Introduction to spatiotemporal modeling in R

2024 Conference on Statistical Practice

Sara Venkatraman (Cornell University, Statistics and Data Science)

February 29, 2024

New Orleans, Louisiana
February 27–29 2024

Overview

0 — What does spatiotemporal data look like?

1 — Exploratory analysis of spatiotemporal data

- Data visualization: maps and time series plotting
- Global and local measures of spatial autocorrelation
- Assessing presence of spatial clusters

2 — Spatiotemporal regression modeling

- Modeling spatial dynamics
- Modeling spatial and temporal dynamics simultaneously
- Parameter interpretation

Code available to follow along with:

https://sara-venkatraman.github.io/SpatiotemporalAnalysis.html

What kinds of spatiotemporal data exist?

Spatiotemporal data is data that is collected over both space and time.

In space, we can have:

- **Lattice data:** observed on finite grid, like counties or ZIP codes
  (e.g. county-level populations in a state)
- **Geostatistical data:** observed at continuous spatial locations
  (e.g. temperature across the country)
- **Point process data:** observed at random locations
  (e.g. bird nest locations across a city)

https://sara-venkatraman.github.io/SpatiotemporalAnalysis.html

What kinds of spatiotemporal data exist?

Spatiotemporal data is data that is collected over both space and time.

In space, we can have:

- **Lattice data:** observed on finite grid, like counties or ZIP codes
  (e.g. county-level populations in a state)
- **Geostatistical data:** observed at continuous spatial locations
  (e.g. temperature across the country)
- **Point process data:** observed at random locations
  (e.g. bird nest locations across a city)

In time, we can have:

- **Deterministic** time points that are **regularly** or **irregularly** spaced
- **Random** time points

https://sara-venkatraman.github.io/SpatiotemporalAnalysis.html

What kinds of spatiotemporal data exist?

Spatiotemporal data is data that is collected over both space and time.

In space, we can have:

- **Lattice data:** observed on finite grid, like counties or ZIP codes
  (e.g. county-level populations in a state)
- **Geostatistical data:** observed at continuous spatial locations
  (e.g. temperature across the country)
- **Point process data:** observed at random locations
  (e.g. bird nest locations across a city)

In time, we can have:

- **Deterministic** time points that are **regularly** or **irregularly** spaced
- **Random** time points

https://sara-venkatraman.github.io/SpatiotemporalAnalysis.html

What do we need to study spatiotemporal data in R?

- **Spatiotemporal measurements:** often in tabular form, e.g. one row per location and one column per time point
- **Spatial shapefile:** collection of files describing a space's geometry in polygons/lines/points. Will need this for plotting on maps and modeling.
- **R packages:** `rgdal` (`readOGR()` function reads `.shp` shapefiles), `spdep`, `maptools`, `ggplot2`

Example dataset: COVID-19 hospitalizations in NYC, March 2020

- **Likely COVID-19 hospitalizations from each NYC ZIP code each day in March 2020** **(source: NYC Dept. of Health and Mental Hygiene)**

```
# Read the COVID case data from a CSV
NYC_COVID <- read.csv("Data/ZipcodeHospitalizations.csv", row.names=1)

# Print first 5 rows and first 7 columns of this dataset
head(NYC_COVID, n=c(5,7))
```

```
      2020.03.01 2020.03.02 2020.03.03 2020.03.04 2020.03.05 2020.03.06 2020.03.07
10001         10         11          6         11          8          5          7
10002          9         19         23         17         15         18         21
10003          3          8          9          6          7          9          7
10004          0          0          3          0          2          1          0
10005          0          0          0          4          0          2          0
 ...         ...                   ...                   ...
```

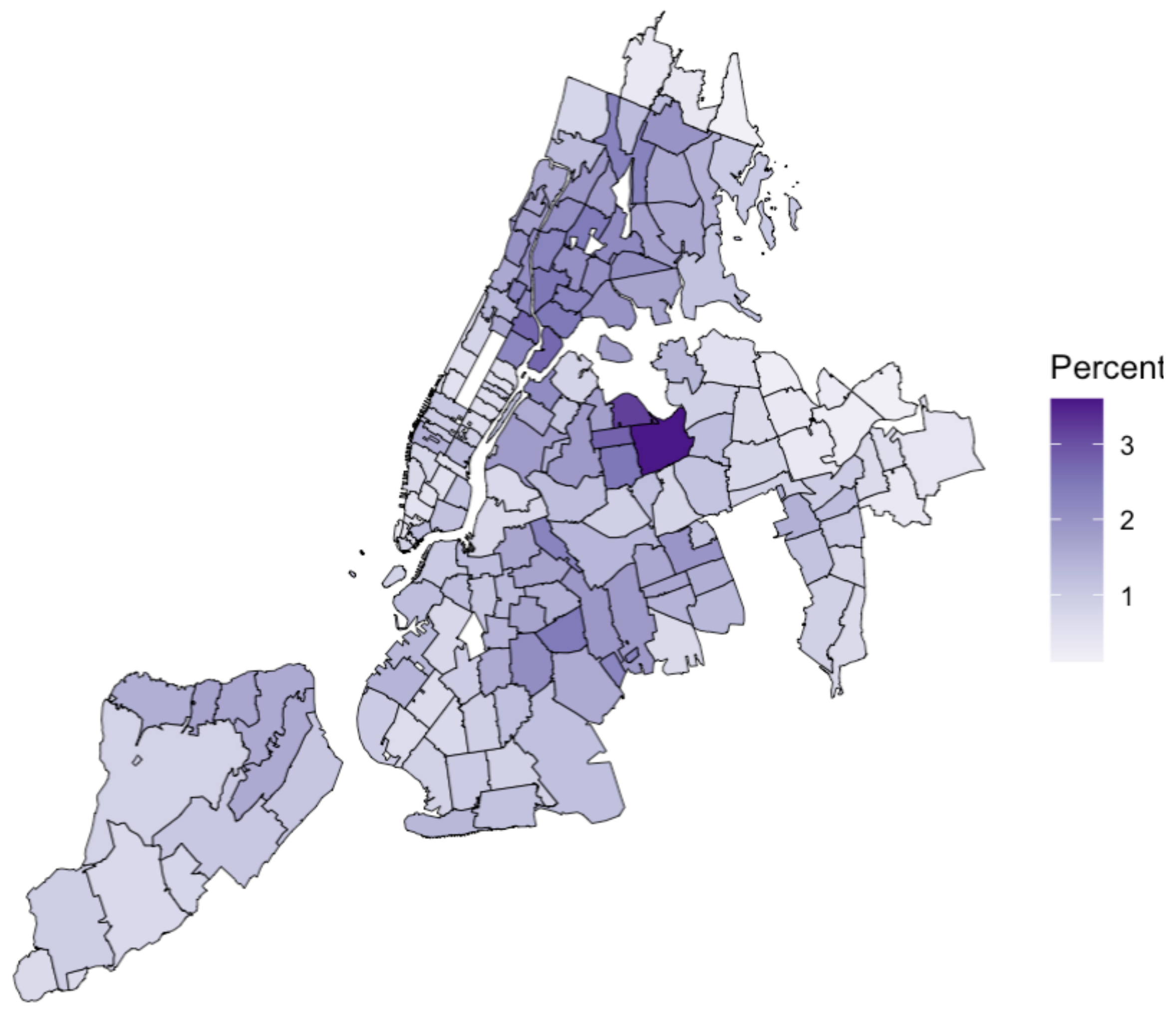
- NYC shapefile with ZIP code boundaries (source: NYC Open Data)

```
library(rgdal)
NYC_SHP <- readOGR("Shapefiles/tl_2010_36_zcta510NYC.shp")
```

Data visualization: choropleth maps

```
# Compute cumulative case counts
COVID_cumulative <- data.frame(zip = zips,
  Percent = 100 * rowSums(NYC_COVID) / Population)

# Merge case counts with map data
plotData <- merge(plotData,
                  COVID_cumulative,
                  by.x="id", by.y="zip")

# Draw the map
ggplot(plotData, aes(x=long, y=lat)) +
  geom_polygon(aes(group=group, fill=Percent)) +
  scale_fill_distiller(palette="Purples")
```

