Homework 05

Minji Jeong

Problem 1:Modeling vote choice with multinomial logit

1a: Fit a multinomial logit model to explain Presidential vote choice

```
## Loading data:

df <- read_csv("https://faculty.washington.edu/cadolph/mle/nes92con.csv")

## Changing data structure:

df <-
  df |>
  # create factor categories
  mutate(vote92_f = case_when(vote92==0 ~ "Bush",
                              vote92==1 ~ "Clinton",
                              vote92==2 ~ "Perot"),
         vote92_f = factor(vote92_f))


pe_multinom <- function(model=x){

  # check class "multinom"
  if ((sum(class(model) == "multinom") == 0)) {

    # if FALSE, stop and warning message
    warning("Please, provide as input model a 'multinom' object.")
    return(NULL)

  }
  # if TRUE, extract estimates
  else {

    # number of levels (categories)
    nlevel <- length(model$lev)

    # number of estimates
    ncoef <- length(model$coefnames)

    # remove the baselines and arrange order
    coef <- model$wts[(ncoef+2):length(model$wts)]
    coef <- coef[-((0:(nlevel-2))*(ncoef+1) + 1)]

    return(coef)
  }
}
```

```r
m1 <- vote92_f ~ rlibcon + rbdist + econ + gulfwar + nonwhite # formula

df1 <- extractdata(m1, df, na.rm=T)

mlogit.result <- multinom(m1,
                          data = df1,
                          Hess=TRUE)
```

```
## # weights:  21 (12 variable)
## initial  value 511.953327
## iter  10 value 364.141750
## final  value 353.587881
## converged
```

```r
# Extract statistics:

pe <- pe_multinom(mlogit.result)
vc <- solve(mlogit.result$Hess)       # var-cov matrix
se <- sqrt(diag(vc))
ll <- mlogit.result$value
# alternatively, logLik(mlogit.result) OR mlogit.result$deviance/2

# Output table:

output <- tibble(
  vars = rep(mlogit.result$coefnames, 2),
  candidate = c(rep(mlogit.result[["lev"]][2], 6),
             rep(mlogit.result[["lev"]][3], 6)),
  pe = as.numeric(pe),
  se = se,
  loglike = ll
)


output[,1:4] |>
  kable(caption = "Nominal Logit of nes92. Max log likelihood = 354" )
```

Table 1: Nominal Logit of nes92. Max log likelihood = 354

vars	candidate	pe	se
(Intercept)	Clinton	-1.1340401	0.9935391
rlibcon	Clinton	-0.5250475	0.1101013
rbdist	Clinton	0.7920291	0.1323943
econ	Clinton	0.5969867	0.1712137
gulfwar	Clinton	-0.7615256	0.3134483
nonwhite	Clinton	0.7537022	0.5164738
(Intercept)	Perot	-2.1462467	0.9889755
rlibcon	Perot	-0.0764688	0.1125668
rbdist	Perot	0.6818460	0.1341595
econ	Perot	0.2918575	0.1586825
gulfwar	Perot	-0.3114140	0.3277164
nonwhite	Perot	-0.9945756	0.7194823

