GEOG 8292 Workshop 02 Network Accessibility – GTFS

In this workshop, we will introduce the General Transit Feed Specification (GTFS), which is a data specification to store and transfer transit-related data among public transit agencies. We will use the Metro Transit schedule data for the Twin Cities metropolitan area as an example to get to know the data structure and specifications and conduct some basic network analysis.

❖ GTFS Data Model

The two figures below illustrate the data model adopted by the GTFS files. Figure 1 shows spatial, non-spatial, and virtual entities and relationships between them (by Martin Davis, a blog posted at http://lin-ear-th-inking.blogspot.com.au/2011/09/data-model-diagrams-for-gtfs.html). And Figure 2 provides less formal but more informative guide to the GTFS (by Ahsan Ijaz, a blog posted at: https://ahsanijaz.github.io/2015-05-03-StopIner/).

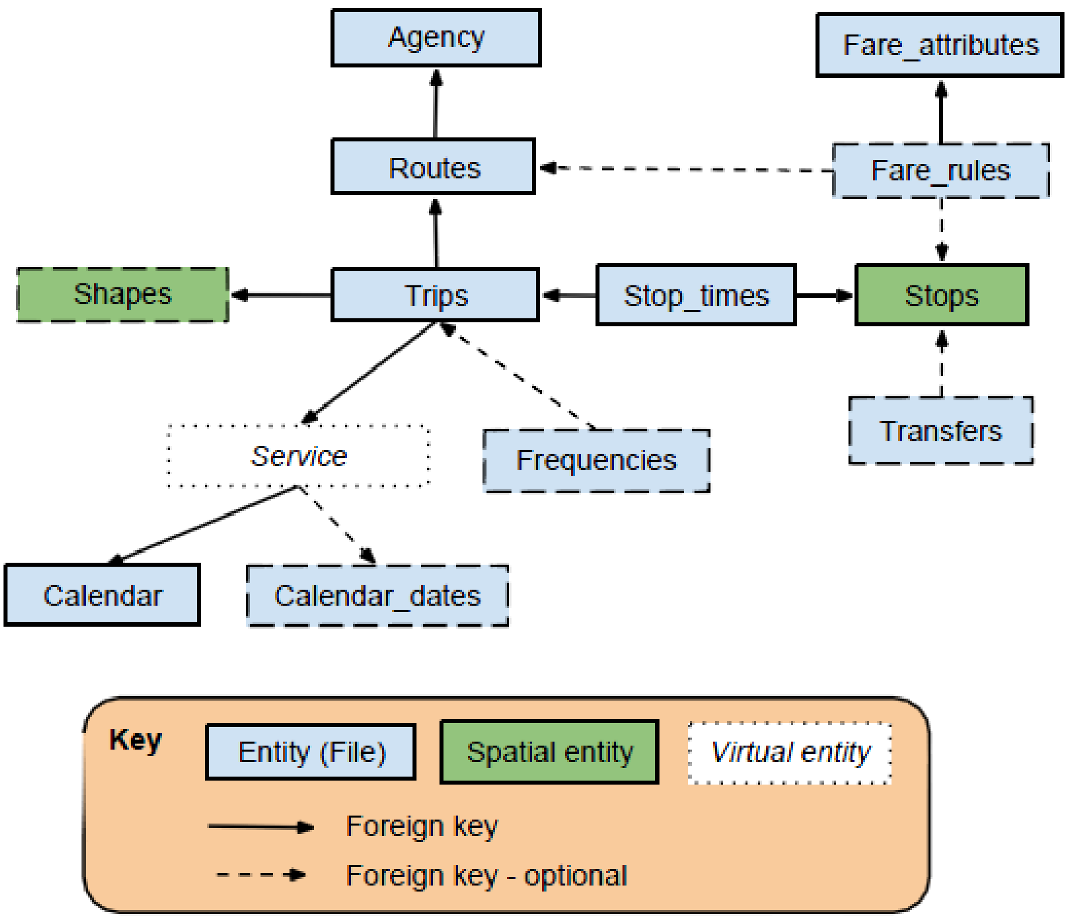


Figure 1 Data Model with Entities and Relationship (Davis, 2011)

