Published in "The Stata journal", 2022, vol. 22, no. 1, pp. 89-102, which should be cited to refer to this work.
DOI:10.1177%2F1536867X221083857

The Stata Journal (2022)
**22**, Number 1, pp. 1–14 DOI: !!

# Travel distance and travel time using Stata: New features and major improvements in georoute

Sylvain Weber
University of Applied Sciences and Arts of Western Switzerland (HES-SO)
Geneva, Switzerland
sylvain.weber@hesge.ch

Martin Péclat
Romande Energie SA
Morges, Switzerland
martin.peclat@bluewin.ch

August Warren
Office of the State Superintendent of Education
Washington, DC
augustjwarren@gmail.com

**Abstract.** The community-contributed command `georoute` is designed to calculate travel distance and travel time between two addresses or two geographical points identified by their coordinates. Since its conception and description by Weber and Péclat (2017, *Stata Journal* 17: 962–971), the command has been gradually maintained and enriched. The new version of `georoute` presented in this article encompasses major improvements, such as the possibility to specify transport mode and departure time. The new features open the way to a multitude of more sophisticated research applications.

**Keywords:** dm0092_1, georoute, georoutei, geocoding, travel distance, travel time

## 1 Introduction

Weber and Péclat (2017) introduced the community-contributed command `georoute`, which calculates travel distance and travel time between two points defined by their addresses or their geographical coordinates. Four years after publication, the Stata Journal article and Stata module have received over 50 citations from applications in various research fields including transportation economics (for example, Theisen [2020]), health and medicine (for example, Fischer, Royer, and White [2018], Myers, Jones, and Upadhyay [2019], and Dayal et al. [2019]), education (for example, Delaney and Devereux [2020]), sustainability (for example, Makov et al. [2020]), and finance (for example, Marques and Alves [2021]), thereby demonstrating the usefulness of the command for many researchers.

The capabilities of the first version of `georoute` were relatively modest. Travel distance was calculated as the number of miles (or kilometers) one should drive by car to join the departure point to the destination point. No other transport means were possible, and it was not possible to introduce any kind of restriction on the road to be followed. Travel time was calculated as the time needed to cover the travel distance

 

"under normal traffic conditions", admittedly a rather vague expression. Specifying departure time was not possible.

The new release of the command presented in this article overcomes several limitations. Important new options have been implemented to let the user specify transport mode, departure time, and various features of the route to be followed. The new version of `georoute` therefore constitutes a major improvement and will allow much more sophisticated applications than its first version.

# 2 The updated command

## 2.1 georoute

The syntax of `georoute` (version 3.0) is as follows:

`georoute` [ *if* ] [ *in* ], `herekey(`*api_key*`)`
{`startaddress(`*varlist*`)` | `startxy(`*xvar yvar*`)`}
{`endaddress(`*varlist*`)` | `endxy(`*xvar yvar*`)`} [ `tmode(`*string* | *varname*`)`
`rtype(`*string* | *varname*`)` `dtime(`{*string*`,`*mask*} | *varname* | `"now")` `avoid(`*string*`)`
`distance(`*newvar*`)` `time(`*newvar*`)` `diagnostic(`*newvar*`)` `coordinates(`*str1 str2*`)`
`replace km timer pause observations nosettings` ]

### Main (compulsory)

`herekey(`*api_key*`)` provides the credentials of the HERE application to be used. `herekey()` is required.

`startaddress(`*varlist*`)` and `endaddress(`*varlist*`)` specify the addresses of departure and destination, respectively. Addresses can be inserted as a single variable or a variable list. Alternatively, `startxy()` and `endxy()` can be used. Either `startaddress()` or `startxy()` is required; either `endaddress()` or `endxy()` is required.

`startxy(`*xvar yvar*`)` and `endxy(`*xvar yvar*`)` specify the geographical coordinates (in decimal degrees) of the departure and destination points, respectively. *xvar* and *yvar* must be numeric variables containing latitude ($x$) and longitude ($y$) coordinates of the starting and ending points. Alternatively, `startaddress()` and `endaddress()` can be used. Either `startxy()` or `startaddress()` is required; either `endxy()` or `endaddress()` is required.