1b: Calculate the probability of white/non-white

```r
# First, set up simbetas

sims <- 10000
```

```r
simbetas <- mvrnorm(n = sims, mu = pe, Sigma = vc)


### b1) Counterfactual analysis for white voter with varying ```rlibcon```

colnames(simbetas)
```

```
##  [1] "Clinton:(Intercept)" "Clinton:rlibcon"     "Clinton:rbdist"
##  [4] "Clinton:econ"        "Clinton:gulfwar"     "Clinton:nonwhite"
##  [7] "Perot:(Intercept)"   "Perot:rlibcon"       "Perot:rbdist"
## [10] "Perot:econ"          "Perot:gulfwar"       "Perot:nonwhite"
```

```r
simB <- array(NA, dim=c(sims,6,2))
# Fill in simulated beta arrays with appropriate matrices
# Remember we have 5 independent variables

simB[,,1] <- simbetas[,1:6]
simB[,,2] <- simbetas[,7:12]


rlibcon_hyp <- seq(from=1,to=7,by=1) # counterfactuals rlibcon values

# Counter factual for non-white respondent

xhyp_nw <- cfFactorial(rlibcon=rlibcon_hyp,
                       rbdist=mean(df1$rbdist),
                       econ=mean(df1$econ),
                       gulfwar=mean(df1$gulfwar),
                       nonwhite=1)

# Counter factual for white respondent

xhyp_w <- cfFactorial(rlibcon=rlibcon_hyp,
                      rbdist=mean(df1$rbdist),
                      econ=mean(df1$econ),
                      gulfwar=mean(df1$gulfwar),
                      nonwhite=0)


# Simulate expected probabilities
(mlogit_ev_w <- mlogitsimev(xhyp_w, simB, ci=0.95))
```

```
## $pe
##           [,1]      [,2]      [,3]
## [1,] 0.7421375 0.1378955 0.1199671
## [2,] 0.6453495 0.1835085 0.1711419
## [3,] 0.5318344 0.2338615 0.2343041
## [4,] 0.4127368 0.2825434 0.3047199
## [5,] 0.3022207 0.3227042 0.3750750
## [6,] 0.2108247 0.3504152 0.4387601
## [7,] 0.1419513 0.3657476 0.4923011
##
## $lower
## , , 1
##
##            [,1]       [,2]       [,3]
## [1,] 0.61467904 0.07196951 0.05618952
## [2,] 0.53665436 0.11691481 0.10072969
## [3,] 0.44920938 0.17327262 0.16479869
## [4,] 0.34979891 0.22873732 0.24083950
## [5,] 0.24014501 0.26357976 0.30634057
## [6,] 0.14751751 0.26744045 0.34325328
## [7,] 0.08231241 0.24924768 0.35627851
##
```

```
##
## $upper
## , , 1
##
##           [,1]      [,2]      [,3]
## [1,] 0.8446871 0.2322041 0.2159334
## [2,] 0.7435549 0.2674885 0.2635423
## [3,] 0.6131577 0.3048308 0.3151656
## [4,] 0.4786934 0.3421053 0.3734473
## [5,] 0.3712240 0.3857051 0.4473634
## [6,] 0.2871592 0.4400818 0.5359317
## [7,] 0.2204623 0.4970314 0.6265656
```

```
(mlogit_ev_nw <- mlogitsimev(xhyp_nw, simB, ci=0.95))
```

```
## $pe
##           [,1]       [,2]       [,3]
## [1,] 0.8879618 0.03580539 0.07623283
## [2,] 0.8347329 0.05025276 0.11501438
## [3,] 0.7609119 0.06900936 0.17007870
## [4,] 0.6650392 0.09161250 0.24334826
## [5,] 0.5514437 0.11618257 0.33237375
## [6,] 0.4312839 0.13966827 0.42904781
## [7,] 0.3187657 0.15911133 0.52212300
##
## $lower
## , , 1
##
##           [,1]        [,2]       [,3]
## [1,] 0.7429441 0.006292763 0.01857162
## [2,] 0.6654274 0.010518870 0.03483018
## [3,] 0.5722447 0.016530997 0.06236597
## [4,] 0.4652503 0.024262765 0.10540057
## [5,] 0.3486643 0.032944904 0.16212031
## [6,] 0.2427336 0.040535287 0.22810172
## [7,] 0.1539422 0.044452348 0.28880925
##
##
## $upper
## , , 1
##
##           [,1]      [,2]      [,3]
## [1,] 0.9654905 0.1098514 0.1966855
## [2,] 0.9385799 0.1418707 0.2591423
## [3,] 0.8952148 0.1799065 0.3405146
## [4,] 0.8286783 0.2236583 0.4340682
## [5,] 0.7398365 0.2721227 0.5413574
## [6,] 0.6348036 0.3195721 0.6511021
## [7,] 0.5245511 0.3656726 0.7450546
```

1c: Visualization

```
## tidy a simcf object:

levels(df1$vote92_f) # Check levels
```

```
## [1] "Bush"    "Clinton" "Perot"
```

