REGION
The Journal of ERSA
Powered by WU

Volume 1, Number 1, 2020, 999–999
DOI: 10.18335/region.v??i??.???

journal homepage: region.ersa.org
ISSN: 2409-5370

A practical guide to spatial interaction modelling

Blinded[1]

1

Received: /Accepted:

Abstract. This document is only a demo explaining how to use the template.

1 Introduction

Spatial interaction models (SIMs) is a core tool to simulate flows between different locations in physical space. They are a valuable resource through which the geographic structure between locations encoded in aggregate flows of people, information and goods can be represented and understood. Intuitively, SIMs seek to capture the spatial interaction between places as a function of three components: origin attributes, destination attributes and their separation. Inspired by Newtonian concepts developed in physics, spatial flows between locations are conceived as the result of their proportional gravitational force and inverse association with spatial separation. Attributes of origin and destination locations are employed to represent gravitational forces pushing and pulling people, information and goods between specific locations. Various forms of distance and costs are used to represent the deterring effects of geographical separation on spatial flows.

SIMs are widely used for prediction and inference. They are used to make inference about the factors contributing to influence spatial flows. They have been used to understand the magnitude and direction of influence of individual and place-level attributes on geographic flows. Understanding the effect of these factors offers valuable evidence to inform the development of appropriate plans, strategies and interventions (Fotheringham, O'Kelly 1989). SIMs are also used to make predictions of the size of spatial flows. These predictions are normally used to assess the impact of interventions and creation of "what-if" scenarios, providing guidance for the identification of optimal locations and size for potential new service units [REF]. In this context, SIMs are often used to evaluate the impact of new bus stops, shopping stores, schools or housing units on their potential demand and traffic changes (Fotheringham, O'Kelly 1989). To these ends, SIMs have been used to address questions in a variety of settings, including retail, migration, transport, trade, commuting, school travel and more broadly urban planning.

Yet, the implementation of SIMs remains a challenge. Algorithms to calibrate the parameters of SIMs have remained locked away, either behind dense algebraic notation in dusty papers from the 1970s, or behind paywalls of commercial software (Rowe et al. 2024). Additionally, Rowe et al. (2024) noted a dearth of knowledge within geographical education as SIMs are not widely taught in undergraduate programmes in the same way as, for instance, regression models are taught in economics or social psychology. This situation is argued to have occurred despite the availability of effective routines to calibrate SIMs via popular linear and general linear modelling frameworks, and as practical expediency is sacrificed at the expense of theoretical or technical prowess (Rowe et al. 2024). The

ways in which calibration procedures are presented as lengthy mathematical derivation or passing reference to ordinary least square tend to hamper accessibility for the easy implementation of SIMs.

This computational notebook contributes to redressing these issues. It aims to provide an intuitive, understandable and practical guide to estimate SIMs in a variety of modelling frameworks. It will include the necessary code to calibrate SIMs, using origin-destination travel-to-work data for the United Kingdom in R programming language. The code provided is generisable and can be adapted to different origin-destination flow data and contexts, including migration, student, transport, trade, currency, data transfer, vessel, shipment and freight flows.

The notebook is structured as follows. The next section sets out some fundamental concepts and definitions relating to SIMs. Section 3 identifies the libraries used before Section 4 describes the data. Section 5 illustrates key techniques to visualise complex spatial interaction data, and Section 6 shows and explains how to estimate SIMs using a range of modelling frameworks. It start with traditional mathematical and Ordinary Least Squares (OLS) approaches to more advanced statistical frameworks, such as Generalised Linear Mixed Models (GLMMs) and machine learning algorithms.

