주성분분석에서는

 - 상관행렬과 공분산행렬 중 어느 것을 선택할 것인가?

 - 주성분의 개수를 몇 개로 할 것인가?

 - 주성분에 영향을 미치는 변수로 어떤 변수를 선택할 것인가?

에 대해서 결정을 해야 하는데요,

한국신용평가정보에서 나온 '국내 증권회사의 주요 재무제표' (2007.3.31 기준)를 가지고 주성분분석을 R로 해보도록 하겠습니다. (다음번 포스팅에서는 똑같은 데이터에 대해 요인분석을 해보겠습니다)

이 데이터는 18개 증권사별로 V1.총자본순이익율, V2.자기자본순이익율, V3.자기자본비율, V4.부채비율, V5.자기자본회전율 재무지표 변수로 구성되어 있습니다.

예제 데이터('국내 증권회사의 주요 재무제표' (2007.3.31 기준)) 다운로드 ☞

[[https://t1.daumcdn.net/tistory_admin/blogs/image/extension/unknown.gif?_version_=2d70ce7a760b0f0473f428ff8ef4be21c33e9133](http://rfriend.tistory.com/attachment/cfile21.uf@2577A64555C56ECF0CF0C8.csv) secu\_com\_finance\_2007.csv](http://rfriend.tistory.com/attachment/cfile21.uf@2577A64555C56ECF0CF0C8.csv)

R로 외부 csv 데이터 불러오기

 secu\_com\_finance\_2007 <- Read.csv (file.choose(), header = True, stringsAsFactors = False)

X <- data.frame(secu\_com\_finance\_2007)

Str(x) 를 사용해서 자료의 구조를 확인 필요

|  |
| --- |
| > ##----------------------------------------------------------------------  > ## 차원축소(dimension reduction) : (1) PCA(Principal Component Analysis)  > ##----------------------------------------------------------------------  >  > # csv 파일 불러오기 (file importing)  > secu\_com\_finance\_2007 <- read.csv("C:/Users/user/Documents/R/secu\_com\_finance\_2007.csv",  + header = TRUE,  + stringsAsFactors = FALSE)  > # V1 : 총자본순이익율  > # V2 : 자기자본순이익율  > # V3 : 자기자본비율  > # V4 : 부채비율  > # V5 : 자기자본회전율 |

주성분분석에서는 변수별로 단위가 다른 raw data를 사용하지 않고 평균과 표준편차를 가지고 표준화(standadization)한 데이터를 사용합니다. 그래야 scale이 다른 문제로 인한 데이터 왜곡을 피할 수 있기 때문입니다.

|  |
| --- |
| > # 표준화 변환 (standardization)  > secu\_com\_finance\_2007 <- transform(secu\_com\_finance\_2007,  + V1\_s = scale(V1),  + V2\_s = scale(V2),  + V3\_s = scale(V3),  + V4\_s = scale(V4),  + V5\_s = scale(V5)) |

V1.총자본순이익율, V2.자기자본순이익율, V3.자기자본비율, V5.자기자본회전율의 네 개의 변수는 숫자가 클 수록 좋다는 뜻이지만 V4.부채비율는 높을 수록 안 좋다고 해석하게 됩니다.  즉 V1, V2, V3, V5와 V4는 반대방향으로 움직이게 되는데요, 서로 같은 방향으로 움직이게 해서 상관도가 높게 나와 같은 주성분에 반영되도록 하기 위해서 아래와 같이 V4.부채비율의 방향을 변환(표준화된 이후의 max 값에서 표준화된 이후의 관찰값을 뺌)하겠습니다. (부채비율 방향 변환 후에는 숫자가 높을 수록 좋은 회사라고 해석할 수 있습니다)  그리고 주성분분석에 필요한 변수(V4\_s가 아니라 V4\_s2 가져온거 유의)만 indexing해서 선별하였습니다.

|  |
| --- |
| > # 부채비율(V4\_s)을 방향(max(V4\_s)-V4\_s) 변환  > secu\_com\_finance\_2007 <- transform(secu\_com\_finance\_2007,  + V4\_s2 = max(V4\_s) - V4\_s)  >  > # variable selection  > secu\_com\_finance\_2007\_2 <- secu\_com\_finance\_2007[,c("company", "V1\_s", "V2\_s", "V3\_s", "V4\_s2", "V5\_s")] |

먼저, 변수들간의 상관계수를 분석해보겠습니다.  주성분분석이 변수들 간의 상관관계가 높다는 것을 가정하고 있기 때문에 한번 확인해보도록 하겠습니다.

