

STK4150/9150 - Environmental and Spatial Statistics

Spatio-temporal data and models

Thordis Thorarinsdottir

Spring 2026

Outline

- visualization (Ch. 5.1)
- empirical orthogonal function (EOF) analysis (Ch. 5.3)
- spatio-temporal covariance functions (Ch 6.1)
- spatio-temporal kriging (Ch. 6.2)
- time series of spatial processes (Ch. 6.4)

Spatio-temporal data: The main points

Spatio-temporal data result from

- observation networks
- remote-sensing platforms
- data-assimilation algorithms (output from computer models)

The characteristics of such data are

- large data sets
- dependencies across a large range of spatial and temporal scales

Key challenge: Extract information

Important tool: Exploratory analysis

Visualization approaches include

- Spatial maps over time
- Animations
- 1D-Space / Time-plot
- Time-series plots of particular features
- Empirical covariance/correlation functions

First, read in some data

```
library(maps)
library(fields)
library(animation)
SST = read.table("images_ch5/SST011970_032003.dat")
lon = c(paste(seq(124,180,2),"E"),
        paste(seq(178,70,-2),"W"))
lat = c(paste(seq(29,1,-2),"S"),
        paste(seq(1,29,2),"N"))
SST3D = array(NA,c(399,84,30))
for(i in 1:399)
  SST3D[i,,] = matrix(SST[,i],ncol=30,byrow=F)
namtime = NULL
for(y in 1970:2003)
  for(m in c("Jan","Feb","Mar","Apr","May","Jun",
            "Jul","Aug","Sep","Oct","Nov","Dec"))
    namtime = c(namtime,paste(y,m))
namtime = namtime[1:399]
```

```
dim(SST3D)
```

```
[1] 399  84  30
```

```
c(length(namtime), length(lon), length(lat))
```

```
[1] 399  84  30
```

```
lon[1:3]
```

```
[1] "124 E" "126 E" "128 E"
```

```
lat[1:3]
```

```
[1] "29 S" "27 S" "25 S"
```

```
namtime[1:3]
```

```
[1] "1970 Jan" "1970 Feb" "1970 Mar"
```

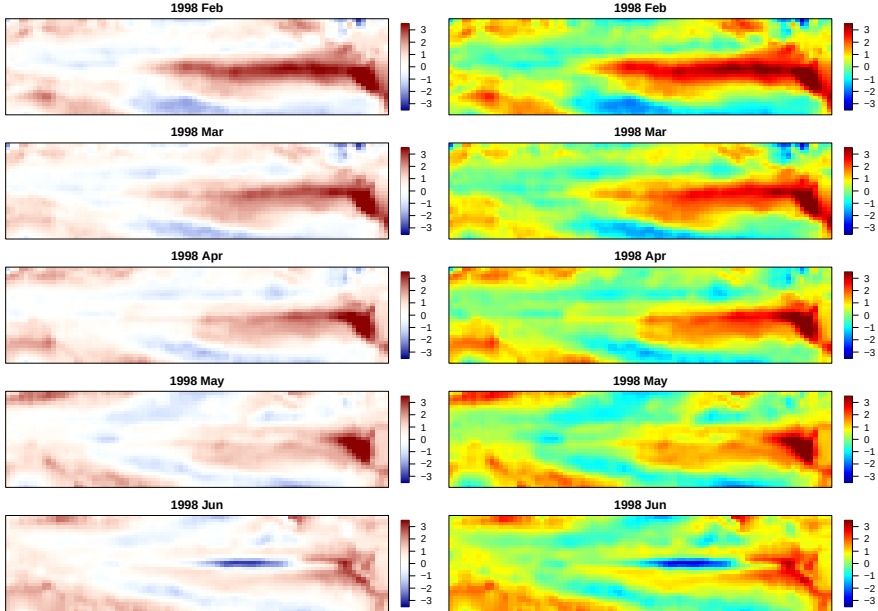
Data is from the book: Sea surface temperature anomalies in the tropical Pacific.

Spatial maps over time: The importance of color

```
#Spatial plots
par(mfrow=c(5,2),mar=c(0.5,0.5,1.9,1.9))
for(i in 1:5)
{
  x = matrix(SST[,337+i],ncol=30,byrow=F)
  x[x>3.5] = 3.5
  image.plot(x, add = FALSE, nlevel = 64, axes=FALSE,
             legend.shrink = 0.9, legend.width = 1.2, legend.mar = NULL,
             graphics.reset = FALSE, horizontal = FALSE, bigplot = NULL,
             smallplot = NULL, legend.only = FALSE,
             col = two.colors(n=256, start="darkblue", end="darkred", middle="white",alpha=1.0),
             lab.breaks=NULL, axis.args=NULL,zlim=c(-3.5,3.5))

  box()
  title(namtime[337+i])
  x = matrix(SST[,337+i],ncol=30,byrow=F)
  x[x>3.5] = 3.5
  image.plot(x, add = FALSE, nlevel = 64, axes=FALSE,
             legend.shrink = 0.9, legend.width = 1.2, legend.mar = NULL,
             graphics.reset = FALSE, horizontal = FALSE, bigplot = NULL,
             smallplot = NULL, legend.only = FALSE, col = tim.colors(256),
             lab.breaks=NULL, axis.args=NULL,zlim=c(-3.5,3.5))

  box()
  title(namtime[337+i])
}
```

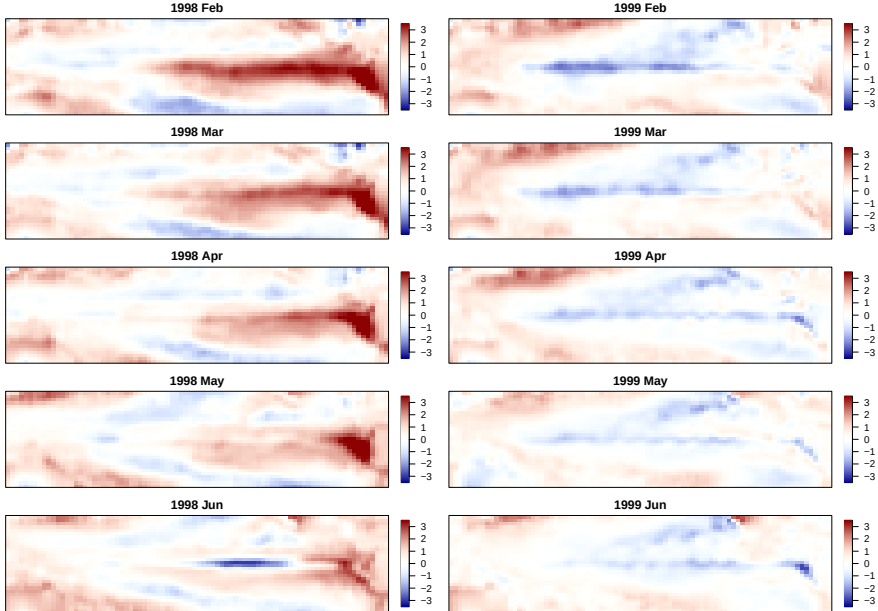


Spatial maps over time

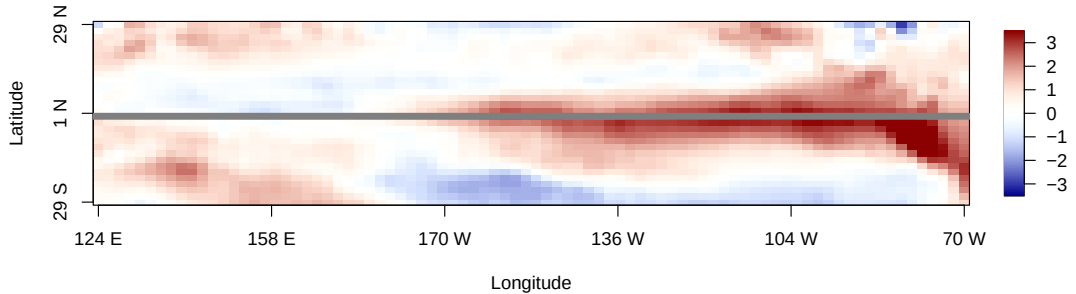
```
#Spatial plots
par(mfrow=c(5,2),mar=c(0.5,0.5,1.9,1.9))
for(i in 1:5)
{
  x = matrix(SST[,337+i],ncol=30,byrow=F)
  x[x>3.5] = 3.5
  image.plot(x, add = FALSE, nlevel = 64, axes=FALSE,
             legend.shrink = 0.9, legend.width = 1.2, legend.mar = NULL,
             graphics.reset = FALSE, horizontal = FALSE, bigplot = NULL,
             smallplot = NULL, legend.only = FALSE,
             col = two.colors(n=256, start="darkblue", end="darkred", middle="white",alpha=1.0),
             lab.breaks=NULL, axis.args=NULL,zlim=c(-3.5,3.5))

  box()
  title(namtime[337+i])
  x = matrix(SST[,337+12+i],ncol=30,byrow=F)
  x[x>3.5] = 3.5
  image.plot(x, add = FALSE, nlevel = 64, axes=FALSE,
             legend.shrink = 0.9, legend.width = 1.2, legend.mar = NULL,
             graphics.reset = FALSE, horizontal = FALSE, bigplot = NULL,
             smallplot = NULL, legend.only = FALSE,
             col = two.colors(n=256, start="darkblue", end="darkred", middle="white",alpha=1.0),
             lab.breaks=NULL, axis.args=NULL,zlim=c(-3.5,3.5))

  box()
  title(namtime[337+12+i])
}
```



