Homework 1 Initial Code + Follow-up Code

Julian McClellan

March 25, 2017

```r
###################################################################
###############   Sports Analytics Problem Set 1    ###############
###############            Spring 2017              ###############
###################################################################

# This R code will estimate the logit models for problem set 1.
# You should only need to change one line of code: you must set the working
# directory to where you saved the field goal data.

####################################
#####  Import Field Goal Data  #####
####################################

# Set the working directory to where you saved the homework folder from chalk
setwd("~/playground/sports_hw1")

# Use this command to load the Field Goal data into R.  The data will be stored
# in what R refers to as a data frame.  Here I've titled that data frame df_raw
df_raw <- read.csv("NFL FIeld Goals 2000-2011.csv")

####################################################
#####  Question 2 (a): Duplicate Clark et al.  #####
####################################################

# Duplicate the logistic regression results_make from the Clark et al. paper.
# The results_make will be stored in the logit_clark object.
# The syntax is as follows: MAKE is the response (dependent variable), the other
# variables DIST, GRASS, etc. are the independent (explanatory) variables,
# family indicates you would like the model to be estimated using a binomial
# logit model, and data indicates which data set/ data frame you would like to
# use to estimate the model.
logit_clark <- glm(MAKE ~ DIST + GRASS + COLD49 + WINDY + ALTITUDE + PRECIP,
               family = "binomial", data = df_raw)
summary.glm(logit_clark)
```

```
##
## Call:
## glm(formula = MAKE ~ DIST + GRASS + COLD49 + WINDY + ALTITUDE +
##     PRECIP, family = "binomial", data = df_raw)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7605   0.2518   0.4182   0.6815   1.7291
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.911882   0.136172  43.415  < 2e-16 ***
```

```
## DIST        -0.106203   0.002995 -35.455  < 2e-16 ***
## GRASS       -0.299270   0.053206  -5.625 1.86e-08 ***
## COLD49      -0.341281   0.060586  -5.633 1.77e-08 ***
## WINDY       -0.140134   0.054950  -2.550  0.01077 *
## ALTITUDE     0.694422   0.156609   4.434 9.25e-06 ***
## PRECIP      -0.280358   0.098720  -2.840  0.00451 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11636.8  on 11895  degrees of freedom
## Residual deviance:  9985.6  on 11889  degrees of freedom
## AIC: 9999.6
##
## Number of Fisher Scoring iterations: 5
```

```r
logLik(logit_clark)
```

```
## 'log Lik.' -4992.814 (df=7)
```

```r
####################################################################
#####  Question 2 (a): Additional Logit Model and Lift Curves   #####
####################################################################

# Create new distance variables
# You don't necessarily need to make new ones, you could also just enter in
# DIST ^ 2 and DIST ^ 3 in the logit_clark_more formula.
df_raw$DIST2 <- df_raw$DIST ^ 2
df_raw$DIST3 <- df_raw$DIST ^ 3

# Create kicker experience variables
# Also don't have to make new ones here but we do anyway.
df_raw$KICKER.EXP <- df_raw$FG.OF.CAREER
df_raw$KICKER.EXP2 <- df_raw$KICKER.EXP ^ 2
df_raw$KICKER.EXP3 <- df_raw$KICKER.EXP ^ 3

# Estimate logit model with additional distance and season variables
logit_clark_more <- glm(MAKE ~ factor(SEASON) + DIST + DIST2 + DIST3 + KICKER.EXP +
                 KICKER.EXP2 + KICKER.EXP3 +GRASS + COLD49 + WINDY + ALTITUDE +
                 PRECIP, family = "binomial", data = df_raw)
summary.glm(logit_clark_more)
```

```
##
## Call:
## glm(formula = MAKE ~ factor(SEASON) + DIST + DIST2 + DIST3 +
##     KICKER.EXP + KICKER.EXP2 + KICKER.EXP3 + GRASS + COLD49 +
##     WINDY + ALTITUDE + PRECIP, family = "binomial", data = df_raw)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1231   0.1856   0.4348   0.6957   2.0017
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)         1.468e+01  1.741e+00   8.432  < 2e-16 ***
