STAT 510: Spatiotemporal Stats

Exploratory Modeling of SPT Data

Prof. Taylor-Rodriguez

ST Modeling Goals

All models are wrong but some are useful

(George Box)

Recall that three main goals for ST (statistical) modeling are

- predicting (with the associated uncertainty) the response at a location in space within the time span of the data
- infer the importance of covariates on the response when ST dependece is present
- forecast (with uncertainty estimates) response values at particular locations

ST Prediction

ST Prediction

Suppose we want to predict (interpolate) the unobserved response at a particular location at a time within the span of the data, given the available data.

Consider that similar factors drive observations that are nearby in time and space.

ST Prediction

Consider that similar factors drive observations that are nearby in time and space.

- The meteorological phenomena that drive rainfall (e.g.,El Nino) in one month typically lasts a few months.
- Religion and race are strong predictors of voters' choices. These are likely to be similar in nearby regions and times.
- School quality is a strong predictor of house prices. Nearby houses belong to the same school district.

What would be a good predictor in general?

ST Prediction

What would be a good predictor in general?

Tobler's Law

everything is related to everything else, but near things are more related than distant things

- **Idea:** A combination of values for observations nearby (in space and time)
- Or use all existing data, but give increasing weights as distances in time and space diminish

Deterministic Prediction: Inverse Distance Weighting

Simplest alternative to implement Tobler's law is use a weighted average, with weights inversely related to distance. For ST data

$$\left\{ Z(\mathbf{s}_{11}; t_1), Z(\mathbf{s}_{21}; t_1), \ldots, Z(\mathbf{s}_{m_1 1}; t_1), \ldots, Z(\mathbf{s}_{m_T T}; t_T)) \right\}$$

The IDW estimator at a location $\mathbf{s}_0$ and a time $t_0 \in [t_1, t_T]$ is

$$\hat{Z}(\mathbf{s}_0; t_0) = \sum_{j=1}^{T} \sum_{i=1}^{m_j} w_{ij}(\mathbf{s}_0; t_0) Z(\mathbf{s}_{ij}; t_j)$$

with weights given by

$$\begin{aligned} w_{ij}(\mathbf{s}_0; t_0) &= \frac{\tilde{w}_{ij}(\mathbf{s}_0; t_0)}{\sum_{k=1}^{T} \sum_{\ell=1}^{m_k} \tilde{w}_{k\ell}(\mathbf{s}_0; t_0)}, \quad \text{with} \\ \tilde{w}_{ij}(\mathbf{s}_0; t_0) &= \frac{1}{d\left((\mathbf{s}_{ij}; t_j), (\mathbf{s}_0; t_0)\right)^{\alpha}} \end{aligned}$$

where $\alpha > 0$ is a smoothing parameter (smaller values lead to more smoothness).

Deterministic Prediction: Inverse Distance Weighting

Things to consider

- What happens if $(\mathbf{s}_0; t_0) = (\mathbf{s}_{k\ell}; t_\ell)$ (i.e., it's an observed point)?
  Exact Interpolation

- What issues can arise from using an exact interpolator?
  If measurement error is present, interpolation is misleading

- Does it make sense to have $d(\cdot;\cdot)$ be a Euclidean distance?
  No, distances in time are not the same as distances in space

- How to choose the value of α?
  Cross-validation

In-class Exercise

Use the function `fields::rdist` (see `?rdist`) to predict the minimum temperature on July 4th, 14th and 29th of 1993 using IDW with $\alpha = 5$ at the spatio temporal prediction grid `pred_grid` defined below with the data from `Tmin_long`. Plot your results with `ggplot2::geom_tile`

```
data("NOAA_df_1990", package = "STRbook")
Tmin_long <- NOAA_df_1990 %>% # now subset the data
  filter(proc == "Tmin" &      # only max temperature
           month %in% 7 &    # May to September
           year == 1993 &
           day != 14) %>%  # year 1993
  mutate(t=as.integer(julian-min(julian-1))) #create time variable

#generate the ST prediction grid
pred_grid <- expand.grid(lon = seq(-100, -80, length = 20),
                         lat = seq(32, 46, length = 20),
                         day = c(4, 14, 29))
```

Deterministic Prediction: Kernel Predictors

IDW is a type of kernel predictor. Kernel predictors are defined as

$$\hat{Z}(\mathbf{s}_0; t_0) = \sum_{j=1}^{T} \sum_{i=1}^{m_j} w_{ij}(\mathbf{s}_0; t_0) Z(\mathbf{s}_{ij}; t_j),$$

with weights

$$\tilde{w}_{ij}(\mathbf{s}_0; t_0) = k\left((\mathbf{s}_{ij}; t_j), (\mathbf{s}_0; t_0); \theta\right),$$

where $k(\cdot, \cdot; \theta)$ is a *kernel function*, which quantfies distance between two locations with bandwidth parameter θ

