R_HW2_엄상준

#1. Heart Disease Example

##1-1. Data 불러오기

이번 Chapter 에서는 Heart Disease 의 발병과 관련한 데이터를 다룰 것이다. 데이터는 'faraway'에 내장되어 있는 데이터인 wcgs 를 사용할 것이다.

```
data(wcgs, package='faraway')
```

데이터에 관하여 간략하게 설명하자면, 39 세에서 59 세 사이의 총 3154 명의 건강한 사람을 대상으로 여러 가지 변수들을 조사한 뒤 8.5 년 뒤에 그들이 심장병이 발병했는 지 여부를 조사한 데이터이다.

wcgs 데이터셋 내에 있는 여러가지 변수 중, 우리는 키와 하루에 피는 담배 개수 그리고 심장병 발병의 변수들을 살펴볼 것이다.

```
summary(wcgs[,c('chd', 'height', 'cigs')])
```

```
##   chd           height           cigs
##  no :2897   Min.   :60.00   Min.   : 0.0
##  yes: 257   1st Qu.:68.00   1st Qu.: 0.0
##             Median :70.00   Median : 0.0
##             Mean   :69.78   Mean   :11.6
##             3rd Qu.:72.00   3rd Qu.:20.0
##             Max.   :78.00   Max.   :99.0
```

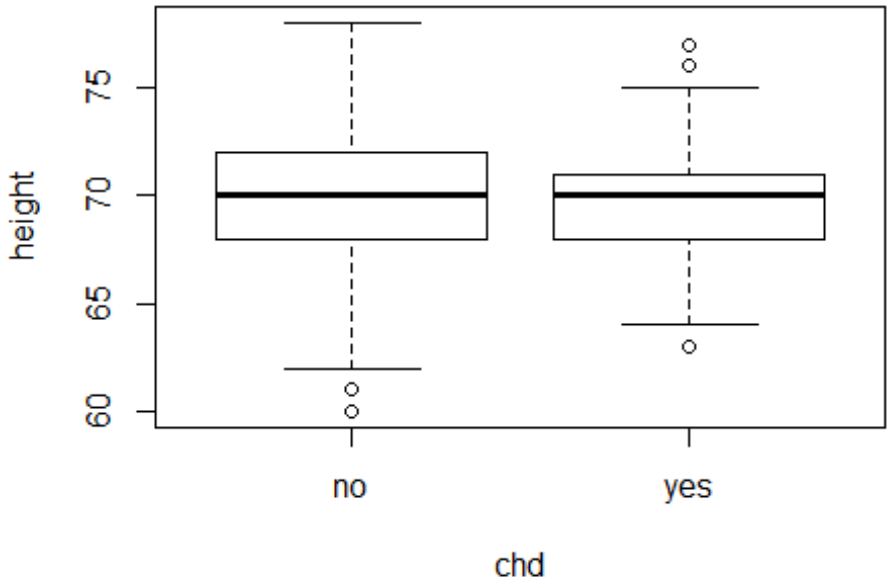
총 3154 명을 대상으로 하였으며 이 중 오직 257 명만 심장병이 발병했다는 것을 알 수 있다.

##1-2. Data 살펴보기

그래프를 그려서 변수들 간의 관계를 살펴보자.

우선 키와 심장병 발병 간의 boxplot 을 그려보면 다음과 같다.

```
plot(height~ chd, wcgs)
```



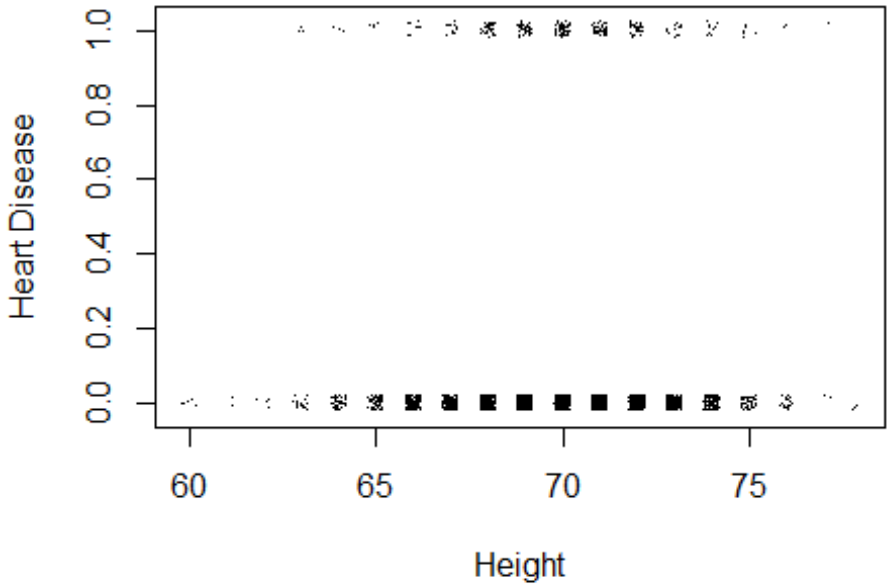
➔ 심장병 발병 여부의 관계없이 키의 분포는 비슷하다는 것을 알 수 있다.

➔ 그런데 여기서 종속 변수는 심장병 발병이므로, 심장병 발병을 Response 로 하는 그래프를 새로 그려보자.

```
wcgs$y <- ifelse(wcgs$chd == 'no', 0, 1)
```

심장병 발병은 categorical 데이터로 Yes or No 로 되어있으므로 이를 Yes 는 1 로 No 는 0 으로 바꿔주는 작업을 진행하였다. 그 후 Plot 을 그리는데, 여기서 jittering 을 수행하였다. jittering 이란, plot 을 그릴 때 같은 값을 가지는 데이터들이 많아서 서로 겹치는 일이 일어나는 경우 데이터의 밀집도를 더 잘 표현하기 위하여 각 데이터들에 조금의 Random Noise 를 추가해주는 것이다. 키와 심장병 발병의 변수들에 대해 서로 같은 값을 가지는 데이터들이 많으므로 두 변수 모두에 jittering 을 해주었다.

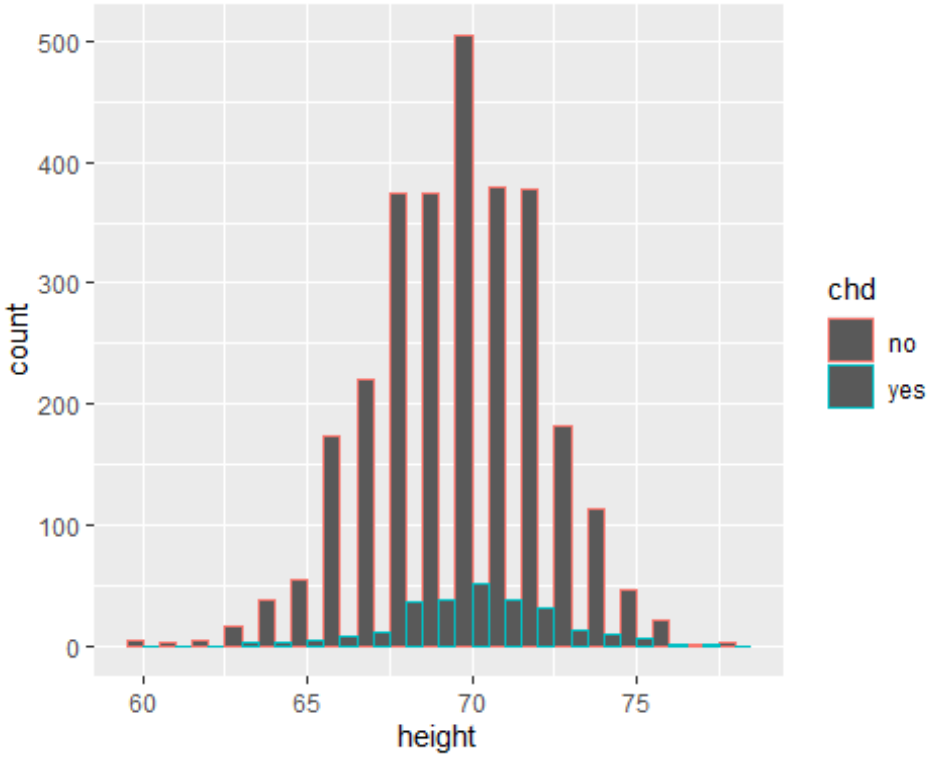
```
plot(jitter(y,0.1) ~ jitter(height), wcgs, xlab='Height', ylab='Heart
Disease',
     pch='.')
```



➔ 이번에도 동일하게 심장병 발병에 관계없이 키는 비슷한 분포를 보인다는 것을 알 수 있다.

ggplot 을 이용하여 histogram 을 그릴 수도 있다. 특히 dodge 를 이용하여 두 histogram 이 교차하도록 즉, 한 그래프 안에 나타나도록 할 수 있다. 그리고 추가적으로 bin width 를 설정해주어서 bin 의 넓이를 지정해줄 수 있다.

```
ggplot(wcgs, aes(x=height, color=chd)) + geom_histogram(position='dodge', binwidth=1)
```

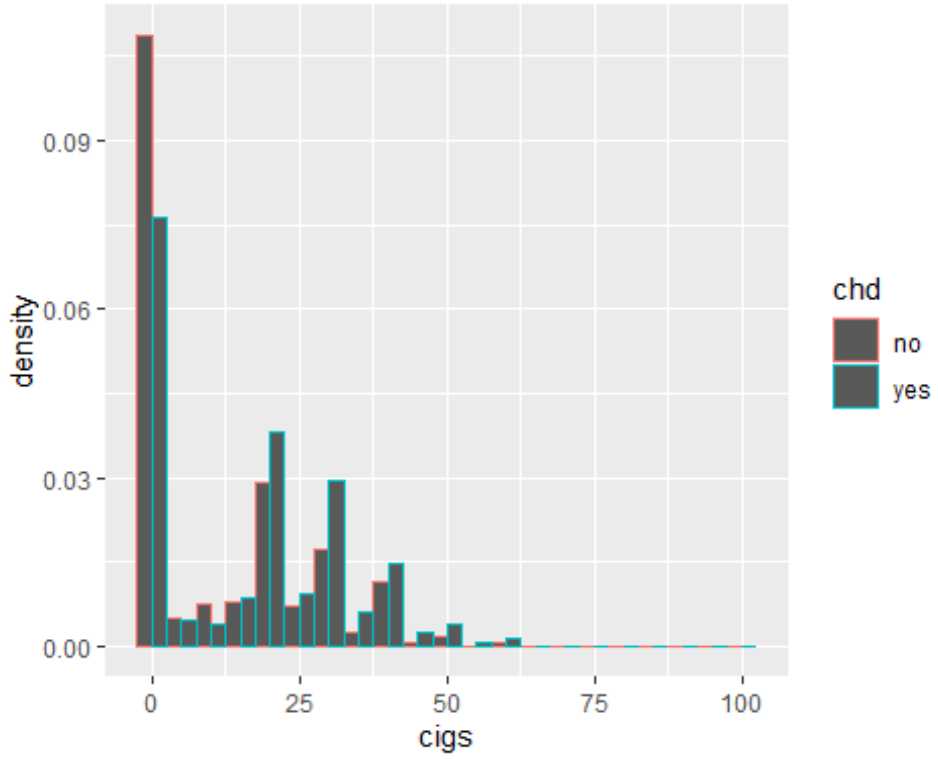


➔ 결과는 마찬가지로 비슷한 분포를 보인다는 것을 알 수 있다.

