

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

Thomas E. Love, Ph.D.

Built 2018-04-02 00:09:10

Contents

Introduction	9
R Packages used in these notes	11
Data used in these notes	13
Special Functions used in these notes	15
1 Building Table 1	17
1.1 Two examples from the <i>New England Journal of Medicine</i>	17
1.2 The MR CLEAN trial	18
1.3 Simulated <code>fakestroke</code> data	20
1.4 Building Table 1 for <code>fakestroke</code> : Attempt 1	21
1.5 <code>fakestroke</code> Table 1: Attempt 2	23
1.6 Obtaining a more detailed Summary	25
1.7 Exporting the Completed Table 1 from R to Excel or Word	28
1.8 A Controlled Biological Experiment - The Blood-Brain Barrier	30
1.9 The <code>bloodbrain.csv</code> file	30
1.10 A Table 1 for <code>bloodbrain</code>	31
2 Linear Regression on a small SMART data set	37
2.1 BRFSS and SMART	37
2.2 The <code>smartcle1</code> data: Cookbook	37
2.3 <code>smartcle2</code> : Omitting Missing Observations: Complete-Case Analyses	38
2.4 Summarizing the <code>smartcle2</code> data numerically	40
2.5 Counting as exploratory data analysis	41
2.6 First Modeling Attempt: Can <code>bmi</code> predict <code>physhealth</code> ?	46
2.7 A New Small Study: Predicting BMI	55
2.8 <code>c2_m1</code> : A simple t-test model	57
2.9 <code>c2_m2</code> : Adding another predictor (two-way ANOVA without interaction)	58
2.10 <code>c2_m3</code> : Adding the interaction term (Two-way ANOVA with interaction)	62
2.11 <code>c2_m4</code> : Using <code>female</code> and <code>sleephrs</code> in a model for <code>bmi</code>	64
2.12 Making Predictions with a Linear Regression Model	66
2.13 Centering the model	68
2.14 Rescaling an input by subtracting the mean and dividing by 2 standard deviations	70
2.15 <code>c2_m5</code> : What if we add more variables?	72
2.16 <code>c2_m6</code> : Would adding self-reported health help?	74
2.17 <code>c2_m7</code> : What if we added the <code>menthealth</code> variable?	75
2.18 Key Regression Assumptions for Building Effective Prediction Models	76
3 Analysis of Variance	79
3.1 The <code>bonding</code> data: A Designed Dental Experiment	79
3.2 A One-Factor Analysis of Variance	79

3.3	A Two-Way ANOVA: Looking at Two Factors	83
3.4	A Means Plot (with standard deviations) to check for interaction	84
3.5	Fitting the Two-Way ANOVA model with Interaction	86
3.6	Comparing Individual Combinations of <code>resin</code> and <code>light</code>	88
3.7	The bonding model without Interaction	89
3.8	<code>cortisol</code> : A Hypothetical Clinical Trial	91
3.9	Creating a factor combining <code>sex</code> and <code>waist</code>	92
3.10	A Means Plot for the <code>cortisol</code> trial (with standard errors)	93
3.11	A Two-Way ANOVA model for <code>cortisol</code> with Interaction	94
3.12	A Two-Way ANOVA model for <code>cortisol</code> without Interaction	95
4	Analysis of Covariance	99
4.1	An Emphysema Study	99
4.2	Does <code>sex</code> affect the mean change in theophylline?	100
4.3	Is there an association between <code>age</code> and <code>sex</code> in this study?	101
4.4	Adding a quantitative covariate, <code>age</code> , to the model	101
4.5	Rerunning the ANCOVA model after simple imputation	102
4.6	Looking at a factor-covariate interaction	103
4.7	Centering the Covariate to Facilitate ANCOVA Interpretation	104
5	Missing Data Mechanisms and Single Imputation	107
5.1	A Toy Example	107
5.2	Missing-data mechanisms	109
5.3	Options for Dealing with Missingness	109
5.4	Complete Case (and Available Case) analyses	109
5.5	Single Imputation	110
5.6	Multiple Imputation	110
5.7	Building a Complete Case Analysis	110
5.8	Single Imputation with the Mean or Mode	110
5.9	Doing Single Imputation with <code>simputation</code>	111
6	A Study of Prostate Cancer	115
6.1	Data Load and Background	115
6.2	Code Book	115
6.3	Additions for Later Use	116
6.4	Fitting and Evaluating a Two-Predictor Model	117
6.5	Exploring Model <code>c5_prost_A</code>	118
6.6	Plotting Model <code>c5_prost_A</code>	122
6.7	Cross-Validation of Model <code>c5_prost_A</code>	125
7	Stepwise Variable Selection	131
7.1	Strategy for Model Selection	131
7.2	A “Kitchen Sink” Model (Model <code>c5_prost_ks</code>)	132
7.3	Sequential Variable Selection: Stepwise Approaches	132
7.4	Forward Selection with the <code>step</code> function	133
7.5	Backward Elimination using the <code>step</code> function	134
7.6	Allen-Cady Modified Backward Elimination	136
7.7	Summarizing the Results	137
8	“Best Subsets” Variable Selection in our Prostate Cancer Study	141
8.1	Four Key Summaries We’ll Use to Evaluate Potential Models	141
8.2	Using <code>regsubsets</code> in the <code>leaps</code> package	141
8.3	Calculating bias-corrected AIC	143
8.4	Plotting the Best Subsets Results using <code>ggplot2</code>	146
8.5	Table of Key Results	151

8.6 Models Worth Considering?	152
8.7 Compare these candidate models in-sample?	152
8.8 AIC and BIC comparisons, within the training sample	153
8.9 Cross-Validation of Candidate Models out of Sample	154
8.10 What about Interaction Terms?	156
9 Adding Non-linear Terms to a Linear Regression Model	157
9.1 The pollution data	157
9.2 Fitting a straight line model to predict y from x2	158
9.3 Quadratic polynomial model to predict y using x2	159
9.4 Orthogonal Polynomials	164
9.5 Fit a cubic polynomial to predict y from x3	167
9.6 Fitting a restricted cubic spline in a linear regression	170
9.7 “Spending” Degrees of Freedom	174
9.8 Spending DF on Non-Linearity: The Spearman ρ^2 Plot	176
10 Using ols from the rms package to fit linear models	179
10.1 Fitting a model with ols	179
10.2 ANOVA for an ols model	181
10.3 Effect Estimates	181
10.4 The Predict function for an ols model	183
10.5 Checking Influence via dfbeta	184
10.6 Model Validation and Correcting for Optimism	187
10.7 Building a Nomogram for Our Model	188
11 Other Variable Selection Strategies	191
11.1 Why not use stepwise procedures?	191
11.2 Ridge Regression	192
11.3 The Lasso	195
11.4 Applying the Lasso to the pollution data	203
12 Logistic Regression: The Foundations	211
12.1 A First Attempt: A Linear Probability Model	211
12.2 Logistic Regression	213
12.3 The Logistic Regression Model	214
12.4 The Link Function	214
12.5 The logit or log odds	215
12.6 Interpreting the Coefficients of a Logistic Regression Model	215
12.7 The Logistic Regression has non-constant variance	216
12.8 Fitting a Logistic Regression Model to our Simulated Data	216
12.9 Plotting the Logistic Regression Model	218
13 Logistic Regression and the resect data	221
13.1 The resect data	221
13.2 Running A Simple Logistic Regression Model	222
13.3 Logistic Regression using glm	222
13.4 Interpreting the Model Summary	226
13.5 Plotting a Simple Logistic Regression Model	228
13.6 How well does Model A classify subjects?	231
13.7 Receiver Operating Characteristic Curve Analysis	232
13.8 The ROC Plot for res_modA	238
13.9 Assessing Residual Plots from Model A	240
13.10 Model B: A “Kitchen Sink” Logistic Regression Model	241
13.11 Plotting Model B	243
13.12 Logistic Regression using lrm	246

13.13Model D: An Augmented Kitchen Sink Model	254
13.14Model E: Fitting a Reduced Model in light of Model D	262
13.15Concordance: Comparing Model C, D and E's predictions	268
13.16Conclusions	271
14 Logistic Regression and the <code>smartcle1</code> data	273
14.1 The <code>smartcle1</code> data	273
14.2 Thinking About Non-Linear Terms	274
14.3 A First Model for <code>exerany</code> (Complete Case Analysis)	275
14.4 Building a Larger Model: Spearman ρ^2 Plot	276
14.5 A Second Model for <code>exerany</code> (Complete Cases)	277
14.6 Dealing with Missing Data via Simple Imputation	279
14.7 Refitting Model 1 with simply imputed data	286
14.8 Refitting Model 2 with simply imputed data	294
14.9 Comparing Model 2 to Model 1 after simple imputation	302
14.10Dealing with Missing Data via Multiple Imputation	303
14.11Combining the Imputation and Outcome Models	307
14.12Models with and without Imputation	320
15 Linear Regression and the <code>smartcle1</code> data	323
15.1 The <code>smartcle1</code> data	323
15.2 Thinking About Non-Linear Terms	324
15.3 A First Model for <code>sleephrs</code> (Complete Case Analysis)	325
15.4 Building a Larger Model: Spearman ρ^2 Plot	326
15.5 A Second Model for <code>sleephrs</code> (Complete Cases)	327
15.6 Dealing with Missing Data via Simple Imputation	329
15.7 Refitting Model A with simply imputed data	335
15.8 Refitting Model B with simply imputed data	345
15.9 Comparing Model B.2 to Model A.2 after simple imputation	351
15.10Dealing with Missing Data via Multiple Imputation	352
15.11Combining the Imputation and Outcome Models	356
16 Colorectal Cancer Screening and Some Special Cases	363
16.1 Logistic Regression for Aggregated Data	363
16.2 Probit Regression	366
17 Cleaning the BRFSS SMART Data	371
17.1 Key resources	371
17.2 Ingesting The Raw Data	372
17.3 The National Data	372
17.4 Cleaning the BRFSS Data	372
17.5 Creating Some Quantitative Variables from Thin Air	401
17.6 Clean Data in the State of Ohio	402
17.7 Clean Cleveland-Elyria Data	405
18 Modeling a Count Outcome in Ohio SMART	409
18.1 Preliminaries	409
18.2 A subset of the Ohio SMART data	409
18.3 Exploratory Data Analysis (in the 18-49 group)	412
18.4 Modeling Strategies Explored Here	417
18.5 The OLS Approach	418
18.6 OLS model on $\log(\text{physhealth} + 1)$ days	422
18.7 A Poisson Regression Model	426
18.8 Overdispersion in a Poisson Model	434
18.9 Fitting the Quasi-Poisson Model	435

18.10 Poisson and Quasi-Poisson models using <code>Glm</code> from the <code>rms</code> package	439
18.11 Negative Binomial Model	443
18.12 The Problem: Too Few Zeros	452
18.13 The Zero-Inflated Poisson Regression Model	452
18.14 The Zero-Inflated Negative Binomial Regression Model	458
18.15 A “hurdle” model (with Poisson)	465
18.16 A “hurdle” model (with negative binomial for overdispersion)	472
18.17 A Tobit (Censored) Regression Model	480
19 Modeling an Ordinal Multi-Categorical Outcome in Ohio SMART	487
19.1 Where to Read this Chapter	487
20 Analyzing Literary Styles with Multinomial Logistic Regression	489
20.1 The Authorship Example	489
20.2 Focus on 11 key words	490
20.3 A Multinomial Logistic Regression Model	492
20.4 Model 2	493
20.5 Classification Table	495
20.6 Probability Curves based on a Single Predictor	496

Introduction

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

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

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

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

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

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

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

R Packages used in these notes

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

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


Data used in these notes

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

```
fakestroke <- read.csv("data/fakestroke.csv") %>%tbl_df  
bloodbrain <- read.csv("data/bloodbrain.csv") %>%tbl_df  
smartcle1 <- read.csv("data/smartcle1.csv") %>%tbl_df  
bonding <- read.csv("data/bonding.csv") %>%tbl_df  
cortisol <- read.csv("data/cortisol.csv") %>%tbl_df  
emphysema <- read.csv("data/emphysema.csv") %>%tbl_df  
prost <- read.csv("data/prost.csv") %>%tbl_df  
pollution <- read.csv("data/pollution.csv") %>%tbl_df  
resect <- read.csv("data/resect.csv") %>%tbl_df  
colscr <- read.csv("data/screening.csv") %>%tbl_df  
colscr2 <- read.csv("data/screening2.csv") %>%tbl_df  
authorship <- read_csv("data/authorship.csv")
```

```
Parsed with column specification:  
cols(  
  .default = col_integer(),  
  Author = col_character()  
)
```

See `spec(...)` for full column specifications.

Special Functions used in these notes

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


Chapter 1

Building Table 1

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

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

1.1.1 A simple Table 1

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

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

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

HER2-positive¹ invasive breast cancer that began the protocol therapy. Age, sex and race (along with severity of illness) are the most commonly identified characteristics in a Table 1.

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

1.1.2 A group comparison

A more typical Table 1 involves a group comparison, for example in this excerpt from Roy et al. (2008). This Table 1 describes a multi-center randomized clinical trial comparing two different approaches to caring for patients with heart failure and atrial fibrillation².

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

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

1.2 The MR CLEAN trial

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

¹HER2 = 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.

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

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

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

Table 1. Baseline Characteristics of the 500 Patients.*

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

The Table was accompanied by the following notes.

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

1.3 Simulated fakestroke data

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

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

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

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

```
fakestroke
```

```
# A tibble: 500 x 18
  studyid trt      age sex   nihss location hx.isch afib    dm mrankin
  <dbl>   <fct>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 z001    Control    53.  Male     21 Right    No       0     0 2
2 z002    Interve~   51.  Male     23 Left     No       1     0 0
3 z003    Control    68.  Fema~    11 Right    No       0     0 0
4 z004    Control    28.  Male     22 Left     No       0     0 0
5 z005    Control    91.  Male     24 Right    No       0     0 0
6 z006    Control    34.  Fema~    18 Left     No       0     0 2
7 z007    Interve~   75.  Male     25 Right    No       0     0 0
8 z008    Control    89.  Fema~    18 Right    No       0     0 0
9 z009    Control    75.  Male     25 Left     No       1     0 2
10 z010   Interve~   26.  Fema~   27 Right    No       0     0 0
# ... with 490 more rows, and 8 more variables: sbp <int>, iv.altep <fct>,
#   time.iv <int>, aspects <int>, ia.occlus <fct>, extra.ica <int>,
#   time.rand <int>, time.punc <int>
```

1.4 Building Table 1 for `fakestroke`: Attempt 1

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

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

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

fs.trt <- c("trt")

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

print(att1)
```

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

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

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

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

1.4.2 `fakestroke` Cleaning Up Categorical Variables

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

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

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

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

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

- rearrange the `ia.occlus` variable to the order⁴ presented in Berkhemer et al. (2015).

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

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

1.5 fakestroke Table 1: Attempt 2

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

	Stratified by trt		p	test
n	Intervention	Control		
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.

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

1.5.1 What summaries should we show?

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

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

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

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

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

```

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

```

1.6 Obtaining a more detailed Summary

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

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

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

```
summary(att2)
```

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

trt: Intervention

	n	miss	p.miss	mean	sd	median	p25	p75	min	max	skew	kurt
age	233	0	0.0	64	18	66	54	76	23	96	-0.34	-0.52
nihss	233	0	0.0	18	5	17	14	21	10	28	0.48	-0.74
sbp	233	0	0.0	146	26	146	129	164	78	214	-0.07	-0.22
time.iv	233	30	12.9	98	45	85	67	110	42	218	1.03	0.08
aspects	233	0	0.0	8	2	9	7	10	5	10	-0.56	-0.98
time.rand	233	2	0.9	203	57	204	152	250	100	300	0.01	-1.16

trt: Control

	n	miss	p.miss	mean	sd	median	p25	p75	min	max	skew	kurt
age	267	0	0.0	65	16	66	56	76	24	94	-0.296	-0.28
nihss	267	0	0.0	18	4	18	14	22	11	25	0.017	-1.24
sbp	267	1	0.4	145	24	145	128	161	82	231	0.156	0.08
time.iv	267	25	9.4	88	26	87	65	116	44	130	0.001	-1.32
aspects	267	4	1.5	9	1	9	8	10	5	10	-1.071	0.36
time.rand	267	0	0.0	214	70	196	149	266	120	360	0.508	-0.93

p-values

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

Standardize mean differences

1 vs 2

age	0.08478764
nihss	0.02405390
sbp	0.04100833
time.iv	0.27691223
aspects	0.19210662
time.rand	0.17720957

Summary of categorical variables

trt: Intervention

var	n	miss	p.miss	level	freq	percent	cum.percent
sex	233	0	0.0	Female	98	42.1	42.1
				Male	135	57.9	100.0
location	233	0	0.0	Left	116	49.8	49.8
				Right	117	50.2	100.0
hx.isch	233	0	0.0	No	204	87.6	87.6
				Yes	29	12.4	100.0
afib	233	0	0.0	0	167	71.7	71.7
				1	66	28.3	100.0

dm	233	0	0.0	0	204	87.6	87.6
				1	29	12.4	100.0
mrankin	233	0	0.0	0	190	81.5	81.5
				1	21	9.0	90.6
				2	12	5.2	95.7
				> 2	10	4.3	100.0
iv.altep	233	0	0.0	No	30	12.9	12.9
				Yes	203	87.1	100.0
ia.occlus	233	0	0.0	Intracranial	ICA	1	0.4
				ICA with	M1	59	25.3
					M1	154	66.1
					M2	18	7.7
					A1 or A2	1	0.4
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
				ICA with	M1	75	28.2
					M1	165	62.0
					M2	21	7.9
					A1 or A2	2	0.8
extra.ica	267	1	0.4	0	196	73.7	73.7
				1	70	26.3	100.0

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

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

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

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

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

1.7.1 Approach A: Save and open in Excel

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

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

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

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

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

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

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

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

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

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

you should get something usable again.

Or, in Word,

1. insert the text

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

After dropping blank columns, the result looks pretty good.

1.8 A Controlled Biological Experiment - The Blood-Brain Barrier

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

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

1.9 The `bloodbrain.csv` file

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

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

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

```
bloodbrain
```

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

```

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

```

1.10 A Table 1 for bloodbrain

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

```

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

dput(names(bloodbrain))

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

```

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

```

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

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

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

summary(bb.att1)

```

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

```

solution: BD

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

      skew kurt
wt.init -0.39  0.7
wt.loss -0.10  0.2

```

```

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

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

Standardize mean differences
      1 vs 2
wt.init  0.08435244
wt.loss  0.14099823
wt.tumor 0.50397184
brain    1.23884159
liver    0.01089667
tlratio  0.34611465
logTL   0.32420504
=====
#### Summary of categorical variables ####

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

```

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

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

solution: NS

var	n	miss	p.miss	level	freq	percent	cum.percent
sactime	17	0	0.0	0.5	4	23.5	23.5
				3	5	29.4	52.9
				24	4	23.5	76.5
				72	4	23.5	100.0
postin	17	0	0.0	9	2	11.8	11.8
				10	13	76.5	88.2
				11	2	11.8	100.0
sex	17	0	0.0	F	13	76.5	76.5
				M	4	23.5	100.0

p-values

	pApprox	pExact
sactime	0.9739246	1
postin	0.8309504	1
sex	1.0000000	1

Standardize mean differences

	1 vs 2
sactime	0.1622214
postin	0.2098877
sex	0.0000000

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

1.10.1 Generate final Table 1 for bloodbrain

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

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

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

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

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

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

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

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

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

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

1.10.2 A More Finished Version (after Cleanup in Word)

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

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

Table 1 Notes:

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

Chapter 2

Linear Regression on a small SMART data set

2.1 BRFSS and SMART

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

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

2.1.1 Key resources

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

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

2.2 The `smartcle1` data: Cookbook

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

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

Variable	Description
<code>physhealth</code>	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?
<code>menthealth</code>	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?
<code>poorhealth</code>	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?
<code>genhealth</code>	Would you say that in general, your health is ... (five categories: Excellent, Very Good, Good, Fair or Poor)
<code>bmi</code>	Body mass index, in kg/m ²
<code>female</code>	Sex, 1 = female, 0 = male
<code>internet30</code>	Have you used the internet in the past 30 days? (1 = yes, 0 = no)
<code>exerany</code>	During the past month, other than your regular job, did you participate in any physical activities or exercises such as running, calisthenics, golf, gardening, or walking for exercise? (1 = yes, 0 = no)
<code>sleephrs</code>	On average, how many hours of sleep do you get in a 24-hour period?
<code>alcdays</code>	How many days during the past 30 days did you have at least one drink of any alcoholic beverage such as beer, wine, a malt beverage or liquor?

```
str(smartcle1)
```

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

2.3 smartcle2: Omitting Missing Observations: Complete-Case Analyses

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

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

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

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

Skim summary statistics
n obs: 1036
n variables: 11

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

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

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

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

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

smartcle2

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

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

11 variables, but with lots of missing data.

2.4 Summarizing the `smartcle2` data numerically

2.4.1 The New Toy: The `skim` function

```
skim(smartcle2, -SEQNO)

Skim summary statistics
n obs: 896
n variables: 10

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

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

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

2.4.2 The usual `summary` for a data frame

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

```
summary(smartcle2)

  SEQNO          physhealth        menthealth        genhealth
Min. :2.016e+09 Min. : 0.00  Min. : 0.000 1_Excellent:155
1st Qu.:2.016e+09 1st Qu.: 0.00  1st Qu.: 0.000 2_VeryGood :306
Median :2.016e+09 Median : 0.00  Median : 0.000 3_Good      :295
Mean   :2.016e+09 Mean   : 3.99  Mean   : 2.693 4_Fair       :102
3rd Qu.:2.016e+09 3rd Qu.: 2.00  3rd Qu.: 2.000 5_Poor       : 38
Max.   :2.016e+09 Max.  :30.00  Max.  :30.000

  bmi           female        internet30        exerany
Min. :12.71  Min. :0.0000  Min. :0.0000  Min. :0.0000
1st Qu.:23.70 1st Qu.:0.0000  1st Qu.:1.0000  1st Qu.:1.0000
Median :26.80  Median :1.0000  Median :1.0000  Median :1.0000
Mean   :27.87  Mean   :0.5848  Mean   :0.8147  Mean   :0.7667
3rd Qu.:30.53 3rd Qu.:1.0000  3rd Qu.:1.0000  3rd Qu.:1.0000
Max.   :66.06  Max.  :1.0000  Max.  :1.0000  Max.  :1.0000
```

```

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

```

2.4.3 The `describe` function in `Hmisc`

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

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

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

```
3 Variables     896 Observations
```

bmi

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

```
lowest : 12.71 13.34 14.72 16.22 17.30, highest: 56.89 57.04 60.95 61.84 66.06
```

genhealth

	n	missing	distinct
896	0	5	

Value	1_Excellent	2_VeryGood	3_Good	4_Fair	5_Poor
Frequency	155	306	295	102	38
Proportion	0.173	0.342	0.329	0.114	0.042

female

	n	missing	distinct	Info	Sum	Mean	Gmd
896	0	2	0.728	524	0.5848	0.4862	

2.5 Counting as exploratory data analysis

Counting things can be amazingly useful.

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

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

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

```

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

```

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

```

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

```

```

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

```

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

2.5.2 What's the distribution of `sleephrs`?

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

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

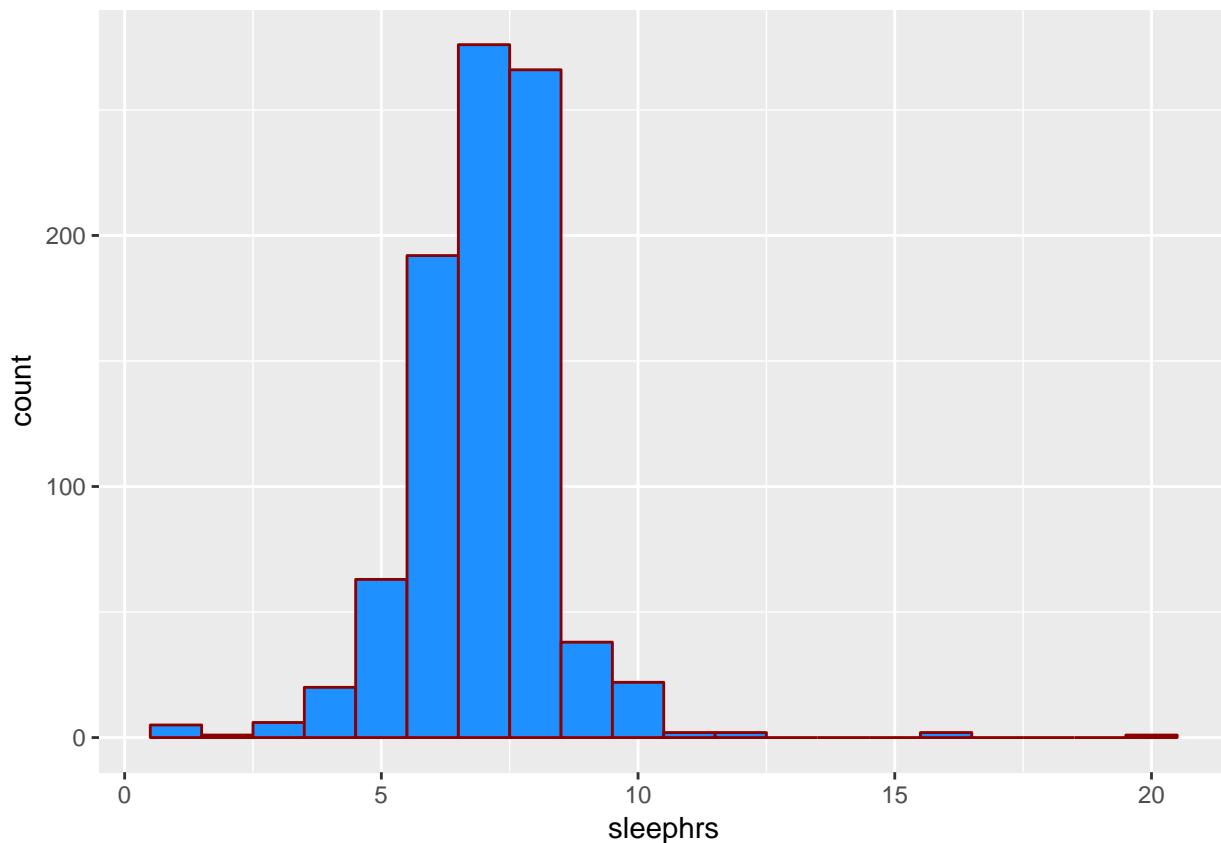
```

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

```

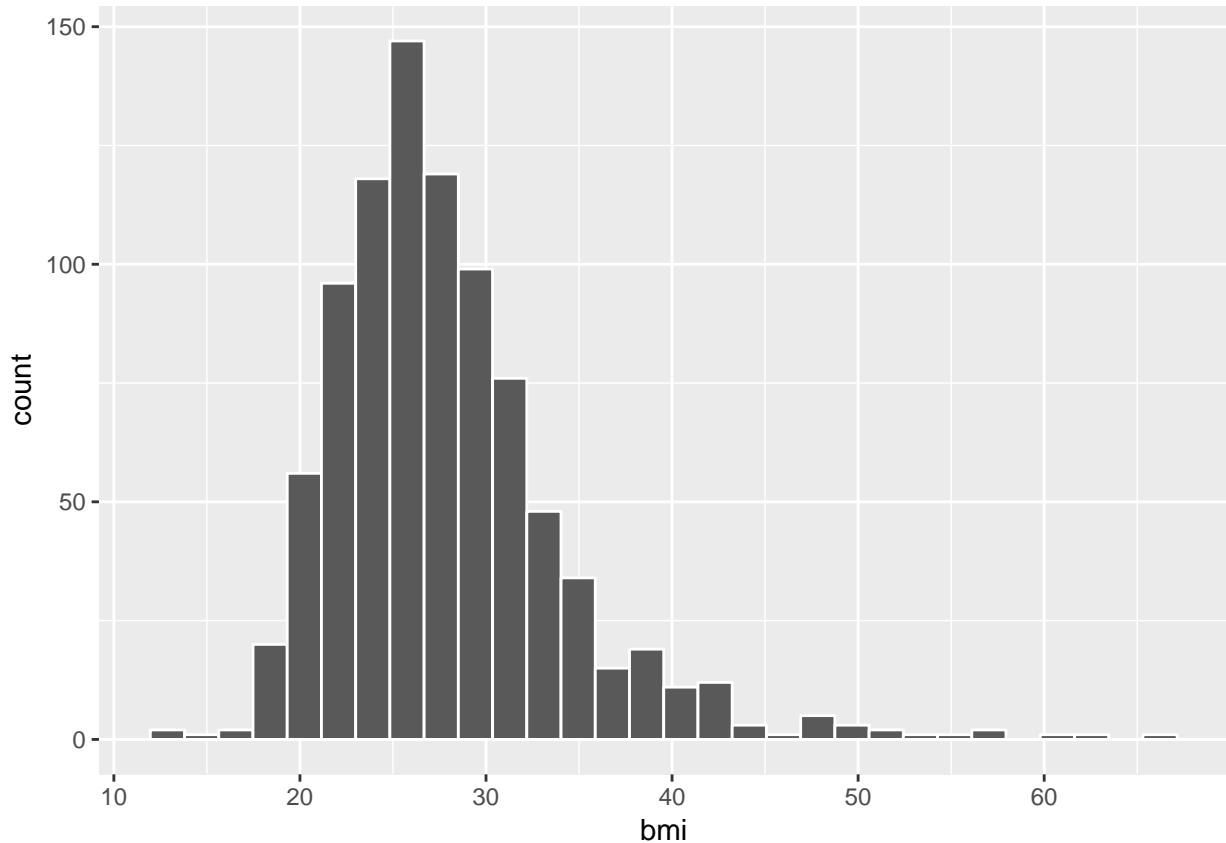
Of course, a natural summary of a quantitative variable like this would be graphical.

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



2.5.3 What's the distribution of BMI?

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



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

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

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

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

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

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

2.5.6 Is obesity associated with sex, in these data?

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

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

2.5.7 Comparing `sleephrs` summaries by obesity status

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

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

# A tibble: 2 x 4
`bmi < 30` `mean(sleephrs)` `median(sleephrs)` `q75
<lgl>          <dbl>           <dbl>      <dbl>
1 FALSE        6.93            7            8.
2 TRUE         7.06            7            8.
```

2.5.8 The `skim` function within a pipe

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

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

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

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

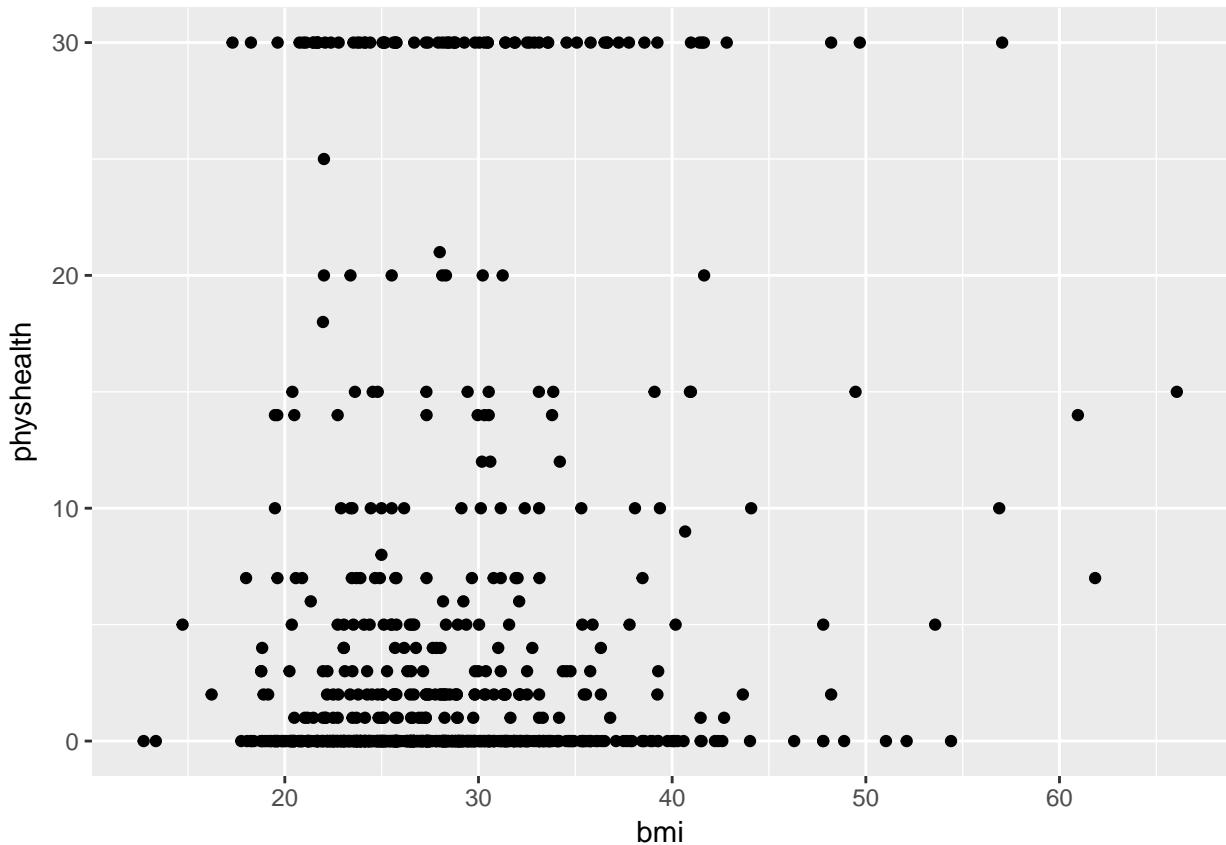
```
Variable type: numeric
exerany variable missing complete    n mean     sd p0 p25 median p75
  0      bmi       0       209 209 29.57 7.46 18   24.11 28.49 33.13
  1      bmi       0       687 687 27.35 5.84 12.71 23.7   26.52 29.81
p100
```

```
66.06
60.95
```

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

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

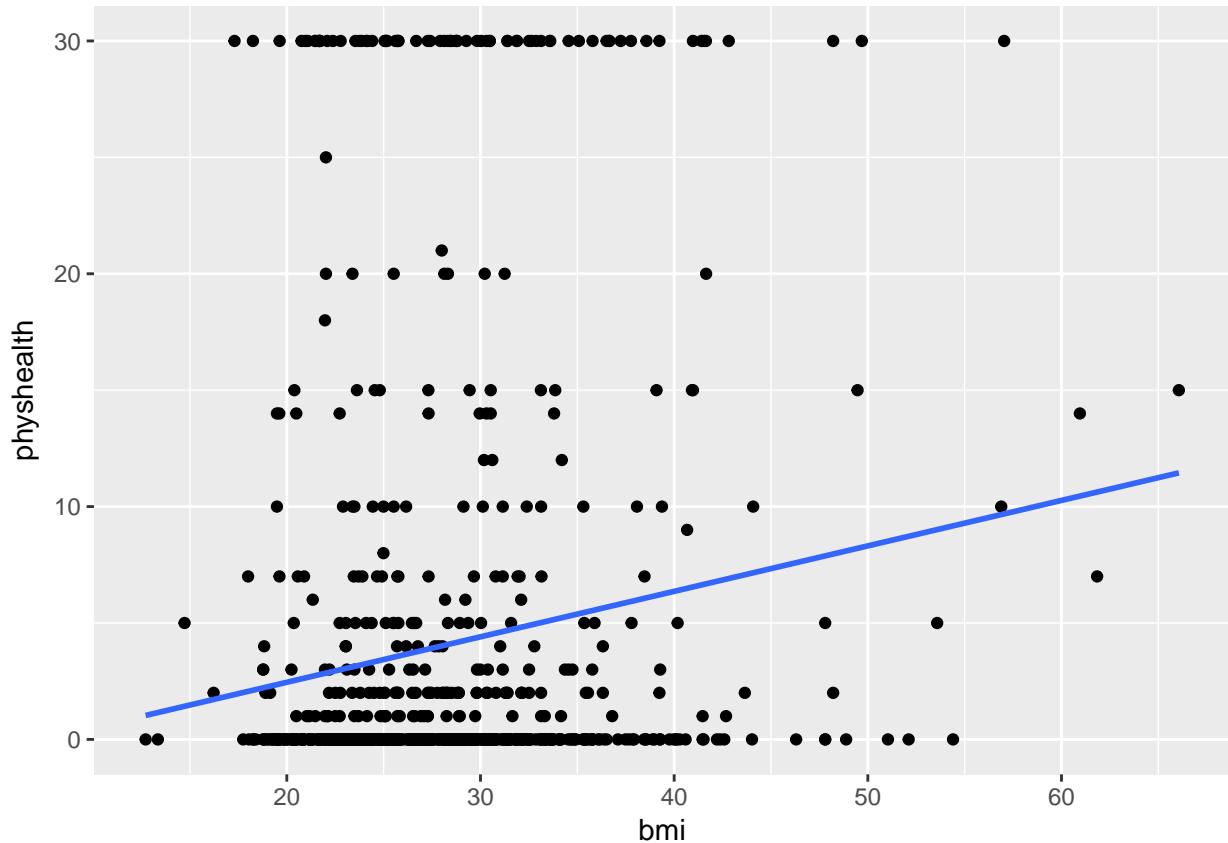
```
ggplot(data = smartcle2, aes(x = bmi, y = physhealth)) +
  geom_point()
```



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

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

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



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

2.6.1 Fitting a Simple Regression Model

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

model_A
```

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

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

```
summary(model_A)
```

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

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)							
(Intercept)	-1.45143	1.29185	-1.124	0.262							
bmi	0.19527	0.04521	4.319	1.74e-05 ***							

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

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

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

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

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

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

The fitted model predicts `physhealth` with the equation $-1.45 + 0.195 \cdot \text{bmi}$, as we can read off from the model coefficients.

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

2.6.2.1 Residuals

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

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

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

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

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

2.6.2.2 Coefficients section

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

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

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

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

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

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

2.6.2.3 Model Fit Summaries

The `summary` of a linear model also displays:

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

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

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

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

```
anova(model_A)
```

Analysis of Variance Table

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

2.6.3 Using the broom package

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

2.6.3.1 The `tidy` function

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

```
tidy(model_A)
```

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

2.6.3.2 The `glance` function

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

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

```
glance(model_A)
```

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

2.6.3.3 The augment function

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

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

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

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

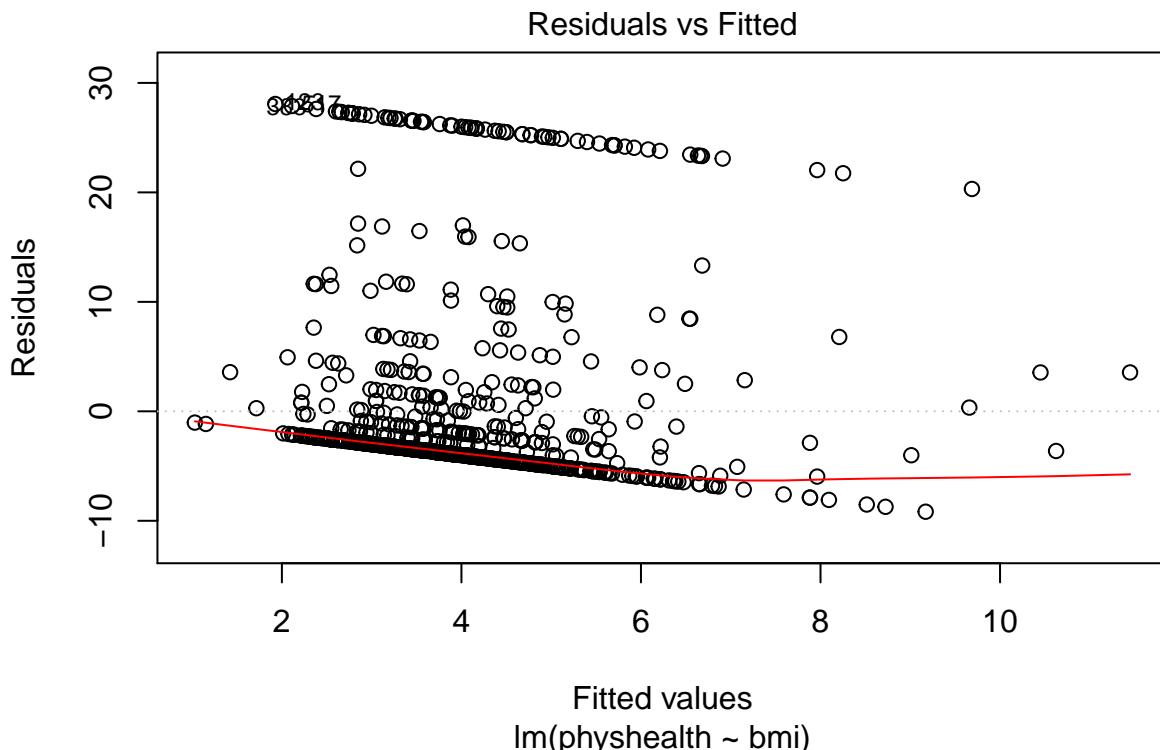
	physhealth	bmi	.fitted	.se.fit	.resid	.hat	.sigma
1	0	26.69	3.760430	0.2907252	-3.76043009	0.001154651	8.559600
2	0	23.70	3.176561	0.3422908	-3.17656119	0.001600574	8.559865
3	1	26.92	3.805343	0.2890054	-2.80534308	0.001141030	8.560010
4	0	21.66	2.778202	0.4005101	-2.77820248	0.002191352	8.560020
5	5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

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

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

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

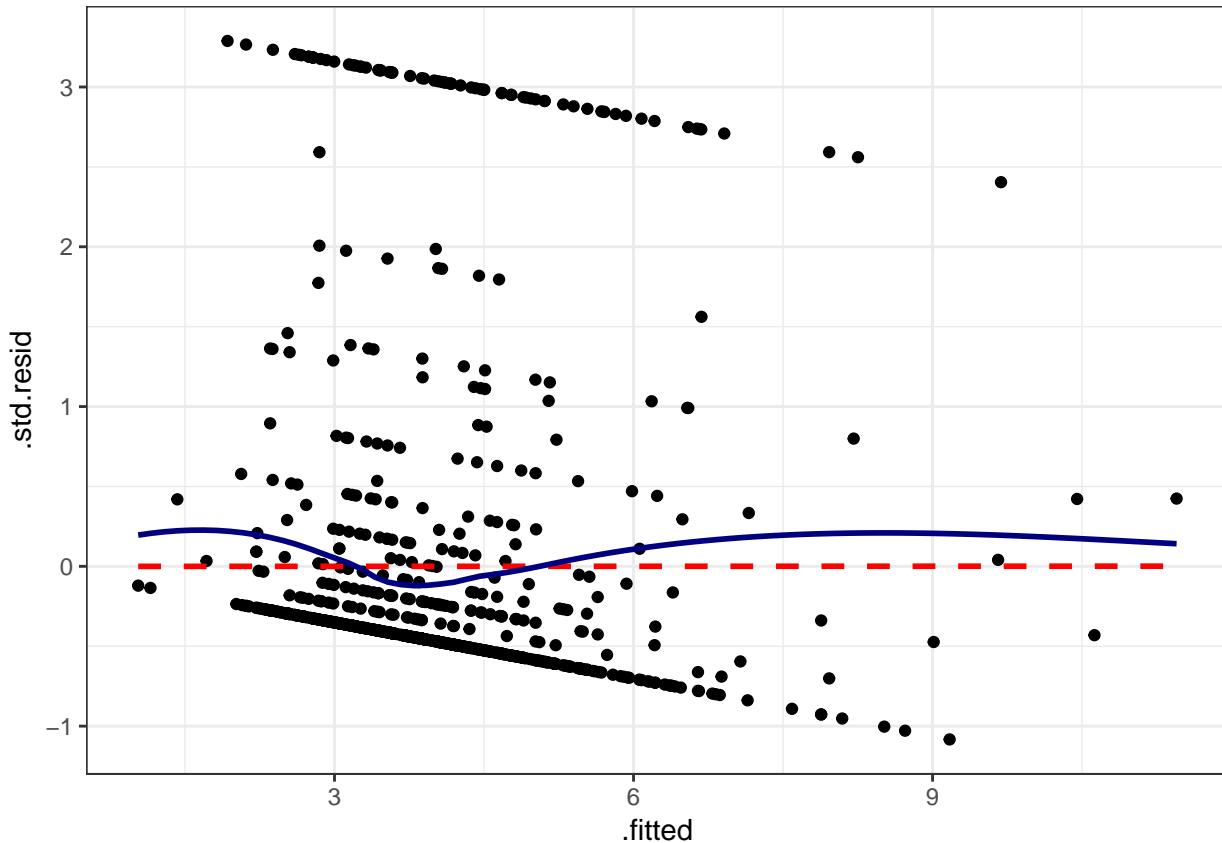
```
plot(model_A, which = 1)
```



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

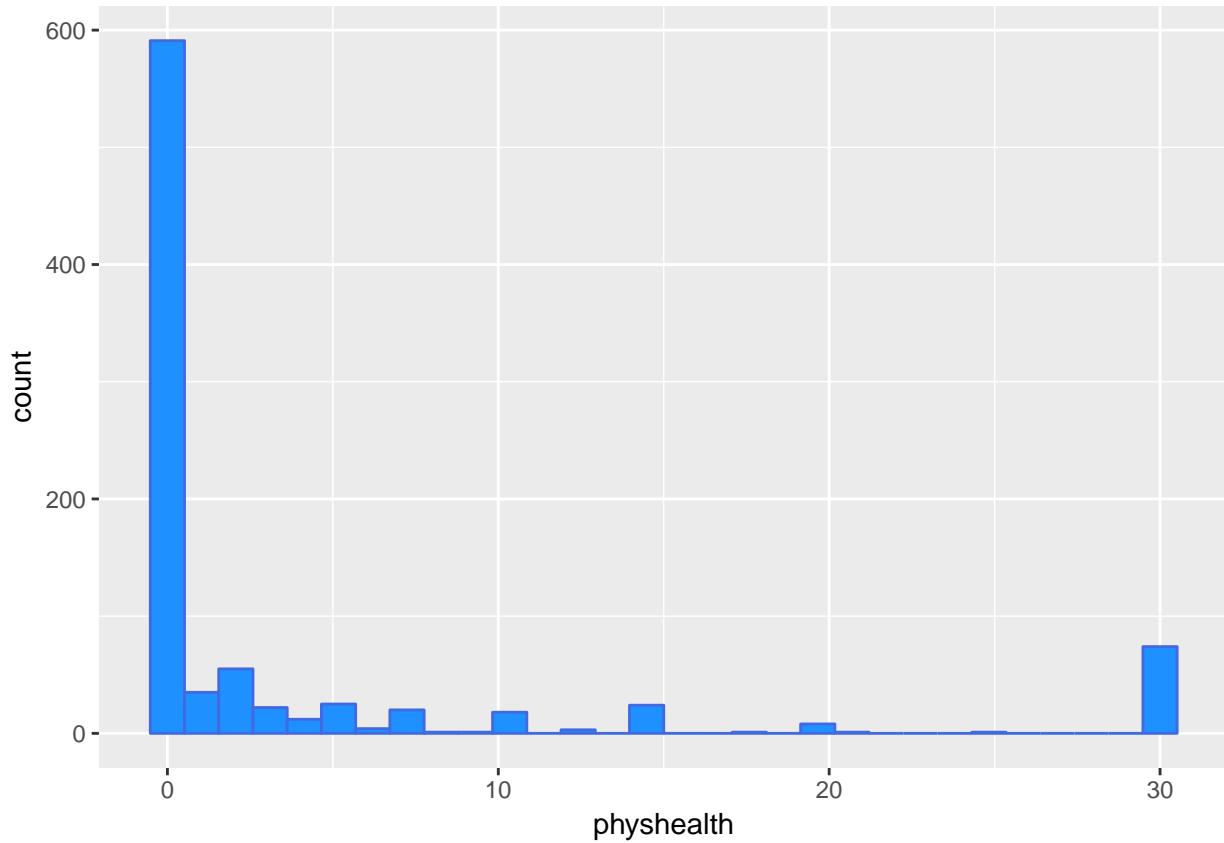
If you prefer, here’s a `ggplot2` version of a similar plot, now looking at standardized residuals instead of raw residuals, and adding a loess smooth and a linear fit to the result.

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



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

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



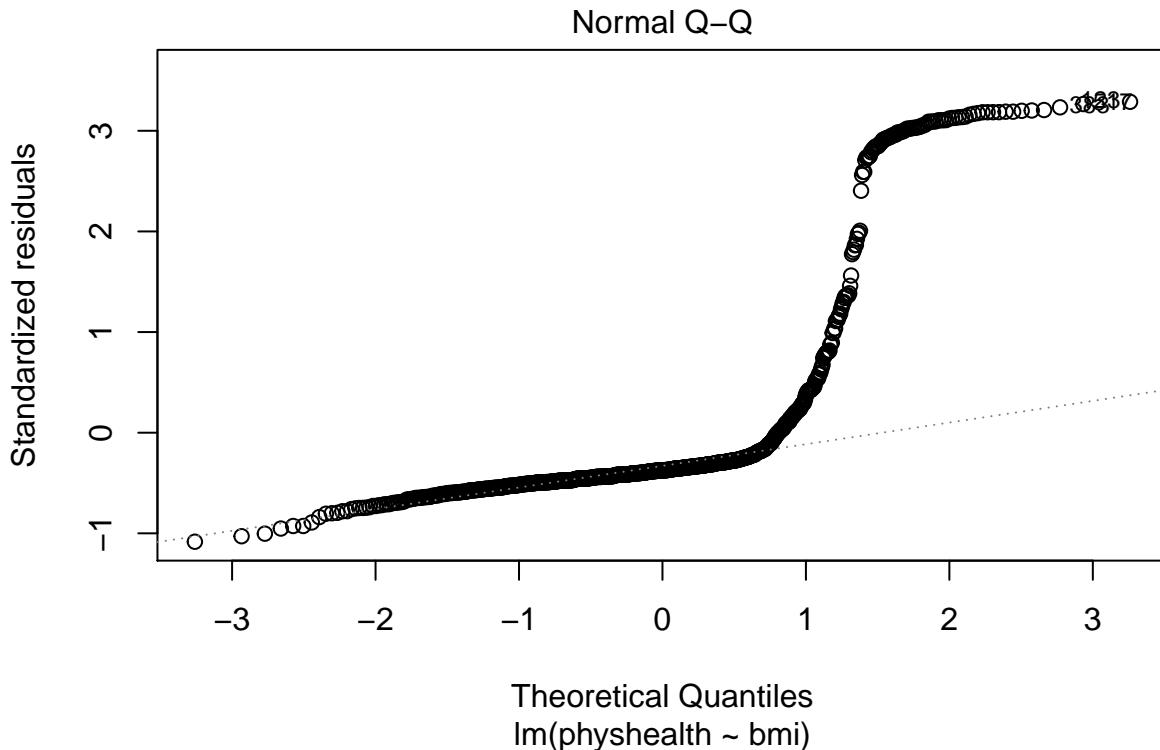
```
smartcle2 %>% count(physhealth == 0, physhealth == 30)
```

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

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

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

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



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

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

2.7 A New Small Study: Predicting BMI

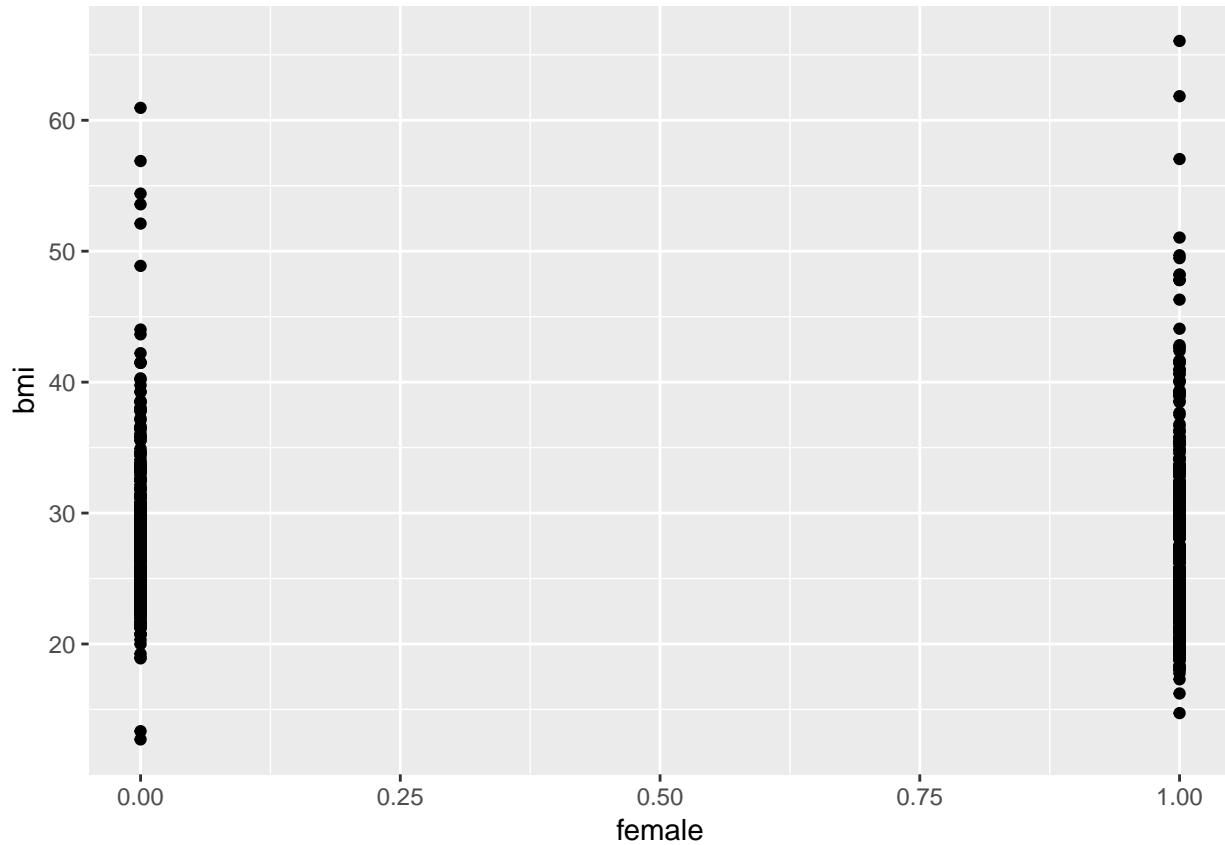
We'll begin by investigating the problem of predicting `bmi`, at first with just three regression inputs: `sex`, `exerany` and `sleephrs`, in our new `smartcle2` data set.

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

2.7.1 Does `female` predict `bmi` well?

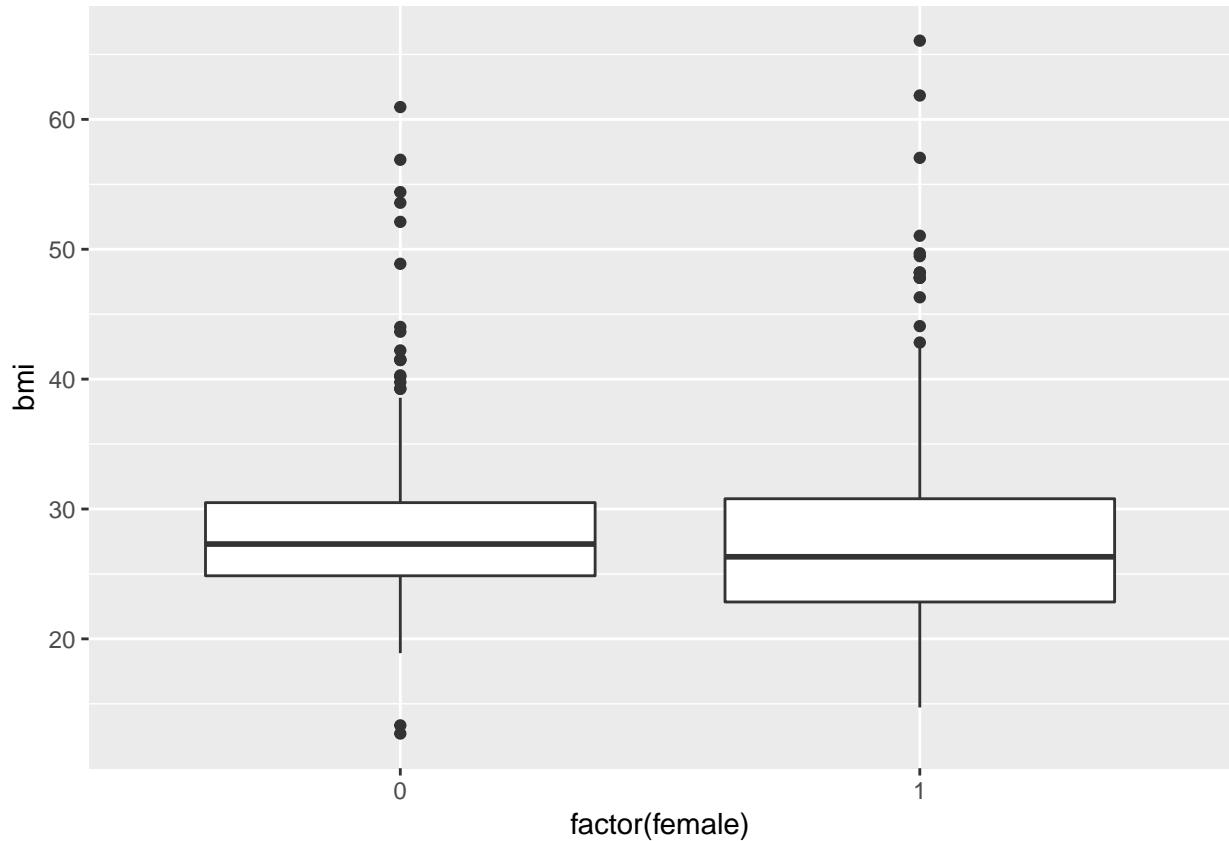
2.7.1.1 Graphical Assessment

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



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

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



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

2.8 c2_m1: A simple t-test model

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

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

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

```
summary(c2_m1)
```

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

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

```
Coefficients:
```

```

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

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

confint(c2_m1)

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

```

The model suggests, based on these 896 subjects, that

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

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

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

Two Sample t-test

```

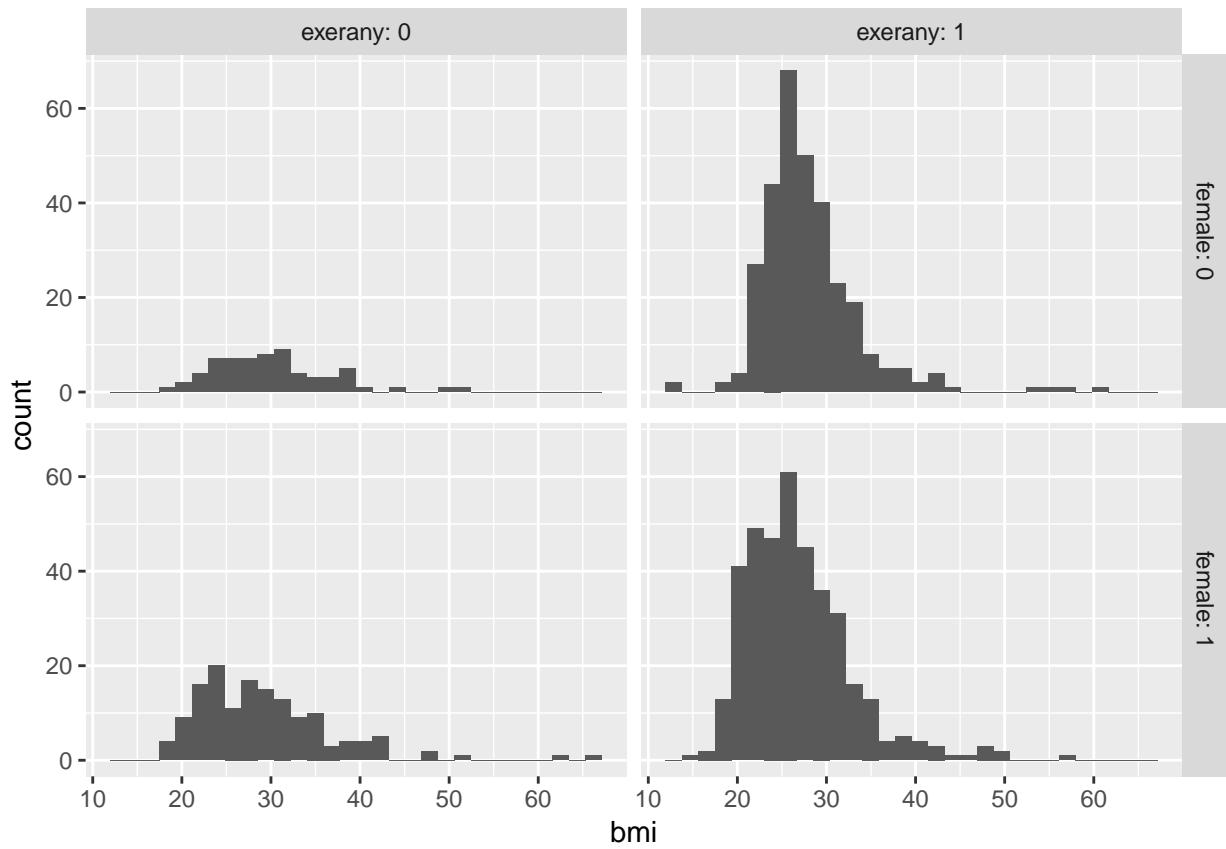
data: bmi by female
t = 1.9752, df = 894, p-value = 0.04855
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.00539878 1.68605160
sample estimates:
mean in group 0 mean in group 1
 28.36000      27.51427

```

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

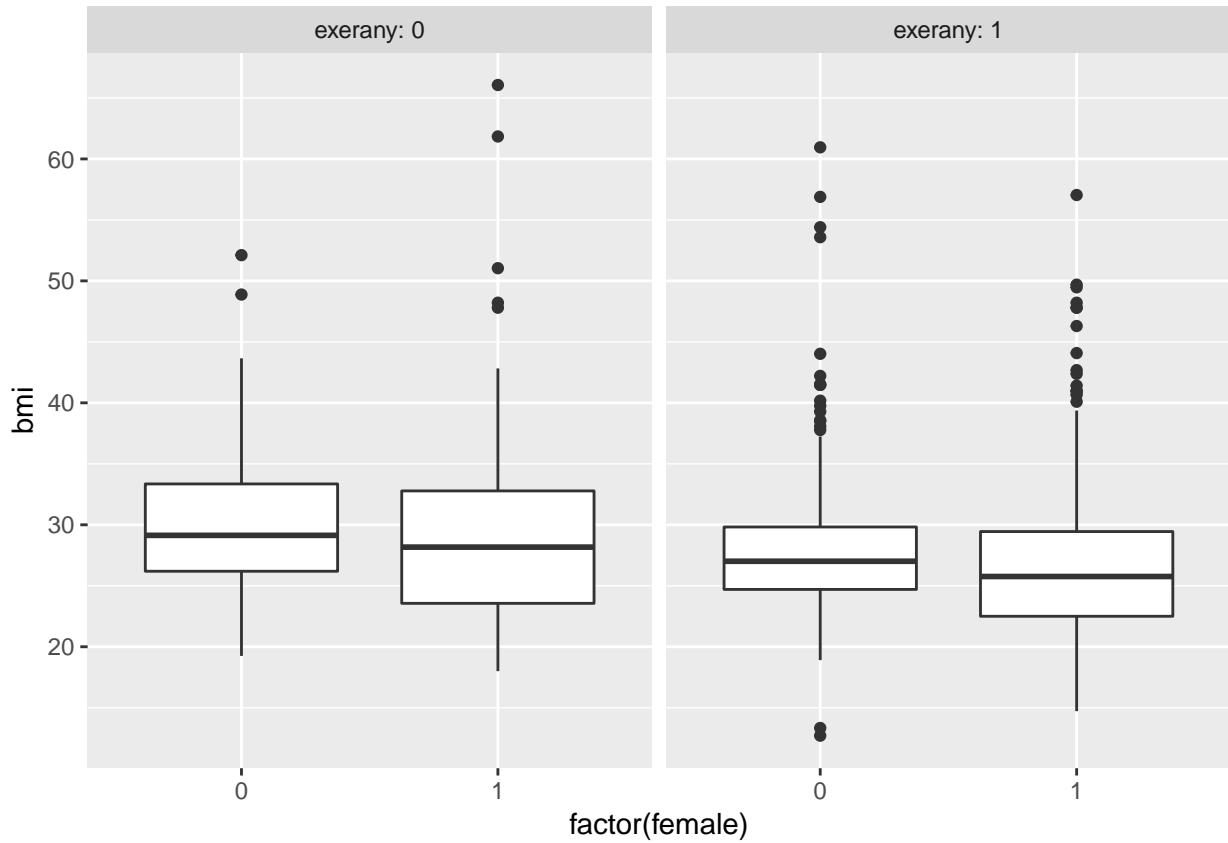
When we add in the information about `exerany` to our original model, we might first picture the data. We could look at separate histograms,

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

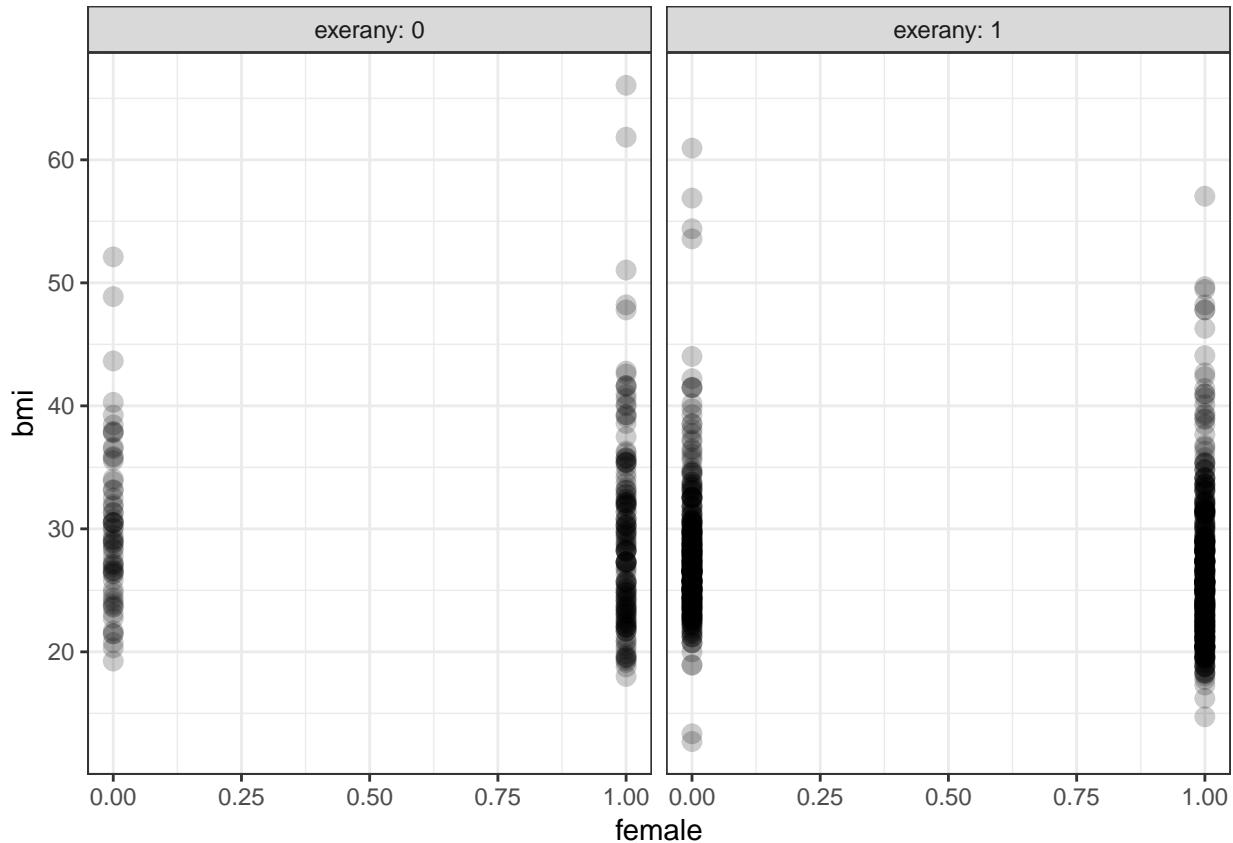


or maybe boxplots?

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



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



OK. Let's try fitting a model.

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

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

Coefficients:

(Intercept)	female	exerany
30.334	-1.095	-2.384

This new model predicts only four predicted values:

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

For those who did not exercise, the model is:

- $bmi = 30.334 - 1.095 \text{ female}$

and for those who did exercise, the model is:

- $bmi = 27.95 - 1.095 \text{ female}$

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

```
summary(c2_m2)
```

Call:

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	30.3335	0.5231	57.99	< 2e-16 ***
female	-1.0952	0.4262	-2.57	0.0103 *
exerany	-2.3836	0.4965	-4.80	1.86e-06 ***

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

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

```
confint(c2_m2)
```

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

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

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

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

```
anova(c2_m2)
```

Analysis of Variance Table

Response: `bmi`

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
female	1	156	155.61	3.9977	0.04586 *
exerany	1	897	896.93	23.0435	1.856e-06 ***
Residuals	893	34759	38.92		

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

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

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

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

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

Coefficients:

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

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

- $\text{bmi} = 30.136 - 0.810(0) - 2.145(1) - 0.359(0)(1) = 30.136 - 2.145 = 27.991$

And for a female who exercises, the model predicts

- $\text{bmi} = 30.136 - 0.810(1) - 2.145(1) - 0.359(1)(1) = 30.136 - 0.810 - 2.145 - 0.359 = 26.822$

For those who did not exercise, the model is:

- $\text{bmi} = 30.136 - 0.81 \text{ female}$

But for those who did exercise, the model is:

- $\text{bmi} = (30.136 - 2.145) + (-0.810 + (-0.359)) \text{ female}$, or „,
- $\text{bmi} = 27.991 - 1.169 \text{ female}$

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

```
summary(c2_m3)
```

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	30.1359	0.7802	38.624	<2e-16 ***
female	-0.8104	0.9367	-0.865	0.3872
exerany	-2.1450	0.8575	-2.501	0.0125 *
female:exerany	-0.3592	1.0520	-0.341	0.7328

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

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

```
confint(c2_m3)
```

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

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

```
anova(c2_m3)
```

Analysis of Variance Table

Response: bmi

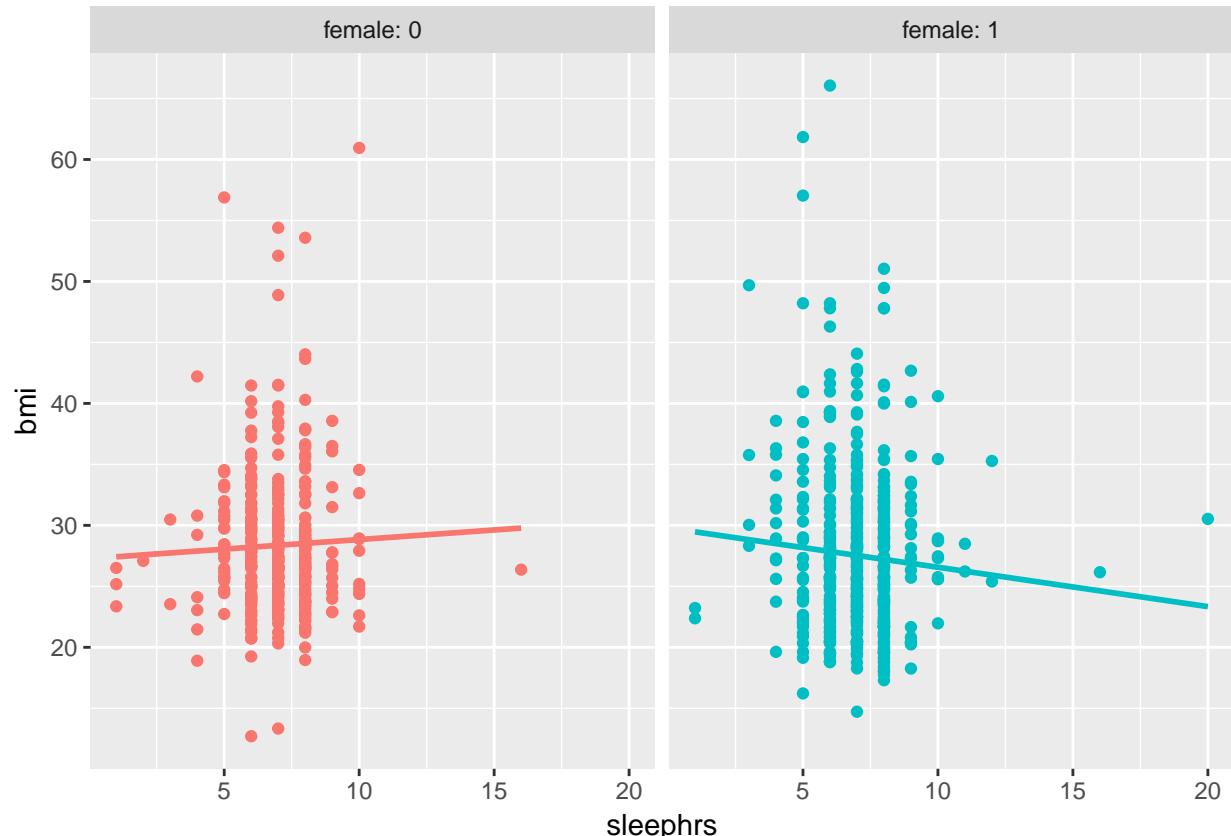
	Df	Sum Sq	Mean Sq	F value	Pr(>F)
female	1	156	155.61	3.9938	0.04597 *
exerany	1	897	896.93	23.0207	1.878e-06 ***
female:exerany	1	5	4.54	0.1166	0.73283
Residuals	892	34754	38.96		

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

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

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

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



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

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

summary(c2_m4)
```

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	27.2661	1.6320	16.707	<2e-16 ***
female	2.5263	2.0975	1.204	0.229
sleephrs	0.1569	0.2294	0.684	0.494
female:sleephrs	-0.4797	0.2931	-1.636	0.102

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

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

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

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

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

```
glance(c2_m4)
```

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

```
glance(c2_m1)
```

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

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

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

Analysis of Variance Table

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

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

2.12 Making Predictions with a Linear Regression Model

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

```
c2_m4
```

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

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

2.12.1 Fitting an Individual Prediction and 95% Prediction Interval

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

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

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

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

2.12.2 Confidence Interval for an Average Prediction

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

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

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

- How does this result compare to the prediction interval?

2.12.3 Fitting Multiple Individual Predictions to New Data

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

```
c2_new2 <- data_frame(subjectid = 1001:1006, female = c(1, 1, 1, 0, 0, 0), sleephrs = c(7, 8, 9, 7, 8, 9)
pred2 <- predict(c2_m4, newdata = c2_new2, interval = "prediction", level = 0.95) %>%tbl_df
result2 <- bind_cols(c2_new2, pred2)
result2
```

```
# A tibble: 6 x 6
  subjectid female sleephrs   fit    lwr    upr
  <int>     <dbl>    <dbl> <dbl> <dbl> <dbl>
1     1001      1.      7.  27.5  15.1  39.9
2     1002      1.      8.  27.2  14.8  39.6
3     1003      1.      9.  26.9  14.5  39.3
4     1004      0.      7.  28.4  16.0  40.8
5     1005      0.      8.  28.5  16.1  40.9
6     1006      0.      9.  28.7  16.2  41.1
```

The `result2` tibble contains predictions for each scenario.

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

2.12.4 Simulation to represent predictive uncertainty in Model 4

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

```
c2_m4
```

```
Call:
lm(formula = bmi ~ female * sleephrs, data = smartcl2)

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

glance(c2_m4)
```

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

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

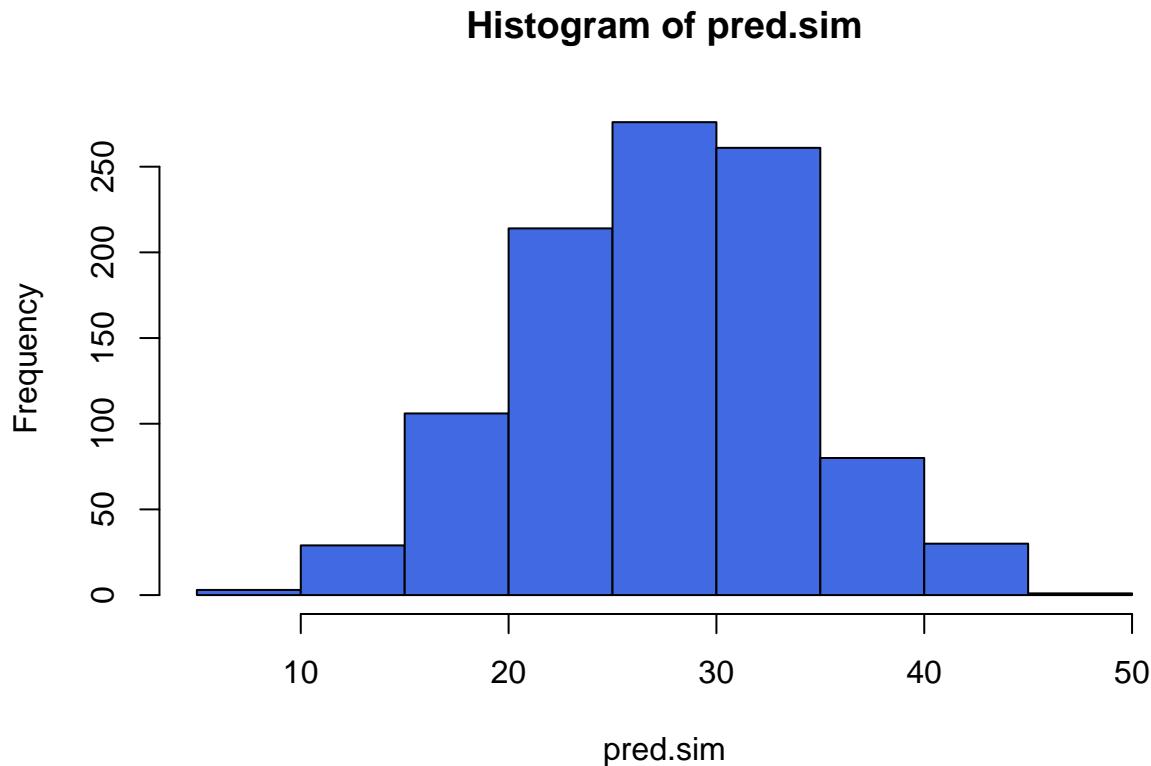
For a female respondent sleeping eight hours, recall that our point estimate (predicted value) of `bmi` is 27.21

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

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

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

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



```
mean(pred.sim)
[1] 27.41856
quantile(pred.sim, c(0.025, 0.975))
 2.5%    97.5%
14.48487 40.16778
```

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

2.13 Centering the model

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

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

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

summary(c2_m4_c)
```

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	28.3681	0.3274	86.658	<2e-16 ***
female	-0.8420	0.4280	-1.967	0.0495 *
sleephrs_c	0.1569	0.2294	0.684	0.4940
female:sleephrs_c	-0.4797	0.2931	-1.636	0.1021

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

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

What has changed as compared to the original c2_m4?

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

So our new model on centered data is:

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

In our new (centered `sleephrs_c`) model,

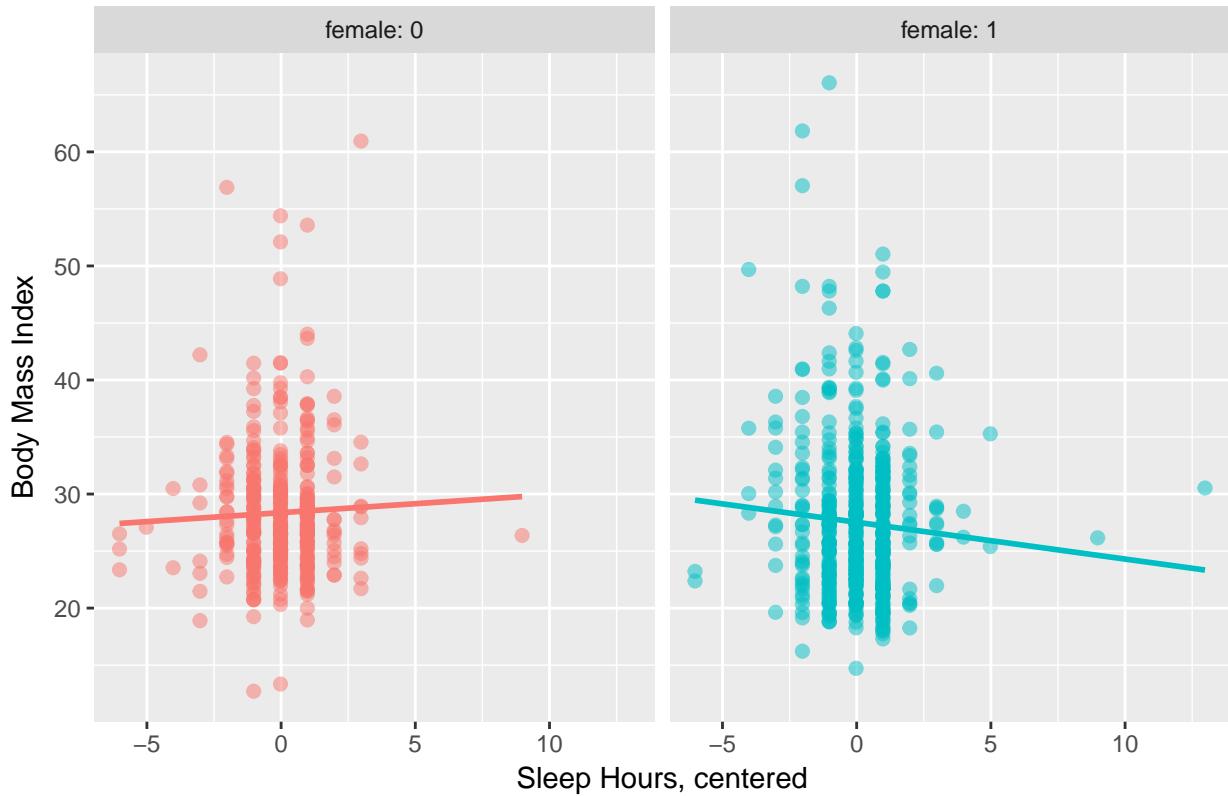
- the main effect of `female` now corresponds to a predictive difference (female - male) in `bmi` with `sleephrs` at its mean value, 7.02 hours,
- the intercept term is now the predicted `bmi` for a male respondent who sleeps an average number of hours, and
- the product term corresponds to the change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

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

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

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

Model `c2_m4` on centered data



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

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

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

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

2.14.1 Refitting model c2_m4 to the rescaled data

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

summary(c2_m4_z)
```

Call:

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	28.3681	0.3274	86.658	<2e-16 ***
female	-0.8420	0.4280	-1.967	0.0495 *
sleephrs_z	0.4637	0.6778	0.684	0.4940
female:sleephrs_z	-1.4173	0.8661	-1.636	0.1021

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

Residual standard error: 6.31 on 892 degrees of freedom

Multiple R-squared: 0.008341, Adjusted R-squared: 0.005006

F-statistic: 2.501 on 3 and 892 DF, p-value: 0.05818

2.14.2 Interpreting the model on rescaled data

What has changed as compared to the original c2_m4?

- Our original model was $bmi = 27.26 + 2.53 \text{female} + 0.16 \text{sleephrs} - 0.48 \text{female} \times \text{sleephrs}$
- Our model on centered sleephrs was $bmi = 28.37 - 0.84 \text{female} + 0.16 \text{centered sleephrs_c} - 0.48 \text{female} \times \text{centered sleephrs_c}$.
- Our new model on rescaled sleephrs is $bmi = 28.37 - 0.84 \text{female} + 0.46 \text{rescaled sleephrs_z} - 1.42 \text{female} \times \text{rescaled sleephrs_z}$.

So our rescaled model is:

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

In this new rescaled (sleephrs_z) model, then,

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

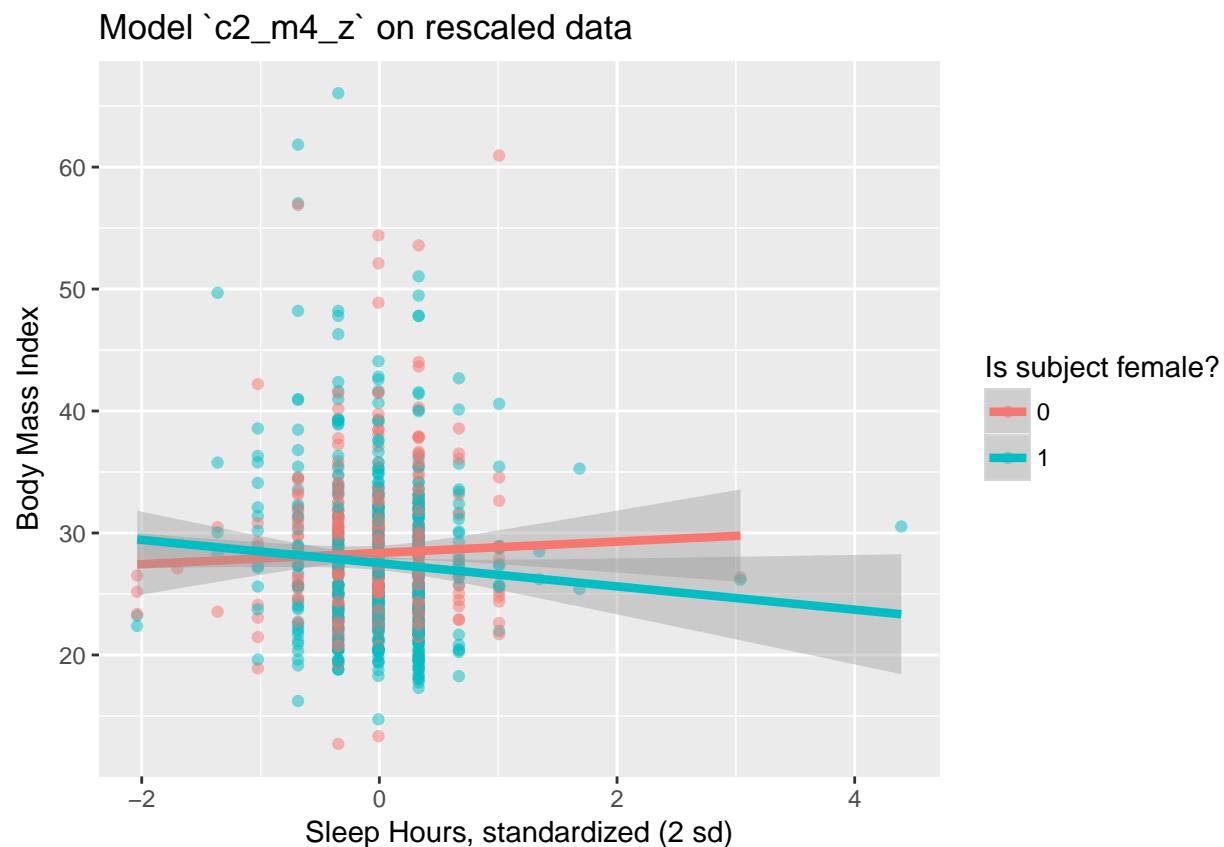
as before, but now we also have that:

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

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

2.14.3 Plot of model on rescaled data

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



2.15 c2_m5: What if we add more variables?

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

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

```

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

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

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

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

```

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

```
anova(c2_m5)
```

Analysis of Variance Table

```

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

```

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

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

Analysis of Variance Table

```

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

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

3. What does the output below let us conclude?

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

Analysis of Variance Table

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

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

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

2.16 c2_m6: Would adding self-reported health help?

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

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

Call:

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	26.49498	1.31121	20.206	< 2e-16 ***
female	-0.85520	0.41969	-2.038	0.041879 *
exerany	-1.61968	0.50541	-3.205	0.001400 **
sleephrs	-0.12719	0.13613	-0.934	0.350368
internet30	2.02498	0.53898	3.757	0.000183 ***
alcdays	-0.08431	0.02537	-3.324	0.000925 ***
genhealth2_VeryGood	2.10537	0.59408	3.544	0.000415 ***
genhealth3_Good	4.08245	0.60739	6.721	3.22e-11 ***
genhealth4_Fair	4.99213	0.80178	6.226	7.37e-10 ***
genhealth5_Poor	3.11025	1.12614	2.762	0.005866 **

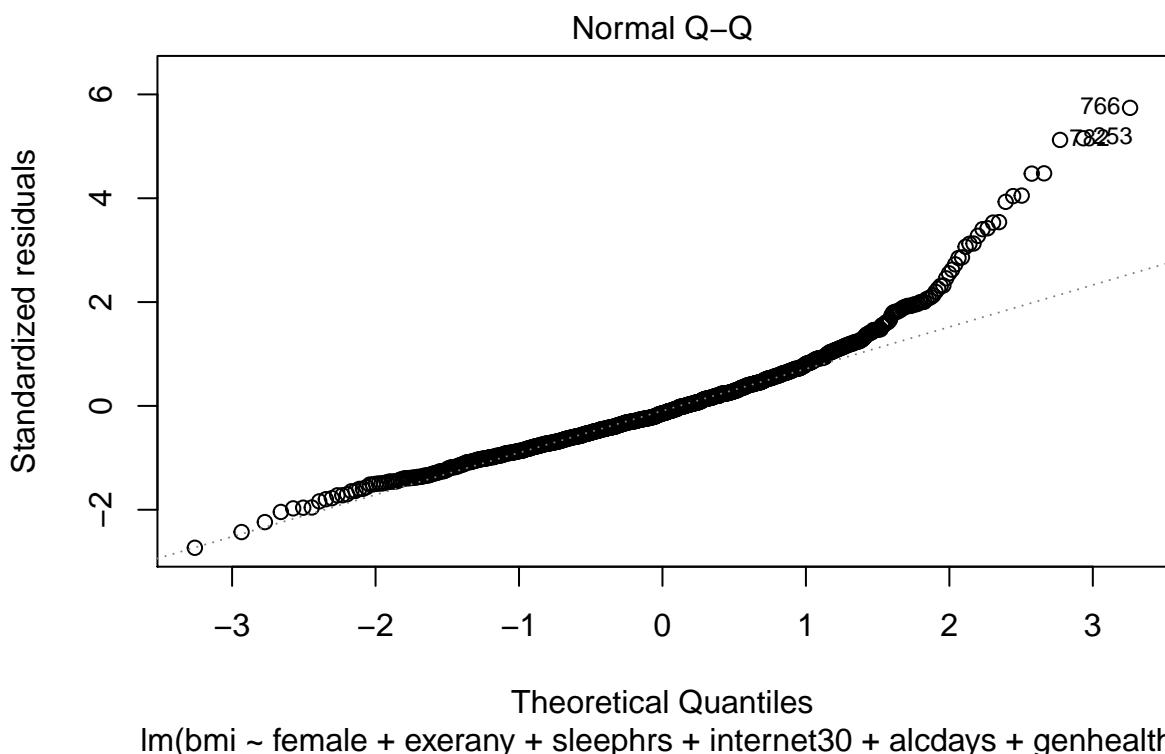
```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 5.993 on 886 degrees of freedom

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

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

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



2.17 c2_m7: What if we added the `menthealth` variable?

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

summary(c2_m7)
```

Call:
 $\text{lm}(\text{formula} = \text{bmi} \sim \text{female} + \text{exerany} + \text{sleephrs} + \text{internet30} +
\text{alcdays} + \text{genhealth} + \text{physhealth} + \text{menthealth}, \text{data} = \text{smartcle2})$

Residuals:

Min	1Q	Median	3Q	Max
-----	----	--------	----	-----

```

-16.060 -3.804 -0.890  2.794  33.972

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

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

```

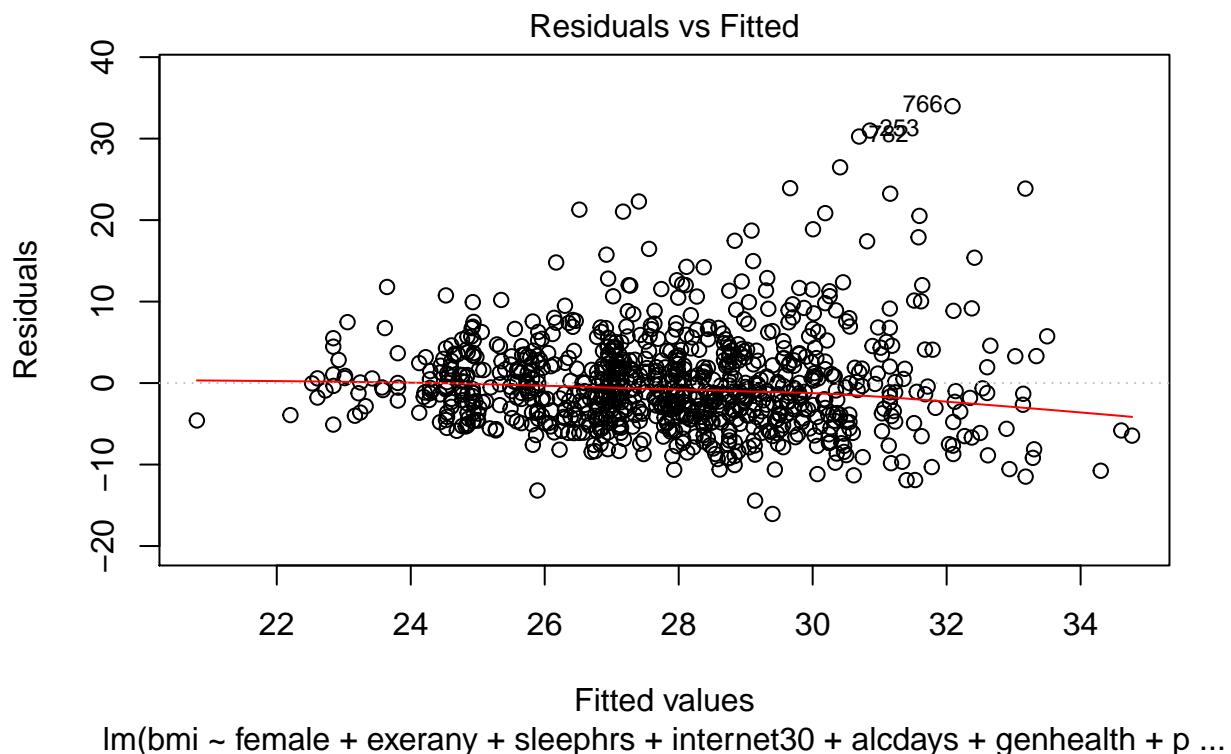
2.18 Key Regression Assumptions for Building Effective Prediction Models

1. Validity - the data you are analyzing should map to the research question you are trying to answer.
 - The outcome should accurately reflect the phenomenon of interest.
 - The model should include all relevant predictors. (It can be difficult to decide which predictors are necessary, and what to do with predictors that have large standard errors.)
 - The model should generalize to all of the cases to which it will be applied.
 - Can the available data answer our question reliably?
2. Additivity and linearity - most important assumption of a regression model is that its deterministic component is a linear function of the predictors. We often think about transformations in this setting.
3. Independence of errors - errors from the prediction line are independent of each other
4. Equal variance of errors - if this is violated, we can more efficiently estimate 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

2.18.1 Checking Assumptions in model c2_m7

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

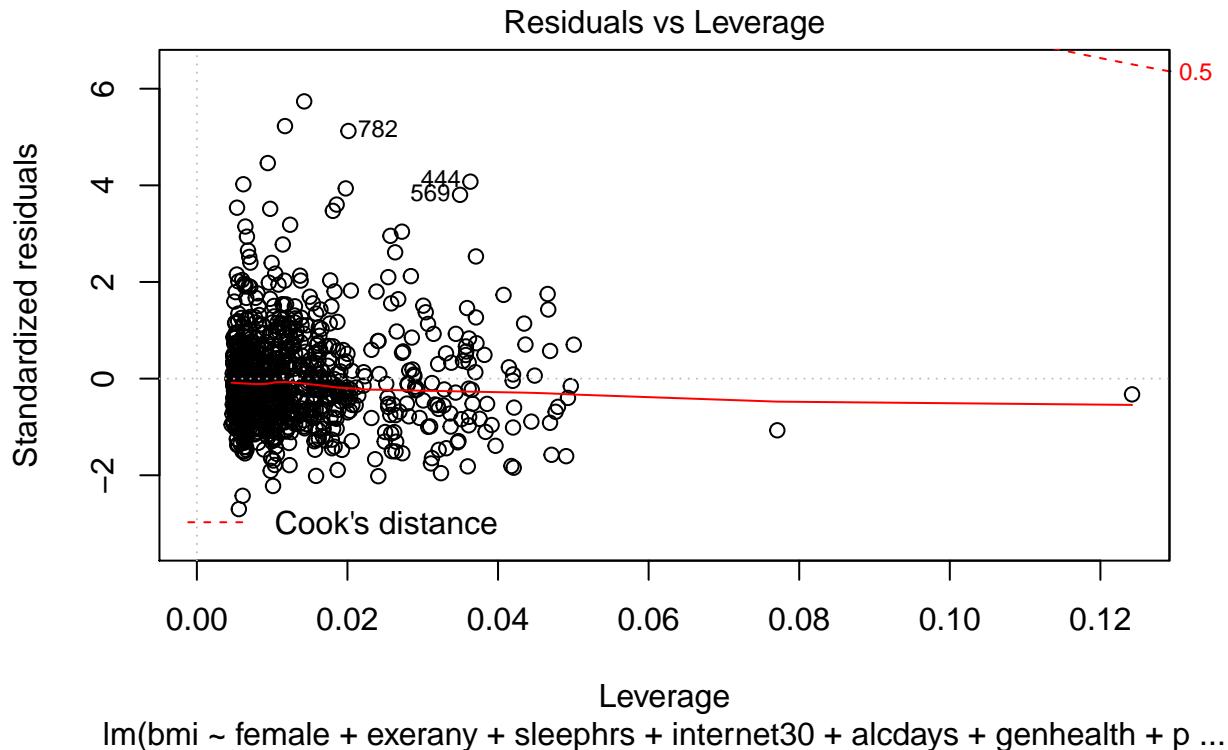
```
plot(c2_m7, which = 1)
```



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

2. What can we conclude from the plot below?

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



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

Chapter 3

Analysis of Variance

3.1 The bonding data: A Designed Dental Experiment

The `bonding` data describe a designed experiment into the properties of four different resin types (`resin` = A, B, C, D) and two different curing light sources (`light` = Halogen, LED) as they relate to the resulting bonding strength (measured in MPa¹) on the surface of teeth. The source is Kim (2014).

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

```
bonding
```

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

3.2 A One-Factor Analysis of Variance

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

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

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

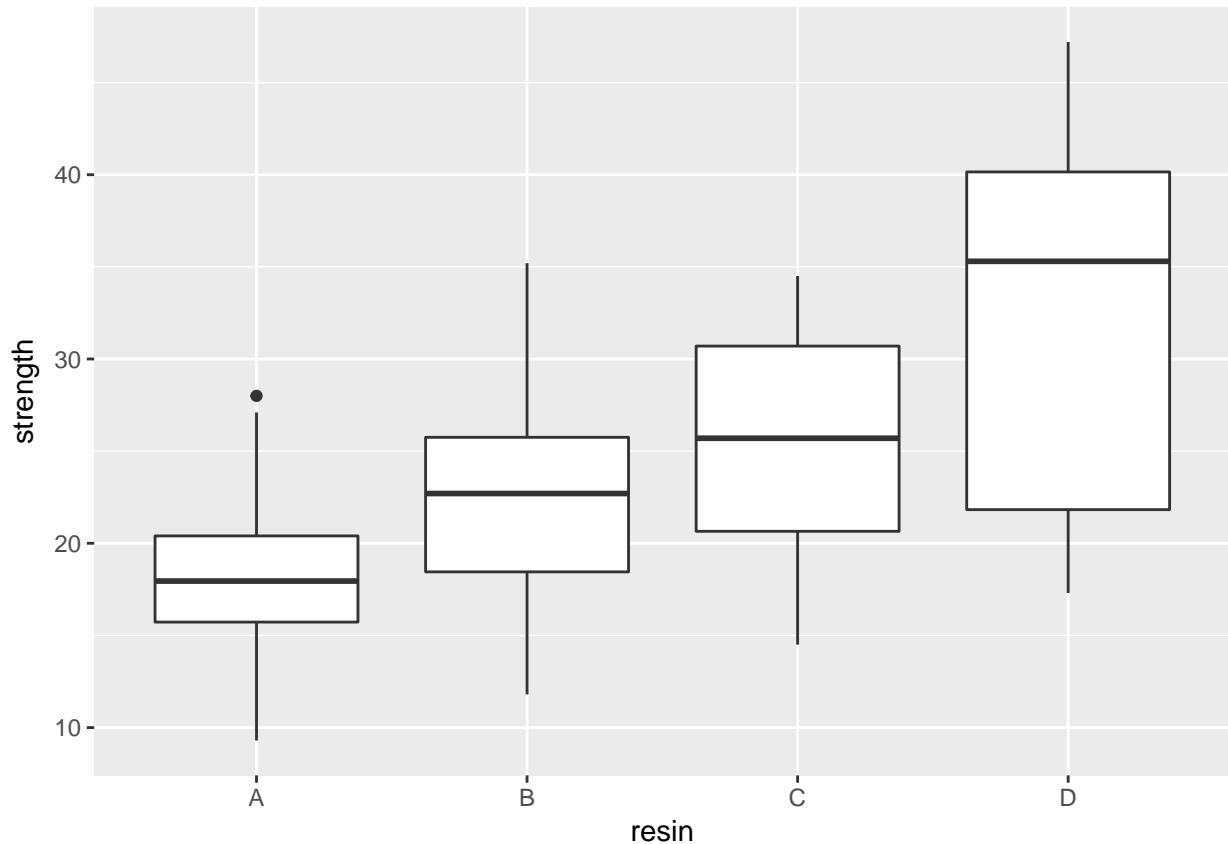
¹The MPa is defined as the failure load (in Newtons) divided by the entire bonded area, in mm².

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

I'd begin serious work with a plot.

3.2.1 Look at the Data!

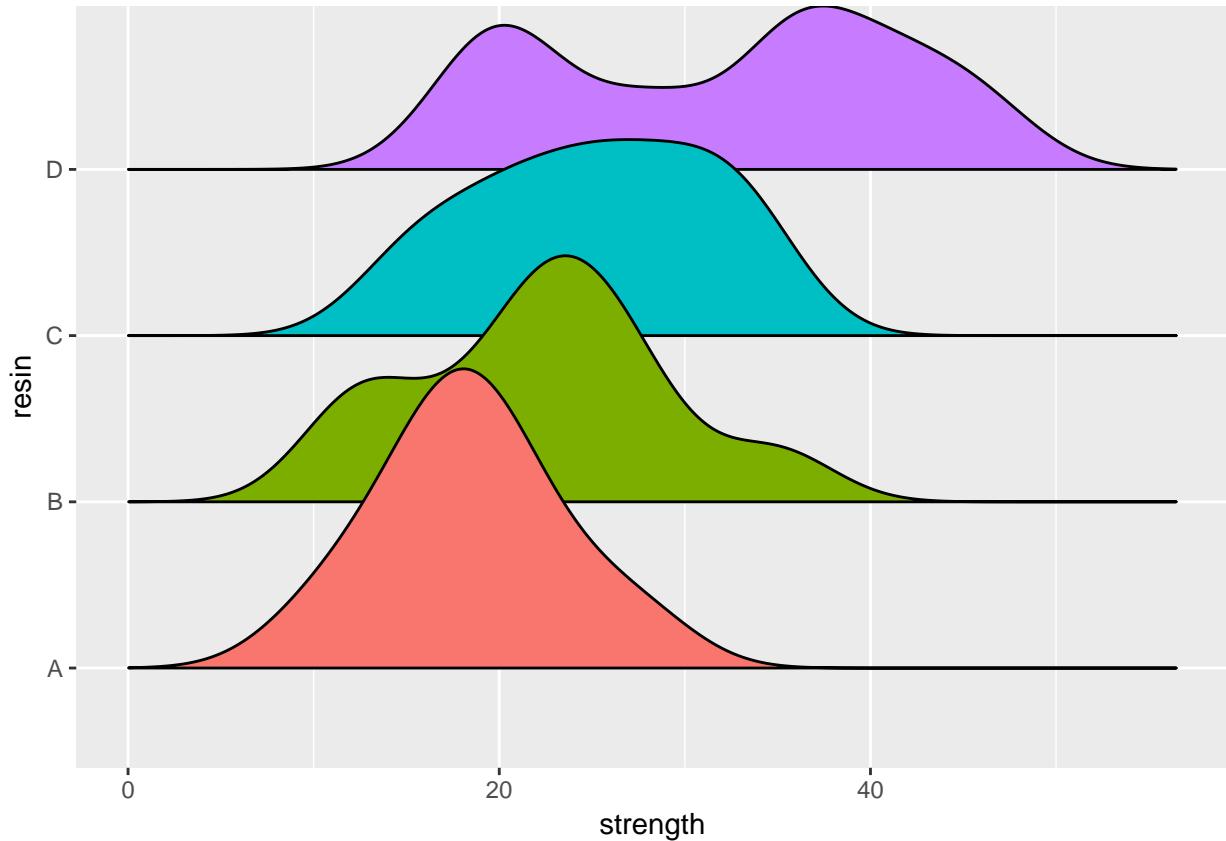
```
ggplot(bonding, aes(x = resin, y = strength)) +
  geom_boxplot()
```



Another good plot for this purpose is a ridgeline plot.

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

Picking joint bandwidth of 3.09



3.2.2 Table of Summary Statistics

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

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

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

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

Since the means and medians within each group are fairly close, and the distributions (with the possible exception of resin D) are reasonably well approximated by the Normal, I'll fit an ANOVA model².

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

Analysis of Variance Table

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

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

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

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

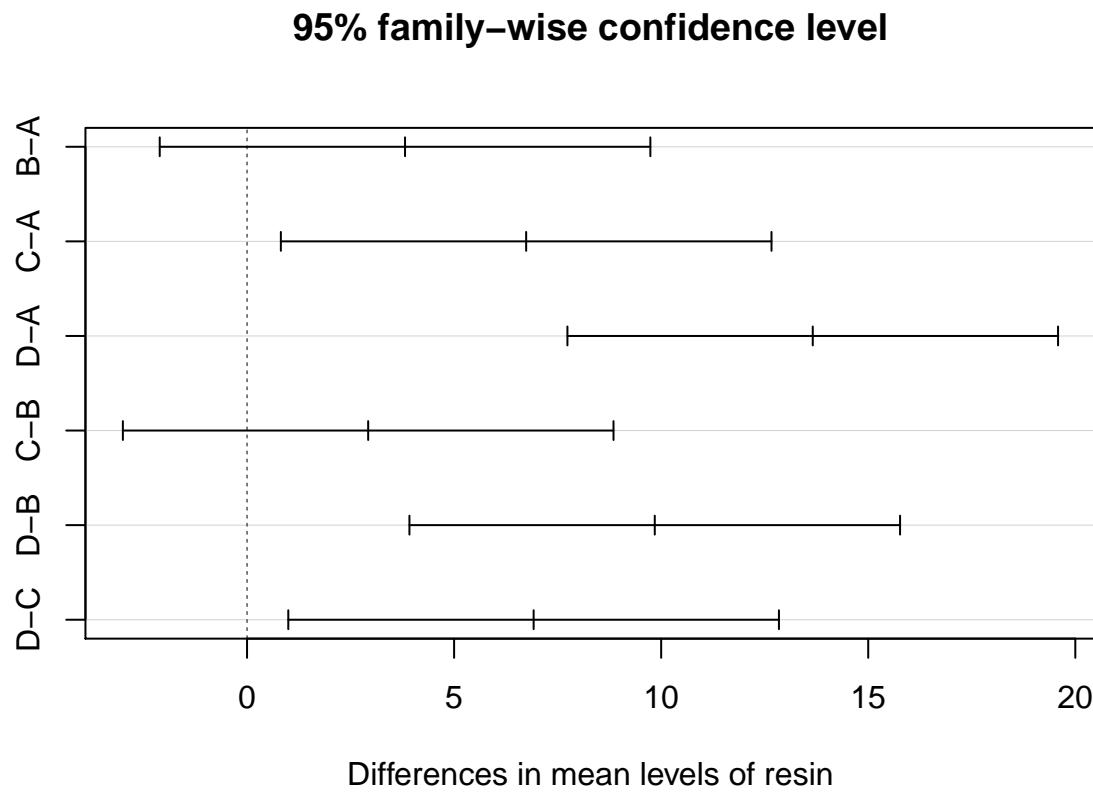
```
Fit: aov(formula = lm(strength ~ resin, data = bonding))
```

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

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

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

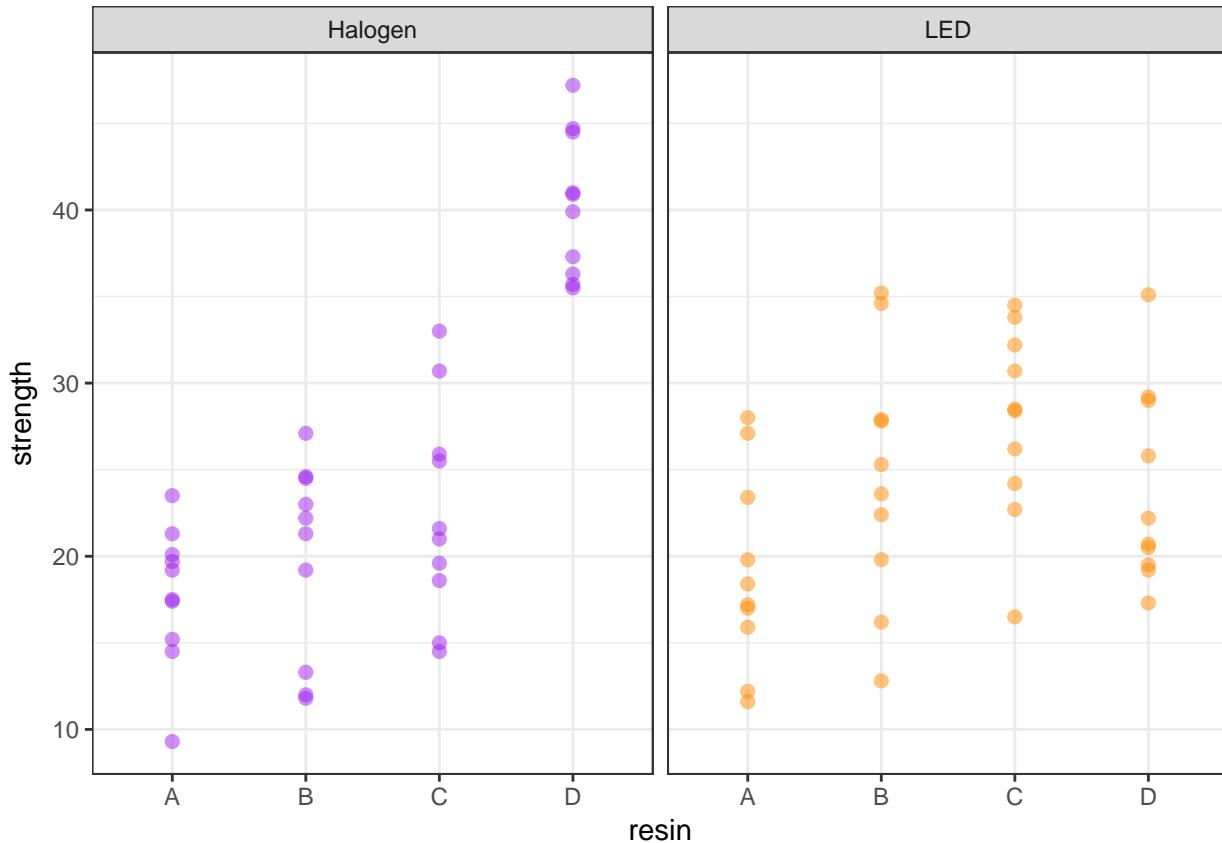
²If the data weren't approximately Normally distributed, we might instead consider a rank-based alternative to ANOVA, like the Kruskal-Wallis test.



3.3 A Two-Way ANOVA: Looking at Two Factors

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

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



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

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

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

bond.sum

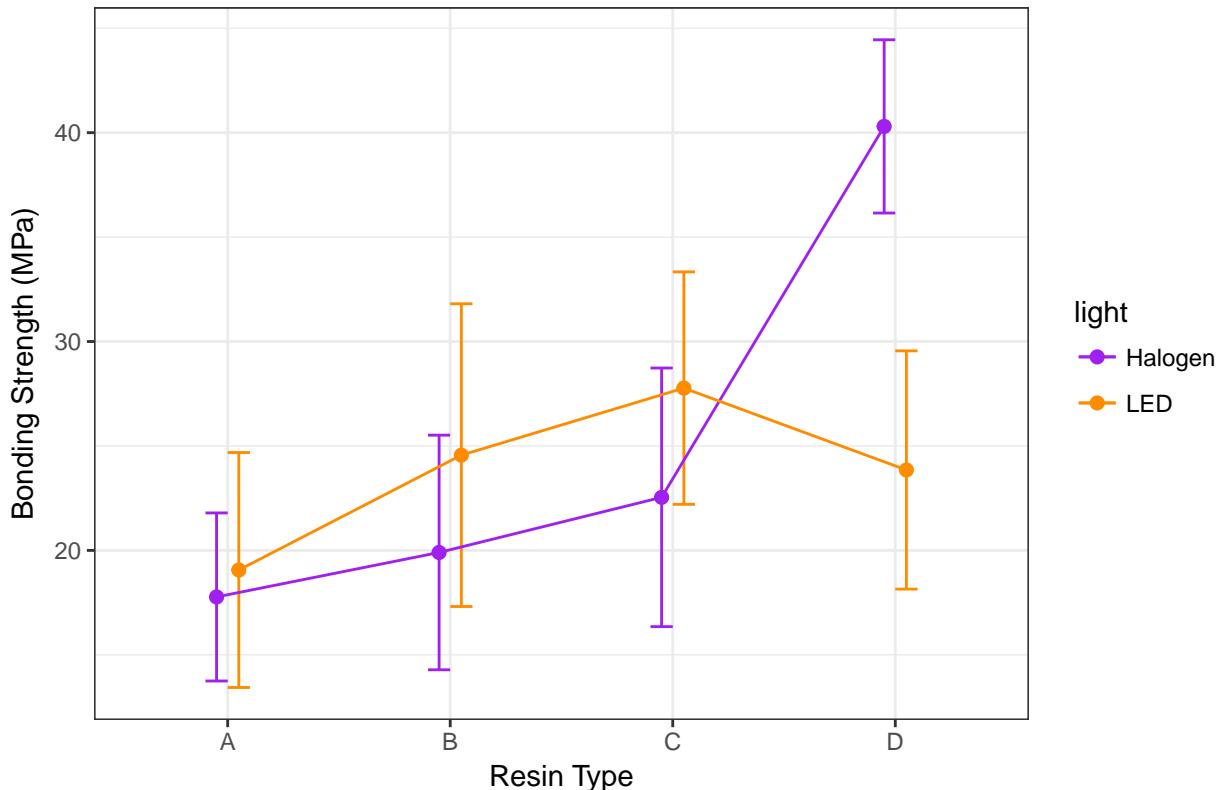
# A tibble: 8 x 4
# Groups:   resin [?]
  resin light   mean.str   sd.str
  <fct> <fct>     <dbl>    <dbl>
1 A     Halogen    17.8     4.02
2 A     LED         19.1     5.63
3 B     Halogen    19.9     5.62
4 B     LED         24.6     7.25
5 C     Halogen    22.5     6.19
6 C     LED         27.8  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

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

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

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

Observed Means (+/- SD) of Bonding Strength



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

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

3.4.1 Skimming the data after grouping by resin and light

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

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

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

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

3.5 Fitting the Two-Way ANOVA model with Interaction

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

summary(c3_m1)
```

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

Residuals:

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

Coefficients:

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

```

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

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

```

3.5.1 The ANOVA table for our model

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

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

```
anova(c3_m1)
```

Analysis of Variance Table

```

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

```

3.5.2 Is the interaction important?

In this case, the interaction:

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

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

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

3.5.3 Interpreting the Interaction

Recall the model equation, which is:

```
c3_m1
```

```

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

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

```

so we have:

$$strength = 17.77 + 2.13resinB + 4.77resinC + 22.53resinD + 1.29lightLED + 3.37resinB*lightLED + 3.94resinC*lightLED - 17.74lightLED*resinD$$

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.

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

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

```

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

summary(c3m1_aov)

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

```

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

```
TukeyHSD(c3m1_aov)
```

```

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

```

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

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

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

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

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

3.7 The bonding model without Interaction

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

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

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	19.074	1.787	10.676	< 2e-16 ***
resinB	3.815	2.260	1.688	0.09555 .
resinC	6.740	2.260	2.982	0.00386 **
resinD	13.660	2.260	6.044	5.39e-08 ***
lightLED	-1.317	1.598	-0.824	0.41229

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

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

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

$$\text{strength} = 19.07 + 3.82\text{resinB} + 6.74\text{resinC} + 13.66\text{resinD}$$

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

$$\text{strength} = 17.75 + 3.82\text{resinB} + 6.74\text{resinC} + 13.66\text{resinD}$$

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

```
anova(c3_m2)
```

Analysis of Variance Table

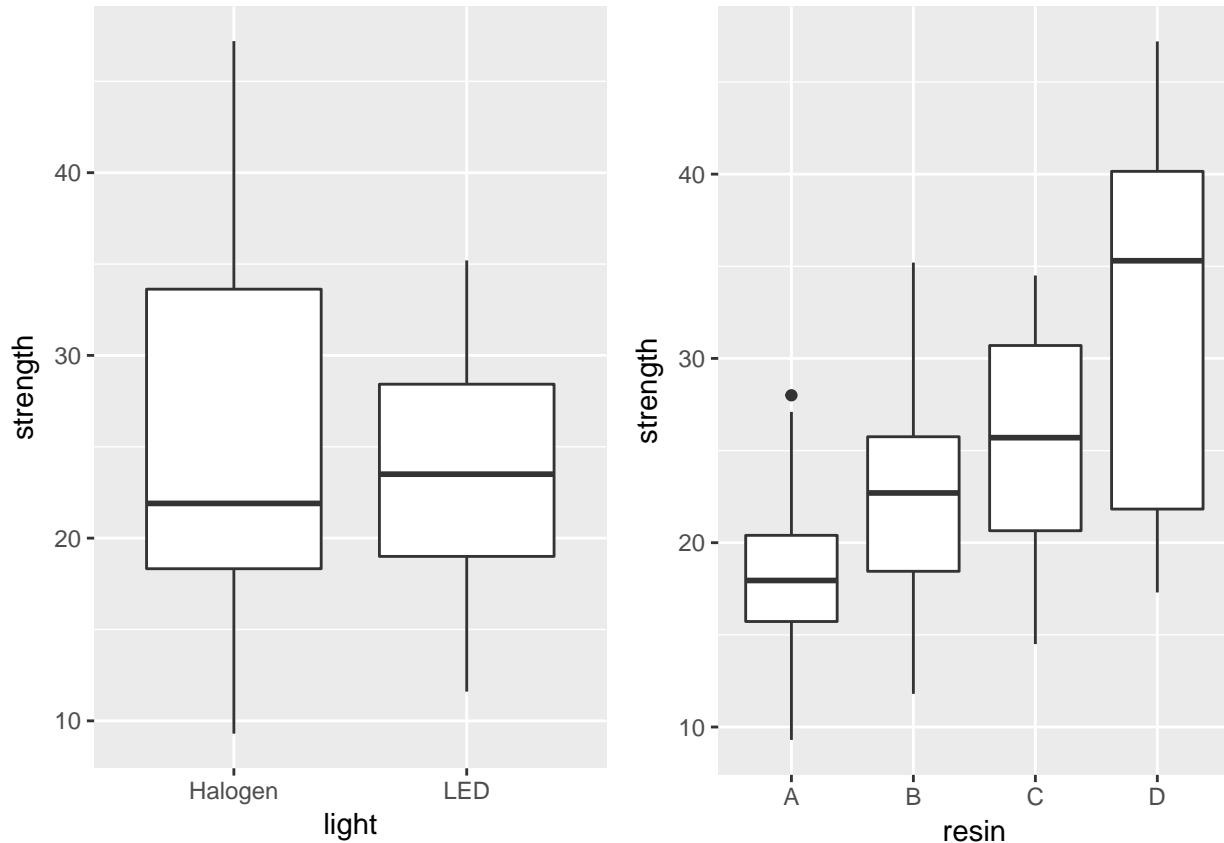
Response: strength	Df	Sum Sq	Mean Sq	F value	Pr(>F)
resin	3	1999.7	666.57	13.0514	6.036e-07 ***
light	1	34.7	34.72	0.6797	0.4123
Residuals	75	3830.5	51.07		

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

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

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

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



3.8 cortisol: A Hypothetical Clinical Trial

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

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

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

3.8.1 Codebook and Raw Data for cortisol

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

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

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

cortisol

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

3.9 Creating a factor combining sex and waist

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

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

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

summary(cortisol)
```

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

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

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

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

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

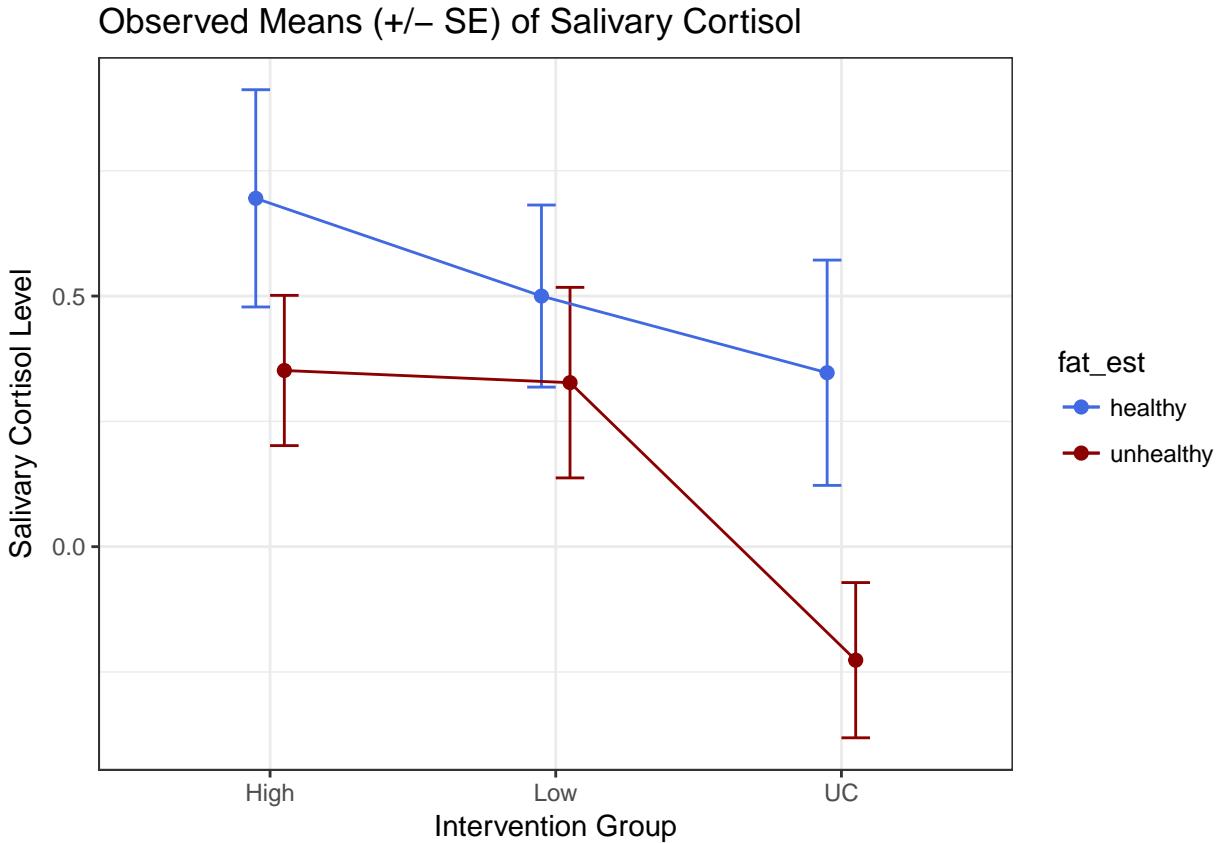
cort.sum
```

	interv	fat_est	mean.cort	se.cort
1	High	healthy	0.695	0.217
2	High	unhealthy	0.352	0.150
3	Low	healthy	0.500	0.182
4	Low	unhealthy	0.327	0.190
5	UC	healthy	0.347	0.225
6	UC	unhealthy	-0.226	0.155

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

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

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



3.11 A Two-Way ANOVA model for cortisol with Interaction

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

Analysis of Variance Table

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

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

```
summary(c3_m3)
```

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

Residuals:

```

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

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

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

```

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

3.12 A Two-Way ANOVA model for cortisol without Interaction

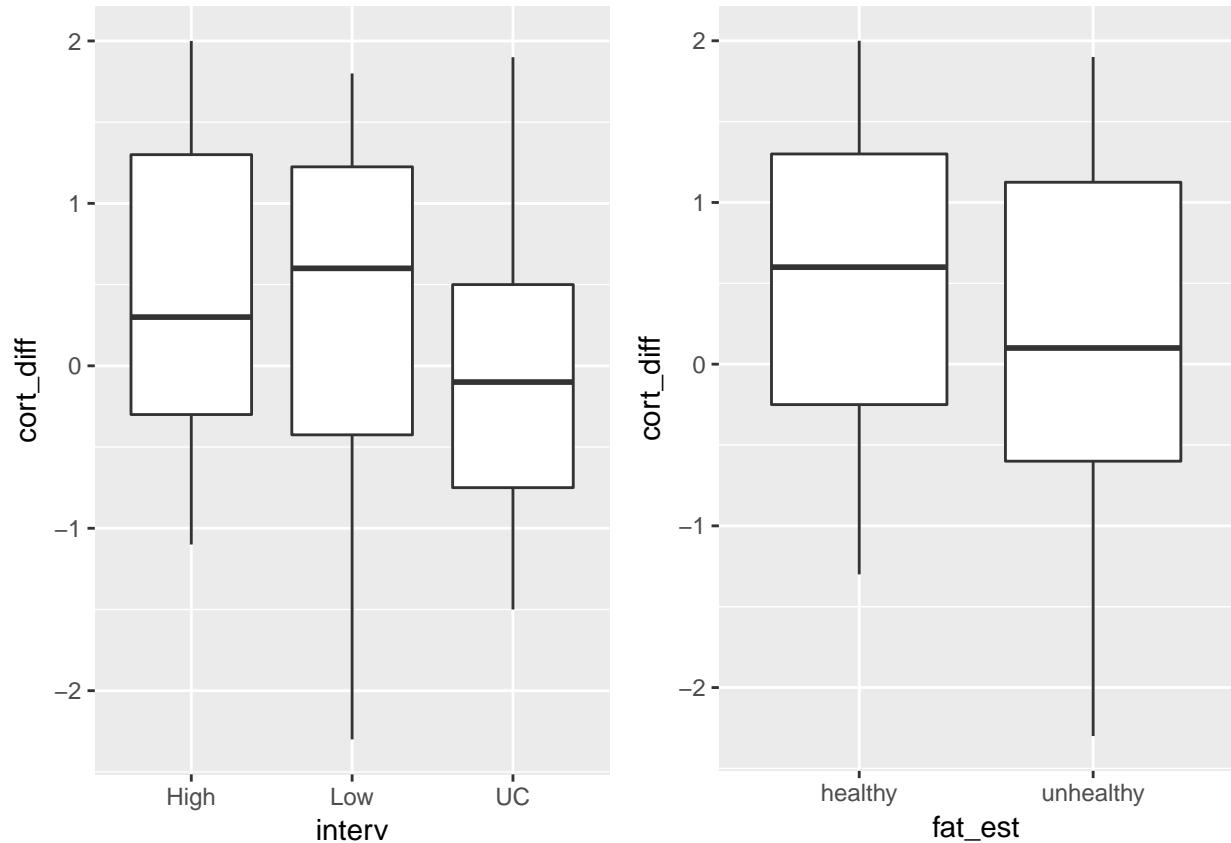
3.12.1 The Graph

```

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

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

```



3.12.2 The ANOVA Model

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

Analysis of Variance Table

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

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

3.12.3 The Regression Summary

```
summary(c3_m4)
```

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)							
(Intercept)	0.70452	0.16093	4.378	2.22e-05 ***							
intervLow	-0.08645	0.18232	-0.474	0.63606							
intervUC	-0.50063	0.18334	-2.731	0.00707 **							
fat_estunhealthy	-0.35878	0.15601	-2.300	0.02283 *							

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

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

3.12.4 Tukey HSD Comparisons

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

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

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

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

\$interv

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

\$fat_est

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

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

Chapter 4

Analysis of Covariance

4.1 An Emphysema Study

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

4.1.1 Codebook

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

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

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

emphysema

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

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

4.2 Does sex affect the mean change in theophylline?

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

Skim summary statistics

n obs: 16

n variables: 7

Variable type: numeric

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

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

Skim summary statistics

n obs: 16

n variables: 7

group variables: sex

Variable type: numeric

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

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

A one-way ANOVA model looks like this:

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

Analysis of Variance Table

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

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

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

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

Skim summary statistics

n obs: 16

n variables: 7

group variables: sex

Variable type: integer

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

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

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

4.4 Adding a quantitative covariate, `age`, to the model

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

4.4.1 The ANCOVA model

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

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

Call:

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-6.90266	7.92948	-0.871	0.4011
sexM	3.52466	1.75815	2.005	0.0681
age	0.05636	0.12343	0.457	0.6561

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

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

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

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

Note that the model yields $st_delta = -6.9 + 3.52 (\text{sex} = \text{male}) + 0.056 \text{ age}$, or

- $st_delta = -6.9 + 0.056 \text{ age}$ for female patients, and
- $st_delta = (-6.9 + 3.52) + 0.056 \text{ age} = -3.38 + 0.056 \text{ age}$ for male patients.

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

4.4.2 The ANCOVA Table

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

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

Analysis of Variance Table

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

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

4.5 Rerunning the ANCOVA model after simple imputation

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

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

```
emph_imp
```

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

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

More on simple imputation and missing data is coming soon.

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

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

Analysis of Variance Table

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

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

4.6 Looking at a factor-covariate interaction

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

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

Analysis of Variance Table

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

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

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

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

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

Our ANCOVA model for `st_delta` incorporating the `age` x `sex` product term is $-5.65 + 1.72 (\text{sex} = M) + 0.037 \text{age} + 0.029 (\text{sex} = M)(\text{age})$. 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 $-3.93 + 0.066 \text{age}$

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

4.7 Centering the Covariate to Facilitate ANCOVA Interpretation

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

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

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

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

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

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

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

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

Analysis of Variance Table

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

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

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

Comparing the two models, we have:

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

- (rescaled): $st_delta = -3.27 + 3.60 (\text{sex} = M) + 0.510 \text{ rescaled age_z} + 0.402 (\text{sex} = M) \times (\text{rescaled age_z})$

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

- $st_delta = -3.27 + 0.510 \text{ age_z}$ for female subjects, and
- $st_delta = (-3.27 + 3.60) + (0.510 + 0.402) \text{ age_z} = 0.33 + 0.912 \text{ age_z}$ for male subjects

Interpreting the centered, rescaled model, we have:

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

Chapter 5

Missing Data Mechanisms and Single Imputation

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

5.1 A Toy Example

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

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

```
# create some temporary variables

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

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

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

sbp_example

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

```

3    103 C      0.     150.     150.
4    104 A      1.      NA      120.
5    105 C     NA     155.     135.
6    106 A      1.      NA      115.
7    107 A      0.     135.     160.
8    108 <NA>    1.      NA      150.
9    109 B     NA     115.     130.
10   110 C      1.     170.     155.
11   111 A      0.     150.     140.
12   112 B      0.     145.     140.
13   113 C      1.     140.     150.
14   114 A      1.     160.     135.
15   115 B     NA     135.     120.

