

- BLANK indicates Not asked or missing, and R recognizes this as `NA` properly.

To clean up `PHYSHLTH` to a new variable called `physhealth`, we'll need: - to convince R that the 77 and 99 values are in fact best interpreted as `NA`, and - to convince R that the 88 should be interpreted as 0.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(physhealth = PHYSHLTH,
           physhealth = replace(physhealth, physhealth %in% c(77, 99), NA),
           physhealth = replace(physhealth, physhealth == 88, 0))

smart_ohio_raw %>% count(PHYSHLTH, physhealth) %>% tail()
```

```
# A tibble: 6 x 3
  PHYSHLTH physhealth     n
     <dbl>      <dbl> <int>
1       28         28    12
2       29         29    14
3       30         30   677
4       77         NA   123
5       88          0  4380
6       99         NA    15
```

Note that we present the `tail` of the counts in this case so we can see what happens to the key values (77, 88, 99) of our original variable `PHYSHLTH`.

2.5.4.2 `MENTHLTH` and its cleanup to `menthealth`

MENTHLTH', the Number of Days Mental Health Not Good variable, which is the response to "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?"

- This is coded just like the `PHYSHLTH` variable, so we need to do the same cleaning we did there.

To clean up `MENTHLTH` to a new variable called `menthealth`, we'll need: - to convince R that the 77 and 99 values are in fact best interpreted as `NA`, and - to convince R that the 88 should be interpreted as 0.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(menthealth = MENTHLTH,
           menthealth = replace(menthealth, menthealth %in% c(77, 99), NA),
           menthealth = replace(menthealth, menthealth == 88, 0))

smart_ohio_raw %>% count(MENTHLTH, menthealth) %>% tail()
```

```
# A tibble: 6 x 3
  MENTHLTH menthealth     n
     <dbl>      <dbl> <int>
1       28         28     7
2       29         29    10
3       30         30   475
4       77         NA    86
5       88          0  4823
6       99         NA    28
```

2.5.4.3 `POORHLTH` and its cleanup to `poorhealth`

`POORHLTH`, the Poor Physical or Mental Health variable, which is the response to "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?"

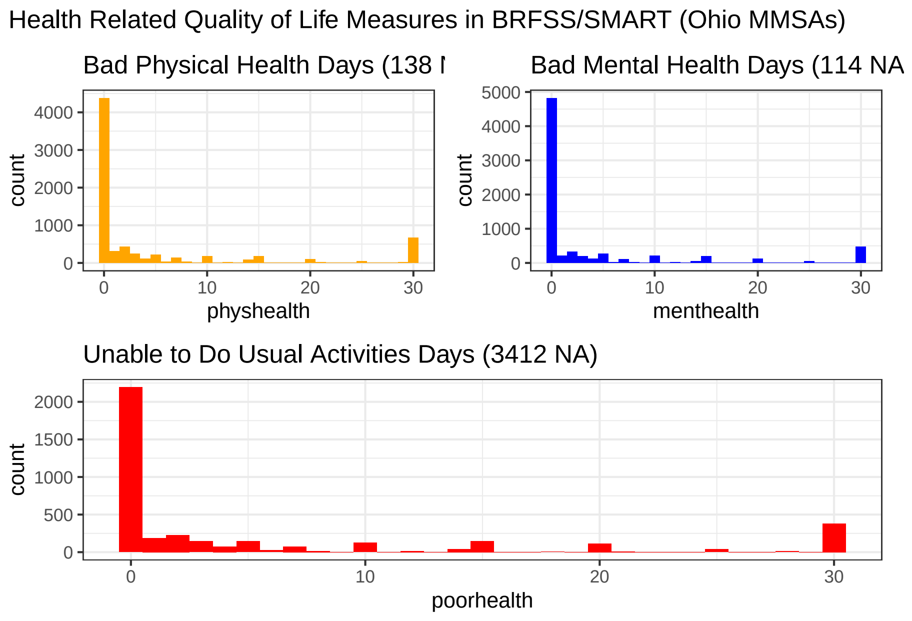
- Again, we recode just like the `PHYSHLTH` variable.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(poorhealth = POORHLTH,
           poorhealth = replace(poorhealth, poorhealth %in% c(77, 99), NA),
           poorhealth = replace(poorhealth, poorhealth == 88, 0))

smart_ohio_raw %>% count(POORHLTH, poorhealth) %>% tail()
```

```
# A tibble: 6 x 3
  POORHLTH poorhealth     n
     <dbl>      <dbl> <int>
1       29         29     4
2       30         30   382
3       77         NA    64
4       88          0  2194
5       99         NA    11
6       NA         NA  3337
```

There's a lot more missingness in the `poorhealth` counts than in the other health-related quality of life measures. There's also a strong mode at 0, and a smaller mode at 30 in each variable.

```
p1 <- ggplot(smart_ohio_raw, aes(x = physhealth)) +
    geom_histogram(binwidth = 1, fill = "orange") +
    labs(title = paste0("Bad Physical Health Days (",
                        sum(is.na(smart_ohio_raw$physhealth)),
                        " NA)"))

p2 <- ggplot(smart_ohio_raw, aes(x = menthealth)) +
    geom_histogram(binwidth = 1, fill = "blue") +
    labs(title = paste0("Bad Mental Health Days (",
                        sum(is.na(smart_ohio_raw$menthealth)),
                        " NA)"))

p3 <- ggplot(smart_ohio_raw, aes(x = poorhealth)) +
    geom_histogram(binwidth = 1, fill = "red") +
    labs(title = paste0("Unable to Do Usual Activities Days (",
                        sum(is.na(smart_ohio_raw$poorhealth)),
                        " NA)"))

(p1 + p2) / p3 +
    plot_annotation(title = "Health Related Quality of Life Measures in BRFSS/SMART (Ohio MMSAs)'
```



2.5.5 Health Care Access (4 items)

The next four variables relate to the subject's health care access.

2.5.5.1 `HLTHPLN1` and its cleanup to `healthplan`

`HLTHPLN1`, the Have any health care coverage variable, is the response to "Do you have any kind of health care coverage, including health insurance, prepaid plans such as HMOs, or government plans such as Medicare, or Indian Health Service?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused

To clean up the `HLTHPLN1` data into a new variable called `healthplan` we'll
- convince R that the 7 and 9 values are in fact best interpreted as `NA`, - and turn it into an indicator variable, e.g., we will leave the variable as numeric, but change the values to 1 = Yes and 0 = No.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(healthplan = HLTHPLN1,
           healthplan = replace(healthplan, healthplan %in% c(7, 9), NA),
           healthplan = replace(healthplan, healthplan == 2, 0))

smart_ohio_raw %>% count(HLTHPLN1, healthplan)
```

```
# A tibble: 4 x 3
  HLTHPLN1 healthplan     n
     <dbl>      <dbl> <int>
1        1          1  6994
2        2          0   398
3        7         NA    10
4        9         NA    10
```

2.5.5.2 `PERSDOC2` and its cleanup to `hasdoc` and to `numdocs2`

`PERSDOC2`, the Multiple Health Care Professionals variable, is the response to "Do you have one person you think of as your personal doctor or health care provider?" where if the response is "No," the survey then asks "Is there more than one or is there no person who you think of as your personal doctor or health care provider?"

- 1 = Yes, only one
- 2 = More than one
- 3 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

To clean up the `PERSDOC2` data into a new variable called `hasdoc` we'll
- convince R that the 7 and 9 values are in fact best interpreted as `NA`, - and turn it into an indicator variable, e.g., we will leave the variable as numeric, but change the values to 1 = Yes and 0 = No, so that the original 1 and 2 become 1, and the original 3 becomes 0.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(hasdoc = PERSDOC2,
           hasdoc = replace(hasdoc, hasdoc %in% c(7, 9), NA),
           hasdoc = replace(hasdoc, hasdoc %in% c(1, 2), 1),
           hasdoc = replace(hasdoc, hasdoc == 3, 0))

smart_ohio_raw %>% count(PERSDOC2, hasdoc)
```

```
# A tibble: 5 x 3
  PERSDOC2 hasdoc     n
     <dbl>  <dbl> <int>
1        1      1  5784
2        2      1   623
3        3      0   990
4        7     NA    14
5        9     NA     1
```

2.5.5.3 `MEDCOST` and its cleanup to `costprob`

`MEDCOST`, the Could Not See Doctor Because of Cost variable, is the response to "Was there a time in the past 12 months when you needed to see a doctor but could not because of cost?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

This is just like `HLTHPLAN`.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(costprob = MEDCOST,
           costprob = replace(costprob, costprob %in% c(7, 9), NA),
           costprob = replace(costprob, costprob == 2, 0))

smart_ohio_raw %>% count(MEDCOST, costprob)
```

```
# A tibble: 4 x 3
  MEDCOST costprob     n
    <dbl>    <dbl> <int>
1       1        1   714
2       2        0  6680
3       7       NA    14
4       9       NA     4
```

2.5.5.4 `CHECKUP1` and its cleanup to `t_checkup`

`CHECKUP1`, the Length of time since last routine checkup variable, is the response to "About how long has it been since you last visited a doctor for a routine checkup? [A routine checkup is a general physical exam, not an exam for a specific injury, illness, or condition.]"

- 1 = Within past year (anytime less than 12 months ago)
- 2 = Within past 2 years (1 year but less than 2 years ago)
- 3 = Within past 5 years (2 years but less than 5 years ago)
- 4 = 5 or more years ago
- 7 = Don't know/Not sure
- 8 = Never
- 9 = Refused
- BLANK = Not asked or missing

To clean up the `CHECKUP1` data into a new variable called `t_checkup` we'll - convince R that the 7 and 9 values are in fact best interpreted as `NA`, - relabel options 1, 2, 3, 4 and 8 while turning the variable into a factor.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(t_checkup = fct_recode(factor(CHECKUP1),
                                 "1_In-past-year" = "1",
                                 "2_1-to-2-years" = "2",
                                 "3_2-to-5-years" = "3",
                                 "4_5_plus_years" = "4",
                                 "8_Never" = "8",
                                 NULL = "7",
                                 NULL = "9"))

smart_ohio_raw %>% count(CHECKUP1, t_checkup)
```

```
# A tibble: 7 x 3
  CHECKUP1 t_checkup          n
     <dbl> <fct>          <int>
1        1 1_In-past-year  5803
2        2 2_1-to-2-years   714
3        3 3_2-to-5-years   413
4        4 4_5_plus_years   376
5        7 <NA>              68
6        8 8_Never           32
7        9 <NA>               6
```

2.5.6 Blood Pressure (2 measures)

2.5.6.1 BPHIGH4 and its cleanup to bp_high

`BPHIGH4` is asking about awareness of a hypertension diagnosis. It's the response to the question: "Have you EVER been told by a doctor, nurse or other health professional that you have high blood pressure?" In addition, if the answer was "Yes" and the respondent is female, they were then asked "Was this only when you were pregnant?"

The available codes are:

- 1 = Yes
- 2 = Yes, but female told only during pregnancy
- 3 = No
- 4 = Told borderline high or pre-hypertensive
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

To clean up the `BPHIGH4` data into a new variable called `bp_high` we'll - convince R that the 7 and 9 values are in fact best interpreted as `NA`, - relabel (and re-order) options 1, 2, 3, 4 while turning the variable into a factor.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(bp_high = fct_recode(factor(BPHIGH4),
                                "0_No" = "3",
                                "1_Yes" = "1",
                                "2_Only_while_pregnant" = "2",
                                "4_Borderline" = "4",
                                NULL = "7",
                                NULL = "9"),
           bp_high = fct_relevel(bp_high,
                                "0_No", "1_Yes",
                                "2_Only_while_pregnant",
                                "4_Borderline"))

smart_ohio_raw %>% count(BPHIGH4, bp_high)
```

```
# A tibble: 6 x 3
  BPHIGH4 bp_high                   n
    <dbl> <fct>                 <int>
1       1 1_Yes                  3161
2       2 2_Only_while_pregnant    67
3       3 0_No                   4114
4       4 4_Borderline             49
5       7 <NA>                     19
6       9 <NA>                      2
```

2.5.6.2 `BPMEDS` and its cleanup to `bp_meds`

`BPMEDS` is the response to the question "Are you currently taking medicine for your high blood pressure?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

To clean up the `BPMEDS` data into a new variable called `bp_meds` we'll treat it just as we did with `HLTHPLN1` and - convince R that the 7 and 9 values are in fact best interpreted as `NA`, - and turn it into an indicator variable, e.g., we will leave the variable as numeric, but change the values to 1 = Yes and 0 = No.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(bp_meds = BPMEDS,
           bp_meds = replace(bp_meds, bp_meds %in% c(7, 9), NA),
           bp_meds = replace(bp_meds, bp_meds == 2, 0))

smart_ohio_raw %>% count(BPMEDS, bp_meds)
```

```
# A tibble: 5 x 3
  BPMEDS bp_meds     n
   <dbl>   <dbl> <int>
1      1       1  2675
2      2       0   481
3      7      NA     4
4      9      NA     1
5     NA      NA  4251
```

What is the relationship between our two blood pressure variables? Only the people with `bp_meds` = "1_Yes" were asked the `bp_meds` question.

```
smart_ohio_raw %>% tabyl(bp_high, bp_meds)
```

```
               bp_high   0    1  NA_
                  0_No   0    0 4114
                 1_Yes 481 2675    5
 2_Only_while_pregnant   0    0   67
          4_Borderline   0    0   49
                  <NA>   0    0   21
```

2.5.7 Cholesterol (3 items)

2.5.7.1 `CHOLCHK1` and its cleanup to `t_chol`

`CHOLCHK1`, the Length of time since cholesterol was checked, is the response to “Blood cholesterol is a fatty substance found in the blood. About how long has it been since you last had your blood cholesterol checked?”

- 1 = Never
- 2 = Within past year (anytime less than 12 months ago)
- 3 = Within past 2 years (1 year but less than 2 years ago)
- 4 = Within past 5 years (2 years but less than 5 years ago)
- 5 = 5 or more years ago
- 7 = Don’t know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

To clean up the `CHOLCHK1` data into a new variable called `t_chol` we’ll - convince R that the 7 and 9 values are in fact best interpreted as `NA`, - relabel options 1, 2, 3, 4 and 8 while turning the variable into a factor.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(t_chol = fct_recode(factor(CHOLCHK1),
                                "1_Never" = "1",
                                "2_In-past-year" = "2",
                                "3_1-to-2-years" = "3",
                                "4_2-to-5-years" = "4",
                                "5_5_plus_years" = "5",
                                NULL = "7",
                                NULL = "9"))

smart_ohio_raw %>% count(CHOLCHK1, t_chol)
```

```
# A tibble: 8 x 3
  CHOLCHK1 t_chol             n
     <dbl> <fct>          <int>
1        1 1_Never          424
2        2 2_In-past-year  5483
3        3 3_1-to-2-years   559
4        4 4_2-to-5-years   289
5        5 5_5_plus_years   272
6        7 <NA>             376
7        9 <NA>               8
8       NA <NA>               1
```

The next two measures are not gathered from the people who answered “Never” to this question.

2.5.7.2 TOLDHI2 and its cleanup to chol_high

TOLDHI2 is asking about awareness of a diagnosis of high cholesterol. It's the response to the question: "Have you EVER been told by a doctor, nurse or other health professional that your blood cholesterol is high?"

The available codes are:

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

To clean up the TOLDHI2 data into a new variable called chol_high we'll treat it like BPMEDS and HLTHPLN1 - convince R that the 7 and 9 values are in fact best interpreted as NA, - and turn it into an indicator variable, e.g., we will leave the variable as numeric, but change the values to 1 = Yes and 0 = No.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(chol_high = TOLDHI2,
           chol_high = replace(chol_high, chol_high %in% c(7, 9), NA),
           chol_high = replace(chol_high, chol_high == 2, 0))

smart_ohio_raw %>% count(TOLDHI2, chol_high)
```

```
# A tibble: 5 x 3
  TOLDHI2 chol_high     n
    <dbl>     <dbl> <int>
1       1         1  2612
2       2         0  4286
3       7        NA    70
4       9        NA     4
5      NA        NA   440
```

2.5.7.3 CHOLMED1 and its cleanup to chol_meds

CHOLMED1 is the response to the question "Are you currently taking medicine prescribed by a doctor or other health professional for your blood cholesterol?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

To clean up the CHOLMED1 data into a new variable called chol_meds we'll treat it just as we did with HLTHPLN1 and - convince R that the 7 and 9 values are in fact best interpreted as `NA`, - and turn it into an indicator variable, e.g., we will leave the variable as numeric, but change the values to 1 = Yes and 0 = No.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(chol_meds = CHOLMED1,
           chol_meds = replace(chol_meds, chol_meds %in% c(7, 9), NA),
           chol_meds = replace(chol_meds, chol_meds == 2, 0))

smart_ohio_raw %>% count(CHOLMED1, chol_meds)
```

```
# A tibble: 4 x 3
  CHOLMED1 chol_meds     n
     <dbl>     <dbl> <int>
1        1         1  1781
2        2         0   826
3        7        NA     5
4       NA        NA  4800
```

2.5.8 Chronic Health Conditions (14 items)

2.5.8.1 Self-reported diagnosis history (11 items)

The next few variables describe whether or not the subject meets a particular standard, and are all coded in the raw data the same way:

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

and we'll recode them all to 1 = Yes, 0 = No, otherwise NA, as we've done previously.

The questions are all started with "Has a doctor, nurse, or other health professional ever told you that you had any of the following? For each, tell me Yes, No, or you're Not sure."

Original	Revised	Details
`CVDINFR4`	`hx_mi`	(Ever told) you had a heart attack, also called a myocardial infarction?
`CVDCRHD4`	`hx_chd`	(Ever told) you had angina or coronary heart disease?
`CVDSTRK3`	`hx_stroke`	(Ever told) you had a stroke?
`ASTHMA3`	`hx_asthma`	(Ever told) you had asthma?
`ASTHNOW`	`now_asthma`	Do you still have asthma? (only asked of those with Yes in `ASTHMA3`)

Original	Revised	Details
`CHCSCNCR`	`hx_skinc`	(Ever told) you had skin cancer?
`CHCOCNCR`	`hx_otherc`	(Ever told) you had any other types of cancer?
`CHCCOPD1`	`hx_copd`	(Ever told) you have Chronic Obstructive Pulmonary Disease or COPD, emphysema or chronic bronchitis?
`HAVARTH3`	`hx_arthr`	(Ever told) you have some form of arthritis, rheumatoid arthritis, gout, lupus, or fibromyalgia? (Arthritis diagnoses include: rheumatism, polymyalgia rheumatica; osteoarthritis (not osteporosis); tendonitis, bursitis, bunion, tennis elbow; carpal tunnel syndrome, tarsal tunnel syndrome; joint infection, etc.)
`ADDEPEV2`	`hx_depress`	(Ever told) you that you have a depressive disorder, including depression, major depression, dysthymia, or minor depression?
`CHCKIDNY`	`hx_kidney`	(Ever told) you have kidney disease? Do NOT include kidney stones, bladder infection or incontinence.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(hx_mi = CVDINFR4,
           hx_mi = replace(hx_mi, hx_mi %in% c(7, 9), NA),
           hx_mi = replace(hx_mi, hx_mi == 2, 0),
           hx_chd = CVDCRHD4,
           hx_chd = replace(hx_chd, hx_chd %in% c(7, 9), NA),
           hx_chd = replace(hx_chd, hx_chd == 2, 0),
           hx_stroke = CVDSTRK3,
           hx_stroke = replace(hx_stroke, hx_stroke %in% c(7, 9), NA),
           hx_stroke = replace(hx_stroke, hx_stroke == 2, 0),
           hx_asthma = ASTHMA3,
           hx_asthma = replace(hx_asthma, hx_asthma %in% c(7, 9), NA),
           hx_asthma = replace(hx_asthma, hx_asthma == 2, 0),
           now_asthma = ASTHNOW,
           now_asthma = replace(now_asthma, now_asthma %in% c(7, 9), NA),
           now_asthma = replace(now_asthma, now_asthma == 2, 0),
           hx_skinc = CHCSCNCR,
           hx_skinc = replace(hx_skinc, hx_skinc %in% c(7, 9), NA),
           hx_skinc = replace(hx_skinc, hx_skinc == 2, 0),
           hx_otherc = CHCOCNCR,
           hx_otherc = replace(hx_otherc, hx_otherc %in% c(7, 9), NA),
           hx_otherc = replace(hx_otherc, hx_otherc == 2, 0),
           hx_copd = CHCCOPD1,
           hx_copd = replace(hx_copd, hx_copd %in% c(7, 9), NA),
           hx_copd = replace(hx_copd, hx_copd == 2, 0),
           hx_arthr = HAVARTH3,
           hx_arthr = replace(hx_arthr, hx_arthr %in% c(7, 9), NA),
           hx_arthr = replace(hx_arthr, hx_arthr == 2, 0),
           hx_depress = ADDEPEV2,
           hx_depress = replace(hx_depress, hx_depress %in% c(7, 9), NA),
           hx_depress = replace(hx_depress, hx_depress == 2, 0),
           hx_kidney = CHCKIDNY,
           hx_kidney = replace(hx_kidney, hx_kidney %in% c(7, 9), NA),
           hx_kidney = replace(hx_kidney, hx_kidney == 2, 0))
```

We definitely should have written a function to do that, of course.

2.5.8.2 `_ASTHMS1` and its cleanup to `asthma`

`_ASTHMS1` categorizes subjects by asthma status as:

- 1 = Current
- 2 = Former
- 3 = Never
- 9 = Don't Know / Not Sure / Refused / Missing

We'll turn this into a factor with appropriate levels and NA information.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(asthma = fct_recode(
        factor(`_ASTHMS1`),
        "Current" = "1",
        "Former" = "2",
        "Never" = "3",
        NULL = "9"))

smart_ohio_raw %>% count(`_ASTHMS1`, asthma)
```

```
# A tibble: 4 x 3
  `_ASTHMS1` asthma      n
       <dbl> <fct>   <int>
1          1 Current   734
2          2 Former    248
3          3 Never    6376
4          9 <NA>       54
```

2.5.8.3 `DIABETE3` and its cleanup to `hx_diabetes` and `dm_status`

`DIABETE3`, the (Ever told) you have diabetes variable, is the response to "(Ever told) you have diabetes (If Yes and respondent is female, ask Was this only when you were pregnant?. If Respondent says pre-diabetes or borderline diabetes, use response code 4.)"

- 1 = Yes
- 2 = Yes, but female told only during pregnancy
- 3 = No
- 4 = No, pre-diabetes or borderline diabetes
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

I'll create one variable called `hx_diabetes` which is 1 if `DIABETE3` = 1, and 0 otherwise, with appropriate NAs, like our other variables. Then I'll create `dm_status` to include all of this information in a factor, but again recode the missing values properly.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(hx_diabetes = DIABETE3,
           hx_diabetes = replace(hx_diabetes, hx_diabetes %in% c(7, 9), NA),
           hx_diabetes = replace(hx_diabetes, hx_diabetes %in% 2:4, 0),
           dm_status = fct_recode(factor(DIABETE3),
                                  "Diabetes" = "1",
                                  "Pregnancy-Induced" = "2",
                                  "No-Diabetes" = "3",
                                  "Pre-Diabetes" = "4",
                                  NULL = "7",
                                  NULL = "9"),
           dm_status = fct_relevel(dm_status,
                                   "No-Diabetes",
                                   "Pre-Diabetes",
                                   "Pregnancy-Induced",
                                   "Diabetes"))

smart_ohio_raw %>% count(DIABETE3, hx_diabetes, dm_status)
```

```
# A tibble: 6 x 4
  DIABETE3 hx_diabetes dm_status             n
     <dbl>       <dbl> <fct>             <int>
1        1           1 Diabetes           1098
2        2           0 Pregnancy-Induced    67
3        3           0 No-Diabetes        6100
4        4           0 Pre-Diabetes        133
5        7          NA <NA>                 12
6        9          NA <NA>                  2
```

2.5.8.4 `DIABAGE2` and its cleanup to `dm_age`

`DIABAGE2`, the Age When Told Diabetic variable, is the response to "How old were you when you were told you have diabetes?" It is asked only of people with

`DIABETE3 = 1` (Yes).

- The response is 1-97, with special values 98 for Don't Know/Not Sure and 99 for refused, with BLANK for missing or not asked. People 97 years of age and above were listed as 97.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(dm_age = DIABAGE2,
           dm_age = replace(dm_age, dm_age > 97, NA))

smart_ohio_raw %>% count(DIABAGE2, dm_age) %>% tail()
```

```
# A tibble: 6 x 3
  DIABAGE2 dm_age     n
     <dbl>  <dbl> <int>
1       84     84     1
2       85     85     2
3       90     90     1
4       98     NA    61
5       99     NA     4
6       NA     NA  6314
```

2.5.9 Arthritis Burden (4 items)

The first two measures are only asked of people with `hx_arthr` = 1, and are coded as:

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

and we'll recode them to 1 = Yes, 0 = No, otherwise NA, as we've done previously.

2.5.9.1 `LMTJOIN3` (Limited because of joint symptoms), and its cleanup to `arth_lims`

This is the response to "Are you now limited in any way in any of your usual activities because of arthritis or joint symptoms?"

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(arth_lims = LMTJOIN3,
           arth_lims = replace(arth_lims, arth_lims %in% c(7, 9), NA),
           arth_lims = replace(arth_lims, arth_lims == 2, 0))

smart_ohio_raw %>% count(hx_arthr, LMTJOIN3, arth_lims)
```

```
# A tibble: 6 x 4
  hx_arthr LMTJOIN3 arth_lims     n
     <dbl>    <dbl>     <dbl> <int>
1        0       NA        NA  4587
2        1        1         1  1378
3        1        2         0  1388
4        1        7        NA    17
5        1        9        NA     2
6       NA       NA        NA    40
```

2.5.9.2 `ARTHDIS2` (Does Arthritis Affect Whether You Work), and its cleanup to `arth_work`

This is the response to "Do arthritis or joint symptoms now affect whether you work, the type of work you do or the amount of work you do?"

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(arth_work = ARTHDIS2,
           arth_work = replace(arth_work, arth_work %in% c(7, 9), NA),
           arth_work = replace(arth_work, arth_work == 2, 0))

smart_ohio_raw %>% count(ARTHDIS2, arth_work)
```

```
# A tibble: 5 x 3
  ARTHDIS2 arth_work     n
     <dbl>     <dbl> <int>
1        1         1   925
2        2         0  1808
3        7        NA    42
4        9        NA    10
5       NA        NA  4627
```

2.5.9.3 `ARTHSOCL` (Social Activities Limited Because of Joint Symptoms) and its cleanup to `arth_soc`

This is the response to "During the past 30 days, to what extent has your arthritis or joint symptoms interfered with your normal social activities, such as going shopping, to the movies, or to religious or social gatherings?"

The responses are:

- 1 = A lot
- 2 = A little
- 3 = Not at all
- 7 = Don't know/Not sure
- 9 = Refused

- BLANK = Not asked or missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(arth_soc = fct_recode(factor(ARTHSOCL),
                                  "A lot" = "1",
                                  "A little" = "2",
                                  "Not at all" = "3",
                                  NULL = "7",
                                  NULL = "9"))

smart_ohio_raw %>% count(ARTHSOCL, arth_soc)
```

```
# A tibble: 6 x 3
  ARTHSOCL arth_soc       n
     <dbl> <fct>      <int>
1        1 A lot        606
2        2 A little     734
3        3 Not at all  1427
4        7 <NA>          15
5        9 <NA>           3
6       NA <NA>        4627
```

2.5.9.4 `JOINPAI1` (How Bad Was Joint Pain - scale of 0-10) and its cleanup to `joint_pain`

This is the response to the following question: "Please think about the past 30 days, keeping in mind all of your joint pain or aching and whether or not you have taken medication. On a scale of 0 to 10 where 0 is no pain or aching and 10 is pain or aching as bad as it can be, DURING THE PAST 30 DAYS, how bad was your joint pain ON AVERAGE?"

The available values are 0-10, plus codes 77 (Don't Know / Not Sure), 99 (Refused) and BLANK.

To clean up `JOINPAI1` to a new variable called `joint_pain`, we'll need to convince R that the 77 and 99 values are, like `BLANK`, in fact best interpreted as `NA`.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(joint_pain = JOINPAI1,
           joint_pain = replace(joint_pain, joint_pain %in% c(77, 99), NA))

smart_ohio_raw %>% count(JOINPAI1, joint_pain) %>% tail()
```

```
# A tibble: 6 x 3
  JOINPAI1 joint_pain     n
     <dbl>      <dbl> <int>
1        8          8   277
```

```
2         9            9    72
3        10           10   158
4        77           NA    28
5        99           NA     5
6        NA           NA  4627
```

2.5.10 Demographics (25 items)

2.5.10.1 `_AGEG5YR`, which we'll edit into `agegroup`

The `_AGEG5YR` variable is a calculated variable (by CDC) obtained from the subject's age. Since the `age` data are not available, we instead get these groupings, which we'll rearrange into the `agegroup` factor.

`_AGEG5YR`	Age range	`agegroup`
1	18 <= AGE <= 24	18-24
2	25 <= AGE <= 29	25-29
3	30 <= AGE <= 34	30-34
4	35 <= AGE <= 39	35-39
5	40 <= AGE <= 44	40-44
6	45 <= AGE <= 49	45-49
7	50 <= AGE <= 54	50-54
8	55 <= AGE <= 59	55-59
9	60 <= AGE <= 64	60-64
10	65 <= AGE <= 69	65-69
11	70 <= AGE <= 74	70-74
12	75 <= AGE <= 79	75-79
13	AGE >= 80	80plus
14	Don't Know, Refused or Missing	NA

```r
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(agegroup = fct_recode(factor(`_AGEG5YR`),
                                "18-24" = "1",
                                "25-29" = "2",
                                "30-34" = "3",
                                "35-39" = "4",
                                "40-44" = "5",
                                "45-49" = "6",
                                "50-54" = "7",
                                "55-59" = "8",
                                "60-64" = "9",
                                "65-69" = "10",
                                "70-74" = "11",
                                "75-79" = "12",
                                "80-96" = "13",
                                NULL = "14"))

smart_ohio_raw %>% count(`_AGEG5YR`, agegroup)
```

```
# A tibble: 14 x 3
   `_AGEG5YR` agegroup     n
        <dbl> <fct>    <int>
 1          1 18-24      448
 2          2 25-29      327
 3          3 30-34      375
 4          4 35-39      446
 5          5 40-44      426
 6          6 45-49      509
 7          7 50-54      604
 8          8 55-59      786
 9          9 60-64      837
10         10 65-69      810
11         11 70-74      685
12         12 75-79      499
13         13 80-96      592
14         14 <NA>        68
```

2.5.10.2 `_MRACE1` recoded to `race`

We'll create three variables describing race/ethnicity. The first comes from the `_MRACE1` variable categorized by CDC, and the available responses are:

- 1 = White only
- 2 = Black or African-American only
- 3 = American Indian or Alaskan Native only
- 4 = Asian only
- 5 = Native Hawaiian or Pacific Islander only
- 6 = Other race only
- 7 = Multiracial
- 77 = Don't know / Not Sure
- 99 = Refused
- BLANK = Missing

We'll create a factor out of this information, with appropriate level names.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(race = fct_recode(factor(`_MRACE1`),
                              "White" = "1",
                              "Black or African A" = "2",
                              "Amer Indian or Alaskan" = "3",
                              "Asian" = "4",
                              "Hawaiian or Pac Island" = "5",
                              "Other Race" = "6",
                              "Multiracial" = "7",
                              NULL = "77",
                              NULL = "99"))

smart_ohio_raw %>% count(`_MRACE1`, race)
```

```
# A tibble: 9 x 3
  `_MRACE1` race                       n
      <dbl> <fct>                  <int>
1         1 White                   6177
2         2 Black or African A       739
3         3 Amer Indian or Alaskan    66
4         4 Asian                    115
5         5 Hawaiian or Pac Island     5
6         6 Other Race                43
7         7 Multiracial              153
8        77 <NA>                      14
9        99 <NA>                     100
```

2.5.10.3 `_HISPANC` recoded to `hispanic`

The `_HISPANC` variable specifies whether or not the respondent is of Hispanic or Latinx origin. The available responses are:

- 1 = Hispanic, Latinx or Spanish origin
- 2 = Not of Hispanic, Latinx or Spanish origin
- 9 = Don't Know, Refused, or Missing

We'll turn the 9s into NA, and create an indicator variable (1 = Hispanic or Latinx, 0 = not)

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(hispanic = 2 - `_HISPANC`,
           hispanic = replace(hispanic, hispanic < 0, NA))

smart_ohio_raw %>% count(`_HISPANC`, hispanic)
```

```
# A tibble: 3 x 3
  `_HISPANC` hispanic     n
       <dbl>    <dbl> <int>
1          1        1   146
2          2        0  7217
```

```
3           9       NA    49
```

2.5.10.4 _RACEGR3 recoded to race_eth

The _RACEGR3 variable is a five-level combination of race and ethnicity. The responses are:

- 1 = White non-Hispanic
- 2 = Black non-Hispanic
- 3 = Other race non-Hispanic
- 4 = Multiracial non-Hispanic
- 5 = Hispanic
- 9 = Don't Know / Not Sure / Refused

We'll create a factor out of this information, with appropriate level names.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(race_eth = fct_recode(
        factor(`_RACEGR3`),
        "White non-Hispanic" = "1",
        "Black non-Hispanic" = "2",
        "Other race non-Hispanic" = "3",
        "Multiracial non-Hispanic" = "4",
        "Hispanic" = "5",
        NULL = "9"))

smart_ohio_raw %>% count(`_RACEGR3`, race_eth)
```

```
# A tibble: 6 x 3
  `_RACEGR3` race_eth                     n
       <dbl> <fct>                    <int>
1          1 White non-Hispanic        6086
2          2 Black non-Hispanic         725
3          3 Other race non-Hispanic    193
4          4 Multiracial non-Hispanic   143
5          5 Hispanic                   146
6          9 <NA>                       119
```

2.5.10.5 SEX recoded to female

The available levels of SEX are:

- 1 = Male
- 2 = Female
- 9 = Refused

We'll recode that to `female` = 1 for Female, 0 Male, otherwise NA. Note the trick here is to subtract one from the coded `SEX` to get the desired `female`, but this requires that we move 9 to NA, rather than 9.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(female = SEX - 1,
           female = replace(female, female == 8, NA))

smart_ohio_raw %>% count(SEX, female)
```

```
# A tibble: 2 x 3
    SEX female     n
  <dbl>  <dbl> <int>
1     1      0  3136
2     2      1  4276
```

2.5.10.6 `MARITAL` status, revised to `marital`

The available levels of `MARITAL` are:

- 1 = Married
- 2 = Divorced
- 3 = Widowed
- 4 = Separated
- 5 = Never married
- 6 = A member of an unmarried couple
- 9 = Refused
- BLANK = Not asked or missing

We'll just turn this into a factor, and move 9 to NA.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(marital = fct_recode(factor(MARITAL),
                                "Married" = "1",
                                "Divorced" = "2",
                                "Widowed" = "3",
                                "Separated" = "4",
                                "Never_Married" = "5",
                                "Unmarried_Couple" = "6",
                                NULL = "9"))

smart_ohio_raw %>% count(MARITAL, marital)
```

```
# A tibble: 7 x 3
  MARITAL marital              n
    <dbl> <fct>            <int>
1       1 Married           3668
```

```
2        2 Divorced          1110
3        3 Widowed            978
4        4 Separated          142
5        5 Never_Married     1248
6        6 Unmarried_Couple   208
7        9 <NA>                58
```

2.5.10.7 EDUCA recoded to educgroup

The available levels of `EDUCA` (Education Level) are responses to: "What is the highest grade or year of school you completed?"

- 1 = Never attended school or only kindergarten
- 2 = Grades 1 through 8 (Elementary)
- 3 = Grades 9 through 11 (Some high school)
- 4 = Grade 12 or GED (High school graduate)
- 5 = College 1 year to 3 years (Some college or technical school)
- 6 = College 4 years or more (College graduate)
- 9 = Refused
- BLANK = Not asked or missing

We'll just turn this into a factor, and move 9 to NA.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(educgroup = fct_recode(factor(EDUCA),
                                  "Kindergarten" = "1",
                                  "Elementary" = "2",
                                  "Some_HS" = "3",
                                  "HS_Grad" = "4",
                                  "Some_College" = "5",
                                  "College_Grad" = "6",
                                  NULL = "9"))

smart_ohio_raw %>% count(EDUCA, educgroup)
```

```
# A tibble: 7 x 3
  EDUCA educgroup        n
  <dbl> <fct>        <int>
1     1 Kindergarten     3
2     2 Elementary     117
3     3 Some_HS        332
4     4 HS_Grad       2209
5     5 Some_College  2079
6     6 College_Grad  2646
7     9 <NA>            26
```

2.5.10.8 `RENTHOM1` recoded to `home_own`

The available levels of `RENTHOM1` (Own or Rent Home) are responses to: "Do you own or rent your home? (Home is defined as the place where you live most of the time/the majority of the year.)"

- 1 = Own
- 2 = Rent
- 3 = Other Arrangement
- 7 = Don't know/Not Sure
- 9 = Refused
- BLANK = Not asked or missing

We'll recode as `home_own` = 1 if they own their home, and 0 otherwise, and dealing with missingness properly.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(home_own = RENTHOM1,
           home_own = replace(home_own, home_own %in% c(7,9), NA),
           home_own = replace(home_own, home_own %in% c(2,3), 0))

smart_ohio_raw %>% count(RENTHOM1, home_own)
```

```
# A tibble: 5 x 3
  RENTHOM1 home_own     n
     <dbl>    <dbl> <int>
1        1        1  5216
2        2        0  1793
3        3        0   348
4        7       NA    28
5        9       NA    27
```

2.5.10.9 `CPDEMO1A` and its cleanup to `cell_own`

`CPDEMO1A` is the response to "Including phones for business and personal use, do you have a cell phone for personal use?"

Available responses are:

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

and we'll recode them to 1 = Yes, 0 = No, otherwise NA, as we've done previously.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(cell_own = 2 - CPDEMO1A,
           cell_own = replace(cell_own, cell_own < 0, NA))

smart_ohio_raw %>% count(CPDEMO1A, cell_own)
```

```
# A tibble: 5 x 3
  CPDEMO1A cell_own     n
     <dbl>    <dbl> <int>
1        1        1  2930
2        2        0   698
3        7       NA     2
4        9       NA    19
5       NA       NA  3763
```

2.5.10.10 `VETERAN3` and its cleanup to `veteran`

`VETERAN3`, the Are You A Veteran variable, is the response to "Have you ever served on active duty in the United States Armed Forces, either in the regular military or in a National Guard or military reserve unit? (Active duty does not include training for the Reserves or National Guard, but DOES include activation, for example, for the Persian Gulf War.)"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(veteran = VETERAN3,
           veteran = replace(veteran, veteran %in% c(7, 9), NA),
           veteran = replace(veteran, veteran == 2, 0))

smart_ohio_raw %>% count(VETERAN3, veteran)
```

```
# A tibble: 3 x 3
  VETERAN3 veteran     n
     <dbl>   <dbl> <int>
1        1       1   927
2        2       0  6479
3        9      NA     6
```

2.5.10.11 `EMPLOY1` and its cleanup to `employment`

`EMPLOY1`, the Employment Status variable, is the response to “Are you currently . . . ?”

- 1 = Employed for wages
- 2 = Self-employed
- 3 = Out of work for 1 year or more
- 4 = Out of work for less than 1 year
- 5 = A homemaker
- 6 = A student
- 7 = Retired
- 8 = Unable to work
- 9 = Refused
- BLANK = Not asked or missing

We’ll just turn this into a factor, and move 9 to NA.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(employment = fct_recode(factor(EMPLOY1),
                                    "Employed_for_wages" = "1",
                                    "Self-employed" = "2",
                                    "Outofwork_1yearormore" = "3",
                                    "Outofwork_lt1year" = "4",
                                    "Homemaker" = "5",
                                    "Student" = "6",
                                    "Retired" = "7",
                                    "Unable_to_work" = "8",
                                    NULL = "9"))

smart_ohio_raw %>% count(EMPLOY1, employment)
```

```
# A tibble: 9 x 3
  EMPLOY1 employment                n
    <dbl> <fct>                 <int>
1       1 Employed_for_wages     3119
2       2 Self-employed           466
3       3 Outofwork_1yearormore   254
4       4 Outofwork_lt1year       134
5       5 Homemaker               411
6       6 Student                 190
7       7 Retired                2202
8       8 Unable_to_work          603
9       9 <NA>                     33
```

2.5.10.12 CHILDREN and its cleanup to kids

CHILDREN, the Number of Children in Household variable, is the response to "How many children less than 18 years of age live in your household?"

- 1-87 = legitimate responses
- 88 = None
- 99 = Refused
- BLANK = Not asked or missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(kids = CHILDREN,
           kids = replace(kids, kids == 99, NA),
           kids = replace(kids, kids == 88, 0))

smart_ohio_raw %>% count(CHILDREN, kids) %>% tail()
```

```
# A tibble: 6 x 3
  CHILDREN  kids     n
     <dbl> <dbl> <int>
1        6     6     7
2        7     7     5
3        8     8     2
4       12    12     1
5       88     0  5449
6       99    NA    43
```

2.5.10.13 INCOME2 to incomegroup

The available levels of INCOME2 (Income Level) are responses to: "Is your annual household income from all sources ..."

- 1 = Less than $10,000
- 2 = $10,000 to less than $15,000
- 3 = $15,000 to less than $20,000
- 4 = $20,000 to less than $25,000
- 5 = $25,000 to less than $35,000
- 6 = $35,000 to less than $50,000
- 7 = $50,000 to less than $75,000
- 8 = $75,000 or more
- 77 = Don't know/Not sure
- 99 = Refused
- BLANK = Not asked or missing

We'll just turn this into a factor, and move 77 and 99 to NA.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(incomegroup = fct_recode(factor(`INCOME2`),
                                "0-9K" = "1",
                                "10-14K" = "2",
                                "15-19K" = "3",
                                "20-24K" = "4",
                                "25-34K" = "5",
                                "35-49K" = "6",
                                "50-74K" = "7",
                                "75K+" = "8",
                                NULL = "77",
                                NULL = "99"))

smart_ohio_raw %>% count(`INCOME2`, incomegroup)
```

```
# A tibble: 11 x 3
   INCOME2 incomegroup     n
     <dbl> <fct>       <int>
 1       1 0-9K          285
 2       2 10-14K        306
 3       3 15-19K        477
 4       4 20-24K        589
 5       5 25-34K        685
 6       6 35-49K        922
 7       7 50-74K        928
 8       8 75K+         1910
 9      77 <NA>          610
10      99 <NA>          678
11      NA <NA>           22
```

2.5.10.14 `INTERNET` and its cleanup to `internet30`

`INTERNET`, the Internet use in the past 30 days variable, is the response to "Have you used the internet in the past 30 days?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(internet30 = INTERNET,
           internet30 = replace(internet30, internet30 %in% c(7, 9), NA),
           internet30 = replace(internet30, internet30 == 2, 0))
```

```
smart_ohio_raw %>% count(INTERNET, internet30)
```

```
# A tibble: 5 x 3
  INTERNET internet30     n
     <dbl>      <dbl> <int>
1        1          1  6020
2        2          0  1335
3        7         NA    10
4        9         NA    10
5       NA         NA    37
```

2.5.10.15 `WTKG3` is `weight_kg`

`WTKG3` is computed by CDC, as the respondent's weight in kilograms with two implied decimal places. We calculate the actual weight in kg, with the following:

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(weight_kg = WTKG3/100)

smart_ohio_raw %>% count(WTKG3, weight_kg) %>% tail()
```

```
# A tibble: 6 x 3
  WTKG3 weight_kg     n
  <dbl>     <dbl> <int>
1 19051      191.     1
2 19278      193.     1
3 19504      195.     1
4 20412      204.     2
5 20865      209.     1
6    NA        NA   462
```

2.5.10.16 `HEIGHT3` is replaced with `height_m`

`HEIGHT3` is strangely gathered to allow people to specify their height in either feet and inches or in meters and centimeters.

- 200-711 indicates height in feet (first digit) and inches (second two digits)
- 9000 - 9998 indicates height in meters (second digit) and centimeters (last two digits)
- 7777 = Don't know/Not sure
- 9999 = Refused

Note that there is one impossible value of 575 in the data set. We'll make that an NA, and we'll also make NA any heights below 3 feet, or above 2.24 meters. Specifically, we calculate the actual height in meters, with the following:

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(height_m = case_when(
        HEIGHT3 >= 300 & HEIGHT3 <= 511 ~ round((12*floor(HEIGHT3/100) + (HEIGHT3 - 10
        HEIGHT3 >= 600 & HEIGHT3 <= 711 ~ round((12*floor(HEIGHT3/100) + (HEIGHT3 - 10
        HEIGHT3 >= 9000 & HEIGHT3 <= 9224 ~ ((HEIGHT3 - 9000)/100)))

smart_ohio_raw %>% count(HEIGHT3, height_m) %>% tail()
```

```
# A tibble: 6 x 3
  HEIGHT3 height_m     n
    <dbl>    <dbl> <int>
1     607     2.01     2
2     608     2.03     6
3     609     2.06     1
4    7777    NA       27
5    9999    NA       86
6      NA    NA       67
```

2.5.10.17 `bmi` is calculated from `height_m` and `weight_kg`

We'll calculate body-mass index from height and weight.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(bmi = round(weight_kg/(height_m)^2,2))

smart_ohio_raw %>% count(height_m, weight_kg, bmi)# %>% tail()
```

```
# A tibble: 1,806 x 4
   height_m weight_kg   bmi     n
      <dbl>     <dbl> <dbl> <int>
 1     1.35      39.0  21.4     1
 2     1.35      52.2  28.6     1
 3     1.4       89.8  45.8     1
 4     1.42      31.8  15.8     1
 5     1.42      45.4  22.5     1
 6     1.42      55.8  27.7     1
 7     1.42      58.5  29.0     1
 8     1.42      59.9  29.7     1
 9     1.42      60.8  30.1     1
10     1.42      71.2  35.3     1
# ... with 1,796 more rows
```

2.5.10.18 `bmigroup` is calculated from `bmi`

We'll then divide the respondents into adult BMI categories, in the usual way.

- BMI < 18.5 indicates underweight
- BMI from 18.5 up to 25 indicates normal weight
- BMI from 25 up to 30 indicates overweight
- BMI of 30 and higher indicates obesity

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(bmigroup = factor(cut2(as.numeric(bmi),
                             cuts = c(18.5, 25.0, 30.0))))

smart_ohio_raw %>% count(bmigroup)
```

```
# A tibble: 5 x 2
  bmigroup        n
* <fct>       <int>
1 [13.3,18.5)   119
2 [18.5,25.0)  2017
3 [25.0,30.0)  2445
4 [30.0,75.5]  2338
5 <NA>          493
```

2.5.10.19 `PREGNANT` and its cleanup to `pregnant`

`PREGNANT`, the Pregnancy Status variable, is the response to “To your knowledge, are you now pregnant?”

- 1 = Yes
- 2 = No
- 7 = Don’t know/Not sure
- 9 = Refused
- BLANK = Not asked or missing (includes SEX = male)

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(pregnant = PREGNANT,
           pregnant = replace(pregnant, pregnant %in% c(7, 9), NA),
           pregnant = replace(pregnant, pregnant == 2, 0))

smart_ohio_raw %>% count(PREGNANT, pregnant)
```

```
# A tibble: 5 x 3
  PREGNANT pregnant     n
     <dbl>    <dbl> <int>
1        1        1    41
2        2        0  1329
3        7       NA     3
4        9       NA     3
5       NA       NA  6036
```

2.5.10.20 `DEAF` and its cleanup to `deaf`

`DEAF`, the Are you deaf or do you have serious difficulty hearing variable, is the response to "Are you deaf or do you have serious difficulty hearing?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(deaf = DEAF,
           deaf = replace(deaf, deaf %in% c(7, 9), NA),
           deaf = replace(deaf, deaf == 2, 0))

smart_ohio_raw %>% count(DEAF, deaf)
```

```
# A tibble: 5 x 3
   DEAF  deaf     n
  <dbl> <dbl> <int>
1     1     1   708
2     2     0  6551
3     7    NA    15
4     9    NA     4
5    NA    NA   134
```

2.5.10.21 `BLIND` and its cleanup to `blind`

`BLIND`, the Blind or Difficulty seeing variable, is the response to "Are you blind or do you have serious difficulty seeing, even when wearing glasses?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(blind = BLIND,
           blind = replace(blind, blind %in% c(7, 9), NA),
           blind = replace(blind, blind == 2, 0))

smart_ohio_raw %>% count(BLIND, blind)
```

```
# A tibble: 5 x 3
  BLIND blind     n
  <dbl> <dbl> <int>
1     1     1   415
2     2     0  6834
3     7    NA    14
4     9    NA     1
5    NA    NA   148
```

2.5.10.22 `DECIDE` and its cleanup to `decide`

`DECIDE`, the Difficulty Concentrating or Remembering variable, is the response to "Because of a physical, mental, or emotional condition, do you have serious difficulty concentrating, remembering, or making decisions?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(decide = DECIDE,
           decide = replace(decide, decide %in% c(7, 9), NA),
           decide = replace(decide, decide == 2, 0))

smart_ohio_raw %>% count(DECIDE, decide)
```

```
# A tibble: 5 x 3
  DECIDE decide     n
   <dbl>  <dbl> <int>
1      1      1   870
2      2      0  6348
3      7     NA    30
4      9     NA     2
5     NA     NA   162
```

2.5.10.23 `DIFFWALK` and its cleanup to `diffwalk`

`DIFFWALK`, the Difficulty Walking or Climbing Stairs variable, is the response to "Do you have serious difficulty walking or climbing stairs?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(diffwalk = DIFFWALK,
           diffwalk = replace(diffwalk, diffwalk %in% c(7, 9), NA),
           diffwalk = replace(diffwalk, diffwalk == 2, 0))

smart_ohio_raw %>% count(DIFFWALK, diffwalk)
```

```
# A tibble: 5 x 3
  DIFFWALK diffwalk     n
     <dbl>    <dbl> <int>
1        1        1  1482
2        2        0  5738
3        7       NA    19
4        9       NA     2
5       NA       NA   171
```

2.5.10.24 `DIFFDRES` and its cleanup to `diffdress`

`DIFFDRES`, the Difficulty Dressing or Bathing variable, is the response to "Do you have difficulty dressing or bathing?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(diffdress = DIFFDRES,
           diffdress = replace(diffdress, diffdress %in% c(7, 9), NA),
           diffdress = replace(diffdress, diffdress == 2, 0))

smart_ohio_raw %>% count(DIFFDRES, diffdress)
```

```
# A tibble: 5 x 3
  DIFFDRES diffdress     n
     <dbl>     <dbl> <int>
1        1         1   352
2        2         0  6868
3        7        NA    12
4        9        NA     1
5       NA        NA   179
```

2.5.10.25 DIFFALON and its cleanup to diffalone

DIFFALON, the Difficulty Doing Errands Alone variable, is the response to "Because of a physical, mental, or emotional condition, do you have difficulty doing errands alone such as visiting a doctor's office or shopping?"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(diffalone = DIFFALON,
           diffalone = replace(diffalone, diffalone %in% c(7, 9), NA),
           diffalone = replace(diffalone, diffalone == 2, 0))

smart_ohio_raw %>% count(DIFFALON, diffalone)
```

```
# A tibble: 5 x 3
  DIFFALON diffalone     n
     <dbl>     <dbl> <int>
1        1         1   636
2        2         0  6560
3        7        NA    15
4        9        NA     4
5       NA        NA   197
```

2.5.11 Tobacco Use (2 items)

2.5.11.1 SMOKE100 and its cleanup to smoke100

SMOKE100, the Smoked at Least 100 Cigarettes variable, is the response to "Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes]"

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(smoke100 = SMOKE100,
           smoke100 = replace(smoke100, smoke100 %in% c(7, 9), NA),
           smoke100 = replace(smoke100, smoke100 == 2, 0))

smart_ohio_raw %>% count(SMOKE100, smoke100)
```

```
# A tibble: 5 x 3
  SMOKE100 smoke100     n
     <dbl>    <dbl> <int>
1        1        1  3294
2        2        0  3881
3        7       NA    31
4        9       NA     4
5       NA       NA   202
```

2.5.11.2 `_SMOKER3` and its cleanup to `smoker`

`_SMOKER3`, is a calculated variable which categorizes subjects by their smoking status:

- 1 = Current smoker who smokes daily
- 2 = Current smoker but not every day
- 3 = Former smoker
- 4 = Never smoked
- 9 = Don't Know / Refused / Missing

We'll reclassify this as a factor with appropriate labels and NAs.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(smoker = fct_recode(factor(`_SMOKER3`),
                                "Current_daily" = "1",
                                "Current_not_daily" = "2",
                                "Former" = "3",
                                "Never" = "4",
                                NULL = "9"))

smart_ohio_raw %>% count(`_SMOKER3`, smoker)
```

```
# A tibble: 5 x 3
  `_SMOKER3` smoker                n
       <dbl> <fct>             <int>
1          1 Current_daily       990
2          2 Current_not_daily   300
3          3 Former             1999
4          4 Never              3881
5          9 <NA>                242
```

2.5.12 E-Cigarettes (2 items)

2.5.12.1 ECIGARET and its cleanup to ecig_ever

ECIGARET, the Ever used an e-cigarette variable, is the response to “Have you ever used an e-cigarette or other electronic vaping product, even just one time, in your entire life?”

- 1 = Yes
- 2 = No
- 7 = Don’t know/Not sure
- 9 = Refused

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(ecig_ever = ECIGARET,
           ecig_ever = replace(ecig_ever, ecig_ever %in% c(7, 9), NA),
           ecig_ever = replace(ecig_ever, ecig_ever == 2, 0))

smart_ohio_raw %>% count(ECIGARET, ecig_ever)
```

```
# A tibble: 5 x 3
  ECIGARET ecig_ever     n
     <dbl>     <dbl> <int>
1        1         1  1354
2        2         0  5799
3        7        NA     9
4        9        NA     3
5       NA        NA   247
```

2.5.12.2 _ECIGSTS and its cleanup to ecigs

_ECIGSTS, is a calculated variable which categorizes subjects by their smoking status:

- 1 = Current and uses daily
- 2 = Current user but not every day
- 3 = Former user
- 4 = Never used e-cigarettes
- 9 = Don’t Know / Refused / Missing

We’ll reclassify this as a factor with appropriate labels and NAs.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(ecigs = fct_recode(factor(`_ECIGSTS`),
                                "Current_daily" = "1",
                                "Current_not_daily" = "2",
                                "Former" = "3",
                                "Never" = "4",
                                NULL = "9"))

smart_ohio_raw %>% count(`_ECIGSTS`, ecigs)
```

```
# A tibble: 5 x 3
  `_ECIGSTS` ecigs                 n
       <dbl> <fct>             <int>
1          1 Current_daily       102
2          2 Current_not_daily   165
3          3 Former             1085
4          4 Never              5799
5          9 <NA>                261
```

2.5.13 Alcohol Consumption (6 items)

2.5.13.1 `ALCDAY5` and its cleanup to `alcdays`

`ALCDAY5`, the Days in past 30 had alcoholic beverage variable, is the response to "During the past 30 days, how many days per week or per month did you have at least one drink of any alcoholic beverage such as beer, wine, a malt beverage or liquor?"

- 101-107 = # of days per week (101 = 1 day per week, 107 = 7 days per week)
- 201-230 = # of days in past 30 days (201 = 1 day in last 30, 230 = 30 days in last 30)
- 777 = Don't know/Not sure
- 888 = No drinks in past 30 days
- 999 = Refused
- BLANK = Not asked or Missing

We're going to convert this to a single numeric value. Answers in days per week (in the past 7 days) will be converted (after rounding) to days in the past 30. This is a little bit of a mess, really, but we can do it.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(alcdays = as.numeric(ALCDAY5)) %>%
    mutate(alcdays = replace(alcdays, alcdays == 888, 0),
           alcdays = replace(alcdays, alcdays %in% c(777, 999), NA)) %>%
    mutate(alcdays = case_when(ALCDAY5 > 199 & ALCDAY5 < 231 ~ ALCDAY5 - 200,
                               ALCDAY5 > 100 & ALCDAY5 < 108 ~ round((ALCDAY5 - 100)*30
                               TRUE ~ alcdays))

smart_ohio_raw %>% count(ALCDAY5, alcdays)
```

```
# A tibble: 39 x 3
   ALCDAY5 alcdays     n
     <dbl>   <dbl> <int>
 1     101       4   263
 2     102       9   197
 3     103      13   142
 4     104      17    76
 5     105      21    53
 6     106      26    18
 7     107      30   114
 8     201       1   621
 9     202       2   448
10     203       3   233
# ... with 29 more rows
```

2.5.13.2 `AVEDRNK2` and its cleanup to `avgdrinks`

`AVEDRNK2`, the Avg alcoholic drinks per day in past 30 variable, is the response to "One drink is equivalent to a 12-ounce beer, a 5-ounce glass of wine, or a drink with one shot of liquor. During the past 30 days, on the days when you drank, about how many drinks did you drink on the average? (A 40 ounce beer would count as 3 drinks, or a cocktail drink with 2 shots would count as 2 drinks.)"

- 1-76 = # of drinks per day
- 77 = Don't know/Not sure
- 99 = Refused
- BLANK = Not asked or Missing (always happens when ALCDAY5 = 777, 888 or 999)

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(avgdrinks = AVEDRNK2,
           avgdrinks = replace(avgdrinks, avgdrinks > 76, NA))

smart_ohio_raw %>% count(AVEDRNK2, avgdrinks) %>% tail()
```

```
# A tibble: 6 x 3
  AVEDRNK2 avgdrinks     n
     <dbl>     <dbl> <int>
1       42        42     1
2       60        60     2
3       76        76     1
4       77        NA    46
5       99        NA     5
6       NA        NA  3876
```

2.5.13.3 `MAXDRNKS` and its cleanup to `maxdrinks`

`MAXDRINKS`, the most drinks on a single occasion in the past 30 days variable, is the response to "During the past 30 days, what is the largest number of drinks you had on any occasion?"

- 1-76 = # of drinks
- 77 = Don't know/Not sure
- 99 = Refused
- BLANK = Not asked or Missing (always happens when ALCDAY5 = 777, 888 or 999)

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(maxdrinks = MAXDRNKS,
           maxdrinks = replace(maxdrinks, maxdrinks > 76, NA))

smart_ohio_raw %>% count(MAXDRNKS, maxdrinks) %>% tail()
```

```
# A tibble: 6 x 3
  MAXDRNKS maxdrinks     n
     <dbl>     <dbl> <int>
1       42        42     1
2       48        48     1
3       76        76     2
4       77        NA    94
5       99        NA    11
6       NA        NA  3899
```

2.5.13.4 `_RFBING5` and its cleanup to `binge`

`_RFBING5` identifies binge drinkers (males having five or more drinks on one occasion, females having four or more drinks on one occasion in the past 30 days)

The values are

- 1 = No
- 2 = Yes
- 9 = Don't Know / Refused / Missing

People who reported no `alcdays` are reported here as "No," so we'll adjust this into an indicator variable, and create the necessary NAs.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(binge = `_RFBING5` - 1,
           binge = replace(binge, binge > 1, NA))

smart_ohio_raw %>% count(`_RFBING5`, binge)
```

```
# A tibble: 3 x 3
  `_RFBING5` binge     n
       <dbl> <dbl> <int>
1          1     0  6035
2          2     1  1000
3          9    NA   377
```

2.5.13.5 `_DRNKWEK` and its cleanup to `drinks_wk`

`_DRNKWEK` provides the computed number of alcoholic drinks per week, with two implied decimal places. The code 99900 is used for "Don't know / Not sure / Refused / Missing" so we'll fix that, and also divide by 100 to get an average with a decimal point.

Note: We're also going to treat all results of 100 or more drinks per week as incorrect, and thus indicate them as missing data here.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(drinks_wk = `_DRNKWEK` / 100,
           drinks_wk = replace(drinks_wk, drinks_wk > 99, NA))

smart_ohio_raw %>% count(`_DRNKWEK`, drinks_wk) %>% tail(12)
```

```
# A tibble: 12 x 3
   `_DRNKWEK` drinks_wk     n
        <dbl>     <dbl> <int>
 1       9333      93.3     2
 2      10000      NA       1
 3      10500      NA       2
 4      11667      NA       1
 5      14000      NA       2
 6      16800      NA       2
 7      17500      NA       1
 8      18200      NA       1
 9      28000      NA       1
10      29400      NA       1
11      53200      NA       1
12      99900      NA     379
```

2.5.13.6 `_RFDRHV5` and its cleanup to `drink_heavy`

`_RFDRHV5` identifies heavy drinkers (males having 14 or more drinks per week, females having 7 or more drinks per week)

The values are

- 1 = No
- 2 = Yes
- 9 = Don't Know / Refused / Missing

People who reported no `alcdays` are reported here as "No," so we'll adjust this into an indicator variable, and create the necessary NAs.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(drink_heavy = `_RFDRHV5` - 1,
           drink_heavy = replace(drink_heavy, drink_heavy > 1, NA))

smart_ohio_raw %>% count(`_RFDRHV5`, drink_heavy)
```

```
# A tibble: 3 x 3
  `_RFDRHV5` drink_heavy     n
       <dbl>       <dbl> <int>
1          1           0  6607
2          2           1   426
3          9          NA   379
```

2.5.14 Fruits and Vegetables (8 items)

2.5.14.1 `_FRUTSU1` and its cleanup to `fruit_day`

`_FRUTSU1` provides the computed number of fruit servings consumed per day, with two implied decimal places. We'll divide by 100 to insert the decimal point.

Note: We're also going to treat all results exceeding 16 servings per day as implausible, and thus indicate them as missing data here, following some CDC procedures.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(fruit_day = `_FRUTSU1` / 100,
           fruit_day = replace(fruit_day, fruit_day > 16, NA))

smart_ohio_raw %>% count(`_FRUTSU1`, fruit_day) %>% tail()
```

```
# A tibble: 6 x 3
  `_FRUTSU1` fruit_day     n
       <dbl>     <dbl> <int>
1        913      9.13     1
2       1000     10        4
3       1400     14        1
4       3000     NA        1
5       7600     NA        1
6         NA     NA      555
```

2.5.14.2 `_VEGESU1` and its cleanup to `veg_day`

`_VEGESU1` provides the computed number of vegetable servings consumed per day, with two implied decimal places. We'll divide by 100 to insert the decimal point.

Note: We're also going to treat all results exceeding 23 servings per day as implausible, and thus indicate them as missing data here, following some CDC procedures.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(veg_day = `_VEGESU1` / 100,
           veg_day = replace(veg_day, veg_day > 23, NA))

smart_ohio_raw %>% count(`_VEGESU1`, veg_day) %>% tail()
```

```
# A tibble: 6 x 3
  `_VEGESU1` veg_day     n
       <dbl>   <dbl> <int>
1       1414    14.1     1
2       1603    16.0     1
3       1891    18.9     1
4       2167    21.7     1
5       3150    NA       1
6         NA    NA     666
```

2.5.14.3 `FTJUDA2_` and its cleanup to `eat_juice`

`FTJUDA2_` provides the servings of fruit juice consumed per day, with two implied decimal places. We'll divide by 100 to insert the decimal point.

