7 Transport applications

Prerequisites

- This chapter requires the following packages to be loaded (it also uses **osmdata** and **nabor** although these do not need to be loaded):

```
library(sf)
library(tidyverse)
library(stplanr)
library(spDataLarge)
```

7.1 Introduction

In few other sectors is geographic space more tangible than transport. The effort of moving (overcoming distance) is central to the 'first law' of geography, defined by Waldo Tobler in 1970 as follows (Miller 2004):

> Everything is related to everything else, but near things are more related than distant things

This 'law' is the basis for spatial autocorrelation and other key geographic concepts. It applies to phenomena as diverse as friendship networks and ecological diversity and can be explained by the costs of transport — in terms of time, energy and money — which constitute the 'friction of distance'. From this perspective transport technologies are disruptive, changing geographic relationships between geographic entities including mobile humans and goods: "the purpose of transportation is to overcome space" (Rodrigue, Comtois, and Slack 2013).

Transport is an inherently geospatial activity. It involves traversing continuous geographic space between A and B, and infinite localities in between. It is therefore unsurprising that transport researchers have long turned to geocomputational methods to understand movement patterns and that transport problems are a motivator of geocomputational methods.

This chapter provides an introduction to geographic analysis of transport systems. We will explore how movement patterns can be understood at multiple geographic levels, including:

- **Areal units**: transport patterns can be understood with reference to zonal aggregates such as the main mode of travel (by car, bike or foot, for example) and average distance of trips made by people living in a particular zone.

- **Desire lines**: straight lines that represent 'origin-destination' data that records how many people travel (or could travel) between places (points or zones) in geographic space.
- **Routes**: these are circuitous (non-straight) routes, typically representing the 'optimal' path along the route network between origins and destinations along the desire lines defined in the previous bullet point.
- **Nodes**: these are points in the transport system that can represent common origins and destinations (e.g. with one centroid per zone) and public transport stations such as bus stops and rail stations.
- **Route networks**: these represent the system of roads, paths and other linear features in an area. They can be represented as geographic features (representing route segments) or structured as an interconnected graph. Each can be assigned values representing the level of traffic on different parts of the network, referred to as 'flow' by transport modelers (Hollander 2016).

Another key level is **agents**, mobile entities like you and me. These can be represented computationally thanks to software such as MATSim, which captures the dynamics of transport systems using an agent-based modelling (ABM) approach at high spatial and temporal resolution (Horni, Nagel, and Axhausen 2016). ABM is a powerful approach to transport research with great potential for integration with R's spatial classes (Thiele 2014; Lovelace and Dumont 2016), but is outside the scope of this chapter. Beyond geographic levels and agents, the basic unit of analysis in most transport models is the **trip**, a single purpose journey from an origin 'A' to a destination 'B' (Hollander 2016). Trips join-up the different levels of transport systems: they are usually represented as *desire lines* connecting *zone* centroids (*nodes*), they can be allocated onto the *route network* as *routes*, and are made by people who can be represented as *agents*.

Another complication is time. Although many trips are regular and predictable — such as the daily commute to work — transport systems are dynamic and constantly evolving over time. The purpose of geographic transport modeling can be interpreted as simplifying this complexity in a way that captures the essence of transport problems. Selecting an appropriate level of geographic analysis can help simplify this complexity, to capture the essence of a transport system without losing its most important features and variables (Hollander 2016).

Typically, models are designed to solve a particular problem. For this reason, this chapter is based around a policy scenario that asks: how to increase walking and cycling in the city of Bristol? Both policies aim to prevent traffic jams, reduce carbon emissions, and promote a healthier life style, all of which makes the city greener and thus more attractive and enjoyable to live in.

7.2 A case study of Bristol

The case study used for this chapter is located in Bristol, a city in the west of England, around 30 km east of the Welsh capital Cardiff. An overview of the region's transport network is illustrated in Figure 7.1, which shows a diversity of transport infrastructure, for cycling, public transport, and private motor vehicles.