```

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

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

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

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

5.1.2 What is the pattern of missing data?

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

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

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

5.1.3 How can we identify the subjects with missing data?

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

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

5.2 Missing-data mechanisms

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

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

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

5.3 Options for Dealing with Missingness

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

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

5.4 Complete Case (and Available Case) analyses

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

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

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

5.5 Single Imputation

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

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

5.6 Multiple Imputation

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

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

5.7 Building a Complete Case Analysis

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

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

5.8 Single Imputation with the Mean or Mode

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

```
skim(sbp_example)
```

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

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

```

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

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

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

```

Here, suppose we decide to impute

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

```

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

si.1

```

```

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

```

We could accomplish the same thing with, for example:

```

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

```

5.9 Doing Single Imputation with simputation

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

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

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

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

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

si.3

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

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

Suppose we wanted to use:

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

```
set.seed(50001)

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

imp.4
```

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

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

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

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

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

Chapter 6

A Study of Prostate Cancer

6.1 Data Load and Background

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

```
prost
```

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

Note that a related `prost` data frame is also available as part of several R packages, including the `faraway` package, but there is an error in the `lweight` data for subject 32 in those presentations. The value of `lweight` for subject 32 should not be 6.1, corresponding to a prostate that is 449 grams in size, but instead the `lweight` value should be 3.804438, corresponding to a 44.9 gram prostate¹.

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

6.2 Code Book

Variable	Description
subject	subject number (1 to 97)

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

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

Notes:

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

6.3 Additions for Later Use

The code below adds to the `prost` tibble:

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

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

glimpse(prost)
```

Observations: 97

Variables: 16

\$ subject <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1...

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

6.4 Fitting and Evaluating a Two-Predictor Model

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

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

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

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2.33134	0.09128	25.540	< 2e-16 ***
lcavol_c	0.58640	0.08207	7.145	1.98e-10 ***
svi	0.60132	0.35833	1.678	0.0967 .
lcavol_c:svi	0.06479	0.26614	0.243	0.8082

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

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

6.4.1 Using tidy

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

```
tidy(c5_prost_A)
```

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

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

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

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

6.4.2 Interpretation

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

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

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

6.5 Exploring Model c5_prost_A

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

```
glance(c5_prost_A)
```

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

This summary includes, in order,

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

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

6.5.1 summary for Model c5_prost_A

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

```
summary(c5_prost_A)
```

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

Residuals:

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

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	2.33134	0.09128	25.540	< 2e-16 ***
lcavol_c	0.58640	0.08207	7.145	1.98e-10 ***
svi	0.60132	0.35833	1.678	0.0967 .
lcavol_c:svi	0.06479	0.26614	0.243	0.8082

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

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

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

6.5.2 Adjusted R²

R² is greedy.

- R² 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 R² so that we wind up with smaller models.
- The **adjusted R²** is often a useful way to compare multiple models for the same response.
 - $R_{adj}^2 = 1 - \frac{(1-R^2)(n-1)}{n-k}$, 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, $R_{adj}^2 = 1 - \frac{(1-0.5806)(97-1)}{97-4} = 0.5671$
 - The adjusted R² value is not, technically, a proportion of anything, but it is comparable across models for the same outcome.
 - The adjusted R² will always be less than the (unadjusted) R².

6.5.3 Coefficient Confidence Intervals

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

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

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

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

6.5.4 ANOVA for Model c5_prost_A

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

```
anova(c5_prost_A)
```

Analysis of Variance Table

Response: lpsa											
	Df	Sum Sq	Mean Sq	F value	Pr(>F)						
lcavol_c	1	69.003	69.003	119.6289	< 2.2e-16 ***						
svi	1	5.237	5.237	9.0801	0.003329 **						
lcavol_c:svi	1	0.034	0.034	0.0593	0.808191						
Residuals	93	53.643	0.577								

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

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

6.5.5 Residuals, Fitted Values and Standard Errors with augment

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

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

Skim summary statistics

n	obs:
97	
n	variables:
10	

Variable type: integer

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

Variable type: numeric

variable	missing	complete	n	mean	sd	p0	p25	median
.cooksdi	0	97	97	0.011	0.02	6.9e-06	0.00078	0.0035
.fitted	0	97	97	2.48	0.88	0.75	1.84	2.4

```

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

```

Elements shown here include:

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

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

6.5.6 Making Predictions with c5_prost_A

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

```

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

```

```

  fit      lwr      upr
1 2.331344 1.060462 3.602226
2 2.932664 1.545742 4.319586

```

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

```

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

```

```

  fit      lwr      upr

```

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

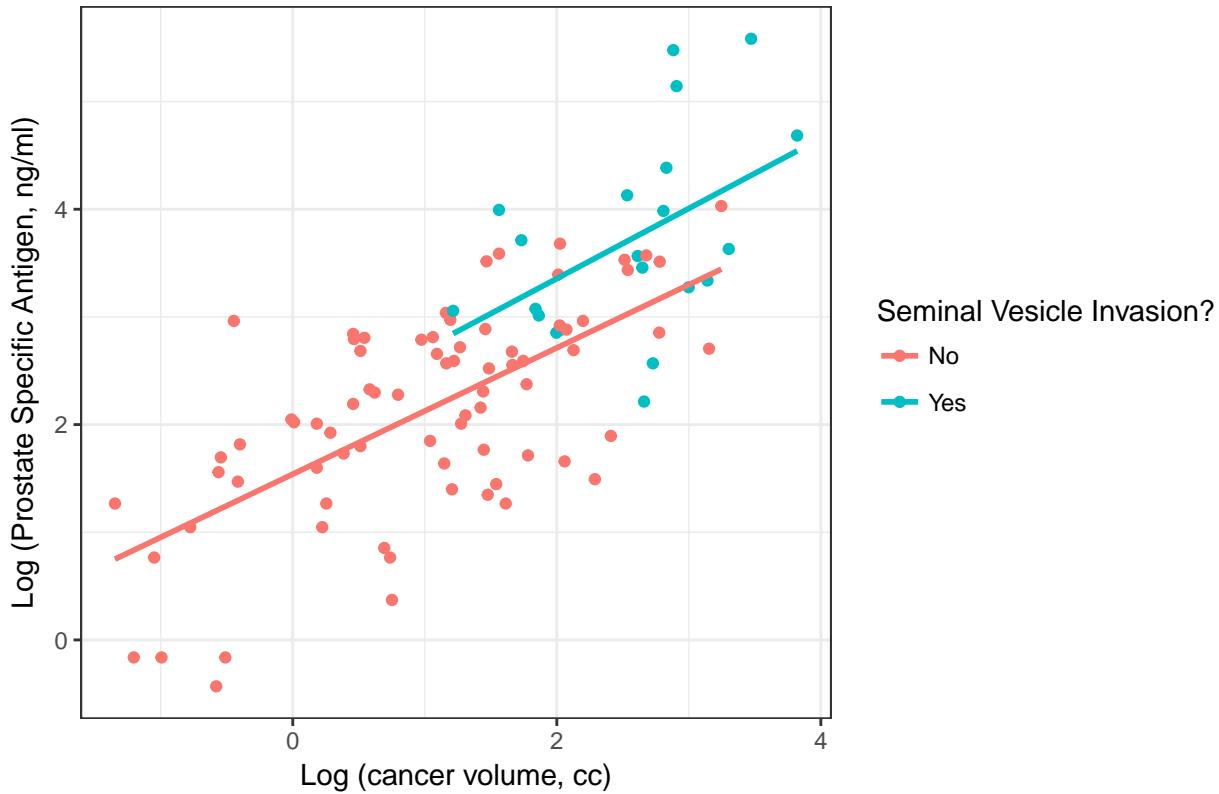
6.6 Plotting Model c5_prost_A

6.6.0.1 Plot logs conventionally

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

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

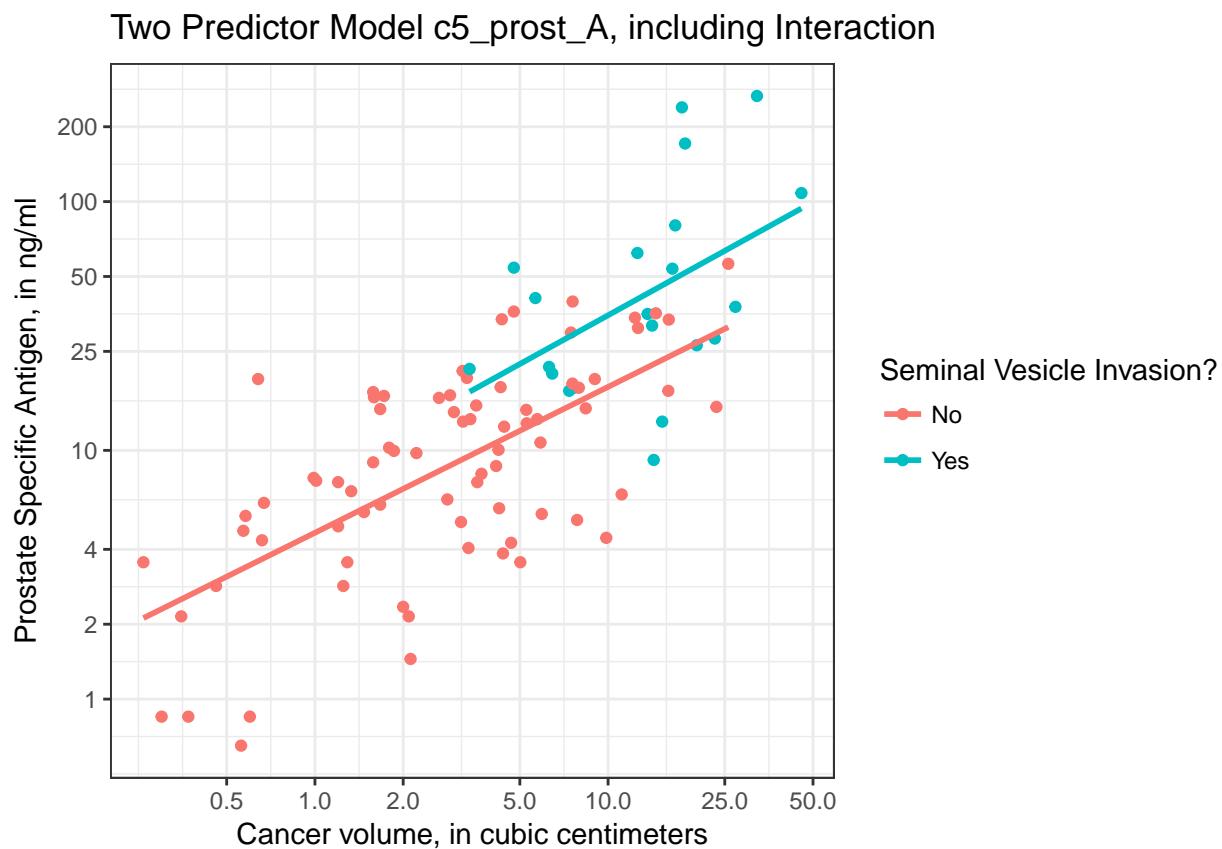
Two Predictor Model c5_prost_A, including Interaction



6.6.0.2 Plot on log-log scale

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

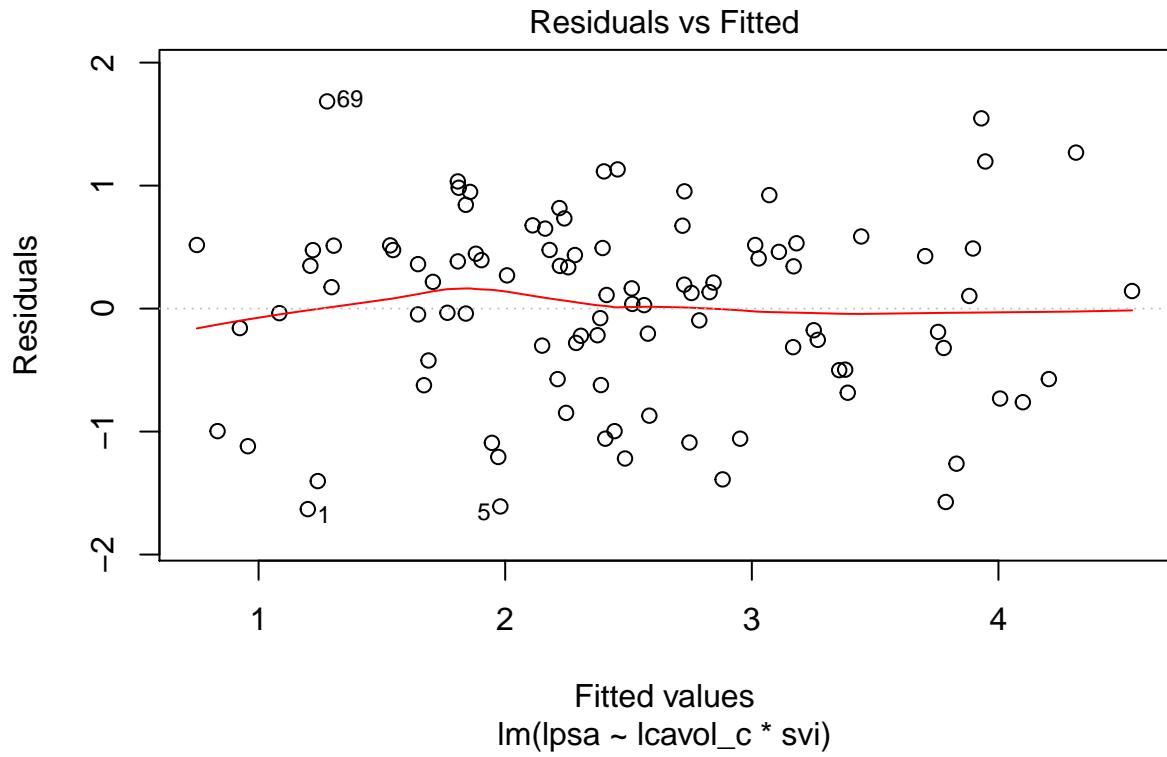
```
ggplot(prost, aes(x = cavol, y = psa, group = svi_f, color = svi_f)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  scale_color_discrete(name = "Seminal Vesicle Invasion?")
  scale_x_continuous(trans = "log",
                      breaks = c(0.5, 1, 2, 5, 10, 25, 50)) +
  scale_y_continuous(trans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

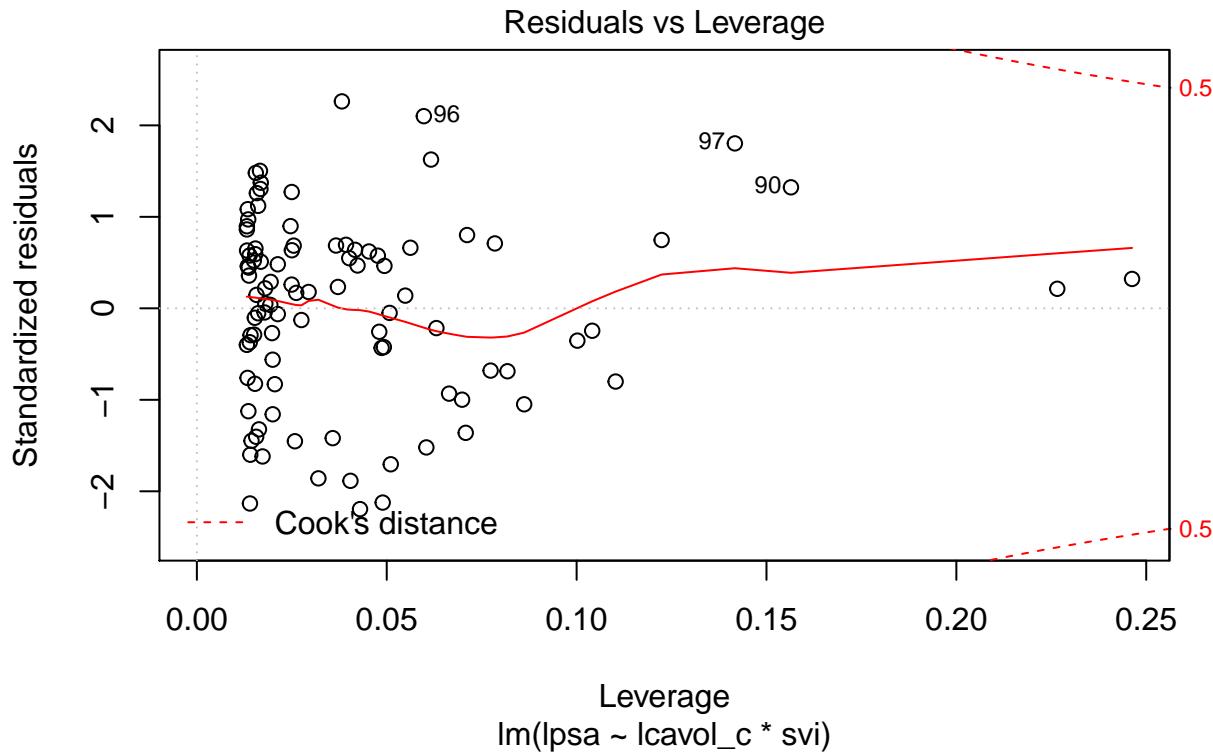


I've used the break point of 4 on the Y axis because of the old rule suggesting further testing for asymptomatic men with PSA of 4 or higher, but the other break points are arbitrary - they seemed to work for me, and used round numbers.

6.6.1 Residual Plots of c5_prost_A

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





6.7 Cross-Validation of Model c5_prost_A

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

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

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

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

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

Suppose that we want to perform what is called *10-crossfold separation*. In words, this approach splits the 97 observations in our `prost` data frame into 10 exclusive partitions of about 90% (so about 87-88 observations) into a training sample, and the remaining 10% (9-10 observations) in a test sample². We then refit a model of interest using the training data, and fit the resulting model on the test data using the `broom` package's `augment` function. This process is then repeated (a total of 10 times) so that each observation is used 9 times in the training sample, and once in the test sample.

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

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

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

```
set.seed(4320308)

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

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

head(prost_predictions)

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

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

6.7.1 Cross-Validated Summaries of Prediction Quality

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

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

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

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

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

	RMSE_intercept	MAE_intercept
1	1.15	0.891

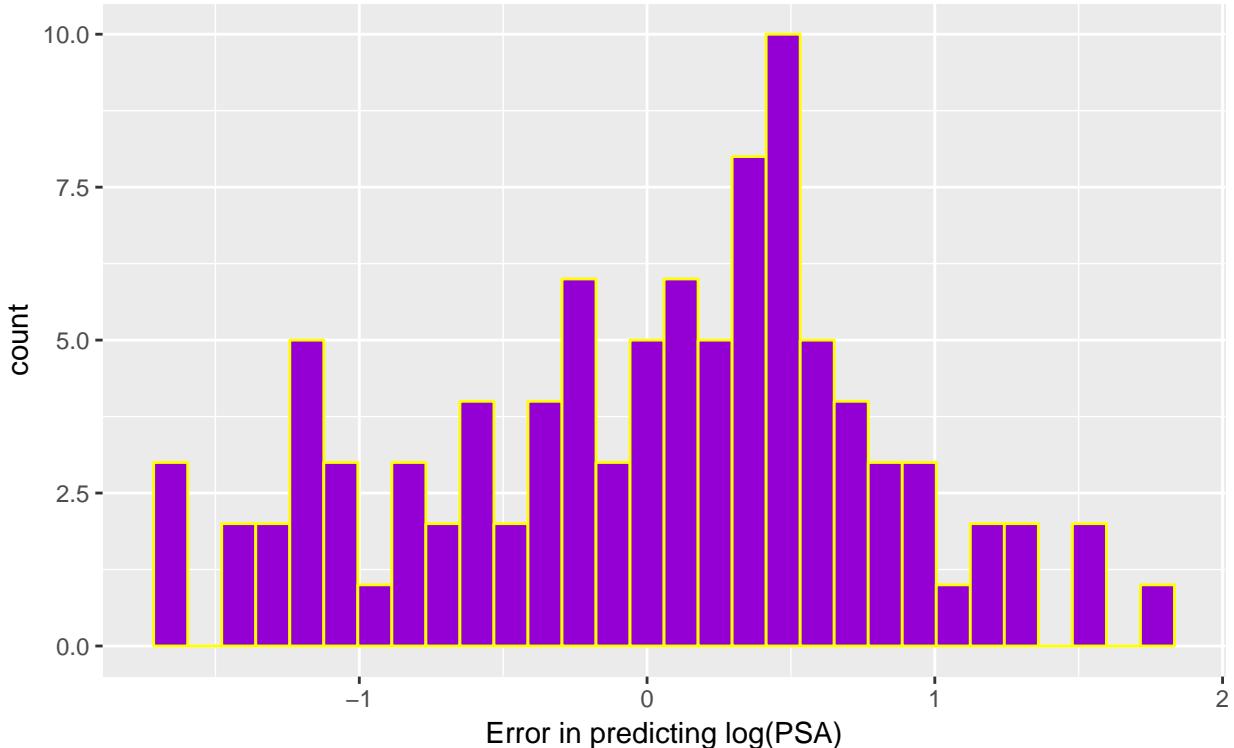
So our model looks meaningfully better than the “intercept only” model, in that both the RMSE and MAE are much lower (better) with our model.

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

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

Cross-Validated Errors in Prediction of log(PSA)

Using a model (`c5_prostA`) including lcavol_c and svi and their interaction



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

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

Skim summary statistics

```
n obs: 97
n variables: 16

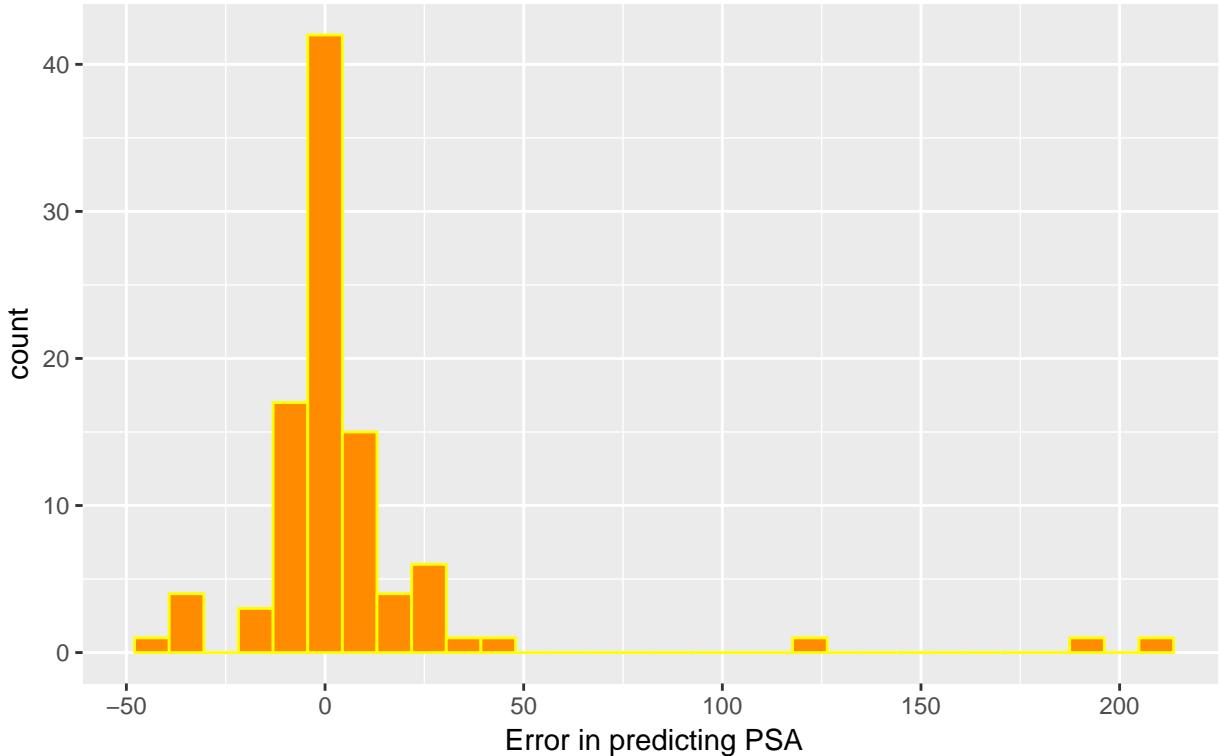
Variable type: numeric
variable missing complete n mean sd p0 p25 median p75 p100
lpsa      0     97 2.48 1.15 -0.43 1.73   2.59 3.06 5.58
```

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

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

Cross-Validated Errors in Prediction of PSA

Using a model (`c5_prostA`) including lcavol_c and svi and their interaction



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

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

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

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

```

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

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

Chapter 7

Stepwise Variable Selection

7.1 Strategy for Model Selection

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

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

The Two Key Aspects of Model Selection are:

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

7.1.1 How Do We Choose Potential Subsets of Predictors?

Choosing potential subsets of predictor variables usually involves either:

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

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

- the estimated coefficients, which are biased, and
- the associated tests and confidence intervals, which are overly optimistic.

Leeb and Potscher (2005) summarize the key issues:

1. Regardless of sample size, the model selection step typically has a dramatic effect on the sampling properties of the estimators that cannot be ignored. In particular, the sampling properties of post-model-selection estimators are typically significantly different from the nominal distributions that arise if a fixed model is supposed.

2. As a consequence, use of inference procedures that do not take into account the model selection step (e.g. using standard t-intervals as if the selected model has been given prior to the statistical analysis) can be highly misleading.

7.2 A “Kitchen Sink” Model (Model `c5_prost_ks`)

Suppose that we now consider a model for the `prost` data we have been working with, which includes main effects (and, in this case, only the main effects) of all eight candidate predictors for `lpsa`, as follows.

```
c5_prost_ks <- lm(lpsa ~ lcavol + lweight + age + bph_f + svi_f +
  lcp + gleason_f + pgg45, data = prost)
```

```
tidy(c5_prost_ks)
```

	term	estimate	std.error	statistic	p.value
1	(Intercept)	0.169937821	0.931332512	0.1824674	8.556454e-01
2	lcavol	0.544313829	0.087979210	6.1868461	2.010505e-08
3	lweight	0.702237531	0.203013089	3.4590751	8.455164e-04
4	age	-0.023857982	0.011081414	-2.1529727	3.412099e-02
5	bph_fMedium	0.364036274	0.182575941	1.9938896	4.933267e-02
6	bph_fHigh	0.248789989	0.195975792	1.2694935	2.076898e-01
7	svi_fYes	0.710949408	0.241990241	2.9379259	4.240326e-03
8	lcp	-0.119311781	0.089458946	-1.3337043	1.858223e-01
9	gleason_f7	0.220746268	0.343065609	0.6434520	5.216430e-01
10	gleason_f6	-0.053096704	0.430098039	-0.1234526	9.020368e-01
11	pgg45	0.003984574	0.004146495	0.9609499	3.392714e-01

```
glance(c5_prost_ks)
```

	r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik
1	0.6790343	0.6417127	0.6909479	18.19414	2.373796e-17	11	-95.93939
	AIC	BIC	deviance	df.residual			
1	215.8788	246.7753	41.05718	86			

We'll often refer to this (all predictors on board) approach as a “kitchen sink” model[This refers to the English idiom “... everything but the kitchen sink” which describes, essentially, everything imaginable. A “kitchen sink regression” is often used as a pejorative term, since no special skill or insight is required to identify it, given a list of potential predictors. For more, yes, there is a Wikipedia page.].

7.3 Sequential Variable Selection: Stepwise Approaches

- Forward Selection
 - We begin with a constant mean and then add potential predictors one at a time according to some criterion (R defaults to minimizing the Akaike Information Criterion) until no further addition significantly improves the fit.
 - Each categorical factor variable is represented in the regression model as a set of indicator variables. In the absence of a good reason to do something else, the set is added to the model as a single unit, and R does this automatically.
- Backwards Elimination
 - Start with the “kitchen sink” model and then delete potential predictors one at a time.
 - Backwards Elimination is less likely than Forward Selection to omit negatively confounded sets of variables, though all stepwise procedures have problems.
- Stepwise Regression can also be done by combining these methods.

7.3.1 The Big Problems with Stepwise Regression

There is no reason to assume that a single best model can be found.

- The use of forward selection, or backwards elimination, or stepwise regression including both procedures, will NOT always find the same model.
- It also appears to be essentially useless to try different stepwise methods to look for agreement.

Users of stepwise regression frequently place all of their attention on the particular explanatory variables included in the resulting model, when there's **no reason** (in most cases) to assume that model is in any way optimal.

Despite all of its problems, let's use stepwise regression to help predict `lpsa` given a subset of the eight predictors in `c5_prost_ks`.

7.4 Forward Selection with the `step` function

1. Specify the null model (intercept only)
2. Specify the variables R should consider as predictors (in the scope element of the `step` function)
3. Specify forward selection only
4. R defaults to using AIC as its stepwise criterion

```
with(prost,
  step(lm(lpsa ~ 1),
    scope=(~ lcavol + lweight + age + bph_f + svi_f +
           lcp + gleason_f + pgg45),
    direction="forward"))
```

Start: AIC=28.84

`lpsa` ~ 1

	Df	Sum of Sq	RSS	AIC
+ lcavol	1	69.003	58.915	-44.366
+ svi_f	1	41.011	86.907	-6.658
+ lcp	1	38.528	89.389	-3.926
+ gleason_f	2	30.121	97.796	6.793
+ lweight	1	24.019	103.899	10.665
+ pgg45	1	22.814	105.103	11.783
+ age	1	3.679	124.239	28.007
<none>			127.918	28.838
+ bph_f	2	4.681	123.237	29.221

Step: AIC=-44.37

`lpsa` ~ `lcavol`

	Df	Sum of Sq	RSS	AIC
+ lweight	1	7.1726	51.742	-54.958
+ svi_f	1	5.2375	53.677	-51.397
+ bph_f	2	3.2994	55.615	-45.956
+ pgg45	1	1.6980	57.217	-45.203
+ gleason_f	2	2.7834	56.131	-45.061
<none>			58.915	-44.366
+ lcp	1	0.6562	58.259	-43.452
+ age	1	0.0025	58.912	-42.370

```
Step: AIC=-54.96
lpsa ~ lcavol + lweight

          Df Sum of Sq    RSS    AIC
+ svi_f      1   5.1737 46.568 -63.177
+ pgg45      1   1.8158 49.926 -56.424
+ gleason_f   2   2.6770 49.065 -56.111
<none>                   51.742 -54.958
+ lcp         1   0.8187 50.923 -54.506
+ age         1   0.6456 51.097 -54.176
+ bph_f       2   1.4583 50.284 -53.731
```

```
Step: AIC=-63.18
lpsa ~ lcavol + lweight + svi_f
```

	Df	Sum of Sq	RSS	AIC
<none>		46.568	-63.177	
+ gleason_f	2	1.60467	44.964	-62.579
+ age	1	0.62301	45.945	-62.484
+ bph_f	2	1.50046	45.068	-62.354
+ pgg45	1	0.50069	46.068	-62.226
+ lcp	1	0.06937	46.499	-61.322

Call:
`lm(formula = lpsa ~ lcavol + lweight + svi_f)`

Coefficients:

(Intercept)	lcavol	lweight	svi_fYes
-0.7772	0.5259	0.6618	0.6657

The resulting model, arrived at after three forward selection steps, includes `lcavol`, `lweight` and `svi_f`.

```
model.fs <- lm(lpsa ~ lcavol + lweight + svi_f,
                 data=prost)
summary(model.fs)$adj.r.squared
```

```
[1] 0.6242063
extractAIC(model.fs)

[1] 4.00000 -63.17744
```

The adjusted R² value for this model is 0.624, and the AIC value used by the stepwise procedure is -63.18, on 4 effective degrees of freedom.

7.5 Backward Elimination using the `step` function

In this case, the backward elimination approach, using reduction in AIC for a criterion, comes to the same conclusion about the “best” model.

```
with(prost,
      step(lm(lpsa ~ lcavol + lweight + age + bph_f +
              svi_f + lcp + gleason_f + pgg45),
            direction="backward"))
```

Start: AIC=-61.4

```
lpsa ~ lcavol + lweight + age + bph_f + svi_f + lcp + gleason_f +
      pgg45
```

	Df	Sum of Sq	RSS	AIC
- gleason_f	2	1.1832	42.240	-62.639
- pgg45	1	0.4409	41.498	-62.359
- lcp	1	0.8492	41.906	-61.409
<none>		41.057	41.057	-61.395
- bph_f	2	2.0299	43.087	-60.714
- age	1	2.2129	43.270	-58.303
- svi_f	1	4.1207	45.178	-54.118
- lweight	1	5.7123	46.769	-50.760
- lcavol	1	18.2738	59.331	-27.683

Step: AIC=-62.64

```
lpsa ~ lcavol + lweight + age + bph_f + svi_f + lcp + pgg45
```

	Df	Sum of Sq	RSS	AIC
- lcp	1	0.8470	43.087	-62.713
<none>		42.240	42.240	-62.639
- pgg45	1	1.2029	43.443	-61.916
- bph_f	2	2.2515	44.492	-61.602
- age	1	2.0730	44.313	-59.992
- svi_f	1	4.6431	46.884	-54.523
- lweight	1	5.5988	47.839	-52.566
- lcavol	1	21.4956	63.736	-24.736

Step: AIC=-62.71

```
lpsa ~ lcavol + lweight + age + bph_f + svi_f + pgg45
```

	Df	Sum of Sq	RSS	AIC
- pgg45	1	0.5860	43.673	-63.403
<none>		43.087	43.087	-62.713
- bph_f	2	2.0214	45.109	-62.266
- age	1	1.7101	44.798	-60.938
- svi_f	1	3.7964	46.884	-56.523
- lweight	1	5.6462	48.734	-52.769
- lcavol	1	22.5152	65.603	-23.936

Step: AIC=-63.4

```
lpsa ~ lcavol + lweight + age + bph_f + svi_f
```

	Df	Sum of Sq	RSS	AIC
<none>		43.673	43.673	-63.403
- bph_f	2	2.2720	45.945	-62.484
- age	1	1.3945	45.068	-62.354
- svi_f	1	5.2747	48.948	-54.343
- lweight	1	5.3319	49.005	-54.230
- lcavol	1	25.5538	69.227	-20.720

Call:

```
lm(formula = lpsa ~ lcavol + lweight + age + bph_f + svi_f)
```

Coefficients:

	lcavol	lweight	age	bph_fMedium
(Intercept)	0.14329	0.54022	0.67283	-0.01819
bph_fHigh	svi_fYes			0.37607
	0.27216	0.68174		

The backwards elimination approach in this case lands on a model with five inputs (one of which includes two `bph` indicators,) eliminating only `gleason_f`, `pgg45` and `lcp`.

7.6 Allen-Cady Modified Backward Elimination

Ranking candidate predictors by importance in advance of backwards elimination can help avoid false-positives, while reducing model size. See Vittinghoff et al. (2012), Section 10.3 for more details.

1. First, force into the model any predictors of primary interest, and any confounders necessary for face validity of the final model.
 - “Some variables in the hypothesized causal model may be such well-established causal antecedents of the outcome that it makes sense to include them, essentially to establish the face validity of the model and without regard to the strength or statistical significance of their associations with the primary predictor and outcome ...”
2. Rank the remaining candidate predictors in order of importance.
3. Starting from an initial model with all candidate predictors included, delete predictors in order of ascending importance until the first variable meeting a criterion to stay in the model hits. Then stop.

Only the remaining variable hypothesized to be least important is eligible for removal at each step. When we are willing to do this sorting before collecting (or analyzing) the data, then we can do Allen-Cady backwards elimination using the `drop1` command in R.

7.6.1 Demonstration of the Allen-Cady approach

Suppose, for the moment that we decided to fit a model for the log of `psa` and we decided (before we saw the data) that we would:

`lcavol + lweight + svi_f + age + bph_f + gleason_f + lcp + pgg45`

- force the `gleason_f` variable to be in the model, due to prior information about its importance,
- and then rated the importance of the other variables as `lcavol` (most important), then `svi_f` then `age`, and then `bph_f`, then `lweight` and `lcp` followed by `pgg45` (least important)

When we are willing to do this sorting before collecting (or analyzing) the data, then we can do Allen-Cady backwards elimination using the `drop1` command in R.

Step 1. Fit the full model, then see if removing `pgg45` improves AIC...

```
with(prost, drop1(lm(lpsa ~ gleason_f + lcavol + svi_f +
  age + bph_f + lweight + lcp + pgg45),
  scope = (~ pgg45)))
```

Single term deletions

Model:

lpsa ~ gleason_f + lcavol + svi_f + age + bph_f + lweight + lcp +		
pgg45		
Df Sum of Sq	RSS	AIC
<none>	41.057	-61.395
pgg45 1 0.44085	41.498	-62.359

Since -62.3 is smaller (i.e. more negative) than -61.4, we delete `pgg45` and move on to assess whether we can remove the variable we deemed next least important (`lcp`)

Step 2. Let's see if removing `lcp` from this model improves AIC...

```
with(prost, drop1(lm(lpsa ~ gleason_f + lcavol + svi_f +
                      age + bph_f + lweight + lcp),
                     scope = (~ lcp)))
```

Single term deletions

Model:

lpsa ~ gleason_f + lcavol + svi_f + age + bph_f + lweight + lcp	Df	Sum of Sq	RSS	AIC
<none>		41.498	-62.359	
lcp	1	0.56767	42.066	-63.041

Again, since -63.0 is smaller than -62.4, we delete `lcp` and next assess whether we should delete `lweight`.

Step 3. Does removing `lweight` from this model improves AIC...

```
with(prost, drop1(lm(lpsa ~ gleason_f + lcavol + svi_f +
                      age + bph_f + lweight),
                     scope = (~ lweight)))
```

Single term deletions

Model:

lpsa ~ gleason_f + lcavol + svi_f + age + bph_f + lweight	Df	Sum of Sq	RSS	AIC
<none>		42.066	-63.041	
lweight	1	5.678	47.744	-52.760

Since the AIC for the model after the removal of `lweight` is larger (i.e. less negative), we stop, and declare our final model by the Allen-Cady approach to include `gleason_f`, `lcavol`, `svi_f`, `age`, `bph_f` and `lweight`.

7.7 Summarizing the Results

Method	Suggested Predictors
Forward selection	<code>lcavol</code> , <code>lweight</code> , <code>svi_f</code>
Backwards elimination	<code>lcavol</code> , <code>lweight</code> , <code>svi_f</code> , <code>age</code> , <code>bph_f</code>
Allen-Cady approach	<code>lcavol</code> , <code>lweight</code> , <code>svi_f</code> , <code>age</code> , <code>bph_f</code> , <code>gleason_f</code>

7.7.1 In-Sample Testing and Summaries

Since these models are nested in each other, let's look at the summary statistics (like R^2 , and AIC) and also run an ANOVA-based comparison of these nested models to each other and to the model with the intercept alone, and the kitchen sink model with all available predictors.

```
prost_m_int <- lm(lpsa ~ 1, data = prost)
prost_m_fw <- lm(lpsa ~ lcavol + lweight + svi_f, data = prost)
prost_m_bw <- lm(lpsa ~ lcavol + lweight + svi_f +
                  age + bph_f + gleason_f, data = prost)
prost_m_ac <- lm(lpsa ~ lcavol + lweight + svi_f +
                  age + bph_f + gleason_f + lcp, data = prost)
```

```
prost_m_ks <- lm(lpsa ~ lcavol + lweight + svi_f +
                  age + bph_f + gleason_f + lcp + pgg45, data = prost)
```

7.7.1.1 Model Fit Summaries (in-sample) from `glance`

Here are the models, at a `glance` from the `broom` package.

```
prost_sum <- bind_rows(glance(prost_m_int), glance(prost_m_fw),
                       glance(prost_m_bw), glance(prost_m_ac),
                       glance(prost_m_ks)) %>% round(., 3)
prost_sum$names <- c("intercept", "lcavol + lweight + svi",
                     "... + age + bhp + gleason", "... + lcp", "... + pgg45")
prost_sum <- prost_sum %>%
  select(names, r.squared, adj.r.squared, AIC, BIC, sigma, df, df.residual)

prost_sum
```

	names	r.squared	adj.r.squared	AIC	BIC	sigma
1	intercept	0.000	0.000	306.112	311.261	1.154
2	lcavol + lweight + svi	0.636	0.624	214.097	226.970	0.708
3	... + age + bhp + gleason	0.671	0.641	214.233	239.980	0.691
4	... + lcp	0.676	0.642	214.915	243.237	0.691
5	... + pgg45	0.679	0.642	215.879	246.775	0.691
	df	df.residual				
1	1	96				
2	4	93				
3	9	88				
4	10	87				
5	11	86				

From these summaries, it looks like:

- the adjusted R² is essentially indistinguishable between the three largest models, but a bit less strong with the three-predictor (4 df) model, and
- the AIC and BIC are (slightly) better (lower) with the three-predictor model (4 df) than any other.

So we might be motivated by these summaries to select any of the three models we're studying closely here.

7.7.1.2 Model Testing via ANOVA (in-sample)

To obtain ANOVA-based test results, we'll run...

```
anova(prost_m_int, prost_m_fw, prost_m_bw, prost_m_ac, prost_m_ks)
```

Analysis of Variance Table

```
Model 1: lpsa ~ 1
Model 2: lpsa ~ lcavol + lweight + svi_f
Model 3: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f
Model 4: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f + lcp
Model 5: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f + lcp +
         pgg45
Res.Df   RSS Df Sum of Sq      F Pr(>F)
1      96 127.918
2      93  46.568  3    81.349 56.7991 <2e-16 ***
```

```

3     88  42.066  5     4.503  1.8863  0.1050
4     87  41.498  1     0.568  1.1891  0.2786
5     86  41.057  1     0.441  0.9234  0.3393
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

What conclusions can we draw on the basis of these ANOVA tests?

- There is a statistically significant improvement in predictive value for Model 2 (the forward selection approach) as compared to Model 1 (the intercept only.)
- The ANOVA test comparing Model 5 (kitchen sink) to Model 4 (Allen-Cady result) shows no statistically significant improvement in predictive value.
- Neither does the ANOVA test comparing Model 3 to Model 2 or Model 4 to Model 3.

This suggests that, **if we are willing to let the ANOVA test decide our best model** than that would be the model produced by forward selection, with predictors `lcavol`, `lweight` and `svi_f`. But we haven't validated the models.

1. If the purpose of the model is to predict new data, some sort of out-of-sample or cross-validation approach will be necessary, and
2. Even if our goal isn't prediction but merely description of the current data, we would still want to build diagnostic plots to regression assumptions in each model, and
3. There is no reason to assume in advance that any of these models is in fact correct, or that any one of these stepwise approaches is necessarily better than any other, and
4. The mere act of running a stepwise regression model, as we'll see, can increase the bias in our findings if we accept the results at face value.

So we'll need some ways to validate the results once we complete the selection process.

7.7.2 Validating the Results of the Various Models

We can use a 5-fold cross-validation approach to assess the predictions made by our potential models and then compare them. Let's compare our three models:

- the three predictor model obtained by forward selection, including `lcavol`, `lweight`, and `svi_f`
- the five predictor model obtained by backwards elimination, including `lcavol`, `lweight`, `svi_f`, and also `age`, and `bph_f`
- the six predictor model obtained by the Allen-Cady approach, adding `gleason_f` to the previous model.

Here's the 5-fold validation work (and resulting RMSE and MAE estimates) for the three-predictor model.

```

set.seed(43201012)

prost3_models <- prost %>%
  crossv_kfold(k = 5) %>%
  mutate(model = map(train, ~ lm(lpsa ~ lcavol + lweight +
                                    svi_f, data = .)))

prost3_preds <- prost3_models %>%
  unnest(map2(model, test, ~ augment(.x, newdata = .y)))

prost3_preds %>%
  summarize(RMSE_prost3 = sqrt(mean((lpsa - .fitted)^2)),
            MAE_prost3 = mean(abs(lpsa - .fitted)))

# A tibble: 1 x 2
RMSE_prost3 MAE_prost3
<dbl>       <dbl>
1        1.1  0.45

```

```
1      0.745      0.587
```

Now, we'll generate the RMSE and MAE estimates for the five-predictor model.

```
set.seed(43206879)

prost5_models <- prost %>%
  crossv_kfold(k = 5) %>%
  mutate(model = map(train, ~ lm(lpsa ~ lcavol + lweight +
                                svi_f + age + bph_f, data = .)))

prost5_preds <- prost5_models %>%
  unnest(map2(model, test, ~ augment(.x, newdata = .y)))

prost5_preds %>%
  summarize(RMSE_prost5 = sqrt(mean((lpsa - .fitted) ^2)),
            MAE_prost5 = mean(abs(lpsa - .fitted)))

# A tibble: 1 x 2
RMSE_prost5 MAE_prost5
<dbl>       <dbl>
1      0.750      0.581
```

And at last, we'll generate the RMSE and MAE estimates for the six-predictor model.

```
set.seed(43236198)

prost6_models <- prost %>%
  crossv_kfold(k = 5) %>%
  mutate(model = map(train, ~ lm(lpsa ~ lcavol + lweight +
                                svi_f + age + bph_f + gleason_f, data = .)))

prost6_preds <- prost6_models %>%
  unnest(map2(model, test, ~ augment(.x, newdata = .y)))

prost6_preds %>%
  summarize(RMSE_prost6 = sqrt(mean((lpsa - .fitted) ^2)),
            MAE_prost6 = mean(abs(lpsa - .fitted)))

# A tibble: 1 x 2
RMSE_prost6 MAE_prost6
<dbl>       <dbl>
1      0.720      0.551
```

It appears that the six-predictor model does better than either of the other two approaches, with smaller RMSE and MAE. The three-predictor model does slightly better in terms of root mean square prediction error and slightly worse in terms of mean absolute prediction error than the five-predictor model.

OK. A mixed bag, with different conclusions depending on which summary we want to look at. But of course, stepwise regression isn't the only way to do variable selection. Let's consider a broader range of potential predictor sets.

Chapter 8

“Best Subsets” Variable Selection in our Prostate Cancer Study

A second approach to model selection involved fitting all possible subset models and identifying the ones that look best according to some meaningful criterion and ideally one that includes enough variables to model the response appropriately without including lots of redundant or unnecessary terms.

8.1 Four Key Summaries We’ll Use to Evaluate Potential Models

1. Adjusted R^2 , which we try to maximize.
2. Akaike’s Information Criterion (AIC), which we try to minimize, and a Bias-Corrected version of AIC due to Hurvich and Tsai (1989), which we use when the sample size is small, specifically when the sample size n and the number of predictors being studied k are such that $n/k \leq 40$. We also try to minimize this bias-corrected AIC.
3. Bayesian Information Criterion (BIC), which we also try to minimize.
4. Mallows’ C_p statistic, which we (essentially) try to minimize.

Choosing between AIC and BIC can be challenging.

For model selection purposes, there is no clear choice between AIC and BIC. Given a family of models, including the true model, the probability that BIC will select the correct model approaches one as the sample size n approaches infinity - thus BIC is asymptotically consistent, which AIC is not. [But, for practical purposes,] BIC often chooses models that are too simple [relative to AIC] because of its heavy penalty on complexity.

- Source: Hastie et al. (2001), page 208.

Several useful tools for running “all subsets” or “best subsets” regression comparisons are developed in R’s `leaps` package.

8.2 Using `regsubsets` in the `leaps` package

We can use the `leaps` package to obtain results in the `prost` study from looking at all possible subsets of the candidate predictors. The `leaps` package isn’t particularly friendly to the tidyverse. In particular, we **cannot have any character variables** in our predictor set. We specify our “kitchen sink” model, and apply the `regsubsets` function from `leaps`, which identifies the set of models.

To start, we'll ask R to find the one best subset (with 1 predictor variable [in addition to the intercept], then with 2 predictors, and then with each of 3, 4, ... 8 predictor variables) according to an exhaustive search without forcing any of the variables to be in or out.

- Use the `nvmax` command within the `regsubsets` function to limit the number of regression inputs to a maximum.
- Use the `nbest` command to identify how many subsets you want to identify for each predictor count.
- If all of your predictors are **quantitative** or **binary** then you can skip the `preds` step, and simply place your kitchen sink model into `regsubsets`.
- But if you have multi-categorical variables (like `gleason_f` or `svi_f` in our case) then you must create a `preds` group, as follows.

```
preds <- with(prost, cbind(lcavol, lweight, age, bph_f,
                           svi_f, lcp, gleason_f, pgg45))

rs.ks <- regsubsets(preds, y = prost$lpsa,
                     nvmax = 8, nbest = 1)
rs.summ <- summary(rs.ks)
rs.summ
```

```
Subset selection object
8 Variables  (and intercept)
      Forced in  Forced out
lcavol      FALSE      FALSE
lweight      FALSE      FALSE
age          FALSE      FALSE
bph_f        FALSE      FALSE
svi_f        FALSE      FALSE
lcp          FALSE      FALSE
gleason_f   FALSE      FALSE
pgg45        FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
      lcavol lweight age bph_f svi_f lcp gleason_f pgg45
1 ( 1 ) "*"   " "   " "   " "   " "   " "   " "
2 ( 1 ) "*"   "*"   " "   " "   " "   " "   " "
3 ( 1 ) "*"   "*"   " "   " "   "*"   " "   " "   "
4 ( 1 ) "*"   "*"   " "   " "   "*"   " "   " "   "
5 ( 1 ) "*"   "*"   "*"   "*"   "*"   " "   " "   "
6 ( 1 ) "*"   "*"   "*"   "*"   "*"   " "   "*"   "
7 ( 1 ) "*"   "*"   "*"   "*"   "*"   "*"   "*"   "
8 ( 1 ) "*"   "*"   "*"   "*"   "*"   "*"   "*"   "*"
```

So...

- the best one-predictor model used `lcavol`
- the best two-predictor model used `lcavol` and `lweight`
- the best three-predictor model used `lcavol`, `lweight` and `svi_f`
- the best four-predictor model added `bph_f`, and
- the best five-predictor model added `age`
- the best six-input model added `gleason_f`,
- the best seven-input model added `lcp`,
- and the eight-input model adds `pgg45`.

All of these “best subsets” are hierarchical, in that each model is a subset of the one below it. This isn't inevitably true.

- To determine which model is best, we can plot key summaries of model fit (adjusted R^2 , Mallows' C_p ,

bias-corrected AIC, and BIC) using either base R plotting techniques (what I've done in the past) or `ggplot2` (what I use now.) I'll show both types of plotting approaches in the next two sections.

8.2.1 Identifying the models with `which` and `outmat`

To see the models selected by the system, we use:

```
rs.summ$which
```

	(Intercept)	lcavol	lweight	age	bph_f	svi_f	lcp	gleason_f	pgg45
1	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
2	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
3	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
4	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
5	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
6	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE
7	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
8	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

Another version of this formatted for printing is:

```
rs.summ$outmat
```

	lcavol	lweight	age	bph_f	svi_f	lcp	gleason_f	pgg45
1	(1)	"*"	" "	" "	" "	" "	" "	" "
2	(1)	"*"	"*"	" "	" "	" "	" "	" "
3	(1)	"*"	"*"	" "	"*"	" "	" "	" "
4	(1)	"*"	"*"	" "	"*"	" "	" "	" "
5	(1)	"*"	"*"	"*"	"*"	" "	" "	" "
6	(1)	"*"	"*"	"*"	"*"	" "	"*"	" "
7	(1)	"*"	"*"	"*"	"*"	"*"	"*"	" "
8	(1)	"*"	"*"	"*"	"*"	"*"	"*"	"*"

We built one subset of each size up to eight predictors, and if we add the intercept term, this means we have models of size $k = 2, 3, 4, 5, 6, 7, 8$ and 9 .

The models are:

Size k	Predictors included (besides intercept)
2	lcavol
3	lcavol and lweight
4	add svi_f
5	add bph_f
6	add age
7	add gleason_f
8	add lcp
9	add pgg45

8.3 Calculating bias-corrected AIC

The bias-corrected AIC formula developed in Hurvich and Tsai (1989) requires three inputs:

- the residual sum of squares for a model
- the sample size (n) or number of observations used to fit the model
- the number of regression inputs, k , including the intercept, used in the model

So, for a particular model fit to n observations, on k predictors (including the intercept) and a residual sum of squares equal to RSS, we have:

$$AIC_c = n \log\left(\frac{RSS}{n}\right) + 2k + \frac{2k(k+1)}{n-k-1}$$

Note that the corrected AIC_c can be related to the original AIC via:

$$AIC_c = AIC + \frac{2k(k+1)}{n-k-1}$$

8.3.1 Calculation of aic.c in our setting

In our case, we have $n = 97$ observations, and built a series of models with $k = 2:9$ predictors (including the intercept in each case), so we will insert those values into the general formula for bias-corrected AIC which is:

```
aic.c <- n * log(rs.summ$rss / n) + 2 * k +
          (2 * k * (k + 1) / (n - k - 1))
```

We can obtain the residual sum of squares explained by each model by pulling `rss` from the `regsubsets` summary contained here in `rs.summ`.

```
data_frame(k = 2:9, RSS = rs.summ$rss)
```

```
# A tibble: 8 x 2
  k     RSS
  <int> <dbl>
1 2     58.9
2 3     51.7
3 4     46.6
4 5     45.7
5 6     44.6
6 7     43.7
7 8     43.0
8 9     42.8
```

In this case, we have:

```
rs.summ$aic.c <- 97*log(rs.summ$rss / 97) + 2*(2:9) +
          (2 * (2:9) * ((2:9)+1) / (97 - (2:9) - 1))
```

```
round(rs.summ$aic.c, 2) # bias-corrected
```

```
[1] -44.24 -54.70 -62.74 -62.29 -62.34 -62.11 -61.17 -59.36
```

The impact of this bias correction can be modest but important. Here’s a little table looking closely at the results in this problem. The uncorrected AIC are obtained using `extractAIC`, as described in the next section.

	Size	2	3	4	5	6	7	8	9
Bias-corrected AIC		-44.2	-54.7	-62.7	-62.3	-62.3	-62.1	-61.2	-59.4
Uncorrected AIC		-44.4	-55.0	-63.2	-62.4	-63.4	-63.0	-62.4	-61.4

8.3.2 The Uncorrected AIC provides no more useful information here

We could, if necessary, also calculate the *uncorrected* AIC value for each model, but we won't make any direct use of that, because that will not provide any new information not already gathered by the C_p statistic for a linear regression model. If you wanted to find the uncorrected AIC for a given model, you can use the `extractAIC` function.

```
extractAIC(lm(lpsa ~ lcavol, data = prost))
```

```
[1] 2.00000 -44.36603
```

```
extractAIC(lm(lpsa ~ lcavol + lweight, data = prost))
```

```
[1] 3.00000 -54.95846
```

Note that:

- these results are fairly comparable to the bias-corrected AIC we built above, and
- the `extractAIC` and `AIC` functions look like they give very different results, but they really don't.

```
AIC(lm(lpsa ~ lcavol, data = prost))
```

```
[1] 232.908
```

```
AIC(lm(lpsa ~ lcavol + lweight, data = prost))
```

```
[1] 222.3156
```

But notice that the differences in AIC are the same, either way, comparing these two models:

```
extractAIC(lm(lpsa ~ lcavol, data = prost)) - extractAIC(lm(lpsa ~ lcavol + lweight, data = prost))
```

```
[1] -1.00000 10.59243
```

```
AIC(lm(lpsa ~ lcavol, data = prost)) - AIC(lm(lpsa ~ lcavol + lweight, data = prost))
```

```
[1] 10.59243
```

- AIC is only defined up to an additive constant.
- Since the difference between two models using either `AIC` or `extractAIC` is the same, this doesn't actually matter which one we use, so long as we use the same one consistently.

8.3.3 Building a Tibble containing the necessary information

Again, note the use of 2:9 for the values of k , because we're fitting one model for each size from 2 through 9.

```
best_mods_1 <- data_frame(
```

```
  k = 2:9,
  r2 = rs.summ$rsq,
  adjr2 = rs.summ$adjr2,
  cp = rs.summ$cp,
  aic.c = rs.summ$aic.c,
  bic = rs.summ$bic
)
```

```
best_mods <- cbind(best_mods_1, rs.summ$which)
```

```
best_mods
```

	k	r2	adjr2	cp	aic.c	bic	(Intercept)	lcavol
1	2	0.5394320	0.5345839	28.213914	-44.23838	-66.05416	TRUE	TRUE

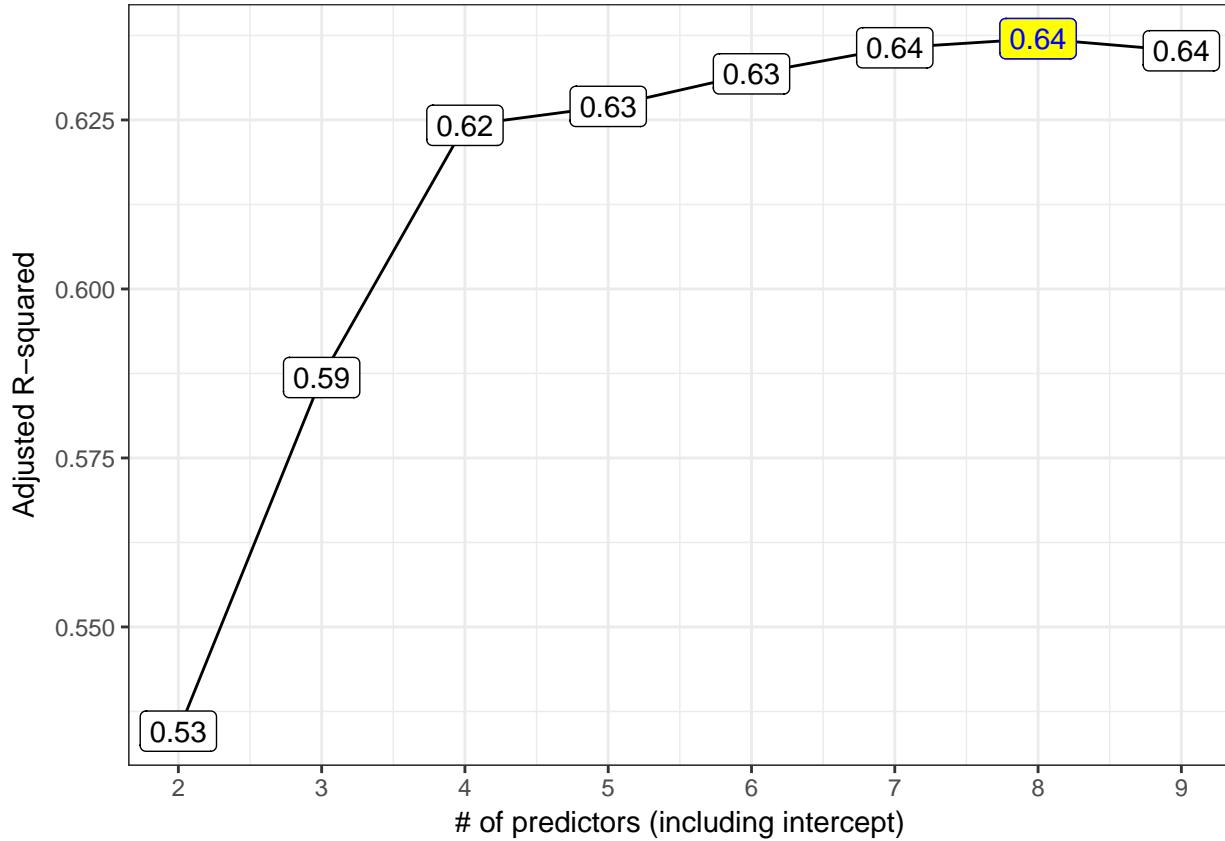
2	3	0.5955040	0.5868977	15.456669	-54.70040	-74.07188	TRUE	TRUE
3	4	0.6359499	0.6242063	6.811986	-62.74265	-79.71614	TRUE	TRUE
4	5	0.6425479	0.6270065	7.075509	-62.29223	-76.91557	TRUE	TRUE
5	6	0.6509970	0.6318211	6.851826	-62.33858	-74.66120	TRUE	TRUE
6	7	0.6584484	0.6356783	6.890739	-62.10692	-72.17992	TRUE	TRUE
7	8	0.6634967	0.6370302	7.562119	-61.17338	-69.04961	TRUE	TRUE
8	9	0.6656326	0.6352355	9.000000	-59.35841	-65.09253	TRUE	TRUE
		lweight	age	bph_f	svi_f	lcp	gleason_f	pgg45
1		FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
2		TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
3		TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
4		TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
5		TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE
6		TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE
7		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE
8		TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE

8.4 Plotting the Best Subsets Results using ggplot2

8.4.1 The Adjusted R² Plot

```
p1 <- ggplot(best_mods, aes(x = k, y = adjr2,
                             label = round(adjr2,2))) +
  geom_line() +
  geom_label() +
  geom_label(data = subset(best_mods,
                          adjr2 == max(adjr2)),
             aes(x = k, y = adjr2, label = round(adjr2,2)),
             fill = "yellow", col = "blue") +
  theme_bw() +
  scale_x_continuous(breaks = 2:9) +
  labs(x = "# of predictors (including intercept)",
       y = "Adjusted R-squared")
```

p1



Models 4-9 all look like reasonable choices here. The maximum adjusted R^2 is seen in the model of size 8.

8.4.2 Mallows' C_p

The C_p statistic focuses directly on the tradeoff between **bias** (due to excluding important predictors from the model) and extra **variance** (due to including too many unimportant predictors in the model.)

If N is the sample size, and we select p regression predictors from a set of K (where $p < K$), then the C_p statistic is

$$C_p = \frac{SSE_p}{MSE_K} - N + 2p$$

where:

- SSE_p is the sum of squares for error (residual) in the model with p predictors
- MSE_K is the residual mean square after regression in the model with all K predictors

As it turns out, this is just measuring the particular model's lack of fit, and then adding a penalty for the number of terms in the model (specifically $2p - N$ is the penalty since the lack of fit is measured as $(N - p)\frac{SSE_p}{MSE_K}$).

- If a model has no meaningful lack of fit (i.e. no substantial bias) then the expected value of C_p is roughly p .
- Otherwise, the expectation is p plus a positive bias term.
- In general, we want to see *smaller* values of C_p .
- We usually select a “winning model” by choosing a subset of predictors that have C_p near the value of p .

8.4.3 The C_p Plot

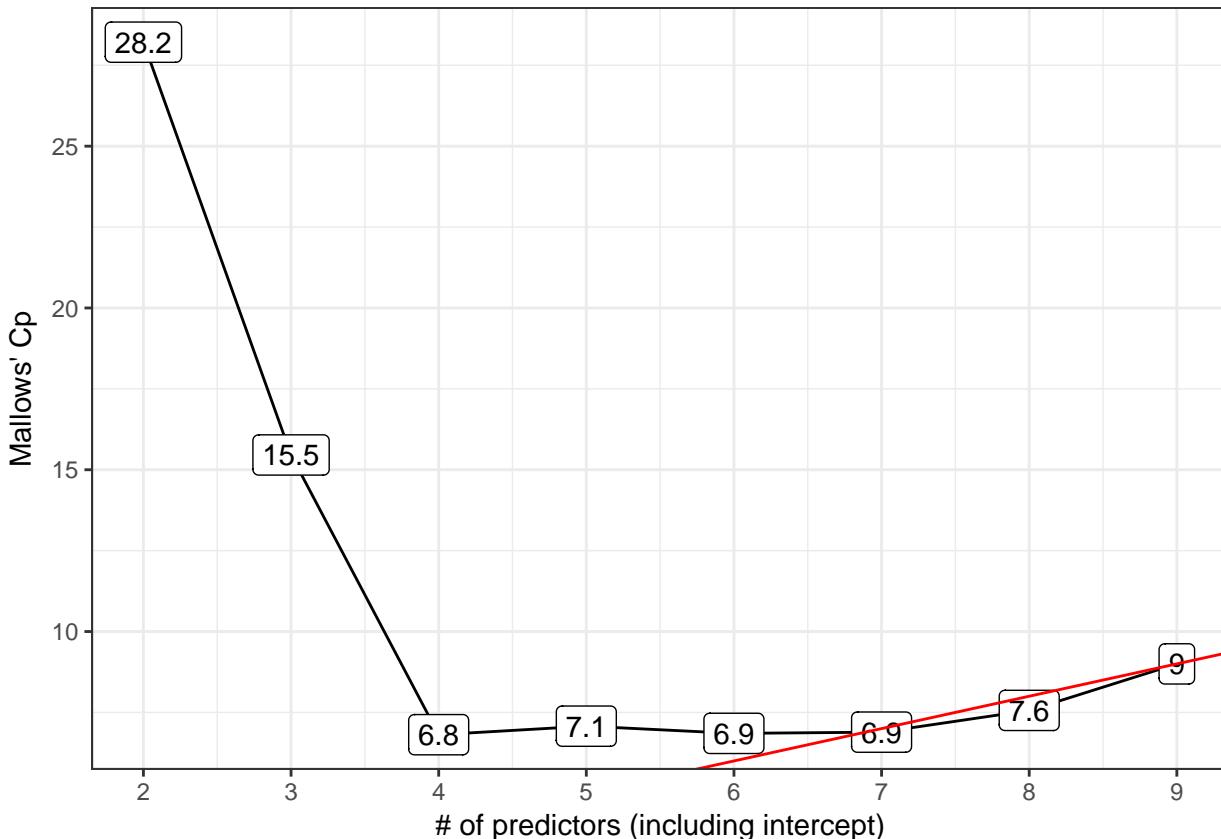
The C_p plot is just a scatterplot of C_p on the Y-axis, and the size of the model (coefficients plus intercept) $p = k$ on the X-axis.

Each of the various predictor subsets we will study is represented in a single point. A model without bias should have C_p roughly equal to p , so we'll frequently draw a line at $C_p = p$ to make that clear. We then select our model from among all models with small C_p statistics.

- My typical approach is to identify the models where $C_p - p \geq 0$, then select from among those models the model where $C_p - p$ is minimized, and if there is a tie, select the model where p is minimized.
- Another good candidate might be a slightly overfit model (where $C_p - p < 0$ but just barely.)

```
p2 <- ggplot(best_mods, aes(x = k, y = cp,
                             label = round(cp,1))) +
  geom_line() +
  geom_label() +
  geom_abline(intercept = 0, slope = 1,
              col = "red") +
  theme_bw() +
  scale_x_continuous(breaks = 2:9) +
  labs(x = "# of predictors (including intercept)",
       y = "Mallows' Cp")
```

p2



- Model 6 is a possibility here, with the difference $C_p - p$ minimized among all models with $C_p \geq p$.
- Model 7 also looks pretty good, with C_p just barely smaller than the size ($p = 7$) of the model.

8.4.4 “All Subsets” Regression and Information Criteria

We might consider any of three main information criteria:

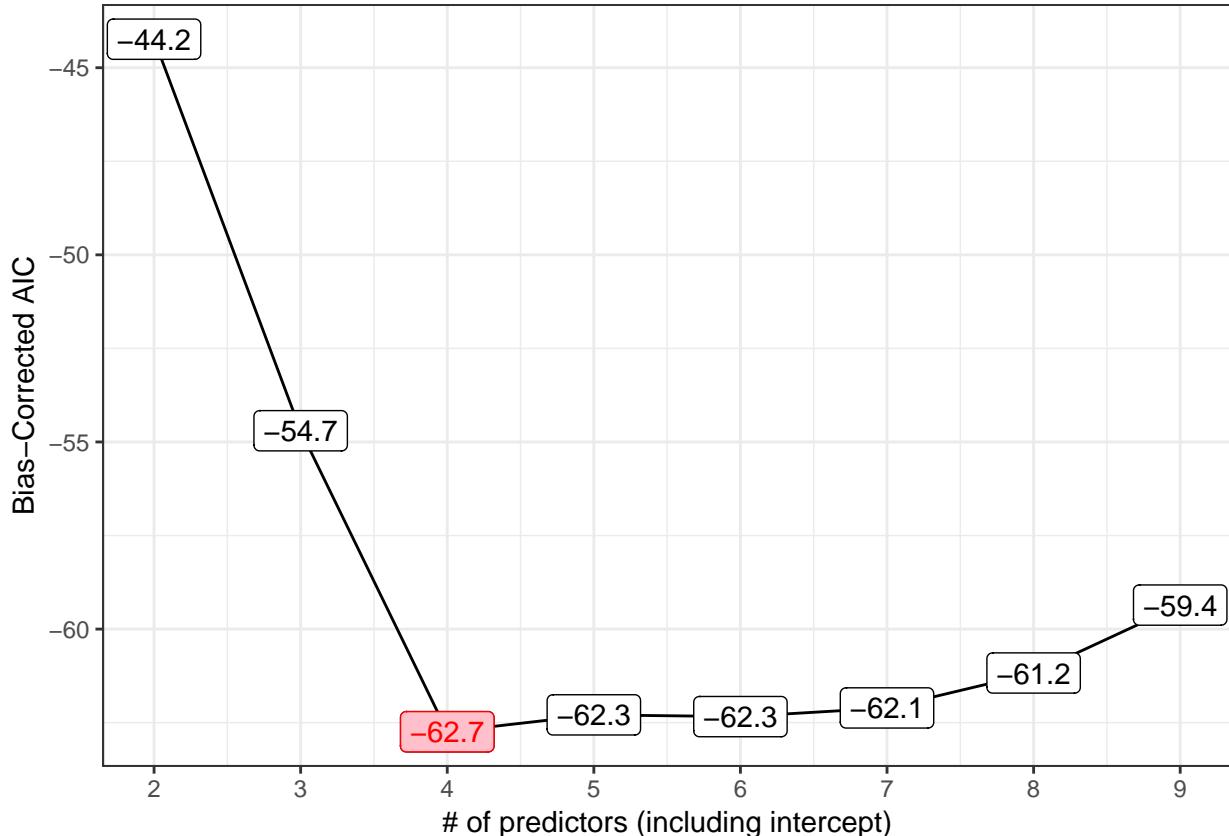
- the Bayesian Information Criterion, called BIC
- the Akaike Information Criterion (used by R’s default stepwise approaches,) called AIC
- a corrected version of AIC due to Hurvich and Tsai (1989), called AIC_c or aic.c

Each of these indicates better models by getting smaller. Since the C_p and AIC results will lead to the same model, I’ll focus on plotting the bias-corrected AIC and on BIC.

8.4.5 The bias-corrected AIC plot

```
p3 <- ggplot(best_mods, aes(x = k, y = aic.c,
                             label = round(aic.c,1))) +
  geom_line() +
  geom_label() +
  geom_label(data = subset(best_mods, aic.c == min(aic.c)),
             aes(x = k, y = aic.c), fill = "pink",
             col = "red") +
  theme_bw() +
  scale_x_continuous(breaks = 2:9) +
  labs(x = "# of predictors (including intercept)",
       y = "Bias-Corrected AIC")
```

p3

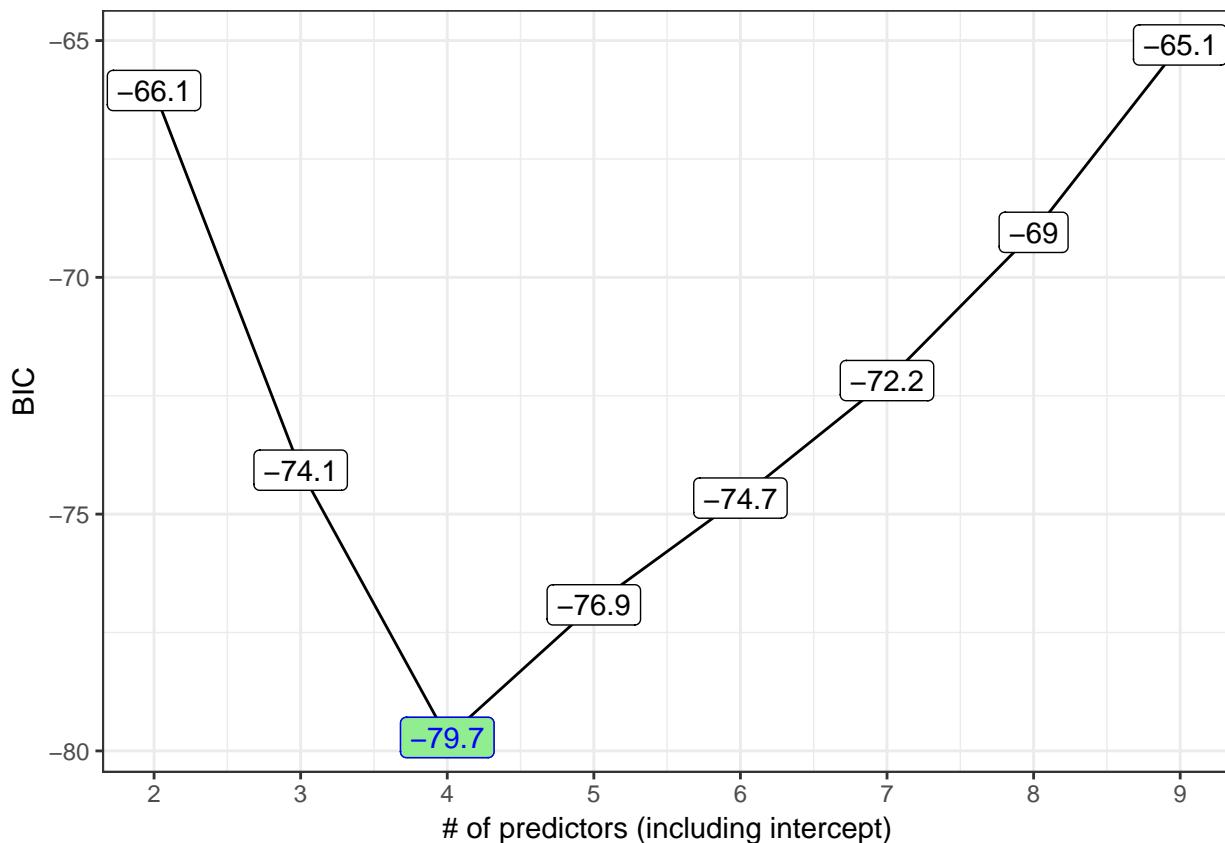


The smallest AIC_c values occur in models 4 and later, especially model 4 itself.

8.4.6 The BIC plot

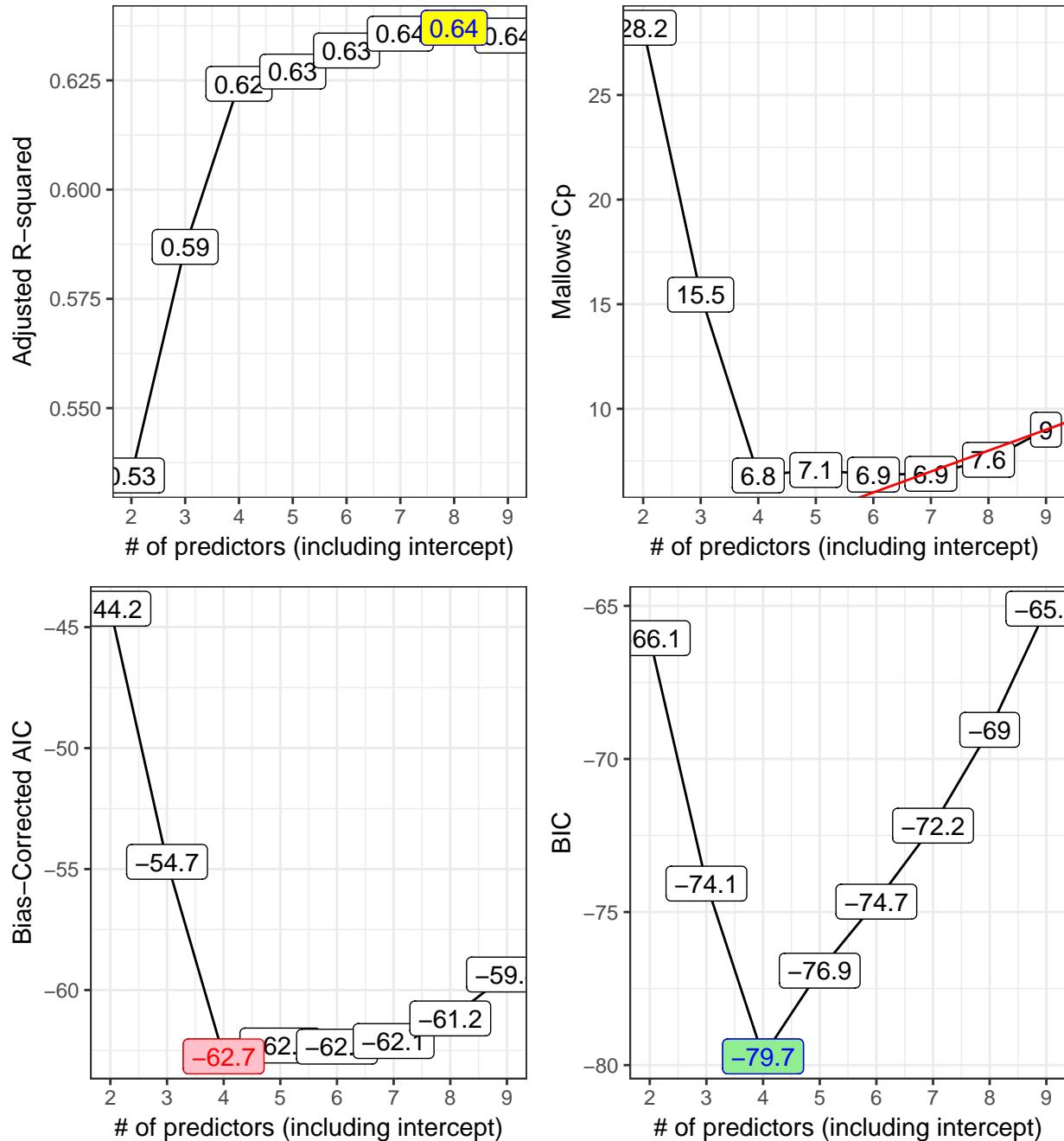
```
p4 <- ggplot(best_mods, aes(x = k, y = bic,
                             label = round(bic,1))) +
  geom_line() +
  geom_label() +
  geom_label(data = subset(best_mods, bic == min(bic)),
             aes(x = k, y = bic),
             fill = "lightgreen", col = "blue") +
  theme_bw() +
  scale_x_continuous(breaks = 2:9) +
  labs(x = "# of predictors (including intercept)",
       y = "BIC")
```

p4



8.4.7 All Four Plots in One Figure (via ggplot2)

```
gridExtra::grid.arrange(p1, p2, p3, p4, nrow = 2)
```



8.5 Table of Key Results

We can build a big table, like this:

```
best_mods
```

k	r2	adjr2	cp	aic.c	bic	(Intercept)	lcavol
1	2	0.5394320	0.5345839	28.213914	-44.23838	-66.05416	TRUE
2	3	0.5955040	0.5868977	15.456669	-54.70040	-74.07188	TRUE
3	4	0.6359499	0.6242063	6.811986	-62.74265	-79.71614	TRUE

```

4 5 0.6425479 0.6270065 7.075509 -62.29223 -76.91557      TRUE  TRUE
5 6 0.6509970 0.6318211 6.851826 -62.33858 -74.66120      TRUE  TRUE
6 7 0.6584484 0.6356783 6.890739 -62.10692 -72.17992      TRUE  TRUE
7 8 0.6634967 0.6370302 7.562119 -61.17338 -69.04961      TRUE  TRUE
8 9 0.6656326 0.6352355 9.000000 -59.35841 -65.09253      TRUE  TRUE
lweight    age bph_f svi_f    lcp gleason_f pgg45
1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE
2 TRUE FALSE FALSE FALSE FALSE FALSE FALSE
3 TRUE FALSE FALSE TRUE FALSE FALSE FALSE
4 TRUE FALSE TRUE TRUE FALSE FALSE FALSE
5 TRUE TRUE TRUE TRUE FALSE FALSE FALSE
6 TRUE TRUE TRUE TRUE FALSE TRUE FALSE
7 TRUE TRUE TRUE TRUE TRUE TRUE FALSE
8 TRUE TRUE TRUE TRUE TRUE TRUE TRUE

```

8.6 Models Worth Considering?

<i>k</i>	Predictors	Reason
4	lcavol lweight svi_f	minimizes BIC, AIC _c
7	+ age bph_f gleason_f	C_p near p
8	+ lcp	max R_{adj}^2

8.7 Compare these candidate models in-sample?

8.7.1 Using `anova` to compare nested models

Let's run an ANOVA-based comparison of these nested models to each other and to the model with the intercept alone.

- The models are **nested** because `m04` is a subset of the predictors in `m07`, which includes a subset of the predictors in `m08`.

```

m.int <- lm(lpsa ~ 1, data = prost)
m04 <- lm(lpsa ~ lcavol + lweight + svi_f, data = prost)
m07 <- lm(lpsa ~ lcavol + lweight + svi_f +
           age + bph_f + gleason_f, data = prost)
m08 <- lm(lpsa ~ lcavol + lweight + svi_f +
           age + bph_f + gleason_f + lcp, data = prost)
m.full <- lm(lpsa ~ lcavol + lweight + svi_f +
              age + bph_f + gleason_f + lcp + pgg45, data = prost)

```

Next, we'll run...

```
anova(m.full, m08, m07, m04, m.int)
```

Analysis of Variance Table

```

Model 1: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f + lcp +
          pgg45
Model 2: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f + lcp
Model 3: lpsa ~ lcavol + lweight + svi_f + age + bph_f + gleason_f
Model 4: lpsa ~ lcavol + lweight + svi_f

```

```
Model 5: lpsa ~ 1
Res.Df   RSS Df Sum of Sq    F Pr(>F)
1     86 41.057
2     87 41.498 -1   -0.441  0.9234 0.3393
3     88 42.066 -1   -0.568  1.1891 0.2786
4     93 46.568 -5   -4.503  1.8863 0.1050
5    96 127.918 -3  -81.349 56.7991 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What conclusions can we draw here, on the basis of these ANOVA tests?

- The first p value, of 0.3393, compares what the `anova` called Model 1, and what we call `m.full` to what the `anova` called Model 2, and what we call `m08`. So there's no significant decline in predictive value observed when we drop from the `m.full` model to the `m08` model. This suggests that the `m08` model may be a better choice.
- The second p value, of 0.2786, compares `m08` to `m07`, and suggests that we lose no significant predictive value by dropping down to `m07`.
- The third p value, of 0.1050, compares `m07` to `m04`, and suggests that we lose no significant predictive value by dropping down to `m04`.
- But the fourth p value, of 2e-16 (or, functionally, zero), compares `m04` to `m.int` and suggests that we do gain significant predictive value by including the predictors in `m04` as compared to a model with an intercept alone.
- So, by the significance tests, the model we'd select would be `m04`, but, of course, in-sample statistical significance alone isn't a good enough reason to select a model if we want to do prediction well.

8.8 AIC and BIC comparisons, within the training sample

Next, we'll compare the three candidate models (ignoring the intercept-only and kitchen sink models) in terms of their AIC values and BIC values, again using the same sample we used to fit the models in the first place.

```
AIC(m04, m07, m08)
```

df	AIC
m04	5 214.0966
m07	10 214.2327
m08	11 214.9148

```
BIC(m04, m07, m08)
```

df	BIC
m04	5 226.9702
m07	10 239.9798
m08	11 243.2366

- The model with the smallest AIC value shows the best performance within the sample on that measure.
- Similarly, smaller BIC values are associated with predictor sets that perform better in sample on that criterion.
- BIC often suggests smaller models (with fewer regression inputs) than does AIC. Does that happen in this case?
- Note that AIC and BIC can be calculated in a few different ways, so we may see some variation if we don't compare apples to apples with regard to the R functions involved.

8.9 Cross-Validation of Candidate Models out of Sample

8.9.1 20-fold Cross-Validation of model m04

Model m04 uses lcavol, lweight and svi_f to predict the lpsa outcome. Let’s do 20-fold cross-validation of this modeling approach, and calculate the root mean squared prediction error and the mean absolute prediction error for that modeling scheme.

```
set.seed(43201)

cv_m04 <- prost %>%
  crossv_kfold(k = 20) %>%
  mutate(model = map(train,
    ~ lm(lpsa ~ lcavol + lweight + svi_f,
         data = .)))

cv_m04_pred <- cv_m04 %>%
  unnest(map2(model, test, ~ augment(.x, newdata = .y)))

cv_m04_results <- cv_m04_pred %>%
  summarize(Model = "m04",
            RMSE = sqrt(mean((lpsa - .fitted) ^ 2)),
            MAE = mean(abs(lpsa - .fitted)))

cv_m04_results

# A tibble: 1 x 3
  Model   RMSE   MAE
  <chr> <dbl> <dbl>
1 m04    0.725  0.574
```

8.9.2 20-fold Cross-Validation of model m07

Model m07 uses lcavol, lweight, svi_f, age, bph_f, and gleason_f to predict the lpsa outcome. Let’s now do 20-fold cross-validation of this modeling approach, and calculate the root mean squared prediction error and the mean absolute prediction error for that modeling scheme. Note the small changes required, as compared to our cross-validation of model m04 a moment ago.

```
set.seed(43202)

cv_m07 <- prost %>%
  crossv_kfold(k = 20) %>%
  mutate(model = map(train,
    ~ lm(lpsa ~ lcavol + lweight +
         svi_f + age + bph_f +
         gleason_f,
         data = .)))

cv_m07_pred <- cv_m07 %>%
  unnest(map2(model, test, ~ augment(.x, newdata = .y)))

cv_m07_results <- cv_m07_pred %>%
  summarize(Model = "m07",
            RMSE = sqrt(mean((lpsa - .fitted) ^ 2)),
```

```

MAE = mean(abs(lpsa - .fitted)))

cv_m07_results

# A tibble: 1 x 3
  Model   RMSE    MAE
  <chr> <dbl> <dbl>
1 m07    0.730  0.556

```

8.9.3 20-fold Cross-Validation of model m08

Model m08 uses lcavol, lweight, svi_f, age, bph_f, gleason_f and lcp to predict the lpsa outcome. Let's now do 20-fold cross-validation of this modeling approach.

```

set.seed(43202)

cv_m08 <- prost %>%
  crossv_kfold(k = 20) %>%
  mutate(model = map(train,
                     ~ lm(lpsa ~ lcavol + lweight +
                           svi_f + age + bph_f +
                           gleason_f + lcp,
                           data = .)))

cv_m08_pred <- cv_m08 %>%
  unnest(map2(model, test, ~ augment(.x, newdata = .y)))

cv_m08_results <- cv_m08_pred %>%
  summarize(Model = "m08",
            RMSE = sqrt(mean((lpsa - .fitted) ^ 2)),
            MAE = mean(abs(lpsa - .fitted)))

cv_m08_results

# A tibble: 1 x 3
  Model   RMSE    MAE
  <chr> <dbl> <dbl>
1 m08    0.729  0.557

```

8.9.4 Comparing the Results of the Cross-Validations

```

bind_rows(cv_m04_results, cv_m07_results, cv_m08_results)

# A tibble: 3 x 3
  Model   RMSE    MAE
  <chr> <dbl> <dbl>
1 m04    0.725  0.574
2 m07    0.730  0.556
3 m08    0.729  0.557

```

It appears that model m04 has the smallest RMSE and MAE in this case. So, that's the model with the strongest cross-validated predictive accuracy, by these two standards.

8.10 What about Interaction Terms?

Suppose we consider for a moment a much smaller and less realistic problem. We want to use best subsets to identify a model out of a set of three predictors for `lpsa`: specifically `lcavol`, `age` and `svi_f`, but now we also want to consider the interaction of `svi_f` with `lcavol` as a potential addition. Remember that `svi` is the 1/0 numeric version of `svi_f`. We could simply add a numerical product term to our model, as follows.

```
pred2 <- with(prost, cbind(lcavol, age, svi_f, svixlcavol = svi*lcavol))

rs.ks2 <- regsubsets(pred2, y = prost$lpsa,
                      nvmax = NULL, nbest = 1)
rs.summ2 <- summary(rs.ks2)
rs.summ2
```

```
Subset selection object
4 Variables  (and intercept)
      Forced in  Forced out
lcavol        FALSE      FALSE
age           FALSE      FALSE
svi_f         FALSE      FALSE
svixlcavol   FALSE      FALSE
1 subsets of each size up to 4
Selection Algorithm: exhaustive
      lcavol age svi_f svixlcavol
1  ( 1 ) "*"   " "   " "   "
2  ( 1 ) "*"   " "   "*"   " "
3  ( 1 ) "*"   " "   "*"   "*"
4  ( 1 ) "*"   "*"   "*"   "*"
```

In this case, best subsets doesn't identify the interaction term as an attractive predictor until it has already included the main effects that go into it. So that's fine. But if that isn't the case, we would have a problem.

To resolve this, we could:

1. Consider interactions beforehand, and force them in if desired.
2. Consider interaction terms outside of best subsets, and only after the selection of main effects.
3. Use another approach to deal with variable selection for interaction terms.

Chapter 9

Adding Non-linear Terms to a Linear Regression Model

9.1 The pollution data

Consider the `pollution` data set, which contain 15 independent variables and a measure of mortality, describing 60 US metropolitan areas in 1959-1961. The data come from McDonald and Schwing (1973), and are available at <http://www4.stat.ncsu.edu/~boos/var.select/pollution.html> and our web site.

```
pollution
```

```
# A tibble: 60 x 16
  x1     x2     x3     x4     x5     x6     x7     x8     x9     x10    x11
  <int> <int> <int> <dbl> <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1   36     27     71  8.10  3.34 11.4   81.5  3243   8.80  42.6  11.7
2   35     23     72 11.1   3.14 11.0   78.8  4281   3.50  50.7  14.4
3   44     29     74 10.4   3.21  9.80   81.6  4260   0.800 39.4  12.4
4   47     45     79  6.50   3.41 11.1   77.5  3125  27.1   50.2  20.6
5   43     35     77  7.60   3.44  9.60   84.6  6441  24.4   43.7  14.3
6   53     45     80  7.70   3.45 10.2   66.8  3325  38.5   43.1  25.5
7   43     30     74 10.9   3.23 12.1   83.9  4679  3.50   49.2  11.3
8   45     30     73  9.30   3.29 10.6   86.0  2140   5.30  40.4  10.5
9   36     24     70  9.00   3.31 10.5   83.2  6582   8.10  42.5  12.6
10  36     27     72  9.50   3.36 10.7   79.3  4213   6.70  41.0  13.2
# ... with 50 more rows, and 5 more variables: x12 <int>, x13 <int>,
#   x14 <int>, x15 <int>, y <dbl>
```

Here's a codebook:

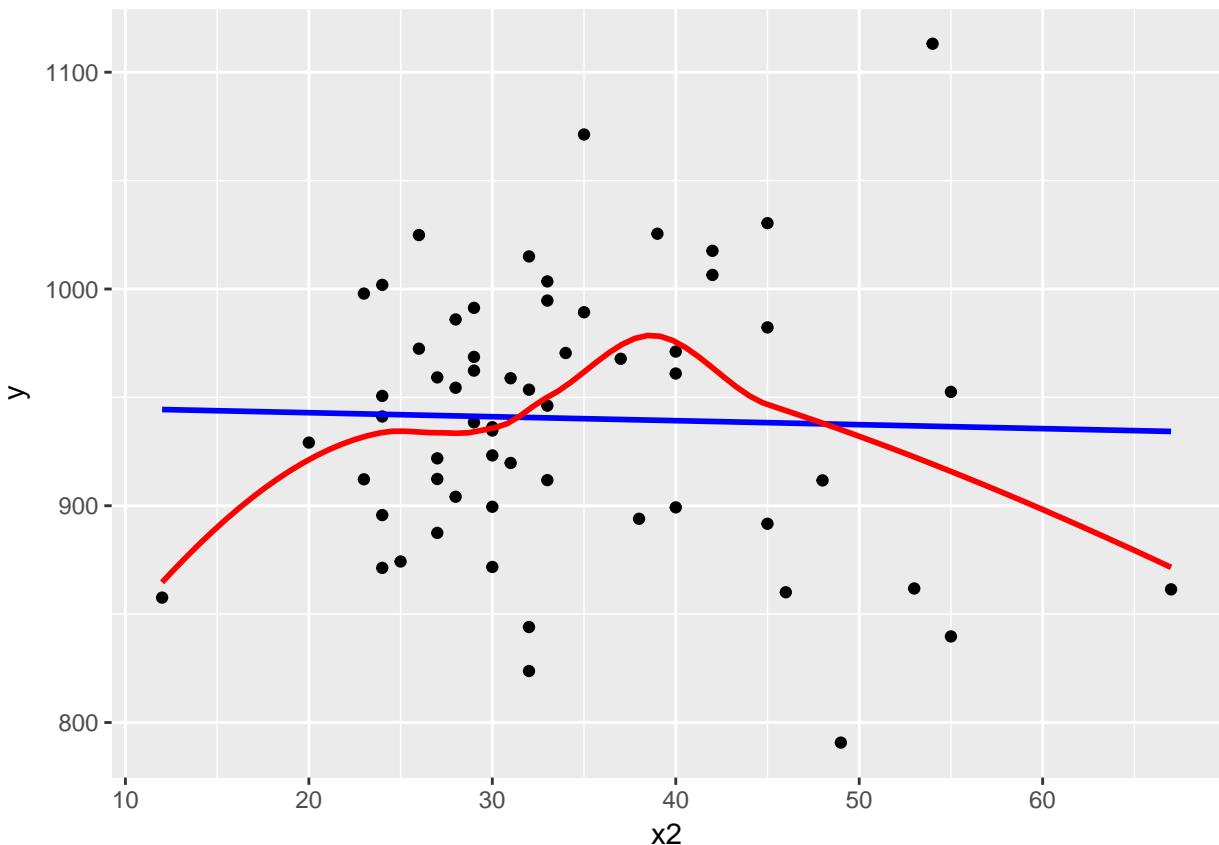
Variable	Description
y	Total Age Adjusted Mortality Rate
x1	Mean annual precipitation in inches
x2	Mean January temperature in degrees Fahrenheit
x3	Mean July temperature in degrees Fahrenheit
x4	Percent of 1960 SMSA population that is 65 years of age or over
x5	Population per household, 1960 SMSA
x6	Median school years completed for those over 25 in 1960 SMSA
x7	Percent of housing units that are found with facilities
x8	Population per square mile in urbanized area in 1960

Variable	Description
x9	Percent of 1960 urbanized area population that is non-white
x10	Percent employment in white-collar occupations in 1960 urbanized area
x11	Percent of families with income under 3,000 in 1960 urbanized area
x12	Relative population potential of hydrocarbons, HC
x13	Relative pollution potential of oxides of nitrogen, NOx
x14	Relative pollution potential of sulfur dioxide, SO ₂
x15	Percent relative humidity, annual average at 1 p.m.

9.2 Fitting a straight line model to predict y from x2

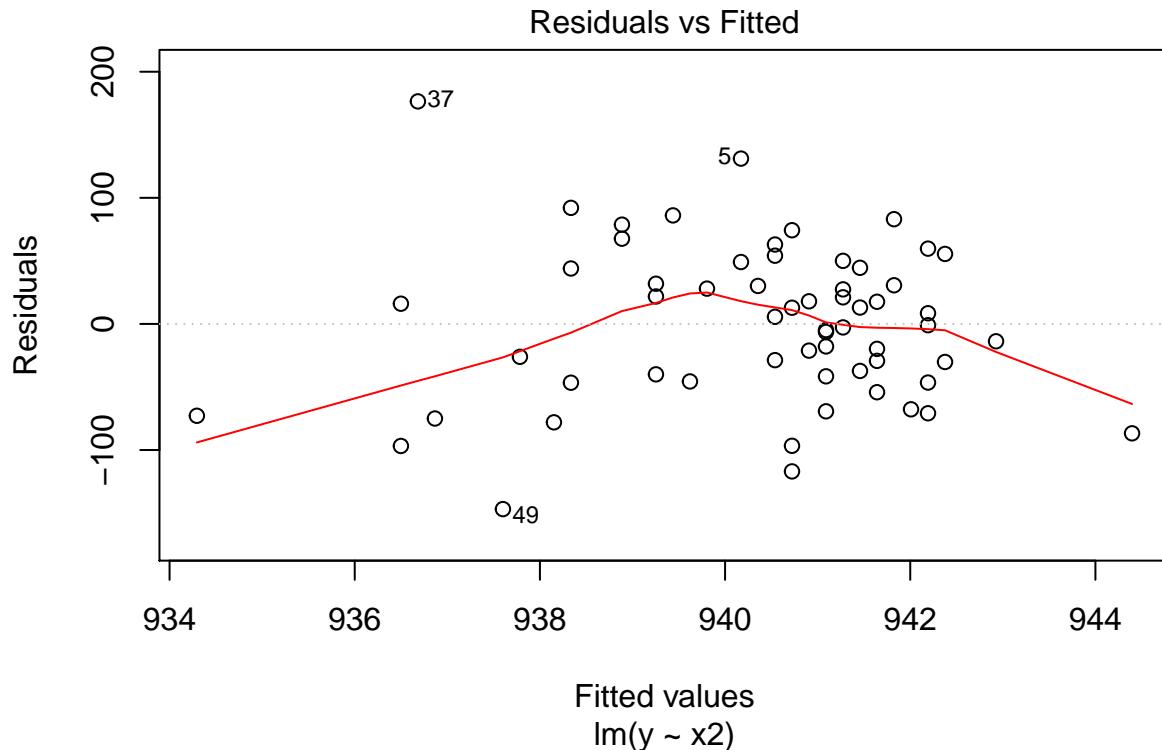
Consider the relationship between y, the age-adjusted mortality rate, and x2, the mean January temperature, across these 60 areas. I'll include both a linear model (in blue) and a loess smooth (in red.) Does the relationship appear to be linear?

```
ggplot(pollution, aes(x = x2, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", col = "blue", se = F) +
  geom_smooth(method = "loess", col = "red", se = F)
```



Suppose we plot the residuals that emerge from the linear model shown in blue, above. Do we see a curve in a plot of residuals against fitted values?

```
plot(lm(y ~ x2, data = pollution), which = 1)
```



9.3 Quadratic polynomial model to predict y using x2

A polynomial in the variable x of degree D is a linear combination of the powers of x up to D.

For example:

- Linear: $y = \beta_0 + \beta_1 x$
- Quadratic: $y = \beta_0 + \beta_1 x + \beta_2 x^2$
- Cubic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$
- Quartic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$
- Quintic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5$

Fitting such a model creates a **polynomial regression**.

9.3.1 The raw quadratic model

Let's look at a **quadratic model** which predicts y using x2 and the square of x2, so that our model is of the form:

$$y = \beta_0 + \beta_1 x_2 + \beta_2 x_2^2 + \text{error}$$

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

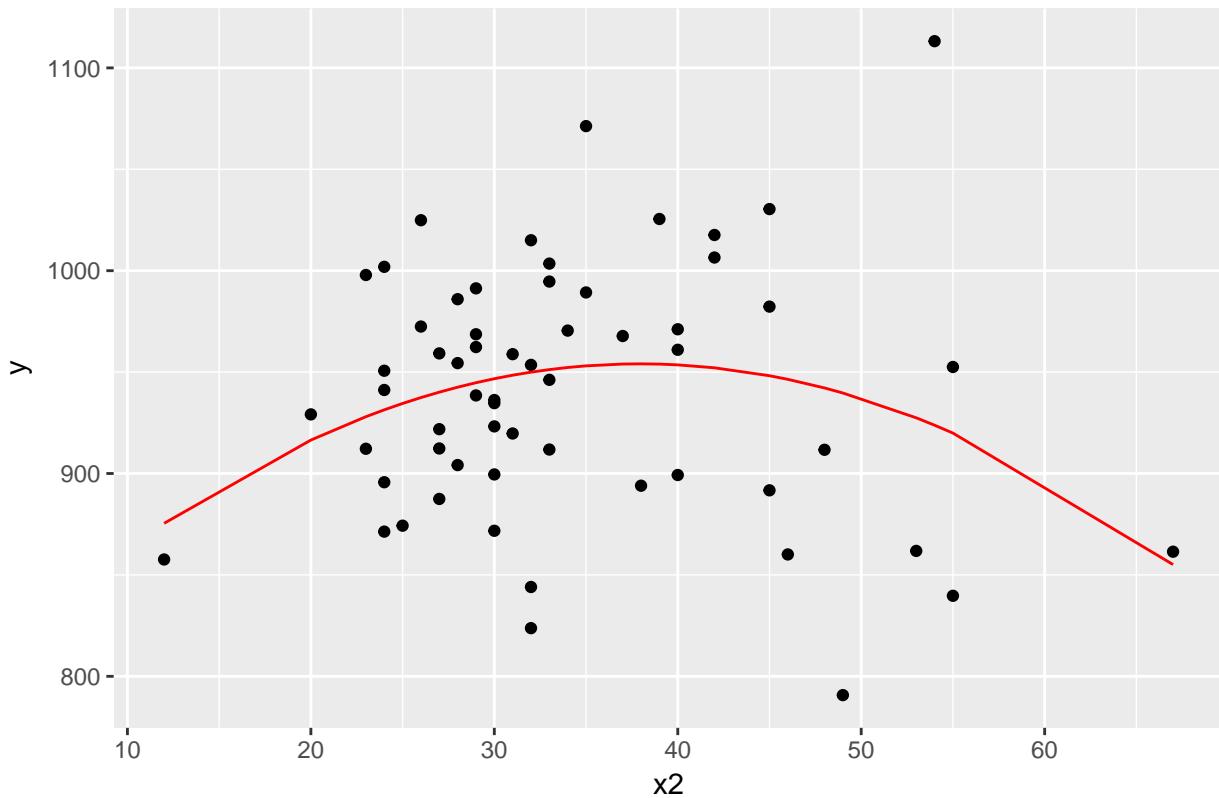
Residual standard error: 60.83 on 57 degrees of freedom
Multiple R-squared:  0.07623, Adjusted R-squared:  0.04382 
F-statistic: 2.352 on 2 and 57 DF,  p-value: 0.1044
```

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")
```

Model 2a: Quadratic fit using x2 and x2^2



```

mod2b.aug <- augment(mod2b, pollution)

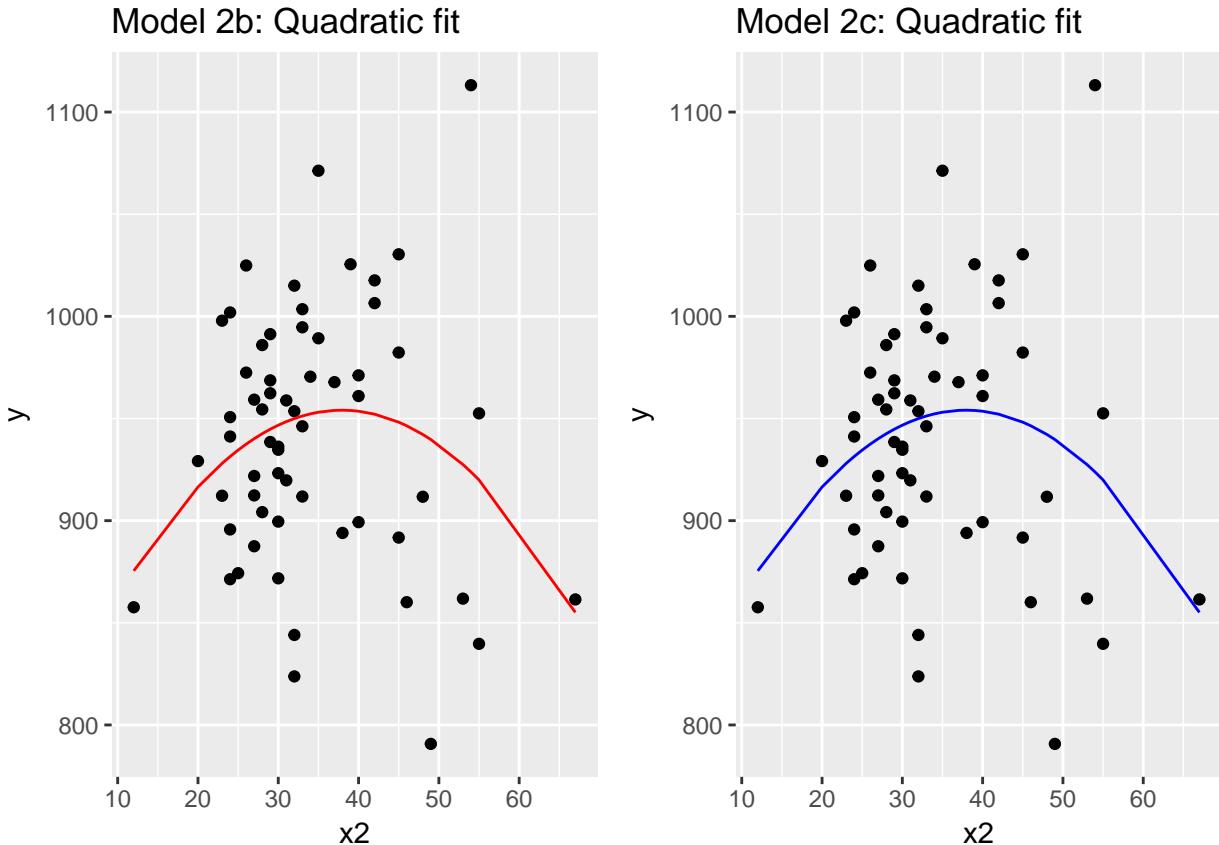
mod2c.aug <- augment(mod2c, pollution)

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")

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

```



9.3.2 Raw quadratic fit after centering x2

Sometimes, we'll center (and perhaps rescale, too) the x2 variable before including it in a quadratic fit like this.

```
pollution <- pollution %>%
  mutate(x2_c = x2 - mean(x2))

mod2d <- lm(y ~ x2_c + I(x2_c^2), data = pollution)

summary(mod2d)
```

Call:
`lm(formula = y ~ x2_c + I(x2_c^2), data = pollution)`

Residuals:

Min	1Q	Median	3Q	Max
-148.977	-38.651	6.889	35.312	189.346

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	952.25941	9.59896	99.204	<2e-16 ***
x2_c	0.92163	0.93237	0.988	0.3271
I(x2_c^2)	-0.11704	0.05429	-2.156	0.0353 *

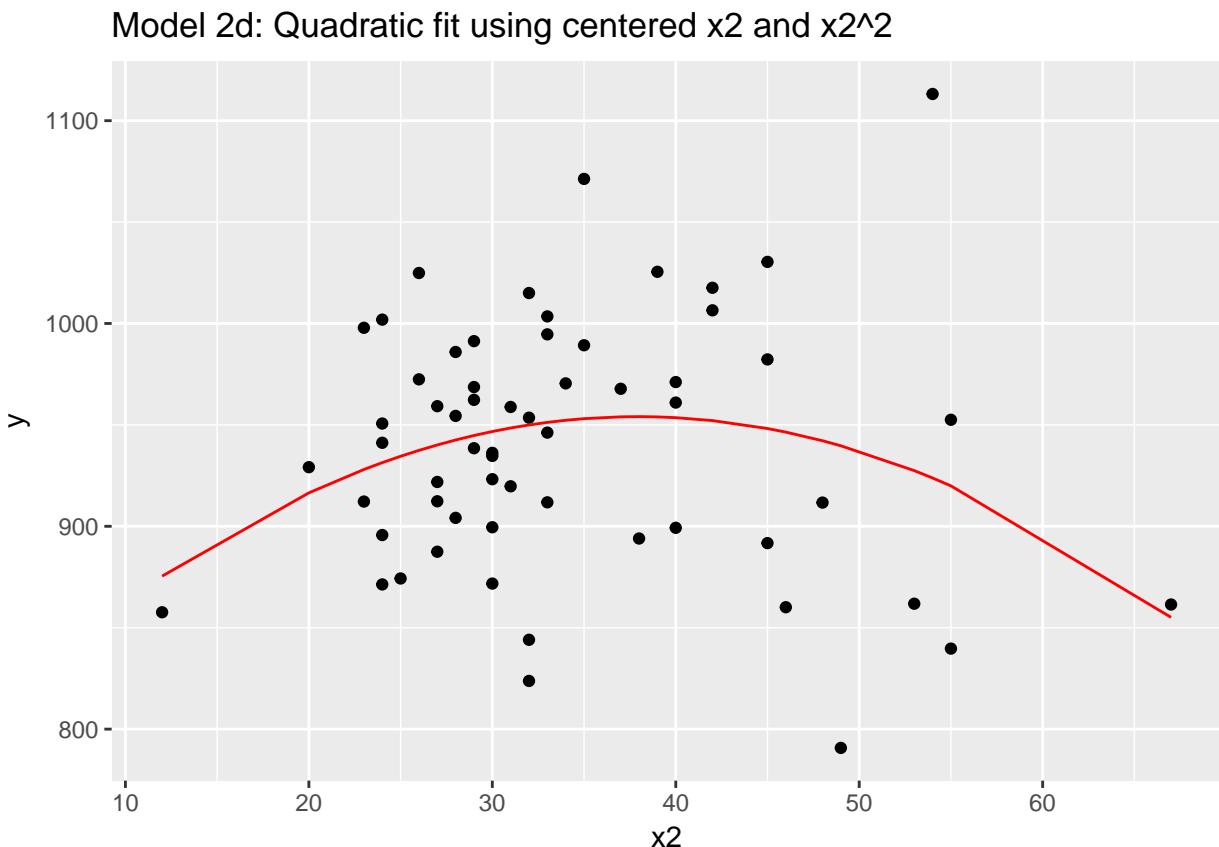
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.83 on 57 degrees of freedom
Multiple R-squared: 0.07623, Adjusted R-squared: 0.04382
F-statistic: 2.352 on 2 and 57 DF, p-value: 0.1044

Note that this model looks very different, with the exception of the second order quadratic term. But, it produces the same fitted values as the models we fit previously.

```
mod2d.aug <- augment(mod2d, 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.

```
mod2a.aug %>% skim(.fitted)
```

Skim summary statistics

n obs: 60

n variables: 24

Variable type: numeric

```
variable missing complete n mean sd p0 p25 median p75
.fitted 0 60 60 940.36 17.18 855.1 936.72 945.6 950.29
p100
```

```

954.07

mod2b.aug %>% skim(.fitted)

Skim summary statistics
n obs: 60
n variables: 24

Variable type: numeric
variable missing complete n    mean     sd    p0    p25 median    p75
.fitted      0       60 60 940.36 17.18 855.1 936.72 945.6 950.29
p100
954.07

mod2c.aug %>% skim(.fitted)

Skim summary statistics
n obs: 60
n variables: 24

Variable type: numeric
variable missing complete n    mean     sd    p0    p25 median    p75
.fitted      0       60 60 940.36 17.18 855.1 936.72 945.6 950.29
p100
954.07

mod2d.aug %>% skim(.fitted)

Skim summary statistics
n obs: 60
n variables: 25

Variable type: numeric
variable missing complete n    mean     sd    p0    p25 median    p75
.fitted      0       60 60 940.36 17.18 855.1 936.72 945.6 950.29
p100
954.07

```

9.4 Orthogonal Polynomials

Now, let's fit an orthogonal polynomial of degree 2 to predict y using x2.

```

mod2_orth <- lm(y ~ poly(x2, 2), data = pollution)

summary(mod2_orth)

```

```

Call:
lm(formula = y ~ poly(x2, 2), data = pollution)

Residuals:
    Min      1Q  Median      3Q      Max 
-148.977 -38.651   6.889  35.312 189.346 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  940.360   17.180   54.88   <2e-16 ***
poly(x2, 2)  17.180    3.467   4.97    0.0002 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

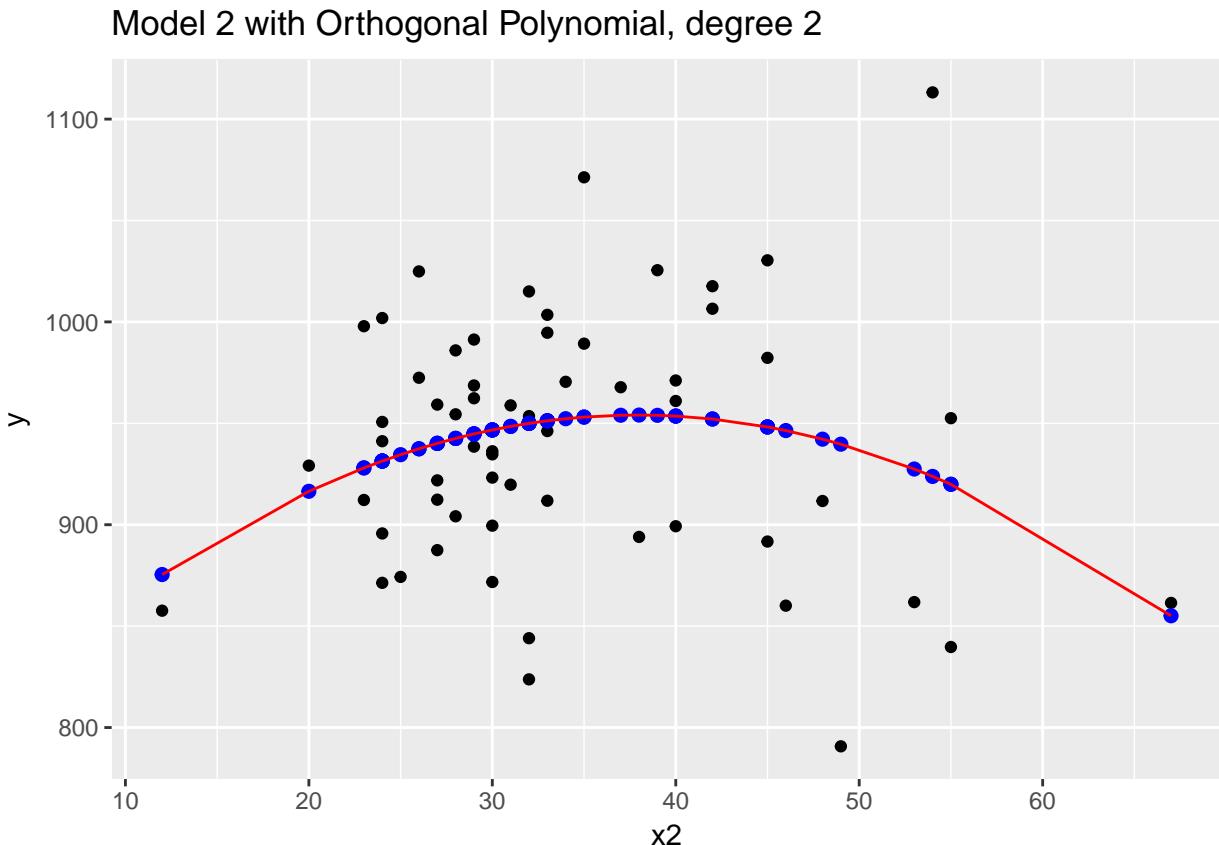
```
(Intercept) 940.358      7.853 119.746    <2e-16 ***
poly(x2, 2)1 -14.345     60.829 -0.236     0.8144
poly(x2, 2)2 -131.142    60.829 -2.156     0.0353 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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

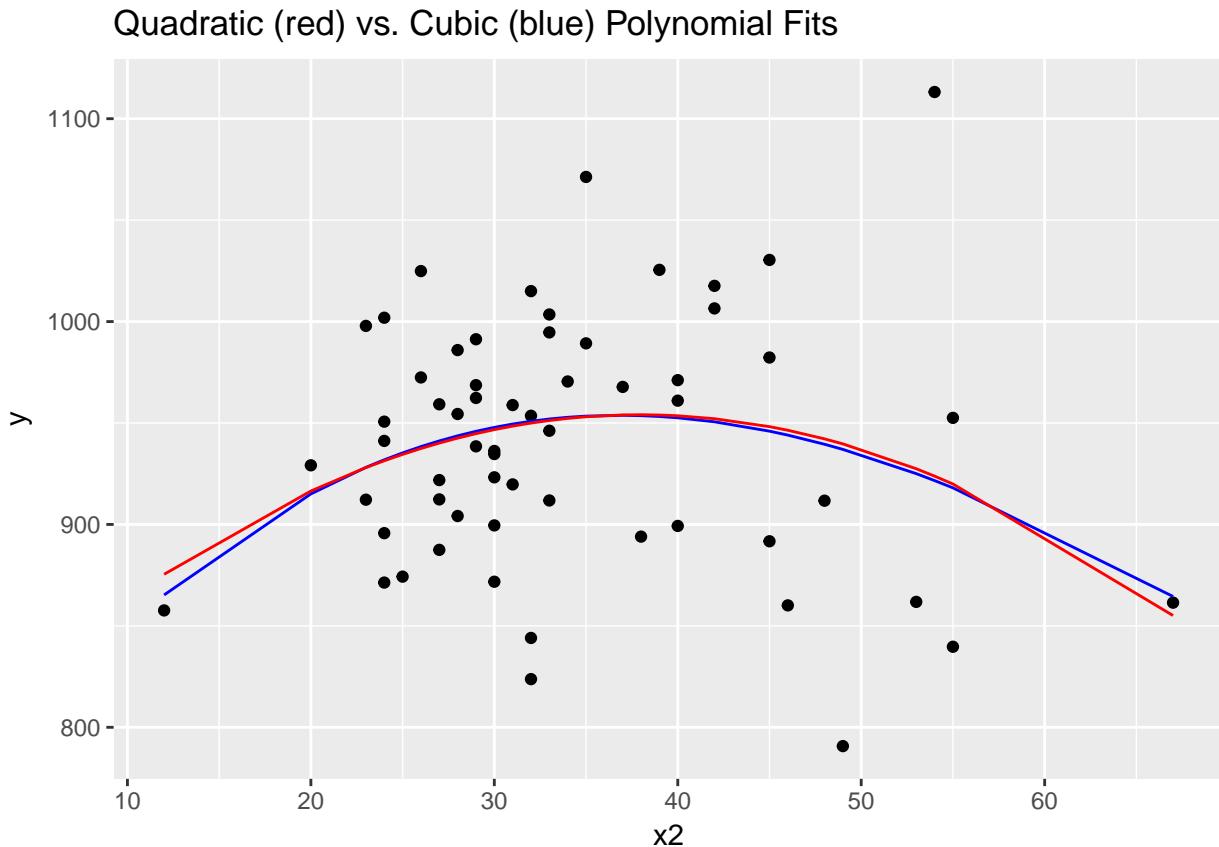
Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'
	0.1	' '	' '	1

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)

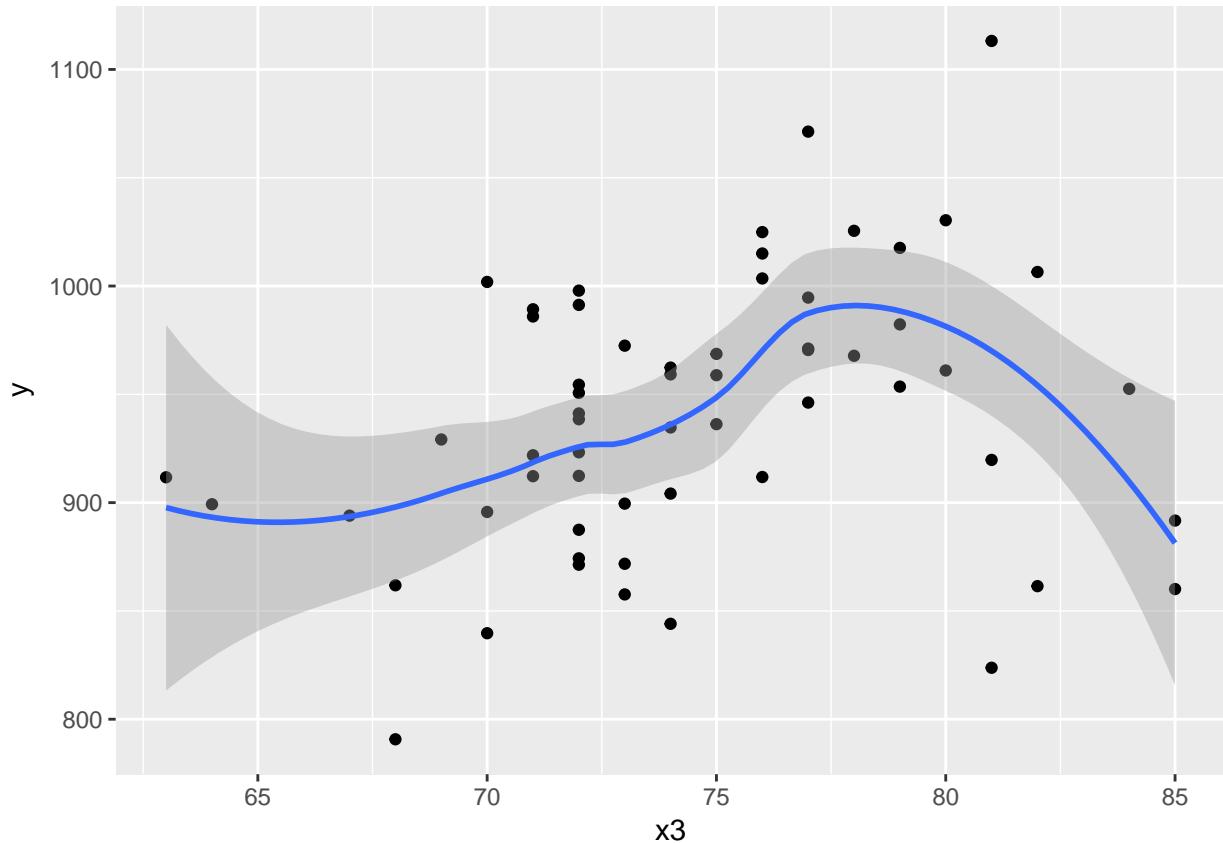
ggplot(mod3.aug, aes(x = x2, y = y)) +
  geom_point() +
  geom_line(aes(x = x2, y = .fitted),
            col = "blue") +
  geom_line(data = mod2orth.aug, aes(x = x2, y = .fitted),
            col = "red") +
  labs(title = "Quadratic (red) vs. Cubic (blue) Polynomial Fits")
```



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")
```



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),

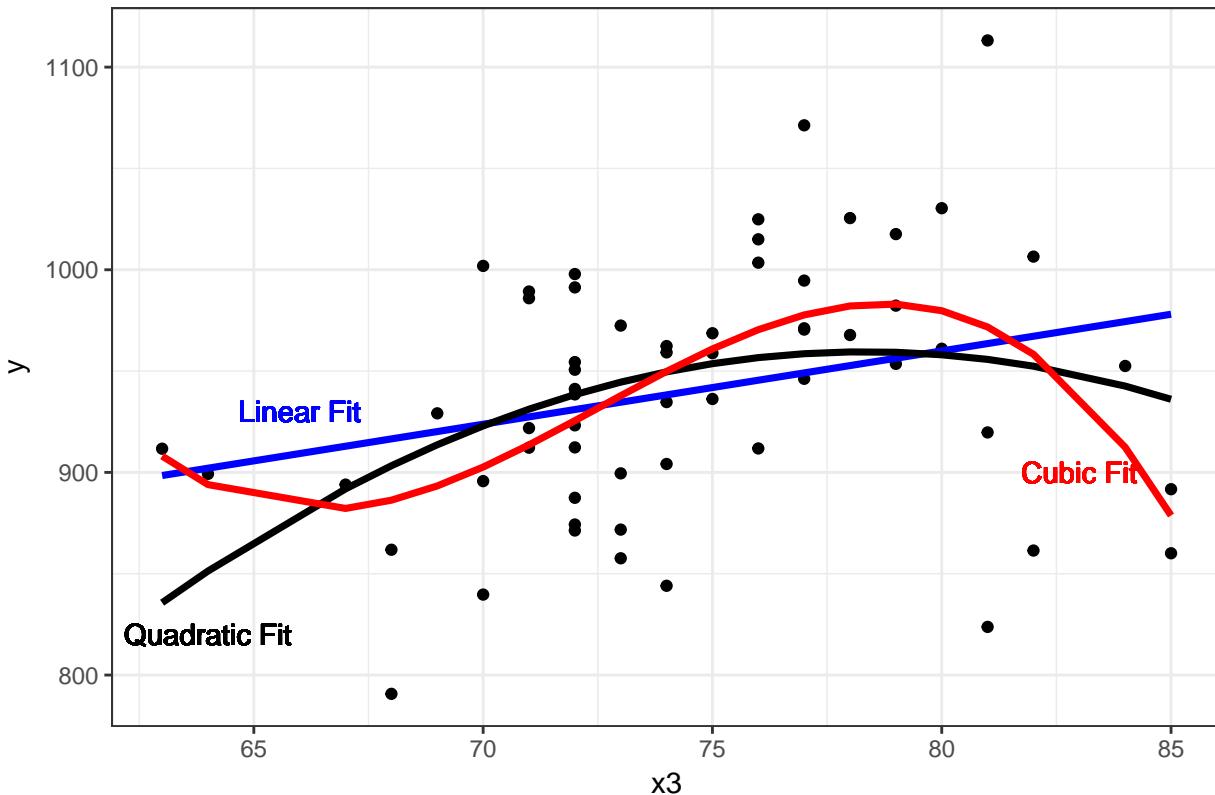
```

```

    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()

```

Linear, Quadratic and Cubic Fits predicting y with x3



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 x_3 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
<code>rcs(x3, 5)x3</code>	6.447	5.749	1.121	0.267
<code>rcs(x3, 5)x3'</code>	-25.633	46.810	-0.548	0.586
<code>rcs(x3, 5)x3''</code>	323.137	293.065	1.103	0.275
<code>rcs(x3, 5)x3'''</code>	-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) +
```

```

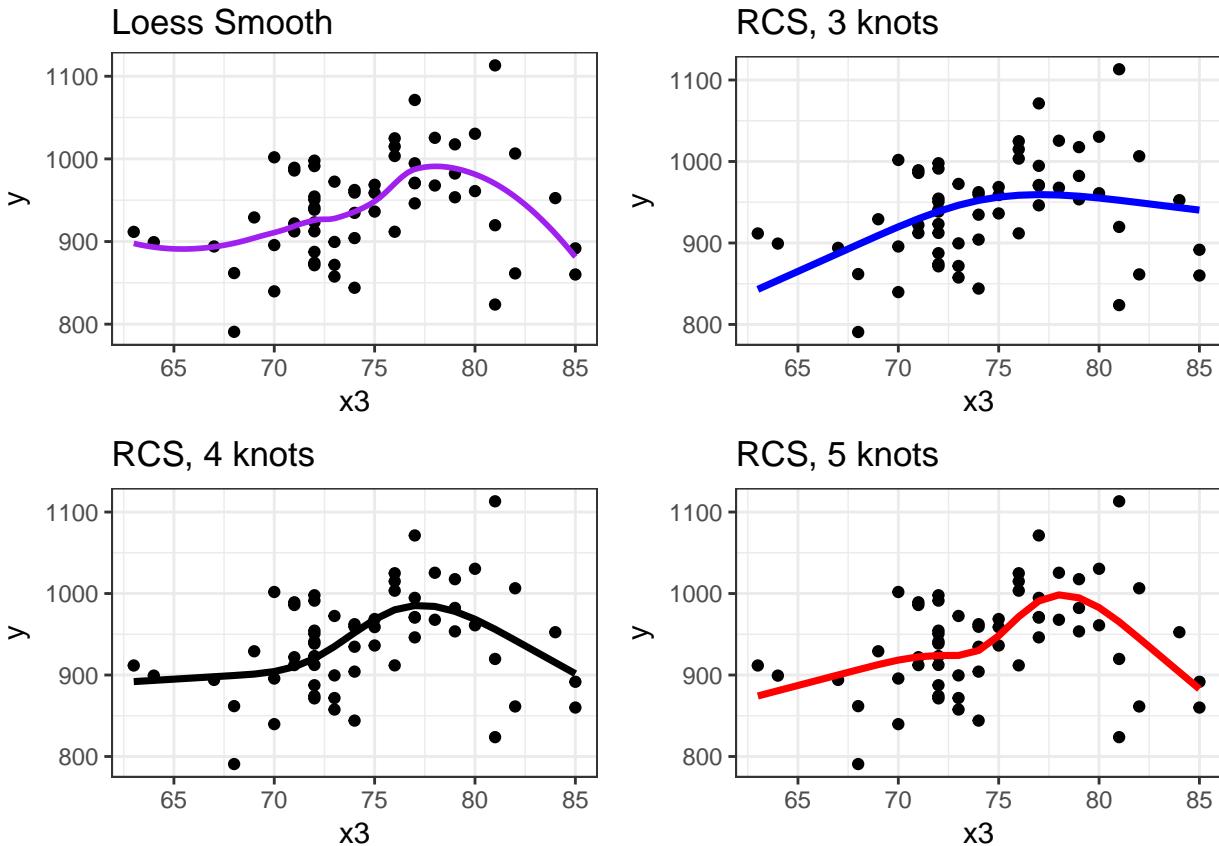
  labs(title = "RCS, 3 knots") +
  theme_bw()

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()

gridExtra::grid.arrange(p2, p3, p4, p5, nrow = 2)

```



Does it seem like the fit improves markedly (perhaps approaching the loess smooth result) as we increase the number of knots?

