Logistic Regression Notes

Lt Col Ken Horton Lt Col Kris Pruitt Professor Bradley Warner

11 December, 2020

Objectives

1) Using `R`, conduct logistic regression and interpret the output, check assumptions, and perform model selection.

2) Write the logistic regression model and predict outputs for given inputs.

3) Use the bootstrap to find confidence intervals for parameter estimates and predictions.

Logistic regression

In this lesson we introduce **logistic regression** as a tool for building models when there is a categorical response variable with two levels. Logistic regression is a type of **generalized linear model** (GLM) for response variables where the assumptions of normally distributed errors is not appropriate. We are prepping you for Math 378, where we will explore predictive models of many different types of response variables including ones that don't assume an underlying functional relationship between inputs and outputs. So cool!

GLMs can be thought of as a two-stage modeling approach. We first model the response variable using a probability distribution, such as the binomial or Poisson distribution. Second, we model the parameter of the distribution using a collection of predictors and a special form of multiple regression.

We will be using the `email` data set for our work in this lesson. These emails were collected from a single email account, and we will work on developing a basic spam filter using these data. The response variable, `spam`, has been encoded to take value 0 when a message is not spam and 1 when it is spam. Our task will be to build an appropriate model that classifies messages as spam or not spam using email characteristics coded as predictor variables. While this model will not be the same as those used in large-scale spam filters, it shares many of the same features.

Email data

In the `email` data set there are many variables available that might be useful for classifying spam. Descriptions of these variables are presented in help menu in `R` under the `openintro` package. The `spam` variable will be the outcome, and the other 10 variables will be the model predictors. While we have limited the predictors used in this section to be categorical variables (where many are represented as indicator variables), numerical predictors may also be used in logistic regression. See the footnote for an additional discussion on this topic.[1]

[1] Recall that if outliers are present in predictor variables, the corresponding observations may be especially influential on the resulting model. This is the motivation for omitting the numerical variables, such as the number of characters and line breaks in emails. These variables exhibited extreme skew. We could resolve this issue by transforming these variables (e.g. using a log-transformation), but we will omit this further investigation for brevity.

variable	description
spam	Specifies whether the message was spam.
to_multiple	An indicator variable for if more than one person was listed in the **To** field of the email.
cc	An indicator for if someone was CCed on the email.
attach	The number of attached emails.
dollar	The number of times a dollar sign or the word dollar appeared in the email.
winner	An indicator for if the word "winner" appeared in the email message.
inherit	The number of times inherit (or an extension, such as inheritance) appeared in the email.
password	The number of times password appeared in the email.
format	Indicates if the email contained special formatting, such as bolding, tables, or links.
re_subj	Indicates whether "Re:" was included at the start of the email subject.
exclaim_subj	Indicates whether any exclamation point was included in the email subject.

Modeling the probability of an event

The outcome variable for a GLM is denoted by Y_i, where the index i is used to represent observation i. In the email application, Y_i will be used to represent whether email i is spam ($Y_i = 1$) or not ($Y_i = 0$).

The predictor variables are represented as follows: $x_{1,i}$ is the value of variable 1 for observation i, $x_{2,i}$ is the value of variable 2 for observation i, and so on.

Logistic regression is a generalized linear model where the outcome is a two-level categorical variable. The outcome, Y_i, takes the value 1 (in our application, this represents a spam message) with probability p_i and the value 0 with probability $1-p_i$. It is the probability p_i that we model in relation to the predictor variables.

The logistic regression model relates the probability an email is spam (p_i) to the predictors $x_{1,i}$, $x_{2,i}$, ..., $x_{k,i}$ through a framework much like that of multiple regression:

$$\text{transformation}(p_i) = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots \beta_k x_{k,i}$$

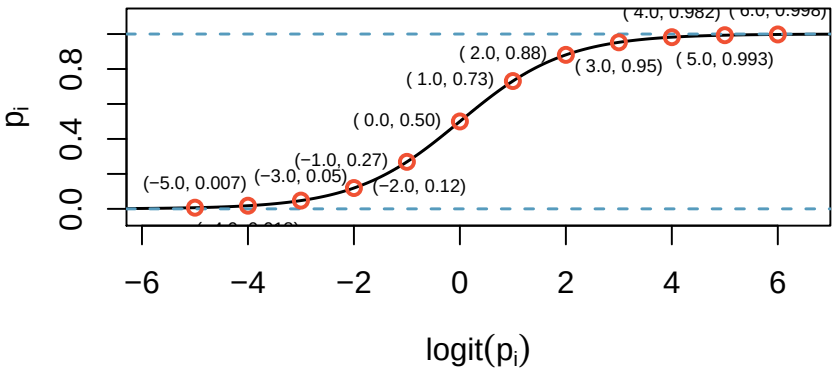
We want to choose a transformation that makes practical and mathematical sense. For example, we want a transformation that makes the range of possibilities on the left hand side of the above equation equal to the range of possibilities for the right hand side. If there was no transformation for this equation, the left hand side could only take values between 0 and 1, but the right hand side could take values outside of this range. A common transformation for p_i is the **logit transformation**, which may be written as

$$\text{logit}(p_i) = \log_e \left(\frac{p_i}{1 - p_i} \right)$$

Below, we expand the equation using the logit transformation of p_i:

$$\log_e \left(\frac{p_i}{1 - p_i} \right) = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \cdots + \beta_k x_{k,i}$$

The logit transformation is shown in the next figure.