### Routing options

`tmode(`*string*|*varname*`)` specifies the transport mode. The following *string*s are available transport modes:

- `"car"` (default)
- `"publicTransit"`
- `"pedestrian"`
- `"bicycle"`

The transport mode can be specified either via a string (for instance, `tmode("car")`) or via a variable (for instance, `tmode(vehicle)`). When a string is used, all observations will use the same transport mode. When a variable is used, it is possible to specify the transport mode at the observation level, in which case the variable must be a string variable composed of the transport modes exactly as above (including capitalization). Any missing values will be assigned the default transport mode (`"car"`).

`rtype(`*string*|*varname*`)` specifies the routing type. The following *string*s are available routing types:

- `"fast"` (default)
- `"short"`

The routing type can be specified either via a string (for instance, `rtype("fast")`) or via a variable (for instance, `rtype(routing)`). When a string is used, all observations will use the same routing type. When a variable is used, it is possible to specify the routing type at the observation level, in which case the variable must be a string variable composed of the routing types exactly as above (possibly abbreviated). Any missing values will be assigned the default routing type (`"fast"`).

`dtime(`{*string*`,`*mask*}|*varname*|`"now")` specifies the date and time travel is expected to start. The default is `dtime("now")`, that is, the current time as of running the calculation.

The departure time can be specified either via a string and a mask (for instance, `dtime("01Nov2020 08:00:00", "DMYhms")`; see [FN] **Date and time functions**) or via a variable (for instance, `dtime(t)`). When a string is used, all observations will use the same departure time. When a variable is used, it is possible to specify the departure time at the observation level, in which case the format of the variable must be `%tc` or `%tC`. Any missing values will be assigned the default departure time (`"now"`).

`dtime()` has no impact for transport modes `"pedestrian"` and `"bicycle"`. The extent to which it is possible to calculate travel times in the past depends on other parameters, in particular, the transport mode specified in `tmode()`. Moreover, it seems that historical traffic data are only available from the HERE API for a few months back, resulting in a travel time independent of departure time for older time periods.

`avoid(`*string*`)` can be used to specify routing features to be avoided. The following *string*s are available routing features:

- "tollRoad"
- "ferry"
- "tunnel"
- "dirtRoad"

### New variables

`distance(`*newvar*`)` creates a new variable containing the travel distance between the departure and destination points. By default, travel distance will be stored in a variable named `travel_distance`.

`time(`*newvar*`)` creates a new variable containing the travel time between the departure and destination points. By default, travel time will be stored in a variable named `travel_time`.

`diagnostic(`*newvar*`)` creates a new variable containing a diagnostic code for the geocoding and georouting outcome of each observation in the database: `0` = `OK`, `1` = `No route found`, `2` = `Start and/or end not geocoded`, `3` = `Start and/or end coordinates missing`, and `4` = `No route searched`. By default, diagnostic codes will be stored in a variable named `georoute_diagnostic`.

`coordinates(`*str1 str2*`)` creates the new variables *str1*`_x`, *str1*`_y`, *str1*`_match`, *str2*`_x`, *str2*`_y`, and *str2*`_match`, which contain the coordinates and the match code of the starting (*str1*`_x`, *str1*`_y`, *str1*`_match`) and ending (*str2*`_x`, *str2*`_y`, *str2*`_match`) addresses. This option is irrelevant if geographical coordinates (rather than addresses) are provided. By default, coordinates and match code are not saved. The match code indicates how well the result matches the request in a 4-point scale: `1` = `exact`, `2` = `ambiguous`, `3` = `upHierarchy`, and `4` = `ambiguousUpHierarchy`.

`replace` specifies that the variables in the `distance()`, `time()`, `diagnostic()`, and `coordinates()` options be replaced if they already exist in the database. `replace` should be used cautiously because it might definitively drop some data.

### Reporting

`km` specifies that distances be returned in kilometers. The default is to return distances in miles.

