Overview of Statistical Analysis of Spatial Data
Geog 210C
Ordinary and Simple Kriging

Chris Funk
Outline

- Examination of the math behind ordinary and simple kriging
  - Ordinary Kriging assumes an unknown constant trend ($\mu$)
    - Interpolation weights sum to 1
  - Simple kriging assumes a known mean ($\mu=0$)
Remember Regression Estimates can be computed via Ordinary Least Squares (OLS)

\[ y = \beta^T X + \varepsilon \]

The OLS solution is based on 1) the covariance matrix of \( X \) and 2) the covariance vector between \( X \) and \( y \).

\[ \beta = \left( X^T X \right)^{-1} X y \]

Kriging is very similar

- The point-to-point distances determine \( \left( X^T X \right)^{-1} \)
- The point-to-target distances determine \( X y \)
The Random Model Function

Setup – We are solving the ordinary kriging system

- Assume we are looking \( n+1 \) random variables
  - \( V(x_i), \ i = 1 \ldots n \) denote the random variables, at our observing sites
  - \( v_i, \ i = 1 \ldots n \) denote the actual values at our observing sites

- We are interested in estimating a value at a location \( x_0 \)
  - \( v_0 \) denotes an actual (unobserved value)
  - \( V(x_0) \) denotes the random variable at location \( x_0 \)

- The point-to-point covariance matrix (based on the semi-variogram model) is denoted as \( \tilde{C}_{ij} \)
- The point-to-interpolation target covariances are denoted as \( \tilde{C}_{i0} \)
We can define our estimates as weighted sum of point observations:

\[ \hat{v} = \sum_{j=1}^{n} w_j v_j \]

And define our mean error term for a new set of k-locations

\[ \text{Error} = \frac{1}{k} \sum_{i=1}^{k} \hat{v}_i - v_i \]

Note, however, that we do not know the true values at these k-points, so we cannot estimate the error directly. We can however estimate the error based on the spatial covariance structure

\[ R(x_0) = \hat{V}(x_0) - V(x_0) = \sum_{i=1}^{n} w_i V(x_i) - V(x_0) \]

Where \( x_0 \) denotes where we are estimating the error, and \( x_i \) denotes a set of neighboring locations. \( V(x_i) \) and \( V(x_0) \) are random variables. So is \( R(x_0) \), which denotes the expected error term solely in terms of the n+1 random variables

c.f. Isaaks & Srivastava, pg. 280-285
What is the expected value of our Error?

\[ R(x_0) = \hat{V}(x_0) - V(x_0) = \sum_{i=1}^{n} w_i V(x_i) - V(x_0) \]

\[ E\{R(x_0)\} = E\left\{ \sum_{i=1}^{n} w_i V(x_i) - V(x_0) \right\} \]

\[ ... = E\left\{ \sum_{i=1}^{n} w_i V(x_i) \right\} - E\{V(x_0)\} \]

Since we assume stationarity

\[ V(x_i) = V(x_0) = V \]

\[ E\{R(x_0)\} = 0 = E\{V\} \sum_{i=1}^{n} w_i - E\{V\} \]

Assume unbiasedness

\[ E\{V\} \sum_{i=1}^{n} w_i = E\{V\} \]

\[ \therefore \sum_{i=1}^{n} w_i = 1 \]

Weights that sum to 1 produce unbiased estimates (why?)
Unlike most other methodologies, kriging is ‘optimal’ in the sense that it
Minimizes the model error variance

\[
\sigma^2_R = \frac{1}{k} \sum_{i=1}^{k} (r_i - m_R)^2
\]

\[
... = \frac{1}{k} \sum_{i=1}^{k} (r_i - 0)^2 = \frac{1}{k} \sum_{i=1}^{k} (\hat{v}_i - v_i)^2
\]

Again, we don’t know \( v_i \)

We can, however, use our random model function to estimate our expected errors

\[
\hat{V}(x_0) = \sum_{i=1}^{n} w_i V(x_i)
\]

\[
R(x_0) = \hat{V}(x_0) - V(x_0)
\]

\( R(x_0) \) are our anticipated residuals

\[
Var\left\{ \sum_{i=1}^{n} w_i V_i \right\} = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i \cdot w_j \cdot Cov(V_i V_j)
\]