V1\_s.총자본순이익률과 V2\_s.자기자본순이익율이 상관관계가 높고(상관계수 0.615), V3\_s.자기자본비율과 V4\_s2.(방향변환 후의)부채비율이 상관관계가 매우 높게(상관계수 0.936) 나왔습니다.  V5\_s.자기자본회전율은 V2\_s.자기자본순이익율과 상관관계가 있고, V3\_s.자기자본비율과 V4\_s2.(방향전환후의 부채비율)과는 역의 상관관계가 나왔네요.

|  |
| --- |
| > # Correlation analysis  > cor(secu\_com\_finance\_2007\_2[,-1]) 여기서 [,-1]은 데이터 메트릭스에서 첫번째 컬럼을 제외하라는 의미  V1\_s V2\_s V3\_s V4\_s2 V5\_s  V1\_s 1.00000000 0.6165153 0.3239780 0.3553930 0.01387883  V2\_s 0.61651527 1.0000000 -0.5124351 -0.4659444 0.42263462  V3\_s 0.32397800 -0.5124351 1.0000000 0.9366296 -0.56340782  V4\_s2 0.35539305 -0.4659444 0.9366296 1.0000000 -0.53954570  V5\_s 0.01387883 0.4226346 -0.5634078 -0.5395457 1.00000000  >  > round(cor(secu\_com\_finance\_2007\_2[,-1]), digits=3) # 반올림 |

 전체 집단을 두개로 나누는 군집분석은 secu\_com\_finance\_2007\_cl = kmeans(secu\_com\_finance\_2007[,-1], center=2)

변수들간의 산점도 행렬도 살펴보도록 하죠.

|  |
| --- |
| > # Scatter plot matrix  > plot(secu\_com\_finance\_2007\_2[,-1])    https://t1.daumcdn.net/cfile/tistory/2363AB3355C57A4F37 |

이제 prcomp() 함수를 사용해서 주성분분석을 실시합니다.  아래 결과에 보면 누적기여율(Cummulative Proportion)에 제1주성분(PC1)이 55.23%, 제 2주성분(PC1 & PC2)까지의 누적기여율dl 87.34%로 매우 높게 나왔습니다.

|  |
| --- |
| > # 주성분분석 PCA(Principal Component Analysis)  > secu\_prcomp <- prcomp(secu\_com\_finance\_2007\_2[,c(2:6)]) # 첫번째 변수 회사명은 빼고 분석  >  > summary(secu\_prcomp)  Importance of components:  PC1 PC2 PC3 PC4 PC5  Standard deviation 1.6618 1.2671 0.7420 0.25311 0.13512  Proportion of Variance 0.5523 0.3211 0.1101 0.01281 0.00365  Cumulative Proportion 0.5523 0.8734 0.9835 0.99635 1.00000 |

아래에 주성분분석 결과를 출력해보았습니다. 제1요인이 표준편차가 1.66으로 가장 크고, 제2요인이 1.26으로 그 다음으로 큰 식으로 순서가 있습니다.

그리고 Rotation 후의 고유벡터(eigenvector)의 계수를 보면 제1요인(PC1)은 자기자본비율(V3\_s)와 부채비율(V4\_s2)와 관련이 있고, 제 2요인(PC2)은 총자본순이익율(V1\_s)과 자기자본순이익율(V2\_s)와 관련이 있으며, 제 3요인(PC3)은 자기자본회전율(V5\_s)와 관련이 있음을 알 수 있습니다.

|  |
| --- |
| > print(secu\_prcomp)  Standard deviations:  [1] 1.6617648 1.2671437 0.7419994 0.2531070 0.1351235  Rotation:  PC1 PC2 PC3 PC4 PC5  V1\_s 0.07608427 -0.77966993 0.0008915975 -0.140755404 0.60540325  V2\_s -0.39463007 -0.56541218 -0.2953216494 0.117644166 -0.65078503  V3\_s 0.56970191 -0.16228156 0.2412221065 -0.637721889 -0.42921686  V4\_s2 0.55982770 -0.19654293 0.2565972887 0.748094314 -0.14992183  V5\_s -0.44778451 -0.08636803 0.8881182665 -0.003668418 -0.05711464 |