## factor(SEASON)2001 -7.424e-02  1.182e-01  -0.628 0.530093
## factor(SEASON)2002 -1.048e-01  1.188e-01  -0.882 0.377856
## factor(SEASON)2003  3.121e-02  1.208e-01   0.258 0.796135
## factor(SEASON)2004  7.696e-02  1.260e-01   0.611 0.541457
## factor(SEASON)2005  9.804e-02  1.227e-01   0.799 0.424468
## factor(SEASON)2006  1.628e-01  1.244e-01   1.309 0.190568
## factor(SEASON)2007  2.443e-01  1.254e-01   1.948 0.051414 .
## factor(SEASON)2008  4.246e-01  1.269e-01   3.346 0.000821 ***
## factor(SEASON)2009  9.798e-02  1.246e-01   0.787 0.431551
## factor(SEASON)2010  2.595e-01  1.253e-01   2.071 0.038343 *
## factor(SEASON)2011  3.419e-01  1.252e-01   2.730 0.006330 **
## DIST               -8.272e-01  1.370e-01  -6.039 1.55e-09 ***
## DIST2               1.803e-02  3.501e-03   5.150 2.61e-07 ***
## DIST3              -1.452e-04  2.910e-05  -4.991 6.00e-07 ***
## KICKER.EXP          3.571e-03  1.005e-03   3.554 0.000379 ***
## KICKER.EXP2        -9.038e-06  3.869e-06  -2.336 0.019492 *
## KICKER.EXP3         6.817e-09  3.980e-09   1.713 0.086709 .
## GRASS              -2.689e-01  5.365e-02  -5.012 5.40e-07 ***
## COLD49             -3.297e-01  6.102e-02  -5.403 6.54e-08 ***
## WINDY              -1.359e-01  5.546e-02  -2.450 0.014291 *
## ALTITUDE            6.695e-01  1.593e-01   4.202 2.65e-05 ***
## PRECIP             -2.553e-01  9.973e-02  -2.559 0.010488 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 11636.8  on 11895  degrees of freedom
## Residual deviance:  9875.4  on 11873  degrees of freedom
## AIC: 9921.4
##
## Number of Fisher Scoring iterations: 6
```

```r
logLik(logit_clark_more)
```

```
## 'log Lik.' -4937.718 (df=23)
```

```r
#####################################################################
#####  Question 2 (a): Plot Some lift Curves  #######################
#####################################################################