➔ 다만 시각적으로 더 깔끔하다.

흡연과 심장병 발병 간의 그래프도 같은 방법으로 그릴 수 있다. 다만 여기서는 y 축을 count 가 아니라 density 로 표현하였다는 차이가 있다. 왜냐하면 단순히 count 로 하면 0 에 너무 많은 No count 가 몰려서 다른 값들을 살펴보기가 어렵기 때문이다.

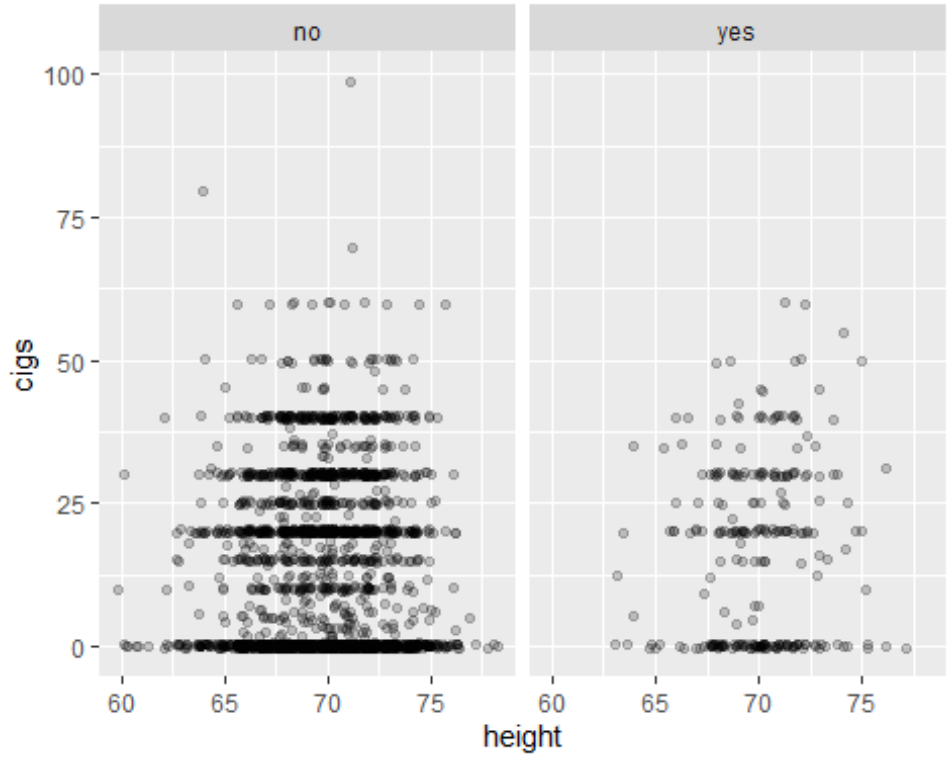
```
ggplot(wcgs, aes(x=cigs, color=chd)) + geom_histogram(position='dodge',
                                                      binwidth = 5,
                                                      aes(y=..density..))
```



➔ 우리의 예상과 비슷하게 흡연을 하면 흡연을 하지 않았을 때 보다 심장병 발병 비율이 더 높다.

세 변수를 하나의 display 에 표현하는 것도 좋은 방법이다. 다음의 그래프는 No 와 Yes 인 상황을 나누어서 Height 와 Cigs 의 관계를 살펴본 것이다. 여기서도 jittering 을 사용하였는데 추가적으로 alpha argument 도 사용하였다. alpha 는 겹쳐진 포인트가 있을수록 더 선명하게 표현해주는 option 이다.

```
ggplot(wcgs, aes(x=height, y=cigs)) +
  geom_point(alpha=0.2, position = position_jitter()) +
   facet_grid(~ chd)
```



➔ 명확한 차이가 있는 것 같지는 않다.

#2. Logistic Regression

우리는 심장병 발병 여부를 예측하기 위하여 outcome 자체보다는 outcome 이 일어날 확률에 대해 Modeling 을 하는 것이 더 정확할 것이다. 즉, outcome 이 1 이 나올 확률에 대한 확률 값을 구하고, 이를 바탕으로 outcome 을 예측하는 것이다. 그런데 standard linear Model 을 통해 얻은 response 값은 확률 값의 범위인 [0,1]를 벗어나기도 한다. 따라서 우리는 다른 방식을 찾아야 한다.

우선 stand linear Model 에서 세웠던 것처럼 모델을 한 번 세워보자.

Y_i for i=1,..., n 은 종속변수로 0 과 1 의 값을 가지고 $P(Y_i = 1) = p_i$ 라고 하자. 그리고 우리는 q 개의 predictor 를 가지고 있다고 하자. 그럼 이 때 linear predictor 를 세워보면 다음과 같다.

$\eta_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_q x_{iq}$

그런데 문제는 위에서 언급했듯이 이 predictor 는 0 과 1 사이의 값을 벗어날 수 있기 때문에 $\eta_i = p_i$라고 둘 수 없다. 따라서 우리는 위의 linear predictor 를 변환하는 작업을 거쳐야 한다. 그 작업이 바로 link function $\eta_i = g(p_i)$를 사용하는 것이다. 즉, η_i값이 p_i를 어떤 function 에 집어넣었을 때 나오는 값이라고 가정하는 것이다.

그러면 어떤 link function 을 사용해야 하는가?

$\eta = \log\left(\frac{p}{1-p}\right)$

또는

$p = \frac{e^{\eta}}{1+e^{\eta}}$

를 사용하면 된다.

이러한 function 을 logit 이라 부른다.

그리고 이 logit link 와 linear predictor 를 결합하여 Regression 을 하는 것을 우리는 Logistic Regression 이라고 부른다.

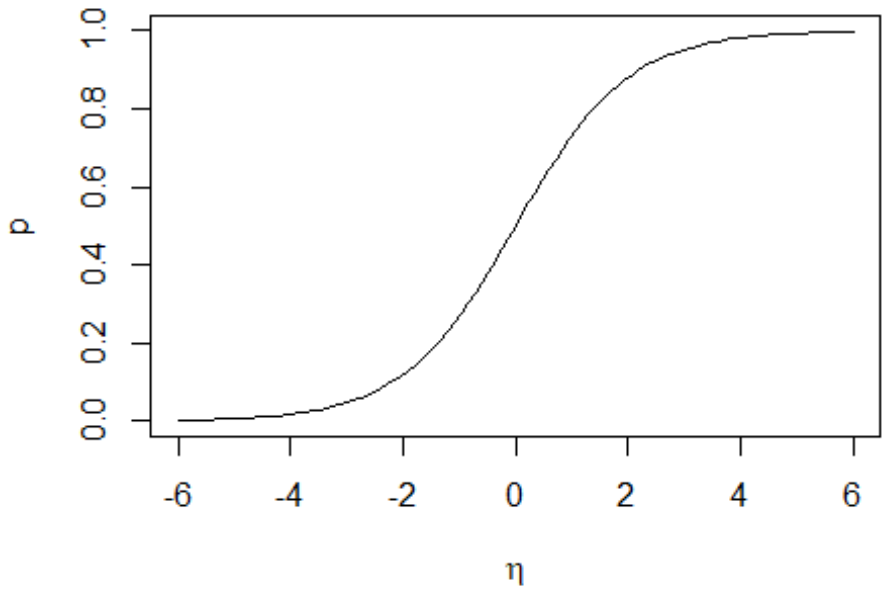
```
##2-1. Logistic Curve
```

p와 η의 관계를 그래프로 그려서 이해해보자.

ilogit 은 logit 의 역함수값을 출력해주는 명령어이다.

η가 -6 에서 6 사이일 때 p 값의 변화는 다음과 같다.

```
curve(ilogit(x), -6, 6, xlab=expression(eta), ylab='p')
```



➔ 그래프에서 알 수 있듯이, η값이 아무리 커지거나 작아져도 p 값은 0 과 1 사이에 위치한다는 것을 알 수 있다.

➔ 또한 η값에 관계없이 p 값은 0 과 1 값이 될 수 없다. 즉 0 이나 1 에 한없이 가까워질 수는 있어도 0 이나 1 값은 나오지 않는다는 것이다. 이를 prediction 의 측면에서 이해하면 linear predictor 값이 아무리 작거나 크더라도 model 은 해당 y 값을 0 이나 1 이라고 단정하지 않는다.

##2-2. Model 세우기

그렇다면 Logistic Regression 에서는 Parameter 들의 Estimation 을 어떻게 할까?

이는 다음의 log-likelihood 의 값을 최대화하는 Parameter 들의 값을 찾으면 된다.

$$l(\beta) = \sum_{i=1}^{n} [y_i \eta_i - \log(1 + e_i^{\eta})]$$

R 에서 Logistic Regression Model 을 세우는 것은 단순하다.

glm function 을 사용하면 된다.

