Simple Logistic

Andres Potapczynski (ap3635)

```
data <- read_delim(file = file, delim = '|')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   state = col_character(),
##   county = col_character(),
##   city = col_character(),
##   zip = col_integer(),
##   vendor_name = col_character(),
##   employer_name = col_character(),
##   age = col_integer(),
##   sex = col_character(),
##   sex_F = col_integer(),
##   condition_U = col_integer(),
##   y = col_integer(),
##   y2 = col_integer(),
##   y3 = col_integer(),
##   effective_pay = col_integer(),
##   vendor_Y = col_integer(),
##   employed_30 = col_integer(),
##   antiquity_20 = col_integer(),
##   credit_score = col_integer(),
##   days_wo_pay = col_integer(),
##   months_wo_pay = col_integer()
##   # ... with 12 more columns
## )
```

```
## See spec(...) for full column specifications.
```

```
# Sample the data
pct = 1
# pct = 0.1
# pct = 0.01
set.seed(seed = 42)
sample_size = round(pct * nrow(data))
sample <- sample(x = nrow(data), size = sample_size, replace = F)
data = data[sample, ]

## Selecting the relevant columns for the analysis
data_sub <- data %>% dplyr::select(
  state,
  city,
  county,
  zip,
  asset_market_value,
  mar_2_app,
  appraisal_value,
  app_2_inc,
  client_income,
```

```r
  mar_2_inc,
  age,
  sex_F,
  condition_U,
  y,
  y2)
summary(data_sub)
```

```
##     state               city              county               zip
##  Length:30499       Length:30499       Length:30499       Min.   : 1000
##  Class :character   Class :character   Class :character   1st Qu.:32680
##  Mode  :character   Mode  :character   Mode  :character   Median :55295
##                                                           Mean   :54236
##                                                           3rd Qu.:76148
##                                                           Max.   :99900
##  asset_market_value   mar_2_app       appraisal_value     app_2_inc
##  Min.   : 120000    Min.   : 0.9184   Min.   :  79521   Min.   :0.004633
##  1st Qu.: 350000    1st Qu.: 1.1259   1st Qu.: 302908   1st Qu.:0.088792
##  Median : 398000    Median : 1.2209   Median : 320052   Median :0.100663
##  Mean   : 491311    Mean   : 1.3378   Mean   : 371064   Mean   :0.098457
##  3rd Qu.: 463000    3rd Qu.: 1.3620   3rd Qu.: 348289   3rd Qu.:0.111070
##  Max.   :4519000    Max.   :21.8468   Max.   :1654602   Max.   :0.367728
##  client_income      mar_2_inc            age            sex_F
##  Min.   : 143.9   Min.   :0.02738   Min.   :18.00   Min.   :0.0000
##  1st Qu.: 284.3   1st Qu.:0.10934   1st Qu.:27.00   1st Qu.:0.0000
##  Median : 310.7   Median :0.12381   Median :32.00   Median :0.0000
##  Mean   : 409.8   Mean   :0.12856   Mean   :34.29   Mean   :0.3082
##  3rd Qu.: 341.7   3rd Qu.:0.13978   3rd Qu.:40.00   3rd Qu.:1.0000
##  Max.   :1887.2   Max.   :1.09440   Max.   :65.00   Max.   :1.0000
##   condition_U           y                 y2
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.0000   Median :0.00000   Median :0.00000
##  Mean   :0.3954   Mean   :0.06403   Mean   :0.03371
##  3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
```

```r
## Group data by state and define the IDs
state_summary <- data_sub %>%
  dplyr::select(state,
              client_income,
              appraisal_value,
