GR5065 Homework 5

Ben Goodrich

Due April 11, 2022 at 4PM

```
set.seed(20220411)
options(mc.cores = parallel::detectCores())
```

1 Employment

```
library(haven)
library(dplyr)
library(rstanarm)
CPS <- as_factor(read_dta(dir(pattern = "^cepr_.*dta$")))
CPS <- filter(CPS, year == 2019, month == 12)
```

1.1 Selection

The general principles for drawing from a reasonable prior predictive distribution are

- Center the predictors, so that the intercept can be interpreted relative to a unit with average predictors
- Choose a prior distribution on the intercept that is roughly consistent with your marginal beliefs about the outcome, which is going to result in a slightly weaker prior on this conditional expectation
- Do not make the prior uncertainty on the coefficients too big

In addition, with any GLM (that does not use the identity inverse link function), you need to consider the (nonlinear) mapping from the linear predictor to the conditional expectation of the outcome. In the case of Bernoulli models, that can equivalently be thought of as using some CDF as the inverse link function or by considering the sign of the linear preditor plus a draw from the error distribution. In the case of count models, the conditional expectation is usually the antilog of the linear predictor, in which case the coefficients should be near zero and if so they can be interpreted as percentage changes.

In this case, perhaps the most important predictor of whether someone is in the labor force is how old they are, with young people often not being in the labor force because they are full-time students and old people often not being in the labor force because they are retired. Thus, labor force participation is related to age in perhaps a nonlinear fashion, and in addition, it differs substantially between men and women, particularly for younger women who are married and have children. Your model should at least attempt to capture these dynamics, in addition to the usual suspects, such as race and education that also affect the probability that someone has a job given that they are in the labor force.

```
CPS <- filter(CPS, !is.na(ownchild))
N <- nrow(CPS)

# pull out predictors
age <- CPS$age / 10 # in decades
female <- CPS$female
Black <- CPS$wbho == "Black"
Hispanic <- CPS$wbho == "Hispanic"
```

```r
Other_race <- CPS$wbho == "Other"
married <- CPS$married
ownchild <- CPS$ownchild
ch05 <- CPS$ch05
HS <- CPS$educ == "HS"
Some <- CPS$educ == "Some college"
College <- CPS$educ == "College"
Advanced <- CPS$educ == "Advanced"

# create interaction terms
age_female <- age * female
married_female <- married * female
ownchild_female <- ownchild * female
ch05_female <- ch05 * female
married_ownchild_female <- married * ownchild_female
married_ch05_female <- married * ch05_female

# center each variable
age <- age - mean(age)
female <- female - mean(female)
Black <- Black - mean(Black)
Hispanic <- Hispanic - mean(Hispanic)
Other_race <- Other_race - mean(Other_race)
married <- married - mean(married)
ownchild <- ownchild - mean(ownchild)
ch05 <- ch05 - mean(ch05)
HS <- HS - mean(HS)
Some <- Some - mean(Some)
College <- College - mean(College)
Advanced <- Advanced - mean(Advanced)
age_female <- age_female - mean(age_female)
married_female <- married_female - mean(married_female)
ownchild_female <- ownchild_female - mean(ownchild_female)
ch05_female <- ch05_female - mean(ch05_female)
married_ownchild_female <- married_ownchild_female - mean(married_ownchild_female)
married_ch05_female <- married_ch05_female - mean(married_ch05_female)

# chose an intercept location that implies about a 2 / 3 participation rate
mu_alpha_1 <- qnorm(0.72)
sd_alpha_1 <- 0.05

# create matrix of predictors for participation
X_1 <- cbind(age, female, Black, Hispanic, Other_race, married, ownchild, ch05,
             HS, Some, College, Advanced, age_female, married_female, ownchild_female,
             ch05_female, married_ownchild_female, married_ch05_female)
sd_beta_1 <- 0.25

# choose an intercept location that implies about 5% unemployment rate among people in the labor force
mu_alpha_2 <- qnorm(0.953)
sd_alpha_2 <- 0.01

# create matrix of predictors for employment, with fewer columns
X_2 <- cbind(age, female, Black, Hispanic, Other_race, married, ownchild, ch05,
```

```
             HS, Some, College, Advanced)
sd_beta_2 <- 0.25

y_ <- t(replicate(1000, {
  alpha_1 <- rnorm(n = 1, mean = mu_alpha_1, sd = sd_alpha_1)
  beta_1 <- rnorm(n = ncol(X_1), mean = 0, sd = sd_beta_1)
  eta_1 <- alpha_1 + c(X_1 %*% beta_1)
  errors_1 <- rnorm(n = N)

  rho <- runif(n = 1, min = 0, max = 0.9)

  alpha_2 <- rnorm(1, mean = mu_alpha_2, sd = sd_alpha_2)
  beta_2 <- rnorm(ncol(X_2), mean = 0, sd = sd_beta_2)
  eta_2 <- alpha_2 + c(X_2 %*% beta_2)
  errors_2 <- rnorm(N, mean = -rho * errors_1, sd = sqrt(1 - rho^2))

  return(ifelse((eta_1 + errors_1) < 0, -1, (eta_2 + errors_2) > 0))
}))
```