# The code below creates lift curves of perfect information, no information,
# and of the two logistic models that were created above.
library(caret) # You will need to install the 'caret' package first
```

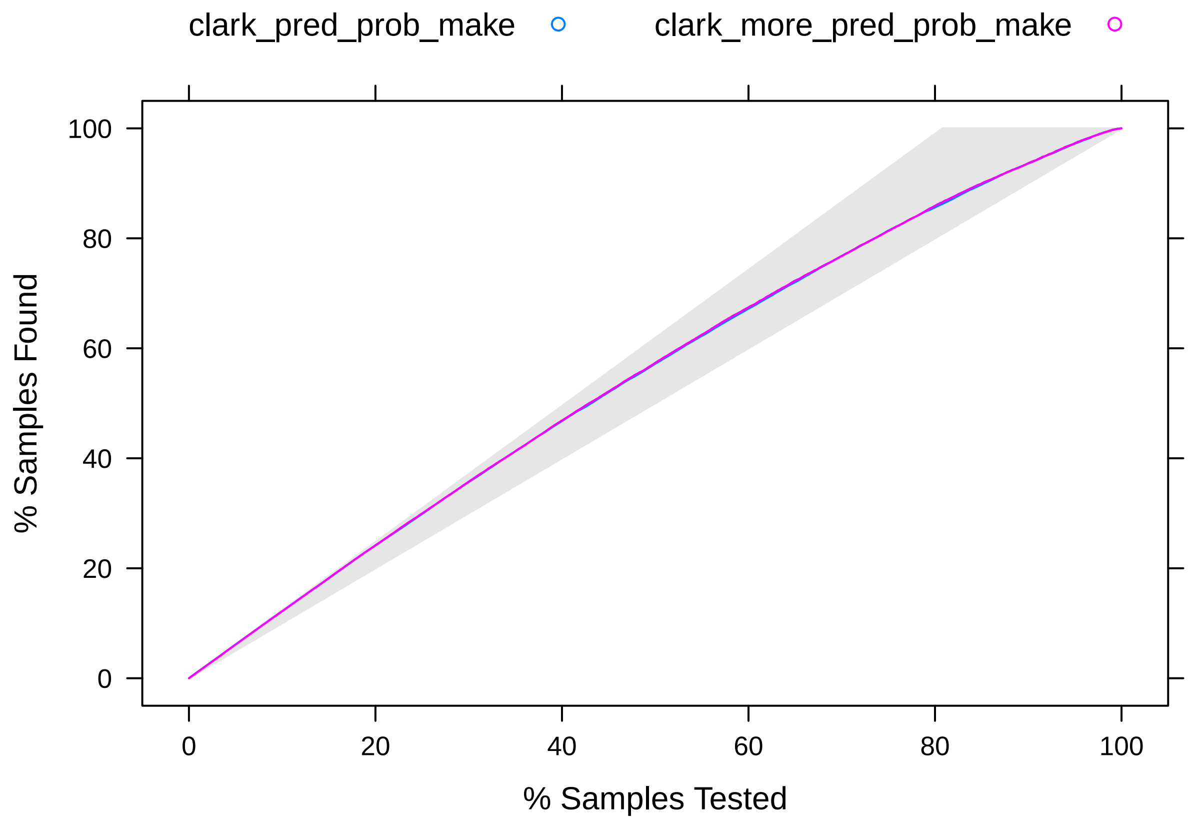
```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
results_make <- data.frame(make = df_raw$MAKE,
                              clark_pred_prob_make = logit_clark$fitted.values,
                      clark_more_pred_prob_make = logit_clark_more$fitted.values)

# Calculating lift info takes a while, be patient
lift_info <- caret::lift(factor(make) ~ clark_pred_prob_make + clark_more_pred_prob_make,
                        data = results_make, plot = 'lift', class = 1)
```

```r
plot(lift_info, auto.key = list(columns = 2))
```




```r
####################################################
###  Question 2 (b): Numerical Summary of Plot.  ###
####################################################
# Here I produce two tables that will be useful in answering 2b
# The information contained within these tables were actually calculated behind
# the scenes to create the lift curve graph above.

library(tidyverse) # You will need to install the 'tidyverse' package
```

```
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr

## Conflicts with tidy packages ----------------------------------------------

## filter(): dplyr, stats
## lag():    dplyr, stats
## lift():   purrr, caret
```

```r
####
# Clark Model
####
clark_plot_dat <- results_make[, c('make', 'clark_pred_prob_make')]

# Sort clark_plot_dat by descending order of probability, with makes coming first
```

```r
clark_plot_dat <- clark_plot_dat %>% arrange(desc(clark_pred_prob_make), desc(make))

# Make perfect information lift, no information lift, and model lift, add them
# to table

