Subset Selection

Nate Wells

Math 243: Stat Learning

October 19th, 2020

Outline

In today's class, we will...

- Discuss data from first midterm
- Investigate algorithms for selecting good subsets of predictors

Section 1

Midterm Data

Overview

- Students fit models of varying complexity based on data on 31 predictors for 200 houses.

Overview

- Students fit models of varying complexity based on data on 31 predictors for 200 houses.

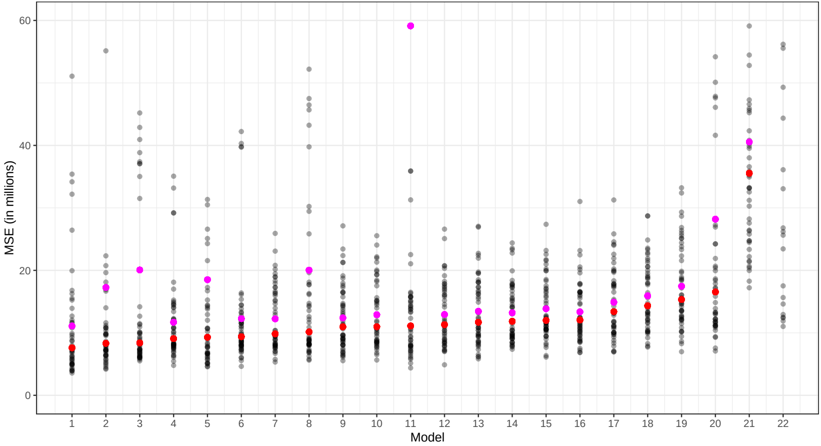
```
## Rows: 200
## Columns: 31
## $ Functional    <chr> "Typ", "Typ", "Typ", "Typ", "Typ", "Typ", "Typ", "Typ...
## $ BldgType      <chr> "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "1Fam", "1Fam...
## $ Foundation    <chr> "CBlock", "PConc", "CBlock", "CBlock", "CBlock", "CBl...
## $ LotShape      <chr> "Reg", "IR1", "IR1", "IR1", "IR1", "Reg", "Reg", "IR1...
## $ LandSlope     <chr> "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl", "Gtl...
## $ SaleCondition <chr> "Normal", "Normal", "Normal", "Normal", "Normal", "No...
## $ RoofMatl      <chr> "CompShg", "CompShg", "CompShg", "CompShg", "CompShg"...
## $ ScreenPorch   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ MSSubClass    <dbl> 20, 60, 60, 20, 20, 20, 20, 85, 60, 20, 60, 30, 20, 9...
## $ GarageCars    <dbl> 2, 2, 2, 2, 1, 2, 3, 2, 2, 3, 2, 1, 2, 0, 0, 2, 0, 2,...
## $ Id            <dbl> 2, 3, 8, 17, 25, 27, 28, 43, 51, 54, 58, 69, 72, 79, ...
## $ BedroomAbvGr  <dbl> 3, 3, 3, 2, 3, 3, 3, 2, 3, 0, 3, 2, 2, 4, 3, 2, 3, 2,...
## $ TotalBsmtSF   <dbl> 1262, 920, 1107, 1004, 1060, 900, 1704, 840, 794, 184...
## $ LotArea       <dbl> 9600, 11250, 10382, 11241, 8246, 7200, 11478, 9180, 1...
## $ OpenPorchSF   <dbl> 0, 42, 204, 0, 90, 32, 50, 0, 75, 72, 70, 0, 0, 0, 0,...
## $ BsmtFullBath  <dbl> 0, 1, 1, 1, 1, 0, 1, 1, 0, 2, 0, 0, 1, 0, 1, 0, 1, 0,...
## $ WoodDeckSF    <dbl> 298, 0, 235, 0, 406, 222, 0, 240, 0, 857, 0, 0, 0, 0,...
## $ OverallCond   <dbl> 8, 5, 6, 7, 8, 7, 5, 7, 6, 5, 5, 6, 6, 5, 5, 3, 5, 5,...
## $ YrSold        <dbl> 2007, 2008, 2009, 2010, 2010, 2010, 2010, 2007, 2007,...
## $ GrLivArea     <dbl> 1262, 1786, 2090, 1004, 1060, 900, 1704, 884, 1470, 1...
## $ MoSold        <dbl> 5, 9, 11, 3, 5, 5, 5, 12, 7, 11, 8, 6, 6, 4, 8, 12, 1...
## $ TotRmsAbvGrd  <dbl> 6, 6, 7, 5, 6, 5, 7, 5, 6, 5, 7, 4, 4, 8, 5, 6, 6, 5,...
## $ PoolArea      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ YearBuilt     <dbl> 1976, 2001, 1973, 1970, 1968, 1951, 2007, 1983, 1997,...
## $ GarageArea    <dbl> 460, 608, 484, 480, 270, 576, 772, 504, 388, 894, 565...
## $ OverallQual   <dbl> 6, 7, 7, 6, 5, 5, 8, 5, 6, 9, 7, 4, 4, 4, 4, 5, 4, 5,...
## $ Fireplaces    <dbl> 1, 1, 2, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ SalePrice     <dbl> 181500, 223500, 200000, 149000, 154000, 134800, 30600...
## $ EnclosedPorch <dbl> 0, 0, 228, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
```

