Categorical Data Analysis Lab

Young-geun Kim
2013310512, Department of Statistics
dudrms33@g.skku.edu

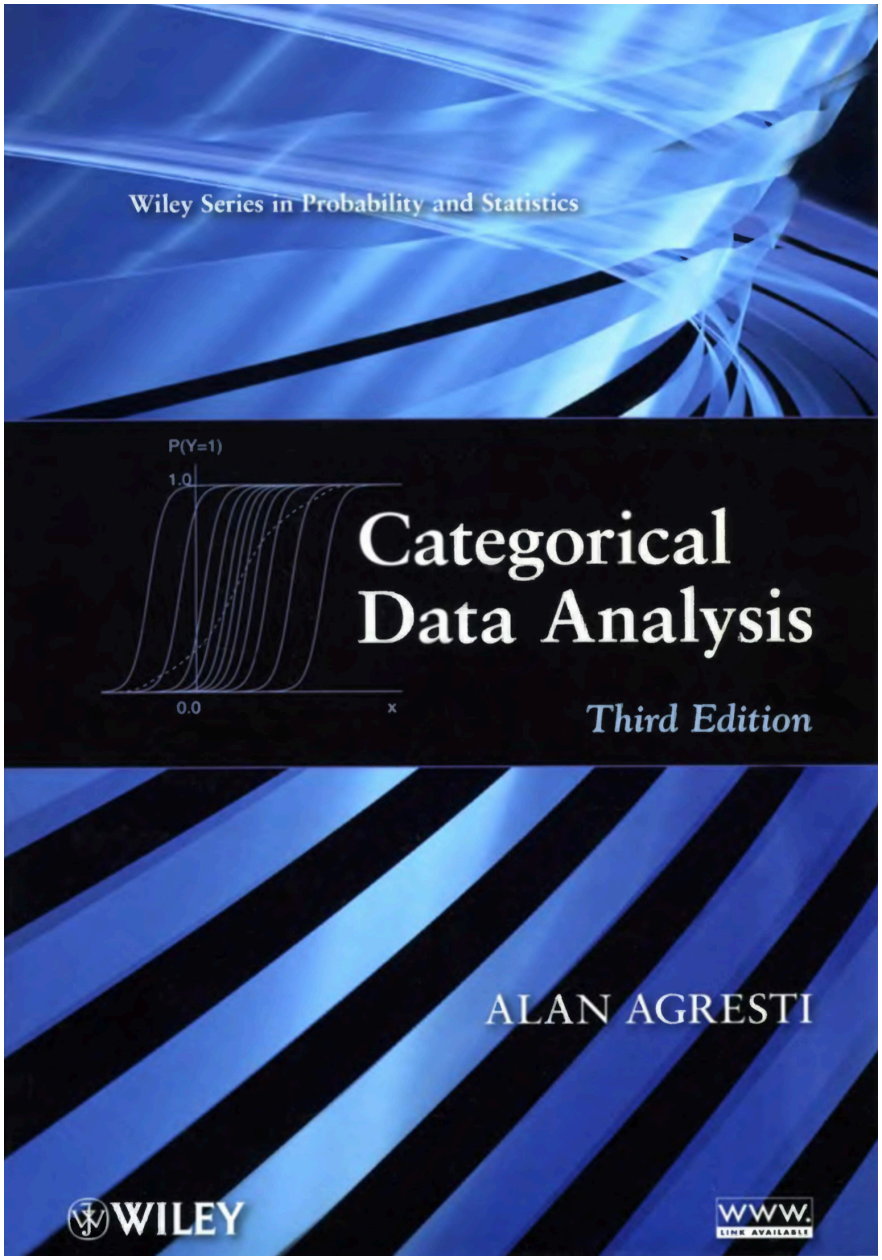
19 Dec, 2018

Contents

Chapter 1

Categorical Data Analysis

Study how to use R for categorical data in view of Agresti (2012)


Wiley Series in Probability and Statistics
P(Y=1)
1.0
0.0
x
Categorical
Data Analysis
Third Edition
ALAN AGRESTI
WILEY
WWW.
LINK AVAILABLE

1.1 Software usage

```
               _
platform       x86_64-apple-darwin15.6.0
arch           x86_64
os             darwin15.6.0
system         x86_64, darwin15.6.0
status
major          3
minor          5.1
year           2018
month          07
day            02
svn rev        74947
language       R
version.string R version 3.5.1 (2018-07-02)
nickname       Feather Spray
```

using IDE:

Chapter 2

Introduction to Generalized Linear Models

```
library(tidyverse)
```

2.1 Generalized Linear Models

Ordinary regression models try to find the best fit for mean response, where the response observations have i.i.d. Normal distribution and linear systematic part. We can extend this model to explain various types of variables such as count data with poisson distribution and probability with binomial distribution, et cetera. We name this model as *Generalized linear models* (GLMs). They have three components:

1. A *random component*: the response variable Y from natural exponential family
2. A *systematic compoenet*: the explanatrory variables form a linear predictor function
3. A *link function*: a link g describes the relationship between the systematic component and expected value of the random component, where g is monotonic and differentiable function

In sum, we can have a form of

$$\boldsymbol{\eta} = g(\boldsymbol{\mu}) = X\boldsymbol{\beta}$$

where X is a sample data matrix.

2.1.1 Link functions

For a given response variable, what link function should we use? Any *monotone differentiable* function can be used as link function. For example, we have used *identity link* for Gaussian response data. log-link might be applied to count data of *Poisson distribution* that have positive support. However, among many link functions, so-called **canonical links** are used practically.

2.1.2 Exponential family

In GLMs, random component is assumed to be from *natrual exponential family*, whose density or mass function is defined by

$$f(y_i;\theta_i) := a(\theta_i)b(y_i)\exp[y_iQ(\theta_i)], \quad i = 1, 2, \ldots, N \tag{2.1}$$

Here $Q(\theta)$ is called the *natural parameter*.

2.1.3 Binomial logit models for binary data

2.1.4 Poisson loglinear models for count data

2.1.5 Generalized linear models for continuous responses

2.1.6 Deviance

2.2 GLMs for Binary data

Let Y be a binary response variable.

$$Y = \begin{cases} 1 & \text{if success} \\ 0 & \text{if failure} \end{cases}$$

Then

$$Y_i \overset{indep}{\sim} Bernoulli(\pi(x_i))$$

The mean and variance of Y are

$$E(Y) = P(Y = 1) = \pi(\mathbf{x})$$

$$Var(Y) = \pi(x)(1 - \pi(\mathbf{x}))$$

2.2.1 Linear probability model

2.2.2 Logistic regression model

2.2.3 snoring and heart disease data

from Agresti (2012)

```
(heart <-
  tribble(
    ~snoring, ~yes, ~no,
    "never", 24, 1355,
    "occasional", 35, 603,
    "nearly_every_night", 21, 192,
    "every_night", 30, 224
  ) %>%
  gather(-snoring, key = "disease", value = "freq"))
```

```
# A tibble: 8 x 3
  snoring            disease  freq
  <chr>              <chr>   <dbl>
1 never              yes        24
2 occasional         yes        35
3 nearly_every_night yes        21
4 every_night        yes        30
5 never              no       1355
6 occasional         no        603
7 nearly_every_night no        192
8 every_night        no        224
```

```
snoring_score <- function(x) {
  if (x == "never") {
    x = 0
  } else if (x == "occasional") {
    x = 2
  } else if (x == "nearly_every_night") {
    x = 4
  } else {
    x = 5
  }
  x
}
#----------------------------------------
(heart <-
  heart %>%
  rowwise() %>%
  mutate(snoring = snoring_score(snoring)))
```

```
Source: local data frame [8 x 3]
Groups: <by row>

# A tibble: 8 x 3
  snoring disease  freq
    <dbl> <chr>   <dbl>
1       0 yes        24
2       2 yes        35
3       4 yes        21
4       5 yes        30
5       0 no       1355
6       2 no        603
7       4 no        192
8       5 no        224
```

```
heart_crosstab <-
   heart %>%
   xtabs(freq ~ snoring + disease, data = .)
addmargins(heart_crosstab, margin = 2)
```

```
       disease
snoring   no  yes  Sum
      0 1355   24 1379
      2  603   35  638
      4  192   21  213
      5  224   30  254
```

We now fit probabilities

$$\pi(\mathbf{x}) = P(Y = \text{yes})$$

```
(row_prop <- prop.table(heart_crosstab, margin = 1))
```

```
       disease
snoring     no    yes
      0 0.9826 0.0174
      2 0.9451 0.0549
```

```
      4 0.9014 0.0986
      5 0.8819 0.1181
```

```
(heart_prop <-
  heart %>%
  group_by(snoring) %>%
  mutate(row_margin = sum(freq), prop_yes = freq / row_margin))
```

```
# A tibble: 8 x 5
# Groups:   snoring [4]
  snoring disease  freq row_margin prop_yes
    <dbl> <chr>   <dbl>      <dbl>    <dbl>
1       0 yes        24       1379   0.0174
2       2 yes        35        638   0.0549
3       4 yes        21        213   0.0986
4       5 yes        30        254   0.118
5       0 no       1355       1379   0.983
6       2 no        603        638   0.945
7       4 no        192        213   0.901
8       5 no        224        254   0.882
```

2.2.4 Linear probability model

- xtabs version:

```
addmargins(heart_crosstab, margin = 2) %>%
  as.data.frame.matrix() %>%
  rownames_to_column(var = "snoring") %>%
  mutate(snoring = as.numeric(snoring)) %>%
  glm(yes/Sum ~ snoring, weights = Sum, family = gaussian(link = "identity"), data = .) %>%
  summary()
```

```
Call:
glm(formula = yes/Sum ~ snoring, family = gaussian(link = "identity"),
    data = ., weights = Sum)

Deviance Residuals:
      1        2        3        4
 0.0197  -0.0528   0.0229   0.0167

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.016872   0.001134    14.9  0.00449 **
snoring     0.020038   0.000509    39.3  0.00065 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.00199)

    Null deviance: 3.080311  on 3  degrees of freedom
Residual deviance: 0.003977  on 2  degrees of freedom
AIC: -34.89

Number of Fisher Scoring iterations: 2
```

- `tibble` version:

```
(linprob_fit <-
  heart_prop %>%
  filter(disease == "yes") %>%
  glm(prop_yes ~ snoring, family = gaussian(link = "identity"), data = ., weights = row_margin) %>%
  summary())
```

```
Call:
glm(formula = prop_yes ~ snoring, family = gaussian(link = "identity"),
    data = ., weights = row_margin)

Deviance Residuals:
      1        2        3        4
 0.0197  -0.0528   0.0229   0.0167

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.016872   0.001134    14.9  0.00449 **
snoring     0.020038   0.000509    39.3  0.00065 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.00199)

    Null deviance: 3.080311  on 3  degrees of freedom
Residual deviance: 0.003977  on 2  degrees of freedom
AIC: -34.89

Number of Fisher Scoring iterations: 2
```

2.2.5 Logistic regression model

```
addmargins(heart_crosstab, margin = 2) %>%
  as.data.frame.matrix() %>%
  rownames_to_column(var = "snoring") %>%
  mutate(snoring = as.numeric(snoring)) %>%
  glm(yes/Sum ~ snoring, weights = Sum, family = binomial(link = "logit"), data = .) %>%
  summary()
```

```
Call:
glm(formula = yes/Sum ~ snoring, family = binomial(link = "logit"),
    data = ., weights = Sum)

Deviance Residuals:
     1       2       3       4
-0.835   1.252   0.276  -0.684

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -3.866      0.166  -23.26  < 2e-16 ***
snoring        0.397      0.050    7.95  1.9e-15 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 65.9045  on 3  degrees of freedom
Residual deviance:  2.8089  on 2  degrees of freedom
AIC: 27.06

Number of Fisher Scoring iterations: 4
```

```
logistc_fit <-
  heart_prop %>%
  filter(disease == "yes") %>%
  glm(prop_yes ~ snoring, family = binomial(link = "logit"), data = ., weights = row_margin) %>%
  summary()
```

2.2.6 Probit model

```
addmargins(heart_crosstab, margin = 2) %>%
  as.data.frame.matrix() %>%
  rownames_to_column(var = "snoring") %>%
  mutate(snoring = as.numeric(snoring)) %>%
  glm(yes/Sum ~ snoring, weights = Sum, family = binomial(link = "probit"), data = .) %>%
  summary()
```

```
Call:
glm(formula = yes/Sum ~ snoring, family = binomial(link = "probit"),
    data = ., weights = Sum)

Deviance Residuals:
     1       2       3       4
-0.619   1.039   0.168  -0.618

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.0606     0.0702   -29.4  < 2e-16 ***
snoring       0.1878     0.0235     8.0  1.3e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 65.9045  on 3  degrees of freedom
Residual deviance:  1.8716  on 2  degrees of freedom
AIC: 26.12

Number of Fisher Scoring iterations: 4
```

```
probit_fit <-
  heart_prop %>%
  filter(disease == "yes") %>%
  glm(prop_yes ~ snoring, family = binomial(link = "probit"), data = ., weights = row_margin) %>%
  summary()
```

2.2.7 data on cancer remission

from Agresti (2007)

```
(remission <- read_table("data/remission.dat") %>% na.omit())
```

```
# A tibble: 14 x 3
      LI cases remissions
   <dbl> <dbl>      <dbl>
 1     8     2          0
 2    10     2          0
 3    12     3          0
 4    14     3          0
 5    16     3          0
 6    18     1          1
 7    20     3          2
 8    22     2          1
 9    24     1          0
10    26     1          1
11    28     1          1
12    32     1          0
13    34     1          1
14    38     3          2
```

```
remission <-
  remission %>%
  mutate_if(is.character, as.numeric)
```

- `LI`: labeling index
  - proliferative activity of cells
  - after injection of tritiated thymidine
  - *percentage of cells that are labeled*
- `cases`: the number of cases
- `remissions`: the number of remissions

We want to *determine the characteristics associated with remission in cancer patients*

$$Y = \begin{cases} 1 & \text{if remission } > 0 \\ 0 & \text{if remission } = 0 \end{cases}$$

Denote that each row is observed `cases` times.

```
(remission_fit <-
  remission %>%
  glm(remissions/cases ~ LI, family = binomial(link = "logit"), data = ., weights = cases) %>%
  summary())
```

```
Call:
glm(formula = remissions/cases ~ LI, family = binomial(link = "logit"),
    data = ., weights = cases)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.557  -0.950  -0.570   0.982   1.697

Coefficients:
```

```
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -3.7771     1.3786   -2.74   0.0061 **
LI            0.1449     0.0593    2.44   0.0146 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 23.961  on 13  degrees of freedom
Residual deviance: 15.662  on 12  degrees of freedom
AIC: 24.29

Number of Fisher Scoring iterations: 4
```

2.2.8 `remission` raw data

```
remission_raw <-
  tribble(
    ~LI, ~remissions,
    8, 0,
    8, 0,
    10, 0,
    10, 0,
    12, 0,
    12, 0,
    12, 0,
    14, 0,
    14, 0,
    14, 0,
    16, 0,
    16, 0,
    16, 0,
    18, 1,
    20, 0,
    20, 1,
    20, 1,
    22, 0,
    22, 1,
    24, 0,
    26, 1,
    28, 1,
    32, 0,
    34, 1,
    38, 0,
    38, 1,
    38, 1
  )
```

We can make `remission`:

```
remission_raw %>%
  group_by(LI) %>%
  summarise(cases = n(), remissions = sum(remissions))
```

```
# A tibble: 14 x 3
```

```
      LI cases remissions
   <dbl> <int>      <dbl>
 1     8     2          0
 2    10     2          0
 3    12     3          0
 4    14     3          0
 5    16     3          0
 6    18     1          1
 7    20     3          2
 8    22     2          1
 9    24     1          0
10    26     1          1
11    28     1          1
12    32     1          0
13    34     1          1
14    38     3          2
```

Thus, `glm` for `remissions ~ LI` for this raw data set gives the same result.

```
glm(remissions ~ LI, family = binomial(link = "logit"), data = remission_raw)
```

```

Call:  glm(formula = remissions ~ LI, family = binomial(link = "logit"),
    data = remission_raw)

Coefficients:
(Intercept)           LI
     -3.777        0.145

Degrees of Freedom: 26 Total (i.e. Null);  25 Residual
Null Deviance:      34.4
Residual Deviance: 26.1     AIC: 30.1
```

Chapter 3

Generalized Linear Models for Counts and Rates

```
library(tidyverse)
library(ggfortify)
```

It is natural to assume the count data

$$Y \sim Poisson(\mu)$$

3.1 Poisson Loglinear Models

As mentioned, we build a GLM with

1. The *random component*: $Y \sim Poisson(\mu)$
2. The *systematic component*: $\alpha + \beta_1 x_1 + \cdots + \beta_p x_p$
3. The *link function*: log-link $\ln(\mu)$ which is canonical link for a Poisson GLM

$$\ln \mu(\mathbf{x}) = \alpha + \beta_1 x_1 + \cdots + \beta_p x_p$$

For an interpretation perspective,

$$\mu(\mathbf{x}) = \exp(\alpha + \beta_1 x_1 + \cdots + \beta_p x_p) = e^{\alpha}(e^{\beta_1})^{x_1} \cdots (e^{\beta_p})^{x_p}$$

i.e. μ is multiplied by e^{β_j} as x_j increases by 1-unit. x_j has a *multiplicative impact* of e^{β_j} on μ.