```
anova(mod5a_rcs, mod5b_rcs, mod5c_rcs)
```

Analysis of Variance Table

Model 1: $y \sim \text{rcs}(x3, 3)$

```

Model 2: y ~ rcs(x3, 4)
Model 3: y ~ rcs(x3, 5)
  Res.Df   RSS Df Sum of Sq    F   Pr(>F)
1      57 194935
2      56 171448  1   23486.9 7.9503 0.006672 **
3      55 162481  1   8967.2 3.0354 0.087057 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Based on an ANOVA comparison, the fourth knot adds significant predictive value ($p = 0.0067$), but the fifth knot is borderline ($p = 0.0871$). From the `glance` function in the `broom` package, we can also look at some key summaries.

```
glance(mod5a_rcs)
```

```

  r.squared adj.r.squared    sigma statistic   p.value df   logLik
1 0.146184     0.1162256 58.48006  4.879558 0.01106323 3 -327.7187
      AIC      BIC deviance df.residual
1 663.4373 671.8147 194935.3          57

```

```
glance(mod5b_rcs)
```

```

  r.squared adj.r.squared    sigma statistic   p.value df   logLik
1 0.2490566    0.2088274 55.33153  6.190953 0.0010423 4 -323.8671
      AIC      BIC deviance df.residual
1 657.7342 668.2059 171448.4          56

```

```
glance(mod5c_rcs)
```

```

  r.squared adj.r.squared    sigma statistic   p.value df   logLik
1 0.2883327    0.2365751 54.35259  5.570826 0.0007734418 5 -322.2555
      AIC      BIC deviance df.residual
1 656.511 669.0771 162481.2          55

```

Model	Knots	R ²	Adj. R ²	AIC	BIC
5a	3	0.146	0.116	663.4	671.8
5b	4	0.249	0.209	657.7	668.2
5c	5	0.288	0.237	656.5	669.1

Within our sample, the five-knot RCS outperforms the 3- and 4-knot versions on adjusted R² and AIC (barely) and does a little worse than the 4-knot RCS on BIC.

Of course, we could also use the cross-validation methods we've developed for other linear regressions to assess predictive capacity of these models. I'll skip that for now.

To see the values of x3 where the splines place their knots, we can use the `attributes` function.

```
attributes(rcs(pollution$x3, 5))
```

```

$dim
[1] 60  4

$dimnames
$dimnames[[1]]
NULL

$dimnames[[2]]
[1] "pollution"    "pollution"    "pollution"    "pollution"

```

```

$class
[1] "rms"

$name
[1] "pollution"

$label
[1] "pollution"

$assume
[1] "rcspline"

$assume.code
[1] 4

$parms
[1] 68 72 74 77 82

$nonlinear
[1] FALSE TRUE TRUE TRUE

$colnames
[1] "pollution"    "pollution''"   "pollution'''"  "pollution'''''"

```

The knots in this particular 5-knot spline are placed by the computer at 68, 72, 74, 77 and 82, it seems.

There are two kinds of Multivariate Regression Models

1. [Prediction] Those that are built so that we can make accurate predictions.
2. [Explanatory] Those that are built to help understand underlying phenomena.

While those two notions overlap considerably, they do imply different things about how we strategize about model-building and model assessment. Harrell's primary concern is effective use of the available data for **prediction** - this implies some things that will be different from what we've seen in the past.

Harrell refers to multivariable regression modeling strategy as the process of **spending degrees of freedom**. The main job in strategizing about multivariate modeling is to

1. Decide the number of degrees of freedom that can be spent
2. Decide where to spend them
3. Spend them, wisely.

What this means is essentially linked to making decisions about predictor complexity, both in terms of how many predictors will be included in the regression model, and about how we'll include those predictors.

9.7 “Spending” Degrees of Freedom

- “Spending” df includes
 - fitting parameter estimates in models, or
 - examining figures built using the outcome variable Y that tell you how to model the predictors.

If you use a scatterplot of Y vs. X or the residuals of the Y-X regression model vs. X to decide whether a linear model is appropriate, then how many degrees of freedom have you actually spent?

Grambsch and O'Brien conclude that if you wish to preserve the key statistical properties of the various estimation and fitting procedures used in building a model, you can't retrieve these degrees of freedom once

they have been spent.

9.7.1 Overfitting and Limits on the # of Predictors

Suppose you have a total sample size of n observations, then you really shouldn't be thinking about estimating more than $n/15$ regression coefficients, at the most.

- If k is the number of parameters in a full model containing all candidate predictors for a stepwise analysis, then k should be no greater than $n/15$.
- k should include all variables screened for association with the response, including interaction terms.

So if you have 97 observations in your data, then you can probably just barely justify the use of a stepwise analysis using the main effects alone of 5 candidate variables (with one additional DF for the intercept term.)

Harrell (2001) also mentions that if you have a **narrowly distributed** predictor, without a lot of variation to work with, then an even larger sample size n should be required. See Vittinghoff et al. (2012), Section 10.3 for more details.

9.7.2 The Importance of Collinearity

Collinearity denotes correlation between predictors high enough to degrade the precision of the regression coefficient estimates substantially for some or all of the correlated predictors

- Vittinghoff et al. (2012), section 10.4.1
- Can one predictor in a model be predicted well using the other predictors in the model?
 - Strong correlations (for instance, $r \geq 0.8$) are especially troublesome.
- Effects of collinearity
 - decreases precision, in the sense of increasing the standard errors of the parameter estimates
 - decreases power
 - increases the difficulty of interpreting individual predictor effects
 - overall F test is significant, but individual t tests may not be

Suppose we want to assess whether variable X_j is collinear with the other predictors in a model. We run a regression predicting X_j using the other predictors, and obtain the R^2 . The VIF is defined as $1 / (1 - \text{this } R^2)$, and we usually interpret VIFs above 5 as indicating a serious multicollinearity problem (i.e. R^2 values for this predictor of 0.8 and above would thus concern us.)

```
vif(lm(y ~ x1 + x2 + x3 + x4 + x5 + x6, data = pollution))
```

x1	x2	x3	x4	x5	x6
2.238862	2.058731	2.153044	4.174448	3.447399	1.792996

Occasionally, you'll see the inverse of VIF reported, and this is called *tolerance*.

- tolerance = $1 / \text{VIF}$

9.7.3 Collinearity in an Explanatory Model

- When we are attempting to **identify multiple independent predictors** (the explanatory model approach), then we will need to choose between collinear variables
 - options suggested by Vittinghoff et al. (2012), p. 422, include choosing on the basis of plausibility as a causal factor,
 - choosing the variable that has higher data quality (is measured more accurately or has fewer missing values.)

- Often, we choose to include a variable that is statistically significant as a predictor, and drop others, should we be so lucky.
- Larger effects, especially if they are associated with predictors that have minimal correlation with the other predictors under study, cause less trouble in terms of potential violation of the $n/15$ rule for what constitutes a reasonable number of predictors.

9.7.4 Collinearity in a Prediction Model

- If we are primarily building a **prediction model** for which inference on the individual predictors is not of interest, then it is totally reasonable to use both predictors in the model, if doing so reduces prediction error.
 - Collinearity doesn't affect predictions in our model development sample.
 - Collinearity doesn't affect predictions on new data so long as the new data have similar relationships between predictors.
 - If our key predictor is correlated strongly with a confounder, then if the predictor remains significant after adjustment for the confounder, then this suggests a meaningful independent effect.
 - * If the effects of the predictor are clearly confounded by the adjustment variable, we again have a clear result.
 - * If neither is statistically significant after adjustment, the data may be inadequate.
 - If the collinearity is between adjustment variables, but doesn't involve the key predictor, then inclusion of the collinear variables is unlikely to cause substantial problems.

9.8 Spending DF on Non-Linearity: The Spearman ρ^2 Plot

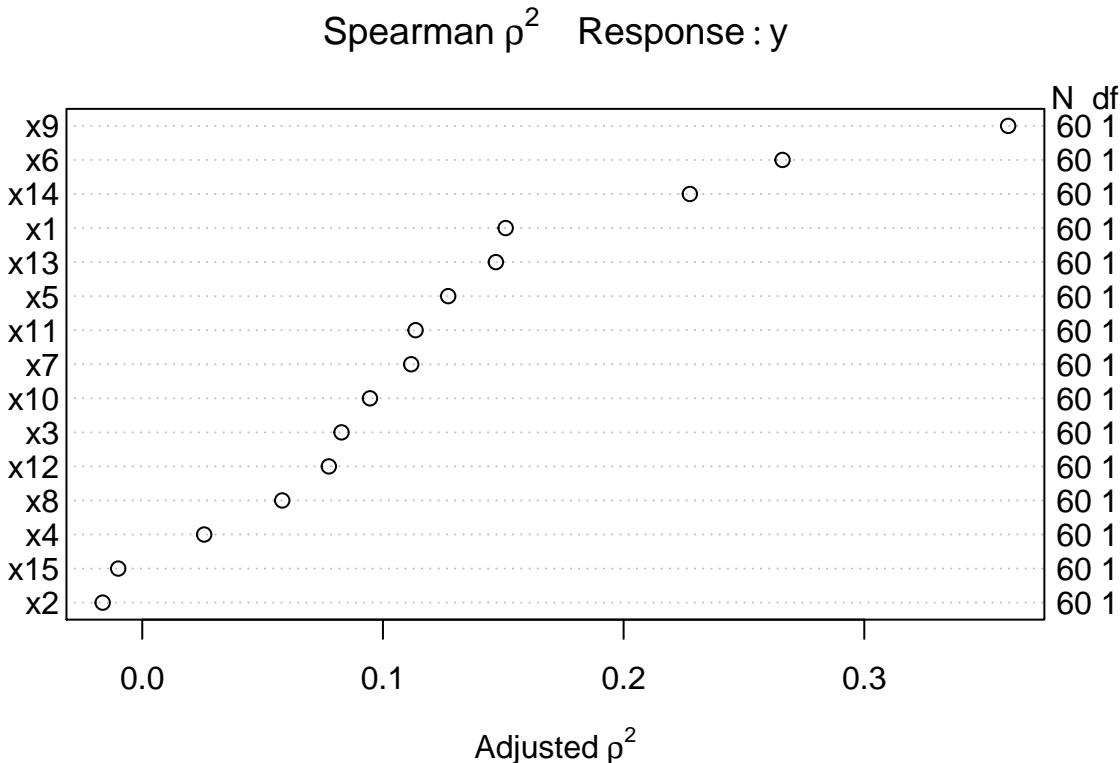
We need a flexible approach to assessing non-linearity and fitting models with non-linear predictors. This will lead us to a measure of what Harrell (2001) calls **potential predictive punch** which hides the true form of the regression from the analyst so as to preserve statistical properties, but that lets us make sensible decisions about whether a predictor should be included in a model, and the number of parameters (degrees of freedom, essentially) we are willing to devote to it.

What if we want to consider where best to spend our degrees of freedom on non-linear predictor terms, like interactions, polynomial functions or curved splines to represent our input data? The approach we'll find useful in the largest variety of settings is a combination of

1. a rank correlation assessment of potential predictive punch (using a Spearman ρ^2 plot, available in the **Hmisc** package), followed by
2. the application of restricted cubic splines to fit and assess models.

Suppose, for instance, that we want to create a model for y using some combination of linear and non-linear terms drawn from the complete set of 15 predictors available in the **pollution** data. I'd begin by running a Spearman ρ^2 plot:

```
plot(Hmisc::spearman2(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 +
                      x8 + x9 + x10 + x11 + x12 + x13 +
                      x14 + x15, data = pollution))
```



The variable with the largest adjusted squared Spearman ρ statistic in this setting is **x9**, followed by **x6** and **x14**. With only 60 observations, we might well want to restrict ourselves to a very small model. What the Spearman plot suggests is that we focus any non-linear terms on **x9** first, and then perhaps **x6** and **x14** as they have some potential predictive power. It may or may not work out that the non-linear terms are productive.

9.8.1 Fitting a Big Model to the pollution data

So, one possible model built in reaction this plot might be to fit:

- a restricted cubic spline with 5 knots on **x9**
- a restricted cubic spline with 3 knots on **x6**
- and a quadratic polynomial on **x14**
- plus a linear fit to **x1** and **x13**

That's way more degrees of freedom (4 for **x9**, 2 for **x6**, 2 for **x14** and 1 each for **x1** and **x13** makes a total of 10 without the intercept term) than we can really justify with a sample of 60 observations. But let's see what happens.

```
mod_big <- lm(y ~ rcs(x9, 5) + rcs(x6, 3) + poly(x14, 2) + x1 + x13, data = pollution)

anova(mod_big)
```

Analysis of Variance Table

```
Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
rcs(x9, 5)     4 100164 25040.9 17.8482 4.229e-09 ***
```

```

rcs(x6, 3)    2  38306 19152.8 13.6513 1.939e-05 ***
poly(x14, 2)  2  15595  7797.7  5.5579  0.006677 **
x1            1   4787  4787.3  3.4122  0.070759 .
x13           1    712   711.9  0.5074  0.479635
Residuals     49  68747  1403.0
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

This `anova` suggests that we have at least some predictive value in each spline (`x9` and `x6`) and some additional value in `x14`, although it's not as clear that the linear terms (`x1` and `x13`) did much good.

9.8.2 Limitations of `lm` for fitting complex linear regression models

We can certainly assess this big, complex model using `lm` in comparison to other models:

- with in-sample summary statistics like adjusted R^2 , AIC and BIC,
- we can assess its assumptions with residual plots, and
- we can also compare out-of-sample predictive quality through cross-validation,

But to really delve into the details of how well this complex model works, and to help plot what is actually being fit, we'll probably want to fit the model using an alternative method for fitting linear models, called `ols`, from the `rms` package developed by Frank Harrell and colleagues. That will be the focus of our next chapter.

Chapter 10

Using `ols` from the `rms` package to fit linear models

At the end of the previous chapter, we had fit a model to the `pollution` data that predicted our outcome $y = \text{Age-Adjusted Mortality Rate}$, 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 $-2(\log \text{likelihood})$, 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.

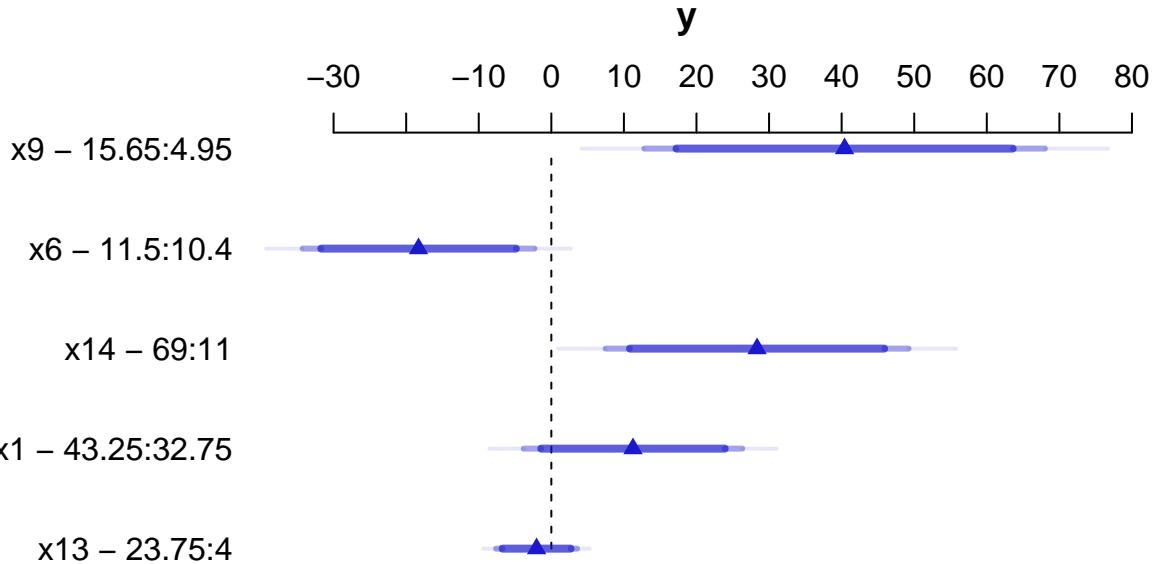
```
summary(newmod)
```

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
x9	4.95	15.65	10.70	40.4060	14.0790	3.12280	77.6890

```
x6      10.40 11.50  1.10 -18.2930  8.1499 -39.87400  3.2882
x14     11.00 69.00 58.00  28.3480 10.6480   0.15192 56.5440
x1      32.75 43.25 10.50  11.2520  7.6833 -9.09340 31.5980
x13     4.00 23.75 19.75 -2.0303  2.8502 -9.57760  5.5171
```

Remember that if you're looking for the usual `lm` summary for an `ols` object, use `summary.lm`, and that the `display` function from `arm` does not recognize `ols` objects.

10.4 The Predict function for an ols model

The `Predict` function is very flexible, and can be used to produce individual or simultaneous confidence limits.

```
Predict(newmod, x9 = 12, x6 = 12, x14 = 40, x1 = 40, x13 = 20) # individual limits

x9 x6 x14 x1 x13      yhat    lower    upper
1 12 12 40 40  923.0982 893.0984 953.098
```

Response variable (y): y

Limits are 0.95 confidence limits

```
Predict(newmod, x9 = 5:15) # individual limits

x9      x6 x14 x1 x13      yhat    lower    upper
1 5 11.05 30 38  9 913.7392 889.4802 937.9983
2 6 11.05 30 38  9 916.3490 892.0082 940.6897
3 7 11.05 30 38  9 921.3093 898.9657 943.6529
4 8 11.05 30 38  9 927.6464 907.0355 948.2574
5 9 11.05 30 38  9 934.3853 913.3761 955.3946
6 10 11.05 30 38  9 940.5510 917.8371 963.2648
7 11 11.05 30 38  9 945.2225 921.9971 968.4479
8 12 11.05 30 38  9 948.2885 926.4576 970.1194
9 13 11.05 30 38  9 950.2608 930.3003 970.2213
10 14 11.05 30 38  9 951.6671 932.2370 971.0971
11 15 11.05 30 38  9 953.0342 932.1662 973.9021
```

Response variable (y): y

Adjust to: x6=11.05 x14=30 x1=38 x13=9

Limits are 0.95 confidence limits

```
Predict(newmod, x9 = 5:15, conf.type = 'simult')

x9      x6 x14 x1 x13      yhat    lower    upper
1 5 11.05 30 38  9 913.7392 882.4311 945.0473
2 6 11.05 30 38  9 916.3490 884.9354 947.7625
3 7 11.05 30 38  9 921.3093 892.4733 950.1454
4 8 11.05 30 38  9 927.6464 901.0465 954.2464
5 9 11.05 30 38  9 934.3853 907.2713 961.4993
6 10 11.05 30 38  9 940.5510 911.2371 969.8649
7 11 11.05 30 38  9 945.2225 915.2484 975.1966
8 12 11.05 30 38  9 948.2885 920.1141 976.4629
9 13 11.05 30 38  9 950.2608 924.5003 976.0212
10 14 11.05 30 38  9 951.6671 926.5912 976.7430
```

```
11 15 11.05 30 38    9 953.0342 926.1025 979.9658
```

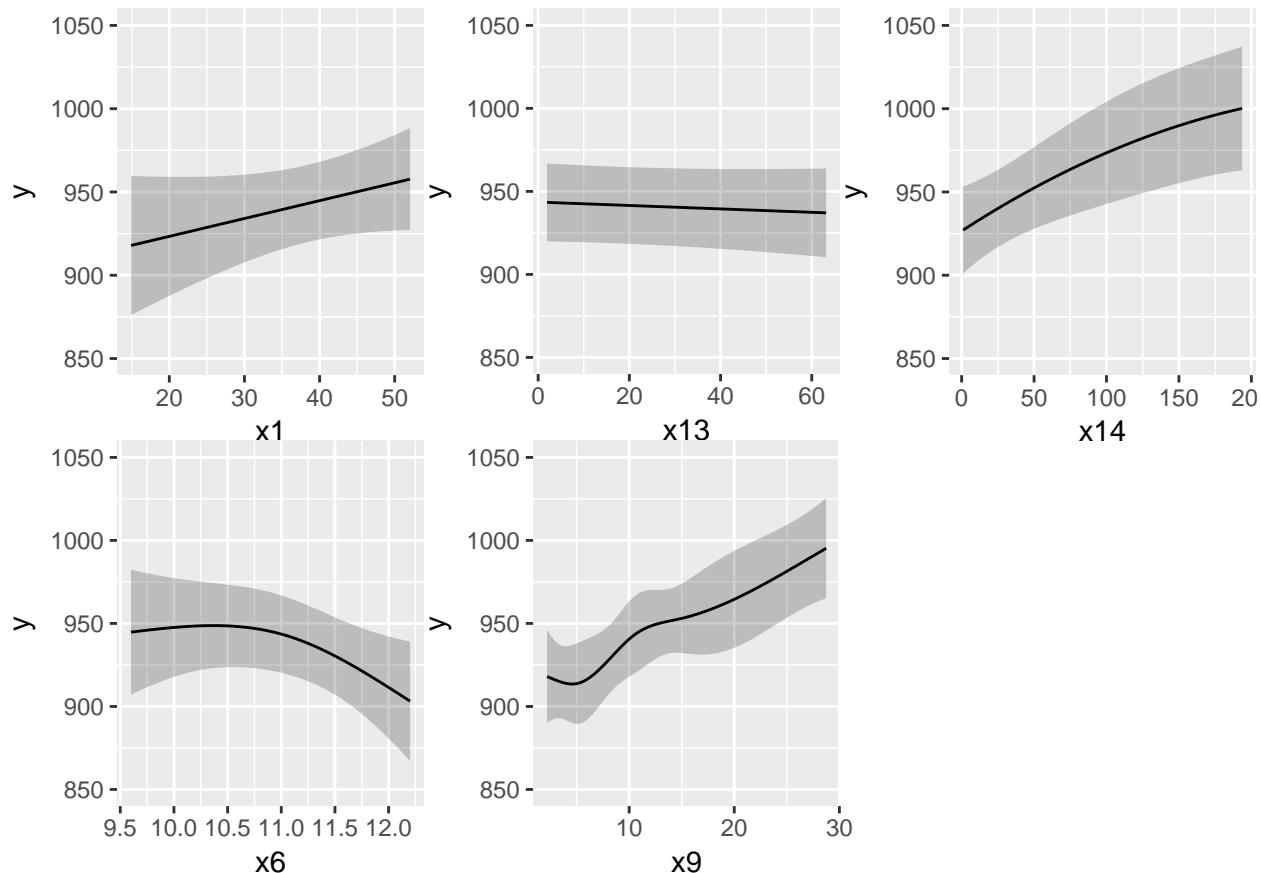
Response variable (y): y

Adjust to: x6=11.05 x14=30 x1=38 x13=9

Limits are 0.95 confidence limits

The plot below shows the individual effects in `newmod` in five subpanels, using the default approach of displaying the same range of values as are seen in the data. Note that each panel shows point and interval estimates of the effects, and spot the straight lines in `x1` and `x13`, the single bends in `x14` and `x6` and the wiggles in `x9`, corresponding to the amount of non-linearity specified in the model.

```
ggplot(Predict(newmod))
```



10.5 Checking Influence via `dfbeta`

For an `ols` object, we have several tools for looking at residuals. The most interesting to me is `which.influence` which is reliant on the notion of `dfbeta`.

- DFBETA is estimated for each observation in the data, and each coefficient in the model.
- The DFBETA is the difference in the estimated coefficient caused by deleting the observation, scaled by the coefficient's standard error estimated with the observation deleted.
- The `which.influence` command applied to an `ols` model produces a list of all of the predictors estimated by the model, including the intercept.
 - For each predictor, the command lists all observations (by row number) that, if removed from the

model, would cause the estimated coefficient (the “beta”) for that predictor to change by at least some particular cutoff.

- The default is that the DFBETA for that predictor is 0.2 or more.

```
which.influence(newmod)
```

```
$Intercept
[1] 2 11 28 32 37 49 59

$x9
[1] 2 3 6 9 31 35 49 57 58

$x6
[1] 2 11 15 28 32 37 50 56 59

$x14
[1] 2 6 7 12 13 16 32 37

$x1
[1] 7 18 32 37 49 57

$x13
[1] 29 32 37
```

The implication here, for instance, is that if we drop row 3 from our data frame, and refit the model, this will have a meaningful impact on the estimate of `x9` but not on the other coefficients. But if we drop, say, row 37, we will affect the estimates of the intercept, `x6`, `x14`, `x1`, and `x13`.

10.5.1 Using the `residuals` command for `dfbetas`

To see the `dfbeta` values, standardized according to the approach I used above, you can use the following code (I’ll use `head` to just show the first few rows of results) to get a matrix of the results.

```
head(residuals(newmod, type = "dfbetas"))
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	0.03071160	-0.023775487	-0.004055111	0.01205425	-0.03260003
[2,]	-0.38276573	-0.048404993	-0.142293606	0.17009666	-0.22350621
[3,]	0.17226780	-0.426153536	0.350913139	-0.32949129	0.25777913
[4,]	0.06175110	-0.006460916	0.024828272	-0.03009337	0.04154812
[5,]	0.16875200	0.039839994	-0.058178534	0.06449504	-0.07772208
[6,]	0.03322073	0.112699877	-0.203543632	0.23987378	-0.35201736
	[,6]	[,7]	[,8]	[,9]	[,10]
[1,]	-0.02392315	0.01175375	-0.06494414	0.060929683	-0.011042644
[2,]	0.44737372	-0.48562818	0.19372285	-0.212186731	-0.107830147
[3,]	-0.10263448	0.05005284	-0.02049877	0.014059330	0.010793169
[4,]	-0.06254145	0.05498432	0.01135031	-0.001877983	-0.005490454
[5,]	-0.18058630	0.16151742	0.02723710	0.065483158	0.003326357
[6,]	-0.04075617	0.02900006	-0.21508009	0.171627718	0.019241676
	[,11]				
[1,]	0.03425156				
[2,]	-0.01503250				
[3,]	0.04924166				
[4,]	-0.01254111				
[5,]	-0.05570035				
[6,]	0.05775536				

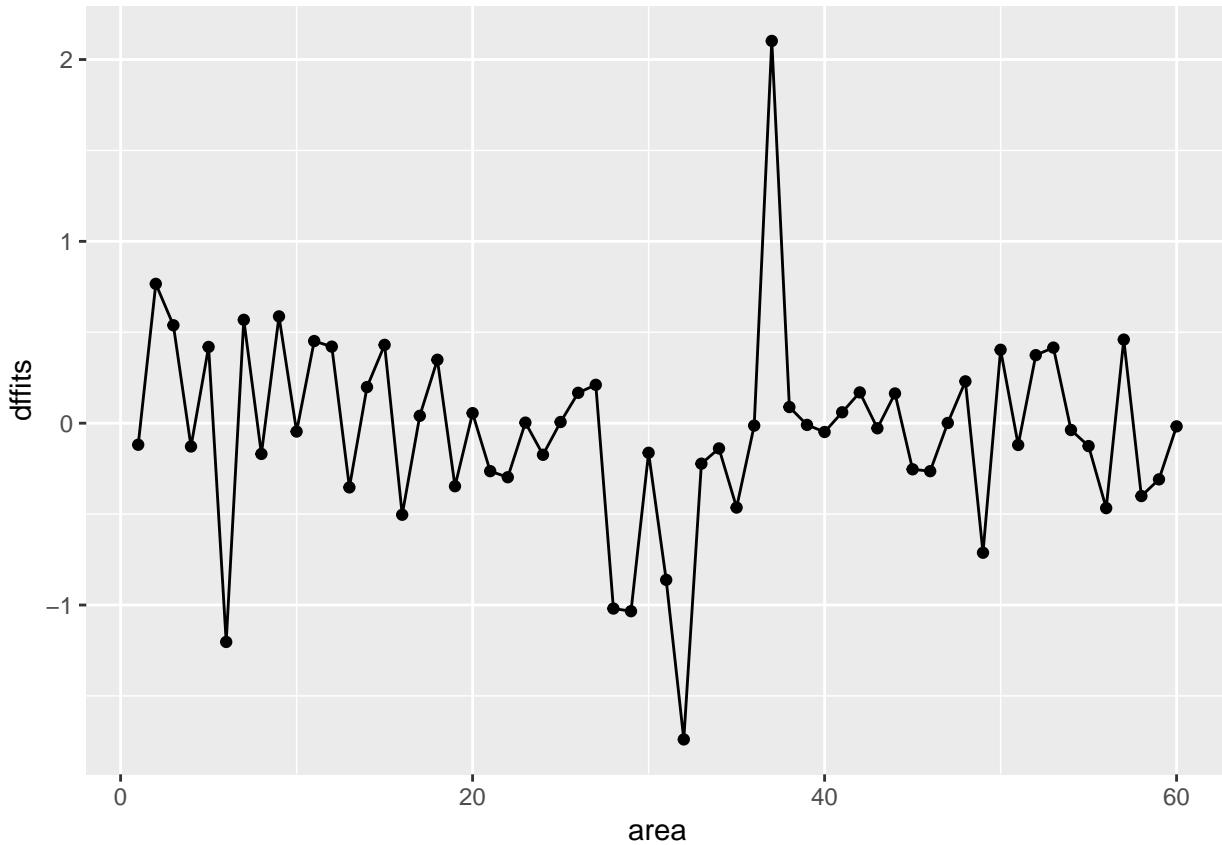
10.5.2 Using the `residuals` command for other summaries

The `residuals` command will also let you get ordinary residuals, leverage values and `dffits` values, which are the normalized differences in predicted values when observations are omitted. See `?residuals.ols` for more details.

```
temp <- data.frame(area = 1:60)
temp$residual <- residuals(newmod, type = "ordinary")
temp$leverage <- residuals(newmod, type = "hat")
temp$dffits <- residuals(newmod, type = "dffits")
tbl_df(temp)
```

```
# A tibble: 60 x 4
  area residual leverage dffits
  <int>    <dbl>     <dbl>    <dbl>
1     1    -13.3    0.0929 -0.119
2     2     81.0    0.0941  0.766
3     3     28.8    0.266   0.539
4     4    -12.5    0.117   -0.128
5     5     27.8    0.204   0.419
6     6    -40.4    0.416   -1.20
7     7     37.0    0.207   0.568
8     8    -14.3    0.145   -0.169
9     9     66.6    0.0863  0.587
10    10    -4.96   0.0997 -0.0460
# ... with 50 more rows
```

```
ggplot(temp, aes(x = area, y = dffits)) +
  geom_point() +
  geom_line()
```



It appears that point 37 has the largest (positive) `dffits` value. Recall that point 37 seemed influential on several predictors and the intercept term. Point 32 has the smallest (or largest negative) `dffits`, and also appears to have been influential on several predictors and the intercept.

```
which.max(temp$dffits)
```

```
[1] 37
```

```
which.min(temp$dffits)
```

```
[1] 32
```

10.6 Model Validation and Correcting for Optimism

In 431, we learned about splitting our regression models into **training** samples and **test** samples, performing variable selection work on the training sample to identify two or three candidate models (perhaps via a stepwise approach), and then comparing the predictions made by those models in a test sample.

At the final project presentations, I mentioned (to many folks) that there was a way to automate this process a bit in 432, that would provide some ways to get the machine to split the data for you multiple times, and then average over the results, using a bootstrap approach. This is it.

The `validate` function allows us to perform cross-validation of our models for some summary statistics (and then correct those statistics for optimism in describing likely predictive accuracy) in an easy way.

`validate` develops:

- Resampling validation with or without backward elimination of variables
- Estimates of the *optimism* in measures of predictive accuracy

- Estimates of the intercept and slope of a calibration model

$$(\text{observed } y) = \text{Intercept} + \text{Slope} (\text{predicted } y)$$

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	0.7500	0.5964	0.1536	0.5452	40
MSE	1145.7800	888.2564	1535.8277	-647.5714	1793.3514	40
g	58.9614	58.9323	55.2085	3.7238	55.2376	40
Intercept	0.0000	0.0000	86.5968	-86.5968	86.5968	40
Slope	1.0000	1.0000	0.9088	0.0912	0.9088	40

So, for R-square we see that our original estimate was 0.6989

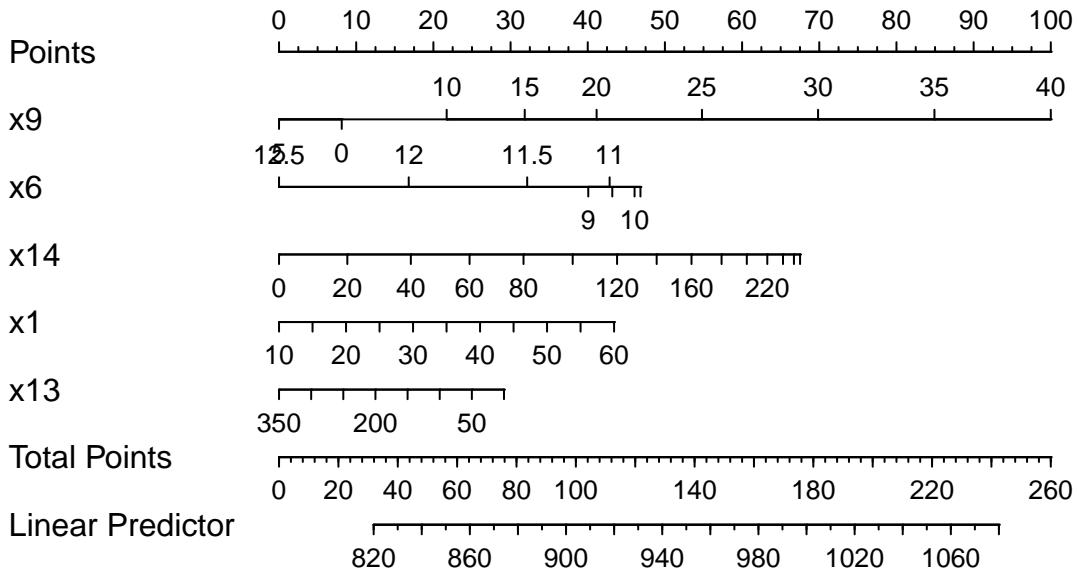
- Our estimated R-square across n = 40 training samples was 0.7500, but in the resulting tests, the average R-square was only 0.5964
- This suggests an optimism of 0.7500 - 0.5964 = 0.1536 (after rounding).
- We then apply that optimism to obtain a new estimate of R² corrected for overfitting, at 0.5452, which is probably a better estimate of what our results might look like in new data that were similar to (but not the same as) the data we used in building newmod than our initial estimate of 0.6989

We also obtain optimism-corrected estimates of the mean squared error (square of the residual standard deviation), the g index, and the intercept and slope of the calibration model. The “corrected” slope is a shrinkage factor that takes overfitting into account.

10.7 Building a Nomogram for Our Model

Another nice feature of an ols model object is that we can picture the model with a nomogram easily. Here is model newmod.

```
plot(nomogram(newmod))
```



For this model, we can use this plot to predict y as follows:

1. find our values of x_9 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 x_6 , x_{14} , x_1 , and x_{13} . 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 x_6 results, for example, which turn around from 9-10 to 11-12. We also see non-linearity's effects in the scales of the non-linear terms in terms of points awarded.

An area with a combination of predictor values leading to a total of 100 points, for instance, would lead to a prediction of a Mortality Rate near 905. An area with a total of 140 points would have a predicted Mortality Rate of 955, roughly.

Chapter 11

Other Variable Selection Strategies

11.1 Why not use stepwise procedures?

1. The R^2 for a model selected in a stepwise manner is biased, high.
2. The coefficient estimates and standard errors are biased.
3. The p values for the individual-variable t tests are too small.
4. In stepwise analyses of prediction models, the final model represented noise 20-74% of the time.
5. In stepwise analyses, the final model usually contained less than half of the actual number of real predictors.
6. It is not logical that a population regression coefficient would be exactly zero just because its estimate was not statistically significant.

This last comment applies to things like our “best subsets” approach as well as standard stepwise procedures.

Sander Greenland’s comments on parsimony and stepwise approaches to model selection are worth addressing...

- Stepwise variable selection on confounders leaves important confounders uncontrolled.
- Shrinkage approaches (like ridge regression and the lasso) are far superior to variable selection.
- Variable selection does more damage to confidence interval widths than to point estimates.

If we are seriously concerned about **overfitting** - winding up with a model that doesn’t perform well on new data - then stepwise approaches generally don’t help.

Vittinghoff et al. (2012) suggest four strategies for minimizing the chance of overfitting

1. Pre-specify well-motivated predictors and how to model them.
2. Eliminate predictors without using the outcome.
3. Use the outcome, but cross-validate the target measure of prediction error.
4. Use the outcome, and **shrink** the coefficient estimates.

The best subsets methods we have studied either include a variable or drop it from the model. Often, this choice is based on only a tiny difference in the quality of a fit to data.

- Harrell (2001): not reasonable to assume that a population regression coefficient would be exactly zero just because it failed to meet a criterion for significance.
- Brad Efron has suggested that a stepwise approach is “overly greedy, impulsively eliminating covariates which are correlated with other covariates.”

So, what’s the alternative?

11.2 Ridge Regression

Ridge regression involves a more smooth transition between useful and not useful predictors which can be obtained by constraining the overall size of the regression coefficients.

Ridge regression assumes that the regression coefficients (after normalization) should not be very large. This is reasonable to assume when you have lots of predictors and you believe *many* of them have some effect on the outcome.

Pros:

1. Some nice statistical properties
2. Can be calculated using only standard least squares approaches, so it's been around for a while.
3. Available in the MASS package.

Ridge regression takes the sum of the squared estimated standardized regression coefficients and constrains that sum to only be as large as some value k .

$$\sum \hat{\beta}_j^2 \leq k.$$

The value k is one of several available measures of the amount of shrinkage, but the main one used in the MASS package is a value λ . As λ increases, the amount of shrinkage goes up, and k goes down.

11.2.1 Assessing a Ridge Regression Approach

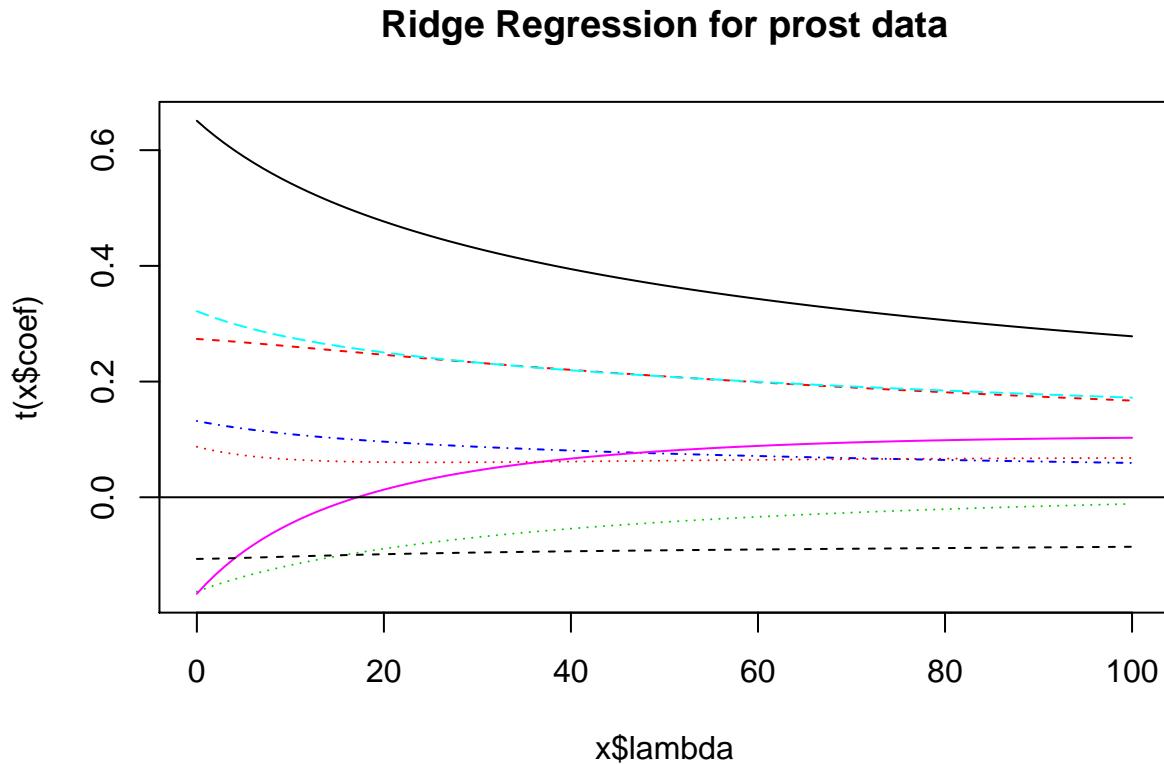
We'll look at a plot produced by the `lm.ridge` function for a ridge regression for the prostate cancer study we worked on when studying Stepwise Regression and Best Subsets methods earlier.

- Several (here 101) different values for λ , our shrinkage parameter, will be tested.
- Results are plotted so that we see the coefficients across the various (standardized) predictors.
 - Each selection of a λ value implies a different vector of covariate values across the predictors we are studying.
 - The idea is to pick a value of λ for which the coefficients seem relatively stable.

```
preds <- with(prost, cbind(lcavol, lweight, age, bph_f,
                           svi_f, lcp, gleason_f, pgg45))

x <- lm.ridge(prost$lpsa ~ preds, lambda = 0:100)

plot(x)
title("Ridge Regression for prost data")
abline(h = 0)
```



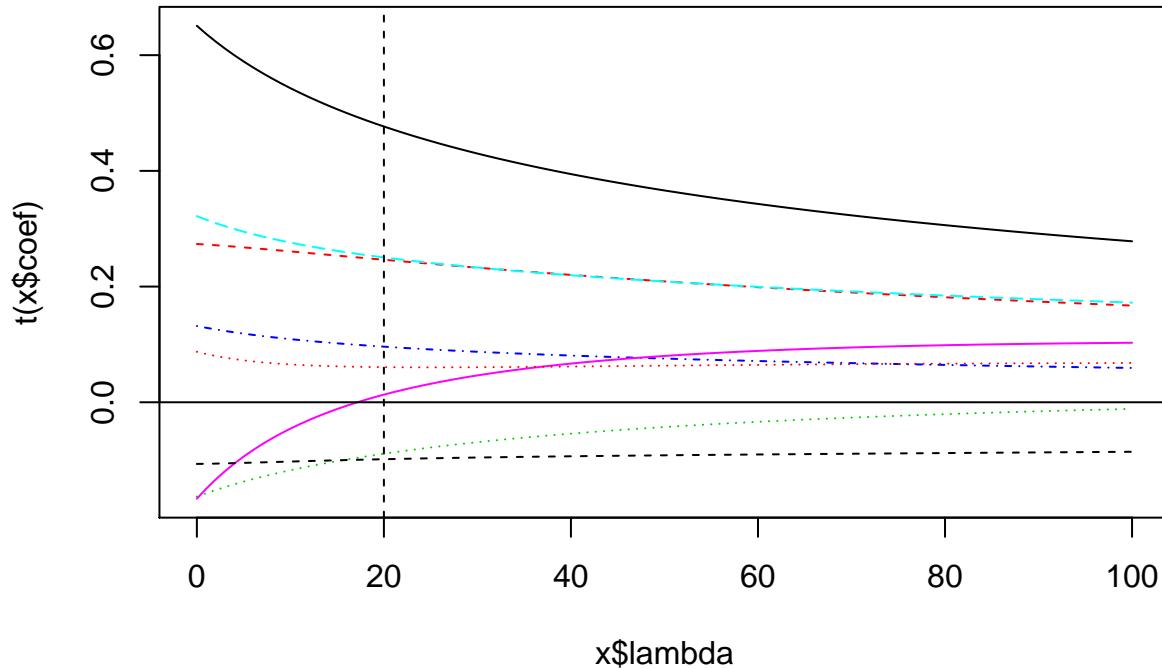
Usually, you need to use trial and error to decide the range of λ to be tested. Here, `0:100` means going from 0 (no shrinkage) to 100 in steps of 1.

11.2.2 The `lm.ridge` plot - where do coefficients stabilize?

Does $\lambda = 20$ seem like a stable spot here?

```
x <- lm.ridge(prost$lpsa ~ preds, lambda = 0:100)
plot(x)
title("Ridge Regression for prost data")
abline(h = 0)
abline(v=20, lty=2, col="black")
```

Ridge Regression for prost data



The coefficients at $\lambda = 20$ can be determined from the `lm.ridge` output. These are fully standardized coefficients. The original predictors are centered by their means and then scaled by their standard deviations and the outcome has also been centered, in these models.

```
round(x$coef[, 20], 3)
```

<code>predslcavol</code>	<code>predslweight</code>	<code>predsage</code>	<code>predsbph_f</code>	<code>predssvi_f</code>
0.482	0.248	-0.091	0.097	0.252
<code>predslcp</code>	<code>predsgleason_f</code>	<code>predspgg45</code>		
0.009	-0.099	0.061		

Was an intercept used?

```
x$Inter
```

```
[1] 1
```

Yes, it was. There is an automated way to pick λ . Use the `select` function in the `MASS` package:

```
MASS:::select(x)
```

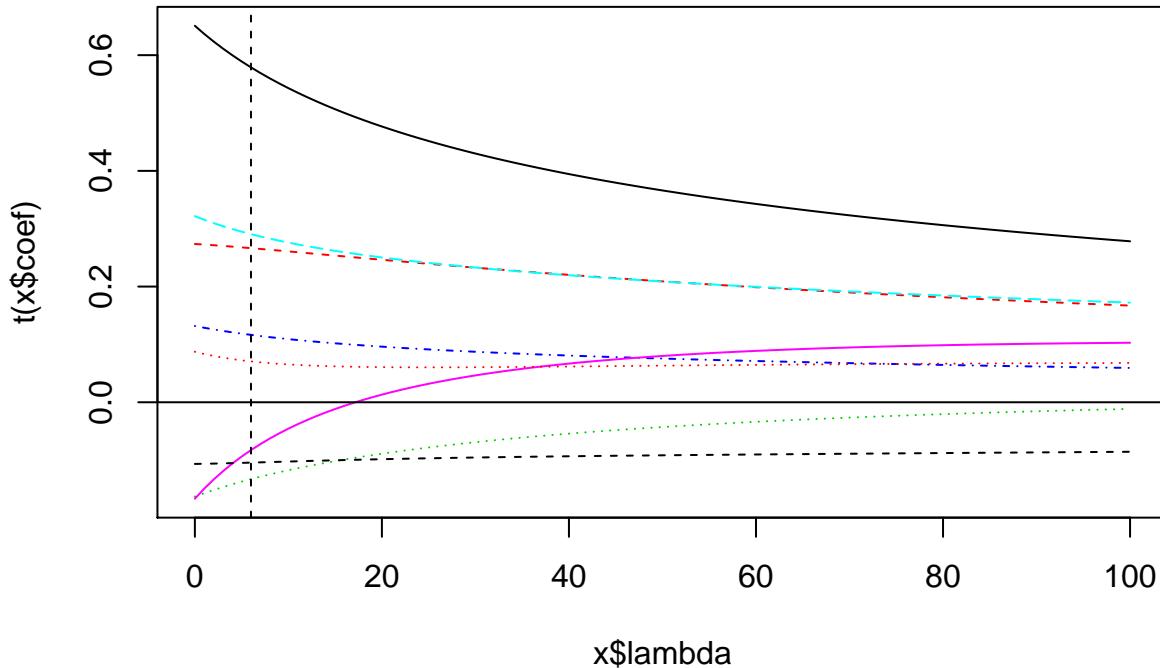
```
modified HKB estimator is 4.210238
modified L-W estimator is 3.32223
smallest value of GCV at 6
```

I'll use the GCV = generalized cross-validation to select $\lambda = 6$ instead.

```
x <- lm.ridge(prost$lpsa ~ preds, lambda = 0:100)
plot(x)
title("Ridge Regression for prost data")
abline(h = 0)
```

```
abline(v=6, lty=2, col="black")
```

Ridge Regression for prost data



```
x$coef [,6]
```

predslcavol	predslweight	predsage	predsbph_f	predssvi_f
0.58911149	0.26773757	-0.13715070	0.11862949	0.29491008
predslcp	predsgleason_f	predspgg45		
-0.09389545	-0.10477578	0.07250609		

11.2.3 Ridge Regression: The Bottom Line

The main problem with ridge regression is that all it does is shrink the coefficient estimates, but it's not so useful in practical settings because it still includes all variables.

1. It's been easy to do ridge regression for many years, so you see it occasionally in the literature.
2. It leads to the **lasso**, which incorporates the positive features of shrinking regression coefficients with the ability to wisely select some variables to be eliminated from the predictor pool.

11.3 The Lasso

The lasso works by takes the sum of the absolute values of the estimated standardized regression coefficients and constrains it to only be as large as some value k.

$$\sum |\hat{\beta}_j| \leq k.$$

This looks like a minor change, but it's not.

11.3.1 Consequences of the Lasso Approach

1. In ridge regression, while the individual coefficients shrink and sometimes approach zero, they seldom reach zero and are thus excluded from the model. With the lasso, some coefficients do reach zero and thus, those predictors do drop out of the model.
 - So the lasso leads to more parsimonious models than does ridge regression.
 - Ridge regression is a method of shrinkage but not model selection. The lasso accomplishes both tasks.
2. If k is chosen to be too small, then the model may not capture important characteristics of the data. If k is too large, the model may over-fit the data in the sample and thus not represent the population of interest accurately.
3. The lasso is far more difficult computationally than ridge regression (the problem requires an algorithm called least angle regression published in 2004), although R has a library (`lars`) which can do the calculations pretty efficiently.

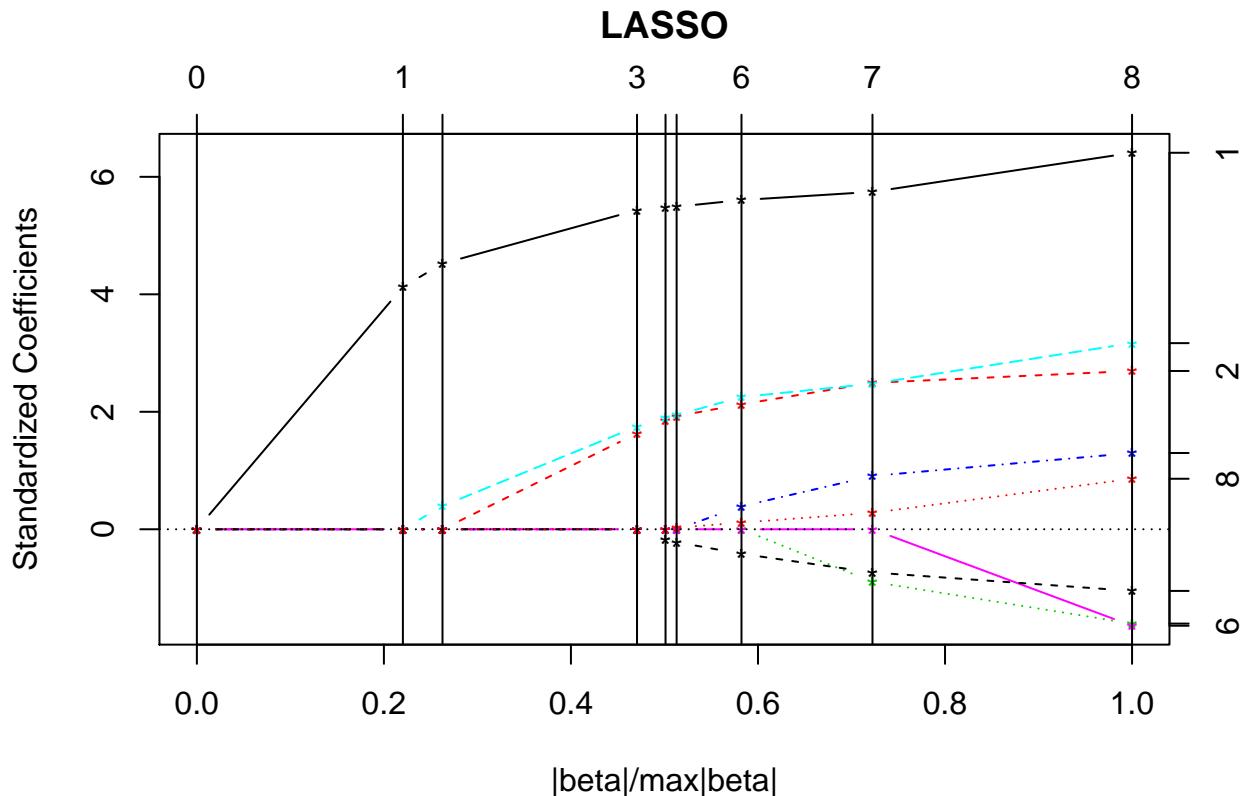
The lasso is not an acronym, but rather refers to cowboys using a rope to pull cattle from the herd, much as we will pull predictors from a model.

11.3.2 How The Lasso Works

The `lars` package lets us compute the lasso coefficient estimates **and** do cross-validation to determine the appropriate amount of shrinkage. The main tool is a pair of graphs.

1. The first plot shows what coefficients get selected as we move from constraining all of the coefficients to zero (complete shrinkage) towards fewer constraints all the way up to ordinary least squares, showing which variables are included in the model at each point.
2. The second plot suggests where on the first plot we should look for a good model choice, according to a cross-validation approach.

```
## requires lars package
lasso1 <- lars(preds, prost$lpsa, type="lasso")
plot(lasso1)
```



- The y axis shows standardized regression coefficients.
 - The `lars` package standardizes all variables so the shrinkage doesn't penalize some coefficients because of their scale.
- The x-axis is labeled $|\beta|/\max|\beta|$.
 - This ranges from 0 to 1.
 - 0 means that the sum of the $|\hat{\beta}_j|$ is zero (completely shrunk)
 - 1 means the ordinary least squares unbiased estimates.

The lasso graph starts at constraining all of the coefficients to zero, and then moves toward ordinary least squares.

Identifiers for the predictors (numbers) are shown to the right of the graph.

The vertical lines in the lasso plot show when a variable has been eliminated from the model, and in fact these are the only points that are actually shown in the default lasso graph. The labels on the top of the graph tell you how many predictors are in the model at that stage.

```
summary(lasso1)
```

```
LARS/LASSO
Call: lars(x = preds, y = prost$lpса, type = "lasso")
   Df    Rss     Cp
0  1 127.918 168.1835
1  2  76.392  64.1722
2  3  70.247  53.5293
3  4  50.598  15.1017
4  5  49.065  13.9485
5  6  48.550  14.8898
6  7  46.284  12.2276
```

```
7 8 44.002 9.5308
8 9 42.772 9.0000
```

Based on the C_p statistics, it looks like the improvements continue throughout, and don't really finish happening until we get pretty close to the full model with 9 df.

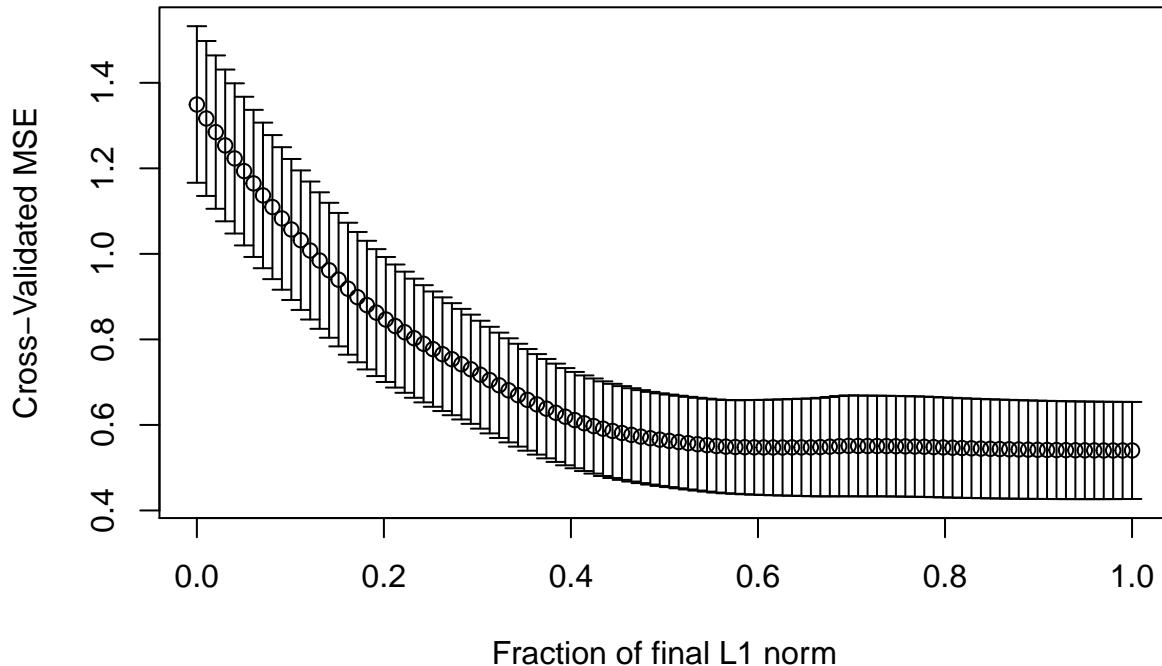
11.3.3 Cross-Validation with the Lasso

Normally, cross-validation methods are used to determine how much shrinkage should be used. We'll use the `cv.lars` function.

- 10-fold ($K = 10$) cross-validation
 - the data are randomly divided into 10 groups.
 - Nine groups are used to predict the remaining group for each group in turn.
 - Overall prediction performance is computed, and the machine calculates a cross-validation criterion (mean squared error) and standard error for that criterion.

The cross-validation plot is the second lasso plot.

```
set.seed(432)
lassocv <- cv.lars(preds, prost$lpsa, K=10)
```



```
## default cv.lars K is 10
```

We're looking to minimize cross-validated mean squared error in this plot, which doesn't seem to happen until the fraction gets very close to 1.

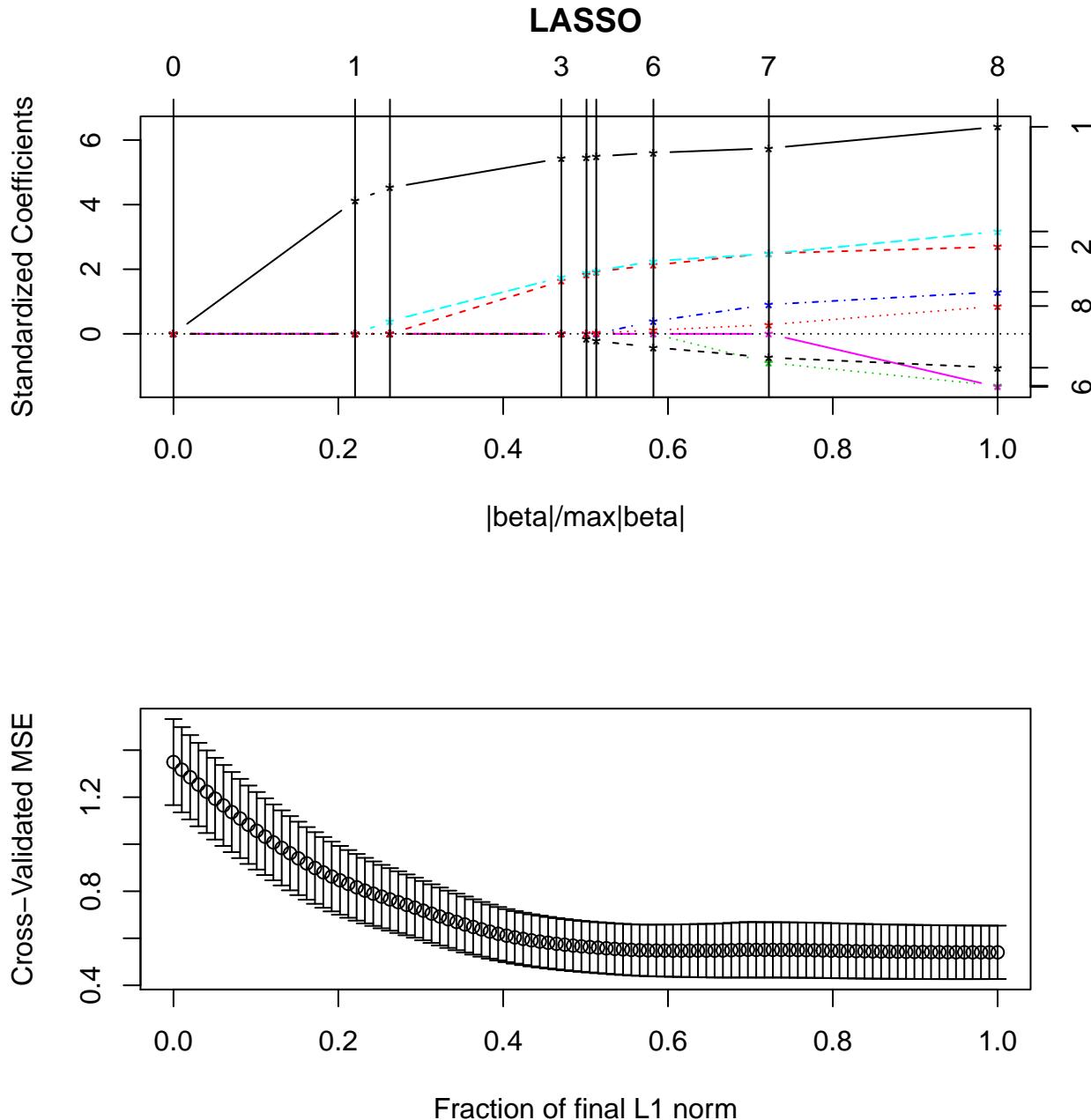
11.3.4 What value of the key fraction minimizes cross-validated MSE?

```
frac <- lassocv$index[which.min(lassocv$cv)]  
frac
```

```
[1] 0.989899
```

The cross-validation plot suggests we use a fraction of nearly 1.0, suggesting that all of the predictors will be kept in, based on the top LASSO plot.

```
par(mfrow=c(2,1))  
lasso1 <- lars(preds, prost$lpsa, type="lasso")  
plot(lasso1)  
set.seed(432)  
lassocv <- cv.lars(preds, prost$lpsa, K=10)
```



```
par(mfrow=c(1,1))
```

11.3.5 Coefficients for the Model Identified by the Cross-Validation

```
coef.cv <- coef(lasso1, s=frac, mode="fraction")
round(coef.cv,4)
```

lcavol	lweight	age	bph_f	svi_f	lcp	gleason_f
0.5529	0.6402	-0.0217	0.1535	0.7750	-0.1155	-0.1826
pgg45						

```
0.0030
```

So the model suggested by the lasso still includes all eight of these predictors.

11.3.6 Obtaining Fitted Values from Lasso

```
fits.cv <- predict.lars(lasso1, preds, s=frac,
                        type="fit", mode="fraction")
fits.cv

$s
[1] 0.989899

$fraction
[1] 0.989899

$mode
[1] "fraction"

$fit
[1] 0.7995838 0.7493971 0.5111634 0.6098520 1.7001847 0.8338020 1.8288518
[8] 2.1302316 1.2487955 1.2661752 1.4704969 0.7782005 2.0755860 1.9129272
[15] 2.1533975 1.8124981 1.2713610 2.3993624 1.3232566 1.7709029 1.9757841
[22] 2.7451649 1.1658326 2.4825521 1.8036338 1.9112578 2.0144298 1.7829219
[29] 1.9706111 2.1688199 2.0377131 1.8657882 1.6955904 1.3580186 1.0516394
[36] 2.9097450 2.1898622 1.0454123 3.8896481 1.7971270 2.0932871 2.3253395
[43] 2.0809295 2.5303655 2.4451523 2.5827203 4.0692397 2.6845105 2.7034959
[50] 1.9590266 2.4522082 2.9801227 2.1902084 3.0559124 3.3447025 2.9765233
[57] 1.7620182 2.3424646 2.2856404 2.6188548 2.3056410 3.5568662 2.9756755
[64] 3.6764122 2.5097586 2.6579014 2.9482717 3.0892917 1.5113015 3.0282296
[71] 3.2887119 2.1083273 2.8889223 3.4903026 3.6959516 3.6070031 3.2749993
[78] 3.4518575 3.4049180 3.1814731 2.0496216 2.8986175 3.6743113 3.3292860
[85] 2.6965297 3.8339856 2.9892543 3.0555536 4.2903885 3.0986508 3.3784385
[92] 4.0205201 3.8309974 4.7531590 3.6290575 4.1347645 4.0982744
```

11.3.7 Complete Set of Fitted Values from the Lasso

```
round(fits.cv$fit,3)

[1] 0.800 0.749 0.511 0.610 1.700 0.834 1.829 2.130 1.249 1.266 1.470
[12] 0.778 2.076 1.913 2.153 1.812 1.271 2.399 1.323 1.771 1.976 2.745
[23] 1.166 2.483 1.804 1.911 2.014 1.783 1.971 2.169 2.038 1.866 1.696
[34] 1.358 1.052 2.910 2.190 1.045 3.890 1.797 2.093 2.325 2.081 2.530
[45] 2.445 2.583 4.069 2.685 2.703 1.959 2.452 2.980 2.190 3.056 3.345
[56] 2.977 1.762 2.342 2.286 2.619 2.306 3.557 2.976 3.676 2.510 2.658
[67] 2.948 3.089 1.511 3.028 3.289 2.108 2.889 3.490 3.696 3.607 3.275
[78] 3.452 3.405 3.181 2.050 2.899 3.674 3.329 2.697 3.834 2.989 3.056
[89] 4.290 3.099 3.378 4.021 3.831 4.753 3.629 4.135 4.098
```

To assess the quality of these predictions, we might plot them against the observed values of our outcome (`lpsa`), or we might look at residuals vs. these fitted values.

```
prost_lasso_res <- data_frame(fitted = fits.cv$fit,
                               actual = prost$lpsa,
```

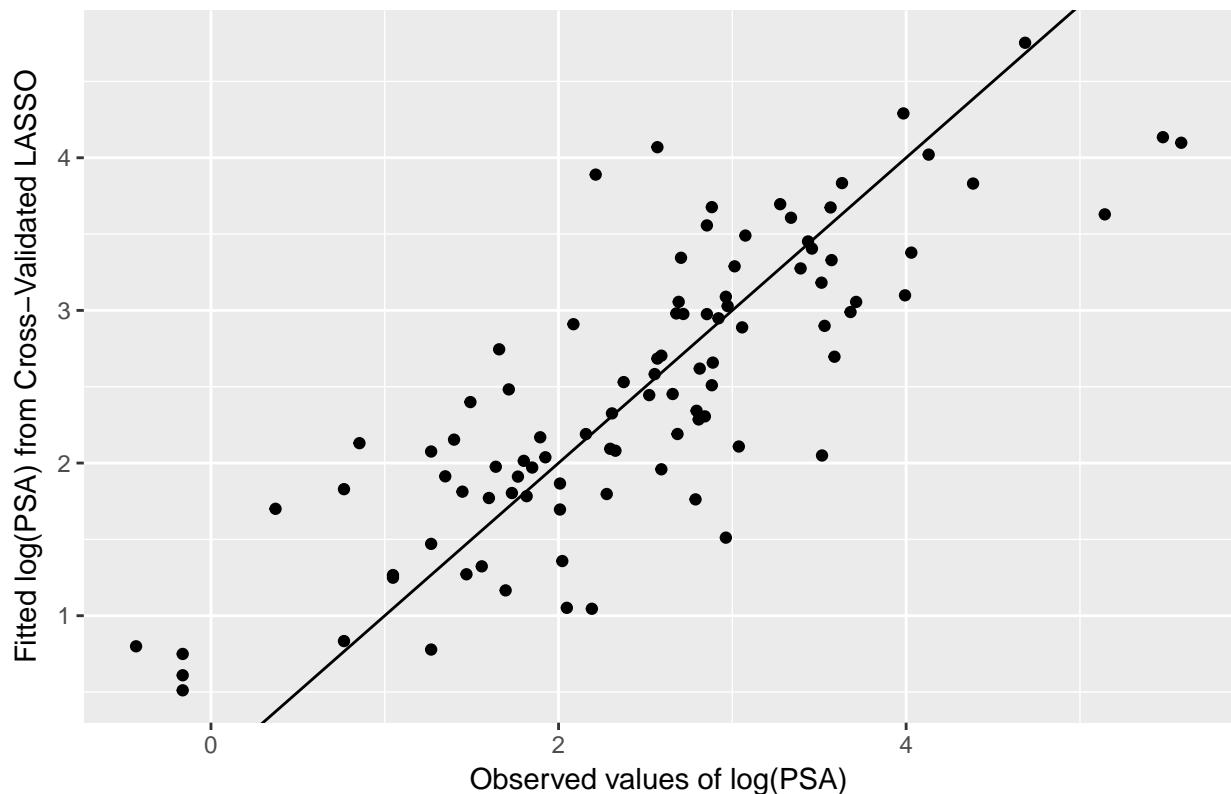
```

        resid = actual - fitted)

ggplot(prost_lasso_res, aes(x = actual, y = fitted)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0) +
  labs(y = "Fitted log(PSA) from Cross-Validated LASSO",
       x = "Observed values of log(PSA)",
       title = "Fitted vs. Actual Values of log(PSA)")

```

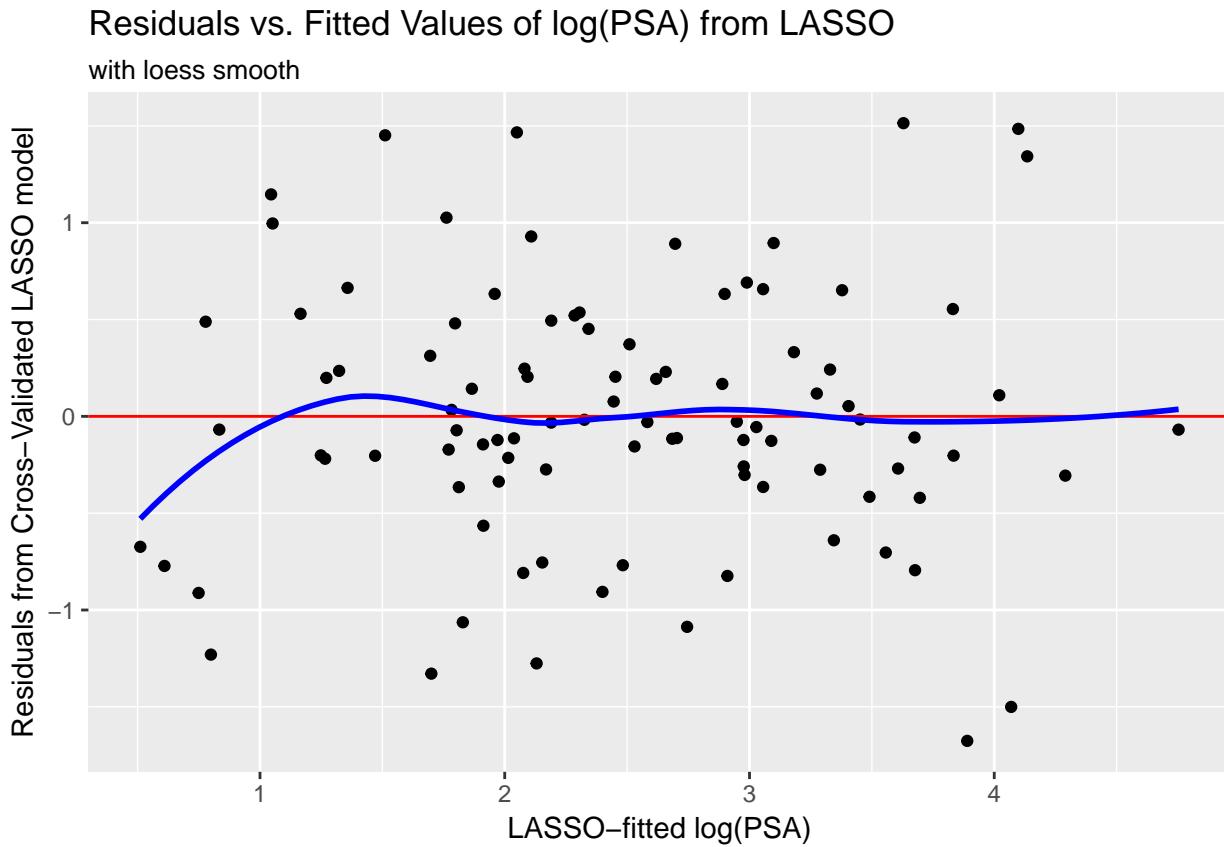
Fitted vs. Actual Values of log(PSA)



```

ggplot(prost_lasso_res, aes(x = fitted, y = resid)) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red") +
  geom_smooth(method = "loess", col = "blue", se = F) +
  labs(x = "LASSO-fitted log(PSA)",
       y = "Residuals from Cross-Validated LASSO model",
       title = "Residuals vs. Fitted Values of log(PSA) from LASSO",
       subtitle = "with loess smooth")

```



11.3.8 When is the Lasso Most Useful?

As Faraway (2015) suggests, the lasso is particularly useful when we believe the effects are sparse, in the sense that we believe that few of the many predictors we are evaluating have a meaningful effect.

Consider, for instance, the analysis of gene expression data, where we have good reason to believe that only a small number of genes have an influence on our response of interest.

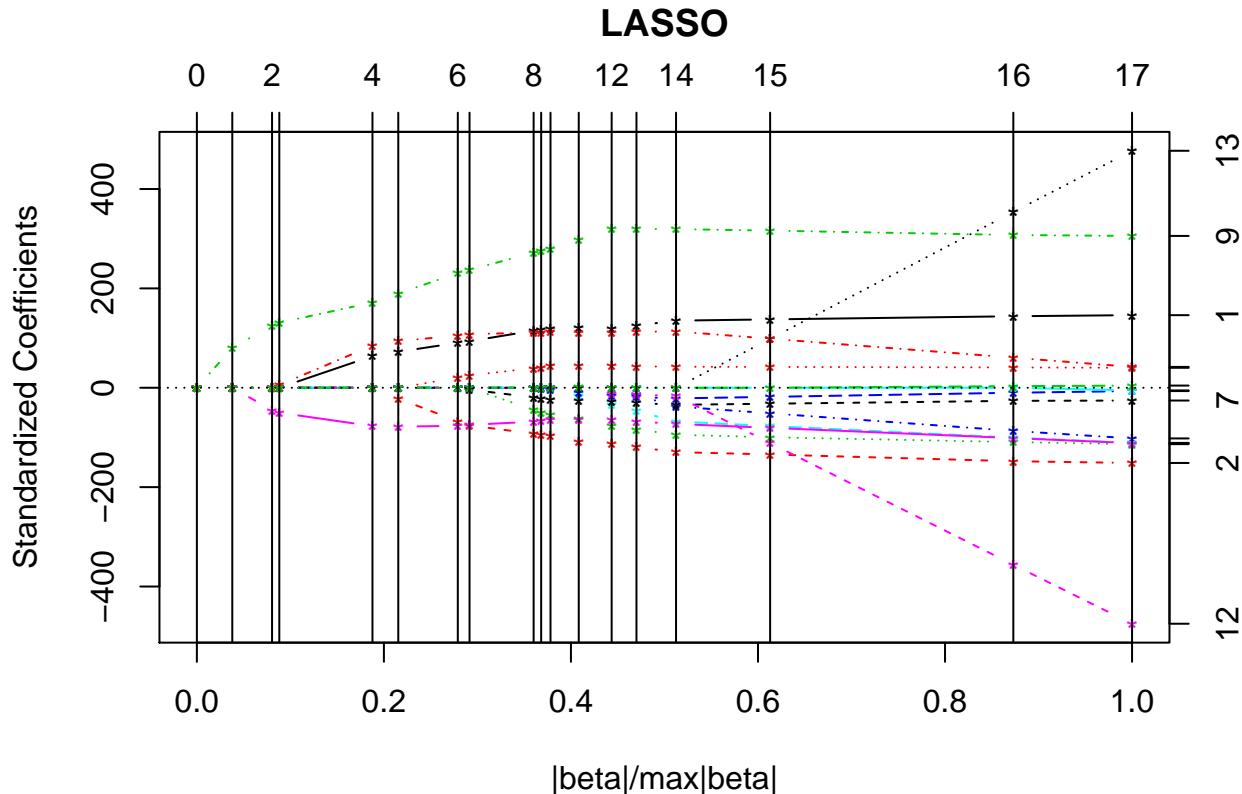
Or, in medical claims data, where we can have thousands of available codes to search through that may apply to some of the people included in a large analysis relating health care costs to outcomes.

11.4 Applying the Lasso to the pollution data

Let's consider the lasso approach in application to the pollution data we've seen previously. Recall that we have 60 observations on an outcome, y , and 15 predictors, labeled x_1 through x_{15} .

```
preds <- with(pollution, cbind(x1, x2, x3, x4, x5, x6, x7,
                                x8, x9, x10, x11, x12, x13,
                                x14, x15))

lasso_p1 <- lars(preds, pollution$y, type="lasso")
plot(lasso_p1)
```

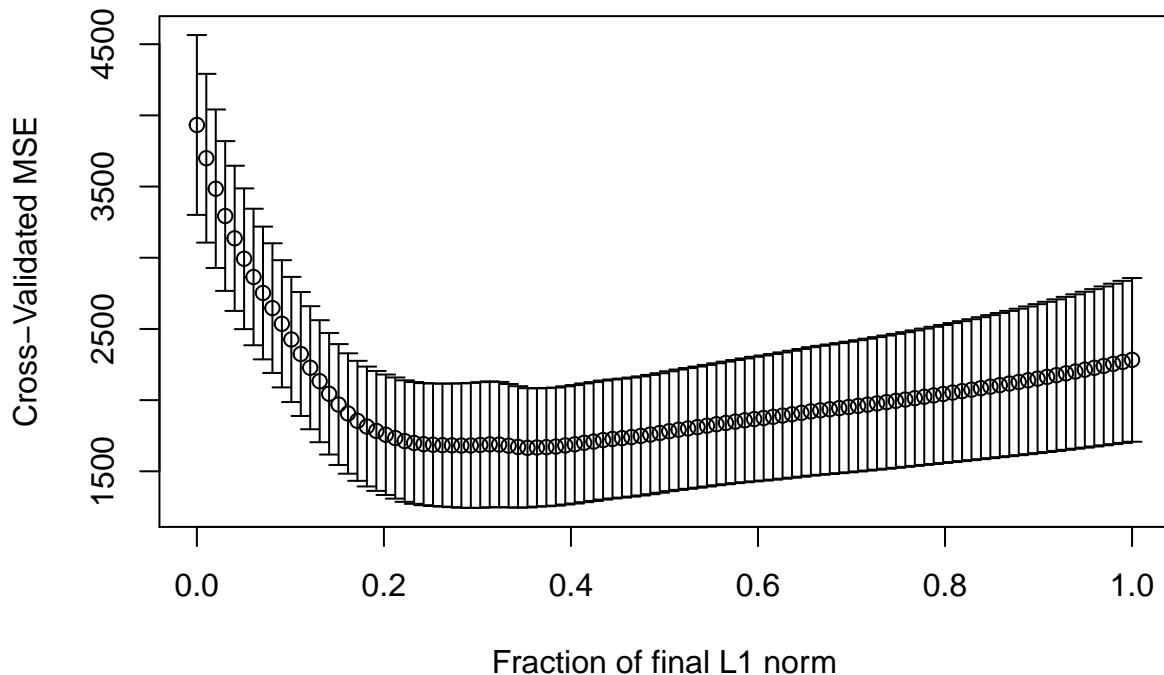


```
summary(lasso_p1)
```

```
LARS/LASSO
Call: lars(x = preds, y = pollution$y, type = "lasso")
      Df    Rss      Cp
0     1 228311 129.1367
1     2 185419  95.9802
2     3 149370  68.4323
3     4 143812  65.8764
4     5  92077  25.4713
5     6  83531  20.4668
6     7  69532  10.9922
7     8  67682  11.4760
8     9  60689  7.7445
9    10  60167  9.3163
10   11  59609  10.8588
11   12  58287  11.7757
12   13  57266  12.9383
13   14  56744  14.5107
14   13  56159  12.0311
15   14  55238  13.2765
16   15  53847  14.1361
17   16  53681  16.0000
```

Based on the C_p statistics, it looks like the big improvements occur somewhere around the move from 6 to 7 df. Let's look at the cross-validation

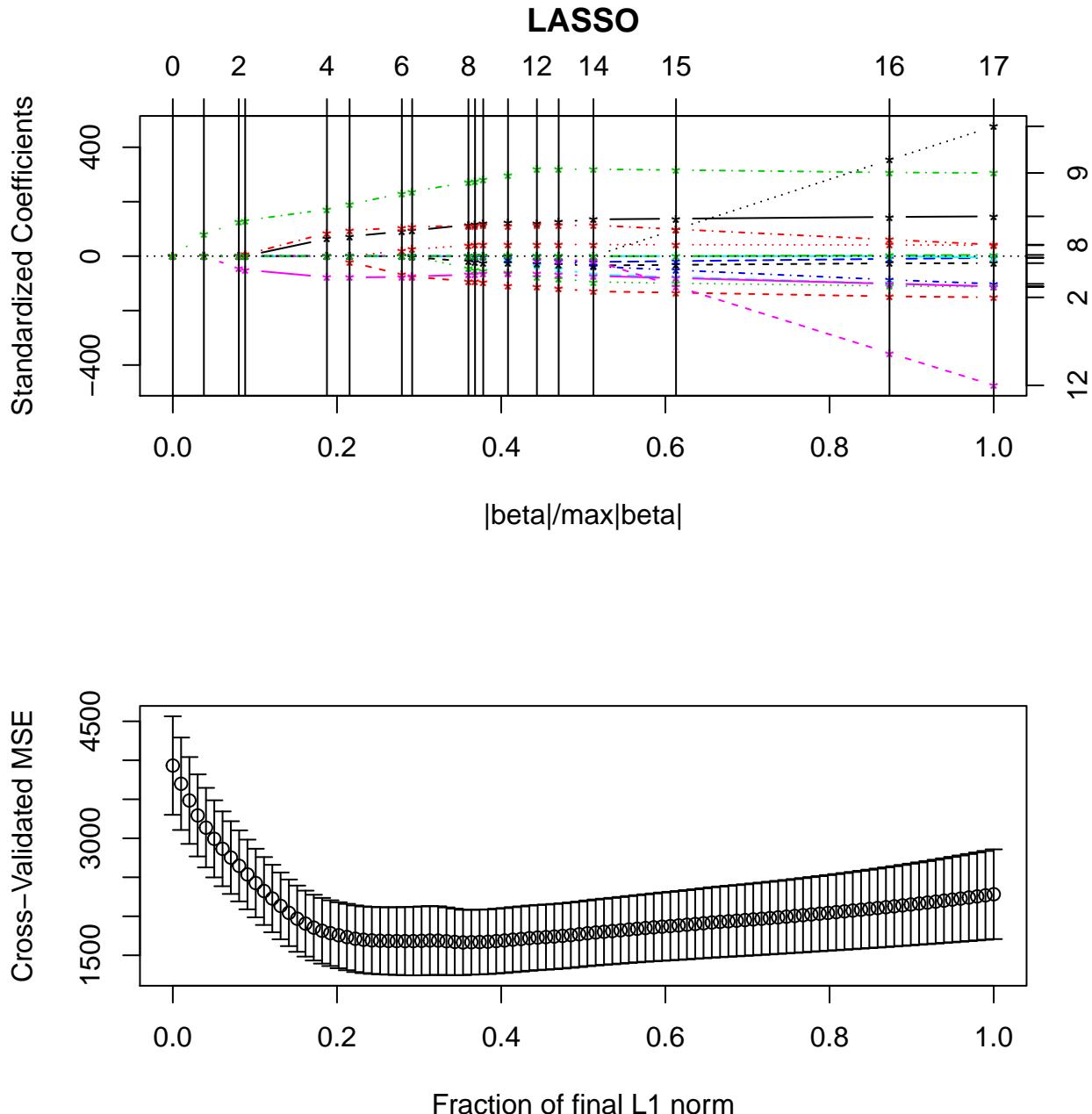
```
set.seed(432012)
pollution_lassocv <- cv.lars(preds, pollution$y, K=10)
```



Here it looks like cross-validated MSE happens somewhere between a fraction of 0.2 and 0.4.

```
frac <- pollution_lassocv$index[which.min(pollution_lassocv$cv)]
frac
```

```
[1] 0.3535354
par(mfrow=c(2,1))
lasso_p1 <- lars(preds, pollution$y, type="lasso")
plot(lasso_p1)
set.seed(432012)
pollution_lassocv <- cv.lars(preds, pollution$y, K=10)
```



```
par(mfrow=c(1,1))
```

It looks like a model with 6-8 predictors will be the most useful. The cross-validated coefficients are as follows:

```
poll.cv <- coef(lasso_p1, s=frac, mode="fraction")
round(poll.cv, 3)
```

x1	x2	x3	x4	x5	x6	x7	x8	x9
1.471	-1.164	-1.102	0.000	0.000	-10.610	-0.457	0.003	3.918
x10	x11	x12	x13	x14	x15			
0.000	0.000	0.000	0.000	0.228	0.000			

Note that by this cross-validated lasso selection, not only are the coefficients for the 8 variables remaining in the model shrunken, but variables `x4`, `x5`, `x10`, `x11`, `x12`, `x13` and `x15` are all dropped from the model, and model `x8` almost is, as well.

```
poll_fits <- predict.lars(lasso_p1, preds, s=frac,
                           type="fit", mode="fraction")
round(poll_fits$fit, 3)
```

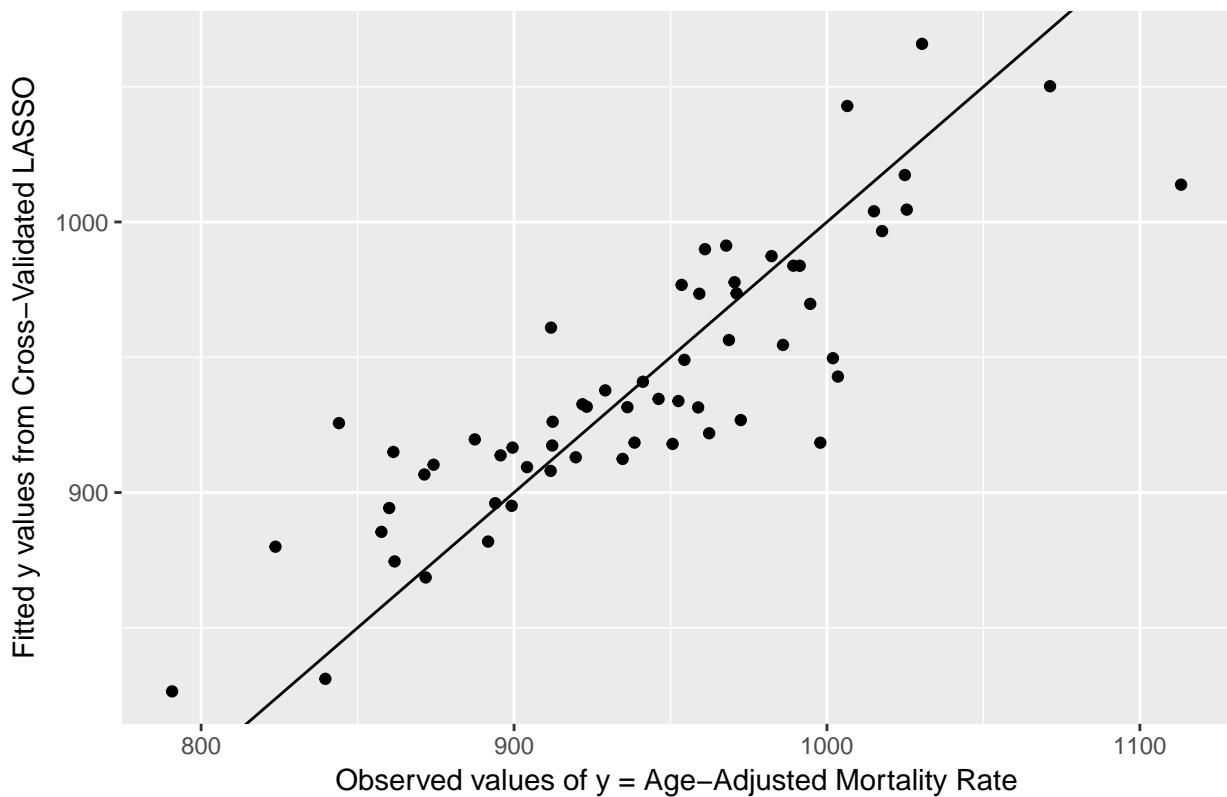
```
[1]  932.627  918.415  921.904  987.396 1050.184 1065.837  912.424
[8]  916.605  949.647  926.168  996.625 1017.362  977.730  954.550
[15]  931.455  894.263  931.551  868.599  973.471  940.937  881.867
[22]  906.666  973.609  919.640  933.821  956.352  913.018  925.650
[29]  874.528  983.829 1042.870  915.002  937.760  885.464  989.947
[36]  931.709 1013.795  969.729 1003.962  983.813  896.042  918.446
[43]  934.609 1004.565  910.273  976.747  831.132  907.996  826.485
[50]  895.082  909.398  917.969  926.777  917.381  991.266  879.972
[57]  942.867  913.737  960.952  949.030
```

Here's a plot of the actual `pollution` `y` values, against these fitted values.

```
poll_lasso_res <- data_frame(fitted = poll_fits$fit,
                               actual = pollution$y,
                               resid = actual - fitted)

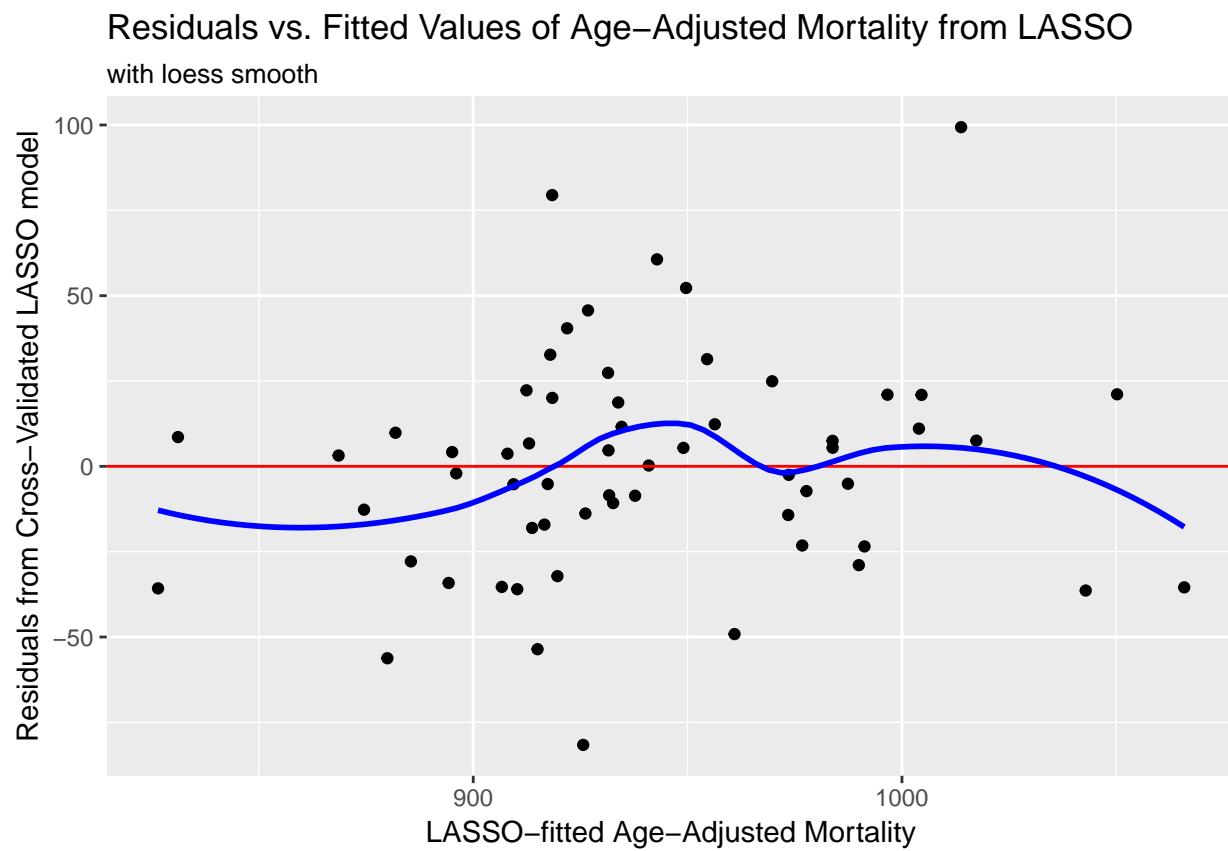
ggplot(poll_lasso_res, aes(x = actual, y = fitted)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0) +
  labs(y = "Fitted y values from Cross-Validated LASSO",
       x = "Observed values of y = Age-Adjusted Mortality Rate",
       title = "Fitted vs. Actual Values of Age-Adjusted Mortality")
```

Fitted vs. Actual Values of Age-Adjusted Mortality



And now, here's a plot of residuals vs. fitted values.

```
ggplot(poll_lasso_res, aes(x = fitted, y = resid)) +
  geom_point() +
  geom_hline(yintercept = 0, col = "red") +
  geom_smooth(method = "loess", col = "blue", se = F) +
  labs(x = "LASSO-fitted Age-Adjusted Mortality",
       y = "Residuals from Cross-Validated LASSO model",
       title = "Residuals vs. Fitted Values of Age-Adjusted Mortality from LASSO",
       subtitle = "with loess smooth")
```



Chapter 12

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).

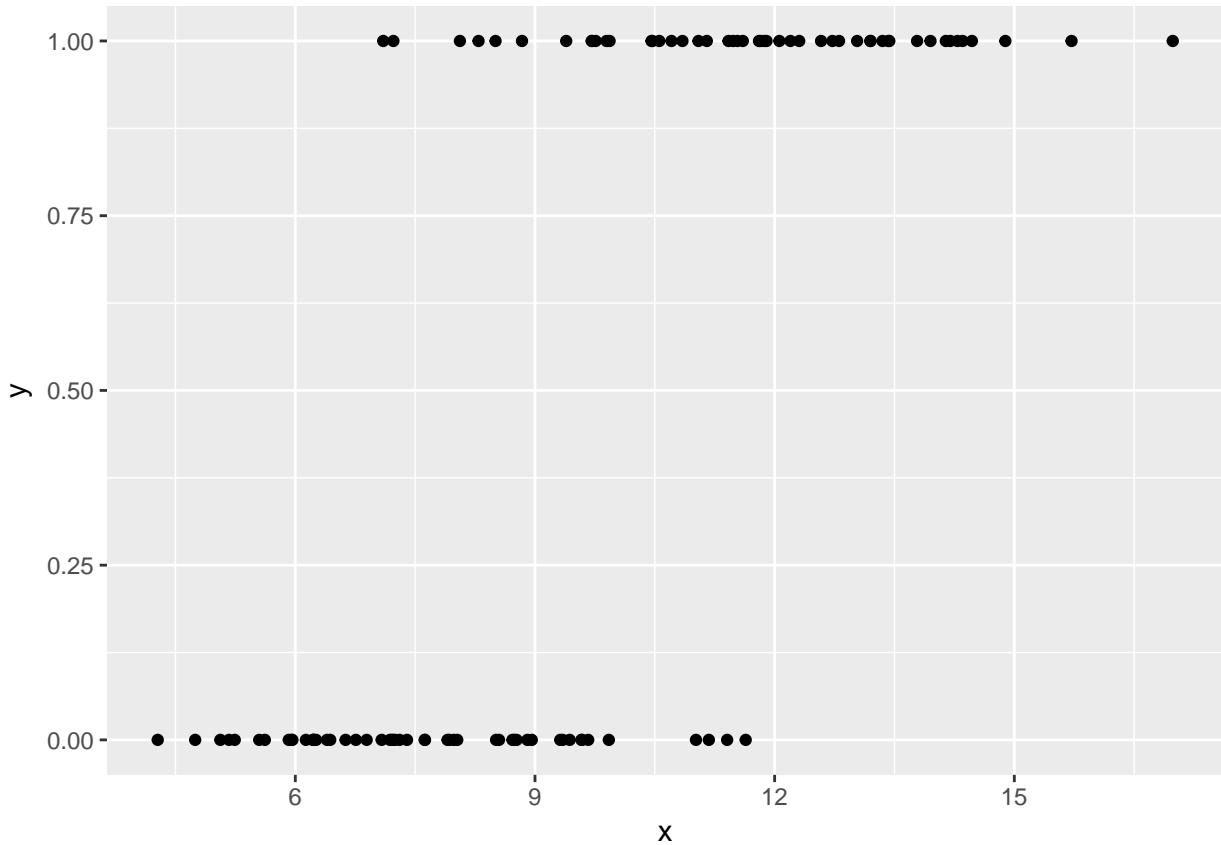
12.1 A First Attempt: A Linear Probability Model

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))

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:

```
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:

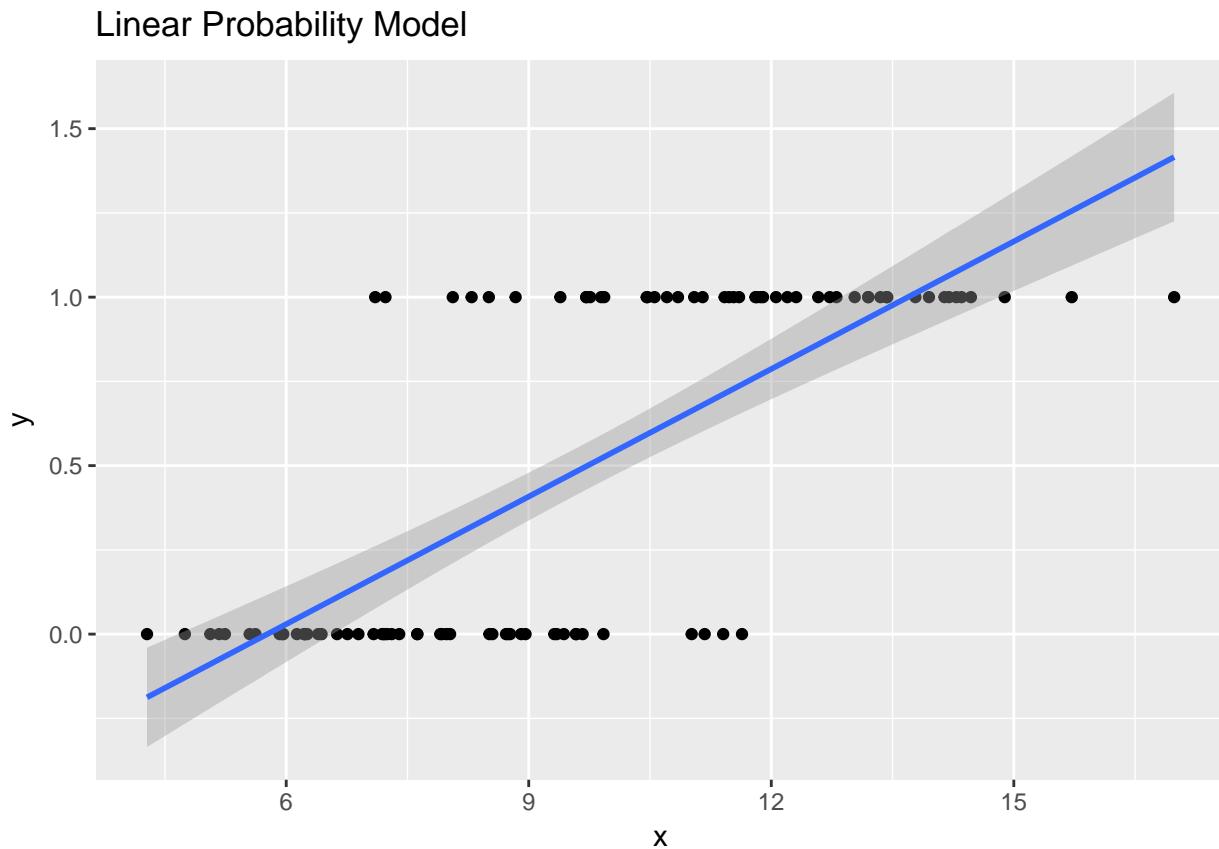
	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.72761	0.12272	-5.929	4.57e-08 ***
x	0.12620	0.01219	10.349	< 2e-16 ***

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?

```
ggplot(sim12, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Linear Probability Model")
```



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 $(0, 1)$?

12.2 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.

12.3 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, \dots, X_k .

$$g(\mu) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k$$

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.

12.4 The Link Function

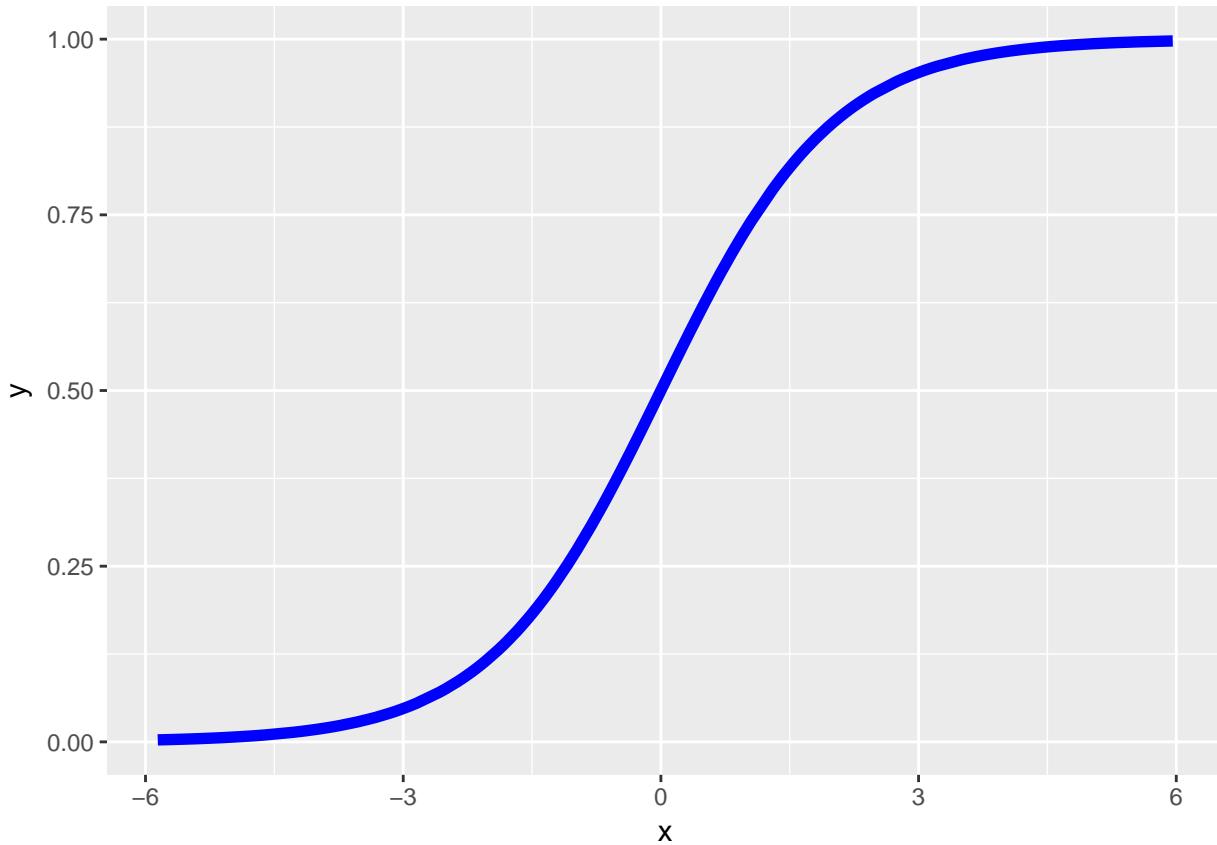
Logistic regression is a non-linear regression approach, since the equation for the mean of the 0/1 Y values conditioned on the values of our predictors X_1, X_2, \dots, X_k turns out to be non-linear in the β coefficients. Its nonlinearity, however, is solely found in its link function, hence the term *generalized* linear model.

The particular link function we use in logistic regression is called the **logit link**.

$$\text{logit}(\pi) = \log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k$$

The inverse of the logit function is called the **logistic function**. If $\text{logit}(\pi) = \eta$, 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.

12.5 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 $\text{logit}(\pi) < 0$.
- If our probability $\pi > 0.5$, then $\text{logit}(\pi) > 0$.
- Finally, if $\pi = 0.5$, then $\text{logit}(\pi) = 0$.

12.6 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 $\pi = \Pr(y = 1)$. Our model holds that:

$$\text{logit}(\pi) = \log \left(\frac{\pi}{1 - \pi} \right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k$$

The odds of a yes response (the odds that $Y = 1$) at the level X_1, X_2, \dots, X_k are:

$$Odds(Y = 1) = \exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k)$$

The **probability** of a yes response ($\Pr(y = 1)$, or π) is just

$$\pi = Pr(Y = 1) = \frac{Odds(Y = 1)}{1 + Odds(Y = 1)} = \frac{\exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k)}{1 + \exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k)}$$

12.7 The Logistic Regression has non-constant variance

In ordinary least squares regression, the variance $Var(Y|X_1, X_2, \dots, X_k) = \sigma^2$ 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.

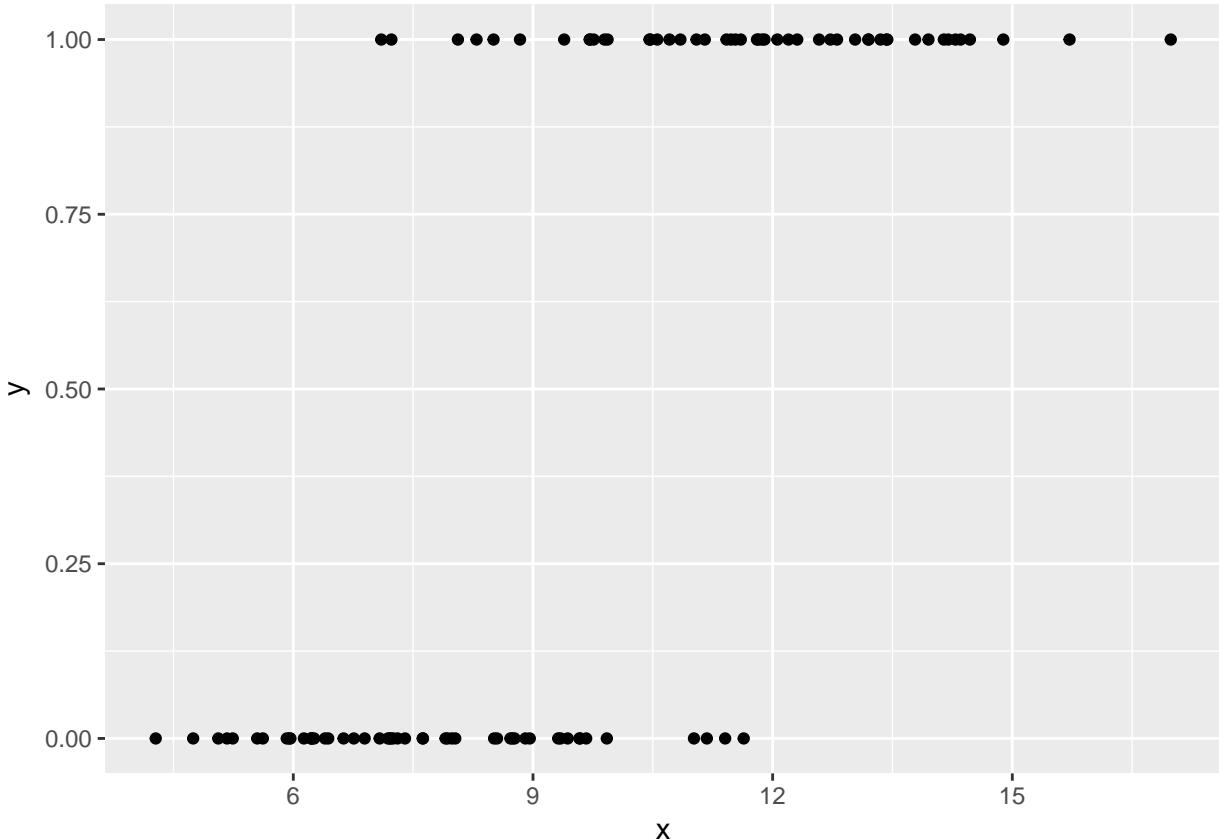
$$logit(\pi) = \log\left(\frac{\pi}{1 - \pi}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k \mu[Y|X_1, \dots, X_k] = \pi, Var[Y|X_1, \dots, X_k] = \pi(1 - \pi)$$

The variance is now a function of the mean, and contains no additional parameter for us to estimate.

12.8 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:

$$\text{logit}(\Pr(y = 1)) = \log\left(\frac{\Pr(y = 1)}{1 - \Pr(y = 1)}\right) = -9.1955 + 0.9566x$$

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 $x = 12$.

$$\text{logit}(\Pr(y = 1)|X = 12) = \log\left(\frac{\Pr(y = 1)}{1 - \Pr(y = 1)}\right) = -9.1955 + 0.9566 * 12 = 2.2837$$

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
```

$$\text{Odds}(Y = 1|X = 12) = \exp(-9.20 + 0.96 * 12) = \exp(2.2837) = 9.812921$$

```
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

$$\pi = \Pr(Y = 1|X = 12) = \frac{\text{Odds}(Y = 1|X = 12)}{1 + \text{Odds}(Y = 1|X = 12)} = \frac{\exp(-9.20 + 0.96x)}{1 + \exp(-9.20 + 0.96x)} = \frac{9.812921}{1 + 9.812921} = 0.908$$

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")
```

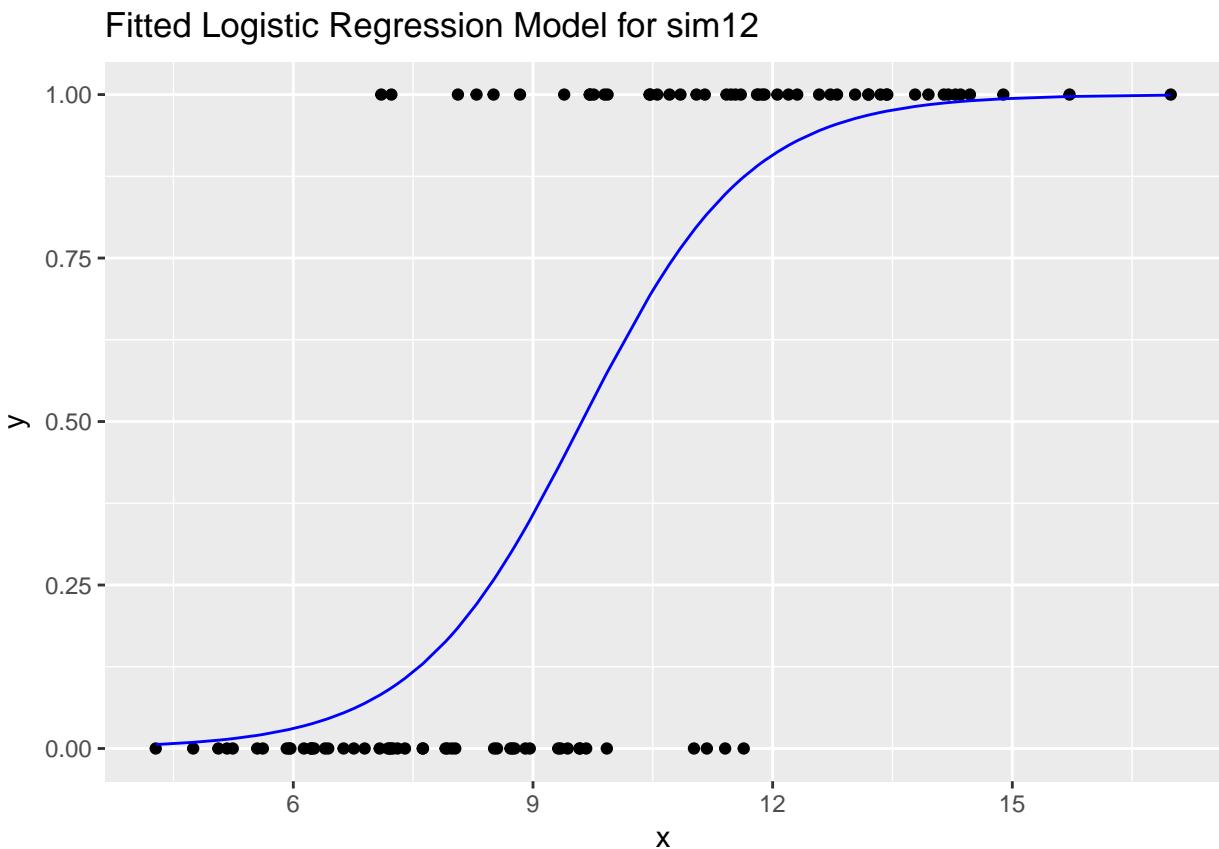
```
1  
0.907549
```

12.9 Plotting the Logistic Regression Model

We can use the `augment` function from the `broom` package to get our fitted probabilities included in the data.

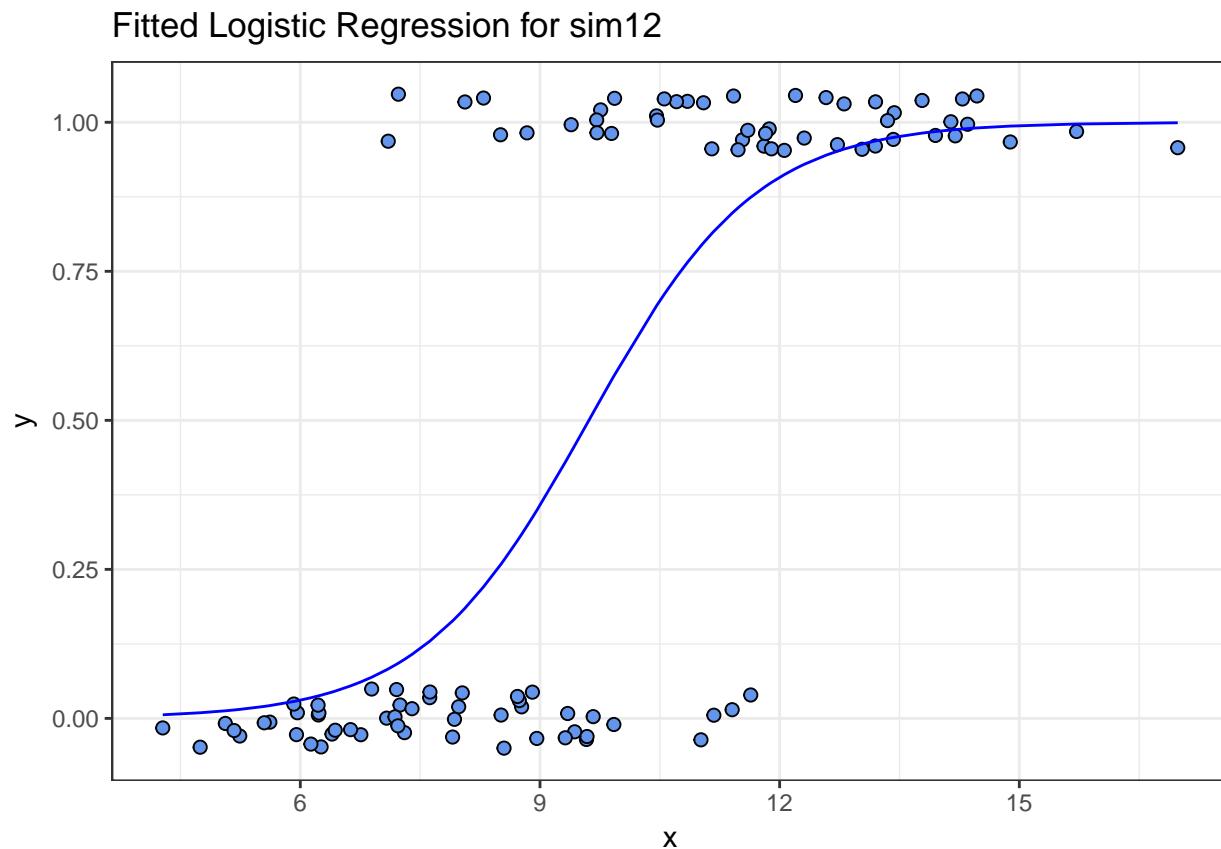
```
m12b.aug <- augment(model12b, sim12, type.predict = "response")
```

```
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 13

Logistic Regression and the `resect` data

13.1 The `resect` data

My source for these data was Riffenburgh (2006). The data describe 134 patients who had undergone resection of the tracheal carina (most often this is done to address tumors in the trachea), and the `resect.csv` data file contains the following variables:

- `id` = a patient ID #,
- `age` = the patient's age at surgery,
- `prior` = prior tracheal surgery (1 = yes, 0 = no),
- `resection` = extent of the resection (in cm),
- `intubated` = whether intubation was required at the end of surgery (1 = yes, 0 = no), and
- `died` = the patient's death status (1 = dead, 0 = alive).

```
resect %>% group_by(died) %>% skim(-id)
```

```
Skim summary statistics
n obs: 134
n variables: 6
group variables: died
```

```
Variable type: integer
died  variable missing complete   n   mean      sd p0 p25 median p75 p100
  0    age      0       117 117 48.05 16.01  8  36     51  62   80
  0  intubated  0       117 117  0.068  0.25  0  0     0  0    1
  0    prior     0       117 117  0.24   0.43  0  0     0  0    1
  1    age      0        17  17 46.41 14.46 26  33     46  60   66
  1  intubated  0        17  17  0.65  0.49  0  0     1  1    1
  1    prior     0        17  17  0.35  0.49  0  0     0  1    1
```

```
Variable type: numeric
died  variable missing complete   n mean      sd p0 p25 median p75 p100
  0  resection   0       117 117 2.82 1.21  1  2     2.5 3.5   6
  1  resection   0        17  17 3.97 1     2 3.5   4  4.5   6
```

We have no missing data, and 17 of the 134 patients died. Our goal will be to understand the characteristics of the patients, and how they relate to the binary outcome of interest, death.

13.2 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.

13.2.1 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 $96 + 8 \times \text{number of candidate parameters}$ 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.

13.3 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.

13.3.1 Interpreting the Coefficients of a Logistic Regression Model

Our model is:

$$\text{logit}(\text{died} = 1) = \log \left(\frac{\Pr(\text{died} = 1)}{1 - \Pr(\text{died} = 1)} \right) = \beta_0 + \beta_1 x = -4.4337 + 0.7417 \times \text{resection}$$

The predicted log odds of death for a subject with a resection of 4 cm is:

$$\log \left(\frac{\Pr(\text{died} = 1)}{1 - \Pr(\text{died} = 1)} \right) = -4.4337 + 0.7417 \times 4 = -1.467$$

The predicted odds of death for a subject with a resection of 4 cm is thus:

$$\frac{\Pr(\text{died} = 1)}{1 - \Pr(\text{died} = 1)} = e^{-4.4337 + 0.7417 \times 4} = e^{-1.467} = 0.2306$$

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:

$$\Pr(\text{died} = 1) = \frac{e^{-4.4337 + 0.7417 \times 4}}{1 + e^{-4.4337 + 0.7417 \times 4}} = \frac{e^{-1.467}}{1 + e^{-1.467}} = \frac{0.2306}{1.2306} = 0.187$$

In general, we can frame the model in terms of a statement about probabilities, like this:

$$\Pr(\text{died} = 1) = \frac{e^{\beta_0 + \beta_1 x}}{1 + e^{\beta_0 + \beta_1 x}} = \frac{e^{-4.4337 + 0.7417 \times \text{resection}}}{1 + e^{-4.4337 + 0.7417 \times \text{resection}}}$$

and so by substituting in values for `resection`, we can estimate the model's fitted probabilities of death.

13.3.2 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 = data_frame(resection = c(4,5)))
```

```
1          2
-1.4669912 -0.7253027
```

- We can also obtain predictions for each subject on the original response (here, probability) scale, backing out of the logit link.

```
predict(res_modA, newdata = data_frame(resection = c(4, 5)),
       type = "response")
```

```
1          2
0.1874004 0.3262264
```

So the predicted probability of death is 0.187 for Sally, the subject with a 4 cm resection, and 0.326 for Harry, the subject with a 5 cm resection.

13.3.3 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:

$$\log \left(\frac{Pr(Harry\ died)}{1 - Pr(Harry\ died)} \right) = -4.4337 + 0.7417 \times 5 = -0.7253, \log \left(\frac{Pr(Sally\ died)}{1 - Pr(Sally\ died)} \right) = -4.4337 + 0.7417 \times 4 = -1.4667.$$

which implies that our estimated odds of death for Harry and Sally are:

$$Odds(Harry\ died) = \frac{Pr(Harry\ died)}{1 - Pr(Harry\ died)} = e^{-4.4337 + 0.7417 \times 5} = e^{-0.7253} = 0.4842 \quad Odds(Sally\ died) = \frac{Pr(Sally\ died)}{1 - Pr(Sally\ died)} = e^{-1.4667}$$

and so the odds ratio is:

$$OR = \frac{Odds(Harry\ died)}{Odds(Sally\ died)} = \frac{0.4842}{0.2307} = 2.099$$

- 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.

13.3.4 Interpreting the rest of the model output from `glm`

```
res_modA
```

```
Call: glm(formula = died ~ resection, family = "binomial", data = resect)

Coefficients:
(Intercept)      resection
-4.4337          0.7417

Degrees of Freedom: 133 Total (i.e. Null); 132 Residual
Null Deviance: 101.9
Residual Deviance: 89.49    AIC: 93.49
```

In addition to specifying the logistic regression coefficients, we are also presented with information on degrees of freedom, deviance (null and residual) and AIC.

- The degrees of freedom indicate the sample size.
 - Recall that we had $n = 134$ subjects in the data. The “Null” model includes only an intercept term (which uses 1 df) and we thus have $n - 1$ (here 133) degrees of freedom available for estimation.
 - In our `res_modA` model, a logistic regression is fit including a single slope (`resection`) and an intercept term. Each uses up one degree of freedom to build an estimate, so we have $n - 2 = 134 - 2 = 132$ residual df remaining.
- The AIC or Akaike Information Criterion (lower values are better) is also provided. This is helpful if we’re comparing multiple models for the same outcome.

13.3.5 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)
```

```
Analysis of Deviance Table
```

```
Model: binomial, link: logit
```

```
Response: died
```

```
Terms added sequentially (first to last)
```

	Df	Deviance	Resid. Df	Resid. Dev
NULL			133	101.943
resection	1	12.45	132	89.493

To complete a deviance test and obtain a p value, we can run the following code to estimate the probability that a chi-square distribution with a single degree of freedom would exhibit an improvement in deviance as large as 12.45.

```
pchisq(12.45, 1, lower.tail = FALSE)
```

```
[1] 0.0004179918
```

The p value for the deviance test here is about 0.0004. But, again, this isn't a test of whether the model is any good - it assumes the model is true, and then tests some consequences.

- Specifically, it tests whether (if the model is TRUE) some of the model's coefficients are non-zero.
- That's not so practically useful, so I discourage you from performing global tests of a logistic regression model with a deviance test.

13.3.6 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 $-2 * \log(Lik)$.

```
glance(res_modA)
```

	null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual
1	101.9431	133	-44.74646	93.49292	99.2886	89.49292	132

The `glance` result also provides the AIC, and the BIC (Bayes Information Criterion), each of which is helpful in understanding comparisons between multiple models for the same outcome (with smaller values of either criterion indicating better models.) The AIC is based on the deviance, but penalizes you for making the model more complicated. The BIC does the same sort of thing but with a different penalty.

Again we see that we have a null deviance of 101.94 on 133 degrees of freedom. Including the `resection` information in the model decreased the deviance to 89.49 points on 132 degrees of freedom, so that's a decrease of 12.45 points while using one degree of freedom, a statistically significant reduction in deviance.

13.4 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.

13.4.1 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 hypotheses being tested here are H_0 : `resection` does not have an effect on the log odds of `died` vs. H_A : `resection` does have such an effect.
- Another way of stating this is that the p value assesses whether the estimated coefficient of `resection`, 0.7417, is statistically detectably different from 0. If the coefficient (on the logit scale) for `resection` was truly 0, this would mean that:
 - the log odds of death did not change based on the `resection` size,
 - the odds of death were unchanged based on the `resection` size (the odds ratio would be 1), and
 - the probability of death was unchanged based on the `resection` size.

In our case, we have a statistically detectable change in the log odds of `died` associated with changes in `resection`, according to this p value. We conclude that `resection` size is associated with a positive impact on death rates (death rates are generally higher for people with larger resections.)

13.4.2 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.

```
exp(coef(res_modA))
```

(Intercept)	resection
0.01186995	2.09947754

```
exp(confint(res_modA))

Waiting for profiling to be done...

