Chapter 7

Matrix OLS and Non-Parametric Bootstrapping (NPBS)

최소자승법(Ordinary least squared)을 이용한 선형회귀분석, 이른바 OLS에서 BLUE란 무엇을 의미하는가? 그리고 표집분포(sampling distribution)란 무엇인가? 모델에 부적절한 변수를 포함시켰을 때, 또는 적절한 변수를 누락시켰을 때 나타날 수 있는 문제들은 무엇이 있는가? 이제는 이와 같은 질문들에 대해 어느 정도 답을 하실 수 있으리라 생각합니다. 또, 아래의 두 선형회귀모델에 대해서 해석할 수 있게 되셨으리라고 생각합니다.

- $y = \alpha + \beta_1 x + \beta_2 z + \beta_3 xz$
- $y - \alpha + \beta_1 I(x = 1) + \beta_2 I(x = 2)$, 이고 이때 $x \in \{0, 1, 2\}$.

마지막으로 가설검정(hypothesis testing)의 기본 논리에 대해서도 조금은 익숙해졌을 것입니다. 어째서 우리는 연구가설(혹은 대안가설)이 맞다라고 직접적으로 검증하지 못하고 경험적 근거를 통해 영가설을 기각하는 "약간은 헷갈릴법한" 방법을 사용하는지 말입니다.

이 챕터에서는 OLS를 다시 한 번 다룰 건데요, 이전과는 다르게 행렬(matrix)을 통해 접근해보고자 합니다. 고등학교 때였나, 행렬이 교과과정에 포함되어 있어 의무적으로 다루었던 것 같기는 한데 혹시 모르니 처음부터 차근차근 살펴보도록 하겠습니다.

7.1 행렬? 벡터? 전치행렬? 역행렬?

저도 OLS 기본을 마무리하고 좀 더 깊게 들어갔을 때, 갑자기 [오랜만에] 등장한 이 행렬 때문에 당황했습니다. 하지만 R로 행렬들이 계산되면서 변화하는 것을 추적하며 공부하면 예전에 수학의 정석 붙잡고 노트에 필기하며 낑낑대었던 것보다는 훨씬 편하게 이해하실 수 있을 겁니다.

대체 행렬(matrix), 벡터(vector), 전치행렬(transposed matrix), 역행렬(inversed matrix)라는 이놈들이 왜 OLS를 공부하는 데 등장하는 것일까요? 그리고 OLS에서 $X\beta$가 제대로 돌아가기 위해서 가정되어야 하는 것들은 무엇이 있을까요? 마지막으로 이것들을 R로 어떻게 보여줄 수 있을까요? 참고로 $X\beta$는 OLS를 통해 얻은 일련의 계수값을 β라는 하나의 집단으로 나타내고 X도 마찬가지로 모델에 포함된 설명변수 일체를 표시한 것입니다.

행렬과 벡터를 정의하기에 앞서, 우리는 스칼라(scalar)라는 것에 대해서 알아야만 합니다. 저도 정치학을 전공한 사람이고 방법론은 연구문제를 풀기 위한 도구로 공부하는 사람이기 때문에 엄청 깊은 증명이나 디테일에 집착하지는 않겠습니다. 저도 지금 큰 문제없이 연구방법을 통해 연구문제들을 풀어가고 있으니, 제가 이해한 정도와 비슷한 정도로만 이해하셔도 연구에는 큰 지장은 없으실겁니다. 물론 주정공이나 부전공이 방법론이라던가 하는 경우에는 조금 얘기가 달라지겠죠? 어디까지나 부외자에 한정된 이야기입니다.

스칼라는 하나의 차원에서 측정된 수치화된 결과를 말합니다. 따라서 스칼라는 방향에 대한 정보는 가지지 않고 크기(magnitude)에 대한 정보만 가지고 있다고 할 수 있습니다. 벡터는 이런 스칼라들의 집합입니다. 스칼라와는 다르게 벡터는 크기뿐아니라 방향에 대한 정보도 가지고 있습니다. 예를 들면, n개의 스칼라를 가진 a라는 벡터가

있다고 해보겠습니다. 우리는 이 벡터를 n차원의 벡터라고 부를 수 있습니다. 이런 벡터는 아래의 수식 7.1과 같이 열 또는 행으로 나타낼 수 있습니다.

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} \text{ or } a = [a_1, a_2, a_3, \cdots, a_n] \tag{7.1}$$

그렇다면 행렬은 무엇일까요? 행렬은 행과 열로 요소들(숫자일 수도 있고 일종의 함수들일 수도 있습니다)이 사각형의 형태를 띄게끔 배열된 결과물입니다. 행렬이라는 이름이 행(raw)과 열(column)이 합쳐진 것이니까 이 정도는 쉽게 유추할 수 있습니다. 행렬을 구성하는 행과 열의 수를 가지고 우리는 행렬의 차원(dimension)이 어떻게 되는지를 말할 수 있습니다. 예를 들면, 아래의 수식 7.2는 요소들의 배열, 즉 행렬의 모습을 보여주고 있습니다. j는 행을, k는 열을 의미하는데요, a_{jk}는 j-행과 k-열에 위치한 요소를 나타냅니다.

$$A = [a_{jk}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{21} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{7.2}$$

행렬의 정의에 따르면, 우리는 벡터 역시도 일종의 행렬이라고 이해할 수 있습니다. 단 하나의 행($1 \times n$) 또는 열($m \times 1$)로 구성된 행렬인 거죠. 그리고 행렬을 전치(transposition)한다는 것은 말 그대로 위치(position)를 뒤바꾸는 것(trans)이기 때문에 행렬의 행과 열을 서로 뒤바꾼다는 것을 의미합니다(수식 7.3). 예를 들면 2개의 행과 3개의 열을 가지고 있던 2×3 행렬이 3개의 행과 2개의 열을 가진 3×2의 행렬로 전치되는 것입니다.

$$A = [a_{jk}] \Rightarrow A^T = [a_{kj}] = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix} \tag{7.3}$$

주어진 행렬의 곱셈에 대한 역원을 역행렬이라고 합니다.모든 행렬이 반드시 그 역행렬을 가지는 것은 아닙니다. A라는 행렬이 역행렬을 가진다고 한다면, 우리는 A라는 행렬이 "역행렬을 가질 수 있는 행렬"이라는 의미에서 가역행렬(invertible matrix)라고 부릅니다. 반면에 역행렬을 가지지 않는 행렬을 특이행렬(singular matrix)라고 합니다. 만약에 어떤 행렬이 역행렬을 가진다면, 이때 그 역행렬은 가역행렬에 대해 유일한 역행렬입니다. 1:1 관계가 성립한다는 거죠. A라는 행렬에 대한 역행렬을 우리는 $\mathtt{A}^{-1}$라고 표기합니다.

우리는 행렬을 이용해서 단선순형회귀모델을 표현할 수 있습니다. 이때 필요한 것은 x와 y로, 실제 관측치 x_i를 요소로 하는 예측변수의 행렬이 모델의 계수값이라고 할 수 있는 행렬을 거치게 되면 $\hat{y}_i$의 행렬을 산출하는 논리입니다. 마찬가지로 다중선형회귀모델도 행렬로 표현할 수 있습니다. 그러나 이때에는 y_i의 변화를 설명하기 위해 두 개 이상의 예측변수들을 나타내는 행렬이 필요합니다. 즉, 최소 세 차원 이상의 행렬이 필요하다는 것입니다.

