Spatial Data Analysis

Week 12: Spatiotemporal Modeling, Machine Learning

Meredith Franklin

Department of Statistical Sciences and School of the Environment

November 21, 2025

Outline

- Spatio-Temporal processes
  - ST Variograms, ST Kriging
  - ST GAM
  - ST SAR, CAR
- Machine Learning

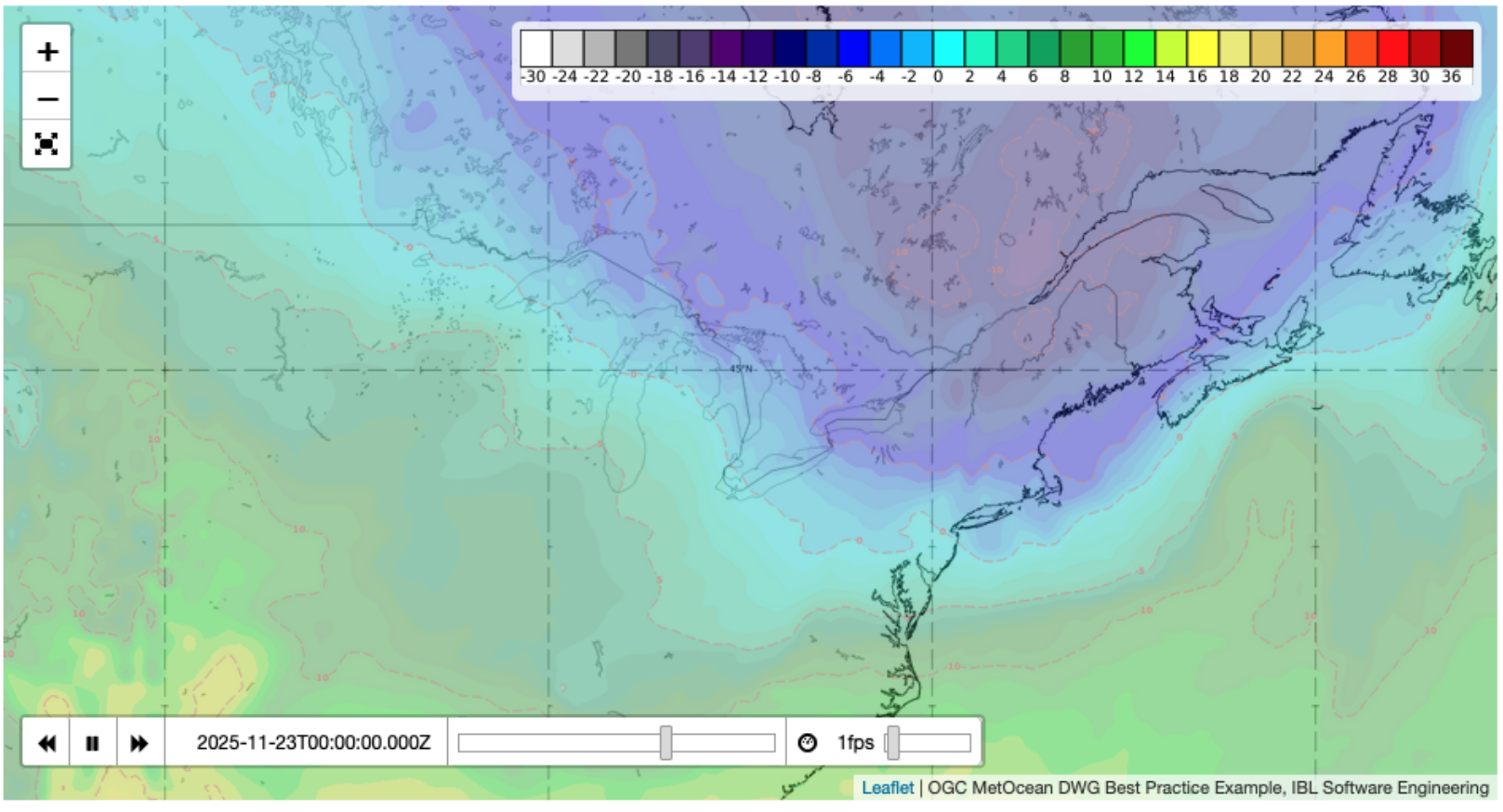
Spatio-Temporal Processes

- Many data that we deal with in spatial statistics are actually both spatial and temporal. Some examples include:
  - $PM_{2.5}$ concentrations over the north east, daily for 1 year.
  - Trends in weather, measured at weather stations, over the last 50 years.
  - Land use changes over time.
  - Housing prices over the last decade.
- We extend the definition of a stochastic spatial process to a stochastic spatio-temporal process:

$$\{Z(s,t) : s \in D(t) \subset \mathbb{R}^2, t \in T\}$$

- The observation of attribute Z at location s is expanded to it's added temporal "location" at time t.
- The dependence of the domain D on time symbolizes the condition where the spatial domain changes over time.
- For simplicity, we may assume no spatial-temporal interaction, ie that $D(t) = D$.

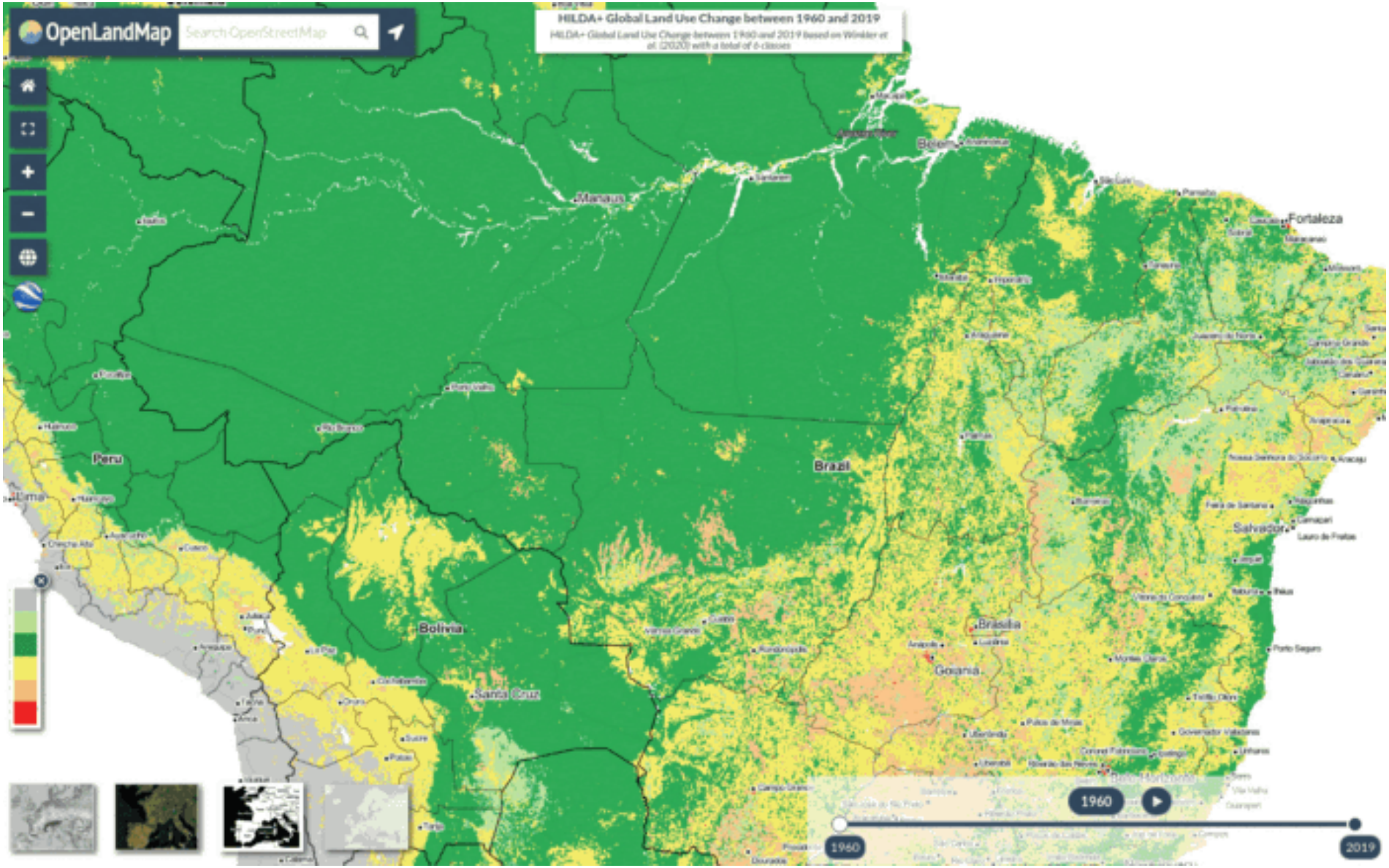
Visualizing Spatio Temporal Data



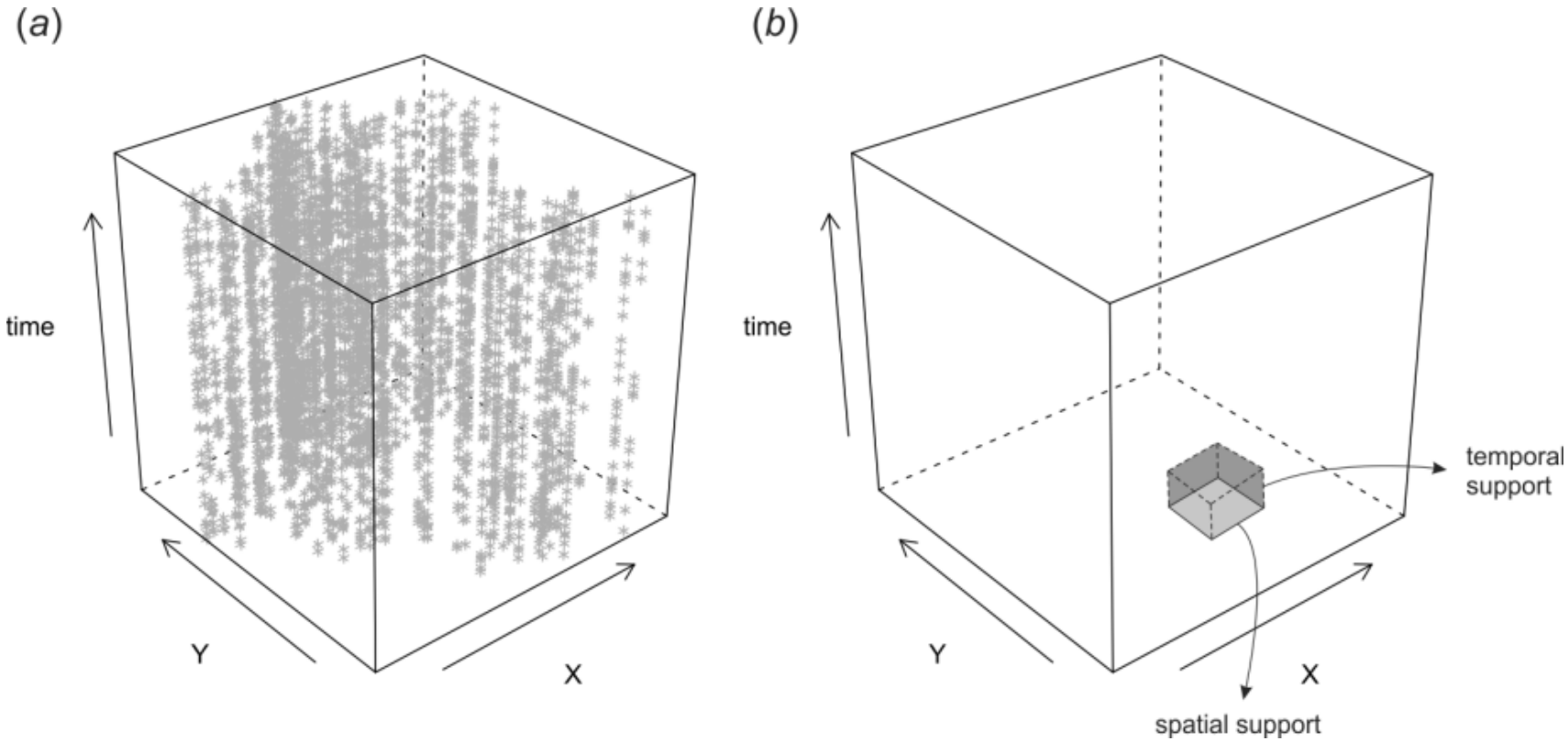
From https://apps.socib.es/Leaflet.TimeDimension/examples/example2.html

