Reference category and interpreting regression coefficients in R.

As published in Benchmarks Research Matters, November 2018
https://benchmarks.it.unt.edu/researchmatters

Jon Starkweather, PhD

Jon Starkweather, PhD
`jonathan.starkweather@unt.edu`
Consultant
Data **S**cience and **A**nalytics

https://www.unt.edu

https://it.unt.edu/research

DSA hosts a number of "Short Courses".
A list of them is available at:
https://it.unt.edu/researchshortcourses

Those interested in learning more about R, or how to use it, can find information here:
`http://bayes.acs.unt.edu:8083/BayesContent/class/Jon/R_SC/`

Reference category and interpreting regression coefficients in R.

The primary purpose of this article is to illustrate the interpretation of categorical variables as predictors and outcome in the context of traditional regression and logistic regression. First, we must understand how R identifies categorical variables. The R language identifies categorical variables as ‘factors’ which can be ‘ordered’ or not. Throughout this article we will be dealing with unordered factors (i.e. strictly discrete categorical variables). The categories of a factor are identified as ‘levels’ of the factor. A ‘reference’ category is so named and identified as a category of comparison for the other categories. In other words, the other categories are *compared to* the reference. By default R uses the alpha-numerically first category as the reference category (e.g. “a” with letters, “0” with numbers). Consider the factor ‘x1’ below, which is created by replicating the first four letters of the alphabet three times. It has four levels: “a”, “b”, “c”, and “d”. The reference category, which was not user-specified, is “a” because it is alphabetically first of the levels.

```
x1 <- factor(rep(letters[1:4], 3))
x1
[1] a b c d a b c d a b c d
Levels: a b c d
summary(x1)
a b c d
3 3 3 3
levels(x1)
[1] "a" "b" "c" "d"
```

We can set a specific reference category by explicitly placing one of the levels first when specifying the levels. We order the levels so whatever level is first will be the one we want as a reference. Below, we use ‘x1’ to create a new factor, ‘x2’ and specify the levels so that “d” is the reference category. It is important to note that the values of ‘x1’ and ‘x2’ are identical, only the reference category is different.

```
x2 <- factor(x1, levels = c("d","a","b","c"))
x2
[1] a b c d a b c d a b c d
Levels: d a b c
summary(x2)
d a b c
3 3 3 3
levels(x2)
[1] "d" "a" "b" "c"
```

Next, we create a numeric outcome variable, ‘y’ such that the values will correlate with the values of ‘x2’. We then demonstrate that by regressing ‘x2’ onto ‘y’.