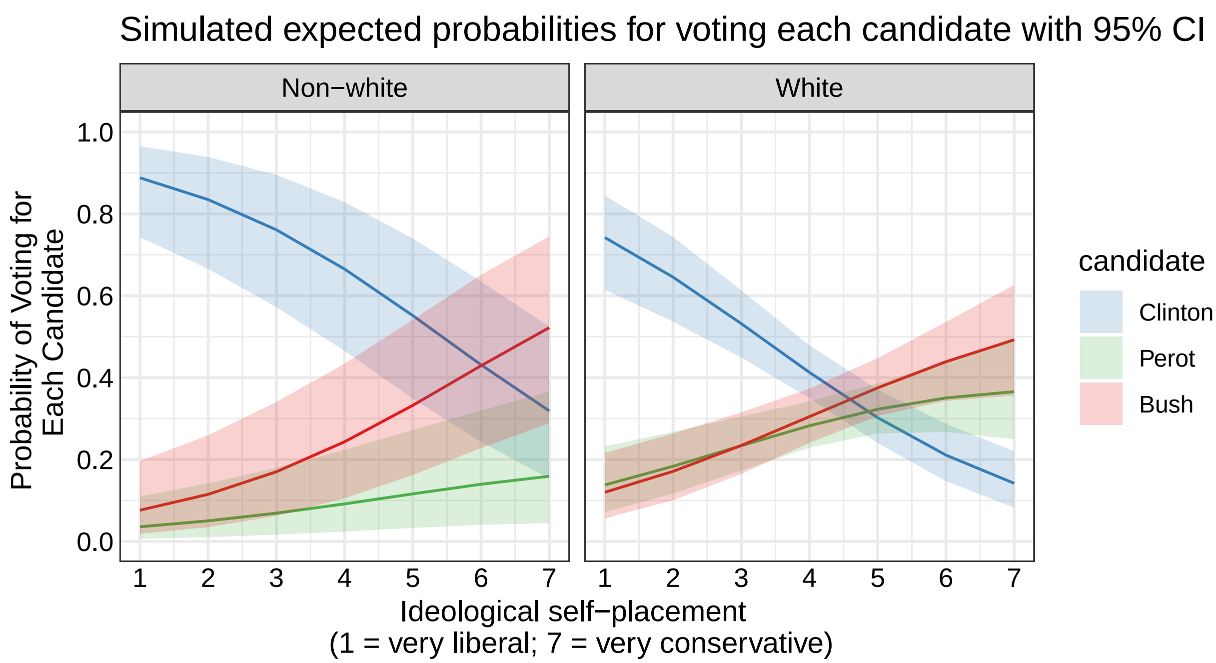
```r
mlogit_ev_w$pe[,3] # George bush predictions
```

```
## [1] 0.1199671 0.1711419 0.2343041 0.3047199 0.3750750 0.4387601 0.4923011
```

```r
# note: mlogitsimev shifts the baseline prediction to the last column

labels <- c("Clinton","Perot","Bush") # labels for mlogitsimev result

mlogit_ev_w_tidy <-
  mlogit_ev_w |>
  # Turns arrays into list
  map(as.data.frame) |>
  # Set category names
  map(set_colnames, labels) |>
  # Add and bind counterfactual scenarios
  map(bind_cols, crossing(rlib = rlibcon_hyp, nonwhite = "White")) |>
  # map the list's objects into a tibble (df),
  # associate each object's name in a new vector called "Type"
  map_df(as_tibble, .id = "Type") |>
  # pivot data from wide to long format
  pivot_longer(labels[1]:labels[3], names_to = "candidate", values_to = "value") |>
  # and now from long to wide, to have a vector for the point estimates and CIs
  pivot_wider(names_from = Type, values_from = value)


mlogit_ev_nw_tidy <-
  mlogit_ev_nw |>
  # Turns arrays into list
  map(as.data.frame) |>
  # Set category names
  map(set_colnames, labels) |>
  # Add and bind counterfactual scenarios
  map(bind_cols, crossing(rlib = rlibcon_hyp, nonwhite = "Non-white")) |>
  # map the list's objects into a tibble (df),
  # associate each object's name in a new vector called "Type"
  map_df(as_tibble, .id = "Type") |>
  # pivot data from wide to long format
  pivot_longer(labels[1]:labels[3], names_to = "candidate", values_to = "value") |>
  # and now from long to wide, to have a vector for the point estimates and CIs
  pivot_wider(names_from = Type, values_from = value)


# merge and prepare for visualization:

dat_vis <-
  bind_rows(
    mlogit_ev_w_tidy, mlogit_ev_nw_tidy
  ) |>
  # change the data structure for visualization
  mutate(candidate = factor(candidate, levels=labels),
         Nonwhite = factor(nonwhite,
                           labels = c("White", "Non-white")))
```

```r
# visualize

dat_vis |>
  ggplot(mapping = aes(x = rlib,
                       y = pe,
                       ymax = upper,
                       ymin = lower,
                       color = candidate,
                       fill = candidate))+
  geom_line(show.legend = F) +
  geom_ribbon(alpha = 0.2, linetype = 0) +
  # Facet to white and non-white
```

```
facet_wrap(~ nonwhite, labeller = label_value) +
scale_color_manual(values = c("Bush"=red, "Clinton"=blue, "Perot"=green)) +
scale_fill_manual(values = c("Bush"=red, "Clinton"=blue, "Perot"=green)) +
scale_y_continuous(breaks = seq(0, 1, by = 0.2), limits = c(0,1))+
scale_x_continuous(breaks = seq(1, 7, by = 1), limits=c(1,7)) +
labs(title="Simulated expected probabilities for voting each candidate with 95% CI",
  x = "Ideological self-placement \n (1 = very liberal; 7 = very conservative)",
     y = "Probability of Voting for \n Each Candidate") +
theme_bw() +
theme(aspect.ratio = 1,
      strip.text = element_text(size = 10, color = "black"),
      axis.text = element_text(size = 10, color = "black"),
      axis.ticks = element_blank(),
      legend.position = "right")
```



