9. 비지도학습

지금까지 다룬 기계학습 방법은 모두 지도학습(supervised learning)에 속한다. 지도학습은 입력값 X에 대해 출력값 y를 예측하도록 학습시키는 것을 말한다. 이때 y의 참값을 가지고 모형의 예측값 $\hat{y}$와 비교하여 그 차이를 줄이도록 하기 때문에 '지도'라고 한다.

비지도학습(unsupervised learnng)은 어떤 구조를 가정하고, 데이터 X를 그 구조에 맞춰 설명하는 방법이다. 보통 비지도학습에서는 직접 관찰되지 않았지만 데이터에 영향을 미치는 **잠재 변수**(latent variable) Z를 가정한다.

즉, 지도학습은 관찰 변수 X로 관찰변수 y를 예측한다면, 비지도학습은 잠재변수 Z로 관찰변수 X를 예측하는 것이다. 잠재변수는 말 그대로 '잠재된', '관찰되지 않은' 변수이기 때문에 데이터에 존재하지 않는다.

비지도학습의 대표적인 예로는 클러스터링(clustering)과 차원 축소(dimensionality reduction)이 있다. 클러스터링은 잠재변수 Z가 이산 변수 또는 범주형 변수인 것을 말한다. 차원 축소는 연속 변수인 경우다.

9.1. 차원 축소

차원 축소는 적은 갯수의 연속 변수(차원) Z로 데이터 X를 나타내는 것이다. 차원 축소의 용도는 다음과 같다. 1) 복잡한 데이터를 2차원으로 시각화해서 나타내기 위해 2) 변수들 사이의 공통적인 정보만 남겨 점수화시키기 위해 3) 차원을 줄여 차원의 저주를 피하고 지도학습의 성능을 높이기 위해.

전통적인 통계에서 차원 축소의 대표적인 방법은 주성분 분석(Principal Component Analysis: 이하 PCA)이 있다. 주성분 분석은 데이터의 분산을 가장 많이 설명하는 순서대로 성분(새로운 축)을 만들고, 설명량이 적은 성분을 없애는 방식이라 차원을 축소한다.

PCA를 텍스트 데이터에 적용한 것을 잠재의미분석(latent semantic analsysis), 이미지 데이터에 적용한 것을 고유이미지(eigenimage)라고 한다.

PCA 이외에도 LLE, Isomap, MDS, t-SNE 등 다양한 차원 축소 방법이 있다. 특히 t-SNE는 복잡한 데이터의 시각화 용도로 많이 쓰인다.

인공신경망에서도 일종의 차원축소가 일어난다고 볼 수 있다. 왜냐하면 보통 입력층의 크기보다 은닉층의 크기가 작기 때문이다. 이러한 특성을 적극적으로 이용한 방법으로 오토인코더(autoencoder)가 있다. 오토인코더는 입력과 출력이 똑같은 형태의 특수한 신경망이다. 오토인코더는 은닉층이 입력층보다 작기 때문에 정보를 더 적은 변수로 줄였다가 다시 원래대로 복원하게 된다. 이 과정에서 은닉층이 입력층의 정보를 가장 잘 복원할 수 있는 형태로 학습이 이뤄지게 된다. 오토인코더는 데이터에 담긴 비선형적인 패턴을 보존할 수 있는 차원 축소 방법으로 많은 각광을 받았다.

9.2. 클러스터링

클러스터링 또는 군집 분석은 데이터를 비슷한 것들끼리 일정한 무리 또는 군집으로 나누는 것이다.

예를 들어 살펴보자. 고객 데이터를 다룰 때 고객을 여러 가지 군집으로 나눌 수 있다. 클러스터링은 고객마다 어떤 Z에 속하고 같은 군집끼리는 서로 비슷하다는 가정을 한다. 그리고 그 가정을 가장 잘 만족하도록 고객들을 군집으로 나눈다.

클러스터링과 지도학습에서 분류(classification)은 똑같이 이것 아니면 저것 식으로 예측을 하기 때문에 비슷해보인다. 그러나 지도학습에서는 클래스 y가 관찰변수고, 클러스터링에서는 군집 Z가 잠재변수라는 것이 차이다. 즉, 데이터에 고객들의 클래스가 어떤 식으로든 존재하고 그 클래스를 직접 학습한다면 분류가 된다. 그러나 데이터에 직접적으로 군집이 존재하지 않고 어떤 가정으로부터 고객들의 군집을 나눈다면 클러스터링이 된다. 다른 말로 한다면 우리가 어떤 가정을 하느냐에 따라 같은 데이터에서도 군집은 전혀 달라질 수도 있다.

군집을 하는 이유는 비슷한 군집에 비슷한 대응을 할 수 있기 때문이다. 고객들을 군집으로 묶으면 같은 군집에는 같은 프로모션을 하거나, 같은 제품을 추천할 수 있다. 모든 고객에게 동일한 대응을 하는 것보다 비슷한 고객들에 비슷한 대응을 하는 것이 더 효과적일 것이다.

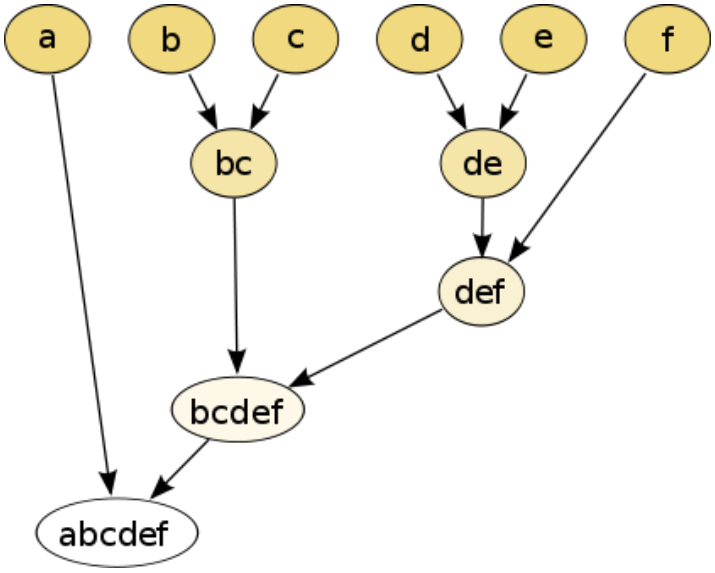
또한 데이터가 많이 있을 때 군집을 하면 데이터를 일일이 살펴보는 수고를 줄일 수있다. 대량의 텍스트 데이터가 있을 경우 모든 텍스트를 읽어보는 것보다 비슷한 텍스트를 군집으로 만들고 각 군집의 대표적인 텍스트를 읽어본다면 시간과 노력을 절약할 수 있다.