Deterministic Prediction: Kernel Predictors

Some commonly used kernel functions

Gaussian radial basis kernel

$$k\left((\mathbf{s}_{ij};t_j),(\mathbf{s}_0;t_0);\theta\right)=\exp\left\{-\frac{1}{\theta}d\left((\mathbf{s}_{ij};t_j),(\mathbf{s}_0;t_0)\right)^2\right\}$$

Epanechnikov

$$k\left((\mathbf{s}_{ij};t_j),(\mathbf{s}_0;t_0);\theta\right)=\frac{3}{4}\left(1-d\left((\mathbf{s}_{ij};t_j),(\mathbf{s}_0;t_0)\right)^2\right)\text{ with }d\left(\cdot,\cdot\right)\in[0,1]$$

Tricube

$$k\left((\mathbf{s}_{ij};t_j),(\mathbf{s}_0;t_0);\theta\right)=\frac{70}{81}\left(1-\left|d\left((\mathbf{s}_{ij};t_j),(\mathbf{s}_0;t_0)\right)\right|^3\right)^3\text{ with }d\left(\cdot,\cdot\right)\in[0,1]$$

Deterministic Prediction: Uncertainty Quantification

- Deterministic methods DO NOT account for measurement or prediction uncertainty
- Non-exact interpolating methods may average away measurement error but have no built-in mechanism to quantify it
- Prediction error can be quantified through Cross-Validation (CV)
- As such, CV can also be used to select the smoothness parameters (α and θ)

K-fold Cross-Validation

*GOAL: get an **independent** assessment of error*

The steps involved in K-fold CV are

1. Partition data randomly in K (often $K \in \{5, 10, n\}$) roughly equally-sized pieces (the "folds")

2. Holding out one fold at a time, train/fit the model with remaining $K - 1$ folds

3. Predict data in hold-out fold using model trained without it

4. Calculate some metric (typically MSPE) to compare predictions and real values for each fold

5. Combine metrics from all K-folds to calculate CV score

K-fold Cross-Validation

Letting $k = 1, 2, \ldots, K$, then

1. Split data $Z_1, \ldots, Z_m$ data into K folds
2. Denote the data in kth folds as $\mathbf{Z}^{(k)} = (Z_1^{(k)}, \ldots, Z_{m_k}^{(k)})$
3. Fit model without $\mathbf{Z}^{(k)}$ and obtain predictions $\hat{Z}_i^{(k)}$ for $Z_i^{(k)}$ with $i = 1, \ldots, m_k$.
4. Compute for $k = 1, \ldots, K$, say, the MSPE

$$MSPE_k = \frac{1}{m_k} \sum_{i=1}^{m_k} (Z_i^{(k)} - \hat{Z}_i^{(k)})^2$$

5. Calculate the CV-score

$$CV_K = \frac{1}{K} \sum_{k=1}^{K} MSPE_k$$

K-fold Cross-Validation

Let's calculate the 5-fold CV-scores using a Gaussian kernel for $\theta = 0.5$ with the following dataset

```
Tmax_long <- NOAA_df_1990 %>% # now subset the data
  filter(proc == "Tmax" &      # only max temperature
           month %in% 7 &    # May to September
           year == 1993) %>%
  mutate(t=as.integer(julian-min(julian-1))) #create time variable
```

K-fold Cross-Validation

Let's write down our own generic CV function

```
library(fields)
# "data" must include variables: lon, lat, t and z
K.fold.cv <- function(data,nfolds=5,weight.fn,theta){
  mT <- nrow(data)
  Z <- data$z
  coords <- data %>% dplyr::select(lon,lat,t)
  dist_mat <- rdist(coords,coords)

  # sample fold label vector
  if(nfolds < mT){
    fold.vec <- sample(1:nfolds,mT,replace = T)
  }else{
    fold.vec <- 1:mT
  }
  w.tilde <- weight.fn(theta,dist_mat)
  MSPEk <- 1:nfolds %>%
    map_dbl(function(x){
      hold.out <- which(fold.vec==x)
      w <- w.tilde[hold.out,-hold.out,drop=F]
      w <- w * (1/rowSums(w))
      Z.hat <-  w %*% Z[-hold.out]
      mean((Z[hold.out]-Z.hat)^2)
      })
  # CV score
  return(mean(MSPEk))
}
```

K-fold Cross-Validation

... now define the weight function, and run the cross-validation procedure for $K = 5$.

```
weight.gauss <- function(theta, dist_mat){
  exp(-dist_mat^2/theta)
}

K.fold.cv(data=Tmax_long,
          nfolds=5,
          weight.fn = weight.gauss,
          theta=5)
```

```
## [1] 11.77615
```

In-class problem

Using the same dataset together with the functions defined above, cross-validate predictions with a Gaussian kernel setting the number of folds to $K = 5, 10, m \times T$ (the last one is leave-one-out CV), and the values of $\theta = 0.2, 0.4, \ldots, 2$.

