# Forward, Backward Subset Selection - MSE

*Quinton Neville*

*12/8/2018*

## Load, clean, manipulate, and tidy the data

```
# Import data
cancer_raw = readr::read_csv("./Data/Cancer_Registry.csv") %>%
  janitor::clean_names()

#dim(cancer_raw)
#head(cancer_raw)

# Check NA values for each column
#n_NA = sapply(cancer_raw[1:34], function(x) sum(length(which(is.na(x)))))
#n_NA

# Check the percentage of NA values for each column
#percentage_NA = sapply(cancer_raw[1:34], function(x) sum(length(which(is.na(x)))) / 3047)
#percentage_NA %>% data.frame()

#Pulling quartiles for study_per_cap categorical manipulation
study.quart <- with(cancer_raw, study_per_cap[study_per_cap > 0]) %>%
  quantile(., probs = c(0.25, 0.5, 0.75))

#Variable Manipulation
cancer.df <- cancer_raw %>%    #Remove Rows with > 20% missing
  dplyr::select(-pct_some_col18_24) %>%  #Remove for too many missing
  mutate(
    pct_non_white = pct_black + pct_asian + pct_other_race, #Creating white, non-white percentages vari
    state = str_split_fixed(geography, ", ", 2)[ ,2] %>% as.factor(), #pulling state variable and casti
    binned_inc_lb = str_split_fixed(binned_inc, ", ", 2)[ ,1] %>% parse_number(), #pulling numeric lowe
    binned_inc_ub = str_split_fixed(binned_inc, ", ", 2)[ ,2] %>% parse_number(), #pulling numeric uppe
    binned_inc_point = (binned_inc_lb + binned_inc_ub)/2, #computing point estimate from ub,lb (interva
    study_quantile = ifelse(study_per_cap == 0, "None",
                            ifelse(study_per_cap > 0 & study_per_cap <= study.quart[1], "Low",
                                   ifelse(study_per_cap > study.quart[1] & study_per_cap <= study.quart[
                                          ifelse(study_per_cap > study.quart[2] & study_per_cap <= study
                                                 "Very High")))),
    study_quantile = as.factor(study_quantile) %>% fct_relevel(., "None", "Low", "Moderate", "High", "V
    avg_deaths_yr_pop = avg_deaths_per_year/pop_est2015,  #incorporate two vars into one (multicollinea
    avg_ann_count_pop = avg_ann_count/pop_est2015 #incorporate two vars into one (multicollinearity)
  ) %>%
  dplyr::select(-c(binned_inc, geography, study_per_cap))

######TO avoid singular matrixes for our model fits, we need remove highly correlated and/or direct lin

cancer.df <- cancer.df %>%
  dplyr::select(-c(avg_deaths_yr_pop, avg_ann_count_pop, pct_non_white, binned_inc_point))
```