`timer` requests that a timer be printed while geocoding. If specified, a dot is printed for every geocoded centile of the dataset, and the number corresponding to every decile is printed. When geocoding large numbers of observations, `timer` will inform the user of the expected end time.

`pause` slows the geocoding process by asking Stata to sleep for 30 seconds every 100th observation. This could be useful for large databases, which might overload the HERE API and result in missing values for batches of observations.

`observations` prints a detailed observation account, showing how many observations were discarded and why.

`nosettings` suppresses display of the settings report.

## 2.2 georoutei

To facilitate quick requests for single pairs of addresses or geographical coordinates, a command with immediate arguments proves handy. The syntax of `georoutei` is similar to that of `georoute`:

`georoutei, herekey(`*api_key*`)` {`startaddress(`*string*`)` | `startxy(#x,#y)`}
{`endaddress(`*string*`)` | `endxy(#x,#y)`} [`tmode(`*string*`)` `rtype(`*string*`)`
`dtime(`{*string*`,`*mask*} | `"now")` `avoid(`*string*`)` `km` `nosettings`]

### 2.2.1 Options

Options are as described in section 2.1, except that all arguments must be parsed as strings.[1]

### 2.2.2 Stored results

`georoutei` stores the following in `r()`:

Scalars

| | |
|---|---|
| `r(dist)` | travel distance |
| `r(time)` | travel time |
| `r(startx)` | $x$ coordinate (latitude) of departure point |
| `r(starty)` | $y$ coordinate (longitude) of departure point |
| `r(endx)` | $x$ coordinate (latitude) of destination point |
| `r(endy)` | $y$ coordinate (longitude) of destination point |

# 3 Examples

## 3.1 Using georoutei

We start our demonstration with the immediate command `georoutei`, which is simpler to use than the full command. For new users, it is probably a good idea to get familiar with the immediate command before trying to use the full one.

1. The options relative to the creation of new variables and some reporting options are not available with the immediate command because they are irrelevant in this context.

Say that we are interested in traveling from the New York Stock Exchange to the Madison Square Garden. We would type the following:[2]

```
. global nyse "11 Wall St, New York, NY 10005, USA"
. global msg "4 Pennsylvania Plaza, New York, NY 10001, USA"
. georoutei, herekey("$apikey") startaddress("$nyse") endaddress("$msg")
SETTINGS
------------------------------------------------------------------------------
Start:     11 Wall St, New York, NY 10005, USA (40.70714,-74.01086)
End:       4 Pennsylvania Plaza, New York, NY 10001, USA (40.7506734,-73.9923478)
Mode:      car (assigned by default)
Route:     fast (assigned by default)
Departure: 22 Nov 2021 16:43:22 (now; assigned by default)
------------------------------------------------------------------------------

ROUTE CALCULATED
------------------------------------------------------------------------------
Travel distance:       4.17 miles
Travel time:          19.52 minutes
------------------------------------------------------------------------------
```

We started by storing the addresses in global macros to simplify the code and also to be able to use the same addresses later without having to retype them.

The output indicates that travel distance between these two places is around 4.2 miles, and it could be covered in a bit less than 20 minutes. Said otherwise, the average speed (that could be computed by typing `display r(dist)/r(time)*60`) would be just 13 MPH.

Because we did not specify anything other than the departure and destination points, all the options were automatically set to their defaults. The first block of output shows the settings and indeed indicates that transport mode was assigned to `"car"`, routing type was assigned to `"fast"`, and departure time was assigned to `"now"` (which is the time when the user runs the command).