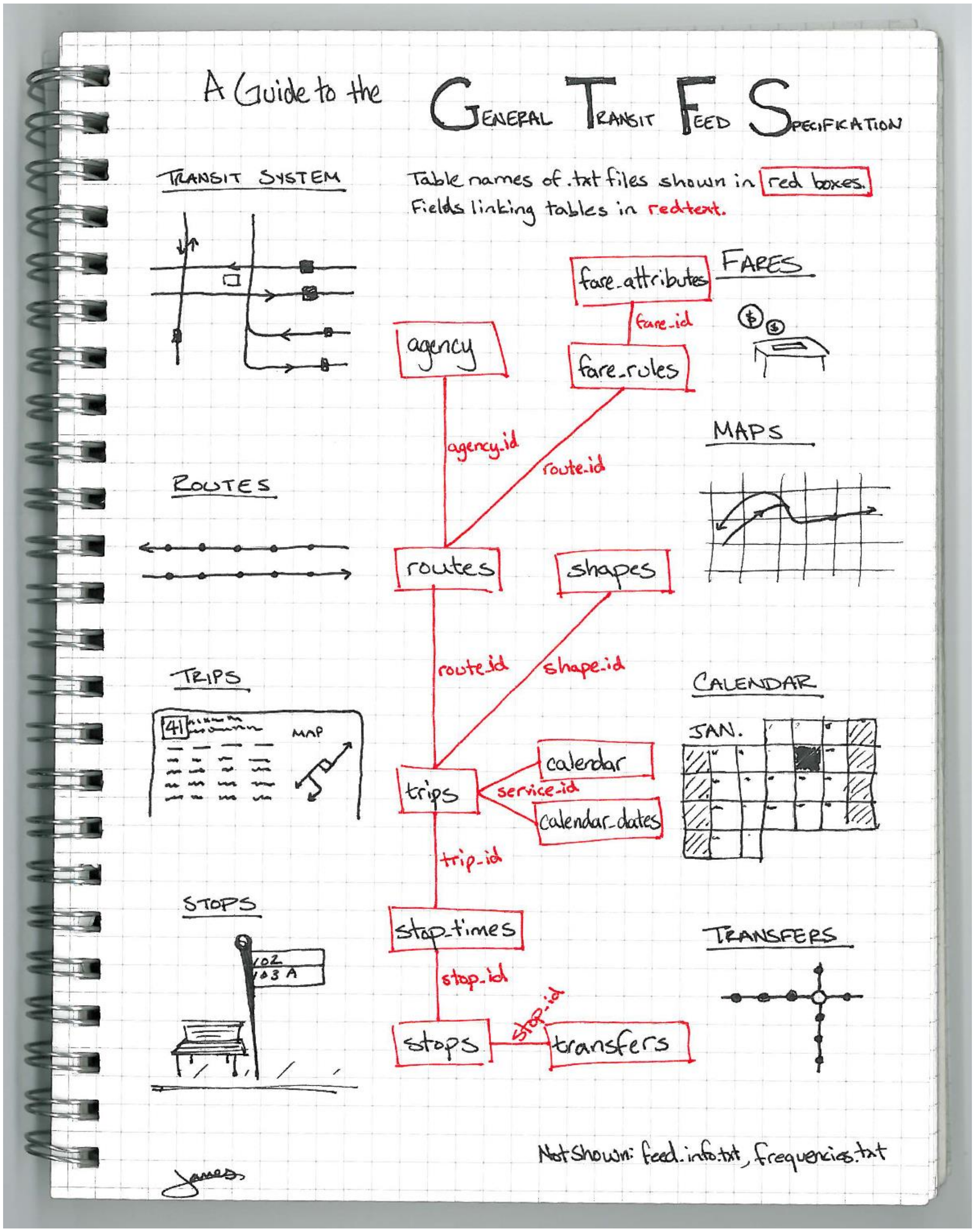


Figure 2. A Graphical Guide to GTFS (Ljaz, 2015)

The GTFS was originally developed under the name *google transit feed specification*, and it has later been applied as a general data specification for STATIC public transit schedules. The reference with detailed specifications is published at https://developers.google.com/transit/gtfs/reference?csw=1, which allows us to create our own feeds. This reference provides all the required and optional files, field types and the field definitions (with types) for each file.

There are five files required for GTFS and two conditional required files:

- **Agency**: transit agencies; an agency provide service along routes
- **Routes**: transit routes; a route is a set of trips that are displayed as a single service
- **Trips**: trips for each route; a trip is a sequence of two or more stops that occur during a specific time period; optionally contain **spatial info (linked to the file Shapes)**
- **Stops**: transit stops; a stop is a location where vehicles pick up or drop off passengers; **contain spatial info (stop_lat, stop_lon)**
- **Stop_times:** transit stops; each stop_time records the scheduled times that a vehicle arrives at and departs at a stop for a given trip
- **Calendar** (conditional required): service dates, specified using a weekly schedule with start and end dates. This file is required if the file *calendar_dates* is omitted.
- **Calendar_dates** (conditional required): exceptions for the services defined in the file *calendar*. This file is required if the file *calendar* is omitted.

The fare information and rules for a transit agency's route is not required. The files are often provided as comma-delimited (CSV) files, with each field values encapsulated by quotation marks. Files must be encoded in UTF-8 to support all Unicode characters and zipped together while publishing and sharing as one dataset (a folder with a set of csv files).

❖ Data and Tool

1. Data

This workshop uses Metro Transit schedule data that includes bus and LRT schedules for all public transit companies in the Twin Cities metropolitan area (except Minnesota Valley Transit Authority). The data can be accessed at the Minnesota Geospatial Commons: https://gisdata.mn.gov/dataset/us-mn-state-metc-trans-transit-schedule-google-fd. This workshop will use the current schedules for, which can be accessed at https://svc.metrotransit.org/mtgtfs/gtfs.zip (accessed on Nov. 15th, 2020).

