# Multivariate Analysis with R

## Linear Discriminant Analysis with R

Shalabh
Department of Mathematics and Statistics
Indian Institute of Technology Kanpur

# Discriminant Analysis: Classification rule

LDA was developed by Sir R.A. Fisher in 1936 to classify subjects into one of the two clearly defined groups. So, the discriminant function is also called Fisher's discriminant function.

Assuming *x* to have equal probability and equal cost to be classified between $N(\underline{\mu}_1, \Sigma)$ and $N(\underline{\mu}_2, \Sigma)$, the classification rule is

$$R_1: \underline{x}'\Sigma^{-1}\left(\underline{\mu}_1 - \underline{\mu}_2\right) - \frac{1}{2}\left(\underline{\mu}_1 + \underline{\mu}_2\right)'\Sigma^{-1}\left(\underline{\mu}_1 - \underline{\mu}_2\right) \geq 0$$

$$R_2: \underline{x}'\Sigma^{-1}\left(\underline{\mu}_1 - \underline{\mu}_2\right) - \frac{1}{2}\left(\underline{\mu}_1 + \underline{\mu}_2\right)'\Sigma^{-1}\left(\underline{\mu}_1 - \underline{\mu}_2\right) < 0$$

The quantities $\underline{\mu}_1$, $\underline{\mu}_2$ and $\Sigma$ are unknown in practice. So the question is how to implement $R_1$ and $R_2$ in a data set.

# Discriminant Analysis: Classification rule

Since the parameters $\underline{\mu}_1$ , $\underline{\mu}_2$ and $\Sigma$ are unknown in practice, so we estimate them using the sample observations and replace the estimated values in place of unknown parameters.

Obtain a sample $x_1^{(1)}, x_2^{(1)}, \ldots, x_{n_1}^{(1)}$ of size $n_1$ from $N(\underline{\mu}_1, \Sigma)$ .

Obtain a sample $x_1^{(2)}, x_2^{(2)}, \ldots, x_{n_2}^{(2)}$ of size $n_2$ from $N(\underline{\mu}_2, \Sigma)$ .

Find $\hat{\underline{\mu}}_1 = \underline{\bar{x}}^{(1)} = \frac{1}{N_1}\sum_{k=1}^{n_1} x_k^{(1)}$ , $\hat{\underline{\mu}}_2 = \underline{\bar{x}}^{(2)} = \frac{1}{N_2}\sum_{k=1}^{n_2} x_k^{(2)}$

$$\hat{\Sigma} = S = \frac{1}{n_1 + n_2 - 2}\left[\sum_{k=1}^{n_1}(\underline{x}_k^{(1)} - \underline{\bar{x}}^{(1)})(\underline{x}_k^{(1)} - \underline{\bar{x}}^{(1)})' + \sum_{k=1}^{n_2}(\underline{x}_k^{(2)} - \underline{\bar{x}}^{(2)})(\underline{x}_k^{(2)} - \underline{\bar{x}}^{(2)})'\right]$$

and replace them in place of $\underline{\mu}_1$ , $\underline{\mu}_2$ and $\Sigma$.

# Discriminant Analysis: Classification rule

**Then the classification rules become**

$$R_1 : \underline{x}' S^{-1}(\bar{\underline{x}}^{(1)} - \bar{\underline{x}}^{(2)}) - \frac{1}{2}(\bar{\underline{x}}^{(1)} + \bar{\underline{x}}^{(2)})' S^{-1}(\bar{\underline{x}}^{(1)} - \bar{\underline{x}}^{(2)}) \geq 0,$$

$$R_2 : \underline{x}' S^{-1}(\bar{\underline{x}}^{(1)} - \bar{\underline{x}}^{(2)}) - \frac{1}{2}(\bar{\underline{x}}^{(1)} + \bar{\underline{x}}^{(2)})' S^{-1}(\bar{\underline{x}}^{(1)} - \bar{\underline{x}}^{(2)}) < 0.$$

**Linear Discriminant Analysis in R takes a data set of cases (also known as observations) as input. For each case, you need to have a categorical variable to define the class.**

# Discriminant Analysis: R command

`lda` command is used for LDA in `MASS` package.

For a given problem, the number of discriminant functions will be equal to the smaller of the number of predictor variables or one less the number of groups on the dependent variables.

```
lda(formula, data, ...,)

lda(x, grouping, ..)
```

Arguments

`formula` A formula of the form `groups ~ x1 + x2 + ...`
The response is the grouping factor and the right hand side specifies the (non-factor) discriminators.