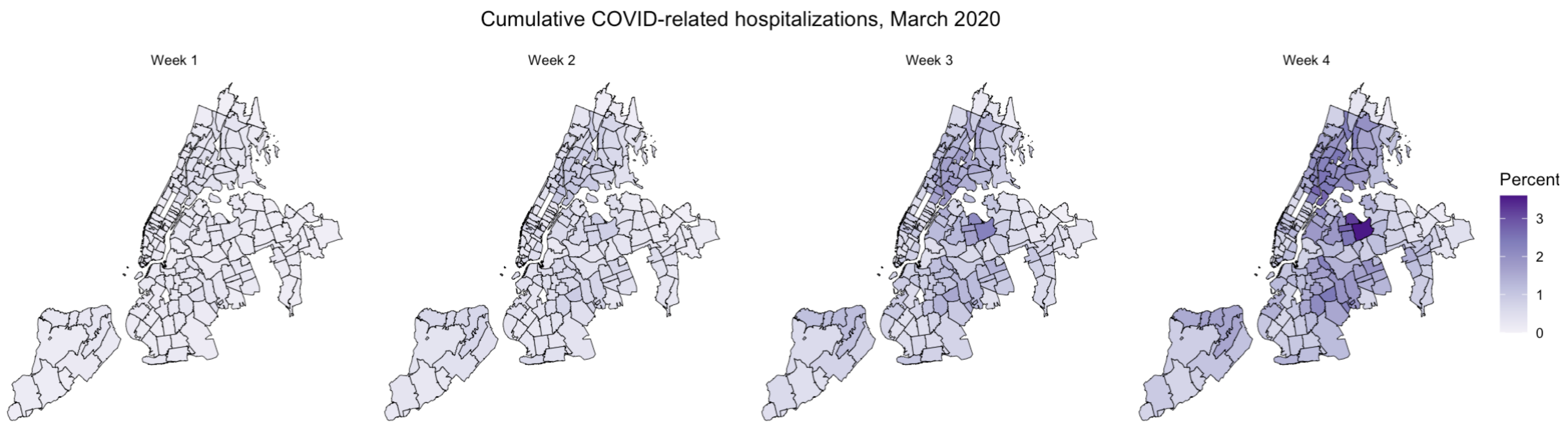
Cumulative COVID-related hospitalizations, March 2020



Surround the plot command with the `ggplotly()` function from the `plotly` package to make it interactive.

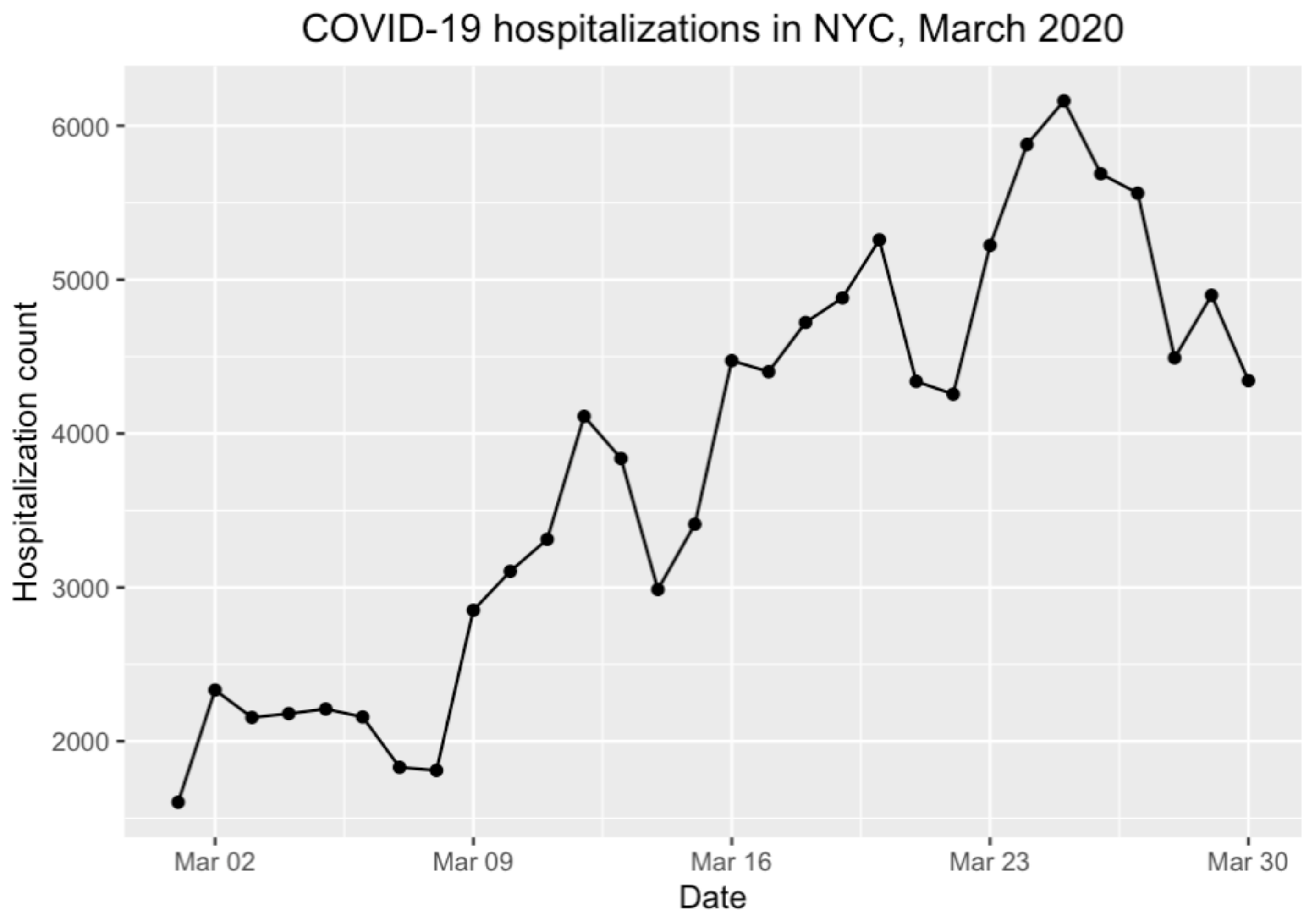
Choropleth maps at successive time points

```
# Compute cumulative case counts up to 8, 15, 23, and 30 days
COVID_cumulative <- data.frame(zip=zips,
           sapply(c(8, 15, 23, 30), function(x) 100 * rowSums(NYC_COVID[,1:x])) / Population)

# Merge case counts with map data
plotData <- merge(plotData, COVID_cumulative, by.x=“id", by.y="zip")

# Draw the maps
ggplot(plotData, aes(x=long, y=lat)) + geom_polygon(aes(group=group, fill=Percent)) +
   facet_wrap(~ Week, ncol=2) + scale_fill_distiller(palette=“Purples")
```



