4. 회귀와 예측

통계학에서 가장 일반적인 목표는 다음 질문들에 대한 답을 찾는 것이다.

변수 X(혹은 $X_1, \cdots, X_p$)가 변수 Y 와 관련이 있는가? 있다면 어떤 관련이 있는가? 이를 이용해 Y를 예측할 수 있는가?

특히 '예측' 변수 값을 기반으로 결과(목표) 변수를 예측하는 영역만큼 통계와 데이터 과학이 서로 강하게 연결되는 부분도 없다. 또 다른 중요한 연결 고리는 **이상 검출**영역이다. 회귀 진단은 원래 데이터 분석을 위해 개발되었고 이것이 발전하면서 비정상적인 데이터를 검출하는 데도 사용되고 있다. 상관관계 및 선형회귀에 대한 이야기는 지금부터 한 세기를 거슬러 올라가야 한다.

4.1 단순선형회귀

단순선형회귀 모델은 한 변수와 또 다른 변수의 크기 사이에 어떤 관계, 예를 들면 X가 증가하면 Y도 증가, 아니면 반대로 X가 증가하면 Y는 감소하는 식의 관계가 있는지를 보여준다. 앞서 다룬 상관관계 역시 두 변수가 서로 어떤 관계인지를 보여주는 방법 중 하나이다. 상관관계가 두 변수 사이의 전체적인 관련 강도를 측정하는 것이라면, 회귀는 관계 자체를 정량화하는 방법이라는 점에서 차이가 있다.

용어정리

- **응답변수(반응변수) : 예측하고자 하는 변수(유의어 : 종속변수, 변수 Y, 목표, 출력)**
- **-독립변수 : 응답치를 예측하기 위해 사용되는 변수(유의어 : 변수 X, 피처, 속성)**
- **레코드 : 한 특정 경우에 대한 입력과 출력을 담고 있는 벡터(유의어 : 행, 사건, 예시, 예제)**
- **절편(intercept) : 회귀직선의 절편, 즉 $X = 0$일때 예측값 (유의어 : b_0, β_0)**
- **회귀계수 (regression efficient) : 회귀직선의 기울기 (유의어 : 기울기, b_1, β_1, 모수 추정값, 가중치)**
- **적합값(fitted value) : 회귀선으로부터 얻은 추정치 $\hat{Y_i}$ (유의어 : 예측값)**
- **잔차(residual) : 관측값과 적합값의 차이 (유의어 : 오차)**
- **최소제곱(least square) : 잔차의 제곱합을 최소화하여 회귀를 피팅하는 방법(유의어 : 보통최소제곱)**

4.1.1 회귀식

단순선형회귀를 통해 X가 어느 정도 변하는지를 정확히 추정할 수 있다. 상관계수의 경우, 변수 X

와 Y가 서로 바뀌어도 상관없다. 회귀에서는 다음과 같은 식으로 선형관계(즉, 직선)를 이용해서 변수 X로부터 변수 Y를 예측하고자 한다.

$Y = b_0 + b_1 X$

이는 'Y는 X에 b_1을 곱하고 거기에 b_0를 더한 값과 같다'라는 뜻이다. b_0는 **절편(상수)** 그리고 b_1는 X의 **기울기**라고 한다. 보통은 b_1을 주로 **계수**라고 하는데, R에서는 둘 다 계수라고 출력된다. 변수 Y는 X에 따라 달라지기 때문에, **응답변수** 또는 **종속변수** 라고 불린다. 변수 X는 **독립변수** 혹은 **예측변수**라고 한다. 머신러닝 분야에서, Y는 **목표벡터**, X는 **피처벡터**라고 달리 불린다.

다음 그림에서 보이는 산점도는 노동자들이 면진(Exposure)에 노출된 연수와 폐활량(PEFR)을 표시한 것이다. PEFR과 Exposure 변수 사이에 어떤 관계가 있을까? 이 그림만 보고는 뭐라 말하기가 어렵다.

```
model <- lm(PEFR ~ Exposure, data=lung)
>model

Call:
lm(formula = PEFR ~ Exposure, data = lung)

Coefficients:
(Intercept)     Exposure
    424.583       -4.185
```

단순선형회귀는 예측변수 Exposure에 대한 함수로 응답변수 PEER을 예측하기 위한 가장 최선의 직선을 찾으려 시도한다.

$PEER = b_0 + b_1$ 노출 R 함수 lm으로 선형회귀 함수를 피팅한다.

lm이라는 함수이름은 **선형모형**을 뜻하는 linear model의 줄임말이다. 그리고 ~기호는 Exposure을 통해 peer을 예측한다는 것을 의미한다.

model 객체를 출력해보면, 다음과 같은 결과를 볼 수 있다.

```
plot(lung$Exposure, lung$PEFR, xlab="Exposure", ylab="PEFR",
    ylim=c(300,450), type="n", xaxs="i")
```

```
abline(a=model$coefficients[1],
b=model$coefficients[2], col="blue", lwd=2)
```

```
text(x=.3, y=model$coefficients[1],
labels=expression("b"[0]),  adj=0, cex=1.5)
```

```
x <- c(7.5, 17.5)
y <- predict(model, newdata=data.frame(Exposure=x))
segments(x[1], y[2], x[2], y[2] , col="red", lwd=2, lty=2)
segments(x[1], y[1], x[1], y[2] , col="red", lwd=2, lty=2)
text(x[1], mean(y), labels=expression(Delta~Y), pos=2, cex=1.5)
text(mean(x), y[2], labels=expression(Delta~X), pos=1, cex=1.5)
text(mean(x), 400, labels=expression(b[1] == frac(Delta ~ Y, Delta ~ X)),
cex=1.5)
```

4.1.2 적합값과 잔차

회귀분석에서 중요한 개념은 **적합값**과 **잔차**이다. 보통 모든 데이터가 정확히 한 직선안에 들어오지는 않는다. 따라서 회귀식은 명시적으로 오차항 e_i를 포함한다.

$Y_i = b_0 + b_1 X_i + e_i$

적합값은 **예측값**을 지칭하는 말로, 보통 $\hat{Y_i}$(Y햇) 으로 나타낸다. 다음과 같이 쓸 수 있다.

$\hat{Y_i} = \hat{b_0} + \hat{b_1} X_i$

$\hat{b_0}$과 $\hat{b_1}$은 이미 알려진 값이 아닌 추정을 통해 얻은 값이라는 것을 의미한다.

TIP_ 햇(^) 표기법 : 추정치

모자 같은 모양(^)의 기호가 문자 위에 있는 이러한 표기법을 햇 표기법이라고 한다. 햇 표기법은 추정치와 미리 알고 있는 값을 구분하기 위해 사용한다, 통계학자들은 왜 이렇게 추정값과 참값을 구분하려는 것일까? 참값은 불변의 확실한 값이라는 의미가 있는 반면, 추정치라는 것은 불확실성을 내포하고 있기 때문이다.

여기서 잔차 $\hat{e_i} = Y_i - \hat{Y_i}$ 으로 구한다.

R에서 제공하는 predict와 residual 함수를 통해 적합값과 잔차를 구할 수 있다.

```
fitted <- predict(model)
resid <- residuals(model)
```

다음 그림은 폐활량에 대한 회귀선으로부터 얻은 잔차를 설명한다. 데이터 포인트에서 직산 사이에 수직으로 그은 점선들이 바로 잔차를 의미한다.

```
png(filename=file.path(PSDS_PATH, 'figures', 'psds_0403.png'),  width = 4,
height=4, units='in', res=300)
par(mar=c(4,4,0,0)+.1)

lung1 <- lung %>%
  mutate(Fitted=fitted,
         positive = PEFR>Fitted) %>%
  group_by(Exposure, positive) %>%
  summarize(PEFR_max = max(PEFR),
            PEFR_min = min(PEFR),
            Fitted = first(Fitted)) %>%
  ungroup() %>%
  mutate(PEFR = ifelse(positive, PEFR_max, PEFR_min)) %>%
  arrange(Exposure)

plot(lung$Exposure, lung$PEFR, xlab="Exposure", ylab="PEFR")
abline(a=model$coefficients[1], b=model$coefficients[2], col="blue", lwd=2)
segments(lung1$Exposure, lung1$PEFR, lung1$Exposure, lung1$Fitted, col="red",
lty=3)
dev.off()
```

4.1.3 최소제곱

그럼 데이터를 피팅한 모델을 어떻게 만들 수 있을까? 관련성이 명확하다면 아마 손으로 피팅하여 선을 그릴 수 있을지도 모르겠다. 실무에서 회귀선은 잔차들을 제곱한 값들의 합인 **잔차제곱합(RSS)**을 최소화하는 선이다.

$$RSS = \sum_{i=1}^{n}(Y - \hat{Y_i})^2$$
$$= \sum_{i=1}^{n}(Y - \hat{b_0} - \hat{b_1}X_i)^2$$

다시 말해 추정치 $\hat{b_0}$ 과 $\hat{b_1}$은 RSS를 최소화하는 값이다.

잔차제곱합을 최소화하는 이러한 방법은 **최소제곱회귀** 혹은 **보통최소제곱(OLS)** 회귀라고 한다.

최소제곱회귀는 계수계산을 위해 다음과 같이 간단한 공식을 사용한다.

$$\hat{b_1} = \sum_{i=1}^{n}(Y_i - \bar{Y})(X_i - \bar{X}) / \sum_{i=1}^{n}(X_i - \bar{X})^2$$

$$\hat{b_0} = \bar{Y} - \hat{b}_1\bar{X}$$

역사적으로, 최소제곱이 회귀에서 널리 쓰이게 된 이유 중 하나가 바로 이 계산의 편의성 때문이다. 빅데이터 시대가 되었어도, 계산 속도가 여전히 중요한 요소 중 하나이다. 최소제곱은 평균과 마찬가지로 특잇값에 매우 민감하다. 이러한 경향은 크기가 작거나 중간 정도 되는 문제에서 심각한 문제가 될 수 있다. 회귀에서 특잇값에 대한 논의는 차후에 하도록 하겠다.

Note_ 회귀용어

데이터를 분석하고 연구하는 사람들이 회귀라는 용어를 사용할 때는 일반적으로 선형회귀를 의미하며, 이들은 예측변수와 수치형 출력값 사이의 관계를 설명하는 선형모형을 만드는 것에 초점을 둔다. 좀 더 공식적으로 통계적 관점에서의 회귀는 예측변수와 결과변수 사이의 일반적인 함수 관계를 다루는 비선형모형도 포함한다. 머신러닝 커뮤니티에서는 예측값이 수치형인 예측 모델을 사용하는 것을 의미하기도 한다(예측값이 이진형 혹은 범주형일 때는 이와 구분해 '분류'라고 부른다).

4.1.4 예측 대 설명(프로파일링)

역사적으로, 예측변수와 결과변수 사이에 있을 것으로 추정되는 선형 관계를 밝히는 것이 회귀분석의 주된 용도였다. 회귀로 피팅한 데이터를 통해, 데이터 간의 관계를 이해하고 그것을 설명하는 것을 목표로 해왔다.

즉, 회귀방정식의 기울기 $\hat{b}$ 을 추정하는 것에 주로 초점이 맞춰졌다. 경제학자들은 소비자 지출과 GDP 성장 간의 관계를 알고 싶어 한다. 공중 보건 기관은 안전한 성생활을 장려하는 데 어떤 홍보 캠페인이 더 효과적인지를 알고 싶어 할 수 있다. 이럴 경우, 개별 사건을 예측하는 것이 아닌 전체적인 관계를 이해하는 데 초점을 두어야 한다.

빅데이터의 출현과 함께 회귀분석은 수중에 있는 데이터를 설명하기보다는 새로운 데이터에 대한 개별 결과를 예측하는 모델(예측 모델)을 구성하는 데 널리 사용된다. 이때 주요 관심 사항은 적합값$\hat{Y}$ 이다. 마케팅에서는 회귀분석을 사용하여 광고 캠페인의 크기에 따른 수익 변화를 예측할 수 있다.

대학에서는 회귀분석을 사용하여 SAT 점수에 따라 학생의 평점을 예측하기도 한다.

데이터를 피팅한 회귀모형은 X의 변화가 Y의 변화를 유도하도록 설정된다. 하지만 회귀방정식 자체가 인과관계를 정확히 증명하는 것은 아니다. 인과관계에 대한 결론은 그 관계에 대한 더 폭넓은 이해를 바탕으로 해야한다. 예를 들면 회귀방정식은 웹 광고에서 클릭 수와 전환률 간의 명확한 관계를 보여줄 수 있다. 회귀방정식이 아닌, 마케팅 프로세스에 대한 지식을 발휘하면 광고 클릭이 판매로 연결된다는 결론을 이끌어낼 수 있다. 그 반대는 상식적으로 말이 안 된다.

주요개념

- **회귀방정식은 응답변수 Y와 예측변수 X 간의 관계를 선형함수로 모델링한다.**
- **회귀모형은 적합값과 잔차, 즉 반응에 대한 예측과 그 예측 오차를 산출한다.**
- **회귀모형은 일반적으로 최소제곱법을 이용해 피팅한다.**
- **회귀는 예측과 설명 모두에 사용된다.**

4.2 다중선형회귀

예측변수가 여러 개라면 수식은 이들을 모두 포함하는 다음과 같은 형태가 된다.

$$Y = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \cdots + b_pX_p + e$$

이것은 더 이상 직선의 형태는 아니지만, 각 계수와 그 변수(특징)들 사이의 관계는 여전히 선형이므로 선형모형이다.