Test MSE

- Models were assesses by computing MSE on 50 test sets of size 100.

Test MSE

- Models were assesses by computing MSE on 50 test sets of size 100.



Magenta indicates *mean* MSE and red indicates *median* MSE.

Retrospective

Trends:

Retrospective

Trends:

- Models with more predictors tended to do better than models with fewer predictors

Retrospective

Trends:

- Models with more predictors tended to do better than models with fewer predictors
- Models with 0 or 1 interaction terms tended to do better than those with more interactions

Retrospective

Trends:

- Models with more predictors tended to do better than models with fewer predictors
- Models with 0 or 1 interaction terms tended to do better than those with more interactions
- Transforming some predictor variables were helpful in stabilizing test MSE

Retrospective

Trends:

- Models with more predictors tended to do better than models with fewer predictors
- Models with 0 or 1 interaction terms tended to do better than those with more interactions
- Transforming some predictor variables were helpful in stabilizing test MSE
- Performing log or root transformation on response greatly reduced test MSE (after predictions returned to original scale), at the cost of stability

Retrospective

Trends:

- Models with more predictors tended to do better than models with fewer predictors
- Models with 0 or 1 interaction terms tended to do better than those with more interactions
- Transforming some predictor variables were helpful in stabilizing test MSE
- Performing log or root transformation on response greatly reduced test MSE (after predictions returned to original scale), at the cost of stability
- By coincidence, the full model was in the middle of the pack.

Suggestions:

Retrospective

Trends:

- Models with more predictors tended to do better than models with fewer predictors
- Models with 0 or 1 interaction terms tended to do better than those with more interactions
- Transforming some predictor variables were helpful in stabilizing test MSE
- Performing log or root transformation on response greatly reduced test MSE (after predictions returned to original scale), at the cost of stability
- By coincidence, the full model was in the middle of the pack.

Suggestions:

- Perform more data exploration

Retrospective

Trends:

- Models with more predictors tended to do better than models with fewer predictors
- Models with 0 or 1 interaction terms tended to do better than those with more interactions
- Transforming some predictor variables were helpful in stabilizing test MSE
- Performing log or root transformation on response greatly reduced test MSE (after predictions returned to original scale), at the cost of stability
- By coincidence, the full model was in the middle of the pack.

Suggestions:

- Perform more data exploration
- When implementing complexity increasing methods (non-linear terms and transformations), assess whether data supports inclusion

Retrospective

Trends:

- Models with more predictors tended to do better than models with fewer predictors
- Models with 0 or 1 interaction terms tended to do better than those with more interactions
- Transforming some predictor variables were helpful in stabilizing test MSE
- Performing log or root transformation on response greatly reduced test MSE (after predictions returned to original scale), at the cost of stability
- By coincidence, the full model was in the middle of the pack.

Suggestions:

- Perform more data exploration
- When implementing complexity increasing methods (non-linear terms and transformations), assess whether data supports inclusion
- Generally, adding predictors increases complexity by less than adding interaction terms or transformations

Section 2

Subset Selection

Methodology

Suppose we wish to find a linear model for Y with p predictors $X_1, \ldots, X_p$.