1.2 Prior Predictive Checking

```
proportions <- prop.table(table(y_))
1 - proportions[1] # participation rate
```

```
##        -1
## 0.6665481
```

```
proportions[2] / (1 - proportions[1]) # unemployment rate
```

```
##          0
## 0.08912627
```

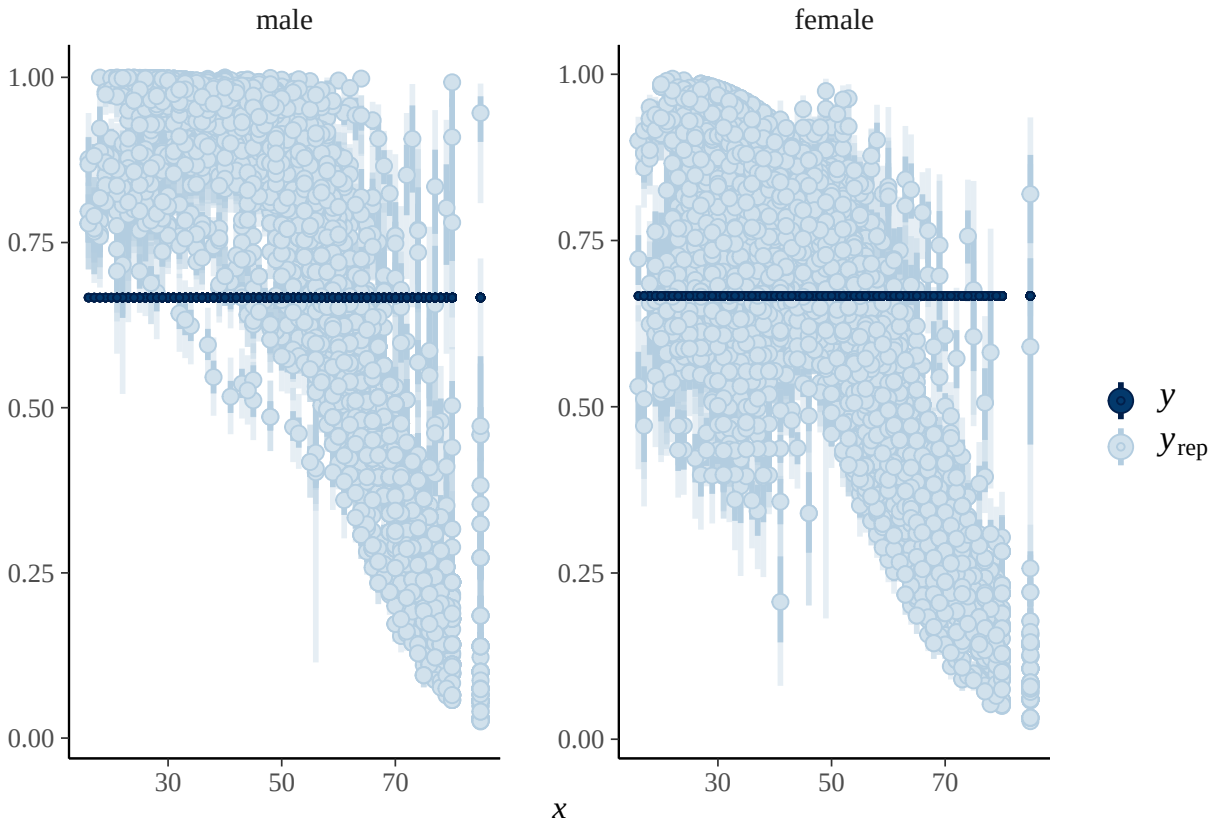
```
summary(colMeans(y_ == 1)) # probability each person has a job
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3950  0.5910  0.6110  0.6071  0.6280  0.6840
```