Note: We're also going to treat all results exceeding 16 servings per day as implausible, and thus indicate them as missing data here.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(eat_juice = `FTJUDA2_` / 100,
           eat_juice = replace(eat_juice, eat_juice > 16, NA))

smart_ohio_raw %>% count(`FTJUDA2_`, eat_juice) %>% tail()
```

```
# A tibble: 6 x 3
  FTJUDA2_ eat_juice     n
     <dbl>     <dbl> <int>
1      500         5     6
2      600         6     1
3      700         7     1
4     1200        12     1
```

```
5     7500         NA     1
6       NA         NA   469
```

2.5.14.4 `FRUTDA2_` and its cleanup to `eat_fruit`

`FRUTDA2_` provides the servings of fruit consumed per day, with two implied decimal places. We'll divide by 100 to insert the decimal point.

Note: We're also going to treat all results exceeding 16 servings per day as implausible, and thus indicate them as missing data here.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(eat_fruit = `FRUTDA2_` / 100,
           eat_fruit = replace(eat_fruit, eat_fruit > 16, NA))

smart_ohio_raw %>% count(`FRUTDA2_`, eat_fruit) %>% tail()
```

```
# A tibble: 6 x 3
  FRUTDA2_ eat_fruit     n
     <dbl>     <dbl> <int>
1      700         7     5
2      800         8     3
3      900         9     1
4     1000        10     1
5     3000        NA     1
6       NA        NA   456
```

2.5.14.5 `GRENDA1_` and its cleanup to `eat_greenveg`

`GRENDA1_` provides the servings of dark green vegetables consumed per day, with two implied decimal places. We'll divide by 100 to insert the decimal point.

Note: We're also going to treat all results exceeding 16 servings per day as implausible, and thus indicate them as missing data here.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(eat_greenveg = `GRENDA1_` / 100,
           eat_greenveg = replace(eat_greenveg, eat_greenveg > 16, NA))

smart_ohio_raw %>% count(`GRENDA1_`, eat_greenveg) %>% tail()
```

```
# A tibble: 6 x 3
  GRENDA1_ eat_greenveg     n
     <dbl>        <dbl> <int>
1      700         7        4
2      786         7.86     1
3      800         8        2
```

```
4      2000         NA        1
5      3000         NA        1
6        NA         NA      447
```

2.5.14.6 FRNCHDA_ and its cleanup to eat_fries

FRNCHDA_ provides the servings of french fries consumed per day, with two implied decimal places. We'll divide by 100 to insert the decimal point.

Note: We're also going to treat all results exceeding 16 servings per day as implausible, and thus indicate them as missing data here.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(eat_fries = `FRNCHDA_` / 100,
           eat_fries = replace(eat_fries, eat_fries > 16, NA))

smart_ohio_raw %>% count(`FRNCHDA_`, eat_fries) %>% tail()
```

```
# A tibble: 6 x 3
  FRNCHDA_ eat_fries     n
     <dbl>     <dbl> <int>
1      300      3        9
2      314      3.14     1
3      400      4        3
4      500      5        1
5      700      7        1
6       NA     NA      453
```

2.5.14.7 POTADA1_ and its cleanup to eat_potato

POTADA1_ provides the servings of potatoes consumed per day, with two implied decimal places. We'll divide by 100 to insert the decimal point.

Note: We're also going to treat all results exceeding 16 servings per day as implausible, and thus indicate them as missing data here.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(eat_potato = `POTADA1_` / 100,
           eat_potato = replace(eat_potato, eat_potato > 16, NA))

smart_ohio_raw %>% count(`POTADA1_`, eat_potato) %>% tail()
```

```
# A tibble: 6 x 3
  POTADA1_ eat_potato     n
     <dbl>      <dbl> <int>
1      314       3.14     1
2      329       3.29     1
```

```
3      400       4        3
4      471       4.71     1
5      700       7        1
6       NA      NA      501
```

2.5.14.8 `VEGEDA2_` and its cleanup to `eat_otherveg`

`VEGEDA2_` provides the servings of other vegetables consumed per day, with two implied decimal places. We'll divide by 100 to insert the decimal point.

Note: We're also going to treat all results exceeding 16 servings per day as implausible, and thus indicate them as missing data here.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(eat_otherveg = `VEGEDA2_` / 100,
           eat_otherveg = replace(eat_otherveg, eat_otherveg > 16, NA))

smart_ohio_raw %>% count(`VEGEDA2_`, eat_otherveg) %>% tail()
```

```
# A tibble: 6 x 3
  VEGEDA2_ eat_otherveg     n
     <dbl>        <dbl> <int>
1      600            6     3
2      700            7    11
3      800            8     1
4     1000           10     2
5     1100           11     1
6       NA           NA   509
```

2.5.15 Exercise and Physical Activity (8 items)

2.5.15.1 `_TOTINDA` and its cleanup to `exerany`

`_TOTINDA`, the Exercise in Past 30 Days variable, is the response to "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
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

This is just like `HLTHPLAN`.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(exerany = `_TOTINDA`,
           exerany = replace(exerany, exerany %in% c(7, 9), NA),
           exerany = replace(exerany, exerany == 2, 0))

smart_ohio_raw %>% count(`_TOTINDA`, exerany)
```

```
# A tibble: 3 x 3
  `_TOTINDA` exerany     n
       <dbl>   <dbl> <int>
1          1       1  4828
2          2       0  2137
3          9      NA   447
```

2.5.15.2 _PACAT1 and its cleanup to activity

_PACAT1 contains physical activity categories, estimated from responses to the BRFSS. The categories are:

- 1 = Highly Active
- 2 = Active
- 3 = Insufficiently Active
- 4 = Inactive
- 9 = Don't Know / Not Sure / Refused / Missing

So we'll create a factor.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(activity = factor(`_PACAT1`),
           activity = fct_recode(activity,
                                 "Highly_Active" = "1",
                                 "Active" = "2",
                                 "Insufficiently_Active" = "3",
                                 "Inactive" = "4",
                                 NULL = "9"))

smart_ohio_raw %>% count(`_PACAT1`, activity)
```

```
# A tibble: 5 x 3
  `_PACAT1` activity                  n
      <dbl> <fct>                 <int>
1         1 Highly_Active          2053
2         2 Active                 1132
3         3 Insufficiently_Active  1293
4         4 Inactive               2211
5         9 <NA>                    723
```

2.5.15.3 `_PAINDX1` and its cleanup to `rec_aerobic`

`_PAINDX1` indicates whether the respondent's stated levels of physical activity meet recommendations for aerobic activity. The responses are:

- 1 = Yes
- 2 = No
- 9 = Don't know/Not sure/Refused/Missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(rec_aerobic = 2 - `_PAINDX1`,
           rec_aerobic = replace(rec_aerobic, rec_aerobic < 0, NA))

smart_ohio_raw %>% count(`_PAINDX1`, rec_aerobic)
```

```
# A tibble: 3 x 3
  `_PAINDX1` rec_aerobic     n
       <dbl>       <dbl> <int>
1          1           1  3228
2          2           0  3504
3          9          NA   680
```

2.5.15.4 `_PASTRNG` and its cleanup to `rec_strength`

`_PASTRNG` indicates whether the respondent's stated levels of physical activity meet recommendations for strength-building activity. The responses are:

- 1 = Yes
- 2 = No
- 9 = Don't know/Not sure/Refused/Missing

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(rec_strength = 2 - `_PASTRNG`,
           rec_strength = replace(rec_strength, rec_strength < 0, NA))

smart_ohio_raw %>% count(`_PASTRNG`, rec_strength)
```

```
# A tibble: 3 x 3
  `_PASTRNG` rec_strength     n
       <dbl>        <dbl> <int>
1          1            1  1852
2          2            0  5004
3          9           NA   556
```

2.5.15.5 `EXRACT11` and its cleanup to `exer1_type`

Respondents are asked "What type of physical activity or exercise did you spend the most time doing during the past month?" and these responses are gathered into a set of 76 named categories, including an "other" category. Codes 77 (Don't Know / Not Sure) and 99 (Refused) are dropped into NA in my code below, and Code 98 ("Other type of activity") remains. Then I went through the tedious work of converting the factor levels from numbers to names, following the value labels provided by BRFSS.

```r
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(exer1_type = factor(EXRACT11),
           exer1_type = fct_recode(
               exer1_type,
               "Active Gaming Devices" = "1",
               "Aerobics video or class" = "2",
               "Backpacking" = "3",
               "Badminton" = "4",
               "Basketball" = "5",
               "Bicycling machine" = "6",
               "Bicycling" = "7",
               "Boating" = "8",
               "Bowling" = "9",
               "Boxing" = "10",
               "Calisthenics" = "11",
               "Canoeing" = "12",
               "Carpentry" = "13",
               "Dancing" = "14",
               "Elliptical machine" = "15",
               "Fishing" = "16",
               "Frisbee" = "17",
               "Gardening" = "18",
               "Golf with cart" = "19",
               "Golf without cart" = "20",
               "Handball" = "21",
               "Hiking" = "22",
               "Hockey" = "23",
               "Horseback riding" = "24",
               "Hunting large game" = "25",
               "Hunting small game" = "26",
               "Inline skating" = "27",
               "Jogging" = "28",
               "Lacrosse" = "29",
               "Mountain climbing" = "30",
               "Mowing lawn" = "31",
               "Paddleball" = "32",
               "Painting house" = "33",
               "Pilates" = "34",
               "Racquetball" = "35",
               "Raking lawn" = "36",
               "Running" = "37",
               "Rock climbing" = "38",
               "Rope skipping" = "39",
               "Rowing machine" = "40",
               "Rugby" = "41",
               "Scuba diving" = "42",
               "Skateboarding" = "43",
               "Skating" = "44",
               "Sledding" = "45",
               "Snorkeling" = "46",
               "Snow blowing" = "47",
               "Snow shoveling" = "48",
               "Snow skiing" = "49",
               "Snowshoeing" = "50",
               "Soccer" = "51",
               "Softball/Baseball" = "52",
               "Squash" = "53",
               "Stair Climbing" = "54",
               "Stream fishing" = "55",
               "Surfing" = "56",
               "Swimming" = "57",
               "Swimming in laps" = "58",
               "Table tennis" = "59",
               "Tai Chi" = "60",
               "Tennis" = "61",
               "Touch football" = "62",
               "Volleyball" = "63",
               "Walking" = "64",
               "Waterskiing" = "66",
               "Weight lifting" = "67",
               "Wrestling" = "68",
               "Yoga" = "69",
               "Child Care" = "71",
               "Farm Work" = "72",
               "Household Activities" = "73",
               "Martial Arts" = "74",
               "Upper Body Cycle" = "75",
               "Yard Work" = "76",
               "Other Activities" = "98",
               NULL = "77",
               NULL = "99")
```

```
    )
```

```
Warning: Problem with `mutate()` input `exer1_type`.
i Unknown levels in `f`: 3, 17, 21, 32, 36, 41, 42, 45, 47, 53, 55, 56, 59
i Input `exer1_type` is `fct_recode(...)`.
```

The warning generated here is caused by the fact that some of the available types of exercise were not mentioned by people in our sample. Looking at the last few results, we can see how many people fell into several categories.

```
smart_ohio_raw %>% count(EXRACT11, exer1_type) %>% tail()
```

```
# A tibble: 6 x 3
  EXRACT11 exer1_type           n
     <dbl> <fct>            <int>
1       75 Upper Body Cycle     6
2       76 Yard Work           78
3       77 <NA>                10
4       98 Other Activities   276
5       99 <NA>                 4
6       NA <NA>              2588
```

The most common activities are:

```
smart_ohio_raw %>% count(exer1_type, sort = TRUE) %>% head(10)
```

```
# A tibble: 10 x 2
   exer1_type                 n
   <fct>                  <int>
 1 Walking                 2605
 2 <NA>                    2602
 3 Running                  324
 4 Other Activities         276
 5 Gardening                242
 6 Weight lifting           189
 7 Aerobics video or class  103
 8 Bicycling machine        103
 9 Bicycling                 96
10 Golf with cart            90
```

2.5.15.6 `EXRACT21` and its cleanup to `exer2_type`

As a follow-up, respondents are asked "What other type of physical activity gave you the next most exercise during the past month?" and these responses are also gathered into the same set of 76 named categories, including an "other" category, but now also adding a "No Other Activity" category (code 88). Codes 77 (Don't Know / Not Sure) and 99 (Refused) are dropped into NA in my code below, and

Code 98 (“Other type of activity”) remains. Then I went through the tedious work of converting the factor levels from numbers to names, following the value labels provided by BRFSS. I’m sure there’s a better way to do this.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(exer2_type = factor(EXRACT21),
           exer2_type = fct_recode(
               exer2_type,
               "Active Gaming Devices" = "1",
               "Aerobics video or class" = "2",
               "Backpacking" = "3",
               "Badminton" = "4",
               "Basketball" = "5",
               "Bicycling machine" = "6",
               "Bicycling" = "7",
               "Boating" = "8",
               "Bowling" = "9",
               "Boxing" = "10",
               "Calisthenics" = "11",
               "Canoeing" = "12",
               "Carpentry" = "13",
               "Dancing" = "14",
               "Elliptical machine" = "15",
               "Fishing" = "16",
               "Frisbee" = "17",
               "Gardening" = "18",
               "Golf with cart" = "19",
               "Golf without cart" = "20",
               "Handball" = "21",
               "Hiking" = "22",
               "Hockey" = "23",
               "Horseback riding" = "24",
               "Hunting large game" = "25",
               "Hunting small game" = "26",
               "Inline skating" = "27",
               "Jogging" = "28",
               "Lacrosse" = "29",
               "Mountain climbing" = "30",
               "Mowing lawn" = "31",
               "Paddleball" = "32",
               "Painting house" = "33",
               "Pilates" = "34",
               "Racquetball" = "35",
               "Raking lawn" = "36",
               "Running" = "37",
               "Rock climbing" = "38",
               "Rope skipping" = "39",
               "Rowing machine" = "40",
               "Rugby" = "41",
               "Scuba diving" = "42",
               "Skateboarding" = "43",
               "Skating" = "44",
               "Sledding" = "45",
               "Snorkeling" = "46",
               "Snow blowing" = "47",
               "Snow shoveling" = "48",
               "Snow skiing" = "49",
               "Snowshoeing" = "50",
               "Soccer" = "51",
               "Softball/Baseball" = "52",
               "Squash" = "53",
               "Stair Climbing" = "54",
               "Stream fishing" = "55",
               "Surfing" = "56",
               "Swimming" = "57",
               "Swimming in laps" = "58",
               "Table tennis" = "59",
               "Tai Chi" = "60",
               "Tennis" = "61",
               "Touch football" = "62",
               "Volleyball" = "63",
               "Walking" = "64",
               "Waterskiing" = "66",
               "Weight lifting" = "67",
               "Wrestling" = "68",
               "Yoga" = "69",
               "Child Care" = "71",
               "Farm Work" = "72",
               "Household Activities" = "73",
               "Martial Arts" = "74",
               "Upper Body Cycle" = "75",
               "Yard Work" = "76",
               "No Other Activity" = "88",
               "Other Activities" = "98",
               NULL = "77",
               NULL = "99")
    )
```

```
Warning: Problem with `mutate()` input `exer2_type`.
i Unknown levels in `f`: 3, 21, 30, 39, 41, 46, 50, 62
i Input `exer2_type` is `fct_recode(...)`.
```

```
smart_ohio_raw %>% count(EXRACT21, exer2_type) %>% tail()
```

```
# A tibble: 6 x 3
  EXRACT21 exer2_type              n
     <dbl> <fct>               <int>
1       76 Yard Work             153
2       77 <NA>                   26
3       88 No Other Activity    1854
4       98 Other Activities      246
5       99 <NA>                   19
6       NA <NA>                 2627
```

The most common activity types in this group are:

```
smart_ohio_raw %>% count(exer2_type, sort = TRUE) %>% head(10)
```

```
# A tibble: 10 x 2
   exer2_type               n
   <fct>                <int>
 1 <NA>                  2672
 2 No Other Activity     1854
 3 Walking                629
 4 Weight lifting         272
 5 Other Activities       246
 6 Gardening              202
 7 Household Activities   169
 8 Yard Work              153
 9 Running                148
10 Bicycling              118
```

2.5.15.7 `_MINAC11` and its cleanup to `exer1_min`

`_MINAC11` is minutes of physical activity per week for the first activity (listed as `exer1_type` above.) Since there are only about 10,080 minutes in a typical week, we'll treat as implausible any values larger than 4200 minutes (which would indicate 70 hours per week.)

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(exer1_min = `_MINAC11`,
           exer1_min = replace(exer1_min, exer1_min > 4200, NA))

smart_ohio_raw %>% count(`_MINAC11`, exer1_min) %>% tail()
```

```
# A tibble: 6 x 3
  `_MINAC11` exer1_min     n
       <dbl>     <dbl> <int>
1       3780      3780     8
```

```
2       3959      3959     1
3       3960      3960     1
4       4193      4193     6
5      27000        NA     1
6         NA        NA  2760
```

2.5.15.8 `_MINAC21` and its cleanup to `exer2_min`

`_MINAC21` is minutes of physical activity per week for the second activity (listed as `exer2_type` above.) Again, we'll treat as implausible any values larger than 4200 minutes (which would indicate 70 hours per week.)

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(exer2_min = `_MINAC21`,
           exer2_min = replace(exer2_min, exer2_min > 4200, NA))

smart_ohio_raw %>% count(`_MINAC21`, exer2_min) %>% tail()
```

```
# A tibble: 6 x 3
  `_MINAC21` exer2_min     n
       <dbl>     <dbl> <int>
1       3360      3360     3
2       3780      3780     7
3       4193      4193     3
4       6120        NA     1
5       8400        NA     1
6         NA        NA  2770
```

2.5.16 Seatbelt Use (1 item)

2.5.16.1 `SEATBELT` and its cleanup to `seatbelt`

This question asks "How often do you use seat belts when you drive or ride in a car?" Possible responses are:

- 1 = Always
- 2 = Nearly always
- 3 = Sometimes
- 4 = Seldom
- 5 = Never
- 7 = Don't know / Not sure
- 8 = Never drive or ride in a car
- 9 = Refused

We'll treat codes 7, 8 and 9 as NA, and turn this into a factor.

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(seatbelt = fct_recode(factor(SEATBELT),
                                 "Always" = "1",
                                 "Nearly_always" = "2",
                                 "Sometimes" = "3",
                                 "Seldom" = "4",
                                 "Never" = "5",
                                 NULL = "7",
                                 NULL = "8",
                                 NULL = "9"))

smart_ohio_raw %>% count(SEATBELT, seatbelt)
```

```
# A tibble: 9 x 3
  SEATBELT seatbelt          n
     <dbl> <fct>         <int>
1        1 Always         6047
2        2 Nearly_always   409
3        3 Sometimes       191
4        4 Seldom           81
5        5 Never           148
6        7 <NA>              7
7        8 <NA>             21
8        9 <NA>              2
9       NA <NA>            506
```

2.5.17 Immunization (3 items)

2.5.17.1 FLUSHOT6 and its cleanup to vax_flu

FLUSHOT6 gives the response to "During the past 12 months, have you had either a flu shot or a flu vaccine that was sprayed in your nose?" The responses are:

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(vax_flu = 2 - FLUSHOT6,
           vax_flu = replace(vax_flu, vax_flu < 0, NA))

smart_ohio_raw %>% count(FLUSHOT6, vax_flu)
```

```
# A tibble: 5 x 3
  FLUSHOT6 vax_flu     n
```

```
     <dbl>    <dbl> <int>
1        1        1  3453
2        2        0  3410
3        7       NA    26
4        9       NA     3
5       NA       NA   520
```

2.5.17.2 PNEUVAC3 and its cleanup to vax_pneumo

PNEUVAC3 gives the response to “A pneumonia shot or pneumococcal vaccine is usually given only once or twice in a person’s lifetime and is different from the flu shot. Have you ever had a pneumonia shot?” The responses are:

- 1 = Yes
- 2 = No
- 7 = Don’t know/Not sure
- 9 = Refused

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(vax_pneumo = 2 - PNEUVAC3,
           vax_pneumo = replace(vax_pneumo, vax_pneumo < 0, NA))

smart_ohio_raw %>% count(PNEUVAC3, vax_pneumo)
```

```
# A tibble: 5 x 3
  PNEUVAC3 vax_pneumo     n
     <dbl>      <dbl> <int>
1        1          1  3112
2        2          0  3262
3        7         NA   509
4        9         NA     3
5       NA         NA   526
```

2.5.17.3 SHINGLE2 and its cleanup to vax_shingles

SHINGLE2 gives the response to “Have you ever had the shingles or zoster vaccine?” The responses are:

- 1 = Yes
- 2 = No
- 7 = Don’t know/Not sure
- 9 = Refused

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(vax_shingles = 2 - SHINGLE2,
           vax_shingles = replace(vax_shingles, vax_shingles < 0, NA))
```

```
smart_ohio_raw %>% count(SHINGLE2, vax_shingles)
```

```
# A tibble: 4 x 3
  SHINGLE2 vax_shingles     n
     <dbl>        <dbl> <int>
1        1            1  1503
2        2            0  2979
3        7           NA    78
4       NA           NA  2852
```

2.5.18 HIV/AIDS (2 items)

2.5.18.1 `HIVTST6` and its cleanup to `hiv_test`

`HIVTST6` gives the response to "Have you ever been tested for HIV? Do not count tests you may have had as part of a blood donation. Include testing fluid from your mouth." The responses are:

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(hiv_test = 2 - HIVTST6,
           hiv_test = replace(hiv_test, hiv_test < 0, NA))

smart_ohio_raw %>% count(HIVTST6, hiv_test)
```

```
# A tibble: 5 x 3
  HIVTST6 hiv_test     n
    <dbl>    <dbl> <int>
1       1        1  2017
2       2        0  4565
3       7       NA   260
4       9       NA    14
5      NA       NA   556
```

2.5.18.2 `HIVRISK5` and its cleanup to `hiv_risk`

`HIVRISK5` gives the response to "I am going to read you a list. When I am done, please tell me if any of the situations apply to you. You do not need to tell me which one. You have injected any drug other than those prescribed for you in the past year. You have been treated for a sexually transmitted disease or STD in the past year. You have given or received money or drugs in exchange for sex in the past year." The responses are:

- 1 = Yes
- 2 = No
- 7 = Don't know/Not sure
- 9 = Refused

```
smart_ohio_raw <- smart_ohio_raw %>%
    mutate(hiv_risk = 2 - HIVRISK5,
           hiv_risk = replace(hiv_risk, hiv_risk < 0, NA))

smart_ohio_raw %>% count(HIVRISK5, hiv_risk)
```

```
# A tibble: 5 x 3
  HIVRISK5 hiv_risk     n
     <dbl>    <dbl> <int>
1        1        1   277
2        2        0  6537
3        7       NA     2
4        9       NA    17
5       NA       NA   579
```

2.6 Imputing Age and Income as Quantitative from Thin Air

This section is purely for teaching purposes. I would never use the variables created in this section for research work.

2.6.1 age_imp: Imputing Age Data

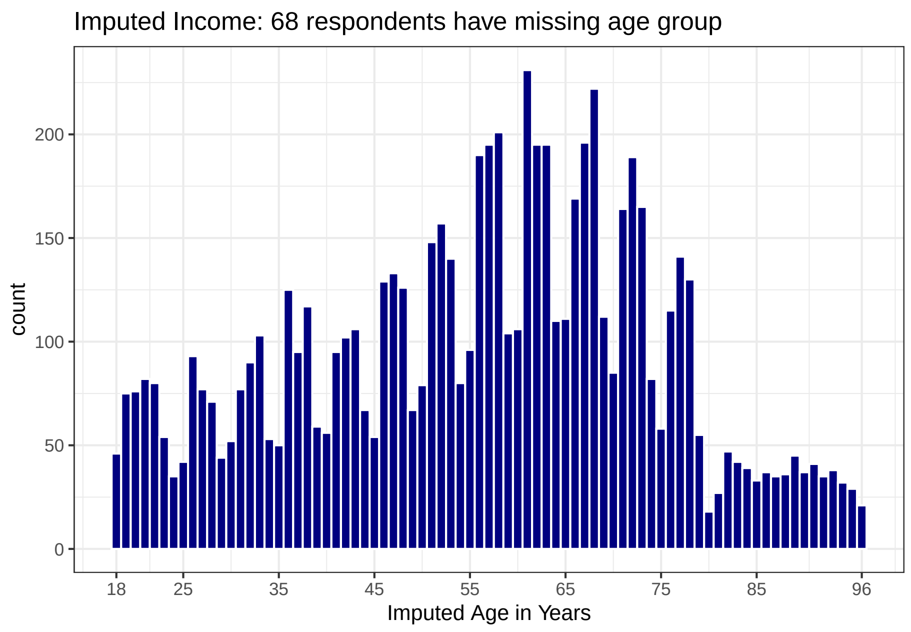
I want a quantitative age variable, so I'm going to create an imputed `age_imp` value for each subject based on their `agegroup`. For each age group, I will assume that each of the ages represented by a value in that age group will be equally likely, and will draw from the relevant uniform distribution to impute age.

```
set.seed(2020432002)

smart_ohio_raw <- smart_ohio_raw %>%
    mutate(age_low = as.numeric(str_sub(as.character(agegroup), 1, 2))) %>%
    mutate(age_high = as.numeric(str_sub(as.character(agegroup), 4, 5))) %>%
    rowwise() %>%
    mutate(age_imp = ifelse(!is.na(agegroup),
                            round(runif(1, min = age_low, max = age_high),0),
                             NA))

smart_ohio_raw %>% count(agegroup, age_imp) #%>% tail()
```

```
# A tibble: 80 x 3
# Rowwise:
   agegroup age_imp     n
   <fct>      <dbl> <int>
 1 18-24         18    46
 2 18-24         19    75
 3 18-24         20    76
 4 18-24         21    82
 5 18-24         22    80
 6 18-24         23    54
 7 18-24         24    35
 8 25-29         25    42
 9 25-29         26    93
10 25-29         27    77
# ... with 70 more rows
```

Here is a histogram of the `age_imp` variable.

```
ggplot(smart_ohio_raw, aes(x = age_imp)) +
    geom_histogram(fill = "navy", col = "white",
                   binwidth = 1) +
    scale_x_continuous(breaks = c(18, 25, 35, 45, 55, 65, 75, 85, 96)) +
    labs(x = "Imputed Age in Years",
         title = paste0("Imputed Income: ",
                        sum(is.na(smart_ohio_raw$age_imp)),
                        " respondents have missing age group"))
```



2.6.2 `inc_imp`: Imputing Income Data

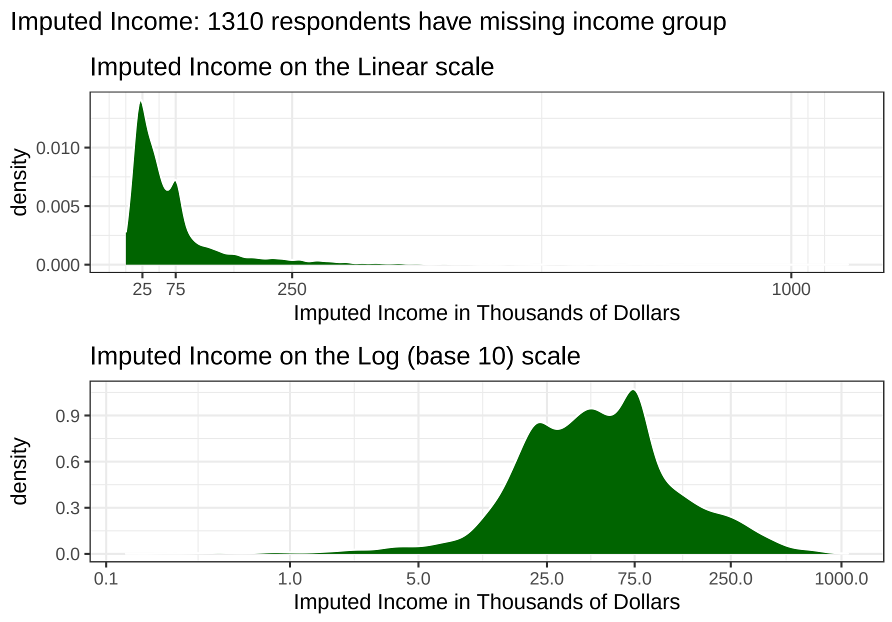
I want a quantitative income variable, so I'm going to create an imputed `inc_imp` value for each subject based on their `incomegroup`. For most income groups, I will assume that each of the incomes represented by a value in that income group will be equally likely, and will draw from the relevant uniform distribution to impute income. The exception is the highest income group, where I will impute a value drawn from a distribution that places all values at $75,000 or more, but has a substantial right skew and long tail.

```r
set.seed(2020432001)

smart_ohio_raw <- smart_ohio_raw %>%
    mutate(inc_imp = case_when(
        incomegroup == "0-9K" ~ round(runif(1, min = 100, max = 9999)),
        incomegroup == "10-14K" ~ round(runif(1, min = 10000, max = 14999)),
        incomegroup == "15-19K" ~ round(runif(1, min = 15000, max = 19999)),
        incomegroup == "20-24K" ~ round(runif(1, min = 20000, max = 24999)),
        incomegroup == "25-34K" ~ round(runif(1, min = 25000, max = 34999)),
        incomegroup == "35-49K" ~ round(runif(1, min = 35000, max = 49999)),
        incomegroup == "50-74K" ~ round(runif(1, min = 50000, max = 74999)),
        incomegroup == "75K+" ~ round((rnorm(n = 1, mean = 0, sd = 300)^2) + 74999)))

smart_ohio_raw %>% count(incomegroup, inc_imp) %>% tail()
```

```
# A tibble: 6 x 3
# Rowwise:
  incomegroup inc_imp     n
  <fct>         <dbl> <int>
1 75K+         774009     1
2 75K+         798174     1
3 75K+         806161     1
4 75K+         847758     1
5 75K+        1085111     1
6 <NA>             NA  1310
```

Here are density plots of the `inc_imp` variable. The top picture shows the results on a linear scale, and the bottom shows them on a log (base 10) scale.

```
p1 <- ggplot(smart_ohio_raw, aes(x = inc_imp/1000)) +
    geom_density(fill = "darkgreen", col = "white") +
    labs(x = "Imputed Income in Thousands of Dollars",
         title = "Imputed Income on the Linear scale") +
    scale_x_continuous(breaks = c(25, 75, 250, 1000))

p2 <- ggplot(smart_ohio_raw, aes(x = inc_imp/1000)) +
    geom_density(fill = "darkgreen", col = "white") +
    labs(x = "Imputed Income in Thousands of Dollars",
         title = "Imputed Income on the Log (base 10) scale") +
    scale_x_log10(breaks = c(0.1, 1, 5, 25, 75, 250, 1000))

p1 / p2 +
    plot_annotation(title =
                        paste0("Imputed Income: ", sum(is.na(smart_ohio_raw$inc_imp)),
```



2.7 Clean Data in the State of Ohio

There are six MMSAs associated with the state of Ohio. We're going to create a `smart_ohio` that includes each of them. First, I'll ungroup the data that I created earlier, so I get a clean tibble.

```
smart_ohio_raw <- smart_ohio_raw %>% ungroup()
```

Next, I'll select the variables I want to retain (they are the ones I created, plus `SEQNO`.)

```
smart_ohio <- smart_ohio_raw %>%
    select(SEQNO, mmsa, mmsa_code, mmsa_name, mmsa_wt, completed,
           landline, hhadults,
           genhealth, physhealth, menthealth, poorhealth,
           agegroup, age_imp, race, hispanic, race_eth,
           female, marital, kids, educgroup, home_own,
           veteran, employment, incomegroup, inc_imp,
           cell_own, internet30,
           weight_kg, height_m, bmi, bmigroup,
           pregnant, deaf, blind, decide,
           diffwalk, diffdress, diffalone,
           smoke100, smoker, ecig_ever, ecigs,
           healthplan, hasdoc, costprob, t_checkup,
           bp_high, bp_meds,
           t_chol, chol_high, chol_meds,
           asthma, hx_asthma, now_asthma,
           hx_mi, hx_chd, hx_stroke, hx_skinc, hx_otherc,
           hx_copd, hx_depress, hx_kidney,
           hx_diabetes, dm_status, dm_age,
           hx_arthr, arth_lims, arth_work, arth_soc,
           joint_pain, alcdays, avgdrinks, maxdrinks,
           binge, drinks_wk, drink_heavy,
           fruit_day, veg_day, eat_juice, eat_fruit,
           eat_greenveg, eat_fries, eat_potato,
           eat_otherveg, exerany, activity, rec_aerobic,
           rec_strength, exer1_type, exer2_type,
           exer1_min, exer2_min, seatbelt,
           vax_flu, vax_pneumo, vax_shingles,
           hiv_test, hiv_risk)

saveRDS(smart_ohio, "data/smart_ohio.Rds")

write_csv(smart_ohio, "data/smart_ohio.csv")
```

The `smart_ohio` file should contain 99 variables, describing 7412 respondents.

2.8 Clean Cleveland-Elyria Data

2.8.1 Cleveland - Elyria Data

The `mmsa_name` variable is probably the simplest way for us to filter our data down to the MMSA we are interested in. Here, I'm using the `str_detect` function to identify the values of `mmsa_name` that contain the text "Cleveland."

```
smart_cle <- smart_ohio %>%
  filter(str_detect(mmsa_name, 'Cleveland'))

saveRDS(smart_cle, "data/smart_cle.Rds")
```

In the Cleveland-Elyria MSA, we have 1133 observations on the same 99 variables.

We'll build a variety of smaller subsets from these data, eventually.

Chapter 3

Dealing with Missingness: Single Imputation

3.1 Selecting Some Variables from the `smart_cle` data

```r
smart_cle <- readRDS("data/smart_cle.Rds")

smart_cle1 <- smart_cle %>%
  select(SEQNO, physhealth, genhealth, bmi,
         age_imp, female, race_eth, internet30,
         smoke100, activity, drinks_wk, veg_day)
```

The `smart_cle.Rds` data file available on the Data and Code page of our website describes information on 99 variables for 1133 respondents to the BRFSS 2017, who live in the Cleveland-Elyria, OH, Metropolitan Statistical Area. The variables in the `smart_cle1.csv` file are listed below, along with the items that generate these responses.

Variable	Description
`SEQNO`	respondent identification number (all begin with 2016)
`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?
`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
`age_imp`	Age, imputed, in years

Variable	Description
`female`	Sex, 1 = female, 0 = male
`race_eth`	Race and Ethnicity, in five categories
`internet30`	Have you used the internet in the past 30 days? (1 = yes, 0 = no)
`smoke100`	Have you smoked at least 100 cigarettes in your life? (1 = yes, 0 = no)
`activity`	Physical activity (Highly Active, Active, Insufficiently Active, Inactive)
`drinks_wk`	On average, how many drinks of alcohol do you consume in a week?
`veg_day`	How many servings of vegetables do you consume per day, on average?

```
str(smart_cle1)
```

```
tibble [1,133 x 12] (S3: tbl_df/tbl/data.frame)
 $ SEQNO     : num [1:1133] 2.02e+09 2.02e+09 2.02e+09 2.02e+09 2.02e+09 ...
 $ physhealth: num [1:1133] 4 0 0 0 0 2 2 0 0 0 ...
 $ genhealth : Factor w/ 5 levels "1_Excellent",..: 1 1 3 3 3 2 3 2 4 1 ...
 $ bmi       : num [1:1133] NA 23.1 26.9 26.5 24.2 ...
 $ age_imp   : num [1:1133] 51 28 37 36 88 43 23 34 58 54 ...
 $ female    : num [1:1133] 1 1 1 1 0 0 0 0 0 1 ...
 $ race_eth  : Factor w/ 5 levels "White non-Hispanic",..: 1 1 3 1 1 1 1 3 2 1 ...
 $ internet30: num [1:1133] 1 1 0 1 1 1 1 1 1 1 ...
 $ smoke100  : num [1:1133] 1 0 0 1 1 1 0 0 0 1 ...
 $ activity  : Factor w/ 4 levels "Highly_Active",..: 4 4 3 1 1 NA 1 1 1 1 ...
 $ drinks_wk : num [1:1133] 0.7 0 0 4.67 0.93 0 2 0 0 0.47 ...
 $ veg_day   : num [1:1133] NA 3 4.06 2.07 1.31 NA 1.57 0.83 0.49 1.72 ...
```

3.2 `smart_cle1`: Seeing our Missing Data

The `naniar` package provides several useful functions for summarizing missingness in our data set. Like all tidy data sets, our `smart_cle1` tibble contains rows which describe observations, sometimes called *cases*, and also contains columns which describe variables.

Overall, there are 1133 cases, and 1133 observations in our `smart_cle1` tibble.

- We can obtain a count of the number of missing cells in the entire tibble.

```
smart_cle1 %>% n_miss()
```

```
[1] 479
```

- We can use the `miss_var_summary` function to get a sorted table of each variable by number missing.

```
miss_var_summary(smart_cle1) %>% knitr::kable()
```

variable	n_miss	pct_miss
activity	109	9.6204766
veg_day	101	8.9143866
bmi	91	8.0317741
drinks_wk	66	5.8252427
smoke100	40	3.5304501
race_eth	26	2.2947926
physhealth	24	2.1182701
age_imp	11	0.9708738
internet30	7	0.6178288
genhealth	4	0.3530450
SEQNO	0	0.0000000
female	0	0.0000000

- Or we can use the `miss_var_table` function to tabulate the number of variables that have each observed level of missingness.

```
miss_var_table(smart_cle1)
```

```
# A tibble: 11 x 3
   n_miss_in_var n_vars pct_vars
 *         <int>  <int>    <dbl>
 1             0      2    16.7
 2             4      1     8.33
 3             7      1     8.33
 4            11      1     8.33
 5            24      1     8.33
 6            26      1     8.33
 7            40      1     8.33
 8            66      1     8.33
 9            91      1     8.33
10           101      1     8.33
11           109      1     8.33
```

- Or we can get a count for a specific variable, like `activity`:

```
smart_cle1 %>% select(activity) %>% n_miss()
```

```
[1] 109
```

- We can also use `prop_miss_case` or `pct_miss_case` to specify the proportion (or percentage) of missing observations across an entire data set, or within a specific variable.

```
prop_miss_case(smart_cle1)
```

```
[1] 0.2127096
```

```
smart_cle1 %>% select(activity) %>% pct_miss_case(.)
```

```
[1] 9.620477
```

- We can also use `prop_miss_var` or `pct_miss_var` to specify the proportion (or percentage) of variables with missing observations across an entire data set.

```
prop_miss_var(smart_cle1)
```

```
[1] 0.8333333
```

```
pct_miss_var(smart_cle1)
```

```
[1] 83.33333
```

- We use `miss_case_table` to identify the number of missing values for each of the cases (rows) in our tibble.

```
miss_case_table(smart_cle1)
```

```
# A tibble: 7 x 3
  n_miss_in_case n_cases pct_cases
*          <int>   <int>     <dbl>
1              0     892    78.7
2              1     129    11.4
3              2      51     4.50
4              3      22     1.94
5              4      21     1.85
6              5      10     0.883
7              6       8     0.706
```

- Use `miss_case_summary` to specify individual observations and count their missing values.

```
miss_case_summary(smart_cle1)
```

```
# A tibble: 1,133 x 3
    case n_miss pct_miss
   <int>  <int>    <dbl>
 1    17      6     50
 2    42      6     50
 3   254      6     50
 4   425      6     50
 5   521      6     50
 6   729      6     50
 7   757      6     50
 8  1051      6     50
 9    89      5     41.7
10    94      5     41.7
# ... with 1,123 more rows
```

The case numbers identified here are row numbers. Extract the data for case 17, for instance, with the `slice` function.

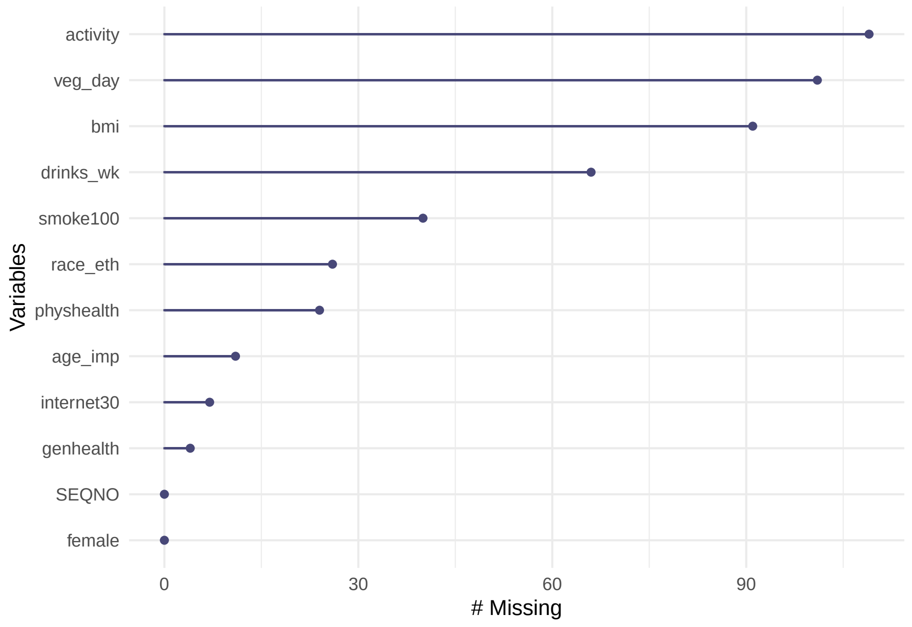
```
smart_cle1 %>% slice(17)
```

```
# A tibble: 1 x 12
   SEQNO physhealth genhealth   bmi age_imp female race_eth internet30 smoke100
   <dbl>      <dbl> <fct>     <dbl>   <dbl>  <dbl> <fct>         <dbl>    <dbl>
1 2.02e9          0 1_Excell~    NA      50      0 White n~         NA       NA
# ... with 3 more variables: activity <fct>, drinks_wk <dbl>, veg_day <dbl>
```

3.2.1 Plotting Missingness

The `gg_miss_var` function plots the number of missing observations in each variable in our data set.

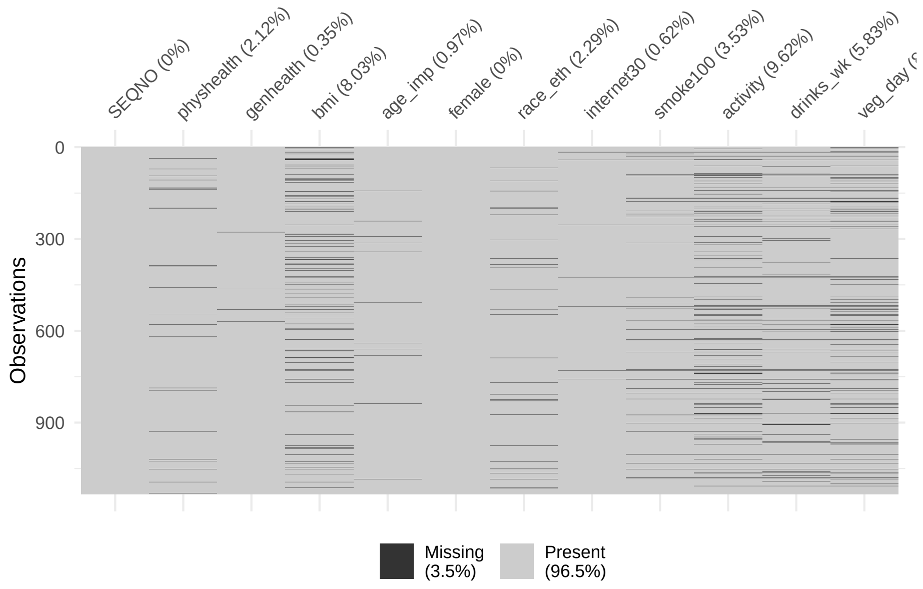
```
gg_miss_var(smart_cle1)
```



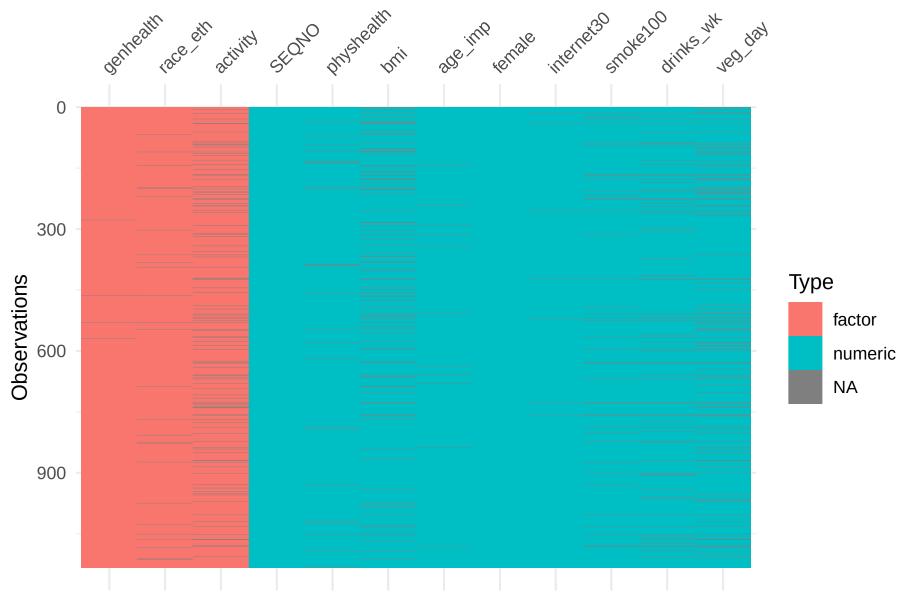
So the most commonly missing variable is `activity` which, as we've seen, has 109 missing values.

To get a general sense of the missingness in our data, we might use either the `vis_dat` or the `vis_miss` function from the `visdat` package.

```
vis_miss(smart_cle1)
```



```
vis_dat(smart_cle1)
```



3.3 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.

3.4 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

3.5 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.

3.6 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.

3.7 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.

3.8 Approach 1: Building a Complete Case Analysis: `smart_cle1_cc`

In the 431 course, we usually dealt with missing data by restricting our analyses to respondents with complete data on all variables. Let's start by doing that here. We'll create a new tibble called `smart_cle1_cc` which includes all respondents with complete data on all of these variables.

```
smart_cle1_cc <- smart_cle1 %>%
  drop_na()

dim(smart_cle1_cc)
```

```
[1] 892  12
```

Our `smart_cle1_cc` tibble now has many fewer observations than its predecessors, but all of the variables in this complete cases tibble have no missing observations.

Data Set	Rows	Columns	Missingness?
`smart_cle`	1133	99	Quite a bit.
`smart_cle1`	1133	12	Quite a bit.
`smart_cle1_cc`	892	12	None.

3.9 Approach 2: Single Imputation to create `smart_cle1_sh`

Next, we'll create a data set which has all of the rows in the original `smart_cle1` tibble, but deals with missingness by imputing (estimating / filling in) new values for each of the missing values. To do this, we'll make heavy use of the `simputation` package in R.

The `simputation` package is designed for single imputation work. Note that we'll eventually adopt a **multiple imputation** strategy in some of our modeling work, and we'll use some specialized tools to facilitate that later.

To begin, we'll create a "shadow" in our tibble to track what we'll need to impute.

```
smart_cle1_sh <- bind_shadow(smart_cle1)

names(smart_cle1_sh)
```

```
 [1] "SEQNO"         "physhealth"    "genhealth"     "bmi"
 [5] "age_imp"       "female"        "race_eth"      "internet30"
 [9] "smoke100"      "activity"      "drinks_wk"     "veg_day"
[13] "SEQNO_NA"      "physhealth_NA" "genhealth_NA"  "bmi_NA"
[17] "age_imp_NA"    "female_NA"     "race_eth_NA"   "internet30_NA"
[21] "smoke100_NA"   "activity_NA"   "drinks_wk_NA"  "veg_day_NA"
```

Note that the `bind_shadow()` function doubles the number of variables in our tibble, specifically by creating a new variable for each that takes the value `!NA` or `NA`. For example, consider

```
smart_cle1_sh %>% count(activity, activity_NA)
```

```
# A tibble: 5 x 3
  activity              activity_NA     n
  <fct>                 <fct>       <int>
```

```
1 Highly_Active         !NA          338
2 Active                !NA          173
3 Insufficiently_Active !NA          201
4 Inactive              !NA          312
5 <NA>                  NA           109
```

The `activity_NA` variable takes the value `!NA` (meaning not missing) when the value of the `activity` variable is known, and takes the value `NA` for observations where the `activity` variable is missing. This background tracking will be helpful to us when we try to assess the impact of imputation on some of our summaries.

3.9.1 What Type of Missingness Do We Have?

There are three types of missingness that we might assume in any given setting: missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR). Together, MCAR and MAR are sometimes called *ignorable* non-response, which essentially means that imputation provides a way to useful estimates. MNAR or missing NOT at random is sometimes called non-ignorable missingness, implying that even high-quality imputation may not be sufficient to provide useful information to us.

Missing Completely at Random means that the missing data points are a random subset of the data. Essentially, there is nothing that makes some data more likely to be missing than others. If the data truly match the standard for MCAR, then a complete-case analysis will be about as good as an analysis after single or multiple imputation.

Missing at Random means that there is a systematic relationship between the observed data and the missingness mechanism. Another way to say this is that the missing value is not related to the reason why it is missing, but is related to the other variables collected in the study. The implication is that the missingness can be accounted for by studying the variables with complete information. Imputation strategies can be very helpful here, incorporating what we know (or think we know) about the relationships between the results that are missing and the results that we see.

- Wikipedia provides a nice example. If men are less likely to fill in a depression survey, but this has nothing to do with their level of depression after accounting for the fact that they are male, then the missingess can be assumed MAR.
- Determining whether missingness is MAR or MNAR can be tricky. We'll spend more time discussing this later.

Missing NOT at Random means that the missing value is related to the reason why it is missing.

- Continuing the Wikipedia example, if men failed to fill in a depression survey because of their level of depression, then this would be MNAR.

- Single imputation is most helpful in the MAR situation, although it is also appropriate when we assume MCAR.
- Multiple imputation will, similarly, be more helpful in MCAR and MAR situations than when data are missing NOT at random.

It's worth noting that many people are unwilling to impute values for outcomes or key predictors in a modeling setting, but are happy to impute for less important covariates. For now, we'll assume MCAR or MAR for all of the missingness in our `smart_cle1` data, which will allow us to adopt a single imputation strategy.

3.9.2 Single imputation into `smart_cle1_sh`

Which variables in `smart_cle1_sh` contain missing data?

```
miss_var_summary(smart_cle1_sh)
```

```
# A tibble: 24 x 3
   variable   n_miss pct_miss
   <chr>       <int>    <dbl>
 1 activity      109    9.62
 2 veg_day       101    8.91
 3 bmi            91    8.03
 4 drinks_wk      66    5.83
 5 smoke100       40    3.53
 6 race_eth       26    2.29
 7 physhealth     24    2.12
 8 age_imp        11    0.971
 9 internet30      7    0.618
10 genhealth       4    0.353
# ... with 14 more rows
```

We will impute these variables using several different strategies, all supported nicely by the `simputation` package.

These include imputation methods based solely on the distribution of the complete cases of the variable being imputed.

- `impute_median`: impute the median value of all non-missing observations into the missing values for the variable
- `impute_rhd`: random "hot deck" imputation involves drawing at random from the complete cases for that variable

Also available are imputation strategies that impute predicted values from models using other variables in the data set besides the one being imputed.

- `impute_pmm`: imputation using predictive mean matching
- `impute_rlm`: imputation using robust linear models
- `impute_cart`: imputation using classification and regression trees

- `impute_knn`: imputation using k-nearest neighbors methods

3.9.3 Imputing Binary Categorical Variables

Here, we'll arbitrarily impute our 1/0 variables as follows:

- For `internet30` we'll use the `impute_rhd` approach to draw a random observation from the existing set of 1s and 0s in the complete `internet30` data.
- For `smoke100` we'll use a method called predictive mean matching (`impute_pmm`) which takes the result from a model based on the (imputed) `internet30` value and whether or not the subject is `female`, and converts it to the nearest value in the observed `smoke100` data. This is a good approach for imputing discrete variables.

These are completely arbitrary choices, for demonstration purposes.

```
set.seed(2020001)
smart_cle1_sh <- smart_cle1_sh %>%
  data.frame() %>%
    impute_rhd(.,
               internet30 ~ 1) %>%
    impute_pmm(., smoke100 ~ internet30 + female) %>%
  tbl_df()
```

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

```
smart_cle1_sh %>% count(smoke100, smoke100_NA)
```

```
# A tibble: 4 x 3
  smoke100 smoke100_NA     n
     <dbl> <fct>       <int>
1        0 !NA           579
2        0 NA             21
3        1 !NA           514
4        1 NA             19
```

```
smart_cle1_sh %>% count(internet30, internet30_NA)
```

```
# A tibble: 4 x 3
  internet30 internet30_NA     n
       <dbl> <fct>         <int>
1          0 !NA             207
2          0 NA                1
3          1 !NA             919
```

```
4           1 NA                    6
```

Other approaches that may be used with 1/0 variables include `impute_knn` and `impute_pmm`.

3.9.4 Imputing Quantitative Variables

We'll demonstrate a different approach for imputing each of the quantitative variables with missing observations. Again, we're making purely arbitrary decisions here about what to include in each imputation. In practical work, we'd want to be a bit more thoughtful about this.

Note that I'm choosing to use `impute_pmm` with the `physhealth` and `age_imp` variables. This is (in part) because I want my imputations to be integers, as the other observations are for those variables. `impute_rhd` would also accomplish this.

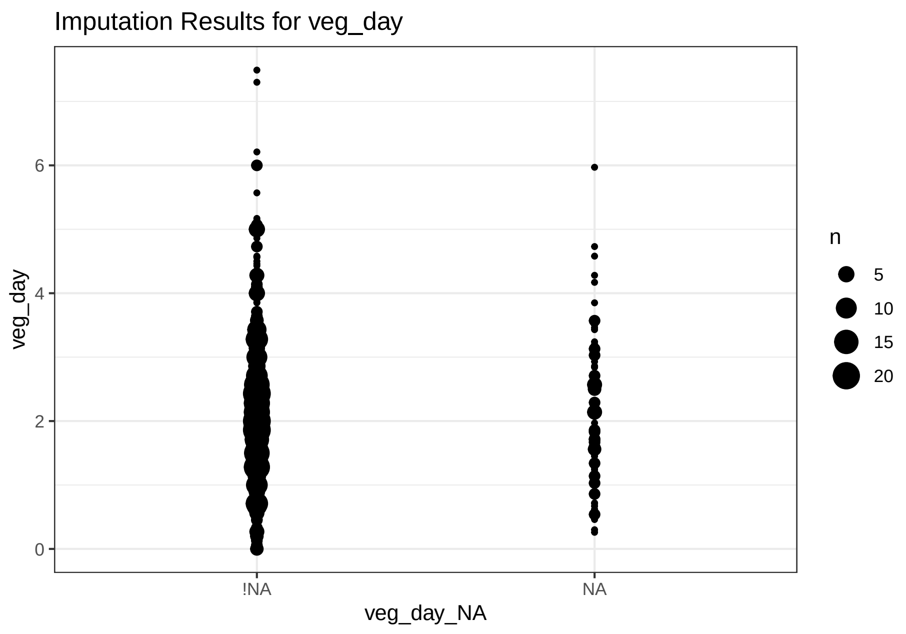
```
set.seed(2020001)
smart_cle1_sh <- smart_cle1_sh %>%
  data.frame() %>%
    impute_rhd(., veg_day ~ 1) %>%
    impute_median(., drinks_wk ~ 1) %>%
    impute_pmm(., physhealth ~
                 drinks_wk + female + smoke100) %>%
    impute_pmm(., age_imp ~ drinks_wk + physhealth) %>%
    impute_rlm(., bmi ~ physhealth + smoke100) %>%
  tbl_df()
```

3.9.5 Imputation Results

Let's plot a few of these results, so we can see what imputation has done to the distribution of these quantities.

1. `veg_day`

```
ggplot(smart_cle1_sh, aes(x = veg_day_NA, y = veg_day)) +
  geom_count() +
  labs(title = "Imputation Results for veg_day")
```



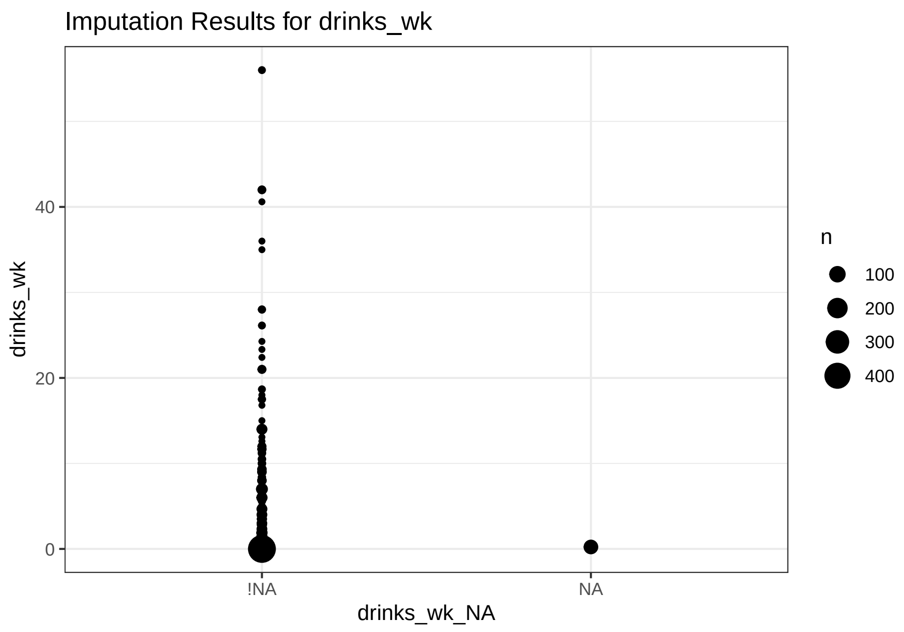
```
smart_cle1_sh %$%
  mosaic::favstats(veg_day ~ veg_day_NA)
```

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

```
  veg_day_NA  min     Q1 median   Q3  max     mean       sd    n missing
1        !NA 0.00 1.2675   1.72 2.42 7.49 1.912548 1.038403 1032       0
2         NA 0.26 1.3400   1.86 2.72 5.97 2.085050 1.062316  101       0
```

2. `drinks_wk` for which we imputed the median value…

```
ggplot(smart_cle1_sh, aes(x = drinks_wk_NA, y = drinks_wk)) +
  geom_count() +
  labs(title = "Imputation Results for drinks_wk")
```



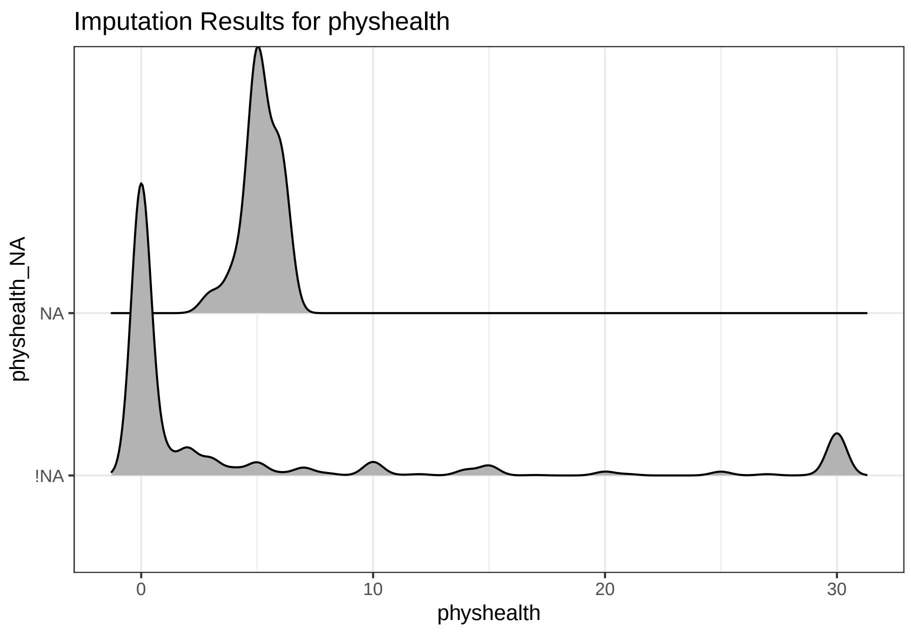
```
smart_cle1_sh %>% filter(drinks_wk_NA == "NA") %>%
  tabyl(drinks_wk)
```

```
 drinks_wk  n percent
      0.23 66       1
```

3. `physhealth`, a count between 0 and 30…

```
ggplot(smart_cle1_sh,
       aes(x = physhealth, y = physhealth_NA)) +
  geom_density_ridges() +
  labs(title = "Imputation Results for physhealth")
```

```
Picking joint bandwidth of 0.426
```

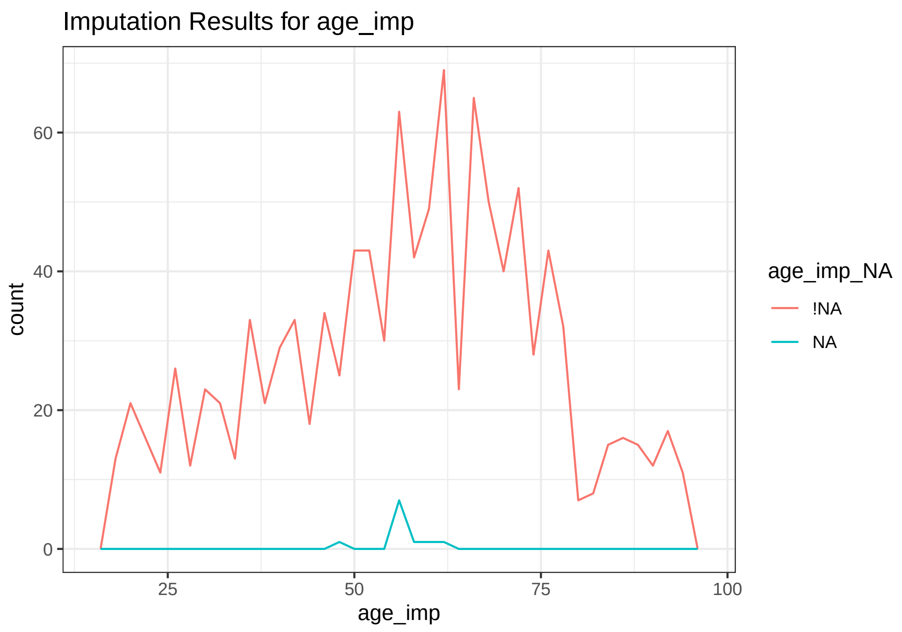


```r
smart_cle1_sh %>% filter(physhealth_NA == "NA") %>%
  tabyl(physhealth)
```

```
 physhealth  n    percent
          3  1 0.04166667
          4  2 0.08333333
          5 13 0.54166667
          6  8 0.33333333
```

4. `age_imp`, in (integer) years

```r
ggplot(smart_cle1_sh,
       aes(x = age_imp, color = age_imp_NA)) +
  geom_freqpoly(binwidth = 2) +
  labs(title = "Imputation Results for age_imp")
```

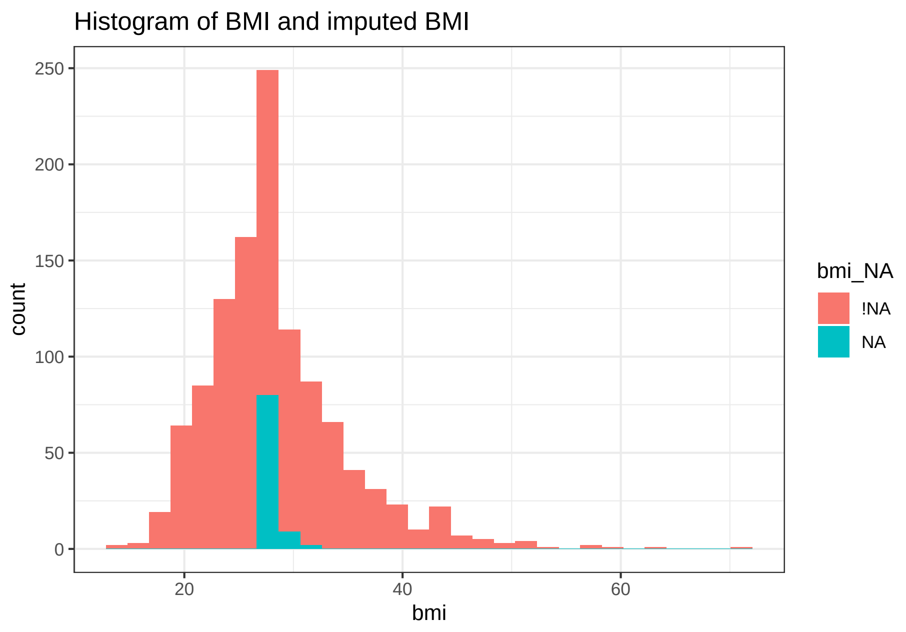


```
smart_cle1_sh %>% filter(age_imp_NA == "NA") %>%
  tabyl(age_imp)
```

```
 age_imp n    percent
      48 1 0.09090909
      57 7 0.63636364
      58 1 0.09090909
      61 1 0.09090909
      63 1 0.09090909
```

5. `bmi` or body mass index

```
ggplot(smart_cle1_sh, aes(x = bmi, fill = bmi_NA)) +
  geom_histogram(bins = 30) +
  labs(title = "Histogram of BMI and imputed BMI")
```



```r
smart_cle1_sh %$% mosaic::favstats(bmi ~ bmi_NA)
```

```
  bmi_NA     min      Q1   median       Q3      max     mean        sd    n
1    !NA 13.3000 24.1100 27.30000 31.68000 70.56000 28.40947 6.6289286 1042
2     NA 27.0693 27.0693 27.50229 27.66574 30.75898 27.66057 0.8964101   91
  missing
1       0
2       0
```

3.9.6 Imputing Multi-Categorical Variables

The three multi-categorical variables we have left to impute are `activity`, `race_eth` and `genhealth`, and each is presented as a factor in R, rather than as a character variable.