1D-Space/Time-plot: Focus on latitude 1 S



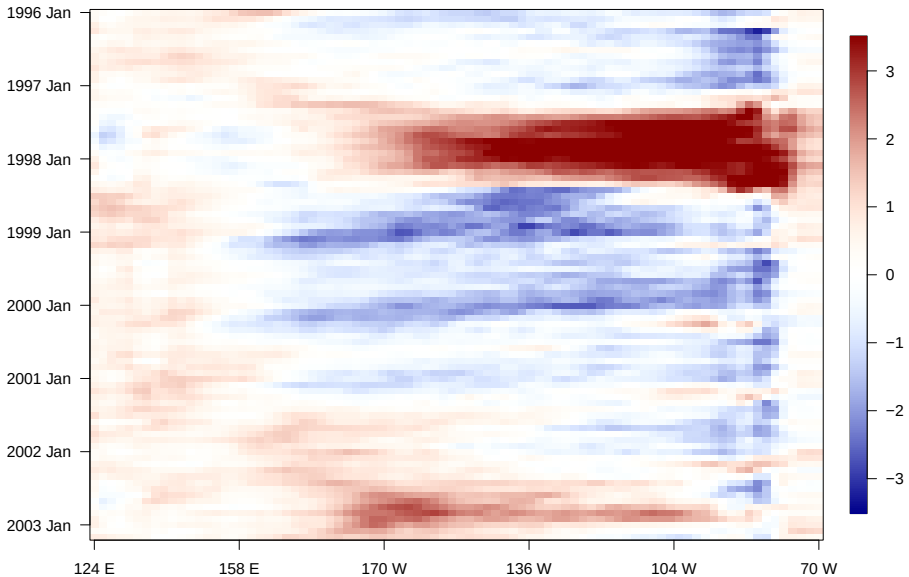
We plot a “time series” of the data slice at a given latitude close to the equator.

1D-Space/Time-plot

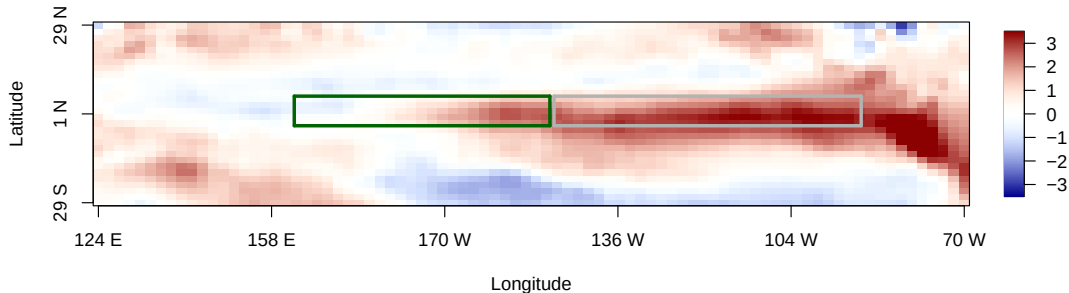
```
# 1D-Space/Time plots
x = SST3D
dim(x)
```

```
[1] 399  84  30
```

```
x[x>3.5] = 3.5
par(mar=c(5,5,1.9,1.9))
image.plot(t(x[399:313,,15]), add = FALSE, nlevel = 64, axes=FALSE,
           legend.shrink = 0.9, legend.width = 1.2, legend.mar = NULL,
           graphics.reset = FALSE, horizontal = FALSE, bigplot = NULL,
           smallplot = NULL, legend.only = FALSE,
           col = two.colors(n=256, start="darkblue", end="darkred", middle="white",alpha=1.0),
           lab.breaks=NULL, axis.args=NULL,zlim=c(-3.5,3.5),
           xlab="Longitude",ylab="")
axis(2, at=seq(1,0,by=-12/(399-313)), label=namtime[seq(313,399,by=12)], las=1)
axis(1, at=c(0,0.2,0.4,0.6,0.8,1), label=lon[round(seq(1,84,length=6))])
box()
```



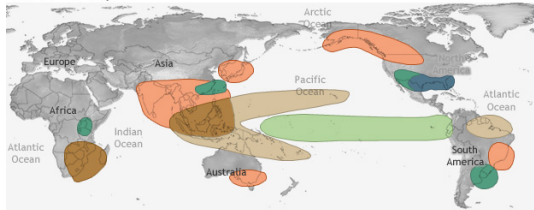
Time-series plots of particular features



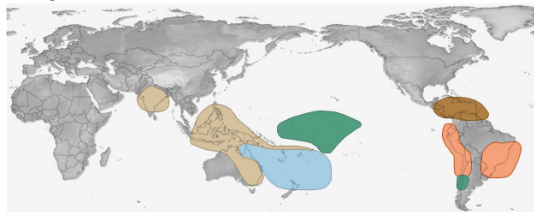
The El Niño-Southern Oscillation (ENSO) is described through various indices that are defined as average sea surface temperature anomalies over certain regions in the tropical Pacific. The green box corresponds to the Niño 3 region, while the grey box corresponds to the Niño 4 region.

EL NIÑO CLIMATE IMPACTS

December-February



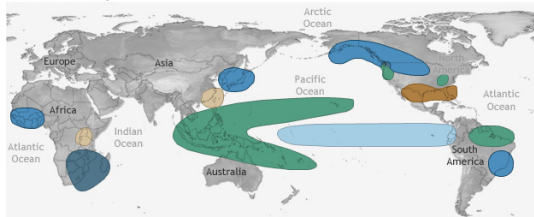
June-August



■ Cool ■ Wet ■ Cool and dry ■ Cool and Wet
■ Warm ■ Dry ■ Warm and dry ■ Warm and wet

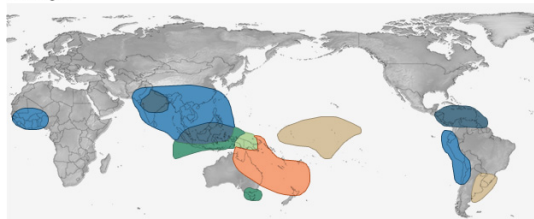
LA NIÑA CLIMATE IMPACTS

December-February



■ Cool ■ Wet ■ Cool and dry ■ Cool and Wet
■ Warm ■ Dry ■ Warm and dry ■ Warm and wet

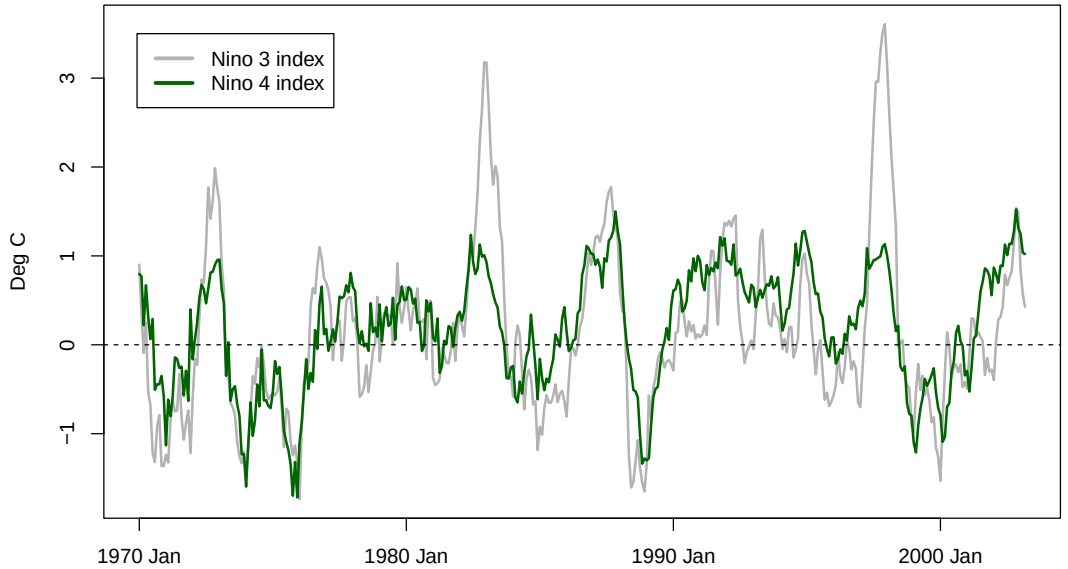
June-August



NOAA Climate.gov

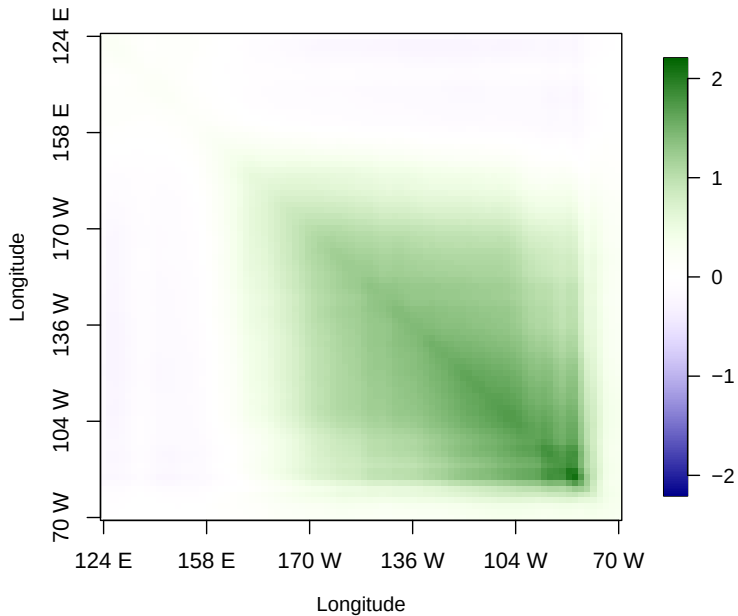
Time-series plots of ENSO

```
#Time series plot of "Nino 3" and "Nino 4" regions
nino3 = NULL;nino4 = NULL
for(i in 1:399)
{
  nino3 = c(nino3,mean(SST3D[i,44:74,13:18]))
  nino4 = c(nino4,mean(SST3D[i,19:44,13:18]))
}
matplot(cbind(1:399,1:399),cbind(nino3,nino4),type="l",col=c("gray70","darkgreen"),lty=1,lwd=2,
        xlab="Time",ylab="Deg C", axes=FALSE)
axis(2); axis(1, at=c(1,121,241,361), label=namtime[c(1,121,241,361)]); box()
legend(0,3.5,c("Nino 3 index","Nino 4 index"),col=c("gray70","darkgreen"),lty=1,lwd=3)
abline(h=0, lty=2)
```



