2 PQHS 471 Notes Week 2

2.1 Week 2 Day 1

ISLR Chapter 3 uses linear regression to illustrate many concepts and issues in regression modeling that are also applicable in other types of regression (e.g., logistic, Poisson, Cox, etc.). Understanding these issues will allow us to see the limitations of various methods and how far we can interpret results.

Seven questions using `Advertising` data as an example; to be answered in ISLR 3.4. All except Q5 are inference questions. Synergy effect (in marketing) is called interaction effect or effect modification in Statistics.

2.1.1 ISLR 3.1: Simple linear regression (a single predictor X)

Many of the concepts in ISLR 3.1 are applicable to multiple linear regression and other regression models. `Advertising` data: `sales` in thousands of units, `TV`/`radio`/`newspaper` in thousands of dollars.

```r
Advertising = read.table("Advertising.csv", header=T, sep=',', row.names=1)
names(Advertising); dim(Advertising)
Advertising$totalcost = with(Advertising, TV + radio + newspaper)
cor(Advertising)
pairs(Advertising)

## nicer plots than pairs()
car::scatterplotMatrix(~ sales + TV + radio + newspaper + totalcost, Advertising)
```

- 3.1.1: **Least squares** is a model fitting criterion. It implicitly assumes (1) equal variance of the residuals across observations and (2) symmetry in the residuals. For the model, sales $= \beta_0 + \beta_1$TV, the equal variance assumption is incorrect, allowing high variance observations on the right to have a high influence on the fitted model. The linearity assumption is also incorrect for low values of `TV`.

In R, a fitted model is a list; `summary()` also returns a list. Use `names()` to learn what a list contains.

```r
fittv = lm(sales ~ TV, data=Advertising)
names(fittv)
sumfittv = summary(fittv)
names(sumfittv)
sumfittv$r.squared
```

Figure 3.2 in the book is incorrect. As shown in its code (https://github.com/JWarmenhoven/ISLR-python/issues/1), a linear model $y = \alpha_0 + \alpha_1 x$ was fit with `y=sales` and `x=TV-mean(TV)` to obtain $\hat{\alpha}_0 = 14.02$ and $\hat{\alpha}_1 = 0.0475$, and the RSS was calculated over a grid of (α_0, α_1) but plotted over a grid of $(\alpha_0 - 0.0475 \cdot \text{meanTV}, \alpha_1)$. The relationship between this model and the intended model, sales $= \beta_0 + \beta_1$TV, is that $\beta_0 = \alpha_0 - \alpha_1 \cdot \text{meanTV}$ and $\beta_1 = \alpha_1$. Thus the grid of (β_0, β_1) is not a simple shift of α_0 on the grid of (α_0, α_1). While $\hat{\alpha}_0$ and $\hat{\alpha}_1$ are indepedent due to the centering of `TV`, $\hat{\beta}_0$ and $\hat{\beta}_1$ are dependent. In addition, the book figure shows RSS divided by 1000, for no obvious reason. The following generates correct figures.

```r
ngrid = 100
grid1 = seq(5.2, 8.8, length.out=ngrid)
grid2 = seq(0.028, 0.068, length.out=ngrid)
## expand.grid(grid1, grid2) gives the same results
grid3 = data.frame(b0=rep(grid1, ngrid), b1=rep(grid2, each=ngrid))

RSS = NULL
for(i in 1:(ngrid^2))
  RSS[i] = with(Advertising, sum((sales - grid3$b0[i] - grid3$b1[i] * TV)^2))
range(RSS)

par(mfrow=c(1,2))
```

```r
contour(grid1, grid2, matrix(RSS, ngrid), xlab='beta0', ylab='beta1', nlevels=20)
persp(grid1, grid2, matrix(RSS, ngrid), theta=30, phi=20, xlab='beta0', ylab='beta1', zlab='RSS')
```

Analysts and programmers sometimes implement shortcut algorithms. When a shortcut is not equivalent to the intended approach, there can be unexpected consequences.

- 3.1.2: 4 things: (1) **least squares line** $\hat{f}(x)$ (estimated from data), (2) **population regression line** when $n = \infty$ (unknown), (3) **true curve** $f(x)$ (unknown), (4) a **future outcome** y_0 at $X = x_0$. At $X = x_0$, (1) yields the predicted value $\hat{f}(x_0)$. The expected squared error loss at $X = x_0$ is

$$E(y_0 - \hat{f}(x_0))^2 = Var[\hat{f}(x_0)] + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon). \tag{2.7}$$

  Here, $Var[\hat{f}(x_0)]$ reflects the discrepancy between (1) and (2) (i.e., sampling variation), $[Bias(\hat{f}(x_0))]^2$ reflects the discrepancy between (2) and (3) (i.e., modeling choice), and $Var(\epsilon)$ reflects the discrepancy between (3) and (4) (i.e., irreducible error). An example can be shown using this code:

```r
truef = function(x) -3 + 2*x + 0.7*x^2
N = 20; x = runif(N, 2, 8); y = truef(x) + rnorm(N, 0, 5)  ## real data
plot(x,y, xlim=c(2,8), ylim=c(-10,60))
curve(truef, add=T, lwd=3, col=2)
abline(lm(y ~ x), lty=2, lwd=2)
n = 1e6; xpop = runif(n, 2, 8); ypop = truef(xpop) + rnorm(n, 0, 5)  ## very large n
abline(lm(ypop ~ xpop), lwd=3)
legend(5, 15, bty='n',
       col=c(2,1,1), lty=c(1,1,2), lwd=c(3,3,2),
       c("True function", "Population line", "Fitted line"))
```

Parameters: In least squares regression model $y = \beta_0 + \beta_1 x$, there are 3 parameters: regression coefficients β_0 and β_1, and residual variance σ^2.

