r5r: Rapid Realistic Routing on Multimodal Transport Networks with R^5 in R

Rafael H. M. Pereira[1], Marcus Saraiva[1], Daniel Herszenhut[1], Carlos Kaue Braga[1], Matthew Wigginton Conway[2]

1 – Institute for Applied Economic Research (Ipea), Brazil

2 – School of Geographical Sciences and Urban Planning, Arizona State University

Abstract:

Routing is a key step in transport planning and research. Nonetheless, researchers and practitioners often face challenges when performing this task due to long computation times and the cost of licensed software. Conveyal's R^5 is a multimodal transport network router that offers multiple routing features, such as calculating travel times over a time window and returning multiple itineraries for origin/destination pairs. This paper describes r5r, an open-source R package that leverages R^5 to efficiently compute travel time matrices and generate detailed itineraries between sets of origins and destinations at no expense using seamless parallel computing.

Keywords: routing, transport networks, travel time, accessibility, travel impedance

RESEARCH QUESTIONS AND HYPOTHESES

Transport routing is the process of finding the fastest or lowest-cost routes that connect places in a given transport network, and is a key step in transport accessibility analysis, fleet allocation and transport simulation and planning more broadly (Levinson and et al. 2020). However, researchers and practitioners often face practical challenges when carrying out routing tasks due to the costs of licensed software, limited data availability, and the long computation times required to run multiple routing scenarios, particularly in large and complex multimodal transport networks.

While there are a growing number of open-source routing packages (OpenTripPlanner n.d.; Lovelace and Ellison 2019; Padgham 2019), most options available do not efficiently process large public transport networks and/or are not user-friendly. This paper presents r5r, a new open-source R package for routing on multimodal transport networks based on the Rapid Realistic Routing on Real-world

and Reimagined networks (R5) package. R5 is a powerful next-generation routing engine written in Java and developed at Conveyal (Conway, Byrd, and van der Linden 2017; Conway, Byrd, and van Eggermond 2018) to provide an efficient backend for analytic applications, such as accessibility analysis. The r5r package provides a simple and friendly interface to run R5 locally from within R, which allows users to efficiently calculate travel time matrices or generate multiple route alternatives between origins and destinations using seamless parallel computing.

METHODS AND DATA

The r5r package has low data requirements and is easily scalable, allowing fast computation of routes and travel times for either city or region-level analysis. It creates a routable transport network using street network data from OpenStreetMap (OSM) and optionally public transport data in the General Transit Feed Specification (GTFS) format.

The r5r package has 3 fundamental functions:

- `setup_r5()`: builds a multimodal transport network used for routing in R5. This function automatically (1) downloads/updates a compiled R5 JAR file and stores it locally for future use; and (2) combines the OSM and GTFS datasets to build a routable network object.

- `travel_time_matrix()`: computes travel time estimates between one or multiple origin/destination pairs for a single departure time or for multiple departure times over a `time_window` set by the user. This function uses an R5-specific extension to the RAPTOR routing algorithm which provides an efficient and systematic sampling of multiple simulated schedules when using frequency-based GTFS data (Conway, Byrd, and van der Linden 2017).

- `detailed_itineraries()`: computes detailed information on routes between one or multiple origin/destination pairs for a single departure time. The output includes detailed information on route alternatives such as the transport mode, waiting time, travel time and distance of each segment of the trip. This function uses an R5-specific extension[1] to the McRAPTOR (Delling, Pajor, and Werneck 2015) routing algorithm to find both optimal and slightly suboptimal paths.

Both routing functions are versatile so users can easily set customized inputs such as transport modes, departure dates and times, walking and cycling speeds, maximum trip duration, walking distances and number of public transport transfers. In the following section, we will focus on results obtained from `travel_time_matrix()`.

---

[1] The specific extension to McRAPTOR to do suboptimal path routing is not documented yet.