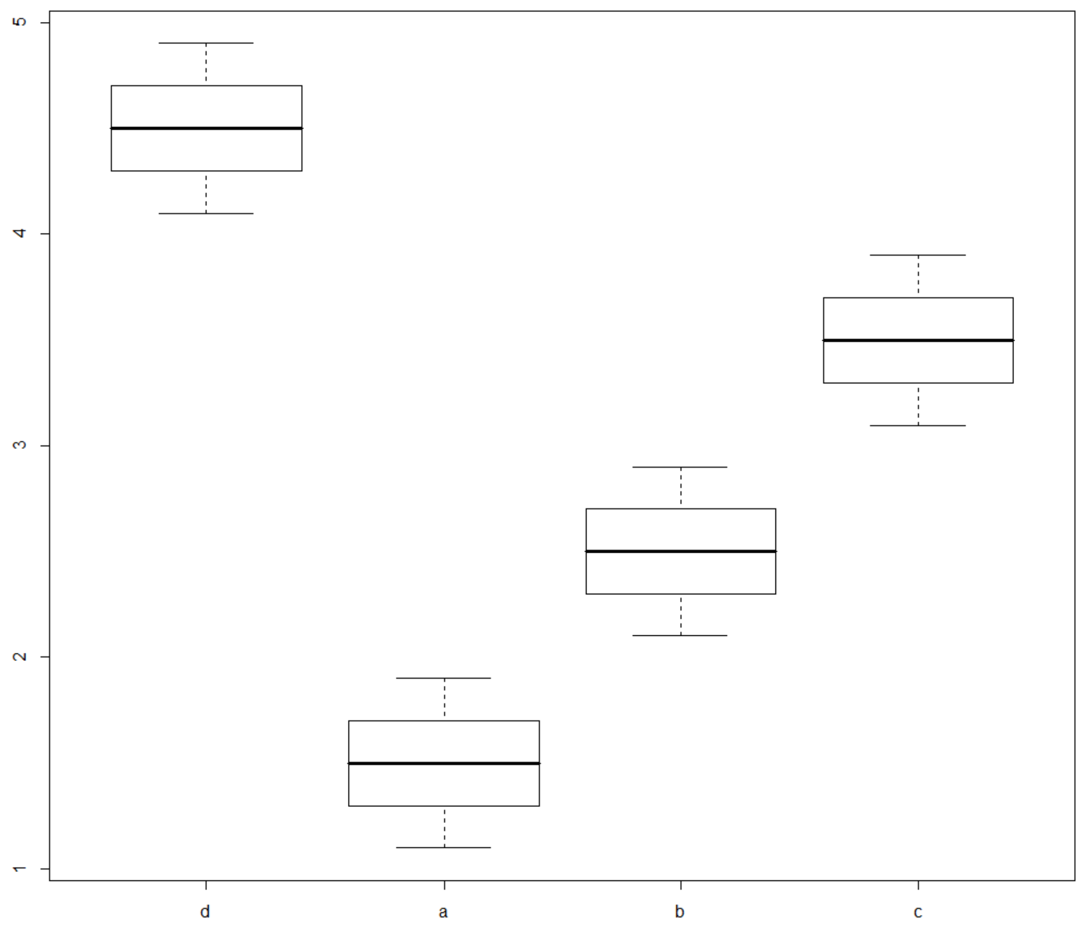
```
y <- c(1.1, 2.1, 3.1, 4.1, 1.5, 2.5, 3.5, 4.5, 1.9, 2.9, 3.9, 4.9)
summary(lm(y ~ x2))
```

```
Call:
lm(formula = y ~ x2)

Residuals:
   Min     1Q Median     3Q    Max 
  -0.4   -0.4    0.0    0.4    0.4 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   4.5000     0.2309  19.486 5.00e-08 ***
x2a          -3.0000     0.3266  -9.186 1.59e-05 ***
x2b          -2.0000     0.3266  -6.124 0.000282 ***
x2c          -1.0000     0.3266  -3.062 0.015539 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 0.4 on 8 degrees of freedom
Multiple R-squared:  0.9214,    Adjusted R-squared:  0.8919 
F-statistic: 31.25 on 3 and 8 DF,  p-value: 9.103e-05
```

So, looking at the 'x2' model, directly above; we see that the mean (y-value) of category, or level, "a" is 3.0 units less than the mean (y-value) of "d" (which is listed as the intercept). Likewise, the mean of "b" is 2.0 units less than the mean of "d" and the mean of "c" is 1.0 units less than the mean of "d" – because "d" is the reference category in the linear regression and the negative coefficients represent the "less than" in the interpretation. The *t*-test is simply testing if the difference between, say the category 'a' coefficient and the reference category, 'd' is different than zero: 4.50 - 1.50 = -3.00; is that absolute difference greater than zero? Yes, $p = 0.0000159$. To see all the coefficients, we can run a no-intercept model or simply plot the two variables.

```
summary(lm(y ~ 0 + x2))

Call:
lm(formula = y ~ 0 + x2)

Residuals:
   Min     1Q Median     3Q    Max 
  -0.4   -0.4    0.0    0.4    0.4 

Coefficients:
    Estimate Std. Error t value Pr(>|t|)    
x2d   4.5000     0.2309  19.486 5.00e-08 ***
x2a   1.5000     0.2309   6.495 0.000189 ***
x2b   2.5000     0.2309  10.825 4.68e-06 ***
x2c   3.5000     0.2309  15.155 3.56e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
```

```
Residual standard error: 0.4 on 8 degrees of freedom
Multiple R-squared:  0.9897,    Adjusted R-squared:  0.9846
F-statistic: 192.2 on 4 and 8 DF,  p-value: 5.58e-08

plot(x2, y)
```



