Feature Selection and Engineering

Prof Wells

STA 295: Stat Learning

March 7th, 2024

Outline

In today's class, we will...

- Perform some exploratory data analysis on a new data set
- Investigate algorithms for selecting good subsets of predictors
- Discuss ways to create and modify predictors

Section 1

Explaratory Data Analysis

Molecular Solubility

The `solubility` data set from the `AppliedPredictiveModeling` package contains solubility and chemical structure for a sample of 1,267 different compounds.

Molecular Solubility

The `solubility` data set from the `AppliedPredictiveModeling` package contains solubility and chemical structure for a sample of 1,267 different compounds.

```
library(AppliedPredictiveModeling)
data(solubility)
```

Molecular Solubility

The `solubility` data set from the `AppliedPredictiveModeling` package contains solubility and chemical structure for a sample of 1,267 different compounds.

```
library(AppliedPredictiveModeling)
data(solubility)
```

- The solubility of a compound indicates how easily it dissolves in a solvent (often water), and is measured as the amount of solvent required to dissolve 1 part of the compound.
  - The less solvent required, the more soluble the compound.
  - In the dataset, the log solubility is reported, since solubility spans many orders of magnitude

Molecular Solubility

The `solubility` data set from the `AppliedPredictiveModeling` package contains solubility and chemical structure for a sample of 1,267 different compounds.

```
library(AppliedPredictiveModeling)
data(solubility)
```

- The solubility of a compound indicates how easily it dissolves in a solvent (often water), and is measured as the amount of solvent required to dissolve 1 part of the compound.
  - The less solvent required, the more soluble the compound.
  - In the dataset, the log solubility is reported, since solubility spans many orders of magnitude
- The data also contains 16 chemical count descriptors, such as "number of bonds" or "number of bromine atoms"

Molecular Solubility

The `solubility` data set from the `AppliedPredictiveModeling` package contains solubility and chemical structure for a sample of 1,267 different compounds.

```
library(AppliedPredictiveModeling)
data(solubility)
```

- The solubility of a compound indicates how easily it dissolves in a solvent (often water), and is measured as the amount of solvent required to dissolve 1 part of the compound.
  - The less solvent required, the more soluble the compound.
  - In the dataset, the log solubility is reported, since solubility spans many orders of magnitude
- The data also contains 16 chemical count descriptors, such as "number of bonds" or "number of bromine atoms"
- Finally, the data contains 4 continuous descriptors, such as "molecular weight" or "surface area"

Molecular Solubility

The `solubility` data set from the `AppliedPredictiveModeling` package contains solubility and chemical structure for a sample of 1,267 different compounds.

```
library(AppliedPredictiveModeling)
data(solubility)
```

- The solubility of a compound indicates how easily it dissolves in a solvent (often water), and is measured as the amount of solvent required to dissolve 1 part of the compound.
  - The less solvent required, the more soluble the compound.
  - In the dataset, the log solubility is reported, since solubility spans many orders of magnitude
- The data also contains 16 chemical count descriptors, such as "number of bonds" or "number of bromine atoms"
- Finally, the data contains 4 continuous descriptors, such as "molecular weight" or "surface area"

We are interested in determining solubility based on these 20 chemical descriptors.

Pre-Processing

- The `solubability` actually consists of 4 data sets: `solTestX`, `solTrainX`, `solTestY`, `solTrainY`

Pre-Processing

- The `solubability` actually consists of 4 data sets: `solTestX`, `solTrainX`, `solTestY`, `solTrainY`
- The `X` and `Y` indicate the data is pre-divided into separate sets for predictors and response.

Pre-Processing

- The `solubability` actually consists of 4 data sets: `solTestX`, `solTrainX`, `solTestY`, `solTrainY`
- The `X` and `Y` indicate the data is pre-divided into separate sets for predictors and response.
- Additionally, data have already been partitioned into test and training sets (25 / 75)

Pre-Processing

- The `solubability` actually consists of 4 data sets: `solTestX`, `solTrainX`, `solTestY`, `solTrainY`
- The `X` and `Y` indicate the data is pre-divided into separate sets for predictors and response.
- Additionally, data have already been partitioned into test and training sets (25 / 75)
- It will be easier to have predictors and response in the same set, so we'll bind columns together:

```
solTest <- cbind(solTestX, Solubility = solTestY)
solTrain <- cbind(solTrainX, Solubility =  solTrainY)
```

Pre-Processing

- The `solubability` actually consists of 4 data sets: `solTestX`, `solTrainX`, `solTestY`, `solTrainY`
- The `X` and `Y` indicate the data is pre-divided into separate sets for predictors and response.
- Additionally, data have already been partitioned into test and training sets (25 / 75)
- It will be easier to have predictors and response in the same set, so we'll bind columns together:

```
solTest <- cbind(solTestX, Solubility = solTestY)
solTrain <- cbind(solTrainX, Solubility =  solTrainY)
```

- The data also contains 218 binary "fingerprints" for each compound indicating presence of particular chemical substructure, each beginning with "FP"

Pre-Processing

- The `solubability` actually consists of 4 data sets: `solTestX`, `solTrainX`, `solTestY`, `solTrainY`
- The `X` and `Y` indicate the data is pre-divided into separate sets for predictors and response.
- Additionally, data have already been partitioned into test and training sets (25 / 75)
- It will be easier to have predictors and response in the same set, so we'll bind columns together:

```
solTest <- cbind(solTestX, Solubility = solTestY)
solTrain <- cbind(solTrainX, Solubility =  solTrainY)
```

- The data also contains 218 binary "fingerprints" for each compound indicating presence of particular chemical substructure, each beginning with "FP"
- We'll ignore these predictors for now.

```
library(dplyr)
solTest <- solTest %>% select(!starts_with("FP"))
solTrain <- solTrain %>% select(!starts_with("FP"))
```

Distribution of Variables

- In our initial exploratory analysis, we will investigate the distribution of the response, as well as correlations between the response and each quantitative predictor.
  - We should do this using only the training set. (Why?)

Distribution of Variables

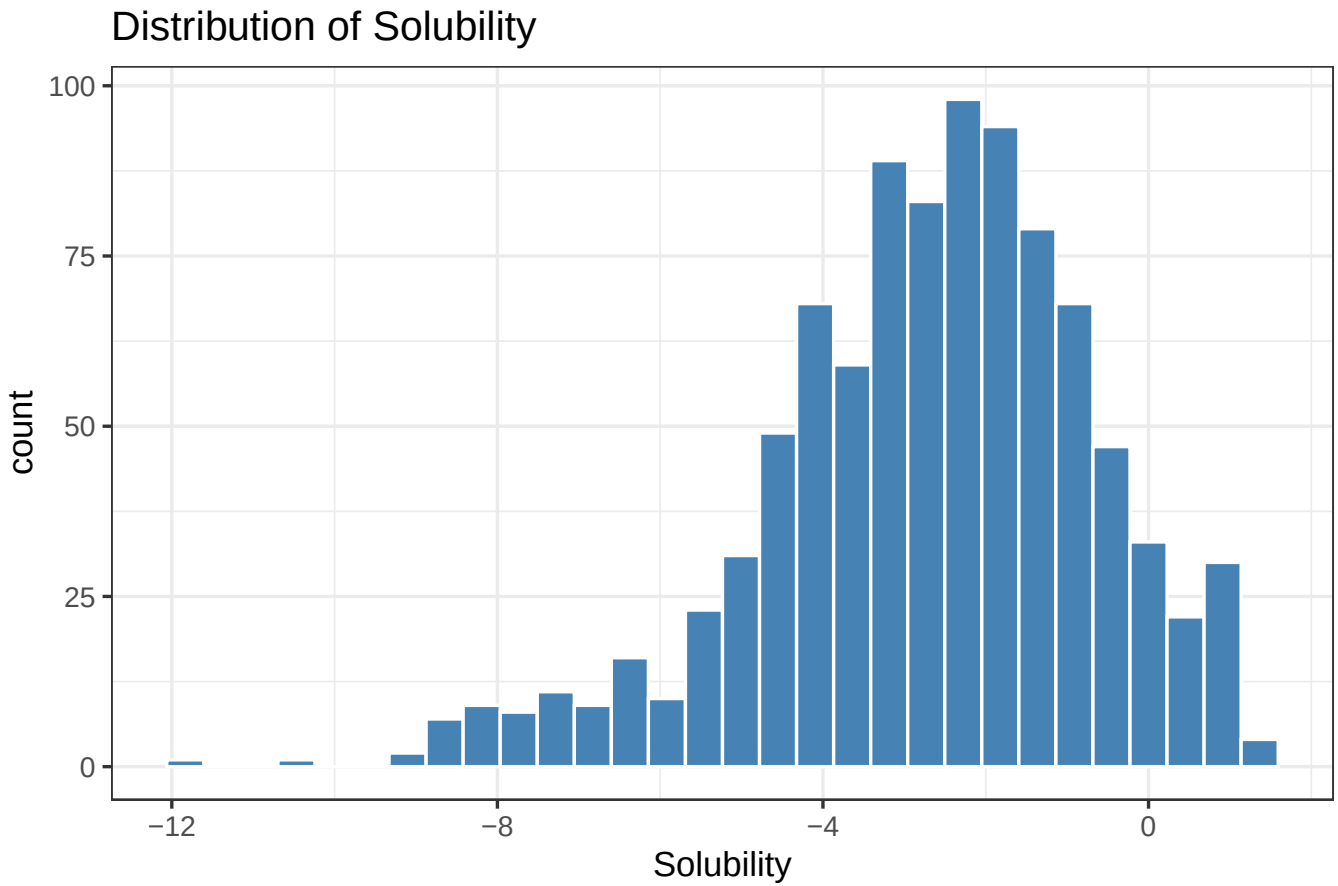
- In our initial exploratory analysis, we will investigate the distribution of the response, as well as correlations between the response and each quantitative predictor.
  - We should do this using only the training set. (Why?)
- If we had categorical predictors, we could also look at side-by-side boxplots, and compute means of the response within each level of the categorical predictor.
  - But we don't have any categorical variables in this data set

Distribution of Variables

- In our initial exploratory analysis, we will investigate the distribution of the response, as well as correlations between the response and each quantitative predictor.
  - We should do this using only the training set. (Why?)
- If we had categorical predictors, we could also look at side-by-side boxplots, and compute means of the response within each level of the categorical predictor.
  - But we don't have any categorical variables in this data set
- We should also assess whether we have any missing values

Response Histogram

```
ggplot(solTrain, aes(x = Solubility))+
  geom_histogram(color = "white", fill = "steelblue")
```



Response Summary Statistics

```
solTrain %>% summarize(
  min = min(Solubility),
  Q1 = quantile(Solubility, 0.25),
  median = median(Solubility),
  Q3 = quantile(Solubility, 0.75),
  max = max(Solubility),
  mean = mean(Solubility),
  sd = sd(Solubility))
```

```
##      min     Q1 median    Q3  max     mean       sd
## 1 -11.62 -3.955  -2.51 -1.36 1.58 -2.71857 2.046641
```

Pairwise Scatterplot

It would be helpful to visualize the relationship between the response and each quantitative variable

Pairwise Scatterplot

It would be helpful to visualize the relationship between the response and each quantitative variable

- We *could* individually code each plot (not too burdensome if there are only few predictors).