1d: What bearing does the independence of irrelevant alternatives assumption have on your results?

The independence of irrelevant alternative (IIA) is closely tied to transitive (or acyclic) preferences. Under IIA, an individual's choice between two options does not change simply because another option is introduced. It holds when the third option is a close substitute to one option, but not to the other. Violations of IIA indicate inaccuracies in our model of the data-generating process (DGP) (Ward and Ahlquist, 2018, p.166-7). Because IIA is a core assumption underlying maximum likelihood estimation in multinomial logit models, violating it can lead to biased and inconsistent point estimates (Ward and Ahlquist, 2018, p. 38).

For example, in an election between a liberal candidate (Clinton) and a conservative candidate (Bush). Under IIA, the entry of a second conservative candidate (Perot) should not affect the relative likelihood that a voter chooses between Clinton and Bush. However, when candidates are close to each other in policy space, voters may see them as substitutes. In such cases, the presence of an additional similar candidate can shift vote shares, indicating that the IIA assumption is unlikely to hold (Alvarez and Nagler, 1998, p. 57).

Given that voter behavior often departs from the IIA assumption due to factors such as strategic voting, ideological overlap, or candidate similarity in election, it's crucial to examine potential violations. If introducing or removing a candidate significantly changes the probability of voting between the remaining candidates, the model may fail to capture true voter preferences. Alternative modeling methods, such as the alternative-specific multinomial probit model or nested logit model, can help relax the IIA assumption.

In summary, multinomial logit assumes IIA, indicating that (1) errors are independent across categories, (2) errors are identically distributed, and (3) errors all follow a type-I extreme value distribution (Ward and Ahlquist, 2018, p.184).