2 Context

SIMs take various forms. Newtonian gravity models are probably the most widely known and used form of SIMs. Inspired by Newton's law of gravity, the basic gravity version of these models assumes that the spatial flows or interactions between an origin (i) and a destination j is proportional to their masses (M_i and M_j) and inversely proportional to their separation (D_{ij}). Locations are expected to interact in a positively reinforcing manner that is multiplicative of their masses, but to diminish with the intervening role of their separation. The parameters μ and γ reflect the proportional relationship between the masses and flows. The separation between locations is often represented by a distance decay function and is measured in terms of the distance, cost or time involved in the interaction. Generally, the model includes a constant (κ) ensuring that the expected flows do not exceed their respective observed counts, and a parameter (β) representing the deterring effect of geographical separation. The task is to estimate the parameters κ, μ, γ and β. Following Wilson (1971), a gravity model can be expressed as:

$$T_{ij} = \kappa \frac{M_i^{\mu} M_j^{\gamma}}{D_{ij}^{\beta}} \tag{1}$$

SIMs have three key inputs: (1) a matrix of flows between a set of origins and destinations; (2) a measure of separation between origins and destinations; and, (3) measures of masses at origin and destination locations. The literature usually considers a family of SIMs taking four forms which refers to various constraints placed on parameters of the model (Wilson 1971). There is an *unconstrained* version which offers a measure of spatial separation, assuming that there is no information on the number of flows originating from each origin to each destination. The number of flows is thus estimated via SIMs using surrogate factors, such as population at the origin and destiination. Constrained versions are used to ensure that specific origin or destination observations are met. Three general formulations of constrained models are used: *production-constrained*, *attraction-constrained* and *doubly-constrained* models. *Production-constrained* versions are used to constrain a model to origin factors so that the predicted flows emanating from individual origins are in proportion to the relative attractiveness of each destination. *Attraction-constrained* versions do the same but constrain models to destination factors so that the predicted flows terminating at each destination are in proportion to the relative attactiveness at individual origins. *Doubly-constrained* versions combine these two sets of constraints to ensure predicted flows are equal to observed flows are constrained by both origin and destination factors.

Various modelling frameworks have been used to calibrate the parameters of SIMs. Originally, mathematical formulations were heavily used but these did not offer any ideas of uncertainty about the estimated parameters and were seen as deterministic. To

mitigate this, statistical frameworks proliferated. A simple and widely used formulation is a linear model in which geographic flows are logged to meet the normality modelling assumptions, and OLS and maximum likelihood optimisation frameworks are used to estimate the model parameters. However, linear modelling has various constraints and cannot easily incorporate flow counts of zero, right-skewed distributions of flows and non-linear relationships between flows and the set of predictors. As result, more advanced modelling frameworks have been used to fit SIMs, including count data approaches such as Poison and Negative Binomial Models (Rowe et al. 2023), Generalised Linear Mixed Models (Aparicio Castro et al. 2023) and more recently machine learning (Rowe et al. 2022) and deep learning models (Simini et al. 2021). This computational notebook will provide a practical guide on how to implement these models using travel-to-work data for the UK. The next sections will first introduce the computational environment and data used for this purpose.

3 Computation environment

```
# data management
library(tidyverse)
# spatial data management
library(sf)
# generalised mixed linear modelling
library(glmmTMB)
# data visualisation
library(scales)
library(patchwork)
source("../../code/style/data-visualisation_theme.R")
```