군집 분석의 대표적인 방법들에는 위계적 군집분석과 K-Means, 가우시안 혼합 모형이 있다.

9.2.1. 위계적 군집 분석

위계적 군집 분석(hierarchical clustering)은 반복해서 데이터를 뭉치거나 쪼개는 방식으로 군집을 만드는 방법이다. 보통

은 뭉치는 방법을 사용한다.



위계적 군집 분석에서 데이터를 뭉치는 방법은 다음과 같다. 1) 모든 데이터를 하나의 군집으로 본다. 데이터가 100개가 있으면 군집도 100개가 있는 것이다. 2) 모든 군집들 사이의 거리를 잰다. 3) 거리가 가장 가까운 두 군집을 골라 하나의 군집으로 합친다. 4) 원하는 갯수의 군집이 될 때까지 2~3을 반복한다.

이때 군집 간의 거리를 재는 방법에 따라 세부적인 방법이 달라진다.

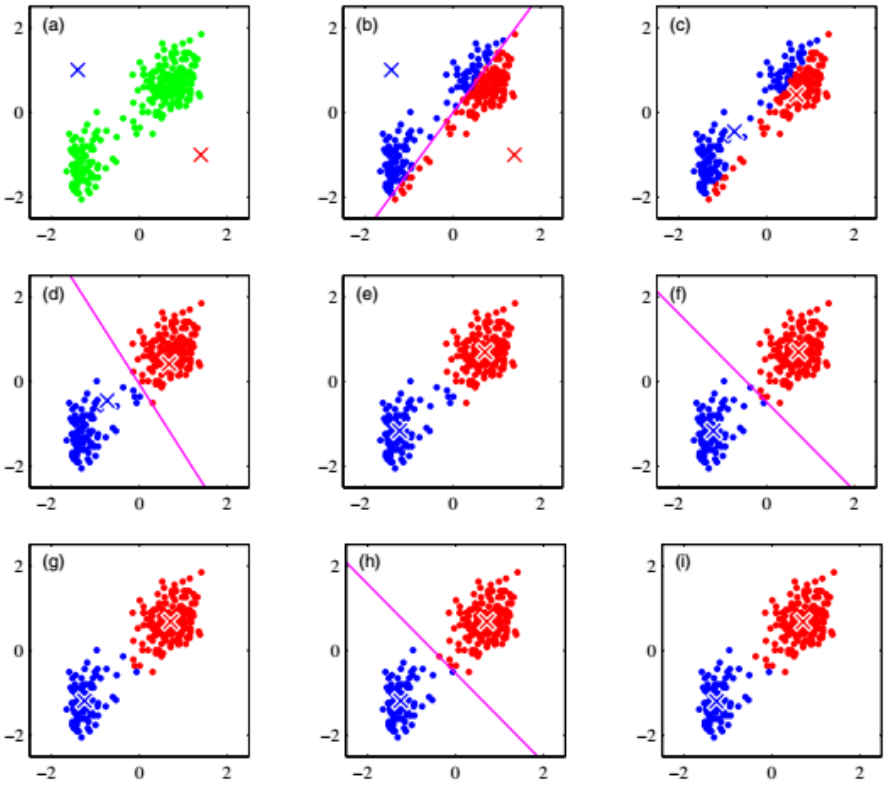
- Single link: 두 군집에서 가장 가까운 멤버들의 거리를 잰다. 긴 체인(chain)을 만드는 경향이 있다.
- Complete link: 두 군집에서 가장 먼 멤버의 거리를 잰다. 구형(spherical)으로 뭉치는 경향이 있다.
- Average link: 모든 멤버들 사이의 거리를 평균낸다.
- Centroids: 군집의 중심과 중심의 거리를 잰다.
- Ward's method: 두 군집을 합쳤을 때 군집 내 거리 분산의 변화를 거리의 척도로 삼는다.

9.2.2. K-Means

K-Means는" k개의 평균들"이라는 뜻이다. K-Means는 데이터와 군집의 중심점의 거리를 재서, 그 거리가 가장 가까운 군집에 속한다고 가정한다.

K-Means는 다음과 같은 방식으로 군집을 만든다. 1) k개의 중심점을 무작위로 정한다. 2) 중심점과 거리를 재서 가장 가까운 군집에 모든 데이터를 할당한다. 3) 군집에 속한 멤버들의 평균을 내서 중심점을 정한다. 4) 더이상 중심점에 변화가 없을 때까지 2~3을 반복한다.

아래 그림은 2개의 군집을 가정하고 K-Means를 실시했을 때 군집을 찾아가는 과정이다.



K-Means와 같이 비지도학습에서는 일단 답을 먼저 가정한 다음에 그 가정으로부터 답을 점점 개선시켜 나가는 방법을 많이 쓴다. 다음에 알아볼 가우시안 혼합 모형은 이를 더 일반환시킨 것이다.

9.2.3. 가우시안 혼합 모형

가우시안 혼합 모형(Gaussian Mixture Models: 이하 GMM)은 전체 데이터의 분포가 여러 개의 가우시안 분포(Gaussian distribution), 즉 정규 분포(normal distribution)가 혼합된 것이라고 가정한다.