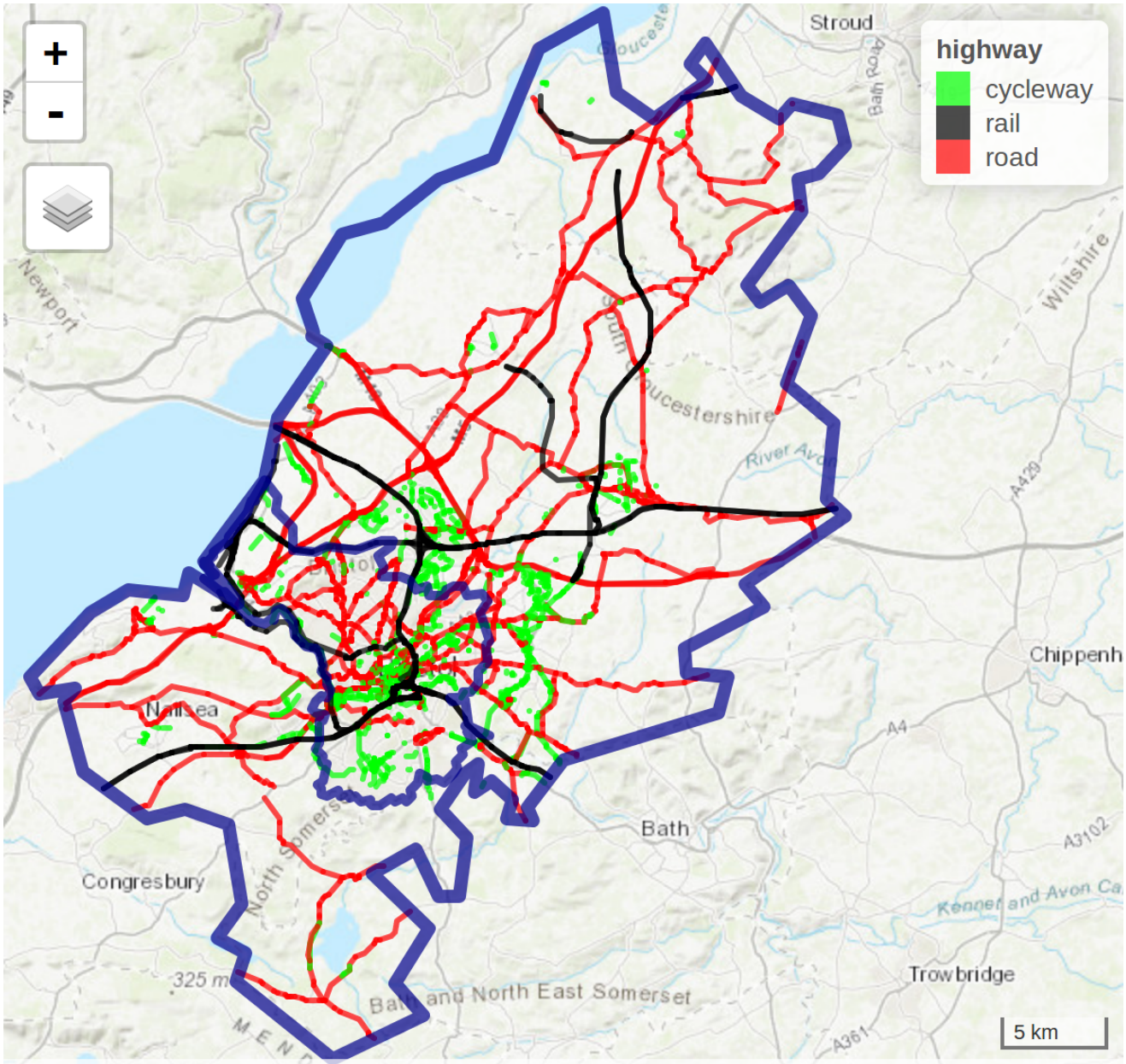

Figure 7.1: Bristol's transport network represented by colored lines for active (green), public (railways, black) and private motor (red) modes of travel. Blue border lines represent the inner city boundary and the larger Travel To Work Area (TTWA).

Bristol is the 10th largest city council in England, with a population of half a million people, although its travel catchment area is larger (see section 7.3). It has a vibrant economy with aerospace, media, financial service and tourism companies, alongside two major universities. Bristol shows a high average income per capita but also contains areas of severe deprivation (Bristol City Council 2015).

In terms of transport, Bristol is well served by rail and road links, and has a relatively high level of active travel. 19% of its citizens cycle and 88% walk at least once per month according to the Active People Survey (the national average is 15% and 81%, respectively). 8% of the population reported to cycle to work in the 2011 census, compared with only 3% nationwide.

Despite impressive walking and cycling statistics, the city has a major congestion problem. Part of the solution is to continue to increase the proportion of trips made by cycling. Cycling has a greater potential to replace car trips than walking because of the speed of this mode, around 3-4 times faster than walking

(with typical speeds of 15-20 km/h vs 4-6 km/h for walking). There is an ambitious plan to double the share of cycling by 2020.

In this policy context the aim of this chapter, beyond demonstrating how geocomputation with R can be used to support sustainable transport planning, is to provide evidence for decision-makers in Bristol to decide how best to increase the share of walking and cycling in particular in the city. This high-level aim will be met via the following objectives:

- Describe the geographical pattern of transport behavior in the city.
- Identify key public transport nodes and routes along which cycling to rail stations could be encouraged, as the first stage in multi-model trips.
- Analyze travel 'desire lines' in the city to identify those with greatest potential for modal shift.
- Building on the desire-line level analysis, identify which routes would most benefit from having dedicated cycleways and improved provision for pedestrians.

To get the wheels rolling on the practical aspects of this chapter, we begin by loading zonal data on travel patterns. These zone-level data are small but often vital for gaining a basic understanding of a settlement's overall transport system.

7.3 Transport zones

Although transport systems are primarily based on linear features and nodes — including pathways and stations — it often makes sense to start with areal data, to break continuous space into tangible units (Hollander 2016). Two zone types will typically be of particular interest: the study region and origin (typically residential areas) and destination (typically containing 'trip attractors' such as schools and shops) zones. Often the geographic units of destinations are the geographic units that comprise the origins, but a different zoning system, such as 'Workplace Zones', may be appropriate to represent the increased density of trip destinations in central areas (Office for National Statistics 2014).

The simplest way to define a study area is often the first matching boundary returned by OpenStreetMap, which can be obtained using **osmdata** as follows:

```
# requires the development version of osmdata
# devtools::install_github("ropensci/osmdata")
bristol_region = osmdata::getbb("Bristol", format_out = "sf_polygon")
```

The result is an `sf` object representing the bounds of the largest matching city region, either a rectangular polygon of the bounding box or a detailed polygonal boundary.[43] In this case the command returns a detailed polygon representing the official boundary of Bristol (see the inner blue boundary in Figure 7.1). There are a couple of issues associated with using OSM definitions of regions, however. First, the first OSM boundary returned by OSM may not be the official boundary used by local authorities. Second, even if OSM returns the official boundary, this may be inappropriate for transport research because they bear little relation to where people travel.

Travel to Work Areas (TTWAs) overcome the second issue by creating a zoning system analogous to hydrological watersheds. TTWAs were first defined as contiguous zones within which 75% of the population travels to work (Coombes, Green, and Openshaw 1986), and this is the definition used in this chapter. Because Bristol is a major employer attracting travel from surrounding towns, its TTWA is substantially larger than the city bounds (see Figure 7.1). The polygon representing this transport-orientated boundary is stored in the object `bristol_ttwa`, provided by the **spDataLarge** package loaded at the beginning of this chapter.

The origin and destination zones used in this chapter are the same: officially defined zones of intermediate geographic resolution (their official name is Middle layer Super Output Areas or MSOAs). Each house around 8,000 people. Such administrative zones can provide vital context to transport analysis, such as the type of people who might benefit most from particular interventions (e.g. Moreno-Monroy, Lovelace, and Ramos 2017).

The geographic resolution of these zones is important: small zones with high geographic resolution are usually preferable but their high number in large regions can have consequences for processing (especially for origin-destination analysis in which the number of possibilities increases as a non-linear function of the number of zones) (Hollander 2016).