## Imputing Values with less than 20% missing (two variables)

- pct_employed16_over ~ 4%
- pct_private_coverage_alone ~ 20%

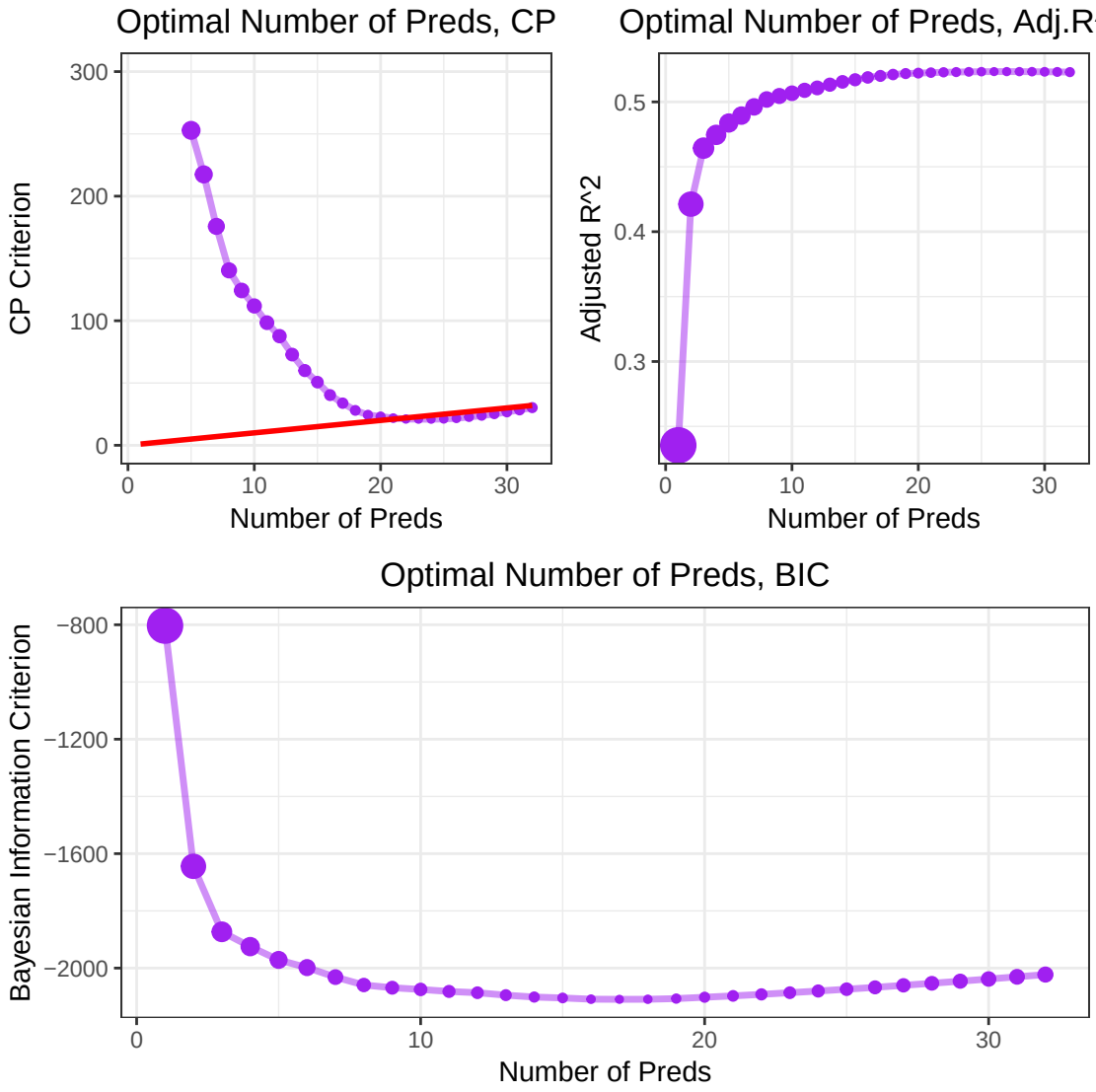
```
#Impute those missing less than 20%
#1. pct_employed16_over
#2. pct_private_coverage_alone

#Set up appropriate test and train for pct_employed16_over (removing other missing % variable and respo
train.df <- cancer.df %>% dplyr::select(-c(pct_private_coverage_alone, target_death_rate)) %>% filter(!
test.df <- cancer.df %>% dplyr::select(-c(pct_private_coverage_alone, target_death_rate)) %>% filter(is

#Function for imputation (after correct test, train set up), charstring must literally be the character
impute.lasso <- function(train.df, test.df, charstring){

  if ((charstring %in% names(train.df))) {

#pull variable index
index <- which(names(train.df) == charstring)

#Set up Matrices
#Create Design Matrix Train
X <- train.df[ ,-index] %>%
  names() %>%
  paste("~ ", paste(., collapse = "+")) %>%
  formula() %>%
  model.matrix(.,train.df)

#Create Design Matrix Test
X1 <- test.df[, -index] %>%
  names() %>%
  paste("~ ", paste(., collapse = "+")) %>%
  formula() %>%
  model.matrix(., test.df)

#Remove Intercept
X <- X[,-1]
X1 <- X1[,-1]

#Create Response vector (as matrix)
Y <- train.df[, index] %>% as.matrix()

#Optimize lambda
lambda.grid <- 10^seq(-3,1,length = 100)

#CV n = 10
cv.lasso <- cv.glmnet(X, Y, alpha = 1, intercept = TRUE, lambda = lambda.grid, family = "gaussian")

#Grab optimal lambda
opt.lambda.lasso <- cv.lasso$lambda.min

#Run model
unemploy.lasso <- glmnet(X, Y, alpha = 1, intercept = TRUE, lambda = opt.lambda.lasso, family = "gaussi
```

```
#Return predictions
predict(unemploy.lasso, newx = X1)
  }else{
    stop("Error: Incorrect variable name")
  }
}

#Impute employed16_over_preds (first since it has less missing data ~4%)
employed16_over_preds <- impute.lasso(train.df = train.df, test.df, "pct_employed16_over")

#Set up appropriate test and train
train.df <- cancer.df %>% dplyr::select(-c(pct_employed16_over, target_death_rate)) %>% filter(!is.na(pc
test.df <- cancer.df %>% dplyr::select(-c(pct_employed16_over, target_death_rate)) %>% filter(is.na(pct_

#Impute pct_private_coverage_alone (second since it has more missing data ~20%)
pct_private_coverage_alone_preds <- impute.lasso(train.df = train.df, test.df, "pct_private_coverage_alo

#Replace Imputed values
cancer.df <- cancer.df %>%
  mutate(imp_pct_employed16_over = ifelse(is.na(pct_employed16_over),
                                          employed16_over_preds, pct_employed16_over),
         imp_pct_private_coverage_alone = ifelse(is.na(pct_private_coverage_alone),
                                          pct_private_coverage_alone_preds, pct_private_coverage_alone)
        )

#Looks good, so we will replace imputed variables in our final data set
cancer.df <- cancer.df %>%
  dplyr::select(-c(pct_employed16_over, pct_private_coverage_alone))
```

See generally what the optimal number of preds in an lm() ought to be (CP, adj$R^2$, BIC (similar to AIC)).

```
#Summary of models for each size (one model per size)
#Set max number of predictors
nvmax <- ncol(cancer.df) - 2
reg.subsets <- cancer.df %>%
  dplyr::select(-c(state)) %>%
  regsubsets(target_death_rate ~ ., data = ., really.big = FALSE, nvmax = nvmax)
rs <- summary(reg.subsets)

# Plots of Cp and Adj-R2 as functions of parameters
r.df <- tibble(
  preds = 1:nvmax,
  cp = rs$cp,
  adjr2 = rs$adjr2,
  bic = rs$bic,
  step = 1:nvmax
)

cp.plot <- r.df %>% ggplot(aes(x = preds, y = cp, size = cp)) +
  geom_point(colour = "purple") +
  geom_line(alpha = 0.5, colour = "purple", size = 1.25) +
 # geom_point(aes(x = preds, step),color = "black",size = 0.5) +
  geom_line(aes(x = preds, step), size = 1, color = "red") +
  labs(
```

```
    x = "Number of Preds",
    y = "CP Criterion",
    title = "Optimal Number of Preds, CP"
  ) +
  #scale_y_continuous(breaks = seq(0, 1000, 100))
  ylim(c(0, 300)) +
  theme(legend.position = "none")

adjr2.plot <- r.df %>% ggplot(aes(x = preds, y = adjr2, size = 1 - adjr2)) +
  geom_point(colour = "purple") +
  geom_line(alpha = 0.5, colour = "purple", size = 1.25) +
  labs(
    x = "Number of Preds",
    y = "Adjusted R^2",
    title = "Optimal Number of Preds, Adj.R^2"
  ) + theme(legend.position = "none")

bic.plot <- r.df %>% ggplot(aes(x = preds, y = bic, size = bic)) +
  geom_point(colour = "purple") +
  geom_line(alpha = 0.5, colour = "purple", size = 1.25) +
  labs(
    x = "Number of Preds",
    y = "Bayesian Information Criterion",
    title = "Optimal Number of Preds, BIC"
  ) + theme(legend.position = "none")

(cp.plot + adjr2.plot) / bic.plot
```




Based on the plots above, $CP$ criterion in the upper left with $p \leq CP$ constraint in red, that somewhere between a 20-30 predictor model ought to be optimal. With respect to adjusted $R^2$, it appears as though we reach a converging maximum starting around 20 predictors through about 25, where additional predictors have diminishing marginal return. Lastly, BIC is more conservative (penalizing more strongly for more predictors) and seems to suggest between a 15-20 predictor model (closer to 20). This may inform our subset selection criterion.