Can display a map by temporal frames, add animation, or a panel of maps.

Visualizing Spatio Temporal Data



Spatio-Temporal Data: Structures & Visualization



Left shows locations of meteorological sites (x,y) with time on the z-axis, the right shows how the data are "blocked" into the spatial and temporal support.

Spatio-Temporal Geostatistical Processes

- As we saw with spatial data, we can model the spatio-temporal structure in either the covariance (semivariograms, kriging) or in the mean (gam, ML).
- Modeling spatio-temporal structure in the covariance leads to kriging-type formulations.
- Most work is in defining valid spatio-temporal covariance models: the classes of these types of models is much more limited than spatial covariance.
- The goal of space-time kriging is to make predictions at unobserved space-time points.
- Statistical strength is borrowed over space and time to give efficient predictions and standard errors.

Spatio-Temporal Geostatistics: Semivariograms

- We can extend our semivariogram to include a temporal component. For a stationary process:

$$\begin{aligned}\gamma(h,k) &= \frac{1}{2}Var(Z(s,t) - Z(s+h,t+k)) \\ &= \frac{1}{2}E((Z(s,t) - Z(s+h,t+k))^2)\end{aligned}$$

- We can bin this:

$$\hat{\gamma}(h,k) = \frac{1}{2N(h,k)} \sum_{N(h,k)} (Z(s_i,t) - Z(s_j,u))^2$$

Spatio-Temporal Geostatistics: Covariance

- For a **second-order stationary** process with constant mean and finite variance, the space–time semivariogram is

$$\gamma(h,k) = \tfrac{1}{2}\,\mathrm{Var}\{Z(s,t) - Z(s+h,t+k)\}\,.$$

- Let $C(h,k) = \mathrm{Cov}\{Z(s,t), Z(s+h,t+k)\}$ with $C(0,0) = \mathrm{Var}(Z)$. Then

$$\gamma(h,k) = C(0,0) - C(h,k).$$

- **Spatial/temporal marginals:** $\gamma_s(h) = \gamma(h,0)$, $\gamma_t(k) = \gamma(0,k)$.
- If the covariance is **separable**, $C(h,k) = C_s(h)\,C_t(k)$, then

$$\gamma(h,k) = C_s(0)\,C_t(0) - C_s(h)\,C_t(k).$$

- If only **intrinsic stationarity** holds, $\gamma(h,k)$ is still well-defined but a covariance $C(h,k)$ need not exist; work directly with $\gamma(h,k)$.

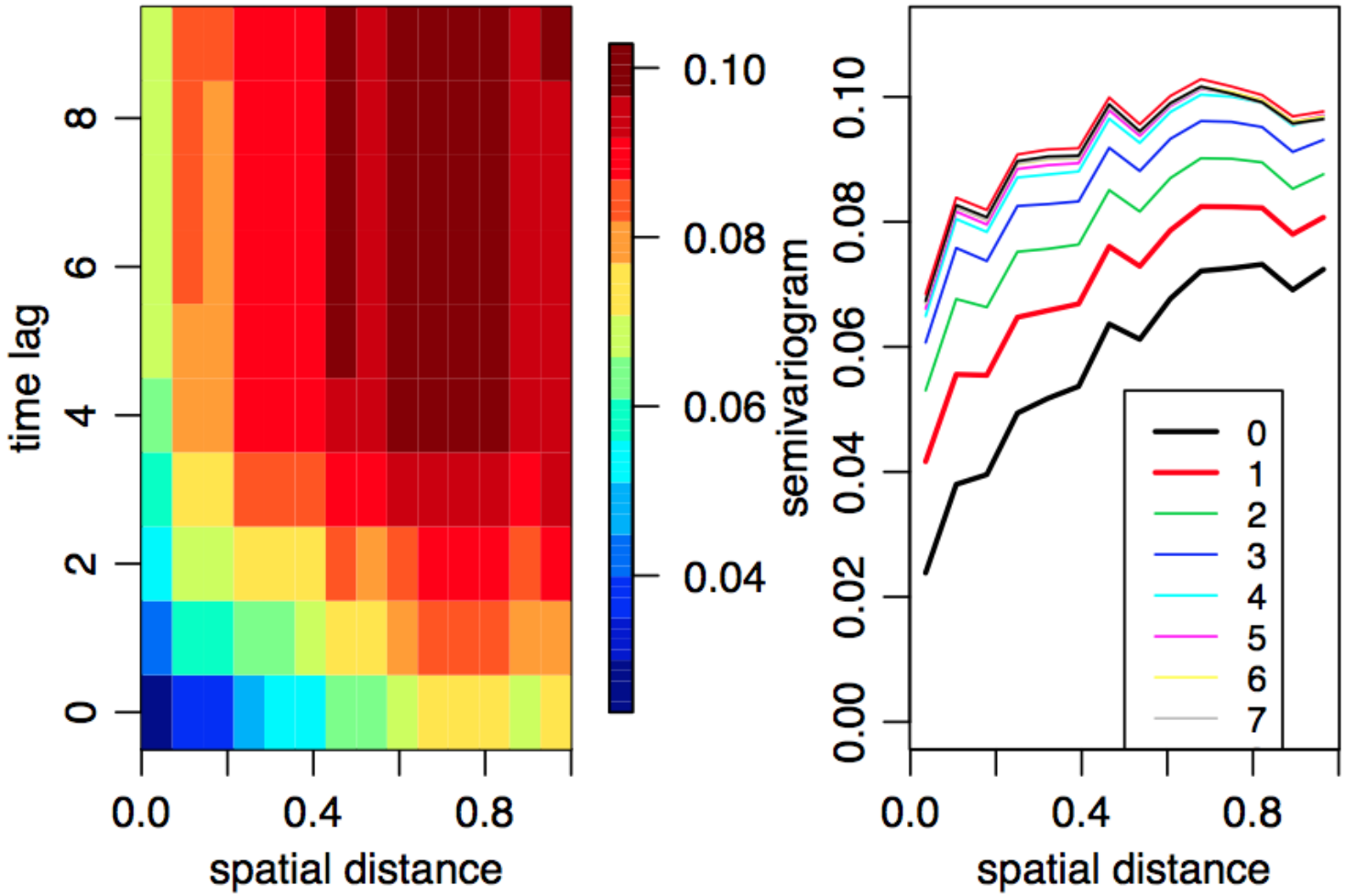
Empirical Space–Time Semivariogram (estimation)

$$\hat{\gamma}(h,k) = \frac{1}{2\,|N(h,k)|} \sum_{(i,j)\in N(h,k)} \big[Z(s_i,t_i) - Z(s_j,t_j)\big]^2,$$

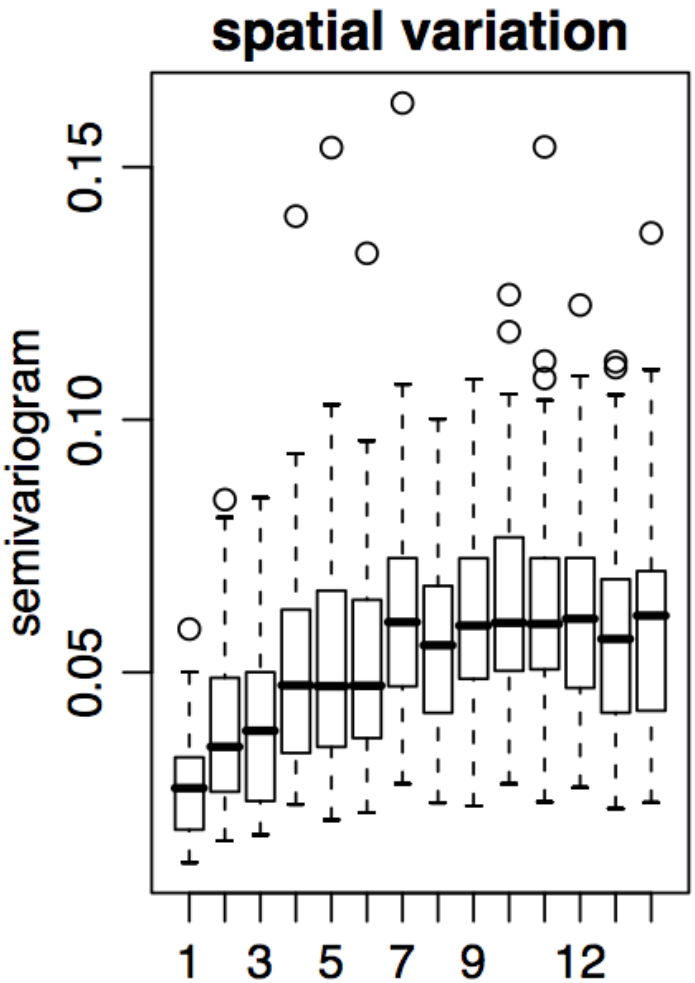
where $N(h,k)$ are pairs whose spatial/temporal lags fall in bins around (h,k).