This is the variance of a weighted linear combination of covarying random variables
What is the expected variance of our errors?

\[ R(x_0) = \hat{V}(x_0) - V(x_0) \]

Start by asking, what is the expected variance of our weighted combinations:

\[ \hat{V}(x_0) = \sum_{i=1}^{n} w_i V(x_i) \]

Remember, for the simple addition of 2 RVs, the expected covariance is:

\[ Cov(U + V) = Cov(UU) + 2Cov(UV) + Cov(VV) \]

This can be generalized:

\[ Var\left\{ \sum_{i=1}^{n} w_i V_i \right\} = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i \cdot w_j Cov(V_i V_j) \]

This is the variance of a weighted linear combination of co-varying random variables.

c.f. Isaaks & Srivastava, pg. 217
\[ R(x_0) = \hat{V}(x_0) - V(x_0) \]

\[ \text{Var}\left\{ \sum_{i=1}^{n} w_i V_i \right\} = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \text{Cov}(V_i V_j) \]

Combining our definition of residuals with the covariance estimate:

\[
\text{foil} \quad \text{Var}\{R(x_0)\} = \text{Cov}\{\hat{V}(x_0)\hat{V}(x_0)\} - 2\text{Cov}\{\hat{V}(x_0)V(x_0)\} + \text{Cov}\{V(x_0)V(x_0)\}
\]

The 1st term can be estimated based on the linear combination weights:

\[
\text{Cov}\{\hat{V}(x_0)\hat{V}(x_0)\} = \text{Var}\{\hat{V}(x_0)\hat{V}(x_0)\} = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \tilde{C}_{ij}
\]

The 3rd term is just the variance of the random variable \( V(x_0) \):

\[
\text{Cov}\{V(x_0)V(x_0)\} = \text{Var}\{V(x_0)V(x_0)\} = \tilde{\sigma}^2
\]

The second terms involve the semi-variances .....
The middle term of our variance estimate:

\[
Var\{R(x_0)\} = Cov\{\hat{V}(x_0)\hat{V}(x_0)\} - 2Cov\{\hat{V}(x_0)V(x_0)\} + Cov\{V(x_0)V(x_0)\}
\]

Weighted sums of point data

Can be written as:

\[
2Cov\{\hat{V}(x_0)V(x_0)\} = 2Cov\left\{\left(\sum_{i=1}^{n} w_i V_i\right) V_0\right\}
\]

\[
= 2 \sum_{i=1}^{n} w_i \, Cov\{V_i, V_0\}
\]

\[
= 2 \sum_{i=1}^{n} w_i \, \tilde{C}_{i0}
\]

Covariance matrix

Combining these three terms again yields

\[
\sigma_R^2 = \hat{\sigma}^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \tilde{C}_{i,j} - 2 \sum_{i=1}^{n} w_i \tilde{C}_{i0}
\]
The middle term of our variance estimate:

\[
\text{Var}\{R(x_0)\} = \text{Cov}\{\hat{V}(x_0)\hat{V}(x_0)\} - 2\text{Cov}\{\hat{V}(x_0)V(x_0)\} + \text{Cov}\{V(x_0)V(x_0)\}
\]

Weighted sums of point data

Can be written as:

\[
2\text{Cov}\{\hat{V}(x_0)V(x_0)\} = 2\text{Cov}\left\{\left(\sum_{i=1}^{n} w_i V_i\right)V_0\right\} \\
= 2\sum_{i=1}^{n} w_i \text{Cov}\{V_i,V_0\} \\
= 2\sum_{i=1}^{n} w_i \widetilde{C}_{i0}
\]

Combining these three terms we now have

\[
\sigma_R^2 = \sigma^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \widetilde{C}_{ij} - 2\sum_{i=1}^{n} w_i \widetilde{C}_{i0}
\]

Yahoo! We now have an equation with \(n\) unknowns \((w)\). Such an equation can often be solved by taking the 1st derivatives and setting them to 0. BUT ... we also have to deal with the unbiasedness constraint. \(w\) must sum to 1.
Right now we have $n$ equations, and $n$ unknowns – one for each point location.