2. Before running these lines, store the API key for your HERE application in the global macro `$apikey`.

In the previous release of the command, none of the default settings could be altered. With this new version, however, we can change, among other options, the transport mode:

```
. georoutei, herekey("$apikey") startaddress("$nyse") endaddress("$msg")
> tmode("bicycle")
SETTINGS
------------------------------------------------------------------------------
Start:     11 Wall St, New York, NY 10005, USA (40.70714,-74.01086)
End:       4 Pennsylvania Plaza, New York, NY 10001, USA (40.7506734,-73.9923478)
Mode:      bicycle
Route:     (no impact with selected transport mode)
Departure: (no impact with selected transport mode)
------------------------------------------------------------------------------

ROUTE CALCULATED
------------------------------------------------------------------------------
Travel distance:       3.53 miles
Travel time:          28.08 minutes
------------------------------------------------------------------------------

. georoutei, herekey("$apikey") startaddress("$nyse") endaddress("$msg")
> tmode("publicTransit")
SETTINGS
------------------------------------------------------------------------------
Start:     11 Wall St, New York, NY 10005, USA (40.70714,-74.01086)
End:       4 Pennsylvania Plaza, New York, NY 10001, USA (40.7506734,-73.9923478)
Mode:      publicTransit
Route:     (no impact with selected transport mode)
Departure: 22 Nov 2021 16:43:24 (now; assigned by default)
------------------------------------------------------------------------------

ROUTE CALCULATED
------------------------------------------------------------------------------
Travel distance:       3.60 miles
Travel time:          18.00 minutes
------------------------------------------------------------------------------
```

We observe that travel distance in this case is a bit shorter with a bicycle than with a car, which is caused by the HERE API considering some roads (such as those located in parks) as open to cyclists but not to cars. Nevertheless, travel time is naturally longer with a bicycle. The output indicates that accounting for departure time would have no impact on the outcome for travel by bicycle. Finally, we see that the fastest travel time is calculated for public transport, with a travel time lower than both car travel and bicycle travel.

Rather than allow the time of departure to be assigned by default (and therefore possibly obtain a different outcome after every run), we can specify the date and time of departure, therefore taking into account historical traffic information:

```
. georoutei, herekey("$apikey") startaddress("$nyse") endaddress("$msg")
> tmode("car") dtime("15nov2021 12:00", "DMYhm") noset
ROUTE CALCULATED
------------------------------------------------------------------------------
Travel distance:       4.17 miles
Travel time:          18.55 minutes
------------------------------------------------------------------------------

. georoutei, herekey("$apikey") startaddress("$nyse") endaddress("$msg")
> tmode("car") dtime("15nov2021 23:59", "DMYhm") noset
ROUTE CALCULATED
------------------------------------------------------------------------------
Travel distance:       3.97 miles
Travel time:          13.68 minutes
------------------------------------------------------------------------------
```

Here we additionally used option `nosettings` to save space by suppressing display of the settings. The results show that both travel distance and travel time would be different if driving at noon versus at midnight on Monday, November 15, 2021. Travel distance changes with hours of the day because of possible time-zone restrictions, while travel time is of course mostly affected by congestion during peak hours.

The last series of options available with `georoutei` allow us to introduce various restrictions on the route to be followed (or not). To illustrate these options, we consider a trip from Geneva, Switzerland to Saint-Tropez, France:

```
. global GVA "Geneva, Switzerland"
. global ST "Saint-Tropez, France"
. georoutei, herekey("$apikey") startaddress("$GVA") endaddress("$ST") km noset
ROUTE CALCULATED
------------------------------------------------------------------------------
Travel distance:     562.90 kilometers
Travel time:         346.33 minutes
------------------------------------------------------------------------------

. georoutei, herekey("$apikey") startaddress("$GVA") endaddress("$ST")
> rtype("short") km noset
ROUTE CALCULATED
------------------------------------------------------------------------------
Travel distance:     445.49 kilometers
Travel time:         470.60 minutes
------------------------------------------------------------------------------

. georoutei, herekey("$apikey") startaddress("$GVA") endaddress("$ST")
> avoid("toll") km noset
ROUTE CALCULATED
------------------------------------------------------------------------------
Travel distance:     483.13 kilometers
Travel time:         492.55 minutes
------------------------------------------------------------------------------
```

For this trip, we observe that the default settings yield a travel distance of more than 560 kilometers.[3]

If we ask the API to look for the shortest distance, by using option `rtype("short")`, we obtain a travel distance more than 100 km shorter. Notice, however, that travel duration is much longer. These differences are because the shortest route goes through mountainous regions using small roads. Contrarily, the highway forces drivers to make detours, lengthening the distance driven, but of course permits a much faster travel speed.