이처럼 변수와 주성분간 관계를 고려해서 주성분에 이름을 명명(naming)해보자면 PC1은 안정성, PC2는 수익성, PC3는 활동성이라고 할 수 있겠네요. (재무제표 배울 때 배우는 대표적인 재무평가 지표)

|  |
| --- |
| https://t1.daumcdn.net/cfile/tistory/2164943E55C589FE2D |

선형대수를 안배우면 용어, 이론 개념이 어려울 수 있는데요, 참고로 고유값(eigenvalue)와 고유벡터(eigenvector)에 대한 정의를 아래에 소개합니다.

|  |
| --- |
| [참고: 고유값(eigenvalue), 고유벡터(eigenvector) 용어설명]    https://t1.daumcdn.net/cfile/tistory/2640C34255C5868926 |

주성분의 개수를 정할 때 누적기여율을 가지고 하기도 하고, 또는 고유값(eigenvalue)을 가지고 아래 처럼 Scree plot 그래프를 그려서 고유값 곡선이 꺽이는 지점의 바로 앞의 주성분 개수를 선택하기도 합니다.  아래 그래프로 보면 주성분 4개째에서 수평으로 드러누웠으므로 한개를 뺀 (4-1 = 3) 3개 주성분이 적합해 보입니다.

|  |
| --- |
| > # Scree Plot  > plot(prcomp(secu\_com\_finance\_2007\_2[,c(2:6)]), type="l",  + sub = "Scree Plot")  Plot (secu\_prcomp)로 명령해도 동일한 효과, 여기서 Type = “l” 1이 아니라 L의 소문자 라인을 의미    https://t1.daumcdn.net/cfile/tistory/222C443455C57F5713 |

주성분1점수(principal component 1 score)과 주성분2점수(principal component 2 score)를 가지고 Biplot을 그려보겠습니다.

가로축 PC1(안정성)을 기준으로 보면 오른쪽에 V3\_s, V4\_s2 화살표가 향하는 쪽에 있는 부국증권, 한양증권, 유화증권사는 안정성이 높은 회사군이라고 해석을 할 수 있게 됩니다.

(참고: **PC1** = 0.076\*V1\_s - 0.394\*V2\_s + **0.569\*V3\_s** + **0.559\*V4\_s2** - 0.447\*V5\_s )

세로축 PC2(수익성)을 기준으로 보면 아래쪽에 있는 대우증권, 미래애셋증권사 등이 수익성이 좋은 축에 속합니다. (아래 식 부호가 (-) 라서 숫자 낮은 것이 수익성 좋다는 뜻)

(참고: **PC2** = **-0.779\*V1\_s - 0.565\*V2\_s** - 0.162\*V3\_s - 0.196\*V4\_s2 - 0.086\*V5\_s )

|  |
| --- |
| > # Biplot  > biplot(prcomp(secu\_com\_finance\_2007\_2[,c(2:6)]), cex = c(0.7, 0.8))  >  > # 관측치별 주성분1, 주성분2 점수 계산(PC1 score, PC2 score)  > secu\_pc1 <- predict(secu\_prcomp)[,1]  > secu\_pc2 <- predict(secu\_prcomp)[,2]  >  >  > # 관측치별 이름 매핑(rownames mapping)  > text(secu\_pc1, secu\_pc2, labels = secu\_com\_finance\_2007\_2$company,  + cex = 0.7, pos = 3, col = "blue")    https://t1.daumcdn.net/cfile/tistory/24328D3E55C587F208 |

이렇게 변환한 주성분점수를 가지고 다른 통계모형이나 데이터마이닝 모형 개발 시 input으로 활용해도 되겠습니다.

참고로, 변수에 대한 설명력의 누적기여율(cummulative proportion)이 80%가 되는 주성분의 개수 k개를 찾아서, 주성분 1번부터 주성분 k번째까지의 주성분점수를 반환하는 사용자 정의함수는 아래와 같습니다.