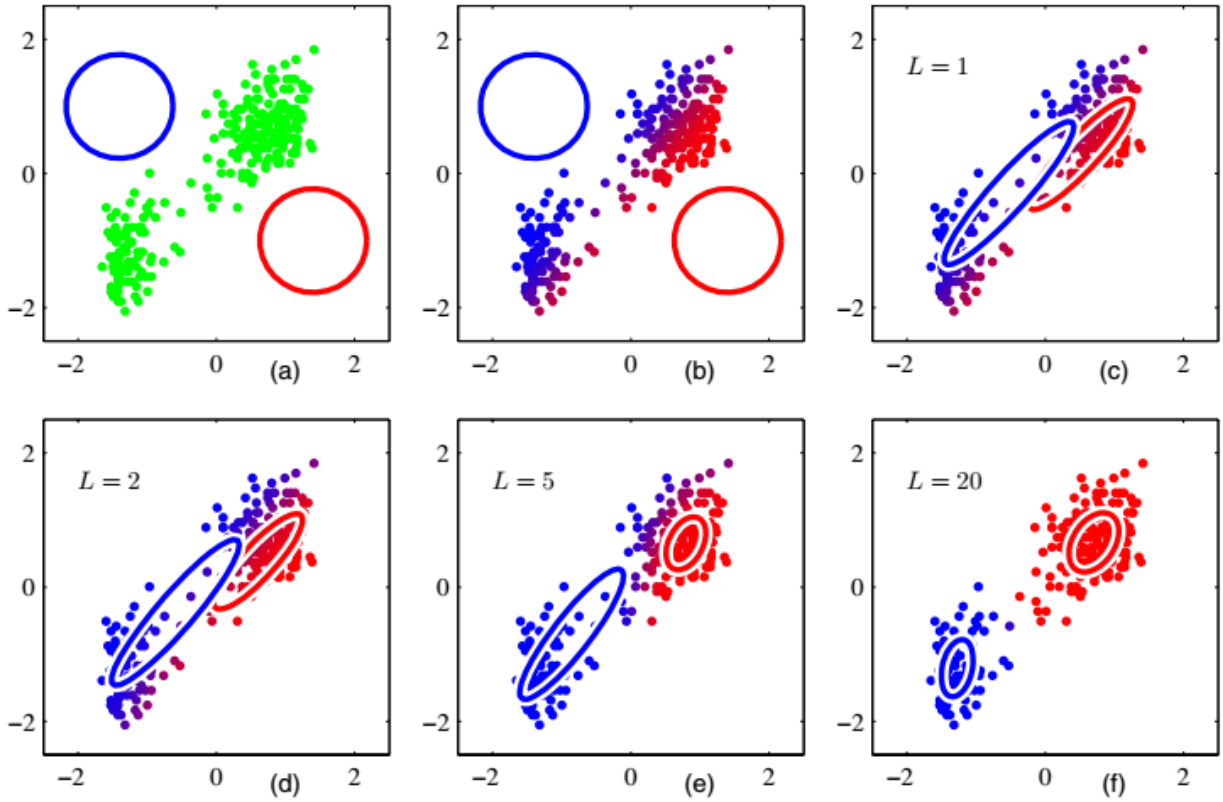
$$P(x) = \sum_{z} P(x|z)P(z)$$

어떤 데이터가 z번째 분포에 속할 확률을 혼합 계수(mixing coefficient) π_k라고 한다. 또 각각의 z 번째 분포는 평균 μ_k와 공분산 Σ_k를 가진다. 따라서 위의 식은 다음과 같이 쓸 수 있다.

$$P(x) = \sum_{k=1}^{K} N(x|\mu_k, \Sigma_k)\pi_k$$

가우시안 혼합 모형에서는 각 데이터가 특정 군집에 속하거나 속하지 않는 것이 아니라 속할 확률을 갖는다. 즉 2개의 군집이 있으면 1번 군집에 속할 확률 .7, 2번 군집에 속할 확률 .3 같은 식이 될 수 있다. 이러한 특성을 soft assignment라고 한다.

GMM의 추정은 EM 알고리즘을 사용한다. EM 알고리즘은 예상(Expectatoin)과 최대화(Maximization)을 반복하는 알고리즘으로 K-Means 알고리즘을 일반화시킨 것으로 볼 수 있다. 1) 각 분포의 평균과 공분산, 혼합계수를 무작위로 초기화한다. E) 각 데이터가 각 분포에 속할 확률 $P(z|x)$를 예상한다. M) 앞의 예상 아래 $P(X)$가 최대화되도록 평균, 공분산, 혼합계수를 조정한다. 4) 더이상 분포에 변화가 없을 때까지 E와 M을 반복한다.