We'll arbitrarily decide to impute

- `activity` and `genhealth` with a classification tree using `physhealth`, `bmi` and `smoke100`,
- and then impute `race_eth` with a random draw from the distribution of complete cases.

```r
set.seed(2020001)
smart_cle1_sh <- smart_cle1_sh %>%
  data.frame() %>%
    impute_cart(., activity + genhealth ~
                  physhealth + bmi + smoke100) %>%
    impute_rhd(., race_eth ~ 1) %>%
  tbl_df()
```

Let's check our results.

```
smart_cle1_sh %>% count(activity_NA, activity)
```

```
# A tibble: 6 x 3
  activity_NA activity                 n
  <fct>       <fct>                <int>
1 !NA         Highly_Active          338
2 !NA         Active                 173
3 !NA         Insufficiently_Active  201
4 !NA         Inactive               312
5 NA          Highly_Active           90
6 NA          Inactive                19
```

```
smart_cle1_sh %>% count(race_eth_NA, race_eth)
```

```
# A tibble: 9 x 3
  race_eth_NA race_eth                     n
  <fct>       <fct>                    <int>
1 !NA         White non-Hispanic         805
2 !NA         Black non-Hispanic         222
3 !NA         Other race non-Hispanic     24
4 !NA         Multiracial non-Hispanic    22
5 !NA         Hispanic                    34
6 NA          White non-Hispanic          19
7 NA          Black non-Hispanic           4
8 NA          Multiracial non-Hispanic     2
9 NA          Hispanic                     1
```

```
smart_cle1_sh %>% count(genhealth_NA, genhealth)
```

```
# A tibble: 7 x 3
  genhealth_NA genhealth       n
  <fct>        <fct>       <int>
1 !NA          1_Excellent   164
2 !NA          2_VeryGood    383
3 !NA          3_Good        364
4 !NA          4_Fair        158
5 !NA          5_Poor         60
6 NA           2_VeryGood      3
7 NA           3_Good          1
```

And now, we should have no missing values in the data, at all.

```
miss_case_table(smart_cle1_sh)
```

```
# A tibble: 1 x 3
  n_miss_in_case n_cases pct_cases
*          <int>   <int>     <dbl>
1              0    1133       100
```

3.9.7 Saving the new tibbles

```
saveRDS(smart_cle1_cc, here("data", "smart_cle1_cc.Rds"))
saveRDS(smart_cle1_sh, here("data", "smart_cle1_sh.Rds"))
```

Chapter 4

Summarizing the `smart_cle1` data

In this chapter, we'll work with the two data files we built in the previous chapter.

```
smart_cle1_sh <- readRDS(here("data", "smart_cle1_sh.Rds"))
smart_cle1_cc <- readRDS(here("data", "smart_cle1_sh.Rds"))
```

Those files (`_sh` contains single imputations, and a shadow set of variables which have _NA at the end of their names, while `_cc` contains only the complete cases) describe information on the following variables from the BRFSS 2017, who live in the Cleveland-Elyria, OH, Metropolitan Statistical Area.

Variable	Description
`SEQNO`	respondent identification number (all begin with 2016)
`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?
`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
`age_imp`	Age, imputed, in years
`female`	Sex, 1 = female, 0 = male
`race_eth`	Race and Ethnicity, in five categories
`internet30`	Have you used the internet in the past 30 days? (1 = yes, 0 = no)
`smoke100`	Have you smoked at least 100 cigarettes in your life? (1 = yes, 0 = no)
`activity`	Physical activity (Highly Active, Active, Insufficiently Active, Inactive)

Variable	Description
`drinks_wk`	On average, how many drinks of alcohol do you consume in a week?
`veg_day`	How many servings of vegetables do you consume per day, on average?

4.1 General Approaches to Obtaining Numeric Summaries

4.1.1 `summary` for a data frame

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

```
summary(smart_cle1_cc)
```

```
     SEQNO               physhealth           genhealth          bmi       
 Min.   :2.017e+09   Min.   : 0.000   1_Excellent:164   Min.   :13.30  
 1st Qu.:2.017e+09   1st Qu.: 0.000   2_VeryGood :386   1st Qu.:24.38  
 Median :2.017e+09   Median : 0.000   3_Good     :365   Median :27.31  
 Mean   :2.017e+09   Mean   : 4.681   4_Fair     :158   Mean   :28.35  
 3rd Qu.:2.017e+09   3rd Qu.: 4.000   5_Poor     : 60   3rd Qu.:31.08  
 Max.   :2.017e+09   Max.   :30.000                     Max.   :70.56  
    age_imp          female                             race_eth  
 Min.   :18.00   Min.   :0.0000   White non-Hispanic      :824  
 1st Qu.:45.00   1st Qu.:0.0000   Black non-Hispanic      :226  
 Median :58.00   Median :1.0000   Other race non-Hispanic : 24  
 Mean   :57.33   Mean   :0.5931   Multiracial non-Hispanic: 24  
 3rd Qu.:70.00   3rd Qu.:1.0000   Hispanic                : 35  
 Max.   :95.00   Max.   :1.0000                                 
   internet30        smoke100                        activity     drinks_wk     
 Min.   :0.0000   Min.   :0.0000   Highly_Active        :428   Min.   : 0.000  
 1st Qu.:1.0000   1st Qu.:0.0000   Active               :173   1st Qu.: 0.000  
 Median :1.0000   Median :0.0000   Insufficiently_Active:201   Median : 0.230  
 Mean   :0.8164   Mean   :0.4704   Inactive             :331   Mean   : 2.474  
 3rd Qu.:1.0000   3rd Qu.:1.0000                               3rd Qu.: 2.100  
 Max.   :1.0000   Max.   :1.0000                               Max.   :56.000  
    veg_day       SEQNO_NA   physhealth_NA genhealth_NA bmi_NA     age_imp_NA
 Min.   :0.000   !NA:1133   !NA:1109      !NA:1129     !NA:1042   !NA:1122  
 1st Qu.:1.270   NA :   0   NA :  24      NA :   4     NA :  91   NA :  11  
 Median :1.730                                                              
 Mean   :1.928                                                              
 3rd Qu.:2.430                                                              
 Max.   :7.490
 female_NA  race_eth_NA internet30_NA smoke100_NA activity_NA drinks_wk_NA
 !NA:1133   !NA:1107    !NA:1126      !NA:1093    !NA:1024    !NA:1067
 NA :   0   NA :  26    NA :   7      NA :  40    NA : 109    NA :  66




 veg_day_NA
 !NA:1032
 NA : 101
```

4.1.2 The `inspect` function from the `mosaic` package

```
smart_cle1_cc %>% mosaic::inspect()
```

```
categorical variables:
       name  class levels    n missing
1 genhealth factor      5 1133       0
2  race_eth factor      5 1133       0
3  activity factor      4 1133       0
                                  distribution
1 2_VeryGood (34.1%), 3_Good (32.2%) ...
2 White non-Hispanic (72.7%) ...
3 Highly_Active (37.8%) ...

quantitative variables:
          name   class       min        Q1       median           Q3
...1      SEQNO numeric 2.017e+09 2.017e+09 2.017001e+09 2.017001e+09
...2 physhealth numeric 0.000e+00 0.000e+00 0.000000e+00 4.000000e+00
...3        bmi numeric 1.330e+01 2.438e+01 2.731000e+01 3.108000e+01
...4    age_imp numeric 1.800e+01 4.500e+01 5.800000e+01 7.000000e+01
...5     female numeric 0.000e+00 0.000e+00 1.000000e+00 1.000000e+00
...6 internet30 numeric 0.000e+00 1.000e+00 1.000000e+00 1.000000e+00
...7   smoke100 numeric 0.000e+00 0.000e+00 0.000000e+00 1.000000e+00
...8  drinks_wk numeric 0.000e+00 0.000e+00 2.300000e-01 2.100000e+00
...9    veg_day numeric 0.000e+00 1.270e+00 1.730000e+00 2.430000e+00
             max         mean          sd    n missing
...1 2.017001e+09 2.017001e+09 327.2132332 1133       0
...2 3.000000e+01 4.681377e+00   9.1208987 1133       0
```

```
...3 7.056000e+01 2.834932e+01   6.3651826 1133       0
...4 9.500000e+01 5.732568e+01  18.0803278 1133       0
...5 1.000000e+00 5.931156e-01   0.4914699 1133       0
...6 1.000000e+00 8.164166e-01   0.3873150 1133       0
...7 1.000000e+00 4.704325e-01   0.4993454 1133       0
...8 5.600000e+01 2.473689e+00   5.6900315 1133       0
...9 7.490000e+00 1.927926e+00   1.0412415 1133       0

shade variables:
            name class levels    n missing
1       SEQNO_NA shade      2 1133       0
2  physhealth_NA shade      2 1133       0
3   genhealth_NA shade      2 1133       0
4         bmi_NA shade      2 1133       0
5     age_imp_NA shade      2 1133       0
6      female_NA shade      2 1133       0
7    race_eth_NA shade      2 1133       0
8  internet30_NA shade      2 1133       0
9    smoke100_NA shade      2 1133       0
10   activity_NA shade      2 1133       0
11  drinks_wk_NA shade      2 1133       0
12    veg_day_NA shade      2 1133       0
                                   distribution
1  !NA (100%), NA (0%)
2  !NA (97.9%), NA (2.1%)
3  !NA (99.6%), NA (0.4%)
4  !NA (92%), NA (8%)
5  !NA (99%), NA (1%)
6  !NA (100%), NA (0%)
7  !NA (97.7%), NA (2.3%)
8  !NA (99.4%), NA (0.6%)
9  !NA (96.5%), NA (3.5%)
10 !NA (90.4%), NA (9.6%)
11 !NA (94.2%), NA (5.8%)
12 !NA (91.1%), NA (8.9%)
```

4.1.3 The `describe` function in `Hmisc`

This provides some useful additional summaries, including a list of the lowest and highest values (which is very helpful when checking data.)

```
smart_cle1_cc %>%
  select(bmi, genhealth, female) %>%
  Hmisc::describe()
```

.

```
 3  Variables      1133  Observations
--------------------------------------------------------------------------------
bmi
       n  missing distinct     Info     Mean      Gmd      .05      .10
    1133        0      558        1    28.35    6.681    20.09    21.37
     .25      .50      .75      .90      .95
   24.38    27.31    31.08    36.37    40.44

lowest : 13.30 13.64 15.59 15.71 15.75, highest: 56.31 57.12 58.98 63.00 70.56
--------------------------------------------------------------------------------
genhealth
       n  missing distinct
    1133        0        5

lowest : 1_Excellent 2_VeryGood  3_Good      4_Fair      5_Poor
highest: 1_Excellent 2_VeryGood  3_Good      4_Fair      5_Poor

Value      1_Excellent  2_VeryGood      3_Good      4_Fair      5_Poor
Frequency          164         386         365         158          60
Proportion       0.145       0.341       0.322       0.139       0.053
--------------------------------------------------------------------------------
female
       n  missing distinct     Info      Sum     Mean      Gmd
    1133        0        2    0.724      672   0.5931   0.4831

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

- The `Info` measure is used for quantitative and binary variables. It is a relative information measure that increases towards 1 for variables with no ties, and is smaller for variables with many ties.
- The `Gmd` is the Gini mean difference. It is a measure of spread (or dispersion), where larger values indicate greater spread in the distribution, like the standard deviation or the interquartile range. It is defined as the mean absolute difference between any pairs of observations.

See the Help file for `describe` in the `Hmisc` package for more details on these measures, and on the settings for `describe`.

4.2 Counting as exploratory data analysis

Counting and/or tabulating things can be amazingly useful. Suppose we want to understand the `genhealth` values, after our single imputation.

```
smart_cle1_sh %>% count(genhealth) %>%
  mutate(percent = 100*n / sum(n))
```

```
# A tibble: 5 x 3
  genhealth       n percent
* <fct>       <int>   <dbl>
1 1_Excellent   164   14.5
2 2_VeryGood    386   34.1
3 3_Good        365   32.2
4 4_Fair        158   13.9
5 5_Poor         60    5.30
```

We might use `tabyl` to do this job…

```
smart_cle1_sh %>%
  tabyl(genhealth) %>%
  adorn_pct_formatting(digits = 1) %>%
  knitr::kable()
```

genhealth	n	percent
1_Excellent	164	14.5%
2_VeryGood	386	34.1%
3_Good	365	32.2%
4_Fair	158	13.9%
5_Poor	60	5.3%

4.2.1 Did `genhealth` vary by smoking status?

```
smart_cle1_sh %>%
  count(genhealth, smoke100) %>%
  mutate(percent = 100*n / sum(n))
```

```
# A tibble: 10 x 4
   genhealth   smoke100     n percent
   <fct>          <dbl> <int>   <dbl>
 1 1_Excellent        0   105    9.27
 2 1_Excellent        1    59    5.21
 3 2_VeryGood         0   220   19.4
 4 2_VeryGood         1   166   14.7
 5 3_Good             0   184   16.2
 6 3_Good             1   181   16.0
 7 4_Fair             0    67    5.91
 8 4_Fair             1    91    8.03
 9 5_Poor             0    24    2.12
10 5_Poor             1    36    3.18
```

Suppose we want to find the percentage within each smoking status group. Here's one approach...

```
smart_cle1_sh %>%
    count(smoke100, genhealth) %>%
    group_by(smoke100) %>%
    mutate(prob = 100*n / sum(n))
```

```
# A tibble: 10 x 4
# Groups:   smoke100 [2]
   smoke100 genhealth       n  prob
      <dbl> <fct>       <int> <dbl>
 1        0 1_Excellent   105 17.5 
 2        0 2_VeryGood    220 36.7 
 3        0 3_Good        184 30.7 
 4        0 4_Fair         67 11.2 
 5        0 5_Poor         24  4   
 6        1 1_Excellent    59 11.1 
 7        1 2_VeryGood    166 31.1 
 8        1 3_Good        181 34.0 
 9        1 4_Fair         91 17.1 
10        1 5_Poor         36  6.75
```

And here's another ...

```
smart_cle1_sh %>%
  tabyl(smoke100, genhealth) %>%
  adorn_totals(where = c("row", "col")) %>%
  adorn_percentages(denominator = "row") %>%
  adorn_pct_formatting(digits = 1) %>%
  adorn_ns(position = "front")
```

```
 smoke100 1_Excellent  2_VeryGood      3_Good      4_Fair    5_Poor
        0 105 (17.5%) 220 (36.7%) 184 (30.7%)  67 (11.2%) 24 (4.0%)
        1  59 (11.1%) 166 (31.1%) 181 (34.0%)  91 (17.1%) 36 (6.8%)
    Total 164 (14.5%) 386 (34.1%) 365 (32.2%) 158 (13.9%) 60 (5.3%)
         Total
  600 (100.0%)
  533 (100.0%)
 1133 (100.0%)
```

4.2.2 What's the distribution of `physhealth`?

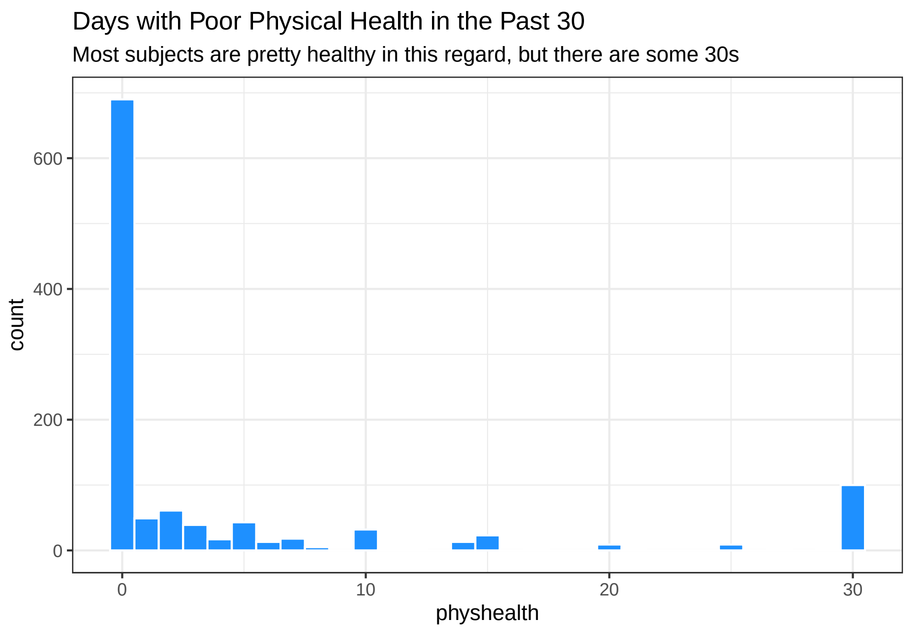
We can count quantitative variables with discrete sets of possible values, like `physhealth`, which is captured as an integer (that must fall between 0 and 30.)

```
smart_cle1_sh %>% count(physhealth)
```

```
# A tibble: 21 x 2
   physhealth     n
 *      <dbl> <int>
 1          0   690
 2          1    49
 3          2    61
 4          3    39
 5          4    17
 6          5    43
 7          6    13
 8          7    18
 9          8     5
10         10    32
# ... with 11 more rows
```

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

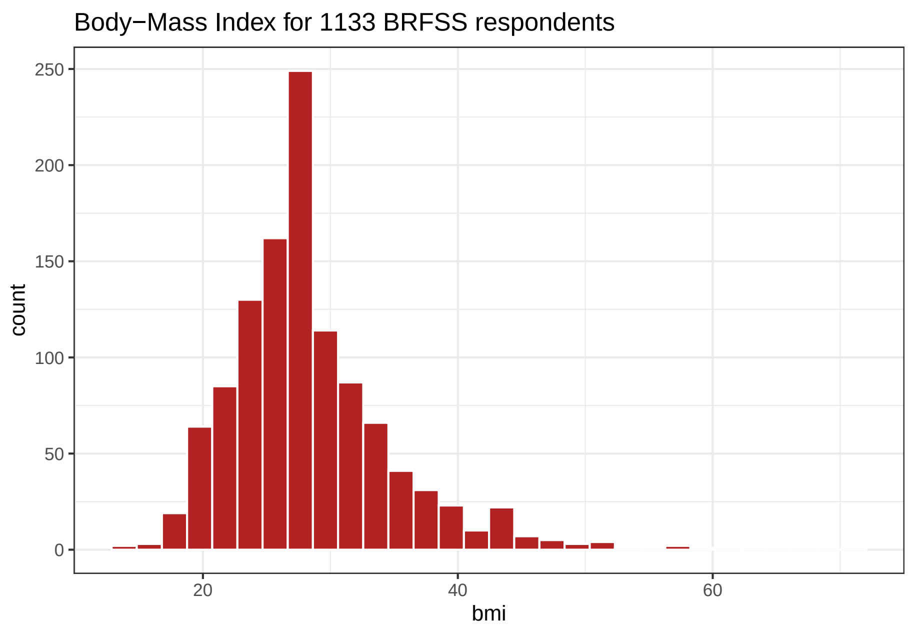
```
ggplot(smart_cle1_sh, aes(physhealth)) +
    geom_histogram(binwidth = 1,
                   fill = "dodgerblue", col = "white") +
    labs(title = "Days with Poor Physical Health in the Past 30",
         subtitle = "Most subjects are pretty healthy in this regard, but there are sor
```



4.2.3 What's the distribution of `bmi`?

`bmi` is the body-mass index, an indicator of size (thickness, really.)

```
ggplot(smart_cle1_sh, aes(bmi)) +
    geom_histogram(bins = 30,
                   fill = "firebrick", col = "white") +
    labs(title = paste0("Body-Mass Index for ",
                        nrow(smart_cle1_sh),
                        " BRFSS respondents"))
```



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

```
smart_cle1_sh %>% count(bmi < 30) %>%
  mutate(proportion = n / sum(n))
```

```
# A tibble: 2 x 3
  `bmi < 30`     n proportion
* <lgl>      <int>      <dbl>
1 FALSE        330      0.291
2 TRUE         803      0.709
```

4.2.5 How many of the respondents with a BMI < 30 are highly active?

```
smart_cle1_sh %>%
  filter(bmi < 30) %>%
  tabyl(activity) %>%
  adorn_pct_formatting()
```

```
              activity   n percent
         Highly_Active 343   42.7%
                Active 133   16.6%
 Insufficiently_Active 129   16.1%
              Inactive 198   24.7%
```

4.2.6 Is obesity associated with smoking history?

```
smart_cle1_sh %>% count(smoke100, bmi < 30) %>%
    group_by(smoke100) %>%
    mutate(percent = 100*n/sum(n))
```

```
# A tibble: 4 x 4
# Groups:   smoke100 [2]
  smoke100 `bmi < 30`     n percent
     <dbl> <lgl>      <int>   <dbl>
1        0 FALSE        163    27.2
2        0 TRUE         437    72.8
3        1 FALSE        167    31.3
4        1 TRUE         366    68.7
```

4.2.7 Comparing `drinks_wk` summaries by obesity status

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

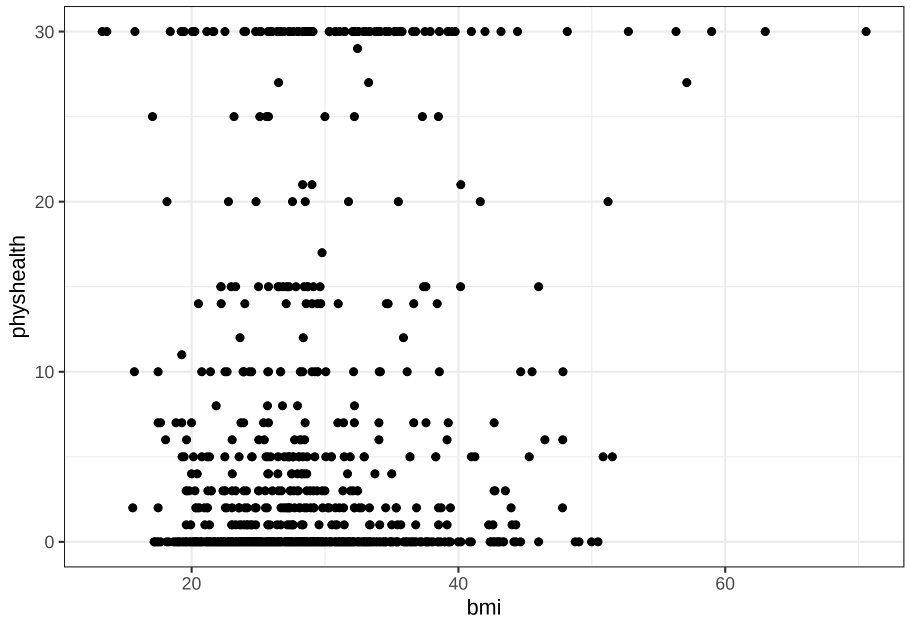
```
smart_cle1_sh %>%
    group_by(bmi < 30) %>%
    summarize(mean(drinks_wk), median(drinks_wk),
              q75 = quantile(drinks_wk, 0.75))
```

```
# A tibble: 2 x 4
  `bmi < 30` `mean(drinks_wk)` `median(drinks_wk)`   q75
* <lgl>                  <dbl>               <dbl> <dbl>
1 FALSE                   1.67                0.23  1.17
2 TRUE                    2.80                0.23  2.8
```

4.3 Can bmi predict physhealth?

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

```
ggplot(data = smart_cle1_sh, aes(x = bmi, y = physhealth)) +
    geom_point()
```

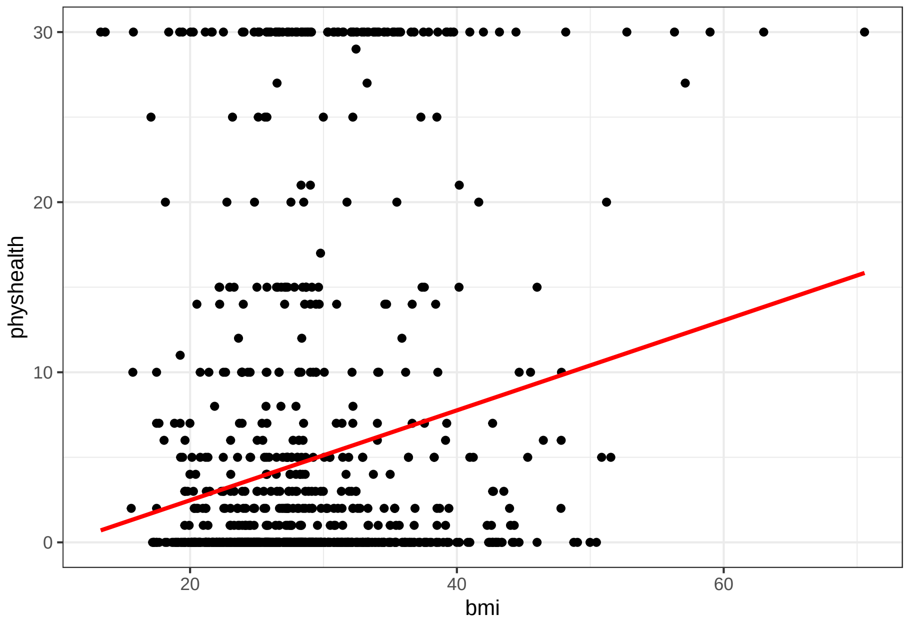


A good question to ask ourselves here might be: "In what BMI range can we make a reasonable prediction of `physhealth`?"

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

```
ggplot(data = smart_cle1_sh, aes(x = bmi, y = physhealth)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE, col = "red")
```

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



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

4.3.1 Fitting a Simple Regression Model

```
model_A <- lm(physhealth ~ bmi, data = smart_cle1_sh)

model_A
```

```

Call:
lm(formula = physhealth ~ bmi, data = smart_cle1_sh)

Coefficients:
(Intercept)          bmi
    -2.8121       0.2643
```

```
summary(model_A)
```

```

Call:
lm(formula = physhealth ~ bmi, data = smart_cle1_sh)

Residuals:
     Min       1Q   Median       3Q      Max
-10.5258  -4.5943  -3.5608  -0.5106  29.2965
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.81208    1.21672  -2.311    0.021 *
bmi          0.26433    0.04188   6.312 3.95e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.968 on 1131 degrees of freedom
Multiple R-squared:  0.03403,	Adjusted R-squared:  0.03317
F-statistic: 39.84 on 1 and 1131 DF,  p-value: 3.95e-10
```

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

```
                 2.5 %     97.5 %
(Intercept) -5.1993624 -0.4247909
bmi          0.1821599  0.3464915
```

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.

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

The fitted model predicts `physhealth` using a prediction equation we can read off from the model coefficient estimates. Specifically, we have:

```
coef(model_A)
```

```
(Intercept)         bmi
 -2.8120766   0.2643257
```

so the equation is `physhealth` = -2.82 + 0.265 `bmi`.

Each of the 1133 respondents included in the `smart_cle1_sh` data makes a contribution to this model.

4.3.2.1 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 -2.82 + 0.265 `bmi` for Harry. So, if Harry's BMI was 20, then Harry's predicted `physhealth` value is -2.82 + 0.265 (20) = 2.48.

- The *residual* for Harry is then his *observed* outcome minus his *fitted* outcome, so Harry has a residual of 3 - 2.48 = 0.52.
- 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.

```
summary(model_A)
```

```

Call:
lm(formula = physhealth ~ bmi, data = smart_cle1_sh)

Residuals:
     Min       1Q   Median       3Q      Max 
-10.5258  -4.5943  -3.5608  -0.5106  29.2965 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -2.81208    1.21672  -2.311    0.021 *  
bmi          0.26433    0.04188   6.312 3.95e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.968 on 1131 degrees of freedom
Multiple R-squared:  0.03403,	Adjusted R-squared:  0.03317 
F-statistic: 39.84 on 1 and 1131 DF,  p-value: 3.95e-10
```

- 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 `smart_cle1_sh` data, the minimum residual was -10.53, so for one subject, the observed value was 10.53 days smaller than the predicted value. This means that the prediction was 10.53 days too large for that subject.
- Similarly, the maximum residual was 29.30 days, so for one subject the prediction was 29.30 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 `smart_cle1_sh` data) of about 9.0 days. We know this because the residual standard error is specified as 8.968 later in the linear model output. Thus, by the definition of a Normal distribution, we'd expect
- about 68% of the residuals to be between -9 and +9 days,
- about 95% of the residuals to be between -18 and +18 days,
- about all (99.7%) of the residuals to be between -27 and +27 days.

4.3.2.2 Coefficients section

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

```
summary(model_A)
```

```
Call:
lm(formula = physhealth ~ bmi, data = smart_cle1_sh)

Residuals:
     Min       1Q   Median       3Q      Max 
-10.5258  -4.5943  -3.5608  -0.5106  29.2965 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -2.81208    1.21672  -2.311    0.021 *  
bmi          0.26433    0.04188   6.312 3.95e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.968 on 1131 degrees of freedom
Multiple R-squared:  0.03403,	Adjusted R-squared:  0.03317 
F-statistic: 39.84 on 1 and 1131 DF,  p-value: 3.95e-10
```

- The Estimates are the point estimates of the intercept and slope of `bmi` in our model.
- In this case, our estimated slope is 0.265, which implies that if Harry's BMI is 20 and Sally's BMI is 21, we predict that Sally's `physhealth` will be 0.265 days larger than Harry's.
- The Standard Errors are also provided for each estimate. We can create rough 95% uncertainty intervals for these estimated coefficients 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% uncertainty interval for the slope of `bmi` is estimated to be (0.18, 0.35). 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 detectably 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 detectably 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.

4.3.2.3 Model Fit Summaries

```
summary(model_A)
```

```
Call:
lm(formula = physhealth ~ bmi, data = smart_cle1_sh)

Residuals:
     Min       1Q   Median       3Q      Max 
-10.5258  -4.5943  -3.5608  -0.5106  29.2965 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -2.81208    1.21672  -2.311    0.021 *  
bmi          0.26433    0.04188   6.312 3.95e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.968 on 1131 degrees of freedom
Multiple R-squared:  0.03403,	Adjusted R-squared:  0.03317 
F-statistic: 39.84 on 1 and 1131 DF,  p-value: 3.95e-10
```

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 3.4% 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 smaller 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   3204  3204.4   39.84 3.95e-10 ***
Residuals 1131  90968    80.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.3.3 Using the `broom` package

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

4.3.3.1 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)
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   -2.81     1.22       -2.31 2.10e- 2
2 bmi            0.264    0.0419      6.31 3.95e-10
```

It's often useful to include other summaries in this tidying, for instance:

```
tidy(model_A, conf.int = TRUE, conf.level = 0.9) %>%
  select(term, estimate, conf.low, conf.high)
```

```
# A tibble: 2 x 4
  term        estimate conf.low conf.high
  <chr>          <dbl>    <dbl>     <dbl>
1 (Intercept)   -2.81    -4.82     -0.809
2 bmi            0.264    0.195     0.333
```

4.3.3.2 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)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
1    0.0340        0.0332  8.97      39.8 3.95e-10     1 -4092. 8190. 8205.
# ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

4.3.3.3 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))
```

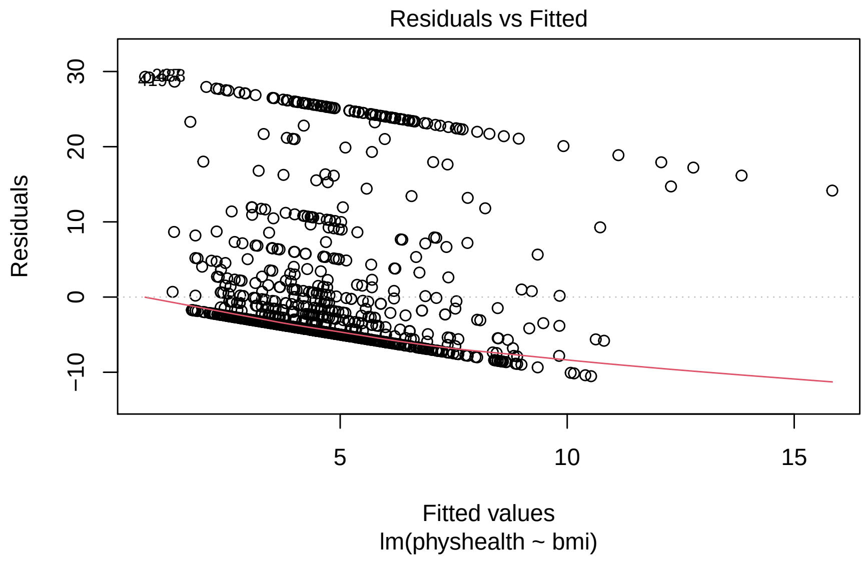
```
# A tibble: 6 x 8
  physhealth   bmi .fitted .resid .std.resid     .hat .sigma    .cooksd
       <dbl> <dbl>   <dbl>  <dbl>      <dbl>    <dbl>  <dbl>      <dbl>
1          4  27.9    4.57 -0.572    -0.0638 0.000886   8.97 0.00000181
2          0  23.0    3.28 -3.28     -0.366  0.00149    8.97 0.000100
3          0  26.9    4.31 -4.31     -0.480  0.000927   8.97 0.000107
4          0  26.5    4.20 -4.20     -0.468  0.000956   8.97 0.000105
5          0  24.2    3.60 -3.60     -0.401  0.00125    8.97 0.000101
6          2  27.7    4.51 -2.51     -0.281  0.000891   8.97 0.0000351
```

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

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

- Remember that the R^2 value was about 3.4%.

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

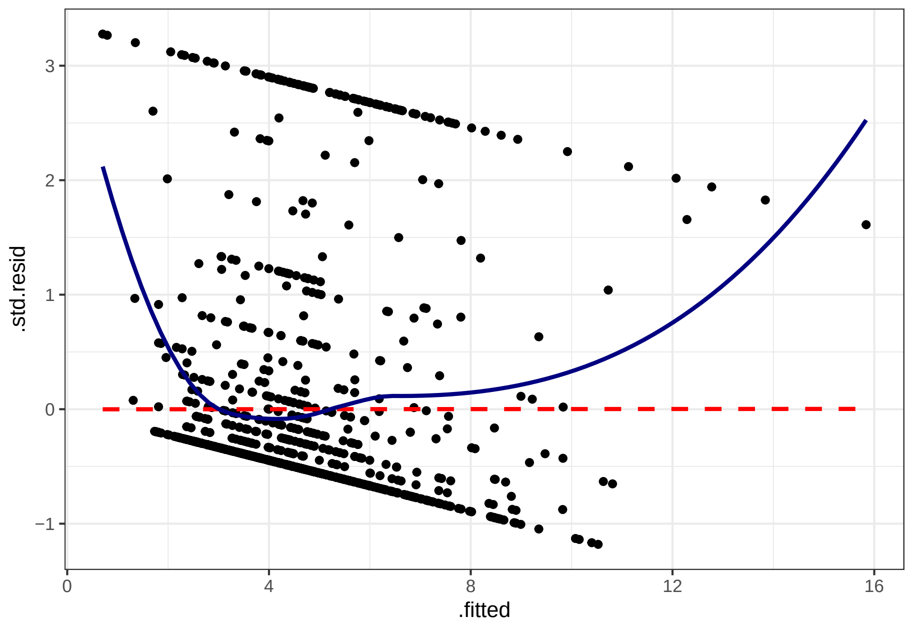


This is a plot of residuals vs. fitted values. The goal here is for this plot to look like a random scatter of points, perhaps like a "fuzzy football," and that's **not** what we have. Why?

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.

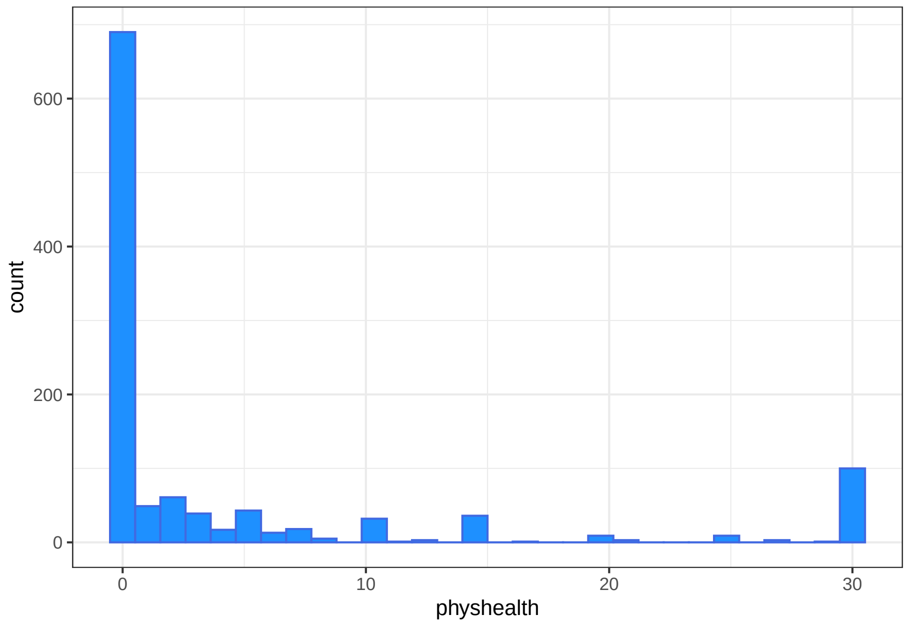
```
ggplot(augment(model_A), aes(x = .fitted, y = .std.resid)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE, col = "red", linetype = "dashed") +
    geom_smooth(method = "loess", se = FALSE, col = "navy") +
    theme_bw()
```

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



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(smart_cle1_sh, aes(x = physhealth)) +
  geom_histogram(bins = 30, fill = "dodgerblue",
                 color = "royalblue")
```



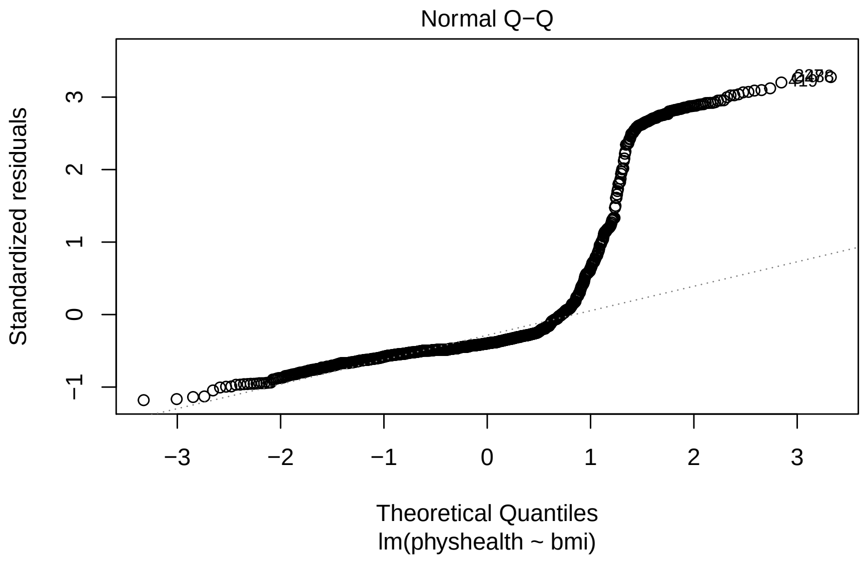
```
smart_cle1_sh %>% count(physhealth == 0, physhealth == 30)
```

```
# A tibble: 3 x 3
  `physhealth == 0` `physhealth == 30`     n
  <lgl>             <lgl>              <int>
1 FALSE             FALSE                343
2 FALSE             TRUE                 100
3 TRUE              FALSE                690
```

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. We'll try again, with a new outcome, in the next chapter.

Chapter 5

Analysis of Variance with SMART

In this chapter, we'll work with the `smart_cle1_sh` data file again.

```
smart_cle1_sh <- readRDS(here("data", "smart_cle1_sh.Rds"))
```

The variables we'll look at in this chapter are as follows.

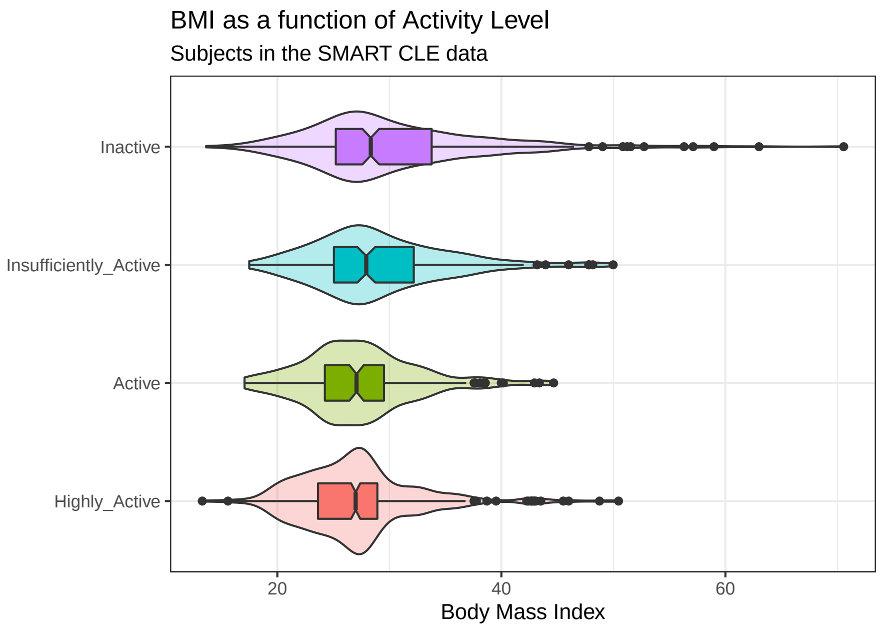
Variable	Description
`SEQNO`	respondent identification number (all begin with 2016)
`bmi`	Body mass index, in kg/m^2
`female`	Sex, 1 = female, 0 = male
`smoke100`	Have you smoked at least 100 cigarettes in your life? (1 = yes, 0 = no)
`activity`	Physical activity (Highly Active, Active, Insufficiently Active, Inactive)
`drinks_wk`	On average, how many drinks of alcohol do you consume in a week?
`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?

5.1 A One-Factor Analysis of Variance

We'll be predicting body mass index, at first using a single factor as a predictor: the `activity` level.

5.1.1 Can activity be used to predict bmi?

```
ggplot(smart_cle1_sh, aes(x = activity, y = bmi,
                          fill = activity)) +
  geom_violin(alpha = 0.3) +
  geom_boxplot(width = 0.3, notch = TRUE) +
  guides(fill = FALSE) +
  coord_flip() +
  labs(title = "BMI as a function of Activity Level",
       subtitle = "Subjects in the SMART CLE data",
       x = "", y = "Body Mass Index")
```



Here's a numerical summary of the distributions of bmi within each activity group.

```
smart_cle1_sh %$% mosaic::favstats(bmi ~ activity)
```

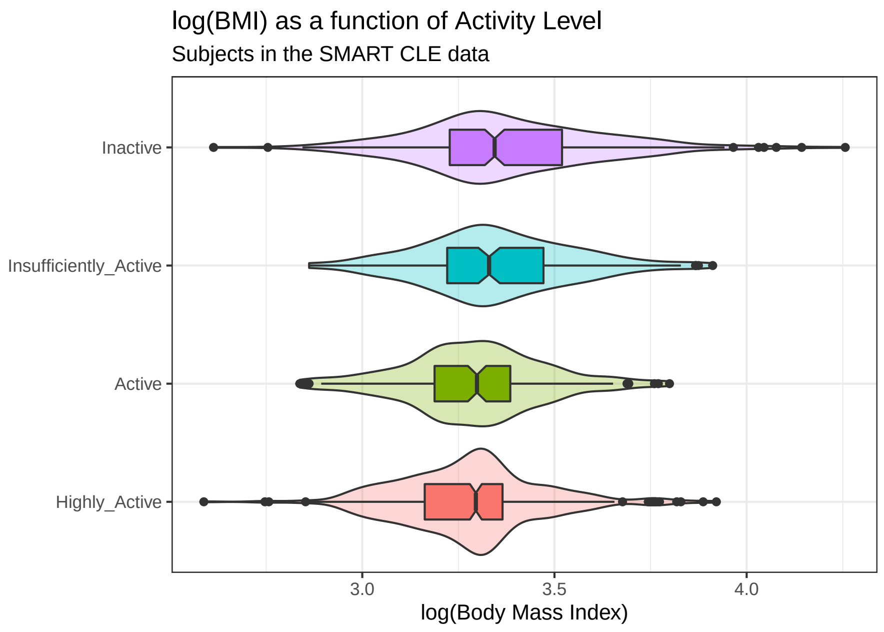
```
               activity   min      Q1   median     Q3   max     mean       sd
1         Highly_Active 13.30 23.6275 26.99000 28.930 50.46 27.02253 5.217496
2                Active 17.07 24.2400 27.06930 29.520 44.67 27.36157 5.151796
3 Insufficiently_Active 17.49 25.0500 27.93776 32.180 49.98 29.04328 6.051823
4              Inactive 13.64 25.2150 28.34000 33.775 70.56 30.15978 7.832675
    n missing
1 428       0
2 173       0
```

```
3 201        0
4 331        0
```

5.1.2 Should we transform bmi?

The analysis of variance is something of a misnomer. What we're doing is using the variance to say something about population means. In light of the apparent right skew of the `bmi` results in each `activity` group, might it be a better choice to use a logarithmic transformation? We'll use the natural logarithm here, which in R, is symbolized by `log`.

```
ggplot(smart_cle1_sh, aes(x = activity, y = log(bmi),
                          fill = activity)) +
  geom_violin(alpha = 0.3) +
  geom_boxplot(width = 0.3, notch = TRUE) +
  guides(fill = FALSE) +
  coord_flip() +
  labs(title = "log(BMI) as a function of Activity Level",
       subtitle = "Subjects in the SMART CLE data",
       x = "", y = "log(Body Mass Index)")
```



The logarithmic transformation yields distributions that look much more symmetric in each `activity` group, so we'll proceed to build our regression model predicting `log(bmi)` using `activity`. Here's the numerical summary of these logged results:

```
smart_cle1_sh %$% mosaic::favstats(log(bmi) ~ activity)
```

```
               activity      min       Q1   median       Q3      max     mean
1         Highly_Active 2.587764 3.162411 3.295466 3.364879 3.921181 3.279246
2                Active 2.837323 3.188004 3.298400 3.385068 3.799302 3.292032
3 Insufficiently_Active 2.861629 3.220874 3.329979 3.471345 3.911623 3.348383
4              Inactive 2.613007 3.227439 3.344274 3.519721 4.256463 3.376468
         sd   n missing
1 0.1851478 428       0
2 0.1850568 173       0
3 0.2007241 201       0
4 0.2411196 331       0
```

5.1.3 Building the ANOVA model

```
model_5a <- smart_cle1_sh %$% lm(log(bmi) ~ activity)

model_5a
```

```
Call:
lm(formula = log(bmi) ~ activity)

Coefficients:
                  (Intercept)                 activityActive
                      3.27925                        0.01279
activityInsufficiently_Active               activityInactive
                      0.06914                        0.09722
```

The `activity` data is categorical and there are four levels. The model equation is:

```
log(bmi) = 3.279 + 0.013 (activity = Active)
                 + 0.069 (activity = Insufficiently Active)
                 + 0.097 (activity = Inactive)
```

where, for example, `(activity = Active)` is 1 if `activity` is Active, and 0 otherwise. The fourth level (Highly Active) is not shown here and is used as a baseline. Thus the model above can be interpreted as follows.

`activity`	Predicted `log(bmi)`	Predicted `bmi`
Highly Active	3.279	exp(3.279) = 26.55
Active	3.279 + 0.013 = 3.292	exp(3.292) = 26.90
Insufficiently Active	3.279 + 0.069 = 3.348	exp(3.348) = 28.45
Inactive	3.279 + 0.097 = 3.376	exp(3.376) = 29.25

Those predicted `log(bmi)` values should look familiar. They are just the means of `log(bmi)` in each group, but I'm sure you'll also notice that the predicted `bmi` values are not exact matches for the observed means of `bmi`.

```
smart_cle1_sh %>% group_by(activity) %>%
  summarise(mean(log(bmi)), mean(bmi))
```

```
# A tibble: 4 x 3
  activity              `mean(log(bmi))` `mean(bmi)`
* <fct>                            <dbl>       <dbl>
1 Highly_Active                     3.28        27.0
2 Active                            3.29        27.4
3 Insufficiently_Active             3.35        29.0
4 Inactive                          3.38        30.2
```

5.1.4 The ANOVA table

Now, let's press on to look at the ANOVA results for this model.

```
anova(model_5a)
```

```
Analysis of Variance Table

Response: log(bmi)
            Df Sum Sq Mean Sq F value    Pr(>F)
activity     3  2.060 0.68652  16.225 2.496e-10 ***
Residuals 1129 47.772 0.04231
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The total variation in `log(bmi)`, our outcome, is captured by the sums of squares here. SS(Total) = 2.058 + 47.770 = 49.828
- Here, the `activity` variable (with 4 levels, so 4-1 = 3 degrees of freedom) accounts for 4.13% (2.058 / 49.828) of the variation in `log(bmi)`. Another way of saying this is that the model R^2 or η^2 is 0.0413.
- The variation accounted for by the `activity` categories meets the standard for a statistically detectable result, according to the ANOVA F test, although that's not really important.
- The square root of the Mean Square(Residuals) is the residual standard error, σ, we've seen in the past. MS(Residual) estimates the variance (0.0423), so the residual standard error is $\sqrt{0.0423} \approx 0.206$.

5.1.5 The Model Coefficients

To address the question of effect size for the various levels of `activity` on `log(bmi)`, we could look directly at the regression model coefficients. For that, we might look at the model `summary`.

```
summary(model_5a)
```

```

Call:
lm(formula = log(bmi) ~ activity)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.76346 -0.12609 -0.00286  0.11055  0.88000 

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   3.279246   0.009943 329.806  < 2e-16 ***
activityActive                0.012785   0.018532   0.690     0.49    
activityInsufficiently_Active 0.069137   0.017589   3.931 8.99e-05 ***
activityInactive              0.097221   0.015056   6.457 1.58e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2057 on 1129 degrees of freedom
Multiple R-squared:  0.04133,	Adjusted R-squared:  0.03878 
F-statistic: 16.22 on 3 and 1129 DF,  p-value: 2.496e-10
```

If we want to see the confidence intervals around these estimates, we could use

```
confint(model_5a, conf.level = 0.95)
```

```
                                    2.5 %     97.5 %
(Intercept)                    3.25973769 3.29875522
activityActive                -0.02357630 0.04914707
activityInsufficiently_Active  0.03462572 0.10364764
activityInactive               0.06767944 0.12676300
```

The model suggests, based on these 1133 subjects, that (remember that the baseline category is Highly Active)

- a 95% confidence (uncertainty) interval for the difference between Active and Highly Active subjects in log(BMI) ranges from -0.024 to 0.049
- a 95% confidence (uncertainty) interval for the difference between Insufficiently Active and Highly Active subjects in log(BMI) ranges from 0.035 to 0.104
- a 95% confidence (uncertainty) interval for the difference between Inactive and Highly Active subjects in log(BMI) ranges from 0.068 to 0.127
- the model accounts for 4.13% of the variation in log(BMI), so that knowing the respondent's activity level somewhat reduces the size of the prediction errors as compared to an intercept only model that would predict the overall mean log(BMI), regardless of activity level, for all subjects.

- from the summary of residuals, we see that one subject had a residual of 0.88 - that means they were predicted to have a log(BMI) 0.88 lower than their actual log(BMI) and one subject had a log(BMI) that is 0.76 larger than their actual log(BMI), at the extremes.

5.1.6 Using `broom::tidy` to explore the coefficients

A better strategy for displaying the coefficients in any regression model is to use the `tidy` function from the `broom` package.

```
tidy(model_5a, conf.int = TRUE, conf.level = 0.95) %>%
  knitr::kable(digits = 3)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	3.279	0.010	329.806	0.00	3.260	3.299
activityActive	0.013	0.019	0.690	0.49	-0.024	0.049
activityInsufficiently_Active	0.069	0.018	3.931	0.00	0.035	0.104
activityInactive	0.097	0.015	6.457	0.00	0.068	0.127

5.1.7 Using `broom::glance` to summarize the model's fit

```
glance(model_5a) %>% select(1:3) %>%
  knitr::kable(digits = c(4, 4, 3))
```

r.squared	adj.r.squared	sigma
0.0413	0.0388	0.206

- The `r.squared` or R^2 value is interpreted for a linear model as the percentage of variation in the outcome (here, `log(bmi)`) that is accounted for by the model.
- The `adj.r.squared` or adjusted R^2 value incorporates a small penalty for the number of predictors included in the model. Adjusted R^2 is useful for models with more than one predictor, not simple regression models like this one. Like R^2 and most of these other summaries, its primary value comes when making comparisons between models for the same outcome.
- The `sigma` or σ is the residual standard error. Doubling this value gives us a good idea of the range of errors made by the model (approximately 95% of the time if the normal distribution assumption for the residuals holds perfectly.)

```
glance(model_5a) %>% select(4:7) %>%
  knitr::kable(digits = c(2, 3, 0, 2))
```

statistic	p.value	df	logLik
16.22	0	3	185.99

- The `statistic` and `p.value` shown here refer to the ANOVA F test and p value. They test the null hypothesis that the `activity` information is of no use in separating out the `bmi` data, or, equivalently, that the true R^2 is 0.
- The `df` indicates the model degrees of freedom, and in this case simply specifies the number of parameters fitted attributed to the model. Models that require more `df` for estimation require larger sample sizes.
- The `logLik` is the log likelihood for the model. This is a function of the sample size, but we can compare the fit of multiple models by comparing this value across different models for the same outcome. You want to maximize the log-likelihood.

```
glance(model_5a) %>% select(8:9) %>%
  knitr::kable(digits = 2)
```

AIC	BIC
-361.98	-336.82

- The `AIC` (or Akaike information criterion) and `BIC` (Bayes information criterion) are also used only to compare models. You want to minimize AIC and BIC in selecting a model. AIC and BIC are unique only up to a constant, so different packages or routines in R may give differing values, but in comparing two models - the difference in AIC (or BIC) should be consistent.

5.1.8 Using `broom::augment` to make predictions

We can obtain residuals and predicted (fitted) values for the points used to fit the model with `augment` from the `broom` package.

```
augment(model_5a, se_fit = TRUE) %>%
  select(1:5) %>% slice(1:4) %>%
  knitr::kable(digits = 3)
```

log(bmi)	activity	.fitted	.se.fit	.resid
3.330	Inactive	3.376	0.011	-0.047
3.138	Inactive	3.376	0.011	-0.239
3.293	Insufficiently_Active	3.348	0.015	-0.055
3.278	Highly_Active	3.279	0.010	-0.002

- The `.fitted` value is the predicted value of `log(bmi)` for this subject.
- The `.se.fit` value shows the standard error associated with the fitted value.
- The `.resid` is the residual value (observed - fitted `log(bmi)`)

```
augment(model_5a, se_fit = TRUE) %>%
  select(1:2, 6:9) %>% slice(1:4) %>%
  knitr::kable(digits = 3)
```

log(bmi)	activity	.std.resid	.hat	.sigma	.cooksd
3.330	Inactive	-0.227	0.003	0.206	0.000
3.138	Inactive	-1.163	0.003	0.206	0.001
3.293	Insufficiently_Active	-0.269	0.005	0.206	0.000
3.278	Highly_Active	-0.008	0.002	0.206	0.000

- The `.hat` value shows the leverage index associated with the observation (this is a function of the predictors - higher leveraged points have more unusual predictor values)
- The `.sigma` value shows the estimate of the residual standard deviation if this observation were to be dropped from the model, and thus indexes how much of an outlier this observation's residual is.
- The `.cooksd` or Cook's distance value shows the influence that the observation has on the model - it is one of a class of leave-one-out diagnostic measures. Larger values of Cook's distance indicate more influential points.
- The `.std.resid` shows the standardized residual (which is designed to have mean 0 and standard deviation 1, facilitating comparisons across models for differing outcomes)

5.2 A Two-Factor ANOVA (without Interaction)

Let's add `race_eth` to the predictor set for `log(BMI)`.

```
model_5b <- smart_cle1_sh %$%
  lm(log(bmi) ~ activity + race_eth)

anova(model_5b)
```

```
Analysis of Variance Table

Response: log(bmi)
            Df Sum Sq Mean Sq F value    Pr(>F)    
activity     3  2.060 0.68652 16.5090 1.676e-10 ***
race_eth     4  0.989 0.24716  5.9435 9.843e-05 ***
Residuals 1125 46.783 0.04158                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Notice that the ANOVA model assesses these variables sequentially, so the SS(activity) = 2.058 is accounted for before we consider the SS(race_eth) = 0.990. Thus, in total, the model accounts for 2.058 + 0.990 = 3.048 of the sums of squares in `log(bmi)` in these data.

If we flip the order in the model, like this:

```
smart_cle1_sh %$%
  lm(log(bmi) ~ race_eth + activity) %>%
  anova()
```

```
Analysis of Variance Table

Response: log(bmi)
            Df Sum Sq Mean Sq F value    Pr(>F)
race_eth     4  1.119 0.27981  6.7287 2.371e-05 ***
activity     3  1.929 0.64299 15.4620 7.332e-10 ***
Residuals 1125 46.783 0.04158
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- After flipping the order of the predictors, `race_eth` accounts for a larger Sum of Squares than it did previously, but `activity` accounts for a smaller amount, and the total between `race_eth` and `activity` remains the same, as 1.121 + 1.927 is still 3.048.

5.2.1 Model Coefficients

The model coefficients are unchanged regardless of the order of the variables in our two-factor ANOVA model.

```
tidy(model_5b, conf.int = TRUE, conf.level = 0.95) %>%
  select(term, estimate, std.error, conf.low, conf.high) %>%
  knitr::kable(digits = 3)
```

term	estimate	std.error	conf.low	conf.high
(Intercept)	3.268	0.010	3.247	3.288
activityActive	0.012	0.018	-0.024	0.048
activityInsufficiently_Active	0.073	0.018	0.039	0.108
activityInactive	0.092	0.015	0.063	0.122
race_ethBlack non-Hispanic	0.066	0.015	0.036	0.096
race_ethOther race non-Hispanic	-0.086	0.042	-0.169	-0.002
race_ethMultiracial non-Hispanic	0.020	0.042	-0.063	0.103
race_ethHispanic	0.012	0.035	-0.057	0.082

The `model_5b` equation is:

```
log(BMI) = 3.268
      + 0.012 (activity = Active)
      + 0.073 (activity = Insufficiently Active)
      + 0.092 (activity = Inactive)
      + 0.066 (race_eth = Black non-Hispanic)
      - 0.086 (race_eth = Other race non-Hispanic)
      + 0.020 (race_eth = Multiracial non-Hispanic)
      + 0.012 (race_eth = Hispanic)
```

and we can make predictions by filling in appropriate 1s and 0s for the indicator variables in parentheses.

For example, the predicted `log(BMI)` for a White Highly Active person is 3.268, as White and Highly Active are the baseline categories in our two factors.

For all other combinations, we can make predictions as follows:

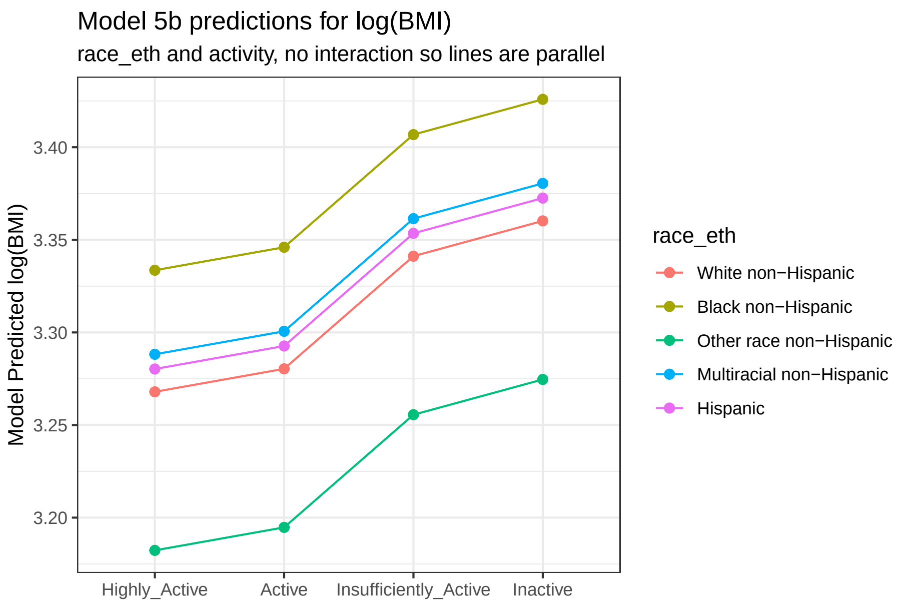
```
new_dat = tibble(
  race_eth = rep(c("White non-Hispanic",
                   "Black non-Hispanic",
                   "Other race non-Hispanic",
                   "Multiracial non-Hispanic",
                   "Hispanic"), 4),
  activity = c(rep("Highly_Active", 5),
               rep("Active", 5),
               rep("Insufficiently_Active", 5),
               rep("Inactive", 5))
  )

augment(model_5b, newdata = new_dat)
```

```
Warning: 'newdata' had 20 rows but variables found have 1133 rows

# A tibble: 20 x 3
   race_eth                 activity              .fitted
   <chr>                    <chr>                   <dbl>
 1 White non-Hispanic       Highly_Active            3.27
 2 Black non-Hispanic       Highly_Active            3.33
 3 Other race non-Hispanic  Highly_Active            3.18
 4 Multiracial non-Hispanic Highly_Active            3.29
 5 Hispanic                 Highly_Active            3.28
 6 White non-Hispanic       Active                   3.28
 7 Black non-Hispanic       Active                   3.35
 8 Other race non-Hispanic  Active                   3.19
 9 Multiracial non-Hispanic Active                   3.30
10 Hispanic                 Active                   3.29
11 White non-Hispanic       Insufficiently_Active    3.34
12 Black non-Hispanic       Insufficiently_Active    3.41
13 Other race non-Hispanic  Insufficiently_Active    3.26
14 Multiracial non-Hispanic Insufficiently_Active    3.36
15 Hispanic                 Insufficiently_Active    3.35
16 White non-Hispanic       Inactive                 3.36
17 Black non-Hispanic       Inactive                 3.43
18 Other race non-Hispanic  Inactive                 3.27
19 Multiracial non-Hispanic Inactive                 3.38
```

```
20 Hispanic                    Inactive                 3.37
```

```
augment(model_5b, newdata = new_dat) %>%
  mutate(race_eth = fct_relevel(factor(race_eth),
                                "White non-Hispanic",
                                "Black non-Hispanic",
                                "Other race non-Hispanic",
                                "Multiracial non-Hispanic",
                                "Hispanic"),
         activity = fct_relevel(factor(activity),
                                "Highly_Active",
                                "Active",
                                "Insufficiently_Active",
                                "Inactive")) %>%
  ggplot(., aes(x = activity, y = .fitted,
                col = race_eth, group = race_eth)) +
  geom_point(size = 2) +
  geom_line() +
  labs(title = "Model 5b predictions for log(BMI)",
       subtitle = "race_eth and activity, no interaction so lines are parallel",
       y = "Model Predicted log(BMI)",
       x = "")
```

```
Warning: 'newdata' had 20 rows but variables found have 1133 rows
```



The lines joining the points for each `race_eth` category are parallel to each other.