`x` a matrix, data frame or Matrix containing explanatory variables.

# Discriminant Analysis: R command

**Arguments**

**`grouping`**: **a factor used to specify the classes of the observations**

**`data`** **An optional data frame, list or environment from which variables specified in `formula` are preferentially to be taken.**

**`prior`** **the prior probabilities of class membership. If unspecified, the class proportions for the training set are used. If present, the probabilities should be specified in the order of the factor levels.**

**`CV`** **If true, returns results (classes and posterior probabilities) for leave-one-out cross-validation. Note that if the prior is estimated, the proportions in the whole dataset are used.**

# Discriminant Analysis: R command

**Details**

**`prior`** **the prior probabilities used.**

**`means`** **the group means.**

**`scaling`** **a matrix which transforms observations to discriminant functions, normalized so that within groups covariance matrix is spherical.**

**`N`** **The number of observations used.**

# Discriminant Analysis: Example (Univariate, Bivariate, and Multivariate Statistics Using R_ Quantitative Tools for Data Analysis and Data Science-Wiley (2020)-Daniel J. Denis)

**Consider the following data set on quantitative ability, verbal ability, and a training group designating whether individuals received no training (1), some (2), or much training (3) in learning these skills.**

| Subject | Quantitative | Verbal | Training |
|---|---|---|---|
| 1 | 5 | 2 | 1 |
| 2 | 2 | 1 | 1 |
| 3 | 6 | 3 | 1 |
| 4 | 9 | 7 | 2 |
| 5 | 8 | 9 | 2 |
| 6 | 7 | 8 | 2 |
| 7 | 9 | 8 | 3 |
| 8 | 10 | 10 | 3 |
| 9 | 10 | 9 | 3 |

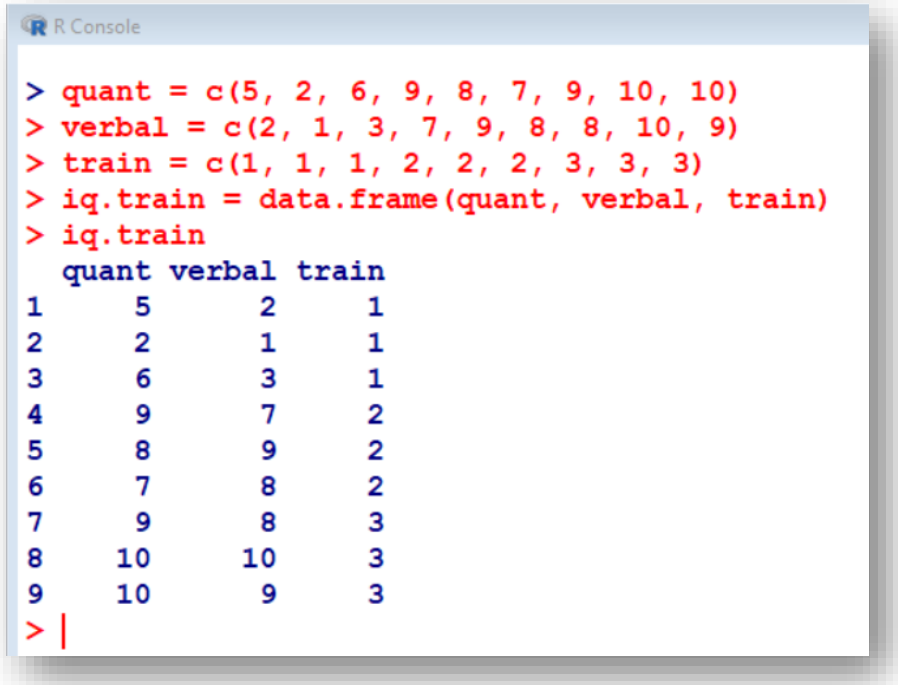
**1 = no training, 2 = some training, 3 = extensive training.**

# Discriminant Analysis: Example

Create a data frame

```
quant = c(5, 2, 6, 9, 8, 7, 9, 10, 10)

verbal = c(2, 1, 3, 7, 9, 8, 8, 10, 9)

train = c(1, 1, 1, 2, 2, 2, 3, 3, 3)

iq.train = data.frame(quant, verbal, train)
iq.train
```

