Biostatistical Methods I Final Project

Benjamin Goebel (bpg2118), Jesse Ames (jra2186), Louis Sharp (lzs2109), Brian Lee (jl6046)

12/17/2021

Let's define helpful functions.

```
# Functions

# Purpose: Calculates the Pearson's correlation coefficient between every
# variable in the data set and a specified variable.
# Arguments: v_name: a variable of type character that is the variable name
# Returns: A knitted table of correlations.
get_cor_by_var <- function(v_name) {
  cdi_slim %>%
  map(~cor(as.numeric(.x), pull(cdi, v_name), method = "pearson")) %>%
  as_tibble() %>%
  pivot_longer(CRM_1000:log_pop_density,
               names_to = "variables",
               values_to = "r") %>%
  mutate(
    sign = ifelse(r < 0, "-", "+"),
    r = abs(r)
  ) %>%
  arrange(desc(r)) %>%
  knitr::kable()
}

# Purpose: Fits the model and gets the model adjusted r-squared.
# Arguments: mod: a variable of type character that is the formula to fit a
#            linear model.
# Returns: A numeric, the model adjusted r-squared.
get_mod_adj_r_squared <- function(mod, data = cdi) {
  lm(mod, data = data) %>%
  broom::glance() %>%
  pull(adj.r.squared)
}

# Purpose: Performs cross validation on a model specified by its formula.
# Arguments: mod: a variable of type character that is the formula to fit a
#            linear model.
# Returns: A column vector of the model root mean squared errors generated by the validation procedure.
get_cv_rmse <- function(mod, data = cdi) {
  set.seed(1)
  crossv_mc(data, 1000) %>%
  mutate(
    train = map(train, as_tibble),
```

```r
    test = map(test, as_tibble)
  ) %>%
  mutate(
    fitted_mod = map(train, ~lm(mod, data = .x))
  ) %>%
  mutate(
    rmse_mod = map2_dbl(fitted_mod, test, ~rmse(model = .x, data = .y))
  ) %>%
  pull(rmse_mod)
}

# Purpose: Fits model.
#          Plots model residual as a function of model prediction for the given
#          model formula.
# Arguments: mod: a variable of type character that is the formula to fit a
#            linear model.
# Returns: The ggplot.
plot_model_residuals <- function(mod, data = cdi) {
  fitted_mod <- lm(mod, data = data)
  cdi %>%
  add_predictions(fitted_mod) %>%
  add_residuals(fitted_mod) %>%
  ggplot(aes(x = pred, y = resid)) +
  geom_point() +
  theme_bw() +
  labs(
    title = ""
  ) +
  theme(plot.title = element_text(hjust = 0.5))
}

# Purpose: Fits the specified model and creates a QQ Plot.
# Arguments: mod: a variable of type character that is the formula to fit a
#            linear model.
# Returns: The plot.
plot_mod_qq <- function(mod, data = cdi) {
  mod %>%
  lm(data = data) %>%
  plot(which = 2)
}


# Purpose: Fits the specified model and creates a leverage plot.
# Arguments: mod: a variable of type character that is the formula to fit a
#            linear model.
# Returns: The plot.
plot_mod_leverage <- function(mod, data = cdi) {
  mod %>%
  lm(data = data) %>%
  plot(which = 5)
}
```

Let's begin by reading in the data and adding a column for the crime rate per 1,000 people in the county population. We will name this column CRM_1000. We will then recode the region variable as a factor.

```
cdi <- read_csv(here::here("data", "cdi.csv")) %>%
  mutate(CRM_1000 = (crimes/pop) * 1000,
         state = as.factor(state),
         region = as.factor(region),
         region = fct_recode(region, "Northeast" = "1", "North Central" = "2",
                             "South" = "3", "West" = "4"),
         pop_density = pop/area,
         log_pop18 = log(pop18),
         log_pop65 = log(pop65),
         log_hsgrad = log(hsgrad),
         log_bagrad = log(bagrad),
         log_poverty = log(poverty),
         log_unemp = log(unemp),
         log_totalinc = log(totalinc),
         log_pcincome = log(pcincome),
         log_pop_density = log(pop_density)
         ) %>%
  dplyr::select(-id, -state, -cty, -docs, -beds, -crimes, -pop, -area) %>%
  dplyr::select(CRM_1000, region, everything())
```

Let's compare each variable's distribution in the cdi dataset as is and log-transformed to see which is more normal.

```
cdi_long =
  cdi %>%
    pivot_longer(
      pop18:log_pop_density,
      names_to = "var",
      values_to = "val"
    ) %>%
  mutate(
    trans = case_when(grepl('log_', var) ~ 'log',
                      TRUE ~ 'ori'
    ),
    trans = factor(trans, levels = c("ori", "log"), labels =c("Untransformed", "log Transformed")),
    var = str_replace(var, "log_", ""),
    var = factor(var, levels = c("pop18", "pop65", "hsgrad", "bagrad", "poverty", "unemp", "pcincome",
  )

dens_plot_gen = function(v, b){

  if(!b){
    cdi_long %>%
      filter(var == v) %>%
      ggplot(aes(x = val)) +
      geom_boxplot() +
      theme_minimal() +
      theme(
        axis.title.y=element_blank(),
        axis.title.x=element_blank(),
        axis.text.y=element_blank(),
        axis.text.x=element_blank()
```

