Introduction to Regression Using R

Ashley I Naimi

Dec 2022

Contents

1 Introduction to Regression with R

In this section, we'll cover how to implement generalized linear models in R. We will not cover the theory behind GLMs, but rather focus on the functions in the R programming language that will enable us to implement regression models for binary and continuous data.

There are several types of regression models that can be fit to data. Below is a table with different regression modeling strategies, and their corresponding packages or functions in R:

Regression Type	R Package or Function
Generalized Linear Models	GLM (base R)
Multinomial/Ordinal/Polytomous	`VGAM`, `nnet`
Quantile Regression	`quantreg`
Cox PH Regression	`survival`
Accelerated Failure Time	`survival`, `flexsurv`

Here, we will explore the `glm()` function in base R, and how it can be used to estimate exposure-outcome associations in different ways[1].

[1] Specifically, outcome modeling and propensity score modeling.

2 The Data: NHEFS

To explore the use of the `glm` function in R, we'll rely on the NHANES Epidemiologic Follow-Up Survey data.[2] These data will be used to explore the association between quitting smoking and change in weight between 1971 and 1982. Let's start by loading and exploring the data:

[2] Details on these data can be found on the "Causal Inference: What If?" book website: https://www.hsph.harvard.edu/miguel-hernan/causal-inference-book/

```
nhefs <- read_csv(here("data", "nhefs.csv")) %>%
    mutate(wt_delta = as.numeric(wt82_71 >
        0))

#' Quick view of data
dim(nhefs)
```

```
## [1] 1394   12
```

```
names(nhefs)
```

```
##  [1] "seqn"     "qsmk"     "sex"      "age"      "income"   "sbp"
##  [7] "dbp"      "price71"  "tax71"    "race"     "wt82_71"  "wt_delta"
```

```
head(nhefs)
```

```
## # A tibble: 6 x 12
##    seqn  qsmk   sex   age income   sbp   dbp price71 tax71  race wt82_71 wt_de~1
##   <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>   <dbl>   <dbl>
## 1   233     0     0    42     19   175    96    2.18 1.10      1  -10.1         0
## 2   235     0     0    36     18   123    80    2.35 1.36      0    2.60        1
## 3   244     0     1    56     15   115    75    1.57 0.551     1    9.41        1
## 4   245     0     0    68     15   148    78    1.51 0.525     1    4.99        1
## 5   252     0     0    40     18   118    77    2.35 1.36      0    4.99        1
## 6   257     0     1    43     11   141    83    2.21 1.15      1    4.42        1
## # ... with abbreviated variable name 1: wt_delta
```

To start, we'll use these data to fit two regression models: one where the outcome is continuous weight change (`wt82_71`), and one where the outcome is binary weight change (`wt_delta`). Let's start with the continuous outcome, and seek to estimate the association between quitting smoking and weight change, adjusted for several variables.

2.1 Continuous Outcome

The most common approach for deploying a glm would be as follows:

```
#' Here, we start fitting relevant regression models to the data.

model1 <- glm(wt82_71 ~ qsmk + sex + age +
    income + sbp + dbp + price71 + tax71 +
    race, data = nhefs, family = gaussian("identity"))
```

In the above code, we've fit the GLM to our data, and have stored the fit of our model to an object we call `model1`. Let's explore what's in this model using the `list` function:

```
ls(model1)
```

```
##  [1] "aic"              "boundary"          "call"
##  [4] "coefficients"     "contrasts"         "control"
##  [7] "converged"        "data"              "deviance"
## [10] "df.null"          "df.residual"       "effects"
## [13] "family"           "fitted.values"     "formula"
## [16] "iter"             "linear.predictors" "method"
## [19] "model"            "null.deviance"     "offset"
## [22] "prior.weights"    "qr"                "R"
## [25] "rank"             "residuals"         "terms"
## [28] "weights"          "xlevels"           "y"
```

We can extract the information we want from the model fit using a few standard functions. For example, the output one would obtain from other software programs such as SAS or Stata can be printed to the console in R using the summary function:

```
summary(model1)
```

```
##
## Call:
## glm(formula = wt82_71 ~ qsmk + sex + age + income + sbp + dbp +
##     price71 + tax71 + race, family = gaussian("identity"), data = nhefs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -43.480   -3.848   -0.104    4.092   46.275
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.804261   4.305590   0.419   0.6752
## qsmk         2.924806   0.478194   6.116 1.24e-09 ***
## sex          0.006865   0.414709   0.017   0.9868
## age         -0.164181   0.019975  -8.219 4.66e-16 ***
```

```
## income       0.051853   0.080763   0.642   0.5210
## sbp          0.015733   0.015081   1.043   0.2970
## dbp          0.061780   0.024793   2.492   0.0128 *
## price71     -0.681677   3.057149  -0.223   0.8236
## tax71        0.951893   3.231757   0.295   0.7684
## race        -0.470639   0.633021  -0.743   0.4573
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 57.20064)
##
##     Null deviance: 86972  on 1393  degrees of freedom
## Residual deviance: 79166  on 1384  degrees of freedom
## AIC: 9608.9
##
## Number of Fisher Scoring iterations: 2
```

