Chapter 12

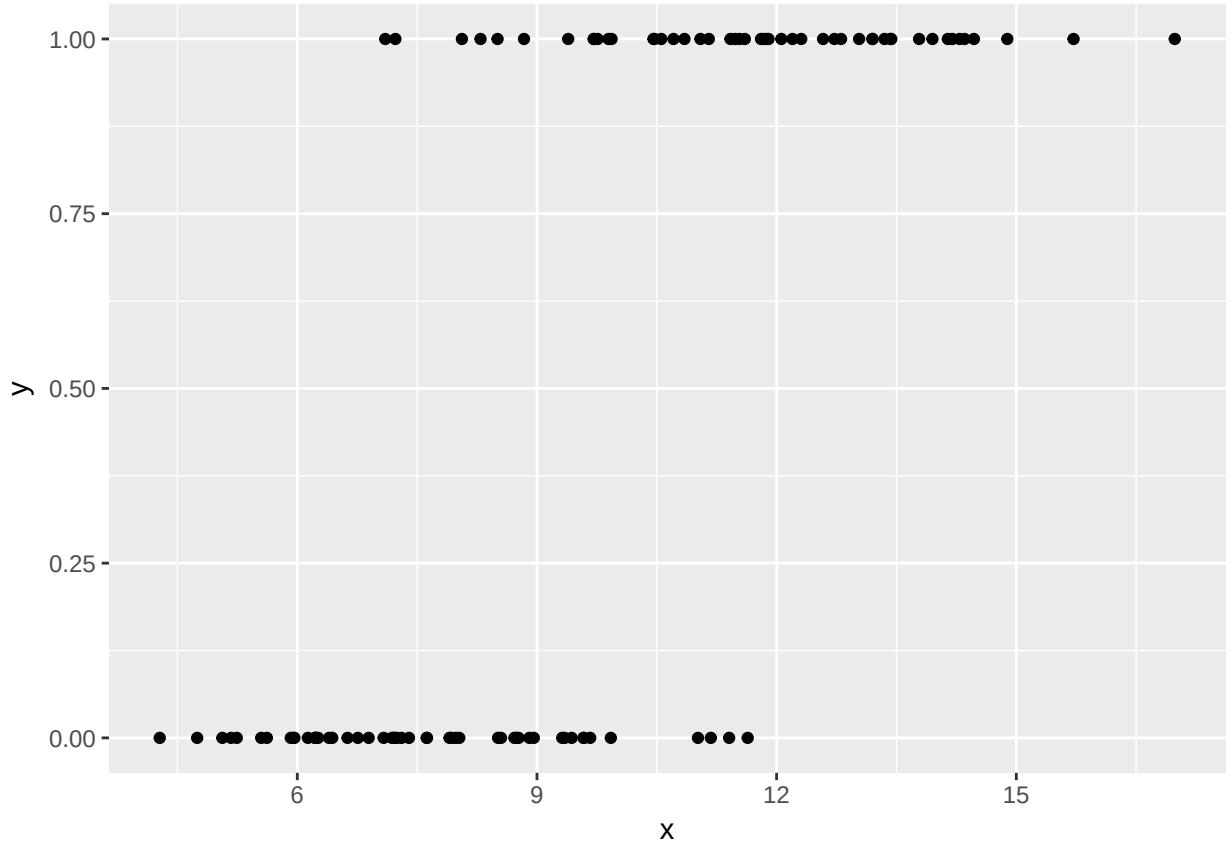
Logistic Regression: The Foundations

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).

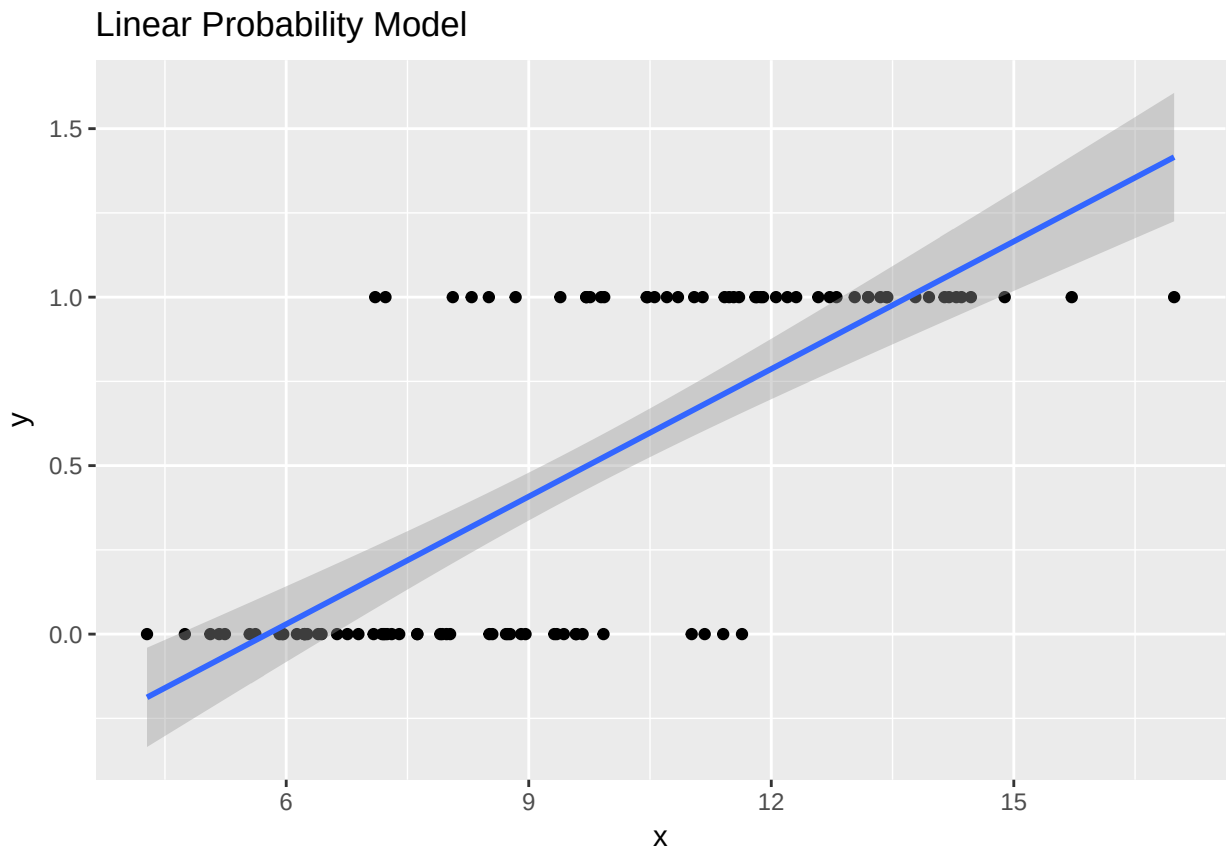
One approach to doing this would be a linear probability model, as follows:

```
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 $X_1, X_2, ..., X_k$.

$$g(\mu) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k$$

In the case of a logistic regression model,

- the mean μ of our 0/1 outcome is represented by π which describes the probability of a "1" outcome.
- the linking function we use in logistic regression makes use of the logit function, which is built on the natural logarithm.

12.4 The Link Function

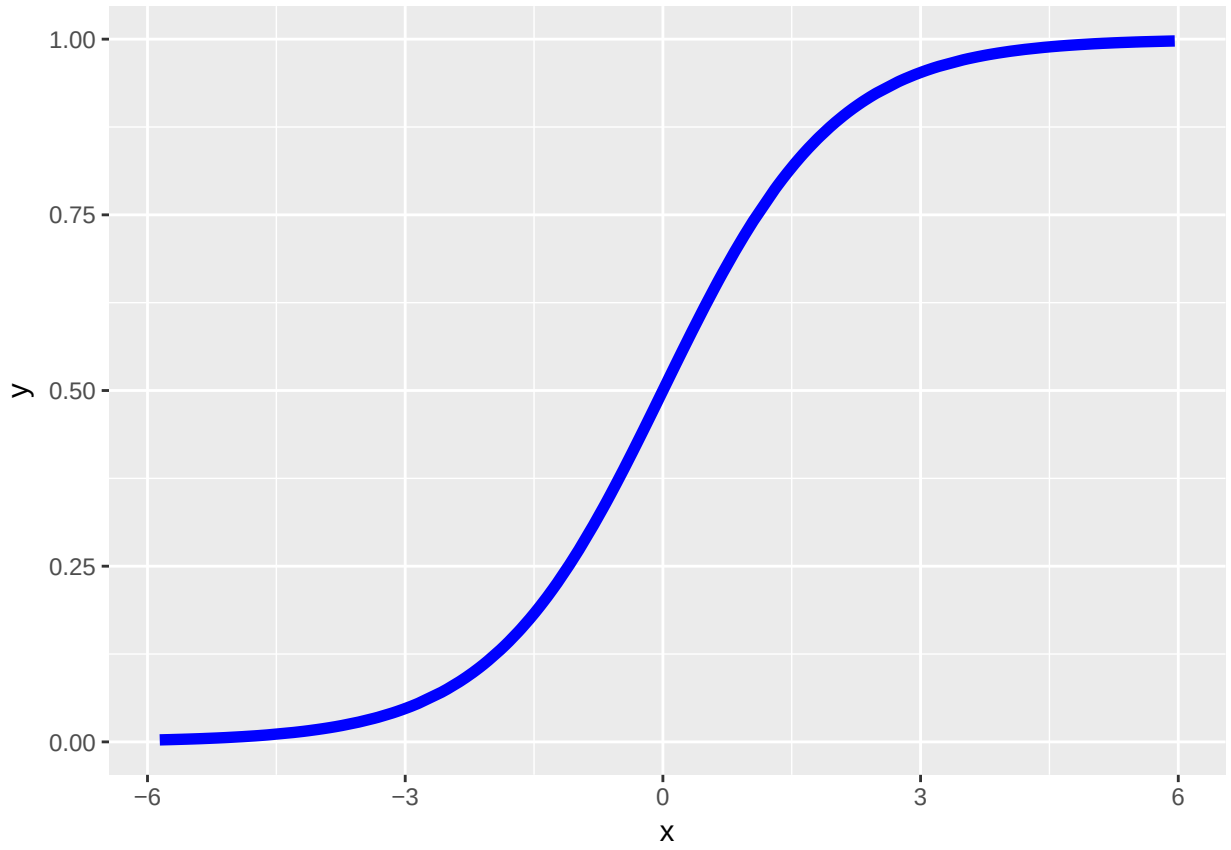
Logistic regression is a non-linear regression approach, since the equation for the mean of the 0/1 Y values conditioned on the values of our predictors $X_1, X_2, ..., X_k$ turns out to be non-linear in the β coefficients. Its nonlinearity, however, is solely found in its link function, hence the term *generalized* linear model.

The particular link function we use in logistic regression is called the **logit link**.

$$logit(\pi) = log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k$$

The inverse of the logit function is called the **logistic function**. If $\text{logit}(\pi) = \eta$, 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 $logit(\pi) < 0$.
- If our probability $\pi > 0.5$, then $logit(\pi) > 0$.
- Finally, if $\pi = 0.5$, then $logit(\pi) = 0$.

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:

$$logit(\pi) = log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k$$

The odds of a yes response (the odds that Y = 1) at the level $X_1, X_2, ..., X_k$ are:

$$Odds(Y = 1) = exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k)$$

The **probability** of a yes response (Pr(y = 1), or π) is just

$$\pi = Pr(Y = 1) = \frac{Odds(Y = 1)}{1 + Odds(Y = 1)} = \frac{exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k)}{1 + exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k)}$$

12.7 The Logistic Regression has non-constant variance

In ordinary least squares regression, the variance $Var(Y|X_1, X_2, ..., X_k) = \sigma^2$ is a constant that does not depend on the predictor values. This is not the case in logistic regression. The mean and variance specifications of the logistic regression model are quite different.

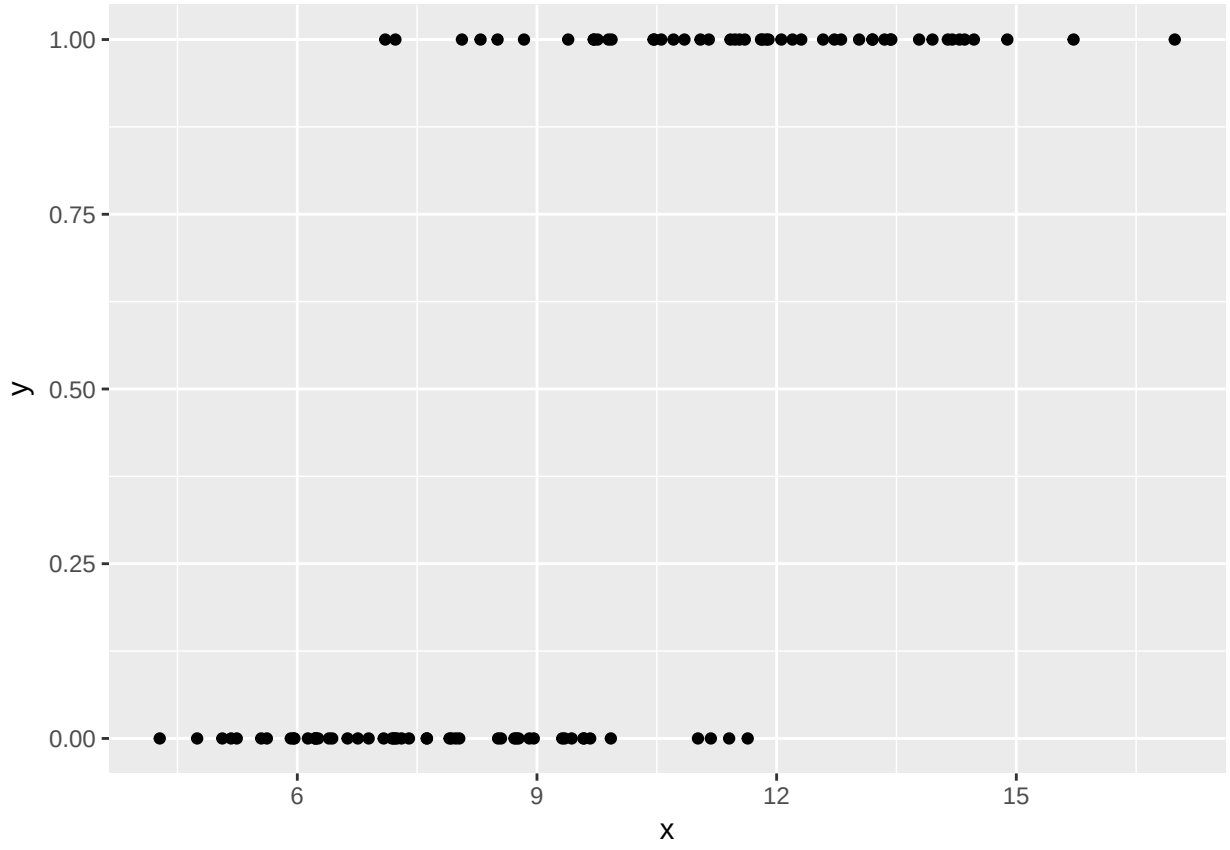
$$logit(\pi) = log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k \mu[Y|X_1, ..., X_k] = \pi, Var[Y|X_1, ..., X_k] = \pi(1-\pi)$$

The variance is now a function of the mean, and contains no additional parameter for us to estimate.

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:

$$logit(Pr(y=1)) = log\left(\frac{Pr(y=1)}{1-Pr(y=1)}\right) = -9.1955 + 0.9566x$$

We can exponentiate the results of this model to get to an equation about odds, and eventually, a prediction about probabilities. Suppose, for instance, that we are interested in the prediction when x = 12.

$$logit(Pr(y=1)|X=12) = log\left(\frac{Pr(y=1)}{1-Pr(y=1)}\right) = -9.1955 + 0.9566 * 12 = 2.2837$$

And we can also get this from the `predict` function applied to our model, although the `predict` approach retains a few more decimal places internally:

```
predict(model12b, newdata = data.frame(x = 12))
```

```
       1
2.284069
```

$$Odds(Y=1|X=12) = exp(-9.20 + 0.96 * 12) = exp(2.2837) = 9.812921$$

```
exp(predict(model12b, newdata = data.frame(x = 12)))
```

```
      1
9.81654
```

The estimated **probability** of a yes response (Pr(y = 1), or π) if x = 12 is just

$$\pi = Pr(Y=1|X=12) = \frac{Odds(Y=1|X=12)}{1+Odds(Y=1|X=12)} = \frac{exp(-9.20+0.96x)}{1+exp(-9.20+0.96x)} = \frac{9.812921}{1+9.812921} = 0.908$$

Does this work out?