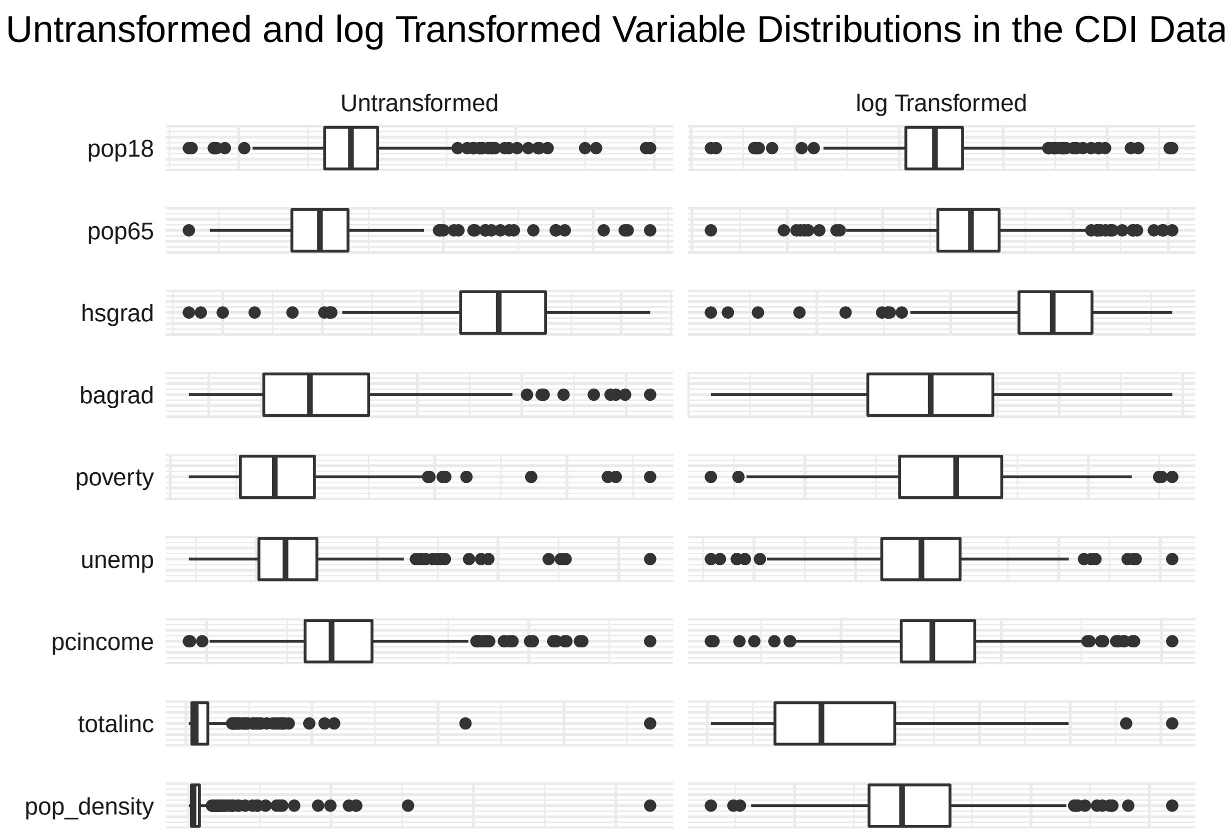
```
      ) +
      facet_grid(var ~ trans, switch = "y", scales = "free") +
      theme(
        strip.text.y.left = element_text(angle = 0),
        strip.text.x = element_blank()
      )
  }else{
    cdi_long %>%
      filter(var == v) %>%
      ggplot(aes(x = val)) +
      geom_boxplot() +
      theme_minimal() +
      theme(
        axis.title.y=element_blank(),
        axis.title.x=element_blank(),
        axis.text.y=element_blank(),
        axis.text.x=element_blank()
      ) +
      facet_grid(var ~ trans, switch = "y", scales = "free") +
      theme(
        strip.text.y.left = element_text(angle = 0)
      )
  }
}

var_list = c("pop18", "pop65", "hsgrad", "bagrad", "poverty", "unemp", "pcincome", "totalinc", "pop_dens

plot_1 = dens_plot_gen(var_list[1], T)
plot_2 = dens_plot_gen(var_list[2], F)
plot_3 = dens_plot_gen(var_list[3], F)
plot_4 = dens_plot_gen(var_list[4], F)
plot_5 = dens_plot_gen(var_list[5], F)
plot_6 = dens_plot_gen(var_list[6], F)
plot_7 = dens_plot_gen(var_list[7], F)
plot_8 = dens_plot_gen(var_list[8], F)
plot_9 = dens_plot_gen(var_list[9], F)

patch_1 = plot_1 / plot_2 / plot_3 / plot_4 / plot_5 / plot_6 / plot_7 / plot_8 / plot_9

patch_1 +
  plot_annotation(title = 'Untransformed and log Transformed Variable Distributions in the CDI Dataset'
                  theme = theme(plot.title = element_text(size = 14)))
```



Based on this boxplot, we decided to log transform all continuous variables except high school graduation. We can remove all unused variables now.

```
cdi_slim <-
  cdi %>%
  select(-c(pop18, pop65, bagrad, poverty, unemp, pcincome, totalinc,
            pop_density, log_hsgrad))
```

Now, let's calculate the Pearson's correlation coefficient between every variable in the data set and `CRM_1000`.

```
get_cor_by_var("CRM_1000")
```

variables	r	sign
CRM_1000	1.0000000	+
log_poverty	0.4823623	+
region	0.3427584	+
log_pop_density	0.3367361	+
log_totalinc	0.3273042	+
hsgrad	0.2264129	-
log_pop18	0.2039079	+
log_pcincome	0.0695287	-
log_bagrad	0.0632119	+
log_pop65	0.0543376	-
log_unemp	0.0362602	+