Pairwise Scatterplot

It would be helpful to visualize the relationship between the response and each quantitative variable

- We *could* individually code each plot (not too burdensome if there are only few predictors).
- But with many predictors, that's lots of redundant coding.

Pairwise Scatterplot

It would be helpful to visualize the relationship between the response and each quantitative variable

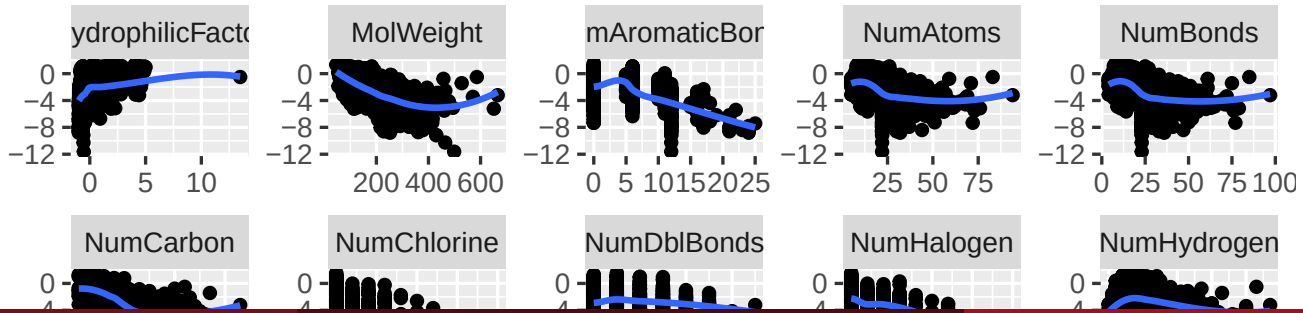
- We *could* individually code each plot (not too burdensome if there are only few predictors).
- But with many predictors, that's lots of redundant coding.
  - Instead, we can make use of the `pivot_longer` function from `tidyr`:

Pairwise Scatterplot

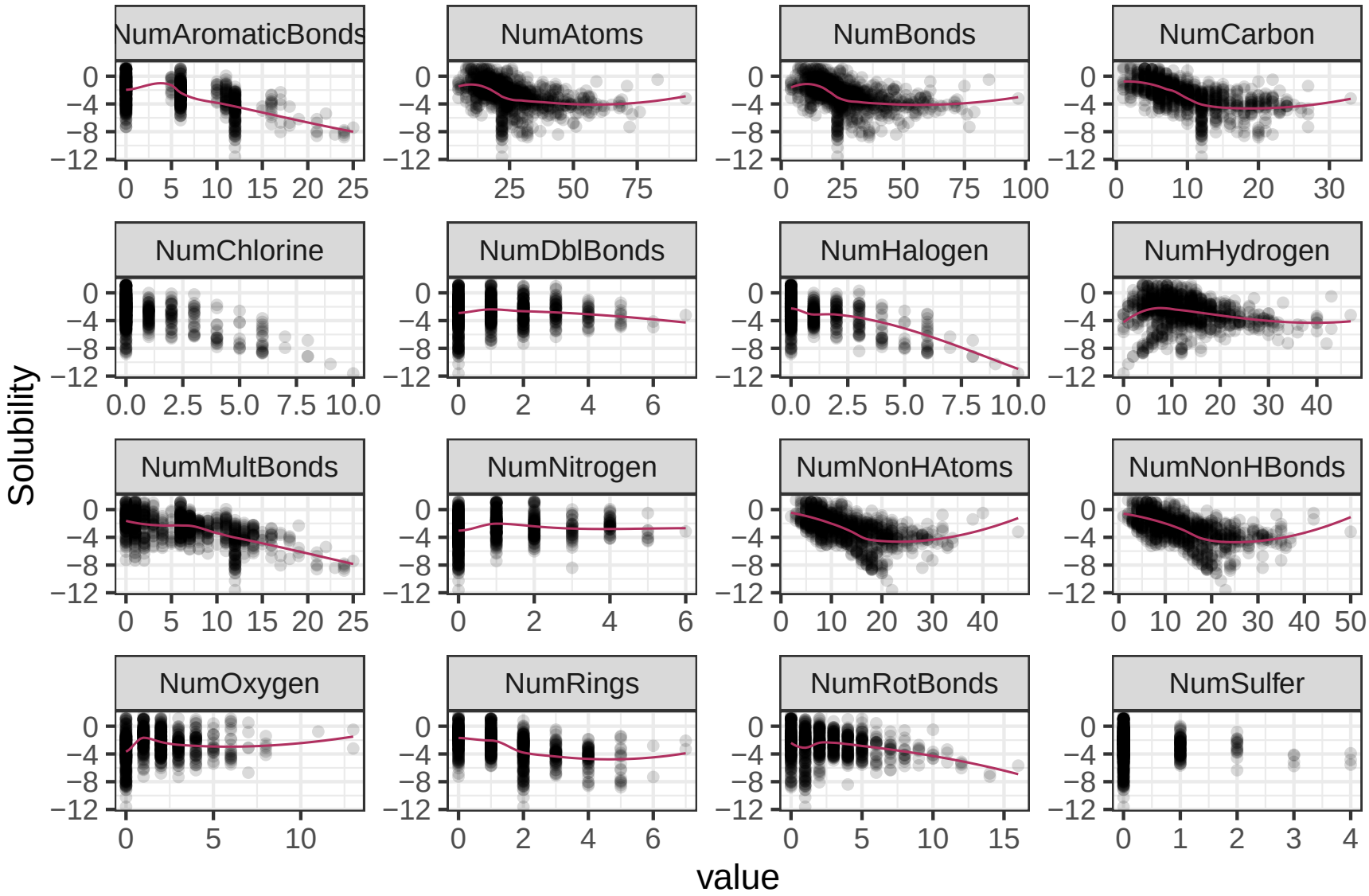
It would be helpful to visualize the relationship between the response and each quantitative variable

- We *could* individually code each plot (not too burdensome if there are only few predictors).
- But with many predictors, that's lots of redundant coding.
  - Instead, we can make use of the `pivot_longer` function from `tidyr`:

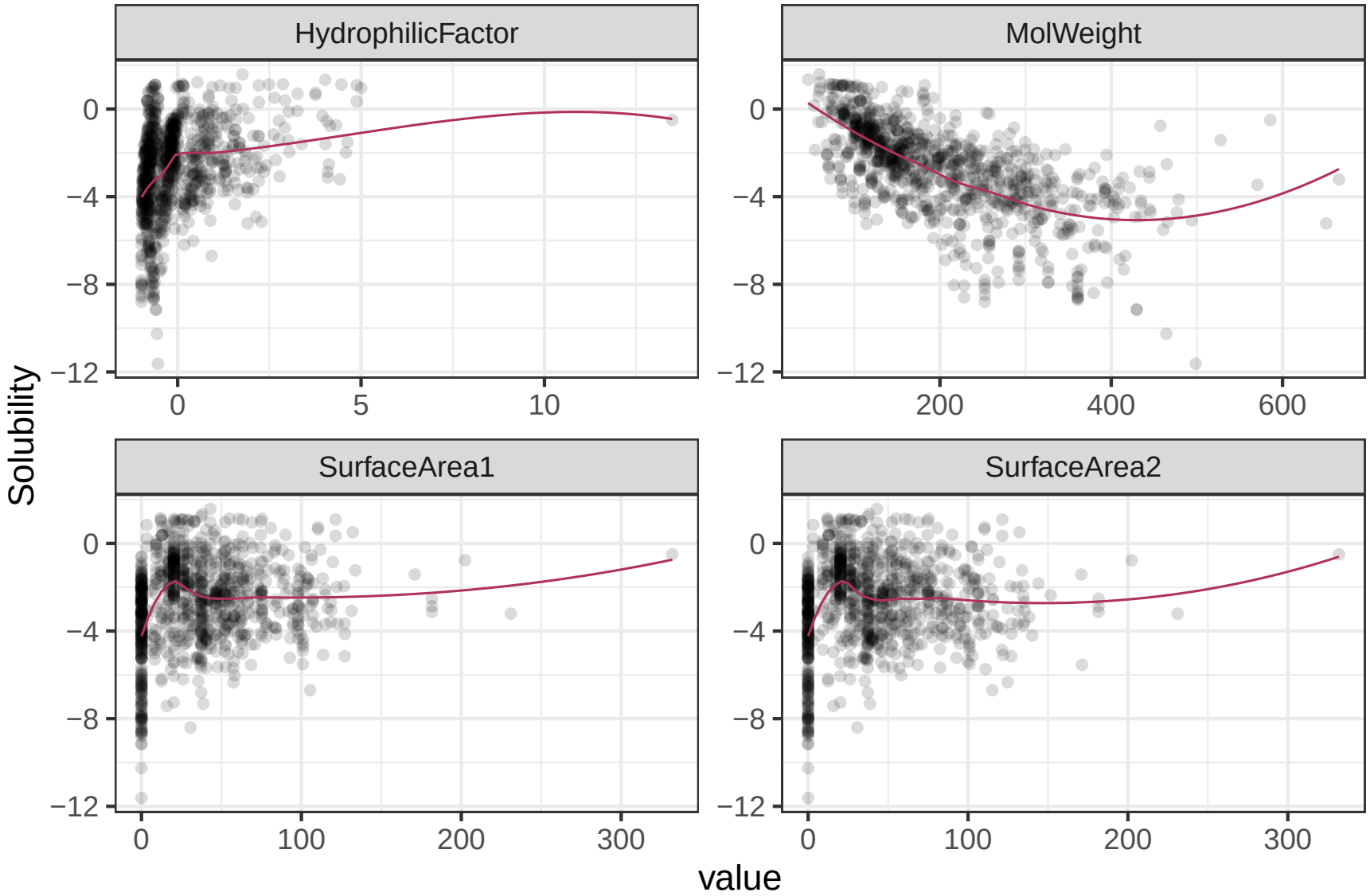
```
solTrain %>% pivot_longer(-Solubility, names_to = "variables",
                          values_to = "value") %>%
  ggplot(aes(x = value, y = Solubility ))+
    geom_point()+
  geom_smooth(se = F)+
  facet_wrap(~variables, scales = "free")
```