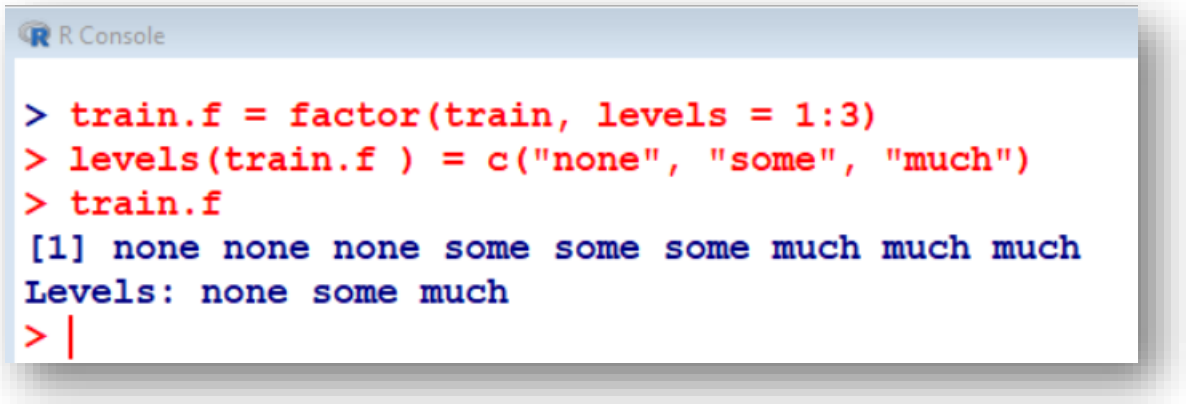
```
  quant verbal train
1     5      2     1
2     2      1     1
3     6      3     1
4     9      7     2
5     8      9     2
6     7      8     2
7     9      8     3
8    10     10     3
9    10      9     3
```



# Discriminant Analysis: Example

**Rename the categories**

```
train.f = factor(train, levels = 1:3)

levels(train.f ) = c("none", "some", "much")

train.f
[1] none none none some some some much much much
Levels: none some much
```



# Discriminant Analysis: Example

**Conduct discriminant analysis**

```
library(MASS)

lda.fit = lda(train.f ~ verbal + quant,

data=iq.train)


lda.fit


Call:
lda(train.f ~ verbal + quant, data = iq.train)
```

(CONTD…)

# Discriminant Analysis: Example

## Conduct discriminant analysis

R Console

```
> library(MASS)
> lda.fit = lda(train.f ~ verbal + quant, data=iq.train)
> lda.fit
Call:
lda(train.f ~ verbal + quant, data = iq.train)

Prior probabilities of groups:
     none      some      much
0.3333333 0.3333333 0.3333333

Group means:
     verbal    quant
none      2 4.333333
some      8 8.000000
much      9 9.666667

Coefficients of linear discriminants:
               LD1        LD2
verbal 0.97946790 -0.5901991
quant  0.02983363  0.8315153

Proportion of trace:
   LD1    LD2
0.9889 0.0111
> |
```

(CONTD…)

# Discriminant Analysis: Example

**Conduct discriminant analysis**

```
Prior probabilities of groups:
     none      some      much
0.3333333 0.3333333 0.3333333
```

(CONTD…)

- The `Prior probabilities of groups`: is what we would expect the probability of classification to be per group in the absence of any predictors.
- In R, this probability is set by default for our data at 0.33 per group, indicating equal priors per group.
- This prior probability can be adjusted if we had good reason to, but for now, we will leave it at default.

# Discriminant Analysis: Example

```
lda.fit
Group means:
      verbal    quant
none       2 4.333333
some       8 8.000000
much       9 9.666667
```

(CONTD…)

- The `Group means`: represents the mean for each level of `verbal` and each level of `quant`.
- The mean for `verbal = none` is 2,
- the mean for `verbal = some` is 8, and
- the mean for `verbal = much` is 9.
- The mean for `quant = none` is 4.33,
- the mean for `quant = some` is 8.00, and
- the mean for `quant = much` is 9.67.

# Discriminant Analysis: Example

```
Coefficients of linear discriminants:
               LD1         LD2
verbal 0.97946790 -0.5901991
quant  0.02983363  0.8315153
```

(CONTD…)

- The `Coefficients of linear discriminants`: are the estimated coefficients for each linear discriminant function.
- R has produced two discriminant functions here.
- The first discriminant function (LD1) weights `verbal` by 0.97946790 and `quant` by 0.02983363.
- The second discriminant function (LD2) weights `verbal` by –0.5901991 and `quant` by 0.8315153.