용어정리

- **제곱근 평균제곱오차 : 회귀 시 평균제곱오차의 제곱근. 회귀모형을 평가는데 가장 널리 사용되는 측정 지표다. (유의어 : RMSE)**
- **잔차 표준오차 : 평균제곱오차와 동일하지만 자유도에 따라 보정된 값(유의어 : RSE)**
- **R-제곱 : 0~1 모델에 의해 설명된 분산의 비율(유의어 : 결정계수, R^2)**
- **t통계량 : 계수의 표준오차로 나눈 예측변수의 계수, 모델에서 변수의 중요도를 비교하는 비준이 된다.**
- **가중회귀 : 다른 가중치를 가진 레코드들을 회귀하는 방법**

최소제곱법을 이용한 피팅, 적합값과 잔차의 정의 같은 단순선형회귀에서 다룬 기타 모든 개념은 다중선형회귀에도 그대로 확장되어 적용된다. 일레로 적합값은 다음과 같다.

$$\hat{Y}_i = \hat{b}_0 + \hat{b}_2 X_{1,i} + \hat{b}_2 X_{2,i} + \cdots + \hat{b}_p X_{p,i}$$

4.2.1 킹 카운티 주택 정보 예제

회귀분석을 사용하는 대표적인 사례 중에 하나는 주택 가치를 추정하는 것이 있다. 주택 가격 평가사는 세금을 산정할 목적으로 주택 가치를 추정해야 한다. 부통산 소비자와 전문가는 질로 닷컴 인기 있는 웹사이트를 참고하여 공정한 가격을 확인한다. 다음은 house라는 변수명을 가진 data.frame 객체에 저장된 킹 카운티의 주택 가격 데이터 일부이다.

```
> head(house[, c("AdjSalePrice", "SqFtTotLiving", "SqFtLot", "Bathrooms",
+                "Bedrooms", "BldgGrade")])
  AdjSalePrice SqFtTotLiving SqFtLot Bathrooms Bedrooms BldgGrade
1       300805          2400    9373      3.00        6         7
2      1076162          3764   20156      3.75        4        10
3       761805          2060   26036      1.75        4         8
4       442065          3200    8618      3.75        5         7
5       297065          1720    8620      1.75        4         7
6       411781           930    1012      1.50        2         8
```

결국 목표는 이런 변수들로부터 판매 금액을 예측하는 것이다. lm 함수의 우변에 더 많은 항을 추가함으로써 다중회귀 사례를 처리한다.

na.action = na.omit 인수는 모델을 만들 때 결측값이 있는 레코드를 삭제하는 옵션이다.

```
> ## Code snippet 4.4
> house_lm <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
+                  Bedrooms + BldgGrade,
+                data=house, na.action=na.omit)
> ## Code snippet 4.5
> house_lm

Call:
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
    Bedrooms + BldgGrade, data = house, na.action = na.omit)

Coefficients:
  (Intercept)  SqFtTotLiving        SqFtLot      Bathrooms       Bedrooms
   -5.219e+05       2.288e+02      -6.051e-02      -1.944e+04      -4.778e+04
    BldgGrade
    1.061e+05
```

계수를 해석하는 방식은 단순선형회귀와 같다. 다른 모든 변수 X_k (단 $k \neq j$) 가 고정되었다고 가정했을 때, X_j가 변하는 정도에 따라, 예측값 $\hat{Y}$도 계수 b_j에 비례해 변화한다. 예를 들어 주택에 1제곱피트를 추가하면 예상 가격이 대략 229달러 정도 증가할 것이다. 1,000제곱피트를 추가하면 228,800달러 증가할 것이다.

4.2.2 모형평가

데이터 과학의 관점에서 가장 중요한 성능 지표는 바로 **제곱근 평균제곱오차(RMSE)**이다. RMSE는 예측된 $\hat{Y}_i$ 값들의 평균제곱오차 제곱근을 말한다.

$$RMSE = \sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2/n}$$

이것은 전반적인 모델의 정확도를 측정하고 다른 모델(머신러닝 기술로 학습된 모델을 포함)과 비교하기 위한 기준이 된다. 이외에도 RMSE와 유사한 **잔차 표준오차(RSE)**가 있다. 예측변수가 p개일 때, RSE는 다음과 같다.

$$RSE = \sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2/(n-p-1)}$$

유일한 한 가지 차이점은 분모가 데이터 수가 아닌 자유도라는 점이다. 실무에서 선형회귀분석을 할 때, RMSE와 RSE의 차이는 아주 작다. 특히 빅데이터 분야에서는 더 그렇다.

R의 summary 함수는 회귀모형에 대한 RSE뿐 아니라 다른 지표들도 계산한다.

```
> ## Code snippet 4.6
> summary(house_lm)

Call:
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
    Bedrooms + BldgGrade, data = house, na.action = na.omit)

Residuals:
     Min       1Q   Median       3Q      Max
-1199508  -118879   -20982    87414  9472982

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -5.219e+05  1.565e+04 -33.349  < 2e-16 ***
SqFtTotLiving  2.288e+02  3.898e+00  58.699  < 2e-16 ***
SqFtLot       -6.051e-02  6.118e-02  -0.989    0.323
Bathrooms     -1.944e+04  3.625e+03  -5.362 8.32e-08 ***
Bedrooms      -4.778e+04  2.489e+03 -19.194  < 2e-16 ***
BldgGrade      1.061e+05  2.396e+03  44.287  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 261200 on 22683 degrees of freedom
Multiple R-squared:  0.5407,    Adjusted R-squared:  0.5406
F-statistic:  5340 on 5 and 22683 DF,  p-value: < 2.2e-16
```

SW 출력에서 볼 수 있는 또 다른 유용한 지표는 **결정계수**라고도 부르는 **R 제곱 통계량** (R^2) 이다 .

R 제곱의 범위는 0~1 이며, 모델 데이터의 변동률을 측정한다. 모델이 데이터에 얼마나 적합한지 평가하고자 할 때, 회귀분석을 설명하기 위한 용도로 활용된다. R^2을 구하는 공식은 다음과 같다.

$$R^2 = 1 - (\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 / \sum_{i=1}^{n}(y_i - \bar{y})^2)$$

분모는 Y의 분산에 비례한다. 위 R 출력에서는 자유도를 고려한 **수정 R 제곱**값을 보여준다. 다중회귀분석에서 이는 일반 R 제곱과 크게 다르지 않다.

R은 추정한 계수들과 함께, 계수의 표준오차(SE)와 **t통계량**을 함께 출력하여 보여준다.

$$t_b = \hat{b}/SE(\hat{b})$$

t통계량, 그리고 늘 함께 따라다니는 p값은 계수가 '통계적으로 유의미한' 정도, 즉 예측변수와 목표변수를 랜덤하게 재배치했을 때 우연히 얻을 수 있는 범위를 어느정도 벗어났는지 측정한다. t통계량이 높을수록 (p값이 낮을수록)예측변수는 더욱 유의미하다. 사고 절약의 원리는 모델을 만드는 데 중요한 특징이므로, 예측변수를 포함할 변수를 어떻게 고르면 좋을지 알 방법이 있다면 아주 유용할 것이다.

CAUTION t통계량 말고도, R 및 다른 SW들은 ***p값과 F통계량*** *을 함께 출력한다. 데이터 과학자들은 일반적으로 이러한 통계해석이나 통계적 유의성 문제에 너무 깊이 관여하지 않는다. 데이터 과학자들은 모델에 예측변수를 포함할지 여부를 판단하기 위해 t통계량을 유용하게 사용한다. 높은 t통계량(p값이 0에 가까워짐)은 예측변수를 모델에 포함해야 된다는 것을 의미하고, 낮은 t통계량은 예측변수를 삭제할 수 있음을 나타낸다.*

4.2.3 교차타당성검사

지금까지 다룬 전형적인 통계적 회귀 측정 지표들(R^2, $F_{통계량}$, $p_{값}$) 은 모두 '표본 내' 지표들이다.

즉, 모델을 구하는 데 사용했던 데이터를 똑같이 그대로 사용한다. 직관적으로, 원래 데이터의 일부를 따로 떼어놓고 적합한 모델을 찾는 데 사용하지 않고, 모델을 만든 후 그 떼어놓았던 (홀드아웃) 데이터를 모델에 적용하면 모델의 성능을 확인할 수 있을 것이다. 일반적으로 데이터의 다수는 적합한 모델을 찾는 데 사용하고, 소수는 모델을 테스트 하는데 사용한다.

'표본 밖' 유효성 검사라는 이 아이디어가 새로운 것은 아니지만, 더 큰 데이터 집합들이 등장하면서 실제로 의미가 생기기 시작했다. 데이터 집합이 작다면, 누구나 일반적으로 가능한 모든 데이터를 사용해서 최상의 모델을 얻고자 할 것이다.

홀드아웃 샘플을 사용한다 하더라도, 상대적으로 작은 홀드아웃 샘플의 변동성으로 인해 불확실성을 초래할 수 있다. 다른 홀드아웃 표본을 선택했다면 평가는 어떻게 달라질까?

교차타당성검사란, 홀드아웃 샘플 아이디어를 여러 개의 연속된 홀드아웃 샘플로 확장한 것이다. 기본적인 **k 다중 교차타당성검사** 알고리즘은 다음과 같다.

1. 1/k의 데이터를 홀드아웃 샘플로 따로 떼어놓는다.
2. 남아 있는 데이터로 모델을 훈련시킨다.
3. 모델을 1/k 홀드아웃에 적용 (점수를 매김)하고 필요한 모델 평가 지표를 기록한다.
4. 데이터의 첫 번째 1/k을 복원하고 다음 1/k (앞에서 선택했던 레코드는 제외)을 따로 보관한다.
5. 2~3단계를 반복한다.
6. 모든 레코드가 홀드아웃 샘플로 사용될 때까지 반복한다.
7. 모델 평가 지표들을 평균과 같은 방식으로 결합한다.

훈련을 위한 샘플과 홀드아웃 샘플로 데이터를 나누는 것을 **폴드**라고 한다.

4.2.4 모형 선택 및 단계적 회귀

어떤 회귀분석 문제에서는 많은 변수를 예측변수로 사용할 수 있다. 예를 들어 주택 가치를 예측하기 위해 지하실 크기나 건축 연도와 같은 변수를 추가로 사용할 수 있다. R에서는 회귀방정식에 다음과 같이 쉽게 변수들을 추가할 수 있다.

```
house_full <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
                   Bedrooms + BldgGrade + PropertyType + NbrLivingUnits +
                   SqFtFinBasement + YrBuilt + YrRenovated + NewConstruction,
                 data=house, na.action=na.omit)
```

그러나 더 많은 변수를 추가한다고 해서 꼭 더 좋은 모델을 얻는 것은 아니다. 통계학자들은 모델 선택을 위한 지침으로 **오컴의 면도날**이라는 원리를 사용한다. 모든 것이 동일한 조건에서는, 복잡한 모델보다는 단순한 모델을 우선 사용해야 한다는 원리이다.

변수를 추가하면 항상 RMSE는 감소하고 R^2은 증가한다. 따라서 이렇게 추가하는 변수들은 모델 선택에 별로 도움이 되지 않는다. 1970년대 일본의 저명한 통계 전문가, 아카이케 히로쓰구는 모델에 항을 추가할수록 불이익을 주는 **AIC(Akaike's information criteria)** 라는 측정 기준을 개발했다.

회귀분석의 경우 AIC는 다음과 같은 형식을 취한다.

$$AIC = 2P + nlog(RSS/n)$$

여기서 P는 변수의 개수이고, n은 레코드의 개수이다. 당연히 목표는 AIC를 최소화하는 모델을 찾는것이다. 모델에 k개의 변수를 추가한다면 2k 만큼 불이익을 받게 된다.

CAUTION_ AIC, BIC, 멜로즈 C_p

AIC에 대한 수식은 살짝 이상할 수 있다. 하지만 사실 이 수식은 정보이론에서 나온 점진적 결론에 의한 것이다. AIC의 몇 가지 변형들이 있다.

- **AICc : 크기가 작은 표본을 위해 수정된 AIC**
- **BIC(Bayesian information criteria) : AIC와 비슷하지만 변수 추가에 대해 더 강한 벌점을 주는 정보 기준**
- **멜로즈 C_p : 콜린 링우드 멜로즈가 제안한 AIC 변형**

데이터 과학자들은 보통 이와 같은 표본 내 측정 지표들 사이의 차이나 이들을 뒷받침하는 이론에 크게 걱정하지 않아도 된다.

AIC를 최소화하는 모델을 어떻게 찾을 수 있을까? 한 가지 방법은 **부분집합회귀**로서 모든 가능한 모델을 검색하는 방법이다. 이것은 계산 비용이 많이 들며, 대용량 데이터와 변수가 많은 문제에 적합하지 않다. 매력적인 대안은 **단계적 회귀**를 사용하는 것인데, 단계적 회귀분석을 사용하면 예측변수를 연속적으로 추가/삭제하여 AIC를 낮추는 모델을 찾을 수 있다. 베너블스와 리플리가 개발한 MASS 패키지는 stepAIC이라는 단계적 회귀 함수를 제공한다.

```
step_lm <- stepAIC(house_full, direction="both")

> step_lm

Call:
lm(formula = AdjSalePrice ~ SqFtTotLiving + Bathrooms + Bedrooms +
    BldgGrade + PropertyType + SqFtFinBasement + YrBuilt, data = house,
    na.action = na.omit)

Coefficients:
               (Intercept)              SqFtTotLiving
                 6.178e+06                  1.993e+02
                 Bathrooms                   Bedrooms
                 4.240e+04                 -5.197e+04
                 BldgGrade  PropertyTypeSingle Family
                 1.372e+05                  2.285e+04
    PropertyTypeTownhouse            SqFtFinBasement
                 8.438e+04                  7.032e+00
                   YrBuilt
```

```
                -3.565e+03
```

함수 실행 결과 house_full 에서 SqFtLot 등 4개의 변수들이 삭제된 모델을 선택했다.

더 단순한 방법으로는 **전진선택**과 **후진선택**이 있다. 전진선택에서는 예측변수 없이 시작하여 각 단계에서 R^2에 가장 큰 기여도를 갖는 예측변수를 하나씩 추가하고 기여도가 통계적으로 더 이상 유의미하지 않을 때 중지한다.

후진선택 또는 후진제거에서는 전체모델로 시작해서, 모든 예측변수가 통계적으로 유의미한 모델이 될 때까지, 통계적으로 유의하지 않은 예측변수들을 제거해나간다.

벌점회귀는 개념적으로 AIC와 같다. 개별 모델 집합들을 명시적으로 검색하는 대신 모델 적합 방정식에 많은 변수(파라미터)에 대해 모델에게 불이익을 주는 제약 조건을 추가한다. 단계적, 전진선택, 후진선택처럼 예측변수를 완전히 제거하는 대신,

벌점회귀에서는 계수 크기를 감소시키거나 경우에 따라 거의 0으로 만들어 벌점을 적용한다. 많이 사용되는 벌점회귀 방법으로는 **능형회귀**와 **라소**가 있다.

단계적 회귀분석과 모든 부분집합회귀는 모델을 평가하고 조정하는 **표본 내** 방법이다. 따라서 선택된 모델이 과적합(오버피팅) 될 수 있으며, 새 데이털르 적용할 때 잘 맞지 않을 수도 있다는 의미다.