Pairwise Scatterplots



Pairwise Scatterplots



Correlation with Response

```
cor_values <- cor(solTrain[,-21], solTrain$Solubility)
# obtains correlations between each variable and the response
# stores results as a 1 x 20 matrix

cor_names <- rownames(cor_values)
# extracts names for each variable

cor_df <- data.frame(correlation = as.numeric(cor_values), variable =  cor_names)
# creates data frame with correlations and variable names
# as.numeric coerces the cor_values matrix into a vector
```

Correlation with Response

```
cor_df
```

```
##     correlation          variable
## 1  -0.629163885         MolWeight
## 2  -0.398943188          NumAtoms
## 3  -0.544646707      NumNonHAtoms
## 4  -0.420459121          NumBonds
## 5  -0.551457131      NumNonHBonds
## 6  -0.525248387      NumMultBonds
## 7  -0.149343282       NumRotBonds
## 8   0.001237051       NumDblBonds
## 9  -0.515883692  NumAromaticBonds
## 10 -0.204082828       NumHydrogen
## 11 -0.582761107         NumCarbon
## 12  0.102230176       NumNitrogen
## 13  0.130774566         NumOxygen
## 14 -0.091418407         NumSulfer
## 15 -0.504054819       NumChlorine
## 16 -0.504136055        NumHalogen
## 17 -0.488295986          NumRings
## 18  0.309022159 HydrophilicFactor
## 19  0.193769382      SurfaceArea1
## 20  0.143941883      SurfaceArea2
```

Correlation with Response

```
cor_df
```

```
##      correlation          variable
## 1  -0.629163885         MolWeight
## 2  -0.398943188          NumAtoms
## 3  -0.544646707      NumNonHAtoms
## 4  -0.420459121          NumBonds
## 5  -0.551457131      NumNonHBonds
## 6  -0.525248387      NumMultBonds
## 7  -0.149343282       NumRotBonds
## 8   0.001237051       NumDblBonds
## 9  -0.515883692  NumAromaticBonds
## 10 -0.204082828      NumHydrogen
## 11 -0.582761107        NumCarbon
## 12  0.102230176      NumNitrogen
## 13  0.130774566        NumOxygen
## 14 -0.091418407        NumSulfer
## 15 -0.504054819      NumChlorine
## 16 -0.504136055       NumHalogen
## 17 -0.488295986         NumRings
## 18  0.309022159 HydrophilicFactor
## 19  0.193769382      SurfaceArea1
## 20  0.143941883      SurfaceArea2
```

```
cor_df %>% arrange(desc(abs(correlation)))
```

```
##      correlation          variable
## 1  -0.629163885         MolWeight
## 2  -0.582761107        NumCarbon
## 3  -0.551457131      NumNonHBonds
## 4  -0.544646707      NumNonHAtoms
## 5  -0.525248387      NumMultBonds
## 6  -0.515883692  NumAromaticBonds
## 7  -0.504136055       NumHalogen
## 8  -0.504054819      NumChlorine
## 9  -0.488295986         NumRings
## 10 -0.420459121          NumBonds
## 11 -0.398943188          NumAtoms
## 12  0.309022159 HydrophilicFactor
## 13 -0.204082828      NumHydrogen
## 14  0.193769382      SurfaceArea1
## 15 -0.149343282       NumRotBonds
## 16  0.143941883      SurfaceArea2
## 17  0.130774566        NumOxygen
## 18  0.102230176      NumNitrogen
## 19 -0.091418407        NumSulfer
## 20  0.001237051       NumDblBonds
```

Correlation Matrix

- We now have a good idea about which variables are most strongly correlated with the response

Correlation Matrix

- We now have a good idea about which variables are most strongly correlated with the response

```
##   correlation     variable
## 1  -0.6291639    MolWeight
## 2  -0.5827611    NumCarbon
## 3  -0.5514571 NumNonHBonds
## 4  -0.5446467 NumNonHAtoms
## 5  -0.5252484 NumMultBonds
```

Correlation Matrix

- We now have a good idea about which variables are most strongly correlated with the response

```
##   correlation     variable
## 1  -0.6291639    MolWeight
## 2  -0.5827611    NumCarbon
## 3  -0.5514571 NumNonHBonds
## 4  -0.5446467 NumNonHAtoms
## 5  -0.5252484 NumMultBonds
```

- But how do these variables relate to each other?

Correlation Matrix

- We now have a good idea about which variables are most strongly correlated with the response

```
##   correlation     variable
## 1  -0.6291639    MolWeight
## 2  -0.5827611    NumCarbon
## 3  -0.5514571 NumNonHBonds
## 4  -0.5446467 NumNonHAtoms
## 5  -0.5252484 NumMultBonds
```

- But how do these variables relate to each other?

- We can use the ggcorr function from the GGally package to quickly create a visual correlation matrix:

```
library(GGally)
ggcorr(solTrain, hjust = 1, size = 2, layout.exp = 5)

# hjust changes the position of the names
# size changes the size of names
# layout.exp expands the horizontal axis to prevent text clipping
# other options are possible (use ?ggcorr)
```

Correlation Matrix

Collinearity

- What are downsides of fitting the full model?

Collinearity

- What are downsides of fitting the full model?
  - Risk of overfitting
  - Perfect linear relationship between some predictors
  - Multicollinearity may lead to higher variance in model estimates, and hence, higher variance in predictions

Collinearity

- What are downsides of fitting the full model?
  - Risk of overfitting
  - Perfect linear relationship between some predictors
  - Multicollinearity may lead to higher variance in model estimates, and hence, higher variance in predictions
- Why should we fit the full model anyway?

Collinearity

- What are downsides of fitting the full model?
  - Risk of overfitting
  - Perfect linear relationship between some predictors
  - Multicollinearity may lead to higher variance in model estimates, and hence, higher variance in predictions
- Why should we fit the full model anyway?
  - We can get a baseline for model performance (using cross-validation)
  - We can identify possible problems using diagnostics
  - We have the variables, so we may as well try to use them

Collinearity

- What are downsides of fitting the full model?
  - Risk of overfitting
  - Perfect linear relationship between some predictors
  - Multicollinearity may lead to higher variance in model estimates, and hence, higher variance in predictions
- Why should we fit the full model anyway?
  - We can get a baseline for model performance (using cross-validation)
  - We can identify possible problems using diagnostics
  - We have the variables, so we may as well try to use them

```
full_mod <- lm(Solubility ~ ., data = solTrain)
```

Model Summary

```
##
## Call:
## lm(formula = Solubility ~ ., data = solTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8499 -0.5963  0.0232  0.5842  2.7848
##
## Coefficients: (3 not defined because of singularities)
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.344876   0.149393   2.309 0.021189 *
## MolWeight         -0.008074   0.001325  -6.093 1.61e-09 ***
## NumAtoms           0.275577   0.086182   3.198 0.001432 **
## NumNonHAtoms       1.536062   0.450948   3.406 0.000687 ***
## NumBonds          -0.612747   0.127856  -4.792 1.92e-06 ***
## NumNonHBonds             NA         NA      NA       NA
## NumMultBonds      -1.694110   0.321514  -5.269 1.70e-07 ***
## NumRotBonds       -0.147637   0.026894  -5.490 5.19e-08 ***
## NumDblBonds        0.771793   0.234853   3.286 0.001053 **
## NumAromaticBonds   1.278539   0.277614   4.605 4.69e-06 ***
## NumHydrogen              NA         NA      NA       NA
## NumCarbon         -0.650678   0.331825  -1.961 0.050187 .
## NumNitrogen       -0.222086   0.373396  -0.595 0.552140
## NumOxygen         -0.300338   0.424632  -0.707 0.479563
## NumSulfer          0.621244   0.298101   2.084 0.037432 *
## NumChlorine       -0.374042   0.061636  -6.069 1.87e-09 ***
## NumHalogen        -1.579937   0.459350  -3.440 0.000609 ***
## NumRings                 NA         NA      NA       NA
## HydrophilicFactor  0.162663   0.073229   2.221 0.026570 *
## SurfaceArea1       0.047692   0.013827   3.449 0.000587 ***
## SurfaceArea2      -0.070007   0.013245  -5.285 1.56e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9044 on 933 degrees of freedom
## Multiple R-squared:  0.8082, Adjusted R-squared:  0.8047
## F-statistic: 231.3 on 17 and 933 DF,  p-value: < 2.2e-16
```

Model Summary

```
##
## Call:
## lm(formula = Solubility ~ ., data = solTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8499 -0.5963  0.0232  0.5842  2.7848
##
## Coefficients: (3 not defined because of singularities)
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.344876   0.149393   2.309 0.021189 *
## MolWeight         -0.008074   0.001325  -6.093 1.61e-09 ***
## NumAtoms           0.275577   0.086182   3.198 0.001432 **
## NumNonHAtoms       1.536062   0.450948   3.406 0.000687 ***
## NumBonds          -0.612747   0.127856  -4.792 1.92e-06 ***
## NumNonHBonds             NA         NA      NA       NA
## NumMultBonds      -1.694110   0.321514  -5.269 1.70e-07 ***
## NumRotBonds       -0.147637   0.026894  -5.490 5.19e-08 ***
## NumDblBonds        0.771793   0.234853   3.286 0.001053 **
## NumAromaticBonds   1.278539   0.277614   4.605 4.69e-06 ***
## NumHydrogen              NA         NA      NA       NA
## NumCarbon         -0.650678   0.331825  -1.961 0.050187 .
## NumNitrogen       -0.222086   0.373396  -0.595 0.552140
## NumOxygen         -0.300338   0.424632  -0.707 0.479563
## NumSulfer          0.621244   0.298101   2.084 0.037432 *
## NumChlorine       -0.374042   0.061636  -6.069 1.87e-09 ***
## NumHalogen        -1.579937   0.459350  -3.440 0.000609 ***
## NumRings                 NA         NA      NA       NA
## HydrophilicFactor  0.162663   0.073229   2.221 0.026570 *
## SurfaceArea1       0.047692   0.013827   3.449 0.000587 ***
## SurfaceArea2      -0.070007   0.013245  -5.285 1.56e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9044 on 933 degrees of freedom
## Multiple R-squared:  0.8082, Adjusted R-squared:  0.8047
## F-statistic: 231.3 on 17 and 933 DF,  p-value: < 2.2e-16
```

NAs in the table mean we have a perfect linear relationship among some of the predictors

Model Summary

```
##
## Call:
## lm(formula = Solubility ~ ., data = solTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8499 -0.5963  0.0232  0.5842  2.7848
##
## Coefficients: (3 not defined because of singularities)
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.344876   0.149393   2.309 0.021189 *
## MolWeight         -0.008074   0.001325  -6.093 1.61e-09 ***
## NumAtoms           0.275577   0.086182   3.198 0.001432 **
## NumNonHAtoms       1.536062   0.450948   3.406 0.000687 ***
## NumBonds          -0.612747   0.127856  -4.792 1.92e-06 ***
## NumNonHBonds             NA         NA      NA       NA
## NumMultBonds      -1.694110   0.321514  -5.269 1.70e-07 ***
## NumRotBonds       -0.147637   0.026894  -5.490 5.19e-08 ***
## NumDblBonds        0.771793   0.234853   3.286 0.001053 **
## NumAromaticBonds   1.278539   0.277614   4.605 4.69e-06 ***
## NumHydrogen              NA         NA      NA       NA
## NumCarbon         -0.650678   0.331825  -1.961 0.050187 .
## NumNitrogen       -0.222086   0.373396  -0.595 0.552140
## NumOxygen         -0.300338   0.424632  -0.707 0.479563
## NumSulfer          0.621244   0.298101   2.084 0.037432 *
## NumChlorine       -0.374042   0.061636  -6.069 1.87e-09 ***
## NumHalogen        -1.579937   0.459350  -3.440 0.000609 ***
## NumRings                 NA         NA      NA       NA
## HydrophilicFactor  0.162663   0.073229   2.221 0.026570 *
## SurfaceArea1       0.047692   0.013827   3.449 0.000587 ***
## SurfaceArea2      -0.070007   0.013245  -5.285 1.56e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9044 on 933 degrees of freedom
## Multiple R-squared:  0.8082, Adjusted R-squared:  0.8047
## F-statistic: 231.3 on 17 and 933 DF,  p-value: < 2.2e-16
```

NAs in the table mean we have a perfect linear relationship among some of the predictors

- As a result, R dropped the linearly related variables.