```
exp(predict(model12b, newdata = data.frame(x = 12))) /
    (1 + exp(predict(model12b, newdata = data.frame(x = 12))))
```

```
       1
0.907549
```

which is also directly available by running `predict` with `type = "response"`.

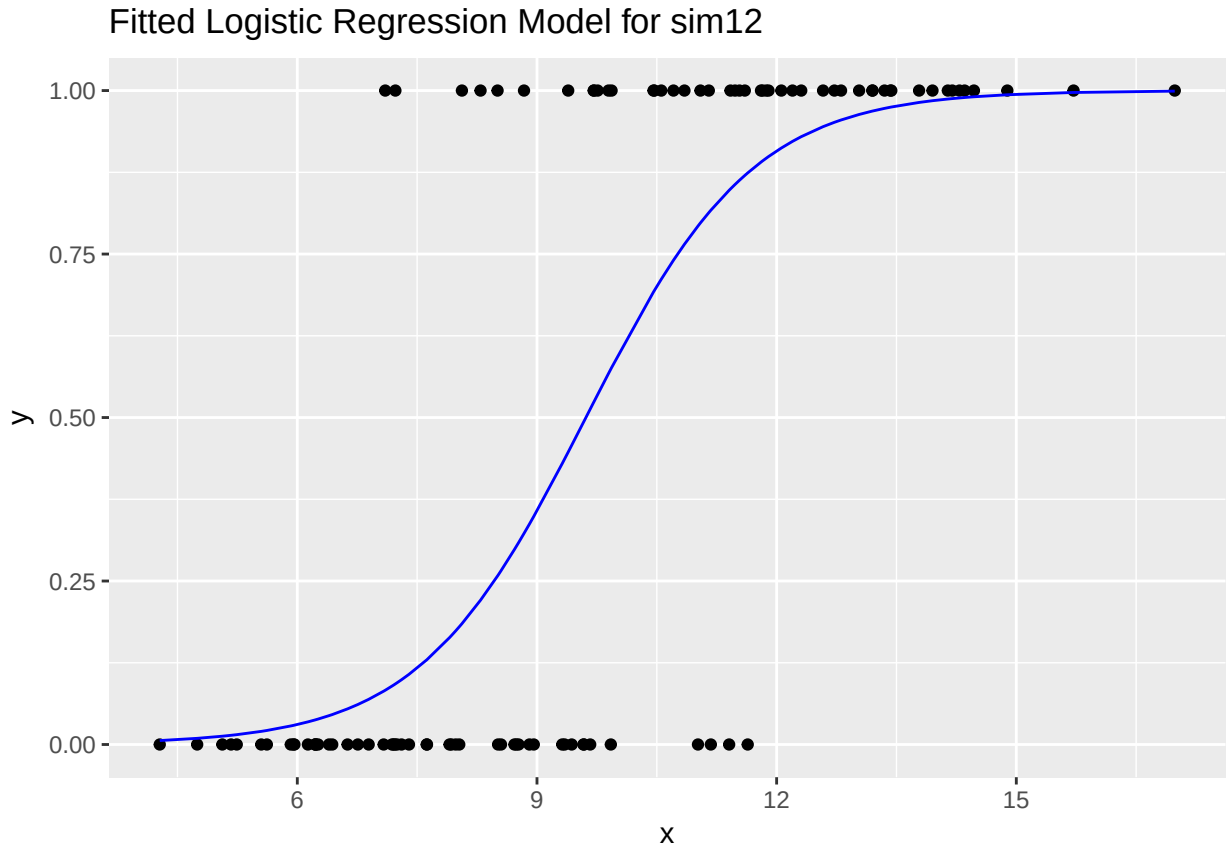
```
predict(model12b, newdata = data.frame(x = 12), type = "response")
```

```
       1
0.907549
```

12.9 Plotting the Logistic Regression Model

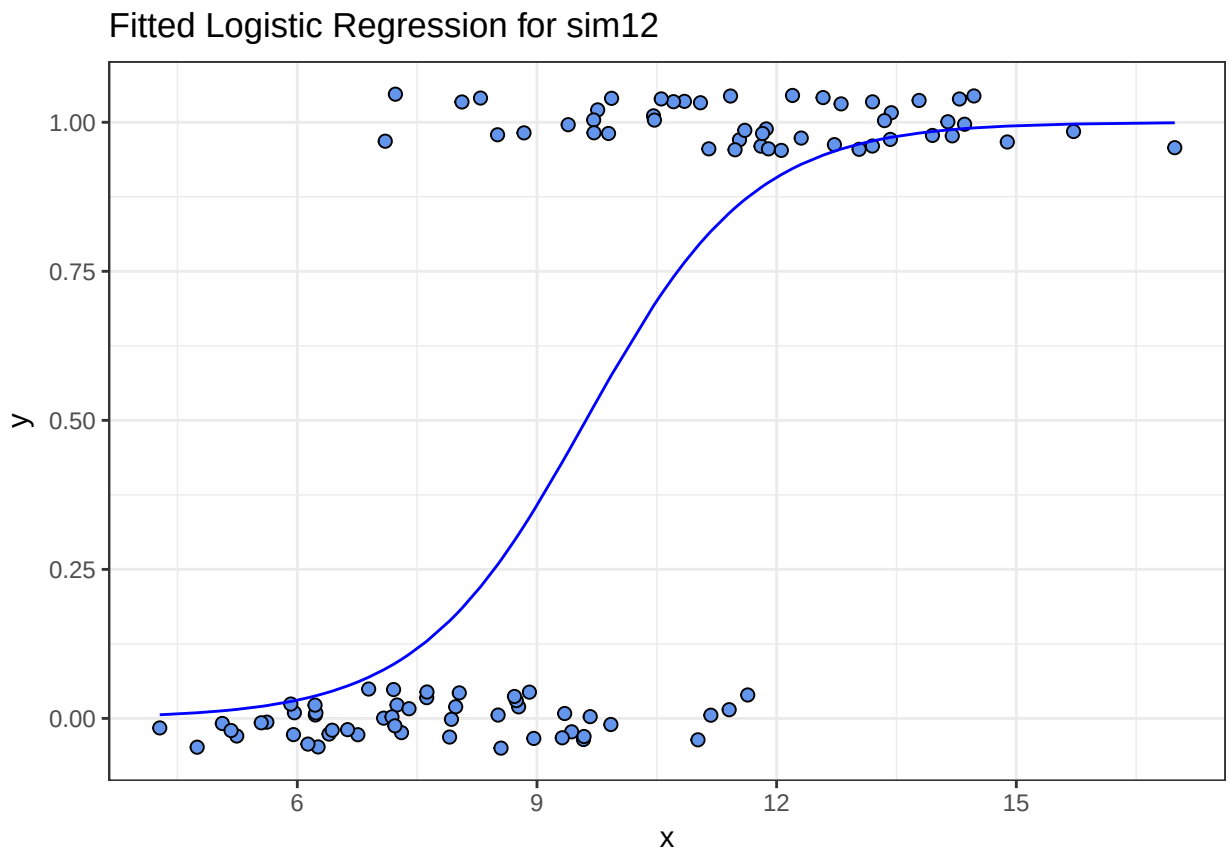
We can use the `augment` function from the `broom` package to get our fitted probabilities included in the data.

```
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 the number of candidate parameters might be reasonable so long as the smallest cell of interest (combination of an outcome and a split of the covariates) is 96 or more observations.
  - Peduzzi et al. (1996) suggest that if we let π be the smaller of the proportions of "yes" or "no" cases in the population of interest, and k be the number of inputs under consideration, then $N = 10k/\pi$ is the minimum number of cases to include, except that if N < 100 by this standard, you should increase it to 100, according to Long (1997).
    * That suggests that if you have an outcome that happens 10% of the time, and you are running a model with 3 predictors, then you could get away with $(10 \times 3)/(0.10) = 300$ observations, but if your outcome happened 40% of the time, you could get away with only $(10 \times 3)/(0.40) = 75$ observations, which you'd round up to 100.

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:

$$logit(died = 1) = log\left(\frac{Pr(died = 1)}{1 - Pr(died = 1)}\right) = \beta_0 + \beta_1 x = -4.4337 + 0.7417 \times resection$$

The predicted log odds of death for a subject with a resection of 4 cm is:

$$log\left(\frac{Pr(died = 1)}{1 - Pr(died = 1)}\right) = -4.4337 + 0.7417 \times 4 = -1.467$$

The predicted odds of death for a subject with a resection of 4 cm is thus:

$$\frac{Pr(died = 1)}{1 - Pr(died = 1)} = e^{-4.4337+0.7417\times4} = e^{-1.467} = 0.2306$$

Since the odds are less than 1, we should find that the probability of death is less than 1/2. With a little algebra, we see that the predicted probability of death for a subject with a resection of 4 cm is:

$$Pr(died = 1) = \frac{e^{-4.4337+0.7417\times4}}{1 + e^{-4.4337+0.7417\times4}} = \frac{e^{-1.467}}{1 + e^{-1.467}} = \frac{0.2306}{1.2306} = 0.187$$

In general, we can frame the model in terms of a statement about probabilities, like this:

$$Pr(died = 1) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}} = \frac{e^{-4.4337+0.7417\times resection}}{1 + e^{-4.4337+0.7417\times resection}}$$

and so by substituting in values for `resection`, we can estimate the model's fitted probabilities of death.

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:

$$log\left(\frac{Pr(Harrydied)}{1-Pr(Harrydied)}\right) = -4.4337+0.7417\times5 = -0.7253, log\left(\frac{Pr(Sallydied)}{1-Pr(Sallydied)}\right) = -4.4337+0.7417\times4 = -1.4667.$$

which implies that our estimated odds of death for Harry and Sally are:

$$Odds(Harrydied) = \frac{Pr(Harrydied)}{1-Pr(Harrydied)} = e^{-4.4337+0.7417\times5} = e^{-0.7253} = 0.4842 Odds(Sallydied) = \frac{Pr(Sallydied)}{1-Pr(Sallydied)} = e$$

and so the odds ratio is:

$$OR = \frac{Odds(Harrydied)}{Odds(Sallydied)} = \frac{0.4842}{0.2307} = 2.099$$

- If the odds ratio was 1, that would mean that Harry and Sally had the same estimated odds of death, and thus the same estimated probability of death, despite having different sizes of resections.
- Since the odds ratio is greater than 1, it means that Harry has a higher estimated odds of death than Sally, and thus that Harry has a higher estimated probability of death than Sally.
- If the odds ratio was less than 1, it would mean that Harry had a lower estimated odds of death than Sally, and thus that Harry had a lower estimated probability of death than Sally.

Remember that the odds ratio is a fraction describing two positive numbers (odds can only be non-negative) so that the smallest possible odds ratio is 0.

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*`logLik`.

```
glance(res_modA)
```

```
  null.deviance df.null    logLik      AIC     BIC deviance df.residual
1      101.9431     133 -44.74646 93.49292 99.2886 89.49292         132
```

The `glance` result also provides the AIC, and the BIC (Bayes Information Criterion), each of which is helpful in understanding comparisons between multiple models for the same outcome (with smaller values of either criterion indicating better models.) The AIC is based on the deviance, but penalizes you for making the model more complicated. The BIC does the same sort of thing but with a different penalty.

Again we see that we have a null deviance of 101.94 on 133 degrees of freedom. Including the `resection` information in the model decreased the deviance to 89.49 points on 132 degrees of freedom, so that's a decrease of 12.45 points while using one degree of freedom, a statistically significant reduction in deviance.

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$^{\text{th}}$ observation in a model predicting y_i (a binary variable), with the estimate being $\hat{\pi}_i$ is:

$$d_i = s_i \sqrt{-2[y_i log\hat{\pi}_i + (1 - y_i) log(1 - \hat{\pi}_i)]},$$

where s_i is 1 if $y_i = 1$ and $s_i = -1$ if $y_i = 0$.

Again, the sum of the deviance residuals is the deviance.

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

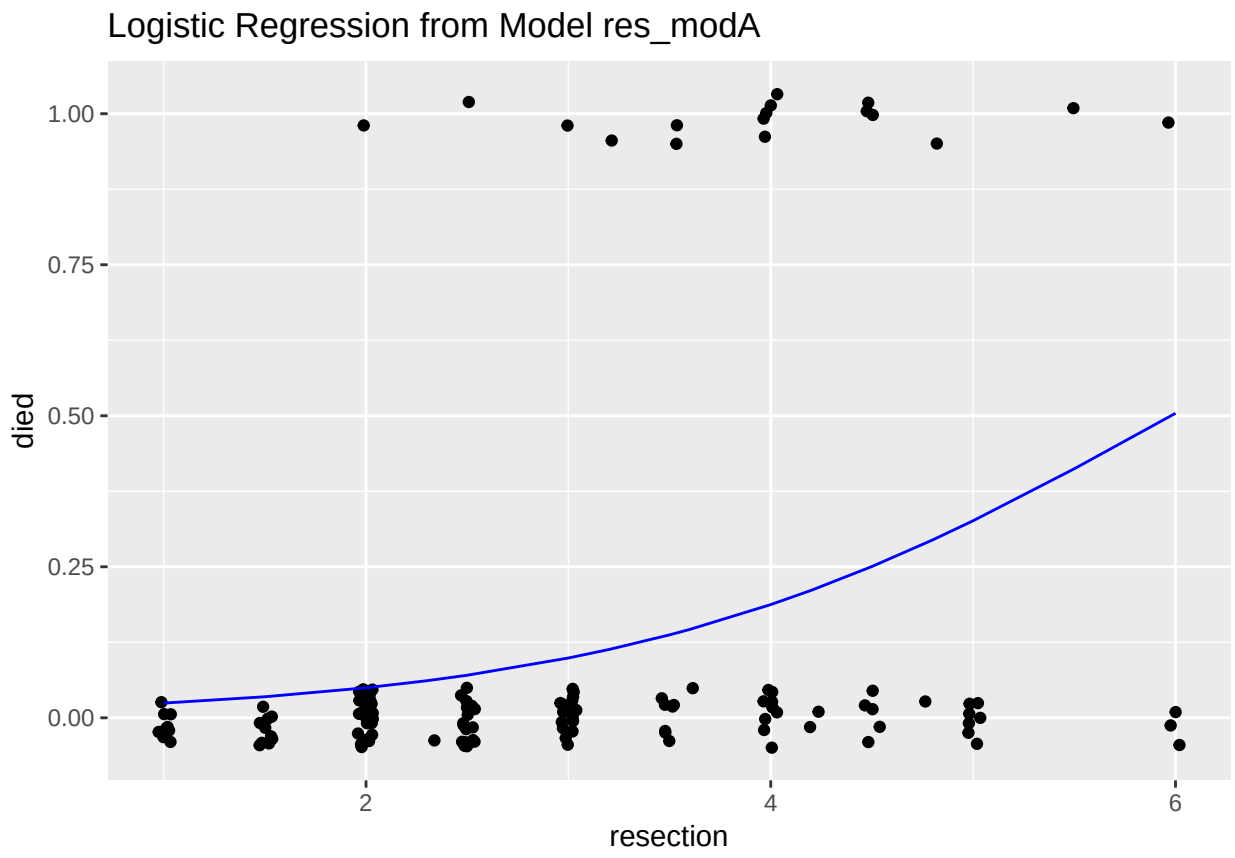
```
res_A_aug <- augment(res_modA, resect,
                     type.predict = "response")
head(res_A_aug)
```

```
  id age prior resection intubated died    .fitted    .se.fit     .resid
1  1  34     1       2.5         0    0 0.07046791 0.02562381 -0.3822929
2  2  57     0       5.0         0    0 0.32622637 0.08605551 -0.8886631
3  3  60     1       4.0         1    1 0.18740037 0.04269795  1.8300317
4  4  62     1       4.2         0    0 0.21104240 0.04871389 -0.6885386
5  5  28     0       6.0         1    1 0.50409637 0.14302982  1.1704596
6  6  52     0       3.0         0    0 0.09897375 0.02867196 -0.4565542
         .hat    .sigma      .cooksd .std.resid
1 0.010024061 0.8258481 0.0003876961 -0.3842235
2 0.033691765 0.8227475 0.0087350915 -0.9040227
3 0.011972088 0.8107264 0.0265893468  1.8410857
4 0.014252277 0.8243062 0.0019617278 -0.6934983
5 0.081835623 0.8196110 0.0477480056  1.2215077
6 0.009218619 0.8255581 0.0005157780 -0.4586733
```

This approach augments the `resect` data set with fitted, residual and other summaries of each observation's impact on the fit, using the "response" type of prediction, which yields the fitted probabilities in the `.fitted` column.