Ideally, we would also like to add a constraint enforcing bias removal, such that the sum of weights $= 1$. This makes our system unsoluble (since we now have $n+1$ equations). We address this by adding another variable, $\mu$, a Lagrange parameter. This gives us an $n+1$ system of constrained equations.

$$\tilde{\sigma}_R^2 = \tilde{\sigma}^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \tilde{C}_{ij} - 2 \sum_{i=1}^{n} w_i \tilde{C}_{i0} + 2 \mu \left( \sum_{i=1}^{n} w_i - 1 \right) \quad \text{forces sum of weights to 1}$$
Minimizing the error variance

Given the variance value and the covariance matrix $C$, we can use this equation to solve for the optimal set of weights for a given location. We take the 1st derivative, and find solutions such that the values are 0. The optimal 1st weight ($w_1$) is

1st term ($\sigma^2$) goes away, since it does not vary with $w_1$

2nd term becomes

$$\frac{\partial}{\partial w_1} \left( \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \tilde{C}_{ij} \right) = 2 \sum_{j=1}^{n} w_j \tilde{C}_{1j}$$

3rd term becomes

$$\frac{\partial}{\partial w_1} \left( \sum_{i=1}^{n} w_i \tilde{C}_{i0} \right) = \frac{\partial}{\partial w_1} \left( w_1 \tilde{C}_{10} \right) = \tilde{C}_{10}$$

4th term becomes

$$\frac{\partial}{\partial w_1} \left( \mu \left( \sum_{i=1}^{n} w_i - 1 \right) \right) = \frac{\partial}{\partial w_1} \left( \mu w_1 \right) = \mu$$
The equation for the 1st weight is now

\[
\frac{\partial (\sigma_R^2)}{\partial w_i} = 2 \sum_{j=1}^{n} w_j \tilde{C}_{1j} - 2 \tilde{C}_{10} + 2 \mu
\]

Setting the 1st derivative to 0 yields

\[
2 \sum_{j=1}^{n} w_j \tilde{C}_{1j} - 2 \tilde{C}_{10} + 2 \mu = 0
\]

\[
\sum_{j=1}^{n} w_i \tilde{C}_{1j} + \mu = \tilde{C}_{10}
\]

We can do this ‘n’ times

\[
\sum_{j=1}^{n} w_j \tilde{C}_{1j} + \mu = \tilde{C}_{10}
\]

\[
\sum_{j=1}^{n} w_j \tilde{C}_{2j} + \mu = \tilde{C}_{20}
\]

\[
\sum_{j=1}^{n} w_j \tilde{C}_{3j} + \mu = \tilde{C}_{30}
\]

\[
\ldots
\]

\[
\sum_{j=1}^{n} w_j \tilde{C}_{nj} + \mu = \tilde{C}_{n0}
\]

We can add a constraint forcing the weights sum to 1.

\[
\sum_{j=1}^{n} w_i \tilde{C}_{ij} + \mu = \tilde{C}_{i0} \forall i = 1 \ldots n
\]

\[
\sum_{j=1}^{n} w_i = 1
\]

Solving this system of equations provides optimal ‘ordinary’ kriging weights, the weights sum to 1.
Matrix formulation

We solve by inverting the matrix formula and multiplying through:

\[ C \cdot w = D \]
\[ C^{-1} \cdot C \cdot w = C^{-1} \cdot D \]
\[ I \cdot w = C^{-1} \cdot D \]
\[ w = C^{-1} \cdot D \]
An intuitive analysis of OK

\[ C \cdot w = D \]
\[ C^{-1} \cdot C \cdot w = C^{-1} \cdot D \]
\[ I \cdot w = C^{-1} \cdot D \]
\[ w = C^{-1} \cdot D \]

C terms provide information about the clustering of the data

Multiplying by \( C^{-1} \) ‘whitens’ the data, reducing the effects of clustering

D terms provide a weighting scheme similar to inverse distance weighting
### Data