Next, we can perform stepwise selection based on AIC.

```
# keep only the predictors needed to build the model using pcincome as income measure
pcincome_model =
  cdi %>%
  dplyr::select(-c(pop18, pop65, bagrad, poverty, unemp, pcincome, totalinc, log_totalinc, pop_density,

# get first full mlr
fit_1 = lm(CRM_1000 ~ ., data = pcincome_model)
## summary(fit_1) # aRs = 0.5309
## boxcox(fit_1) # we ignore the square root tranformation for a more sensible interpretation

# run stepwise and get a list of highly effective predictors
step(fit_1, direction = 'both')
```

```
## Start:  AIC=2588.89
## CRM_1000 ~ region + hsgrad + log_pop18 + log_pop65 + log_bagrad +
##     log_poverty + log_unemp + log_pcincome + log_pop_density
##
##                   Df Sum of Sq    RSS    AIC
## - hsgrad           1       7.3 149659 2586.9
## - log_pop65        1      21.0 149672 2586.9
## - log_bagrad       1      41.6 149693 2587.0
## - log_unemp        1     150.4 149802 2587.3
## <none>                         149651 2588.9
## - log_pop18        1     860.1 150512 2589.4
## - log_pcincome     1    4505.8 154157 2599.9
## - log_pop_density  1   18265.0 167916 2637.6
## - region           3   24405.2 174057 2649.4
## - log_poverty      1   30268.1 179920 2667.9
##
## Step:  AIC=2586.91
## CRM_1000 ~ region + log_pop18 + log_pop65 + log_bagrad + log_poverty +
##     log_unemp + log_pcincome + log_pop_density
##
##                   Df Sum of Sq    RSS    AIC
## - log_pop65        1        21 149680 2585.0
## - log_bagrad       1        35 149694 2585.0
## - log_unemp        1       164 149823 2585.4
## <none>                         149659 2586.9
## - log_pop18        1       875 150534 2587.5
## + hsgrad           1         7 149651 2588.9
## - log_pcincome     1      4702 154361 2598.5
## - log_pop_density  1     18484 168143 2636.2
## - region           3     24678 174337 2648.1
## - log_poverty      1     38495 188154 2685.6
##
## Step:  AIC=2584.97
## CRM_1000 ~ region + log_pop18 + log_bagrad + log_poverty + log_unemp +
##     log_pcincome + log_pop_density
##
##                   Df Sum of Sq    RSS    AIC
## - log_bagrad       1        42 149721 2583.1
## - log_unemp        1       174 149853 2583.5
```

```
## <none>                           149680 2585.0
## + log_pop65        1        21 149659 2586.9
## + hsgrad           1         7 149672 2586.9
## - log_pop18        1      1472 151152 2587.3
## - log_pcincome     1      4829 154509 2596.9
## - log_pop_density  1     18551 168230 2634.4
## - region           3     27888 177568 2654.2
## - log_poverty      1     44711 194390 2698.0
##
## Step:  AIC=2583.09
## CRM_1000 ~ region + log_pop18 + log_poverty + log_unemp + log_pcincome +
##     log_pop_density
##
##                   Df Sum of Sq    RSS    AIC
## - log_unemp        1       132 149854 2581.5
## <none>                           149721 2583.1
## + log_bagrad       1        42 149680 2585.0
## + log_pop65        1        27 149694 2585.0
## + hsgrad           1         1 149720 2585.1
## - log_pop18        1      2731 152452 2589.1
## - log_pcincome     1      8413 158134 2605.2
## - log_pop_density  1     18534 168255 2632.4
## - region           3     29058 178780 2655.1
## - log_poverty      1     44988 194710 2696.7
##
## Step:  AIC=2581.48
## CRM_1000 ~ region + log_pop18 + log_poverty + log_pcincome +
##     log_pop_density
##
##                   Df Sum of Sq    RSS    AIC
## <none>                           149854 2581.5
## + log_unemp        1       132 149721 2583.1
## + log_pop65        1        31 149823 2583.4
## + hsgrad           1        10 149844 2583.4
## + log_bagrad       1         1 149853 2583.5
## - log_pop18        1      2657 152511 2587.2
## - log_pcincome     1      8281 158135 2603.2
## - log_pop_density  1     18596 168450 2630.9
## - region           3     31014 180868 2658.2
## - log_poverty      1     51449 201303 2709.3
```

```
##
## Call:
## lm(formula = CRM_1000 ~ region + log_pop18 + log_poverty + log_pcincome +
##     log_pop_density, data = pcincome_model)
##
## Coefficients:
##         (Intercept)  regionNorth Central          regionSouth
##            -495.713                8.469               22.131
##          regionWest            log_pop18          log_poverty
##              20.054               18.008               31.676
##        log_pcincome      log_pop_density
##              37.688                7.655
```

```r
fit_2 = lm(CRM_1000 ~ region + log_pop18 + log_poverty + log_pcincome + log_pop_density, data = pcincome
## summary(fit_2) # aRs = 0.5355; improved with 8 significant coefs
## plot(fit_2) # rows 6, 215, 371 seem to contain outliers; treat them as influential observations...

# check collinearity and found low correlation
## check_collinearity(fit_2)

fit_3 = lm(CRM_1000 ~ (region + log_pop18 + log_poverty + log_pcincome + log_pop_density)*(region + log_
## summary(fit_3) # aRs = 0.5748; improved, but each predictor doesn't seem as significant
fit_4 = lm(CRM_1000 ~ (region + log_pop18 + log_poverty + log_pcincome + log_pop_density)*region, data =
## summary(fit_4) # aRs = 0.5437, 6 significant coefs
fit_5 = lm(CRM_1000 ~ (region + log_pop18 + log_poverty + log_pcincome + log_pop_density)*log_pop18, dat
## summary(fit_5) # aRs = 0.5358, 2 significant coefs
fit_6 = lm(CRM_1000 ~ (region + log_pop18 + log_poverty + log_pcincome + log_pop_density)*log_poverty, d
## summary(fit_6) # aRs = 0.562, 6 significant coefs
fit_7 = lm(CRM_1000 ~ (region + log_pop18 + log_poverty + log_pcincome + log_pop_density)*log_pcincome,
## summary(fit_7) # aRs = 0.5553, 4 significant coefs
fit_8 = lm(CRM_1000 ~ (region + log_pop18 + log_poverty + log_pcincome + log_pop_density)*log_pop_densi
## summary(fit_8) # aRs = 0.5717, 8 significant coefs
```