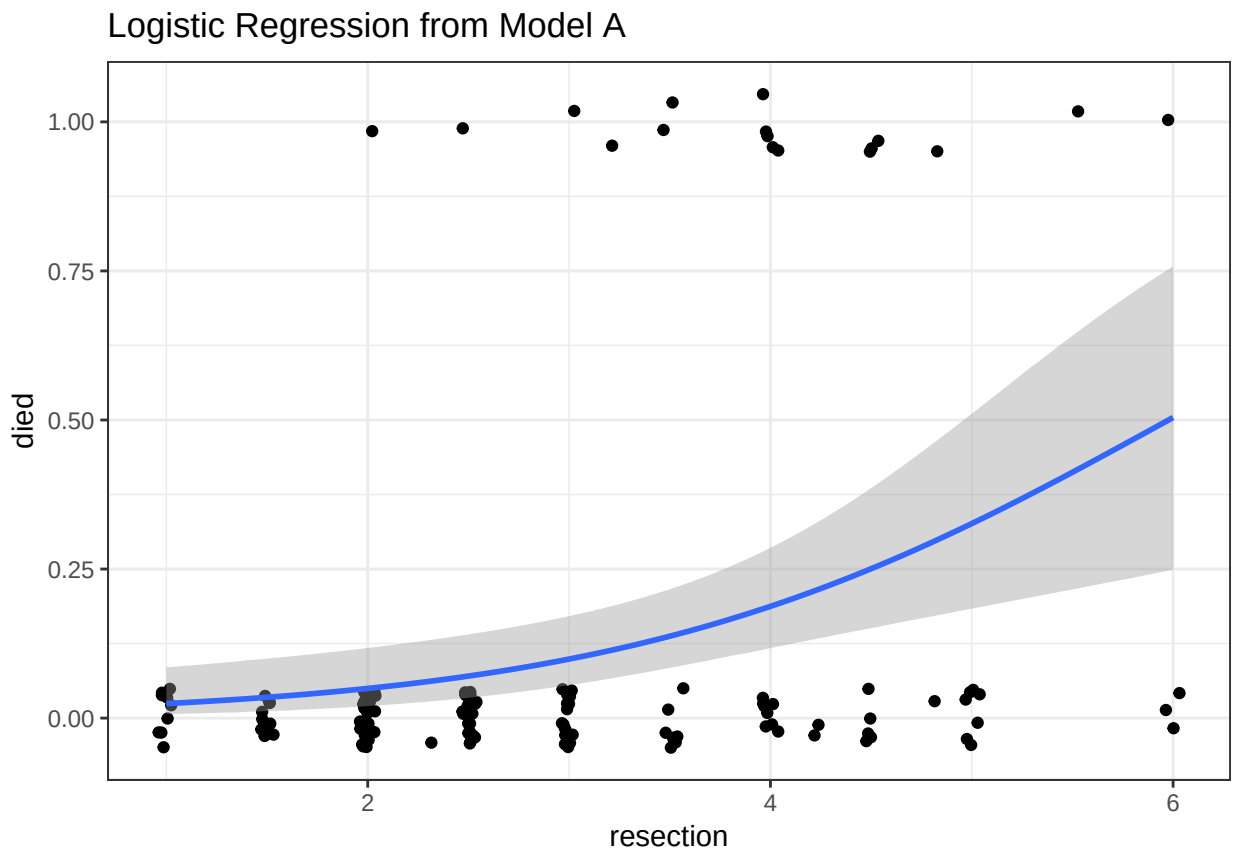
13.5.2 Plotting a Logistic Regression Model's Fitted Values

```
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).

$$TPF(R) = Pr(resection > R | subjectdied)$$

Similarly, the false positive fraction (FPF) for a specific cutoff R is the probability of a positive test (prediction that the patient will die) among the people with died = 0 (those who don't actually die)

$$FPF(R) = Pr(resection > R | subjectdidnotdie)$$

The True Positive Rate is referred to as the sensitivity of a diagnostic test, and the True Negative rate (1 - the False Positive rate) is referred to as the specificity of a diagnostic test.

Since the cutoff R is not fixed in advanced, we can plot the value of TPF (on the y axis) against FPF (on the x axis) for all possible values of R, and this is what the ROC curve is. Others refer to the Sensitivity on the Y axis, and 1-Specificity on the X axis, and this is the same idea.

Before we get too far into the weeds, let me show you some simple situations so you can understand what you might learn from the ROC curve. The web page http://blog.yhat.com/posts/roc-curves.html provides source materials.

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.

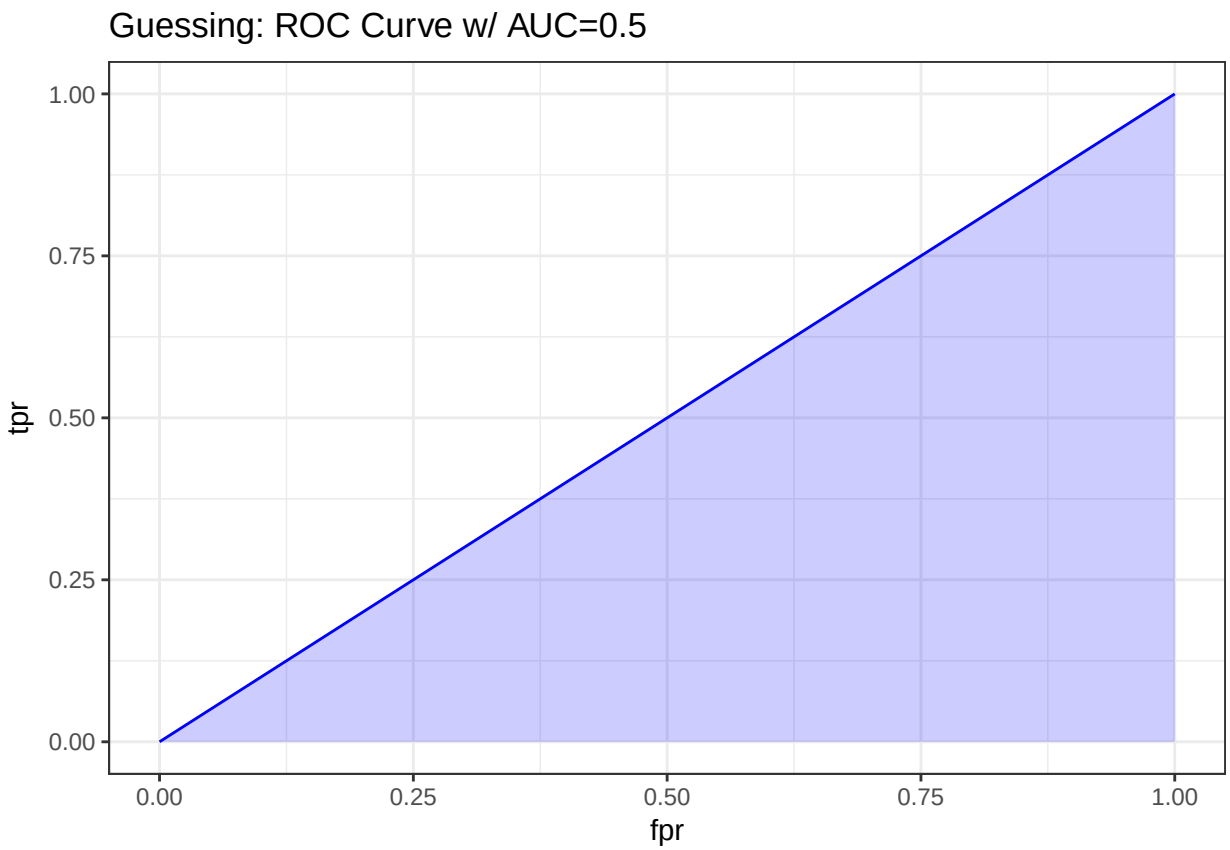
```
set.seed(432999)
sim.temp <- data_frame(x = rnorm(n = 200),
                       prob = exp(x)/(1 + exp(x)),
                       y = as.numeric(1 * runif(200) < prob))

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

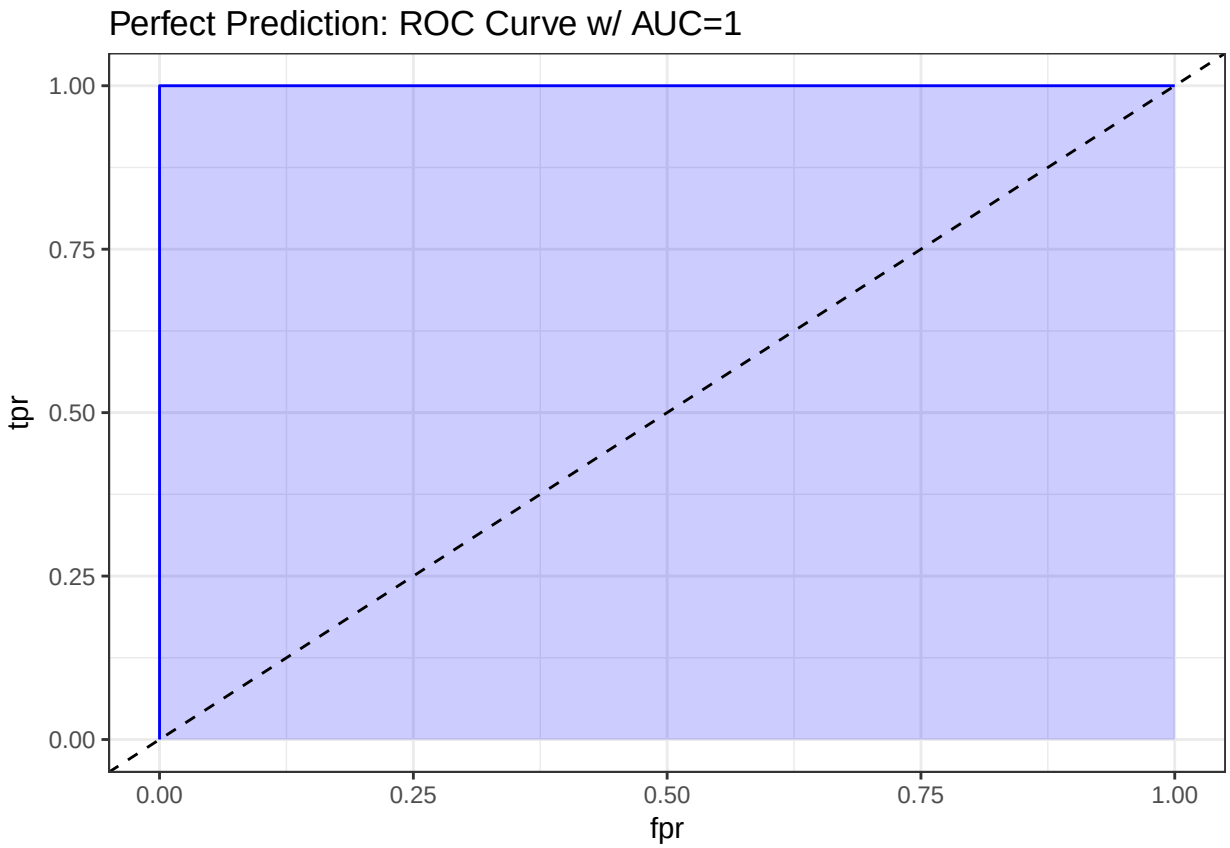
```
pred_guess <- prediction(sim.temp$p_guess, sim.temp$y)
perf_guess <- performance(pred_guess, measure = "tpr", x.measure = "fpr")
auc_guess <- performance(pred_guess, measure="auc")

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.

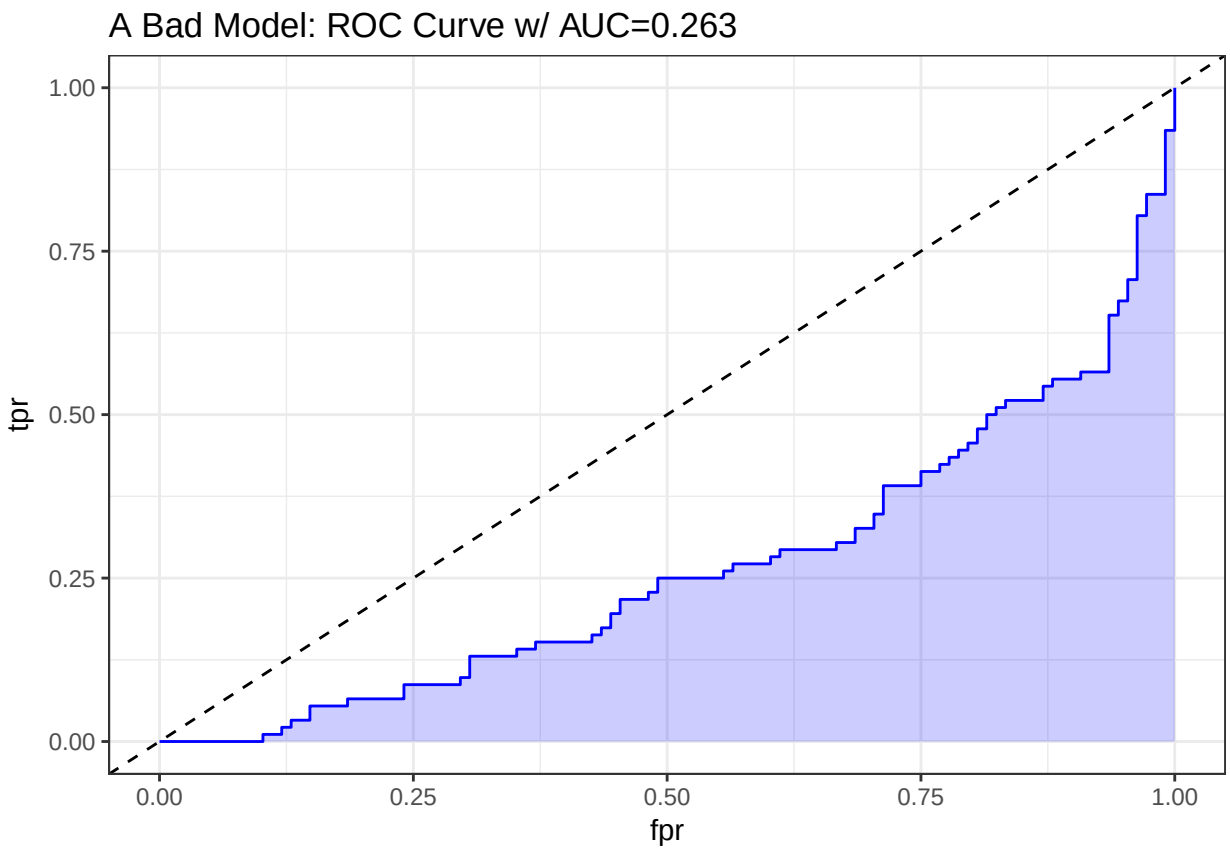
```
pred_perf <- prediction(sim.temp$p_perfect, sim.temp$y)
perf_perf <- performance(pred_perf, measure = "tpr", x.measure = "fpr")
auc_perf <- performance(pred_perf, measure="auc")

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.