To interpret all four coefficients (listed in the 'no-intercept' model)...we would say that all cases with a value of "d" on 'x2' would be predicted to have a value of 4.5 on 'y' because, that is the average of the "d" cases on 'y'. So the coefficients for this model are the average outcome variable value for the category (or level) of the predictor variable. The *t*-tests are simply testing whether the coefficients are different than zero (here, all four are).

Logistic Regression.

In this article the term logistic regression (Cox, 1958) will be used for binary logistic regression rather than also including multinomial logistic regression. Logistic regression is used to regress categorical and numeric variables onto a binary outcome variable. This is accomplished by transforming the raw outcome values into probability (for one of the two categories), odds or odds ratio, and log odds (literally the ‘log’ of the odds / odds ratio). The log odds of the non-reference category are referred to as the logit; that is what is being predicted. To understand the logistic regression, one must have an idea of the relationships between the probability, odds, and log odds or logit.

```
prob <- c(.001,.01,.05,.1,.15,.2,.25,.3,.35,.4,.45,.5,.55,.6,.65,.7,.75,.8,
odds <- round(prob/(1-prob), 4)
logodds <- round(log(odds), 4)
logit.df <- data.frame(prob,odds,logodds)
rm(prob,odds,logodds)
logit.df
```

```
    prob     odds logodds
1  0.001   0.0010 -6.9078
2  0.010   0.0101 -4.5952
3  0.050   0.0526 -2.9450
4  0.100   0.1111 -2.1973
5  0.150   0.1765 -1.7344
6  0.200   0.2500 -1.3863
7  0.250   0.3333 -1.0987
8  0.300   0.4286 -0.8472
9  0.350   0.5385 -0.6190
10 0.400   0.6667 -0.4054
11 0.450   0.8182 -0.2006
12 0.500   1.0000  0.0000
13 0.550   1.2222  0.2007
14 0.600   1.5000  0.4055
15 0.650   1.8571  0.6190
16 0.700   2.3333  0.8473
17 0.750   3.0000  1.0986
18 0.800   4.0000  1.3863
19 0.850   5.6667  1.7346
20 0.900   9.0000  2.1972
21 0.950  19.0000  2.9444
22 0.990  99.0000  4.5951
23 0.999 999.0000  6.9068
```

As the data frame above illustrates, the probability ranges between zero and one. The odds or odds ratio ranges between zero and 1000. The log odds range between negative infinity and positive infinity. The transformation between log odds and probability is direct and allows us to interpret a logistic regression in terms of the predicted probability *when the model includes only one predictor*. When the model contains more than one predictor variable, the transformation is no longer straight forward because the predicted probability of the non-reference category (and, of course, that of the reference category) is based upon all predictors’ coefficients. Below, a binary outcome variable is created. It is a factor with two levels; “0” and “s” with “0” specified as the reference category.

```
f <- factor(c("0","0","s","s","0","0","s","0","0","0","0","s"),
            levels = c("0","s"))
```

```
f
[1] 0 0 s s 0 0 s 0 0 0 0 s
Levels: 0 s
summary(f)
0 s
8 4
levels(f)
[1] "0" "s"
```