Methodology

Suppose we wish to find a linear model for Y with p predictors $X_1, \ldots, X_p$.

How do we determine the optimal collection of predictors?

Methodology

Suppose we wish to find a linear model for Y with p predictors $X_1, \ldots, X_p$.

How do we determine the optimal collection of predictors?

Determine an appropriate selection criteria.

Methodology

Suppose we wish to find a linear model for Y with p predictors $X_1, \ldots, X_p$.

How do we determine the optimal collection of predictors?

Determine an appropriate selection criteria.

- Cross-validation: Computationally expensive

Methodology

Suppose we wish to find a linear model for Y with p predictors $X_1, \ldots, X_p$.

How do we determine the optimal collection of predictors?

Determine an appropriate selection criteria.

- Cross-validation: Computationally expensive

Methodology

Suppose we wish to find a linear model for Y with p predictors $X_1, \ldots, X_p$.

How do we determine the optimal collection of predictors?

Determine an appropriate selection criteria.

- Cross-validation: Computationally expensive
- Adjusted R^2: Penalizes non-helpful predictors, but may overestimate test error rate.

Methodology

Suppose we wish to find a linear model for Y with p predictors $X_1, \ldots, X_p$.

How do we determine the optimal collection of predictors?

Determine an appropriate selection criteria.

- Cross-validation: Computationally expensive
- Adjusted R^2: Penalizes non-helpful predictors, but may overestimate test error rate.
- C_p: penalizes training RSS by typical discrepancy between test and training.

$$C_p = \frac{1}{n}(\mathrm{RSS} + 2d\hat{\sigma}^2)$$

Methodology

Suppose we wish to find a linear model for Y with p predictors $X_1, \ldots, X_p$.

How do we determine the optimal collection of predictors?

Determine an appropriate selection criteria.

- Cross-validation: Computationally expensive
- Adjusted R^2: Penalizes non-helpful predictors, but may overestimate test error rate.
- C_p: penalizes training RSS by typical discrepancy between test and training.

$$C_p = \frac{1}{n}(\mathrm{RSS} + 2d\hat{\sigma}^2)$$

- Akaike information criterion (AIC): uses method of maximum likelihood, assuming Normal errors

$$\mathrm{AIC} = \frac{1}{n\hat{\sigma}^2}(\mathrm{RSS} + 2d\hat{\sigma}^2)$$

Methodology

Suppose we wish to find a linear model for Y with p predictors $X_1, \ldots, X_p$.

How do we determine the optimal collection of predictors?

Determine an appropriate selection criteria.

- Cross-validation: Computationally expensive
- Adjusted R^2: Penalizes non-helpful predictors, but may overestimate test error rate.
- C_p: penalizes training RSS by typical discrepancy between test and training.

$$C_p = \frac{1}{n}(\mathrm{RSS} + 2d\hat{\sigma}^2)$$

- Akaike information criterion (AIC): uses method of maximum likelihood, assuming Normal errors

$$\mathrm{AIC} = \frac{1}{n\hat{\sigma}^2}(\mathrm{RSS} + 2d\hat{\sigma}^2)$$

- Bayesian information criterion (BIC): uses method of maximum likelihood and Bayes' Rule

$$\mathrm{BIC} = \frac{1}{n\hat{\sigma}^2}(\mathrm{RSS} + \ln nd\hat{\sigma}^2)$$

Best Subset

With p predictors, there are a total of 2^p possible MLR models.

- There are $\binom{p}{k}$ models using exactly k of p predictors

Best Subset

With p predictors, there are a total of 2^p possible MLR models.

- There are $\binom{p}{k}$ models using exactly k of p predictors

Theoretically, we can find the best model by fitting each possible model and selecting the best via appropriate selection criteria (C_p, AIC, BIC, R^2, CV)

Best Subset

With p predictors, there are a total of 2^p possible MLR models.

- There are $\binom{p}{k}$ models using exactly k of p predictors

Theoretically, we can find the best model by fitting each possible model and selecting the best via appropriate selection criteria (C_p, AIC, BIC, R^2, CV)

Downsides?