앞서 정리한 것처럼 벡터는 스칼라들의 집합입니다. 따라서 우리는 k개의 예측변수들과 n개의 관측치를 이용한 행렬들로 수식 7.4와 같이 다중선형회귀모델을 나타낼 수 있습니다.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} X = \begin{bmatrix} 1 & x_{21} & x_{31} & \cdots & x_{k1} \\ 1 & x_{22} & x_{32} & \cdots & x_{k2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{2n} & x_{3n} & \cdots & x_{kn} \end{bmatrix} \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \tag{7.4}$$

수식 7.4에 입각해 우리는 선형모델을 각각의 관측된 y값과 X, 그리고 추정된 β와 u를 가지로 수식 7.5로 보여줄 수 있습니다.

$$\begin{aligned} y_1 &= \beta_1 + \beta_2 x_{21} + \cdots + \beta_k x_{k1} + u_1 \\ y_2 &= \beta_1 + \beta_2 x_{22} + \cdots + \beta_k x_{k2} + u_2 \\ &\vdots \\ y_n &= \beta_1 + \beta_2 x_{2n} + \cdots + \beta_k x_{kn} + u_n \end{aligned} \tag{7.5}$$

여기서 X와 y는 실제 관측된 데이터로부터 나온 예측변수와 종속변수의 개별 관측값을 의미합니다. 따라서 현실의 데이터는 항상 우리가 알지 못하는 요인들에 의해 영향을 받을 수 있기 때문에 우리는 이 '관측하지 못한' 그리고 '비체계적인' 요인들을 나타내는 오차항을 모델에 포함시킵니다. 그러므로 다중선형회귀모델을 보여주는 행렬은 아래의 수식 7.6와 같이 정리할 수 있습니다.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_{21} & x_{31} & \cdots & x_{k1} \\ 1 & x_{22} & x_{32} & \cdots & x_{k2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{2n} & x_{3n} & \cdots & x_{kn} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \tag{7.6}$$

그리고 예측변수 X의 1은 절편값을 나타냅니다. 행렬 계산 공식을 떠올리지 못하더라도 이 수식 7.6을 직관적으로 이해해보자면, 관측된 종속변수 y_i는 관측된 k개의 예측변수의 관측치들인 x_{ki}가 모델에 의해 추정된 β들과 결합하여 계산되는 예측값 $\hat{y}_i$에 현실에서 관측하지 못한 오차들이 더해진 결과라는 것입니다. 우리는 위의 행렬을 계산해서 OLS, 즉 오차의 최소제곱합(least sum of squared errors)을 구할 수 있습니다. 이건 그냥 참고로 알아두세요.

$$\begin{aligned} S &= \Sigma_{i=1}^{n} \epsilon_i^2 = \Sigma_{i=1}^{n} (Y_i - \beta_1 - \beta_2 X_i 1 - \cdots - \beta_k X_{ik})^2 \\ &= \epsilon'\epsilon = (Y - X\beta)'(Y - X\beta) = Y'Y - Y'X\beta - \beta' X'Y + \beta' X'X\beta \end{aligned}$$

이렇게 계산된 S에 편미분을 취하면?

$$\begin{aligned} \frac{\partial S}{\partial \beta} &= -2X'Y + 2X'Y\beta = 0 \\ &\rightarrow X'X\hat{\beta} = X'Y \\ &\rightarrow \hat{\beta} = (X'X)^{-1}X'Y \end{aligned}$$

이렇게 됩니다. 이 행렬계산을 통해서 $\hat{\beta}_k$를 얻을 수 있고(이게 계수값이죠!), R은 이 행렬계산을 구현할 수 있습니다.

우리가 수많은 변수들과 관측치들을 가지고 있다고 해봅시다. $i \leq n$, 즉 n개의 관측치를 가지고 있다고 할 때, 우리는

$$y_i = \alpha + \beta_1 x_{i1} + \cdots + \beta_K x_{iK} + \epsilon_i$$

라는 다중선형회귀모델을 구축할 수 있게 됩니다. 이는 예측변수들의 수에 절편까지를 고려한 $K+1$개의 제한된 변수를 가진 함수를 가지 모델이라는 것을 의미하며, 우리는 이 모델에서 오차의 제곱합이 최소가 되는 계수값들을 구할 수가 있게 되는 것입니다. 조금 더 직관적이게 행렬을 보여드리자면,

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, x_k = \begin{bmatrix} x_{1k} & \\ x_{2k} & \\ \vdots & x_{Nk} \end{bmatrix}, \text{ 그리고 }, \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

이라고 할 때, 각각의 y, x_k, ϵ은 $N \times 1$의 벡터입니다. 관측된 종속변수, 예측변수, 그리고 오차항이죠. 이를 선형회귀모델에 집어넣으면 아래와 같이 다시 써볼 수 있습니다.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \alpha + \beta_1 \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \quad x_{N1} \end{bmatrix} + \cdots + \beta_k \begin{bmatrix} x_{1K} \\ x_{2K} \\ \vdots \quad x_{NK} \end{bmatrix} + v$$

이렇게 보면 조금 더 명확하죠? 우리가 지겹도록 보았던 $y = \alpha + \beta_1 x_1 + \cdots + \beta_k x_k + \epsilon$과 형태가 같습니다. 그런데 이렇게 길게 쓰는 건 너무 비효율적이니 계수값들의 벡터를 $\beta = (\alpha \;\; \beta_k \;\; \cdots \;\; \beta_k)'$라고 나타내고 예측변수들을 $\mathbf{X} = (1 \;\; x_1 \;\; \cdots \;\; x_k)$라고 표현합니다. 이때 $\mathbf{X}$는 $N \times (K+1)$의 행렬이며, β는 $1 \times (K+1)$의 벡터입니다. 이해하기는 어렵지 않겠죠? 왜냐면 예측변수들이야 변수들의 개수(K)에 절편을 표현할 항이 하나 더 추가되서 $(K+1)$에 실제 관측된 관측치의 수(N)을 고려해준 것이고, 계수값은 변수들에 절편을 포함한 개수만큼 계산되는데 위에서 살펴본 것처럼 열로 표현되니까 벡터라고 할 수 있는 것입니다. 그렇다면 우리는 행렬로 나타내는 OLS를 다음과 같이 심플하게 보여줄 수 있습니다.

$$\mathbf{y} = \mathbf{X}\beta + \epsilon$$

이 하나의 식이 모든 N개의 관측치와 K개의 예측변수들에 대한 정보를 함축하고 있는 것입니다.

7.1.1 행렬 OLS에서의 가우스-마르코프

당연히 행렬 OLS도 우리가 이전 챕터들에서 보았던 OLS와 동일하니만큼 OLS의 기본가정, 가우스-마르코프 가정을 $\mathbf{y} = \mathbf{X}\beta + \epsilon$을 이용해 보여줄 수 있습니다. 먼저 우리의 추정치로부터 잔차의 벡터를 얻어냅니다. 그냥 식의 좌변과 우변을 요리조리 옮겨주면 됩니다.

$$\hat{\epsilon} = \mathbf{y} - \mathbf{X}\hat{\beta}$$

개별 종속변수의 관측치에서 우리가 추정한 계수—모델에 일련의 예측변수들의 관측치들을 하나하나 대입하여 계산된 예측값($\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}$)을 제해주면 나오는 것이 우리가 모델로 설명하지 못한 종속변수의 변동(variations), 오차(errors)가 될 것입니다. 물론 표본을 가지고 하는 것이니 잔차(residuals)라고 하는 게 정확한 표현입니다. 그리고 우리는 이렇게 얻은 잔차의 제곱합을 최소한으로 하고 싶은거죠. 이게 OLS의 핵심이니까요. 최소로 만드는 값을 찾는 것은 여러 가지 방법이 있겠습니다만, 여기서는 편미분을 통해 제곱합이 0이 되는 해를 구하는 방식을 취하겠습니다.