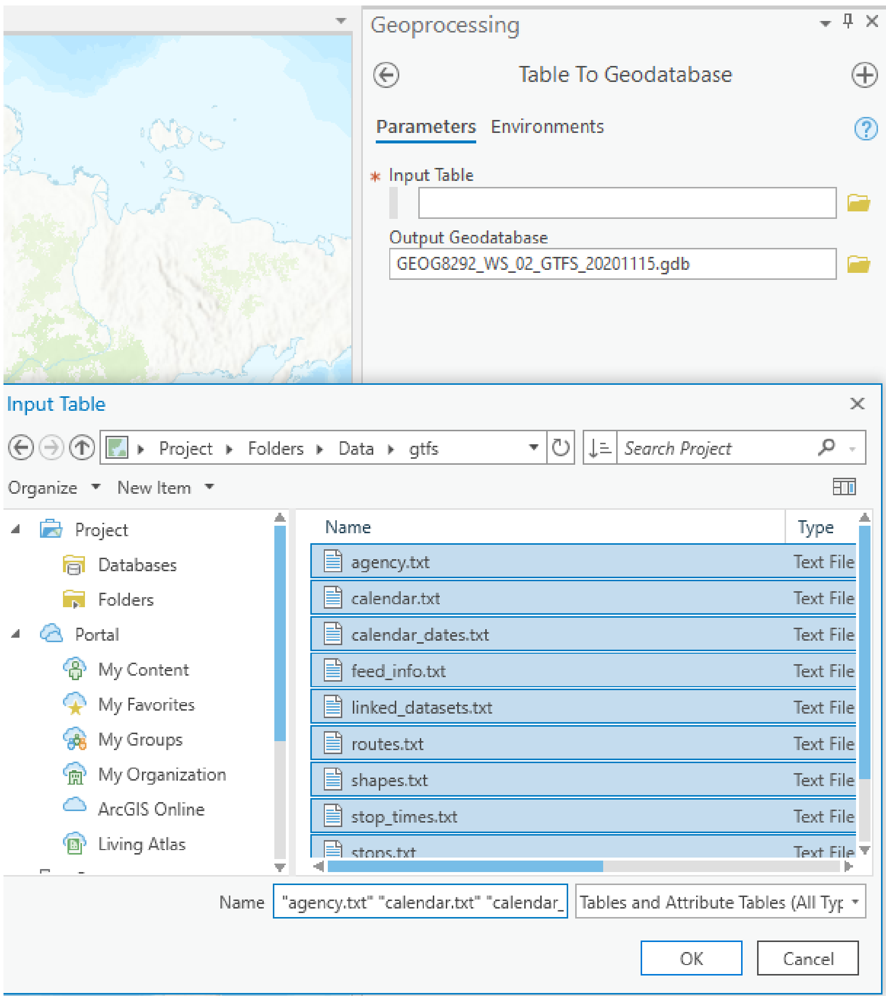
Beside the GTFS data, we will also use the road centerline data to "map-match" the stops and routes to the road network. There are different sources to obtain road network data. The most common two used are OSM (worldwide) and TIGER/Line Shapefiles (U.S.). For OSM, there are packages to extract roads at different levels in a given region, such as the OSMnx python package. For regional roads, we can use road centerline dataset provided by the Metro Regional Centerline Collaborative (MRCC) that can also be accessed at the Minnesota Geospatial Commons: https://gisdata.mn.gov/dataset/us-mn-state-metc-trans-fnctnl-cls-rds (accessed on Nov. 15, 2020). This workshop uses the ESRI enterprise online routing service and special network provided by Metro Transit for routing purposes (accessed at https://gisdata.mn.gov/dataset/us-mn-state-metc-trans-transit-streets the transit rights-of-way lands at https://gisdata.mn.gov/dataset/us-mn-state-metc-trans-transit-row-segments in addition to all roads at https://gisdata.mn.gov/dataset/us-mn-state-metrogis-trans-road-centerlines-gac).

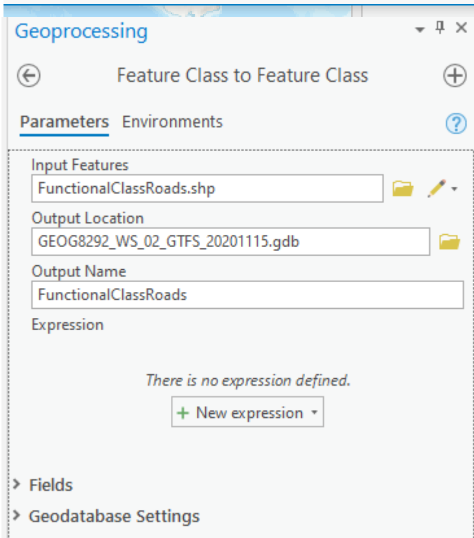
2. Tool

This workshop uses the Transit Feed (GTFS) toolset in ArcPro since programming is not required for this seminar. There are other open source toolsets to generate features and conduct analysis such as Peartree (https://awesomeopensource.com/project/kuanb/peartree) and google transit APIs with Realtime GTFS (https://developers.google.com/transit/gtfs-realtime/examples/python-sample).