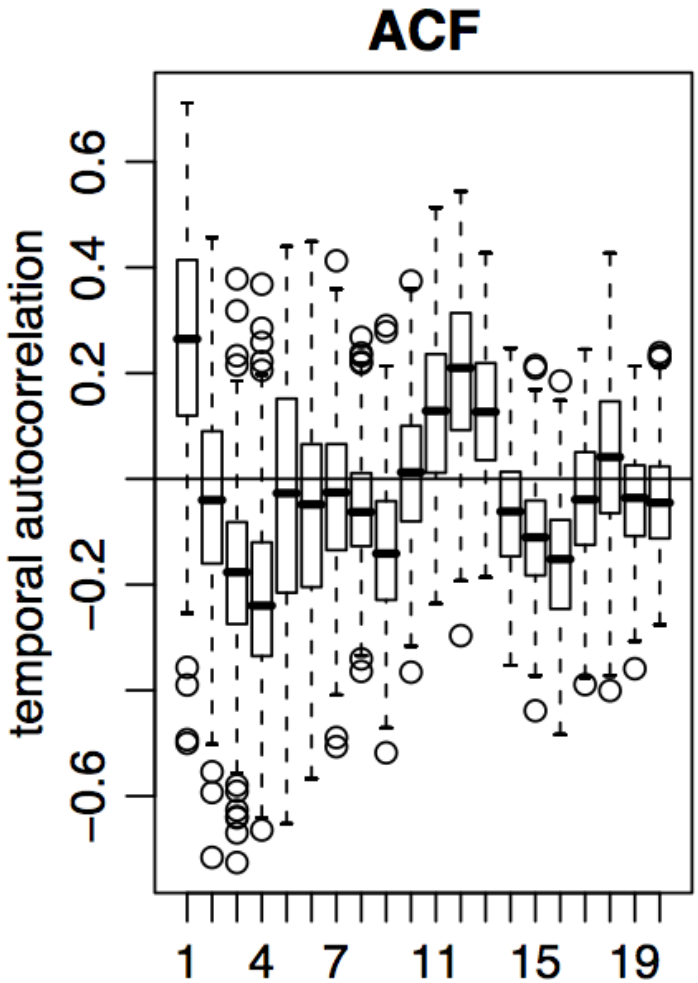
- Use marginal binning in h and k (e.g., distance classes & time lags), or a joint metric $m(h,k) = \sqrt{h^2 + (ak)^2}$.
- Robust alternative: Cressie–Hawkins estimator for outlier resistance.
- Visualize $\hat{\gamma}(h,k)$ as *trace variograms* (k fixed, vary h) and *temporal variograms* (h fixed, vary k).

Spatio-Temporal Geostatistics: $PM_{2.5}$ Example



Spatio-Temporal Geostatistics: $PM_{2.5}$ Example





Spatio-Temporal Covariance Functions

- For a **second-order stationary** process in space and time with **spatial isotropy**:

$$C\{(s,t),(s',u)\} = C(h,k), \quad h = \|s - s'\|,\ k = t - u.$$

- $C(h,k)$ must be (semi)positive definite: for any $a_1, \ldots, a_n$,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j\, C(h_{ij}, k_{ij}) \geq 0.$$

- **Separable product model:**

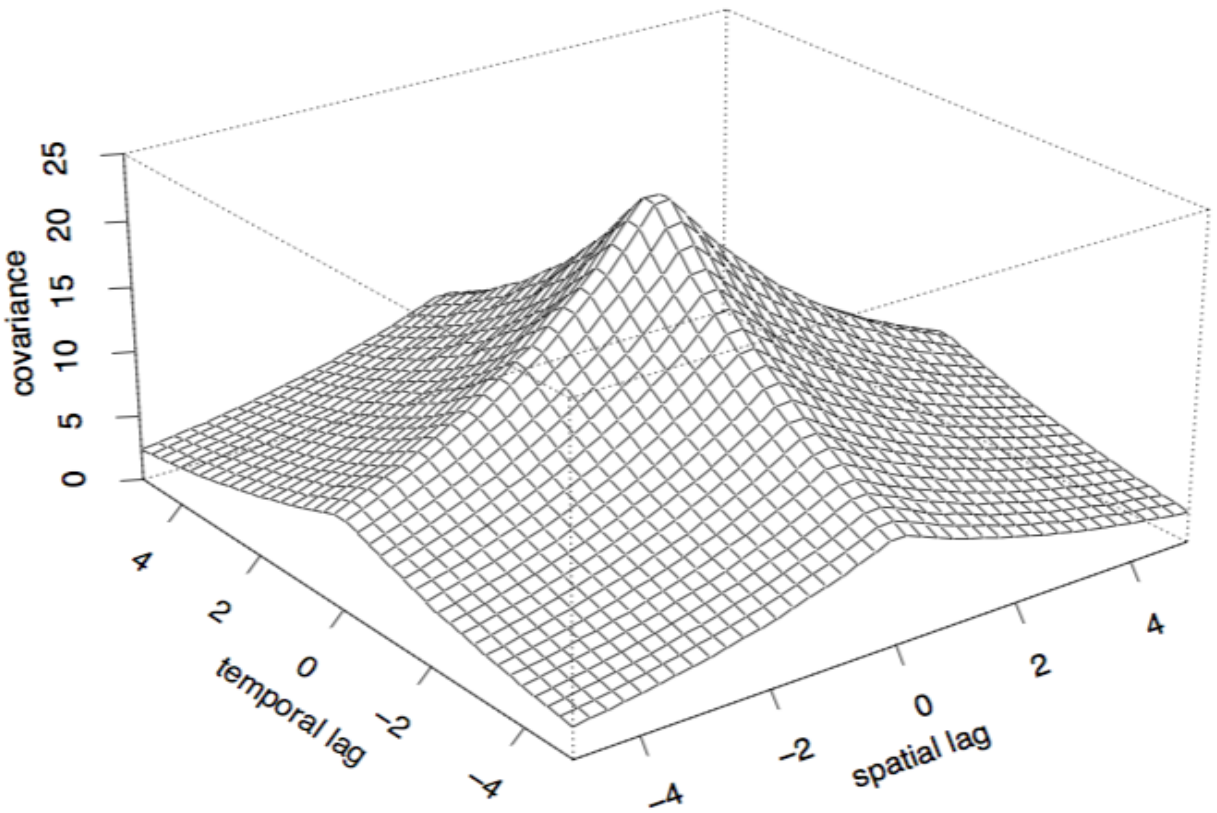
$$C(h,k) = C_s(h)\, C_t(k).$$

- **Separable additive (sum) model:**

$$C(h,k) = C_s(h) + C_t(k) \quad \text{(valid but no space–time interaction)}.$$

- Both forms are valid if C_s and C_t are valid covariance functions (often plus a nugget $\tau^2 \mathbf{1}_{\{h=0,k=0\}}$).

Separable Space-Time Covariance



Space–Time Covariance: separable vs. nonseparable

- **Separable (means no interaction):**

$$C(h,k) = C_s(h)\, C_t(k) \quad \text{or} \quad C(h,k) = C_s(h) + C_t(k).$$

  Valid if C_s and C_t are valid covariances; however they *cannot capture* transport/advection patterns (space–time interaction).

- **Nonseparable (with interaction):**

$$\text{Product–sum: } C(h,k) = \kappa C_s(h) C_t(k) + C_s(h) + C_t(k),\ \kappa > 0.$$

$$\text{Gneiting class (in } d \text{ dims): } C(h,k) = \frac{\sigma^2}{\{\psi(|k|^2)\}^{d/2}}\, \varphi\left(\frac{h}{\sqrt{\psi(|k|^2)}}\right),$$

  with φ completely monotone and ψ a Bernstein function.

- Nonseparable modeling is a major theme in the work of Stein, Cressie–Huang, Fuentes, and Gneiting.

Spectral routes to nonseparable space–time

Spectral representation (Bochner/Cramér):

$$C(\mathbf{x}, \tau) = \int_{\mathbb{R}^d} e^{i\omega^\top \mathbf{x}}\, k(\omega) f_\omega(\tau)\, \mathrm{d}\omega,$$

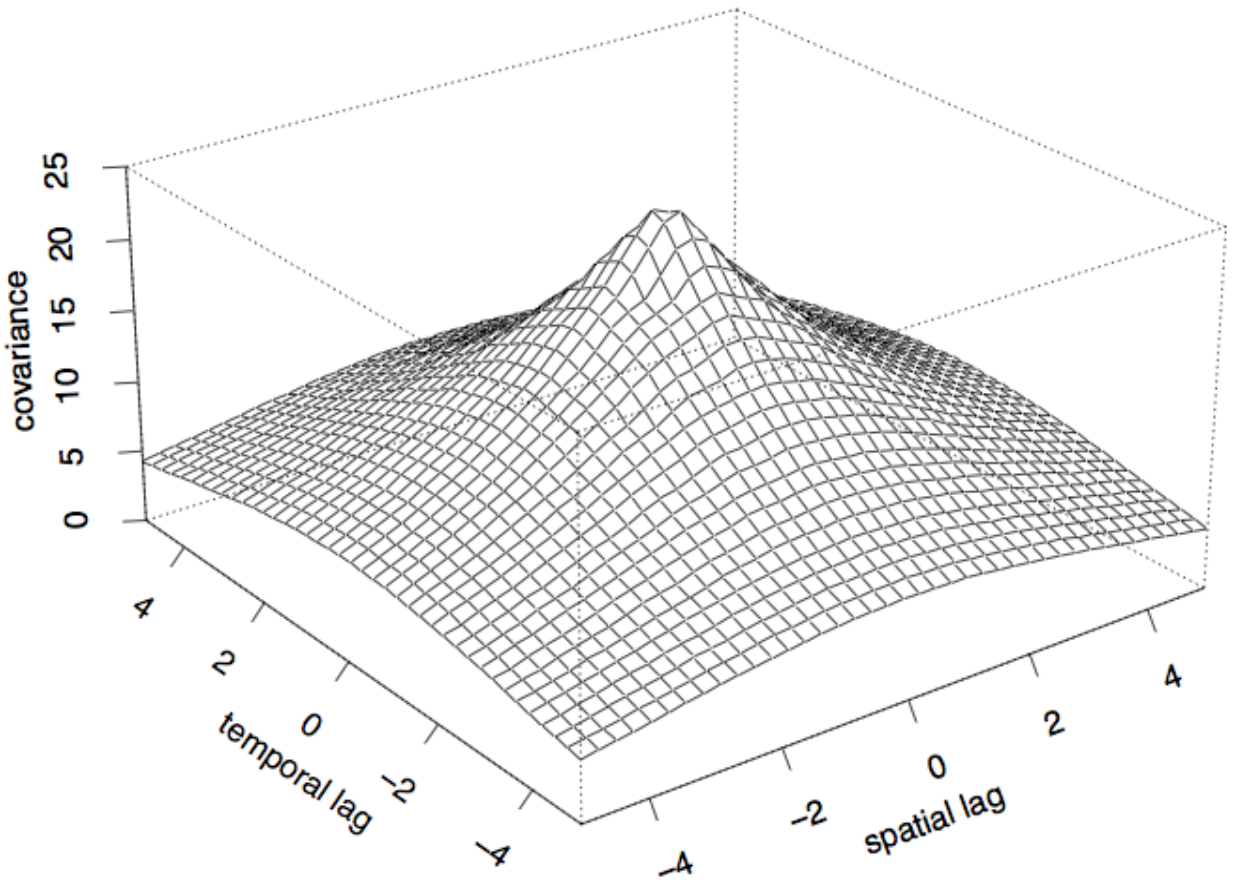
where $k(\omega) \geq 0$ and, for each frequency ω, $f_\omega(\tau)$ is a valid temporal autocovariance.