이를 방지하기 위한 공통적인 접근법 중 하나는 교차타당성검사를 통해 모델의 유효성을 알아보는 것이다. 선형회귀분석에서는 데이터에 주어진 모델이 단순한 (선형) 전역 구조를 갖고 있기에, 일반로 과적합 문제가 크게 나타나지 않는다. 더 복잡한 모델 유형, **특히 좁은 영역의 데이터 구조에 반응하는 반복적인 절차의 경우 교차타당성검사는 매우 중요한 도구이다.**

4.2.5 가중회귀

통계학자들은 다양한 목적으로 가중회귀를 사용한다. 특히 복잡한 설문 분석에 중요하다. 데이터 과학자들은 아래 두 가지의 장점에서 가중회귀의 유용성을 발견할 수 있다.

- **서로다른 관측치를 다른 정밀도로 측정했을 때, 역분산 가중치를 얻을 수 있다.**
- **가중치 변수가 집계된 데이터의 각 행이 나타내는 원본 관측치의 수를 인코딩하도록 집계된 형식의 데이터를 분석할 수 있다.**

예를 들면 주택 가격 데이터의 경우, 오래된 매매 정보일수록 최근 정보보다는 신뢰하기 어렵다. 다음과 같이 DocumentDate 값을 이용해서 2005년 (데이터 수집을 시작한) 이래 지난 연수를 가중치로 사용할 수 있다.

```
library(lubridate)
house$Year = year(house$DocumentDate)
house$Weight = house$Year - 2005
```

다음 예제와 같이 lm함수에서 weight 인수를 사용하면 가중회귀를 계산할 수 있다.

```
> house_wt <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
+                  Bedrooms + BldgGrade,
+                data=house, weight=Weight, na.action=na.omit)
> round(cbind(house_lm=house_lm$coefficients,
+             house_wt=house_wt$coefficients), digits=3)
                house_lm    house_wt
(Intercept)   -521924.722 -584265.244
SqFtTotLiving     228.832     245.017
SqFtLot            -0.061      -0.292
Bathrooms       -19438.099  -26079.171
Bedrooms        -47781.153  -53625.404
BldgGrade       106117.210  115259.026
```

가중회귀의 계수를 보면 기존 회귀분석의 결과와 살짝 다른 것을 알 수 있다.

주요 개념

- **다중선형회귀 모델은 한 응답변수 Y와 여러 개의 예측변수 ($X_1, \cdots, X_p$) 간의 관계를 나타낸다.**
- **모델을 평가하는 가장 중요한 지표는 제곱근 평균제곱오차(RMSE)와 R제곱(R^2) 이다.**
- **계수들의 표준오차는 모델에 대해 변수 기여도의 신뢰도를 측정하는 데 사용된다.**
- **단계적 회귀는 모델을 만드는 데 필요한 변수들을 자동으로 결정하는 방법이다 .**
- **가중회귀는 방정식을 피팅할 때, 레코드 별로 가중치를 주기 위해 사용한다.**

4.3 회귀를 이용한 예측

데이터 과학에서 주된 목적은 예측이다. 기존의 오랫동안 자리 잡은 전통적인 의미의 통계학에서, 회귀는 예측보다는 설명을 위한 모델링에 더 적합했다는 점을 눈여겨 볼 필요가 있다.

용어정리

- **예측구간 : 개별 예측값 주위의 불확실한 구간**
- **외삽법 : 모델링에 사용한 데이터 범위를 벗어난 부분까지 모델을 확장하는 것**

4.3.1 외삽의 위험

회귀모형을 데이터 범위를 초과하면서까지 외삽하는 데 사용해서는 안된다. 회귀모형은 충분한 데이터 값이 있는 예측변수에 대해서만 유효하다(충분한 데이터가 있다 하더라도 다른 문제가 있을 수 있다.) 극단적으로 model_lm을 가지고 5,000제곱피트의 공터 가격을 예측하는 데 사용한다고 하자. 이때 건물과 관련된 모든 예측변수의 값은 0이 되고, 회귀방정식의 결과는 -521,900 + 5,000x = -522,202달러 라는 황당한 예측 결과가 나온다. 데이터에는 건물이 있는 구획만 포함되어 있다. 빈 땅에 해당하는 레코드는 없다. 결과적으로 이 모델에는 고어 가격을 예측하는 방법을 알려줄 정보가 없다.

4.3.2 신뢰구간과 예측구간

통계학은 변동성(불확실성)을 이해하고 측정하는 것을 포함한다. 회귀분석 결과에 나오는 t퉁계량과 p값은 이 변동성을 다루는 아주 일반적인 방법이고, 종종 변수 선택을 위해 활용된다. 이외에 더 유용한 지표로 회귀계수와 예측을 둘러싼 불확실성 구간을 의미하는 신뢰구간이 있다. 부트스트랩을 통해 이것을 쉽게 이해할 수 있다. SW출력에서 가장 일반적으로 마주치는 회귀 관련 신뢰구간은 회귀 파라미터(계수)의 신뢰구간이다. 다음은 P개의 예측변수와 n개의 레코드(행)가 있는 데이터에 대해, 회귀 파라미터(계수)에 대한 신뢰구간을 생성하기 위한 부트스트랩 알고리즘이다.

1. **각 행(결과변수를 포함)을 하나의 티켓으로 생각하고 개수가 모두 n개인 티켓을 박스에 넣었다고 가정하자.**
2. **무작위로 티켓을 뽑아 값을 기록하고 다시 박스에 넣는다.**
3. **2번 과정을 n번 반복한다. 이를 통해, 부트스트랩 재표본을 하나 만든다.**
4. **이 부트스트랩 표본을 가지고 회귀모형을 구한다. 그리고 추정된 계수들을 기록한다.**
5. **2~4번 과정을 1,000번 반복한다.**
6. **이제 각 계수에 대해 1,000개의 부트스트랩 값을 갖게 된다. 각각에 대해 적합한 백분위 수를 구한다. (90% 신뢰구간을 위해 5번째에서 95번째 백분위수를 구한다).**

R의 Boot 함수를 사용하면 계수에 대한 실제 부트스트랩 신뢰구간을 구하거나 일반적인 R의 출력 결과같이 신뢰구간을 간단하게 구할 수 있다. 데이터 과학자들은 회귀계수가 얼마인지에 관심이 있지 이들에 대한 개념적 의미와 해석에는 별로 관심이 없다. 데이터 과학자들이 더 큰 관심이 있는 것은 **예측된 y값($\hat{Y}_i$)의 구간**이다. 이 값의 불확실성은 아래 두 가지 원인에서 비롯된다.

- **무엇이 적합한 예측변수인지, 그리고 계수가 얼마인지에 따른 불확실성(부트스트랩 알고리즘 참고)**
- **개별 데이터 값에 존재하는 추가적인 오류**

개별 데이터 값의 오차는 다음과 같이 생각할 수 있다. 회귀방정식이 무엇인지 정확히 알았다하더라도 (예를 들어 엄청난 수의 데이터가 있어서) , 주어진 예측변수 값에 대한 **실제** 결괏값은 달라질 수 있다. 예를 들면 방이 8개, 실 평수 6,500제곱피트, 욕실이 3개, 그리고 지하실까지 있는 주택들이 여러 채 있다고 했을 때, 이들의 값은 서로 다를 수 있다. 우리는 이 개별 오차를 적합값으로부터의 잔차로 모델링할 수 있다. 회귀모형에 따른 오차와 개별 데이터 값에 따른 오차를 모두 모델링하기 위한 부트스트랩 알고리즘은 다음과 같다.

1. 데이터로부터 부트스트랩 표본을 뽑는다.
2. 회귀모형을 찾고 새로운 값을 예측한다.
3. 원래의 회귀 적합도에서 임의로 하나의 잔차를 취하여 예측값에 더하고 그 결과를 기록한다.
4. 1~3단계를 1,000번 반복한다.
5. 결과의 2.5번째 백분위수와 97.5번째 백분위수를 찾는다.

CAUTION_ 예측구간이냐, 신뢰구간이냐?

예측구간은 하나의 값에 대한 불확실성과 관련되는 반면, 신뢰구간은 여러 값에서 계산된 평균이나 다른 통계량과 관련된다. 따라서 예측구간은 일반적으로 같은 값에 대해 신뢰구간보다 훨씬 넓다. 부트스트랩 모델에서 예측값에 추가한 개별 잔차를 선택하는 방식으로 이 개별 값의 오차를 모델링한다. 어느 것을 사용해야 할까? 이는 상황과 분석 목적에 따라 다르지만 일반적으로 데이터 과학자는 특정 개별 예측에 관심이 있으므로 예측구간이 더 적절할 수 있다. 예측구간을 사용해야 하는 대신 신뢰구간을 사용하면 주어진 예측값의 불확실성이 지나치게 낮은 것으로 나올 수 있다.

주요개념

- *데이터 범위를 벗어나는 외삽은 오류를 유발할 수 있다.*
- *신뢰구간은 회귀계수 주변의 불확실성을 정량화 한다.*
- *예측구간은 개별 예측값의 불확실성을 정량화한다.*
- ***R을 포함한 대부분의 SW는 수식을 사용하여 예측/신뢰구간을 기본 또는 지정된 출력으로 생성한다.***
- *수식 대신 부트스트랩을 사용할 수도 있다. 해석과 개념은 같다.*

4.4 회귀에서의 요인변수

범주형 변수라고도 불리는 **요인변수(factor variable)**는 개수가 제한된 이산값을 취한다. 예를 들면 대출목적이라는 변수는 '부채 정리', '결혼' , '자동차' 등의 값을 가질 수 있다. **지표변수(indicator variable)** 라고도 불리는 이진변수(예/아니오)는 요인변수의 특수한 경우이다. 회귀분석에는 수치 입력이 필요하기 때문에, 모델에 사용할 수 있도록 요인변수를 다시 수치화 해야 한다. 이를 위한 가장 일반적인 방법은 변수를 이진 **가변수** 들의 집합으로 변환하는 것이다.

용어정리

가변수(dummy variable) : 회귀나 다른 모델에서 요인 데이터를 사용하기 위해 0과 1의 이진변수로 부호화한 변수

기준 부호화(reference coding) : 통계학자들이 많이 사용하는 부호화 형태. 여기서 한 요인을 기준으로 하고 다른 요인들이 이 기준에 따라 비교할 수 있도록 한다. (유의어 : 처리 부호화(treatment coding))

원-핫 인코딩(one-hot-encoding) : 머신러닝 분야에서 많이 사용되는 부호화. 모든 요인 수준이 계속 유지된다. 어떤 머신러닝 알고리즘에서는 유용한 반면, 다중선형회귀에는 적합하지 않다.

편차 부호화(deviation coding) : 기준 수준과는 반대로 전체 평균에 대해 각 수준을 비교하는 부호화 방법(유의어 : 총합 대비(sum contrast))

4.4.1 가변수 표현

킹 카운티 주택 가격 데이터에서 주거 형태에 관한 요인변수를 찾아볼 수 있다. 그 가운데 일부 6개를 출력하면 다음과 같다.

```
> head(house[, 'PropertyType'])
[1] Multiplex     Single Family
[3] Single Family Single Family
[5] Single Family Townhouse
3 Levels: Multiplex ...
```

이때 가능한 값은 Multiplex, Single Family, Townhouse 세 가지다. 이 요인변수를 사용하기 위해 이진변수들로 변환해보겠다. 요인변수에서 각각의 가능한 값에 대해 이진변수를 만들어야 한다. R의 model.matrix 함수를 통해 다음과 같이 구현할 수 있다.

```
> prop_type_dummies <- model.matrix(~PropertyType-1 , data=house)
> head(prop_type_dummies)
  PropertyTypeMultiplex PropertyTypeSingle Family PropertyTypeTownhouse
1                     1                         0                     0
2                     0                         1                     0
3                     0                         1                     0
4                     0                         1                     0
5                     0                         1                     0
6                     0                         0                     1
```

model.matrix 함수는 데이터 프레임 객체를 선형모형에 적합한 행렬 형태로 변환한다.

결과를 보면 3가지 수준을 갖는 요인변수 Property Type이 3개의 열을 갖는 행렬로 표현되었다. 머신러닝 커뮤니티에서는 이런 식의 표현법을 **원-핫 인코딩** 이라고 부른다. 최근접 이웃 알고리즘이나 트리 모델 같은 머신러닝 알고리즘에서, 요인변수를 표현하는 데 원-핫 인코딩을 많이 사용한다.

회귀분석에서 P개의 개별 수준을 갖는 요인변수는 보통 $P-1$ 개의 열을 갖는 행렬로 표시된다. 회귀모형에 일반적으로 절편이 포함되기 때문이다. 절편이 있기 때문에 $P-1$개의 이진변수 값을 정의하고 나면, P 번째 값을 알 수 있고, 따라서 P번째 값까지 넣게 되면 이러한 중복성이 문제가 될 수 있다. 다시말해 P번째 열을 추가하면 다중공선성 오류가 발생할 수 있다.

R의 기본 표현법은 첫 번째 요인 수준을 **기준**으로 하고, 나머지 수준을 이 기준에 상대적인 것으로 해석한다.

cf) model.matrix 함수에서 -1 이라는 인수는 원-핫 인코딩 표시법을 사용한다는 의미이다. (- 기호는 절편을 삭제한다는 것을 뜻한다.) 이렇게 하지 않으면 기본 설정대로, 첫 요인을 기준으로 삼는 P-1 개 열로 이뤄진 행렬을 출력한다.

```
> ## Code snippet 4.13
> lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
+       Bedrooms +  BldgGrade + PropertyType, data=house)

Call:
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
    Bedrooms + BldgGrade + PropertyType, data = house)

Coefficients:
               (Intercept)              SqFtTotLiving
                -4.469e+05                 2.234e+02
                   SqFtLot                 Bathrooms
                -7.041e-02                -1.597e+04
```

```
                 Bedrooms                  BldgGrade
                -5.090e+04                  1.094e+05
PropertyTypeSingle Family      PropertyTypeTownhouse
                -8.469e+04                 -1.151e+05
```

R 회귀분석 결과에서 두 PropertyType에 대응하는 두 계수 PropertyTypeSingle Family와 PropertyTypeTownhouse 를 볼 수 있다. PropertyTypeSingle Family == 0 이고 PropertyTypeTownhouse == 0 일 때, **Multiplex는 암묵적으로 정의되기 때문에 Multiplex 에 대한 계수를 따로 설정할 필요는 없다.** 이 계수들은 Multiplex에 상대적인 값이기 때문에, 결과적으로 Single Family인 주택은 거의 $85,000 정도 가치가 더 낮고, Townhouse는 $150,000 정도 가치가 더 낮다.