Finally, note that avoiding toll roads, by using option `avoid("toll")`, results in the longest trip duration. This clearly illustrates how the French road legislation imposes tradeoffs between travel costs and travel time, which certainly contributed to the discontent that dramatically led to the "yellow vests movement".[4]

## 3.2 Using georoute

The full command, `georoute`, makes it possible to calculate travel distances and travel times for batches of observations. Concretely, it executes the exact same requests as `georoutei` but for each observation of the dataset; it therefore returns the results in variables rather than printing them in the Results window.

To show the equivalence between the two commands, we create a small dataset and replicate the calculations of travel distance and travel time between the New York Stock Exchange and the Madison Square Garden:

```
. clear
. set obs 3
Number of observations (_N) was 0, now 3.
. generate place1 = "11 Wall St, New York, NY 10005, USA"
. generate place2 = "4 Pennsylvania Plaza, New York, NY 10001, USA"
. quietly generate vehicle = "car" in 1
. quietly replace vehicle = "bicycle" in 2
. quietly replace vehicle = "publicTransit" in 3
. georoute, herekey("$apikey") startaddress(place1) endaddress(place2)
> tmode(vehicle)
georoute has successfully calculated travel distance and travel time for 3
> observations based on the following settings:
------------------------------------------------------------------------------
Start:     addresses in place1 (variable)
End:       addresses in place2 (variable)
Mode:      transport modes in vehicle (variable)
Route:     fast (assigned by default) for all observations (hard coded)
Departure: 22 Nov 2021 17:36:28 (now; assigned by default) for all observations
> (hard coded)
------------------------------------------------------------------------------
. format travel_distance travel_time %4.2f
```

3. We used option `km` for consistency with European standards for this example.
4. See, for instance, *The Economist* (27.11.2018): "What, and who, are France's 'gilets jaunes'?".

```
. list vehicle travel_distance travel_time
```

| | vehicle | trave~ce | trave~me |
|---|---|---|---|
| 1. | car | 4.17 | 19.07 |
| 2. | bicycle | 3.53 | 28.08 |
| 3. | publicTransit | 3.60 | 19.00 |

The results are equivalent to those obtained with `georoutei` in section 3.1.[5]

To demonstrate the value added by option `dtime()`, let us consider the trip from the United Nations' building in Geneva, Switzerland to Geneva's International Airport. In distance, this is a short trip (less than 5 kilometers). However, because of important traffic congestion, travel time may sometimes be quite long (which may have serious consequences when trying to catch a plane). For this specific trip, we compute travel times using three different transport modes and for each hour of a week:

```
. clear
. capture georoutei, herekey($apikey)
> startad("Palais des Nations, 1211 Genève, Switzerland")
> endad("Route de l'Aéroport 21, 1215 Le Grand-Saconnex, Switzerland")
. set obs 3
Number of observations (_N) was 0, now 3.
. generate startx = r(startx)
. generate starty = r(starty)
. generate endx = r(endx)
. generate endy = r(endy)
. quietly generate vehicle = "car" in 1
. quietly replace vehicle = "bicycle" in 2
. quietly replace vehicle = "publicTransit" in 3
. expand 24*7
(501 observations created)
. local today = date(c(current_date),"DMY")
. local Monday1: di %td `today' - 6 - dow(`today')
. local Monday2: di %td td(`Monday1') + 7
. bysort vehicle: generate double t = tc(`Monday1' 00:00) + msofminutes(60)*(_n-1)
. format t %tc
```