## K-fold cross validation funciton for subset selecion (MSE, AIC, R^2 criterion)

```
sampleSize <- nrow(cancer.df)

mseCV <- function(data.df, kfolds = 10){
  folds <- sample(1:kfolds, sampleSize, rep = T)
  mse <- rep(0, kfolds)
```

```
  for (k in 1:kfolds) {
    train.df <- data.df[folds != k,]
    test.df <- data.df[folds == k,]

    lm.mod <- lm(target_death_rate ~ ., data = train.df)
    preds <- predict(lm.mod, newdata = test.df)
    mse[k] <- with(test.df,mean((target_death_rate - preds)^2))
  }
  mean(mse)
}

aicCV <- function(data.df, kfolds = 10){
  folds <- sample(1:kfolds, sampleSize, rep = T)
  aic <- rep(0, kfolds)
  for (k in 1:kfolds) {
    train.df <- data.df[folds != k,]
    test.df <- data.df[folds == k,]

    lm.mod <- lm(target_death_rate ~ ., data = train.df)
    aic[k] <- AIC(lm.mod)
  }
  mean(aic)
}

r2CV <- function(data.df, kfolds = 10){
  folds <- sample(1:kfolds, sampleSize, rep = T)
  r2 <- rep(0, kfolds)
  for (k in 1:kfolds) {
    train.df <- data.df[folds != k,]
    test.df <- data.df[folds == k,]

    lm.mod <- lm(target_death_rate ~ ., data = train.df)
    r2[k] <- summary(lm.mod)$adj.r.squared
  }
  mean(r2)
}
```

## Forward Subset Selection with k-Fold CV and MSE criterion

## Set up data for subset selection algorithm

```
#Reorder the data so the response comes first, necessary for indexing
sub.cancer.df <- cancer.df %>%
  dplyr::select(target_death_rate, everything()) %>%
  dplyr::select(-state) # need to take out state for CV to run, too many levels (51) will be fine for f
```

Function to run the subset algorithm, it will output the variable selection process so you visualize how it works. Comment out the set seed to get true variability in the subsets, only leave it in for reproducibility.

- The function `f.subset.mse(sub.cancer.df, maxPreds = 10, nfolds = 5)` takes in a data frame, max number of predictors you want the algorithm to run through (can't be larger than the number of

predictors in the data frame), and the number of folds for the cross validation.

**Note mustreorder data frame so response is the first column of the data frame (see above code chunk)**

**Also have to take out state variable, too many factor levels and CV process can fail (re add later to see if it's useful)**

- The way the algorithm works is

1. Starts with an empty (null) set of current predictors in the data frame input
2. Starts with a full set of available predictors
3. Loops through all available preds and adds one at a time, splits the data into kfolds test/train (CV), fits lm() with current preds, stores MSE
4. Selects the predictor whose CV MSE was smallest
5. Adds 'best' predictor to current predictors, removes it from available predictors
6. Stores that predictor set as 'best' and records the indices and CV MSE in a matrix
7. Repeats 3.- 5. until you have met the maximum number of predictors you specified
8. Returns a list with all 'best' predictor sets at each iteration, matrix of results, and prespecified max preds
9. Last few lines of code output the optimal (minimum MSE) predictor set