Every regression coefficient has a unit; e.g., $\hat{\beta}_1 = 0.0475$ means 0.0475K (or 47.5) units sold for every \$1000 spent on TV advertising. If `TV` is recorded in dollars, $\hat{\beta}_1$ would be 0.0000475.

Parameter estimation: *Point estimate* $\hat{\beta}_1$. *Standard error* SE($\hat{\beta}_1$) is the accuracy of the estimate $\hat{\beta}_1$. A *confidence interval* $(\hat{\beta}_{1l}, \hat{\beta}_{1u})$ is an interval estimate of β_1. When the model is correct, a 95% CI should cover the true β_1 95% of the time (i.e., for 95% of all possible datasets). A 99% CI is wider than a 95% CI.

- *Bias* of $\hat{\beta}_1$ is $Bias(\hat{\beta}_1) = E(\hat{\beta}_1) - \beta_1$. It is a long-term property. An estimator is unbiased if its bias is zero.
- *Mean squared error* (MSE) of $\hat{\beta}_1$ is $E[(\hat{\beta}_1 - \beta_1)^2] = Var(\hat{\beta}_1) + [Bias(\hat{\beta}_1)]^2$. MSE is a better measure of performance of an estimator than unbiasedness because MSE takes both variance and bias into account.

```r
fittv = lm(sales ~ TV, data=Advertising)
summary(fittv)  ## nice printout of summary
summary(fittv)$coef; summary(fittv)$r.squared  ## extract results
confint(fittv)  ## CI for parameter estimates
```

Many of these measures are calculated with the assumption that the model is correct. They can be misleading when the linearity assumption is invalid.

Hypothesis testing: $H_0 : \beta_1 = 0$ vs. $H_a : \beta_1 \neq 0$. The two hypotheses are not treated equally. The p-value is computed under H_0 (i.e., assuming H_0 is correct). It is the probability of seeing a higher level of departure from H_0 than the data when H_0 is correct. Different ways of quantifying the "departure" lead to different tests. The smaller p the more statistically significant. But a non-significant p-value does not mean we accept H_0.
Absence of evidence is not evidence of absence. — Altman and Bland (1995, BMJ)

Notes: (1) A p-value is **not** the probability that H_a would be correct. (2) Statistical significance is different from biological significance. (3) For simple tests, there is often a correspondence between p-value and CI (e.g., $p < 0.05$ corresponds to 95% CI not covering the true β). (4) Ignore the p-value for intercept.

- 3.1.3: **Measures of fit**: Residual SE (RSE) depends on the scale of outcome. R^2 is scale independent. R^2 increases with model complexity, and thus it cannot be used for model selection. R^2 has three interpretations:

(1) $R^2 = [corr(y, \hat{y})]^2$; (2) R^2 is the fraction of outcome total SS explained by the model; (3) R^2 is closely related to the likelihood-ratio statistic when the residuals are assumed to be normal.

2.1.2 ISLR 3.2 Multiple linear regression (multiple predictors)

- 3.2.1: Interpretation of **an individual p-value** in a regression table is the significance of the *item* **after** all other *items* in the regression table have been taken into account. In other words, the interpretation of an individual p-value always depends on the context of what other variables or transformations are considered.

As a result, the p-value for a variable should be ignored if the variable is also involved in a transformation or an interaction term (details below). Also see my comment on ISLR 3.3.2 for interpretion of the coefficient of a variable when it is also involved in an interaction term.

Also as a result, the following can happen in practice:

- x_1 becomes less significant after x_2 is added to regression (often due to collinearity between x_1 and x_2)
- x_1 becomes more significant after x_2 is added to regression (often due to both being important predictors)

```
cor(Advertising)
summary(lm(sales ~ TV, data=Advertising))$coef  ## Table 3.1
summary(lm(sales ~ radio, data=Advertising))$coef  ## Table 3.3
summary(lm(sales ~ newspaper, data=Advertising))$coef  ## Table 3.3
summary(lm(sales ~ TV + radio + newspaper, data=Advertising))$coef  ## Table 3.4
summary(lm(sales ~ TV + radio, data=Advertising))$coef
summary(lm(sales ~ radio + newspaper, data=Advertising))$coef
summary(lm(sales ~ TV + newspaper, data=Advertising))$coef
```

Tables 3.1, 3.3, 3.4: (1) `TV` and `radio` are nearly independent, and both have effects on `sales`. With both in the model, each benefits from the other's presence and becomes a lot more significant than being alone in a model. This is because the residual noise is a lot reduced when both are in the model. Similar phenomenon occurs between `TV` and `newspaper`. (2) `radio` and `newspaper` are correlated. With both in the model, both have reduced significance.