5. The equivalence holds as long as the two commands are issued approximately at the same time, because departure time is set by default at the current time.

```
. georoute, herekey("$apikey") startxy(startx starty) endxy(endx endy)
> tmode(vehicle) dtime(t) timer observations

---------
Geocoding
---------
......... 10% ......... 20% ......... 30% ......... 40% ......... 50% .........
> 60% ......... 70% ......... 80% ......... 90% ......... 100%

georoute has successfully calculated travel distance and travel time for 485
> observations based on the following settings:
------------------------------------------------------------------------------

Start:     coordinates in startx starty (variables)
End:       coordinates in endx endy (variables)
Mode:      transport modes in vehicle (variable)
Route:     fast (assigned by default) for all observations (hard coded)
Departure: times in t (variable)
------------------------------------------------------------------------------

Detailed observation account:
------------------------------------------------------------------------------

Total in database:                          504
Total after if/in condition(s):             504
Considered for geocoding:                   504
Successfully coded:                         504
------------------------------------------------------------------------------

. encode vehicle, gen(v)

. xtset v t, delta(60 min)
       panel variable:  v (strongly balanced)
        time variable:  t, 15nov2021 00:00:00 to 21nov2021 23:00:00
                delta:  1 hour

. xtline travel_time, over ylabel(10(10)60) yscale(r(10 60))
> xlabel(`=tc(`Monday1' 00:00)'(`=msofminutes(60)*24')`=tc(`Monday2' 00:00)',
> format(%tcDDMon)) xtitle("") legend(row(1))
```

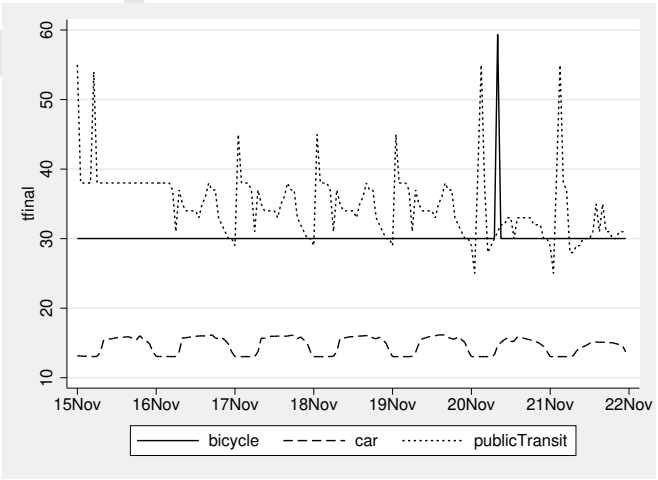


Figure 1. Travel time over the week and for different transport modes

In this example, we use `georoutei` in a preliminary step to geocode the addresses. The obtained geographical coordinates are then saved in variables and fed into the command `georoute`. We do this to avoid multiple requests for the same addresses. If the addresses were introduced in `georoute`, the same pair of addresses would in fact be geocoded as many times as the number of observations, which is clearly a waste of resources (the number of requests is capped at 250,000 per month with a free HERE account) and time. Providing the geographical coordinates to `georoute` hence saves requests to the HERE API and speeds up the process. The database is then expanded, and the variable `t` is filled with each hour of the previous week (in our case, the week that started on Monday, November 15, 2021 at 00:00).

Travel times (and distances) for all 504 observations of the database are then calculated using a single call to `georoute`. Because geocoding could take a while, we use the `timer` option to keep track of the evolution of the process.

Figure 1 displays the travel times calculated for each transport mode. We observe that travel time is constant when traveling by bicycle, as expected because this transport mode is not affected by traffic and departure time.[6] There are, however, large variations for traveling by car and by public transport.[7] Travel time by car is longer during the day, especially on weekdays. The pattern is different for public transportation, for which travel time is longer at night when connections are less frequent. We also notice that, on average, travel time is generally shorter with a bicycle than with public transportation, a well-known specificity of the city of Geneva.

## 4 Caveats

The community-contributed Stata command `georoute` automates requests from the HERE API, which is convenient for extracting travel distances and times. However, the HERE API might suffer from some weaknesses, which then also affect `georoute`'s outcome. For instance, the authors of this article have noticed that travel times and distances obtained for a series of similar observations—such as those in the example of section 3.2—are sometimes aberrant or missing for no apparent reason. In that case, one working solution is to run `georoute` twice on the same dataset and keep the smallest time and distance values obtained.

`georoute` should not be considered responsible for the quality of results, which depend entirely on the HERE API. `georoute` users are invited to use the HERE documentation (https://developer.here.com/) to understand precisely each parameter of the requests. The authors of this article are not affiliated in any respect with HERE and should not be considered experts of this platform.

