Model Selection

STAT-471/571/701: Modern Data Mining

Contents

Contents

Introduction

Read:

- Chapter 6.1 and 2.2-2.2.2

Objectives:

1. Exploratory Data Analysis (EDA)
    - Abnormality (missing data, outliners, etc.)
    - Pairwise scatter plots
    - Correlation tables
    - Correlation heat-map
    - **Candidate variables and possible new variables**
2. Criterion of accuracy
    - **Prediction Error**
        - Cp: Mallows's Cp
    - **Other targets**
        - AIC: Akaike Information Criterion
        - BIC: Bayesian Information Criterion
3. Model Building
    - All subset
    - Forward Selection
    - Backward Selection
    - Model Diagnostics
4. Findings and Reports
    - Final Report
    - Ending Remarks
5. Summary

Case Study: Performance and Salary in baseball

- We will use ISLR's baseball players' salaries data named `Hitters`
- Hitters: 20 variables about players including performance in 1986 or before (predictors), salaries in 1987 (response), etc.

Goal of the study: * How do players' performance affect their salaries? * We would like to predict a player's salary based on the past performance.

1. Exploratory Data Analysis (EDA)

As good Data Scientists, we must always look at the data to identify potential problems. Here are some main things that you should be in the lookout for:

- Do we have a set of sensible variables?
- Abnormality in the data
- Make a set of candidates to be chosen
- make new variables
- transformations on some variables

Additionally, just by looking at the data, we might gain significant insights for decision making. Finally, it's easier for managers to understand plots than to understand p-values, and doing Exploratory Data Analysis (EDA) is a great way to practice producing meaningful plots!

Take a quick look at the data

Pull out the information about the data: `Hitters` is packaged in `ISLR`

How many players (observations) and variables are contained in the data-set?

```
dim(Hitters)
```

We have 322 players (observations) and 20 variables

Variable names

```
names(Hitters)
```

Structure of the data

```
str(Hitters)  #summary(Hitters) a good way to check abnormality
```

Questions about our variables:

1. Do we have all the key players' features which might affect their salaries?
2. Should we look into average career statistics instead of total numbers?
3. Should we create new variables such as batting average, CHits/CAtBat, or Hits/AtBat?
4. Do career variables include the performance in 86?

For the purpose of illustration of the model selection, we will use the original variables.

Let's see if we have any missing values

```
sum(is.na(Hitters))
```

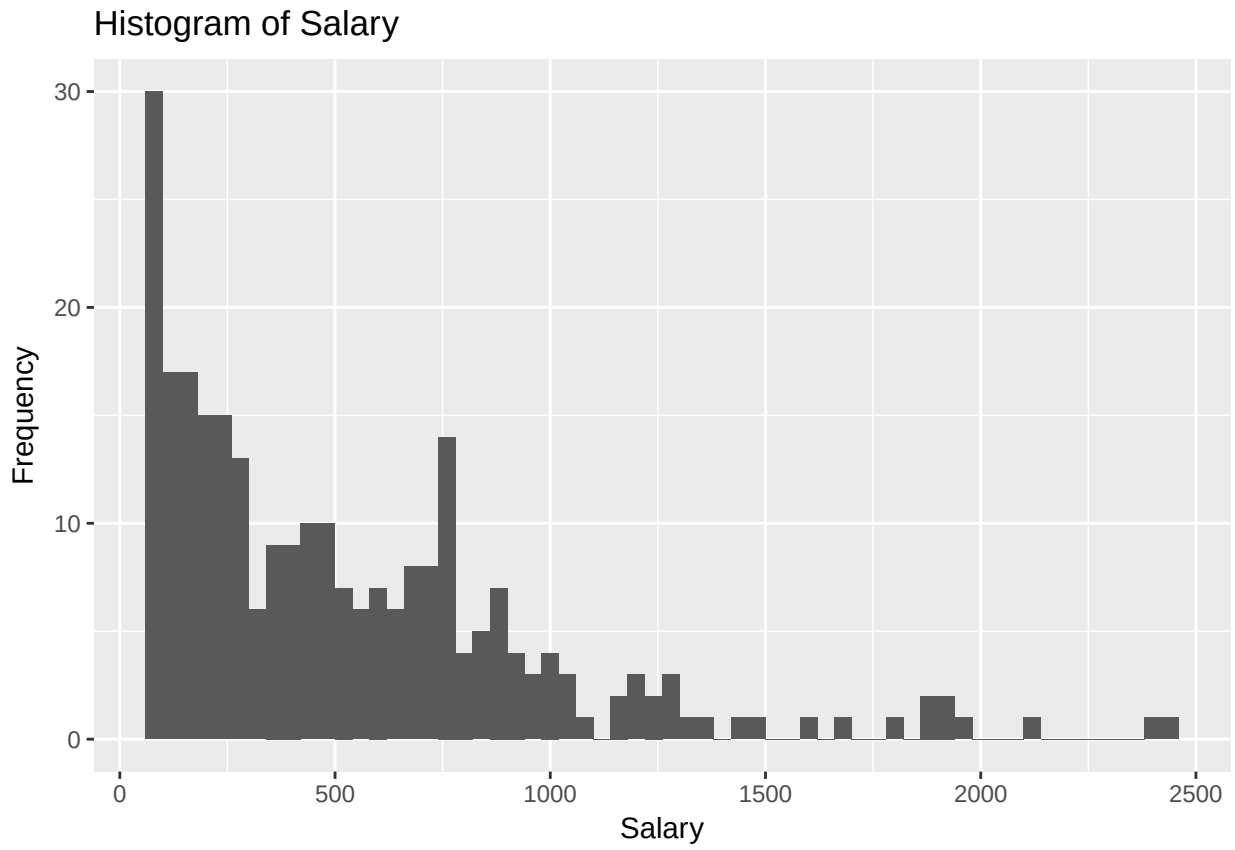
We have 59 missing values. (Note that some missing data might not be coded as NA) **Is there a way to know in which columns these missing values are?**

```r
sapply(Hitters, function(x) any(is.na(x))) # any(is.na(var)) is very useful, it returns T/F for each va
apply(Hitters, 2, function(x) any(is.na(x)))  # apply function by columns: column-wise missing
apply(Hitters, 1, function(x) any(is.na(x)))  # apply function by rows: row-wise missing
```

It looks like all our missing values are for the Salary variable.

```r
sum(is.na(Hitters$Salary))
```

```r
# hist(Hitters$Salary, breaks = 40, col = "blue", main = "Histogram of Salary", ylab = "Frequency")  # 
ggplot(Hitters, aes(x = Salary)) +
  geom_histogram(binwidth = 40) +
  ggtitle("Histogram of Salary") +
  ylab("Frequency")
```



Notice the warning message from ggplot regarding missing data.

Let's see the players with missing salary data

```r
rownames(Hitters)[is.na(Hitters$Salary)]
```

We may want to ask:

1. Do the players with missing salaries have something in common?
2. If possible, we should try to fill in the missing value. We could probably search for the data online.

