Data_analysis

Justice Akuoko-Frimpong, Jonathan Ta, Edward Shao

2025-04-04

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 4.4.3
```

```
library(here)
```

```
## Warning: package 'here' was built under R version 4.4.3
```

```
## here() starts at C:/Lecture slides/Lecture slides/Lecture Slides/Winter 2025/Biostat 815/Quiz2/svc
```

```
load(here("data", "ASTER.RData"))
```

Exploratory data analysis

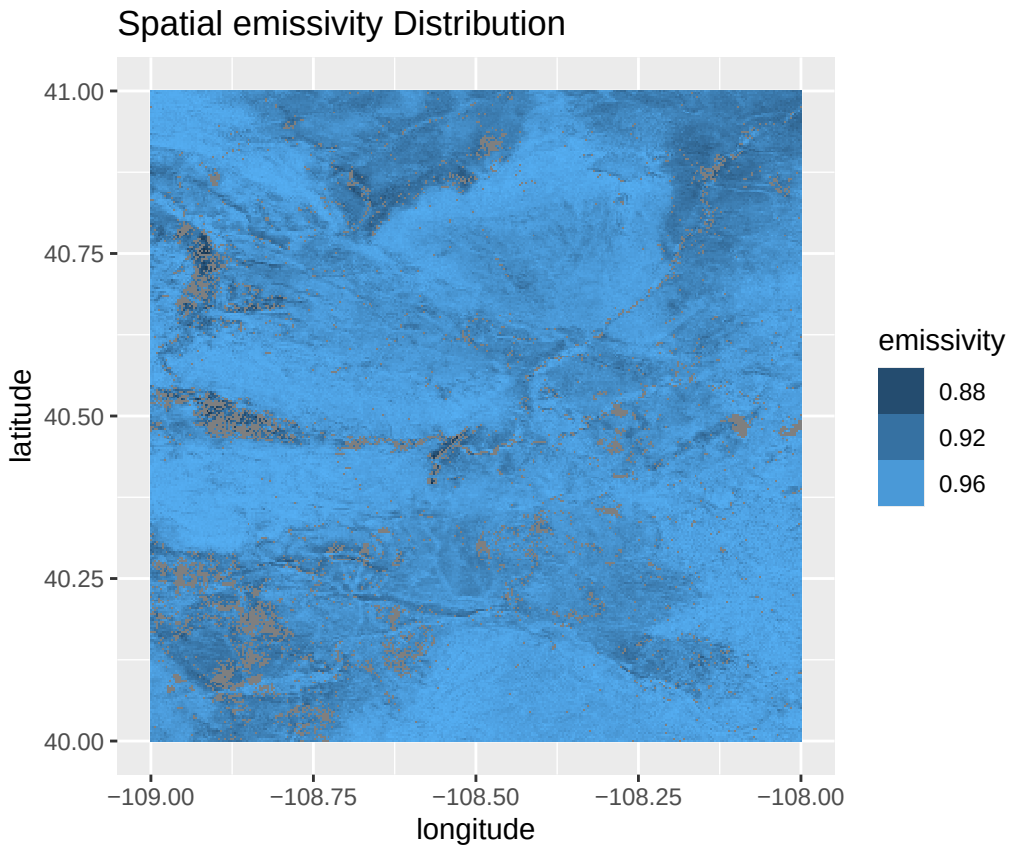
```
library(viridis)    # for the viridis color scale
```

```
## Loading required package: viridisLite
```

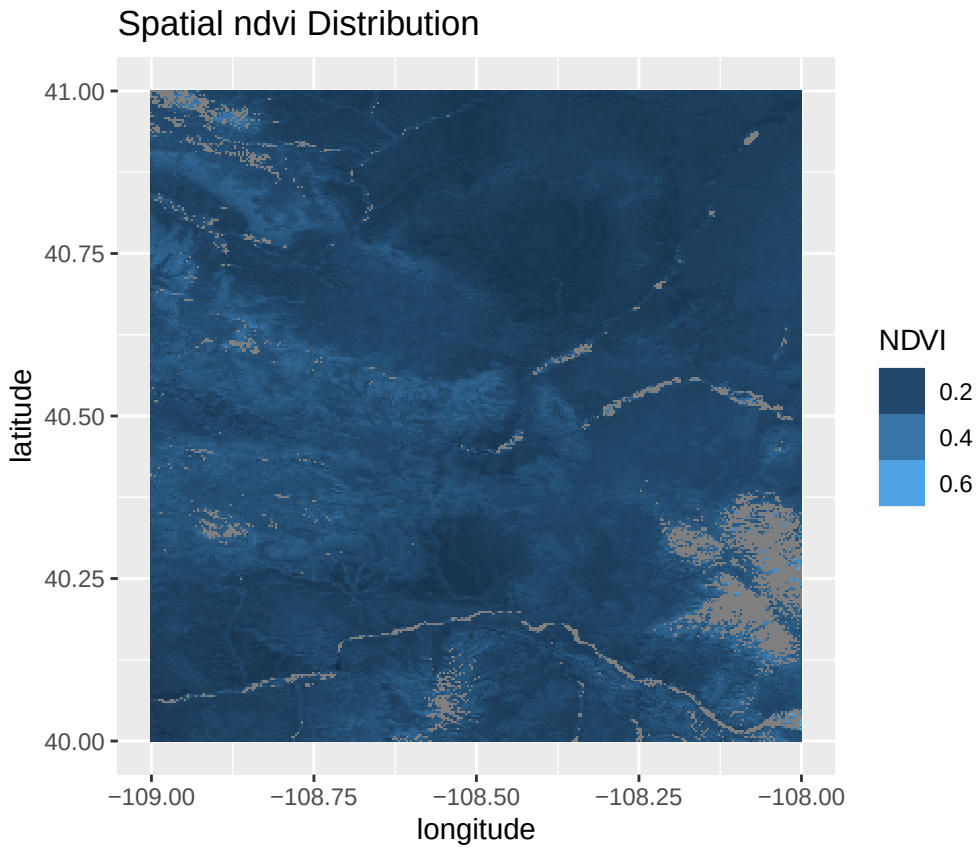
```
library(geoR)       # for calculating the empirical variogram
```