안에 들어가는 방정식의 형태는 Chapter.1 에서 사용했던 방정식의 방식과 동일하다. 다만 여기서의 차이는 family 에 binomial 을 추가해주는 것인데, 이는 glm 이 단순히 logistic regression model 을 세우는 데에만 사용되는 것이 아니기 때문이다. Logistic Model 은 generalized linear model 의 한 case 이다.

```
lmod <- glm(chd ~ height + cigs, family= binomial, wcgs)
```

##2-3. Model 요약

Model 에 대한 정보를 보는 방식은 standard linear regression 때와 동일하다.

summary command 를 사용하거나

```
summary(lmod)
```

```
##
## Call:
## glm(formula = chd ~ height + cigs, family = binomial, data = wcgs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0041  -0.4425  -0.3630  -0.3499   2.4357
```

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.50161    1.84186  -2.444   0.0145 *
## height       0.02521    0.02633   0.957   0.3383
## cigs         0.02313    0.00404   5.724 1.04e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1781.2  on 3153  degrees of freedom
## Residual deviance: 1749.0  on 3151  degrees of freedom
## AIC: 1755
##
## Number of Fisher Scoring iterations: 5
```

더 짧은 버전을 원한다면 sumary command 를 사용할 수 있다.

```
sumary(lmod)
```

```
##               Estimate Std. Error z value  Pr(>|z|)
## (Intercept) -4.5016140  1.8418627 -2.4441   0.01452
## height       0.0252078  0.0263274  0.9575   0.33833
## cigs         0.0231274  0.0040402  5.7243 1.038e-08
##
## n = 3154 p = 3
## Deviance = 1749.04923 Null Deviance = 1781.24374 (Difference = 32.19451)
```

분석 결과

$\widehat{\beta_0} = -4.50, \widehat{\beta_1} = 0.025, \widehat{\beta_2} = 0.023$ 의 값을 얻었다.

##2-4. Plot 그리기

위의 결과를 바탕으로 키나 흡연 둘 중 하나의 변수를 고정하고 나머지 변수의 변화에 따라 심장병 발병이 어떻게 되는 지를 그래프로 나타내보자.

그러기 위해서 우선 coefficient 값들을 따로 뽑아내준다.

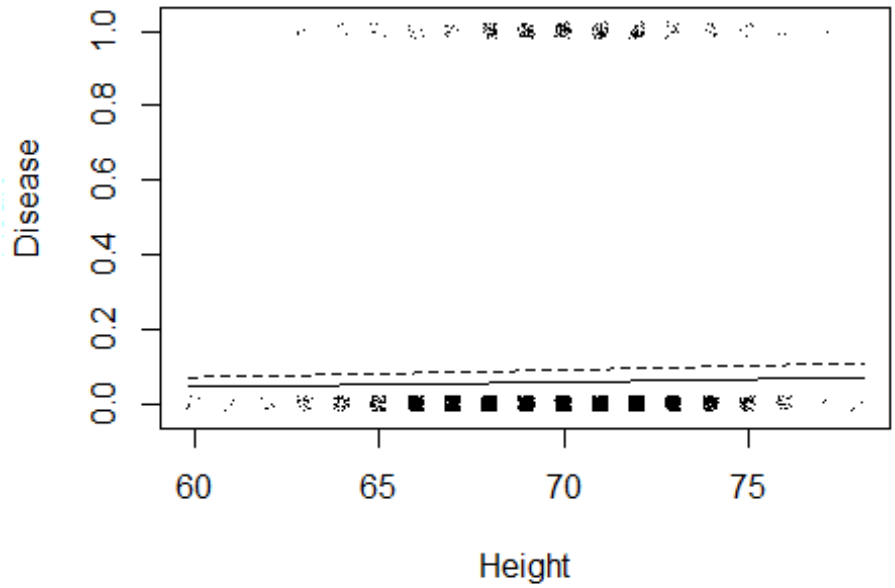
coef command 를 이용해서 손쉽게 할 수 있다.

```
(beta <- coef(lmod))
```

```
## (Intercept)      height        cigs
## -4.50161397  0.02520779  0.02312740
```

우선 비흡연자와 하루에 한 갑을 피는 사람에 대해 키의 변화에 따라 심장병 발병이 어떻게 변화하는 지를 알아보자. 즉, cigs 변수에 해당하는 값을 0 과 20 으로 고정하고 height 변수에 해당하는 x 값만 변화해주는 것이다.

```
plot(jitter(y, 0.1) ~ jitter(height), wcgs, xlab='Height', ylab='Heart
     Disease', pch='.')
curve(ilogit(beta[1] + beta[2]*x + beta[3]*0), add=TRUE)
curve(ilogit(beta[1] + beta[2]*x + beta[3]*20), add=TRUE, lty=2)
```



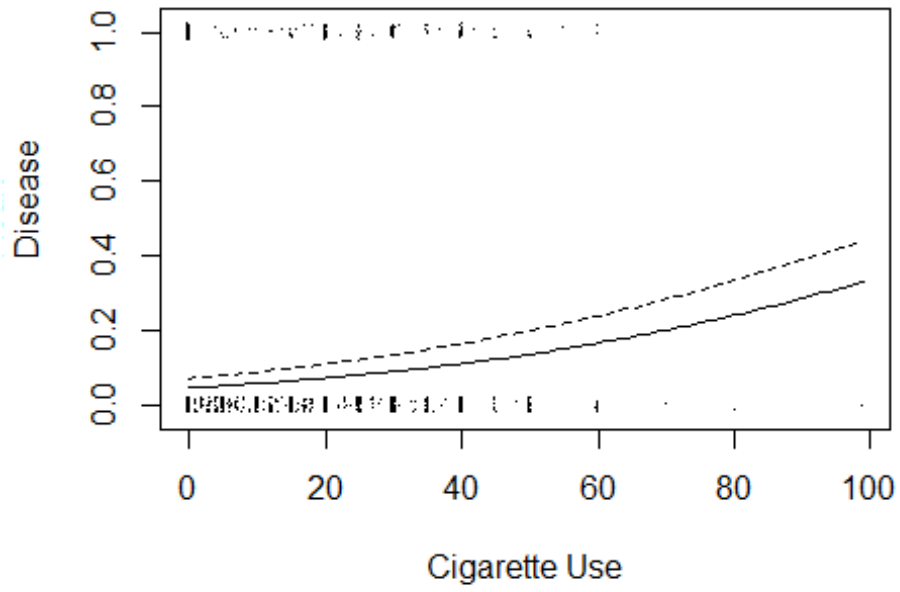
➔ Solid Line 이 비흡연자, Dash line 이 흡연자를 나타난다.

➔ 비흡연자와 흡연자 모두 키가 커질수록 근소하지만 심장병 발병 확률이 높아지는 것을 알 수 있다(거의 선형적).

➔ Dash line 이 Solid line 보다 위에 위치하는 것을 보았을 때 흡연자는 비흡연자에 비해 심장병 발병 확률이 높다는 것을 알 수 있다.

이번에는 Height 변수를 고정하고 Cigs 변수의 변화에 따른 심장병 발병 확률 변화에 대해서 살펴보자. 키가 60 인치로 작은 사람과 78 인치로 큰 사람에 대해 Cig 변수 변화에 따른 심장병 발병 확률 변화이다.

```r
plot(jitter(y, 0.1) ~ jitter(cigs), wcgs, xlab='Cigarette Use', ylab='Heart
     Disease', pch='.')
curve(ilogit(beta[1] + beta[2]*60 + beta[3]*x), add=TRUE)
curve(ilogit(beta[1] + beta[2]*78 + beta[3]*x), add=TRUE, lty=2)
```



➔ Solid line 이 60 인치인 사람이고 Dash line 이 78 인치인 사람이다.

➔ Solid line 이 Dash line 밑에 위치한 것을 보아 키가 크면 심장병 발병 확률이 더 높다는 것을 알 수 있다.

➔ 또한 Cigarette Use 가 높아짐에 따라 심장병 발병 확률을 기하급수적으로 상승한다.

##2-5. Odds 해석

이제 Coefficient 값들에 대해 해석을 해야 되는데, linear model 때와는 해석방식이 다르며 우선 odds 에 대해 이해할 필요가 있다.

Odds 는 어떠한 가능성을 표현해주는 확률의 대체적인 scale 이다.

즉, 가능성을 나타내지만 정확히 확률과 동일하지는 않다.

Odds 는 어떤 일이 일어날 확률을 일어나지 않을 확률로 나눠준 값이다.

Odds=p/(1-p)

Odds 의 수학적인 이점은 확률값과 달리 bounded above 가 정해지지 않았다는 점이고, 따라서 어떠한 modeling 목적에서는 더 간편하다는 것이다.

Logistic Model 에서 odds 와 변수들 간의 관계를 살펴보자.

$$\log(odds) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

따라서

$$odds = e^{\beta_0} e^{\beta_1 x_1} e^{\beta_2 x_2}$$

라고 할 수 있다. 이에 따라 β_1을 해석하면 다음과 같다.

x_2값이 고정되어 있고 x_1이 한 단위 상승하면 성공의 log-odds 의 값이 β_1값만큼 상승한다.

또는 성공의 odds 가 e^{β_1}값만큼 상승한다.

위의 해석 결과를 보면 coefficient 에 exponential 을 취해주는 것이 해석에 더 용이할 수 있다는 것을 알 수 있다.

```
exp(beta)
```

```
## (Intercept)      height        cigs
##  0.01109108  1.02552819  1.02339691
```

이에 따르면 키가 1 인치 커질수록 심장병 발병의 odds 는 2.6% 높아지며, 하루에 피는 담배의 개수가 한 개 늘수록 odds 가 2.3%높아진다.

하루 한 갑의 영향은 다음과 같다.

```
exp(beta[3]*20)
```

```
##     cigs
## 1.588115
```

하루에 한 갑을 피면 그렇지 않을 때보다 심장병 발병의 odds 가 59%나 높아짐을 알 수 있다.

Odds 가 직관적으로 이해되지 않는다면 Relative Risk 를 통해 가능성을 비교할 수도 있다. Relative Risk 는 어떤 상황에서의 확률값을 다른 상황에서의 확률값으로 나눠준 것으로 Odds 보다 직관적이다. 만약 이 Relative Risk 가 1 보다 크다면 전자의 상황이 일어날 확률이 후자의 상황보다 높다는 것을 의미한다.

키는 68 인치로 동일하고 흡연 유무에 따른 Relative Risk 를 살펴보자. 흡연의 유무는 하루 담배 한 갑을 기준으로 한다.

ilogit command 를 통해 p 값을 구할 수 있으므로 다음을 이용한다.