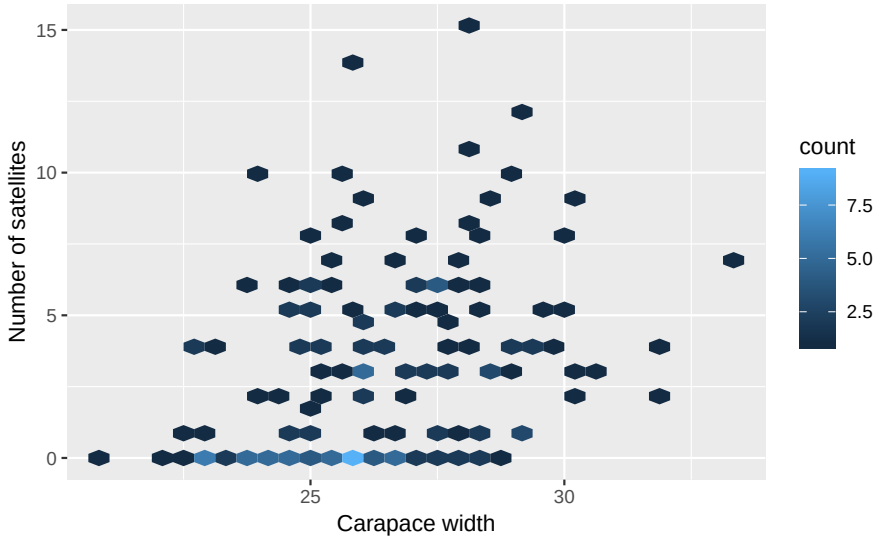
3.2 Horseshoe crab mating data

```
(crab <- read_table("data/Crabs.dat"))
```

```
# A tibble: 173 x 7
    crab   sat     y weight width color spine
   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
 1     1     8     1   3.05  28.3     2     3
 2     2     0     0   1.55  22.5     3     3
 3     3     9     1   2.3   26       1     1
```

```
 4     4     0     0   2.1   24.8     3     3
 5     5     4     1   2.6   26       3     3
 6     6     0     0   2.1   23.8     2     3
 7     7     0     0   2.35  26.5     1     1
 8     8     0     0   1.9   24.7     3     2
 9     9     0     0   1.95  23.7     2     1
10    10     0     0   2.15  25.6     3     3
# ... with 163 more rows
```

```
crab %>%
  ggplot() +
  aes(x = width, y = sat) +
  geom_hex() +
  labs(
    x = "Carapace width",
    y = "Number of satellites"
  )
```



- Large variability is observed.
- Outlier: analysis with this observation and without it

For clarification,

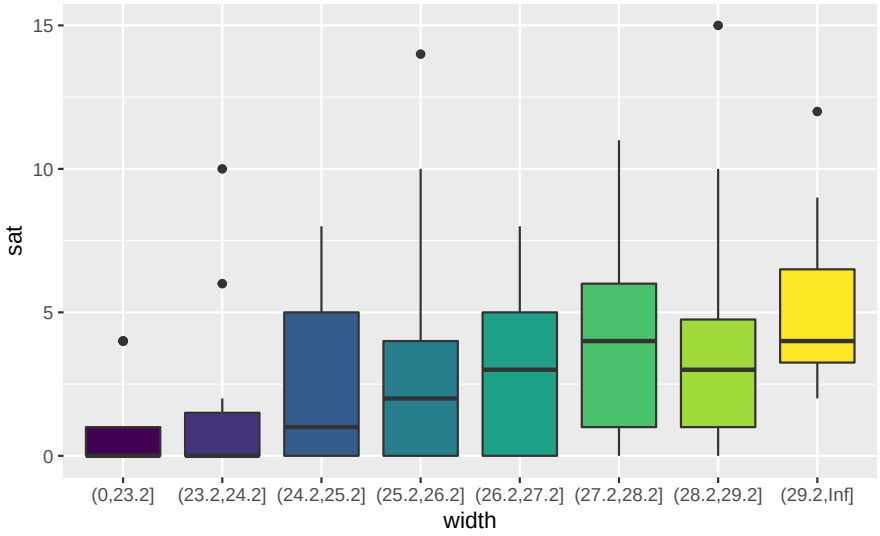
```
(width_interval <-
  crab %>%
  group_by(width = cut(width,
                       breaks = c(0, seq(23.25, 29.25, by = 1), Inf),
                       ordered_result = TRUE)) %>%
  summarise(cases = n(), S = sum(sat), Mean = mean(sat), Variance = var(sat)))
```

```
# A tibble: 8 x 5
  width       cases     S  Mean Variance
  <ord>       <int> <dbl> <dbl>    <dbl>
1 (0,23.2]       14    14  1        2.77
2 (23.2,24.2]    14    20  1.43     8.88
3 (24.2,25.2]    28    67  2.39     6.54
4 (25.2,26.2]    39   105  2.69    11.4
5 (26.2,27.2]    22    63  2.86     6.89
6 (27.2,28.2]    24    93  3.88     8.81
```

```
7 (28.2,29.2]    18    71  3.94    16.9
8 (29.2,Inf]     14    72  5.14     8.29
```

Looking at the table, we can see the *nonlinear relationship* between sateliite counts and width.

```
crab %>%
  group_by(width = cut(width,
                       breaks = c(0, seq(23.25, 29.25, by = 1), Inf),
                       ordered_result = TRUE)) %>%
  ggplot() +
  aes(x = width, y = sat, fill = width) +
  geom_boxplot(show.legend = FALSE)
```



3.2.1 Logistic regression model

Consider binary response

$$Y = \begin{cases} 1 & \text{if satellite } > 0 \\ 0 & \text{if satellite } = 0 \end{cases}$$

```
crab %>%
  select(y, width)
```

```
# A tibble: 173 x 2
       y width
   <dbl> <dbl>
 1     1  28.3
 2     0  22.5
 3     1  26
 4     0  24.8
 5     1  26
 6     0  23.8
 7     0  26.5
 8     0  24.7
 9     0  23.7
10     0  25.6
# ... with 163 more rows
```

For $\pi(x) = P(Y = 1) = \mu$,

$$\text{logit}\pi(x) \equiv \ln \frac{\pi(x)}{1 - \pi(x)} = \alpha + \beta x$$

```
logistic_fit <-
  crab %>%
  glm(y ~ width, family = binomial(link = "logit"), data = .)
summary(logistic_fit)
```

```

Call:
glm(formula = y ~ width, family = binomial(link = "logit"), data = .)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.028  -1.046   0.548   0.907   1.694

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -12.351      2.629   -4.70  2.6e-06 ***
width          0.497      0.102    4.89  1.0e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 225.76  on 172  degrees of freedom
Residual deviance: 194.45  on 171  degrees of freedom
AIC: 198.5

Number of Fisher Scoring iterations: 4
```

Estimated odds of having satellites for each unit change in `width` is multiplied by

$$\frac{\hat{\pi}}{1 - \hat{\pi}} = \exp(\hat{\beta}) = 1.64$$

3.2.2 Poisson regression

Consider count response

$$Y = \text{the number of satellites} \sim Poisson(\mu)$$

```
crab %>%
  select(sat, width)
```

```
# A tibble: 173 x 2
     sat width
   <dbl> <dbl>
 1     8  28.3
 2     0  22.5
 3     9  26
 4     0  24.8
 5     4  26
```

```
 6     0  23.8
 7     0  26.5
 8     0  24.7
 9     0  23.7
10     0  25.6
# ... with 163 more rows
```

```
pois_fit <-
 crab %>%
  glm(sat ~ width, data = ., family = poisson(link = "log"))
summary(pois_fit)
```

```
Call:
glm(formula = sat ~ width, family = poisson(link = "log"), data = .)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.853  -1.988  -0.493   1.097   4.922

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -3.305      0.542   -6.09  1.1e-09 ***
width          0.164      0.020    8.22  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 632.79  on 172  degrees of freedom
Residual deviance: 567.88  on 171  degrees of freedom
AIC: 927.2

Number of Fisher Scoring iterations: 6
```

3.2.3 Goodness-of-fit

Deviance of *exponential family* can be given by

$$\begin{aligned} D(\mathbf{y}, \hat{\boldsymbol{\mu}}) &:= -2(L_M - L_S) \\ &= LRT \quad \text{for } H_0 : M \\ &= 2\sum_i \frac{y_i\tilde{\theta}_i - b(\tilde{\theta}_i)}{a(\phi)} - 2\sum_i \frac{y_i\hat{\theta}_i - b(\hat{\theta}_i)}{a(\phi)} \\ &\text{where} \quad \tilde{\theta} = \text{of saturated, and} \quad \hat{\theta} = \text{of the current model} \end{aligned} \tag{3.1}$$

For $a(\phi) = \frac{\phi}{w_i}$, we compute *scaled deviance*

$$\frac{D(\mathbf{y}, \hat{\boldsymbol{\mu}})}{\phi} \approx \chi^2$$

It can be simplified in the Poisson GLM as

$$D(\mathbf{y}, \hat{\boldsymbol{\mu}}) = 2 \sum_i \ln \frac{y_i}{\hat{\mu}_i}$$

To measure the goodness-of-fit, **analysis of deviance** can be conducted. Comparing two nested models with $\phi = 1$, construct a test

$$M_0 : \text{simpler model} \qquad \text{vs} \quad M_1 : \text{complex model}$$

where M_0 is *nested within* M_1. For each model, we can obtain scaled deviance written as

$$D_0 \equiv D(\mathbf{y}, \hat{\boldsymbol{\mu}}_\mathbf{0}) \leq D(\mathbf{y}, \hat{\boldsymbol{\mu}}_\mathbf{1}) \equiv D_1$$

Then *likelihood-ratio-test statistic* is applicable to the above test structure

$$G^2(M_0 \mid M_1) = D_0 - D_1 \overset{H_0}{\approx} \chi^2(\text{difference of parameters})$$

In case of canonical link GLMs,

$$G^2(M_0 \mid M_1) = 2 \sum_i \hat{\mu}_{1i} \ln \frac{\hat{\mu}_{1i}}{\hat{\mu}_{0i}}$$

```
anova(pois_fit, test = "LRT")
```

```
Analysis of Deviance Table

Model: poisson, link: log

Response: sat

Terms added sequentially (first to last)

      Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                    172        633
width  1     64.9       171        568  7.8e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here,

$$M_0 : \text{null model} \qquad \text{vs} \quad M_1 : \text{only width}$$

For this hypothesis, reject M_0, i.e. the fit of null model is poor compared to the current model. However, the size of residual deviance seems quite large while its degrees of freedom is only `1`.

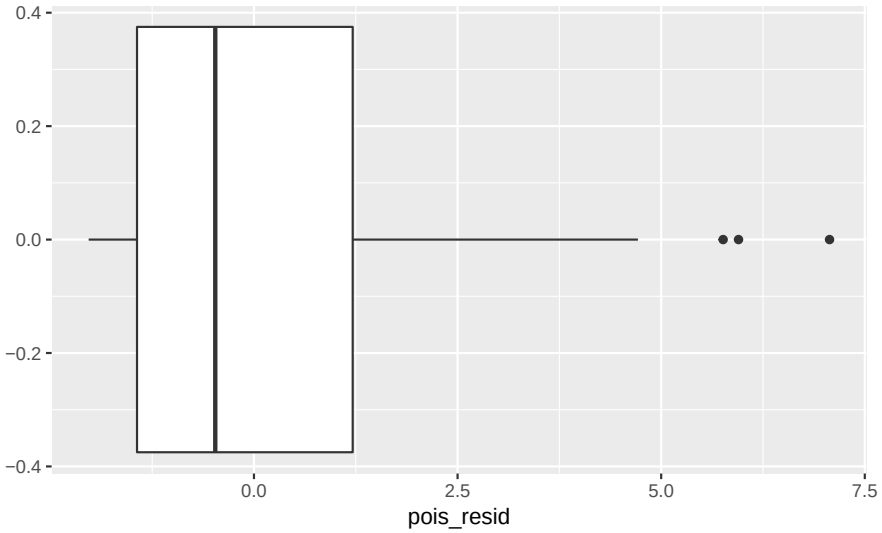
3.2.4 Residual analysis

We now examine residual analysis. In general, *standardized Pearson residual* is prefered.

$$r_i = \frac{y_i - \hat{\mu}_i}{\sqrt{V(\hat{\mu}_i)(1 - \hat{h}_i)}} = \frac{e_i}{\sqrt{V(\hat{\mu}_i)(1 - \hat{h}_i)}}$$

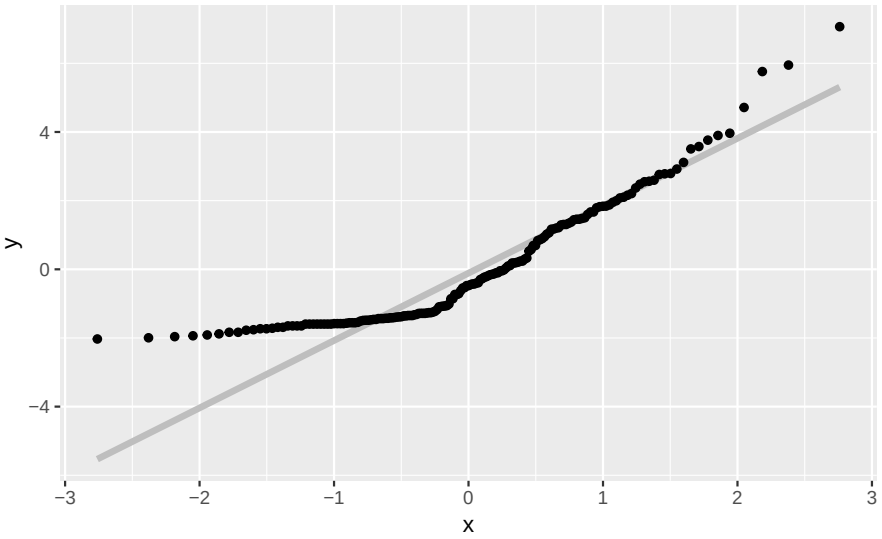
where $\hat{h}_i$ is the hat values.

```
pois_resid <- rstandard(pois_fit, type = "pearson")
tibble(pois_resid) %>%
  ggplot(aes(y = pois_resid)) +
  geom_boxplot() +
  coord_flip()
```



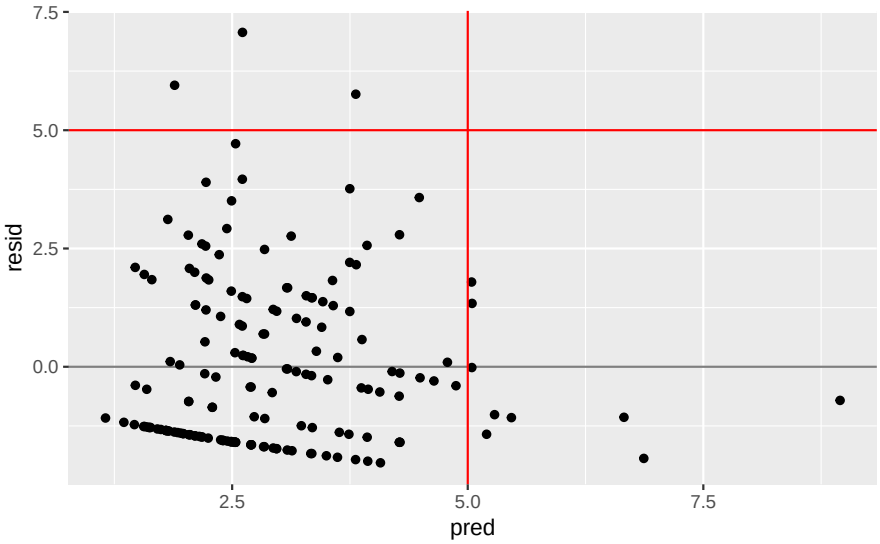
Three large residuals can be observed.