> Another issue with small zones is related to anonymity rules. To make it impossible to infer the identity of individuals in zones, detailed socio-demographic variables are often only available at low geographic resolution. Breakdowns of travel mode by age and sex, for example, are available at the Local Authority level in the UK, but not at the much higher Output Area level, each of which contains around 100 households — see ons.gov.uk for further details.

The 102 zones used are illustrated in Figure 7.2. These are stored in the `bristol_zones` object from the **spDataLarge** package, which divides-up space into sensible parcels, with each zone housing a similar number of people. However at present the dataset contains no attribute data, other than the zone name and zone code:

```
names(bristol_zones)
#> [1] "geo_code" "name"     "geometry"
```

Chapter 3 demonstrated how to join attribute data onto geographic variables. This is a common task which usually involves joining a table containing official zonal statistics to a geographic object via a shared key variable, as described in section 3.2.3. In this case we use data provided by ons.gov.uk, which maintains datasets at various geographic levels across the UK.

Instead of accessing an area-level travel dataset directly, we will create zonal data by aggregating an object (`bristol_od` from **spDataLarge**) representing origin-destination (OD) data by zone of origin (exactly what this means and the nature of this OD dataset is described in more detail in section 7.4). This is demonstrated below:

```
zones_attr = group_by(bristol_od, o) %>%
  summarize_if(is.numeric, sum) %>%
  rename(geo_code = o)
```

What just happened? The code first read-in data to create `bristol_od`, a data frame object representing travel to work between zones from the UK's 2011 Census in which the first column is the ID of the zone of origin and the second column is the zone of destination (`bristol_od` is described more fully in the next section). Then the chained operation performed three main steps:

- Grouped the data by zone of origin (contained in the column `o`).
- Aggregated the variables in the `bristol_od` dataset *if* they were numeric, to find the total number of people living in each zone by mode of transport.[44]
- Renamed the grouping variable `o` so it matches the ID column `geo_code` in the `bristol_zones` object.

The resulting object `zones_attr` is a data frame with rows representing zones and an ID variable. We can verify that the IDs match those in the `zones` dataset using `%in%` operator as follows:

```
summary(zones_attr$geo_code %in% bristol_zones$geo_code)
#>    Mode    TRUE
#> logical     102
```

The results show that all 102 zones are present in the new object and that `zone_attr` is in a form that can be joined onto the zones.[45] This is done using the joining function `left_join()` (note that `inner_join()` would produce here the same result):

```
zones_joined = left_join(bristol_zones, zones_attr)
#> Joining, by = "geo_code"
sum(zones_joined$all)
#> [1] 238805
names(zones_joined)
#> [1] "geo_code"   "name"       "all"        "bicycle"    "foot"
#> [6] "car_driver" "train"      "geometry"
```

The result is `zones_joined`, which contains new columns representing the total number of trips originating in each zone in the study area (almost 1/4 of a million) and their mode of travel (by bicycle, foot, car and train). The geographic distribution of trip origins is illustrated in the left-hand map in Figure 7.2. This shows that most zones have between 0 and 4,000 trips originating from them in the study area. More trips are made by people living near the center of Bristol and fewer on the outskirts. Why is this? Remember that we are only dealing with trips within the study region: low trip numbers in the outskirts of the region can be explained by the fact that many people in these peripheral zones will travel to other regions outside of the study area. Trips outside the study region can be included in regional model by a special destination ID covering any trips that go to a zone not represented in the model (Hollander 2016). The data in `bristol_od`, however, simply ignores such trips: it is an 'intra-zonal' model.

In the same way that OD datasets can be aggregated to the zone of origin, they can also be aggregated to provide information about destination zones. People tend to gravitate towards central places. This explains why the spatial distribution represented in the right panel in Figure 7.2 is relatively uneven, with the most common destination zones concentrated in Bristol city center. The result is `zones_od`, which contains a new column reporting the number of trip destinations by any mode, is created as follows:

```
zones_od = group_by(bristol_od, d) %>%
  summarize_if(is.numeric, sum) %>%
  dplyr::select(geo_code = d, all_dest = all) %>%
  inner_join(zones_joined, ., by = "geo_code")
```

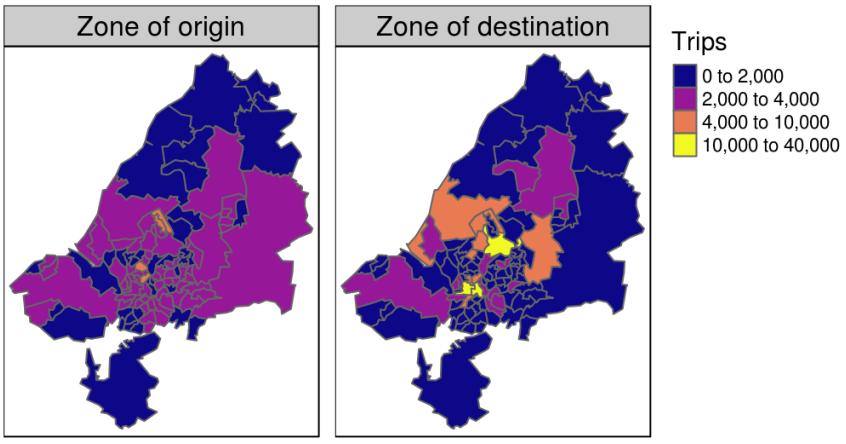

Figure 7.2: Number of trips (commuters) living and working in the region. The left map shows zone of origin of commute trips; the right map shows zone of destination (generated by the script 07-zones.R).

7.4 Desire lines

Unlike zones, which represent trip origins and destinations, desire lines connect the centroid of the origin and the destination zone, and thereby represent where people *desire* to go between zones. If it were not for obstacles such as buildings and windy roads, people would travel in a 'bee-line' from one place to the next, explaining why desire lines are straight (we will see how to convert desire lines into routes in the next section).