Sometimes, we may be interested in storing this model output as a dataset, or saving particular elements from the summary. There are functions in the `broom` package that allow us to do this easily:

```
library(broom)

tidy(model1)
```

```
## # A tibble: 10 x 5
##    term        estimate std.error statistic  p.value
##    <chr>          <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)  1.80       4.31      0.419  6.75e- 1
##  2 qsmk         2.92       0.478     6.12   1.24e- 9
##  3 sex          0.00686    0.415     0.0166 9.87e- 1
##  4 age         -0.164      0.0200   -8.22   4.66e-16
##  5 income       0.0519     0.0808    0.642  5.21e- 1
##  6 sbp          0.0157     0.0151    1.04   2.97e- 1
##  7 dbp          0.0618     0.0248    2.49   1.28e- 2
##  8 price71     -0.682      3.06     -0.223  8.24e- 1
```

```
##  9 tax71        0.952      3.23      0.295  7.68e- 1
## 10 race        -0.471      0.633    -0.743  4.57e- 1
```

Oftentimes, we need to extract additional information from the GLM fit, such as model residuals, fitted values, or predictions. This is easy in R. For example, we can extract the residuals from the above model:

```
model1_residuals <- tibble(residuals = model1$residuals,
    index = 1:length(model1$residuals))

plot1 <- ggplot(model1_residuals) + geom_point(aes(y = residuals,
    x = index))

ggsave(here("figures", "residual_plot.pdf"))
```

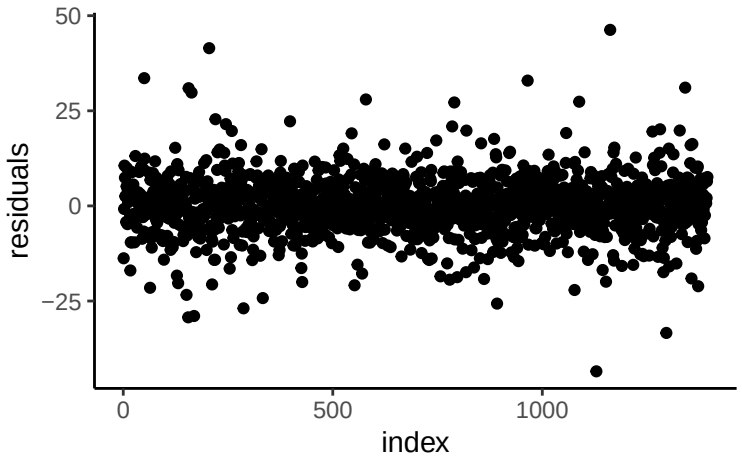

Figure 1: Plot of residuals from model1.

We may also be interested in obtaining predictions from `model1`. There are different types of predictions we may want to obtain. For example, we may want predictions under the observed values of all the variables in the dataset. These are often referred to as fitted values, which we can get using the following:

```
model1_fitted <- tibble(fitted_values = model1$fitted.values,
    index = 1:length(model1$fitted.values))
```

```
head(model1_fitted)
```

```
## # A tibble: 6 x 2
##   fitted_values index
##           <dbl> <int>
## 1          3.67     1
## 2          3.40     2
## 3         -1.18     3
## 4         -2.43     4
## 5          2.48     5
## 6          1.79     6
```

Alternatively, we can use the `predict` function to obtain fitted values under the observed data, or under values other than the observed data[3].

[3] We will see how we can get predictions under different observed data values in the subsequent section.

```
model1_fitted <- tibble(fitted_values = model1$fitted.values,
    fitted_values2 = predict(model1, newdata = nhefs),
    index = 1:length(model1$fitted.values))

head(model1_fitted)
```

```
## # A tibble: 6 x 3
##   fitted_values fitted_values2 index
##           <dbl>          <dbl> <int>
## 1          3.67           3.67     1
## 2          3.40           3.40     2
## 3         -1.18          -1.18     3
## 4         -2.43          -2.43     4
## 5          2.48           2.48     5
## 6          1.79           1.79     6
```

```
plot1 <- ggplot(model1_fitted) + geom_point(aes(y = fitted_values,
    x = fitted_values2))

ggsave(here("figures", "fitted_plot.pdf"))
```

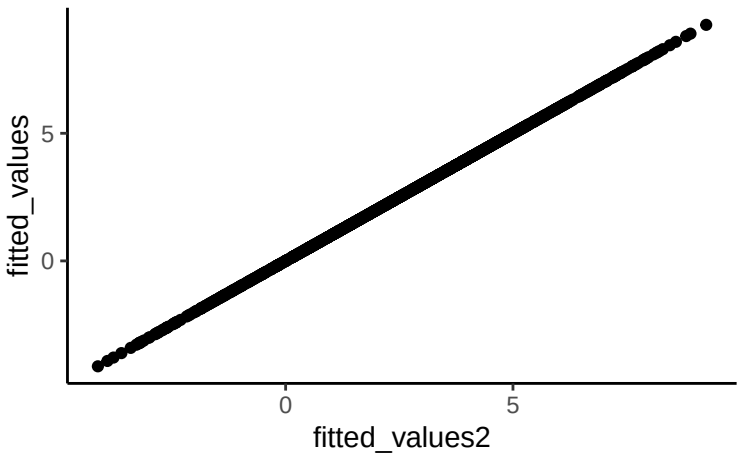

Figure 2: Plot of fitted values from model1 obtained from the glm function and the predict function.