$$\frac{\partial \hat{\epsilon}'\hat{\epsilon}}{\partial \hat{\beta}} = \frac{\partial (\mathbf{y} - \mathbf{X}\hat{\beta})'(\mathbf{y} - \mathbf{X}\hat{\beta})}{\partial \hat{\beta}} = 2\mathbf{X}'\mathbf{X}\hat{\beta} - 2\mathbf{X}'\mathbf{y} = 0$$

여기서 $\hat{\epsilon}'\hat{\epsilon}$은 제곱합, $\sum_{i=1}^{N} \epsilon_i^2$를 벡터로 보여준 것입니다. 아무튼 $2\mathbf{X}'\mathbf{X}\hat{\beta} - 2\mathbf{X}'\mathbf{y} = 0$라고 할 때, 우리는 이 식을 다시 $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$로 나타낼 수 있습니다. 이것이 회귀계수(물론 절편까지!)를 구하는 보다 간명한 해(solution)가 되겠습니다. 그리고 이때 회귀계수의 분산은 $Var(\hat{\beta}) = \sigma^2(X'X)^{-1}$가 됩니다. $Var(\hat{\beta})$는 $(K+1) \times (K+1)$의 차원을 갖는 행렬이 되고, 이를 우리는 분산-공분산행렬(variance-covariance matrix)이라고 합니다. 이 분산-공분산행렬은 무엇이고, 왜 중요한지를 이제부터 살펴보겠습니다.

7.1.2 표집분포 (Sampling distribution)

지겹게 들어왔던 표집분포로 다시 돌아왔습니다. 그런데 이번에는 조금 다른 방식으로 표집분포를 나타내보겠습니다. 행렬에 대해서 살펴보았으니 표집분포를 행렬의 형태로 나타내 보겠습니다.

$$\hat{\beta} \sim N(\beta, \hat{\sigma}^2(\mathbf{X}'\mathbf{X})^{-1})$$

수없이 많은 표본을 뽑아서 그 표본들에 동일한 선형회귀모델을 적용해 계산한 계수값 $\hat{\beta}_k$들은 표집에 문제가 없다면 모집단의 모수, β를 평균으로 하고 설명변수들 간에 나타날 수 있는 분산-공분산, 일종의 표집과정의 본연적 한계로 나타나는 오차에 따라 분산($\hat{\sigma}^2(\mathbf{X}'\mathbf{X})^{-1}$)을 가지게 될 것이라는 이야기입니다. 이때, 우리는 모집단의 분산을 알 수 없으므로 σ^2 대신에 표본의 분산, $\hat{\sigma}^2$를 사용합니다.

7.1.3 행렬 OLS의 유용성

행렬 OLS는 어떠한 장점을 가지고 있길래 사용하는 걸까요? 무엇보다도 예측값에 있어서 분석적 접근(analytical approach)보다 유용하다고 할 수 있습니다. 분석적 접근이란 쉽게 말해서 통계 이론적으로 OLS에 접근하는 방식을 의미하는데요, 모집단과 표본의 관계를 바탕으로 오차와 잔차의 관계, 계수값과 영가설을 바탕으로 t와 p의 값을 계산하고, 표준편차와 관측치의 수를 바탕으로 표준오차를 계산하는 등의 접근법을 말합니다. 이제까지 해오던 방식입니다.

반면에 행렬 OLS는 우리가 직접적으로 관측한 데이터를 바탕으로 모델을 통해 일종의 예측값들을 얻을 수 있도록 해줍니다. 정확하게는 우리가 관심을 가지고 있는 행렬 $\bar{\mathbf{X}}$를 가지고 모델을 거쳐 나올 예측값을 계산할 수 있게 됩니다. 다시 말하면, $\bar{\mathbf{X}}$는 우리가 임의로 만들 수 있습니다. 이 $\bar{\mathbf{X}}$는 관측된 실제 변수 X와 마찬가지로 모든 K 개의 변수들에 대한 가정을 포함하고 있습니다. 즉, 원래 변수는 1, 4, 11, 5, 6 이런 관측치의 배열을 가지고 있다고 할 때, 우리는 이 변수가 1부터 11까지 1씩 증가할 때 나타날 $\mathbf{y}$, 예측값 $\hat{\mathbf{y}}$를 얻을 수 있습니다. $\hat{\mathbf{y}} = \bar{\mathbf{X}}\hat{\beta}$라는 과정을 통해서 우리는 변수들의 관계를 쉽게 이해하고 실질적인 의미를 파악할 수 있습니다. 그리고 무엇보다 이 $\hat{\mathbf{y}} = \bar{\mathbf{X}}\hat{\beta}$는 R을 통해 구현될 수 있습니다.

7.1.4 행렬 OLS, R로 보기

행렬 OLS가 과연 분석적 접근법으로 계산해낸 OLS 추정 결과와 같은 결과를 가질지를 알아보기 위해서 실제의 데이터를 이용한 R 분석을 수행해보겠습니다. 언제나와 같이 QoG 데이터셋에서 2015년 기준의 POLITY2, 무역개방성, 2010년 기준 1인당 GDP를 따로 서브셋을 만들어 사용할 것입니다. 먼저 모델이 어떻게 생겼는지를 보겠습니다.

```
# 먼저 model1이 우리가 추정할 모델입니다.
model1 <- lm(log2(wdi_gdpcapcon2010) ~ ## 종속변수는 로그를 취한 1인당 GDP
               1 + p_polity2 + wdi_trade, data=QOG2) ## 1은 절편을 의미
```

위의 모델은 로그값을 취한 1인당 GDP를 종속변수로 하고, POLITY2 즉, 민주주의와 무역개방성을 예측변수로 합니다. 민주주의 수준과 무역개방성을 가지고 1인당 GDP를 설명하고자 하는 모델인 것이죠. 그럼 행렬 OLS로 이 모델을 재구성해보겠습니다.

```
X <- model.matrix(object = log2(wdi_gdpcapcon2010) ~ 1 +
                    p_polity2 + wdi_trade, data=QOG2)
head(X)
```

```
##   (Intercept) p_polity2 wdi_trade
## 1           1        -1  55.12553
## 2           1         9  71.80101
## 3           1         2  59.69513
## 4           1        -2  62.88852
## 5           1        -7  72.60137
## 6           1         9  22.48623
```

먼저 우리가 필요로 하는 예측변수들의 행렬, $\mathbf{X}$를 만들어줍니다. 로그화된 1인당 GDP의 개별 관측치에 대응하는 절편과 실제 관측된 민주주의, 무역개방성의 관측치가 행렬의 형태로 저장됩니다.

```
Y <- log2(QOG2$wdi_gdpcapcon2010)
head(Y)
```

```
## [1]  9.16537 12.14360 12.22182 11.87203 12.56599 13.36744
```

마찬가지로 로그값을 취한 1인당 GDP도 **Y**라는 행렬에 저장해줍니다. 벡터도 행렬이니까요. 이제 저 둘을 계산해보겠습니다.

```
beta.Hat <- solve(t(X) %*% X) %*% t(X) %*% Y
```