```
tibble(pois_resid) %>%
  ggplot() +
  aes(sample = pois_resid) +
  geom_qq_line(col = "grey", size = 1.5) +
  geom_qq()
```




```
data_frame(pred = predict(pois_fit, type = "response"),
           resid = pois_resid) %>%
  ggplot() +
  aes(pred, resid) +
  geom_hline(yintercept = 0, alpha = .5) +
  geom_jitter() +
```

```
  geom_vline(xintercept = 5, col = "red") +
  geom_hline(yintercept = 5, col = "red")
```



The set of r_i seems variable. It cannot be said to be a good fit.

3.2.5 Overdispersion for Poisson GLMs

```
width_interval
```

```
# A tibble: 8 x 5
  width       cases     S  Mean Variance
  <ord>       <int> <dbl> <dbl>    <dbl>
1 (0,23.2]       14    14  1         2.77
2 (23.2,24.2]    14    20  1.43      8.88
3 (24.2,25.2]    28    67  2.39      6.54
4 (25.2,26.2]    39   105  2.69     11.4
5 (26.2,27.2]    22    63  2.86      6.89
6 (27.2,28.2]    24    93  3.88      8.81
7 (28.2,29.2]    18    71  3.94     16.9
8 (29.2,Inf]     14    72  5.14      8.29
```

Theoretically, for Poisson distribution,

$$E(Y) = Var(Y) = \mu$$

However as we can see, sample variance(`Variance`) is much larger than sample mean(`Mean`). In this data set, `width` is not the only predictor that affects the response. Not only that, but also `weight`, `color`, and `spine` can be in the systematic component. Thus, μ is varied for each combination of (width, weight, color, spine)$^T = \mathbf{x}$. We now have *conditional distribution*

$$Y \mid \mu \sim Poisson(\mu)$$

which gives

$$E(Y \mid \mu) = Var(Y \mid \mu) = \mu$$

Let

$$\theta := E(\mu)$$

Then

$$E(Y) = E\big[E(Y \mid \mu)\big] = E(\mu) = \theta$$

and

$$Var(Y) = E\big[Var(Y \mid \mu)\big] + Var\big[E(Y \mid \mu)\big] = E(\mu) + Var(\mu) > \theta$$

Hence,

$$Var(Y) > E(Y)$$

3.2.6 Negative Binomial GLMs

Negative binomial distribution which also takes into account count response can be a good candidates. Let Y be the negative binomial random variable with parameters μ and $\gamma = \frac{1}{k}$. Then

$$E(Y) = \mu \quad < Var(Y) = \mu + \gamma\mu^2$$

Here, $\gamma > 0$ is called *dispersion parameter*. `MASS` library enables to fit the negative binomial random component and corresponding link functions. There are two ways to fit this random component.

- `MASS::glm.nb()` itself performs what we want, by default `link = log`
  - link function must be specified among: `log`, `sqrt`, or `identity`
  - `init.theta` = dispersion parameter is optional: if omitted, moment estimator by Poisson GLM is used
- Specify `family` option by `MASS::negative.binomial(theta, link)` of base `glm()`
  - *dispersion parameter* `theta` must be chosen

```
crab %>%
  MASS::glm.nb(sat ~ width, data = .,
               link = identity,
               mustart = predict(pois_fit, type = "response"))
```

```
Call:  MASS::glm.nb(formula = sat ~ width, data = ., mustart = predict(pois_fit,
    type = "response"), link = identity, init.theta = 0.9316967133)

Coefficients:
(Intercept)        width
    -11.634        0.554

Degrees of Freedom: 172 Total (i.e. Null);  171 Residual
Null Deviance:      217
Residual Deviance: 196  AIC: 754
```

```
crab %>%
  glm(sat ~ width, data = .,
      family = MASS::negative.binomial(theta = .9, link = "identity"),
      start = coef(pois_fit))
```

```
Call:  glm(formula = sat ~ width, family = MASS::negative.binomial(theta = 0.9,
    link = "identity"), data = ., start = coef(pois_fit))

Coefficients:
(Intercept)        width
    -11.634        0.554

Degrees of Freedom: 172 Total (i.e. Null);  171 Residual
Null Deviance:      212
Residual Deviance: 192  AIC: 752
```

We now implement `log` link which is more typically used.

```
neg_fit <-
  crab %>%
  glm(sat ~ width, data = .,
      family = MASS::negative.binomial(theta = .9, link = "log"))
summary(neg_fit)
```

```
Call:
glm(formula = sat ~ width, family = MASS::negative.binomial(theta = 0.9,
    link = "log"), data = .)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.778  -1.410  -0.250   0.476   2.014

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -4.0534     1.0779   -3.76  0.00023 ***
width         0.1921     0.0405    4.74  4.5e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(0.9) family taken to be 0.843)

    Null deviance: 212.46  on 172  degrees of freedom
Residual deviance: 195.28  on 171  degrees of freedom
AIC: 755.3

Number of Fisher Scoring iterations: 5
```

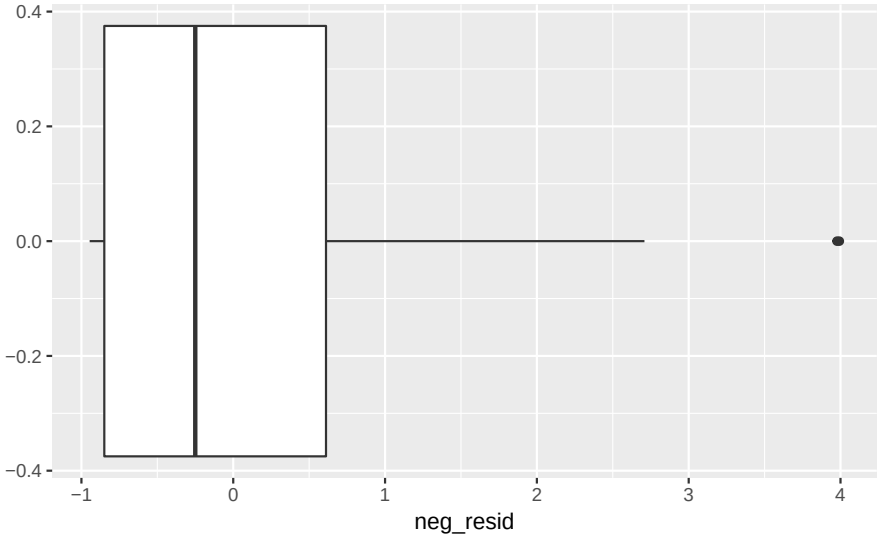
```
anova(neg_fit, test = "LRT")
```

```
Analysis of Deviance Table

Model: Negative Binomial(0.9), link: log

Response: sat

Terms added sequentially (first to last)

      Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                    172        212
```

```
width  1     17.2       171        195  6.4e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

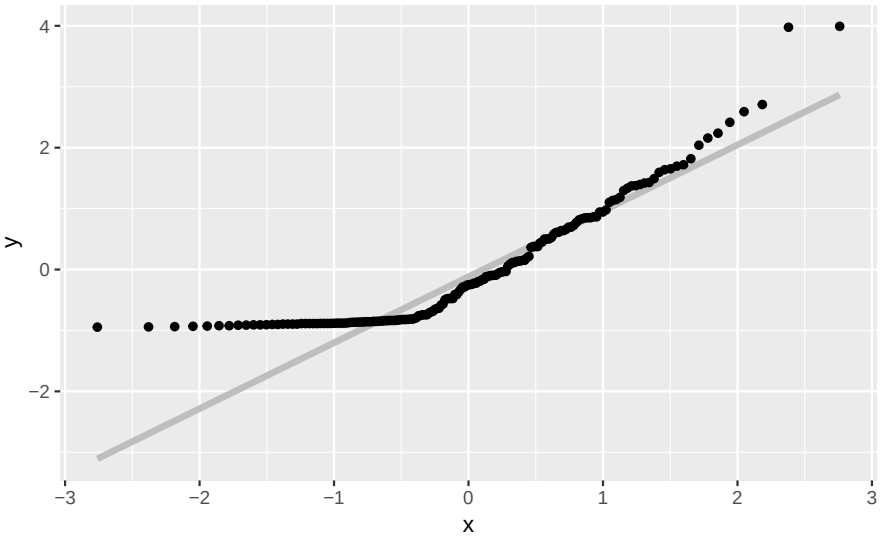
Residual deviance is much smaller than poisson regression.

```
neg_resid <- rstandard(neg_fit, type = "pearson")
tibble(neg_resid) %>%
  ggplot(aes(y = neg_resid)) +
  geom_boxplot() +
  coord_flip()
```

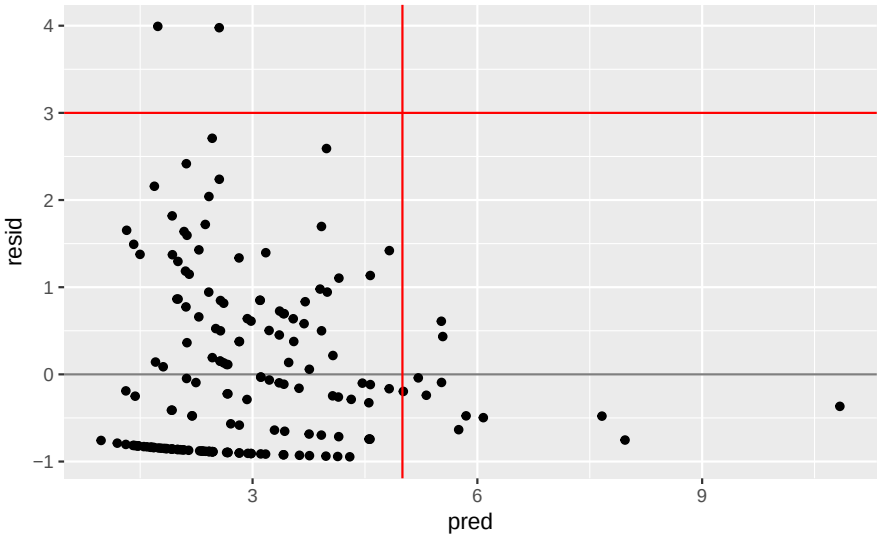


Standardized residuals are not large.

```
tibble(neg_resid) %>%
  ggplot() +
  aes(sample = neg_resid) +
  geom_qq_line(col = "grey", size = 1.5) +
  geom_qq()
```



```
data_frame(pred = predict(neg_fit, type = "response"),
           resid = neg_resid) %>%
  ggplot() +
  aes(pred, resid) +
  geom_hline(yintercept = 0, alpha = .5) +
  geom_jitter() +
  geom_vline(xintercept = 5, col = "red") +
  geom_hline(yintercept = 3, col = "red")
```



Compared to the poisson regression model, this model results in less variable residuals.

Chapter 4

Logistic Regression

```
library(tidyverse)
# library(knitr)
# library(kableExtra)
# library(formattable)
```

4.1 Horseshoe crab data

```
(crab <- read_table("data/Crabs.dat"))
```

```
# A tibble: 173 x 7
    crab   sat     y weight width color spine
   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
 1     1     8     1   3.05  28.3     2     3
 2     2     0     0   1.55  22.5     3     3
 3     3     9     1   2.3   26       1     1
 4     4     0     0   2.1   24.8     3     3
 5     5     4     1   2.6   26       3     3
 6     6     0     0   2.1   23.8     2     3
 7     7     0     0   2.35  26.5     1     1
 8     8     0     0   1.9   24.7     3     2
 9     9     0     0   1.95  23.7     2     1
10    10     0     0   2.15  25.6     3     3
# ... with 163 more rows
```

```
crab %>%
  mutate(
    sat = color_tile("white", "red")(sat),
    y = color_tile("white", "red")(y),
    weight = color_bar("lightblue")(weight),
    width = color_bar("lightgreen")(width),
    color = cell_spec(
      color,
      color = spec_color(color, direction = -1)
    ),
    spine = cell_spec(
      spine,
      color = spec_color(spine)
```

```
    )
  ) %>%
  head() %>%
  kable(format = "latex", escape = FALSE,
        col.names = c("crab", "Satellites", "y", "Weight(kg)", "carapace width(cm)", "Color", "spine co
  kable_styling("hover")
```

$$y_i = \begin{cases} 1 & \text{crab } i \text{ has at least one satellite} \\ 0 & \text{crab } i \text{ does not have satellite} \end{cases}$$

Does the female crab's carapace width is related to this binary response?

Looking at the above data set in the eye, large width can help the crab have satellites. Let's check it out.

```
crab %>%
  ggplot() +
  aes(width, sat) +
  geom_hex() +
  labs(
    x = "Width",
    y = "Number of Satellite"
  )
```

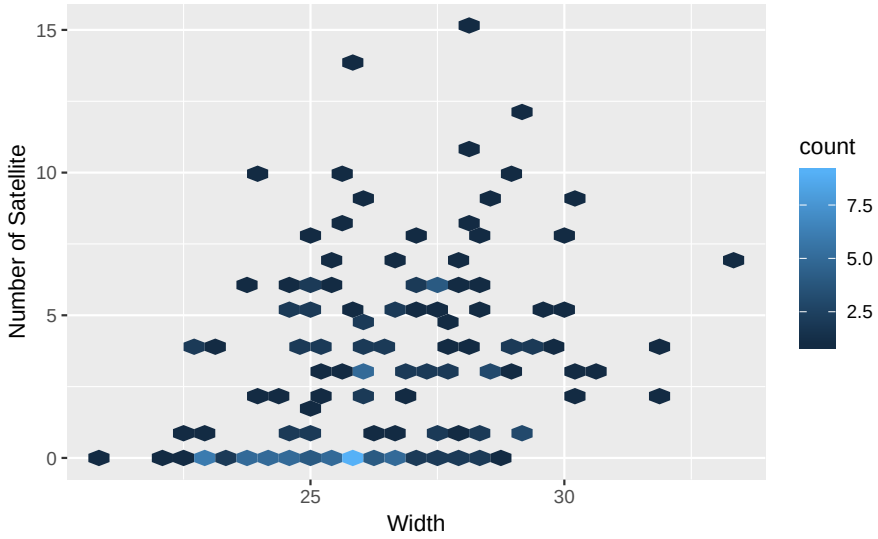


Figure 4.1: Number of satellites by width of female crab

large variability.

```
crab %>%
  group_by(width_cut = cut(width, 8, ordered_result = TRUE)) %>%
  ggplot() +
  aes(width_cut, sat) +
  geom_boxplot() +
  labs(
    x = "Levels of width",
    y = "Number of Satellite"
  )
```

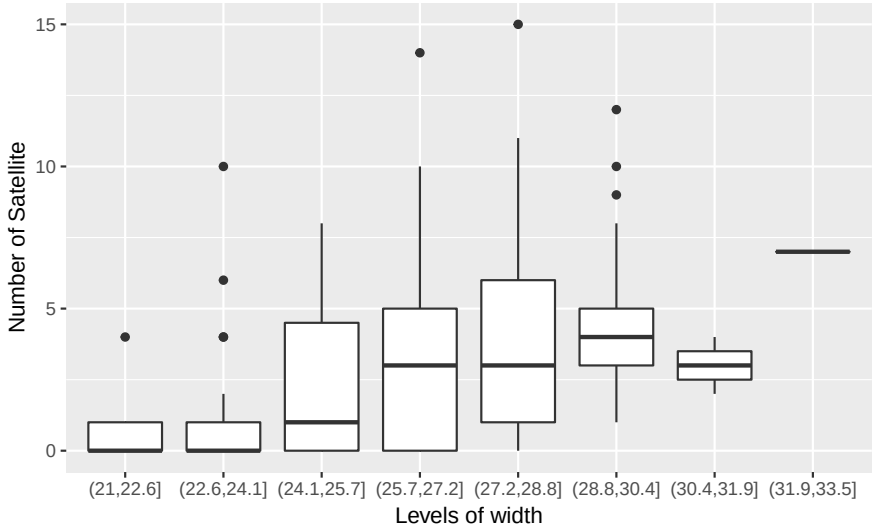


Figure 4.2: Distribution of satellites by width of female crab

4.2 Inference for logistic regression

$$logit[\pi(x)] = \alpha + \beta x$$

```
(width_fit <-
  crab %>%
  select(y, width) %>%
  glm(y ~ ., data = ., family = binomial())) %>%
  summary()
```

```
Call:
glm(formula = y ~ ., family = binomial(), data = .)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.028  -1.046   0.548   0.907   1.694

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -12.351      2.629   -4.70  2.6e-06 ***
width          0.497      0.102    4.89  1.0e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 225.76  on 172  degrees of freedom
Residual deviance: 194.45  on 171  degrees of freedom
AIC: 198.5

Number of Fisher Scoring iterations: 4
```

4.2.1 Wald test

$$Z = \frac{\hat{\beta} - \beta_0}{SE} \overset{H_0}{\approx} N(0,1)$$

Equivalently,

$$Z^2 \overset{H_0}{\approx} \chi^2_1$$

If multivariate,

$$W = (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})^T \left[Cov(\hat{\boldsymbol{\beta}}) \right]^{-1} (\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \overset{H_0}{\approx} \chi^2_p$$

```
broom::tidy(width_fit) %>%
  bind_cols(broom::confint_tidy(width_fit)) %>%
  pander::pander()
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	-12.35	2.629	-4.698	2.622e-06	-17.81	-7.457
width	0.4972	0.1017	4.887	1.021e-06	0.3084	0.709

4.2.2 Likelihood ratio test

$$G^2 = -2(L_0 - L_1)$$

```
(width_lr <- anova(width_fit, test = "LRT"))
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: y

Terms added sequentially (first to last)

      Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                    172        226
width  1     31.3       171        194  2.2e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2.3 Score test

With dispersion of 1, we have

$$Var(Y_i) = V(\mu_i)$$

and so

$$X^2 = \sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)}$$

```
(width_sc <- anova(width_fit, test = "Rao"))
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: y

Terms added sequentially (first to last)

      Df Deviance Resid. Df Resid. Dev  Rao Pr(>Chi)
NULL                    172        226
width  1     31.3       171        194 27.9  1.3e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In sum,

Test	Chi-Square	DF	Pr > ChiSq
LRT	31.31	1	2.204e-08
Score	27.88	1	1.294e-07

4.2.4 Confidence interval for logit

$Cov(\hat{\boldsymbol{\beta}})$ is given as

```
vcov(width_fit)
```

```
            (Intercept)   width
(Intercept)       6.910 -0.2668
width            -0.267  0.0104
```

Then

$$Cov(\hat{\alpha} + \hat{\beta}x_0) = \begin{bmatrix} 1 & x_0 \end{bmatrix} Cov(\hat{\boldsymbol{\beta}}) \begin{bmatrix} 1 \\ x_0 \end{bmatrix} = Var(\hat{\alpha}) + x_0^2 Var(\hat{\beta}) + 2x_0 Cov(\hat{\alpha}, \hat{\beta})$$

For $x_0 = 26.5$, for instance,

```
x0 <- c(1, 26.5)
t(x0) %*% vcov(width_fit) %*% x0
```

```
       [,1]
[1,] 0.0356
```

Then we can calculate

$$(\hat{\alpha} + \hat{\beta}x_0) + z_{\frac{\alpha}{2}} SE$$

On the other hand, `predict.glm(se.fit = TRUE)` gives above value in `$se.fit` as *standard error*, i.e. squared value.