Exercise 1:

Model residuals are typically calculated as the predicted minus the observed values. Can you generate a figure that plots the residuals obtained from `model1` above against residuals that you create using the `predict()` function in R?

2.2 Binary Outcome

When our outcome is binary, we may be interested in estimating the risk difference, risk ratio, or odds ratio for the exposure-outcome association. We can estimate each of these using GLMs but with different link functions and distributions.

We'll focus on the odds ratio first:

```
#' Here, we start fitting relevant regression models to the data.

model1 <- glm(wt_delta ~ qsmk + sex + age +
    income + sbp + dbp + price71 + tax71 +
    race, data = nhefs, family = binomial("logit"))
```

We can get the odds ratio from this model using the following code:

```
model1_OR <- tidy(model1)[2, ] %>%
    mutate(OR = exp(estimate), LCL = exp(estimate -
```

```
        1.96 * std.error), UCL = exp(estimate +
        1.96 * std.error)) %>%
    select(term, OR, LCL, UCL)

model1_OR
```

```
## # A tibble: 1 x 4
##   term     OR   LCL   UCL
##   <chr> <dbl> <dbl> <dbl>
## 1 qsmk   1.87  1.40  2.51
```

We can now take the results we obtained from the `model1_OR` object and incorporate them into our manuscript or report. For example, if you are writing a report in Microsoft Word, you may opt to save these results as an Excel spreadsheet:

```
write_excel_csv(model1_OR, file = here("misc",
    "OddsRatioResults.csv"))
```

Alternatively, if you are using a Markdown language such as RMarkdown, you can print the table directly with, for example, the `kable` package:

```
knitr::kable(model1_OR)
```

term	OR	LCL	UCL
qsmk	1.874316	1.402316	2.505184

There are many stylng themes for a `kable` table, and you can find additional information on `kable` and the themes available here.

3 Estimating Risk Differences and Ratios with GLM

Generalized linear models consist of a family of regression models that are fully characterized by a selected distribution and a link function. That is, to fully specify a GLM, one must select a distribution (which determines the form of the conditional mean and variance of the outcome) and a link function (which determines how the conditional mean of the outcome relates to the covariates).

There are a wide variety of distributions and link functions available in standard statistical software programs that fit GLMs. Here, we'll consider a binary outcome Y with probability $P(Y = 1)$, and focus attention on three link functions:

1. Logit, or the log-odds: $\log P(Y = 1)/[1 - P(Y = 1)]$

We saw above how to use R's GLM function to fit a logistic regression model.

2. Log: $\log[P(Y = 1)]$
3. Identity: $P(Y = 1)$.

A common misconception is that to use GLMs correctly, one must choose the distribution that best characterizes the data, as well as the canonical link function corresponding to this distribution. For example, if the outcome is binary, one "must" choose the binomial distribution with the logit link.

While the binomial distribution and logit link work well together for binary outcomes, they do not easily provide contrasts like the risk difference or risk ratio, because of the selected link function. Alternative specification of the distribution and link function for GLMs can address this limitation.

3.1 Link Functions and Effect Measures

There is an important relation between the chosen link function, and the interpretation of the coefficients from a GLM. For models of a binary outcome and the logit or log link, this relation stems from the properties and rules governing the natural logarithm. Specifically, the quotient rule for logarithms states that:

$$\log(X/Y) = \log(X) - \log(Y)$$

Because of this relation, the natural exponent of the coefficient in a logistic regression model yields an estimate of the odds ratio. However, by the same reasoning, exponentiating the coefficient from a GLM with a log link function and a binomial distribution (i.e., log-binomial regression) yields an estimate of the risk ratio.

Alternately, for GLM models with a binomial distribution and identity link function, because logarithms are not used, the unexponentiated coefficient yields an estimate of the risk difference.

Unfortunately, using a binomial distribution can lead to convergence problems with the $\log()$ or identity link functions for reasons that have been explored (Zou, 2004).

This will occur when, for example, the combined numerical value of all the independent variables in the model is very large. This can result in estimated probabilities that exceed 1, which violates the definition of a probability (binomial) model (probabilities can only lie between zero and one) and hence, convergence problems.

Let's see how these problems can be overcome.