The groups always hold the same position relative to each other, regardless of their activity levels, and vice versa. There is no interaction in this model allowing the predicted effects of, say, `activity` on `log(BMI)` values to differ for the various `race_eth` groups. To do that, we'd have to fit the two-factor ANOVA model incorporating an interaction term.

5.3 A Two-Factor ANOVA (with Interaction)

Let's add the interaction of `activity` and `race_eth` (symbolized in R by `activity * race_eth`) to the model for `log(BMI)`.

```
model_5c <- smart_cle1_sh %$%
  lm(log(bmi) ~ activity * race_eth)

anova(model_5c)
```

```
Analysis of Variance Table

Response: log(bmi)
                    Df Sum Sq Mean Sq F value    Pr(>F)
activity             3  2.060 0.68652 16.4468 1.839e-10 ***
race_eth             4  0.989 0.24716  5.9211 0.0001026 ***
activity:race_eth   12  0.324 0.02700  0.6469 0.8028368
Residuals         1113 46.459 0.04174
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The ANOVA model shows that the SS(interaction) = SS(activity:race_eth) is 0.324, and uses 12 degrees of freedom. The model including the interaction term now accounts for 2.058 + 0.990 + 0.324 = 3.372, which is 6.8% of the variation in `log(BMI)` overall (which is calculated as SS(Total) = 2.058 + 0.990 + 0.324 + 46.456 = 49.828.)

5.3.1 Model Coefficients

The model coefficients now include additional product terms that incorporate indicator variables for both activity and race_eth. For each of the product terms to take effect, both their activity and race_eth status must yield a 1 in the indicator variables.

```
tidy(model_5c, conf.int = TRUE, conf.level = 0.95) %>%
  select(term, estimate, std.error, conf.low, conf.high) %>%
  knitr::kable(digits = 3)
```

term	estimate	std.error	conf.low	con
(Intercept)	3.264	0.011	3.242	
activityActive	0.021	0.021	-0.021	
activityInsufficiently_Active	0.079	0.020	0.039	
activityInactive	0.097	0.018	0.063	
race_ethBlack non-Hispanic	0.062	0.026	0.011	
race_ethOther race non-Hispanic	-0.070	0.078	-0.223	
race_ethMultiracial non-Hispanic	0.067	0.060	-0.051	
race_ethHispanic	0.110	0.060	-0.008	
activityActive:race_ethBlack non-Hispanic	-0.001	0.048	-0.096	
activityInsufficiently_Active:race_ethBlack non-Hispanic	0.005	0.046	-0.086	
activityInactive:race_ethBlack non-Hispanic	0.008	0.037	-0.065	
activityActive:race_ethOther race non-Hispanic	-0.065	0.165	-0.389	
activityInsufficiently_Active:race_ethOther race non-Hispanic	-0.035	0.101	-0.233	
activityInactive:race_ethOther race non-Hispanic	0.033	0.129	-0.221	
activityActive:race_ethMultiracial non-Hispanic	-0.208	0.134	-0.470	
activityInsufficiently_Active:race_ethMultiracial non-Hispanic	-0.050	0.120	-0.285	
activityInactive:race_ethMultiracial non-Hispanic	-0.056	0.110	-0.272	
activityActive:race_ethHispanic	-0.104	0.096	-0.291	
activityInsufficiently_Active:race_ethHispanic	-0.240	0.214	-0.660	
activityInactive:race_ethHispanic	-0.169	0.082	-0.331	

The `model_5c` equation is:

```
log(BMI) = 3.264
  + 0.021 (activity = Active)
  + 0.079 (activity = Insufficiently Active)
  + 0.097 (activity = Inactive)
  + 0.062 (race_eth = Black non-Hispanic)
  - 0.070 (race_eth = Other race non-Hispanic)
  + 0.067 (race_eth = Multiracial non-Hispanic)
  + 0.110 (race_eth = Hispanic)
  - 0.002 (activity = Active)(race_eth = Black non-Hispanic)
  + 0.005 (Insufficiently Active)(Black non-Hispanic)
  + 0.008 (Inactive)(Black non-Hispanic)
  - 0.065 (Active)(Other race non-Hispanic)
  - 0.035 (Insufficiently Active)(Other race non-Hispanic)
  + 0.033 (Inactive)(Other race non-Hispanic)
  - 0.208 (Active)(Multiracial non-Hispanic)
  - 0.050 (Insufficiently Active)(Multiracial non-Hispanic)
  - 0.056 (Inactive)(Multiracial non-Hispanic)
  - 0.104 (Active)(Hispanic)
  - 0.240 (Insufficiently Active)(Hispanic)
  - 0.169 (Inactive)(Hispanic)
```

and again, we can make predictions by filling in appropriate 1s and 0s for the indicator variables in parentheses.

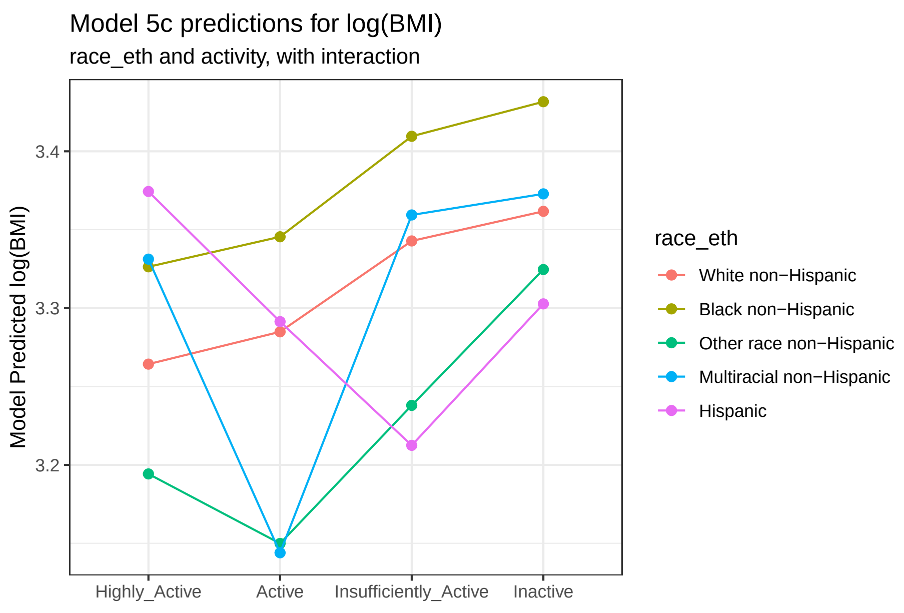
For example, the predicted `log(BMI)` for a White Highly Active person is 3.264, as White and Highly Active are the baseline categories in our two factors.

But the predicted `log(BMI)` for a Hispanic Inactive person would be 3.264 + 0.097 + 0.110 - 0.169 = 3.302.

Again, we'll plot the predicted `log(BMI)` predictions for each possible combination.

```
new_dat = tibble(
  race_eth = rep(c("White non-Hispanic",
                   "Black non-Hispanic",
                   "Other race non-Hispanic",
                   "Multiracial non-Hispanic",
                   "Hispanic"), 4),
  activity = c(rep("Highly_Active", 5),
               rep("Active", 5),
               rep("Insufficiently_Active", 5),
               rep("Inactive", 5))
  )

augment(model_5c, newdata = new_dat) %>%
  mutate(race_eth = fct_relevel(factor(race_eth),
                                "White non-Hispanic",
                                "Black non-Hispanic",
                                "Other race non-Hispanic",
                                "Multiracial non-Hispanic",
                                "Hispanic"),
         activity = fct_relevel(factor(activity),
                                "Highly_Active",
                                "Active",
                                "Insufficiently_Active",
                                "Inactive")) %>%
  ggplot(., aes(x = activity, y = .fitted,
                col = race_eth, group = race_eth)) +
  geom_point(size = 2) +
  geom_line() +
  labs(title = "Model 5c predictions for log(BMI)",
       subtitle = "race_eth and activity, with interaction",
       y = "Model Predicted log(BMI)",
       x = "")
```

```
Warning: 'newdata' had 20 rows but variables found have 1133 rows
```



Note that the lines joining the points for each `race_eth` category are no longer parallel to each other. The race-ethnicity group relative positions on `log(BMI)` is now changing depending on the `activity` status.

5.3.2 Is the interaction term necessary?

We can assess this in three ways, in order of importance:

1. With an interaction plot
2. By assessing the fraction of the variation in the outcome accounted for by the interaction
3. By assessing whether the interaction accounts for statistically detectable outcome variation

5.3.2.1 The Interaction Plot

A simple interaction plot is just a plot of the unadjusted outcome means, stratified by the two factors. For example, consider this plot for our two-factor ANOVA model. To obtain this plot, we first summarize the means within each group.

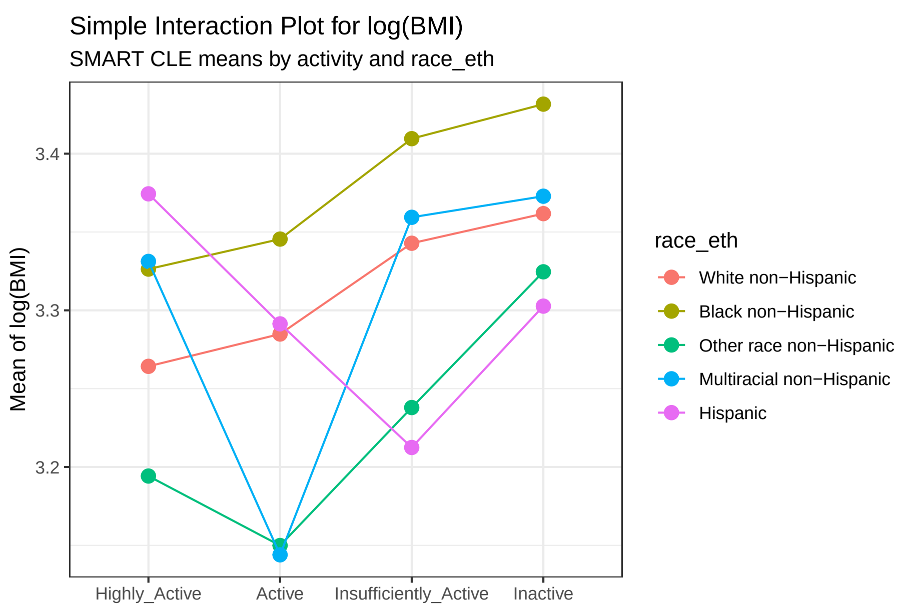
```
summaries_5 <- smart_cle1_sh %>%
  group_by(activity, race_eth) %>%
  summarize(n = n(), mean = mean(log(bmi)),
            sd = sd(log(bmi)))
```

```
`summarise()` has grouped output by 'activity'. You can override using the `.groups` argument.
```

```
summaries_5
```

```
# A tibble: 20 x 5
# Groups:   activity [4]
   activity              race_eth                      n  mean      sd
   <fct>                 <fct>                     <int> <dbl>   <dbl>
 1 Highly_Active         White non-Hispanic          320  3.26  0.176
 2 Highly_Active         Black non-Hispanic           77  3.33  0.190
 3 Highly_Active         Other race non-Hispanic       7  3.19  0.198
 4 Highly_Active         Multiracial non-Hispanic     12  3.33  0.187
 5 Highly_Active         Hispanic                     12  3.37  0.296
 6 Active                White non-Hispanic          129  3.28  0.173
 7 Active                Black non-Hispanic           31  3.35  0.224
 8 Active                Other race non-Hispanic       2  3.15  0.0845
 9 Active                Multiracial non-Hispanic      3  3.14  0.121
10 Active                Hispanic                      8  3.29  0.213
11 Insufficiently_Active White non-Hispanic          150  3.34  0.194
12 Insufficiently_Active Black non-Hispanic           35  3.41  0.213
13 Insufficiently_Active Other race non-Hispanic      11  3.24  0.137
14 Insufficiently_Active Multiracial non-Hispanic      4  3.36  0.374
15 Insufficiently_Active Hispanic                      1  3.21 NA
16 Inactive              White non-Hispanic          225  3.36  0.238
17 Inactive              Black non-Hispanic           83  3.43  0.247
18 Inactive              Other race non-Hispanic       4  3.32  0.238
19 Inactive              Multiracial non-Hispanic      5  3.37  0.129
20 Inactive              Hispanic                     14  3.30  0.264
```

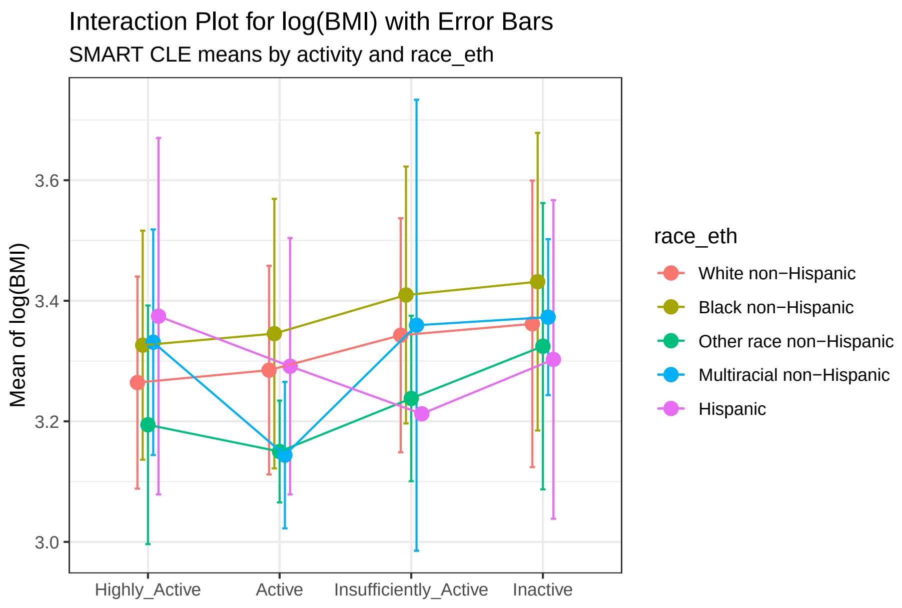
```
ggplot(summaries_5, aes(x = activity, y = mean,
                        color = race_eth,
                        group = race_eth)) +
  geom_point(size = 3) +
  geom_line() +
  labs(title = "Simple Interaction Plot for log(BMI)",
       subtitle = "SMART CLE means by activity and race_eth",
       x = "", y = "Mean of log(BMI)")
```



The interaction plot suggests that there is a modest interaction here. The White non-Hispanic and Black non-Hispanic groups appear pretty parallel (and they are the two largest groups) and Other race non-Hispanic has a fairly similar pattern, but the other two groups (Hispanic and Multiracial non-Hispanic) bounce around quite a bit based on activity level.

An alternative would be to include a small "dodge" for each point and include error bars (means ± standard deviation) for each combination.

```r
pd = position_dodge(0.2)
ggplot(summaries_5, aes(x = activity, y = mean,
                        color = race_eth,
                        group = race_eth)) +
  geom_errorbar(aes(ymin = mean - sd,
                    ymax = mean + sd),
                width = 0.2, position = pd) +
  geom_point(size = 3, position = pd) +
  geom_line(position = pd) +
  labs(title = "Interaction Plot for log(BMI) with Error Bars",
       subtitle = "SMART CLE means by activity and race_eth",
       x = "", y = "Mean of log(BMI)")
```



Here, we see a warning flag because we have one combination (which turns out to be Insufficiently Active and Hispanic) with only one observation in it, so a standard deviation cannot be calculated. In general, I'll stick with the simpler means plot most of the time.

5.3.2.2 Does the interaction account for substantial variation?

In this case, we can look at the fraction of the overall sums of squares accounted for by the interaction.

```
anova(model_5c)
```

```
Analysis of Variance Table

Response: log(bmi)
                    Df Sum Sq Mean Sq F value    Pr(>F)    
activity             3  2.060 0.68652 16.4468 1.839e-10 ***
race_eth             4  0.989 0.24716  5.9211 0.0001026 ***
activity:race_eth   12  0.324 0.02700  0.6469 0.8028368    
Residuals         1113 46.459 0.04174                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here we have

$$\eta^2(Interaction) = \frac{0.324}{2.058 + 0.990 + 0.324 + 46.456} = 0.0065$$

so the interaction accounts for 0.65% of the variation in `bmi`. That looks pretty modest.

5.3.2.3 Does the interaction account for statistically detectable variation?

We can test this directly with the p value from the ANOVA table, which shows p = 0.803, which is far above any of our usual standards for a statistically detectable effect.

On the whole, I don't think the interaction term is especially helpful in improving this model.

In the next chapter, we'll look at two different examples of ANOVA models, now in more designed experiments. We'll also add some additional details on how the analyses might proceed.

We'll return to the SMART CLE data later in these Notes.

Chapter 6

Analysis of Variance

6.1 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
   <chr>  <chr>   <chr>    <dbl>
 1 R101   LED     B         12.8
 2 R102   Halogen B         22.2
 3 R103   Halogen B         24.6
 4 R104   LED     A         17
 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
```

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

```
10 R110   Halogen A         19.7
# ... with 70 more rows
```

6.2 A One-Factor Analysis of Variance

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

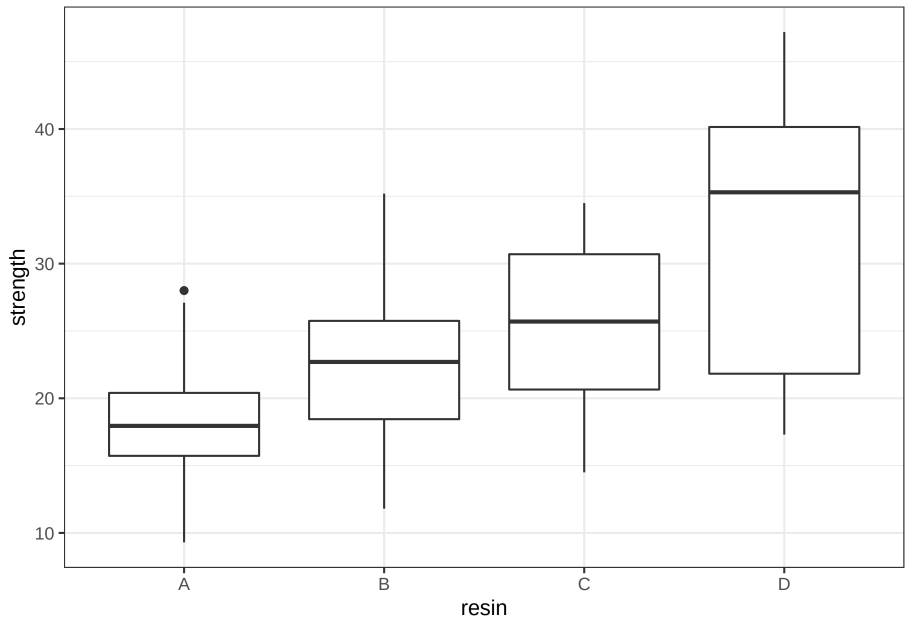
```
bonding %>% group_by(resin) %>% summarize(n = n(), mean(strength), median(strength))
```

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

6.2.1 Look at the Data!

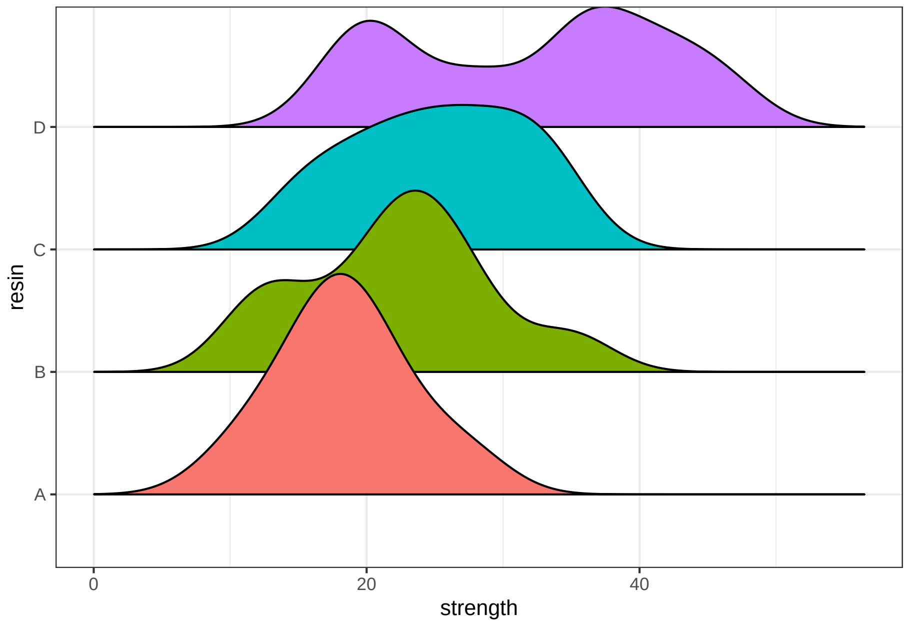
```
ggplot(bonding, aes(x = resin, y = strength)) +
    geom_boxplot()
```



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
```



6.2.2 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.

```
bonding %$% mosaic::favstats(strength ~ resin)
```

```
  resin  min     Q1 median    Q3  max   mean       sd  n missing
1     A  9.3 15.725  17.95 20.40 28.0 18.415 4.805948 20       0
2     B 11.8 18.450  22.70 25.75 35.2 22.230 6.748263 20       0
3     C 14.5 20.650  25.70 30.70 34.5 25.155 6.326425 20       0
4     D 17.3 21.825  35.30 40.15 47.2 32.075 9.735063 20       0
```

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
```

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

```
          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
```

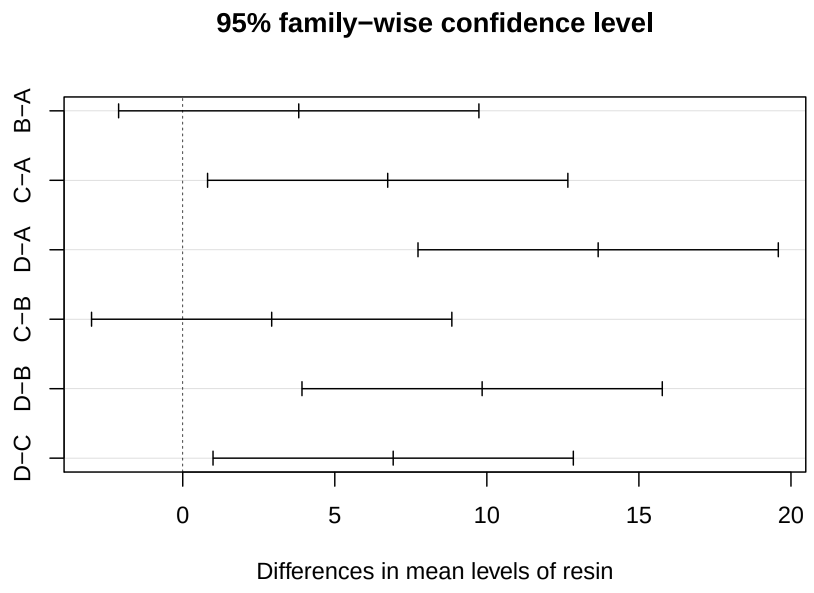
It appears that the `resin` types have a significant association with mean `strength` of the bonds. Can we identify which `resin` types have generally higher or lower `strength`?

```
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.

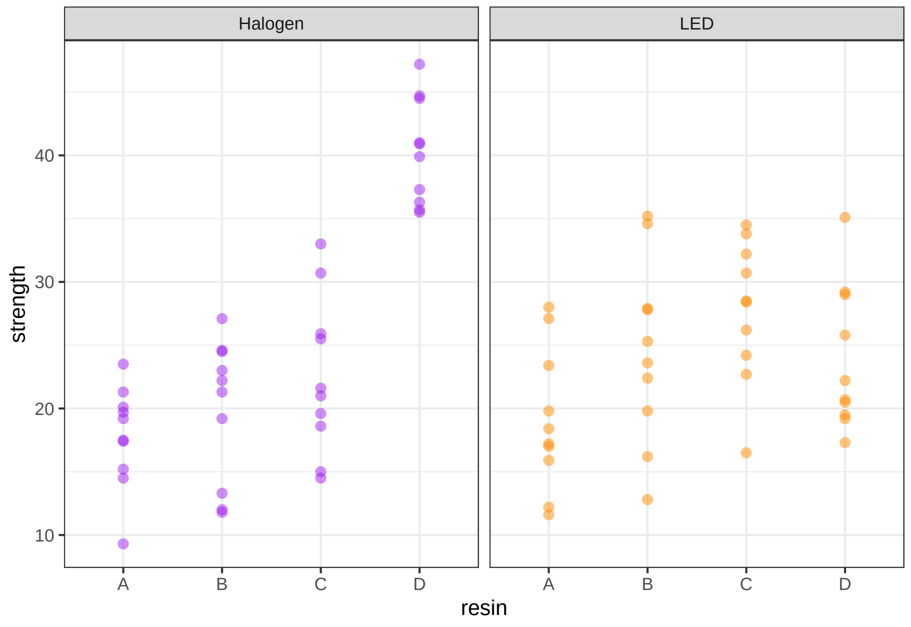
```
plot(TukeyHSD(aov(lm(strength ~ resin, data = bonding))))
```



6.3 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()
```



6.4 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))
```

```
`summarise()` has grouped output by 'resin'. You can override using the `.groups` argument.
```

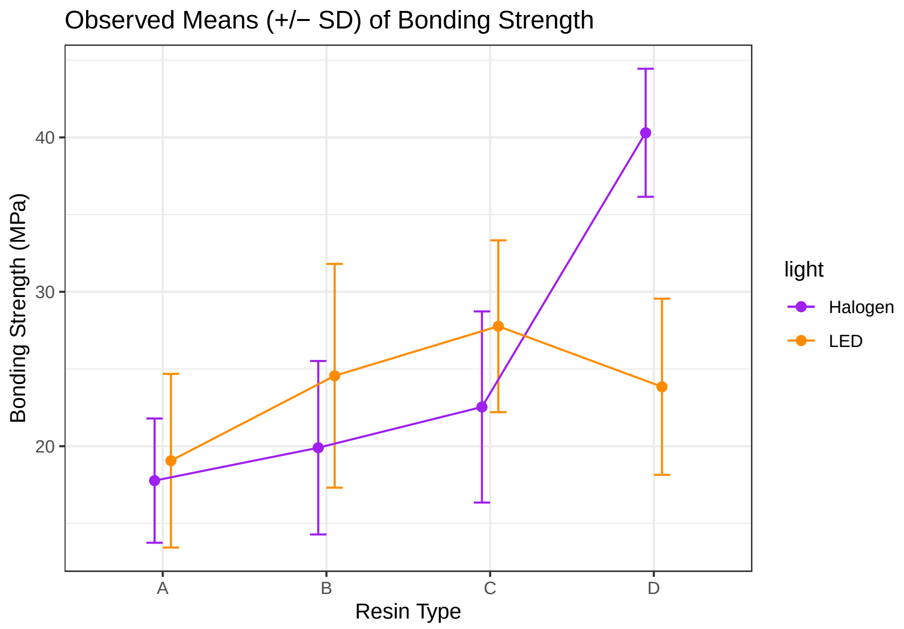
```
bond.sum
```

```
# A tibble: 8 x 4
# Groups:   resin [4]
  resin light   mean.str sd.str
  <chr> <chr>      <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.

6.4.1 Summarizing 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 %$% mosaic::favstats(strength ~ resin + light) %>%
    select(resin.light, median, mean, sd, n, missing)
```

```
  resin.light median  mean       sd  n missing
1   A.Halogen  18.35 17.77 4.024108 10       0
2   B.Halogen  21.75 19.90 5.617631 10       0
3   C.Halogen  21.30 22.54 6.191069 10       0
4   D.Halogen  40.40 40.30 4.147556 10       0
5       A.LED  17.80 19.06 5.625181 10       0
6       B.LED  24.45 24.56 7.246792 10       0
7       C.LED  28.45 27.77 5.564980 10       0
8       D.LED  21.45 23.85 5.704043 10       0
```

6.5 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
```

6.5.1 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
```

6.5.2 Is the interaction important?

In this case, the interaction:

- is evident in the means plot, and
- is highly statistically significant, and
- accounts for a sizable 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.

6.5.3 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:

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

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

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

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

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

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.

6.6 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.

6.7 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:

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

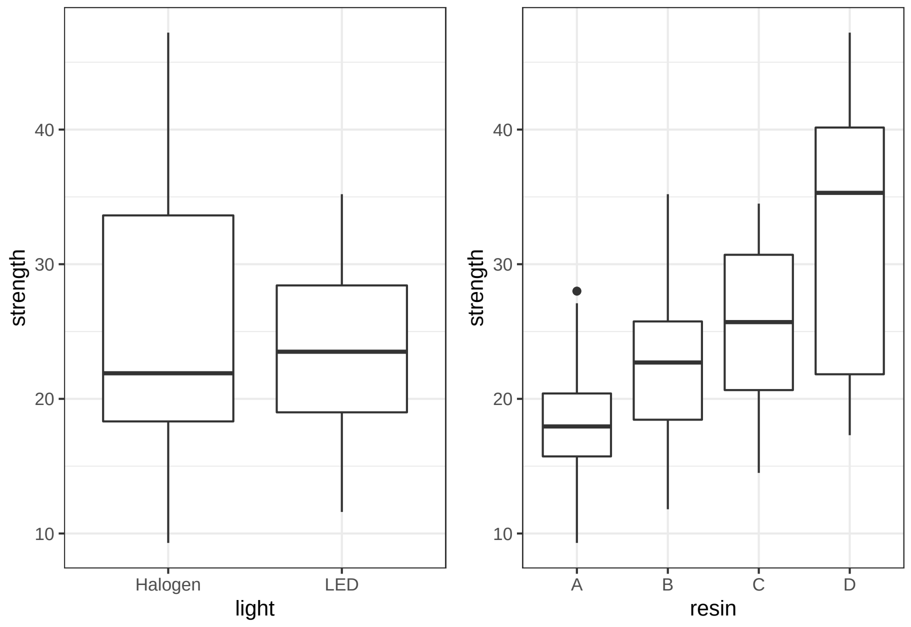
So, in the no-interaction model, only the intercept changes.

```r
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.

```r
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)
```



6.8 `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.

6.8.1 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)
`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
     <dbl> <chr>  <dbl> <chr>  <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   M       14.4   12.7
 4    1004 Low     44.9 M        9      9.8
 5    1005 High    46.1 M       14.2   14.2
 6    1006 UC      41.3 M       14.8   15.1
 7    1007 Low     51   F       13.7   16
 8    1008 UC      42   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
```

6.9 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           
 Min.   :1001   Length:156         Min.   :20.80   Length:156        
 1st Qu.:1040   Class :character   1st Qu.:33.27   Class :character  
 Median :1078   Mode  :character   Median :40.35   Mode  :character  
 Mean   :1078                      Mean   :40.42                     
 3rd Qu.:1117                      3rd Qu.:47.77                     
 Max.   :1156                      Max.   :59.90                     
     cort.1           cort.5          fat_est      cort_diff      
 Min.   : 6.000   Min.   : 4.2   healthy  : 56   Min.   :-2.3000  
 1st Qu.: 9.675   1st Qu.: 9.6   unhealthy:100   1st Qu.:-0.5000  
 Median :12.400   Median :12.6                   Median : 0.2000  
 Mean   :12.686   Mean   :12.4                   Mean   : 0.2821  
 3rd Qu.:16.025   3rd Qu.:15.7                   3rd Qu.: 1.2000  
 Max.   :19.000   Max.   :19.7                   Max.   : 2.0000  
```

6.10 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()))
```

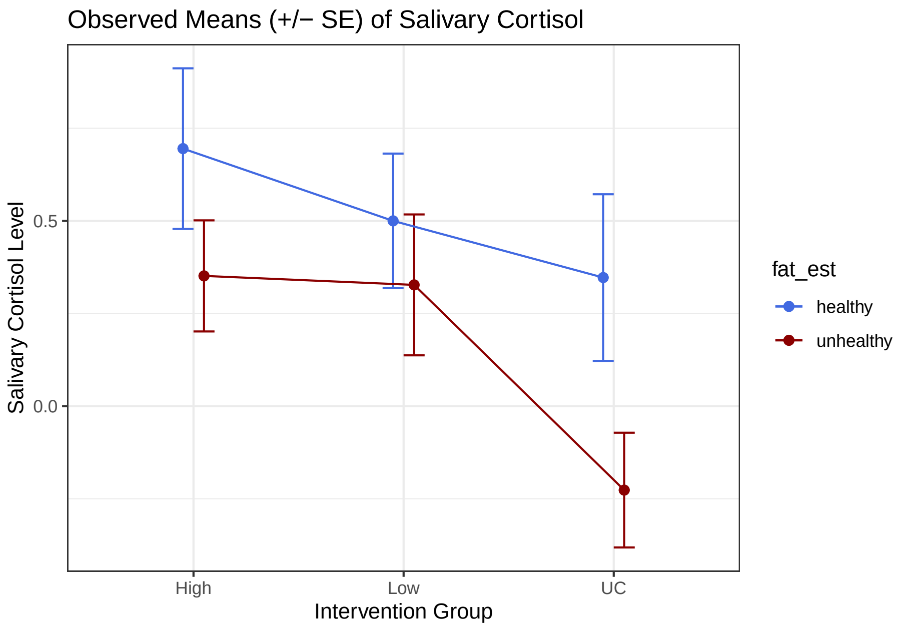
```
`summarise()` has grouped output by 'interv'. You can override using the `.groups` argument.
```

```
cort.sum
```

```
# A tibble: 6 x 4
# Groups:   interv [3]
  interv fat_est   mean.cort se.cort
  <chr>  <fct>         <dbl>   <dbl>
1 High   healthy       0.695   0.217
2 High   unhealthy     0.352   0.150
3 Low    healthy       0.5     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")
```



6.11 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
```

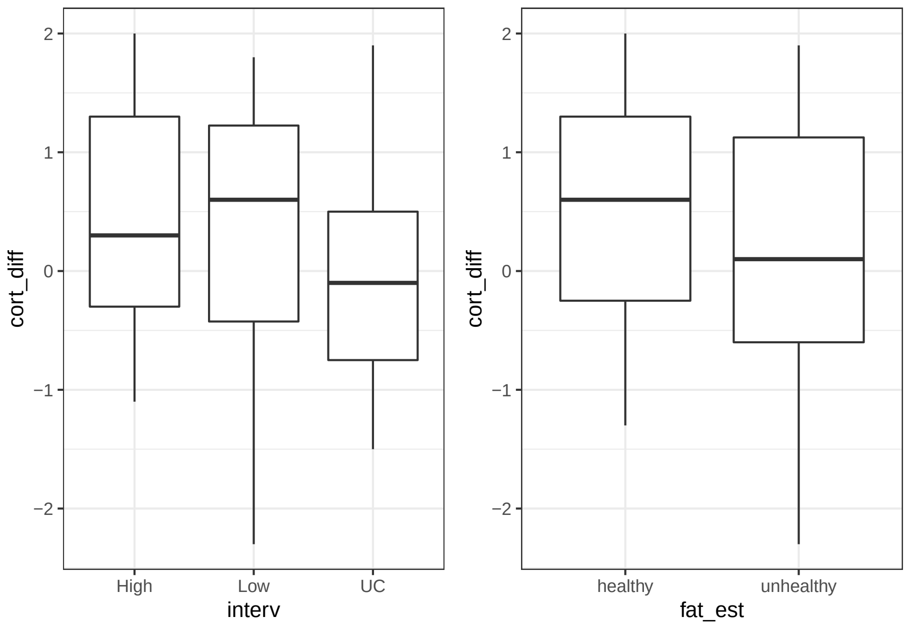
How do you reconcile the apparent difference in significance levels between this regression summary and the ANOVA table above?

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

6.12.1 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()
```

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



6.12.2 The ANOVA Model

```r
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?

6.12.3 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
```

6.12.4 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 7

Analysis of Covariance

7.1 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.

7.1.1 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
     <dbl> <dbl> <chr>   <dbl>   <dbl>    <dbl>    <dbl>
 1       1    61 F        14.1     2.3     10.3  -11.8
 2       2    70 F         7.2     5.4      7.3   -1.8
 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         9.9    10.7     11.7    0.800
 6       6    76 M         5.2     6.8      4.2    1.60
 7       7    72 M        10.4    14.6     14.1    4.20
 8       8    69 F        10.5     7.2      5.4   -3.3
 9       9    66 M         5       5        5.1    0
10      10    62 M         8.6     8.1      7.4   -0.5
11      11    65 F        16.6    14.9     13     -1.7
12      12    71 M        16.4    18.6     17.1    2.2
13      13    51 F        12.2    11       12.3   -1.20
14      14    71 M         6.6     3.7      4.5   -2.90
15      15    64 F        15.4    15.2     13.6   -0.2
16      16    50 M        10.2    10.8     11.2    0.6
```

7.2 Does sex affect the mean change in theophylline?

```
emphysema %$% mosaic::favstats(st_delta)
```

```
   min     Q1 median   Q3 max     mean       sd  n missing
 -11.8 -1.925  -0.35 0.65 4.2 -0.99375 3.484149 16       0
```

```
emphysema %$% mosaic::favstats(st_delta ~ sex)
```

```
  sex   min     Q1 median     Q3  max      mean       sd  n missing
1   F -11.8 -2.925  -1.75 -1.325 -0.2 -3.333333 4.267864  6       0
2   M  -2.9 -0.375   0.50  1.400  4.2  0.410000 2.067446 10       0
```

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?

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

```
emphysema %$% mosaic::favstats(age ~ sex)
```

```
  sex min    Q1 median Q3 max     mean       sd n missing
1   F  51 61.75   64.5 68  70 63.33333 6.889606 6       0
2   M  50 65.00   66.0 71  76 66.44444 7.568208 9       1
```

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)?

7.4 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**.

7.4.1 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.

7.4.2 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.

7.5 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
     <dbl> <dbl> <chr>   <dbl>   <dbl>    <dbl>    <dbl>
 1       1  61   F        14.1     2.3     10.3  -11.8  
 2       2  70   F         7.2     5.4      7.3   -1.8  
 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  65.2 M         9.9    10.7     11.7    0.800
 6       6  76   M         5.2     6.8      4.2    1.60 
 7       7  72   M        10.4    14.6     14.1    4.20 
 8       8  69   F        10.5     7.2      5.4   -3.3  
 9       9  66   M         5       5        5.1    0    
10      10  62   M         8.6     8.1      7.4   -0.5  
11      11  65   F        16.6    14.9     13     -1.7  
12      12  71   M        16.4    18.6     17.1    2.2  
13      13  51   F        12.2    11       12.3   -1.20 
14      14  71   M         6.6     3.7      4.5   -2.90 
15      15  64   F        15.4    15.2     13.6   -0.2  
16      16  50   M        10.2    10.8     11.2    0.6  
```

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.

7.6 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))
```

```
# A tibble: 4 x 5
  term        estimate std.error statistic p.value
  <chr>          <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)  -5.65      13.5      -0.420   0.682
2 sexM          1.72      16.8       0.102   0.920
3 age           0.0365     0.211     0.173   0.866
4 sexM:age      0.0289     0.260     0.111   0.914
```

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.

7.7 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 %$% mosaic::favstats(age)
```

```
 min    Q1 median    Q3 max mean       sd  n missing
  50 63.5   65.1 70.25  76 65.2 6.978061 16       0
```

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))
```

```
# A tibble: 4 x 5
  term        estimate std.error statistic p.value
  <chr>          <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)   -3.27       1.39    -2.35   0.0364
2 sexM           3.60       1.74     2.08   0.0601
3 age_z          0.510      2.95     0.173  0.866
4 sexM:age_z     0.403      3.63     0.111  0.914
```

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 8

Analysis of Covariance with the SMART data

In this chapter, we'll work with the `smart_cle1_sh` data file again.

```
smart_cle1_sh <- readRDS(here("data", "smart_cle1_sh.Rds"))
```

8.1 A New Small Study: Predicting BMI

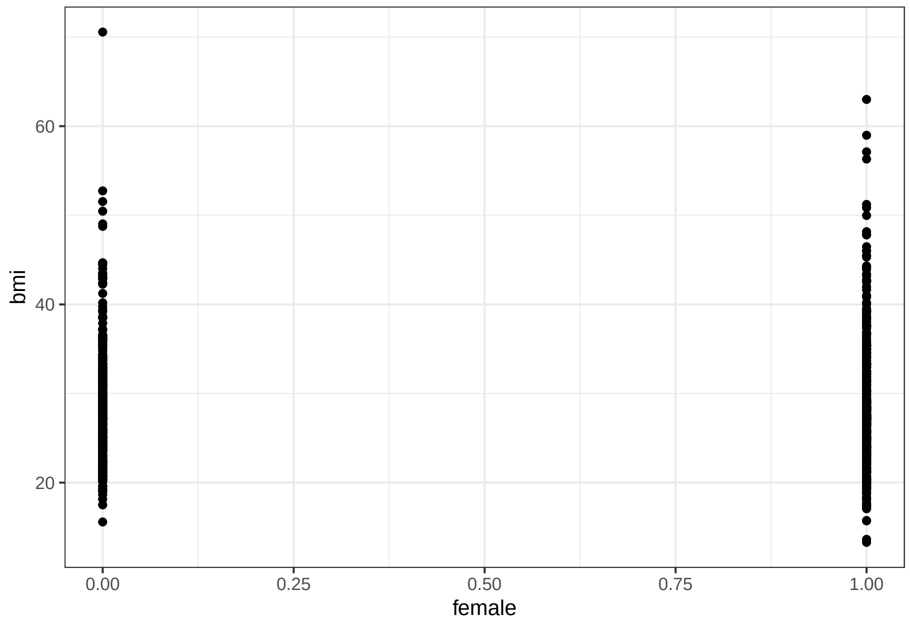
We'll begin by investigating the problem of predicting `bmi`, at first with just three regression inputs: `sex`, `smoke100` and `physhealth`, in our `smart_cle1_sh` data set.

- 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
  - `smoke100` = 1 if the subject has smoked 100 cigarettes in their lifetime
  - `physhealth` = number of poor physical health days in past 30 (treated as quantitative)

8.1.1 Does `female` predict `bmi` well?

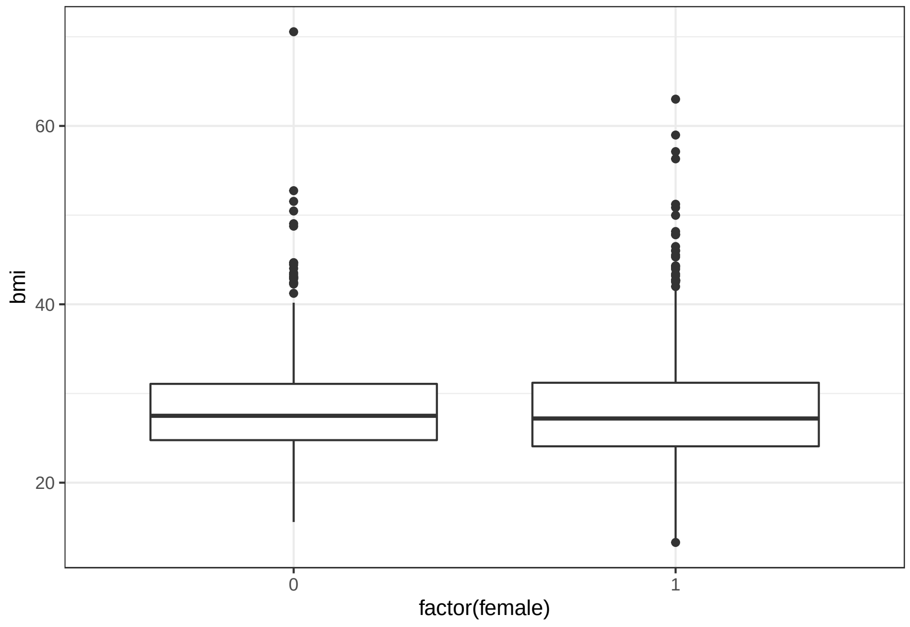
8.1.1.1 Graphical Assessment

```
ggplot(smart_cle1_sh, aes(x = female, y = bmi)) +
    geom_point()
```



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

```r
ggplot(smart_cle1_sh, 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.

8.2 c8_m1: A simple t-test model

```
c8_m1 <- lm(bmi ~ female, data = smart_cle1_sh)
c8_m1
```

```

Call:
lm(formula = bmi ~ female, data = smart_cle1_sh)

Coefficients:
(Intercept)       female
    28.4941      -0.2442
```

```
summary(c8_m1)
```

```

Call:
lm(formula = bmi ~ female, data = smart_cle1_sh)

Residuals:
    Min      1Q  Median      3Q     Max
-14.950  -4.060  -1.024   2.740  42.066

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  28.4941     0.2965  96.090   <2e-16 ***
female       -0.2442     0.3850  -0.634    0.526
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.367 on 1131 degrees of freedom
Multiple R-squared:  0.0003554, Adjusted R-squared:  -0.0005284
F-statistic: 0.4021 on 1 and 1131 DF,  p-value: 0.5261
```

```
confint(c8_m1)
```

```
                 2.5 %     97.5 %
(Intercept) 27.9123220 29.0759609
female      -0.9996392  0.5113054
```

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.

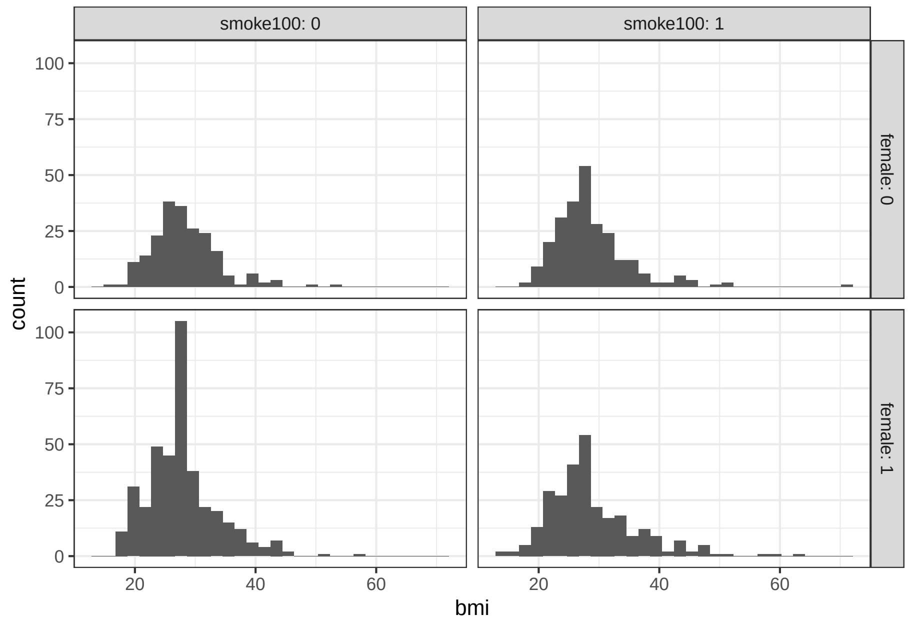
```
t.test(bmi ~ female, var.equal = TRUE, data = smart_cle1_sh)
```

```

    Two Sample t-test

data:  bmi by female
t = 0.63413, df = 1131, p-value = 0.5261
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.5113054  0.9996392
sample estimates:
mean in group 0 mean in group 1 
       28.49414        28.24997 
```

8.3 c8_m2: Adding another predictor (two-way ANOVA without interaction)

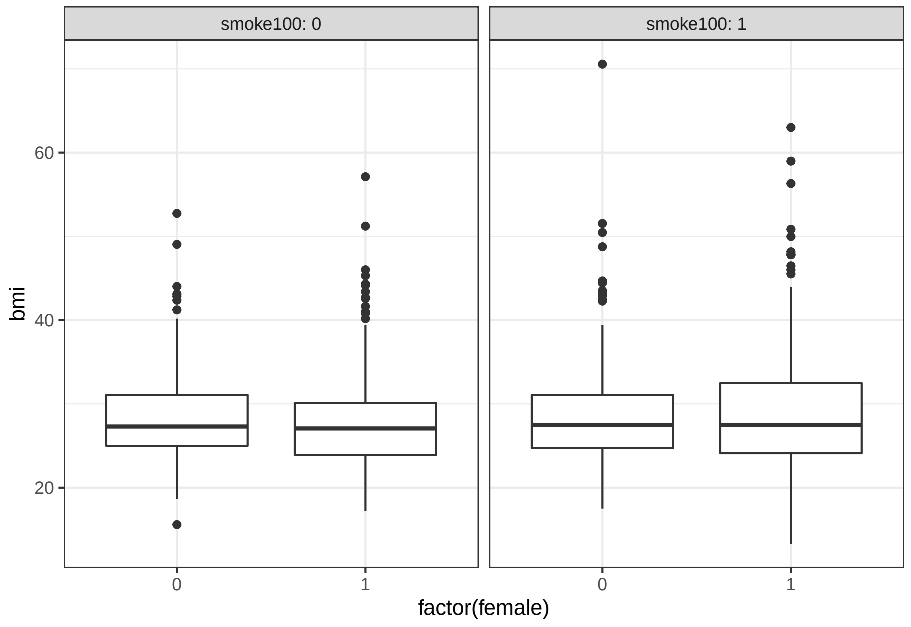
When we add in the information about `smoke100` to our original model, we might first picture the data. We could look at separate histograms,

```
ggplot(smart_cle1_sh, aes(x = bmi)) +
    geom_histogram(bins = 30) +
    facet_grid(female ~ smoke100, labeller = label_both)
```

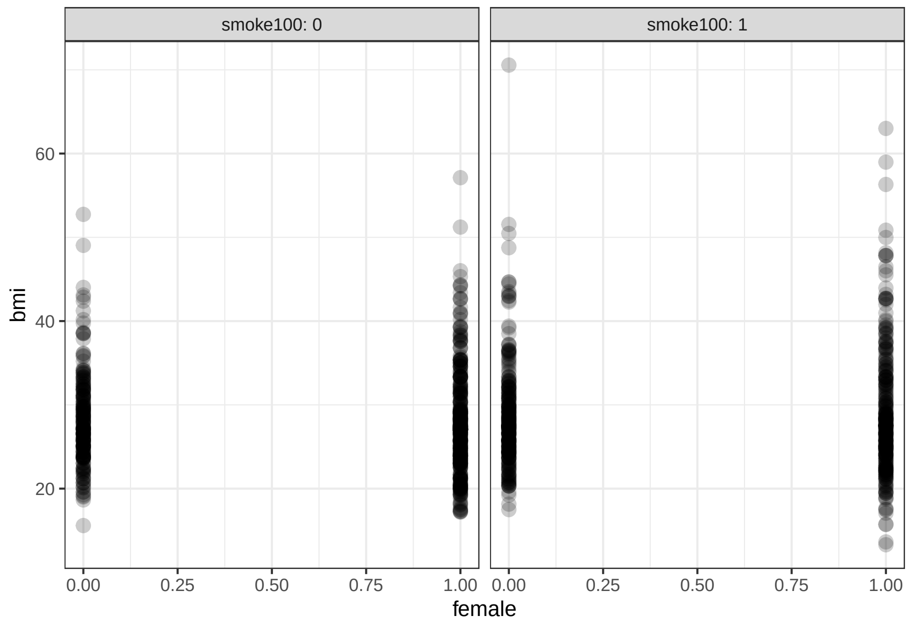


or maybe boxplots?

```
ggplot(smart_cle1_sh, aes(x = factor(female), y = bmi)) +
    geom_boxplot() +
    facet_wrap(~ smoke100, labeller = label_both)
```



```r
ggplot(smart_cle1_sh, aes(x = female, y = bmi))+
    geom_point(size = 3, alpha = 0.2) +
    theme_bw() +
    facet_wrap(~ smoke100, labeller = label_both)
```



OK. Let's try fitting a model.

```r
c8_m2 <- lm(bmi ~ female + smoke100, data = smart_cle1_sh)
c8_m2
```

```

Call:
lm(formula = bmi ~ female + smoke100, data = smart_cle1_sh)

Coefficients:
(Intercept)       female     smoke100
    28.0265      -0.1342       0.8555
```

This new model predicts only four predicted values:

- `bmi` = 28.035 if the subject is male and has not smoked 100 cigarettes (so `female` = 0 and `smoke100` = 0)
- `bmi` = 28.035 - 0.144 = 27.891 if the subject is female and has not smoked 100 cigarettes (`female` = 1 and `smoke100` = 0)
- `bmi` = 28.035 + 0.859 = 28.894 if the subject is male and has smoked 100 cigarettes (so `female` = 0 and `smoke100` = 1), and, finally
- `bmi` = 28.035 - 0.144 + 0.859 = 28.750 if the subject is female and has smoked 100 cigarettes (so both `female` and `smoke100` = 1).

Another way to put this is that for those who have not smoked 100 cigarettes, the model is:

- `bmi` = 28.035 - 0.144 `female`

and for those who have smoked 100 cigarettes, the model is:

- `bmi` = 28.894 - 0.144 `female`

Only the intercept of the `bmi-female` model changes depending on `smoke100`.

```
summary(c8_m2)
```

```
Call:
lm(formula = bmi ~ female + smoke100, data = smart_cle1_sh)

Residuals:
    Min      1Q  Median      3Q     Max 
-15.448  -3.972  -0.823   2.774  41.678 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  28.0265     0.3620  77.411   <2e-16 ***
female       -0.1342     0.3875  -0.346   0.7291    
smoke100      0.8555     0.3814   2.243   0.0251 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.356 on 1130 degrees of freedom
Multiple R-squared:  0.004788,	Adjusted R-squared:  0.003027 
F-statistic: 2.718 on 2 and 1130 DF,  p-value: 0.06642
```

```
confint(c8_m2)
```

```
                 2.5 %     97.5 %
(Intercept) 27.3161140 28.7368281
female      -0.8944773  0.6259881
smoke100     0.1072974  1.6037825
```

The slopes of both `female` and `smoke100` have confidence intervals that are completely below zero, indicating that both `female` sex and `smoke100` 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(c8_m2)
```

```
Analysis of Variance Table

Response: bmi
            Df Sum Sq Mean Sq F value  Pr(>F)
female       1     16  16.301  0.4036 0.52538
smoke100     1    203 203.296  5.0330 0.02506 *
Residuals 1130  45644  40.393
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

8.4 c8_m3: Adding the interaction term (Two-way ANOVA with interaction)

Suppose we want to let the effect of `female` vary depending on the `smoke100` status. Then we need to incorporate an interaction term in our model.

```
c8_m3 <- lm(bmi ~ female * smoke100, data = smart_cle1_sh)
c8_m3
```

```
Call:
lm(formula = bmi ~ female * smoke100, data = smart_cle1_sh)

Coefficients:
    (Intercept)           female         smoke100  female:smoke100
        28.2690          -0.5064           0.4119           0.7536
```

So, for example, for a male who has smoked 100 cigarettes, this model predicts

- `bmi` = 28.275 - 0.513 (0) + 0.419 (1) + 0.746 (0)(1) = 28.275 + 0.419 = 28.694

And for a female who has smoked 100 cigarettes, the model predicts

- `bmi` = 28.275 - 0.513 (1) + 0.419 (1) + 0.746 (1)(1) = 28.275 - 0.513 + 0.419 + 0.746 = 28.927

For those who have not smoked 100 cigarettes, the model is:

- `bmi` = 28.275 - 0.513 `female`

But for those who have smoked 100 cigarettes, the model is:

- `bmi` = (28.275 + 0.419) + (-0.513 + 0.746) `female`, or „,
- `bmi` = 28.694 - 0.233 `female`

Now, both the slope and the intercept of the `bmi-female` model change depending on `smoke100`.

```
summary(c8_m3)
```

```
Call:
lm(formula = bmi ~ female * smoke100, data = smart_cle1_sh)

Residuals:
    Min      1Q  Median      3Q     Max 
-15.628  -3.938  -0.829   2.759  41.879 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)      28.2690     0.4396  64.301   <2e-16 ***
female           -0.5064     0.5446  -0.930    0.353    
smoke100          0.4119     0.5946   0.693    0.489    
female:smoke100   0.7536     0.7750   0.972    0.331    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.356 on 1129 degrees of freedom
Multiple R-squared:  0.005621,  Adjusted R-squared:  0.002979 
F-statistic: 2.127 on 3 and 1129 DF,  p-value: 0.09507
```

```
confint(c8_m3)
```

```
                     2.5 %     97.5 %
(Intercept)     27.4063783 29.1315563
female          -1.5749026  0.5621793
smoke100        -0.7547605  1.5786121
female:smoke100 -0.7670239  2.2742178
```

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(c8_m3)
```

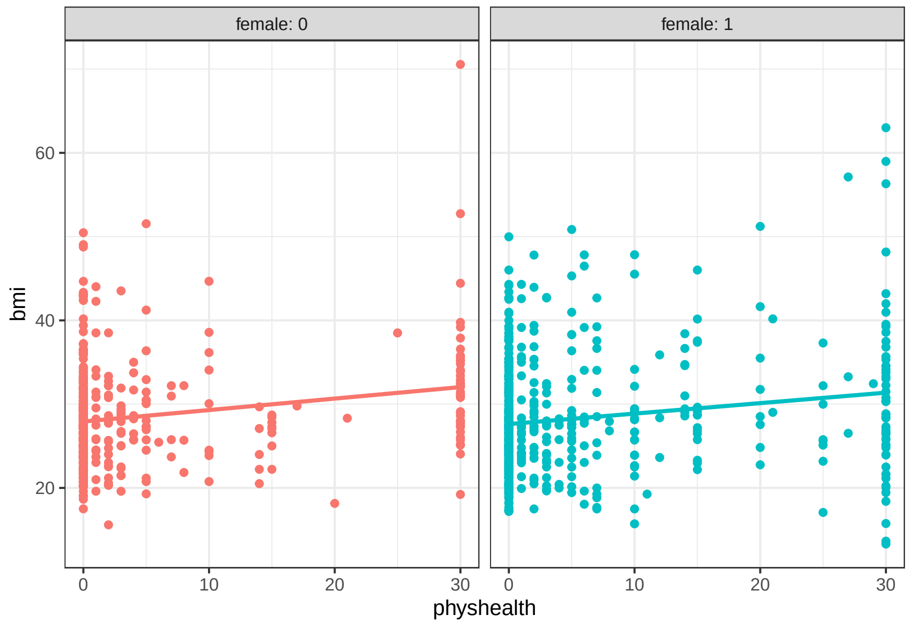
```
Analysis of Variance Table

Response: bmi
                  Df Sum Sq Mean Sq F value  Pr(>F)  
female             1     16  16.301  0.4035 0.52539  
smoke100           1    203 203.296  5.0327 0.02507 *
female:smoke100    1     38  38.194  0.9455 0.33107  
Residuals       1129  45606  40.395                  
---
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.

8.5 c8_m4: Using female and physhealth in a model for bmi

```
ggplot(smart_cle1_sh, aes(x = physhealth, y = bmi, color = factor(female))) +
    geom_point() +
    guides(col = FALSE) +
    geom_smooth(method = "lm", se = FALSE) +
    facet_wrap(~ female, labeller = label_both)
```

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



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

```
c8_m4 <- lm(bmi ~ female * physhealth, data = smart_cle1_sh)

summary(c8_m4)
```

```
Call:
lm(formula = bmi ~ female * physhealth, data = smart_cle1_sh)

Residuals:
    Min      1Q  Median      3Q     Max
-18.069  -3.825  -0.624   2.516  38.526

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       27.92386    0.32196  86.731  < 2e-16 ***
female            -0.30335    0.42346  -0.716    0.474
physhealth         0.13700    0.03277   4.180 3.14e-05 ***
female:physhealth -0.01203    0.04191  -0.287    0.774
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.262 on 1129 degrees of freedom
Multiple R-squared:  0.03486,   Adjusted R-squared:  0.03229
F-statistic: 13.59 on 3 and 1129 DF,  p-value: 1.027e-08
```

Does it seem as though the addition of `physhealth` has improved our model substantially over a model with `female` alone (which, you recall, was `c8_m1`)?

Since the `c8_m4` model contains the `c8_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(c8_m4)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
1    0.0349        0.0323  6.26      13.6 1.03e-8     3 -3684. 7378. 7403.
# ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
glance(c8_m1)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
1  0.000355     -0.000528  6.37     0.402   0.526     1 -3704. 7414. 7429.
# ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

- The R^2 is much larger for the model with `physhealth`, but still very tiny.
- Smaller AIC and smaller BIC statistics are more desirable. Here, there's little to choose from, so `c8_m4` looks better, too.
- We might also consider a significance test by looking at an ANOVA model comparison. This is only appropriate because `c8_m1` is nested in `c8_m4`.

```
anova(c8_m4, c8_m1)
```

```
Analysis of Variance Table
```

```
Model 1: bmi ~ female * physhealth
Model 2: bmi ~ female
  Res.Df   RSS Df Sum of Sq     F    Pr(>F)
1   1129 44265
2   1131 45847 -2   -1582.4 20.18 2.448e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The addition of the `physhealth` term appears to be a statistically detectable improvement, not that that means very much.

8.6 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 days of poor physical health (`physhealth`) and their interaction.

```
c8_m4
```

```

Call:
lm(formula = bmi ~ female * physhealth, data = smart_cle1_sh)

Coefficients:
      (Intercept)             female         physhealth  female:physhealth
         27.92386           -0.30335            0.13700           -0.01203
```

8.6.1 Fitting an Individual Prediction and 95% Prediction Interval

What do we predict for the `bmi` of a subject who is `female` and had 8 poor physical health days in the past 30?

```
c8_new1 <- tibble(female = 1, physhealth = 8)
predict(c8_m4, newdata = c8_new1, interval = "prediction", level = 0.95)
```

```
       fit      lwr     upr
1 28.62022 16.32454 40.9159
```

The predicted `bmi` for this new subject is shown above. The prediction interval shows the bounds of a 95% uncertainty interval for a predicted `bmi` for an individual female subject who has 8 days of poor physical health out of the past 30. From the `predict` function applied to a linear model, we can get the prediction intervals for any new data points in this manner.

8.6.2 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 have `physhealth = 8`?