              asset_market_value) %>%
  group_by(state) %>%
  summarize(n_state = n(),
            income_mean_state = mean(client_income),
            appraisal_mean_state = mean(appraisal_value),
            market_mean_state = mean(asset_market_value)) %>%
  arrange(desc(n_state)) %>%
  ungroup()

state_summary$ID_state = seq.int(nrow(state_summary))

## Group data by city and define the IDs
```

```r
city_summary <- data_sub %>%
  dplyr::select(city) %>%
  group_by(city) %>%
  summarize(n_city = n()) %>%
  arrange(desc(n_city)) %>%
  ungroup()

city_summary$ID_city = seq.int(nrow(city_summary))

## Merge back into data
data_sub <- data_sub %>%
  inner_join(y = state_summary, by = 'state') %>%
  inner_join(y = city_summary[, c('city', 'ID_city')], by = 'city')
```

```r
## Rescaling
inputs <- data_sub %>%
  mutate(

    income_st = (log(client_income) - mean(log(client_income))) /
      sd(log(client_income)),

    appraisal_st = (log(appraisal_value) - mean(log(appraisal_value))) /
      sd(log(appraisal_value)),

    market_st = (log(asset_market_value) - mean(log(asset_market_value))) /
      sd(log(asset_market_value)),

    market_state_st = (log(market_mean_state) - mean(log(market_mean_state))) /
      sd(log(market_mean_state)),

    income_state_st = (log(income_mean_state) - mean(log(income_mean_state))) /
      sd(log(income_mean_state)),

    appraisal_state_st = (log(appraisal_mean_state) -
                          mean(log(appraisal_mean_state))) /
      sd(log(appraisal_mean_state)),

    mar_2_inc_st = (mar_2_inc - mean(mar_2_inc)) / sd(mar_2_inc),
    app_2_inc_st = (app_2_inc - mean(app_2_inc)) / sd(app_2_inc),
    mar_2_app_st = (mar_2_app - mean(mar_2_app)) / sd(mar_2_app),
    age_st = (age - mean(age)) / sd(age)
         ) %>%
  dplyr::select(
    income_st,
    mar_2_inc_st,
    appraisal_st,
    app_2_inc_st,
    mar_2_app_st,
    market_st,
    age_st,
    income_state_st,
    market_state_st,
    appraisal_state_st,
    y,
```

```r
    y2
  )

## Train / Test split
set.seed(seed = 81989843)
pct_train = 0.9
sample_size = round(pct_train * nrow(inputs))
sample <- sample(x = nrow(inputs), size = sample_size, replace = F)

## Allocate train
y = inputs$y

inputs_train = inputs[sample, ]
y_train = y[sample]

## Allocate test
inputs_test = inputs[-sample, ]
y_test = y[-sample]

## Create inputs for STAN
data_stan_train <- list(N=nrow(inputs_train),
                        D=ncol(inputs_train),
                        X=inputs_train,
                        y=y_train)

## Inputs for STAN
X_train = inputs_train %>% dplyr::select(-y, -y2)
X_test = inputs_test %>% dplyr::select(-y, -y2)
N = nrow(X_train)
D = ncol(X_train)
S = length(unique(data_sub$state))
state = data_sub$ID_state[sample]

data_stan_train = list(N=N, D=D, S=S, state=state, X=X_train, y=y_train)
```

```