```
## ------------------------------------------------------------
##  Analysis of Geostatistical Data
##  For an Introduction to geoR go to http://www.leg.ufpr.br/geoR
##  geoR version 1.9-4 (built on 2024-02-14) is now loaded
## ------------------------------------------------------------
```

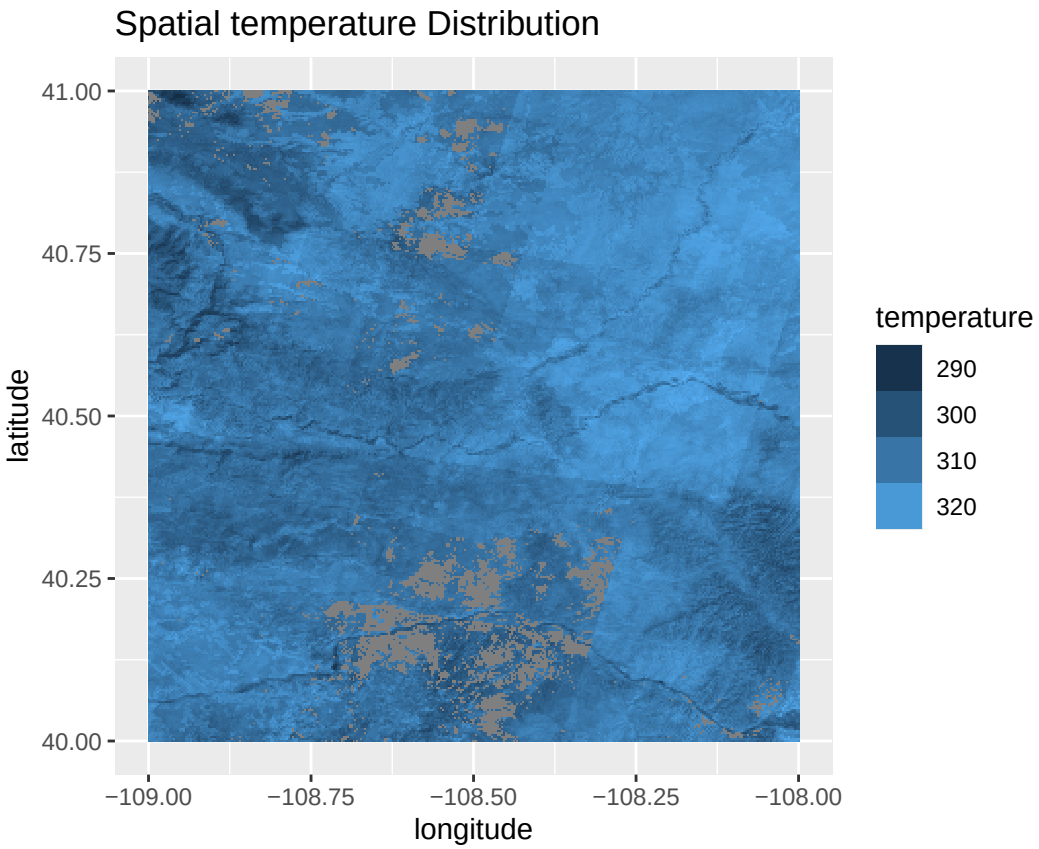
```
library(gridExtra)  # for arranging plots

# Spatial distribution of temperature
library(ggplot2)
ggplot(dt, aes(lon, lat)) +
  geom_tile(aes(fill = emis)) +
  xlab("longitude") +
  ylab("latitude") +
  guides(fill = guide_legend(title = "emissivity")) +
  ggtitle("Spatial emissivity Distribution") +
  coord_fixed()
```




```
ggplot(dt, aes(lon, lat)) +
  geom_tile(aes(fill = ndvi)) +
  xlab("longitude") +
  ylab("latitude") +
  guides(fill = guide_legend(title = "NDVI")) +
  ggtitle("Spatial ndvi Distribution") +
  coord_fixed()
```