이게 우리가 수식으로 썼던 $\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$에 대응하는 R 코드입니다. 순서대로 한 번 보겠습니다. 결과가 어떻게 변화하는지 보세요.

```
step1 <- solve(t(X) %*% X)
step1
```

```
##               (Intercept)     p_polity2     wdi_trade
## (Intercept)  0.0277598402 -7.062349e-04 -2.138999e-04
## p_polity2   -0.0007062349  1.794912e-04 -9.410982e-07
## wdi_trade   -0.0002138999 -9.410982e-07  2.572063e-06
```

```
step2 <- solve(t(X) %*% X) %*% t(X)
step2 %>% as.data.frame() %>% dplyr::select(1:4) %>% head()
```

```
##                          1             2             3             4
## (Intercept)  1.667473e-02  6.045499e-03  1.357859e-02  1.572046e-02
## p_polity2   -9.376046e-04  8.416139e-04 -4.034315e-04 -1.124402e-03
## wdi_trade   -7.117243e-05 -3.769303e-05 -6.224242e-05 -5.026442e-05
```

```
# 이런게 153개 열이 있습니다
step3 <- solve(t(X) %*% X) %*% t(X) %*% Y
step3
```

```
##                    [,1]
## (Intercept) 10.68907645
## p_polity2    0.08195834
## wdi_trade    0.01479342
```

자, 위의 저 결과가 바로 행렬 OLS로 구한 $\hat{\beta}$가 됩니다. 그럼 이번에는 표준오차를 구해볼까요? 이미 위에서 수식으로 다 보여드린 내용을 R로 다시 풀어놓은 것과 다름이 없습니다. 여기서 처음보는 건 아마 chol 함수일텐데요, 이는 촐레스키 분해(Cholesky Factorization) 함수입니다.[1] 저는 여기서 자세한 설명은 하지 않을 것이기 때문에 참고할만한 블로그 링크를 공유하겠습니다.

```
sig.sq <- sum((Y - X%*%beta.Hat)^2)/(nrow(X)-ncol(X)) ## 분산이죠?
VCV <- sig.sq*chol2inv(chol(t(X)%*%X))
SE <- sqrt(diag(VCV))
# 행렬 OLS 결과
cbind(beta.Hat, SE) %>% knitr::kable()
```

		SE
(Intercept)	10.6890764	0.3279945
p_polity2	0.0819583	0.0263742
wdi_trade	0.0147934	0.0031572

```
# R lm()을 이용한 OLS 결과
coef(summary(model1)) %>% knitr::kable()
```

[1] https://issactoast.com/129

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	10.6890764	0.3279945	32.589190	0.0000000
p_polity2	0.0819583	0.0263742	3.107517	0.0022573
wdi_trade	0.0147934	0.0031572	4.685646	0.0000062

두 결과의 계수값과 표준오차 추정치가 일치하는 것을 확인하실 수 있습니다.

7.2 비모수 부트스트랩(Nonparametric Bootstrap: NPBS)

통계학에서 표본은 잊을만하면 다시금 튀어나와 생각하게 만듭니다. 사실 영어로 통계학을 의미하는 `Statistics`에서 `Statistic` 자체가 모집단의 모수(parameters)와 대비되는 표본에서의 수치를 의미한다고 할 수 있으니까요. 여기서도 익숙지 않은 비모수 부트스트랩이라는 것을 다루기 전에, 표본에 대한 이야기를 조금 하고 넘어갈까 합니다.

우리는 모집단을 관측할 수 없기때문에 모집단의 특성을 파악하기 위해서 모집단의 제반을 고루 보여줄 수 있는 대표성이 있는 표본을 얻기를 원하고, 이때 어떠한 체계적인 편향(bias)가 개입되길 원치 않습니다. 대표성이 있는 표본을 뽑을 수 있는 표집방법 중 하나는 바로 무작위 추출입니다. 따라서 우리는 모집단으로부터 무작위로 추출된 표본을 기대합니다.

한편, 모집단에서 표본을 추출할 때, 뽑고 난 후 그 값이 다음 차례에서도 또 뽑힐 수 있는, 이른바 무작위 복원추출을 하게 된다면 우리는 모집단으로부터 또 다른 표본을 무수히 얻을 수 있을 것입니다. 즉, 여러 색의 공이 뒤섞여 담긴 항아리에서 10개의 공을 꺼낸 것이 우리의 표본이라고 할 때, 꺼낸 공을 다시 원래의 항아리에 집어넣는다면 또 10개의 공을 꺼낼 수 있고, 그렇게 우리는 무수히 많은 10개의 공들의 기록을 얻을 수 있을 것입니다. 비모수 부트스트랩은 이 지점에서 한 가지 질문으로부터 시작됩니다. 만약 우리가 모집단에서 표본을 무작위 복원추출하듯이 꺼낼 수 있다면, 혹시 표본으로부터 또 다른 표본을 무작위 복원추출한다면 어떻게 될까?

7.2.1 기본 아이디어

우리가 알고싶은 것은 모집단의 특성입니다. 그러나 모집단을 관측하거나 혹은 모집단에 관한 완전한 데이터를 얻을 수 없기 때문에 우리는 모집단의 특성을 잘 반영할 것이라고 기대된 표본을 통해 모집단이 어떠할 것이라는 통계적인 추론을 하게 됩니다. 즉, 추론의 기저에는 표본이 모집단을 잘 대표하고 있다면 표본에서 우리가 관측한 분포, 특성들이 모집단에서도 그러할 것이라고 주장하는 것이 가능하다는 가정이 깔려 있습니다.

이론적으로 우리는 모집단에서 무수히 많은 표본을 추출할 수 있고, 우리는 이렇게 반복된 표본들로 연구를 반복할 수 있습니다. 그러나 모집단으로부터 표본을 추출하는 표집방법의 근본적인 문제로 동일한 모집단에서 뽑은 표본들이라고 할지라도 표본들 간에는 약간의 차이가 존재할 수 있습니다. 선형회귀모델을 한 번 생각해보겠습니다. 우리는 기본적으로 함수적 관계를 통해 어떠한 예측변수들이 종속변수와 관계를 맺을 것이라고 주장합니다. 그러나 약간씩 다른 표본의 자료들을 모델에 투입하는 순간, 당연히 결과로 추정되는 계수값들도 차이가 나게 될 것입니다. 우리는 이렇게 서로 다른 표본들로부터 얻은 통계치, 계수값들을 한 데 모아 그 분포를 그려볼 수 있습니다. 이것이 바로 표집분포입니다.

가우스-마르코프 가정에 따르면, 우리는 이렇게 한 데 그러모은 표본들로부터 나온 계수들의 기대값이 모집단 수준의 모수, 단 하나의 진실된 값과 동일할 것이라고 기대합니다. 그러나 현실적으로 모집단으로부터 수없이 많은 표본들을 얻는 것은 불가능합니다. 예를 들어, 정치학에서 흔히 사용되는 데이터셋인 `POLITY IV`를 생각해보겠습니다. `POLITY IV`는 어디까지나 측정가능한 민주주의 정치체제에 대한 제반 속성들을 수치화한 것입니다. 민주주의라고 하는 모집단을 의미하는 것은 아니죠. 어디까지나 표본입니다. 하지만 이 데이터셋 하나를 구축하는 데만도 엄청난 자원과 비용이 소요되었습니다. 네, 현실에서는 이렇게 표본 하나를 제대로 구하는 것도 쉽지 않습니다. 그러므로 사실 우리는 단 하나의 표본에서 나타나는 특성과 통계치를 이용해서 표집분포를 추정하고 있습니다. 이렇게 하기 위해서는 몇 가지 가정들이 충족되어야만 합니다. 예를 들어, 우리가 선형회귀모델을 단 하나의 표본만을 가지고 추정한다고 할 때, 오차항의 독립성과 정규분포가 가정되어야 합니다. 만약 그렇지 않다면 우리는 추정한 계수값이 일관되거나 편향되지 않았다고 하기 어려울 것입니다.