3.2 GLMs for risk differences and ratios

For our analyses of the data described above using GLM with a binomial distributed outcome with a log link function to estimate the risk ratio and identity link function to estimate risk difference, an error is returned:

```
#' Here, we start fitting relevant regression models to the data.
#' modelForm is a regression argument that one can use to regress the
#' outcome (wt_delta) against the exposure (qsmk) and selected confounders.

#' This model can be used to quantify a conditionally adjusted risk
#' ratio with with correct standard error
#' However, error it returns an error and thus does not provide any results.

modelRR_binom <- glm(wt_delta ~ qsmk + sex +
    age + income + sbp + dbp + price71 +
    tax71 + race, data = nhefs, family = binomial("log"))
```

```
## Error: no valid set of coefficients has been found: please supply starting values
```

Why is this error returned? The most likely explanation in this context is as follows: We are modeling $P(Y = 1 \mid X) = \exp\{\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p\}$. In this context, there may be *no set of values* for the parameters in the model that yield $P(Y = 1 \mid X) < 1$ for every observation in the sample. Because R's glm function (under a binomial distribution) correctly recognizes this as a problem, it returns an error.

Instead, one may resort to using different distributions that are more compatible with the link functions that return the association measures of interest. For the risk ratio, one may use a GLM with a Poisson distribution and log link function. Doing so will return an exposure coefficient whose natural exponent can be interpreted as a risk ratio.

```
#' This model can be used to quantify a conditionally risk ratio
#' using the Poisson distribution and log link function.
#' However, because the Poisson distribution is used, the model
#' provides incorrect standard error estimates.
modelRR <- glm(wt_delta ~ qsmk + sex + age +
    income + sbp + dbp + price71 + tax71 +
    race, data = nhefs, family = poisson("log"))
tidy(modelRR)[2, ]
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic p.value
##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>
## 1 qsmk     0.177    0.0744      2.38  0.0173
```

It's important to recognize what we're doing here. We are using this model as a tool to quantify the log mean ratio contrasting $P(Y = 1 \mid X_{qsmk} = 1)$ to $P(Y = 1 \mid X_{qsmk} = 0)$ (all other things being equal). However, we should not generally assume that ever aspect of this model is correct. In particular, note that the max predicted probability from this model is 1.017:

```
summary(modelRR$fitted.values)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3728  0.5797  0.6686  0.6729  0.7639  1.0175
```

We can use the `augment` function in the `broom` package to evaluate the distribution of these probabilities (among other things):

```
fitted_dat <- augment(modelRR, type.predict = "response")

fitted_dat
```

```
## # A tibble: 1,394 x 16
##    wt_delta  qsmk   sex   age income   sbp   dbp price71 tax71  race .fitted
##       <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>   <dbl>
##  1        0     0     0    42     19   175    96    2.18 1.10      1   0.673
##  2        1     0     0    36     18   123    80    2.35 1.36      0   0.694
##  3        1     0     1    56     15   115    75    1.57 0.551     1   0.454
##  4        1     0     0    68     15   148    78    1.51 0.525     1   0.373
##  5        1     0     0    40     18   118    77    2.35 1.36      0   0.650
##  6        1     0     1    43     11   141    83    2.21 1.15      1   0.591
##  7        0     0     0    51     18   163    79    2.35 1.36      0   0.583
##  8        1     0     0    43     16   184   106    2.35 1.36      0   0.680
##  9        1     1     0    43     19   135    89    2.35 1.36      0   0.783
## 10        0     0     0    34     18   118    69    2.35 1.36      0   0.696
## # ... with 1,384 more rows, and 5 more variables: .resid <dbl>,
## #   .std.resid <dbl>, .hat <dbl>, .sigma <dbl>, .cooksd <dbl>
```

```
plot_hist <- ggplot(fitted_dat) + geom_histogram(aes(.fitted)) +
    scale_y_continuous(expand = c(0, 0)) +
    scale_x_continuous(expand = c(0, 0))

ggsave(here("figures", "2022_02_21-rr_hist_plot.pdf"),
    plot = plot_hist)
```

This distribution is shown in margin Figure 3. We can also see that there are only two observations in the sample with predicted risks greater than 1.

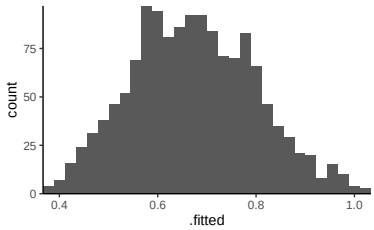

Figure 3: Distribution of fitted values from the Poisson GLM with log link function to obtain an estimate of the adjusted risk ratio for the association between quitting smoking and greater than median weight gain in the NHEFS.

```
fitted_dat %>%
    filter(.fitted >= 1) %>%
    select(wt_delta, qsmk, age, .fitted)
```

```
## # A tibble: 6 x 4
##   wt_delta  qsmk   age .fitted
##      <dbl> <dbl> <dbl>   <dbl>
## 1        1     1    30    1.01
## 2        1     1    25    1.00
## 3        1     1    28    1.01
```

```
## 4         1     1    26    1.01
## 5         1     1    25    1.00
## 6         1     1    35    1.02
```

For these reasons, we are not particularly concerned about the fact that the model predicts risks that are slightly large than 1. However, the model-based standard errors (i.e., the SEs that one typically obtains directly from the GLM output) are no longer valid. Instead, one should use the robust (or sandwich) variance estimator to obtain valid SEs (the bootstrap can also be used) (Zou, 2004).

```
library(sandwich)
#' To obtain the correct variance, we use the 'sandwich'
#' function to obtain correct sandwich (robust) standard
#' error estimates.
sqrt(sandwich(modelRR)[2, 2])
```

```
## [1] 0.03788821
```