```
ggplot(dt, aes(lon, lat)) +
  geom_tile(aes(fill = temp)) +
  xlab("longitude") +
  ylab("latitude") +
  guides(fill = guide_legend(title = "temperature")) +
  ggtitle("Spatial temperature Distribution") +
  coord_fixed()
```



Semivariograms

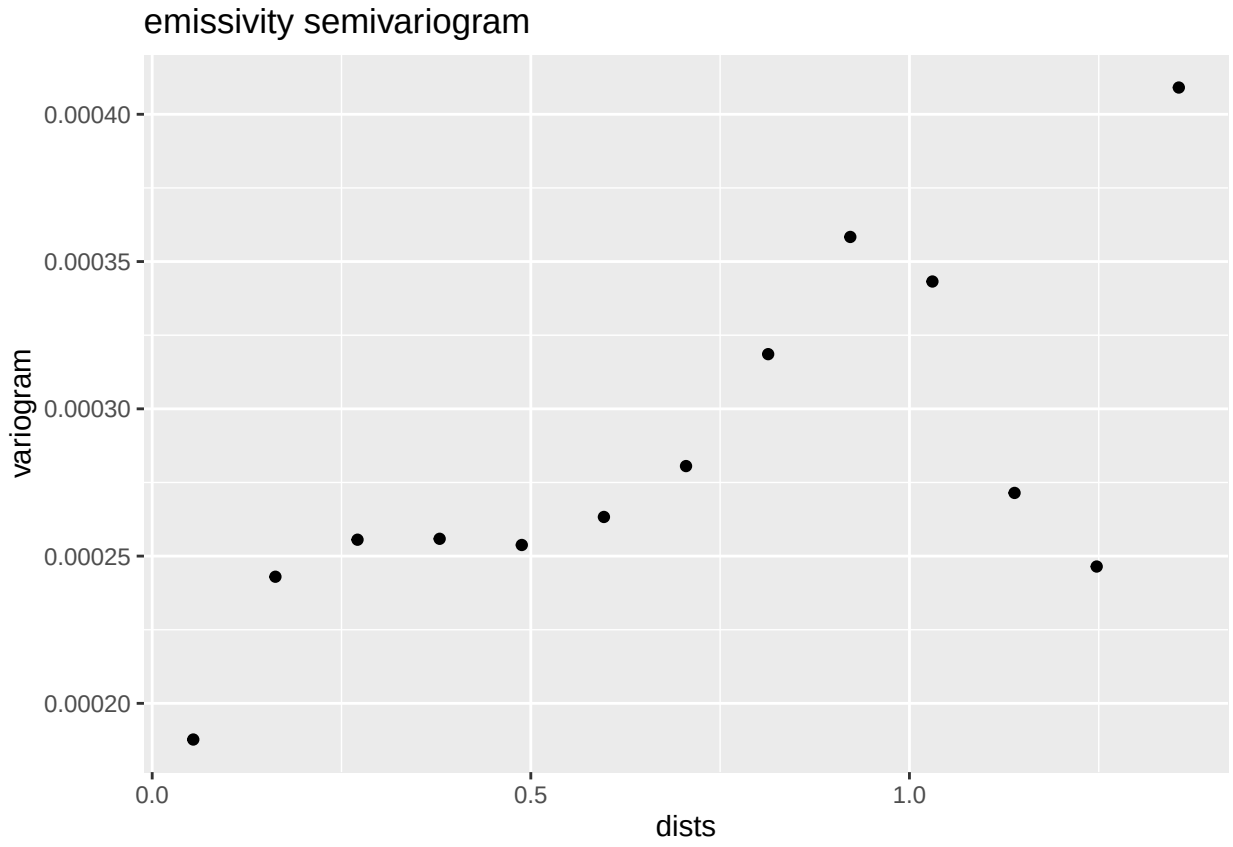
```
# subset data further for more efficient semivariogram analysis
lat.sub = dt[, unique(lat)]
lat.sub = lat.sub[seq(1, length(lat.sub), by = 2)]

lon.sub = dt[, unique(lon)]
lon.sub = lon.sub[seq(1, length(lon.sub), by = 2)]

dt.sub = dt[lat %in% lat.sub & lon %in% lon.sub]
dt.sub = dt.sub[complete.cases(dt.sub[, c("lat", "lon", "ndvi", "temp")]), ]
```

```
# semivariogram of emissivity
sv = variog(
  coords = cbind(dt.sub[!is.na(emis), lat], dt.sub[!is.na(emis), lon]),
  data = dt.sub[!is.na(emis), emis]
  )
```

```
## variog: computing omnidirectional variogram
```

```
sv.dt = data.table(
  dists = sv$u,
  variogram = sv$v,
  npairs = sv$n,
  sd = sv$sd
)
```

```
ggplot(sv.dt, aes(x = dists, y = variogram)) +
  geom_point() +
  labs(title = "emissivity semivariogram")
```



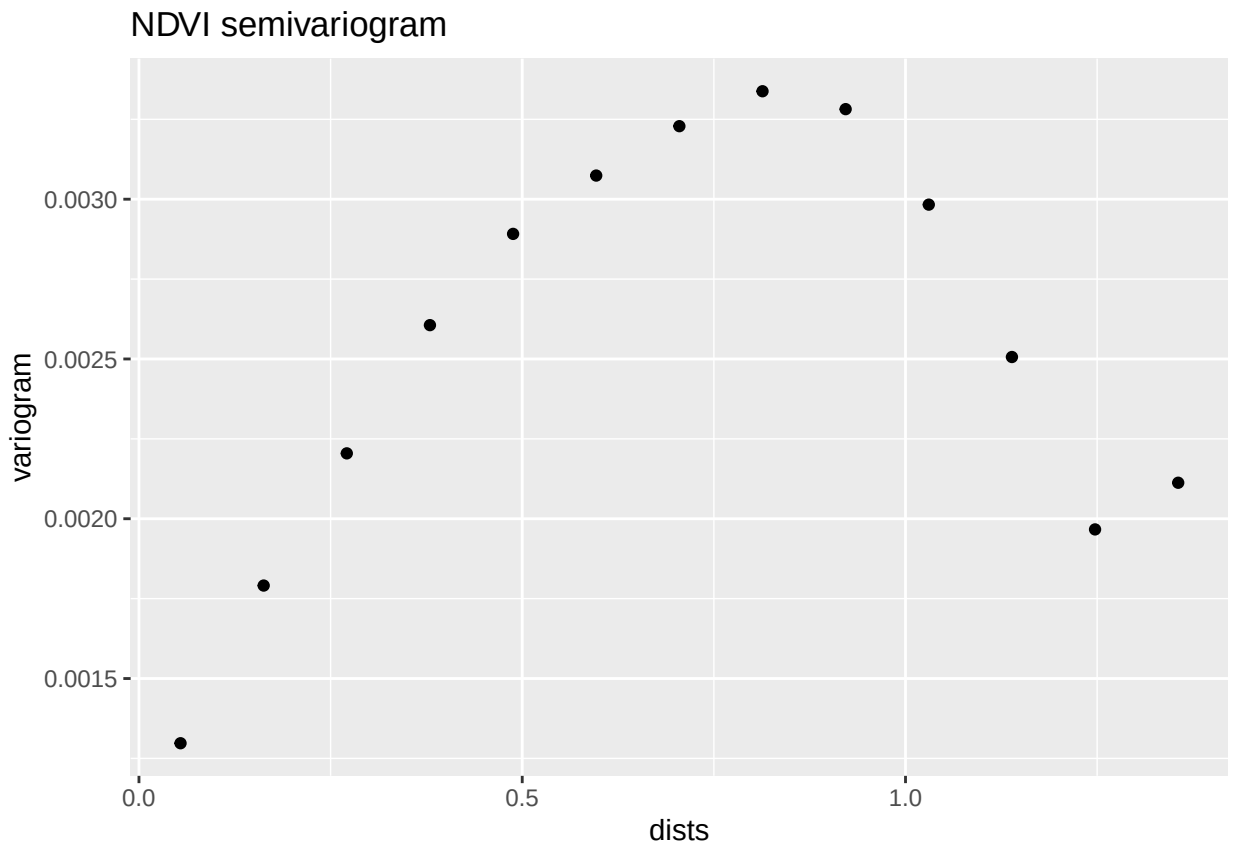