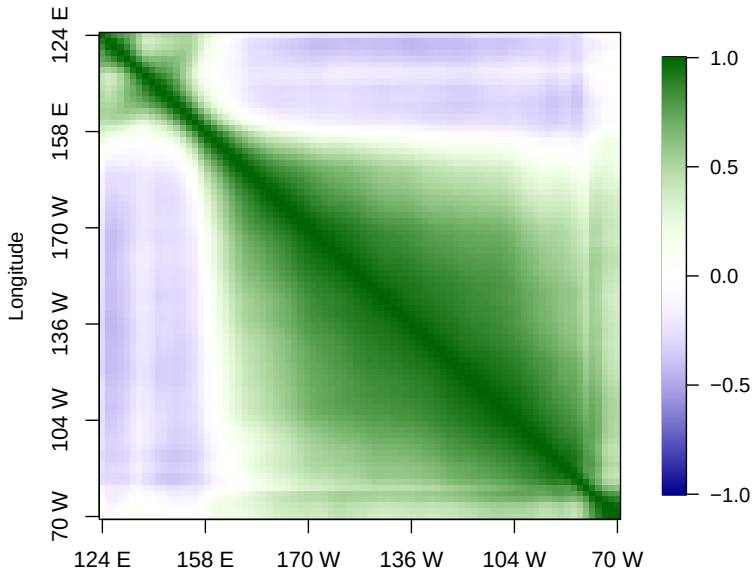
Empirical covariance functions (for latitude 1 S)

```
# Empirical covariance function
Covmat = var(SST3D[, ,15])
Covmat2 <- Covmat[,84:1]
image.plot(Covmat2, add = FALSE, nlevel = 64, legend.shrink = 0.9,
           legend.width = 1.2, legend.mar = NULL, axes=FALSE,
           graphics.reset = FALSE, horizontal = FALSE, bigplot = NULL,
           smallplot = NULL, legend.only = FALSE,
           col = two.colors(n=256, start="darkblue", end="darkgreen",
                           middle="white",alpha=1.0),
           lab.breaks=NULL, axis.args=NULL,
           xlab="Longitude",ylab="Longitude",zlim=c(-2.2,2.2))
box()
axis(1, at=c(0,0.2,0.4,0.6,0.8,1), label=lon[round(seq(1,84,length=6))])
axis(2, at=c(0,0.2,0.4,0.6,0.8,1), label=lon[round(seq(84,1,length=6))])
```



Empirical correlation functions (for latitude 1 S)

```
# Empirical correlation function
D = diag(diag(Covmat)^(-0.5))
Cormat = D%*%Covmat%*%D
Cormat2 <- Cormat[,84:1]
image.plot(Cormat2, add = FALSE, nlevel = 64, legend.shrink = 0.9,
           legend.width = 1.2, legend.mar = NULL, axes=FALSE,
           graphics.reset = FALSE, horizontal = FALSE, bigplot = NULL,
           smallplot = NULL, legend.only = FALSE,
           col = two.colors(n=256, start="darkblue", end="darkgreen",
                           middle="white",alpha=1.0),
           lab.breaks=NULL, axis.args=NULL,
           xlab="Longitude",ylab="Longitude", zlim=c(-1,1))
box()
axis(1, at=c(0,0.2,0.4,0.6,0.8,1), label=lon[round(seq(1,84,length=6))])
axis(2, at=c(0,0.2,0.4,0.6,0.8,1), label=lon[round(seq(84,1,length=6))])
```



Empirical orthogonal function (EOF) analysis

- Empirical orthogonal function (EOF) analysis attempts to find a relatively small number of independent variables (factors) which convey as much of the original information as possible without redundancy.
- EOF analysis can be used to explore the structure of the variability in spatio-temporal data, and to analyze relationships within a set of variables.
- In its discrete formulation, EOF analysis equals principal component analysis (PCA).

Spatially continuous formulation

Let $\{Z_t(\mathbf{s}) : \mathbf{s} \in D, t = 1, 2, \dots\}$ be a continuous spatial process measured at discrete times.

We want to write

$$Z_t(\mathbf{s}) = \sum_{k=1}^{\infty} \alpha_t(k) \phi_k(\mathbf{s}),$$

such that $\text{Var}(\alpha_t(1)) > \text{Var}(\alpha_t(2)) > \dots$ and $\text{Cov}(\alpha_t(i), \alpha_t(k)) = 0$ for all $i \neq k$.

Spatially continuous formulation, part II

Assume $\mathbb{E}(Z_t(\mathbf{s})) = 0$ and set $C_Z^{(0)}(\mathbf{s}, \mathbf{s}') := \mathbb{E}(Z_t(\mathbf{s})Z_t(\mathbf{s}'))$. Then we can write

$$C_Z^{(0)}(\mathbf{s}, \mathbf{s}') = \sum_{k=1}^{\infty} \lambda_k \phi_k(\mathbf{s}) \phi_k(\mathbf{s}'),$$

where $\{\phi_k(\cdot) : k = 1, 2, \dots\}$ are eigenfunctions and $\{\lambda_k : k = 1, 2, \dots\}$ are the associated eigenvalues of the Fredholm integral equation

$$\int_D C_Z^{(0)}(\mathbf{s}, \mathbf{s}') \phi_k(\mathbf{s}) d\mathbf{s} = \lambda_k \phi_k(\mathbf{s}'),$$

where $\int_D \phi_k(\mathbf{s}) \phi_l(\mathbf{s}) d\mathbf{s} = 1$ if $k = l$ and 0 otherwise.

Spatially continuous formulation, part III

Under minor regularity conditions (see Sec. 5.3.1 and 3.5.1), we can then write

$$Z_t(\mathbf{s}) = \sum_{k=1}^{\infty} \alpha_t(k) \phi_k(\mathbf{s}),$$

where we call $\{\phi_k(\mathbf{s}) : \mathbf{s} \in D\}$ the k -th *EOF*, and $\{\alpha_t(k) : t = 1, 2, \dots\}$ the k -th *principal-component*, or *factor loading, time series*.

We have

$$\alpha_t(k) = \int_D Z_t(\mathbf{s}) \phi_k(\mathbf{s}) d\mathbf{s}, \quad t = 1, 2, \dots$$

and $\mathbb{E}(\alpha_t(i) \alpha_t(k)) = \lambda_k \mathbb{1}\{i = k\}$.

Spatially continuous formulation, part IV

If we truncate the expansion at p_α ,

$$Z_{p_\alpha,t}(\mathbf{s}) = \sum_{k=1}^{p_\alpha} \alpha_t(k) \phi_k(\mathbf{s}),$$

it can be shown that the finite EOF decomposition minimizes the variance of the truncation error, $\mathbb{E}(Z_t(\mathbf{s}) - Z_{p_\alpha,t}(\mathbf{s}))^2$, and is thus optimal compared to all other bases (in this regard).

Spatially discrete formulation

In practice, we have discrete data $\mathbf{Z}_t = (Z_t(\mathbf{s}_1), \dots, Z_t(\mathbf{s}_m))'$ and we assume $\text{Cov}[Z_t(\mathbf{s}_i), Z_t(\mathbf{s}_j)] = C_{ij}$ is independent of time.

Can we find a linear combination $a_t = \boldsymbol{\psi}'\mathbf{Z}_t$ describing most of the variability under the restriction $\boldsymbol{\psi}'\boldsymbol{\psi} = 1$?

Spatially discrete formulation, part II

Note that $\text{Var}(a_t) = \boldsymbol{\psi}' \mathbf{C} \boldsymbol{\psi}$, where in practice we use $\widehat{\mathbf{C}} = \sum_t (\mathbf{z}_t - \bar{\mathbf{z}})(\mathbf{z}_t - \bar{\mathbf{z}})'$ with $\bar{\mathbf{z}} = T^{-1} \sum_t \mathbf{z}_t$. Then, we have

- Mathematical formulation: Find $\max \boldsymbol{\psi}' \mathbf{C} \boldsymbol{\psi}$ subject to $\boldsymbol{\psi}' \boldsymbol{\psi} = 1$
- Mathematical solution: $\boldsymbol{\psi}$ is the eigenvector of $\mathbf{C}$ corresponding to the largest eigenvalue, ψ_1 .