Notice the output of the `logit` function restricts the values between 0 and 1. The curve is fairly flat on the edges with a sharp rise in the center. There are other functions that achieve this same result. However, for reasons beyond the scope of this class, the logit function has desirable mathematical properties that relate making sure all the common GLMs fall within the exponential family of distributions. This topic is at the graduate school level and not needed for our studies.

In our spam example, there are 10 predictor variables, so $k = 10$. This model isn't very intuitive, but it still has some resemblance to multiple regression, and we can fit this model using software. In fact, once we look at results from software, it will start to feel like we're back in multiple regression, even if the interpretation of the coefficients is more complex.

First model

Here we create a spam filter with a single predictor: `to_multiple`. This variable indicates whether more than one email address was listed in the **To** field of the email.

Read in the data:

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

Let's take the ten predictors we listed above.

```
email <- email %>%
  select(spam, to_multiple,cc,attach,dollar,winner,inherit,password,format, re_subj,exclaim_subj)
```

Let's convert the character strings to factors.

```
email <- email %>%
  mutate(spam=factor(spam), winner= factor(winner), format= factor(format))
```

```
str(email)
```

```
## tibble [3,921 x 11] (S3: tbl_df/tbl/data.frame)
```

```
##  $ spam        : Factor w/ 2 levels "not spam","spam": 1 1 1 1 1 1 1 1 1 1 ...
##  $ to_multiple : num [1:3921] 0 0 0 0 0 0 1 1 0 0 ...
##  $ cc          : num [1:3921] 0 0 0 0 0 0 0 1 0 0 ...
##  $ attach      : num [1:3921] 0 0 0 0 0 0 0 1 0 0 ...
##  $ dollar      : num [1:3921] 0 0 4 0 0 0 0 0 0 0 ...
##  $ winner      : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ inherit     : num [1:3921] 0 0 1 0 0 0 0 0 0 0 ...
##  $ password    : num [1:3921] 0 0 0 0 2 2 0 0 0 0 ...
##  $ format      : Factor w/ 2 levels "HTML","text": 1 1 1 1 2 2 1 1 2 1 ...
##  $ re_subj     : num [1:3921] 0 0 0 0 0 0 0 0 0 0 ...
##  $ exclaim_subj: num [1:3921] 0 0 0 0 0 0 0 0 0 0 ...
```

Finally, let's summarize the data for the model we want to build:

```
email %>%
  select(spam,to_multiple) %>%
  summary()
```

```
##        spam       to_multiple
##  not spam:3554   Min.   :0.0000
##  spam    : 367   1st Qu.:0.0000
##                  Median :0.0000
##                  Mean   :0.1581
##                  3rd Qu.:0.0000
##                  Max.   :1.0000
```

In R we use the `glm()` function to fit a logistic regression model. It has the same formula format as `lm` but also requires a `family` argument. Since our response is binary, we use `binomial`. If we wanted to use `glm()` for linear regression assuming normally distributed residual, the family argument would be `gaussian`. This implies that multiple linear regression with the assumption of normally distributed errors is a special case of a generalized linear model. In R, the response is a 0/1 variable, we can control the outcome of interest, the 1, by using a logical argument in the formula.

```
email_mod <- glm(spam=="spam"~to_multiple,data=email,family="binomial")
```

Notice in the code chunk that the left hand side of the formula has a logical argument, this gives a 0/1 output with 1 being the value we want to predict.

```
summary(email_mod)
```

```
##
## Call:
## glm(formula = spam == "spam" ~ to_multiple, family = "binomial",
##     data = email)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -0.477  -0.477  -0.477  -0.477   2.809
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.11609    0.05618 -37.665  < 2e-16 ***
```

```
## to_multiple -1.80918    0.29685  -6.095  1.1e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2437.2  on 3920  degrees of freedom
## Residual deviance: 2372.0  on 3919  degrees of freedom
## AIC: 2376
##
## Number of Fisher Scoring iterations: 6
```

The following logistic regression model was found using `R`:

$$\log\left(\frac{p_i}{1-p_i}\right) = -2.12 - 1.81 \times \text{to\_multiple}$$

> **Exercise**:
> If an email is randomly selected and it has just one address in the **To** field, what is the probability it is spam? What if more than one address is listed in the **To** field?

If there is only one email in the **To** field, then `to_multiple` takes value 0 and the right side of the model equation equals -2.12. Solving for p_i: $\frac{e^{-2.12}}{1+e^{-2.12}} = 0.11$. Just as we labeled a fitted value of y_i with a "hat" in single-variable and multiple regression, we will do the same for this probability: $\hat{p}_i = 0.11$.

If there is more than one address listed in the **To** field, then the right side of the model equation is $-2.12 - 1.81 \times 1 = -3.93$, which corresponds to a probability $\hat{p}_i = 0.02$.

Notice that we could examine -2.12 and -3.93 in our logistic graph above to estimate the probability before formally calculating the value.