      2.5 %    97.5 %
(Intercept) 0.001756429 0.05750655
resection   1.381124459 3.34782604
```

From this output, we can estimate the odds ratio for death associated with a 1 cm increase in resection size is 2.099, with a 95% CI of (1.38, 3.35). - If the odds ratio was 1, it would indicate that the odds of death did not change based on the change in resection size. - Here, it's clear that the estimated odds of death will be larger (odds > 1) for subjects with larger resection sizes. Larger odds of death also indicate larger probabilities of death. This confidence interval indicates that with 95% confidence, we conclude that increases in resection size are associated with statistically detectable increases in the odds of death. - If the odds ratio was less than 1 (remember that it cannot be less than 0) that would mean that subjects with larger resection sizes were associated with smaller estimated odds of death.

13.4.3 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^{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.

13.4.4 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.

13.4.5 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.

13.5 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.

13.5.1 Using `augment` to capture the fitted probabilities

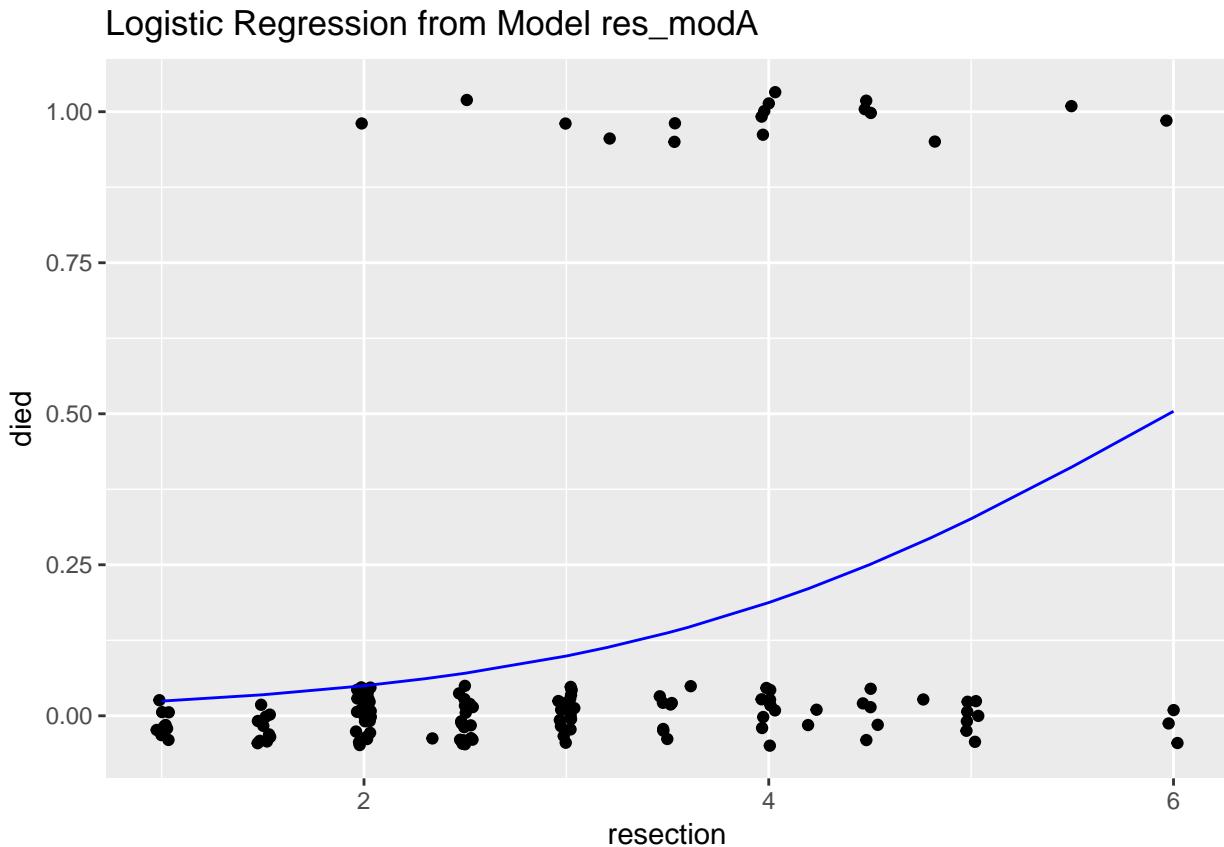
```
res_A_aug <- augment(res_modA, resect,
                      type.predict = "response")
head(res_A_aug)
```

	<code>id</code>	<code>age</code>	<code>prior</code>	<code>resection</code>	<code>intubated</code>	<code>died</code>	<code>.fitted</code>	<code>.se.fit</code>	<code>.resid</code>
1	1	34	1	1	2.5	0	0 0.07046791	0.02562381	-0.3822929
2	2	57	0	0	5.0	0	0 0.32622637	0.08605551	-0.8886631
3	3	60	1	1	4.0	1	1 0.18740037	0.04269795	1.8300317
4	4	62	1	1	4.2	0	0 0.21104240	0.04871389	-0.6885386
5	5	28	0	0	6.0	1	1 0.50409637	0.14302982	1.1704596
6	6	52	0	0	3.0	0	0 0.09897375	0.02867196	-0.4565542
							<code>.hat</code>	<code>.sigma</code>	<code>.cooksdi</code>
	1	0.010024061	0.8258481	0.0003876961	-0.3842235				
	2	0.033691765	0.8227475	0.0087350915	-0.9040227				
	3	0.011972088	0.8107264	0.0265893468	1.8410857				
	4	0.014252277	0.8243062	0.0019617278	-0.6934983				
	5	0.081835623	0.8196110	0.0477480056	1.2215077				
	6	0.009218619	0.8255581	0.0005157780	-0.4586733				

This approach augments the `resect` data set with fitted, residual and other summaries of each observation's impact on the fit, using the “response” type of prediction, which yields the fitted probabilities in the `.fitted` column.

13.5.2 Plotting a Logistic Regression Model's Fitted Values

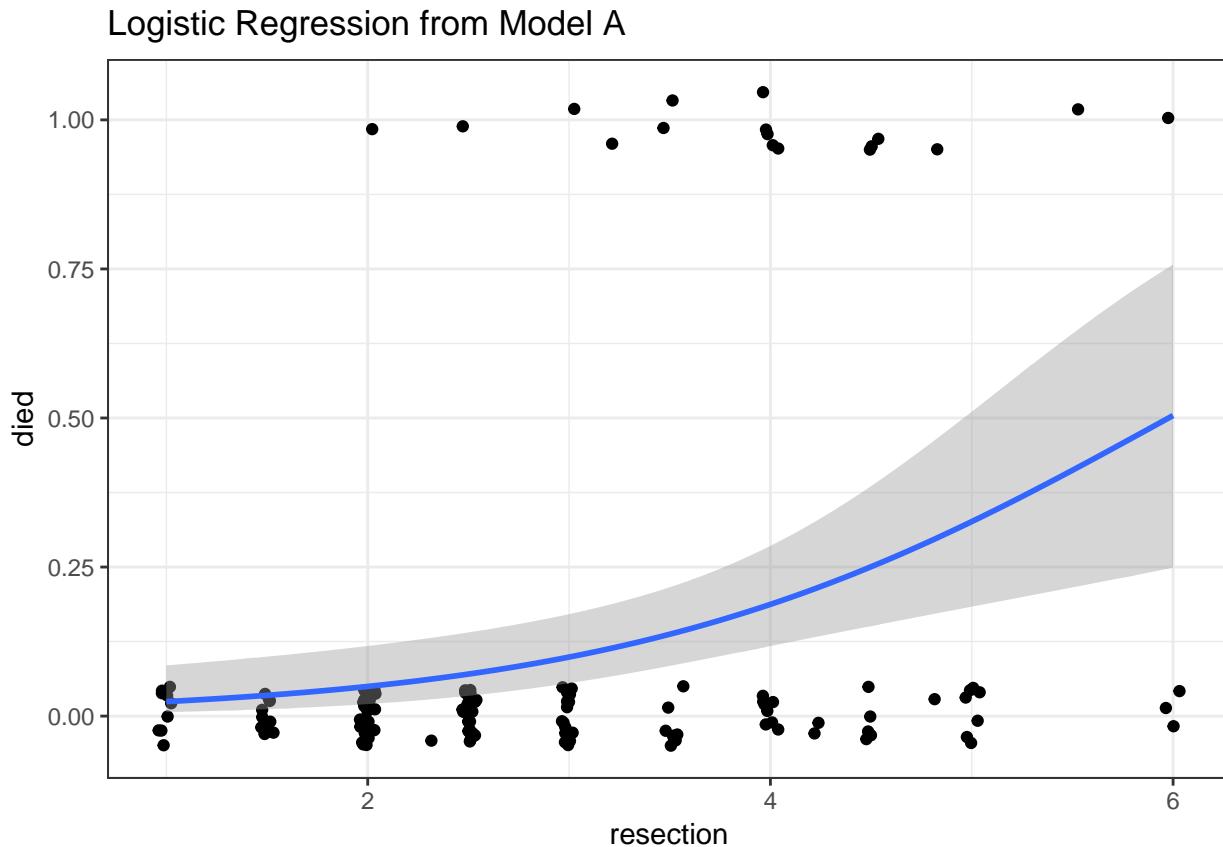
```
ggplot(res_A_aug, aes(x = resection, y = died)) +
  geom_jitter(height = 0.05) +
  geom_line(aes(x = resection, y = .fitted),
            col = "blue") +
  labs(title = "Logistic Regression from Model res_modA")
```



13.5.3 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()
```



As expected, we see an increase in the model probability of death as the extent of the resection grows larger.

13.6 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.

13.7 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).

$$TPF(R) = Pr(\text{resection} > R | \text{subjectdied})$$

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)

$$FPF(R) = Pr(\text{resection} > R | \text{subjectdidnotdie})$$

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.

13.7.1 Interpreting the Area under the ROC curve

The AUC or Area under the ROC curve is the amount of space underneath the ROC curve. Often referred to as the C statistic, the AUC represents the quality of your TPR and FPR overall in a single number. The C statistic ranges from 0 to 1, with C = 0.5 for a prediction that is no better than random guessing, and C = 1 for a perfect prediction model.

Next, I'll build a simulation to demonstrate several possible ROC curves in the sections that follow.

```
set.seed(432999)
sim.temp <- data_frame(x = rnorm(n = 200),
                        prob = exp(x)/(1 + exp(x)),
                        y = as.numeric(1 * runif(200) < prob))

sim.temp <- sim.temp %>%
  mutate(p_guess = 1,
        p_perfect = y,
        p_bad = exp(-2*x) / (1 + exp(-2*x)),
        p_ok = prob + (1-y)*runif(1, 0, 0.05),
        p_good = prob + y*runif(1, 0, 0.27))
```

13.7.1.1 What if we are guessing?

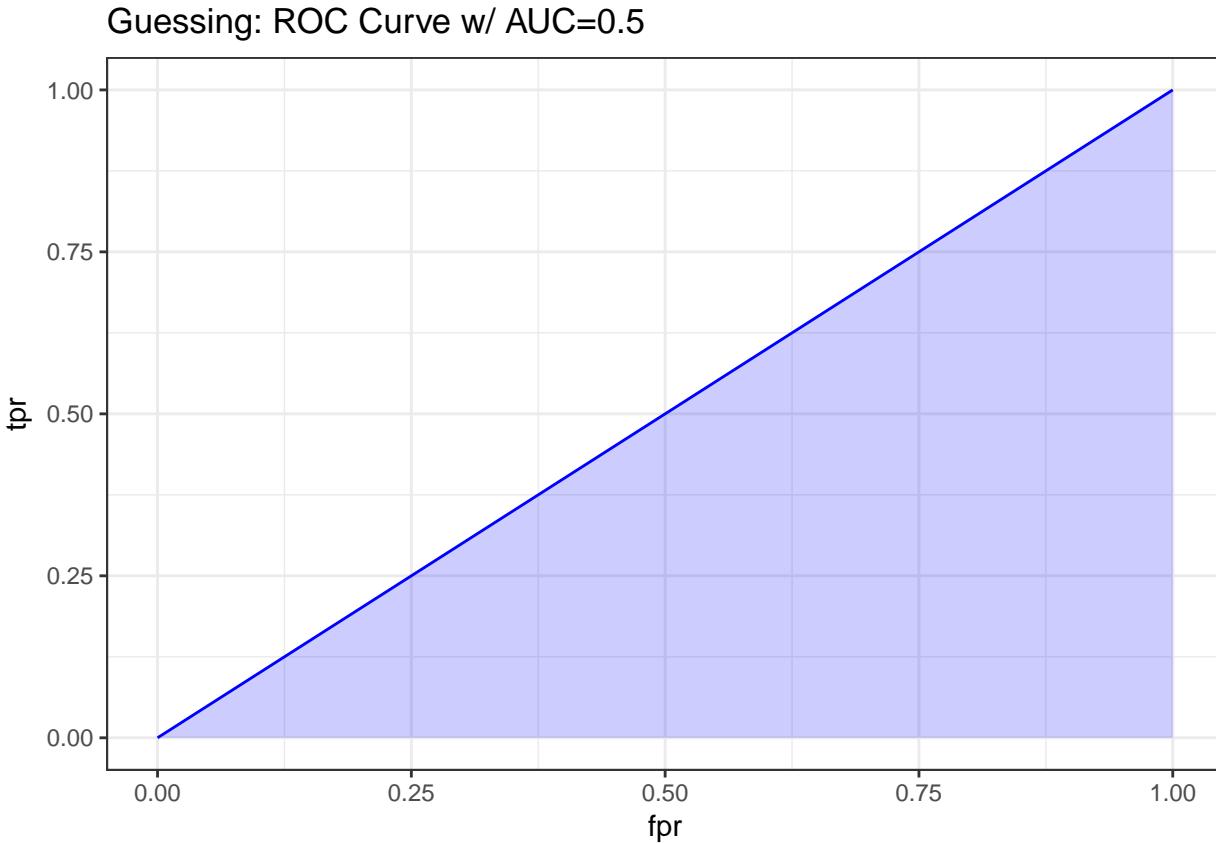
If we're guessing completely at random, then the model should correctly classify a subject (as died or not died) about 50% of the time, so the TPR and FPR will be equal. This yields a diagonal line in the ROC curve, and an area under the curve (C statistic) of 0.5.

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()
```



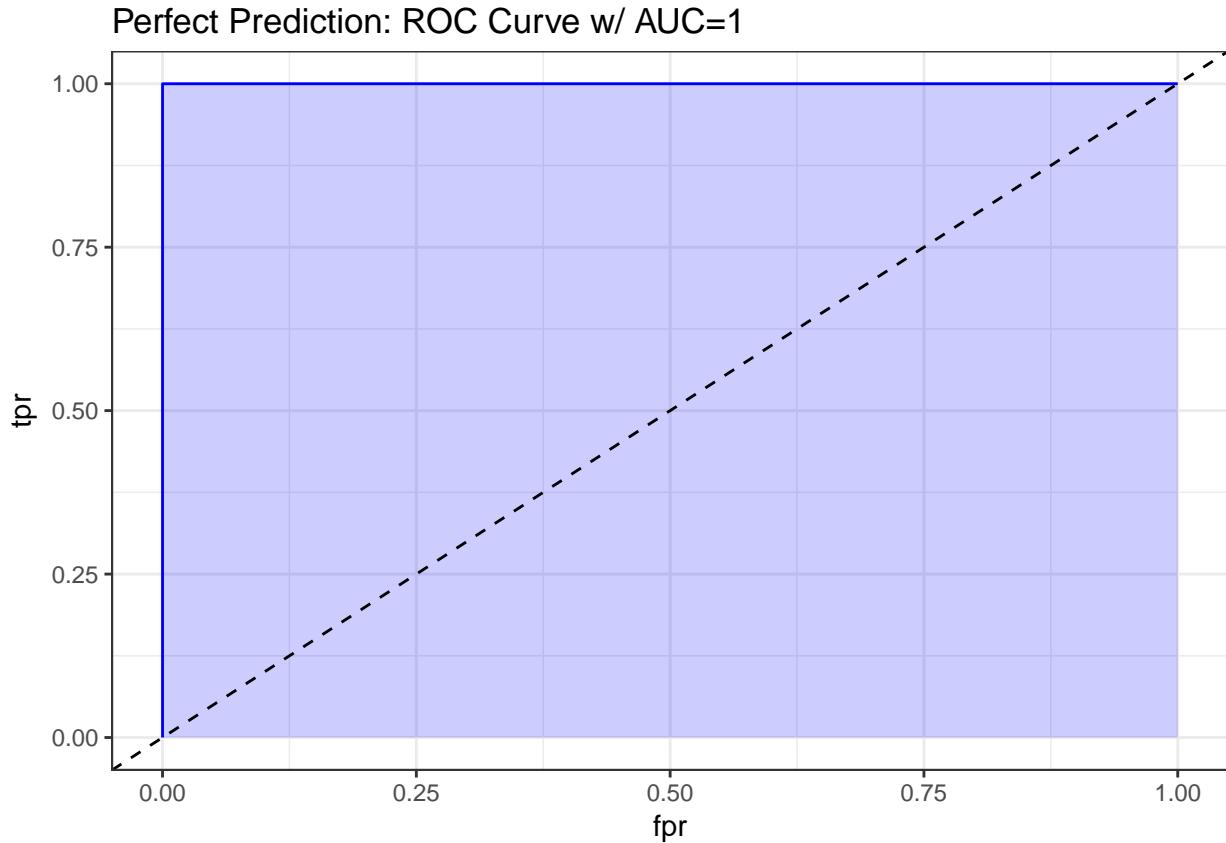
13.7.1.2 What if we classify things perfectly?

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()
```



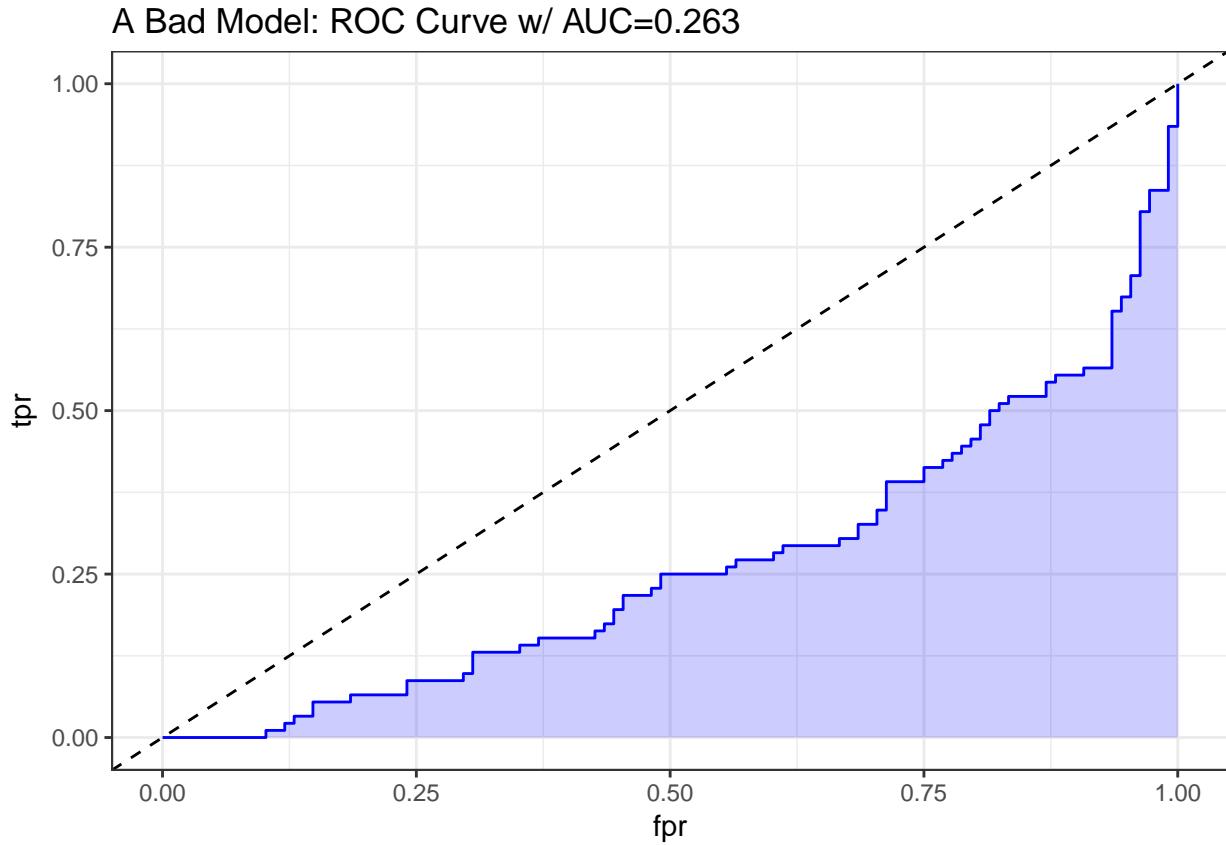
13.7.1.3 What does “worse than guessing” look like?

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()
```



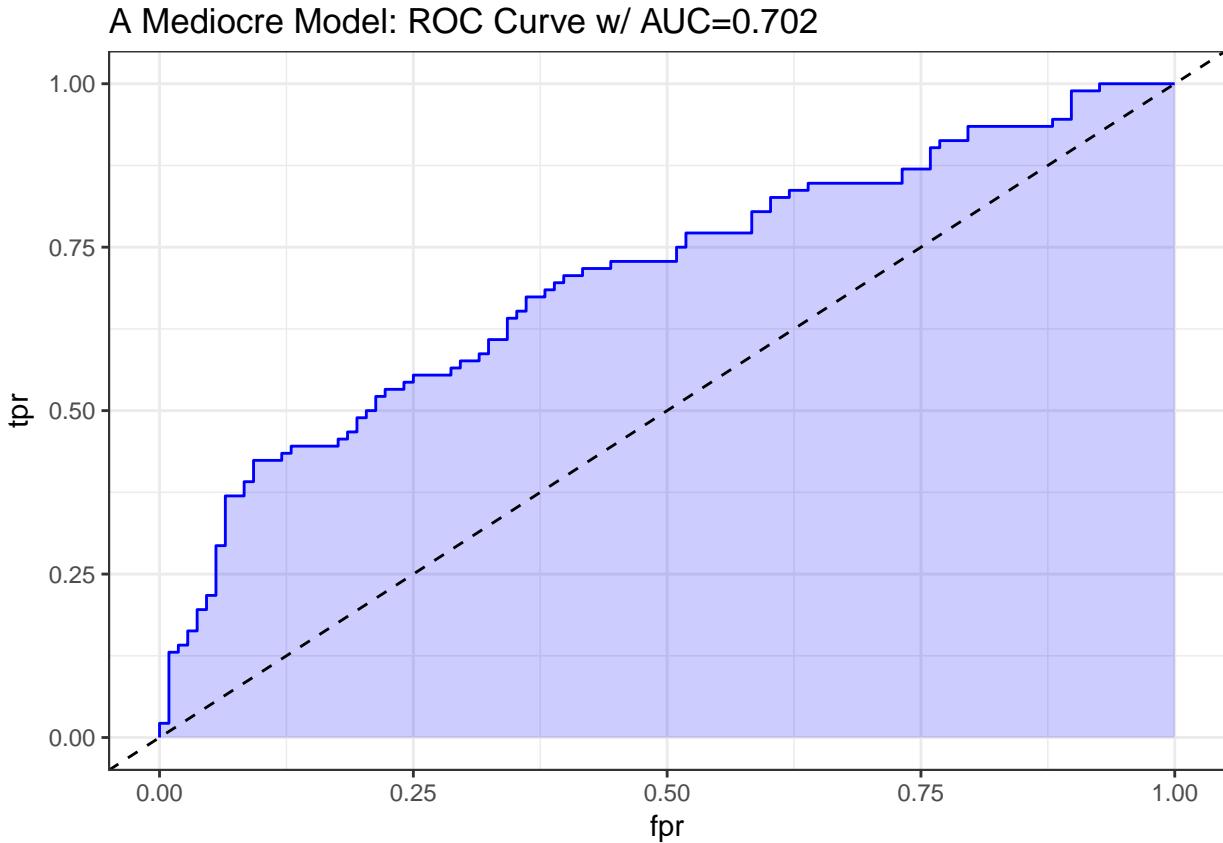
13.7.1.4 What does “better than guessing” look like?

An “OK” classifier will appear above and to the left of the diagonal line we’d see if we were completely guessing. Such a model will have a c statistic above 0.5, and might have some value. The plot below shows a very fairly poor model, but at least it’s better than guessing.

```
pred_ok <- prediction(sim.temp$p_ok, sim.temp$y)
perf_ok <- performance(pred_ok, measure = "tpr", x.measure = "fpr")
auc_ok <- performance(pred_ok, measure="auc")

auc_ok <- round(auc_ok@y.values[[1]],3)
roc_ok <- data.frame(fpr=unlist(perf_ok@x.values),
                      tpr=unlist(perf_ok@y.values),
                      model="GLM")

ggplot(roc_ok, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2, fill = "blue") +
  geom_line(aes(y=tpr), col = "blue") +
  geom_abline(intercept = 0, slope = 1, lty = "dashed") +
  labs(title = paste0("A Mediocre Model: ROC Curve w/ AUC=", auc_ok)) +
  theme_bw()
```



Sometimes people grasp for a rough guide as to the accuracy of a model’s predictions based on the area under the ROC curve. A common thought is to assess the C statistic much like you would a class grade.

C statistic	Interpretation
0.90 to 1.00	model does an excellent job at discriminating “yes” from “no” (A)
0.80 to 0.90	model does a good job (B)
0.70 to 0.80	model does a fair job (C)
0.60 to 0.70	model does a poor job (D)
0.50 to 0.60	model fails (F)
below 0.50	model is worse than random guessing

13.7.1.5 What does “pretty good” look like?

A strong and good classifier will appear above and to the left of the diagonal line we’d see if we were completely guessing, often with a nice curve that is continually increasing and appears to be pulled up towards the top left. Such a model will have a c statistic well above 0.5, but not as large as 1. The plot below shows a stronger model, which appears substantially better than guessing.

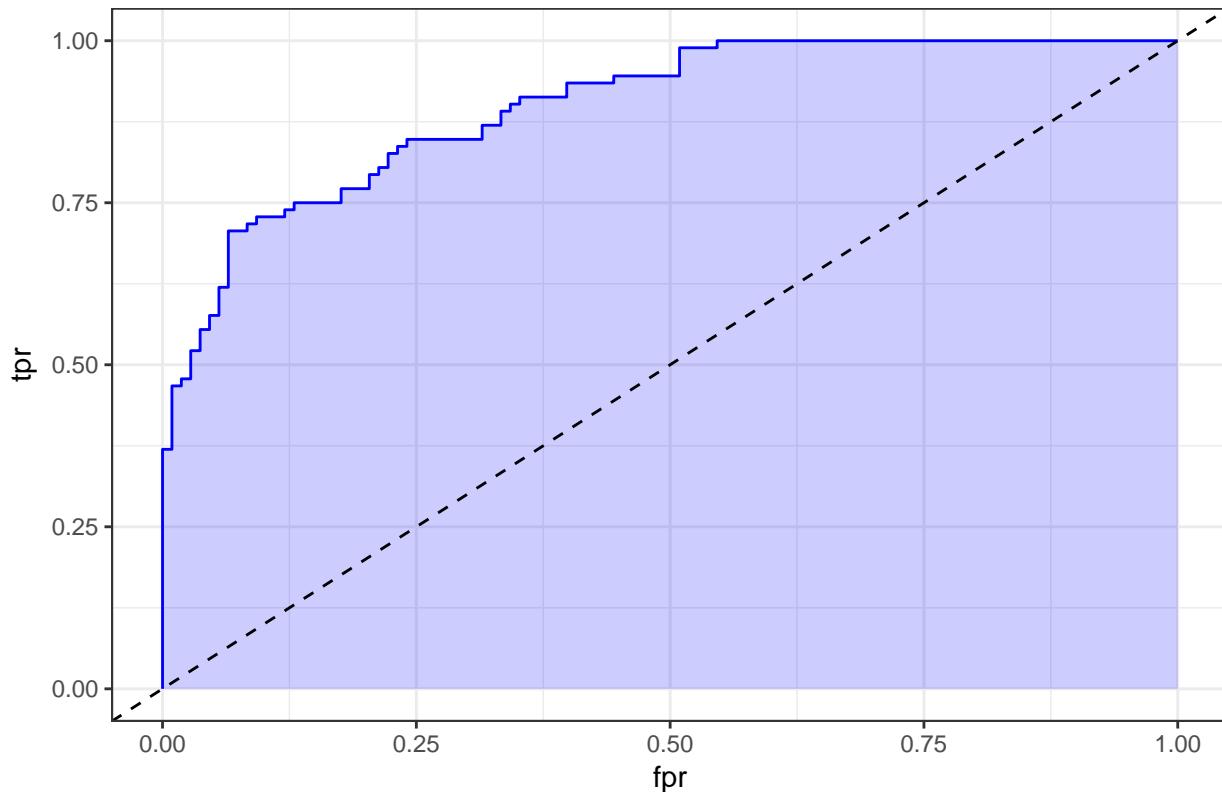
```
pred_good <- prediction(sim.temp$p_good, sim.temp$y)
perf_good <- performance(pred_good, measure = "tpr", x.measure = "fpr")
auc_good <- performance(pred_good, measure="auc")

auc_good <- round(auc_good@y.values[[1]],3)
roc_good <- data.frame(fpr=unlist(perf_good@x.values),
```

```
tpr=unlist(perf_good@y.values),
model="GLM")

ggplot(roc_good, aes(x=fpr, ymin=0, ymax=tpr)) +
  geom_ribbon(alpha=0.2, fill = "blue") +
  geom_line(aes(y=tpr), col = "blue") +
  geom_abline(intercept = 0, slope = 1, lty = "dashed") +
  labs(title = paste0("A Pretty Good Model: ROC Curve w/ AUC=", auc_good)) +
  theme_bw()
```

A Pretty Good Model: ROC Curve w/ AUC=0.899



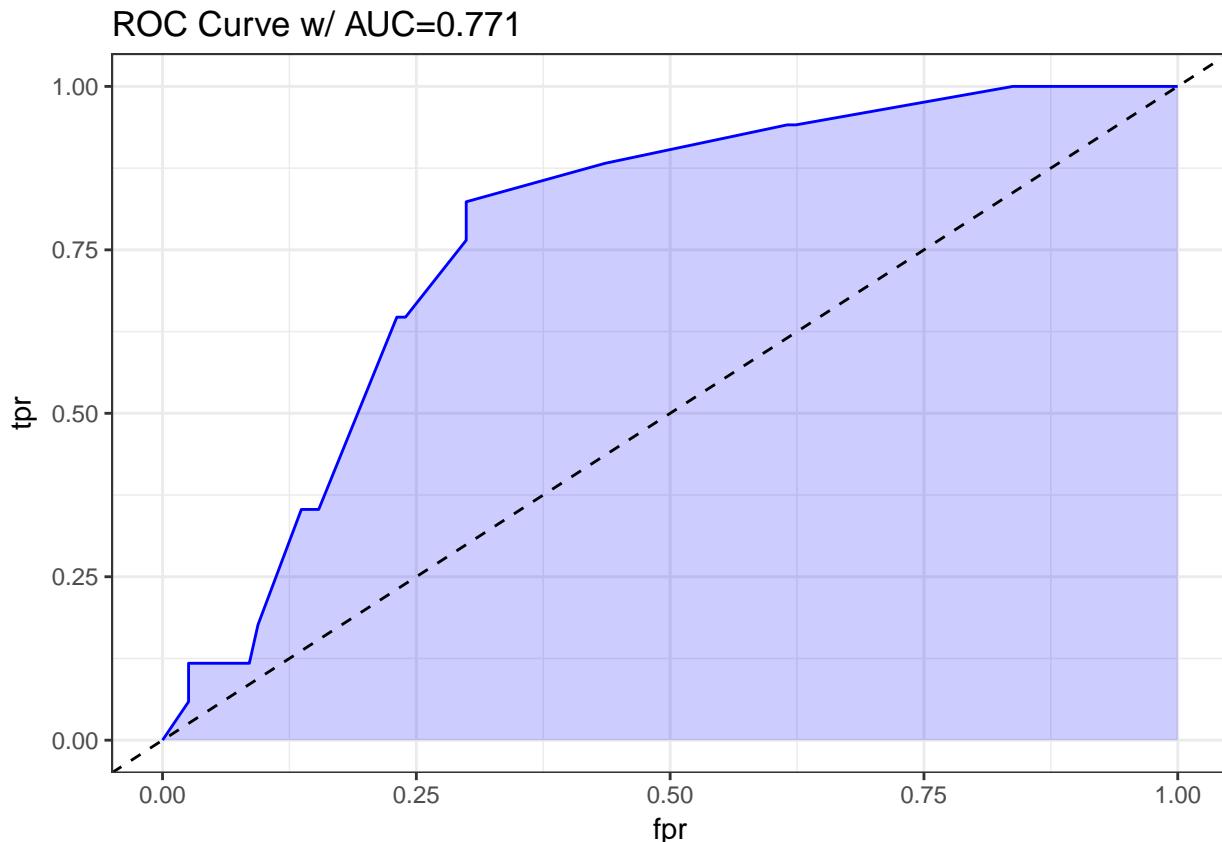
13.8 The ROC Plot for res_modA

Let me show you the ROC curve for our res_modA model.

```
## requires ROOCR 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.

13.8.1 Another way to plot the ROC Curve

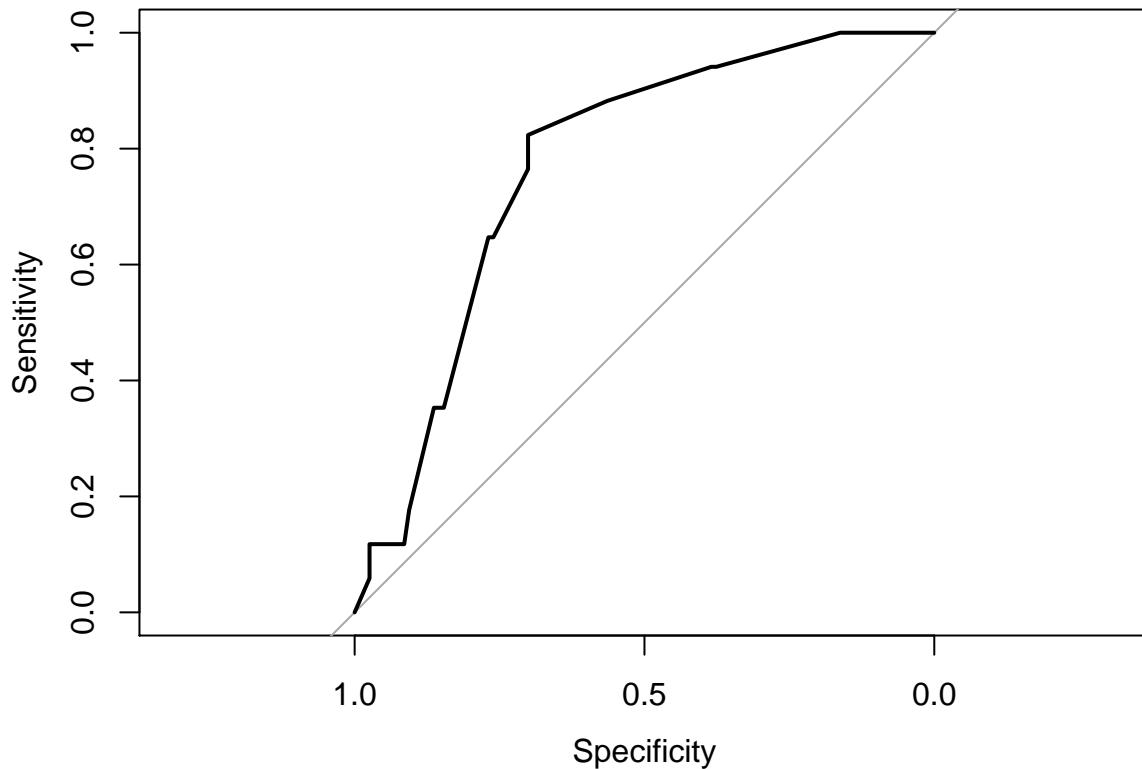
If we’ve loaded the pROC package, we can also use the following (admittedly simpler) approach to plot the ROC curve, without ggplot2, and to obtain the C statistic, and a 95% confidence interval around that C statistic.

```
## requires pROC package
roc.modA <-
  roc(resect$died ~ predict(res_modA, type="response"),
      ci = TRUE)

roc.modA
```

Call:
 roc.formula(formula = resект\$died ~ predict(res_modA, type = "response"), ci = TRUE)

```
Data: predict(res_modA, type = "response") in 117 controls (resect$died 0) < 17 cases (resect$died 1).
Area under the curve: 0.7707
95% CI: 0.67-0.8715 (DeLong)
plot(roc.modA)
```



13.9 Assessing Residual Plots from Model A

Residuals are certainly less informative for logistic regression than they are for linear regression: not only do yes/no outcomes inherently contain less information than continuous ones, but the fact that the adjusted response depends on the fit hampers our ability to use residuals as external checks on the model.

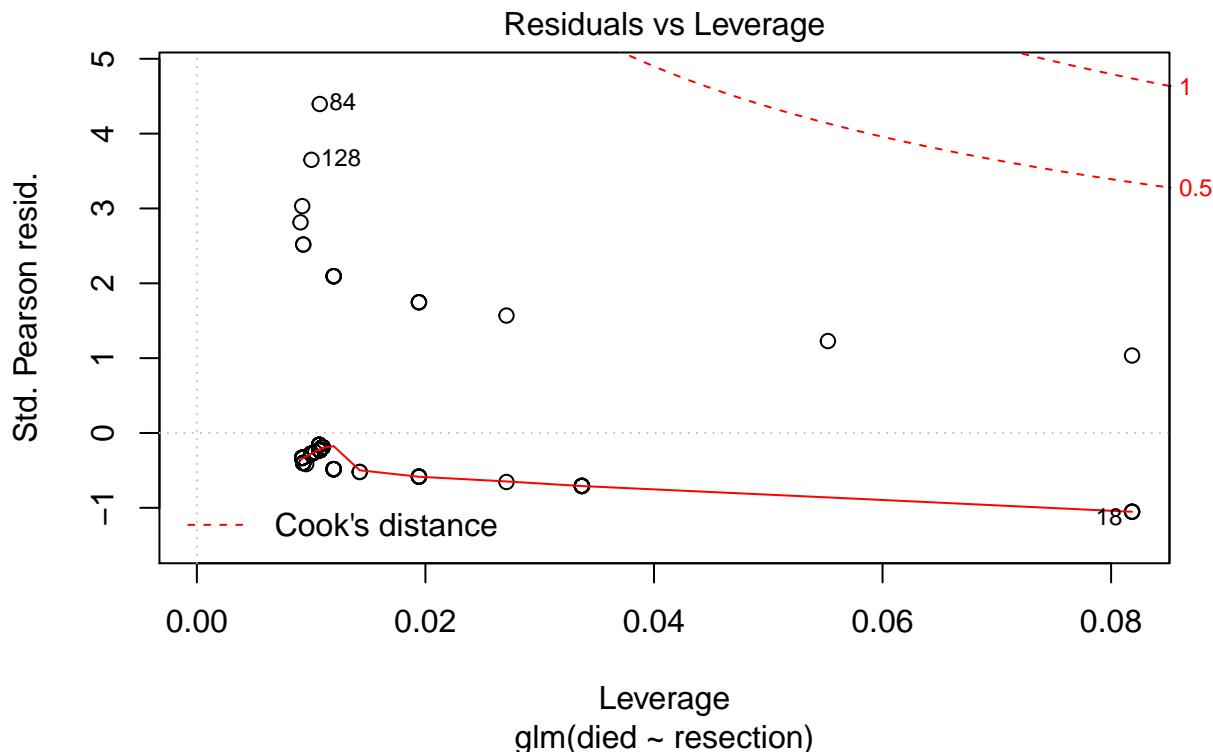
This is mitigated to some extent, however, by the fact that we are also making fewer distributional assumptions in logistic regression, so there is no need to inspect residuals for, say, skewness or heteroskedasticity.

- Patrick Breheny, University of Kentucky, Slides on GLM Residuals and Diagnostics

The usual residual plots are available in R for a logistic regression model, but most of them are irrelevant in the logistic regression setting. The residuals shouldn't follow a standard Normal distribution, and they will not show constant variance over the range of the predictor variables, so plots looking into those issues aren't helpful.

The only plot from the standard set that we'll look at in many settings is plot 5, which helps us assess influence (via Cook's distance contours), and a measure related to leverage (how unusual an observation is in terms of the predictors) and standardized Pearson residuals.

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



In this case, I don't see any highly influential points, as no points fall outside of the Cook's distance (0.5 or 1) contours.

13.10 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
```

13.10.1 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.	Df	Deviance
1	132	89.493		
2	129	67.359	3	22.134

The addition of `age`, `prior` and `intubated` reduces the lack of fit by 22.134 points, at a cost of 3 degrees of freedom.

```
glance(res_modA)
```

	null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual
1	101.9431	133	-44.74646	93.49292	99.2886	89.49292	132

```
glance(res_modB)
```

	null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual
1	101.9431	133	-33.6793	77.3586	91.8478	67.3586	129

By either AIC or BIC, the larger model (`res_modB`) looks more effective.

13.10.2 Interpreting Model B

```
summary(res_modB)
```

Call:

```
glm(formula = died ~ resection + age + prior + intubated, family = binomial,
    data = resect)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.7831	-0.3741	-0.2386	-0.2014	2.5228

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-5.152886	1.469453	-3.507	0.000454 ***
resection	0.612211	0.282807	2.165	0.030406 *
age	0.001173	0.020646	0.057	0.954700
prior	0.814691	0.704785	1.156	0.247705
intubated	2.810797	0.658395	4.269	1.96e-05 ***

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

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 101.943 on 133 degrees of freedom
Residual deviance: 67.359 on 129 degrees of freedom
AIC: 77.359
```

Number of Fisher Scoring iterations: 6

It appears that the `intubated` predictor adds significant value to the model, by the Wald test.

Let's focus on the impact of these variables through odds ratios.

```
exp(coef(res_modB))
```

	(Intercept)	resection	age	prior	intubated
0.005782692	1.844504859	1.001173503	2.258476846	16.623153519	

```
exp(confint(res_modB))
```

Waiting for profiling to be done...

	2.5 %	97.5 %
(Intercept)	0.0002408626	0.0837263
resection	1.0804548590	3.3495636
age	0.9618416869	1.0442885
prior	0.5485116610	9.1679931
intubated	4.7473282453	64.6456919

At a 5% significance level, we might conclude that:

- larger sized `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.

13.11 Plotting Model B

Let's think about plotting the fitted values from our model, in terms of probabilities.

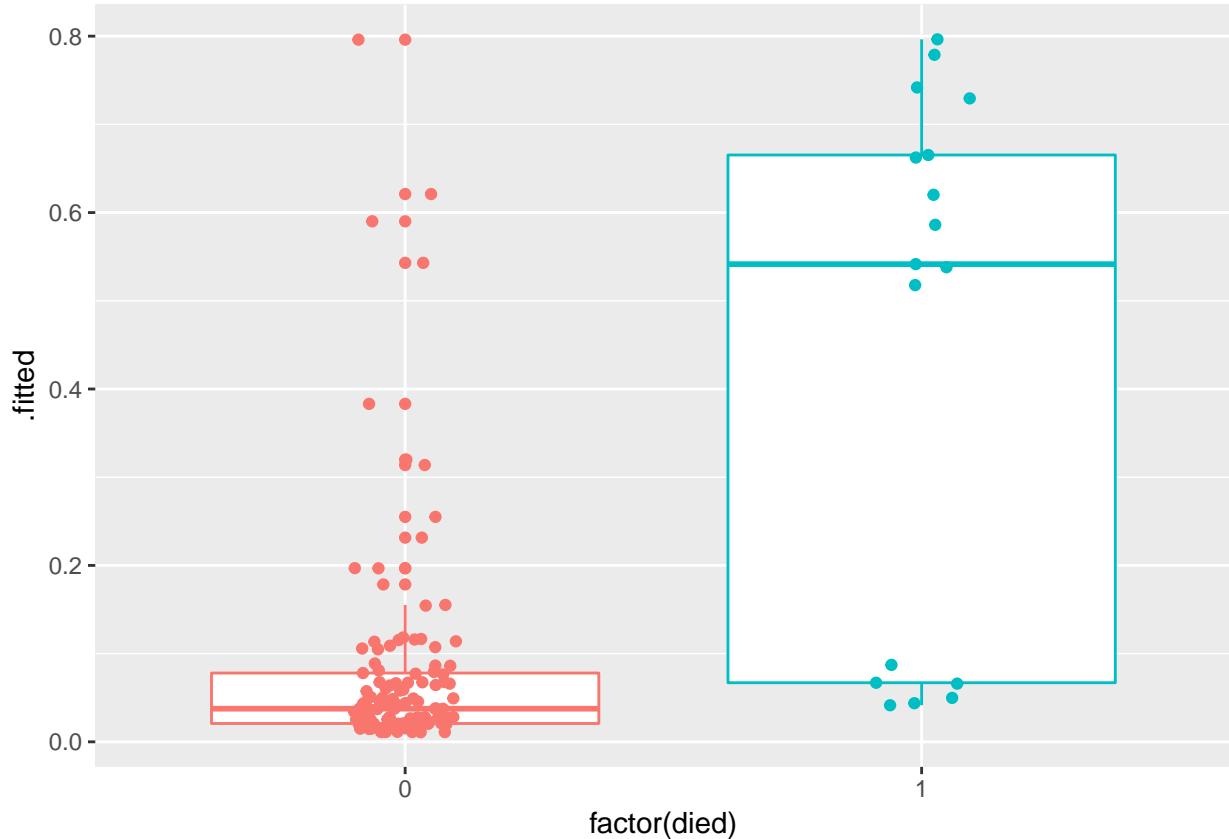
13.11.1 Using `augment` to capture the fitted probabilities

```
res_B_aug <- augment(res_modB, resect,
                      type.predict = "response")
head(res_B_aug)
```

	id	age	prior	resection	intubated	died	.fitted	.se.fit	.resid
1	1	34	1	2.5	0	0	0.05908963	0.03851118	-0.3490198
2	2	57	0	5.0	0	0	0.11660492	0.06253774	-0.4979613
3	3	60	1	4.0	1	1	0.72944600	0.15010423	0.7943172
4	4	62	1	4.2	0	0	0.15522494	0.09607978	-0.5808354
5	5	28	0	6.0	1	1	0.79641141	0.14588554	0.6747435
6	6	52	0	3.0	0	0	0.03713809	0.01933270	-0.2751191
	.hat	.sigma	.cooksdi	.std.resid					
1	0.02667562	0.7247491	0.0003536652	-0.3537702					
2	0.03796756	0.7240341	0.0010829917	-0.5076925					
3	0.11416656	0.7215778	0.0107925872	0.8439524					
4	0.07039819	0.7234665	0.0029937671	-0.6024273					
5	0.13126049	0.7225958	0.0088920280	0.7239256					
6	0.01045207	0.7250114	0.0000823406	-0.2765683					

13.11.2 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.

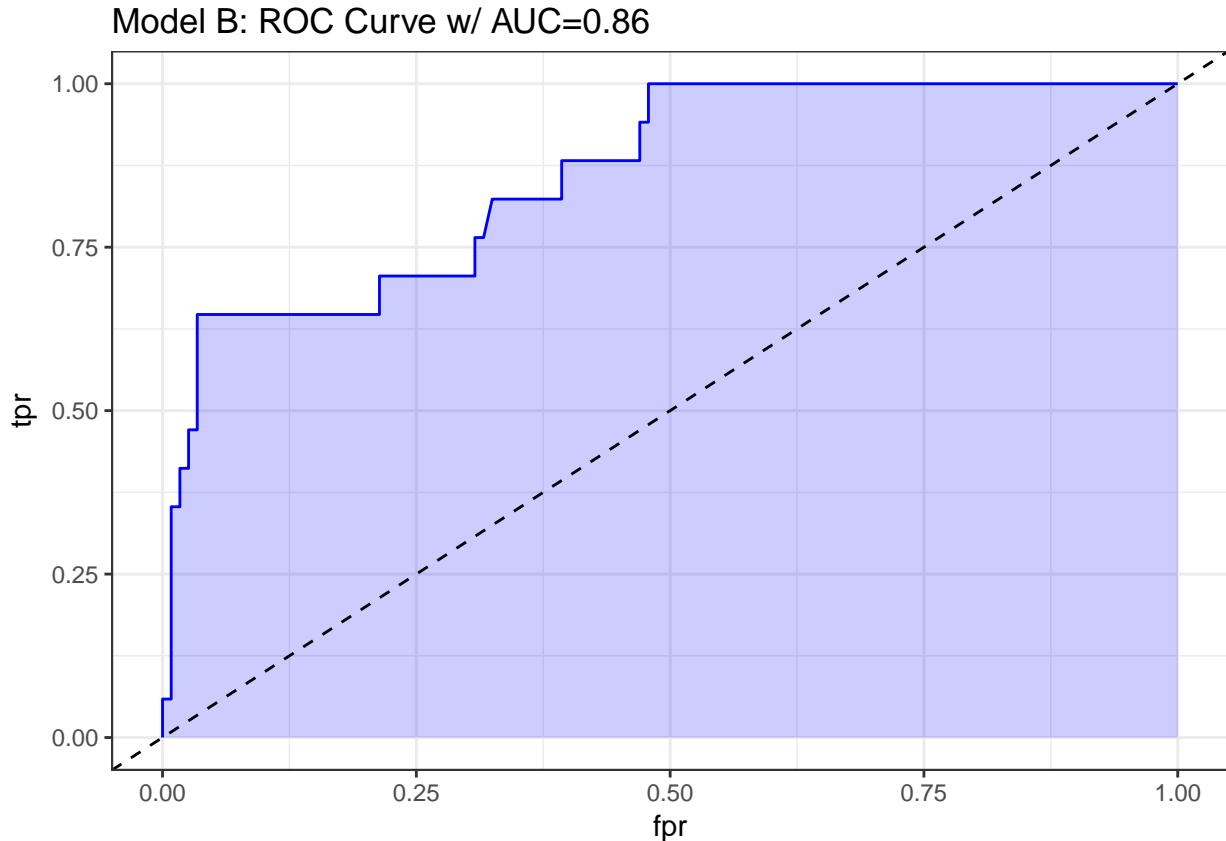
13.11.3 The ROC curve for Model B

```
## requires ROOCR 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)) +
```

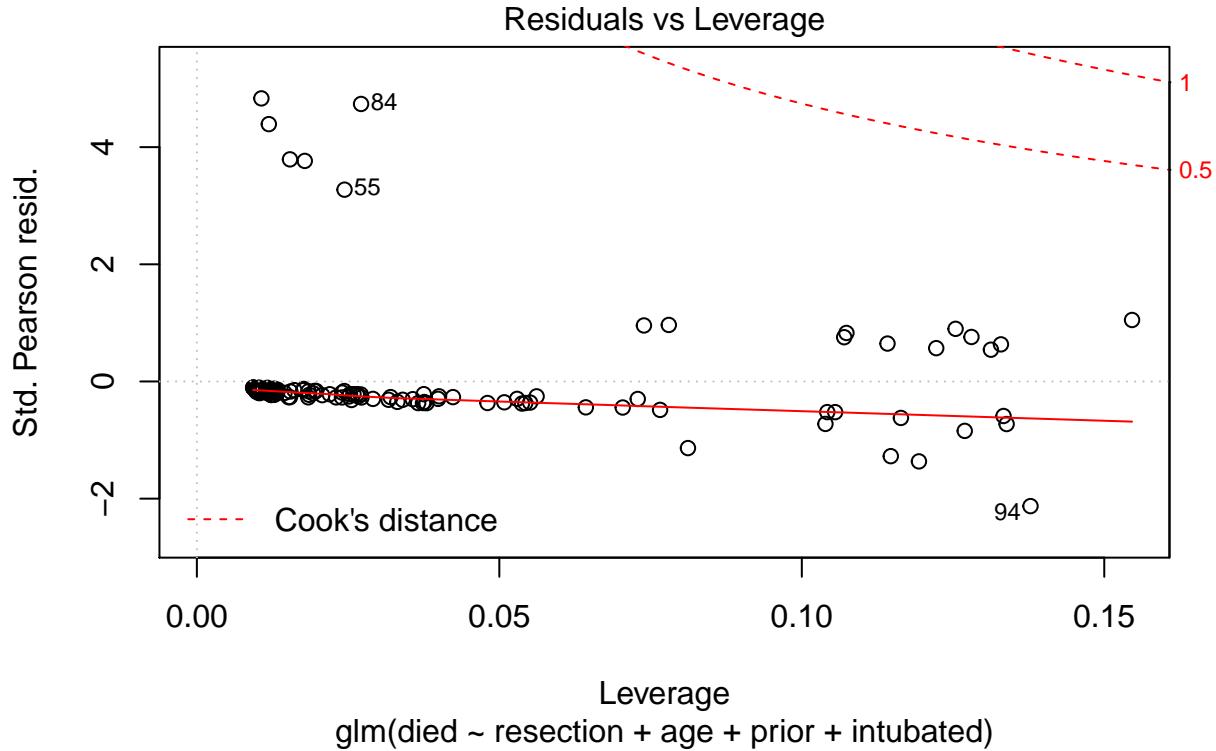
```
geom_ribbon(alpha=0.2, fill = "blue") +  
  geom_line(aes(y=tpr), col = "blue") +  
  geom_abline(intercept = 0, slope = 1, lty = "dashed") +  
  labs(title = paste0("Model B: ROC Curve w/ AUC=", auc)) +  
  theme_bw()
```



The area under the curve (C-statistic) is 0.86, which certainly looks like a more discriminating fit than model A with resection alone.

13.11.4 Residuals, Leverage and Influence

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



Again, we see no signs of deeply influential points in this model.

13.12 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)
```

	Obs	Model Likelihood		Discrimination		Rank Discrim.	
		Ratio	Test	R2	Indexes	C	Indexes
0	134	LR	chi2	12.45	0.167	0.771	
1	117	d.f.		1	g	1.037	Dxy 0.541
	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.

13.12.1 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.

13.12.2 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)

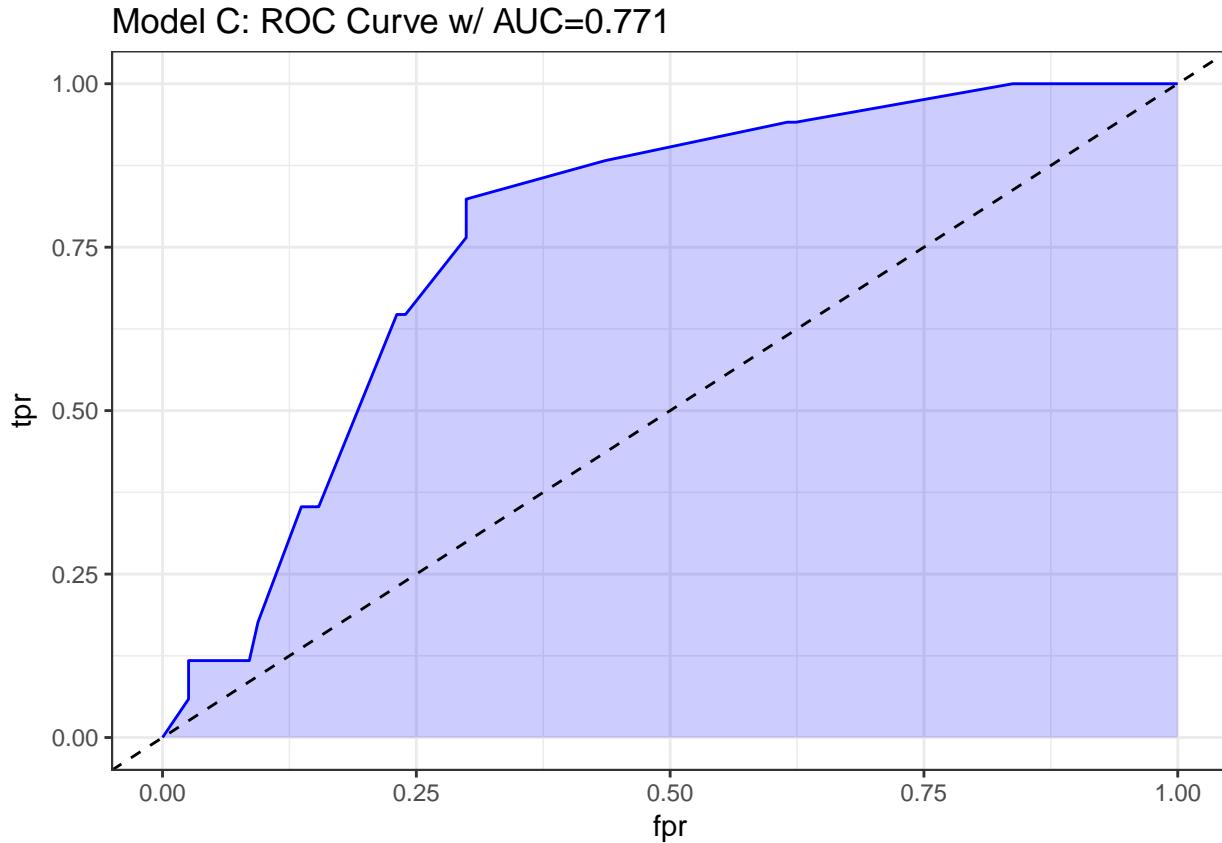
We’ve seen the ROC curve for this model before, when we looked at model `res_modA` fitted using `glm` in the previous chapter. But, just for completeness, I’ll include it.

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.

```
## 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()
```



13.12.3 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 R² 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 (D_{xy}) = .541, so the C statistic is 0.771.

13.12.4 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
------------	----------	------	----------	-----------------	---

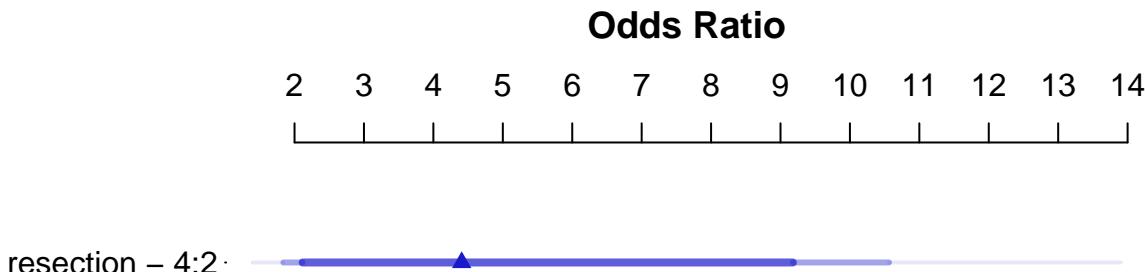
Dxy	0.5415	0.5325	0.5415	-0.0090	0.5505	100
R2	0.1666	0.1664	0.1666	-0.0002	0.1668	100
Intercept	0.0000	0.0000	0.1425	-0.1425	0.1425	100
Slope	1.0000	1.0000	1.0742	-0.0742	1.0742	100
Emax	0.0000	0.0000	0.0416	0.0416	0.0416	100
D	0.0854	0.0872	0.0854	0.0017	0.0837	100
U	-0.0149	-0.0149	-0.0004	-0.0145	-0.0004	100
Q	0.1004	0.1021	0.0859	0.0162	0.0841	100
B	0.1025	0.1032	0.1046	-0.0014	0.1039	100
g	1.0369	1.0247	1.0369	-0.0122	1.0491	100
gp	0.1101	0.1082	0.1101	-0.0019	0.1119	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 corrected estimates are Dxy = 0.551, so C = 0.776, and Nagelkerke R² = 0.167.

13.12.5 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)
```

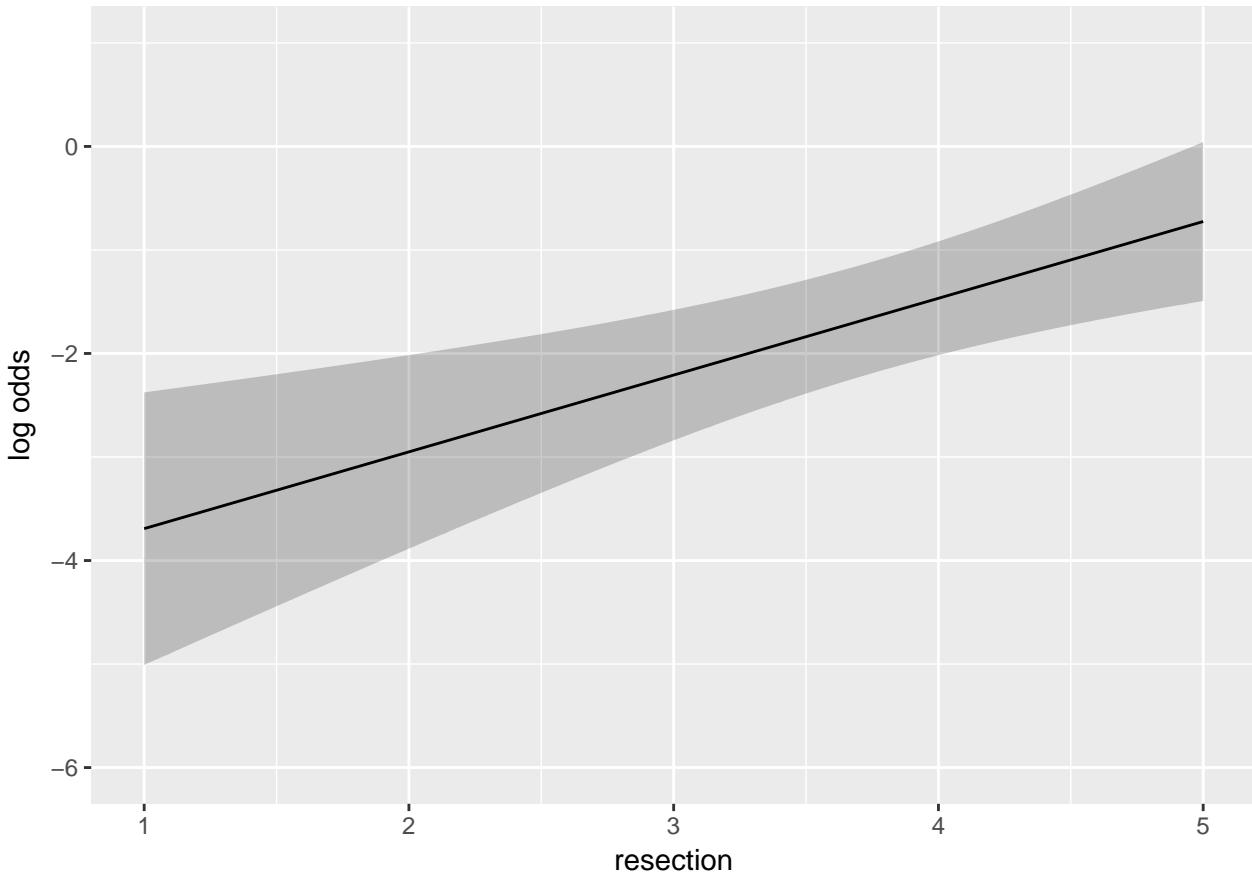
Effects	Response : died									
	Factor	Low	High	Diff.	Effect	S.E.	Lower	0.95	Upper	0.95
resection	2	4	2	1.4834	0.44591	0.6094	2.3574			
Odds Ratio	2	4	2	4.4078	NA	1.8393	10.5630			

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).

13.12.6 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.

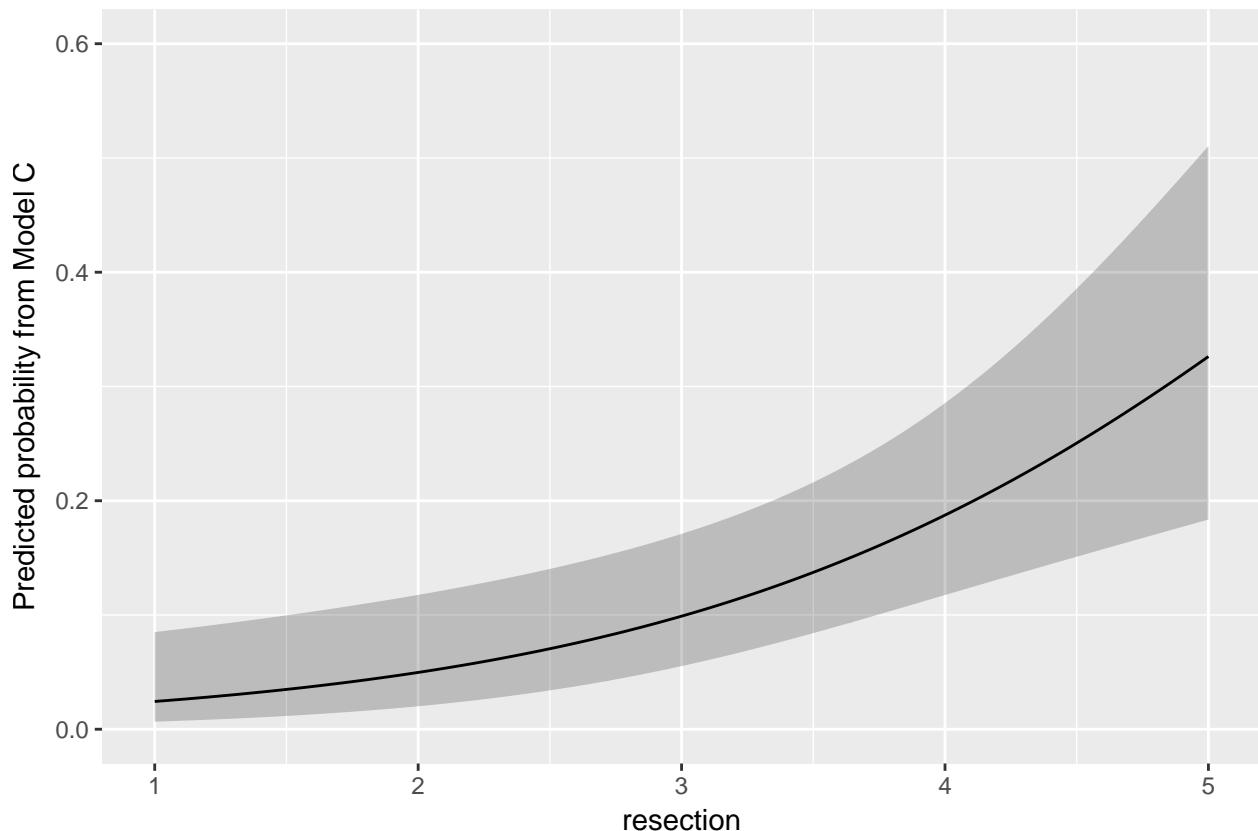
```
ggplot(Predict(res_modC))
```



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")
```

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

13.12.7 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 9×10^{-4} .

13.12.8 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))
```

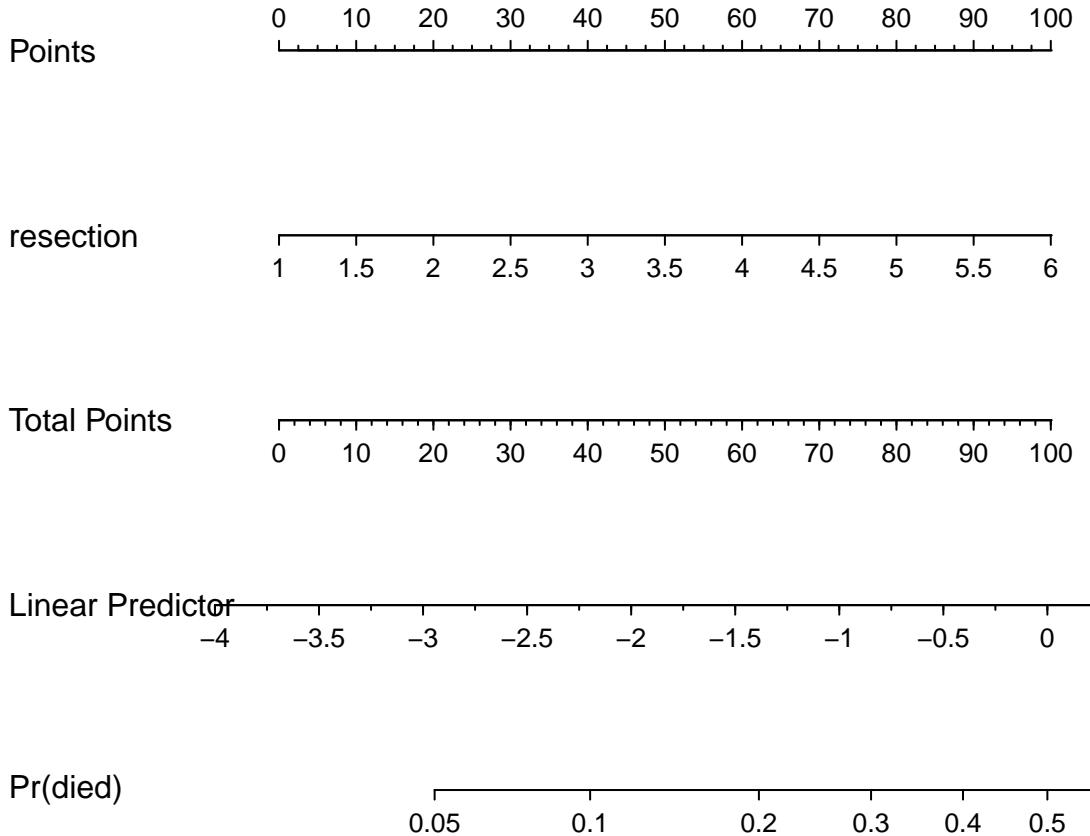
```
Count resection
84      2      *2.0
128      1      2.5
```

It appears that observation 84 may have a meaningful effect on both the intercept and the coefficient for `resection`.

13.12.9 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.

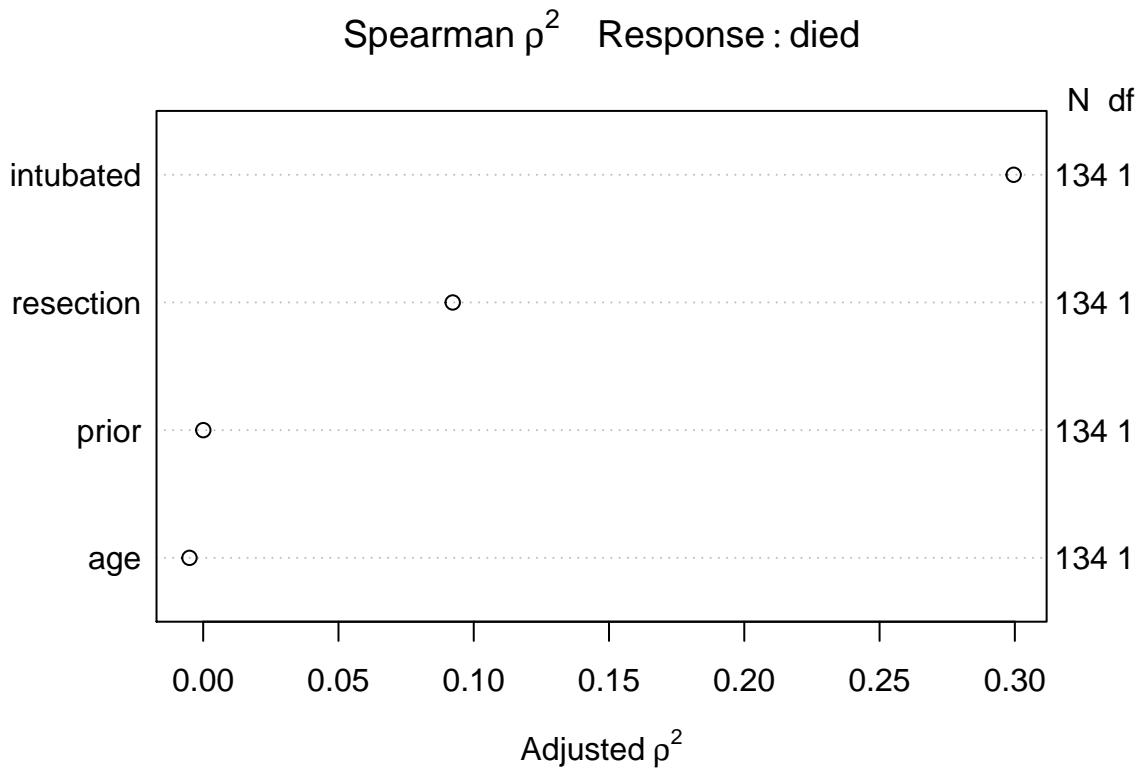
13.13 Model D: An Augmented Kitchen Sink Model

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?

13.13.1 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.

13.13.2 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)
```

13.13.3 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)
```

Obs	134	Model Likelihood		Discrimination		Rank Discrim.	
		LR	chi2	R2	Indexes	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 R² 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.

13.13.4 ANOVA and Wald Tests for Model D

```
anova(res_modD)
```

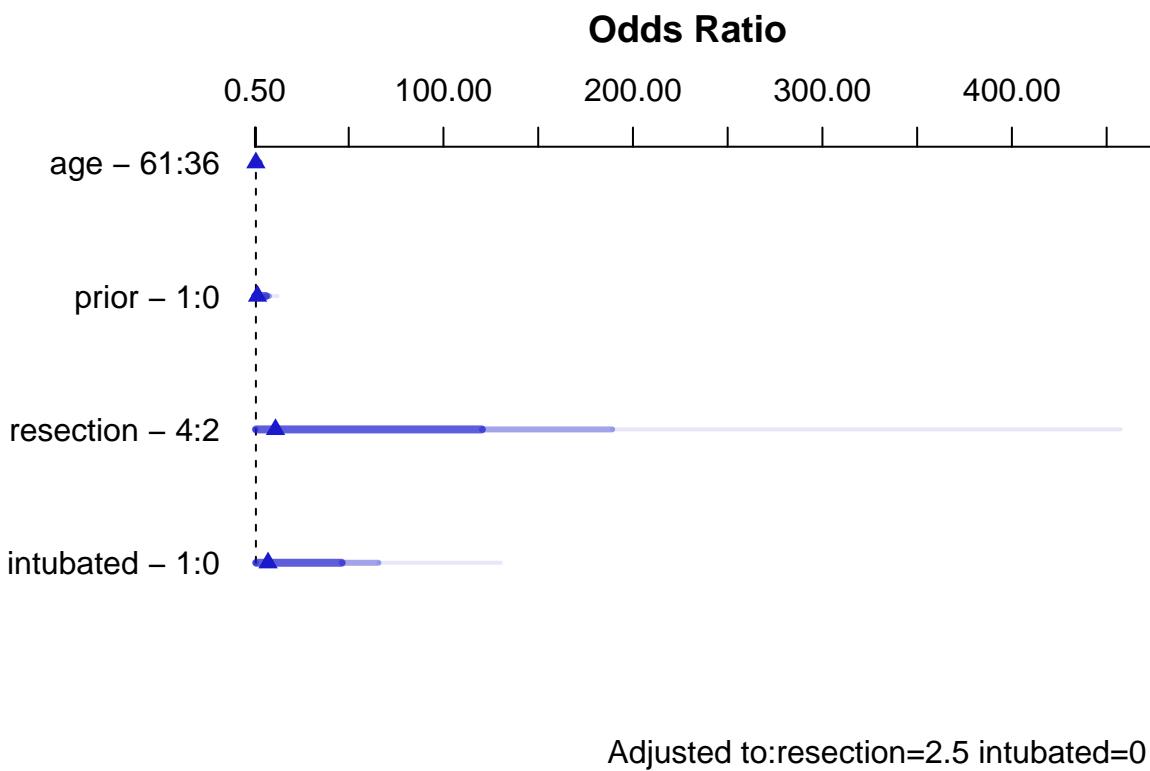
	Wald Statistics	Response: died	
Factor		Chi-Square	d.f.
age		0.00	1
prior		0.72	1
resection (Factor+Higher Order Factors)		4.95	3
All Interactions		0.72	1
Nonlinear		2.43	1
intubated (Factor+Higher Order Factors)		16.45	2
All Interactions		0.72	1
intubated * resection (Factor+Higher Order Factors)		0.72	1
TOTAL NONLINEAR + INTERACTION		2.56	2
TOTAL		23.90	6
P			
0.9988			
0.3947			
0.1753			
0.3970			
0.1194			
0.0003			
0.3970			
0.3970			

```
0.2783
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.

13.13.5 Effect Sizes in Model D

```
plot(summary(res_modD))
```



```
summary(res_modD)
```

Factor	Effects			Response : died					
	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	

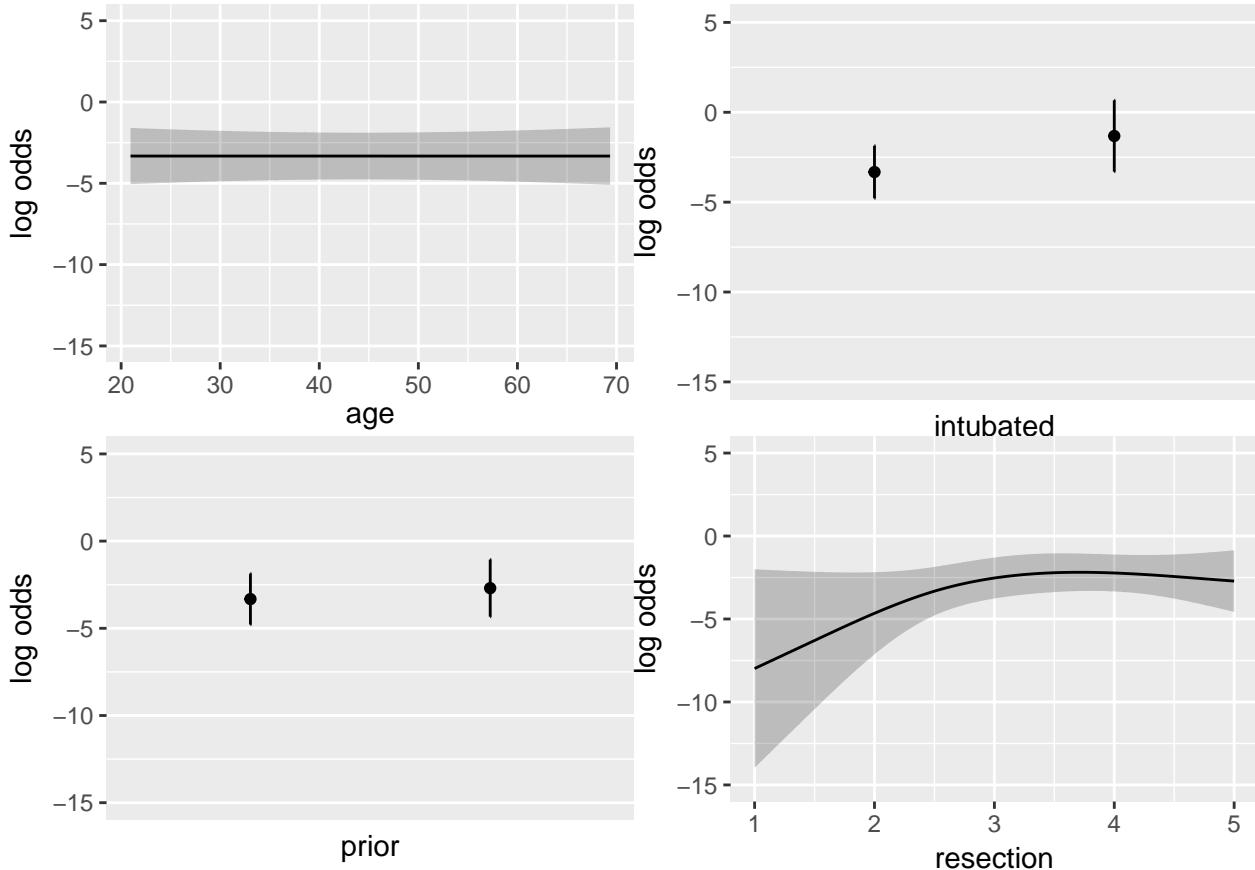
Adjusted to: resection=2.5 intubated=0

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`.

13.13.6 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.

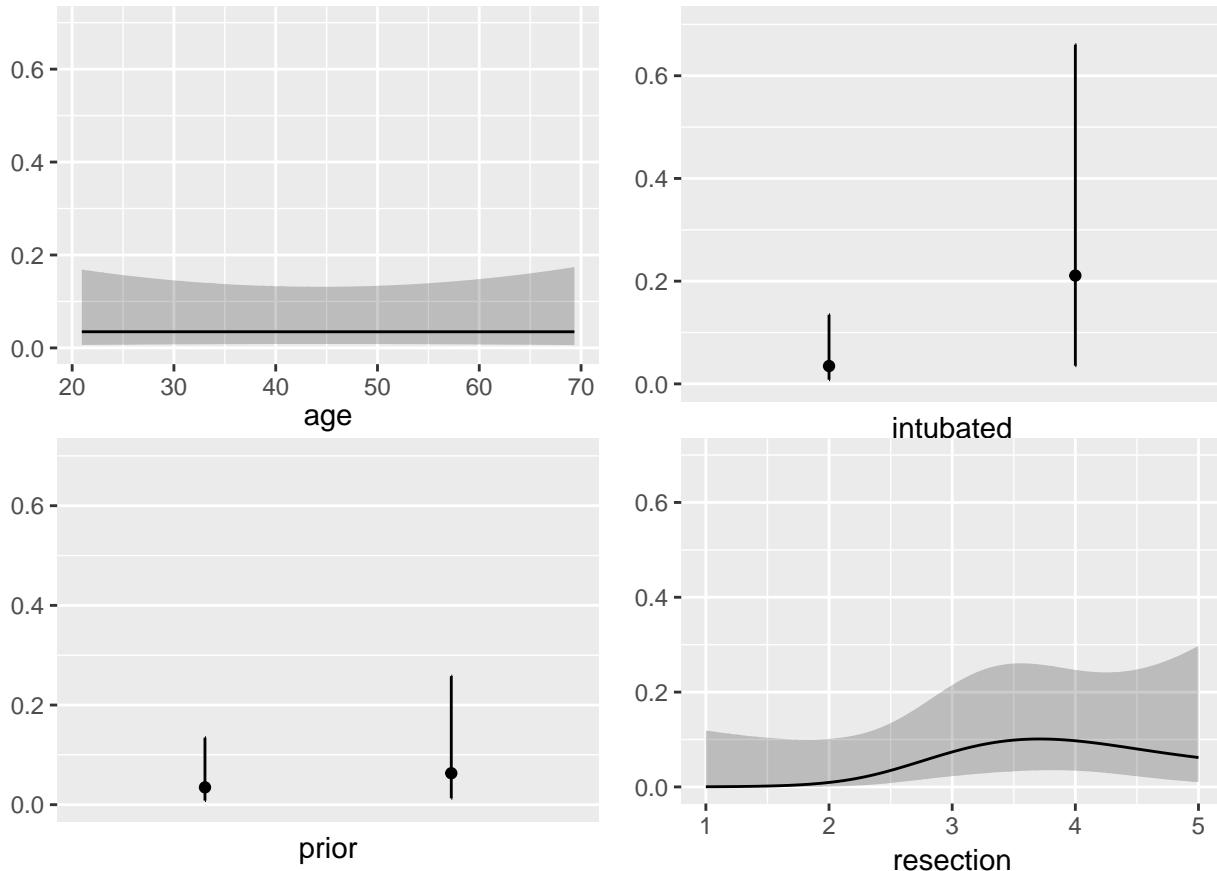
```
ggplot(Predict(res_modD))
```



We can also capture and plot these results on the probability scale, as follows¹.

```
ggplot(Predict(res_modD, fun = plogis))
```

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.



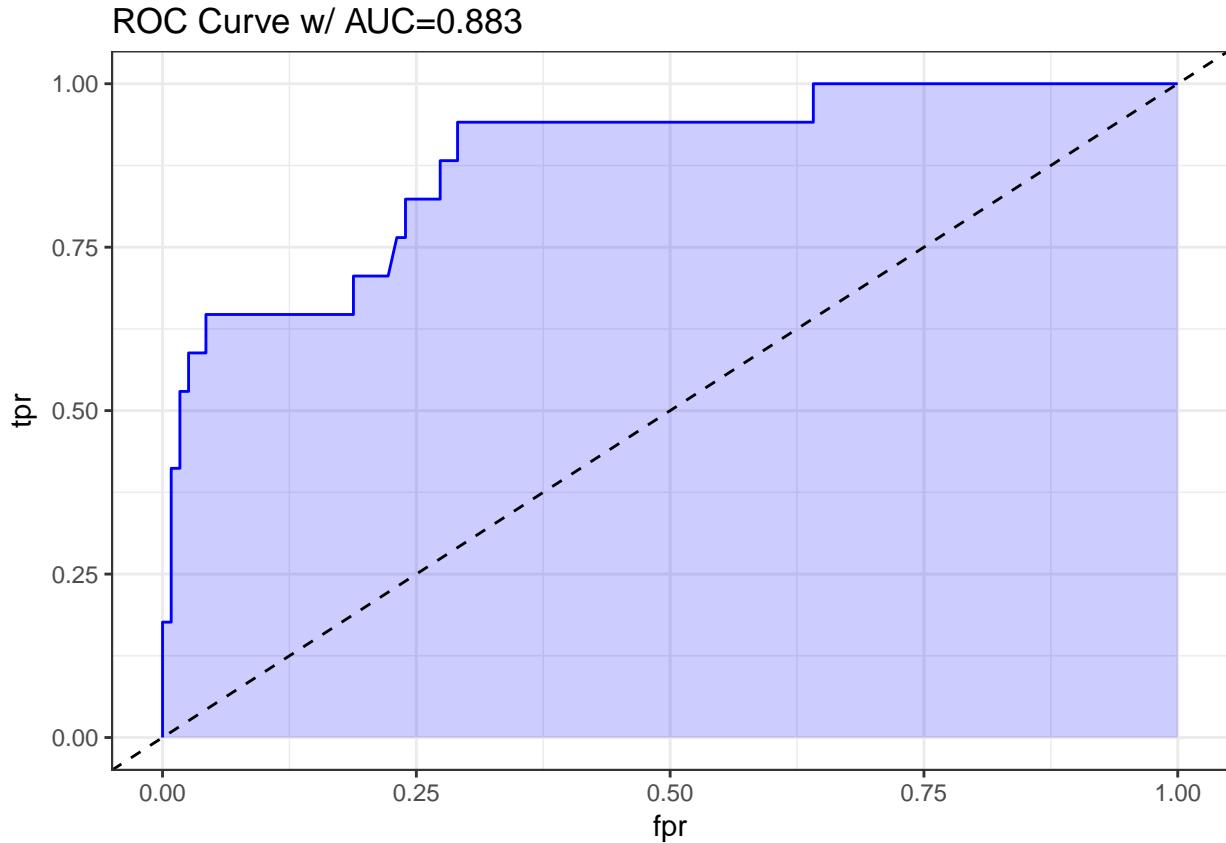
13.13.7 Plotting the ROC curve for Model D

Again, remember to use `type = "fitted"` with a `lrm` fit.

```
## 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:

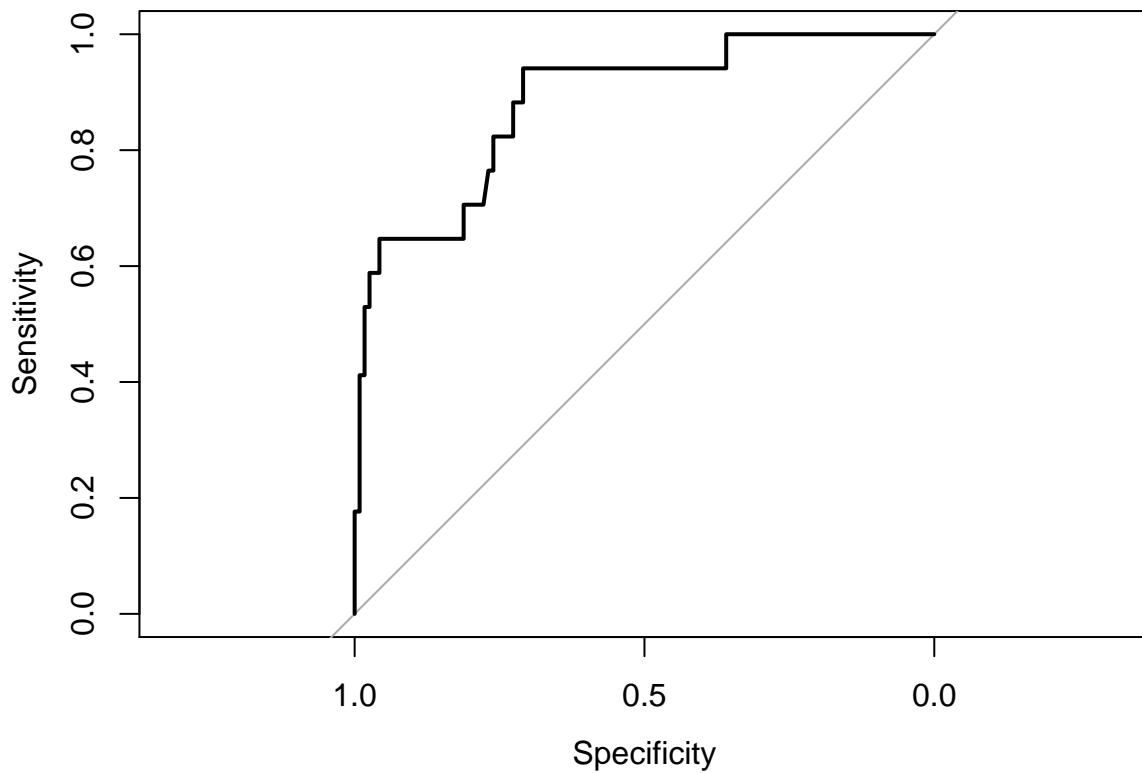
```
roc.formula(formula = resect$died ~ predict(res_modD, type = "fitted"),      ci = TRUE)
```

Data: `predict(res_modD, type = "fitted")` in 117 controls (`resect$died 0`) < 17 cases (`resect$died 1`).

Area under the curve: 0.8826

95% CI: 0.7952–0.97 (DeLong)

```
plot(roc.modD)
```



13.13.8 Validation of Model D summaries

```
set.seed(432002)
validate(res_modD, B = 100)
```

Divergence or singularity in 6 samples

	index.orig	training	test	optimism	index.corrected	n
Dxy	0.7652	0.8162	0.7142	0.1020	0.6632	94
R2	0.4643	0.5382	0.3991	0.1391	0.3252	94
Intercept	0.0000	0.0000	-0.3919	0.3919	-0.3919	94
Slope	1.0000	1.0000	0.7201	0.2799	0.7201	94
Emax	0.0000	0.0000	0.1530	0.1530	0.1530	94
D	0.2767	0.3258	0.2322	0.0936	0.1831	94
U	-0.0149	-0.0149	0.1998	-0.2147	0.1998	94
Q	0.2916	0.3407	0.0324	0.3083	-0.0167	94
B	0.0673	0.0595	0.0739	-0.0144	0.0817	94
g	2.3819	4.2214	2.2024	2.0190	0.3629	94
gp	0.1720	0.1777	0.1591	0.0186	0.1534	94

The C statistic indicates fairly strong discrimination, at $C = 0.88$, although after validation, this looks much weaker (based on $Dxy = 0.6632$, we would have $C = 0.5 + 0.6632/2 = 0.83$) 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 0.33.

13.14 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.

```
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 $-4.637 + 2.864 \text{ intubated} + 0.548 \text{ resection}$.

This implies that:

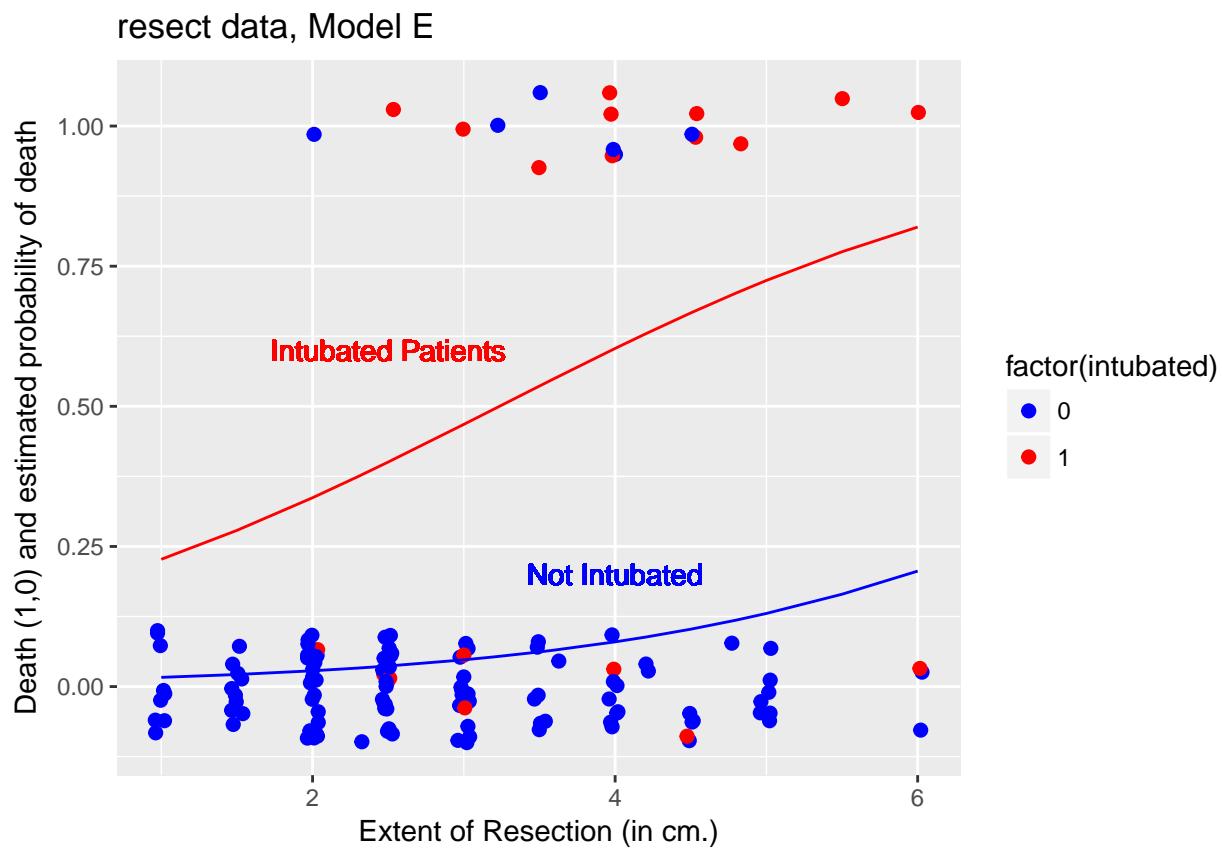
- for intubated patients, the equation is $-1.773 + 0.548 \text{ resection}$, while
- for non-intubated patients, the equation is $-4.637 + 0.548 \text{ resection}$.

We can use the `ilogit` function within the `faraway` package to help plot this.

13.14.1 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")
```

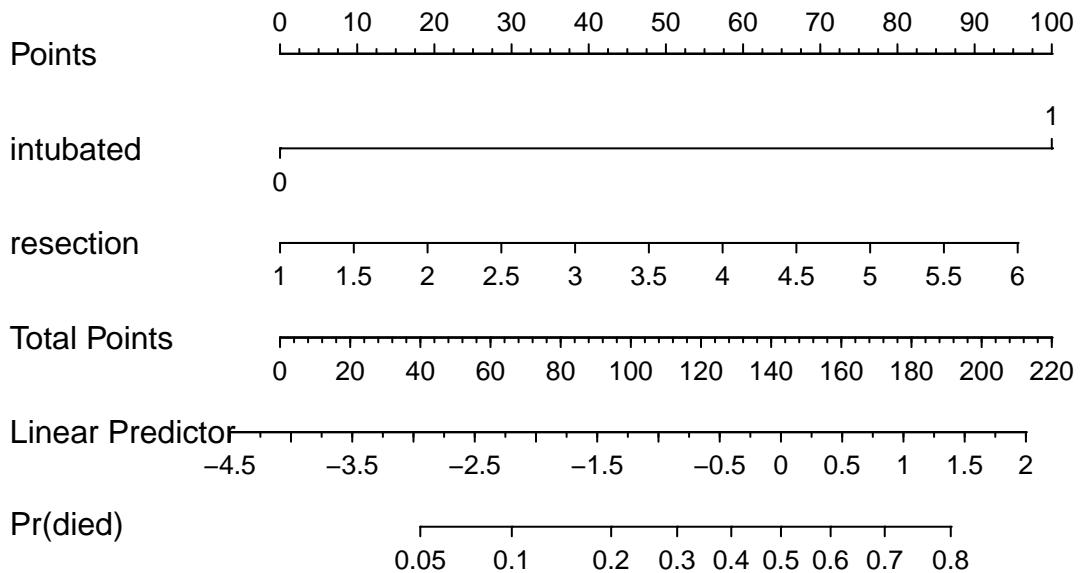


The effect of intubation appears to be very large, compared to the resection size effect.

13.14.2 Nomogram for Model E

A nomogram of the model would help, too.

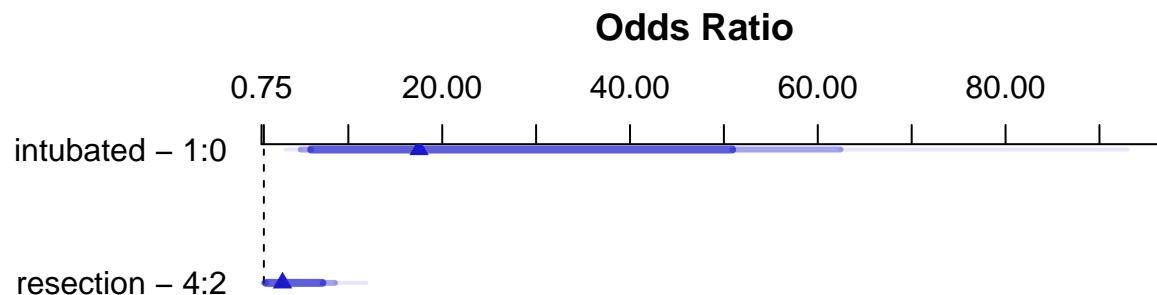
```
plot(nomogram(res_modE, fun=plogis,
              fun.at=c(0.05, seq(0.1, 0.9, by=0.1), 0.95),
              funlabel="Pr(died)"))
```



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.

13.14.3 Effect Sizes from Model E

```
plot(summary(res_modE))
```



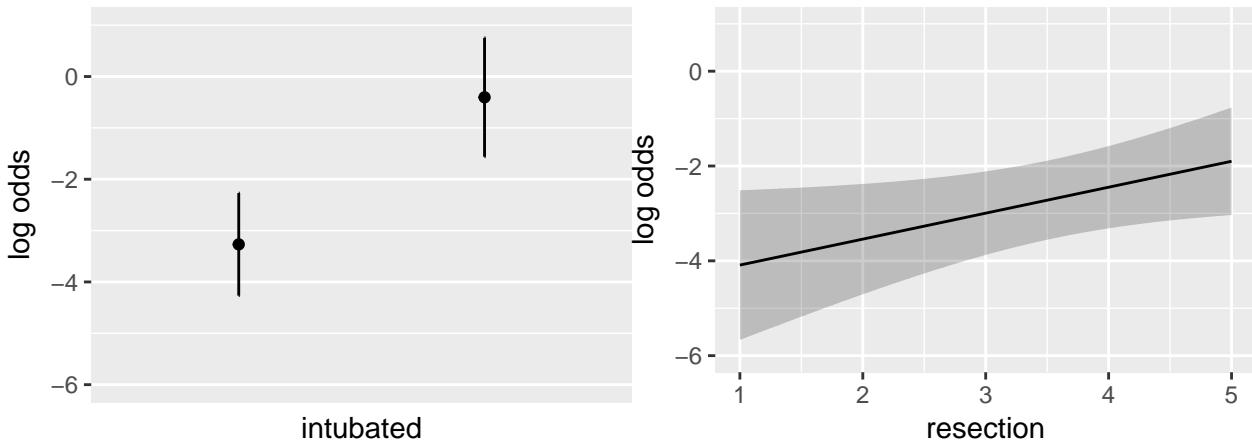
```
summary(res_modE)
```

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		

13.14.4 Plot In-Sample Predictions for Model E

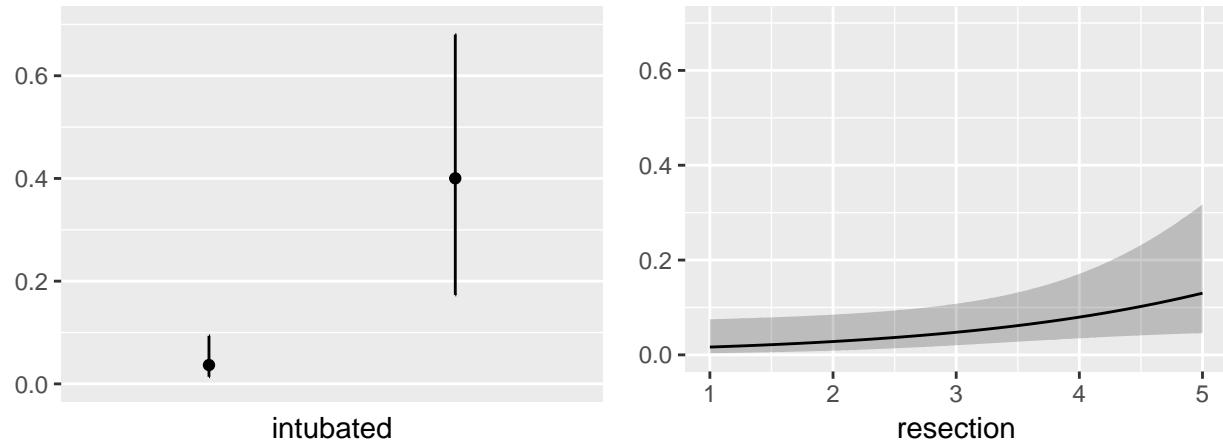
Here are plots of the effects across the range of each predictor (holding the other constant) on the log odds scale.

```
ggplot(Predict(res_modE))
```



We can also capture and plot these results on the probability scale, as follows.

```
ggplot(Predict(res_modE, fun = plogis))
```



13.14.5 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	

13.14.6 Validation of Model E

```
validate(res_modE, method="boot", B=40)
```

	index.orig	training	test	optimism	index.corrected	n
Dxy	0.7340	0.7113	0.7327	-0.0213	0.7554	40
R2	0.4128	0.4072	0.4039	0.0033	0.4095	40
Intercept	0.0000	0.0000	0.0346	-0.0346	0.0346	40
Slope	1.0000	1.0000	1.0128	-0.0128	1.0128	40
Emax	0.0000	0.0000	0.0096	0.0096	0.0096	40
D	0.2408	0.2444	0.2348	0.0096	0.2313	40
U	-0.0149	-0.0149	0.0023	-0.0173	0.0023	40
Q	0.2558	0.2593	0.2325	0.0269	0.2289	40

B	0.0727	0.0737	0.0762	-0.0025	0.0751	40
g	1.3970	1.4016	1.3642	0.0374	1.3596	40
gp	0.1597	0.1590	0.1568	0.0022	0.1575	40

Our bootstrap validated assessments of discrimination and goodness of fit look somewhat more reasonable now.

13.14.7 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"

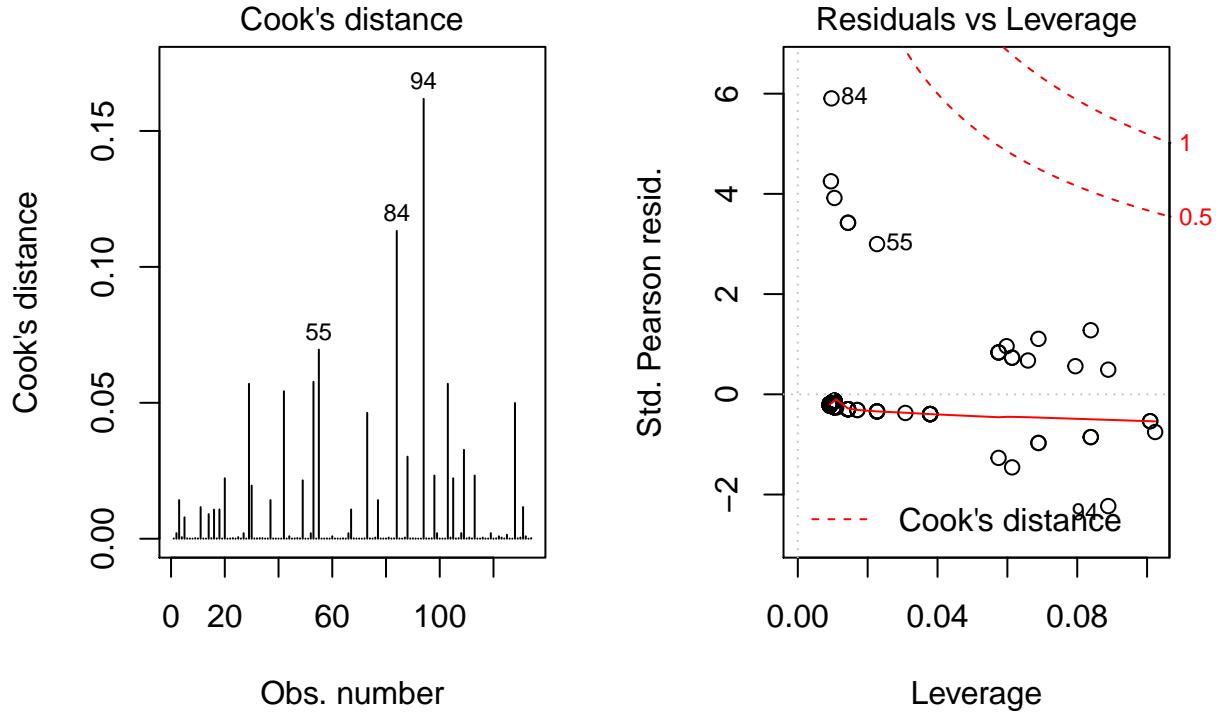
$resection
[1] "84" "94"

show.influence(inf.E, dframe = data.frame(resect))
```

```
Count resection
84      2      *2
94      2      *6
```

13.14.8 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.

13.15 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
  <int> <int> <int>    <dbl>    <int> <int>    <dbl>    <dbl>    <dbl>
1     1    34      1     2.50      0     0 0.0705 0.0632 0.0367
2     2    57      0     5.00      0     0 0.326 0.0620 0.130
3     3    60      1     4.00      1     1 0.187 0.791 0.603
4     4    62      1     4.20      0     0 0.211 0.158 0.0881
5     5    28      0     6.00      1     1 0.504 0.711 0.819
6     6    52      0     3.00      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)
```

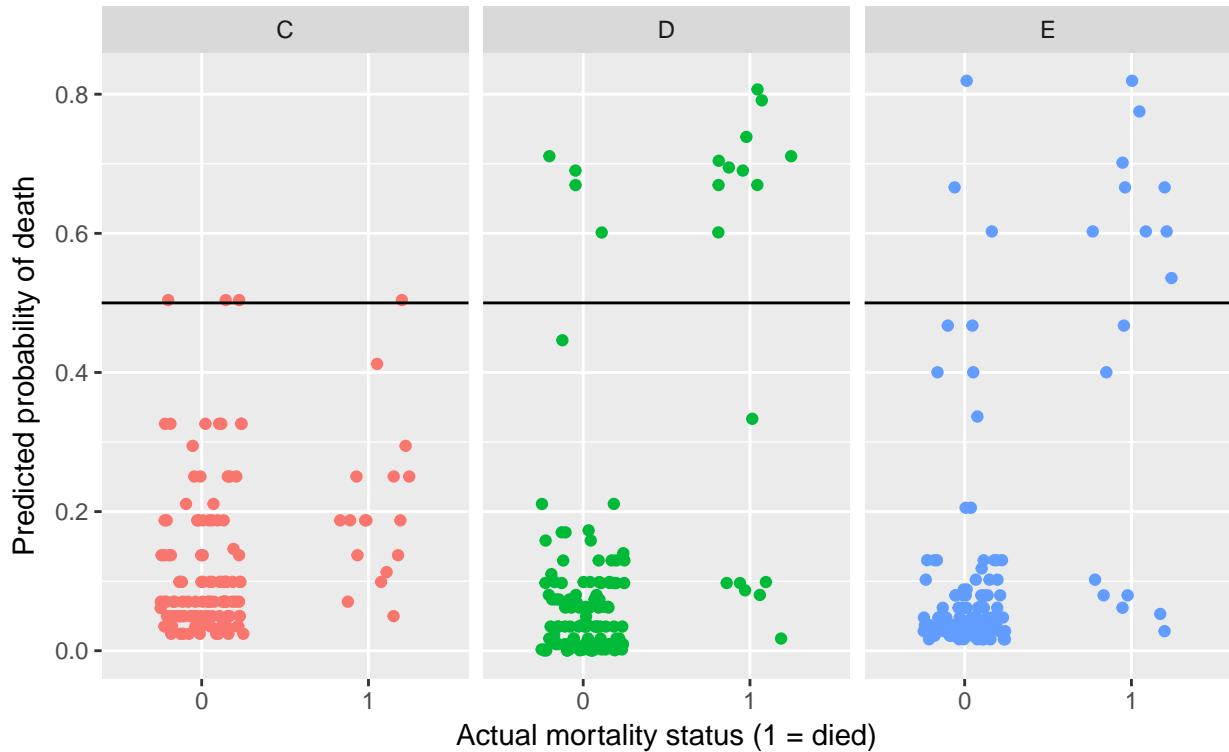
```
# A tibble: 6 x 4
  id   died model prediction
  <int> <int> <chr>     <dbl>
1     1     0 C        0.0705
2     2     0 C        0.326 
3     3     1 C        0.187 
4     4     0 C        0.211 
5     5     1 C        0.504 
6     6     0 C        0.0990
```

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")
```

Comparing Predictions for our Three Models

A graphical view of concordance



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.

13.16 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 14

Logistic Regression and the `smartcle1` data

14.1 The `smartcle1` data

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

Variable	Description
<code>SEQNO</code>	respondent identification number (all begin with 2016)
<code>physhealth</code>	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?
<code>menthealth</code>	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?
<code>poorhealth</code>	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?
<code>genhealth</code>	Would you say that in general, your health is ... (five categories: Excellent, Very Good, Good, Fair or Poor)
<code>bmi</code>	Body mass index, in kg/m ²
<code>female</code>	Sex, 1 = female, 0 = male
<code>internet30</code>	Have you used the internet in the past 30 days? (1 = yes, 0 = no)
<code>exerany</code>	During the past month, other than your regular job, did you participate in any physical activities or exercises such as running, calisthenics, golf, gardening, or walking for exercise? (1 = yes, 0 = no)
<code>sleephrs</code>	On average, how many hours of sleep do you get in a 24-hour period?
<code>alcdays</code>	How many days during the past 30 days did you have at least one drink of any alcoholic beverage such as beer, wine, a malt beverage or liquor?

In this section, we'll use some of the variables described above to predict the binary outcome: `exerany`.

14.2 Thinking About Non-Linear Terms

We have enough observations here to consider some non-linearity for our model.

In addition, since the `genhealth` variable is an ordinal variable and multi-categorical, we should consider how to model it. We have three options:

1. include it as a factor in the model (the default approach)
2. build a numeric version of that variable, and then restrict our model to treat that numeric variable as ordinal (forcing the categories to affect the `exerany` probabilities in an ordinal way), rather than as a simple nominal factor (so that if the effect of fair vs. good was to decrease the probability of ‘exerany’, then the effect of poor vs. good would have to decrease the probability at least as much as fair vs. good did.) Treating the `genhealth` variable as ordinal could be accomplished with the `scored` function in the `rms` package.
3. build a numeric version of `genhealth` and then use the `catg` function to specify the predictor as nominal and categorical, but this will lead to essentially the same model as choice 1.

Suppose we’ve decided to treat the `genhealth` data as categorical, without restricting the effect of its various levels to be ordinal. Suppose also that we’ve decided to include the following seven variables in our model for `exerany`:

- `physhealth`
- `menthealth`
- `genhealth`
- `bmi`
- `female`
- `internet30`
- `sleephrs`

Suppose we have a subject matter understanding that:

- the impact of `bmi` on `exerany` is affected by `female`, so we plan a `female` x `bmi` interaction term
- we’re using `internet30` as a proxy for poverty, and we think that an interaction with self-reported `genhealth` will be helpful in our model as well.

Note that we do have some missing values in some of these predictors, so we’ll have to deal with that soon.

```
smartcle1 %>% select(exerany, physhealth, menthealth,
                      genhealth, bmi, female, internet30,
                      sleephrs) %>%
  skim()
```

```
Skim summary statistics
n obs: 1036
n variables: 8

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

Variable type: integer
  variable missing complete    n mean    sd p0 p25 median p75 p100
  exerany        3       1033 1036 0.76 0.43  0   1      1   1   1
  female         0       1036 1036 0.6  0.49  0   0      1   1   1
  internet30     6       1030 1036 0.81 0.39  0   1      1   1   1
  menthealth     11      1025 1036 2.72 6.82  0   0      0   2   30
  physhealth     17      1019 1036 3.97 8.67  0   0      0   2   30
```

```

sleephrs      8     1028 1036 7.02 1.53  1   6     7   8   20
Variable type: numeric
variable missing complete    n  mean   sd   p0  p25 median   p75  p100
bmi        84     952 1036 27.89 6.47 12.71 23.7  26.68 30.53 66.06

```

14.3 A First Model for exerany (Complete Case Analysis)

Suppose we develop a main-effects kitchen sink model (model `m1` below) fitted to these predictors without the benefit of any non-linear terms except the two pre-planned interactions. We'll run the model quickly here to ensure that the code runs, in a complete case analysis, without drawing any conclusions, really.

```

m1 <- lrm(exerany ~ internet30 * genhealth + bmi * female +
            physhealth + menthealth + sleephrs,
            data = smartcle1)
m1

```

```

Frequencies of Missing Values Due to Each Variable
exerany internet30  genhealth          bmi      female  physhealth
      3           6           3         84           0          17
menthealth    sleephrs
      11          8

```

Logistic Regression Model

```

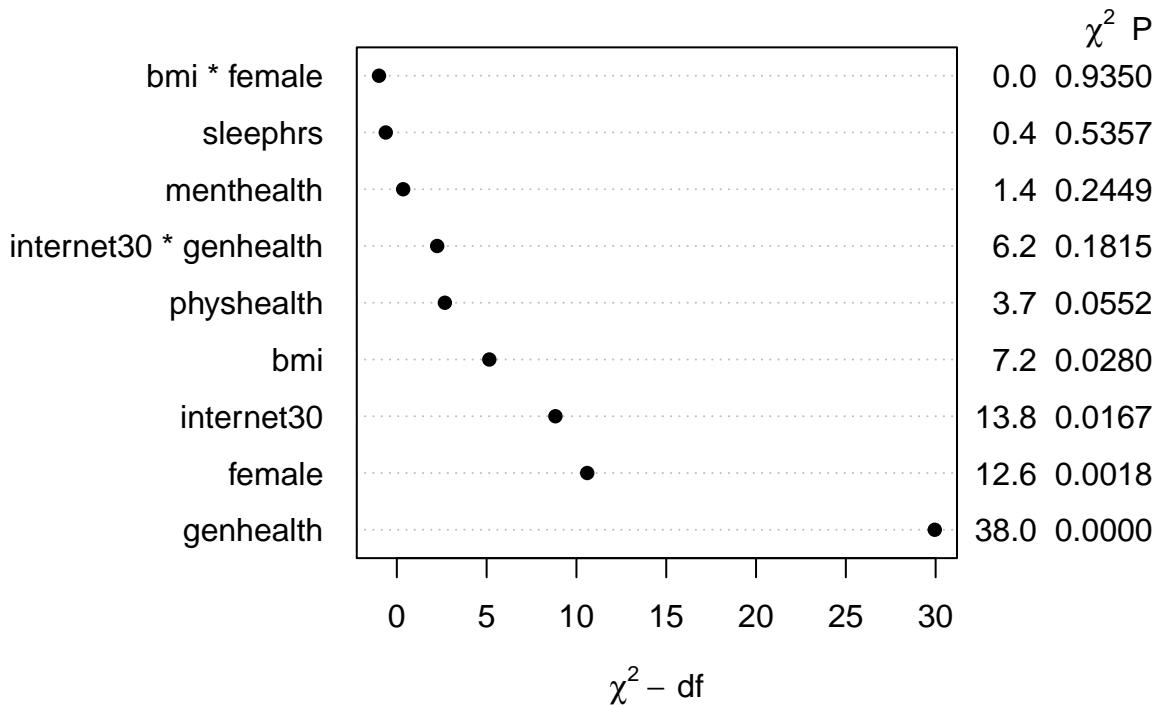
lrm(formula = exerany ~ internet30 * genhealth + bmi * female +
    physhealth + menthealth + sleephrs, data = smartcle1)

```

	Obs	Model Likelihood			Discrimination		Rank Discrim.	
		Ratio	Test		Indexes		Indexes	
0	924	LR	chi2	134.17	R2	0.203	C	0.739
1	219	d.f.		15	g	1.021	Dxy	0.479
	705	Pr(> chi2)	<0.0001		gr	2.776	gamma	0.479
					gp	0.174	tau-a	0.173
					Brier	0.153		

	Coef	S.E.	Wald Z	Pr(> Z)
Intercept	2.9284	0.9951	2.94	0.0033
internet30	0.7234	0.7095	1.02	0.3079
genhealth=2_VeryGood	-0.2267	0.7653	-0.30	0.7671
genhealth=3_Good	-0.7950	0.7100	-1.12	0.2629
genhealth=4_Fair	-1.2818	0.7482	-1.71	0.0867
genhealth=5_Poor	-0.5215	0.8590	-0.61	0.5438
bmi	-0.0323	0.0220	-1.47	0.1427
female	-0.5789	0.7883	-0.73	0.4627
physhealth	-0.0208	0.0109	-1.92	0.0552
menthealth	-0.0142	0.0122	-1.16	0.2449
sleephrs	-0.0338	0.0547	-0.62	0.5357
internet30 * genhealth=2_VeryGood	0.1524	0.8295	0.18	0.8542
internet30 * genhealth=3_Good	-0.0064	0.7702	-0.01	0.9934
internet30 * genhealth=4_Fair	-0.3235	0.8272	-0.39	0.6957
internet30 * genhealth=5_Poor	-1.7847	0.9994	-1.79	0.0741
bmi * female	-0.0022	0.0266	-0.08	0.9350

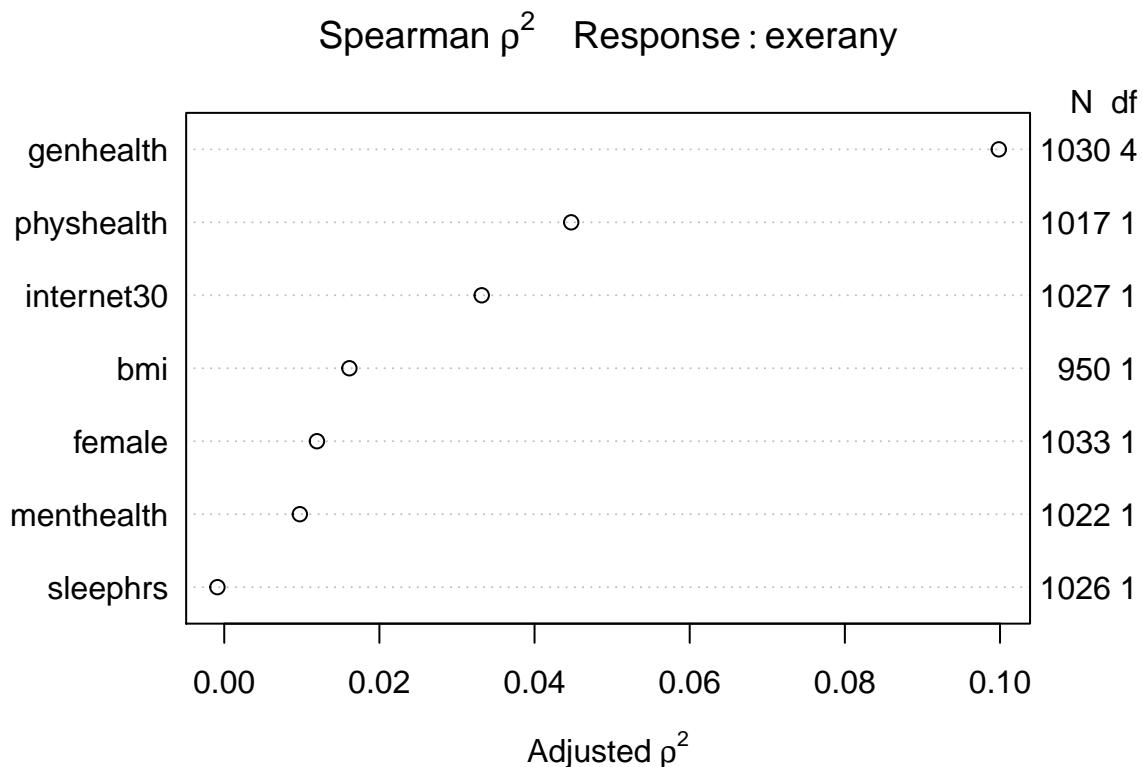
```
plot(anova(m1))
```



14.4 Building a Larger Model: Spearman ρ^2 Plot

Before we impute, we might also consider the use of a Spearman ρ^2 plot to decide how best to spend degrees of freedom on non-linear terms in our model for `exerany` using these predictors. Since we're already planning some interaction terms, I'll keep them in mind as I look at this plot.

```
sp_smart <- spearman2(exerany ~ physhealth + menthealth +  
                      genhealth + internet30 +  
                      bmi + female + sleephrs,  
                      data = smartcle1)  
  
plot(sp_smart)
```



We see that the best candidate for a non-linear term is the `genhealth` variable, according to this plot, followed by the `physhealth` and `internet30` predictors, then `bmi`. I will wind up fitting a model including the following non-linear terms...

- our pre-planned `female` x `bmi` and `internet30` x `genhealth` interaction terms,
- a new `genhealth` x `physhealth` interaction term,
- a restricted cubic spline with 5 knots for `physhealth`
- a restricted cubic spline with 4 knots for `bmi` (so the interaction term with `female` will need to account for this and restrict our interaction to the linear piece of `bmi`)

14.5 A Second Model for `exerany` (Complete Cases)

Here's the resulting model fit without worrying about imputation yet. This is just to make sure our code works. Note that I'm inserting the main effects of our interaction terms explicitly before including the interaction terms themselves, and that I need to use `%ia%` to include the interaction terms where one of the terms is included in the model with a spline. Again, I won't draw any serious conclusions yet.

```
m2 <- lrm(exerany ~ rcs(bmi, 4) + rcs(physhealth, 5) +
            female + internet30 * genhealth +
            genhealth %ia% physhealth + female %ia% bmi +
            menthealth + sleephrs,
            data = smartcle1)
m2
```

Frequencies of Missing Values Due to Each Variable
 exerany bmi physhealth female internet30 genhealth

3	84	17	0
menthealth	sleephrs		6
11	8		3

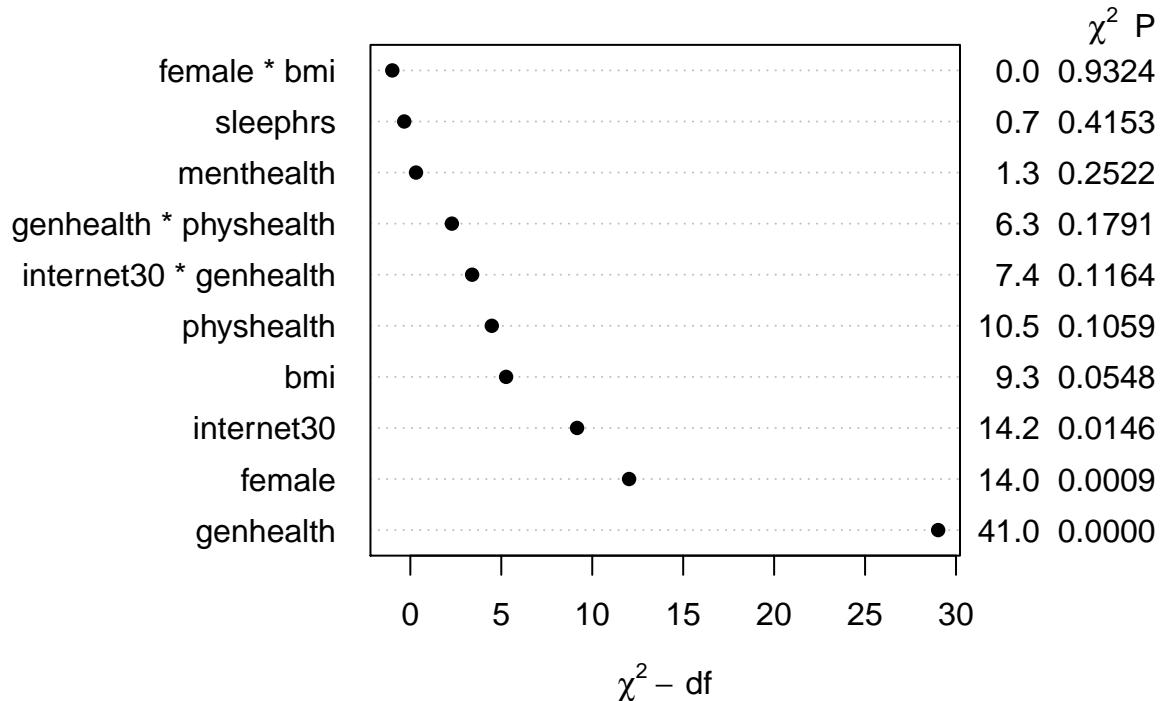
Logistic Regression Model

```
lrm(formula = exerany ~ rcs(bmi, 4) + rcs(physhealth, 5) + female +
internet30 * genhealth + genhealth %ia% physhealth + female %ia%
bmi + menthealth + sleephrs, data = smartcle1)
```

	Model	Likelihood		Discrimination		Rank Discrim.	
		Ratio	Test		Indexes		Indexes
Obs	924	LR	chi2	142.51	R2	0.215	C 0.743
0	219	d.f.		22	g	1.053	Dxy 0.485
1	705	Pr(> chi2)	<0.0001		gr	2.867	gamma 0.485
max deriv	6e-13				gp	0.178	tau-a 0.176
					Brier	0.151	

	Coef	S.E.	Wald Z	Pr(> Z)
Intercept	2.2998	1.9789	1.16	0.2452
bmi	0.0052	0.0801	0.06	0.9482
bmi'	-0.2988	0.3555	-0.84	0.4006
bmi''	0.8809	0.9615	0.92	0.3596
physhealth	0.0410	0.0868	0.47	0.6362
physhealth'	-0.2944	0.3888	-0.76	0.4489
female	-0.6307	0.8228	-0.77	0.4434
internet30	0.6722	0.7134	0.94	0.3460
genhealth=2_VeryGood	-0.3092	0.7700	-0.40	0.6880
genhealth=3_Good	-0.7559	0.7194	-1.05	0.2934
genhealth=4_Fair	-1.1611	0.7646	-1.52	0.1289
genhealth=5_Poor	-2.3680	1.2846	-1.84	0.0653
genhealth=2_VeryGood * physhealth	0.0099	0.0824	0.12	0.9043
genhealth=3_Good * physhealth	-0.0406	0.0772	-0.53	0.5992
genhealth=4_Fair * physhealth	-0.0407	0.0773	-0.53	0.5988
genhealth=5_Poor * physhealth	0.0534	0.0868	0.62	0.5382
female * bmi	-0.0023	0.0275	-0.08	0.9324
menthealth	-0.0143	0.0125	-1.15	0.2522
sleephrs	-0.0457	0.0561	-0.81	0.4153
internet30 * genhealth=2_VeryGood	0.2174	0.8330	0.26	0.7941
internet30 * genhealth=3_Good	0.0375	0.7767	0.05	0.9615
internet30 * genhealth=4_Fair	-0.3080	0.8371	-0.37	0.7129
internet30 * genhealth=5_Poor	-1.9411	1.0150	-1.91	0.0558

```
plot(anova(m2))
```



14.6 Dealing with Missing Data via Simple Imputation

One approach we might take in this problem is to use simple imputation to deal with our missing values. I will proceed as follows:

1. Omit all cases where the outcome `exerany` is missing.
2. Determine (and plot) the remaining missingness.
3. Use simple imputation for all predictors, and build a new data set with “complete” data.
4. Re-fit the proposed models using this new data set.