Spatially discrete formulation, part II

Note that $\text{Var}(a_t) = \boldsymbol{\psi}' \mathbf{C} \boldsymbol{\psi}$, where in practice we use $\widehat{\mathbf{C}} = \sum_t (\mathbf{Z}_t - \bar{\mathbf{Z}})(\mathbf{Z}_t - \bar{\mathbf{Z}})'$ with $\bar{\mathbf{Z}} = T^{-1} \sum_t \mathbf{Z}_t$. Then, we have

- Mathematical formulation: Find $\max \boldsymbol{\psi}' \mathbf{C} \boldsymbol{\psi}$ subject to $\boldsymbol{\psi}' \boldsymbol{\psi} = 1$
- Mathematical solution: $\boldsymbol{\psi}$ is the eigenvector of $\mathbf{C}$ corresponding to the largest eigenvalue, $\boldsymbol{\psi}_1$.
- Continue: Find $\max \boldsymbol{\psi}' \mathbf{C} \boldsymbol{\psi}$ subject to $\boldsymbol{\psi}' \boldsymbol{\psi} = 1$ and $\boldsymbol{\psi}' \boldsymbol{\psi}_1 = 0$
- Solution: $\boldsymbol{\psi}$ is the eigenvector of $\mathbf{C}$ corresponding to the second largest eigenvalue, $\boldsymbol{\psi}_2$.

Spatially discrete formulation, part II

Note that $\text{Var}(a_t) = \boldsymbol{\psi}' \mathbf{C} \boldsymbol{\psi}$, where in practice we use $\widehat{\mathbf{C}} = \sum_t (\mathbf{Z}_t - \bar{\mathbf{Z}})(\mathbf{Z}_t - \bar{\mathbf{Z}})'$ with $\bar{\mathbf{Z}} = T^{-1} \sum_t \mathbf{Z}_t$. Then, we have

- Mathematical formulation: Find $\max \boldsymbol{\psi}' \mathbf{C} \boldsymbol{\psi}$ subject to $\boldsymbol{\psi}' \boldsymbol{\psi} = 1$
- Mathematical solution: $\boldsymbol{\psi}$ is the eigenvector of $\mathbf{C}$ corresponding to the largest eigenvalue, $\boldsymbol{\psi}_1$.
- Continue: Find $\max \boldsymbol{\psi}' \mathbf{C} \boldsymbol{\psi}$ subject to $\boldsymbol{\psi}' \boldsymbol{\psi} = 1$ and $\boldsymbol{\psi}' \boldsymbol{\psi}_1 = 0$
- Solution: $\boldsymbol{\psi}$ is the eigenvector of $\mathbf{C}$ corresponding to the second largest eigenvalue, $\boldsymbol{\psi}_2$.
- ...

Direct implementation

```
SST = as.matrix(SST)
#EOF spatial, direct calculation (numerically unstable)
Cmat = var(t(SST))
e = eigen(Cmat,symmetric=TRUE)
Spec.covmat = e$vec%*%diag(e$val)%*%t(e$vec)
print(range(Cmat-Spec.covmat))
```

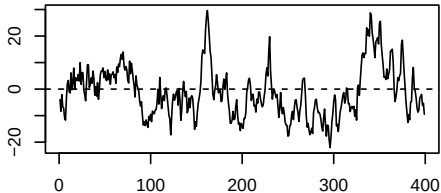
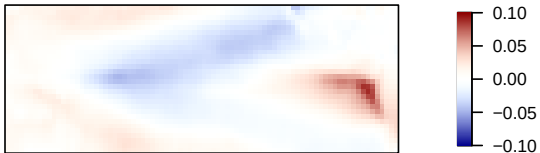
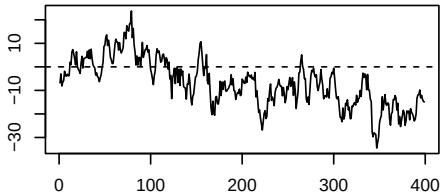
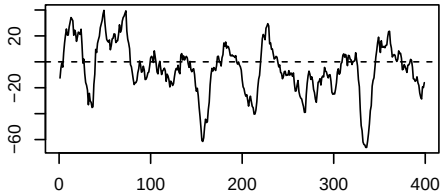
```
[1] -1.245115e-13  1.222911e-13
```

```
A = t(e$vec[,1:4])%*%as.matrix(SST)
```

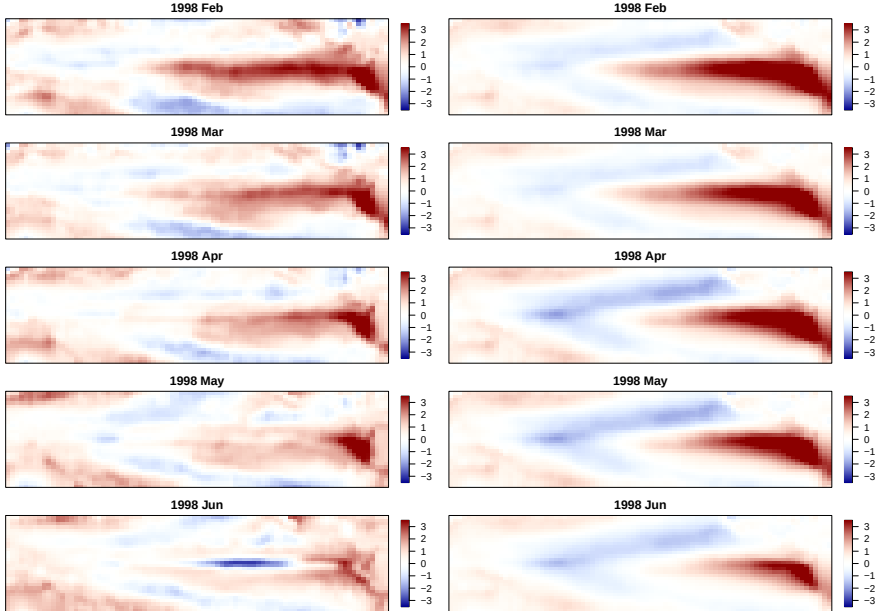
What does this look like?

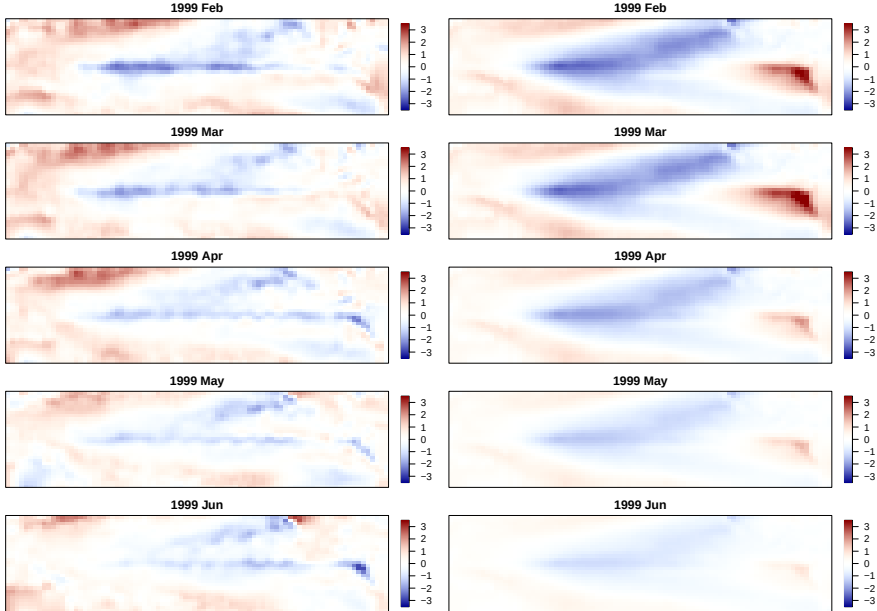
```
par(mfrow=c(3,2), mar=c(2,5,1,5))
for(i in 1:3)
{
  x = matrix(e$vec[,i],ncol=30,byrow=F)
  image.plot(x,add = FALSE, nlevel = 64, legend.shrink = 0.9,
             legend.width = 1.2, legend.mar = NULL, axes=FALSE,
             graphics.reset = FALSE, horizontal = FALSE, bigplot = NULL,
             smallplot = NULL, legend.only = FALSE,
             col = two.colors(n=256, start="darkblue", end="darkred", middle="white",alpha=1.0),
             lab.breaks=NULL, axis.args=NULL,
             xlab="",ylab="", zlim=c(-0.1,0.1))

  box()
  plot.ts(A[i,],ylab="")
  abline(h=0, lty=2)
}
```

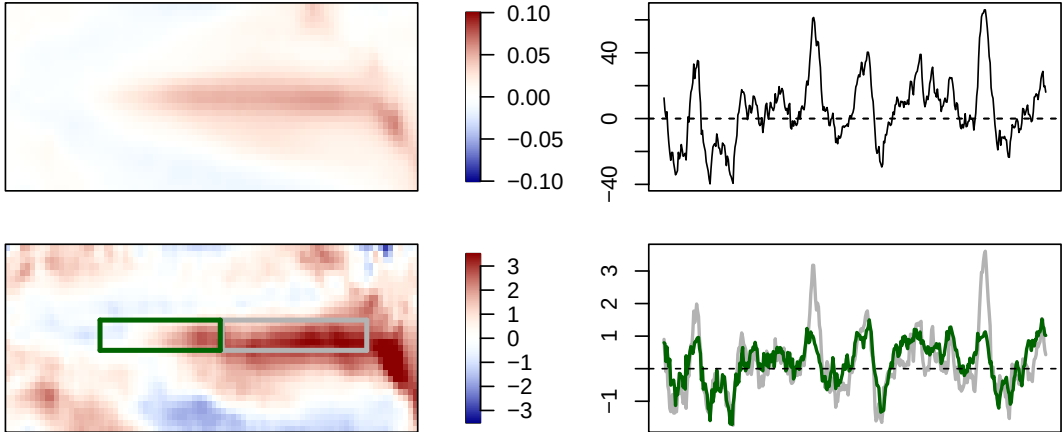


How well does it work?





Interpretation: First EOF vs. Nino index



EOFs when $m > T$

When $M > T$, $\hat{\mathbf{C}} = \sum_t (\mathbf{z}_t - \bar{\mathbf{z}})(\mathbf{z}_t - \bar{\mathbf{z}})'$ only has $T - 1$ non-zero eigenvalues and the calculation of eigenvectors is numerically unstable.