# Discriminant Analysis: Example

```
Coefficients of linear discriminants:
                LD1         LD2
verbal 0.97946790 -0.5901991
quant  0.02983363  0.8315153
```

(CONTD…)

- The first discriminant function (LD1) is

> LD1 = 0.97946790 * verbal + 0.02983363 * quant

- The second discriminant function (LD2) is

> LD2 = –0.5901991 * verbal + 0.8315153 * quant

- The coefficients generated here are the raw coefficients for the discriminant functions. Standardized coefficients are also possible using `lda.fit$scaling`.

# Discriminant Analysis: Example

```
Proportion of trace:
   LD1    LD2
0.9889 0.0111
```

(CONTD…)

- The `Proportion of trace`: values for `LD1` of 0.9889 and `LD2` of 0.0111 can be considered as measures of importance for each function.
- Higher value of LD indicates more importance and preference.
- The relevance of each discriminant function can be computed by contrasting its eigenvalue to the sum of eigenvalues generated by the entire discriminant analysis.
- R doesn't report eigenvalues, but they are used "behind the scenes" for obtaining the proportion of trace output.

# Discriminant Analysis: Example

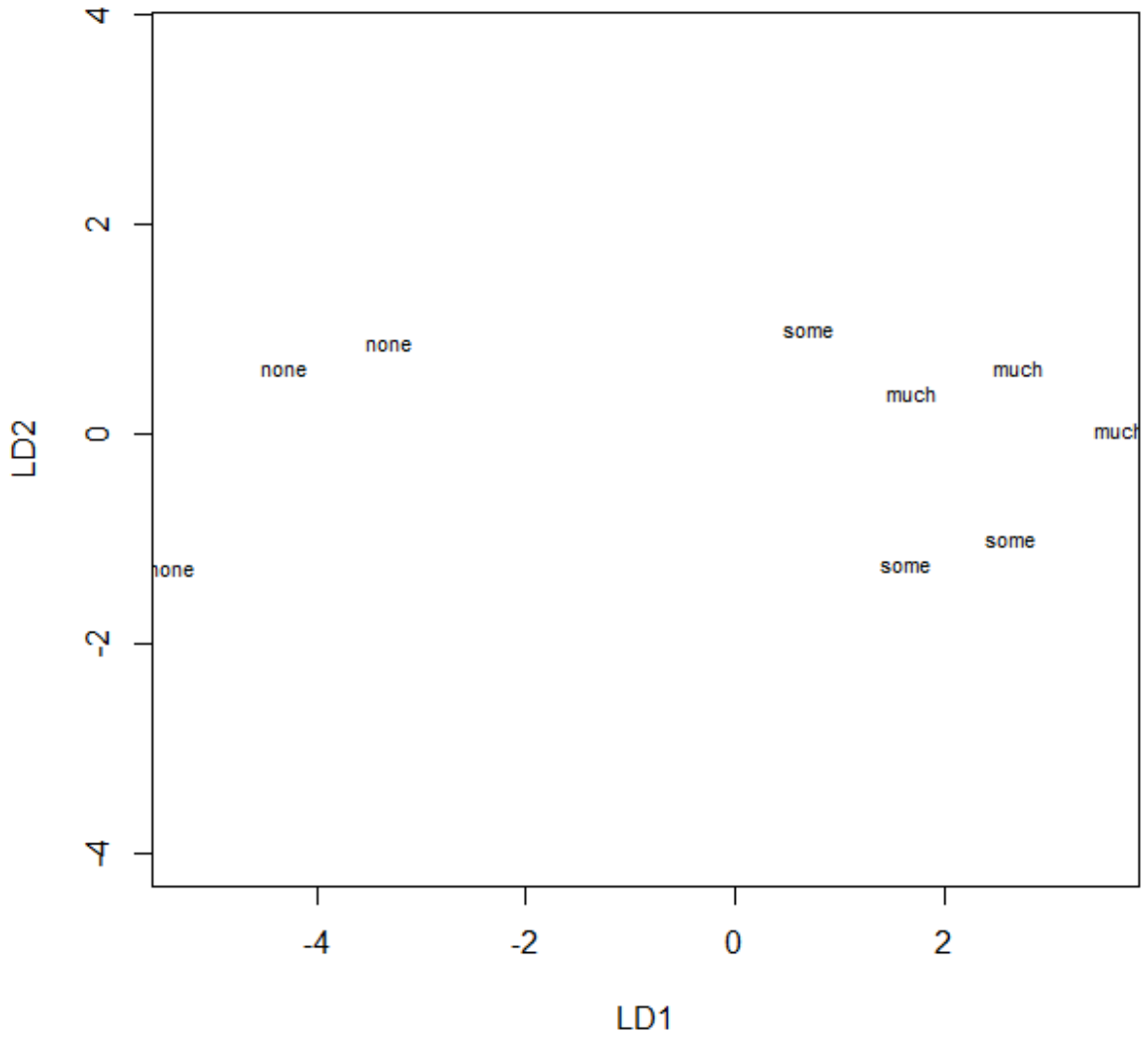
```
Proportion of trace:
   LD1    LD2
0.9889 0.0111
```