Compare the 5, 10 and LOO cv procedures by contrasting their corresponding CV-scores vs θ curves.

Trend-Surface Estimation

Trend-Surface Estimation

An alternative to doing prediction based on deterministic methods is to use simple statistical models

- The idea is to try to capture all ST dependence in the *trend*

So what is gained by doing this?

- Easily implementable
- Provides model based error estimate
- Provides model based prediction-error variance
- We can also use cv to assess performance

Trend-Surface Estimation

For simplicity assume we have all locations $\{\mathbf{s}_1, \ldots, \mathbf{s}_m\}$ measured at all time points $\{t_1, \ldots, t_T\}$, such that

$$Z(\mathbf{s}_i; t_j) = \beta_0 + \beta_1 X_1(\mathbf{s}_i; t_j) + \cdots + + \beta_1 X_p(\mathbf{s}_i; t_j) + \epsilon(\mathbf{s}_i; t_j),$$

- $\epsilon(\mathbf{s}_i; t_j) \overset{iid}{\sim} N(0, \sigma^2)$.
- $X_j(\cdot; \cdot)$'s represent spatially varying, temporally varying, and/or spatio-temporally varying predictors
- could also represent ST *basis functions*

Basis Functions

Under certain regularity conditions, it is possible to decompose curves or surfaces using a linear combination of *elemental basis functions*.

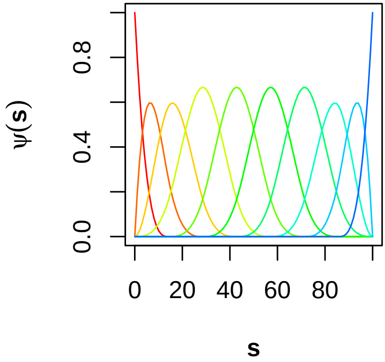
For example, a surface $Y(\mathbf{s})$ in space be represented as

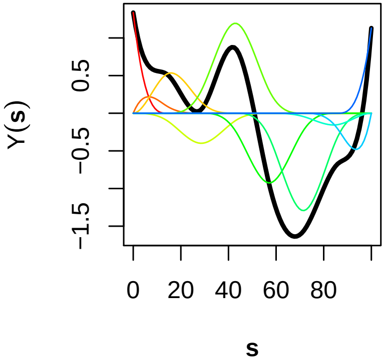
$$Y(\mathbf{s}) = \alpha_1\phi_1(\mathbf{s}) + \alpha_2\phi_2(\mathbf{s}) + \cdots + \alpha_r\phi_r(\mathbf{s})$$

- $\{\phi_k(\mathbf{s})\}$ denoting a **known** set of basis functions (can have local or global support)
- $\{\alpha_k\}$ represent constants that weight the relative importance of each basis function

Note here the absence of error, we are not dealing with data but with the *process* function

Basis Functions





Basis Functions

Some examples of basis functions are

polynomials, splines, wavelets, sines and cosines

If $Y(\mathbf{s})$ is a random process, a statistical model would assume known basis functions $\{\phi_k(\mathbf{s})\}$ and random weights $\{\alpha_k\}$, with a data model, for example, given by

$$\begin{aligned} Z(\mathbf{s}) &= Y(\mathbf{s}) + \epsilon(\mathbf{s}) \\ &= \alpha_1\phi_1(\mathbf{s}) + \alpha_2\phi_2(\mathbf{s}) + \cdots + \alpha_r\phi_r(\mathbf{s}) + \epsilon(\mathbf{s}) \end{aligned}$$

Very cool... these models are easy to fit and can be super flexible

Trend-Surface Estimation: Example

Consider the NOAA daily Tmax data for July of 1993, which has $m = 138$ locations, each measured every day of the month (i.e., $T = 31$). Let's use as covariates:

$X_0(\cdot;\cdot) = 1$: Intercept
$X_1(\cdot;\cdot)$: lon
$X_2(\cdot;\cdot)$: lat
$X_3(\cdot;\cdot)$: t
$X_4(\cdot;\cdot)$: lon×lat

$X_5(\cdot;\cdot)$: lon×t
$X_6(\cdot;\cdot)$: lat×t
$X_k(\cdot;\cdot) = \phi_{k-6}(\cdots)$: with $k = 7, \ldots, 18$ spatial-only basis functions