```
predict(c8_m4, newdata = c8_new1, interval = "confidence", level = 0.95)
```

```
       fit      lwr      upr
1 28.62022 28.12256 29.11788
```

- How does this result compare to the prediction interval?

8.6.3 Fitting Multiple Individual Predictions to New Data

- How does our prediction change for a respondent if they instead have 7, or 9 poor physical health days? What if they are male, instead of female?

```
c8_new2 <- tibble(subjectid = 1001:1006, female = c(1, 1, 1, 0, 0, 0), physhealth = c(7, 8, 9, 7,
pred2 <- predict(c8_m4, newdata = c8_new2, interval = "prediction", level = 0.95) %>% tbl_df

result2 <- bind_cols(c8_new2, pred2)
result2
```

```
# A tibble: 6 x 6
  subjectid female physhealth   fit   lwr   upr
      <int>  <dbl>      <dbl> <dbl> <dbl> <dbl>
1      1001      1          7  28.5  16.2  40.8
2      1002      1          8  28.6  16.3  40.9
3      1003      1          9  28.7  16.4  41.0
4      1004      0          7  28.9  16.6  41.2
5      1005      0          8  29.0  16.7  41.3
6      1006      0          9  29.2  16.9  41.5
```

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 `physhealth`?

8.7 Centering the model

Our model `c8_m4` has four predictors (the constant, `physhealth`, `female` and their interaction) but just two inputs (`female` and `physhealth`.) If we **center** the quantitative input `physhealth` before building the model, we get a more interpretable interaction term.

```
smart_cle1_sh_c <- smart_cle1_sh %>%
    mutate(physhealth_c = physhealth - mean(physhealth))

c8_m4_c <- lm(bmi ~ female * physhealth_c, data = smart_cle1_sh_c)

summary(c8_m4_c)
```

```
Call:
lm(formula = bmi ~ female * physhealth_c, data = smart_cle1_sh_c)

Residuals:
    Min      1Q  Median      3Q     Max 
-18.069  -3.825  -0.624   2.516  38.526 

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)         28.56520    0.29213  97.784  < 2e-16 ***
female              -0.35969    0.37917  -0.949    0.343    
physhealth_c         0.13700    0.03277   4.180 3.14e-05 ***
female:physhealth_c -0.01203    0.04191  -0.287    0.774    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.262 on 1129 degrees of freedom
Multiple R-squared:  0.03486,	Adjusted R-squared:  0.03229 
F-statistic: 13.59 on 3 and 1129 DF,  p-value: 1.027e-08
```

What has changed as compared to the original `c8_m4`?

- Our original model was `bmi` = 27.93 - 0.31 `female` + 0.14 `physhealth` - 0.01 `female` x `physhealth`
- Our new model is `bmi` = 28.58 - 0.37 `female` + 0.14 centered `physhealth` - 0.01 `female` x centered `physhealth`.

So our new model on centered data is:

- 28.58 + 0.14 centered `physhealth_c` for male subjects, and
- (28.58 - 0.37) + (0.14 - 0.01) centered `physhealth_c`, or 28.21 - 0.13 centered `physhealth_c` for female subjects.

In our new (centered `physhealth_c`) model,

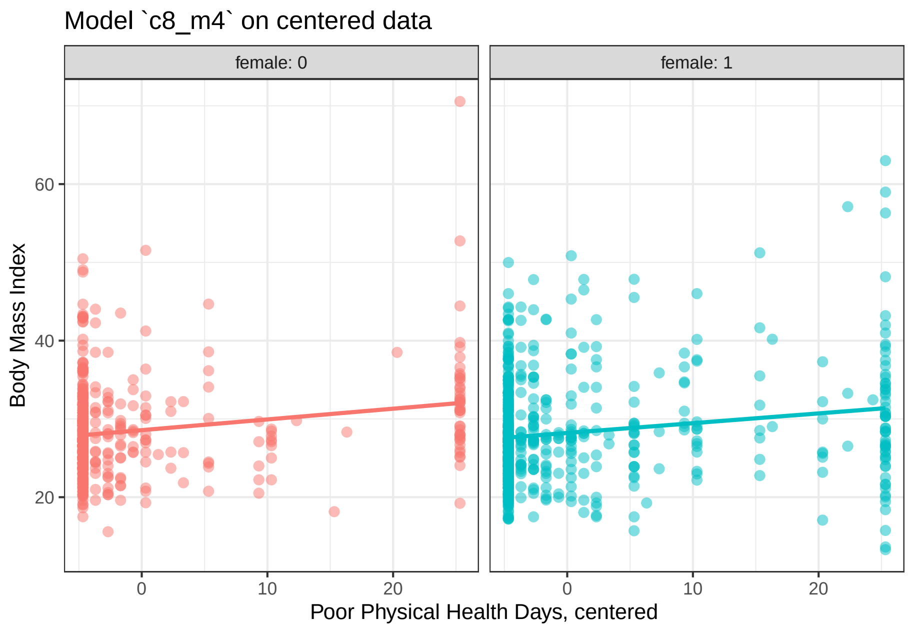
- the main effect of `female` now corresponds to a predictive difference (female - male) in `bmi` with `physhealth` at its mean value, 4.68 days,
- the intercept term is now the predicted `bmi` for a male respondent with an average `physhealth`, and
- the product term corresponds to the change in the slope of centered `physhealth_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.

8.7.1 Plot of Model 4 on Centered `physhealth`: `c8_m4_c`

```r
ggplot(smart_cle1_sh_c, aes(x = physhealth_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 = "Poor Physical Health Days, centered", y = "Body Mass Index",
         title = "Model `c8_m4` on centered data") +
    facet_wrap(~ female, labeller = label_both)
```

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



8.8 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 `physhealth`, but this is misleading, considering that we are comparing the complete change in one variable (sex = female or not) to a 1-day change in `physhealth`.
- 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.

```
smart_cle1_sh_rescale <- smart_cle1_sh %>%
    mutate(physhealth_z = (physhealth - mean(physhealth))/(2*sd(physhealth)))
```

8.8.1 Refitting model c8_m4 to the rescaled data

```
c8_m4_z <- lm(bmi ~ female * physhealth_z, data = smart_cle1_sh_rescale)

summary(c8_m4_z)
```

```
Call:
lm(formula = bmi ~ female * physhealth_z, data = smart_cle1_sh_rescale)

Residuals:
    Min      1Q  Median      3Q     Max 
-18.069  -3.825  -0.624   2.516  38.526 

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)          28.5652     0.2921  97.784  < 2e-16 ***
female               -0.3597     0.3792  -0.949    0.343    
physhealth_z          2.4991     0.5978   4.180 3.14e-05 ***
female:physhealth_z  -0.2195     0.7645  -0.287    0.774    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.262 on 1129 degrees of freedom
Multiple R-squared:  0.03486,   Adjusted R-squared:  0.03229 
F-statistic: 13.59 on 3 and 1129 DF,  p-value: 1.027e-08
```

8.8.2 Interpreting the model on rescaled data

What has changed as compared to the original `c8_m4`?

- Our original model was `bmi` = 27.93 - 0.31 `female` + 0.14 `physhealth` - 0.01 `female` x `physhealth`
- Our model on centered `physhealth` was `bmi` = 28.58 - 0.37 `female` + 0.14 centered `physhealth` - 0.01 `female` x centered `physhealth`.
- Our new model on rescaled `physhealth` is `bmi` = 28.58 - 0.37 `female` + 2.51 rescaled `physhealth_z` - 0.23 `female` x rescaled `physhealth_z`.

So our rescaled model is:

- 28.58 + 2.51 rescaled `physhealth_z` for male subjects, and
- (28.58 - 0.37) + (2.51 - 0.23) rescaled `physhealth_z`, or 28.21 + 2.28 rescaled `physhealth_z` for female subjects.

In this new rescaled (`physhealth_z`) model, then,

- the main effect of `female`, -0.37, still corresponds to a predictive difference (female - male) in `bmi` with `physhealth` at its mean value, 4.68 days,
- the intercept term is still the predicted `bmi` for a male respondent with an average `physhealth` count, and
- the residual standard deviation and the R-squared values remain unchanged,

as before, but now we also have that:

- the coefficient of `physhealth_z` indicates the predictive difference in `bmi` associated with a change in `physhealth` of 2 standard deviations (from one standard deviation below the mean of 4.68 to one standard deviation above 4.68.)
  - Since the standard deviation of `physhealth` is 9.12 (see below), this covers a massive range of potential values of `physhealth` from 0 all the way up to 4.68 + 2(9.12) = 22.92 days.

```
mosaic::favstats(~ physhealth, data = smart_cle1_sh)
```

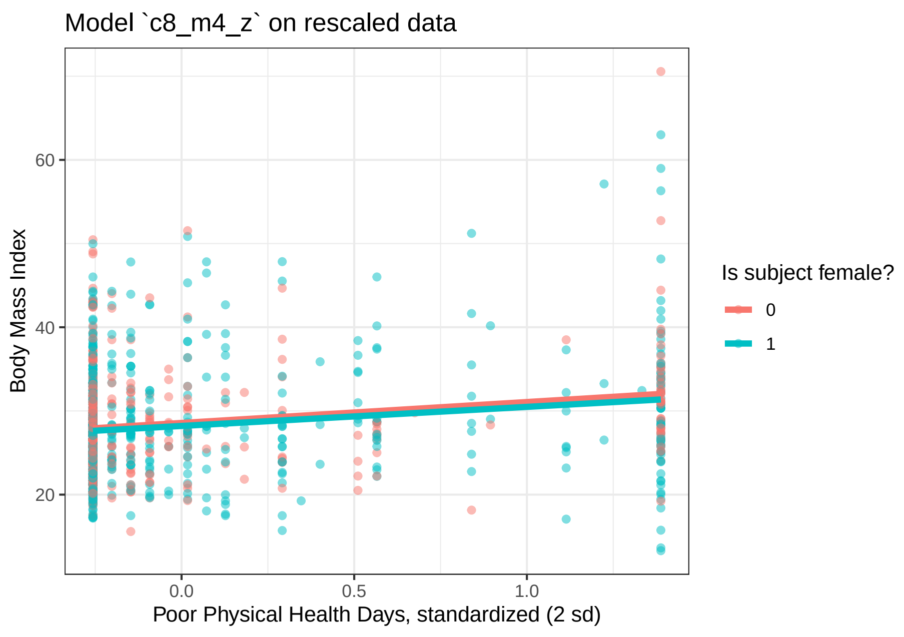
```
 min Q1 median Q3 max     mean       sd    n missing
   0  0      0  4  30 4.681377 9.120899 1133       0
```

- the coefficient of the product term (-0.23) corresponds to the change in the coefficient of `physhealth_z` for females as compared to males.

8.8.3 Plot of model on rescaled data

```
ggplot(smart_cle1_sh_rescale, aes(x = physhealth_z, y = bmi,
                              group = female, col = factor(female))) +
    geom_point(alpha = 0.5) +
    geom_smooth(method = "lm", se = FALSE, size = 1.5) +
    scale_color_discrete(name = "Is subject female?") +
    labs(x = "Poor Physical Health Days, standardized (2 sd)", y = "Body Mass Index",
         title = "Model `c8_m4_z` on rescaled data")
```

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



There's very little difference here.

8.9 c8_m5: What if we add more variables?

We can boost our R^2 a bit, to nearly 5%, by adding in two new variables, related to whether or not the subject (in the past 30 days) used the internet, and the average number of alcoholic drinks per week consumed by ths subject.

```
c8_m5 <- lm(bmi ~ female + smoke100 + physhealth + internet30 + drinks_wk,
         data = smart_cle1_sh)
summary(c8_m5)
```

```
Call:
lm(formula = bmi ~ female + smoke100 + physhealth + internet30 +
    drinks_wk, data = smart_cle1_sh)

Residuals:
    Min      1Q  Median      3Q     Max 
-18.358  -3.846  -0.657   2.534  38.049 
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 27.52400    0.56076  49.083  < 2e-16 ***
female      -0.43272    0.38510  -1.124  0.26140    
smoke100     0.82654    0.37739   2.190  0.02872 *  
physhealth   0.12469    0.02074   6.012 2.47e-09 ***
internet30   0.44287    0.48830   0.907  0.36462    
drinks_wk   -0.10193    0.03352  -3.041  0.00241 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.231 on 1127 degrees of freedom
Multiple R-squared:  0.04582,   Adjusted R-squared:  0.04159 
F-statistic: 10.82 on 5 and 1127 DF,  p-value: 3.48e-10
```

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

```
anova(c8_m5)
```

```
Analysis of Variance Table

Response: bmi
             Df Sum Sq Mean Sq F value    Pr(>F)    
female        1     16   16.30  0.4198  0.517171    
smoke100      1    203  203.30  5.2354  0.022316 *  
physhealth    1   1508 1508.08 38.8372 6.497e-10 ***
internet30    1     15   14.69  0.3783  0.538650    
drinks_wk     1    359  359.05  9.2466  0.002414 ** 
Residuals  1127  43762   38.83                      
---
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 ~ smoke100 + internet30 + drinks_wk + female + physhealth,
         data = smart_cle1_sh))
```

```
Analysis of Variance Table

Response: bmi
             Df Sum Sq Mean Sq F value    Pr(>F)    
smoke100      1    215  214.75  5.5304 0.0188606 *  
internet30    1      8    7.81  0.2010 0.6539723    
drinks_wk     1    444  443.79 11.4288 0.0007479 ***
female        1     32   31.58  0.8132 0.3673566    
physhealth    1   1403 1403.49 36.1438 2.472e-09 ***
Residuals  1127  43762   38.83                      
```

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

3. What does the output below let us conclude?

```
anova(lm(bmi ~ smoke100 + internet30 + drinks_wk + female + physhealth,
         data = smart_cle1_sh),
      lm(bmi ~ smoke100 + female + drinks_wk,
         data = smart_cle1_sh))
```

```
Analysis of Variance Table

Model 1: bmi ~ smoke100 + internet30 + drinks_wk + female + physhealth
Model 2: bmi ~ smoke100 + female + drinks_wk
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1   1127 43762
2   1129 45166 -2   -1403.7 18.075 1.877e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

8.10 c8_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.

```
c8_m6 <- lm(bmi ~ female + smoke100 + physhealth + internet30 + drinks_wk + genhealth,
         data = smart_cle1_sh)
summary(c8_m6)
```

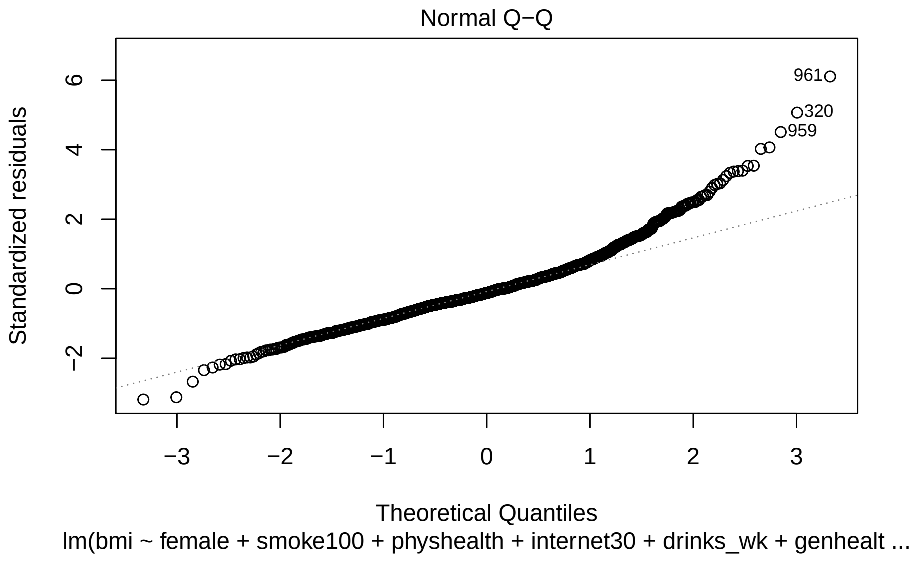
```
Call:
lm(formula = bmi ~ female + smoke100 + physhealth + internet30 +
    drinks_wk + genhealth, data = smart_cle1_sh)

Residuals:
    Min      1Q  Median      3Q     Max
-19.216  -3.659  -0.736   2.669  36.810

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         25.20736    0.71106  35.450  < 2e-16 ***
female              -0.31949    0.37667  -0.848   0.3965
smoke100             0.45866    0.37214   1.232   0.2180
physhealth           0.04353    0.02506   1.737   0.0827 .
```

```
internet30           0.93270    0.48273   1.932   0.0536 .
drinks_wk           -0.07712    0.03294  -2.341   0.0194 *
genhealth2_VeryGood  1.21169    0.56838   2.132   0.0332 *
genhealth3_Good      3.22783    0.58009   5.564 3.29e-08 ***
genhealth4_Fair      4.14497    0.73284   5.656 1.96e-08 ***
genhealth5_Poor      5.86335    1.09253   5.367 9.73e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.089 on 1123 degrees of freedom
Multiple R-squared:  0.09206,   Adjusted R-squared:  0.08478
F-statistic: 12.65 on 9 and 1123 DF,  p-value: < 2.2e-16
```

1. If Harry and Marty have the same values of `female`, `smoke100`, `physhealth`, `internet30` and `drinks_wk`, 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(c8_m6, which = 2)
```



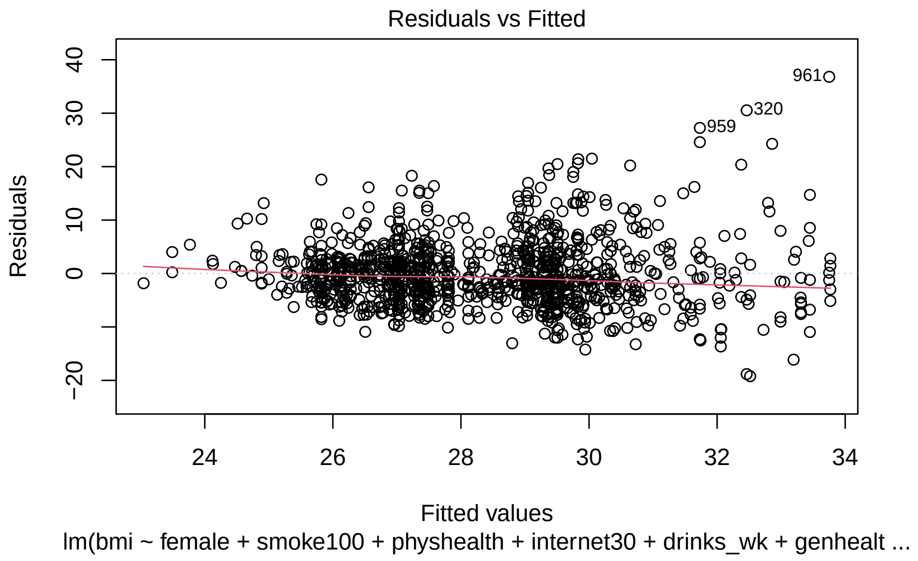
8.11 Key Regression Assumptions for Building Effective Prediction Models

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 parameters 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

8.11.1 Checking Assumptions in model `c8_m6`

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

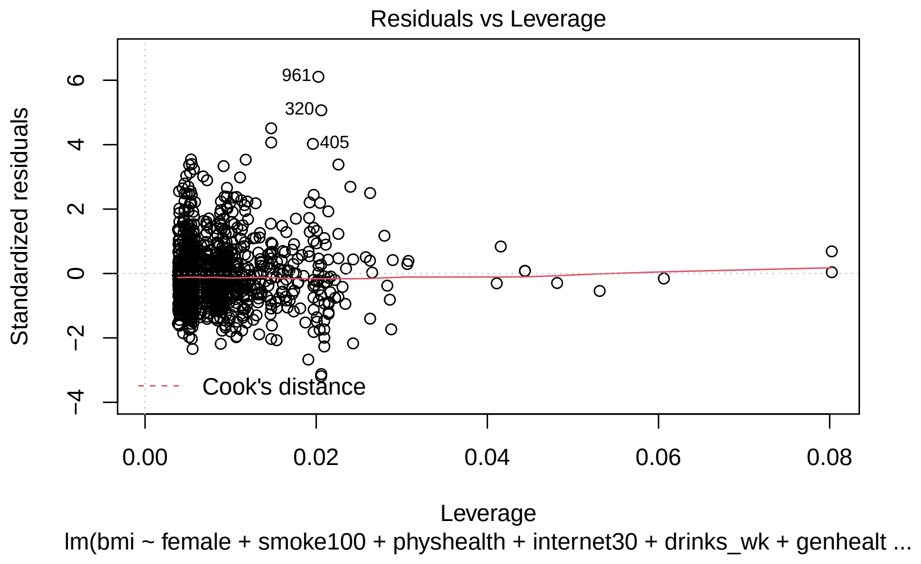
```
plot(c8_m6, which = 1)
```



We see no strong signs of serious non-linearity here. There's no obvious curve in the plot, for example. We may have a problem with increasing variance as we move to the right.

2. What can we conclude from the plot below?

```
plot(c8_m6, 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.)

We might want to identify the point listed here as 961, which appears to have an enormous standardized residual. To do so, we can use the `slice` function from `dplyr`.

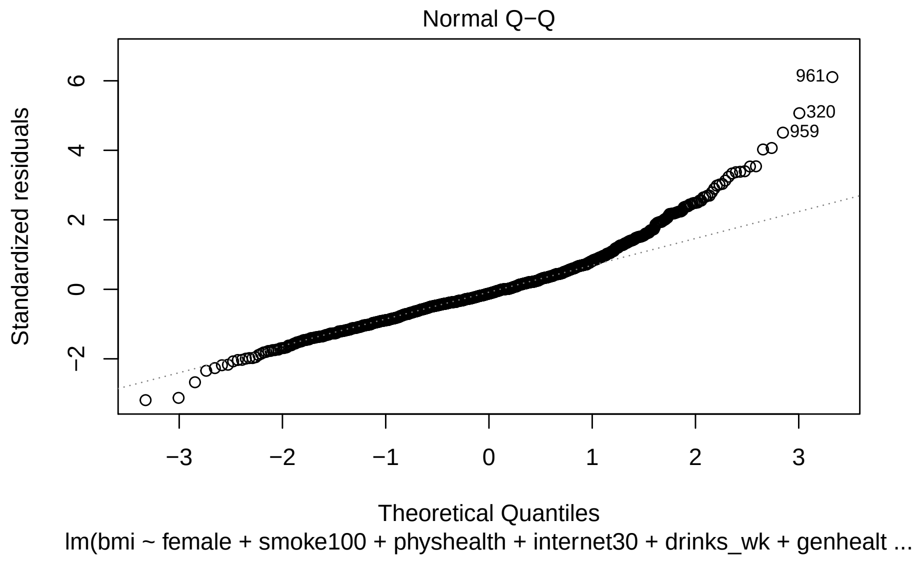
```
smart_cle1_sh %>% slice(961) %>% select(SEQNO)
```

```
# A tibble: 1 x 1
       SEQNO
       <dbl>
1 2017000961
```

Now we know exactly which subject we're talking about.

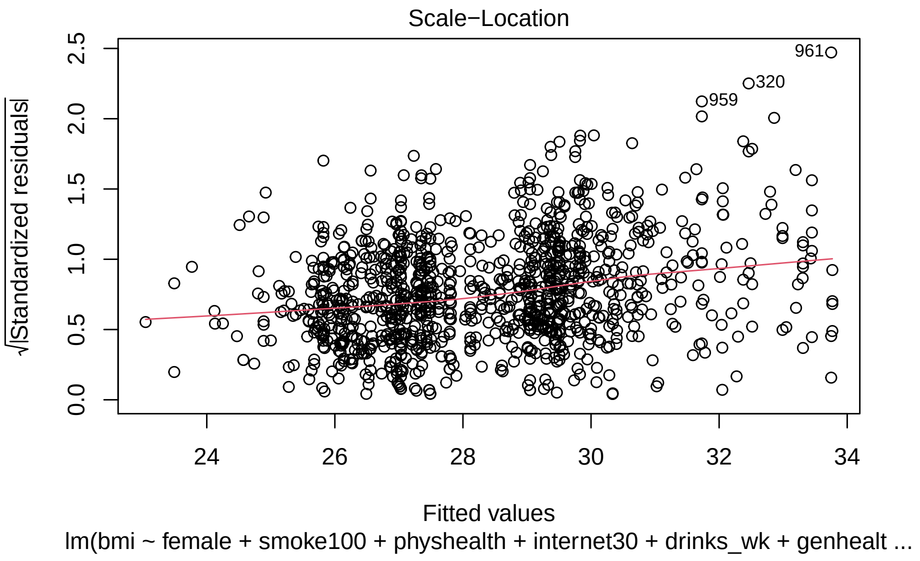
3. What other residual plots are available with `plot` and how do we interpret them?

```
plot(c8_m6, which = 2)
```



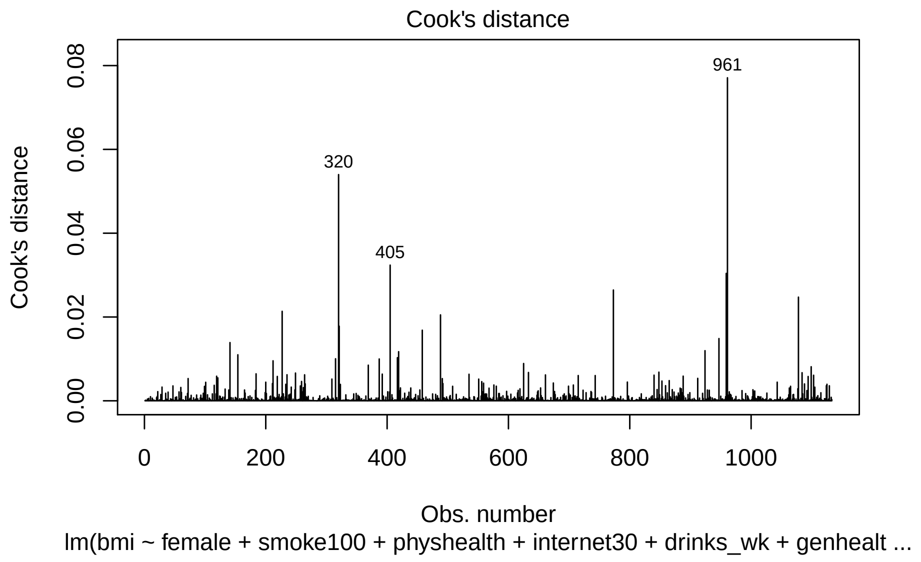
This plot is simply a Normal Q-Q plot of the standardized residuals from our model. We're looking here for serious problems with the assumption of Normality.

```
plot(c8_m6, which = 3)
```



This is a scale-location plot, designed to help us see non-constant variance in the residuals as we move across the fitted values as a linear trend, rather than as a fan shape, by plotting the square root of the residuals on the vertical axis.

```
plot(c8_m6, which = 4)
```



Finally, this is an index plot of the Cook's distance values, allowing us to identify points that are particularly large. Remember that a value of 0.5 (or perhaps even 1.0) is a reasonable boundary for a substantially influential point.

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   x12   x13
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1    36    27    71   8.1  3.34  11.4  81.5  3243   8.8  42.6  11.7    21    15
 2    35    23    72  11.1  3.14  11    78.8  4281   3.5  50.7  14.4     8    10
 3    44    29    74  10.4  3.21   9.8  81.6  4260   0.8  39.4  12.4     6     6
 4    47    45    79   6.5  3.41  11.1  77.5  3125  27.1  50.2  20.6    18     8
 5    43    35    77   7.6  3.44   9.6  84.6  6441  24.4  43.7  14.3    43    38
 6    53    45    80   7.7  3.45  10.2  66.8  3325  38.5  43.1  25.5    30    32
 7    43    30    74  10.9  3.23  12.1  83.9  4679   3.5  49.2  11.3    21    32
 8    45    30    73   9.3  3.29  10.6  86    2140   5.3  40.4  10.5     6     4
 9    36    24    70   9    3.31  10.5  83.2  6582   8.1  42.5  12.6    18    12
10    36    27    72   9.5  3.36  10.7  79.3  4213   6.7  41    13.2    12     7
# ... with 50 more rows, and 3 more variables: x14 <dbl>, x15 <dbl>, y <dbl>
```

Here's a codebook:

Variable	Description

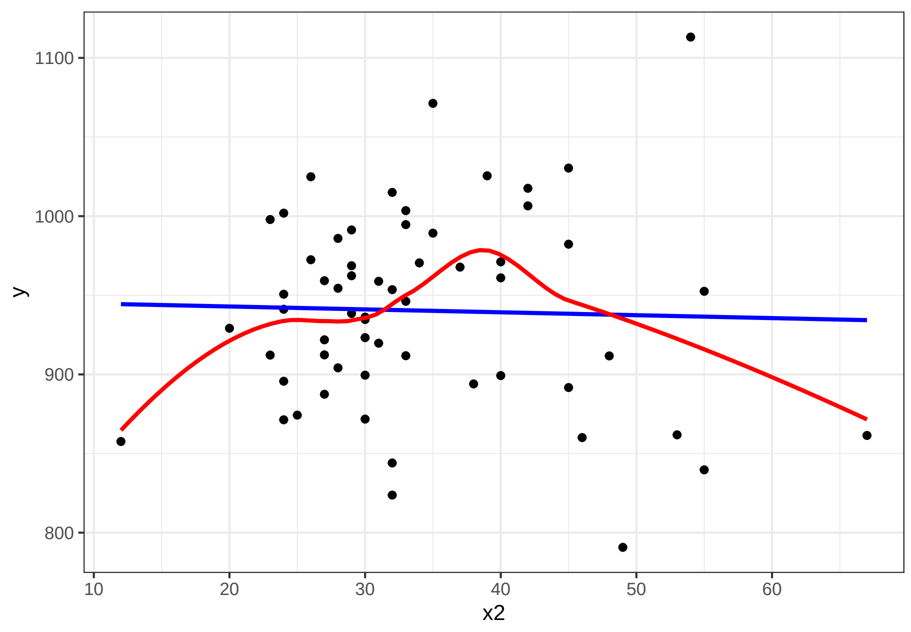
Variable	Description
y	Total Age Adjusted Mortality Rate
x1	Mean annual precipitation in inches
x2	Mean January temperature in degrees Fahrenheit
x3	Mean July temperature in degrees Fahrenheit
x4	Percent of 1960 SMSA population that is 65 years of age or over
x5	Population per household, 1960 SMSA
x6	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
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?

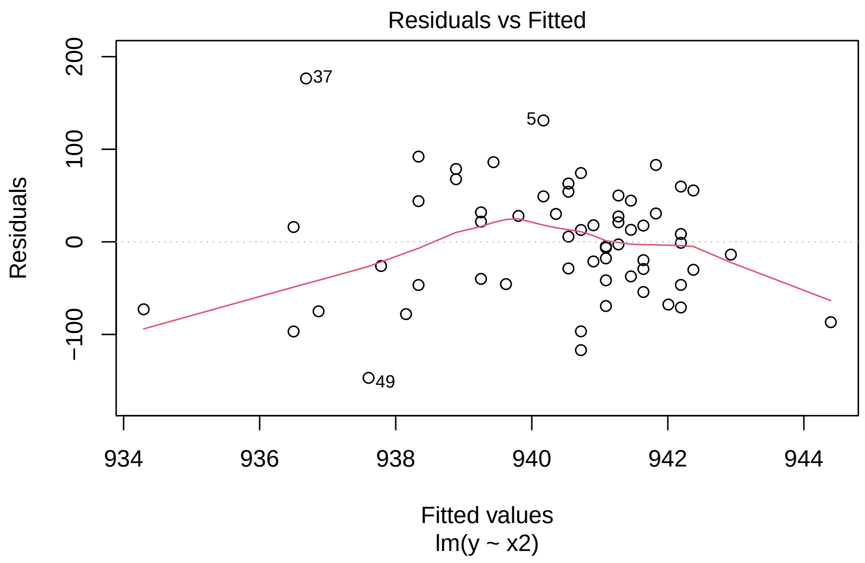
```
ggplot(pollution, aes(x = x2, y = y)) +
    geom_point() +
    geom_smooth(method = "lm", col = "blue", se = F) +
    geom_smooth(method = "loess", col = "red", se = F)
```

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



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.

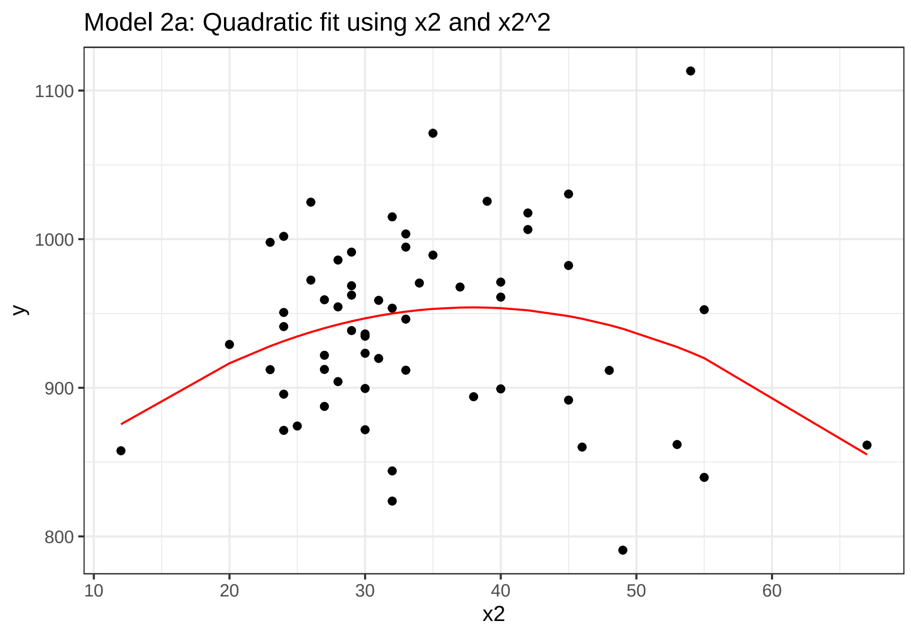
```
summary(mod2a)
```

```
Call:
lm(formula = y ~ x2 + x2squared, data = pollution)
```

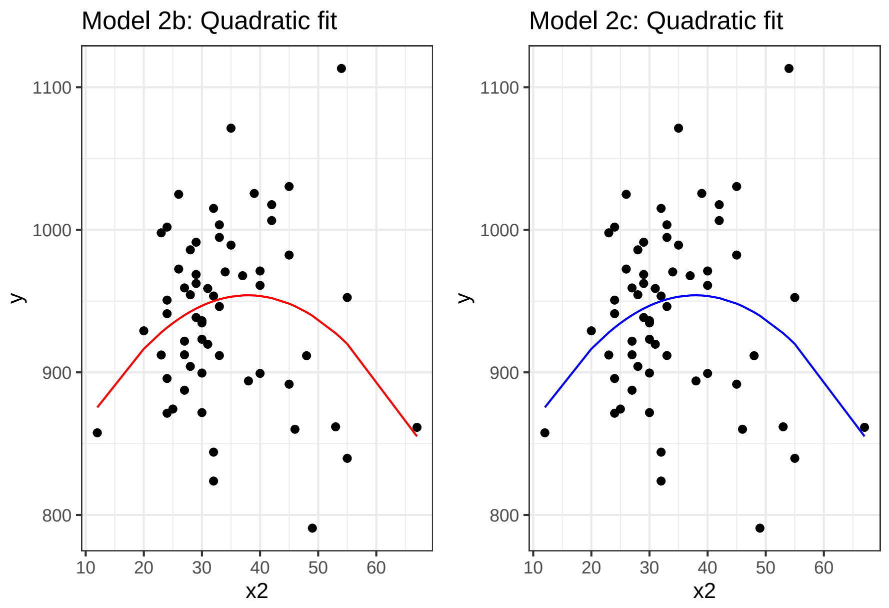
```
Residuals:
     Min       1Q   Median       3Q      Max 
-148.977  -38.651    6.889   35.312  189.346 

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, pollution)

ggplot(mod2a.aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(aes(x = x2, y = .fitted), col = "red") +
    labs(title = "Model 2a: Quadratic fit using x2 and x2^2")
```



```
mod2b.aug <- augment(mod2b, pollution)

mod2c.aug <- augment(mod2c, pollution)

p1 <- ggplot(mod2b.aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(aes(x = x2, y = .fitted), col = "red") +
    labs(title = "Model 2b: Quadratic fit")

p2 <- ggplot(mod2c.aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(aes(x = x2, y = .fitted), col = "blue") +
    labs(title = "Model 2c: Quadratic fit")

p1 + p2
```



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.

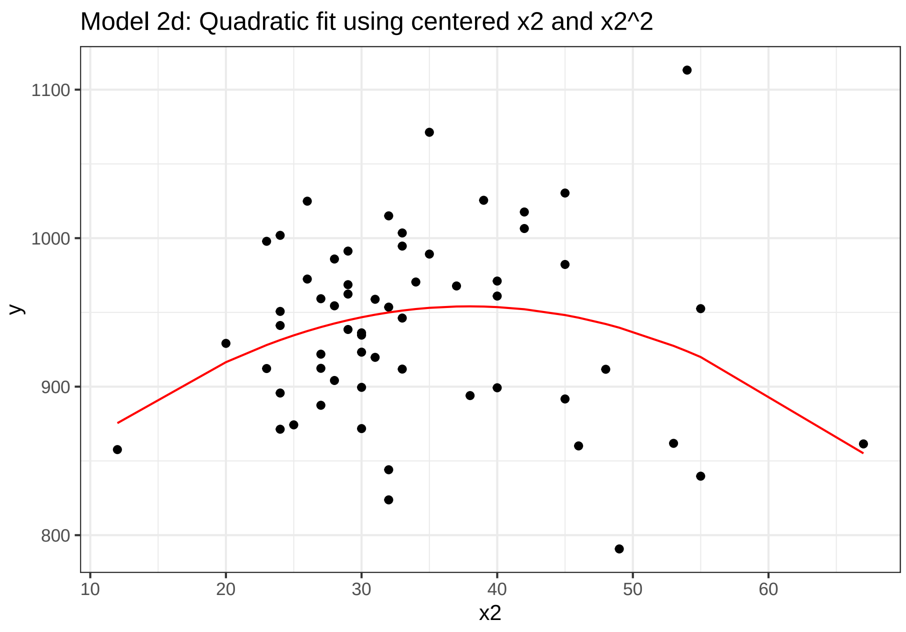
```
pollution <- pollution %>%
    mutate(x2_c = x2 - mean(x2))
```

```r
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.

```r
mod2d.aug <- augment(mod2d, pollution)

ggplot(mod2d.aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(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.

```
mosaic::favstats(~ .fitted, data = mod2a.aug)
```

```
      min       Q1  median       Q3     max     mean       sd  n missing
 855.1041 936.7155 945.597 950.2883 954.073 940.3585 17.17507 60       0
```

```
mosaic::favstats(~ .fitted, data = mod2b.aug)
```

```
      min       Q1  median       Q3     max     mean       sd  n missing
 855.1041 936.7155 945.597 950.2883 954.073 940.3585 17.17507 60       0
```

```
mosaic::favstats(~ .fitted, data = mod2c.aug)
```

```
      min       Q1  median       Q3     max     mean       sd  n missing
 855.1041 936.7155 945.597 950.2883 954.073 940.3585 17.17507 60       0
```

```
mosaic::favstats(~ .fitted, data = mod2d.aug)
```

```
      min       Q1  median       Q3     max     mean       sd  n missing
 855.1041 936.7155 945.597 950.2883 954.073 940.3585 17.17507 60       0
```

9.4 Orthogonal Polynomials

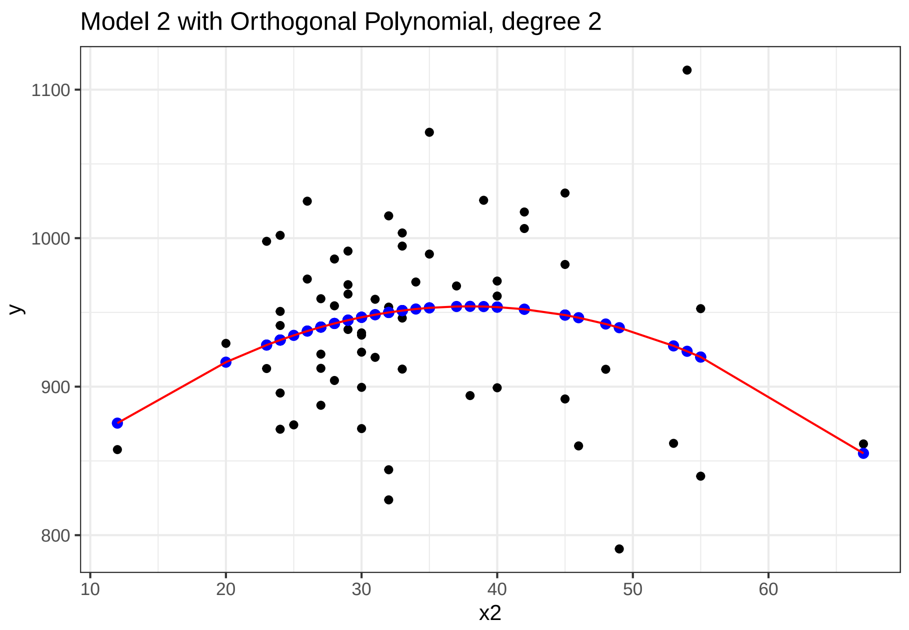
Now, let's fit an orthogonal polynomial of degree 2 to predict `y` using `x2`.

```
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, pollution)

ggplot(mod2orth.aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_point(aes(x = x2, y = .fitted),
               col = "blue", size = 2) +
    geom_line(data = mod2a.aug, aes(x = x2, y = .fitted),
              col = "red") +
    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.

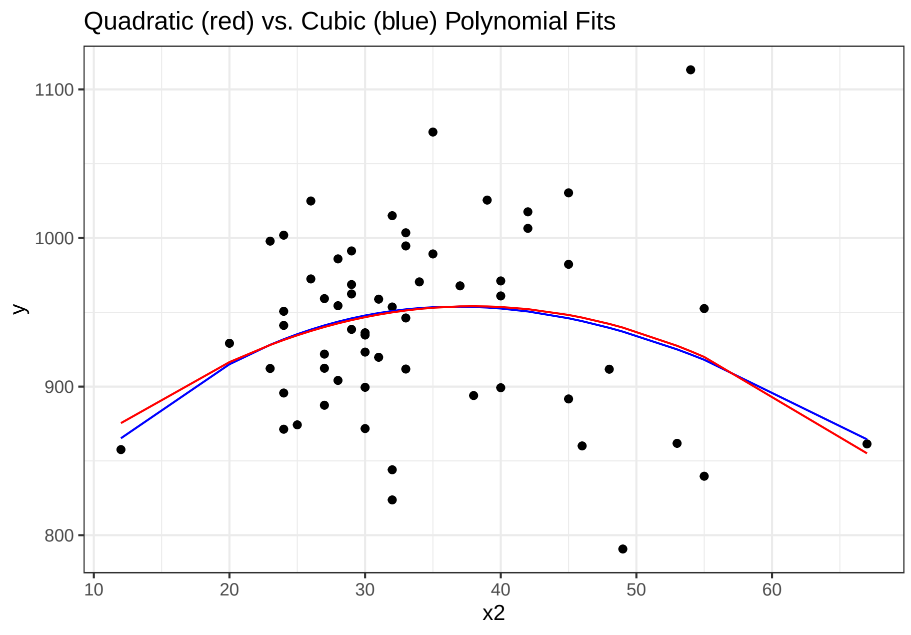
Would the addition of a cubic term help us much in predicting `y` from `x2`?

```
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, pollution)

ggplot(mod3.aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(aes(x = x2, y = .fitted),
              col = "blue") +
    geom_line(data = mod2orth.aug, aes(x = x2, y = .fitted),
              col = "red") +
    labs(title = "Quadratic (red) vs. Cubic (blue) Polynomial Fits")
```

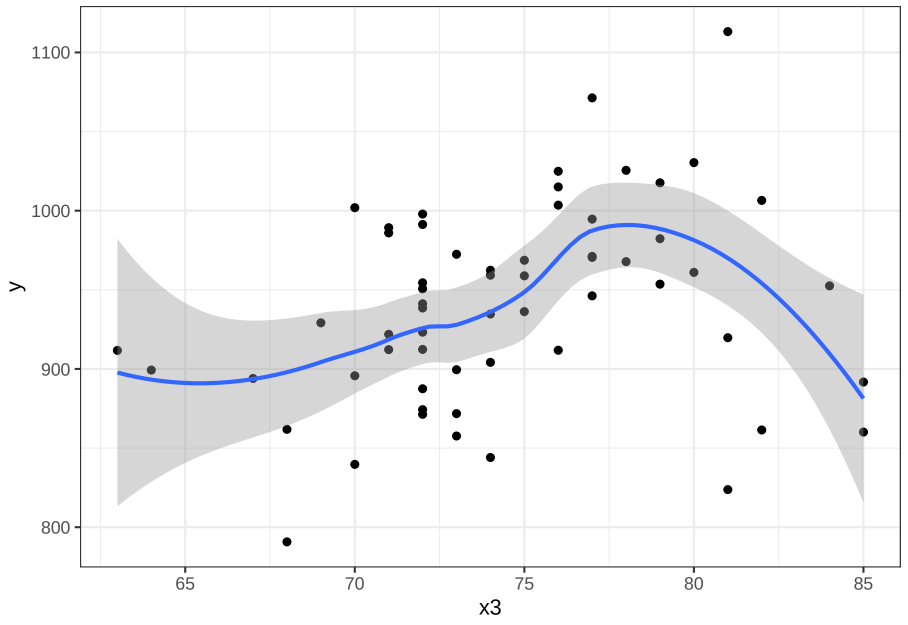


9.5 Fit a cubic polynomial to predict y from x3

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")
```

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



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)
```

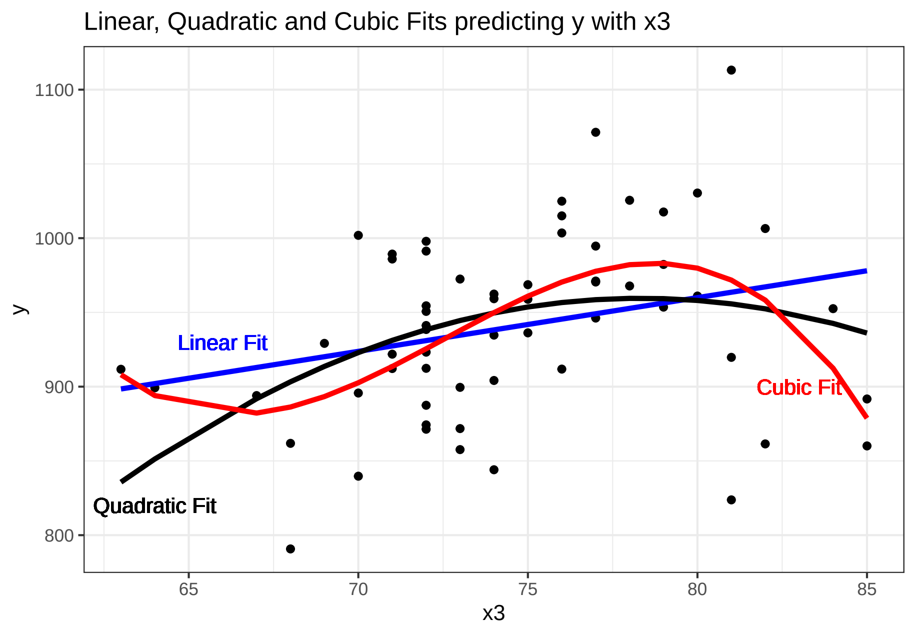
```
Call:
lm(formula = y ~ poly(x3, 2), data = pollution)

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, pollution)

mod4_Q.aug <- augment(mod4_Q, pollution)

mod4_C.aug <- augment(mod4_C, pollution)

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)
```

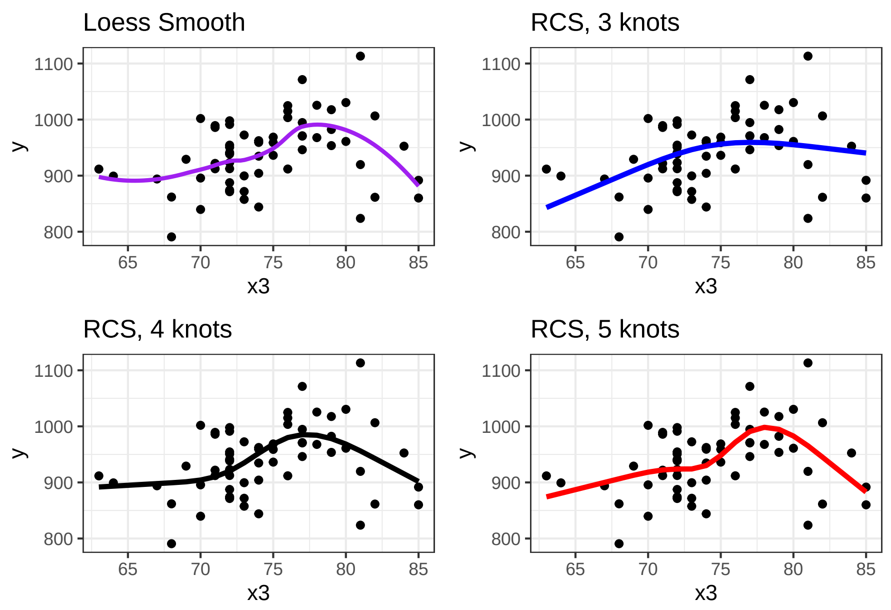
```

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, pollution)

mod5b.aug <- augment(mod5b_rcs, pollution)

mod5c.aug <- augment(mod5c_rcs, pollution)
```

```
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(mod5a.aug, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(aes(x = x3, y = .fitted),
              col = "blue", size = 1.25) +
    labs(title = "RCS, 3 knots") +
    theme_bw()

p4 <- ggplot(mod5b.aug, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(aes(x = x3, y = .fitted),
              col = "black", size = 1.25) +
    labs(title = "RCS, 4 knots") +
    theme_bw()

p5 <- ggplot(mod5c.aug, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(aes(x = x3, y = .fitted),
              col = "red", size = 1.25) +
    labs(title = "RCS, 5 knots") +
    theme_bw()

(p2 + p3) / (p4 + p5)
```

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



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)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.146         0.116  58.5      4.88  0.0111     2  -328.  663.  672.
# ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
glance(mod5b_rcs)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.249         0.209  55.3      6.19 0.00104     3  -324.  658.  668.
# ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
glance(mod5c_rcs)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.288         0.237  54.4      5.57 7.73e-4     4  -322.  657.  669.
# ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

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

Model	Knots	R^2	Adj. R^2	AIC	BIC

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.
- Sometimes I hold myself to a tougher standard, or $n/50$ predictors, at maximum.

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) using the $n/15$ limit.

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.)

```
car::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 Plot

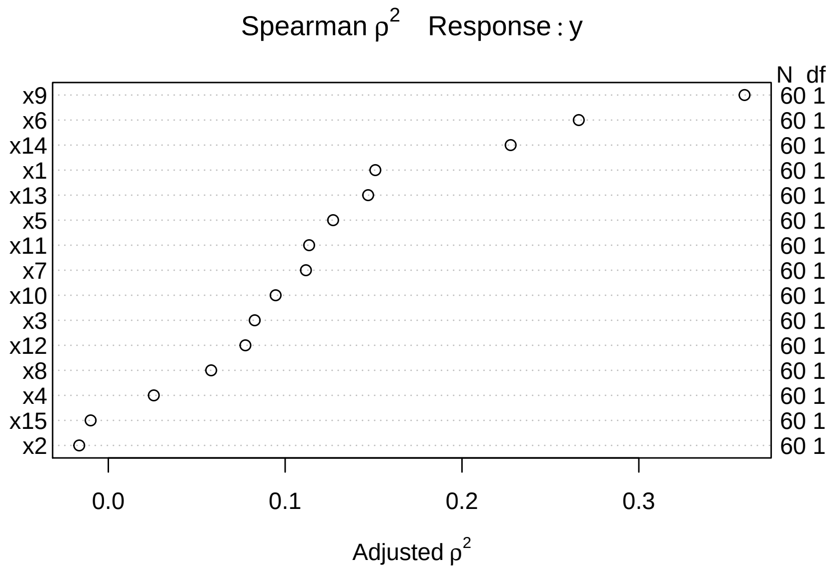
We need a flexible approach to assessing non-linearity and fitting models with non-linear predictors. This will lead us to a measure of what Harrell (2001) calls **potential predictive punch** which hides the true form of the regression from the analyst so as to preserve statistical properties, but that 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.

Let's try such a plot for our fifteen predictors:

```
sp2 <- Hmisc::spearman2(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 +
                            x8 + x9 + x10 + x11 + x12 + x13 +
                            x14 + x15, data = pollution)

plot(sp2)
```



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`,
- a quadratic polynomial on `x14`, and
- 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 one 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.

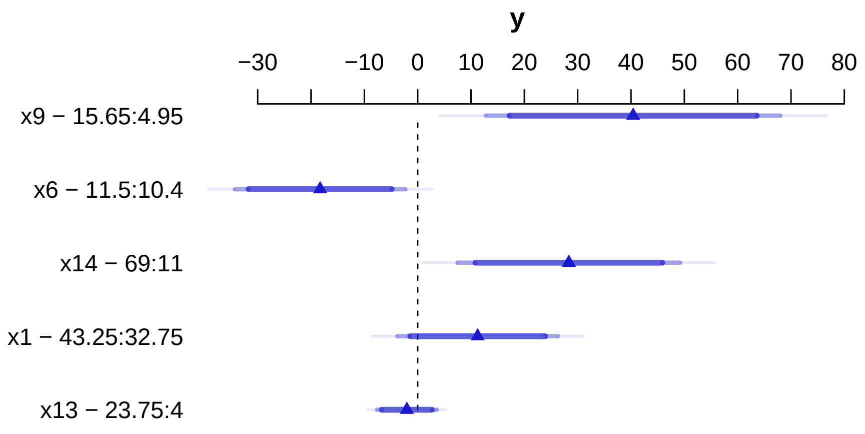
```
summary(newmod)
```

```
             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, five - one for each predictor) confidence intervals for effect sizes has a family-wise 95% confidence level. You'll note that the effect estimates and standard errors are unchanged from those shown above, 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   3.09830  77.7130
 x6     10.40 11.50  1.10 -18.2930  8.1499 -39.88900   3.3024
 x14    11.00 69.00 58.00  28.3480 10.6480   0.13337  56.5630
 x1     32.75 43.25 10.50  11.2520  7.6833  -9.10670  31.6120
 x13     4.00 23.75 19.75  -2.0303  2.8502  -9.58260   5.5221
```

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
```

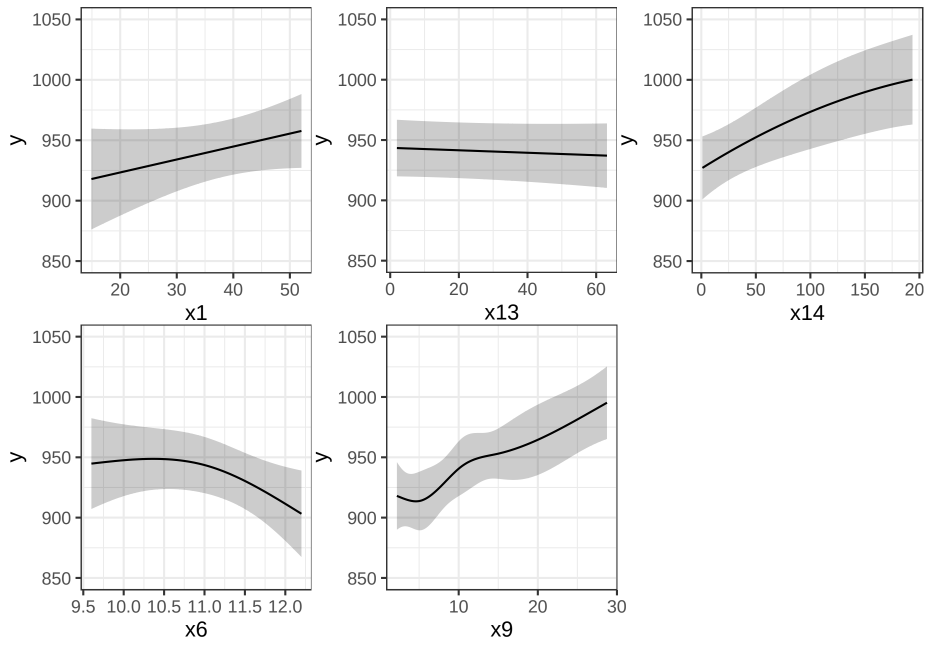
```
Adjust to: x6=11.05 x14=30 x1=38 x13=9

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 882.4115 945.0669
2   6 11.05  30 38   9 916.3490 884.9158 947.7822
3   7 11.05  30 38   9 921.3093 892.4552 950.1635
4   8 11.05  30 38   9 927.6464 901.0299 954.2630
5   9 11.05  30 38   9 934.3853 907.2544 961.5163
6  10 11.05  30 38   9 940.5510 911.2187 969.8832
7  11 11.05  30 38   9 945.2225 915.2296 975.2154
8  12 11.05  30 38   9 948.2885 920.0965 976.4805
9  13 11.05  30 38   9 950.2608 924.4842 976.0374
10 14 11.05  30 38   9 951.6671 926.5755 976.7587
11 15 11.05  30 38   9 953.0342 926.0856 979.9827

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.

```r
head(residuals(newmod, type = "dfbetas"))
```

```
            [,1]         [,2]         [,3]        [,4]        [,5]        [,6]
[1,]  0.03071160 -0.023775487 -0.004055111  0.01205425 -0.03260003 -0.02392315
[2,] -0.38276573 -0.048404993 -0.142293606  0.17009666 -0.22350621  0.44737372
[3,]  0.17226780 -0.426153536  0.350913139 -0.32949129  0.25777913 -0.10263448
[4,]  0.06175110 -0.006460916  0.024828272 -0.03009337  0.04154812 -0.06254145
[5,]  0.16875200  0.039839994 -0.058178534  0.06449504 -0.07772208 -0.18058630
[6,]  0.03322073  0.112699877 -0.203543632  0.23987378 -0.35201736 -0.04075617
            [,7]        [,8]         [,9]        [,10]       [,11]
[1,]  0.01175375 -0.06494414  0.060929683 -0.011042644  0.03425156
[2,] -0.48562818  0.19372285 -0.212186731 -0.107830147 -0.01503250
[3,]  0.05005284 -0.02049877  0.014059330  0.010793169  0.04924166
[4,]  0.05498432  0.01135031 -0.001877983 -0.005490454 -0.01254111
[5,]  0.16151742  0.02723710  0.065483158  0.003326357 -0.05570035
[6,]  0.02900006 -0.21508009  0.171627718  0.019241676  0.05775536
```

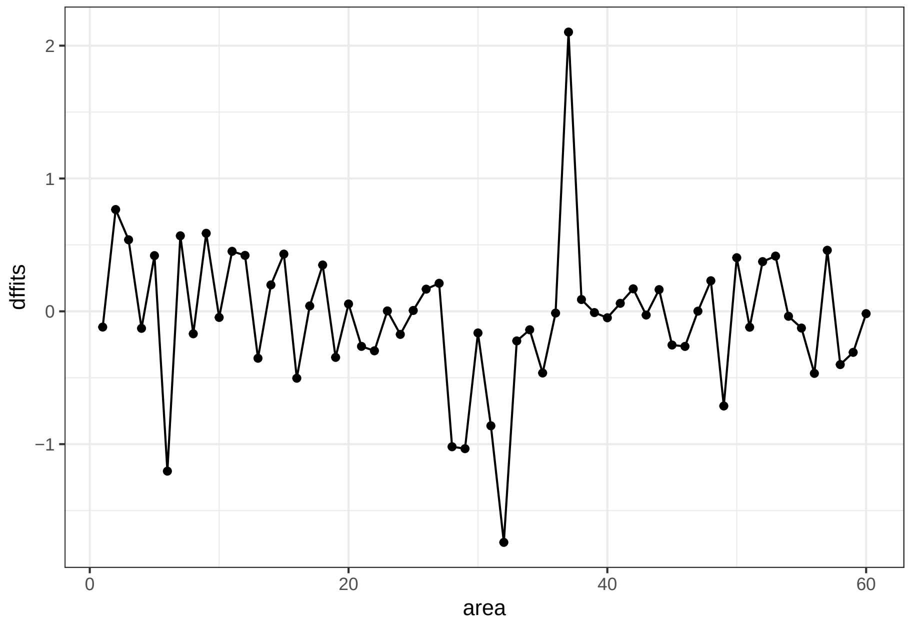
10.5.2 Using the `residuals` command for other summaries

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

$$(\text{observed y}) = \text{Intercept} + \text{Slope } (\text{predicted y})$$

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   0.7426    0.5749    0.1676          0.5312 40
MSE        1145.7800 963.9565 1617.4042 -653.4478       1799.2278 40
g            58.9614  59.7891   54.6444    5.1447         53.8168 40
Intercept     0.0000   0.0000   96.6990  -96.6990         96.6990 40
Slope         1.0000   1.0000    0.8961    0.1039          0.8961 40
```

So, for `R-square` we see that our original estimate was 0.6989

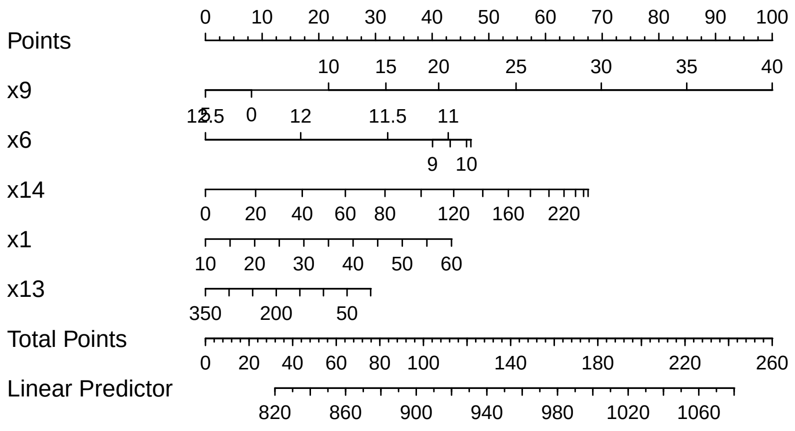
- Our estimated `R-square` across `n` = 40 training samples was 0.7426, but in the resulting tests, the average `R-square` was only 0.5749
- This suggests an optimism of 0.7426 - 0.5749 = 0.1676 (after rounding).
- We then apply that optimism to obtain a new estimate of R^2 corrected for overfitting, at 0.5312, 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

Logistic Regression: The Foundations

Sources for this material include Harrell (2001), Harrell (2018), Ramsey and Schafer (2002) (chapters 20-21), Vittinghoff et al. (2012) (chapter 5) and Faraway (2006) (chapter 2).

