Lab 5

Week 5 Logistic Regression and Model Evaluation

Alexander Baxter - edited by Samuel D. Aragones

Winter Quarter 2024

Note: This version contains a few minor updates from when Alexander did the lab on 2/17/23.

Objectives

1. Logistic Regression
2. Model evaluation and comparison
3. How to interpret parameters when the model becomes more complex

```r
library(ggplot2)
library(ggpubr)
library(scales)
library(dplyr)
library(broom)
library(kableExtra)
```

We will be using the `phobia` data set today.

```r
load("phobia.Rdata")

phobia$diagnosis <-
  ifelse(phobia$ADIS_IV == 1,
         "Yes Diagnosis",
         "No Diagnosis") %>%
  factor(levels = c("No Diagnosis",
                    "Yes Diagnosis"))

phobia$gender.code <-
  ifelse(phobia$gender == 'male',
         1,
         0)
```

As a reminder, the `phobia` dataset contains the following variables:

- *SPAI_SP* is the total score on the Social Phobia and Anxiety Inventory: Sobial Phobia Subscale
- *SAS_FNE* is the total score on the Social Anxiety Scale for Adolescents: Fear of worries of negative evaluations from peers subscale
- *SAS_N* is the total score on the Social Anxiety Scale for Adolescents: Social avoidance and distress in new social situations subscale

- *SAS_G* is the total score on the Social Anxiety Scale for Adolescents: Social avoidance and distress generalized social inhibition subscale
- *FNE* is the total score on the Fear of Negative Evaluation Scale
- *SAD* is the total score on the Social Avoidance and Distress Scale
- *ADIS_IV* is the Anxiety Disorders Interview Schedule for DSM-IV (coded as 0 if no social phobia and 1 if social phobic)
- *gender* is biological sex with two levels (*male*, *female*)

The code above created two additional variables in the data that we will use later in the lab

- *gender.code* is a dummy code for sex (male = 1, female = 0)
- *diagnosis* is a text string with values "Yes Diagnosis" and "No Diagnosis" to indicate whether someone has a Social Phobia Diagnosis (based on *ADIS_IV*).

Logistic Regression

Logistic regression is a type of regression that predicts dichotomous variables as the outcome from one or more predictor. This type of analysis is often used to predict how like a specific outcome was based on various predictors in an experimental design (e.g., improving vs not improving, recovering vs not recovering, winning vs losing, etc.). However, it can technically be used to predict any type of binary variables, even if the design is correlational (for example, you could use height to predict the probability of someone being male vs female).

Logistic regression is a type of "general linear model". The "general" indicates that the way the predictors and linked to the outcome are no longer based on a normal distribution. In this specific case, the predictors are linked to the binary outcome using a logistic binary distribution.

Logistic regression and continuous regression are similar in that both analyses can tell us something about the association between two variables. However, there are important differences between logistic regression and continuous regression about how these associations are measured and conveyed.

In a continuous regression, the slope of each predictor conveys information about the predicted **value** of the outcome variable for every one-unit change in the predictor. However, in a logistic regression model, the slope of each predictor conveys information about the predicted **likelihood** of the outcome variable occurring for every one-unit change in the predictor. Logistic regression analyses *do not predict specific scores* of an outcome variable; they predict probabilities of outcomes occurring.

To do logistic regression in R, we can use the `glm` function. This function works similarly to the `lm` function, but requires one additional argument for `family`. This specifies the type of linking parameters that will be used in the analysis. To do a logistic regression, we must specify `family = "binomial"` in the `glm` function.

Let's say that we want to investigate if individuals' level of fear of negative evaluation (*FNE*) can predict whether they have social phobia or not (*ADIS_IV*).

```r
# The logistic regression model
model1 <-
  glm(ADIS_IV ~ FNE,
      family = "binomial",
      data = phobia)

# A summary of the model coefficients
coef(model1)
```

```
## (Intercept)         FNE
##  -3.9266578   0.2468106
```

We will go through a full summary of the output later in the lab. For now, lets focus on the coefficients.

The Logit Scale

By default, the `glm` function gives us regression coefficients on the "logit" scale (i.e., the log of the odds). We will explain what this means momentarily. For now, the main thing to know is that logits are on a linear scale. Hence, we could write the regression equation using a similar format that we have been using for continuous regression:

$$\widehat{logit}_{diagnosis} = -3.93 + 0.25X_{Fear}$$

In this equation, X represents a given value of FNE (the predictor).

Interpretations:

- Intercept: The model intercept was -3.93. This indicates that when the score of *FNE* equals 0, the predicted logit (*log odds*) of being diagnosed with social phobia (*diagnosis* = Yes) is -3.93.
- Slope of FNE: The slope is 0.25. This indicates that for every one-unit increase in the score of *FNE*, the logit (*log odds*) of being diagnosed with social phobia increases by 0.25.

At this point, you might be wondering what it means for "the predicted logit of a diagnosis to be -3.93" or what it means "for the logit of a diagnosis to increase by 0.25".

As we talked about in class, interpreting the results from Logistic Regression can be complicated because "a logit" is the **log odds** of a particular data point being in one category over another (when holding other predictors constant, if any). So in most cases, we don't usually interpret the coefficients on the logit scale. Instead, the outputs of a logistic regression model are usually converted to **odds ratios** or **probabilities**.

The next several chunks will demonstrate how we can convert the logistic regression outputs to these other scales.

The Odds Scale

The odds ratio scale is helpful because it expresses the relationship between a predictor and an outcome in terms of how much more likely an outcome is (whatever is coded as 1; a social phobia diagnosis in this example) relative to the reference group (whatever is coded as 0; not having a social phobia diagnosis in this example).

Some properties of the odds scale:

- The odds scale is bounded between 0 and infinity.
- A value of 1 indicates that the odds of the outcome occurring is equal to the odds of the outcome not occurring (there is equal chance).
- Values between 0 and 1 indicate the odds of the outcome occurring are less likely than the odds of the outcome not occurring (i.e., it is more likely for the outcome to not occur).
- Values above 1 indicate that the odds of the outcome occurring are more likely than not.

To convert from *logits* to *odds ratios*, we exponentiate the *logits* using the `exp()` function in R. Before we do that, lets review what exponentiation is.

```r
# Exponentiation is the inverse of a logarithm.

# Whenever you use exp(x), the output is interpreted as
# "The log of [the resulting number] is equal to x"

# For example, if you exponentiate 1, you get:
exp(1) # 2.718282
```

```
## [1] 2.718282
```

```r
# This tells us that "the log of 2.718282 is equal to 1"

# The proof:

log(2.718282)
```

```
## [1] 1
```

```r
# Hence, if you run the code below, it will always result in
# the same number that you input, as the log cancels out the exponentiation:

log(exp(1))
```

```
## [1] 1
```

```r
log(exp(47))
```

```
## [1] 47
```

```r
log(exp(-2.34))
```

```
## [1] -2.34
```

Because exponentiation is the inverse of a log, when we exponentiate the *logits* (the "log odds"), we get rid of the "log" and keep the "odds".

The code below converts the model intercept and slope from *logit* (the scale it is already on) to the *odds ratio* scale. See below for an example of how we could arrive at the same number arithmetically.

```r
# odds scale
exp(model1$coefficients)
```

```
## (Intercept)         FNE
##  0.01970944  1.27993665
```

Now that we are on the odds ratio scale, we would write the regression equation in the following format.

$$\widehat{Odds}_{Diagnosis} = 0.02 * 1.28^{X_{FNE}}$$

Interpretations:

- Intercept: When the score of *FNE* is 0, the odds of having a social phobia diagnosis ($ADIS\_IV = 1$) are 0.02.
- Slope of FNE (odds): For every one-unit increase in the score of *FNE*, the odds of having a social phobia diagnosis ($ADIS\_IV = 1$) increases by a factor of 1.28.

IMPORTANT: Now that we are on the odds-scale, the relationship between x and y is non-linear; it is exponential. So for each unit increase in *FNE*, the odds of being socially phobic are multiplied by 1.28. The intercept is multiplied by the slope, rather than added. We will demonstrate this principle later in the lab.

If you care to know, we could also have arrived at the same equation arithmetically by exponentiating the equation on the logit scale. The steps below demonstrate that process. You do not need to know how to do this by hand, it is just provided as proof of concept.

Step 1: Raise the natural logarithm (i.e., Euler's number) to the power of the entire regression equation on the logit scale.

$$\widehat{Odds}_{Diagnosis} = e^{-3.93+0.25X_{FNE}}$$

Step 2: Use the rules of exponents to separate the intercept and slope (for a refresher on these rules, see this link).

$$\widehat{Odds}_{Diagnosis} = e^{-3.93} * e^{0.25X_{FNE}}$$

Step 3: Simplify the equation.

$$e^{-3.93} = 0.02$$

$$e^{0.25X_{FNE}} = 1.28^{X_{FNE}}$$

Hence:

$$\widehat{Odds}_{Diagnosis} = 0.02 * 1.28^{X_{FNE}}$$

The Probability Scale

In most cases, you will report the results of a logistic regression equation on the odds ratio scale. However, in some cases, the probability scale could also be useful.

Some properties of the probability scale:

- The probability scale is bounded between 0 and 1.
- A value of 0.5 indicates 50% (i.e., the probability of the outcome occurring is the same as the probability of the outcome not occurring).
- A value of 0 indicates a 0% chance of the outcome occurring, and a value of 1 indicates a 100% chance of the outcome occurring. Predicted probabilities will almost never be equal to exactly 0 or exactly 1, as this would imply absolute certainty in the prediction (this is the same reason why p values are never equal to exactly 0).
- Values above 50% indicate that the probability of the outcome occurring is more likely than not.

- Values below 50% indicate that the probability of the event occurring is less likely than not.

To go from Odds Ratio to Probability, we apply the following equation:

$$probability = \frac{Odds}{1 + Odds}$$

We could apply this conversion factor to the intercept from the equation above.

```
# For the intercept:

exp(model1$coefficients[1]) / (1 + exp(model1$coefficients[1]))
```

```
## (Intercept)
##  0.01932848
```

Interpretation:

- Intercept: When FNE is 0, the probability of having a social phobia diagnosis is 1.9%.

IMPORTANT: The probability function is sigmoidal (S-shaped), and so even if we converted the slope for FNE from odds ratios to probability, it could not be meaningfully interpreted. The probability scale is only useful for interpreting the predicted likelihood at specific values of x (such as the intercept).

For example, if we plugged in specific values of X to the regression equation, we could get an estimated likelihood on the logit, odds, and probability scale. For example, lets assess someone with average levels of *FNE*.

```
# The average level of FNE is:
mean(phobia$FNE)
```

```
## [1] 19.74917
```

```
# How likely is a social phobia diagnosis for someone with average levels of FNE?

predicted.logit_example.1 <-
  coef(model1)[1] +    # Intercept (on the logit scale)
  (coef(model1)[2] *   # Slope (on the logit scale)
     mean(phobia$FNE) # Input value of x (average FNE)
   )

# Logit of diagnosis at average FNE
predicted.logit_example.1
```

```
## (Intercept)
##   0.9476476
```

```
# Odds of a diagnosis at average FNE
exp(predicted.logit_example.1)
```

```
## (Intercept)
##    2.579634
```

```
# Probability of a diagnosis at average FNE
exp(predicted.logit_example.1) / (1 + exp(predicted.logit_example.1))
```

```
## (Intercept)
##   0.7206418
```

Interpretations:

- **Logit scale**: Someone with average levels of FNE has 0.94 log-odds of having a social phobia diagnosis (this is not very interpretable or meaningful; on to the next scales!).
- **Odds scale**: Someone with average levels of FNE is 2.5 times more likely to have a social phobia diagnosis than not.
- **Probability scale**: Someone with average levels of FNE has a predicted 72% chance of having a social phobia diagnosis.

Note, if you are ever in a situation where you need to go from probability to odds, you can convert with this equation:

$$Odds = \frac{p}{1-p}$$

Demonstration with Raw Data

In the section above, we assessed the probability of someone having a social phobia diagnosis for someone with 0 units of FNE (the intercept) and for someone with average levels of FNE. The next section will provide further demonstration of how logistic regression works by visualizing the predicted likelihoods across all possible values of FNE.

To start, we will create three new columns in our data frame for predicted *logits*, predicted *odds ratios*, and predicted *probability*.

To get the predicted *logits*, we can use the regression equation from the model. To get the odds and probabilities, we can apply to conversion formulas from above.

```
# Logits
phobia$logit <-
  coef(model1)[1] +              # The intercept
  (coef(model1)[2] * phobia$FNE) # The slope times each value of FNE in the data

# Odds
phobia$odds <-
  exp(phobia$logit) # Predicted Logit

# Probability
phobia$prob <-
  phobia$odds / (1 + phobia$odds) # Predicted Probability
```

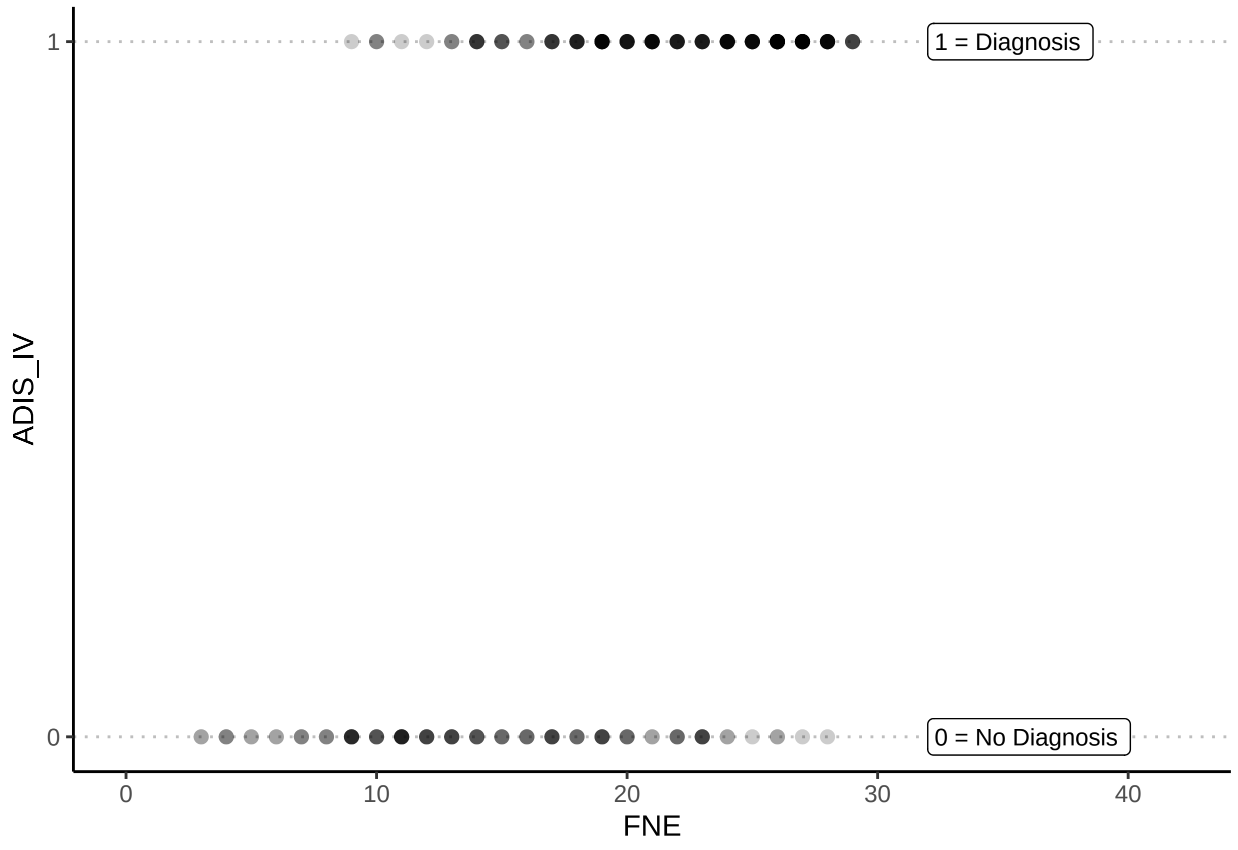
Lets do some graphing now. We are going to work from probabilities back to the logit scale.