```
# identifying min and max distances
summary(sv.dt)
```

```
##      dists           variogram            npairs              sd
##  Min.   :0.05423   Min.   :0.0001877   Min.   :   32415   Min.   :0.0003170
##  1st Qu.:0.37961   1st Qu.:0.0002538   1st Qu.: 6795116   1st Qu.:0.0003826
##  Median :0.70498   Median :0.0002633   Median :25738477   Median :0.0003929
##  Mean   :0.70498   Mean   :0.0002836   Mean   :22252725   Mean   :0.0003961
##  3rd Qu.:1.03036   3rd Qu.:0.0003186   3rd Qu.:36813489   3rd Qu.:0.0004039
##  Max.   :1.35574   Max.   :0.0004091   Max.   :43946889   Max.   :0.0004684
```

```
# semivariogram of NDVI
sv = variog(
  coords = cbind(dt.sub[, lat], dt.sub[, lon]),
  data = dt.sub[, ndvi]
  )
```

```
## variog: computing omnidirectional variogram
```

```
sv.dt = data.table(
  dists = sv$u,
  variogram = sv$v,
  npairs = sv$n,
  sd = sv$sd
)
ggplot(sv.dt, aes(x = dists, y = variogram)) +
  geom_point() +
  labs(title = "NDVI semivariogram")
```



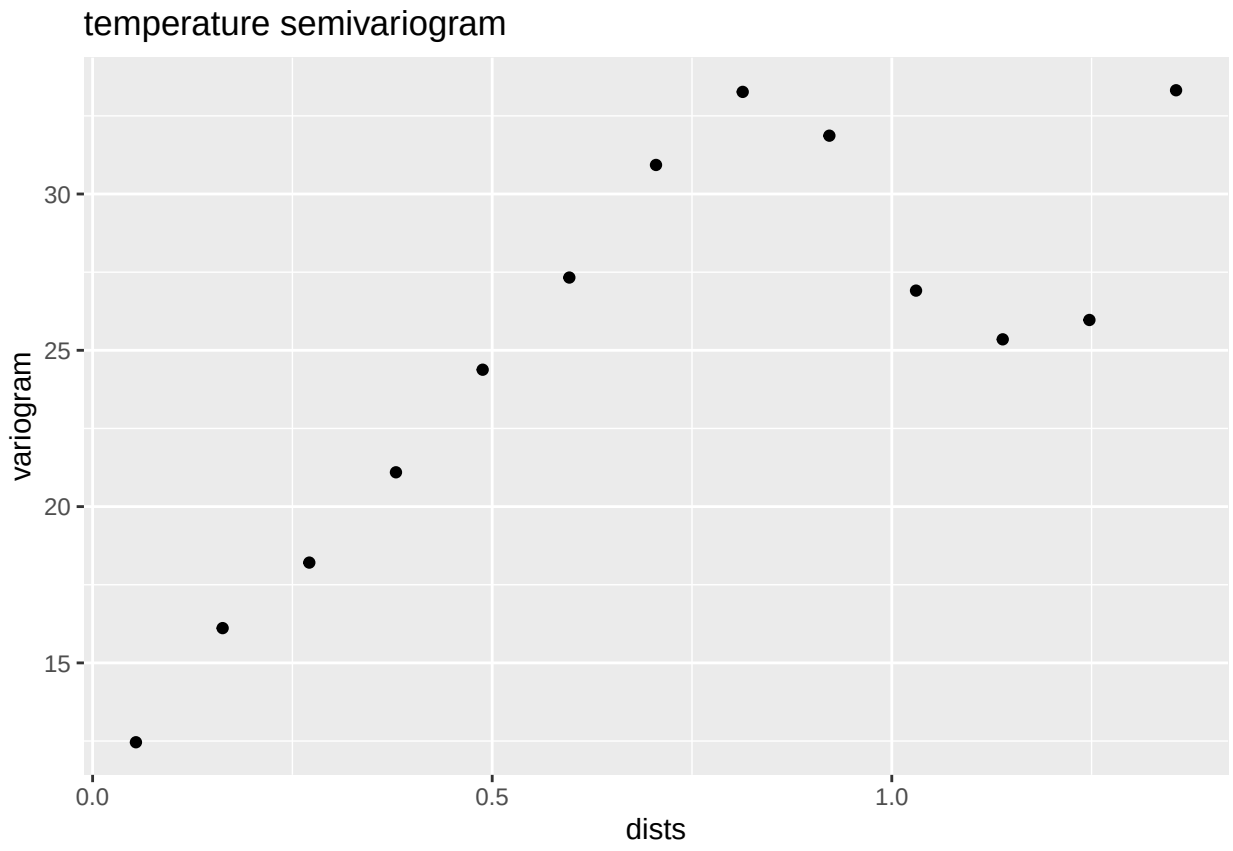