(Dr.Kevin 님의 댓글 덕분에 오류 잡아서 프로그램 수정하였습니다. Dr.Kevin님 감사합니다)

|  |
| --- |
| > ###########################################################  > ## PCA (Principal Component Analysis)  > ## User Defined Function  > ## - finding PC k which Cumulative Proportion is over 0.8  > ###########################################################  >  > pca <- function(dataset){  + pc = prcomp(dataset, scale = TRUE)  +  + k = 0  + R = 0  +  + while(R < 0.8) {  + k = k + 1  + R = sum(pc[[1]][1:k]^2)/sum(pc[[1]]^2)  +  + cat("When number of Principal Component(k) is ", k,  + ", Cumulative Proportion(R) is ", R, "\n", "\n", sep="")  + }  +  + SelectedDataSet = pc[[5]][,1:k]  + return(SelectedDataSet)  + }  >  > pca(secu\_com\_finance\_2007\_2[,c(2:6)])  When number of Principal Component(k) is 1, Cumulative Proportion(R) is 0.5522924  When number of Principal Component(k) is 2, Cumulative Proportion(R) is 0.8734231  PC1 PC2  [1,] -1.4870243 0.6066594  [2,] -0.2063797 -0.0804627  [3,] 0.1968538 0.9704605  [4,] -2.3542884 -3.5056480  [5,] -0.8953707 1.4552899  [6,] -0.3682082 -0.5976313  [7,] -0.9354306 -1.4144519  [8,] 2.4129728 -0.6785064  [9,] 2.6991862 -0.7596591  [10,] -0.4050098 0.2800099  [11,] 1.3958199 1.1353513  [12,] -1.5381192 1.1576616  [13,] 0.3217681 -0.2378023  [14,] -2.0306806 0.9646122  [15,] 3.0389460 0.8841645  [16,] 2.0064063 -1.2831337  [17,] -0.4211779 -0.2987099  [18,] -1.4302634 1.4017959 |

출처: <http://rfriend.tistory.com/61> [R, Python 분석과 프로그래밍 (by R Friend)]

**데이터 - Iris**

Iris 데이터는 다음과 같이 구성되어 있다.

head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species

## 1 5.1 3.5 1.4 0.2 setosa

## 2 4.9 3.0 1.4 0.2 setosa

## 3 4.7 3.2 1.3 0.2 setosa

## 4 4.6 3.1 1.5 0.2 setosa

## 5 5.0 3.6 1.4 0.2 setosa

## 6 5.4 3.9 1.7 0.4 setosa

plot(iris)

종(Species) 정보를 제외하고 꽃잎과 꽃받침의 길이와 두께 정보만을 이용하여 군집화를 수행해 보도록 한다. scale() 함수를 이용하여 변수의 값을 표준화하였다.

iris.data <- scale(iris[-5])

summary(iris.data)

## Sepal.Length Sepal.Width Petal.Length Petal.Width

## Min. :-1.8638 Min. :-2.426 Min. :-1.562 Min. :-1.442

## 1st Qu.:-0.8977 1st Qu.:-0.590 1st Qu.:-1.222 1st Qu.:-1.180

## Median :-0.0523 Median :-0.132 Median : 0.335 Median : 0.132

## Mean : 0.0000 Mean : 0.000 Mean : 0.000 Mean : 0.000

## 3rd Qu.: 0.6722 3rd Qu.: 0.557 3rd Qu.: 0.760 3rd Qu.: 0.788

## Max. : 2.4837 Max. : 3.080 Max. : 1.780 Max. : 1.706

**K-평균 군집화**

K-평 군집화를 하기 위해서는 stats 패키지에서 제공하는 kmeans() 함수를 이용해야 한다. 이 때 hclust의 첫째 인자는 요소는 데이터이고, 두째 인자는 간의 군집 수(k)여야 한다.

**K=3인 경우**

iris 데이터는 3개의 종으로 구성된 데이터이므로 군집의 수를 3개로 해서 군집화를 시도해 본다.

