HW2

Anna Ma

2/18/2022

```
library(tidyverse)
library(dplyr)
```

Problem 1

```

response_df = tibble(
  death = c(2, 8, 15, 23, 27),
  survive = 30 - death)
response_matrix = response_df %>% as.matrix()
dose = c(0:4)
```

a)

1. Fit models

```
#logit model
mod_logit = glm(response_matrix ~ dose, family = binomial(link = 'logit'))
#probit model
mod_probit = glm(response_matrix ~ dose, family = binomial(link = 'probit'))
#cloglog model
mod_cloglog = glm(response_matrix ~ dose, family = binomial(link = 'cloglog'))
```

2. 95% Confidence Interval for each model

```
#logit model

ci_logit = mod_logit %>%
  broom::tidy() %>%
 mutate(lower = estimate - 1.96 * std.error,
        upper = estimate + 1.96 * std.error)

#probit model

ci_probit = mod_probit %>%
  broom::tidy() %>%
 mutate(lower = estimate - 1.96 * std.error,
        upper = estimate + 1.96 * std.error)

#cloglog model
ci_cloglog = mod_logit %>%
  broom::tidy() %>%
 mutate(lower = estimate - 1.96 * std.error,
        upper = estimate + 1.96 * std.error)
```

```
ci_table = ci_logit %>% mutate(term = recode(term,dose = "logit")) %>%   rbind(ci_probit %>% mutate(tern
  rbind(ci_cloglog %>% mutate(term = recode(term,dose = "cloglog"))) %>% as_tibble %>% rename(model = tc
```

3. Deviance

```
dev_logit = deviance(mod_logit)
dev_probit = deviance(mod_probit)
dev_cloglog = deviance(mod_cloglog)

dev_table = rbind(dev_logit,dev_probit,dev_cloglog) %>% as_tibble() %>%
  rename(deviance = V1)
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is
## Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

4. p(dying|x=0.01)

```
p_hat = data.frame(dose = 0.01)
predict_logit = predict(mod_logit, p_hat, type = "response")
predict_probit = predict(mod_probit, p_hat, type = "response")
predict_cloglog = predict(mod_cloglog, p_hat, type = "response")

p_table = rbind(predict_logit,predict_probit,predict_cloglog) %>% as_tibble() %>% rename(p_estimate = 1)
```

```
table = tibble(
  model = c("Logit", "Probit", "c-log-log"),
  Estimate_of_beta =
    c(mod_logit$coefficients[2],
      mod_probit$coefficients[2],
      mod_cloglog$coefficients[2]),
  CI_for_beta =
    c(str_c("(", round(ci_logit$lower[2],3) ,", ", round(ci_logit$upper[2],3), ")"),
      str_c("(", round(ci_probit$lower[2],3), ", ", round(ci_probit$upper[2],3), ")"),
      str_c("(", round(ci_cloglog$lower[2],3), ", ", round(ci_cloglog$upper[2],3), ")")),
  Deviance =
    c(mod_logit$deviance, mod_probit$deviance, mod_cloglog$deviance),
  p_estimate =
    c(predict_logit, predict_probit, predict_cloglog))


knitr::kable(table, digit = 3)
```

model	Estimate_of_beta	CI_for_beta	Deviance	p_estimate
Logit	1.162	(0.806, 1.517)	0.379	0.090
Probit	0.686	(0.497, 0.876)	0.314	0.085
c-log-log	0.747	(0.806, 1.517)	2.230	0.128

Comments:

- From the table, we can see that all of the estimate of β are positive, meaning that the number of death increases with dose. For example, the logit model shows that the log odds ratio of death increase by 1.1618949 with every one unit increase in the dose.

- Out of the three models, the logit model has the widest confidence interval. All the confidence intervals are positive, again indicate that the risk of death increases with dose.
- The c-log-log model has the largest deviance of 2.23, which is much larger than the other two model, suggesting that it might not fit as good as the other two.

```
#logit
# coefficients
logit_b0 = mod_logit$coefficients[1]
logit_b1 = mod_logit$coefficients[2]
# fisher information inverse
logit_bcov = vcov(mod_logit)
# point estimate of ln(LD50)
logit_x0 = -logit_b0/logit_b1
# asymptotic variance
logit_varx0 = logit_bcov[1,1]/(logit_b1^2) + logit_bcov[2,2]*(logit_b0^2)/(logit_b1^4) -
              2*logit_bcov[1,2]*logit_b0/(logit_b1^3)

logit_estimate = exp(logit_x0)
logit_CI = exp(logit_x0 + c(qnorm(0.05), -qnorm(0.05)) *sqrt(logit_varx0))
```

```
# probit
# coefficients
probit_b0 = mod_probit$coefficients[1]
probit_b1 = mod_probit$coefficients[2]
# fisher information inverse
probit_bcov = vcov(mod_probit)
# point estimate of ln(LD50)
probit_x0 = -probit_b0/probit_b1
# asymptotic variance
probit_varx0 = probit_bcov[1,1]/(probit_b1^2) + probit_bcov[2,2]*(probit_b0^2)/(probit_b1^4) -
              2*probit_bcov[1,2]*probit_b0/(probit_b1^3)

probit_estimate = exp(probit_x0)
probit_CI = exp(probit_x0 + c(qnorm(0.05), -qnorm(0.05)) *sqrt(probit_varx0))
```