Fourier pair (Lebesgue integration):

$$f(\omega, \lambda) = \frac{1}{(2\pi)^{d+1}} \int_{\mathbb{R}^d} \int_{\mathbb{R}} e^{-i(\omega^\top \mathbf{x} + \lambda\tau)}\, C(\mathbf{x}, \tau)\, \mathrm{d}\tau\, \mathrm{d}\mathbf{x}.$$

- This framework underlies nonseparable constructions in Stein, Cressie–Huang, and Fuentes, and motivates classes like Gneiting's.
- Practically: spectral views explain tilted covariance "ridges" caused by transport/advection (interaction between space and time).

Non-separable Space-Time Covariance



Anisotropy and Space–Time

Spatial anisotropy via metric transform:

$$\gamma(\mathbf{h}) = \gamma_0(\|A\mathbf{h}\|), \quad Q = A^\top A \succ 0,$$

where γ_0 is isotropic.

Space–time as componentwise anisotropy:

$\gamma(h_x, h_y, k) = \gamma_1\big(\sqrt{h_x^2 + h_y^2}\big) + \gamma_2(|k|)$ (zonal anisotropy; treat time as a special axis).

Validity cautions and constructions

- Semivariograms must be *conditionally negative definite*; arbitrary linear directional sums can violate this.
- Safer choices:

  Mixtures: $\gamma(\mathbf{h}) = \sum_m \mu_m \gamma_0^{(m)}(\|A_m\mathbf{h}\|),\ \mu_m \geq 0;$

  Product models (covariance side): $C(\mathbf{h}) = \prod_{i=1}^{n} C_i(h_i),$

  or nonseparable classes (e.g., product–sum, Gneiting) to encode interaction.

Sample ST variogram with gstat

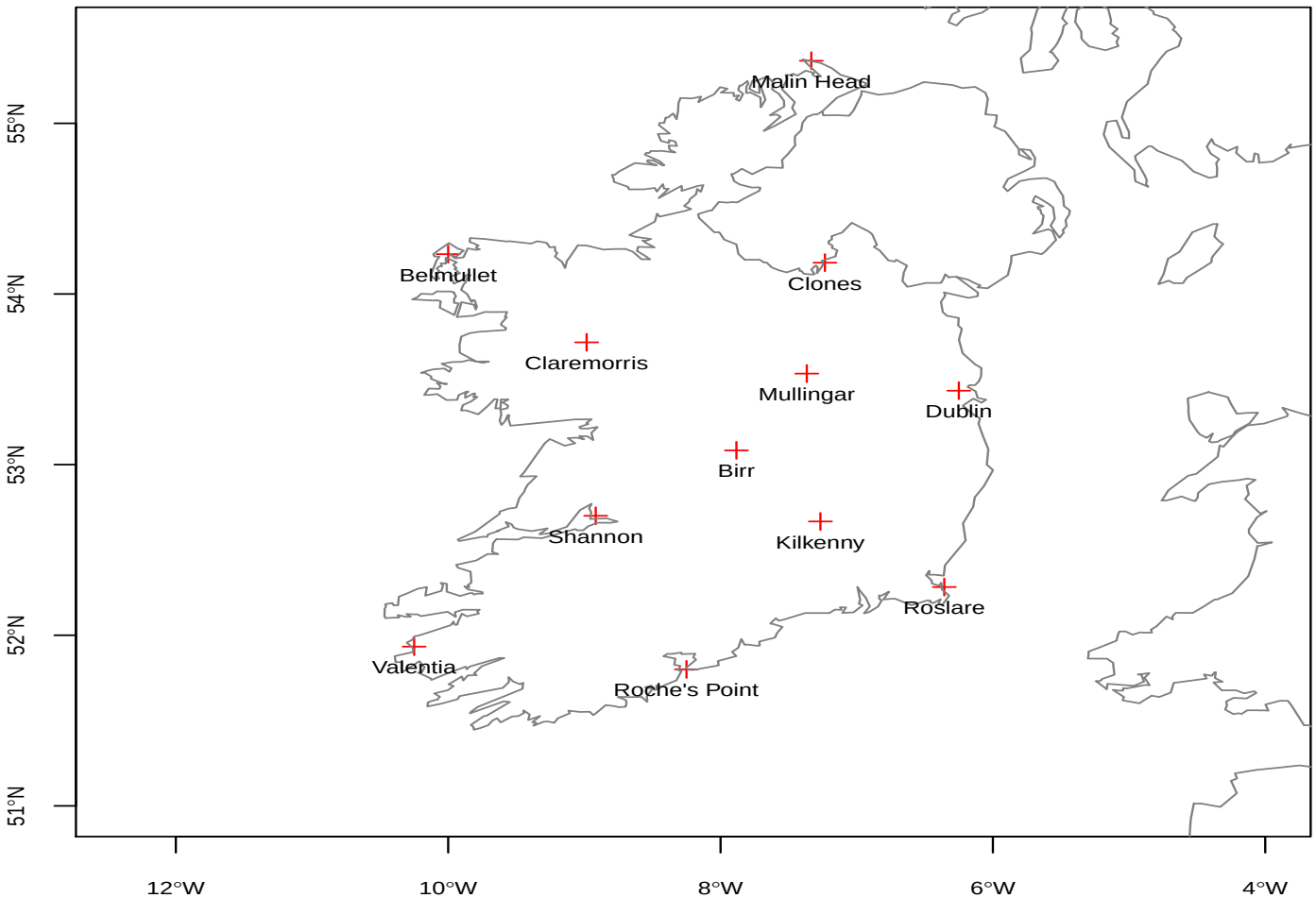
```
library(gstat); library(spacetime)
# 'z' is a matrix n_locations x n_times
st_obj <- STFDF(pts, times, data = data.frame(z = as.vector(Z)))
vg <- variogramST(z ~ 1, data = st_obj,
                  tlags = 0:10,
                  assumeRegular = TRUE,
                  cutoff = 1.5, width = 0.15)
```

Fit ST variogram model

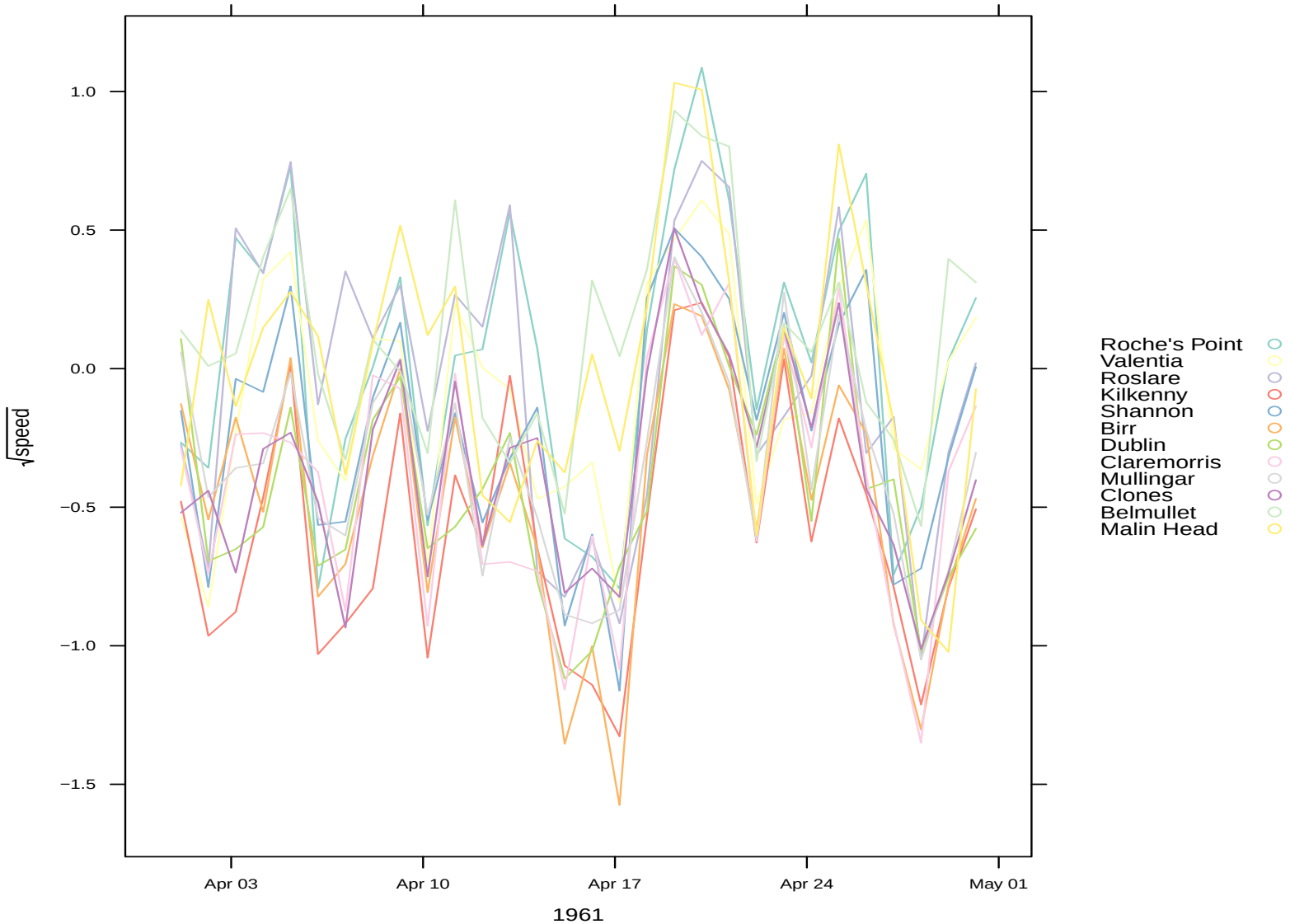
```
model <- vgmST("separable",
               space = vgm(1, "Exp", 50, 0.1),
               time  = vgm(1, "Exp", 5,  0.1), sill = 1)
fit   <- fit.StVariogram(vg, model)
```

Spatio-Temporal Geostatistics in R

An example of regular space-time data (STFDF) is daily wind (m/s) measurements at 12 weather stations in Ireland.