```
# identifying min and max distances
summary(sv.dt)
```

```
##      dists           variogram           npairs              sd
##  Min.   :0.05423   Min.   :0.001297   Min.   :   32633   Min.   :0.002737
##  1st Qu.:0.37961   1st Qu.:0.002113   1st Qu.: 7568325   1st Qu.:0.004350
##  Median :0.70498   Median :0.002606   Median :28411563   Median :0.005389
##  Mean   :0.70498   Mean   :0.002560   Mean   :24608143   Mean   :0.005285
##  3rd Qu.:1.03036   3rd Qu.:0.003074   3rd Qu.:40464068   3rd Qu.:0.006355
##  Max.   :1.35574   Max.   :0.003338   Max.   :48530435   Max.   :0.007990
```

```
# semivariogram of temperature
sv = variog(
  coords = cbind(dt.sub[, lat], dt.sub[, lon]),
  data = dt.sub[, temp]
  )
```

```
## variog: computing omnidirectional variogram
```

```
sv.dt = data.table(
  dists = sv$u,
  variogram = sv$v,
  npairs = sv$n,
  sd = sv$sd
)
 ggplot(sv.dt, aes(x = dists, y = variogram)) +
  geom_point() +
  labs(title = "temperature semivariogram")
```




```
# identifying min and max distances
summary(sv.dt)
```

```
##      dists           variogram         npairs              sd
##  Min.   :0.05423   Min.   :12.46   Min.   :   32633   Min.   :20.49
##  1st Qu.:0.37961   1st Qu.:21.10   1st Qu.: 7568325   1st Qu.:28.34
##  Median :0.70498   Median :25.97   Median :28411563   Median :34.47
##  Mean   :0.70498   Mean   :25.17   Mean   :24608143   Mean   :33.61
##  3rd Qu.:1.03036   3rd Qu.:30.93   3rd Qu.:40464068   3rd Qu.:39.26
##  Max.   :1.35574   Max.   :33.32   Max.   :48530435   Max.   :45.79
```

Data preparation

```
load(here("data", "ASTER.RData"))
```

```
library(svc)

complete_cases <- complete.cases(dt)
dt_complete <- dt[complete_cases, ]

# Create standardized inputs
Y <- dt_complete$temp
X <- scale(as.matrix(dt_complete[, c("ndvi", "emis")]))
s <- scale(as.matrix(dt_complete[, c("lon", "lat")]))

# Knot creation function with checks
create_knots <- function(coords, k = 50) {
  if(!all(c("lat", "lon") %in% colnames(coords))) {
    stop("coords must contain 'lat' and 'lon' columns")
  }

  lat_knots <- unique(na.omit(coords[,"lat"]))
  lon_knots <- unique(na.omit(coords[,"lon"]))

  lat_knots <- lat_knots[seq(1, length(lat_knots), by = k)]
  lon_knots <- lon_knots[seq(1, length(lon_knots), by = k)]

  knots <- as.matrix(expand.grid(lat_knots, lon_knots))
  colnames(knots) <- c("lat", "lon")
  return(knots)
}

# Create and validate knots
coords <- as.data.frame(dt_complete[, c("lat", "lon")])
knots <- create_knots(coords, k = 50)
knots_df <- merge(data.frame(knots), dt_complete, by = c("lat", "lon"), all.x = TRUE)

complete_knots <- complete.cases(knots_df[, c("temp", "ndvi", "emis")])

# Filter all components to only keep complete cases
knots_df <- knots_df[complete_knots, ]
Y_knots <- knots_df$temp
X_knots <- as.matrix(knots_df[, c("ndvi", "emis")])
knots_matrix <- as.matrix(knots_df[, c("lon", "lat")])

# Now scale the complete data
X_knots <- scale(X_knots)
knots_matrix <- scale(knots_matrix)
```

Model initilizations

```r
# Initial values with small phi's for stability
init_values <- list(
  beta_knots_start = matrix(rnorm(nrow(knots_matrix)*ncol(X), sd = 0.1),
                            nrow = nrow(knots_matrix), ncol = ncol(X)),
  w_knots_start = rnorm(nrow(knots_matrix), sd = 0.1),
  phi_beta_start = rep(0.01, ncol(X)),  # Very small initial phi values
  phi_w_start = 0.01,
  sigmasq_beta_start = rep(0.1, ncol(X)),
  sigmasq_w_start = 0.1,
  tausq_start = 0.001
)

# bounds and small proposal sizes for phi's
tuning_params <- list(
  phi_beta_proposal_sd = rep(0.005, ncol(X)),  # Small steps
  phi_w_proposal_sd = 0.005,
  lower_beta = rep(0.001, ncol(X)),  # Tight bounds
  upper_beta = rep(0.5, ncol(X)),
  lower_w = 0.001,
  upper_w = 0.5
)

