Binary Logistic Regression
Week 8

PH 700A, Spring 2025

Rick Calvo

Table of contents

1 Week 8

1.1 Packages

`library(stats)`

`library(glmtoolbox)` - Could take a little bit of time to install.

1.2 Session Overview

- Logistic Regression
- Background
- Distribution
- Assumptions
- Commands
- Data Requirements
- Model Development
- Diagnostics
- Visualization

2 Logistic Regression

2.1 Categorical Outcomes

Your dependent variable is a factor data type.

- Binary *YES* vs. *NO*
- Nominal *SICK* vs. *HEALTHY*
- Ordered *LOW* vs. *HIGH*
- Derived *APGAR < 8* vs. *APGAR >=8*

2.2 Background

Models the log-odds of a **categorical outcome** as a linear combination (an equation) of one or more independent variables.

$$ln(\frac{p_{event}}{p_{1-event}}) = b_0 + b_1(x_1) + b_2(x_2) + ... + b_k(x_k)$$

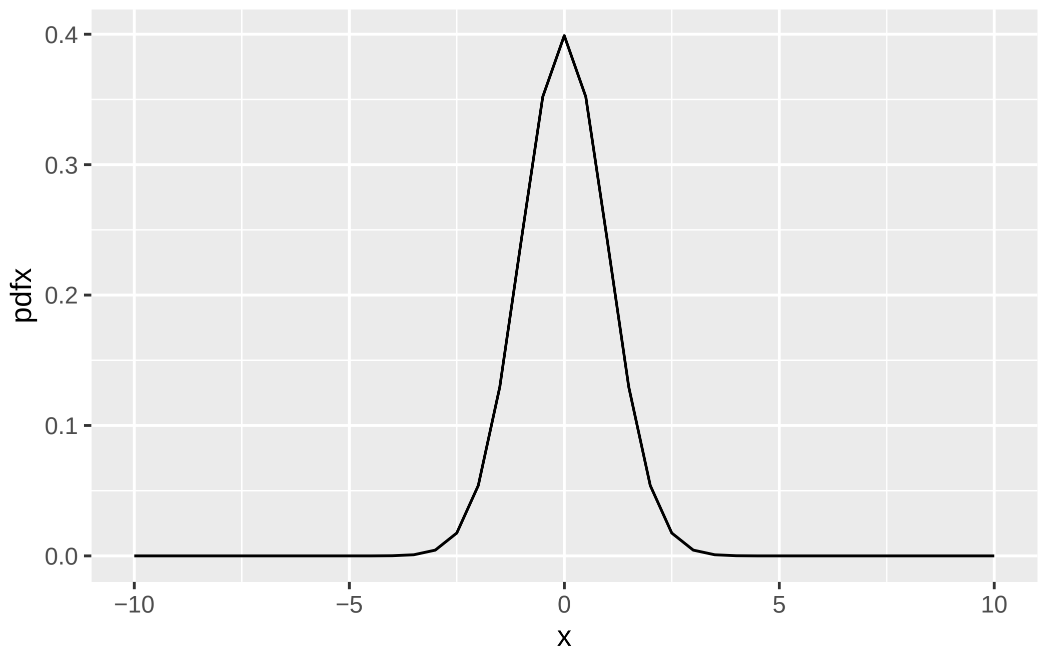
Types of logistic regression:

- Binary
- Multinomial/Polytomous
- Ordinal
- Conditional Binary

All make use of the **logistic curve** and **Maximum Likelihood Estimation** techniques to generate probability estimates.