(CONTD…)

- For our data, the first eigenvalue turns out to be 14.35 and the second eigenvalue is equal to 0.16. These two eigenvalues, when summed, make up the trace of the matrix.
- Therefore, the proportion of the trace accounted for by the first eigenvalue is 14.35/(14.35 + 0.16) = 14.35/14.51 = 0.989, while the proportion of the trace accounted for by the second eigenvalue is 0.16/14.51 = 0.01.
- Clearly, the first discriminant function is much more relevant in discriminating between groups than the second.

# Discriminant Analysis: Example

```
plot(lda.fit)
```



# Discriminant Analysis: Example- Predicting Group Membership:

Having obtained our discriminant functions, we predict group membership, and in this way, we will be able to validate the goodness of our functions using the command

```
predict(lda.fit)
```

We first find predicted probabilities for each group of the dependent variable based on discriminant functions derived

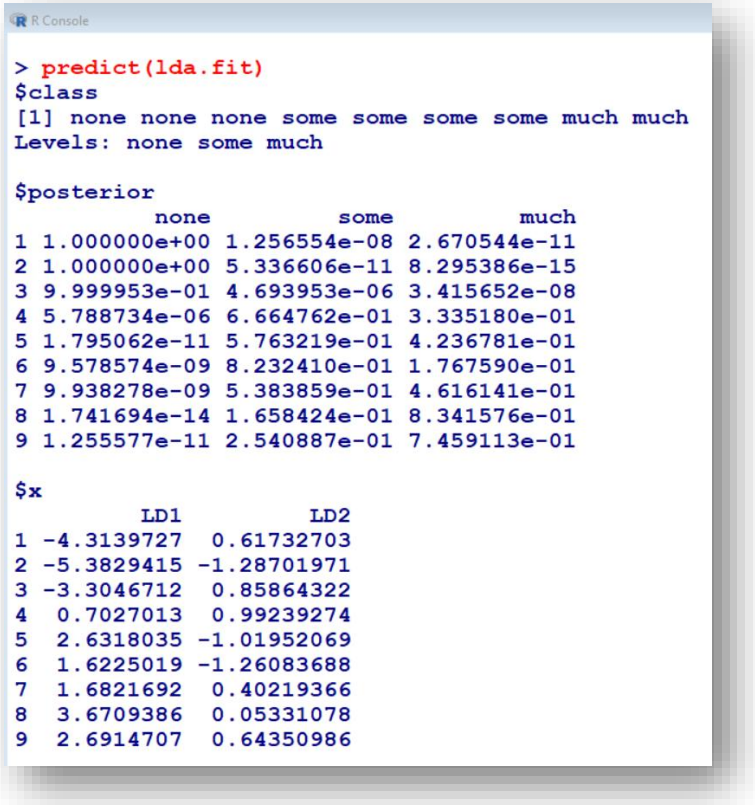
# Discriminant Analysis: Example- Predicting Group Membership:

R Console

```
> predict(lda.fit)
$class
[1] none none none some some some some much much
Levels: none some much

$posterior
           none          some          much
1 1.000000e+00 1.256554e-08 2.670544e-11
2 1.000000e+00 5.336606e-11 8.295386e-15
3 9.999953e-01 4.693953e-06 3.415652e-08
4 5.788734e-06 6.664762e-01 3.335180e-01
5 1.795062e-11 5.763219e-01 4.236781e-01
6 9.578574e-09 8.232410e-01 1.767590e-01
7 9.938278e-09 5.383859e-01 4.616141e-01
8 1.741694e-14 1.658424e-01 8.341576e-01
9 1.255577e-11 2.540887e-01 7.459113e-01

$x
         LD1          LD2
1 -4.3139727  0.61732703
2 -5.3829415 -1.28701971
3 -3.3046712  0.85864322
4  0.7027013  0.99239274
5  2.6318035 -1.01952069
6  1.6225019 -1.26083688
7  1.6821692  0.40219366
8  3.6709386  0.05331078
9  2.6914707  0.64350986
```