Now, let's look at a binomial generalized linear model (GLM). First, start with the simplest version, an intercept only model.

```
summary(glm(f ˜ 1, family = "binomial"))

Call:
glm(formula = f ˜ 1, family = "binomial")

Deviance Residuals:
    Min       1Q    Median       3Q      Max
-0.9005  -0.9005  -0.9005   1.4823   1.4823

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.6931     0.6124  -1.132    0.258

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 15.276  on 11  degrees of freedom
Residual deviance: 15.276  on 11  degrees of freedom
AIC: 17.276

Number of Fisher Scoring iterations: 4
```

The above model produces one coefficient, the intercept term, which can be interpreted as the log odds of predicting the non-reference category. Here, the reference category is "0" and the non-reference category is "s". We can then take the log odds coefficient and back-transform it into the probability of "s". We have no information for this model, only the outcome variable which has only 4 values of "s" and 8 values of "0".

```
exp(-.6931)/(1+exp(-.6931))
[1] 0.3333438
```

So, just looking at the outcome we see 4 of 12 values are "s" and 4 divided by 12 equals .3333. Next, what happens when we introduce a numeric predictor term?

```
X3 <- y
summary(glm(f ˜ x3, family = "binomial"))
```

```
Call:
glm(formula = f ~ x3, family = "binomial")

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5920  -0.6181  -0.3540   0.7362   1.5796

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -4.9954     2.8916  -1.728   0.0841 .
x3            1.3182     0.8052   1.637   0.1016
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 15.276  on 11  degrees of freedom
Residual deviance: 11.197  on 10  degrees of freedom
AIC: 15.197

Number of Fisher Scoring iterations: 5
```

The intercept coefficient is the log odds of someone with a ‘x3’ value of zero having a ‘f’ value of “s” (i.e. the non-reference category of the outcome variable). We can translate log odds into just odds or even probability (of “s”), both of which are extremely small; but, not the same value.

```
exp(-4.9954)
[1] 0.006769013
exp(-4.9954)/(1+exp(-4.9954))
[1] 0.006723501
```

A typical logistic regression coefficient (i.e. the coefficient for a numeric variable) is the expected amount of change in the logit for each unit change in the predictor. The logit is what is being predicted; it is the log odds of membership in the non-reference category of the outcome variable value (here “s”, rather than “0”). The closer a logistic coefficient is to zero, the less influence it has in predicting the logit. So, for every unit change in ‘x3’, we expect the log odds (or logit) to increase by 1.3182. So, to put the logistic coefficient in context, consider the equation of our model: logit = -4.9954 + (1.3182*x3). If we give ‘x3’ a value, say 5.0 then we get:

```
-4.9954 + (1.3182*5.0)
[1] 1.5956
```

and if we give it 6.0, we get:

```
-4.9954 + (1.3182*6.0)
[1] 2.9138
```

The difference between the two equations above is the value of the coefficient:

```
2.9138 - 1.5956
[1] 1.3182
```

Similar to other forms of regression coefficients, the logistic coefficient is the amount of change in the outcome (i.e. the logit) for every unit change in our predictor ('x3'). So, for every unit of 'x3' we would expect a greater probability of "s" on the outcome. Keep in mind, the reason for using the log odds is to attempt to represent the linear form (i.e. linear equation in the logit). If there is a non-linear relationship between a predictor and the outcome, then the standard logistic model is inappropriate. For example there may be a situation where low values of the predictor increase the log odds of the logit and medium values of the predictor decrease the log odds and high values of the predictor increase the log odds – the popular U-shaped curve.

Next we explore the binomial logistic regression with a single categorical predictor.

```
summary(glm(f ~ x1, family = "binomial"))

Call:
glm(formula = f ~ x1, family = "binomial")

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.48230  -0.00008  -0.00008   0.90052   0.90052

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.957e+01  6.209e+03  -0.003    0.997
x1b         -5.389e-09  8.781e+03   0.000    1.000
x1c          2.026e+01  6.209e+03   0.003    0.997
x1d          2.026e+01  6.209e+03   0.003    0.997

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 15.2763  on 11  degrees of freedom
Residual deviance:  7.6382  on  8  degrees of freedom
AIC: 15.638

Number of Fisher Scoring iterations: 18
```

