Matrix Decomposition with Kronecker Weights

Jan de Leeuw

April 23, 2026

The familiar orthogonal decomposition of a matrix into row effect, column effects, and interactions is generalized to a matrix space with the Kronecker inner product.

Table of contents

Note: This is a working manuscript which will be expanded/updated frequently. All suggestions for improvement are welcome. All Rmd, tex, html, pdf, R, and C files are in the public domain. Attribution will be appreciated, but is not required. The files can be found at https://github.com/deleeuw/decomp

1 Kronecker Weights

R and C are positive semidefinite matrices of order n and m. They define an inner product on the space $\mathbb{R}^{n\times m}$ of $n \times m$ matrices as $\langle X, Y\rangle_{RC} = \text{tr}\ RXCY'$ and a corresponding squared norm (pseudo-norm if either R or C is singular) as $\|X\|^2_{RC} = \langle X, X\rangle_{RC}$. From now on we leave out the subscript RC.

2 Matrix Decomposition

Define three subspaces of $\mathbb{R}^{n\times m}$. We use e for vectors with all elements equal to one. The number of elements in e will be clear from the context, otherwise we will use e_m and e_n.

1. $\mathbb{L}_\mu$, constant matrices, $X = \mu e_n e_m'$.
2. $\mathbb{L}_a$, matrices with all columns the same, $X = ae_m'$.
3. $\mathbb{L}_b$, matrices with all rows the same $X = e_n b'$.

We will hoose a, b, μ such that these three subspacs are orthogonal in the Kronecker inner product. For this we need

$$\text{tr}\ R\mu ee'Cea' = \mu a'Re \times e'Ce = 0, \tag{1a}$$
$$\text{tr}\ R\mu ee'Cbe' = \mu b'Ce \times e'Re, \tag{1b}$$
$$\text{tr}\ Rae'Cbe' = a'Re \times b'Ce. \tag{1c}$$

It follows the three subspaces are orthogonal if $a'Re = b'Ce = 0$. Thus we redefine our subspaces as

1. $\mathbb{L}_\mu$: all X with $X = \mu ee'$.
2. $\mathbb{L}_a$: all X with $X = ae'$ where $a'Re = 0$.
3. $\mathbb{L}_b$: all X with $X = eb'$ where $b'Ce = 0$.

Now project, in the Kronecker metric, matrix X on each of these orthogonal subspaces.

First on $\mathbb{L}_\mu$. Minimize over μ

$$\|X - \mu ee'\|^2 = \|X\|^2 - 2\mu e'RXCe + \mu^2 e'Re \times e'Ce, \tag{2}$$

and thus

$$\hat{\mu} = \frac{e'RXCe}{e'Re \times e'Ce}. \tag{3}$$

Now over $\mathbb{L}_a$. We have

$$\|X - ae'\|^2 = \|X\|^2 - 2a'RXCe + a'Ra \times e'Ce, \tag{4}$$

which must be minimized over all $a'Re = 0$. Use a Lagrange multiplier λ. Stationary equations are

$$RXCe - \lambda Re = e'Ce \times Ra. \tag{5}$$

Premultiply by e' gives

$$\hat{\lambda} = \frac{e'RXCe}{e'Re}, \tag{6}$$

and thus

$$e'Ce \times \hat{a} = XCe - e\frac{e'RXCe}{e'Re}, \tag{7}$$

or

$$\hat{a} = \left(I - \frac{ee'R}{e'Re}\right) X \left(\frac{Ce}{e'Ce}\right). \tag{8}$$

It is convenient to define the projectors

$$P := \frac{ee'R}{e'Re}, \tag{9}$$

$$Q := \frac{Cee'}{e'Ce}. \tag{10}$$

The projection on $\mathbb{L}_a$ is

$$\hat{a}e' = (I - P)XQ. \tag{11}$$

Projecting on $\mathbb{L}_b$ is similar, by symmetry. Thus

$$e\hat{b}' = PX(I - Q). \tag{12}$$

In addition, from (3),

$$\hat{\mu}ee' = PXQ. \tag{13}$$

It follows that if we define

$$\hat{H} := X - \hat{\mu}ee' - \hat{a}e' - e\hat{b}' \tag{14}$$

then

$$\hat{H} = (I - P)X(I - Q). \tag{15}$$

Define $\mathbb{L}_h$ as the subspace of all matrices of this form, or equivalently the matrices with $HCe = 0$ and $e'RH = 0$.

Because $(I-Q)Ce = 0$ it follows that

$$\text{tr } R\widehat{H}C(\hat{\mu}ee') = 0 \tag{16a}$$

and

$$\text{tr } R\widehat{H}C(e\hat{b}') = 0 \tag{16b}$$

and

$$\text{tr } R\hat{a}e'C\widehat{H}' = 0 \tag{16c}$$

Thus $\mathbb{L}_h$ is also orthogonal in the Kronecker inner product to the other three subspaces. It follows that in the Kronecker norm

$$\|X\|^2 = \|PXQ\|^2 + \|(I-P)XQ\|^2 + \|PX(I-Q)\|^2 + \|(I-P)X(I-Q)\|^2, \tag{17}$$

and thus we not only have a decomposition of the data X, but also of its Kronecker sum of squares.

It also follows that if $Y = \mu ee' + ae' + eb' + H$, with $a'Re = b'Ce = 0$, then

$$\|X-Y\|^2 = \|PXQ - \mu ee'\|^2 + \|(I-P)XQ - ae'\|^2 + \\ \|PX(I-Q) - eb'\|^2 + \|(I-P)X(I-Q) - H\|^2 \tag{18}$$

THus if we are fitting Y to X with seperable restrictions on a, b, and H (in addition to $a'Re = b'Ce = 0$) then we can minmize each of the four components separately.