14.6.1 Omit cases where the outcome is missing

We need to drop the cases where `exerany` is missing in `smartcle1`. We'll begin creating an imputed data set, called `smartcle_imp0`, by filtering on complete data for `exerany`, as follows.

```
Hmisc::describe(smartcle1$exerany)
```

```
smartcle1$exerany
  n  missing distinct      Info      Sum      Mean      Gmd
  1033      3        2    0.546     786  0.7609  0.3642
```

```
smartcle_imp0 <- smartcle1 %>%
  filter(complete.cases(exerany)) %>%
  select(SEQNO, exerany, physhealth, menthealth,
         genhealth, bmi, female, internet30, sleephrs)
```

```
Hmisc::describe(smartcle_imp0$exerany)
```

smartcle_imp0\$exerany						
n	missing	distinct	Info	Sum	Mean	Gmd
1033	0	2	0.546	786	0.7609	0.3642

14.6.2 Plot the remaining missingness

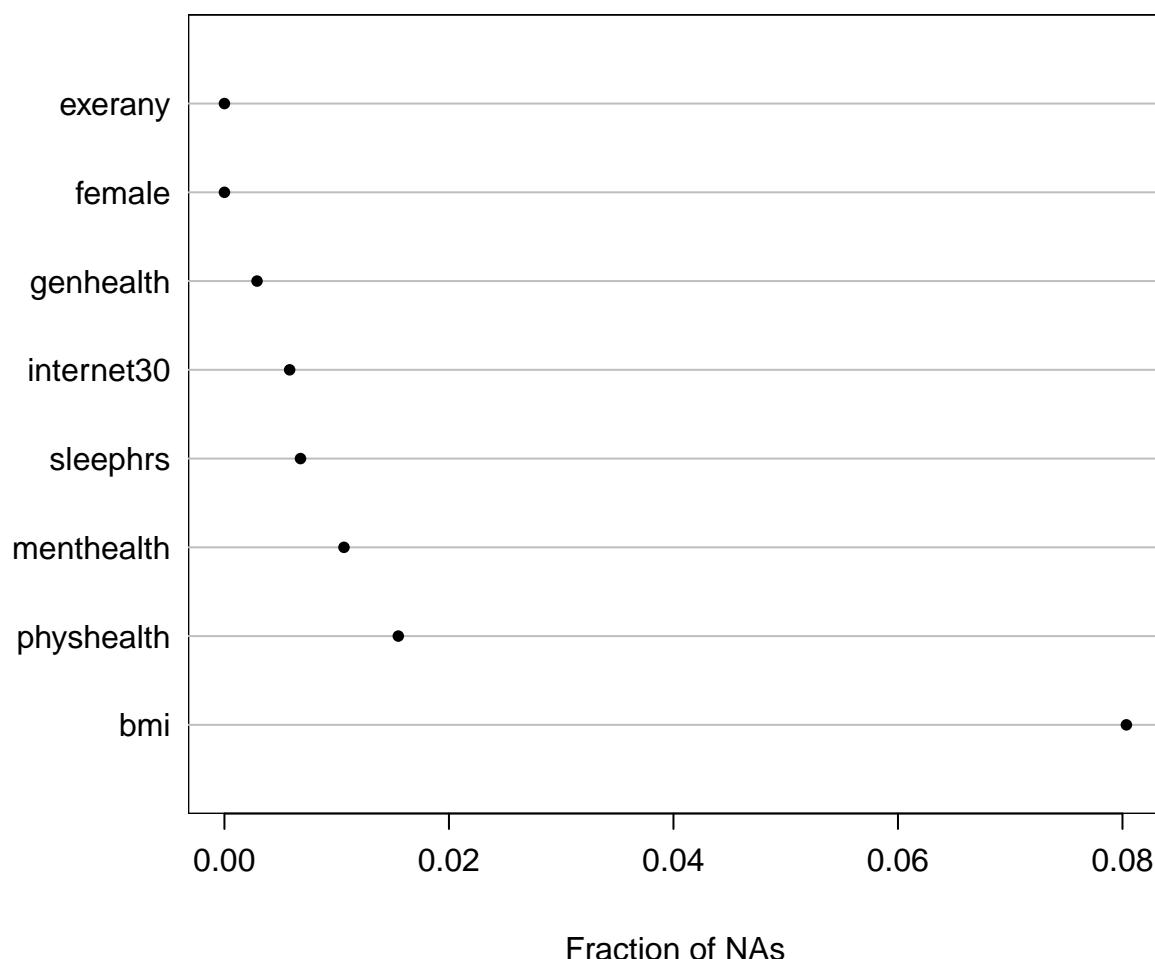
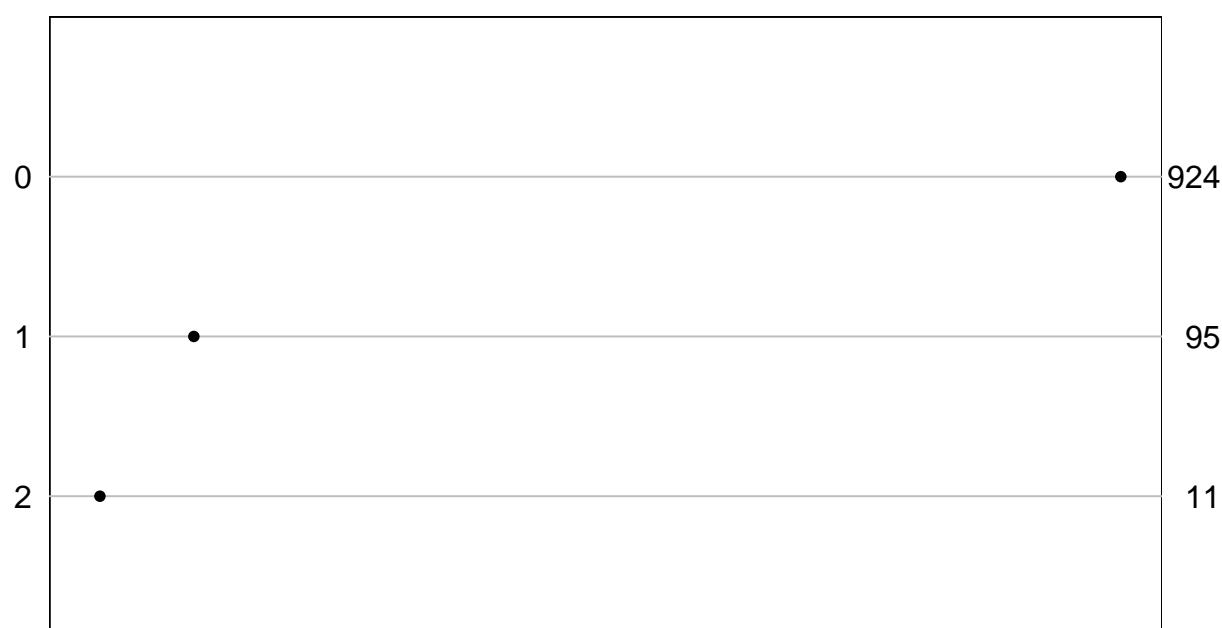
We'll look at the missing values (excluding the subject ID: SEQNO) in our new data set. Of course, we can get a count of missing values within each variable with `skim` or with:

```
colSums(is.na(smartcle_imp0))
```

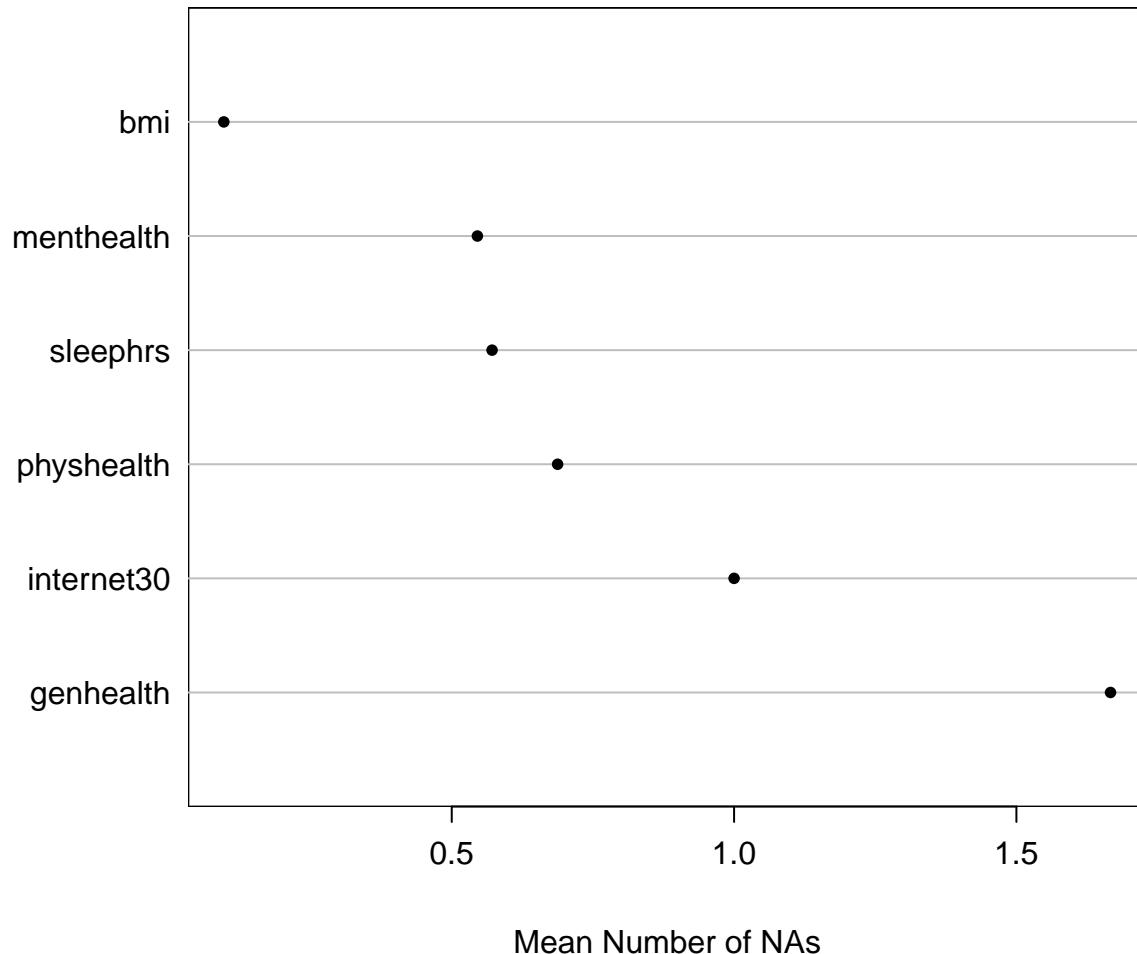
SEQNO	exerany	physhealth	menthealth	genhealth	bmi
0	0	16	11	3	83
female	internet30	sleephrs			
0	6	7			

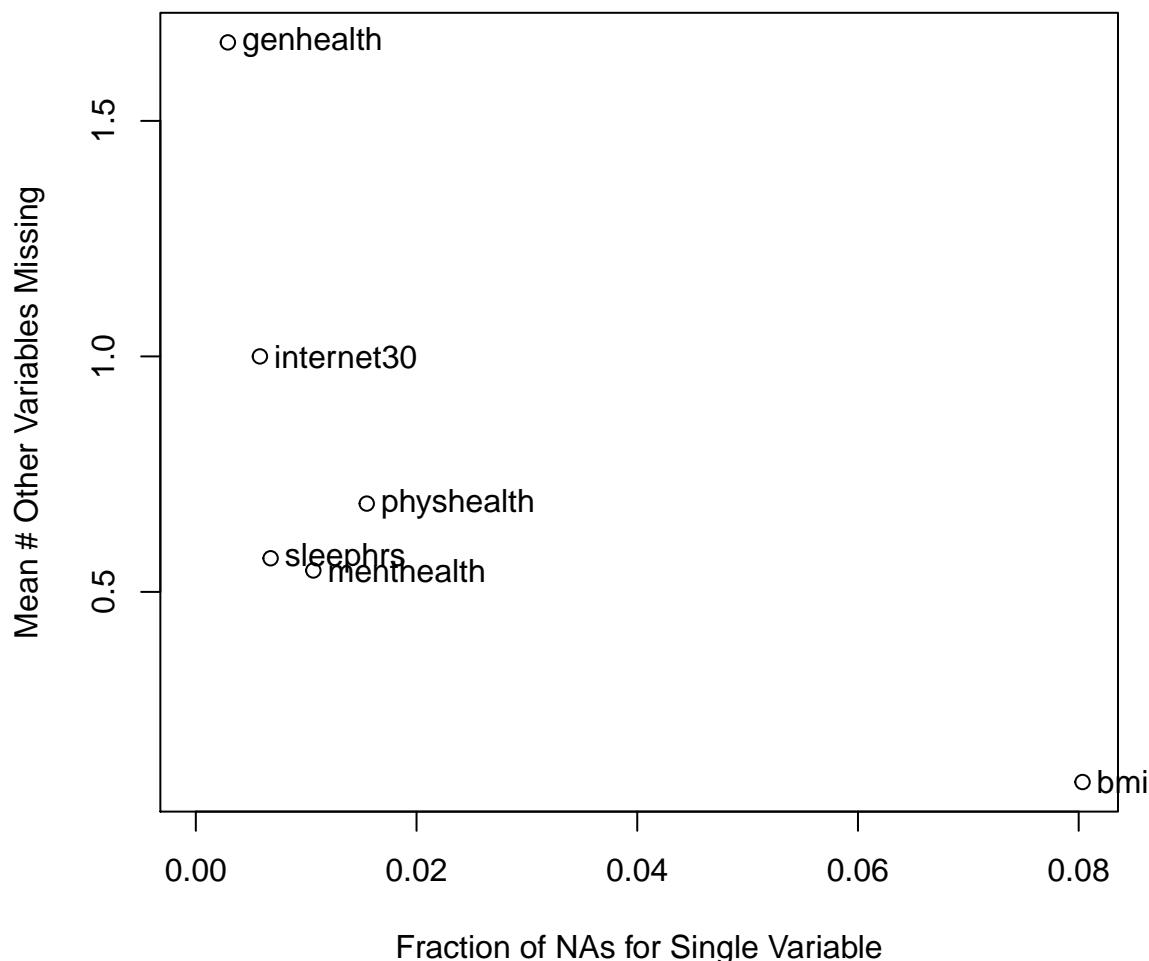
The `Hmisc` package has a plotting approach which can help identify missingness, too.

```
naplot(naclus(select(smartcle_imp0, -SEQNO)))
```

Fraction of NAs in each Variable**Number of Missing Variables Per Observation**

Mean Number of Other Variables Missing for Observations where Indicated Variable is NA





We can also get a useful accounting of missing data patterns, with the `md.pattern` function in the `mice` package.

```
mice::md.pattern(smartcle_imp0)
```

SEQNO	exerany	female	genhealth	internet30	sleephrs	menthealth
924	1	1	1	1	1	1
8	1	1	1	1	1	1
6	1	1	1	1	1	1
76	1	1	1	1	1	1
1	1	1	1	1	0	1
4	1	1	1	1	1	0
3	1	1	1	1	1	0
1	1	1	1	0	1	1
1	1	1	1	1	1	1
4	1	1	1	1	0	1
1	1	1	1	1	1	0
1	1	1	1	1	1	1

```

1   1   1   1   0   1   1   0
1   1   1   1   1   0   1   1
1   1   1   1   0   1   0   1
0   0   0   3   6   7   11

  physhealth bmi
924      1   1   0
8       0   1   1
6       1   1   1
76      1   0   1
1       1   1   1
4       1   1   1
3       0   1   2
1       1   1   2
1       0   0   2
4       1   0   2
1       0   1   2
1       1   0   2
1       0   1   3
1       0   0   3
1       0   1   3
16    83  126

```

We can also do this with `na.pattern` in the `Hmisc` package, but then we have to get the names of the columns, too, so that we can read off the values.

```
na.pattern(smartcle_imp0)
```

```

pattern
0000000000 0000000001 0000000010 000001000 000001001 000001010 000100000
         924        4           1          76           1           4           6
000110000 001000000 001000001 001001000 001001010 001010001 001100000
         1         8           1           1           1           1           3
001110000
         1

```

```
names(smartcle_imp0)
```

```

[1] "SEQNO"      "exerany"     "physhealth"  "menthealth"  "genhealth"
[6] "bmi"        "female"      "internet30" "sleephrs"

```

14.6.3 Use simple imputation, build a new data set

The only variables that require no imputation are `exerany` and `female`. In this case, we need to impute:

- 83 `bmi` values (which are quantitative)
- 16 `physhealth` values (quantitative, must fall between 0 and 30)
- 11 `menthealth` values (quantitative, must fall between 0 and 30)
- 7 `sleephrs` values (quantitative, must fall between 0 and 24)
- 6 `internet30` values (which are 1/0)
- and 3 `genhealth` values (which are multi-categorical, so we need to convert them to numbers in order to get the imputation process to work properly)

```

smartcle_imp0 <- smartcle_imp0 %>%
  mutate(genh_n = as.numeric(genhealth))

smartcle_imp0 %>% count(genhealth, genh_n)

```

```
# A tibble: 6 x 3
  genhealth  genh_n     n
  <fct>      <dbl> <int>
1 1_Excellent    1.    172
2 2_VeryGood     2.    350
3 3_Good         3.    344
4 4_Fair          4.    121
5 5_Poor          5.     43
6 <NA>            NA     3
```

I'll work from the bottom up, using various `simputation` functions to accomplish the imputations I want. In this case, I'll use predictive mean matching for the categorical data, and linear models or elastic net approaches for the quantitative data. Be sure to set a seed beforehand so you can replicate your work.

```
set.seed(432234)
```

```
smartcle_imp1 <- smartcle_imp0 %>%
  impute_pmm(genh_n ~ female) %>%
  impute_pmm(internet30 ~ female + genh_n) %>%
  impute_lm(sleephrs ~ female + genh_n) %>%
  impute_lm(menthealth ~ female + sleephrs) %>%
  impute_en(phshealth ~ female + genh_n + sleephrs) %>%
  impute_en(bmi ~ phshealth + genh_n)
```

After the imputations are complete, I'll back out of the numeric version of `genhealth`, called `genh_n` back to my original variable, then check to be sure I now have no missing values.

```
smartcle_imp1 <- smartcle_imp1 %>%
  mutate(genhealth = fct_recode(factor(genh_n),
                                "1_Excellent" = "1",
                                "2_VeryGood" = "2",
                                "3_Good" = "3",
                                "4_Fair" = "4",
                                "5_Poor" = "5"))

smartcle_imp1 %>% count(genhealth, genh_n)
```

```
# A tibble: 5 x 3
  genhealth  genh_n     n
  <fct>      <dbl> <int>
1 1_Excellent    1.    172
2 2_VeryGood     2.    351
3 3_Good         3.    346
4 4_Fair          4.    121
5 5_Poor          5.     43

colSums(is.na(smartcle_imp1))
```

SEQNO	exerany	phshealth	menthealth	genhealth	bmi
0	0	0	0	0	0
female	internet30	sleephrs	genh_n		
0	0	0	0		

OK. Looks good. I now have a data frame called `smartcle_imp1` with no missingness, which I can use to fit my logistic regression models. Let's do that next, and then return to the problem of accounting for missingness through multiple imputation.

14.7 Refitting Model 1 with simply imputed data

Using the numeric version of the `genhealth` data, called `genh_n`, will ease the reviewing of later output, so we'll do that here, making sure R knows that `genh_n` describes a categorical factor.

```
d <- datadist(smartcle_imp1)
options(datadist = "d")

m1_a <- lrm(exerany ~ internet30 * catg(genh_n) + bmi * female +
              physhealth + menthealth + sleephrs,
              data = smartcle_imp1, x = TRUE, y = TRUE)
m1_a
```

Logistic Regression Model

```
lrm(formula = exerany ~ internet30 * catg(genh_n) + bmi * female +
    physhealth + menthealth + sleephrs, data = smartcle_imp1,
    x = TRUE, y = TRUE)
```

	Model Likelihood		Discrimination		Rank Discrim.	
	Ratio	Test		Indexes		Indexes
Obs	1033	LR	chi2	151.09	R2	0.204
0	247	d.f.		15	g	1.023
1	786	Pr(> chi2)	<0.0001		gr	2.780
max deriv	5e-14				gp	0.175
					Brier	0.154

	Coef	S.E.	Wald Z	Pr(> Z)
Intercept	3.3390	0.9731	3.43	0.0006
internet30	0.6487	0.7010	0.93	0.3548
genh_n=2	-0.4938	0.7459	-0.66	0.5080
genh_n=3	-0.9270	0.6972	-1.33	0.1836
genh_n=4	-1.3060	0.7378	-1.77	0.0767
genh_n=5	-0.8093	0.8329	-0.97	0.3312
bmi	-0.0412	0.0217	-1.90	0.0570
female	-0.9473	0.7722	-1.23	0.2199
physhealth	-0.0229	0.0102	-2.25	0.0246
menthealth	-0.0200	0.0114	-1.76	0.0784
sleephrs	-0.0415	0.0492	-0.84	0.3994
internet30 * genh_n=2	0.3701	0.8065	0.46	0.6463
internet30 * genh_n=3	0.1295	0.7543	0.17	0.8637
internet30 * genh_n=4	-0.2194	0.8108	-0.27	0.7867
internet30 * genh_n=5	-1.4080	0.9636	-1.46	0.1440
bmi * female	0.0118	0.0261	0.45	0.6497

All right. We've used 1033 observations, which is correct (after deleting the three with missing `exerany`). The model shows a Nagelkerke R² value of 0.204, and a C statistic of 0.741 after imputation. The likelihood ratio (drop in deviance) test is highly significant.

14.7.1 Validating Summary Statistics

```
set.seed(432099)
validate(m1_a)
```

	index.orig	training	test	optimism	index.corrected	n
Dxy	0.4823	0.5012	0.4632	0.0380	0.4444	40
R2	0.2039	0.2218	0.1878	0.0340	0.1699	40
Intercept	0.0000	0.0000	0.0936	-0.0936	0.0936	40
Slope	1.0000	1.0000	0.9001	0.0999	0.9001	40
Emax	0.0000	0.0000	0.0401	0.0401	0.0401	40
D	0.1453	0.1592	0.1329	0.0264	0.1189	40
U	-0.0019	-0.0019	0.0013	-0.0032	0.0013	40
Q	0.1472	0.1612	0.1316	0.0295	0.1177	40
B	0.1539	0.1507	0.1566	-0.0058	0.1597	40
g	1.0226	1.0868	0.9722	0.1146	0.9080	40
gp	0.1751	0.1818	0.1673	0.0145	0.1606	40

It appears that the model's description of summary statistics is a little optimistic for both the C statistic (remember that $C = 0.5 + Dxy/2$) and the Nagelkerke R^2 . This output suggests that in a new sample of data, our model might be better expected to show a C statistic near ...

$$C = 0.5 + \frac{Dxy}{2} = 0.5 + \frac{0.4444}{2} = 0.7222$$

rather than the 0.741 we saw initially, and that the Nagelkerke R^2 in new data will be closer to 0.17, than to the nominal 0.204 we saw above. So, as we walk through some other output for this model, remember that the C statistic wasn't great here (0.72 after validation), so our ability to discriminate exercisers from non-exercisers is still a problem.

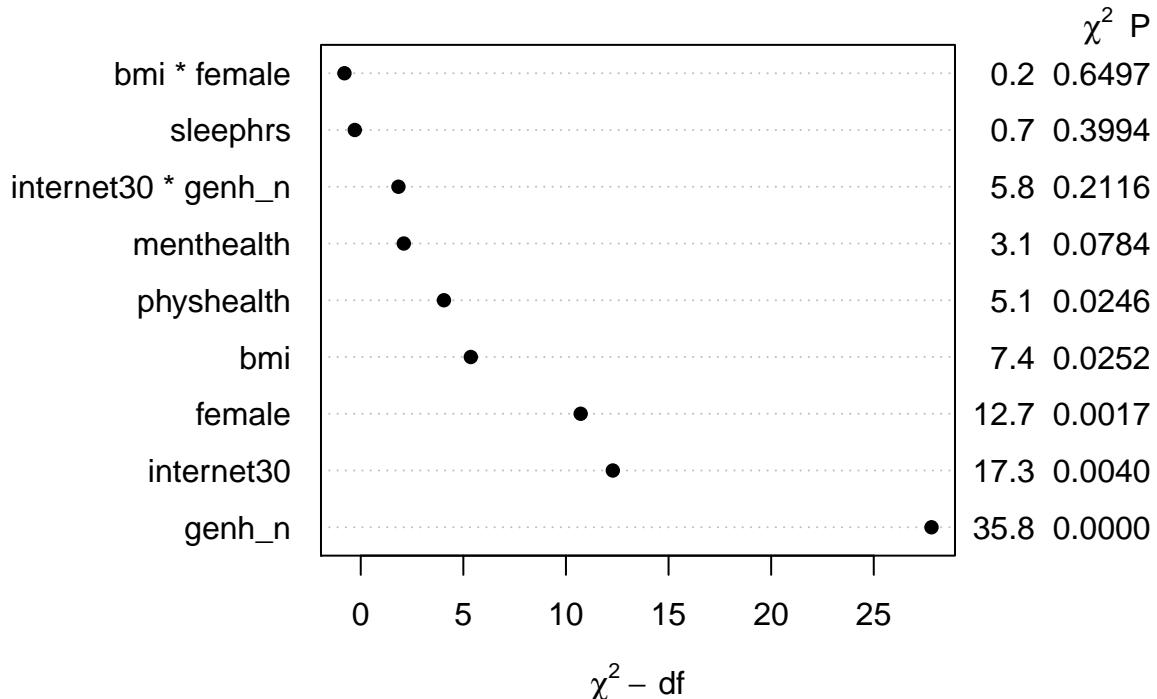
14.7.2 ANOVA for the model

Next, let's look at the ANOVA comparisons for this model.

```
anova(m1_a)
```

	Wald Statistics	Response: exerany		
Factor		Chi-Square	d.f.	P
internet30 (Factor+Higher Order Factors)		17.28	5	0.0040
All Interactions		5.84	4	0.2116
genh_n (Factor+Higher Order Factors)		35.81	8	<.0001
All Interactions		5.84	4	0.2116
bmi (Factor+Higher Order Factors)		7.36	2	0.0252
All Interactions		0.21	1	0.6497
female (Factor+Higher Order Factors)		12.72	2	0.0017
All Interactions		0.21	1	0.6497
physhealth		5.05	1	0.0246
menthealth		3.10	1	0.0784
sleephrs		0.71	1	0.3994
internet30 * genh_n (Factor+Higher Order Factors)		5.84	4	0.2116
bmi * female (Factor+Higher Order Factors)		0.21	1	0.6497
TOTAL INTERACTION		6.15	5	0.2917
TOTAL		126.95	15	<.0001

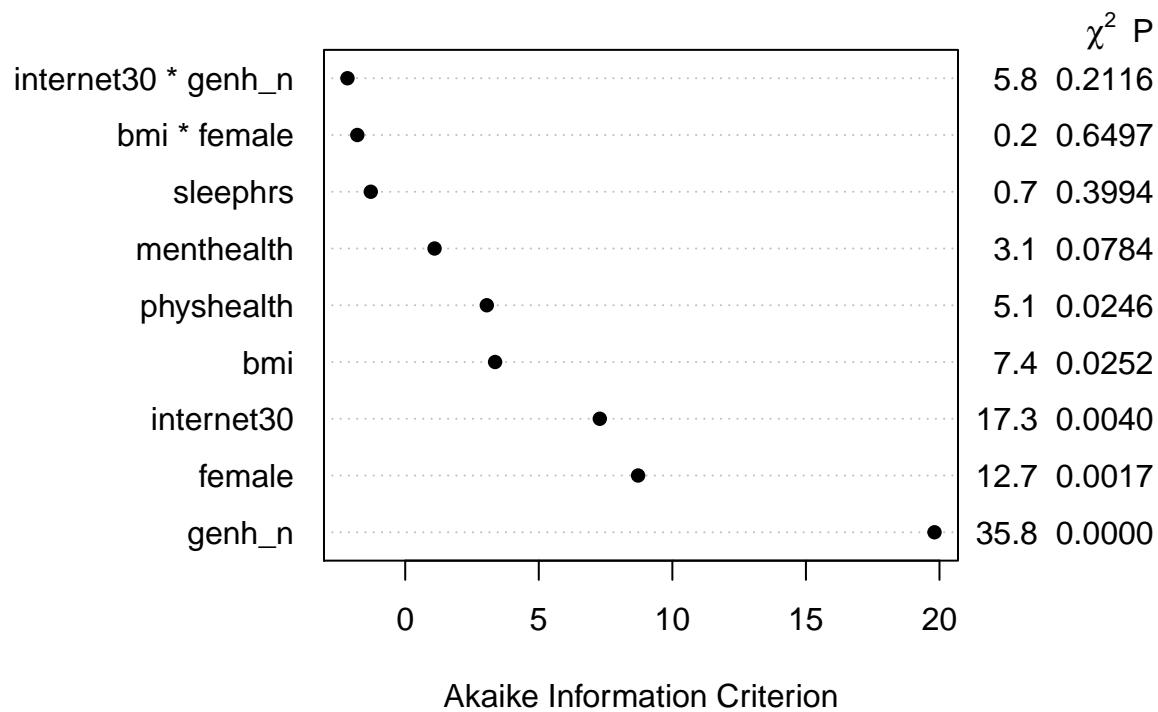
```
plot(anova(m1_a))
```



It looks like several of the variables (`genhealth`, `internet30`, `female`, `bmi` and `physhealth`) are carrying statistically significant predictive value here.

We can also build a plot of the AIC values attributable to each piece of the model.

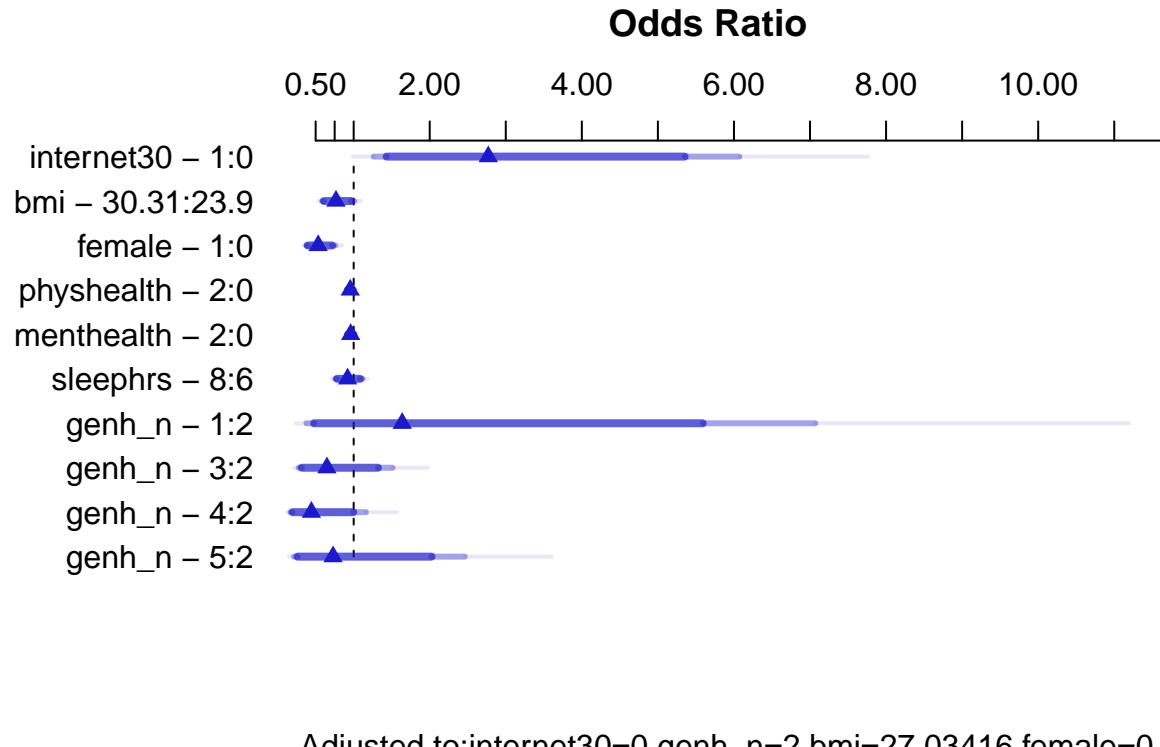
```
plot(anova(m1_a), what="aic")
```



14.7.3 Summarizing Effect Size

How big are the effects we see?

```
plot(summary(m1_a))
```



```
summary(m1_a)
```

Effects			Response : exerany						
Factor	Low	High	Diff.	Effect	S.E.	Lower	0.95	Upper	0.95
internet30	0.0	1.00	1.00	1.018800	0.400350	0.234160	1.8035000		
Odds Ratio	0.0	1.00	1.00	2.770000		NA	1.263800	6.0710000	
bmi	23.9	30.31	6.41	-0.264180	0.138810	-0.536240	0.0078783		
Odds Ratio	23.9	30.31	6.41	0.767840		NA	0.584950	1.0079000	
female	0.0	1.00	1.00	-0.627270	0.177840	-0.975830	-0.2787200		
Odds Ratio	0.0	1.00	1.00	0.534050		NA	0.376880	0.7567500	
physhealth	0.0	2.00	2.00	-0.045843	0.020393	-0.085813	-0.0058727		
Odds Ratio	0.0	2.00	2.00	0.955190		NA	0.917770	0.9941400	
menthealth	0.0	2.00	2.00	-0.040089	0.022778	-0.084733	0.0045549		
Odds Ratio	0.0	2.00	2.00	0.960700		NA	0.918760	1.0046000	
sleephrs	6.0	8.00	2.00	-0.082947	0.098439	-0.275880	0.1099900		
Odds Ratio	6.0	8.00	2.00	0.920400		NA	0.758900	1.1163000	
genh_n - 1:2	2.0	1.00	NA	0.493760	0.745930	-0.968240	1.9558000		
Odds Ratio	2.0	1.00	NA	1.638500		NA	0.379750	7.0693000	
genh_n - 3:2	2.0	3.00	NA	-0.433290	0.431840	-1.279700	0.4131100		
Odds Ratio	2.0	3.00	NA	0.648370		NA	0.278120	1.5115000	
genh_n - 4:2	2.0	4.00	NA	-0.812200	0.490270	-1.773100	0.1487200		
Odds Ratio	2.0	4.00	NA	0.443880		NA	0.169800	1.1603000	
genh_n - 5:2	2.0	5.00	NA	-0.315510	0.620110	-1.530900	0.8998700		
Odds Ratio	2.0	5.00	NA	0.729420		NA	0.216340	2.4593000	

```
Adjusted to: internet30=0 genh_n=2 bmi=27.03416 female=0
```

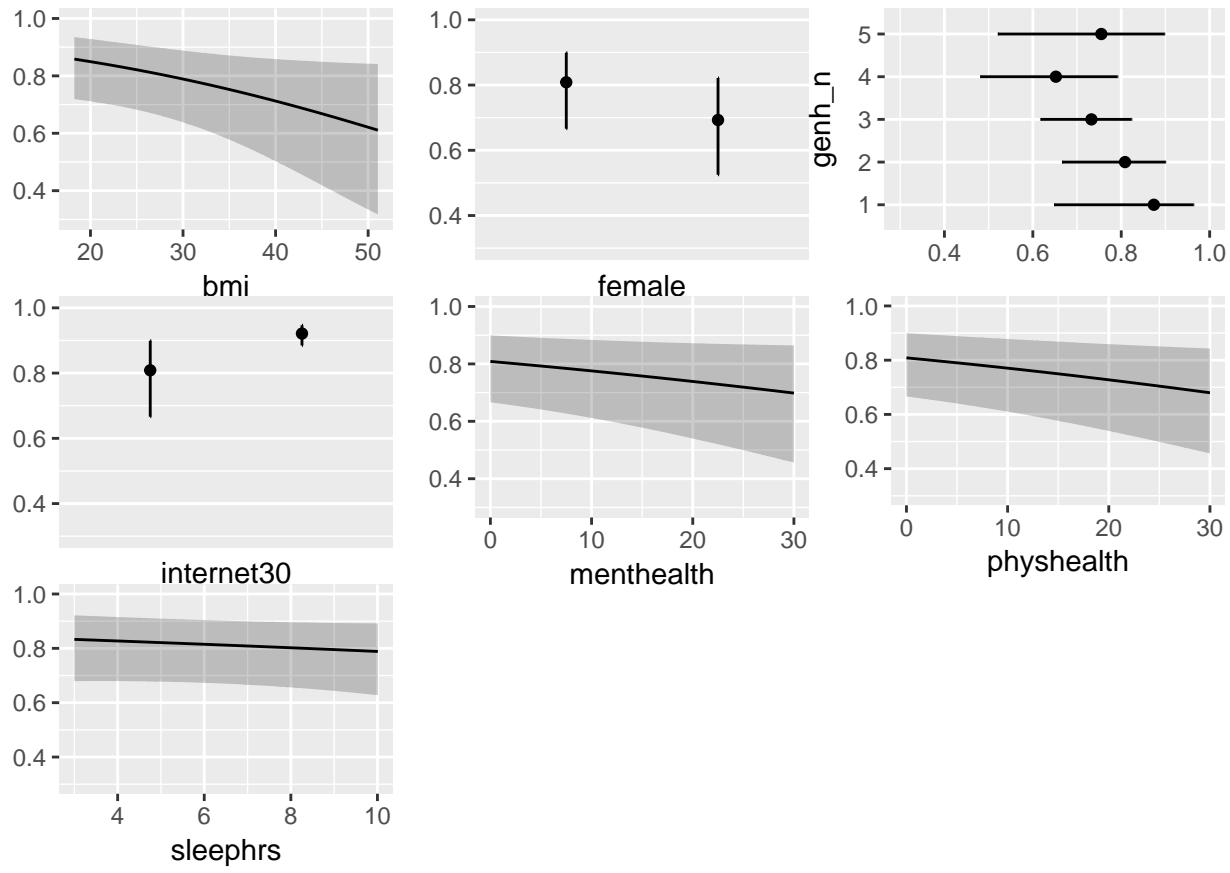
This output is easier to read as a result of using small *numeric* labels in `genh_n`, rather than the lengthy labels in `genhealth`. The sensible things to interpret are the odds ratios.

- holding all other predictors constant, the effect of moving from `internet30 = 0` to `internet30 = 1` is that the odds of `exerany` increase by a factor of 2.77.
 - Suppose Harry and Steve have the same values of all predictors in this model except that Harry used the internet and Steve did not.
 - So the odds of exercising for Harry (who used the internet) are 2.77 times higher than the odds of exercising for Steve (who didn't use the internet), assuming that all other predictors are the same.
 - We also have a 95% confidence interval for this odds ratio, which is (1.26, 6.07). Since 1 is not in that interval, the data don't seem to be consistent with `internet30` having no effect on `exerany`.
- the odds ratio comparing two subjects with the same predictors except that Harry has a BMI of 30.31 (the 75th percentile of observed BMIs in our sample) and Marty has a BMI of 23.9 (the 25th percentile) is that Harry has 0.767 times the odds of exercising that Marty does. So Harry's probability of exercise will also be lower than Marty's.
 - The 95% confidence interval in this case is (0.58, 1.01), and because 1 is in that interval, we cannot conclude that the effect of `bmi` meets the standard for statistical significance at the 5% level.
- A similar approach can be used to describe the odds ratios associated with each predictor.
- Note that each of the categories in `genh_n` is compared to a single baseline category. Here, that's category 2. R will pick the modal category: the one that appears most often in the data. The comparisons of each category against category 2 are not significant in each case, at the 5% level.

14.7.4 Plotting the Model with `ggplot` and `Predict`

Let's look at a series of plots describing the model on the probability scale.

```
ggplot(Predict(m1_a, fun = plogis))
```



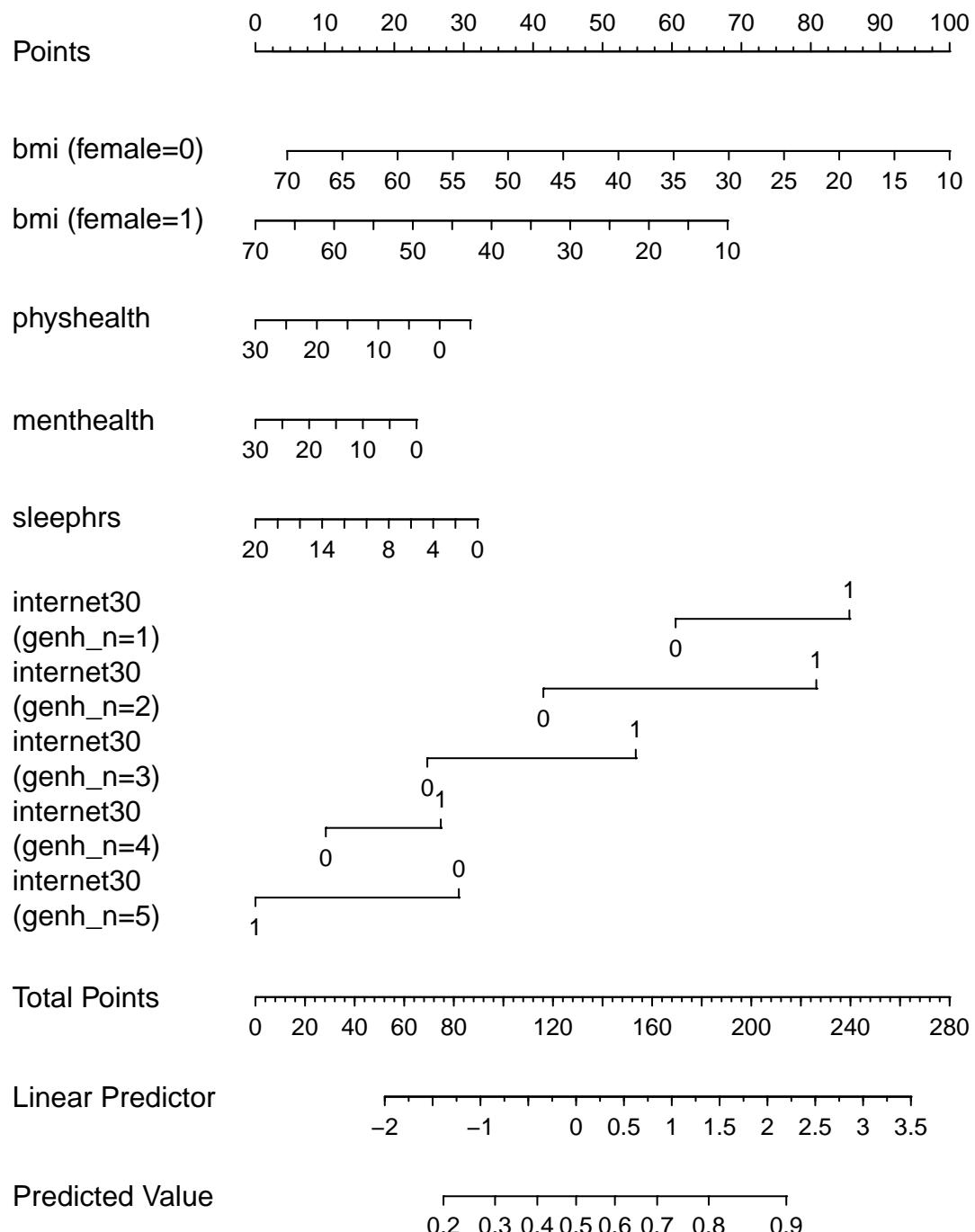
This helps us describe what is happening in terms of direction at least. For example,

- As `bmi` increases, predicted $\text{Pr}(\text{exerany})$ decreases.
- People who accessed the internet in the past 30 days have higher model probabilities of exercising.

Do any of these plots fail to make sense to you? Is anything moving in a surprising direction?

14.7.5 Plotting the model with a nomogram

```
plot(nomogram(m1_a, fun = plogis))
```



Note the impact of our interaction terms, and how we have two lines for `bmi` and five lines for `internet30` that come out of our product terms. As with any nomogram, to make a prediction we:

1. find the values of each of our predictors in the scales, and travel vertically up to the Points line to read off the Points for that predictor.
2. sum up the Points across all predictors, and find that location in the Total Points line.
3. move vertically down from the total points line to find the estimated “linear predictor” (log odds ratio) and finally the “predicted value” (probability of our outcome `exerany` = 1.)

14.7.6 Checking the Goodness of Fit of our model

To test the goodness of fit, we can use the following omnibus test:

```
round(residuals(m1_a, type = "gof"), 3)
```

Sum of squared errors	Expected value H0	SD
158.955	158.740	0.525
Z	P	
0.408	0.683	

Our non-significant *p* value suggests that we cannot detect anything that's obviously wrong in the model in terms of goodness of fit. That's comforting.

14.8 Refitting Model 2 with simply imputed data

I'll walk through the same tasks for Model `m2` that I did above for Model `m1`. Again, we're running this model after simple imputation of missing values.

Using the numeric version of the `genhealth` data, called `genh_n`, will ease the reviewing of later output, so we'll do that here, making sure R knows that `genh_n` describes a categorical factor.

```
d <- datadist(smartcle_imp1)
options(datadist = "d")

m2_a <- lrm(exerany ~ rcs(bmi, 4) + rcs(physhealth, 5) +
              female + internet30 * catg(genh_n) +
              catg(genh_n) %ia% physhealth + female %ia% bmi +
              menthealth + sleephrs,
              data = smartcle_imp1, x = TRUE, y = TRUE)
m2_a
```

Logistic Regression Model

```
lrm(formula = exerany ~ rcs(bmi, 4) + rcs(physhealth, 5) + female +
    internet30 * catg(genh_n) + catg(genh_n) %ia% physhealth +
    female %ia% bmi + menthealth + sleephrs, data = smartcle_imp1,
    x = TRUE, y = TRUE)
```

	Obs	Model Likelihood		Discrimination		Rank Discrim.	
		Ratio	Test	R2	Indexes	C	Indexes
0	1033	LR	chi2	159.51	0.214	0.744	
1	247		d.f.	22	g	0.487	
	786		Pr(> chi2)	<0.0001	gr	0.487	
max deriv	5e-12				gp	0.177	
					Brier	0.152	

	Coef	S.E.	Wald Z	Pr(> Z)
Intercept	2.7525	1.9074	1.44	0.1490
bmi	-0.0072	0.0763	-0.09	0.9250
bmi'	-0.2772	0.3288	-0.84	0.3992
bmi''	0.9199	0.9873	0.93	0.3515
physhealth	0.0287	0.0905	0.32	0.7514
physhealth'	-0.1396	0.3555	-0.39	0.6946
female	-0.9550	0.8017	-1.19	0.2336
internet30	0.6099	0.7040	0.87	0.3863
genh_n=2	-0.5059	0.7507	-0.67	0.5004
genh_n=3	-0.8611	0.7053	-1.22	0.2221
genh_n=4	-1.1827	0.7535	-1.57	0.1165
genh_n=5	-2.6842	1.1987	-2.24	0.0251
genh_n=2 * physhealth	-0.0253	0.0864	-0.29	0.7698
genh_n=3 * physhealth	-0.0475	0.0842	-0.56	0.5723
genh_n=4 * physhealth	-0.0471	0.0842	-0.56	0.5758
genh_n=5 * physhealth	0.0497	0.0917	0.54	0.5879
female * bmi	0.0111	0.0268	0.41	0.6803
menthealth	-0.0200	0.0116	-1.72	0.0859
sleephrs	-0.0536	0.0503	-1.07	0.2868
internet30 * genh_n=2	0.4193	0.8091	0.52	0.6043
internet30 * genh_n=3	0.1624	0.7592	0.21	0.8306
internet30 * genh_n=4	-0.1802	0.8176	-0.22	0.8255
internet30 * genh_n=5	-1.6615	0.9820	-1.69	0.0907

All right. We've again used 1033 observations, which is correct (after deleting the three with missing `exerany`). The model shows a Nagelkerke R² value of 0.214, and a C statistic of 0.744 after imputation. Each of these results are a little better than what we saw with `m1_a` but only a little. The likelihood ratio (drop in deviance) test is still highly significant.

14.8.1 Validating Summary Statistics

```
set.seed(432009)
validate(m2_a)
```

	index.orig	training	test	optimism	index.corrected	n
Dxy	0.4870	0.5058	0.4590	0.0468	0.4402	40
R2	0.2145	0.2340	0.1849	0.0491	0.1654	40
Intercept	0.0000	0.0000	0.1729	-0.1729	0.1729	40
Slope	1.0000	1.0000	0.8473	0.1527	0.8473	40
Emax	0.0000	0.0000	0.0692	0.0692	0.0692	40
D	0.1534	0.1695	0.1307	0.0387	0.1147	40
U	-0.0019	-0.0019	0.0041	-0.0060	0.0041	40
Q	0.1554	0.1714	0.1266	0.0448	0.1106	40
B	0.1521	0.1502	0.1567	-0.0065	0.1585	40
g	1.0507	1.1335	0.9626	0.1708	0.8799	40
gp	0.1786	0.1866	0.1642	0.0224	0.1562	40

Again, the model's description of summary statistics is a little optimistic for both the C statistic and the Nagelkerke R². In a new sample of data, model `m2_a` might be better expected to show a C statistic near ...

$$C = 0.5 + \frac{D_{xy}}{2} = 0.5 + \frac{0.4402}{2} = 0.7201$$

rather than the 0.744 we saw initially, and that the Nagelkerke R² in new data will be closer to 0.165, than to the nominal 0.215 we saw above. So, after validation, this model actually looks worse than model m1_a.

Model	nominal C	nominal R ²	validated C	validated R ²
m1_a	0.741	0.204	0.722	0.170
m2_a	0.744	0.214	0.720	0.165

Again, as we walk through other output for model m2_a, remember that the our ability to discriminate exercisers from non-exercisers is still very much in question using either model.

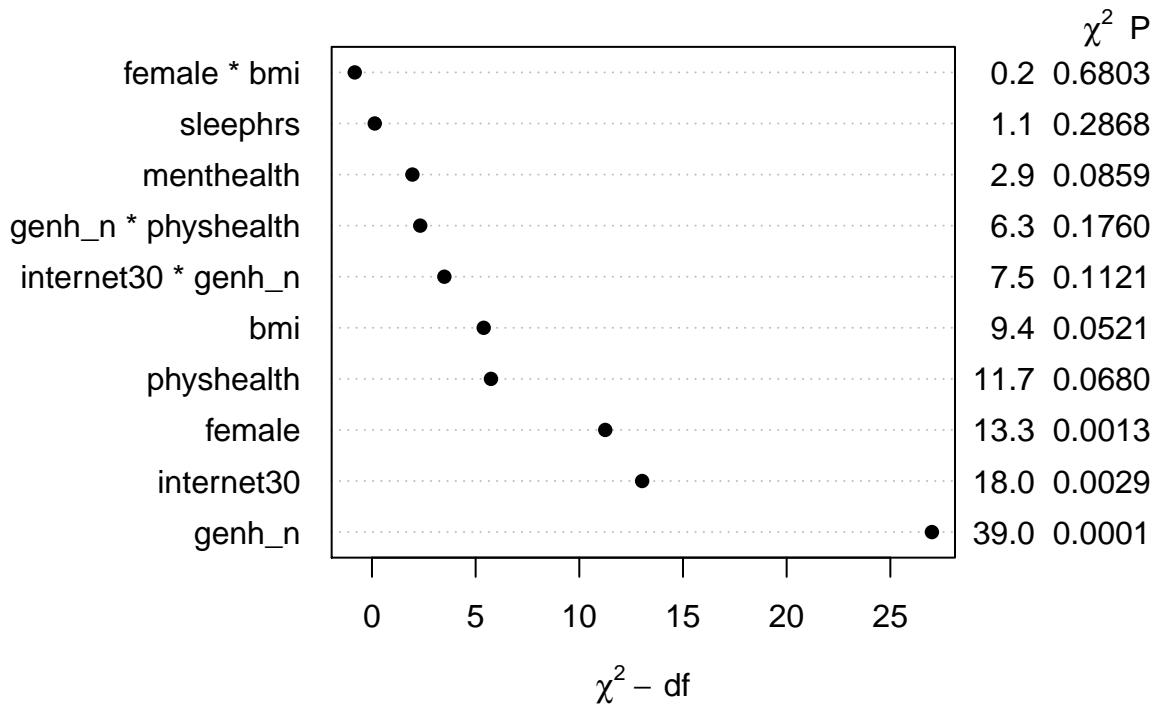
14.8.2 ANOVA for the model

Next, let's look at the ANOVA comparisons for this model.

```
anova(m2_a)
```

	Wald Statistics	Response: exerany		
		Chi-Square	d.f.	P
Factor				
bmi (Factor+Higher Order Factors)		9.39	4	0.0521
All Interactions		0.17	1	0.6803
Nonlinear		1.46	2	0.4831
physhealth (Factor+Higher Order Factors)		11.74	6	0.0680
All Interactions		6.33	4	0.1760
Nonlinear		0.15	1	0.6946
female (Factor+Higher Order Factors)		13.25	2	0.0013
All Interactions		0.17	1	0.6803
internet30 (Factor+Higher Order Factors)		18.03	5	0.0029
All Interactions		7.49	4	0.1121
genh_n (Factor+Higher Order Factors)		39.00	12	0.0001
All Interactions		11.80	8	0.1604
genh_n * physhealth (Factor+Higher Order Factors)		6.33	4	0.1760
female * bmi (Factor+Higher Order Factors)		0.17	1	0.6803
menthealth		2.95	1	0.0859
sleephrs		1.13	1	0.2868
internet30 * genh_n (Factor+Higher Order Factors)		7.49	4	0.1121
TOTAL NONLINEAR		1.65	3	0.6474
TOTAL INTERACTION		12.10	9	0.2075
TOTAL NONLINEAR + INTERACTION		13.40	12	0.3409
TOTAL		131.03	22	<.0001

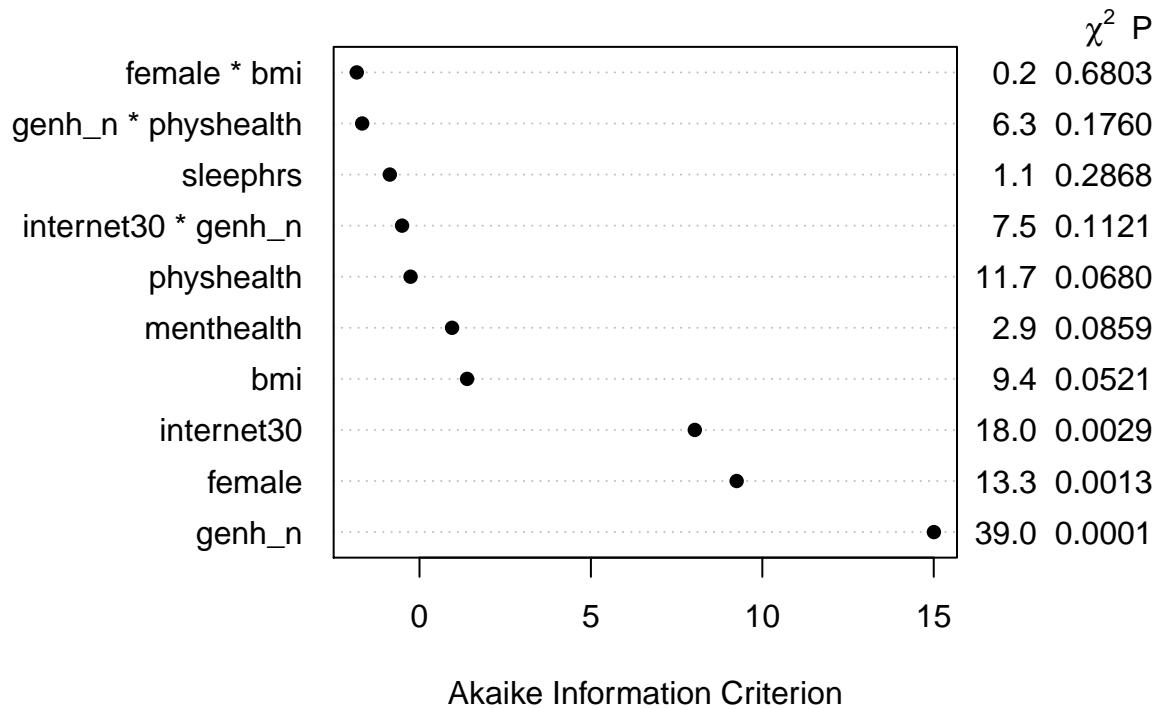
```
plot(anova(m2_a))
```



Here, it looks like just three of the variables (`genhealth`, `internet30`, and `female`) are carrying statistically significant predictive value.

Here is the AIC plot.

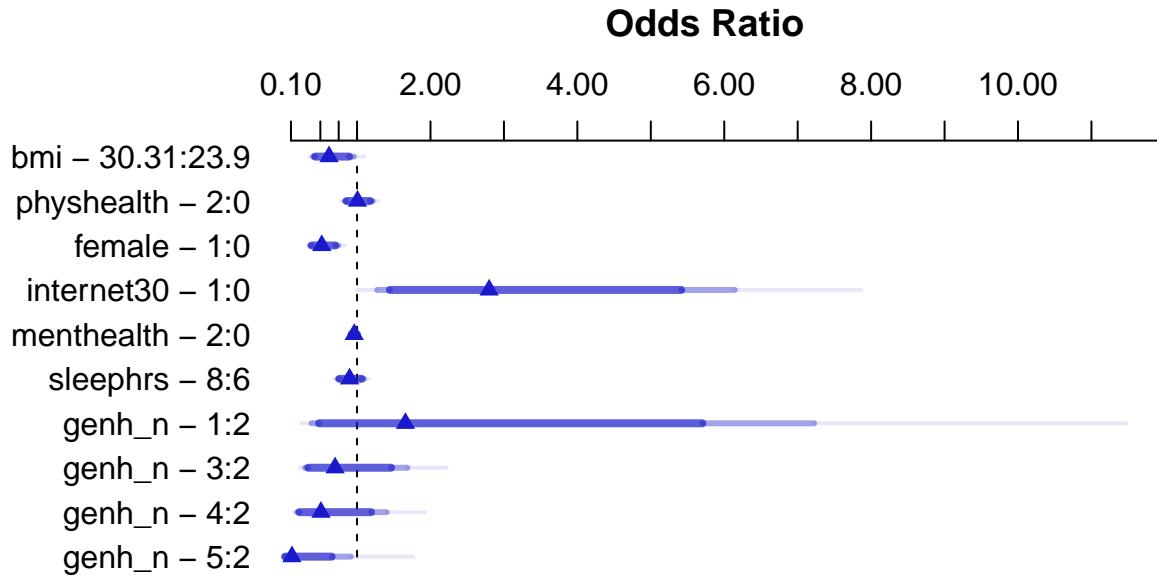
```
plot(anova(m2_a), what="aic")
```



14.8.3 Summarizing Effect Size

How big are the effects we see?

```
plot(summary(m2_a))
```



Adjusted to:bmi=27.03416 physhealth=0 female=0 internet30=0 genh_n=2

```
summary(m2_a)
```

Factor	Effects			Response : exerany					
	Low	High	Diff.	Effect	S.E.	Lower	0.95	Upper	0.95
bmi	23.9	30.31	6.41	-0.4797500	0.221460	-0.913810	-0.045691		
Odds Ratio	23.9	30.31	6.41	0.6189400		NA	0.400990	0.955340	
physhealth	0.0	2.00	2.00	0.0055399	0.095955	-0.182530	0.193610		
Odds Ratio	0.0	2.00	2.00	1.0056000		NA	0.833160	1.213600	
female	0.0	1.00	1.00	-0.6558700	0.183810	-1.016100	-0.295600		
Odds Ratio	0.0	1.00	1.00	0.5189900		NA	0.361990	0.744080	
internet30	0.0	1.00	1.00	1.0291000	0.401140	0.242910	1.815300		
Odds Ratio	0.0	1.00	1.00	2.7986000		NA	1.275000	6.143200	
menthealth	0.0	2.00	2.00	-0.0399460	0.023258	-0.085530	0.005638		
Odds Ratio	0.0	2.00	2.00	0.9608400		NA	0.918030	1.005700	
sleephrs	6.0	8.00	2.00	-0.1072500	0.100690	-0.304600	0.090096		
Odds Ratio	6.0	8.00	2.00	0.8983000		NA	0.737420	1.094300	
genh_n - 1:2	2.0	1.00	NA	0.5059000	0.750690	-0.965430	1.977200		
Odds Ratio	2.0	1.00	NA	1.6585000		NA	0.380820	7.222700	
genh_n - 3:2	2.0	3.00	NA	-0.3552100	0.447130	-1.231600	0.521140		
Odds Ratio	2.0	3.00	NA	0.7010200		NA	0.291830	1.684000	
genh_n - 4:2	2.0	4.00	NA	-0.6768400	0.517450	-1.691000	0.337350		
Odds Ratio	2.0	4.00	NA	0.5082200		NA	0.184330	1.401200	
genh_n - 5:2	2.0	5.00	NA	-2.1783000	1.066000	-4.267700	-0.089045		
Odds Ratio	2.0	5.00	NA	0.1132300		NA	0.014015	0.914800	

```
Adjusted to: bmi=27.03416 physhealth=0 female=0 internet30=0 genh_n=2
```

This output is easier to read as a result of using small *numeric* labels in `genh_n`, rather than the lengthy labels in `genhealth`. The sensible things to interpret are the odds ratios. For example,

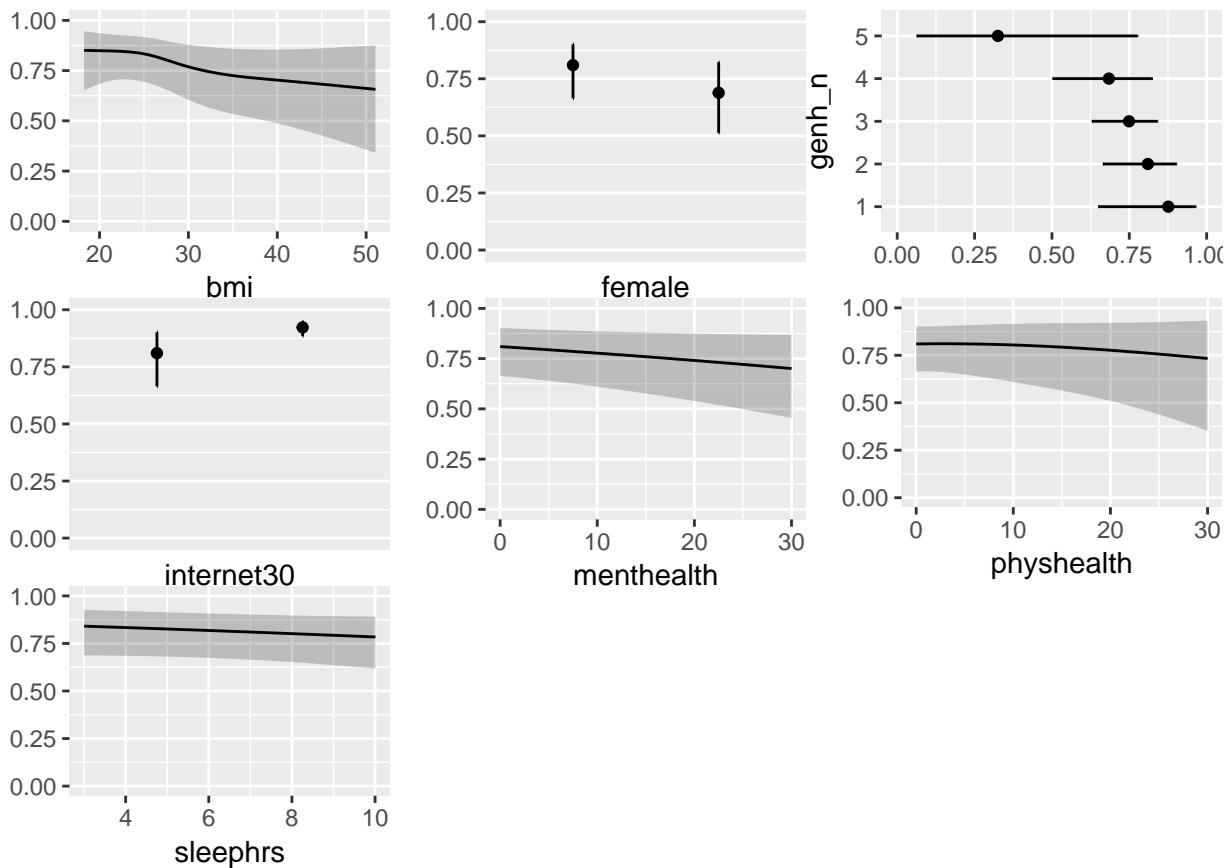
- holding all other predictors constant, the effect of moving from `internet30 = 0` to `internet30 = 1` is that the odds of `exerany` increase by a factor of 2.80.
- the odds ratio comparing two subjects with the same predictors except that Harry has a BMI of 30.31 (the 75th percentile of observed BMIs in our sample) and Marty has a BMI of 23.9 (the 25th percentile) is that Harry has 0.619 times the odds of exercising that Marty does. So Harry's probability of exercise will also be lower.

By sex, which group has a larger probability of `exerany`, holding all other predictors constant, by this model? Females or Males?

14.8.4 Plotting the Model with ggplot and Predict

Again, consider a series of plots describing the model `m2_a` on the probability scale.

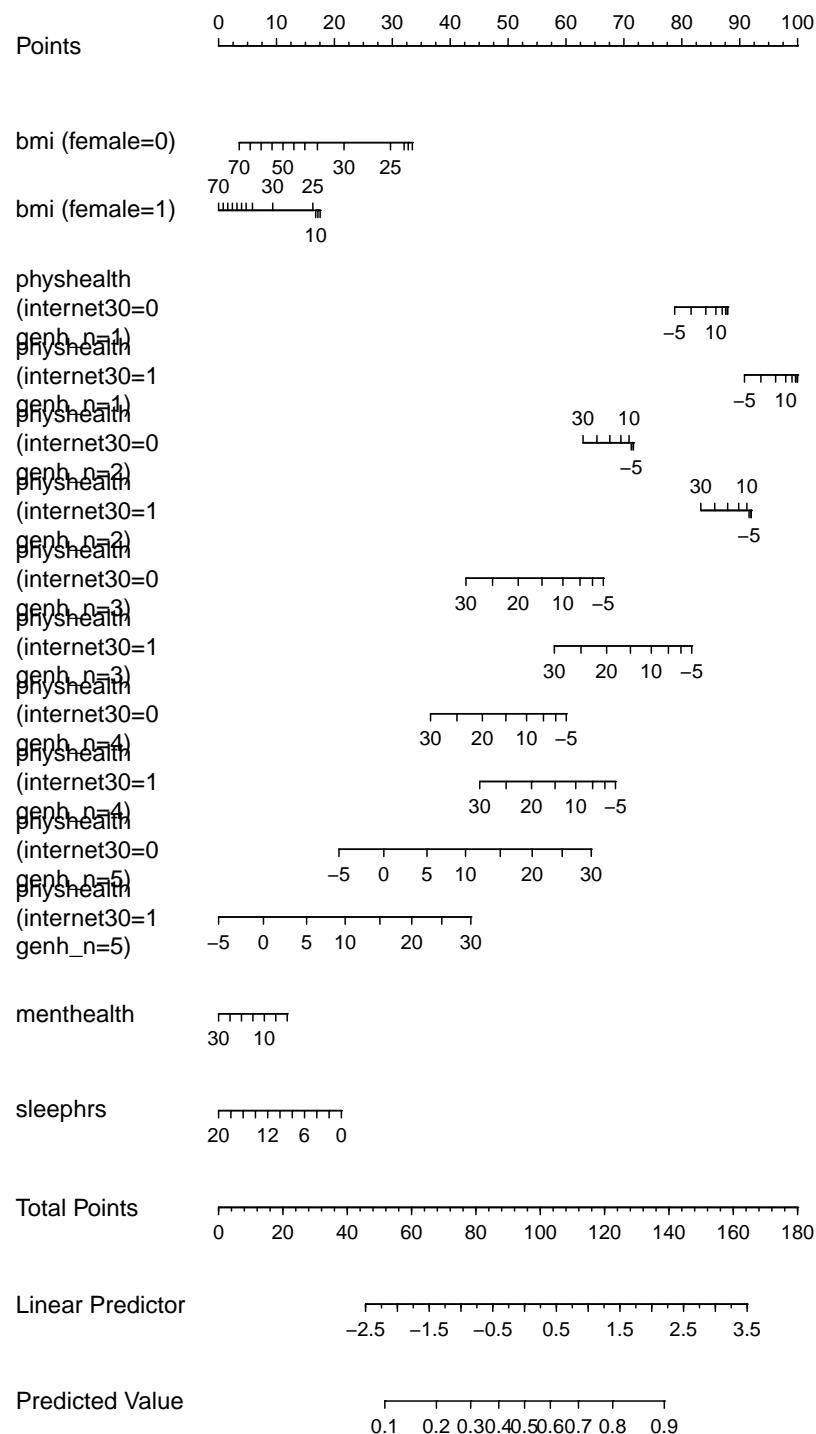
```
ggplot(Predict(m2_a, fun = plogis))
```



Note the small kink in the `bmi` plot. To what do you attribute this?

14.8.5 Plotting the model with a nomogram

```
plot(nomogram(m2_a, fun = plogis))
```



Note the impact of our interaction terms, **and** the cubic splines in `bmi` and `physhealth`. As with any nomogram, to make a prediction we:

1. find the values of each of our predictors in the scales, and travel vertically up to the Points line to read off the Points for that predictor.
2. sum up the Points across all predictors, and find that location in the Total Points line.
3. move vertically down from the total points line to find the estimated “linear predictor” (log odds ratio) and finally the “predicted value” (probability of our outcome `exerany` = 1.)

14.8.6 Checking the Goodness of Fit of our model

To test the goodness of fit, we can use the following omnibus test:

```
round(residuals(m2_a, type = "gof"), 3)
```

Sum of squared errors	Expected value H0	SD
157.102	157.011	0.522
Z	P	
0.174	0.862	

Our non-significant *p* value suggests that we cannot detect anything that’s obviously wrong in the model in terms of goodness of fit.

14.9 Comparing Model 2 to Model 1 after simple imputation

We can refit the models with `glm` and then compare them with `anova`, `aic` and `bic` approaches, if we like.

```
m1_a_glm <- glm(exerany ~ internet30 * factor(genh_n) +
                    bmi * female + physhealth + menthealth +
                    sleephrs,
                    data = smartcle_imp1,
                    family = binomial)

m2_a_glm <- glm(exerany ~ rcs(bmi, 4) + rcs(physhealth, 5) +
                    female + internet30 * factor(genh_n) +
                    factor(genh_n) %ia% physhealth + female %ia% bmi +
                    menthealth + sleephrs,
                    data = smartcle_imp1,
                    family = binomial)
```

14.9.1 Comparison by Analysis of Deviance

```
anova(m1_a_glm, m2_a_glm)
```

Analysis of Deviance Table

Model 1: exerany ~ internet30 * factor(genh_n) + bmi * female + physhealth +	menthealth + sleephrs		
Model 2: exerany ~ rcs(bmi, 4) + rcs(physhealth, 5) + female + internet30 *	factor(genh_n) + factor(genh_n) %ia% physhealth + female %ia%		
bmi + menthealth + sleephrs			
Resid. Df	Resid. Df	Dev Df	Deviance
1	1017		985.32

```
2      1010    976.89  7   8.4245
```

To obtain a p value, we can compare this drop in deviance to a χ^2 distribution with 7 df, as follows:

```
pchisq(8.4245, 7, lower.tail = FALSE)
```

```
[1] 0.2966531
```

So there's no statistically significant advantage apparent from fitting the larger `m2_a` model.

14.9.2 Comparing AIC and BIC

```
glance(m1_a_glm)
```

	null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual
1	1136.406	1032	-492.6596	1017.319	1096.363	985.3192	1017

```
glance(m2_a_glm)
```

	null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual
1	1136.406	1032	-488.4473	1022.895	1136.52	976.8946	1010

Model `m1_a_glm` shows lower AIC and BIC than does `m2_a_glm`, again suggesting no meaningful advantage for the larger model.

14.10 Dealing with Missing Data via Multiple Imputation

Next, we'll use the `aregImpute` function within the `Hmisc` package to predict all missing values for all of our variables, using additive regression bootstrapping and predictive mean matching. The steps for this work are as follows:

1. `aregImpute` draws a sample with replacement from the observations where the target variable is observed, not missing.
2. `aregImpute` then fits a flexible additive model to predict this target variable while finding the optimum transformation of it.
3. `aregImpute` then uses this fitted flexible model to predict the target variable in all of the original observations.
4. Finally, `aregImpute` imputes each missing value of the target variable with the observed value whose predicted transformed value is closest to the predicted transformed value of the missing value.

We'll start with the `smartcle_imp0` data set, which contains only the subjects in the original `smartcle1` data where `exerany` is available, and which includes only the variables of interest to us, including both the factor (`genhealth`) and numeric (`genh_n`) versions of the genhealth data.

```
summary(smartcle_imp0)
```

SEQNO	exerany	physhealth	menthealth
Min. :2.016e+09	Min. :0.0000	Min. : 0.000	Min. : 0.000
1st Qu.:2.016e+09	1st Qu.:1.0000	1st Qu.: 0.000	1st Qu.: 0.000
Median :2.016e+09	Median :1.0000	Median : 0.000	Median : 0.000
Mean :2.016e+09	Mean :0.7609	Mean : 3.949	Mean : 2.701
3rd Qu.:2.016e+09	3rd Qu.:1.0000	3rd Qu.: 2.000	3rd Qu.: 2.000
Max. :2.016e+09	Max. :1.0000	Max. :30.000	Max. :30.000
		NA's :16	NA's :11
genhealth	bmi	female	internet30
1_Excellent:172	Min. :12.71	Min. :0.0000	Min. :0.0000
2_VeryGood :350	1st Qu.:23.70	1st Qu.:0.0000	1st Qu.:1.0000

```

3_Good      :344   Median :26.68   Median :1.0000   Median :1.0000
4_Fair       :121   Mean    :27.85   Mean    :0.6012   Mean    :0.8101
5_Poor       : 43   3rd Qu.:30.53   3rd Qu.:1.0000   3rd Qu.:1.0000
NA's        :  3   Max.   :66.06   Max.   :1.0000   Max.   :1.0000
                           NA's   :83                   NA's   :6

sleephrs          genh_n
Min.   : 1.000   Min.   :1.000
1st Qu.: 6.000   1st Qu.:2.000
Median  : 7.000   Median  :2.000
Mean   : 7.022   Mean   :2.527
3rd Qu.: 8.000   3rd Qu.:3.000
Max.   :20.000   Max.   :5.000
NA's   : 7       NA's   :3

```

The `smartcle_imp0` data set contains 1033 rows (subjects) and 10 columns (variables.)

14.10.1 Using `aregImpute` to fit a multiple imputation model

To set up `aregImpute` here, we'll need to specify:

- a suitable random seed with `set.seed` so we can replicate our work later
- a data set via the `datadist` stuff shown below
- the variables we want to include in the imputation process, which should include, at a minimum, any variables with missing values, and any variables we want to include in our outcome models
- `n.impute` = number of imputations, we'll run 20 here¹
- `nk` = number of knots to describe level of complexity, with our choice `nk = c(0, 3)` we'll fit both linear models and models with restricted cubic splines with 3 knots (this approach will wind up throwing some warnings here because some of our variables with missing values have only a few options so fitting splines is tough.)
- `tlinear = FALSE` allows the target variable for imputation to have a non-linear transformation when `nk` is 3 or more. Here, I'll use `tlinear = TRUE`, the default.
- `B = 10` specifies 10 bootstrap samples will be used
- `pr = FALSE` tells the machine not to print out which iteration is running as it goes.
- `data` specifies the source of the variables

```

set.seed(432074)
dd <- datadist(smartcle_imp0)
options(datadist = "dd")

imp_fit <- aregImpute(~ exerany + physhealth + menthealth +
                      genh_n + bmi + female +
                      internet30 + sleephrs,
                      nk = c(0, 3), tlinear = TRUE,
                      data = smartcle_imp0, B = 10,
                      n.impute = 20, pr = FALSE)

```

OK. Here is the imputation model. The summary here isn't especially critical. We want to see what was run, but to see what the results look like, we'll need a plot, to come.

```
imp_fit
```

Multiple Imputation using Bootstrap and PMM

¹100 is generally safe but time-consuming. In the old days, we used to say 5. A reasonable idea is to identify the fraction of missingness in your variable with the most missingness, and if that's 0.10, then you should run at least $100(0.10) = 10$ sets of imputations.

```

aregImpute(formula = ~exerany + physhealth + menthealth + genh_n +
  bmi + female + internet30 + sleephrs, data = smartcle_imp0,
  n.impute = 20, nk = c(0, 3), tlinear = TRUE, pr = FALSE,
  B = 10)

n: 1033      p: 8      Imputations: 20      nk: 0

Number of NAs:
  exerany physhealth menthealth      genh_n      bmi      female
    0         16        11          3        83          0
internet30     sleephrs
    6           7

      type d.f.
exerany      1   1
physhealth    s   1
menthealth    s   1
genh_n        s   1
bmi           s   1
female        1   1
internet30    1   1
sleephrs      s   1

```

Transformation of Target Variables Forced to be Linear

R-squares for Predicting Non-Missing Values for Each Variable
 Using Last Imputations of Predictors

physhealth	menthealth	genh_n	bmi	internet30	sleephrs
0.336	0.107	0.304	0.127	0.111	0.056

Resampling results for determining the complexity of imputation models

Variable being imputed: physhealth

	nk=0	nk=3
Bootstrap bias-corrected R ²	0.289	0.399
10-fold cross-validated R ²	0.288	0.381
Bootstrap bias-corrected mean error	4.911	4.014
10-fold cross-validated mean error	4.253	4.091
Bootstrap bias-corrected median error	3.261	1.784
10-fold cross-validated median error	0.896	0.666

Variable being imputed: menthealth

	nk=0	nk=3
Bootstrap bias-corrected R ²	0.103	0.116
10-fold cross-validated R ²	0.110	0.128
Bootstrap bias-corrected mean error	3.660	3.651
10-fold cross-validated mean error	3.018	3.075
Bootstrap bias-corrected median error	1.957	1.780
10-fold cross-validated median error	0.715	0.786

Variable being imputed: genh_n

	nk=0	nk=3
Bootstrap bias-corrected R ²	0.323	0.319

```

10-fold cross-validated R^2           0.324 0.323
Bootstrap bias-corrected mean |error| 0.690 0.691
10-fold cross-validated mean |error| 2.537 2.538
Bootstrap bias-corrected median |error| 0.644 0.648
10-fold cross-validated median |error| 2.555 2.571

Variable being imputed: bmi
                                nk=0    nk=3
Bootstrap bias-corrected R^2       0.0715  0.0919
10-fold cross-validated R^2       0.0825  0.0892
Bootstrap bias-corrected mean |error| 4.5861  4.5264
10-fold cross-validated mean |error| 27.8377 27.8712
Bootstrap bias-corrected median |error| 3.5091  3.5217
10-fold cross-validated median |error| 26.8401 26.7857

Variable being imputed: internet30
                                nk=0    nk=3
Bootstrap bias-corrected R^2       0.0843  0.0760
10-fold cross-validated R^2       0.1016  0.0825
Bootstrap bias-corrected mean |error| 0.2798  0.2857
10-fold cross-validated mean |error| 0.9636  0.9594
Bootstrap bias-corrected median |error| 0.1883  0.1819
10-fold cross-validated median |error| 0.7757  0.7467

Variable being imputed: sleephrs
                                nk=0    nk=3
Bootstrap bias-corrected R^2       0.0075 -0.0072
10-fold cross-validated R^2       0.0330  0.0149
Bootstrap bias-corrected mean |error| 1.0102  1.0506
10-fold cross-validated mean |error| 7.0313  7.0163
Bootstrap bias-corrected median |error| 0.9287  0.9000
10-fold cross-validated median |error| 7.0066  6.9739

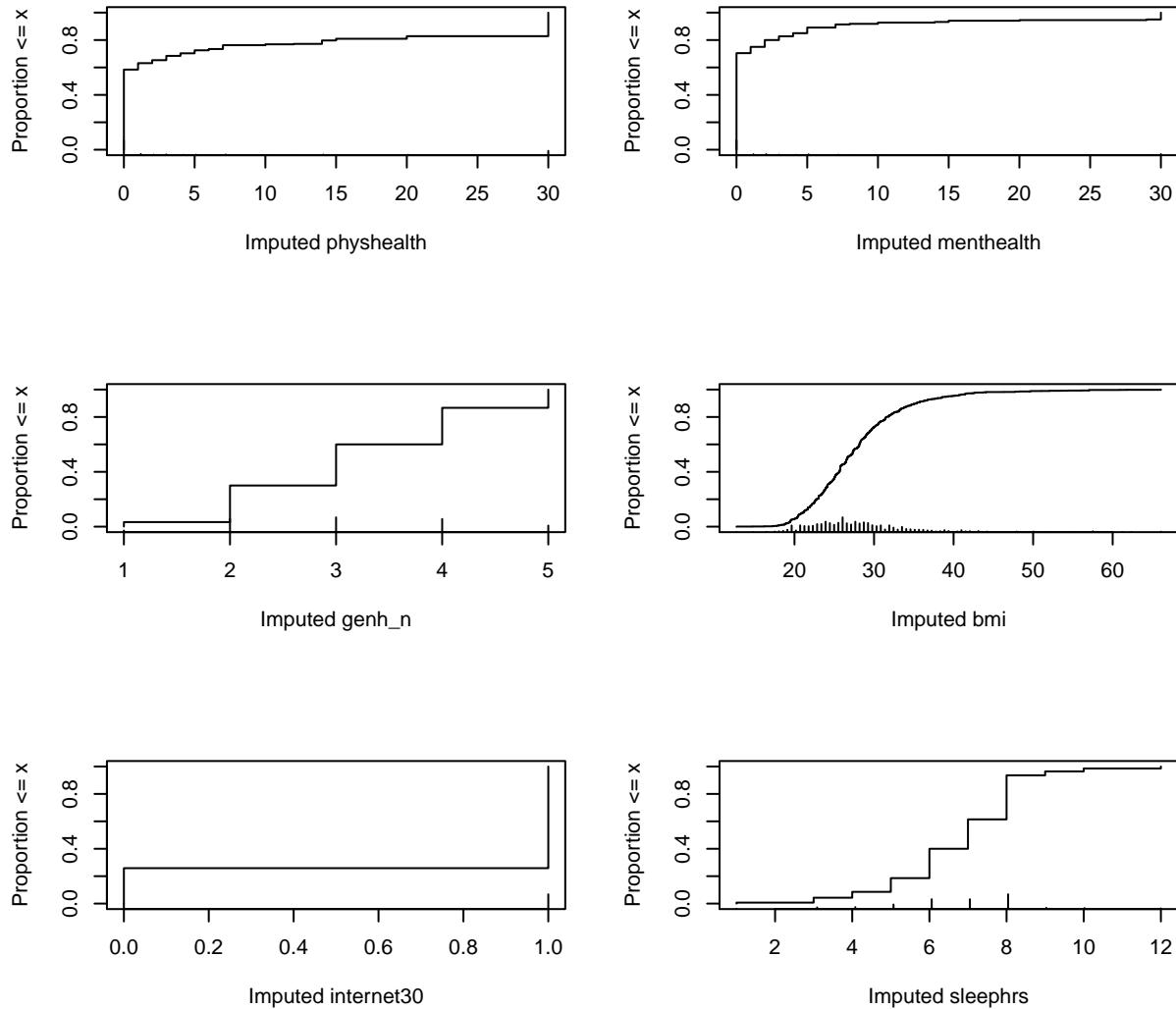
```

OK, let's plot these imputed values. Note that we had six predictors with missing values in our data set, and so if we plot each of those, we'll wind up with six plots. I'll arrange them in a grid with three rows and two columns.

```

par(mfrow = c(3,2))
plot(imp_fit)

```



```
par(mfrow = c(1,1))
```

From these cumulative distribution functions, we can see that, for example,

- we imputed `bmi` values mostly between 20 and 35, with a few values below 20 or above 40.
- most of our imputed `sleephrs` were between 5 and 10
- we imputed 1 for `internet30` for about 70% of the subjects, and 0 for the other 30%.

This predictive mean matching method never imputes a value for a variable that does not already exist in the data.

14.11 Combining the Imputation and Outcome Models

So, now we have an imputation model, called `imp_fit`. and two outcome models: `m1` and `m2`. What do we do with them?

14.11.1 Model 1 with Multiple Imputation

To build the `m1_imp` multiple imputation fit for model `m1`, we use the `fit.mult.impute` command, and specify the model, the fitter (here, `lrm`), the imputation model (`xtrans = imp_fit`) and the data set prior to imputation (`smartcle_imp0`).

```
m1_imp <- fit.mult.impute(exerany ~
  internet30 * catg(genh_n) + bmi * female +
  physhealth + menthealth + sleephrs,
  fitter = lrm, xtrans = imp_fit,
  data = smartcle_imp0, x = TRUE, y = TRUE)
```

Variance Inflation Factors Due to Imputation:

Intercept	internet30	genh_n=2
1.03	1.00	1.01
genh_n=3	genh_n=4	genh_n=5
1.00	1.01	1.02
bmi	female	physhealth
1.06	1.08	1.05
menthealth	sleephrs	internet30 * genh_n=2
1.01	1.02	1.01
internet30 * genh_n=3	internet30 * genh_n=4	internet30 * genh_n=5
1.00	1.01	1.03
bmi * female		
1.08		

Rate of Missing Information:

Intercept	internet30	genh_n=2
0.03	0.00	0.01
genh_n=3	genh_n=4	genh_n=5
0.00	0.01	0.02
bmi	female	physhealth
0.05	0.07	0.04
menthealth	sleephrs	internet30 * genh_n=2
0.01	0.02	0.01
internet30 * genh_n=3	internet30 * genh_n=4	internet30 * genh_n=5
0.00	0.01	0.03
bmi * female		
0.08		

d.f. for t-distribution for Tests of Single Coefficients:

Intercept	internet30	genh_n=2
21695.29	4552554.82	342982.63
genh_n=3	genh_n=4	genh_n=5
2393040.31	340219.46	43771.65
bmi	female	physhealth
6985.80	3594.27	9984.27
menthealth	sleephrs	internet30 * genh_n=2
87510.83	39656.28	292980.96
internet30 * genh_n=3	internet30 * genh_n=4	internet30 * genh_n=5
6057039.66	610087.87	23370.34

```
bmi * female
3152.92
```

The following fit components were averaged over the 20 model fits:

```
stats linear.predictors
```

OK. Let's get the familiar description of an `lrm` model, after this multiple imputation.

```
m1_imp
```

Logistic Regression Model

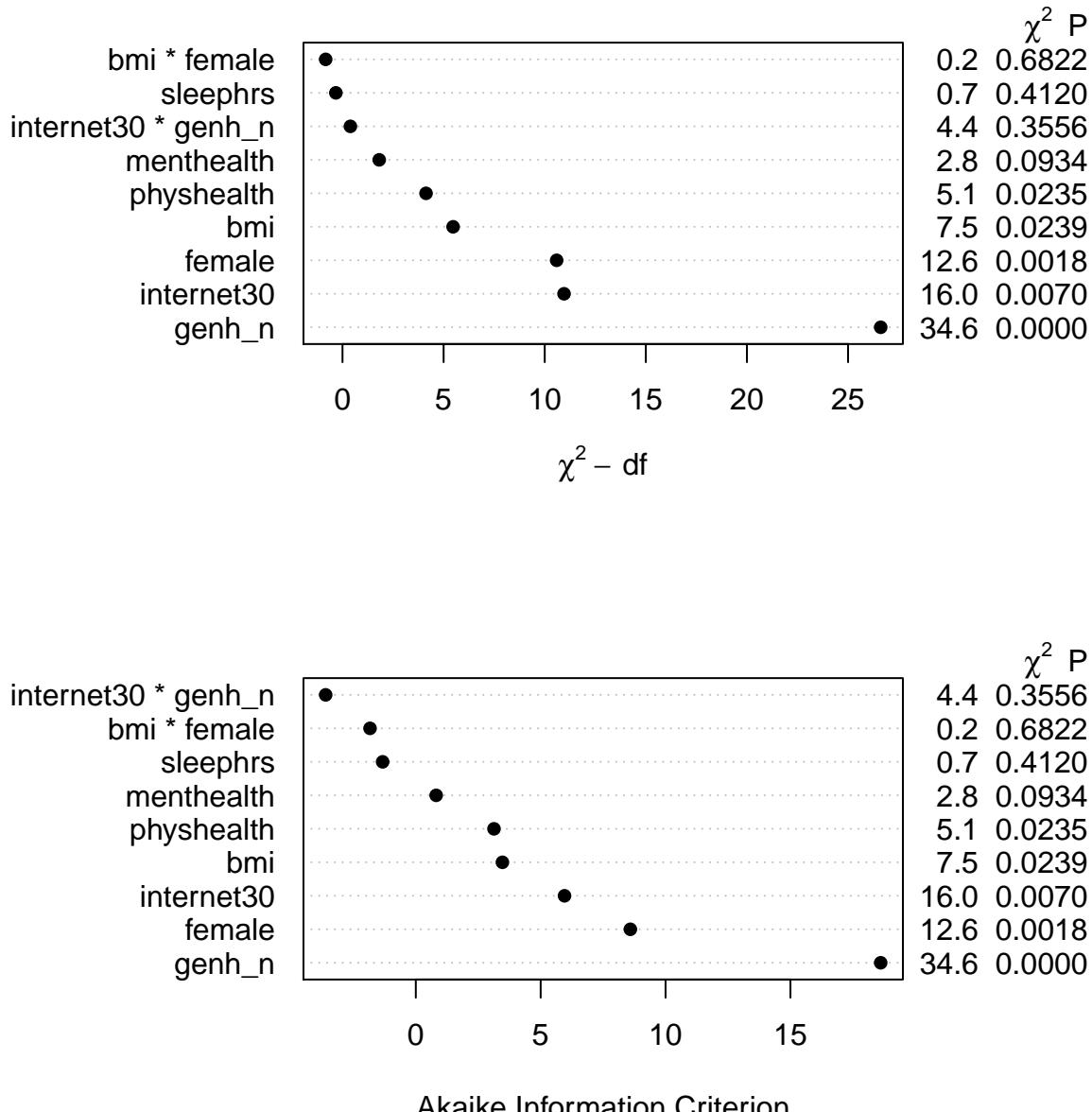
```
fit.mult.impute(formula = exerany ~ internet30 * catg(genh_n) +
  bmi * female + physhealth + menthealth + sleephrs, fitter = lrm,
  xtrans = imp_fit, data = smartcle_imp0, x = TRUE, y = TRUE)

      Model Likelihood          Discrimination       Rank Discrim.
      Ratio Test            Indexes           Indexes
Obs      1033    LR chi2     153.52      R2      0.207      C      0.743
0        247      d.f.         15      g      1.031      Dxy     0.487
1        786      Pr(> chi2) <0.0001      gr      2.803      gamma   0.487
max |deriv| 1e-12                               gp      0.176      tau-a   0.177
                                         Brier     0.153

      Coef    S.E.   Wald Z Pr(>|Z|)
Intercept      3.3183  0.9782   3.39  0.0007
internet30     0.6499  0.7013   0.93  0.3540
genh_n=2      -0.4613  0.7507  -0.61  0.5389
genh_n=3      -0.9069  0.6978  -1.30  0.1937
genh_n=4      -1.3287  0.7392  -1.80  0.0722
genh_n=5      -0.9257  0.8342  -1.11  0.2671
bmi        -0.0408  0.0216  -1.89  0.0588
female      -0.9153  0.7786  -1.18  0.2398
physhealth   -0.0234  0.0103  -2.27  0.0235
menthealth   -0.0192  0.0115  -1.68  0.0934
sleephrs     -0.0407  0.0496  -0.82  0.4120
internet30 * genh_n=2  0.3503  0.8117   0.43  0.6661
internet30 * genh_n=3  0.1110  0.7549   0.15  0.8832
internet30 * genh_n=4 -0.1851  0.8117  -0.23  0.8196
internet30 * genh_n=5 -1.2268  0.9748  -1.26  0.2082
bmi * female    0.0108  0.0263   0.41  0.6822
```

We can obtain an ANOVA plot and an AIC plot to look at the predictors:

```
par(mfrow = c(2,1))
plot(anova(m1_imp))
plot(anova(m1_imp), what="aic")
```



```
par(mfrow = c(1,1))
```

Here's the summary of effect sizes.

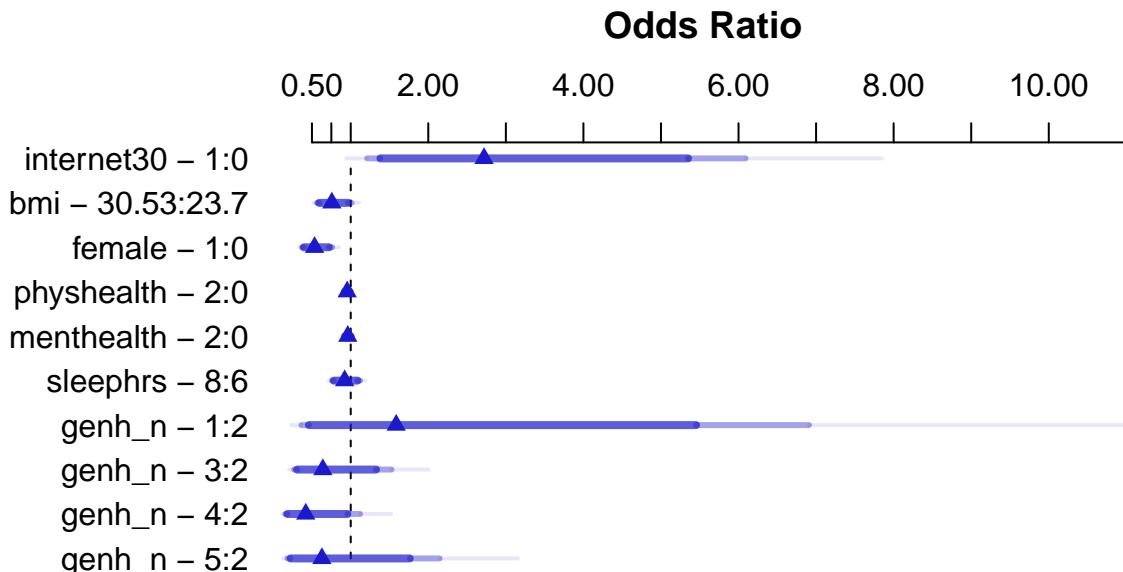
```
summary(m1_imp)
```

Effects	Response : exerany								
	Factor	Low	High	Diff.	Effect	S.E.	Lower	0.95	Upper
internet30	0.0	1.00	1.00	1.000300	0.411430	0.193880	1.8066000		
Odds Ratio	0.0	1.00	1.00	2.719000		NA	1.214000	6.0899000	
bmi	23.7	30.53	6.83	-0.279000	0.147670	-0.568430	0.0104310		

Odds Ratio	23.7	30.53	6.83	0.756540	NA	0.566410	1.0105000
female	0.0	1.00	1.00	-0.628260	0.181140	-0.983290	-0.2732400
Odds Ratio	0.0	1.00	1.00	0.533520	NA	0.374080	0.7609100
physhealth	0.0	2.00	2.00	-0.046783	0.020654	-0.087264	-0.0063023
Odds Ratio	0.0	2.00	2.00	0.954290	NA	0.916440	0.9937200
menthealth	0.0	2.00	2.00	-0.038477	0.022932	-0.083422	0.0064689
Odds Ratio	0.0	2.00	2.00	0.962250	NA	0.919960	1.0065000
sleephrs	6.0	8.00	2.00	-0.081454	0.099287	-0.276050	0.1131500
Odds Ratio	6.0	8.00	2.00	0.921770	NA	0.758770	1.1198000
genh_n - 1:2	2.0	1.00	NA	0.461310	0.750680	-1.010000	1.9326000
Odds Ratio	2.0	1.00	NA	1.586200	NA	0.364220	6.9076000
genh_n - 3:2	2.0	3.00	NA	-0.445620	0.442710	-1.313300	0.4220800
Odds Ratio	2.0	3.00	NA	0.640430	NA	0.268930	1.5251000
genh_n - 4:2	2.0	4.00	NA	-0.867380	0.500240	-1.847800	0.1130700
Odds Ratio	2.0	4.00	NA	0.420050	NA	0.157580	1.1197000
genh_n - 5:2	2.0	5.00	NA	-0.464420	0.626480	-1.692300	0.7634700
Odds Ratio	2.0	5.00	NA	0.628500	NA	0.184090	2.1457000

Adjusted to: internet30=0 genh_n=2 bmi=26.68 female=0

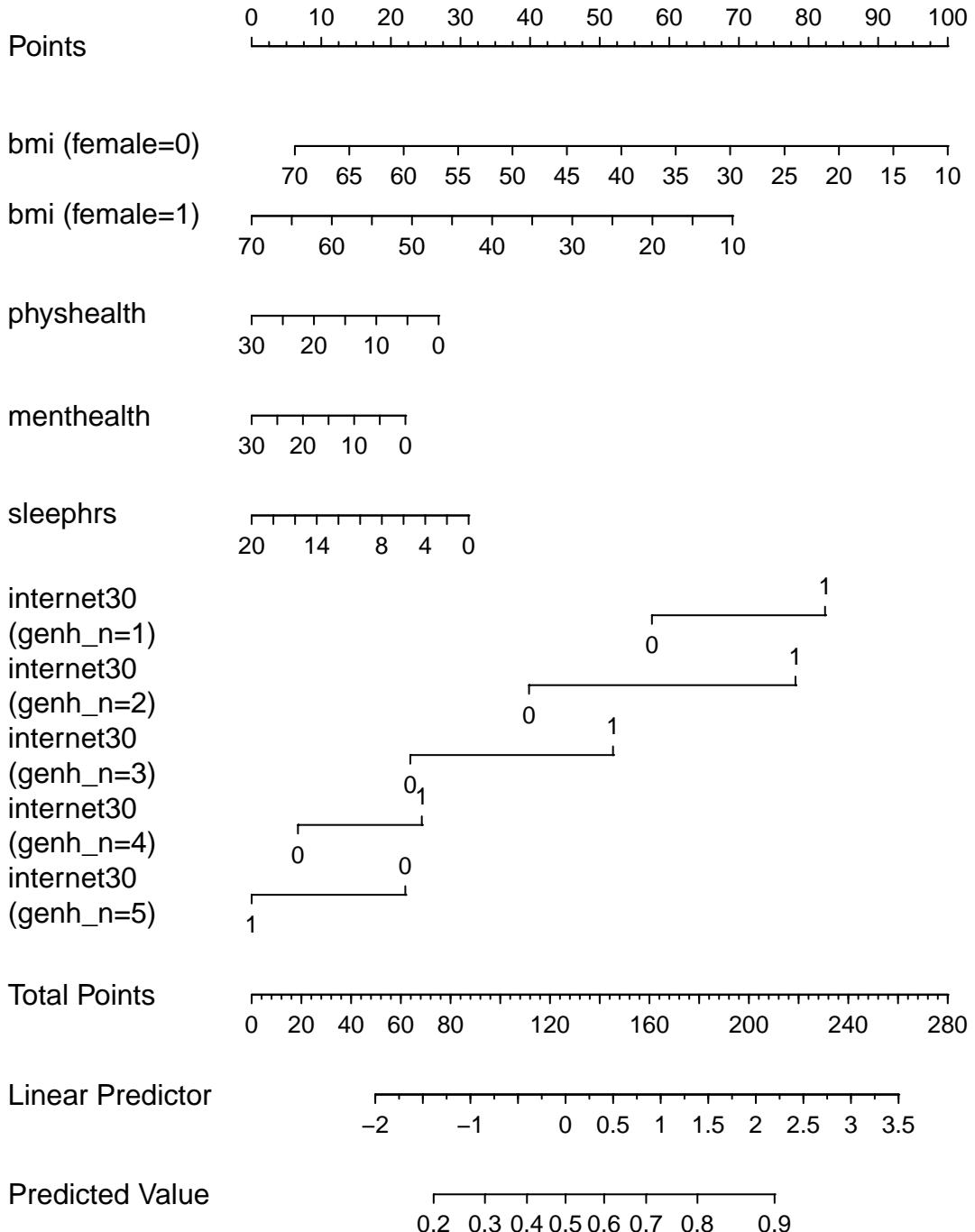
```
plot(summary(m1_imp))
```



Adjusted to: internet30=0 genh_n=2 bmi=26.68 female=0

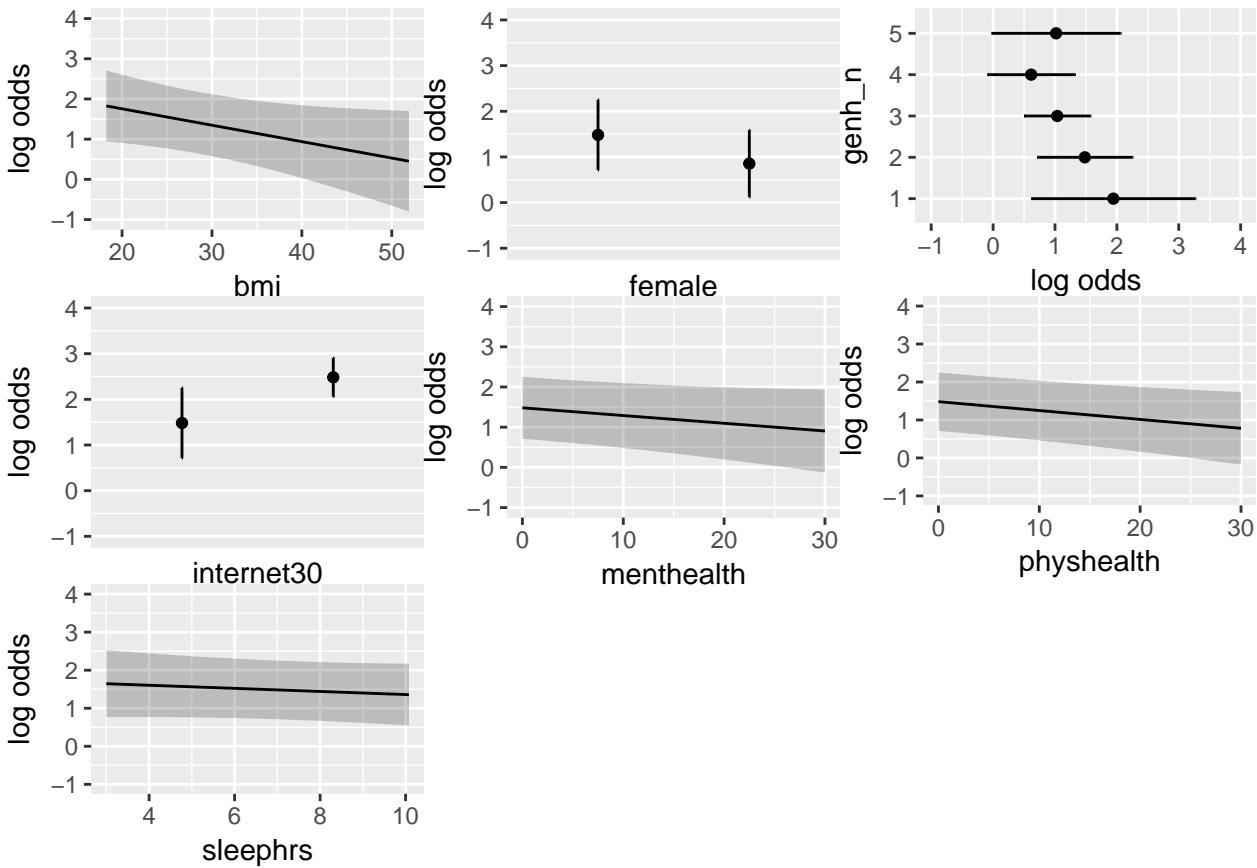
And here is the nomogram.

```
plot(nomogram(m1_imp, fun = plogis))
```



Here are the descriptive model plots, on the original probability scale for our `exerany` outcome:

```
ggplot(Predict(m1_imp), fun = plogis)
```



We can still do things like validate the summary statistics, too.

```
validate(m1_imp)
```

	index.orig	training	test	optimism	index.corrected	n
Dxy	0.4889	0.5155	0.4739	0.0416	0.4473	40
R2	0.2070	0.2299	0.1912	0.0388	0.1682	40
Intercept	0.0000	0.0000	0.1047	-0.1047	0.1047	40
Slope	1.0000	1.0000	0.8922	0.1078	0.8922	40
Emax	0.0000	0.0000	0.0442	0.0442	0.0442	40
D	0.1476	0.1659	0.1355	0.0304	0.1173	40
U	-0.0019	-0.0019	0.0022	-0.0041	0.0022	40
Q	0.1496	0.1678	0.1333	0.0345	0.1151	40
B	0.1533	0.1499	0.1563	-0.0064	0.1597	40
g	1.0307	1.1056	0.9773	0.1283	0.9024	40
gp	0.1764	0.1851	0.1686	0.0165	0.1599	40

14.11.2 Model 2 with Multiple Imputation

The same approach is used to build the `m2_imp` multiple imputation fit for model `m2`, using the `fit.mult.impute` command, and specifying the model, the fitter (here, `lrm`), the imputation model (`xtrans = imp_fit`) and the data set prior to imputation (`smartcle_imp0`).

```
m2_imp <- fit.mult.impute(exerany ~
  rcs(bmi, 4) + rcs(physhealth, 5) +
  female + internet30 * catg(genh_n) +
  catg(genh_n) %ia% physhealth + female %ia% bmi +
  menthealth + sleephrs,
  fitter = lrm, xtrans = imp_fit,
  data = smartcle_imp0, x = TRUE, y = TRUE)
```

Variance Inflation Factors Due to Imputation:

Intercept	bmi	bmi'
1.07	1.08	1.06
bmi''	physhealth	physhealth'
1.06	1.04	1.02
female	internet30	genh_n=2
1.08	1.00	1.01
genh_n=3	genh_n=4	genh_n=5
1.00	1.01	1.01
genh_n=2 * physhealth	genh_n=3 * physhealth	genh_n=4 * physhealth
1.03	1.03	1.02
genh_n=5 * physhealth	female * bmi	menthealth
1.03	1.09	1.01
sleephrs	internet30 * genh_n=2	internet30 * genh_n=3
1.03	1.01	1.00
internet30 * genh_n=4	internet30 * genh_n=5	
1.01	1.04	

Rate of Missing Information:

Intercept	bmi	bmi'
0.07	0.07	0.06
bmi''	physhealth	physhealth'
0.06	0.04	0.02
female	internet30	genh_n=2
0.08	0.00	0.01
genh_n=3	genh_n=4	genh_n=5
0.00	0.01	0.01
genh_n=2 * physhealth	genh_n=3 * physhealth	genh_n=4 * physhealth
0.03	0.03	0.02
genh_n=5 * physhealth	female * bmi	menthealth
0.03	0.08	0.01
sleephrs	internet30 * genh_n=2	internet30 * genh_n=3
0.03	0.01	0.00
internet30 * genh_n=4	internet30 * genh_n=5	
0.01	0.04	

d.f. for t-distribution for Tests of Single Coefficients:

Intercept	bmi	bmi'
4380.39	3700.10	5557.48
bmi''	physhealth	physhealth'
5611.05	14728.96	42868.19
female	internet30	genh_n=2

3252.44	3009564.42	305690.58
genh_n=3	genh_n=4	genh_n=5
1329373.30	318205.87	190432.29
genh_n=2 * physhealth	genh_n=3 * physhealth	genh_n=4 * physhealth
21003.64	22615.76	38202.98
genh_n=5 * physhealth	female * bmi	menthealth
21998.83	3002.51	111867.55
sleephrs internet30 * genh_n=2	internet30 * genh_n=3	
26186.82	211617.58	2865315.54
internet30 * genh_n=4	internet30 * genh_n=5	
551890.44	10977.39	

The following fit components were averaged over the 20 model fits:

```
stats linear.predictors
```

OK. Let's get the familiar description of an `lrm` model, after this multiple imputation.

```
m2_imp
```

Logistic Regression Model

```
fit.mult.impute(formula = exerany ~ rcs(bmi, 4) + rcs(physhealth,
5) + female + internet30 * catg(genh_n) + catg(genh_n) %ia%
physhealth + female %ia% bmi + menthealth + sleephrs, fitter = lrm,
xtrans = imp_fit, data = smartcle_imp0, x = TRUE, y = TRUE)
```

	Obs	Model Likelihood		Discrimination		Rank Discrim.	
		Ratio	Test	R2	Indexes	C	Indexes
0	1033	LR	chi2	162.26	0.218	0.746	
1	247	d.f.		22	1.062	0.492	
max deriv	786	Pr(> chi2)	<0.0001		2.892	0.492	
				gp	0.180	tau-a	0.179
				Brier	0.152		

	Coef	S.E.	Wald Z	Pr(> Z)
Intercept	2.9915	1.9954	1.50	0.1338
bmi	-0.0164	0.0812	-0.20	0.8397
bmi'	-0.2350	0.3356	-0.70	0.4837
bmi''	0.7236	0.9089	0.80	0.4260
physhealth	0.0261	0.0862	0.30	0.7624
physhealth'	-0.2434	0.3696	-0.66	0.5102
female	-0.9680	0.8140	-1.19	0.2344
internet30	0.6182	0.7059	0.88	0.3812
genh_n=2	-0.4851	0.7558	-0.64	0.5210
genh_n=3	-0.8620	0.7066	-1.22	0.2225
genh_n=4	-1.2144	0.7554	-1.61	0.1079
genh_n=5	-2.7928	1.2040	-2.32	0.0204
genh_n=2 * physhealth	-0.0061	0.0803	-0.08	0.9397
genh_n=3 * physhealth	-0.0325	0.0775	-0.42	0.6750
genh_n=4 * physhealth	-0.0322	0.0772	-0.42	0.6766
genh_n=5 * physhealth	0.0634	0.0856	0.74	0.4591
female * bmi	0.0111	0.0272	0.41	0.6831
menthealth	-0.0197	0.0117	-1.68	0.0923
sleephrs	-0.0554	0.0510	-1.09	0.2774

```

internet30 * genh_n=2  0.3882  0.8163  0.48  0.6344
internet30 * genh_n=3  0.1368  0.7617  0.18  0.8575
internet30 * genh_n=4 -0.1712  0.8203 -0.21  0.8347
internet30 * genh_n=5 -1.4939  0.9985 -1.50  0.1346

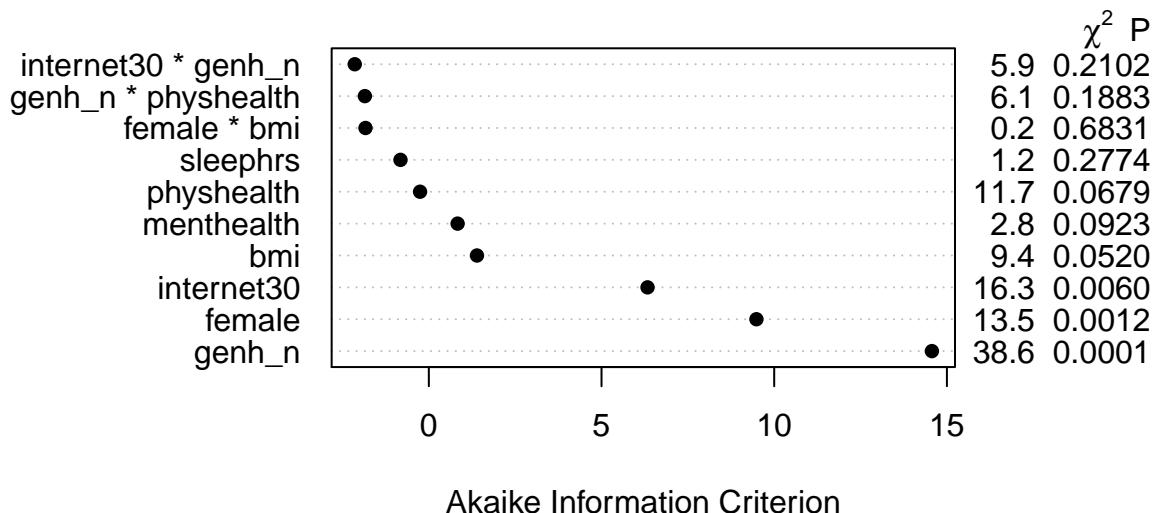
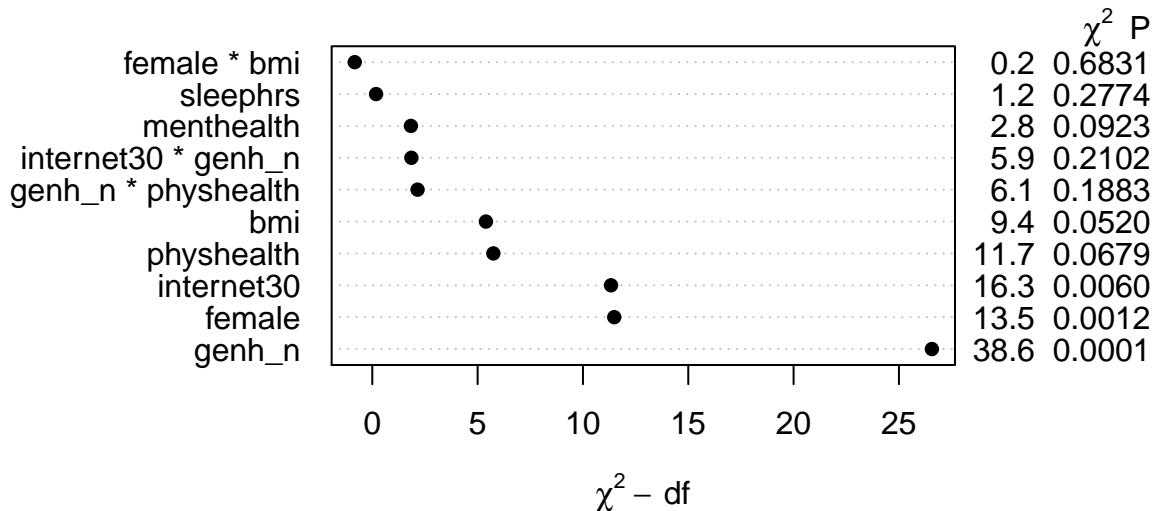
```

We can obtain an ANOVA plot and an AIC plot to look at the predictors:

```

par(mfrow = c(2,1))
plot(anova(m2_imp))
plot(anova(m2_imp), what="aic")

```



```
par(mfrow = c(1,1))
```

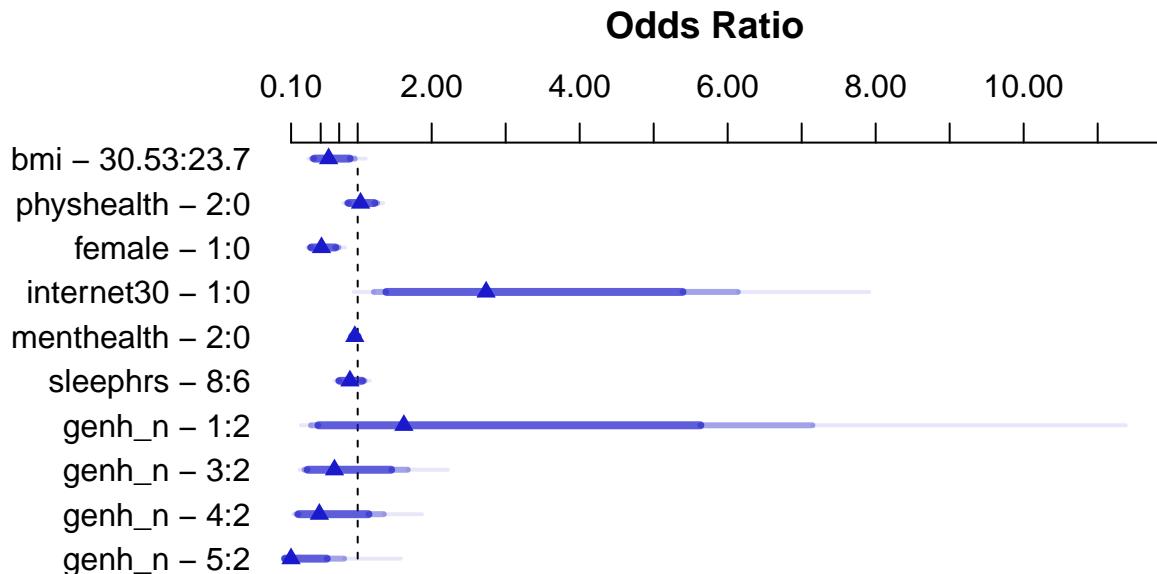
Here's the summary of effect sizes.

```
summary(m2_imp)
```

	Effects							Response : exerany		
Factor	Low	High	Diff.	Effect	S.E.	Lower	0.95	Upper	0.95	
bmi	23.7	30.53	6.83	-0.498810	0.233540	-0.956540		-0.0410830		
Odds Ratio	23.7	30.53	6.83	0.607250		NA	0.384220	0.9597500		
physhealth	0.0	2.00	2.00	0.037808	0.100250	-0.158680	0.2343000			
Odds Ratio	0.0	2.00	2.00	1.038500		NA	0.853270	1.2640000		
female	0.0	1.00	1.00	-0.671780	0.189360	-1.042900	-0.3006400			
Odds Ratio	0.0	1.00	1.00	0.510800		NA	0.352420	0.7403400		
internet30	0.0	1.00	1.00	1.006400	0.412260	0.198370	1.8144000			
Odds Ratio	0.0	1.00	1.00	2.735700		NA	1.219400	6.1374000		
menthealth	0.0	2.00	2.00	-0.039395	0.023405	-0.085267	0.0064773			
Odds Ratio	0.0	2.00	2.00	0.961370		NA	0.918270	1.0065000		
sleephrs	6.0	8.00	2.00	-0.110820	0.102030	-0.310800	0.0891530			
Odds Ratio	6.0	8.00	2.00	0.895100		NA	0.732860	1.0932000		
genh_n - 1:2	2.0	1.00	NA	0.485110	0.755780	-0.996190	1.9664000			
Odds Ratio	2.0	1.00	NA	1.624400		NA	0.369290	7.1450000		
genh_n - 3:2	2.0	3.00	NA	-0.376840	0.455990	-1.270600	0.5168800			
Odds Ratio	2.0	3.00	NA	0.686030		NA	0.280670	1.6768000		
genh_n - 4:2	2.0	4.00	NA	-0.729310	0.526230	-1.760700	0.3020700			
Odds Ratio	2.0	4.00	NA	0.482240		NA	0.171920	1.3527000		
genh_n - 5:2	2.0	5.00	NA	-2.307700	1.074000	-4.412600	-0.2026900			
Odds Ratio	2.0	5.00	NA	0.099493		NA	0.012123	0.8165300		

Adjusted to: bmi=26.68 physhealth=0 female=0 internet30=0 genh_n=2

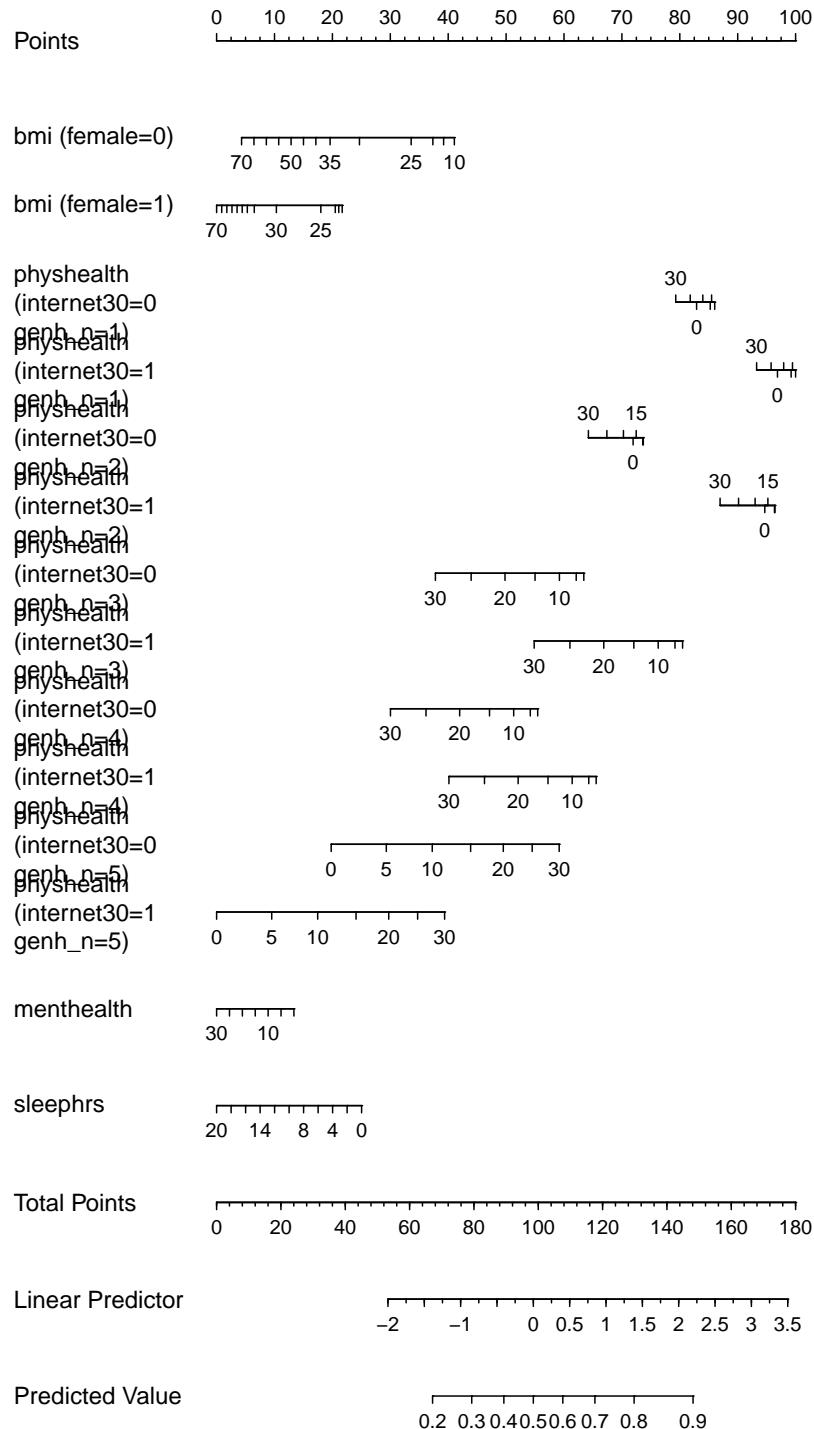
```
plot(summary(m2_imp))
```



Adjusted to:bmi=26.68 physhealth=0 female=0 internet30=0 genh_n=2

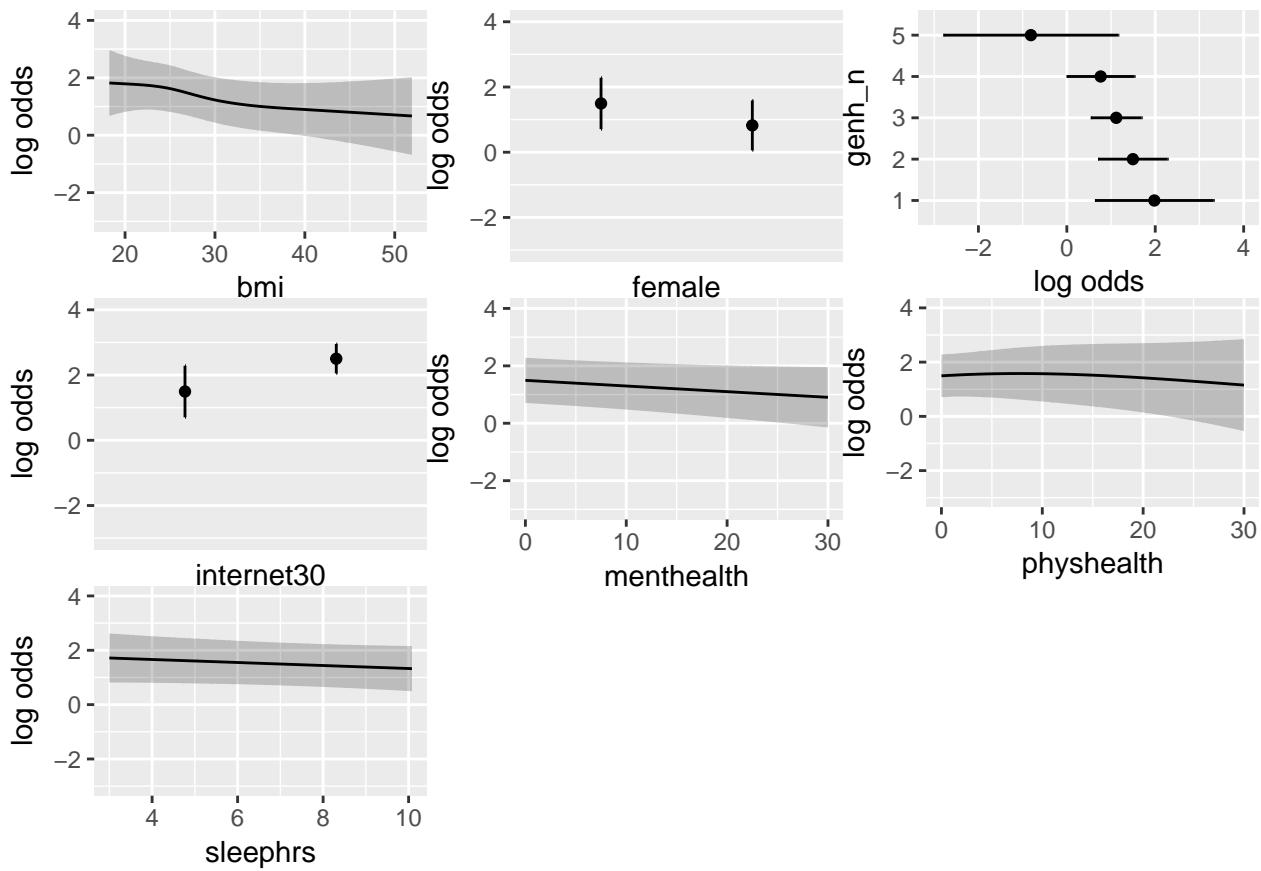
And here is the nomogram.

```
plot(nomogram(m2_imp, fun = plogis))
```



Here are the descriptive model plots, on the scale of $\text{Pr}(\text{exerany} = 1)$:

```
ggplot(Predict(m2_imp), fun = plogis)
```



Validation of summary statistics:

```
validate(m2_imp)
```

	index.orig	training	test	optimism	index.corrected	n
Dxy	0.4950	0.5193	0.4672	0.0521	0.4430	40
R2	0.2179	0.2461	0.1885	0.0576	0.1603	40
Intercept	0.0000	0.0000	0.1847	-0.1847	0.1847	40
Slope	1.0000	1.0000	0.8311	0.1689	0.8311	40
Emax	0.0000	0.0000	0.0759	0.0759	0.0759	40
D	0.1561	0.1787	0.1335	0.0452	0.1109	40
U	-0.0019	-0.0019	0.0054	-0.0073	0.0054	40
Q	0.1580	0.1807	0.1281	0.0526	0.1055	40
B	0.1514	0.1473	0.1557	-0.0084	0.1598	40
g	1.0618	1.1583	0.9678	0.1905	0.8713	40
gp	0.1801	0.1901	0.1651	0.0250	0.1551	40

14.12 Models with and without Imputation

Model	1	1	2	2
Imputation	Simple	Multiple	Sim.	Mult.

Model	1	1	2	2
nominal R ²	0.204	0.204	0.214	0.218
nominal C	0.741	0.741	0.744	0.746
validated R ²	0.170	0.168	0.165	0.160
validated C	0.722	0.724	0.720	0.722

So, what can we conclude about the discrimination results?

Chapter 15

Linear Regression and the `smartcle1` data

15.1 The `smartcle1` data

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

Variable	Description
<code>SEQNO</code>	respondent identification number (all begin with 2016)
<code>physhealth</code>	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?
<code>menthealth</code>	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?
<code>poorhealth</code>	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?
<code>genhealth</code>	Would you say that in general, your health is ... (five categories: Excellent, Very Good, Good, Fair or Poor)
<code>bmi</code>	Body mass index, in kg/m ²
<code>female</code>	Sex, 1 = female, 0 = male
<code>internet30</code>	Have you used the internet in the past 30 days? (1 = yes, 0 = no)
<code>exerany</code>	During the past month, other than your regular job, did you participate in any physical activities or exercises such as running, calisthenics, golf, gardening, or walking for exercise? (1 = yes, 0 = no)
<code>sleephrs</code>	On average, how many hours of sleep do you get in a 24-hour period?
<code>alcdays</code>	How many days during the past 30 days did you have at least one drink of any alcoholic beverage such as beer, wine, a malt beverage or liquor?

In this section, we'll use some of the variables described above to predict the quantitative outcome: `sleephrs`.

15.2 Thinking About Non-Linear Terms

We have enough observations here to consider some non-linearity for our model.

In addition, since the `genhealth` variable is an ordinal variable and multi-categorical, we should consider how to model it. We have three options:

1. include it as a factor in the model (the default approach)
2. build a numeric version of that variable, and then restrict our model to treat that numeric variable as ordinal (forcing the categories to affect the `exerany` probabilities in an ordinal way), rather than as a simple nominal factor (so that if the effect of fair vs. good was to decrease the probability of ‘exerany’, then the effect of poor vs. good would have to decrease the probability at least as much as fair vs. good did.) Treating the `genhealth` variable as ordinal could be accomplished with the `scored` function in the `rms` package.
3. build a numeric version of `genhealth` and then use the `catg` function to specify the predictor as nominal and categorical, but this will lead to essentially the same model as choice 1.

Suppose we’ve decided to treat the `genhealth` data as categorical, without restricting the effect of its various levels to be ordinal. Suppose also that we’ve decided to include the following eight variables in our model for `sleephrs`:

- `genhealth`
- `menthealth`
- `bmi`
- `female`
- `internet30`
- `exerany`
- `alcdays`

Suppose we have a subject matter understanding that:

- the impact of `bmi` on `sleephrs` is affected by `female`, so we plan a `female` x `bmi` interaction term
- we’re using `internet30` as a proxy for poverty, and we think that an interaction with `menthealth` will be helpful in our model as well.

Note that we do have some missing values in some of these predictors and in our outcome, so we’ll have to deal with that soon.

```
smartcle1 %>% select(sleephrs, alcdays, bmi, exerany, female, genhealth, internet30, menthealth) %>%
  skim()
```

Skim summary statistics

n obs: 1036
n variables: 8

Variable type: factor

variable	missing	complete	n	n_unique
<code>genhealth</code>	3	1036	1036	5

top_counts ordered
`2_V: 350, 3_G: 344, 1_E: 173, 4_F: 122` FALSE

Variable type: integer

variable	missing	complete	n	mean	sd	p0	p25	median	p75	p100
<code>alcdays</code>	46	990	1036	4.65	8.05	0	0	1	4	30
<code>exerany</code>	3	1033	1036	0.76	0.43	0	1	1	1	1
<code>female</code>	0	1036	1036	0.6	0.49	0	0	1	1	1
<code>internet30</code>	6	1030	1036	0.81	0.39	0	1	1	1	1
<code>menthealth</code>	11	1025	1036	2.72	6.82	0	0	0	2	30
<code>sleephrs</code>	8	1028	1036	7.02	1.53	1	6	7	8	20

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

15.3 A First Model for sleephrs (Complete Case Analysis)

Suppose we develop a main-effects kitchen sink model (model `mod.A` below) fitted to these predictors without the benefit of any non-linear terms except the two pre-planned interactions. We'll run the model quickly here to ensure that the code runs, in a complete case analysis, without drawing any conclusions, really.

```
mod.A <- ols(sleephrs ~ alcdays + bmi*female + exerany +
               genhealth + internet30*menthealth,
               data = smartcle1)
mod.A
```

```
Frequencies of Missing Values Due to Each Variable
sleephrs    alcdays        bmi     female    exerany  genhealth
          8         46        84          0         3         3
internet30 menthealth
          6         11
```

Linear Regression Model

```
ols(formula = sleephrs ~ alcdays + bmi * female + exerany + genhealth +
    internet30 * menthealth, data = smartcle1)
```

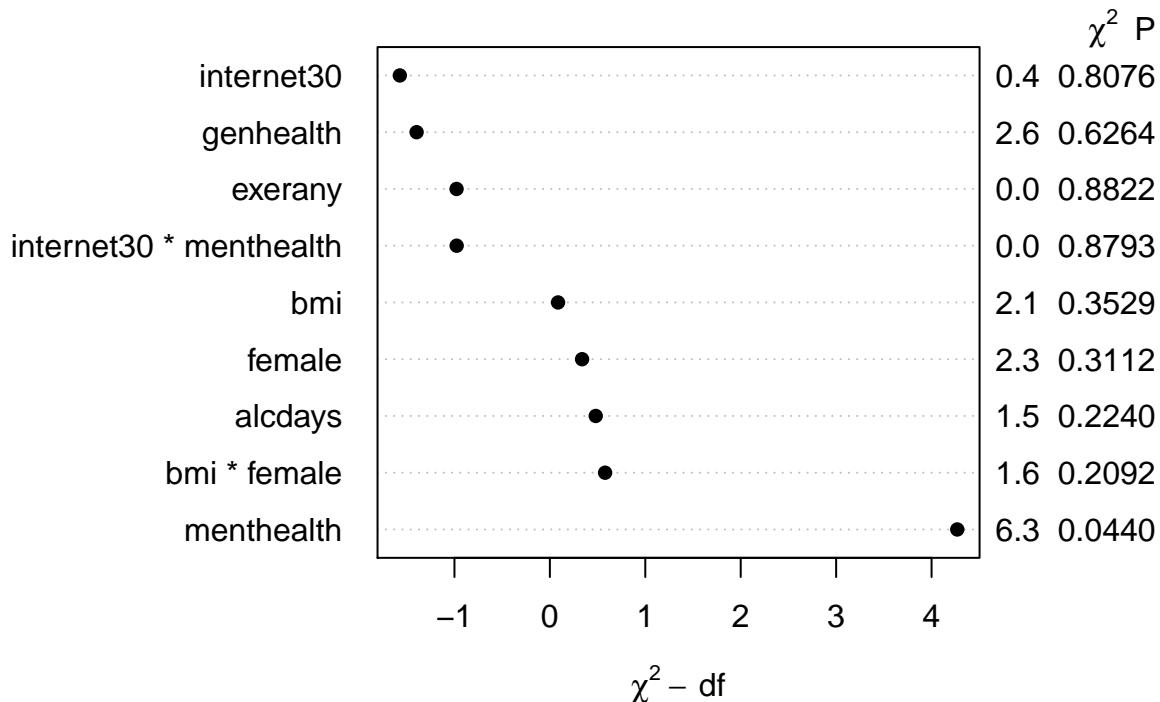
	Model Likelihood		Discrimination	
	Ratio	Test	Indexes	
Obs	903	LR chi2	16.80	R2 0.018
sigma1.4784	d.f.		12	R2 adj 0.005
d.f.	890	Pr(> chi2)	0.1571	g 0.209

Residuals

	Min	1Q	Median	3Q	Max
	-6.10511	-0.96284	-0.02392	0.89816	13.66445

	Coef	S.E.	t	Pr(> t)
Intercept	6.8562	0.4336	15.81	<0.0001
alcdays	-0.0077	0.0063	-1.22	0.2240
bmi	0.0075	0.0135	0.56	0.5784
female	0.6701	0.4729	1.42	0.1569
exerany	-0.0185	0.1250	-0.15	0.8822
genhealth=2_VeryGood	0.1267	0.1474	0.86	0.3903
genhealth=3_Good	0.0930	0.1530	0.61	0.5433
genhealth=4_Fair	-0.1099	0.2019	-0.54	0.5864
genhealth=5_Poor	-0.1118	0.2828	-0.40	0.6927
internet30	-0.0768	0.1434	-0.54	0.5926
menthealth	-0.0171	0.0161	-1.06	0.2902
bmi * female	-0.0207	0.0165	-1.26	0.2092
internet30 * menthealth	-0.0027	0.0179	-0.15	0.8793

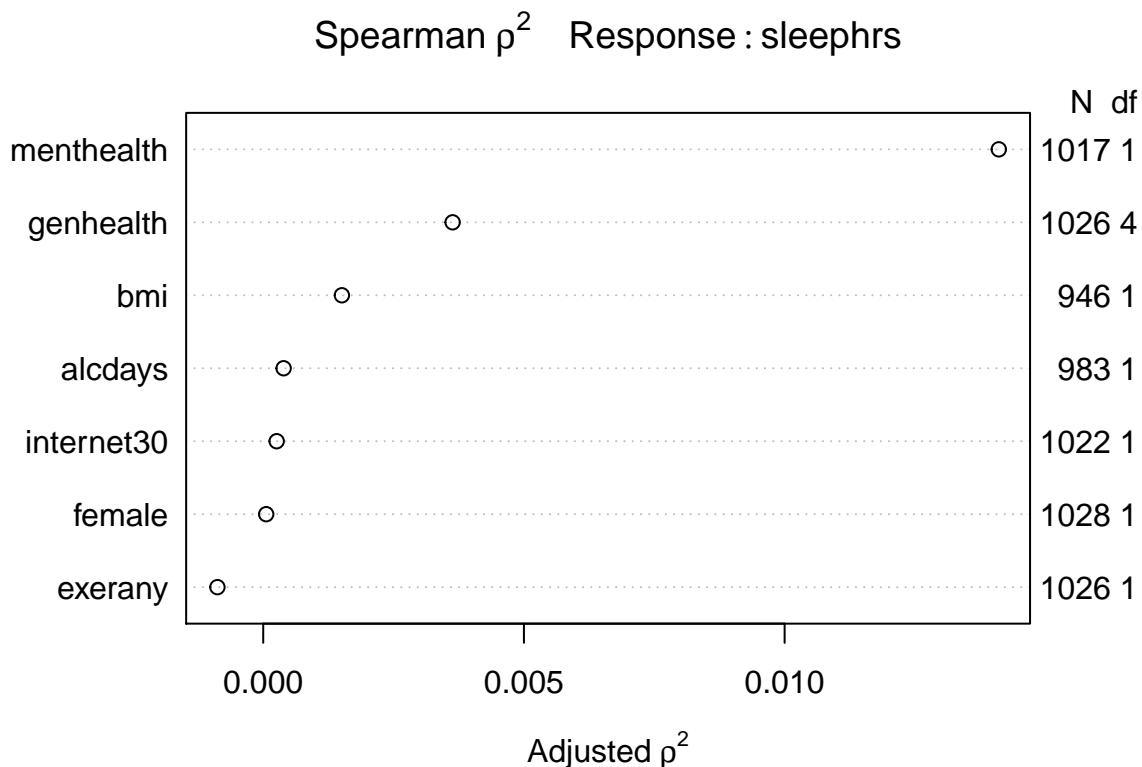
```
plot(anova(mod.A))
```



15.4 Building a Larger Model: Spearman ρ^2 Plot

Before we impute, we might also consider the use of a Spearman ρ^2 plot to decide how best to spend degrees of freedom on non-linear terms in our model for `sleephrs` using these predictors. Since we're already planning some interaction terms, I'll keep them in mind as I look at this plot.

```
sp_smart2 <- spearman2(sleephrs ~ genhealth + exerany +
                           female + internet30 + menthealth +
                           bmi + alcdays, data = smartcle1)
plot(sp_smart2)
```



We see that the best candidate for a non-linear term is the `menthealth` variable, according to this plot, followed by the `genhealth` and `bmi` predictors. I will wind up fitting a model including the following non-linear terms...