Let's ignore the players with missing salary data for simplicity's sake. However, this is not usually a good idea, and as data scientists we have to think about why the data is missing. **Can you come up with some reasons?**

```
data.comp <- na.omit(Hitters)
```

We keep 263 players out of the original 322

```
dim(data.comp)
```

Pairwise scatter plots

Let's now take a quick look at the pairwise relationship. We might spot some proper transformations for predicted or predictor variables.

First, we will use dplyr to filter the non-numeric columns. Then we will feed the output into our ggpairs function to produce the paired scatter-plots. **WARNING: Running the pairwise scatter plots will take a while if you have many variables.**

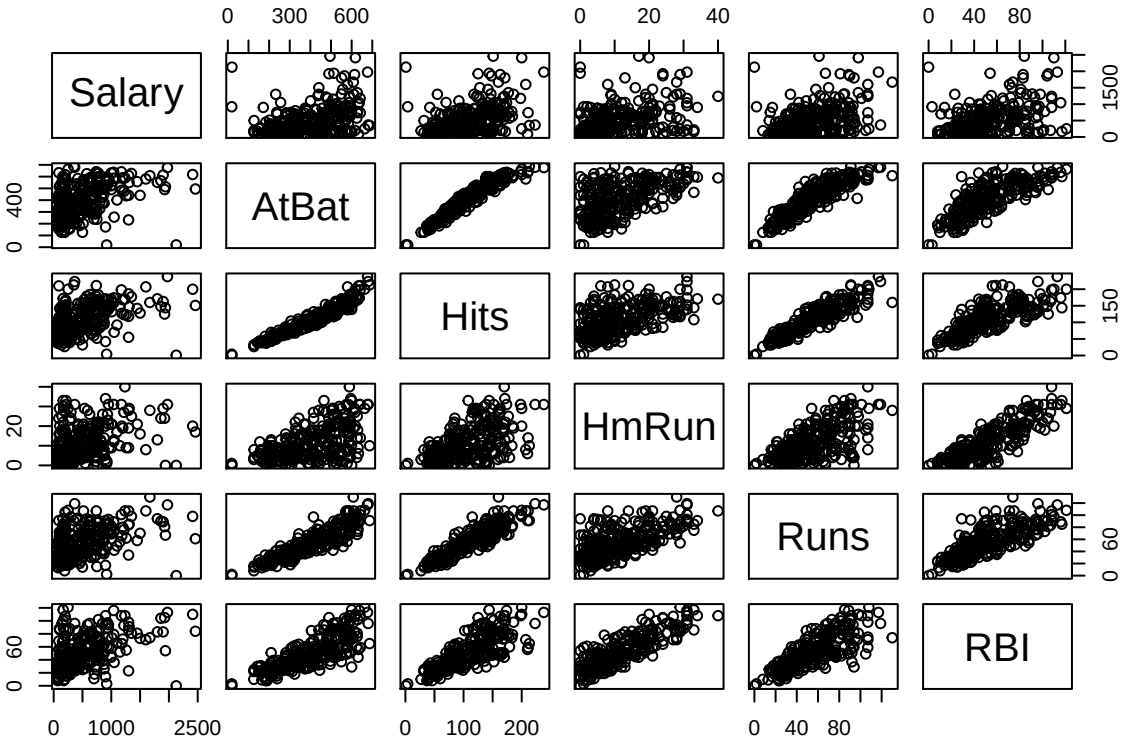
```
data.comp %>%
  select_if(is.numeric) %>%
  ggpairs()
```

Let's save the numeric columns for future use.

```
name.num <- sapply(data.comp, is.numeric)
```

It's hard to recognize anything in this graph! Let's try it with fewer variables.

```
data.comp %>%
  select_if(is.numeric) %>%
  select(Salary, AtBat, Hits, HmRun, Runs, RBI) %>%
  pairs()  # base pair-wise scatter plots
```




```
  #ggpairs()
```

This looks much better, but we can still improve.

Correlation tables

Let's look at the pairwise correlations among all quantitative variables. This will tell us which variable is highly correlated with the response and to each other.

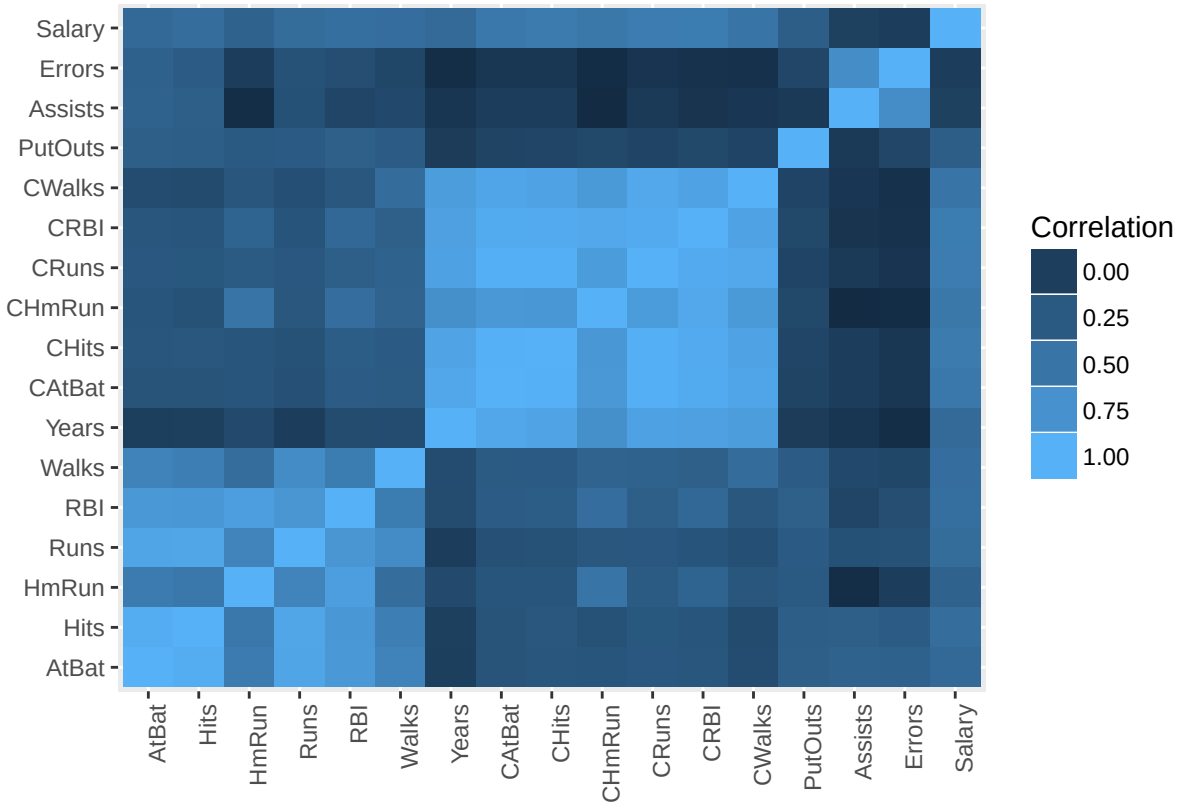
```
cor(data.comp[name.num])  # pairwise cor's among all quantitative var's
```

Correlation heatmap

Once again we can display the correlation table through a heat-map. Let's now use a correlation heat-map.