Another way to obtain robust (sandwich) variance estimates that are accurate is to use the `coeftest` function in the `lmtest` package. To use this function, we take the model object created, and generate a variance covariance matrix for the model parameters using the `vcovHC` function. We then use this variance-covariance matrix as an argument in the `coeftest` function to obtain results from our risk ratio model with appropriate standard errors.

```
library(sandwich)
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
# first create sandwich vcov matrix
vcov_RR <- vcovHC(modelRR, type = "HC3",
    sandwich = T)

# then construct summary results using
# sandwich vcov matrix
coeftest(modelRR, vcov = vcov_RR, type = "HC3")
```

```
##
## z test of coefficients:
##
##                Estimate  Std. Error z value  Pr(>|z|)
## (Intercept) -0.89084451  0.38334831 -2.3239   0.02013 *
## qsmk         0.17716666  0.03819590  4.6384 3.512e-06 ***
## sex          0.04895944  0.03762030  1.3014   0.19312
## age         -0.01394634  0.00198279 -7.0337 2.011e-12 ***
## income       0.01342167  0.00765426  1.7535   0.07952 .
## sbp          0.00091467  0.00156077  0.5860   0.55785
## dbp          0.00188010  0.00237372  0.7920   0.42833
## price71      0.33398752  0.26732654  1.2494   0.21153
## tax71       -0.19088453  0.28027368 -0.6811   0.49583
## race        -0.03388321  0.06051136 -0.5599   0.57551
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# compare these results to model-based
# results
summary(modelRR)
```

```
##
## Call:
## glm(formula = wt_delta ~ qsmk + sex + age + income + sbp + dbp +
##     price71 + tax71 + race, family = poisson("log"), data = nhefs)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -1.4036  -1.0238   0.2407   0.4029   0.8480
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.8908445  0.6938982  -1.284   0.1992
## qsmk         0.1771667  0.0744447   2.380   0.0173 *
## sex          0.0489594  0.0670385   0.730   0.4652
## age         -0.0139463  0.0033089  -4.215  2.5e-05 ***
## income       0.0134217  0.0134123   1.001   0.3170
## sbp          0.0009147  0.0025233   0.362   0.7170
## dbp          0.0018801  0.0040506   0.464   0.6425
## price71      0.3339875  0.4884548   0.684   0.4941
## tax71       -0.1908845  0.5141144  -0.371   0.7104
## race        -0.0338832  0.1048866  -0.323   0.7467
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 743.24  on 1393  degrees of freedom
## Residual deviance: 710.24  on 1384  degrees of freedom
## AIC: 2606.2
##
## Number of Fisher Scoring iterations: 5
```

For the risk difference, one may use a GLM with a Gaussian (i.e., normal) distribution and identity link function, or, equivalently, an ordinary least squares estimator. Doing so will return an exposure coefficient that can be interpreted as a risk difference. However, once again the robust variance estimator (or bootstrap) should be used to obtain valid SEs.

```
#' This model can be used to obtain a risk difference
#' with the gaussian distribiton or using ordinary least
#' squares (OLS, via the lm function). Again, the model
#' based standard error estimates are incorrect.
```

```
modelRD <- glm(wt_delta ~ qsmk + sex + age +
    income + sbp + dbp + price71 + tax71 +
    race, data = nhefs, family = gaussian("identity"))
modelRD <- lm(wt_delta ~ qsmk + sex + age +
    income + sbp + dbp + price71 + tax71 +
    race, data = nhefs)
tidy(modelRD)[2, ]
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic   p.value
##   <chr>    <dbl>     <dbl>     <dbl>     <dbl>
## 1 qsmk     0.123    0.0287      4.30 0.0000182
```

```
#' To obtain the correct variance, we use the 'sandwich' function
#' to obtain correct sandwich (robust) standard error estimates.
sqrt(sandwich(modelRD)[2, 2])
```

```
## [1] 0.02685004
```

The risk ratio and difference, as well as the 95% sandwich variance confidence intervals, obtained for the relation between quitting smoking and greater than median weight change are provided Table 1.

```
knitr::kable(table1_data)
```

Method	Risk Difference	Risk Ratio
GLM	0.14 (0.09, 0.20)	1.32 (1.19, 1.46)
Marginal Standardization	0.14 (0.09, 0.21)	1.31 (1.18, 1.46)

Results in this table obtained using a conditionally adjusted regression model without interactions. Gaussian distribution and identity link was used to obtain the risk difference. A Poisson distribution and log link was used to obtain the risk ratio. 95% CIs obtained via the sandwich variance estimator. 95% CIs obtained using the bias-corrected and accelerated bootstrap CI estimator.