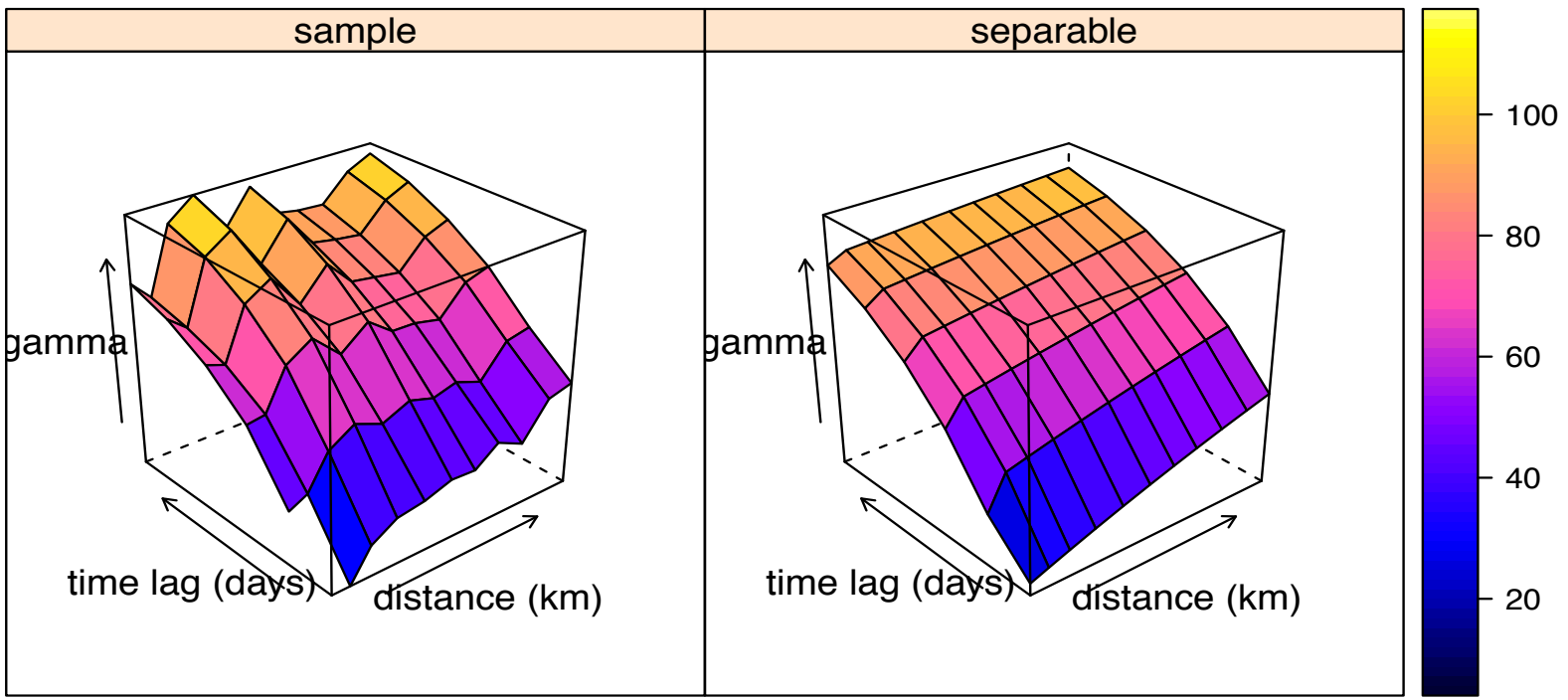


Time series at each location.



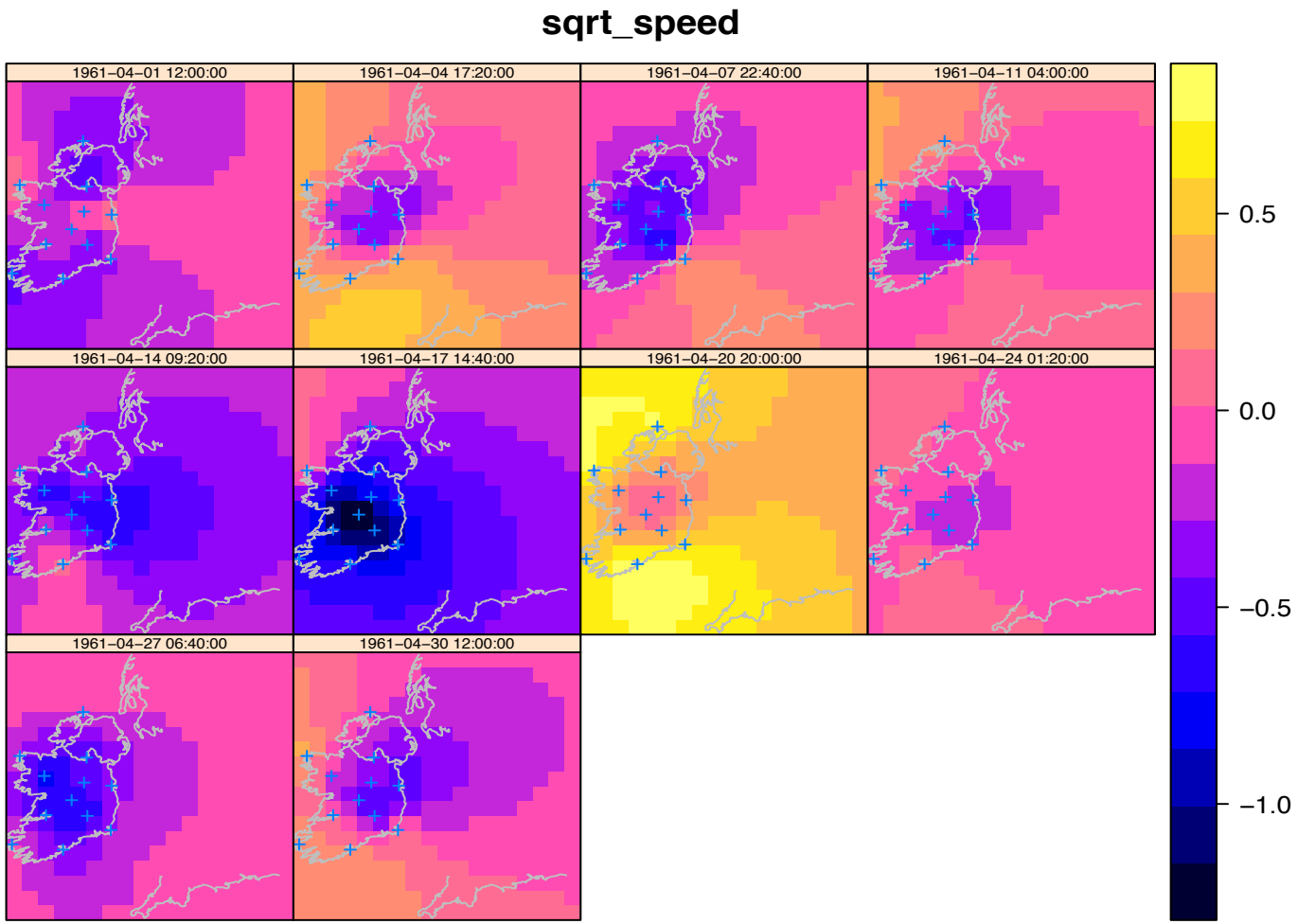
Spatio-Temporal Geostatistics in R

Applying a space-time variogram using `vgmST()`, we specify a spatial variogram and a temporal variogram:
Spatial: give initial parameters for partial sill, model (exponential, spherical, gaussian, matern), range, nugget. These will be in terms of distance units (e.g. km) Temporal: give initial parameters for partial sill, model (same as above), range. These will be in terms of temporal units (e.g. month, or day, or year)



Spatio-Temporal Kriging in R

Using that S-T variogram to perform kriging, we get the following predicted surfaces (for 10 days)



Spatio-Temporal GAMs: separable mean)

$$Z_{it} = g(s_i) + f(t) + \varepsilon_{it}, \qquad \varepsilon_{it} \sim N(0, \sigma^2)$$

- **Idea:** encode space–time structure in the *mean* via smooths.
- **Separable space time in the mean:** spatial smooth $g(s)$ plus temporal smooth $f(t)$.
- **Smoothing selection:** prefer `method = "REML"` for stable penalties.

Thin-plate for spatial, cubic regression for temporal

```
library(mgcv)
mod_sep <- gam(z ~ s(x, y, bs = "tp", k = 150) +
s(t, bs = "cr", k = 30),
               data = dat, method = "REML")
```

Temporal structure: seasonality & AR(1) residuals

Seasonality (e.g., day-of-year, hour-of-day): use a *cyclic* spline.

```
mod_cyc <- gam(z ~ s(x, y) + s(doy, bs = "cc", k = 20),
data = dat, method = "REML")
```

Serial correlation in residuals (within series) with `bam()` AR(1):

- Require an indicator `AR.start` marking the first row of each series.

```
mod_ar1 <- bam(z ~ s(x, y, k = 150) + s(t, k = 30),
               data = dat, method = "fREML", discrete = TRUE,
               rho = 0.7, AR.start = dat$start)
```

`bs="cc"` enforces periodic endpoints; `bam` is memory-efficient and supports AR(1).

Group trends & factor-smooths; random effects

Group-specific temporal smooths (e.g., counties, sensors):

```
mod_fs <- gam(z ~ s(x, y) + s(t, by = region, bs = "fs", m = 1, k = 10) +
                  s(region, bs = "re"),   # random intercepts
              data = dat, method = "REML")
```

Spatially varying coefficients: allow a covariate's effect to vary over space.

```
mod_svc <- gam(z ~ s(x, y) + s(t) + covar + s(x, y, by = covar, k = 100),
               data = dat, method = "REML")
```

Can center/scale `covar` to improve identifiability; include the main effect (`covar`) when using `by=` to model variations around it.

Non-separable space–time: tensor products

- We can consider non-separable space-time smoothing using space-time basis functions

$$Z_{it} = g(s_i) + f(t) + \eta(s_i, t) + \varepsilon_{it}$$

- The basis function η has both space and time in it, so presumably we can multiply a spatial and a temporal basis function together: $b_s(s) \otimes b_t(t)$.
- This cross product basis is known as a tensor-product basis function, and it allows for smoothing over space and time simultaneously.