```
c(ilogit(sum(beta*c(1,68,20))), ilogit(sum(beta*c(1,68,0))))
```

```
## [1] 0.08907868 0.05800425
```

```
ilogit(sum(beta*c(1,68,20)))/ilogit(sum(beta*c(1,68,0)))
```

```
## [1] 1.535727
```

결과의 값이 1 을 넘으므로 전자가 일어날 확률이 더 높다는 것을 의미한다. 즉, 담배를 피면 그렇지 않을 때보다 심장병이 발병할 확률이 높다.

위의 값은 앞서 구했던 odds ratio 의 값인 1.59 와 거의 동일하다는 것을 알 수 있다.

성공확률이 낮을 때, relative risk 와 odds ratio 의 값은 거의 동일하다. 하지만 성공확률이 크다면, 둘의 값에는 차이가 많이 나게 된다.

#3. Inference

##3-1. Current Model Test

l 개의 parameter 와 L_L의 likelihood 를 가진 larger model 과 s 개의 parameter 와 L_S의 likelihood 를 가진 smaller model 을 고려해보자(이 때 l > s).

그럼 이 때 두 모델을 비교하기 위한 test statistic 인 Likelihood Ratio statistic 은 다음과 같다.

$$2\log \frac{L_L}{L_S}$$

이번에는 Deviance 에 대해 알아보자.

Deviance 란 어떤 모형의 log-likelihood 에서 saturated model(모든 변수들이 포함된 model)의 log-likelihood 를 뺀 것에 -2 를 곱한 값이다.

Logistic Regression Model 에서 이를 수식으로 표현하면

$$D = -2\sum_{i=1}^{n} \hat{p}_i logit(\hat{p}_i) + \log(1-\hat{p}_i)$$

이다. 이 때 $\hat{p}_i$는 saturated 가 아닌 model 의 fitted value 이다.

다른 GLM 의 예시에서 Deviance 는 바로 모델의 적합도를 나타내는 데에 사용할 수 있지만, 이 case 에서 Deviance 값은 단순히 fitted values 의 function 일 뿐이므로 Deviance 값을 이용해서 다른 값을 도출해야 한다.

우리는 Deviance 를 두 nested model 을 비교하는 데에 사용할 수 있다.

앞서 언급했던 Likelihood Ratio Statistic 은

$D_s - D_L$로 생각할 수 있다.

그리고 smaller model 이 맞다는 가정하에서 $D_s - D_L$은 asymptotic 하게 χ^2_{l-s}분포를 따른다.

그럼 우리의 예시를 가지고 Model test 를 해보자.

우리의 모델(height 와 cig 변수가 포함된 모델)의 결과값을 살펴보면,

```
Deviance = 1749.04923 Null Deviance = 1781.24374 (Difference = 32.19451)
```

를 확인할 수 있다.

여기서 `Null Deviance` 는 변수가 하나도 포함되지 않았을 때의 `Deivance` 값을 의미한다.

그러면 여기서 $D_s - D_L$ 은 약 `32.2` 이고 자유도는 `2` 라는 것을 알 수 있다(변수가 `2` 개).

따라서 적어도 하나의 `predictor` 가 `response` 와 관련이 있을 것이라는 `test` 에 대한 `p-value` 값은 다음과 같이 구할 수 있다.

```
1-pchisq(32.2,2)
```

```
## [1] 1.01826e-07
```

`Value` 가 매우 작으므로 귀무가설을 기각한다. 즉, 우리가 고른 `predictor` 와 `response` 간에 어떠한 관계가 있다.

##3-2. Individual Predictor Test

앞선 방법은 우리의 현재 모델을 통으로 `test` 하는 방법이었다면 우리가 선택한 `predictor` 들 각각에 대해서 `test` 를 할 수도 있다.

이 방식은 앞서 `Chapter.1` 에서 시행했던 `anova command` 를 이용했던 방식과 동일하다.

```r
lmodc <- glm(chd ~ cigs, family= binomial, wcgs)
anova(lmodc, lmod, test='Chi')
```

```
## Analysis of Deviance Table
##
## Model 1: chd ~ cigs
## Model 2: chd ~ height + cigs
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      3152       1750
## 2      3151       1749  1  0.92025   0.3374
```

p-value 값이 여기서는 0.05 이상이다. 즉, 귀무가설을 기각하지 않으므로 model1 을 그대로 채택한다. 즉, Height 는 통계적으로 유의하지 않다.

drop1 방식을 logistic regression 에서도 사용할 수 있다.

```r
drop1(lmod, test='Chi')
```

```
## Single term deletions
##
## Model:
## chd ~ height + cigs
##        Df Deviance    AIC     LRT Pr(>Chi)
## <none>      1749.0 1755.0
## height  1   1750.0 1754.0  0.9202   0.3374
## cigs    1   1780.1 1784.1 31.0695 2.49e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

여기서도 Cigs 변수는 통계적으로 유의한 반면, height 변수는 통계적으로 유의하지 않다는 것을 알 수 있다.

##3-3. Confidence Interval for the regression parameters

β_i에 대한 Confidence Interval 은 다음과 같다.

$\widehat{\beta_i} \pm z^{\alpha/2} se(\widehat{\beta_i})$

(여기서 $se(\widehat{\beta_i})$는 standard error, $z^{\alpha/2}$는 normal distribution 에서의 quantile)

β_1에 대한 95% confidence interval 을 수동적으로 구해보면,

```r
0.02521 + c(-1,1) * 1.96 * 0.02633
```

```
## [1] -0.0263968  0.0768168
```

일일이 매번 수동적으로 confidence interval 을 구하는 것은 어렵다. 따라서 confint command 를 활용하면 쉽게 구할 수 있다.

```
confint(lmod)
```

```
## Waiting for profiling to be done...
```

```
##                   2.5 %      97.5 %
## (Intercept) -8.13475465 -0.91297018
## height      -0.02619902  0.07702835
## cigs         0.01514949  0.03100534
```

#4. Diagnostics

Regression Diagnostics 은 우리의 모델에 대한 가정들을 확인하고 특이한 지점이 있는 지를 확인하는 데에 유용하다.

Diagnostic 에서 가장 기본적이고 중요한 것은 Residuals(difference between observed and fitted values)를 살펴보는 것이다. Residuals 를 통해 우리의 모델이 얼마나 데이터에 잘 맞고(fit) 개선점이 있는 지를 살펴볼 수 있다.

##4-1. Residuals and Plots

Fitted Values 는 predict command 를 통해 구할 수 있다. 그런데 단순히 predict function 을 사용하면 우리가 얻는 것은 linear predictor 에 대한 결과값, 즉 η값을 얻게 된다. 따라서 type='response'라는 argument 를 추가해서 확률값을 얻도록 하자. η값을 얻은 뒤 ilogit command 를 통해 p 값을 얻을 수도 있다.

```
linpred <- predict(lmod)
predprob <- predict(lmod, type='response')
```

```
head(linpred)
```

```
##      2001      2002      2003      2004      2005      2006
## -2.083261 -2.274521 -2.762277 -2.324936 -2.274521 -2.686653
```

```
head(predprob)
```

```
##       2001       2002       2003       2004       2005       2006
## 0.11073449 0.09325523 0.05939705 0.08907868 0.09325523 0.06376553
```

```
head(ilogit(linpred))
```

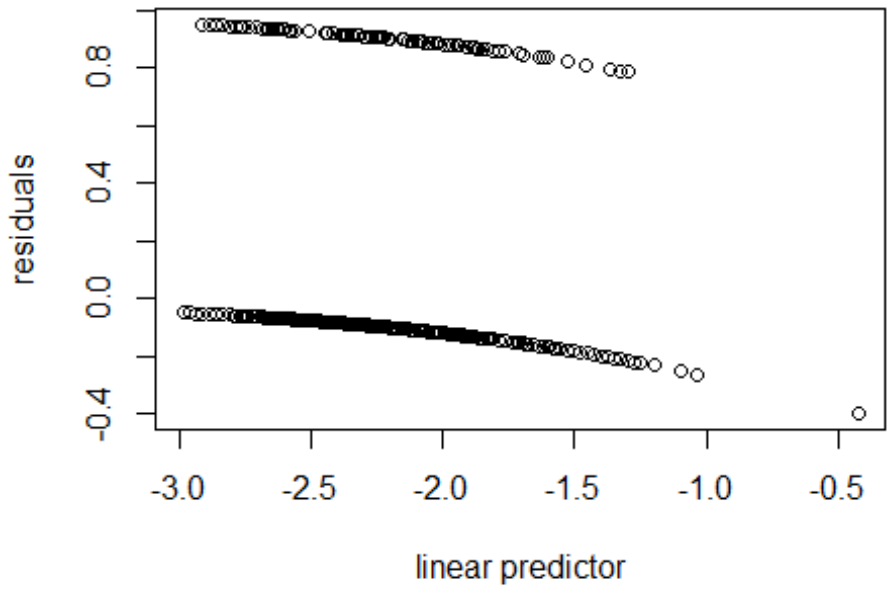
```
##        2001        2002        2003        2004        2005        2006
## 0.11073449 0.09325523 0.05939705 0.08907868 0.09325523 0.06376553
```

Residuals 는 observed 값에서 fitted value 를 뺀 것이므로 다음과 같이 구할 수 있다.

```
rawres <- wcgs$y - predprob
```

그리고 구해진 residuals 를 linear predictor 값과 비교해보자.

```
plot(rawres ~ linpred, xlab='linear predictor', ylab='residuals')
```



➔ 그런데 이 그래프는 그다지 쓸모가 없다. 왜냐하면 y=0 또는 1 로, 주어진 linear predictor 값에 대해 residual 은 오직 두 개의 값만 취할 수 있기 때문이다.

➔ Upper line 은 y=1 에 해당하고 Lower line 은 y=0 에 해당한다.

➔ 우리는 이 그래프를 통해 모델 적합도와 관련해서 insight 를 얻기 힘들다.

위와 같은 문제가 나타난 이유는 residuals 를 잘못 이해했기 때문이다. Raw Residuals 를 이용한다면 위와 같이 plot 에서 insight 를 얻기 어려울 뿐만 아니라, 그들이 등분산을 가지고 있을 것이라는 기대를 할 수 없다. 따라서 우리는 standardization 을 해주어야 한다.