Plotting city-wide variation over time

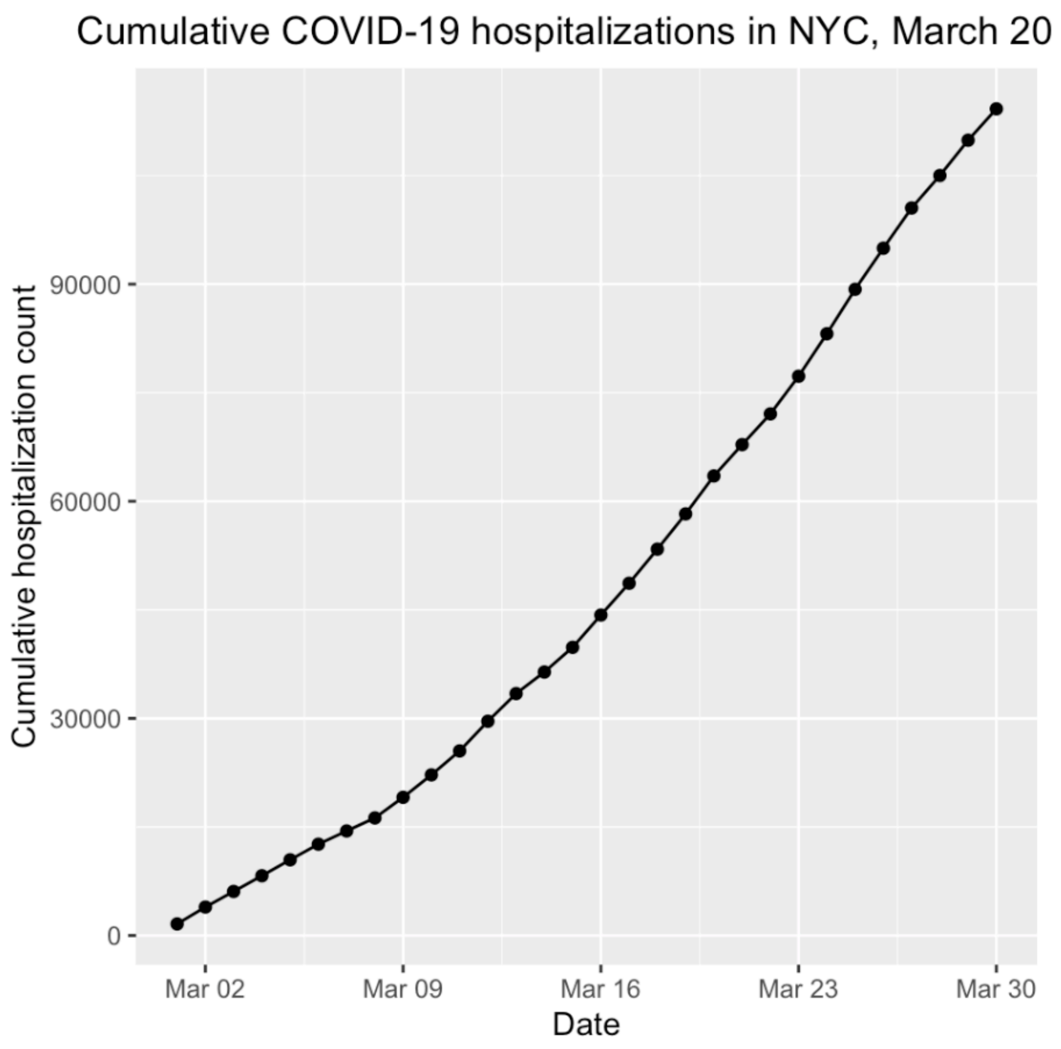
```
# Compute daily total of hospitalizations across NYC (not cumulative)
plotData <- data.frame(date=dates, count=colSums(NYC_COVID))

ggplot(plotData, aes(x=date, y=count)) +
  geom_point() + geom_line() +
  labs(x="Date", y="Hospitalization count”)
```



Plotting city-wide variation over time

```r
# Compute daily total of hospitalizations across NYC (not cumulative)
plotData <- data.frame(date=dates, count=colSums(NYC_COVID))

ggplot(plotData, aes(x=date, y=count)) +
  geom_point() + geom_line() +
  labs(x="Date", y="Hospitalization count”)
```

We can also look at the cumulative hospitalizations over time and consider how the rate at which they increase changes (or does not)



Spatial autocorrelation: Moran’s I-statistic

We’d like to understand spatial dependence:

How similar are observations to those that are spatially nearby?

In this context:

Is a ZIP’s COVID case count similar to those in neighboring ZIPs?

The Moran’s I statistic is a common measure of spatial dependence:

Spatial autocorrelation: Moran's I-statistic

We'd like to understand spatial dependence:

How similar are observations to those that are spatially nearby?

In this context:

Is a ZIP's COVID case count similar to those in neighboring ZIPs?

The Moran's I statistic is a common measure of spatial dependence:

If $x_1, \ldots, x_N$ are observations (COVID cases) at N spatial points (ZIP codes),

$$\text{Moran's } I = \frac{\sum_{i=1}^{N} \text{covariance between loc. } i \text{ and its neighbors}}{\text{variance over all locations}}$$

- Between -1 (negative correlation) and 1 (positive correlation). **Can test for significance.**

Spatial autocorrelation: Moran's I-statistic

We'd like to understand spatial dependence:

How similar are observations to those that are spatially nearby?

In this context:

Is a ZIP's COVID case count similar to those in neighboring ZIPs?

The Moran's I statistic is a common measure of spatial dependence:

If $x_1, \ldots, x_N$ are observations (COVID cases) at N spatial points (ZIP codes),

$$\text{Moran's } I = \frac{\sum_{i=1}^{N} \text{covariance between loc. } i \text{ and its neighbors}}{\text{variance over all locations}} = \frac{N \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{W \sum_{i=1}^{N} (x_i - \bar{x})^2}$$

- Between -1 (negative correlation) and 1 (positive correlation). **Can test for significance.**
- $\bar{x}$ = overall average
- w_{ij} = 1 if locations i and j are adjacent, 0 otherwise, and $W =$ sum of all w_{ij}

Spatial autocorrelation: Moran’s I-statistic

Compute Moran’s I associated with the total case count in each ZIP:

```
# Identify each ZIP’s neighbors using poly2nb() from spdep package
NYC_neighbors <- poly2nb(NYC_SHP)

# Compute Moran's I-statistic
moran.test(rowSums(NYC_COVID), NYC_neighbors)
```

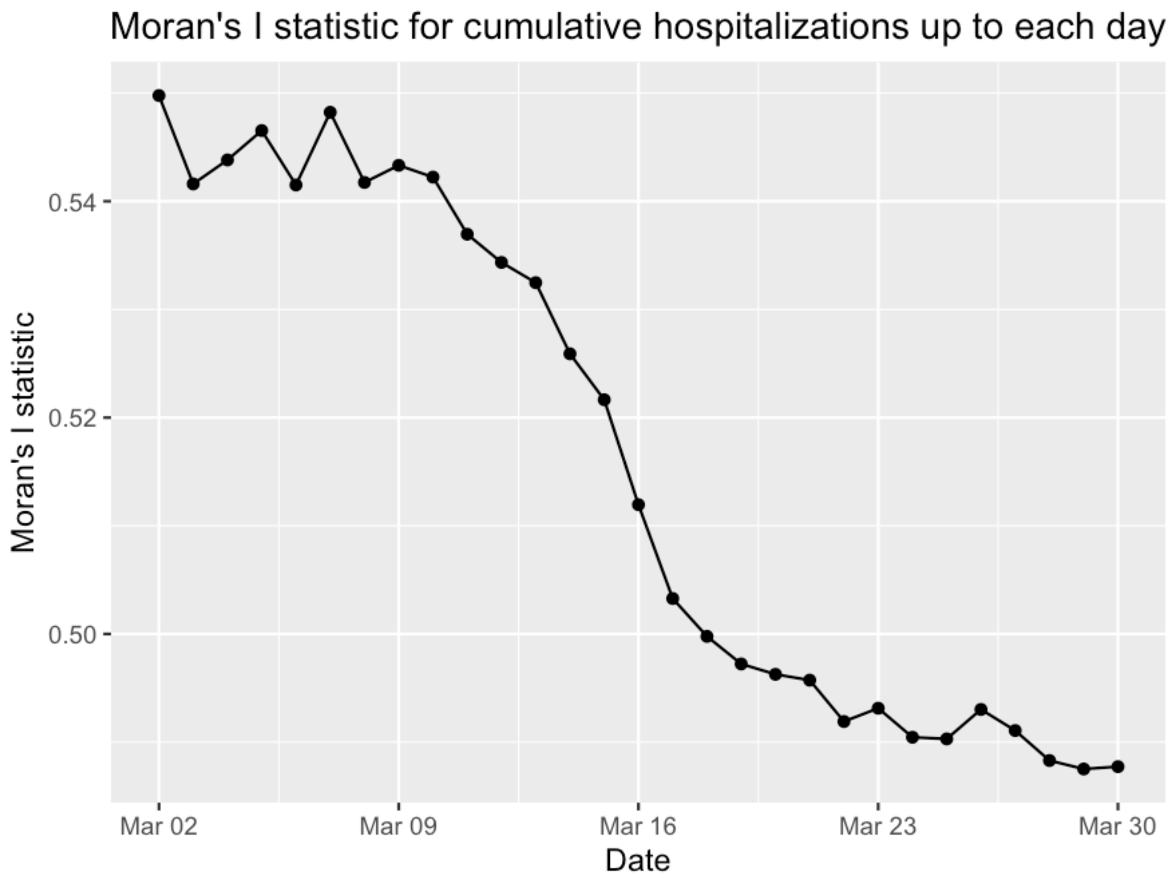
```
              Moran I test under randomisation

data:  rowSums(NYC_COVID)
weights: NYC_neighbors

Moran I statistic standard deviate = 9.488, p-value < 2.2e-16
alternative hypothesis: greater
sample estimates:
Moran I statistic       Expectation          Variance
      0.487735784      -0.005813953       0.002705903
```

Spatial autocorrelation: Moran’s I-statistic

How does the Moran’s I-statistic **vary over time?**

We’ll compute the statistic for cumulative cases up to **each time point.**

```
# Initialize data frame for storing daily Moran’s I
cumulative_moran <- data.frame(date=dates, statistic=0)

for(i in 1:30) {
  COVID_moran_day <-
     moran.test(rowSums(NYC_COVID[,1:i]), NYC_neighbors)
  cumulative_moran[i,2] <- COVID_moran_day$estimate[1]
}

ggplot(cumulative_moran, aes(x=date, y=statistic))
   + geom_point() + geom_line()
```



Other measures of spatial dependence include: Getis-Ord statistic, Geary’s C-statistic

Local spatial autocorrelation

Moran’s I is a **global** statistic that summarizes an entire area:
it might not capture varying strengths in spatial dependence.