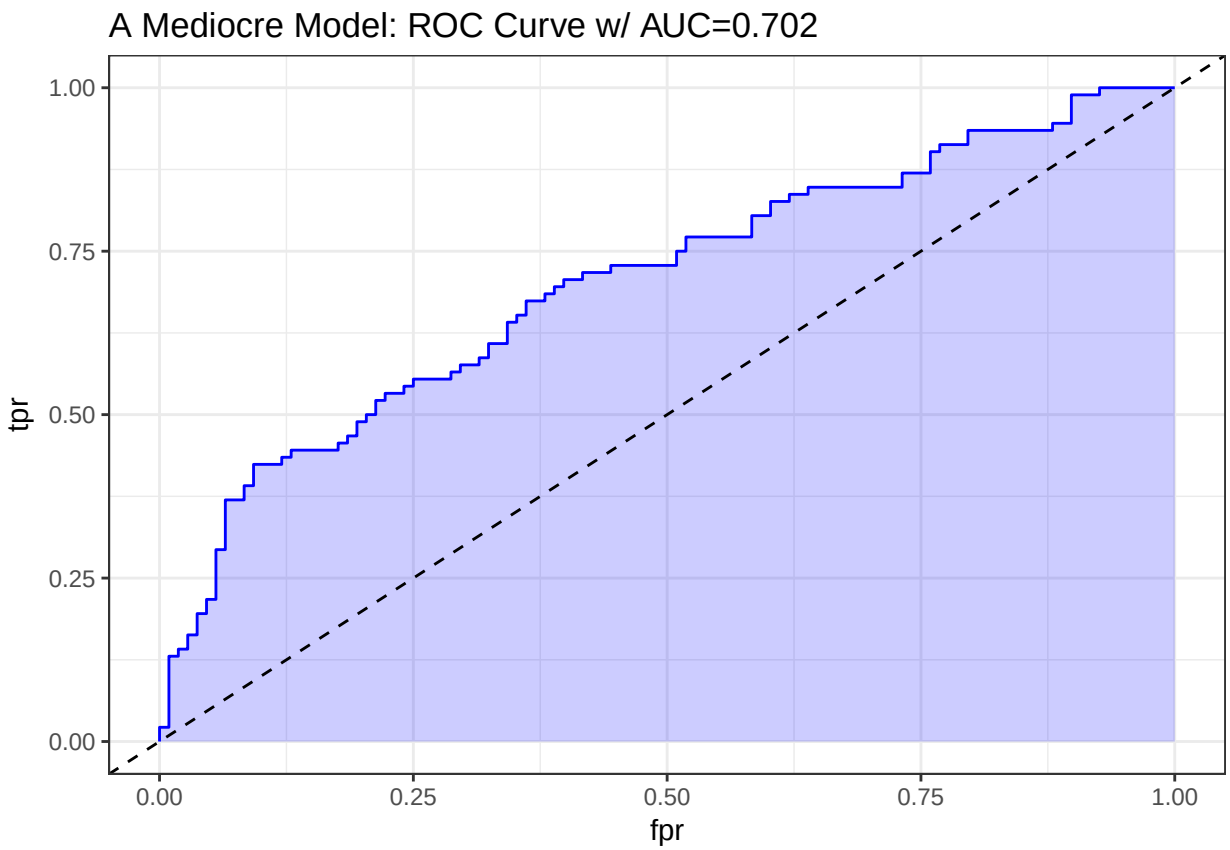
```
pred_bad <- prediction(sim.temp$p_bad, sim.temp$y)
perf_bad <- performance(pred_bad, measure = "tpr", x.measure = "fpr")
auc_bad <- performance(pred_bad, measure="auc")

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.

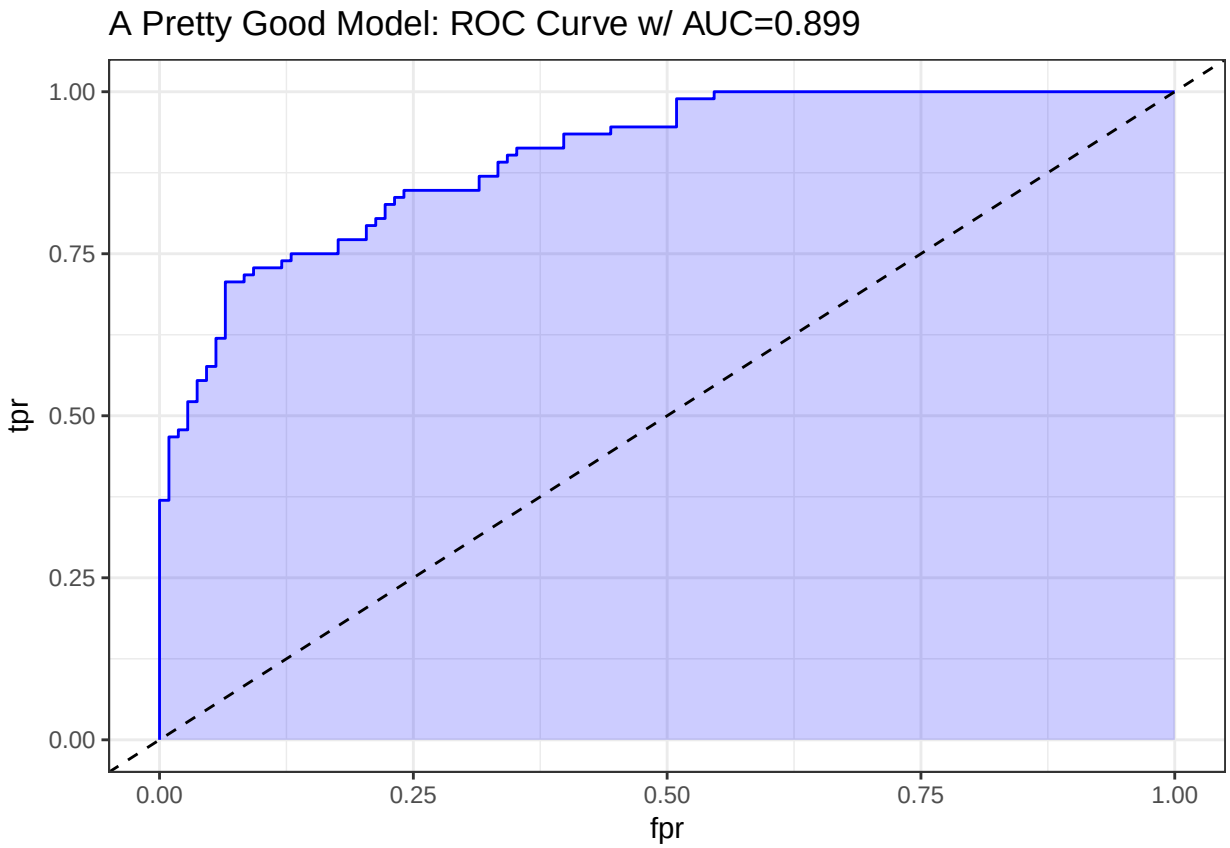
C statistic	Interpretation
0.90 to 1.00	model does an excellent job at discriminating "yes" from "no" (A)
0.80 to 0.90	model does a good job (B)
0.70 to 0.80	model does a fair job (C)
0.60 to 0.70	model does a poor job (D)
0.50 to 0.60	model fails (F)
below 0.50	model is worse than random guessing

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()
```

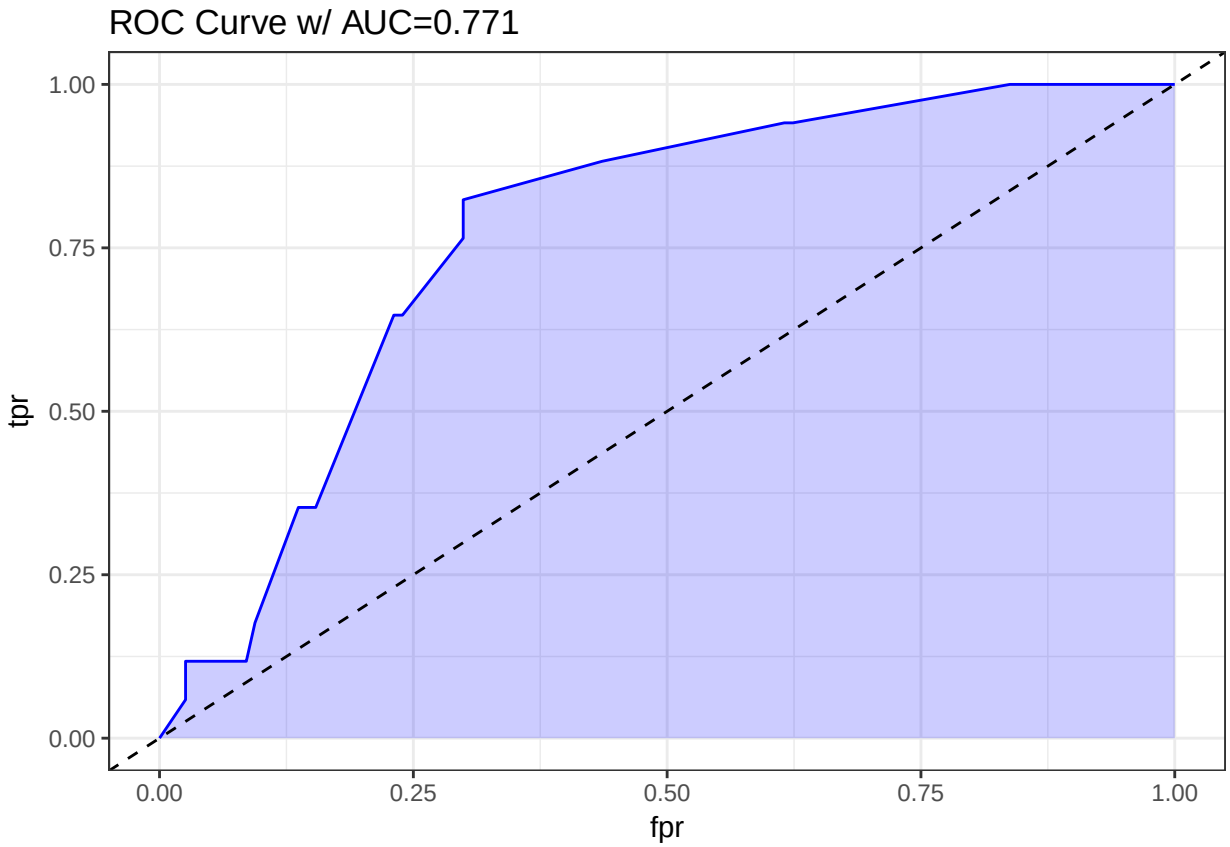


13.8 The ROC Plot for `res_modA`

Let me show you the ROC curve for our `res_modA` model.

```
## requires ROCR package
prob <- predict(res_modA, resect, type="response")
pred <- prediction(prob, resect$died)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
auc <- performance(pred, measure="auc")

auc <- round(auc@y.values[[1]],3)
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values),
                       model="GLM")
```

```
ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
    geom_ribbon(alpha=0.2, fill = "blue") +
    geom_line(aes(y=tpr), col = "blue") +
    geom_abline(intercept = 0, slope = 1, lty = "dashed") +
    labs(title = paste0("ROC Curve w/ AUC=", auc)) +
    theme_bw()
```



Based on the C statistic (AUC = 0.771) this would rank somewhere near the high end of a "fair" predictive model by this standard, not quite to the level of a "good" model.

13.8.1 Another way to plot the ROC Curve

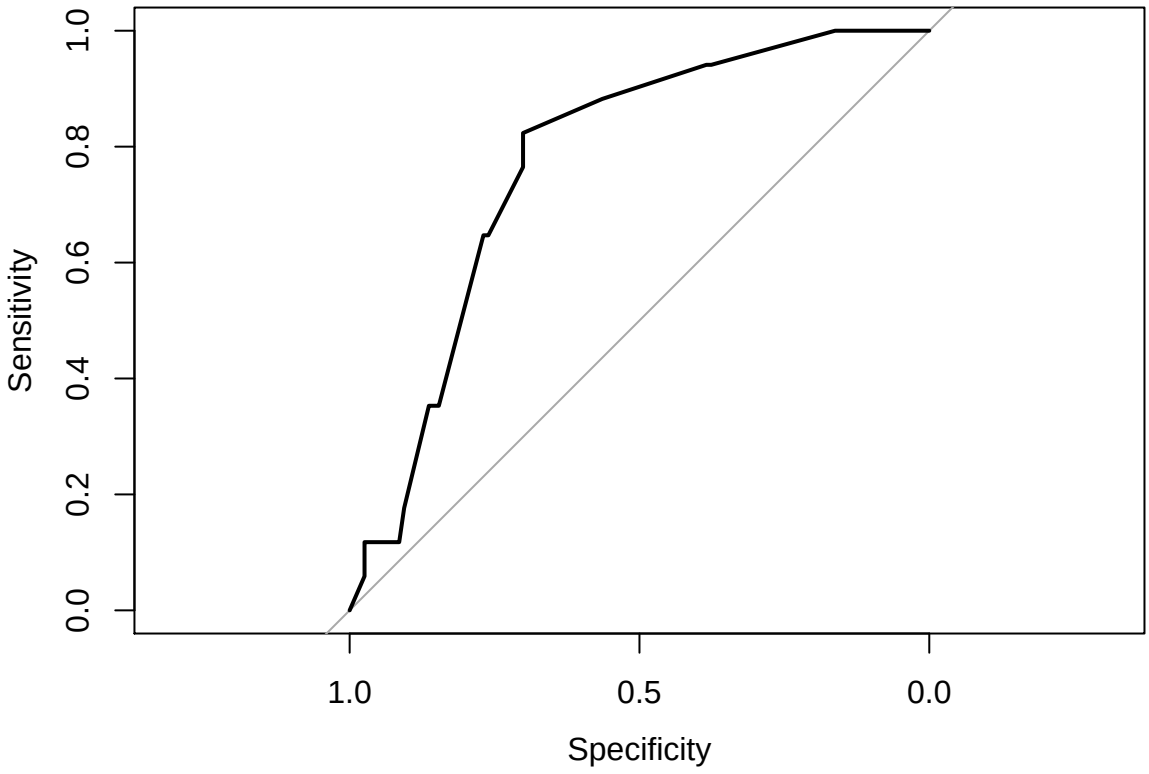
If we've loaded the `pROC` package, we can also use the following (admittedly simpler) approach to plot the ROC curve, without `ggplot2`, and to obtain the C statistic, and a 95% confidence interval around that C statistic.

```
## requires pROC package
roc.modA <-
    roc(resect$died ~ predict(res_modA, type="response"),
        ci = TRUE)

roc.modA
```

```
Call:
roc.formula(formula = resect$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

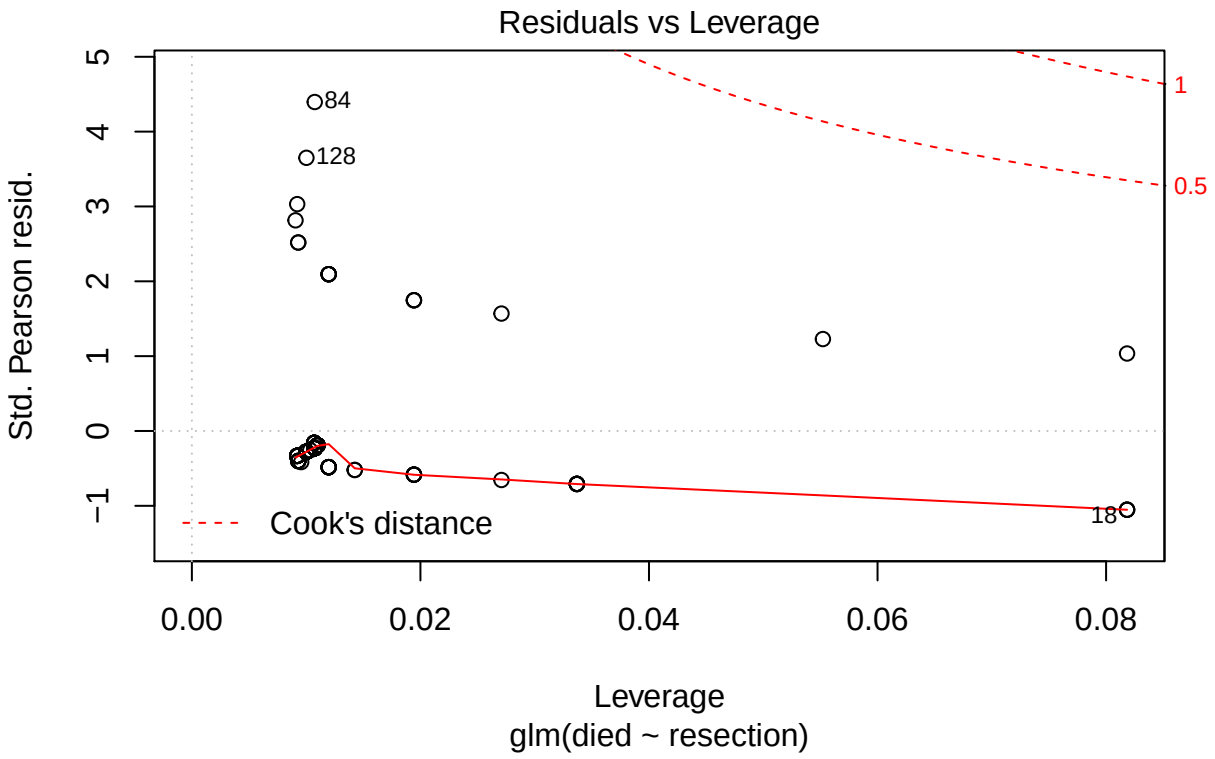
> Residuals are certainly less informative for logistic regression than they are for linear regression: not only do yes/no outcomes inherently contain less information than continuous ones, but the fact that the adjusted response depends on the fit hampers our ability to use residuals as external checks 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. Dev Df Deviance
1       132     89.493
2       129     67.359  3   22.134
```

The addition of `age`, `prior` and `intubated` reduces the lack of fit by 22.134 points, at a cost of 3 degrees of freedom.

```
glance(res_modA)
```

```
  null.deviance df.null    logLik      AIC     BIC deviance df.residual
1      101.9431     133 -44.74646 93.49292 99.2886 89.49292         132
```

```
glance(res_modB)
```

```
  null.deviance df.null   logLik     AIC     BIC deviance df.residual
1      101.9431     133 -33.6793 77.3586 91.8478  67.3586         129
```

By either AIC or BIC, the larger model (`res_modB`) looks more effective.

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 `resection`s are associated with a meaningful rise (est OR: 1.84, 95% CI 1.08, 3.35) in the odds of death, holding all other predictors constant,
- the need for `intubation` at the end of surgery is associated with a substantial rise (est OR: 16.6, 95% CI 4.7, 64.7) in the odds of death, holding all other predictors constant, but that
- older `age` as well as having a `prior` tracheal surgery appears to be associated with an increase in death risk, but not to an extent that we can declare statistically significant.

13.11 Plotting Model B

Let's think about plotting the fitted values from our model, in terms of probabilities.