We can also perform backward elimination based on BIC.

```r
totalincome_model_df =
  cdi %>%
  dplyr::select(-c(pop18, pop65, bagrad, poverty, unemp, pcincome, totalinc, log_pcincome, pop_density,
lin_transform <- lm(CRM_1000 ~., data = totalincome_model_df)
bic_model5 <- step(lin_transform, direction = "backward", trace = FALSE,
                   k = log(nobs(lin_transform)))
summary(bic_model5)
```

```
##
## Call:
## lm(formula = CRM_1000 ~ region + log_bagrad + log_poverty + log_totalinc +
##     log_pop_density, data = totalincome_model_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -43.370  -9.973  -0.441   8.826 186.410
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -130.622     13.032 -10.023  < 2e-16 ***
## regionNorth Central    8.804      2.603   3.383 0.000783 ***
## regionSouth           21.386      2.576   8.304 1.31e-15 ***
## regionWest            15.723      3.222   4.880 1.50e-06 ***
## log_bagrad             9.061      3.244   2.794 0.005444 **
## log_poverty           26.620      2.078  12.810  < 2e-16 ***
## log_totalinc           6.907      1.604   4.307 2.05e-05 ***
## log_pop_density        6.038      1.252   4.823 1.96e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.56 on 432 degrees of freedom
```

```
## Multiple R-squared:  0.546,  Adjusted R-squared:  0.5387
## F-statistic: 74.23 on 7 and 432 DF,  p-value: < 2.2e-16
```

Now, let's define our candidate models.

```
# Define model formulas and put in a list
model_A <- "CRM_1000 ~ region + log_pop_density + log_totalinc + log_bagrad + log_poverty"
model_B <- "CRM_1000 ~ region + log_pop_density + log_totalinc + log_poverty"
model_C <- "CRM_1000 ~ region + log_pop_density + log_pcincome + log_poverty"


model_list <-
  list(
    model_A = model_A,
    model_B = model_B,
    model_C = model_C
  )
```

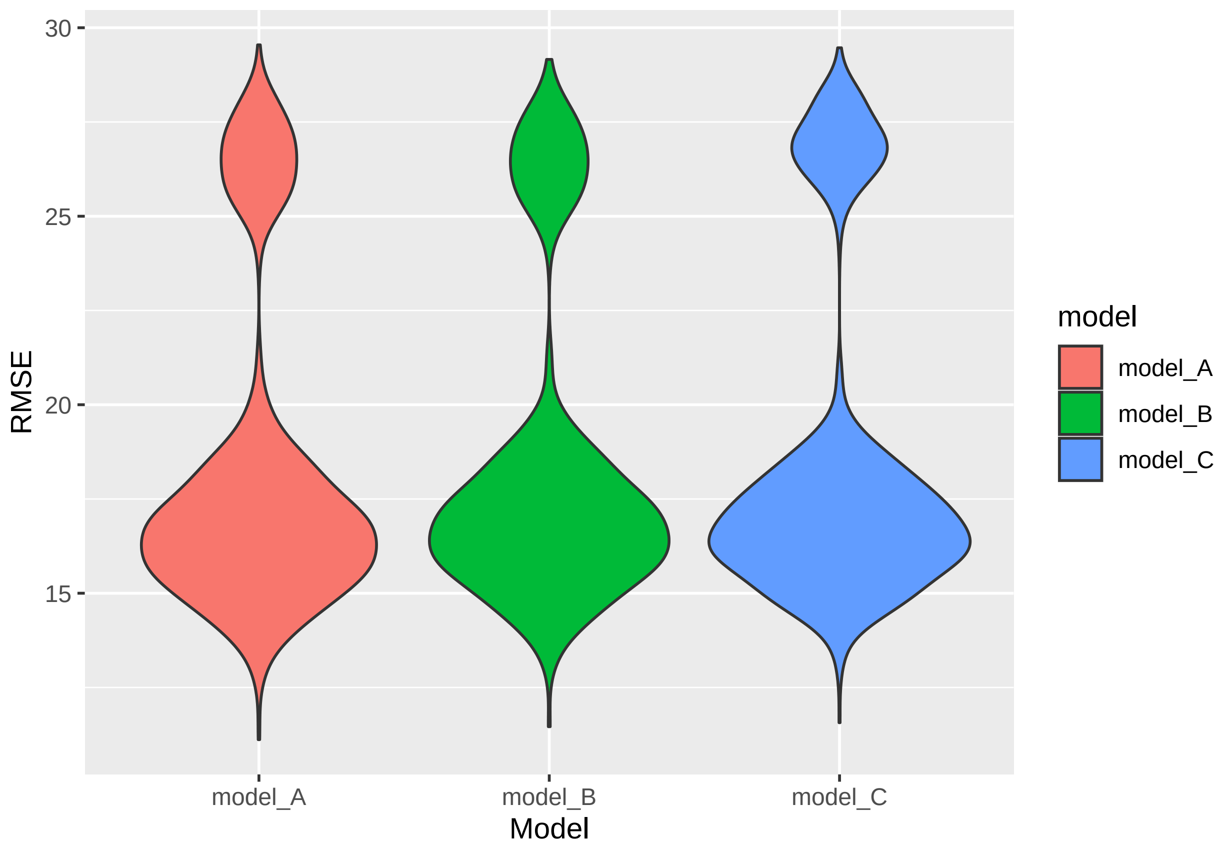
Here is each model's adjusted R-squared value.

```
# Get each model's adjusted r-squared
map(model_list, get_mod_adj_r_squared) %>%
  as_tibble() %>%
  pivot_longer(model_A:model_C,
               names_to = "model",
               values_to = "adj_r_squared") %>%
  arrange(desc(adj_r_squared)) %>%
  knitr::kable()
```

model	adj_r_squared
model_A	0.5386912
model_B	0.5314421
model_C	0.5283651

Here is the distribution of each model's cross-validation root mean squared error.