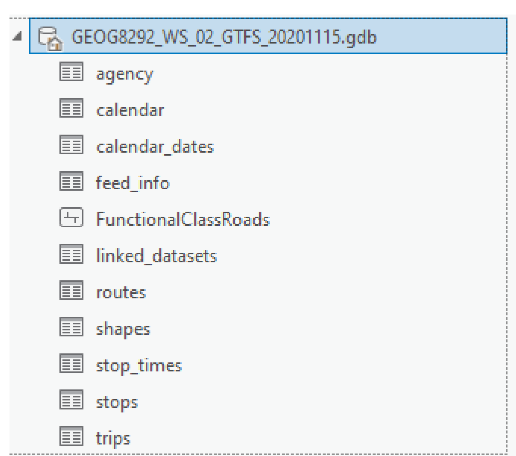
❖ Procedures – Part One: Understand GTFS Data Structure

1. Open ArcGIS Pro, create a new project, and import data into the default geodatabase.
   (I also suggest adding a folder connection to where you store the downloaded data and docs).





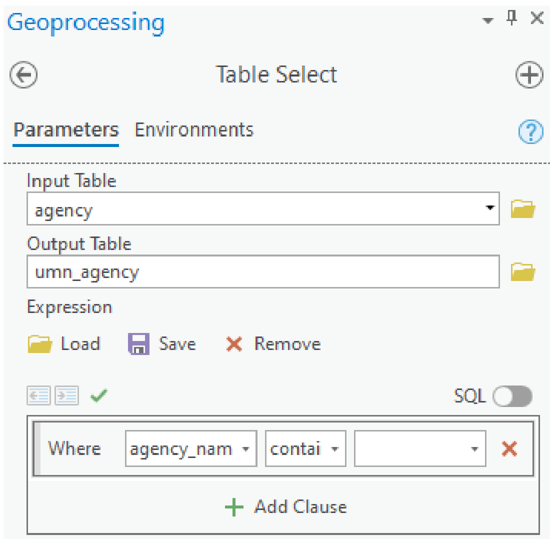
(Result GDB)



2. Explore the tables
Please use Figure 2 as reference to understand the relationship between these tables.
(1) agency

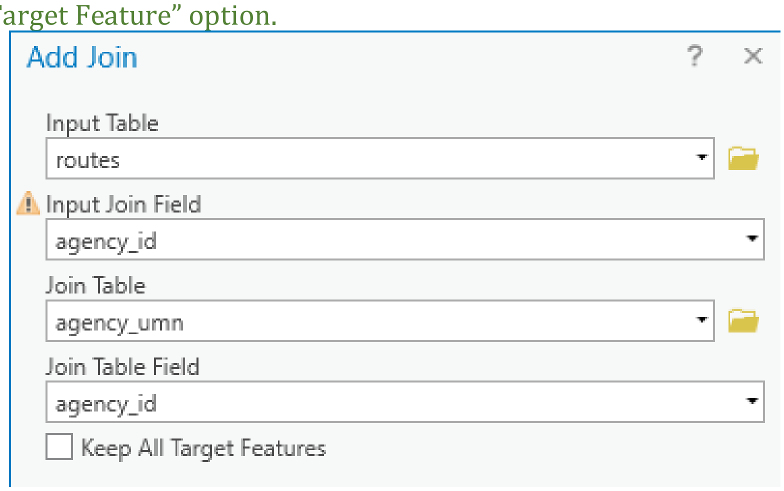
There are 9 transit agencies included in GTFS data for the Twin Cities metro areas (e.g. the Metro Transit) and beyond (e.g. Maple Grove, Plymouth).
- *agency_url* contains the link to the official website of each agency
- *agency_id* will be used to join to the routes table in Step (2)

Table select the agency "University of Minnesota".



(2) **routes**
- *route_type*: We will need this to determine what type of transit we have in our dataset. For the complete list, please go to: https://sites.google.com/site/gtfschanges/proposals/route-type. For the Twin Cities data, we have 0-Tram, Light Rail, Streetcar, 2-Rail, and 3-Bus.
- *agency_id* will be used to join to the agency table in this step.
- *route_id* will be used to join to the trips table in Step (3).

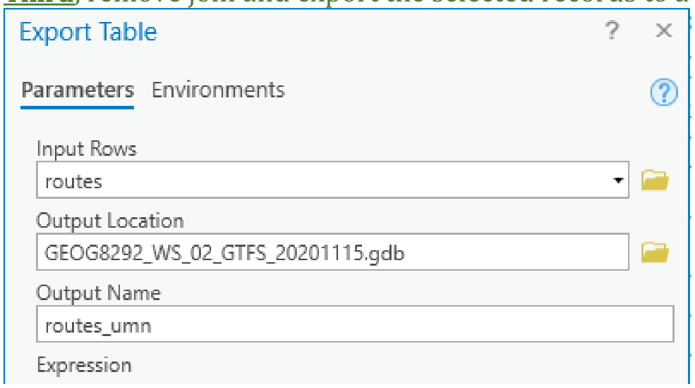
Select all routes operated by UMN
First, add join to select routes operated by UMN, which can be done by unchecking the "Keep All Target Feature" option.



Second, open the attribute table of routes and select all records in the table. There are 5 routes operated by UMN as shown in the joined result.

OBJECTID *	route_id	agency_id *	route_short_name	route_long_name	route_desc	route_type	route_url
48	120	11	120	<Null>	U of M - East Bank Cir...	3	https://www.n
49	121	11	121	<Null>	U of M - Campus Con...	3	https://www.n
50	122	11	122	<Null>	U of M - University Av...	3	https://www.n
51	123	11	123	<Null>	U of M - 4th Street Cir...	3	https://www.n
52	124	11	124	<Null>	U of M - Saint Paul Cir...	3	https://www.n

Third, remove join and export the selected records to a new table.