The local Moran’s I can be used to find localized clusters
(e.g., clusters of ZIPs with COVID outbreaks).

Local spatial autocorrelation

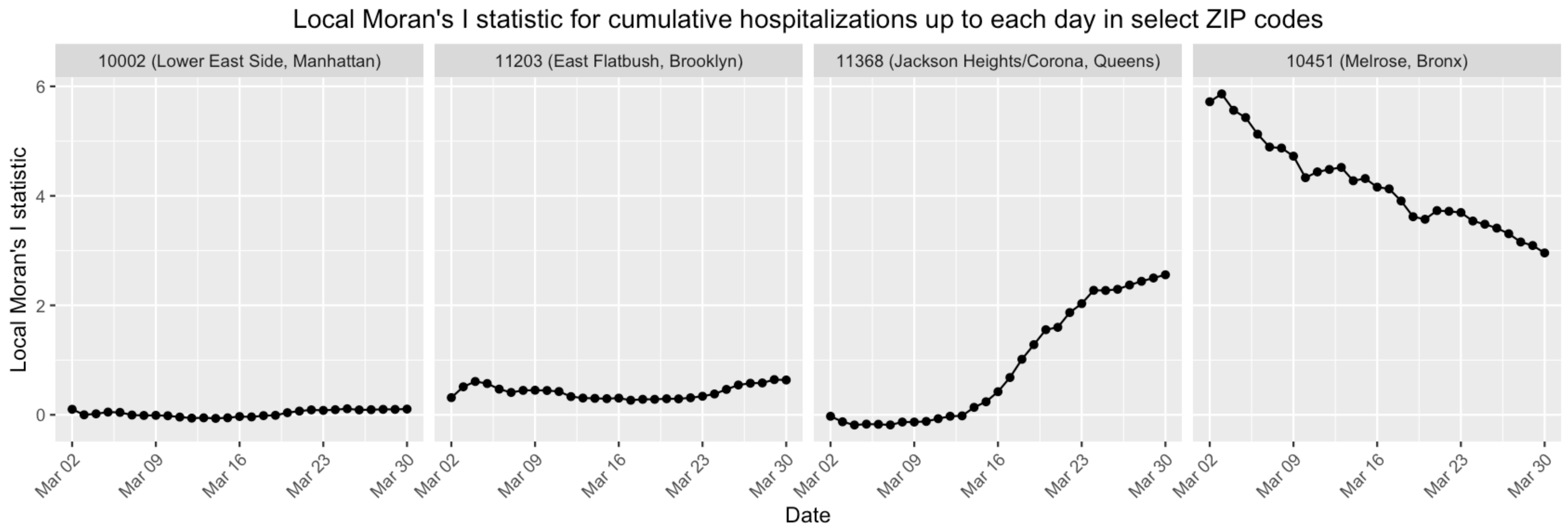
Moran's I is a **global** statistic that summarizes an entire area:
it might not capture varying strengths in spatial dependence.

The local Moran's I can be used to find localized clusters
(e.g., clusters of ZIPs with COVID outbreaks).

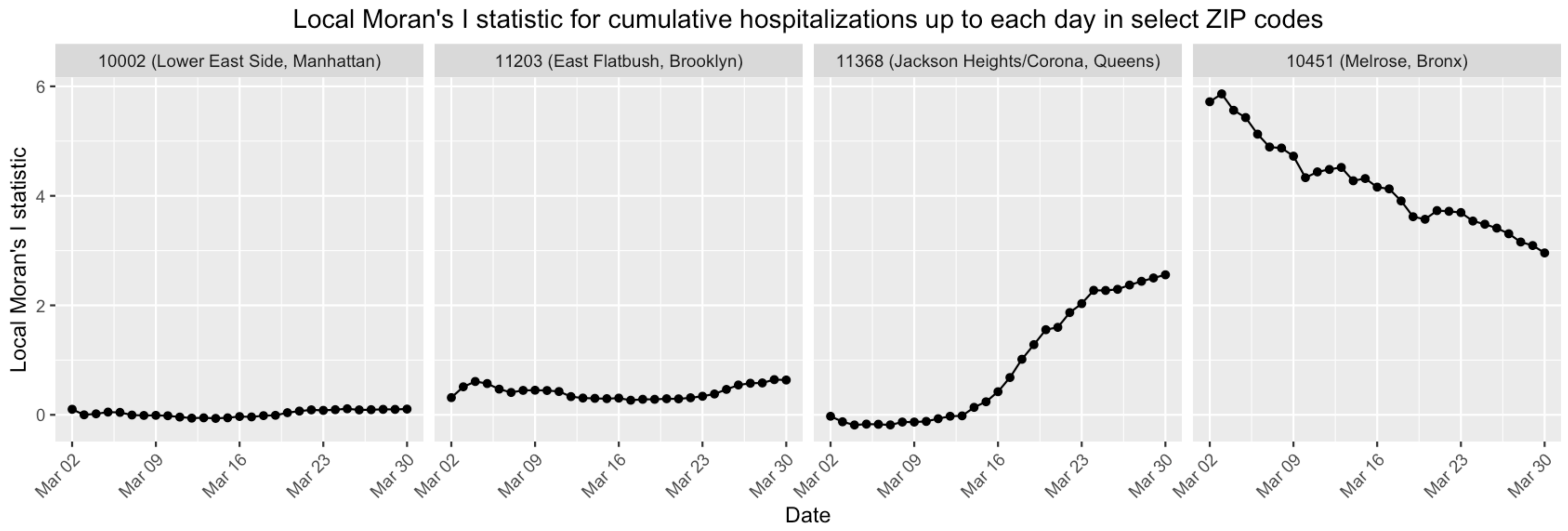
$$\text{Moran's } I_i = \frac{\text{covariance between loc. } i \text{ and its neighbors}}{\text{variance over all locations}}$$