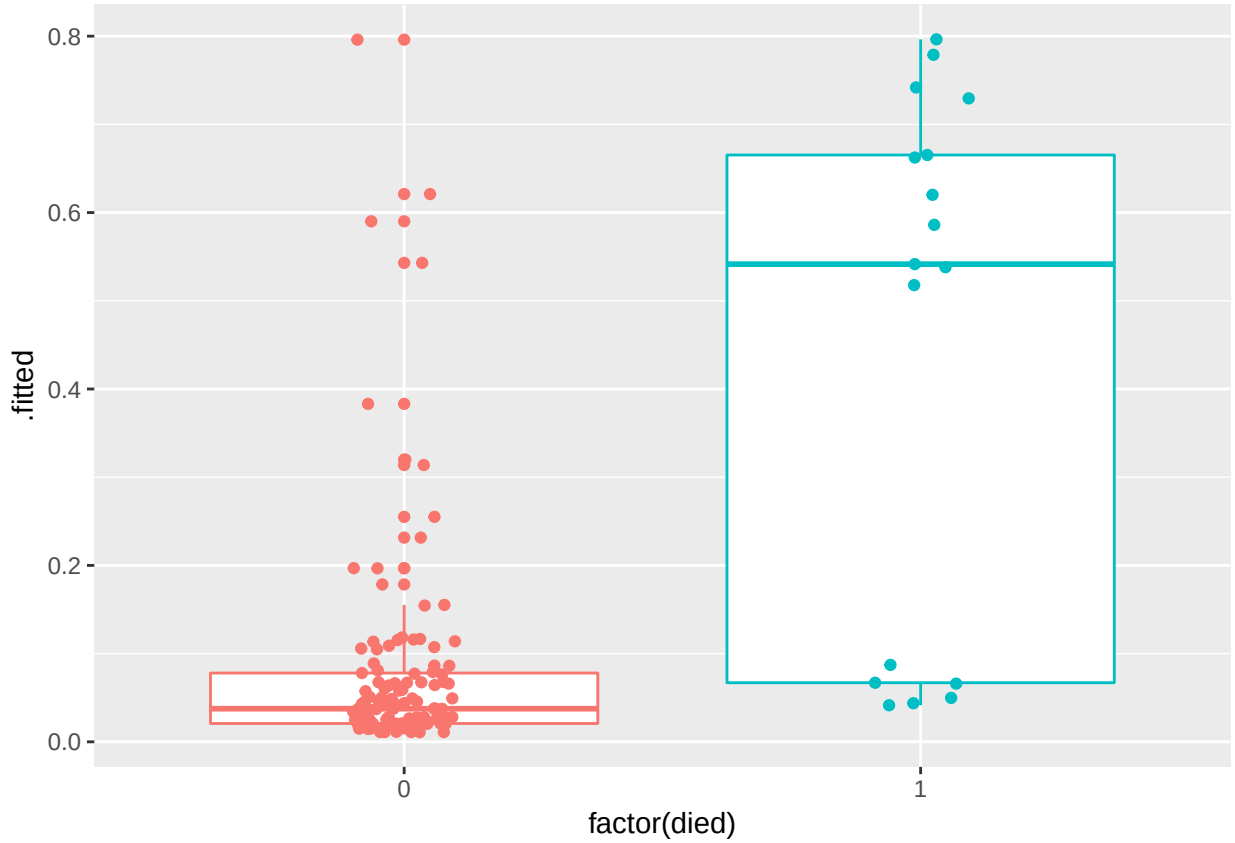
13.11.1 Using `augment` to capture the fitted probabilities

```
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      .cooksd .std.resid
1 0.02667562 0.7247491 0.0003536652 -0.3537702
2 0.03796756 0.7240341 0.0010829917 -0.5076925
3 0.11416656 0.7215778 0.0107925872  0.8439524
4 0.07039819 0.7234665 0.0029937671 -0.6024273
5 0.13126049 0.7225958 0.0088920280  0.7239256
6 0.01045207 0.7250114 0.0000823406 -0.2765683
```

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)
```

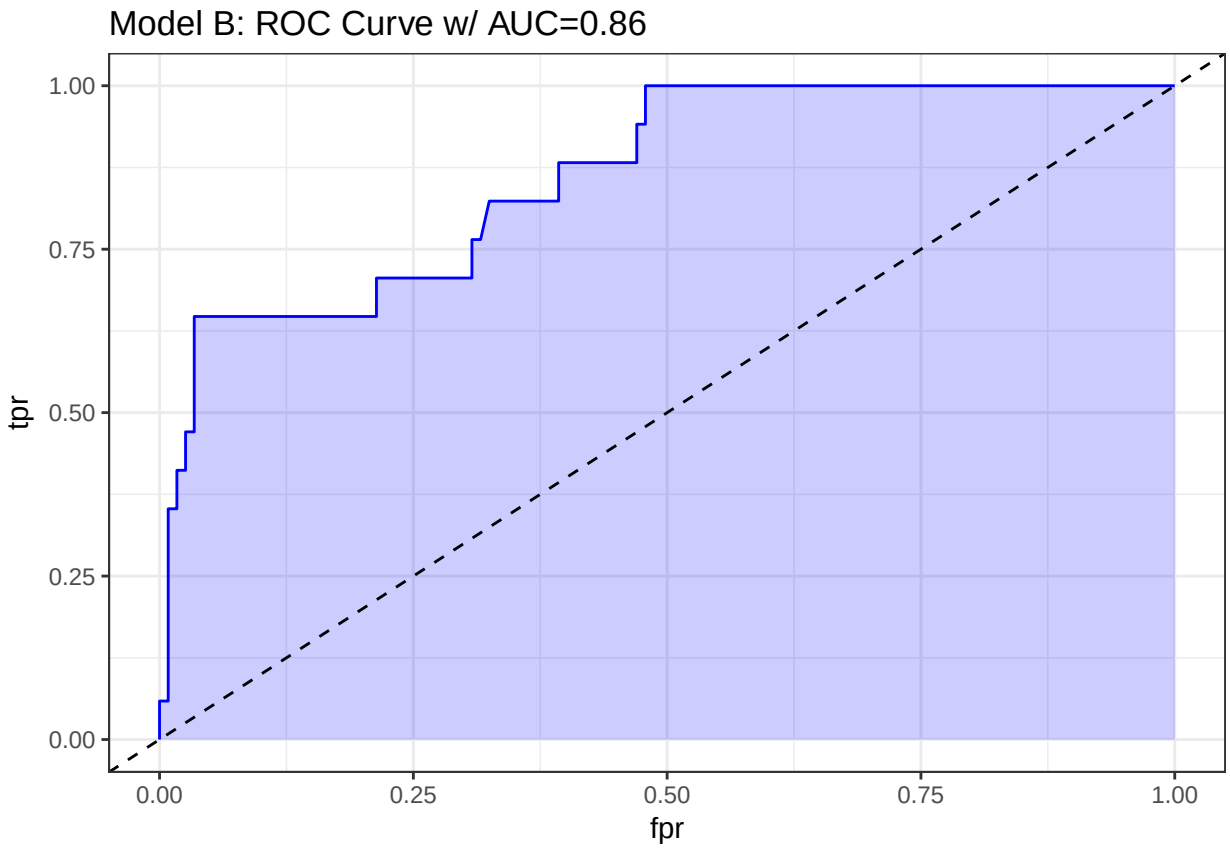


Certainly it appears as though most of our predicted probabilities (of death) for the subjects who actually survived are quite small, but not all of them. We also have at least 6 big "misses" among the 17 subjects who actually died.

13.11.3 The ROC curve for Model B

```
## requires ROCR package
prob <- predict(res_modB, resect, type="response")
pred <- prediction(prob, resect$died)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
auc <- performance(pred, measure="auc")

auc <- round(auc@y.values[[1]],3)
roc.data <- data.frame(fpr=unlist(perf@x.values),
                       tpr=unlist(perf@y.values),
                       model="GLM")

ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +
```

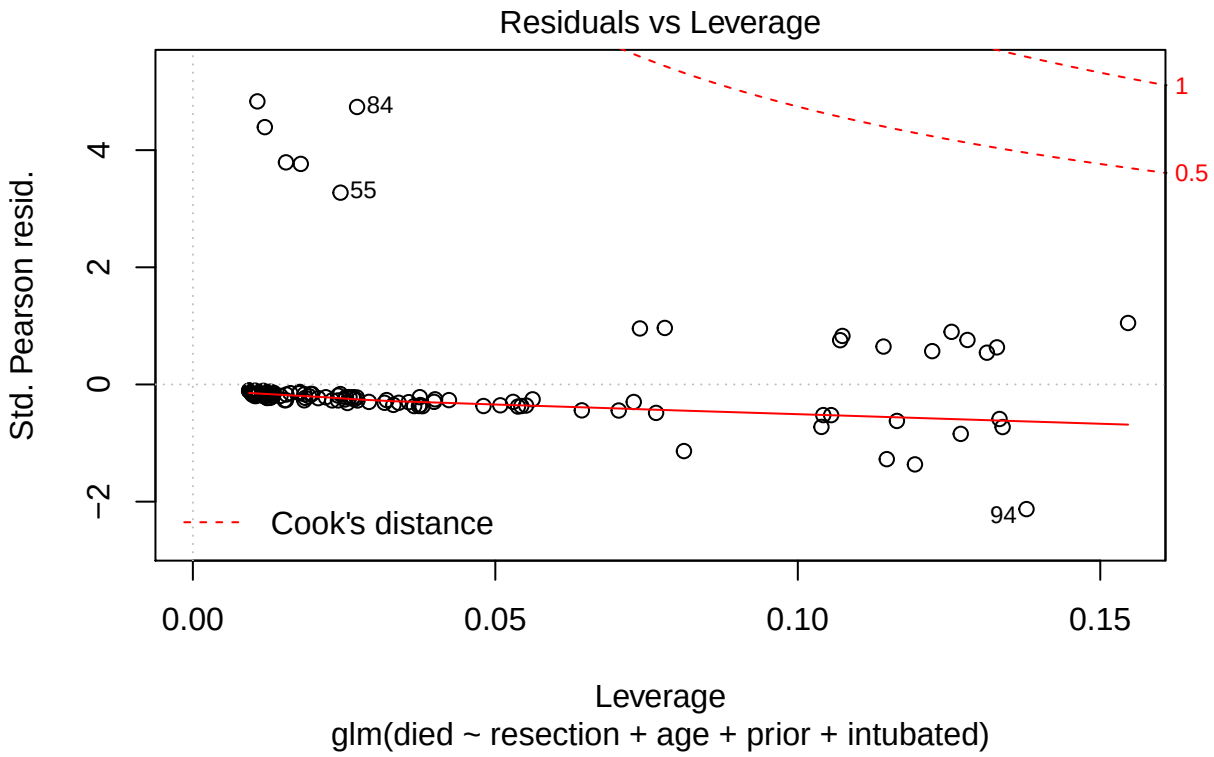
```
    geom_ribbon(alpha=0.2, fill = "blue") +
    geom_line(aes(y=tpr), col = "blue") +
    geom_abline(intercept = 0, slope = 1, lty = "dashed") +
    labs(title = paste0("Model B: ROC Curve w/ AUC=", auc)) +
    theme_bw()
```



The area under the curve (C-statistic) is 0.86, which certainly looks like a more discriminating fit than model A with resection alone.

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)

                       Model Likelihood     Discrimination    Rank Discrim.
                          Ratio Test           Indexes           Indexes
 Obs           134    LR chi2     12.45    R2       0.167    C       0.771
  0            117    d.f.            1    g        1.037    Dxy     0.541
  1             17    Pr(> chi2) 0.0004    gr       2.820    gamma   0.582
 max |deriv| 2e-06                         gp       0.110    tau-a   0.121
                                           Brier    0.103
```

```
           Coef    S.E.   Wald Z Pr(>|Z|)
Intercept -4.4337 0.8799 -5.04  <0.0001
resection  0.7417 0.2230  3.33  0.0009
```

This output specifies the following:

- `Obs` = The number of observations used to fit the model, with `0` = the number of zeros and `1` = the number of ones in our outcome, `died`. Also specified is the maximum absolute value of the derivative at the point where the maximum likelihood function was estimated. I wouldn't worry about that practically, as all you will care about is whether the iterative function-fitting process converged, and R will warn you in other ways if it doesn't.
- A likelihood ratio test (drop in deviance test) subtracting the residual deviance from the null deviance obtain the Likelihood Ratio χ^2 statistic, subtracting residual df from null df to obtain degrees of freedom, and comparing the resulting test statistic to a χ^2 distribution with the appropriate degrees of freedom to determine a p value.
- A series of discrimination indexes, including the Nagelkerke R^2, symbolized R2, and several others we'll discuss shortly.
- A series of rank discrimination indexes, including the C statistic (area under the ROC curve) and Somers' D (Dxy), and several others.
- A table of coefficients, standard errors, Wald Z statistics and p values based on those Wald statistics.

The C statistic is estimated to be 0.771, with an associated (Nagelkerke) $R^2 = 0.167$, both indicating at best mediocre performance for this model, as it turns out.

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.

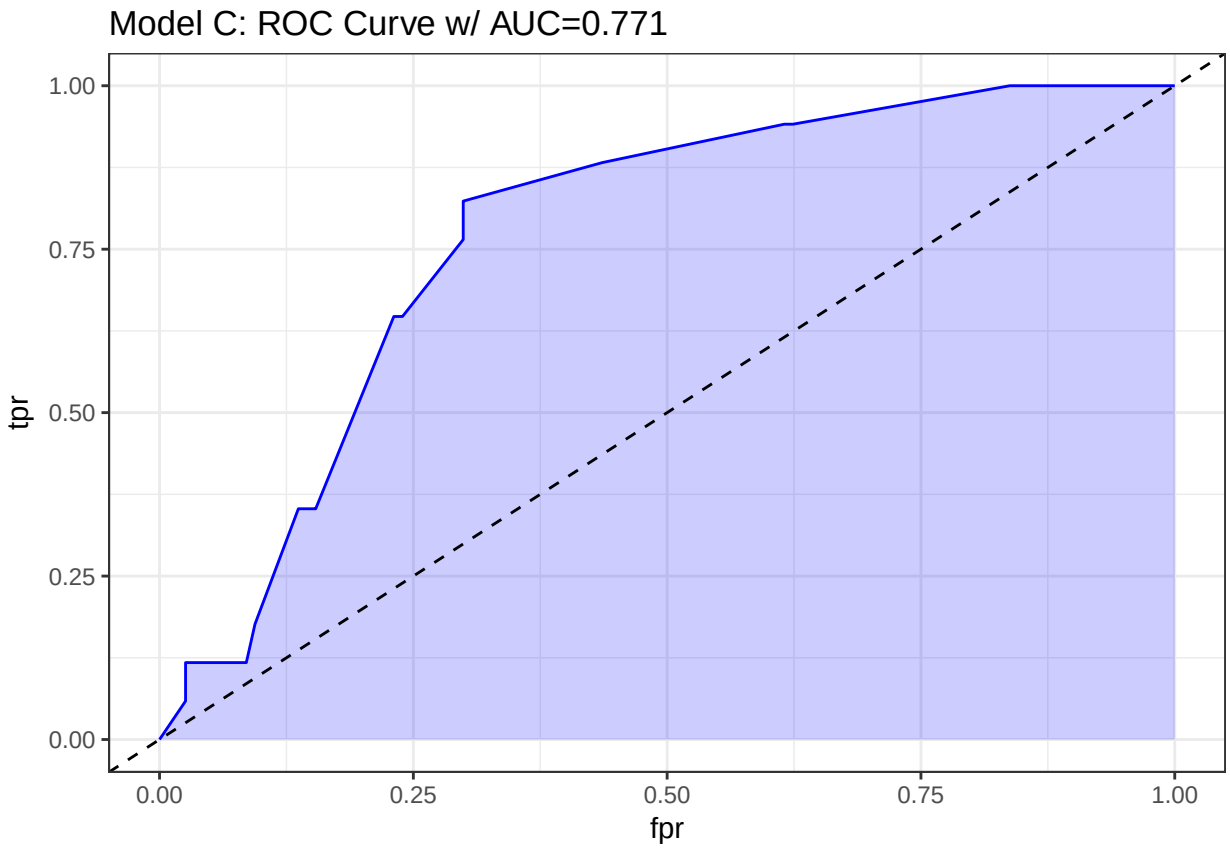
- Recall that C ranges from 0 to 1. 0 = BAD, 1 = GOOD.
  - values of C less than 0.5 indicate that your prediction model is not even as good as simple random guessing of "yes" or "no" for your response.
  - C = 0.5 for random guessing

  - 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 R^2 may be more sensitive.
- But as it turns out, we sometimes have to look at the ROC shapes, as the summary statistic alone isn't enough.

In our case, Somers D (Dxy) = .541, so the C statistic is 0.771.