Problem 2: Can a Model Be Too Good? Logit Revisited

2a: Show the profile likelihood of β over the domain -10:10

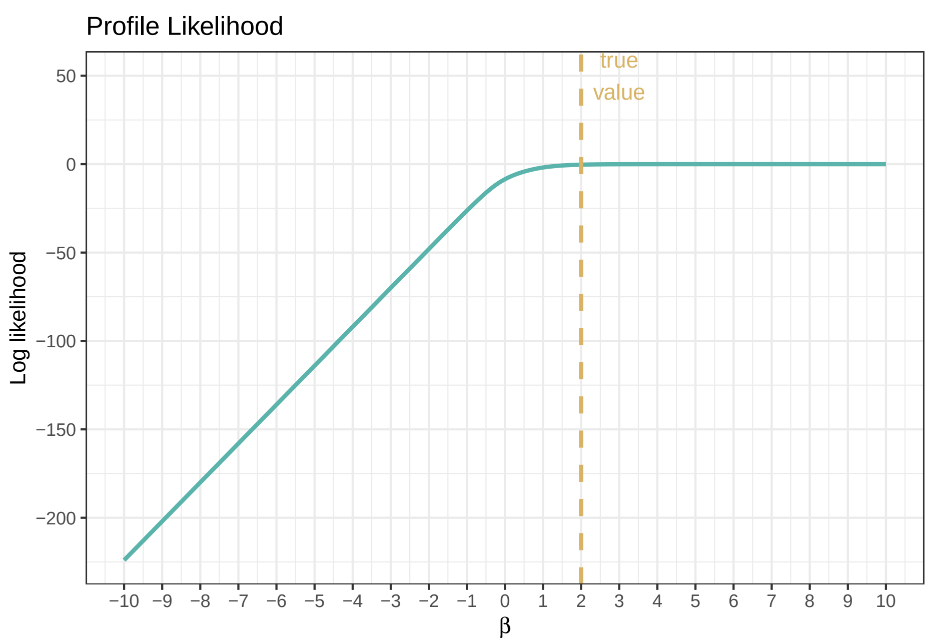
```
y = c(1, 0, 0, 0, 1, 0, 0, 0)

x = c(5,-2,-2,-2, 5,-2,-2,-2)

# Beta will be infinity because there is perfect collinearity

# Likelihood function for logit
llk.logit <- function(param,y,x) {
  os <- rep(1,length(x))
  x <- as.matrix(cbind(os,x))
  b <- param[ 1:ncol(x) ]
  xb <- x%*%b
  -sum( y*log(1+exp(-xb)) + (1-y)*log(1+exp(xb)))
}

llk.logit(param=c(1,1),y=y,x=x)
```

```
## [1] -1.884521
```

```
b = seq(-10, 10, 0.01) # coefficient domain

os <- rep(1,length(b)) # intercept

lls <- vector(mode = "numeric", length = length(b)) # create a vector to store the likelihoods

for(i in 1:length(b)) {
  lls[i] <- llk.logit(y = y,x = x, param= c(os[i],b[i]))
```

```
}

df2 <- data.frame(b,lls)

ggplot(data = df2,
       aes(x = b, y = lls)) +
  geom_line(aes(x = b,
                y = lls),
            color = turquoise, size = 1) +
  geom_vline(xintercept = 2,
             size=1,
             color=brown,
             linetype="dashed") +
  annotate("text",label="true\nvalue", color=brown,
           x=3,y=50)+
  theme_bw() +
  labs(title="Profile Likelihood") +
  scale_y_continuous(name = "Log likelihood") +
  scale_x_continuous(limits = c(-10, 10),
                     breaks = seq(-10, 10, 1),
                     name = expression(beta))
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```



2b: Estimate a logit model of y as a function of x using MLE and GLM

Note that the variables **y** and **x** exhibit perfect collinearity. Gelman et al. (2020) highlight one of the reasons logistic regression may not be identified: perfect collinearity between outcome and a predictor.In the context of binary regression (p. 256). Ward and Ahlquist (2018) call this problem as perfect separation, which occurs when some covariate in a model perfectly predicts an outcome, or so nearly so that the MLE cannot be identified (p. 89). In here, every time we observe $y = 1$ we also observe $x = 5$. Similarly, when $y = 0$ we observe $x = -5$. Hence, perfect collinearity causes the computation of the point estimates unfeasible.

```
# Likelihood function for logit
llk.logit <- function(param,y,x) {
  os <- rep(1,length(x))
  x <- cbind(os,x)
  b <- param[ 1 : ncol(x) ]
  xb <- x%*%b
  sum( y*log(1+exp(-xb)) + (1-y)*log(1+exp(xb)))
}

# Fit logit model using optim
ls.result <- lm(y~x)  # use ls estimates as starting values
stval <- ls.result$coefficients  # initial guesses
# call minimizer procedure
logit.m1 <- optim(stval,llk.logit,method="BFGS",hessian=T,y=y,x=x)
# Nelder-Mead fails to give Hessian; BFGS gives Hessian, but it is meaningless

pe.m1 <- logit.m1$par   # point estimates
vc.m1 <- solve(logit.m1$hessian)  # var-cov matrix
se.m1 <- sqrt(diag(vc.m1))    # standard errors
ll.m1 <- -logit.m1$value  # likelihood at maximum