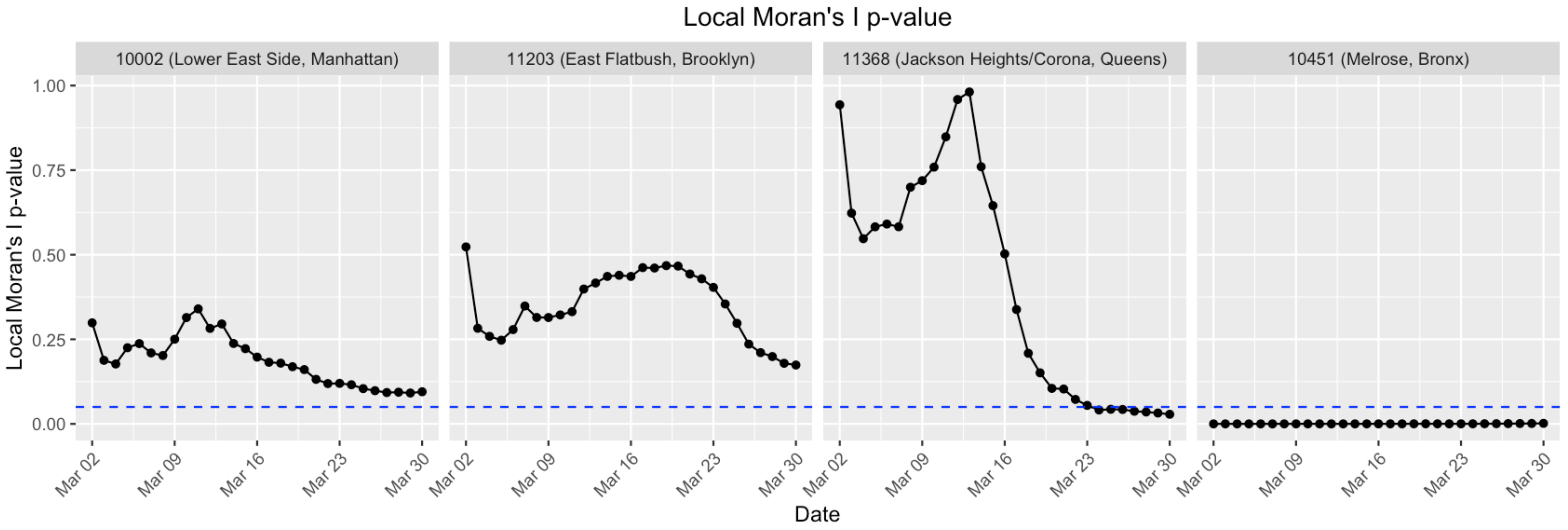
We'll compute this for every ZIP and test significance of each.

Local spatial autocorrelation



Local spatial autocorrelation





Local spatial autocorrelation

Local Moran’s ***I*** is a **LISA** (local indicator of spatial association) and lets us assign each ZIP to categories:

Local spatial autocorrelation

Local Moran's *I* is a **LISA** (local indicator of spatial association) and lets us assign each ZIP to categories:

- **High-high** or **low-low:** as cases rise (fall) in a ZIP, they rise (fall) in neighboring ZIPs

Local spatial autocorrelation

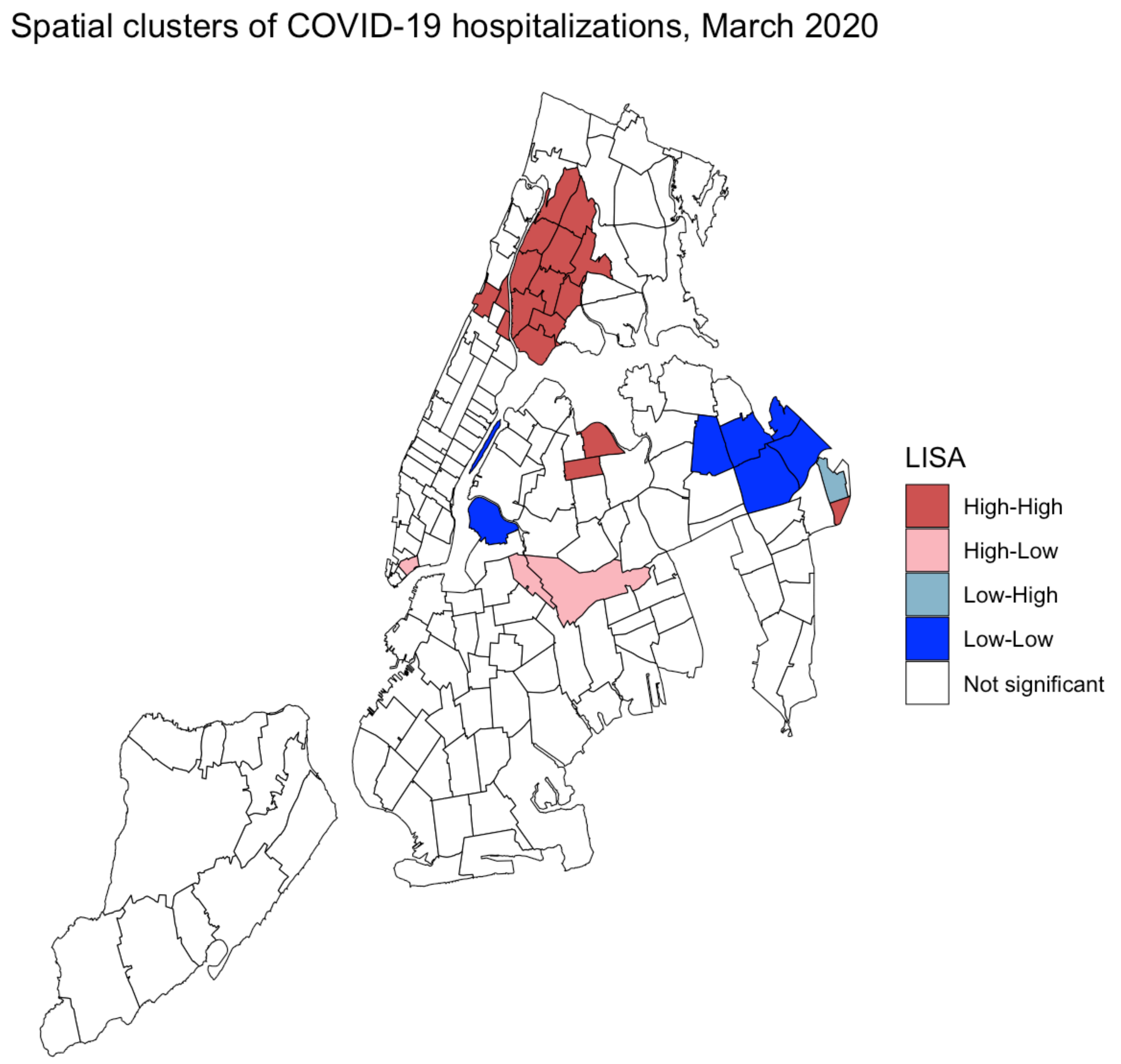
Local Moran's *I* is a **LISA** (local indicator of spatial association) and lets us assign each ZIP to categories:

- **High-high** or **low-low:** as cases rise (fall) in a ZIP, they rise (fall) in neighboring ZIPs
- **Low-high:** low cases in a ZIP associated with neighboring high cases
- **High-low**: high cases in a ZIP associated with neighboring low cases

This is based on both the sign (+ or –) of each ZIP's local Moran's I and its statistical significance.

Local spatial autocorrelation

```
# Extract LISA categories from localmoran() func.
COVID_LISA <-
    attr(localmoran(rowSums(NYC_COVID),
    NYC_neighbors), "quadr")["mean"]

# Mark non-significant local statistics
COVID_LISA[which(cumulative_local_pvalue > 0.05)]
   <- "Not significant"

# Merge map data and LISA categories
plotData <- merge(mapData, COVID_LISA, by.x="id",
   by.y="zip")

# Draw the map
ggplot(plotData, aes(x=long, y=lat, text=id)) +
   geom_polygon(aes(group=group, fill=LISA))
```



Spatial modeling (without time, for now)

Suppose we just have spatial observations $x_1, \ldots, x_N$ (e.g. each ZIP's total COVID cases).

A common model for the expected value of x_i is:

$$\mathbb{E}(x_i) = \alpha + \beta_i$$

- α is a **fixed effect**: the average value (case count) over all locations
- β_i is a **random effect**: an additional effect specific to location i

Spatial modeling (without time, for now)

Suppose we just have spatial observations $x_1, \dots, x_N$ (e.g. each ZIP's total COVID cases).

A common model for the expected value of x_i is:

$$\mathbb{E}(x_i) = \alpha + \beta_i$$

- α is a **fixed effect**: the average value (case count) over all locations
- β_i is a **random effect**: an additional effect specific to location i

Specifically, the "BYM" (Besag-York-Mollié model) says $\beta_i = \beta_{1,i} + \beta_{2,i}$, where

- $\beta_{1,i}$ is a **spatially structured residual** (its value depends on neighboring ZIPs)
- $\beta_{2,i}$ is **unstructured** (just i.i.d. normally-distributed error)

Spatial modeling (without time, for now)

Suppose we just have spatial observations $x_1, \ldots, x_N$ (e.g. each ZIP's total COVID cases).