(iris.kmeans <- kmeans(iris.data, centers = 3, nstart = 5))

## K-means clustering with 3 clusters of sizes 50, 47, 53

##

## Cluster means:

## Sepal.Length Sepal.Width Petal.Length Petal.Width

## 1 -1.01119 0.85041 -1.3006 -1.2507

## 2 1.13218 0.08813 0.9928 1.0141

## 3 -0.05005 -0.88043 0.3466 0.2806

##

## Clustering vector:

## [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

## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 3 3 2 3 3 3 3 3 3 3 3 2 3 3 3 3

## [71] 2 3 3 3 3 2 2 2 3 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2

## [106] 2 3 2 2 2 2 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 2 2 2 2 2 2 3 3 2 2 2 3 2

## [141] 2 2 3 2 2 2 3 2 2 3

##

## Within cluster sum of squares by cluster:

## [1] 47.35 47.45 44.09

## (between\_SS / total\_SS = 76.7 %)

##

## Available components:

##

## [1] "cluster" "centers" "totss" "withinss"

## [5] "tot.withinss" "betweenss" "size" "iter"

## [9] "ifault"

결과에서 47, 50, 53개의 데이터로 구성된 3개의 군집이 만들어졌음을 볼 수 있다. kmeans() 함수의 결과로 cluster, centers 등 다양한 요소가 생성되었음을 볼 수 있다. cluster는 데이터 각 요소가 어떤 군집으로 분류되었는지에 대한 정보를 준다.

iris.kmeans$cluster

## [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

## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 3 3 2 3 3 3 3 3 3 3 3 2 3 3 3 3

## [71] 2 3 3 3 3 2 2 2 3 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 2 2 2

## [106] 2 3 2 2 2 2 2 2 3 3 2 2 2 2 3 2 3 2 3 2 2 3 2 2 2 2 2 2 3 3 2 2 2 3 2

## [141] 2 2 3 2 2 2 3 2 2 3

iris.kmeans$iter

## [1] 3

이 정보를 이용하여 plot() 함수를 이용하면 iris 데이터와 군집간의 관계를 살펴보자.

plot(iris[-5], pch = iris.kmeans$cluster, col = iris.kmeans$cluster)

세개로 나누어진 군집과 종의 교차표는 다음과 같다. 결과에서 알 수 잇듯이 setosa는 하나의 군집 잘 분리되나 versicolor와 virginica가 두개의 군집으로 섞여서 분리되었음을 볼 수 있다.

table(iris.kmeans$cluster, iris$Species)

##

## setosa versicolor virginica

## 1 50 0 0

## 2 0 11 36

## 3 0 39 14

**k=4인 경우**

(iris.kmeans <- kmeans(iris.data, centers = 4, nstart = 5))

## K-means clustering with 4 clusters of sizes 25, 47, 25, 53

##

## Cluster means:

## Sepal.Length Sepal.Width Petal.Length Petal.Width

## 1 -1.30344 0.19884 -1.3040 -1.2848

## 2 1.13218 0.08813 0.9928 1.0141

## 3 -0.71894 1.50199 -1.2972 -1.2166

## 4 -0.05005 -0.88043 0.3466 0.2806

##

## Clustering vector:

## [1] 3 1 1 1 3 3 1 1 1 1 3 1 1 1 3 3 3 3 3 3 3 3 3 1 1 1 1 3 3 1 1 3 3 3 1

## [36] 1 3 3 1 1 3 1 1 3 3 1 3 1 3 1 2 2 2 4 4 4 2 4 4 4 4 4 4 4 4 2 4 4 4 4

## [71] 2 4 4 4 4 2 2 2 4 4 4 4 4 4 4 2 2 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 2 2 2

## [106] 2 4 2 2 2 2 2 2 4 4 2 2 2 2 4 2 4 2 4 2 2 4 2 2 2 2 2 2 4 4 2 2 2 4 2

## [141] 2 2 4 2 2 2 4 2 2 4

##

## Within cluster sum of squares by cluster:

## [1] 9.646 47.450 12.148 44.088

## (between\_SS / total\_SS = 81.0 %)

##

## Available components:

##

## [1] "cluster" "centers" "totss" "withinss"

## [5] "tot.withinss" "betweenss" "size" "iter"

## [9] "ifault"

plot(iris[-5], pch = iris.kmeans$cluster, col = iris.kmeans$cluster)

table(iris.kmeans$cluster, iris$Species)

##

## setosa versicolor virginica

## 1 25 0 0

## 2 0 11 36

## 3 25 0 0

## 4 0 39 14

결과에서 보듯이 군집간의 거리의 제곱의 합(between\_SS)의 전체 거리의 제곱합(total\_SS)에서 81%를 차지함을 알 수 있다. k=3인 경우 76.7 %였으므로 큰 향상은 없었음을 볼 수 있다.