Standard linear model 에서 우리는 residual sum of squares($\sum_i \hat{\varepsilon}_i^2$)를 구했는데, Logistic Regression 에서 이에 대응되는 것이 deviance 이다. 그리고 deviance = $\sum_i r_i$ 이다. 그리고 이 때 r_i 값을 deviance residual 이라 부른다.

Deviance residual 은 각 observation 이 총 deviance 에 기여하는 정도를 나타낸다.

그리고 $r_i = sign(y_i - \hat{p}_i)\sqrt{r_i^2}$ 이다.

Deviance residual 은 logistic model 에서 residuals 를 이용하면 구할 수 있다.

따라서 이를 이용해서 새롭게 residuals 를 구하고 이를 linpred 와 비교해서 plot 을 그려보자.

우리의 data frame 에 deviance residual 과 linpred 의 값에 대한 행렬을 추가해주자.

이 때 dyplr packagae 의 mutate function 을 이용할 수 있다.

```
wcgs <- mutate(wcgs, residuals=residuals(lmod), linpred = predict(lmod))
```

그 다음 모든 observation 값들을 살펴볼 수도 있지만, 비슷한 linpred 를 가지고 있는 값들끼리 group 화를 시켜줘서 확인해보자. Cut command 와 breaks argument 를 활용하면 어떤 값의 범위를 breaks 를 기준으로 하여 나눌 수 있다. 여기서 우리는 linpred 의 범위를 100 개로 나누어서 약 30 개의 값들이 각각의 구간에 들어갈 수 있도록 해보자(우리의 observation 의 개수가 약 3000 개). 그리고 최종적으로 그룹화를 시켜주는 command 는 group_by 를 사용한다.

```
gdf <- group_by(wcgs, cut(linpred, breaks=unique(quantile(linpred, (1:100)/101))))
```

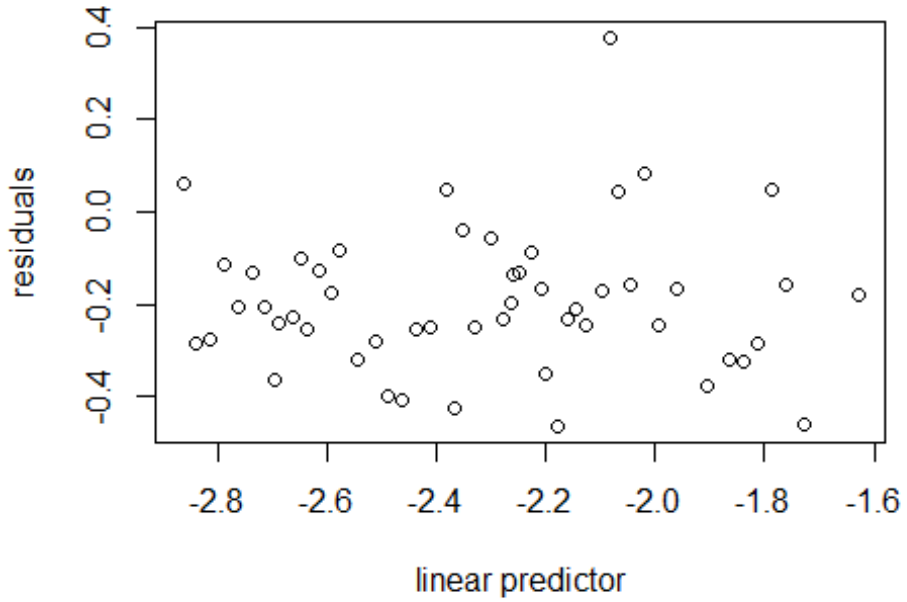
각 그룹에 대해서 residuals 와 linpred 의 mean 을 구해주도록 하자.

```
diagdf <- summarise(gdf, residuals=mean(residuals), linpred=mean(linpred))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

그 다음 plot 을 그리면

```
plot(residuals ~ linpred, diagdf, xlab='linear predictor')
```



➔ Deviance residuals 는 평균이 0 이라는 제약 조건이 없으므로 mean level 은 고려하지 않아도 된다.

➔ Linear predictor 의 변화에 따라 even variation 을 보이는 것을 알 수 있다. 따라서 모델의 부적절성을 찾을 수 없다.

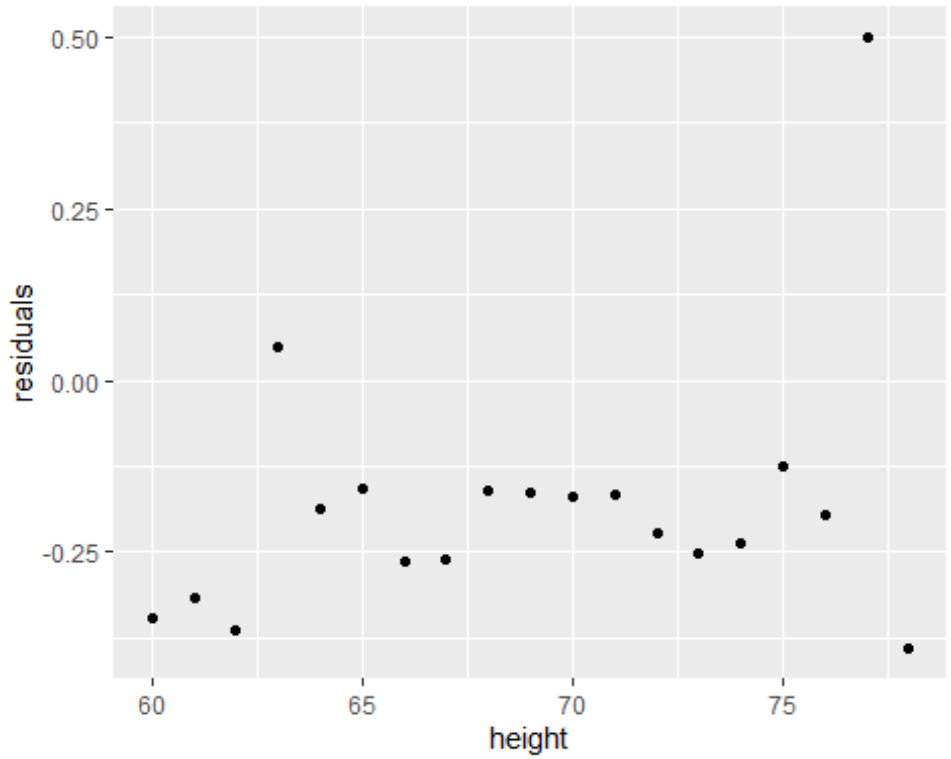
우리는 또한 binned 된 residuals 와 predictor 간의 plot 을 그릴 수도 있다.

우선 height 를 기준으로 group 화를 시키고 height 와 residual 간의 관계를 살펴보자.

```
gdf <- group_by(wcgs, height)
diagdf <- summarise(gdf, residuals=mean(residuals))

## `summarise()` ungrouping output (override with `.groups` argument)

ggplot(diagdf, aes(x=height, y=residuals)) + geom_point()
```



➔ 77in tall 에서 극단적으로 높은 residual 이 있다는 것 말고는 특이점이 없다.

77in tall 값에서 특이점을 발견했으니 dplyr package 에 있는 filter command 를 이용해서 키가 77 인 그룹의 값들을 골라내자. 이 때 추가적으로 select command 를 이용해서 해당 변수의 값들만 확인할 수 있다.

```
filter(wcgs, height==77) %>% select(height, cigs, chd, residuals)
```

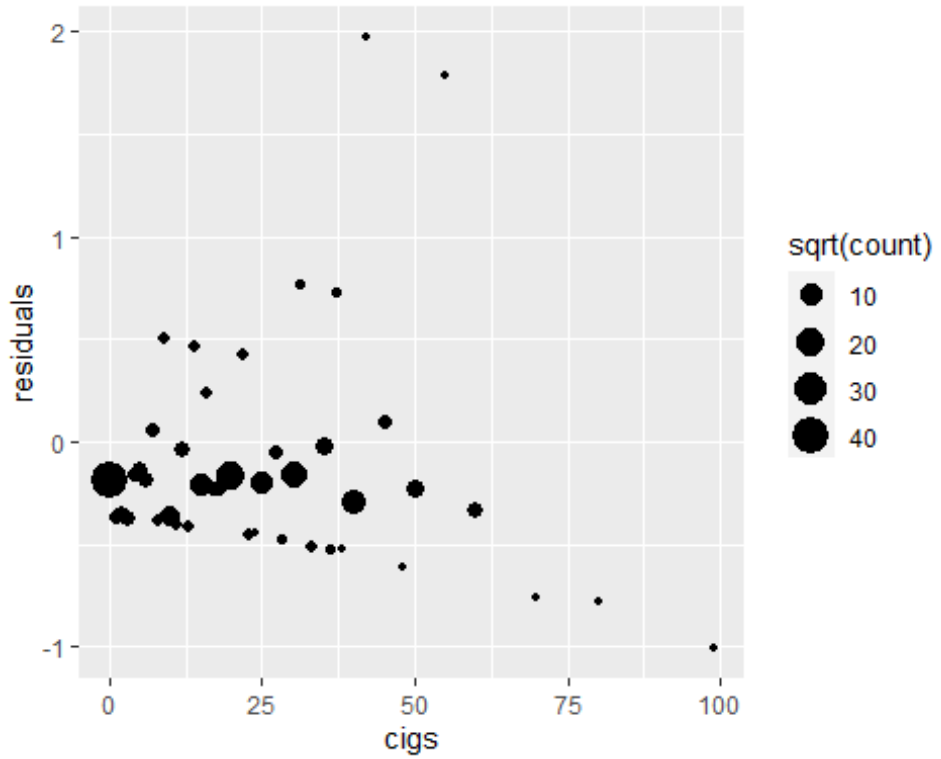
```
##   height cigs chd  residuals
## 1     77    0  no -0.3857933
## 2     77    0 yes  2.2956622
## 3     77    5  no -0.4078515
```

여기서 우리는 키가 77 인 경우는 세 가지 경우밖에 없으며 그 중 단 한 가지 경우만 심장병 발병이 일어났음을 알 수 있다. 따라서 residual plot 에서 이 point 는 예외적이지 않다.

이번에는 흡연량을 기준으로 살펴보자. 방식은 동일하다. 다만 이번에는 n function 을 사용해서 residuals 의 수를 세주었고 이를 이용해서 각 point 들의 크기가 residuals 의 수를 반영할 수 있도록 해주었다.