부트스트랩 방법은 표집분포를 추정하는 데 있어서 다른 접근법을 취합니다. 부트스트랩 방법은 대개 복원추출을 통해 모집단으로부터 얻은 n개의 관측치를 가진 표본으로부터 다시 n개의 관측치를 가진 표본을 뽑아냅니다.

즉, 표본에서 표본을 뽑습니다. 이렇게 재추출된 표본들은 원래 모집단에서 추출한 표본과 동일한 값을 지닐수도, 아닐수도 있습니다. 왜냐하면 무작위 복원추출을 했기 때문에 어떤 값은 중복되고 어떤 값은 재추출된 표본에는 포함되지 않을수도 있으니까요. 부트스트랩은 이 과정을 반복합니다. 그러고나면 우리는 모집단에서 얻은 단 하나의 표본으로부터 무수히 많은 재추출 표본들을 얻을 수 있게 됩니다.

왜 이런 방법을 사용할까요? 목적은 다르지 않습니다. 우리는 여전히 우리가 알 수 없고, 관측할 수 없는 모집단의 특성을 알고 싶습니다. 그 모집단의 특성을 알기 위해서는 그 특성의 확률분포를 알아야할텐데, 우리는 그 분포가 어떠할 것이라는 확신을 가질 수가 없습니다. 즉, 우리는 사전에 모집단의 확률분포가 어떠할 것이라고 알지 못합니다. 우리는 오직 관측된 표본 통계치들만을 가지고 모수를 추정할 뿐입니다. 한정된 데이터로 인한 불확실성은 우리의 추론을 제약합니다. 또한, 대부분의 통계방법들은 표본의 크기에 따라서 다른 결과들을 내어놓기도 합니다. 표본의 크기 자체가 커질수록 우리는 모수에 대한 더 안정적인 결과를 얻을 수 있을 것이라 기대합니다. 하지만 비용과 시간의 제약으로 인하여 표본 규모를 늘리기란 쉽지 않습니다. 작은 규모의 표본을 가지고 할 수 있는 대안이 없을까? 여기서 부트스트랩 방법이 시작된 것입니다. 부트스트랩 방법은 기본적으로 하나의 표본으로부터 수많은 부트스트랩 표본들을 추출함으로서 앞서의 문제들을 극복하고자 하며, 나아가 하나의 표본 특성을 부트스트랩 표본들로부터 얻은 표집분포로 추론하고, 나아가 그것을 모집단의 모수를 추론하는데까지 사용하고자 합니다.

부트스트랩 방법을 사용하게 되면 우리는 무수히 많은 부스트스랩 표본들을 하나의 표본으로부터 얻게 됩니다. 그리고 부트스트랩 표본들은 하나의 표본으로부터 무작위 복원추출을 한 결과물입니다. 이제 기본적인 논리가 이해가 가실 겁니다. 부트스트랩 표본들 $\rightarrow$ 표본 $\rightarrow$ 모집단으로 가는 경로인 것이죠. 부트스트랩 방법은 원 표본(original sample)과 동일한 규모의 표본으로 재추출하고, 부트스트랩 표본들로부터 얻은 표본 통계치의 분포를 표집분포로 사용합니다. 즉, 부트스트랩 방법은 원 표본을 일종의 모집단에 대한 대리(proxy)로 간주하고 무작위 반복 표집을 하는 것이죠. 이렇게 새로 뽑은 부트스트랩 표본은 흡사 원표본에 대한 '하위 표본'으로 보이지만 정확하게는 '하위 표본'은 아닙니다. 모집단에 대한 또 다른 표본이죠. 이것이 가능하기 위해서는 하나의 가정이 필요합니다. 바로 원 표본이 모집단을 대표할 수 있는 무작위 반복추출된 표본이어야 합니다. 어디까지나 원 표본이 잘 추출되었다면, 거기서 무작위로 반복추출해서 다시 만든 부트스트랩 표본들은 모집단에서 무작위 반복추출한 결과와 다르지 않을 것이라는 굉장히 강력한 무작위화(randomization)에 대한 믿음이 뒷받침하고 있습니다.

결과적으로 부트스트랩의 재표집 과정은 무수히 많은 표본들을 얻을 수 있고, 부트스트랩 표본들로부터 얻은 표본 통계치들은 동일한 모집단에서 추출된 무작위 표본들 간의 차이(variability)를 추정하는 데 사용될 수 있습니다. 이러한 부트스트랩 표집분포를 이용해 우리는 실제 관측치를 이용한 신뢰구간 측정이나 가설검정을 수행할 수 있게 됩니다.

여기서 한 가지 착각하면 안 되는 것은 비모수 부트스트랩 방법이 결코 모집단의 분포에 대한 어떠한 가정을 하고 있는 것은 아니라는 점입니다. 우리는 작은 규모의 표본을 가지고 있을 때, 이 표본이 최소자승법을 사용하기 위한 가정들을 충족시키는지 여부를 알지 못하기 때문에 OLS의 추정 결과를 얻더라도 그것이 과연 BLUE인지를 검정할 방법이 없습니다. 하지만 부트스트랩 방법을 이용하면 이 문제를 시뮬레이션한 표집 분포를 이용해 해결할 수 있습니다. 이것이 소규모 표본을 이용할 때 표본 통계치의 신뢰도[2]가 낮아 생길 수 있는 문제들을 다룰 수 있는 부트스트랩의 장점입니다.

부트스트랩 방법의 장점은 우리가 작은 표본을 가지고 있거나 모집단의 분포를 모른다고 하더라도 모집단의 특성을 추정할 수 있도록 해준다는 것입니다. 시뮬레이션된 표집분포를 가지고 우리는 계수, 표준오차, 그리고 신뢰구간 등 기존에 추정하던 통계치들을 모두 추정할 수 있습니다. 또한 OLS가 회귀선을 그리는 방법을 부트스트랩 방법과 비교하자면, 부트스트랩 방법은 사전에 모집단의 분포가 어떠할 것이라고 가정을 하지 않기 때문에 모집단의 확률분포가 정규분포가 아니라고 가정하는 다른 여러 추정기법들과도 함께 사용될 수 있습니다.

개인적으로는 부트스트랩 방법은 대개 소규모 표본의 문제로 여러가지 어려움을 겪는 사회과학 분야의 경험연구에 있어서 특히나 유용한 대안이라고 생각합니다. 그러나 분명 부트스트랩 방법에도 한계는 있습니다. 바로 무작위화의 가정이 깨어질 경우입니다. 만약 우리가 가진 단 하나의 표본이 모집단으로부터 무작위 복원추출되었다고 할 근거가 충분하지 못하다면 어떻게 될까요? 원 표본이 편향된 순간, 부트스트랩 표본들도 모집단의 특성을 대표하는 것이 아니라 사실상 원 표본의 체계적 편향을 반영한 표본들이 되고 맙니다.

[2] 여기서의 신뢰도란 reliability로 모집단에서 다른 표본을 추출해 동일한 모델을 돌리더라도 우리가 처음에 얻은 결과가 재생산될 것이라는 개념입니다. 화살 과녁에 비유하자면 내가 쏜 10발의 화살이 모두 원하는 장소에 적중하는 것은 정확하고 적실한 개념을 사용했다하여 타당성(validity)이 충족되었다고 하고, 10점은 아니지만 10개의 화살이 쏠 때마다 7점이면 7점, 8점이면 8점에 고루 모여있는 것을 다음 결과 역시 어떻게 나올지 신뢰할만하다 하여 신뢰도(reliability)가 높다고 합니다.