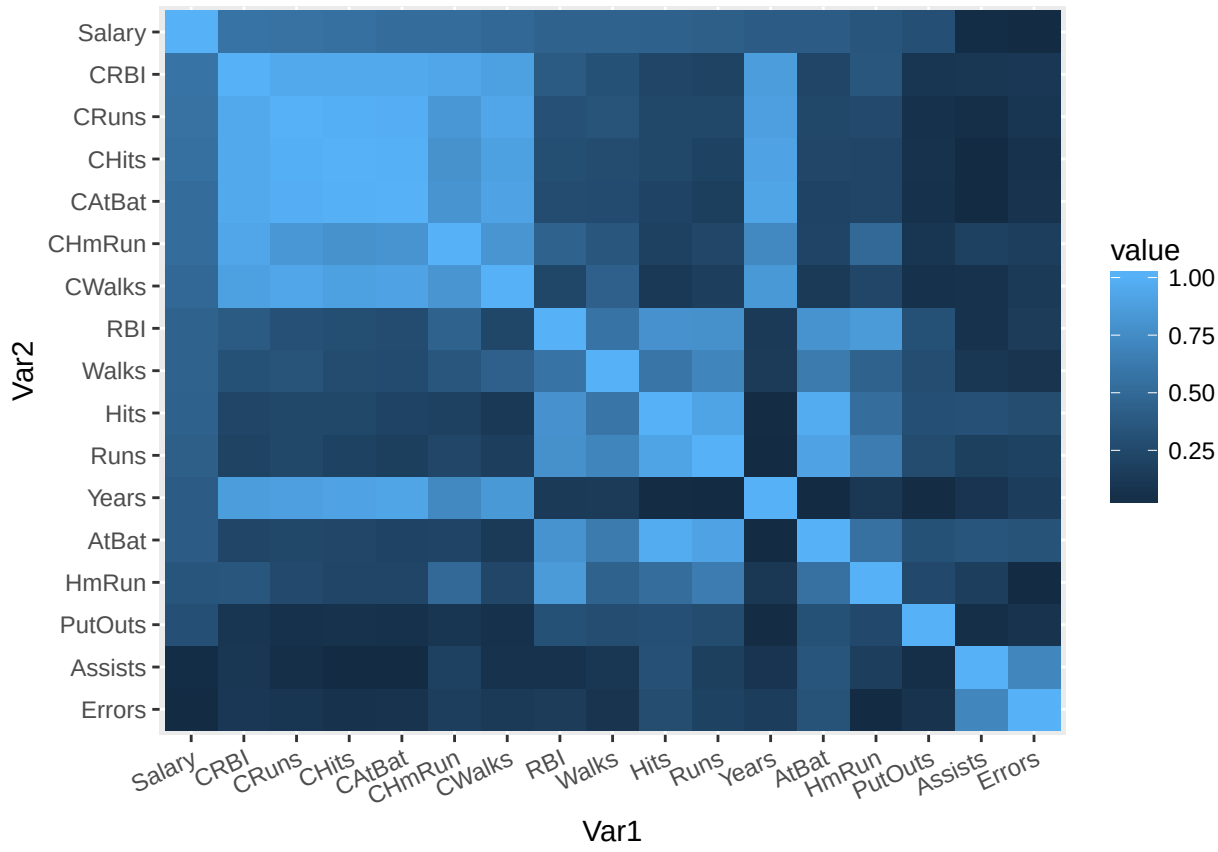
```
data.comp %>%
  select_if(is.numeric) %>%
  qplot(x = Var1,
        y = Var2,
        data = melt(cor(
          data.comp %>%
          select_if(is.numeric))),
        fill = value,
        geom = "tile") +
    xlab("") +
    ylab("") +
```

```
    guides(fill = guide_legend(title = "Correlation")) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



We next rearrange the heat-map by sorting the cor(salary, variables) in a decreasing order

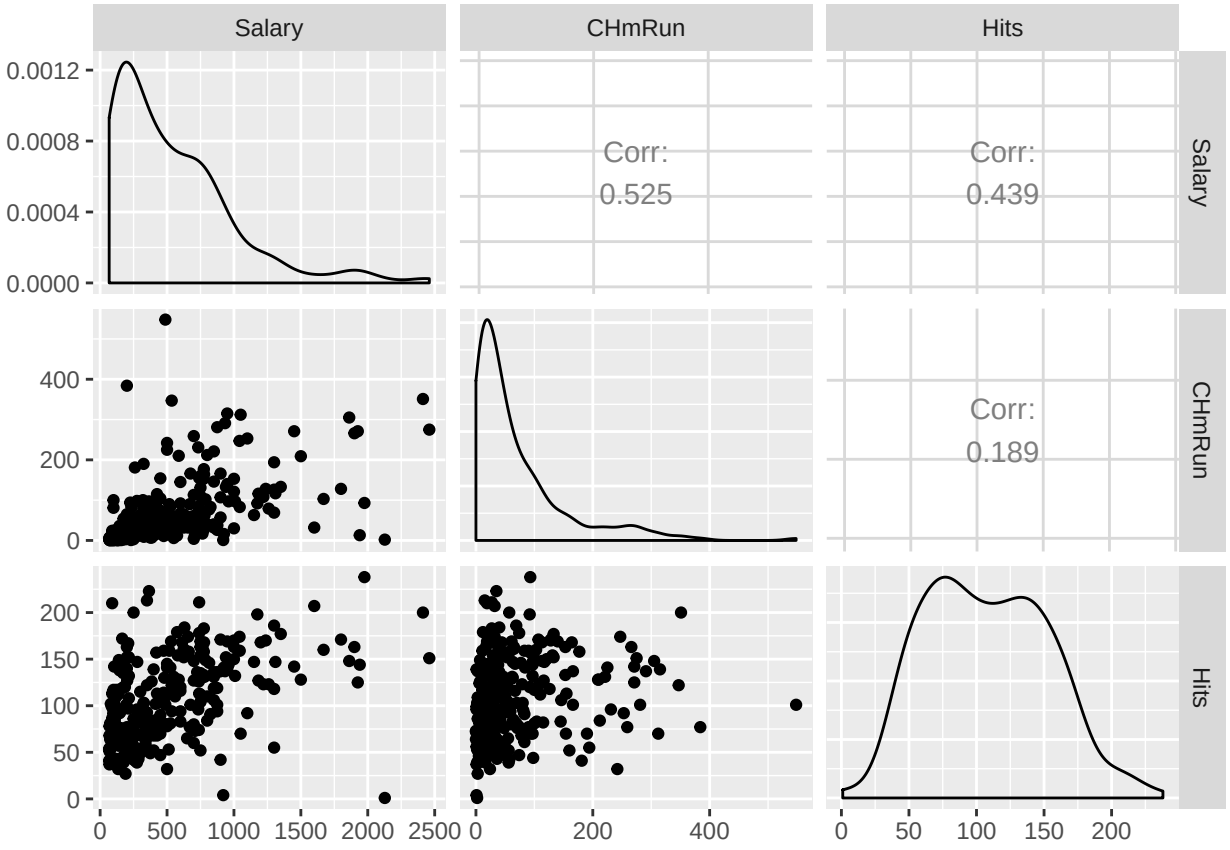
```
# pick the numeric columns
data.comp.numeric <- data.comp %>% select_if(is.numeric)
# correlation table
corr.table <- melt(cor(data.comp.numeric)) %>% mutate(value = abs(value))
# reorder the columns by the abs corr with Salary
corr.table.salary <- corr.table %>% filter(Var2 == "Salary")
col.order <- order(corr.table.salary$value)
data.comp.numeric.2 <- data.comp.numeric[, col.order]

# ordered correlation table
corr.table <- melt(cor(data.comp.numeric.2)) %>% mutate(value = abs(value))

ggplot(corr.table, aes(x=Var1, y=Var2)) +
  geom_tile(aes(fill=value)) +
  scale_x_discrete(limits = rev(levels(corr.table$Var1))) +
  theme(axis.text.x = element_text(angle = 25, hjust = 1))
```



Here we see that career summaries are highly correlated to salaries (first column). We can easily identify variables with high correlation between them. **Can you name a few? Why might this be a problem?**

Now let's zoom in into some scatter plots.