9.3. 비지도학습 실습

9.3.1. iris 데이터

데이터는 붓꽃(iris) 데이터를 사용한다. 해당 데이터는 R에 내장되어 있다.

```
data(iris)
```

붓꽃 데이터에는 꽃잎의 길이와 너비, 꽃받침의 길이와 너비, 그리고 품종의 5가지 변수가 포함되어 있다.

```
names(iris)
```

```
[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

9.3.2. 시각화

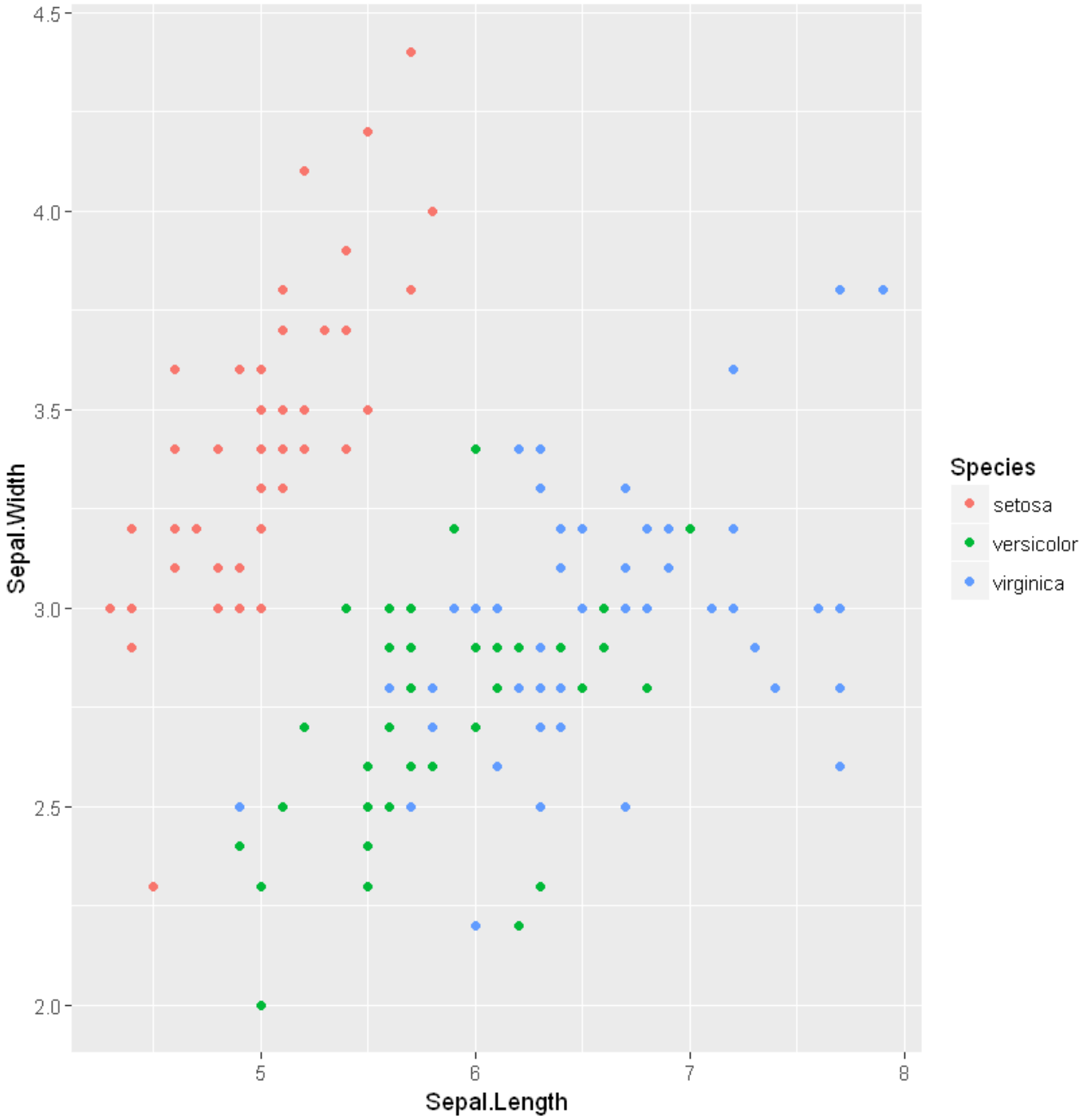
`ggplot2`라이브러리를 통해 시각화를 해보자.

```
library(ggplot2)
```

품종별로 색깔을 달리해서 꽃잎의 길이와 너비를 그래프로 그려보면 아래와 같다.

```
ggplot(iris) + geom_point(aes(x=Sepal.Length, y=Sepal.Width, color=Species))
```

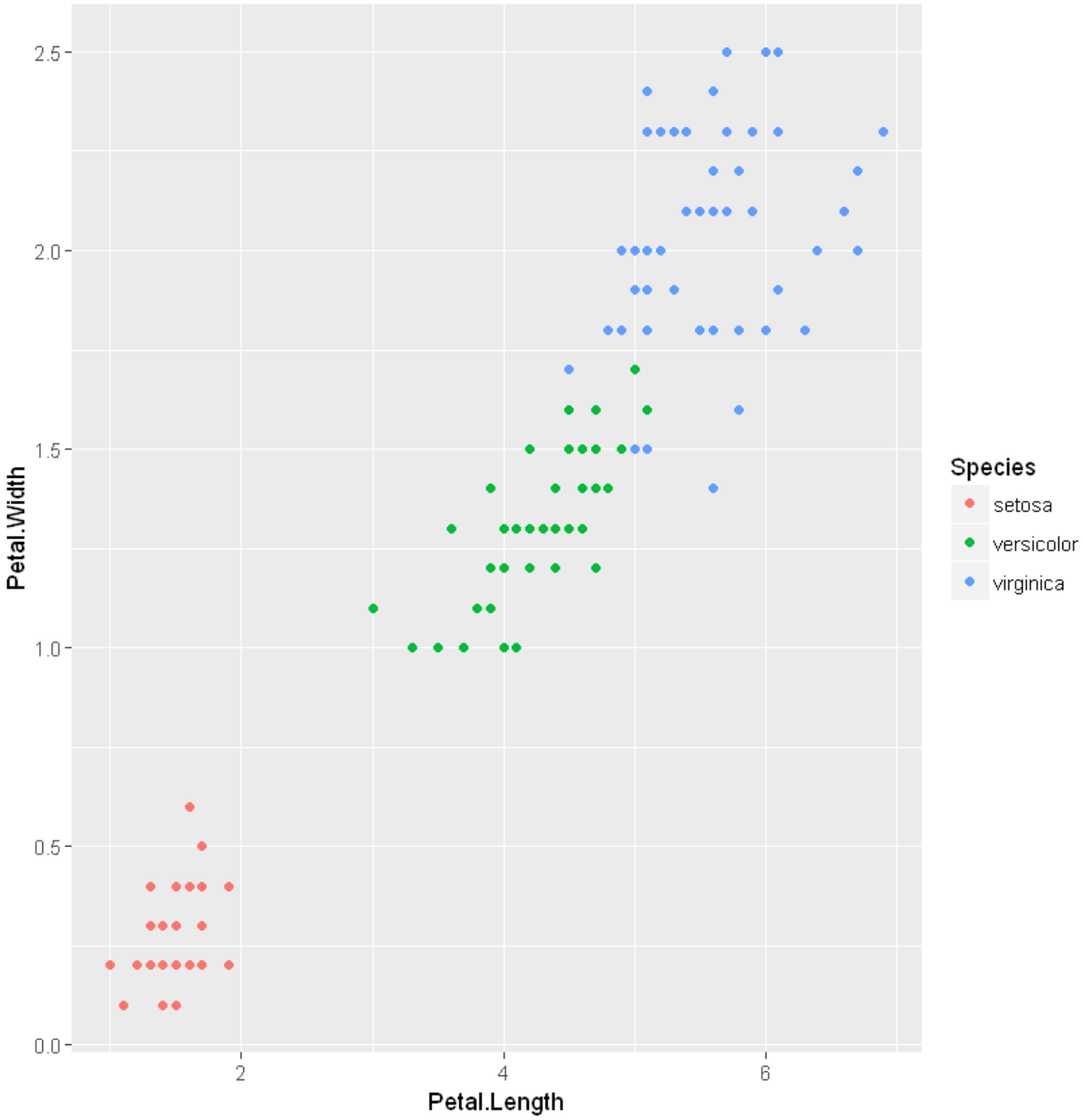
```
plot without title
```



꽃받침의 길이와 너비로 시각화해보면 아래와 같다.