4 Data

We use an origin-destination matrix capturing travel-to-work commuting flows from the 2021 Census for England and Wales. The data provide estimates on usual residents aged 16 years and over and in employment before the Census week at the Lower Tier Local Authority (LTLA) level. The estimates capture the movement of people between their LTLA area of residence and work. The data are available in a long origin-destination pair format. The first seven columns contain the core components of a spatial interaction dataset, including code and names for origin and destination locations, and the population count moving between a origin-destination pair. The remainder of the dataset comprises the origin and destination attributes, including the total population count and population counts by socioeconomic categories. Columns 9 to 19 contain the attributes at origins and columns 20 to 31 contain the attributes at destinations. Looking at the data frame vertically, the first row shown below displays the count of people who reported Hartlepool as their place of residence and work. The third row shows the count of people who were residing in Hartlepool but reported to travel to work in Middlesbrough.

```
# read data
df <- read_csv("../../data/output/sim_uk-travel-to-work_2021.csv") %>%
  # rename variables
  rename(
    ltla_res_code = "Lower tier local authorities code",
    ltla_res_name = "Lower tier local authorities label",
    ltla_work_code = "LTLA of workplace code",
    ltla_work_name = "LTLA of workplace label",
    ltla_work_id_code = "Place of work indicator (4 categories) code",
    ltla_work_id_lbl = "Place of work indicator (4 categories) label",
    count = "Count"
  ) %>%
  # exclude the following observations:
```

```
  dplyr::filter(!ltla_work_name %in%
                  c("Does not apply",
                    "Workplace is offshore installation",
                    "Workplace is outside the UK") ) %>%
  # exclude the following variables:
  select( -c(ltla_work_id_code, ltla_work_id_lbl) ) %>%
  # exclude stayers
  dplyr::filter(ltla_res_code != ltla_work_code)
head(df, n = 5 )
```

```
# A tibble: 5 x 31
  ltla_res_code ltla_res_name ltla_work_code ltla_work_name   count population_o
  <chr>         <chr>         <chr>          <chr>            <dbl>        <dbl>
1 E06000001     Hartlepool    E06000002      Middlesbrough     1500        92338
2 E06000001     Hartlepool    E06000003      Redcar and Clev~   671        92338
3 E06000001     Hartlepool    E06000004      Stockton-on-Tees  4240        92338
4 E06000001     Hartlepool    E06000005      Darlington         398        92338
5 E06000001     Hartlepool    E06000007      Warrington           2        92338
# i 25 more variables: higher_managerial_administrative_professional_o <dbl>,
#   lower_managerial_administrative_professional_o <dbl>, intermediate_o <dbl>,
#   small_employers_own_account_o <dbl>, lower_supervisory_technical_o <dbl>,
#   semi_routine_o <dbl>, routine_o <dbl>, never_worked_unemployed_o <dbl>,
#   ft_students_o <dbl>, no_qualifications_o <dbl>, level4_o <dbl>,
#   population_d <dbl>, higher_managerial_administrative_professional_d <dbl>,
#   lower_managerial_administrative_professional_d <dbl>, ...
```

5 Visualising spatial interaction data

6 Estimating spatial interaction models

This section focuses on how to calibrate spatial interaction models in the form of gravity models. It starts with the classical mathematical formulation, then moves to basic statistical frameworks and their extensions.

6.1 Mathematical gravity models

6.1.1 Unconstrained model

We start by implementing the mathematical formulation of gravity models using Equation 1. We adopt the standard Newtonian formulation assuming that μ and γ equal to 1 and β to -2, and fit an *unconstrained* model. Recall here that this model assumes that no information on flows is available. We use the population at origin and destination LDAs and distance between LDA centroids in kilometers to define this model. To assess the model's predictive capacity, we graphically compare predicted flows *versus* observed flows. We focus on movements between LDAs, removing movements within LDAs with distance zero. A perfect model would product predicted flows displaying a perfect positive linear relationship with the orbserved flows. The figure below shows the unconstrained model results displaying little correspondence between the predicted and observed flows. The model predicts unrealistic flows of over 10 million which exceeds the size of the observed flows. This reflects the unconstrained nature of the model producing predictions based on distance, origin and destination only.