Unfortunately, use of a Poisson or Gaussian distribution for GLMs for a binomial outcome can introduce different problems. For one, while not entirely worrysome in our setting, a model that predicts probabilities greater than one

should not instill confidence in the user. Second, performance of the robust variance estimator is notoriously poor with small sample sizes. Finally, the interpretation of the risk differences and ratios becomes more complex when the exposure interacts with other variables in the model.

Table 2: Methods to use for quantifying conditionally adjusted odds ratios, risk ratios, and risk differences.

Odds Ratio	Risk Ratio	Risk Difference
GLM Family = Binomial	GLM Family = Binomial	GLM Family = Binomial
GLM Link = Logistic	GLM Link = Log	GLM Link = Identity
Standard Errors = Model Based	Standard Errors = Model Based	Standard Errors = Model Based
	GLM Family = Poisson	GLM Family = Gaussian
	GLM Link = Log	GLM Link = Identity
	Standard Errors = Sandwich	Standard Errors = Sandwich
		Least Squares Regression
		Standard Errors = Sandwich

For instance, let's assume that in the NHEFS data, the association between quitting smoking and weight gain interacts with race:

```
#' Potential evidence for interaction
#' between smoking and exercise on the risk difference scale?

table(nhefs$race)
```

```
##
##    0    1
## 1210  184
```

```
summary(glm(wt_delta ~ qsmk + sex + age +
    income + sbp + dbp + price71 + tax71 +
    race, data = nhefs, family = binomial(link = "identity")))
```

```
##
## Call:
```

```
## glm(formula = wt_delta ~ qsmk + sex + age + income + sbp + dbp +
##     price71 + tax71 + race, family = binomial(link = "identity"),
##     data = nhefs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2397  -1.2251   0.7255   0.9161   1.5192
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.2692755  0.2517021   1.070   0.2847
## qsmk         0.1249774  0.0257910   4.846 1.26e-06 ***
## sex          0.0402632  0.0242204   1.662   0.0964 .
## age         -0.0081188  0.0011734  -6.919 4.55e-12 ***
## income       0.0084146  0.0048692   1.728   0.0840 .
## sbp          0.0001755  0.0009152   0.192   0.8479
## dbp          0.0016306  0.0014482   1.126   0.2602
## price71      0.2900214  0.1761072   1.647   0.0996 .
## tax71       -0.2060878  0.1853473  -1.112   0.2662
## race        -0.0099910  0.0379658  -0.263   0.7924
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1762.3  on 1393  degrees of freedom
## Residual deviance: 1665.7  on 1384  degrees of freedom
## AIC: 1685.7
##
## Number of Fisher Scoring iterations: 7
```

```
summary(glm(wt_delta ~ qsmk + sex + age +
    income + sbp + dbp + price71 + tax71 +
    race + qsmk * race, data = nhefs, family = binomial(link = "identity")))
```

```
##
## Call:
## glm(formula = wt_delta ~ qsmk + sex + age + income + sbp + dbp +
##     price71 + tax71 + race + qsmk * race, family = binomial(link = "identity"),
##     data = nhefs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2372  -1.2252   0.7257   0.9141   1.5219
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.2684311  0.2519564   1.065   0.2867
## qsmk         0.1243713  0.0270304   4.601 4.20e-06 ***
## sex          0.0401692  0.0242218   1.658   0.0972 .
## age         -0.0081206  0.0011735  -6.920 4.52e-12 ***
## income       0.0084319  0.0048706   1.731   0.0834 .
## sbp          0.0001792  0.0009154   0.196   0.8448
## dbp          0.0016331  0.0014484   1.127   0.2596
## price71      0.2899049  0.1762274   1.645   0.1000 .
## tax71       -0.2056912  0.1854535  -1.109   0.2674
## race        -0.0113103  0.0421883  -0.268   0.7886
## qsmk:race    0.0081338  0.0890813   0.091   0.9272
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1762.3  on 1393  degrees of freedom
## Residual deviance: 1665.7  on 1383  degrees of freedom
## AIC: 1687.7
##
## Number of Fisher Scoring iterations: 7
```

If this were the case, to properly interpret the association, the interaction between race and qsmk should be considered. But how could we do this? One

approach would be to include the interaction term and interpret the association between quitting smoking and weight change separately for each racial category.

For example, in the model that includes the interaction term with exercise, we can no longer simply interpret the coefficient for `qsmk` as the treatment effet of interest. Instead, (under causal identifiability) we have two treatment effects: The effect of `qsmk` for those with `race = 0` and `1`. If we were, in fact, interested in the average treatment effect in the sample (and not a unique treatment effect for each racial group), we would have to take a weighted average of the coefficients for these effects, where the weights are defined as a function of the proportion of individuals in each racial category.

Clearly, this approach can quickly become too burdensome when there are several relevant interactions in the model, and is not worth the effort when we are simply interested in the marginal association. As an alternative, we can use marginal or model-based standardization, which can greatly simplify the process.