Model Summary

```
##
## Call:
## lm(formula = Solubility ~ ., data = solTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8499 -0.5963  0.0232  0.5842  2.7848
##
## Coefficients: (3 not defined because of singularities)
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.344876   0.149393   2.309 0.021189 *
## MolWeight         -0.008074   0.001325  -6.093 1.61e-09 ***
## NumAtoms           0.275577   0.086182   3.198 0.001432 **
## NumNonHAtoms       1.536062   0.450948   3.406 0.000687 ***
## NumBonds          -0.612747   0.127856  -4.792 1.92e-06 ***
## NumNonHBonds             NA         NA      NA       NA
## NumMultBonds      -1.694110   0.321514  -5.269 1.70e-07 ***
## NumRotBonds       -0.147637   0.026894  -5.490 5.19e-08 ***
## NumDblBonds        0.771793   0.234853   3.286 0.001053 **
## NumAromaticBonds   1.278539   0.277614   4.605 4.69e-06 ***
## NumHydrogen              NA         NA      NA       NA
## NumCarbon         -0.650678   0.331825  -1.961 0.050187 .
## NumNitrogen       -0.222086   0.373396  -0.595 0.552140
## NumOxygen         -0.300338   0.424632  -0.707 0.479563
## NumSulfer          0.621244   0.298101   2.084 0.037432 *
## NumChlorine       -0.374042   0.061636  -6.069 1.87e-09 ***
## NumHalogen        -1.579937   0.459350  -3.440 0.000609 ***
## NumRings                 NA         NA      NA       NA
## HydrophilicFactor  0.162663   0.073229   2.221 0.026570 *
## SurfaceArea1       0.047692   0.013827   3.449 0.000587 ***
## SurfaceArea2      -0.070007   0.013245  -5.285 1.56e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9044 on 933 degrees of freedom
## Multiple R-squared:  0.8082, Adjusted R-squared:  0.8047
## F-statistic: 231.3 on 17 and 933 DF,  p-value: < 2.2e-16
```

NAs in the table mean we have a perfect linear relationship among some of the predictors

- As a result, R dropped the linearly related variables.
- For better clarity, we should refit the model without them

Model Summary

```
##
## Call:
## lm(formula = Solubility ~ . - NumNonHBonds - NumHydrogen - NumRings,
##     data = solTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8499 -0.5963  0.0232  0.5842  2.7848
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.344876   0.149393   2.309 0.021189 *
## MolWeight         -0.008074   0.001325  -6.093 1.61e-09 ***
## NumAtoms           0.275577   0.086182   3.198 0.001432 **
## NumNonHAtoms       1.536062   0.450948   3.406 0.000687 ***
## NumBonds          -0.612747   0.127856  -4.792 1.92e-06 ***
## NumMultBonds      -1.694110   0.321514  -5.269 1.70e-07 ***
## NumRotBonds       -0.147637   0.026894  -5.490 5.19e-08 ***
## NumDblBonds        0.771793   0.234853   3.286 0.001053 **
## NumAromaticBonds   1.278539   0.277614   4.605 4.69e-06 ***
## NumCarbon         -0.650678   0.331825  -1.961 0.050187 .
## NumNitrogen       -0.222086   0.373396  -0.595 0.552140
## NumOxygen         -0.300338   0.424632  -0.707 0.479563
## NumSulfer          0.621244   0.298101   2.084 0.037432 *
## NumChlorine       -0.374042   0.061636  -6.069 1.87e-09 ***
## NumHalogen        -1.579937   0.459350  -3.440 0.000609 ***
## HydrophilicFactor  0.162663   0.073229   2.221 0.026570 *
## SurfaceArea1       0.047692   0.013827   3.449 0.000587 ***
## SurfaceArea2      -0.070007   0.013245  -5.285 1.56e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9044 on 933 degrees of freedom
## Multiple R-squared:  0.8082, Adjusted R-squared:  0.8047
## F-statistic: 231.3 on 17 and 933 DF,  p-value: < 2.2e-16
```

Model Summary

```
##
## Call:
## lm(formula = Solubility ~ . - NumNonHBonds - NumHydrogen - NumRings,
##     data = solTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8499 -0.5963  0.0232  0.5842  2.7848
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.344876   0.149393   2.309 0.021189 *
## MolWeight         -0.008074   0.001325  -6.093 1.61e-09 ***
## NumAtoms           0.275577   0.086182   3.198 0.001432 **
## NumNonHAtoms       1.536062   0.450948   3.406 0.000687 ***
## NumBonds          -0.612747   0.127856  -4.792 1.92e-06 ***
## NumMultBonds      -1.694110   0.321514  -5.269 1.70e-07 ***
## NumRotBonds       -0.147637   0.026894  -5.490 5.19e-08 ***
## NumDblBonds        0.771793   0.234853   3.286 0.001053 **
## NumAromaticBonds   1.278539   0.277614   4.605 4.69e-06 ***
## NumCarbon         -0.650678   0.331825  -1.961 0.050187 .
## NumNitrogen       -0.222086   0.373396  -0.595 0.552140
## NumOxygen         -0.300338   0.424632  -0.707 0.479563
## NumSulfer          0.621244   0.298101   2.084 0.037432 *
## NumChlorine       -0.374042   0.061636  -6.069 1.87e-09 ***
## NumHalogen        -1.579937   0.459350  -3.440 0.000609 ***
## HydrophilicFactor  0.162663   0.073229   2.221 0.026570 *
## SurfaceArea1       0.047692   0.013827   3.449 0.000587 ***
## SurfaceArea2      -0.070007   0.013245  -5.285 1.56e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9044 on 933 degrees of freedom
## Multiple R-squared:  0.8082, Adjusted R-squared:  0.8047
## F-statistic: 231.3 on 17 and 933 DF,  p-value: < 2.2e-16
```

Model Summary

```
##
## Call:
## lm(formula = Solubility ~ . - NumNonHBonds - NumHydrogen - NumRings,
##     data = solTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8499 -0.5963  0.0232  0.5842  2.7848
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.344876   0.149393   2.309 0.021189 *
## MolWeight         -0.008074   0.001325  -6.093 1.61e-09 ***
## NumAtoms           0.275577   0.086182   3.198 0.001432 **
## NumNonHAtoms       1.536062   0.450948   3.406 0.000687 ***
## NumBonds          -0.612747   0.127856  -4.792 1.92e-06 ***
## NumMultBonds      -1.694110   0.321514  -5.269 1.70e-07 ***
## NumRotBonds       -0.147637   0.026894  -5.490 5.19e-08 ***
## NumDblBonds        0.771793   0.234853   3.286 0.001053 **
## NumAromaticBonds   1.278539   0.277614   4.605 4.69e-06 ***
## NumCarbon         -0.650678   0.331825  -1.961 0.050187 .
## NumNitrogen       -0.222086   0.373396  -0.595 0.552140
## NumOxygen         -0.300338   0.424632  -0.707 0.479563
## NumSulfer          0.621244   0.298101   2.084 0.037432 *
## NumChlorine       -0.374042   0.061636  -6.069 1.87e-09 ***
## NumHalogen        -1.579937   0.459350  -3.440 0.000609 ***
## HydrophilicFactor  0.162663   0.073229   2.221 0.026570 *
## SurfaceArea1       0.047692   0.013827   3.449 0.000587 ***
## SurfaceArea2      -0.070007   0.013245  -5.285 1.56e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9044 on 933 degrees of freedom
## Multiple R-squared:  0.8082, Adjusted R-squared:  0.8047
## F-statistic: 231.3 on 17 and 933 DF,  p-value: < 2.2e-16
```