FINDINGS

After it is installed with the `install.packages("r5r")` command, the package can be attached (alongside other packages to reproduce this article), as follows:

```
library(r5r)
library(sf)
library(data.table)
library(ggplot2)
library(akima)
library(dplyr)
```

Code 1: Load required libraries

For this article, we used r5r version v0.3-2 and R[5] version v6.0.1.

The package includes sample datasets for the cities of São Paulo and Porto Alegre (both in Brazil). Each dataset includes:

- An OSM network in `.pbf` format.
- A public transport network in `GTFS.zip` format.
- The spatial coordinates of points covering the area in `.csv` format, including information on the size of resident population and the number of schools in each location.

Building a routable transport network

To build a routable transport network with r5r and load it into memory, the user needs to call `setup_r5` with the path to the directory where OSM and GTFS data are stored. In the examples herein, we use the provided Porto Alegre dataset.

```
# system.file returns the directory with example data inside the r5r
package
# set data path to directory containing your own data if not using the
examples
data_path <- system.file("extdata/poa", package = "r5r")

r5r_core <- setup_r5(data_path, verbose = FALSE)
```

Code 2: Set up routable transport network

The function uses the `.pbf` and the `GTFS.zip` files in the directory pointed by `data_path` to create a multimodal transport network used for routing by R[5]. If multiple GTFS files are present, R[5] will merge them into a single transport network.

The resulting `network.dat` as well as some other files used by R5 are saved inside the supplied directory for later reuse.

Calculating a travel time matrix

The `travel_time_matrix()` function takes, as inputs, the spatial location of origins/destinations (either as a spatial `sf POINT` object, or as a `data.frame` containing the columns `id`, `lon` and `lat`) and a few travel parameters such as *maximum trip duration*, or *walking distance*. It outputs travel time estimates for each origin-destination pair at a set `departure_datetime`.

Since service levels can significantly vary across the day (Stępniak et al. 2019), r5r provides a `time_window` parameter. When this parameter is set, R5 will compute travel times for trips at the specified departure time and every minute for `time_window` minutes after. The `percentiles` parameter allows the user to retrieve travel time estimates at different points of the distribution (by default the median). These percentiles reflect service variation over the time window, but do not reflect schedule deviation not represented in the GTFS, though tools exist to create GTFS which reflects schedule deviations (Wessel, Allen, and Farber 2017).

An example of the function's usage is presented below. Computing this 1227x1227 travel time matrix with a 120-minute time window takes less than two minutes on a Windows machine with a 1.9GHz Intel i7 and 16GB RAM.

```r
# read points of origin and destination
points <- fread(file.path(data_path, "poa_hexgrid.csv"))

# routing inputs
mode <- c("WALK", "TRANSIT")
max_walk_dist <- 1000 # in meters
max_trip_duration <- 120 # in minutes
departure_datetime <- as.POSIXct("13-05-2019 14:00:00",
                                 format = "%d-%m-%Y %H:%M:%S")

time_window <- 120 # in minutes
percentiles <- c(5, 25, 50, 75, 95)

# calculate travel time matrix
computation_time <- system.time(ttm <- travel_time_matrix(r5r_core,
                         origins = points,
                         destinations = points,
                         mode = mode,
                         departure_datetime = departure_datetime,
                         max_walk_dist = max_walk_dist,
                         max_trip_duration = max_trip_duration,
                         time_window = time_window,
                         percentiles = percentiles,
                         verbose = FALSE))
print(paste('travel time matrix computed in',
computation_time[['elapsed']], 'seconds'))
#> [1] "travel time matrix computed in 58.14 seconds"
head(ttm)
#>             fromId            toId travel_time_p005 travel_time_p025
#> 1: 89a901291abffff 89a901291abffff                2                 2
#> 2: 89a901291abffff 89a901295b7ffff               38                41
#> 3: 89a901291abffff 89a9012809bffff               40                44
#> 4: 89a901291abffff 89a901285cfffff               32                34
#> 5: 89a901291abffff 89a90e934d7ffff               58                60
#> 6: 89a901291abffff 89a90129b47ffff               55                60
#>    travel_time_p050 travel_time_p075 travel_time_p095
#> 1:                2                2                2
#> 2:               45               48               51
#> 3:               48               51               55
#> 4:               38               41               43
#> 5:               62               65               69
#> 6:               64               68               76
```