# Alternative estimation technique:  GLM
glm.m1 <- glm(y~x, data=df2, family="binomial")

glm.m1 |>
  broom::tidy() |>
  mutate(estimate.glm = estimate,
         estimate.optim = pe.m1,
         std.error.glm = std.error,
         std.error.optim = se.m1) |>
  select(-estimate, -std.error, -statistic, -p.value)|>
  kable(caption = "Model comparison between logit MLE and GLM" )
```

Table 2: Model comparison between logit MLE and GLM

term	estimate.glm	estimate.optim	std.error.glm	std.error.optim
(Intercept)	-10.528315	-2.640884	46476.57	82191.24
x	7.018877	9.761216	15281.40	38745.32

```
tibble(logll.glm = logLik(glm.m1),
       logll.optim = ll.m1)|>
  kable()
```

logll.glm	logll.optim
-1.714673e-10	0

2c: Now consider real world dara of government collapse in India

See below the table reporting the estimates of collapse regressed on the rest of the variables within the dataset govcollpase. As we can see, the standard errors are unrealistic.

Something is wrong, but in this case, there is no perfect collinearity between the outcome and the covariates, or is it?

Looking carefully at cross-tabulated data for each variable, we can see something odd with the variable **govt_end** - it shows perfect separation. The observations with $Y = 0$ have values of $govtend < 3$, and when $Y = 1$ we always observe $govtend = 3$.

Note the warning message *"glm.fit: algorithm did not converge"*. Ward and Ahlquist (2018) show a similar case, indicating that the optimizer called by `glm()` is spinning off towards +/- infinity, forcing the predicted probabilities to one of the boundaries (pp. 89-90). In particular, this means there is probably no unique set of parameters that maximizes the likelihood. Therefore, these results are not reliable.

```
df3 <- read.csv("data/govcollapse.csv")

# Using glm to estimate logit

fit.glm <- glm(collapse ~ .,      # . for all the covariates
               family = binomial (link = "logit"),
               data = df3)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# The variable govt_end shows complete separation:

table(df3$collapse,
      df3$govt_end)
```

```
##
##      1  2  3
##   0 10 11  0
##   1  0  0 26
```

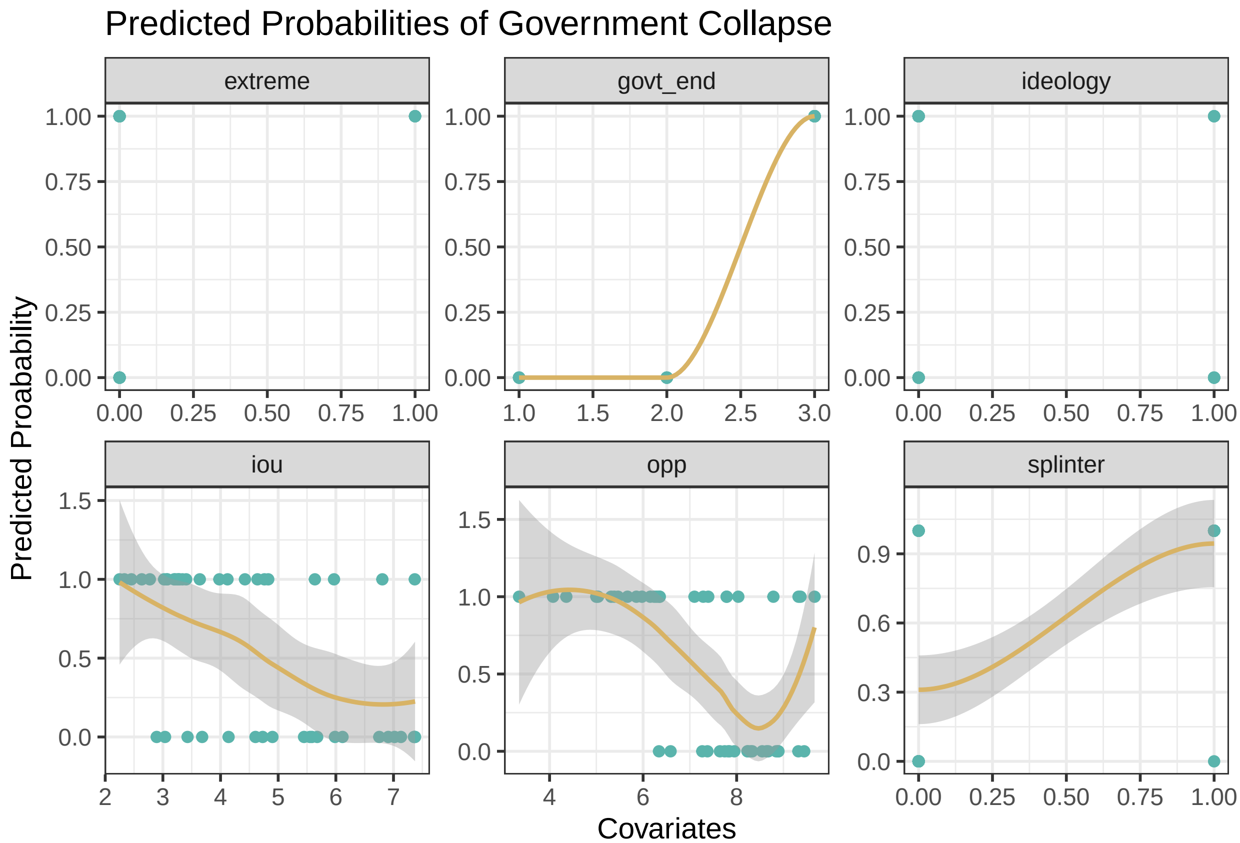
```
stargazer::stargazer(fit.glm,
                     type = "latex")
```

% Table created by stargazer v.5.2.3 by Marek Hlavac, Social Policy Institute. E-mail: marek.hlavac at gmail.com % Date and time: Mon, Dec 15, 2025 - 7:56:05 AM