A common model for the expected value of x_i is:

$$\mathbb{E}(x_i) = \alpha + \beta_i$$

- α is a **fixed effect**: the average value (case count) over all locations
- β_i is a **random effect**: an additional effect specific to location i

Specifically, the "BYM" (Besag-York-Mollié model) says $\beta_i = \beta_{1,i} + \beta_{2,i}$, where

- $\beta_{1,i}$ is a **spatially structured residual** (its value depends on neighboring ZIPs)
- $\beta_{2,i}$ is **unstructured** (just i.i.d. normally-distributed error)

BYM says $\beta_{1,i}$ is conditionally autoregressive (CAR):

it is normally-distributed given the values of the neighboring $\beta_{1,j}$ terms,

with mean and variance equal to the mean and variance of those terms.

INLA software for fitting spatial models

INLA = “Integrated nested Laplace approximation”

How do we account for spatial (and/or temporal) correlation in a model?

- If there is spatial correlation, the precision matrix of the parameters should have a non-zero block pattern corresponding to neighborhoods.
  (I.e. we can assume a Gaussian Markov random field prior on the parameters)

INLA software for fitting spatial models

INLA = “Integrated nested Laplace approximation”

How do we account for spatial (and/or temporal) correlation in a model?

- If there is spatial correlation, the precision matrix of the parameters should have a non-zero block pattern corresponding to neighborhoods.
  (I.e. we can assume a Gaussian Markov random field prior on the parameters)

- INLA uses this assumption to produce a Laplace approximation to the posterior distributions of the parameters

- This approximation yields more computationally-efficient parameter estimation than other Bayesian approaches that rely on integration.

INLA software for fitting spatial models

INLA = “Integrated nested Laplace approximation”

Using the INLA R package, we can fit models using commands similar to those for other models in R:

```
                  Outcome Intercept     Covariates
>    formula <- y    ~ 1    +   x1 + x2 +
                     f(z1, model = “[how is z1 structured?]” …)
```

INLA software for fitting spatial models

INLA = “Integrated nested Laplace approximation”

Using the INLA R package, we can fit models using commands similar to those for other models in R:

```
                 Outcome  Intercept      Covariates
>   formula <- y    ~ 1    +   x1 + x2 +
                 f(z1, model = “[how is z1 structured?]” …)
                      spatially- or temporally- structured term
              (e.g, BYM structure for spatial term, or random walk for time term)
```

INLA software for fitting spatial models

INLA = "Integrated nested Laplace approximation"

Using the INLA R package, we can fit models using commands similar to those for other models in R:

```
                  Outcome  Intercept      Covariates
>   formula <- y    ~ 1    +   x1 + x2 +
                  f(z1, model = "[how is z1 structured?]" …)
                      spatially- or temporally- structured term
                (e.g, BYM structure for spatial term, or random walk for time term)


>   model <- inla(formula,
                      family = "[how is outcome distributed?]", data = …)
```

INLA software for fitting spatial models

INLA = “Integrated nested Laplace approximation”

Using the INLA R package, we can fit models using commands similar to those for other models in R:

```
                    Outcome  Intercept      Covariates
>   formula <- y    ~ 1    +   x1 + x2 +
                       f(z1, model = “[how is z1 structured?]” …)
                          spatially- or temporally- structured term
                   (e.g, BYM structure for spatial term, or random walk for time term)
```

```
>   model <- inla(formula,
                         family = “[how is outcome distributed?]”, data = …)
                         Distribution could be Poisson, Gaussian, etc.
```

Spatial modeling (without time, for now)

When our data $x_1, \ldots, x_N$ are **counts**, as in COVID cases in *N* ZIP codes, we can use **Poisson regression**, which specifies:

$$\log(\lambda_i) = \alpha + \beta_i$$

Where λ_i is the **incidence rate** in ZIP i.

Spatial modeling (without time, for now)

When our data $x_1, \ldots, x_N$ are **counts**, as in COVID cases in *N* ZIP codes, we can use **Poisson regression**, which specifies:

$$\log(\lambda_i) = \alpha + \beta_i$$

Where λ_i is the **incidence rate** in ZIP i.

Recall that α is the avg. value over all locations, β_i is a location-specific effect.

We can fit this model, i.e. estimate the parameters α and β_i for each i, via the `INLA` R package.

Spatial modeling (without time, for now)

```
# Gather the data needed for the model
modelData <- data.frame(zipID=1:length(zips), count=rowSums(NYC_COVID), population=Population)

# Create list of each ZIP's neighbors
nb2INLA("NYC.graph", poly2nb(NYC_SHP))

# Define the formula for this model
model0Formula <- count ~ 1 + f(zipID, model="bym", graph=paste("NYC.graph"))

# Fit the model using the INLA algorithm
spatialModel <- inla(model0Formula, family="poisson", offset=log(population), data=modelData)
```

Spatial modeling (without time, for now)

```
# Gather the data needed for the model
modelData <- data.frame(zipID=1:length(zips), count=rowSums(NYC_COVID), population=Population)

# Create list of each ZIP’s neighbors
nb2INLA("NYC.graph", poly2nb(NYC_SHP))

# Define the formula for this model
model0Formula <- count ~ 1 + f(zipID, model="bym", graph=paste("NYC.graph"))

# Fit the model using the INLA algorithm
spatialModel <- inla(model0Formula, family="poisson", offset=log(population), data=modelData)
```

Since $\log(\lambda_i) = \alpha + \beta_i$, we need to look at $\exp(\alpha)$ to get the estimated city-wide hospitalization rate (~1.2%)

```
exp(spatialModel$summary.fixed)
```

```
                  mean          sd   0.025quant   0.5quant   0.975quant
(Intercept) 0.01217149    1.030081    0.0114818 0.01217186   0.01290044
```

Spatial modeling (without time, for now)

Next we obtain the ZIP-specific additional effects, $\exp(\beta_i)$, and map them:

```
# Initialize dataframe to store beta_i terms
zipResiduals <- data.frame(zip=zips, resid=0)

# Compute the beta_i's
for(i in 1:length(zips))
  zipResiduals[i, "resid"] <- sum(spatialModel$summary.random$zipID$mean[i])

# Exponentiate beta_i terms
zipResiduals$resid <- exp(zipResiduals$resid)

# Plot
plotData <- merge(plotData, zipResiduals, by.x="id",
     by.y="zip")
ggplot(plotData, aes(x=long, y=lat)) +
     geom_polygon(aes(group=group, fill=resid)) +
     scale_fill_distiller(palette="Purples")
```

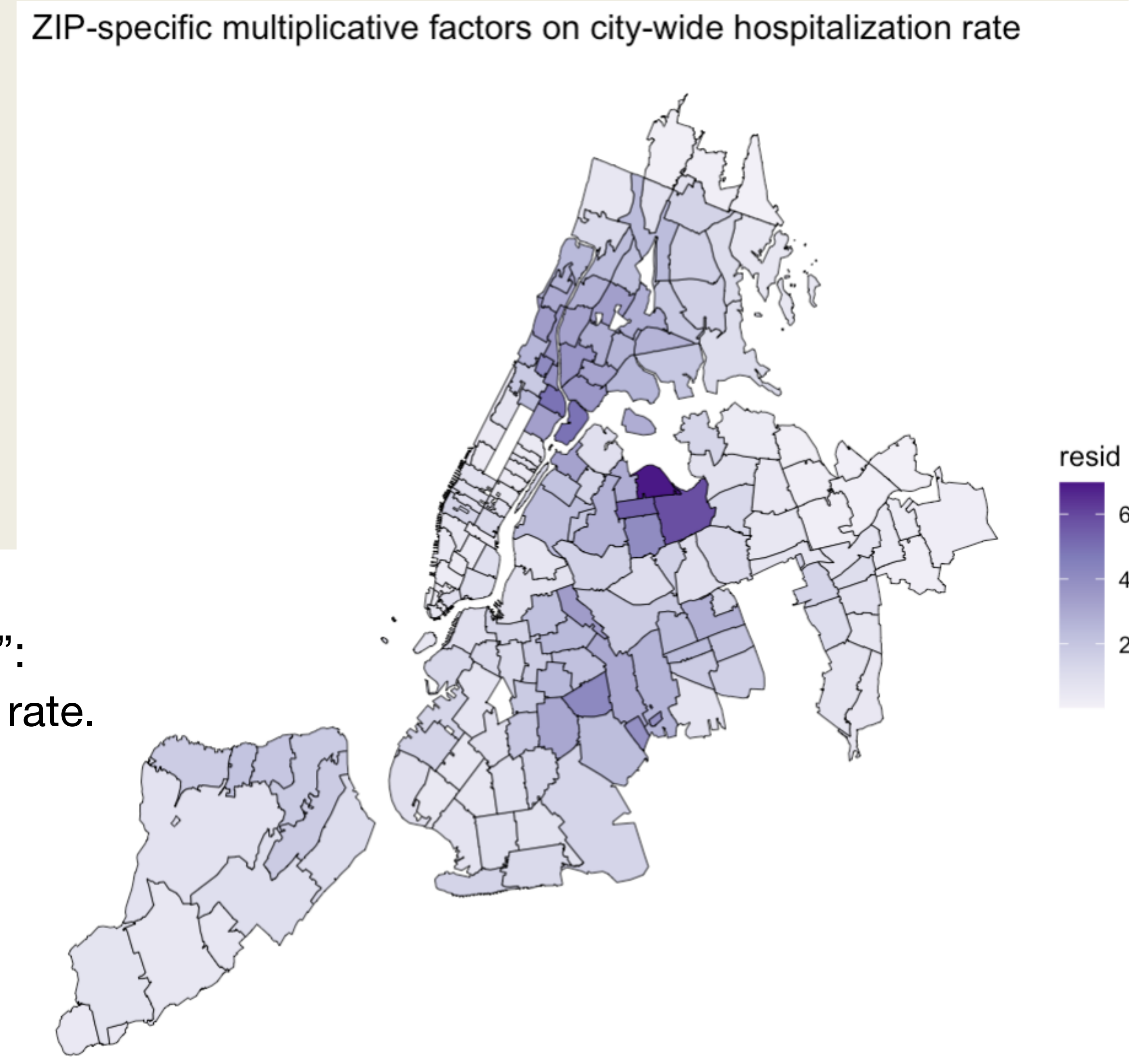
In the $\log(\lambda_i) = \alpha + \beta_i$ formulation, β_i looks like a “residual”:
additional ZIP-level variation on top of log-mean (city-wide) rate.