- our pre-planned `female` x `bmi` and `internet30` x `menthealth` interaction terms,
- a restricted cubic spline with 5 knots for `menthealth`
- an interaction between `genhealth` and the linear part of `menthealth`
- a restricted cubic spline with 4 knots for `bmi` (so the interaction term with `female` will need to account for this and restrict our interaction to the linear piece of `bmi`)

15.5 A Second Model for `sleephrs` (Complete Cases)

Here's the resulting model fit without worrying about imputation yet. This is just to make sure our code works. Note that I'm inserting the main effects of our interaction terms explicitly before including the interaction terms themselves, and that I need to use `%ia%` to include the interaction terms where one of the terms is included in the model with a spline. Again, I won't draw any serious conclusions yet.

```
mod.B <- ols(sleephrs ~ rcs(menthealth, 5) + genhealth +
               genhealth %ia% menthealth + rcs(bmi, 4) +
               female + female %ia% bmi + internet30 +
               internet30 %ia% menthealth + alcdays +
               exerany, data = smartcle1)
mod.B
```

Frequencies of Missing Values Due to Each Variable
 sleephrs menthealth genhealth bmi female internet30

8	11	3	84
alcdays	exerany		0
46	3		6

Linear Regression Model

```
ols(formula = sleephrs ~ rcs(menthealth, 5) + genhealth + genhealth %ia%
  menthealth + rcs(bmi, 4) + female + female %ia% bmi + internet30 +
  internet30 %ia% menthealth + alcdays + exerany, data = smartcle1)
```

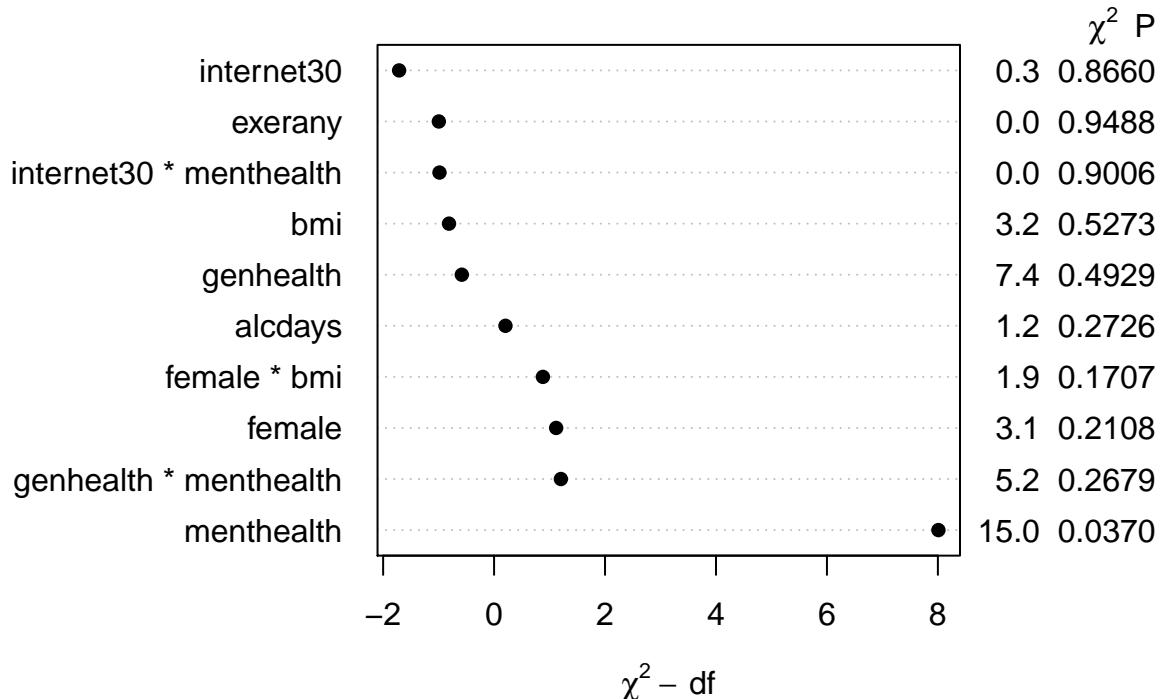
	Model Likelihood		Discrimination		
	Ratio	Test	Indexes		
Obs	903	LR chi2	27.03	R2	0.029
sigma1	4.759	d.f.	19	R2 adj	0.009
d.f.	883	Pr(> chi2)	0.1040	g	0.258

Residuals

Min	1Q	Median	3Q	Max
-6.11835	-0.93539	-0.02598	0.87085	13.68653

	Coef	S.E.	t	Pr(> t)
Intercept	5.9217	1.0518	5.63	<0.0001
menthealth	-0.0759	0.0424	-1.79	0.0736
menthealth'	0.5218	0.3285	1.59	0.1126
genhealth=2_VeryGood	0.1716	0.1554	1.10	0.2699
genhealth=3_Good	0.0630	0.1624	0.39	0.6982
genhealth=4_Fair	-0.0269	0.2216	-0.12	0.9035
genhealth=5_Poor	-0.1593	0.3468	-0.46	0.6461
genhealth=2_VeryGood * menthealth	-0.0382	0.0287	-1.33	0.1836
genhealth=3_Good * menthealth	0.0157	0.0227	0.69	0.4893
genhealth=4_Fair * menthealth	-0.0112	0.0239	-0.47	0.6415
genhealth=5_Poor * menthealth	0.0109	0.0286	0.38	0.7044
bmi	0.0469	0.0449	1.05	0.2962
bmi'	-0.1237	0.2050	-0.60	0.5463
bmi''	0.2755	0.5607	0.49	0.6233
female	0.7588	0.4796	1.58	0.1140
female * bmi	-0.0228	0.0166	-1.37	0.1707
internet30	-0.0765	0.1443	-0.53	0.5962
internet30 * menthealth	0.0023	0.0184	0.12	0.9006
alcdays	-0.0069	0.0063	-1.10	0.2726
exerany	0.0080	0.1254	0.06	0.9488

```
plot(anova(mod.B))
```



It looks like `menthealth` may be the only significant term.

15.6 Dealing with Missing Data via Simple Imputation

One approach we might take in this problem is to use simple imputation to deal with our missing values. I will proceed as follows:

1. Omit all cases where the outcome `sleephrs` is missing.
2. Determine (and plot) the remaining missingness.
3. Use simple imputation for all predictors, and build a new data set with “complete” data.
4. Re-fit the proposed models using this new data set.

15.6.1 Omit cases where the outcome is missing

We need to drop the cases where `sleephrs` is missing in `smartcle1`. We'll begin creating an imputed data set, called `smartcle2_imp0`, by filtering on complete data for `sleephrs`, as follows. Note that the `describe` function in `Hmisc` specifies the number of non-missing observations in `n`.

```
Hmisc::describe(smartcle1$sleephrs)
```

```
smartcle1$sleephrs
  n  missing distinct   Info    Mean     Gmd     .05     .10
 1028      8       14  0.935  7.018  1.505      5       5
  .25     .50       .75     .90     .95
  6       7       8        8       9
```

Value	1	2	3	4	5	6	7	8	9	10
Frequency	5	1	9	25	72	221	318	301	42	24
Proportion	0.005	0.001	0.009	0.024	0.070	0.215	0.309	0.293	0.041	0.023

Value	11	12	16	20
Frequency	2	4	2	2
Proportion	0.002	0.004	0.002	0.002

```
smartcle2_imp0 <- smartcle1 %>%
  filter(complete.cases(sleephrs)) %>%
  select(SEQNO, sleephrs, alcdays, bmi, exerany, female,
         genhealth, internet30, menthealth)

Hmisc::describe(smartcle2_imp0$sleephrs)
```

smartcle2_imp0\$sleephrs									
	n	missing	distinct	Info	Mean	Gmd	.05	.10	
1028		0	14	0.935	7.018	1.505	5	5	
.25		.50	.75	.90	.95				
6		7	8	8	9				

Value	1	2	3	4	5	6	7	8	9	10
Frequency	5	1	9	25	72	221	318	301	42	24
Proportion	0.005	0.001	0.009	0.024	0.070	0.215	0.309	0.293	0.041	0.023

Value	11	12	16	20
Frequency	2	4	2	2
Proportion	0.002	0.004	0.002	0.002

15.6.2 Plot the remaining missingness

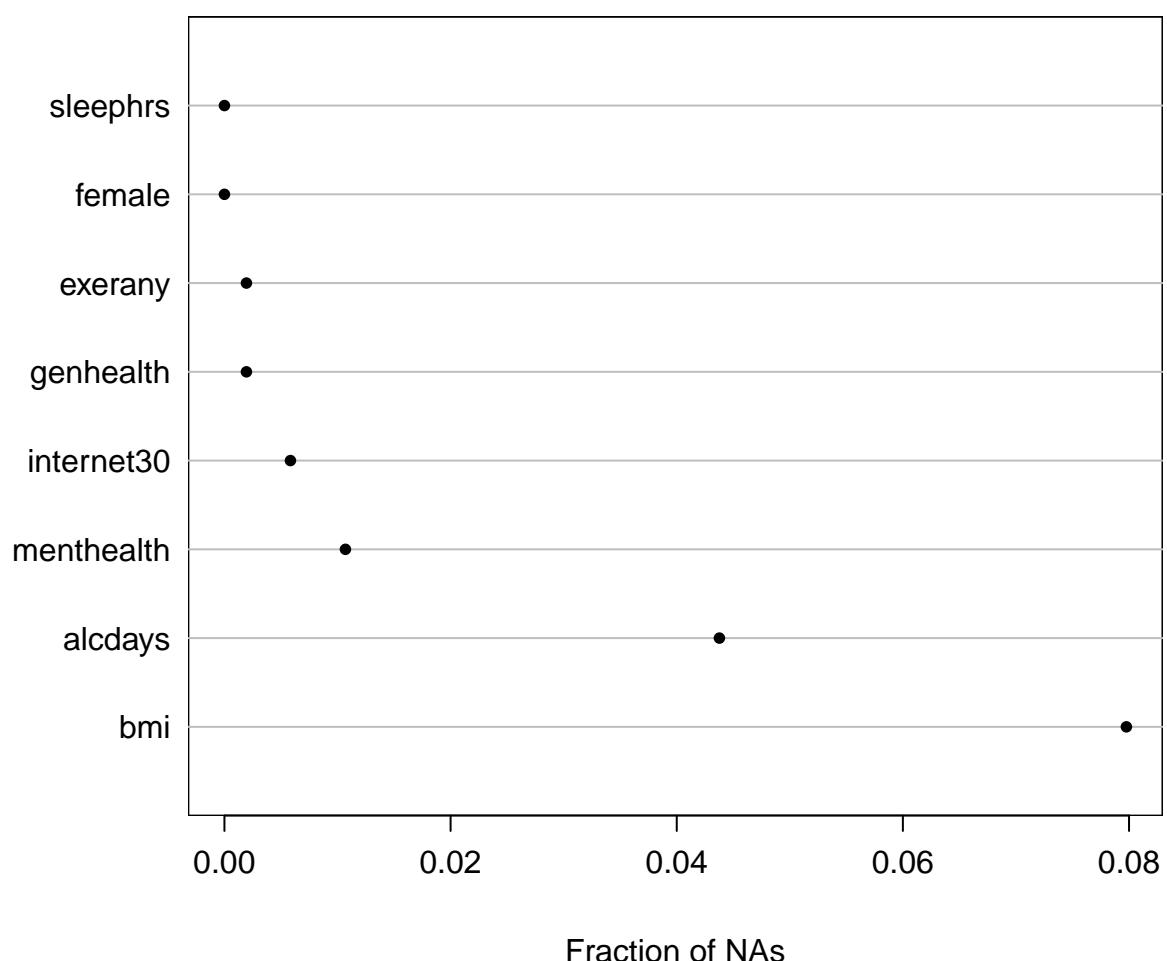
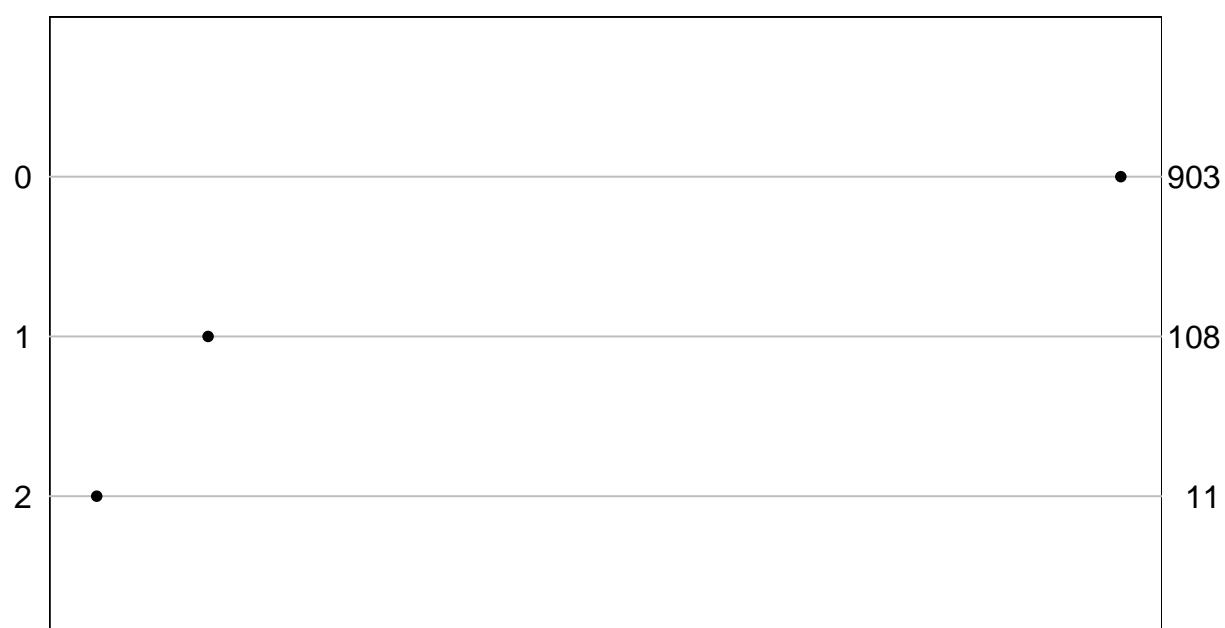
We'll look at the missing values (excluding the subject ID: SEQNO) in our new data set. Of course, we can get a count of missing values within each variable with `skim` or with:

```
colSums(is.na(smartcle2_imp0))
```

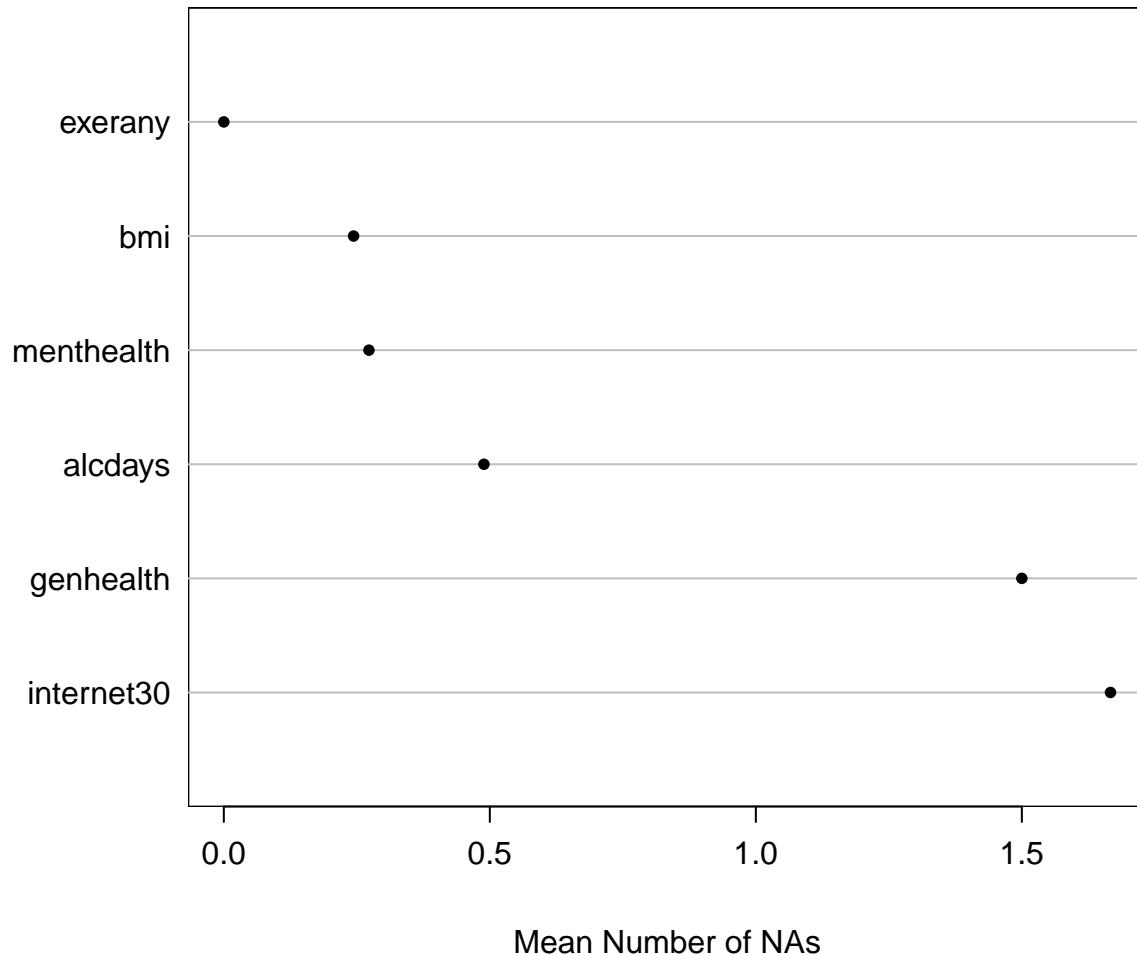
SEQNO	sleephrs	alcdays	bmi	exerany	female
0	0	45	82	2	0
genhealth	internet30	menthealth			
2	6	11			

The `Hmisc` package has a plotting approach which can help identify missingness, too.

```
naplot(naclus(select(smartcle2_imp0, -SEQNO)))
```

Fraction of NAs in each Variable**Number of Missing Variables Per Observation**

Mean Number of Other Variables Missing for Observations where Indicated Variable is NA




```

903  1  0
29   1  1
67   0  1
2    1  1
1    1  1
9    1  1
10   0  2
1    1  2
5    0  3
1    1  3
               82 148

```

We can also do this with `na.pattern` in the `Hmisc` package, but then we have to get the names of the columns, too, so that we can read off the values.

```

na.pattern(smartcle2_imp0)

pattern
0000000000 0000000001 000000010 000000101 000010000 000100000 001000000
      903         9          1          1          2          67         29
001000101 001100000 001100010
      1          10         5

names(smartcle2_imp0)

[1] "SEQNO"      "sleephrs"    "alcdays"     "bmi"        "exerany"
[6] "female"     "genhealth"   "internet30" "menthealth"

```

15.6.3 Use simple imputation, build a new data set

The only variables that require no imputation are `sleephrs` and `female`. In this case, we need to impute:

- 82 `bmi` values (which are quantitative)
- 45 `alcdays` values (quantitative, must fall between 0 and 30)
- 11 `menthealth` values (quantitative, must fall between 0 and 30)
- 6 `internet30` values (which are 1/0)
- 2 `exerany` values (which are also 1/0)
- and 2 `genhealth` values (which are multi-categorical, so we need to convert them to numbers in order to get the imputation process to work properly)

```

smartcle2_imp0 <- smartcle2_imp0 %>%
  mutate(genh_n = as.numeric(genhealth))

smartcle2_imp0 %>% count(genhealth, genh_n)

# A tibble: 6 x 3
  genhealth  genh_n    n
  <fct>       <dbl> <int>
1 1_Excellent 1.    172
2 2_VeryGood  2.    349
3 3_Good      3.    341
4 4_Fair      4.    120
5 5_Poor      5.     44
6 <NA>          NA     2

```

I'll work from the bottom up, using various `imputation` functions to accomplish the imputations I want. In this case, I'll use predictive mean matching for the categorical data, and linear models or elastic net

approaches for the quantitative data. Be sure to set a seed beforehand so you can replicate your work.

```
set.seed(432109)

smartcle2_imp1 <- smartcle2_imp0 %>%
  impute_pmm(genh_n ~ female) %>%
  impute_pmm(exerany + internet30 ~ female + genh_n) %>%
  impute_lm(menthealth ~ female + genh_n + exerany) %>%
  impute_en(alcdays ~ female + genh_n + menthealth) %>%
  impute_en(bmi ~ alcdays + exerany + genh_n)
```

After the imputations are complete, I'll back out of the numeric version of `genhealth`, called `genh_n` back to my original variable, then check to be sure I now have no missing values.

```
smartcle2_imp1 <- smartcle2_imp1 %>%
  mutate(genhealth = fct_recode(factor(genh_n),
    "1_Excellent" = "1",
    "2_VeryGood" = "2",
    "3_Good" = "3",
    "4_Fair" = "4",
    "5_Poor" = "5"))

smartcle2_imp1 %>% count(genhealth, genh_n)
```

```
# A tibble: 5 x 3
  genhealth   genh_n     n
  <fct>       <dbl> <int>
1 1_Excellent 1.    172
2 2_VeryGood  2.    350
3 3_Good      3.    342
4 4_Fair      4.    120
5 5_Poor      5.     44
```

```
colSums(is.na(smartcle2_imp1))
```

SEQNO	sleephrs	alcdays	bmi	exerany	female
0	0	0	0	0	0
genhealth	internet30	menthealth	genh_n		
0	0	0	0		

OK. Looks good. I now have a data frame called `smartcle2_imp1` with no missingness, which I can use to fit my logistic regression models. Let's do that next, and then return to the problem of accounting for missingness through multiple imputation.

15.7 Refitting Model A with simply imputed data

Using the numeric version of the `genhealth` data, called `genh_n`, will ease the reviewing of later output, so we'll do that here, making sure R knows that `genh_n` describes a categorical factor.

```
d <- datadist(smartcle2_imp1)
options(datadist = "d")

mod.A2 <- ols(sleephrs ~ alcdays + bmi*female + exerany +
  genhealth + internet30*menthealth,
  data = smartcle2_imp1, x = TRUE, y = TRUE)
mod.A2
```

Linear Regression Model

```
ols(formula = sleephrs ~ alcdays + bmi * female + exerany + genhealth +
    internet30 * menthealth, data = smartcle2_imp1, x = TRUE,
    y = TRUE)
```

	Model	Likelihood	Discrimination
		Ratio Test	Indexes
Obs	1028	LR chi2	24.61
sigma1	5.193	d.f.	12
d.f.	1015	Pr(> chi2)	0.0168
		R2	0.024
		R2 adj	0.012
		g	0.245

Residuals

Min	1Q	Median	3Q	Max
-6.15720	-0.95552	-0.03011	0.86804	13.80609

	Coef	S.E.	t	Pr(> t)
Intercept	6.9916	0.4305	16.24	<0.0001
alcdays	-0.0101	0.0062	-1.62	0.1048
bmi	0.0080	0.0135	0.59	0.5558
female	0.7360	0.4682	1.57	0.1162
exerany	-0.0898	0.1201	-0.75	0.4550
genhealth=2_VeryGood	0.1248	0.1430	0.87	0.3832
genhealth=3_Good	0.0912	0.1483	0.62	0.5386
genhealth=4_Fair	-0.2083	0.1945	-1.07	0.2844
genhealth=5_Poor	0.1361	0.2722	0.50	0.6173
internet30	-0.1282	0.1374	-0.93	0.3510
menthealth	-0.0228	0.0164	-1.39	0.1635
bmi * female	-0.0248	0.0163	-1.52	0.1284
internet30 * menthealth	0.0004	0.0180	0.02	0.9811

All right. We've used 1028 observations, which is correct (after deleting the eight with missing `sleephrs`. The model shows a lousy R² value of 0.024 after imputation.

15.7.1 Validating Summary Statistics

```
set.seed(432487)
validate(mod.A2)
```

	index.orig	training	test	optimism	index.corrected	n
R-square	0.0237	0.0402	0.0067	0.0335	-0.0099	40
MSE	2.2791	2.2812	2.3186	-0.0374	2.3165	40
g	0.2453	0.3112	0.1902	0.1210	0.1243	40
Intercept	0.0000	0.0000	2.5979	-2.5979	2.5979	40
Slope	1.0000	1.0000	0.6302	0.3698	0.6302	40

As poor as the nominal R² value is, it appears that the model's description of summary statistics is still optimistic. After validation, we cannot claim any meaningful predictive value at all, with a negative (impossible) R² value. This output suggests that in a new sample of data, our model shouldn't be expected to do anything useful at all.

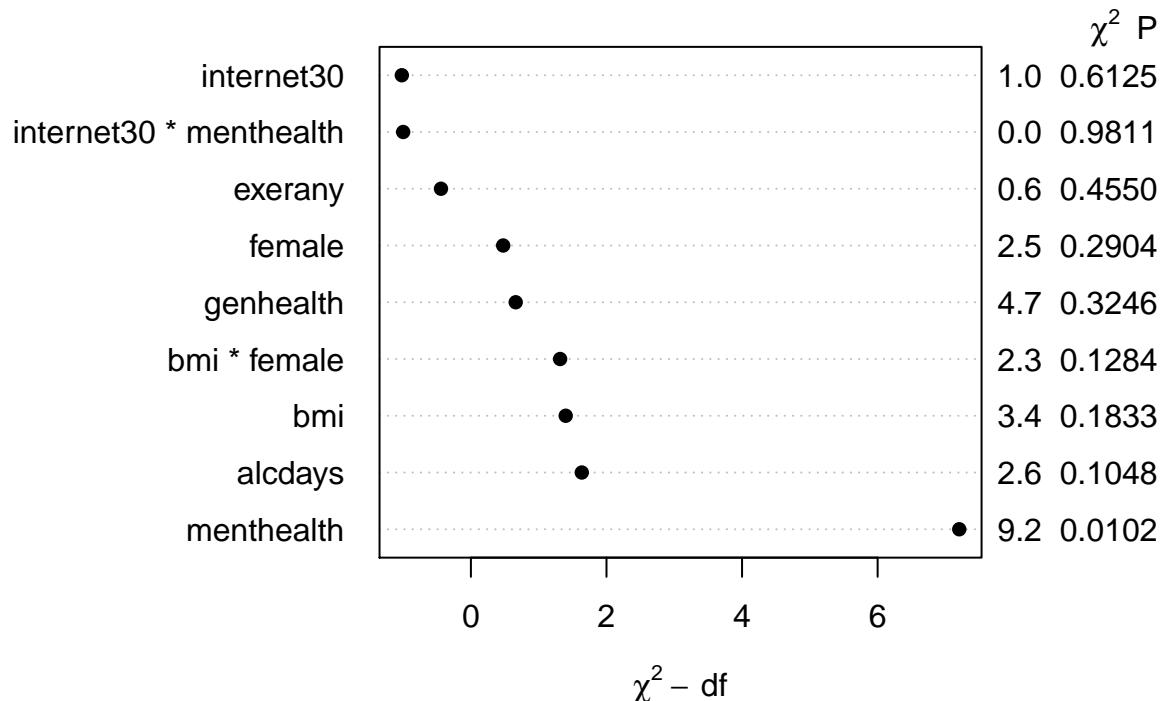
15.7.2 ANOVA for the model

Next, let's look at the ANOVA comparisons for this (admittedly terrible) prediction model.

```
anova(mod.A2)
```

Analysis of Variance			Response: sleephrs		
Factor	d.f.	Partial SS			
alcdays	1	6.083831e+00			
bmi (Factor+Higher Order Factors)	2	7.844991e+00			
All Interactions	1	5.345522e+00			
female (Factor+Higher Order Factors)	2	5.715093e+00			
All Interactions	1	5.345522e+00			
exerany	1	1.289411e+00			
genhealth	4	1.075881e+01			
internet30 (Factor+Higher Order Factors)	2	2.264350e+00			
All Interactions	1	1.301337e-03			
menthealth (Factor+Higher Order Factors)	2	2.124566e+01			
All Interactions	1	1.301337e-03			
bmi * female (Factor+Higher Order Factors)	1	5.345522e+00			
internet30 * menthealth (Factor+Higher Order Factors)	1	1.301337e-03			
TOTAL INTERACTION	2	5.350214e+00			
REGRESSION	12	5.676073e+01			
ERROR	1015	2.342924e+03			
MS	F	P			
6.083830768	2.64	0.1048			
3.922495636	1.70	0.1833			
5.345521775	2.32	0.1284			
2.857546649	1.24	0.2904			
5.345521775	2.32	0.1284			
1.289411070	0.56	0.4550			
2.689702722	1.17	0.3246			
1.132174866	0.49	0.6125			
0.001301337	0.00	0.9811			
10.622828725	4.60	0.0102			
0.001301337	0.00	0.9811			
5.345521775	2.32	0.1284			
0.001301337	0.00	0.9811			
2.675106916	1.16	0.3142			
4.730061148	2.05	0.0178			
2.308299597					

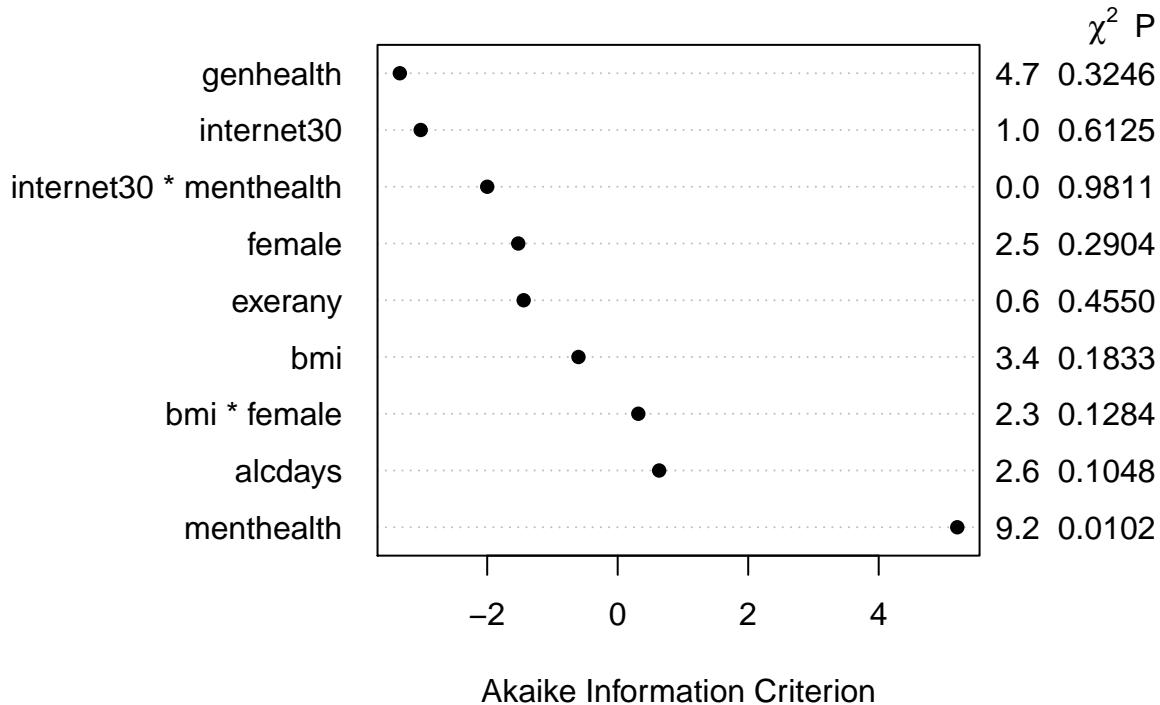
```
plot(anova(mod.A2))
```



Only `menthealth` appears to carry statistically significant predictive value here.

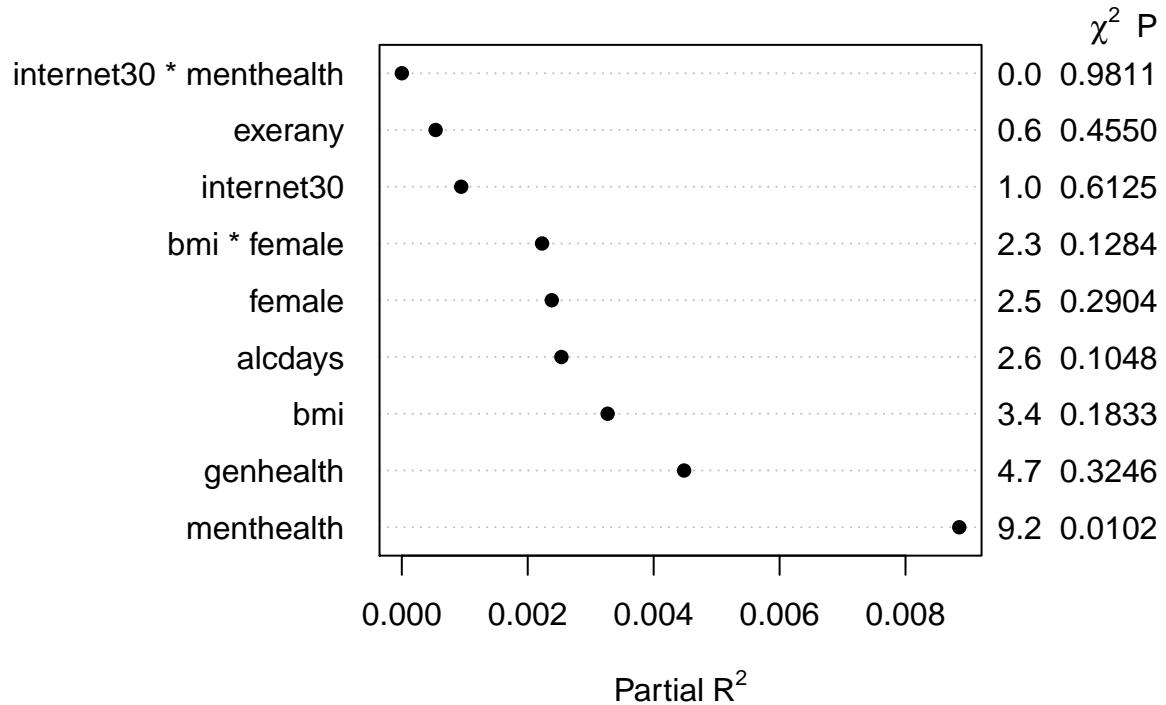
We can also build a plot of the AIC values attributable to each piece of the model.

```
plot(anova(mod.A2), what="aic")
```



We can also plot the Partial R² values for each predictor. The partial R² for `internet30`, for instance, is the R² value that you would obtain if you first regress `internet30` on every other predictor in the model, take the residuals, and then regress those on `internet30`. It simply tells you, then, how much of the tiny amount of variation explained by the model as a whole is accounted for by each predictor after all of the other ones have already been accounted for. The partial R² values, therefore, do not sum up to the total R² explained by the model. In our case, the `menthealth` variable is again far and away the most “useful” variable.

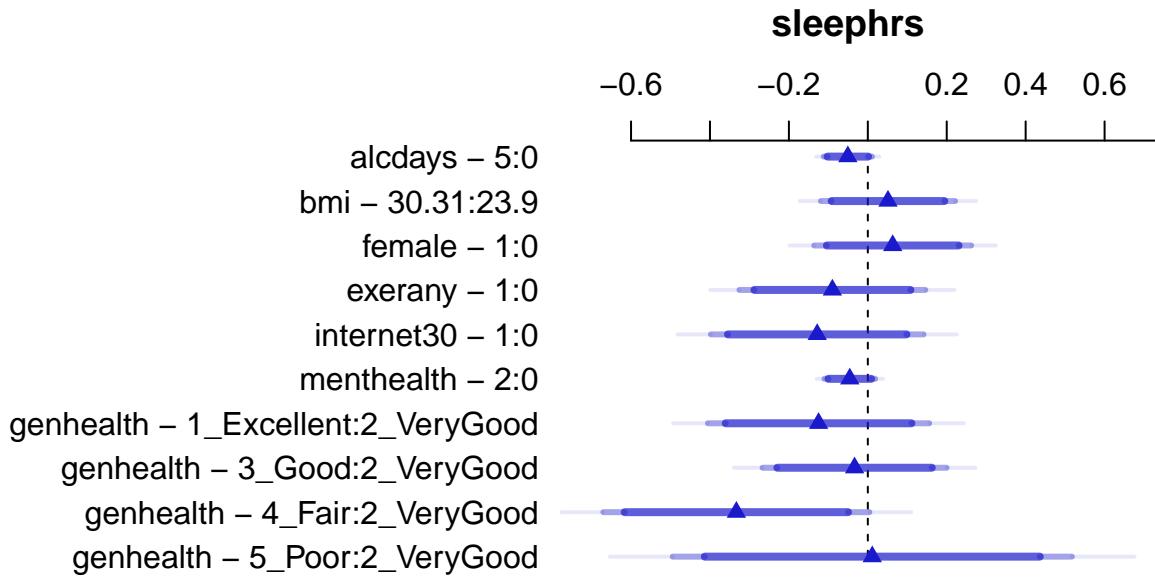
```
plot(anova(mod.A2), what="partial")
```



15.7.3 Summarizing Effect Size

How big are the effects we see?

```
plot(summary(mod.A2))
```



Adjusted to:bmi=27.09563 female=0 internet30=0 menthealth=0

```
summary(mod.A2)
```

Factor	Effects			Response : sleephrs		
	Low	High	Diff.	Effect	S.E.	
alcdays	0.0	5.00	5.00	-0.050632	0.031188	
bmi	23.9	30.31	6.41	0.051110	0.086734	
female	0.0	1.00	1.00	0.063083	0.101450	
exerany	0.0	1.00	1.00	-0.089786	0.120130	
internet30	0.0	1.00	1.00	-0.128250	0.137430	
menthealth	0.0	2.00	2.00	-0.045668	0.032753	
genhealth - 1_Excellent:2_VeryGood	2.0	1.00	NA	-0.124770	0.143030	
genhealth - 3_Good:2_VeryGood	2.0	3.00	NA	-0.033557	0.118930	
genhealth - 4_Fair:2_VeryGood	2.0	4.00	NA	-0.333090	0.172000	
genhealth - 5_Poor:2_VeryGood	2.0	5.00	NA	0.011282	0.258040	
Lower 0.95						
-0.11183	0.0105680					
-0.11909	0.2213100					
-0.13600	0.2621600					
-0.32552	0.1459500					
-0.39792	0.1414300					
-0.10994	0.0186030					
-0.40544	0.1558900					
-0.26694	0.1998200					
-0.67059	0.0044216					
-0.49508	0.5176400					
Upper 0.95						

Adjusted to: bmi=27.09563 female=0 internet30=0 menthealth=0

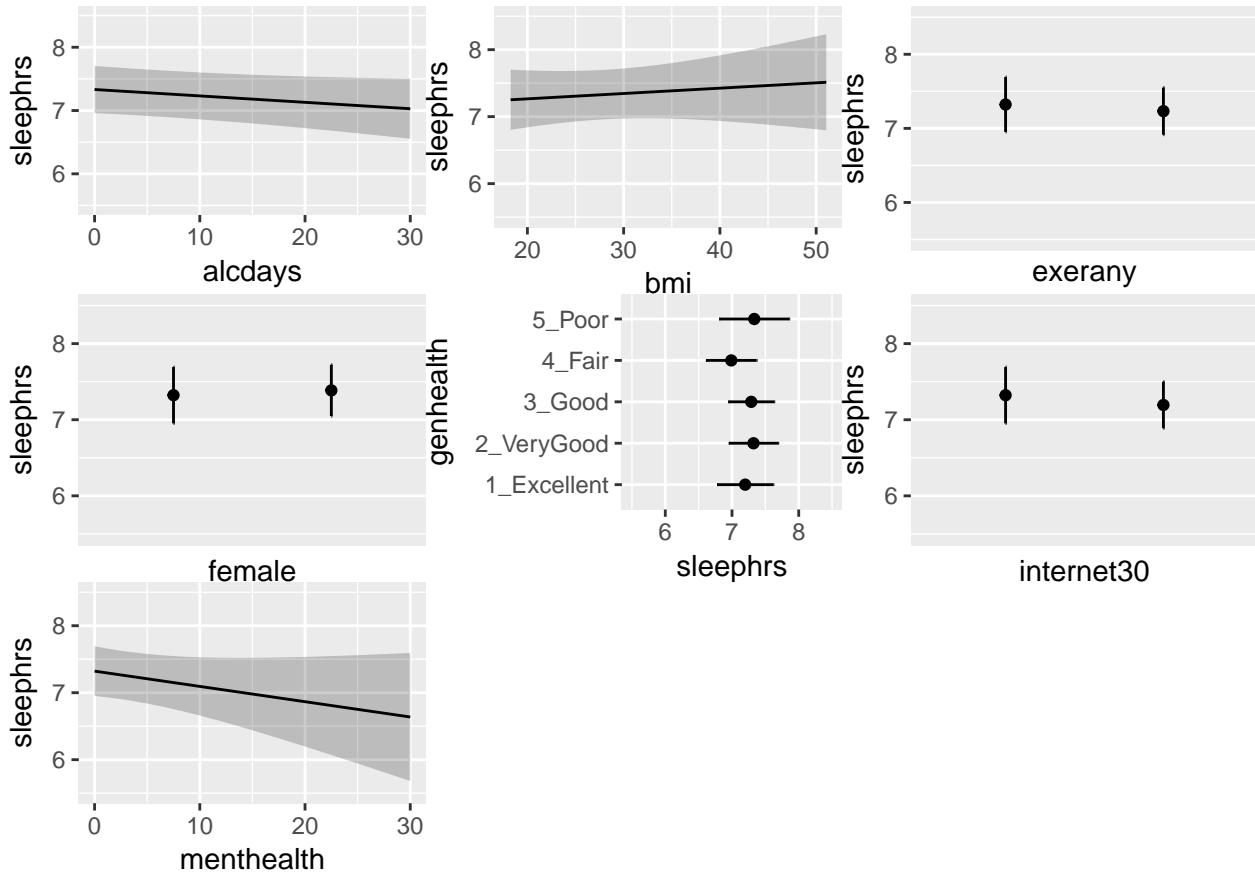
This output is easier to read as a result of using small *numeric* labels in `genh_n`, rather than the lengthy labels in `genhealth`. Interpret the results as differences in our outcome `sleephrs` associated with the expressed changes in predictors, holding the others constant.

- holding all other predictors constant, the effect of moving from `alcdays` = 0 to `alcdays` = 5 is -0.05 hours of sleep.
 - We also have a 95% confidence interval for this estimate, which is (-0.11, 0.01). Since 0 is in that interval, we cannot conclude that the effect of `alcdays` on `sleephrs` is either definitely positive or definitely negative.
- A similar approach can be used to describe the effects on `sleephrs` associated with each predictor.
- Note that each of the categories in `genh_n` is compared to a single baseline category. Here, that's category 2. R will pick the modal category: the one that appears most often in the data. The comparisons of each category against category 2 are not significant in each case, at the 5% level.

15.7.4 Plotting the Model with `ggplot` and `Predict`

Let's look at a series of plots describing the model for `sleephrs`.

```
ggplot(Predict(mod.A2))
```

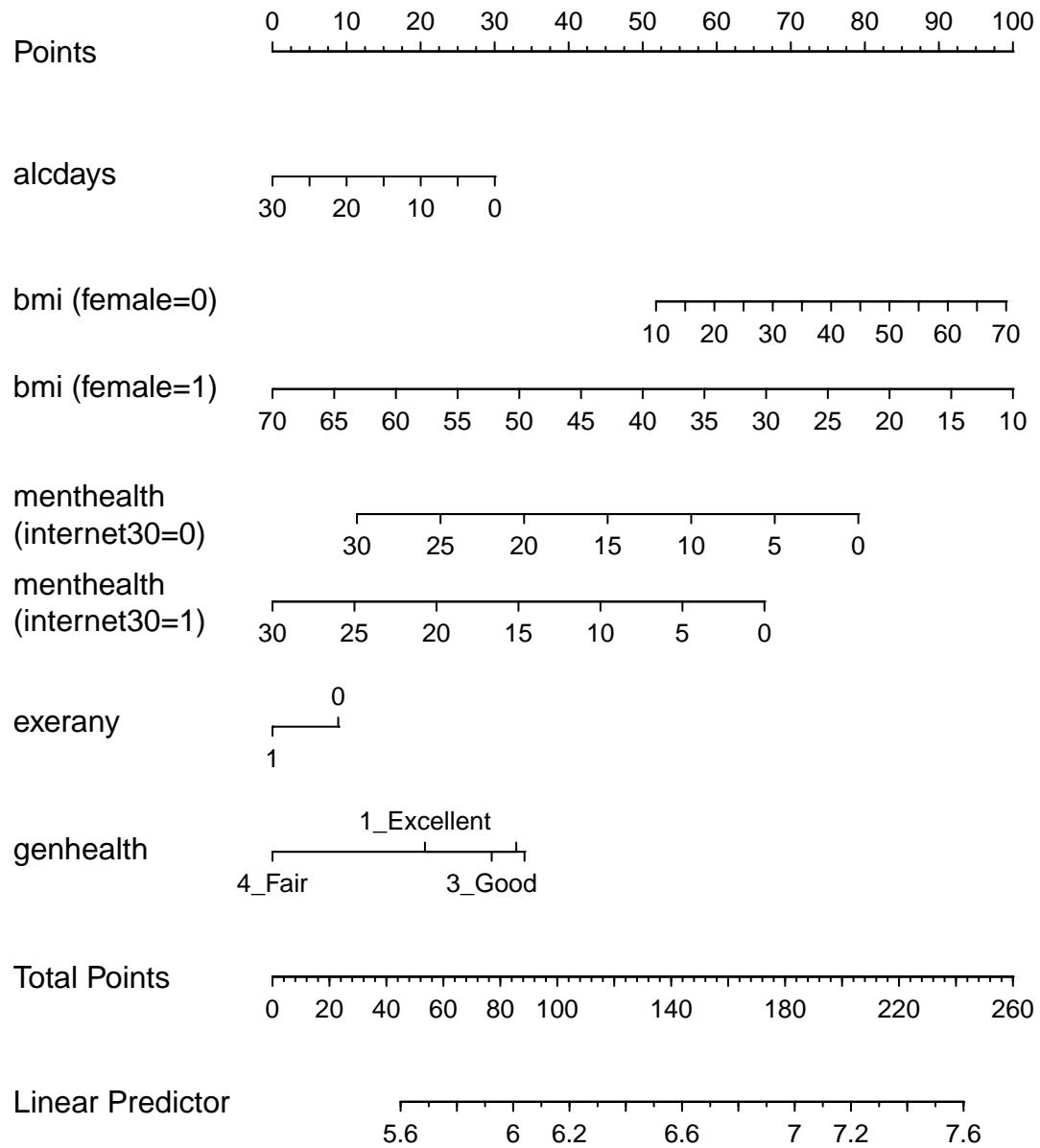


This helps us describe what is happening in terms of direction at least. For example,

- As `menthealth` increases, predicted `sleephrs` actually decreases.
- In general, though, the impact of these predictors on `sleephrs` appears minimal.

15.7.5 Plotting the model with a nomogram

```
plot(nomogram(mod.A2))
```



Note the impact of our interaction terms, and how we have two lines for `bmi` and two lines for `menthealth`

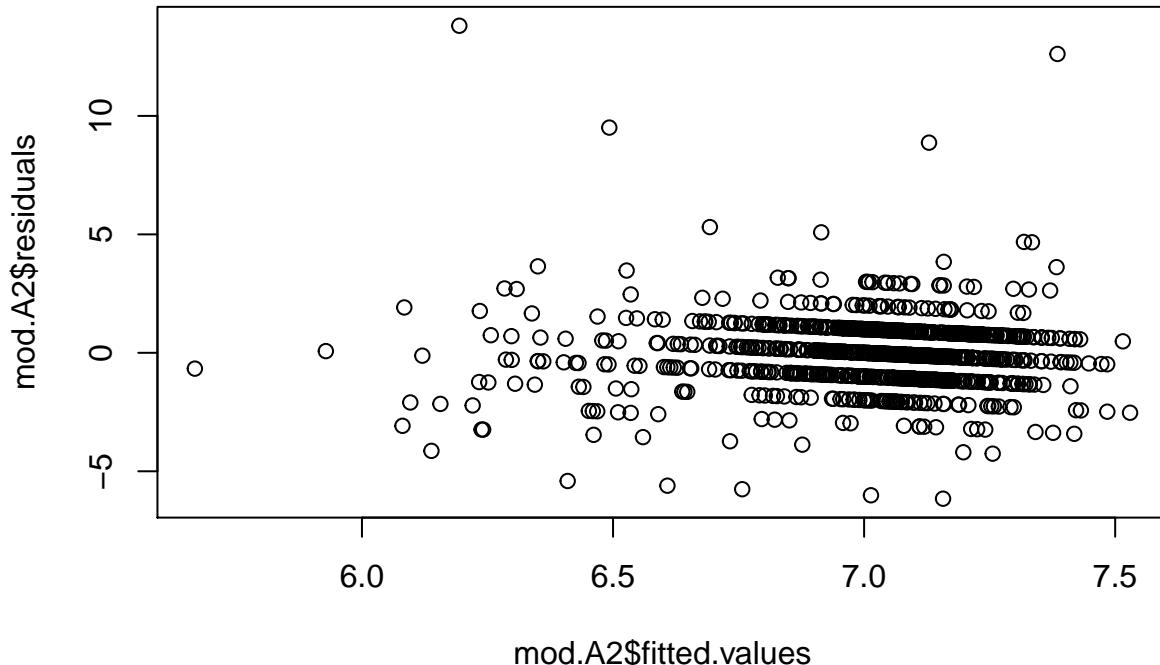
that ‘ that come out of our product terms. As with any nomogram, to make a prediction we:

1. find the values of each of our predictors in the scales, and travel vertically up to the Points line to read off the Points for that predictor.
2. sum up the Points across all predictors, and find that location in the Total Points line.
3. move vertically down from the total points line to find the estimated “linear predictor” (`sleephrs`)

15.7.6 Residual Plots for mod.A2

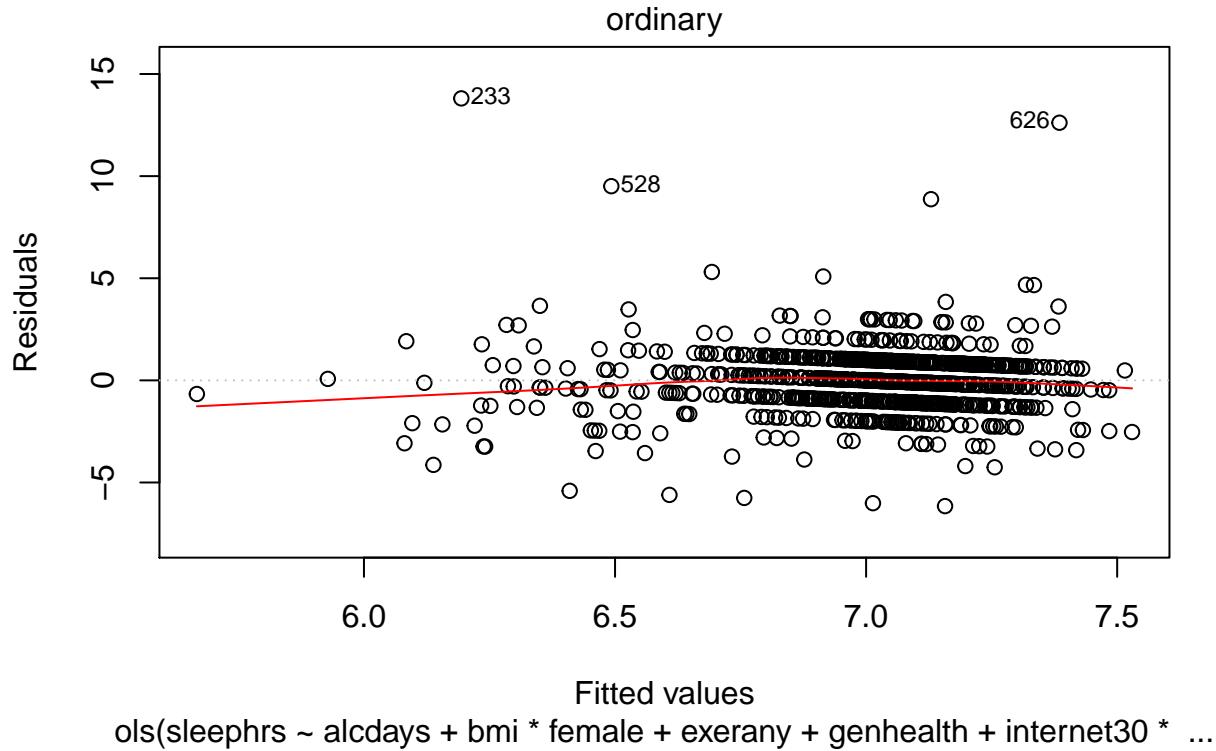
We can obtain our usual residual plots for a linear model. Or, we can obtain things like the residuals and fitted values directly, to produce a plot. For example,

```
plot(mod.A2$residuals ~ mod.A2$fitted.values)
```



Or we can get the same residuals vs. fitted values plot with:

```
plot(mod.A2, "ordinary", which = 1)
```



15.8 Refitting Model B with simply imputed data

I'll walk through the same tasks for Model `m2` that I did above for Model `m1`. Again, we're running this model after simple imputation of missing values.

Using the numeric version of the `genhealth` data, called `genh_n`, will ease the reviewing of later output, so we'll do that here, making sure R knows that `genh_n` describes a categorical factor.

```
d <- datadist(smartcle2_imp1)
options(datadist = "d")

mod.B2 <- ols(sleephrs ~ rcs(menthealth, 5) + genhealth +
               genhealth %ia% menthealth + internet30 +
               internet30 %ia% menthealth + rcs(bmi, 4) +
               female + female %ia% bmi + alcdays +
               exerany,
               data = smartcle2_imp1, x = TRUE, y = TRUE)
mod.B2
```

Linear Regression Model

```
ols(formula = sleephrs ~ rcs(menthealth, 5) + genhealth + genhealth %ia%
    menthealth + internet30 + internet30 %ia% menthealth + rcs(bmi,
    4) + female + female %ia% bmi + alcdays + exerany, data = smartcle2_imp1,
    x = TRUE, y = TRUE)
```

	Model	Likelihood	Discrimination		
		Ratio Test	Indexes		
Obs	1028	LR chi2	31.84	R2	0.031
sigma1	5.192	d.f.	19	R2 adj	0.012
d.f.	1008	Pr(> chi2)	0.0325	g	0.274

Residuals					
	Min	1Q	Median	3Q	Max
	-6.12337	-0.95462	-0.03322	0.86403	13.69249

	Coef	S.E.	t	Pr(> t)
Intercept	6.5183	1.0161	6.41	<0.0001
menthealth	-0.0605	0.0412	-1.47	0.1417
menthealth'	0.3943	0.3175	1.24	0.2146
genhealth=2_VeryGood	0.1637	0.1513	1.08	0.2793
genhealth=3_Good	0.0691	0.1580	0.44	0.6618
genhealth=4_Fair	-0.1475	0.2140	-0.69	0.4908
genhealth=5_Poor	0.2443	0.3336	0.73	0.4641
genhealth=2_VeryGood * menthealth	-0.0333	0.0270	-1.23	0.2177
genhealth=3_Good * menthealth	0.0065	0.0217	0.30	0.7636
genhealth=4_Fair * menthealth	-0.0113	0.0233	-0.48	0.6292
genhealth=5_Poor * menthealth	-0.0137	0.0275	-0.50	0.6190
internet30	-0.1221	0.1386	-0.88	0.3788
internet30 * menthealth	0.0034	0.0185	0.18	0.8544
bmi	0.0253	0.0430	0.59	0.5563
bmi'	0.0129	0.1925	0.07	0.9467
bmi''	-0.1314	0.5808	-0.23	0.8210
female	0.7775	0.4746	1.64	0.1017
female * bmi	-0.0255	0.0165	-1.54	0.1228
alcdays	-0.0094	0.0063	-1.50	0.1334
exerany	-0.0713	0.1206	-0.59	0.5545

The model still uses 1028 observations, and shows an R² value of 0.031, marginally better than what we saw in mod.A2. The likelihood ratio (drop in deviance) test is still highly significant.

15.8.1 Validating Summary Statistics

	index.orig	training	test	optimism	index.corrected	n
R-square	0.0305	0.0616	-0.0006	0.0622	-0.0317	40
MSE	2.2631	2.1637	2.3358	-0.1721	2.4352	40
g	0.2738	0.3691	0.1919	0.1772	0.0966	40
Intercept	0.0000	0.0000	3.3422	-3.3422	3.3422	40
Slope	1.0000	1.0000	0.5243	0.4757	0.5243	40

Again, the model's description of summary statistics is optimistic and we have no reason to expect the model is of any predictive value at all.

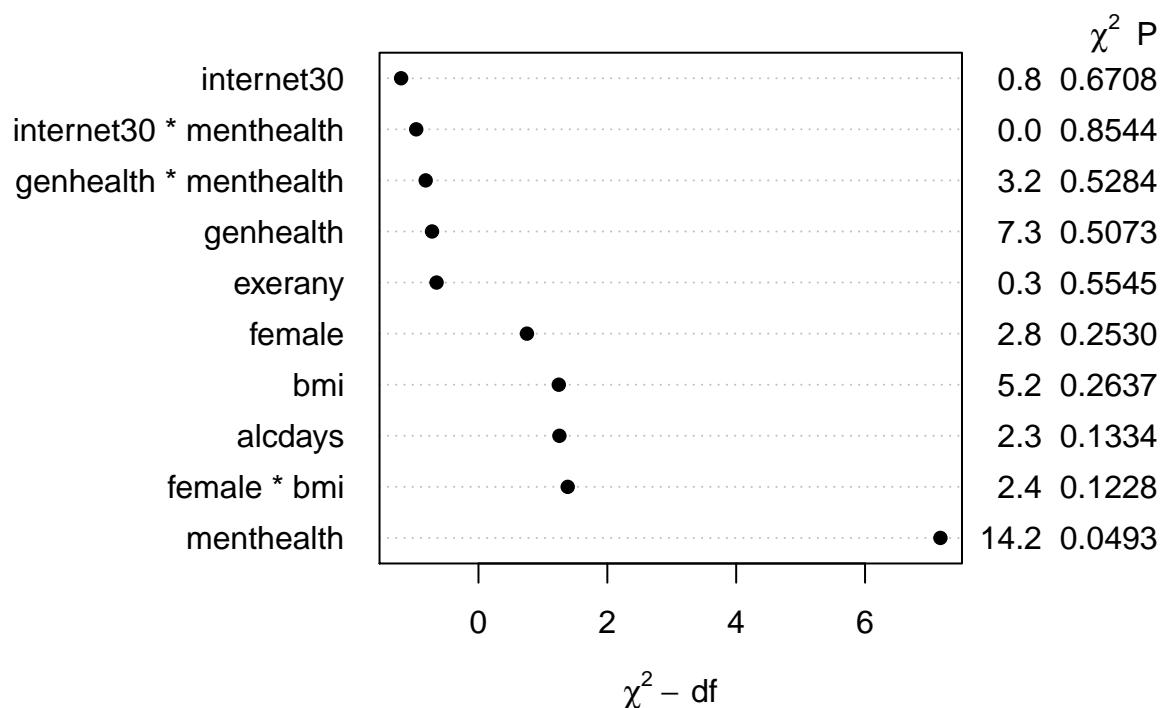
15.8.2 ANOVA for the model

Next, let's look at the ANOVA comparisons for this model.

```
anova(mod.B2)
```

	Analysis of Variance		Response: sleephrs
Factor	d.f.	Partial SS	
menthealth (Factor+Higher Order Factors)	7	3.270460e+01	
All Interactions	5	7.342058e+00	
Nonlinear	1	3.559092e+00	
genhealth (Factor+Higher Order Factors)	8	1.680021e+01	
All Interactions	4	7.340686e+00	
genhealth * menthealth (Factor+Higher Order Factors)	4	7.340686e+00	
internet30 (Factor+Higher Order Factors)	2	1.844060e+00	
All Interactions	1	7.778417e-02	
internet30 * menthealth (Factor+Higher Order Factors)	1	7.778417e-02	
bmi (Factor+Higher Order Factors)	4	1.210885e+01	
All Interactions	1	5.505060e+00	
Nonlinear	2	4.530786e+00	
female (Factor+Higher Order Factors)	2	6.352010e+00	
All Interactions	1	5.505060e+00	
female * bmi (Factor+Higher Order Factors)	1	5.505060e+00	
alcdays	1	5.207146e+00	
exerany	1	8.067239e-01	
TOTAL NONLINEAR	3	8.358126e+00	
TOTAL INTERACTION	6	1.310220e+01	
TOTAL NONLINEAR + INTERACTION	9	2.178490e+01	
REGRESSION	19	7.319542e+01	
ERROR	1008	2.326489e+03	
MS	F	P	
4.67208631	2.02	0.0493	
1.46841162	0.64	0.6721	
3.55909180	1.54	0.2146	
2.10002607	0.91	0.5073	
1.83517159	0.80	0.5284	
1.83517159	0.80	0.5284	
0.92203018	0.40	0.6708	
0.07778417	0.03	0.8544	
0.07778417	0.03	0.8544	
3.02721254	1.31	0.2637	
5.50505988	2.39	0.1228	
2.26539276	0.98	0.3751	
3.17600485	1.38	0.2530	
5.50505988	2.39	0.1228	
5.50505988	2.39	0.1228	
5.20714639	2.26	0.1334	
0.80672393	0.35	0.5545	
2.78604205	1.21	0.3059	
2.18370069	0.95	0.4609	
2.42054479	1.05	0.3988	
3.85239069	1.67	0.0357	
2.30802520			

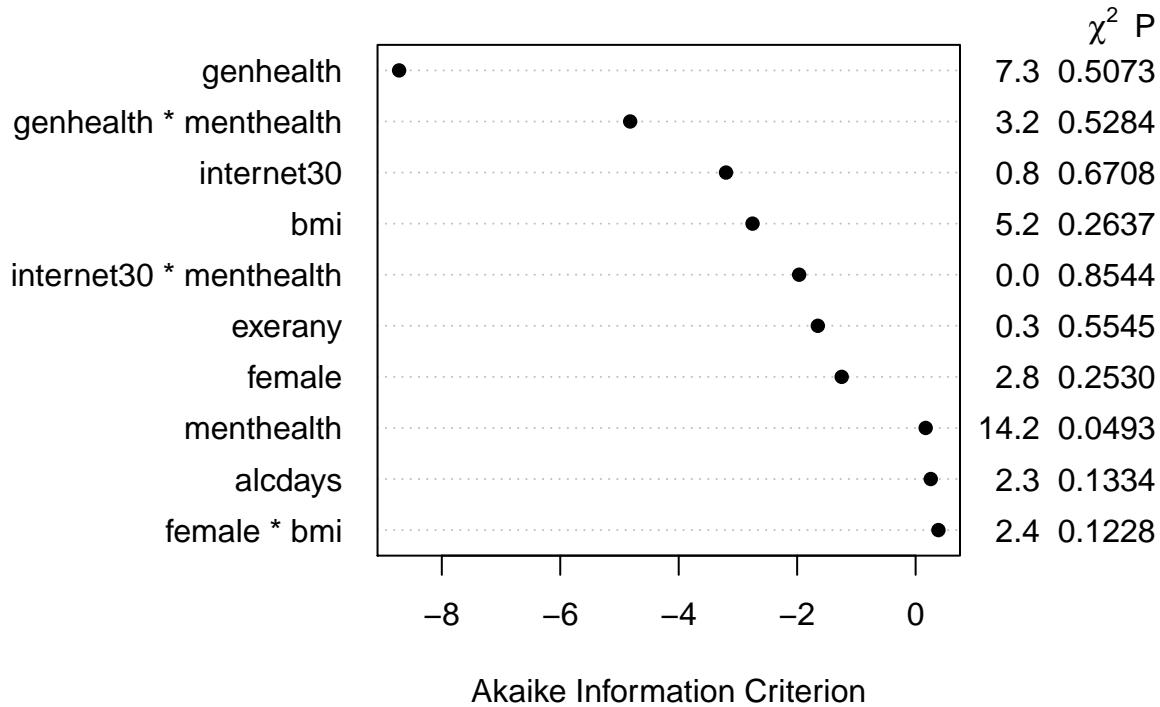
```
plot(anova(mod.B2))
```



Again, only `menthealth` (and that just barely) is carrying statistically significant predictive value.

Here is the AIC plot.

```
plot(anova(mod.B2), what="aic")
```



15.8.3 Summarizing Effect Size

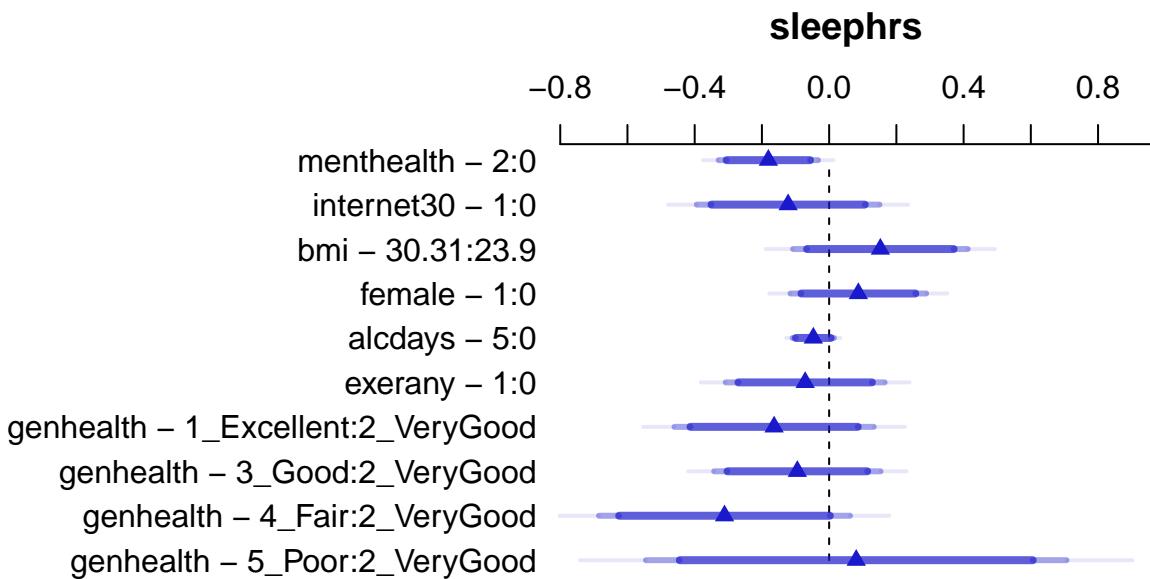
How big are the effects we see?

```
summary(mod.B2)
```

Effects	Response : sleephrs					
Factor	Low	High	Diff.	Effect	S.E.	
menthealth	0.0	2.00	2.00	-0.180800	0.075371	
internet30	0.0	1.00	1.00	-0.122060	0.138610	
bmi	23.9	30.31	6.41	0.152360	0.132460	
female	0.0	1.00	1.00	0.086907	0.102950	
alcdays	0.0	5.00	5.00	-0.047077	0.031342	
exerany	0.0	1.00	1.00	-0.071311	0.120620	
genhealth - 1_Excellent:2_VeryGood	2.0	1.00	NA	-0.163750	0.151290	
genhealth - 3_Good:2_VeryGood	2.0	3.00	NA	-0.094635	0.126260	
genhealth - 4_Fair:2_VeryGood	2.0	4.00	NA	-0.311280	0.190520	
genhealth - 5_Poor:2_VeryGood	2.0	5.00	NA	0.080564	0.319140	
Lower 0.95		Upper 0.95				
-0.32870		-0.032900				
-0.39406		0.149940				
-0.10757		0.412290				
-0.11511		0.288920				
-0.10858		0.014427				
-0.30800		0.165380				

```
-0.46062  0.133120
-0.34239  0.153120
-0.68515  0.062577
-0.54569  0.706820
```

```
Adjusted to: menthealth=0 genhealth=2_VeryGood internet30=0 bmi=27.09563 female=0
plot(summary(mod.B2))
```



```
ed to:menthealth=0 genhealth=2_VeryGood internet30=0 bmi=27.09563 female=0
```

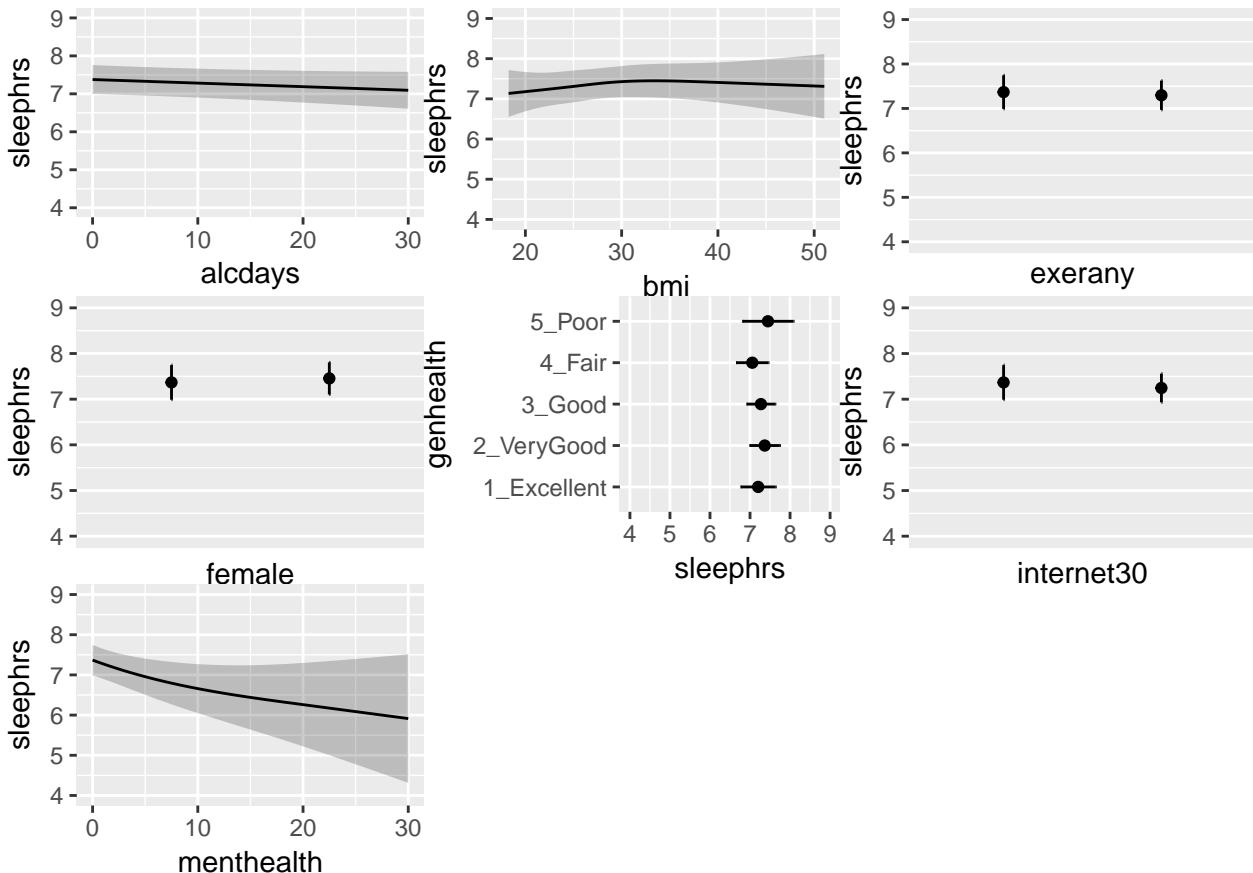
This output is easier to read as a result of using small *numeric* labels in `genh_n`, rather than the lengthy labels in `genhealth`. The results are, again, interpreted as differences in predicted `sleephrs`. For example,

- holding all other predictors constant, the effect of moving from `menthealth = 0` to `menthealth = 2` is a decline of 0.18 in predicted `sleephrs`, with 95% CI (-0.33, -0.03) hours.

15.8.4 Plotting the Model with ggplot and Predict

Again, consider a series of plots describing the model `mod.B2`.

```
ggplot(Predict(mod.B2))
```



Note the small kink in the `bmi` plot. To what do you attribute this?

15.9 Comparing Model B.2 to Model A.2 after simple imputation

We can refit the models with `glm` and then compare them with `anova`, `aic` and `bic` approaches, if we like.

```
mA2_lm <- lm(sleephrs ~ alcdays + bmi*female + exerany +
               genhealth + internet30*menthealth,
               data = smartcle2_imp1)

mB2_lm <- lm(sleephrs ~ rcs(menthealth, 5) + genhealth +
               genhealth %ia% menthealth + internet30 +
               internet30 %ia% menthealth + rcs(bmi, 4) +
               female + female %ia% bmi + alcdays +
               exerany,
               data = smartcle2_imp1)
```

15.9.1 Comparison by Analysis of Variance

```
anova(mA2_lm, mB2_lm)
```

Analysis of Variance Table

Model 1: sleephrs ~ alcdays + bmi * female + exerany + genhealth + internet30 *

```

menthealth
Model 2: sleephrs ~ rcs(menthealth, 5) + genhealth + genhealth %ia% menthealth +
    internet30 + internet30 %ia% menthealth + rcs(bmi, 4) + female +
    female %ia% bmi + alcdays + exerany
Res.Df   RSS Df Sum of Sq   F Pr(>F)
1   1015 2342.9
2   1008 2326.5  7   16.435 1.0172 0.4172

```

The additional terms in model B2 don't seem to improve the fit significantly.

15.9.2 Comparing AIC and BIC

```
glance(mA2_lm)
```

	r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik
1	0.02365341	0.0121104	1.519309	2.049154	0.01784591	13	-1882.094
	AIC	BIC	deviance	df.residual			
1	3792.188	3861.283	2342.924			1015	

```
glance(mB2_lm)
```

	r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik
1	0.0305021	0.01222783	1.519219	1.669129	0.03567068	20	-1878.476
	AIC	BIC	deviance	df.residual			
1	3798.952	3902.595	2326.489			1008	

Model `mA2_lm` shows lower AIC and BIC than does `mB2_lm`, but as we see in the R^2 values, they are both terrible models.

15.10 Dealing with Missing Data via Multiple Imputation

Next, we'll use the `aregImpute` function within the `Hmisc` package to predict all missing values for all of our variables, using additive regression bootstrapping and predictive mean matching. The steps for this work are as follows:

1. `aregImpute` draws a sample with replacement from the observations where the target variable is observed, not missing.
2. `aregImpute` then fits a flexible additive model to predict this target variable while finding the optimum transformation of it.
3. `aregImpute` then uses this fitted flexible model to predict the target variable in all of the original observations.
4. Finally, `aregImpute` imputes each missing value of the target variable with the observed value whose predicted transformed value is closest to the predicted transformed value of the missing value.

We'll start with the `smartcle2_imp0` data set, which contains only the subjects in the original `smartcle1` data where `sleephrs` is available, and which includes only the variables of interest to us, including both the factor (`genhealth`) and numeric (`genh_n`) versions of the `genhealth` data.

```
summary(smartcle2_imp0)
```

SEQNO	sleephrs	alcdays	bmi
Min. :2.016e+09	Min. : 1.000	Min. : 0.000	Min. :12.71
1st Qu.:2.016e+09	1st Qu.: 6.000	1st Qu.: 0.000	1st Qu.:23.70
Median :2.016e+09	Median : 7.000	Median : 1.000	Median :26.68
Mean :2.016e+09	Mean : 7.018	Mean : 4.672	Mean :27.86
3rd Qu.:2.016e+09	3rd Qu.: 8.000	3rd Qu.: 4.000	3rd Qu.:30.53

```

Max.    :2.016e+09   Max.    :20.000   Max.    :30.000   Max.    :66.06
                           NA's     :45      NA's     :82
exerany       female      genhealth    internet30
Min.    :0.0000   Min.    :0.0000   1_Excellent:172   Min.    :0.0000
1st Qu.:1.0000  1st Qu.:0.0000   2_VeryGood :349    1st Qu.:1.0000
Median  :1.0000  Median  :1.0000   3_Good     :341    Median  :1.0000
Mean    :0.7622  Mean    :0.6012   4_Fair     :120    Mean    :0.8112
3rd Qu.:1.0000  3rd Qu.:1.0000   5_Poor     :44     3rd Qu.:1.0000
Max.    :1.0000  Max.    :1.0000   NA's      : 2     Max.    :1.0000
NA's    :2          NA's      : 2     NA's      :6
menthealth      genh_n
Min.    : 0.000   Min.    :1.000
1st Qu.: 0.000   1st Qu.:2.000
Median  : 0.000   Median  :2.000
Mean    : 2.707   Mean    :2.527
3rd Qu.: 2.000   3rd Qu.:3.000
Max.    :30.000   Max.    :5.000
NA's    :11        NA's    :2

```

The `smartcle2_imp0` data set contains 1028 rows (subjects) and 10 columns (variables.)

15.10.1 Using `aregImpute` to fit a multiple imputation model

To set up `aregImpute` here, we'll need to specify:

- a suitable random seed with `set.seed` so we can replicate our work later
- a data set via the `datadist` stuff shown below
- the variables we want to include in the imputation process, which should include, at a minimum, any variables with missing values, and any variables we want to include in our outcome models
- `n.impute` = number of imputations, we'll run 20 here¹
- `nk` = number of knots to describe level of complexity, with our choice `nk = c(0, 3)` we'll fit both linear models and models with restricted cubic splines with 3 knots (this approach will wind up throwing some warnings here because some of our variables with missing values have only a few options so fitting splines is tough.)
- `tlinear` = `FALSE` allows the target variable for imputation to have a non-linear transformation when `nk` is 3 or more. Here, I'll use `tlinear = TRUE`, the default.
- `B` = 10 specifies 10 bootstrap samples will be used
- `pr` = `FALSE` tells the machine not to print out which iteration is running as it goes.
- `data` specifies the source of the variables

```

set.seed(432365)
dd <- datadist(smartcle2_imp0)
options(datadist = "dd")

imp_fit <- aregImpute(~ sleephrs + alcdays + bmi + exerany +
                      female + genh_n + internet30 +
                      menthealth,
                      nk = c(0, 3), tlinear = TRUE,
                      data = smartcle2_imp0, B = 10,
                      n.impute = 20, pr = FALSE)

```

¹ 100 is generally safe but time-consuming. In the old days, we used to say 5. A reasonable idea is to identify the fraction of missingness in your variable with the most missingness, and if that's 0.10, then you should run at least $100(0.10) = 10$ sets of imputations.

OK. Here is the imputation model. The summary here isn't especially critical. We want to see what was run, but to see what the results look like, we'll need a plot, to come.

```
imp_fit
```

Multiple Imputation using Bootstrap and PMM

```
aregImpute(formula = ~sleephrs + alcdays + bmi + exerany + female +
  genh_n + internet30 + menthealth, data = smartcle2_imp0,
  n.impute = 20, nk = c(0, 3), tlinear = TRUE, pr = FALSE,
  B = 10)
```

n: 1028 p: 8 Imputations: 20 nk: 0

Number of NAs:

	sleephrs	alcdays	bmi	exerany	female	genh_n
	0	45	82	2	0	2
internet30	6	11				
menthealth						

	type	d.f.
sleephrs	s	1
alcdays	s	1
bmi	s	1
exerany	l	1
female	l	1
genh_n	s	1
internet30	l	1
menthealth	s	1

Transformation of Target Variables Forced to be Linear

R-squares for Predicting Non-Missing Values for Each Variable
Using Last Imputations of Predictors

alcdays	bmi	exerany	genh_n	internet30	menthealth
0.071	0.094	0.166	0.212	0.151	0.083

Resampling results for determining the complexity of imputation models

Variable being imputed: alcdays

	nk=0	nk=3
Bootstrap bias-corrected R^2	0.0687	0.0612
10-fold cross-validated R^2	0.0706	0.0678
Bootstrap bias-corrected mean error	5.3804	5.2618
10-fold cross-validated mean error	4.9122	4.9570
Bootstrap bias-corrected median error	3.7094	3.7051
10-fold cross-validated median error	1.6790	1.7335

Variable being imputed: bmi

	nk=0	nk=3
Bootstrap bias-corrected R^2	0.0904	0.100
10-fold cross-validated R^2	0.1061	0.103
Bootstrap bias-corrected mean error	4.3705	4.400
10-fold cross-validated mean error	27.8053	27.767

```
Bootstrap bias-corrected median |error| 3.2927 3.409
10-fold cross-validated median |error| 26.8306 26.687
```

Variable being imputed: exerany

	nk=0	nk=3
Bootstrap bias-corrected R ²	0.135	0.116
10-fold cross-validated R ²	0.135	0.125
Bootstrap bias-corrected mean error	0.317	0.317
10-fold cross-validated mean error	0.930	0.899
Bootstrap bias-corrected median error	0.248	0.231
10-fold cross-validated median error	0.740	0.671

Variable being imputed: genh_n

	nk=0	nk=3
Bootstrap bias-corrected R ²	0.198	0.236
10-fold cross-validated R ²	0.200	0.225
Bootstrap bias-corrected mean error	0.744	0.731
10-fold cross-validated mean error	2.538	2.531
Bootstrap bias-corrected median error	0.646	0.650
10-fold cross-validated median error	2.532	2.524

Variable being imputed: internet30

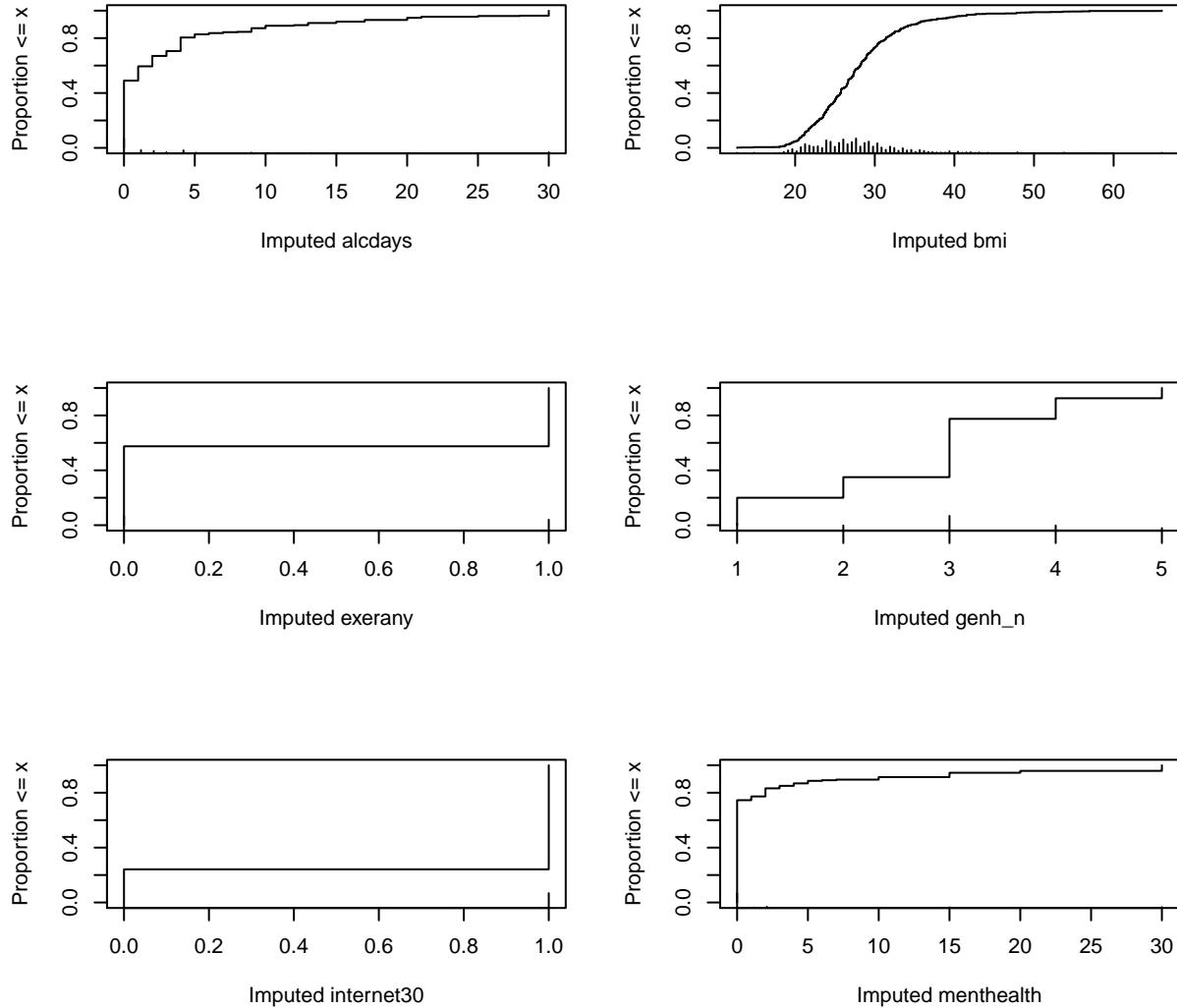
	nk=0	nk=3
Bootstrap bias-corrected R ²	0.101	0.125
10-fold cross-validated R ²	0.109	0.121
Bootstrap bias-corrected mean error	0.279	0.272
10-fold cross-validated mean error	0.992	0.983
Bootstrap bias-corrected median error	0.190	0.190
10-fold cross-validated median error	0.817	0.819

Variable being imputed: menthealth

	nk=0	nk=3
Bootstrap bias-corrected R ²	0.0573	0.102
10-fold cross-validated R ²	0.0969	0.116
Bootstrap bias-corrected mean error	3.8132	3.712
10-fold cross-validated mean error	3.1552	3.112
Bootstrap bias-corrected median error	2.3749	1.979
10-fold cross-validated median error	0.9384	0.783

OK, let's plot these imputed values. Note that we had six predictors with missing values in our data set, and so if we plot each of those, we'll wind up with six plots. I'll arrange them in a grid with three rows and two columns.

```
par(mfrow = c(3,2))
plot(imp_fit)
```



```
par(mfrow = c(1,1))
```

From these cumulative distribution functions, we can see that, for example,

- we imputed `bmi` values mostly between 20 and 35, with a few values below 20 or above 40.
- most of our imputed `alcdays` were between 0 and 5
- we imputed 1 for `internet30` for about 70% of the subjects, and 0 for the other 30%.

This predictive mean matching method never imputes a value for a variable that does not already exist in the data.

15.11 Combining the Imputation and Outcome Models

So, now we have an imputation model, called `imp_fit`. and two outcome models: `mod.A` and `mod.B`. What do we do with them?

15.11.1 Model A with Multiple Imputation

To build the `mA_imp` multiple imputation fit for model `mod.A`, we use the `fit.mult.impute` command, and specify the model, the fitter (here, `lrm`), the imputation model (`xtrans = imp_fit`) and the data set prior to imputation (`smartcle2_imp0`).

```
mA_imp <- fit.mult.impute(sleephrs ~ alcdays + bmi*female +
                           exerany + catg(genh_n) +
                           internet30*menthealth,
                           fitter = ols, xtrans = imp_fit,
                           data = smartcle2_imp0,
                           x = TRUE, y = TRUE)
```

Variance Inflation Factors Due to Imputation:

Intercept	alcdays	bmi
1.22	1.02	1.29
female	exerany	genh_n=2
1.21	1.01	1.00
genh_n=3	genh_n=4	genh_n=5
1.01	1.01	1.00
internet30	menthealth	bmi * female
1.01	1.01	1.22
internet30 * menthealth		
1.01		

Rate of Missing Information:

Intercept	alcdays	bmi
0.18	0.02	0.22
female	exerany	genh_n=2
0.18	0.01	0.00
genh_n=3	genh_n=4	genh_n=5
0.01	0.01	0.00
internet30	menthealth	bmi * female
0.01	0.01	0.18
internet30 * menthealth		
0.01		

d.f. for t-distribution for Tests of Single Coefficients:

Intercept	alcdays	bmi
593.81	56125.67	384.31
female	exerany	genh_n=2
618.43	204840.97	2743445.99
genh_n=3	genh_n=4	genh_n=5
197711.45	230304.96	812145.77
internet30	menthealth	bmi * female
573166.43	100933.56	582.27
internet30 * menthealth		
149596.38		

The following fit components were averaged over the 20 model fits:

```
fitted.values stats linear.predictors
```

OK. Let's get the familiar description of an `ols` model, after this multiple imputation.

```
mA_imp
```

Linear Regression Model

```
fit.mult.impute(formula = sleephrs ~ alcdays + bmi * female +
  exerany + catg(genh_n) + internet30 * menthealth, fitter = ols,
  xtrans = imp_fit, data = smartcle2_imp0, x = TRUE, y = TRUE)
```

	Model Likelihood	Discrimination	
	Ratio Test	Indexes	
Obs	1028	LR chi2 25.38	R2 0.024
sigma1	5.187	d.f. 12	R2 adj 0.013
d.f.	1015	Pr(> chi2) 0.0131	g 0.249

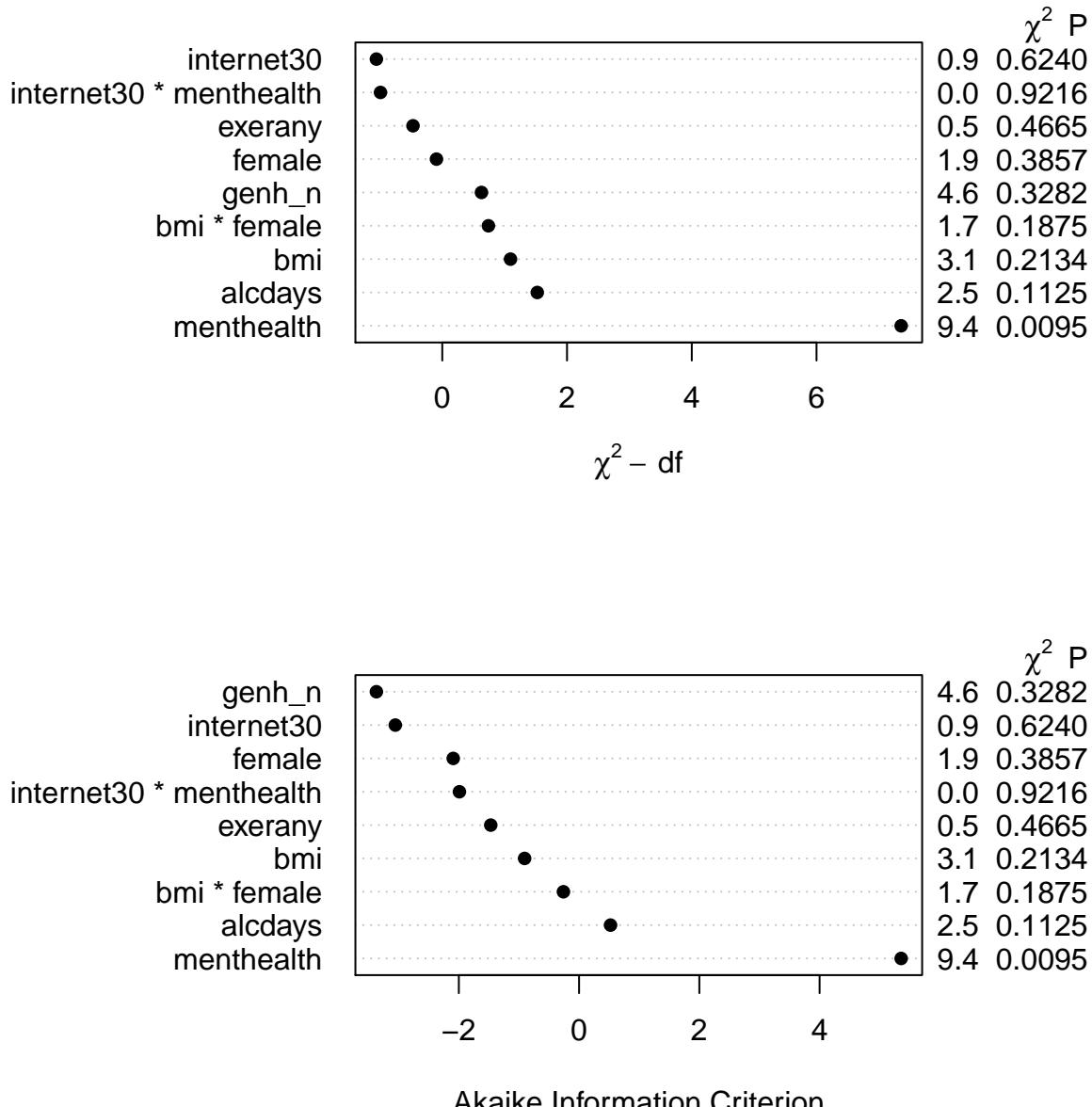
Residuals

Min	1Q	Median	3Q	Max
-6.1580	-0.9603	-0.0241	0.8660	13.8147

	Coef	S.E.	t	Pr(> t)
Intercept	7.0186	0.4653	15.09	<0.0001
alcdays	-0.0098	0.0062	-1.59	0.1125
bmi	0.0065	0.0150	0.43	0.6642
female	0.6880	0.5007	1.37	0.1697
exerany	-0.0879	0.1207	-0.73	0.4665
genh_n=2	0.1274	0.1431	0.89	0.3735
genh_n=3	0.0966	0.1487	0.65	0.5160
genh_n=4	-0.2022	0.1950	-1.04	0.3000
genh_n=5	0.1381	0.2727	0.51	0.6126
internet30	-0.1200	0.1375	-0.87	0.3829
menthealth	-0.0212	0.0162	-1.31	0.1915
bmi * female	-0.0231	0.0175	-1.32	0.1875
internet30 * menthealth	-0.0018	0.0178	-0.10	0.9216

We can obtain an ANOVA plot and an AIC plot to look at the predictors:

```
par(mfrow = c(2,1))
plot(anova(mA_imp))
plot(anova(mA_imp), what="aic")
```



```
par(mfrow = c(1,1))
```

Here's the summary of effect sizes.

```
summary(mA_imp)
```

Factor	Effects			Response : sleephrs					
	Low	High	Diff.	Effect	S.E.	Lower	0.95	Upper	
alcdays	0.0	4.00	4.00	-0.039305	0.024745	-0.087862	0.0092510		
bmi	23.7	30.53	6.83	0.044453	0.102380	-0.156450	0.2453600		
female	0.0	1.00	1.00	0.072470	0.103040	-0.129720	0.2746600		

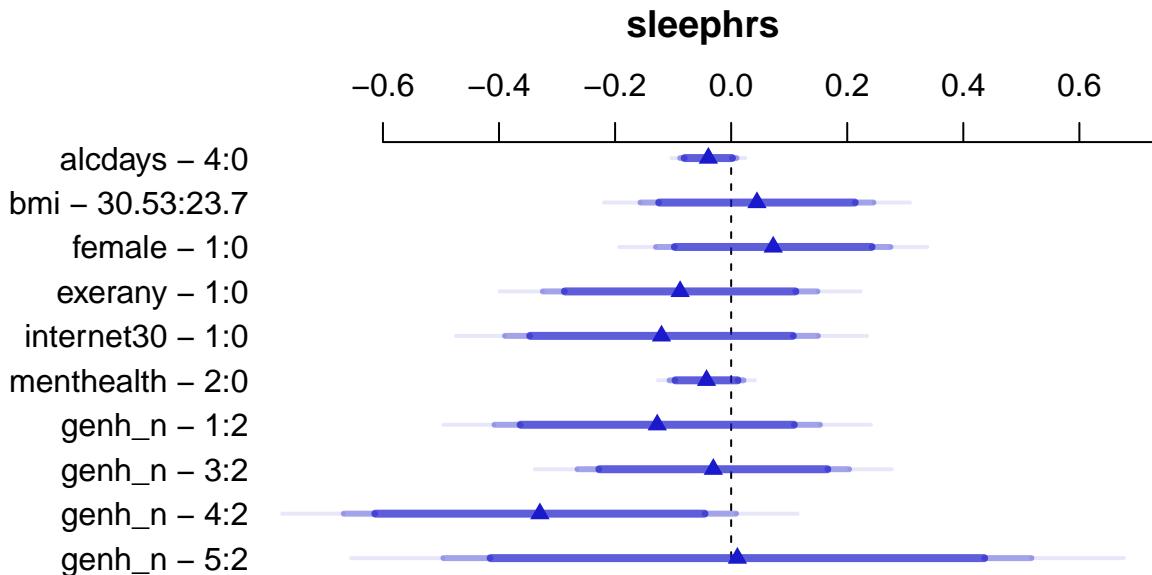
```

exerany      0.0  1.00 1.00  -0.087925 0.120690 -0.324760  0.1489100
internet30   0.0  1.00 1.00  -0.120040 0.137520 -0.389900  0.1498100
menthealth    0.0  2.00 2.00  -0.042465 0.032493 -0.106230  0.0212970
genh_n - 1:2  2.0  1.00   NA  -0.127370 0.143050 -0.408080  0.1533300
genh_n - 3:2  2.0  3.00   NA  -0.030744 0.119400 -0.265030  0.2035500
genh_n - 4:2  2.0  4.00   NA  -0.329540 0.172410 -0.667870  0.0087869
genh_n - 5:2  2.0  5.00   NA   0.010733 0.258480 -0.496490  0.5179500

```

Adjusted to: bmi=26.68 female=0 internet30=0 menthealth=0

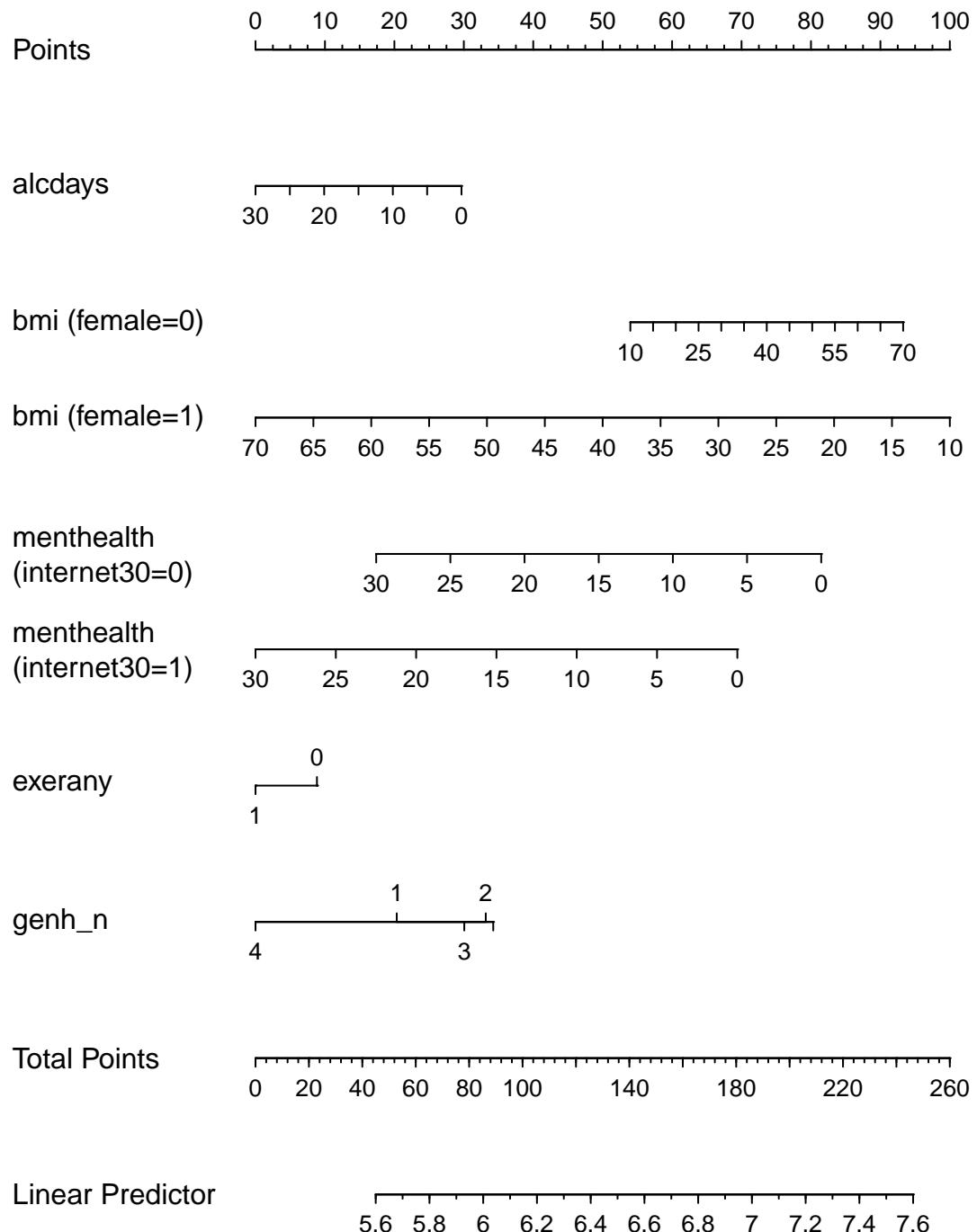
```
plot(summary(mA_imp))
```



Adjusted to:bmi=26.68 female=0 internet30=0 menthealth=0

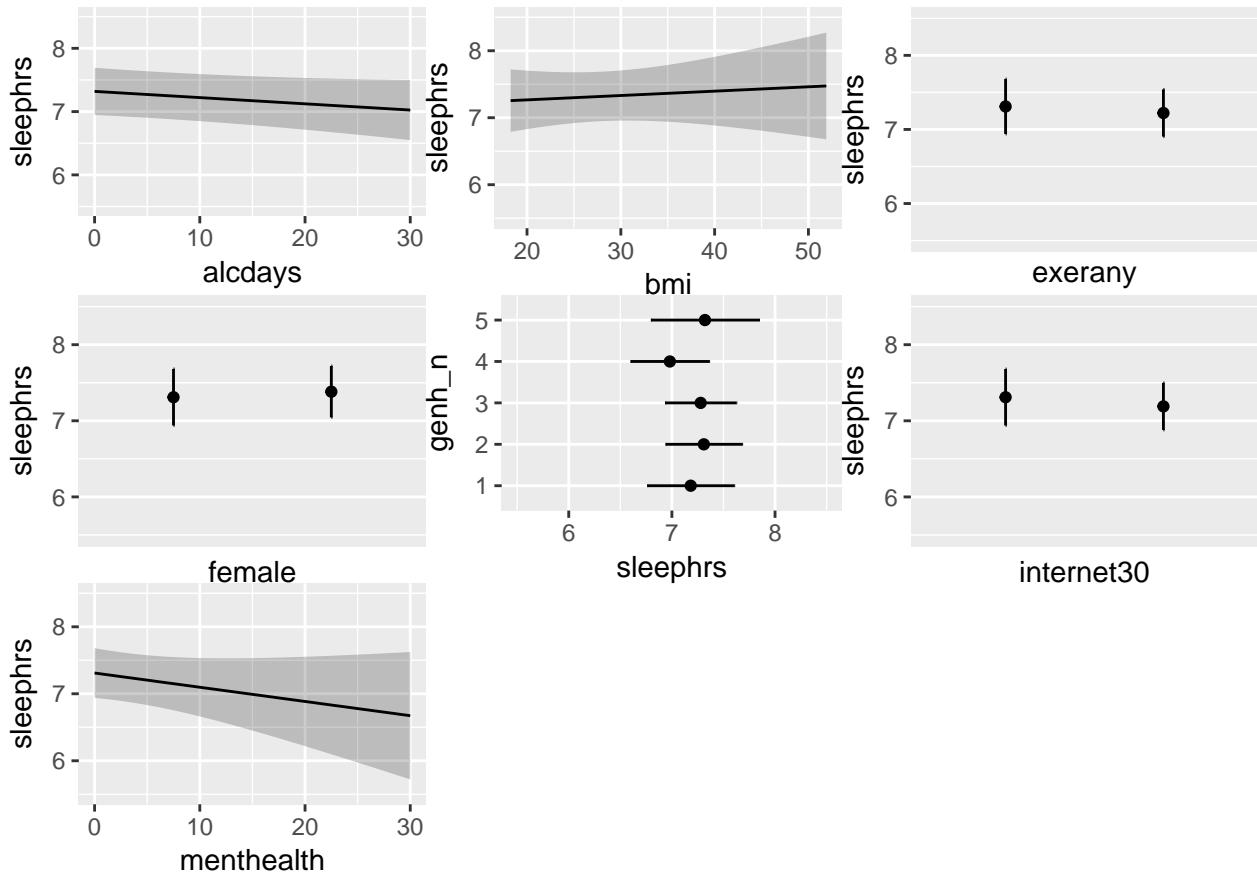
And here is the nomogram.

```
plot(nomogram(mA_imp))
```



Here are the descriptive model plots:

```
ggplot(Predict(mA_imp))
```



We can still do things like validate the summary statistics, too.

```
validate(mA_imp)
```

	index.orig	training	test	optimism	index.corrected	n
R-square	0.0249	0.0434	0.0041	0.0394	-0.0145	40
MSE	2.2762	2.2940	2.3249	-0.0309	2.3071	40
g	0.2458	0.3247	0.1889	0.1358	0.1100	40
Intercept	0.0000	0.0000	2.8426	-2.8426	2.8426	40
Slope	1.0000	1.0000	0.5940	0.4060	0.5940	40

The same approach can be used to build a `mB_imp` multiple imputation fit for `mod.B`, using the `fit.mult.impute` command, and specifying the model, the fitter (here, `ols`), the imputation model (`xtrans = imp_fit`) and the data set prior to imputation (`smartcle2_imp0`). We'll skip it for now. The model remains terrible.

Chapter 16

Colorectal Cancer Screening and Some Special Cases

In this Chapter, we discuss two issues as yet unraised regarding regression on a binary outcome.

1. What do we do if our binary outcome is not available for each subject individually, but instead aggregated?
2. What is probit regression, and how can we use it as an alternative to logistic regression on a binary outcome?

16.1 Logistic Regression for Aggregated Data

16.1.1 Colorectal Cancer Screening Data

The `screening.csv` data (imported into the R tibble `colscre`) are simulated. They mirror a subset of the actual results from the Better Health Partnership's pilot study of colorectal cancer screening in primary care clinics in Northeast Ohio, but the numbers have been fuzzed slightly, and the clinics have been de-identified and moved around from system to system.

Available to us are the following variables:

Variable	Description
<code>location</code>	clinic code
<code>subjects</code>	number of subjects reported by clinic
<code>screen_rate</code>	proportion of <code>subjects</code> who were screened
<code>screened</code>	number of <code>subjects</code> who were screened
<code>notscreened</code>	number of <code>subjects</code> not screened
<code>meanage</code>	mean age of clinic's subjects, years
<code>female</code>	% of clinic's subjects who are female
<code>pct_lowins</code>	% of clinic's subjects who have Medicaid or are uninsured
<code>system</code>	system code

```
colscre %>% skim()
```

```
Skim summary statistics
n obs: 26
n variables: 9
```

```

Variable type: factor
  variable missing complete n n_unique                      top_counts
location        0      26 26      26          A: 1, B: 1, C: 1, D: 1
  system        0      26 26      4 Sys: 7, Sys: 7, Sys: 6, Sys: 6
ordered
  FALSE
  FALSE

Variable type: integer
  variable missing complete n      mean      sd    p0     p25 median
notscreened      0      26 26 663.23 271.17 231 508.75 611
  screened       0      26 26 2584.04 1765.11 572 1395.25 2169.5
  subjects       0      26 26 3247.27 1945.83 803 1914.75 2765.5
  p75   p100
  791    1356
  2716    6947
  3607.75 7677

Variable type: numeric
  variable missing complete n      mean      sd    p0     p25 median    p75
  female         0      26 26 58.72  6.29  46.2  55.42 60.05 62.62
  meanage        0      26 26 60.58  1.93  58    58.82 60.5  61.98
  pct_lowins     0      26 26 24.47 19.13  0.3   4.8   23.95 44.03
  screen_rate    0      26 26 0.77  0.072  0.64  0.72  0.76  0.81
p100
70.3
65.9
51.3
0.9

```

16.1.2 Fitting a Logistic Regression Model to Proportion Data

Here, we have a binary outcome (was the subject screened or not?) but we have aggregated results. We can use the counts of the numbers of subjects at each clinic (in `subjects`) and the proportion who were screened (in `screen_rate`) to fit a logistic regression model, as follows:

```

m_screen1 <- glm(screen_rate ~ meanage + female +
                     pct_lowins + system, family = binomial,
                     weights = subjects, data = colscr)

summary(m_screen1)

```

Call:
`glm(formula = screen_rate ~ meanage + female + pct_lowins + system,`
`family = binomial, data = colscr, weights = subjects)`

Deviance Residuals:

Min	1Q	Median	3Q	Max
-9.017	-3.705	-2.000	1.380	11.500

Coefficients:

Estimate	Std. Error	z value	Pr(> z)
----------	------------	---------	----------

```
(Intercept) -1.3270393 0.5530782 -2.399  0.0164 *
meanage      0.0679866 0.0089754  7.575 3.60e-14 ***
female       -0.0193142 0.0015831 -12.200 < 2e-16 ***
pct_lowins   -0.0134547 0.0008585 -15.672 < 2e-16 ***
systemSys_2  -0.1382189 0.0246591 -5.605 2.08e-08 ***
systemSys_3  -0.0400170 0.0254505 -1.572  0.1159
systemSys_4   0.0229273 0.0294207  0.779  0.4358
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2825.28  on 25  degrees of freedom
Residual deviance: 816.39  on 19  degrees of freedom
AIC: 1037.9

Number of Fisher Scoring iterations: 4
```

16.1.3 Fitting a Logistic Regression Model to Counts of Successes and Failures

```
m_screen2 <- glm(cbind(screened, notscreened) ~
                  meanage + female + pct_lowins + system,
                  family = binomial, data = colscr)
summary(m_screen2)
```

Call:
`glm(formula = cbind(screened, notscreened) ~ meanage + female +
 pct_lowins + system, family = binomial, data = colscr)`

Deviance Residuals:

Min	1Q	Median	3Q	Max
-9.017	-3.705	-2.000	1.380	11.500

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-1.3270392	0.5530782	-2.399	0.0164 *
meanage	0.0679866	0.0089754	7.575	3.60e-14 ***
female	-0.0193142	0.0015831	-12.200	< 2e-16 ***
pct_lowins	-0.0134547	0.0008585	-15.672	< 2e-16 ***
systemSys_2	-0.1382189	0.0246591	-5.605	2.08e-08 ***
systemSys_3	-0.0400170	0.0254505	-1.572	0.1159
systemSys_4	0.0229273	0.0294207	0.779	0.4358

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

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2825.28 on 25 degrees of freedom
Residual deviance: 816.39 on 19 degrees of freedom
AIC: 1037.9

Number of Fisher Scoring iterations: 4

16.1.4 How does one address this problem in rms?

We can use `Glm`. As an example to mirror `m_screen1`, we have the following...

```
d <- datadist(colsr)
options(datadist = "d")

mod_screen_1 <-  glm(screen_rate ~ meanage + female +
                      pct_lowins + system,
                      family = binomial, weights = subjects,
                      data = colsr, x = T, y = T)

mod_screen_1
```

General Linear Model

```
Glm(formula = screen_rate ~ meanage + female + pct_lowins + system,
     family = binomial, data = colsr, weights = subjects, x = T,
     y = T)
```

Model Likelihood Ratio Test					
Obs	26	LR chi2	2008.90		
Residual d.f.	19	d.f.	6		
g	0.4614539		Pr(> chi2)	<0.0001	
		Coef	S.E.	Wald Z	Pr(> Z)
Intercept	-1.3270	0.5531	-2.40	0.0164	
meanage	0.0680	0.0090	7.57	<0.0001	
female	-0.0193	0.0016	-12.20	<0.0001	
pct_lowins	-0.0135	0.0009	-15.67	<0.0001	
system=Sys_2	-0.1382	0.0247	-5.61	<0.0001	
system=Sys_3	-0.0400	0.0255	-1.57	0.1159	
system=Sys_4	0.0229	0.0294	0.78	0.4358	

16.2 Probit Regression

16.2.1 Colorectal Cancer Screening Data on Individuals

The data in the `cols2` data frame describe (disguised) data on the status of 172 adults who were eligible for colon cancer screening, with the following information included:

Variable	Description
<code>subject</code>	subject ID code
<code>age</code>	subject's age (years)
<code>race</code>	subject's race (White/Black/Other)
<code>hispanic</code>	subject of Hispanic ethnicity (1 = yes / 0 = no)
<code>insurance</code>	Commercial, Medicaid, Medicare, Uninsured
<code>bmi</code>	body mass index at most recent visit
<code>sbp</code>	systolic blood pressure at most recent visit
<code>up_to_date</code>	meets colon cancer screening standards

The goal is to use the other variables (besides subject ID) to predict whether or not a subject is up to date.

16.2.2 A logistic regression model

Here is a logistic regression model.

```
colscr2 %>% summary()
```

subject	age	race	hispanic
Min. :101.0	Min. :51.00	Black:118	Min. :0.00000
1st Qu.:143.8	1st Qu.:54.00	Other: 9	1st Qu.:0.00000
Median :186.5	Median :57.00	White: 45	Median :0.00000
Mean :186.5	Mean :57.80		Mean :0.06395
3rd Qu.:229.2	3rd Qu.:61.25		3rd Qu.:0.00000
Max. :272.0	Max. :69.00		Max. :1.00000
insurance	bmi	sbp	up_to_date
Commercial:32	Min. :17.20	Min. : 89.0	Min. :0.0000
Medicaid :81	1st Qu.:25.48	1st Qu.:118.0	1st Qu.:0.0000
Medicare :46	Median :30.05	Median :127.0	Median :1.0000
Uninsured :13	Mean :31.24	Mean :128.9	Mean :0.6047
	3rd Qu.:36.03	3rd Qu.:138.0	3rd Qu.:1.0000
	Max. :55.41	Max. :198.0	Max. :1.0000

```
m_scr2_logistic <- glm(up_to_date ~ age + race + hispanic +
                         insurance + bmi + sbp,
                         family = binomial, data = colscr2)
```

```
summary(m_scr2_logistic)
```

Call:

```
glm(formula = up_to_date ~ age + race + hispanic + insurance +
     bmi + sbp, family = binomial, data = colscr2)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.8604	-1.1540	0.6933	0.9564	1.6108

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	2.7040470	2.7418625	0.986	0.3240
age	0.0204901	0.0396920	0.516	0.6057
raceOther	-1.9722351	1.0023237	-1.968	0.0491 *
raceWhite	-0.3210458	0.4001744	-0.802	0.4224
hispanic	0.0005855	0.7953482	0.001	0.9994
insuranceMedicaid	-1.0151860	0.4945169	-2.053	0.0401 *
insuranceMedicare	-0.5216006	0.5629935	-0.926	0.3542
insuranceUninsured	0.1099966	0.7906196	0.139	0.8893
bmi	0.0155894	0.0213547	0.730	0.4654
sbp	-0.0241777	0.0099138	-2.439	0.0147 *

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

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 230.85 on 171 degrees of freedom
Residual deviance: 210.55 on 162 degrees of freedom
AIC: 230.55
```

Number of Fisher Scoring iterations: 4

```
confint(m_scr2_logistic)
```

Waiting for profiling to be done...

	2.5 %	97.5 %
(Intercept)	-2.64274441	8.157738367
age	-0.05703934	0.099298904
raceOther	-4.24604744	-0.175953229
raceWhite	-1.10800168	0.469396096
hispanic	-1.53691431	1.688738936
insuranceMedicaid	-2.03640881	-0.079142467
insuranceMedicare	-1.66246932	0.564088260
insuranceUninsured	-1.40110091	1.759391281
bmi	-0.02568426	0.058563761
sbp	-0.04436143	-0.005282686

In this model, there appears to be some link between `sbp` and screening, as well as, perhaps, some statistically significant differences between some race groups and some insurance groups. We won't look at this much further for now, though. Instead, we'll simply describe predictions for two subjects, Harry and Sally.

16.2.3 Predicting status for Harry and Sally

- Harry is age 65, White, non-Hispanic, with Medicare insurance, a BMI of 28 and SBP of 135.
- Sally is age 60, Black, Hispanic, with Medicaid insurance, a BMI of 22 and SBP of 148.

```
newdat_s2 <- data_frame(subject = c("Harry", "Sally"),
                           age = c(65, 60),
                           race = c("White", "Black"),
                           hispanic = c(0, 1),
                           insurance = c("Medicare", "Medicaid"),
                           bmi = c(28, 22),
                           sbp = c(135, 148))

predict(m_scr2_logistic, newdata = newdat_s2,
       type = "response")
```

```
1      2
0.5904364 0.4215335
```

The prediction for Harry is 0.59, and for Sally, 0.42, by this logistic regression model.

16.2.4 A probit regression model

Now, consider a probit regression, fit by changing the default link for the `binomial` family as follows:

```
m_scr2_probit <- glm(up_to_date ~ age + race + hispanic +
                        insurance + bmi + sbp,
                        family = binomial(link = "probit"),
                        data = colscr2)
```

```
summary(m_scr2_probit)

Call:
glm(formula = up_to_date ~ age + race + hispanic + insurance +
    bmi + sbp, family = binomial(link = "probit"), data = colscr2)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-1.8608 -1.1561  0.6933  0.9607  1.6073 

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 1.584604  1.658489  0.955   0.3394    
age          0.013461  0.024107  0.558   0.5766    
raceOther   -1.238445  0.587093 -2.109   0.0349 *  
raceWhite   -0.199260  0.243505 -0.818   0.4132    
hispanic    0.029483  0.484819  0.061   0.9515    
insuranceMedicaid -0.619277  0.293205 -2.112   0.0347 *  
insuranceMedicare -0.322881  0.333549 -0.968   0.3330    
insuranceUninsured 0.052776  0.463798  0.114   0.9094    
bmi          0.009652  0.012887  0.749   0.4539    
sbp         -0.014696  0.005944 -2.472   0.0134 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 230.85 on 171 degrees of freedom
Residual deviance: 210.49 on 162 degrees of freedom
AIC: 230.49
```

Number of Fisher Scoring iterations: 4

```
confint(m_scr2_probit)
```

Waiting for profiling to be done...

	2.5 %	97.5 %
(Intercept)	-1.70739455	4.894713408
age	-0.03400934	0.061437784
raceOther	-2.53819772	-0.119812915
raceWhite	-0.67910311	0.282566349
hispanic	-0.92476109	1.011089937
insuranceMedicaid	-1.21212866	-0.049749873
insuranceMedicare	-0.99315267	0.329851215
insuranceUninsured	-0.83335121	0.966504724
bmi	-0.01577917	0.035632196
sbp	-0.02659318	-0.003148488

16.2.5 Interpreting the Probit Model's Coefficients

It is possible to use any number of link functions to ensure that predicted values in a generalized linear model fall between 0 and 1. The probit regression model, for instance, uses the inverse of the cumulative

distribution function of the Normal model as its link function. Let's look more closely at the coefficients of the probit model we just fit.

```
m_scr2_probit$coef
```

	age	raceOther
(Intercept)	1.584603569	-1.238445198
raceWhite	0.013461338	hispanic insuranceMedicaid
-0.199260184	0.029483051	-0.619276718
insuranceMedicare	insuranceUninsured	bmi
-0.322880519	0.052775722	0.009652339
sbp		
-0.014695526		

The probit regression coefficients give the change in the z-score of the outcome of interest (here, `up_to_date`) for a one-unit change in the target predictor, holding all other predictors constant.

- So, for a one-year increase in age, holding all other predictors constant, the z-score for `up_to_date` increases by 0.013
- And for a Medicaid subject as compared to a Commercial subject of the same age, race, ethnicity, bmi and sbp, the z-score for the Medicaid subject is predicted to be -0.619 lower, according to this model.

16.2.6 What about Harry and Sally?

Do the predictions for Harry and Sally change much with this probit model, as compared to the logistic regression?

```
predict(m_scr2_probit, newdata = newdat_s2, type = "response")
```

```
1          2
0.5885511 0.4364027
```

Chapter 17

Cleaning the BRFSS SMART Data

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

In this work, we will focus on data from the 2016 SMART, and in particular on data from the 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 all of the cleaning I have done (so far) on the BRFSS SMART data, and break it out into national, statewide, and local samples.

The data files produced by this chapter include:

- `smart_ohio.csv` which includes data on 57 variables for 6014 subjects in five MMSAs in Ohio.
- `smartcle.csv` which includes data on those same 57 variables for the 1036 subjects in the Cleveland-Elyria-Lorain OH MMSA.
- `smartcle1.csv` which includes a subset of 11 variables for those same 1036 subjects.

SMART and BRFSS

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

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

17.1 Key resources

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

17.2 Ingesting The Raw Data

```
library(skimr)
library(broom); library(modelr)
library(haven); library(tidyverse)
```

17.3 The National 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("data/MMSA2016.xpt") %>%tbl_df
dim(smart.raw)
```

This gives the nationwide data, which has 249,011 rows and 146 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, called `smartorig.csv`, which I developed by

- importing the MMSA2016.xpt file
- filtering away all observations except those from the state of Ohio
- saving it

So, for purposes of these notes, our complete data set is actually coming from `smartorg.csv` and consists only of the 6,014 observations from Ohio.

```
smart.raw <- read_csv("data/smartorig.csv") %>%tbl_df
```

Parsed with column specification:

```
cols(
  .default = col_integer(),
  HIVTSTD3 = col_double(),
  `_BMI5` = col_double(),
  `_DRNKWEK` = col_double(),
  `_MMSAWT` = col_double(),
  SEQNO = col_double(),
  MMSANAME = col_character()
)
```

See `spec(...)` for full column specifications.

Warning in rbind(names(probs), probs_f): number of columns of result is not a multiple of vector length (arg 1)

Warning: 4 parsing failures.

row #	A tibble: 4 x 5	col	row	col	expected	actual	file	expected	<int>	<chr>	<chr>

```
dim(smart.raw)
```

[1] 6014 146

17.4 Cleaning the BRFSS Data

My source for learning about the codes in these variables is the enormous Codebook for the 2016 BRFSS Survey PDF which identifies the variables by name for us. The first two variables in the raw file are

- `DISPCODE` which is 1100 if the subject completed the interview, and 1200 if they partially completed the interview.

```
smart.raw %>% count(DISPCODE)
```

```
# A tibble: 2 x 2
  DISPCODE     n
  <int> <int>
1    1100    5162
2    1200     852
```

- `HHADULT` which is the response to “How many members of your household, including yourself, are 18 years of age or older?”
 - The response is 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.raw <- smart.raw %>%
  mutate(hhadults = HHADULT,
        hhadults = replace(hhadults, hhadults > 76, NA))
```

```
smart.raw %>% count(HHADULT, hhadults) %>% tail()
```

```
# A tibble: 6 x 3
  HHADULT hhadults     n
  <int>     <int> <int>
1       6         6     9
2       7         7     3
3      10        10     2
4      77        NA     4
5      99        NA     1
6      NA        NA  3419
```

17.4.1 Health Status (1 item)

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

17.4.1.1 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.raw <- smart.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.raw %>% count(GENHLTH, genhealth) %>% tail()

# A tibble: 6 x 3
  GENHLTH genhealth     n
  <int>   <fct>      <int>
1       2 2_VeryGood  2014
2       3 3_Good      1895
3       4 4_Fair       798
4       5 5_Poor      298
5       7 <NA>          8
6       9 <NA>          4

```

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

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

17.4.2.1 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.raw <- smart.raw %>%
  mutate(physhealth = PHYSHLTH,
        physhealth = replace(physhealth, physhealth %in% c(77, 99), NA),
        physhealth = replace(physhealth, physhealth == 88, 0))

smart.raw %>% count(PHYSHLTH, physhealth) %>% tail()

```

```

# A tibble: 6 x 3
  PHYSHLTH physhealth     n
  <int>      <dbl> <int>
1       28      28.     5
2       29      29.     1
3       30      30.    482

```

```

4      77      NA     82
5      88      0.   3798
6      99      NA     12

```

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`.

17.4.2.2 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.raw <- smart.raw %>%
  mutate(menthealth = MENTHLTH,
        menthealth = replace(menthealth, menthealth %in% c(77, 99), NA),
        menthealth = replace(menthealth, menthealth == 88, 0))

smart.raw %>% count(MENTHLTH, menthealth) %>% tail()

# A tibble: 6 x 3
  MENTHLTH menthealth     n
  <int>      <dbl> <int>
1      28      28.     3
2      29      29.     8
3      30      30.    340
4      77      NA     58
5      88      0.   4114
6      99      NA     16

```

17.4.2.3 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.raw <- smart.raw %>%
  mutate(poorhealth = POORHLTH,
        poorhealth = replace(poorhealth, poorhealth %in% c(77, 99), NA),
        poorhealth = replace(poorhealth, poorhealth == 88, 0))

smart.raw %>% count(POORHLTH, poorhealth) %>% tail()

# A tibble: 6 x 3
  POORHLTH poorhealth     n
  <int>      <dbl> <int>
1      29      29.     2
2      30      30.    272
3      77      NA     60
4      88      0.   1679

```

5	99	NA	6
6	NA	NA	2977

17.4.3 Health Care Access (4 items)

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

17.4.3.1 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.raw <- smart.raw %>%
  mutate(healthplan = HLTHPLN1,
        healthplan = replace(healthplan, healthplan %in% c(7, 9), NA),
        healthplan = replace(healthplan, healthplan == 2, 0))

smart.raw %>% count(HLTHPLN1, healthplan)

# A tibble: 4 x 3
  HLTHPLN1 healthplan     n
    <int>      <dbl> <int>
1       1        1.  5722
2       2        0.   271
3       7       NA    11
4       9       NA    10
```

17.4.3.2 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

Let's build two variables here. We'll create a `hasdoc` variable which is simply Yes or No, and a factor variable called `numdocs2` which keeps the first three levels in place.

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.

To clean up the PERSDOC2 data into a new variable called numdocs2 we'll - convince R that the 7 and 9 values are in fact best interpreted as NA, - relabel options 1, 2 and 3 while turning the variable into a factor.