We have already loaded data representing desire lines in the dataset `bristol_od`. This origin-destination (OD) data frame object represents the number of people traveling between the zone represented in `o` and `d`, as illustrated in Table 7.1. To arrange the OD data by all trips and then filter-out only the top 5, type (please refer to Chapter 3 for a detailed description of non-spatial attribute operations):

```
od_top5 = arrange(bristol_od, desc(all)) %>%
  top_n(5, wt = all)
```

Table 7.1: Sample of the origin-destination data stored in the data frame object `bristol_od`. These represent the top 5 most common desire lines between zones in the study area.

o	d	all	bicycle	foot	car_driver	train
E02003043	E02003043	1493	66	1296	64	8
E02003047	E02003043	1300	287	751	148	8
E02003031	E02003043	1221	305	600	176	7
E02003037	E02003043	1186	88	908	110	3
E02003034	E02003043	1177	281	711	100	7

The resulting table provides a snapshot of Bristolian travel patterns in terms of commuting (travel to work). It demonstrates that walking is the most popular mode of transport among the top 5 origin-destination pairs, that zone `E02003043` is a popular destination (Bristol city center, the destination of all the top 5 OD pairs), and that the *intrazonal* trips, from one part of zone `E02003043` to another (first row of table 7.1), constitute the most traveled OD pair in the dataset. But from a policy perspective 7.1 is of limited use: aside from the fact that it contains only a tiny portion of the 2,910 OD pairs, it tells us little about *where* policy measures are needed. What is needed is a way to plot this origin-destination data on the map.

The solution is to convert the non-geographic `bristol_od` dataset into geographical desire lines that can be plotted on a map. The geographic representation of the *interzonal* OD pairs (in which the destination is different from the origin) presented in Table 7.1 are displayed as straight black lines in 7.3. These are clearly more useful from a policy perspective. The conversion from `data.frame` to `sf` class is done by the **stplanr** function `od2line()`, which matches the IDs in the first two columns of the `bristol_od` object to the `zone_code` ID column in the geographic `zones_od` object.[46]

```
od_intra = filter(bristol_od, o == d)
od_inter = filter(bristol_od, o != d)
desire_lines = od2line(od_inter, zones_od)
#> Warning in st_centroid.sfc(st_geometry(x), of_largest_polygon =
#> of_largest_polygon): st_centroid does not give correct centroids for
#> longitude/latitude data
```

The first two lines of the preceding code chunk split the `bristol_od` dataset into two mutually exclusive objects, `od_intra` (which only contains OD pairs representing intrazone trips) and `od_inter` (which represents interzonal travel). The third line generates a geographic object `desire_lines` (of class `sf`) that allows a subsequent geographic visualization of interzone trips. An illustration of the results is presented in Figure 7.3 (we will cover the visualization methods that produced this plot in Chapter 9). The map shows that the city center dominates transport patterns in the region, suggesting policies should be prioritized there, although a number of peripheral sub-centers can also be seen. Next it would be interesting to have a look at the distribution of interzonal modes, e.g. between which zones is cycling the least or the most common means of transport.

```
plot(desire_lines$geometry, lwd = desire_lines$all / 500)
```

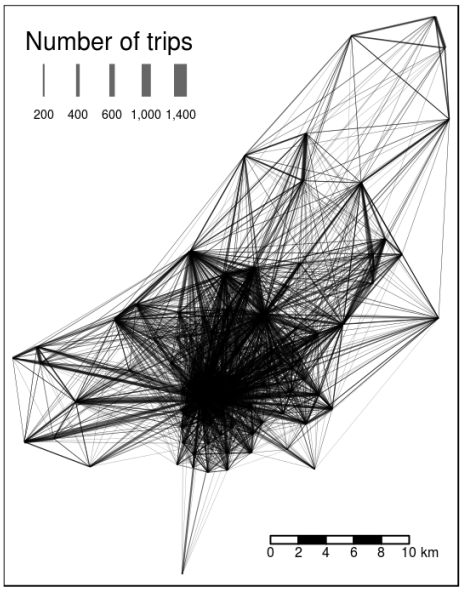

Figure 7.3: Desire lines representing trip patterns in the Bristol Travel to Work Area. The four black lines represent the object the top 5 desire lines illustrated in Table 7.1.

7.5 Routes

From a geographical perspective routes are desire lines that are no longer straight: the origin and destination points are the same, but the pathway to get from A to B is more complex. Desire lines contain only two vertices (their beginning and end points) but routes can contain hundreds of vertices if they cover a large distance or represent travel patterns on an intricate road network (routes on simple grid-based road networks require relatively few vertices).

Routes are generated from desire lines — or more commonly origin-destination pairs — using routing algorithms. Routing algorithms can either run locally or remotely. Routing is computationally intensive and might be slow on a local machine (depending on the hardware). In any case, local routing requires the route network to be stored on a local computer (we will see how in section 7.7). By contrast, remote routing services use a web API to receive queries about origin and destination points, run the routing on the route network on a powerful server and return just the results, the routes, to the user, the so-called client. There are many advantages of using remote routing services, including the fact that they can be up-to-date, have global coverage, and run on a set-up that was designed for the job, explaining this section's focus on online routing services.

Before proceeding, there are couple of important disadvantages of online routing services to consider: they can be slow (because they rely on data transfer over the internet) and expensive. The Google routing API, for example, has a limit of 2500 free queries per day.[47]

A popular and free routing service is the Open Source Routing Machine (OSRM).[48] Instead of routing *all* desire lines generated in the previous section, which would be time and memory-consuming, we will focus on the desire lines of policy interest. The benefits of cycling trips are greatest when they replace car trips. Clearly not all car trips can realistically be replaced by cycling, but 5 km Euclidean distance (or around 6-

8 km of route distance) is easily accessible within 30 minutes for most people. Based on this reasoning we will only route desire lines along which a high (300+) number of car trips take place that are up to 5 km in distance. This routing is done by the **stplanr** function `line2route()` which takes straight lines in `Spatial` or `sf` objects, and returns 'bendy' lines representing routes on the transport network in the same class as the input.

```
desire_lines$distance = as.numeric(st_length(desire_lines))
desire_carshort = dplyr::filter(desire_lines, car_driver > 300 & distance < 5000)


route_carshort = line2route(desire_carshort, route_fun = route_osrm)
```

`st_length()` determines the length of a linestring, and falls into the distance relations category (see also section 4.2.6). Subsequently, we apply a simple attribute filter operation (see section 3.2.1) before letting the OSRM service do the routing on a remote server. Note that the routing only works with a working internet connection.