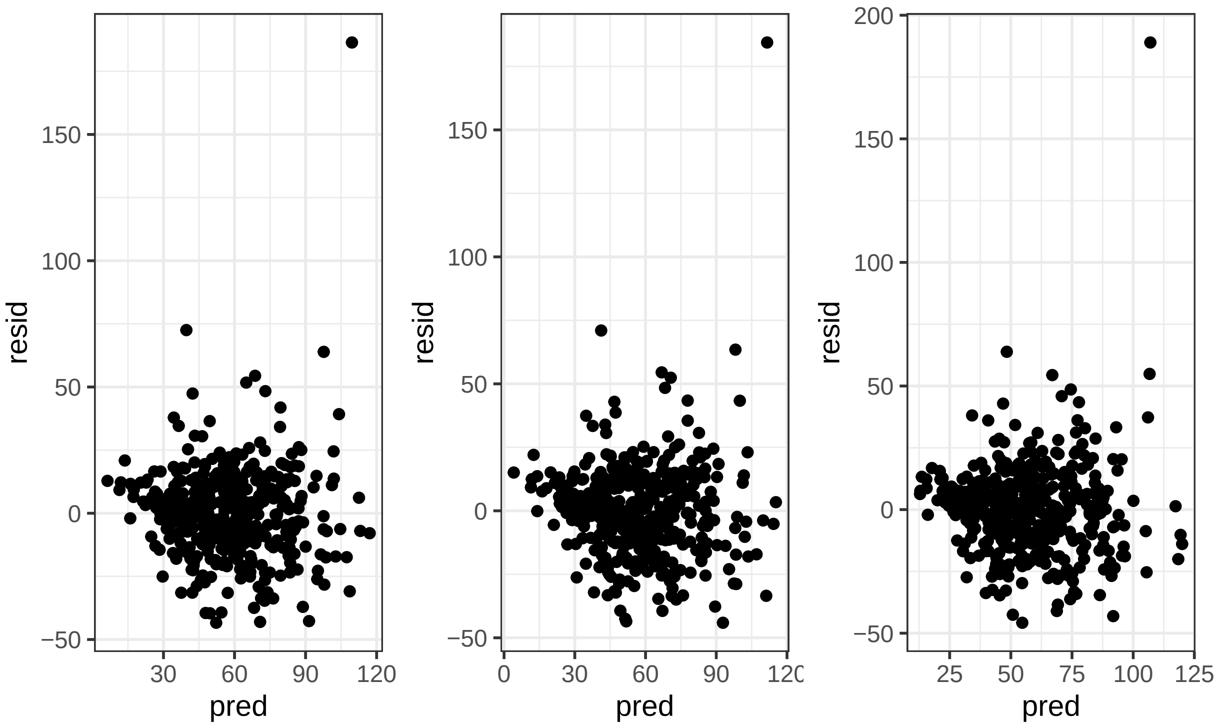
```
# Perform cross validation for each model
map(model_list, get_cv_rmse) %>%
  as_tibble() %>%
  pivot_longer(model_A:model_C,
               names_to = "model",
               values_to = "RMSE") %>%
  arrange(RMSE) %>%
  ggplot(aes(x = model, y = RMSE, fill = model)) +
  geom_violin() + labs(x = "Model", y = "RMSE")
```



Further, we can plot the model residuals as a function of the model predictions.

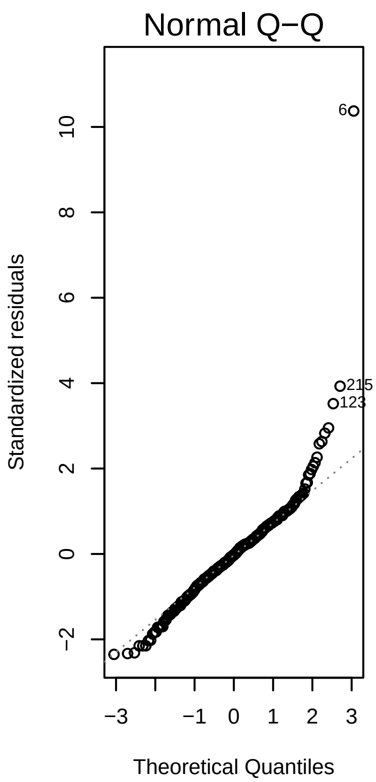
```
plot_model_residuals(model_A) +
plot_model_residuals(model_B) +
plot_model_residuals(model_C) +
plot_annotation(title = 'Model Residual as a function of Model Prediction',
                theme = theme(plot.title = element_text(size = 14)))
```

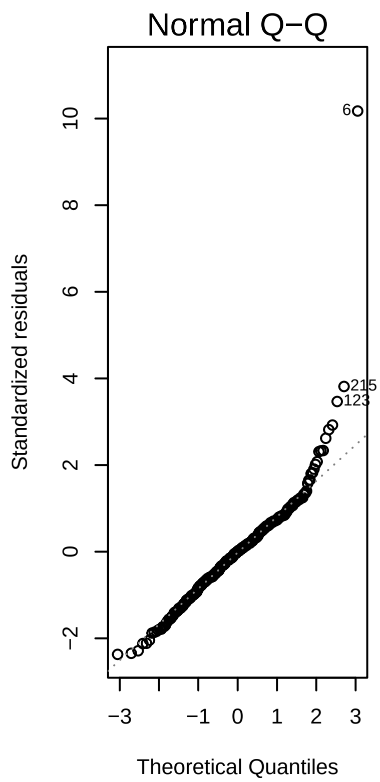
Model Residual as a function of Model Prediction

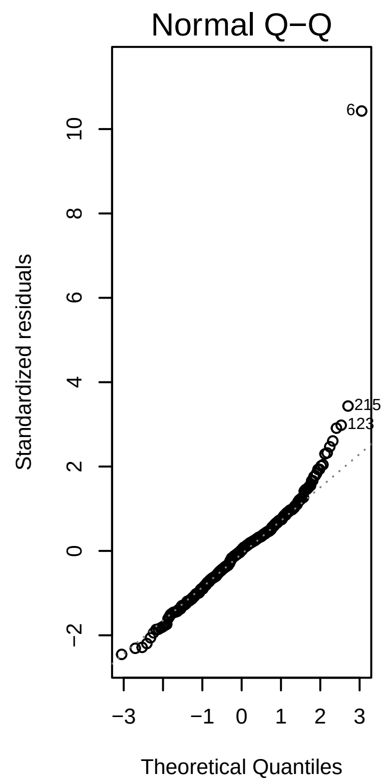


Q-Q plots

```
par(mfrow = c(1,3))
plot_mod_qq(model_A)
plot_mod_qq(model_B)
plot_mod_qq(model_C)
```