- 3.2.2: 4 questions:
  1. F-test for $H_0 : \beta_a = \cdots = \beta_b = 0$ vs. H_a : Any of these $\neq 0$. The two hypotheses can be viewed as **nested models**: the model under H_1 is a "full" model and the model under H_0 is a "reduced" model. We test if the "full" model explains the outcome significantly better than the "reduced" model. (Note that likelihood-ratio methods also require nested models.)
  2. Model selection criteria: Mallow's C_p, AIC, BIC, adjusted R^2 (details in ISLR 6). Stepwise procedure for model building.
  3. Measures of model fit: RSE, R^2.
  4. Confidence interval vs. prediction interval.

```
fittv = lm(sales ~ TV, data=Advertising)
predict(fittv, data.frame(TV=seq(100,300,100)), interval="confidence")  ## CI for fitted values
predict(fittv, data.frame(TV=seq(100,300,100)), interval="prediction")  ## PI for predicted values
```

2.1.3 3D plots in R

Various ways of generating scatter plots in R: http://www.statmethods.net/graphs/scatterplot.html . In Ubuntu Linux 16.04, installation of the `rgl` package requires `mesa-common-dev` and `libglu1-mesa-dev` installed in the OS.

```
fit1 = lm(sales ~ TV + radio, data=Advertising)

library(scatterplot3d)  ## 3D graphs using scatterplot3d()
s3d = with(Advertising, scatterplot3d(TV, radio, sales, highlight.3d=TRUE, angle=20))
s3d$plane3d(fit1)  ## add the fitted plane

library(rgl)            ## 3D interactive graphs using plot3d()
```

```
with(Advertising, plot3d(TV, radio, sales, col=2, size=5))
planes3d(fit1$coef[2], fit1$coef[3], -1, fit1$coef[1], alpha=.5)

## traditional residual plot
with(Advertising, plot(TV, resid(fit1))); abline(h=0)
```

2.1.4 A model building exercise

The plots above show a clear lack of fit in model `fit1`. We demonstrate two ways of model building, one traditional and one with cross-validation. We now fit an interaction model, where the effect of `TV` on `sales` is linear, but with both intercept and slope changing as linear functions of `radio` (and vise versa).

$$\begin{aligned}\text{sales} &= (\beta_0 + \beta_1\text{radio}) + (\beta_2 + \beta_3\text{radio}) \cdot \text{TV} = \beta_0 + \beta_1\text{radio} + \beta_2\text{TV} + \beta_3\text{radio} \cdot \text{TV} \\ &= (\beta_0 + \beta_2\text{TV}) + (\beta_1 + \beta_3\text{TV}) \cdot \text{radio}.\end{aligned}$$

```
fit2 = lm(sales ~ TV * radio, data=Advertising)
summary(fit2)
anova(fit1, fit2)  ## significant improvement

## draw the fitted curve
ngrid = 200
grid1 = seq(min(Advertising$TV), max(Advertising$TV), length.out=ngrid)
grid2 = seq(min(Advertising$radio), max(Advertising$radio), length.out=ngrid)
grid3 = data.frame(TV=rep(grid1, ngrid), radio=rep(grid2, each=ngrid))

fit2pred = predict(fit2, grid3)
with(Advertising, plot3d(TV, radio, sales, col=2, size=5))
plot3d(grid3$TV, grid3$radio, fit2pred, add=T, alpha=.5)

with(Advertising, plot(TV, resid(fit2))); abline(h=0)
```

There is still a clear lack of fit in model `fit2`. We model the effect of `TV` using a quadratic function, with all three coefficients changing linearly with `radio`.

$$\begin{aligned}\text{sales} &= (\beta_0 + \beta_1\text{radio}) + (\beta_2 + \beta_3\text{radio}) \cdot \text{TV} + (\beta_4 + \beta_5\text{radio}) \cdot \text{TV}^2 \\ &= \beta_0 + \beta_1\text{radio} + \beta_2\text{TV} + \beta_4\text{TV}^2 + \beta_3\text{radio} \cdot \text{TV} + \beta_5\text{radio} \cdot \text{TV}^2 \\ &= (\beta_0 + \beta_2\text{TV} + \beta_4\text{TV}^2) + (\beta_1 + \beta_3\text{TV} + \beta_5\text{TV}^2) \cdot \text{radio}.\end{aligned}$$

Thus, models `fit3` and `fit3b` below are the same, although they have different names for the model items.

```
fit3 = lm(sales ~ poly(TV, 2, raw=T) * radio, data=Advertising)
fit3b = lm(sales ~ TV * radio + I(TV^2)*radio, data=Advertising)
summary(fit3)$coef
summary(fit3b)$coef
anova(fit2, fit3)  ## significant improvement

fit3pred = predict(fit3, grid3)
with(Advertising, plot3d(TV, radio, sales, col=2, size=5))
plot3d(grid3$TV, grid3$radio, fit3pred, add=T, alpha=.5)

with(Advertising, plot(TV, resid(fit3))); abline(h=0)
```

There is still noticeable lack of fit in model `fit3`. We model the effect of `TV` using a cubic function, with all four coefficients changing as linear functions of `radio`.

```
fit4 = lm(sales ~ poly(TV, 3, raw=T) * radio, data=Advertising)
summary(fit4)
anova(fit3, fit4)  ## significant improvement

fit4pred = predict(fit4, grid3)
with(Advertising, plot3d(TV, radio, sales, col=2, size=5))
plot3d(grid3$TV, grid3$radio, fit4pred, add=T, alpha=.5)

with(Advertising, plot(TV, resid(fit4))); abline(h=0)

## Every model is a significant improvement over the previous one.
anova(fit1, fit2, fit3, fit4)
```