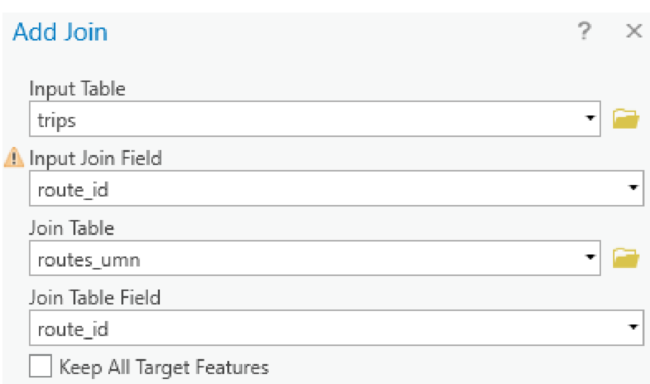


(3) **trips**

Each route often has several trips.

- *route_id* will be used to join to the routes table in this step in this step.
- *shape_id* will be used to join to the shapes table in Step (4)
- *trip_id* will be used to join to the stop_times table in Step (5)
- *service_id* will be used to join to the calendar and calendar_dates tables in Step (7)

First, add join to select trips along all UMN routes



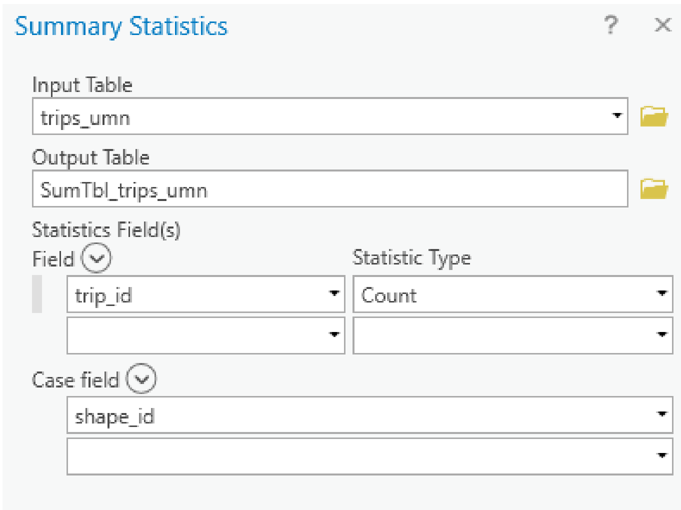
Second, open the attribute table of trips and select all records in the joint table.

Third, remove join and export the selected records to a new table "trips_umn".

(4) **shapes**

Each “shape” has an ordered sequence of points. Trips along the same routes may share the shape.
- *shape_id*: you can view it as the ID of a line to represent the route of a trip. It will be used in this step to join to the trips table.

First, in the trips table, summarize by *shape_id*. To get all “shapes” we need for UMN transit trips. We can also see how many trips share the same shape.



As we can see, there are 9 shapes for the 9 routes in step 2. For some of the buses runed by Metro Transit, their shapes may change over the time, which means the route and shape may not be one-to-one relationship.

Field: Add Calculate Selection: Select By Attribute

OBJECTID *	shape_id	FREQUENCY	COUNT_trip_id
1	1200001	70	70
2	1210001	372	372
3	1210002	372	372
4	1210003	298	298
5	1210004	300	300
6	1220001	132	132
7	1220002	420	420
8	1230001	90	90
9	1240002	68	68

Click to add new row.

Second, we join the table shapes with the summary table SumTbl_trips_umn. This would allow us to “select” points along those 9 shapes.

Third, open the attribute table of shapes and select all records in the joint table (1,794 records).

Fourth, remove join and export the selected records to a new table “shapes_umn”.

(5) **stop_times**

Each record in the table stop_times stores the *arrival_time* and the *departure_time* at a stop along a trip.

- *trip_id* will be used to join to the trips table in this step.
- *stop_id* will be used to join to the stops table in Step (6).

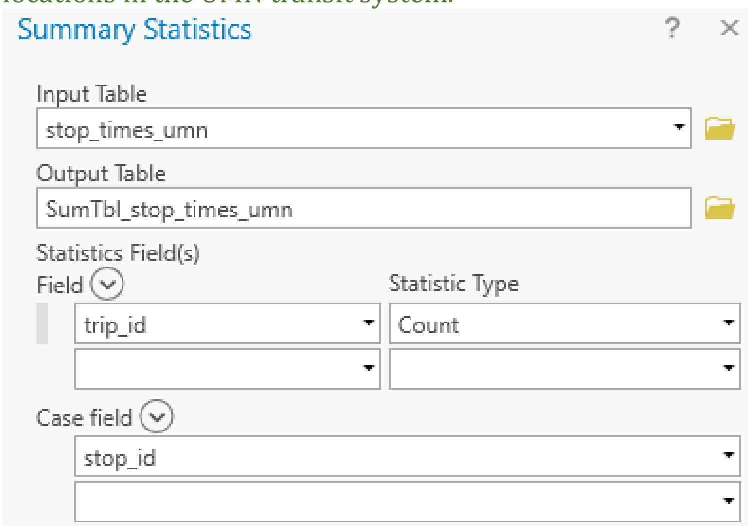
Like the previous steps, we first join stop_times with trips, then select all records in the joint table and remove the join, and finally export the selected records to a new table "stop_times_umn". There are 18,722 stop_times records in the exported table.