Code 3: Compute travel time matrix

Visualizing travel-time uncertainty

The plot below shows how the travel times to arrive at the central bus station from several origin points vary within the time window (5th, 25th, 50th, 75th, and 95th percentiles), reflecting that travel times are more uncertain when leaving from some places than others. While there is little to no uncertainty when departing from places that are very close (walking distance) to the central bus station, travel times from places farther away are more affected by departure time variations and service frequency levels.

```
# subset travel time matrix departing from a given origin
central_bus_stn <- points[291,]
ttm_tw <- subset(ttm, toId %in% central_bus_stn$id)

# reshape data
plot_data <- setnames(ttm_tw, 'travel_time_p050', 'mediantt') %>%
             melt(., measure = patterns("^travel_time_p"),
                     variable = "percentile",
                     value = "travel_time")

# plot
ggplot(data=plot_data, aes(y = travel_time, x = reorder(fromId,
mediantt))) +
  geom_point(alpha = 0.1, size = .7) +
  geom_line(aes(y=mediantt, group=toId), color="#FE9F45", size=1.5) +
  expand_limits(y = 120) +
  scale_y_continuous(breaks = c(0, 30, 60, 90, 120)) +
  theme_minimal() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks = element_blank(),
        panel.border = element_rect(fill = NA, colour = "grey80",
size=1)) +
  labs(title = " ",
       y = "Travel Time (min)", x='Origins ordered by median travel
time')
```