- None of the estimates or p-values changed after refitting the model

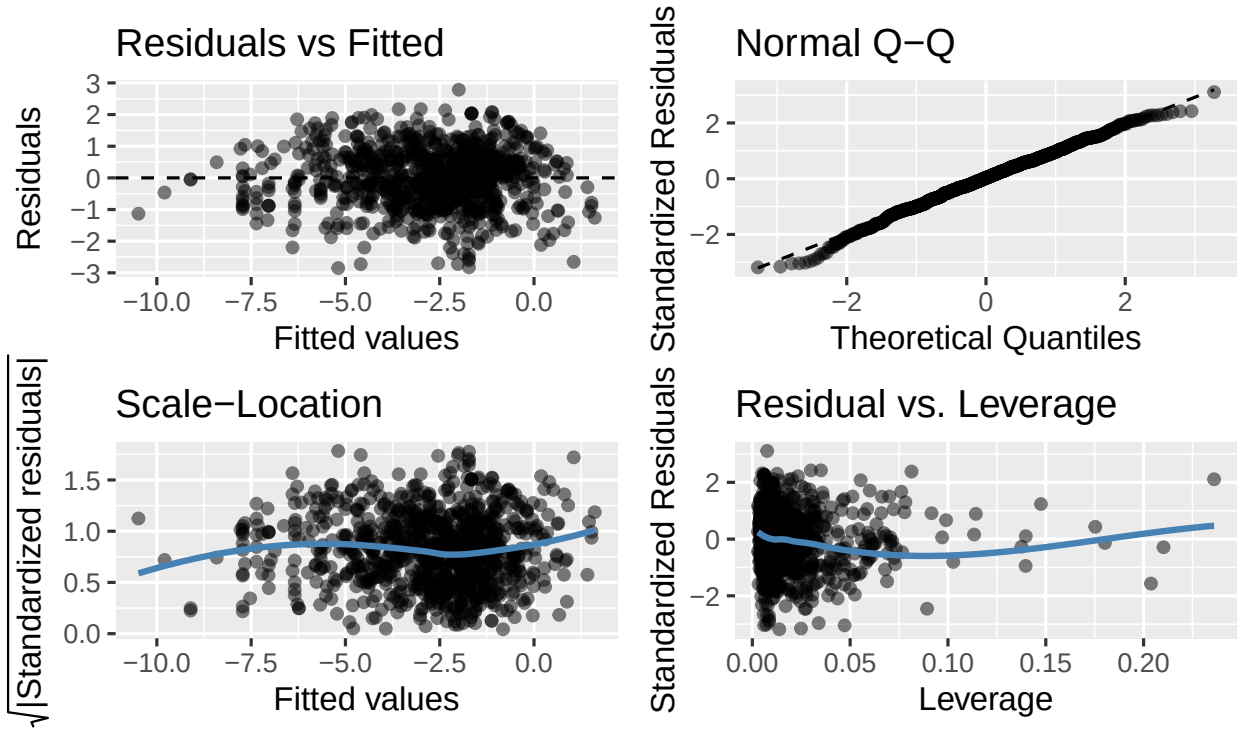
Model Summary

```
##
## Call:
## lm(formula = Solubility ~ . - NumNonHBonds - NumHydrogen - NumRings,
##     data = solTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8499 -0.5963  0.0232  0.5842  2.7848
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.344876   0.149393   2.309 0.021189 *
## MolWeight         -0.008074   0.001325  -6.093 1.61e-09 ***
## NumAtoms           0.275577   0.086182   3.198 0.001432 **
## NumNonHAtoms       1.536062   0.450948   3.406 0.000687 ***
## NumBonds          -0.612747   0.127856  -4.792 1.92e-06 ***
## NumMultBonds      -1.694110   0.321514  -5.269 1.70e-07 ***
## NumRotBonds       -0.147637   0.026894  -5.490 5.19e-08 ***
## NumDblBonds        0.771793   0.234853   3.286 0.001053 **
## NumAromaticBonds   1.278539   0.277614   4.605 4.69e-06 ***
## NumCarbon         -0.650678   0.331825  -1.961 0.050187 .
## NumNitrogen       -0.222086   0.373396  -0.595 0.552140
## NumOxygen         -0.300338   0.424632  -0.707 0.479563
## NumSulfer          0.621244   0.298101   2.084 0.037432 *
## NumChlorine       -0.374042   0.061636  -6.069 1.87e-09 ***
## NumHalogen        -1.579937   0.459350  -3.440 0.000609 ***
## HydrophilicFactor  0.162663   0.073229   2.221 0.026570 *
## SurfaceArea1       0.047692   0.013827   3.449 0.000587 ***
## SurfaceArea2      -0.070007   0.013245  -5.285 1.56e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9044 on 933 degrees of freedom
## Multiple R-squared:  0.8082, Adjusted R-squared:  0.8047
## F-statistic: 231.3 on 17 and 933 DF,  p-value: < 2.2e-16
```

- None of the estimates or p-values changed after refitting the model
- However, extraneous rows were removed (and the table now fits on the slide. Yay!)

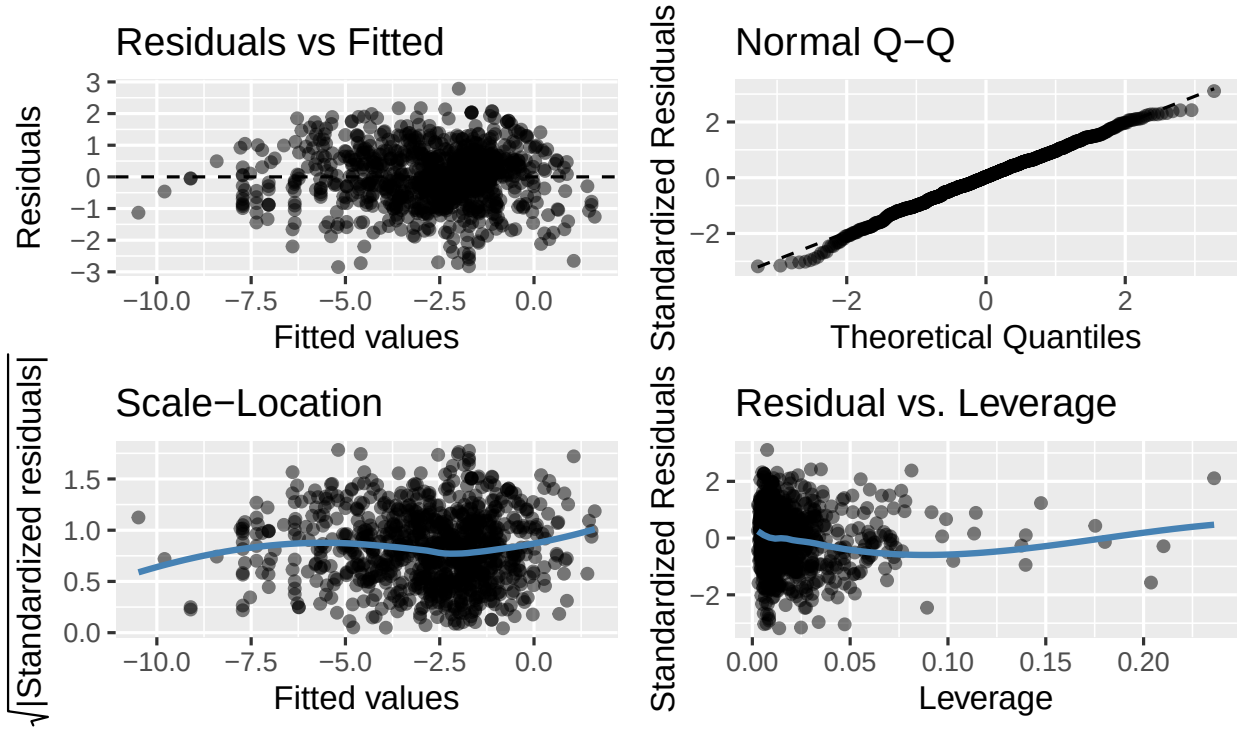
Model Diagnostics

```
library(gglm)
gglm(full_mod)
```



Model Diagnostics

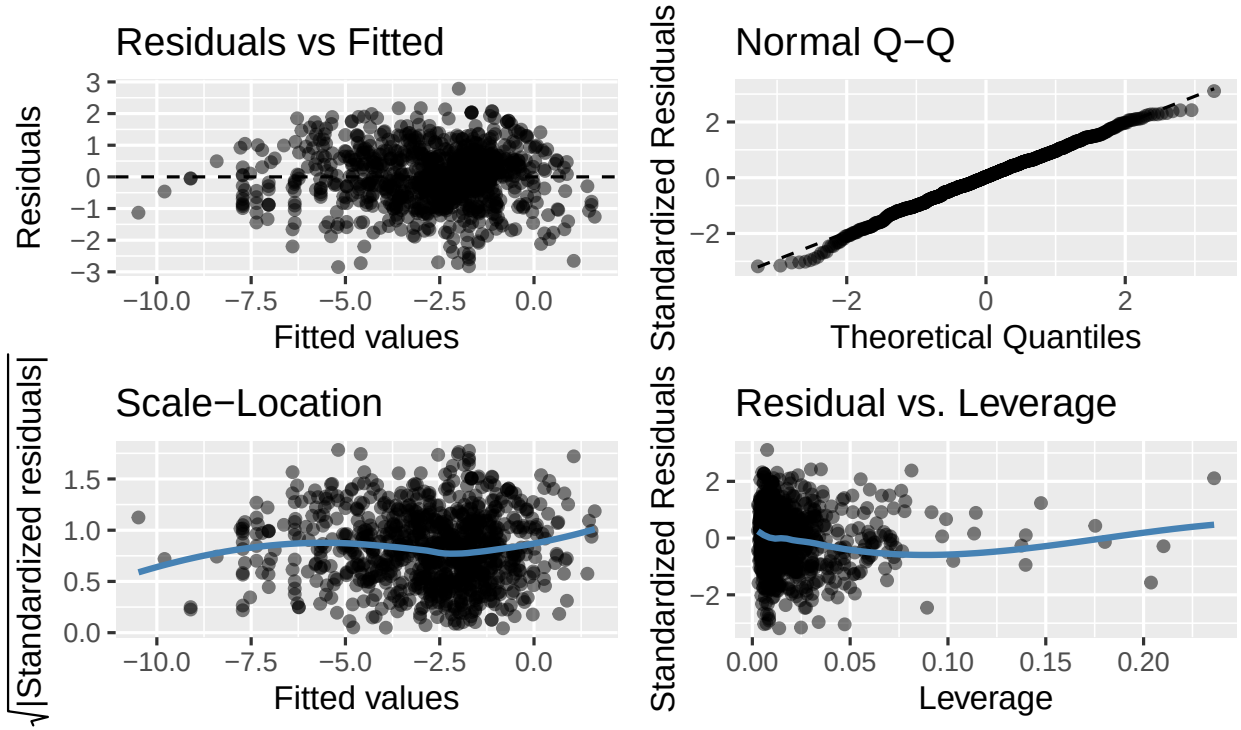
```
library(gglm)
gglm(full_mod)
```



- Diagnostic plots reveal no concerns about model assumptions.

Model Diagnostics

```
library(gglm)
gglm(full_mod)
```



- Diagnostic plots reveal no concerns about model assumptions.
- But we likely still have multicollinearity, and some variables might not be that helpful

Section 2

Subset Selection

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?

First, determine an appropriate selection procedure:

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?

First, determine an appropriate selection procedure:

- **Cross-validation**: Computationally expensive, but also most accurate; requires no model or data assumptions; best overall

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?

First, determine an appropriate selection procedure:

- **Cross-validation**: Computationally expensive, but also most accurate; requires no model or data assumptions; best overall
- **Validation set**: Subject to variability in test/training split; but can be alright for large data sets, or initial exploration

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?

First, determine an appropriate selection procedure:

- **Cross-validation**: Computationally expensive, but also most accurate; requires no model or data assumptions; best overall
- **Validation set**: Subject to variability in test/training split; but can be alright for large data sets, or initial exploration
- **Training set assessment**: using RSS alone on training will lead to overfitting and biased estimate of test MSE;
  - Instead, can apply penalty to RSS based on number of predictors in the model, in order to better estimate test MSE
  - However, can use if validation set is not available, or if large number of models need to be considered.