7.2.2 부트스트랩… 어디다 써먹지?

정리하자면 비모수 부트스트랩을 통해 우리는 표집분포를 얻을 수 있습니다. 그리고 이 표집분포에 대한 함수적 관계를 상정하는 것도 자유롭습니다. 분석적 접근법(analytic approach)에 따르면 모델 $y = \beta_0 + \beta_1 x + \beta 2_z + \beta_3 xz + \epsilon$에서 상호작용을 나타내는 편미분 결과 $\hat{\beta}_1 + \hat{\beta}_3 z$의 표준오차, 즉 표본의 차이에 따라 그 상호작용 효과가 다르게 나타날 수 있는 변동성은 $Var(\hat{\beta}_1) + z^2 Var(\hat{\beta}_3) + 2zCov(\hat{\beta}_1, \hat{\beta}_3)$ 으로 나타납니다. 하지만 만약 우리가 $g = 1, \cdots, G$개의 부트스트랩 표본들을 추출할 수 있게 된다면, 그 부트스트랩 표본들로부터 얻은 $\hat{\beta}_1^{(1)} + \hat{\beta}_3^{(1)} z, \cdots, \hat{\beta}_1^{(G)} + \hat{\beta}_3^{(G)} z$를 가지고 요약통계를 보여주면 됩니다. 훨씬 간단하죠.

행렬 OLS를 통해 우리가 얻을 수 있는 예측값이 $\hat{\mathbf{y}} = \bar{\mathbf{X}}\hat{\beta}$이라는 것을 다시 떠올려보겠습니다. 부트스트랩 표본을 $g = 1, \cdots, G$개 가졌다고 한다면 이 부트스트랩 표본들로부터 얻은 계수값은 $K \times 1$의 벡터로 나타낼 수 있습니다: $\hat{\beta}^{(g)}$. 따라서 하나의 부트스트랩 표본에서 얻어진 예측값은 $\hat{\mathbf{y}}^{(g)} = \bar{\mathbf{X}}\hat{\beta}^{(g)}$라고 할 수 있습니다. 그리고 G개의 부트스트랩 표본들을 이용해 알고 싶은 예측값에 대한 전체 부트스트랩 분포를 구할 수 있습니다. 간단히 말하면, x_1 변수에 대한 $\hat{beta}_1$은 부트스트랩 표본들 전체를 모델에 집어넣고 나온 $\hat{\beta}^{(g)}$의 분포로 나타낼 수 있다는 것입니다.

- 부트스트랩 표본들로부터 추정된 일련의 계수값들을 $\hat{\mathbf{b}} = (\hat{\beta}^{(1)} \cdots \hat{\beta}^{(G)})$라고 하겠습니다.
- 그리고 부트스트랩 표본들에 모델을 적용하여 얻은 일련의 예측값들을 $\hat{\mathbf{Y}} = (\hat{y}^{(1)} \cdots \hat{y}^{(G)})$라고 해보겠습니다.
- $\hat{\mathbf{Y}} = \bar{\mathbf{X}}\hat{\mathbf{b}}$는 우리가 관심을 가지는 특정한 M이라는 예측값 각각에 대한 G개의 부트스트랩 표본들이라고 할 수 있습니다.

7.2.3 부트스트랩 R로 보기: 비모수 부트스트랩의 추출횟수와 결과

실제 데이터를 가지고 부트스트랩 방법을 적용해보겠습니다. 논리대로라면 부트스트랩 표본의 추출 횟수가 증가할수록, 부트스트랩 표본을 통해 얻은 계수 추정치들은 우리가 표집분포를 이용해 도출한 분석적 접근법의 결과에 근사하게 될 것입니다.

```
# 비모수 부트스트랩 표본추출 횟수 변화에 따른 결과의 차이를 보겠습니다.
# 먼저 100번 부트스트랩을 할 경우입니다.
holder100 <- matrix(NA, 100) ## 일단 100번 추출한 결과가 들어갈 빈 행렬을 만듭니다.
for(i in 1:100){  ## 함수를 만듭니다. i는 1부터 100까지를 의미합니다.
  ind <- sample(1:nrow(QOG2), ## ind는 전체 관측치(각 행의 번호) 중에서
                ## QOG2와 동일한 표본규모로 표집됩니다.
                  size=nrow(QOG2), replace=T) ## 복원표집된 결과입니다.
  # 이 ind가 헷갈리실 텐데, 뭐냐면 넘버링입니다. 매번 무작위로 넘버링이
  # 나올거고 이 넘버링에 속하는 관측치들은 개별 부트스트랩 표본을 구성합니다.
  # 그 과정이 i번, 즉 1~100까지 반복되어 총 100개의 부트스트랩 표본이 생깁니다.
  mod <- lm(log2(wdi_gdpcapcon2010) ~ ## 모델은 앞서와 동일하게 적용합니다.
              1 +  p_polity2 + wdi_trade,
            data=QOG2[ind,]) ## ind에 속하는 행의 관측치만을 가지고 모델분석
  holder100[i,] <- coef(mod)[2] ## 이렇게 새로 부트스트랩 표본으로 모델을
  # 분석해 나온 계수값만을 아까 만든 깡통행렬에 차곡차곡 하나씩 저장.
  # 민주주의에 대한 계수의 결과만을 보겠습니다.
}
head(holder100) ## 그려면 이렇게 총 100개의 계수값이 담긴 행렬이 완성됩니다.
```

```
##             [,1]
## [1,] 0.09520917
## [2,] 0.10421496
## [3,] 0.10719345
## [4,] 0.06325093
```

```
## [5,] 0.07453695
## [6,] 0.11119179
```