```
ggplot(iris) + geom_point(aes(x=Petal.Length, y=Petal.Width, color=Species))
```

```
plot without title
```

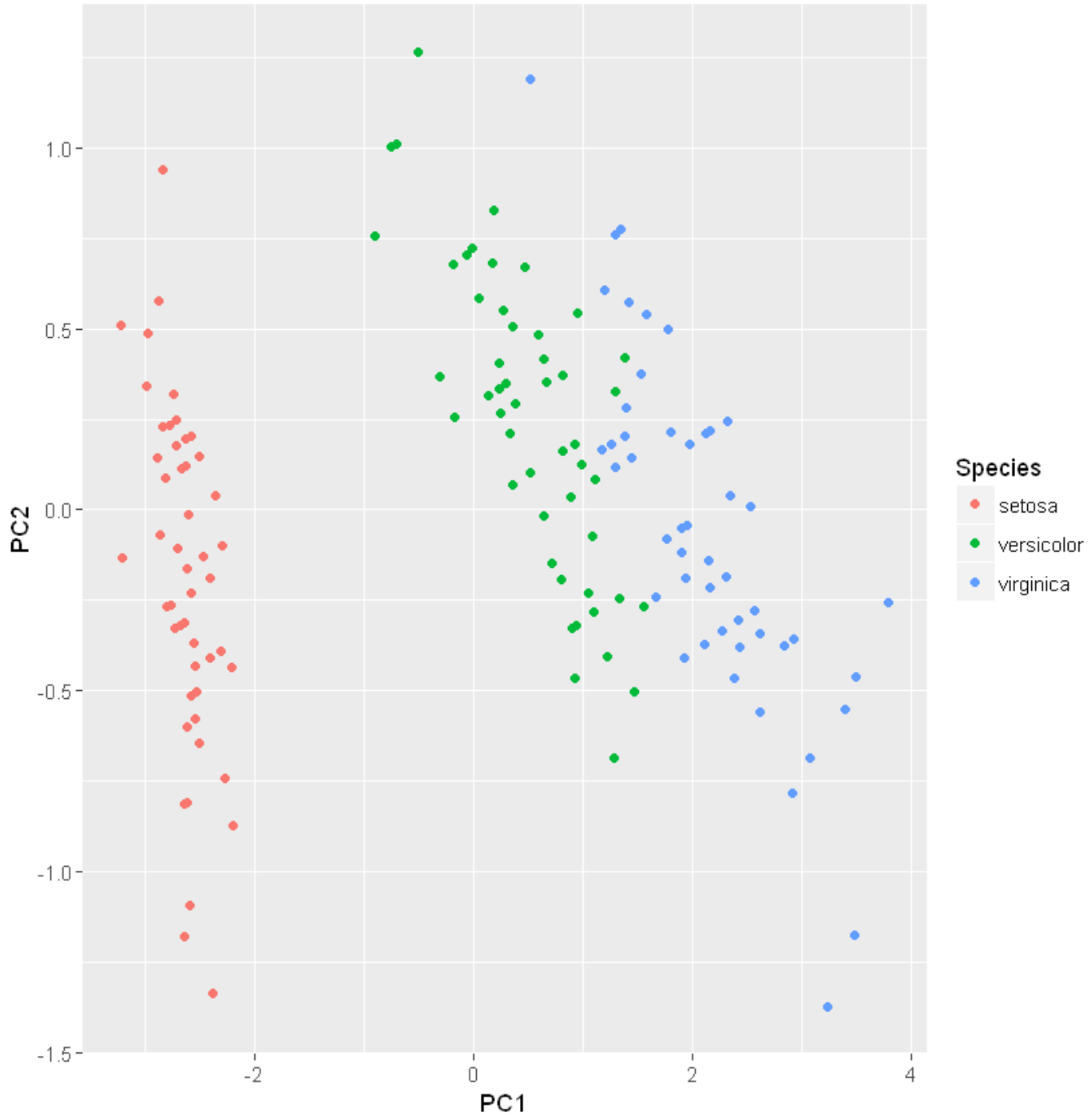


9.3.3. 차원 축소

꽃의 정보가 4가지가 있기 때문에 이 4가지를 한 화면에 그리기 어렵다. PCA로 2차원으로 축소해서 그려보도록 하자.