*CAUTION_ **그 밖의 요인변수 코딩법들***

*요인변수를 인코딩하는 데는 **대비 부호화** 시스템이라는 여러 다른 방법이 있다. 예를 들어 **총합 대비**라고도 하는 **편차 부호화** 방법은 각 수준을 전반적인 평균과 비교한다. 또 다른 대비 방법은 **다항식 부호화**이며, 이는 순서가 있는 요인변수에 적합하다. 이런 순서가 있는 요인변수를 제외하면, 데이터 과학자는 일반적으로 기준 코딩이나 원-핫 인코딩 이외에 다른 유형의 코딩을 접할 일이 거의 없다.*

4.4.2 다수의 수준을 갖는 요인변수들

어떤 요인변수는 가능한 수준의 수가 많아, 많은 양의 이진 더미를 생성할 수 있다. 예를 들면 우편번호는 요인변수이며 미국에는 43,000개의 우편번호가 있다.

이러한 경우 데이터와 예측변수와 결과 간의 관계를 탐색하여 유용한 정보가 범주에 포함되는지 여부를 판단해야 한다. 그렇다면 모든 요소를 유지하는 것이 좋을지 아니면, 수준을 통합하는 것이 나을 지를 결정해야 한다.

킹 카운티의 주택 매매 데이터에는 82개의 우편번호가 있다.

```
> table(house$ZipCode)

 9800 89118 98001 98002 98003 98004 98005 98006 98007 98008 98010 98011 98014
    1     1   358   180   241   293   133   460   112   291    56   163    85
98019 98022 98023 98024 98027 98028 98029 98030 98031 98032 98033 98034 98038
  242   188   455    31   366   252   475   263   308   121   517   575   788
98039 98040 98042 98043 98045 98047 98050 98051 98052 98053 98055 98056 98057
   47   244   641     1   222    48     7    32   614   499   332   402     4
98058 98059 98065 98068 98070 98072 98074 98075 98077 98092 98102 98103 98105
  420   513   430     1    89   245   502   388   204   289   106   671   313
98106 98107 98108 98109 98112 98113 98115 98116 98117 98118 98119 98122 98125
  361   296   155   149   357     1   620   364   619   492   260   380   409
98126 98133 98136 98144 98146 98148 98155 98166 98168 98177 98178 98188 98198
  473   465   310   332   287    40   358   193   332   216   266   101   225
98199 98224 98288 98354
  393     3     4     9
```

Zipcode는 주택 가격에 대한 위치의 효과를 볼 수 있는 중요한 변수이다. 다만 모든 수준을 포함하려면 자유도 81에 해당하는 81개의 계수가 필요하다. 원래 모델인 house_lm은 자유도가 단지 5에 불과했다. 또한 우편번호 중 몇몇은 매물이 하나뿐인 경우도 있다. 어떤 상황에서는, 대도시 지역의 지리적 위치에 대응되는 처음 두 자리 또는 세자리 만을 사용하여 우편번호를 통합할 수 있다. 하지만 킹 카운티의 경우 거의 모든 판매가 980xx 또는 981xx에서 이뤄지기에 이런 방법은 큰 도움이 안된다.

대안은 매매 가격과 같은 변수에 따라 우편번호로 그룹을 짓는 것이다. 아니면 초기 모델의 잔차를 사용하여 우편번호 그룹을 만드는 방법도 좋다. 다음 dplyr 코드는 82개의 우편번호를 house_lm 회귀 결과의 잔차값의 중간값을 기준으로 5개의 그룹으로 통합한다.

```
## Code snippet 4.15
zip_groups <- house %>%
  mutate(resid = residuals(house_lm)) %>%
  group_by(ZipCode) %>%
  summarize(med_resid = median(resid),
            cnt = n()) %>%
  # sort the zip codes by the median residual
  arrange(med_resid) %>%
  mutate(cum_cnt = cumsum(cnt),
         ZipGroup = factor(ntile(cum_cnt, 5)))
house <- house %>%
  left_join(select(zip_groups, ZipCode, ZipGroup), by='ZipCode')
```

각 우편번호에 대해 잔차의 중간값을 계산하고, ntile 함수를 사용해 중간값으로 정렬한 우편번호를 5개의 그룹으로 분할한다. 회귀분석에서 원래 피팅 결과를 개선하기 위해 이것을 어떻게 사용하는지 예를 보려면 4.5.3절을 참고하자.

회귀 적합화를 돕는데 잔차를 사용한다는 개념은 모델링 프로세스의 기본 단계다. 이에 대한 내용은 4.6절을 참고하자.

4.4.3 순서가 있는 요인변수

일부 요인변수는 요인의 수준이 순서를 갖는다. 이것을 **순서 요인변수 (ordered factor variable)** 또는 **순서 범주형 변수(ordered categorical variable)** 라고 한다. 예를 들면 대출 등급은 A,B,C 등이 될 수 있다. 각 등급은 이전 등급보다 위험이 더 크다. 순서 요인변수는 일반적인 숫자 값으로 변환하여 그대로 사용할 수 있다. 예를 들면 변수 BldgGrade는 순서 요인변수이다. 등급 유형 중 한 예를 다음 표에서 볼 수 있다. 등급이 특별한 의미를 가지는 반면, 수치형 변수는 낮은 값에서 높은 값으로 정렬이 가능하다.

값	설명
1	오두막
2	기준 미달
5	적당함
10	매우 좋음
12	호화로움
13	대저택

순서 요인변수를 수치형 변수로 다루는 것은, 그냥 요인변수로 다루면 잃어버릴 수 있는 순서에 담긴 정보를 유지하기 위함이다.

주요 개념

- **요인변수는 회귀를 위해 수치형 변수로 변환해야 한다.**
- **요인변수를 P개의 개별 값으로 인코딩하기 위한 가장 흔한 방법은 P-1개의 가변수를 만들어 사용하는 것이다.**
- **다수의 수준을 갖는 요인변수의 경우, 더 적은 수의 수준을 갖는 변수가 되도록 수준들을 통합해야 한다.**
- **순서를 갖는 요인변수의 경우, 수치형 변수로 변환하여 사용할 수 있다.**

4.5 회귀방정식의 해석

다시 말하지만 데이터 과학에서 회귀의 가장 중요한 용도는 일부 종속변수(결과변수)를 예측하는 것이다. 하지만 때로는 예측변수와 결과 간 관계의 본질을 이해하기 위해 방정식 자체로부터 통찰을 얻는 것이 중요할 때도 있다. 이 섹션에서는 회귀방정식을 검토하고 해석하는 방법에 대한 지침을 제공한다.

용어 정리

- **변수 간 상관 (correlated variable) : 예측변수들끼리 서로 높은 상관성을 갖을 때, 개별 계수를 해석하는 것은 어렵다.**
- **다중공선성(multicollinearity) : 예측변수들이 완벽하거나 거의 완벽에 가까운 상관성을 갖는다고 할 때, 회귀는 불안정하며 계산도 불가능 하다.(유의어 : 공선성(collinearity))**
- **교란변수(confounding variable) : 중요한 예측변수이지만 회귀방정식에 누락되어 결과를 잘못되게 끄는 변수**
- **주효과(main effect) : 다른 변수들과 독립된 하나의 예측변수와 결과변수 사이의 결과**
- **상호작용(interaction) : 둘 이상의 예측변수와 응답변수 사이의 상호 의존적인 관계**

4.5.1 예측변수 간 상관

다중회귀분석에서 예측변수는 종종 서로 상관성이 있다. 한 예로 4.2.4절에서 구한 회귀모형 step_lm 의 회귀계수를 검토해 보자.

```
step_lm$coefficients
> step_lm$coefficients
               (Intercept)             SqFtTotLiving                 Bathrooms
              6.177658e+06              1.992747e+02              4.240306e+04
                  Bedrooms                 BldgGrade PropertyTypeSingle Family
             -5.197477e+04              1.371811e+05              2.285488e+04
    PropertyTypeTownhouse           SqFtFinBasement                   YrBuilt
              8.437893e+04              7.032179e+00             -3.564935e+03
```

여기서 침실 개수를 뜻하는 Bedrooms의 계수가 음수인 것을 볼 수 있다. 즉 침실 개수를 늘릴수록 그 가치가 감소한다는 것을 의미한다. 어떻게 이럴 수 있을까? 이는 예측변수들이 서로 연관되어 있기 때문이다. 집이 클수록 침실이 더 많은 경향이 있으며, 침실 수보다는 주택의 크기가 주택 가격에 더 큰 영향을 준다. 똑같은 크기의 두 집이 있다고 하면, 작은 크기의 실이 여러 개 있는 것을 선호하는 것이 합리적이다.

이렇게 상호 연관된 예측변수들을 사용하면 회귀계수의 부호와 값의 의미를 해석하기가 어려울 수 있다(또한 추정치의 표준오차가 커진다). 침실 수, 평수, 욕실 수에 대한 변수들은 모두 상관관계가 있다. 방정식에서 SqFtTotLiving, SqFtFinBasement, Bathrooms를 제거한 후 얻은 회귀모형을 통해 이것을 설명할 수 있다.

```
> update(step_lm, . ~ . -SqFtTotLiving - SqFtFinBasement - Bathrooms)

Call:
lm(formula = AdjSalePrice ~ Bedrooms + BldgGrade + PropertyType +
    YrBuilt, data = house, na.action = na.omit)

Coefficients:
               (Intercept)                   Bedrooms
                   4913025                      27136
                 BldgGrade  PropertyTypeSingle Family
                    249019                     -19932
    PropertyTypeTownhouse                    YrBuilt
                    -47424                      -3211
```

update 함수를 사용해서, 모델의 변수를 추가하거나 제외할 수 있다. 이제 침실 수에 대한 계수가 우리가 기대했던 대로 양수인 것을 확인할 수 있다(침실 수가 실제 주택 크기에 대한 대리변수로 작용했지만, 이제는 해당 변수가 제거된 상태이다).

변수 간 상관관계는 회귀계수를 해석할 때 고려해야 할 여러 문제들 가운데 한 가지일 뿐이다.

모델 house_lm에는 주택의 위치를 고려할 변수가 따로 없는 상태에서, 서로 다른 유형의 지역들 정보가 섞여 있다. 이 경우, 위치 정보는 **교란변수** 일 수 있다.

4.5.2 다중공선성

변수 상관의 극단적인 경우 다중공선성이 나타난다. 이는 예측변수 사이의 중복성을 판단하는 조건이 된다. 완전 다중공선성은 한 예측변수가 다른 변수들의 선형결합으로 표현된다는 것을 의미한다. 다중공선성은 다음 경우 발생한다.

- **오류로 인해 한 변수가 여러 번 포함된 경우**
- **요인변수로부터 P-1개가 아닌 P의 가변수가 만들어진 경우**
- **두 변수가 서로 거의 완벽하게 상관성이 있는 있는 경우**

회귀분석에서는 다중공선성 문제를 반드시 해결해야 한다. 다중공선성이 사자딜 때까지 변수를 제거해야 한다. 완전 다중공선성이 존재하는 상황에서는 회귀를 통해 제대로 된 답을 얻을 수가 없다. R을 포함한 많은 SW패키지는 특정 유형의 다중공선성 문제를 자동으로 처리한다. 예를 들면 SqFtTotLiving을 주택 가격 데이터의 회귀에 두 번 포함하더라도 결과는 기존 house_lm 모델의 경우와 동일하다. 하지만 불완전 다중공선성의 경우, SW를 통해 답을 얻을 수는 있지만 결과가 불안정할 수 있다.

NOTE 다중공선성은 트리, 클러스터링, 최근접 이웃 알고리즘 등 회귀 유형이 아닌 방법에서는 그다지 문제가 되지 않으며, 이들 방법에서는 P-1 대신에 P개의 가변수를 유지하는 것이 좋다. 물론 이러한 방법에서도 예측변수의 비중복성은 유지하는 것은 당연하다.

4.5.3 교란변수

변수 상관은 응답변수와 비슷한 예측 관계를 갖는 다른 변수가 포함되는 바람에 비롯된 문제인 반면, **교란변수**는 회귀방정식에 중요한 변수가 포함되지 못해서 생기는 누락의 문제이다. 이 경우 방정식 계수에 대한 순진한 해석은 잘못된 결론으로 이어질 수 있다.

예를 들면 4.2.1절에서 킹 카운티 관련 데이털르 통해 얻은 회귀모형 house_lm을 다시 생각해 보자. SqFtLot, Bathrooms, Bedrooms의 회귀계수는 모두 음수 였다. 원래 회귀모형에는 주택 가격에 아주 결정적인, 위치를 나타내는 변수가 포함되어 있지 않았다.

```
> ## Code snippet 4.4
> house_lm <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
+                  Bedrooms + BldgGrade,
+                data=house, na.action=na.omit)
> ## Code snippet 4.5
> house_lm

Call:
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
    Bedrooms + BldgGrade, data = house, na.action = na.omit)

Coefficients:
  (Intercept)  SqFtTotLiving        SqFtLot      Bathrooms       Bedrooms
   -5.219e+05      2.288e+02     -6.051e-02     -1.944e+04     -4.778e+04
    BldgGrade
    1.061e+05
```

위치 정보를 고려하기 위해, 우편번호를 가장 싼 지역(1)에서 가장 비싼 지역(5)까지 5개의 그룹으로 분류하는 새로운 변수 ZipGroup를 포함해보자.

```
> lm(AdjSalePrice ~  SqFtTotLiving + SqFtLot +
+      Bathrooms + Bedrooms +
+      BldgGrade + PropertyType + ZipGroup,
+    data=house, na.action=na.omit)

Call:
lm(formula = AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
    Bedrooms + BldgGrade + PropertyType + ZipGroup, data = house,
    na.action = na.omit)

Coefficients:
               (Intercept)              SqFtTotLiving
                 -6.709e+05                  2.112e+02
                    SqFtLot                  Bathrooms
                  4.692e-01                  5.537e+03
                   Bedrooms                  BldgGrade
                 -4.139e+04                  9.893e+04
PropertyTypeSingle Family      PropertyTypeTownhouse
                  2.113e+04                 -7.741e+04
                  ZipGroup2                  ZipGroup3
                  5.169e+04                  1.142e+05
                  ZipGroup4                  ZipGroup5
                  1.783e+05                  3.391e+05
```

ZipGrou이 분명히 중요한 변수라는 것을 알 수 있다. 가장 비싼 우편번호 그룹의 주택가격이 약 340,000달러나 더 높았다. SqFtLot과 Bathrooms의 계수는 이제 양수이며, 욕실을 하나 추가하면 판매 가격이 5,537달러 정도 증가한다.