*notes - CV is set to 5 for speed, changing to 10 will take ~1.5 times as long to run notes - the print() lines within the function are so you can visualize how the algorithm is working, they can be commented out to reduce clutter when you want to run the selection function iteratively (avoids reams of output)*

```
#Forward Subset#
set.seed(4) #Comment this out if you want to really get a different subset

f.subset <- function(sub.cancer.df, maxPreds = 10, nfolds = 5, criterion = "mse") {
#Selection
#number of possible predictors
## Allthe predictors (their indices).
allPreds <- 1:ncol(sub.cancer.df)
allPreds <- allPreds[-1]

#current set of preds (starts empty), and available
currPreds <- c()
availPreds <- setdiff(allPreds,currPreds)
#Record the min errors
minError <- c()
#The maximimum size of our predictor set
#maxPreds <- 30
#Initalize pred list and mse result matrix
pred.list <- list()
result.mat <- matrix(nrow = ncol(cancer.df) - 1, ncol = 2)
i <- 1
#Forward selection loop
while (length(currPreds) < maxPreds) {
  ##add predictor which decreases MSE (as determined by CV or
  ##Bootstrapping)
```

```
  ## The MSEs computed as we add each of the available predictors
  allError <- c()
  for (id in availPreds) {
    data.df <- sub.cancer.df[,c(currPreds,id,1)]
    if (criterion == "mse") {
      error <- mseCV(data.df, nfolds)
    } else if (criterion == "aic") {
      error <- aicCV(data.df, nfolds)
    } else {
      stop("Wrong criterion input")
    }
  #  error <- mseCV(data.df, nfolds)
  #  error <- aicCV(data.df, nfolds)
   # error <- r2CV(data.df, nfolds)
    allError <- c(allError,error)
  }
  ##Find the min
  id <- which.min(allError)
  ##get the best predictor and MSW
  bestPred <- availPreds[id]
  bestError <- min(allError)
  ##Add these into the collection
  currPreds <- c(currPreds,bestPred)
  minError <- c(minError,bestError)
  availPreds <- setdiff(allPreds,currPreds)
  ## Print stuff out for debugging and attention-grabbing
  print(sprintf("Iteration: %i Predictor Added: %s %s Value: %s",i, names(sub.cancer.df[,bestPred]), cr
  print(currPreds)
  result.mat[i,] <- c(bestPred,bestError)         #You can also comment out this print output, it's jus
  pred.list[[i]] <- currPreds
  i <- i + 1
    }
  return(list(pred.list = pred.list, result.mat = result.mat, maxPreds = maxPreds))
}

#Run Subset, call function, output is a list with predictor sets, reslut matrix and maxPreds
f.mse.list <- f.subset(sub.cancer.df, maxPreds = ncol(sub.cancer.df) - 1, nfolds = 5, criterion = "mse"
```

```
## [1] "Iteration: 1 Predictor Added: pct_bach_deg25_over mse Value: 589.013984487223"
## [1] 17
## [1] "Iteration: 2 Predictor Added: incidence_rate mse Value: 446.820254189342"
## [1] 17  4
## [1] "Iteration: 3 Predictor Added: poverty_percent mse Value: 414.153773458571"
## [1] 17  4  7
## [1] "Iteration: 4 Predictor Added: pct_other_race mse Value: 405.48773268966"
## [1] 17  4  7 26
## [1] "Iteration: 5 Predictor Added: pct_hs18_24 mse Value: 398.510139381854"
## [1] 17  4  7 26 14
## [1] "Iteration: 6 Predictor Added: pct_married_households mse Value: 396.436091534216"
## [1] 17  4  7 26 14 27
## [1] "Iteration: 7 Predictor Added: pct_unemployed16_over mse Value: 392.765588752657"
## [1] 17  4  7 26 14 27 18
## [1] "Iteration: 8 Predictor Added: median_age_male mse Value: 392.18240237804"
## [1] 17  4  7 26 14 27 18  9
```

```
## [1] "Iteration: 9 Predictor Added: binned_inc_lb mse Value: 388.67206708603"
## [1] 17  4  7 26 14 27 18  9 29
## [1] "Iteration: 10 Predictor Added: birth_rate mse Value: 386.217531639082"
##  [1] 17  4  7 26 14 27 18  9 29 28
## [1] "Iteration: 11 Predictor Added: binned_inc_ub mse Value: 384.574566169899"
##  [1] 17  4  7 26 14 27 18  9 29 28 30
## [1] "Iteration: 12 Predictor Added: pct_white mse Value: 382.972460759841"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23
## [1] "Iteration: 13 Predictor Added: pct_bach_deg18_24 mse Value: 384.120457279973"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15
## [1] "Iteration: 14 Predictor Added: pct_public_coverage_alone mse Value: 381.419018037792"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22
## [1] "Iteration: 15 Predictor Added: percent_married mse Value: 379.907771913967"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12
## [1] "Iteration: 16 Predictor Added: imp_pct_employed16_over mse Value: 379.780138831602"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32
## [1] "Iteration: 17 Predictor Added: pct_emp_priv_coverage mse Value: 380.959933367791"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20
## [1] "Iteration: 18 Predictor Added: pct_public_coverage mse Value: 379.248792659147"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21
## [1] "Iteration: 19 Predictor Added: med_income mse Value: 376.957979256377"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5
## [1] "Iteration: 20 Predictor Added: pct_private_coverage mse Value: 377.378392461666"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19
## [1] "Iteration: 21 Predictor Added: pop_est2015 mse Value: 377.134787329998"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6
## [1] "Iteration: 22 Predictor Added: pct_asian mse Value: 377.974102243425"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25
## [1] "Iteration: 23 Predictor Added: avg_deaths_per_year mse Value: 377.910061876026"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25  3
## [1] "Iteration: 24 Predictor Added: avg_ann_count mse Value: 377.575253544331"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25  3
## [24]  2
## [1] "Iteration: 25 Predictor Added: pct_hs25_over mse Value: 374.422849540578"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25  3
## [24]  2 16
## [1] "Iteration: 26 Predictor Added: pct_no_hs18_24 mse Value: 375.219882992857"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25  3
## [24]  2 16 13
## [1] "Iteration: 27 Predictor Added: pct_black mse Value: 373.971658867921"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25  3
## [24]  2 16 13 24
## [1] "Iteration: 28 Predictor Added: imp_pct_private_coverage_alone mse Value: 375.271194762825"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25  3
## [24]  2 16 13 24 33
## [1] "Iteration: 29 Predictor Added: study_quantile mse Value: 375.022357173491"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25  3
## [24]  2 16 13 24 33 31
## [1] "Iteration: 30 Predictor Added: avg_household_size mse Value: 374.015208359661"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25  3
## [24]  2 16 13 24 33 31 11
## [1] "Iteration: 31 Predictor Added: median_age mse Value: 376.702219488803"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25  3
## [24]  2 16 13 24 33 31 11  8
```

```
## [1] "Iteration: 32 Predictor Added: median_age_female mse Value: 379.975411443246"
##  [1] 17  4  7 26 14 27 18  9 29 28 30 23 15 22 12 32 20 21  5 19  6 25  3
## [24]  2 16 13 24 33 31 11  8 10
```