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
```

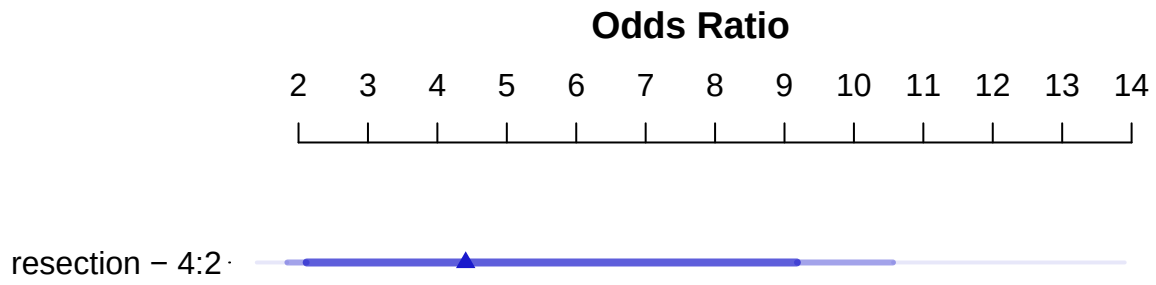
```
Dxy          0.5415   0.5325  0.5415  -0.0090          0.5505 100
R2           0.1666   0.1664  0.1666  -0.0002          0.1668 100
Intercept    0.0000   0.0000  0.1425  -0.1425          0.1425 100
Slope        1.0000   1.0000  1.0742  -0.0742          1.0742 100
Emax         0.0000   0.0000  0.0416   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 R^2 = 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)
```

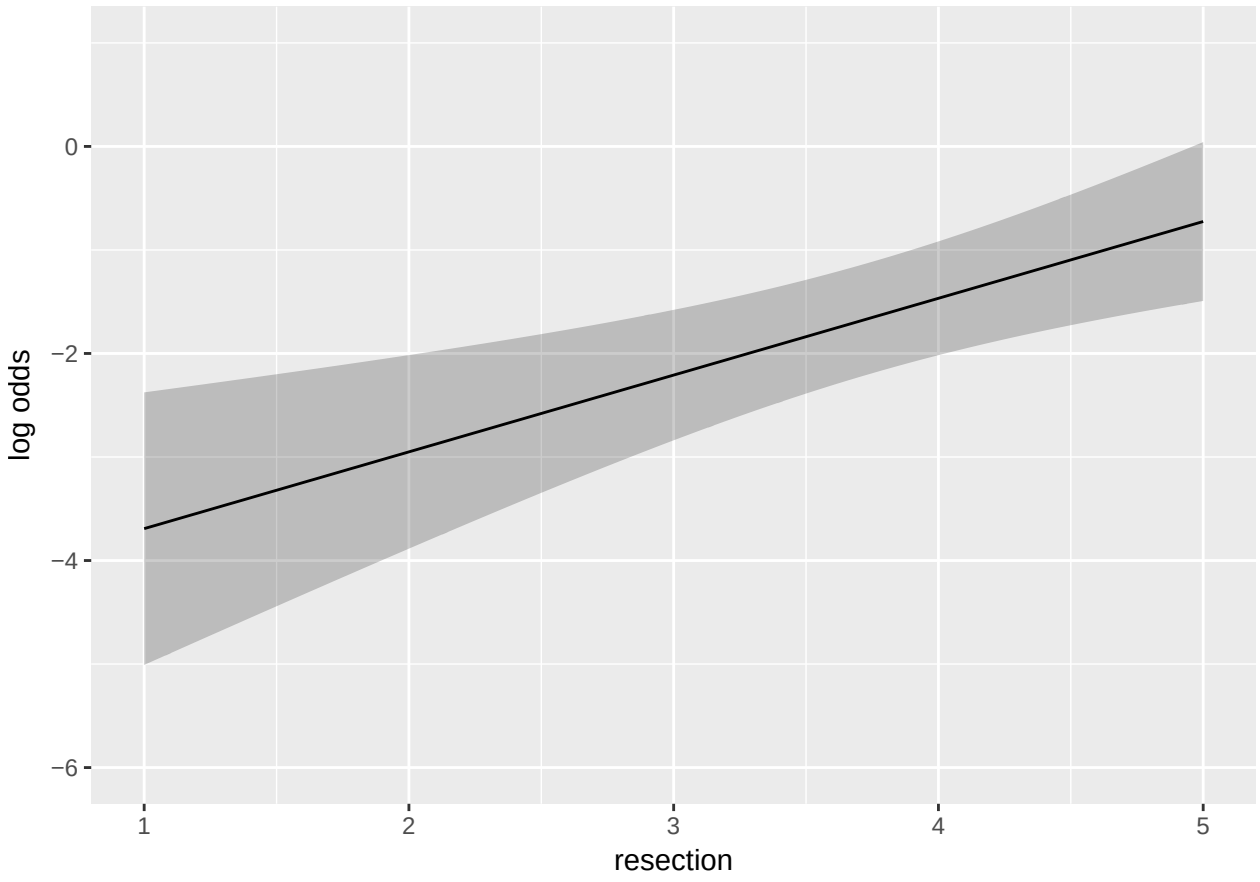
```
             Effects              Response : died

 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

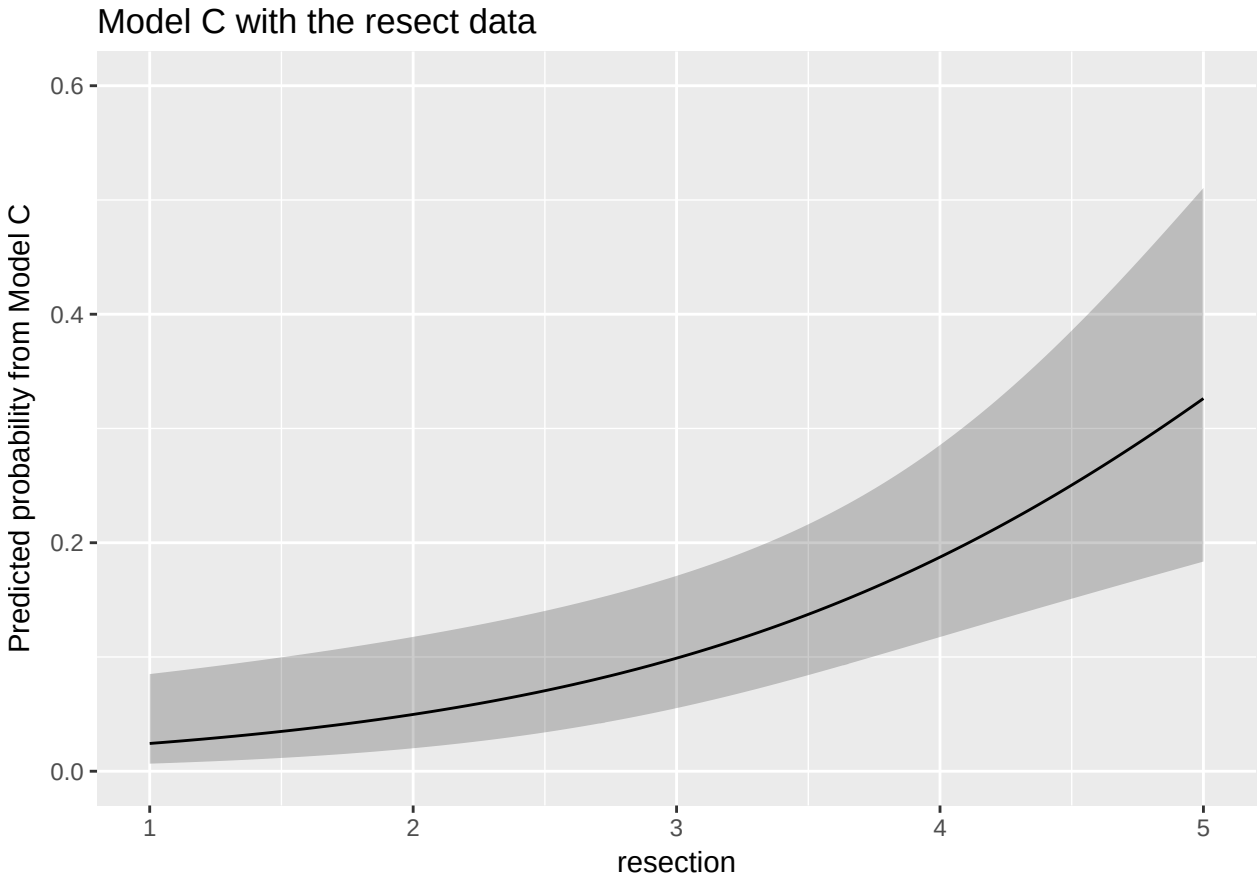
Here we plot the effect of `resection` (and 95% confidence intervals) across the range of observed values of `resection` on the log odds of death. Note the linear effect of `resection` size on the log odds scale.

```
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")
```



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 x 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))
```

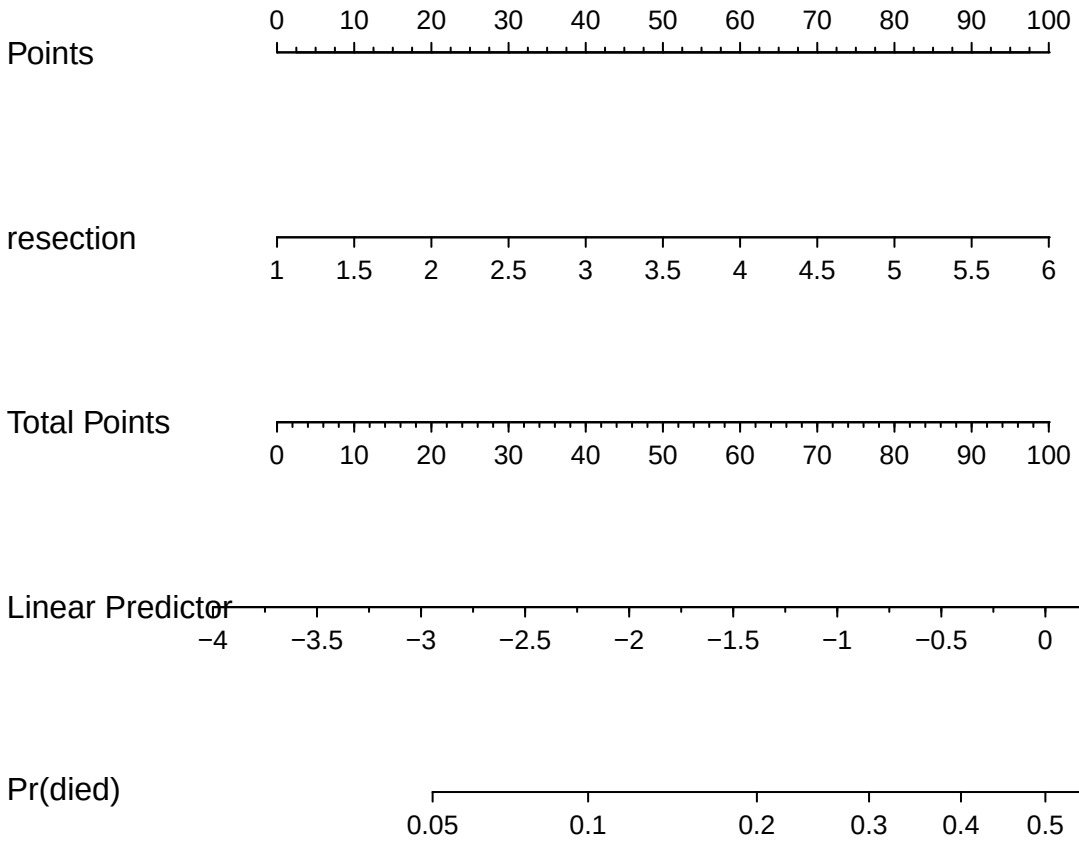
```
    Count resection
84      2      *2.0
128     1       2.5
```

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.

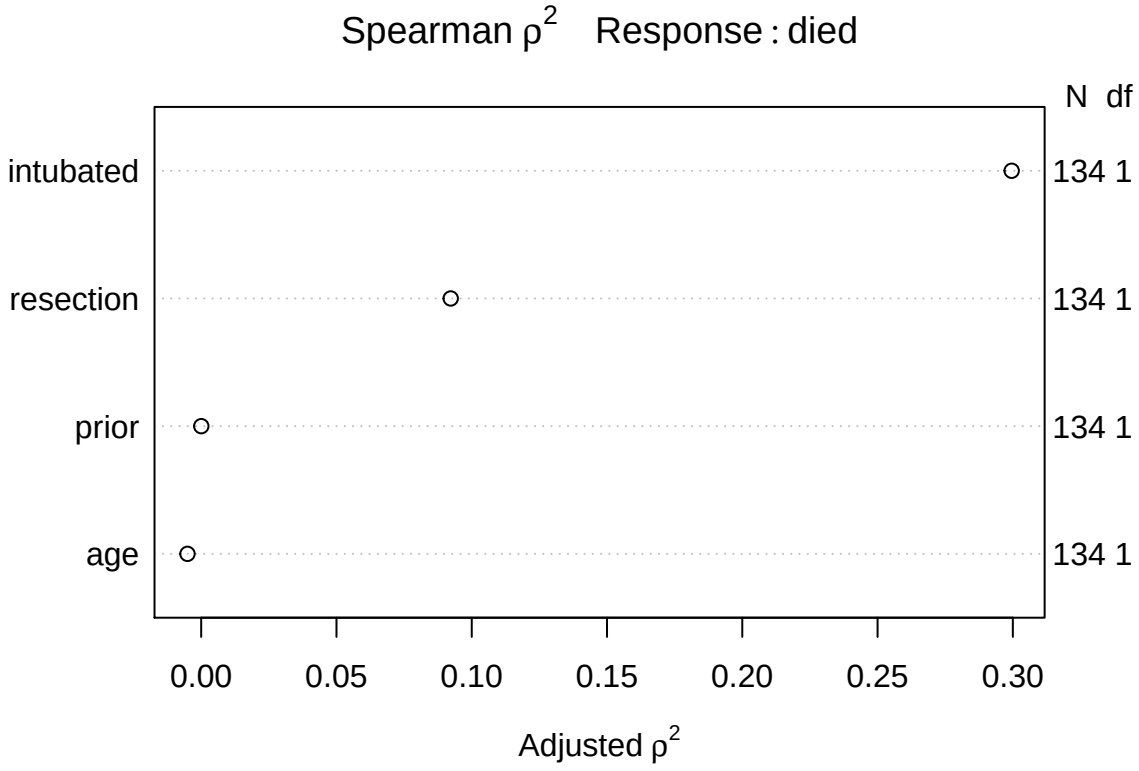
13.13 Model D: An Augmented Kitchen Sink Model

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)

                      Model Likelihood     Discrimination    Rank Discrim.
                         Ratio Test           Indexes           Indexes
Obs           134    LR chi2      38.08    R2       0.464    C       0.880
 0            117    d.f.             6    g        2.382    Dxy     0.759
 1             17    Pr(> chi2) <0.0001    gr      10.825    gamma   0.770
max |deriv| 9e-08                          gp       0.172    tau-a   0.169
                                           Brier    0.067

                     Coef     S.E.   Wald Z Pr(>|Z|)
Intercept            -11.3636 4.9099 -2.31  0.0206
age                    0.0000 0.0210  0.00  0.9988
prior                  0.6269 0.7367  0.85  0.3947
resection              3.3799 1.9700  1.72  0.0862
resection'            -4.2104 2.7035 -1.56  0.1194
intubated              0.4576 2.7848  0.16  0.8695
intubated * resection  0.6188 0.7306  0.85  0.3970
```

- The model likelihood ratio test suggests that at least some of these predictors are helpful.
- The Nagelkerke R^2 of 0.46, and the C statistic of 0.88 indicate a meaningful improvement in discrimination over our model with `resection` alone.
- The Wald Z tests see some potential need to prune the model, as none of the elements reaches statistical significance without the others. The product term between `intubated` and `resection`, in particular, doesn't appear to have helped much, once we already had the main effects.