Probability Scale

First, lets start with a simple graphic depicting the analysis we just did.

```r
ggplot(data = phobia,
       aes(x = FNE, y = ADIS_IV)) +
  geom_point(alpha = .2, size = 2) +
  theme_classic()  +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0,42)) +
  scale_y_continuous(breaks = c(0, 1)) +
  annotate(geom = "label",
           label = c("0 = No Diagnosis", "1 = Diagnosis"),
           x = 32,
           hjust = 0,
           y = c(0, 1),
           size = 3)
```

```
## Warning: The ‘size‘ argument of ‘element_line()‘ is deprecated as of ggplot2 3.4.0.
## i Please use the ‘linewidth‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



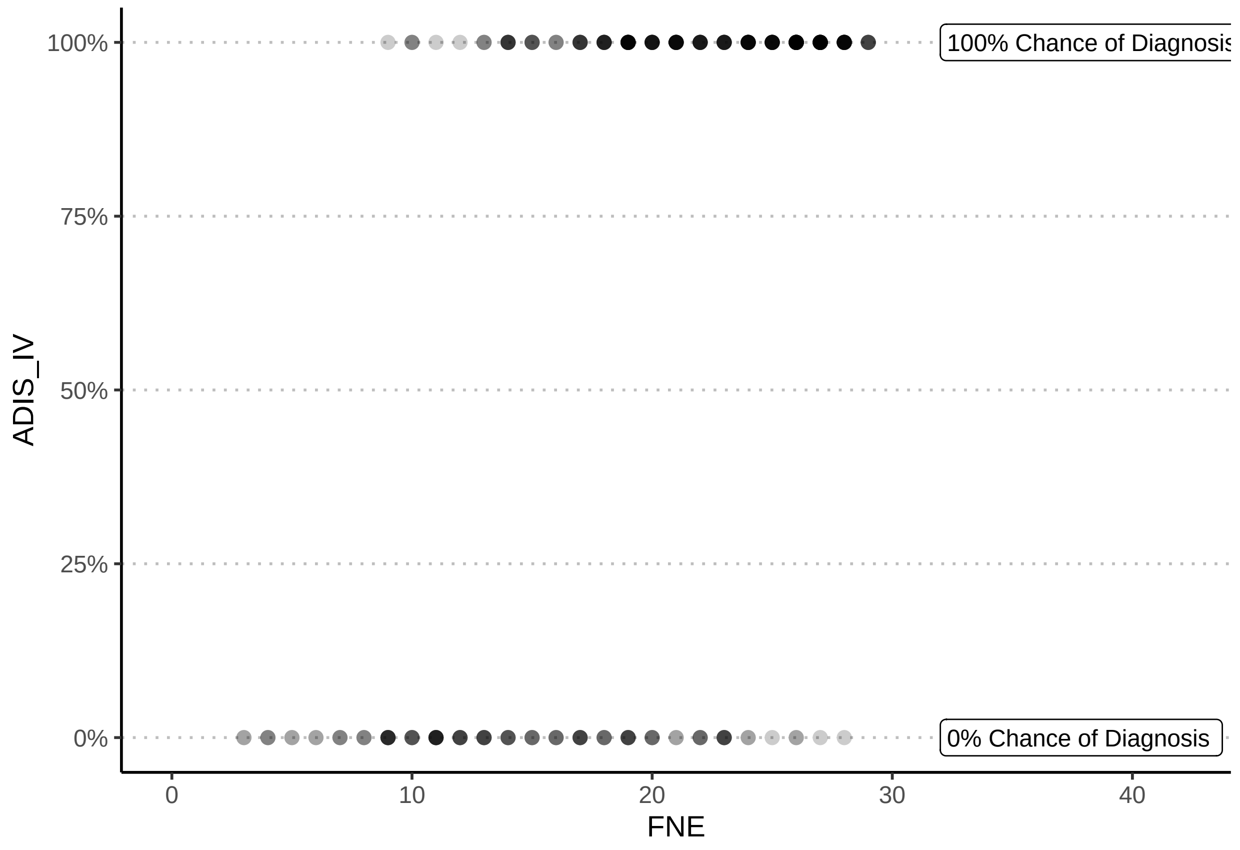
Remember, we coded *ADIS_IV* so that 0 = no diagnosis, 1 = social phobia diagnosis. Right now, the numbers are somewhat arbitrary; they only represent certain groups (people with and without a social phobia diagnosis).

The nice thing about 0 and 1 is that they could also represent probabilities. In other words, someone who is diagnosed with a social phobia disorder (ADIS_IV = 1) has a 100% chance of having a social phobia disorder, whereas someone with no social phobia diagnosis (ADIS_IV = 0) has a 0% chance of having a social phobia disorder. Note, in the real world, there could potentially be cases in which people have mis-diagnosed or un-diagnosed social phobia disorders; in this case, when I say "someone has a 100% chance of having a social phobia disorder", I mean they have a 100% chance of being coded as having a social phobia disorder in this data set.

This logic might seem overly-simple, but it is an important bridge from going from the observed data to the logistic regression.

Using this logic, we could recreate the graph above using percentages.

```r
ggplot(data = phobia,
       aes(x = FNE, y = ADIS_IV)) +
  geom_point(alpha = .2, size = 2)  +
  theme_classic()  +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0,42)) +
  scale_y_continuous(breaks = c(0,.25, .5, .75, 1),
                     label = percent) +
  annotate(geom = "label",
           label = c("0% Chance of Diagnosis", "100% Chance of Diagnosis"),
           x = 32,
           hjust = 0,
           y = c(0, 1),
           size = 3)
```



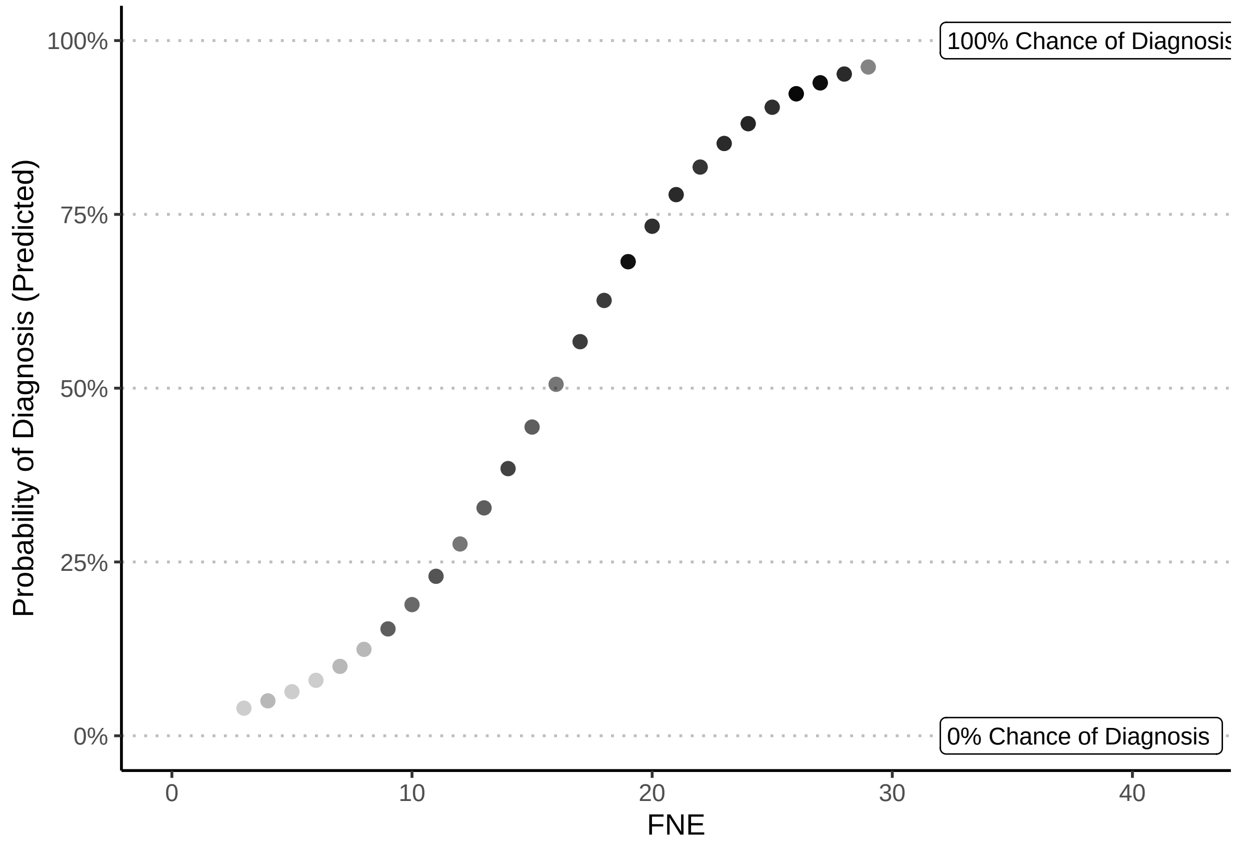
As you look at the graph, you will start to notice some patterns. For example, the people coded as ADIS_IV = 1 (100%) seem to be clustered further to the right.

If we were to flip this analysis (and use ADIS_IV to predict FNE, rather than the other way around), we might conclude that, on average, people with a social phobia diagnosis have higher levels of FNE than people without a social phobia diagnosis. We could use a t-test to formally test this hypothsis.

In this case, though, we are interested in whether FNE predicts ADIS_IV. This is the key idea behind logistic regression: We are testing whether the score of x (FNE in this case) predicts greater likelihood of y (ADIS_IV in this case).

Let's assess whether this initial observation is supported by the logistic regression analysis that we did above. To do this, we will start on the probability scale, and we will graph the predicted probabilty of having a social phobia diagnosis for each value of FNE in the data. We graph these predicted probabilities instead of the actual data.

```
ggplot(data = phobia,
       aes(x = FNE, y = prob)) +
  geom_point(alpha = .1, size = 2) +
  labs(y = "Probability of Diagnosis (Predicted)") +
  theme_classic()  +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0,42)) +
  scale_y_continuous(breaks = c(0,.25, .5, .75, 1),
                     label = percent) +
  annotate(geom = "label",
           label = c("0% Chance of Diagnosis", "100% Chance of Diagnosis"),
           x = 32,
           hjust = 0,
           y = c(0, 1),
           size = 3)
```



This graph shows us the predicted probability of having a social phobia diagnosis for each value of FNE. We can see that, in general, higher scores on *FNE* are associated with higher probability of a social phobia diagnosis. This matches the general trend we saw in the previous graph.

Now, you might be wondering: How is it possible to have anything but a 0% or 100% probability of a diagnosis, when it is a Yes/No variable.

The answer is that this the *predicted* probability. The reason it is not 0% or 100% is because there is overlap between the points. For example, at most values of *FNE*, there are some people who do have a diagnosis (even when the probability is low) and some people who do not (even when the probability is high). The probabilities ranging between 0-100% reflect this overlap. We will get into this point in greater detail later in the lab when we talk about error rate.

In the previous section, we used the probability scale to interpret specific predicted likelihoods from the logistic regression equation (using FNE = 0 and the mean of FNE as an example). The graph above summarizes all possible predicted values.

If you wanted to determine the predicted probabilities for any other specific values of FNE, you could either use the code above, or you could use the `predict()` function. This function will return the predicted probability (if you set `type = 'resp'` as one of the argument) from a given list of inputs (i.e., values for each coefficient in the regression equation). The chunk below demonstrates how this would work at different values of *FNE* (the max, min, and mean).

```
# At the max of FNE:
predict(model1,
        list(FNE = max(phobia$FNE, na.rm = TRUE)),
        type = 'resp')
```

```
##         1
## 0.9619788
```

```
# At the min of FNE:
predict(model1,
        list(FNE = min(phobia$FNE, na.rm = TRUE)),
        type = 'resp')
```

```
##          1
## 0.03968736
```

```
# At the mean of FNE:
predict(model1,
        list(FNE = mean(phobia$FNE, na.rm = TRUE)),
        type = 'resp')
```