2.3 Binomial Probability Density Function

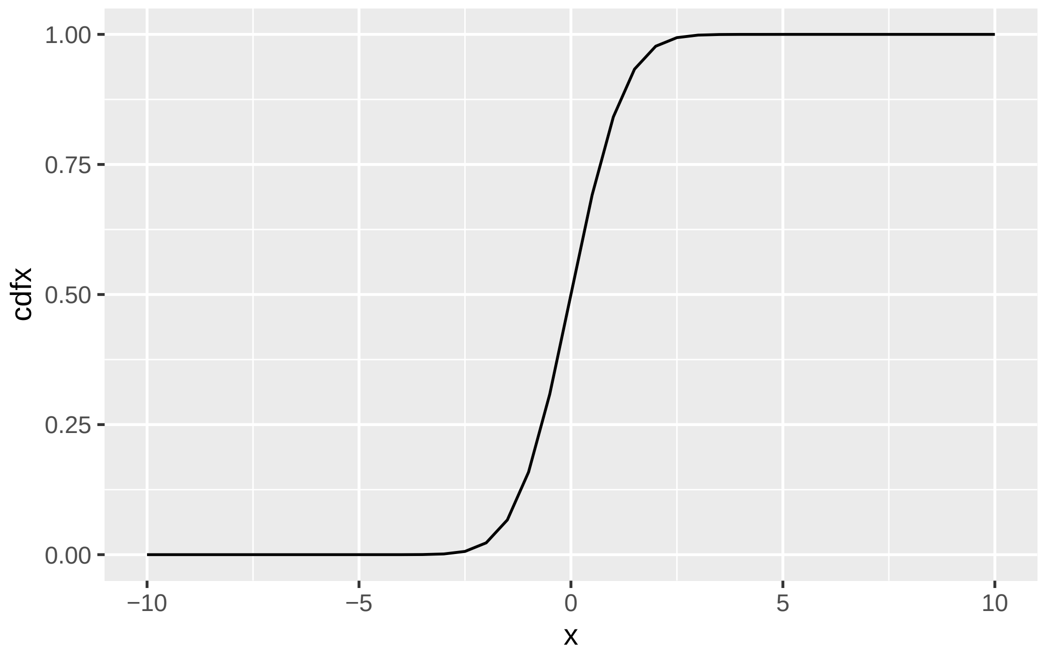
```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```



A standard binomial (success vs. failure) probability distribution appears Gaussian around a mean `x` value, demonstrating the highest rate of success exists at the mean.

The area of under the curve can be used to calculate the *cumulative distribution function* to translate `x` values to outcome probabilities.

2.4 Logistic Curve - Cumulative Distribution Function



A mapping of `x` to `p`. At the mean `x`, a threshold is reached that *should* relate to a change in outcome probability.

Where:

- `p` is your probability of the outcome bounded by 0 and 1
- `x` represents values of your independent variable.
- The distribution of `p` is binomial in nature; the risk for the outcome grows based on rising `x`
- A monotonic relationship exists between `p` and `x` (assuming `x` is ordinal in some regard)

https://campus.datacamp.com/courses/intermediate-regression-in-r/multiple-logistic-regression?ex=6

2.5 Commands

`glm()` ["generalized linear model"] allows for flexible modeling strategies.

Linear regression can be achieved with `lm()` or `glm()`

```
lm(y ~ x1 + x2 + ... + xn, data = df)

glm(y ~ x1 + x2 + ... + xn, data = df, family = gaussian)
```

Logistic regression is a linear equation that models *probability* instead of a *value*.

It can be achieved with `glm()`.

```
library(stats)

model <- glm(formula = y ~ x1 + x2 + ... + xn,
             data = df,
             family = binomial)`
```

`model` is the object that will contain your logistic regression results

`x1 + x2 + ... + xn` are independent variables in the model, separated by `+` signs

`family = binomial` specifies that the `glm` procedure will apply a `binomial` probability distribution

2.6 Usage in Public Health

Design Types:

- Cohort
- Case-control
- Cross-sectional

Logistic regression can be used in all these designs.

Exposure and Outcome

Your independent variables can be continuous or categorical. The outcome must be dichotomous categorical for standard logistic regression. Logistic regression *does not calculate risk* for an outcome, though risk can be approximated depending on the design type.

2.7 Assumptions

Basically the same as linear regression.

- Outcome frequency is sufficient to prevent a type II error
- Continuous variables are normal
- Categorical variables are coded as *factors* or *ordinal*
- Observations must take only one of two possible choices (`0` [non-event] vs. `1` [event])
- Missing values must be completely at random
- No collinearity among independent variables
- Suspected interactions are evaluated and addressed accordingly

2.8 Variable Assessment

Start with your bivariate analyses to identify candidates.

Select variables that are:

- Your primary exposure
- Confounding your primary relationship
- Interesting to answering your hypothesis
- Affecting model fitness

2.9 Model Development

Modeling Methods:

- Manual
- Backward Stepwise
- Forward Stepwise
- Hierarchical Stepwise

2.10 Automated Modeling

2.10.1 Backward Stepwise

Backward stepwise logit starts with the full model and by default will work down.