We could keep the new `route_carshort` object separate from the straight line representation of the same trip in `desire_carshort` but, from a data management perspective, it makes more sense to combine them: they represent the same trip. The new route dataset contains `distance` (referring to route distance this time) and `duration` fields (in seconds) which could be useful. However, for the purposes of this chapter we are only interested in the geometry, from which route distance can be calculated. The following command makes use of the ability of simple features objects to contain multiple geographic columns:

```
desire_carshort$geom_car = route_carshort$geometry
```

This allows plotting the desire lines along which many short car journeys take place alongside likely routes traveled by cars, with the width of the routes proportional to the number of car journeys that could potentially be replaced. The code below results in Figure 7.4, demonstrating along which routes people are driving short distances[49]:

```
plot(desire_carshort$geometry)
plot(desire_carshort$geom_car, col = "red", add = TRUE)
plot(st_centroid(zones_od)$geometry, add = TRUE)
```

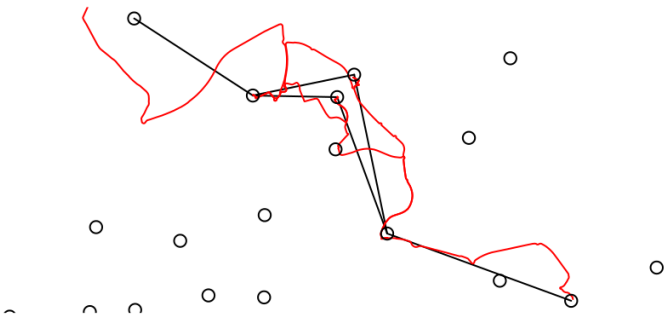

Figure 7.4: Routes along which many (300+) short (<5km Euclidean distance) car journeys are made (red) overlaying desire lines representing the same trips (black) and zone centroids (dots).

The results show that the short desire lines along which most people travel by car are geographically clustered. Plotting the results on an interactive map, for example, with `mapview::mapview(desire_carshort)`, reveals that these car trips take place in and around Bradley Stoke. According to Wikipedia, Bradley Stoke is "Europe's largest new town built with private investment", which may help understand its high level of car dependency due to limited public transport provision. The excessive number of short car journeys in this area can also be understood in terms of the car-orientated transport infrastructure surrounding this northern 'edge city' which includes "major transport nodes such as junctions on both the M4 and M5 motorways" (Tallon 2007).

There are many benefits of converting travel desire lines into likely routes of travel from a policy perspective, primary among them the ability to understand what it is about the surrounding environment that makes people travel by a particular mode. We discuss future directions of research building on the routes in section 7.9. For the purposes of this case study, suffice to say that the roads along which these short car journeys travel should be prioritized for investigation to understand how they can be made more conducive to sustainable transport modes. One option would be to add new public transport nodes to the network. Such nodes are described in the next section.

7.6 Nodes

Nodes in geographic transport data are zero dimensional features (points) among the predominantly one dimensional features (lines) that comprise the network. There are two types of transport nodes:

1. Nodes not directly on the network such as zone centroids — covered in the next section — or individual origins and destinations such as houses and workplaces.
2. Nodes that are a part of transport networks, representing individual pathways, intersections between pathways (junctions) and points for entering or exiting a transport network such as bus stops and train stations.

From a mathematical perspective transport networks can be represented as graphs, in which each segment is connected (via edges representing geographic lines) to one or more other edges in the network. The first type of node can be connected to the network with “centroid connectors”, new route segments joining nodes outside the network with one or more nearby nodes on the network (Hollander 2016). The location of these connectors should be chosen carefully because they can lead to over-estimates of traffic volumes in their immediate surroundings (Jafari et al. 2015). The second type of nodes are nodes on the graph, each of which is connected by one or more straight ‘edges’ that represent individual segments on the network. We will see how transport networks can be represented as mathematical graphs in section 7.7.

Public transport stops are particularly important nodes that can be represented as either type of node: a bus stop that is part of a road, or a large rail station that is represented by its pedestrian entry point hundreds of meters from railway tracks. We will use railway stations to illustrate public transport nodes, in relation to the research question of increasing cycling in Bristol. These stations are provided by **spDataLarge** in `bristol_stations`.

A common barrier preventing people from switching away from cars for commuting to work is that the distance from home to work is too far to walk or cycle. Public transport can reduce this barrier by providing a fast and high-volume option for common routes into cities. From an active travel perspective public transport ‘legs’ of longer journeys divide trips into three:

- The origin leg, typically from residential areas to public transport stations.
- The public transport leg, which typically goes from the station nearest a trip’s origin to the station nearest its destination.
- The destination leg, from the station of alighting to the destination.

Building on the analysis conducted in section 7.4, public transport nodes can be used to construct three-part desire lines for trips that can be taken by bus and (the mode used in this example) rail. The first stage is to identify the desire lines with most public transport travel, which in our case is easy because our previously created dataset `desire_lines` already contains a variable describing the number of trips by train (the public transport potential could also be estimated using public transport routing services such as OpenTripPlanner). To make our approach easy to follow we will select just the top three desire lines in terms of rails use:

```
desire_rail = top_n(desire_lines, n = 3, wt = train)
```

The challenge now is to ‘break-up’ each of these lines into three pieces, representing travel via public transport nodes. This can be done by converting a desire line into a multiline object consisting of three line geometries representing origin, public transport and destination legs of the trip. The first stage is to create matrices of coordinates that will subsequently be used to create matrices representing each leg:

```
mat_orig = as.matrix(line2df(desire_rail)[c("fx", "fy")])
mat_dest = as.matrix(line2df(desire_rail)[c("tx", "ty")])
mat_rail = st_coordinates(bristol_stations)
```

