There are two ways to three ways to specify a multivariate normal density function in Stan. The first is the most straightforward – given a mean vector `mu` and covariance matrix `Sigma` we can implement the *centered parameterization* as

```
f ~ multi_normal(mu, Sigma).
```

If we have a Cholesky factorization of the covariance matrix that satisfies

$$\Sigma = L_{\Sigma} \cdot L_{\Sigma}^{T}$$

then we can equivalently implement this as

```
f ~ multi_normal_cholesky(mu, L_Sigma).
```

Due to some historical hiccups the `multi_normal` density function is actually less efficient than the `multi_normal_cholesky` density function so constructing the Choleksy factor yourself is often a good idea even with this centered parameterization.

The third option also uses a Cholesky factor. A *non-centered parameterization* starts with independent normal densities and then reconstructs the multivariate normal through a transformation,

```
f_tilde ~ normal(0, 1)
f = mu + L_Sigma * f_tilde.
```

As in the scalar case the non-centered parameterization leads to nicer posterior geometries when `f` is only weakly informed by the likelihood function while the centered parameterization is better when the likelihood function is strongly informative.

There's one more trick that we can play with the Cholesky factor. The covariance matrix is often parameterized as

$$\Sigma_{ij} = \sigma_i \cdot \sigma_j \cdot \rho_{ij}$$

with $\rho_{ii} = 0$ and $\rho_{ij} = \rho_{ji}$. More explicitly,

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \cdots & \sigma_1 \cdot \sigma_j \cdot \rho_{1j} & \cdots & \sigma_1 \cdot \sigma_N \cdot \rho_{1N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sigma_i \cdot \sigma_1 \cdot \rho_{1i} & \cdots & \sigma_i^2 & \cdots & \sigma_i \cdot \sigma_N \cdot \rho_{Ni} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sigma_N \cdot \sigma_1 \cdot \rho_{1N} & \cdots & \sigma_N \cdot \sigma_j \cdot \rho_{jN} & \cdots & \sigma_N^2 \end{pmatrix}.$$

In this case we can factor out the scales σ_i into diagonal matrices,

$$\Sigma = \begin{pmatrix} \sigma_1 & \cdots & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \sigma_i & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & \cdots & \sigma_N \end{pmatrix} \cdot \begin{pmatrix} 1 & \cdots & \rho_{1j} & \cdots & \rho_{1N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{1i} & \cdots & 1 & \cdots & \rho_{Ni} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{1N} & \cdots & \rho_{jN} & \cdots & 1 \end{pmatrix} \cdot \begin{pmatrix} \sigma_1 & \cdots & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \sigma_i & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & \cdots & \sigma_N \end{pmatrix},$$

or in matrix notation,

$$\Sigma = \text{diag}(\sigma) \cdot \Gamma \cdot \text{diag}(\sigma)$$

where

$$\Gamma = \begin{pmatrix} 1 & \cdots & \rho_{1j} & \cdots & \rho_{1N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{1i} & \cdots & 1 & \cdots & \rho_{Ni} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \rho_{1N} & \cdots & \rho_{jN} & \cdots & 1 \end{pmatrix}$$

is the *correlation matrix*.

This pattern of diagonal matrix/dense matrix/diagonal matrix is particularly convenient for Cholesky factorizations. If L_Γ is the Cholesky factor for the correlation matrix,

$$\Gamma = L_\Gamma \cdot L_\Gamma^T,$$

then

$$\begin{aligned} (\text{diag}(\sigma) \cdot L_\Gamma) \cdot (\text{diag}(\sigma) \cdot L_\Gamma)^T &= \text{diag}(\sigma) \cdot L_\Gamma \cdot L_\Gamma^T \cdot \text{diag}(\sigma)^T \\ &= \text{diag}(\sigma) \cdot L_\Gamma \cdot L_\Gamma^T \cdot \text{diag}(\sigma) \\ &= \text{diag}(\sigma) \cdot \Gamma \cdot \text{diag}(\sigma) \\ &= \Sigma. \end{aligned}$$

In words $\text{diag}(\sigma) \cdot L_\Gamma$ is the Cholesky factor for the covariance matrix Σ! If we've already constructed L_Γ then we can immediately construct L_Σ with a cheap multiplication by a diagonal matrix instead of first constructing Σ and then running a Cholesky factorization directly.

This is where Ben's suggestion comes from. Given this covariance-correlation factorization one can write the non-centered parameterization as

```
f_tilde ~ normal(0, 1)
f = mu + diag(sigma) * L_Gamma * f_tilde,
```

or more efficiently,

```
f_tilde ~ normal(0, 1)
f = mu + diag_pre_multiply(sigma, L_Gamma) * f_tilde.
```

That said we can also apply this to the centered parameterization,

```
L_Sigma = diag_pre_multiply(sigma, L_Gamma)
f ~ multi_normal_cholesky(mu, L_Sigma).
```

So why does any of this matter? Computing a Cholesky decomposition is expensive, and that computational expense scales terribly with dimension. If the Cholesky decomposition takes 1 second for a 10x10 matrix it will take 1000 seconds for a 100x100 matrix! A Stan program that runs reasonably fast for a small problem will become incredibly slow for larger problems if we have to take the Cholesky factorization every time we evaluate the model.

We only need to re-evaluate the Cholesky factorization, however, if the matrix depends on the changing parameter configuration. For example let's say that the correlation matrix is fixed, say by an estimated phylogeny, but the scales are not. In other words we have

$$\Sigma = \text{diag}(\sigma) \cdot \Gamma \cdot \text{diag}(\sigma)$$

where Γ is constant but σ depends on parameters. Because Σ also depends on parameters L_Σ will too, and will have to be recomputed every model evaluation. If we take a full Cholesky factorization, i.e. if you see a `cholesky` call in the `transformed parameters` or `models` blocks, then these recomputations become really, really expensive.

In this case, however, we don't have to take a full Cholesky decomposition! Because Γ is constant we can compute L_Γ once, for example in the `transformed data` block, and then just compute

```
L_Sigma = diag_pre_multiply(sigma, L_Gamma)
```

as needed in the `transformed parameters` or `models` blocks. This gives the same answer but is *much* less expensive.

Unfortunately this isn't quite what you have. The problem is that λ parameter that scales everything *but* the diagonal of the phylogeny-induced correlation matrix. There's no way to factor that kind of scaling out into a diagonal matrix multiplication as we can for the σ parameter that scales *all* of the entries of the matrix. Consequently there's no way to update the Cholesky factor efficiently here.

Anyone more versed in linear algebra might also consider an alternative strategy; feel free to ignore this if you're not familiar with any of the concepts! Your covariance matrix could also be written as

$$\begin{aligned}\Sigma &= \text{diag}(\sigma) \cdot \left[\text{diag}(\sqrt{\lambda}) \cdot \Gamma \cdot \text{diag}(\sqrt{\lambda}) + \mathbb{I} - \text{diag}(\lambda)\right] \cdot \text{diag}(\sigma) \\ &= \text{diag}(\sigma) \cdot \text{diag}(\sqrt{\lambda}) \cdot \Gamma \cdot \text{diag}(\sqrt{\lambda}) \cdot \text{diag}(\sigma) + \text{diag}(\sigma^2(1-\lambda)).\end{aligned}$$

The first term admits the efficient factorization that we want, and the second term can be written as the sum of N rank-1 updates. There is a way to update a Cholesky factor to account for a rank-1 update, but accounting for N rank-1 updates ends up being the same cost as re-evaluating the entire Cholesky factor so you don't end up with any savings.