6.1.2 Constrained models

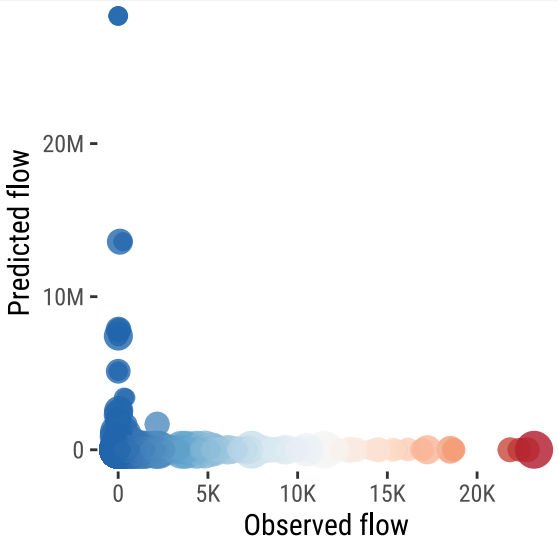
Production-constrained model

To address these inconsistencies, we turn to constrained models and first focus on the production-constrained model. This model assumes that we observe the total flows from each origin but not those arriving in each destination. So we can use information on the total number of outflows to constrain the model and distribute the total outflows

```
# Unconstrained model
# assuming these parameters
mu <- 1
gamma <- 1
beta <- 2
kappa <- 1

# using equation 1 to estimate unconstrained model
df$predicted_flow <- round( kappa *
  (df$population_o^mu * df$population_d^gamma) / (df$distance_km^beta)
  )

# predicted versus observed flows
df %>%
  # remove predictions
  filter( distance_km != 0) %>%
  ggplot( aes( x = count, y = predicted_flow)) +
  geom_point( alpha = .8, aes(colour = count,
                              size = abs(population_o / 1e1)) ) +
  # change axis label to a shorthand
  scale_y_continuous(
    labels = label_number(scale_cut = cut_short_scale())
    ) +
  scale_x_continuous(
    labels = label_number(scale_cut = cut_short_scale())
    ) +
  # change colour scale
  scale_colour_distiller(palette = "RdBu", direction = -1) +
  labs(y = "Predicted flow",
       x = "Observed flow") +
  theme_plot_tufte() +
  theme(legend.position = "none",
        axis.text.y = element_text(size = 9),
        axis.text.x = element_text(size = 9),
        axis.title=element_text(size = 11)
        )
```



Note: Colour represents the size of observed counts. Point size represents the size of population at origins.

Figure 1: Unconstrained model predicted versus observed flows

proportionally to the population of destinations, and as before, inversely proportional to distance.

```
# Production-constrained model
# total outflow from each origin
outflow <- df %>%
  group_by(ltla_res_code) %>%
  summarize(total_outflow = sum(count)) %>%
  ungroup()

# merge outflows with original data
df <- df %>%
  left_join(outflow, by = "ltla_res_code")

# compute denominator for each origin
df <- df %>%
  group_by(ltla_res_code) %>%
  mutate(
    kappa_production =
      sum((population_d^gamma) / (distance_km^beta),
                            na.rm = TRUE)) %>%
  ungroup()

# compute predicted flows
df <- df %>% mutate(
  production_constrained_flow =
    total_outflow * ((population_d^gamma) / (distance_km^beta)) /
    kappa_production
  )
```

Attraction-constrained model

Alternatively, we could assume we only have data on the total number of flows arriving at each destination but we have no information on the number of flows being generated from each origin. In such situations, we can use an attraction-constrained model as it uses the known information on total inflows for individual destinations and a set of surrogate variables to represent the total number of flows from each origin.

```
# Attraction-constrained model
# total inflow from each origin
inflow <- df %>%
  group_by(ltla_work_code ) %>%
  summarize(total_inflow = sum(count)) %>%
  ungroup()

# merge inflows with original data
df <- df %>%
  left_join(inflow, by = "ltla_work_code")

# compute denominator for each destination
df <- df %>%
  group_by(ltla_work_code) %>%
  mutate(
    kappa_attraction =
      sum((population_o^mu) / (distance_km^beta))) %>%
  ungroup()

# compute predicted flows
df <- df %>% mutate(
  attraction_constrained_flow =
```

```
    total_inflow * ((population_o^mu) / (distance_km^beta)) /
    kappa_attraction
  )
```

Doubly-constrained model

We can also adjust the number of flows so that both total outflows from each origin and total inflows to each destination match observed totals. This model requires an iterative solution to meet these constraints. To this end, we first set the parameters for the algorithm to converge to a solution i.e. tolerance, maximum iterations (max_iter) and iteration steps (iter). Intuitively the algorithm iteratively adjust the flows by first applying the production constrain, and subsequently the attraction constrain in a similar manner as used above. It then assesses the difference between total predicted and observed flows. It stops when this difference is smaller than the established tolerance parameter. The algorithm uses the unconstrained model predicted flow as a starting point for the iterative process.

```
# Doubly-constrained model
# set parameters for iterations
tolerance <- 1e-6
max_iter <- 1000
iter <- 1
converged <- FALSE

# iteratively adjust flows to satisfy both constraints
while (!converged && iter <= max_iter) {
  # Step 1: apply production constraint for each origin
  df <- df %>%
    group_by(ltla_res_code ) %>%
    mutate(
      predicted_flow =
        predicted_flow * (total_outflow / sum(predicted_flow))) %>%
    ungroup()

  # Step 2: apply attraction constraint for each destination
  df <- df %>%
    group_by(ltla_work_code) %>%
    mutate(
      predicted_flow =
        predicted_flow * (total_inflow / sum(predicted_flow))) %>%
    ungroup()

  # assess convergence comparing row sums of predicted and observed flows
  origin_check <- df %>%
    group_by(ltla_res_code) %>%
    summarize(
      total_predicted =
        sum(predicted_flow), total_observed = unique(total_outflow)
      ) %>%
    mutate(
      difference =
        abs(total_predicted - total_observed))

  destination_check <- df %>%
    group_by(ltla_work_code) %>%
    summarize(
      total_predicted =
        sum(predicted_flow), total_observed = unique(total_inflow)
      ) %>%
```

```
    mutate(
      difference = abs(total_predicted - total_observed)
      )

  max_difference <- max(c(origin_check$difference,
                          destination_check$difference))

  # check if the maximum difference is below the tolerance level
  if (max_difference < tolerance) {
    converged <- TRUE
  } else {
    iter <- iter + 1
  }
}
```