```
data.comp %>%
  select_if(is.numeric) %>%
  select(Salary, CHmRun, Hits) %>%
  ggpairs()
```



Transformations on variables included in the model and possible new variables

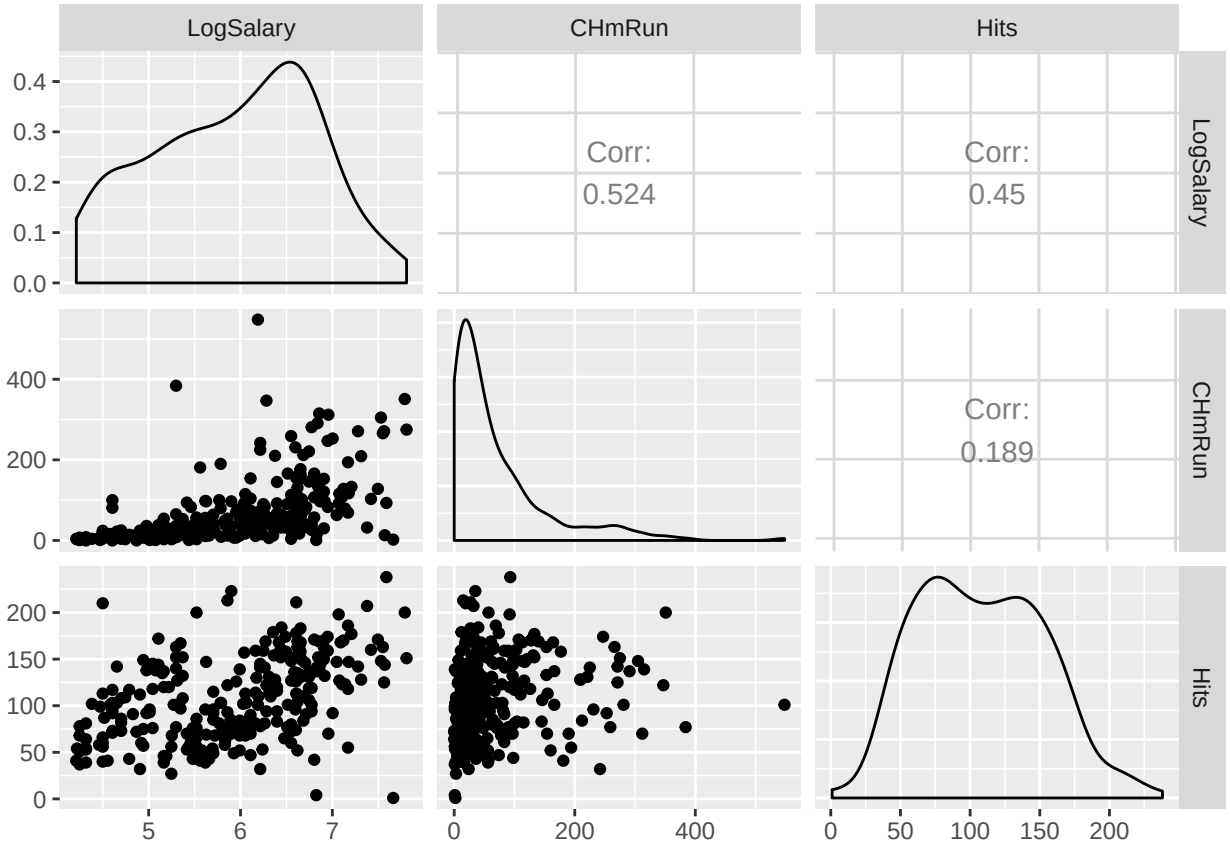
We mainly care about percentage changes in salary, so we will make a log transformation to `Salary` and save it into a new data frame. Why is the difference in log salary equivalent to percentage changes in salaries?

Transform `Salary` to `log(Salary)` and rename the column

```
data1 <- cbind(log(data.comp$Salary), data.comp)
# data1 <- data.frame(log(data.comp$Salary), data.comp) # Another way of doing the same
# data1 <- data.comp %>% mutate(log_salary = log(Salary)) # dplyr solution
names(data1)[1] <- "LogSalary" # Rename it
```

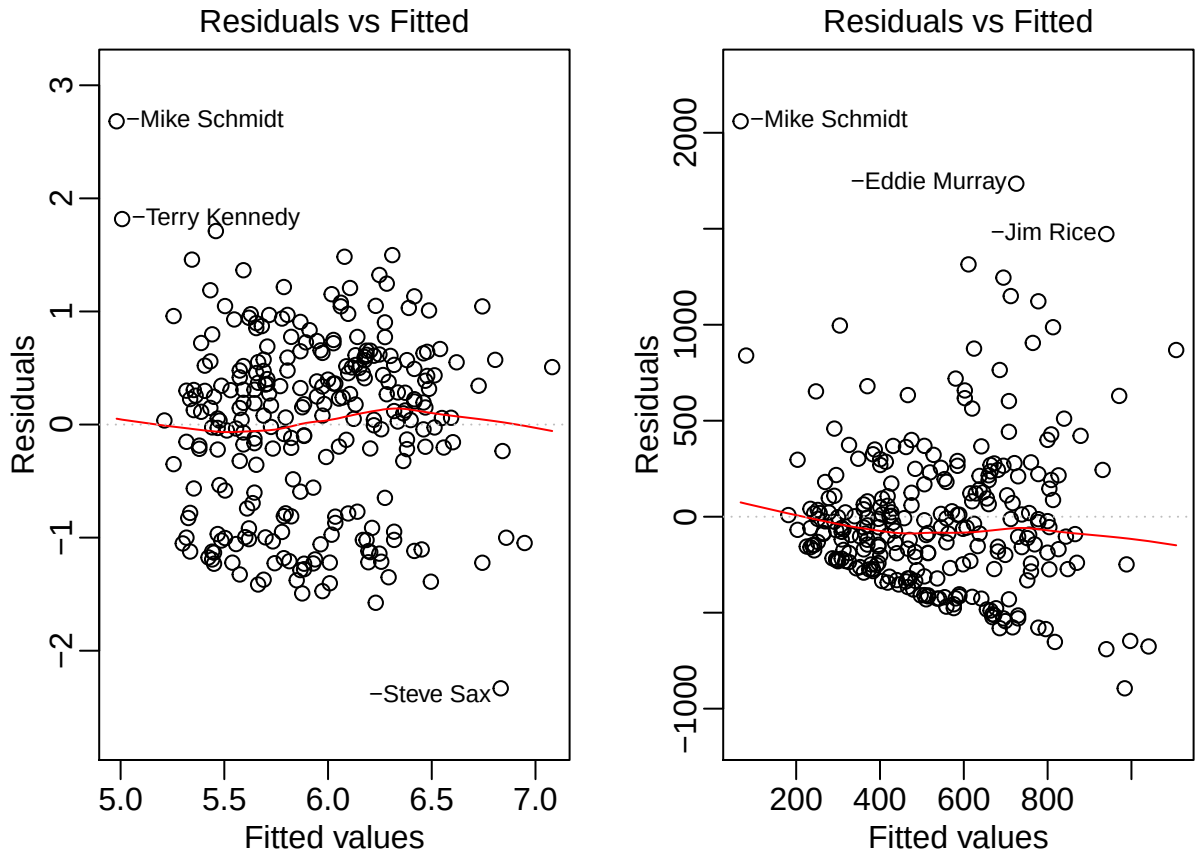
Let's look at some variable transformation candidates.