Non-separable space–time: tensor products

$$Z_{it} = g(s_i) + f(t) + \eta(s_i, t) + \varepsilon_{it}, \quad \eta(s,t) = \text{interaction smooth}$$

Tensor products handle different units/scales across dimensions. ANOVA decomposition:

$$\underbrace{\text{ti}(x,y)}_{\text{space}} + \underbrace{\text{ti}(t)}_{\text{time}} + \underbrace{\text{ti}(x,y,t)}_{\text{space–time interaction}}$$

R interaction, test non-separability

```
mod_st <- gam(z ~ ti(x, y, bs = "tp", k = 150) +
                  ti(t,    bs = "cr", k = 30) +
                  ti(x, y, t, bs = c("tp","cr"),
                  d = c(2,1), k = c(100,30)),
                 data = dat, method = "REML")
anova(mod_sep, mod_st, test = "Chisq")  # evidence for interaction?
```

`ti()` is usually preferable to `te()` when you want explicit main effects + interaction and to test separability. However, `ti()` is very computationally intensive.

Boundaries, anisotropy, and constraints

Coastlines or barriers: soap-film smooths respect complex boundaries.

```
mod_soap <- gam(z ~ s(x, y, bs = "so",
xt = list(bnd = boundary_poly)) + s(t),
data = dat, method = "REML")
```

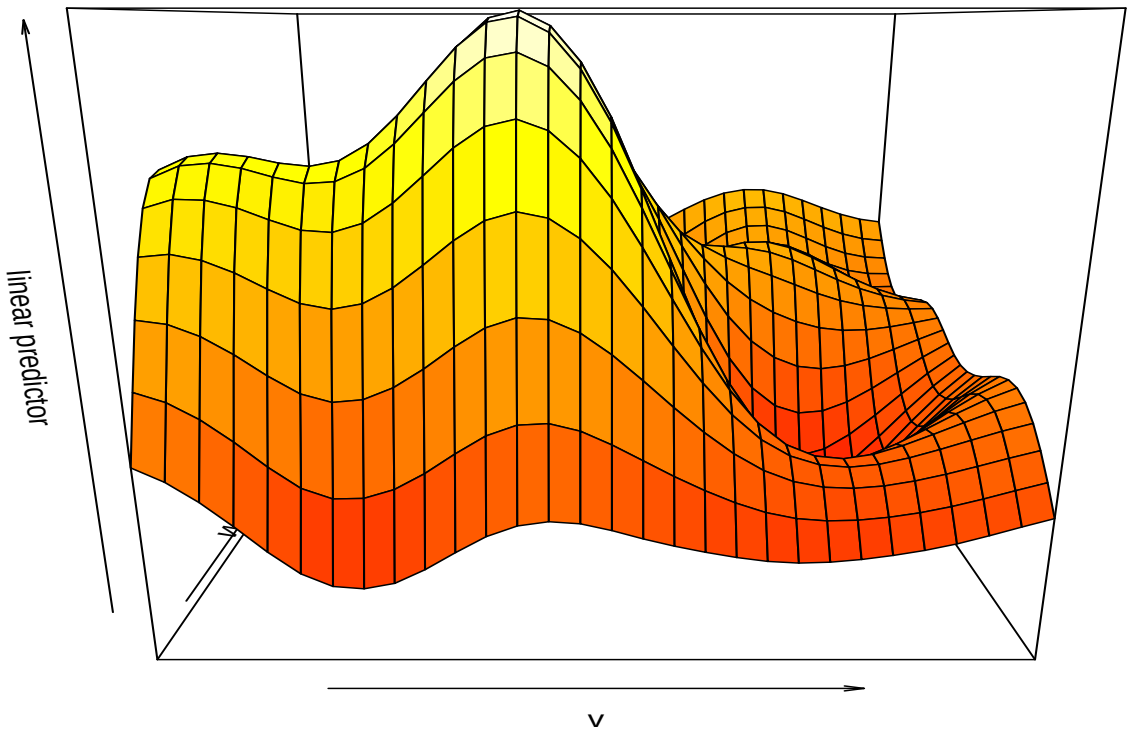
Anisotropy in space: rotate/scale coordinates or use tensor products with separate penalties per axis:

```
mod_aniso <- gam(z ~ te(x, y, bs = c("cr","cr"), K = c(80,80),
anisotropic = TRUE) + s(t), data = dat, method = "REML")
```

Soap needs a valid boundary polygon (`bnd`). For mild anisotropy, tensor products often suffice without explicit rotation.

Spatio-Temporal GAM models

```
b = gam(y ~ te(v,w,u,k=c(25,5),
d=c(2,1),bs=c("tp","cr"),full=TRUE),
method="REML")
```



Model checking & inference

Basis adequacy & gam diagnostics

```
gam.check(mod_st)            # k-index, residual patterns
concurvity(mod_st)           # non-linear collinearity among smooths
```

Visualize smooths (with uncertainty)

```
# component-wise
plot(mod_st, pages = 1, scheme = 2, shade = TRUE)
# or with vis.gam
```

Space–time interaction test

```
anova(mod_sep, mod_st, test = "Chisq")  # use ti(x,y,t)?
```

Temporal autocorrelation left in residuals?

```
acf(residuals(mod_st), na.action = na.pass)
```

Template: from separable to non-separable

```
# Separable mean model
mod_sep <- gam(z ~ s(x, y, k = 150) +
s(t, k = 30), data = dat, method = "REML")

# Add interaction (non-separable)
mod_st  <- gam(z ~ ti(x, y, k = 150) + ti(t, k = 30) +
ti(x, y, t, k = c(100,30), bs = c("tp","cr"), d = c(2,1)),
data = dat, method = "REML")

# Compare
anova(mod_sep, mod_st, test = "Chisq")

# If residual correlation remains, refit with bam + AR(1)
mod_st_ar1 <- bam(formula(mod_st), data = dat, method = "fREML",
discrete = TRUE, rho = 0.6, AR.start = dat$start)
```

Areal space–time: Data & W

- We have $i = 1, \ldots, n$ areas (blocks); $t = 1, \ldots, T$ times; response y_{it}; covariates X_{it}.
- **Weights** W: as before rook/queen contiguity or kNN
- Keep W fixed over t and align with area ordering.
- **Stack by time**: $y = (y_1^\top, \ldots, y_T^\top)^\top$; use time dummy variables for shifts.
- **Lags**: spatial Wy_t; temporal y_{t-1}; cross-lag Wy_{t-1}; lagged covariates WX_t.

Static areal SAR/SEM/SDM/SARAR (per time t)

- **SAR (lag)** $y_t = \rho W y_t + X_t \beta + u_t$
- **SEM (error)** $y_t = X_t \beta + u_t,\ u_t = \lambda M u_t + \varepsilon_t$
- **SDM (Durbin)** $y_t = \rho W y_t + X_t \beta + W X_t \gamma + u_t$
- **SARAR (SAC)** $y_t = \rho W y_t + X_t \beta + u_t,\ \ u_t = \lambda M u_t + \varepsilon_t$
- W and M often identical; row-standardization common in SAR/SDM.
- Identification: $\rho \in \left(-1/\omega_{\max}(W), 1/\omega_{\max}(W)\right)$ where $\omega_{\max}$.

Static Temporal SAR

$$y_t = \rho W y_t + X_t \beta + u_t, \quad u_t \sim N(0, \sigma^2 I), \qquad S(\rho) = (I - \rho W)^{-1}.$$

- **Meaning:** dependent variable spillovers via Wy_t; "static" = no y_{t-1}.
- **Stability:** $(I - \rho W)$ invertible (e.g., $|\rho| < 1/\omega_{\max}(W)$).
- **Time:** fit per year *or* pool with time fixed effects: $+\tau_t$.
- **Interpretation via impacts:** like before, direct/indirect/total from $S(\rho)\beta$ (not raw β).
- Treat years as replicates and pool, adding time fixed effects (still static):

$$y_t = \rho W y_t + X_t \beta + \tau_t \mathbf{1} + u_t$$

Estimate a single ρ and β, while τ_t captures shifts across time.

Dynamic SAR lag and Dynamic SAR error

Spatial–temporal lag model:

$$y_t = \rho W y_t + \phi y_{t-1} + X_t \beta + \varepsilon_t.$$

Spatial–temporal error model:

$$y_t = \rho W y_t + X_t \beta + u_t, \qquad u_t = \lambda W u_t + \psi u_{t-1} + \eta_t.$$

- Can add $W y_{t-1}$ (cross-lag) and $W X_t$ (Durbin) to separate direct vs indirect effects.

Dynamic SAR lag and Dynamic SAR error

Model A: Dynamic SAR (lag in the mean)

$$y_t = \rho\, W y_t + \phi\, y_{t-1} + X_t\beta + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma^2 I).$$

- ρ: **contemporaneous spatial spillover** (neighbors' outcomes today).
- ϕ: **temporal persistence** (yesterday's own outcome).
- Reduced form: $y_t = S(\rho)\{\phi\, y_{t-1} + X_t\beta + \varepsilon_t\}$ with $S(\rho) = (I - \rho W)^{-1}$.

Model B: Spatio-temporal error SAR

$$y_t = \rho\, W y_t + X_t\beta + u_t, \qquad u_t = \lambda\, W u_t + \psi\, u_{t-1} + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma_\eta^2 I).$$

- ρ: **contemporaneous spatial spillover** (in the mean).
- λ: **spatial correlation in the errors**.
- ψ: **temporal correlation in the errors** (AR(1) noise).
- Implies $u_t = (I - \lambda W)^{-1}(\psi\, u_{t-1} + \eta_t)$; covariance carries across space and time.

Dynamic SAR (with AR(1) in the mean)

Joint over time (panel):

$$y_t = \rho\, W y_t + \alpha\, y_{t-1} + X_t\beta + \varepsilon_t, \qquad S(\rho) = (I - \rho W)^{-1}.$$

Short-run (impact at time t, holding y_{t-1} fixed):

$$\frac{\partial \mathbb{E}[y_t]}{\partial x_k} = S(\rho)\,\beta_k.$$

Long-run (effect after temporal feedback accumulates; steady state):

$$\frac{\partial \mathbb{E}[y]}{\partial x_k} = \underbrace{(I - \alpha\, S(\rho))^{-1}\, S(\rho)}_{\text{spatio-temporal multiplier}}\ \beta_k.$$

Report **direct / indirect / total** as the average diagonal / average off-diagonal row sums / average row sums of the corresponding multiplier matrix (short-run or long-run).