To convert from values on the regression-scale (e.g. -2.12 and -3.93 in our example), we used the following formula, which is the result of solving for p_i in the regression model:

$$p_i = \frac{e^{\beta_0+\beta_1 x_{1,i}+\cdots+\beta_k x_{k,i}}}{1 \ + \ e^{\beta_0+\beta_1 x_{1,i}+\cdots+\beta_k x_{k,i}}}$$

As with most applied data problems, we substitute the point estimates for the parameters (the β_i) so that we may make use of this formula. In our example, the probabilities were calculated as

$$\frac{e^{-2.12}}{1 \ + \ e^{-2.12}} = 0.11, \frac{e^{-2.12-1.81}}{1 \ + \ e^{-2.12-1.81}} = 0.02$$

While the information about whether the email is addressed to multiple people is a helpful start in classifying email as spam or not, the probabilities of 11% and 2% are not dramatically different, and neither provides very strong evidence about which particular email messages are spam. To get more precise estimates, we'll need to include many more variables in the model.

Model with multiple predictors

> **Exercise**: Fit a logistic regression model with all ten predictors listed above.

Let's build the model and summarize it. As a reminder, we can use `.` on the right-hand side of the formula to tell `R` to include all predictors.

```
email_mod2 <- glm(spam=="spam"~.,data=email,family="binomial")
```

```
summary(email_mod2)
```

```
##
## Call:
## glm(formula = spam == "spam" ~ ., family = "binomial", data = email)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6348  -0.4325  -0.2566  -0.0945   3.8846
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.32260    0.09332 -24.889  < 2e-16 ***
## to_multiple  -2.84097    0.31158  -9.118  < 2e-16 ***
## cc            0.03134    0.01895   1.654 0.098058 .
## attach        0.20351    0.05851   3.478 0.000505 ***
## dollar       -0.07304    0.02306  -3.168 0.001535 **
## winneryes     1.83103    0.33641   5.443 5.24e-08 ***
## inherit       0.32999    0.15223   2.168 0.030184 *
## password     -0.75953    0.29597  -2.566 0.010280 *
## formattext    1.52284    0.12270  12.411  < 2e-16 ***
## re_subj      -3.11857    0.36522  -8.539  < 2e-16 ***
## exclaim_subj  0.24399    0.22502   1.084 0.278221
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2437.2  on 3920  degrees of freedom
## Residual deviance: 1936.2  on 3910  degrees of freedom
## AIC: 1958.2
##
## Number of Fisher Scoring iterations: 7
```

Like multiple regression, the results are presented in a summary table. The structure of this table is almost identical to that of multiple regression; the only notable difference is that the p-values are calculated using the normal distribution rather than the t distribution.

In Math 378 we will learn more about tuning models but in this lesson we will do a simple procedure of just trimming variables from the model using the p-value. Using a method called backwards elimination with a p-value cutoff of 0.05 (start with the full model and trim the predictors with p-values greater than 0.05), we ultimately eliminate the `exclaim_subj` and `cc` predictors. The remainder of this section will rely on this smaller model. The code for this process is below, note there are more efficient ways to do this.

Take out `exclaim_subj` because it has largest p-value.

```
email_mod2 <- glm(spam=="spam"~to_multiple+cc+attach+dollar+winner+
                  inherit+password+format+re_subj,
                  data=email,family="binomial")
summary(email_mod2)
```

```
##
## Call:
## glm(formula = spam == "spam" ~ to_multiple + cc + attach + dollar +
##     winner + inherit + password + format + re_subj, family = "binomial",
##     data = email)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6496  -0.4362  -0.2549  -0.0936   3.8713
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.30467    0.09143 -25.207  < 2e-16 ***
## to_multiple -2.82306    0.31120  -9.071  < 2e-16 ***
## cc           0.03107    0.01891   1.643 0.100381
## attach       0.20265    0.05841   3.470 0.000521 ***
## dollar      -0.06891    0.02234  -3.085 0.002039 **
## winneryes    1.85422    0.33543   5.528 3.24e-08 ***
## inherit      0.33128    0.15049   2.201 0.027708 *
## password    -0.75634    0.29635  -2.552 0.010706 *
## formattext   1.51455    0.12232  12.381  < 2e-16 ***
## re_subj     -3.12436    0.36504  -8.559  < 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: 2437.2  on 3920  degrees of freedom
## Residual deviance: 1937.3  on 3911  degrees of freedom
## AIC: 1957.3
##
## Number of Fisher Scoring iterations: 7
```

Take out `cc` because it has largest p-value.

```
email_mod2 <- glm(spam=="spam"~to_multiple+attach+dollar+winner+
                  inherit+password+format+re_subj,
                  data=email,family="binomial")
summary(email_mod2)
```

```
##
## Call:
## glm(formula = spam == "spam" ~ to_multiple + attach + dollar +
##     winner + inherit + password + format + re_subj, family = "binomial",
##     data = email)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6591  -0.4373  -0.2544  -0.0944   3.8707
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.29909    0.09127 -25.190  < 2e-16 ***
```

```
## to_multiple -2.77682    0.30752  -9.030  < 2e-16 ***
## attach       0.20419    0.05789   3.527  0.00042 ***
## dollar      -0.06970    0.02239  -3.113  0.00185 **
## winneryes    1.86675    0.33652   5.547  2.9e-08 ***
## inherit      0.33614    0.15073   2.230  0.02575 *
## password    -0.76035    0.29680  -2.562  0.01041 *
## formattext   1.51770    0.12226  12.414  < 2e-16 ***
## re_subj     -3.11329    0.36519  -8.525  < 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: 2437.2  on 3920  degrees of freedom
## Residual deviance: 1939.6  on 3912  degrees of freedom
## AIC: 1957.6
##
## Number of Fisher Scoring iterations: 7
```