```
data1 %>%
  select_if(is.numeric) %>%
  select(LogSalary, CHmRun, Hits) %>%
  ggpairs()
```



We examine some residual points for a linear regression to see if there are some outliers, heteroscedasticity, normality etc. Compare the residual plot in which the `Salary` variable isn't transformed.

```
par(mfrow=c(1,2), mar=c(2.5,3,1.5,1), mgp=c(1.5,0.5,0))      # Compare different criteria
plot(lm(LogSalary ~ Hits, data=data1), 1)
plot(lm(Salary ~ Hits, data=data1), 1)
```



Which one is better? Why?

We remove `Salary` from data1 so that we can work more easily.

```
data2 <- data1[, -20]
```

We can now use all variables as predictors.

```
names(data2)
```

It's good practice to save our working data-set to an external file. In this case, we are saving the file as a .csv. **What other formats could you use? How can you save your R session, including variables and formulas?**

```
write.csv(data2, file = "Hitters_comp", row.names = T)
```

2. Criteria

We do not do model selection if we could avoid it. For example if there is solid science behind a model we may only need to estimate some unknow parameters.

Given a set of p predictors, there will be 2^p (In this case there are $2^{19} = 524{,}288$) models.

Goal: build a parsimonious (simple) model which predicts the response as "accurately" as possible. **But how do we define accuracy?**

- If we use RSS as a criterion, then the best model would be using all variables! **Do you remember why?**

```r
data2 <- read.csv(file = "Hitters_comp", row.names = "X")
fit.all <- lm(LogSalary ~., data = data2)
summary(fit.all)
```

Remarks:

1. It is hard to interpret the model due to colinearity
2. However, we can still use this model to do prediction
3. Would the model with all the predictors whose `p-value` $< .05$ be the "best" one?

Answer: Not neccesarily the case!

Criterion of accuracy

Most often a model whose prection error is small tends to also return a set of reasonable variables. By prediction error we mean:

$$\text{Prediction Error} = E(y - \hat{y}_x)^2$$

Which is the sum of prediction errors for all the $x_i's$ in our data.

1. Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

MSE is simple but it does not give us a good estimator for Prediction Error

2. Mallows's (Cp)

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

Note:

1. d: Number of the predictors in the model
2. $\hat{\sigma}^2$: Mean Squared Error (MSE) from the full model

Fact: If the full model is true, C_p is an unbiased estimator of average prediction errors, namely $\frac{1}{n}\sum_{i=1}^{n} E(y_i|x_i - \hat{y}_i)^2$ (differ by some fixed quantity).

3. C_p's might be defined differently, but it will not affect the final model chosen.
4. C_p's will not always decreasing as a function of d!
5. We may choose a model with smallest (or nearlly smallest) C_p's.

Other criteria

1. Akaike Information Criterion (AIC)

$$AIC = 2k - 2ln(\hat{L})$$

- k: Number of estimated parameters
- $\hat{L}$: maximum value of the likelihood function for the model

- It is equivalent to Cp in this set up
- Notion of likelihood function (to be discussed in logistic regressions)

2. Bayesian Information Criterion (BIC)

$$BIC = n \cdot ln\Big(\frac{RSS}{n}\Big) + k \cdot ln(n)$$

- k: Number of free parameters to be estimated
- BIC is derived from Bayesian route
- It tends to output smaller models
- The `leaps` package outputs an alternative form of BIC

3. Model Building

All Subsets

1. Given a model size d, find the one with min RSS, then compute its Cp.
2. Min Cp (or BIC) to obtain the final model or use elbow rule!
3. Elbow: the point where the scree plot is leveling off.
4. Package `leaps`: `regsubsets()` does the job

Regsubsets

Let's now try to select a good model. We will use the library `leaps`, which gives us model selection functions. It will output the best submodel within each model size together with `rsq`, `rss`, `adjr2`, `cp` and `bic`.

All subset selection: for each model size, report the one with the smallest RSS Pro: identify the "best" model. Con: computation expensive. In our example: 2^{19}=524,288 models

```
fit.exh <- regsubsets(LogSalary ~., data2, nvmax=25, method="exhaustive")
```

The default settings:

- nvmax=8
- method=c("exhaustive", "backward", "forward", "seqrep"), "exhaustive" is the default setting
- nbest=1: output candidate models whose RSS's are similar.

```
names(fit.exh)
```

List the model with the smallest RSS among each size of the model

```
summary(fit.exh)
```

```
f.e <- summary(fit.exh)
names(f.e)
```

Let's look at the `which` table

```
f.e$which
```

The row indicates the number of variables in the model with the smallest RSS. For example, if we want the best model with only one variable, then `CRuns` is the best predictors. For two variables, `CRuns` and `CAtBat` are the best predictors.

We can see the value for each selection criterion. For example, let's look at R^2

```
data.frame(variables=(1:length(f.e$rsq)), r_squared=f.e$rsq)
```

Notice that as we increase the number of variables, R^2 increases. Let's now look at all the criterion options.