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>V</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>61</td>
<td>139</td>
<td>477</td>
<td>4.5</td>
</tr>
<tr>
<td>2</td>
<td>63</td>
<td>140</td>
<td>696</td>
<td>3.6</td>
</tr>
<tr>
<td>3</td>
<td>64</td>
<td>129</td>
<td>227</td>
<td>8.1</td>
</tr>
<tr>
<td>4</td>
<td>68</td>
<td>128</td>
<td>646</td>
<td>9.5</td>
</tr>
<tr>
<td>5</td>
<td>71</td>
<td>140</td>
<td>606</td>
<td>6.7</td>
</tr>
<tr>
<td>6</td>
<td>73</td>
<td>141</td>
<td>791</td>
<td>8.9</td>
</tr>
<tr>
<td>7</td>
<td>75</td>
<td>128</td>
<td>783</td>
<td>13.5</td>
</tr>
</tbody>
</table>
Covariance model

\[ C(h) = 10 \exp\left(-\frac{3h}{10}\right) \]
# Distances and $C$ and $C^{-1}$

## Distances

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>0.000000</td>
<td>2.236068</td>
<td>10.440307</td>
<td>13.038405</td>
<td>10.049876</td>
<td>12.165525</td>
<td>17.80449</td>
</tr>
</tbody>
</table>

## $C$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>10.000000</td>
<td>5.11288948</td>
<td>0.4362644</td>
<td>0.2001003</td>
<td>0.4904767</td>
<td>0.2600003</td>
<td>0.04789410</td>
<td>1</td>
</tr>
<tr>
<td>[2]</td>
<td>5.1128895</td>
<td>10.000000</td>
<td>0.3638465</td>
<td>0.2024191</td>
<td>0.9071795</td>
<td>0.4904767</td>
<td>0.06150826</td>
<td>1</td>
</tr>
<tr>
<td>[3]</td>
<td>0.4362644</td>
<td>0.3638465</td>
<td>10.000000</td>
<td>2.9027350</td>
<td>0.2001003</td>
<td>0.1110900</td>
<td>0.3638465</td>
<td>1</td>
</tr>
<tr>
<td>[4]</td>
<td>0.2001003</td>
<td>0.2024191</td>
<td>2.9027350</td>
<td>10.000000</td>
<td>0.2445807</td>
<td>0.1532122</td>
<td>1.22456428</td>
<td>1</td>
</tr>
<tr>
<td>[5]</td>
<td>0.4904767</td>
<td>0.9071795</td>
<td>0.2001003</td>
<td>0.2445807</td>
<td>10.000000</td>
<td>5.1128895</td>
<td>0.22488905</td>
<td>1</td>
</tr>
<tr>
<td>[6]</td>
<td>0.2600003</td>
<td>0.4904766</td>
<td>0.1110900</td>
<td>0.1532122</td>
<td>5.1128895</td>
<td>10.000000</td>
<td>0.19334119</td>
<td>1</td>
</tr>
<tr>
<td>[7]</td>
<td>0.0478941</td>
<td>0.06150826</td>
<td>0.3638465</td>
<td>1.2245643</td>
<td>0.2248891</td>
<td>0.1933412</td>
<td>10.000000</td>
<td>1</td>
</tr>
<tr>
<td>[8]</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0</td>
</tr>
</tbody>
</table>

## $C^{-1}$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>0.127013252</td>
<td>-0.076697116</td>
<td>-0.013056968</td>
<td>-0.009041150</td>
<td>-0.007680222</td>
<td>-0.008713079</td>
<td>-0.01182472</td>
<td>0.1361155</td>
</tr>
<tr>
<td>[2]</td>
<td>-0.076697116</td>
<td>0.129492450</td>
<td>-0.010253764</td>
<td>-0.008425832</td>
<td>-0.015243522</td>
<td>-0.008232264</td>
<td>-0.01063995</td>
<td>0.1211765</td>
</tr>
<tr>
<td>[3]</td>
<td>-0.013056968</td>
<td>-0.010253764</td>
<td>0.098176809</td>
<td>-0.041622890</td>
<td>-0.009545242</td>
<td>-0.01013049</td>
<td>-0.01358490</td>
<td>0.1564555</td>
</tr>
<tr>
<td>[4]</td>
<td>-0.009041150</td>
<td>-0.008425832</td>
<td>-0.041622890</td>
<td>0.101856313</td>
<td>-0.009099830</td>
<td>-0.009256027</td>
<td>-0.02441059</td>
<td>0.1393079</td>
</tr>
<tr>
<td>[5]</td>
<td>-0.007680222</td>
<td>-0.015243522</td>
<td>-0.009545242</td>
<td>-0.009099830</td>
<td>0.129933149</td>
<td>-0.076767527</td>
<td>-0.01159681</td>
<td>0.1175116</td>
</tr>
<tr>
<td>[6]</td>
<td>-0.008713079</td>
<td>-0.008232264</td>
<td>-0.010113049</td>
<td>-0.009256027</td>
<td>-0.076767527</td>
<td>0.126296973</td>
<td>-0.01321503</td>
<td>0.1409339</td>
</tr>
<tr>
<td>[7]</td>
<td>-0.011824718</td>
<td>-0.010639952</td>
<td>-0.01358490</td>
<td>-0.02441059</td>
<td>-0.01159681</td>
<td>0.08527198</td>
<td>0.1884991</td>
<td>0.1884991</td>
</tr>
<tr>
<td>[8]</td>
<td>0.136115493</td>
<td>0.121176509</td>
<td>0.156455503</td>
<td>0.139307891</td>
<td>0.117511616</td>
<td>0.140933893</td>
<td>0.1884991</td>
<td>-2.1801561</td>
</tr>
</tbody>
</table>
D and W