1.3 Probit Model

```
post_probit <- stan_glm(1 - nilf ~ age * female * married * ownchild * ch05 + wbho + educ,
                        data = CPS, family = binomial(link = "probit"),
                        prior_intercept = normal(location = mu_alpha_1, scale = sd_alpha_1),
                        prior = normal(location = 0, scale = sd_beta_1),
                        iter = 1000, init_r = 0.25, QR = TRUE)
```

```
library(bayesplot)
mu <- posterior_epred(post_probit)
mf <- model.frame(post_probit)
age <- as.array(mf$age)
sex <- factor(mf$female, levels = 0:1, labels = c("male", "female"))
ppc_intervals_grouped(rep(2/3, ncol(mu)), mu, x = age, group = sex)
```



1.4 Interpretation

Almost all of these are quite near 0 or 1, indicating that we are essentially certain about their signs conditional on these N observations (and the priors that we started with).

```
colMeans(as.matrix(post_probit) > 0)
```

```
##                   (Intercept)                           age
##                        1.0000                        0.0000
##                        female                       married
##                        0.9950                        1.0000
##                     ownchild                          ch05
##                        0.6850                        0.0355
##                     wbhoBlack                   wbhoHispanic
##                        0.0335                        0.0675
##                     wbhoOther                       educHS
##                        0.0005                        1.0000
##              educSome college                   educCollege
##                        1.0000                        1.0000
##                  educAdvanced                    age:female
##                        1.0000                        0.0025
##                   age:married                 female:married
##                        0.0000                        0.0000
##                  age:ownchild                female:ownchild
##                        0.7830                        0.0365
##             married:ownchild                       age:ch05
##                        0.0115                        0.9850
##                   female:ch05                   married:ch05
```

```
##                           0.4860                           0.6385
##                     ownchild:ch05              age:female:married
##                           0.8900                           1.0000
##              age:female:ownchild            age:married:ownchild
##                           0.8625                           0.9295
##          female:married:ownchild                age:female:ch05
##                           0.9330                           0.3370
##                 age:married:ch05             female:married:ch05
##                           0.1260                           0.3085
##                age:ownchild:ch05            female:ownchild:ch05
##                           0.0435                           0.5560
##             married:ownchild:ch05     age:female:married:ownchild
##                           0.3595                           0.1220
##          age:female:married:ch05        age:female:ownchild:ch05
##                           0.8165                           0.5680
##        age:married:ownchild:ch05    female:married:ownchild:ch05
##                           0.8530                           0.5265
## age:female:married:ownchild:ch05
##                           0.3005
```

2 Auto Insurance

```
UBI <- readr::read_csv("http://www2.math.uconn.edu/~valdez/telematics_syn-032021.csv")
UBI <- mutate(UBI, # create sane units for integer valued variables
              Insured.age = Insured.age / 10,
              Credit.score = Credit.score / 100,
              Annual.miles.drive = Annual.miles.drive / 10000,
              Years.noclaims = Years.noclaims / 10,
              Total.miles.driven = Total.miles.driven / 10000,
              Brake.06miles = Brake.06miles / 100,
              Brake.08miles = Brake.08miles / 100,
              Brake.09miles = Brake.09miles / 100,
              Brake.11miles = Brake.11miles / 100,
              Brake.12miles = Brake.12miles / 100,
              Brake.14miles = Brake.14miles / 100)

training <- UBI[1:50000, ]
testing  <- UBI[-(1:50000), ]
```

2.1 SMOTE

SMOTE is deterministic, unless there is some bootstrapping or K-folding involved to choose hyperparameters, whereas in order to draw from the prior predictive distribution of the outcomes, you first have to draw from the prior distribution of the unknown parameters and then condition on those realizations when drawing outcome data from their conditional distribution given the parameters. Both SMOTE and prior predictive checking have some reference to establish reasonableness, However, SMOTE would be more like using the *posterior* predictive distribution, although SMOTE still does not propagate uncertainty in the parameters to uncertainty in the predicted outcomes. Also, SMOTE involves getting something reasonable for both the outcomes and the predictors, whereas the Bayesian approaches always take the predictors as given and generate a predictive distribution of the outcomes only. Thus, a Bayesian approach might well be unacceptable in this case where you want to generate a synthetic dataset that does not disclose any private information about actual individuals or trade secrets of the auto insurance company. In order to get noise into the predictors in the synthetic dataset, you would have to model them, in which case they would not

be exclusively predictors anymore.