```
##         1
## 0.7206418
```

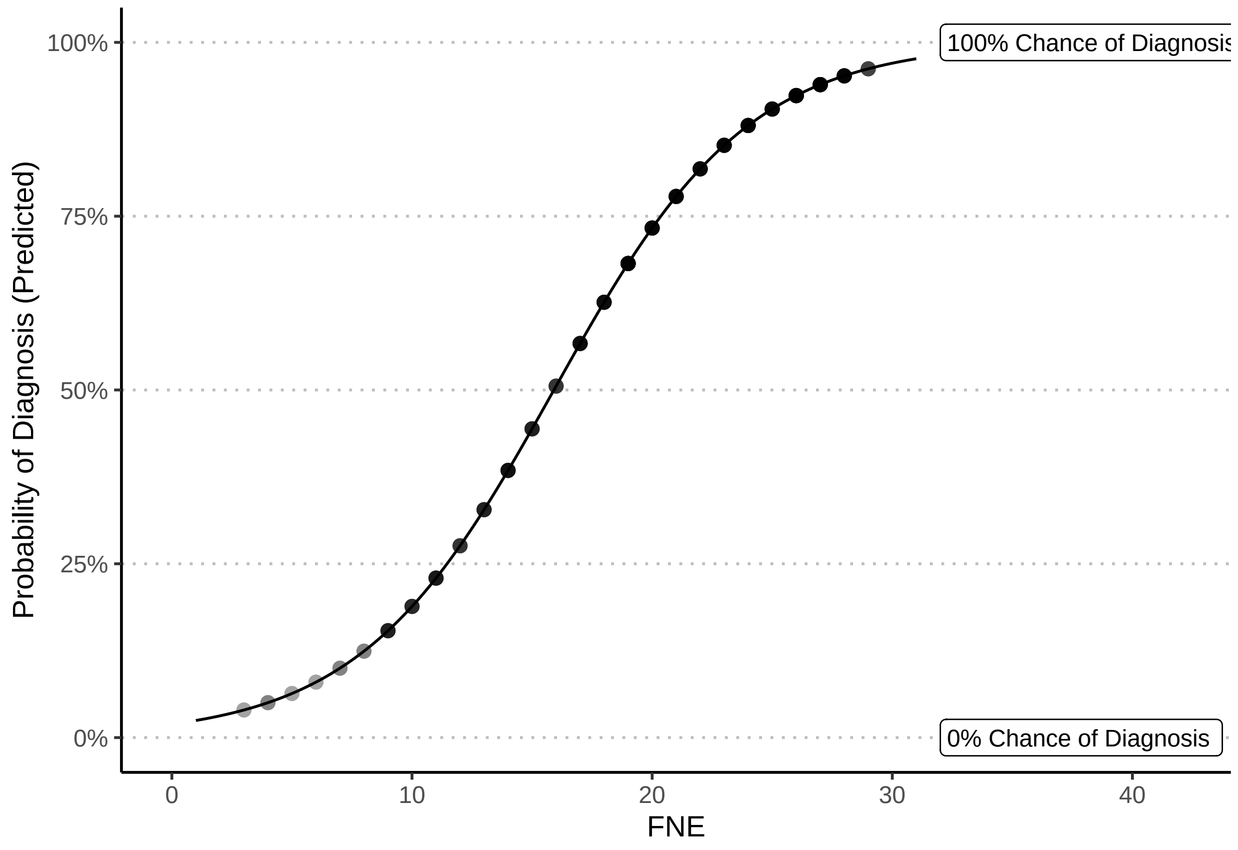
- At the highest value of FNE (29), the probability of a social phobia diagnosis is 96%.
- At the lowest value of FNE (3), the probability of a social phobia diagnosis is 4%.
- At the average value of FNE (19.75), the probability of a social phobia diagnosis is 72%.

Note on the predict function: Each value in the list must be named according to how the variables were entered in the regression object that you input. In this case, this means that because the 'model1' object had FNE as a predictor, we must name what value to use for FNE. This value must have the exact same name as the predictor in the model (FNE). If this is confusing or if you just don't want to deal with the `predict()` function, an alternative approach is to manually calculate the predicted likelihoods on the logit scale (by using the estimates from the regression equation and plugging in values of X), and then convert to estimated logit to the odds ratio or probability scale.

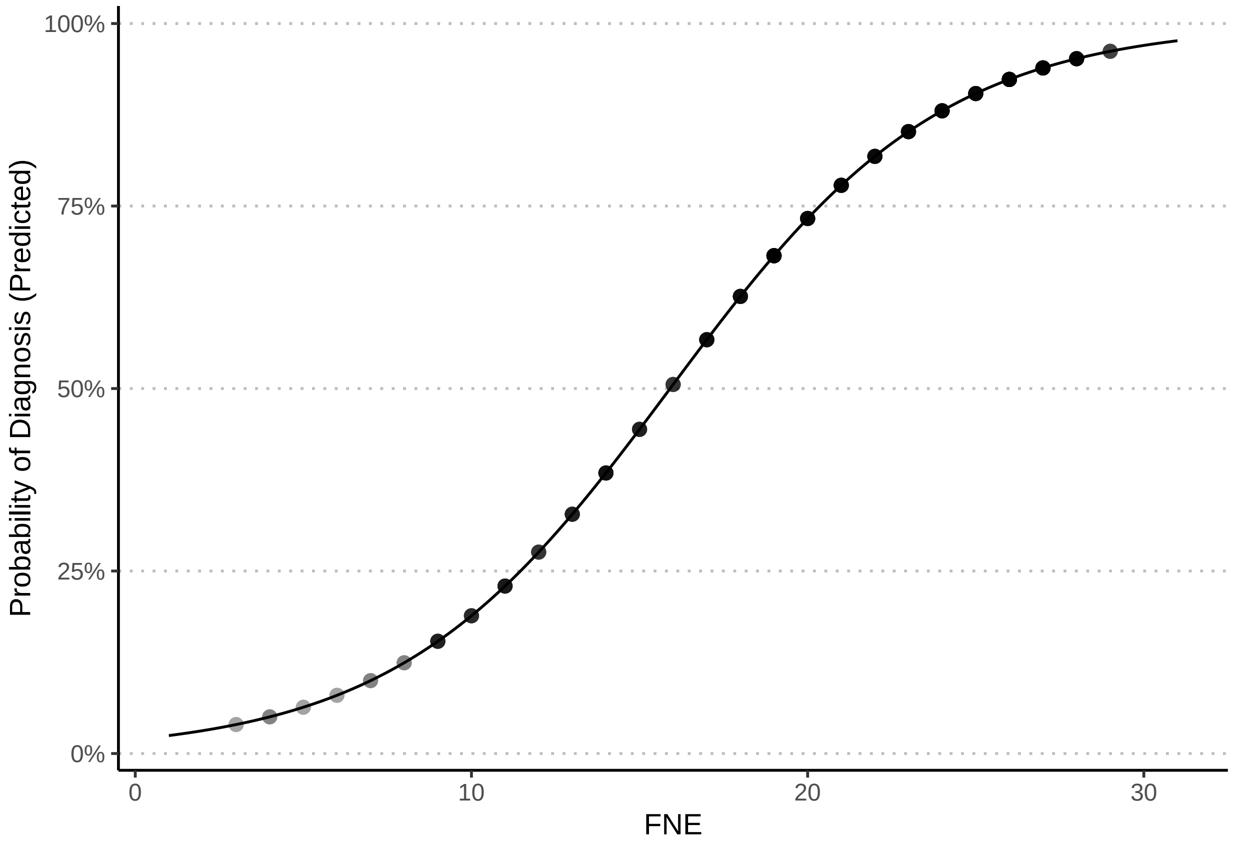
Returning to the plot above: In the plot above, we added the predicted probabilities for each value of x; however, we did not add a fit line. Adding the fit line to the plot is a little complicated, because there is not a nice function to add custom non-linear fit lines. A way around this is to just create a separate data frame. One column will contain the range of *x* (or a range that encompasses x and beyond) in small increments (i.e., increments of 0.01, or potentially less depending on the metric of the original data). The `seq` function is useful for creating sequences like this. You can then feed each of these values into the logistic regression equation, and create additional columns for the predicted logit, odds, and/or probabilities at each value in the range of x. This extra data frame can then be added to the graph as a separate layer on top of the data.

```
# First, create a new data frame with a single column with a sequence of values
# that spans the range of x (or even that exceeds the range of x). A good way
# to do this is to use the min and max values of x, plus and minus 1. We will call
# our new data frame 'line.data'.

line.data <-
```

```r
  data.frame(FNE.range =
               seq(from = min(phobia$FNE, na.rm = T) - 2,
                   to = max(phobia$FNE, na.rm = T) + 2,
                   by = .01))

# Next, determine the predicted probability for each value in the sequence by
# running it through the logisitic regression equation. This chunk does this
# manually using the logistic regression equation. See the next chunk for an
# example of how you could do this using the predict function instead.

# First, determine predicted logit using regression equation

line.data$logit <-
  model1$coefficients[1] +
  (line.data$FNE.range * model1$coefficients[2])

# Then, convert the predicted logits to the odds and probability scale:

line.data$odds <- exp(line.data$logit)
line.data$prob <- line.data$odds / (1 + line.data$odds)

# Last, add these values as a geom_line to the graph:

# The graph:
ggplot(data = phobia,
       aes(x = FNE, y = prob)) +

  # The prediction line
  geom_line(data = line.data,
            aes(x = FNE.range, y = prob)) +

  # The rest of the graph
  geom_point(alpha = .2, size = 2) +
  labs(y = "Probability of Diagnosis (Predicted)") +
  theme_classic()  +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0,42)) +
  scale_y_continuous(breaks = c(0,.25, .5, .75, 1),
                     label = percent) +
  annotate(geom = "label",
           label = c("0% Chance of Diagnosis", "100% Chance of Diagnosis"),
           x = 32,
           hjust = 0,
           y = c(0, 1),
           size = 3)
```



```
# Example of how you could determine the fit line with the predict function
## This chunk replicates the graph above

# The external data frame:
line.data <-
  data.frame(FNE.range =
             seq(from = min(phobia$FNE, na.rm = T) - 2,
                 to = max(phobia$FNE, na.rm = T) + 2,
                 by = .01))


# First, determine the predicted probabilities using the predict function

line.data$prob <-
  predict(model1,
          list(FNE = line.data$FNE.range),
          type = "resp")

# Then convert the predicted probabilities to odds and logits

line.data$odds <- line.data$prob / (1 - line.data$prob)
line.data$logit <- log(line.data$odds)

# The graph

ggplot(data = phobia,
       aes(x = FNE, y = prob)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = prob)) +
  geom_point(alpha = .2, size = 2) +
  labs(y = "Probability of Diagnosis (Predicted)") +
  theme_classic()  +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_y_continuous(breaks = c(0,.25, .5, .75, 1),
                     label = percent)
```



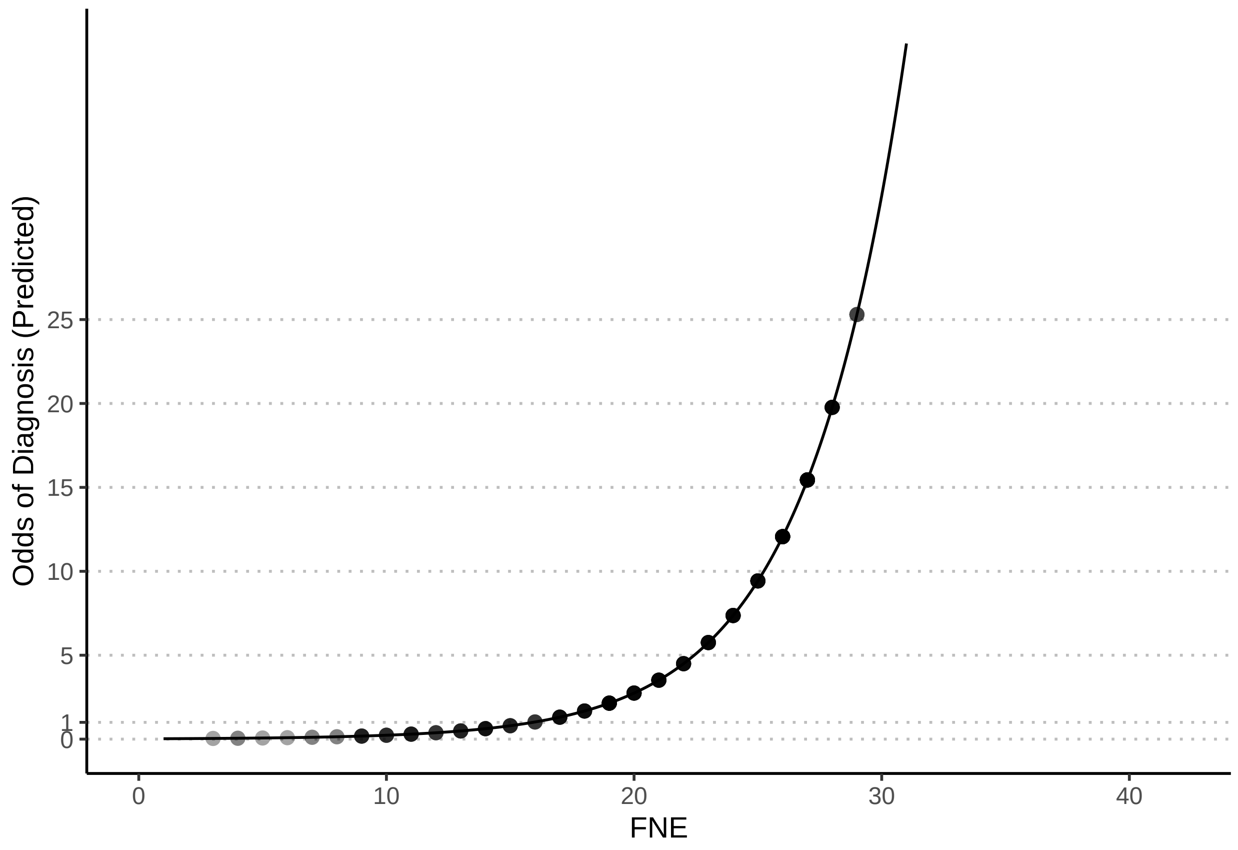
```
# Note: For this graph, I removed the annotations (those were just for
# demonstration, and you normally would not include something like that on a final
# product).
```

Odds Scale

Next, we will look at the predicted odds of having a social phobia diagnosis on the *odds ratio* scale.

```
ggplot(data = phobia,
       aes(x = FNE, y = odds)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = odds)) +
  geom_point(alpha = .2, size = 2) +
```

```r
  theme_classic()  +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0, 42)) +
  scale_y_continuous(breaks = c(0,1,5,10,15,20,25)) +
  labs(y = "Odds of Diagnosis (Predicted)")
```

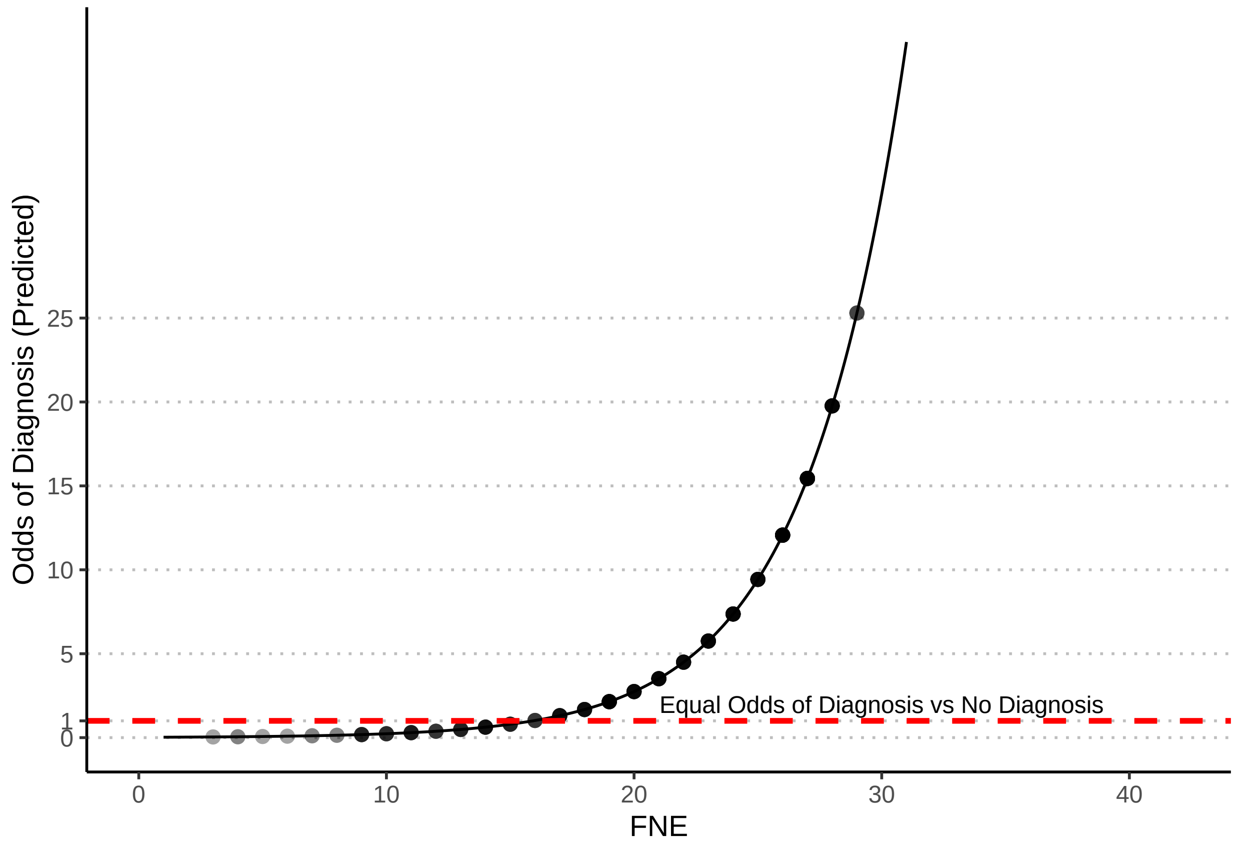


The y-axis is now on the odd-scale, and the relationship between *FNE* and the *odds ratio* of a diagnosis is exponential.

Remember, an odds ratio of 1 is equivalent to a 50% probability. This means that at the values of *FNE* with predicted odds lower than 1, it is more likely to have no diagnosis; for the values of FNE with predicted odds greater than 1, it is more likely to have a diagnosis. The graph below shows a summary of this principle.