The first thing to state is that the model above has the same outcome variable ('f') as the previous two models. So, we know we are predicting membership in the non-reference category of 'f' (which is the log odds, odds, or probability of the value "s"). However, the coefficients in this model are a little different. The coefficient for the intercept term is actually the coefficient of the reference category of the 'x1' predictor, "a" = -19.57. That number is the log odds of "s" for cases with "a" on the 'x1' predictor. Again, we can translate that value into odds or probability, as was done previously, both of which are incredibly small (but not the same value):

```
exp(-19.57)
[1] 3.168524e-09
exp(-19.57)/(1+exp(-19.57))
[1] 3.168524e-09
```

The other 3 coefficients (for “b”, “c”, & “d”) are not actually coefficients but, instead they represent the difference between their respective coefficients and the reference category coefficient: “a” = -19.57. Notice that the difference (“Estimate”), standard error, *z*-value, and *p*-value is the same for both “c” and “d”. That may seem strange, but it will be explained in the next paragraph. The *p*-values (and *Z*-test values) are used to determine if these ‘difference scores’ (“Estimate” for “b”, “c”, & “d”) are statistically different than zero. Note, the *p*-value for the reference category (“a”) is testing if that (true) coefficient is different than zero. Naturally, one would now be wondering what the (true) coefficients are for “b”, “c”, and “d”. If we run a no-intercept model, we are provided with all the coefficients (and their associated tests to determine if they are different from zero):

```
summary(glm(f ~ 0 + x1, family = "binomial"))

Call:
glm(formula = f ~ 0 + x1, family = "binomial")

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.48230  -0.00008  -0.00008   0.90052   0.90052

Coefficients:
     Estimate Std. Error z value Pr(>|z|)
x1a  -19.5661  6208.8323  -0.003    0.997
x1b  -19.5661  6208.8323  -0.003    0.997
x1c    0.6931     1.2247   0.566    0.571
x1d    0.6931     1.2247   0.566    0.571

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 16.6355  on 12  degrees of freedom
Residual deviance:  7.6382  on  8  degrees of freedom
AIC: 15.638

Number of Fisher Scoring iterations: 18
```

Most of the output is identical to what was produced with an intercept term; only the 3 rows in the “Coefficients” table for “b”, “c”, and “d” are different. It may seem strange that we have two sets of identical coefficients; but this can happen and there is a perfectly simple explanation for it. You can see why those coefficients are the same (for the two pairs of categories) if you review a cross-tabulation of the two variables ‘f’ and ‘x1’.

```
xtabs( ~ f + x1)
   x1
f    a b c d
  0 3 3 1 1
  s 0 0 2 2
```

Categories “a” and “b” have the same number of “0” and “s”, and categories “c” and “d” have the same number of “0” and “s”. We can also see why “a” and “b” have such large negative coefficients (and corresponding tiny odds and probabilities); because, they have no values of “s”. The *p*-value(s) for “a” and “b” are extremely misleading in this example. First of all, the tiny sample size contributes to large standard error(s); but even duplicating the data 1000 times, we still get a large standard error because there are no cases of “a” and “b” which have a value of “s” on the outcome.

```
df.1 <- data.frame(f, x1)
df.2 <- df.1
for (i in 1:1000){
  df.2 <- rbind(df.2, df.1)
}; rm(i, df.1)
nrow(df.2)
[1] 12012
summary(glm(f ~ 0 + x1, df.2, family = "binomial"))

Call:
glm(formula = f ~ 0 + x1, family = "binomial", data = df.2)

Deviance Residuals:
     Min         1Q     Median         3Q        Max
-1.48230   -0.00008   -0.00008   0.90052   0.90052

Coefficients:
     Estimate Std. Error z value Pr(>|z|)
x1a -19.56607  196.24242   -0.10    0.921
x1b -19.56607  196.24242   -0.10    0.921
x1c   0.69315    0.03871   17.91   <2e-16 ***
x1d   0.69315    0.03871   17.91   <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: 16652.2  on 12012  degrees of freedom
Residual deviance:  7645.8  on 12008  degrees of freedom
AIC: 7653.8

Number of Fisher Scoring iterations: 18
```