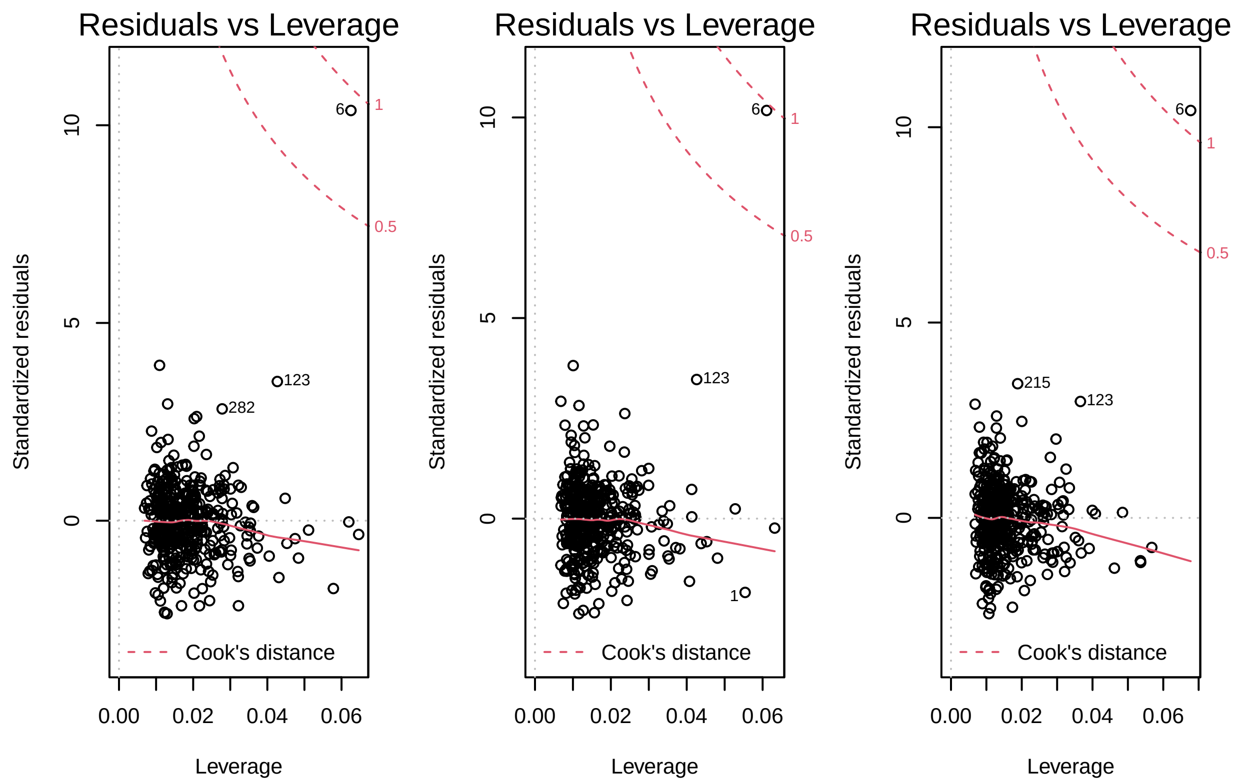






Leverage plots

```
par(mfrow = c(1,3))
plot_mod_leverage(model_A)
plot_mod_leverage(model_B)
plot_mod_leverage(model_C)
```



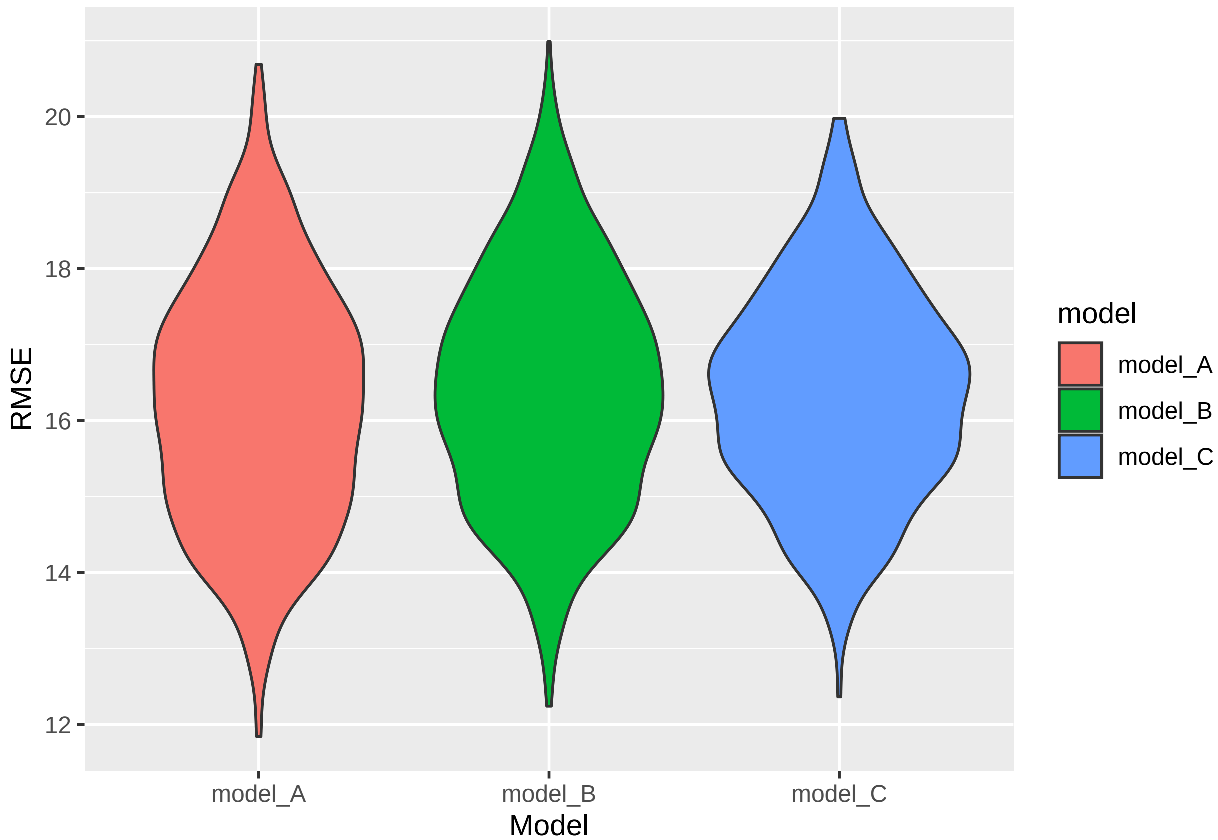
No wonder we had those weird distributions of RMSE in the Monte Carlo simulations - we had a serious outlier. Let's remove it and refit and re-validate the models.

```
cdi_2 <- cdi %>% slice(-6)
# Get each model's adjusted r-squared
map(model_list, get_mod_adj_r_squared, data = cdi_2) %>%
  as_tibble() %>%
  pivot_longer(model_A:model_C,
               names_to = "model",
               values_to = "adj_r_squared") %>%
  arrange(desc(adj_r_squared)) %>%
  knitr::kable()
```