```
data_frame(width = 26.5) %>%
  predict(width_fit, newdata = ., type = "link", se.fit = TRUE)
```

```
$fit
    1
0.826

$se.fit
[1] 0.189

$residual.scale
[1] 1
```

Interpolation:

```
(width_logit <-
  crab %>%
  bind_cols(predict(width_fit, newdata = ., type = "link", se.fit = TRUE) %>% tbl_df()) %>%
  select(sat, width, fit, se.fit) %>%
  mutate(
    lower = fit - se.fit * qnorm(.25, lower.tail = FALSE),
    upper = fit + se.fit * qnorm(.25, lower.tail = FALSE)
  ))
```

```
# A tibble: 173 x 6
     sat width     fit se.fit  lower   upper
   <dbl> <dbl>   <dbl>  <dbl>  <dbl>   <dbl>
 1     8  28.3  1.72    0.310  1.51   1.93
 2     0  22.5 -1.16    0.377 -1.42  -0.909
 3     9  26    0.577   0.175  0.459  0.695
 4     0  24.8 -0.0195  0.201 -0.155  0.116
 5     4  26    0.577   0.175  0.459  0.695
 6     0  23.8 -0.517   0.266 -0.696 -0.337
 7     0  26.5  0.826   0.189  0.699  0.953
 8     0  24.7 -0.0692  0.206 -0.208  0.0696
 9     0  23.7 -0.566   0.274 -0.751 -0.382
10     0  25.6  0.378   0.175  0.260  0.496
# ... with 163 more rows
```

4.2.5 Inverse transformation

Noting that

$$\pi(x_0) = \frac{\exp(logit)}{1 + \exp(logit)}$$

```
width_logit %>%
  transmute(
    sat,
    width,
    lower = exp(lower) / (1 + exp(lower)),
    upper = exp(upper) / (1 + exp(upper))
  )
```

```
# A tibble: 173 x 4
     sat width lower upper
   <dbl> <dbl> <dbl> <dbl>
 1     8  28.3 0.819 0.873
 2     0  22.5 0.195 0.287
```

```
 3     9  26   0.613 0.667
 4     0  24.8 0.461 0.529
 5     4  26   0.613 0.667
 6     0  23.8 0.333 0.417
 7     0  26.5 0.668 0.722
 8     0  24.7 0.448 0.517
 9     0  23.7 0.321 0.406
10     0  25.6 0.565 0.622
# ... with 163 more rows
```

All at once: `type = "response"`

```
predict(width_fit, type = "response", se.fit = TRUE) %>%
  tbl_df() %>%
  mutate(
    lower = fit - se.fit * qnorm(.25, lower.tail = FALSE),
    upper = fit + se.fit * qnorm(.25, lower.tail = FALSE)
  )
```

```
# A tibble: 173 x 5
     fit se.fit residual.scale lower upper
   <dbl>  <dbl>          <dbl> <dbl> <dbl>
 1 0.848 0.0399              1 0.821 0.875
 2 0.238 0.0683              1 0.192 0.284
 3 0.640 0.0404              1 0.613 0.668
 4 0.495 0.0502              1 0.461 0.529
 5 0.640 0.0404              1 0.613 0.668
 6 0.374 0.0623              1 0.332 0.416
 7 0.695 0.0400              1 0.669 0.722
 8 0.483 0.0514              1 0.448 0.517
 9 0.362 0.0633              1 0.319 0.405
10 0.593 0.0422              1 0.565 0.622
# ... with 163 more rows
```

4.3 Goodness of fit

```
anova(width_fit, test = "Chisq")
```

```
Analysis of Deviance Table

Model: binomial, link: logit

Response: y

Terms added sequentially (first to last)

      Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                    172        226
width  1     31.3       171        194  2.2e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Consider more complex models: *quadtratic model with centered predictor*

```
width_comp <-
  crab %>%
  mutate(width = width - mean(width)) %>%
  select(y, width) %>%
  do(
    null_fit = glm(y ~ 1, data = ., family = binomial()),
    center_fit = glm(y ~ ., data = ., family = binomial()),
    quad_fit = glm(y ~ poly(width, 2), data = ., family = binomial())
  )
```

```
(quad_aov <-
  anova(width_comp$null_fit[[1]],
        width_comp$center_fit[[1]],
        width_comp$quad_fit[[1]], test = "LRT"))
```

```
Analysis of Deviance Table

Model 1: y ~ 1
Model 2: y ~ width
Model 3: y ~ poly(width, 2)
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       172        226
2       171        194  1    31.31  2.2e-08 ***
3       170        194  1     0.83     0.36
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since quadratic model has 0.364 of p-value, there is no evidence to support the model.

4.4 Hosmer-Lemeshow goodness of fit

```
MKmisc::HLgof.test(fit = fitted(width_fit), obs = crab$y, ngr = 8)
```

```
$C

    Hosmer-Lemeshow C statistic

data:  fitted(width_fit) and crab$y
X-squared = 6, df = 6, p-value = 0.4


$H

    Hosmer-Lemeshow H statistic

data:  fitted(width_fit) and crab$y
X-squared = 8, df = 6, p-value = 0.2
```

```
ResourceSelection::hoslem.test(crab$y, fitted(width_fit), g = 10)
```

```
    Hosmer and Lemeshow goodness of fit (GOF) test

data:  crab$y, fitted(width_fit)
```

```
X-squared = 4, df = 8, p-value = 0.8
```

Chapter 5

Logit Model for Qualitative Predictors

```
library(tidyverse) # handling data
```

```
aids <- read_table("data/AIDS.dat")
```

```
aids %>% # https://haozhu233.github.io/kableExtra/awesome_table_in_html.html
  group_by(race) %>%
  gather(yes, no, key = symptom, value = count, factor_key = TRUE) %>% # just to consider symptom colum
  mutate_if(is.numeric, function(x) {
    cell_spec(x, bold = TRUE,
              color = spec_color(x, begin = .3, end = .6),
              font_size = spec_font_size(-x, begin = 11))
  }) %>%
  mutate(azt = cell_spec(
    azt, color = "white", bold = TRUE,
    background = spec_color(1:2, begin = .2, end = .7, option = "plasma", direction = 1)
  )) %>%
  spread(symptom, count) %>% # return to the original set
  arrange(desc(race)) %>% # return to the original set
  kable(escape = FALSE, format = "html", row.names = FALSE, booktabs = TRUE,
        col.names = c("Race", "AZT Use", "Yes", "No"),
        caption = "Development of AIDS Symptoms by AZT Use and Race",
        align = "c") %>%
  kable_styling(bootstrap_options = "striped", latex_options = "HOLD_position", full_width = FALSE) %>%
  add_header_above(header = c(" ", " ", "Symptoms" = 2)) %>%
  collapse_rows(columns = 1)
```

Looking at the above table, direct usage of `azt` is likely to result in *slowing the development of AIDS symptoms* (we can check this visually in `Table 1`). Our main interest is to analyze this relationship. To model this, we first define the binary response by

$$Y = \text{symptoms} = \begin{cases} \text{yes} = 1 \\ \text{no} = 0 \end{cases}$$

Denote that the (AZT)`azt` is also categorical predictor.

$$X = \text{AZT} = \begin{cases} \text{yes} \\ \text{no} \end{cases}$$

There is another factor `race` that has possibility to be covariate.

$$Z = \text{Race} = \begin{cases} \text{White} \\ \text{Black} \end{cases}$$

Based on our interest, we need to control the effect of this covariate.

5.1 ANOVA-Type Representation of Factors

5.1.1 One-way ANOVA representation

First consider a signle factor case, with I categories(here, $I = 2$).

```
aids %>%
  select(-race)
```

```
# A tibble: 4 x 3
  azt     yes    no
  <chr> <dbl> <dbl>
1 yes      14    93
2 no       32    81
3 yes      11    52
4 no       12    43
```

For each row i of the table, denote

$$\begin{cases} n_i = \text{yes} + \text{no} \\ y_i = \text{yes} = \text{binomial parameter with } \pi_i \end{cases}$$

Then the model can be specified in ANOVA term.

$$\ln \frac{\pi_i}{1-\pi_i} = \alpha + \beta_i,\ i = 1, 2, \ldots, I \tag{5.1}$$

For redunduncies, we add a constraint. Among the three, we can choose anything. We can set the frist term zero.

$$\beta_1 = 0 \tag{5.2}$$

Similarly, the last term can be set zero. Any other single j-term can be chosen.

$$\beta_I = 0 \tag{5.3}$$

By setting the whole sum as zero, we can guarantee the uniqueness.

$$\sum_i \beta_i = 0 \tag{5.4}$$

5.1.2 Two-way ANOVA representation

```
aids
```

```
# A tibble: 4 x 4
  race  azt     yes    no
  <chr> <chr> <dbl> <dbl>
1 white yes      14    93
2 white no       32    81
3 black yes      11    52
4 black no       12    43
```

$$\ln \frac{\pi_i}{1-\pi_i} = \alpha + \beta_i^X + \beta_k^Z, \quad i = 1, \ldots, I, \; j = 1, \ldots, I$$

constraint to β_i^X and β_i^Z among (5.2) - (5.4). This model induces the relationship between Y and X given Z, i.e. conditional dependence.

5.2 Indicator Variables

ANOVA-type model have presented various restrictions to allow parameters have non-negative degrees of freedom. Recall that in ANOVA, it might be important to construct *orthogonal design*. This leads to the following indicator variables coding for qualitative predictors.

5.2.1 Dummy Coding

From (5.2) or (5.3), we can implement so-called *dummy coding*. For example, (5.3) results in

	x_1	x_2	$\cdots$	x_{I-1}
1	1	0	$\cdots$	0
2	0	1	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
I-1	0	0	$\cdots$	1
I	0	0	$\cdots$	0

```
C(aids$azt %>% factor(levels = c("yes", "no")),
  contr = contr.treatment, base = 2)
```

```
[1] yes no  yes no
attr(,"contrasts")
    1
yes 1
no  0
Levels: yes no
```

Here, dummy coding $\beta_2 = 0$ corresponds to

$$logit\pi = \beta_1 - \beta_2 = \beta_1$$

Thus, the estimate for reference category is the difference in logit (at a fixed level of Z).

On the other hand, when $\beta_1 = 0$ restriction is applied, the default `base = 1` can be used.

```
C(aids$azt %>% factor(levels = c("yes", "no")),
  contr = contr.treatment)
```

```
[1] yes no  yes no
attr(,"contrasts")
    2
yes 0
no  1
Levels: yes no
```

This can be interpreted as

$$logit\pi = \beta_1 - \beta_2 = -\beta_2$$

We might observe that the estimated coefficient will have reversed sign with the above $\beta_2 = 0$ coding.

5.2.2 Effect Coding

From `(4)`, the last value can be coded as `-1`.

	x_1	x_2	$\cdots$	x_{I-1}
1	1	0	$\cdots$	0
2	0	1	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
I-1	0	0	$\cdots$	1
I	-1	-1	$\cdots$	-1

```
C(aids$azt %>% factor(levels = c("yes", "no")),
  contr = contr.sum)
```

```
[1] yes no  yes no
attr(,"contrasts")
    [,1]
yes    1
no    -1
Levels: yes no
```

This corresponds to

$$logit\pi = \beta_1 - \beta_2 = 2\beta_1$$

The log odds ratio becomes twice of dummy coding induced by (5.3). In terms of model parameter estimates, it would be the half of the dummy coding.

5.3 Linear Logit Model for Contingency Tables

5.3.1 Ordering categories

ANOVA-type model (5.2) is invariant to the factor-ordering.

$$logit(\pi_i) = \alpha + \beta x_i \tag{5.5}$$

5.3.2 Long data

To easily fit `glm`, we change the data to long format. We can handle `symptom` in two way. Using `factor` or dichotomous `1-0` numeric. `CRAN documentation` for `binomial()` link function gives those approach.

> As a numerical vector with values between 0 and 1, interpreted as the proportion of successful cases (with the total number of cases given by the weights).

For this, we set development of AIDS as $Y = 1$, otherwise $Y = 0$.

```
aids %>%
  gather(yes, no, key = symptom, value = count) %>%
  mutate(symptom = ifelse(symptom == "yes", 1, 0)) %>%
  mutate_if(is.character, factor) # to apply C() function
```

```
# A tibble: 8 x 4
  race  azt   symptom count
  <fct> <fct>   <dbl> <dbl>
1 white yes         1    14
2 white no          1    32
3 black yes         1    11
4 black no          1    12
5 white yes         0    93
6 white no          0    81
7 black yes         0    52
8 black no          0    43
```

> As a factor: 'success' is interpreted as the factor not having the first level (and hence usually of having the second level).

This means that if we put `no` followed by `yes`, `glm()` will fit $P(AIDS = yes)$.

```
(long_aids <-
  aids %>%
  gather(no, yes, key = symptom, value = count, factor_key = TRUE) %>% # yes after no
  mutate_if(is.character, function(x) {
    factor(x, levels = unique(x)) # levels = same order as in data
  }))
```

```
# A tibble: 8 x 4
  race  azt   symptom count
  <fct> <fct> <fct>   <dbl>
1 white yes   no         93
2 white no    no         81
3 black yes   no         52
4 black no    no         43
5 white yes   yes        14
6 white no    yes        32
7 black yes   yes        11
8 black no    yes        12
```

5.3.3 Dummy Coding induce by (5.2)

By default, `base = 1` is used.