## Inference for Stan model: logistic_base_var_v02.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##           mean se_mean   sd  2.5%   50% 97.5% n_eff Rhat
## alpha    -2.69    0.00 0.07 -2.82 -2.69 -2.54  1181    1
## beta[1]   0.16    0.01 0.26 -0.35  0.16  0.64  1622    1
## beta[2]   0.20    0.00 0.10  0.01  0.20  0.39  2106    1
## beta[3]   0.55    0.00 0.19  0.17  0.55  0.92  1758    1
## beta[4]  -0.16    0.00 0.11 -0.38 -0.16  0.04  2155    1
## beta[5]   0.10    0.00 0.07 -0.06  0.10  0.22  2174    1
## beta[6]  -0.91    0.00 0.15 -1.20 -0.92 -0.61  3571    1
## beta[7]   0.02    0.00 0.03 -0.04  0.02  0.08  6232    1
## beta[8]   0.11    0.01 0.44 -0.77  0.12  0.95  1206    1
## beta[9]  -0.12    0.01 0.35 -0.79 -0.12  0.55  1463    1
## beta[10]  0.12    0.01 0.49 -0.84  0.11  1.08  1068    1
##
## Samples were drawn using NUTS(diag_e) at Sat Dec  1 10:33:37 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
```

```
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
sims <- rstan::extract(sm.logistic_v01)
alpha_median <- median(sims$alpha)
alpha_s_median <- apply(X = sims$alpha_s, MARGIN = 2, FUN = median)
beta_median <- apply(X = sims$beta, MARGIN = 2, FUN = median)
```

```r
threshold = 0.15

y_hat_baseline <- rep(0, times = length(y_test))

accuracy_base = sum(y_hat_baseline == y_test) / length(y_test)

cat('\nBaseline accuracy: ', accuracy_base * 100)
```

```
##
## Baseline accuracy:  93.54098
```

```r
ID_state_test = data_sub$ID_state[-sample]

proba_hat <- invlogit(as.matrix(X_test) %*% beta_median +
                        alpha_s_median[ID_state_test])
proba_hat <- as.numeric(proba_hat)

y_hat = rep(0, times = length(y_test))
y_hat[proba_hat > threshold] = 1

accuracy = sum(y_hat == y_test) / length(y_test)

cat('\nProba max: ', max(proba_hat))
```

```
##
## Proba max:  0.1814413
```

```r
cat('\nLogistic accuracy: ', accuracy * 100)
```

```
##
## Logistic accuracy:  93.40984
```

```r
cat('\nConfusion table\n')
```

```
##
## Confusion table
```

```r
print(table(y_test, y_hat))
```

```
##       y_hat
## y_test    0    1
##      0 2848    5
##      1  196    1
```

```r
test_df = data.frame(ID_state = data_sub$ID_state[-sample],
                        state = data_sub$state[-sample],
                        y_test = y_test,
                        y_hat = y_hat)
test_df <- test_df %>%
  group_by(state) %>%
```

```r
  summarize(y_sum_test = sum(y_test),
            y_sum_hat = sum(y_hat)) %>%
  arrange(desc(y_sum_test)) %>%
  ungroup()

accuracy_baseline = mean(abs(test_df$y_sum_test) ** 2)
accuracy = mean(abs(test_df$y_sum_hat - test_df$y_sum_test) ** 2)

cat('\nBaseline MSE: ', accuracy_baseline * 100)
```

```
##
## Baseline MSE:  8021.875
```

```r
cat('\nLogistic MSE: ', accuracy * 100)
```

```
##
## Logistic MSE:  8021.875
```

```r
test_df = data.frame(ID_city = data_sub$ID_city[-sample],
                     city = data_sub$city[-sample],
                     y_test = y_test,
                     y_hat = y_hat)
test_df <- test_df %>%
  group_by(city) %>%
  summarize(y_sum_test = sum(y_test),
            y_sum_hat = sum(y_hat)) %>%
  arrange(desc(y_sum_test)) %>%
  ungroup()

accuracy_baseline = mean(abs(test_df$y_sum_test) ** 2)
accuracy = mean(abs(test_df$y_sum_hat - test_df$y_sum_test) ** 2)

cat('\nBaseline MSE: ', accuracy_baseline * 100)
```

```
##
## Baseline MSE:  154.7425
```

```r
cat('\nLogistic MSE: ', accuracy * 100)
```

```
##
## Logistic MSE:  161.2466
```

```r
y_sum_train = sum(y_train)
y_sum_rep = apply(X = sims$y_rep, MARGIN = 1, FUN = sum)
cat('\nTotal training defaults: ', y_sum_train)
```

```
##
## Total training defaults:  1756
```

```r
cat('\nTotal replicated defaults: ', mean(y_sum_rep))
```

```
##
## Total replicated defaults:  1756.493
```