지금 예처럼 k에 따라 군집의 결과가 차이를 보일 수 있으므로 여러 숫자를 적용해 본 후 최적의 군집화 결과를 도출하는 것이 좋다.

IRIS 신경망 예제에서 에러발생 문제

It’s that moment we all know and love, somewhere in our code something has gone wrong. We think we have done everything right, but instead of expected glory we find only terse red text lain below our lintel.

This can be very frustrating, and trouble shooting these issues can often be very time consuming.

All is not lost. There are a few bits of R that can greatly help finding out what exactly has gone wrong and where, which in turn should suggest a reasonable course of action.

First we will look at some simple methods we can use to track down issues, namely the warn option and traceback, and then we will look at stepping through functions with debug.

First Steps

I’m going to use an example that has stuck with me from when I first started using R, using neural nets for classification with the iris data.

Let’s take a look at the error:

library(nnet)

**X <- iris[,1:4]**

**Y <- iris[,5]**

mod <- nnet(X, Y, size=2)

# weights: 19

Error in nnet.default(X, Y, size = 2) :

  NA/NaN/Inf in foreign function call (arg 2)

In addition: Warning message:

In nnet.default(X, Y, size = 2) : NAs introduced by coercion

Urgh. Really? What the hell does that mean?

We can use the built in traceback() function to see where this error occured

**> traceback()**

2: nnet.default(X, Y, size = 2)

1: nnet(X, Y, size = 2)

>

We can see our call to nnet(), which in turn has called nnet.default() and which is where our error has come from.

In the error output, we can see there was also a warning “NAs introduced by coercion”. As we weren’t expecting any warnings, let’s track down that, as errors tend to compound.

Warnings

To find out where that message was coming from, we will use options(warn = 2) which will turn warning messages into errors. We can do this by setting the warn option to a specific level, in this case 2.

The default is warn = 0, which means warnings will be stored until the top level function returns. We could use warn = 1 which wil print the warning as it is encountered, but in this case we want to stop straight away, so will set it to 2.

options(warn = 2)

Let’s try again:

> mod <- nnet(X, Y, size=2)

# weights: 19

Error in nnet.default(X, Y, size = 2) :

  (converted from warning) NAs introduced by coercion

>

Hmm, still coming from nnet.default, let’s see if traceback() is offering any new information.

> traceback()

6: doWithOneRestart(return(expr), restart)

5: withOneRestart(expr, restarts[[1L]])

4: withRestarts({

       .Internal(.signalCondition(simpleWarning(msg, call), msg,

           call))

       .Internal(.dfltWarn(msg, call))

   }, muffleWarning = function() NULL)

3: .signalSimpleWarning(“NAs introduced by coercion”, quote(nnet.default(X,

       Y, size = 2)))

2: nnet.default(X, Y, size = 2)

1: nnet(X, Y, size = 2)

>

We see a whole bunch of extra stuff in the traceback(), but from 3 onwards it appears to be sideffects of having set warn = 2. We do however see our warning has come from nnet.default again, so we will dig into that to see if we can find out what is going on.

Getting dirty with debug()

To do this, we can use the debug function. We will turn on debugging for nnet, which will let us step through the code line by line as it is executed.

> debug(nnet)

> mod <- nnet(X, Y, size=2)

debugging in: nnet(X, Y, size = 2)

debug: UseMethod(“nnet”)

Browse[2]>

The Browse> prompt tells us we are in the debugger. The debug: UseMethod(“nnet”) tells us the next line of code to be executed is UseMethod(“nnet”). We could enter ‘n’ here to continue to the next line, however a convenient default is just hitting enter (i.e. an empty line).

Browse[2]>

debugging in: nnet.default(X, Y, size = 2)

debug: {

    net <- NULL

…

    class(net) <- “nnet”

    net

}

Browse[3]>

Here R has printed out the R source for the function we just entered. We can see we are still at our Browse> prompt, so will continue on by hitting enter again and again:

Browse[3]>

debug: net <- NULL

Browse[3]>

debug: x <- as.matrix(x)

Browse[3]>

debug: y <- as.matrix(y)

Browse[3]>

debug: if (any(is.na(x))) stop(“missing values in ‘x'”)

Browse[3]>

debug: NULL

Browse[3]>

Well this is neat. Hitting enter at the prompt, R shows us each line that is about to be executed. We will continue on hitting enter until we see our error message.