# Let's make a function to do these things for us. (We'll use it again for the
# extended model.)
add_plot_info <- function(make_prob_df, track = 'make'){
  no_info_lift <- c()
  perf_info_lift <- c()
  model_lift <- c()
  num_obs <- nrow(make_prob_df)
  num_events <- sum(make_prob_df[[track]])

  for (i in seq(num_obs)){
    no_info_lift <- append(no_info_lift, i / num_obs)
    perf_info_lift <- append(perf_info_lift, i / max(num_events, i))


    model_lift_i <- sum(make_prob_df[1:i,][[track]]) / num_events
    model_lift <- append(model_lift, model_lift_i)
  }

  make_prob_df <-  make_prob_df %>%
    mutate(no_info_lift = no_info_lift,
           perf_info_lift = perf_info_lift,
           model_lift = model_lift)
}

# Call function to add info
clark_plot_dat <- add_plot_info(clark_plot_dat)

# Write the table to a csv file
write.csv(clark_plot_dat, 'clark_standard_model_plotting_table.csv')

# Let's take a peek at what it looks like
head(clark_plot_dat)
```

```
##   make clark_pred_prob_make no_info_lift perf_info_lift   model_lift
## 1    1            0.9864695 8.406187e-05   0.0001040366 0.0001040366
## 2    1            0.9860090 1.681237e-04   0.0002080732 0.0002080732
## 3    1            0.9849762 2.521856e-04   0.0003121099 0.0003121099
## 4    1            0.9849762 3.362475e-04   0.0004161465 0.0004161465
## 5    1            0.9849762 4.203093e-04   0.0005201831 0.0005201831
## 6    1            0.9844657 5.043712e-04   0.0006242197 0.0006242197
```

```r
################################################################################
# Clark Model + Additional
################################################################################
clark_more_plot_dat <- results_make[, c('make', 'clark_more_pred_prob_make')]

# Sort clark_more_plot_dat by descending order of probability, with makes coming first
clark_more_plot_dat <- clark_more_plot_dat %>% arrange(desc(clark_more_pred_prob_make), desc(make))
```

```r
# Make perfect information lift, no information lift, and model lift, add them
# to the table
clark_more_plot_dat <- add_plot_info(clark_more_plot_dat)

# Write the table to a csv file
write.csv(clark_more_plot_dat, 'clark_moreended_model_plotting_table.csv')

# Let's take a peek and what it looks like
head(clark_more_plot_dat)
```

```
##   make clark_more_pred_prob_make no_info_lift perf_info_lift   model_lift
## 1    1                 0.9965430 8.406187e-05   0.0001040366 0.0001040366
## 2    1                 0.9953010 1.681237e-04   0.0002080732 0.0002080732
## 3    1                 0.9952982 2.521856e-04   0.0003121099 0.0003121099
## 4    1                 0.9952946 3.362475e-04   0.0004161465 0.0004161465
## 5    1                 0.9950508 4.203093e-04   0.0005201831 0.0005201831
## 6    1                 0.9947539 5.043712e-04   0.0006242197 0.0006242197
```

```r
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
# Follow-up Section
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------

# Define a function to calculate relative area given the tables that were
# produced
calc_relative_area <- function(plot_info_df){
  model_pseudo_area <- sum(plot_info_df$model_lift - plot_info_df$no_info_lift)
  perf_info_pseudo_area <- sum(plot_info_df$perf_info_lift - plot_info_df$no_info_lift)
  relative_area <- model_pseudo_area / perf_info_pseudo_area
  relative_area
}

# You'll notice that within this function that for the model and perfect
# information we do not exactly calculate the 'area' under the curve, but rather
# the sum of all the differences in height. If we wanted to calculate area we'd
# ahve to multiply each of these heights by the appropriate change in x. However
# since both the model and perfect information have the same changes in x, by
# dividing them we would effectively cancel them out, so we don't bother to
# include them in the first place.

# Question 2b Answer: Relative Area Calculation (Clark Model)

(ra_clark_make <- calc_relative_area(clark_plot_dat))
```

```
## [1] 0.5232546
```

```r
# ------------------------------------------------------------------------------
# Question 2b Answer: Relative Area Calculation (Clark Model - Extended )

(ra_clark_more_make <- calc_relative_area(clark_more_plot_dat))
```

```
## [1] 0.5301066
```

```r
# ------------------------------------------------------------------------------
# Question 2b Extension: RA calculations with "makes" last
```