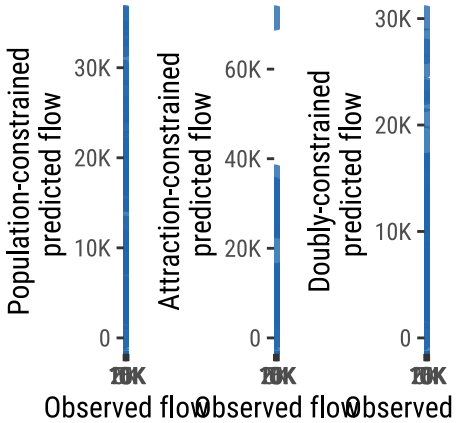
We can visualise the predictions from the three constrained models. While the predicted flows are closer to the observed flows compared to the unconstrained model outputs, large discrepancies still exist, with differences of over 10 thousand. So far, we have assumed that the parameters μ and γ that moderate the relationship between commuting flows and population counts at origins and destinations is 1, and that the β distance decay parameter is 2. Yet, these are arbitrary numbers and need to be empirically estimated. We will do this by using statistical models.

```
# predicted versus observed flows
production_constrained_plot <- df %>%
  # remove predictions
  filter( distance_km != 0) %>%
  ggplot( aes( x = count, y = production_constrained_flow)) +
  geom_point( alpha = .8, aes(colour = count,
                              size = abs(population_o / 1e1)) ) +
  # change axis label to a shorthand
  scale_y_continuous(
    labels = label_number(scale_cut = cut_short_scale())
    ) +
  scale_x_continuous(
    labels = label_number(scale_cut = cut_short_scale())
    ) +
  # change colour scale
  scale_colour_distiller(palette = "RdBu", direction = -1) +
  labs(y = "Population-constrained \n predicted flow",
       x = "Observed flow") +
  theme_plot_tufte() +
  theme(legend.position = "none",
        axis.text.y = element_text(size = 9),
        axis.text.x = element_text(size = 9),
        axis.title=element_text(size=11)
        )

attraction_constrained_plot <- df %>%
  # remove predictions
  filter( distance_km != 0) %>%
  ggplot( aes( x = count, y = attraction_constrained_flow)) +
  geom_point( alpha = .8, aes(colour = count,
                              size = abs(population_o / 1e1)) ) +
  # change axis label to a shorthand
  scale_y_continuous(
    labels = label_number(scale_cut = cut_short_scale())
    ) +
```

```
  scale_x_continuous(
    labels = label_number(scale_cut = cut_short_scale())
    ) +
  # change colour scale
  scale_colour_distiller(palette = "RdBu", direction = -1) +
  labs(y = "Attraction-constrained \n predicted flow",
       x = "Observed flow") +
  theme_plot_tufte() +
  theme(legend.position = "none",
        axis.text.y = element_text(size = 9),
        axis.text.x = element_text(size = 9),
        axis.title=element_text(size=11)
        )

doubly_constrained_plot <- df %>%
  # remove predictions
  filter( distance_km != 0) %>%
  ggplot( aes( x = count, y = predicted_flow)) +
  geom_point( alpha = .8, aes(colour = count,
                              size = abs(population_o / 1e1)) ) +
  # change axis label to a shorthand
  scale_y_continuous(
    labels = label_number(scale_cut = cut_short_scale())
    ) +
  scale_x_continuous(
    labels = label_number(scale_cut = cut_short_scale())
    ) +
  # change colour scale
  scale_colour_distiller(palette = "RdBu", direction = -1) +
  labs(y = "Doubly-constrained \n predicted flow",
       x = "Observed flow") +
  theme_plot_tufte() +
  theme(legend.position = "none",
        axis.text.y = element_text(size = 9),
        axis.text.x = element_text(size = 9),
        axis.title=element_text(size=11)
        )
```