```
group_by(wcgs, cigs) %>% summarise(residuals=mean(residuals), count=n()) %>%
  ggplot(aes(x=cigs, y=residuals, size=sqrt(count))) + geom_point()
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



➔ 흡연을 하지 않는 사람들의 수가 많아서 0 에 가까운 point 들의 크기가 큼을 알 수 있다.

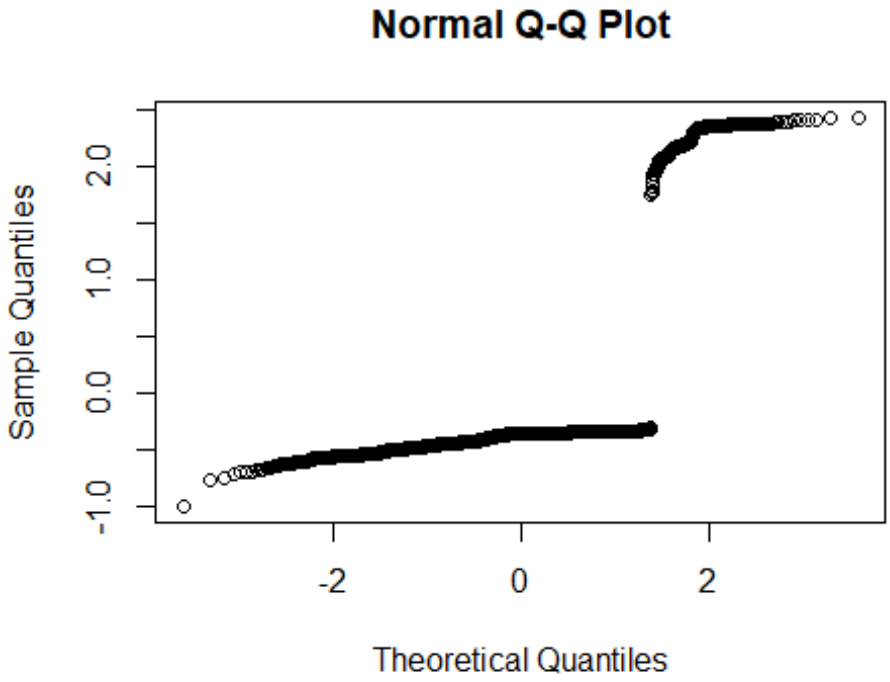
➔ Unusual point 들이 보이지만, 이들의 크기가 상대적으로 꽤 작기 때문에 무시해도 좋을 정도라고 생각할 수 있다.

➔ 따라서 lack of fit 또는 predictor 의 변형에 대한 필요성을 보이고 있지는 않다고 할 수 있다.

```
이번에는 unusual point 의 발견에 초점을 맞춰보자.

Standard linear model 에서 그랬던 것처럼 이번에도 QQ plot 을 그려서 확인해보자.

qqnorm(residuals(lmod))
```

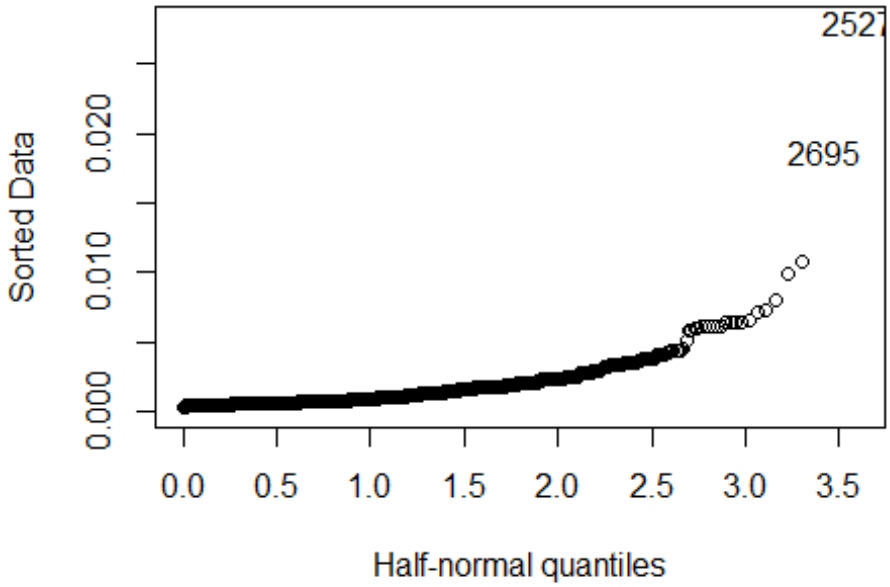


➔ Y=0 과 1 에 대응하여 두 개의 cluster 가 형성된 것을 확인할 수 있다.

➔ 그러나 우리는 residual 들이 normally distributed 하다는 가정을 한 적이 없기 때문에 이는 딱히 문제가 되지 않는다.

우리는 standard linear model 에서 leverage 를 살펴봄으로써 predictor 공간에서 특이점을 찾았었다. 이번에도 같은 방식을 사용할 수 있다.

```
halfnorm(hatvalues(lmod))
```



➔ 두 포인트가 특이점을 보인다.

특이점을 보이는 두 hatvalue 의 값이 약 0.015 를 넘는 것으로 보이므로 hatvalue 가 0.015 를 넘는 point 들을 filter 를 통해 찾아보자.

```
filter(wcgs, hatvalues(lmod) > 0.015) %>% select(height, cigs, chd)
```

```
##   height cigs chd
## 1     71   99  no
## 2     64   80  no
```

흡연량이 매우 높은 case 라는 것을 확인할 수 있다.

Dataset 이 상대적으로 크고 두 case 의 경우가 그렇게 extreme 하지 않다는 것을 고려하면 특별히 문제가 되는 것은 없다고 생각할 수 있다.

#5. Model Selection

우리는 키와 흡연 두 가지 변수만 현재 predictor 로 사용했지만 다른 변수들을 사용한다면 response 에 대해 더 좋은 model 을 세울 수도 있다.

물론 그 변수들은 현재 가지고 있는 변수들일 수도 있지만 현재는 없고 새롭게 만들어낸 변수일 수도 있다.

하지만 여기서는 변수를 새롭게 만드는 것보다 어떻게 하면 현재 가지고 있는 변수들 중에서 제일 괜찮은 subset 을 구할 수 있는 지에 대해서 알아보자.

우선 Backward Elimination 방법이 model choosing 의 한 방법으로 사용될 수 있지만 future response 를 가장 잘 예측하는 set 을 찾을 수는 없다. 따라서 Backward Elimination 은 선호되지 않는다.

Chapter 1 에서 사용되었던 기준인 AIC 가 여기서도 똑같이 사용될 수 있었다.

그 때 AIC 는

$AIC = -2logL + 2q$ (q 는 predictor 의 개수)

였는데 deviance 의 관점에서 보면

$AIC = deviance + 2q$

로 쓸 수 있다.

그리고 step command 를 사용하면 AIC 를 minimize 해주는 model 을 찾아준다.

우선 bmi 라는 새로운 변수를 추가하고 full model 을 만들어보자.

```
wcgs$bmi <- with(wcgs, 703*wcgs$weight/(wcgs$height^2))
lmod <- glm(chd ~ age + height + weight + bmi + sdp + dbp + chol + dibep +
              cigs + arcus, family=binomial, wcgs)
```

만들어진 full model 을 가지고 step command 를 사용하자. 이 때 trace=0 을 사용하면 중간 과정이 print 되는 것을 방지할 수 있다.

```
lmodr <- step(lmod, trace=0)
```

그렇게 만든 model 을 살펴보면 다음과 같다.

```
sumary(lmodr)
```

```
##                 Estimate  Std. Error z value  Pr(>|z|)
## (Intercept)  -15.9575989   2.2860760 -6.9803 2.945e-12
## age            0.0615904   0.0123968  4.9683 6.756e-07
## height         0.0501608   0.0278236  1.8028   0.07142
## bmi            0.0603846   0.0265986  2.2702   0.02319
## sdp            0.0177284   0.0041547  4.2671 1.981e-05
## chol           0.0107089   0.0015285  7.0062 2.450e-12
## dibepB         0.6576159   0.1458984  4.5074 6.564e-06
## cigs           0.0210406   0.0042625  4.9363 7.963e-07
```

```
## arcuspresent   0.2109985   0.1437175  1.4681   0.14206
##
## n = 3140 p = 9
## Deviance = 1569.32520 Null Deviance = 1769.17129 (Difference = 199.84609)
```

Height 와 diastolic blood pressure 변수만 제외된 것을 확인할 수 있다.

그런데 diastolic blood pressure 가 heart disease 와 관련이 없기 때문에 제외된 것이라고 이해해서는 안 된다. 위에서 diastolic blood pressure 가 제외된 이유는 단지 AIC 를 minimize 하는 데에 그게 도움이 되기 때문이다. drop1 을 이용해서 response 와의 관계를 살펴보면

```
drop1(glm(chd ~ dbp, family=binomial, wcgs), test='Chi')
```

```
## Single term deletions
##
## Model:
## chd ~ dbp
##        Df Deviance    AIC    LRT  Pr(>Chi)
## <none>      1751.7 1755.7
## dbp     1   1781.2 1783.2 29.548 5.454e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

p-value 값이 매우 낮으므로 통계적으로 유의하다는 것을 알 수 있다.

따라서 step command 방법이 무조건 좋은 방식은 아니라는 것을 알 수 있다.

#6. Goodness of Fit

앞서 언급했듯이 binary response GLM 에서 deviance 는 measure of fit 으로 사용할 수 없다. 그래서 우리는 binned residuals 를 이용해서 diagnostic plots 을 그렸는데, 이 또한 model 이 잘 fit 하는 지에 대해서는 말해주지 않는다. 다만 그렇더라도 binning 은 model fit 에 관한 test 를 개발하는 데에 도움이 된다.

몇 가지를 가정해보자. 우선 observation 값들을 linear predictor 에 기반하여 J 개의 bin 을 기준으로 나눴다고 하자. 그리고 이 때 j 번째 bin 안의 observation 들의 mean response 를 y_j 라고 하고 mean predicted probability 를 $\hat{p_j}$라고 하자. 이 때 bin 안의 observation 의 개수는 m_j라고 한다. 이 값들을 계산해보자.