Bedrooms에 대한 계수는 여전히 음수이다. 이 결과가 그렇게 직관적이지는 않지만, 이것은 부동산 업계에서 잘 알려진 사실이다. 살기 좋은 지역에서는 욕실 수가 같은 주택의 경우 작은 침실이 여러 개 있으면 오히려 값어치가 떨어진다.

4.5.4 상호작용과 주효과

통계학자는 **주효과**(독립변수)와 주효과 사이의 **상호작용**을 구별하기 좋아한다. 주효과는 회귀 방정식에서 종종 **예측변수**라고 불린다. 모델에서 주효과만 사용한다면 여기에는 예측변수와 응답변수 간의 관계가 다른 예측변수들에 대해 독립적이라는 암묵적인 가정이 있다. 하지만 이것은 종종 사실이 아니다.

예를 들면 4.5.3 절의 킹 카운티 주택 데이터에 적합한 모델은 ZipCode를 비롯한 여러 변수를 주효과로 포함하고 있다. 부동산에서 위치가 가장 중요하다는 사실은 모두 알고 있다. 주택 크기와 매매 가격 간의 관계가 위치에 달려 있다고 가정하는 것은 자연스러운 일이다. 임대료가 싼 지역에 지어진 큰 집은 비싼 지역에 지어진 큰 집과 같은 가치를 유지하기가 어려울 것이다. R에서 *연산자를 사용하면 모델에 변수의 상호작용을 포함시킬 수 있다. 다음은 킹 카운티 데이터를 모델링할 때, SqFtTotLiving 과 ZipGroup 간의 상호작용을 고려하는 예를 보여준다.

```
> #  Interactions
> ## Code snippet 4.18
> lm(AdjSalePrice ~  SqFtTotLiving*ZipGroup + SqFtLot +
+      Bathrooms + Bedrooms +
+      BldgGrade + PropertyType,
+    data=house, na.action=na.omit)
```

```
Call:
lm(formula = AdjSalePrice ~ SqFtTotLiving * ZipGroup + SqFtLot +
    Bathrooms + Bedrooms + BldgGrade + PropertyType, data = house,
    na.action = na.omit)

Coefficients:
               (Intercept)              SqFtTotLiving
                -4.919e+05                  1.176e+02
                  ZipGroup2                  ZipGroup3
                -1.342e+04                  2.254e+04
                  ZipGroup4                  ZipGroup5
                 1.776e+04                 -1.555e+05
                    SqFtLot                  Bathrooms
                 7.176e-01                 -5.130e+03
                   Bedrooms                  BldgGrade
                -4.181e+04                  1.053e+05
PropertyTypeSingle Family      PropertyTypeTownhouse
                 1.603e+04                 -5.629e+04
  SqFtTotLiving:ZipGroup2    SqFtTotLiving:ZipGroup3
                 3.165e+01                  3.893e+01
  SqFtTotLiving:ZipGroup4    SqFtTotLiving:ZipGroup5
                 7.051e+01                  2.298e+02
```

이 결과에서는 SqFtTotLiving:ZipGroup3, SqFtTotLiving:ZipGroup4 등의 4가지 새로운 정보들을 볼 수 있다.

주택의 위치와 크기는 강한 상호작용이 있는 것으로 보인다. 가격대가 가장 낮은 ZipGroup에서 집에 대한 기울기는 제곱피트당 118달러로, 주효과 SqFtTotLiving 에 해당하는 계수와 같다(이는 R이 요인변수에 대해 **기준 부호화** 사용하고 있기 때문이다.). 가장 비싼 ZipGroup에 대한 계수는 이 주효과 계수에 SqFtTotLiving:ZipGroup5의 경우를 더한 합 즉 118+230=348달러와 같다. 다시 말해, 가격대가 가장 비싼 지역에서는 주택의 크기가 1제곱피트 늘어날 때 가장 낮은 지역에 비해 예상 매매가가 거의 3배가 차이난다.

TIP 상호작용 항들을 이용한 모델 선택

다수의 변수가 존재하는 문제의 경우, 모델에서 어떤 상호작용을 고려해야 할지 결정하기가 매우 어렵다. 이러한 문제에 접근하는 몇 가지 방법에 대해 알아보자.

- *어떤 문제에서는 사전 지식이나 직관이 일헌 상호작용을 결정하는 데 큰 도움이 된다.*
- *단계적 선택(4.2.4절 참고)을 사용해서 다양한 모델들을 걸러낼 수 있다.*
- *벌점을 부여하는 방식의 회귀 방법을 사용하여 자동으로 가능한 상호작용들을 최대한 가려내도록 한다.*
- *아마도 **랜덤 포레스트**나 **그레이디언트 부스팅 트리** 같은 **트리 모델**을 가장 일반적으로 사용하지 않을까 생각한다. 이러한 모델들은 자동으로 최적의 상호작용 항들을 걸러낸다.*

주요개념

- **예측변수들 사이의 상관성 때문에, 다중선형회귀에서 계수들을 해석할 때는 주의해야 한다.**
- **다중공선성은 회귀방정식을 피팅할 때, 수치 불안정성을 유발할 수 있다.**
- **교란변수란 모델에서 생략된 중요한 예측변수를 의미하며, 이에 따라 실제로 관계가 없는데 허위로 있는 것처럼 회귀 결과가 나올 수 있다.**
- **변수와 결과가 서로 의존적일 때, 두 변수 사이의 상호작용을 고려할 필요가 있다.**

4.6 가정검정 : 회귀 진단

설명을 위한 모델링에서는 (즉, 연구 목적에서는) 앞서 설명한 여러 측정 지표들을 고려하여, 매 단계마다 모델이 데이터에 얼마나 적합한지를 평가한다. 대부분은 모델을 뒷받침하는 가정들을 검정할 수 있는 잔차 분석을 기본으로 한다. 이런 단계들을 직접적으로 예측 정확도를 다루는 것은 아니지만 예측 설정에 중요한 통찰을 줄 수 있다.

용어정리

- **표준화잔차 : 잔차를 표준오차로 나눈 값**
- **특잇값 : 나머지 데이터(혹은 예측값)와 멀리 떨어진 레코드 (혹은 출력값)**
- **영향값 : 있을 때와 없을 때 회귀방정식이 큰 차이를 보이는 값 혹은 레코드**
- **지렛대(레버리지) : 회귀식에 한 레코드가 미치는 영향력의 정도(유의어 : 햇 값)**
- **비정규 잔차 : 정규분포를 따르지 않는 잔차는 회귀분석의 요건을 무효로 만들 수 있다. 데이터 과학에서는 별로 중요하게 다뤄지지 않는다.**
- **이분산성 : 어떤 범위 내 출력값의 잔차가 매우 높은 분산을 보이는 경향 (어떤 예측변수를 회귀식이 놓치고 있다는 것을 의미한다.)**
- **편잔차그림 : 결과변수와 특정 예측변수 사이의 관계를 진단하는 그림(유의어 : 추가변수그림)**

4.6.1 특잇값

일반적으로, 소위 **특잇값**이라고 부르는 극단값은 대부분의 측정치에서 멀리 벗어난 값을 의미한다. 위치와 변이를 추정할 때 이미 살펴본 것처럼, 회귀모형에서도 동일한 문제를 야기할 수 있다. 회귀에서 특잇값은 실제 y값이 예측된 값에서 멀리 떨어져 있는 경우를 말한다. 잔차를 표준오차로 나눈 값을 **표준화잔차**라고 하는데 바로 이 값을 조사해서 특잇값을 발견할 수 있다.

특잇값을 정상값들과 구분하는 데에 대한 통계이론은 없다. 그보다, 어떤 관측값을 특잇값이라고 부르려면 다수 데이터로부터 얼마나 떨어져 있어야 하는지에 대한 (임의의) 경험칙이 존재한다. 예를 들어 상자그림에서 상자 경계선 바깥에 위치한 점들을 특잇값이라고 본다. 여기서 '너무 멀리 떨어져 있다'라는 말은 곧 '사분위범위의 1.5배 보다 더 바깥에 있다'는 말과 같다.

이와 비슷하게, 회귀에서는 표준화잔차가 특잇값을 검출하는 데 주로 사용된다. 표준화잔차는 곧 '회귀선으로부터 떨어진 정도를 표준오차 개수로 표현한 값'정도로 해석할 수 있다.

킹 카운티 주택 매매 데어터에서 우편번호가 98105인 지역의 데이터만 가직 회귀모형을 구해보자.

```
# outlier anaysis
## Code snippet 4.19
house_98105 <- house[house$ZipCode == 98105,]
lm_98105 <- lm(AdjSalePrice ~ SqFtTotLiving + SqFtLot + Bathrooms +
                Bedrooms + BldgGrade, data=house_98105)
```

다음과 같이 rstandard 함수를 이용해 표준화잔차를 구하고, order 함수를 통해 가장 작은 잔차의 위치를 얻을 수 있다.

```
> sresid <- rstandard(lm_98105)
> idx <- order(sresid, decreasing=FALSE)
> sresid[idx[1]]
    20431
-4.326732
> resid(lm_98105)[idx[1]]
    20431
-757753.6
```

결과를 보면 표준오차의 4배 이상이나 회귀식과 차이를 보이는데 이에 해당하는 추정치는 757,753달러이다. 이 특잇값에 해당하는 레코드는 다음과 같다.

```
> ## Code snippet 4.21
> house_98105[idx[1], c('AdjSalePrice', 'SqFtTotLiving', 'SqFtLot',
+                       'Bathrooms', 'Bedrooms', 'BldgGrade')]
      AdjSalePrice SqFtTotLiving SqFtLot Bathrooms Bedrooms BldgGrade
20431       119748          2900     776         3        6         7
```

이 경우, 레코드가 뭔가 잘못됐다는 것을 알 수 있다. 이 우편번호에 해당하는 지역에서 이 정도 평수라면 119,748달러 보다는 더 비싸야 정상이다. 이런 경우는 비정상적인 판매에 해당하며 회귀에 포함되서는 안된다. 특잇값은 '부주의한' 데이터 입력 또는 실수 (예를 들면 천 단위를 그냥 일 단위로 기재하는 경우) 같은 문제들 때문에 발생할 수 있다.

빅데이터의 경우, 새로운 데이터를 예측하기 위한 회귀분석에서 특잇값이 그렇게 문제가 되지 않는다. 그러나 특잇값을 찾는 것이 주목적인 특잇값 검출의 경우 이 값들이 매우 중요해진다. 또한 특잇값은 사기 사건이나 갑작스러운 사건 발생과도 관련이 있다. 따라서 특잇값을 발견하는 것은 아주 중요한 사업 가치가 될 수 있다.

4.6.2 영향값

회귀모형에서 제외됐을 때 모델에 중요한 변화를 가져오는 값을 **주영향관측값**이라고 한다. 회귀분석에서, 잔차가 크다고 해서 모두 이런 값이 되는 것은 아니다. 예를 들면 다음 그림의 회귀선을 보자. 실선은 모든 데이터를 고려한 회귀선에 해당하며 점선은 오른쪽 위의 의 한 점을 제거했을 때의 회귀선을 보여준다. 분명히, 그 데이터 값은 회귀 결과에 큰 영향을 미치지만, 원래 회귀에서 큰 특잇값으로 나타난 것은 아니다. 이 데이터 값은 회귀에 대한 높은 **레버리지**를 가진 것으로 볼 수 있다.

표준화잔차 외에도 통계학자들은 회귀분석에서 단일 레코드의 영향력을 결정하는 몇 가지 지표를 개발했다. 레버리지를 측정하는 일반적인 척도는 **햇 값**이다. 2(P+1)/n 이상의 값들은 레버리지가 높은 데이터 값을 나타낸다.

또 다른 측정 지표는 **쿡의 거리**이다. 이것은 레버리지와 잔차의 크기를 합쳐서 영향력을 판단한다.

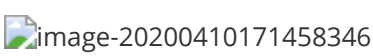
경험칙에 따르면 쿡의 거리가 4/(n-P-1) 보다 크면 영향력이 높다고 보는 편이다.

영향력 그림 또는 거품그림은 표준화잔차, 햇 값, 쿡의 거리를 모두 한 그림에 표현한다. 다음 그림은 킹 카운티 주택 데이터에 대한 영향력 그림을 보여준다. 다음과 같은 R코드를 이용해 그릴 수 있다.

```
std_resid <- rstandard(lm_98105)
cooks_D <- cooks.distance(lm_98105)
hat_values <- hatvalues(lm_98105)
plot(hat_values, std_resid, cex=10*sqrt(cooks_D))
abline(h=c(-2.5, 2.5), lty=2)
```

회귀에서 몇몇 데이터 포인트가 정말로 큰 영향력을 보임을 알 수 있다 . 쿡의 거리는 cooks.distance 함수를 사용해 계산하고 hatvalues 함수를 이용해 회귀 진단을 구할 수 있다. 햇 값은 x축, 잔차 정보는 y축에 위치하며, 쿡의 거리에 해당하는 값은 원의 크기로 나타낸다.

다음표는 전체 데이터를 모두 사용했을 때의 회귀 결과와 가장 영향력이 큰 데이터들을 제외하고 얻은 회귀 결과를 비교한다. Bathrooms 변수의 회귀계수가 엄청나게 변화하는 것을 볼 수 있다.



회귀모형을 구하는 목적이 새로 들어오는 값에 대해 믿을 만한 예측값을 얻기 위함이라면, 데이터의 크기가 작을 경우에만 영향력이 큰 관측 데이터를 확인하는 작업이 유용하다. 데이터 수가 클 경우, 어떤 한 값이 회귀방정식에 엄청난 변화를 가져오기란 쉽지 않다(회귀 결과 여전히 특잇값들이 존재한다 하더라도). 물론 이상 검출이 목적이라면 높은 영향력의 값들을 찾는 것이 큰 도움이 된다!!

4.6.3 이분산성, 비정규성, 오차 간 상관

통계학자들은 잔차 분포에 상당한 주의를 기울인다. 보통최소 추정은 다양한 분포 가정하에서 편향성도 없고 경우에 따라 '최적'이라고 할 수 있는 추정을 제공한다. 즉, 대부분의 문제에서 데이터 과학자라면 잔차 분포에 너무 많은 신경을 쓸 필요는 없다.