(6) **stops**

Each record in the table stop stores the *stop_lon* and *stop_lat* of a point. Note that one stop may be shared by several trips, and therefore has several stop_times.

- *stop_id* will be used to join to the stop_times table in this step.

Like Step (4), we first get stop locations (stop_id) for all stop_times. There are totally 49 stop locations in the UMN transit system.



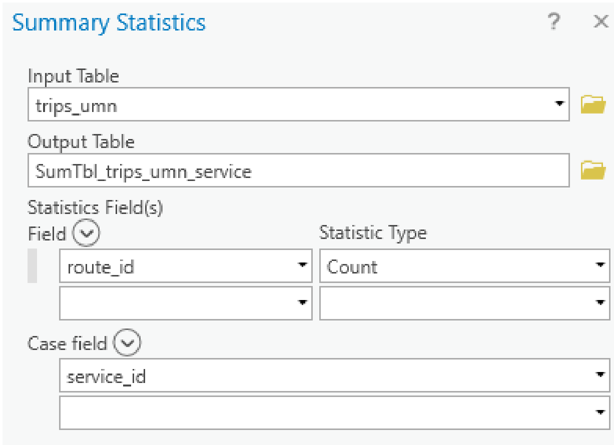
Then, we join stops with the summary table, select all records in the joint table and remove the join, and export the selected records to a new table "stops_umn". There are 49 stops exported.

(6) **calendar** and **calendar dates**

The calendar and calendar dates store the service information (e.g. the day of the week when services are provided and the start and end date of such services).

- *service_id* will be used to join to the trips table in this step.

Like Step (4), we first get services (service_id) for all stop_times. There are totally 49 stop locations in the UMN transit system.

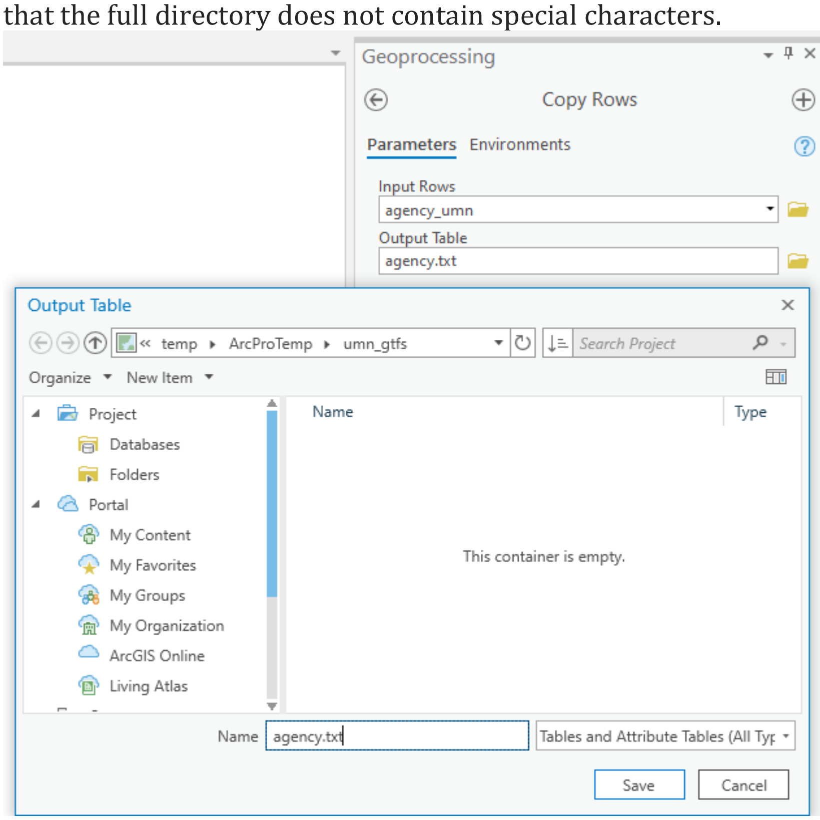


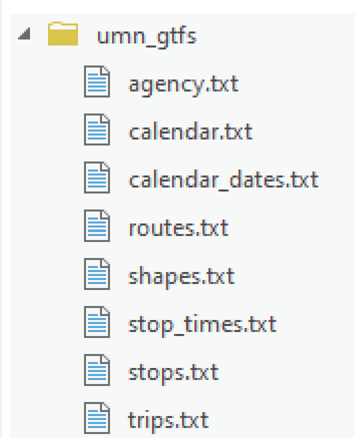
Then, we join calendar_dates table with the summary table, select all records in the joint table and remove the join, and export the selected records to a new table "calendar_dates _umn". There are 2 records exported.

Also, repeat the same steps for calendar table and export the table as "calendar_umn". There are 6 records exported.