Second, even with a large sample the *p*-value might mislead, because logistic regression is all about predicting the logit (log odds, odds, probability of the non-reference category of the outcome variable). So, if the probability of “s” on the outcome is nearly zero (0.000000003168524; see above intercept version of the model)...

```
exp(-19.57)/(1+exp(-19.57))
[1] 3.168524e-09
```

...then the probability of “0” (zero) on the outcome variable for a particular predictor or category (e.g. “a” & “b”) is...virtually 1.0.

```
1 - (exp(-19.57)/(1+exp(-19.57)))
[1] 1
```

The coefficients for “c” and “d” are the log odds of “s” for those two categories of the predictor. We can translate that coefficient to probability using the formula from above...

```
exp(.69315)/(1+exp(.69315))
[1] 0.6666673
```

...or, because of the simplicity of the data, we can simply look again at the cell counts.

```
xtabs( ˜ f + x1)
   x1
f   a b c d
  0 3 3 1 1
  s 0 0 2 2
```

With 2 out of 3 values being “s” (for both “c” and “d”), 2/3rds = 0.6666. So, we might interpret the result as: An observation of “c” has a probability of 0.66 of displaying a value of “s” and a probability of 0.33 of displaying a value of “0” on the outcome variable; because we only have one predictor in this model. This would not be the case with multiple predictor variables.

Next, we introduce a one categorical and one numeric predictor. The interpretation of predictor coefficients does not change with the addition of more than one predictor when staying in the realm of interpretation involving the logit or log odds. However, the predicted probabilities associated with the values of the outcome are no long simply a transformation of the coefficients; which is why the caveat at the end of the previous paragraph is critically important.

```
summary(glm(f ˜ x1 + x3, family = "binomial"))

Call:
glm(formula = f ˜ x1 + x3, family = "binomial")

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.56399  -0.00008  -0.00003   0.46586   1.35897
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   -16.077   9552.278  -0.002    0.999
x1b             3.229  13508.960   0.000    1.000
x1c            28.254   9552.280   0.003    0.998
x1d            31.483   9552.284   0.003    0.997
x3             -3.229      3.274  -0.986    0.324

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 15.2763  on 11  degrees of freedom
Residual deviance:  6.4364  on  7  degrees of freedom
AIC: 16.436

Number of Fisher Scoring iterations: 19
```

To interpret the output above, we would maintain the logit (or log odds) scale of the coefficients. So, the intercept coefficient is the log odds of the logit (i.e. a value of “s” on the outcome ‘f’) when a case has a value of “a” on predictor ‘x1’ – “a” is the reference category for the predictor ‘x1’ and a value of zero on ‘x3’. The coefficient for category “b” on predictor ‘x1’ represents the difference in the logit between cases with a value of “b” and cases with a value of “a” (the reference category). We could say, holding everything else constant; the logit for cases with “b” on predictor ‘x1’ is then: -12.838 = 3.229 + (-16.077). The coefficient for category “c” on predictor ‘x1’ represents the difference in the logit between cases with a value of “c” and cases with a value of “a” (the reference category). We could say, holding everything else constant; the logit for cases with “b” on predictor ‘x1’ is 12.177 = 28.254 + (-16.077). The coefficient for category “d” on predictor ‘x1’ represents the difference in the logit between cases with a value of “d” and cases with a value of “a” (the reference category). We could say, holding everything else constant; the logit for cases with “b” on predictor ‘x1’ is 15.406 = 31.483 + (-16.077). The coefficient for the (continuous) predictor ‘x3’ is simply the amount of change in the logit with each unit change in ‘x3’, holding everything else constant (i.e. values of ‘x1’ or any other predictors included in the model). Stated another way, the log odds of “s” on the outcome variable ‘f’ would *decrease* by about 3 (-3.229) for every *increase* in the value of ‘x3’ while holding all else constant.