Trend-Surface Estimation: Example

Now, let's fit the model

$$Z(\mathbf{s}_i; t_j) = \beta_0 + \beta_1 X_1(\mathbf{s}_i; t_j) + \cdots + \beta_{18} X_{18}(\mathbf{s}_i; t_j) + \epsilon(\mathbf{s}_i; t_j),$$

using *ordinary least squares*

$$RSS = \sum_{j=1}^{T} \sum_{i=1}^{m} (Z(\mathbf{s}_i; t_j) - \hat{Z}(\mathbf{s}_i; t_j))^2$$

and find parameter estimates $\hat{\boldsymbol{\beta}} = (\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_{18})'$

Trend-Surface Estimation: Example

Let's make the spatial basis fns with FRK::auto_basis()

```
G <- auto_basis(data = (Tmax_long[,c("lon","lat")] %>%
                          SpatialPoints()), # make Tmax a spp object
                nres = 1,
                type = "Gaussian")
```

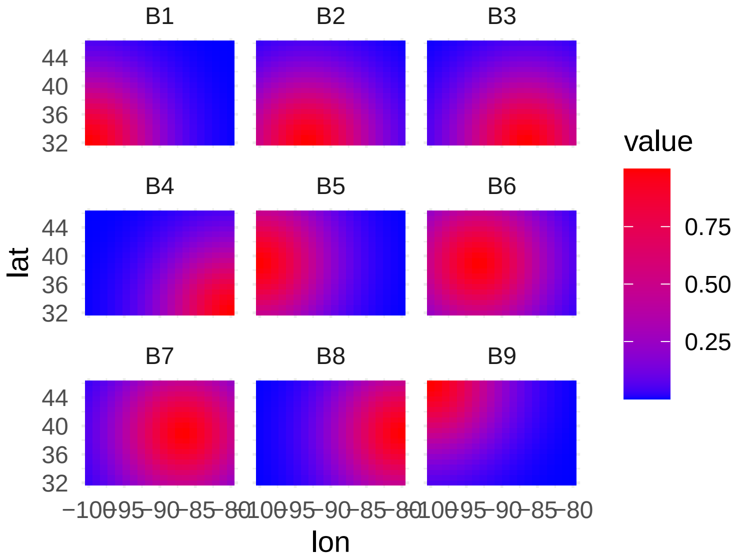
Evaluate basis fns at locations of interest

```
coords <- as.matrix(Tmax_long[,c("lon","lat")])
S <- eval_basis(basis = G,    # basis functions
                s =  coords   # eval at these locations
                ) %>%
  as.matrix()                 # conv. to matrix
colnames(S) <- paste0("B", 1:ncol(S))

Tmax2 <- cbind(Tmax_long, S) %>%
  dplyr::select(-year,-month,-proc,-julian,-date)
```

Trend-Surface Estimation: Example

Let's make the spatial basis fns with FRK::auto_basis()



Trend-Surface Estimation: Example

```
##
## Call:
## lm(formula = z ~ (lon + lat + day)^2 + ., data = dplyr::select(Tmax_no_14,
##     -id, -t))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.5136  -2.4797   0.1098   2.6644  14.1659
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 192.243242  97.854126   1.965 0.049531 *
## lon           1.756918   1.088175   1.615 0.106486
## lat          -1.317402   2.555626  -0.515 0.606239
## day          -1.216456   0.133547  -9.109  < 2e-16 ***
## B1           16.646617   4.832399   3.445 0.000577 ***
## B2           18.528159   3.056082   6.063 1.46e-09 ***
## B3           -6.606896   3.171759  -2.083 0.037312 *
## B4           30.545361   4.369591   6.990 3.20e-12 ***
## B5           14.739147   2.746866   5.366 8.52e-08 ***
## B6          -17.541177   3.423081  -5.124 3.13e-07 ***
## B7           28.472198   3.551900   8.016 1.42e-15 ***
## B8          -27.348145   3.164317  -8.643  < 2e-16 ***
## B9          -10.234777   4.456735  -2.296 0.021701 *
## B10          10.558234   3.327370   3.173 0.001519 **
## B11         -22.757661   3.532508  -6.442 1.32e-10 ***
## B12          21.864383   4.812940   4.543 5.72e-06 ***
## lon:lat      -0.026021   0.028232  -0.922 0.356755
## lon:day      -0.022696   0.001288 -17.615  < 2e-16 ***
## lat:day      -0.019032   0.001876 -10.147  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.225 on 3970 degrees of freedom
## Multiple R-squared:  0.7023, Adjusted R-squared:  0.701
```