(7) Export the required tables as text file

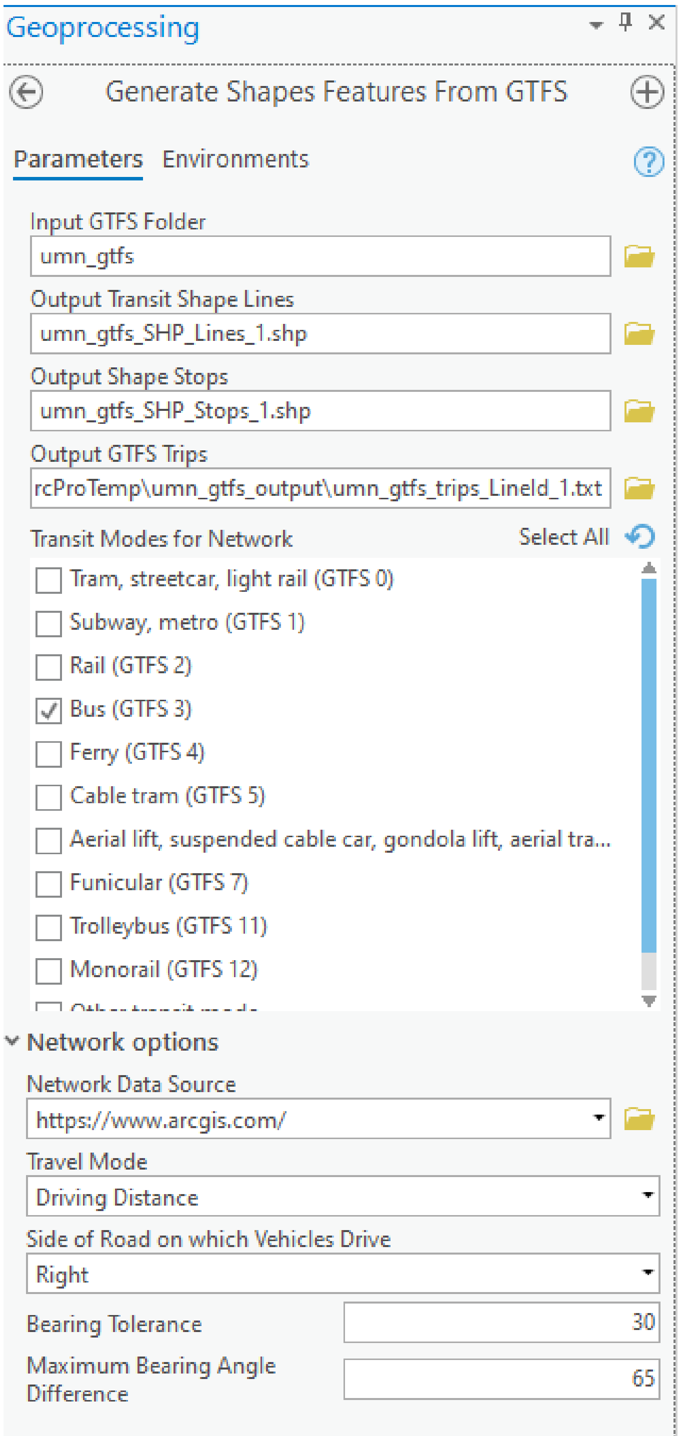
To use the tools in ArcPro, we need to create a new folder for all required (or conditional required) tables. Here, I create a folder in my working folder and name it as "umn_gtfs". Again, make sure that the full directory does not contain special characters.