```
smart.raw <- smart.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)) %>%
  mutate(numdocs2 = fct_recode(factor(PERSDOC2),
                               "1_One" = "1",
                               "2_MoreThanOne" = "2",
                               "3_Zero" = "3",
                               NULL = "7",
                               NULL = "9"))

smart.raw %>% count(PERSDOC2, hasdoc, numdocs2)

# A tibble: 5 x 4
  PERSDOC2 hasdoc numdocs2      n
  <int>    <dbl> <fct>       <int>
1       1     1. 1_One       4783
2       2     1. 2_MoreThanOne 463
3       3     0. 3_Zero      744
4       7     NA <NA>          14
5       9     NA <NA>          10
```

17.4.3.3 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.raw <- smart.raw %>%
  mutate(costprob = MEDCOST,
        costprob = replace(costprob, costprob %in% c(7, 9), NA),
        costprob = replace(costprob, costprob == 2, 0))

smart.raw %>% count(MEDCOST, costprob)

# A tibble: 4 x 3
  MEDCOST costprob      n
  <int>    <dbl> <int>
1       1     1.    503
2       2     0.   5498
3       7     NA     11
4       9     NA      2
```

17.4.3.4 CHECKUP1 and its cleanup to checktime

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 `checktime` 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.raw <- smart.raw %>%
  mutate(checktime = 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.raw %>% count(CHECKUP1, checktime)

# A tibble: 7 x 3
  CHECKUP1 checktime      n
  <int> <fct>     <int>
1       1 1_In-past-year 4742
2       2 2_1-to-2-years  571
3       3 3_2-to-5-years  299
4       4 4_5_plus_years  299
5       7 <NA>          66
6       8 8_Never        30
7       9 <NA>          7
```

17.4.4 Exercise (1 item)

17.4.4.1 EXERANY2 and its cleanup to exerany

EXERANY2, 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.raw <- smart.raw %>%
  mutate(exerany = EXERANY2,
         exerany = replace(exerany, exerany %in% c(7, 9), NA),
         exerany = replace(exerany, exerany == 2, 0))

smart.raw %>% count(EXERANY2, exerany)

# A tibble: 4 x 3
  EXERANY2 exerany     n
  <int>    <dbl> <int>
1       1      1.  4464
2       2      0.  1537
3       7      NA   10
4       9      NA    3
```

17.4.5 Inadequate Sleep (1 item)

17.4.5.1 SLEPTIM1 and its cleanup to sleephrs

SLEPTIM1, the How Much Time Do You Sleep variable, is the response to “I would like to ask you about your sleep pattern. On average, how many hours of sleep do you get in a 24-hour period?”

- The response is 1-24, 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 24 to NA for our analyses (the blanks are already regarded as NAs by R in the ingestion process.)

```
smart.raw <- smart.raw %>%
  mutate(sleephrs = SLEPTIM1,
         sleephrs = replace(sleephrs, sleephrs > 24, NA))

smart.raw %>% count(SLEPTIM1, sleephrs) %>% tail()

# A tibble: 6 x 3
  SLEPTIM1 sleephrs     n
  <int>    <int> <int>
1      18      18     2
2      20      20     4
3      22      22     1
4      23      23     1
5      77      NA    47
6      99      NA     8
```

17.4.6 Chronic Health Conditions (13 items)

17.4.6.1 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)
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 osteoporosis); tendonitis, bursitis, bunion, tennis elbow; carpal tunnel syndrome, tarsal tunnel syndrome; joint infection, etc.)
ADDEPEV2	hx.depress	(Ever told) 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.raw <- smart.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.

17.4.6.2 DIABETE3 and its cleanup to hx.diabetes and diabetes3

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 `diabetes3` to include all of this information in a factor, but again recode the missing values properly.

```

smart.raw <- smart.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),
        diabetes3 = fct_recode(factor(DIABETE3),
                               "Diabetes" = "1",
                               "Pregnancy-Induced" = "2",
                               "No-Diabetes" = "3",
                               "Pre-Diabetes" = "4",
                               NULL = "7",
                               NULL = "9"))

smart.raw %>% count(DIABETE3, hx.diabetes, diabetes3)

```

	DIABETE3	hx.diabetes	diabetes3	n
	<int>	<dbl>	<fct>	<int>
1	1	1.0	1. Diabetes	833
2	2	0.0	0. Pregnancy-Induced	49
3	3	0.0	0. No-Diabetes	5037
4	4	0.0	0. Pre-Diabetes	85
5	7	NA	<NA>	8
6	9	NA	<NA>	2

17.4.6.3 DIABAGE2 and its cleanup to diab.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.raw <- smart.raw %>%
  mutate(diab.age = DIABAGE2,
        diab.age = replace(diab.age, diab.age > 97, NA))

smart.raw %>% count(DIABAGE2, diab.age) %>% tail()

# A tibble: 6 x 3
  DIABAGE2 diab.age     n
  <int>     <int> <int>
1     81       81     2
2     83       83     1
3     85       85     2
4     98       NA    32
5     99       NA     3
6     NA       NA  5181
```

17.4.7 Oral Health (2 items)

17.4.7.1 LASTDEN3 and its cleanup to denttime

LASTDEN3, the Last Visited Dentist or Dental Clinic variable, is the response to “How long has it been since you last visited a dentist or a dental clinic for any reason? Include visits to dental specialists, such as orthodontists.”

- 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

We’ll again use the approach from CHECKUP1...

```
smart.raw <- smart.raw %>%
  mutate(denttime = fct_recode(factor(LASTDEN3),
                               "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.raw %>% count(LASTDEN3, denttime)

# A tibble: 7 x 3
  LASTDEN3 denttime     n
  <int>     <fct> <int>
1       1 1_In-past-year     1
2       2 2_1-to-2-years     2
3       3 3_2-to-5-years     3
4       4 4_5_plus_years    32
5       7 8_Never             3
6      NA NULL                1
7      NA NULL                1
```

	<int> <fct>	<int>
1	1 1_In-past-year	4221
2	2 2_1-to-2-years	638
3	3 3_2-to-5-years	453
4	4 4_5_plus_years	617
5	7 <NA>	53
6	8 8_Never	24
7	9 <NA>	8

17.4.7.2 RMVTEETH3 and its cleanup to remteeth

RMVTEETH3, the Number of Permanent Teeth Removed variable, is the response to “How many of your permanent teeth have been removed because of tooth decay or gum disease? Include teeth lost to infection, but do not include teeth lost for other reasons, such as injury or orthodontics. (If wisdom teeth are removed because of tooth decay or gum disease, they should be included in the count for lost teeth).”

- 1 = 1 to 5
- 2 = 6 or more but not all
- 3 = All
- 7 = Don’t know/Not sure
- 8 = None
- 9 = Refused
- BLANK = Not asked or missing

We’ll recode this to `remteeth` which will be 0 for None, 1 for 1-5, 2 for 6+ but not all, and 3 for All.

```
smart.raw <- smart.raw %>%
  mutate(remteeth = fct_recode(factor(RMVTEETH3),
    "None" = "8",
    "1-5" = "1",
    "6+_butnotAll" = "2",
    "All" = "3",
    NULL = "7",
    NULL = "9"))

smart.raw %>% count(RMVTEETH3, remteeth)
```

	RMVTEETH3	remteeth	n
	<int> <fct>	<int>	
1	1 1-5		1632
2	2 6+_butnotAll		775
3	3 All		485
4	7 <NA>		118
5	8 None		2993
6	9 <NA>		11

17.4.8 Demographics (23 items)

17.4.8.1 _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.

<u>_AGEG5YR</u>	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

```

smart.raw <- smart.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.raw %>% count(`_AGEG5YR`, agegroup)

# A tibble: 14 x 3
`_AGEG5YR` agegroup     n
<int> <fct>     <int>
1 1 18-24     348
2 2 25-29     280
3 3 30-34     322
4 4 35-39     298
5 5 40-44     346
6 6 45-49     437
7 7 50-54     504
8 8 55-59     666
9 9 60-64     651
10 10 65-69    711
11 11 70-74    482
12 12 75-79    395
13 13 80-96    498
14 <NA>        76

```

17.4.8.2 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 0.

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

```
smart.raw %>% count(SEX, female)
```

```
# A tibble: 3 x 3
  SEX   female      n
  <int> <dbl> <int>
1     1     0.  2559
2     2     1.  3454
3     9     NA    1
```

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

```
smart.raw %>% count(MARITAL, marital)
```

```
# A tibble: 7 x 3
  MARITAL   marital      n
  <int> <fct>       <int>
1     1 Married     2989
2     2 Divorced    888
3     3 Widowed    813
4     4 Separated   107
```

5	5 Never_Married	1019
6	6 Unmarried_Couple	148
7	9 <NA>	50

17.4.8.4 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.raw <- smart.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.raw %>% count(EDUCA, educgroup)
```

	EDUCA	educgroup	n
	<int>	<fct>	<int>
1	1	Kindergarten	5
2	2	Elementary	80
3	3	Some_HS	287
4	4	HS_Grad	1745
5	5	Some_College	1611
6	6	College_Grad	2270
7	9	<NA>	16

17.4.8.5 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.raw <- smart.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.raw %>% count(RENTHOM1, home.own)

# A tibble: 5 x 3
  RENTHOM1 home.own     n
  <int>     <dbl> <int>
1       1      1.  4296
2       2      0.  1486
3       3      0.   196
4       7      NA     9
5       9      NA    27
```

17.4.8.6 NUMHHOL2 and its cleanup to phones2

NUMHHOL2, the Household Telephones variable, is the response to “Do you have more than one telephone number in your household? Do not include cell phones or numbers that are only used by a computer or fax machine.”

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

```
smart.raw <- smart.raw %>%
  mutate(phones2 = NUMHHOL2,
        phones2 = replace(phones2, phones2 %in% c(7, 9), NA),
        phones2 = replace(phones2, phones2 == 2, 0))

smart.raw %>% count(NUMHHOL2, phones2)

# A tibble: 5 x 3
  NUMHHOL2 phones2     n
  <int>     <dbl> <int>
1       1      1.   222
2       2      0.  3151
3       7      NA     2
4       9      NA     8
5      NA      NA  2631
```

17.4.8.7 NUMPHON2 and its cleanup to numphones

NUMPHON2, the Residential Phones variable, is the response to “How many of these telephone numbers are residential numbers?”

- 1-6 = legitimate responses (6 = 6 or more)
- 7 = Don’t know/Not sure
- 9 = Refused
- BLANK = Not asked or missing

```
smart.raw <- smart.raw %>%
  mutate(numphones = NUMPHON2,
        numphones = replace(numphones, numphones > 6, NA))

smart.raw %>% count(NUMPHON2, numphones)

# A tibble: 9 x 3
  NUMPHON2 numphones     n
  <int>      <int> <int>
1       1          1    131
2       2          2     66
3       3          3     10
4       4          4      5
5       5          5      4
6       6          6      2
7       7         NA      2
8       9         NA      2
9      NA         NA   5792
```

17.4.8.8 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.raw <- smart.raw %>%
  mutate(veteran = VETERAN3,
        veteran = replace(veteran, veteran %in% c(7, 9), NA),
        veteran = replace(veteran, veteran == 2, 0))

smart.raw %>% count(VETERAN3, veteran)
```

```
# A tibble: 5 x 3
  VETERAN3 veteran     n
  <int>      <dbl> <int>
1       1      1.    736
2       2      0.   5264
3       7     NA      1
4       9     NA     12
5      NA     NA      1
```

17.4.8.9 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.raw <- smart.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.raw %>% count(EMPLOY1, employment)

# A tibble: 10 x 3
  EMPLOY1 employment      n
  <int> <fct>        <int>
1     1 Employed_for_wages 2614
2     2 Self-employed      378
3     3 Outofwork_1yearormore 103
4     4 Outofwork_lt1year    110
5     5 Homemaker          302
6     6 Student             144
7     7 Retired            1868
8     8 Unable_to_work      453
9     9 <NA>                41
10    NA <NA>                 1
```

17.4.8.10 CHILDREN and its cleanup to numkids

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.raw <- smart.raw %>%
  mutate(numkids = CHILDREN,
    numkids = replace(numkids, numkids == 99, NA),
    numkids = replace(numkids, numkids == 88, 0))

smart.raw %>% count(CHILDREN, numkids) %>% tail()

# A tibble: 6 x 3
```

	CHILDREN	numkids	n
	<int>	<dbl>	<int>
1	7	7.	2
2	8	8.	2
3	9	9.	4
4	88	0.	4408
5	99	NA	39
6	NA	NA	1

17.4.8.11 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.raw <- smart.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.raw %>% count(`INCOME2`, incomegroup)

# A tibble: 11 x 3
  INCOME2 incomegroup     n
  <int> <fct>      <int>
1      1 0-9K        212
2      2 10-14K       235
3      3 15-19K       374
4      4 20-24K       482
5      5 25-34K       567
6      6 35-49K       723
7      7 50-74K       814
8      8 75K+       1631
9     77 <NA>        387
```

```
10      99 <NA>          565
11      NA <NA>           24
```

17.4.8.12 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.raw <- smart.raw %>%
  mutate(internet30 = INTERNET,
        internet30 = replace(internet30, internet30 %in% c(7, 9), NA),
        internet30 = replace(internet30, internet30 == 2, 0))

smart.raw %>% count(INTERNET, internet30)
```

```
# A tibble: 5 x 3
  INTERNET internet30     n
  <int>       <dbl> <int>
1      1         1.  4895
2      2         0.  1078
3      7         NA    6
4      9         NA    6
5     NA         NA   29
```

17.4.8.13 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.raw <- smart.raw %>%
  mutate(weight_kg = WTKG3/100)

smart.raw %>% count(WTKG3, weight_kg) # %>% tail()

# A tibble: 257 x 3
  WTKG3 weight_kg     n
  <int>     <dbl> <int>
1  3084     30.8     1
2  3266     32.7     1
3  3447     34.5     1
4  3629     36.3     2
5  3674     36.7     1
6  3856     38.6     3
7  3946     39.5     1
8  4037     40.4     1
9  4082     40.8     3
10 4128     41.3     1
# ... with 247 more rows
```

17.4.8.14 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.raw <- smart.raw %>%
  mutate(height_m = case_when(
    HEIGHT3 >= 300 & HEIGHT3 <= 511 ~ round((12*floor(HEIGHT3/100) + (HEIGHT3 - 100*floor(HEIGHT3/100))/100)),
    HEIGHT3 >= 600 & HEIGHT3 <= 711 ~ round((12*floor(HEIGHT3/100) + (HEIGHT3 - 100*floor(HEIGHT3/100))/100)),
    HEIGHT3 >= 9000 & HEIGHT3 <= 9224 ~ ((HEIGHT3 - 9000)/100)))

smart.raw %>% count(HEIGHT3, height_m) # %>% tail()

# A tibble: 35 x 3
  HEIGHT3 height_m      n
  <int>     <dbl> <int>
1     406     1.37     1
2     407     1.40     1
3     408     1.42     4
4     409     1.45    14
5     410     1.47    23
6     411     1.50    63
7     500     1.52   184
8     501     1.55   171
9     502     1.57   415
10    503     1.60   408
# ... with 25 more rows
```

17.4.8.15 bmi is calculated from height_m and weight_kg

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

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

smart.raw %>% count(height_m, weight_kg, bmi) # %>% tail()

# A tibble: 1,579 x 4
  height_m weight_kg      bmi      n
  <dbl>     <dbl>     <dbl> <int>
1     1.37     68.0    36.2     1
2     1.40     81.6    41.7     1
3     1.42     40.4    20.0     1
4     1.42     66.2    32.8     1
5     1.42     72.6    36.0     2
6     1.45     44.4    21.1     1
7     1.45     45.4    21.6     1
```

```

8      1.45      46.7  22.2      1
9      1.45      52.2  24.8      2
10     1.45      53.5  25.5      1
# ... with 1,569 more rows

```

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

```

```
smart.raw %>% count(bmigroup)
```

```

# A tibble: 5 x 2
  bmigroup          n
  <fct>        <int>
1 [ 11.6, 18.5)    101
2 [ 18.5, 25.0)   1722
3 [ 25.0, 30.0)   2001
4 [ 30.0,110.9]   1765
5 <NA>            425

```

17.4.8.17 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.raw <- smart.raw %>%
  mutate(pregnant = PREGNANT,
         pregnant = replace(pregnant, pregnant %in% c(7, 9), NA),
         pregnant = replace(pregnant, pregnant == 2, 0))

```

```
smart.raw %>% count(PREGNANT, pregnant)
```

```

# A tibble: 5 x 3
  PREGNANT pregnant      n
  <int>      <dbl> <int>
1       1        1.    33
2       2        0.   797
3       7        NA     5
4       9        NA     1
5      NA        NA  5178

```

17.4.8.18 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.raw <- smart.raw %>%
  mutate(deaf = DEAF,
        deaf = replace(deaf, deaf %in% c(7, 9), NA),
        deaf = replace(deaf, deaf == 2, 0))

smart.raw %>% count(DEAF, deaf)
```

```
# A tibble: 5 x 3
  DEAF   deaf     n
  <int> <dbl> <int>
1     1    1.    507
2     2    0.    5392
3     7    NA     9
4     9    NA     7
5    NA    NA    99
```

17.4.8.19 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.raw <- smart.raw %>%
  mutate(blind = BLIND,
        blind = replace(blind, blind %in% c(7, 9), NA),
        blind = replace(blind, blind == 2, 0))

smart.raw %>% count(BLIND, blind)
```

```
# A tibble: 5 x 3
  BLIND blind     n
  <int> <dbl> <int>
1     1    1.    290
2     2    0.    5605
3     7    NA     9
4     9    NA     2
5    NA    NA    108
```

17.4.8.20 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.raw <- smart.raw %>%
  mutate(decide = DECIDE,
         decide = replace(decide, decide %in% c(7, 9), NA),
         decide = replace(decide, decide == 2, 0))

smart.raw %>% count(DECIDE, decide)
```

```
# A tibble: 5 x 3
  DECIDE decide     n
  <int>   <dbl> <int>
1      1     1.  583
2      2     0.  5293
3      7     NA    16
4      9     NA     5
5     NA     NA   117
```

17.4.8.21 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.raw <- smart.raw %>%
  mutate(diffwalk = DIFFWALK,
         diffwalk = replace(diffwalk, diffwalk %in% c(7, 9), NA),
         diffwalk = replace(diffwalk, diffwalk == 2, 0))

smart.raw %>% count(DIFFWALK, diffwalk)
```

```
# A tibble: 5 x 3
  DIFFWALK diffwalk     n
  <int>   <dbl> <int>
1      1     1.  1036
2      2     0.  4841
3      7     NA    11
4      9     NA     5
5     NA     NA   121
```

17.4.8.22 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.raw <- smart.raw %>%
  mutate(difffress = DIFFDRES,
        difffress = replace(difffress, difffress %in% c(7, 9), NA),
        difffress = replace(difffress, difffress == 2, 0))

smart.raw %>% count(DIFFDRES, difffress)
```

	DIFFDRES	difffress	n
	<int>	<dbl>	<int>
1	1	1.	221
2	2	0.	5658
3	7	NA	3
4	9	NA	2
5	NA	NA	130

17.4.8.23 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.raw <- smart.raw %>%
  mutate(diffalone = DIFFALON,
        diffalone = replace(diffalone, diffalone %in% c(7, 9), NA),
        diffalone = replace(diffalone, diffalone == 2, 0))

smart.raw %>% count(DIFFALON, diffalone)
```

	DIFFALON	diffalone	n
	<int>	<dbl>	<int>
1	1	1.	437
2	2	0.	5419
3	7	NA	13
4	9	NA	6
5	NA	NA	139

17.4.9 Tobacco Use (5 items)

17.4.9.1 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.raw <- smart.raw %>%
  mutate(smoke100 = SMOKE100,
        smoke100 = replace(smoke100, smoke100 %in% c(7, 9), NA),
        smoke100 = replace(smoke100, smoke100 == 2, 0))

smart.raw %>% count(SMOKE100, smoke100)
```

	SMOKE100	smoke100	n
	<int>	<dbl>	<int>
1	1	1.	2567
2	2	0.	3260
3	7	NA	30
4	9	NA	6
5	NA	NA	151

17.4.9.2 SMOKDAY2

17.4.9.3 STOPSMK2

17.4.9.4 LASTSMK2

17.4.9.5 USENOW3

17.4.10 E-Cigarettes (2 items)

17.4.10.1 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.raw <- smart.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.raw %>% count(ECIGARET, ecig.ever)
```

```
# A tibble: 5 x 3
  ECIGARET ecig.ever     n
  <int>      <dbl> <int>
1       1        1.  1028
2       2        0.  4818
3       7        NA   8
4       9        NA   1
5      NA        NA  159
```

17.4.10.2 ECIGNOW

17.4.11 Alcohol Consumption (4 items)

17.4.11.1 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.raw <- smart.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)*30/7, 0),
                             TRUE ~ alcdays))

smart.raw %>% count(ALCDAY5, alcdays)
```

```
# A tibble: 38 x 3
  ALCDAY5 alcdays     n
  <int>    <dbl> <int>
1     101     4.   256
2     102     9.   197
3     103    13.   142
4     104    17.    62
5     105    21.    51
6     106    26.    11
7     107    30.   124
8     201     1.   471
9     202     2.   363
10    203     3.   216
# ... with 28 more rows
```

17.4.11.2 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.raw <- smart.raw %>%
  mutate(avgdrinks = AVEDRNK2,
        avgdrinks = replace(avgdrinks, avgdrinks > 76, NA))

smart.raw %>% count(AVEDRNK2, avgdrinks) %>% tail()
```

```
# A tibble: 6 x 3
  AVEDRNK2 avgdrinks     n
  <int>      <int> <int>
1     40        40     2
2     50        50     2
3     75        75     1
4     77        NA    36
5     99        NA    10
6     NA        NA  2981
```


17.4.11.3 DRNK3GE5

17.4.11.4 MAXDRNKS

17.4.12 Immunization (4 items)

17.4.12.1 FLUSHOT6

17.4.12.2 FLSHTMY2

17.4.12.3 PNEUVAC3

17.4.12.4 TETANUS

17.4.13 Falls (2 items)

17.4.13.1 FALL12MN

17.4.13.2 FALLINJ2

17.4.14 Seatbelt Use (1 item)

17.4.14.1 SEATBELT

17.4.15 Drinking and Driving (1 item)

17.4.15.1 DRNKDRI2

17.4.16 Breast and Cervical Cancer Screening (7 items)

17.4.16.1 HADMAM

17.4.16.2 HOWLONG

17.4.16.3 HADPAP2

17.4.16.4 LASTPAP2

17.4.16.5 HPVTEST

17.4.16.6 HPLSTTST

17.4.16.7 HADHYST2

17.4.17 Prostate Cancer Screening (6 items)

17.4.17.1 PCPSAAD2

17.4.17.2 PCPSADI1

17.4.17.3 PCPSARE1

17.4.17.4 PSATEST1

17.4.17.5 PSATIME

17.4.17.6 PCPSARS1

17.4.18 Colorectal Cancer Screening (5 items)

17.4.18.1 PLRSTCOL

17.5.1 age_imp: Creating Age Data out of Thin Air

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(20180114)

smart.raw <- smart.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.raw %>% count(agegroup, age_imp) %>% tail()

# A tibble: 6 x 3
  agegroup age_imp     n
  <fct>    <dbl> <int>
1 80–96      92.    30
2 80–96      93.    30
3 80–96      94.    33
4 80–96      95.    33
5 80–96      96.    14
6 <NA>        NA     76
```

17.6 Clean Data in the State of Ohio

There are five MMSAs associated with the state of Ohio, associated with Cincinnati, Cleveland, Columbus, Dayton and Toledo.

```
smart_ohio <- smart.raw %>%
  filter(str_detect(MMSANAME, ', OH')) %>%
  select(SEQNO, MMSANAME, hhadults,
         genhealth, physhealth, menthealth,
         poorhealth, healthplan, hasdoc, numdocs2,
         costprob, checktime, exerany, sleephrs,
         hx.mi, hx.chd, hx.stroke, hx.asthma,
         now.asthma, hx.skinc, hx.otherc, hx.copd,
         hx.arthr, hx.depress, hx.kidney,
         hx.diabetes, diabetes3, diab.age,
         denttime, remteeth, agegroup, female,
         marital, educgroup, home.own, phones2,
         numphones, veteran, employment, numkids,
         incomegroup, internet30, weight_kg,
         height_m, bmi, bmigroup, pregnant, deaf,
         blind, decide, diffwalk, diffdress,
         diffalone, smoke100, ecig.ever, alcdays,
         avgdrinks, age_imp)

skim_with(numeric = list(hist = NULL))
skim(select(smart_ohio, -SEQNO))
```

Skim summary statistics

n obs: 6014
n variables: 57

Variable type: character

	variable	missing	complete	n	min	max	empty	n_unique
MMSANAME		0	6014	6014	41	51	0	5

Variable type: factor

	variable	missing	complete	n	n_unique	top_counts	ordered
agegroup	76	5938	6014	13			
bmi	425	5589	6014	4			
checktime	73	5941	6014	5			
denttime	61	5953	6014	5			
diabetes3	10	6004	6014	4			
educgroup	16	5998	6014	6			
employment	42	5972	6014	8			
genhealth	12	6002	6014	5			
incomegroup	976	5038	6014	8			
marital	50	5964	6014	6			
numdocs2	24	5990	6014	3			
remteeth	129	5885	6014	4			
Col:	2270	HS_:	1745	Som:	1611	Som:	287
Emp:	2614	Ret:	1868	Una:	453	Sel:	378
2_V:	2014	3_G:	1895	1_E:	997	4_F:	798
75K:	1631	NA:	976	50-:	814	35-:	723
Mar:	2989	Nev:	1019	Div:	888	Wid:	813
1_O:	4783	3_Z:	744	2_M:	463	NA:	24
Non:	2993	1-5:	1632	6+_-:	775	All:	485

Variable type: integer

	variable	missing	complete	n	mean	sd	p0	p25	median	p75	p100
avgdrinks	3027	2987	6014	2.28	2.96	1	1	2	2	75	
diab.age	5216	798	6014	50.83	14.12	1	43	51	60	85	
hhadults	3424	2590	6014	2.08	0.97	1	1	2	2	10	
numphones	5796	218	6014	1.58	0.94	1	1	1	2	6	
sleephrs	55	5959	6014	6.97	1.51	1	6	7	8	23	

Variable type: numeric

	variable	missing	complete	n	mean	sd	p0	p25	median	p75
age_imp	76	5938	6014	55.96	18.34	18	43	57	68	
alcdays	217	5797	6014	4.76	8.02	0	0	1	5	
blind	119	5895	6014	0.049	0.22	0	0	0	0	
bmi	425	5589	6014	28.36	6.69	11.61	23.92	27.28	31.41	
costprob	13	6001	6014	0.084	0.28	0	0	0	0	
deaf	115	5899	6014	0.086	0.28	0	0	0	0	
decide	138	5876	6014	0.099	0.3	0	0	0	0	
diffalone	158	5856	6014	0.075	0.26	0	0	0	0	


```

1
1
1
1
1
1
1
30
1
9
1
30
30
1
1
1
249.48

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

```

17.7 Clean Cleveland-Elyria Data

17.7.1 Cleveland - Elyria Raw Data

The MMSANAME 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 MMSANAME that contain the text "Cleveland".

```

smart.cle.raw <- smart.raw %>%
  filter(str_detect(MMSANAME, 'Cleveland'))

```

In the Cleveland-Elyria MSA, we have 1036 observations on the same 204 variables.

17.7.2 Clean Data - Larger

```

smart.cle <- smart.cle.raw %>%
  select(SEQNO, MMSANAME, hhadults,
         genhealth, physhealth, menthealth,
         poorhealth, healthplan, hasdoc, numdocs2,
         costprob, checktime, exerany, sleephrs,
         hx.mi, hx.chd, hx.stroke, hx.asthma,
         now.asthma, hx.skinc, hx.otherc, hx.copd,
         hx.arthr, hx.depress, hx.kidney,
         hx.diabetes, diabetes3, diab.age,
         denttime, remteeth, agegroup, female,
         marital, educgroup, home.own, phones2,
         numphones, veteran, employment, numkids,
         incomegroup, internet30, weight_kg,
         height_m, bmi, bmigroup, pregnant, deaf,
         blind, decide, diffwalk, diffdress,
         diffalone, smoke100, ecig.ever, alcdays,
         avgdrinks, age_imp)

```

```
skim_with(numeric = list(hist = NULL))
skim(select(smart.cle, -SEQNO))
```

Skim summary statistics

n obs: 1036

n variables: 57

Variable type: character

	variable	missing	complete	n	min	max	empty	n_unique
MMSANAME		0	1036	1036	51	51	0	1

Variable type: factor

	variable	missing	complete	n	n_unique			
agegroup		12	1024	1036	13			
bmigroup		84	952	1036	4			
checktime		9	1027	1036	5			
denttime		6	1030	1036	5			
diabetes3		1	1035	1036	4			
educgroup		2	1034	1036	6			
employment		10	1026	1036	8			
genhealth		3	1033	1036	5			
incomegroup		170	866	1036	8			
marital		10	1026	1036	6			
numdocs2		5	1031	1036	3			
remteeth		25	1011	1036	4			
			top_counts	ordered				
65-:	128,	60-:	122,	55-:	105,	80-:	103	FALSE
[2:	349,	[1:	322,	[3:	268,	NA:	84	FALSE
1_I:	820,	2_1:	101,	3_2:	52,	4_5:	49	FALSE
1_I:	752,	2_1:	113,	4_5:	91,	3_2:	70	FALSE
No-:	884,	Dia:	128,	Pre:	15,	Pre:	8	FALSE
Col:	385,	HS_:	292,	Som:	290,	Som:	54	FALSE
Emp:	427,	Ret:	345,	Sel:	70,	Una:	57	FALSE
2_V:	350,	3_G:	344,	1_E:	173,	4_F:	122	FALSE
75K:	258,	NA:	170,	50-:	146,	35-:	115	FALSE
Mar:	476,	Nev:	213,	Wid:	157,	Div:	143	FALSE
1_O:	795,	3_Z:	135,	2_M:	101,	NA:	5	FALSE
Non:	510,	1-5:	293,	6+:	127,	All:	81	FALSE

Variable type: integer

	variable	missing	complete	n	mean	sd	p0	p25	median	p75	p100
avgdrinks		510	526	1036	2.17	2.44	1	1	2	2	40
diab.age		912	124	1036	52.35	12.96	5	46.75	55	60	83
hhadults		586	450	1036	2.07	1.02	1	1	2	2	6
numphones		994	42	1036	1.74	0.99	1	1	2	2	5
sleephrs		8	1028	1036	7.02	1.53	1	6	7	8	20

Variable type: numeric

	variable	missing	complete	n	mean	sd	p0	p25	median	p75
age_imp		12	1024	1036	57.3	18.86	18	45	60	70
alcdays		46	990	1036	4.65	8.05	0	0	1	4
blind		20	1016	1036	0.037	0.19	0	0	0	0
bmi		84	952	1036	27.89	6.47	12.71	23.7	26.68	30.53


```
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
1  
30  
1  
6  
1  
30  
30  
1  
1  
1  
204.12  
write_csv(smart.cle, "data/smartercle.csv")
```

17.7.3 Creation of the smartcle1.csv data

```
smart.cle1 <- smart.raw %>%  
  filter(str_detect(MMSANAME, 'Cleveland')) %>%  
  select(SEQNO, physhealth, menthealth,  
         poorhealth, genhealth, bmi, female,  
         internet30, exerany, sleephrs, alcdays)  
  
# write_csv(smart.cle1, "data/smartercle1.csv")  
  
## this last line commented out because the file was  
## created earlier in these notes
```

Chapter 18

Modeling a Count Outcome in Ohio SMART

In this chapter, I use a count outcome (# of poor physical health days out of the last 30) to demonstrate regression models for count outcomes.

Methods discussed in the chapter include the following:

- Ordinary Least Squares
- Poisson Regression
- Overdispersed Quasi-Poisson Regression
- Negative Binomial Regression
- Zero-Inflated Poisson Regression
- Zero-Inflated Negative Binomial Regression
- Hurdle Models with Poisson counts
- Hurdle Models with Negative Binomial counts
- Tobit Regression

18.1 Preliminaries

```
library(boot)
library(lmtest)
library(sandwich)
library(countreg)
library(VGAM)

smart_oh <- read.csv("data/smart_ohio.csv") %>%tbl_df
```

18.2 A subset of the Ohio SMART data

Let's consider the following data. I'll include the subset of all observations in `smart_oh` with complete data on these 14 variables.

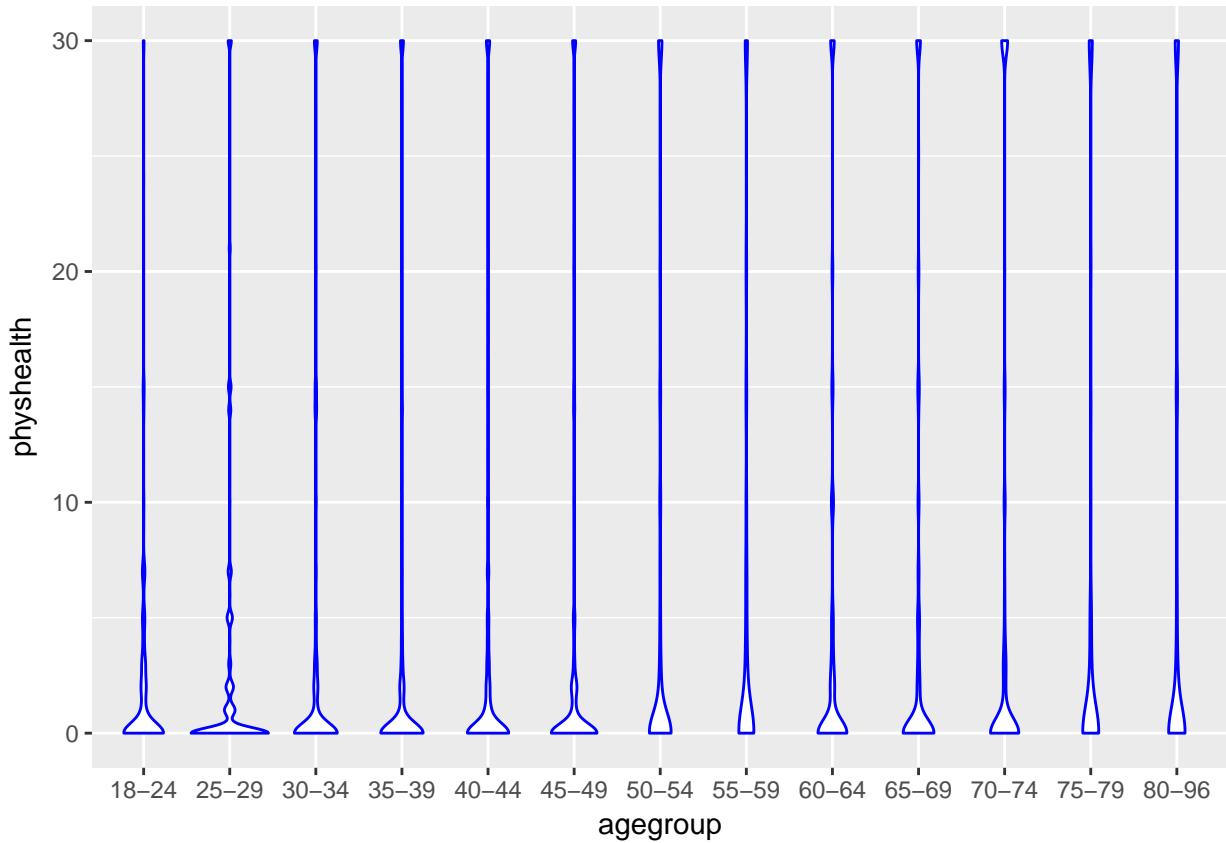
Variable	Description
SEQNO	Subject identification code
MMSANAME	Name of statistical area

Variable	Description
genhealth	Five categories (E, VG, G, F, P) on general health
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?
menthlth	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?
healthplan	1 if the subject has any kind of health care coverage, 0 otherwise
costprob	1 indicates Yes to “Was there a time in the past 12 months when you needed to see a doctor but could not because of cost?”
sleephrs	average amount of sleep the subject gets in a 24-hour period
agegroup	13 age groups from 18 through 80+
female	1 if subject is female
incomegroup	8 income groups from < 10,000 to 75,000 or more
bmi	body-mass index
smoke100	1 if Yes to “Have you smoked at least 100 cigarettes in your entire life?”
alcdays	# of days out of the past 30 on which the subject had at least one alcoholic drink

```
sm_oh_A <- smart_oh %>%
  select(SEQNO, MMSANAME, genhealth, physhealth,
         menthlth, healthplan, costprob, sleephrs,
         agegroup, female, incomegroup, bmi, smoke100,
         alcdays) %>%
  drop_na
```

18.2.1 Is age group associated with physhealth?

```
ggplot(sm_oh_A, aes(x = agegroup, y = physhealth)) +
  geom_violin(col = "blue")
```



It's hard to see much of anything here. The main conclusion seems to be that 0 is by far the most common response.

Here's a table by age group of:

- the number of respondents in that age group,
- the group's mean `physhealth` response (remember that these are the number of poor physical health days in the last 30),
- their median `physhealth` response (which turns out to be 0 in each group), and
- the percentage of group members who responded 0.

```
sm_oh_A %>% group_by(agegroup) %>%
  summarize(n = n(), mean = round(mean(physhealth), 2),
            median = median(physhealth),
            percent_0s = round(100*sum(physhealth == 0)/n, 1))
```

	agegroup	n	mean	median	percent_0s
1	18-24	243	1.91	0.	65.4
2	25-29	220	2.39	0.	66.4
3	30-34	282	3.11	0.	68.8
4	35-39	241	2.95	0.	70.1
5	40-44	285	2.91	0.	66.0
6	45-49	364	2.95	0.	69.8
7	50-54	403	5.11	0.	61.5
8	55-59	510	5.37	0.	57.3
9	60-64	515	4.33	0.	61.2

10	65-69	550	4.34	0.	64.2
11	70-74	349	5.38	0.	64.5
12	75-79	270	5.54	0.	60.7
13	80-96	320	5.68	0.	61.6

We can see a real change between the 45-49 age group and the 50-54 age group. The mean difference is clear from the table above, and the plot below (of the percentage with a zero response) in each age group identifies the same story.

```
sm_oh_A %>% group_by(agegroup) %>%
  summarize(n = n(),
            percent_0s = round(100*sum(physhealth == 0)/n,1)) %>%
  ggplot(aes(y = agegroup, x = percent_0s)) +
  geom_label(aes(label = percent_0s)) +
  labs(x = "% with no Bad Physical Health Days in last 30",
       y = "Age Group")
```



It looks like we have a fairly consistent result in the younger age range (18-49) or the older range (50+). On the theory that most of the people reading this document are in that younger range, we'll focus on those respondents in what follows.

18.3 Exploratory Data Analysis (in the 18-49 group)

We want to predict the 0-30 physhealth count variable for the 18-49 year old respondents.

To start, we'll use two predictors:

- the respondent's body mass index, and
- whether the respondent has smoked 100 cigarettes in their lifetime.

We anticipate that each of these variables will have positive associations with the `physhealth` score. That is, heavier people, and those who have used tobacco will be less healthy, and thus have higher numbers of poor physical health days.

18.3.1 Build a subset of those ages 18-49

First, we'll identify the subset of respondents who are between 18 and 49 years of age.

```
sm_oh_A_young.raw <- sm_oh_A %>%
  filter(agegroup %in% c("18-24", "25-29", "30-34",
                        "35-39", "40-44", "45-49")) %>%
  droplevels()

sm_oh_A_young.raw %>%
  select(physhealth, bmi, smoke100, agegroup) %>%
  summary
```

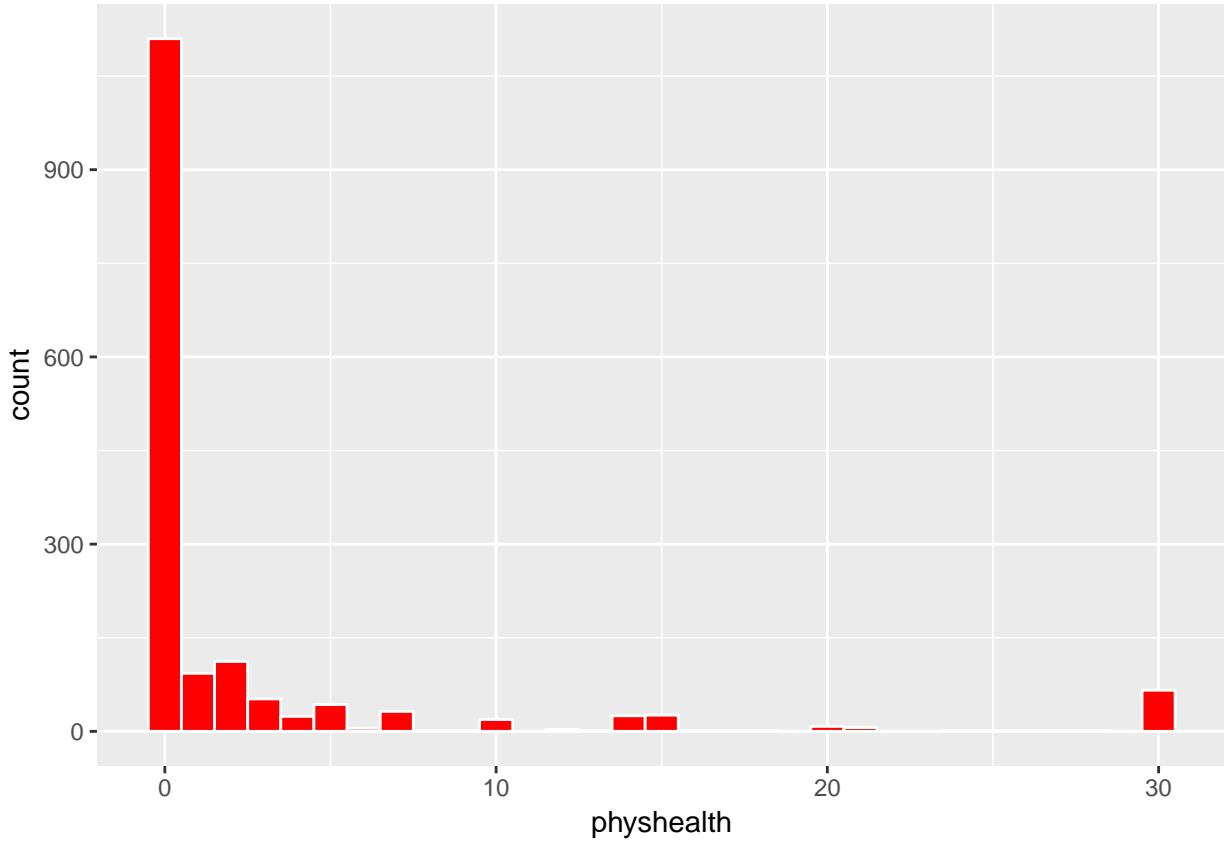
physhealth	bmi	smoke100	agegroup
Min. : 0.000	Min. : 12.71	Min. :0.0000	18-24:243
1st Qu.: 0.000	1st Qu.: 23.50	1st Qu.:0.0000	25-29:220
Median : 0.000	Median : 26.69	Median :0.0000	30-34:282
Mean : 2.741	Mean : 28.21	Mean :0.3676	35-39:241
3rd Qu.: 2.000	3rd Qu.: 31.39	3rd Qu.:1.0000	40-44:285
Max. :30.000	Max. :110.88	Max. :1.0000	45-49:364

Now, let's look more closely at the distribution of these variables, starting with our outcome.

18.3.2 Distribution of the Outcome

What's the distribution of `physhealth`?

```
ggplot(sm_oh_A_young.raw, aes(x = physhealth)) +
  geom_histogram(binwidth = 1, fill = "red", col = "white")
```



```
sm_oh_A_young.raw %>%
  count(physhealth == 0, physhealth == 30)
```

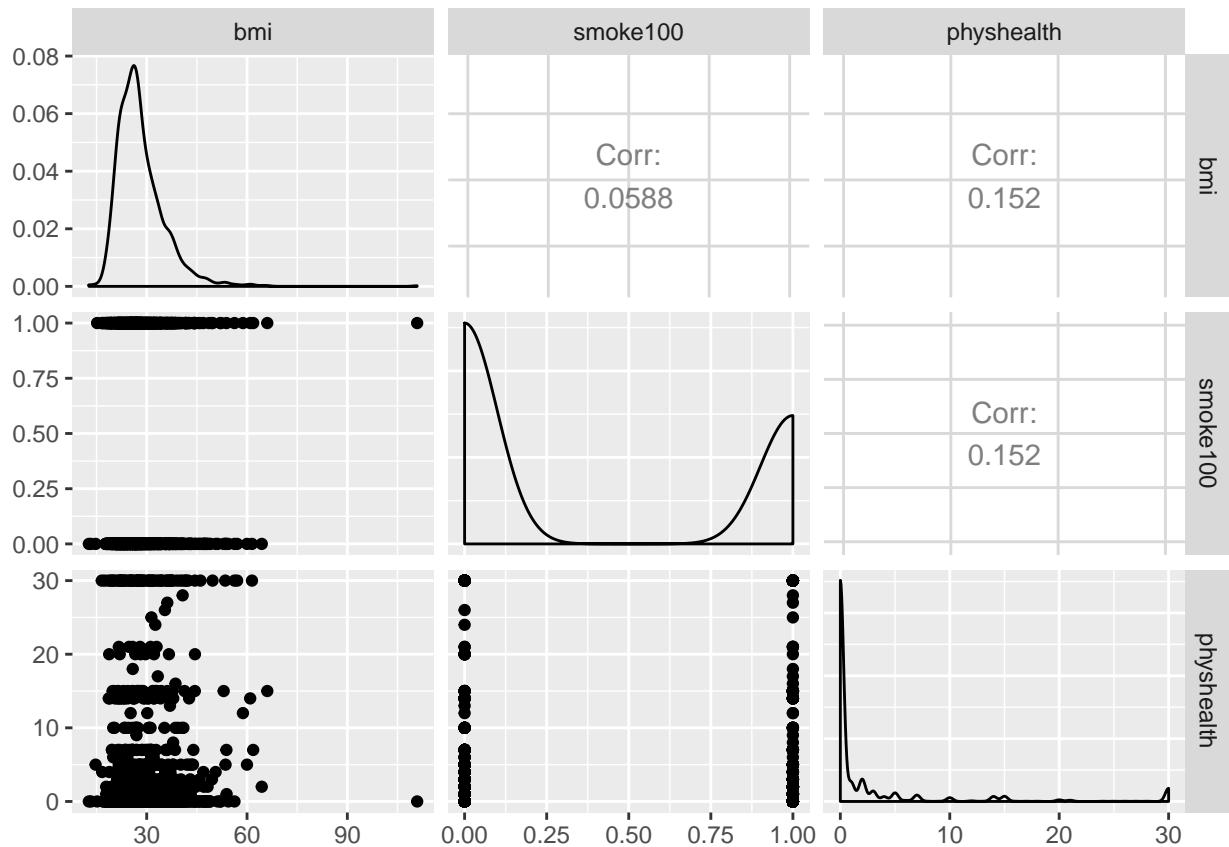
```
# A tibble: 3 x 3
  `physhealth == 0` `physhealth == 30`     n
  <lgl>              <lgl>             <int>
1 FALSE              FALSE             459
2 FALSE              TRUE              66
3 TRUE               FALSE            1110
```

Most of our respondents said zero, the minimum allowable value, although there is also a much smaller bump at 30, the maximum value we will allow.

Dealing with this distribution is going to be a bit of a challenge. We will develop a series of potential modeling approaches for this sort of data, but before we do that, let's look at the distribution of our other two variables, and the pairwise associations, in a scatterplot matrix.

18.3.3 Initial Scatterplot Matrix

```
GGally::ggpairs(sm_oh_A_young.raw %>%
  select(bmi, smoke100, physhealth))
```



18.3.4 Dropping the subject with an unreasonably large `bmi`

It looks like there is a very large outlier in the `bmi` data. This subject appears to be a full 40 kg/m^2 larger than the next largest subject. Without a good explanation for that discrepancy, I will remove that subject before fitting models. I'm also going to center the `bmi` variable to help me interpret the final models later.

```
sm_oh_A_young <- sm_oh_A_young.raw %>%
  mutate(bmi_c = bmi - mean(bmi)) %>%
  filter(bmi < 80)

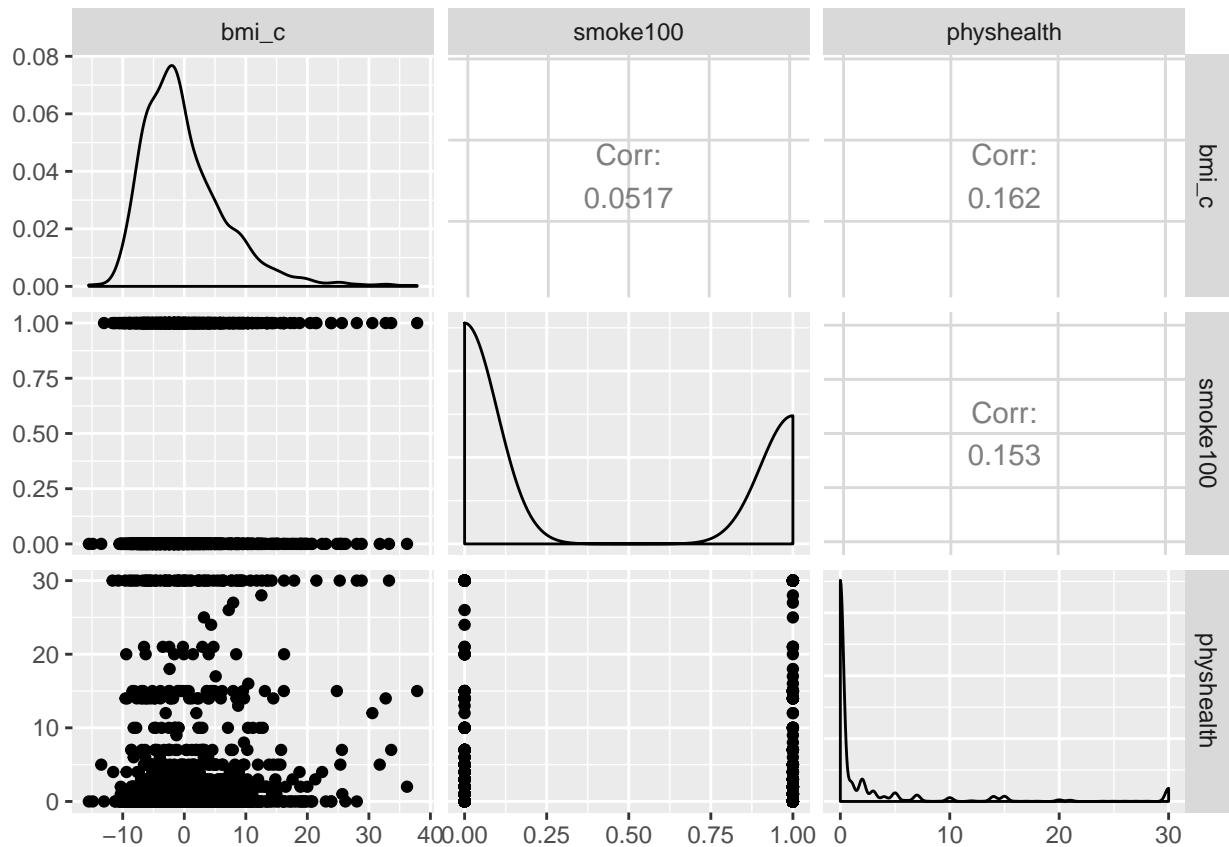
nrow(sm_oh_A_young)
```

[1] 1634

18.3.5 Revised (Final) Scatterplot Matrix

Now, here's the revised scatterplot matrix for those 1634 subjects, now using the centered `bmi` data captured in the `bmi_c` variable.

```
GGally::ggpairs(sm_oh_A_young %>%
  select(bmi_c, smoke100, physhealth))
```



So `bmi_c` and `smoke100` each have modest positive correlations with `physhealth` and only a very small correlation with each other. Here are some summary statistics for this final data.

18.3.6 Summary of the final subset of data

Remember that since the mean of `bmi` is 28.2, the `bmi_c` values are just `bmi` - 28.2 for each subject.

```
sm_oh_A_youth %>%
  select(bmi, bmi_c, smoke100, physhealth) %>%
  skim()
```

Skim summary statistics

n obs: 1634

n variables: 4

Variable type: integer

	variable	missing	complete	n	mean	sd	p0	p25	median	p75	p100
physhealth		0	1634	1634	2.74	6.77	0	0	0	2	30
smoke100		0	1634	1634	0.37	0.48	0	0	0	1	1

Variable type: numeric

	variable	missing	complete	n	mean	sd	p0	p25	median	p75	p100
bmi		0	1634	1634	28.16	6.84	12.71	23.5	26.69	31.39	66.06
bmi_c		0	1634	1634	-0.051	6.84	-15.5	-4.72	-1.53	3.18	37.85

18.4 Modeling Strategies Explored Here

We are going to predict `physhealth` using `bmi_c` and `smoke100`.

- Remember that `physhealth` is a count of the number of poor physical health days in the past 30.
- As a result, `physhealth` is restricted to taking values between 0 and 30.

We will demonstrate the use of each of the following regression models, some of which are better choices than others.

1. Ordinary Least Squares (OLS) predicting `physhealth`
2. OLS predicting the logarithm of (`physhealth` + 1)
3. Poisson regression, which is appropriate for predicting counts
4. Poisson regression, adjusted to account for overdispersion
5. Negative binomial regression, also used for counts and which adjusts for overdispersion
6. Zero-inflated models, in both the Poisson and Negative Binomial varieties, which allow us to fit counts that have lots of zero values
7. A “hurdle” model, which allows us to separately fit a model to predict the incidence of “0” and then a separate model to predict the value of `physhealth` when we know it is not zero
8. Tobit regression, where a lower (and upper) bound may be set, but the underlying model describes a latent variable which can extend beyond these boundaries

18.4.1 What Will We Demonstrate?

With each approach, we will fit the model and specify procedures for doing so in R. Then we will:

1. Specify the fitted model equation
2. Interpret the model’s coefficient estimates and 95% confidence intervals around those estimates.
3. Perform a test of whether each variable adds value to the model, when the other one is already included.
4. Store the fitted values and appropriate residuals for each model.
5. Summarize the model’s apparent R^2 value, the proportion of variation explained, and the model log likelihood.
6. Perform checks of model assumptions as appropriate.
7. Describe how predictions would be made for two new subjects.
 - Harry has a BMI that is 10 kg/m^2 higher than the average across all respondents and has smoked more than 100 cigarettes in his life.
 - Sally has a BMI that is 5 kg/m^2 less than the average across all respondents and has not smoked more than 100 cigarettes in her life.

In addition, for some of the new models, we provide a little of the mathematical background, and point to other resources you can use to learn more about the model.

18.4.2 Extra Data File for Harry and Sally

To make our lives a little easier, I’ll create a little tibble containing the necessary data for Harry and Sally.

```
hs_data <- data_frame(subj = c("Harry", "Sally"),
                      bmi_c = c(10, -5),
                      smoke100 = c(1, 0))
```

`hs_data`

```
# A tibble: 2 x 3
  subj   bmi_c smoke100
  <chr>  <dbl>    <dbl>
1 Harry    10.     1.
2 Sally    -5.     0.
```

18.5 The OLS Approach

```
mod_ols1 <- lm(physhealth ~ bmi_c + smoke100,
                 data = sm_oh_A_young)

summary(mod_ols1)

Call:
lm(formula = physhealth ~ bmi_c + smoke100, data = sm_oh_A_young)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.320 -3.119 -1.715 -0.722 29.364 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 2.00443   0.20576   9.741 < 2e-16 ***
bmi_c        0.15283   0.02394   6.385 2.23e-10 ***
smoke100     2.03065   0.33978   5.976 2.80e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.612 on 1631 degrees of freedom
Multiple R-squared:  0.04711, Adjusted R-squared:  0.04595 
F-statistic: 40.32 on 2 and 1631 DF,  p-value: < 2.2e-16

confint(mod_ols1)

          2.5 %    97.5 %    
(Intercept) 1.6008443 2.4080211  
bmi_c        0.1058826 0.1997872  
smoke100     1.3642003 2.6971050
```

18.5.1 The Fitted Equation

The OLS fitted model equation is

```
physhealth = 2.00 + 0.15 bmi_c + 2.03 smoke100
```

18.5.2 Interpreting the Coefficients

- The intercept, 2.00, is the predicted `physhealth` (in days) for a subject with average BMI who has not smoked 100 cigarettes or more.
- The `bmi_c` coefficient, 0.15, indicates that for each additional kg/m² of BMI, while holding `smoke100` constant, the predicted `physhealth` value increases by 0.15 day.
- The `smoke100` coefficient, 2.03, indicates that a subject who has smoked 100 cigarettes or more has a predicted `physhealth` value 2.03 days larger than another subject with the same `bmi` but who has not smoked 100 cigarettes.

18.5.3 Testing the Predictors

We can use the t test results provided above to assess the significance of each term, after the other is already included in the model. Each p value is well below our usual α levels, so we conclude that each predictor adds statistically detectable predictive value to the model.

18.5.4 Store fitted values and residuals

We can use `broom` to do this. Here, for instance, is a table of the first six predictions and residuals.

```
sm_ols_1 <- augment(mod_ols1, sm_oh_A_young)

sm_ols_1 %>% select(physhealth, .fitted, .resid) %>% head()

  physhealth   .fitted     .resid
1          0  1.767314 -1.7673143
2          0  2.912048 -2.9120475
3          0  1.619064 -1.6190645
4          2  2.219706 -0.2197055
5          4  5.427710 -1.4277096
6          6  1.397454  4.6025461
```

It turns out that 3 of the 1634 predictions that we make are below 0, and the largest prediction made by this model is 9.82 days.

18.5.5 Specify the R^2 and log(likelihood) values

The `glance` function in the `broom` package gives us the raw and adjusted R^2 values, and the model log(likelihood), among other summaries.

```
glance(mod_ols1) %>% round(3)

  r.squared adj.r.squared sigma statistic p.value df    logLik      AIC
1     0.047         0.046 6.612     40.321      0  3 -5403.463 10814.93
      BIC deviance df.residual
1 10836.52 71303.07      1631
```

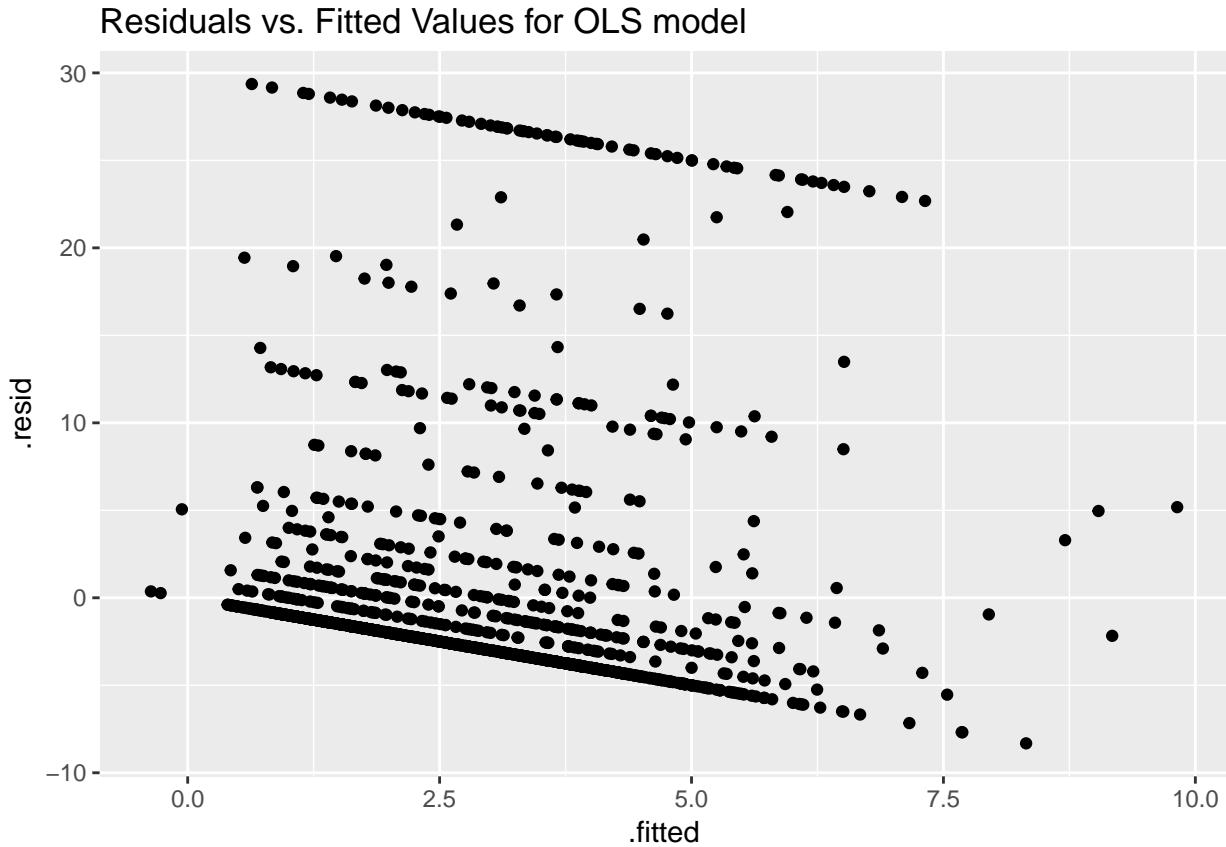
Here, we have

Model	R^2	log(likelihood)
OLS	.047	-5409.3

18.5.6 Check model assumptions

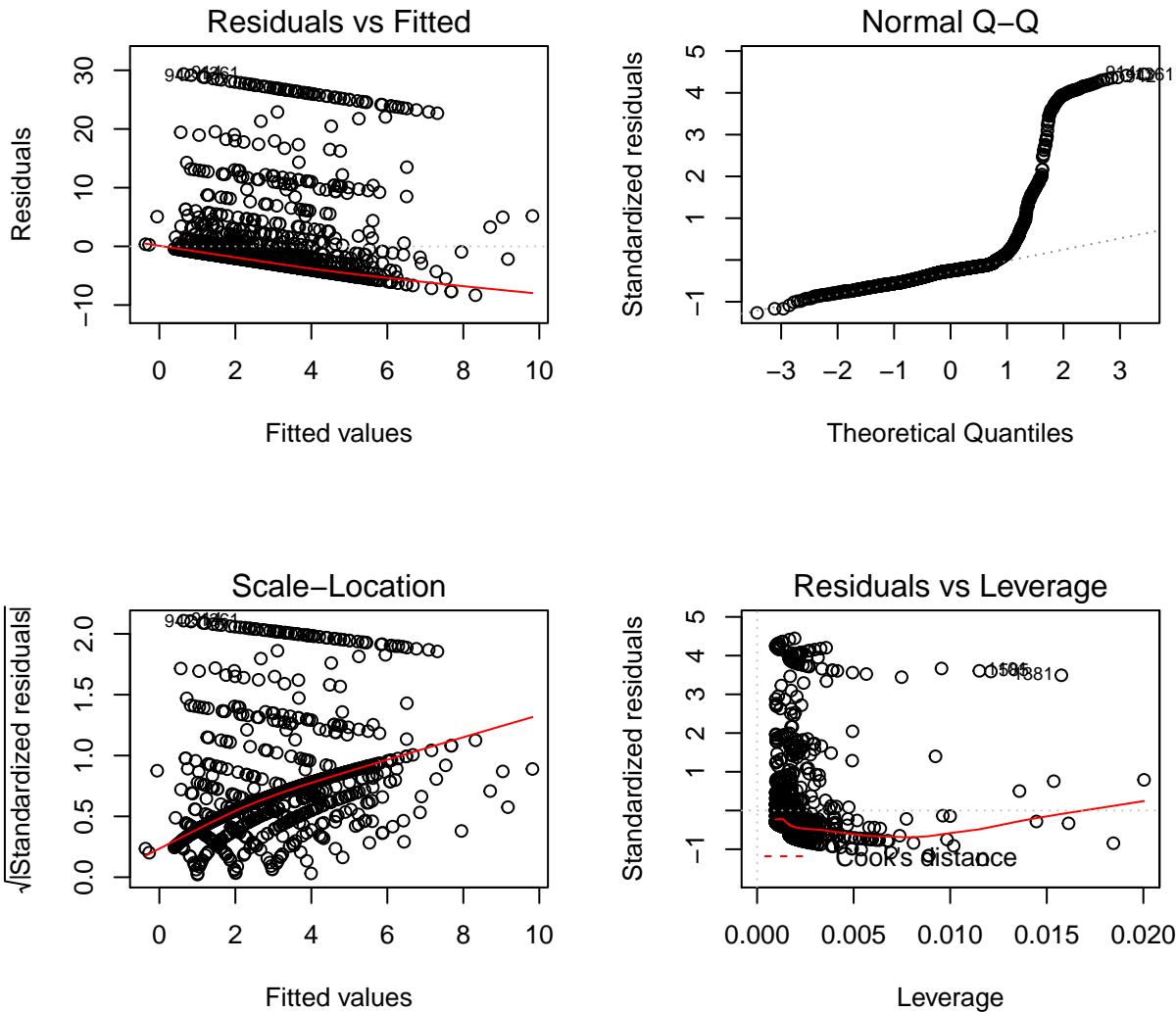
Here is a plot of the residuals vs. the fitted values for this OLS model.

```
ggplot(sm_ols_1, aes(x = .fitted, y = .resid)) +
  geom_point() +
  labs(title = "Residuals vs. Fitted Values for OLS model")
```



As usual, we can check OLS assumptions (linearity, homoscedasticity and normality) with R's complete set of residual plots.

```
par(mfrow = c(2,2))
plot(mod_ols1)
```



```
par(mfrow = c(1,1))
```

We see the problem with our residuals. They don't follow a Normal distribution.

18.5.7 Predictions for Harry and Sally

```
predict(mod_ols1, newdata = hs_data,
       interval = "prediction")
```

	fit	lwr	upr
1	5.563434	-7.423906	18.55077
2	1.240258	-11.736608	14.21712

The prediction for Harry is 5.6 days, and for Sally is 1.2 days. The prediction intervals for each include some values below 0, which is the smallest possible value.

18.5.8 Notes

- This model could have been estimated using the `ols` function in the `rms` package, as well.

```
dd <- datadist(sm_oh_A_young)
options(datadist = "dd")

(mod_ols1a <- ols(physhealth ~ bmi_c + smoke100,
                   data = sm_oh_A_young, x = TRUE, y = TRUE))
```

Linear Regression Model

```
ols(formula = physhealth ~ bmi_c + smoke100, data = sm_oh_A_young,
    x = TRUE, y = TRUE)
```

	Model Likelihood		Discrimination	
	Ratio	Test		Indexes
Obs	1634	LR chi2	78.86	R2 0.047
sigma6.6119	d.f.		2	R2 adj 0.046
d.f.	1631	Pr(> chi2)	0.0000	g 1.641

Residuals

	Min	1Q	Median	3Q	Max
	-8.320	-3.119	-1.715	-0.722	29.364

	Coef	S.E.	t	Pr(> t)
Intercept	2.0044	0.2058	9.74	<0.0001
bmi_c	0.1528	0.0239	6.38	<0.0001
smoke100	2.0307	0.3398	5.98	<0.0001

18.6 OLS model on $\log(\text{physhealth} + 1)$ days

We could try to solve the problem of fitting some predictions below 0 by log-transforming the data, so as to force values to be at least 0. Since we have undefined values when we take the log of 0, we'll add one to each of the `physhealth` values before taking logs, and then transform back when we want to make predictions.

```
mod_ols_log1 <- lm(log(physhealth + 1) ~ bmi_c + smoke100,
                     data = sm_oh_A_young)

summary(mod_ols_log1)
```

Call:

```
lm(formula = log(physhealth + 1) ~ bmi_c + smoke100, data = sm_oh_A_young)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-1.4934	-0.6119	-0.3893	0.3721	3.1911

Coefficients:

Estimate	Std. Error	t value	Pr(> t)
----------	------------	---------	----------

```
(Intercept) 0.477789  0.030053 15.898 < 2e-16 ***
bmi_c       0.026244  0.003496  7.506 9.96e-14 ***
smoke100    0.279783  0.049627  5.638 2.03e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9657 on 1631 degrees of freedom
Multiple R-squared:  0.05381, Adjusted R-squared:  0.05265
F-statistic: 46.38 on 2 and 1631 DF,  p-value: < 2.2e-16

confint(mod_ols_log1)

          2.5 %      97.5 %
(Intercept) 0.41884235 0.53673474
bmi_c       0.01938609 0.03310135
smoke100    0.182444395 0.37712164
```

18.6.1 The Fitted Equation

The equation here is

$$\log(\text{physhealth} + 1) = 0.48 + 0.026 \text{bmi_c} + 0.28 \text{smoke100}$$

18.6.2 Interpreting the Coefficients

- The intercept, 0.48, is the predicted logarithm of $(\text{physhealth} + 1)$ (in days) for a subject with average BMI who has not smoked 100 cigarettes or more.
 - We can exponentiate to see that the prediction for $(\text{physhealth} + 1)$ here is $\exp(0.48) = 1.62$ so the predicted **physhealth** for a subject with average BMI who has not smoked 100 cigarettes is 0.62 days.
- The **bmi_c** coefficient, 0.15, indicates that for each additional kg/m² of BMI, while holding **smoke100** constant, the predicted logarithm of $(\text{physhealth} + 1)$ increases by 0.026
- The **smoke100** coefficient, 0.28, indicates that a subject who has smoked 100 cigarettes or more has a predicted log of $(\text{physhealth} + 1)$ value that is 0.28 larger than another subject with the same **bmi** but who has not smoked 100 cigarettes.

18.6.3 Testing the Predictors

We are still fitting an OLS model here, so we can still use the t test results provided above to assess the significance of each term, after the other is already included in the model. Each *p* value is well below our usual α levels, so we conclude that each predictor adds statistically detectable predictive value to the model.

18.6.4 Store fitted values and residuals

We can use **broom** to help us with this. Here, for instance, is a table of the first six predictions and residuals, on the scale of our transformed response, $\log(\text{physhealth} + 1)$.

```
sm_ols_log1 <- augment(mod_ols_log1, sm_oh_A_young)

sm_ols_log1 <- sm_ols_log1 %>%
  mutate(outcome = log(physhealth + 1))

sm_ols_log1 %>%
```

```
  select(physhealth, outcome, .fitted, .resid) %>%
  head()

  physhealth  outcome   .fitted     .resid
1          0 0.000000 0.4370723 -0.4370723
2          0 0.000000 0.6336377 -0.6336377
3          0 0.000000 0.4116159 -0.4116159
4          2 1.098612 0.5147537  0.5838586
5          4 1.609438 1.0656093  0.5438286
6          6 1.945910 0.3735625  1.5723477
```

Note that the `outcome` used in this model is $\log(\text{physhealth} + 1)$, so the `.fitted` and `.resid` values react to that outcome, and not to our original `physhealth`.

Another option would be to calculate the model-predicted `physhealth`, which I'll call *ph* for a moment, with the formula:

$$ph = e^{.fitted} - 1$$

```
sm_ols_log1 <- sm_ols_log1 %>%
  mutate(pred.physhealth = exp(.fitted) - 1,
         res.physhealth = physhealth - pred.physhealth)

sm_ols_log1 %>%
  select(physhealth, pred.physhealth, res.physhealth) %>%
  head()

  physhealth pred.physhealth res.physhealth
1          0      0.5481679    -0.5481679
2          0      0.8844532    -0.8844532
3          0      0.5092545    -0.5092545
4          2      0.6732263    1.3267737
5          4      1.9026070    2.0973930
6          6      0.4529013    5.5470987
```

It turns out that 0 of the 1634 predictions that we make are below 0, and the largest prediction made by this model is 4.76 days.

18.6.5 Specify the R² and log(likelihood) values

The `glance` function in the `broom` package gives us the raw and adjusted R² values, and the model log(likelihood), among other summaries.

```
glance(mod_ols_log1) %>% round(3)

  r.squared adj.r.squared sigma statistic p.value df    logLik      AIC
1 0.054        0.053 0.966    46.376       0 3 -2260.022 4528.044
  BIC deviance df.residual
1 4549.639 1521.046      1631
```

Here, we have

	Model	Scale	R ²	log(likelihood)
	OLS on log log(physhealth + 1)		.054	-2262.0

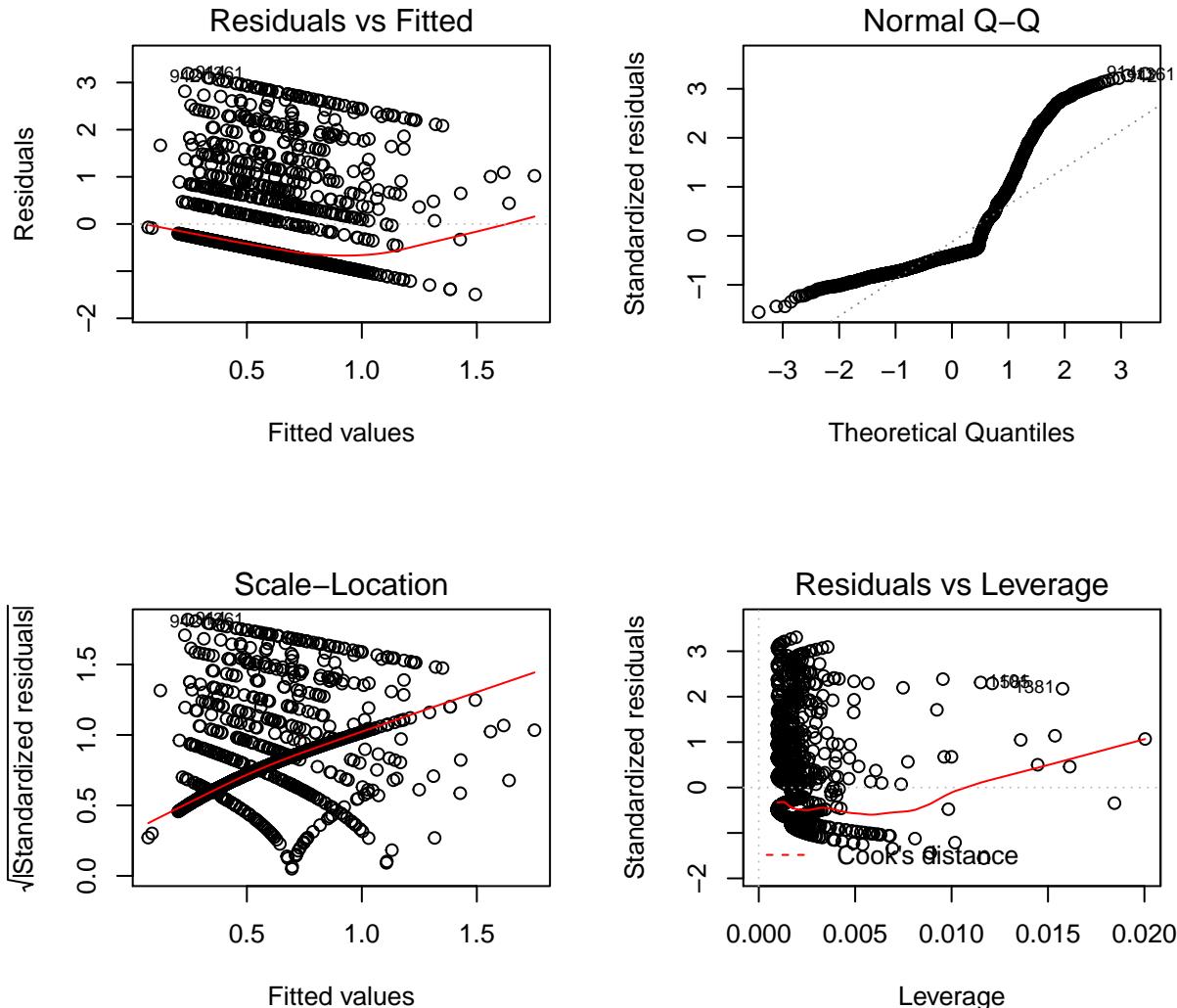
18.6.6 Getting R^2 on the scale of physhealth

We could find the correlation of our model-predicted `physhealth` values, after back-transformation, and our observed `physhealth` values, if we wanted to, and then square that to get a sort of R^2 value. But this model is not linear in `physhealth`, of course, so it's not completely comparable to our prior OLS model.

18.6.7 Check model assumptions

As usual, we can check OLS assumptions (linearity, homoscedasticity and normality) with R's complete set of residual plots. Of course, these residuals and fitted values are now on the $\log(\text{physhealth} + 1)$ scale.

```
par(mfrow = c(2,2))
plot(mod_ols_log1)
```



```
par(mfrow = c(1,1))
```

18.6.8 Predictions for Harry and Sally

```
predict(mod_ols_log1, newdata = hs_data,
       interval = "prediction", type = "response")

      fit      lwr      upr
1 1.020009 -0.8768604 2.916877
2 0.346570 -1.5487692 2.241909
```

Again, these predictions are on the $\log(\text{physhealth} + 1)$ scale, and so we have to exponentiate them, and then subtract 1, to see them on the original `physhealth` scale.

```
exp(predict(mod_ols_log1, newdata = hs_data,
            interval = "prediction", type = "response")) - 1

      fit      lwr      upr
1 1.7732184 -0.5839128 17.483482
2 0.4142084 -0.7874906  8.411281
```

The prediction for Harry is now 1.77 days, and for Sally is 0.41 days. The prediction intervals for each again include some values below 0, which is the smallest possible value.

18.7 A Poisson Regression Model

The `physhealth` data describe a count. Specifically a count of the number of days where the subject felt poorly in the last 30. Why wouldn't we model this count with linear regression?

- A count can only be positive. Linear regression would estimate some subjects as having negative counts.
- A count is unlikely to follow a Normal distribution. In fact, it's far more likely that the log of the counts will follow a Poisson distribution.

So, we'll try that. The Poisson distribution is used to model a *count* outcome - that is, an outcome with possible values (0, 1, 2, ...). The model takes a somewhat familiar form to the models we've used for linear and logistic regression¹. If our outcome is y and our linear predictors X , then the model is:

$$y_i \sim \text{Poisson}(\lambda_i)$$

The parameter λ must be positive, so it makes sense to fit a linear regression on the logarithm of this...

$$\lambda_i = \exp(\beta_0 + \beta_1 X_1 + \dots + \beta_k X_k)$$

The coefficients β can be exponentiated and treated as multiplicative effects.

We'll run a generalized linear model with a log link function, ensuring that all of the predicted values will be positive, and using a Poisson error distribution. This is called **Poisson regression**.

Poisson regression may be appropriate when the dependent variable is a count of events. The events must be independent - the occurrence of one event must not make any other more or less likely. That's hard to justify in our case, but we can take a look.

```
mod_poiss1 <- glm(physhealth ~ bmi_c + smoke100,
                    family = poisson(),
                    data = sm_oh_A_young)
```

¹This discussion is motivated by Section 6.2 of Gelman and Hill.

```
summary(mod_poiss1)
```

Call:
`glm(formula = physhealth ~ bmi_c + smoke100, family = poisson(),
 data = sm_oh_A_young)`

Deviance Residuals:

Min	1Q	Median	3Q	Max
-5.0671	-2.3418	-1.8163	-0.7165	11.4801

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	0.634600	0.022548	28.14	<2e-16 ***
bmi_c	0.043260	0.001707	25.34	<2e-16 ***
smoke100	0.704834	0.030062	23.45	<2e-16 ***

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

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 15034 on 1633 degrees of freedom
 Residual deviance: 13880 on 1631 degrees of freedom
 AIC: 15678

Number of Fisher Scoring iterations: 7

```
confint(mod_poiss1)
```

Waiting for profiling to be done...

	2.5 %	97.5 %
(Intercept)	0.59008614	0.67847727
bmi_c	0.03989052	0.04658361
smoke100	0.64596664	0.76381419

18.7.1 The Fitted Equation

The model equation is

```
log(physhealth) = 0.63 + 0.043 bmi_c + 0.71 smoke100
```

It looks like both `bmi` and `smoke_100` have confidence intervals excluding 0.

18.7.2 Interpreting the Coefficients

Our new model for y_i = counts of poor `physhealth` days in the last 30, follows the regression equation:

$$y_i \sim \text{Poisson}(\exp(0.63 + 0.043bmi_c + 0.71smoke100))$$

where `smoke100` is 1 if the subject has smoked 100 cigarettes (lifetime) and 0 otherwise, and `bmi_c` is just the centered body-mass index value in kg/m². We interpret the coefficients as follows:

- The constant term, 0.63, gives us the intercept of the regression - the prediction if `smoke100 = 0` and `bmi_c = 0`. In this case, because we've centered BMI, it implies that $\exp(0.63) = 1.88$ is the predicted days of poor `physhealth` for a non-smoker with average BMI.
- The coefficient of `bmi_c`, 0.043, is the expected difference in count of poor `physhealth` days (on the log scale) for each additional kg/m^2 of body mass index. The expected multiplicative *increase* is $e^{0.043} = 1.044$, corresponding to a 4.4% difference in the count.
- The coefficient of `smoke100`, 0.71, tells us that the predictive difference between those who have and who have not smoked 100 cigarettes can be found by multiplying the `physhealth` count by $\exp(0.71) = 2.03$, yielding essentially a doubling of the `physhealth` count.

As with linear or logistic regression, each coefficient is interpreted as a comparison where one predictor changes by one unit, while the others remain constant.

18.7.3 Testing the Predictors

We can use the Wald tests (z tests) provided with the Poisson regression output, or we can fit the model and then run an ANOVA to obtain a test based on the deviance (a simple transformation of the log likelihood ratio.)

- By the Wald tests shown above, each predictor clearly adds significant predictive value to the model given the other predictor, and we note that the *p* values are as small as R will support.
- The ANOVA approach for this model lets us check the impact of adding `smoke100` to a model already containing `bmi_c`.

```
anova(mod_poiss1, test = "Chisq")
```

Analysis of Deviance Table

Model: poisson, link: log

Response: `physhealth`

Terms added sequentially (first to last)

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL		1633	1633	15034	
<code>bmi_c</code>	1	602.50	1632	14432	< 2.2e-16 ***
<code>smoke100</code>	1	551.04	1631	13880	< 2.2e-16 ***
<hr/>					
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1					

To obtain a *p* value for `smoke100`'s impact after `bmi_c` is accounted for, we compare the difference in deviance to a chi-square distribution with 1 degree of freedom. To check the effect of `bmi_c`, we could refit the model with `bmi_c` entering last, and again run an ANOVA.

We could also run a likelihood-ratio test for each predictor, by fitting the model with and without that predictor.

```
mod_poiss1_without_bmi <- glm(physhealth ~ smoke100,
                                family = poisson(),
                                data = sm_oh_A_young)
```

```
anova(mod_poiss1, mod_poiss1_without_bmi, test = "Chisq")
```

Analysis of Deviance Table

Model 1: `physhealth ~ bmi_c + smoke100`

```
Model 2: physhealth ~ smoke100
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       1631      13880
2       1632    14426 -1   -545.19 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

18.7.4 Correcting for Overdispersion with `coeftest/coefci`

The main assumption we'll think about in a Poisson model is about **overdispersion**. We might deal with the overdispersion we see in this model by changing the nature of the tests we run within this model, using the `coeftest` or `coefci` approaches from the `lmtest` package, as I'll demonstrate next, or we might refit the model using a quasi-likelihood approach, as I'll show in the material to come.

Here, we'll use the `coeftest` and `coefci` approach from `lmtest` combined with robust sandwich estimation (via the `sandwich` package) to re-compute the Wald tests.

```
coeftest(mod_poiss1, vcov. = sandwich)
```

`z test of coefficients:`

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	0.6346000	0.0849574	7.4696	8.042e-14 ***
bmi_c	0.0432600	0.0068998	6.2697	3.617e-10 ***
smoke100	0.7048343	0.1199970	5.8738	4.260e-09 ***

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

```
coefci(mod_poiss1, vcov. = sandwich)
```

	2.5 %	97.5 %
(Intercept)	0.46808660	0.80111331
bmi_c	0.02973658	0.05678343
smoke100	0.46964440	0.94002412

Both predictors are still significant, but the standard errors are more appropriate. Later, we'll fit this approach by changing the estimation method to a quasi-likelihood approach.

18.7.5 Store fitted values and residuals

What happens if we try using the `broom` package in this case? We can, if we like, get our residuals and predicted values right on the scale of our `physhealth` response.

```
sm_poiss1 <- augment(mod_poiss1, sm_oh_A_young,
                      type.predict = "response",
                      type.residuals = "response")

sm_poiss1 %>%
  select(physhealth, .fitted, .resid) %>%
  head()

  physhealth .fitted .resid
1          0 1.763823 -1.763822711
2          0 2.438787 -2.438787228
3          0 1.691340 -1.691340221
```

```

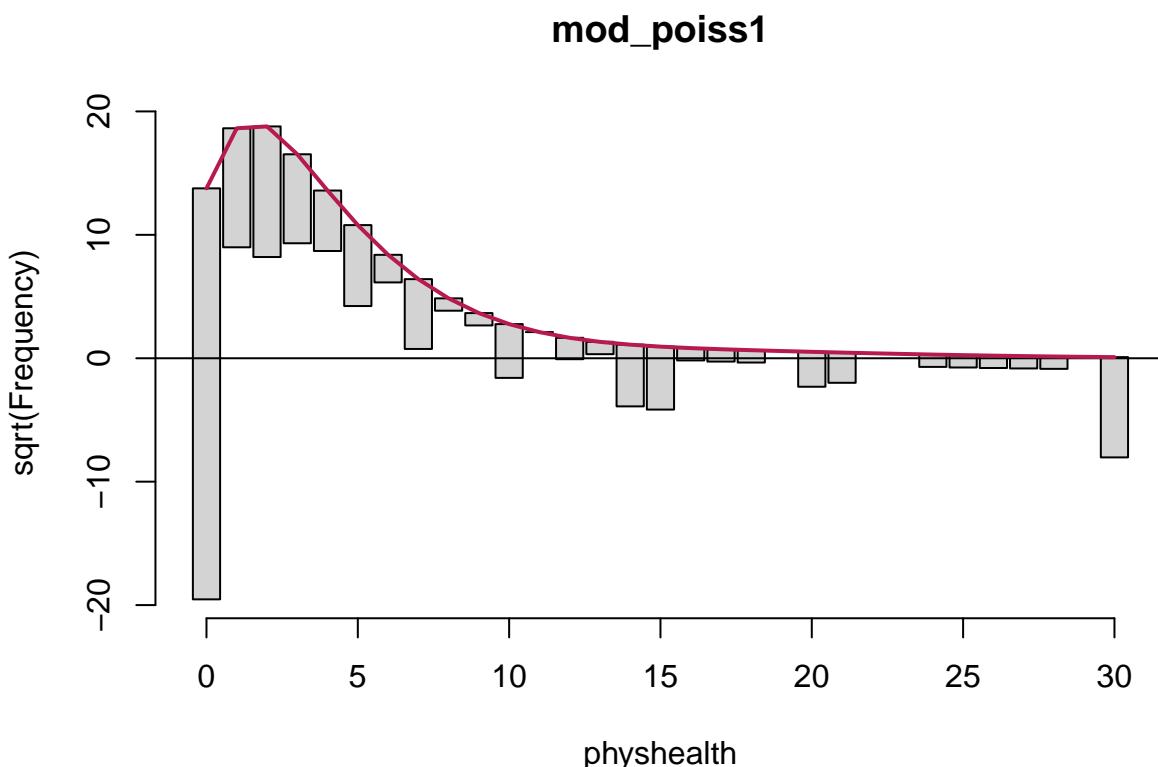
4      2 2.004777 -0.004777456
5      4 4.970699 -0.970699342
6      6 1.588506  4.411493544

```

18.7.6 Rootogram: see the fit of a count regression model

A **rootogram** is a very useful way to visualize the fit of a count regression model². The `rootogram` function in the `countr` package makes this pretty straightforward. By default, this fits a hanging rootogram on the square root of the frequencies.

```
rootogram(mod_poiss1, max = 30)
```



The red curved line is the theoretical Poisson fit. “Hanging” from each point on the red line is a bar, the height of which represents the difference between expected and observed counts. A bar hanging below 0 indicates underfitting. A bar hanging above 0 indicates overfitting. The counts have been transformed with a square root transformation to prevent smaller counts from getting obscured and overwhelmed by larger counts. We see a great deal of underfitting for counts of 0, and overfitting for most other counts, especially 1-6, with some underfitting again by `physhealth` above 14 days.

18.7.7 Specify the R² and log(likelihood) values

We can calculate the R² as the squared correlation of the fitted values and the observed values.

```

# The correlation of observed and fitted values
(poiss_r <- with(sm_poiss1, cor(physhealth, .fitted)))

```

²See <http://data.library.virginia.edu/getting-started-with-negative-binomial-regression-modeling/>

```
[1] 0.2217254
```

```
# R-square
poiss_r^2
```

```
[1] 0.04916215
```

The `glance` function in the `broom` package gives us model `log(likelihood)`, among other summaries.

```
glance(mod_poiss1) %>% round(3)
```

	null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual
1	15034.01	1633	-7835.914	15677.83	15694.02	13880.47	1631

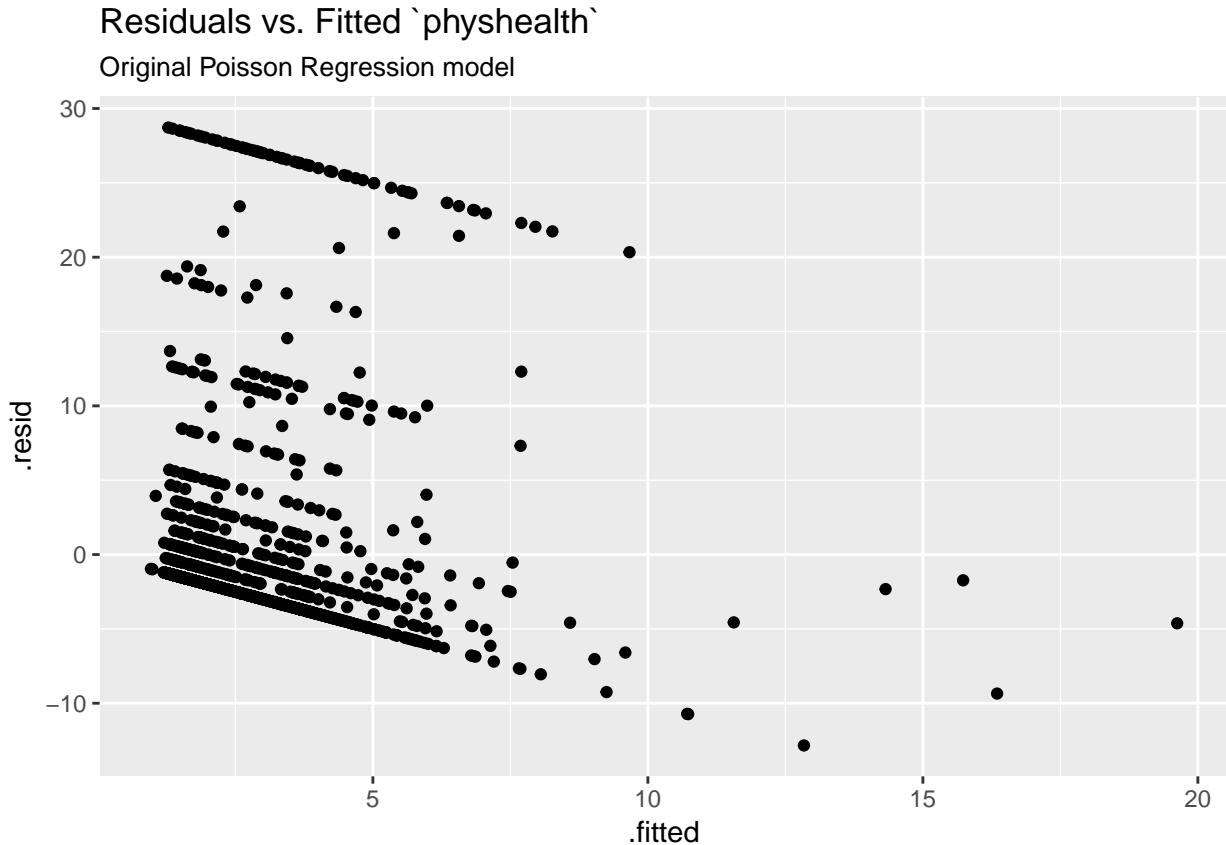
Here, we have

Model	Scale	R ²	log(likelihood)
Poisson	log(physhealth)	.049	-7841.69

18.7.8 Check model assumptions

The Poisson model is a classical generalized linear model, estimated using the method of maximum likelihood. While the default `plot` option for a `glm` still shows the plots we would use to assess the assumptions of an OLS model, we don't actually get much from that, since our Poisson model has different assumptions. It can be useful to look at a plot of residuals vs. fitted values on the original `physhealth` scale.

```
ggplot(sm_poiss1, aes(x = .fitted, y = .resid)) +
  geom_point() +
  labs(title = "Residuals vs. Fitted `physhealth`",
       subtitle = "Original Poisson Regression model")
```

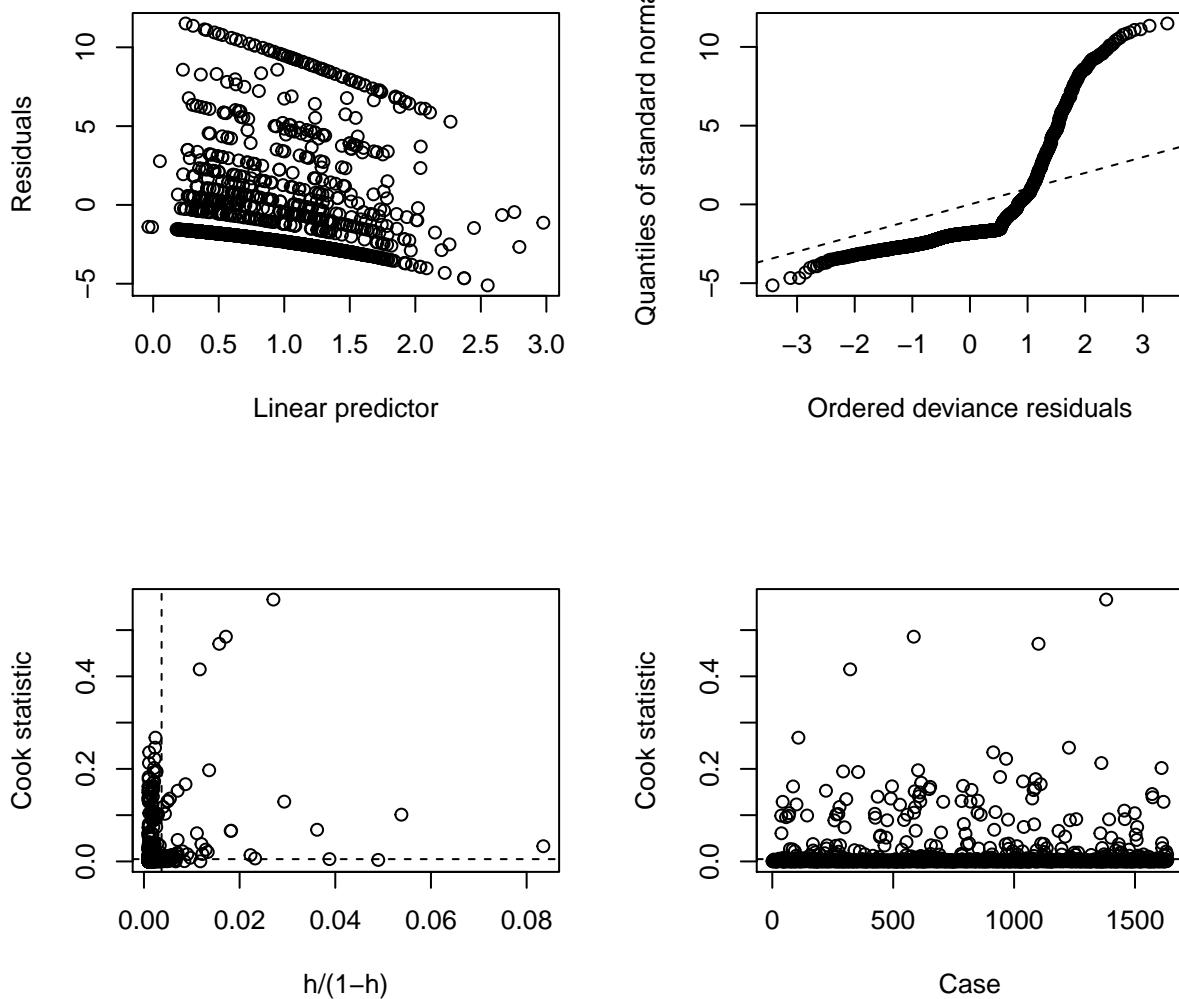


18.7.9 Using `glm.diag.plots` from the `boot` package

The `glm.diag.plots` function from the `boot` package makes a series of diagnostic plots for generalized linear models.

- (Top, Left) Jackknife deviance residuals against fitted values. This is essentially identical to what you obtain with `plot(mod_poiss1, which = 1)`. A *jackknife deviance* residual is also called a likelihood residual. It is the change in deviance when this observation is omitted from the data.
- (Top, Right) Normal Q-Q plot of standardized deviance residuals. (Dotted line shows expectation if those standardized residuals followed a Normal distribution, and these residuals generally should.) The result is similar to what you obtain with `plot(mod_poiss1, which = 2)`.
- (Bottom, Left) Cook statistic vs. standardized leverage
 - n = # of observations, p = # of parameters estimated
 - Horizontal dotted line is at $\frac{8}{n-2p}$. Points above the line have high influence on the model.
 - Vertical line is at $\frac{2p}{n-2p}$. Points to the right of the line have high leverage.
- (Bottom, Right) Index plot of Cook's statistic to help identify the observations with high influence. This is essentially the same plot as `plot(mod_poiss1, which = 4)`

```
glm.diag.plots(mod_poiss1)
```



When working with these plots, it is possible to use the `iden` command to perform some interactive identification of points in your R terminal. But that doesn't play out effectively in an HTML summary document like this, so we'll leave that out.

18.7.10 Predictions for Harry and Sally

The predictions from a `glm` fit like this don't include prediction intervals. But we can get predictions on the scale of our original response variable, `physhealth`, like this.

```
predict(mod_poiss1, newdata = hs_data, se.fit = TRUE,
       type = "response")
```

\$fit
1 2
5.882808 1.519376

```
$se.fit
```

```
1          2
0.13739453 0.03858279
```

```
$residual.scale
[1] 1
```

By using `response` as the type, these predictions fall on the original `physhealth` scale. The prediction for Harry is now 5.87 days, and for Sally is 1.52 days.

18.8 Overdispersion in a Poisson Model

Poisson regressions do not supply an independent variance parameter σ , and as a result can be overdispersed, and usually are. Under the Poisson distribution, the variance equals the mean - so the standard deviation equals the square root of the mean. The notion of **overdispersion** arises here. When fitting generalized linear models with Poisson error distributions, the residual deviance and its degrees of freedom should be approximately equal if the model fits well.

If the residual deviance is far greater than the degrees of freedom, then overdispersion may well be a problem. In this case, the residual deviance is about 8.5 times the size of the residual degrees of freedom, so that's a clear indication of overdispersion. We saw earlier that the Poisson regression model requires that the outcome (here the `physhealth` counts) be independent. A possible reason for the overdispersion we see here is that `physhealth` on different days likely do not occur independently of one another but likely "cluster" together.

18.8.1 Testing for Overdispersion?

Gelman and Hill provide an overdispersion test in R for a Poisson model as follows...

```
yhat <- predict(mod_poiss1, type = "response")
n <- arm::display(mod_poiss1)$n

glm(formula = physhealth ~ bmi_c + smoke100, family = poisson(),
    data = sm_oh_A_young)
    coef.est coef.se
(Intercept) 0.63      0.02
bmi_c        0.04      0.00
smoke100     0.70      0.03
---
n = 1634, k = 3
residual deviance = 13880.5, null deviance = 15034.0 (difference = 1153.5)

k <- arm::display(mod_poiss1)$k

glm(formula = physhealth ~ bmi_c + smoke100, family = poisson(),
    data = sm_oh_A_young)
    coef.est coef.se
(Intercept) 0.63      0.02
bmi_c        0.04      0.00
smoke100     0.70      0.03
---
n = 1634, k = 3
residual deviance = 13880.5, null deviance = 15034.0 (difference = 1153.5)

z <- (sm_oh_A_young$physhealth - yhat) / sqrt(yhat)
cat("overdispersion ratio is ", sum(z^2)/ (n - k), "\n")
```

```
overdispersion ratio is 15.49983
cat("p value of overdispersion test is ", pchisq(sum(z^2)/(n-k), n-k), "\n")
```

p value of overdispersion test is 0

The p value here is 0, indicating that the probability is essentially zero that a random variable from a χ^2 distribution with $(n - k) = 1633$ degrees of freedom would be as large as what we observed in this case.

In summary, the polyps counts are overdispersed by a factor of 15.499, which is enormous (even a factor of 2 would be considered large) and also highly statistically significant. The basic correction for overdispersion is to multiply all regression standard errors by $\sqrt{15.499} = 3.94$.

The `quasipoisson` model and the negative binomial model that we'll fit below are very similar. We write the overdispersed "quasiPoisson" model as:

$$y_i \sim \text{overdispersed Poisson}(\mu_i \exp(X_i \beta), \omega)$$

where ω is the overdispersion parameter, 15.499, in our case. The Poisson model we saw previously is then just the overdispersed Poisson model with $\omega = 1$.

18.9 Fitting the Quasi-Poisson Model

To deal with overdispersion, one useful approach is to apply a **quasi-likelihood estimation procedure**, as follows:

```
mod_poiss_od1 <- glm(physhealth ~ bmi_c + smoke100,
                      family = quasipoisson(),
                      data = sm_oh_A_young)

summary(mod_poiss_od1)

Call:
glm(formula = physhealth ~ bmi_c + smoke100, family = quasipoisson(),
     data = sm_oh_A_young)

Deviance Residuals:
    Min      1Q      Median      3Q      Max 
-5.0671 -2.3418 -1.8163 -0.7165 11.4801 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 0.634600   0.088771   7.149 1.32e-12 ***
bmi_c       0.043260   0.006722   6.436 1.61e-10 ***
smoke100    0.704834   0.118352   5.955 3.17e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasipoisson family taken to be 15.49983)

Null deviance: 15034  on 1633  degrees of freedom
Residual deviance: 13880  on 1631  degrees of freedom
AIC: NA
```

Number of Fisher Scoring iterations: 7

```
confint(mod_poiss_od1)

Waiting for profiling to be done...
```

	2.5 %	97.5 %
(Intercept)	0.4555430	0.80378993
bmi_c	0.0297236	0.05608646
smoke100	0.4734806	0.93793099

This “quasi-Poisson regression” model uses the same mean function as Poisson regression, but now estimated by quasi-maximum likelihood estimation or, equivalently, through the method of generalized estimating equations, where the inference is adjusted by an estimated dispersion parameter. Sometimes, though I won’t demonstrate this here, people fit an “adjusted” Poisson regression model, where this estimation by quasi-ML is augmented to adjust the inference via sandwich estimates of the covariances³.

18.9.1 The Fitted Equation

The model equation is still `log(physhealth) = 0.63 + 0.04 bmi_c + 0.71 smoke100`. The estimated coefficients are still statistically significant, but the standard errors for each coefficient are considerably larger when we account for overdispersion.

The dispersion parameter for the quasi-Poisson family is now taken to be a bit less than the square root of the ratio of the residual deviance and its degrees of freedom. This is a much more believable model, as a result.

18.9.2 Interpreting the Coefficients

No meaningful change from the Poisson model we saw previously.

- The constant term, 0.63, gives us the intercept of the regression - the prediction if `smoke100 = 0` and `bmi_c = 0`. In this case, because we’ve centered BMI, it implies that $\exp(0.63) = 1.88$ is the predicted days of poor `physhealth` for a non-smoker with average BMI.
- The coefficient of `bmi_c`, 0.043, is the expected difference in count of poor `physhealth` days (on the log scale) for each additional kg/m^2 of body mass index. The expected multiplicative *increase* is $e^{0.043} = 1.044$, corresponding to a 4.4% difference in the count.
- The coefficient of `smoke100`, 0.71, tells us that the predictive difference between those who have and who have not smoked 100 cigarettes can be found by multiplying the `physhealth` count by $\exp(0.71) = 2.03$, yielding essentially a doubling of the `physhealth` count.

18.9.3 Testing the Predictors

Again, we can use the Wald tests (z tests) provided with the Poisson regression output, or we can fit the model and then run an ANOVA to obtain a test based on the deviance (a simple transformation of the log likelihood ratio.)

- By the Wald tests shown above, each predictor clearly adds significant predictive value to the model given the other predictor, and we note that the *p* values are as small as R will support.
- The ANOVA approach for this model lets us check the impact of adding `smoke100` to a model already containing `bmi_c`.

```
anova(mod_poiss_od1, test = "Chisq")
```

³See Zeileis A Kleiber C Jackman S *Regression Models for Count Data in R* Vignette at <https://cran.r-project.org/web/packages/pscl/vignettes/countreg.pdf>

Analysis of Deviance Table

```
Model: quasipoisson, link: log

Response: physhealth

Terms added sequentially (first to last)

          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL           1633      15034
bmi_c         1   602.50    1632      14432 4.526e-10 ***
smoke100     1   551.04    1631      13880 2.484e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The result is unchanged. To obtain a p value for `smoke100`'s impact after `bmi_c` is accounted for, we compare the difference in deviance to a chi-square distribution with 1 degree of freedom. The result is incredibly statistically significant.

To check the effect of `bmi_c`, we could refit the model with and without `bmi_c`, and again run an ANOVA. I'll skip that here.

18.9.4 Store fitted values and residuals

What happens if we try using the `broom` package in this case? We can, if we like, get our residuals and predicted values right on the scale of our `physhealth` response.

```
sm_poiss_od1 <- augment(mod_poiss_od1, sm_oh_A_young,
                         type.predict = "response",
                         type.residuals = "response")

sm_poiss_od1 %>%
  select(physhealth, .fitted, .resid) %>%
  head()
```

	physhealth	.fitted	.resid
1	0	1.763823	-1.763822711
2	0	2.438787	-2.438787228
3	0	1.691340	-1.691340221
4	2	2.004777	-0.004777456
5	4	4.970699	-0.970699342
6	6	1.588506	4.411493544

It turns out that 0 of the 1634 predictions that we make are below 0, and the largest prediction made by this model is 19.62 days.

The `rootogram` function we've shown doesn't support overdispersed Poisson models at the moment.

18.9.5 Specify the R^2 and log(likelihood) values

We can calculate the R^2 as the squared correlation of the fitted values and the observed values.

```
# The correlation of observed and fitted values
(poiss_od_r <- with(sm_poiss_od1, cor(physhealth, .fitted)))
```

[1] 0.2217254

```
# R-square
poiss_od_r^2
```

```
[1] 0.04916215
```

The `glance` function in the `broom` package gives us model `log(likelihood)`, among other summaries.

```
glance(mod_poiss_od1) %>% round(3)
```

```
null.deviance df.null logLik AIC BIC deviance df.residual
1      15034.01     1633      NA  NA  NA 13880.47      1631
```

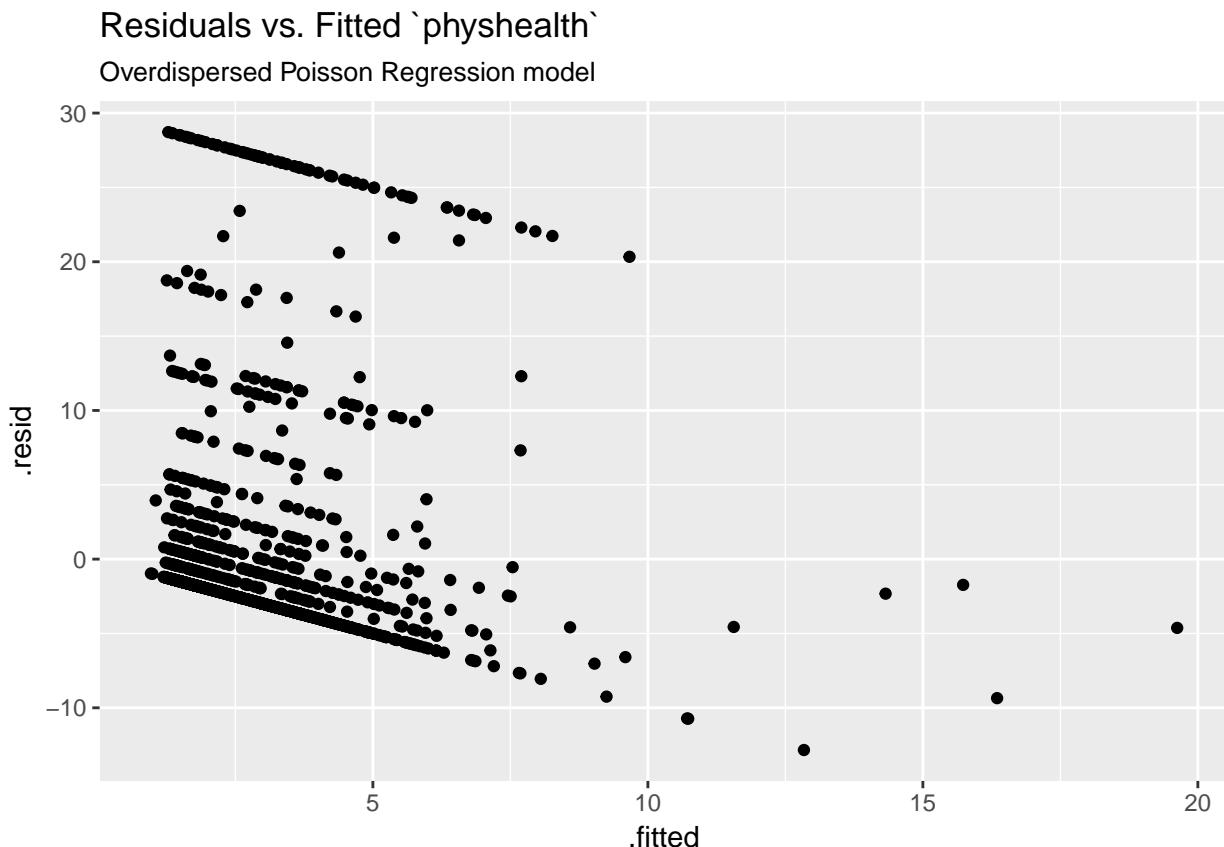
Here, we have

Model	Scale	R ²	log(likelihood)
Poisson	log(physhealth)	.049	NA

18.9.6 Check model assumptions

Having dealt with the overdispersion, this should be a cleaner model in some ways, but the diagnostics (other than the dispersion) will be the same. Here is a plot of residuals vs. fitted values on the original `physhealth` scale.

```
ggplot(sm_poiss_od1, aes(x = .fitted, y = .resid)) +
  geom_point() +
  labs(title = "Residuals vs. Fitted `physhealth`",
       subtitle = "Overdispersed Poisson Regression model")
```



I'll skip the `glm.diag.plots` results, since you've already seen them.

18.9.7 Predictions for Harry and Sally

The predictions from this overdispersed Poisson regression will match those in the original Poisson regression, but the standard error will be larger.

```
predict(mod_poiss_od1, newdata = hs_data, se.fit = TRUE,
       type = "response")
```

```
$fit
 1      2
5.882808 1.519376
```

```
$se.fit
 1      2
0.5409199 0.1518998
```

```
$residual.scale
[1] 3.936983
```

By using `response` as the type, these predictions fall on the original `physhealth` scale. Again, the prediction for Harry is 5.87 days, and for Sally is 1.52 days.

18.10 Poisson and Quasi-Poisson models using `Glm` from the `rms` package

The `Glm` function in the `rms` package can be used to fit both the original Poisson regression and the quasi-Poisson model accounting for overdispersion.

18.10.1 Refitting the original Poisson regression with `Glm`

```
d <- datadist(sm_oh_A_young)
options(datadist = "d")

mod_poi_Glm_1 <- Glm(physhealth ~ bmi_c + smoke100,
                      family = poisson(),
                      data = sm_oh_A_young,
                      x = T, y = T)

mod_poi_Glm_1
```

General Linear Model

```
Glm(formula = physhealth ~ bmi_c + smoke100, family = poisson(),
     data = sm_oh_A_young, x = T, y = T)

          Model Likelihood
          Ratio Test
  Obs    1634        LR chi2   1153.54
  Residual d.f. 1631        d.f.      2
  g 0.5185238        Pr(> chi2) <0.0001
```

	Coef	S.E.	Wald Z	Pr(> Z)
Intercept	0.6346	0.0225	28.14	<0.0001
bmi_c	0.0433	0.0017	25.34	<0.0001
smoke100	0.7048	0.0301	23.45	<0.0001

18.10.2 Refitting the overdispersed Poisson regression with `Glm`

```
d <- datadist(sm_oh_A_young)
options(datadist = "d")

mod_poi_od_Glm_1 <- glm(physhealth ~ bmi_c + smoke100,
                         family = quasipoisson(),
                         data = sm_oh_A_young,
                         x = T, y = T)

mod_poi_od_Glm_1
```

General Linear Model

```
Glm(formula = physhealth ~ bmi_c + smoke100, family = quasipoisson(),
     data = sm_oh_A_young, x = T, y = T)

      Model Likelihood
      Ratio Test
Obs    1634        LR chi2    1153.54
Residual d.f. 1631        d.f.        2
g 0.5185238        Pr(> chi2) <0.0001

      Coef    S.E.    Wald Z Pr(>|Z|)
Intercept 0.6346 0.0888 7.15    <0.0001
bmi_c      0.0433 0.0067 6.44    <0.0001
smoke100  0.7048 0.1184 5.96    <0.0001
```

The big advantage here is that we have access to the usual `ANOVA`, `summary`, and `nomogram` features that `rms` brings to fitting models.

18.10.3 ANOVA on a `Glm` fit

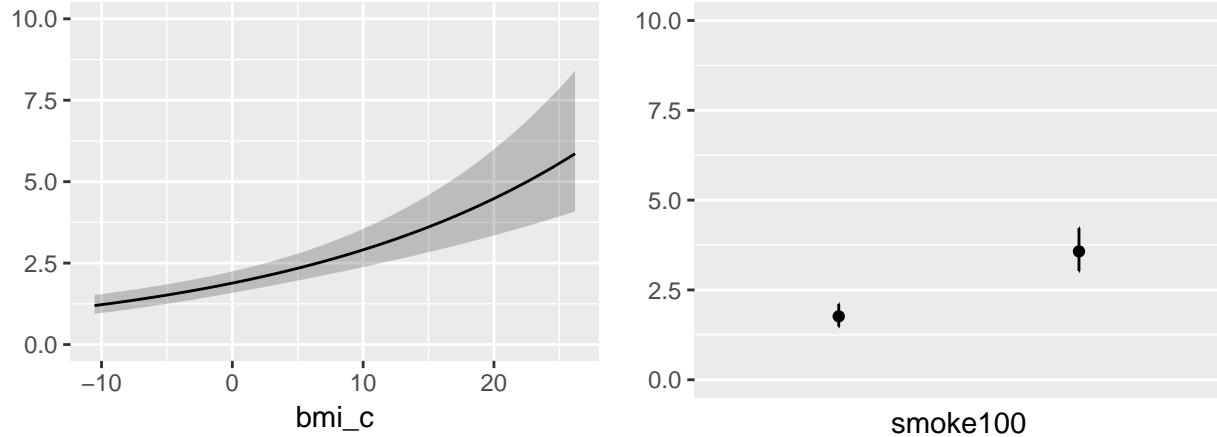
```
anova(mod_poi_od_Glm_1)
```

Wald Statistics				Response: physhealth
Factor	Chi-Square	d.f.	P	
bmi_c	41.42	1	<.0001	
smoke100	35.47	1	<.0001	
TOTAL	81.11	2	<.0001	

This shows the individual Wald χ^2 tests without having to refit the model.

18.10.4 ggplots from `Glm` fit

```
ggplot(Predict(mod_poi_od_Glm_1, fun = exp))
```



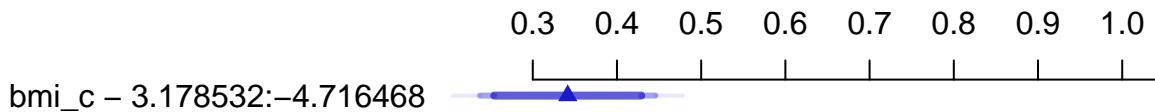
18.10.5 Summary of a `Glm` fit

```
summary(mod_poi_od_Glm_1)
```

	Effects			Response : physhealth			
Factor	Low	High	Diff. Effect	S.E.	Lower	0.95	Upper
<code>bmi_c</code>	-4.7165	3.1785	7.895	0.34154	0.05307	0.23745	0.44563
<code>smoke100</code>	0.0000	1.0000	1.000	0.70483	0.11835	0.47270	0.93697

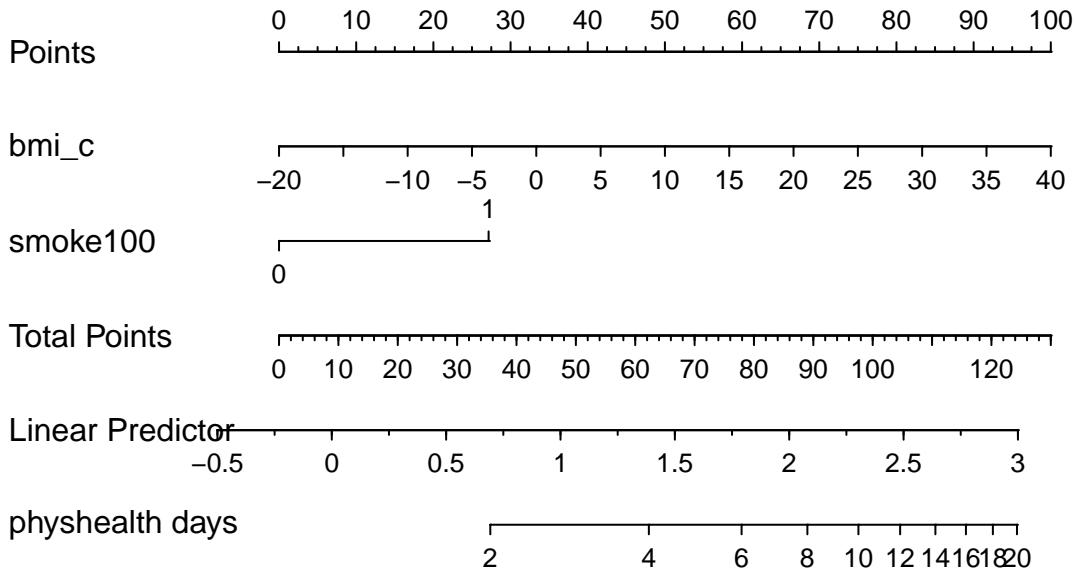
18.10.6 Plot of the Summary

```
plot(summary(mod_poi_od_Glm_1))
```



18.10.7 Nomogram of a `Glm` fit

```
plot(nomogram(mod_poi_od_Glm_1, fun = exp,
               funlabel = "physhealth days"))
```



Note the use of `fun=exp` in both the ggplot of Predict and the nomogram. What's that doing?

18.11 Negative Binomial Model

Another approach to dealing with overdispersion is to fit a negative binomial model⁴ to predict the `log(physhealth)` counts. This involves the fitting of an additional parameter, θ . That's our dispersion parameter⁵

Sometimes, people will fit a model where θ is known, for instance a geometric model (where $\theta = 1$), and then this can be directly plugged into a `glm()` fit, but the more common scenario is that we are going to iteratively estimate the β coefficients and θ . To do this, I'll use the `glm.nb` function from the `MASS` package.

```
mod_nb1 <- MASS::glm.nb(physhealth ~ bmi_c + smoke100, link = log,
                         data = sm_oh_A_young)
```

```
summary(mod_nb1)
```

Call:

```
MASS::glm.nb(formula = physhealth ~ bmi_c + smoke100, data = sm_oh_A_young,
             link = log, init.theta = 0.1343652349)
```

Deviance Residuals:

⁴See <https://cran.r-project.org/web/packages/pscl/vignettes/countreg.pdf> for more details.

⁵This θ is the inverse of the dispersion parameter estimated for these models by most other software packages, like SAS, Stata and SPSS. See <https://stats.idre.ucla.edu/r/dae/negative-binomial-regression/> for more details.

```

      Min       1Q   Median      3Q      Max
-1.1087 -0.9046 -0.8288 -0.1662  2.2323

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.61095   0.08804  6.939 3.94e-12 ***
bmi_c       0.04238   0.01010  4.198 2.70e-05 ***
smoke100    0.75724   0.14363  5.272 1.35e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(0.1344) family taken to be 1)

Null deviance: 1132.9 on 1633 degrees of freedom
Residual deviance: 1081.9 on 1631 degrees of freedom
AIC: 5171.4

Number of Fisher Scoring iterations: 1

      Theta:  0.13437
      Std. Err.:  0.00743

 2 x log-likelihood:  -5163.36000

confint(mod_nb1)

Waiting for profiling to be done...

      2.5 %     97.5 %
(Intercept) 0.44260603 0.78783368
bmi_c       0.02436096 0.06150643
smoke100    0.47867108 1.04219722

```

18.11.1 The Fitted Equation

The form of the model equation for a negative binomial regression is the same as that for Poisson regression.

```
log(physhealth) = 0.61 + 0.043 bmi_c + 0.756 smoke100
```

18.11.2 Comparison with the (raw) Poisson model

To compare the negative binomial model to the Poisson model (without the overdispersion) we can use the `logLik` function to make a comparison. Note that the Poisson model is a subset of the negative binomial.

```
logLik(mod_nb1)
```

```
'log Lik.' -2581.68 (df=4)
```

```
logLik(mod_poiss1)
```

```
'log Lik.' -7835.914 (df=3)
```

```
pchisq(2 * (logLik(mod_nb1) - logLik(mod_poiss1)), df = 1, lower.tail = FALSE)
```

```
'log Lik.' 0 (df=4)
```

Here, the difference in the log likelihoods, multiplied by 2, is 10512 with 1 degree of freedom. This strongly suggests that the negative binomial model, estimating the dispersion parameter, is more appropriate than the raw Poisson model.

However, both the regression coefficients and the standard errors are rather similar to the quasi-Poisson and the sandwich-adjusted Poisson results above. Thus, in terms of predicted means, all three models give very similar results; the associated Wald tests also lead to the same conclusions.

18.11.3 Interpreting the Coefficients

There's only a small change here from the Poisson models we saw previously.

- The constant term, 0.61, gives us the intercept of the regression - the prediction if `smoke100 = 0` and `bmi_c = 0`. In this case, because we've centered BMI, it implies that $\exp(0.61) = 1.84$ is the predicted days of poor `physhealth` for a non-smoker with average BMI.
- The coefficient of `bmi_c`, 0.043, is the expected difference in count of poor `physhealth` days (on the log scale) for each additional kg/m^2 of body mass index. The expected multiplicative *increase* is $e^{0.043} = 1.044$, corresponding to a 4.4% difference in the count.
- The coefficient of `smoke100`, 0.756, tells us that the predictive difference between those who have and who have not smoked 100 cigarettes can be found by multiplying the `physhealth` count by $\exp(0.756) = 2.13$, yielding essentially a doubling of the `physhealth` count.

18.11.4 Interpretation of Coefficients in terms of IRRs

We might be interested in looking at incident rate ratios rather than coefficients. The coefficients have an additive effect in the $\log(y)$ scale, and the IRR have a multiplicative effect in the y scale. To do this, we can exponentiate our model coefficients. This also applies to the confidence intervals.

```
exp(coef(mod_nb1))
```

	(Intercept)	bmi_c	smoke100
	1.842189	1.043292	2.132374

```
exp(confint(mod_nb1))
```

Waiting for profiling to be done...

	2.5 %	97.5 %
(Intercept)	1.556759	2.198628
bmi_c	1.024660	1.063437
smoke100	1.613928	2.835440

As an example, then, the incident rate for `smoke100 = 1` is 2.13 times the incident rate of `physhealth` days for the reference group (`smoke100 = 0`). The percent change in the incident rate of `physhealth` is a 4.3% increase for every kg/m^2 increase in centered `bmi`.

18.11.5 Testing the Predictors

Again, we can use the Wald tests (z tests) provided with the negative binomial regression output.

As an alternative, we probably should not use the standard `anova` process, because the models there don't re-estimate θ for each new model, as the warning message below indicates.

```
anova(mod_nb1)
```

Warning in anova.negbin(mod_nb1): tests made without re-estimating 'theta'

Analysis of Deviance Table

```

Model: Negative Binomial(0.1344), link: log

Response: physhealth

Terms added sequentially (first to last)

          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL           1633    1132.9
bmi_c      1   22.000    1632    1110.9 2.726e-06 ***
smoke100  1   29.004    1631    1081.9 7.224e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

So, instead, if we want, for instance to assess the significance of `bmi_c`, after `smoke100` is already included in the model, we fit both models (with and without `bmi_c`) and then compare those models with a likelihood ratio test.

```

mod_nb1_without_bmi <- MASS::glm.nb(physhealth ~ smoke100,
                                      link = log,
                                      data = sm_oh_A_young)

anova(mod_nb1, mod_nb1_without_bmi)

```

Likelihood ratio tests of Negative Binomial Models

```

Response: physhealth
          Model     theta Resid. df    2 x log-lik.    Test    df
1         smoke100 0.1300079    1632      -5186.181
2 bmi_c + smoke100 0.1343652    1631      -5163.360 1 vs 2      1
LR stat.      Pr(Chi)
1
2 22.82102 1.778106e-06

```

And we could compare the negative binomial models with and without `smoke100` in a similar way.

```

mod_nb1_without_smoke <- MASS::glm.nb(physhealth ~ bmi_c,
                                         link = log,
                                         data = sm_oh_A_young)

anova(mod_nb1, mod_nb1_without_smoke)

```

Likelihood ratio tests of Negative Binomial Models

```

Response: physhealth
          Model     theta Resid. df    2 x log-lik.    Test    df
1         bmi_c 0.1289940    1632      -5191.807
2 bmi_c + smoke100 0.1343652    1631      -5163.360 1 vs 2      1
LR stat.      Pr(Chi)
1
2 28.44694 9.630178e-08

```

18.11.6 Store fitted values and residuals

The `broom` package works in this case, too. We'll look here at residuals and predicted (fitted) values on the scale of our `physhealth` response.

```
sm_nb1 <- augment(mod_nb1, sm_oh_A_young,
                    type.predict = "response",
                    type.residuals = "response")

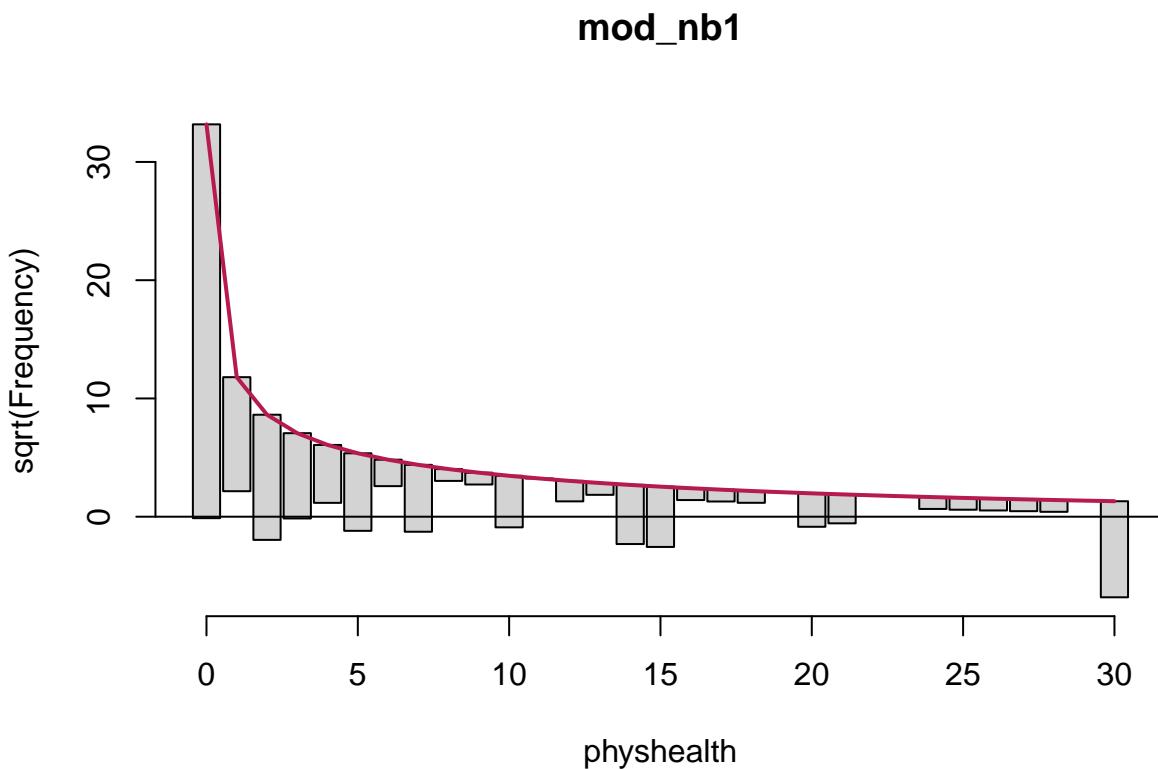
sm_nb1 %>%
  select(physhealth, .fitted, .resid) %>%
  head()

  physhealth   .fitted      .resid
1          0 1.724956 -1.72495599
2          0 2.369399 -2.36939884
3          0 1.655481 -1.65548138
4          2 1.955507  0.04449258
5          4 4.759915 -0.75991504
6          6 1.556811  4.44318950
```

18.11.7 Rootogram for Negative Binomial model

Here's the rootogram for the negative binomial model.

```
rootogram(mod_nb1, max = 30)
```



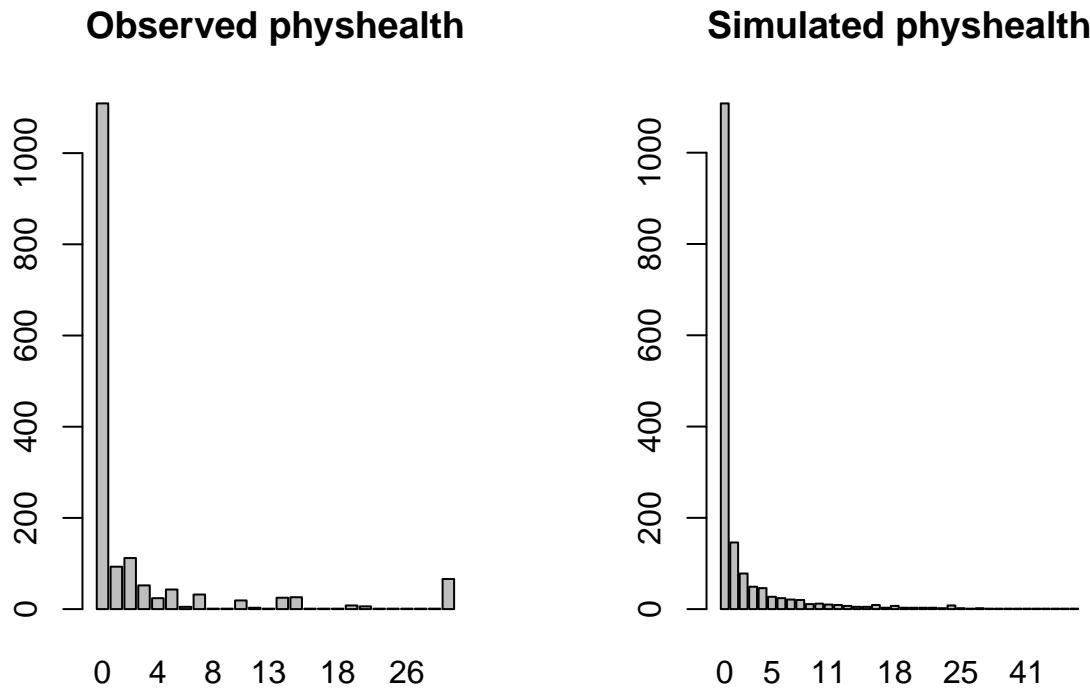
Again, the red curved line is the theoretical (negative binomial) fit. “Hanging” from each point on the red line is a bar, the height of which represents the difference between expected and observed counts. A bar hanging below 0 indicates underfitting. A bar hanging above 0 indicates overfitting. The counts have been transformed with a square root transformation to prevent smaller counts from getting obscured and overwhelmed by larger counts.