```r
# When there are multiple observations with the same predicted
# probabilities and different outcomes, sorting the observations merely by the
# predicted probabilities does not uniquely define the Lift Curve. The Lift
# curve also depends on how the makes and misses are sorted within a set of
# observations that have the same predicted probability. If within this set of
# observations the makes occur first then the Lift Curve for success (for
# failure) is maximized (minimized).

# Clark Model
clark_plot_dat_makel <- results_make[, c('make', 'clark_pred_prob_make')]
clark_plot_dat_makel <- clark_plot_dat_makel %>% arrange(desc(clark_pred_prob_make), make)
clark_plot_dat_makel <- add_plot_info(clark_plot_dat_makel)

(ra_clark_makel <- calc_relative_area(clark_plot_dat_makel))
```

```
## [1] 0.5150841
```

```r
# Clark Model - Extended
# The 4 lines below take advantage of 'pipelines' (%>%). The code is written
# more efficiently then that above but does the exact same thing. With pipelines
# we don't have to retype arguments as much and the flow of work being done is
# easy to interpret. Here we say 'ra_calrk_more_makel' is assigned to have the
# value of results_make[] after it is sent through to (%>%) the arrange() function
# and sent through the (%>%) add_plot_info() function, etc.
(ra_clark_more_makel <- results_make[, c('make', 'clark_more_pred_prob_make')] %>%
  arrange(desc(clark_more_pred_prob_make), make) %>%
  add_plot_info() %>%
  calc_relative_area())
```

```
## [1] 0.5301054
```

```r
print(sprintf('Range of Clark Model Relative Area: %.3f , %.3f', ra_clark_makel,
              ra_clark_make))
```

```
## [1] "Range of Clark Model Relative Area: 0.515 , 0.523"
```

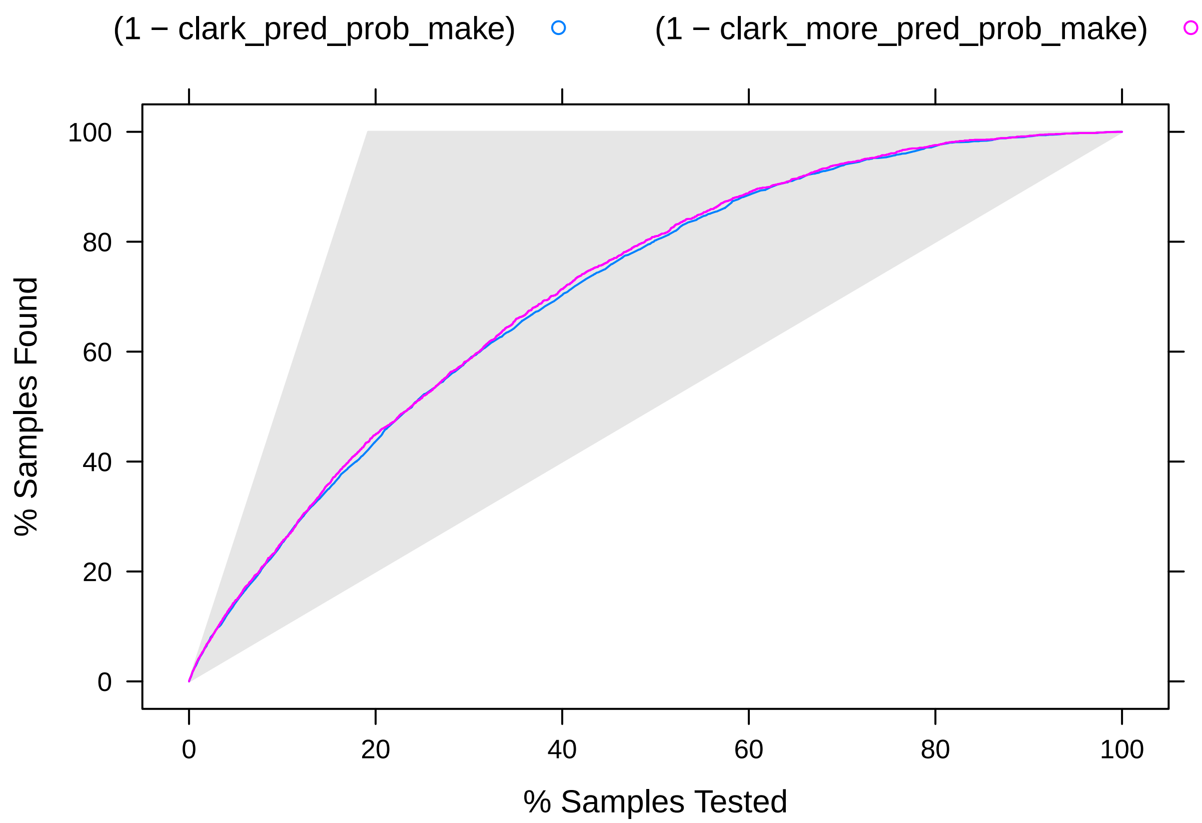
```r
print(sprintf('Range of Clark Model + More Variables Relative Area: %.8f , %.8f',
              ra_clark_more_makel,
              ra_clark_more_make))
```

```
## [1] "Range of Clark Model + More Variables Relative Area: 0.53010539 , 0.53010658"
```

```r
# ---------------------------------------------------------------------------
# Question 2b Extension: Plotting Lift Curves for Failure