```r
ggplot(data = phobia,
       aes(x = FNE, y = odds)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = odds)) +
  geom_point(alpha = .2, size = 2) +
  theme_classic()  +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0, 42)) +
  scale_y_continuous(breaks = c(0,1,5,10,15,20,25)) +
  labs(y = "Odds of Diagnosis (Predicted)")  +
  geom_hline(yintercept = 1,
             size = 1,
             color = "red",
             linetype = "dashed") +
  annotate(geom = "text",
           label = "Equal Odds of Diagnosis vs No Diagnosis",
           x = 30,
           y = 2,
           size = 3)
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

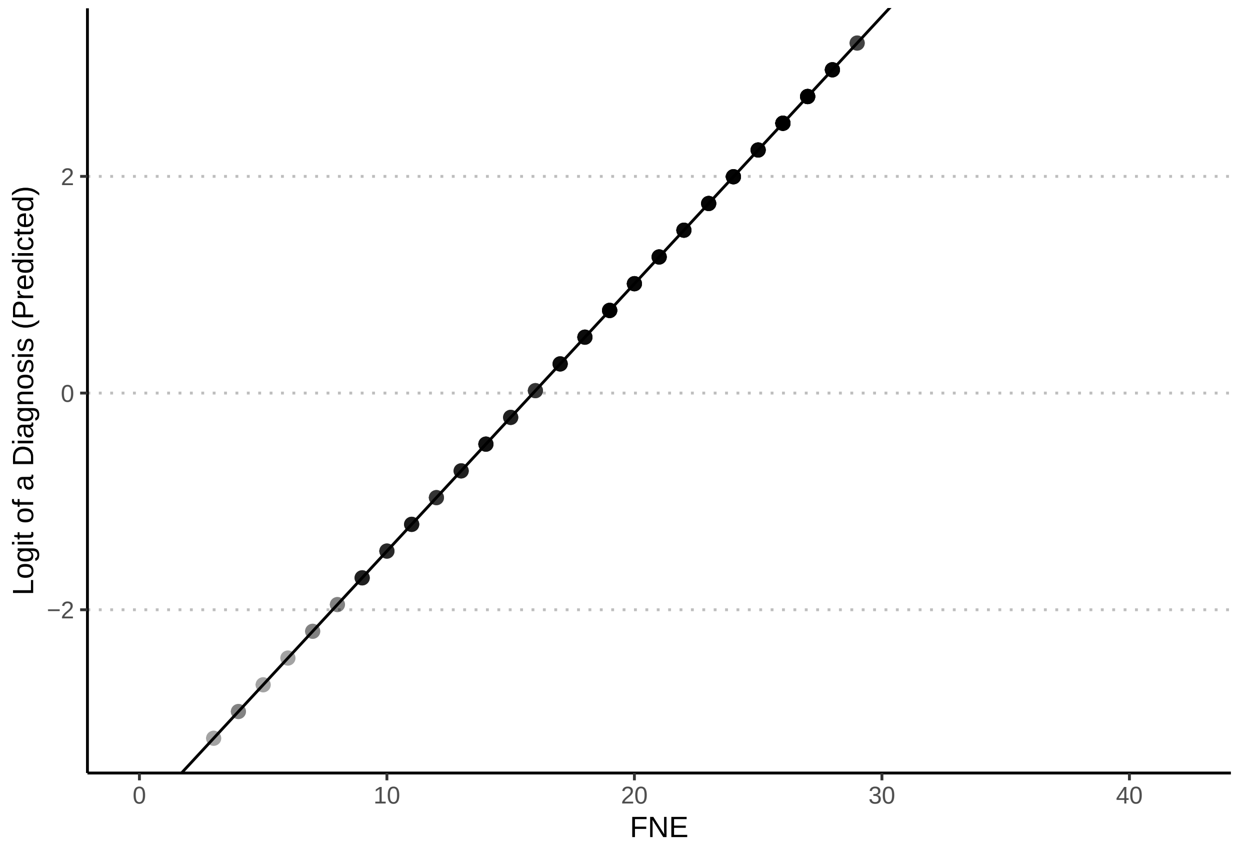


The fit line is an exponential function, and corresponds to the parameters estimated above (i.e., $y = 0.02 * 1.28^{X_{FNE}}$).

Logit Scale

Last, lets look at the relationship between *FNE* and the *logit* of having a diagnosis.

```
ggplot(data = phobia,
       aes(x = FNE, y = logit)) +
  geom_abline(intercept = model1$coefficients[1],
              slope = model1$coefficients[2]) +
  geom_point(alpha = .2, size = 2) +
  theme_classic() +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0, 42)) +
  labs(y = "Logit of a Diagnosis (Predicted)")
```

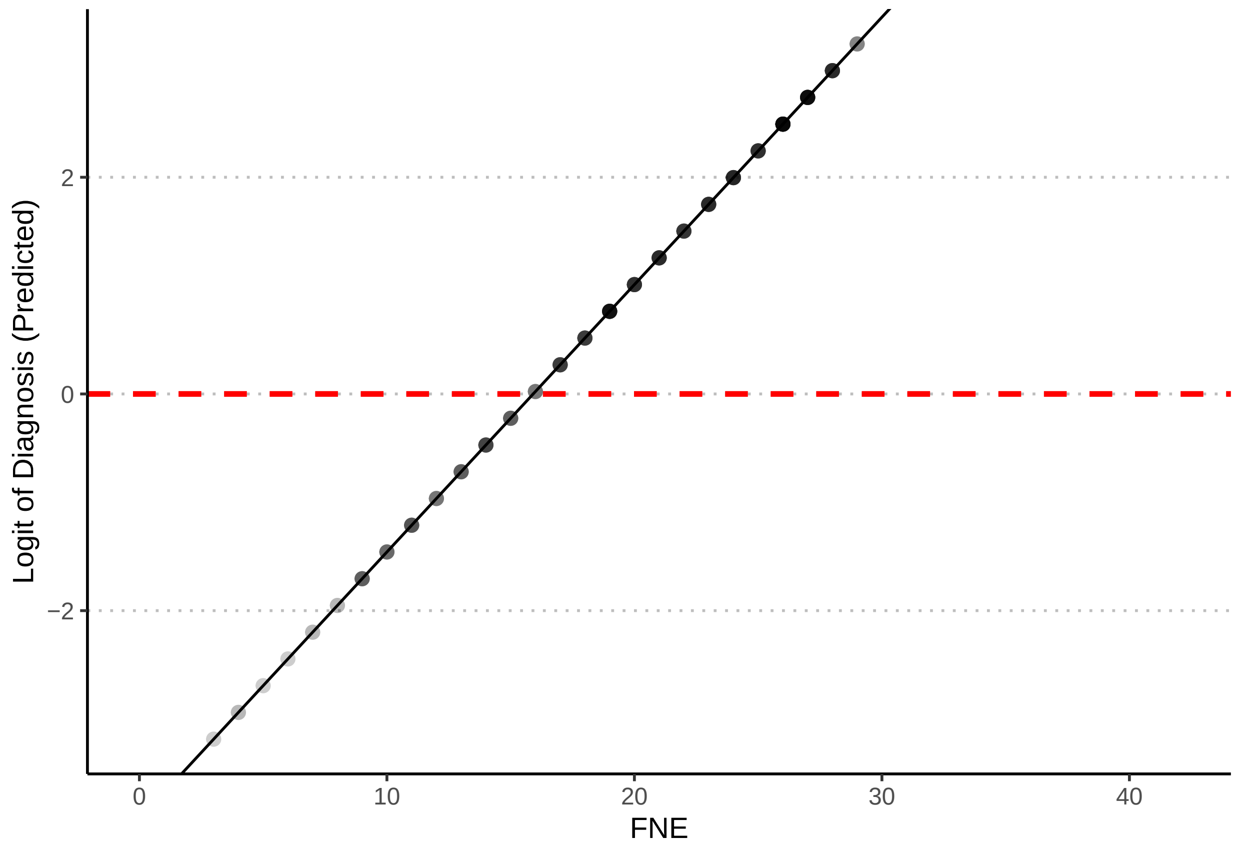


```
# Note, We don't need to use the line.data to add the fit line to this graph,
# as we are on a linear scale, so we can just add a custom linear line to the graph
# with geom_abline, and input the slope and intercept.
```

The relationship between *FNE* and the *logit* of diagnosis is perfectly linearly, and is defined by the equation we wrote above (i.e., $y = -3.93 + 0.25X_{FNE}$).

These *logit* units represent the predicted log-odds of having a social phobia diagnosis. These units are not very intuitive. However, if it helps, remember that a *logit* of $0 =$ *odds-ratio* of $1 =$ *probability* of 50%. Hence, positive logits represent greater likelihood of a diagnosis and negative logits represent lower likelihood of a diagnosis.

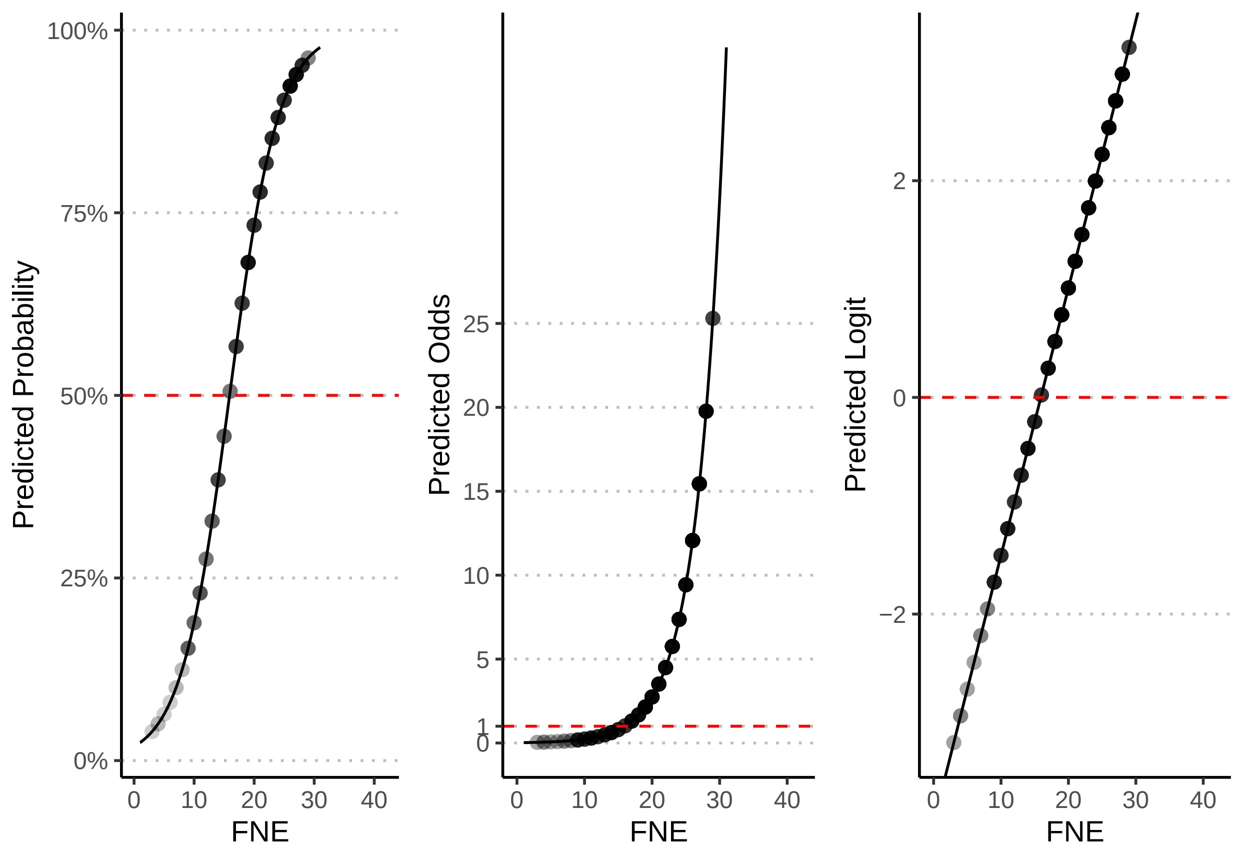
```
ggplot(data = phobia,
       aes(x = FNE, y = logit)) +
  geom_abline(intercept = model1$coefficients[1],
              slope = model1$coefficients[2]) +
  geom_point(alpha = .1, size = 2) +
  theme_classic()  +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0, 42)) +
  labs(y = "Logit of Diagnosis (Predicted)") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red", size = 1)
```



Key Points: Logits vs Odds Ratio vs Probability

This section provides a summary of everything we just did.

```
g1 <-
  ggplot(data = phobia,
       aes(x = FNE, y = prob)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = prob)) +
  geom_point(alpha = .1, size = 2)  +
  theme_classic()  +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0,42)) +
  scale_y_continuous(breaks = c(0,.25, .5, .75, 1),
                     label = percent) +
  labs(y = "Predicted Probability") +
  geom_hline(yintercept = .5, linetype = "dashed", color = "red")


g2 <-
  ggplot(data = phobia,
       aes(x = FNE, y = odds)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = odds)) +
  geom_point(alpha = .2, size = 2) +
  theme_classic()  +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0, 42)) +
  scale_y_continuous(breaks = c(0,1,5,10,15,20,25)) +
  labs(y = "Predicted Odds")  +
  geom_hline(yintercept = 1, linetype = "dashed", color = "red")

g3 <-
  ggplot(data = phobia,
       aes(x = FNE, y = logit)) +
  geom_abline(intercept = model1$coefficients[1],
              slope = model1$coefficients[2]) +
  geom_point(alpha = .2, size = 2) +
  theme_classic() +
  theme(panel.grid.major.y =
          element_line(size = .5, color = "grey", linetype = "dotted")) +
  scale_x_continuous(limits = c(0, 42)) +
  labs(y = "Predicted Logit") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red")


ggarrange(g1,g2,g3, ncol = 3)
```



And now lets compare the estimates:

```r
data.frame(
  scale = c("probability",
            "odds-ratio",
            "logit"),
  intercept = c(
    exp(model1$coefficients[1]) / (1 + exp(model1$coefficients[1])),
    exp(model1$coefficients[1]),
    model1$coefficients[1]
  ),
  slope = c(
    exp(model1$coefficients[2]) / (1 + exp(model1$coefficients[2])),
    exp(model1$coefficients[2]),
    model1$coefficients[2]
  )) %>%
  kable(digits = 2)
```

scale	intercept	slope
probability	0.02	0.56
odds-ratio	0.02	1.28
logit	-3.93	0.25

In all three cases, the Intercept can always be interpreted as the predicted value of y (on the respective scale) when x is 0. Hence:

- When *FNE* is 0, the predicted probability of a diagnosis is 0.0193 (1.93%)

- When *FNE* is 0, the predicted odds of a diagnosis is 0.0197
- When *FNE* is 0, the predicted logit of a diagnosis is -3.93

The interpretation of the slope depends on the scale.

- For probability, the slope can't be interpreted. Instead we interpret the probability of the outcome occurring at specific values of x.
- For odds-ratio, the odds of a diagnosis increases by a factor of 1.28 (i.e., exponentially) for every one-unit increase in *FNE*.
- For logit, the logit of a diagnosis increases by 0.24 (i.e., linearly) for every one unit increase in *FNE*.

Here are a few key points to remember:

- **Logits**: Linear scale, bounded between negative and positive infinity (i.e., and value possible)
- **Odds Ratio**: Exponential Scale, bounded between 0 and infinity (i.e., no negative numbers)
- **Probability**: Bounded between 0 and 1, typically has a sigmoid (S-shaped) distribution pattern

Interpreting the Model

Now that we have walked through the coefficients from the logistic regression model, lets walk through the the `summary` output for the logistic model.

```r
summary(model1)
```

```
##
## Call:
## glm(formula = ADIS_IV ~ FNE, family = "binomial", data = phobia)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.92666    0.54969  -7.143 9.11e-13 ***
## FNE          0.24681    0.02909   8.484  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 385.73  on 302  degrees of freedom
## Residual deviance: 273.46  on 301  degrees of freedom
## AIC: 277.46
##
## Number of Fisher Scoring iterations: 5
```

- **Call**: This is the code that was used to generate the model. This can be a helpful reminder about what you did, especially if you call the regression object later on in your code after defining it (as we did in this example).

- **Deviance Residuals**: In a regular linear regression, the residuals measure how "off" each prediction was. In a binomial logistic regression, we use "deviance" as a measure of error, rather than residuals. We will cover more information about this in next week's lab. For now, just know that deviance is a measure of dispersion. The output above lets you see how an "at a glance" summary of how the residuals are distributed (one of the model assumptions is that the deviance residuals are normally distributed).

- **Coefficients**: These are the estimates for the intercept and slope of each predictor (see the sections above for a detailed description of how to interpret the estimates). The p value for FNE is < 0.05, which indicates that FNE is a significant predictor of social phobia diagnosis. Note, although the estimates are displayed on the logit scale, the p value will not change, regardless of if you convert the logit to an odds ratio or probability.
- **Other information at the bottom of the output**: These parameters pertain to other measures of model fit and deviance. These values will be explained in greater detail in a future lab.