```
data.frame(variables = (1:length(f.e$rsq)),
           r_squared = f.e$rsq,
           rss = f.e$rss,
           bic = f.e$bic,
           cp = f.e$cp)
```

Regardless which criteria to be used, given a fixed number of predictors, we will have the same set of covariates which achieves the min value of `RSS`. **Can you prove why is this true???**

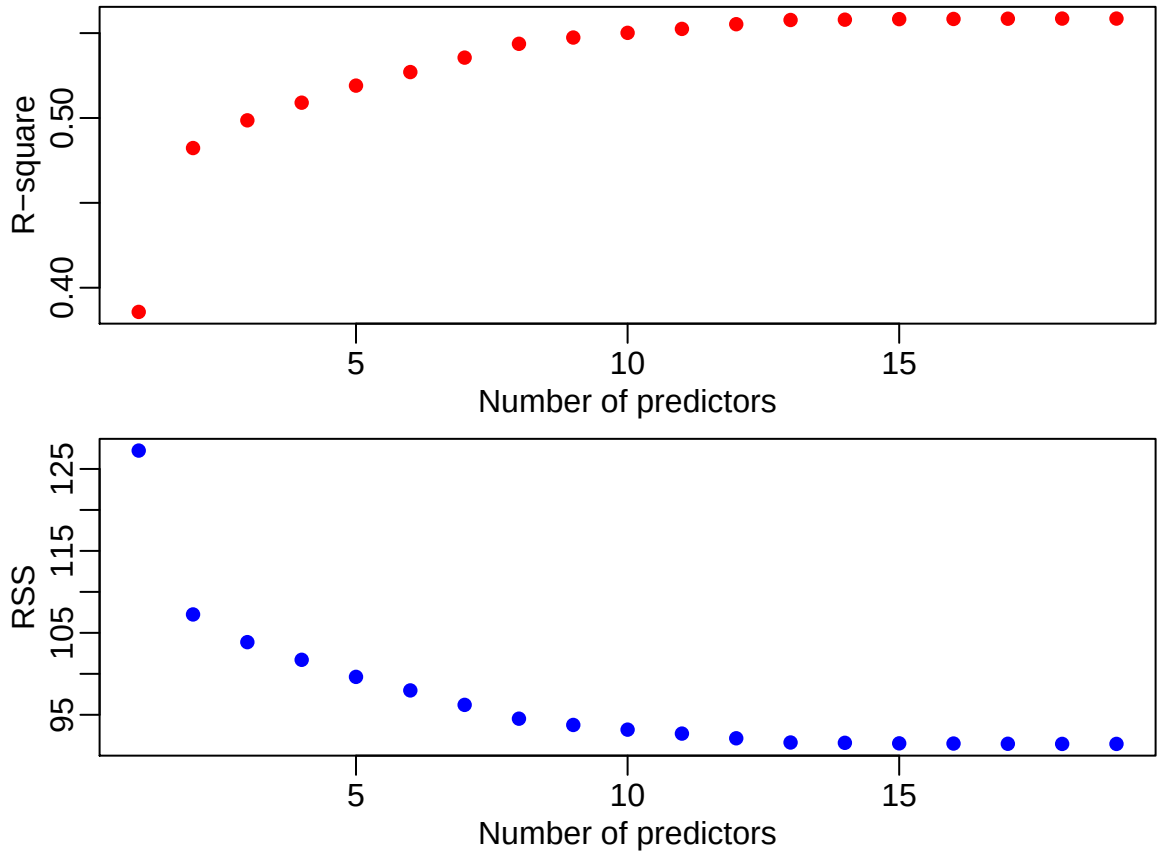
```
coef(fit.exh, 6)
```

```
coef(fit.exh,7)
```

Compare different criterion

Let's compare the different criterion.

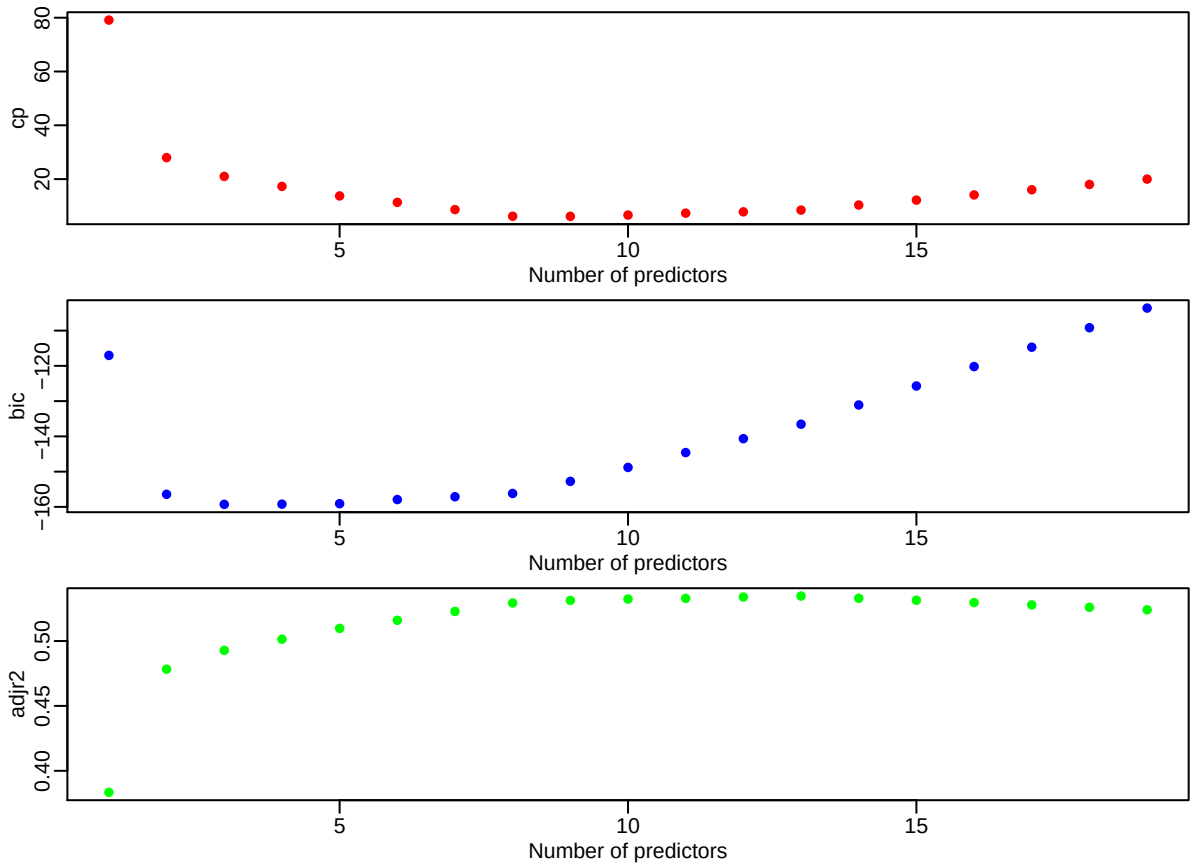
```
par(mfrow=c(2,1), mar=c(2.5,4,0.5,1), mgp=c(1.5,0.5,0))    # Compare different criterions: as expected
plot(f.e$rsq, xlab="Number of predictors", ylab="R-square", col="red", type="p", pch=16)
plot(f.e$rss, xlab="Number of predictors", ylab="RSS", col="blue", type="p", pch=16)
```



As expected R^2 increases as d increases. R^2 or RSS will not be good criteria to use in order to find a model which has the least average prediction squared error. Cp or BIC will be used.

Here are the plots of Cp vs number of predictors. Similarly we have the plots of BIC vs number of the predictors