EOFs when $m > T$

When $M > T$, $\hat{\mathbf{C}} = \sum_t (\mathbf{z}_t - \bar{\mathbf{z}})(\mathbf{z}_t - \bar{\mathbf{z}})'$ only has $T - 1$ non-zero eigenvalues and the calculation of eigenvectors is numerically unstable.

From linear algebra: Let $\mathbf{W}$ be $m \times T$ matrix.

Then $\mathbf{W}\mathbf{W}'$ (dim $m \times m$) and $\mathbf{W}'\mathbf{W}$ (dim $T \times T$) have the same eigenvalues.

EOFs when $m > T$

When $M > T$, $\hat{\mathbf{C}} = \sum_t (\mathbf{z}_t - \bar{\mathbf{z}})(\mathbf{z}_t - \bar{\mathbf{z}})'$ only has $T - 1$ non-zero eigenvalues and the calculation of eigenvectors is numerically unstable.

From linear algebra: Let $\mathbf{W}$ be $m \times T$ matrix.

Then $\mathbf{W}\mathbf{W}'$ (dim $m \times m$) and $\mathbf{W}'\mathbf{W}$ (dim $T \times T$) have the same eigenvalues.

If ξ_i is i -th eigenvector of $\mathbf{W}\mathbf{W}'$, then

$$\psi_i = \mathbf{W}\xi_i / \sqrt{\xi_i' \mathbf{W}' \mathbf{W} \xi_i}$$

is the i -th eigenvector of $\mathbf{W}'\mathbf{W}$.

EOFs when $m > T$, part II

Now define

$$\tilde{\mathbf{z}}_t = \mathbf{z}_t - \bar{\mathbf{z}}$$

$$\tilde{\mathbf{Z}} = (\tilde{\mathbf{z}}_1, \dots, \tilde{\mathbf{z}}_T), \quad m \times T \text{ matrix}$$

$$\hat{\mathbf{C}} = \tilde{\mathbf{Z}}\tilde{\mathbf{Z}}', \quad m \times m \text{ matrix}$$

$$\mathbf{A} = \tilde{\mathbf{Z}}'\tilde{\mathbf{Z}}, \quad T \times T \text{ matrix}$$

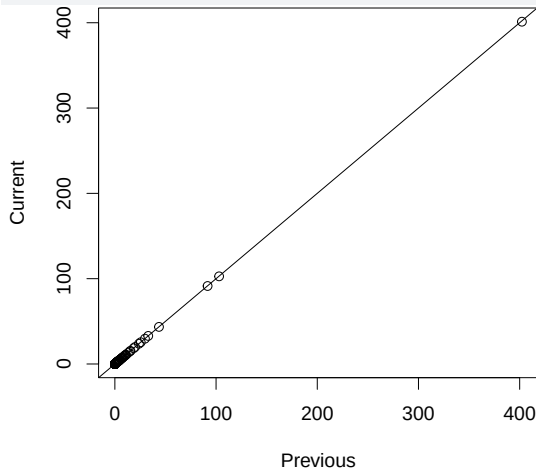
Then we calculate eigenvalues/vectors from $\mathbf{A}$.

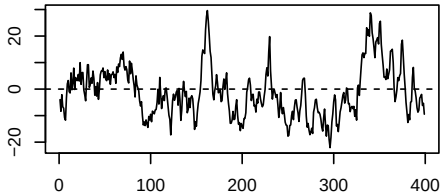
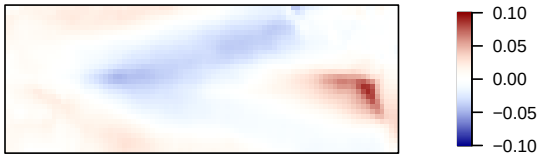
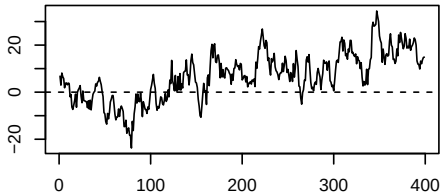
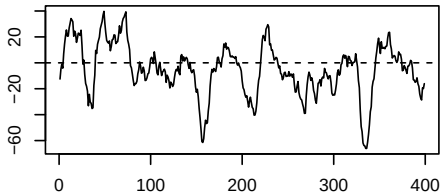
```
#Robust version (see sec 5.3.4)
N = dim(SST)[1]; nT = dim(SST)[2]
mu.hat = rowMeans(SST)
W = (SST-matrix(rep(mu.hat,nT),ncol=nT))/sqrt(nT)
WW = t(W)%*%W
e2 = eigen(WW)
#Checking that last eigenvalues are (approx.) 0
range(e$val[400:N])
```

```
[1] -4.387767e-13  1.865675e-13
```

```
#Calculating eigenvectors
Phi = NULL
for(i in 1:4)
{
  x = W%*%e2$vec[,i]
  phi.i = x/sqrt(sum(x*x))
  Phi = cbind(Phi,phi.i)
}
A = t(Phi)%*%as.matrix(SST)
```

```
#Checking that eigenvalues are equal
par(cex=2)
plot(e$val[1:399],e2$val[1:399], xlab="Previous",
      ylab="Current"); abline(c(0,1))
```

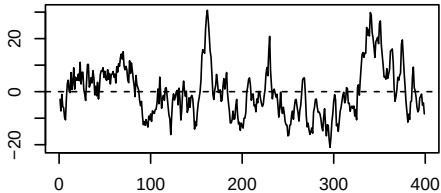
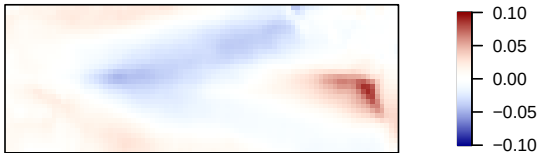
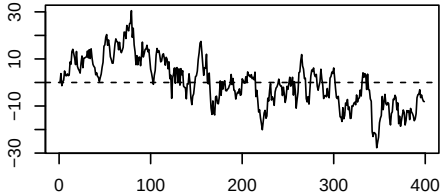
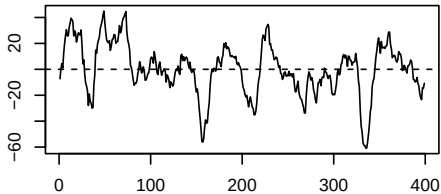




We can also perform the PCA using R functions

```
res <- prcomp(t(SST), retx=TRUE)
par(mfrow=c(3,2), mar=c(2,5,1,5))
for(i in 1:3)
{
  x = matrix(t(res$rotation)[i,],ncol=30,byrow=F)
  image.plot(x,add = FALSE, nlevel = 64, axes=FALSE,
             legend.shrink = 0.9, legend.width = 1.2, legend.mar = NULL,
             graphics.reset = FALSE, horizontal = FALSE, bigplot = NULL,
             smallplot = NULL, legend.only = FALSE,
             col = two.colors(n=256, start="darkblue", end="darkred", middle="white",alpha=1.0),
             lab.breaks=NULL, axis.args=NULL,
             xlab="", ylab="", zlim=c(-0.1,0.1))

  box()
  plot.ts(t(res$x)[i,],ylab="")
  abline(h=0, lty=2)
}
```



We can also perform the PCA using R functions, part II

```
princomp(t(SST))
```

```
Error in princomp.default(t(SST)): 'princomp' can only be used with more units than variables
```

EOFs: Temporal formulation

EOF analysis is usually focused on spatial EOFs as above. However, we can also obtain temporal EOFs.

The data is now $\mathbf{Z}(\mathbf{s}_i) = (Z_1(\mathbf{s}_i), \dots, Z_T(\mathbf{s}_i))'$, where we assume $\text{Cov}[Z_t(\mathbf{s}_i), Z_v(\mathbf{s}_i)] = \Gamma_{tv}$, i.e. independent of space.

Which linear combination $a(\mathbf{s}_i) = \boldsymbol{\psi}'\mathbf{Z}(\mathbf{s}_i)$ describes most of the variability under the restriction $\boldsymbol{\psi}$ to $\boldsymbol{\psi}'\boldsymbol{\psi} = 1$?

Note that $\text{Var}[a(\mathbf{s}_i)] = \boldsymbol{\psi}'\mathbf{\Gamma}\boldsymbol{\psi}$. Solution is thus: $\boldsymbol{\psi}$ is the eigenvector of $\mathbf{\Gamma}$ corresponding to the largest eigenvalue, $\boldsymbol{\psi}_1$. Etc.

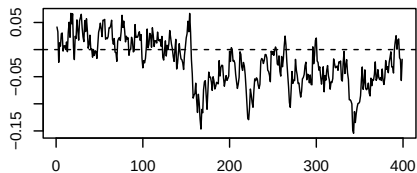
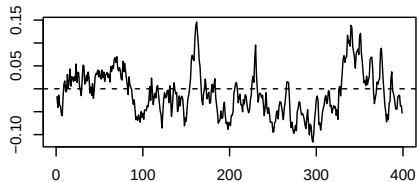
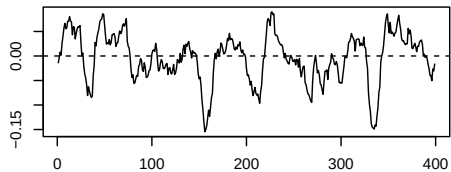
In practice we use $\hat{\mathbf{\Gamma}} = \sum_i (\mathbf{Z}(\mathbf{s}_i) - \bar{\mathbf{Z}})(\mathbf{Z}(\mathbf{s}_i) - \bar{\mathbf{Z}})', \bar{\mathbf{Z}} = m^{-1} \sum_i \mathbf{Z}(\mathbf{s}_i)$.