model	adj_r_squared
model_A	0.5804990
model_C	0.5723062
model_B	0.5682312

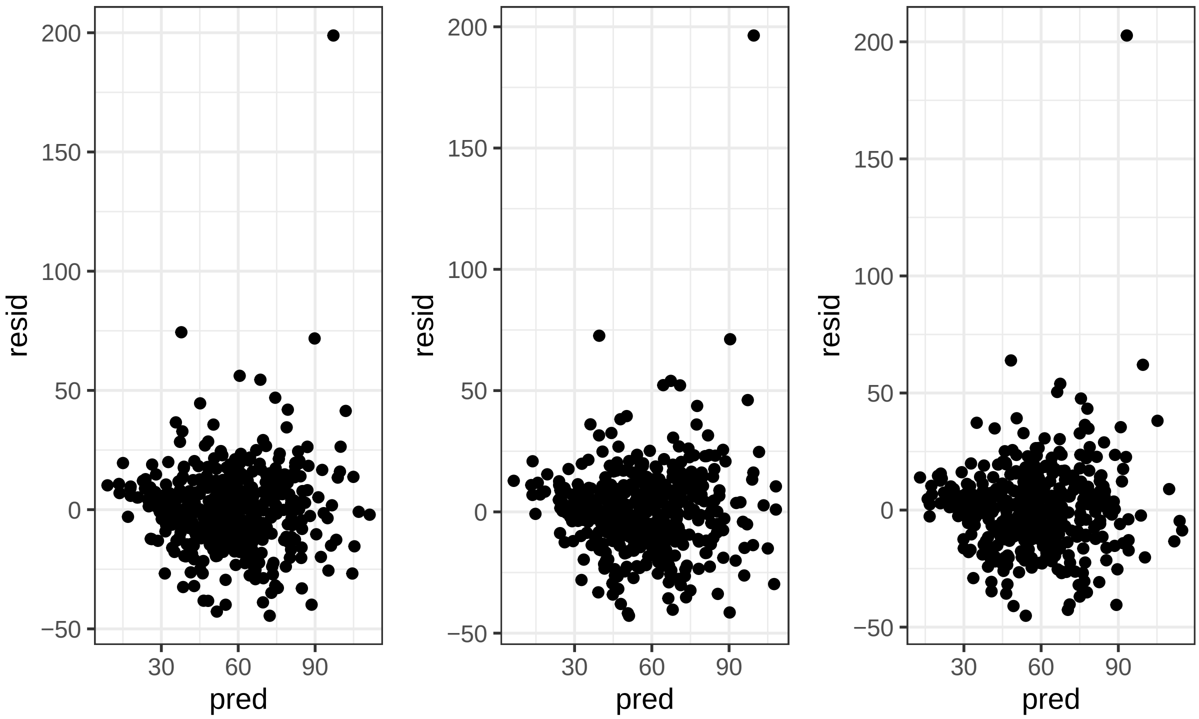
```
# Perform cross validation for each model
map(model_list, get_cv_rmse, data = cdi_2) %>%
  as_tibble() %>%
  pivot_longer(model_A:model_C,
               names_to = "model",
               values_to = "RMSE") %>%
```

```
  arrange(RMSE) %>%
  ggplot(aes(x = model, y = RMSE, fill = model)) +
  geom_violin() + labs(x = "Model", y = "RMSE")
```