We have defined H in (14) as the residual after projecting on the three orthogonal spaces. The orthogonality of the residual space is YAPT (Yet Another Pythogorean Theorem). Alternatively, we could have defined $\mathbb{L}_h$ as the space of $n \times m$ matrices satisfying $HCe = 0$ and $H'Re = 0$ and project on this space. This will produce the same $\widehat{H}$.

The developments in this paper can be generalized in several ways. In the first place the same reasoning applies to multiway arrays. If the array has p "dimensions" then there are 2^p components. Secondly, vectors e_n and e_m in the formulas can be replaced by any vectors u and v with n and m elements. In fact, with some additional notation, they can be replaced by matrices U and V. And finally the results apply to any finite sequence of orthogonal subspaces in any (finite-dimensional) inner product space.

3 Example

```r
set.seed(12345)
ymat <- matrix(rnorm(12), 4, 3)
rmat <- crossprod(matrix(rnorm(40), 10, 4)) / 10
cmat <- crossprod(matrix(rnorm(30), 10, 3)) / 10
```

```
[1] "ymat"
```

```
           [,1]       [,2]       [,3]
[1,]  0.5855288  0.6058875 -0.2841597
[2,]  0.7094660 -1.8179560 -0.9193220
[3,] -0.1093033  0.6300986 -0.1162478
[4,] -0.4534972 -0.2761841  1.8173120
```

```
[1] "cmat"
```

```
           [,1]       [,2]       [,3]
[1,]  1.3744581  0.7610956 -0.2692483
[2,]  0.7610956  1.2428719 -0.1845269
[3,] -0.2692483 -0.1845269  1.3504256
```

```
[1] "rmat"
```

```
          [,1]       [,2]       [,3]       [,4]
[1,] 0.6606540  0.6316750  0.2060831  0.4388240
[2,] 0.6316750  1.5141559 -0.7485539  0.5940988
[3,] 0.2060831 -0.7485539  2.0102613 -0.4442019
[4,] 0.4388240  0.5940988 -0.4442019  1.3318363
```

We now apply the decompose() function from the code section.

```
h<-decompose(ymat, rmat, cmat)
```

The four components are:

```
[[1]]
            [,1]        [,2]        [,3]
[1,] -0.01383449 -0.01383449 -0.01383449
[2,] -0.01383449 -0.01383449 -0.01383449
[3,] -0.01383449 -0.01383449 -0.01383449
[4,] -0.01383449 -0.01383449 -0.01383449

[[2]]
          [,1]       [,2]      [,3]
[1,] 0.2414392 -0.3254692 0.1578904
[2,] 0.2414392 -0.3254692 0.1578904
[3,] 0.2414392 -0.3254692 0.1578904
[4,] 0.2414392 -0.3254692 0.1578904

[[3]]
            [,1]        [,2]        [,3]
[1,]  0.43727229  0.43727229  0.43727229
[2,] -0.59892198 -0.59892198 -0.59892198
[3,]  0.19675185  0.19675185  0.19675185
[4,]  0.07507512  0.07507512  0.07507512

[[4]]
            [,1]        [,2]       [,3]
[1,] -0.07934821  0.50791885 -0.8654880
[2,]  1.08078326 -0.87973030 -0.4644560
[3,] -0.53365990  0.77265039 -0.4570556
[4,] -0.75617703 -0.01195553  1.5981810
```

The Kronecker inner products of the four components are:

```
              [,1]          [,2]          [,3]          [,4]
[1,]  6.027679e-03  5.637851e-18 -3.122502e-17 -6.938894e-18
[2,] -1.127570e-17  8.540491e-01 -1.734723e-16  1.110223e-16
[3,] -3.816392e-17  1.075529e-16  2.740579e+00 -1.110223e-16
[4,]  2.081668e-17 -5.551115e-17 -8.604228e-16  9.280312e+00
```

The trace of the inner product matrix and the squared norm of the data:

```
[1] 12.88097 12.88097
```

4 Code

```
set.seed(12345)
ymat <- matrix(rnorm(12), 4, 3)
rmat <- crossprod(matrix(rnorm(40), 10, 4)) / 10
cmat <- crossprod(matrix(rnorm(30), 10, 3)) / 10

decompose <- function(xmat, rmat, cmat) {
  n <- nrow(xmat)
  m <- ncol(xmat)
  pmat <- matrix(colSums(rmat), n, n, byrow = TRUE) / sum(rmat)
  qmat <- matrix(rowSums(cmat), m, m) / sum(cmat)
  px <- pmat %*% xmat
  xq <- xmat %*% qmat
  dd <- rep(list(0),4)
  dd[[1]] <- px %*% qmat
  dd[[2]] <- px - dd[[1]]
  dd[[3]] <- xq - dd[[1]]
  dd[[4]] <- xmat - dd[[1]] - dd[[2]] - dd[[3]]
  ip <- matrix(0, 4, 4)
  for (i in 1:4) {
    for (j in 1:4) {
      ip[i, j] <- kroneckerIP(dd[[i]], dd[[j]], rmat, cmat)
    }
  }
  ssq <- c(kroneckerIP(xmat, xmat, rmat, cmat), sum(diag(ip)))
  return(list(dd = dd, ip = ip, ssq = ssq))
}

kroneckerIP <- function(xmat, ymat, rmat, cmat) {
  return(sum(ymat * (rmat %*% xmat %*% cmat)))
}
```