Model `fit4` does not show any visual sign of lack of fit, although the model can be further improved "significantly", as is shown below. Note that a model building process like this involves multiple testing, and might overfit the data.

```
fit5 = lm(sales ~ poly(TV, 4, raw=T) * radio, data=Advertising)
fit6 = lm(sales ~ poly(TV, 5, raw=T) * radio, data=Advertising)
fit7 = lm(sales ~ poly(TV, 6, raw=T) * radio, data=Advertising)
fit8 = lm(sales ~ poly(TV, 7, raw=T) * radio, data=Advertising)
fit9 = lm(sales ~ poly(TV, 8, raw=T) * radio, data=Advertising)
fit10= lm(sales ~ poly(TV, 9, raw=T) * radio, data=Advertising)
anova(fit1, fit2, fit3, fit4, fit5, fit6, fit7, fit8, fit9, fit10)

## The model fit9 has a really nice fit.
fit9pred = predict(fit9, grid3)
with(Advertising, plot3d(TV, radio, sales, col=2, size=5))
plot3d(grid3$TV, grid3$radio, fit9pred, add=T, alpha=.5)
with(Advertising, plot(TV, resid(fit9))); abline(h=0)
```

Cross-validation: Here the degree of polynomial is a hyperparameter. It can be determined using cross-validation. We use the `caret` package to do it. Since $n = 200$ is small, the results can vary from one split to another and vary with the test set proportion.

```
library(caret)

## Split data into training and test sets
inTraining <- createDataPartition(Advertising$sales, p=.8, list=FALSE)
training <- Advertising[inTraining, ]; dim(training)
testing <- Advertising[-inTraining, ]; dim(testing)

## Set up 8 repeats of 5-fold cross-validation of the training set.
fitControl = trainControl(method="repeatedcv", number=5, repeats=8)
degree=1:15
R2 = cvRMSE = cvRMSESD = testRMSE = NULL
for(dd in degree) {
    polymod = train(as.formula(bquote(sales ~ poly(TV, .(dd))*radio)),
                    data=training, method="lm", trControl=fitControl)
    R2[dd] = polymod$results$Rsquared
    cvRMSE[dd] = polymod$results$RMSE
    cvRMSESD[dd] = polymod$results$RMSESD
    testRMSE[dd] = sqrt(mean((testing$sales - predict(polymod, testing))^2))
}

library(ggplot2)
polymodresults = data.frame(degree, R2, cvRMSE, cvRMSESD, testRMSE)
ggplot(polymodresults, aes(x=degree, y=cvRMSE)) +
    geom_line() +
    geom_line(aes(y=cvRMSE+cvRMSESD), lty=2) +
    geom_line(aes(y=cvRMSE-cvRMSESD), lty=2) +
    geom_line(aes(y=testRMSE), color='red')
```

2.1.5 In a model with a quadratic term, what can be tested?

```
fit1b = lm(sales ~ TV + I(TV^2) + radio, data=Advertising)
summary(fit1b)
```

We can test: (1) if the quadratic term `TV^2` is significant after `radio` is taken into account (in two equivalent ways); (2) if TV advertising cost as modeled in `fit1b` contributes significantly after `radio` is taken into account. The answer might be different when the effect of `TV` is modeled differently (e.g., without the quadratic term, or when `radio` is not in the model). It is meaningless to test (3) if `TV` "itself" is significant when `TV^2` is still in the model.

```
## (1a) t-test for I(TV^2)
summary(fit1b)  ## p-value is summary(fit1b)$coef[3,4]
## (1b) F-test for I(TV^2)
anova(fit1, fit1b)
## (2) F-test for TV+TV^2 when radio is included
fitra = lm(sales ~ radio, data=Advertising)
anova(fit1b, fitra)
## F-test for TV+TV^2 when radio is not included
summary(lm(sales ~ TV + I(TV^2), data=Advertising))
```

2.1.6 Assignment

1. Reading for next lecture: ISLR 3.3–3.5; HOML Chapter 2 (to section "Discover and Visualize the Data to Gain Insights").
2. ISLR Chapter 3 R Labs