6. The spike at almost 60 minutes obtained on November 20 is obviously an aberrant value. More on this in section 4.
7. The constant travel time obtained for public transport for most of November 15 is also suspicious.

# 5 Conclusion

In this article, we presented the new functionalities available in the updated version of `georoute`, a command to determine travel distances and times between addresses or geographical coordinates. As illustrated in various examples, the improved version of the command opens the way to a multitude of potential applications. The updated command allows comparisons of the performance of competing transport modes in terms of time and distance for a specific trip. It is also now possible to determine the most appropriate departure time.

The `georoute` command, like all API-based commands, is dependent on the stability of these interfaces. This risk, which we already mentioned in Weber and Péclat (2017), has so far proved to be moderate. Although we had to implement updates in recent years, these have been marginal and mostly related to the HERE credentials. The new functionalities offered by HERE, which enabled us to improve `georoute`, have been added without modifying the basic operation of the API. Because of this, we believe that `georoute` will continue to benefit the Stata user community.

# 6 Programs and supplemental materials

To install a snapshot of the corresponding software files as they existed at the time of publication of this article, type

```
. net sj 22-1
. net install dm0092_1      (to install program files, if available)
. net get dm0092_1          (to install ancillary files, if available)
```

# 7 References

Dayal, P., C. H. Chang, W. S. Benko, A. M. Ulmer, S. S. Crossen, B. H. Pollock, J. S. Hoch, J. L. Kissee, L. Warner, and J. P. Marcin. 2019. Appointment completion in pediatric neurology telemedicine clinics serving underserved patients. *Neurology: Clinical Practice* 9: 314–321. https://doi.org/10.1212/CPJ.0000000000000649.

Delaney, J. M., and P. J. Devereux. 2020. Choosing differently? College application behavior and the persistence of educational advantage. *Economics of Education Review* 77: 101998. https://doi.org/10.1016/j.econedurev.2020.101998.

Fischer, S., H. Royer, and C. White. 2018. The impacts of reduced access to abortion and family planning services on abortions, births, and contraceptive purchases. *Journal of Public Economics* 167: 43–68. https://doi.org/10.1016/j.jpubeco.2018.08.009.

Makov, T., A. Shepon, J. Krones, C. Gupta, and M. Chertow. 2020. Social and environmental analysis of food waste abatement via the peer-to-peer sharing economy. *Nature Communications* 11: 1156. https://doi.org/10.1038/s41467-020-14899-5.

Marques, B. P., and C. F. Alves. 2021. The profitability and distance to distress of European banks: Do business choices matter? *European Journal of Finance* 27: 1553–1580. https://doi.org/10.1080/1351847X.2021.1897638.

Myers, C., R. Jones, and U. Upadhyay. 2019. Predicted changes in abortion access and incidence in a post-Roe world. *Contraception* 100: 367–373. https://doi.org/10.1016/j.contraception.2019.07.139.

Theisen, T. 2020. The impact of an urban toll ring on housing prices. *Research in Transportation Economics* 82: 100882. https://doi.org/10.1016/j.retrec.2020.100882.

Weber, S., and M. Péclat. 2017. A simple command to calculate travel distance and travel time. *Stata Journal* 17: 962–971. https://doi.org/10.1177/1536867X1801700411.

**About the authors**

Sylvain Weber is an assistant professor at the University of Applied Sciences and Arts of Western Switzerland (HES-SO). His research areas are in energy and labor economics.

Martin Péclat is a quantitative analyst at Romande Energie SA. He recently obtained his PhD in economics, which mainly focuses on the spatial diffusion of solar photovoltaic technology in Switzerland. This paper was mostly written when Martin was affiliated with the Institute of Economic Research of the University of Neuchâtel (Switzerland).

August Warren is a data scientist and researcher currently working in cybersecurity. Most recently, August worked in the Office of Research, Analysis, and Reporting of the DC Office of the State Superintendent of Education, where he frequently used `georoute` to examine patterns in the geographic distribution and socioeconomic status of students in research on attendance, statewide accountability programs, and assessments in Washington, DC.