Implementation

```
#EOF temporal, direct calculation
Cmat = var(SST)
e = eigen(Cmat,symmetric=TRUE)
Spec.covmat = e$vec%*%diag(e$val)%*%t(e$vec)
print(range(Cmat-Spec.covmat))
```

```
[1] -2.384204e-14  9.159340e-15
```

```
A = t(e$vec[,1:4])%*%as.matrix(t(SST))
```



Given a spatio-temporal data set, how do we know if we need to worry about a spatio-temporal structure?

Permutation test for spatio-temporal structure

Testing for spatial structure: Moran's I (sec 4.2):

$$I \equiv \frac{\mathbf{Z}'\mathbf{W}\mathbf{Z}}{\mathbf{Z}'\mathbf{W}} = \frac{\sum_{ij} w_{ij}Z(\mathbf{s}_i)Z(\mathbf{s}_j)}{\sum_i Z(\mathbf{s}_i)^2}$$

Space-Time Index (STI):

$$STI \equiv \frac{\sum_{t=2}^T \sum_{i=1}^m \sum_{j=1}^m w_{ij,t-1} Z(\mathbf{s}_i; t) Z(\mathbf{s}_j; t-1)}{m(T-1) (\sum_{t=2}^T \sum_{i=1}^m \sum_{j=1}^m w_{ij,t-1}) (\sum_{t=1}^T \sum_{i=1}^m Z(\mathbf{s}_i; t)^2)}$$

Testing by permutation (page 303):

- Randomly permute the values of $\{Z(\mathbf{s}_i; t)\}$ and calculate STI
- Compare STI from original data with simulated values

STI for SST data

```
library(Matrix)
SST = read.table("images_ch5/SST011970_032003.dat"); SST = as.matrix(SST)
SSTlonlat = read.table("images_ch5/SSTlonlat.dat", col.names=c("lon", "lat"))
d = as.matrix(dist(SSTlonlat, diag=TRUE, upper=TRUE))
w = 1/d; w[d>4] = 0; diag(w) = 0
wsparse = Matrix(w, sparse=TRUE)
stifunc = function(Z,w) {
  m = dim(Z)[1]
  nT = dim(Z)[2]
  wsum = (nT-1)*sum(w)
  z2sum = sum(Z*Z)
  wzsum = 0
  for(i in 2:nT) {
    zt = matrix(Z[,i], ncol=1)
    ztlag = matrix(Z[,i-1], ncol=1)
    wzsum = wzsum + t(zt)%*%w%*%ztlag
  }
  wzsum/wsum*z2sum
}
```

STI for SST data, part II

```

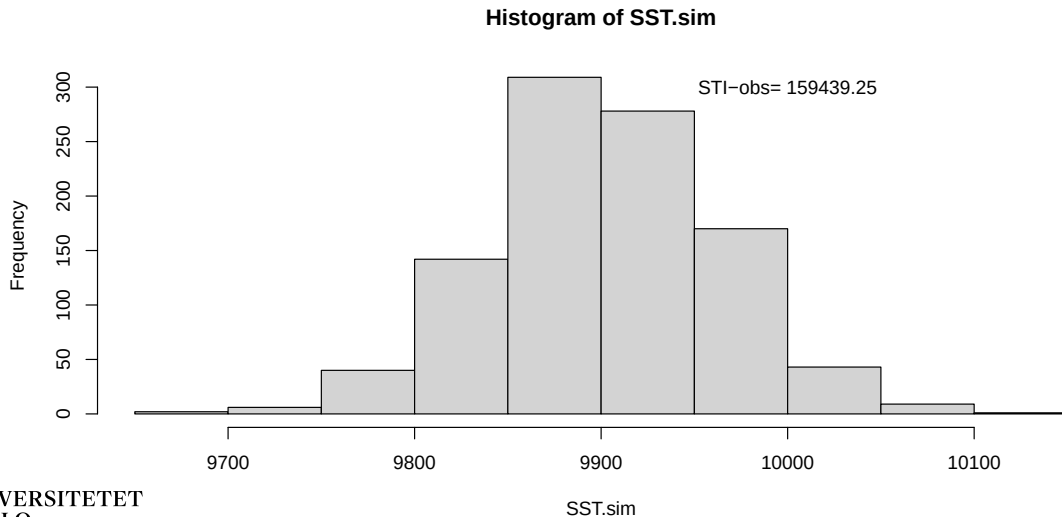
SST.obs = as.numeric(stifunc(SST,wsparse))
m = dim(SST)[1]
nT = dim(SST)[2]
if(0) {
  B = 1000
  SST.sim = rep(NA,B)
  for(b in 1:B) {
    print(b)
    Z = array(sample(SST,nT*m),dim(SST))
    SST.sim[b] = as.numeric(stifunc(Z,wsparse))
  }
}
SST.sim = scan("../r/SST_sim.txt")
hist(SST.sim); abline(h=SST.obs)
text(10000,300,paste("STI-obs=",round(SST.obs,2)))

#P-value
mean(SST.sim>SST.obs)

```

[1] 0

STI for SST data, part III



Near and far

In general, nearby values tend to be more alike than those far apart

Now: “Near” both in space and time

Time $\neq$ Space: Order in time, no order in space

The book uses *conditional* distributions of the type

$$[Y(\mathbf{s}; t) | \{Y(\mathbf{x}; r) : \mathbf{x} \in D_s, r < t\} \cup \{Y(\mathbf{x}, t) : \mathbf{x} \neq s\}]$$

Modelling strategies

- *Spatio-temporal covariance functions* (sec 6.1,6.2)
- Stochastic differential/difference equations (sec 6.3)
- *Time series of spatial processes* (sec 6.4)

Spatio-temporal covariance functions

Our general model is

$$Y(\mathbf{s}; t) = \mu(\mathbf{s}; t) + \beta(\mathbf{s}) + \gamma(t) + \kappa(\mathbf{s}; t) + \delta(\mathbf{s}; t), \quad \mathbf{s} \in D_s, t \in D_t$$

where

$\mu(\mathbf{s}; t)$ = fixed (covariate) term

$$\text{cov}[\beta(\mathbf{s}), \beta(\mathbf{x})] := C_\beta(\mathbf{s}, \mathbf{x})$$

$$\text{cov}[\gamma(t), \gamma(r)] := C_\gamma(t, r)$$

$$\text{cov}[\kappa(\mathbf{s}; t), \kappa(\mathbf{x}; r)] := C_\kappa(\mathbf{s}, \mathbf{x}; t, r)$$

$$\text{cov}[\delta(\mathbf{s}; t), \delta(\mathbf{x}; r)] := \sigma_\delta^2 I(\mathbf{s} = \mathbf{x}, t = r)$$

Many important special cases. Main focus now: Specification of $C_\kappa(\mathbf{s}, \mathbf{x}; t, r)$.

Spatio-temporal covariance functions, part II

Positive definiteness:

$$\text{var}\left[\sum_i a_i Y(s_i, t_i)\right] \geq 0 \text{ implies } \sum_i \sum_j a_i a_j C((\mathbf{s}_i, t_i), (\mathbf{s}_j, t_j)) \geq 0$$

A function $C(\cdot)$ satisfying this for any $\{Y(s_i, t_i)\}$ and all $\{a_i\}$ is *positive definite*.

Stationarity:

$$\begin{aligned} \text{cov}[Y(\mathbf{s}; t), Y(\mathbf{x}; r)] &= C(\mathbf{s} - \mathbf{x}, t, r), & \text{spatial stationarity} \\ \text{cov}[Y(\mathbf{s}; t), Y(\mathbf{x}; r)] &= C(\mathbf{s}, \mathbf{x}, t - r), & \text{temporal stationarity} \\ \text{cov}[Y(\mathbf{s}; t), Y(\mathbf{x}; r)] &= C(\mathbf{s} - \mathbf{x}, t - r), & \text{spatio-temporal stationarity} \end{aligned}$$

Constructing positive definite functions is difficult (even under stationarity).

Separability

$$\text{cov}[Y(\mathbf{s}; t), Y(\mathbf{x}; r)] = C(\mathbf{s}, t, \mathbf{x}, r) = C^{(s)}(\mathbf{s}, \mathbf{x}) \cdot C^{(t)}(t, r)$$

If $C^{(s)}(\mathbf{s}, \mathbf{x})$ is a valid spatial covariance function and $C^{(t)}(t, r)$ is a valid temporal covariance function, then the product is also valid.

Assuming stationarity, we get

$$C(\mathbf{s}, t, \mathbf{x}, r) = C(\mathbf{s} - \mathbf{x}, t - r) = C^{(s)}(\mathbf{s} - \mathbf{x}) \cdot C^{(t)}(t - r)$$

which gives

$$\rho(\mathbf{h}; \tau) = C(\mathbf{h}; \tau) / C(\mathbf{0}; 0) = \rho(\mathbf{h}; 0) \rho(\mathbf{0}; \tau)$$

That is, $\rho(\cdot; \tau)$ is *proportional* to $\rho(\cdot; 0)$ for all τ . This is commonly used because of computational simplifications.