Training Set Criteria

To compare models on the training set, we can use the following metrics, each of which penalizes the training RSS. Suppose we have d predictors in the model and n observations:

Training Set Criteria

To compare models on the training set, we can use the following metrics, each of which penalizes the training RSS. Suppose we have d predictors in the model and n observations:

- Adjusted R^2: Provides unbiased estimate of population R^2 and is *slightly* smaller than R^2. Best models have largest $textrmadjR^2$

$$\text{adj } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

Training Set Criteria

To compare models on the training set, we can use the following metrics, each of which penalizes the training RSS. Suppose we have d predictors in the model and n observations:

- Adjusted R^2: Provides unbiased estimate of population R^2 and is *slightly* smaller than R^2. Best models have largest $textrmadjR^2$

$$\text{adj } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

- C_p: penalizes training RSS by typical discrepancy between test and training. Best models have smallest C_p

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

Training Set Criteria

To compare models on the training set, we can use the following metrics, each of which penalizes the training RSS. Suppose we have d predictors in the model and n observations:

- Adjusted R^2: Provides unbiased estimate of population R^2 and is *slightly* smaller than R^2. Best models have largest $textrmadjR^2$

$$\text{adj } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

- C_p: penalizes training RSS by typical discrepancy between test and training. Best models have smallest C_p

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

- Akaike information criterion (AIC): uses method of maximum likelihood and information theory. Best models have smallest AIC

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

Training Set Criteria

To compare models on the training set, we can use the following metrics, each of which penalizes the training RSS. Suppose we have d predictors in the model and n observations:

- Adjusted R^2: Provides unbiased estimate of population R^2 and is *slightly* smaller than R^2. Best models have largest $textrmadjR^2$

$$\text{adj } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

- C_p: penalizes training RSS by typical discrepancy between test and training. Best models have smallest C_p

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

- Akaike information criterion (AIC): uses method of maximum likelihood and information theory. Best models have smallest AIC

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

- Bayesian information criterion (BIC): uses likelihood function and Bayesian posteriors. Best models have smallest BIC

$$\text{BIC} = \frac{1}{\quad}(\text{RSS} + \log(n)d \cdot \hat{\sigma}^2)$$

Comparison of Formulas

Criteria Formulas:

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2) \qquad \text{adj } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

$$\text{AIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + 2d \cdot \hat{\sigma}^2) \qquad \text{BIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + \log(n)d \cdot \hat{\sigma}^2)$$

Comparison of Formulas

Criteria Formulas:

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2) \qquad \text{adj } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

$$\text{AIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + 2d \cdot \hat{\sigma}^2) \qquad \text{BIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + \log(n)d \cdot \hat{\sigma}^2)$$

- When n is large relative to d, $R^2 \approx \text{adj } R^2$. Since R^2 overfits models, $\text{adj } R^2$ will also tend to overfit, and so shouldn't be used.

Comparison of Formulas

Criteria Formulas:

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2) \qquad \text{adj } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

$$\text{AIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + 2d \cdot \hat{\sigma}^2) \qquad \text{BIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + \log(n)d \cdot \hat{\sigma}^2)$$

- When n is large relative to d, $R^2 \approx \text{adj } R^2$. Since R^2 overfits models, $\text{adj } R^2$ will also tend to overfit, and so shouldn't be used.
- Other than a multiplicative constant (that doesn't depend on the model), C_p and AIC are equal, and so will always preference the same models

Comparison of Formulas

Criteria Formulas:

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2) \qquad \text{adj } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

$$\text{AIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + 2d \cdot \hat{\sigma}^2) \qquad \text{BIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + \log(n)d \cdot \hat{\sigma}^2)$$

- When n is large relative to d, $R^2 \approx \text{adj } R^2$. Since R^2 overfits models, $\text{adj } R^2$ will also tend to overfit, and so shouldn't be used.
- Other than a multiplicative constant (that doesn't depend on the model), C_p and AIC are equal, and so will always preference the same models
- Both AIC and BIC apply a penalty for adding additional predictors; since $\log(n) > 2$, this penalty is greater for BIC than AIC; hence, BIC will select models with fewer variables.

Comparison of Formulas

Criteria Formulas:

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2) \qquad \text{adj } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

$$\text{AIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + 2d \cdot \hat{\sigma}^2) \qquad \text{BIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + \log(n)d \cdot \hat{\sigma}^2)$$

- When n is large relative to d, $R^2 \approx \text{adj } R^2$. Since R^2 overfits models, $\text{adj } R^2$ will also tend to overfit, and so shouldn't be used.
- Other than a multiplicative constant (that doesn't depend on the model), C_p and AIC are equal, and so will always preference the same models
- Both AIC and BIC apply a penalty for adding additional predictors; since $\log(n) > 2$, this penalty is greater for BIC than AIC; hence, BIC will select models with fewer variables.
- When comparing models with the **same** number of variables, differences in these criteria values will only depend on differences in RSS.

Comparison of Formulas

Criteria Formulas:

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2) \qquad \text{adj } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

$$\text{AIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + 2d \cdot \hat{\sigma}^2) \qquad \text{BIC} = \frac{1}{n\hat{\sigma}^2}(\text{RSS} + \log(n)d \cdot \hat{\sigma}^2)$$

- When n is large relative to d, $R^2 \approx \text{adj } R^2$. Since R^2 overfits models, $\text{adj } R^2$ will also tend to overfit, and so shouldn't be used.
- Other than a multiplicative constant (that doesn't depend on the model), C_p and AIC are equal, and so will always preference the same models
- Both AIC and BIC apply a penalty for adding additional predictors; since $\log(n) > 2$, this penalty is greater for BIC than AIC; hence, BIC will select models with fewer variables.
- When comparing models with the **same** number of variables, differences in these criteria values will only depend on differences in RSS.
  - Hence, for *fixed* number of variables, we can choose the model that has the **smallest** RSS.

Best Subset

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

- There are $\binom{p}{d} = \frac{p!}{d!(p-d)!}$ models using exactly d of p predictors

Best Subset

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

- There are $\binom{p}{d} = \frac{p!}{d!(p-d)!}$ models using exactly d of p predictors

Theoretically, we can find the best model by fitting each possible model and selecting the best via appropriate selection criteria (AIC, BIC, $\text{adj}R^2$, CV)

Best Subset

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

- There are $\binom{p}{d} = \frac{p!}{d!(p-d)!}$ models using exactly d of p predictors

Theoretically, we can find the best model by fitting each possible model and selecting the best via appropriate selection criteria (AIC, BIC, $\text{adj}R^2$, CV)

Downsides?

Best Subset

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

- There are $\binom{p}{d} = \frac{p!}{d!(p-d)!}$ models using exactly d of p predictors

Theoretically, we can find the best model by fitting each possible model and selecting the best via appropriate selection criteria (AIC, BIC, $\text{adj}R^2$, 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 $\binom{p}{d} = \frac{p!}{d!(p-d)!}$ models using exactly d of p predictors

Theoretically, we can find the best model by fitting each possible model and selecting the best via appropriate selection criteria (AIC, BIC, $\mathrm{adj}R^2$, CV)

Downsides?

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

Best Subset

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

- There are $\binom{p}{d} = \frac{p!}{d!(p-d)!}$ models using exactly d of p predictors

Theoretically, we can find the best model by fitting each possible model and selecting the best via appropriate selection criteria (AIC, BIC, $\text{adj}R^2$, CV)

Downsides?

- Computation time and storage grows exponentially in p
- May have low marginal improvement despite number of models fitted
- We are performing a large number of *tests*, which corresponds to a relatively flexible model. Likely to overfit.

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, across the range supplied

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, across the range supplied
- 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, across the range supplied
- Be default, `regsubsets` only returns up to the best eight models. But `nvmax` can be used to return as many variables as desired
- The best model for each fixed number of predictors 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, across the range supplied
- Be default, `regsubsets` only returns up to the best eight models. But `nvmax` can be used to return as many variables as desired
- The best model for each fixed number of predictors is determined by *RSS*
- The `regsubsets` function returns *RSS*, $\text{adj}R^2$, C_p, *BIC* for the best model of each number of predicts.

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, across the range supplied
- Be default, `regsubsets` only returns up to the best eight models. But `nvmax` can be used to return as many variables as desired
- The best model for each fixed number of predictors is determined by *RSS*
- The `regsubsets` function returns *RSS*, $\text{adj}R^2$, C_p, *BIC* for the best model of each number of predicts.
- The **overall** best model can be selected using any of these criteria.

Using regsubsets

```
library(leaps)
best_subset<-regsubsets(Solubility~.-NumNonHBonds -NumHydrogen -NumRings,
                        data = solTrain, nvmax = 17)
```

Using regsubsets

```
library(leaps)
best_subset<-regsubsets(Solubility~.-NumNonHBonds -NumHydrogen -NumRings,
                        data = solTrain, nvmax = 17)
```

- The regsubsets function itself outputs a special regsubsets object, which contains data but is not user-accessible.

Using regsubsets

```
library(leaps)
best_subset<-regsubsets(Solubility~.-NumNonHBonds -NumHydrogen -NumRings,
                        data = solTrain, nvmax = 17)
```

- The regsubsets function itself outputs a special regsubsets object, which contains data but is not user-accessible.
- We'll use the summary function, which provides the following elements:

Using regsubsets

```
library(leaps)
best_subset<-regsubsets(Solubility~.-NumNonHBonds -NumHydrogen -NumRings,
                        data = solTrain, nvmax = 17)
```

- The regsubsets function itself outputs a special regsubsets object, which contains data but is not user-accessible.
- We'll use the summary function, which provides the following elements:
  - which: a list of which predictors are in each model
  - outmat: a version of which for printing
  - Several metrics: rsq, rss, adjr2, cp, bic

Summary of regsubsets

- Stars indicate variable is included in model.
- For readability, I've only shown models with 5 or fewer variables

```
summary(best_subset)$outmat
```

```
##          MolWeight NumAtoms NumNonHAtoms NumBonds NumMultBonds NumRotBonds
## 1  ( 1 ) "*"       " "      " "          " "      " "          " "
## 2  ( 1 ) "*"       " "      " "          " "      " "          " "
## 3  ( 1 ) "*"       " "      " "          " "      "*"          " "
## 4  ( 1 ) " "       " "      "*"          " "      " "          " "
## 5  ( 1 ) " "       " "      "*"          "*"      " "          "*"
##          NumDblBonds NumAromaticBonds NumCarbon NumNitrogen NumOxygen NumSulfer
## 1  ( 1 ) " "         " "              " "       " "         " "       " "
## 2  ( 1 ) " "         " "              " "       " "         " "       " "
## 3  ( 1 ) " "         " "              " "       " "         " "       " "
## 4  ( 1 ) " "         " "              "*"       "*"         "*"       " "
## 5  ( 1 ) " "         " "              " "       "*"         "*"       " "
##          NumChlorine NumHalogen HydrophilicFactor SurfaceArea1 SurfaceArea2
## 1  ( 1 ) " "         " "        " "               " "         " "
## 2  ( 1 ) " "         " "        " "               "*"         " "
## 3  ( 1 ) " "         " "        " "               "*"         " "
## 4  ( 1 ) " "         " "        " "               " "         " "
## 5  ( 1 ) " "         " "        " "               " "         " "
```

Other Selection Metrics

The summary function can return selection metrics for each model.