The outputs are three matrices representing the starting points of the trips, their destinations and possible intermediary points at public transport nodes (named `orig`, `dest` and `rail` respectively). But how to identify *which* intermediary points to use for each desire line? The `knn()` function from the **nabor** package (which is used internally by **stplanr** so it should already be installed) solves this problem by finding *k nearest neighbors* between two sets of coordinates. By setting the `k` parameter, one can define how many nearest neighbors should be returned. Of course, `k` cannot exceed the number of observations in the input (here: `mat_rail`). We are interested in just one nearest neighbor, namely, the closest railway station. :

```
knn_orig = nabor::knn(mat_rail, query = mat_orig, k = 1)$nn.idx
knn_dest = nabor::knn(mat_rail, query = mat_dest, k = 1)$nn.idx
```

This results not in matrices of coordinates, but row indices that can subsequently be used to subset the `mat_rail`. It is worth taking a look at the results to ensure that the process has worked properly, and to explain what has happened:

```
as.numeric(knn_orig)
#> [1] 27  3  2
as.numeric(knn_dest)
#> [1] 29 29 29
```

The output demonstrates that each object contains three whole numbers (the number of rows in `desire_rail`) representing the rail station closest to the origin and destination of each desire line. Note that while each 'origin station' is different, the destination (station `30`) is the same for all desire lines. This is to be expected because rail travel in cities tends to converge on a single large station (in this case Bristol Temple Meads). The indices can now be used to create matrices representing the rail station of origin and destination:

```
mat_rail_o = mat_rail[knn_orig, ]
mat_rail_d = mat_rail[knn_dest, ]
```

The final stage is to convert these matrices into meaningful geographic objects, in this case simple feature 'multilinestrings' that capture the fact that each stage is a separate line, but part of the same overall trip:

```
mats2line = function(mat1, mat2) {
  lapply(1:nrow(mat1), function(i) {
    rbind(mat1[i, ], mat2[i, ]) %>%
    st_linestring()
  }) %>% st_sfc()
}
desire_rail$leg_orig = mats2line(mat_orig, mat_rail_o)
desire_rail$leg_rail = mats2line(mat_rail_o, mat_rail_d)
desire_rail$leg_dest = mats2line(mat_rail_d, mat_dest)
```

Now we are in a position to visualize the results, in Figure 7.5: the three initial `desire_rail` lines now have three additional geometry list-columns representing travel from home to the origin station, from there to the destination, and finally from the destination station to the destination. In this case the destination leg is very short (walking distance) but the origin legs may be sufficiently far to justify investment in cycling infrastructure to encourage people to cycle to the stations on the outward leg of peoples' journey to work in the residential areas surrounding the three origin stations in Figure 7.5.

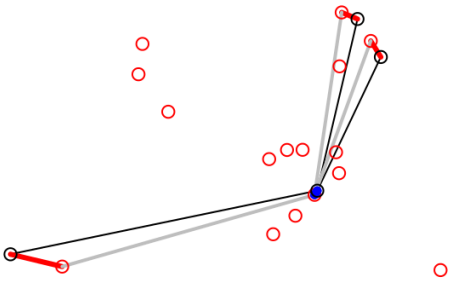

Figure 7.5: Station nodes (red dots) used as intermediary points that convert straight desire lines with high rail usage (black) into three legs: to the origin station (red) via public transport (grey) and to the destination (a very short blue line).

7.7 Route networks

The data used in this section was downloaded using **osmdata**. To avoid having to request the data from OSM repeatedly, we will use the `bristol_ways` object, which contains point and line data for the case study area (see `?bristol_ways`):

```
summary(bristol_ways)
#>      highway        maxspeed         ref                geometry
#>  cycleway:1262   30 mph : 834   A38    : 202   LINESTRING   :4619
#>  rail    : 813   20 mph : 456   M5     : 138   epsg:4326    :   0
#>  road    :2544   40 mph : 332   A432   : 131   +proj=long...:   0
#>                  70 mph : 323   A4018  : 120
#>                  50 mph : 137   A420   : 114
#>                  (Other): 470   (Other):1697
#>                  NA's   :2067   NA's   :2217
```

The above code chunk loaded a simple feature object representing around 3,000 segments on the transport network. This an easily manageable dataset size (transport datasets can be large but it's best to start small).

As mentioned, route networks can usefully be represented as mathematical graphs, with nodes on the network connected by edges. A number of R packages have been developed for dealing with such graphs, notably **igraph**. One can manually convert a route network into an `igraph` object, but that process risks loosing the geographic attributes. To overcome this issue `SpatialLinesNetwork()` was developed in the **stplanr** package to represent route networks simultaneously as graphs *and* a set of geographic lines. This function is demonstrated below using a subset of the `bristol_ways` object used in previous sections.

```
ways_freeway = bristol_ways %>% filter(maxspeed == "70 mph")
ways_sln = SpatialLinesNetwork(ways_freeway)
slotNames(ways_sln)
#> [1] "sl"          "g"           "nb"          "weightfield"
weightfield(ways_sln)
#> [1] "length"
class(ways_sln@g)
#> [1] "igraph"
```

The output of the previous code chunk shows that `ways_sln` is a composite object with various 'slots'. These include: the spatial component of the network (named `sl`), the graph component (`g`) and the 'weightfield', the edge variable used for shortest path calculation (by default segment distance). `ways_sln` is of class `sfNetwork`, defined by the S4 class system. This means that each component can be accessed using the `@` operator, which is used below to extract its graph component and process it using the **igraph** package, before plotting the results in geographic space. In the example below the 'edge betweeness', meaning the number of shortest paths passing through each edge, is calculated (see `?igraph::betweenness` for further details). The results demonstrate that each graph edge represents a segment: the segments near the center of the road network have the greatest betweeness scores.

```
g = ways_sln@g
e = igraph::edge_betweenness(ways_sln@g)
plot(ways_sln@sl$geometry, lwd = e / 500)
```

Figure 7.6: Illustration of a small route network, with segment thickness proportional to its betweeness, generated using the **igraph** package and described in the text.