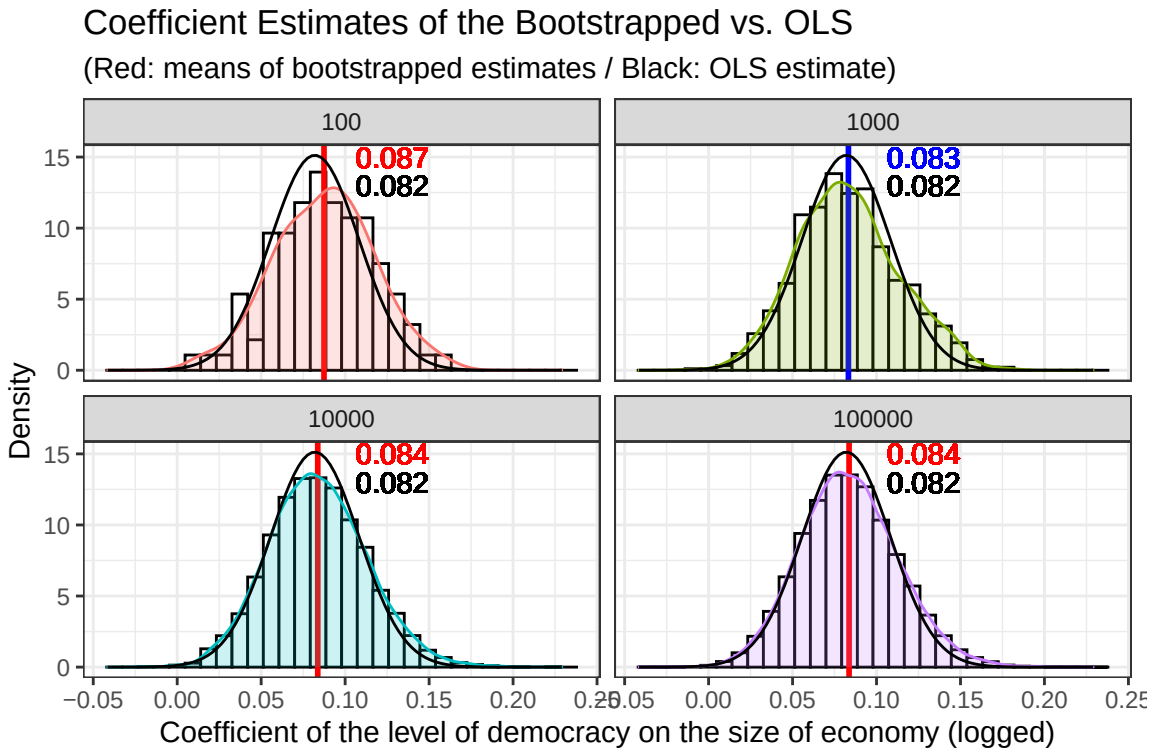
이 다음에는 단순하게 추출 횟수만 바꾸어주면 되겠죠? 빠르게 가겠습니다.

```
## 1000번 Bootstrapping
holder1000 <- matrix(NA, 1000)
for(i in 1:1000){
  ind <- sample(1:nrow(QOG2),
                size=nrow(QOG2), replace=T)
  mod <- lm(log2(wdi_gdpcapcon2010) ~
              1 +  p_polity2 + wdi_trade,
            data=QOG2[ind,])
  holder1000[i,] <- coef(mod)[2]
}

## 10000번 Bootstrapping
holder10000 <- matrix(NA, 10000)
for(i in 1:10000){
  ind <- sample(1:nrow(QOG2),
                size=nrow(QOG2), replace=T)
  mod <- lm(log2(wdi_gdpcapcon2010) ~
              1 +  p_polity2 + wdi_trade,
            data=QOG2[ind,])
  holder10000[i,] <- coef(mod)[2]
}

## 100000번 Bootstrapping
holder100000 <- matrix(NA, 100000)
for(i in 1:100000){
  ind <- sample(1:nrow(QOG2),
                size=nrow(QOG2), replace=T)
  mod <- lm(log2(wdi_gdpcapcon2010) ~
              1 +  p_polity2 + wdi_trade,
            data=QOG2[ind,])
  holder100000[i,] <- coef(mod)[2]
}
```

이렇게 계산된 각각의 100번, 1,000번, 10,000번, 100,000번의 부트스트래핑 결과로 얻어진 계수값들을 하나의 데이터로 만들어서 비교해보도록 하겠습니다. `tidyverse` 패키지의 `tibble`을 이용하여 하나의 데이터로 만들겠습니다.

```
# 각각의 부트스트랩한 계수값 결과를 하나의 데이터로 만들어줍니다.
npbs <- bind_rows(
  holder100 %>%
    as_tibble() %>% mutate(NPBS = "100") %>%
    rename(Estimates = V1),
  holder1000 %>%
    as_tibble() %>% mutate(NPBS = "1000") %>%
    rename(Estimates = V1),
  holder10000 %>%
    as_tibble() %>% mutate(NPBS = "10000") %>%
    rename(Estimates = V1),
  holder100000 %>%
    as_tibble() %>% mutate(NPBS = "100000") %>%
```

```
    rename(Estimates = V1))

# 각 부트스트랩 횟수별로 평균을 구해주겠습니다. 그리고 비교대상인 OLS의
# 민주주의 계수값 결과도 추가해주고요.
npbs <- npbs %>% group_by(NPBS) %>% mutate(
  `NPBS Mean` = mean(Estimates, na.rm = T),
  `OLS estimates` = coef(summary(model1))[2,1],
  `OLS se` = coef(summary(model1))[2,2]
) %>% ungroup()
```



부트스트랩으로 얻은 회귀계수와 OLS로 추정한 회귀계수를 비교해보도록 하겠습니다. 각 패널은 부트스트랩 복원 표본추출 횟수를 보여줍니다. 그리고 x-축은 회귀계수 값을, y-축은 밀도입니다. 좌측 상단의 패널은 부트스트랩 패널을 100번 추출했을 때의 결과입니다. 붉은 선이 부트스트랩으로 얻은 회귀계수의 평균이고 소수점 셋째 자리에서 0.084입니다. 그리고 붉은 색으로 표시된 분포와 히스토그램이 부트스트랩으로 얻은 계수 관측치들의 분포를 보여줍니다. 검정색 선으로 표시된 분포는 OLS 분석으로 얻은 회귀계수값을 평균으로 하고 표준오차를 표준편차로 하는 정규분포입니다. 말이 되죠? 왜냐면 OLS의 가우스-마르코프 가정이 충족된다면 OLS의 회귀계수는 BLUE 일테니, 그렇게 얻은 회귀계수의 기대값인 평균은 모집단의 모수, β와 같을 것이고 표집분포에서의 표준편차는 표본추출로 인한 표본들 간에 나타날 수 있는 차이를 보여주는 표준오차로 나타나니까요.

일단 부트스트랩 표본추출 100회차를 보면 벌써 평균이 0.002밖에 차이가 안 납니다. 다만 분포 양상은 정규분포라고 하기 조금 힘듭니다. 아무래도 관측치가 100개밖에 없다보니 좀 삐뚤빼뚤합니다. 히스토그램만 봐도 그렇죠.

두 번째로는 부트스트랩을 1,000번 시뮬레이션한 결과입니다. 평균 차이는 뭐 여전히 조금 존재하기는 하지만 분포가 한층 더 정규분포에 비슷해진 것을 확인할 수 있습니다. 마찬가지로 나머지 하단의 두 패널들도 점점 분포 양상이 정규분포 모양으로 수렴하는 것을 확인할 수 있습니다. 즉, 이를 통해서 우리는 부트스트랩 표본 추출 횟수가 증가할수록 부트스트랩 표본들을 이용해 얻은 계수값들의 분포는 OLS 회귀계수의 정규분포에 근사한다는 것을 알 수 있습니다. 시뮬레이션이 분석적 접근법의 대안일 수 있다는 것이죠.

7.2.4 부트스트랩 R로 보기(2): 신뢰구간과 점추정치

챕터 4에서 "변수에 대한 심화"라는 항에서 Gelman (2008)[3]에 따라 변수를 표준화하되 $b = 1sd(x)$가 아니라 $b = 2sd(x)$로 표준화를 해준 적이 있습니다. 결과적으로 $b = 2sd(x)$로 표준화했을 때, 표준화 이전 계수의 경향성과 표준화한 이후의 계수의 경향성이 비슷한 양상을 보인다는 것을 확인했습니다. 어차피 표준화의 결과로 실질적 변수의 효과가 다르게 나타나는 것은 아니니만큼 $b = 2sd(x)$ 표준화가 직관적인 이해까지 도울 수 있다는 제언을 한 적이 있습니다. 이번에는 당시의 분석을 비모수 부트스트랩을 이용해서 재현해보도록 하겠습니다.