잔차의 분포는 주로 공식적인 통계적 추론 (가설검정 및 p값)의 유효성과 관련이 있으므로 예측 정확도를 중요하게 생각하는 데이터 과학자들에게는 별로 중요하지 않다. 형식적 추론이 완전히 유효하려면 잔차는

1) 동일한 분산을 가지며 (등분산 가정)

2) 정규분포를 따르고 (정규성 가정)

3)서로 독립이라는 가정 (독립성 가정) 이라고 부르며, 각 가정을 검정하는 잘 알려진 방법들이 존재한다.

데이터 과학자가 신경 쓰는 것 한가지는, 잔차에 대한 가정을 기반으로 예상 값에 대한 신뢰구간을 계산하는 방법이다. 3가지 가정 중 먼저 **이분산성**은 다양한 범위의 예측값에 따라 잔차의 분산이 일정하지 않은 것을 의미한다. 다시말해서, 어떤 일부분에서의 오차가 다른 데보다 훨씬 크게 나타나는 것을 말한다. ggplot2 패키지를 사용하면 잔차들의 분포를 쉽게 분석할 수 있다.

```
df <- data.frame(
  resid = residuals(lm_98105),
  pred = predict(lm_98105))
ggplot(df, aes(pred, abs(resid))) +
  geom_point() +
  geom_smooth()
```

위 그림은 코드의 결과이다. geom_smooth 함수를 이용해, 절대잔차들을 부드럽게 연결하는 선을 분포 그림 위에 쉽게 추가할 수 있다. 이렇게 산점도 에서 x축과 y축 변수들 사이의 관계를 부드럽게 연결하는 추세선을 얻기 위해 이 함수에서는 loess메서드를 호출한다.

(뒤에 나올 '산점도 평활기'노트 참고)

분명히 잔차의 분산은 고가의 주택일수록 증가하는 경향이 있다. 가격이 낮은 주택의 경우에도 마찬가지로 큰 편이다. 이를 통해 회귀모형 lm_98105는 **이분산성** 오차를 갖고 있다고 볼 수 있다.

TIP_ 데이터 과학자가 이분산성에 관심을 가져야 할까?

이분산성은 예측값의 어떤 경우에는 맞고 어떤 경우에는 틀리다는 것을 나타내며, 얻은 모델이 불완전하다는 것을 알려준다. 예를 들면 lm_98105의 이분산성은 회귀모형의 가격이 너무 높거나 주택에 대해서는 잘 설명하지 못한다는 것을 나타낸다.

두 번째로, 다음 그림은 lm_98105 회귀모형에서 표준화잔차에 대한 히스토그램이다. 정규분포보다 확연히 더 긴 꼬리를 가지며, 더 큰 잔차에 대해 약간의 왜곡을 보인다.

끝으로 통계학자들은 오차가 독립적이라는 가정을 점검하기도 한다. 시간에 따라 데이터를 수집하는 경우 특히 그렇다. 시계열 데이터를 다루는 회귀분석에서 유의미한 자기상관이 있는지를 탐지하는 데에는 **더빈-왓슨 통계량**을 사용할 수 있다.

회귀가 분포 가정 중 한가지만 위반해도 신경 써야 할끼? 데이터 과학에서 가장 중요한 것은 보통 예측 정확도이기에, 이분산성에 대한 검토는 그 다음으로 이어질 수 있다. 검토 결과 모델이 설명하지 못하는 데이터가 있음을 발견할 수 있을지도 모른다. 그러나 단순히 통계적 추론(p값, F 통계량)을 입증하기 위해서 분포 가정을 만족시키는 것은 데이터 과학자에게 그리 중요한 일이 아니다.

Note_ 산점도 평활기

회귀분석은 응답변수와 예측변수 간의 관계를 모델링하는 것이다. 회귀모형을 평가할 때는 두 변수 사이의 관계를 시각적으로 강조하기 위해 산점도 평활기를 사용하는 것이 좋다.

예를 들면 위 히스토그램에서 절대잔차와 예측값 간의 관계를 부드럽게 나타낸 곡선을 통해, 잔차의 분산이 잔차의 값에 의존한다는 것을 쉽게 알 수 있다. 이 경우 loess함수를 사용한다. loess함수는 일련의 구간별 지역 회귀모형을 구한 후 그것들을 연속적으로 부드럽게 만들어낸다(이를 평활화 라고 한다). loess가 가장 널리 사용되는 평활기일 것이다. 하지만 R에서는 슈퍼평활(supmu 함수)이나 커널 평활(smooth 함수)과 같이, 다른 산점도 평활기를 사용할 수도 있다. 회귀모형을 평가할 목적이라면, 일반적으로 이러한 산점도 평활기에 대한 구체적인 내용을 알 필요는 없다.

4.6.4 편잔차그림과 비선형성

편잔차그림은 예측 모델이 예측변수와 결과변수 간의 관계를 얼마나 잘 설명하는지 시각화하는 방법이다. **특잇점의 검출과 함께 이것은 데이터 과학자들에게 가장 중요한 모델 진단 방법이다.** 편잔차그림의 기본 개념은 하나의 예측변수와 응답변수 사이의 관계를 모든 다른 예측변수로부터 분리하는 것이다. 편잔차는 단일 예측변수를 기반으로 한 예측값과 전체를 고려한 회귀식의 실제 잔차를 결합하여 '만든 결과'라고 할 수 있다. 예측변수 X_i 의 편잔차는 일반 잔차에 X_i 와 연관된 회귀 항을 더한 값이다.

편잔차 = 잔차 + $\hat{b}_i X_i$

여기서 $\hat{b}_i$ 는 물론 회귀계수의 추정치를 의미한다. R의 predict 함수를 이용해서 개별 회귀 항 $\hat{b}_i X_i$ 를 얻을 수 있다.

```
terms <- predict(lm_98105, type='terms')
partial_resid <- resid(lm_98105) + terms
```

편잔차그림에서 x축은 X_i 를, 그리고 y축은 편잔차를 의미한다. ggplot2 를 사용하면 편잔차 분포 위에 평활 곡선을 손쉽게 추가할 수 있다.

```
df <- data.frame(SqFtTotLiving = house_98105[, 'SqFtTotLiving'],
                 Terms = terms[, 'SqFtTotLiving'],
                 PartialResid = partial_resid[, 'SqFtTotLiving'])
ggplot(df, aes(SqFtTotLiving, PartialResid)) +
  geom_point(shape=1) + scale_shape(solid = FALSE) +
  geom_smooth(linetype=2) +
  geom_line(aes(SqFtTotLiving, Terms))
dev.off()
```

위 의 결과를 보자. 편잔차는 SqFtTotLiving 변수가 주택 가격에 얼마나 영향을 미치는지 보여준다.

SqFtTotLiving 변수와 가격 사이의 관계는 분명히 비선형이다. 회귀선에 따르면 1,000 제곱피트 보다 작은 평수의 집에 대해서는 가격을 원래보다 낮게 추정하고, 2,000~3,000 제곱피트 집에 대해서는 더 높게 추정하고 있다. 4,000제곱피트 이상에 대해서는 데이터 개수가 너무 작아 뭐라고 결론 내릴 수 없다.

이러한 비선형성은 이렇게 이해할 수 있다. 원래 큰집에 500제곱피트를 추가하는 것 보다, 작은 집에 500제곱피트를 추가하는 것이 훨씬 더 큰 차이를 만든다. 따라서 SqFtTotLiving에 대해 단순성형 항 대신, 비선형항을 고려할 것을 생각해볼 수 있다.

주요개념

- **특잇점은 데이터 크기가 작을 때 문제를 일으킬 수 있지만, 주요 관심사는 데이터에서 문제점을 발견한다든지와 같은 이상을 찾아내는 것이다.**
- **데이터 크기가 작을 때는 단일 레코드 (회귀 특잇값 포함)가 회귀방정식에 큰 영향을 미치는 겨우도 있다. 하지만 빅데이터에서는 이러한 효과가 대부분 사라진다.**
- **회귀모형을 일반적인 추론 (p값 등)을 위해 사용할 경우 잔차 분포에 대한 특정 가정을 확인해야 한다. 하지만 보통 데이터 과학에서 잔차의 분포는 그렇게 중요하지 않다.**
- **편잔차그림을 사용하여 각 회귀 항의 적합서을 정량적으로 평가할 수 있다. 즉, 대체 모델에 대한 아이디어를 얻을 수 있다.**

4.7 다중회귀와 스플라인 회귀

응답변수와 예측변수 가의 관계가 반드시 선형일 필요가 없다. 약물 복용량에 따른 반응은 일반적으로 비선형을 띈다. 복용량을 두 배로 늘린다고 두 배의 반응이 나타나지 않는다. 제품에 대한 수요 역시 어떤 시점에서는 포화 상태가 되기 쉽다보니, 마케팅 비용은 선형함수가 아니다. 비선형 효과를 회귀분석에 담기 위해 회귀모형을 확장하는 몇 가지 방법이 있다.

용어 정리

- **다항회귀 : 회귀모형에 다항식(제곱, 세제곱 등) 항을 추가한 방식**
- **스플라인 회귀 : 다항 구간들을 부드러운 곡선 형태로 피팅한다. (비선형 효과 나타내기 위해서 사용)**
- **매듭 : 스플라인 구간을 구분하는 값들**
- **일반화가법모형(generalized additive model) : 자동으로 구간을 결정하는 스플라인 모델(유의어 : GAM)**

TIP_비선형회귀

통계학자가 비선형회귀에 대해 논한다면, 이는 최소제곱 방법으로 피팅할 수 없는 모델을 의미한다. 어떤 모델이 비선형일까? 본질적으로 예측변수들의 선형 결합 또는 일부 변환으로 응답변수를 표현할 수 없는 모든 모델을 말한다. 비선형회귀 모델은 수치 최적화가 필요하기에 피팅하기가 어렵고 더 많은 계산을 필요로 한다. 이러한 이유로 가능하면 선형모형을 이용하는 것이 일반적이다.

4.7.1 다항식

다항회귀란, 회귀식에 다항 항을 포함한 것을 말한다.

예를 들어 응답변수 Y와 예측변수 X 사이의 이차 회귀는 다음과 같은 식으로 표현 할 수 있다.

$$Y = b_0 + b_1 X + b_2 X^2 + e$$

다항회귀는 poly함수를 이용해 구할 수 있다. 예를 들면 다음은 킹 카운티 주택 데이터로 구한 SqFtTotLiving에 대해 이차 다항식을 피팅하는 과정을 보여준다.

```
> lm(AdjSalePrice ~ poly(SqFtTotLiving, 2) + SqFtLot +
+    BldgGrade + Bathrooms +  Bedrooms,
+    data=house_98105)

Call:
lm(formula = AdjSalePrice ~ poly(SqFtTotLiving, 2) + SqFtLot +
    BldgGrade + Bathrooms + Bedrooms, data = house_98105)

Coefficients:
            (Intercept)  poly(SqFtTotLiving, 2)1  poly(SqFtTotLiving, 2)2
              -402530.47               3271519.49                776934.02
                 SqFtLot                 BldgGrade                Bathrooms
                   32.56                 135717.06                 -1435.12
                Bedrooms
                -9191.94
```

결과를 통해 SqFtTotLiving에 대한 두 가지 계수가 있는 것을 볼 수 있다. 하나는 선형 항(일차 항)이고 나머지 하나는 이차항 이다.

편잔차그림은 SqFtTotLiving에 관한 회귀식의 곡률을 보여준다. 회귀 결과가 선형회귀 때의 결과에 비해 편잔차의 평활 곡선에 훨씬 더 가까운 것을 볼 수 있다.

다중회귀식 적용

단일회귀식 적용

4.7.2 스플라인

다중회귀는 비선형 관계에 대해 어느 정도의 곡률을 담아낼 수 있다. 하지만 3차, 4차 다항식과 같이 고차 항을 추가하는 것은 종종 회귀 방정식에 바람직하지 않은 '흔들림'을 초래한다. 비선형 관계를 모델링하는 또 다른 더 나은 방법은 **스플라인**을 사용하는 것이다. 스플라인은 고정된 점들 사이를 부드럽게 보간하는 방법을 말한다. 스플라인은 원래 선박이나 항공기를 건조할 때, 부드러운 곡선을 그리기 위해 사용됐다.

구간별 다항식은 예측변수를 위한 일련의 고정된 점(**매듭 Knot**) 사이를 부드럽게 연결한다. 스플라인을 구하는 것은 다중회귀보다 훨씬 복잡하다. 통계 SW는 일반적으로 스플라인 피팅을 위한 구체적인 사항을 모두 다룰 수 있듀륙 지원한다.

R패키지 splines는 회귀모형에서 **b-스플라인** 항을 만드는 데 사용할 수 있는 bs함수를 포함한다.

예를 들면 다음은 b- 스플라인 항을 주택 회귀모형에 추가하는 방법이다.

```
knots <- quantile(house_98105$SqFtTotLiving, p=c(.25, .5, .75))
lm_spline <- lm(AdjSalePrice ~ bs(SqFtTotLiving, knots=knots, degree=3)
                +SqFtLot + Bathrooms + Bedrooms + BldgGrade,
                data=house_98105)
```

이를 위해서는 **다항식의 차수와 매듭의 위치**, 이 두 가지 파라미터를 설정해야 한다.

위의 경우 예측변수 SqFtTotLiving은 3차 스플라인(degree = 3) 을 사용하여 모델에 포함되었다. bs의 기본 설정은 매듭을 경계에 배치한다. 하위에서는 매듭을 하위 사분위(.25), 중간 사분위(.5), 상위 사분위(.75)에 배치했다.

선형 항에서는 계수가 변수에 대한 직접적인 의미를 갖는 반면, 스플라인 항의 계수는 해석하기 어렵다.