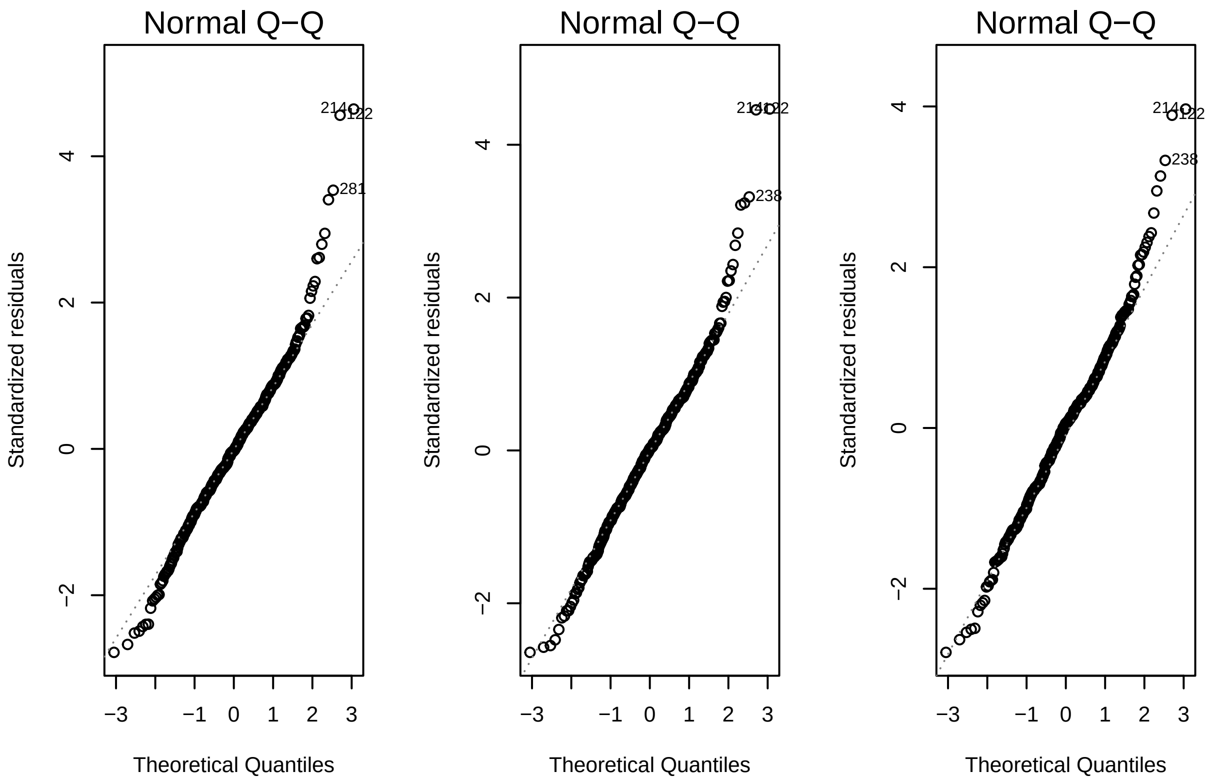



```
#Residual plots
plot_model_residuals(model_A, data = cdi_2) +
plot_model_residuals(model_B, data = cdi_2) +
plot_model_residuals(model_C, data = cdi_2) +
plot_annotation(title = 'Model Residual as a function of Model Prediction',
                theme = theme(plot.title = element_text(size = 14)))
```

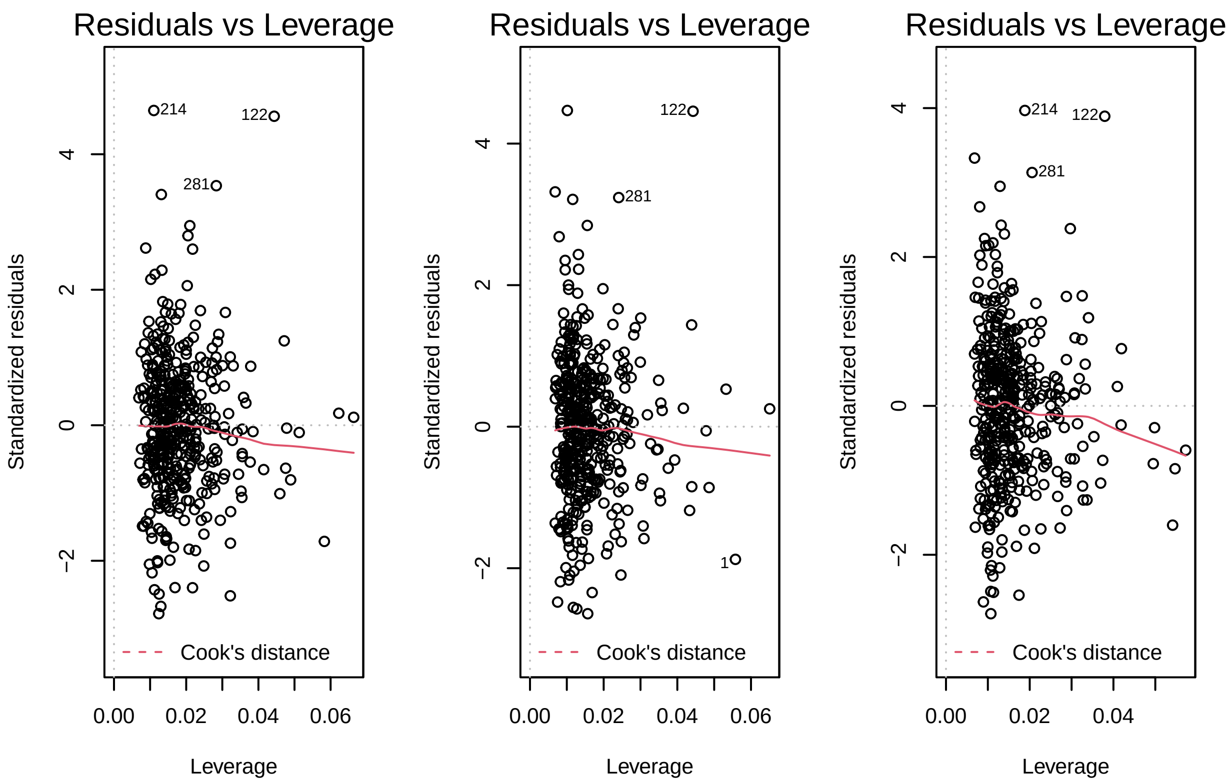
Model Residual as a function of Model Prediction




```r
#Q-Q plots
par(mfrow = c(1,3))
plot_mod_qq(model_A, data = cdi_2)
plot_mod_qq(model_B, data = cdi_2)
plot_mod_qq(model_C, data = cdi_2)
```




```
#Leverage plots
plot_mod_leverage(model_A, data = cdi_2)
plot_mod_leverage(model_B, data = cdi_2)
plot_mod_leverage(model_C, data = cdi_2)
```



For reasons listed in the paper, we will use model B. Let's summarize the model.

```
broom::tidy(lm(model_B, data = cdi_2)) %>%
  knitr::kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	-98.510093	9.181019	-10.729756	0.00e+00
regionNorth Central	11.058492	2.298291	4.811615	2.10e-06
regionSouth	25.572291	2.235019	11.441643	0.00e+00
regionWest	18.224954	2.786970	6.539343	0.00e+00
log_pop_density	4.555372	1.104612	4.123956	4.47e-05
log_totalinc	8.316132	1.386287	5.998855	0.00e+00
log_poverty	21.188450	1.605139	13.200384	0.00e+00