```
bwLogit <- step(fullModel, direction = "backward")
```

2.10.2 Forward Stepwise

Forward stepwise logit starts with the empty model but you have to give it a range of models it can work with

```
fwLogit <- step(smallModel, scope = list(lower = formula(smallModel), upper = formula(fullModel)),
direction = "forward")
```

- `lower = formula(smallModel)` is the lowest model of the modeling range – essentially the smallest permissible model appropriate to your study
- `upper = formula(fullModel)` is the upper bound of the modeling range – it should be the fully-filled model with all covariates
- R will start at `smallModel` and add variables until it evaluates all variables in `fullModel`

2.10.3 Hierarchical Stepwise

Hierarchical "both ways" starts at any initial model provided (in this case `smallModel`) and will add and eliminate variables within the range of models provided

```
bothLogit <- step(smallModel, scope = list(lower = formula(smallModel), upper = formula(fullModel)),
direction = "both", trace=0)
```

2.11 Diagnostics and Performance

AICs for Selection

Hosmer-Lemeshow Goodness-of-fit for Data Fitness

Sensivitity and Specificity (aka Validity) - **SAVING THIS FOR LATER!**

2.12 Analysis Initialization

First, evaluate outcome frequency to make sure you have sufficient events.

```r
library(tidyverse)
library(explore)

table(df$outcome)

df %>% explore(outcome)
```

- Assess missing value patterns
- Evaluate coding errors
- Establish variable as a factor with appropriate reference category

2.13 Independent Variables

Example variable schema:

Domain	Continuous	Categorical
Demographics	age	age4cat, sex, ethnicity
Hospital Factors		pedlvl, combolvl, teaching
Injury	iss, totalgcs	tbiConcuss, tbiEdema, tbiBleed primaryMechanism
Treatments	days2surg	craniect, surgtype
Complications		comp_infectious
Course of Care	hlosdays, totaliculos	

Evaluate variable forms, distributions, frequencies, missing values, coding errors.

Transform or categorize appropriately.

2.14 Model Development

Manual model development allows for analyst expertise to guide selection.

```r
model <- glm(formula = admitted ~ heartrate + o2sat + sbp + weakness +
                                  highpain + gender + whitebin + pain_noscore +
                                  hypotensive,
             family = binomial,
             data = df)

summary(model)
```

```
Call:
glm(formula = admitted ~ heartrate + o2sat + sbp + weakness +
    highpain + gender + whitebin + pain_noscore + hypotensive,
    family = binomial, data = df)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  36.451686  10.249985   3.556 0.000376 ***
heartrate     0.015871   0.009469   1.676 0.093737 .
o2sat        -0.361558   0.101853  -3.550 0.000386 ***
sbp          -0.013171   0.006783  -1.942 0.052173 .
weakness      1.576459   0.818444   1.926 0.054084 .
highpain     -0.145016   0.376904  -0.385 0.700420
genderM       0.699216   0.382599   1.828 0.067618 .
whitebin     -0.507492   0.415243  -1.222 0.221648
pain_noscore  0.714625   0.873879   0.818 0.413493
hypotensive  -0.193227   0.819669  -0.236 0.813636
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 252.59  on 196  degrees of freedom
Residual deviance: 223.12  on 187  degrees of freedom
  (25 observations deleted due to missingness)
AIC: 243.12

Number of Fisher Scoring iterations: 5
```

The `summary()` command will output all the calculated statistics and model characteristics.

2.15 Collinearity Check

```r
library(car)

vif(model)
```

```
   heartrate        o2sat          sbp     weakness     highpain       gender
    1.138671     1.157102     1.332648     1.062609     1.160973     1.368240
    whitebin pain_noscore  hypotensive
    1.532529     1.082887     1.265378
```

Variables that are collinear will have to be dealt with accordingly.

When variables explain the same facet, it's appropriate to only retain one.

2.16 AICs for Variable Selection