R: Static SAR and impacts

```
library(sf); library(sfdep); library(spatialreg)

W (row-standardized) once, reused across time

nb <- st_contiguity(areas_sf)
lw <- st_weights(nb, style = \W")

Pooled static SAR with time fixed effects

fit_sar <- lagsarlm(y ~ x1 + x2 + factor(year),
data = panel_df, listw = lw, method = \eigen")
summary(fit_sar)

Direct / indirect / total impacts with simulation intervals

imp <- impacts(fit_sar, listw = lw, R = 1000)
summary(imp, zstats = TRUE)
```

R: Dynamic SAR and impacts

```
library(plm); library(splm); library(spatialreg); library(Matrix)

pd <- pdata.frame(panel_df, index = c(\area_id",\year"))
pd$y_lag <- lag(pd$y, 1)

#Dynamic SAR via GMM lag = TRUE turns on Wy_t

fit_dyn <- spgm(y ~ y_lag + x1 + x2 + factor(year),
data = pd, listw = lw,
lag = TRUE,                # include Wy_t
model = \within",          # unit fixed effects
moments = \weights")       # GMM

summary(fit_dyn)
rho_hat   <- coef(fit_dyn)[\lambda"]
alpha_hat <- coef(fit_dyn)[\y_lag"]

Build S(rho) and compute average multipliers

S <- invIrW(lw, rho = rho_hat, method = \LU")
n <- nrow(S)
avg_diag_S   <- mean(diag(S))
avg_rowsum_S <- mean(rowSums(S))

#Short-run impacts for coefficient beta_k:

direct_SR = beta_k * avg_diag_S; indirect_SR = beta_k * (avg_rowsum_S - avg_diag_S)

Long-run uses M = (I - alpha * S)^{-1} %*% S  (careful: large n -> use sparse solvers)

I <- Diagonal(n)
M_LR <- solve(I - alpha_hat * S, S)
avg_diag_M   <- mean(diag(M_LR))
avg_rowsum_M <- mean(rowSums(M_LR))

#Long-run impacts:

direct_LR = beta_k * avg_diag_M; indirect_LR = beta_k * (avg_rowsum_M - avg_diag_M)
```

Spatio-Temporal CAR

- Recall CAR models have a value that depends conditionally on the values of neighboring locations. The spatial dependence is typically defined using a spatial adjacency matrix W and in ST-CAR the temporal component is often modeled as an autoregressive process where the value of the outcome at time t depends on its value at previous time points.

$$Y_{it} \sim N(\mu_{it}, \sigma^2)$$
$$\mu_{it} = X_{it}\beta + \phi_{it}$$

where X_{it} are covariates at location i and time t, β are coefficients for the covariates, and ϕ_{it} is spatiotemporal random effects with form:

$$\phi_{it} = \phi_{i,t-1} + \rho \sum_{j \in N(i)} W_{ij}\phi_{jt} + \varepsilon_{it}$$

where W_{ij} is the adjacency matrix describing spatial neighbors, ρ is the spatial autoregressive parameter $\phi_{i,t-1}$ is the temporal dependency and ε_{it} is an independent error term.

Spatio-Temporal CAR

- A CAR model is effectively a random effects model with a covariance structure defined by the spatial weights matrix W.
- The spatial dependence structure in a CAR model can be viewed as an extension of a random intercept model, where the random intercepts vary smoothly across space.
- These models are typically fit using a Bayesian hierarchical formulation in R-INLA. There is a CARBayesST package that implements spatiotemporal CAR models.

Spatio-Temporal CAR

$$Y_{it} \mid \mu_{it} \sim D(\mu_{it}), \quad g(\mu_{it}) = X_{it}\beta + \phi_{it}.$$

Spatio-temporal random effects:

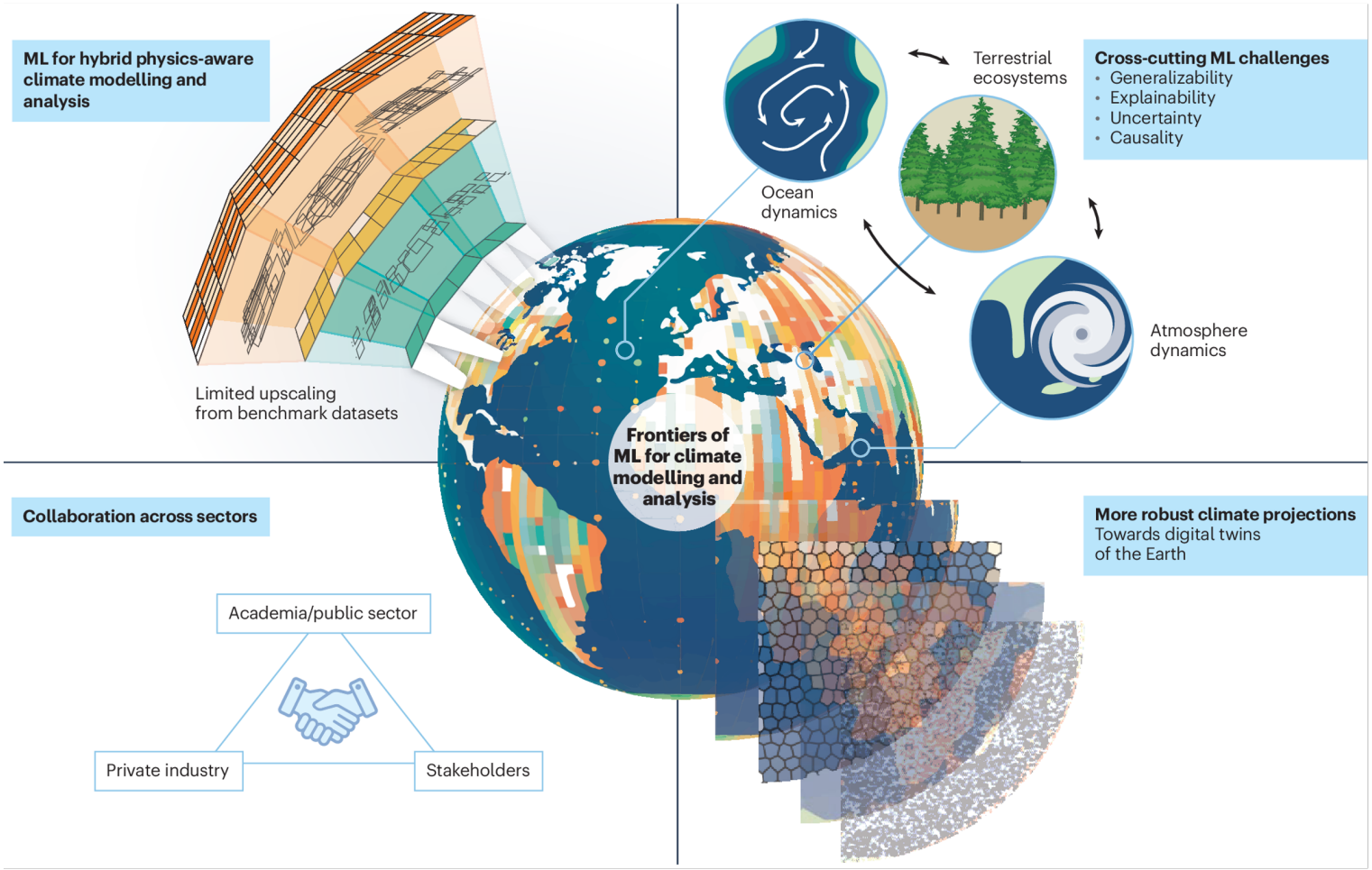
$\phi_t \mid \phi_{t-1} \sim N\big(\alpha\,\phi_{t-1},\, \tau^{-2}Q^{-1}\big), \quad Q = D - \rho W$ (CAR precision, proper if $|\rho| < 1$).

Variants:

- **ST-CAR(AR1)**: $\alpha \in (-1, 1)$ controls temporal persistence.
- **Separable**: $\mathrm{Cov}(\mathrm{vec}(\Phi)) \approx \Sigma_T \otimes \Sigma_S$.

Fit with INLA (fast latent Gaussian) or MCMC (e.g., `CARBayesST`).

Machine Learning for Spatiotemporal Data



Eyring et al (2024) Pushing the frontiers in climate modelling and analysis with machine learning, Nature Climate Change

Article | Open access | Published: 17 December 2020

A novel framework for spatio-temporal prediction of environmental data using deep learning

Federico Amato ✉, Fabian Guignard, Sylvain Robert & Mikhail Kanevski

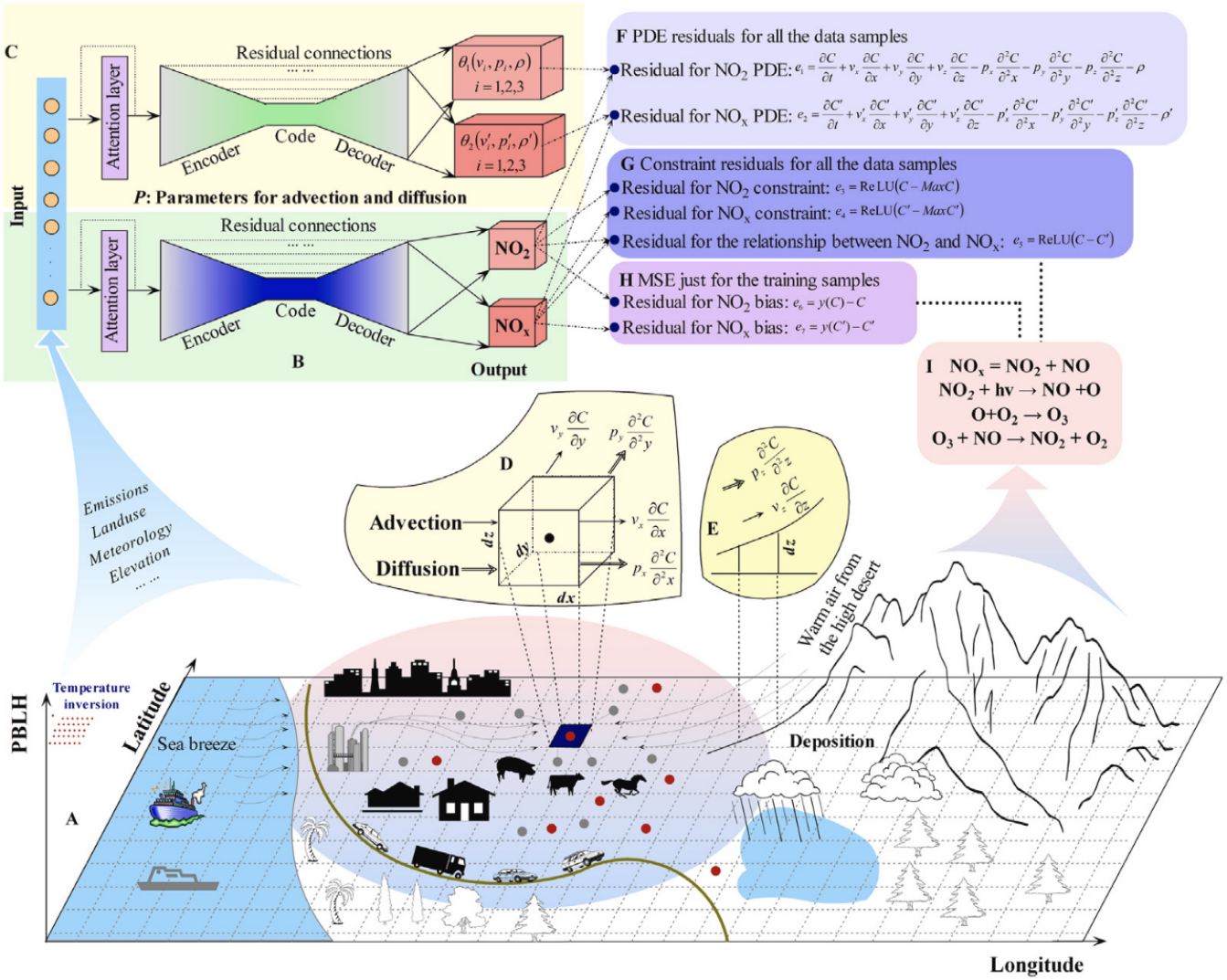
Scientific Reports **10**, Article number: 22243 (2020) | Cite this article