Now `inherit` is significant but since we are generating multiple p-values we may want to adjust these p-values for the multiple comparison problem.[2] This p-value is close to the 0.05 threshold. We can do this adjustment using the `broom` package.

```
library(broom)
```

```
tidy(email_mod2) %>%
  mutate(p_adjusted=p.adjust(p.value))
```

```
## # A tibble: 9 x 6
##   term        estimate std.error statistic   p.value p_adjusted
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>      <dbl>
## 1 (Intercept)  -2.30      0.0913    -25.2  5.09e-140  4.58e-139
## 2 to_multiple  -2.78      0.308      -9.03 1.72e- 19  1.21e- 18
## 3 attach        0.204     0.0579      3.53 4.20e-  4  1.68e-  3
## 4 dollar       -0.0697    0.0224     -3.11 1.85e-  3  5.56e-  3
## 5 winneryes     1.87      0.337       5.55 2.90e-  8  1.45e-  7
## 6 inherit       0.336     0.151       2.23 2.57e-  2  2.57e-  2
## 7 password     -0.760     0.297      -2.56 1.04e-  2  2.08e-  2
## 8 formattext    1.52      0.122      12.4  2.21e- 35  1.77e- 34
## 9 re_subj      -3.11      0.365      -8.53 1.52e- 17  9.15e- 17
```

Based on this work, none of the adjusted p-value exceed 0.05. we will stop our model selection process at this point and use this reduced model for the remainder of the lesson.

> **Exercise**:
> Examine the summary of the reduced model above, and in particular, examine the `to_multiple` row. Is the point estimate the same as we found before, -1.81, or is it different? Explain why this might be.[3]

[2] For a further discussion on the multiple comparison problem see the linked video.

[3] The new estimate is different: -2.78. This new value represents the estimated coefficient when we are also accounting for other variables in the logistic regression model.

Point estimates will generally change a little – and sometimes a lot – depending on which other variables are included in the model. This is usually due to colinearity in the predictor variables. We previously saw this in the Ebay auction example when we compared the coefficient of `cond` in a single-variable model and the corresponding coefficient in the multiple regression model that used three additional variables.

> *Example*: Spam filters are built to be automated, meaning a piece of software is written to collect information about emails as they arrive, and this information is put in the form of variables. These variables are then put into an algorithm that uses a statistical model, like the one we've fit, to classify the email. Suppose we write software for a spam filter using the reduced model in the `email_mod2` object. If an incoming email has the word "winner" in it, will this raise or lower the model's calculated probability that the incoming email is spam?

The estimated coefficient of `winner` is positive (1.86675). A positive coefficient estimate in logistic regression, just like in multiple regression, corresponds to a positive association between the predictor and response variables when accounting for the other variables in the model. Since the response variable takes value 1 if an email is spam and 0 otherwise, the positive coefficient indicates that the presence of "winner" in an email raises the model probability that the message is spam.

> *Example*: Suppose the same email from the last example was in HTML format, meaning the `format` variable took value 1. Does this characteristic increase or decrease the probability that the email is spam according to the model?

Since HTML corresponds to a value of 1 in the `format` variable and the coefficient of this variable is negative (-1.51770), this would lower the probability estimate returned from the model.

Practical decisions in the email application

The last two examples highlight a key feature of logistic multiple regression. In the spam filter example, some email characteristics will push an email's classification in the direction of spam while other characteristics will push it in the opposite direction.

If we were to implement a spam filter using the model we have fit, then each future email we analyze would fall into one of three categories based on the email's characteristics:

1. The email characteristics generally indicate the email is not spam, and so the resulting probability that the email is spam is quite low, say, under 0.05.
2. The characteristics generally indicate the email is spam, and so the resulting probability that the email is spam is quite large, say, over 0.95.
3. The characteristics roughly balance each other out in terms of evidence for and against the message being classified as spam. Its probability falls in the remaining range, meaning the email cannot be adequately classified as spam or not spam.

If we were managing an email service, we would have to think about what should be done in each of these three instances. In an email application, there are usually just two possibilities: filter the email out from the regular inbox and put it in a "spambox", or let the email go to the regular inbox.

> **Exercise**:
> The first and second scenarios are intuitive. If the evidence strongly suggests a message is not spam, send it to the inbox. If the evidence strongly suggests the message is spam, send it to the spambox. How should we handle emails in the third category?[4]

[4] In this particular application, we should err on the side of sending more mail to the inbox rather than mistakenly putting good messages in the spambox. So, in summary: emails in the first and last categories go to the regular inbox, and those in the second scenario go to the spambox.