Model Evaluation: Accuracy Rate

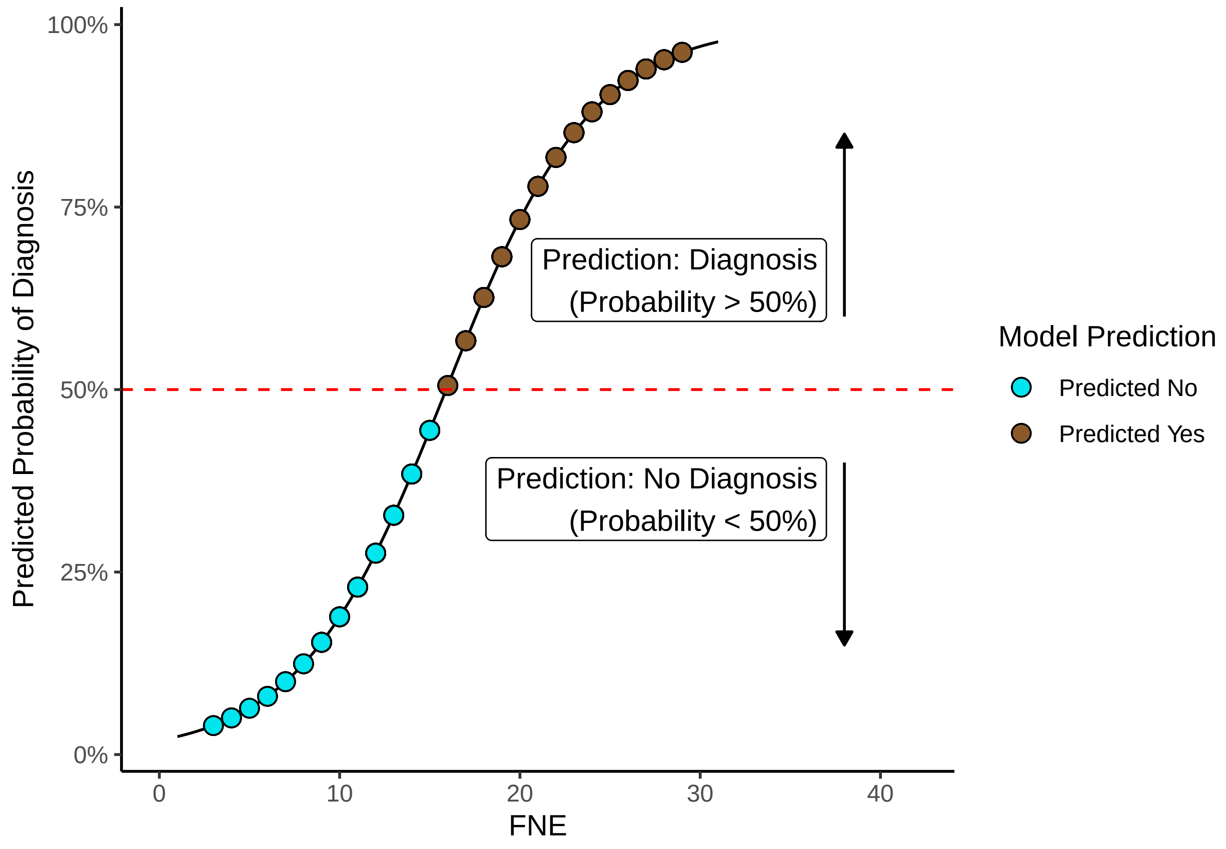
When we do logistic regression, we get p values for the intercept and slope, but we do not get an overall model p value (like we do when we do continuous regression). This is because the logistic regression model is inherently about probabilities. So, how can we know if our model is a good one?

Error Rate: Better than chance?

One way to evaluate your model is to look at the error rate.

Lets return to our graph of predicted probabilities. Recall, that when the probability is above 50%, it means that the probability of the outcome occurring is greater than the probability of it not occurring. Although none of the predicted probabilities were 100% certain, if we HAD to use the predicted probabilities to guess whether each data point had a social phobia diagnosis or not, the safest guess would be to guess Yes if the probability was greater than 50% and to guess no if the probability was less than 50%. The graph below demonstrates this process.

```r
phobia$pred.class <-
  ifelse(phobia$prob > .5,
         "Predicted Yes",
         "Predicted No")

phobia  %>%
ggplot(aes(x = FNE, y = prob)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = prob)) +
  geom_point(aes(fill = pred.class),
             shape = 21,
             color = "black",
             size = 3) +
  scale_fill_manual("Model Prediction",
                    values = c("turquoise2","tan4")) +
  theme_classic() +
  scale_y_continuous(labels = percent) +
  scale_x_continuous(limits = c(0, 42)) +
  labs(y = "Predicted Probability of Diagnosis") +
  geom_hline(yintercept = .5,
             linetype = "dashed",
             color = "red") +
  annotate(geom = "label",
           label = c("Prediction: No Diagnosis\n(Probability < 50%)",
                     "Prediction: Diagnosis\n(Probability > 50%)"),
           x = 37,
           y = c(.35, .65),
           hjust = 1) +
  annotate(geom = "segment",
           x = 38,
           xend = 38,
           y = c(.4, .6),
           yend = c(.15, .85),
           arrow = arrow(type = "closed", length = unit(0.02, "npc")))
```



The graph above shows the PREDICTED classification of social phobia diagnosis based on FNE. How well did the model do? In this case, we KNOW each participant's diagnosis status (ADIS_IV), so we can use that information to evaluate how well the model classified social phobia diagnosis.

Lets first look at this in a tabular format.

```
tab <- table(phobia$pred.class, phobia$diagnosis)

# Raw Numbers
tab
```

```
##
##                  No Diagnosis Yes Diagnosis
##   Predicted No             58            21
##   Predicted Yes            43           181
```

```r
# Percentages
round(tab / sum(tab), 2)
```

```
##
##                  No Diagnosis Yes Diagnosis
##   Predicted No           0.19          0.07
##   Predicted Yes          0.14          0.60
```

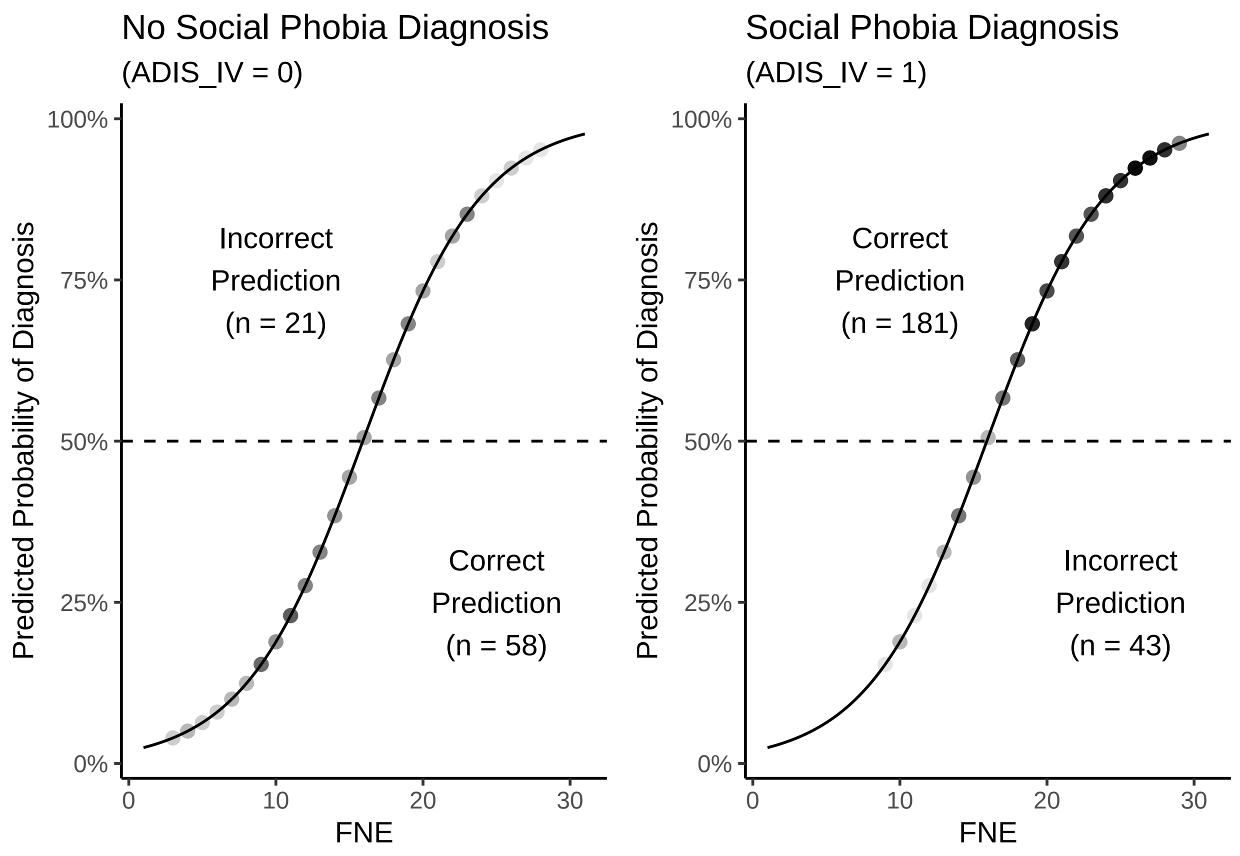
The output tells us the following: Based only on FNE, there were...

- 58 people correctly predicted to NOT have a diagnosis (19% of cases).
- 181 people correctly predicted to HAVE a diagnosis (60% of cases).
- 21 people incorrectly predicted to NOT have a diagnosis (when they really did have a diagnosis) (7% of cases).
- 43 people incorrectly predicted to HAVE a diagnosis (when they really did not have a diagnosis) (14% of cases).

In total, there were 239 correct predictions (58+181) (79% of cases) and 64 incorrect predictions (21+43) (21% of cases).

The graph below shows a visual summary of this tabulation.

```r
g_outcome.0 <-
  phobia %>%
  filter(ADIS_IV == 0) %>%
  ggplot(aes(x = FNE, y = prob)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = prob)) +
  geom_point(alpha = .1,
             size = 2) +
  theme_classic() +
  scale_y_continuous(labels = percent) +
  labs(y = "Predicted Probability of Diagnosis",
       title = "No Social Phobia Diagnosis",
       subtitle = "(ADIS_IV = 0)") +
  geom_hline(yintercept = .5, linetype = "dashed") +
  annotate(geom = "text",
           label = "Correct\nPrediction\n(n = 58)",
           x = 25,
           y = .25)  +
  annotate(geom = "text",
           label = "Incorrect\nPrediction\n(n = 21)",
           x = 10,
           y = .75)

g_outcome.1 <-
  phobia %>%
  filter(ADIS_IV == 1) %>%
  ggplot(aes(x = FNE, y = prob)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = prob)) +
  geom_point(alpha = .1, size = 2) +
  theme_classic() +
  scale_y_continuous(labels = percent) +
  labs(y = "Predicted Probability of Diagnosis",
       title = "Social Phobia Diagnosis",
       subtitle = "(ADIS_IV = 1)") +
  geom_hline(yintercept = .5, linetype = "dashed") +
  annotate(geom = "text",
           label = "Incorrect\nPrediction\n(n = 43)",
           x = 25,
           y = .25)  +
  annotate(geom = "text",
           label = "Correct\nPrediction\n(n = 181)",
           x = 10,
           y = .75)

ggarrange(g_outcome.0, g_outcome.1)
```



The code below automates these tabulations.

```r
# Prep Work
## These were steps that we completed in the chunks above. These steps
## are copied here for convenience, as these steps must be completed for the
## code in the second half of this chunk to work.
phobia$pred.class <-
  predict(model1,
          list(FNE = phobia$FNE),
          type = "resp")

phobia$pred.class <-
  ifelse(phobia$prob > .5,
         "Predicted Yes",
         "Predicted No")

tab <- table(phobia$pred.class, phobia$diagnosis)

# Automated Process
tot.correct <- sum(tab[c(1,4)])
tot.incorrect <- sum(tab[c(2,3)])
tot <- sum(tab)
perc.correct <- tot.correct/tot*100
error.rate <- 100 - perc.correct

paste0("Correct Classifications: ",tot.correct)
```

```
## [1] "Correct Classifications: 239"
```

```r
paste0("Incorrect Classifications: ",tot.incorrect)
```

```
## [1] "Incorrect Classifications: 64"
```

```r
paste0("Total: ",tot)
```

```
## [1] "Total: 303"
```

```r
paste0("Percent Correct: ",round(perc.correct,2),"%")
```

```
## [1] "Percent Correct: 78.88%"
```

```r
paste0("Error Rate: ", round(error.rate,2),"%")
```

```
## [1] "Error Rate: 21.12%"
```

Is a 79% accuracy rate good? It depends on what we consider to be "chance levels".

One possible way that we could define "chance" is based on how many possible outcomes there are. In this case, there are only two possible outcomes: having a diagnosis or not having a diagnosis. So, we could potentially define chance as 50%. Under this definition of "chance", the model does better than chance. However, as will soon be shown in the next section, there are some important limitation to this comparison.

```r
# Number of participants by diagnosis status
table(phobia$diagnosis)
```

```
##
##  No Diagnosis Yes Diagnosis
##           101           202
```

```r
# Percentages
table(phobia$diagnosis) / sum(table(phobia$diagnosis))
```

```
##
##  No Diagnosis Yes Diagnosis
##     0.3333333     0.6666667
```

The outputs tell us that the data are biased towards people with a diagnosis. In other words, even if we knew nothing about a person's FNE, if we guessed that a person had a social phobia diagnosis, we would be right more times than not (in this case, there are twice as many people with a diagnosis than people without a diagnosis, so we would be right 66% of the time).

These numbers put our accuracy rate of 79% in perspective. Although 79% accuracy is impressive and is higher than 50%, it is not as impressive when we consider that we could correctly guess 66% of participant's diagnosis status just by guessing "Yes" for all participants.

Error Rate: Comparison to the Null Model

Are the chances of having a social phobia diagnosis truly 50-50? Unlikely.

If we wanted to, we could look up clinical data to see how many people in the country/world currently have a social phobia diagnosis. My guess is that the prevalancy rate would be relatively low (1% to 5%, if I had to guess). This means that if we randomly selected a person out of anyone in the world, it would be relatively unlikely that that specific person would have a social phobia diagnosis.

However, the `phobia` data set is a sample, and may not reflect the distribution of social phobia across the entire population. In fact, the researchers who collected these data specifically targetted people with social anxiety disorders. The table below confirms this, and indicates that there were twice as many people in this sample with a diagnosis than people without a diagnosis:

```r
# Number of participants by diagnosis status
table(phobia$diagnosis)
```

```
##
##  No Diagnosis Yes Diagnosis
##           101           202
```

```r
# Percentages
table(phobia$diagnosis) / sum(table(phobia$diagnosis))
```

```
##
##  No Diagnosis Yes Diagnosis
##     0.3333333     0.6666667
```

The outputs tell us that the data are biased towards people with a diagnosis. In other words, even if we knew nothing about a person's FNE, if we guessed that a person had a social phobia diagnosis, we would be right more times than not (in this case, there are twice as many people with a diagnosis than people without a diagnosis, so we would be right 66% of the time).

By comparing our test model (the model with FNE predicted ADIS_IV) to a **null model**, we can account for these baseline biases in the data. In most cases, this comparison will be more informative and useful than comparing the accuracy rate to an arbitrary 50% rate.

A null model is a model without any predictors in it (except for the intercept). The null model predicts a single value for all values of the data. In a linear regression model, the null model would be a flat line of best fit, and the value of y that minimizes sum of squared residuals. In a binomial logistic regression model, the null model would just be based on the baseline number of 0's and 1's in the outcome variable.