47k Accesses | **84** Citations | **14** Altmetric | Metrics

Abstract

As the role played by statistical and computational sciences in climate and environmental modelling and prediction becomes more important, Machine Learning researchers are becoming more aware of the relevance of their work to help tackle the climate crisis. Indeed, being universal nonlinear function approximation tools, Machine Learning algorithms are efficient in analysing and modelling spatially and temporally variable environmental data. While Deep Learning models have proved to be able to capture spatial, temporal, and spatio-temporal dependencies through their automatic feature representation learning, the problem of the interpolation of continuous spatio-temporal fields measured on a set of irregular points in space is still under-investigated. To fill this gap, we introduce here a framework for spatio-temporal prediction of climate and environmental data using deep learning. Specifically, we show how spatio-temporal processes can be decomposed in terms of a sum of products of temporally referenced basis functions, and of stochastic spatial coefficients

Deep Learning for Spatiotemporal Data



https://www.sciencedirect.com/science/article/pii/S016041202500666X?via%3Dihub

Machine Learning for Spatiotemporal Data

Different types of features (outcomes) can be predicted using machine learning or deep learning.

- Numeric outcomes: using regression, machine learning regression, deep learning regression. Examples include carbon dioxide over space and time, change in greenspace over space and time, median household income over space and time, etc.
- Binary or categorical outcomes: spatiotemporal logistic regression, machine learning and deep learning for classification. Categorical predictions involve assigning features to categories, such as types of land cover, types of forests, types of wetlands, types of animals, etc.
- Probabilistic outcomes: logistic regression, maximum entropy, and deep learning. Examples include probability of weather, probability of a particular species moving somewhere, etc.

Classical Machine Learning

- These methods incorporate spatial and temporal dependencies into traditional machine learning models through "feature engineering" or custom kernels.
- Examples include Random Forest, eXtreme gradient boosting. Spatiotemporal variables (e.g., spatial coordinates, temporal lags, and time-based features like seasons) are explicitly included as input features.
- Support vector machines can be enhanced with custom kernels to account for spatiotemporal proximity (Gaussian kernel with spatial and temporal distances).
- Temporal lags include creating new variables that are lagged time (e.g. moving average, lag by certain number of hours or days).
- Spatial lags include creating new variable that distance weights the nearby observations. For example, define your neighbors (e.g. knn-4) then weight the values of the neighbors by the squared inverse distance from the target variable.

Neural Network Based Methods (Deep Learning)

Deep learning models can effectively model complex spatiotemporal relationships, but a lot of data are required!

- Convolutional neural networks (CNNs) are used to extract spatial patterns from grid-based data (e.g., satellite images or spatial grids). Use sequences of images or grids (e.g., video or time-series images) for the temporal part.
- Recurrent neural networks (RNNs), these include Long Short-Term Memory (LSTM) networks, are well-suited for temporal modeling and can be extended to spatiotemporal data by processing temporal sequences for each spatial location.
- Graph neural networks (GNNs) are graph based models such as spatiotemporal graph convolutional networks designed to model relationships between spatial nodes (regions) and their evolution over time.
- Emerging techniques include spatiotemporal encoders, where autoencodes compress spatiotemporal data into latent representations.
- Another emerging method includes deep reinforcement learning.
- Hybrid methods include deep learning combined with kriging and ARIMA with predictors from ML or DL.

Deep Learning for Spatiotemporal Data

npj | climate and atmospheric science

www.nature.com/npjclimatsci

ARTICLE OPEN

Check for updates

Improving air quality assessment using physics-inspired deep graph learning

Lianfa Li[1,2]✉, Jinfeng Wang[1]✉, Meredith Franklin[2,3], Qian Yin[1], Jiajie Wu[1], Gustau Camps-Valls[4], Zhiping Zhu[1], Chengyi Wang[5], Yong Ge[1] and Markus Reichstein[6]

Existing methods for fine-scale air quality assessment have significant gaps in their reliability. Purely data-driven methods lack any physically-based mechanisms to simulate the interactive process of air pollution, potentially leading to physically inconsistent or implausible results. Here, we report a hybrid multilevel graph neural network that encodes fluid physics to capture spatial and temporal dynamic characteristics of air pollutants. On a multi-air pollutant test in China, our method consistently improved extrapolation accuracy by an average of 11–22% compared to several baseline machine learning methods, and generated physically consistent spatiotemporal trends of air pollutants at fine spatial and temporal scales.

npj Climate and Atmospheric Science (2023)6:152 ; https://doi.org/10.1038/s41612-023-00475-3

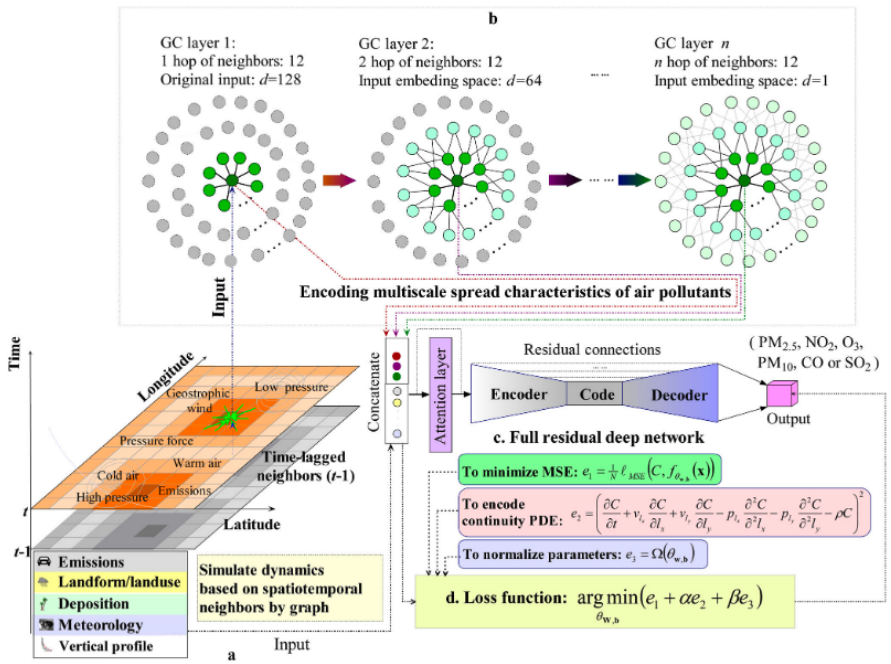


Fig. 1 Architecture of the deep graph network for air quality assessment. a The input data included atmospheric and surface grids, emissions or their proxies, meteorology, dry deposition, landuse and vertical profile. **b** The graph convolutions are constructed in multiple layers to simulate spatiotemporal dynamics of air pollutants, encoding their local spreading characteristics. **c** The GC multiscale outputs are concatenated with the input to enter full residual encoder-decoder to account for transformation and deposition. **d** The loss function included e_1 (mean square error (MSE) between observed and predicted values), e_2 (residual to encode PDE) and e_3 (normalization).

Spatiotemporal ML

Goal is high-quality prediction (and uncertainty) for $y(s,t)$ using flexible learners.
When it helps:

- Many, possibly nonlinear covariates, complex interactions, large data.
- You are willing to trade some interpretability for predictive lift.

Core idea: encode space & time into *features* and validate so we generalize across new places/times.

Feature engineering: spatial (from what we learned)

Let $X(s,t)$ be raw predictors.

- **Neighborhood summaries:** spatial lags $WX(t)$; kNN means/medians.
- **Distance variables:** distances to sources/sinks; kernel densities (KDE) from `spatstat`.
- **Smooth bases:** thin-plate spline bases $\phi_{sp}(s)$ (as in GAMs) evaluated at s and fed to ML.
- **Graph features:** degree, Laplacian embeddings, Moran eigenvector maps (MEM) for broad-scale patterns.
- **Geostatistics as features:** kriging mean/variance from used as covariates.

Feature engineering: temporal

- **Lags & differences:** X_{t-1}, X_{t-7}; $\Delta X_t = X_t - X_{t-1}$.
- **Rolling windows:** means, mins, maxes, variance over past k steps.
- **Seasonality encodings:** cyclic splines or $\sin(2\pi t/P), \cos(2\pi t/P)$ for $P \in 24, 7, 365$.
- **Calendar effects:** month, day-of-week, holiday flags.

Space–time interaction features

- **Tensor combos:** products between spatial bases and temporal bases $\phi_{sp}(s) \otimes \phi_t(t)$ (lightweight version of GAM).
- **Transport proxies:** wind (speed, direction), advection kernels, travel time—allow ML to pick up advection-like patterns.
- **Cross-lag terms:** WX_{t-1} (spatially lagged, temporally lagged predictors).

Learners: gradient boosting

XGBoost / LightGBM / CatBoost

- Handle nonlinearities and interactions automatically; robust to monotone transforms.
- Control complexity via trees (depth, learning rate, subsampling).
- **Good defaults (regression):** depth 4–8, η 0.03–0.1, 1000–3000 rounds with early stopping; subsample 0.6–0.8; colsample 0.6–0.8.
- Give the model spatial/temporal features either covariates or generated features.

When to use deep learning

- **CNNs (rasters):** convolve gridded covariate stacks per time slice; 3D CNN for (x, y, t) blocks.
- **RNN/TCN/Transformers (sequences):** per-site time series with shared weights; add site embeddings and spatial neighbors.
- **GNNs (graphs):** nodes = areal units/sensors; edges from contiguity or distance; temporal message passing for ST diffusion.
- Use DL you have many gridded covariates, long sequences, or complex adjacency; otherwise boosting often wins on tabular ST.

Validation: get the CV right

- **Spatial blocking:** leave-location-out (LOLO) or spatial blocks.
- **Temporal blocking:** rolling origin / leave-future-out with gaps.
- **ST blocking:** combine both: leave regions $\times$ time slices out.
- **Compute features inside each training fold only**. Recompute neighborhood/kriging features per fold to prevent leakage.

Report RMSE/MAE overall and by region/time; calibration of intervals if you model uncertainty.