```
par(mfrow=c(3,1), mar=c(2.5,4,0.5,1), mgp=c(1.5,0.5,0))      # Compare different criteria
plot(f.e$cp, xlab="Number of predictors",
     ylab="cp", col="red", type="p", pch=16)
plot(f.e$bic, xlab="Number of predictors",
     ylab="bic", col="blue", type="p", pch=16)
plot(f.e$adjr2, xlab="Number of predictors",
     ylab="adjr2", col="green", type="p", pch=16)
```



Notice that the final model can be different in terms of the number of predictors depending on which criterion to use. BIC tends to give the model with least number of predictors. In this case we may use five variable models.

Optimal Model by Cp

Let's locate the optimal model size by Cp's

```
opt.size <- which.min(f.e$cp)
opt.size
```

Now we look for the optimal variables selected

```
fit.exh.var <- f.e$which
fit.exh.var[opt.size,]
```

```
colnames(fit.exh.var)[fit.exh.var[opt.size,]]
```

We could choose a smaller model, say size 5 as the final model

```
colnames(fit.exh.var)[fit.exh.var[5,]]
```

```
fit.final <- lm(LogSalary ~ Hits + Walks + Years + CHits + Division, data2)    # Division has two level
# fit.final <- lm(LogSalary ~., data2[f.e$which[5,]])
summary(fit.final)
```

Note: there is no guarantee that all the var's in the final model are significant at $\alpha = .05$ say.

Reminder: `Anova` gives us the test for each var at a time.

```
Anova(fit.final)
```

When `d` is too large or in the situation `d` is even larger than `n`, it is impossible to search all subsets to find the least `RSS` model for each given number of predictors. One possibility is through `Forward Selection`.

Forward Selection

```
fit.forward <- regsubsets(LogSalary ~., data2, nvmax=25, method="forward")
fit.forward
```

```
f.f <- summary(fit.forward)
f.f
```

At any given number, the predictors selected may vary depending on the selection method.

Exhaustive

```
f.e
```

1 Variable: `CRuns` 2 Variables: `Hits`, `CAtBat`

Forward Selection

```
f.f
```

1 Variable: `CRuns` 2 Variables: `Hits`, `CRuns`

Notice that Exhaustive and Forward selection give us different variables when we look for the best 2 variables.

For any fixed number, the model selected from `All Subset Selection` will have larger R^2 (or smaller RSS) than that from `Forward Selection` and **WHY?**

```
plot(f.f$rsq, ylab="rsq", col="red", type="p", pch=16,
     xlab="Forward Selection")
lines(f.e$rsq, ylab="rsq", col="blue", type="p", pch=16,
   xlab="All Subset Selection")
```

If we decided to use a model with 6 predictors by forward selection here it is:

```
coef(fit.forward, 6)
```

```
summary(lm(LogSalary ~ Hits + Walks + Years + CRuns + Division + PutOuts, data2))
```

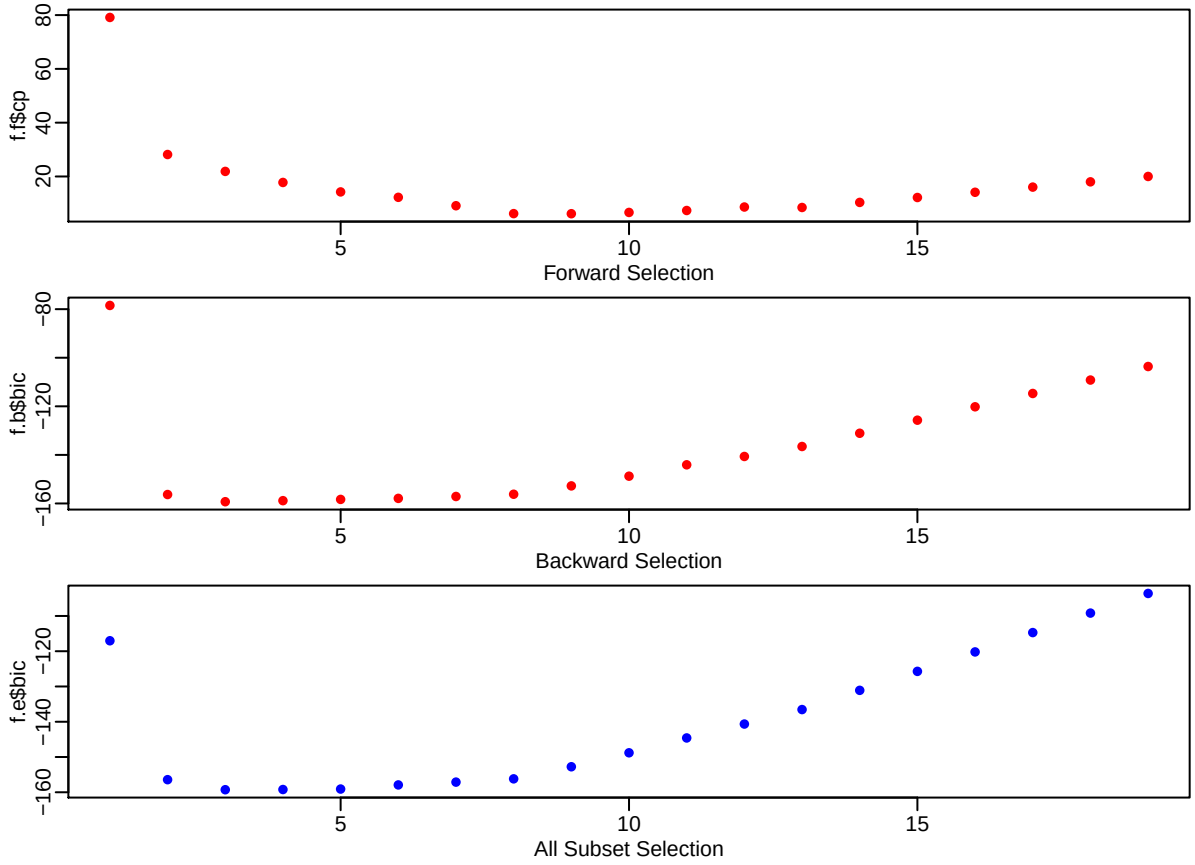
All of the above variables have coefficients different from 0 at .05 significance level.

Backward Selection

Especially useful when `p` is large (still smaller than `n`)!

```
fit.backward <- regsubsets(LogSalary ~., data2, nvmax=19, method="backward")
f.b <- summary(fit.backward)
```

```
par(mfrow=c(3,1), mar=c(2.5,4,0.5,1), mgp=c(1.5,0.5,0))
plot(f.f$cp,  col="red", type="p", pch=16,
   xlab="Forward Selection")
plot(f.b$bic,  col="red", type="p", pch=16,
   xlab="Backward Selection")
plot(f.e$bic,  col="blue", type="p", pch=16,
   xlab="All Subset Selection")
```




```
coef(fit.backward, 6)
```

```
summary(lm(LogSalary ~ AtBat + Hits + Walks + Years + CRuns + PutOuts, data2))
```

```
coef(fit.exh,6)
```

```
summary(lm(LogSalary ~ AtBat + Hits + Walks + Years + CHits + Division, data2))
```

force.in

Sometimes, we want to build model based on some already selected variables.