Separability and computation

Assume $C(\mathbf{s} - \mathbf{x}, t - r) = C^{(s)}(\mathbf{s} - \mathbf{x}) \cdot C^{(t)}(t - r)$, with $C^{(t)}(0) = 1$

Assume we have observed $\{Y(s_i, t_j), i = 1, \dots, m, j = 1, \dots, n\}$. Then,

$$\text{Cov}[\mathbf{Y}] = \Sigma_0 \otimes \Gamma_0$$

$$\Sigma_0 = \text{Var}[\mathbf{Y}_j]$$

$$\Gamma_0 = \text{Var}[(Y(s_i, t_1), \dots, Y(s_i, t_n))']$$

This implies

$$\mathbf{Q} = \Sigma_0^{-1} \otimes \Gamma_0^{-1}$$

That is, we only need to invert

- Σ_0 , an $m \times m$ matrix
- Γ_0 , an $n \times n$ matrix

Direct computation: Need to work with an $nm \times nm$ matrix

Additive structures

$$Y(\mathbf{s}; t) = \beta(\mathbf{s}) + \gamma(t)$$

$$\text{cov}[Y(\mathbf{s}; t), Y(\mathbf{x}; r)] = C_\beta(\mathbf{s}, \mathbf{x}) + C_\gamma(t, r)$$

Typically used as part of a larger model, for instance,

$$Y(\mathbf{s}; t) = \mu(\mathbf{s}; t) + \beta(\mathbf{s}) + \gamma(t) + \kappa(\mathbf{s}; t) + \delta(\mathbf{s}; t),$$

In general, we have that

$$\sum_j w_j C^j(\mathbf{s}, t; \mathbf{x}, r), w_j > 0$$

is a valid covariance function if each $C^j(\mathbf{s}, t; \mathbf{x}, r)$ is valid.

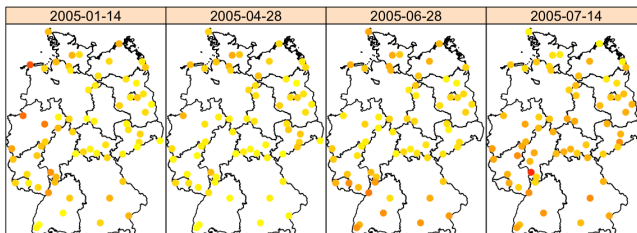
Additive structures, part II

Example: Adding two separable covariance functions

$$pC^{s1}(\mathbf{h})C^{t1}(\tau) + qC^{s2}(\mathbf{h})C^{t2}(\tau)$$

gives a *non-separable* covariance function.

Example: PM10 values in Germany



```

library(spacetime)
library(gstat)
library(sp)
rm(list=ls())
data(air)
ls()[1:3]

[1] "air"    "dates" "DE"

ls()[4:5]

[1] "DE_NUTS1" "stations"

rural = STFDF(stations, dates,
              data.frame(PM10 = as.vector(air)))
rr = rural[, "2005::2010"]
unsel = which(apply(as(rr, "xts"), 2,
                  function(x) all(is.na(x))))
r5to10 = rr[-unsel,]

```

```
summary(r5to10)
```

```

Object of class STFDF
  with Dimensions (s, t, attr): (53, 1826, 1)
[[Spatial:]]
Object of class SpatialPoints
Coordinates:
      min      max
coords.x1  6.28107 14.78617
coords.x2 47.80847 54.92497
Is projected: FALSE
proj4string : [+proj=longlat +datum=WGS84 +no_defs]
Number of points: 53
[[Temporal:]]
      Index      timeIndex
Min.   :2005-01-01  Min.    :2558
1st Qu.:2006-04-02  1st Qu. :3014
Median :2007-07-02  Median  :3470
Mean   :2007-07-02  Mean    :3470
3rd Qu.:2008-09-30  3rd Qu. :3927
Max.   :2009-12-31  Max.    :4383
[[Data attributes:]]
      PM10
Min.   : 0.560
1st Qu.: 9.275
Median :13.852
Mean   :16.261
3rd Qu.:20.333
Max.   :269.079
NA's   :21979

```

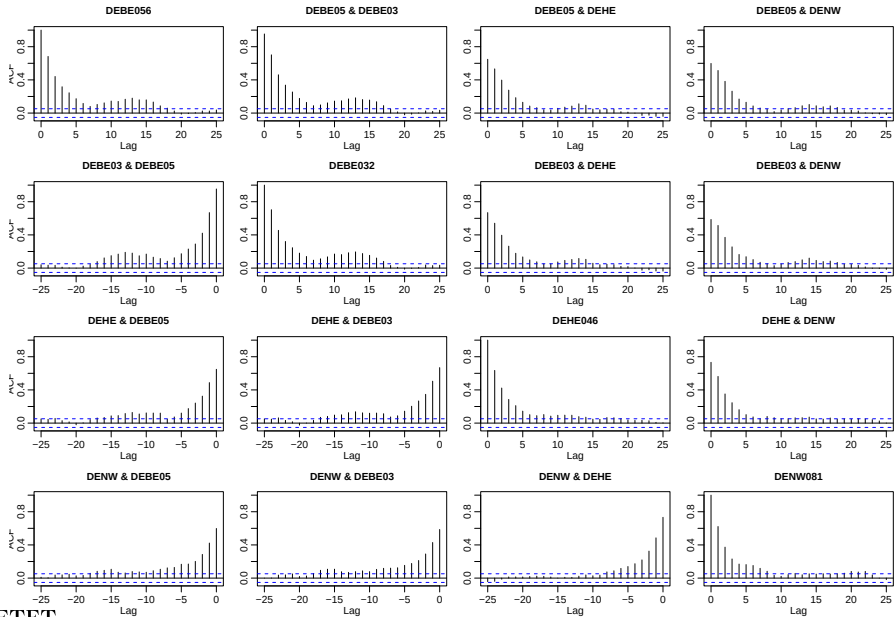
```
rn = row.names(r5to10@sp)[4:7]
rn
```

```
[1] "DEBE056" "DEBE032" "DEHE046" "DENW081"
```

```
print(spDists(r5to10[4:7,]@sp), digits=3)
```

```
      [,1] [,2] [,3] [,4]
[1,]  0.0 28.8 344 468
[2,] 28.8  0.0 317 440
[3,] 343.6 317.3  0 150
[4,] 467.9 439.7 150  0
```

```
acf(na.omit(as(r5to10[rn,], "xts")))
```



Some observations

- Autocorrelations for lag 0 are always 1
- Cross-correlations for lag 0 are not always 1
- Cross-correlations can be asymmetric

Here, the asymmetry is not very strong, but the cross-correlations are fairly strong and of similar form as the autocorrelations.

Spatio-temporal variogram

The *spatio-temporal variogram* of the process $Y(\cdot; \cdot)$ is

$$2\gamma(\mathbf{s}, \mathbf{x}; t, r) := \text{Var}(Y(\mathbf{s}; t) - Y(\mathbf{x}; r)),$$

with the stationary version given by $2\gamma(\mathbf{h}; \tau)$ for $\mathbf{h} \in \mathbb{R}^2, \tau \in \mathbb{R}$.

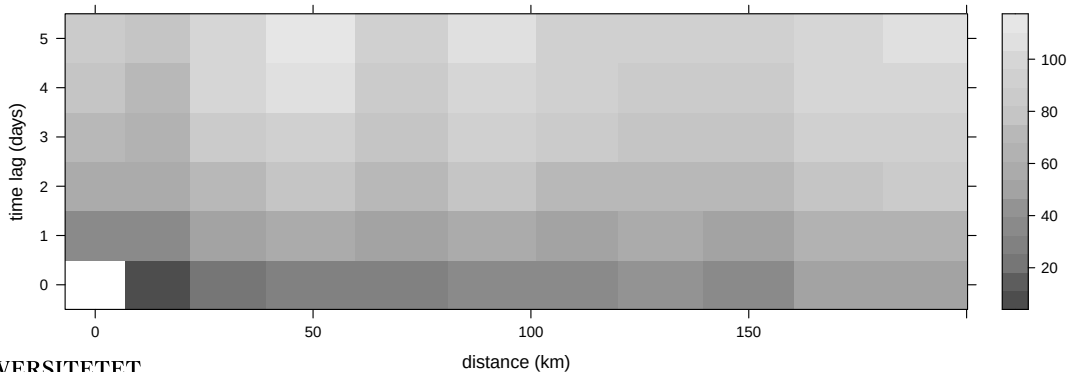
The process $Y(\cdot; \cdot)$ is *intrinsically stationary*, if it has constant mean and a stationary variogram.

If $Y(\cdot; \cdot)$ is second-order stationary with (stationary) covariance function $C(\mathbf{h}; \tau)$, then $Y(\cdot; \cdot)$ is intrinsically stationary with semivariogram

$$\gamma(\mathbf{h}; \tau) := C(\mathbf{0}; 0) - C(\mathbf{h}; \tau), \quad \mathbf{h} \in \mathbb{R}^2, \tau \in \mathbb{R}.$$

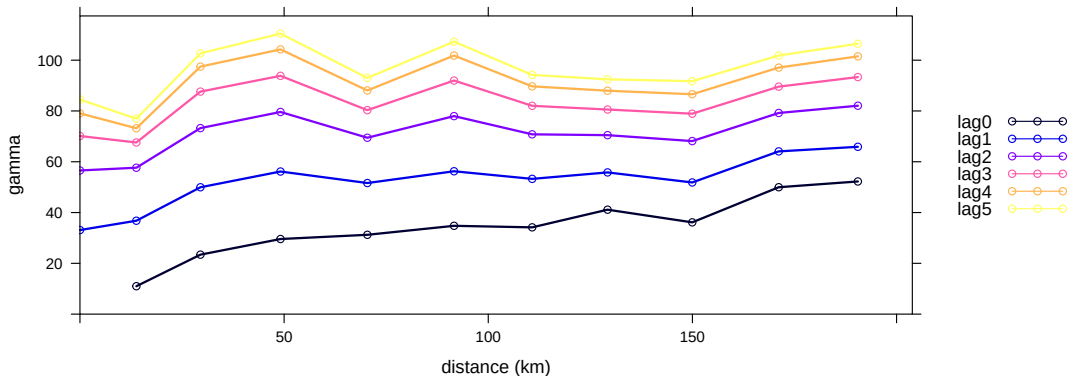
Empirical variogram

```
library(grDevices)
## vv <- variogram(PM10~1, r5to10, width=20, cutoff=200, tlags=0:5)
load("images_ch5/vv.RData")
plot(vv, col=grey.colors(20))
```