Best Subset

With p predictors, there are a total of 2^p possible MLR models.

- There are $\binom{p}{k}$ models using exactly k of p predictors

Theoretically, we can find the best model by fitting each possible model and selecting the best via appropriate selection criteria (C_p, AIC, BIC, R^2, CV)

Downsides?

- Computation time and storage grows exponentially in p

Best Subset

With p predictors, there are a total of 2^p possible MLR models.

- There are $\binom{p}{k}$ models using exactly k of p predictors

Theoretically, we can find the best model by fitting each possible model and selecting the best via appropriate selection criteria (C_p, AIC, BIC, R^2, CV)

Downsides?

- Computation time and storage grows exponentially in p
- May have low marginal improvement despite number of models fitted

Best Subset in R

We use the `regsubsets` function in the `leaps` library.

Best Subset in R

We use the `regsubsets` function in the `leaps` library.

- `regsubsets` uses the same syntax as `lm`. The `summary` function outputs the best set of variables for the given number of predictors

Best Subset in R

We use the `regsubsets` function in the `leaps` library.

- `regsubsets` uses the same syntax as `lm`. The `summary` function outputs the best set of variables for the given number of predictors
- Be default, `regsubsets` only returns up to the best eight models. But `nvmax` can be used to return as many variables as desired

Best Subset in R

We use the `regsubsets` function in the `leaps` library.

- `regsubsets` uses the same syntax as `lm`. The `summary` function outputs the best set of variables for the given number of predictors
- Be default, `regsubsets` only returns up to the best eight models. But `nvmax` can be used to return as many variables as desired
- Best is determined by *RSS*.

Best Subset in R

We use the `regsubsets` function in the `leaps` library.

- `regsubsets` uses the same syntax as `lm`. The `summary` function outputs the best set of variables for the given number of predictors
- Be default, `regsubsets` only returns up to the best eight models. But `nvmax` can be used to return as many variables as desired
- Best is determined by *RSS*.

```
library(palmerpenguins)
library(leaps)
penguins<-penguins %>% drop_na()

best_subset<-regsubsets(body_mass_g ~. , data = penguins, nvmax = 8)
```

Summary of regsubsets

- Stars indicate variable is included in model

```
## Subset selection object
## Call: regsubsets.formula(body_mass_g ~ ., data = penguins, nvmax = 8)
## 9 Variables  (and intercept)
##                   Forced in Forced out
## speciesChinstrap      FALSE      FALSE
## speciesGentoo         FALSE      FALSE
## islandDream           FALSE      FALSE
## islandTorgersen       FALSE      FALSE
## bill_length_mm        FALSE      FALSE
## bill_depth_mm         FALSE      FALSE
## flipper_length_mm     FALSE      FALSE
## sexmale               FALSE      FALSE
## year                  FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          speciesChinstrap speciesGentoo islandDream islandTorgersen
## 1  ( 1 ) " "              " "           " "         " "
## 2  ( 1 ) " "              "*"           " "         " "
## 3  ( 1 ) " "              "*"           " "         " "
## 4  ( 1 ) " "              "*"           " "         " "
## 5  ( 1 ) "*"              "*"           " "         " "
## 6  ( 1 ) "*"              "*"           " "         " "
## 7  ( 1 ) "*"              "*"           " "         " "
## 8  ( 1 ) "*"              "*"           " "         "*"
##          bill_length_mm bill_depth_mm flipper_length_mm sexmale year
## 1  ( 1 ) " "            " "           "*"               " "     " "
## 2  ( 1 ) " "            " "           " "               "*"     " "
## 3  ( 1 ) " "            " "           "*"               "*"     " "
## 4  ( 1 ) " "            "*"           "*"               "*"     " "
## 5  ( 1 ) " "            "*"           "*"               "*"     " "
## 6  ( 1 ) "*"            "*"           "*"               "*"     " "
## 7  ( 1 ) "*"            "*"           "*"               "*"     "*"
## 8  ( 1 ) "*"            "*"           "*"               "*"     "*"
```

Other Selection Metrics

The `summary` function can return selection metrics for each model.