Exercise:
Suppose we apply the logistic model we have built as a spam filter and that 100 messages are placed in the spambox over 3 months. If we used the guidelines above for putting messages into the spambox, about how many legitimate (non-spam) messages would you expect to find among the 100 messages?[5]

Almost any classifier will have some error. In the spam filter guidelines above, we have decided that it is okay to allow up to 5% of the messages in the spambox to be real messages. If we wanted to make it a little harder to classify messages as spam, we could use a cutoff of 0.99. This would have two effects. Because it raises the standard for what can be classified as spam, it reduces the number of good emails that are classified as spam. However, it will also fail to correctly classify an increased fraction of spam messages. No matter the complexity and the confidence we might have in our model, these practical considerations are absolutely crucial to making a helpful spam filter. Without them, we could actually do more harm than good by using our statistical model. This tradeoff is similar to the one we found between Type 1 and Type 2 errors.

Diagnostics for the email classifier

There are two key conditions for fitting a logistic regression model:

1. Each predictor x_i is linearly related to $\text{logit}(p_i)$ if all other predictors are held constant. This is similar to our linear fit diagnostic in linear multiple regression.
2. Each outcome Y_i is independent of the other outcomes.

The first condition of the logistic regression model is not easily checked without a fairly sizable amount of data. In our data set we have mostly categorical variables where this check is irrelevant. The second problem we have with our data is that `spam` is a fairly rare outcome, only about 10% of the data has `spam` as the outcome.

Let's first visualize these data by plotting the true classification of the emails against the model's fitted probabilities.

First get the predicted values using `augment()` from the `broom` package. We have to convert the first column from a logic variable into a numeric variable and changed `.fitted` into a predicted probability. If the first assumption is valid we would expect a 45 degree line which is close to what we have. Note we could use the code option `type.predict = "response"` in the `augment()` function to have `R` calculate the probability of success directly.

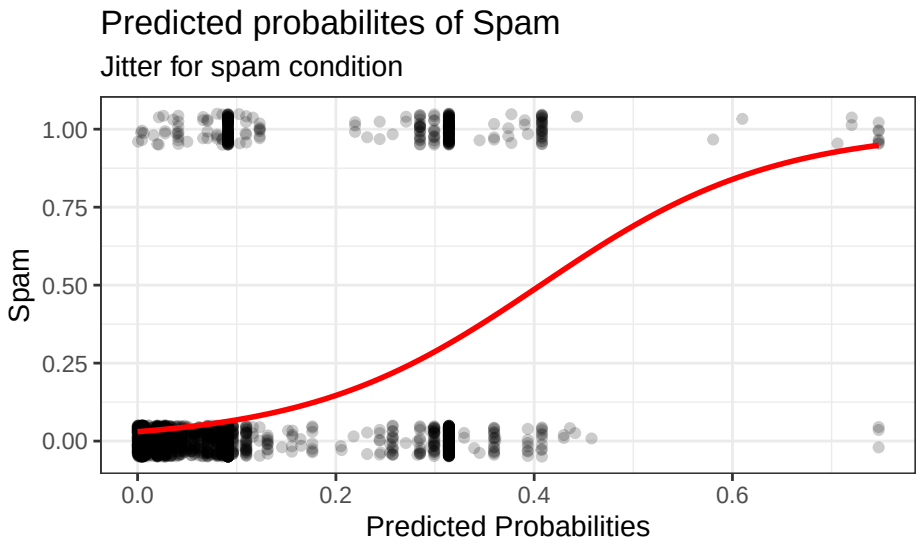
```
mod_data <- augment(email_mod2) %>%
  mutate(p_hat=exp(.fitted)/(1+exp(.fitted)),spam_num=as.numeric(`spam == "spam"`))
```

Now we can plot the data, since there is significant overlap, we will use the `gf_jitter()` geom to move the data point in the vertical direction to create visual separation. We also add the fitted line from using a `glm` model.

```
gf_jitter(mod_data,spam_num~p_hat,width=0,height=0.05,alpha=0.2)  %>%
  gf_theme(theme_bw()) %>%
  gf_labs(title="Predicted probabilites of Spam",
          subtitle="Jitter for spam condition",
```

[5]First, note that we proposed a cutoff for the predicted probability of 0.95 for spam. In a worst case scenario, all the messages in the spambox had the minimum probability equal to about 0.95. Thus, we should expect to find about 5 or fewer legitimate messages among the 100 messages placed in the spambox.

```
          x="Predicted Probabilities",
          y="Spam") %>%
  gf_smooth(method = "glm", se = FALSE,
            color = "red",method.args = list(family = "binomial"))
```



Many of the emails, including those that are truly spam, have fitted probabilities below 0.5. This may at first seem very discouraging: we have fit a logistic model to create a spam filter, but few emails have a fitted probability of being spam above 0.5. Don't despair; we will discuss ways to improve the model in Math 378.