# The predicted probability of failure is simply 1 - our predicted probabilities
# of success. We also set class = 0 to tell the function we are event of interest
# is 0 (the misses).
lift_info_miss <- caret::lift(factor(make) ~ (1 - clark_pred_prob_make) +
                           (1 - clark_more_pred_prob_make),
                         data = results_make,
                         plot = 'lift', class = 0)

plot(lift_info_miss, auto.key = list(columns = 2))
```




```
# ------------------------------------------------------------------------------
# Question 2b: Extension: Relative Area for Failure

# Let's make a dataframe 'results_miss' from our previous 'results_make'
results_miss <- results_make %>% transmute(miss = ifelse(make == 0, 1, 0),
                                  clark_pred_prob_miss = 1 - clark_pred_prob_make,
                            clark_more_pred_prob_miss = 1 - clark_more_pred_prob_make)


# Now we use that dataframe to calculate the relsative areas for when misses
# come first on the clark model
(ra_clark_miss <- results_miss[, c('miss', 'clark_pred_prob_miss')] %>%
  arrange(desc(clark_pred_prob_miss), desc(miss)) %>%
  add_plot_info(track = 'miss') %>%
  calc_relative_area())
```

```
## [1] 0.5232546
```

```
# And now when misses come last
(ra_clark_missl <- results_miss[, c('miss', 'clark_pred_prob_miss')] %>%
    arrange(desc(clark_pred_prob_miss), miss) %>%
    add_plot_info(track = 'miss') %>%
    calc_relative_area())
```

```
## [1] 0.5150841
```

```
# To be complete we also use that dataframe to calculate the relative areas for
# when misses comes first on the clark extended model
(ra_clark_more_miss <- results_miss[, c('miss', 'clark_more_pred_prob_miss')] %>%
    arrange(desc(clark_more_pred_prob_miss), desc(miss)) %>%
```

```r
    add_plot_info(track = 'miss') %>%
    calc_relative_area())
```

```
## [1] 0.5301066
```

```r
# And now when misses come last
(ra_clark_more_missl <- results_miss[, c('miss', 'clark_more_pred_prob_miss')] %>%
    arrange(desc(clark_more_pred_prob_miss), miss) %>%
    add_plot_info(track = 'miss') %>%
    calc_relative_area())
```

```
## [1] 0.5301054
```

```r
# We see that it doesn't matter whether we calculate relative area for misses
# or makes, the area turns out to be the same.
(ra_clark_make == ra_clark_miss)
```

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

```r
(ra_clark_makel == ra_clark_missl)
```

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

```r
(ra_clark_more_make == ra_clark_more_miss)
```

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

```r
(ra_clark_more_makel == ra_clark_more_missl)
```

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

```r
# -----------------------------------------------------------------------------
# Question 2b (Extension): McFadden R^2 for model fit measure
# We can also use McFadden R^2 to measure the goodness of fit of our logistic
# models. Much like actual R^2 and relative area, this value ranges between 0 and
# 1.

library(pscl) # Install this package to easily get McFadden R ^ 2
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## Classes and Methods for R developed in the

## Political Science Computational Laboratory

## Department of Political Science

## Stanford University

## Simon Jackman

## hurdle and zeroinfl functions by Achim Zeileis
```

```r
(pR2(logit_clark)[4])
```

```
##  McFadden
```

```
## 0.1418934
```

```r
(pR2(logit_clark_more)[4])
```

```
##  McFadden
## 0.1513625
```

```r
# We see that the McFadden R^2 is higher for the clark model with more variables
# -------------------------------------------------------------------------------
# -------------------------------------------------------------------------------
```