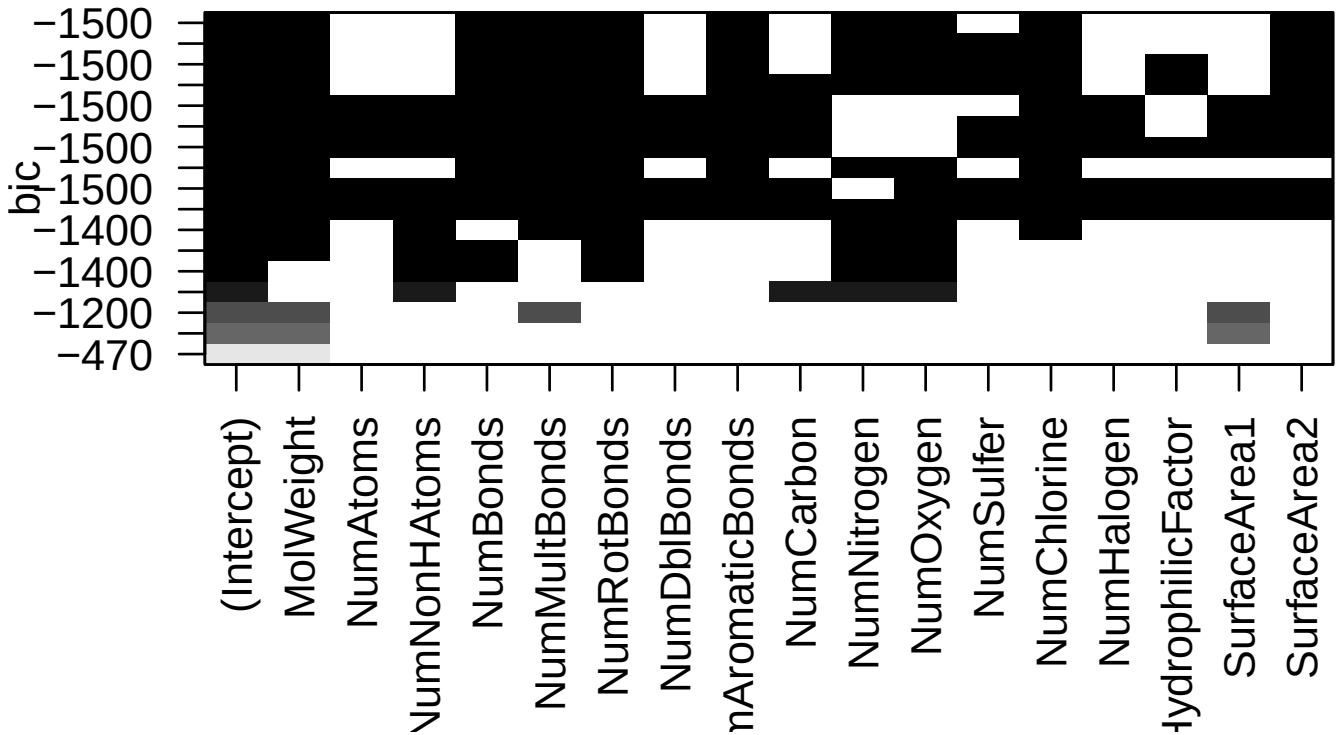
```
d <- data.frame(model = 1:17,
  adjr2 = summary(best_subset)$adjr2,
  rss = summary(best_subset)$rss,
  cp = summary(best_subset)$cp,
  bic = summary(best_subset)$bic)
d %>% head()
```

```
##   model     adjr2       rss        cp        bic
## 1     1 0.3952106 2404.1073 1992.4929  -465.5206
## 2     2 0.6590876 1353.7381  710.2104 -1004.8309
## 3     3 0.7120856 1142.0806  453.4176 -1159.6606
## 4     4 0.7447217 1011.5526  295.8216 -1268.2214
## 5     5 0.7742668  893.5334  153.5199 -1379.3431
## 6     6 0.7813296  864.6602  120.2167 -1403.7232
```

Vizualizing Variables

The variables present can also be plotted directly using `plot`:

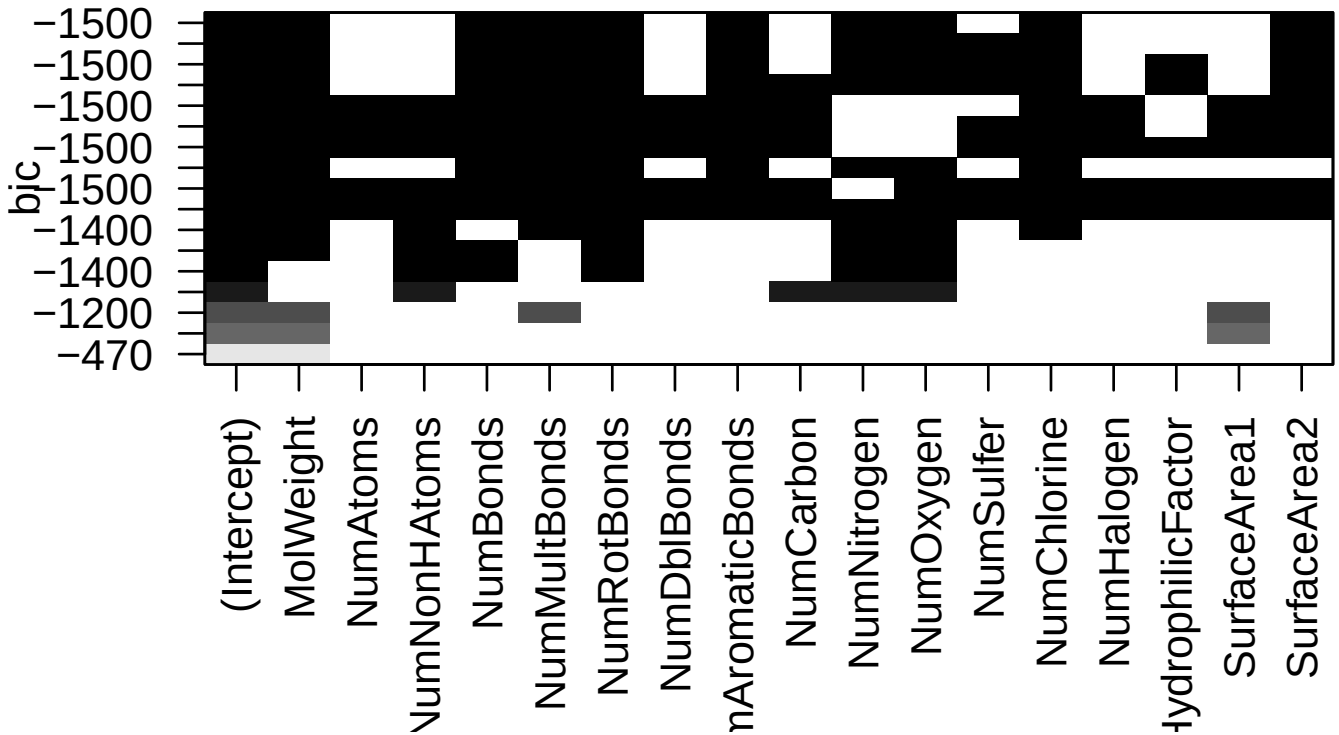
```
plot(best_subset, scale = "bic")
```



Vizualizing Variables

The variables present can also be plotted directly using `plot`:

```
plot(best_subset, scale = "bic")
```

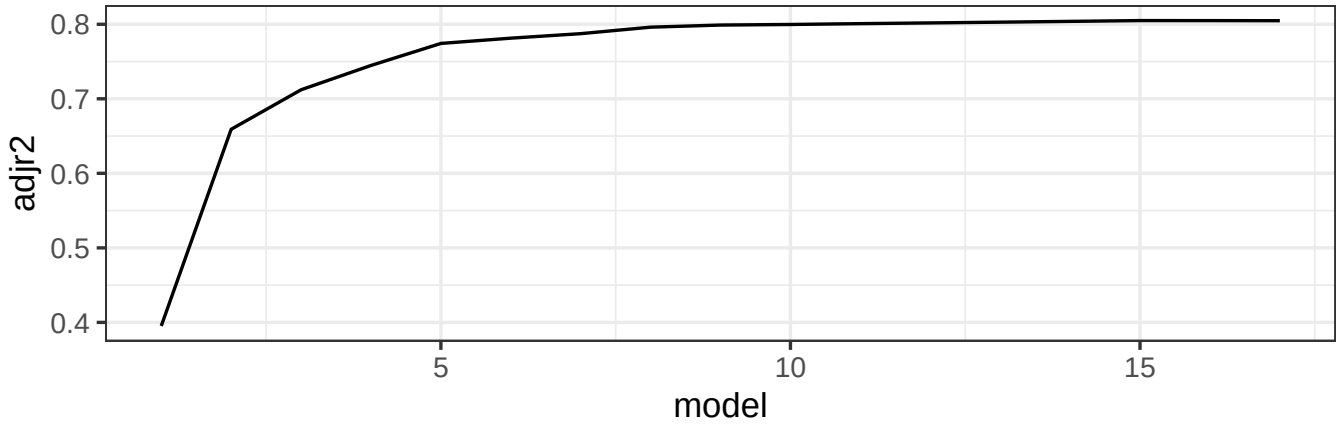


- Models are ordered by values of BIC criteria. Dark rectangles indicate variable is present in the best model for that criteria's value.