One can also find the shortest route between origins and destinations using this graph representation of the route network. This is can be done using the `sum_network_routes()` function from **stplanr**, which uses local route network data instead of the online routing service described in section 7.5. This finds the shortest path between arbitrary nodes 1 and 20, on the network — 'shortest' with reference to the `weightfield` slot of `ways_sln` (route distance by default).[50] The result is a spatial linestring object that can be plotted using **sf** plotting methods (result not shown — readers are encouraged to plot the result locally):

```
path = sum_network_routes(ways_sln, 1, 20, "length")


plot(path$geometry, col = "red", lwd = 10)
plot(ways_sln@sl$geometry, add = TRUE)
```

7.8 Prioritizing new infrastructure

This chapter's final practical section demonstrates the policy-relevance of geocomputation for transport applications by identifying locations where new transport infrastructure may be needed. The next chapter (8) demonstrates another application: prioritising the location of new bike shops. Bike shops may benefit from new cycling infrastructure, demonstrating an important feature of transport systems: they are closely linked to broader social, economic and land-use patterns. This section ties-together the various strands that explored some geographic features of Bristol's transport system, covered in sections 7.3 to 7.7.

Clearly the types of analysis presented here would need to be extended and complimented by other methods to be used in real-world applications, as discussed in section 7.9. However each stage could be useful on its own, and feed-into wider analyses. To summarize, these were: identifying short but car-dependent commuting routes (generated from desire lines)in section 7.5; creating desire lines representing trips to rail stations in section 7.6; and analysis of transport systems at the route network using graph theory in section 7.7.

The final code chunk of this chapter combines these strands of analysis. It adds the car-dependent routes in `route_carshort` with a newly-created object, `route_rail` and creates a new column representing the amount of travel along the centroid-to-centroid desire lines they represent:

```
route_rail = st_set_geometry(desire_rail, "leg_orig") %>%
  line2route(route_fun = route_osrm) %>%
  st_set_crs(4326)


route_cycleway = rbind(route_rail, route_carshort)
route_cycleway$all = c(desire_rail$all, desire_carshort$all)
```

The result of this code, visualized in Figure 7.7, identifies routes of interest in terms of car dependency and key opportunities to invest in public transport-cycling integration. Although other routes between zones are likely to be used — in reality people do not travel to zone centroids or always use the shortest route algorithm for a particular mode — the results demonstrate routes along which cycle paths could be prioritized.

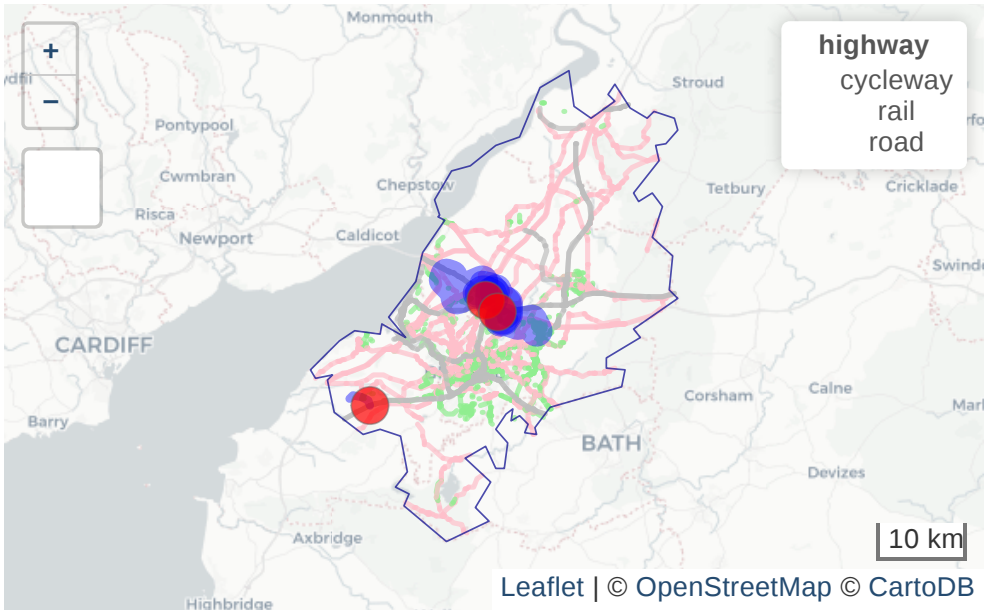

Figure 7.7: Potential routes along which to prioritise cycle infrastructure in Bristol, based on access key rail stations (red dots) and routes with many short car journeys (north of Bristol surrounding Stoke Bradley). Line thickness is proportional to number of trips.

The results may look more attractive in an interactive map, but what do they mean? The routes highlighted in Figure 7.7 suggest that transport systems are intimately linked to the wider economic and social context. The example of Stoke Bradley is a case in point: its location, lack of public transport services and active travel infrastructure help explain why it is so highly car-dependent. The wider point is that car dependency has a spatial distribution which has implications for sustainable transport policies (Hickman, Ashiru, and Banister 2011).

7.9 Future directions of travel

This chapter provides a taster of the possibilities of using geocomputation for transport research. It has explored some key geographic elements that make-up a city's transport system using open data and reproducible code. The results could help plan where investment is needed.

Transport systems operate at multiple interacting levels, meaning that geocomputational methods have great potential to generate insights into how they work. There is much more that could be done in this area: it would be possible to build on the foundations presented in this chapter in many directions. Transport is the fastest growing source of greenhouse gas emissions in many countries, and is set to become "the largest GHG emitting sector, especially in developed countries" (see EURACTIV.com). Because of the highly unequal distribution of transport-related emissions across society, and the fact that transport (unlike food and heating) is not essential for well-being, there is great potential for the sector to

rapidly decarbonize through demand reduction, electrification of the vehicle fleet and the uptake of active travel modes such as walking and cycling. Further exploration of such 'transport futures' at the local level represents promising direction of travel for transport-related geocomputational research.

Methodologically the foundations presented in this chapter could be extended by including more variables in the analysis. Characteristics of the route such as speed limits, busyness and the provision of protected cycling and walking paths could be linked to 'mode-split' (the proportion of trips made by different modes of transport). By aggregating OpenStreetMap data using buffers and spatial data methods presented in Chapters 3 and 4, for example, it would be possible to detect the presence of green space in close proximity to transport routes. Using R's statistical modelling capabilities this could then be used to predict current and future levels of cycling, for example.