```
model.x1 <- glm(formula = admitted ~ heartrate + o2sat + weakness + gender +
                          whitebin + highpain + pain_noscore,
             family = binomial,
             data = df)

model.x2 <- glm(formula = admitted ~ o2sat + sbp + weakness + gender +
                          whitebin + highpain + pain_noscore,
             family = binomial,
             data = df)

model.x3 <- glm(formula = admitted ~ heartrate + o2sat + sbp + weakness +
                          gender + whitebin + highpain,
             family = binomial,
             data = df)

model.x4 <- glm(formula = admitted ~ heartrate + o2sat + sbp + weakness +
                          gender + whitebin + pain_noscore,
             family = binomial,
             data = df)

extractAIC(model.x1)
```

```
[1]   8.0000 243.3811
```

```
extractAIC(model.x2)
```

```
[1]   8.0000 242.8478
```

```
extractAIC(model.x3)
```

```
[1]   8.0000 239.8968
```

```
extractAIC(model.x4)
```

```
[1]   8.0000 239.3133
```

Four models were generated:

- `model.x1` does not contain `sbp`
- `model.x2` does not contain `heartrate`
- `model.x3` does not contain `pain_noscore`
- `model.x4` does not contain `highpain`.

Evaluate the AIC values of each model and use the one with the lowest value.

2.17 Model Fitness

The Hosmer-Lemeshow Goodness-of-Fit test evaluates how well the model fits the data.

Package `glmtoolbox` contains the appropriate commands.

The command to perform the Goodness-of-Fit test is `hltest()`.

```
library(glmtoolbox)

hltest(model.x1)
```

```

   The Hosmer-Lemeshow goodness-of-fit test

 Group Size Observed  Expected
     1   20       10  7.651545
     2   20       12  9.565188
     3   20       13 11.078373
     4   20       10 11.896552
     5   20       10 12.628249
     6   20       11 13.647349
     7   20       13 14.728058
     8   20       14 15.765984
     9   20       20 17.062008
    10   17       17 15.976693

         Statistic =  13.18594
degrees of freedom =  8
           p-value =  0.10561
```

```
hltest(model.x2)
```

```

   The Hosmer-Lemeshow goodness-of-fit test

 Group Size Observed  Expected
     1   20        7  7.098023
     2   20       12  9.542949
     3   20       11 10.764206
     4   20       12 11.654281
     5   20       12 12.468769
     6   20        9 13.752449
     7   20       16 14.902954
     8   20       17 15.916145
     9   20       16 17.046998
    10   18       18 16.853226

         Statistic =  8.89072
```

```
degrees of freedom =  8
           p-value =  0.3516
```

```r
hltest(model.x3)
```

```

   The Hosmer-Lemeshow goodness-of-fit test

 Group Size Observed  Expected
     1   20        9  7.180343
     2   20       12  9.482347
     3   20       11 10.621840
     4   20       10 11.730537
     5   20       11 12.795633
     6   20       12 13.930953
     7   20       16 14.999815
     8   20       12 15.813107
     9   20       20 17.383997
    10   17       17 16.061428

         Statistic =  12.88026
degrees of freedom =  8
           p-value =  0.11604
```

```r
hltest(model.x4)
```

```

   The Hosmer-Lemeshow goodness-of-fit test

 Group Size Observed  Expected
     1   20        9  7.114673
     2   20       12  9.436482
     3   20       13 10.637177
     4   20        7 11.735109
     5   20       13 12.788119
     6   20       11 13.948149
     7   20       15 14.914752
     8   20       13 15.835572
     9   20       20 17.503585
    10   17       17 16.086381