```
wcgsm <- na.omit(wcgs)
```

우선 dataset 에 결측치가 존재하여 이를 na.omit command 를 이용해서 제거해주었다.

```
wcgsm <- mutate(wcgsm, predprob=predict(lmod, type='response'))
```

```
gdf <- group_by(wcgsm, cut(linpred, breaks=unique(quantile(linpred, (1:100)/1
01))))
```

Linpred 를 기준으로 binning 해주는 방식은 이전과 동일

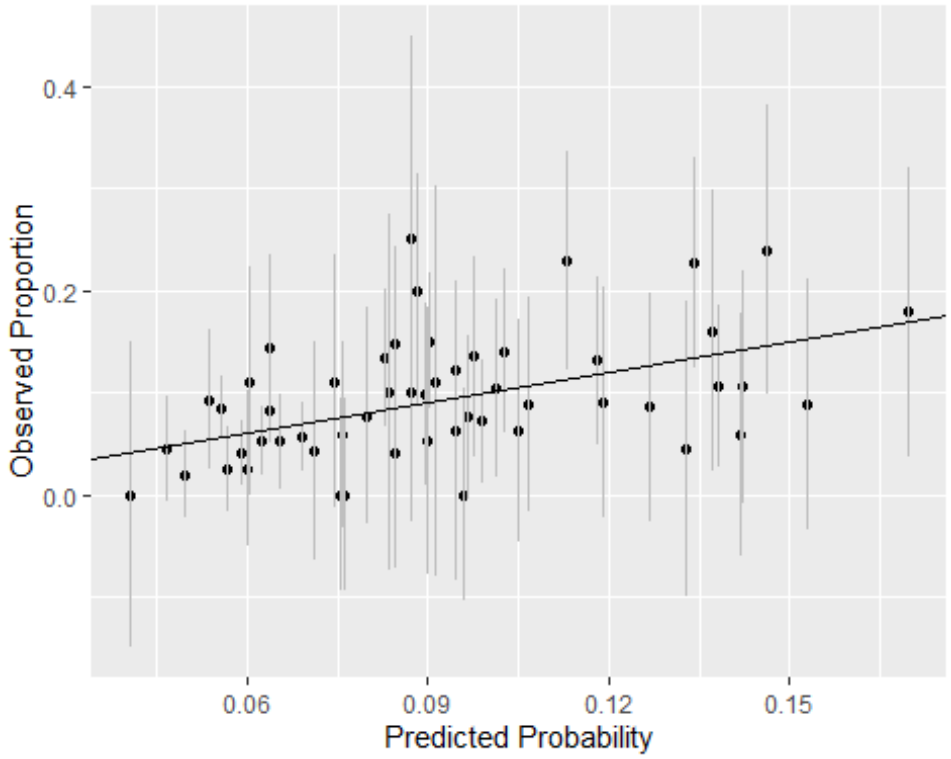
추가적으로 심장병 발병 횟수, bin 내의 observations 들의 개수, mean predicted probabilities 또한 계산해주었다.

```
hldf <- summarise(gdf, y=sum(y), ppred=mean(predprob), count=n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

우리가 이제 구하고자 하는 것은 observed proportion 이다. 우리가 probability p 를 구할 때 우리는 실제 발생 비율이 그 p 값에 대응되기를 바란다. 즉, observed proportion 과 predicted probability 를 비교한다면 우리는 우리의 모델이 얼마나 실제 데이터에 잘 fit 하는가를 판단할 수 있다.

```
hldf <- mutate(hldf, se.fit=sqrt(ppred*(1-ppred)/count))
ggplot(hldf, aes(x=ppred, y=y/count, ymin=y/count-2*se.fit, ymax=y/count+2*se.
fit)) +
  geom_point() + geom_linerange(color=grey(0.75)) + geom_abline(intercept=0,
slope=1) +
  xlab('Predicted Probability') + ylab('Observed Proportion')
```



➔ 비록 약간의 variation 이 보이긴 하지만 consistent deviation 이 보이지는 않는다.

➔ 또한 approximate 95% confidence interval 을 binomial variation 을 사용하여 그려주었는데 line 이 대부분의 confidence interval 을 지나가는 모양을 확인할 수 있다. Variation from the expected 가 크지 않다는 것을 확인시켜준다.

Hosmer-Lemeshow statistic 은 이러한 평가방법을 수식으로 정형화시킨 것이다.

$$X^2_{HL} = \sum_{j=1}^{J} \frac{(y_i - m_j\hat{p}_j)^2}{m_j\hat{p}_j(1-\hat{p}_j)}$$

이 statistic 은 approximate 하게 카이제곱 분포를 따르며 이 때의 자유도는 J-1 이다.

이 때 카이제곱 근사의 정확도를 어느정도 보장하기 위해서는 bin 마다 충분한 observation 이 필요하며 그렇다고 너무 적은 bin 개수는 test 를 하기 힘들어지므로 적절하게 정하는 것이 중요하다.

HL statistic 을 구해보면,

```
hlstat <- with(hldf, sum( (y-count*ppred)^2/(count*ppred*(1-ppred))))
c(hlstat, nrow(hldf))
```

```
## [1] 62.83215 54.00000
```

```
1-pchisq(63.83215, 54-1)
```

```
## [1] 0.1671565
```

p-value 가 0.05 이상이기 때문에 lack of fit 은 발견되지 않았다.

우리는 binning method 에 의존하지 않고 prediction 의 quality 를 계산하고 싶을 수 있다. 그럴 때 사용되는 것이 예측한 결과와 실제 값을 table 로 만들어 비교하는 것이다. 이 때 예측한 결과는 probability 가 아니라 그 확률 값을 바탕으로 no 일지 yes 일지 결정한 값을 의미한다.

일반적으로 0.5 를 기준으로 낮다면 no 를, 높다면 yes 로 분류(classify)한다. 이를 이용하여 table 을 만들어보면,

```
wcgsm <- mutate(wcgsm, predout=ifelse(predprob < 0.5, 'no', 'yes'))
xtabs( ~ chd + predout, wcgsm)
```

```
##       predout
## chd      no  yes
##   no   2882    3
##   yes   253    2
```

위의 table 을 바탕으로 Correct Classification rate 를 구하면 다음과 같다.

```
(2882+2)/(2882+3+253+2)
```

```
## [1] 0.9184713
```

이는 바꿔 말하면, error 또는 misclassification rate 가 약 8%에 지나지 않는다는 것을 의미한다.

이러한 방법은 model fit 을 판단하는 데에 유용하지만 중요한 variation 을 놓칠 위험이 있다.

우선 Specificity 와 Sensitivity 의 정의에 대해 알아보자. Specificity 란 원래 No 인 것을 제대로 No 라고 분류한 비율을 의미한다. 여기서 Specificity 값은 2882/(2882+3)=0.999 로 매우 높다. 즉 심장병이 발병하지 않을 경우를 잘 예측한 것이다.

반대로 Sensitivity 란 실제로 Yes 인 것을 잘 예측하는 비율이다. 여기서 Sensitivity 값은 2/(253+2)=0.00784 이다. 즉, 실제로 심장병이 발병하는 것을 예측하는 비율은 매우 낮다는 것을 알 수 있다. 단순히 overall error rate 만 고려한다면 이러한 문제를 발견할 수 없었을 것이다.

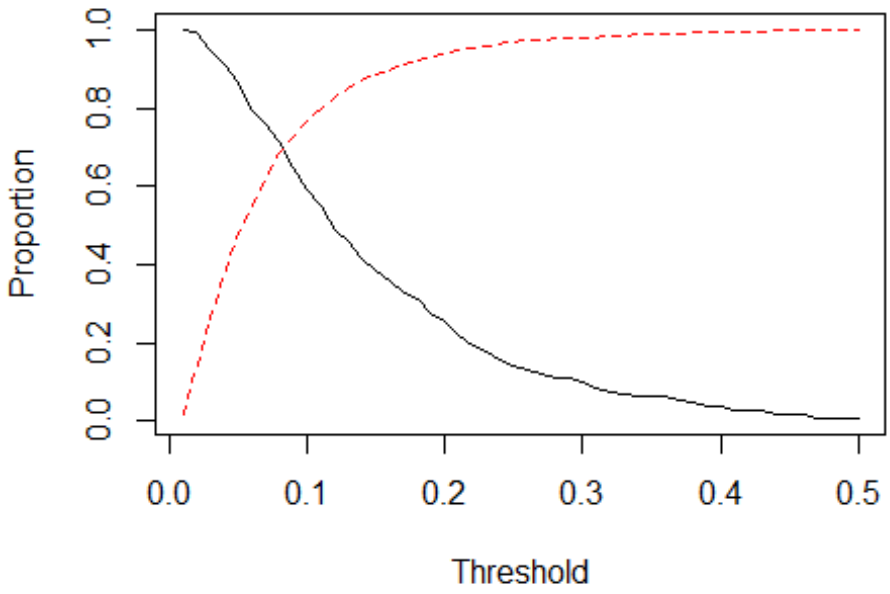
우리는 위에서 설정했던 classification 의 기준인 0.5(이를 threshold 라고 한다)를 조정해줌으로써 Sensitivity 를 높일 수 있다. Threshold 변화에 따른 그래프를 그려보자.

```
thresh <- seq(0.01, 0.5, 0.01)
```

threshold 의 값을 0.01 부터 0.5 까지로 설정한다.

```
Sensitivity <- numeric(length(thresh))
Specificity <- numeric(length(thresh))
for(j in seq(along=thresh)){
  pp <- ifelse(wcgsm$predprob < thresh[j], 'no', 'yes')
  xx <- xtabs( ~ chd + pp, wcgsm)
  Specificity[j] <- xx[1,1]/(xx[1,1]+xx[1,2])
  Sensitivity[j] <- xx[2,2]/(xx[2,1]+xx[2,2])
}

matplot(thresh, cbind(Sensitivity, Specificity), type='l', xlab='Threshold',
        ylab='Proportion', lty=1:2)
```



➔ Solid Line 은 Sensitivity 를, Dash Line 은 Specificity 를 나타낸다.