3.3 Marginal or Model-Based Standardization

Another approach to obtaining risk differences and ratios from GLMs that are not subject to the limitations noted above is to use marginal standardization, which is equivalent to g computation when the exposure is measured at a single time point (Naimi et al., 2017). This process can be implemented by fitting a single logistic model, regressing the binary outcome against all confounder variables, including all relevant interactions. But instead of reading the coefficients the model, one can obtain odds ratios, risk ratios, or risk differences by using this model to generate predicted risks for each individual under "exposed" and "unexposed" scenarios in the dataset. To obtain standard errors, the entire procedure must be bootstrapped (see supplemental material for code). These marginal risk differences and ratios, as well as their bootstrapped CIs are presented in the table above.

Here is some code to implement this marginal standardization in the NHEFS data:

```
#' Marginal Standardization: version 1
```

```r
#' Regress the outcome against the confounders with interaction
ms_model <- glm(wt_delta ~ qsmk + sex + age +
    income + sbp + dbp + price71 + tax71 +
    race + race * qsmk, data = nhefs, family = binomial("logit"))
##' Generate predictions for everyone in the sample to obtain
##' unexposed (mu0 predictions) and exposed (mu1 predictions) risks.
mu1 <- predict(ms_model, newdata = transform(nhefs,
    qsmk = 1), type = "response")
mu0 <- predict(ms_model, newdata = transform(nhefs,
    qsmk = 0), type = "response")

#' Marginally adjusted odds ratio
marg_stand_OR <- (mean(mu1)/mean(1 - mu1))/(mean(mu0)/mean(1 -
    mu0))
#' Marginally adjusted risk ratio
marg_stand_RR <- mean(mu1)/mean(mu0)
#' Marginally adjusted risk difference
marg_stand_RD <- mean(mu1) - mean(mu0)

#' Using the bootstrap to obtain confidence intervals for the marginally adjusted
#' risk ratio and risk difference.
bootfunc <- function(data, index) {
    boot_dat <- data[index, ]
    ms_model <- glm(wt_delta ~ qsmk + sex +
        age + income + sbp + dbp + price71 +
        tax71 + race + race * qsmk, data = boot_dat,
        family = binomial("logit"))
    mu1 <- predict(ms_model, newdata = transform(boot_dat,
        qsmk = 1), type = "response")
    mu0 <- predict(ms_model, newdata = transform(boot_dat,
        qsmk = 0), type = "response")

    marg_stand_OR_ <- (mean(mu1)/mean(1 -
        mu1))/(mean(mu0)/mean(1 - mu0))
```

```
    marg_stand_RR_ <- mean(mu1)/mean(mu0)
    marg_stand_RD_ <- mean(mu1) - mean(mu0)
    res <- c(marg_stand_RD_, marg_stand_RR_,
        marg_stand_OR_)
    return(res)
}

#' Run the boot function. Set a seed to obtain reproducibility
set.seed(123)
boot_res <- boot(nhefs, bootfunc, R = 2000)

boot_RD <- boot.ci(boot_res, index = 1)
```

```
## Warning in boot.ci(boot_res, index = 1): bootstrap variances needed for
## studentized intervals
```

```
boot_RR <- boot.ci(boot_res, index = 2)
```

```
## Warning in boot.ci(boot_res, index = 2): bootstrap variances needed for
## studentized intervals
```

```
boot_OR <- boot.ci(boot_res, index = 3)
```

```
## Warning in boot.ci(boot_res, index = 3): bootstrap variances needed for
## studentized intervals
```

```
marg_stand_OR
```

```
## [1] 1.797173
```

```
marg_stand_RR
```

```
## [1] 1.188549
```

```
marg_stand_RD
```

```
## [1] 0.1211166
```

```
boot_RD
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, index = 1)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.0695,  0.1722 )   ( 0.0710,  0.1749 )
##
## Level     Percentile            BCa
## 95%   ( 0.0673,  0.1713 )   ( 0.0651,  0.1692 )
## Calculations and Intervals on Original Scale
```

```
boot_RR
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, index = 2)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 1.103,  1.272 )   ( 1.101,  1.275 )
##
## Level     Percentile            BCa
## 95%   ( 1.102,  1.276 )   ( 1.099,  1.271 )
## Calculations and Intervals on Original Scale
```

```
boot_OR
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, index = 3)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 1.282,  2.263 )   ( 1.236,  2.218 )
##
## Level     Percentile            BCa
## 95%   ( 1.376,  2.359 )   ( 1.354,  2.325 )
## Calculations and Intervals on Original Scale
```

While this marginal standardization approach is more flexible in that it accounts for the interaction between quitting smoking and race, and still yields an estimate of the average treatment effect (again, under identifiability), it still assumes a constant effect of qsmk on weight change across levels of all of the other variables in the model. This constant effect assumption might be true, but if one wanted to account for potential interactions between the exposure and all of the confounders in the model, there is an easy way. We call this the "stratified modeling approach."