In the $\lambda_i = \exp(\alpha)\exp(\beta_i)$ formulation, $\exp(\beta_i)$ looks like a
multiplicative factor on the mean (city-wide) rate.

Spatial modeling (without time, for now)

Next we obtain the ZIP-specific additional effects, $\exp(\beta_i)$, and map them:

```
# Initialize dataframe to store beta_i terms
zipResiduals <- data.frame(zip=zips, resid=0)

# Compute the beta_i's
for(i in 1:length(zips))
  zipResiduals[i, "resid"] <- sum(spatialModel$summary.random$zipID$mean[i])

# Exponentiate beta_i terms
zipResiduals$resid <- exp(zipResiduals$resid)

# Plot
plotData <- merge(plotData, zipResiduals, by.x="id",
     by.y="zip")
ggplot(plotData, aes(x=long, y=lat)) +
     geom_polygon(aes(group=group, fill=resid)) +
     scale_fill_distiller(palette="Purples")
```


ZIP-specific multiplicative factors on city-wide hospitalization rate
resid
6
4
2

In the $\log(\lambda_i) = \alpha + \beta_i$ formulation, β_i looks like a "residual": additional ZIP-level variation on top of log-mean (city-wide) rate.

In the $\lambda_i = \exp(\alpha)\exp(\beta_i)$ formulation, $\exp(\beta_i)$ looks like a multiplicative factor on the mean (city-wide) rate.

Spatiotemporal modeling

Now observations are x_{it} at **spatial locations** $i = 1,..., N$ and time points $t = 1,..., T$.
(e.g. COVID cases in each ZIP each day in March 2020).

Our (Poisson) model now becomes:

$$\log(\lambda_{it}) = \alpha + \beta_i + \gamma_t$$

- α is a **fixed effect**: the average value (case count) over all locations and all time

- β_i is a **spatial random effect**: effect specific to location i.

- γ_t is a **temporal random effect**

Spatiotemporal modeling

Now observations are x_{it} at **spatial locations** $i = 1,..., N$ and time points $t = 1,..., T$.
(e.g. COVID cases in each ZIP each day in March 2020).

Our (Poisson) model now becomes:

$$\log(\lambda_{it}) = \alpha + \beta_i + \gamma_t$$

- α is a **fixed effect**: the average value (case count) over all locations and all time
- β_i is a **spatial random effect**: effect specific to location i.
  As before, $\beta_i = \beta_{1,i} + \beta_{2,i}$, where $\beta_{1,i}$ is spatially-structured (dependent on neighboring locations)and $\beta_{2,i}$ is additional error.
- γ_t is a **temporal random effect:** like β_i, we have $\gamma_t = \gamma_{1,t} + \gamma_{2,t}$, where $\gamma_{1,t}$ is temporally-structured (its value depends on previous time points) and $\gamma_{2,t}$ is additional error.

Spatiotemporal modeling

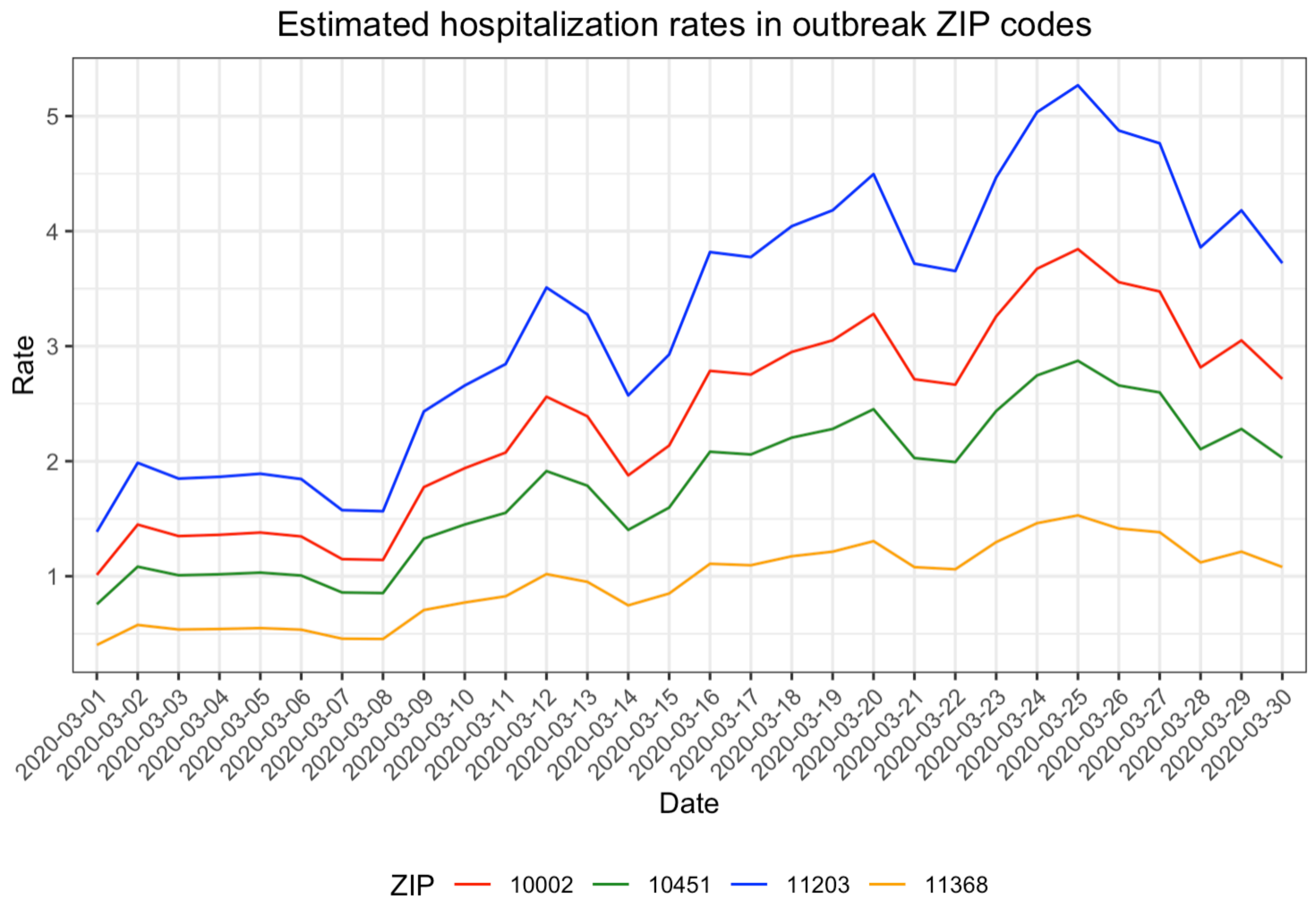
```
# Gather data needed for modeling
modelData <- reshape2::melt(t(cbind(NYC_COVID)))
modelData$zipID <- sort(rep(1:length(zips), length(dates)))
modelData$time <- rep(1:length(dates), length(zips))
modelData$population <- rep(NYC_Demographics$Population, times=rep(length(dates), length(zips))

# Define the model formula with spatial and temporal components
model1Formula <- value ~ 1 + f(zipID, model="bym", graph=paste("NYC.graph"))
   + f(time, model="rw1") + f(time2, model="iid")

# Fit the model
spatiotemporalModel <- inla(model1Formula, family="poisson", offset=log(population), data=modelData)

# Examine fixed effects
exp(spatiotemporalModel$summary.fixed)
```