```
C(long_aids$azt,
  contr = contr.treatment, base = 1)
```

```
[1] yes no  yes no  yes no  yes no
```

```
attr(,"contrasts")
    2
yes 0
no  1
Levels: yes no
```

```
C(long_aids$race,
  contr = contr.treatment, base = 1)
```

```
[1] white white black black white white black black
attr(,"contrasts")
      2
white 0
black 1
Levels: white black
```

`factor()` choose its level by an *alphabetical order*. Here we have set the levels manually with `levels` option in the function, which are the same as the order of the data. In `glm()`, there is an argument `contrasts = NULL`. This leads to `contr.treatment(base = 1)`.

$$AZT_{no} = \begin{cases} 0 & \text{if yes} \\ 1 & \text{if no} \end{cases}$$

$$RACE_{black} = \begin{cases} 0 & \text{if white} \\ 1 & \text{if black} \end{cases}$$

```
(dummy_first <-
  long_aids %>%
  glm(symptom ~ azt + race, data = ., weights = count, family = binomial())) %>%
  summary()
```

```
Call:
glm(formula = symptom ~ azt + race, family = binomial(), data = .,
    weights = count)

Deviance Residuals:
    1      2      3      4      5      6      7      8
-5.49  -7.07  -4.00  -5.03   7.29   9.21   6.54   5.73

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.7375     0.2404   -7.23  4.9e-13 ***
aztno         0.7195     0.2790    2.58   0.0099 **
raceblack    -0.0555     0.2886   -0.19   0.8475
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 342.12  on 7  degrees of freedom
Residual deviance: 335.15  on 5  degrees of freedom
AIC: 341.2
```

```
Number of Fisher Scoring iterations: 5
```

```
dummy_first$contrasts
```

```
$azt
[1] "contr.treatment"

$race
[1] "contr.treatment"
```

$$logit(\hat{\pi}) = -1.738 \underset{p-val=0.01}{0.719} AZT_{no} \underset{p-val=0.848}{-0.055} RACE_{black} \tag{5.6}$$

Controlling `raceblack`, we can say that `aztno` significantly affects aids `symptom`.

5.3.4 Dummy Coding induced by (5.3)

By changing the dataset, we can freely implement the other qualitative coding. We now try `contr.treatment(base = 2)`, i.e. setting the last term zero.

```
C(long_aids$azt,
  contr = contr.treatment, base = 2)
```

```
[1] yes no  yes no  yes no  yes no
attr(,"contrasts")
    1
yes 1
no  0
Levels: yes no
```

```
C(long_aids$race,
  contr = contr.treatment, base = 2)
```

```
[1] white white black black white white black black
attr(,"contrasts")
      1
white 1
black 0
Levels: white black
```

$$AZT_{yes} = \begin{cases} 1 & \text{if yes} \\ 0 & \text{if no} \end{cases}$$

$$RACE_{white} = \begin{cases} 1 & \text{if white} \\ 0 & \text{if black} \end{cases}$$

```
make_dummy <- function(x, contr, ...) { # for coefficients' names
  new_x <- C(x, contr = contr, ...)
  attr_x <- attributes(new_x)$contrasts
  colnames(attributes(new_x)$contrasts) <- rownames(attr_x)[attr_x > 0]
  new_x
}
```

Applying `C(contr = contr.treatment, base = 2)`, we get

```
(dummy_last <-
  long_aids %>%
  mutate_at(.vars = vars(azt, race),
            .funs = funs(make_dummy(., contr = contr.treatment, base = 2))) %>%
  glm(symptom ~ azt + race, data = ., weights = count, family = binomial())) %>%
  summary()
```

```

Call:
glm(formula = symptom ~ azt + race, family = binomial(), data = .,
    weights = count)

Deviance Residuals:
    1      2      3      4      5      6      7      8
-5.49  -7.07  -4.00  -5.03   7.29   9.21   6.54   5.73

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.0736     0.2629   -4.08  4.4e-05 ***
aztyes       -0.7195     0.2790   -2.58   0.0099 **
racewhite     0.0555     0.2886    0.19   0.8475
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 342.12  on 7  degrees of freedom
Residual deviance: 335.15  on 5  degrees of freedom
AIC: 341.2

Number of Fisher Scoring iterations: 5
```

```
dummy_last$contrasts
```

```
$azt
    yes
yes   1
no    0

$race
      white
white     1
black     0
```

$$logit(\hat{\pi}) = -1.0736 - \underset{p-val=0.0099}{0.7195} AZT_{yes} + \underset{p-val=0.8476}{0.0555} RACE_{white} \tag{5.7}$$

As mentioned, the absolute values of the estimates are exactly same. The only differences are *their signs* by changing the definition of their variables.

5.3.5 Effect Coding

In the same manner, `contr.sum` adjust $\sum_i \beta_i = 0$ contraints. Or we can specify the `contrasts` argument, e.g. `contrasts = list(azt = "contr.sum", race = "contr.sum")`.

```
C(long_aids$azt,
  contr = contr.sum)
```

```
[1] yes no  yes no  yes no  yes no
attr(,"contrasts")
    [,1]
yes    1
no    -1
Levels: yes no
```

```
C(long_aids$race,
  contr = contr.sum)
```

```
[1] white white black black white white black black
attr(,"contrasts")
      [,1]
white    1
black   -1
Levels: white black
```

$$AZT_1 = \begin{cases} 1 & \text{if yes} \\ -1 & \text{if no} \end{cases}$$

$$RACE_1 = \begin{cases} 1 & \text{if white} \\ -1 & \text{if black} \end{cases}$$

```
(effect_sum <-
  long_aids %>%
  glm(symptom ~ azt + race, data = ., weights = count, family = binomial(),
      contrasts = list(azt = "contr.sum", race = "contr.sum"))) %>%
  summary()
```

```
Call:
glm(formula = symptom ~ azt + race, family = binomial(), data = .,
    weights = count, contrasts = list(azt = "contr.sum", race = "contr.sum"))

Deviance Residuals:
    1      2      3      4      5      6      7      8
-5.49  -7.07  -4.00  -5.03   7.29   9.21   6.54   5.73

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.4056     0.1467   -9.58   <2e-16 ***
azt1         -0.3597     0.1395   -2.58   0.0099 **
race1         0.0277     0.1443    0.19   0.8475
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 342.12  on 7  degrees of freedom
Residual deviance: 335.15  on 5  degrees of freedom
AIC: 341.2
```

```
Number of Fisher Scoring iterations: 5
```

```
effect_sum$contrasts
```

```
$azt
[1] "contr.sum"

$race
[1] "contr.sum"
```

$$logit(\hat{\pi}) = -1.4056 - \underset{p-val=0.01}{0.36} AZT_{effect} - \underset{p-val=0.8476}{0.0277} RACE_{effect} \tag{5.8}$$

```
(Intercept)        azt1       race1
      FALSE        TRUE        TRUE
```

We can see the both β_1 and β_2 of (5.8) are half of the model (5.7).

5.4 Confidence Intervals

`confint()` gives CI computed by profile likelihood, if `MASS` package is installed.

```
lapply(list(First_zero = dummy_first, Last_zero = dummy_last, Sum_zero = effect_sum), confint)
```

```
$First_zero
             2.5 % 97.5 %
(Intercept) -2.232 -1.286
aztno        0.180  1.277
raceblack   -0.633  0.502

$Last_zero
             2.5 % 97.5 %
(Intercept) -1.609 -0.573
aztyes      -1.277 -0.180
racewhite   -0.502  0.633

$Sum_zero
             2.5 %  97.5 %
(Intercept) -1.704 -1.1271
azt1        -0.639 -0.0899
race1       -0.251  0.3167
```

or `confint.default()` gives CI computed by the standard error.

```
lapply(list(First_zero = dummy_first, Last_zero = dummy_last, Sum_zero = effect_sum), confint.default)
```

```
$First_zero
             2.5 % 97.5 %
(Intercept) -2.209  -1.27
aztno        0.173   1.27
raceblack   -0.621   0.51

$Last_zero
            2.5 % 97.5 %
(Intercept) -1.59 -0.558
aztyes      -1.27 -0.173
```

```
racewhite   -0.51  0.621

$Sum_zero
             2.5 %  97.5 %
(Intercept) -1.693 -1.1181
azt1        -0.633 -0.0863
race1       -0.255  0.3106
```

5.5 Fitted values

Using the fitted logit model, we can estimate the expected frequency of contingency table. Before that, estimate each success conditional probabiltiy, i.e. probability of AIDS development by

$$\hat{\pi}_{j|i} = P(Y_i = 1 \mid Z = j) = \frac{\exp(\hat{\alpha} + \hat{\beta}_1 X + \hat{\beta}_2 Z)}{1 + \exp(\hat{\alpha} + \hat{\beta}_1 X + \hat{\beta}_2 Z)}$$

`predict()` for `glm` object gives various values with `type` option.

1. By default, `type = "link"` gives the linear fit on link scale: this is same as `fit$linear.predictors`
2. `type = "response"`: on the scale of the response variable, here gives probabilities on logit we want = `fit$fitted.values`
3. `type = "terms"`: on the scale of linear predictor scale

```
predict(dummy_last, type = "response")
```

```
    1     2     3     4     5     6     7     8
0.150 0.265 0.143 0.255 0.150 0.265 0.143 0.255
```

```
dummy_last$fitted.values
```

```
    1     2     3     4     5     6     7     8
0.150 0.265 0.143 0.255 0.150 0.265 0.143 0.255
```

```
long_aids %>%
  mutate(pred_dummy1 = predict(dummy_last,
                               newdata = data_frame(race = race, azt = azt),
                               type = "response"),
         pred_dummy2 = predict(dummy_last,
                               newdata = data_frame(race = race, azt = azt),
                               type = "response"),
         pred_effect = predict(dummy_last,
                               newdata = data_frame(race = race, azt = azt),
                               type = "response")) %>%
  spread(symptom, count) %>% # return to contingency table format
  select(race, azt, yes, no,
         pred_dummy1, pred_dummy2, pred_effect)
```

```
# A tibble: 4 x 7
  race  azt     yes    no pred_dummy1 pred_dummy2 pred_effect
  <fct> <fct> <dbl> <dbl>       <dbl>       <dbl>       <dbl>
1 white yes      14    93       0.150       0.150       0.150
2 white no       32    81       0.265       0.265       0.265
3 black yes      11    52       0.143       0.143       0.143
4 black no       12    43       0.255       0.255       0.255
```

In fact, coding does not affect goodness-of-fit.

```
(pred_prob <-
  long_aids %>%
  mutate(pred_prob = predict(dummy_last,
                             newdata = data_frame(race = race, azt = azt),
                             type = "response")
         ) %>%
  spread(symptom, count) %>% # return to contingency table format
  select(race, azt, yes, no, pred_prob)) %>%
  gather(yes, no, key = symptom, value = count) %>% # to plot
  ggplot() +
  aes(x = race, y = pred_prob, group = azt) +
  geom_line(aes(colour = azt)) +
  labs(
    x = "Race",
    y = expression(pi),
    parse = TRUE
  )
```

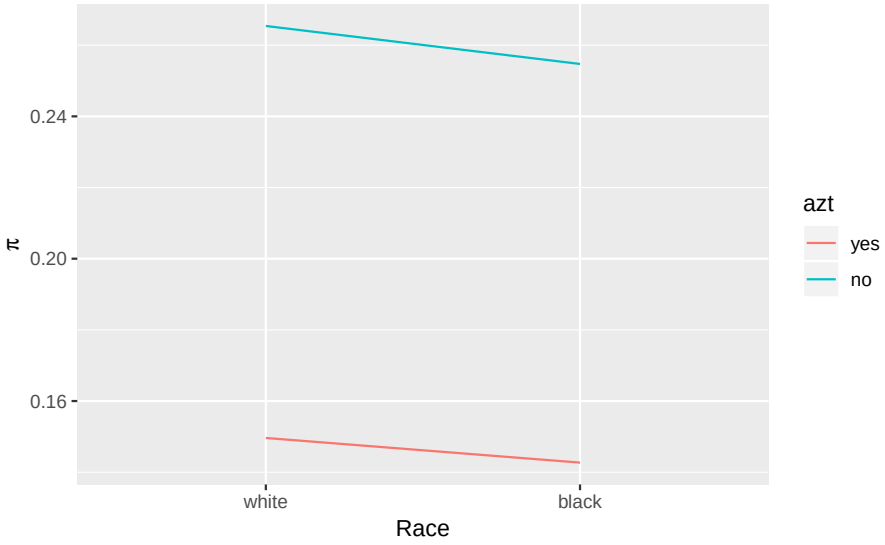


Figure 5.1: Conditional probability of developing AIDS symptoms

Obviously, less probability is likely occur when the patient dose AZT directly. In Figure 5.1, the probability differs between AZT usage, while not between race. Now we can estimate the number of successes by

$$\{\hat{\mu}_{ij} = n_{i+}\hat{\pi}_{j|i}\}$$

```
pred_prob %>%
  mutate(yes_fit = (yes + no) * pred_prob,
         no_fit = (yes + no) * (1 - pred_prob)) %>%
  arrange(desc(race), desc(azt)) %>%
  select(race, azt, yes, yes_fit, no, no_fit, pred_prob)
```

```
# A tibble: 4 x 7
  race  azt     yes yes_fit    no no_fit pred_prob
  <fct> <fct> <dbl>   <dbl> <dbl>  <dbl>     <dbl>
1 black no       12   14.0     43   41.0     0.255
```

```
2 black yes      11    8.99    52   54.0     0.143
3 white no       32   30.0     81   83.0     0.265
4 white yes      14   16.0     93   91.0     0.150
```

It can be estimated as

```
pred_prob %>%
  mutate(yes_fit = (yes + no) * pred_prob %>% round(digits = 1),
         no_fit = (yes + no) * (1 - pred_prob) %>% round(digits = 1)) %>%
  unite("Yes", starts_with("yes"), sep = " vs ") %>%
  unite("No", starts_with("no"), sep = " vs ")
```

```
# A tibble: 4 x 5
  race  azt   Yes         No          pred_prob
  <fct> <fct> <chr>       <chr>           <dbl>
1 white yes   14 vs 10.7  93 vs 96.3      0.150
2 white no    32 vs 33.9  81 vs 79.1      0.265
3 black yes   11 vs 6.3   52 vs 56.7      0.143
4 black no    12 vs 16.5  43 vs 38.5      0.255
```

In sum,

```
pred_prob %>%
  mutate(yes_fit = round((yes + no) * pred_prob, digits = 1),
         no_fit = round((yes + no) * (1 - pred_prob), digits = 1),
         pred_no = round(1 - pred_prob, digits = 3),
         pred_prob = round(pred_prob, digits = 3)) %>%
  select(race, azt, pred_prob, pred_no, yes_fit, no_fit) %>%
  gather(yes_fit, no_fit, key = fit, value = count) %>%
  gather(pred_prob, pred_no, key = prob, value = pred) %>%
  mutate_if(is.numeric, function(x) {
    cell_spec(x, bold = TRUE,
              color = spec_color(-x, end = .9),
              font_size = spec_font_size(x, begin = 10))
  }) %>%
  group_by(race) %>%
  mutate(azt = cell_spec(
    azt, color = "white", bold = TRUE,
    background = spec_color(1:2, begin = .2, end = .7, option = "plasma", direction = 1)
  )) %>%
  ungroup(race) %>%
  spread(prob, pred) %>%
  spread(fit, count) %>%
  select(race, azt, pred_prob, pred_no, yes_fit, no_fit) %>%
  kable(escape = FALSE, format = "latex", row.names = FALSE, booktabs = TRUE,
        col.names = c("Race", "AZT Use", "Yes", "No", "Yes", "No"),
        caption = "Estimated probability and Fitted number",
        align = "c") %>%
  kable_styling(bootstrap_options = "striped", latex_options = "HOLD_position", full_width = FALSE) %>%
  add_header_above(header = c(" ", " ", "Fitted Probability" = 2, "Fitted number" = 2))
```

Appendix

We can also use contingency table form for `glm()`

```
aids %>%
  mutate(Sum = yes + no) %>%
  glm(yes/Sum ~ azt + race, data = ., weights = Sum, family = binomial())
```

```
Call:  glm(formula = yes/Sum ~ azt + race, family = binomial(), data = .,
    weights = Sum)

Coefficients:
(Intercept)       aztyes    racewhite
    -1.0736      -0.7195       0.0555

Degrees of Freedom: 3 Total (i.e. Null);  1 Residual
Null Deviance:      8.35
Residual Deviance: 1.38     AIC: 24.9
```

Chapter 6

Multinomial Responses

```
# wrangling data -------------------------------------
library(tidyverse)
# library(data.table)
# fitting the models ---------------------------------
library(VGAM)
# coloring tables -code will be hidden----------------
library(formattable)
# https://stackoverflow.com/questions/34983822/how-to-have-r-formattable-rendered-to-pdf-output-and-how
export_formattable <- function(f, file, background = "white", delay = 0.2, ...)
{
  w <- as.htmlwidget(f, ...)
  path <- htmltools::html_print(w, background = background, viewer = NULL)
  url <- paste0("file:///", gsub("\\\\", "/", normalizePath(path)))
  webshot::webshot(url,
          file = file,
          selector = ".formattable_widget",
          delay = delay)
}
```

6.1 Nomial Response

Alligator Food Choice

```
(ali <- read_delim("data/Alligators.dat", delim = " "))
```

```
# A tibble: 80 x 5
    lake gender  size  food count
   <dbl>  <dbl> <dbl> <dbl> <dbl>
 1     1      1     1     1     7
 2     1      1     1     2     1
 3     1      1     1     3     0
 4     1      1     1     4     0
 5     1      1     1     5     5
 6     1      1     2     1     4
 7     1      1     2     2     0
 8     1      1     2     3     0
```

```
 9     1      1     2     4     1
10     1      1     2     5     2
# ... with 70 more rows
```

primary food choice of alligators: Fish(1), Invertible(2), Reptile(3), Bird(4), Other(5)

- lake
  - Hancock(1)
  - Oklahoma(2)
  - Trafford(3)
  - George(4)
- gender
  - Male(1)
  - Female(2)
- size
  - <= 2.3 meters long(1)
  - > 2.3 meters long(2)

```
ali <-
  ali %>%
  mutate_at(
    .vars = vars(-count),
    .funs = funs(factor)
  ) %>%
  mutate(food = fct_recode(
    food,
    "fish" = "1",
    "inverebrate" = "2",
    "reptile" = "3",
    "bird" = "4",
    "other" = "5"
  ))
```

Contingency table:

```
(ali_contin <-
  ali %>%
  group_by(lake, gender, size) %>%
  spread(food, count))
```

```
# A tibble: 16 x 8
# Groups:   lake, gender, size [16]
   lake  gender size   fish inverebrate reptile  bird other
   <fct> <fct>  <fct> <dbl>       <dbl>   <dbl> <dbl> <dbl>
 1 1     1      1         7           1       0     0     5
 2 1     1      2         4           0       0     1     2
 3 1     2      1        16           3       2     2     3
 4 1     2      2         3           0       1     2     3
 5 2     1      1         2           2       0     0     1
 6 2     1      2        13           7       6     0     0
 7 2     2      1         3           9       1     0     2
 8 2     2      2         0           1       0     1     0
 9 3     1      1         3           7       1     0     1
10 3     1      2         8           6       6     3     5
11 3     2      1         2           4       1     1     4
12 3     2      2         0           1       0     0     0
13 4     1      1        13          10       0     2     2
```

```
14 4     1      2         9           0       0     1     2
15 4     2      1         3           9       1     0     1
16 4     2      2         8           1       0     0     1
```

6.2 Baseline-category logistic model

In `cbind()`, `vglm()` takes final component as baseline. Here, we take `fish` as baseline: food categories are reversed.