```
fit.force.in <- regsubsets(LogSalary ~., data2, nvmax=25, method="forward", force.in=c("Hits"))
fit.force.in
```

```
fit.force.in.s <- summary(fit.force.in)
fit.force.in.s
```

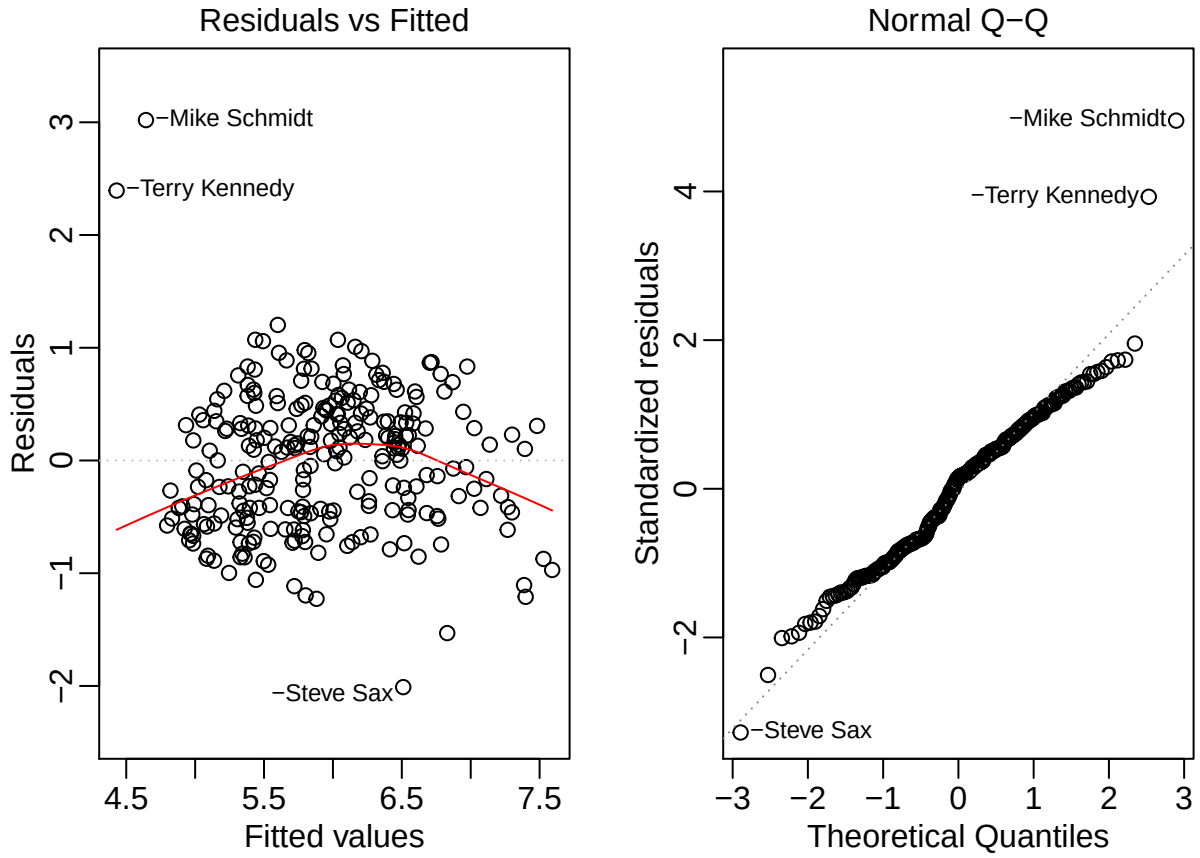
Notice that `Hits` is forced to be in the model now.

Model Diagnostics

Remember: we want to examine the residual plot to look for possible violations of the Regression Model, such as:

- Auto-correlated residuals (violate independence)
- Heteroscedastic residuals (violate equal error variance)
- Patterns (violate normality)

```
par(mfrow=c(1,2), mar=c(2.5,3,1.5,1), mgp=c(1.5,0.5,0))
plot(fit.final,1)
plot(fit.final,2)
```



Everything looks reasonably fine. We mostly care about the first two plots.

4. Findings and Reports

Final Report

1. Collectively the five features: Hits, Walks, Years, CHits, and Division do a good job to predict `log(Salary)`.
2. Interpret each LS estimates of the coefficients
3. We may want to estimate the mean salary or individual salary. For example for one with the following predictor values:

Hits = 75, Walks = 50, Years = 4, CHits = 1200, and Division = E

a) We first make a dataframe:

```
player <- data2[1,] # Get the right format and the variable names
player[1] <- NA
player$Hits <- 75
player$Walks <- 50
player$Years <- 4
player$Chits <- 1200
player$Division <- as.factor("E")    # To make sure it is a dataframe
```

b) Get a 95% CI for the mean salaries for all such players

```
player.m <- predict(fit.final, player, interval="confidence", se.fit=TRUE)
player.m  # in log scale
exp(player.m$fit)  # 319.6469 (250.2053, 408.3612) as the mean salary after transforming back to the or
```

c) Get a 95% Prediction Interval of the salary for the player

```
player.p <- predict(fit.final, player, interval="prediction", se.fit=TRUE)
player.p  # in log scale
exp(player.p$fit) #319.6469 (91.55162, 1116.028)
```

Remarks

1. If you want to only start with a collection of predictors, you may use the usual formula: $y = x_1 + x_2$ inside the regsubsets. For example if you only want to search a best model using AtBat, Hits, Runs and AtBat^2, you may do the following:

```
summary(regsubsets(LogSalary ~ AtBat + Hits + Runs + I(AtBat^2), data2))
```

2. We could also restrict the maximum possible model size so that we only search the best models up to `nvmax`. This is useful when `p` is large!

```
summary(regsubsets(LogSalary ~., nvmax=5, method="exhaus", data2)) #We restrict a model with no more th
```

3. Let's incorporate some other features such as BA = Hits/AtBat

```
data3 <- data2
data3$BA <- data3$Hits/data3$AtBat
mod.new <- regsubsets(LogSalary ~., nvmax = 21, method="exhaus", data3)
mod.new.s <- summary(mod.new)
mod.new.s
```

Any model around 8 features is good.

```
plot(mod.new.s$cp)
```

```
summary(lm(LogSalary ~., data = data3[mod.new.s$which[6, ]])) # can we make sense out of the negative c
```

Can you guess why `BA` has a negative coefficient?

```
summary(lm(LogSalary ~., data = data3[mod.new.s$which[7, ]]))
```

Summary

1. You should always do EDA. By looking at your data, you might be able to identify some candidate variables and to find problems such as colinearity, non-linearity, missing data, and outliners. If you identify any problem, and fix it, your model will improve. `Pairwise scatter plots`, `Correlation heatmaps`, and `Correlation tables` are very useful in EDA.
2. There are different criteria for measuring how good a `fit` is. We recomment to use `Cp` and `BIC`. We will introduce more methods later.
3. When building a model, you can use All Subsets, which searches for all possible combinations. In some cases, it can work, but when you have many variables, is computationally expensive. You have two other options: Forward and Backward Selection. They each have their advantages and disadvantages.
4. After building your model, you should always run model diagnostics to identify issues such as: auto-correlated residuals, violations of normality or equal variance of residuals. If you find any of these problems, it's very likely that you might need to do some variable transformations. Before writing a report, make sure the model makes sense to you.