Next, we can retrieve the predicted probabilities of each case (i.e. fitted values) using the following script.

```
glm(f ~ x1 + x3, family = "binomial")$fitted.values
           1            2            3            4            5 
2.985770e-09 2.985770e-09 8.971670e-01 8.971670e-01 8.204912e-10 
           6            7            8            9           10 
8.204912e-10 7.056659e-01 7.056659e-01 2.254714e-10 2.254715e-10 
          11           12 
3.971670e-01 3.971670e-01 
```

The fitted values above are the predicted probabilities of each case having the non-reference category of the outcome variable (i.e. a value of “s” on the variable ‘f’) given the model of ‘x1’ and ‘x3’ predicting ‘f’. You may notice that the fitted values / probabilities are either extremely small (approximately 0.1) or extremely large (approximately 0.9), but they vary among those two extremes. This is because, in this small data set (*n* =12) only four cases have "s" values; two cases have values of "c" and two cases have values of "d" on 'x1' and each of the four cases have different values of 'x3' (see the data, below):

```
df.1 <- data.frame(x1,x3,f)
df.1
   x1  x3 f
1   a 1.1 0
2   b 2.1 0
3   c 3.1 s
4   d 4.1 s
5   a 1.5 0
6   b 2.5 0
7   c 3.5 s
8   d 4.5 0
9   a 1.9 0
10  b 2.9 0
11  c 3.9 0
12  d 4.9 s
```

Lastly, we can use the likelihood ratio test (LRT) to compare models and determine if a variable should stay in the model (i.e. it is important) or not. This is a very rudimentary way of checking variable importance.

```
mod.1 <- glm(f ~ x1 + x3, family = "binomial")
mod.2 <- glm(f ~ x3, family = "binomial")
anova(mod.1, mod.2, test = "LRT")
Analysis of Deviance Table

Model 1: f ~ x1 + x3
Model 2: f ~ x3
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1         7     6.4364
2        10    11.1972 -3  -4.7608   0.1902
```

The LRT is NON-significant, so we could leave out variable 'x1'; meaning there is no significant difference (in model fit) between the two models (with and without the 'x1' variable).

A version of the R script used in this article can be found on the Data Science and Analytics (DSA) *Do-It-Yourself Introduction to R* website[1]. Data Science and Analytics (DSA)[2] and High-Performance Computing (HPC)[3] services are available to help facilitate quality research.

[1]http://bayes.acs.unt.edu:8083/BayesContent/class/Jon/R_SC/
[2]https://it.unt.edu/research
[3]https://hpc.unt.edu/

If you found these materials useful, please participate in the DSA Client Feedback Survey.
`https://unt.az1.qualtrics.com/SE/?SID=SV_diLLVU9iuJZN8C9`

References & Resources

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society, Series B (Methodological), 20(2)*, 215 - 242.

Gelman, A., & Hill, J. (2007). *Data analysis using regression and multilevel / hierarchical models*. New York: Cambridge University Press.

Harrell, F. E. (2015). *Regression modeling strategies: With applications to linear models, logistic and ordinal regression, and survival analysis* (2nd ed.). Switzerland: Springer International Publishing.

Liao, T. F. (1994). *Interpreting probability models: Logit, probit, and other generalized linear models* (Sage University Paper series on Quantitative Applications in the Social Sciences, series no. 07-101). Thousand Oaks, CA: Sage.

Pedhazur, E. J. (1997). *Multiple regression in behavioral research: Explanation and prediction* (3rd ed.). New York: Wadsworth Thomson Learning, Inc.

This article was last updated on November 2, 2018.

This document was created using LaTeX