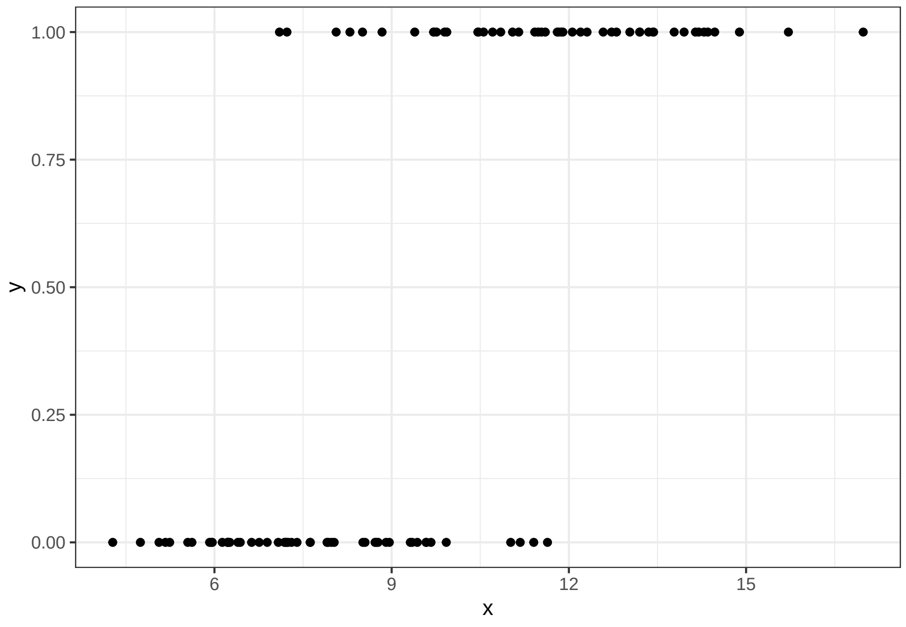
11.1 A First Attempt: A Linear Probability Model

Suppose we want to predict a binary outcome which takes on the value 1 or 0, based on a single quantitative predictor. Let y be a 1/0 outcome, and x be a quantitative predictor in the following simulation.

```
set.seed(432)
sim12 <- data_frame(x = rnorm(100, 10, 3),
                    err = rnorm(100, 0, 2),
                    y = ifelse(x + err > 10, 1, 0))
```

```
Warning: `data_frame()` is deprecated as of tibble 1.1.0.
Please use `tibble()` instead.
This warning is displayed once every 8 hours.
Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
sim12 <- select(sim12, x, y)

ggplot(sim12, aes(x = x, y = y)) + geom_point()
```



Now, we want to use our variable x here to predict our variable y (which takes on the values 0 and 1).

One approach to doing this would be a linear probability model, as follows:

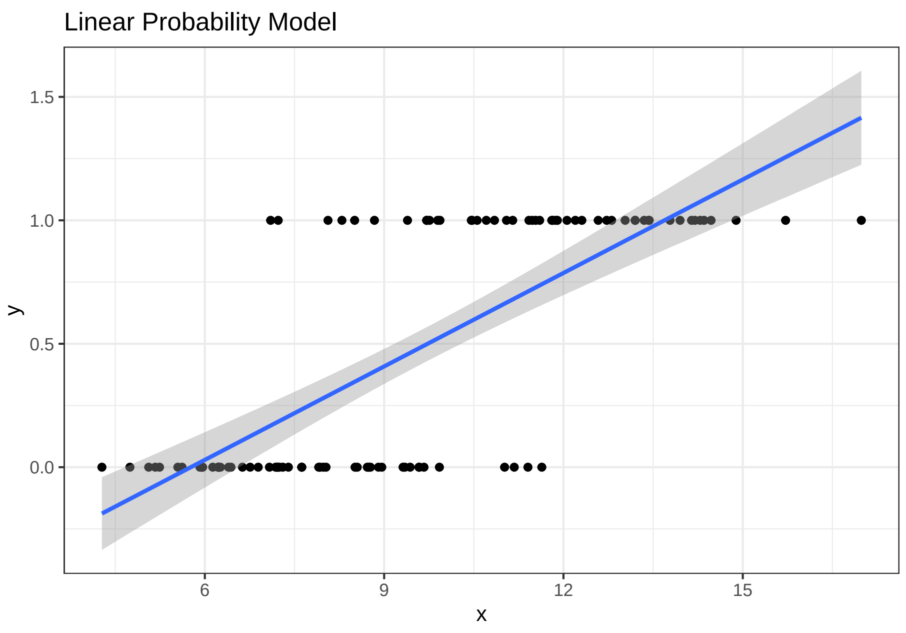
```r
mod12a <- lm(y ~ x, data = sim12)

summary(mod12a)
```

```
Call:
lm(formula = y ~ x, data = sim12)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.74104 -0.23411 -0.02894  0.23117  0.83153 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.72761    0.12272  -5.929 4.57e-08 ***
x            0.12620    0.01219  10.349  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3491 on 98 degrees of freedom
Multiple R-squared:  0.5222,    Adjusted R-squared:  0.5173 
F-statistic: 107.1 on 1 and 98 DF,  p-value: < 2.2e-16
```

Here's a picture of this model. What's wrong here?

```r
ggplot(sim12, aes(x = x, y = y)) +
    geom_point() +
    geom_smooth(method = "lm") +
    labs(title = "Linear Probability Model")
```

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



If y can only take the values 0 and 1 (or, more precisely, if we're trying to predict the value $\pi = \Pr(y = 1)$) then what do we do with the predictions that are outside the range of (0, 1)?

11.2 Logistic Regression

Logistic regression is the most common model used when the outcome is binary. Our response variable is assumed to take on two values, zero or one, and we then describe the probability of a "one" response, given a linear function of explanatory predictors. We use logistic regression rather than linear regression for predicting binary outcomes. Linear regression approaches to the problem of predicting probabilities are problematic for several reasons - not least of which being that they predict probabilities greater than one and less than zero. There are several available alternatives, including probit regression and binomial regression, for the problem of predicting a binary outcome.

Logistic regression is part of a class called **generalized linear models** which extend the linear regression model in a variety of ways. There are also several extensions to the logistic regression model, including multinomial logistic regression (which is used for nominal categorical outcomes with more than two levels) and ordered logistic regression (used for ordered multi-categorical outcomes.) The methods involved in binary logistic regression may also be extended to the case where the outcomes are proportions based on counts, often through grouped binary responses (the proportion of cells with chromosomal aberrations, or the proportion of subjects who develop a particular condition.)

Although the models are different in some crucial ways, the practical use of logistic regression tracks well with much of what we've learned about linear regression.

11.3 The Logistic Regression Model

A generalized linear model (or GLM) is a probability model in which the mean of an outcome is related to predictors through a regression equation. A link function g is used to relate the mean, μ, to a linear regression of the predictors $X_1, X_2, ..., X_k$.

$$g(\mu) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k$$

In the case of a logistic regression model,

- the mean μ of our 0/1 outcome is represented by π which describes the probability of a "1" outcome.
- the linking function we use in logistic regression makes use of the logit function, which is built on the natural logarithm.

11.4 The Link Function

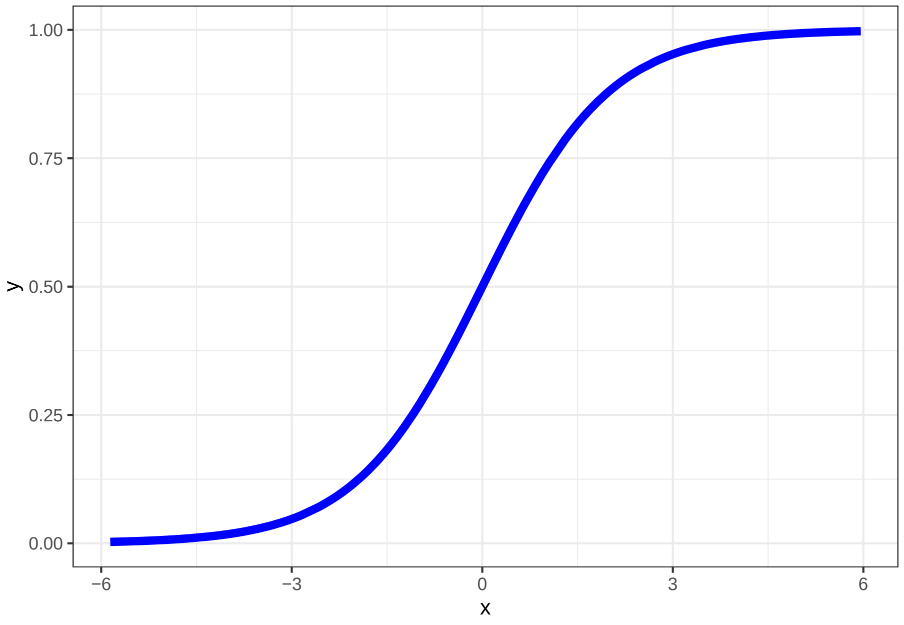
Logistic regression is a non-linear regression approach, since the equation for the mean of the 0/1 Y values conditioned on the values of our predictors $X_1, X_2, ..., X_k$ turns out to be non-linear in the β coefficients. Its nonlinearity, however, is solely found in its link function, hence the term *generalized* linear model.

The particular link function we use in logistic regression is called the **logit link**.

$$logit(\pi) = log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k$$

The inverse of the logit function is called the **logistic function**. If logit$(\pi) = \eta$, then $\pi = \frac{exp(\eta)}{1+exp(\eta)}$

The plot below displays the logistic function $y = \frac{e^x}{1+e^x}$



As you can see in the figure above, the logistic function $\frac{e^x}{1+e^x}$ takes any value x in the real numbers and returns a value between 0 and 1.

11.5 The logit or log odds

We usually focus on the **logit** in statistical work, which is the inverse of the logistic function.

- If we have a probability $\pi < 0.5$, then $logit(\pi) < 0$.
- If our probability $\pi > 0.5$, then $logit(\pi) > 0$.
- Finally, if $\pi = 0.5$, then $logit(\pi) = 0$.

11.6 Interpreting the Coefficients of a Logistic Regression Model

The critical thing to remember in interpreting a logistic regression model is that the logit is the log odds function. Exponentiating the logit yields the odds.

So, suppose we have a yes/no outcome variable, where yes = 1, and no = 0, and π = Pr(y = 1). Our model holds that:

$$logit(\pi) = log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k$$

The odds of a yes response (the odds that Y = 1) at the level $X_1, X_2, ..., X_k$ are:

$$Odds(Y=1) = exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k)$$

The **probability** of a yes response (Pr(y = 1), or π) is just

$$\pi = Pr(Y=1) = \frac{Odds(Y=1)}{1+Odds(Y=1)} = \frac{exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k)}{1+exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k)}$$

11.7 The Logistic Regression has non-constant variance

In ordinary least squares regression, the variance $Var(Y|X_1, X_2, ..., X_k) = \sigma^2$ is a constant that does not depend on the predictor values. This is not the case in logistic regression. The mean and variance specifications of the logistic regression model are quite different.

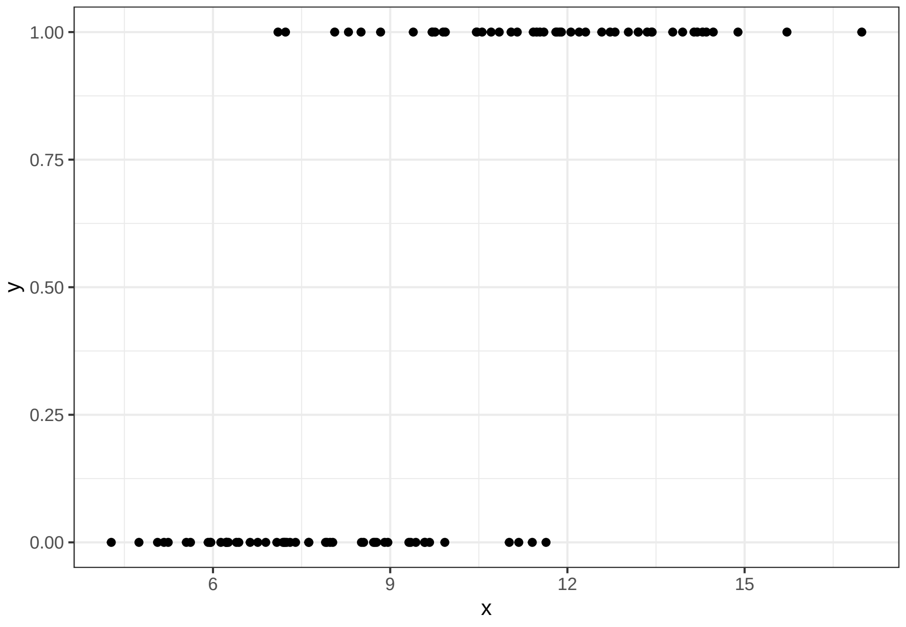
$$logit(\pi) = log\left(\frac{\pi}{1-\pi}\right) = \beta_0+\beta_1 X_1+\beta_2 X_2+...+\beta_k X_k \mu[Y|X_1, ..., X_k] = \pi, Var[Y|X_1, ..., X_k] = \pi(1-\pi)$$

The variance is now a function of the mean, and contains no additional parameter for us to estimate.

11.8 Fitting a Logistic Regression Model to our Simulated Data

Recall the `sim12` data we built earlier.

```
ggplot(sim12, aes(x = x, y = y)) + geom_point()
```



Here is the fitted logistic regression model.

```
model12b <- glm(y ~ x, data = sim12, family = binomial)

model12b
```

```
Call:  glm(formula = y ~ x, family = binomial, data = sim12)

Coefficients:
(Intercept)            x
    -9.1955       0.9566

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:      138.6
Residual Deviance: 70.03    AIC: 74.03
```

The logistic regression equation is:

$$logit(Pr(y=1)) = log\left(\frac{Pr(y=1)}{1-Pr(y=1)}\right) = -9.1955 + 0.9566x$$

We can exponentiate the results of this model to get to an equation about odds, and eventually, a prediction about probabilities. Suppose, for instance, that we are interested in the prediction when x = 12.

$$logit(Pr(y=1)|X=12) = log\left(\frac{Pr(y=1)}{1-Pr(y=1)}\right) = -9.1955{+}0.9566{*}12 = 2.2837$$

And we can also get this from the `predict` function applied to our model, although the `predict` approach retains a few more decimal places internally:

```
predict(model12b, newdata = data.frame(x = 12))
```

```
       1
2.284069
```

$$Odds(Y=1|X=12) = exp(-9.20 + 0.96 * 12) = exp(2.2837) = 9.812921$$

```
exp(predict(model12b, newdata = data.frame(x = 12)))
```

```
      1
9.81654
```

The estimated **probability** of a yes response (Pr(y = 1), or π) if x = 12 is just

$$\pi = Pr(Y=1|X=12) = \frac{Odds(Y=1|X=12)}{1+Odds(Y=1|X=12)} = \frac{exp(-9.20+0.96x)}{1+exp(-9.20+0.96x)} = \frac{9.812921}{1+9.812921} = 0.908$$

Does this work out?

```
exp(predict(model12b, newdata = data.frame(x = 12))) /
    (1 + exp(predict(model12b, newdata = data.frame(x = 12))))
```

```
       1
0.907549
```

which is also directly available by running `predict` with `type = "response"`.

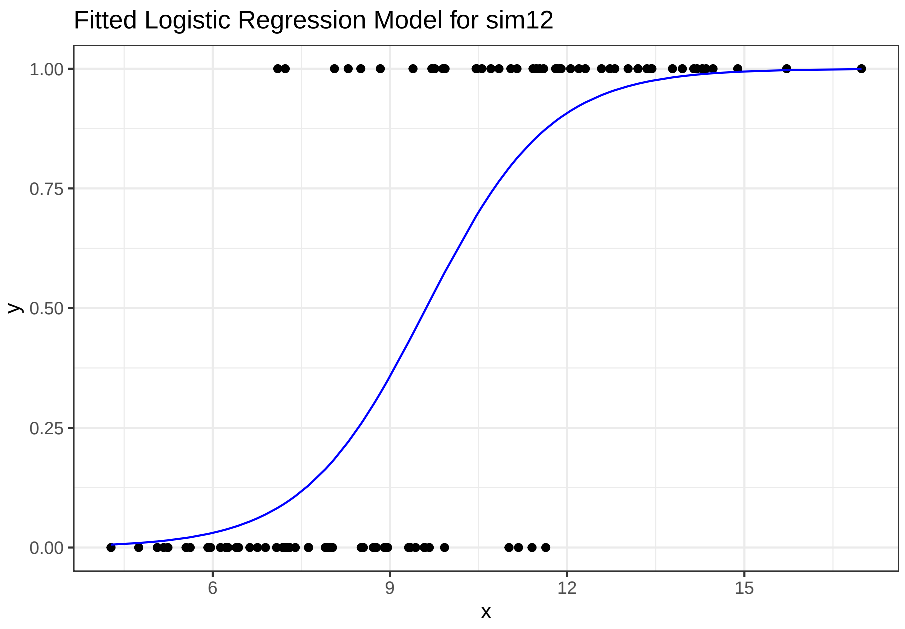
```
predict(model12b, newdata = data.frame(x = 12), type = "response")
```

```
       1
0.907549
```

11.9 Plotting the Logistic Regression Model

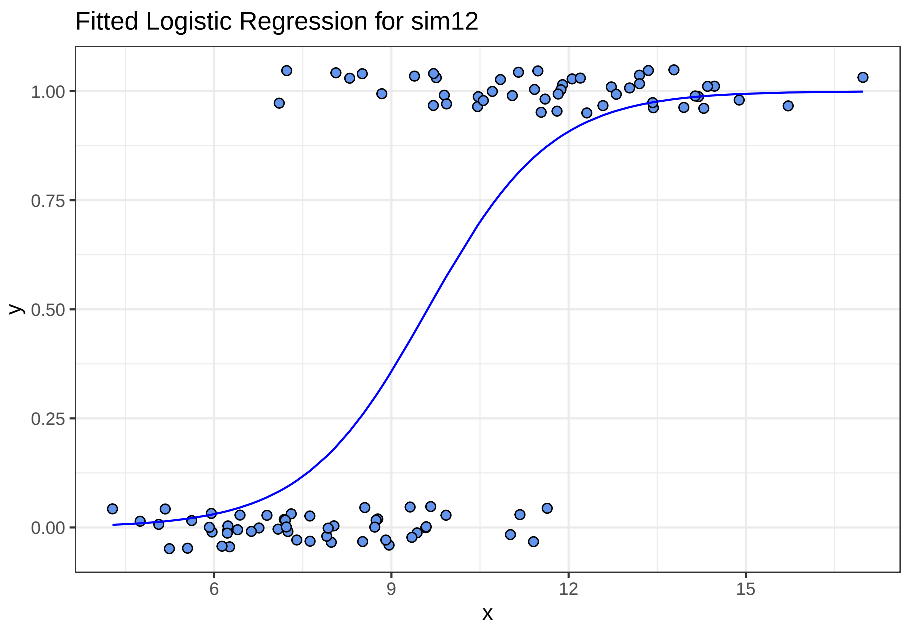
We can use the `augment` function from the `broom` package to get our fitted probabilities included in the data.

```
m12b.aug <- augment(model12b, sim12, type.predict = "response")

ggplot(m12b.aug, aes(x = x, y = y)) +
    geom_point() +
    geom_line(aes(x = x, y = .fitted), col = "blue") +
    labs(title = "Fitted Logistic Regression Model for sim12")
```



I'll add a little jitter on the vertical scale to the points, so we can avoid overlap, and also make the points a little bigger.

```
ggplot(m12b.aug, aes(x = x, y = y)) +
    geom_jitter(height = 0.05, size = 2, pch = 21,
                fill = "cornflowerblue") +
    geom_line(aes(x = x, y = .fitted), col = "blue") +
    labs(title = "Fitted Logistic Regression for sim12") +
    theme_bw()
```



All right, it's time to move on to fitting models. We'll do that in the next Chapter.

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).

```
miss_var_summary(resect)
```

```
# A tibble: 6 x 3
  variable  n_miss pct_miss
  <chr>      <int>    <dbl>
1 id             0        0
2 age            0        0
3 prior          0        0
4 resection      0        0
5 intubated      0        0
6 died           0        0
```

```
resect %>% count(died, prior)
```

```
# A tibble: 4 x 3
   died prior     n
  <dbl> <dbl> <int>
1     0     0    89
2     0     1    28
3     1     0    11
4     1     1     6
```

```
resect %>% mosaic::inspect()
```

```
quantitative variables:
          name   class min    Q1 median     Q3 max       mean         sd   n
...1        id numeric   1 34.25   67.5 100.75 134 67.5000000 38.8265373 134
...2       age numeric   8 36.00   51.0  61.00  80 47.8432836 15.7775202 134
...3     prior numeric   0  0.00    0.0   0.75   1  0.2537313  0.4367785 134
...4 resection numeric   1  2.00    2.5   4.00   6  2.9634328  1.2402123 134
...5 intubated numeric   0  0.00    0.0   0.00   1  0.1417910  0.3501447 134
...6      died numeric   0  0.00    0.0   0.00   1  0.1268657  0.3340713 134
     missing
...1       0
...2       0
...3       0
...4       0
...5       0
...6       0
```

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 first, and save the `lrm` ideas for later in this Chapter.

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.

12.3 Logistic Regression using `glm`

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.

12.3.1 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 substituting in values for `resection`, we can estimate the model's fitted probabilities of death.

12.3.2 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 = tibble(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 = tibble(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.

12.3.3 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:

$$log\left(\frac{Pr(Harrydied)}{1-Pr(Harrydied)}\right) = -4.4337{+}0.7417{\times}5 = -0.7253, log\left(\frac{Pr(Sallydied)}{1-Pr(Sallydied)}\right) = -4.4337{+}0.74$$

which implies that our estimated odds of death for Harry and Sally are:

$$Odds(Harrydied) = \frac{Pr(Harrydied)}{1-Pr(Harrydied)} = e^{-4.4337+0.7417\times 5} = e^{-0.7253} = 0.4842 Odds(Sallydied) = \frac{}{1-}$$

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.

12.3.4 Interpreting the rest of the model output from `glm`

```r
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.

12.3.5 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 describe 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, test = "LRT")
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: died

Terms added sequentially (first to last)

          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                        133    101.943
resection  1    12.45       132     89.493 0.0004179 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `test = "LRT"` section completes a deviance test and provides a *p* value, which just estimates the probability that a chi-square distribution with a single degree of freedom would exhibit an improvement in deviance as large as 12.45.

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 practically useful, so I discourage you from performing global tests of a logistic regression model with a deviance test.

12.3.6 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)
```

```
# A tibble: 1 x 8
  null.deviance df.null logLik   AIC   BIC deviance df.residual  nobs
          <dbl>   <int>  <dbl> <dbl> <dbl>    <dbl>       <int> <int>
1          102.     133  -44.7  93.5  99.3     89.5         132   134
```

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.

12.4 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.

12.4.1 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 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.)

12.4.2 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.

```
tidy(res_modA, exponentiate = TRUE, conf.int = TRUE) %>%
  select(term, estimate, conf.low, conf.high)
```

```
# A tibble: 2 x 4
  term        estimate conf.low conf.high
  <chr>          <dbl>    <dbl>     <dbl>
1 (Intercept)   0.0119  0.00176    0.0575
2 resection     2.10    1.38       3.35
```

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.

12.4.3 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.

12.4.4 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.

12.4.5 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.

12.5 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.

12.5.1 Using `augment` to capture the fitted probabilities

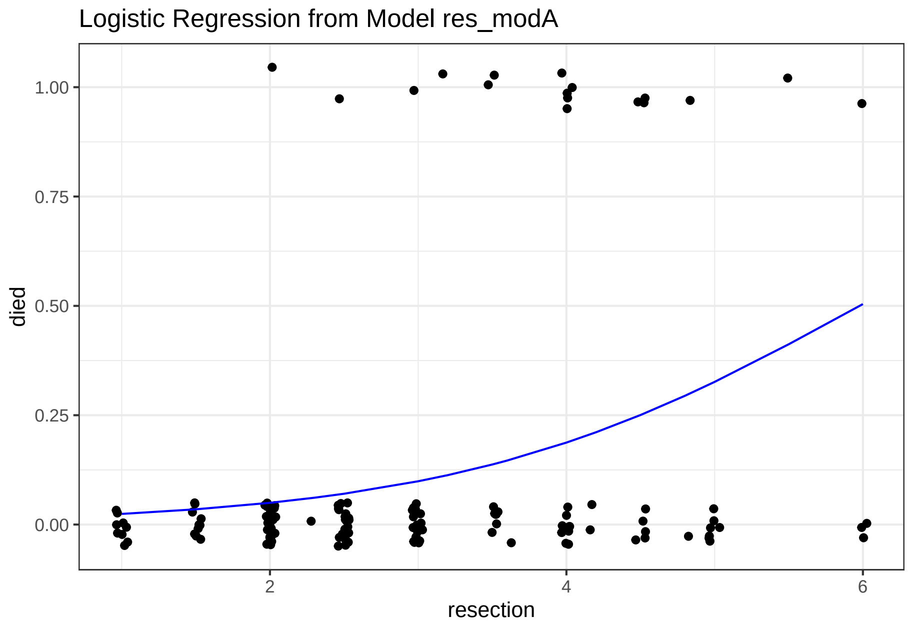
```
res_A_aug <- augment(res_modA, resect,
                     type.predict = "response")
head(res_A_aug)
```

```
# A tibble: 6 x 12
     id   age prior resection intubated  died .fitted .resid .std.resid    .hat
  <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>   <dbl>  <dbl>      <dbl>   <dbl>
1     1    34     1       2.5         0     0  0.0705 -0.382     -0.384 0.0100
2     2    57     0       5           0     0  0.326  -0.889     -0.904 0.0337
3     3    60     1       4           1     1  0.187   1.83       1.84  0.0120
4     4    62     1       4.2         0     0  0.211  -0.689     -0.693 0.0143
5     5    28     0       6           1     1  0.504   1.17       1.22  0.0818
6     6    52     0       3           0     0  0.0990 -0.457     -0.459 0.00922
# ... with 2 more variables: .sigma <dbl>, .cooksd <dbl>
```

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.

12.5.2 Plotting a Logistic Regression Model's Fitted Values

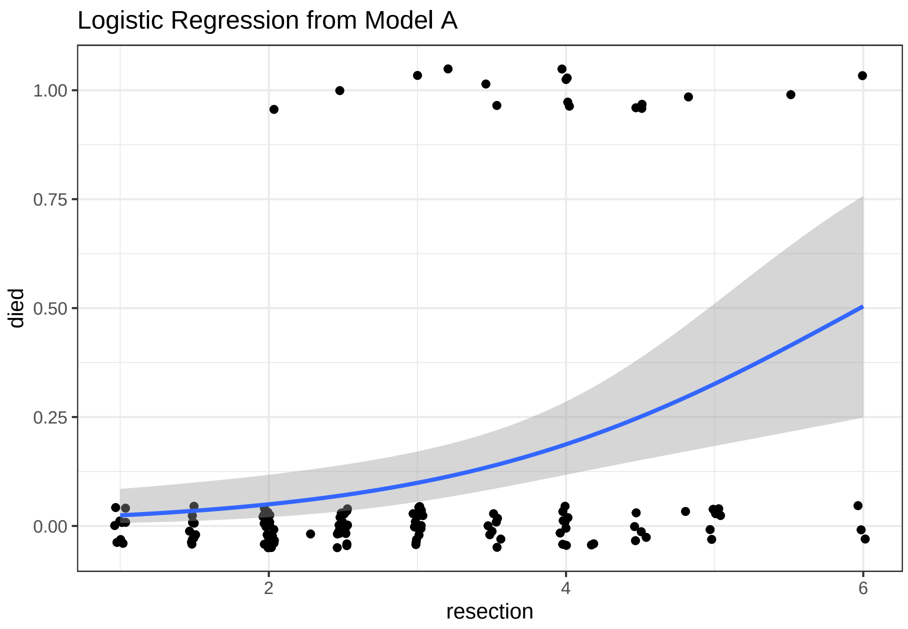
```
ggplot(res_A_aug, aes(x = resection, y = died)) +
    geom_jitter(height = 0.05) +
    geom_line(aes(x = resection, y = .fitted),
              col = "blue") +
    labs(title = "Logistic Regression from Model res_modA")
```



12.5.3 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()
```

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



As expected, we see an increase in the model probability of death as the extent of the resection grows larger.

12.6 How well does Model A classify subjects?

A natural question to ask is how well does our model classify patients in terms of likelihood of death.

We could specify a particular rule, for example: if the predicted probability of death is 0.5 or greater, then predict "Died."

```
res_A_aug$rule.5 <- ifelse(res_A_aug$.fitted >= 0.5,
                           "Predict Died", "Predict Alive")

table(res_A_aug$rule.5, res_A_aug$died)
```

```

                 0   1
  Predict Alive 114  16
  Predict Died    3   1
```

And perhaps build the linked table of row probabilities which tells us, for example, that 87.69% of the patients predicted by the model to be alive actually did survive.

```
round(100*prop.table(
    table(res_A_aug$rule.5, res_A_aug$died), 1), 2)
```

```
                     0     1
  Predict Alive 87.69 12.31
  Predict Died  75.00 25.00
```

Or the table of column probabilities which tell us, for example, that 97.44% of those who actually survived were predicted by the model to be alive.

```
round(100*prop.table(
    table(res_A_aug$rule.5, res_A_aug$died), 2), 2)
```

```
                     0     1
  Predict Alive 97.44 94.12
  Predict Died   2.56  5.88
```

We'll discuss various measures of concordance derived from this sort of classification later.

12.7 The Confusion Matrix

Let's build this misclassification table in standard epidemiological format.

```
confuseA_small <-
  res_A_aug %>%
  mutate(death_predicted = factor(.fitted >= 0.5),
         death_actual = factor(died == "1"),
         death_predicted = fct_relevel(death_predicted, "TRUE"),
         death_actual = fct_relevel(death_actual, "TRUE")) %$%
  table(death_predicted, death_actual)

confuseA_small
```

```
               death_actual
death_predicted TRUE FALSE
          TRUE     1     3
          FALSE   16   114
```

In total, we have 134 observations.

- 115 correct predictions, or 85.8% accuracy
- 17 subjects who died, or 12.6% prevalence of death
- 4 subjects who were predicted to die, or 3.0% detection prevalence.

The sensitivity (also called recall) here is 1 / (1 + 16) = 5.9%.

- 5.9% of the subjects who actually died were predicted to die by the model.

The specificity here is 114 / (114 + 3) = 97.4%.

- 97.4% of the subjects who actually survived were predicted to survive by the model.

The positive predictive value (PPV: also called precision) is 1 / (1 + 3) = 25%

- Our predictions of death were correct 25% of the time.

The negative predictive value (NPV) is 114 / (114 + 16) = 87.7%

- Our predictions of survival were correct 87.7% of the time.

12.8 Using the `confusionMatrix` tool from the `caret` package

This provides a more detailed summary of the classification results from our logistic regression model.

```r
res_A_aug %$%
  confusionMatrix(
    data = factor(.fitted >= 0.5),
    reference = factor(died == 1),
    positive = "TRUE"
  )
```

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE   114   16
     TRUE      3    1

               Accuracy : 0.8582
                 95% CI : (0.7875, 0.9124)
    No Information Rate : 0.8731
    P-Value [Acc > NIR] : 0.747802

                  Kappa : 0.0493
```

```
 Mcnemar's Test P-Value : 0.005905

            Sensitivity : 0.058824
            Specificity : 0.974359
         Pos Pred Value : 0.250000
         Neg Pred Value : 0.876923
             Prevalence : 0.126866
         Detection Rate : 0.007463
   Detection Prevalence : 0.029851
      Balanced Accuracy : 0.516591

       'Positive' Class : TRUE
```

- The No Information Rate or NIR is just the percentage of correct predictions we'd get if we just predicted the more common classification (not dead) for every subject.
- Kappa is a correlation statistic ranging from -1 to +1. It measures the inter-rater reliability of our predictions and the true classifications, in this context. Complete agreement would be +1, and complete disagreement would be -1.

12.9 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`. Suppose 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.

12.9.1 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.

```r
set.seed(432999)
sim.temp <- data_frame(x = rnorm(n = 200),
                       prob = exp(x)/(1 + exp(x)),
                       y = as.numeric(1 * runif(200) < prob))
```

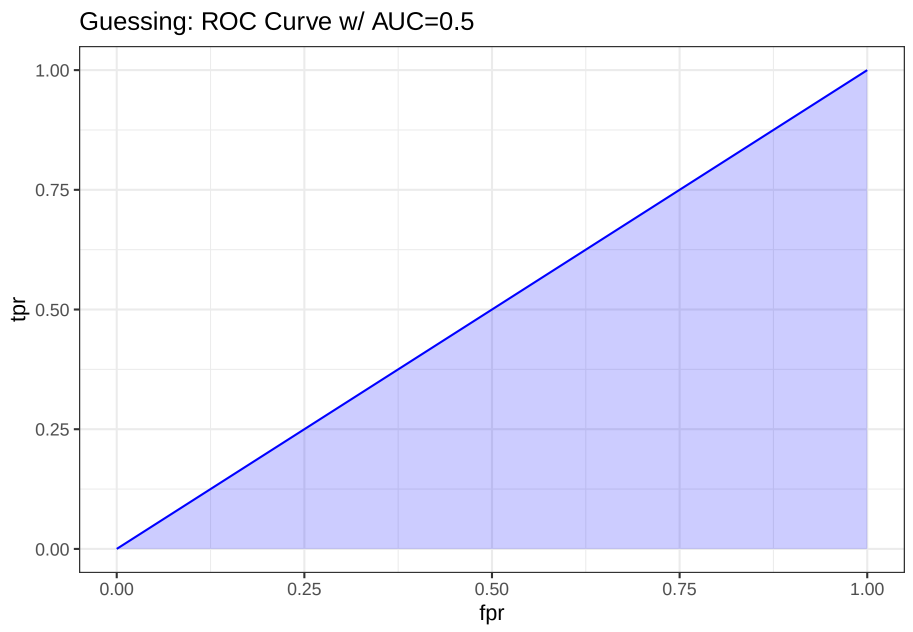
```
sim.temp <- sim.temp %>%
    mutate(p_guess = 1,
           p_perfect = y,
           p_bad = exp(-2*x) / (1 + exp(-2*x)),
           p_ok = prob + (1-y)*runif(1, 0, 0.05),
           p_good = prob + y*runif(1, 0, 0.27))
```

12.9.1.1 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

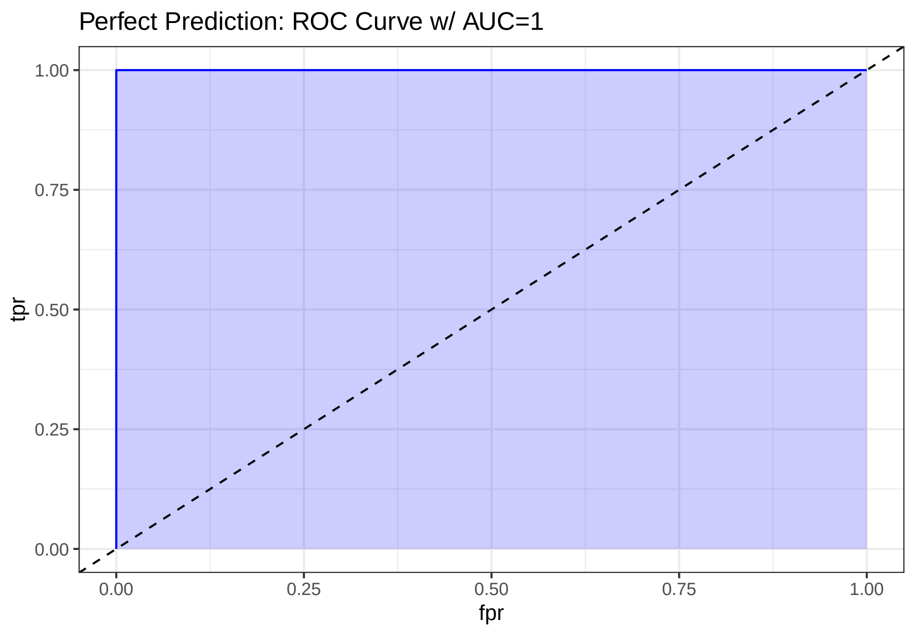
```
pred_guess <- prediction(sim.temp$p_guess, sim.temp$y)
perf_guess <- performance(pred_guess, measure = "tpr", x.measure = "fpr")
auc_guess <- performance(pred_guess, measure="auc")

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()
```



12.9.1.2 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.

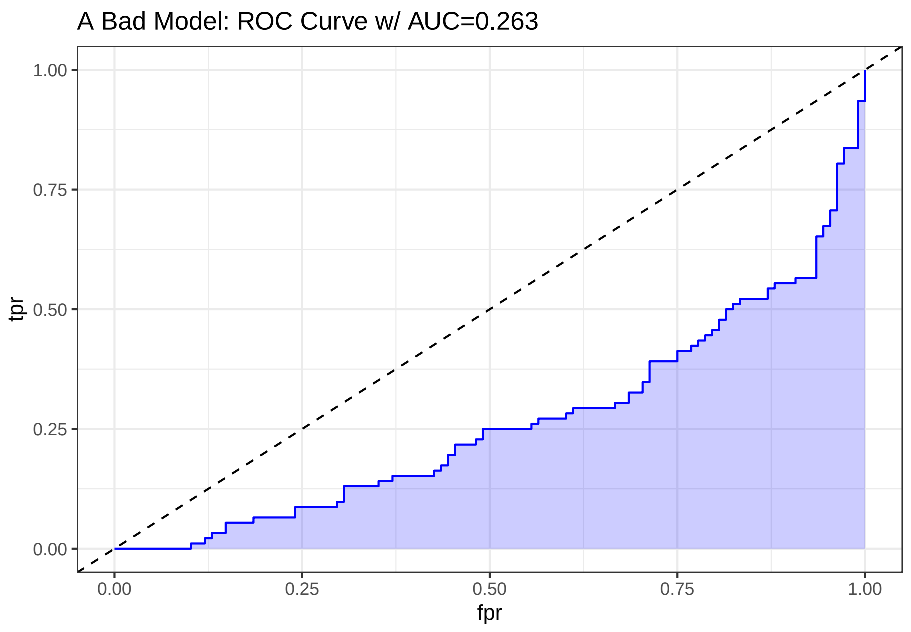
```
pred_perf <- prediction(sim.temp$p_perfect, sim.temp$y)
perf_perf <- performance(pred_perf, measure = "tpr", x.measure = "fpr")
auc_perf <- performance(pred_perf, measure="auc")

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()
```



12.9.1.3 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.

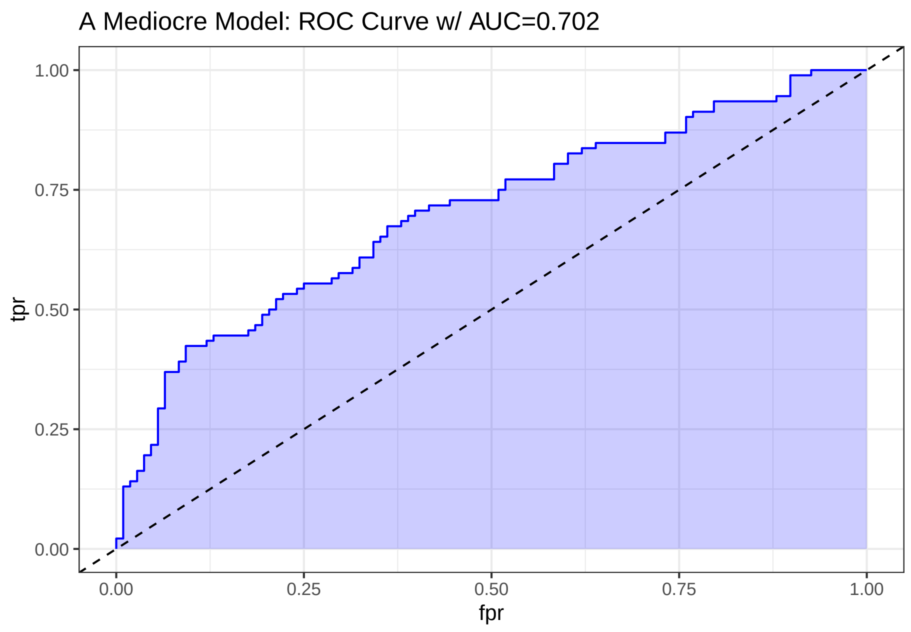
```
pred_bad <- prediction(sim.temp$p_bad, sim.temp$y)
perf_bad <- performance(pred_bad, measure = "tpr", x.measure = "fpr")
auc_bad <- performance(pred_bad, measure="auc")

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()
```



12.9.1.4 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.

```r
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()
```



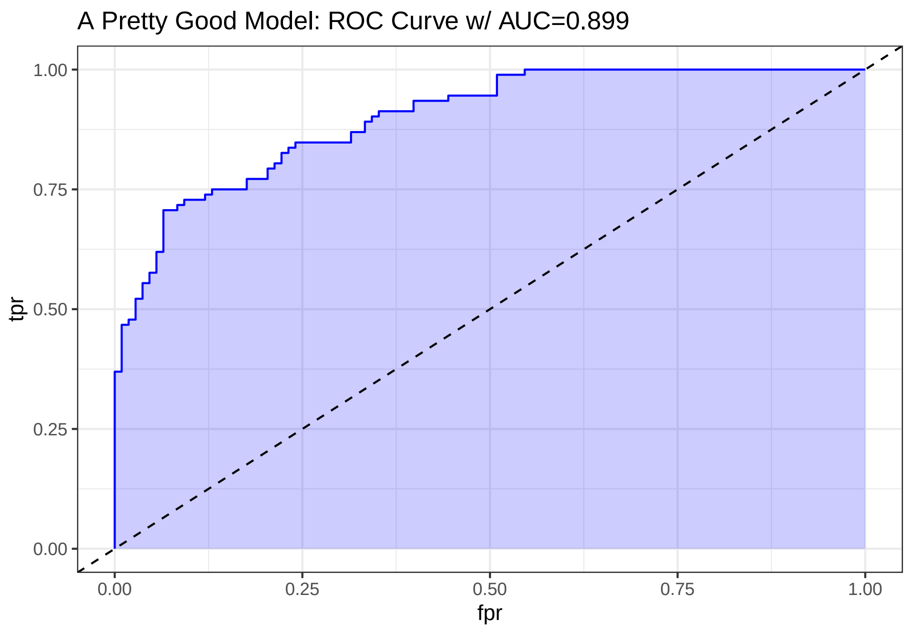
Sometimes people grasp for a rough guide as to the accuracy of a model's predictions based on the area under the ROC curve. A common thought is to assess the C statistic much like you would a class grade.

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

12.9.1.5 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.

```r
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()
```

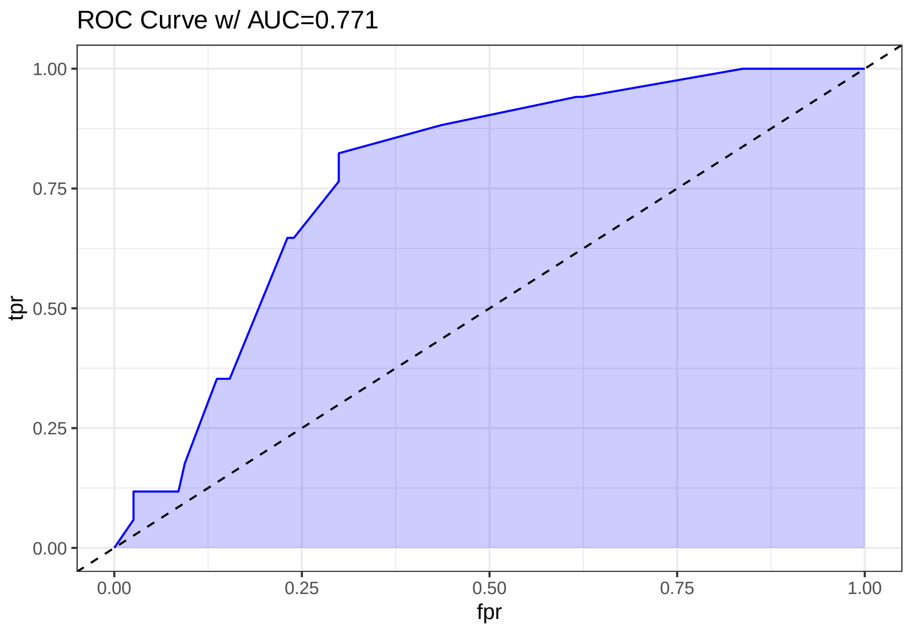


12.10 The ROC Plot for `res_modA`

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")

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.

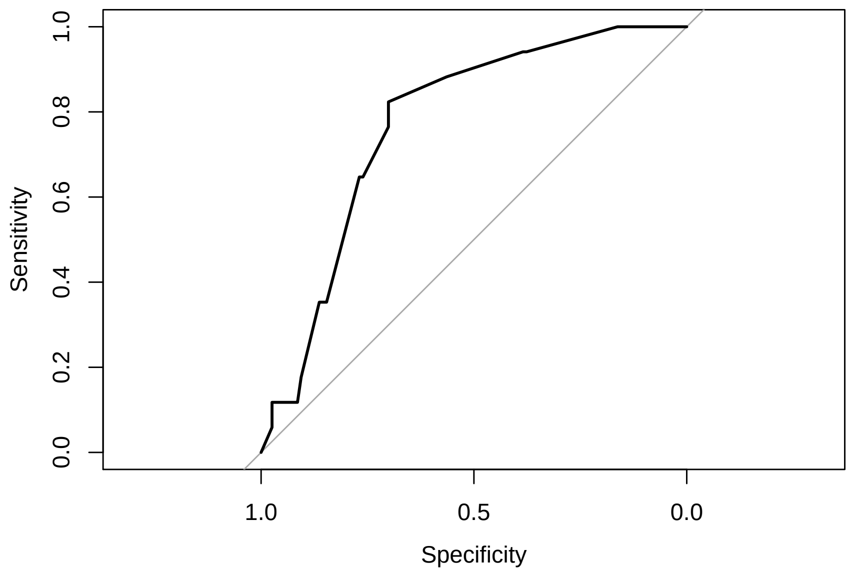
12.10.1 Another way to plot the ROC Curve

If we’ve loaded the `pROC` package, we can also use the following (admittedly simpler) approach to plot the ROC curve, without `ggplot2`, and to obtain the C statistic, and a 95% confidence interval around that C statistic.

```
## requires pROC package
roc.modA <-
    roc(resect$died ~ predict(res_modA, type="response"),
        ci = TRUE)

roc.modA
```

```
Call:
roc.formula(formula = resect$died ~ predict(res_modA, type = "response"),     ci = TRUE

Data: predict(res_modA, type = "response") in 117 controls (resect$died 0) < 17 cases
Area under the curve: 0.7707
95% CI: 0.67-0.8715 (DeLong)
```

```
plot(roc.modA)
```



12.11 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.

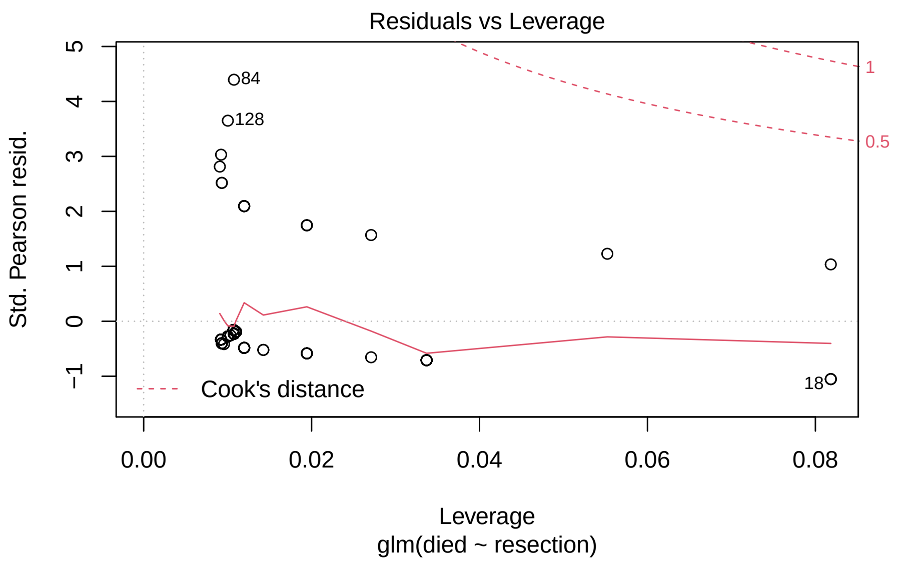
This is mitigated to some extent, however, by the fact that we are also making fewer distributional assumptions in logistic regression, so there is no need to inspect residuals for, say, skewness or heteroskedasticity.

- 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.

```r
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.

12.12 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
```

12.12.1 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)
```

```
# A tibble: 1 x 8
  null.deviance df.null logLik   AIC   BIC deviance df.residual  nobs
          <dbl>   <int>  <dbl> <dbl> <dbl>    <dbl>       <int> <int>
1          102.     133  -44.7  93.5  99.3     89.5         132   134
```

```
glance(res_modB)
```

```
# A tibble: 1 x 8
  null.deviance df.null logLik   AIC   BIC deviance df.residual  nobs
```

```
          <dbl>   <int>  <dbl> <dbl> <dbl>    <dbl>       <int> <int>
1          102.     133  -33.7  77.4  91.8     67.4          129   134
```

By either AIC or BIC, the larger model (`res_modB`) looks more effective.

12.12.2 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.

```
tidy(res_modB, exponentiate = TRUE, conf.int = TRUE) %>%
  select(term, estimate, conf.low, conf.high)
```

```
# A tibble: 5 x 4
  term        estimate conf.low conf.high
  <chr>          <dbl>    <dbl>     <dbl>
1 (Intercept)  0.00578 0.000241    0.0837
```

```
2 resection    1.84    1.08        3.35
3 age          1.00    0.962       1.04
4 prior        2.26    0.549       9.17
5 intubated   16.6     4.75       64.6
```

At a 5% significance level, we might conclude that:

- larger sized `resections` 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.

12.13 Plotting Model B

Let's think about plotting the fitted values from our model, in terms of probabilities.

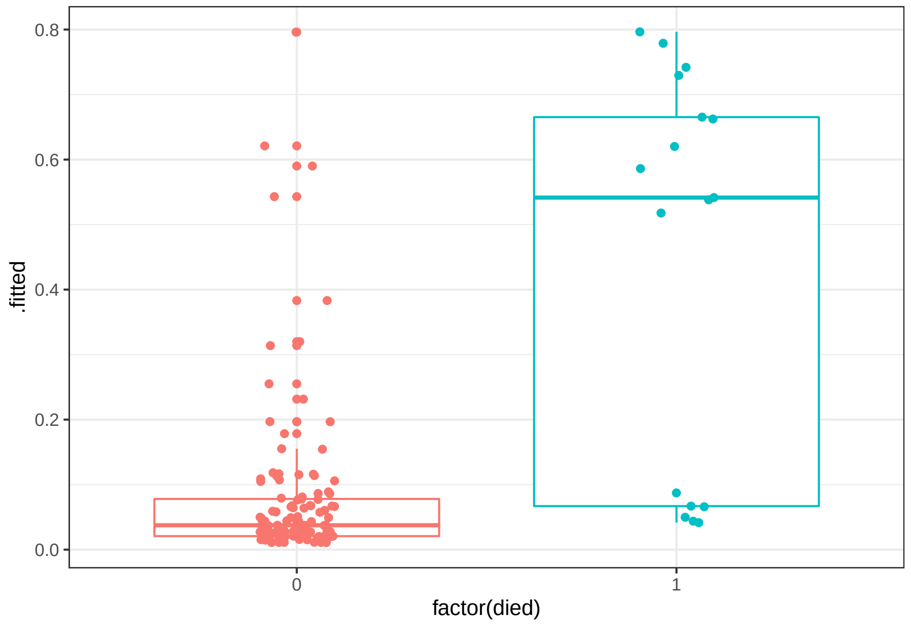
12.13.1 Using `augment` to capture the fitted probabilities

```
res_B_aug <- augment(res_modB, resect,
                     type.predict = "response")
head(res_B_aug)
```

```
# A tibble: 6 x 12
     id   age prior resection intubated  died .fitted .resid .std.resid   .hat
  <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>   <dbl>  <dbl>      <dbl>  <dbl>
1     1    34     1       2.5         0     0  0.0591 -0.349     -0.354 0.0267
2     2    57     0       5           0     0  0.117  -0.498     -0.508 0.0380
3     3    60     1       4           1     1  0.729   0.794      0.844 0.114
4     4    62     1       4.2         0     0  0.155  -0.581     -0.602 0.0704
5     5    28     0       6           1     1  0.796   0.675      0.724 0.131
6     6    52     0       3           0     0  0.0371 -0.275     -0.277 0.0105
# ... with 2 more variables: .sigma <dbl>, .cooksd <dbl>
```

12.13.2 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.

12.13.3 Confusion Matrix for Model B

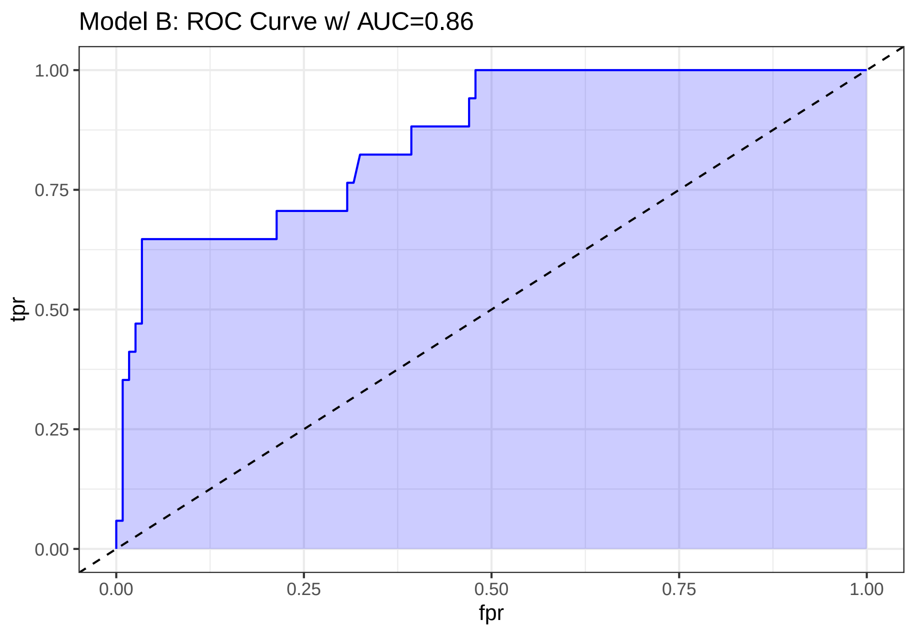
```
res_B_aug %$%
  confusionMatrix(
    data = factor(.fitted >= 0.5),
    reference = factor(died == 1),
    positive = "TRUE"
  )
```

```
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE   113    6
     TRUE      4   11
```

```
               Accuracy : 0.9254
                 95% CI : (0.867, 0.9636)
    No Information Rate : 0.8731
    P-Value [Acc > NIR] : 0.03897

                  Kappa : 0.6453

 Mcnemar's Test P-Value : 0.75183

            Sensitivity : 0.64706
            Specificity : 0.96581
         Pos Pred Value : 0.73333
         Neg Pred Value : 0.94958
             Prevalence : 0.12687
         Detection Rate : 0.08209
   Detection Prevalence : 0.11194
      Balanced Accuracy : 0.80644

       'Positive' Class : TRUE
```

12.13.4 The ROC curve for Model B

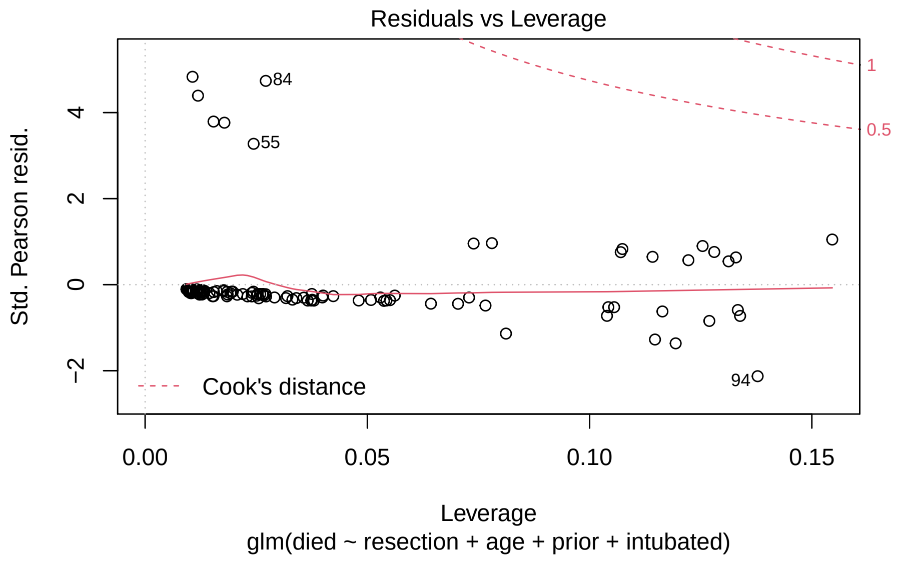
```r
## requires ROCR package
prob <- predict(res_modB, resect, type="response")
pred <- prediction(prob, resect$died)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
auc <- performance(pred, measure="auc")

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.

12.13.5 Residuals, Leverage and Influence

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



Again, we see no signs of deeply influential points in this model.

12.14 Logistic Regression using `lrm`

To obtain the Nagelkerke R^2 and the C statistic, as well as some other summaries, I'll now demonstrate the use of `lrm` from the `rms` package to fit a logistic regression model.

We'll return to the original model, predicting death using resection size alone.

```
dd <- datadist(resect)
options(datadist="dd")

res_modC <- lrm(died ~ resection, data=resect, x=TRUE, y=TRUE)
res_modC
```

```
Logistic Regression Model

 lrm(formula = died ~ resection, data = resect, x = TRUE, y = TRUE)

                      Model Likelihood    Discrimination    Rank Discrim.
                         Ratio Test           Indexes          Indexes
 Obs           134    LR chi2      12.45    R2        0.167    C       0.771
  0            117    d.f.             1    g         1.037    Dxy     0.541
  1             17    Pr(> chi2) 0.0004    gr        2.820    gamma   0.582
 max |deriv| 2e-06                             gp       0.110    tau-a   0.121
                                               Brier    0.103

           Coef    S.E.   Wald Z Pr(>|Z|)
 Intercept -4.4337 0.8799 -5.04  <0.0001
 resection  0.7417 0.2230  3.33  0.0009
```

This output specifies the following:

- `Obs` = The number of observations used to fit the model, with `0` = the number of zeros and `1` = the number of ones in our outcome, `died`. Also specified is the maximum absolute value of the derivative at the point where the maximum likelihood function was estimated. I wouldn't worry about that practically, as all you will care about is whether the iterative function-fitting process converged, and R will warn you in other ways if it doesn't.
- A likelihood ratio test (drop in deviance test) subtracting the residual deviance from the null deviance obtain the Likelihood Ratio χ^2 statistic, subtracting residual df from null df to obtain degrees of freedom, and comparing the resulting test statistic to a χ^2 distribution with the appropriate degrees of freedom to determine a p value.
- A series of discrimination indexes, including the Nagelkerke R^2, symbolized R2, and several others we'll discuss shortly.
- A series of rank discrimination indexes, including the C statistic (area under the ROC curve) and Somers' D (Dxy), and several others.
- A table of coefficients, standard errors, Wald Z statistics and p values based on those Wald statistics.

The C statistic is estimated to be 0.771, with an associated (Nagelkerke) R^2 = 0.167, both indicating at best mediocre performance for this model, as it turns out.

12.14.1 Interpreting Nagelkerke R^2

There are many ways to calculate R^2 for logistic regression.

- At the unfortunate URL linked here (unfortunate because the term "pseudo" is misspelled) there is a nice summary of the key issue, which is that there are at least three different ways to think about R^2 in linear regression that are equivalent in that context, but when you move to a categorical outcome, which interpretation you use leads you down a different path for extension to the new type of outcome.
- Paul Allison, for instance, describes several at this link in a post entitled "What's the Best R-Squared for Logistic Regression?"
- Jonathan Bartlett looks at McFadden's pseudo R^2 in some detail (including some R code) at this link, in a post entitled "R squared in logistic regression"

The Nagelkerke approach that is presented as `R2` in the `lrm` output is as good as most of the available approaches, and has the positive feature that it does reach 1 if the fitted model shows as much improvement as possible over the null model (which predicts the mean response for all subjects, and has $R^2 = 0$). The greater the improvement, the higher the Nagelkerke R^2.

For model A, our Nagelkerke $R^2 = 0.167$, which is pretty poor. It doesn't technically mean that 16.7% of any sort of variation has been explained, though.

12.14.2 Interpreting the C statistic and Plotting the ROC Curve

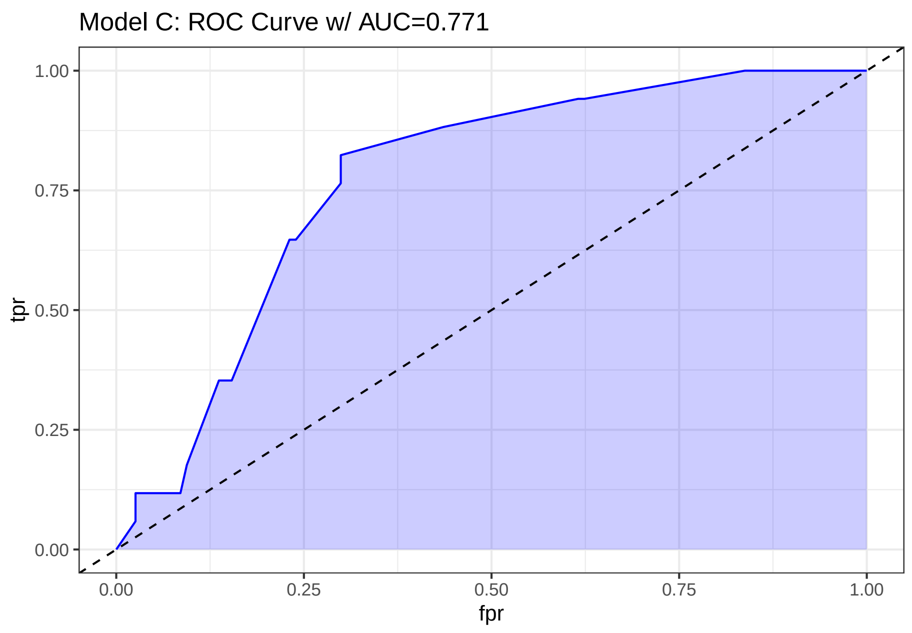
The C statistic is a measure of the area under the receiver operating characteristic curve. This link has some nice material that provides some insight into the C statistic and ROC curve.

- Recall that C ranges from 0 to 1. 0 = BAD, 1 = GOOD.
  - values of C less than 0.5 indicate that your prediction model is not even as good as simple random guessing of "yes" or "no" for your response.
  - C = 0.5 for random guessing
  - C = 1 indicates a perfect classification scheme - one that correctly guesses "yes" for all "yes" patients, and for none of the "no" patients.
- The closer C is to 1, the happier we'll be, most of the time.
  - Often we'll call models with $0.5 < C < 0.8$ poor or weak in terms of predictive ability by this measure
  - $0.8 \leq C < 0.9$ are moderately strong in terms of predictive power (indicate good discrimination)
  - $C \geq 0.9$ usually indicates a very strong model in this regard (indicate excellent discrimination)

We've seen the ROC curve for this model before, when we looked at model `res_modA` fitted using `glm` in the previous chapter. But, just for completeness, I'll include it.