13.13.4 ANOVA and Wald Tests for Model D

```
anova(res_modD)
```

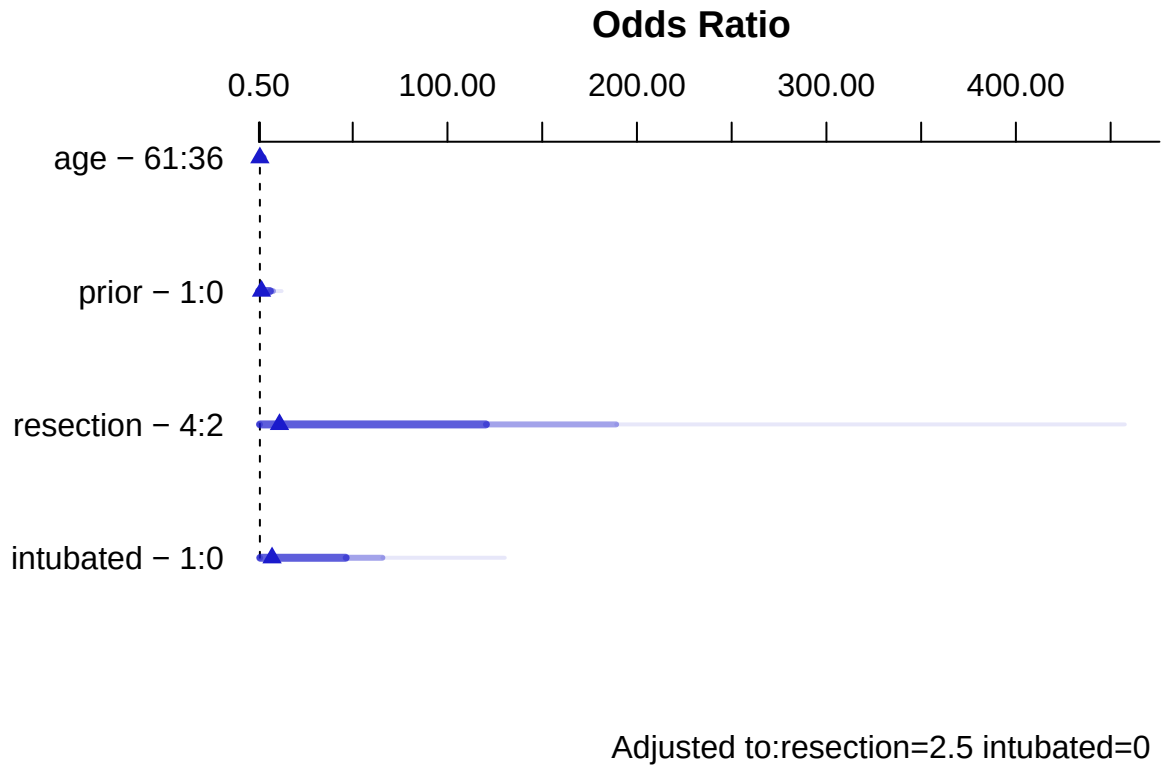
```
               Wald Statistics          Response: died

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)
```

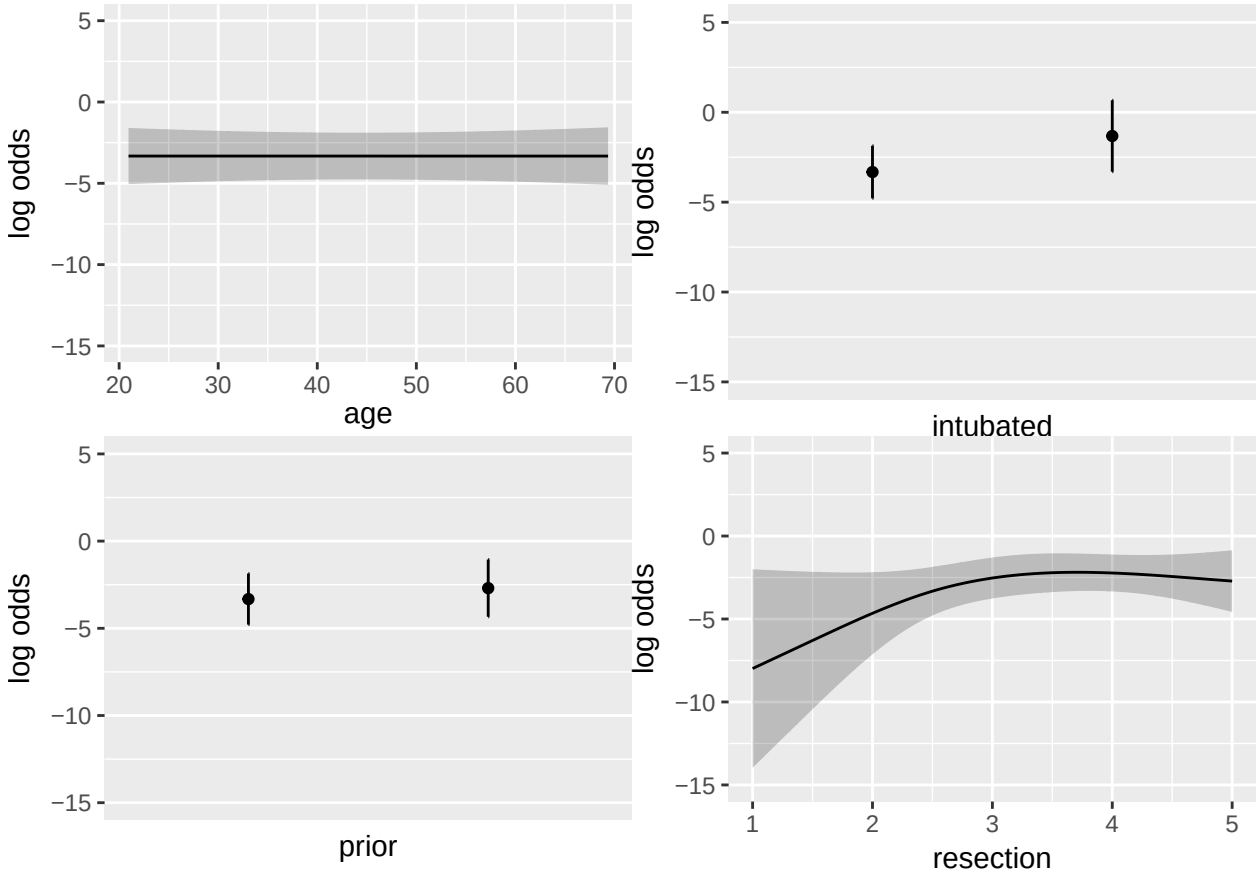
```
             Effects              Response : died

 Factor      Low High Diff. Effect      S.E.    Lower 0.95 Upper 0.95
 age         36  61   25    -0.00080933 0.52409 -1.02800     1.0264
  Odds Ratio 36  61   25     0.99919000      NA  0.35772     2.7910
 prior        0   1    1     0.62693000 0.73665 -0.81688     2.0707
  Odds Ratio  0   1    1     1.87190000      NA  0.44181     7.9307
 resection    2   4    2     2.42930000 1.43510 -0.38342     5.2419
  Odds Ratio  2   4    2    11.35000000      NA  0.68153   189.0400
 intubated    0   1    1     2.00470000 1.11220 -0.17513     4.1845
  Odds Ratio  0   1    1     7.42380000      NA  0.83934    65.6610

Adjusted to: resection=2.5 intubated=0
```

The effect sizes are perhaps best described in terms of odds ratios. The odds ratio for death isn't significantly different from 1 for any variable, but the impact of `resection` and `intubated`, though not strong enough to be significant, look more substantial (if poorly estimated) than the effects of `age` and `prior`.

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.

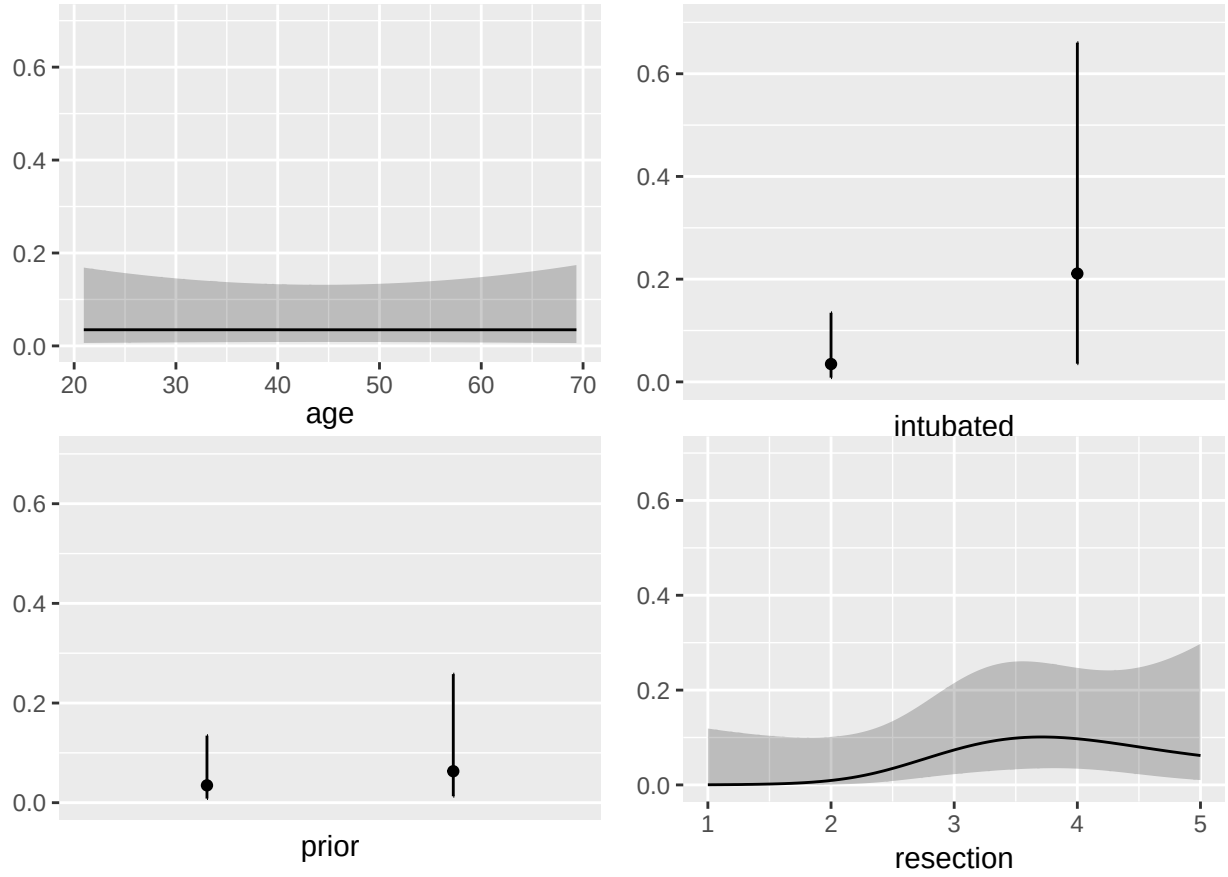
```
ggplot(Predict(res_modD))
```



We can also capture and plot these results on the probability scale, as follows[1].

```
ggplot(Predict(res_modD, fun = plogis))
```

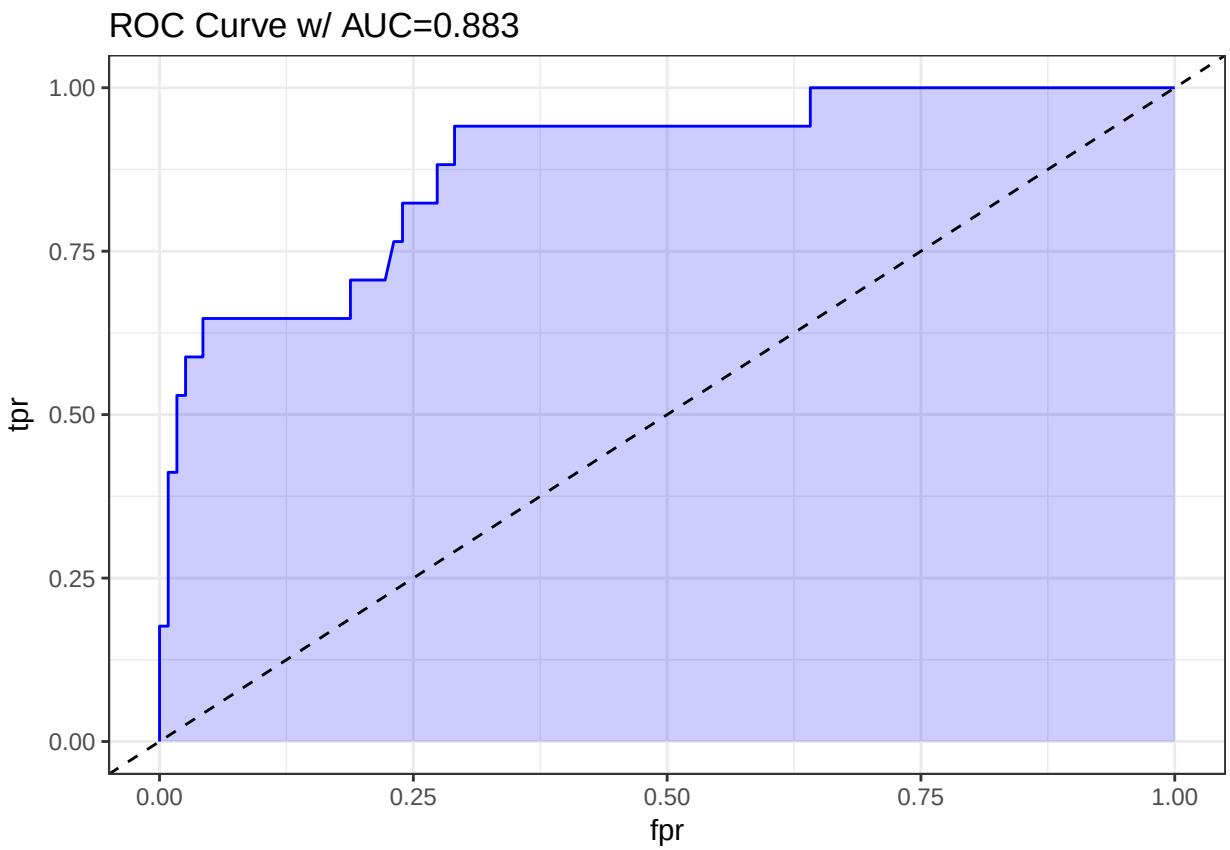
[1] Although I've yet to figure out how to get the y axis relabeled properly without simply dumping the Predict results into a new tibble and starting over with creating the plots.



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()
```

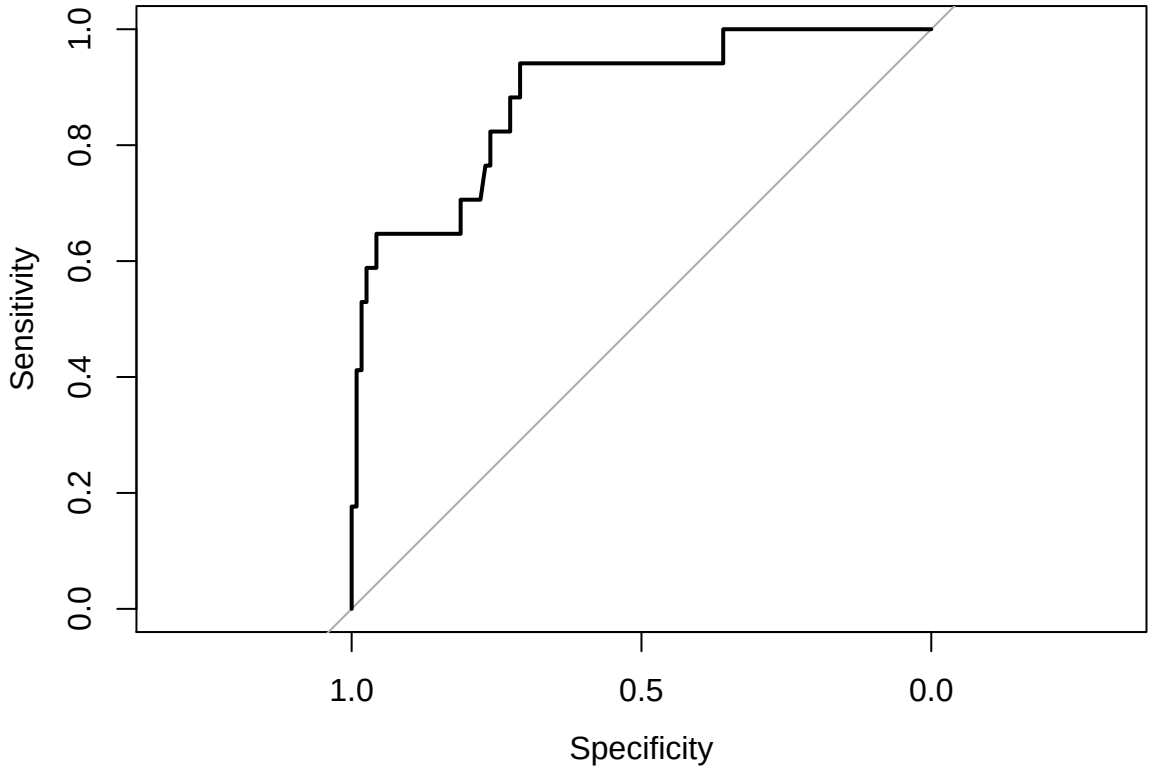


The AUC fitted with ROCR (0.883) is slightly different than what lrm has told us (0.880), and this also happens if we use the pROC approach, demonstrated below.

```
## 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 `intubated` + 0.548 `resection`.