2.2 Week 2 Day 2

2.2.1 ISLR 3.3

- 3.3.1: Qualitative predictors: **Choice of baseline** when there are more than two categories. When a categorical variable is converted to a factor in R, by default its levels are determined by alphabetical order. When a predictor is a factor, by default its first level is the baseline category, and indicator variables are created for all other categories. The regression results are effectively the comparisons of all other categories with the baseline. If the baseline has a small sample size, the comparisons can be unreliable. It is often desirable to choose the most common category as the baseline.

```
## Credit data (decribed in 3.3.1)
Credit = read.table("Credit.csv", header=T, sep=',', row.names=1)
names(Credit); dim(Credit)
str(Credit)

## coding of a discrete preditor.  Hard to see that males are " Male"
summary(lm(Balance ~ Gender, data=Credit))
Credit$Gender
summary(lm(Balance ~ I(Gender=="Male"), data=Credit))  ## incorrect
levels(Credit$Gender)
summary(lm(Balance ~ I(Gender==" Male"), data=Credit))
```

```
## Default is to use the first level of the factor variable as the baseline.
summary(lm(Balance ~ Ethnicity, data=Credit))$coef
levels(Credit$Ethnicity)
table(Credit$Ethnicity)
## Use "Caucasian" as the baseline
summary(lm(Balance ~ I(Ethnicity=="Asian") + I(Ethnicity=="African American"), data=Credit))$coef

## I intentionally drop 90 AA subjects, leaving 9 in the data
toremove = sample(which(Credit$Ethnicity == "African American"), 90)
Credittest = Credit[-toremove, ]
table(Credittest$Ethnicity)
## AA as the baseline
summary(lm(Rating ~ Ethnicity, data=Credittest))$coef
## Caucasian as the baseline
summary(lm(Rating ~ I(Ethnicity=="Asian") + I(Ethnicity=="African American"), data=Credittest))$coef
```

- 3.3.2: **Modeling non-additive effects** between two predictors. **Interaction** is also called effect modification. One type of interaction is

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon && \\ &= (\beta_0 + \beta_2 X_2) + (\beta_1 + \beta_3 X_2) X_1 + \epsilon && = \alpha_0 + \alpha_1 X_1 + \epsilon \\ &= (\beta_0 + \beta_1 X_1) + (\beta_2 + \beta_3 X_1) X_2 + \epsilon && = \gamma_0 + \gamma_1 X_2 + \epsilon \end{aligned} \tag{3.31}$$

Product interaction is just one type of interaction. Because it is simple (and sometimes the only interaction taught in statistics curriculum), it has been widely used and sometimes mistakenly treated as **the** interaction. Note that when both predictors are binary, there is only one way of non-additivity (see Chapter 4 notes), and the product interaction can be treated as the interaction.

When interaction terms are included, the "main effect" term is a misnomer in commonly used variable coding. The "main effect" often reflects a certain baseline effect. Using (3.35) as an example,

$$\begin{aligned} \text{balance} &= \beta_0 + \beta_1 \text{income} + \beta_2 I(\text{student}) + \beta_3 \text{income} \cdot I(\text{student}) + \epsilon \\ &= \begin{cases} (\beta_0 + \beta_2) + (\beta_1 + \beta_3)\text{income} + \epsilon & \text{if student} \\ \beta_0 + \beta_1 \text{income} + \epsilon & \text{if not student} \end{cases} \end{aligned} \tag{3.35}$$

In this model, the "main effect" for income, β_1, is the effect of income among non-students, not the overall effect of income on balance. The "main effect" for student, β_2, is the shift of intercept for a student vs. a non-student, not the overall effect of being a student because being a student modifies not only the intercept but also the slope. Here, the effect of income is β_1 for non-students and $\beta_1 + \beta_3$ for students; the effect of being a student modifies the effect of income, thus the name "effect modification".

In an interaction model, what can be tested?

```
fit2 = lm(sales ~ TV * radio, data=Advertising)
summary(fit2)
```

We can test the following: (1) if the interaction term is significant (in two equivalent ways):

```
## (1a) t-test for TV:radio
summary(fit2)  ## p-value is summary(fit2)$coef[4,4]
## (1b) F-test for TV:radio
fit1 = lm(sales ~ TV + radio, data=Advertising)
anova(fit1, fit2)  ## p-value is anova(fit1, fit2)$Pr[2]
```

(2) if TV advertising cost as modeled in `fit2` contributes significantly after `radio` has been taken into account. (This is different from the question if `TV` as modeled in `fit1` is significant.)

```
fitra = lm(sales ~ radio, data=Advertising)
anova(fit2, fitra)
```

(3) if radio advertising cost contributes significantly in the interaction model after `TV` has been taken into account.

```r
fittv = lm(sales ~ TV, data=Advertising)
anova(fit2, fittv)
```

It is meaningless to test (4) if `TV` "itself" is significant (when `TV:radio` is still in the model); or (5) if `radio` "itself" is significant (when `TV:radio` is still in the model). That is, ignore the p-values for these "main effect" terms.