6.2 Statistical gravity models

6.2.1 Linear regression model

```
Call:
```

```
lm(formula = count ~ population_o + population_d + distance_km,
    data = df)

Residuals:
    Min      1Q  Median      3Q     Max
 -384.0  -105.2   -77.1   -55.9 22872.3

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.933e+00  7.378e+00   0.262    0.793
population_o 2.167e-04  1.720e-05  12.592   <2e-16 ***
population_d 2.382e-04  1.713e-05  13.907   <2e-16 ***
distance_km  4.726e-04  4.974e-04   0.950    0.342
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 607.5 on 69648 degrees of freedom
Multiple R-squared:  0.004866,  Adjusted R-squared:  0.004823
F-statistic: 113.5 on 3 and 69648 DF,  p-value: < 2.2e-16
```

6.2.2 Log-linear regression model

```
Call:
lm(formula = log(count) ~ log(population_o) + log(population_d) +
    log(distance_km), data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-2.5912 -1.2159 -0.4253  0.5771  7.8842

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       -7.187371   0.217141 -33.100   <2e-16 ***
log(population_o)  0.421124   0.011617  36.250   <2e-16 ***
log(population_d)  0.315283   0.011083  28.448   <2e-16 ***
log(distance_km)   0.005887   0.010177   0.578    0.563
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.732 on 69648 degrees of freedom
Multiple R-squared:  0.02867,   Adjusted R-squared:  0.02863
F-statistic: 685.2 on 3 and 69648 DF,  p-value: < 2.2e-16
```

6.2.3 Logistic regression model with aggregated data

6.2.4 Poisson regression model

```
Call:
glm(formula = count ~ population_o + population_d + distance_km,
    family = poisson, data = df)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.813e+00  1.154e-03 3303.33   <2e-16 ***
population_o 1.604e-06  2.055e-09  780.54   <2e-16 ***
population_d 1.710e-06  1.986e-09  861.12   <2e-16 ***
```

```
distance_km  4.436e-06  8.285e-08   53.55   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 38519457  on 69651  degrees of freedom
Residual deviance: 37499031  on 69648  degrees of freedom
AIC: 37751790

Number of Fisher Scoring iterations: 8
```

6.3 Extensions

6.3.1 Including push-pull factors

6.3.2 Negative binomial regression model

6.3.3 Generalised linear mixed gravity models

6.3.4 Machine learning gravity models

References

Aparicio Castro A, Wiśniowski A, Rowe F (2023, 11) A bayesian approach to estimate annual bilateral migration flows for south america using census data. *Journal of the Royal Statistical Society Series A: Statistics in Society* 187: 410–435. CrossRef

Fotheringham A, O'Kelly M (Eds.) (1989) *Spatial Interaction Models: Formulations and Applications*. Kluwer Academic Publishers. Dordrechit

Rowe F, Calafiore A, Arribas-Bel D, Samardzhiev K, Fleischmann M (2023) Urban exodus? Understanding human mobility in Britain during the COVID-19 pandemic using Meta-Facebook data. *Population, Space and Place* 29: e2637

Rowe F, Lovelace R, Dennett A (2024, 05) *Spatial interaction modelling: a manifesto*, 177–196. Edward Elgar Publishing

Rowe F, Mahony M, Tao S (2022) Assessing machine learning algorithms for near-real time bus ridership prediction during extreme weather. CrossRef

Simini F, Barlacchi G, Luca M, Pappalardo L (2021, 11) A deep gravity model for mobility flows generation. *Nature Communications* 12. CrossRef

Wilson AG (1971, 03) A family of spatial interaction models, and associated developments. *Environment and Planning A: Economy and Space* 3: 1–32. CrossRef