```
pc = prcomp(iris[1:4])
new_iris = data.frame(pc$x, Species=iris$Species)
ggplot(new_iris) + geom_point(aes(x=PC1, y=PC2, color=Species))
```

plot without title

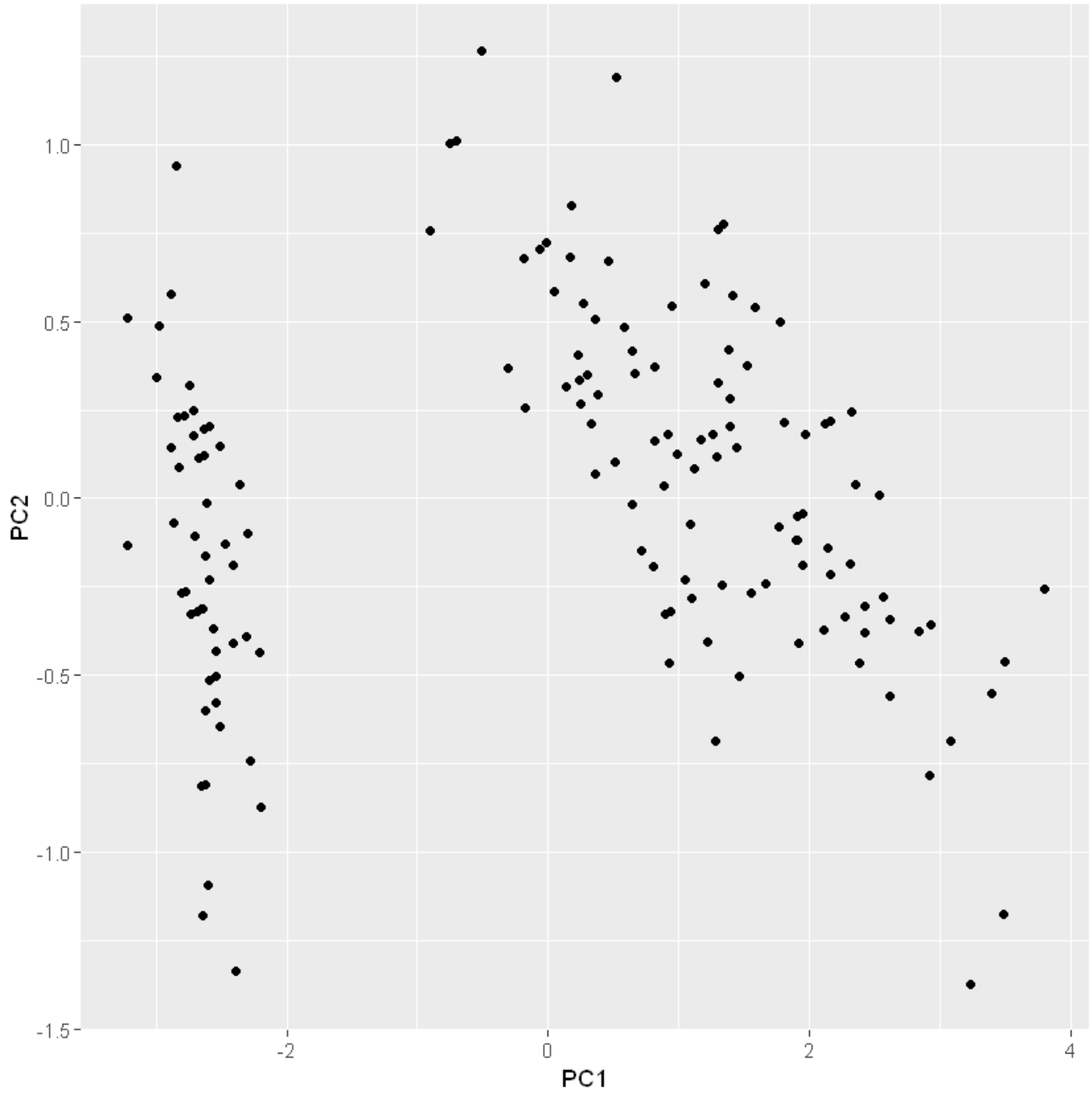


9.3.4. 클러스터링

이제 클러스터링을 하기 전에 데이터에서 원래 품종 정보를 제외하고 시각화를 해보자. 만약 아래의 그림만 보고 클러스터링을 한다면 몇 개의 클러스터로 나뉘겠는까? 또 각각의 데이터는 어떤 클러스터에 속하겠는가?

```
ggplot(new_iris) + geom_point(aes(x=PC1, y=PC2))
```

plot without title



위계적 군집분석

아이리스 원본 데이터에서 클래스를 제외한 나머지 정보는 아래와 같다.

```
head(iris[1:4])
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width
1 5.1          3.5         1.4          0.2
2 4.9          3.0         1.4          0.2
3 4.7          3.2         1.3          0.2
4 4.6          3.1         1.5          0.2
5 5.0          3.6         1.4          0.2
6 5.4          3.9         1.7          0.4
```

이 모든 점들 사이의 거리를 측정해보자.

```
d = dist(iris[1:4])

as.matrix(d)[1:3, 1:3]
```

```
1         2         3
1 0.0000000 0.5385165 0.509902
2 0.5385165 0.0000000 0.300000
```

```
3 0.5099020 0.3000000 0.000000
```

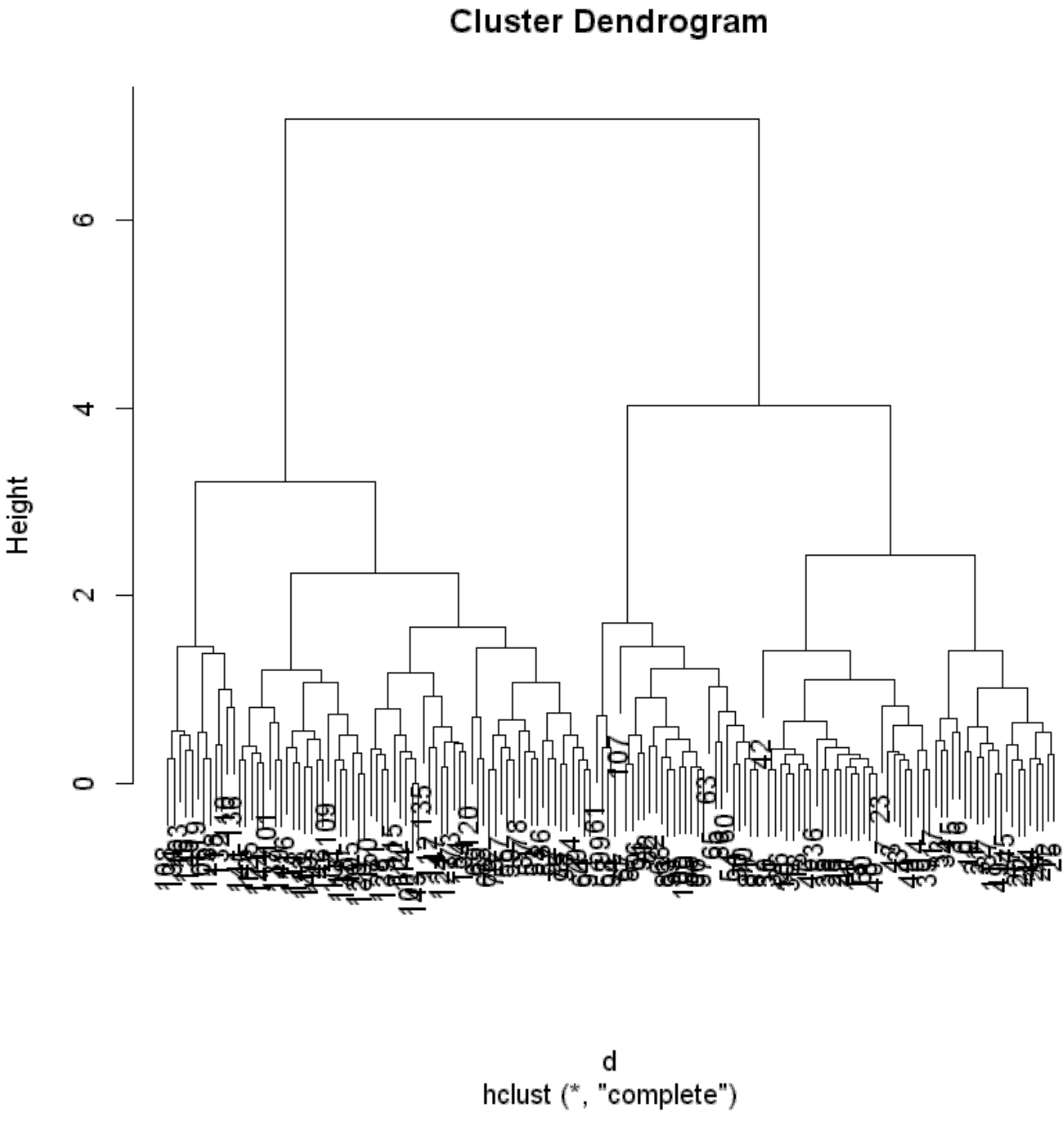
이 거리를 바탕으로 위계적 군집분석을 실시한다. 거리 측정 방법은 `single`, `complete`, `average`, `ward`를 쓸 수 있다.