Modeling nonlinearity of a predictor. Polynomials are not as good as splines (ISLR 7). Here we demonstrate various nonlinear models using polynomials, natural splines, smoothing spline, kernel smoothing, and lowess.

```r
library(splines)  ## for splines, ns()
library(ISLR)
str(Auto)

attach(Auto)

plot(horsepower, mpg, bty='n')
## linear model fitted line in black; df=1 from x; total df=2
abline(lm(mpg ~ horsepower))
idx = order(horsepower)
## 2nd order polynomial (quadratic model) fitted curve in green; df=2 from x
points(horsepower[idx], fitted(lm(mpg ~ poly(horsepower, 2, raw=T)))[idx],
       type='l', col=3, lwd=1)
## 5th order polynomial fitted curve in green (thick); df=5 from x
points(horsepower[idx], fitted(lm(mpg ~ poly(horsepower, 5, raw=T)))[idx],
       type='l', col=3, lwd=3)
## natural cubic spline in red; df=2 from x
points(horsepower[idx], fitted(lm(mpg ~ ns(horsepower, df=2)))[idx],
       type='l', col=2, lwd=1)
## natural cubic spline in red (thick); df=5 from x
points(horsepower[idx], fitted(lm(mpg ~ ns(horsepower, df=5)))[idx],
       type='l', col=2, lwd=3)
## cubic smoothing spline in blue (thick); df=5 from x
lines(smooth.spline(horsepower, mpg, df=5), col=4, lwd=3)
## kernel smoothing
lines(ksmooth(horsepower, mpg, "normal", bandwidth=50), col=6, lwd=1)
lines(ksmooth(horsepower, mpg, "normal", bandwidth=25), col=6, lwd=3)
## loess
lines(horsepower[idx], predict(loess(mpg ~ horsepower, degree=1))[idx], col=5, lwd=1)
lines(horsepower[idx], predict(loess(mpg ~ horsepower, degree=2))[idx], col=5, lwd=3)
legend(100, 48, bty='n',
       col=c(1,3,3,2,2,4,6,6,5,5), lwd=c(1,1,3,1,3,3,1,3,1,3),
       c('linear regression (df=1)', 'polynomial (df=2)', 'polynomial (df=5)',
         'natural cubic spline (df=2)', 'natural cubic spline (df=5)',
         'cubic smoothing spline (df=5)',
         'gaussian kernel smoothing (bandwidth=50)',
         'gaussian kernel smoothing (bandwidth=25)',
         'gaussian kernel local linear regression',
         'gaussian kernel local quadratic regression'))

detach()
```

- 3.3.3: 6 issues in regression (shown in 7 Figures 3.9 – 3.15):
  1. **Nonlinearity**.
  2. **Correlation of errors** for time series data. Also for clustered and longitudinal data.
  3. **Non-constant variance**. Also called heteroscedasticity.
  4. **Outliers**: In regression, an outlier is one that has a very large discrepancy between observed and fitted

values compared to the other observations. It is about y and can be flagged using studentized residuals (Figure 3.12). In other context outside regression, an outlier is an observation that is very distant from other observations. Outliers may not be errors.

5. High-leverage points: depends only on X. The **leverage** of x_i is the i-th value on the diagonal of the design matrix $H = X(X'X)^{-1}X'$. It is $\frac{1}{n} + \frac{1}{n-1}d_i^2$, where d_i is the Mahalanobis distance (i.e. sphered distance) between x_i and the center $\bar{x}$ of all x's. The average leverage is $\frac{p+1}{n}$.
   An **influential observation** is one that, once removed, has a large effect on parameter estimates. It can be an outlier or an observation with a high leverage.

A high-leverage point can make SE(slope) a lot smaller, giving misleadingly high "confidence" on the accuracy of the slope estimate. An outlier can make SE(slope) a lot larger. We demonstrate these two concepts below:

```r
set.seed(100)
n = 30
x0 = rnorm(n)
y0 = x0 + rnorm(n, 0, .5)
mod0 = lm(y0~x0)
x1 = c(x0, 6); y1 = c(y0, 2); mod1 = lm(y1~x1)  ## add (6,2), leverage
x2 = c(x0, 1); y2 = c(y0, 5); mod2 = lm(y2~x2)  ## add (1,5), outlier
x3 = c(x0, 6); y3 = c(y0, 5); mod3 = lm(y3~x3)  ## add (6,5), leverage
x4 = c(x0, 0); y4 = c(y0, 5); mod4 = lm(y4~x4)  ## add (0,5), outlier, not influential

plot(x0, y0, xlim=c(-2, 6), ylim=c(-2.5, 5)); abline(mod0)
points(6, 2, col=2); abline(mod1, col=2, lty=2)
points(1, 5, col=3); abline(mod2, col=3, lty=2)
points(6, 5, col=4); abline(mod3, col=4, lty=2)
points(0, 5, col=5); abline(mod4, col=5, lty=2)

plot(lm(y4 ~ x4))
summary(mod0)$coef
```

- 6. **Collinearity**. Variance inflation factor $\text{VIF}(x_j) = (1 - R^2_{x_j|X_{-j}})^{-1}$. The higher $R^2_{x_j|X_{-j}}$ the higher VIF. Rule of thumb: VIF>5 is high collinearity. VIF is also $var_F(\hat{\beta}_j)/var_j(\hat{\beta}_j)$, the ratio of variances of $\hat{\beta}_j$, one from the full model and one from the model with only x_j.