Note. I change the initial `predict` call from `type = "response"` for a `glm` fit to `type = "fitted"` in a `lrm` fit. Otherwise, this is the same approach.

```r
## requires ROCR package
prob <- predict(res_modC, resect, type="fitted")
pred <- prediction(prob, resect$died)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
auc <- performance(pred, measure="auc")

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 C: ROC Curve w/ AUC=", auc)) +
    theme_bw()
```



12.14.3 The C statistic and Somers' D

- The C statistic is directly related to **Somers' D statistic**, abbreviated D_{xy}, by the equation C = 0.5 + (D/2).
  - Somers' D and the ROC area only measure how well predicted values from the model can rank-order the responses. For example, predicted probabilities of 0.01 and 0.99 for a pair of subjects are no better than probabilities of 0.2 and 0.8 using rank measures, if the first subject had a lower response value than the second.
  - Thus, the C statistic (or D_{xy}) may not be very sensitive ways to choose between models, even though they provide reasonable summaries of the models individually.
  - This is especially true when the models are strong. The Nagelkerke R^2 may be more sensitive.

- But as it turns out, we sometimes have to look at the ROC shapes, as the summary statistic alone isn't enough.

In our case, Somers D (Dxy) = .541, so the C statistic is 0.771.

12.14.4 Validating the Logistic Regression Model Summary Statistics

Like other regression-fitting tools in `rms`, the `lrm` function has a special `validate` tool to help perform resampling validation of a model, with or without backwards step-wise variable selection. Here, we'll validate our model's summary statistics using 100 bootstrap replications.

```
set.seed(432001)
validate(res_modC, B = 100)
```

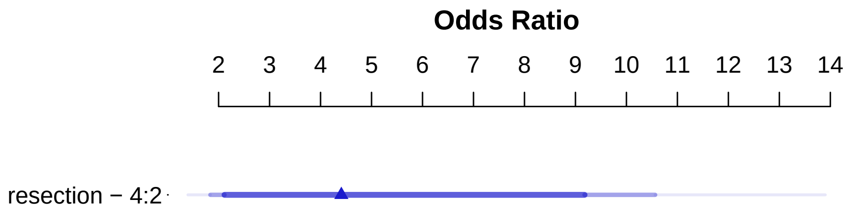
```
          index.orig training   test optimism index.corrected   n
Dxy           0.5415   0.5422 0.5415   0.0007          0.5408 100
R2            0.1666   0.1748 0.1666   0.0083          0.1583 100
Intercept     0.0000   0.0000 0.1631  -0.1631          0.1631 100
Slope         1.0000   1.0000 1.0463  -0.0463          1.0463 100
Emax          0.0000   0.0000 0.0428   0.0428          0.0428 100
D             0.0854   0.0909 0.0854   0.0055          0.0800 100
U            -0.0149  -0.0149 0.0017  -0.0167          0.0017 100
Q             0.1004   0.1058 0.0837   0.0221          0.0783 100
B             0.1025   0.0986 0.1051  -0.0065          0.1090 100
g             1.0369   1.0677 1.0369   0.0308          1.0061 100
gp            0.1101   0.1080 0.1101  -0.0021          0.1122 100
```

Recall that our area under the curve (C statistic) = `0.5 + (Dxy/2)`, so that we can also use the first row of statistics to validate the C statistic. Accounting for optimism in this manner, our validation-corrected estimates are Dxy = 0.5408, so C = 0.7704, and, from the second row of statistics, we can read off the validated Nagelkerke R^2, which is 0.1583.

12.14.5 Plotting the Summary of the `lrm` approach

The `summary` function applied to an `lrm` fit shows the effect size comparing the 25th to the 75th percentile of resection.

```
plot(summary(res_modC))
```



```
summary(res_modC)
```

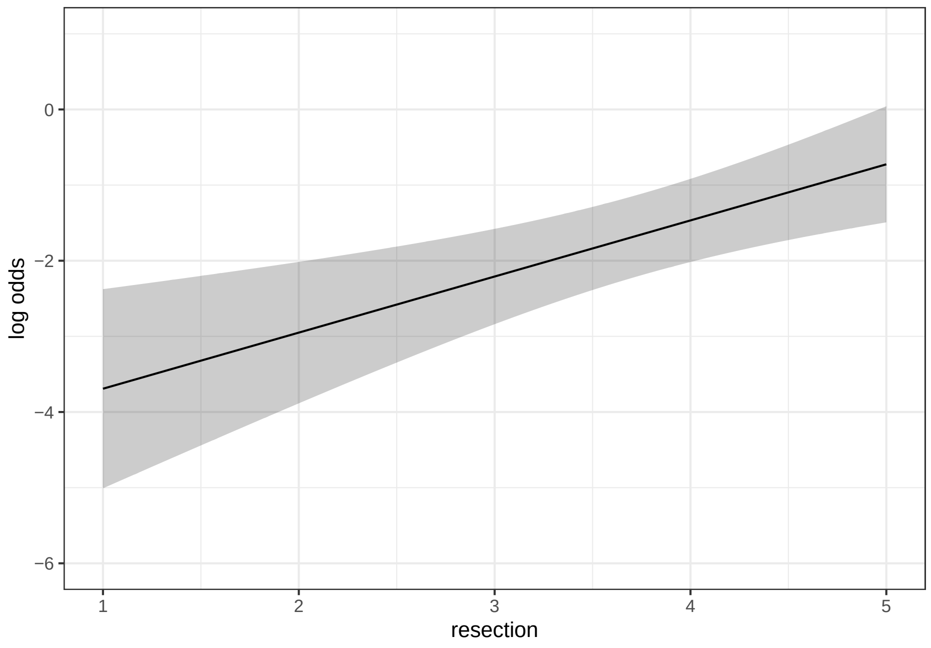
```
             Effects              Response : died

 Factor      Low High Diff. Effect S.E.    Lower 0.95 Upper 0.95
 resection   2   4    2     1.4834 0.44591 0.6094      2.3574
  Odds Ratio 2   4    2     4.4078      NA 1.8393     10.5630
```

So, a move from a resection of 2 cm to a resection of 4 cm is associated with an estimated effect on the log odds of death of 1.48 (with standard error 0.45), or with an estimated effect on the odds ratio for death of 4.41, with 95% CI (1.84, 10.56).

12.14.6 Plot In-Sample Predictions for Model C

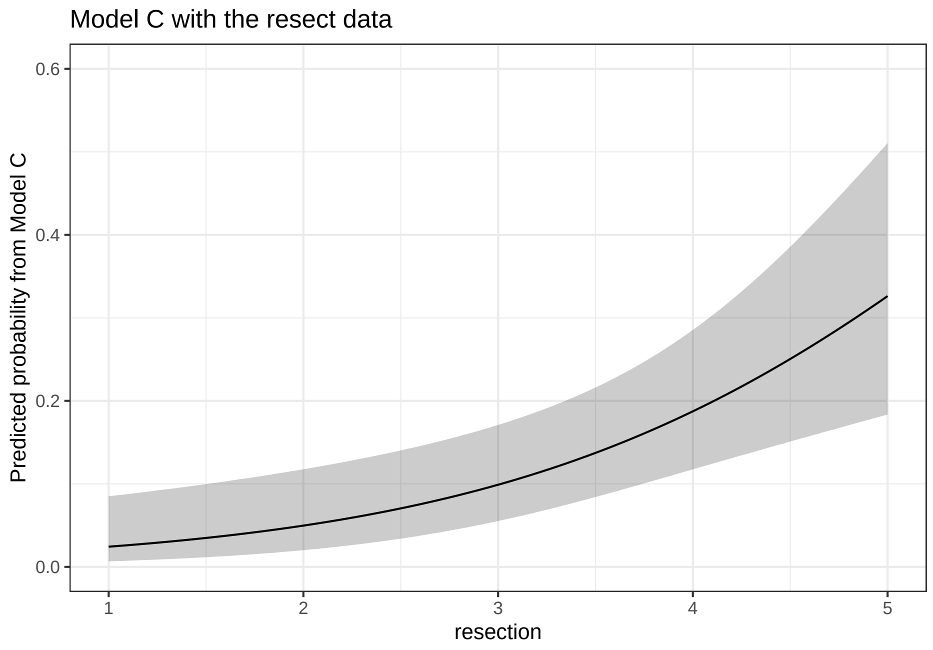
Here we plot the effect of `resection` (and 95% confidence intervals) across the range of observed values of `resection` on the log odds of death. Note the linear effect of `resection` size on the log odds scale.

```
ggplot(Predict(res_modC))
```



By applying the `plogis` function within the `Predict` command, we can plot the effect of `resection` on the estimated probability of death. Note the non-linear effect on this probability in this logistic regression model.

```
ggplot(Predict(res_modC, fun = plogis)) +
    labs(y = "Predicted probability from Model C",
         title = "Model C with the resect data")
```



The `Predict` function itself provides the raw material being captured in this plot.

```
head(Predict(res_modC, fun = plogis))
```

```
             resection       yhat       lower      upper .predictor.
resection.1   1.000000 0.02431476 0.006636502 0.08505223   resection
resection.2   1.020101 0.02467096 0.006789313 0.08559056   resection
resection.3   1.040201 0.02503224 0.006945549 0.08613277   resection
resection.4   1.060302 0.02539867 0.007105283 0.08667889   resection
resection.5   1.080402 0.02577033 0.007268589 0.08722896   resection
resection.6   1.100503 0.02614728 0.007435542 0.08778304   resection

Response variable (y):

Limits are 0.95 confidence limits
```

12.14.7 ANOVA from the `lrm` approach

```
anova(res_modC)
```

```
                Wald Statistics          Response: died

 Factor     Chi-Square d.f. P
 resection  11.07      1    9e-04
 TOTAL       11.07       1     9e-04
```

The ANOVA approach applied to a `lrm` fit provides a Wald test for the model as a whole. Here, the use of `resection` is a significant improvement over a null (intercept-only) model. The p value is 9 x 10^{-4}.

12.14.8 Are any points particularly influential?

I'll use a cutoff for `dfbeta` here of 0.3, instead of the default 0.2, because I want to focus on truly influential points. Note that we have to use the data frame version of `resect` as `show.influence` isn't tibble-friendly.

```
inf.C <- which.influence(res_modC, cutoff=0.3)
inf.C
```

```
$Intercept
[1]  84 128

$resection
[1] 84
```

```
show.influence(object = inf.C, dframe = data.frame(resect))
```

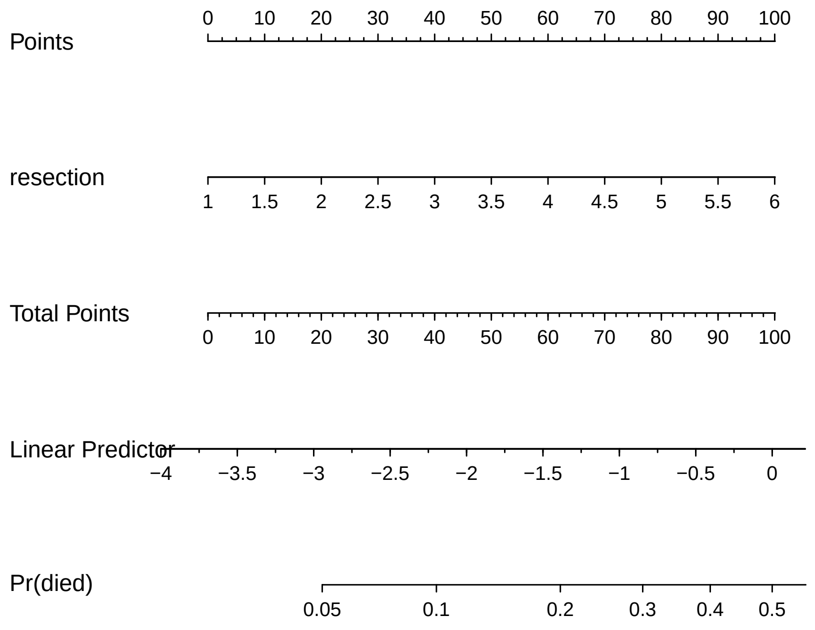
```
    Count resection
84      2      *2.0
128     1       2.5
```

It appears that observation 84 may have a meaningful effect on both the intercept and the coefficient for `resection`.

12.14.9 A Nomogram for Model C

We use the `plogis` function within a nomogram call to get R to produce fitted probabilities (of our outcome, `died`) in this case.

```
plot(nomogram(res_modC, fun=plogis,
              fun.at=c(0.05, seq(0.1, 0.9, by = 0.1), 0.95),
              funlabel="Pr(died)"))
```



Since there's no non-linearity in the right hand side of our simple logistic regression model, the nomogram is straightforward. We calculate the points based on the resection by traveling up, and then travel down in a straight vertical line from total points through the linear (log odds) predictor straight to a fitted probability. Note that fitted probabilities above 0.5 are not possible within the range of observed `resection` values in this case.

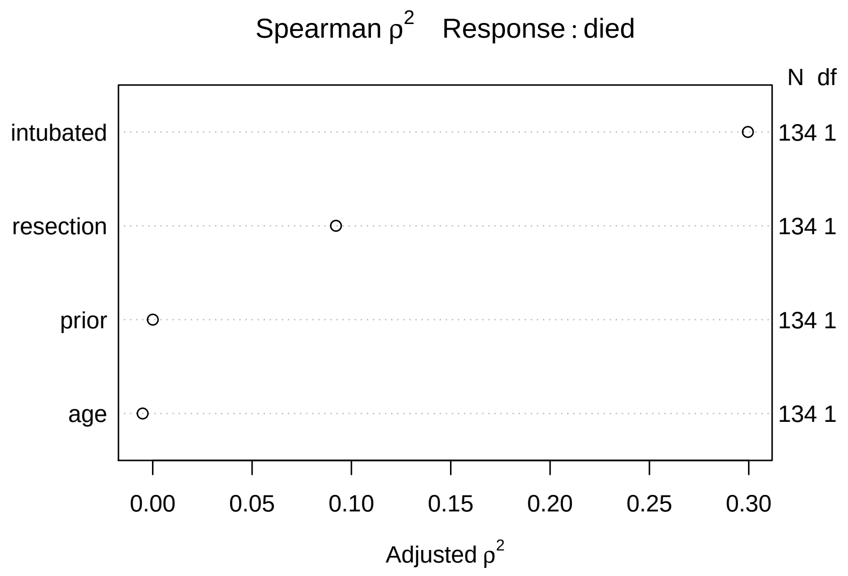
12.15 Model D: An Augmented Kitchen Sink Model

Can we predict survival from the patient's age, whether the patient had prior tracheal surgery or not, the extent of the resection, and whether intubation was required at the end of surgery?

12.15.1 Spearman ρ^2 Plot

Let's start by considering the limited use of non-linear terms for predictors that look important in a Spearman ρ^2 plot.

```
plot(spearman2(died ~ age + prior + resection + intubated, data=resect))
```



The most important variable appears to be whether intubation was required, so I'll include `intubated`'s interaction with the linear effect of the next most (apparently) important variable, `resection`, and also a cubic spline for `resection`, with three knots. Since `prior` and `age` look less important, I'll simply add them as linear terms.

12.15.2 Fitting Model D using `lrm`

Note the use of `%ia%` here. This insures that only the linear part of the `resection` term will be used in the interaction with `intubated`.

```
dd <- datadist(resect)
options(datadist="dd")

res_modD <- lrm(died ~ age + prior + rcs(resection, 3) +
                    intubated + intubated %ia% resection,
                data=resect, x=TRUE, y=TRUE)
```

12.15.3 Assessing Model D using `lrm`'s tools

```
res_modD
```

```
Logistic Regression Model

 lrm(formula = died ~ age + prior + rcs(resection, 3) + intubated +
     intubated %ia% resection, data = resect, x = TRUE, y = TRUE)

                       Model Likelihood    Discrimination    Rank Discrim.
                             Ratio Test           Indexes          Indexes
 Obs           134    LR chi2      38.08    R2       0.464    C       0.880
  0            117    d.f.             6    g        2.382    Dxy     0.759
  1             17    Pr(> chi2) <0.0001    gr      10.825    gamma   0.770
 max |deriv| 9e-08                          gp       0.172    tau-a   0.169
                                            Brier    0.067

                      Coef     S.E.   Wald Z Pr(>|Z|)
 Intercept            -11.3636 4.9099 -2.31  0.0206
 age                    0.0000 0.0210  0.00  0.9988
 prior                  0.6269 0.7367  0.85  0.3947
 resection              3.3799 1.9700  1.72  0.0862
 resection'            -4.2104 2.7035 -1.56  0.1194
 intubated              0.4576 2.7848  0.16  0.8695
 intubated * resection  0.6188 0.7306  0.85  0.3970
```

- The model likelihood ratio test suggests that at least some of these predictors are helpful.
- The Nagelkerke R^2 of 0.46, and the C statistic of 0.88 indicate a meaningful improvement in discrimination over our model with `resection` alone.
- The Wald Z tests see some potential need to prune the model, as none of the elements reaches statistical significance without the others. The product term between `intubated` and `resection`, in particular, doesn't appear to have helped much, once we already had the main effects.

12.15.4 ANOVA and Wald Tests for Model D

```
anova(res_modD)
```

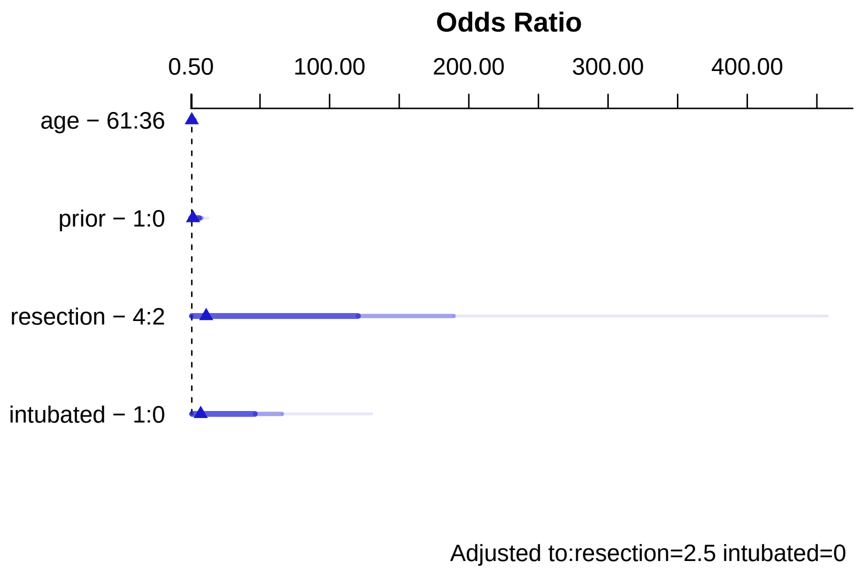
```
                Wald Statistics          Response: died

 Factor                                          Chi-Square d.f. P
 age                                              0.00      1    0.9988
 prior                                            0.72      1    0.3947
```

```
resection  (Factor+Higher Order Factors)            4.95      3    0.1753
 All Interactions                                   0.72      1    0.3970
 Nonlinear                                          2.43      1    0.1194
intubated  (Factor+Higher Order Factors)           16.45      2    0.0003
 All Interactions                                   0.72      1    0.3970
intubated * resection  (Factor+Higher Order Factors)  0.72    1    0.3970
TOTAL NONLINEAR + INTERACTION                       2.56      2    0.2783
TOTAL                                              23.90      6    0.0005
```

Neither the interaction term nor the non-linearity from the cubic spline appears to be statistically significant, based on the Wald tests via ANOVA. However it is clear that `intubated` has a significant impact as a main effect.

12.15.5 Effect Sizes in Model D

```
plot(summary(res_modD))
```



```
summary(res_modD)
```

```
             Effects              Response : died

 Factor      Low High Diff. Effect      S.E.    Lower 0.95 Upper 0.95
 age         36  61   25    -0.00080933 0.52409 -1.02800     1.0264
  Odds Ratio 36  61   25     0.99919000      NA  0.35772     2.7910
 prior        0   1    1     0.62693000 0.73665 -0.81688     2.0707
```

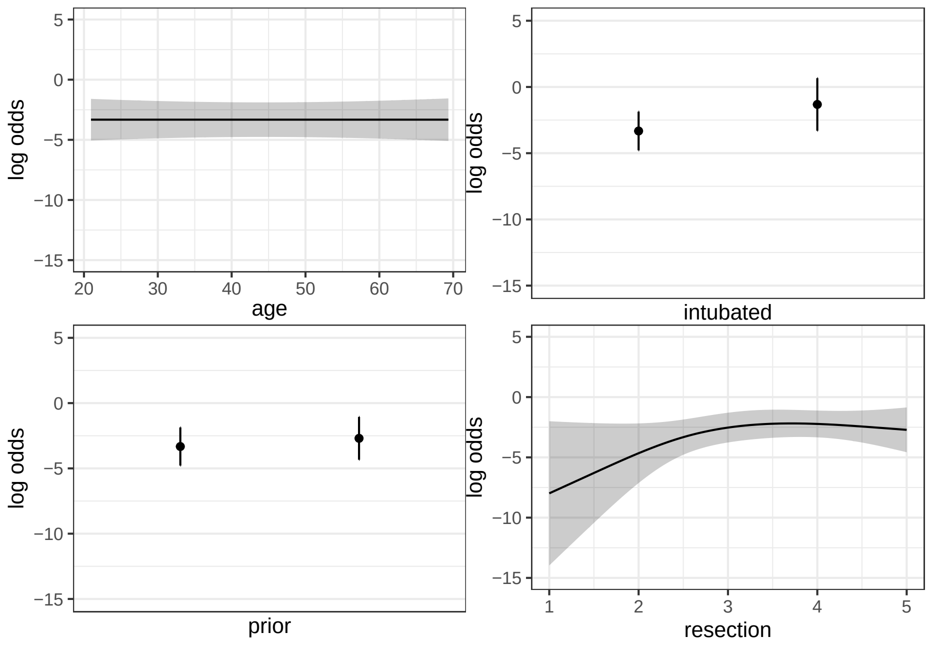
```
  Odds Ratio  0   1    1     1.87190000       NA  0.44181     7.9307
 resection    2   4    2     2.42930000 1.43510 -0.38342     5.2419
  Odds Ratio  2   4    2    11.35000000       NA  0.68153   189.0400
 intubated    0   1    1     2.00470000 1.11220 -0.17513     4.1845
  Odds Ratio  0   1    1     7.42380000       NA  0.83934    65.6610

Adjusted to: resection=2.5 intubated=0
```

The effect sizes are perhaps best described in terms of odds ratios. The odds ratio for death isn't significantly different from 1 for any variable, but the impact of `resection` and `intubated`, though not strong enough to be significant, look more substantial (if poorly estimated) than the effects of `age` and `prior`.

12.15.6 Plot In-Sample Predictions for Model D

Here are plots of the effects across the range of each predictor (holding the others constant) on the log odds scale. Note the non-linear effect of resection implied by the use of a spline there.

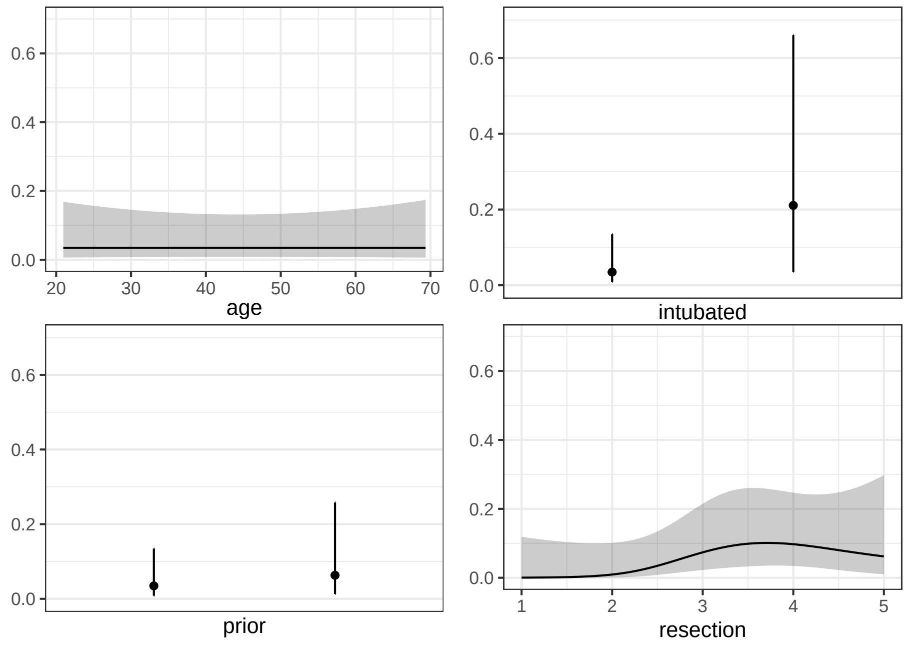
```
ggplot(Predict(res_modD))
```



We can also capture and plot these results on the probability scale, as follows[1].

[1]Although I've yet to figure out how to get the y axis relabeled properly without simply dumping the Predict results into a new tibble and starting over with creating the plots.

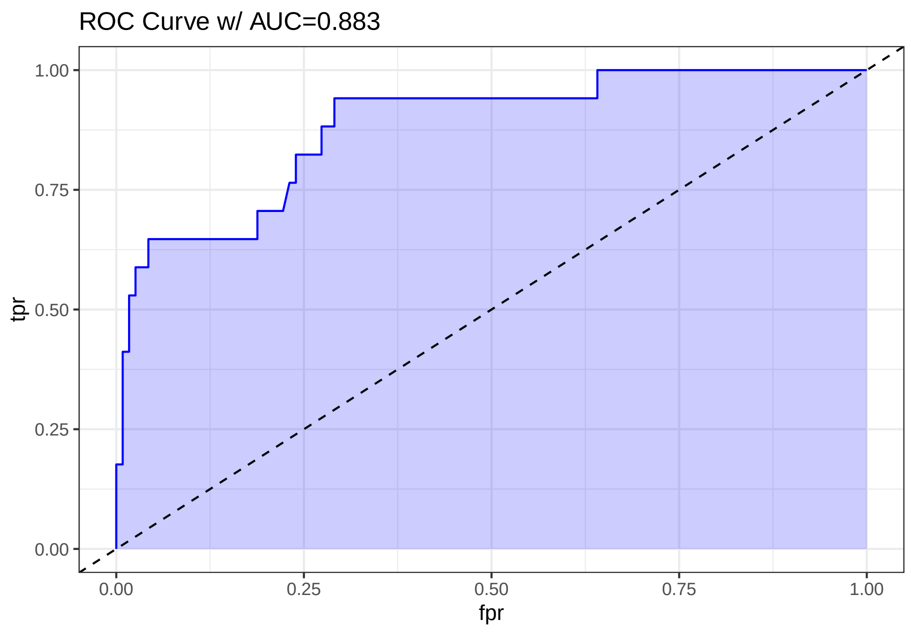
```r
ggplot(Predict(res_modD, fun = plogis))
```



12.15.7 Plotting the ROC curve for Model D

Again, remember to use `type = "fitted"` with a `lrm` fit.

```r
## requires ROCR package
prob <- predict(res_modD, resect, type="fitted")
pred <- prediction(prob, resect$died)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
auc <- performance(pred, measure="auc")

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()
```

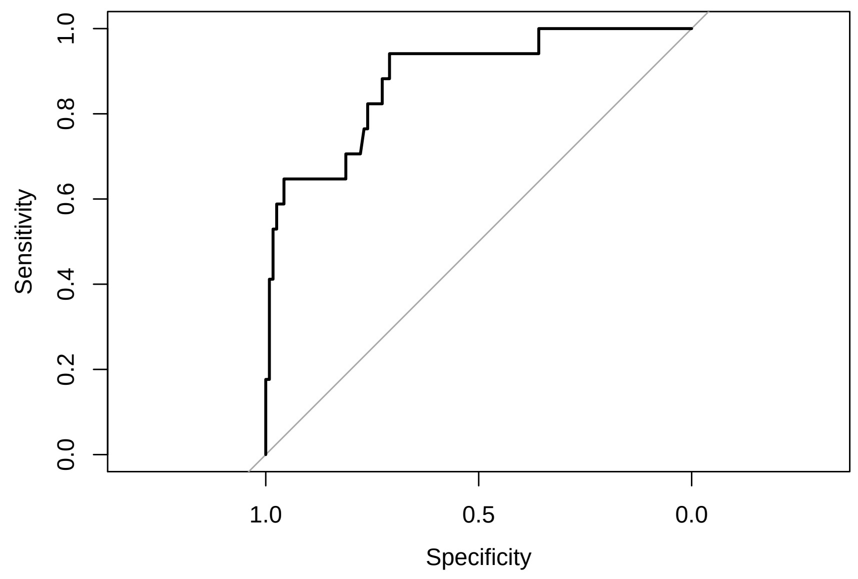


The AUC fitted with `ROCR` (0.883) is slightly different than what `lrm` has told us (0.880), and this also happens if we use the `pROC` approach, demonstrated below.

```
## requires pROC package
roc.modD <-
    roc(resect$died ~ predict(res_modD, type="fitted"),
        ci = TRUE)

roc.modD
```

```
Call:
roc.formula(formula = resect$died ~ predict(res_modD, type = "fitted"),     ci = TRUE)

Data: predict(res_modD, type = "fitted") in 117 controls (resect$died 0) < 17 cases (resect$died
Area under the curve: 0.8826
95% CI: 0.7952-0.97 (DeLong)
```

```
plot(roc.modD)
```



12.15.8 Validation of Model D summaries

```
set.seed(432002)
validate(res_modD, B = 100)
```

```
Divergence or singularity in 5 samples

          index.orig training    test optimism index.corrected  n
Dxy           0.7652   0.8080  0.7352   0.0727          0.6925 95
R2            0.4643   0.5347  0.4119   0.1228          0.3416 95
Intercept     0.0000   0.0000 -0.3533   0.3533         -0.3533 95
Slope         1.0000   1.0000  0.7658   0.2342          0.7658 95
Emax          0.0000   0.0000  0.1308   0.1308          0.1308 95
D             0.2767   0.3415  0.2407   0.1008          0.1759 95
U            -0.0149  -0.0149  0.0883  -0.1032          0.0883 95
Q             0.2916   0.3564  0.1524   0.2040          0.0876 95
B             0.0673   0.0640  0.0736  -0.0096          0.0769 95
g             2.3819   4.0387  2.4635   1.5751          0.8068 95
gp            0.1720   0.1910  0.1632   0.0278          0.1442 95
```

The C statistic indicates fairly strong discrimination, at C = 0.88, although after validation, this looks much weaker (based on Dxy = 0.6925, we would have C = 0.5 + 0.6925/2 = 0.85) and the Nagelkerke R^2 is also reasonably good, at 0.46, although this, too, is overly optimistic, and we bias-correct through our validation study to 0.34.

12.16 Model E: Fitting a Reduced Model in light of Model D

Can you suggest a reduced model (using a subset of the independent variables in model D) that adequately predicts survival?

Based on the anova for model D and the Spearman rho-squared plot, it appears that a two-predictor model using intubation and resection may be sufficient. Neither of the other potential predictors shows a statistically detectable effect in its Wald test.

```r
res_modE <- lrm(died ~ intubated + resection, data=resect,
                x=TRUE, y=TRUE)
res_modE
```

```
Logistic Regression Model

 lrm(formula = died ~ intubated + resection, data = resect, x = TRUE,
     y = TRUE)

                        Model Likelihood    Discrimination    Rank Discrim.
                              Ratio Test           Indexes          Indexes
 Obs            134    LR chi2      33.27    R2       0.413    C       0.867
  0             117    d.f.             2    g        1.397    Dxy     0.734
  1              17    Pr(> chi2) <0.0001    gr       4.043    gamma   0.757
 max |deriv| 5e-10                           gp       0.160    tau-a   0.164
                                             Brier    0.073

           Coef    S.E.   Wald Z Pr(>|Z|)
 Intercept -4.6370 1.0430 -4.45  <0.0001
 intubated  2.8640 0.6479  4.42  <0.0001
 resection  0.5475 0.2689  2.04  0.0418
```

The model equation is that the log odds of death is -4.637 + 2.864 `intubated` + 0.548 `resection`.

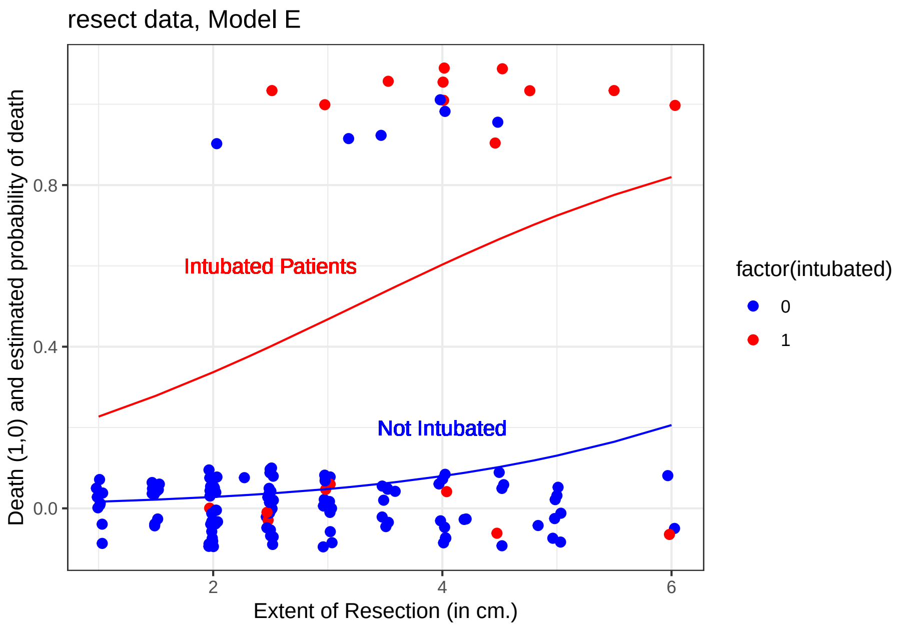
This implies that:

- for intubated patients, the equation is -1.773 + 0.548 `resection`, while
- for non-intubated patients, the equation is -4.637 + 0.548 `resection`.

We can use the `ilogit` function within the `faraway` package to help plot this.

12.16.1 A Plot comparing the two intubation groups

```
ggplot(resect, aes(x = resection, y = died,
                   col = factor(intubated))) +
    scale_color_manual(values = c("blue", "red")) +
    geom_jitter(size = 2, height = 0.1) +
    geom_line(aes(x = resection,
                  y = faraway::ilogit(-4.637 + 0.548*resection)),
              col = "blue") +
    geom_line(aes(x = resection,
                  y = faraway::ilogit(-1.773 + 0.548*resection)),
              col = "red") +
    geom_text(x = 4, y = 0.2, label = "Not Intubated",
              col="blue") +
    geom_text(x = 2.5, y = 0.6, label = "Intubated Patients",
              col="red") +
    labs(x = "Extent of Resection (in cm.)",
         y = "Death (1,0) and estimated probability of death",
         title = "resect data, Model E")
```

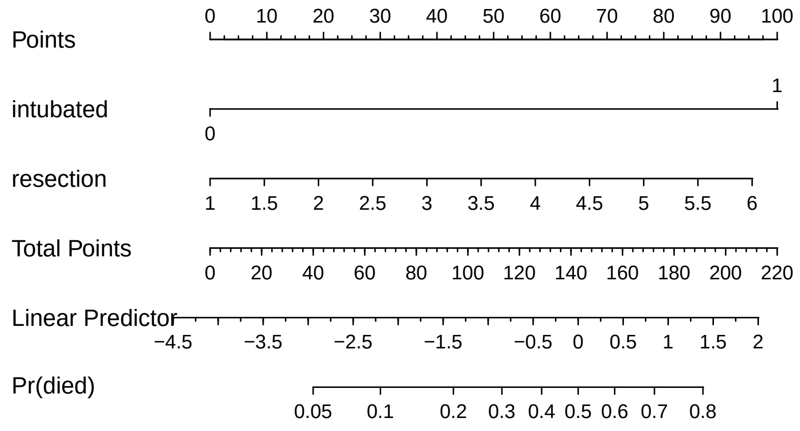


The effect of `intubation` appears to be very large, compared to the resection size effect.

12.16.2 Nomogram for Model E

A nomogram of the model would help, too.

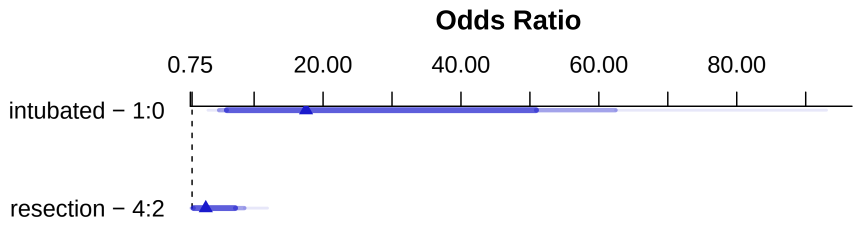
```
plot(nomogram(res_modE, fun=plogis,
              fun.at=c(0.05, seq(0.1, 0.9, by=0.1), 0.95),
              funlabel="Pr(died)"))
```



Again, we see that the effect of intubation is enormous, compared to the effect of resection. Another way to see this is to plot the effect sizes directly.

12.16.3 Effect Sizes from Model E

```
plot(summary(res_modE))
```



c effect plot-1.pdf

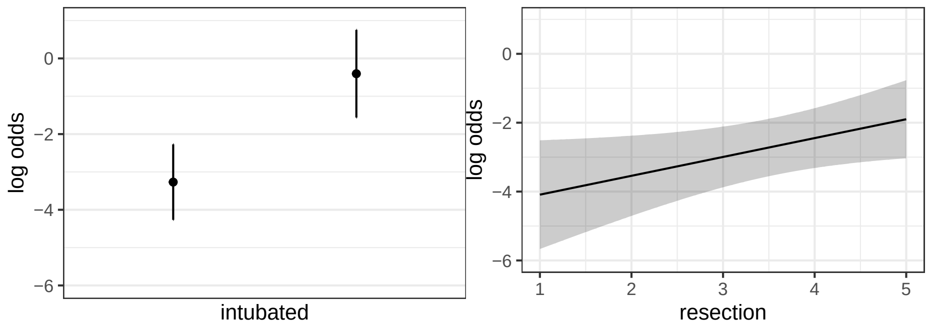
```
summary(res_modE)
```

```
             Effects              Response : died

 Factor      Low High Diff. Effect  S.E.    Lower 0.95 Upper 0.95
 intubated   0   1    1      2.8640 0.64790 1.59410     4.1338
  Odds Ratio 0   1    1     17.5310      NA 4.92390    62.4160
 resection   2   4    2      1.0949 0.53783 0.04082     2.1491
  Odds Ratio 2   4    2      2.9890      NA 1.04170     8.5769
```

12.16.4 Plot In-Sample Predictions for Model E

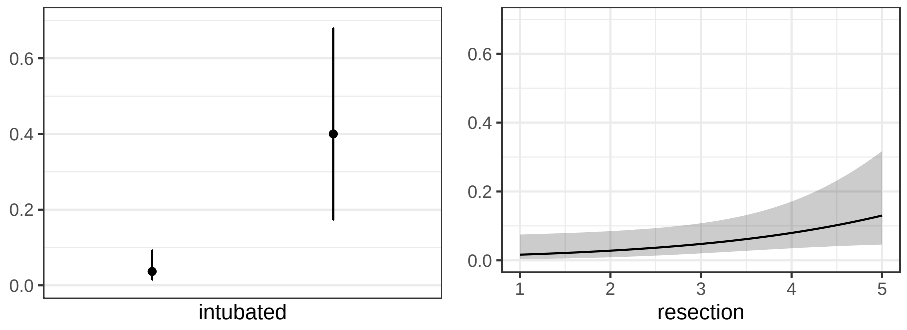
Here are plots of the effects across the range of each predictor (holding the other constant) on the log odds scale.

```
ggplot(Predict(res_modE))
```



We can also capture and plot these results on the probability scale, as follows.

```
ggplot(Predict(res_modE, fun = plogis))
```



12.16.5 ANOVA for Model E

```
anova(res_modE)
```

```
                Wald Statistics          Response: died 

 Factor     Chi-Square d.f. P     
 intubated  19.54      1    <.0001
 resection   4.14      1    0.0418
 TOTAL      25.47      2    <.0001
```

12.16.6 Validation of Model E

```
validate(res_modE, method="boot", B=40)
```

```
          index.orig training    test optimism index.corrected  n
Dxy           0.7340   0.6896  0.7326  -0.0430          0.7771 40
R2            0.4128   0.3814  0.4025  -0.0211          0.4339 40
Intercept     0.0000   0.0000  0.1367  -0.1367          0.1367 40
Slope         1.0000   1.0000  1.0472  -0.0472          1.0472 40
Emax          0.0000   0.0000  0.0369   0.0369          0.0369 40
D             0.2408   0.2183  0.2339  -0.0157          0.2565 40
U            -0.0149  -0.0149 -0.0001  -0.0148         -0.0001 40
Q             0.2558   0.2332  0.2340  -0.0009          0.2566 40
B             0.0727   0.0727  0.0759  -0.0032          0.0759 40
g             1.3970   1.3391  1.3577  -0.0186          1.4156 40
gp            0.1597   0.1446  0.1563  -0.0117          0.1714 40
```

Our bootstrap validated assessments of discrimination and goodness of fit look somewhat more reasonable now.

12.16.7 Do any points seem particularly influential?

As a last step, I'll look at influence, and residuals, associated with model E.

```
inf.E <- which.influence(res_modE, cutoff=0.3)

inf.E
```

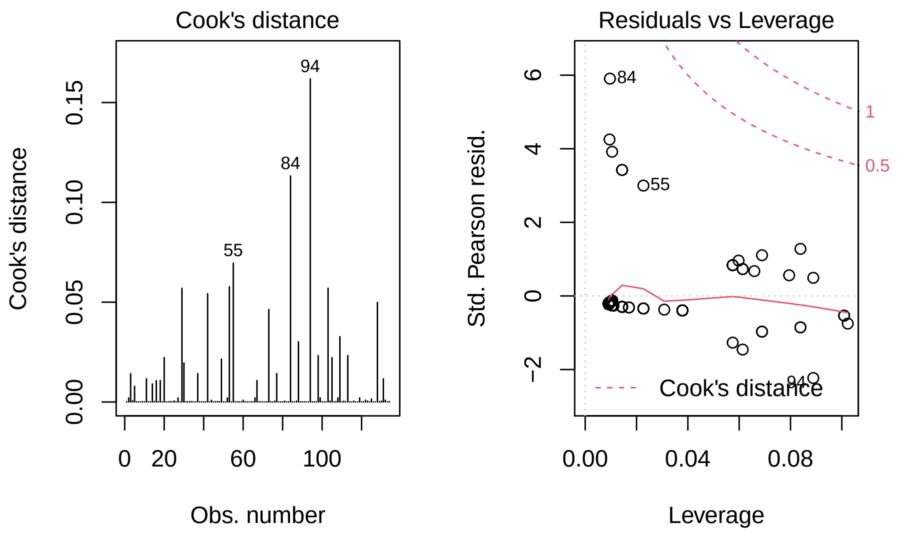
```
$Intercept
[1] 84 94

$resection
[1] 84 94
```

```
show.influence(inf.E, dframe = data.frame(resect))
```

```
   Count resection
84     2        *2
94     2        *6
```

12.16.8 Fitting Model E using `glm` to get plots about influence

```
res_modEglm <- glm(died ~ intubated + resection,
                   data=resect, family="binomial")
par(mfrow=c(1,2))
plot(res_modEglm, which=c(4:5))
```



Using this `glm` residuals approach, we again see that points 84 and 94 have the largest influence on our model E.

12.17 Concordance: Comparing Model C, D and E's predictions

To start, we'll gather the predictions fomade by each model (C, D and E) on the probability scale, in one place. Sadly, `augment` from `broom` doesn't work well with `lrm` fits, so we have to do this on our own.

```
resect_preds <- resect %>%
    mutate(C = predict(res_modC, type = "fitted"),
           D = predict(res_modD, type = "fitted"),
           E = predict(res_modE, type = "fitted"))

head(resect_preds)
```

```
# A tibble: 6 x 9
     id   age prior resection intubated  died      C      D      E
  <dbl> <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>
1     1    34     1       2.5         0     0 0.0705 0.0632 0.0367
2     2    57     0       5           0     0 0.326  0.0620 0.130
3     3    60     1       4           1     1 0.187  0.791  0.603
4     4    62     1       4.2         0     0 0.211  0.158  0.0881
5     5    28     0       6           1     1 0.504  0.711  0.819
6     6    52     0       3           0     0 0.0990 0.0737 0.0477
```

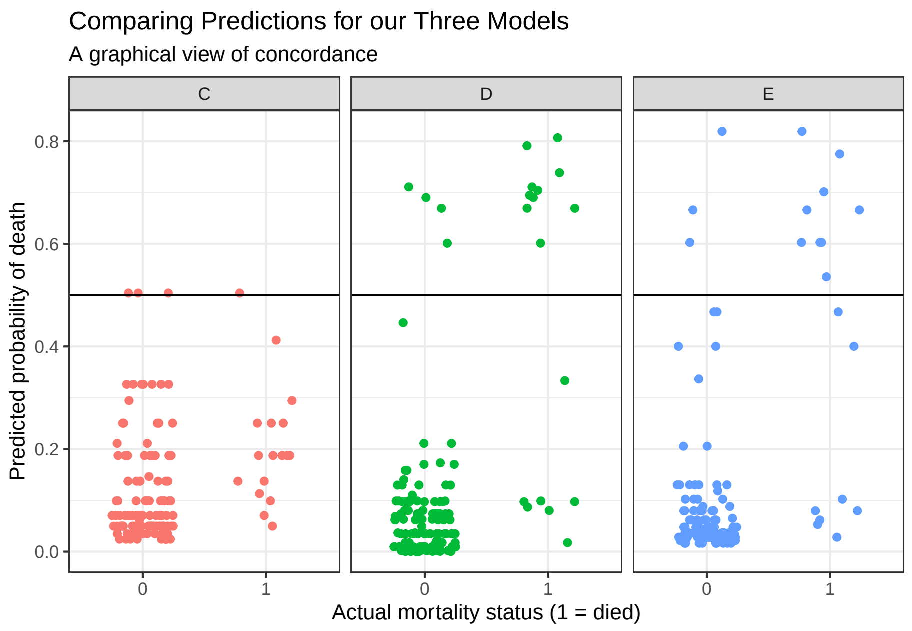
And now, we'll use the `gather` command to arrange the models and predicted probabilities in a more useful manner for plotting.

```
res_p <- resect_preds %>%
    gather("model", "prediction", 7:9) %>%
    select(id, died, model, prediction)

head(res_p)
```

```
# A tibble: 6 x 4
     id  died model prediction
  <dbl> <dbl> <chr>      <dbl>
1     1     0 C         0.0705
2     2     0 C         0.326
3     3     1 C         0.187
4     4     0 C         0.211
5     5     1 C         0.504
6     6     0 C         0.0990
```

Here's the resulting plot.

```
ggplot(res_p, aes(x = factor(died), y = prediction,
                  group = model, col = model)) +
    geom_jitter(width = 0.25) +
    geom_hline(yintercept = 0.5) +
    facet_wrap( ~ model) +
    guides(color = FALSE) +
    labs(title = "Comparing Predictions for our Three Models",
         subtitle = "A graphical view of concordance",
         x = "Actual mortality status (1 = died)",
         y = "Predicted probability of death")
```



We could specify a particular rule, for example: if the predicted probability of death is 0.5 or greater, then predict "Died."

```
res_p$rule.5 <- ifelse(res_p$prediction >= 0.5,
                       "Predict Died", "Predict Alive")

ftable(table(res_p$model, res_p$rule.5, res_p$died))
```

```
                   0   1
                        
C Predict Alive  114  16
  Predict Died     3   1
D Predict Alive  113   7
  Predict Died     4  10
E Predict Alive  114   8
  Predict Died     3   9
```

And perhaps build the linked table of row probabilities...

```
round(100*prop.table(
    ftable(table(res_p$model, res_p$rule.5, res_p$died))
    ,1),2)
```

```
                     0      1
```

```
C Predict Alive  87.69 12.31
  Predict Died   75.00 25.00
D Predict Alive  94.17  5.83
  Predict Died   28.57 71.43
E Predict Alive  93.44  6.56
  Predict Died   25.00 75.00
```

For example, in model E, 93.44% of those predicted to be alive actually survived, and 75% of those predicted to die actually died.

- Model D does a little better in one direction (94.17% of those predicted by Model D to be alive actually survived) but worse in the other (71.43% of those predicted by Model D to die actually died.)
- Model C does worse than each of the others in both predicting those who survive and those who die.

Note that the approaches discussed here would be useful if we had a new sample to predict on, as well. We could then compare the errors for that new data made by this sort of classification scheme either graphically or in a table.

12.18 Conclusions

It appears that `intubated` status and, to a lesser degree, the extent of the `resection` both play a meaningful role in predicting death associated with tracheal carina resection surgery. Patients who are intubated are associated with worse outcomes (greater risk of death) and more extensive resections are also associated with worse outcomes.

Chapter 13

A Study of Prostate Cancer

13.1 Data Load and Background

The data in `prost.csv` is derived from Stamey and others (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
     <dbl>  <dbl>  <dbl>   <dbl> <dbl> <chr>  <dbl> <dbl> <chr>   <dbl>
 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.

13.2 Code Book

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

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

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.

13.3 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)
```

```
Rows: 97
Columns: 16
$ subject   <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17...
$ lpsa      <dbl> -0.4307829, -0.1625189, -0.1625189, -0.1625189, 0.3715636...
$ lcavol    <dbl> -0.5798185, -0.9942523, -0.5108256, -1.2039728, 0.7514161...
$ lweight   <dbl> 2.769459, 3.319626, 2.691243, 3.282789, 3.432373, 3.22882...
$ age       <dbl> 50, 58, 74, 58, 62, 50, 64, 58, 47, 63, 65, 63, 63, 67, 5...
$ bph       <chr> "Low", "Low", "Low", "Low", "Low", "Low", "Medium", "High...
$ svi       <dbl> 0, 0, 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, -1.386294...
$ gleason   <chr> "6", "6", "7", "6", "6", "6", "6", "6", "6", "6", "6", "6...
$ pgg45     <dbl> 0, 0, 20, 0, 0, 0, 0, 0, 0, 0, 0, 0, 30, 5, 5, 0, 30, 0, ...
$ svi_f     <fct> No, No, No, No, No, No, No, No, No, No, No, No, No, No, N...
$ gleason_f <fct> 6, 6, 7, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 6, 7, 6, 6, ...
$ bph_f     <fct> Low, Low, Low, Low, Low, Low, Medium, High, Low, Low, Low...
$ lcavol_c  <dbl> -1.9298281, -2.3442619, -1.8608352, -2.5539824, -0.598593...
$ cavol     <dbl> 0.56, 0.37, 0.60, 0.30, 2.12, 0.35, 2.09, 2.00, 0.46, 1.2...
$ psa       <dbl> 0.65, 0.85, 0.85, 0.85, 1.45, 2.15, 2.15, 2.35, 2.85, 2.8...
```

13.4 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 `c11_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.].

```
c11_prost_A <- lm(lpsa ~ lcavol_c * svi, data = prost)
summary(c11_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
```

13.4.1 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(c11_prost_A)
```

```
# A tibble: 4 x 5
  term         estimate std.error statistic  p.value
  <chr>           <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    2.33      0.0913    25.5   8.25e-44
2 lcavol_c       0.586     0.0821     7.15  1.98e-10
3 svi            0.601     0.358      1.68  9.67e- 2
4 lcavol_c:svi   0.0648    0.266      0.243 8.08e- 1
```

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(c11_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.

13.4.2 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`, 0.6, 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`.

13.5 Exploring Model c11_prost_A

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

```
glance(c11_prost_A)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
```

```
1     0.581         0.567 0.759      42.9 1.68e-17     3  -109.  228.  241.
# ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

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

13.5.1 `summary` for Model `c11_prost_A`

If necessary, we can also run `summary` on this `c11_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(c11_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.

13.5.2 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.

13.5.3 Coefficient Confidence Intervals

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

```
confint(c11_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?

13.5.4 ANOVA for Model c11_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(c11_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.

13.5.5 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.

```
c11_prost_A_frame <- augment(c11_prost_A) %>% tbl_df
summary(c11_prost_A_frame)
```

```
      lpsa             lcavol_c             svi              .fitted
 Min.   :-0.4308   Min.   :-2.69708   Min.   :0.0000   Min.   :0.7498
 1st Qu.: 1.7317   1st Qu.:-0.83719   1st Qu.:0.0000   1st Qu.:1.8404
 Median : 2.5915   Median : 0.09691   Median :0.0000   Median :2.3950
 Mean   : 2.4784   Mean   : 0.00000   Mean   :0.2165   Mean   :2.4784
 3rd Qu.: 3.0564   3rd Qu.: 0.77703   3rd Qu.:0.0000   3rd Qu.:3.0709
 Max.   : 5.5829   Max.   : 2.47099   Max.   :1.0000   Max.   :4.5417
     .resid          .std.resid            .hat              .sigma
 Min.   :-1.6305   Min.   :-2.194508   Min.   :0.01316   Min.   :0.7423
 1st Qu.:-0.5007   1st Qu.:-0.687945   1st Qu.:0.01562   1st Qu.:0.7569
 Median : 0.1266   Median : 0.168917   Median :0.02498   Median :0.7617
 Mean   : 0.0000   Mean   : 0.001249   Mean   :0.04124   Mean   :0.7595
 3rd Qu.: 0.4886   3rd Qu.: 0.653612   3rd Qu.:0.04939   3rd Qu.:0.7631
 Max.   : 1.6847   Max.   : 2.261830   Max.   :0.24627   Max.   :0.7636
    .cooksd
 Min.   :0.0000069
 1st Qu.:0.0007837
 Median :0.0034699
 Mean   :0.0111314
 3rd Qu.:0.0103533
 Max.   :0.1341093
```

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.

13.5.6 Making Predictions with `c11_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(c11_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(c11_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
```

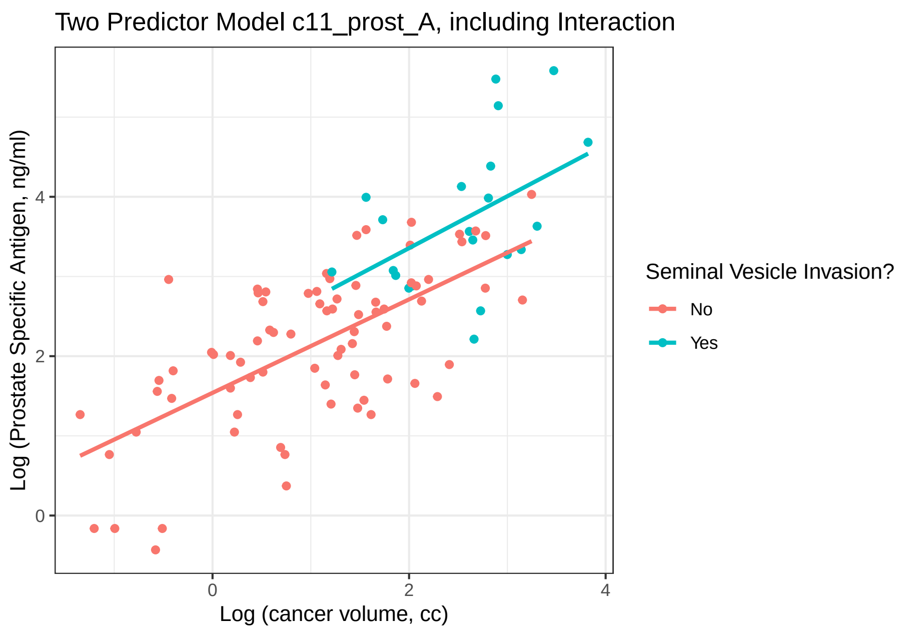
13.6 Plotting Model `c11_prost_A`

13.6.0.1 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 c11_prost_A, including Interaction")
```

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



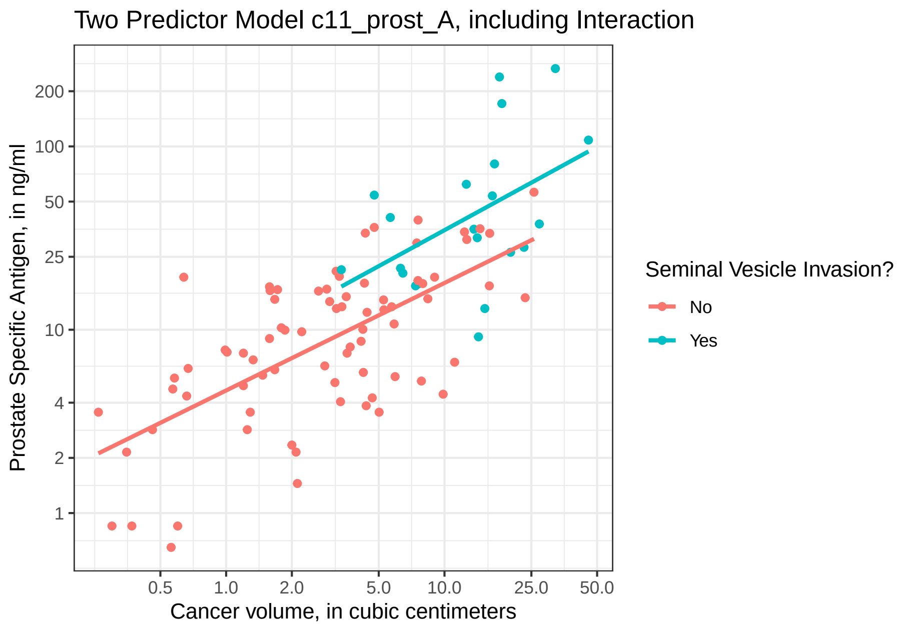
13.6.0.2 Plot on log-log scale

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 c11_prost_A, including Interaction")
```

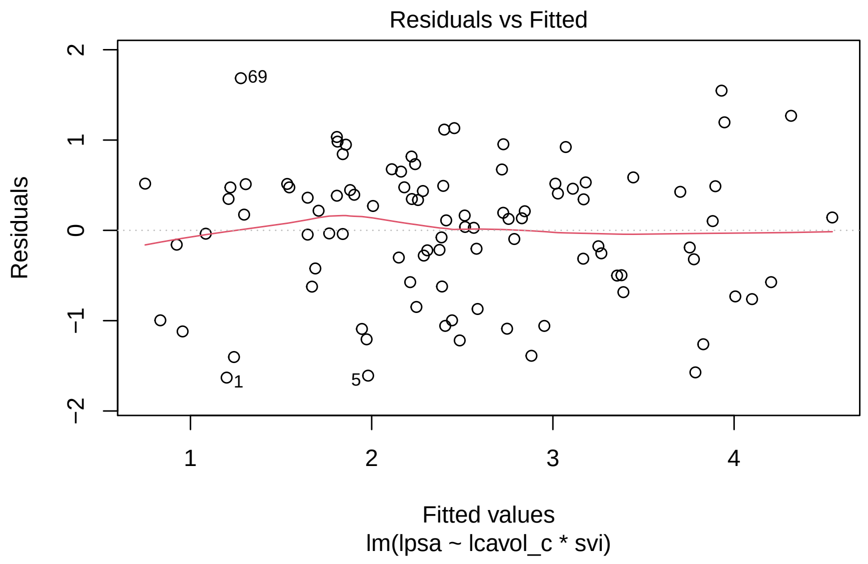
```
`geom_smooth()` using formula 'y ~ x'
```



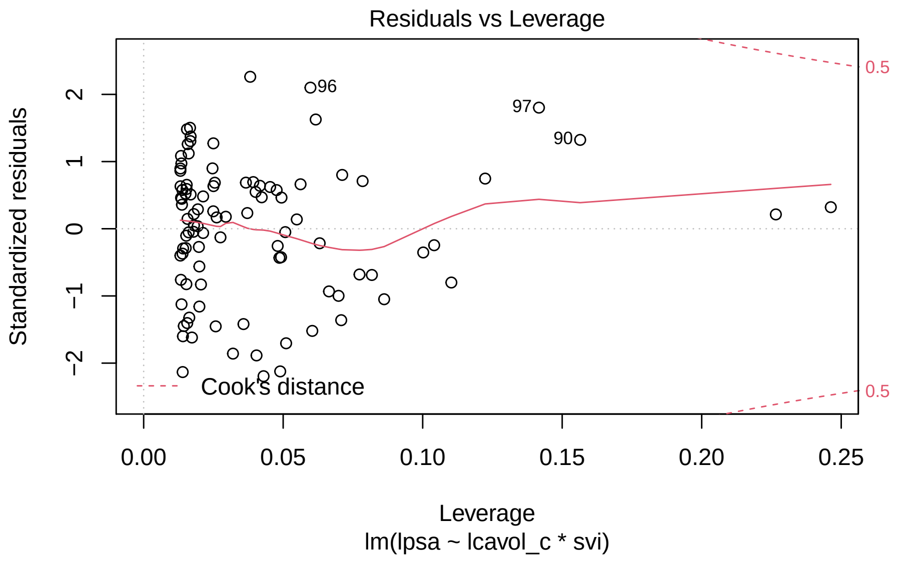
I've used the break point of 4 on the Y axis because of the old rule suggesting further testing for asymptomatic men with PSA of 4 or higher, but the other break points are arbitrary - they seemed to work for me, and used round numbers.

13.6.1 Residual Plots of `c11_prost_A`

```
plot(c11_prost_A, which = 1)
```



```
plot(c11_prost_A, which = 5)
```



13.7 Cross-Validation of Model `c11_prost_A`

Suppose we want to evaluate whether our model `c11_prost_A` predicts effectively in new data.

13.7.1 A Validation Split Approach

We'll first demonstrate a validation split approach (used, for instance, in 431) which splits our sample into a separate training (perhaps 70% of the data) and test (perhaps 30% of the data) samples, and then:

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

Our goal will be to cross-validate model `c11_prost_A`, which, you'll recall, uses `lcavol_c`, `svi` and their interaction, to predict `lpsa` in the `prost` data.

We'll start by identifying a random sample of 70% of our `prost` data in a training sample (which we'll call `prost_train`, and leave the rest as our test sample, called `prost_test`. To do this, we'll use the `createDataPartition` function from the `caret` package. We need only specify the data set and outcome variable, like so.

```
set.seed(4322020)
split_samples <- prost$lpsa %>%
    createDataPartition(p = 0.7, list = FALSE)

prost_train <- prost[split_samples,]
```

```
Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
Convert to a vector.
This warning is displayed once every 8 hours.
Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
prost_test <- prost[-split_samples,]
```

- Note the need for a comma after `split_samples` in the isolation of the training and test samples.
- Don't forget to pre-specify the random seed, for replicability, as I've done here.

Let's verify that we now have the samples we expect...

```
dim(prost_train)
```

```
[1] 70 16
```

```
dim(prost_test)
```

```
[1] 27 16
```

OK. Next, we'll run the `c11_prost_A` model in the training sample.

```
c11_prost_A_train <- prost_train %$%
    lm(lpsa ~ lcavol_c * svi)

c11_prost_A_train
```

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

Coefficients:
 (Intercept)      lcavol_c           svi  lcavol_c:svi
      2.2956        0.5919        0.6282        0.1597
```

Then we'll use the coefficients from this model to obtain predicted `lpsa` values in the test sample.

```
c11_prost_A_preds <-
    c11_prost_A_train %>% predict(prost_test)
```

```
c11_prost_A_preds[1:3]
```

```
       1        2        3 
1.647217 2.210338 2.408746 
```

Now, we can use the `postResample` function from the `caret` package to obtain several key summaries of fit quality for our model. Here, we specify the estimates (or predictions), and then the observed values to the `postResample` function.

```
postResample(c11_prost_A_preds, prost_test$lpsa)
```

```
     RMSE  Rsquared       MAE 
0.6651311 0.5503311 0.5524172 
```

These summary statistics are:

- the RMSE or root mean squared error, which measures the average difference (i.e. prediction error) between the observed known outcome values and the values predicted by the model by first squaring all of the errors, averaging them, and then taking the square root of the result. The lower the RMSE, the better the model.
- the Rsquared or R^2, which is just the square of the Pearson correlation coefficient relating the predicted and observed values, so we'd like this to be as large as possible, and
- the MAE or mean absolute error, which is a bit less sensitive to outliers than the RMSE, because it measures the average prediction error by taking the absolute value of each error, and then grabbing the average of those values. The lower the MAE, the better the model.