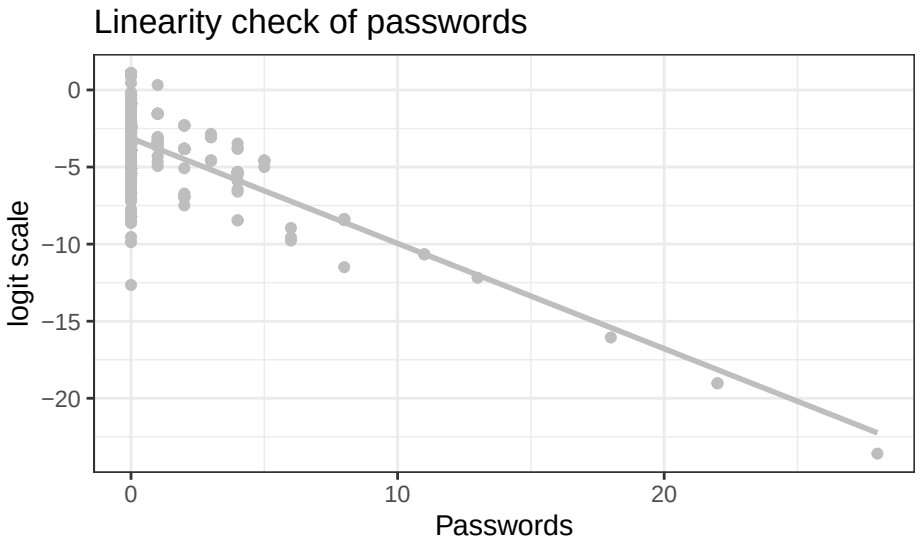
We'd like to assess the quality of our model. For example, we might ask: if we look at emails that we modeled as having a 10% chance of being spam, do we find about 10% of them actually are spam? In Math 378 you will learn about ROC curves and smoothing splines that may help to assess the quality of the fit. That is beyond the scope of this class.

The problem with this data is that essentially all the predictors are binary. Some like `dollar` and `password` are counting variables but most of the values are zero. However, we will use the variable `passwords` to show how we can tell if the variable is linear on the logit scale.

```
email$logit <- log(email_mod2$fitted.values/(1-email_mod2$fitted.values))
```

```
# Graph the logit variable against password
ggplot(data = email, aes(x = password, y = logit))+
  geom_point(color = "gray") +
  geom_smooth(method = "lm", se = FALSE, color = "gray") +
  theme_bw() +
  labs(title="Linearity check of passwords",
            x="Passwords", y="logit scale")
```

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



Again, not enough data for the larger counts to make this plot of value.

We could evaluate the second logistic regression model assumption – independence of the outcomes – using the model residuals. The residuals for a logistic regression model are calculated the same way as with multiple regression: the observed outcome minus the expected outcome. For logistic regression, the expected value of the outcome is the fitted probability for the observation, and the residual may be written as

$$e_i = Y_i - \hat{p}_i$$

We could plot these residuals against a variety of variables or in their order of collection, as we did with the residuals in multiple regression. However, since the model will need to be revised to effectively classify spam and you have already seen similar residual plots in previous lessons on regression, we won't investigate the residuals here.

Confidence intervals

In this section we will generate confidence intervals. This section is experimental since we are not sure how `do()` from the `mosaic` package will work with the `glm()` function, but let's experiment.

Confidence intervals for a parameter

First, let's use the `R` built-in function `confint()` to find the confidence interval for the simple logistic regression model.

```
confint(email_mod)
```

```
## Waiting for profiling to be done...
```

```
##                 2.5 %    97.5 %
## (Intercept) -2.227822 -2.007528
## to_multiple -2.447050 -1.272989
```

Now, let's work with the first model and experiment with `do()`.

```
do(1)*email_mod
```

```
##   Intercept to_multiple .row .index
## 1 -2.116086   -1.809182    1      1
```

```
summary(email_mod)
```

```
##
## Call:
## glm(formula = spam == "spam" ~ to_multiple, family = "binomial",
##     data = email)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -0.477  -0.477  -0.477  -0.477   2.809
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.11609    0.05618 -37.665  < 2e-16 ***
## to_multiple -1.80918    0.29685  -6.095  1.1e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2437.2  on 3920  degrees of freedom
## Residual deviance: 2372.0  on 3919  degrees of freedom
## AIC: 2376
##
## Number of Fisher Scoring iterations: 6
```

It looks like `do()` is performing as expected. Let's now perform one resample to see what happens.

```
do(1)*glm(formula = spam=="spam" ~ to_multiple, family = "binomial", data = resample(email))
```

```
##   Intercept to_multiple .row .index
## 1 -2.040221   -1.842136    1      1
```

Again, it looks like what we expect. Now let's bootstrap the coefficient.

```
set.seed(5011)
results <- do(1000)*glm(formula = spam=="spam" ~ to_multiple,
                        family = "binomial", data = resample(email))
```