We already calculated the null model's accuracy rate in the chunk above. However, we could also get this information by running a logistic regression model without any predictors in it, as demonstrated in the chunk below. Note, when doing this, it is important that you subset the data down so that the null model includes all the same data points that were included in the test model. For example, if there were any data points that were dropped from the analysis in the test model (due to missing data), you will need to manually exclude these data points when coding the null model. In this case, there was one missing data point in FNE, which means that we need to exclude missing data points when creating the null model.

```r
# To subset the data to have not missing data points:
phobia_sub <- phobia[which(!is.na(phobia$FNE) & !is.na(phobia$ADIS_IV)),]

# To create the null model
mod.null = glm(ADIS_IV ~ 1,
              family = binomial,
              data = phobia_sub)

coef(mod.null)
```

```
## (Intercept)
##   0.6931472
```

Note: The ~ 1 in the glm() function above means "predicted by the intercept"

Lets look at the coefficients on the logit, odds ratio, and probability scales.

```r
data.frame("Logit" = coef(mod.null)) %>%
  mutate(Odds = exp(Logit),
         Probability = Odds / (1 + Odds)) %>%
  mutate_all(function(x){round(x, 2)}) %>% t()
```

```
##             (Intercept)
## Logit              0.69
## Odds               2.00
## Probability        0.67
```

Interpreting the Null Model Intercept:

- Logit Scale: The logit of having a diagnosis of "Yes" is 0.69.
- Odds: The odds of having a diagnosis of "Yes" is 2 times more likely than not having a diagnosis. This makes sense, given that there were twice as many people with a diagnosis than not.
- Probability: The probability of having a diagnosis of "Yes" is 67%; this makes sense, given that 2/3rds of the people in the data file had a social phobia diagnosis.

The regression equation for the null model could be written as such (on the odds ratio scale):

$$\widehat{Odds}_{Diagnosis} = 2$$

Notice that there are no X values to input. This is because there were no predictors in the model. The model makes the same prediction for every value in the data.

Just as we used the regression equation from the test model to classify each participant's predicted classification, we could use the null model's equation to classify each participant's predicted classification.

```
phobia$null.class <-
  ifelse(mod.null$coefficients[1] > 0,
         "Predicted Yes",
         "Predicted No")

  # If the intercept of the Null Model is greater than 0 (a positive number on
  # the logit scale),that means there are more 1's than 0's,
  # and so the Null Model predicts that everyone is a 1.
  # If the intercept is less than 0 (a negative number on the logit scale), then
  # the Null Model predicts everyone is a 0.

tab.null <- table(phobia$null.class,phobia$diagnosis)
tab.null
```

```
##
##                  No Diagnosis Yes Diagnosis
##   Predicted Yes           101           202
```

- There were 101 people incorrectly predicted to have a social phobia diagnosis (they actually did have a social phobia diagnosis) (33% of cases).
- There were 202 people correctly predicted to have a social phobia diagnosis (66% of cases).

The code below automates these caluclations.

```
tot.correct.null <-
  ifelse(mod.null$coefficients[1] > 0,
         tab.null[2], # Observed "Yes" is correct predictions if intercept is +
         tab.null[1]) # Observed "No" is correct predictions if intercept is -

tot.incorrect.null <-
  ifelse(mod.null$coefficients[1] > 0,
         tab.null[1], # Observed "No" is incorrect predictions if intercept is +
         tab.null[2]) # Observed "Yes" is incorrect predictions if intercept is -

tot.null <- sum(tab.null[1:2])
perc.correct.null <- (tot.correct.null / tot.null)*100
error.rate.null <- 100 - perc.correct.null

paste0("Total Correct: ",tot.correct.null)
```

```
## [1] "Total Correct: 202"
```

```
paste0("Total Incorrect: ",tot.incorrect.null)
```

```
## [1] "Total Incorrect: 101"
```

```r
paste0("Total: ",tot.null)
```

```
## [1] "Total: 303"
```

```r
paste0("Percent Correct (null): ",round(perc.correct.null,2),"%")
```

```
## [1] "Percent Correct (null): 66.67%"
```

```r
paste0("Percent Incorrect (null): ",round(error.rate.null,2),"%")
```

```
## [1] "Percent Incorrect (null): 33.33%"
```

Although the original accuracy rate for the model with FNE seemed impressive (79%) in comparison to a 50% chance rate, the null model provides important context that we actually could guess 66% of the cases correctly even without knowing any other information about the participants. Hence, by including FNE as a predictor of social phobia diagnosis, we only improved the accuracy rate by 13%.

Error Rate: Chance vs Null Models

So which is the better benchmark for error rate: Chance? Or the "Null Model"?

As always, it depends.

To answer this question, you might think about whether your outcome variable is randomly determined or not, and whether the binary outcomes are equally expected. You might also consider whether you believe the distribution of your binary variable in your sample reflects the true population distribution.

Lets apply this to the analysis we did above. In the phobia data, there are twice as many people with social phobia diagnosis as people without. In this case, I think the researchers intentionally over-sampled people with a social phobia diagnosis. So in this case, I think it would make more sense to compare the error rate to the Null Model error rate (66%), rather than binary chance (50%).

But lets say that we generated a regression model from a separate set of data, and wanted to test it's accuracy in this phobia data set. Perhaps it would make more sense to use 50% as the bench mark, as the original model did not consider baseline differences in THIS sample.

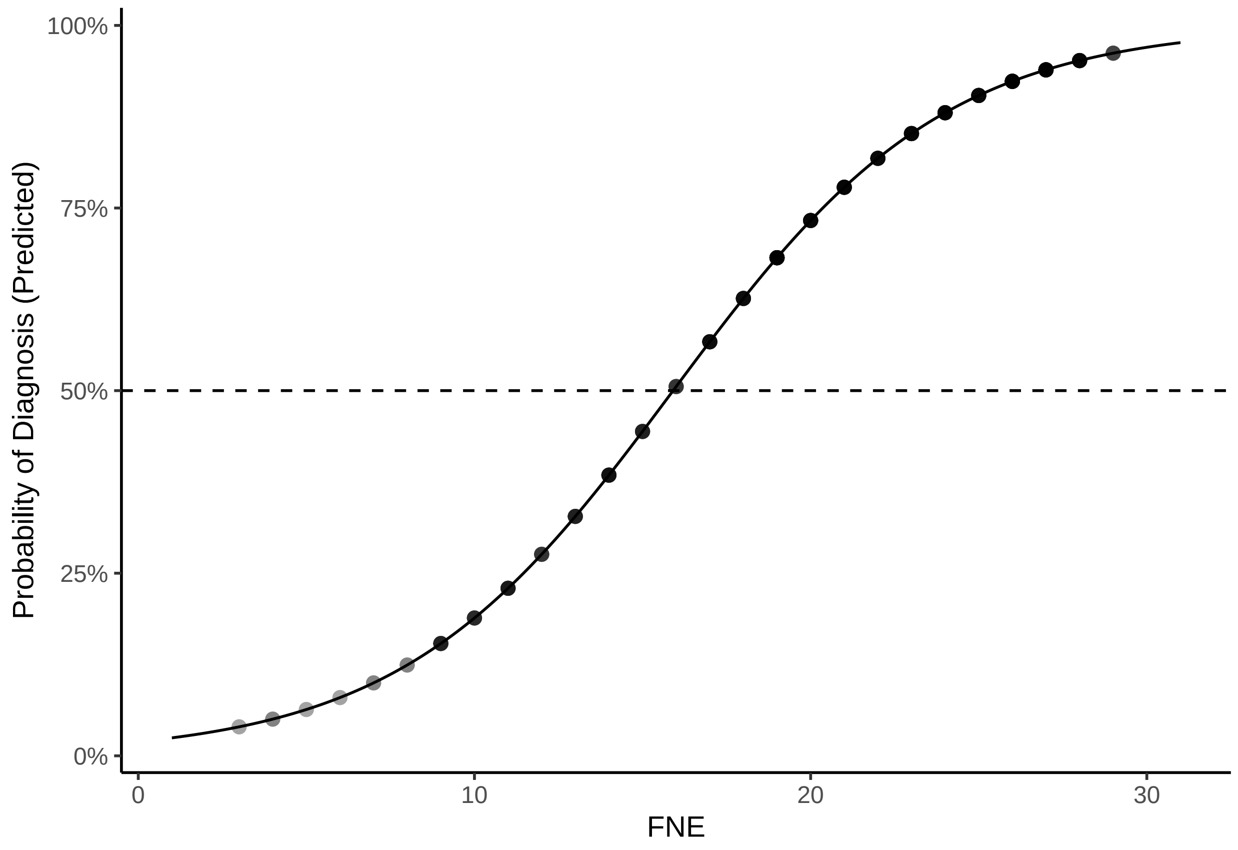
There could be other cases where a 50% benchmark makes more sense. For example, if you had a large sample and you believe your outcome variable is truly a random occurrence (e.g., results of a coin flip), it might make sense to use the 50% benchmark, even if there are more of one outcome in the data file than the other (in this case, we would suspect this to be due to sampling variation).

Graphing a Logistic Regression

Thus far, the graphs above are more for demonstration than presentation.

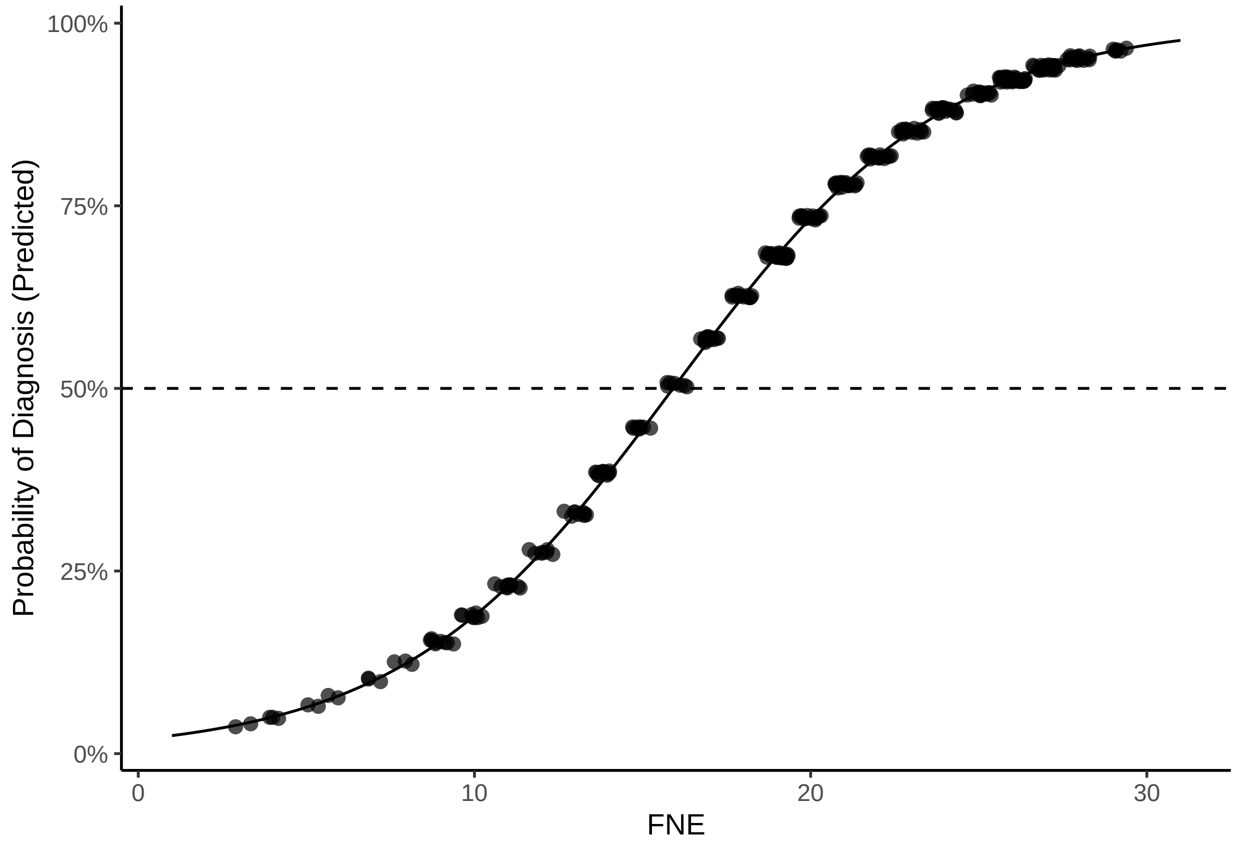
One of the most common way that the results of logistic regressions are presented is with the x variable graphed against the predicted probabilities. The simplest version of this one of the graphs we saw above:

```r
ggplot(data = phobia,
       aes(x = FNE, y = prob)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = prob)) +
  geom_point(alpha = .2, size = 2)  +
  geom_hline(yintercept = .5, linetype = "dashed") +
  labs(y = "Probability of Diagnosis (Predicted)") +
  theme_classic()  +
  scale_y_continuous(breaks = c(0,.25, .5, .75, 1),
                     label = percent)
```



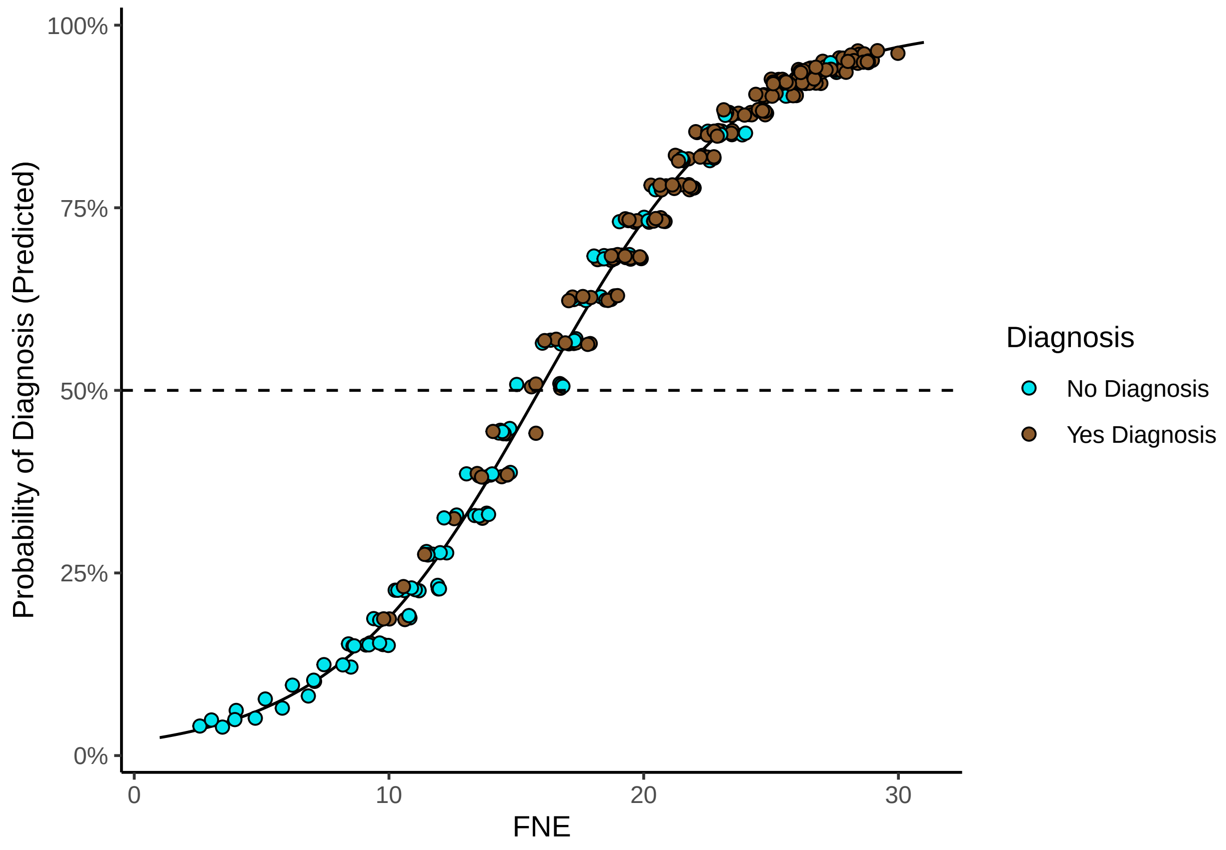
Thus far, we used a lower alpha setting so we could get an idea of how many points were in each spot. Coloring the points does not work so well when you do this (particularly because we have many overlapping points). If you have a lot of overlapping points, there are a few options for cleaning up the graph. For example, you could add a jitter to the graph. You may need to try a few different options for the width (depending on the scale of the x variable).

```r
ggplot(data = phobia,
       aes(x = FNE, y = prob)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = prob)) +
  geom_point(size = 2,
             alpha = .7,
             position = position_jitter(width = .4)) +
  labs(y = "Probability of Diagnosis (Predicted)") +
  theme_classic()   +
  geom_hline(yintercept = .5, linetype = "dashed") +
  scale_y_continuous(breaks = c(0,.25, .5, .75, 1),
                     label = percent)
```