# Weakly informative priors
priors <- list(
  a_beta = rep(2, ncol(X)),  # Slightly informative
  b_beta = rep(1, ncol(X)),
  a_w = 2,
  b_w = 1,
  a_t = 2,
  b_t = 1
)
```

Model running

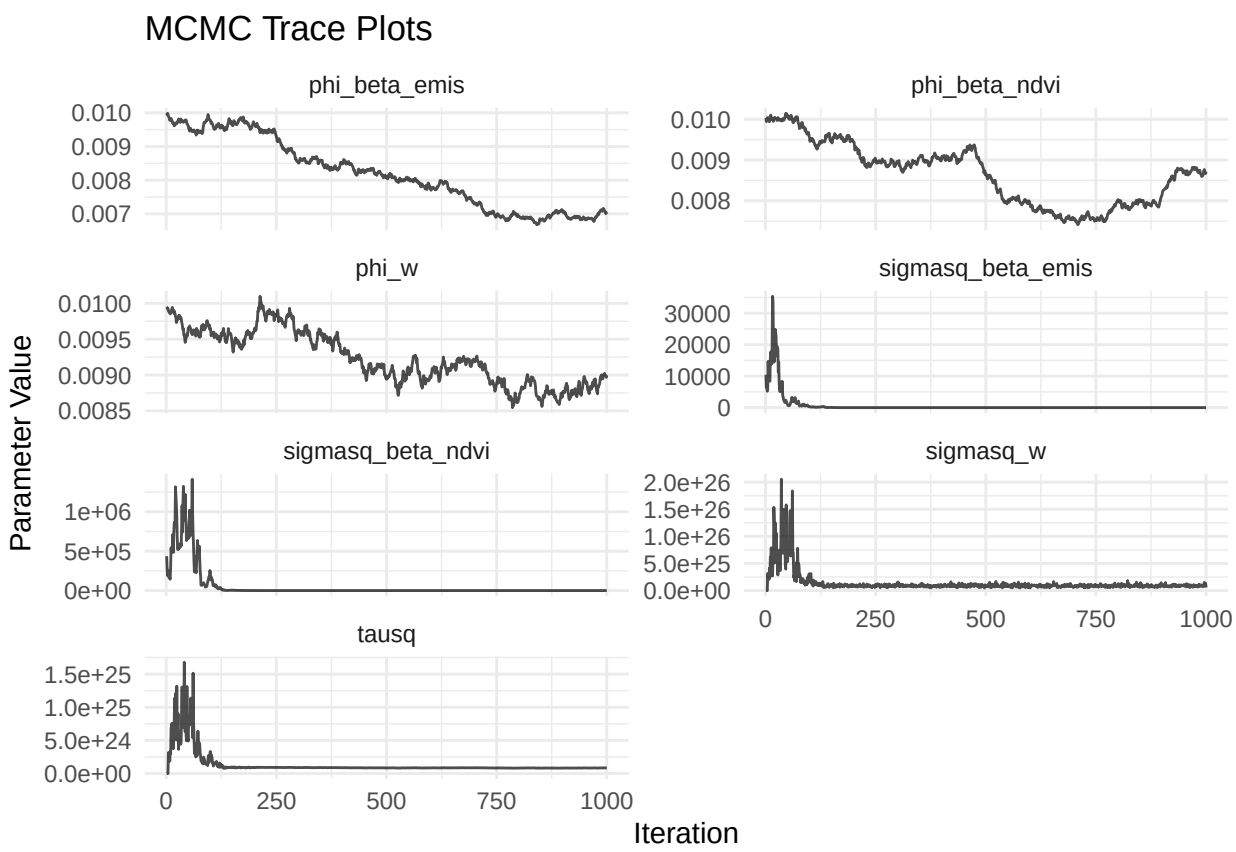
Trace plots

```r
library(ggplot2)
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 4.4.2
```

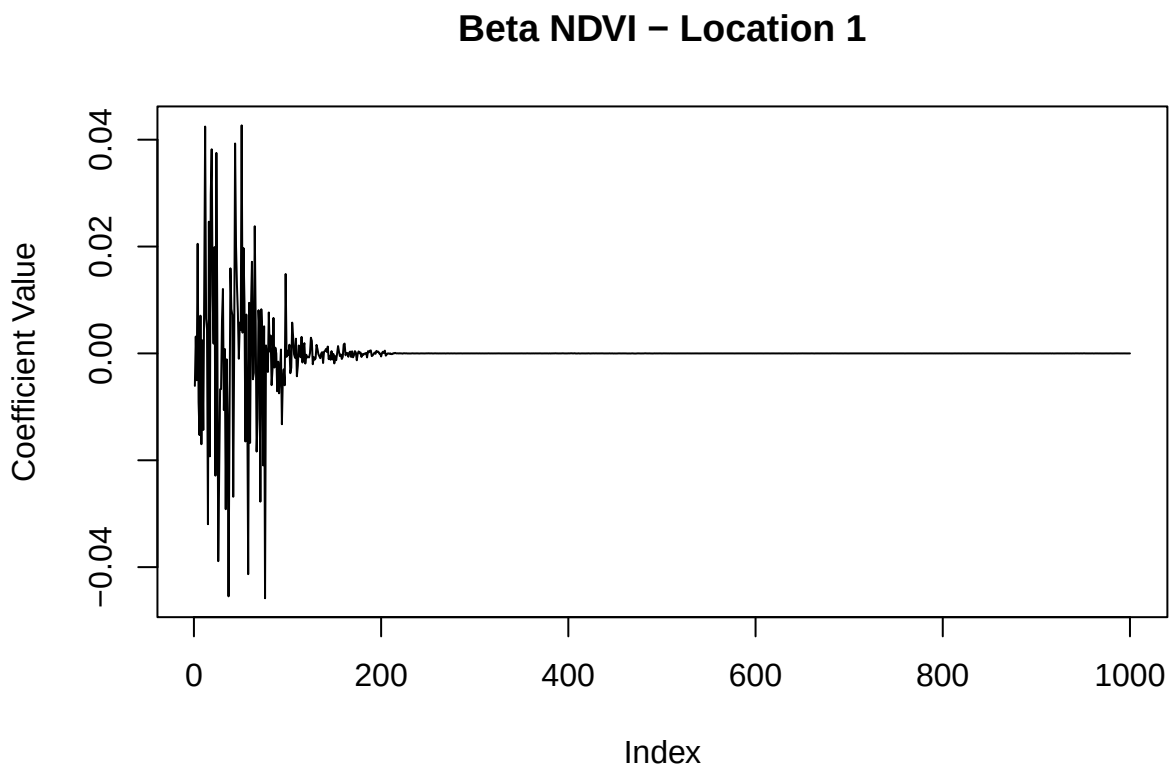
```r
# Convert samples to data frame
trace_data <- data.frame(
  iteration = 1:length(results$phi_w_samples),
  phi_beta_ndvi = results$phi_beta_samples[,1],
  phi_beta_emis = results$phi_beta_samples[,2],
  phi_w = results$phi_w_samples,
  sigmasq_beta_ndvi = results$sigmasq_beta_samples[,1],
  sigmasq_beta_emis = results$sigmasq_beta_samples[,2],
  sigmasq_w = results$sigmasq_w_samples,
```