Browse[3]>

…

Browse[3]>

debug: if (length(weights) != ntr || any(weights < 0)) stop(“invalid weights vector”)

Browse[3]>

debug: NULL

Browse[3]>

One thing of interest, we can see a conditional if statement is about to be run. When the conditional is evaluated as false, meaning the conditioning code won’t be executed, we will see the debug: NULL printed out.

Browse[3]>

debug: Z <- as.double(cbind(x, y))

Browse[3]>

Error in nnet.default(X, Y, size = 2) :

  (converted from warning) NAs introduced by coercion

>

Well, there is our warning message. Unfortunately we have lost the Browse> prompt, meaning we are no longer inside the function being debugged, but back at the main prompt.

This is a side effect of our rather aggressive options(“warn”) setting. Let’s tone it down a bit and set it to 1 so the warnings will be printed as they occur, then jump back into debugging.

> options(warn=1)

> mod <- nnet(X, Y, size=2)

debugging in: nnet(X, Y, size = 2)

debug: UseMethod(“nnet”)

Browse[2]>

debugging in: nnet.default(X, Y, size = 2)

debug: {

    net <- NULL

…

Browse[3]>

…

Browse[3]>

debug: Z <- as.double(cbind(x, y))

Browse[3]>

Warning in nnet.default(X, Y, size = 2) : NAs introduced by coercion

debug: storage.mode(weights) <- “double”

Browse[3]>

After some time, we get our warning message, and we are still in our debugged function. The warning is coming from the line

Z <- as.double(cbind(x, y))

What could be the problem here? Something is going wrong when nnet is converting x and y to doubles. Let’s take a look at them to see if there is anything going on.

We can do this by using get(“variable”), where variable is the quoted name of the variable. First let’s take a look at other two variables in the function to see how it works:

Browse[3]> get(“nout”)

[1] 1

Browse[3]> get(“ntr”)

[1] 150

Browse[3]>

Looking back at the debug output of each line, we can see they were set as some of the dimension info for X and Y, nout has a value of 1 and ntr has a value of 150.

Let’s take a look at x and y now:

Browse[3]> get(“x”)

       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

…

Browse[3]>

This is our input X data.

Browse[3]> get(“y”)

       [,1]

  [1,] “setosa”

  [2,] “setosa”

  [3,] “setosa”

  [4,] “setosa”

…

Browse[3]>

And here is our Y data, just as we passed in.

Recall the warning was triggered by this line:

Z <- as.double(cbind(x, y))

Which is converting x and y to the numeric double type.

We can see that x is numeric data, while y is character data. What happens if you convert character data to numeric? It doesn’t seem to make sense, but let’s try:

Browse[3]> as.numeric(get(“y”))

Warning: NAs introduced by coercion

  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA

 [37] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA

 [73] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA

[109] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA

[145] NA NA NA NA NA NA

Browse[3]>

A-ha!

This is just the warning we see, and thinking about it, we can understand why converting strings to numeric data is probably not going to be particularly meaningful.

Let’s get out of the debugger and think about what might be going on. Hit ‘c’ to continue execution

Browse[3]> c

Error in nnet.default(X, Y, size = 2) :

  NA/NaN/Inf in foreign function call (arg 2)

>

We are now back at the main prompt.

Now what??

What is going on? It seems reasonable that we should be able to pass a factor for classification, in fact we are pretty sure that’s what we saw being used in the examples in the nnet package documentation.

A careful reading of help(nnet) reveals some details. In particular:

If the response in formula is a factor, an appropriate classification network is constructed … If the response is not a factor, it is passed on unchanged to nnet.default.

It is possible to pass a factor in Y, but we must use the formula syntax. Looking at the examples, we see the matrix syntax in use as well, however it is transforming the y values using class.ind().

Right about now I contemplate calling the police, as I’m pretty sure someone has snuck in and changed the docs while I wasn’t looking. There is no way I would make such a simple mistake …

Anyway, let’s turn off debugging, and see if we can get this working:

> undebug(nnet)

**> mod <- nnet(X, class.ind(Y), size=2)**

**> mod <- nnet(Y~., data=cbind(X, Y), size=2)**

Both methods run without error (or warning). Success at last.