These statistics are more helpful, generally, for comparing multiple models to each other, than for making final decisions on their own. The `caret` package also provides individual functions to gather the elements of `postResample` as follows.

```
prost_A_summaries <- tibble(
    RMSE = RMSE(c11_prost_A_preds, prost_test$lpsa),
    R2 = R2(c11_prost_A_preds, prost_test$lpsa),
    MAE = MAE(c11_prost_A_preds, prost_test$lpsa)
)

prost_A_summaries
```

```
# A tibble: 1 x 3
   RMSE    R2   MAE
  <dbl> <dbl> <dbl>
1 0.665 0.550 0.552
```

13.7.2 K-Fold Cross-Validation

One problem with the validation split approach is that with a small data set like `prost`, 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 **K-fold cross-validation**, which involves partitioning our data into a series of K training-test subsets, and then combining the results. Specifically, we'll try a *5-fold cross validation* here. (K is usually taken to be either 5 or 10.)

The approach includes the following steps.

1. Randomly split the `prost` data into 5 subsets (for 5-fold validation).
2. Reserve one subset and train the model on all other subsets.
3. Test the model on the reserved subset and record the prediction error.
4. Repeat this process until each of the k subsets has served as the test set.
5. Compute the average of the k recorded errors. This is called the cross-validation error and serves as the primary performance metric for the model.

Again using tools from the `caret` packages, we'll first define our `trainControl` approach.

```
set.seed(43220201)
train.control <- trainControl(method = "cv", number = 5)
```

Then we train the model, and obtain the usual summaries of model fit quality.

```
c11_modelA_cv <- train(lpsa ~ lcavol_c * svi,
                  data = prost, method = "lm",
                  trControl = train.control)

c11_modelA_cv
```

```
Linear Regression

97 samples
 2 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 77, 77, 77, 79, 78
Resampling results:

  RMSE       Rsquared   MAE
  0.7583897  0.5657784  0.6213951

Tuning parameter 'intercept' was held constant at a value of TRUE
```

We can then look at the model fit by this cross-validation approach.

```
summary(c11_modelA_cv)
```

```
Call:
lm(formula = .outcome ~ ., data = dat)

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
```

or, if you prefer,

```
tidy(summary(c11_modelA_cv), conf.int = TRUE) %>%
    kable(digits = 3)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	2.331	0.091	25.540	0.000	2.150	2.513
lcavol_c	0.586	0.082	7.145	0.000	0.423	0.749
svi	0.601	0.358	1.678	0.097	-0.110	1.313
‘lcavol_c:svi‘	0.065	0.266	0.243	0.808	-0.464	0.593

and

```
glance(summary(c11_modelA_cv)) %>%
    kable(digits = c(3,3,3,2,3,0))
```

r.squared	adj.r.squared	sigma	statistic	p.value	df	df.residual	nobs
0.581	0.567	0.759	42.92	0	3	93	97

13.7.3 Comparing Models with 5-fold Cross-Validation

To make this a bit more realistic, let’s compare two models:

- our existing linear model lpsa ~ lcavol_c * svi, and

- a *robust* linear model fit with the `rlm` function in R, to predict `lpsa` using `lcavol_c` and `svi` but not the interaction between them.

The main purpose of *robust* linear models is to reduce the influence of specifically outlier or high leverage data points.

Here's that robust fit in the original `prost_train` data set. Note that fitting a robust linear model requires the choice of a `psi` (ψ) function, for which R provides three approaches, called the Huber, Hampel and Tukey bisquare approaches. In this fit, I'll just let R pick its default choice.

```
modelR <- prost_train %$% rlm(lpsa ~ lcavol_c + svi)

summary(modelR)
```

```
Call: rlm(formula = lpsa ~ lcavol_c + svi)
Residuals:
    Min      1Q  Median      3Q     Max 
-1.5904 -0.5869  0.1139  0.4902  1.7254 

Coefficients:
            Value   Std. Error t value
(Intercept)  2.3241  0.1356    17.1459
lcavol_c     0.6050  0.1139     5.3123
svi          0.7526  0.3296     2.2836

Residual standard error: 0.7819 on 67 degrees of freedom
```

Compare this with the standard ordinary least squares fit to the same data (again without the interaction term), and you'll see that in this case, the main differences are in the estimated standard errors, but the slope coefficients are also a bit smaller in the robust model.

```
modelO <- prost_train %$% lm(lpsa ~ lcavol_c + svi)

summary(modelO)
```

```
Call:
lm(formula = lpsa ~ lcavol_c + svi)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.5661 -0.5616  0.1128  0.5130  1.7535 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2.30169    0.11291  20.386  < 2e-16 ***
lcavol_c     0.60820    0.09487   6.411 1.69e-08 ***
```

```
svi          0.79147    0.27451   2.883  0.00529 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7948 on 67 degrees of freedom
Multiple R-squared:  0.5924,    Adjusted R-squared:  0.5803 
F-statistic:  48.7 on 2 and 67 DF,  p-value: 8.738e-14
```

So, how can we do 5-fold cross-validation on our model R, and also let the computer pick which of the three types of initial weights (Huber, Hampel or Tukey Bisquare) might be most appropriate? As follows...

```
c11_modelR_cv <- train(lpsa ~ lcavol_c + svi,
                   data = prost, method = "rlm",
                   trControl = train.control)

c11_modelR_cv
```

```
Robust Linear Model 

97 samples
 2 predictor

No pre-processing
Resampling: Cross-Validated (5 fold) 
Summary of sample sizes: 78, 77, 77, 78, 78 
Resampling results across tuning parameters:

  intercept  psi           RMSE       Rsquared   MAE      
  FALSE      psi.huber     2.1261462  0.3643515  1.8841211
  FALSE      psi.hampel    2.1253158  0.3652711  1.8837252
  FALSE      psi.bisquare  2.1260854  0.3560150  1.8818567
   TRUE      psi.huber     0.7683316  0.5741134  0.6358550
   TRUE      psi.hampel    0.7634386  0.5773235  0.6352990
   TRUE      psi.bisquare  0.7690475  0.5745092  0.6371126

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were intercept = TRUE and psi = psi.hampel.
```

Compare these RMSE, Rsquared and MAE values to those we observed in the interaction model with `lm` earlier...

```
c11_modelA_cv
```

```
Linear Regression 

97 samples
 2 predictor
```

```
No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 77, 77, 77, 79, 78
Resampling results:

  RMSE       Rsquared   MAE
  0.7583897  0.5657784  0.6213951

Tuning parameter 'intercept' was held constant at a value of TRUE
```

The robust model showes a larger R-Squared, smaller RMSE and smaller MAE than the interaction model. Perhaps we'll focus further on model R going forward…

```
summary(c11_modelR_cv)
```

```
Call: rlm(formula = .outcome ~ ., data = dat, psi = psi)
Residuals:
    Min      1Q  Median      3Q     Max
-1.6203 -0.5297  0.1191  0.4852  1.6939

Coefficients:
            Value   Std. Error t value
(Intercept)  2.3340  0.0939    24.8529
lcavol_c     0.5930  0.0807     7.3521
svi          0.6685  0.2296     2.9111

Residual standard error: 0.7483 on 94 degrees of freedom
```

Let's stop there for now. Next, we'll consider the problem of considering adding more predictors to a linear model, and then making sensible selections as to which predictors actually should be incorporated.

Chapter 14

Multiple Imputation and Linear Regression

14.1 Developing a `smart_16` data set

In this chapter, we'll return to the `smart_ohio` file based on data from BRFSS 2017 that we built and cleaned back at the start of these Notes.

```r
smart_ohio <- readRDS(here("data", "smart_ohio.Rds"))
```

We're going to look at a selection of variables from this tibble, among subjects who have been told they have diabetes, and who also provided a response to our `physhealth` (Number of Days Physical Health Not Good) variable, which asks "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?" We'll build two models. In this chapter, we'll look at a linear model for `physhealth` and in the next chapter, we'll look at a logistic regression describing whether or not the subject's `physhealth` response was at least 1.

```r
smart_16 <- smart_ohio %>%
    filter(dm_status == "Diabetes") %>%
    filter(complete.cases(physhealth)) %>%
    mutate(bad_phys = ifelse(physhealth > 0, 1, 0),
           comor = hx_mi + hx_chd + hx_stroke + hx_asthma +
               hx_skinc + hx_otherc + hx_copd + hx_arthr) %>%
    select(SEQNO, mmsa, physhealth, bad_phys, age_imp, smoke100,
           comor, hx_depress, bmi, activity)
```

The variables included in this `smart_16` tibble are:

Variable	Description
`SEQNO`	respondent identification number (all begin with 2016)
`mmsa`	
`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?
`bad_phys`	Is `physhealth` 1 or more?
`age_imp`	Age in years (imputed from age categories)
`smoke100`	Have you smoked at least 100 cigarettes in your life? (1 = yes, 0 = no)
`hx_depress`	Has a doctor, nurse, or other health professional ever told you that you have a depressive disorder, including depression, major depression, dysthymia, or minor depression?
`bmi`	Body mass index, in kg/m^2
`activity`	Physical activity (Highly Active, Active, Insufficiently Active, Inactive)
`comor`	Sum of 8 potential groups of comorbidities (see below)

The `comor` variable is the sum of the following 8 variables, each of which is measured on a 1 = Yes, 0 = No scale, and begin with "Has a doctor, nurse, or other health professional ever told you that you had ..."

- `hx_mi`: a heart attack, also called a myocardial infarction?
- `hx_chd`: angina or coronary heart disease?
- `hx_stroke`: a stroke?
- `hx_asthma`: asthma?
- `hx_skinc`: skin cancer?
- `hx_otherc`: any other types of cancer?
- `hx_copd`: Chronic Obstructive Pulmonary Disease or COPD, emphysema or chronic bronchitis?
- `hx_arthr`: some form of arthritis, rheumatoid arthritis, gout, lupus, or fibromyalgia?

```
smart_16 %>% tabyl(comor)
```

```
 comor   n     percent valid_percent
     0 224 0.211920530   0.221782178
     1 315 0.298013245   0.311881188
     2 228 0.215704825   0.225742574
     3 130 0.122989593   0.128712871
     4  72 0.068117313   0.071287129
     5  29 0.027436140   0.028712871
     6   9 0.008514664   0.008910891
     7   3 0.002838221   0.002970297
    NA  47 0.044465468            NA
```

14.1.1 Any missing values?

We have 1057 observations (rows) in the `smart_16` data set, of whom 860 have complete data on all variables.

```
dim(smart_16)
```

```
[1] 1057   10
```

```
n_case_complete(smart_16)
```

```
[1] 860
```

Which variables are missing?

```
miss_var_summary(smart_16)
```

```
# A tibble: 10 x 3
   variable   n_miss pct_miss
   <chr>       <int>    <dbl>
 1 activity       85    8.04
 2 bmi            84    7.95
 3 comor          47    4.45
 4 smoke100       24    2.27
 5 age_imp        12    1.14
 6 hx_depress      3    0.284
 7 SEQNO           0    0
 8 mmsa            0    0
 9 physhealth      0    0
10 bad_phys        0    0
```

Note that our outcomes (`physhealth` and the derived `bad_phys`) have no missing values here, by design. We will be performing multiple imputation to account appropriately for missingness in the predictors with missing values.

14.2 Obtaining a Simple Imputation with `mice`

The `mice` package provides several approaches we can use for imputation in building models of all kinds. Here, we'll use it just to obtain a single set of imputed results that we can apply to "complete" our data for the purposes of thinking about (a) transforming our outcome and (b) considering the addition of non-linear predictor terms.

```
# requires library(mice)

set.seed(432)

# create small data set including only variables to
# be used in building the imputation model

sm16 <- smart_16 %>%
    select(physhealth, activity, age_imp, bmi, comor,
           hx_depress, smoke100)

smart_16_mice1 <- mice(sm16, m = 1)
```

```
 iter imp variable
  1   1  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   1  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   1  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   1  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   1  activity  age_imp  bmi  comor  hx_depress  smoke100
```

```
smart_16_imp1 <- mice::complete(smart_16_mice1)

n_case_miss(smart_16_imp1)
```

```
[1] 0
```

And now we'll use this completed `smart_16_imp1` data set (the product of just a single imputation) to help us address the next two issues.

14.3 Linear Regression: Considering a Transformation of the Outcome

A plausible strategy here would be to try to identify an outcome transformation only after some accounting for missing predictor values, perhaps through a simple imputation approach. However, to keep things simple here, I'll just use the complete cases in this section.

Recall that our outcome here, `physhealth` can take the value 0, and is thus not strictly positive.

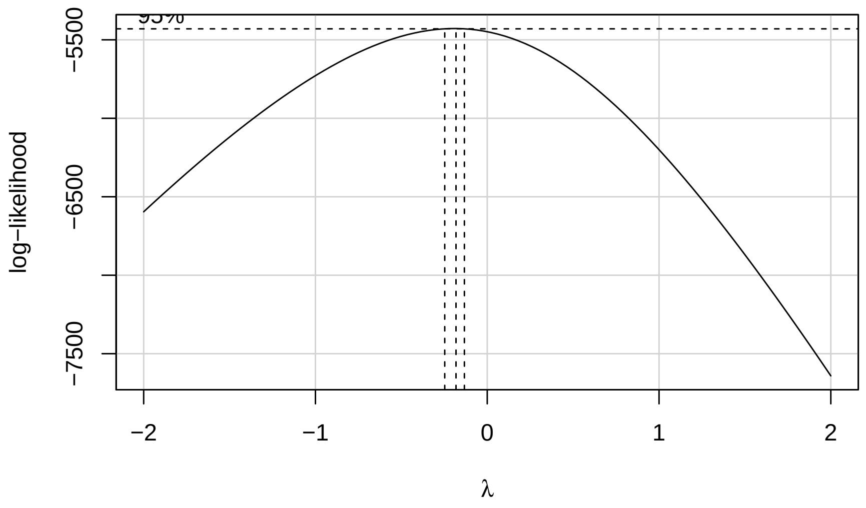
```
mosaic::favstats(~ physhealth, data = smart_16_imp1)
```

```
 min Q1 median Q3 max     mean       sd    n missing
   0  0      2 20  30 9.227058 11.92676 1057       0
```

So, if we want to investigate a potential transformation with a Box-Cox plot, we'll have to add a small value to each `physhealth` value. We'll add 1, so that the range of potential values is now from 1-31.

```
# requires library(car)

smart_16_imp1 %$%
```

```
    boxCox((physhealth + 1) ~ age_imp + comor + smoke100 +
               hx_depress + bmi + activity)
```



It looks like the logarithm is a reasonable transformation in this setting. So we'll create a new outcome, that is the natural logarithm of (`physhealth` + 1), which we'll call `phys_tr` to remind us that a transformation is involved that we'll eventually need to back out of to make predictions. We'll build this new variable in both our original `smart_16` data set and in the simply imputed data set we're using for just these early stages.

```
smart_16_imp1 <- smart_16_imp1 %>%
    mutate(phys_tr = log(physhealth + 1))

smart_16 <- smart_16 %>%
    mutate(phys_tr = log(physhealth + 1))
```

So we have `phys_tr` = log(`physhealth` + 1)

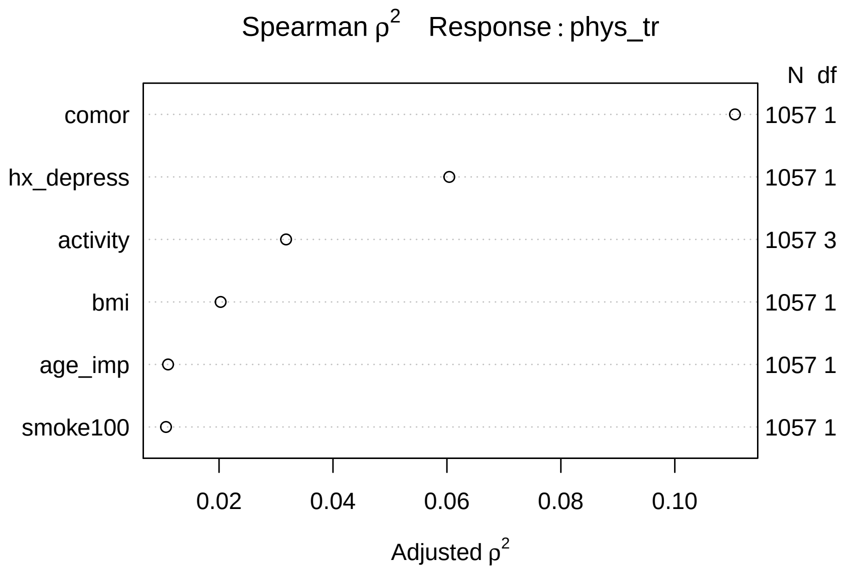
- where we are referring above to the natural (base e logarithm).

We can also specify our back-transformation to the original `physhealth` values from our new `phys_tr` as `physhealth` = exp(`phys_tr`) - 1.

14.4 Linear Regression: Considering Non-Linearity in the Predictors

Consider the following Spearman ρ^2 plot.

```
# requires rms package
# (technically Hmisc, which is loaded by rms)

smart_16_imp1 %$%
    plot(spearman2(phys_tr ~ age_imp + comor + smoke100 +
           hx_depress + bmi + activity))
```



After our single imputation, we have the same `N` value in all rows of this plot, which is what we want to see. It appears that in considering potential non-linear terms, `comor` and `hx_depress` and perhaps `activity` are worthy of increased attention. I'll make a couple of arbitrary choices, to add a raw cubic polynomial to represent the `comor` information, and we'll add an interaction term between `hx_depress` and `activity`.

14.5 “Main Effects” Linear Regression with `lm` on the Complete Cases

Recall that we have 860 complete cases in our `smart_16` data, out of a total of 1057 observations in total. A model using only the complete cases should thus drop the remaining 197 subjects. Let’s see if a main effects only model for our newly transformed `phys_tr` outcome does in fact do this.

```
m_1cc <- smart_16 %$%
    lm(phys_tr ~ age_imp + comor + smoke100 +
           hx_depress + bmi + activity)

summary(m_1cc)
```

```

Call:
lm(formula = phys_tr ~ age_imp + comor + smoke100 + hx_depress +
    bmi + activity)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.0801 -1.0389 -0.2918  1.1029  2.8478 

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   0.581959   0.370847   1.569  0.11696    
age_imp                      -0.007043   0.003813  -1.847  0.06511 .  
comor                         0.301773   0.033105   9.116  < 2e-16 ***
smoke100                      0.099038   0.090280   1.097  0.27295    
hx_depress                    0.471949   0.104232   4.528 6.81e-06 ***
bmi                           0.016375   0.006295   2.601  0.00945 ** 
activityActive               -0.229927   0.154912  -1.484  0.13812    
activityInsufficiently_Active -0.116998   0.139440  -0.839  0.40168    
activityInactive              0.256118   0.115266   2.222  0.02655 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.303 on 851 degrees of freedom
  (197 observations deleted due to missingness)
Multiple R-squared:  0.1806,    Adjusted R-squared:  0.1729 
F-statistic: 23.45 on 8 and 851 DF,  p-value: < 2.2e-16
```

Note that the appropriate number of observations are listed as “deleted due to missingness.”

14.5.1 Quality of Fit Statistics

```
glance(m_1cc) %>%
    select(r.squared, adj.r.squared, sigma, AIC, BIC) %>%
    kable(digits = c(3, 3, 2, 1, 1))
```

r.squared	adj.r.squared	sigma	AIC	BIC
0.181	0.173	1.3	2906.3	2953.8

14.5.2 Interpreting Effect Sizes

```
tidy(m_1cc, conf.int = TRUE) %>%
    select(term, estimate, std.error, conf.low, conf.high) %>%
    kable(digits = 3)
```

term	estimate	std.error	conf.low	conf.high
(Intercept)	0.582	0.371	-0.146	1.310
age_imp	-0.007	0.004	-0.015	0.000
comor	0.302	0.033	0.237	0.367
smoke100	0.099	0.090	-0.078	0.276
hx_depress	0.472	0.104	0.267	0.677
bmi	0.016	0.006	0.004	0.029
activityActive	-0.230	0.155	-0.534	0.074
activityInsufficiently_Active	-0.117	0.139	-0.391	0.157
activityInactive	0.256	0.115	0.030	0.482

We'll interpret three of the predictors here to demonstrate ideas: `comor`, `hx_depress` and `activity`.

- If we have two subjects with the same values of `age_imp`, `smoke100`, `hx_depress`, `bmi`, and `activity`, but Harry has a `comor` score that is one point higher than Sally's, then the model predicts that Harry's transformed outcome (specifically the natural logarithm of (his `physhealth` days + 1)) will be 0.302 higher than Sally's, with a 95% confidence interval around that estimate ranging from (`round(a$conf.low,3)`, `round(a$conf.high,3)`).
- If we have two subjects with the same values of `age_imp`, `comor`, `smoke100`, `bmi`, and `activity`, but Harry has a history of depression (`hx_depress` = 1) while Sally does not have such a history (so Sally's `hx_depress` = 0), then the model predicts that Harry's transformed outcome (specifically the natural logarithm of (his `physhealth` days + 1)) will be 0.472 higher than Sally's, with a 95% confidence interval around that estimate ranging from (`round(a$conf.low,3)`, `round(a$conf.high,3)`).
- The `activity` variable has four categories as indicated in the table below. The model uses the "Highly_Active" category as the reference group.

```
smart_16_imp1 %>% tabyl(activity)
```

```
              activity   n   percent
         Highly_Active 252 0.2384106
                Active 135 0.1277200
 Insufficiently_Active 193 0.1825922
              Inactive 477 0.4512772
```

- From the tidied set of coefficients, we can describe the `activity` effects as follows.
  - If Sally is “Highly Active” and Harry is “Active” but they otherwise have the same values of all predictors, then our prediction is that Harry’s transformed outcome (specifically the natural logarithm of (his `physhealth` days + 1)) will be 0.23 lower than Sally’s, with a 95% confidence interval around that estimate ranging from (`round(a$conf.low,3)`, `round(a$conf.high,3)`).
  - If instead Harry is “Insufficiently Active” but nothing else changes, then our prediction is that Harry’s transformed outcome will be 0.117 lower than Sally’s, with a 95% confidence interval around that estimate ranging from (`round(a2$conf.low,3)`, `round(a2$conf.high,3)`).
  - If instead Harry is “Inactive” but nothing else changes, then our prediction is that Harry’s transformed outcome will be -0.117 higher than Sally’s, with a 95% confidence interval around that estimate ranging from (`round(a2$conf.low,3)`, `round(a2$conf.high,3)`).

14.5.3 Making Predictions with the Model

Let’s describe two subjects, and use this model (and the ones that follow) to predict their `physhealth` values.

- Sheena is age 50, has 2 comorbidities, has smoked 100 cigarettes in her life, has no history of depression, a BMI of 25, and is Highly Active.
- Jacob is age 65, has 4 comorbidities, has never smoked, has a history of depression, a BMI of 32 and is Inactive.

We’ll first build predictions for Sheena and Jacob (with 95% prediction intervals) for `phys_tr`.

```
new2 <- tibble(
    name = c("Sheena", "Jacob"),
    age_imp = c(50, 65),
    comor = c(2, 4),
    smoke100 = c(1, 0),
    hx_depress = c(0, 1),
    bmi = c(25, 32),
    activity = c("Highly_Active", "Inactive")
)

preds_m_1cc <- predict(m_1cc, newdata = new2,
                       interval = "prediction")

preds_m_1cc
```

```
       fit      lwr      upr
1 1.341778 -1.22937 3.912925
2 2.583336  0.01399 5.152681
```

The model makes predictions for our transformed outcome, `phys_tr`. Now, we need to back-transform the predictions and the confidence intervals to build predictions for `physhealth`.

```
preds_m_1cc <- preds_m_1cc %>%
    tbl_df() %>%
    mutate(names = c("Sheena", "Jacob"),
           pred_physhealth = exp(fit) - 1,
           conf_low = exp(lwr) - 1,
           conf_high = exp(upr) - 1) %>%
    select(names, pred_physhealth, conf_low, conf_high,
           everything())

preds_m_1cc %>% kable(digits = 3)
```

names	pred_physhealth	conf_low	conf_high	fit	lwr	upr
Sheena	2.826	-0.708	49.045	1.342	-1.229	3.913
Jacob	12.241	0.014	171.894	2.583	0.014	5.153

14.6 “Augmented” Linear Regression with `lm` on the Complete Cases

Now, we'll add the non-linear terms we discussed earlier. We'll add a (raw) cubic polynomial to represent the `comor` information, and we'll add an interaction term between `hx_depress` and `activity`.

```
# requires rms package (and co-loading Hmisc)

m_2cc <- smart_16 %$%
    lm(phys_tr ~ age_imp + pol(comor, 3) + smoke100 +
           bmi + hx_depress*activity)

summary(m_2cc)
```

```
Call:
```

```
lm(formula = phys_tr ~ age_imp + pol(comor, 3) + smoke100 + bmi +
    hx_depress * activity)

Residuals:
   Min     1Q Median     3Q    Max
-2.907 -1.063 -0.267  1.143  2.924

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                               0.514823   0.376203   1.368  0.17153
age_imp                                  -0.008100   0.003865  -2.096  0.03640
pol(comor, 3)comor                        0.634274   0.160630   3.949 8.51e-05
pol(comor, 3)comor^2                     -0.130626   0.073525  -1.777  0.07599
pol(comor, 3)comor^3                      0.012508   0.008977   1.393  0.16386
smoke100                                  0.089345   0.090336   0.989  0.32294
bmi                                       0.015203   0.006315   2.408  0.01627
hx_depress                                0.647054   0.229696   2.817  0.00496
activityActive                           -0.202196   0.172300  -1.174  0.24092
activityInsufficiently_Active            -0.005815   0.166221  -0.035  0.97210
activityInactive                          0.290380   0.132198   2.197  0.02832
hx_depress:activityActive                -0.124836   0.395415  -0.316  0.75230
hx_depress:activityInsufficiently_Active -0.376355   0.310160  -1.213  0.22531
hx_depress:activityInactive              -0.172952   0.267427  -0.647  0.51798

(Intercept)
age_imp                                  *
pol(comor, 3)comor                       ***
pol(comor, 3)comor^2                     .
pol(comor, 3)comor^3
smoke100
bmi                                      *
hx_depress                               **
activityActive
activityInsufficiently_Active
activityInactive                         *
hx_depress:activityActive
hx_depress:activityInsufficiently_Active
hx_depress:activityInactive
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.301 on 846 degrees of freedom
  (197 observations deleted due to missingness)
Multiple R-squared:  0.187, Adjusted R-squared:  0.1745
F-statistic: 14.97 on 13 and 846 DF,  p-value: < 2.2e-16
```

Note again that the appropriate number of observations are listed as "deleted due to missingness."

14.6.1 Quality of Fit Statistics

```
glance(m_2cc) %>%
    select(r.squared, adj.r.squared, sigma, AIC, BIC) %>%
    kable(digits = c(3, 3, 2, 1, 1))
```

r.squared	adj.r.squared	sigma	AIC	BIC
0.187	0.175	1.3	2909.5	2980.9

14.6.2 ANOVA assessing the impact of the non-linear terms

```
anova(m_1cc, m_2cc)
```

```
Analysis of Variance Table

Model 1: phys_tr ~ age_imp + comor + smoke100 + hx_depress + bmi + activity
Model 2: phys_tr ~ age_imp + pol(comor, 3) + smoke100 + bmi + hx_depress *
    activity
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    851 1444.0
2    846 1432.8  5    11.265 1.3303  0.249
```

The difference between the models doesn't meet the standard for statistical detectabilty at our usual α levels.

14.6.3 Interpreting Effect Sizes

```
tidy(m_2cc, conf.int = TRUE) %>%
    select(term, estimate, std.error, conf.low, conf.high) %>%
    kable(digits = 3)
```

term	estimate	std.error	conf.low	conf.high
(Intercept)	0.515	0.376	-0.224	1.253
age_imp	-0.008	0.004	-0.016	-0.001
pol(comor, 3)comor	0.634	0.161	0.319	0.950
pol(comor, 3)comor^2	-0.131	0.074	-0.275	0.014
pol(comor, 3)comor^3	0.013	0.009	-0.005	0.030
smoke100	0.089	0.090	-0.088	0.267
bmi	0.015	0.006	0.003	0.028
hx_depress	0.647	0.230	0.196	1.098
activityActive	-0.202	0.172	-0.540	0.136
activityInsufficiently_Active	-0.006	0.166	-0.332	0.320
activityInactive	0.290	0.132	0.031	0.550
hx_depress:activityActive	-0.125	0.395	-0.901	0.651
hx_depress:activityInsufficiently_Active	-0.376	0.310	-0.985	0.232
hx_depress:activityInactive	-0.173	0.267	-0.698	0.352

Let's focus first on interpreting the interaction terms between `hx_depress` and `activity`.

Assume first that we have a set of subjects with the same values of `age_imp`, `smoke100`, `bmi`, and `comor`.

- Arnold has `hx_depress` = 1 and is Inactive
- Betty has `hx_depress` = 1 and is Insufficiently Active
- Carlos has `hx_depress` = 1 and is Active
- Debbie has `hx_depress` = 1 and is Highly Active
- Eamon has `hx_depress` = 0 and is Inactive
- Florence has `hx_depress` = 0 and is Insufficiently Active
- Garry has `hx_depress` = 0 and is Active
- Harry has `hx_depress` = 0 and is Highly Active

So the model, essentially can be used to compare each of the first seven people on that list to Harry (who has the reference levels of both `hx_depress` and `activity`.) Let's compare Arnold to Harry.

For instance, as compared to Harry, Arnold is expected to have a transformed outcome (specifically the natural logarithm of (his `physhealth` days + 1)) that is:

- 0.647 higher because Arnold's `hx_depress` = 1, and
- 0.29 higher still because Arnold's `activity` is "Inactive," and
- 0.173 lower because of the combination (see the 'hx_depress:activityInactive" row)

So, in total, we expect Arnold's transformed outcome to be 0.647 + 0.29 + (-0.173), or 0.764 higher than Harry's.

If we want to compare Arnold to, for instance, Betty, we first calculate Betty's difference from Harry, and then compare the two differences.

As compared to Harry, Betty is expected to have a transformed outcome (specifically the natural logarithm of (her `physhealth` days + 1)) that is:

- 0.647 higher because Betty's `hx_depress` = 1, and
- 0.006 lower still because Betty's `activity` is "Insufficiently Active," and
- 0.376 lower because of the combination (see the 'hx_depress:activityInsufficiently_Active" row)

So, in total, we expect Betty's transformed outcome to be 0.647 + (-0.006) + (-0.376), or 0.265 higher than Harry's.

And thus we can compare Betty and Arnold directly.

- Arnold is predicted to have an outcome that is 0.764 higher than Harry's.
- Betty is predicted to have an outcome that is 0.265 higher than Harry's.
- And so Arnold's predicted outcome (`phys_tr`) is 0.499 larger than Betty's.

Now, suppose we want to look at our cubic polynomial in `comor`.

- Suppose Harry and Sally have the same values for all other predictors in the model, but Harry has 1 comorbidity where Sally has none. Then the three terms in the model related to `comor` will be 1 for Harry and 0 for Sally, and the interpretation becomes pretty straightforward.
- But suppose instead that nothing has changed except Harry has 2 comorbidities and Sally has just 1. The size of the impact of this Harry - Sally difference is far larger in this situation, because the `comor` variable enters the model in a non-linear way. This is an area where fitting the model using `ols` can be helpful because of the ability to generate plots (of effects, nomograms, etc.) that can show this non-linearity in a clear way.

Suppose for instance, that Harry and Sally share the following values for the other predictors: each is age 40, has never smoked, has no history of depression, a BMI of 30 and is Highly Active.

- Now, if Harry has 1 comorbidity and Sally has none, the predicted `phys_tr` values for Harry and Sally are as indicated below.

```r
hands1 <- tibble(
    name = c("Harry", "Sally"),
    age_imp = c(40, 40),
    comor = c(1, 0),
    smoke100 = c(0, 0),
    hx_depress = c(0, 0),
    bmi = c(30, 30),
    activity = c("Highly_Active", "Highly_Active")
)

predict(m_2cc, newdata = hands1)
```

```
        1         2
```

```
1.1630840 0.6469282
```

But if Harry has 2 comorbidities and Sally 1, the predictions are:

```
hands2 <- tibble(
    name = c("Harry", "Sally"),
    age_imp = c(40, 40),
    comor = c(2, 1), # only thing that changes
    smoke100 = c(0, 0),
    hx_depress = c(0, 0),
    bmi = c(30, 30),
    activity = c("Highly_Active", "Highly_Active")
)

predict(m_2cc, newdata = hands2)
```

```
       1        2
1.493035 1.163084
```

Note that the difference in predictions between Harry and Sally is much smaller now than it was previously.

14.6.4 Making Predictions with the Model

As before, we'll use the new model to predict `physhealth` values for Sheena and Jacob.

- Sheena is age 50, has 2 comorbidities, has smoked 100 cigarettes in her life, has no history of depression, a BMI of 25, and is Highly Active.
- Jacob is age 65, has 4 comorbidities, has never smoked, has a history of depression, a BMI of 32 and is Inactive.

We'll first build predictions for Sheena and Jacob (with 95% prediction intervals) for `phys_tr`.

```
new2 <- tibble(
    name = c("Sheena", "Jacob"),
    age_imp = c(50, 65),
    comor = c(2, 4),
    smoke100 = c(1, 0),
    hx_depress = c(0, 1),
    bmi = c(25, 32),
    activity = c("Highly_Active", "Inactive")
)

preds_m_2cc <- predict(m_2cc, newdata = new2,
                       interval = "prediction")
```

```
preds_m_2cc
```

```
       fit         lwr      upr
1 1.425362 -1.14707613 3.997801
2 2.486907 -0.08635658 5.060171
```

Now, we need to back-transform the predictions and the confidence intervals that describe `phys_tr` to build predictions for `physhealth`.

```
preds_m_2cc <- preds_m_2cc %>%
    tbl_df() %>%
    mutate(names = c("Sheena", "Jacob"),
           pred_physhealth = exp(fit) - 1,
           conf_low = exp(lwr) - 1,
           conf_high = exp(upr) - 1) %>%
    select(names, pred_physhealth, conf_low, conf_high,
           everything())

preds_m_2cc %>% kable(digits = 3)
```

names	pred_physhealth	conf_low	conf_high	fit	lwr	upr
Sheena	3.159	-0.682	53.478	1.425	-1.147	3.998
Jacob	11.024	-0.083	156.617	2.487	-0.086	5.060

14.7 Using `mice` to perform Multiple Imputation

Let's focus on the main effects model, and look at the impact of performing multiple imputation to account for the missing data. Recall that in our `smart_16` data, the most "missingness" is shown in the `activity` variable, which is still missing less than 10% of the time. So we'll try a set of 10 imputations, using the default settings in the `mice` package.

```
# requires library(mice)

set.seed(432)

# create small data set including only variables to
# be used in building the imputation model

sm16 <- smart_16 %>%
    select(physhealth, phys_tr, activity, age_imp, bmi, comor,
           hx_depress, smoke100)

smart_16_mice10 <- mice(sm16, m = 10)
```

```
 iter imp variable
  1   1  activity  age_imp  bmi  comor  hx_depress  smoke100
  1   2  activity  age_imp  bmi  comor  hx_depress  smoke100
  1   3  activity  age_imp  bmi  comor  hx_depress  smoke100
  1   4  activity  age_imp  bmi  comor  hx_depress  smoke100
  1   5  activity  age_imp  bmi  comor  hx_depress  smoke100
  1   6  activity  age_imp  bmi  comor  hx_depress  smoke100
  1   7  activity  age_imp  bmi  comor  hx_depress  smoke100
  1   8  activity  age_imp  bmi  comor  hx_depress  smoke100
  1   9  activity  age_imp  bmi  comor  hx_depress  smoke100
  1   10  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   1  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   2  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   3  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   4  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   5  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   6  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   7  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   8  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   9  activity  age_imp  bmi  comor  hx_depress  smoke100
  2   10  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   1  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   2  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   3  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   4  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   5  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   6  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   7  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   8  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   9  activity  age_imp  bmi  comor  hx_depress  smoke100
  3   10  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   1  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   2  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   3  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   4  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   5  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   6  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   7  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   8  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   9  activity  age_imp  bmi  comor  hx_depress  smoke100
  4   10  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   1  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   2  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   3  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   4  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   5  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   6  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   7  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   8  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   9  activity  age_imp  bmi  comor  hx_depress  smoke100
  5   10  activity  age_imp  bmi  comor  hx_depress  smoke100
```

```
summary(smart_16_mice10)
```

```
Class: mids
Number of multiple imputations:  10
Imputation methods:
physhealth    phys_tr   activity    age_imp        bmi      comor hx_depress
        ""         ""  "polyreg"      "pmm"      "pmm"      "pmm"      "pmm"
  smoke100
     "pmm"
PredictorMatrix:
           physhealth phys_tr activity age_imp bmi comor hx_depress smoke100
physhealth          0       1        1       1   1     1          1        1
phys_tr             1       0        1       1   1     1          1        1
activity            1       1        0       1   1     1          1        1
age_imp             1       1        1       0   1     1          1        1
bmi                 1       1        1       1   0     1          1        1
comor               1       1        1       1   1     0          1        1
```

14.8 Running the Linear Regression in `lm` with Multiple Imputation

Next, we'll run the linear model (main effects) on each of the 10 imputed data sets.

```
m10_mods <-
    with(smart_16_mice10, lm(phys_tr ~ age_imp + comor +
                                 smoke100 + hx_depress +
                                 bmi + activity))

summary(m10_mods)
```

```
# A tibble: 90 x 6
   term                       estimate std.error statistic  p.value  nobs
   <chr>                         <dbl>     <dbl>     <dbl>    <dbl> <int>
 1 (Intercept)                 0.317     0.326       0.971 3.32e- 1  1057
 2 age_imp                    -0.00489   0.00334    -1.47  1.43e- 1  1057
 3 comor                       0.313     0.0295     10.6   4.72e-25  1057
 4 smoke100                    0.135     0.0799      1.69  9.22e- 2  1057
 5 hx_depress                  0.500     0.0929      5.38  9.14e- 8  1057
 6 bmi                              0.0187    0.00564     3.31  9.64e- 4  1057
 7 activityActive                  -0.202     0.138      -1.46  1.44e- 1  1057
 8 activityInsufficiently_Active -0.0695    0.124      -0.561 5.75e- 1  1057
 9 activityInactive                 0.262     0.103       2.54  1.11e- 2  1057
10 (Intercept)                      0.363     0.332       1.10  2.74e- 1  1057
# ... with 80 more rows
```

Then, we'll pool results across the 10 imputations

```
m10_pool <- pool(m10_mods)
summary(m10_pool, conf.int = TRUE) %>%
    select(-statistic, -df) %>%
    kable(digits = 3)
```

term	estimate	std.error	p.value	2.5 %	97.5 %
(Intercept)	0.444	0.342	0.194	-0.227	1.114
age_imp	-0.005	0.003	0.128	-0.012	0.002
comor	0.309	0.031	0.000	0.249	0.369
smoke100	0.114	0.083	0.171	-0.049	0.278
hx_depress	0.512	0.094	0.000	0.327	0.696
bmi	0.016	0.006	0.009	0.004	0.027
activityActive	-0.204	0.140	0.146	-0.479	0.071
activityInsufficiently_Active	-0.044	0.129	0.735	-0.298	0.210
activityInactive	0.260	0.106	0.014	0.052	0.469

And we can compare these results to the complete case analysis we completed earlier.

```
tidy(m_1cc, conf.int = TRUE) %>%
    select(term, estimate, std.error, p.value, conf.low, conf.high) %>%
    kable(digits = 3)
```

term	estimate	std.error	p.value	conf.low	conf.high
(Intercept)	0.582	0.371	0.117	-0.146	1.310
age_imp	-0.007	0.004	0.065	-0.015	0.000
comor	0.302	0.033	0.000	0.237	0.367
smoke100	0.099	0.090	0.273	-0.078	0.276
hx_depress	0.472	0.104	0.000	0.267	0.677
bmi	0.016	0.006	0.009	0.004	0.029
activityActive	-0.230	0.155	0.138	-0.534	0.074
activityInsufficiently_Active	-0.117	0.139	0.402	-0.391	0.157
activityInactive	0.256	0.115	0.027	0.030	0.482

Note that there are some sizeable differences here, although nothing enormous.

If we want the pooled R^2 or pooled adjusted R^2 after imputation, R will provide it (and a 95% confidence interval around the estimate) with …

```r
pool.r.squared(m10_mods)
```

```
          est     lo 95     hi 95 fmi
R^2 0.1912561 0.1482819 0.2369623 NaN
```

```r
pool.r.squared(m10_mods, adjusted = TRUE)
```

```
              est     lo 95     hi 95 fmi
adj R^2 0.1850807 0.1425132 0.2305277 NaN
```

We can see the fraction of missing information about each coefficient due to non-response (`fmi`) and other details with the following code...

```r
m10_pool
```

```
Class: mipo    m = 10
                           term  m    estimate         ubar            b
1                   (Intercept) 10  0.44377168 1.078194e-01 8.002900e-03
2                       age_imp 10 -0.00522810 1.123668e-05 4.824290e-07
3                         comor 10  0.30871888 8.801039e-04 5.466082e-05
4                      smoke100 10  0.11415718 6.474388e-03 4.130673e-04
5                    hx_depress 10  0.51155722 8.669413e-03 1.582684e-04
6                           bmi 10  0.01576150 3.182381e-05 3.425191e-06
7                activityActive 10 -0.20412627 1.914250e-02 4.851040e-04
8 activityInsufficiently_Active 10 -0.04383739 1.565925e-02 9.855183e-04
9              activityInactive 10  0.26046070 1.069627e-02 5.023887e-04
             t dfcom       df        riv     lambda        fmi
1 1.166226e-01  1048 599.8172 0.08164758 0.07548446 0.07855177
2 1.176735e-05  1048 814.9042 0.04722675 0.04509697 0.04743197
3 9.402308e-04  1048 677.6361 0.06831796 0.06394909 0.06669961
4 6.928762e-03  1048 666.2487 0.07018023 0.06557795 0.06837041
5 8.843508e-03  1048 982.0512 0.02008154 0.01968621 0.02167660
6 3.559152e-05  1048 432.0874 0.11839279 0.10585976 0.10996992
7 1.967611e-02  1048 939.5064 0.02787591 0.02711992 0.02918437
8 1.674332e-02  1048 672.0473 0.06922874 0.06474643 0.06751736
9 1.124890e-02  1048 785.1905 0.05166545 0.04912727 0.05154007
```

14.9 Fit the Multiple Imputation Model with `aregImpute`

Here, we'll use `aregImpute` to deal with missing values through multiple imputation, and use the `ols` function in the `rms` package to fit the model.

The first step is to fit the multiple imputation model. We'll use `n.impute` = 10 imputations, with `B` = 10 bootstrap samples for the preditive mean matching, and fit both linear models and models with restricted cubic splines with 3 knots (`nk = c(0, 3)`) allowing the target variable to have a non-linear transformation when `nk` is 3, via `tlinear = FALSE`.

```
set.seed(43201602)
dd <- datadist(smart_16)
options(datadist = "dd")

fit16_imp <-
    aregImpute(~ phys_tr + age_imp + comor + smoke100 +
                   hx_depress + bmi + activity,
               nk = c(0, 3), tlinear = FALSE,
               data = smart_16, B = 10, n.impute = 10)
```

```
Iteration 1 Iteration 2 Iteration 3 Iteration 4 Iteration 5 Iteration 6 Iteration 7 Iteration 8 I
```

Here are the results of that imputation model.

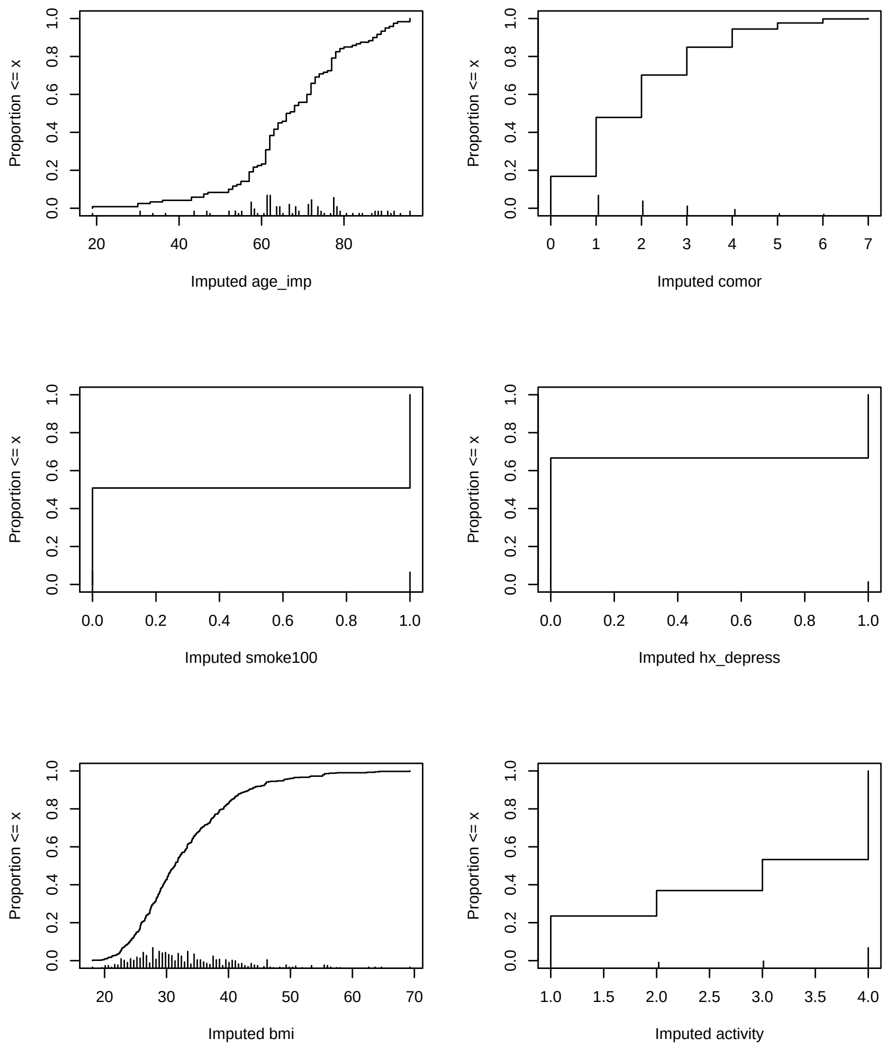
```
fit16_imp
```

```
Multiple Imputation using Bootstrap and PMM

aregImpute(formula = ~phys_tr + age_imp + comor + smoke100 +
    hx_depress + bmi + activity, data = smart_16, n.impute = 10,
    nk = c(0, 3), tlinear = FALSE, B = 10)

n: 1057     p: 7    Imputations: 10     nk: 0

Number of NAs:
   phys_tr    age_imp      comor   smoke100 hx_depress        bmi   activity
         0         12         47         24          3         84         85

           type d.f.
phys_tr       s    1
age_imp       s    1
comor         s    1
smoke100      l    1
hx_depress    l    1
bmi           s    1
activity      c    3

R-squares for Predicting Non-Missing Values for Each Variable
Using Last Imputations of Predictors
   age_imp      comor   smoke100 hx_depress        bmi   activity
     0.224      0.206      0.059      0.167      0.169      0.057

Resampling results for determining the complexity of imputation models

Variable being imputed: age_imp
```

```
                                           nk=0   nk=3
Bootstrap bias-corrected R^2              0.186  0.215
10-fold cross-validated  R^2              0.211  0.215
Bootstrap bias-corrected mean   |error|  9.108 10.894
10-fold cross-validated  mean   |error| 65.169 10.919
Bootstrap bias-corrected median |error|  7.290  8.784
10-fold cross-validated  median |error| 66.006  8.613

Variable being imputed: comor
                                          nk=0  nk=3
Bootstrap bias-corrected R^2             0.183 0.182
10-fold cross-validated  R^2             0.184 0.193
Bootstrap bias-corrected mean   |error| 0.987 1.184
10-fold cross-validated  mean   |error| 1.759 1.171
Bootstrap bias-corrected median |error| 0.828 0.910
10-fold cross-validated  median |error| 1.574 0.892

Variable being imputed: smoke100
                                           nk=0   nk=3
Bootstrap bias-corrected R^2             0.0224 0.0187
10-fold cross-validated  R^2             0.0358 0.0217
Bootstrap bias-corrected mean   |error| 0.4853 0.4866
10-fold cross-validated  mean   |error| 0.9462 0.9561
Bootstrap bias-corrected median |error| 0.4788 0.4772
10-fold cross-validated  median |error| 0.8479 0.8706

Variable being imputed: hx_depress
                                          nk=0  nk=3
Bootstrap bias-corrected R^2             0.157 0.138
10-fold cross-validated  R^2             0.147 0.148
Bootstrap bias-corrected mean   |error| 0.355 0.360
10-fold cross-validated  mean   |error| 0.801 0.783
Bootstrap bias-corrected median |error| 0.333 0.337
10-fold cross-validated  median |error| 0.711 0.673

Variable being imputed: bmi
                                           nk=0  nk=3
Bootstrap bias-corrected R^2              0.125 0.122
10-fold cross-validated  R^2              0.134 0.133
Bootstrap bias-corrected mean   |error|  5.221 6.822
10-fold cross-validated  mean   |error| 32.458 6.884
Bootstrap bias-corrected median |error|  4.178 5.782
10-fold cross-validated  median |error| 31.330 5.932

Variable being imputed: activity
                                           nk=0   nk=3
```

```
Bootstrap bias-corrected R^2             0.0312 0.0275
10-fold cross-validated  R^2             0.0450 0.0402
Bootstrap bias-corrected mean   |error| 1.8774 1.8884
10-fold cross-validated  mean   |error| 1.1121 1.1043
Bootstrap bias-corrected median |error| 2.0000 2.0000
10-fold cross-validated  median |error| 1.0000 1.0000
```

```r
par(mfrow = c(3,2))
plot(fit16_imp)
```



```
par(mfrow = c(1,1))
```

The plot helps us see where the imputations are happening.

14.10 Fit Linear Regression using ols and fit.mult.impute

```
m16_imp <-
    fit.mult.impute(phys_tr ~ age_imp + comor + smoke100 +
                        hx_depress + bmi + activity,
                    fitter = ols, xtrans = fit16_imp,
                    data = smart_16, x = TRUE, y = TRUE)
```

```
Variance Inflation Factors Due to Imputation:

                     Intercept                         age_imp
                          1.03                            1.01
                         comor                        smoke100
                          1.03                            1.06
                    hx_depress                             bmi
                          1.02                            1.06
               activity=Active activity=Insufficiently_Active
                          1.19                            1.14
             activity=Inactive
                          1.23

Rate of Missing Information:

                     Intercept                         age_imp
                          0.03                            0.01
                         comor                        smoke100
                          0.03                            0.06
                    hx_depress                             bmi
                          0.02                            0.06
               activity=Active activity=Insufficiently_Active
                          0.16                            0.13
             activity=Inactive
                          0.19

d.f. for t-distribution for Tests of Single Coefficients:

                     Intercept                         age_imp
                       8176.67                        45410.80
                         comor                          smoke100
                      13030.27                           2670.64
                    hx_depress                               bmi
                      28199.30                           2935.89
               activity=Active activity=Insufficiently_Active
                        354.62                            571.56
             activity=Inactive
                        258.42

The following fit components were averaged over the 10 model fits:

  fitted.values stats linear.predictors
```

14.10.1 Summaries and Coefficients

Here are the results:

```
m16_imp
```

```
Linear Regression Model

 fit.mult.impute(formula = phys_tr ~ age_imp + comor + smoke100 +
     hx_depress + bmi + activity, fitter = ols, xtrans = fit16_imp,
     data = smart_16, x = TRUE, y = TRUE)

                 Model Likelihood    Discrimination
                       Ratio Test           Indexes
 Obs    1057    LR chi2    219.94    R2       0.188
 sigma1.2881    d.f.            8    R2 adj   0.182
 d.f.   1048    Pr(> chi2) 0.0000    g        0.687

 Residuals

     Min      1Q  Median      3Q     Max
 -3.0621 -1.0327 -0.2878  1.1104  2.8018


                                Coef    S.E.   t     Pr(>|t|)
 Intercept                       0.4052 0.3352  1.21 0.2271
 age_imp                        -0.0049 0.0034 -1.46 0.1437
 comor                           0.3078 0.0302 10.20 <0.0001
 smoke100                        0.1259 0.0830  1.52 0.1296
 hx_depress                      0.5120 0.0940  5.45 <0.0001
 bmi                             0.0164 0.0058  2.83 0.0048
 activity=Active                -0.1773 0.1513 -1.17 0.2416
 activity=Insufficiently_Active -0.0396 0.1342 -0.30 0.7680
 activity=Inactive              0.2401 0.1144  2.10 0.0360
```

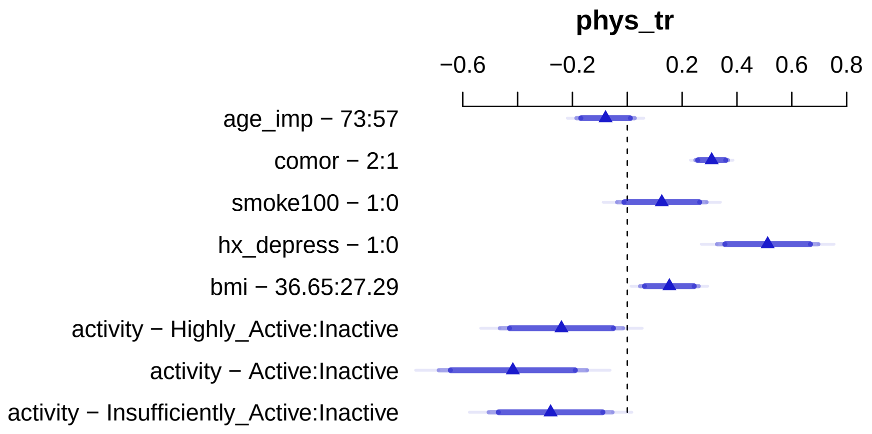
14.10.2 Effect Sizes

We can plot and summarize the effect sizes using the usual `ols` tools:

```
summary(m16_imp)
```

```
             Effects              Response : phys_tr 

 Factor                                    Low   High  Diff. Effect    S.E.    
 age_imp                                   57.00 73.00 16.00 -0.079163 0.054107
 comor                                      1.00  2.00  1.00  0.307790 0.030171
 smoke100                                   0.00  1.00  1.00  0.125890 0.083000
 hx_depress                                 0.00  1.00  1.00  0.511980 0.094007
 bmi                                       27.29 36.65  9.36  0.153530 0.054322
 activity - Highly_Active:Inactive          4.00  1.00    NA -0.240070 0.114350
 activity - Active:Inactive                 4.00  2.00    NA -0.417320 0.137640
 activity - Insufficiently_Active:Inactive  4.00  3.00    NA -0.279650 0.115000
 Lower 0.95 Upper 0.95
 -0.185330   0.027008 
  0.248590   0.366990 
 -0.036973   0.288760 
  0.327520   0.696450 
  0.046932   0.260120 
 -0.464450  -0.015686 
 -0.687400  -0.147250 
 -0.505310  -0.054002 
```

```
plot(summary(m16_imp))
```



14.10.3 Making Predictions with this Model

Once again, let's make predictions for our two subjects, and use this model (and the ones that follow) to predict their `physhealth` values.

- Sheena is age 50, has 2 comorbidities, has smoked 100 cigarettes in her life, has no history of depression, a BMI of 25, and is Highly Active.
- Jacob is age 65, has 4 comorbidities, has never smoked, has a history of depression, a BMI of 32 and is Inactive.

```
new2 <- tibble(
    name = c("Sheena", "Jacob"),
    age_imp = c(50, 65),
    comor = c(2, 4),
    smoke100 = c(1, 0),
    hx_depress = c(0, 1),
    bmi = c(25, 32),
    activity = c("Highly_Active", "Inactive")
)

preds_m_16imp <- predict(m16_imp,
                         newdata = data.frame(new2))

preds_m_16imp
```

```
       1        2
1.309306 2.591649
```

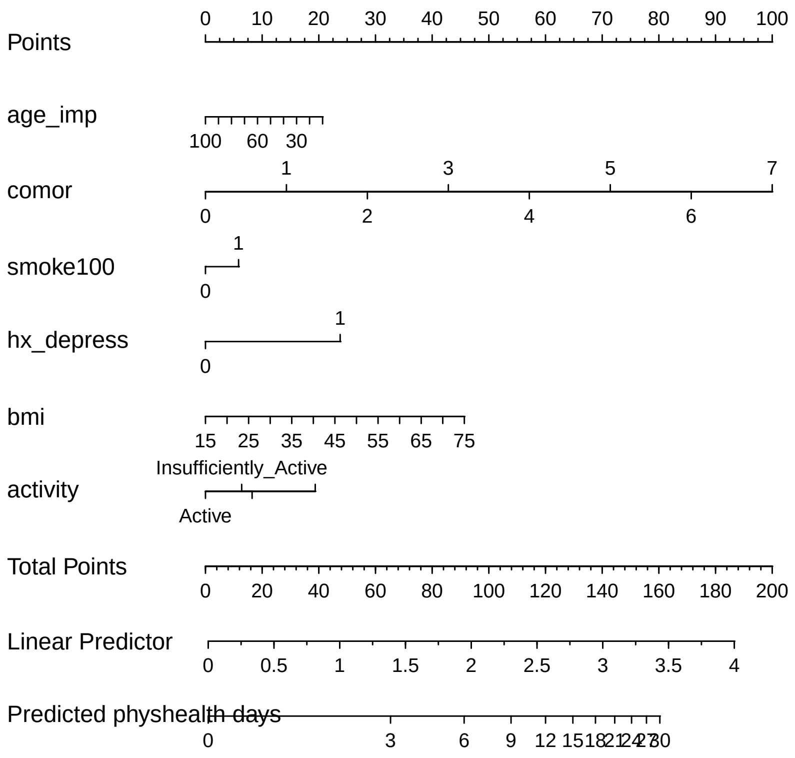
```
preds_m_16imp <- preds_m_16imp %>%
    tbl_df() %>%
    mutate(names = c("Sheena", "Jacob"),
           pred_physhealth = exp(value) - 1) %>%
    select(names, pred_physhealth)

preds_m_16imp %>% kable(digits = 3)
```

names	pred_physhealth
Sheena	2.704
Jacob	12.352

14.10.4 Nomogram

We can also develop a nomogram, if we like. As a special touch, we'll add a prediction at the bottom which back-transforms out of the predicted `phys_tr` back to the `physhealth` days.

```
plot(nomogram(m16_imp,
              fun = list(function(x) exp(x) - 1),
              funlabel = "Predicted physhealth days",
              fun.at = seq(0, 30, 3)))
```



We can see the big role of `comor` and `hx_depress` in this model.

14.10.5 Validating Summary Statistics

We can cross-validate summary measures, like R^2…

```
validate(m16_imp)
```

```
          index.orig training   test optimism index.corrected  n
R-square      0.1867   0.1984 0.1793   0.0192          0.1676 40
MSE           1.6472   1.6168 1.6623  -0.0455          1.6927 40
g             0.6876   0.7031 0.6749   0.0282          0.6594 40
Intercept     0.0000   0.0000 0.0677  -0.0677          0.0677 40
Slope         1.0000   1.0000 0.9636   0.0364          0.9636 40
```

Chapter 15

Using tidymodels to fit linear regressions

Chapter to come.

Chapter 16

Using tidymodels to fit logistic regressions

Chapter to come.

Chapter 17

Using tidymodels approaches: Next Steps

Chapter to come.

Barnett, Peggy A., Simon Roman-Golstein, Fred Ramsey, and others. 1995. "Differential Permeability and Quantitative MR Imaging of a Human Lung Carcinoma Brain Xenograft in the Nude Rat." *American Journal of Pathology* 146(2): 436–49. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1869863/.

Berkhemer, Olvert A., Puck S. S. Fransen, Debbie Buemer, and others. 2015. "A Randomized Trial of Intraarterial Treatment for Acute Ischemic Stroke." *New England Journal of Medicine* 372: 11–20. http://www.nejm.org/doi/full/10.1056/NEJMoa1411587.

Faraway, Julian J. 2006. *Extending the Linear Model with r.* Boca Raton, FL: CRC Press.

Gelman, Andrew, and Jennifer Hill. 2007. *Data Analysis Using Regression and Multilevel/Hierarchical Models.* New York: Cambridge University Press.

Harrell, Frank E. 2001. *Regression Modeling Strategies.* New York: Springer.

———. 2018. *Regression Modeling Strategies Course Notes.* http://www.fharrell.com/#links.

Kim, Hae-Young. 2014. "Statistical Notes for Clinical Researchers: Two-Way Analysis of Variance (ANOVA) - Exploring Possible Interaction Between Factors." *Restorative Dentistry & Endodontics* 39(2): 143–47. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3978106/.

Long, J. Scott. 1997. *Regression Models for Categorical and Limited Dependent Variables.* Thousand Oaks, CA: Sage Publications.

McDonald, Gary C., and Richard C. Schwing. 1973. "Instabilities of Regression Estimates Relating Air Pollution to Mortality." *Technometrics* 15 (3): 463–81.

Peduzzi, Peter, John Concato, Elizabeth Kemper, Theodore R. Holford, and Alvan R. Feinstein. 1996. "A Simulation Study of the Number of Events Per Variable in Logistic Regression Analysis." *Journal of Clinical Epidemiology* 49 (12): 1373–79.

Ramsey, Fred L., and Daniel W. Schafer. 2002. *The Statistical Sleuth: A Course in Methods of Data Analysis.* Second Edition. Pacific Grove, CA: Duxbury.

Riffenburgh, Robert H. 2006. *Statistics in Medicine.* Second Edition. Burlington, MA: Elsevier Academic Press.

Rosenbaum, Paul R. 2017. *Observation and Experiment: An Introduction to Causal Inference.* Cambridge, MA: Harvard University Press.

Roy, Denis, Mario Talajic, Stanley Nattel, and others. 2008. "Rhythm Control Versus Rate Control for Atrial Fibrillation and Heart Failure." *New England Journal of Medicine* 358: 2667–77. http://www.nejm.org/doi/full/10.1056/NEJMoa0708789.

Stamey, J. N. Kabalin, T. A., and others. 1989. "Prostate Specific Antigen in the Diagnosis and Treatment of Adenocarcinoma of the Prostate: II. Radical Prostatectomy Treated Patients." *Journal of Urology* 141(5): 1076–83. https://www.ncbi.nlm.nih.gov/pubmed/2468795.

Tolaney, Sara M, William T. Barry, T. Dang Chau, and others. 2015. "Adjuvant Paclitaxel and Trastuzumab for Node-Negative, Her2-Positive Breast Cancer." *New England Journal of Medicine* 372: 134–41. http://www.nejm.org/doi/full/10.1056/NEJMoa1406281.

Vittinghoff, Eric, David V. Glidden, Stephen C. Shiboski, and Charles E. McCulloch. 2012. *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models.* Second Edition. Springer-Verlag, Inc. http://www.biostat.ucsf.edu/vgsm/.