\[ D = [2.59 \ 3.40 \ 0.88 \ 0.58 \ 1.34 \ 0.69 \ 0.17 \ 1.00] \]

\[
> w = C1 \ \%*\% \ D
> w

[,1]
[1,]  0.16981550
[2,]  0.32021697
[3,]  0.12798067
[4,]  0.08667481
[5,]  0.15108005
[6,]  0.05843369
[7,]  0.08579831
[8,] -0.90956663
\]
**Simple Kriging**

**SK prediction:**  
\[ \hat{y}(t_m) - \mu(t_m) = \sum_{n=1}^{N} w_m(s_n)[y(s_n) - \mu(s_n)] = w_m^T r_s = w_m^T [y_s - \mu_s] \]

- \( w_m = [w_m(s_n), n = 1, \ldots, N]^T \): \((N \times 1)\) vector of SK-weights assigned to \(N\) source supports for prediction at target support \(t_m\).

- \( r_s = [y(s_n) - \mu(s_n), n = 1, \ldots, N]^T \): \((N \times 1)\) vector of residual data from known expectations \(\mu(s_n)\) at source supports

\[ \begin{bmatrix} y(s_1) - \mu(s_1) \\ \vdots \\ y(s_N) - \mu(s_N) \end{bmatrix} \]

\[ \hat{y}(t_m) - \mu(t_m) = \begin{bmatrix} w_m(s_1) & \cdots & w_m(s_n) & \cdots & w_m(s_N) \end{bmatrix} \begin{bmatrix} y(s_1) - \mu(s_1) \\ \vdots \\ y(s_n) - \mu(s_n) \\ \vdots \\ y(s_N) - \mu(s_N) \end{bmatrix} = w_m^T r_s \]

*Use semivariogram model to determine \(N\) weights at each target support \(t_m\); typically, we use the covariogram model (the kernel) due to computational reasons.*
Requisites for Kriging

Source-to-target & source-to-source distances:

\[ d_m = \begin{bmatrix} d_{1m} \\ \vdots \\ d_{nm} \\ \vdots \\ d_{Nm} \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} 0 & \cdots & d_{1n'} & \cdots & d_{1N} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{n1} & \cdots & 0 & \cdots & d_{nN} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{N1} & \cdots & d_{Nn'} & \cdots & 0 \end{bmatrix} \]

- as any other spatial interpolation method, one accounts for the proximity of the \( N \) source supports to the target support \( t_m \). **Note:** Vector \( d_m \) changes from one target support \( t_m \) to another, hence the subscript \( m \)

- unlike other interpolation methods, Kriging also accounts for the proximity between source supports themselves (sample configuration or data layout). **Note:** For global interpolation, matrix \( D \) of source-to-source distances is the same for all target supports
Inputs

(N x 1) vector of source-to-target distances:

\[ d_m = \begin{bmatrix} 3.61 & 4.47 & 6.71 & 8.06 & 8.94 & 9.49 & 13.45 \end{bmatrix}^T \]

\( n' \)-th element of \( d_m \) is \( d_{nm} \)

