Untitled

Miao Cai

5/28/2020

Aggregating ping data into trips

The self-defined function `segment_0()` below is used to separate the fake ping data into trips according to a threshold value.

```r
segment_0 = function(speed, threshold, time_diff) {
  speed1 = speed
  speed[time_diff >= threshold] <- 0
  r1 = rle(speed != 0)
  r1$values <- replicate(length(r1$values), 1)
  r1$values <- cumsum(r1$values)
  order_tmp <- inverse.rle(r1)
  dat_tmp1 <- data.table::data.table(speed, order_tmp, time_diff)
  dat_tmp2 <- dat_tmp1[,.(sumdiff = sum(time_diff)), by = order_tmp]
  r2 = rle(speed != 0); first_rle = r2$values[1]
  r2$values[r2$values == 0 & dat_tmp2$sumdiff < threshold] <- TRUE
  r2$values[1] = first_rle
  r2 <- rle(inverse.rle(r2))
  r2$values[r2$values] = cumsum(r2$values[r2$values])
  id = inverse.rle(r2)
  jump_speed = which(id == 0 & speed1 != 0)
  id[jump_speed] = id[jump_speed + 1]
  return(id)
}
```

Bayesian NB regression using `rstanarm`

The code below shows the code for Bayesian NB regression models. For demonstration purpose, we only use the first 1,000 observations of the data, 1 Markov chain with 1,000 iterations and the first 500 of them are warm-up iterations.

```r
pacman::p_load(rstanarm, broom)
fit <-
  stan_glm(
    crash ~ SCE + speed + age + gender + bus_unit + d_type,
    offset = log(distance / 1000),
    data = data,
    family = neg_binomial_2,
    prior = normal(0, 10),
    prior_intercept = normal(0, 10),
    QR = TRUE,
    iter = 4000,
    chains = 4,
    cores = 4,
    seed = 123
  )
```

```
broom::tidy(fit, intervals = TRUE, prob = 0.95) %>%
  mutate(estimate = exp(estimate),
         lower = exp(lower),
         upper = exp(upper)) %>%
  select(term, IRR = estimate, `95% CI left` = lower, `95% CI right` = upper) %>%
  knitr::kable(align = "c",
               caption = "Posterior estimates of Bayesian NB model.")
```

Model comparison and diagnostics using `loo`

```
prop_zero <- function(y) mean(y == 0)
pp_check(fit, plotfun = "stat", stat = "prop_zero")
```

The code above will give a figure showing the posterior predictive checks, which is a measure of the prediction accuracy. It compares the observed data to 100 replicated datasets generated from the posterior parameters distributions. For each simulated dataset, the proportion of zero crashes was computed, and the blue histograms shows the simulated distribution of the proportions. The black solid vertical lines are the observed proportion of zero crashes in observed data. When the observed proportion (black solid line) is near the center of the plot, it demonstrates good model fit.

```
(fit_loo = loo(fit))
```

The above block shows the expected log predicted density (`elpd_loo`), estimate number of parameters (`p_loo`), and the LOO Information Criterion (`looic`) for a new dataset from Pareto smoothed importance-sampling leave-one-out (PSIS-LOO) cross-validation (CV).

```
fit_new <- stan_glm(
  crash ~ SCE + speed + age + gender,
  offset = log(distance / 1000),
  data = data,
  family = neg_binomial_2,
  prior = normal(0, 10),
  prior_intercept = normal(0, 10),
  QR = TRUE,
  iter = 4000,
  chains = 4,
  cores = 4,
  seed = 123
)
fit_new_loo = loo(fit_new)
```

```
loo::compare(fit_loo, fit_new_loo)
```

With two model fits `fit` and `fit_new` above, researchers can compare the model fit using `compare()` from the `loo` package, as shown above. It compares the expected predictive accuracy by the difference in `elpd_loo`, with positive difference `elpd_diff` suggesting the second model while negative difference favoring the first model.