The match looks much better than the Poisson model, which is a sign that accounting for overdispersion is very important. Even this model badly underfits the number of 30 values, however.

18.11.8 Simulating what the Negative Binomial model predicts

We can use the parameters of the negative binomial model to simulate data⁶ and compare the simulated results to our observed `physhealth` data.

```
par(mfrow=c(1,2))
sm_oh_A_young$physhealth %>%
  table() %>% barplot(main = "Observed physhealth")
set.seed(432122)
rnbinom(n = nrow(sm_oh_A_young),
        size = mod_nb1$theta,
        mu = exp(coef(mod_nb1)[1])) %>%
  table() %>% barplot(main = "Simulated physhealth")
```



Again we see that the simulated data badly underfits the 30 values, and includes some predictions larger than 30.

⁶See <http://data.library.virginia.edu/getting-started-with-negative-binomial-regression-modeling/>

18.11.9 Specify the R² and log(likelihood) values

We can calculate the R² as the squared correlation of the fitted values and the observed values.

```
# The correlation of observed and fitted values
(nb_r <- with(sm_nb1, cor(physhealth, .fitted)))
```

```
[1] 0.220983
```

```
# R-square
nb_r^2
```

```
[1] 0.04883348
```

The `glance` function in the `broom` package gives us model log(likelihood), among other summaries.

```
glance(mod_nb1) %>% round(3)
```

	null.deviance	df.null	logLik	AIC	BIC	deviance	df.residual
1	1132.866	1633	-2581.68	5171.36	5192.956	1081.862	1631

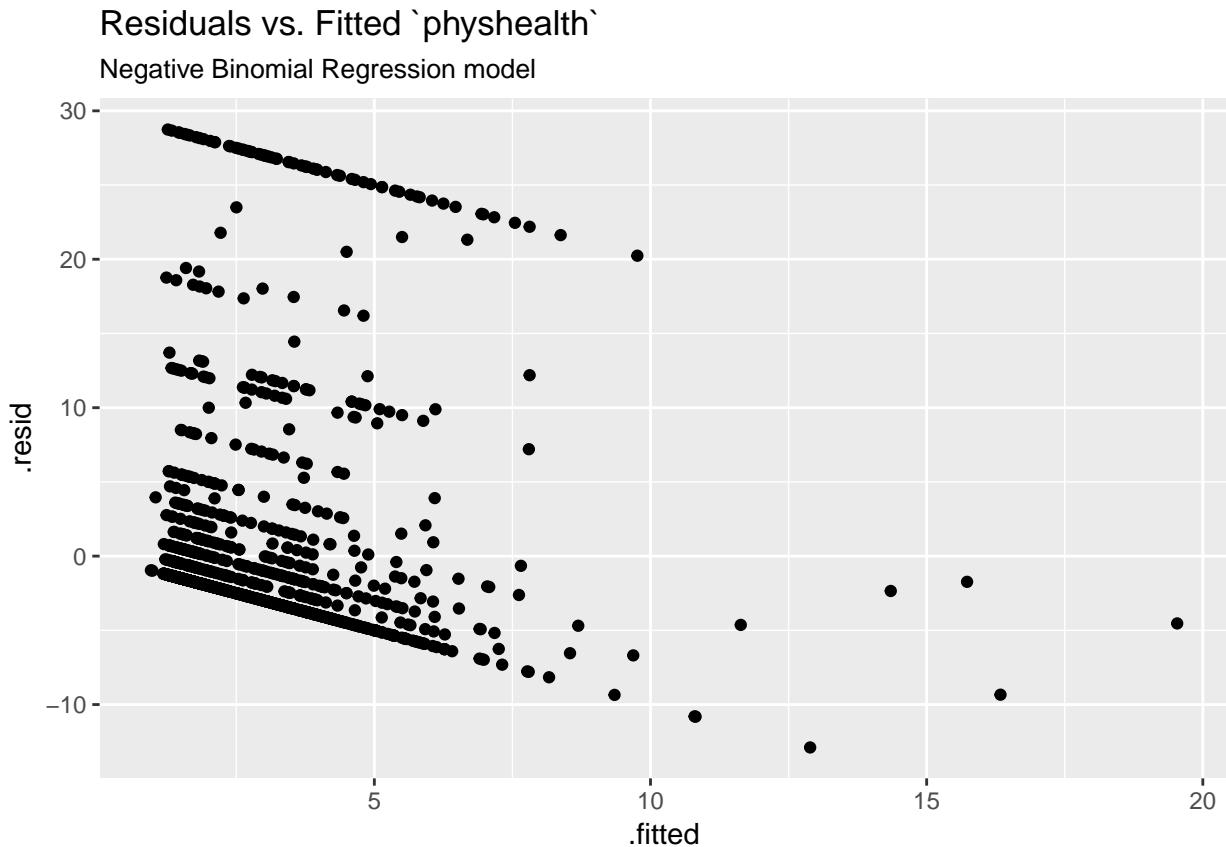
Here, we have

	Model	Scale	R ²	log(likelihood)
	Negative Binomial	log(physhealth)	.049	-2585.6

18.11.10 Check model assumptions

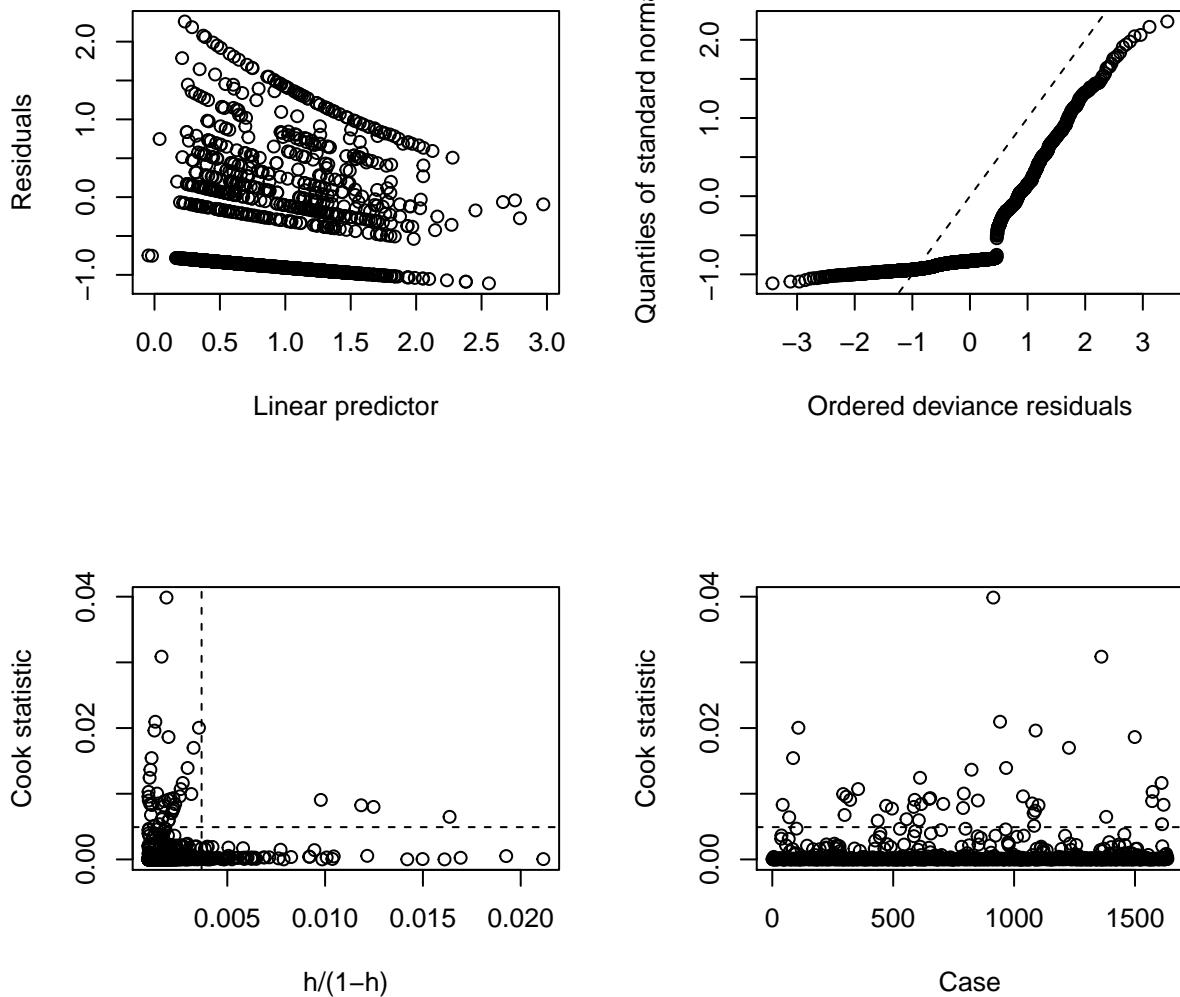
Here is a plot of residuals vs. fitted values on the original `physhealth` scale.

```
ggplot(sm_nb1, aes(x = .fitted, y = .resid)) +
  geom_point() +
  labs(title = "Residuals vs. Fitted `physhealth`",
       subtitle = "Negative Binomial Regression model")
```



Here are the `glm` diagnostic plots from the `boot` package.

```
glm.diag.plots(mod_nb1)
```



From the lower left plot, we see fewer points with large values of both Cook's distance and leverage, so that's a step in the right direction. The upper right plot still has some issues, but we're closer to a desirable result there, too.

18.11.11 Predictions for Harry and Sally

The predictions from this negative binomial regression model will be only a little different than those from the Poisson models.

```
predict(mod_nb1, newdata = hs_data, se.fit = TRUE,
       type = "response")
```

```
$fit
 1      2
6.001454 1.490406
```

```
$se.fit
```

```
1          2
0.8920031 0.1497235
```

```
$residual.scale
[1] 1
```

As we've seen in the past, when we use `response` as the type, the predictions fall on the original `physhealth` scale. The prediction for Harry is 5.99 days, and for Sally is 1.49 days.

18.12 The Problem: Too Few Zeros

Remember that we observe more than 1000 zeros in our `physhealth` data.

```
sm_oh_A_young %>% count(physhealth == 0)

# A tibble: 2 x 2
`physhealth == 0`     n
<lg1>                 <int>
1 FALSE                  525
2 TRUE                   1109
```

Let's go back to our Poisson model (without overdispersion) for a moment, and concentrate on the zero values.

```
# predict expected mean physhealth for each subject
mu <- predict(mod_poiss1, type = "response")

# sum the probabilities of a zero count for each mean
exp <- sum(dpois(x = 0, lambda = mu))

# predicted number of zeros from Poisson model
round(exp)
```

```
[1] 190
```

As we've seen previously, we're severely underfitting zero counts. We can compare the observed number of zero `physhealth` results to the expected number of zero values from the likelihood-based models.

```
round(c("Obs" = sum(sm_oh_A_young$physhealth == 0),
      "Poisson" = sum(dpois(0, fitted(mod_poiss1))),
      "NB" = sum(dnbinom(0, mu = fitted(mod_nb1), size = mod_nb1$theta))),0)

Obs Poisson      NB
1109      190      1101
```

There are at least two ways to tackle this problem.

- Fitting a model which deliberately inflates the number of zeros that are fitted
- Fitting a hurdle model

We'll look at those options, next.

18.13 The Zero-Inflated Poisson Regression Model

The zero-inflated Poisson or (ZIP) model is used to describe count data with an excess of zero counts⁷. The model posits that there are two processes involved:

⁷See <https://stats.idre.ucla.edu/r/dae/zip/> for more on the zero-inflated poisson model.

- a logit model is used to predict excess zeros
- while a Poisson model is used to predict the counts, generally

The `pscl` package is used here, which can conflict with the `countreg` package we used to fit rootograms. That's why I'm loading it here.

```
library(pscl)
```

To run the zero-inflated Poisson model, we use the following:

```
mod_zip1 <- zeroinfl(physhealth ~ bmi_c + smoke100,
                      data = sm_oh_A_young)

summary(mod_zip1)
```

Call:

```
zeroinfl(formula = physhealth ~ bmi_c + smoke100, data = sm_oh_A_young)
```

Pearson residuals:

Min	1Q	Median	3Q	Max
-1.4159	-0.6461	-0.5315	-0.2562	11.0564

Count model coefficients (poisson with log link):

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.897651	0.022696	83.613	<2e-16 ***
bmi_c	0.013984	0.001698	8.235	<2e-16 ***
smoke100	0.437642	0.030107	14.536	<2e-16 ***

Zero-inflation model coefficients (binomial with logit link):

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	0.921514	0.069911	13.181	< 2e-16 ***
bmi_c	-0.050103	0.007793	-6.430	1.28e-10 ***
smoke100	-0.416061	0.110166	-3.777	0.000159 ***

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

Number of iterations in BFGS optimization: 14

Log-likelihood: -4178 on 6 Df

```
confint(mod_zip1)
```

	2.5 %	97.5 %
count_(Intercept)	1.85316852	1.94213386
count_bmi_c	0.01065621	0.01731264
count_smoke100	0.37863312	0.49664993
zero_(Intercept)	0.78449191	1.05853706
zero_bmi_c	-0.06537671	-0.03482990
zero_smoke100	-0.63198306	-0.20013969

The output describes two separate regression models. Below the model call, we see information on a Poisson regression model. Then we see another block describing the inflation model.

Each predictor (`bmi_c` and `smoke100`) appears to be statistically significant in each part of the model.

18.13.1 Comparison to a null model

To show that this model fits better than the null model (the model with intercept only), we can compare them directly with a chi-squared test. Since we have two predictors in the full model, the degrees of freedom for this test is 2.

```
mod_zipnull <- zeroinfl(physhealth ~ 1,
                         data = sm_oh_A_young)

summary(mod_zipnull)

Call:
zeroinfl(formula = physhealth ~ 1, data = sm_oh_A_young)

Pearson residuals:
    Min     1Q   Median     3Q    Max
-0.6355 -0.6355 -0.6355 -0.1720  6.3162

Count model coefficients (poisson with log link):
  Estimate Std. Error z value Pr(>|z|)
(Intercept) 2.14401   0.01495 143.4   <2e-16 ***
Zero-inflation model coefficients (binomial with logit link):
  Estimate Std. Error z value Pr(>|z|)
(Intercept) 0.74753   0.05298 14.11   <2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 9
Log-likelihood: -4351 on 2 Df
pchisq(2 * (logLik(mod_zip1) - logLik(mod_zipnull)), df = 2, lower.tail = FALSE)

'log Lik.' 1.470287e-75 (df=6)
```

18.13.2 Comparison to a Poisson Model with the Vuong test

```
vuong(mod_zip1, mod_poiss1)

Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
null that the models are indistinguishable)
-----
          Vuong z-statistic      H_A      p-value
Raw              15.54767 model1 > model2 < 2.22e-16
AIC-corrected    15.53492 model1 > model2 < 2.22e-16
BIC-corrected    15.50049 model1 > model2 < 2.22e-16
```

Certainly, the ZIP model is a significant improvement over the standard Poisson model, as the Vuong test reveals.

18.13.3 The Fitted Equation

The form of the model equation for a zero-inflated Poisson regression requires us to take two separate models into account. First we have a logistic regression model to predict the log odds of zero `physhealth` days...

```
logit(physhealth = 0) = 0.919 - 0.051 bmi_c - 0.411 smoke100
```

That takes care of the *extra* zeros. Then, to predict the number of `physhealth` days, we have:

```
log(physhealth) = 1.90 + 0.014 bmi_c + 0.441 smoke100
```

which may produce some additional zero count estimates.

18.13.4 Interpreting the Coefficients

We can exponentiate the logistic regression coefficients to obtain results in terms of odds ratios for that model, and that can be of some help in understanding the process behind excess zeros.

Also, exponentiating the coefficients of the count model help us describe those counts on the original scale of `physhealth`.

```
exp(coef(mod_zip1))
```

	count_(Intercept)	count_bmi_c	count_smoke100	zero_(Intercept)
	6.6702090	1.0140827	1.5490495	2.5130936
zero_bmi_c		zero_smoke100		
	0.9511312		0.6596398	

For example,

- in the model for `physhealth` = 0, the odds of `physhealth` = 0 are 66% as high for subjects with `smoke100` = 1 as for non-smokers with the same BMI.
- in the Poisson model for `physhealth`, the `physhealth` count is estimated to increase by 1.55 for smokers as compared to non-smokers with the same BMI.

18.13.5 Testing the Predictors

We can test the model with and without `bmi_c`, for example, by fitting the model both ways, and comparing the results with either a Wald or Likelihood Ratio test, each of which is available in the `lmtest` package.

```
mod_zip1_nobmi <- zeroinfl(physhealth ~ smoke100,
                             data = sm_oh_A_young)
```

```
lmtest::waldtest(mod_zip1, mod_zip1_nobmi)
```

Wald test

```
Model 1: physhealth ~ bmi_c + smoke100
Model 2: physhealth ~ smoke100
  Res.Df Df Chisq Pr(>Chisq)
1     1628
2     1630 -2 109.3 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
lmtest::lrtest(mod_zip1, mod_zip1_nobmi)
```

Likelihood ratio test

```

Model 1: physhealth ~ bmi_c + smoke100
Model 2: physhealth ~ smoke100
#Df LogLik Df Chisq Pr(>Chisq)
1   6 -4178.5
2   4 -4232.0 -2 106.96 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

18.13.6 Store fitted values and residuals

The `broom` package does not work with the `zeroinfl` tool. So we need to build up the fitted values and residuals ourselves.

```

sm_zip1 <- sm_oh_A_young %>%
  mutate(fitted = fitted(mod_zip1, type = "response"),
         resid = resid(mod_zip1, type = "response"))

sm_zip1 %>%
  dplyr::select(physhealth, fitted, resid) %>%
  head()

# A tibble: 6 x 3
  physhealth fitted    resid
  <int>     <dbl>    <dbl>
1          0     1.76 -1.76
2          0     2.53 -2.53
3          0     1.67 -1.67
4          2     2.04 -0.0357
5          4     5.02 -1.02
6          6     1.55  4.45

```

18.13.7 Modeled Number of Zero Counts

The zero-inflated model is designed to perfectly match the number of observed zeros. We can compare the observed number of zero `physhealth` results to the expected number of zero values from the likelihood-based models.

```

round(c("Obs" = sum(sm_oh_A_young$physhealth == 0),
      "Poisson" = sum(dpois(0, fitted(mod_pois1))),
      "NB" = sum(dnbinom(0, mu = fitted(mod_nb1), size = mod_nb1$theta)),
      "ZIP" = sum(predict(mod_zip1, type = "prob")[,1])), 0)

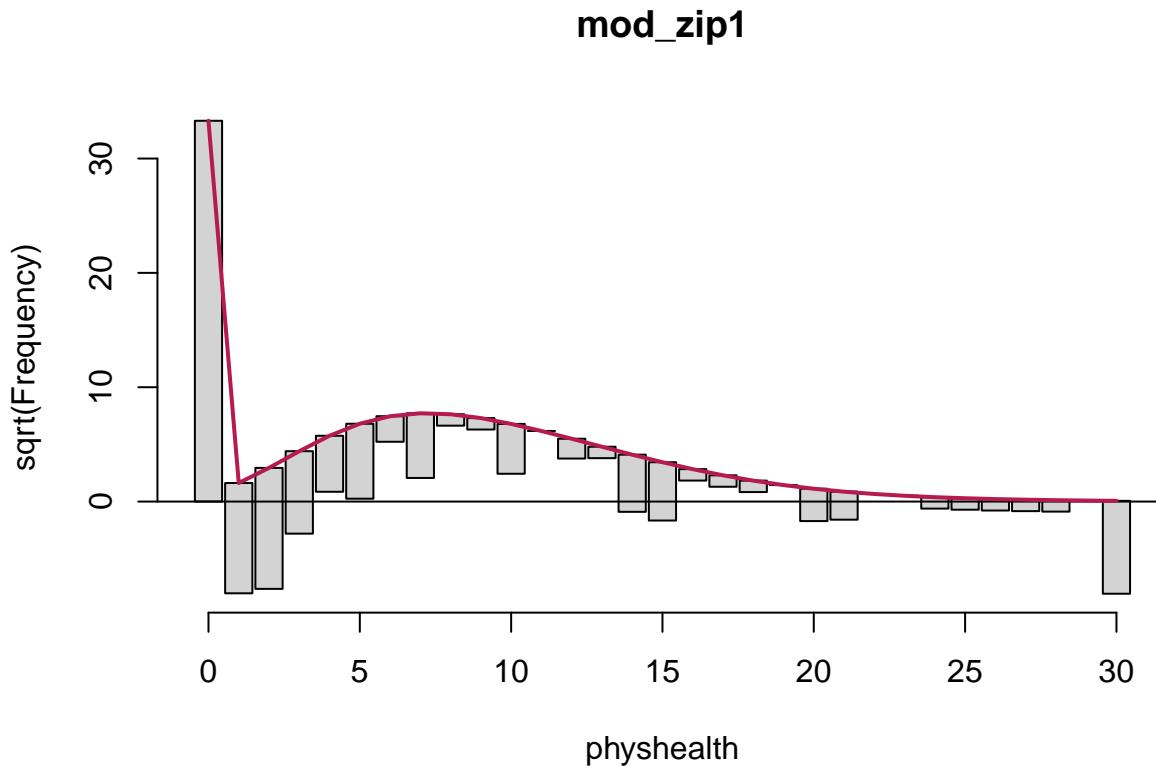
```

Obs	Poisson	NB	ZIP
1109	190	1101	1109

18.13.8 Rootogram for ZIP model

Here's the rootogram for the zero-inflated Poisson model.

```
countreg::rootogram(mod_zip1, max = 30)
```



The zero frequencies are perfectly matched here, but we can see that counts of 1 and 2 are now substantially underfit, and values between 6 and 13 are overfit.

18.13.9 Specify the R^2 and log (likelihood) values

We can calculate a proxy for R^2 as the squared correlation of the fitted values and the observed values.

```
# The correlation of observed and fitted values
(zip_r <- with(sm_zip1, cor(physhealth, fitted)))
```

```
[1] 0.2236014
```

```
# R-square
zip_r^2
```

```
[1] 0.04999758
```

```
logLik(mod_zip1)
```

```
'log Lik.' -4178.496 (df=6)
```

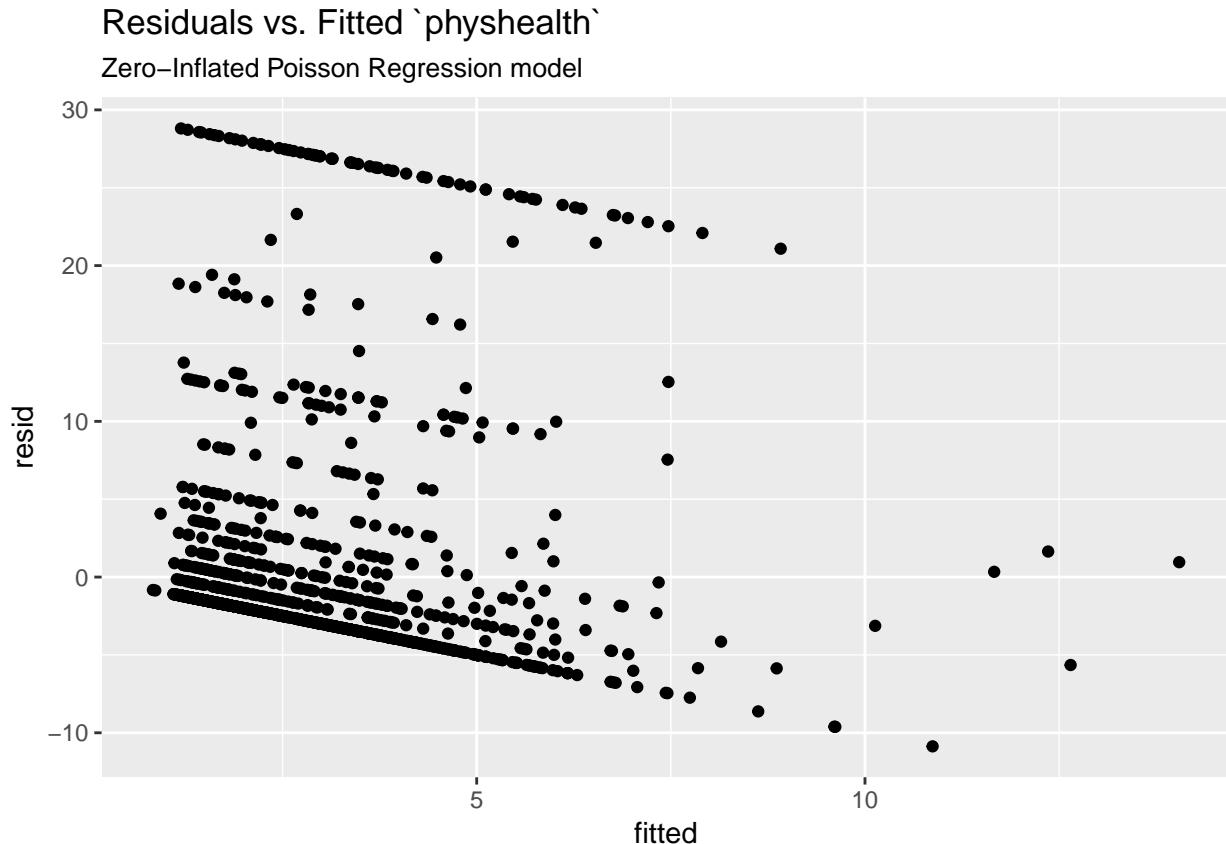
Here, we have

Model	Scale	R^2	log(likelihood)
Zero-Inflated Poisson	Complex: log(physhealth)	.050	-4184.1

18.13.10 Check model assumptions

Here is a plot of residuals vs. fitted values on the original `physhealth` scale.

```
ggplot(sm_zip1, aes(x = fitted, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs. Fitted `physhealth`",
       subtitle = "Zero-Inflated Poisson Regression model")
```



18.13.11 Predictions for Harry and Sally

The predictions from this ZIP regression model are obtained as follows...

```
predict(mod_zip1, newdata = hs_data, type = "response")
```

1	2
5.928536	1.470896

As we've seen in the past, when we use `response` as the type, the predictions fall on the original `physhealth` scale. The prediction for Harry is 5.92 days, and for Sally is 1.47 days.

18.14 The Zero-Inflated Negative Binomial Regression Model

As an alternative to the ZIP model, we might consider a zero-inflated negative binomial regression⁸. This will involve a logistic regression to predict the probability of a 0, and then a negative binomial model to

⁸See <https://stats.idre.ucla.edu/r/dae/zinb/>

describe the counts of `physhealth`.

To run the zero-inflated negative binomial model, we use the following code:

```
mod_zinb1 <- zeroinfl(physhealth ~ bmi_c + smoke100,
                        dist = "negbin", data = sm_oh_A_young)

summary(mod_zinb1)
```

Call:

```
zeroinfl(formula = physhealth ~ bmi_c + smoke100, data = sm_oh_A_young,
         dist = "negbin")
```

Pearson residuals:

Min	1Q	Median	3Q	Max
-0.5432	-0.3914	-0.3471	-0.1542	7.7462

Count model coefficients (negbin with log link):

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.353151	0.126131	10.728	< 2e-16 ***
bmi_c	0.020419	0.007825	2.609	0.00907 **
smoke100	0.573530	0.135112	4.245	2.19e-05 ***
Log(theta)	-1.028636	0.172175	-5.974	2.31e-09 ***

Zero-inflation model coefficients (binomial with logit link):

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	0.04144	0.20698	0.200	0.8413
bmi_c	-0.06160	0.01354	-4.550	5.37e-06 ***
smoke100	-0.34373	0.17023	-2.019	0.0435 *

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

Theta = 0.3575

Number of iterations in BFGS optimization: 25

Log-likelihood: -2564 on 7 Df

```
confint(mod_zinb1)
```

	2.5 %	97.5 %
count_(Intercept)	1.105938639	1.60036316
count_bmi_c	0.005081859	0.03575643
count_smoke100	0.308714260	0.83834496
zero_(Intercept)	-0.364229869	0.44711077
zero_bmi_c	-0.088141789	-0.03506528
zero_smoke100	-0.677368528	-0.01009594

18.14.1 Comparison to a null model

To show that this model fits better than the null model (the model with intercept only), we can compare them directly with a chi-squared test. Since we have two predictors in the full model, the degrees of freedom for this test is 2.

```
mod_zinbnnull <- zeroinfl(physhealth ~ 1, dist = "negbin",
                           data = sm_oh_A_young)
```

```
summary(mod_zinbnull)

Call:
zeroinfl(formula = physhealth ~ 1, data = sm_oh_A_young, dist = "negbin")

Pearson residuals:
    Min     1Q   Median     3Q    Max 
-0.3700 -0.3700 -0.3700 -0.1002  3.6775 

Count model coefficients (negbin with log link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  1.5356    0.1499 10.246 < 2e-16 ***
Log(theta)   -1.3053    0.2330 -5.603  2.1e-08 ***

Zero-inflation model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -0.3660    0.3352 -1.092   0.275    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Theta = 0.2711
Number of iterations in BFGS optimization: 13
Log-likelihood: -2603 on 3 Df
pchisq(2 * (logLik(mod_nb1) - logLik(mod_zinbnull)), df = 2, lower.tail = FALSE)

'log Lik.' 4.170115e-10 (df=4)
```

18.14.2 Comparison to a Negative Binomial Model: Vuong test

```
vuong(mod_zinb1, mod_nb1)

Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
null that the models are indistinguishable)
-----
          Vuong z-statistic      H_A      p-value
Raw           3.113058 model1 > model2 0.0009258
AIC-corrected 2.574484 model1 > model2 0.0050195
BIC-corrected 1.120661 model1 > model2 0.1312160
```

The zero-inflated negative binomial model is a significant improvement over the standard negative binomial model according to the raw or AIC-corrected Vuong tests, but not according to the BIC-corrected test.

18.14.3 The Fitted Equation

Like the ZIP, the zero-inflated negative binomial regression also requires us to take two separate models into account. First we have a logistic regression model to predict the log odds of zero `physhealth` days...

```
logit(physhealth = 0) = 0.044 - 0.062 bmi_c - 0.337 smoke100
```

That takes care of the *extra* zeros. Then, to predict the number of `physhealth` days, we have:

```
log(physhealth) = 1.36 + 0.020 bmi_c + 0.575 smoke100
```

and there's a θ term, which is estimated as $\exp(-1.023)$ or 0.36, and this negative binomial regression model may also produce some additional zero count estimates.

18.14.4 Interpreting the Coefficients

As with the zip, we can exponentiate the logistic regression coefficients to obtain results in terms of odds ratios for that model, and that can be of some help in understanding the process behind excess zeros.

```
exp(coef(mod_zinb1))
```

count_(Intercept)	count_bmi_c	count_smoke100	zero_(Intercept)
3.8695991	1.0206290	1.7745194	1.0423111
zero_bmi_c	zero_smoke100		
0.9402556	0.7091188		

For example,

- in the model for `physhealth = 0`, the odds of `physhealth = 0` are 71.4% as high for subjects with `smoke100 = 1` as for non-smokers with the same BMI.

Interpreting the negative binomial piece works the same way as it did in the negative binomial regression.

18.14.5 Testing the Predictors

We can test the model with and without `bmi_c`, for example, by fitting the model both ways, and comparing the results with either a Wald or Likelihood Ratio test, each of which is available in the `lmtest` package.

```
mod_zinb1_nobmi <- zeroinfl(physhealth ~ smoke100,
                               dist = "negbin",
                               data = sm_oh_A_young)
```

```
lmtest::waldtest(mod_zinb1, mod_zinb1_nobmi)
```

Wald test

```
Model 1: physhealth ~ bmi_c + smoke100
Model 2: physhealth ~ smoke100
  Res.Df Df  Chisq Pr(>Chisq)
1   1627
2   1629 -2 33.611  5.028e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
lmtest::lrtest(mod_zinb1, mod_zinb1_nobmi)
```

Likelihood ratio test

```
Model 1: physhealth ~ bmi_c + smoke100
Model 2: physhealth ~ smoke100
  #Df LogLik Df  Chisq Pr(>Chisq)
1    7 -2564.3
2    5 -2587.1 -2 45.442  1.357e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

18.14.6 Store fitted values and residuals

Again, we need to build up the fitted values and residuals without the `broom` package.

```
sm_zinb1 <- sm_oh_A_young %>%
  mutate(fitted = fitted(mod_zinb1, type = "response"),
        resid = resid(mod_zinb1, type = "response"))

sm_zip1 %>%
  dplyr::select(physhealth, fitted, resid) %>%
  head()

# A tibble: 6 x 3
  physhealth fitted   resid
  <int>     <dbl>   <dbl>
1          0     1.76 -1.76
2          0     2.53 -2.53
3          0     1.67 -1.67
4          2     2.04 -0.0357
5          4     5.02 -1.02
6          6     1.55  4.45
```

18.14.7 Modeled Number of Zero Counts

Once again, we can compare the observed number of zero `physhealth` results to the expected number of zero values from the likelihood-based models.

```
round(c("Obs" = sum(sm_oh_A_young$physhealth == 0),
  "Poisson" = sum(dpois(0, fitted(mod_poiss1))),
  "NB" = sum(dnbinom(0, mu = fitted(mod_nb1), size = mod_nb1$theta)),
  "ZIP" = sum(predict(mod_zip1, type = "prob")[,1]),
  "ZINB" = sum(predict(mod_zinb1, type = "prob")[,1])),0)
```

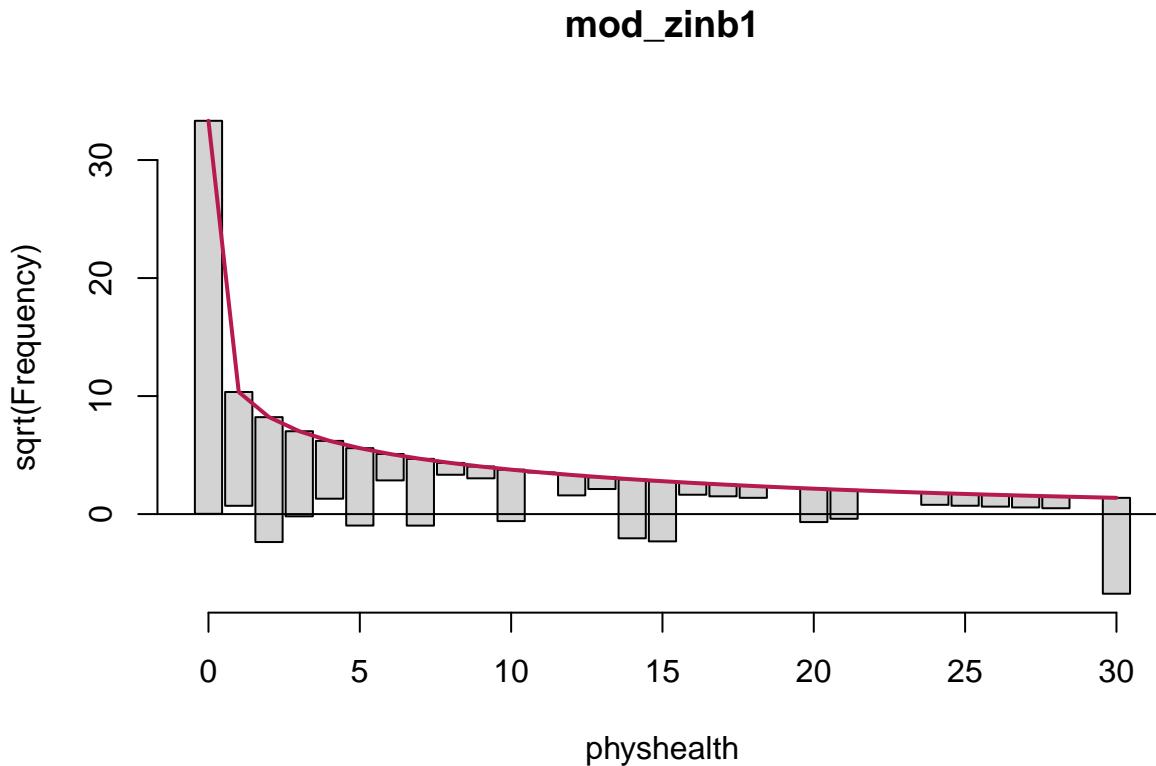
Obs	Poisson	NB	ZIP	ZINB
1109	190	1101	1109	1111

So, the Poisson model is clearly inappropriate, but the zero-inflated (Poisson and NB) and the negative binomial model all give reasonable fits in this regard.

18.14.8 Rootogram for Zero-Inflated Negative Binomial model

Here's the rootogram for the zero-inflated negative binomial model.

```
countreg::rootogram(mod_zinb1, max = 30)
```



As in the ZIP model, the zero frequencies are perfectly matched here, but we can see that counts of 1 and 2 are now closer to the data we observe than in the ZIP model. We are still substantially underfitting values of 30.

18.14.9 Specify the R^2 and log (likelihood) values

We can calculate a proxy for R^2 as the squared correlation of the fitted values and the observed values.

```
# The correlation of observed and fitted values
(zinb_r <- with(sm_zinb1, cor(physhealth, fitted)))
```

```
[1] 0.2218276
```

```
# R-square
zinb_r^2
```

```
[1] 0.04920749
```

```
logLik(mod_zinb1)
```

```
'log Lik.' -2564.34 (df=7)
```

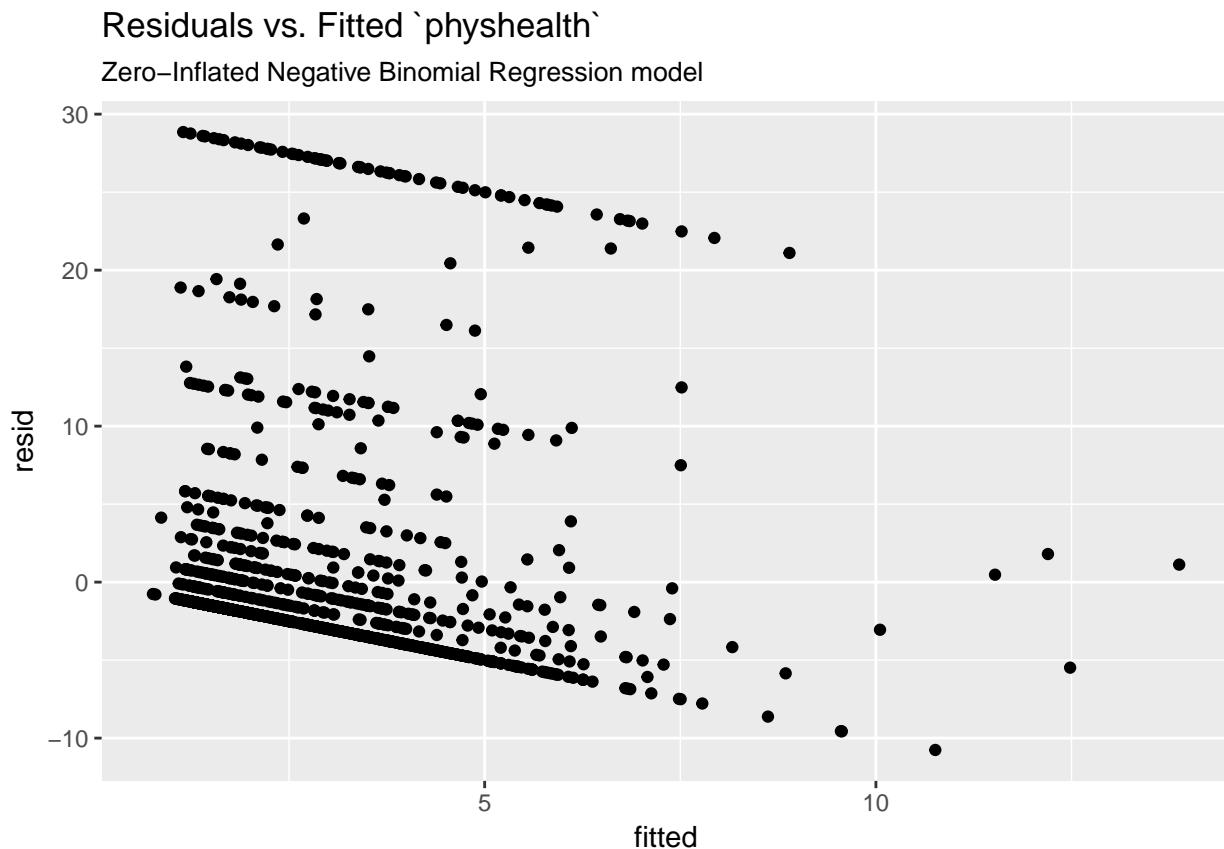
Here, we have

Model	Scale	R^2	log(likelihood)
Zero-Inflated Negative Binomial	Complex: log(physhealth)	.049	-2567.8

18.14.10 Check model assumptions

Here is a plot of residuals vs. fitted values on the original physhealth scale.

```
ggplot(sm_zinb1, aes(x = fitted, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs. Fitted `physhealth`",
       subtitle = "Zero-Inflated Negative Binomial Regression model")
```



18.14.11 Predictions for Harry and Sally

The predictions from this zero-inflated negative binomial regression model are obtained as follows...

```
predict(mod_zinb1, newdata = hs_data, type = "response")
```

1	2
6.019361	1.444829

As we've seen in the past, when we use `response` as the type, the predictions fall on the original `physhealth` scale. The prediction for Harry is 6.01 days, and for Sally is 1.45 days.

18.15 A “hurdle” model (with Poisson)

Much of the discussion here of hurdle models comes from Clay Ford at the University of Virginia⁹. Ford describes a hurdle model as follows:

The hurdle model is a two-part model that specifies one process for zero counts and another process for positive counts. The idea is that positive counts occur once a threshold is crossed, or put another way, a hurdle is cleared. If the hurdle is not cleared, then we have a count of 0.

The first part of the model is typically a binary logit model. This models whether an observation takes a positive count or not. The second part of the model is usually a truncated Poisson or Negative Binomial model. Truncated means we’re only fitting positive counts. If we were to fit a hurdle model to our [medicare] data, the interpretation would be that one process governs whether a patient visits a doctor or not, and another process governs how many visits are made.

To fit a hurdle model, we’ll use the `hurdle` function in the `pscl` package.

```
mod_hur1 <- hurdle(physhealth ~ bmi_c + smoke100,
                     dist = "poisson", zero.dist = "binomial",
                     data = sm_oh_A_young)

summary(mod_hur1)
```

Call:

```
hurdle(formula = physhealth ~ bmi_c + smoke100, data = sm_oh_A_young,
       dist = "poisson", zero.dist = "binomial")
```

Pearson residuals:

Min	1Q	Median	3Q	Max
-1.4170	-0.6460	-0.5316	-0.2561	11.0526

Count model coefficients (truncated poisson with log link):

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.897676	0.022696	83.612	<2e-16 ***
bmi_c	0.013985	0.001698	8.235	<2e-16 ***
smoke100	0.437649	0.030107	14.537	<2e-16 ***

Zero hurdle model coefficients (binomial with logit link):

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-0.923456	0.069872	-13.216	< 2e-16 ***
bmi_c	0.050172	0.007791	6.440	1.2e-10 ***
smoke100	0.417888	0.110144	3.794	0.000148 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Number of iterations in BFGS optimization: 14

Log-likelihood: -4178 on 6 Df

```
confint(mod_hur1)
```

	2.5 %	97.5 %
count_(Intercept)	1.85319158	1.94215943

⁹<http://data.library.virginia.edu/getting-started-with-hurdle-models/> is an excellent introduction, by Clay Ford, a Statistical Research Consultant at the University of Virginia Library. I can also recommend https://rpubs.com/kaz_yos/pscl-2 as a place to learn more about the `pscl` package, and the fitting and interpretation of both hurdle and zero-inflated regression models. That `rpubs` site has a link to this article by Hu, Pavlicova and Nunes from the Am J Drug Alcohol Abuse which provides a real set of examples from a trial of a behavioral health intervention meant to reduce the risk of unprotected sexual occasions as part of a strategy to reduce HIV risk.

```

count_bmi_c      0.01065671  0.01731347
count_smoke100   0.37864083  0.49665711
zero_(Intercept) -1.06040337 -0.78650834
zero_bmi_c       0.03490142  0.06544259
zero_smoke100    0.20201037  0.63376638

```

We are using the default settings here, using the same predictors for both models:

- a **Binomial** model to predict the probability of `physhealth` = 0 given our predictors, as specified by the `zero.dist` argument in the `hurdle` function, and
- a (truncated) **Poisson** model to predict the positive-count of `physhealth` given those same predictors, as specified by the `dist` argument in the `hurdle` function.

18.15.1 Comparison to a null model

To show that this model fits better than the null model (the model with intercept only), we can compare them directly with a chi-squared test. Since we have two predictors in the full model, the degrees of freedom for this test is 2.

```

mod_hurnull <- hurdle(physhealth ~ 1, dist = "poisson",
                      zero.dist = "binomial",
                      data = sm_oh_A_young)

summary(mod_hurnull)

```

```

Call:
hurdle(formula = physhealth ~ 1, data = sm_oh_A_young, dist = "poisson",
       zero.dist = "binomial")

Pearson residuals:
    Min     1Q Median     3Q    Max 
-0.6355 -0.6355 -0.6355 -0.1720  6.3162 

Count model coefficients (truncated poisson with log link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  2.14401   0.01495  143.4   <2e-16 *** 
Zero hurdle model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -0.74782   0.05298  -14.12   <2e-16 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 8
Log-likelihood: -4351 on 2 Df
pchisq(2 * (logLik(mod_hur1) - logLik(mod_hurnull)), df = 2, lower.tail = FALSE)

'log Lik.' 1.46309e-75 (df=6)

```

18.15.2 Comparison to a Poisson Model: Vuong test

```

vuong(mod_hur1, mod_poiss1)

```

Vuong Non-Nested Hypothesis Test-Statistic:

```
(test-statistic is asymptotically distributed N(0,1) under the
null that the models are indistinguishable)
-----
```

	Vuong z-statistic	H_A	p-value
Raw	15.54734	model1 > model2	< 2.22e-16
AIC-corrected	15.53459	model1 > model2	< 2.22e-16
BIC-corrected	15.50016	model1 > model2	< 2.22e-16

The hurdle model is a significant improvement over the standard Poisson model according to this test.

18.15.3 Comparison to a Zero-Inflated Poisson Model: Vuong test

Is the hurdle model comparable to the zero-inflated Poisson?

```
vuong(mod_hur1, mod_zip1)
```

```
Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
null that the models are indistinguishable)
-----
```

	Vuong z-statistic	H_A	p-value
Raw	0.4811757	model1 > model2	0.3152
AIC-corrected	0.4811757	model1 > model2	0.3152
BIC-corrected	0.4811757	model1 > model2	0.3152

The hurdle model is not a significant improvement over the zero-inflated Poisson model according to this test.

18.15.4 The Fitted Equation

The form of the model equation for this hurdle also requires us to take two separate models into account. First we have a logistic regression model to predict the log odds of zero physhealth days...

```
logit(physhealth = 0) = 0.921 - 0.051 bmi_c - 0.413 smoke100
```

That takes care of the zeros. Then, to predict the number of physhealth days, we use the following truncated Poisson model:

```
log(physhealth) = 1.90 + 0.014 bmi_c + 0.441 smoke100
```

which is truncated to produce only estimates greater than zero.

18.15.5 Interpreting the Coefficients

We can exponentiate the logistic regression coefficients to obtain results in terms of odds ratios for that model, and that can be of some help in understanding the process behind excess zeros.

Also, exponentiating the coefficients of the count model help us describe those counts on the original scale of physhealth.

```
exp(coef(mod_hur1))
```

count_(Intercept)	count_bmi_c	count_smoke100	zero_(Intercept)
6.6703712	1.0140833	1.5490611	0.3971442
zero_bmi_c	zero_smoke100		
1.0514519	1.5187511		

For example,

- in the model for `physhealth = 0`, the odds of `physhealth = 0` are 151% as high for subjects with `smoke100 = 1` as for non-smokers with the same BMI.
- in the Poisson model for `physhealth`, the `physhealth` count is estimated to increase by 1.55 for smokers as compared to non-smokers with the same BMI.

18.15.6 Testing the Predictors

We can test the model with and without `bmi_c`, for example, by fitting the model both ways, and comparing the results with either a Wald or Likelihood Ratio test, each of which is available in the `lmtest` package.

```
mod_hur1_nobmi <- hurdle(physhealth ~ smoke100,
                           dist = "poisson",
                           zero.dist = "binomial",
                           data = sm_oh_A_young)

lmtest::waldtest(mod_hur1, mod_hur1_nobmi)

Wald test

Model 1: physhealth ~ bmi_c + smoke100
Model 2: physhealth ~ smoke100
  Res.Df Df  Chisq Pr(>Chisq)
1    1628
2    1630 -2 109.29 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

lmtest::lrtest(mod_hur1, mod_hur1_nobmi)

Likelihood ratio test

Model 1: physhealth ~ bmi_c + smoke100
Model 2: physhealth ~ smoke100
  #Df  LogLik Df  Chisq Pr(>Chisq)
1    6 -4178.5
2    4 -4232.0 -2 106.97 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

18.15.7 Store fitted values and residuals

The `broom` package does not work with the `hurdle` class of models. Again we need to build up the fitted values and residuals ourselves.

```
sm_hur1 <- sm_oh_A_young %>%
  mutate(fitted = fitted(mod_hur1, type = "response"),
         resid = resid(mod_hur1, type = "response"))

sm_hur1 %>%
  dplyr::select(physhealth, fitted, resid) %>%
  head()

# A tibble: 6 x 3
  physhealth fitted   resid
  <int>     <dbl>   <dbl>
```

```

1      0  1.76 -1.76
2      0  2.53 -2.53
3      0  1.67 -1.67
4      2  2.04 -0.0353
5      4  5.02 -1.02
6      6  1.55  4.45

```

18.15.8 Modeled Number of Zero Counts

Once again, we can compare the observed number of zero `physhealth` results to the expected number of zero values from the likelihood-based models.

```

round(c("Obs" = sum(sm_oh_A_young$physhealth == 0),
  "Poisson" = sum(dpois(0, fitted(mod_poiss1))),
  "NB" = sum(dnbinom(0, mu = fitted(mod_nb1), size = mod_nb1$theta)),
  "ZIP" = sum(predict(mod_zip1, type = "prob")[,1]),
  "ZINB" = sum(predict(mod_zinb1, type = "prob")[,1]),
  "Hurdle" = sum(predict(mod_hur1, type = "prob")[,1])),0)

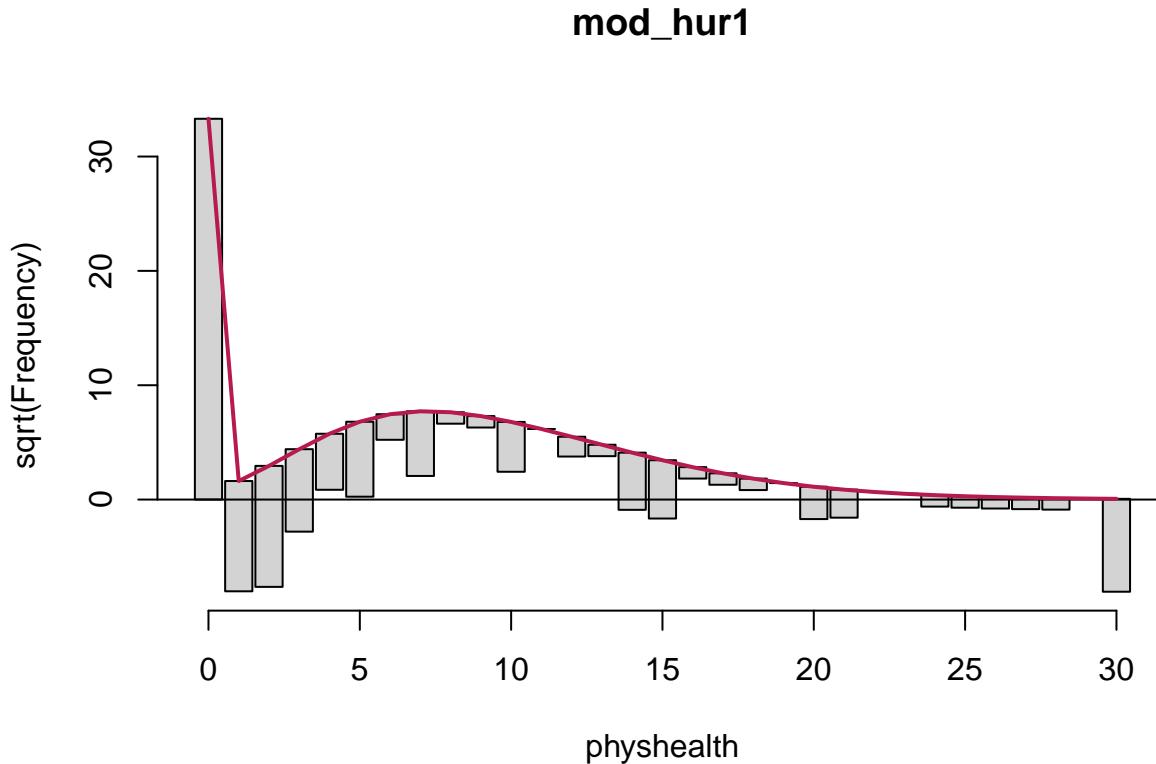
```

Obs	Poisson	NB	ZIP	ZINB	Hurdle
1109	190	1101	1109	1111	1109

The hurdle model does about as well as the negative binomial and zero-inflated models. All but the Poisson give reasonable fits in this regard.

18.15.9 Rootogram for Hurdle Model

```
countreg::rootogram(mod_hur1, max = 30)
```



The results are still not perfect, of course. In fitting the zeros exactly, we're underfitting counts of 1, 2, and 30, and overfitting many of the counts between 6 and 20. We still have a problem here with overdispersion. That's why we'll consider a hurdle model with a negative binomial regression for the counts in a moment.

18.15.10 Understanding the Modeled Counts in Detail

The expected mean count uses both parts of the hurdle model. Mathematically, we want...

$$E[y|\mathbf{x}] = \frac{1 - f_1(0|\mathbf{x})}{1 - f_2(0|\mathbf{x})} \mu_2(\mathbf{x})$$

where

- our count of `physhealth` is y
- our predictors are represented by \mathbf{x}
- and the expected count is the product of a ratio and a mean.

The ratio is the probability of a non-zero in the first process divided the probability of a non-zero in the second untruncated process. The f symbols represent distributions. Recall these are logistic and Poisson, respectively, by default but can be others. The mean is for the untruncated version of the positive-count process.

If we want to see the expected hurdle counts, we can get them using some clever applications of the `predict` function.

The first six expected mean counts ($E[y|\mathbf{x}]$ from the equation above) are:

```
head(predict(mod_hur1, type = "response"))
```

1	2	3	4	5	6
1.756341	2.527945	1.671933	2.035348	5.018026	1.551978

The ratio of non-zero probabilities, $\frac{1-f_1(0|\mathbf{x})}{1-f_2(0|\mathbf{x})}$, from the mathematical expression above can be extracted by:

```
head(predict(mod_hur1, type = "zero"))
```

1	2	3	4	5	6
0.2690804	0.3487780	0.2596470	0.2991808	0.5499740	0.2459557

The mean for the untruncated process, $\mu_2(\mathbf{x})$, can also be obtained by:

```
head(predict(mod_hur1, type = "count"))
```

1	2	3	4	5	6
6.527200	7.248006	6.439253	6.803070	9.124116	6.309990

and we can multiply these last two pieces together to verify that they match our expected hurdle counts.

```
head(predict(mod_hur1, type = "zero") * predict(mod_hur1, type = "count"), 5)
```

1	2	3	4	5
1.756341	2.527945	1.671933	2.035348	5.018026

18.15.11 Specify the R² and log (likelihood) values

We can calculate a proxy for R² as the squared correlation of the fitted values and the observed values.

```
# The correlation of observed and fitted values
(hur1_r <- with(sm_hur1, cor(loghealth, fitted)))
```

```
[1] 0.2235982
```

```
# R-square
hur1_r^2
```

```
[1] 0.04999615
```

```
logLik(mod_hur1)
```

```
'log Lik.' -4178.491 (df=6)
```

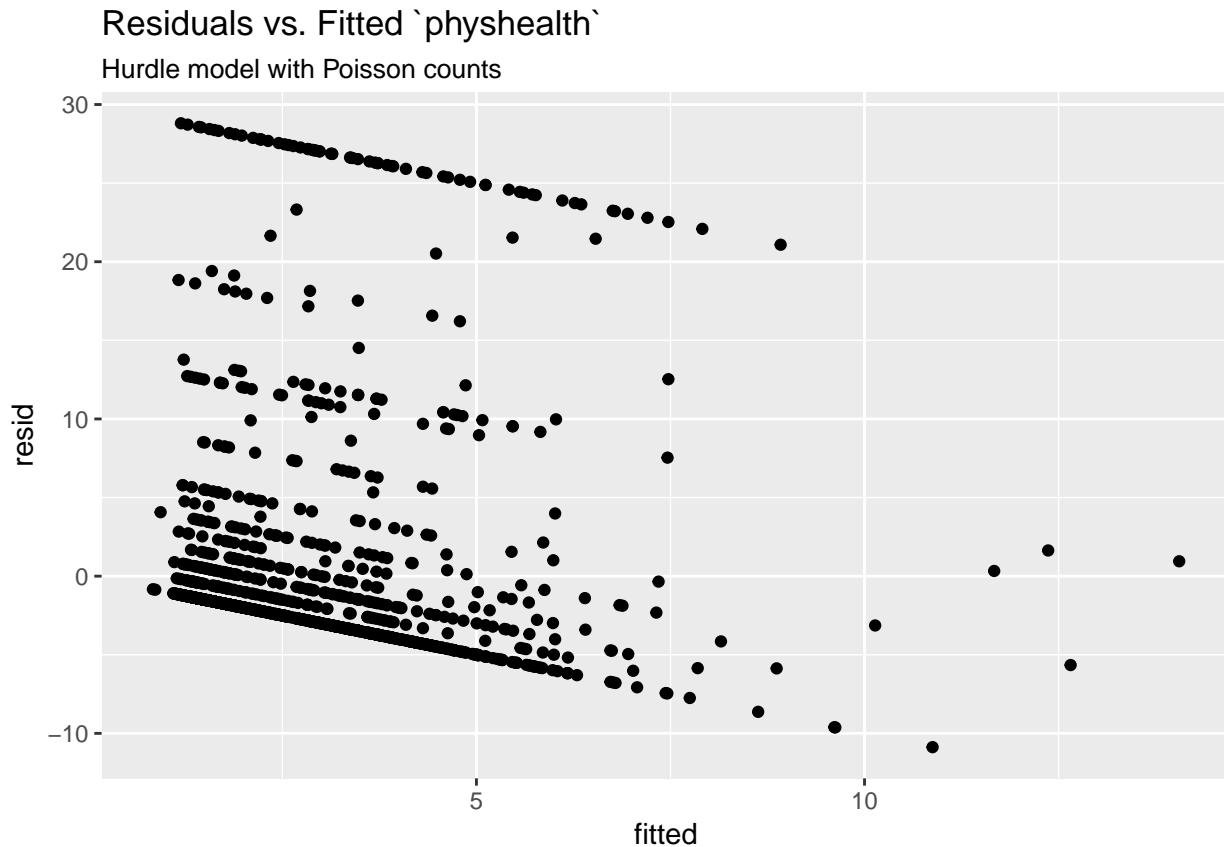
Here, we have

Model	Scale	R ²	log(likelihood)
Hurdle Model (Poisson)	Complex: log(loghealth)	.050	-4184.1

18.15.12 Check model assumptions

Here is a plot of residuals vs. fitted values on the original physhealth scale.

```
ggplot(sm_hur1, aes(x = fitted, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs. Fitted `physhealth`",
       subtitle = "Hurdle model with Poisson counts")
```



18.15.13 Predictions for Harry and Sally

The predictions from this zero-inflated negative binomial regression model are obtained as follows...

```
predict(mod_hur1, newdata = hs_data, type = "response")
```

```
1      2
5.930506 1.471289
```

As we've seen in the past, when we use `response` as the type, the predictions fall on the original `physhealth` scale. The prediction for Harry is 5.93 days, and for Sally is 1.47 days.

18.16 A “hurdle” model (with negative binomial for overdispersion)

Let's account for overdispersion better with a negative binomial model for the counts in our hurdle model. We specify that the positive-count process be fit with this NB model using `dist = negbin`.

```
mod_hur_nb1 <- hurdle(physhealth ~ bmi_c + smoke100,
                        dist = "negbin", zero.dist = "binomial",
                        data = sm_oh_A_young)

summary(mod_hur_nb1)
```

```

Call:
hurdle(formula = physhealth ~ bmi_c + smoke100, data = sm_oh_A_young,
       dist = "negbin", zero.dist = "binomial")

Pearson residuals:
    Min      1Q  Median      3Q     Max
-0.5763 -0.3843 -0.3424 -0.1548  7.4930

Count model coefficients (truncated negbin with log link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 1.305260  0.144512  9.032 < 2e-16 ***
bmi_c       0.018524  0.008131  2.278  0.0227 *
smoke100   0.573519  0.138747  4.134 3.57e-05 ***
Log(theta) -1.123030  0.206483 -5.439 5.36e-08 ***
Zero hurdle model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.923456  0.069872 -13.216 < 2e-16 ***
bmi_c        0.050172  0.007791  6.440 1.2e-10 ***
smoke100    0.417888  0.110144  3.794 0.000148 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Theta: count = 0.3253
Number of iterations in BFGS optimization: 17
Log-likelihood: -2563 on 7 Df
confint(mod_hur_nb1)

```

	2.5 %	97.5 %
count_(Intercept)	1.022021965	1.58849765
count_bmi_c	0.002586927	0.03446173
count_smoke100	0.301580258	0.84545695
zero_(Intercept)	-1.060403367	-0.78650834
zero_bmi_c	0.034901419	0.06544259
zero_smoke100	0.202010366	0.63376638

18.16.1 Comparison to a null model

To show that this model fits better than the null model (the model with intercept only), we can compare them directly with a chi-squared test. Since we have two predictors in the full model, the degrees of freedom for this test is 2.

```

mod_hur_nb_null <- hurdle(physhealth ~ 1, dist = "negbin",
                           zero.dist = "binomial",
                           data = sm_oh_A_young)

summary(mod_hur_nb_null)

```

```

Call:
hurdle(formula = physhealth ~ 1, data = sm_oh_A_young, dist = "negbin",
       zero.dist = "binomial")

Pearson residuals:
    Min      1Q  Median      3Q     Max

```

```

-0.3700 -0.3700 -0.3700 -0.1002  3.6775

Count model coefficients (truncated negbin with log link):
  Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.5356     0.1499 10.246 < 2e-16 ***
Log(theta)   -1.3053     0.2330 -5.603 2.1e-08 ***
Zero hurdle model coefficients (binomial with logit link):
  Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.74782    0.05298 -14.12 <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Theta: count = 0.2711
Number of iterations in BFGS optimization: 11
Log-likelihood: -2603 on 3 Df
pchisq(2 * (logLik(mod_hur_nb1) - logLik(mod_hur_nb_null)), df = 2, lower.tail = FALSE)

'log Lik.' 3.064911e-18 (df=7)

```

18.16.2 Comparison to a Negative Binomial Model: Vuong test

```
vuong(mod_hur_nb1, mod_nb1)
```

Vuong Non-Nested Hypothesis Test-Statistic:
 (test-statistic is asymptotically distributed $N(0,1)$ under the null that the models are indistinguishable)

	Vuong z-statistic	H_A	p-value
Raw	3.224162	model1 > model2	0.00063171
AIC-corrected	2.707707	model1 > model2	0.00338749
BIC-corrected	1.313591	model1 > model2	0.09449192

The hurdle model is a significant improvement over the standard negative binomial model according to the raw and AIC-corrected versions of this test, but not the BIC-corrected version.

18.16.3 Comparison to a Zero-Inflated NB Model: Vuong test

Is the hurdle model comparable to the zero-inflated Poisson?

```
vuong(mod_hur_nb1, mod_zinb1)
```

Vuong Non-Nested Hypothesis Test-Statistic:
 (test-statistic is asymptotically distributed $N(0,1)$ under the null that the models are indistinguishable)

	Vuong z-statistic	H_A	p-value
Raw	1.625551	model1 > model2	0.052023
AIC-corrected	1.625551	model1 > model2	0.052023
BIC-corrected	1.625551	model1 > model2	0.052023

The hurdle model is just barely a significant improvement over the zero-inflated Negative Binomial model.

18.16.4 Comparing the Hurdle Models with AIC and BIC

```
AIC(mod_hur1); BIC(mod_hur1)

[1] 8368.981
[1] 8401.374

AIC(mod_hur_nb1); BIC(mod_hur_nb1)

[1] 5139.903
[1] 5177.695
```

The negative binomial approach certainly looks better than the Poisson here.

18.16.5 The Fitted Equation

The form of the model equation for this hurdle also requires us to take two separate models into account. First we have a logistic regression model to predict the log odds of zero `physhealth` days...

```
logit(physhealth = 0) = -0.921 + 0.051 bmi_c + 0.413 smoke100
```

That takes care of the zeros. Then, to predict the number of `physhealth` days, we use the following truncated negative binomial model:

```
log(physhealth) = 1.30 + 0.018 bmi_c + 0.576 smoke100
```

which is truncated to produce only estimates greater than zero, with θ estimated as `exp(-1.123)` or 0.325.

18.16.6 Interpreting the Coefficients

We can exponentiate the logistic regression coefficients to obtain results in terms of odds ratios for that model, and that can be of some help in understanding the process behind excess zeros.

```
exp(coef(mod_hur_nb1))

count_(Intercept)      count_bmi_c      count_smoke100   zero_(Intercept)
3.6886473             1.0186970        1.7744998       0.3971442
zero_bmi_c            zero_smoke100
1.0514519             1.5187511
```

For example,

- in the model for `physhealth` = 0, the odds of `physhealth` = 0 are 151% as high for subjects with `smoke100` = 1 as for non-smokers with the same BMI.

18.16.7 Testing the Predictors

We can test the model with and without `bmi_c`, for example, by fitting the model both ways, and comparing the results with either a Wald or Likelihood Ratio test, each of which is available in the `lmtest` package.

```
mod_hurnb1_nobmi <- hurdle(physhealth ~ smoke100,
                             dist = "negbin",
                             zero.dist = "binomial",
                             data = sm_oh_A_young)

lmtest::waldtest(mod_hur_nb1, mod_hurnb1_nobmi)
```

Wald test

```
Model 1: physhealth ~ bmi_c + smoke100
Model 2: physhealth ~ smoke100
  Res.Df Df  Chisq Pr(>Chisq)
1     1627
2     1629 -2 46.657 7.388e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
lmtest::lrtest(mod_hur_nb1, mod_hurnb1_nobmi)
```

Likelihood ratio test

```
Model 1: physhealth ~ bmi_c + smoke100
Model 2: physhealth ~ smoke100
#Df LogLik Df  Chisq Pr(>Chisq)
1    7 -2562.9
2    5 -2587.1 -2 48.218 3.385e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

18.16.8 Store fitted values and residuals

Again we need to build up the fitted values and residuals, without `broom` to help.

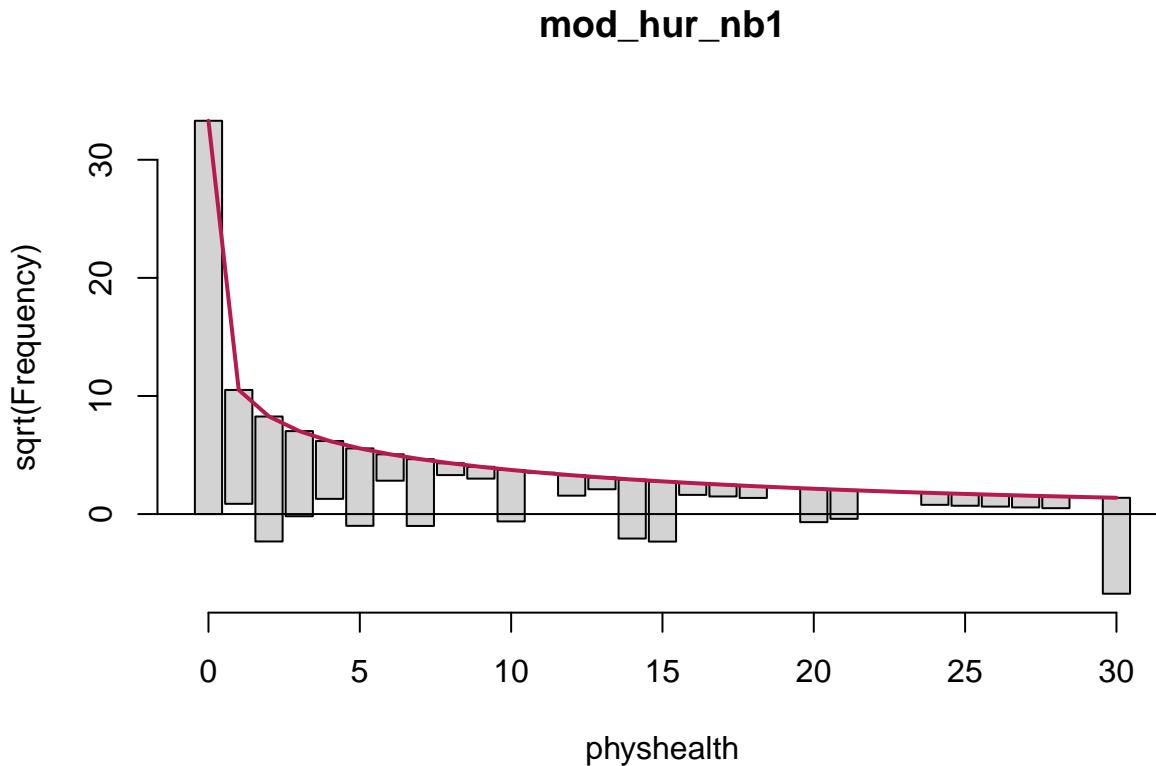
```
sm_hur_nb1 <- sm_oh_A_young %>%
  mutate(fitted = fitted(mod_hur_nb1, type = "response"),
         resid = resid(mod_hur_nb1, type = "response"))

sm_hur_nb1 %>%
  dplyr::select(physhealth, fitted, resid) %>%
  head()

# A tibble: 6 x 3
  physhealth fitted   resid
  <int>     <dbl>   <dbl>
1        0     1.74 -1.74
2        0     2.51 -2.51
3        0     1.65 -1.65
4        2     2.01 -0.0138
5        4     5.03 -1.03
6        6     1.53  4.47
```

18.16.9 Rootogram for NB Hurdle Model

```
countreg::rootogram(mod_hur_nb1, max = 30)
```



This improves the situation, but we’re still underfitting the 30s.

18.16.10 Specify the R^2 and log (likelihood) values

We can calculate a proxy for R^2 as the squared correlation of the fitted values and the observed values.

```
# The correlation of observed and fitted values
(hurnb1_r <- with(sm_hur_nb1, cor(physhealth, fitted)))
```

```
[1] 0.2232941
```

```
# R-square
hurnb1_r^2
```

```
[1] 0.04986026
```

```
logLik(mod_hur_nb1)
```

```
'log Lik.' -2562.952 (df=7)
```

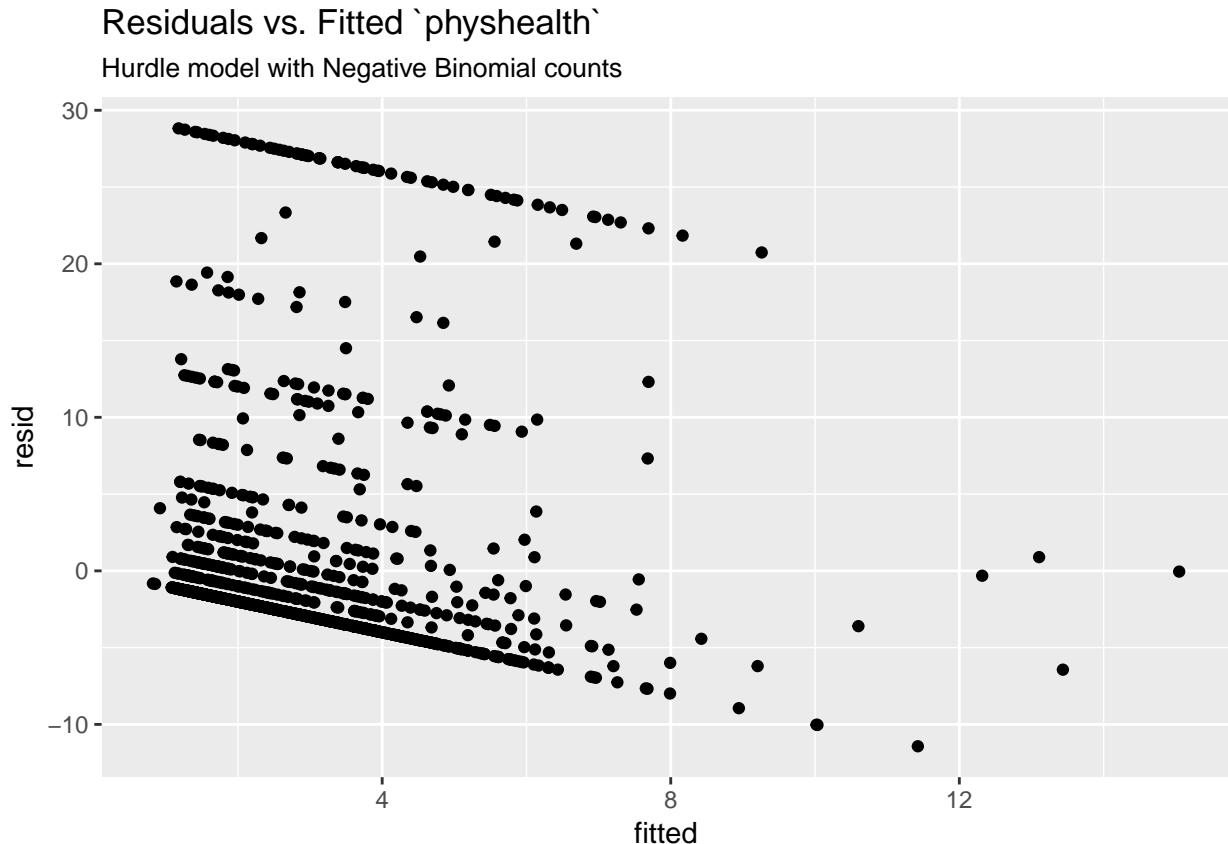
Here, we have

Model	Scale	R^2	log(likelihood)
Hurdle Model (Neg. Bin.)	Complex: log(physhealth)	.050	-2566.3

18.16.11 Check model assumptions

Here is a plot of residuals vs. fitted values on the original physhealth scale.

```
ggplot(sm_hur_nb1, aes(x = fitted, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs. Fitted `physhealth`",
       subtitle = "Hurdle model with Negative Binomial counts")
```



18.16.12 Predictions for Harry and Sally

The predictions from this zero-inflated negative binomial regression model are obtained as follows...

```
predict(mod_hur_nb1, newdata = hs_data, type = "response")
```

1	2
6.047813	1.453596

The prediction for Harry is 6.04 days, and for Sally is 1.45 days.

18.16.13 Note: Fitting a Different Hurdle Model for Counts and Pr(zero)

Suppose we wanted to use only bmi_c to predict the probability of a zero count, but use both predictors in the model for the positive counts. We use the | command.

```
mod_hur_new1 <-
  hurdle(physhealth ~ bmi_c + smoke100 | bmi_c,
```

```

dist = "negbin", zero.dist = "binomial",
data = sm_oh_A_young)

summary(mod_hur_new1)

Call:
hurdle(formula = physhealth ~ bmi_c + smoke100 | bmi_c, data = sm_oh_A_young,
       dist = "negbin", zero.dist = "binomial")

Pearson residuals:
    Min      1Q  Median      3Q      Max
-0.5681 -0.3809 -0.3488 -0.1517  7.0650

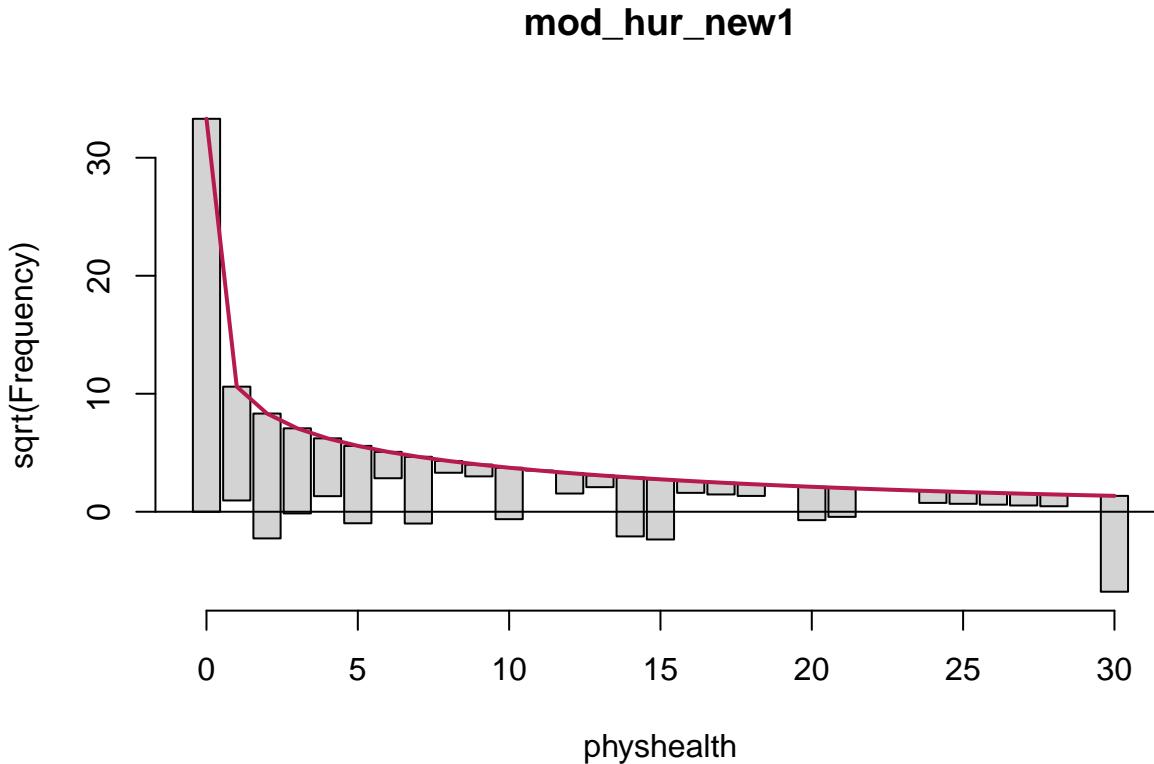
Count model coefficients (truncated negbin with log link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 1.305260  0.144512  9.032 < 2e-16 ***
bmi_c       0.018524  0.008131  2.278  0.0227 *
smoke100   0.573519  0.138747  4.134 3.57e-05 ***
Log(theta) -1.123030  0.206483 -5.439 5.36e-08 ***
Zero hurdle model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.762681  0.053905 -14.149 < 2e-16 ***
bmi_c       0.051171  0.007757  6.597 4.2e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Theta: count = 0.3253
Number of iterations in BFGS optimization: 17
Log-likelihood: -2570 on 6 Df

```

18.16.14 Hanging Rootogram for this new Hurdle Model

```
countreg::rootogram(mod_hur_new1, max = 30)
```



Not a meaningful improvement, certainly.

18.17 A Tobit (Censored) Regression Model

The idea of the **tobit** model (sometimes called a **censored regression** model) is to estimate associations for outcomes where we can see either left-censoring (censoring from below) or right-censoring (censoring from above.)

Consider the variable `physhealth`, which is restricted to fall between 0 and 30 by the way the measure was constructed. But supposed we think about a broader and latent (unobserved) variable describing physical health. Among the people with `physhealth` = 0, some would be incredible athletes and others would be in much poorer physical health but still healthy enough to truthfully answer 0. On the other end, some of the people responding 30 are in substantially worse physical health than others with that same response.

- Censoring from below takes place when values at or below a threshold (in this case 0) take that value.
- Censoring from above takes place when values at or above a threshold (here, 30) take that value.

Several examples of tobit analysis are available at <https://stats.idre.ucla.edu/r/dae/tobit-models/>, which is my primary source for the material here on those models.

The tobit model postulates that the value 0 in our model is just the lower limit of the underlying measure of poor physical health that we would actually observe in the population if we had a stronger measure. Similarly, we'll postulate that 30 is just the upper limit of "poor health" that we can see. The approach I'll take to run the tobit model comes from the `vglm` function in the `VGAM` package.

Here's the model, and its summary. Note that the default Lower value for a tobit model is 0, so we didn't technically have to list that here.

```

mod_tob1 <- vglm(physhealth ~ bmi_c + smoke100,
                  tobit(Lower = 0, Upper = 30),
                  type.fitted = "censored",
                  data = sm_oh_A_young)

summary(mod_tob1)

```

Call:
`vglm(formula = physhealth ~ bmi_c + smoke100, family = tobit(Lower = 0,
 Upper = 30), data = sm_oh_A_young, type.fitted = "censored")`

Pearson residuals:

	Min	1Q	Median	3Q	Max
<code>mu</code>	-4.963	-0.6059	-0.4371	0.9127	2.558
<code>loge(sd)</code>	-1.065	-0.3039	-0.2862	-0.1981	9.019

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)							
(Intercept):1	-11.02997	0.84462	-13.059	< 2e-16 ***							
(Intercept):2	2.81768	0.03553	79.310	< 2e-16 ***							
<code>bmi_c</code>	0.49105	0.07289	6.737	1.62e-11 ***							
<code>smoke100</code>	5.69619	1.06207	5.363	8.17e-08 ***							

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

Number of linear predictors: 2

Names of linear predictors: `mu`, `loge(sd)`

Log-likelihood: -2563.135 on 3264 degrees of freedom

Number of iterations: 7

No Hauck-Donner effect found in any of the estimates

```
confint(mod_tob1)
```

	2.5 %	97.5 %
(Intercept):1	-12.6854002	-9.3745415
(Intercept):2	2.7480495	2.8873146
<code>bmi_c</code>	0.3481883	0.6339102
<code>smoke100</code>	3.6145827	7.7778052

18.17.1 The Fitted Equation

Because we've used the censoring approach, our model will limit its predictions to the range of [0, 30], where any predictions outside that range are censored. Values below 0 are fitted as 0, and values above 30 are fitted as 30.

The model equation is

```
physhealth = -11.00 + 0.49 bmi_c + 5.67 smoke100
```

18.17.2 Interpreting the Coefficients

Tobit model regression coefficients are interpreted as we would a set of OLS coefficients, *except* that the linear effect is on the uncensored *latent variable*, rather than on the observed outcome.

In our case,

- a one-unit increase in `bmi_c` is associated with a 0.49 day increase in the predicted value of `physhealth`, holding `smoke100` constant
- a move from `smoke100 = 0` to 1 is associated with a 5.67 day increase in the predicted value of `physhealth`, holding `bmi_c` constant
- the coefficient labeled `(Intercept):1` is the intercept for the model and is the predicted value of `physhealth` when `smoke100 = 0` and `bmi_c = 0`. Note that this value is -11, which is outside the range of `physhealth` values we observed.
- the coefficient labeled `(Intercept):2` is a statistic we can use after we exponentiate it, as follows:
 - here `(Intercept):2 = 2.82`, and $\exp(2.82) = 16.7768507$, which is analogous to the square root of the residual variance in OLS regression, which is summarized for our OLS model as `Residual standard error: 6.609`.

18.17.3 Testing the Predictors

We can test the model with and without `bmi_c`, for example, by fitting the model both ways, and comparing the results with either a Wald or Likelihood Ratio test, each of which is available in the `lmtest` package.

```
mod_tob_nobmi <- vglm(physhealth ~ smoke100,
                        tobit(Lower = 0, Upper = 30),
                        type.fitted = "censored",
                        data = sm_oh_A_young)

lmtest::waldtest(mod_tob1, mod_tob_nobmi)
```

Wald test

```
Model 1: physhealth ~ bmi_c + smoke100
Model 2: physhealth ~ smoke100
  Res.Df Df  Chisq Pr(>Chisq)
1   3264
2   3265 -1 45.386  1.618e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The likelihood ratio test we have used in some other settings isn't available here.

18.17.4 Store fitted values and residuals

The residuals and fitted values from the tobit model can be stored and then summarized in several ways:

```
sm_tob1 <- sm_oh_A_young %>%
  mutate(fitted = fitted(mod_tob1,
                        type.fitted = "censored"),
         resid = physhealth - fitted)

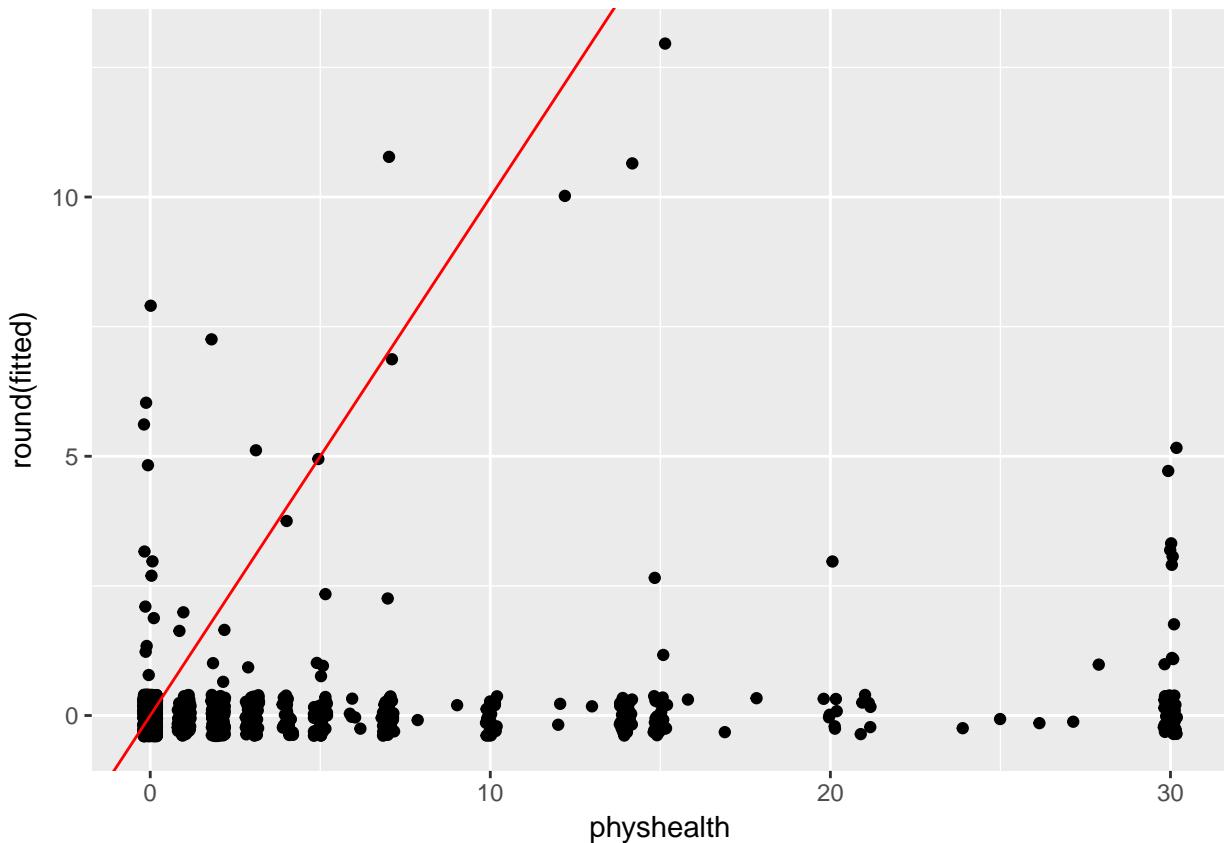
sm_tob1 %>%
  dplyr::select(physhealth, fitted, resid) %>%
  head()
```

```
# A tibble: 6 x 3
  physhealth fitted resid
    <int>     <dbl>   <dbl>
1          0      0.     0.
2          0      0.     0.
3          0      0.     0.
4          2      0.     2.
5          4      0.     4.
6          6      0.     6.
```

18.17.5 Building Something Like a Rootogram

Building a rootogram is tricky for a tobit model, to say the least, but we can approximate a piece of the issue by plotting the rounded fitted values against the observed `physhealth` data.

```
ggplot(sm_tob1, aes(x = physhealth, y = round(fitted))) +
  geom_jitter(width = 0.2) +
  geom_abline(intercept = 0, slope = 1, col = "red")
```



Note that the model never predicts a subject to have an underlying `physhealth` worse than about 12 (remember that larger numbers are worse here.)

18.17.6 Tables of the Observed and Fitted `physhealth` from Tobit

```
addmargins(table(round(sm_tob1$physhealth)))
```

0	1	2	3	4	5	6	7	8	9	10	12	13	14	15
1109	93	112	52	24	43	5	32	1	1	19	3	1	25	26
16	17	18	20	21	24	25	26	27	28	30	Sum			
1	1	1	8	6	1	1	1	1	1	66	1634			

```
addmargins(table(round(sm_tob1$fitted)))
```

0	1	2	3	4	5	6	7	8	10	11	13	Sum
1588	14	8	9	1	5	2	2	1	1	2	1	1634

18.17.7 Specify the R² and log (likelihood) values

We can calculate a proxy for R² as the squared correlation of the fitted values and the observed values.

```
# The correlation of observed and fitted values
(tob1_r <- with(sm_tob1, cor(physhealth, fitted)))
```

```
[,1]
[1,] 0.1440334
```

```
# R-square
tob1_r^2
```

```
[,1]
[1,] 0.02074562
```

```
logLik(mod_tob1)
```

```
[1] -2563.135
```

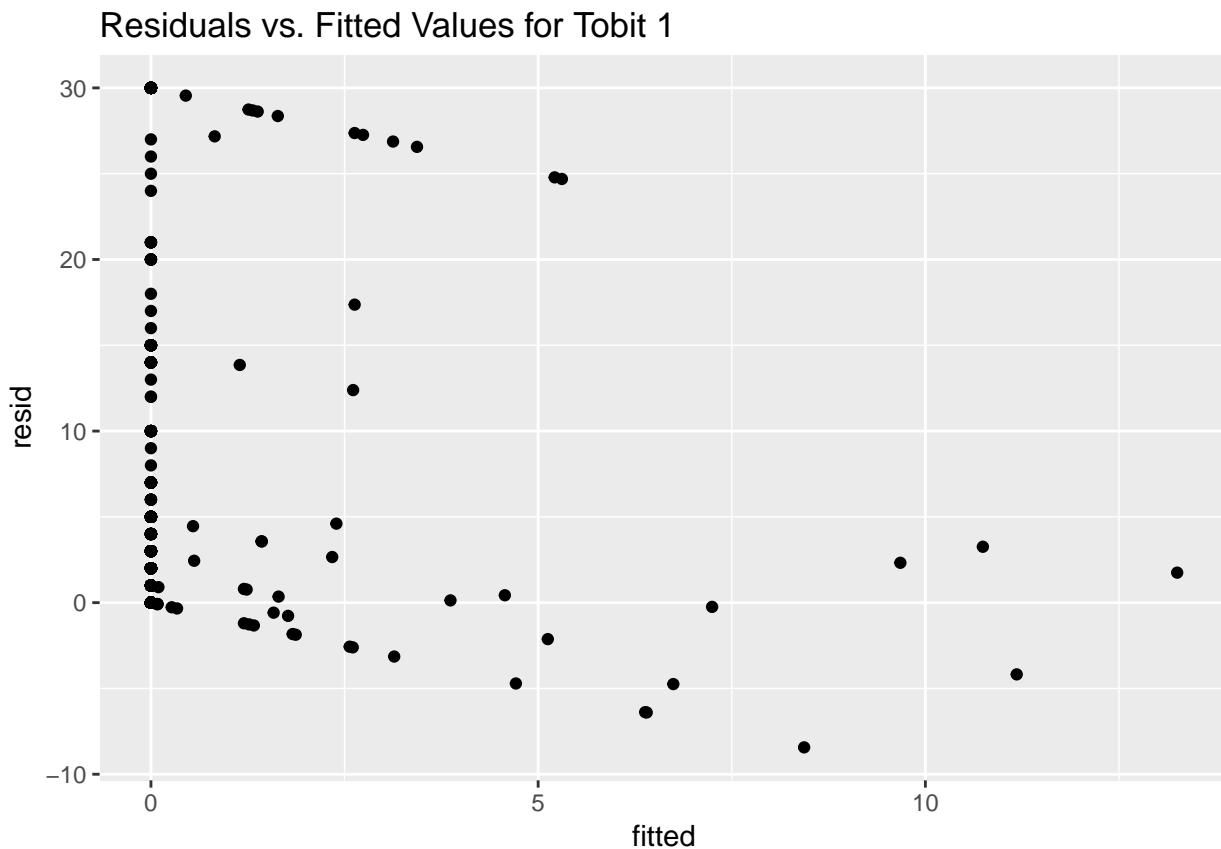
Here, we have

Model	Scale	R ²	log(likelihood)
Tobit	physhealth	.021	-2567.3

18.17.8 Check model assumptions

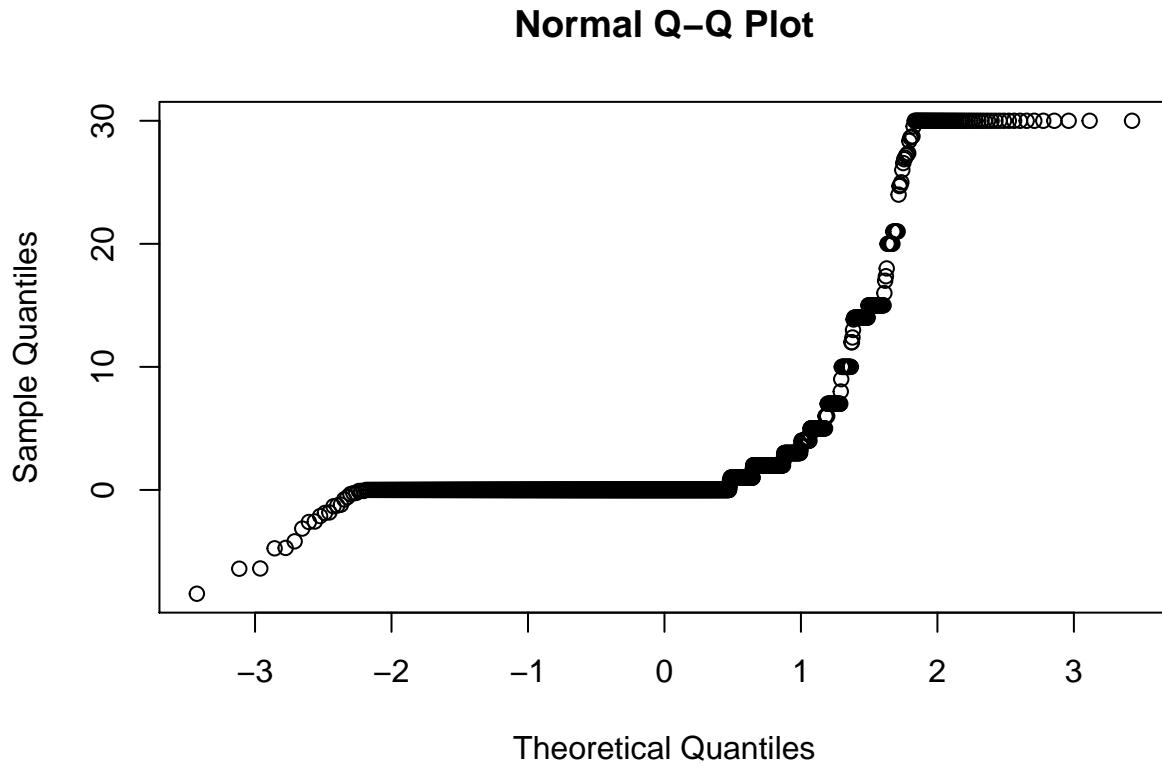
Here is a plot of residuals vs. fitted values.

```
ggplot(sm_tob1, aes(x = fitted, y = resid)) +
  geom_point() +
  labs(title = "Residuals vs. Fitted Values for Tobit 1")
```



Here is a normal Q-Q plot of the Tobit Model 1 residuals.

```
qqnorm(sm_tob1$resid)
```



18.17.9 Predictions for Harry and Sally

The predictions from this tobit model are obtained as follows...

```
predict(mod_tob1, newdata = hs_data, type = "response")  
[1] -0.4232846  
[2] -13.4852170
```

The prediction for both Harry and Sally under the tobit model would be truncated to 0 days.

Chapter 19

Modeling an Ordinal Multi-Categorical Outcome in Ohio SMART

In this chapter, I use an ordered multi-categorical outcome (self-reported general health: `genhealth` = 1 = Excellent / 2 = Very Good / 3 = Good / 4 = Fair / 5 = Poor) to demonstrate regression models for such outcomes.

Methods discussed in the chapter include the following:

- Tabulating and Plotting Cross-Categorical Data (Cross-Tabulations)
- Proportional Odds Logistic Regression using `polr`
- Proportional Odds Logistic Regression using `lrm`
- Interpreting these models and Displaying their fit
- Assessing the assumption of Proportional Odds

19.1 Where to Read this Chapter

- The current version of this chapter is available for you to read at this link.

Some additional information on the chapter can be found in this README file.

Chapter 20

Analyzing Literary Styles with Multinomial Logistic Regression

20.1 The Authorship Example

This example is based on the work of Jeffrey S. Simonoff (2003) *Analyzing Categorical Data* in Chapter 10. Related data and R code are available at <http://people.stern.nyu.edu/jsimonof/AnalCatData/Splus/>. Meanwhile, the data set and analysis are based on the work of Peng RD and Hengartner NW (2002) Quantitative analysis of literary styles, *The American Statistician*, 56, 175-185.

The `authorship.csv` data file contains 841 rows. Each row describes a block of text that contains 1700 total words from one of several books by four authors: Jane Austen (samples from 7 books), Jack London (6 books), John Milton (2 books), or William Shakespeare (12 books). The data include counts within the blocks of text of 69 function words, such as “a”, “by”, “no”, “that” and “with”. The goal of our analysis, mirroring that of Simonoff, will be to use the incidence of these function words to build a model that distinguishes the authors.

```
authorship$Author <- factor(authorship$Author,
  levels = c("Shakespeare", "Austen", "London", "Milton"))

authorship

# A tibble: 841 x 71
# ... with 831 more rows, and 60 more variables: be <int>, been <int>,
#   but <int>, by <int>, can <int>, do <int>, down <int>, even <int>,
#   every <int>, `for` <int>, from <int>, had <int>, has <int>,
#   have <int>, her <int>, his <int>, `if` <int>, `in` <int>, into <int>,
```

```
#  is <int>, it <int>, its <int>, may <int>, more <int>, must <int>,
#  my <int>, no <int>, not <int>, now <int>, of <int>, on <int>,
#  one <int>, only <int>, or <int>, our <int>, should <int>, so <int>,
#  some <int>, such <int>, than <int>, that <int>, the <int>,
#  their <int>, then <int>, there <int>, things <int>, this <int>,
#  to <int>, up <int>, upon <int>, was <int>, were <int>, what <int>,
#  when <int>, which <int>, who <int>, will <int>, with <int>,
#  would <int>, your <int>
```

To-morrow, and to-morrow, and to-morrow, Creeps in this petty pace from day to day, To the last syllable of recorded time; And all our yesterdays have lighted fools The way to dusty death. Out, out, brief candle! Life's but a walking shadow, a poor player, That struts and frets his hour upon the stage, And then is heard no more. It is a tale Told by an idiot, full of sound and fury, Signifying nothing.

20.2 Focus on 11 key words

Again, following Simonoff, we will focus on 11 words from the set of 69 potential predictors in the data, specifically...

- “be”, “been”, “had”, “it”, “may”, “not”, “on”, “the”, “upon”, “was” and “which”

```
auth2 <- authorship %>%
  select(Author, BookID, be, been, had, it, may, not,
         on, the, upon, was, which)
```

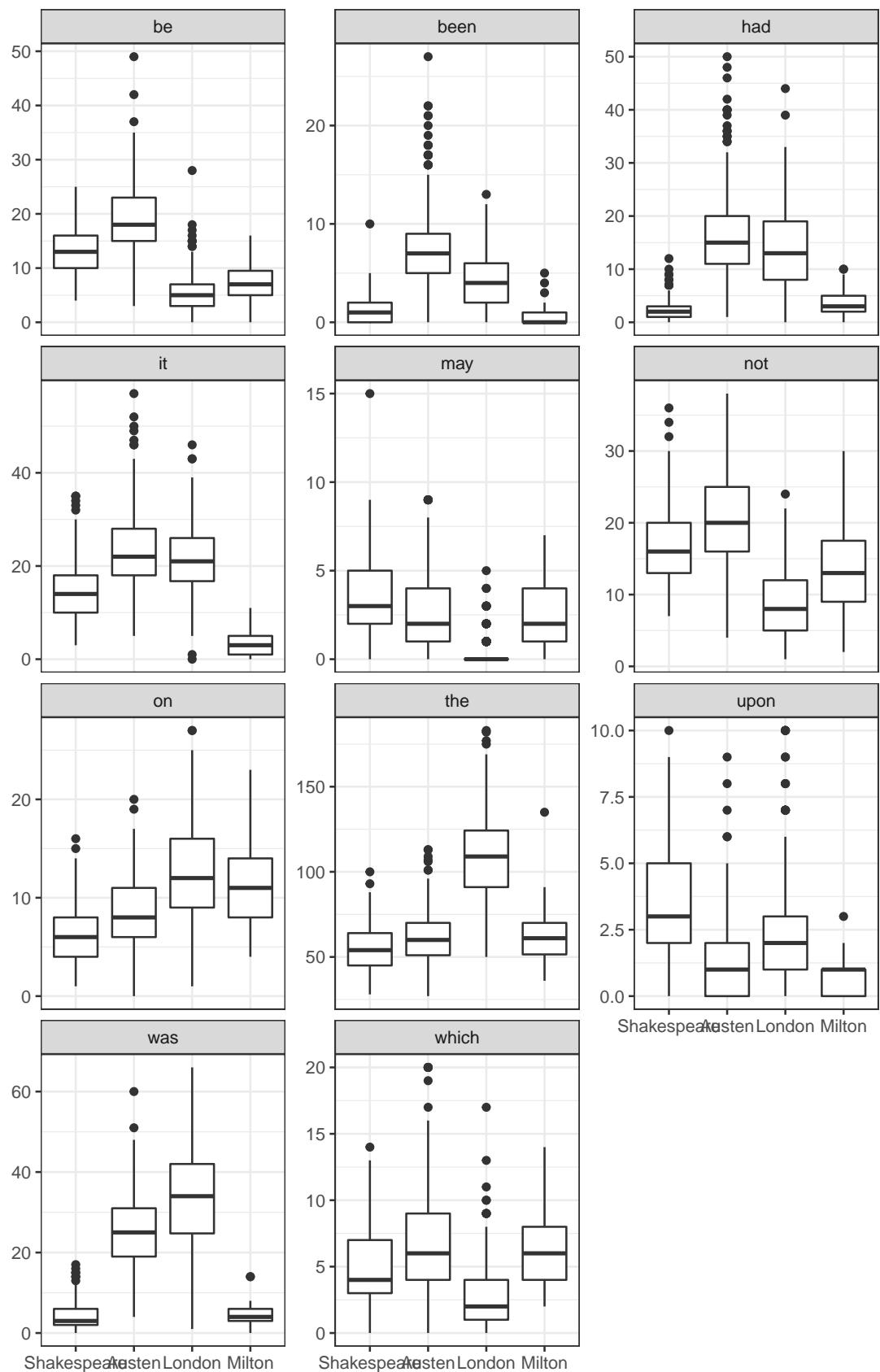
```
auth2.long <- auth2 %>%
  gather("word", "n", 3:13)
```

```
auth2.long
```

```
# A tibble: 9,251 x 4
  Author BookID word      n
  <fct>   <int> <chr> <int>
  1 Austen     1 be      13
  2 Austen     1 be      9
  3 Austen     1 be     23
  4 Austen     1 be     20
  5 Austen     1 be     16
  6 Austen     1 be     12
  7 Austen     1 be     11
  8 Austen     1 be     21
  9 Austen     1 be      7
 10 Austen    1 be     14
# ... with 9,241 more rows
```

20.2.1 Side by Side Boxplots

```
ggplot(auth2.long, aes(x = Author, y = n)) +
  geom_boxplot() +
  facet_wrap(~ word, ncol = 3, scales = "free_y") +
  theme_bw() +
  labs(x = "", y = "")
```



Oh! do not attack me with your watch. A watch is always too fast or too slow. I cannot be dictated to by a watch.

20.3 A Multinomial Logistic Regression Model

Let's start with a multinomial model to predict Author on the basis of these 11 key predictors, using the `multinom` function from the `nnet` package.

```
authnom1 <- multinom(Author ~ be + been + had + it + may + not + on +
                      the + upon + was + which, data=authorship,
                      maxit=200)
```

```
# weights:  52 (36 variable)
initial  value 1165.873558
iter  10 value 293.806160
iter  20 value 273.554538
iter  30 value 192.309644
iter  40 value 71.091334
iter  50 value 48.419335
iter  60 value 46.808141
iter  70 value 46.184752
iter  80 value 46.026162
iter  90 value 45.932823
iter 100 value 45.897793
iter 110 value 45.868017
iter 120 value 45.863256
final  value 45.863228
converged
```

```
summary(authnom1)
```

Call:

```
multinom(formula = Author ~ be + been + had + it + may + not +
          on + the + upon + was + which, data = authorship, maxit = 200)
```

Coefficients:

	(Intercept)	be	been	had	it	may	not	on	the	upon	was
Austen	-15.504834	0.48974946	0.5380318	0.4620513	0.00388835						
London	-14.671720	-0.07497073	0.1733116	0.4842272	0.08674782						
Milton	-1.776866	-0.10891178	-0.9127155	0.5319573	-0.82046587						
		may	not	on	the	upon	was				
Austen	-0.15025084	-0.08861462	0.5967404	-0.02361614	-2.119001	0.7021371					
London	-0.01590702	-0.32567063	0.5749969	0.12039782	-1.914428	0.6767581					
Milton	-0.06760436	0.05575887	0.5198173	0.08739368	-2.042475	0.3048202					
		which									
Austen	0.10370827										
London	-0.59121054										
Milton	-0.05939104										

Std. Errors:

	(Intercept)	be	been	had	it	may
Austen	4.892258	0.1643694	0.3117357	0.2695081	0.09554376	0.3008164
London	5.372898	0.1916618	0.3308759	0.2812555	0.11355697	0.4946804
Milton	5.417300	0.1613282	0.5910561	0.3187304	0.23015421	0.3500753

```

not      on      the      upon      was      which
Austen 0.1078329 0.2213827 0.03457288 0.6426484 0.1808681 0.1646472
London 0.1440760 0.2251642 0.03088881 0.7072129 0.1768901 0.2542886
Milton 0.1309306 0.2223575 0.04783646 0.6436399 0.1820885 0.2111105

```

Residual Deviance: 91.72646

AIC: 163.7265

20.3.1 Testing Model 1

```

z1 <- summary(authnom1)$coefficients/summary(authnom1)$standard.errors
round(z1,2)

```

```

(Intercept)    be   been   had   it    may   not   on    the   upon
Austen       -3.17  2.98  1.73  1.71  0.04 -0.50 -0.82  2.70 -0.68 -3.30
London        -2.73 -0.39  0.52  1.72  0.76 -0.03 -2.26  2.55  3.90 -2.71
Milton        -0.33 -0.68 -1.54  1.67 -3.56 -0.19  0.43  2.34  1.83 -3.17
                           was which
Austen        3.88  0.63
London        3.83 -2.32
Milton        1.67 -0.28

```

```

p1 <- (1 - pnorm(abs(z1), 0, 1)) * 2
pander(round(p1,3))

```

Table 20.1: Table continues below

	(Intercept)	be	been	had	it	may	not
Austen	0.002	0.003	0.084	0.086	0.968	0.617	0.411
London	0.006	0.696	0.6	0.085	0.445	0.974	0.024
Milton	0.743	0.5	0.123	0.095	0	0.847	0.67

	on	the	upon	was	which
Austen	0.007	0.495	0.001	0	0.529
London	0.011	0	0.007	0	0.02
Milton	0.019	0.068	0.002	0.094	0.778

Simonoff suggests that “been” and “may” can be dropped. What do we think?

The proper function of man is to live, not to exist. I shall not waste my days in trying to prolong them. I shall use my time.

20.4 Model 2

```

authnom2 <- multinom(Author ~ be + had + it + not + on +
                      the + upon + was + which, data=authorship,
                      maxit=200)

```

```

# weights: 44 (30 variable)
initial value 1165.873558

```

```

iter 10 value 304.985478
iter 20 value 285.428679
iter 30 value 143.301103
iter 40 value 54.589791
iter 50 value 52.140470
iter 60 value 51.421454
iter 70 value 51.012790
iter 80 value 50.888718
iter 90 value 50.834262
iter 100 value 50.743136
final value 50.743111
converged

```

```
summary(authnom2)
```

Call:

```
multinom(formula = Author ~ be + had + it + not + on + the +
upon + was + which, data = authorship, maxit = 200)
```

Coefficients:

	(Intercept)	be	had	it	not	on
Austen	-16.55647	0.45995950	0.6698612	0.02621612	-0.03684654	0.4676716
London	-16.06419	-0.13378141	0.6052164	0.10517792	-0.27934022	0.4958923
Milton	-2.22344	-0.07031256	0.1737526	-0.81984885	0.05444678	0.5363108
	the	upon	was	which		
Austen	-0.001852454	-1.950761	0.6543956	0.06363998		
London	0.128565811	-1.643829	0.6418607	-0.54690144		
Milton	0.074236636	-1.762533	0.2932065	-0.08748272		

Std. Errors:

	(Intercept)	be	had	it	not	on
Austen	4.723001	0.1293729	0.2201823	0.08657746	0.08771157	0.1949021
London	5.202732	0.1587639	0.2306803	0.10117217	0.11608348	0.2072383
Milton	4.593806	0.1499103	0.2057258	0.21551377	0.12103678	0.1895226
	the	upon	was	which		
Austen	0.02945139	0.5620273	0.1524982	0.1466250		
London	0.02739965	0.6219927	0.1512911	0.2087120		
Milton	0.04463721	0.6246766	0.1601393	0.1928361		

Residual Deviance: 101.4862

AIC: 161.4862

20.4.1 Comparing Model 2 to Model 1

```
anova(authnom1, authnom2)
```

Likelihood ratio tests of Multinomial Models

Response: Author

		Model
1	be + had + it + not + on + the + upon + was + which	
2	be + been + had + it + may + not + on + the + upon + was + which	
	Resid. df Resid. Dev Test Df LR stat. Pr(Chi)	
1	2493 101.48622	

```
2      2487  91.72646 1 vs 2      6 9.759767 0.1351402
```

20.4.2 Testing Model 2

```
z2 <- summary(authnom2)$coefficients/summary(authnom2)$standard.errors
round(z2,2)
```

	(Intercept)	be	had	it	not	on	the	upon	was	which
Austen	-3.51	3.56	3.04	0.30	-0.42	2.40	-0.06	-3.47	4.29	0.43
London	-3.09	-0.84	2.62	1.04	-2.41	2.39	4.69	-2.64	4.24	-2.62
Milton	-0.48	-0.47	0.84	-3.80	0.45	2.83	1.66	-2.82	1.83	-0.45

```
p2 <- (1 - pnorm(abs(z2), 0, 1)) * 2
round(p2,3)
```

	(Intercept)	be	had	it	not	on	the	upon	was	which
Austen	0.000	0.000	0.002	0.762	0.674	0.016	0.950	0.001	0.000	0.664
London	0.002	0.399	0.009	0.299	0.016	0.017	0.000	0.008	0.000	0.009
Milton	0.628	0.639	0.398	0.000	0.653	0.005	0.096	0.005	0.067	0.650

20.4.3 A little history

Simonoff has an interesting note: Consider the lifetimes of these four authors:

- William Shakespeare was born in 1564 and died in 1616
- John Milton was born in 1608 (44 years after Shakespeare) and died in 1674
- Jane Austen was born in 1775 (211 years after Shakespeare) and died in 1817
- Jack London was born in 1876 (312 years after Shakespeare) and died in 1916

How many significant coefficients does each author display relative to Shakespeare?

20.5 Classification Table

How well does this model (model 2) distinguish these authors based on blocks of 1700 words of text?

```
table(authorship$Author, predict(authnom2))
```

	Shakespeare	Austen	London	Milton
Shakespeare	168	3	1	1
Austen	4	308	5	0
London	0	1	294	1
Milton	2	0	1	52

Based on this classification table, I'd say it does a nice job. Almost 98% of the blocks of text are correctly classified.

Fly, envious Time, till thou run out thy race; Call on the lazy leaden-stepping hours, Whose speed is but the heavy plummet's pace; And glut thyself with what thy womb devours, Which is no more than what is false and vain, And merely mortal dross; So little is our loss, So little is thy gain. For when, as each thing bad thou hast entomb'd And last of all thy greedy self consumed, Then long Eternity shall greet our bliss, With an individual kiss; And Joy shall overtake us, as a flood, When every thing that is sincerely good, And perfectly divine, With truth, and peace, and love, shall ever shine, About the supreme throne Of Him, to whose happy-making sight, alone,

When once our heavenly-guided soul shall climb,
Then all this earthly grossness quit, Attired
with stars, we shall for ever sit, Triumphing over Death, and Chance, and thee, O Time!

20.6 Probability Curves based on a Single Predictor

In situations where only one predictor is used, we can develop nice plots of estimated probabilities for each group as a function of the predictor. Suppose we look at the single word “been” (note that this was left out of Model 2.)

Note that the possible values for counts of “been” in the data range from 0 to 27...

```
summary(authorship$been)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.000	2.000	4.000	4.614	7.000	27.000

Now, we’ll build a model to predict the author based solely on the counts of the word “been”.

```
authnom3 <- multinom(Author ~ been,
                      data=authorship, maxit=200)
```

```
# weights: 12 (6 variable)
initial value 1165.873558
iter 10 value 757.915093
iter 20 value 755.454631
final value 755.454551
converged
```

Next, we’ll build a grid of the predicted log odds for each author (as compared to Shakespeare) using the fitted coefficients. The grid will cover every possible value from 0 to 27, increasing by 0.1, using the following trick in R.

```
beengrid <- cbind(1,c(0:270)/10)
austenlogit <- beengrid %*% coef(authnom3)[1,]
londonlogit <- beengrid %*% coef(authnom3)[2,]
miltonlogit <- beengrid %*% coef(authnom3)[3,]
```

Next, we’ll use that grid of logit values to estimate the fitted probabilities for each value of “been” between 0 and 27.

```
austenprob <- exp(austenlogit) /
  (exp(austenlogit) + exp(londonlogit) +
   exp(miltonlogit) + 1)
londonprob <- exp(londonlogit) /
  (exp(austenlogit) + exp(londonlogit) +
   exp(miltonlogit) + 1)
miltonprob <- exp(miltonlogit) /
  (exp(austenlogit) + exp(londonlogit) +
   exp(miltonlogit) + 1)
shakesprob <- 1 - austenprob - londonprob - miltonprob

been_dat <- data_frame(been_count = beengrid[,2],
                        austen = austenprob[,1],
                        london = londonprob[,1],
                        milton = miltonprob[,1],
                        shakespeare = shakesprob[,1])
been_dat
```

```
# A tibble: 271 x 5
  been_count austen london milton shakespeare
  <dbl>     <dbl>   <dbl>    <dbl>      <dbl>
1 0.        0.0258  0.136   0.285     0.553
2 0.100    0.0288  0.147   0.272     0.553
3 0.200    0.0321  0.158   0.258     0.551
4 0.300    0.0357  0.171   0.245     0.548
5 0.400    0.0396  0.184   0.232     0.545
6 0.500    0.0438  0.197   0.219     0.540
7 0.600    0.0484  0.211   0.207     0.534
8 0.700    0.0534  0.225   0.195     0.527
9 0.800    0.0587  0.240   0.183     0.518
10 0.900   0.0644  0.256   0.171     0.509
# ... with 261 more rows
```

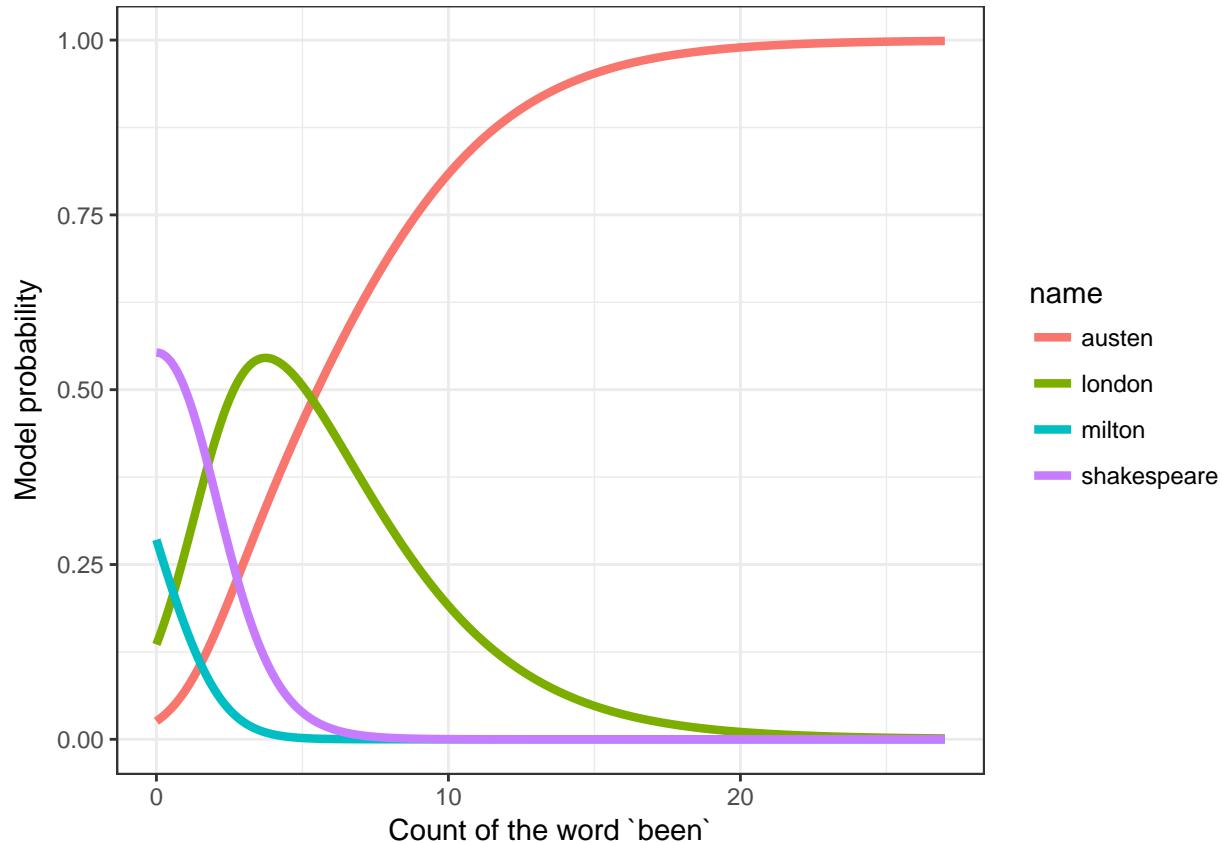
Now, we gather the data by author name and probability

```
been_dat_long <- been_dat %>%
  gather("name", "prob", 2:5)
been_dat_long
```

```
# A tibble: 1,084 x 3
  been_count name      prob
  <dbl>     <chr>    <dbl>
1 0.        austen 0.0258
2 0.100    austen 0.0288
3 0.200    austen 0.0321
4 0.300    austen 0.0357
5 0.400    austen 0.0396
6 0.500    austen 0.0438
7 0.600    austen 0.0484
8 0.700    austen 0.0534
9 0.800    austen 0.0587
10 0.900   austen 0.0644
# ... with 1,074 more rows
```

20.6.1 Produce the Plot of Estimated Probabilities based on “been” counts

```
ggplot(bnnn_dat_long, aes(x = been_count, y = prob,
                           col = name)) +
  geom_line(size = 1.5) +
  theme_bw() +
  labs(x = "Count of the word `been`",
       y = "Model probability")
```

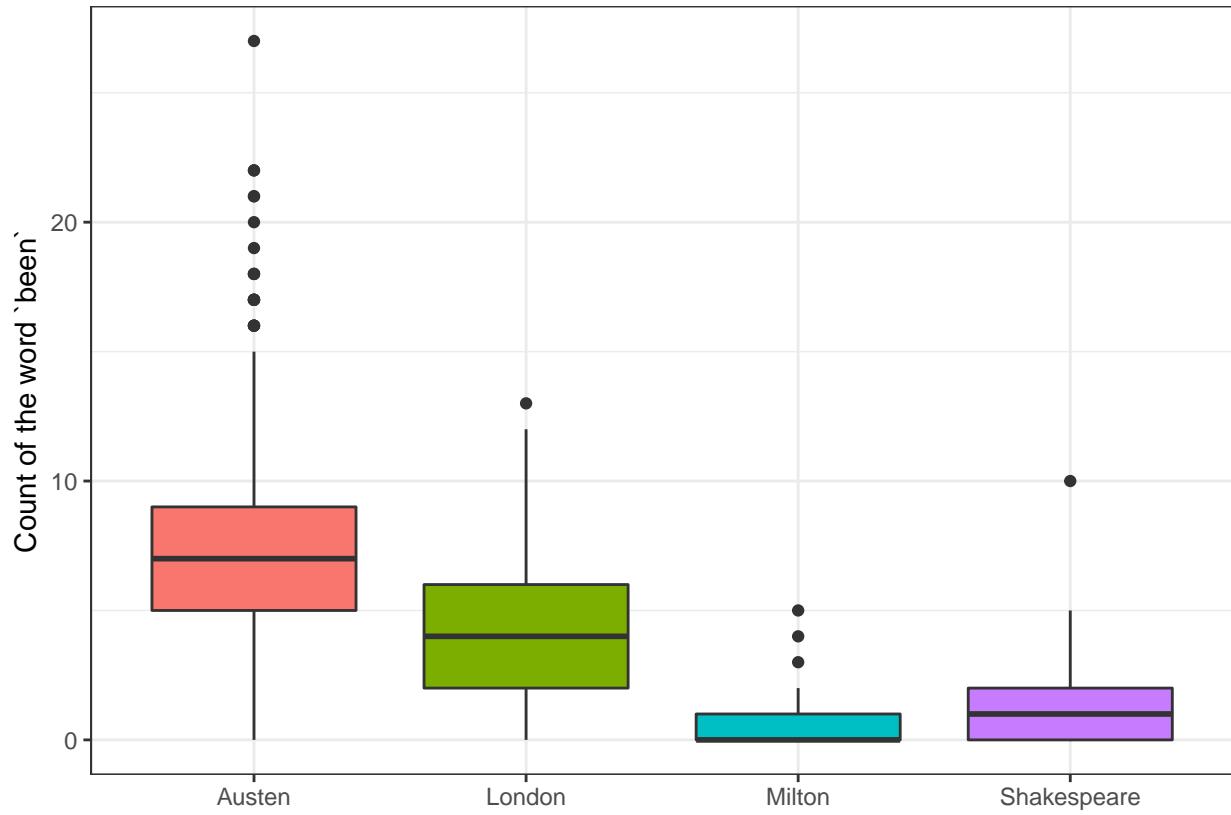


20.6.2 Boxplot of “been” counts

Compare this to what we see in the raw counts of the word “been”.

```
been.long <- filter(auth2.long, word == "been")
been.long$Auth <- fct_relevel(been.long$Author,
  "Austen", "London", "Milton", "Shakespeare")
# releveling to make the colors match the model plot

ggplot(been.long, aes(x = Auth, y = n, fill = Auth)) +
  geom_boxplot() +
  guides(fill = FALSE) +
  theme_bw() +
  labs(x = "", y = "Count of the word `been`")
```



20.6.3 Quote Sources

1. To-morrow, and to-morrow, and to-morrow ... Shakespeare *Macbeth* Act 5.
2. Oh! do not attack me with your watch. ... Jane Austen *Mansfield Park*
3. The proper function of man is to live, not to exist. ... Jack London *The Bulletin* San Francisco 1916-12-02.
4. Fly, envious Time, till thou run out thy race ... John Milton *On Time*

Bibliography

- Barnett, P. A., Roman-Golstein, S., Ramsey, F., et al. (1995). Differential permeability and quantitative mr imaging of a human lung carcinoma brain xenograft in the nude rat. *American Journal of Pathology*, 146(2):436–449.
- Berkhemer, O. A., Fransen, P. S. S., Buemer, D., et al. (2015). A randomized trial of intraarterial treatment for acute ischemic stroke. *New England Journal of Medicine*, 372:11–20.
- Faraway, J. J. (2006). *Extending the Linear Model with R*. CRC Press, Boca Raton, FL.
- Faraway, J. J. (2015). *Linear Models with R*. CRC Press, Boca Raton, FL, second edition.
- Gelman, A. and Hill, J. (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, New York.
- Harrell, F. E. (2001). *Regression Modeling Strategies*. Springer, New York.
- Harrell, F. E. (2018). *Regression Modeling Strategies Course Notes*.
- Hastie, T., Tibshirani, R., and Frideman, J. H. (2001). *The Elements of Statistical Learning*. Springer, New York.
- Hurvich, C. M. and Tsai, C.-L. (1989). Regression and time series model selection in small samples. *Biometrika*, 76:297–307.
- Kim, H.-Y. (2014). Statistical notes for clinical researchers: Two-way analysis of variance (anova) - exploring possible interaction between factors. *Restorative Dentistry & Endodontics*, 39(2):143–147.
- Leeb, H. and Potscher, B. M. (2005). Model selection and inference: Facts and fiction. *Econometric Theory*, 21(1):21–59.
- Long, J. S. (1997). *Regression Models for categorical and limited dependent variables*. Sage Publications, Thousand Oaks, CA.
- McDonald, G. C. and Schwing, R. C. (1973). Instabilities of regression estimates relating air pollution to mortality. *Technometrics*, 15(3):463–481.
- Peduzzi, P., Concato, J., Kemper, E., Holford, T. R., and Feinstein, A. R. (1996). A simulation study of the number of events per variable in logistic regression analysis. *Journal of Clinical Epidemiology*, 49(12):1373–1379.
- Ramsey, F. L. and Schafer, D. W. (2002). *The Statistical Sleuth: A Course in Methods of Data Analysis*. Duxbury, Pacific Grove, CA, second edition edition.
- Riffenburgh, R. H. (2006). *Statistics in Medicine*. Elsevier Academic Press, Burlington, MA, second edition edition.
- Rosenbaum, P. R. (2017). *Observation and Experiment: An Introduction to Causal Inference*. Harvard University Press, Cambridge, MA.

- Roy, D., Talajic, M., Nattel, S., et al. (2008). Rhythm control versus rate control for atrial fibrillation and heart failure. *New England Journal of Medicine*, 358:2667–2677.
- Stamey, T.A., J. K. I. J. et al. (1989). Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate: II. radical prostatectomy treated patients. *Journal of Urology*, 141(5):1076–1083.
- Tolaney, S. M., Barry, W. T., Chau, T. D., et al. (2015). Adjuvant paclitaxel and trastuzumab for node-negative, her2-positive breast cancer. *New England Journal of Medicine*, 372:134–141.
- Vittinghoff, E., Glidden, D. V., Shiboski, S. C., and McCulloch, C. E. (2012). *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. Springer-Verlag, Inc., second edition edition.