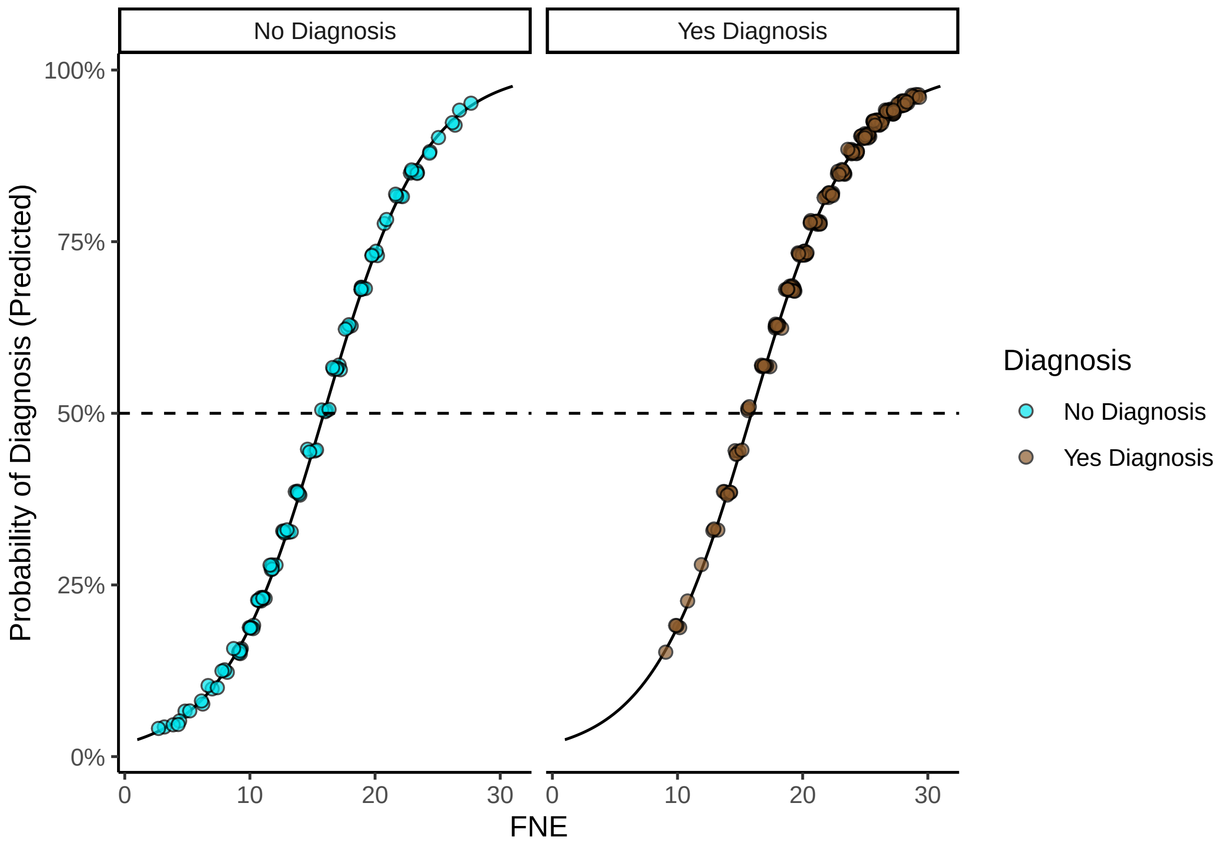
A good graph can also convey information about error rate and model accuracy. One way to do this is to color the points by the outcome variable.

```
ggplot(data = phobia,
       aes(x = FNE, y = prob)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = prob)) +
  geom_point(aes(fill = diagnosis),
             alpha = 1,
             size = 2,
             shape = 21,
             position = position_jitter(width = 1)) +
  scale_fill_manual("Diagnosis", values = c("turquoise2","tan4")) +
  theme_classic()  +
  scale_y_continuous(labels = scales::percent) +
  labs(y = "Probability of Diagnosis (Predicted)") +
  geom_hline(yintercept = .5, linetype = "dashed")
```



Or, you could put “Yes” and “No” in different panels:

```r
ggplot(data = phobia,
       aes(x = FNE, y = prob)) +
  geom_line(data = line.data,
            aes(x = FNE.range, y = prob)) +
  geom_point(aes(fill = diagnosis),
             position = position_jitter(width = .4),
             alpha = .7, size = 2, shape = 21) +
  scale_fill_manual("Diagnosis", values = c("turquoise2","tan4")) +
  theme_classic()  +
  scale_y_continuous(labels = scales::percent) +
  labs(y = "Probability of Diagnosis (Predicted)") +
  geom_hline(yintercept = .5, linetype = "dashed") +
  facet_wrap(~diagnosis)
```



You should play around with changing the jitter width and alpha level (transparency) until you are happy with how the graph looks. Careful: Don't make the jitter too wide! NEVER change the height of the jitter!

If you want a really good example of a paper that used graphs like these effectively, check out Figure 1 in this article here:

https://www.science.org/doi/full/10.1126/scitranslmed.aam9100?casa_token=E7p52FxSx6YAAAAA%3Aj23eOHoj9qLa_q-SMbG8l9-KAOIfWbdXzjo2ju0KvadQ2gLG3y0wDsTurpa99ZkFmfuWD_txVXy0ZKc

Evaluating Model Fit: Deviance Measures

An alternate way we can measure "how good" a model is is with a measure of deviance. Deviance is a statistical summary of model fit, defined for generalized linear models to be analogous to residual standard deviation.

Deviance

Deviance is a measure of error: lower deviance means the model (regression equation) fits the data better.

The more predictors a model has, the lower the deviance is expected to be. If a predictor that is simply random noise is added to a model, we would expect the deviance to decrease by 1 (on average). But, when an informative predictor is added to a model, we would expect deviance to decrease by more than 1.

Hence, when k predictors are added to a model, we expect deviance to decrease by more than k.

For models that do not include random effects (i.e., models that are not multilevel), deviance is equal to:

$$-2 * log(likelihoodfunction)$$

However, you do not need to use this equation to determine the deviance; we can obtain it from the summary() of the glm.

```r
summary(model1)
```

```
##
## Call:
## glm(formula = ADIS_IV ~ FNE, family = "binomial", data = phobia)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.92666    0.54969  -7.143 9.11e-13 ***
## FNE          0.24681    0.02909   8.484  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 385.73  on 302  degrees of freedom
## Residual deviance: 273.46  on 301  degrees of freedom
## AIC: 277.46
##
## Number of Fisher Scoring iterations: 5
```

Notice that this summary output gives the Null deviance (385.73) and the Residual deviance (273.46) (this is the deviance of the model).

We could interpret the deviance as such: By adding *FNE* to the model, deviance decreased by approximately 113 units. Because deviance decreased by more than one unit, this suggests that adding FNE to the model led to a significant reduction in deviance.

Comparing the model's deviance to the null deviance is not very informative when there is only one predictor in the model, as the p value for the predictor's slope can tell you everything you need to know about whether the model was significant. However, deviance comparisons can be more helpful when multiple predictors are involved.

AIC

We can also evaluate AIC, which is also a measure of deviance that takes the model complexity into consideration (i.e., the number of predictors). AIC is a **more conservative** estimate of deviance.

AIC is simply the Deviance plus a penalty on the increasing number of parameters that are estimated in the model (*penalty* is equal to 2p with p = k + 1, where k refers to the number of additional parameters added to the model).

$$AIC = Deviance + 2 * (1 + k)$$

From the summary output above, the model AIC is 277.46 (notice that it is exactly 4 points higher than the deviance; this is because we added 1 additional parameter, *FNE*). However, the model does not directly give us the null AIC.

A simple way to calculate the null AIC is to add 2 to the null deviance. We could also use the AIC() functions, if we have created a null regression object.

```r
# Null Model AIC
AIC(mod.null)
```

```
## [1] 387.7276
```

```r
# Test model AIC
AIC(model1)
```

```
## [1] 277.4631
```

In general, changes in AIC that are greater than 2 units are considered statistically significant. We could interpret the above output as such: The AIC decreased by more than 2 units when FNE was added to the null model. This suggests that adding FNE as a predictor significantly improved the model ($p < 0.05$).

Complex Logistic Regression

Lets explore how we would interpret the results of a more complex logistic regression.

We will predict *ADIS_IV* by *FNE*, *SAD*, and *gender*, and the interaction between *gender* and *SAD*.

```r
model2 <- glm(ADIS_IV ~ FNE + SAD*gender,
              family = binomial,
              data = phobia)

summary(model2)$coefficients
```

```
##                   Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)    -13.5287520 2.05822615 -6.573015 4.930639e-11
## FNE              0.2533547 0.04445647  5.698938 1.205563e-08
## SAD              0.7138366 0.13343174  5.349826 8.803878e-08
## gendermale       7.4816564 1.82213343  4.105987 4.025919e-05
## SAD:gendermale  -0.4435830 0.14833913 -2.990330 2.786761e-03
```

We could write the regression equation as follows. Note, X indicates "the value of" the variable indicated by the subscript. I included parentheses to emphasize the order of operation.

$$\widehat{Logit}_{\texttt{Diagnosis}} = \texttt{-13.53} + (\texttt{0.25}X_{FNE}) + (\texttt{0.71}X_{SAD}) + (\texttt{7.48}X_{male}) - (\texttt{0.44}X_{SAD}X_{male})$$

Remember, the intercept ALWAYS refers to "the predicted outcome when all variables in the model are 0". The trick, then, is to figure out what that means for your model (see below). Also note, now that we are working with a multiple regression, for each predictor, we HAVE to include the caveat "when [all other] predictors are held constant".

Logit scale

Interpretations:

- *Intercept*: For *females* (*gender* = 0) with 0 *FNE* and 0 *SAD*, the predicted logit of a social phobia diagnosis is -13.53.

- Slope of *FNE*: For every one-unit increase in *FNE* the logit of a social phobia diagnoses increases by 0.25, when all other variables are held constant.
- Slope of *SAD*: For females (gender = 0), for every one-unit increase in *SAD* the logit of a social phobia diagnoses increases by 0.71, when FNE is held constant.
- Slope of *gender*: When SAD is 0, the logit of a social phobia diagnoses is 7.48 units higher in *males* than in *females* (the reference group), when FNE is held constant.
- Slope of the *interaction*: For males, for every one-unit increase in *SAD* there is a 0.71 - 0.44 unit increase in the logit of a social phobia diagnoses, when FNE is held constant.

Odds Scale

On the odds scale:

```r
exp(coef(model2))
```

```
##    (Intercept)            FNE            SAD     gendermale SAD:gendermale
##   1.332103e-06   1.288340e+00   2.041810e+00   1.775179e+03   6.417330e-01
```

$$\widehat{Odds}_{Diagnosis} = .000001 * 1.29^{X_{FNE}} * 2.04^{X_{SAD}} * 1775.18^{X_{male}} * 0.64^{X_{SAD}X_{male}}$$

Interpretation:

- *Intercept*: The predicted odds of a "Yes" diagnosis ($ADIS\_IV = 1$) for *females* ($gender = 0$) when *SAD* and *FNE* are equal to 0 is .000001. Because none of the variables are centered, this intercept is not particularly meaningful.
- Slope of *FNE*: For every one-unit increase in the score of *FNE*, the odds of a "Yes" diagnosis increase by a factor of 1.29, when holding the other predictors constant.
- Slope of *SAD*: For females (gender = 0), for every one-unit increase in *SAD*, the odds of a "Yes" diagnosis increases by a factor of 2.04, when holding FNE constant.
- Slope of *gender*: When SAD is 0, for every one-unit increase in *gender* (i.e., going from female, the reference group, to male), the odds of "Yes" diagnosis increase by a factor of 1775.18. So, *males* with 0 SAD are about 1775.18 times more likely to have a "Yes" diagnosis than females with 0 SAD, when holding FNE constant.
- Slope of *interaction*: This coefficient indicates that for each additional unit of *SAD*, the odds of *ADIS_IV* change by an additional factor of 0.64 for *males* (beyond the factor of 2.04 for females), when FNE is held constant.

Alternatively, we could have written the equation on the odds scale as such (by exponentiating each term from the linear logit equation). However, this would not be in fully simplified form.

$$\widehat{Odds}_{Diagnosis} = e^{-13.53} * e^{0.25X_{FNE}} * e^{0.71X_{SAD}} * e^{7.48X_{male}} * e^{-0.44X_{SAD}X_{male}}$$

Probability

The best way to interpret this regression model on the probability scale is to pick specific values.

For example, what is the probability of a female with average *FNE* and average *SAD* having a social phobia diagnosis?

```r
predict(model2,
        list(FNE = mean(phobia$FNE, na.rm = T),
             SAD = mean(phobia$SAD, na.rm = T),
             gender = 'female'),
        type = 'resp')
```

```
##         1
## 0.5430837
```

- A female with average levels of FNE and SAD has a predicted 54% probability of having a social phobia diagnosis.