This stratified modeling approach avoids the exposure effect homogeneity assumption across levels of all the confounders. In effect, the approach fits a separate model for each exposure stratum. To obtain predictions under the "exposed" scenario, we use the model fit to the exposed individuals to generate predicted outcomes in the entire sample. To obtain predictions under the "unexposed" scenario, we repeat the same procedure, but with the model fit among the unexposed. One can then average the risks obtained under each exposure scenario, and take their difference and ratio to obtain the risk differences and ratios of interest.

```r
#' Marginal Standardization
##' To avoid assuming no interaction between
##' smoking and any of the other variables
##' in the model, we subset modeling among
##' exposed/unexposed. This code removes smoking from the model,
##' which will allow us to regress the outcome
##' against the confounders among the exposed and
##' the unexposed searately. Doing so will allow us
##' to account for any potential exposure-covariate interactions
##' that may be present.

#' Regress the outcome against the confounders
#' among the unexposed (model0) and then among the exposed (model1)
model0 <- glm(wt_delta ~ sex + age + income +
    sbp + dbp + price71 + tax71 + race, data = subset(nhefs,
    qsmk == 0), family = binomial("logit"))
model1 <- glm(wt_delta ~ sex + age + income +
    sbp + dbp + price71 + tax71 + race, data = subset(nhefs,
    qsmk == 1), family = binomial("logit"))
##' Generate predictions for everyone in the sample using the model fit to only the
##' unexposed (mu0 predictions) and only the exposed (mu1 predictions).
mu1 <- predict(model1, newdata = nhefs, type = "response")
mu0 <- predict(model0, newdata = nhefs, type = "response")

#' Marginally adjusted odds ratio
marg_stand_OR <- (mean(mu1)/mean(1 - mu1))/(mean(mu0)/mean(1 -
    mu0))
#' Marginally adjusted risk ratio
marg_stand_RR <- mean(mu1)/mean(mu0)
#' Marginally adjusted risk difference
marg_stand_RD <- mean(mu1) - mean(mu0)

#' Using the bootstrap to obtain confidence intervals for the marginally adjusted
#' risk ratio and risk difference.
```

```
bootfunc <- function(data, index) {
    boot_dat <- data[index, ]
    model0 <- glm(wt_delta ~ sex + age +
        income + sbp + dbp + price71 + tax71 +
        race, data = subset(boot_dat, qsmk ==
        0), family = binomial("logit"))
    model1 <- glm(wt_delta ~ sex + age +
        income + sbp + dbp + price71 + tax71 +
        race, data = subset(boot_dat, qsmk ==
        1), family = binomial("logit"))
    mu1 <- predict(model1, newdata = boot_dat,
        type = "response")
    mu0 <- predict(model0, newdata = boot_dat,
        type = "response")

    marg_stand_OR_ <- (mean(mu1)/mean(1 -
        mu1))/(mean(mu0)/mean(1 - mu0))
    marg_stand_RR_ <- mean(mu1)/mean(mu0)
    marg_stand_RD_ <- mean(mu1) - mean(mu0)
    res <- c(marg_stand_RD_, marg_stand_RR_,
        marg_stand_OR_)
    return(res)
}

#' Run the boot function. Set a seed to obtain reproducibility
set.seed(123)
boot_res <- boot(nhefs, bootfunc, R = 2000)

boot_RD <- boot.ci(boot_res, index = 1)
```

```
## Warning in boot.ci(boot_res, index = 1): bootstrap variances needed for
## studentized intervals
```

```
boot_RR <- boot.ci(boot_res, index = 2)
```

```
## Warning in boot.ci(boot_res, index = 2): bootstrap variances needed for
## studentized intervals
```

```
boot_OR <- boot.ci(boot_res, index = 2)
```

```
## Warning in boot.ci(boot_res, index = 2): bootstrap variances needed for
## studentized intervals
```

```
marg_stand_OR
```

```
## [1] 1.789638
```

```
marg_stand_RR
```

```
## [1] 1.187268
```

```
marg_stand_RD
```

```
## [1] 0.1203232
```

```
boot_RD
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, index = 1)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.0676,  0.1730 )   ( 0.0685,  0.1767 )
##
```

```
## Level     Percentile            BCa
## 95%   ( 0.0639,  0.1722 )   ( 0.0636,  0.1719 )
## Calculations and Intervals on Original Scale
```

```
boot_RR
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, index = 2)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 1.100,  1.273 )   ( 1.098,  1.277 )
##
## Level     Percentile            BCa
## 95%   ( 1.097,  1.277 )   ( 1.097,  1.276 )
## Calculations and Intervals on Original Scale
```

```
boot_OR
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, index = 2)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 1.100,  1.273 )   ( 1.098,  1.277 )
##
## Level     Percentile            BCa
## 95%   ( 1.097,  1.277 )   ( 1.097,  1.276 )
## Calculations and Intervals on Original Scale
```

When predicted risks are estimated using a logistic model, relying on marginal standardization will not result in probability estimates outside the bounds [0, 1]. And because the robust variance estimator is not required, model-based standardization will not be as affected by small sample sizes. However, the bootstrap is more computationally demanding than alternative variance estimators, which may pose problems in larger datasets.

References

Ashley I Naimi, Stephen R Cole, and Edward H Kennedy. An Introduction to G Methods. *Int J Epidemiol*, 46(2):756–62, 2017.

Guangyong Zou. A modified poisson regression approach to prospective studies with binary data. *Am J Epidemiol*, 159(7):702–706, Apr 2004.