```
  tausq = results$tausq_samples
)

# Create faceted trace plots
trace_long <- trace_data %>%
  pivot_longer(-iteration, names_to = "parameter", values_to = "value")

ggplot(trace_long, aes(x = iteration, y = value)) +
  geom_line(alpha = 0.7) +
  facet_wrap(~ parameter, scales = "free_y", ncol = 2) +
  labs(title = "MCMC Trace Plots", x = "Iteration", y = "Parameter Value") +
  theme_minimal()
```

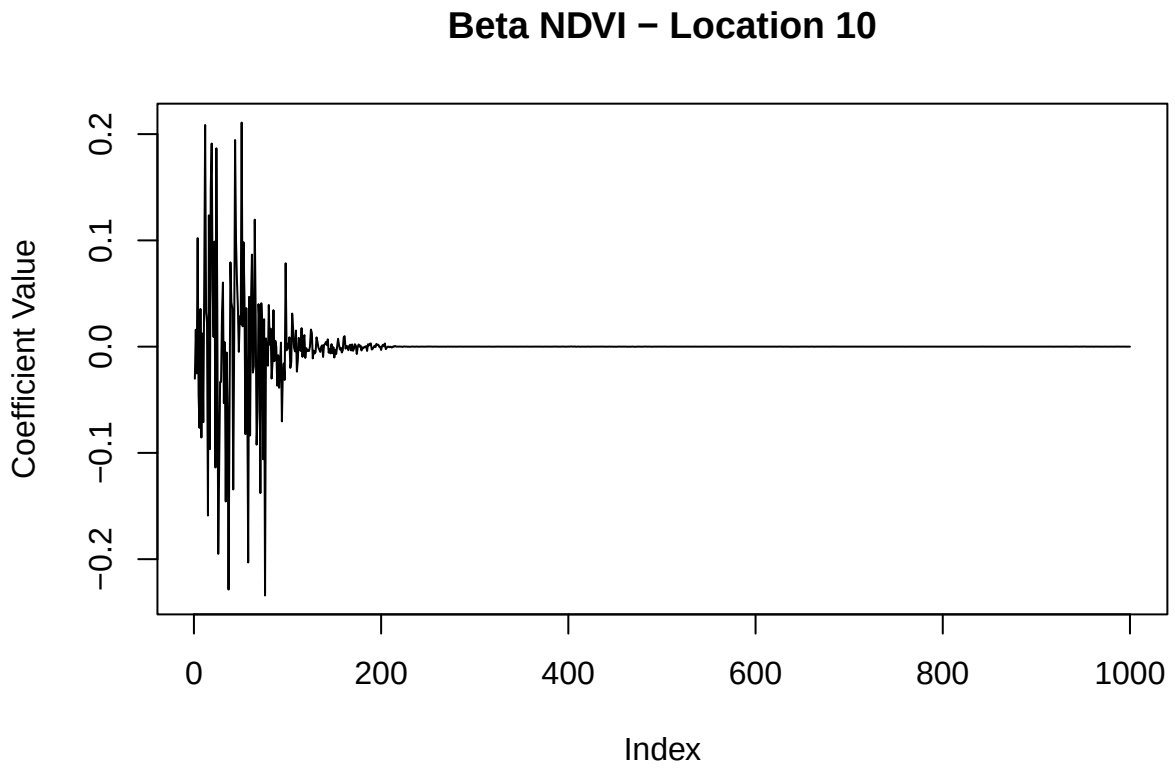


plots of beta at selected locations

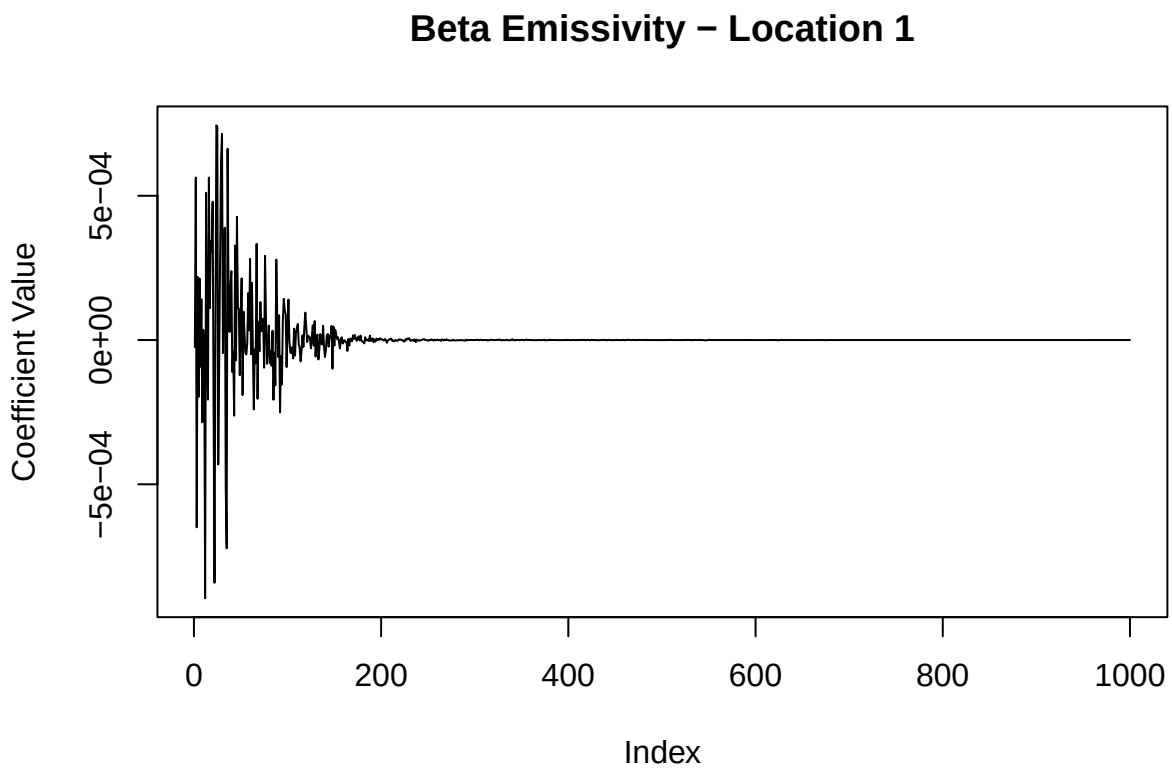
```
# Trace plots for NDVI coefficient at selected locations
plot(results$beta_samples[5, 1, ], type = 'l',
     main = "Beta NDVI - Location 1", ylab = "Coefficient Value")
```



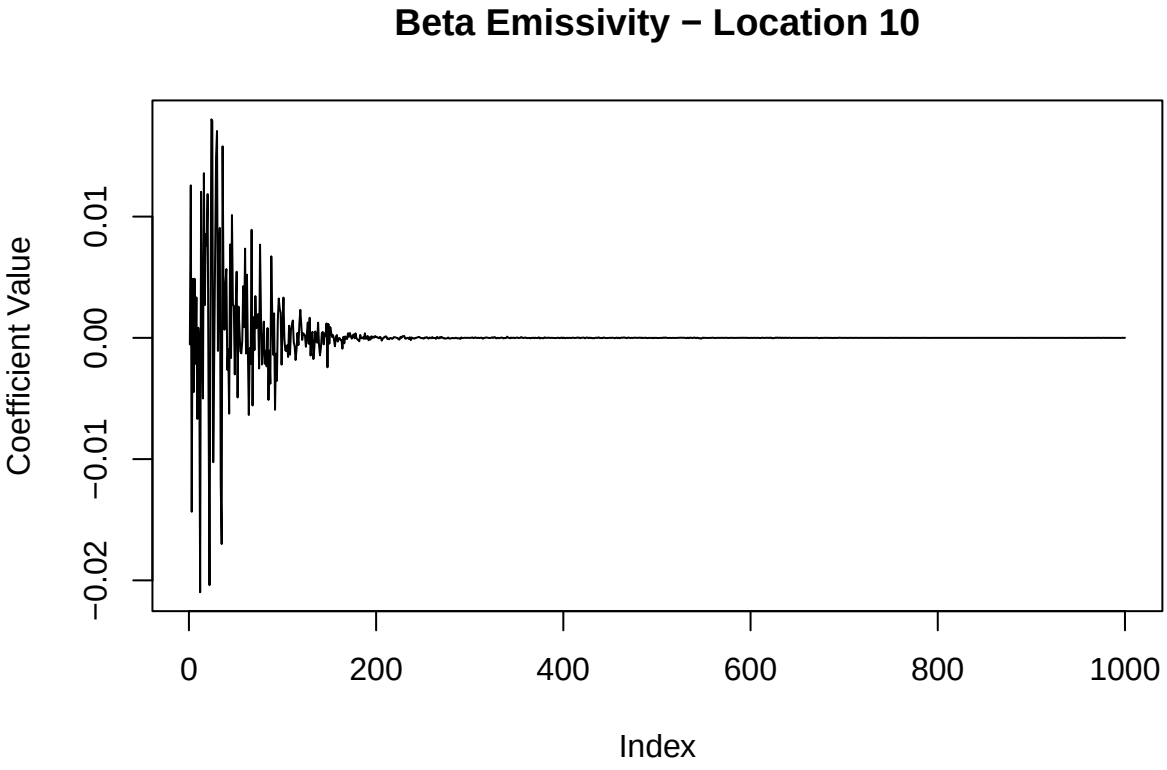

```r
plot(results$beta_samples[10, 1, ], type = 'l',
     main = "Beta NDVI - Location 10", ylab = "Coefficient Value")
```



```r
# Trace plots for Emissivity coefficient at selected locations
plot(results$beta_samples[1, 2, ], type = 'l',
     main = "Beta Emissivity - Location 1", ylab = "Coefficient Value")
```




```r
plot(results$beta_samples[10, 2, ], type = 'l',
     main = "Beta Emissivity - Location 10", ylab = "Coefficient Value")
```



Interpreations and Problems

1.) The covariate effects converged near zero, suggesting weak linear relationships between NDVI/emissivity and temperature in this system 2.) The large estimated variance components indicate substantial unexplained spatial autocorrelation

Together, these results suggest:

1. Identifiability Problem:

- The spatial random effects (w) may be absorbing all the spatial signal
- This leaves little variation for the covariate effects to explain

2. Problem with some specifications

Next steps:

1. Recode our function to perform similarly to the spBayes package's spatially varying coefficient (SVC) model