2.2 Bayesian Optimization

As mentioned in the Wikipedia page, the term "Bayesian optimization" originally dates to the 1970s and 1980s (although you have to be careful that some people today might mean something different by it). Thus, it does not involve MCMC, which is the hallmark of "modern" Bayesian analysis.

But it does involve putting a prior distribution over an unknown function, choosing hyperparameters via numerical optimization (rather than MCMC), and obtaining updated beliefs (albeit without uncertainty) about the value of the unknown function at a particular set of points, along with some procedure to predict the outcome at a new point. It is optimal with respect to its objective function, but not optimal for responsible scientific inference.

Thus, a more modern treatment would use something like `rstanarm::stan_gamm4` or `brms::brm` with `s()` to estimate a spline for the unknown function but use MCMC to obtain draws of all the unknowns conditional on the known data. Another choice that is mentioned in the Wikipedia article and can be done with Stan is to use a Gaussian process, which can be approximated by a spline (rstanarm) or a series of basis functions (brms).

2.3 Count Models

```
prior_poisson <- stan_glm(NB_Claim ~ Insured.age + Insured.sex + Credit.score + Annual.miles.drive +
                            Marital + Years.noclaims +
                            Pct.drive.wkend + `Pct.drive.rush am` + `Pct.drive.rush pm` +
                            Brake.06miles + Brake.08miles + Brake.09miles + Brake.11miles +
                            Brake.12miles + Brake.14miles,
                          data = training, family = poisson, offset = log(Duration),
                          prior_intercept = normal(location = -8, scale = 1),
                          prior = normal(location = 0, scale = 0.2, autoscale = TRUE),
                          QR = TRUE, seed = 20220411, prior_PD = TRUE, iter = 1000)
```

```
posterior_predict(prior_poisson, draws = 100, offset = log(training$Duration)) %>%
  table %>% prop.table %>% head(n = 5) %>% round(digits = 2)
```

```
## .
##    0    1    2    3    4
## 0.86 0.11 0.02 0.01 0.00
```

```
post_poisson <- update(prior_poisson, prior_PD = FALSE)
```

```
post_NB <- update(prior_poisson, family = neg_binomial_2,
                  prior_aux = exponential(rate = 0.1))
```

2.4 Posterior Predictive Checking of Count Models

The confusion matrix for the Poisson model is fine, which goes to show the futility of trying to ascertain whether a zero-inflated *model* is necessary from the observed number of zeros in the *marginal* outcome. Conditional on the predictors and the intercept, the *model* may well be predicting the zeros with the appropriate probability, but you have to actually estimate the model to find out.

The problem with the negative binomial model is not so much that it is predicting too few zeros, as it is that it is predicting way too many people having hundreds or even millions of car accidents (when the auto insurance company would have dropped such people after a few accidents).

```r
posterior_predict(post_poisson, draws = 1) %>%
  table(y = training$NB_Claim) %>%
  prop.table
```

```
##    y
## .         0       1       2       3
##   0 0.91446 0.03848 0.00180 0.00012
##   1 0.04146 0.00234 0.00016 0.00000
##   2 0.00110 0.00008 0.00000 0.00000
```

```r
posterior_predict(post_NB, draws = 1) %>%
  table(y = training$NB_Claim) %>%
  prop.table
```