         Statistic =  16.16598
degrees of freedom =  8
           p-value =  0.040065
```

Characteristic	OR	95% CI	p-value
heartrate	1.02	1.00, 1.04	0.058
o2sat	0.70	0.56, 0.84	<0.001
sbp	0.99	0.98, 1.00	0.042
weakness	4.97	1.19, 34.2	0.050
gender			
F	—	—	
M	1.95	0.93, 4.15	0.078
whitebin	0.59	0.26, 1.31	0.2
highpain	0.92	0.45, 1.92	0.8

Abbreviations: CI = Confidence Interval, OR = Odds Ratio

3 Visualizing Results

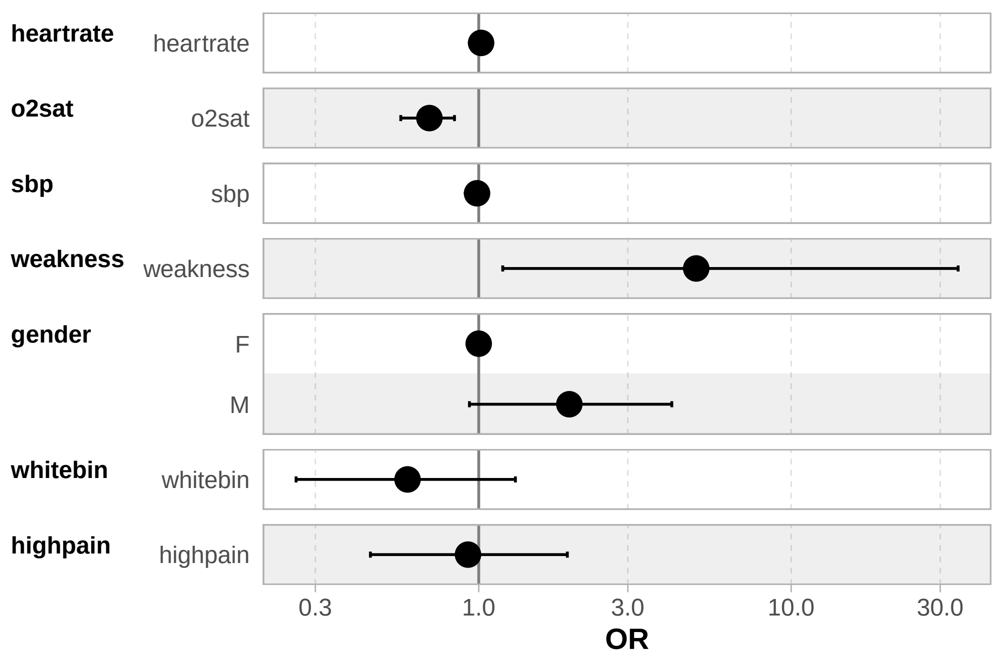
3.1 Regression Table

Utilizes table production features of `library(gtsummary)`.

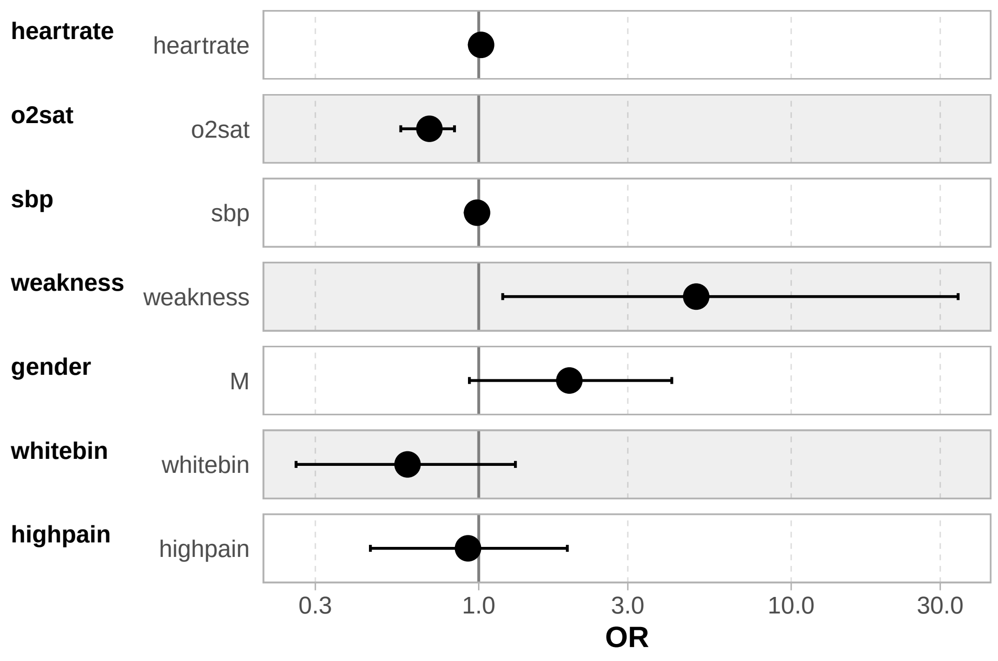
```r
library(gtsummary)

tbl_regression(model.x3, exponentiate = TRUE)
```

3.2 Forest Plot of Estimates

```r
# without reference cats
tbl_regression(model.x3,
               add_estimate_to_reference_rows = TRUE,
               exponentiate = TRUE) %>%
  plot(remove_reference_rows = FALSE)
```



```r
# with reference cats
tbl_regression(model.x3,
               add_estimate_to_reference_rows = TRUE,
               exponentiate = TRUE) %>%
  plot(remove_reference_rows = TRUE)
```


heartrate
heartrate
o2sat
o2sat
sbp
sbp
weakness
weakness
gender
M
whitebin
whitebin
highpain
highpain
OR