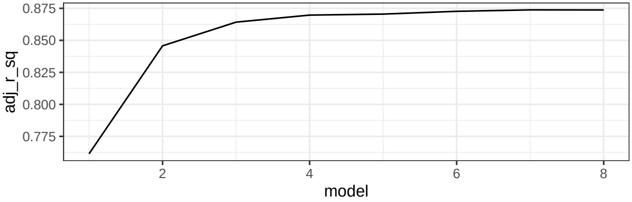
```
adj_r_sq<-summary(best_subset)$adjr2
rss<-summary(best_subset)$rss
cp<-summary(best_subset)$cp

d<-data.frame(model = 1:8, adj_r_sq, rss, cp )
d
```

```
##   model  adj_r_sq      rss         cp
## 1     1 0.7613734 51211963 294.805584
## 2     2 0.8457078 33012815  75.124367
## 3     3 0.8642104 28965893  27.829395
## 4     4 0.8697020 27709979  14.531285
## 5     5 0.8704945 27457472  13.455534
## 6     6 0.8726606 26915647   8.855638
## 7     7 0.8737834 26596486   6.967990
## 8     8 0.8737208 26527820   8.131576
```

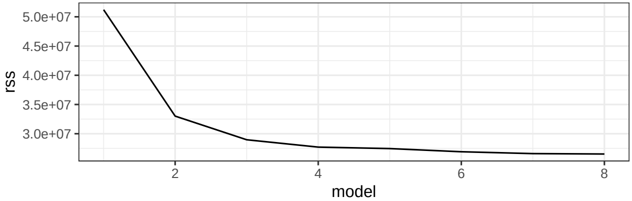
Plotting

We can use ggplot2 to visualize selection metric as a function of variable number

```
ggplot(d, aes(x = model, y = adj_r_sq))+geom_line()+theme_bw()
```




```
ggplot(d, aes(x = model, y = rss))+geom_line()+theme_bw()
```



Forward Selection

Forward selection is a *computationally efficient* alternative to best subset

Forward Selection

Forward selection is a *computationally efficient* alternative to best subset

- To perform forward selection, create the best 1 variable model. Then create $p - 1$ new 2 variable models by adding each other predictor one-at-a-time to the existing 1-variable model. Repeat for 3 variables and so on.

Forward Selection

Forward selection is a *computationally efficient* alternative to best subset

- To perform forward selection, create the best 1 variable model. Then create $p-1$ new 2 variable models by adding each other predictor one-at-a-time to the existing 1-variable model. Repeat for 3 variables and so on.
- Compared to Best Subset, forward selection computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$

Forward Selection

Forward selection is a *computationally efficient* alternative to best subset

- To perform forward selection, create the best 1 variable model. Then create $p - 1$ new 2 variable models by adding each other predictor one-at-a-time to the existing 1-variable model. Repeat for 3 variables and so on.
- Compared to Best Subset, forward selection computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Forward selection tends to favor parsimonious models

Forward Selection

Forward selection is a *computationally efficient* alternative to best subset

- To perform forward selection, create the best 1 variable model. Then create $p - 1$ new 2 variable models by adding each other predictor one-at-a-time to the existing 1-variable model. Repeat for 3 variables and so on.
- Compared to Best Subset, forward selection computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Forward selection tends to favor parsimonious models
- Downsides?

Forward Selection

Forward selection is a *computationally efficient* alternative to best subset

- To perform forward selection, create the best 1 variable model. Then create $p-1$ new 2 variable models by adding each other predictor one-at-a-time to the existing 1-variable model. Repeat for 3 variables and so on.
- Compared to Best Subset, forward selection computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Forward selection tends to favor parsimonious models
- Downsides?
    - Not guaranteed to find the best model (or even something close to the best model)

Forward Selection

Forward selection is a *computationally efficient* alternative to best subset

- To perform forward selection, create the best 1 variable model. Then create $p-1$ new 2 variable models by adding each other predictor one-at-a-time to the existing 1-variable model. Repeat for 3 variables and so on.
- Compared to Best Subset, forward selection computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Forward selection tends to favor parsimonious models
- Downsides?
    - Not guaranteed to find the best model (or even something close to the best model)
    - Early predictors may become redundant