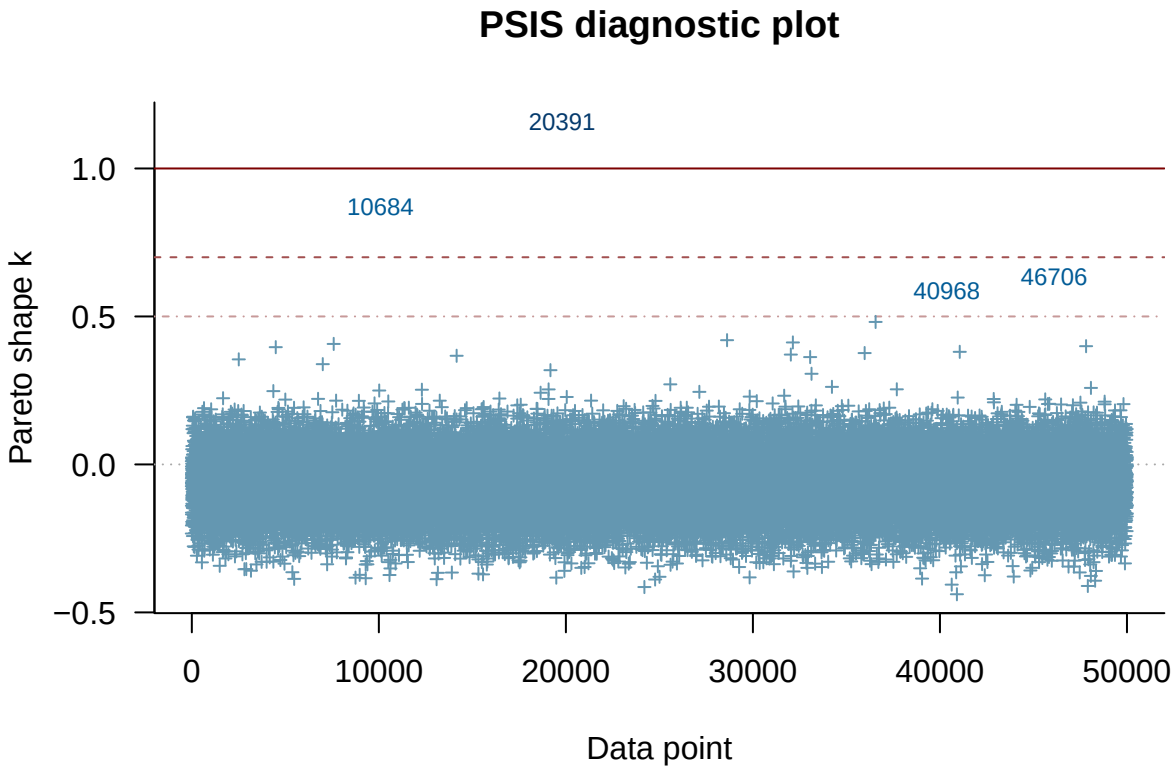
```
##                        y
## .                            0       1       2       3
##   0                    0.80338 0.03322 0.00168 0.00008
##   1                    0.13526 0.00692 0.00024 0.00004
##   2                    0.01520 0.00068 0.00004 0.00000
##   3                    0.00206 0.00006 0.00000 0.00000
##   4                    0.00026 0.00002 0.00000 0.00000
##   5                    0.00014 0.00000 0.00000 0.00000
##   6                    0.00006 0.00000 0.00000 0.00000
##   7                    0.00004 0.00000 0.00000 0.00000
##   11                   0.00002 0.00000 0.00000 0.00000
##   12                   0.00002 0.00000 0.00000 0.00000
##   21                   0.00002 0.00000 0.00000 0.00000
##   27                   0.00002 0.00000 0.00000 0.00000
##   31                   0.00002 0.00000 0.00000 0.00000
##   36                   0.00002 0.00000 0.00000 0.00000
##   40                   0.00002 0.00000 0.00000 0.00000
##   56                   0.00002 0.00000 0.00000 0.00000
##   63                   0.00004 0.00000 0.00000 0.00000
##   118                  0.00002 0.00000 0.00000 0.00000
##   155                  0.00002 0.00000 0.00000 0.00000
##   216                  0.00002 0.00000 0.00000 0.00000
##   294                  0.00002 0.00000 0.00000 0.00000
##   297                  0.00002 0.00000 0.00000 0.00000
##   305                  0.00002 0.00000 0.00000 0.00000
##   327                  0.00002 0.00000 0.00000 0.00000
##   391                  0.00002 0.00000 0.00000 0.00000
##   433                  0.00002 0.00000 0.00000 0.00000
##   778                  0.00002 0.00000 0.00000 0.00000
##   789                  0.00002 0.00000 0.00000 0.00000
##   1520                 0.00002 0.00000 0.00000 0.00000
##   2216                 0.00002 0.00000 0.00000 0.00000
##   3452                 0.00002 0.00000 0.00000 0.00000
##   126754               0.00002 0.00000 0.00000 0.00000
##   3150188              0.00002 0.00000 0.00000 0.00000
##   9190746              0.00002 0.00000 0.00000 0.00000
##   737943768            0.00002 0.00000 0.00000 0.00000
##   4875787976           0.00002 0.00000 0.00000 0.00000
##   164527644225709344   0.00002 0.00000 0.00000 0.00000
##   3032112036153881088  0.00002 0.00000 0.00000 0.00000
```