As before, we can look at $\exp(\beta_i)$ terms to see ZIP-specific effects, and $\exp(\gamma_t)$ terms to see the multiplicative effect of each day passing.

Spatiotemporal modeling

We can look at the fitted daily infection rate in a few specific ZIP codes.

- Recall our model is: $\log(\lambda_{it}) = \alpha + \beta_i + \gamma_t$
- The fitted values of γ_t give us an estimated mean time series for the infections over time across the whole city.
- We add the location-specific term β_i to this time series to get the trajectory for a specific ZIP.



These are the fitted values in four ZIP codes that had more severe outbreaks.

Spatiotemporal modeling

An alternative formulation:

- We might want to allow our model to have an interaction between space and time, so that we can explain different time trends in different areas:

  $$\log(\lambda_{it}) = \alpha + \beta_i + \gamma_t + \delta_{it}$$

  (Recall that both β_i and γ_t are comprised of the sum of a spatially- or temporally-structured term and an i.i.d. term)

Spatiotemporal modeling

An alternative formulation:

- We might want to allow our model to have an interaction between space and time, so that we can explain different time trends in different areas:

  $$\log(\lambda_{it}) = \alpha + \beta_i + \gamma_t + \delta_{it}$$

  (Recall that both β_i and γ_t are comprised of the sum of a spatially- or temporally-structured term and an i.i.d. term)

- In code, this would be:

```
formula <- y ~ 1 +
               f(spaceID, model = “bym”, graph = A) +
               f(timeID, model = “rw”) +
               f(timeID2, model = “iid”) +
               f(spaceTimeID, model = “iid)
```

Spatiotemporal modeling

An alternative formulation:

- We might want to allow our model to have an interaction between space and time, so that we can explain different time trends in different areas:

  $$\log(\lambda_{it}) = \alpha + \beta_i + \gamma_t + \delta_{it}$$

  (Recall that both β_i and γ_t are comprised of the sum of a spatially- or temporally-structured term and an i.i.d. term)

- In code, this would be:

```
formula <- y ~ 1 +
               f(spaceID, model = “bym”, graph = A) +
               f(timeID, model = “rw”) +
               f(timeID2, model = “iid”) +
               f(spaceTimeID, model = “iid”)
```

Intercept (overall rate)

Spatial term β_i, defined using adjacency matrix *A* specifying each ZIP’s neighbors

Temporal term $\gamma_{1,t}$ defined as random walk

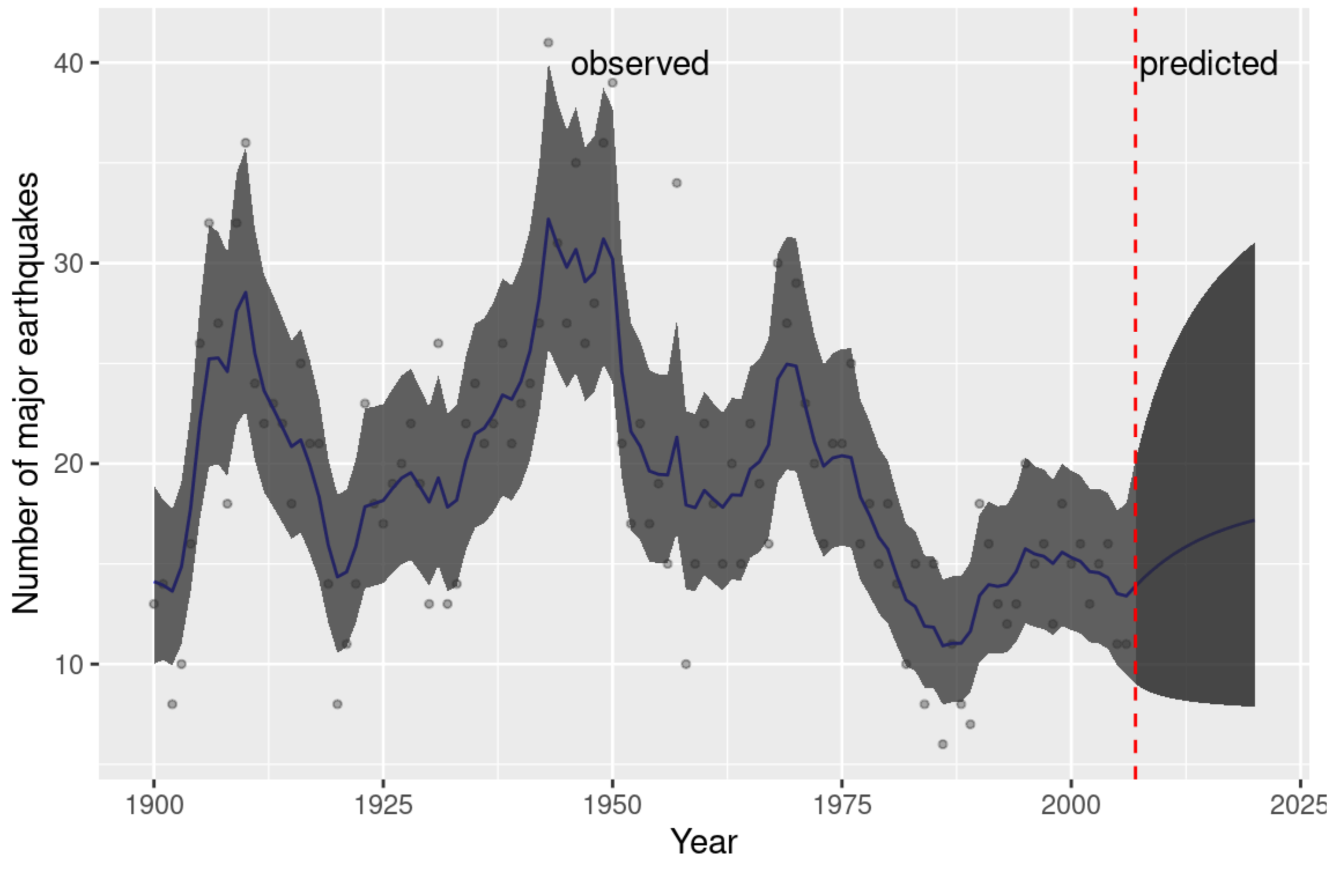
Temporal term $\gamma_{2,t}$ defined as i.i.d. noise

Space-time interaction term δ_{it}

Forecasting

- To forecast future observations in time in the Bayesian framework,
    - We can add missing observations to our dataset at “future” time points
    - Fit the same INLA model to the augmented dataset and look at the predictive distribution of those missing points.
    - We add the following piece to the `inla()` function call:

```
control.predictor = list(compute = TRUE, link = 1)
```



Example forecasting on a dataset about the number of major earthquakes each year

Resources

Spatiotemporal Statistics with R (full textbook accessible online)

C. Wikle, A. Zammit-Mangion, N. Cressie.

“Spatial and spatio-temporal models with R-INLA”

M. Blangiardo, M. Cameletti, G. Baio, H. Rue.

GeoDa documentation articles:

https://geodacenter.github.io/documentation.html

L. Anselin.

Feel free to email me: **skv24@cornell.edu**

Thanks!