# Discriminant Analysis: Example- Predicting Group Membership:

```
> predict(lda.fit)
$class
[1] none none none some some some some much much
Levels: none some much

$posterior
           none          some          much
1 1.000000e+00 1.256554e-08 2.670544e-11
2 1.000000e+00 5.336606e-11 8.295386e-15
3 9.999953e-01 4.693953e-06 3.415652e-08
4 5.788734e-06 6.664762e-01 3.335180e-01
5 1.795062e-11 5.763219e-01 4.236781e-01
6 9.578574e-09 8.232410e-01 1.767590e-01
7 9.938278e-09 5.383859e-01 4.616141e-01
8 1.741694e-14 1.658424e-01 8.341576e-01
9 1.255577e-11 2.540887e-01 7.459113e-01

$x
         LD1         LD2
1 -4.3139727  0.61732703
2 -5.3829415 -1.28701971
3 -3.3046712  0.85864322
4  0.7027013  0.99239274
5  2.6318035 -1.01952069
6  1.6225019 -1.26083688
7  1.6821692  0.40219366
8  3.6709386  0.05331078
9  2.6914707  0.64350986
```



# Discriminant Analysis: Example- Predicting Group Membership:

The `$posterior` title indicates the posterior probabilities associated with prediction into each of the three groups on the response variable.

For the first case, we see the posterior probability is near 1.0 for the group "`none`," while it is much less for the other two groups.

The probability of classification into the group "`some`" for that first case is exceedingly low.

Likewise for the third group "`much`."

# Discriminant Analysis: Example- Predicting Group Membership:

- We can see that when classification is based on the highest probability,
  - ✓ cases 1 through 3 are classified into the first group "none," ,
  - ✓ cases 4 through 7 are classified into the second group "some,"
  - ✓ cases 8 and 9 are classified into the third group "much".

- It is clear by inspecting the classification results that the discriminant analysis did not result in perfect prediction. Otherwise, we would have expected 3 cases per group.

# Discriminant Analysis: Example:

## How Well Did the Discriminant Function Analysis Do?

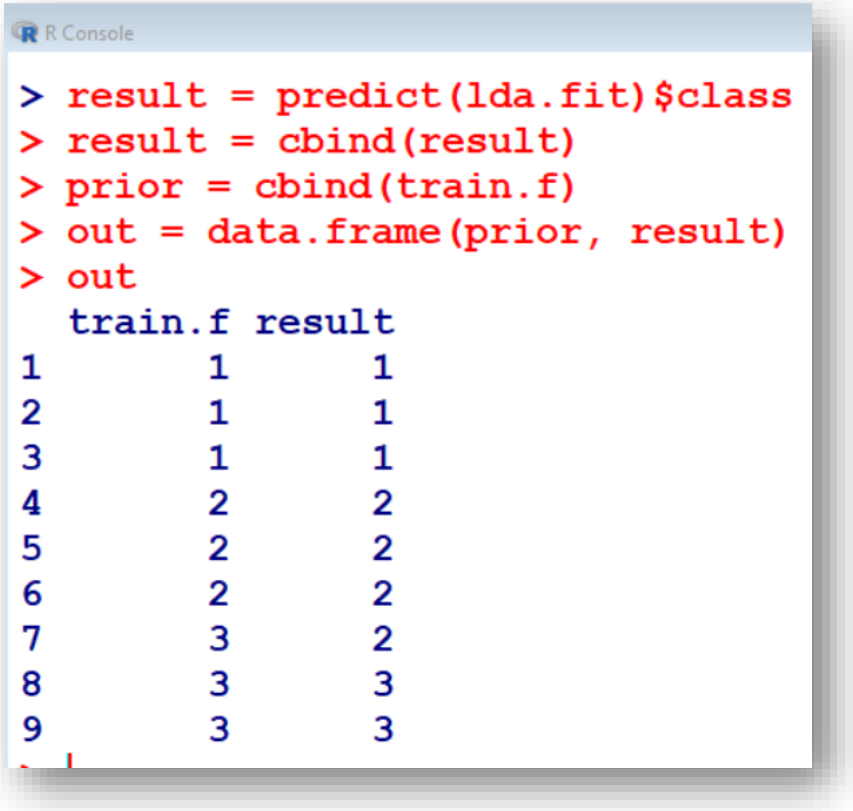
- We have discriminant functions that did their best to maximize group discrimination.
- Because it maximized group separation does not necessarily mean it did it well.
- After performing the LDA, the next question is to evaluate how well the classifier did its job.
- We already know based on the above probabilities of group membership that it did not do a perfect job.
- We'd like a summary of how well it did.