```
(ali_base <-
  ali_contin %>%
  vglm(cbind(other, bird, reptile, inverebrate, fish) ~ lake + size,
       data = ., family = multinomial(),
       contrasts = list(lake = contr.treatment(n = 4, base = 4),
                        size = contr.treatment(n = 2, base = 2)))) %>%
  summary()
```

```
Call:
vglm(formula = cbind(other, bird, reptile, inverebrate, fish) ~
    lake + size, family = multinomial(), data = ., contrasts = list(lake = contr.treatment(n = 4,
    base = 4), size = contr.treatment(n = 2, base = 2)))


Pearson residuals:
                     Min     1Q  Median    3Q  Max
log(mu[,1]/mu[,5]) -1.587 -0.319 -0.0159 1.033 1.41
log(mu[,2]/mu[,5]) -0.987 -0.508 -0.1144 0.237 3.99
log(mu[,3]/mu[,5]) -0.830 -0.585 -0.2309 0.223 2.24
log(mu[,4]/mu[,5]) -1.372 -0.438 -0.0248 0.244 1.99

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept):1 -1.90427    0.52583   -3.62  0.00029 ***
(Intercept):2 -2.09308    0.66223   -3.16  0.00157 **
(Intercept):3 -3.31453    1.05307      NA       NA
(Intercept):4 -1.54902    0.42492   -3.65  0.00027 ***
lake1:1        0.82620    0.55754    1.48  0.13838
lake1:2        0.69512    0.78126    0.89  0.37361
lake1:3        1.24278    1.18542    1.05  0.29446
lake1:4       -1.65836    0.61288   -2.71  0.00681 **
lake2:1        0.00565    0.77657    0.01  0.99419
lake2:2       -0.65321    1.20192   -0.54  0.58681
lake2:3        2.45887    1.11811    2.20  0.02787 *
lake2:4        0.93722    0.47191    1.99  0.04703 *
lake3:1        1.51637    0.62143    2.44  0.01468 *
lake3:2        1.08777    0.84167    1.29  0.19622
lake3:3        2.93525    1.11639    2.63  0.00856 **
lake3:4        1.12198    0.49051    2.29  0.02217 *
size1:1        0.33155    0.44825    0.74  0.45951
size1:2       -0.63066    0.64247   -0.98  0.32629
size1:3       -0.35126    0.58003   -0.61  0.54479
size1:4        1.45820    0.39594    3.68  0.00023 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors:  4

Names of linear predictors:
log(mu[,1]/mu[,5]), log(mu[,2]/mu[,5]), log(mu[,3]/mu[,5]), log(mu[,4]/mu[,5])

Residual deviance: 52.5 on 44 degrees of freedom

Log-likelihood: -74.4 on 44 degrees of freedom

Number of iterations: 5

Warning: Hauck-Donner effect detected in the following estimate(s):
'(Intercept):3'

Reference group is level  5  of the response
```

6.2.1 Goodness of fit

$$H_0 : \boldsymbol{\beta} = 0$$

```
ali_basegood <-
  ali_contin %>%
  ungroup() %>%
  do(
    null_fit = vglm(cbind(other, bird, reptile, inverebrate, fish) ~ 1,
                    data = ., family = multinomial()),
    gender_fit = vglm(cbind(other, bird, reptile, inverebrate, fish) ~ gender,
                    data = ., family = multinomial(),
                    contrasts = list(gender = contr.treatment(n = 2, base = 2))),
    size_fit = vglm(cbind(other, bird, reptile, inverebrate, fish) ~ size,
                    data = ., family = multinomial(),
                    contrasts = list(size = contr.treatment(n = 2, base = 2))),
    lake_fit = vglm(cbind(other, bird, reptile, inverebrate, fish) ~ lake,
                    data = ., family = multinomial(),
                    contrasts = list(lake = contr.treatment(n = 4, base = 4))),
    add_fit = vglm(cbind(other, bird, reptile, inverebrate, fish) ~ lake + size,
                    data = ., family = multinomial(),
                   contrasts = list(lake = contr.treatment(n = 4, base = 4),
                                    size = contr.treatment(n = 2, base = 2))),
    full_fit = vglm(cbind(other, bird, reptile, inverebrate, fish) ~ gender + lake + size,
                    data = ., family = multinomial(),
                    contrasts = list(lake = contr.treatment(n = 4, base = 4),
                                     size = contr.treatment(n = 2, base = 2),
                                     gender = contr.treatment(n = 2, base = 2)))
  )
```

```
each_mod <- function(x, test = "LRT", ...) {
  mod_name <-
    as.character(x[[1]]@call)[2] %>%
    str_extract(pattern = "(?<=~).*") %>%
    str_trim()
  # mod_aov <- broom::tidy(anova(x[[1]], type = 1, test = "LRT"))
  # mod_aov %>%
```

```
  #   add_column(model = rep(mod_name, nrow(mod_aov)), .before = 1)
  broom::tidy(anova(x[[1]], type = 1, test = test, ...)) %>%
    slice(n()) %>%
    add_column(model = mod_name, .before = 1)
}
#-----------
(ali_good <-
  ali_basegood %>%
  map(each_mod, test = "LRT") %>%
  bind_rows()) %>%
  pander::pander()
```

model	term	df	Deviance	Resid..Df	Resid..Dev	p.value
1	NULL			60	116.8	
gender	gender	4	2.104	56	114.7	0.7166
size	size	4	15.15	56	101.6	0.004401
lake	lake	12	43.2	48	73.57	2.092e-05
lake + size	size	4	21.09	44	52.48	0.0003043
gender + lake + size	size	4	17.6	40	50.26	0.001477

Each differene between next row represents each

$$G^2\Big[\text{simple} \mid \text{complex}\Big]$$

For example,

$$G^2\Big[(L+S) \mid (G+L+S)\Big] = 52.478 - 50.264 = 2.215$$

6.3 Estimating probabilities

$$\pi_j(\mathbf{x}) = \frac{\exp(\alpha_j + \boldsymbol{\beta}_j^T \mathbf{x})}{1 + \sum_{h=1}^{J-1} \exp(\alpha_h + \boldsymbol{\beta}_h^T \mathbf{x})}$$

```
(ali_pred <-
  ali_contin %>%
  ungroup() %>%
  select(lake, size) %>%
  bind_cols(predict(ali_base, newdata = ., type = "response") %>% tbl_df()))
```

```
# A tibble: 16 x 7
   lake  size   other    bird reptile inverebrate  fish
   <fct> <fct>  <dbl>   <dbl>   <dbl>       <dbl> <dbl>
 1 1     1     0.254  0.0704   0.0475      0.0931 0.535
 2 1     2     0.194  0.141    0.0718      0.0231 0.570
 3 1     1     0.254  0.0704   0.0475      0.0931 0.535
 4 1     2     0.194  0.141    0.0718      0.0231 0.570
 5 2     1     0.0539 0.00882  0.0772      0.602  0.258
 6 2     2     0.0687 0.0294   0.195       0.249  0.458
 7 2     1     0.0539 0.00882  0.0772      0.602  0.258
```

```
 8 2     2     0.0687 0.0294   0.195       0.249  0.458
 9 3     1     0.174  0.0359   0.0888      0.517  0.184
10 3     2     0.201  0.108    0.202       0.193  0.296
11 3     1     0.174  0.0359   0.0888      0.517  0.184
12 3     2     0.201  0.108    0.202       0.193  0.296
13 4     1     0.0938 0.0297   0.0116      0.413  0.452
14 4     2     0.0979 0.0811   0.0239      0.140  0.657
15 4     1     0.0938 0.0297   0.0116      0.413  0.452
16 4     2     0.0979 0.0811   0.0239      0.140  0.657
```

6.4 Cumulative Logits

Happiness and Traumatic Events

```
gss <- read_delim("data/happiness.dat", delim = " ")
```

Race	Happiness	Average number of Traumatic Events
0	1	1.14
0	2	1.55
0	3	2.67
1	2	1.67
1	3	1.50

- There are two explanatory variables in this dataset
  - total number of traumatic accidents: x_1
  - whether they are Caucasian or African American: (x_2, 1=black, 0=white)
- these variables are used to predict the degree of happiness, which is multinomial response measured on three point scale: 1=very happy, 2=pretty happy, 3=not too happy.

With this dataset, *we want to figure out how to estimate response variable according to category and what explanatory factors influence on the response.* Look at the data. In case of white people, the more traumatic events they have had, the less happy they are. On the other hands, this relationships seems vague for black people.

Cumulative logit model is proper tool to analyze this data because response variable is ordinal response with categorical scales.

6.4.1 Cumulative Probability

Consider category ordering.

$$P(Y \le j \mid \mathbf{x}) = \pi_1(\mathbf{x}) + \cdots + \pi_j(\mathbf{x}), \quad j = 1, \ldots, J \tag{6.1}$$

6.4.2 Cumulative Logits

Define logits for the cumulative probablities (6.1) as in the other settings.

$$\begin{aligned} logitP(Y \le j \mid \mathbf{x}) &= \ln \frac{P(Y \le j \mid \mathbf{x})}{1 - P(Y \le j \mid \mathbf{x})} \\ &= \ln \frac{\pi_1(\mathbf{x}) + \cdots \pi_j(\mathbf{x})}{\pi_{j+1}(\mathbf{x}) + \pi_J(\mathbf{x})}, \quad j = 1, \ldots J \end{aligned} \tag{6.2}$$

This is an ordinary logit for a binary response in which categories 1 to j from a single category $j+1$ to J from the second category.

6.4.3 Proportional Odds Property

Cumulative logit (6.2) can be modeled as GLM. Each $logitP(Y \le j)$ becomes an ordinary logistic model for a binary response, i.e. $J-1$ model with last one redundant category.

$$L_j(\mathbf{x}) := logitP(Y \le j \mid \mathbf{x}) = \alpha_j + \boldsymbol{\beta}^T \mathbf{x}, \quad j = 1, \ldots J-1 \tag{6.3}$$

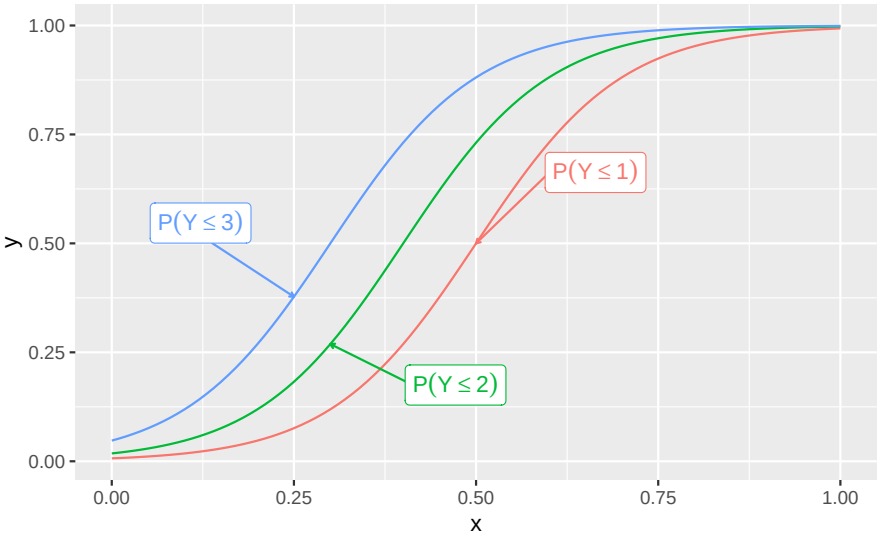


Figure 6.1: Cumulative Logit for each three probabilities in a four-category case

Althogh each logit has its own model, every logit shares the coefficient $\boldsymbol{\beta}^T$, i.e. *parsimonious*. The only difference between each model is intercept terms. This setting is quite natural. See Figure 6.1. Since the same effect is assumed for the models, the shapes of the fitted logit lines are same. It is moving horizontaly so that it would never catch up with the next line. If they were not of same shape by modeling $\boldsymbol{\beta}_j^T$, the lines would cross each other. This contradicts to the construction of cumulative probability, $P(Y \le j \mid \mathbf{x})$ increases in j for fixed $\mathbf{x}$. This is why the model assumes $\forall j : \boldsymbol{\beta}_j^T = \boldsymbol{\beta}^T$.

$$\begin{aligned} L_j(\mathbf{x}_1) - L_j(\mathbf{x}_2) &= logitP(Y \le j \mid \mathbf{x}_1) - logitP(Y \le j \mid \mathbf{x}_2) \\ &= \ln \frac{P(Y \le j \mid \mathbf{x}_1)/P(Y > j \mid \mathbf{x}_1)}{P(Y \le j \mid \mathbf{x}_2)/P(Y > j \mid \mathbf{x}_2)} \\ &\overset{(6.3)}{=} \boldsymbol{\beta}^T(\mathbf{x}_1 - \mathbf{x}_2) \end{aligned} \tag{6.4}$$

The difference between log-odds of $\le j$ at $\mathbf{x}_1$ and of $\mathbf{x}_2$ is $\boldsymbol{\beta}^T(\mathbf{x}_1 - \mathbf{x}_2)$, i.e. it is *proportional to the distance between the two*. Since the proportionality constant applies to each logit, the model is called *proportional odds model*. Also, an odds ratio of cumulative probailities is defined by *cumulative odds ratio*.

$$\text{odds of making response } \leq j \text{ at } \mathbf{x} = \mathbf{x}_1 = \exp\left[\boldsymbol{\beta}^T(\mathbf{x}_1 - \mathbf{x}_2)\right] \tag{6.5}$$

Consider univariate case. (6.5) implies the cumulative odds ratio equals e^{β} which is the *constant cumulative odds ratio whenever* $x_1 - x_2 = 1$.

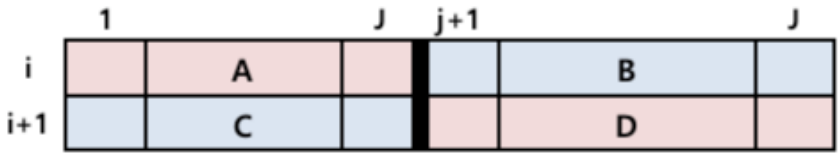


Figure 6.2: Uniform odds ratios AD/BC

Figure 6.2 illustrates the uniform odds ratio $\frac{AD}{BC} = \frac{A/B}{C/D}$ for all pair of adjustment rows and all response cut-point for the cumulative logit Uniform association model.

6.4.4 Relationship between Y and x

See Figure 6.1. Y tends to be smaller at the higher values of x_i. This can be less intuitive, so somtimes we reparameterize the model (6.3) using $-\boldsymbol{\beta}$ (McCullagh and Nelder (1989)).

$$L_j(\mathbf{x}) = \alpha_j - \boldsymbol{\beta}^T\mathbf{x}, \quad j = 1, \ldots J-1 \tag{6.6}$$

In this model, Y tends to be large at higher values of $\mathbf{x}_i$.

Or we can just *recode higher level to be smaller value*, like in the data we are looking at.

6.4.5 Inference

```
(gss_contin <-
  gss %>%
  mutate(happy = fct_recode(happy, "very" = "1", "pretty" = "2", "not" = "3")) %>%
  group_by(race, trauma, happy) %>%
  summarise(N = n()) %>%
  spread(happy, N, fill = 0))
```

```
# A tibble: 10 x 5
# Groups:   race, trauma [10]
   race  trauma  very pretty   not
   <fct>  <dbl> <dbl>  <dbl> <dbl>
 1 0          0     7     15     1
 2 0          1     8     12     1
 3 0          2     5     16     1
 4 0          3     1      9     1
 5 0          4     1      4     0
 6 0          5     0      0     2
 7 1          0     0      1     1
 8 1          1     0      3     1
 9 1          2     0      3     1
10 1          3     0      2     1
```

6.4.6 Baseline-category logit model

We first try to fit model for nominal response, *multinomial logistic regression*. Set baseline as `happy` $= 3$.

$$\ln\left(\frac{\pi_j}{\pi_3}\right) = \alpha_j + \boldsymbol{\beta}_j^T \mathbf{x}, \quad j = 1, 2$$

bcl-model

```
(fit_baseline <-
  gss_contin %>%
  vglm(cbind(very, pretty, not) ~ race + trauma, data = ., family = multinomial)) %>%  #<<
  summary()
```

```
Call:
vglm(formula = cbind(very, pretty, not) ~ race + trauma, family = multinomial,
    data = .)