Code: Figure 1

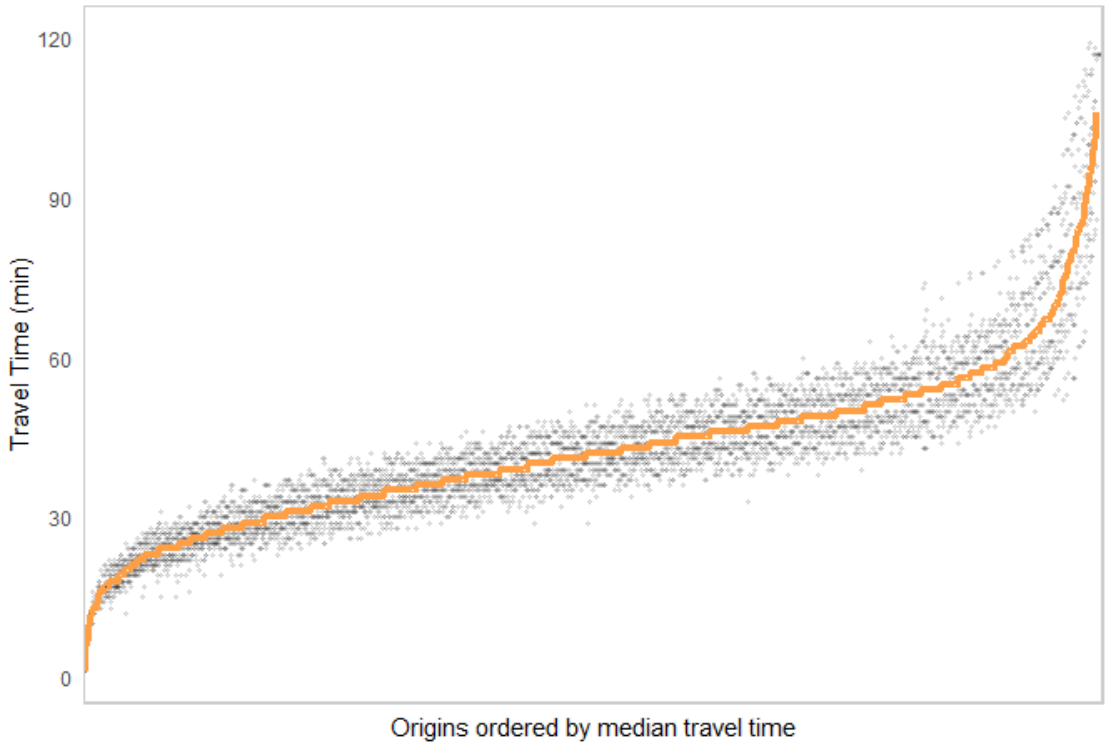

Figure 1: Travel time uncertainty by trip origin

Visualizing Isochrones

In our example, we can visualize the isochrone (area reachable within a certain amount of time) departing from the central bus station as follows:

```
# extract OSM network
street_net <- street_network_to_sf(r5r_core)

# select trips departing the bus central station and add coordinates of
destinations
travel_times <- ttm[fromId %in% central_bus_stn$id]
travel_times[points, on=c('toId' ='id'), `:=`(lon = i.lon, lat = i.lat)]

# interpolate estimates to get spatially smooth result
travel_times.interp <- with(na.omit(travel_times), interp(lon, lat,
travel_time_p050)) %>%
                        with(cbind(travel_time=as.vector(z),  # Column-
major order
                                   x=rep(x, times=length(y)),
                                   y=rep(y, each=length(x)))) %>%
  as.data.frame() %>% na.omit()

# find isochrone's bounding box to crop the map below
bb_x <- c(min(travel_times.interp$x), max(travel_times.interp$x))
bb_y <- c(min(travel_times.interp$y), max(travel_times.interp$y))

# plot
ggplot(travel_times.interp) +
  geom_contour_filled(aes(x=x, y=y, z=travel_time), alpha=.8) +
  geom_sf(data = street_net$edges, color = "gray55", size=0.1, alpha =
0.7) +
  geom_point(aes(x=lon, y=lat, color='Central bus\nstation'),
data=central_bus_stn) +
  scale_fill_viridis_d(direction = -1, option = 'B') +
  scale_color_manual(values=c('Central bus\nstation'='black')) +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_continuous(expand=c(0,0)) +
  coord_sf(xlim = bb_x, ylim = bb_y) +
  labs(fill = "travel time (minutes)", color='') +
  theme_minimal() +
  theme(axis.title = element_blank())
```