```
#cloglog
# coefficients
cloglog_b0 = mod_cloglog$coefficients[1]
cloglog_b1 = mod_cloglog$coefficients[2]
# fisher information inverse
cloglog_bcov = vcov(mod_cloglog)
# point estimate of ln(LD50)
cloglog_x0 = (log(-log(0.5)) - cloglog_b0)/cloglog_b1
# asymptotic variance
cloglog_varx0 = cloglog_bcov[1,1]/(cloglog_b1^2) + cloglog_bcov[2,2]*(cloglog_b0^2)/(cloglog_b1^4) -
              2*cloglog_bcov[1,2]*cloglog_b0/(cloglog_b1^3)

cloglog_estimate = exp(cloglog_x0)
cloglog_CI = exp(cloglog_x0 + c(qnorm(0.05), -qnorm(0.05)) * sqrt(cloglog_varx0))
```

```r
LD50_estimate =
  tibble(
  model = c("logit", "probit", "cloglog"),
  estimate = c(logit_estimate, probit_estimate, cloglog_estimate),
  CI = c(str_c("(", round(logit_CI[1],3) ,", ", round(logit_CI[2],3),")"),
         str_c("(", round(probit_CI[1],3) ,", ", round(probit_CI[2],3),")"),
         str_c("(", round(cloglog_CI[1],3) ,", ", round(cloglog_CI[2],3), ")"))
  )
knitr::kable(LD50_estimate, digit = 3)
```

b)

model	estimate	CI
logit	7.389	(5.51, 9.91)
probit	7.436	(5.583, 9.904)
cloglog	8.841	(6.636, 11.779)

Problem 2

```r
scholarship_df = tibble(
  amount = seq(from = 10, to = 90, by = 5),
  offers = c(4, 6, 10, 12, 39, 36, 22, 14, 10, 12, 8, 9, 3, 1, 5, 2, 1),
  enrolls = c(0, 2, 4, 2, 12, 14, 10, 7, 5, 5, 3, 5, 2, 0, 4, 2, 1),
  nonenrolls = offers - enrolls
)

amount = scholarship_df$amount %>% as.matrix()
enrolls = scholarship_df %>% select(enrolls, nonenrolls) %>% as.matrix()
```

Fit logistic model

```r
log_fit = glm(enrolls ~ amount, family = binomial(link = 'logit'))
```

```r
dev = log_fit$deviance
dev
```

a)

```
## [1] 10.61271
```

```r
chi_stat = sum(residuals(log_fit, type = "pearson")^2)
chi_stat
```

```
## [1] 8.814299
```

```r
p = 1 - pchisq(dev,15)
p
```

```
## [1] 0.7795345
```

Since the data is grouped, we can calculate the person chi-squared and deviance to evaluate the goodness of fit. For this model, the deviance is 10.6127107, and the statistics of the Pearson chi-squared is 8.8142993.

The calculated p value is 0.7795345 > 0.05. Therefore, we fail to reject the hypothesis that the model we have is different from the full model. Thus, we can conclude that the model is close to the full model, and therefore a good fit of the data.

```
# Odds of enrollment with no scholarship
exp(log_fit$coefficients[1])
```

b)

```
## (Intercept)
##    0.192504
```

```
# beta 1
exp(log_fit$coefficients[2])
```

```
##   amount
## 1.031434
```

According to the model, the odds of enrollments is 0.192504 when there's no scholarship given; with every unit increase of scholarship, the odds of enrollment is 1.0314344 times larger, such that we can expect a 3% increase in the enrollment.

```
ci_95 = log_fit %>%
  broom::tidy() %>%
  mutate(lower = estimate - 1.96 * std.error,
        upper = estimate + 1.96 * std.error) %>%
 select(term, estimate, lower, upper) %>%
 filter(term == "amount")

exp_ci = str_c("(",round(exp(ci_95$lower),3),",", round(exp(ci_95$upper),3),")")
```

The 95% confidence interval is (1.012,1.051) for the odds ratio.

```
log_fit_b0 = log_fit$coefficients[1]
log_fit_b1 = log_fit$coefficients[2]

log_fit_bcov = vcov(log_fit)

log_fit_x0 = (log(0.4/0.6) - log_fit_b0) / log_fit_b1

log_fit_x0
```

c)

```
## (Intercept)
##    40.13429
```

We should provide 40.134 thousand dollars to get a 40% yield rate

```
log_fit_varx0 = log_fit_bcov[1,1]/(log_fit_b1^2) + log_fit_bcov[2,2]*((log_fit_b0 - log(0.4/0.6))^2)/(lo
              2*log_fit_bcov[1,2]*(log_fit_b0 - log(0.4/0.6))/(log_fit_b1^3)

ci_log_fit = (log_fit_x0 + c(qnorm(0.025),
                     -qnorm(0.025))*sqrt(log_fit_varx0))
CI = c(str_c("(", round(ci_log_fit[1],3) ,", ", round(ci_log_fit[2],3),")"))
```

The 95 confidence interval is (30.583, 49.686), We are 95% confident that the amount needed to provide a 40% yield rate is between 30.583 and 49.686.