Pearson residuals:
                      Min     1Q   Median    3Q   Max
log(mu[,1]/mu[,3]) -1.06 -0.350 8.47e-06 0.206 0.820
log(mu[,2]/mu[,3]) -2.15 -0.473 1.87e-01 0.439 0.729

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept):1    2.556      0.820    3.12   0.0018 **
(Intercept):2    3.087      0.775    3.98  6.8e-05 ***
race1:1        -19.583   2662.468      NA       NA
race1:2         -1.543      0.766   -2.01   0.0440 *
trauma:1        -0.730      0.333   -2.19   0.0283 *
trauma:2        -0.432      0.279   -1.55   0.1221
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors:  2

Names of linear predictors: log(mu[,1]/mu[,3]), log(mu[,2]/mu[,3])

Residual deviance: 11.2 on 14 degrees of freedom

Log-likelihood: -19.6 on 14 degrees of freedom

Number of iterations: 17

Warning: Hauck-Donner effect detected in the following estimate(s):
'race1:1'

Reference group is level  3  of the response
```

Observe that this model nees to estimate 6 pameters. Consider the test

$$M_0 : \text{without trauma variables} \Leftrightarrow \cdots = \beta_{j6} = 0 \qquad \text{vs} \qquad M_1 : \text{this model}$$

bcl-goodness

```
# lrtest(vglm(happy ~ race, data = gss, family = multinomial), fit_baseline)
(aov_dev <-
  gss_contin %>%
  vglm(cbind(very, pretty, not) ~ race, data = ., family = multinomial) %>% # without trauma
  anova(fit_baseline, type = 1, test = "LRT"))
```

```
Analysis of Deviance Table

Model 1: cbind(very, pretty, not) ~ race
Model 2: cbind(very, pretty, not) ~ race + trauma
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        16       16.4
2        14       11.2  2     5.21    0.074 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

$$G^2(M_0 \mid M_1) = \text{difference of deviance} = 5.205 \overset{M_0}{\approx} \chi^2(df = 2)$$

Since the p-value is 0.074, the model with `trauma` does not explain the data set well. This might be due to the missing *ordinal information*.

6.4.7 Proportional odds model

Going back to the topic, we now fit *cumulative logit model* (6.3). It predicts the cumulative probability of a certain level of response on an ordinal scale. In other words, the purpose of analysis is to analyze how the ordinal response is predicted by explanatory variables.

`parallel = TRUE` of `family = cumulative()` link is implemented to assume proportional odds. To fit the other form of model (6.6) of McCullagh and Nelder (1989), `family = propodds()` can also be considered.

prop-model

```
(fit_cumul <-
  gss_contin %>%
  vglm(cbind(very, pretty, not) ~ race + trauma, data = ., family = cumulative(parallel = TRUE))) %>%
  summary()
```

```
Call:
vglm(formula = cbind(very, pretty, not) ~ race + trauma, family = cumulative(parallel = TRUE),
    data = .)


Pearson residuals:
                  Min     1Q Median     3Q   Max
logit(P[Y<=1]) -0.658 -0.446 -0.309 0.0671 0.992
logit(P[Y<=2]) -2.799 -0.221  0.158 0.4730 0.874

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept):1   -0.518      0.338   -1.53   0.1255
(Intercept):2    3.401      0.565    6.02  1.7e-09 ***
race1           -2.036      0.691   -2.95   0.0032 **
trauma          -0.406      0.181   -2.24   0.0249 *
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors:  2

Names of linear predictors: logit(P[Y<=1]), logit(P[Y<=2])

Residual deviance: 12.7 on 16 degrees of freedom

Log-likelihood: -20.3 on 16 degrees of freedom

Number of iterations: 5

No Hauck-Donner effect found in any of the estimates

Exponentiated coefficients:
 race1 trauma
 0.131  0.667
```

6.4.8 Extra power

Compared to bcl-model, every effect in cummulative logit model is significant.

prop-goodness

```
(aov_cumul <-
  gss_contin %>%
  vglm(cbind(very, pretty, not) ~ race, data = ., family = cumulative(parallel = TRUE)) %>% # without t
  anova(fit_cumul, type = 1, test = "LRT"))
```

```
Analysis of Deviance Table

Model 1: cbind(very, pretty, not) ~ race
Model 2: cbind(very, pretty, not) ~ race + trauma
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        17       17.8
2        16       12.7  1     5.07    0.024 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Looking at the goodness-of-fit test versus non-`trauma` model.

$$G^2(M_0 \mid M_1) = 5.068 \overset{M_0}{\approx} \chi^2(df = 1)$$

We can see that the degrees of freedom is less than of baseline-category model. This gives the test more power.

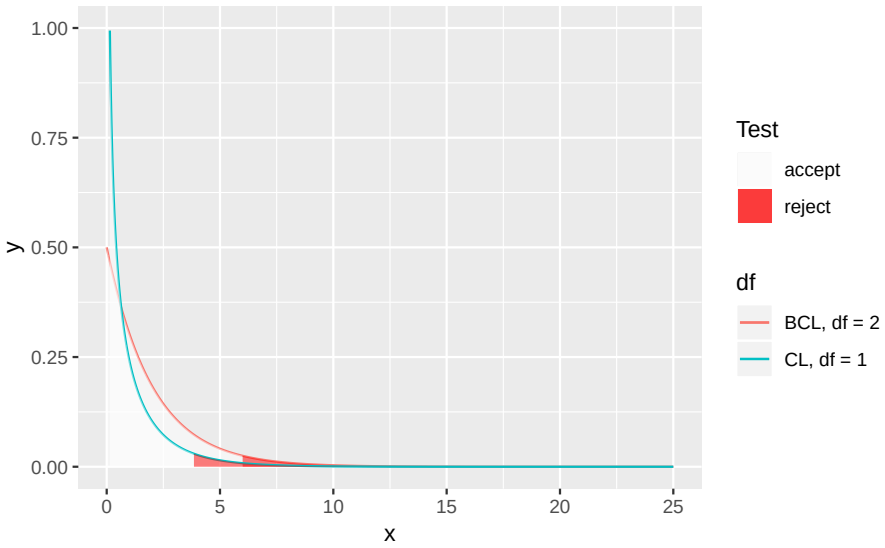


Figure 6.3: Benefits of utilizing the ordinality

For same observed statistic, χ^2 distribution with small df can reject the test more easily thanks to its narrow tail. We can see with the eye in Figure 6.3.

6.4.9 Residual degrees of freedom

Saturated Model	Model
$n(J-1)=20$	$n(J-1)-p$

$$\text{residual df} = \begin{cases} 14 & \text{for baseline-category model} \\ 16 & \text{for cumulative logit model} \end{cases}$$

6.4.10 Inference concerning cumulative logits

As other GLMs, cumulative logits can use *Wald test statistic* or *likelihood test statistic*.

```
summary(fit_cumul, lrt0 = TRUE)
```

```
Call:
vglm(formula = cbind(very, pretty, not) ~ race + trauma, family = cumulative(parallel = TRUE),
    data = .)


Pearson residuals:
                  Min     1Q Median     3Q   Max
logit(P[Y<=1]) -0.658 -0.446 -0.309 0.0671 0.992
logit(P[Y<=2]) -2.799 -0.221  0.158 0.4730 0.874

Likelihood ratio test coefficients:
       Estimate z value Pr(>|z|)
race1    -2.036   -3.04   0.0024 **
trauma   -0.406   -2.25   0.0244 *
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors:  2

Names of linear predictors: logit(P[Y<=1]), logit(P[Y<=2])

Residual deviance: 12.7 on 16 degrees of freedom

Log-likelihood: -20.3 on 16 degrees of freedom

Number of iterations: 5

Exponentiated coefficients:
 race1 trauma
 0.131  0.667
```

We have already seen these in prop-model, the model is estimated as

$$\begin{cases} logitP(Y \leq 1 \mid \mathbf{x}) = -0.518 + \underset{p=0.003}{-2.036}\text{race1} + \underset{p=0.003}{-0.406}\text{trauma} \\ logitP(Y \leq 2 \mid \mathbf{x}) = 3.401 + \underset{p=0.003}{-2.036}\text{race1} + \underset{p=0.003}{-0.406}\text{trauma} \end{cases}$$

The first equation is, when response variable is `very happy` and the Second logit model is the one when response variable is `pretty happy`. Coefficient estimates for explanatory variables(race and trauma) has less than 0.05 p-value, thus it is reasonable to use the parameters.

Denote that both effects are negative. $\hat{\beta}_1 = -0.406$ suggest that the subject is not happy as she had have more and more traumatic events. $\hat{\beta}_1 = -2.036$ indicates the blacks might be less happy compared to the whites. This $race1$ variable shows lot difference. Given the number of traumatic events, the estimated odds for feeling very happy of observations in the white category is $e^{race1} = 0.131$ times of observations in the black category. **?** states that this estimates might be imprecise because these two categories are too imblanced.

```
# A tibble: 2 x 2
  race      N
  <fct> <int>
1 0        84
2 1        13
```

This is reflected as wide confidence interval.

```
confint(fit_cumul, method = "profile")
```

```
               2.5 % 97.5 %
(Intercept):1 -1.202  0.139
(Intercept):2  2.378  4.627
race1         -3.429 -0.716
trauma        -0.773 -0.052
```

By construction, if we change the ordering reversely, the signs will be changed, either.

6.4.11 Checking the proportional odds assumption

Modeling each $\boldsymbol{\beta}_j$ might fit better than single $\boldsymbol{\beta}$. As mentioned, however, this results in non-parallelism of curves for different cumulative probabilites and makes them cross. Moreover, proportional odds model is

simple to be summarized in terms of *parsimony principle*. Conducting *score test* or *likelihood ratio test* for the nonparallel model helps us to choose between parallel or non-parallel models.

```
(beta_check <-
  gss_contin %>%
  vglm(cbind(very, pretty, not) ~ ., data = ., family = cumulative(parallel = FALSE)) %>%
  anova(fit_cumul, type = 1, test = "LRT"))
```

```
Analysis of Deviance Table

Model 1: cbind(very, pretty, not) ~ .
Model 2: cbind(very, pretty, not) ~ race + trauma
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        14       11.3
2        16       12.7 -2    -1.41     0.49
```

p-value is 0.494. Thus, we can say that the proportional odds assumption is acceptable.

6.4.12 Nonparallelism

In some situation, this *parallelism setting* might not fit. This proportional odds just follows proper order of cumulative probabilities. If we try to implement different odds, this order might be broken. In this case, *some constraints* should be introduced.

- Adding additional terms
  - e.g. interactions
- link function for which the response curve is nonsymmetric
  - e.g. complementary log-log
- alternative ordinal model for which the more complex non-proportional-odds form
- dispersion parameters
- separate effects for each logit for some but not all predictors
  - e.g. partial proportional odds
- baseline-category logit models and using the ordinality in an informal way in interpreting the associations

6.5 Interpretation

6.5.1 Comparing cumulative probabilities

Logistic regression model has been used odds ratio for interpretation. However, in case of ordinal variable, *using cumulative probabilities can be more intuitive*(**?**). It is easier to conceptualize the size of effects. We can *compare each probability*.

$$\begin{aligned}
\hat{P}(Y=1) &= \hat{P}(Y \leq 1) \\
\hat{P}(Y=2) &= \hat{P}(Y \leq 2) - \hat{P}(Y \leq 1) \\
\hat{P}(Y=3) &= \hat{P}(Y \leq 3) - \hat{P}(Y \leq 2) \\
&\vdots \\
\hat{P}(Y=J) &= 1 - \hat{P}(Y \leq J-1)
\end{aligned} \tag{6.7}$$

Using the fitted values of (6.7), we can interpret the model in various aspects.

- At the extreme values, we can describe effects of quantitive one.
- On the other hands, at the different categories, we can describe effects of qualitative one.

```
gss_pred <-
  gss_contin %>%
  select(race, trauma) %>%
  bind_cols(predict(fit_cumul, newdata = ., type = "response") %>% tbl_df())
```

race	trauma	very	pretty	not
0	0	0.3733	0.594	0.0323
0	1	0.2842	0.668	0.0477
0	2	0.2093	0.721	0.0698
0	3	0.1500	0.749	0.1012
0	4	0.1052	0.750	0.1445
0	5	0.0727	0.725	0.2022
1	0	0.0721	0.724	0.2035
1	1	0.0493	0.674	0.2771
1	2	0.0334	0.602	0.3651
1	3	0.0225	0.514	0.4631

For instance, when the white subject overcomes traumatic event zero, then $\hat{P}(Y = 1 = \text{very happy}) = 0.373$ and $\hat{P}(Y = 2 = \text{pretty happy}) = 0.594$.

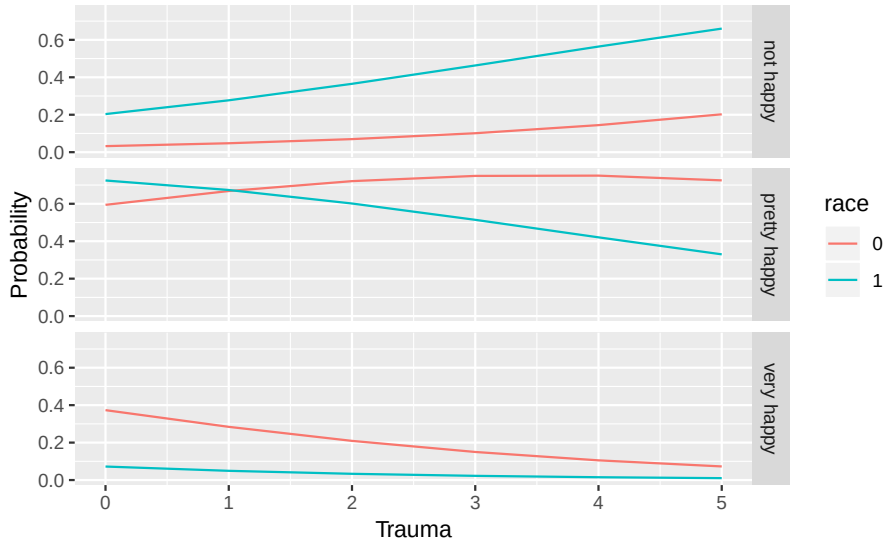


Figure 6.4: Estimated value for each probability for happiness

We can describe for each mean, minimum, and maximum numer of traumatic events.

```
# A tibble: 1 x 3
   mean   max   min
  <dbl> <dbl> <dbl>
1   2.1     5     0
```

Look at the fitted probability of *very happy* in Figure 6.4.

- At the mean number, the difference between blacks and whites is almost `0.2`.
- At the minimum, the difference is almost `0.4`.
- At the maximum, on the other hand, the difference becomes very small.

Comparing blacks to whites, the change in whites is far more large. Again, black people are observed 13 times. Moreover, *none of them has more than 3 traumatic events.*

```
# A tibble: 1 x 2
  race      N
```

```
  <fct> <int>
1 0         7
```

6.6 Cheese Tasting

6.6.1 Data Description

This data is from McCullagh and Nelder (1989). Dr Graeme Newell obtained this data from experiments conducted to investigate the effect of taste on the various cheese additives. In this example, subjects were randomly assigned to taste one of four different cheeses. There are nine levels in response category.

- x = different cheeses: `A`, `B`, `C`, and `D`
- y = `strong dislike`(1) to `excellent taste`(9)

Note that the respopnse variable is ordinal. To interpret the model (6.3) easily, we would recode the taste factor reversely. `strong dislike` would be `9`, and `excellent taste` would be `1`.