```
pred <- predict(fit.glm, type="response")

vis <- cbind(df3,pred)

covariates <- names(df3[,-1])

longdata <- vis %>%
  pivot_longer(cols = covariates, names_to = "Covariate", values_to = "Covariate_value")
# Plot using ggplot
ggplot(longdata, aes(x = Covariate_value, y = pred)) +
  geom_point(color = turquoise, size = 1.5) +
  geom_smooth(method = "loess", col = brown, size=0.8) +
  facet_wrap(~Covariate, scales = "free", ncol = 3) +
  theme_bw() +
  labs(title = "Predicted Probabilities of Government Collapse",
       y="Predicted Proabability",
       x="Covariates") +
  theme(legend.position = "none")
```

Table 4:

	Dependent variable:
	collapse
iou	−0.188
	(32,684.990)
ideology	0.486
	(92,476.730)
splinter	0.174
	(104,241.400)
opp	−0.206
	(30,026.850)
extreme	0.222
	(169,868.300)
govt_end	50.363
	(82,900.660)
Constant	−123.145
	(441,282.900)
Observations	47
Log Likelihood	−0.000
Akaike Inf. Crit.	14.000
Note:	*p<0.1; **p<0.05; ***p<0.01



2d: Examine the in-sample predicted probabilities

In the in-sample predicted probabilities of government collapse against each covariate, all the predictions are perfectly allocated in either 0 or 1. As mentioned earlier, this is a consequence of perfect separation on a set of covariates.

The second reason that logistic regression can be nonidentified is discreteness of data. Gelman et al. (2020) explain that if a predictor $x_{govtend}$ is completely aligned with the outcome, so that y = 1 for all the cases where x exceeds some threshold T, and y = 0 for all cases where $x_{govtend} < T$, then the best estimate for the coefficient β is $+\infty$. Conversely, if y = 1 for all cases where $x_{govtend} < T$, and y = 0 for all cases where $x_{govtend} > T$, then β will be $-\infty$ (p. 256).

There are several ways to handle a nonidentification problem raised from perfect or complete separation. We can (1) use penalized likelihood methods (e.g., LASSO) that remove first-order bias from the ML estimates, (2) go Bayesian and use weak prior that will provide some amount of information that will allow the optimizer to converge, or (3) drop the variable or the cases that induce the problem of complete separation.

In conclusion, we should carefully analyze before fitting a logistic regression as concerns such as perfect separation may arise.