Code: Figure 2

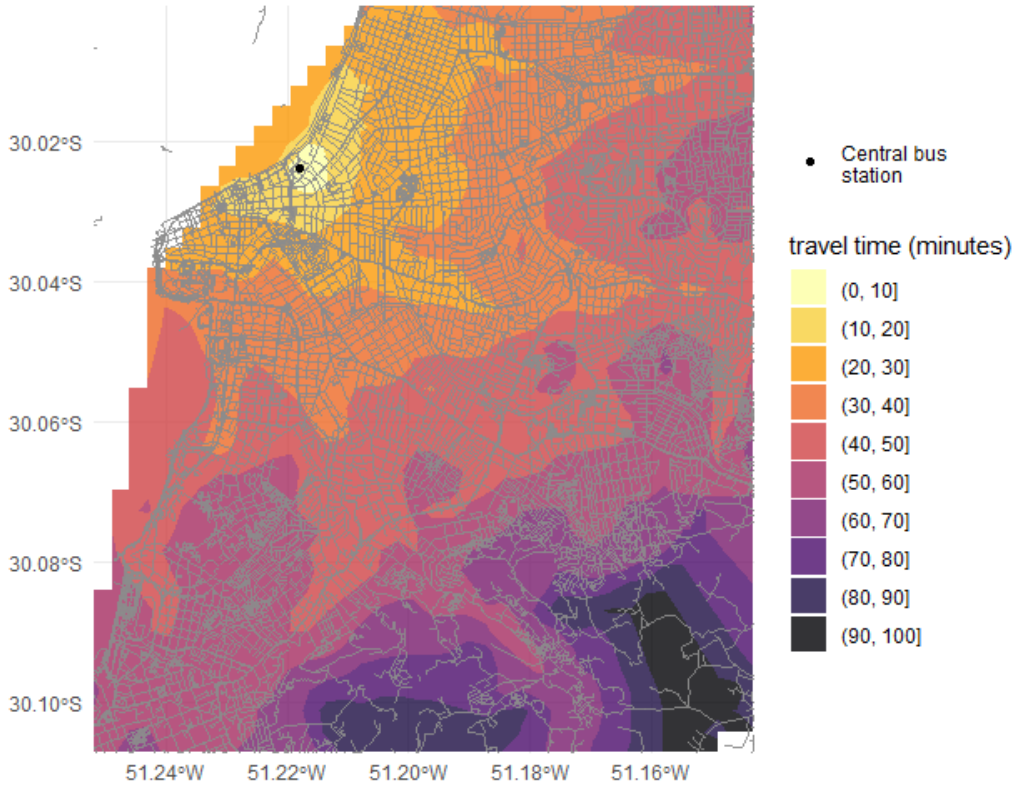

Figure 2: Isochrones by public transport departing from the central bus station

Creating accessibility metrics

Accessibility metrics measure the opportunities, such as jobs, a traveler could reach from a particular location (Levinson and et al. 2020). One of the simplest forms is a cumulative-opportunities metric, which sums all of the opportunities accessible from each location in less than a cutoff time. Using the travel time matrix and information on the number of opportunities available at each location, we can calculate and map accessibility. In the example below we compute the number of schools accessible by public transport in less than 20 minutes.

```
# merge schools information to travel time matrix
ttm[points, on=c('toId' ='id'), schools := i.schools]

# calculate number of schools accessible
access <- ttm[travel_time_p050 <= 20, .(acc = sum(schools)), by=fromId]

# interpolate estimates to get spatially smooth result
access.interp <- access %>%
  inner_join(points, by=c('fromId'='id')) %>%
  with(interp(lon, lat, acc)) %>%
                        with(cbind(acc=as.vector(z),  # Column-major
order
                                   x=rep(x, times=length(y)),
                                   y=rep(y, each=length(x)))) %>%
as.data.frame()

# plot
ggplot(na.omit(access.interp)) +
  geom_contour_filled(aes(x=x, y=y, z=acc), alpha=.8) +
  geom_sf(data = street_net$edges, color = "gray55", size=0.1, alpha =
0.7) +
  scale_fill_viridis_d(direction = -1, option = 'B') +
  scale_x_continuous(expand=c(0,0)) +
  scale_y_continuous(expand=c(0,0)) +
  coord_sf(xlim = bb_x, ylim = bb_y) +
  labs(fill = "Schools within\n20 minutes\n(median travel time)") +
  theme_minimal() +
  theme(axis.title = element_blank())
```