Empirical variogram, part II

```
plot(vv, map=FALSE, lwd=2)
```



Estimation of variograms

```
metricVgm <- vgmST("metric", joint=vgm(50,"Exp",100,0), stAni=50)
metricVgm <- fit.StVariogram(vv, metricVgm)
extractPar(metricVgm)
```

```
      sill      range      nugget      anis
96.766513 304.400266  9.448679 141.373317
```

```
sepVgm <- vgmST("separable", space=vgm(0.9,"Exp", 123, 0.1),
               time=vgm(0.9,"Exp", 2.9, 0.1), sill=100)
sepVgm <- fit.StVariogram(vv, sepVgm, method = "L-BFGS-B",
                          lower = c(10,0,0.01,0,1), upper = c(500,1,20,1,200))
extractPar(sepVgm)
```

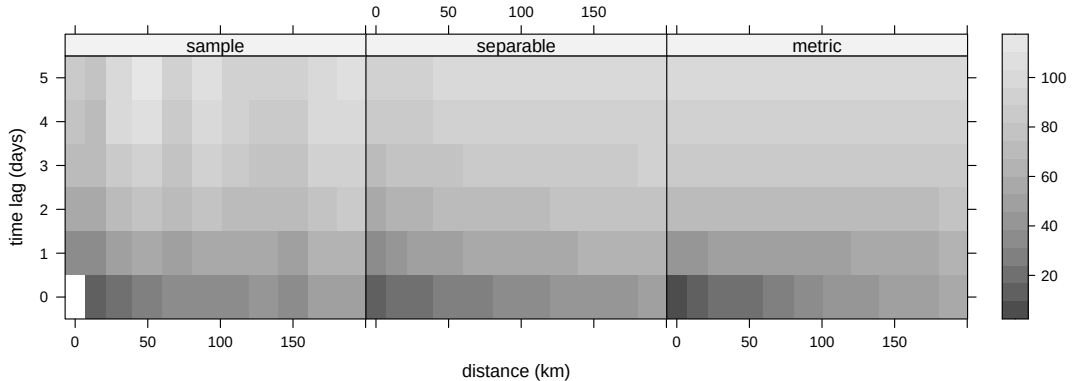
```
      range.s      nugget.s      range.t      nugget.t      sill
479.0029905  0.1410691  3.0414757  0.0000000 113.2907258
```

```
c(attr(metricVgm, "optim")$value, attr(sepVgm, "optim")$value) ## the RMSE between the surfaces
```

```
[1] 60.60356 46.38986
```

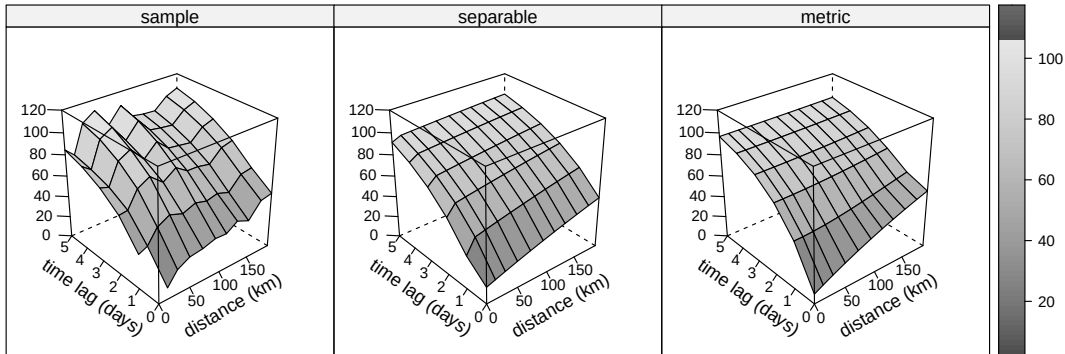
Resulting estimates

```
plot(vv, list(sepVgm, metricVgm), all=T, col=grey.colors(64))
```



Resulting estimates, part II

```
library(lattice)
plot(vv, list(sepVgm, metricVgm), all=T, wireframe=T, zlim=c(0,120),
      xlab=list("distance (km)", rot=30), ylab=list("time lag (days)", rot=-35),
      scales=list(arrows=F, z = list(distance = 5)), col=grey.colors(64))
```



Spatio-temporal kriging

Assume observed $\{Z(\mathbf{s}_i); t_{ij}, i = 1, \dots, m, j = 1, \dots, T_i\}$

$$Z(\mathbf{s}_i; t_{ij}) = Y(\mathbf{s}_i; t_{ij}) + \varepsilon(\mathbf{s}_i; t_{ij})$$

Aim: Predict $Y(\mathbf{s}_0; t_0)$

- Mathematics as for pure spatial processes
- Notationally more complex
- Matrices involved much larger

In general

$$[Y(\mathbf{s}_0; t_0) | \mathbf{Z}] \sim N(\mu(\mathbf{s}_0; t_0) + \mathbf{c}_0' \mathbf{C}_Z^{-1} (\mathbf{Z} - \boldsymbol{\mu}), C_{0,0} - \mathbf{c}_0' \mathbf{C}_Z^{-1} \mathbf{c}_0)$$

Separable covariance functions: $\mathbf{C}_Z$ simplifies considerable (if $T_i = T$ and $t_{ij} = t_j$)

Similar extensions when parameters are to be estimated

Time series of spatial processes

Assume $\mathbf{Y}_t(\cdot) = \{Y(\mathbf{s}; t), \mathbf{s} \in D_s\}$ with $t \in D_t$, $D_t = \{0, 1, 2, \dots\}$.

That is, possibly infinite dimensional space, discrete time.

AR-type model:

$$\mathbf{Y}_t(\mathbf{s}) = \mathcal{M}_t(\mathbf{s}, \mathbf{Y}_{t-1}) + \delta_t(\mathbf{s})$$

$\{\delta_t(\mathbf{s})\}$ independent in time.

Special case is

$$\mathbf{Y}_t(\mathbf{s}) = \int_{R^d} m(\mathbf{s}, \mathbf{x}) \mathbf{Y}_{t-1}(\mathbf{x}) d\mathbf{x} + \delta_t(\mathbf{s})$$

- Similar to discretized models considered earlier
- Now: Perform modelling directly as AR-type models

Linear spatial time series

$$\mathbf{Y}_t = \mathbf{M}_t \mathbf{Y}_{t-1} + \boldsymbol{\delta}_t$$

gives

$$\begin{aligned}\boldsymbol{\Sigma}_{Y,t} &= \text{Var}(\mathbf{Y}_t) = \mathbf{M}_t \boldsymbol{\Sigma}_{Y,t-1} \mathbf{M}_t' + \boldsymbol{\Sigma}_{\delta,t} \\ \text{cov}(\mathbf{Y}_{t+\tau}, \mathbf{Y}_t) &= \mathbf{M}_{t+\tau} \mathbf{M}_{t+\tau-1} \cdots \mathbf{M}_{t+1} \boldsymbol{\Sigma}_{Y,t}\end{aligned}$$

Allows for

- nonstationarity in time ($\mathbf{M}_t$ changes with time)
- nonstationarity in space ($M_t(\mathbf{s}, \mathbf{x})$ depends on spatial locations)

Higher order models

In general

$$[\mathbf{Y}_0, \mathbf{Y}_1, \dots, \mathbf{Y}_T] = [\mathbf{Y}_0][\mathbf{Y}_1|\mathbf{Y}_0][\mathbf{Y}_2|\mathbf{Y}_0, \mathbf{Y}_1] \cdots [\mathbf{Y}_T|\mathbf{Y}_0, \dots, \mathbf{Y}_{T-1}]$$

Need simplifications for

- simpler modelling
- possibility of estimating parameters
- computation

Typical simplifications:

- $[\mathbf{Y}_t|\mathbf{Y}_{t-1}, \dots, \mathbf{Y}_0] = [\mathbf{Y}_t|\mathbf{Y}_{t-1}, \dots, \mathbf{Y}_{t-r}]$ (vector AR-structure)
- $[\mathbf{Y}_t|\mathbf{Y}_{t-1}, \dots, \mathbf{Y}_{t-r}] = [\mathbf{Y}_r|\mathbf{Y}_{r-1}, \dots, \mathbf{Y}_0]$ (stationarity in time)
- spatial sparsity in transitions
- stationarity in space

Vector autoregressive model

Linear model

$$\mathbf{Y}_t = \mathbf{M}_t \mathbf{Y}_{t-1} + \boldsymbol{\delta}_t$$

In general $T \times n^2$ parameters in the $\mathbf{M}$'s.

Simplifications:

- $\mathbf{M}_t = \mathbf{M}$ (vector autoregressive model)
- $M_{ij} = 0$ for $\|\mathbf{s}_i - \mathbf{s}_j\| > h$

Similarly:

- $\boldsymbol{\Sigma}_{\delta,t} = \boldsymbol{\Sigma}_{\delta}$
- $\boldsymbol{\Sigma}_{\delta}$ sparse

References

Pebesma, E. and Gräler, B. (2023) *Introduction to Spatio-Temporal Variography*. Vignette for the R-package `gstat`. Available at <https://cran.r-project.org/web/packages/gstat/vignettes/st.pdf>.