```
f.aic.list <- f.subset(sub.cancer.df, maxPreds = ncol(sub.cancer.df) - 1, nfolds = 5, criterion = "aic")
```

```
## [1] "Iteration: 1 Predictor Added: pct_bach_deg25_over aic Value: 22468.4035230514"
## [1] 17
## [1] "Iteration: 2 Predictor Added: incidence_rate aic Value: 21791.1409767042"
## [1] 17  4
## [1] "Iteration: 3 Predictor Added: poverty_percent aic Value: 21602.4757740336"
## [1] 17  4  7
## [1] "Iteration: 4 Predictor Added: pct_hs18_24 aic Value: 21556.9352993351"
## [1] 17  4  7 14
## [1] "Iteration: 5 Predictor Added: pct_other_race aic Value: 21515.2885573371"
## [1] 17  4  7 14 26
## [1] "Iteration: 6 Predictor Added: pct_married_households aic Value: 21490.1511101971"
## [1] 17  4  7 14 26 27
## [1] "Iteration: 7 Predictor Added: median_age_female aic Value: 21476.5820287675"
## [1] 17  4  7 14 26 27 10
## [1] "Iteration: 8 Predictor Added: birth_rate aic Value: 21457.9013922956"
## [1] 17  4  7 14 26 27 10 28
## [1] "Iteration: 9 Predictor Added: pct_private_coverage aic Value: 21443.3163650499"
## [1] 17  4  7 14 26 27 10 28 19
## [1] "Iteration: 10 Predictor Added: pct_emp_priv_coverage aic Value: 21431.171398017"
##  [1] 17  4  7 14 26 27 10 28 19 20
## [1] "Iteration: 11 Predictor Added: binned_inc_lb aic Value: 21415.2402236759"
##  [1] 17  4  7 14 26 27 10 28 19 20 29
## [1] "Iteration: 12 Predictor Added: binned_inc_ub aic Value: 21402.0437844298"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30
## [1] "Iteration: 13 Predictor Added: percent_married aic Value: 21395.6989750298"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12
## [1] "Iteration: 14 Predictor Added: imp_pct_employed16_over aic Value: 21382.7626002531"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32
## [1] "Iteration: 15 Predictor Added: pct_hs25_over aic Value: 21373.8203736031"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16
## [1] "Iteration: 16 Predictor Added: median_age_male aic Value: 21369.3907988935"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9
## [1] "Iteration: 17 Predictor Added: pct_white aic Value: 21366.1083507099"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23
## [1] "Iteration: 18 Predictor Added: pct_no_hs18_24 aic Value: 21363.5188920356"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13
## [1] "Iteration: 19 Predictor Added: avg_ann_count aic Value: 21361.3777473399"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2
## [1] "Iteration: 20 Predictor Added: avg_deaths_per_year aic Value: 21348.6361329823"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3
## [1] "Iteration: 21 Predictor Added: pop_est2015 aic Value: 21342.5236434066"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6
## [1] "Iteration: 22 Predictor Added: imp_pct_private_coverage_alone aic Value: 21341.9193959471"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33
## [1] "Iteration: 23 Predictor Added: pct_black aic Value: 21338.7623597895"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33 24
## [1] "Iteration: 24 Predictor Added: pct_unemployed16_over aic Value: 21339.2873750964"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33 24
## [24] 18
```

```
## [1] "Iteration: 25 Predictor Added: pct_public_coverage_alone aic Value: 21339.5465391676"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33 24
## [24] 18 22
## [1] "Iteration: 26 Predictor Added: med_income aic Value: 21341.002883976"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33 24
## [24] 18 22  5
## [1] "Iteration: 27 Predictor Added: pct_asian aic Value: 21342.2789454706"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33 24
## [24] 18 22  5 25
## [1] "Iteration: 28 Predictor Added: pct_bach_deg18_24 aic Value: 21341.5964731798"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33 24
## [24] 18 22  5 25 15
## [1] "Iteration: 29 Predictor Added: median_age aic Value: 21344.3175693357"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33 24
## [24] 18 22  5 25 15  8
## [1] "Iteration: 30 Predictor Added: pct_public_coverage aic Value: 21346.3671741647"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33 24
## [24] 18 22  5 25 15  8 21
## [1] "Iteration: 31 Predictor Added: avg_household_size aic Value: 21345.8267487972"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33 24
## [24] 18 22  5 25 15  8 21 11
## [1] "Iteration: 32 Predictor Added: study_quantile aic Value: 21353.0074948132"
##  [1] 17  4  7 14 26 27 10 28 19 20 29 30 12 32 16  9 23 13  2  3  6 33 24
## [24] 18 22  5 25 15  8 21 11 31
```

```
#Show the 'best' final selection with minimal MSE
present.fs.result <- function(f.result.list, criterion) {
lm.fs.preds <- with(f.result.list, pred.list[[which.min(result.mat[,2])]]) #Pick out indices from best
fs.mse <- with(f.result.list, result.mat[which.min(result.mat[,2]), 2])
print(sprintf("The best predictor set of %s predictors, out of a max of %s, (%s = %s)",
              length(lm.fs.preds), f.result.list$maxPreds, criterion, round(fs.mse, 3)))
print(names(sub.cancer.df[,c(lm.fs.preds)]))
}

present.fs.result(f.mse.list, "MSE")
```

```
## [1] "The best predictor set of 27 predictors, out of a max of 32, (MSE = 373.972)"
##  [1] "pct_bach_deg25_over"       "incidence_rate"
##  [3] "poverty_percent"           "pct_other_race"
##  [5] "pct_hs18_24"               "pct_married_households"
##  [7] "pct_unemployed16_over"     "median_age_male"
##  [9] "binned_inc_lb"             "birth_rate"
## [11] "binned_inc_ub"             "pct_white"
## [13] "pct_bach_deg18_24"         "pct_public_coverage_alone"
## [15] "percent_married"           "imp_pct_employed16_over"
## [17] "pct_emp_priv_coverage"     "pct_public_coverage"
## [19] "med_income"                "pct_private_coverage"
## [21] "pop_est2015"               "pct_asian"
## [23] "avg_deaths_per_year"       "avg_ann_count"
## [25] "pct_hs25_over"             "pct_no_hs18_24"
## [27] "pct_black"
```

```
present.fs.result(f.aic.list, "AIC")
```

```
## [1] "The best predictor set of 23 predictors, out of a max of 32, (AIC = 21338.762)"
```

```
##  [1] "pct_bach_deg25_over"           "incidence_rate"
##  [3] "poverty_percent"               "pct_hs18_24"
##  [5] "pct_other_race"                "pct_married_households"
##  [7] "median_age_female"             "birth_rate"
##  [9] "pct_private_coverage"          "pct_emp_priv_coverage"
## [11] "binned_inc_lb"                 "binned_inc_ub"
## [13] "percent_married"               "imp_pct_employed16_over"
## [15] "pct_hs25_over"                 "median_age_male"
## [17] "pct_white"                     "pct_no_hs18_24"
## [19] "avg_ann_count"                 "avg_deaths_per_year"
## [21] "pop_est2015"                   "imp_pct_private_coverage_alone"
## [23] "pct_black"
```

```
#fs.lm <- lm(target_death_rate ~ ., data = cancer.df[,c(lm.fs.preds,1)])
```