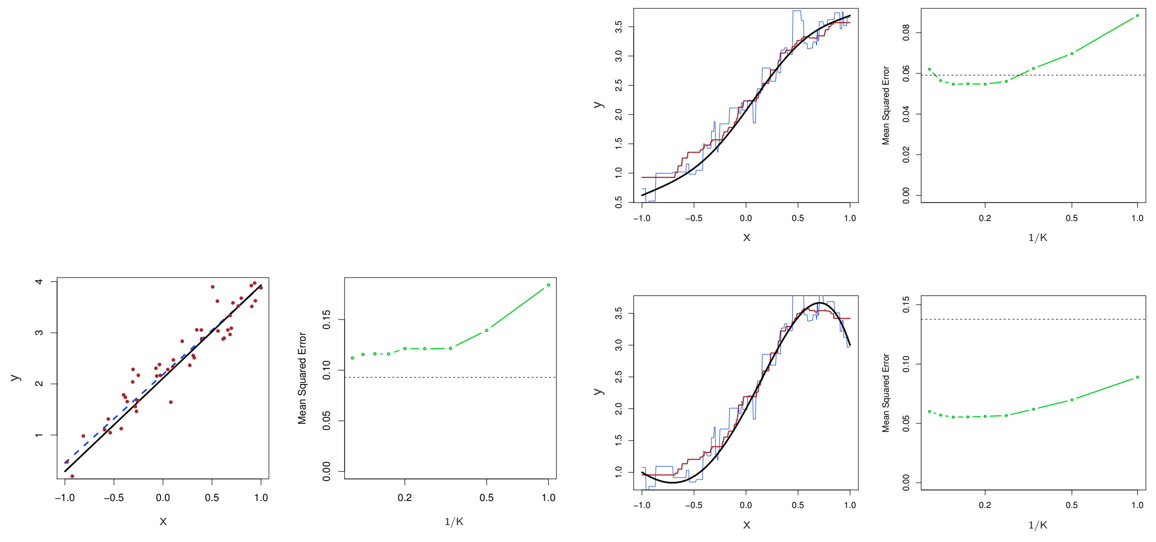
```r
#### Collinearity
Credit = read.table("Credit.csv", header=T, sep=',', row.names=1)  ## simulated data
names(Credit)
pairs(Credit[, c(1:3,11)])
cor(Credit[, c(1:3,11)])
summary(lm(Balance ~ Limit, data=Credit))$coef
summary(lm(Balance ~ Rating, data=Credit))$coef
summary(lm(Balance ~ Limit + Rating, data=Credit))$coef
summary(lm(Balance ~ Income, data=Credit))$coef
summary(lm(Balance ~ Rating + Income, data=Credit))$coef
summary(lm(Balance ~ Limit + Income, data=Credit))$coef
summary(lm(Balance ~ Limit + Rating + Income, data=Credit))$coef
car::vif(lm(Balance ~ Limit + Rating + Income, data=Credit))  ## VIF

## When viewed from a certain angle in 3D, two layers are present (simulation artifact?).
library(rgl)
with(Credit, plot3d(Rating, Income, Balance))
```

2.2.2 ISLR 3.4

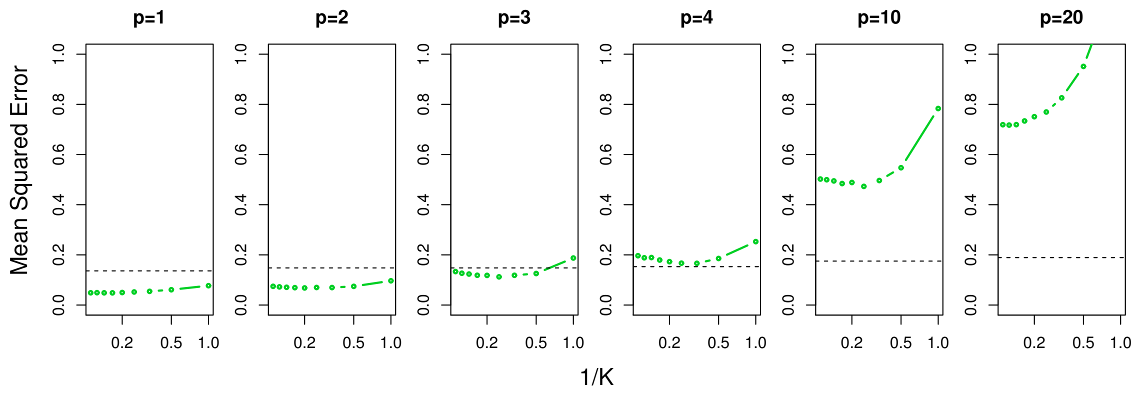
Answer the 7 questions at the beginning of chapter. For Q1, I would also use the total advertising cost.

2.2.3 ISLR 3.5

Linear regression vs. KNN (Figures 3.18–3.19)



Curse of dimensionality: In Figure 3.20, only one predictor is relevant to the outcome and all others are noise.



In general, for local estimation methods (e.g., KNN), the more noise predictors the worse performance. When many noise predictors are present, two points can be "close" because they have similar values on the noise predictors, not the relevant ones. Neighbors that are close with respect to the relevant predictors can be pulled away by noise predictors. We try to demonstrate this below.

```r
set.seed(42); N=30
x1 = runif(N); x2 = runif(N); x3 = runif(N)
y = x1 + rnorm(N, 0, .4)

plot(x1, y)
points(x1, rep(par("usr")[3]+0.01,N), col=2, pch=19)
rgl::plot3d(x1, x2, x3, col=2, size=8)
```

2.2.4 Chapter 3 R Labs

For a linear model, the following functions can be used: `confint()`, `predict()`, `plot()`, `residuals()`, `rstudent()`, `hatvalues()`, `car::vif()`.

2.2.5 HOML 2

A few good suggestions not only for machine learning tasks, but also for general data analysis tasks.

