- The matrix $A$ there is our $W$ here, known.
- The vector $D$ there is our $A_{\cdot j}$ here, known.
- The vector $b$ there is our $H_{\cdot j}$ here, unknown and to be solved for.

So we compute

```
> h[,j] <- lm(a[,j] ~ w - 1)$coef
```

for each $j$.[7]

On the other hand, suppose we know $H$ but not $W$. We could take transposes,

$$A' = H'W' \tag{5.12}$$

and then just interchange the roles of $W$ and $H$ above. Here a call to **lm()** gives us a column of $W'$, thus a row of $W$, and we do this for all rows.

Putting all this together, we first choose initial guesses, say random numbers, for $W$ and $H$; **nmf()** gives us various choices as to how to do this. Then we alternate: Compute the new guess for $W$ assuming $H$ is correct, then choose the new guess for $H$ based on that new $W$, and so on.

During the above process, we may generate some negative values. If so, we simply truncate to 0.

### 5.9.3 Back to Recommender Systems: Dealing with the Missing Values

In our recommender systems setting, of course, most of $A$ is missing. But we can easily adapt to that. Roughly speaking, in (5.11), do these replacements:

- replace $A_{\cdot j}$ by the known portion of $A_{\cdot j}$
- replace $W$ by the corresponding rows of $W$

Then proceed as before.

Here is a little example. Say $A$ ix $5 \times 5$ and we want rank 3. Then $W$ and $H$ are of sizes $5 \times 3$ and $3 \times 5$.

Note too that $(WH)_{\cdot j}$, thus column $j$ of our approximation to $A$, is a linear combination of the columns of $W$, with coefficients being $H_{\cdot j}$.

[7] The -1 specifies that we do not want a constant term in the model.

Suppose

$$A_{.4} = \begin{pmatrix} NA \\ 3 \\ NA \\ 8 \\ 2 \end{pmatrix} \tag{5.13}$$

Then in (5.11) we replace $A_{.5}$ by

$$\begin{pmatrix} 3 \\ 8 \\ 2 \end{pmatrix} \tag{5.14}$$

Also, replace $W$ by

$$\begin{pmatrix} w_{21} & w_{22} & w_{23} \\ w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \end{pmatrix} \tag{5.15}$$

Remember, at this stage, $W$ is assumed known. So, we just use **lm()**, "predicting" (5.14) from (5.15) to find $h_{.4}$.

### 5.9.4 Convergence and Uniqueness Issues

There are no panaceas for applications considered here. Every solution has potential problems. I like to call this the Pillow Theorem — pound down on one fluffy part and another part pops up.

Unlike the PCA case, one issue with NMF is uniqueness — there might not be a unique pair $(W, H)$ that minimizes (5.8).[8] In fact, one can see this immediately: Doubling $W$ while having $H$ leaves the product $WH$ unchanged. Of course, the product is all that really counts, but in turn, this may result in convergence problems. The NMF documentation recommends running **nmf()** multiple times; it will use a different seed for the random initial values each time.

The Alternating Least Squares method used here is considered by some to have better convergence properties, since the solution at each iteration is unique. This may come at the expense of slower convergence.

[8] See Donoho and Stodden, *When Does Non-Negative Matrix Factorization Give a Correct Decomposition into Parts?*, `https://web.stanford.edu/~vcs/papers/NMFCDP.pdf`.