If you want to repeat the process to get a feel for what you think might actually be the best of the 'best' subset selections (they very slightly from iteration to iteration by cross validation). Pick the number of iterations `len` and let the chunk run, you can see the process work (reccomend commenting out the print() calls in the functions above to reduce all the extra output), then at the end it will print all the subsets selected and you can sift through which variables you want for a final subset model. See which ones get selected most often basically.

```
#Repeat if you want to
#Number of repetitions
len <- 10
f.result.list <- list()

#Repeat Subset Selection algorithm
for (i in 1:len) {
  f.result.list[[i]] <- f.subset(sub.cancer.df, maxPreds = 10, nfolds = 5, criterion = "mse")
}

#View the results
for (i in 1:len) {
present.fs.result(f.result.list[[i]], "MSE")
}

#See which vars seem to be selected most often, make a subjective call what subset you like best.

#fs.lm <- lm(target_death_rate ~ ., data = cancer.df[,c(lm.fs.preds,1)])
```

# Repeat the process for Backwards Subset

## NOTE this will take much longer to run

**Again must reorder data frame so response is the first column of the data frame (see above code chunk)**

**Again state variable removed, too many factor levels and CV process can fail (re add later to see if it's useful)**

- The way the algorithm works is

1. Starts with an exhaustive set of current predictors in the data frame input (all preds in the data frame)
2. Starts with a empty set of predictors removed

3. Loops through all current preds and removes one at a time, splits the data into kfolds test/train (CV), fits lm() with current preds, stores MSE
4. Selects the set whose CV MSE was smallest
5. The optimal set had one of the predictors removed, selects that predictor as 'worst; removes it from current predictors

6. Stores that predictor set as worst and records the indices and CV MSE in a matrix for the model without that pred
7. Repeats 3.- 5. until you have met the minimum number of predictors you specified
8. Returns a list with all 'best' predictor sets at each iteration, matrix of results, and prespecified max preds
9. Last few lines of code output the optimal (minimum MSE) predictor set