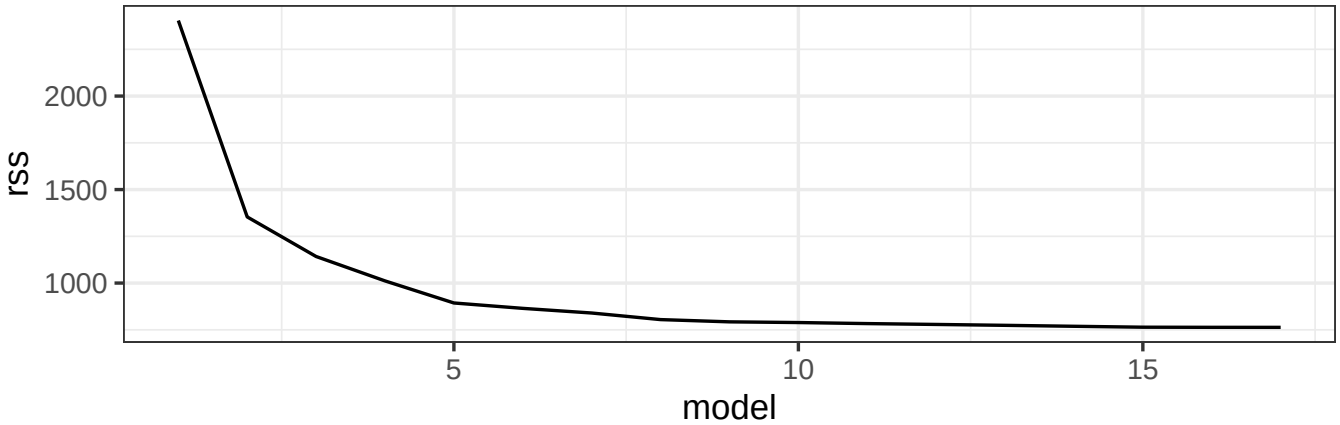
Plotting

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

```
ggplot(d, aes(x = model, y = adjr2))+geom_line()+theme_bw()
```

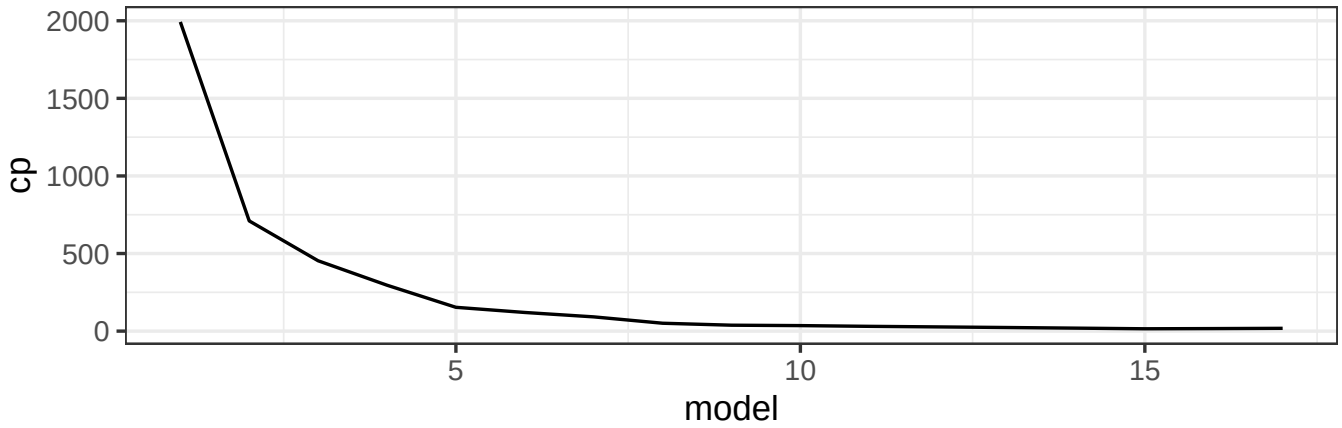



```
ggplot(d, aes(x = model, y = rss))+geom_line()+theme_bw()
```

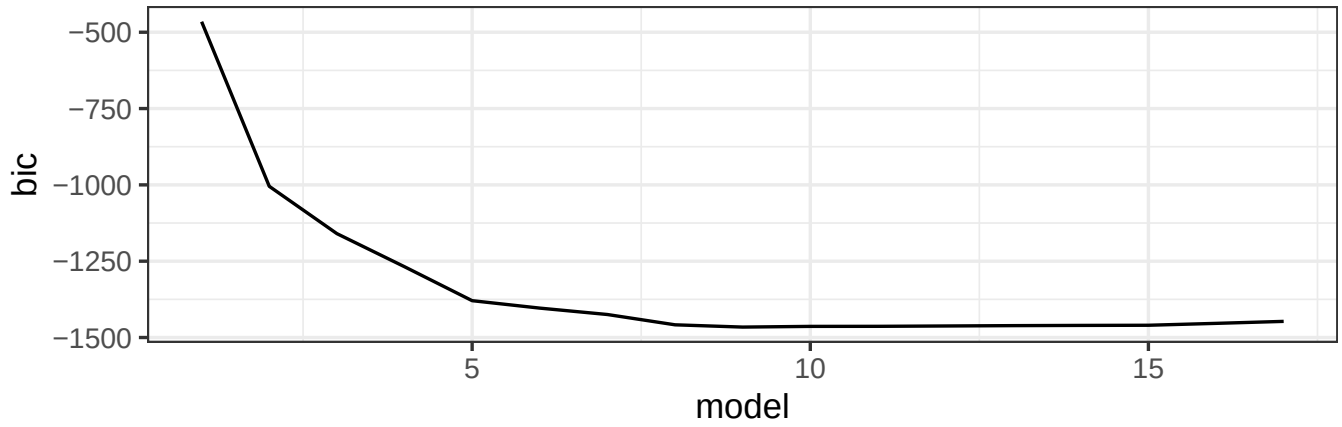


Plotting

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




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



Finding Best Subset

- To calculate the absolute best `cp`, `bic`, etc. we use either the `which.min` or `which.max` function

Finding Best Subset

- To calculate the absolute best cp, bic, etc. we use either the `which.min` or `which.max` function

```
adjr2.max <- which.max(summary(best_subset)$adjr2)
rss.min <- which.min(summary(best_subset)$rss)
cp.min <- which.min(summary(best_subset)$cp)
bic.min <- which.min(summary(best_subset)$bic)
data.frame(adjr2.max, rss.min, cp.min, bic.min)
```

```
##   adjr2.max rss.min cp.min bic.min
## 1        15      17     15       9
```

Finding Best Subset

- To calculate the absolute best cp, bic, etc. we use either the `which.min` or `which.max` function

```
adjr2.max <- which.max(summary(best_subset)$adjr2)
rss.min <- which.min(summary(best_subset)$rss)
cp.min <- which.min(summary(best_subset)$cp)
bic.min <- which.min(summary(best_subset)$bic)
data.frame(adjr2.max, rss.min, cp.min, bic.min)
```

```
##   adjr2.max rss.min cp.min bic.min
## 1        15      17     15       9
```

- So what model is best?

Finding Best Subset

- To calculate the absolute best cp, bic, etc. we use either the `which.min` or `which.max` function

```
adjr2.max <- which.max(summary(best_subset)$adjr2)
rss.min <- which.min(summary(best_subset)$rss)
cp.min <- which.min(summary(best_subset)$cp)
bic.min <- which.min(summary(best_subset)$bic)
data.frame(adjr2.max, rss.min, cp.min, bic.min)
```

```
##   adjr2.max rss.min cp.min bic.min
## 1        15      17     15       9
```

- So what model is best?
  - Usually the simplest model.

Model Coefficients

- To show coefficients associated with the model with lowest `bic`, use `coef`:

```
coef(best_subset, bic.min)
```

```
##      (Intercept)        MolWeight         NumBonds     NumMultBonds
##      0.179049978     -0.007776351     -0.042507435     -0.368292209
##      NumRotBonds NumAromaticBonds      NumNitrogen        NumOxygen
##     -0.138979290      0.225474767      0.628386933      0.782490751
##      NumChlorine     SurfaceArea2
##     -0.386474357     -0.008279467
```

Model Coefficients

- To show coefficients associated with the model with lowest `bic`, use `coef`:

```
coef(best_subset, bic.min)
```

```
##      (Intercept)        MolWeight         NumBonds     NumMultBonds
##      0.179049978     -0.007776351     -0.042507435     -0.368292209
##      NumRotBonds NumAromaticBonds      NumNitrogen        NumOxygen
##     -0.138979290      0.225474767      0.628386933      0.782490751
##      NumChlorine     SurfaceArea2
##     -0.386474357     -0.008279467
```

- And to get a vector of variable names, use `names`:

```
names(coef(best_subset, bic.min))
```

```
##  [1] "(Intercept)"      "MolWeight"        "NumBonds"         "NumMultBonds"
##  [5] "NumRotBonds"      "NumAromaticBonds" "NumNitrogen"      "NumOxygen"
##  [9] "NumChlorine"      "SurfaceArea2"
```

Model Testing

- Let's go with 4 models, based on best subset (since we have it)
  - 5 variables (elbow of bic plot)
  - 9 variables (best bic)
  - 15 variables (best adjusted R^2 and C_p)
  - 17 variables (the full model)
- We'll use 10-fold cross-validation to compare:

```
##      mod5      mod9     mod15     mod17
## 0.9685227 0.9121188 0.8955325 0.8950262
```

- It appears the full-model performed best!

Code for Cross-Validation (Reference)

```
mod5 <-lm(Solubility ~ NumNonHAtoms + NumBonds + NumRotBonds+
            NumNitrogen + NumOxygen,
          data = solTrain)

mod9 <-lm(Solubility ~ MolWeight + NumBonds + NumMultBonds+
            NumRotBonds + NumAromaticBonds + NumNitrogen +
            NumOxygen + NumChlorine + SurfaceArea2,
          data = solTrain)

mod15 <-lm(Solubility ~.
           -NumNonHBonds -NumHydrogen -NumRings - NumNitrogen - NumOxygen,
           data = solTrain)

mod17 <-lm(Solubility~.
           -NumNonHBonds -NumHydrogen -NumRings  ,
           data = solTrain)
```

Code for Cross-Validation (Reference)

```
set.seed(100)
library(rsample)

my_cv <- vfold_cv(solTrain, v = 5, repeats = 10)

get_rmse <- function(split, model){
  train <- analysis(split)
  test <- assessment(split)
  preds <- predict(model, newdata = test)
  obs <- test$Solubility
  rmse <- sqrt(mean((obs-preds)^2))
  rmse
}
```

Code for Cross-Validation (Reference)

```
library(purrr)
my_rmse_df <- data.frame(
  mod5 = map_dbl(my_cv$splits, get_rmse, model = mod5),
  mod9 = map_dbl(my_cv$splits, get_rmse, model = mod9),
  mod15 = map_dbl(my_cv$splits, get_rmse, model = mod15),
  mod17 = map_dbl(my_cv$splits, get_rmse, model = mod17)
)

map_dbl(my_rmse_df, mean)
```

Section 3

Other Selection Algorithms

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 parsimonous 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 parsimonous 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 parsimonous 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 parsimonous 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 parsimonous 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(Solubility~.-NumNonHBonds -NumHydrogen -NumRings,
                           data = solTrain, nvmax = 17, method = "forward")

backward_elim<-regsubsets(Solubility~.-NumNonHBonds -NumHydrogen -NumRings,
                           data = solTrain, nvmax = 17, method = "backward")
```

- All of the same tools used for best subsets are available for forward and backward selection

Comparison of Models