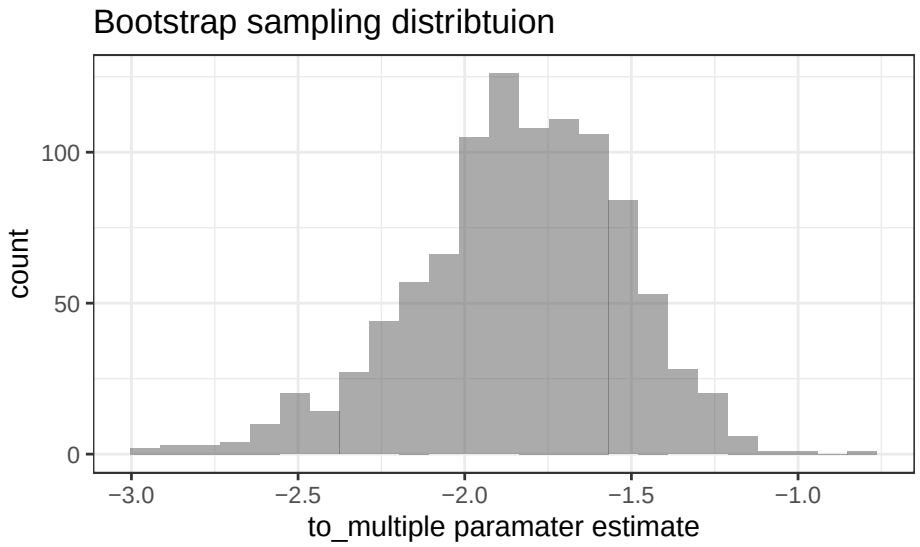
```
head(results)
```

```
##   Intercept to_multiple .row .index
## 1 -2.122210   -1.706431    1      1
## 2 -2.040950   -1.346113    1      2
```

```
## 3 -2.083222   -1.967772    1      3
## 4 -2.047951   -1.384422    1      4
## 5 -2.086667   -1.874146    1      5
## 6 -2.089351   -1.917982    1      6
```

Now we will plot the bootstrap sampling distribution on the slope parameter.

```
results %>%
  gf_histogram(~to_multiple) %>%
  gf_theme(theme_bw()) %>%
  gf_labs(title="Bootstrap sampling distribtuion",
          x="to_multiple paramater estimate")
```



The printout from the logistic regression model assumes normality for the sampling distribution of the `to_multiple` coefficient, but it appears to be negatively skewed, skewed to the left. The 95% confidence interval found using `cdata()`.

```
cdata(~to_multiple,data=results)
```

```
##          lower     upper central.p
## 2.5% -2.518803 -1.277094      0.95
```

Since this does not include the value of zero, we can be 95% confident that it is not zero. This is close to what we found using the `R` function `confint()`.

We can use `results` to get a confidence interval on probability of success if the email has more than one recipient in the **To** line.

```
results_pred <- results %>%
  mutate(pred=exp(Intercept+to_multiple)/(1+exp(Intercept+to_multiple)))
```

```
cdata(~pred,data=results_pred)
```

```
##            lower      upper central.p
## 2.5% 0.009630433 0.03144811      0.95
```

We are 95% confident that expected probability an email with more than one recipient is spam is between 0.96% and 3.1%%.

Bootstrap for multiple logistic regression

As a reminder, the output from our multiple logistic regression model is:

```
summary(email_mod2)
```

```
##
## Call:
## glm(formula = spam == "spam" ~ to_multiple + attach + dollar +
##     winner + inherit + password + format + re_subj, family = "binomial",
##     data = email)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6591  -0.4373  -0.2544  -0.0944   3.8707
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.29909    0.09127 -25.190  < 2e-16 ***
## to_multiple -2.77682    0.30752  -9.030  < 2e-16 ***
## attach       0.20419    0.05789   3.527  0.00042 ***
## dollar      -0.06970    0.02239  -3.113  0.00185 **
## winneryes    1.86675    0.33652   5.547  2.9e-08 ***
## inherit      0.33614    0.15073   2.230  0.02575 *
## password    -0.76035    0.29680  -2.562  0.01041 *
## formattext   1.51770    0.12226  12.414  < 2e-16 ***
## re_subj     -3.11329    0.36519  -8.525  < 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: 2437.2  on 3920  degrees of freedom
## Residual deviance: 1939.6  on 3912  degrees of freedom
## AIC: 1957.6
##
## Number of Fisher Scoring iterations: 7
```

Let's see if the `do()` function works on this model:

```
do(1)*glm(formula = spam=="spam" ~ to_multiple + attach + dollar + winner +
    inherit + password + format + re_subj, family = "binomial",
    data = email)
```

```
##   Intercept to_multiple    attach      dollar winneryes   inherit   password
## 1 -2.299085   -2.776816 0.2041927 -0.06969693  1.866754 0.3361351 -0.7603502
```

```
##   formattext   re_subj .row .index
## 1   1.517701 -3.113292    1      1
```

This looks good. Let's bootstrap the data.