(N x N) matrix of source-to-source distances:

\[
D = \begin{bmatrix}
0.00 & 2.24 & 8.00 & 11.05 & 10.05 & 13.00 & 16.97 \\
2.24 & 0.00 & 10.05 & 10.44 & 12.17 & 13.04 & 17.80 \\
8.00 & 10.05 & 0.00 & 13.04 & 2.24 & 12.37 & 12.65 \\
11.05 & 10.44 & 13.04 & 0.00 & 15.00 & 4.12 & 11.05 \\
10.05 & 12.17 & 2.24 & 15.00 & 0.00 & 13.93 & 13.15 \\
13.00 & 13.04 & 12.37 & 4.12 & 13.93 & 0.00 & 7.00 \\
16.97 & 17.80 & 12.65 & 11.05 & 13.15 & 7.00 & 0.00 \\
\end{bmatrix}
\]

\( n, n' \)-th element of \( D \) is \( d_{nn'} \)
Covariance transform

From distance matrices to model covariance matrices: Take any distance value $d_{nm}$ and $d_{nn'}$, i.e., any entry in $d_m$ and $D$, and transform it, via the covariogram model or kernel, to a covariance value $\sigma(d_{nm})$ and $\sigma(d_{nn'})$.

Source-to-target & source-to-source model covariances:

$\sigma_m = \begin{bmatrix} \sigma(d_{1m}) \\ \vdots \\ \sigma(d_{nm}) \\ \vdots \\ \sigma(d_{Nm}) \end{bmatrix}$

and

$\Sigma = \begin{bmatrix} \sigma(0) & \cdots & \sigma(d_{1n'}) & \cdots & \sigma(d_{1N}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma(d_{n1}) & \cdots & \sigma(0) & \cdots & \sigma(d_{nN}) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma(d_{N1}) & \cdots & \sigma(d_{Nn'}) & \cdots & \sigma(0) \end{bmatrix}$

- source-to-target covariance vector $\sigma_m$: $(N \times 1)$ vector with model covariance values $\sigma(d_{nm})$ between $N$ source supports and target support $t_m$.

- source-to-source covariance matrix $\Sigma$: $(N \times N)$ matrix with model covariance values $\sigma(d_{nn'})$ between any two source supports separated by distance $d_{nn'}$. 
Calculating weights

\[
\begin{bmatrix}
3.61 \\
4.47 \\
6.71 \\
8.06 \\
8.94 \\
9.49 \\
13.45
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\exp(-3 \times 3.61/10) \\
\exp(-3 \times 4.47/10) \\
\exp(-3 \times 6.71/10) \\
\exp(-3 \times 8.06/10) \\
\exp(-3 \times 8.94/10) \\
\exp(-3 \times 9.49/10) \\
\exp(-3 \times 13.45/10)
\end{bmatrix}
= \begin{bmatrix}
0.34 \\
0.26 \\
0.13 \\
0.09 \\
0.07 \\
0.06 \\
0.02
\end{bmatrix}
\]

These would be the weights, had one ignored auto-correlation between source data.
Source-to-target & source-to-source model covariances:

\[
\mathbf{\sigma}_m = \begin{bmatrix}
\sigma(d_{1m}) \\
\vdots \\
\sigma(d_{nm}) \\
\vdots \\
\sigma(d_{Nm})
\end{bmatrix}
\quad \text{and} \quad
\mathbf{\Sigma} = \begin{bmatrix}
\sigma(0) & \cdots & \sigma(d_{1n'}) & \cdots & \sigma(d_{1N}) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\sigma(d_{n1}) & \cdots & \sigma(0) & \cdots & \sigma(d_{nN}) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\sigma(d_{N1}) & \cdots & \sigma(d_{Nn'}) & \cdots & \sigma(0)
\end{bmatrix}
\]

- **source-to-target covariance vector** $\mathbf{\sigma}_m$: encapsulates statistical proximity (correlation) between source data and unknown target value $y(t_m)$; that correlation is a function of distance between source and target supports, not of the actual (unknown) value $y(t_m)$; the larger the entries of vector $\mathbf{\sigma}_m$, the stronger the predictive power of source data (had each source datum been considered in isolation)

- **source-to-source covariance matrix** $\mathbf{\Sigma}$: encapsulates redundancy between source supports for positive spatial auto-correlation, the more clustered are the source supports, the more redundant are the corresponding source data (less information content); a clustered source support layout typically translates into larger entries in $\mathbf{\Sigma}$
There is no requirement: that a stationary specification of the covariance vector $\sigma_m$ and the covariance matrix $\Sigma$ be adopted. In other words, one can specify unequal source variances (diagonal elements of $\Sigma$), as well as covariances between two supports (source-to-target or source-to-source) which are not functions of distance.