*notes - CV is set to 5 for speed, changing to 10 will take ~1.5 times as long to run notes - the print() lines within the function are so you can visualize how the algorithm is working, they can be commented out to reduce clutter when you want to run the selection function iteratively (avoids reams of output)*

```
sub.cancer.df <- cancer.df %>%
  dplyr::select(target_death_rate, everything()) %>%
  dplyr::select(-state)
  #dplyr::select(-c(state, avg_deaths_yr_pop, avg_ann_count_pop, pct_non_white, binned_inc_point))
#For backwards to work well we need remove vars we created that are combos of other vars,
#This will work better when we have a smaller non-interdependent data set with reduced variables
```

```
#Backward Selection
#set.seed(44)  #Set seed for reproducibility
################################################

maxPreds <- ncol(sub.cancer.df - 1)

b.subset <- function(sub.cancer.df, minPreds = 1, nfolds = 5, criterion = "mse"){

## Allthe predictors (their indices).
allPreds <- 1:ncol(sub.cancer.df)
allPreds <- allPreds[-1]

#current set of preds (starts empty), and available
currPreds <- allPreds
availPreds <- allPreds
#Record the min errors
minError <- c()
#The minimum size of our predictor set minPreds

pred.list <- list()
result.mat <- matrix(nrow = ncol(sub.cancer.df) - 1, ncol = 2)
i <- 1
#Forward selection loop
while (length(currPreds) >= minPreds) {
  ##add predictor which decreases MSE (as determined by CV)
  ## The MSEs computed as we add each of the available predictors
  allError <- c()
  for (id in availPreds) {
```

```
    data.df <- sub.cancer.df[,c(currPreds[-id],1)]
    if (criterion == "mse") {
      error <- mseCV(data.df, nfolds)
    } else if (criterion == "aic") {
      error <- aicCV(data.df, nfolds)
    } else {
      stop("Wrong criterion input")
    }
    allError <- c(allError,error)
  }
  ##Find the min
  id <- which.min(allError)
  ##get the worst predictor and MSE
  worstPred <- availPreds[id]
  bestError <- min(allError)
  ##Add these into the collection
  currPreds <- currPreds[-id]
  minError <- c(minError, bestError)
  availPreds <- currPreds
  ## Print stuff out for debugging and attention-grabbing
  print(sprintf("Predictor Removed: %s  %s Value: %s",names(sub.cancer.df[,worstPred]), criterion, bestE
  print(currPreds)
  result.mat[i,] <- c(worstPred,bestError)
  pred.list[[i]] <- currPreds

  i <- i + 1
  }
  list(pred.list = pred.list, result.mat = result.mat)
}

mse.result.list <- b.subset(sub.cancer.df, minPreds = 10, nfolds = 5, criterion = "mse")
```

```
## [1] "Predictor Removed: imp_pct_employed16_over  mse Value: 372.949521184514"
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
## [24] 25 26 27 28 29 30 31 33
## [1] "Predictor Removed: percent_married  mse Value: 374.856857015884"
##  [1]  2  3  4  5  6  7  8  9 10 11 13 14 15 16 17 18 19 20 21 22 23 24 25
## [24] 26 27 28 29 30 31 33
## [1] "Predictor Removed: poverty_percent  mse Value: 378.967262137518"
##  [1]  2  3  4  5  6  8  9 10 11 13 14 15 16 17 18 19 20 21 22 23 24 25 26
## [24] 27 28 29 30 31 33
## [1] "Predictor Removed: pct_bach_deg18_24  mse Value: 378.33778033959"
##  [1]  2  3  4  5  6  8  9 10 11 13 14 16 17 18 19 20 21 22 23 24 25 26 27
## [24] 28 29 30 31 33
## [1] "Predictor Removed: study_quantile  mse Value: 375.837280497219"
##  [1]  2  3  4  5  6  8  9 10 11 13 14 16 17 18 19 20 21 22 23 24 25 26 27
## [24] 28 29 30 33
## [1] "Predictor Removed: birth_rate  mse Value: 377.02651709691"
##  [1]  2  3  4  5  6  8  9 10 11 13 14 16 17 18 19 20 21 22 23 24 25 26 27
## [24] 29 30 33
## [1] "Predictor Removed: median_age_male  mse Value: 378.446934424529"
##  [1]  2  3  4  5  6  8 10 11 13 14 16 17 18 19 20 21 22 23 24 25 26 27 29
## [24] 30 33
## [1] "Predictor Removed: pct_bach_deg25_over  mse Value: 378.600552432524"
```

```
##  [1]  2  3  4  5  6  8 10 11 13 14 16 18 19 20 21 22 23 24 25 26 27 29 30
## [24] 33
## [1] "Predictor Removed: pct_asian  mse Value: 387.848921922906"
##  [1]  2  3  4  5  6  8 10 11 13 14 16 18 19 20 21 22 23 24 26 27 29 30 33
## [1] "Predictor Removed: pop_est2015  mse Value: 387.017592366578"
##  [1]  2  3  4  5  8 10 11 13 14 16 18 19 20 21 22 23 24 26 27 29 30 33
## [1] "Predictor Removed: pct_hs18_24  mse Value: 388.282430652285"
##  [1]  2  3  4  5  8 10 11 13 16 18 19 20 21 22 23 24 26 27 29 30 33
## [1] "Predictor Removed: med_income  mse Value: 392.999661362095"
##  [1]  2  3  4  8 10 11 13 16 18 19 20 21 22 23 24 26 27 29 30 33
## [1] "Predictor Removed: binned_inc_ub  mse Value: 389.844606022852"
##  [1]  2  3  4  8 10 11 13 16 18 19 20 21 22 23 24 26 27 29 33
## [1] "Predictor Removed: imp_pct_private_coverage_alone  mse Value: 392.578002106159"
##  [1]  2  3  4  8 10 11 13 16 18 19 20 21 22 23 24 26 27 29
## [1] "Predictor Removed: pct_private_coverage  mse Value: 393.471762554426"
##  [1]  2  3  4  8 10 11 13 16 18 20 21 22 23 24 26 27 29
## [1] "Predictor Removed: incidence_rate  mse Value: 396.503583798636"
##  [1]  2  3  8 10 11 13 16 18 20 21 22 23 24 26 27 29
## [1] "Predictor Removed: avg_deaths_per_year  mse Value: 495.138491126215"
##  [1]  2  8 10 11 13 16 18 20 21 22 23 24 26 27 29
## [1] "Predictor Removed: pct_white  mse Value: 497.867034899924"
##  [1]  2  8 10 11 13 16 18 20 21 22 24 26 27 29
## [1] "Predictor Removed: pct_public_coverage_alone  mse Value: 496.247253635513"
##  [1]  2  8 10 11 13 16 18 20 21 24 26 27 29
## [1] "Predictor Removed: pct_hs25_over  mse Value: 500.355096098049"
##  [1]  2  8 10 11 13 18 20 21 24 26 27 29
## [1] "Predictor Removed: avg_household_size  mse Value: 545.235442557452"
##  [1]  2  8 10 13 18 20 21 24 26 27 29
## [1] "Predictor Removed: avg_ann_count  mse Value: 543.384497622327"
##  [1]  8 10 13 18 20 21 24 26 27 29
## [1] "Predictor Removed: pct_married_households  mse Value: 545.609964096024"
## [1]  8 10 13 18 20 21 24 26 29
```

```
aic.result.list <- b.subset(sub.cancer.df, minPreds = 10, nfolds = 5, criterion = "aic")
```

```
## [1] "Predictor Removed: binned_inc_ub  aic Value: 21345.3315455278"
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
## [24] 25 26 27 28 29 31 32 33
## [1] "Predictor Removed: binned_inc_lb  aic Value: 21350.7556566582"
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
## [24] 25 26 27 28 31 32 33
## [1] "Predictor Removed: pct_emp_priv_coverage  aic Value: 21371.6072998913"
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 21 22 23 24 25
## [24] 26 27 28 31 32 33
## [1] "Predictor Removed: pct_married_households  aic Value: 21377.9226472136"
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 21 22 23 24 25