# Discriminant Analysis: Example

**We can obtain predicted group membership by**

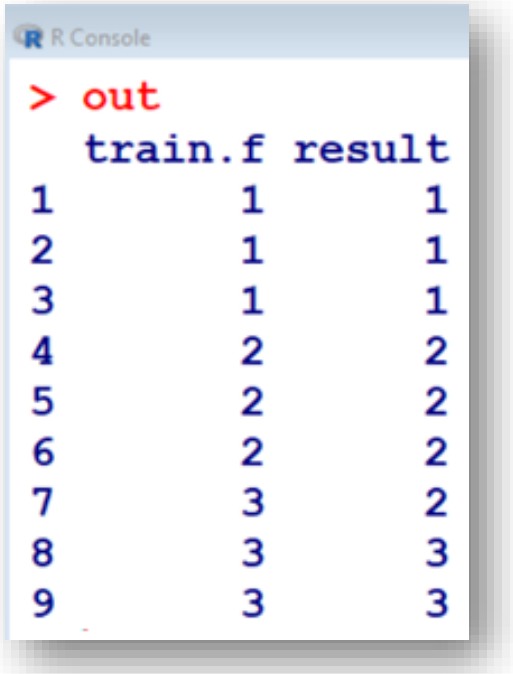
```
result = predict(lda.fit)$class
result = cbind(result)
prior = cbind(train.f)
out = data.frame(prior, result)
out
```

```
  train.f result
1       1      1
2       1      1
3       1      1
4       2      2
5       2      2
6       2      2
7       3      2
8       3      3
9       3      3
```



# Discriminant Analysis: Example

- `train.f` contains the actual membership group, that is, the natural group membership.
- `result` contains the predicted group membership based on the previously observed probabilities.



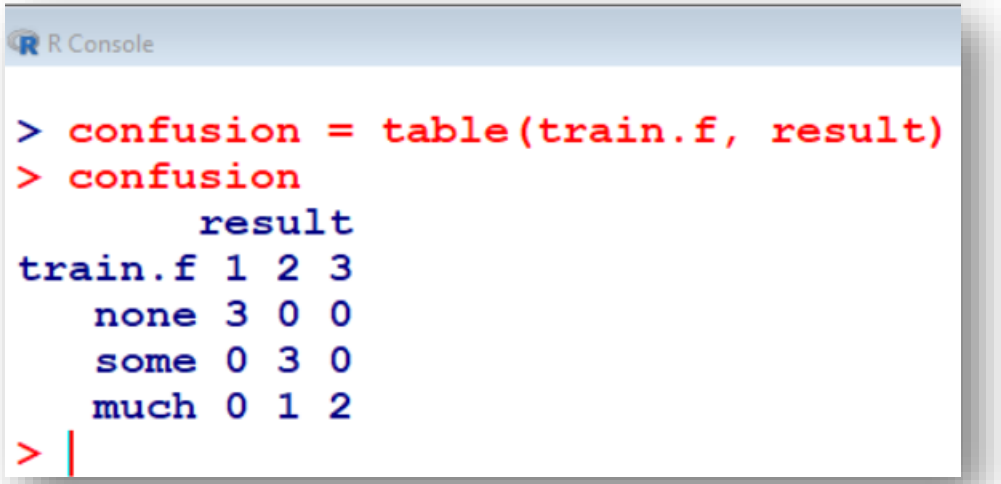

- As we noted, the LDA got the first three cases correct, and also the following three cases, but then misclassified the seventh case. It was supposed to go into the third group ("`much`") but was classified into the second group instead.
- This constitutes an error in classification or a misclassified case.