Forward Selection

Forward selection is a *computationally efficient* alternative to best subset

- To perform forward selection, create the best 1 variable model. Then create $p-1$ new 2 variable models by adding each other predictor one-at-a-time to the existing 1-variable model. Repeat for 3 variables and so on.
- Compared to Best Subset, forward selection computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Forward selection tends to favor parsimonious models
- Downsides?
  - Not guaranteed to find the best model (or even something close to the best model)
  - Early predictors may become redundant
  - Can be unstable

Backward Elimination

Backward Elimination is another *computationally efficient* alternative to best subset

Backward Elimination

Backward Elimination is another *computationally efficient* alternative to best subset

- To perform backward selection, begin with full model. Then create $p-1$ new $p-1$ variable models by removing one-at-a-time each other predictor from the existing p-variable model. Repeat for $p-2$ variables and so on.

Backward Elimination

Backward Elimination is another *computationally efficient* alternative to best subset

- To perform backward selection, begin with full model. Then create $p-1$ new $p-1$ variable models by removing one-at-a-time each other predictor from the existing p-variable model. Repeat for $p-2$ variables and so on.
- Compared to Best Subset, backward elimination computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$

Backward Elimination

Backward Elimination is another *computationally efficient* alternative to best subset

- To perform backward selection, begin with full model. Then create $p-1$ new $p-1$ variable models by removing one-at-a-time each other predictor from the existing p-variable model. Repeat for $p-2$ variables and so on.
- Compared to Best Subset, backward elimination computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Backward elimination tends to favor in-depth models

Backward Elimination

Backward Elimination is another *computationally efficient* alternative to best subset

- To perform backward selection, begin with full model. Then create $p-1$ new $p-1$ variable models by removing one-at-a-time each other predictor from the existing p-variable model. Repeat for $p-2$ variables and so on.
- Compared to Best Subset, backward elimination computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Backward elimination tends to favor in-depth models
- Downsides?

Backward Elimination

Backward Elimination is another *computationally efficient* alternative to best subset

- To perform backward selection, begin with full model. Then create $p-1$ new $p-1$ variable models by removing one-at-a-time each other predictor from the existing p-variable model. Repeat for $p-2$ variables and so on.
- Compared to Best Subset, backward elimination computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Backward elimination tends to favor in-depth models
- Downsides?
  - Not guaranteed to find the best model (or even something close to the best model)

Backward Elimination

Backward Elimination is another *computationally efficient* alternative to best subset

- To perform backward selection, begin with full model. Then create $p-1$ new $p-1$ variable models by removing one-at-a-time each other predictor from the existing p-variable model. Repeat for $p-2$ variables and so on.
- Compared to Best Subset, backward elimination computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Backward elimination tends to favor in-depth models
- Downsides?
  - Not guaranteed to find the best model (or even something close to the best model)
  - Requires fewer predictors than observations

Backward Elimination

Backward Elimination is another *computationally efficient* alternative to best subset

- To perform backward selection, begin with full model. Then create $p-1$ new $p-1$ variable models by removing one-at-a-time each other predictor from the existing p-variable model. Repeat for $p-2$ variables and so on.
- Compared to Best Subset, backward elimination computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Backward elimination tends to favor in-depth models
- Downsides?
  - Not guaranteed to find the best model (or even something close to the best model)
  - Requires fewer predictors than observations
  - Susceptible to multicollinearity

Backward Elimination

Backward Elimination is another *computationally efficient* alternative to best subset

- To perform backward selection, begin with full model. Then create $p-1$ new $p-1$ variable models by removing one-at-a-time each other predictor from the existing p-variable model. Repeat for $p-2$ variables and so on.
- Compared to Best Subset, backward elimination computation time grows polynomially in p: $\text{Num. Models} = 1 + \frac{p(p+1)}{2}$
- Backward elimination tends to favor in-depth models
- Downsides?
  - Not guaranteed to find the best model (or even something close to the best model)
  - Requires fewer predictors than observations
  - Susceptible to multicollinearity
  - Can be unstable