## [24] 26 28 31 32 33
## [1] "Predictor Removed: pct_private_coverage  aic Value: 21421.1597209573"
##  [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 21 22 23 24 25 26
## [24] 28 31 32 33
## [1] "Predictor Removed: median_age_male  aic Value: 21428.1505735716"
##  [1]  2  3  4  5  6  7  8 10 11 12 13 14 15 16 17 18 21 22 23 24 25 26 28
## [24] 31 32 33
## [1] "Predictor Removed: pct_black  aic Value: 21427.0956505806"
##  [1]  2  3  4  5  6  7  8 10 11 12 13 14 15 16 17 18 21 22 23 25 26 28 31
```

```
## [24] 32 33
## [1] "Predictor Removed: pct_white  aic Value: 21426.8744078779"
##  [1]  2  3  4  5  6  7  8 10 11 12 13 14 15 16 17 18 21 22 25 26 28 31 32
## [24] 33
## [1] "Predictor Removed: imp_pct_employed16_over  aic Value: 21434.9240380372"
##  [1]  2  3  4  5  6  7  8 10 11 12 13 14 15 16 17 18 21 22 25 26 28 31 33
## [1] "Predictor Removed: pct_public_coverage_alone  aic Value: 21432.2584019312"
##  [1]  2  3  4  5  6  7  8 10 11 12 13 14 15 16 17 18 21 25 26 28 31 33
## [1] "Predictor Removed: pct_public_coverage  aic Value: 21445.8818907682"
##  [1]  2  3  4  5  6  7  8 10 11 12 13 14 15 16 17 18 25 26 28 31 33
## [1] "Predictor Removed: pct_bach_deg25_over  aic Value: 21450.2002816906"
##  [1]  2  3  4  5  6  7  8 10 11 12 13 14 15 16 18 25 26 28 31 33
## [1] "Predictor Removed: median_age_female  aic Value: 21505.8481546742"
##  [1]  2  3  4  5  6  7  8 11 12 13 14 15 16 18 25 26 28 31 33
## [1] "Predictor Removed: pct_unemployed16_over  aic Value: 21524.5607306291"
##  [1]  2  3  4  5  6  7  8 11 12 13 14 15 16 25 26 28 31 33
## [1] "Predictor Removed: incidence_rate  aic Value: 21539.9788522397"
##  [1]  2  3  5  6  7  8 11 12 13 14 15 16 25 26 28 31 33
## [1] "Predictor Removed: pct_hs25_over  aic Value: 22117.9241076267"
##  [1]  2  3  5  6  7  8 11 12 13 14 15 25 26 28 31 33
## [1] "Predictor Removed: pct_bach_deg18_24  aic Value: 22261.7540935003"
##  [1]  2  3  5  6  7  8 11 12 13 14 25 26 28 31 33
## [1] "Predictor Removed: pct_hs18_24  aic Value: 22272.5325341076"
##  [1]  2  3  5  6  7  8 11 12 13 25 26 28 31 33
## [1] "Predictor Removed: pct_no_hs18_24  aic Value: 22407.161264547"
##  [1]  2  3  5  6  7  8 11 12 25 26 28 31 33
## [1] "Predictor Removed: percent_married  aic Value: 22405.1392289245"
##  [1]  2  3  5  6  7  8 11 25 26 28 31 33
## [1] "Predictor Removed: avg_household_size  aic Value: 22405.130627951"
##  [1]  2  3  5  6  7  8 25 26 28 31 33
## [1] "Predictor Removed: pop_est2015  aic Value: 22407.2371413353"
##  [1]  2  3  5  7  8 25 26 28 31 33
## [1] "Predictor Removed: med_income  aic Value: 22434.8589365168"
## [1]  2  3  7  8 25 26 28 31 33
```

```
present.bs.result <- function(result.list, criterion) {
lm.bs.preds <- with(result.list, pred.list[[which.min(result.mat[,2])]])
lm.bs.mse <- with(result.list, result.mat[which.min(result.mat[,2]), 2])
print(sprintf("The best predictor set of %s predictors, out of a max of %s, (%s = %s)",
              length(lm.bs.preds), maxPreds, criterion, round(lm.bs.mse, 3)))
names(sub.cancer.df[,c(lm.bs.preds)])
}

present.bs.result(mse.result.list, "MSE")
```

```
## [1] "The best predictor set of 31 predictors, out of a max of 33, (MSE = 372.95)"
```

```
##  [1] "avg_ann_count"                "avg_deaths_per_year"
##  [3] "incidence_rate"               "med_income"
##  [5] "pop_est2015"                  "poverty_percent"
##  [7] "median_age"                   "median_age_male"
##  [9] "median_age_female"            "avg_household_size"
## [11] "percent_married"              "pct_no_hs18_24"
## [13] "pct_hs18_24"                  "pct_bach_deg18_24"
## [15] "pct_hs25_over"                "pct_bach_deg25_over"
```

```
## [17] "pct_unemployed16_over"          "pct_private_coverage"
## [19] "pct_emp_priv_coverage"          "pct_public_coverage"
## [21] "pct_public_coverage_alone"      "pct_white"
## [23] "pct_black"                      "pct_asian"
## [25] "pct_other_race"                 "pct_married_households"
## [27] "birth_rate"                     "binned_inc_lb"
## [29] "binned_inc_ub"                  "study_quantile"
## [31] "imp_pct_private_coverage_alone"
```

```
present.bs.result(aic.result.list, "AIC")
```

```
## [1] "The best predictor set of 31 predictors, out of a max of 33, (AIC = 21345.332)"
```

```
##  [1] "avg_ann_count"                   "avg_deaths_per_year"
##  [3] "incidence_rate"                  "med_income"
##  [5] "pop_est2015"                     "poverty_percent"
##  [7] "median_age"                      "median_age_male"
##  [9] "median_age_female"               "avg_household_size"
## [11] "percent_married"                 "pct_no_hs18_24"
## [13] "pct_hs18_24"                     "pct_bach_deg18_24"
## [15] "pct_hs25_over"                   "pct_bach_deg25_over"
## [17] "pct_unemployed16_over"           "pct_private_coverage"
## [19] "pct_emp_priv_coverage"           "pct_public_coverage"
## [21] "pct_public_coverage_alone"       "pct_white"
## [23] "pct_black"                       "pct_asian"
## [25] "pct_other_race"                  "pct_married_households"
## [27] "birth_rate"                      "binned_inc_lb"
## [29] "study_quantile"                  "imp_pct_employed16_over"
## [31] "imp_pct_private_coverage_alone"
```

```
#bs.lm <- lm(life_exp ~ ., data = sub.cancer.df[,c(lm.bs.preds,1)])
```

In general, backwards selection selects larger models than forawrds. I would suggest only using forward selection in this case to choose a smaller model. Or stepwise based on some other criterion.

Repeat to get a good feel for which preds get selected most often (note this will take a long time if the original p-set you are using is large, wouldnt use the full set for these iterations). This can be useful to get a feel for what you think might actually be the best of the 'best' subset selections (they very slightly from iteration to iteration by cross validation). Pick the number of iterations `len` and let the chunk run, you can see the process work (reccomend commenting out the print() calls in the functions above to reduce all the extra output), then at the end it will print all the subsets selected and you can sift through which variables you want for a final subset model. See which ones get selected most often basically.

*This will take a long time to run*

```
#Repeat if you want to
#Number of repetitions
len <- 10
b.result.list <- list()

#Repeat Subset Selection algorithm
for (i in 1:len) {
  b.result.list[[i]] <- b.subset(sub.cancer.df, minPreds = 30, nfolds = 5, "mse")
}

#View the results
for (i in 1:len) {
```

```
present.bs.result(b.result.list[[i]], "MSE")
}

#See which vars seem to be selected most often, make a subjective call what subset you like best.

#fs.lm <- lm(target_death_rate ~ ., data = cancer.df[,c(lm.fs.preds,1)])
```