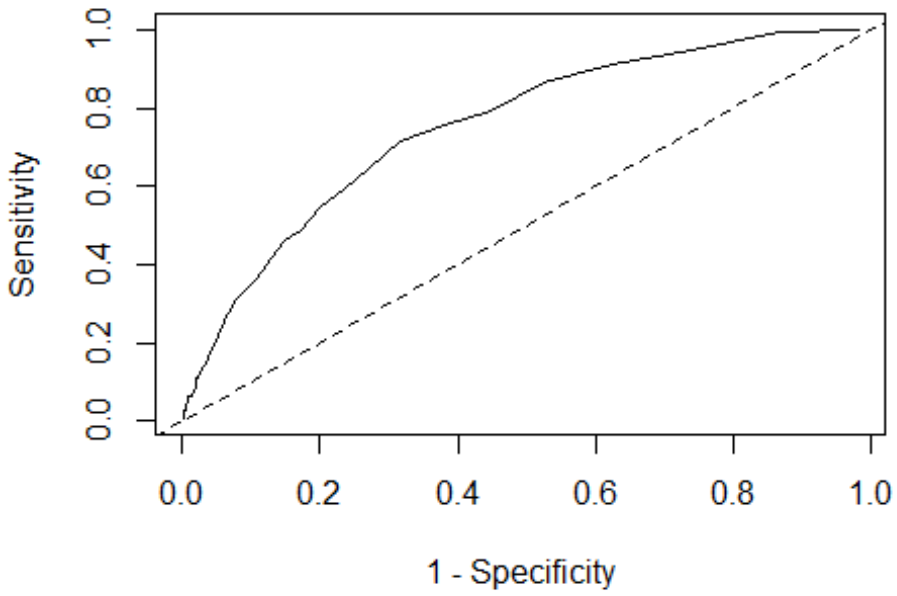
➔ Sensitivity 는 Threshold 가 높아질수록 내려가는 반면, Specificity 는 높아진다는 것을 알 수 있다.

➔ 우리의 경우에는 Sensitivity 를 높여야 하므로 Threshold 를 낮춰야 한다.

대체적인 방법으로 위와 같은 정보를 보여주는 것을 receiver operating characteristic(ROC) curve 라고 한다. ROC Curve 에서는 Sensitivity(=true positive rate)와 1-Specificity(=false positive rate)의 관계를 보여준다.

이를 실제로 그려보면

```
plot(1-Specificity, Sensitivity, type='l')
abline(0,1,lty=2)
```



➔ Dash line(y=x)은 useless test 즉, 반반으로 랜덤으로 찍은 test 를 의미한다.

➔ Solid Line 이 왼쪽 구석에 가까이 붙어있을수록 좋은 test 라고 볼 수 있다.

➔ 따라서 ROC Curve 밑 쪽의 영역을 계산하면, 다른 test 와 비교했을 때 어떤 test 가 더 좋은 지를 판별할 수 있는 기준이 된다.

Normal Linear Model 에서 Model 의 적합도를 측정할 때 흔히 사용되는 방법은 proportion of variance explained(R square)이다. 비슷하게 Binomial Regression 에서는 proportion of deviance explained 를 이용할 수 있지만 더 좋은 statistic 이 존재한다.(Nagelkerke's R Square)

$$R^2 = \frac{1 - \left(\frac{\hat{L}_0}{\hat{L}}\right)^{\frac{2}{n}}}{1 - \hat{L}_0^{\frac{2}{n}}} = \frac{1 - \exp\left(\frac{D - D_{null}}{n}\right)}{1 - \exp\left(-\frac{D_{nll}}{n}\right)}$$

이 때 n 은 binary observation 의 개수이며 $\hat{L}_0$은 null 상황 하에서 maximized likelihood 이다.

이 값이 높을수록 우리의 모델이 데이터에 잘 fit 하다는 것을 의미한다.

```
lmodr <- glm(chd ~ age + height + bmi + sdp + chol + dibep + cigs + arcus,
             family=binomial, wcgs)
(1-exp((lmodr$dev - lmodr$null)/3140))/(1-exp(-lmodr$null/3140))
```

```
## [1] 0.1431517
```

값이 매우 낮은 것으로 나타난다. 하지만, 기준을 Standard Linear Model 때와 동일하게 생각하면 안 된다. Natural Variance 하에서 binary response model 의 경우 Nagelkerke 의 R Square 또는 다른 R Square substitute 들의 값들은 model 이 좋은 경우에도 낮은 경우가 흔하다. 따라서 절대적인 것보다는 model 끼리 비교할 때 사용하는 것이 좋다.

#7. Estimation Problems

Logistic Regression model estimation 에 있어서 종종 문제가 발생할 수 있는데 이를 iris 데이터를 이용해서 살펴보자.

우선 iris 데이터에서 setosa 와 versicolor 두 종만 선택하고 predictor 도 sepal.width, sepal.length 만 고려한다고 하자.

```
irisr <- filter(iris, Species != 'virginica') %>% select(Sepal.Width,
                                                         Sepal.Length, Specie
s)
```

Sepal.width 와 Sepal.length 에 따른 두 종의 분포는 다음과 같이 그릴 수 있다.

```
(p <- ggplot(irisr, aes(x=Sepal.Width, y=Sepal.Length, shape=Species)) + geom
_point())
```

이제 두 sepal dimension 을 통해 species 를 예측할 수 있는지 logistic regression model 을 fit 해서 알아보는 것을 시도해보자.

```
lmod <- glm(Species ~ Sepal.Width + Sepal.Length, family=binomial, irisr)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Warning message 가 뜨면서 문제가 발생했음을 알 수 있다. Convergence 문제가 발생했다.

Sumary 를 통해 무엇이 문제인지 더 정확히 살펴보자.

```
sumary(lmod)
```

```
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -360.61  195972.86 -0.0018   0.9985
## Sepal.Width    -110.13   55361.50 -0.0020   0.9984
## Sepal.Length    131.79   64576.99  0.0020   0.9984
##
## n = 100 p = 3
## Deviance = 0.00000 Null Deviance = 138.62944 (Difference = 138.62944)
```

Residual Deviance 가 0 으로 완벽한 fit 을 보이지만, predictor 들의 standard error 가 지나치게 높아서 어떠한 것도 유의하다고 뜨지 않는다는 것을 알 수 있다.

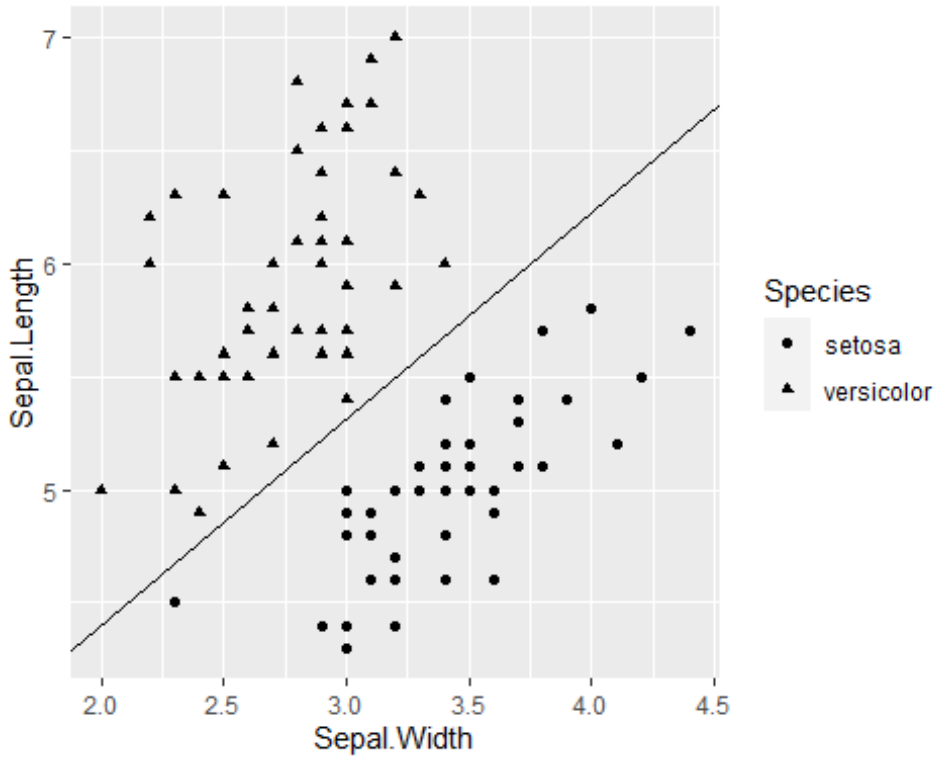
위와 같은 문제가 발생하는 이유는 그래프를 그려봤을 때 두 종이 linear separable 하기 때문이다. 따라서 perfect fit 이 가능하다. 이러한 경우에는 exact logistic regression 또는 bias reduction method 가 고려되어야 한다. 그 중 bias reduction method 는 brglm package 를 이용해서 구현할 수 있다.

```
library(brglm)
bmod <- brglm(Species ~ Sepal.Width + Sepal.Length, family = binomial, irisr)
summary(bmod)
```

```
##
## Call:
## brglm(formula = Species ~ Sepal.Width + Sepal.Length, family = binomial,
##     data = irisr)
##
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -24.508     12.493  -1.962  0.04979 *
## Sepal.Width    -8.897      2.748  -3.237  0.00121 **
## Sepal.Length    9.732      3.334   2.919  0.00351 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 130.638  on 99  degrees of freedom
## Residual deviance:   3.323  on 97  degrees of freedom
## Penalized deviance: 6.60971
## AIC:  9.323
```

결과를 보면 이번에는 significant predictor 가 나왔다는 것을 알 수 있다. 또한 standard error 역시 확 줄었다. 이를 이용해 1/2 predicted probability line 을 그려보면 다음과 같이 나타난다.

```
p + geom_abline(intercept=(0.5+24.51)/9.73, slope=8.9/9.73)
```



➔ Line 이 정확하게 setosa 와 versicolor 두 종을 분류하고 있음을 알 수 있다.