General specification of source-to-target & source-to-source covariances:

$$
\sigma_m = \begin{bmatrix}
\sigma(s_1, t_m) \\
\vdots \\
\sigma(s_n, t_m) \\
\sigma(s_N, t_m)
\end{bmatrix}
\quad \text{and} \quad
\Sigma = \begin{bmatrix}
\sigma(s_1) & \cdots & \sigma(s_1, s_N) \\
\vdots & \ddots & \vdots \\
\sigma(s_n, s_1) & \cdots & \sigma(s_n) \\
\vdots & \ddots & \vdots \\
\sigma(s_N, s_1) & \cdots & \sigma(s_N, s_N)
\end{bmatrix}
$$

Second-order stationarity: is a typical working hypothesis when dealing with spatial data (or a single cross-section from spatio-temporal data), whereby all diagonal entries of $\Sigma$ are equal to the variogram sill $\sigma(0)$ and are functions of distance between supports.
The Simple Kriging (SK) System of Equations

\[
\begin{bmatrix}
\sigma(s_1) & \cdots & \sigma(s_1, s_N) \\
\vdots & \ddots & \vdots \\
\sigma(s_N, s_1) & \cdots & \sigma(s_N)
\end{bmatrix}
\begin{bmatrix}
w_m(s_1) \\
\vdots \\
w_m(s_N)
\end{bmatrix}
= 
\begin{bmatrix}
\sigma(s_1, t_m) \\
\vdots \\
\sigma(s_N, t_m)
\end{bmatrix}
\]

\[\Sigma w_m = \sigma_m\]

- a system of \(N\) equations in \(N\) unknowns (the weights in \(w_m\)) for prediction at support \(t_m\); there are \(M\) such systems for \(M\) target supports, since \(\sigma_m\) changes from one target support to another

- a version of the system of normal equations used in multiple linear regression. For Kriging, the dependent variable pertains to the target support \(t_m\), and there are \(N\) predictor (lagged) variables pertaining to the \(N\) source supports

- the SK system is also known with different names in different disciplines, e.g., collocation in surveying, Yule-Walker equations in time-series modeling, Wiener prediction in electrical engineering, objective interpolation in atmospheric sciences

- under 2nd-order stationarity: \(\sigma(s_n) = \sigma(0), \ \sigma(s_n, s_n') = \sigma(0) \rho(d_{nn'})\) and \(\sigma(s_n, t_m) = \sigma(0) \rho(d_{nm});\) in this case, the weights do not depend on the sill \(\sigma(0)\)
\[
\begin{bmatrix}
\sigma(s_1) & \cdots & \sigma(s_1, s_N) \\
\vdots & \ddots & \vdots \\
\sigma(s_N, s_1) & \cdots & \sigma(s_N)
\end{bmatrix}^{-1}
\begin{bmatrix}
\sigma(s_1, t_m) \\
\vdots \\
\sigma(s_N, t_m)
\end{bmatrix}
\]

\[
w_m = \Sigma^{-1} \sigma_m
\]

- the weights vector \(w_m\) is obtained by solving the SK system anew for each target support \(t_m\) since the entries of \(\sigma_m\) change from one target support to another.

- the SK system has a unique solution (there is one and only one weights vector \(w_m\)) if and only if the source-to-source covariance matrix \(\Sigma\) is positive definite; for 2nd-order stationarity, this implies that a valid covariogram model \(\sigma(d; \theta)\), e.g., exponential distance decay, with \(\theta\) containing the sill and range, is used to populate \(\Sigma\); in this case, \(\sigma(d; \theta) = \sigma(0)\rho(d; \theta)\) and the weights do not depend on the sill \(\sigma(0)\):

\[
\begin{bmatrix}
w_m(s_1) \\
\vdots \\
w_m(s_N)
\end{bmatrix} = \frac{1}{\sigma(0)}
\begin{bmatrix}
1 & \cdots & \rho(d_{1N}) \\
\vdots & \ddots & \vdots \\
\rho(d_{N1}) & \cdots & 1
\end{bmatrix}^{-1}
\begin{bmatrix}
\rho(d_{1m}) \\
\vdots \\
\rho(d_{Nm})
\end{bmatrix}
\]