1) Work with real data!
2) Know the context of your problem and select your approach accordingly: (a) How your model/prediction is going to be used in the overall pipeline or decision making; (b) What are the current approaches on this problem; (c) Any insights or preferences.
3) Code up the steps for processing (including downloading and uncompression) into scripts or functions. This increases reproducibility.
4) Explore the data. Simple summaries and plots on individual variables and pairs of variables are very helpful. Such explorations help you not only understand the data, but also identify errors, discrepancies, and outliers.
5) Create new variables if you think they may be relevant (e.g., rooms/household, bedrooms/room).
6) Once the model is being used: (a) Monitor its performance over time for signs of performance degradation; (b) Monitor the input variables for signs of change (e.g., shifted mean, reduced or increased variation); (c) Frequently retrain the model with new data.
7) Never look at your test set. Avoid **data snooping bias**.

Data splitting: When splitting data to form training/test sets or to create subsets of the training set for cross-validation, there are two goals.

(1) The split of training/testing is stable between the current dataset and future possibly expanded datasets. That is, an observation should always end up in the same set. If a unique ID exists or can be created (e.g., using stable values such as longitude and latitude in the housing data), then hashing the ID and taking the last byte of the hash yield a function $f : X \to \{0, 1, \cdots, 255\}$ with $f(X)$ effectively independent of X. Then to take a random 20% subset, one can use $f(X) \leq 50$. If one cannot obtain or define a unique ID and has to do random selection using row number, make sure future data are appended to the end of current data.

(2) The subsets are comparable on key characteristics (e.g., sex, income category). Do **stratified splitting**. (In R, `caret::createDataPartition()`; in Python, `sklearn.model_selection.StratifiedShuffleSplit`) (Similar techniques are used in other areas: stratified sampling in survey designs and block randomization in clinical trials.)

Other good points:

- Correlation measures linear relationships; magnitude of correlation (Figure 2-14).
- Coding of a (non-ordered) categorical variable to dummy variables (one-hot encoding). In R, `model.matrix(~ var-1)`; in Python, `sklearn.preprocessing.LabelBinarizer.fit_transform()` ?
  Sparse matrix coding: In R, use `Matrix` package; in Python `LabelBinarizer.fit_transform()` with argument `sparce_output=TRUE`.
- Feature scaling: min-max scaling $f(x) = (x - x_{\min})/(x_{\max} - x_{\min})$ and standardization $f(x) = (x - \bar{x})/sd(x)$. In Statistics, "normalization" is synonymous to "standardization".

Misleading recommendations: (1) Remove "data quirks" (shown in Figure 2-16). (2) Missing data strategies: dropping variables/observations or single imputation with median (in "Data Cleaning" on pages 60–61).

(1) Data quirks: One should consult subject matter experts to learn why they are there and then decide how to deal with them. If one decides to remove them, it is better to do two sets of analyses, one including the "data quirks" and one excluding them to see if the results are sensitive to their removal. This is called sensitivity analysis.

(2) Missing data:

- Dropping a variable with missing data may remove an informative variable. Not recommended unless the missing rate is high for the variable.
- Dropping observations with missing data. If the data are missing completely at random (MCAR), this strategy loses information but has no bias. For non-random missing data, this leads to at least sampling bias. Investigate if there is any sign of non-randomness of the missing data. You may create an indicator variable for missingness and investigate if it correlates with other variables or can be predicted by other variables.

- Single imputation: with a constant (mean or median), with "last observation carried forward", or with a predicted value from a model. One common problem of single imputation is lack of variation/uncertainty because the imputed values will be treated as if they are observed in downstream analyses. Imputing with a constant creates (partial) independence between the imputed variable and other variables. Unfortunately, because of its simplicity, single imputation is prevalent in many fields. When imputing with a constant, make sure the same value is used across all subsets (training/validation/test sets).
- Multiple imputation is highly recommended. But it is computationally intensive. The advantage of MI over SI has been well demonstrated with respect to parameter estimation. (I expect the advantage also exists with respect to prediction, but I am not aware of any studies on this.)

For those who are curious, missing data fall in three categories: MCAR, missing at random (MAR) (e.g., if males tends to ignore questions on depression but this ignorance does not depend on their level of depression), missing not at random (MNAR) or nonignorable nonresponse (e.g., if ignoring questions on depression when their level of depression is high).

2.2.6 Assignment

1. **Homework**: ISLR Chapter 3 Exercises 9, 15
2. **Homework**: Play with the Titanic dataset to predict survival status. Do all model training and tuning on the training set. The data are available in an R package `titanic`; same data in .csv format are available from https://www.kaggle.com/c/titanic/data . The test set does not contain the outcome. R also has an object `Titanic`, which is a simple data tabulation with a slight discrepancy with the `titanic` package: `Titanic` indicates there were 1316 passengers, while the `titanic` package contains 1309 (891 in training set and 418 in test set). In the R package, ignore the two objects `titanic_gender_class_model` and `titanic_gender_model`. They are probably leftover objects from the package creator's own models. There are many analysis demonstrations of this dataset online, both in R and Python. Do not look at them. Try to get as much as possible yourself in 2 weeks. Then look at other poeple's solutions online (by just googling). Summarize what you learned from them and what you did better than them. Turn in your final prediction model and the predictions on the test set (as a 418×2 object or a file with two columns: PassengerId, Survived).
3. Reading for next lecture: ISLR 4.1-4.3; HOML Chapter 2 ("Prepare the Data for Machine Learning Algorithms").
4. ISLR Chapter 4 R Labs