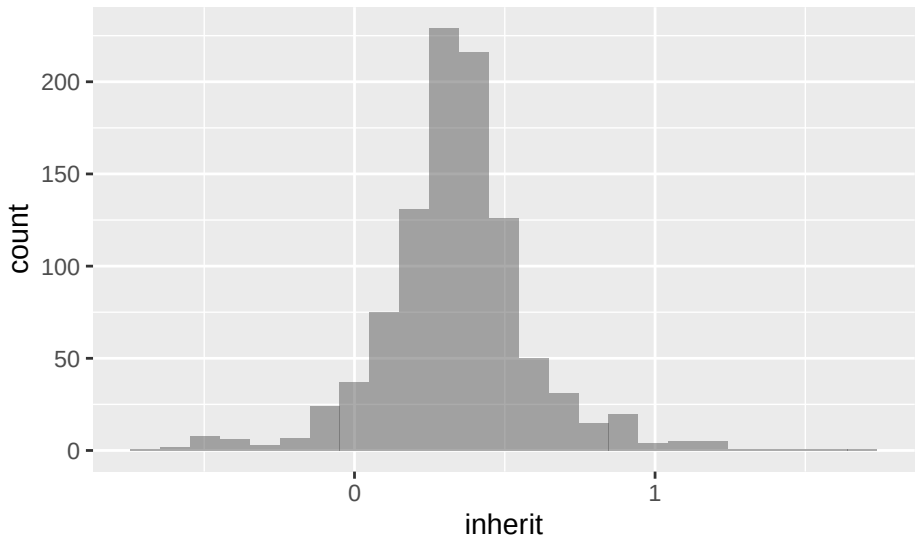
```
set.seed(54321)
results <-do(1000)*glm(formula = spam=="spam" ~ to_multiple + attach + dollar + winner +
    inherit + password + format + re_subj, family = "binomial",
    data = resample(email))
```

```
head(results)
```

```
##   Intercept to_multiple    attach      dollar winneryes    inherit   password
## 1 -2.279571   -2.669680 0.1734052 -0.05262787  1.885284 0.05350223 -0.5952306
## 2 -2.274495   -2.341891 0.1738882 -0.07338149  1.460816 0.41325306 -0.7753607
## 3 -2.265789   -2.545705 0.2662586 -0.07530142  1.581667 0.32499460 -0.6123178
## 4 -2.403802   -2.674211 0.2182831 -0.07203271  1.740070 0.45966567 -0.5936609
## 5 -2.510450   -2.128942 0.1925530 -0.04607213  1.947440 0.25965030 -0.3598016
## 6 -2.359724   -2.601156 0.1914139 -0.06043166  1.714917 0.34210004 -0.4658336
##   formattext   re_subj .row .index
## 1   1.578081 -3.193185    1      1
## 2   1.495599 -2.878605    1      2
## 3   1.412463 -3.155869    1      3
## 4   1.558086 -3.299953    1      4
## 5   1.756101 -3.051380    1      5
## 6   1.453567 -2.616672    1      6
```

In our model `inherit` had the largest p-value, let's build a confidence interval for this parameter.

```
results %>%
  gf_histogram(~inherit)
```



```
cdata(~inherit,data=results)
```

```
##           lower     upper central.p
## 2.5% -0.2119833 0.8939506      0.95
```

In the output, we rejected the null hypothesis that `inherit` was equal to zero, but in the confidence interval we fail to reject. This variable may not be important for the model. In Math 378, we will look at model selection from the perspective of predictive performance.

Improving the set of variables for a spam filter

If we were building a spam filter for an email service that managed many accounts (e.g. Gmail or Hotmail), we would spend much more time thinking about additional variables that could be useful in classifying emails as spam or not. We also would use transformations or other techniques that would help us include strongly skewed numerical variables as predictors.

Take a few minutes to think about additional variables that might be useful in identifying spam. Below is a list of variables we think might be useful:

(1) An indicator variable could be used to represent whether there was prior two-way correspondence with a message's sender. For instance, if you sent a message to john@example.com and then John sent you an email, this variable would take value 1 for the email that John sent. If you had never sent John an email, then the variable would be set to 0.

(2) A second indicator variable could utilize an account's past spam flagging information. The variable could take value 1 if the sender of the message has previously sent messages flagged as spam.

(3) A third indicator variable could flag emails that contain links included in previous spam messages. If such a link is found, then set the variable to 1 for the email. Otherwise, set it to 0.

The variables described above take one of two approaches. Variable (1) is specially designed to capitalize on the fact that spam is rarely sent between individuals that have two-way communication. Variables (2) and (3) are specially designed to flag common spammers or spam messages. While we would have to verify using the data that each of the variables is effective, these seem like promising ideas.

The table below shows a contingency table for spam and also for the new variable described in (1) above. If we look at the 1,090 emails where there was correspondence with the sender in the preceding 30 days, not one of these message was spam. This suggests variable (1) would be very effective at accurately classifying some messages as not spam. With this single variable, we would be able to send about 28% of messages through to the inbox with confidence that almost none are spam.

	prior correspondence		
	no	yes	Total
spam	367	0	367
not spam	2464	1090	3554
Total	2831	1090	3921

The variables described in (2) and (3) would provide an excellent foundation for distinguishing messages coming from known spammers or messages that take a known form of spam. To utilize these variables, we would need to build databases: one holding email addresses of known spammers, and one holding URLs found in known spam messages. Our access to such information is limited, so we cannot implement these two variables in this lesson. However, if we were hired by an email service to build a spam filter, these would be important next steps.

In addition to finding more and better predictors, we would need to create a customized logistic regression model for each email account. This may sound like an intimidating task, but its complexity is not as daunting as it may at first seem. We'll save the details for a course where large data sets and better computer programming play a more central role.

For what is the extremely challenging task of classifying spam messages, we have made a lot of progress. We have seen that simple email variables, such as the format, inclusion of certain words, and other circumstantial characteristics, provide helpful information for spam classification. Many challenges remain, from better understanding logistic regression to carrying out the necessary computer programming, but completing such a task is very nearly within your reach.

File Creation Information

- File creation date: 2020-12-11
- Windows version: Windows 10 x64 (build 18362)
- R version 3.6.3 (2020-02-29)
- `mosaic` package version: 1.7.0
- `tidyverse` package version: 1.3.0