This implies that:

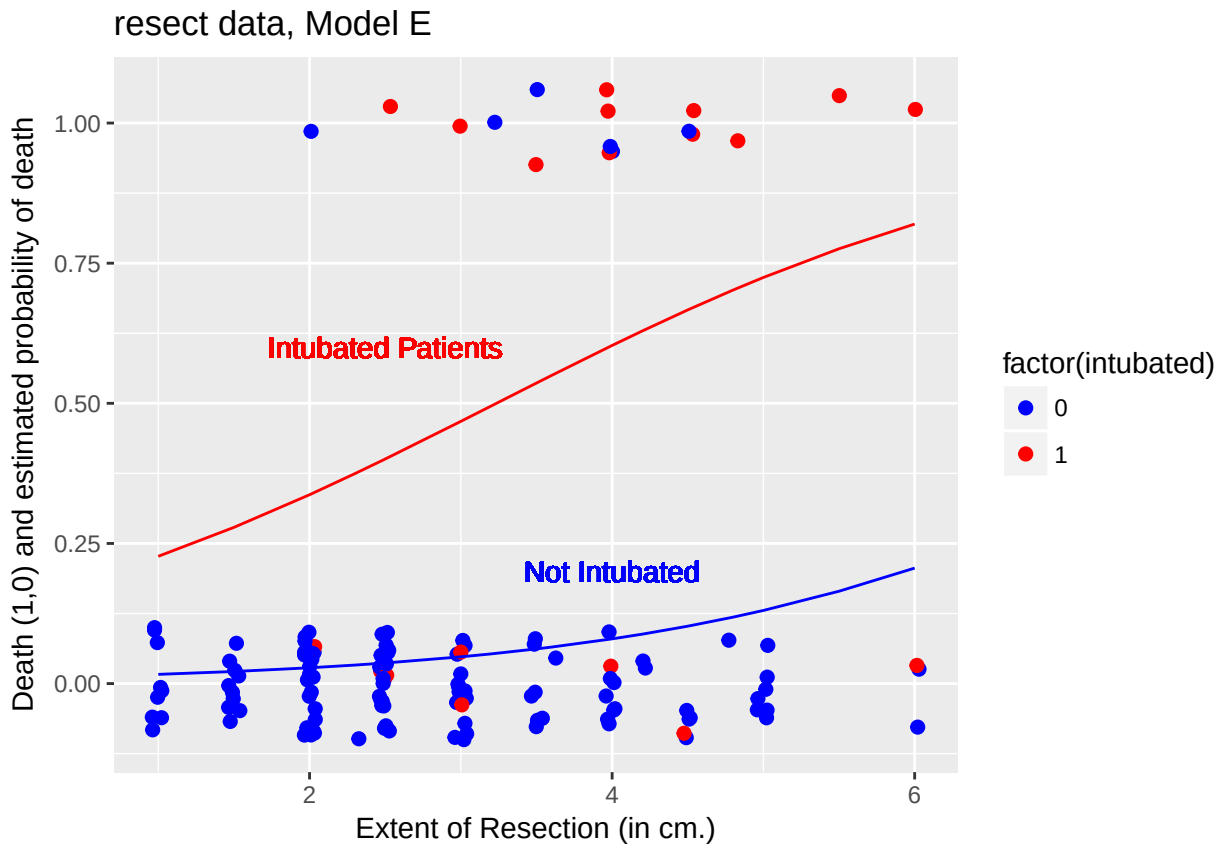
- for intubated patients, the equation is -1.773 + 0.548 `resection`, while
- for non-intubated patients, the equation is -4.637 + 0.548 `resection`.

We can use the `ilogit` function within the `faraway` package to help plot this.

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")
```

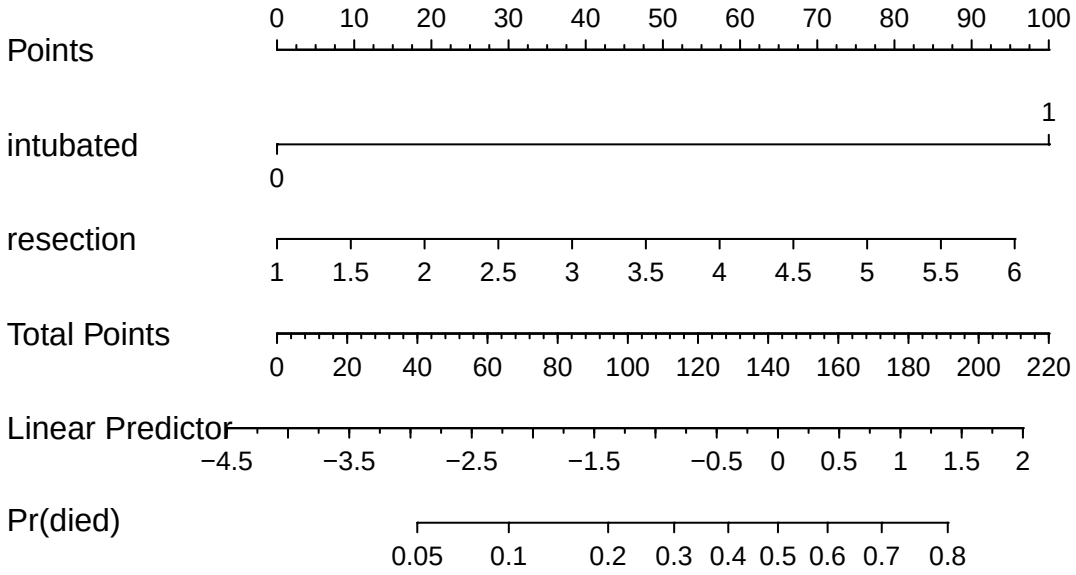


The effect of `intubation` appears to be very large, compared to the resection size effect.

13.14.2 Nomogram for Model E

A nomogram of the model would help, too.

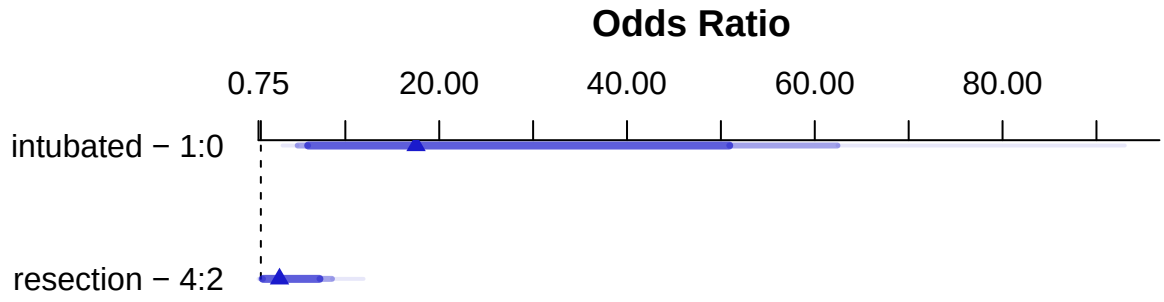
```
plot(nomogram(res_modE, fun=plogis,
              fun.at=c(0.05, seq(0.1, 0.9, by=0.1), 0.95),
              funlabel="Pr(died)"))
```



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))
```



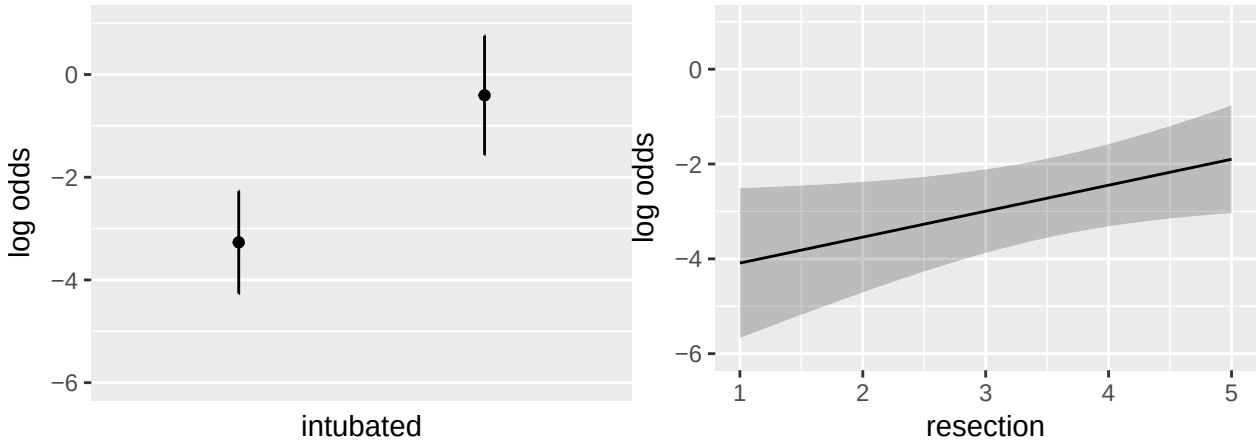
```
summary(res_modE)
```

```
             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

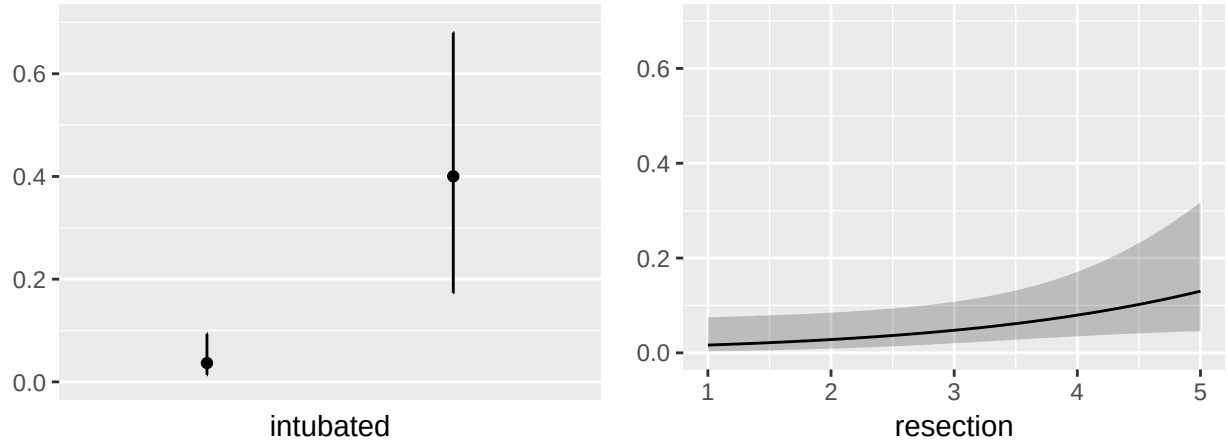
Here are plots of the effects across the range of each predictor (holding the other constant) on the log odds scale.

```
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
```

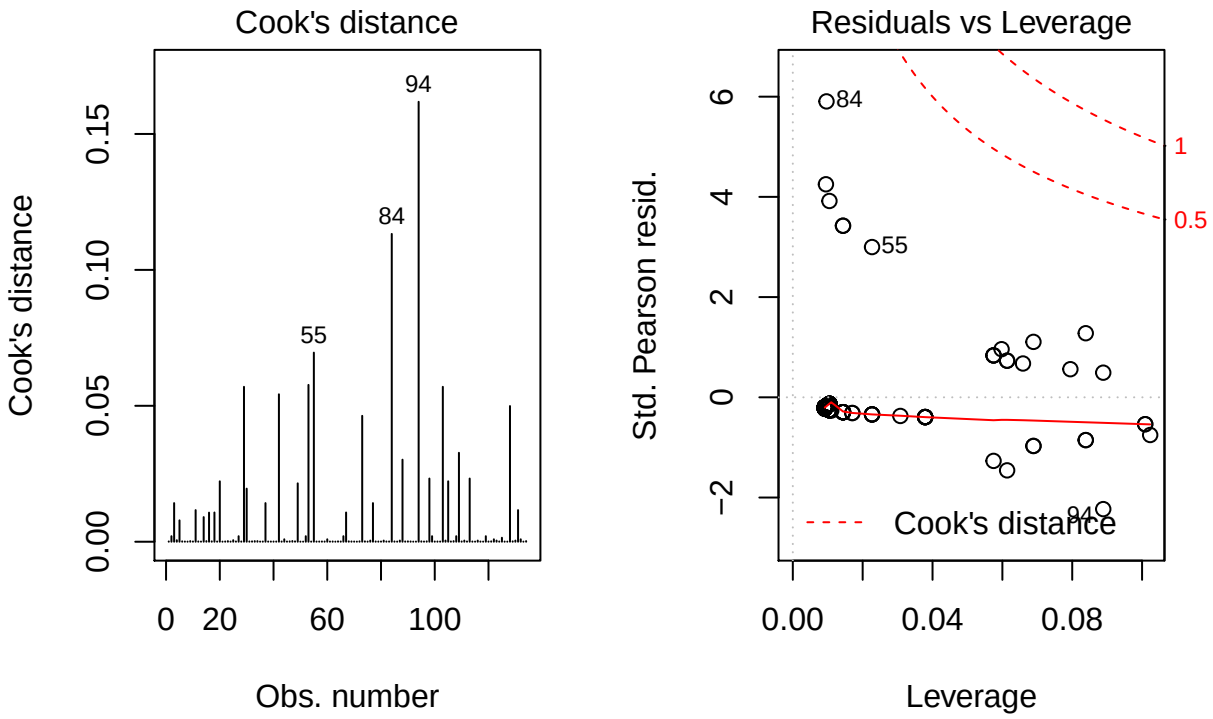
```
$Intercept
[1] "84" "94"

$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
```

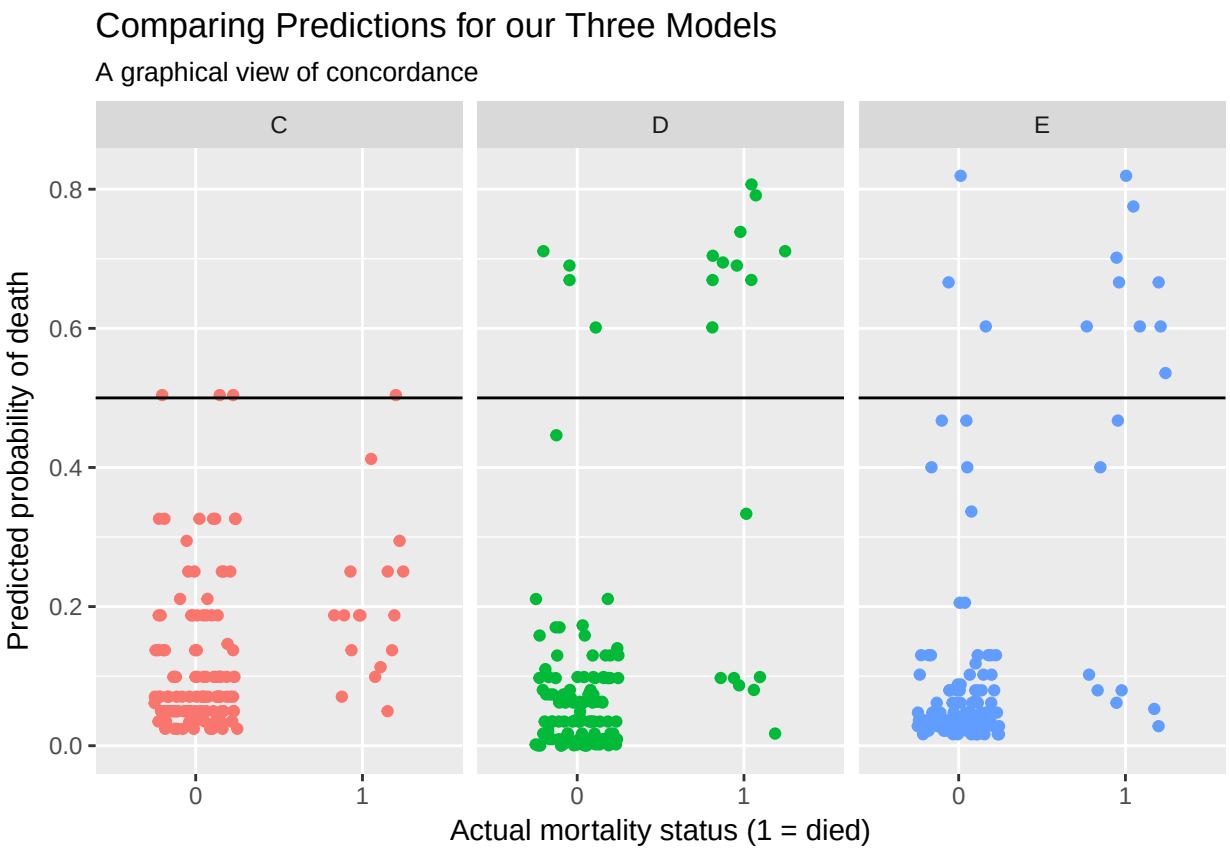
And now, we'll use the `gather` command to arrange the models and predicted probabilities in a more useful manner for plotting.

```
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")
```



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.