Error Rate

Lets evaluate the error rate of the more complex model. We can do it the same way we did for the simpler model.

```r
# To calculate the predicted logit for each row in the model
## To mannualy dummy code gender
model2$model$gender.code <-
  ifelse(model2$model$gender == "male",1,0)

## To calculate the logit for each row
model2$model$logit <-
  coef(model2)[1] +
  coef(model2)[2] * model2$model$FNE +
  coef(model2)[3] * model2$model$SAD +
  coef(model2)[4] * model2$model$gender.code +
  coef(model2)[5] * model2$model$SAD * model2$model$gender.code

# To convert the logits to odds
model2$model$odds <- exp(model2$model$logit)

# To convert the odds to probability
model2$model$prob <- model2$model$odds / (1 + model2$model$odds)

# To create a variable for the outcome based on predicted probability:
model2$model$predicted.ADIS_IV <-
  ifelse(model2$model$prob > .5,
         "Predicted Yes",
         "Predicted No")

# To create a a variable for the outcome based on observed (actual) classification:
model2$model$diagnosis <-
  ifelse(model2$model$ADIS_IV == 1,
         "Yes",
         "No")

# To create a 2x2 table for predicted vs actual outcome:
tab2 <- table(model2$model$predicted.ADIS_IV,
              model2$model$diagnosis)

# To view the tabulation
tab2
```

```
##
##                   No Yes
##   Predicted No    78  11
##   Predicted Yes   23 191
```

```
# To calculate the error rate:
perc.correct.2 <- (sum(tab2[c(1,4)]) / sum(tab2))*100
error.rate.2 <- 100 - perc.correct.2

paste0("Percent Correct: ",round(perc.correct.2,2),"%")
```

```
## [1] "Percent Correct: 88.78%"
```

```
paste0("Error Rate: ",round(error.rate.2,2),"%")
```

```
## [1] "Error Rate: 11.22%"
```

Lets look at it all together now.

```
data.frame(
  Parameter = c("Intercept (Odds)",
                "FNE (Odds)",
                "SAD (Odds)",
                "Gender [male] (Odds)",
                "SAD by Gender [male] Interaction",
                "Error Rate",
                "Deviance",
                "AIC"),
  Null = c(exp(mod.null$coefficients), # This only has the intercept
           NA,
           NA,
           NA,
           NA,
           error.rate.null,
           mod.null$deviance,
           mod.null$aic) %>% round(2),
  Model1 = c(exp(model1$coefficients), # This only has the intercept and FNE
             NA,
             NA,
             NA,
             error.rate,
             model1$deviance,
             model1$aic) %>% round(2),
  Model2 = c(exp(model2$coefficients), # This is the full model
             error.rate.2,
             model2$deviance,
             model2$aic) %>% round(2)
           )
```

```
##                           Parameter   Null Model1  Model2
## 1                  Intercept (Odds)   2.00   0.02    0.00
## 2                        FNE (Odds)     NA   1.28    1.29
## 3                        SAD (Odds)     NA     NA    2.04
## 4                Gender [male] (Odds)     NA     NA 1775.18
## 5 SAD by Gender [male] Interaction     NA     NA    0.64
## 6                        Error Rate  33.33  21.12   11.22
## 7                          Deviance 385.73 273.46  151.24
## 8                               AIC 387.73 277.46  161.24
```

```
## If you copy this code, make sure to triple check that all the
## cells are lined up correctly!
```

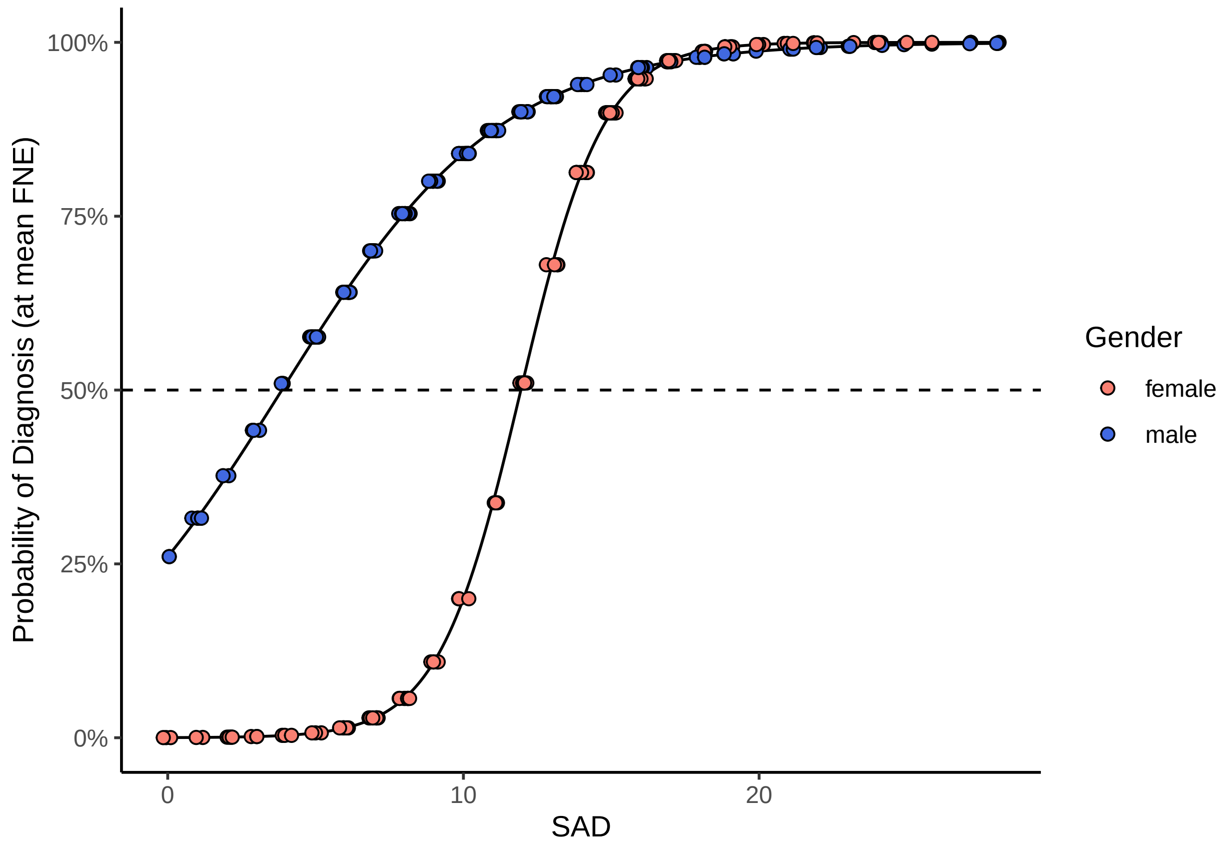
Graphing Complex Models

Because we have 3 different predictors in the model above, it gets a little bit tricky to visualize the data. There isn't an easy way to visualize all three variables (plus the interaction) at the same time. However, based on what we want to visualize (i.e., what effects we want to emphasize), there are a few options we have.

Lets try to visualize the interaction between SAD and Gender. One good way to do this is to graph *SAD* against the probability of a diagnosis, with separate lines and points for each gender. Because we also have *FNE* in the complex model, we will just need to pick a value of *FNE* to use in the calculations, and keep it constant. One common approach people use in these cases is to hold all the other variables at the mean, so we will use the mean of *FNE*.

```
# First, lets start with a fresh data frame, using only the raw
# variables from the model.

graph.data <- model2$model[,1:5]

# Now, lets calculate the predicted probability of a diagnosis using the predict
# function. For the mean of FNE, we need to supply a vector that is the same
# length as all the other inputs (it doesn't recycle by default - very annoying).

graph.data$prob <-
  predict(model2,
          list(FNE = rep(mean(graph.data$FNE, na.rm = T), nrow(graph.data)),
               SAD = graph.data$SAD,
               gender = graph.data$gender),
          type = "resp")

# We can now calculate fit lines for each level of gender:
line.data.2 <-
  data.frame(x.range_SAD =
```

```r
               seq(from = min(model2$model$SAD,na.rm = T),
                   to = max(model2$model$SAD,na.rm = T),
                   by = .01))

line.data.2$prob.M <-
  predict(model2,
          list(FNE = rep(mean(graph.data$FNE, na.rm = T), nrow(line.data.2)),
               SAD = line.data.2$x.range_SAD,
               gender = rep("male", nrow(line.data.2))),
          type = "resp")

line.data.2$prob.F <-
  predict(model2,
          list(FNE = rep(mean(graph.data$FNE, na.rm = T), nrow(line.data.2)),
               SAD = line.data.2$x.range_SAD,
               gender = rep("female", nrow(line.data.2))),
          type = "resp")


# To put it all together
ggplot(data = graph.data,
       aes(x = SAD, y = prob)) +
  geom_line(data = line.data.2,
            aes(x = x.range_SAD, y = prob.F)) +
  geom_line(data = line.data.2,
            aes(x = x.range_SAD, y = prob.M)) +
  geom_point(aes(fill = gender),alpha = 1, size = 2, shape = 21,
             position = position_jitter(width = .2)) +
  theme_classic()  +
  scale_y_continuous(labels = scales::percent) +
  labs(y = "Probability of Diagnosis (at mean FNE)") +
  scale_fill_manual("Gender", values = c("salmon","royalblue")) +
  geom_hline(yintercept = .5, linetype = "dashed")
```



Note on Standardization

Just like in linear regression, we cannot directly compare the strength of the estimates from each parameter in the model unless you standardize the variables. Of course, for logistic regression, you can only standardize the predictors, you cannot standardize the outcome variable (or else it won't be on a 0/1 scale anymore).

Researchers have mixed opinions about whether it is appropriate to standardized variables in a logistic regression or not (probably because you can compare model deviance parameters instead). I have done it both ways, myself.

This section goes through everything from the previous section using standardized predictors instead, and demonstrates that standardizing doesn't change any of the p values, any of the error rates, or any of deviance measures. So, the decision to standardize or not to standardize is ultimately a style choice; whatever makes your results the most clear to the audience is probably the best choice.

```
model1 <- glm(ADIS_IV ~ scale(FNE),
              family = binomial('logit'),
              data = phobia)

model2 <- glm(ADIS_IV ~ scale(FNE) + scale(SAD)*scale(gender.code),
              family = binomial,
              data = phobia)

model3 <- glm(ADIS_IV ~ scale(FNE) + scale(SAD)*scale(gender.code) + scale(SPAI_SP),
              family = binomial,
```

```r
              data = phobia)

model4 <- glm(ADIS_IV ~ scale(FNE) + scale(SAD)*scale(gender.code) + scale(SPAI_SP) + scale(SAS_FNE),
              family = binomial,
              data = phobia)

model5 <- glm(ADIS_IV ~ scale(FNE) + scale(SAD)*scale(gender.code) + scale(SPAI_SP) + scale(SAS_FNE) + s
              family = binomial,
              data = phobia)
```

```r
# Error Rate for Model 1
model1$model$logit <- coef(model1)[1] + coef(model1)[2]*model1$model$`scale(FNE)`
model1$model$predicted.ADIS_IV <- ifelse(model1$model$logit > 0, "Predicted Yes","Predicted No")
tab <- table(model1$model$predicted.ADIS_IV ,model1$model$ADIS_IV)
perc.correct <- sum(tab[c(1,4)]) / sum(tab)*100
error.rate <- 100 - perc.correct


# Error Rate for Model 2
model2$model$logit <-
  coef(model2)[1] +
  coef(model2)[2]*model2$model$`scale(FNE)` +
  coef(model2)[3]*model2$model$`scale(SAD)` +
  coef(model2)[4]*model2$model$`scale(gender.code)` +
  coef(model2)[5]*model2$model$`scale(SAD)`*model2$model$`scale(gender.code)`

model2$model$predicted.ADIS_IV <-
  ifelse(model2$model$logit > 0,"Predicted Yes","Predicted No")

model2$model$diagnosis <- ifelse(model2$model$ADIS_IV == 1, "Yes","No")

tab2 <- table(model2$model$predicted.ADIS_IV,
              model2$model$diagnosis)

perc.correct.2 <- (sum(tab2[c(1,4)]) / sum(tab2))*100
error.rate.2 <- 100 - perc.correct.2


# Error Rate for Model 3
model3$model$logit <-
  coef(model3)[1] +
  coef(model3)[2]*model3$model$`scale(FNE)` +
  coef(model3)[3]*model3$model$`scale(SAD)` +
  coef(model3)[4]*model3$model$`scale(gender.code)`+
  coef(model3)[5]*model3$model$`scale(SPAI_SP)` +
  coef(model3)[6]*model3$model$`scale(SAD)`*model3$model$`scale(gender.code)`

model3$model$predicted.ADIS_IV <-
  ifelse(model3$model$logit > 0,"Predicted Yes","Predicted No")

# Note: Because a logit of 0 is equivalent to 50% probability,
# we can just use the logit for predicted outcome
```

```r
model3$model$diagnosis <- ifelse(model3$model$ADIS_IV == 1, "Yes","No")

tab3 <- table(model3$model$predicted.ADIS_IV,
              model3$model$diagnosis)

perc.correct.3 <- (sum(tab3[c(1,4)]) / sum(tab3))*100
error.rate.3 <- 100 - perc.correct.3


# Error Rate for Model 4
model4$model$logit <-
  coef(model4)[1] +
  coef(model4)[2]*model4$model$`scale(FNE)` +
  coef(model4)[3]*model4$model$`scale(SAD)` +
  coef(model4)[4]*model4$model$`scale(gender.code)`+
  coef(model4)[5]*model4$model$`scale(SPAI_SP)` +
  coef(model4)[6]*model4$model$`scale(SAS_FNE)` +
  coef(model4)[7]*model4$model$`scale(SAD)`*model4$model$`scale(gender.code)`

model4$model$predicted.ADIS_IV <-
  ifelse(model4$model$logit > 0,"Predicted Yes","Predicted No")

model4$model$diagnosis <- ifelse(model4$model$ADIS_IV == 1, "Yes","No")

tab4 <- table(model4$model$predicted.ADIS_IV,
              model4$model$diagnosis)

perc.correct.4 <- (sum(tab4[c(1,4)]) / sum(tab4))*100
error.rate.4 <- 100 - perc.correct.4


# Error Rate for Model 5
model5$model$logit <-
  coef(model5)[1] +
  coef(model5)[2]*model5$model$`scale(FNE)` +
  coef(model5)[3]*model5$model$`scale(SAD)` +
  coef(model5)[4]*model5$model$`scale(gender.code)`+
  coef(model5)[5]*model5$model$`scale(SPAI_SP)` +
  coef(model5)[6]*model5$model$`scale(SAS_FNE)` +
  coef(model5)[7]*model5$model$`scale(SAS_G)` +
  coef(model5)[8]*model5$model$`scale(SAD)`*model5$model$`scale(gender.code)`

model5$model$predicted.ADIS_IV <-
  ifelse(model5$model$logit > 0,"Predicted Yes","Predicted No")

model5$model$diagnosis <- ifelse(model5$model$ADIS_IV == 1, "Yes","No")

tab5 <- table(model5$model$predicted.ADIS_IV,
              model5$model$diagnosis)

perc.correct.5 <- (sum(tab5[c(1,4)]) / sum(tab5))*100
error.rate.5 <- 100 - perc.correct.5
```

```r
results.table <-
  data.frame(Parameter = c("Intercept (Odds)",
                           "FNE (Odds)",
                           "SAD (Odds)",
                           "Gender [male] (Odds)",
                           "SAD by Gender [male] Interaction (Odds)",
                           "SPAI_SP (Odds)",
                           "SAS_FNE (Odds)",
                           "SAS_G (Odds)",
                           "Error Rate",
                           "Deviance",
                           "AIC"),
           Null = round(c(exp(mod.null$coefficients),
                          NA,NA,NA,NA,NA,NA,NA,
                          error.rate.null,
                          mod.null$deviance,
                          mod.null$aic),2),
           Model1 = round(c(exp(model1$coefficients),
                            NA,NA,NA,NA,NA,NA,
                            error.rate,
                            model1$deviance,
                            model1$aic),2),
           Model2 = round(c(exp(model2$coefficients),
                            NA,NA,NA,
                            error.rate.2,
                            model2$deviance,
                            model2$aic),2),
           Model3 = round(c(exp(model3$coefficients),
                            NA,NA,
                            error.rate.3,
                            model3$deviance,
                            model3$aic),2),
           Model4 = round(c(exp(model4$coefficients),
                            NA,
                            error.rate.4,
                            model4$deviance,
                            model4$aic),2),
           Model5 = round(c(exp(model5$coefficients),
                            error.rate.4,
                            model5$deviance,
                            model5$aic),2))

results.table %>% kable()
```

Parameter	Null	Model1	Model2	Model3	Model4	Model5
Intercept (Odds)	2.00	2.58	3.56	6.87	7.16	6.89
FNE (Odds)	NA	4.82	5.03	5.68	5.01	4.64
SAD (Odds)	NA	NA	20.91	9.16	8.61	7.53
Gender [male] (Odds)	NA	NA	2.82	2.61	2.65	2.84
SAD by Gender [male] Interaction (Odds)	NA	NA	0.25	23.15	22.41	19.79
SPAI_SP (Odds)	NA	NA	NA	0.18	1.29	1.06
SAS_FNE (Odds)	NA	NA	NA	NA	0.19	1.86
SAS_G (Odds)	NA	NA	NA	NA	NA	0.19
Error Rate	33.33	21.12	11.22	8.61	6.95	6.95
Deviance	385.73	273.46	151.24	103.55	103.15	100.29
AIC	387.73	277.46	161.24	115.55	117.15	116.29

The interpretation is now: For everyone one SD increase in each predictor, the odds of surviving increase by a factor of [the odds ratio]. The estimates are now comparable between each model, and we could say whether certain estimates are stronger than other.

Notice that standardization does not change the error rate, deviance, or AIC.

Again, I have encountered mixed opinions from researchers about whether predictors should be standardized for logistic regression. Some argue that its not helpful to standardize variables because we usually use the model comparisons to evaluate how good different models are, regardless of the scale that the predictors are on. But others argue that its helpful to standardize your variables, especially if you used the same standardized variables in other analyses (for example, in a linear regression). My opinion is that it doesn't really matter too much. I'm inclined to leave them unstandardized, but I've done it both ways.