2.5 PSISLOOCV

Observations 10684 and 20391 bear further scrutiny, and perhaps a couple of others as well. But we can easily omit each of those points whose estimated Pareto k parameter is greater than 0.7 when refitting the model in order to estimate the ELPD without having to unnecessarily refit the model $49,998$ other times when those $49,998$ points individually have a small effect on the posterior distribution

```
plot(loo(post_poisson), label_points = TRUE)
```

```
## Warning: Found 2 observation(s) with a pareto_k > 0.7. We recommend calling 'loo' again with argumen
```




```
loo(post_poisson, k_threshold = 0.7)
```

```
## 2 problematic observation(s) found.
## Model will be refit 2 times.

##
## Fitting model 1 out of 2 (leaving out observation 10684)

##
## Fitting model 2 out of 2 (leaving out observation 20391)

##
## Computed from 2000 by 50000 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo  -8985.5 148.5
## p_loo        13.7   1.7
## looic     17971.0 297.0
## ------
## Monte Carlo SE of elpd_loo is 0.2.
##
## Pareto k diagnostic values:
```

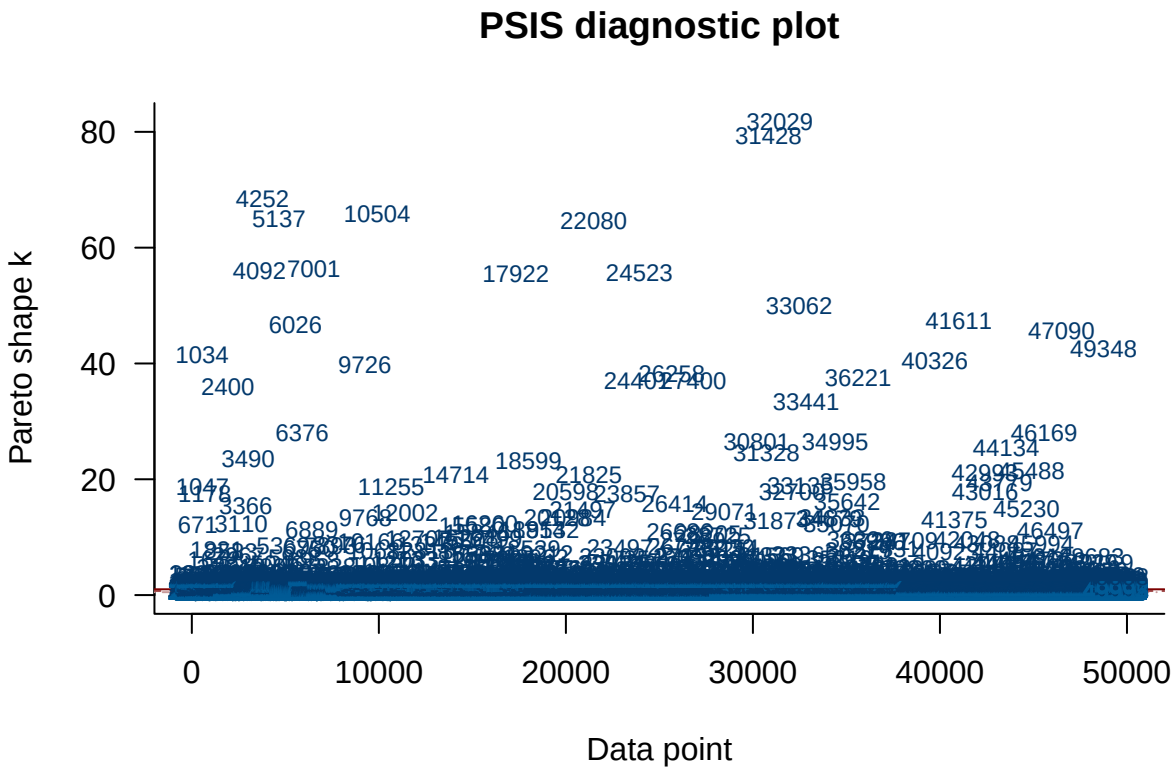
```
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     49996 100.0%  642
##  (0.5, 0.7]   (ok)           2   0.0%  705
##    (0.7, 1]   (bad)          0   0.0%  <NA>
##    (1, Inf)   (very bad)     0   0.0%  <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

In contrast, many of the points have high Pareto k estimates, so refitting the model omitting each of them in turn would take a long time and probably would not result in a good estimated ELPD anyway.