```
h.comp = hclust(d, 'complete')
```

클러스터링 과정을 그림으로 그려보면 아래와 같다.

```
plot(h.comp)
```

```
Plot with title "Cluster Dendrogram"
```



클러스터를 3개로 나눠보자.

```
groups <- cutree(h.comp, k=3)
```

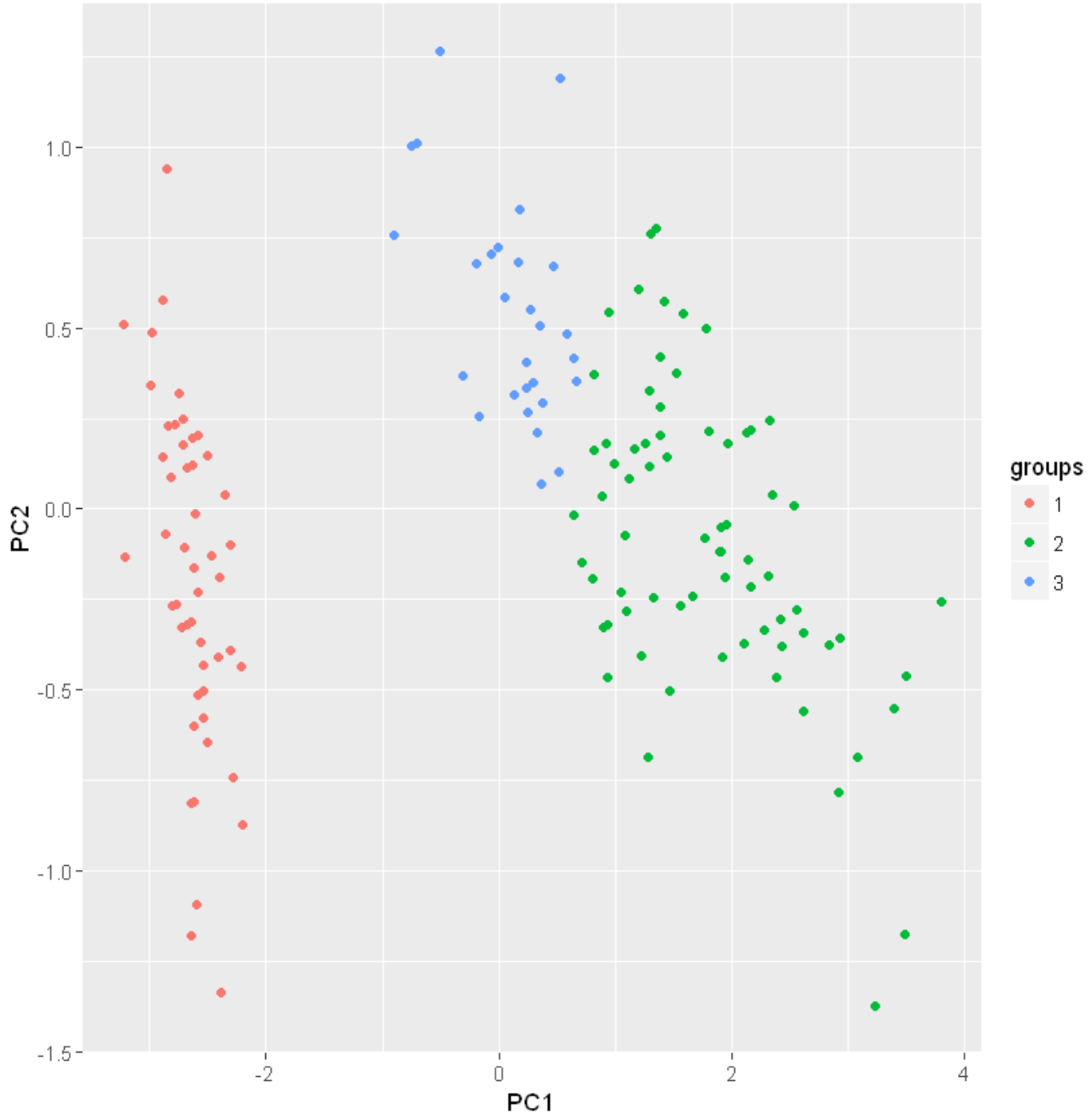
```
groups
```

```
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 2 3 2 3 2 3 3 3 3 2 3 2 3 3 2 3 2 3 2 2
 [75] 2 2 2 2 2 3 3 3 3 2 3 2 2 2 3 3 3 2 3 3 3 3 2 3 3 2 2 2 2 2 2 3 2 2 2 2
[112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[149] 2 2
```

나눠진 클러스터를 시각화해보자.