```
newell <- read_table("data/GLM175.txt",
                     col_names = str_c("taste", 9:1, sep = "_"))
newell <-
  newell %>%
  add_column(cheese = letters[1:4], .before = 1)
```

Taste	A	B	C	D
1	1	0	0	11
2	8	0	1	16
3	19	1	5	14
4	8	6	7	7
5	8	7	23	3
6	7	11	8	1
7	1	12	6	0
8	0	9	1	0
9	0	6	1	0

In the above table, we can see the distribution of each count of taste vote, i.e. the cheese variable has the ordering $D > A > C > B$. However, this is an empirical measure and statistical modeling is needed to determine if there is really a difference between assessments of flavor depending on the type of cheese. Therefore, the researcher intended to identify that the result is reliable through a proportional-odds cumulative-logit model.

6.6.2 Proportional odds model

```
fit_vglm <- function(.data, y_start, parallel = TRUE, ...) {
  y_names <- names(.data)
  y_mat <-
    .data %>%
    select(starts_with(y_start)) %>%
    as.matrix()
  .data %>%
    select(-starts_with(y_start)) %>%
    vglm(y_mat ~ ., data = ., family = cumulative(parallel = parallel), ...)
}
```

```
(fit_cheese <- fit_vglm(newell, y_start = "taste", parallel = TRUE)) %>%
  summary(lrt0 = TRUE)
```

```
Call:
```

```
vglm(formula = y_mat ~ ., family = cumulative(parallel = parallel),
    data = .)


Pearson residuals:
  logit(P[Y<=1]) logit(P[Y<=2]) logit(P[Y<=3]) logit(P[Y<=4])
1        -0.3071         -0.718         -0.843         2.1286
2         0.0615          0.612          0.184        -0.0796
3         0.0137         -0.805          0.174        -1.3671
4        -0.1256         -0.219         -0.632         0.1733
  logit(P[Y<=5]) logit(P[Y<=6]) logit(P[Y<=7]) logit(P[Y<=8])
1         0.2506         -1.191         -0.098          0.902
2        -2.1764          0.923          0.577          0.200
3         1.1730          0.328          0.453          0.571
4        -0.0755          0.996         -0.103         -0.585

Likelihood ratio test coefficients:
        Estimate z value Pr(>|z|)
cheeseb     3.35    8.44  < 2e-16 ***
cheesec     1.71    4.73  2.2e-06 ***
cheesed    -1.61   -4.37  1.2e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors:  8

Residual deviance: 20.3 on 21 degrees of freedom

Log-likelihood: -47.7 on 21 degrees of freedom

Number of iterations: 5

Exponentiated coefficients:
cheeseb cheesec cheesed
 28.555   5.528   0.199
```

Here, we have used *dummy coding* with $\beta_1 = 0$.

```
[1] a b c d
attr(,"contrasts")
  2 3 4
a 0 0 0
b 1 0 0
c 0 1 0
d 0 0 1
Levels: a b c d
```

Table 6.3: Table continues below

coefficients	logit(P[Y<=1])	logit(P[Y<=2])	logit(P[Y<=3])
(Intercept)	-5.467	-4.412	-3.313
cheeseb	3.352	3.352	3.352
cheesec	1.71	1.71	1.71
cheesed	-1.613	-1.613	-1.613

Table 6.4: Table continues below

logit(P[Y<=4])	logit(P[Y<=5])	logit(P[Y<=6])	logit(P[Y<=7])
-2.244	-0.9078	0.04425	1.546
3.352	3.352	3.352	3.352
1.71	1.71	1.71	1.71
-1.613	-1.613	-1.613	-1.613

logit(P[Y<=8])	p_value
3.106	
3.352	2.485e-15
1.71	4.573e-06
-1.613	1.962e-05

Every β_j is significant. Recall that the dummy coding with $\beta_1 = 0$ leads to

$$logit P(Y \leq j \mid x = \text{cheese } l) = \beta_1 - \beta_l = -\beta_l$$

Then the effect estimates $\hat{\beta}_1 = 3.352$ and $\hat{\beta}_2 = 1.71$ suggest that the odds ratio of Cheese `B` and `C` is smaller than for Cheese `A`. For example, the estimated log odds ratio of Cheese C to A is -3.352, and the tendency of C to receive a good response is $\exp(-3.352) = 0.035$ times lower than that of A. Also, the possibility that the response of D is $\exp(1.613) = 5.017$ times higher than the estimated odds of A. We see that the implied ordering of cheeses in terms of quality is $D > A > C > B$.

6.6.3 Effect

```
# type 3 error, and type 1 here is equivalent to type 3
cheese_eff <- anova(fit_cheese, type = 1, test = "LRT")
```

Table 6.6: Analysis of Deviance Table (Type I tests: terms added sequentially from

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL			24	168.8	
cheese	3	148.5	21	20.31	5.679e-32

We can see `cheese` is significant.

6.6.4 Proportional odds assumption

We try to test proportional odds assumption using LRT.

```
cheese_assume <-
  fit_cheese %>%
  anova(
    fit_vglm(newell, y_start = "taste", parallel = FALSE),
    type = 1,
    test = "LRT"
  )
```

Table 6.7: Analysis of Deviance Table

Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
21	20.31			
0	1.062e-12	21	20.31	0.5018

With univariate model, *non-parallism produces saturated model*. Here non-parallel model has $p = (J - 1)(\text{cheese category} - 1) = (8)(4) = 32$. Thus, its df becomes $n(J-1) - p = 0$.

Testing this model as *alternative hypothesis*, the p-value is 0.502. It can be seen that *the proportional odds assumption of the model is valid*. The model will have 8 intercepts (one for each of the logit equations) and 3 slopes, for a total of 11 free parameters.

```
cheese_pred <-
  newell %>%
  select(cheese) %>%
  bind_cols(predict(fit_cheese, newdata = ., type = "response") %>% tbl_df())
```

Taste	A	B	C	D
1	0.04287	0.00157	0.00804	0.183476
2	0.13281	0.00584	0.02908	0.333237
3	0.31326	0.02501	0.11041	0.310862
4	0.22360	0.04745	0.16204	0.097995
5	0.19159	0.16840	0.32087	0.053732
6	0.06073	0.24192	0.20196	0.013491
7	0.02316	0.25256	0.10476	0.004796
8	0.00778	0.14965	0.04003	0.001571
9	0.00420	0.10760	0.02281	0.000841

For example, Subjects who had eaten `A` cheese answered that it taste `3`(quite tasty) about 0.313 of probability.

```
cheese_pred %>%
  gather(-cheese, key = tasty, value = pp) %>%
  ggplot(aes(x = cheese, y = pp)) +
  geom_bar(aes(fill = tasty), stat = "identity") +
  scale_fill_discrete(labels = 1:9) +
  labs(x = "Cheese", y = "Probability")
```

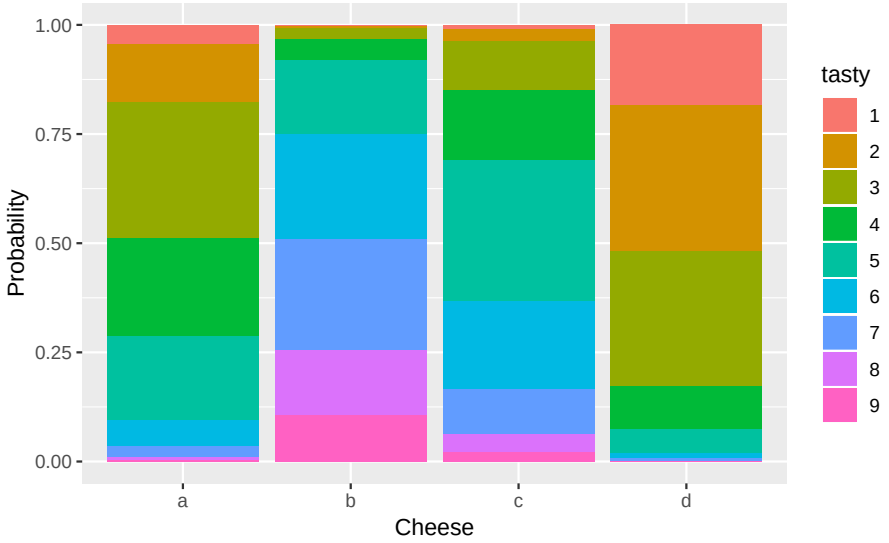


Figure 6.5: Estimated Probability for each chease

In Figure 6.5, we can check how that cheese had been measured. The larger interval means larger probability. As the `tasty` is close to `1`, it means the cheese is preferable. $D > A > C > B$ of our first guess seems right.

Chapter 7

Loglinear Models for Contingency Tables

```
library(tidyverse)
```

7.1 Loglinear models for Two-way tables

Both variables are response variables.

7.1.1 Association between responses

Consider $I \times J$ contingency table $\{n_{ij}\}$. Then

$$n_{ij} \sim Poisson(\mu_{ij})$$

7.2 Loglinear models for Three-way tables

Alcohol, cigarette, and marijuana use

```
(substance <-
  read_delim("data/Substance_use.dat", delim = " ") %>%
  mutate(alcohol = str_trim(alcohol))) # due to messy data file
```

```
# A tibble: 8 x 4
  alcohol cigarettes marijuana count
  <chr>   <chr>      <chr>     <dbl>
1 yes     yes        yes         911
2 yes     yes        no          538
3 yes     no         yes          44
4 yes     no         no          456
5 no      yes        yes           3
6 no      yes        no           43
7 no      no         yes           2
8 no      no         no          279
```

To fit loglinear model, long data format like this is easy.

```
substance <-
  substance %>%
  mutate_if(is.character, factor) %>%
  mutate_if(is.factor, fct_rev)
```

Below is *mutual independence model* (`A, C, M`).

```
(subs_log <-
  substance %>%
  glm(count ~ alcohol + cigarettes + marijuana, data = ., family = poisson())) %>%
  summary()
```

```
Call:
glm(formula = count ~ alcohol + cigarettes + marijuana, family = poisson(),
    data = .)

Deviance Residuals:
     1       2       3       4       5       6       7       8
 14.52   -7.82  -17.68    3.43  -12.44   -8.44   -8.83   19.64

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    6.2915     0.0367  171.56  < 2e-16 ***
alcoholno     -1.7851     0.0598  -29.87  < 2e-16 ***
cigarettesno  -0.6493     0.0442  -14.71  < 2e-16 ***
marijuanano    0.3154     0.0424    7.43  1.1e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 2851.5  on 7  degrees of freedom
Residual deviance: 1286.0  on 4  degrees of freedom
AIC: 1343

Number of Fisher Scoring iterations: 6
```

7.2.1 Chi-square Goodness-of-fit tests

$$G^2 = 2\sum n_{ijk} \ln \frac{n_{ijk}}{\hat{\mu}_{ijk}}$$

$$X^2 = \sum \frac{(n_{ijk} - \hat{\mu}_{ijk})^2}{\hat{\mu}_{ijk}}$$

with

$$\text{residual df} = \text{the number of cell counts} - \text{the number of non-redundant parameters}$$

```
subs_hierarchy <-
  substance %>%
  do(
    indep = glm(count ~ alcohol + cigarettes + marijuana, data = ., family = poisson()),
```

```
    ac_m = glm(count ~ alcohol + cigarettes + marijuana + alcohol:cigarettes,
               data = ., family = poisson()),
    amcm = glm(count ~ alcohol + cigarettes + marijuana + alcohol:marijuana + cigarettes:marijuana,
               data = ., family = poisson()),
    acamcm = glm(count ~ alcohol + cigarettes + marijuana + alcohol:cigarettes + alcohol:marijuana + ci
               data = ., family = poisson()),
    acm = glm(count ~ alcohol * cigarettes * marijuana,
              data = ., family = poisson())
  )
```

```
good_loglin <- function(x, test = "LRT", ...) {
  mod_name <-
    as.character(x[[1]]$call)[2] %>%
    str_extract(pattern = "(?<=~).*") %>%
    str_trim()
  broom::tidy(anova(x[[1]], test = test, ...)) %>%
    slice(n()) %>%
    add_column(model = mod_name, .before = 1) %>%
    select(-term)
}
#-----------------------------------------------
(subs_good <-
  subs_hierarchy %>%
  map(good_loglin, test = "LRT") %>%
  bind_rows()) %>%
  pander::pander()
```

Table 7.1: Table continues below

model	df	Deviance	Resid..Df	Resid..Dev
alcohol + cigarettes + marijuana	1	55.91	4	1286
alcohol + cigarettes + marijuana + alcohol:cigarettes	1	442.2	3	843.8
alcohol + cigarettes + marijuana + alcohol:marijuana + cigarettes:marijuana	1	751.8	2	187.8
alcohol + cigarettes + marijuana + alcohol:cigarettes + alcohol:marijuana + cigarettes:marijuana	1	497	1	0.374
alcohol * cigarettes * marijuana	1	0.374	0	-4.152e-14

p.value
7.575e-14
3.607e-98
1.623e-165
4.283e-110
0.5408

From above G^2, we compare reduced model to complex model

$$G^2(M_0 \mid M_1) = G^2(M_0) - G^2(M_1) \approx \chi^2\Big(df = df(M_0) - df(M_1)\Big)$$

```
subs_good %>%
  select(-Deviance, -p.value) %>%
  rename(alternative = model) %>%
  mutate(goodness = c(Resid..Dev[1], -diff(Resid..Dev)),
         df_good = c(Resid..Df[1], -diff(Resid..Df))) %>%
  mutate(p_value = pchisq(goodness, df = df_good, lower.tail = FALSE)) %>%
  pander::pander()
```

Table 7.3: Table continues below

alternative	df	Resid..Df	Resid..Dev	goodness
alcohol + cigarettes + marijuana	1	4	1286	1286
alcohol + cigarettes + marijuana + alcohol:cigarettes	1	3	843.8	442.2
alcohol + cigarettes + marijuana + alcohol:marijuana + cigarettes:marijuana	1	2	187.8	656.1
alcohol + cigarettes + marijuana + alcohol:cigarettes + alcohol:marijuana + cigarettes:marijuana	1	1	0.374	187.4
alcohol * cigarettes * marijuana	1	0	-4.152e-14	0.374

df_good	p_value
4	3.574e-277
1	3.607e-98
1	1.068e-144
1	1.186e-42
1	0.5408

1. saturated model `(ACM)`: cannot reject M_0, so we choose next model
2. three factor interaction `(AC,AM,CM)`: reject M_0

Thus, we use model *(AC, AM, CM)*.

7.2.2 Fitted values

```
fit_loglin <- function(x, ...) {
  mod_name <-
    as.character(x[[1]]$call)[2] %>%
    str_extract(pattern = "(?<=~).*") %>%
    str_trim()
  x[[1]]$model %>%
    bind_cols(predict(x[[1]], newdata = ., type = "response", ...) %>% tbl_df()) %>%
    rename_at(.vars = vars(value), .funs = funs(return(mod_name)))
}
#--------------------------------
(subs_fit <-
  subs_hierarchy %>%
```

```
map(fit_loglin) %>%
plyr::join_all(by = c("count", "alcohol", "cigarettes", "marijuana"))) %>%
pander::pander()
```

Table 7.5: Table continues below

count	alcohol	cigarettes	marijuana	alcohol + cigarettes + marijuana
911	yes	yes	yes	540
538	yes	yes	no	740.2
44	yes	no	yes	282.1
456	yes	no	no	386.7
3	no	yes	yes	90.6
43	no	yes	no	124.2
2	no	no	yes	47.33
279	no	no	no	64.88

Table 7.6: Table continues below

alcohol + cigarettes + marijuana + alcohol:cigarettes	alcohol + cigarettes + marijuana + alcohol:marijuana + cigarettes:marijuana
611.2	909.2
837.8	438.8
210.9	45.76
289.1	555.2
19.4	4.76
26.6	142.2
118.5	0.2396
162.5	179.8

alcohol + cigarettes + marijuana + alcohol:cigarettes + alcohol:marijuana + cigarettes:marijuana	alcohol * cigarettes * marijuana
910.4	911
538.6	538
44.62	44
455.4	456
3.617	3
42.38	43
1.383	2
279.6	279

Bibliography

Agresti, A. (2007). *An Introduction to Categorical Data Analysis*. Wiley, 2 edition.

Agresti, A. (2012). *Categorical Data Analysis*. Wiley, 3 edition.

McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models, Second Edition*. CRC Press.