# Discriminant Analysis: Example

**It is better to obtain a summary of how well the LDA performed.**

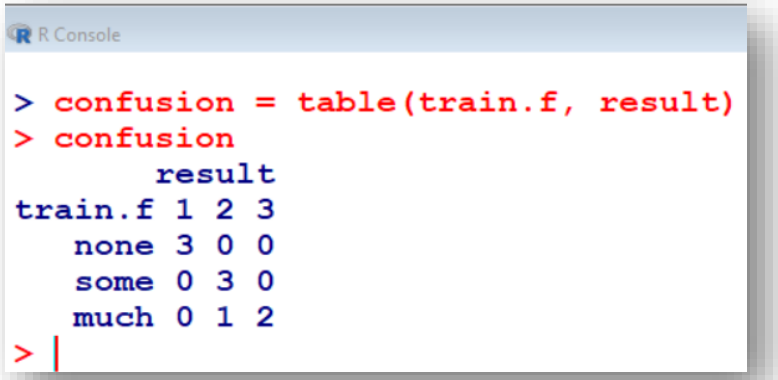
**We can obtain this in R by computing a confusion matrix that depicts the results of the classification:**

```
confusion = table(train.f, result)
confusion
        result
train.f 1 2 3
   none 3 0 0
   some 0 3 0
   much 0 1 2
```



# Discriminant Analysis: Example

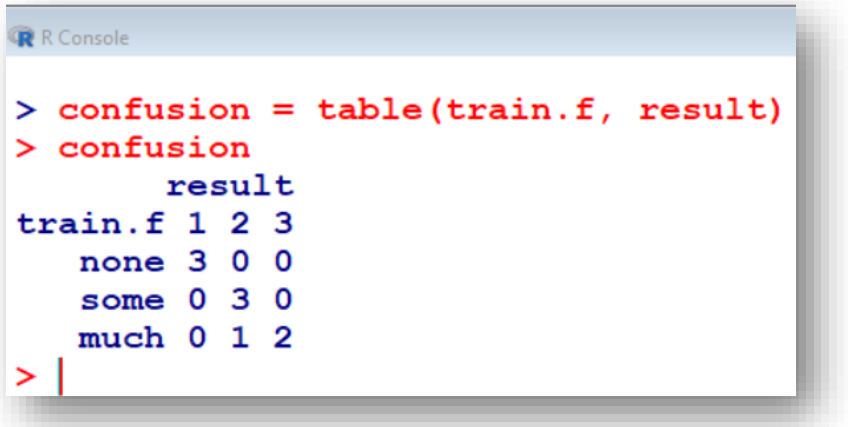
- Of those originally in the none group on `train.f`, all were correctly classified into the first group on the result, which corresponds also to "`none`." That is, the LDA got those first 3 cases correct in terms of classification.




- Of those originally in the some group on `train.f`, all were correctly classified into the second group on the result, which corresponds also to "`some`." That is, the LDA got those 3 cases correct as well in terms of classification.

# Discriminant Analysis: Example

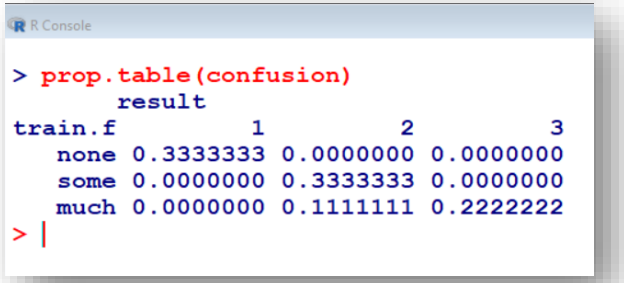
- However, of those originally in the much group on `train.f`, only 2 were classified correctly. One was misclassified into the some group on the result (i.e., group 2 on the result). This is the error in classification we noticed earlier.




- A perfect LDA is that the diagonal of the matrix would contain all the numbers, and no numbers would appear in the off-diagonal. Hence, perfect classification would have resulted in 3's across the main diagonal.

# Discriminant Analysis: Example

We can also produce the relevant proportions in a 3 × 3 table as follows:




```
prop.table(confusion)
        result
train.f         1         2         3
   none 0.3333333 0.0000000 0.0000000
   some 0.0000000 0.3333333 0.0000000
   much 0.0000000 0.1111111 0.2222222
```

In addition to seeing the proportions along the main diagonal of 0.33, 0.33, and 0.22, we also see the proportion of 0.11 in the cell in row 3, column 2. That's the 1 out of 9 cases (i.e. 1/9 = 0.11) that was misclassified.