```
hcdata = data.frame(pc$x, groups=as.factor(groups))
ggplot(hcdata) + geom_point(aes(x=PC1, y=PC2, color=groups))
```

```
plot without title
```



K-Means

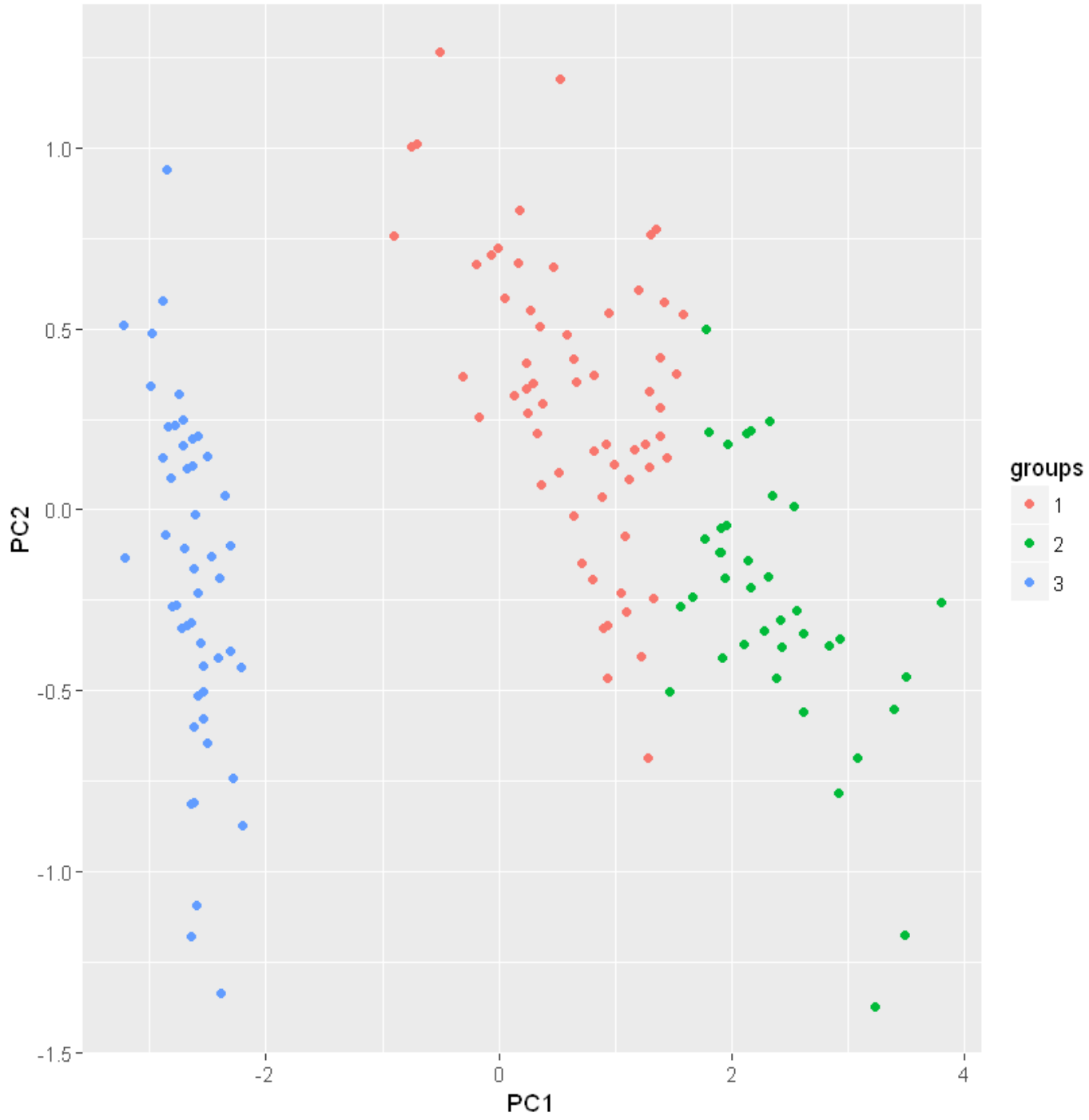
이번에는 K-Means다. K-Means는 간단하다.

```
kc = kmeans(iris[1:4], 3)
```

```
kc
```

```
K-means clustering with 3 clusters of sizes 62, 38, 50

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     5.901613    2.748387     4.393548    1.433871
2     6.850000    3.073684     5.742105    2.071053
3     5.006000    3.428000     1.462000    0.246000

Clustering vector:
```

```
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2
[112] 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2
[149] 2 1

Within cluster sum of squares by cluster:
[1] 39.82097 23.87947 15.15100
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"

kc$cluster

[1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [38] 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [75] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2
[112] 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2
[149] 2 1

kmdata = data.frame(pc$x, groups=as.factor(kc$cluster))
ggplot(kmdata) + geom_point(aes(x=PC1, y=PC2, color=groups))

plot without title
```



가우시안 혼합 모형

마지막으로 가우시안 혼합 모형이다. GMM은 `mclust` 라이브러리가 필요하다. 없는 경우 먼저 `install.packages('mclust')`로 설치를 하자.

```
install.packages('mclust', repos='http://cloud.r-project.org')
```

```
package 'mclust' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
 C:\Users\user\AppData\Local\Temp\RtmpqY7ZQW\downloaded_packages
```

```
library(mclust)
```

```
Warning message:
: package 'mclust' was built under R version 3.3.3Package 'mclust' version 5.4
Type 'citation("mclust")' for citing this R package in publications.
```

```
mc = Mclust(iris[1:4], 3)
```

```
summary(mc)
```

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
```

```
----------------------------------------------------

Mclust VEV (ellipsoidal, equal shape) model with 3 components:

 log.likelihood   n df       BIC       ICL
       -186.074 150 38 -562.5522 -566.4673

Clustering table:
 1  2  3
50 45 55
```

GMM은 soft assignment를 하므로 데이터마다 정해진 군집이 없다. 대신 가장 확률이 높은 군집을 얻을 수 있다.

```
mc$classification
```

```
[1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 3 2
 [75] 2 2 2 3 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[149] 3 3
```

```
mcdata = data.frame(pc$x, groups=as.factor(mc$classification))
ggplot(mcdata) + geom_point(aes(x=PC1, y=PC2, color=groups))
```

```
plot without title
```