대신, 스플라인의 적합도를 확인하기 위해 시각화 방법을 사용하는 것이 더 유용하다. 다음 그림은 회귀분석에 대한 편잔차그림을 보여준다. 다항회귀 모델과는 달리 스프라인 모델은 좀 더 매끄럽게 매칭되며 스플라인의 유연성이 뛰어난 것을 볼 수 있다. 이 경우, 회귀선이 데이터에 더 가깝게 맞는 것을 알 수 있다. 하지만 이것이 스플라인 회귀가 더 좋다는 것을 의미할까? 반드시 그런 것은 아니다. 크기가 아주 작은 주택(1,000제곱피트 미만)이 약간 큰 주택보다 더 높은 가치를 가질 것이라는 예측결과는 경제적으로 맞지 않다. 이것은 교란변수 때문일 수 있다.

```
df1 <- data.frame(SqFtTotLiving = house_98105[, 'SqFtTotLiving'],
                  Terms = terms1[, 1],
                  PartialResid = partial_resid1[, 1])
ggplot(df1, aes(SqFtTotLiving, PartialResid)) +
  geom_point(shape=1) + scale_shape(solid = FALSE) +
  geom_smooth(linetype=2) +
  geom_line(aes(SqFtTotLiving, Terms))+
  theme_bw()
```

4.7.3 일반화가법모형

사전 지식이나 회귀 진단을 통해 응답변수와 예측변수 사이에 비선형 관계가 있다는 것을 알았다고 하자. 다항 항은 관계를 포착하기에 유연성이 부족할 수 있으며 스플라인 항은 매듭을 어디로 할지 정해줘야 한다. **일반화가법모형(GAM)**은 스플라인 회귀를 자동으로 찾는 기술이다.

R의 GAM패키지로 주택 데이터에 GAM모델을 피팅해보자.

```
lm_gam <- gam(AdjSalePrice ~ s(SqFtTotLiving) + SqFtLot +
                Bathrooms +  Bedrooms + BldgGrade,
              data=house_98105)
```

s(SqFtTotLiving)라는 옵션이 스플라인 항에 대한 '최적' 매듭 점을 찾도록 gam함수에 지시한다.

```
df <- data.frame(SqFtTotLiving = house_98105[, 'SqFtTotLiving'],
                 Terms = terms[, 5],
                 PartialResid = partial_resid[, 5])
ggplot(df, aes(SqFtTotLiving, PartialResid)) +
  geom_point(shape=1) + scale_shape(solid = FALSE) +
  geom_smooth(linetype=2) +
  geom_line(aes(SqFtTotLiving, Terms))  +
  theme_bw()
```

주요개념

- **회귀분석에서 특잇값은 잔차가 큰 레코드를 말한다.**
- **다중공선성은 회귀방정식을 피팅할 때 수치 불안정성을 가져올 수 있다.**
- **교란변수는 모델에서 생략된 중요한 예측변수이며 허위 관계를 보여주는 회귀 결과를 낳을 수 있다.**
- **한 변수의 효과가 다른 변수의 수준에 영향을 받는다면, 두 변수 사이의 상호작용을 고려할 항이 필요하다.**
- **다중회귀분석은 예측변수와 결과변수 간의 비선형관계를 검증할 수 있다.**
- **스플라인은 매듭들로 함께 묶여 있는 일련의 구간별 다항식을 말한다.**
- **일반화기법모형(GAM)은 스플라인의 매듭을 자동으로 결정하는 프로세스를 가지고 있다.**