This type of analysis underlies the Propensity to Cycle Tool (PCT), a publicly accessible (see www.pct.bike) mapping tool developed in R that is being used to prioritize investment in cycling across England (Lovelace et al. 2017). Similar tools could be used to encourage evidence-based transport policies related to other topics such as air pollution and public transport access around the world.

7.10 Exercises

1. What is the total distance of cycleways that would be constructed if all the routes presented in Figure 7.7 were to be constructed?
    - Bonus: find two ways of arriving at the same answer.
2. What proportion of trips represented in the `desire_lines` are accounted for in the `route_cycleway` object?
    - Bonus: what proportion of trips cross the proposed routes?
    - Advanced: write code that would increase this proportion.
3. The analysis presented in this chapter is designed for teaching how geocomputation methods can be applied to transport research. If you were to do this 'for real' for local government or a transport consultancy what top 3 things would you do differently?
4. Clearly the routes identified in Figure 7.7 only provide part of the picture. How would you extend the analysis to incorporate more trips that could potentially be cycled?
5. Imagine that you want to extend the scenario by creating key *areas* (not routes) for investment in place-based cycling policies such as car-free zones, cycle parking points and reduced car parking strategy. How could raster data assist with this work?
    - Bonus: develop a raster layer that divides the Bristol region into 100 cells (10 by 10) and provide a metric related to transport policy, such as number of people trips that pass through each cell by walking or the average speed limit of roads (from the `bristol_ways` dataset).

References

Miller, Harvey J. 2004. "Tobler's First Law and Spatial Analysis." *Annals of the Association of American Geographers* 94 (2).

Rodrigue, Jean-Paul, Claude Comtois, and Brian Slack. 2013. *The Geography of Transport Systems*. 3 edition. London ; New York: Routledge.

Hollander, Yaron. 2016. *Transport Modelling for a Complete Beginner*. CTthink!

Horni, Andreas, Kai Nagel, and Kay W. Axhausen. 2016. *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press. https://www.ubiquitypress.com/site/books/10.5334/baw/.

Thiele, J. 2014. “R Marries NetLogo: Introduction to the RNetLogo Package.” *Journal of Statistical Software* 58 (2): 1–41. http://www.jstatsoft.org/v58/i02/paper.

Lovelace, Robin, and Morgane Dumont. 2016. *Spatial Microsimulation with R*. CRC Press. https://spatial-microsim-book.robinlovelace.net/.

Bristol City Council. 2015. “Deprivation in Bristol 2015.” Bristol City Council. https://www.bristol.gov.uk/statistics-census-information/deprivation.

Office for National Statistics. 2014. “Workplace Zones: A New Geography for Workplace Statistics - Datasets.” https://data.gov.uk/dataset/workplace-zones-a-new-geography-for-workplace-statistics3.

Coombes, M. G., A. E. Green, and S. Openshaw. 1986. “An Efficient Algorithm to Generate Official Statistical Reporting Areas: The Case of the 1984 Travel-to-Work Areas Revision in Britain.” *The Journal of the Operational Research Society* 37 (10): 943. doi:10.2307/2582282.

Moreno-Monroy, Ana I., Robin Lovelace, and Frederico R. Ramos. 2017. “Public Transport and School Location Impacts on Educational Inequalities: Insights from São Paulo.” *Journal of Transport Geography*, September. doi:10.1016/j.jtrangeo.2017.08.012.

Tallon, Andrew R. 2007. “Bristol.” *Cities* 24 (1): 74–88. doi:10.1016/j.cities.2006.10.004.

Jafari, Ehsan, Mason D. Gemar, Natalia Ruiz Juri, and Jennifer Duthie. 2015. “Investigation of Centroid Connector Placement for Advanced Traffic Assignment Models with Added Network Detail.” *Transportation Research Record: Journal of the Transportation Research Board* 2498 (June): 19–26. doi:10.3141/2498-03.

Hickman, Robin, Olu Ashiru, and David Banister. 2011. “Transitions to Low Carbon Transport Futures: Strategic Conversations from London and Delhi.” *Journal of Transport Geography* 19 (6): 1553–62.

Lovelace, Robin, Anna Goodman, Rachel Aldred, Nikolai Berkoff, Ali Abbas, and James Woodcock. 2017. “The Propensity to Cycle Tool: An Open Source Online System for Sustainable Transport Planning.” *Journal of Transport and Land Use* 10 (1). doi:10.5198/jtlu.2016.862.

43. In cases where the first match does not provide the right name, the country or region should be specified, for example `Bristol Tennessee` for a Bristol located in America.↩

44. the `_if` affix requires a `TRUE` / `FALSE` question to be asked of the variables, in this case ‘is it numeric?’ and only variables returning true are summarized.↩

45. It would also be important to check that IDs match in the opposite direction on real data. This could be done by reversing the order of the ID’s in the commend — `summary(bristol_zones$geo_code %in% zones_attr$geo_code)` — or by using `setdiff()` as follows: `setdiff(bristol_zones$geo_code, zones_attr$geo_code)` .↩

46. Note that the operation emits a warning: this is `od2line()` works by allocating the start and end points of each origin-destination pair to the *centroid* of its zone of origin and destination. This represents a straight line between the centroid of zone `E02003047` and the centroid of `E02003043` for the second origin-destination pair represented in Table 7.1, for example. For real-world use one would use centroid values generated from projected data or, preferably, use *population-weighted* centroids (Lovelace et al. 2017).↩

47. `ggmap::route()` lets you use Google's routing API through R and the **dodgr** package allows for rapid routing on locally stored route network data.↩

48. Another popular OSM routing API is https://openrouteservice.org/.↩

49. In this plot the origins of the red routes and black desire lines are not identical. This is because zone centroids rarely lie on the route network: instead the route originate from the transport network node nearest the centroid. Note also that routes are assumed to originate in the zone centroids, a simplifying assumption which is used in transport models to reduce the computational resources needed to calculate the shortest path between all combinations of possible origins and destinations (Hollander 2016).↩

50. To select nodes by geographic location the **stplanr** function `find_network_nodes()` can be used.↩