기억을 되살려보면 $y_i = \delta_0 + \delta_1 x_i,\ i \leq n$이라고 할 때, $a = \frac{1}{n}\sum_i x_i$이고 $b = 2sd(x)$인 $y_i = \lambda_0 + \lambda_1 \frac{x_i - a}{b}$를 추정하는 것이었습니다. x에 맞추어 재구성하면 결국 아래와 같은 모델이라는 것을 알 수 있죠.

$$y = \lambda_0 + \lambda_1 (\frac{x_i - \bar{x}}{2 \times \sigma_x})$$

이번에는 QoG 데이터셋의 1인당 GDP와 민주주의 수준을 이용해 분석해보겠습니다. 데이터를 고려해보면 제 모델은 아래와 같습니다.

$$\text{경제규모}_i = \lambda_0 + \lambda_1 \frac{(\text{민주주의 수준}) - (\text{민주주의 수준의 평균})}{\text{민주주의 수준의 표준편차} \times 2}$$

```
# 2sd 표준화
QOG2 <- QOG2 %>%
  mutate(
    std.polity = p_polity2 - mean(p_polity2)/2*sd(p_polity2))

# 새롭게 모델링
model.sd <- lm(log2(wdi_gdpcapcon2010) ~ std.polity, data=QOG2)

# 결과
model.sd %>% broom::tidy() %>%
  mutate_if(is.numeric, round, 3) %>% knitr::kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	13.079	0.302	43.279	0.000
std.polity	0.087	0.028	3.107	0.002

```
# 비교를 위한 원 변수 모델
model2 <- lm(log2(wdi_gdpcapcon2010) ~ p_polity2, data=QOG2)
model2 %>% broom::tidy() %>%
  mutate_if(is.numeric, round, 3) %>% knitr::kable()
```

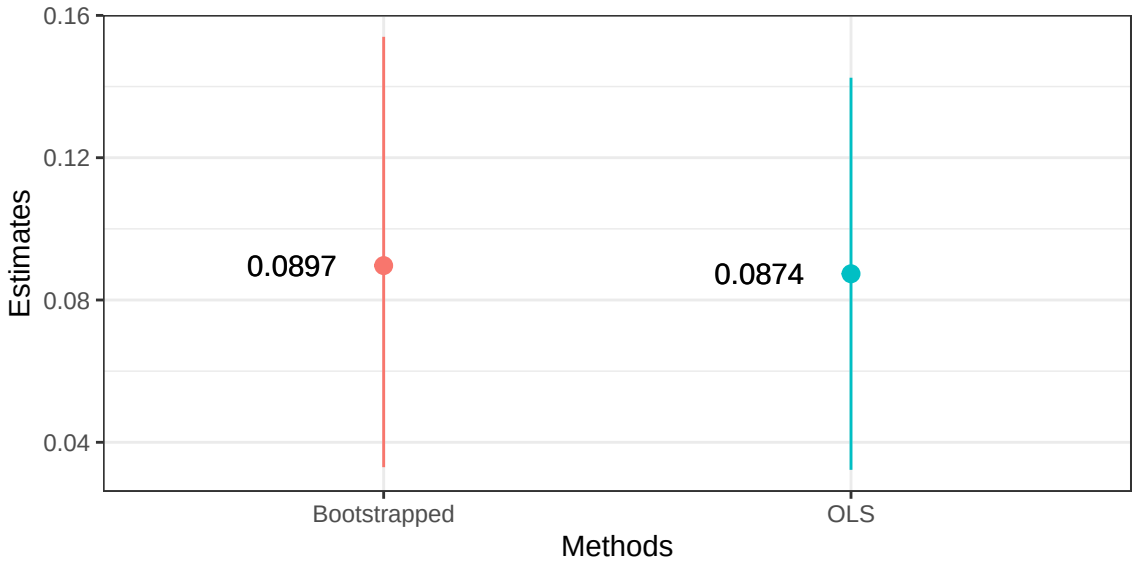
term	estimate	std.error	statistic	p.value
(Intercept)	11.919	0.210	56.820	0.000
p_polity2	0.087	0.028	3.107	0.002

표준화를 했기 때문에 이제 예측변수는 측정단위에 따라 결과가 좌우되지는 않습니다. 즉, 변수가 여러개 있다면 표준화한 변수들의 효과들 간에는 상대적 비교가 가능합니다. 예를 들어, x_1이 x_2보다 y에 미치는 효과가 상대적으로 크다, 작다 등으로 해석이 가능하다는 얘기지요. 그것이 실질적인 효과와 직결되느냐는 조금 다른 문제입니다만 여하튼 표준화는 서로 분석단위가 kg 대 ml 등으로 상이한 변수들 간의 효과를 비교할 수 있도록 스케일링해주는 것입니다.

물론 여기서는 단 하나의 예측변수만을 사용하기 때문에 결과는 표준화된 예측변수—민주주의의 수준과 로그값을 취한 종속변수의 관계만을 보여줄 것입니다. 이 분석을 위하여 4,000번의 부트스트랩 복원표본추출을 해보도록 하겠습니다.

[3] Gelman, Andrew. 2008. "Scaling Regression Inputs by Dividing by Two Standard Deviations." Statistics in Medicine 27: 2865-73.

```
holder.std <- matrix(NA, 4000)
for(i in 1:4000){
  ind <- sample(1:nrow(QOG2), size=nrow(QOG2), replace=T)
  mod <- lm(log2(wdi_gdpcapcon2010) ~ std.polity, data=QOG2[ind,])
  holder.std[i,] <- coef(mod)[2]
}

# 어떤 방식으로 데이터를 구성했는지 조금 직관적으로 이해할 수 있게
# 단순하고 늘어지는 코드를 써보았습니다.
# 먼저 부트스트랩 결과입니다.
std.data1 <- data.frame(ID="Bootstrapped",
                        Low=quantile(holder.std, 0.025),
                        M=mean(holder.std),
                        High=quantile(holder.std, 0.975))
# 부트스트랩은 이미 우리가 4000개의 관측치를 가지고 있기 때문에 이 관측치에서
# 왼쪽꼬리 2.5백분위와 우측꼬리 2.5백분위에 해당하는 값이 95% 신뢰구간의
# 기준값이 됩니다.

# OLS 결과입니다.
std.data2 <- data.frame(ID="OLS",
                        Low=coef(summary(model.sd))[2,1] -
                          1.96*coef(summary(model.sd))[2,2],
                        M=coef(summary(model.sd))[2,1],
                        High=coef(summary(model.sd))[2,1] +
                          1.96*coef(summary(model.sd))[2,2])
std.data <- rbind(std.data1, std.data2)
```



자, 어떤가요? 부트스트랩 표본추출로 구한 계수 추정치와 OLS 추정치의 95% 신뢰구간 결과입니다. 보시다시피 부트스트랩 결과가 조금 큰 계수값과 넓은 신뢰구간을 가지고 있지만 두 계수 추정치는 매우 근사하고 이 차이는 실질적인 분석을 수행하는 데 있어서 무시할만합니다. 뭐 이렇게 말하면 조금 위험할수도 있지만요. 하지만 부트스트랩의 평균과 표준편차가 OLS의 분석적 결과인 계수 추정치와 표준오차 계산을 통한 신뢰구간보다 조금 과대추정되었더라도 부트스트랩 결과는 그 나름의 장점이 있습니다. 우리는 실질적으로 계수값들의 관측치를 통해 분포를 가지고 있기 때문에 그 분포를 이용해 원하는 값을 보여줄 수 있기 때문입니다. 게다가 부트스트래핑은 표본 규모가 작더라도 시도할 수 있기 때문에 어떤 부분에서는 OLS에 대해 상대적 이점이 있다고도 할 수 있습니다.

다음 챕터에서는 비모수 부트스트랩을 넘어 이제 모수적 부트스트랩(parametric bootstrap), 흔히 정치학에서는

AJPS 2000년의 Gary King과 Michael Tomz, 그리고 Jason Wittenberg 논문으로 유명한 Clarify에 대해 살펴보도록 하겠습니다.