Code: Figure 3

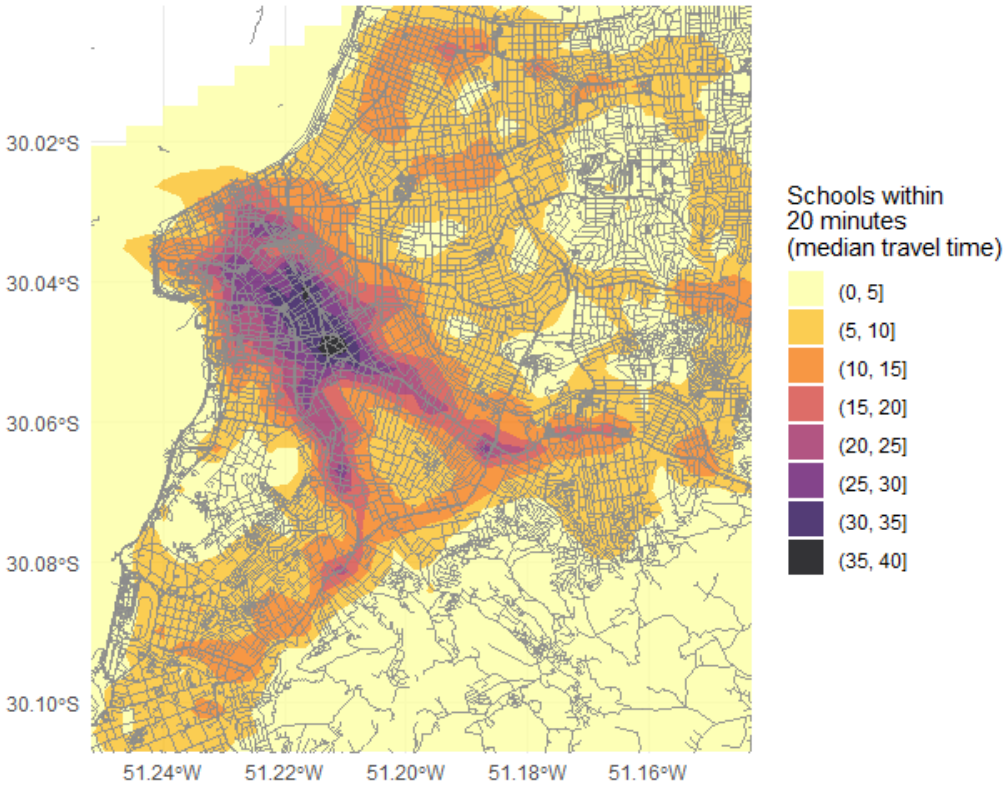

Figure 3: Number of schools accessible by public transport in less than 20 minutes

Acknowledgments

The R5 routing engine is developed at Conveyal with contributions from several developers. This work was supported by the Brazilian Institute for Applied Economic Research (Ipea).

References

Conway, Matthew Wigginton, Andrew Byrd, and Michael van Eggermond. 2018. "Accounting for Uncertainty and Variation in Accessibility Metrics for Public Transport Sketch Planning." *Journal of Transport and Land Use* 11 (1). https://doi.org/10.5198/jtlu.2018.1074.

Conway, Matthew Wigginton, Andrew Byrd, and Marco van der Linden. 2017. "Evidence-Based Transit and Land Use Sketch Planning Using Interactive Accessibility Methods on Combined Schedule and Headway-Based Networks." *Transportation Research Record: Journal of the Transportation Research Board* 2653 (1): 45–53. https://doi.org/10.3141/2653-06.

Delling, Daniel, Thomas Pajor, and Renato F. Werneck. 2015. "Round-Based Public Transit Routing." *Transportation Science* 49 (3): 591–604. https://doi.org/10.1287/trsc.2014.0534.

Levinson, David, and et al. 2020. "Transport Access Manual: A Guide for Measuring Connection Between People and Places," January. https://ses.library.usyd.edu.au/handle/2123/23733.

Lovelace, Robin, and Richard Ellison. 2019. "Stplanr: A Package for Transport Planning." *The R Journal* 10 (2): 7. https://doi.org/10.32614/RJ-2018-053.

OpenTripPlanner. n.d. "OpenTripPlanner." Accessed September 17, 2020. http://www.opentripplanner.org/.

Padgham, Mark. 2019. "Dodgr: An R Package for Network Flow Aggregation." *Transport Findings*. https://doi.org/10.32866/6945.

Stępniak, Marcin, John P. Pritchard, Karst T. Geurs, and Sławomir Goliszek. 2019. "The Impact of Temporal Resolution on Public Transport Accessibility Measurement: Review and Case Study in Poland." *Journal of Transport Geography* 75 (February): 8–24. https://doi.org/10.1016/j.jtrangeo.2019.01.007.

Wessel, Nate, Jeff Allen, and Steven Farber. 2017. "Constructing a Routable Retrospective Transit Timetable from a Real-Time Vehicle Location Feed and GTFS." *Journal of Transport Geography* 62 (June): 92–97. https://doi.org/10.1016/j.jtrangeo.2017.04.012.