4.8 마치며

여러 예측변수와 결과변수 간의 관계를 설정하는 과정, 즉 회귀만큼 오랫동안 사용된 통계방법도 아마 없을 것이다. 기본 형태는 선형이다. 각 예측변수는 결과변수와의 선형 관계를 뜻하는 계수를 갖는다. 다항회귀나 스플라인 회귀와 같이 일반 회귀보다 발전된 형태에서는 비선형 관계도 가능하다. 고전적인 통계에서는, 어떤 현상을 설명하거나 묘사하기 위해 관측한 데이터에 적합한 모델을 찾는 것을 강조한다. 또한 모델을 사용할 때 적합도가 얼마나 되는지를 판단하기 위해 기존 ('표본 내') 측정 지표를 사용한다. 이와는 대조적으로 데이터 과학에서는 일반적으로 새 데이터의 값을 예측하는 것이 목적이다 보니, 외부 데이터에 대한 예측 정확도를 기반으로 한 지표들을 사용한다. 차원을 줄이고 더 컴팩한 모델을 만들기 위한 변수 선택 방법을 사용한다.

```
> zhvi <- unlist(zhvi[13,-(1:5)])
> zhvi
x1996.04 x1996.05 x1996.06 x1996.07
  165600   165300   165200   165300
x1996.08 x1996.09 x1996.10 x1996.11
  165400   165600   166300   167500
x1996.12 x1997.01 x1997.02 x1997.03
  168400   169300   170500   171900
x1997.04 x1997.05 x1997.06 x1997.07
  173200   174600   176000   177600
x1997.08 x1997.09 x1997.10 x1997.11
  179400   181200   182500   184000
x1997.12 x1998.01 x1998.02 x1998.03
  186000   187800   189500   191300
x1998.04 x1998.05 x1998.06 x1998.07
  193300   195300   197600   199800
x1998.08 x1998.09 x1998.10 x1998.11
  201800   203900   205700   206800
x1998.12 x1999.01 x1999.02 x1999.03
  207900   209300   210700   212200
x1999.04 x1999.05 x1999.06 x1999.07
  214000   215800   217700   219300
x1999.08 x1999.09 x1999.10 x1999.11
  221000   222700   224700   226700
x1999.12 x2000.01 x2000.02 x2000.03
  229100   231300   232900   233900
x2000.04 x2000.05 x2000.06 x2000.07
  235200   236600   238000   239400
x2000.08 x2000.09 x2000.10 x2000.11
  240500   241800   243200   244400
x2000.12 x2001.01 x2001.02 x2001.03
  245400   246500   247500   248400
x2001.04 x2001.05 x2001.06 x2001.07
  249100   249400   249700   250800
x2001.08 x2001.09 x2001.10 x2001.11
  252000   252900   254000   255400
x2001.12 x2002.01 x2002.02 x2002.03
  256500   257500   258500   259400
x2002.04 x2002.05 x2002.06 x2002.07
  260600   262000   263300   264500
```

```
x2002.08 x2002.09 x2002.10 x2002.11
  265900   267300   268400   269100
x2002.12 x2003.01 x2003.02 x2003.03
  269700   270600   272100   273400
x2003.04 x2003.05 x2003.06 x2003.07
  274100   274600   275500   276400
x2003.08 x2003.09 x2003.10 x2003.11
  277300   278200   279500   281300
x2003.12 x2004.01 x2004.02 x2004.03
  283500   286200   289400   292700
x2004.04 x2004.05 x2004.06 x2004.07
  296100   299600   302800   305500
x2004.08 x2004.09 x2004.10 x2004.11
  307800   310100   313100   317200
x2004.12 x2005.01 x2005.02 x2005.03
  321600   325300   329000   333200
x2005.04 x2005.05 x2005.06 x2005.07
  338200   343500   348800   353900
x2005.08 x2005.09 x2005.10 x2005.11
  359100   364500   369500   374100
x2005.12 x2006.01 x2006.02 x2006.03
  378700   383200   387600   392100
x2006.04 x2006.05 x2006.06 x2006.07
  396500   400600   404400   407700
x2006.08 x2006.09 x2006.10 x2006.11
  411100   414800   418300   421200
x2006.12 x2007.01 x2007.02 x2007.03
  423400   425600   427800   429600
x2007.04 x2007.05 x2007.06 x2007.07
  430900   432100   433200   434200
x2007.08 x2007.09 x2007.10 x2007.11
  434600   433500   431300   428400
x2007.12 x2008.01 x2008.02 x2008.03
  425200   421900   418400   415100
x2008.04 x2008.05 x2008.06 x2008.07
  411700   407400   403200   400300
x2008.08 x2008.09 x2008.10 x2008.11
  397900   394900   390600   385800
x2008.12 x2009.01 x2009.02 x2009.03
  381600   378200   374400   369800
x2009.04 x2009.05 x2009.06 x2009.07
  365000   360700   357100   354400
x2009.08 x2009.09 x2009.10 x2009.11
  352600   351800   351200   350900
x2009.12 x2010.01 x2010.02 x2010.03
  350600   350800   350900   350300
x2010.04 x2010.05 x2010.06 x2010.07
  349100   347800   345800   342800
x2010.08 x2010.09 x2010.10 x2010.11
  339900   337200   334700   332700
x2010.12 x2011.01 x2011.02 x2011.03
  330600   328100   325000   321600
x2011.04 x2011.05 x2011.06 x2011.07
  318700   316900   315500   314100
x2011.08 x2011.09 x2011.10 x2011.11
  312700   312000   312000   312200
x2011.12 x2012.01 x2012.02 x2012.03
  311900   311600   312300   313600
```

```
x2012.04 x2012.05 x2012.06 x2012.07
  315200   318000   322000   325300
x2012.08 x2012.09 x2012.10 x2012.11
  327500   329600   332400   335900
x2012.12 x2013.01 x2013.02 x2013.03
  339400   342800   347000   351600
x2013.04 x2013.05 x2013.06 x2013.07
  356000   360700   366100   370700
x2013.08 x2013.09 x2013.10 x2013.11
  374300   377500   380400   382600
x2013.12 x2014.01 x2014.02 x2014.03
  385000   387900   389800   392100
x2014.04 x2014.05 x2014.06 x2014.07
  395300   398100   400200   402000
x2014.08 x2014.09 x2014.10 x2014.11
  403500   405100   407400   410700
x2014.12 x2015.01 x2015.02 x2015.03
  413400   415400   417900   423200
x2015.04 x2015.05
  429900   435200
```

```
> parse_date_time(paste(substr(names(zhvi), start=2, stop=8), "01", sep="."),
"Ymd")
  [1] "1996-04-01 UTC"
  [2] "1996-05-01 UTC"
  [3] "1996-06-01 UTC"
  [4] "1996-07-01 UTC"
  [5] "1996-08-01 UTC"
  [6] "1996-09-01 UTC"
  [7] "1996-10-01 UTC"
  [8] "1996-11-01 UTC"
  [9] "1996-12-01 UTC"
 [10] "1997-01-01 UTC"
 [11] "1997-02-01 UTC"
 [12] "1997-03-01 UTC"
 [13] "1997-04-01 UTC"
 [14] "1997-05-01 UTC"
 [15] "1997-06-01 UTC"
 [16] "1997-07-01 UTC"
 [17] "1997-08-01 UTC"
 [18] "1997-09-01 UTC"
 [19] "1997-10-01 UTC"
 [20] "1997-11-01 UTC"
 [21] "1997-12-01 UTC"
 [22] "1998-01-01 UTC"
 [23] "1998-02-01 UTC"
 [24] "1998-03-01 UTC"
 [25] "1998-04-01 UTC"
 [26] "1998-05-01 UTC"
 [27] "1998-06-01 UTC"
 [28] "1998-07-01 UTC"
 [29] "1998-08-01 UTC"
 [30] "1998-09-01 UTC"
 [31] "1998-10-01 UTC"
 [32] "1998-11-01 UTC"
 [33] "1998-12-01 UTC"
```

```
 [34] "1999-01-01 UTC"
 [35] "1999-02-01 UTC"
 [36] "1999-03-01 UTC"
 [37] "1999-04-01 UTC"
 [38] "1999-05-01 UTC"
 [39] "1999-06-01 UTC"
 [40] "1999-07-01 UTC"
 [41] "1999-08-01 UTC"
 [42] "1999-09-01 UTC"
 [43] "1999-10-01 UTC"
 [44] "1999-11-01 UTC"
 [45] "1999-12-01 UTC"
 [46] "2000-01-01 UTC"
 [47] "2000-02-01 UTC"
 [48] "2000-03-01 UTC"
 [49] "2000-04-01 UTC"
 [50] "2000-05-01 UTC"
 [51] "2000-06-01 UTC"
 [52] "2000-07-01 UTC"
 [53] "2000-08-01 UTC"
 [54] "2000-09-01 UTC"
 [55] "2000-10-01 UTC"
 [56] "2000-11-01 UTC"
 [57] "2000-12-01 UTC"
 [58] "2001-01-01 UTC"
 [59] "2001-02-01 UTC"
 [60] "2001-03-01 UTC"
 [61] "2001-04-01 UTC"
 [62] "2001-05-01 UTC"
 [63] "2001-06-01 UTC"
 [64] "2001-07-01 UTC"
 [65] "2001-08-01 UTC"
 [66] "2001-09-01 UTC"
 [67] "2001-10-01 UTC"
 [68] "2001-11-01 UTC"
 [69] "2001-12-01 UTC"
 [70] "2002-01-01 UTC"
 [71] "2002-02-01 UTC"
 [72] "2002-03-01 UTC"
 [73] "2002-04-01 UTC"
 [74] "2002-05-01 UTC"
 [75] "2002-06-01 UTC"
 [76] "2002-07-01 UTC"
 [77] "2002-08-01 UTC"
 [78] "2002-09-01 UTC"
 [79] "2002-10-01 UTC"
 [80] "2002-11-01 UTC"
 [81] "2002-12-01 UTC"
 [82] "2003-01-01 UTC"
 [83] "2003-02-01 UTC"
 [84] "2003-03-01 UTC"
 [85] "2003-04-01 UTC"
 [86] "2003-05-01 UTC"
 [87] "2003-06-01 UTC"
 [88] "2003-07-01 UTC"
 [89] "2003-08-01 UTC"
 [90] "2003-09-01 UTC"
 [91] "2003-10-01 UTC"
```

```
 [92] "2003-11-01 UTC"
 [93] "2003-12-01 UTC"
 [94] "2004-01-01 UTC"
 [95] "2004-02-01 UTC"
 [96] "2004-03-01 UTC"
 [97] "2004-04-01 UTC"
 [98] "2004-05-01 UTC"
 [99] "2004-06-01 UTC"
[100] "2004-07-01 UTC"
[101] "2004-08-01 UTC"
[102] "2004-09-01 UTC"
[103] "2004-10-01 UTC"
[104] "2004-11-01 UTC"
[105] "2004-12-01 UTC"
[106] "2005-01-01 UTC"
[107] "2005-02-01 UTC"
[108] "2005-03-01 UTC"
[109] "2005-04-01 UTC"
[110] "2005-05-01 UTC"
[111] "2005-06-01 UTC"
[112] "2005-07-01 UTC"
[113] "2005-08-01 UTC"
[114] "2005-09-01 UTC"
[115] "2005-10-01 UTC"
[116] "2005-11-01 UTC"
[117] "2005-12-01 UTC"
[118] "2006-01-01 UTC"
[119] "2006-02-01 UTC"
[120] "2006-03-01 UTC"
[121] "2006-04-01 UTC"
[122] "2006-05-01 UTC"
[123] "2006-06-01 UTC"
[124] "2006-07-01 UTC"
[125] "2006-08-01 UTC"
[126] "2006-09-01 UTC"
[127] "2006-10-01 UTC"
[128] "2006-11-01 UTC"
[129] "2006-12-01 UTC"
[130] "2007-01-01 UTC"
[131] "2007-02-01 UTC"
[132] "2007-03-01 UTC"
[133] "2007-04-01 UTC"
[134] "2007-05-01 UTC"
[135] "2007-06-01 UTC"
[136] "2007-07-01 UTC"
[137] "2007-08-01 UTC"
[138] "2007-09-01 UTC"
[139] "2007-10-01 UTC"
[140] "2007-11-01 UTC"
[141] "2007-12-01 UTC"
[142] "2008-01-01 UTC"
[143] "2008-02-01 UTC"
[144] "2008-03-01 UTC"
[145] "2008-04-01 UTC"
[146] "2008-05-01 UTC"
[147] "2008-06-01 UTC"
[148] "2008-07-01 UTC"
[149] "2008-08-01 UTC"
```

```
[150] "2008-09-01 UTC"
[151] "2008-10-01 UTC"
[152] "2008-11-01 UTC"
[153] "2008-12-01 UTC"
[154] "2009-01-01 UTC"
[155] "2009-02-01 UTC"
[156] "2009-03-01 UTC"
[157] "2009-04-01 UTC"
[158] "2009-05-01 UTC"
[159] "2009-06-01 UTC"
[160] "2009-07-01 UTC"
[161] "2009-08-01 UTC"
[162] "2009-09-01 UTC"
[163] "2009-10-01 UTC"
[164] "2009-11-01 UTC"
[165] "2009-12-01 UTC"
[166] "2010-01-01 UTC"
[167] "2010-02-01 UTC"
[168] "2010-03-01 UTC"
[169] "2010-04-01 UTC"
[170] "2010-05-01 UTC"
[171] "2010-06-01 UTC"
[172] "2010-07-01 UTC"
[173] "2010-08-01 UTC"
[174] "2010-09-01 UTC"
[175] "2010-10-01 UTC"
[176] "2010-11-01 UTC"
[177] "2010-12-01 UTC"
[178] "2011-01-01 UTC"
[179] "2011-02-01 UTC"
[180] "2011-03-01 UTC"
[181] "2011-04-01 UTC"
[182] "2011-05-01 UTC"
[183] "2011-06-01 UTC"
[184] "2011-07-01 UTC"
[185] "2011-08-01 UTC"
[186] "2011-09-01 UTC"
[187] "2011-10-01 UTC"
[188] "2011-11-01 UTC"
[189] "2011-12-01 UTC"
[190] "2012-01-01 UTC"
[191] "2012-02-01 UTC"
[192] "2012-03-01 UTC"
[193] "2012-04-01 UTC"
[194] "2012-05-01 UTC"
[195] "2012-06-01 UTC"
[196] "2012-07-01 UTC"
[197] "2012-08-01 UTC"
[198] "2012-09-01 UTC"
[199] "2012-10-01 UTC"
[200] "2012-11-01 UTC"
[201] "2012-12-01 UTC"
[202] "2013-01-01 UTC"
[203] "2013-02-01 UTC"
[204] "2013-03-01 UTC"
[205] "2013-04-01 UTC"
[206] "2013-05-01 UTC"
[207] "2013-06-01 UTC"
```

```
[208] "2013-07-01 UTC"
[209] "2013-08-01 UTC"
[210] "2013-09-01 UTC"
[211] "2013-10-01 UTC"
[212] "2013-11-01 UTC"
[213] "2013-12-01 UTC"
[214] "2014-01-01 UTC"
[215] "2014-02-01 UTC"
[216] "2014-03-01 UTC"
[217] "2014-04-01 UTC"
[218] "2014-05-01 UTC"
[219] "2014-06-01 UTC"
[220] "2014-07-01 UTC"
[221] "2014-08-01 UTC"
[222] "2014-09-01 UTC"
[223] "2014-10-01 UTC"
[224] "2014-11-01 UTC"
[225] "2014-12-01 UTC"
[226] "2015-01-01 UTC"
[227] "2015-02-01 UTC"
[228] "2015-03-01 UTC"
[229] "2015-04-01 UTC"
[230] "2015-05-01 UTC"
```

```
> zhvi <- data.frame(ym=dates, zhvi_px=zhvi, row.names = NULL) %>%
+   mutate(zhvi_idx=zhvi/last(zhvi))
> zhvi
            ym zhvi_px  zhvi_idx
1   1996-04-01  165600 0.3805147
2   1996-05-01  165300 0.3798254
3   1996-06-01  165200 0.3795956
4   1996-07-01  165300 0.3798254
5   1996-08-01  165400 0.3800551
6   1996-09-01  165600 0.3805147
7   1996-10-01  166300 0.3821232
8   1996-11-01  167500 0.3848805
9   1996-12-01  168400 0.3869485
10  1997-01-01  169300 0.3890165
11  1997-02-01  170500 0.3917739
12  1997-03-01  171900 0.3949908
13  1997-04-01  173200 0.3979779
14  1997-05-01  174600 0.4011949
15  1997-06-01  176000 0.4044118
16  1997-07-01  177600 0.4080882
17  1997-08-01  179400 0.4122243
18  1997-09-01  181200 0.4163603
19  1997-10-01  182500 0.4193474
20  1997-11-01  184000 0.4227941
21  1997-12-01  186000 0.4273897
22  1998-01-01  187800 0.4315257
23  1998-02-01  189500 0.4354320
24  1998-03-01  191300 0.4395680
25  1998-04-01  193300 0.4441636
26  1998-05-01  195300 0.4487592
27  1998-06-01  197600 0.4540441
28  1998-07-01  199800 0.4590993
29  1998-08-01  201800 0.4636949
```

```
30  1998-09-01  203900 0.4685202
31  1998-10-01  205700 0.4726562
32  1998-11-01  206800 0.4751838
33  1998-12-01  207900 0.4777114
34  1999-01-01  209300 0.4809283
35  1999-02-01  210700 0.4841452
36  1999-03-01  212200 0.4875919
37  1999-04-01  214000 0.4917279
38  1999-05-01  215800 0.4958640
39  1999-06-01  217700 0.5002298
40  1999-07-01  219300 0.5039062
41  1999-08-01  221000 0.5078125
42  1999-09-01  222700 0.5117188
43  1999-10-01  224700 0.5163143
44  1999-11-01  226700 0.5209099
45  1999-12-01  229100 0.5264246
46  2000-01-01  231300 0.5314798
47  2000-02-01  232900 0.5351562
48  2000-03-01  233900 0.5374540
49  2000-04-01  235200 0.5404412
50  2000-05-01  236600 0.5436581
51  2000-06-01  238000 0.5468750
52  2000-07-01  239400 0.5500919
53  2000-08-01  240500 0.5526195
54  2000-09-01  241800 0.5556066
55  2000-10-01  243200 0.5588235
56  2000-11-01  244400 0.5615809
57  2000-12-01  245400 0.5638787
58  2001-01-01  246500 0.5664062
59  2001-02-01  247500 0.5687040
60  2001-03-01  248400 0.5707721
61  2001-04-01  249100 0.5723805
62  2001-05-01  249400 0.5730699
63  2001-06-01  249700 0.5737592
64  2001-07-01  250800 0.5762868
65  2001-08-01  252000 0.5790441
66  2001-09-01  252900 0.5811121
67  2001-10-01  254000 0.5836397
68  2001-11-01  255400 0.5868566
69  2001-12-01  256500 0.5893842
70  2002-01-01  257500 0.5916820
71  2002-02-01  258500 0.5939798
72  2002-03-01  259400 0.5960478
73  2002-04-01  260600 0.5988051
74  2002-05-01  262000 0.6020221
75  2002-06-01  263300 0.6050092
76  2002-07-01  264500 0.6077665
77  2002-08-01  265900 0.6109835
78  2002-09-01  267300 0.6142004
79  2002-10-01  268400 0.6167279
80  2002-11-01  269100 0.6183364
81  2002-12-01  269700 0.6197151
82  2003-01-01  270600 0.6217831
83  2003-02-01  272100 0.6252298
84  2003-03-01  273400 0.6282169
85  2003-04-01  274100 0.6298254
86  2003-05-01  274600 0.6309743
87  2003-06-01  275500 0.6330423
```

```
88  2003-07-01  276400 0.6351103
89  2003-08-01  277300 0.6371783
90  2003-09-01  278200 0.6392463
91  2003-10-01  279500 0.6422335
92  2003-11-01  281300 0.6463695
93  2003-12-01  283500 0.6514246
94  2004-01-01  286200 0.6576287
95  2004-02-01  289400 0.6649816
96  2004-03-01  292700 0.6725643
97  2004-04-01  296100 0.6803768
98  2004-05-01  299600 0.6884191
99  2004-06-01  302800 0.6957721
100 2004-07-01  305500 0.7019761
101 2004-08-01  307800 0.7072610
102 2004-09-01  310100 0.7125460
103 2004-10-01  313100 0.7194393
104 2004-11-01  317200 0.7288603
105 2004-12-01  321600 0.7389706
106 2005-01-01  325300 0.7474724
107 2005-02-01  329000 0.7559743
108 2005-03-01  333200 0.7656250
109 2005-04-01  338200 0.7771140
110 2005-05-01  343500 0.7892923
111 2005-06-01  348800 0.8014706
112 2005-07-01  353900 0.8131893
113 2005-08-01  359100 0.8251379
114 2005-09-01  364500 0.8375460
115 2005-10-01  369500 0.8490349
116 2005-11-01  374100 0.8596048
117 2005-12-01  378700 0.8701746
118 2006-01-01  383200 0.8805147
119 2006-02-01  387600 0.8906250
120 2006-03-01  392100 0.9009651
121 2006-04-01  396500 0.9110754
122 2006-05-01  400600 0.9204963
123 2006-06-01  404400 0.9292279
124 2006-07-01  407700 0.9368107
125 2006-08-01  411100 0.9446232
126 2006-09-01  414800 0.9531250
127 2006-10-01  418300 0.9611673
128 2006-11-01  421200 0.9678309
129 2006-12-01  423400 0.9728860
130 2007-01-01  425600 0.9779412
131 2007-02-01  427800 0.9829963
132 2007-03-01  429600 0.9871324
133 2007-04-01  430900 0.9901195
134 2007-05-01  432100 0.9928768
135 2007-06-01  433200 0.9954044
136 2007-07-01  434200 0.9977022
137 2007-08-01  434600 0.9986213
138 2007-09-01  433500 0.9960938
139 2007-10-01  431300 0.9910386
140 2007-11-01  428400 0.9843750
141 2007-12-01  425200 0.9770221
142 2008-01-01  421900 0.9694393
143 2008-02-01  418400 0.9613971
144 2008-03-01  415100 0.9538143
145 2008-04-01  411700 0.9460018
```

```
146 2008-05-01  407400 0.9361213
147 2008-06-01  403200 0.9264706
148 2008-07-01  400300 0.9198070
149 2008-08-01  397900 0.9142923
150 2008-09-01  394900 0.9073989
151 2008-10-01  390600 0.8975184
152 2008-11-01  385800 0.8864890
153 2008-12-01  381600 0.8768382
154 2009-01-01  378200 0.8690257
155 2009-02-01  374400 0.8602941
156 2009-03-01  369800 0.8497243
157 2009-04-01  365000 0.8386949
158 2009-05-01  360700 0.8288143
159 2009-06-01  357100 0.8205423
160 2009-07-01  354400 0.8143382
161 2009-08-01  352600 0.8102022
162 2009-09-01  351800 0.8083640
163 2009-10-01  351200 0.8069853
164 2009-11-01  350900 0.8062960
165 2009-12-01  350600 0.8056066
166 2010-01-01  350800 0.8060662
167 2010-02-01  350900 0.8062960
168 2010-03-01  350300 0.8049173
169 2010-04-01  349100 0.8021599
170 2010-05-01  347800 0.7991728
171 2010-06-01  345800 0.7945772
172 2010-07-01  342800 0.7876838
173 2010-08-01  339900 0.7810202
174 2010-09-01  337200 0.7748162
175 2010-10-01  334700 0.7690717
176 2010-11-01  332700 0.7644761
177 2010-12-01  330600 0.7596507
178 2011-01-01  328100 0.7539062
179 2011-02-01  325000 0.7467831
180 2011-03-01  321600 0.7389706
181 2011-04-01  318700 0.7323070
182 2011-05-01  316900 0.7281710
183 2011-06-01  315500 0.7249540
184 2011-07-01  314100 0.7217371
185 2011-08-01  312700 0.7185202
186 2011-09-01  312000 0.7169118
187 2011-10-01  312000 0.7169118
188 2011-11-01  312200 0.7173713
189 2011-12-01  311900 0.7166820
190 2012-01-01  311600 0.7159926
191 2012-02-01  312300 0.7176011
192 2012-03-01  313600 0.7205882
193 2012-04-01  315200 0.7242647
194 2012-05-01  318000 0.7306985
195 2012-06-01  322000 0.7398897
196 2012-07-01  325300 0.7474724
197 2012-08-01  327500 0.7525276
198 2012-09-01  329600 0.7573529
199 2012-10-01  332400 0.7637868
200 2012-11-01  335900 0.7718290
201 2012-12-01  339400 0.7798713
202 2013-01-01  342800 0.7876838
203 2013-02-01  347000 0.7973346
```

```
204 2013-03-01  351600 0.8079044
205 2013-04-01  356000 0.8180147
206 2013-05-01  360700 0.8288143
207 2013-06-01  366100 0.8412224
208 2013-07-01  370700 0.8517923
209 2013-08-01  374300 0.8600643
210 2013-09-01  377500 0.8674173
211 2013-10-01  380400 0.8740809
212 2013-11-01  382600 0.8791360
213 2013-12-01  385000 0.8846507
214 2014-01-01  387900 0.8913143
215 2014-02-01  389800 0.8956801
216 2014-03-01  392100 0.9009651
217 2014-04-01  395300 0.9083180
218 2014-05-01  398100 0.9147518
219 2014-06-01  400200 0.9195772
220 2014-07-01  402000 0.9237132
221 2014-08-01  403500 0.9271599
222 2014-09-01  405100 0.9308364
223 2014-10-01  407400 0.9361213
224 2014-11-01  410700 0.9437040
225 2014-12-01  413400 0.9499081
226 2015-01-01  415400 0.9545037
227 2015-02-01  417900 0.9602482
228 2015-03-01  423200 0.9724265
229 2015-04-01  429900 0.9878217
230 2015-05-01  435200 1.0000000
```