```
plot(loo(post_NB), label_points = TRUE)
```

```
## Warning: Found 21363 observations with a pareto_k > 0.7. With this many problematic observations we
```

```
## Warning in plot.psis_loo(loo(post_NB), label_points = TRUE): 0.038% of Pareto k
## estimates are Inf/NA/NaN and not plotted.
```




```
# loo(post_NB, k_threshold = 0.7)
```

2.6 ELPD

The ELPD *calculated* over the N observations in the `testing` data, is similar to the ELPD *estimated* from the N observations in the `training` data for the Poisson model only, especially when you take into account the estimated standard error of the estimated ELPD. This implies that splitting the original `UBI` dataset into `training` and `testing` was completely unnecessary for a Bayesian analysis, wasted the observations in `testing` that could have been used to further update your beliefs about the unknown parameters, and did not provide additional insight as to which model to prefer. Moreover, the fact that the calculated and estimated ELPDs were similar illustrates that the model did not badly overfit the data that was conditioned on once you consider the average of the fit metric over the posterior draws rather than calculating the fit

metric only at a mode.

```
log_lik(post_poisson, newdata = testing, offset = log(testing$Duration)) %>% rowSums %>% mean
```

```
## [1] -8877.303
```

```
log_lik(post_NB, newdata = testing, offset = log(testing$Duration)) %>% rowSums %>% mean
```

```
## [1] -16076.84
```

Moreover, an *estimated* ELPD with an estimated standard error is actually better than a *calculated* ELPD in the testing data, because the calculated value in the testing data could easily have been somewhat different had the observations that were randomly put into the testing data turned out differently. The estimated standard error in the estimated ELPD from the previous subproblem gives us a sense of the possible magnitude of the effect of randomizing observations into training and testing.

However, for the negative binomial model, the *calculated* ELPD over the N observations in the `testing` data differs from the *estimated* ELPD based on the N observations in the `training` data because the assumptions of the ELPD estimator were grossly violated by the many points that have an outsized influence on the posterior distribution. Nevertheless, the negative binomial model predicts much worse in this case than does the Poisson model.

2.7 Claim Models

```
prior_gamma <- stan_glm(AMT_Claim / NB_Claim ~ Insured.age + Insured.sex + Credit.score +
                          Car.age + Marital + Pct.drive.wkend +
                          `Pct.drive.rush am` + `Pct.drive.rush pm`,
                          data = training, family = Gamma(link = "log"), subset = AMT_Claim > 0,
                          prior_intercept = normal(location = 9, scale = 2),
                          prior = normal(location = 0, scale = 0.5, autoscale = TRUE),
                          QR = TRUE, seed = 20220411, prior_PD = TRUE, iter = 1000)
```

```
post_gamma <- update(prior_gamma, prior_PD = FALSE)
```

```
nd <- training
nd$Pct.drive.wkend <- 0
mu_0 <- posterior_epred(post_gamma, newdata = nd)
nd$Pct.drive.wkend <- 1
mu_1 <- posterior_epred(post_gamma, newdata = nd)
mu_diff <- mu_1 - mu_0
summary(c(mu_diff))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10677.0   -382.0    309.6    480.9   1157.3  23029.2
```

Thus, a person who only drives on weekends is expected to have an average claim that is $491 higher than a person who only drives on weekdays, but there is more than a one-quarter probability that the direction is negative. Consequently, the best answer is that we do not know with much certainty about whether weekend or weekday drivers have more costly accidents.

2.8 Premiums

The posterior predictive distribution of the previous subproblem could be used to estimate the expected damage of a claim. In other words, it would be a conditional expectation given that there is a claim. It does not consider the probability that a claim will happen, which would be necessary to calculate the marginal expectated damage of a policyholder.