Forward/Backward Selection in R

We again use the `regsubsets` function in the `leaps` library.

```
forward_select<-regsubsets(body_mass_g ~. , data = penguins, nvmax = 8,
                           method = "forward")

backward_elim<-regsubsets(body_mass_g ~. , data = penguins, nvmax = 8,
                          method = "backward")
```

Summary of Forward Selection

```r
summary(forward_select)
```

```
## Subset selection object
## Call: regsubsets.formula(body_mass_g ~ ., data = penguins, nvmax = 8,
##     method = "forward")
## 9 Variables  (and intercept)
##                   Forced in Forced out
## speciesChinstrap      FALSE      FALSE
## speciesGentoo         FALSE      FALSE
## islandDream           FALSE      FALSE
## islandTorgersen       FALSE      FALSE
## bill_length_mm        FALSE      FALSE
## bill_depth_mm         FALSE      FALSE
## flipper_length_mm     FALSE      FALSE
## sexmale               FALSE      FALSE
## year                  FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: forward
##          speciesChinstrap speciesGentoo islandDream islandTorgersen
## 1  ( 1 ) " "              " "           " "         " "
## 2  ( 1 ) " "              " "           " "         " "
## 3  ( 1 ) " "              "*"           " "         " "
## 4  ( 1 ) " "              "*"           " "         " "
## 5  ( 1 ) "*"              "*"           " "         " "
## 6  ( 1 ) "*"              "*"           " "         " "
## 7  ( 1 ) "*"              "*"           " "         " "
## 8  ( 1 ) "*"              "*"           " "         "*"
##          bill_length_mm bill_depth_mm flipper_length_mm sexmale year
## 1  ( 1 ) " "            " "           "*"               " "     " "
## 2  ( 1 ) " "            " "           "*"               "*"     " "
## 3  ( 1 ) " "            " "           "*"               "*"     " "
## 4  ( 1 ) " "            "*"           "*"               "*"     " "
## 5  ( 1 ) " "            "*"           "*"               "*"     " "
## 6  ( 1 ) "*"            "*"           "*"               "*"     " "
## 7  ( 1 ) "*"            "*"           "*"               "*"     "*"
## 8  ( 1 ) "*"            "*"           "*"               "*"     "*"
```

Summary of Backward Elimination

```r
summary(backward_elim)
```

```
## Subset selection object
## Call: regsubsets.formula(body_mass_g ~ ., data = penguins, nvmax = 8,
##     method = "backward")
## 9 Variables  (and intercept)
##                   Forced in Forced out
## speciesChinstrap      FALSE      FALSE
## speciesGentoo         FALSE      FALSE
## islandDream           FALSE      FALSE
## islandTorgersen       FALSE      FALSE
## bill_length_mm        FALSE      FALSE
## bill_depth_mm         FALSE      FALSE
## flipper_length_mm     FALSE      FALSE
## sexmale               FALSE      FALSE
## year                  FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: backward
##          speciesChinstrap speciesGentoo islandDream islandTorgersen
## 1  ( 1 ) " "              "*"           " "         " "
## 2  ( 1 ) " "              "*"           " "         " "
## 3  ( 1 ) " "              "*"           " "         " "
## 4  ( 1 ) " "              "*"           " "         " "
## 5  ( 1 ) "*"              "*"           " "         " "
## 6  ( 1 ) "*"              "*"           " "         " "
## 7  ( 1 ) "*"              "*"           " "         " "
## 8  ( 1 ) "*"              "*"           " "         "*"
##          bill_length_mm bill_depth_mm flipper_length_mm sexmale year
## 1  ( 1 ) " "            " "           " "               " "     " "
## 2  ( 1 ) " "            " "           " "               "*"     " "
## 3  ( 1 ) " "            " "           "*"               "*"     " "
## 4  ( 1 ) " "            "*"           "*"               "*"     " "
## 5  ( 1 ) " "            "*"           "*"               "*"     " "
## 6  ( 1 ) "*"            "*"           "*"               "*"     " "
## 7  ( 1 ) "*"            "*"           "*"               "*"     "*"
## 8  ( 1 ) "*"            "*"           "*"               "*"     "*"
```