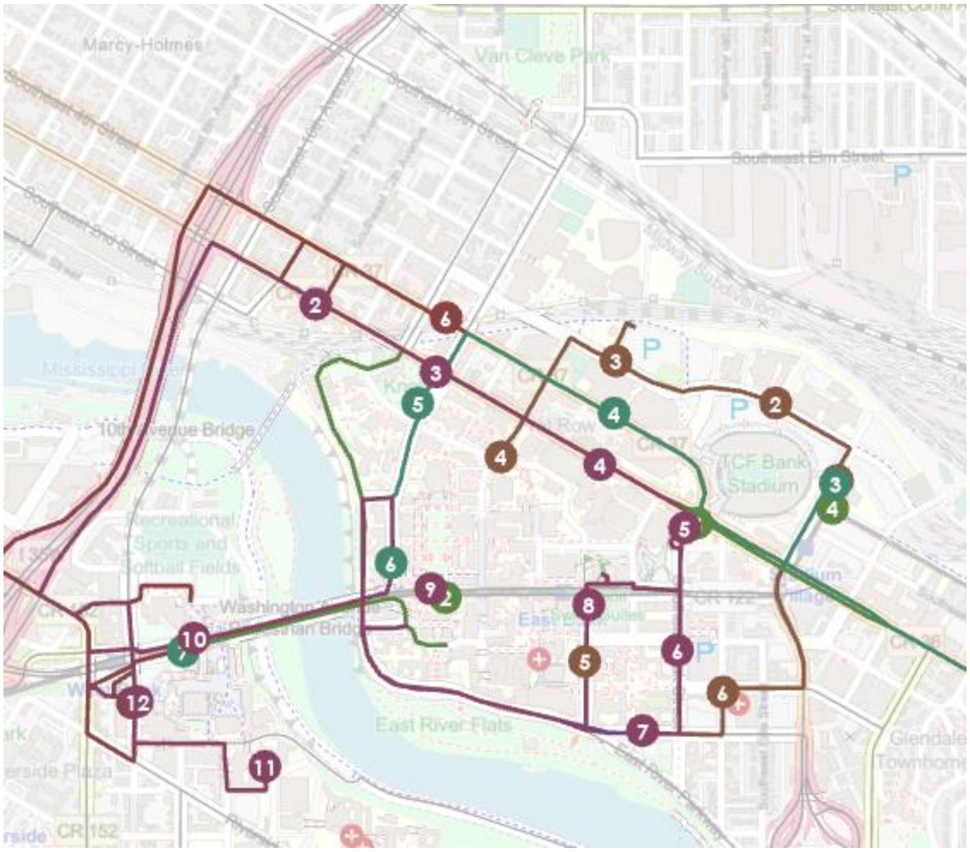


Part Two: Create and compare spatial objects using different methods

3. Create spatial objects based on GTFS files
(1) Directly use the Generate Shapes Features From GTFS and ESRI ArcGIS routing service
Note: I have tried to save the outputs in the geodatabase but failed. But it works if we save the output as individual shapefiles.



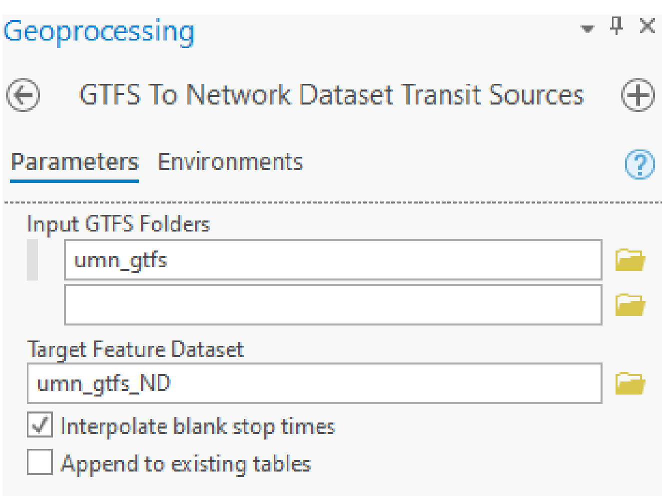
Due to the restricted driving along Washington Ave, the result maps do not reflect the actual trip routes of UMN shuttle services. Also, these points and lines are shapefiles, which do not contain any topological information.



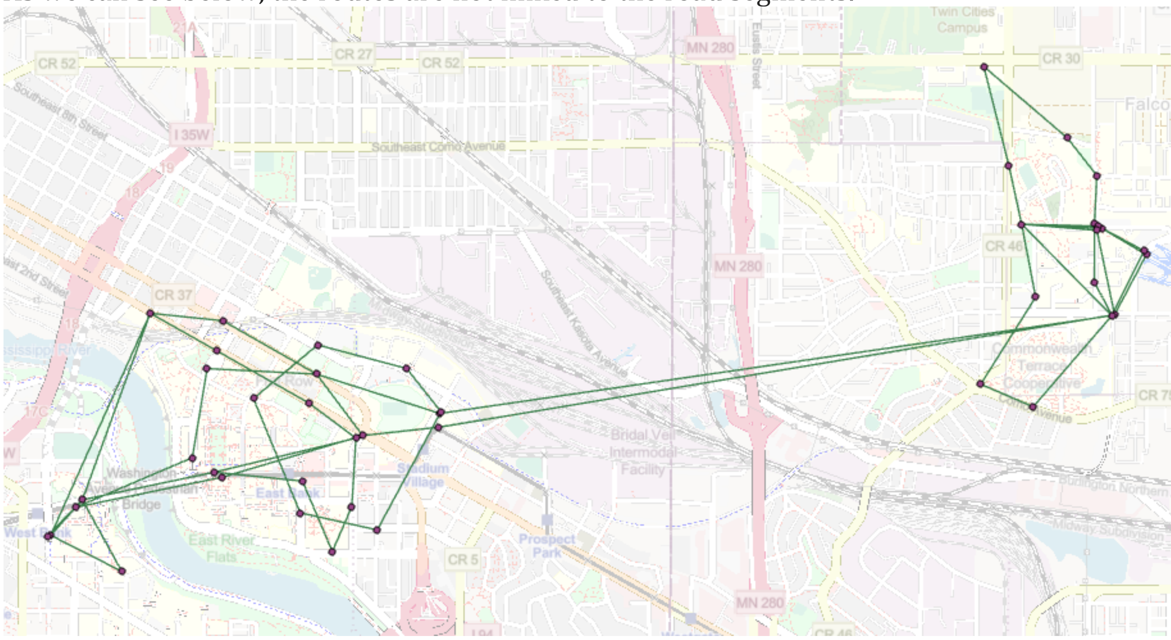
(2) Use the GTFS To Network Dataset Transit Sources and Transit Network Dataset

__First__, create a new feature dataset to save the output called "umn_gtfs_ND".

__Second__, use the GTFS To Network Dataset Transit Sources.



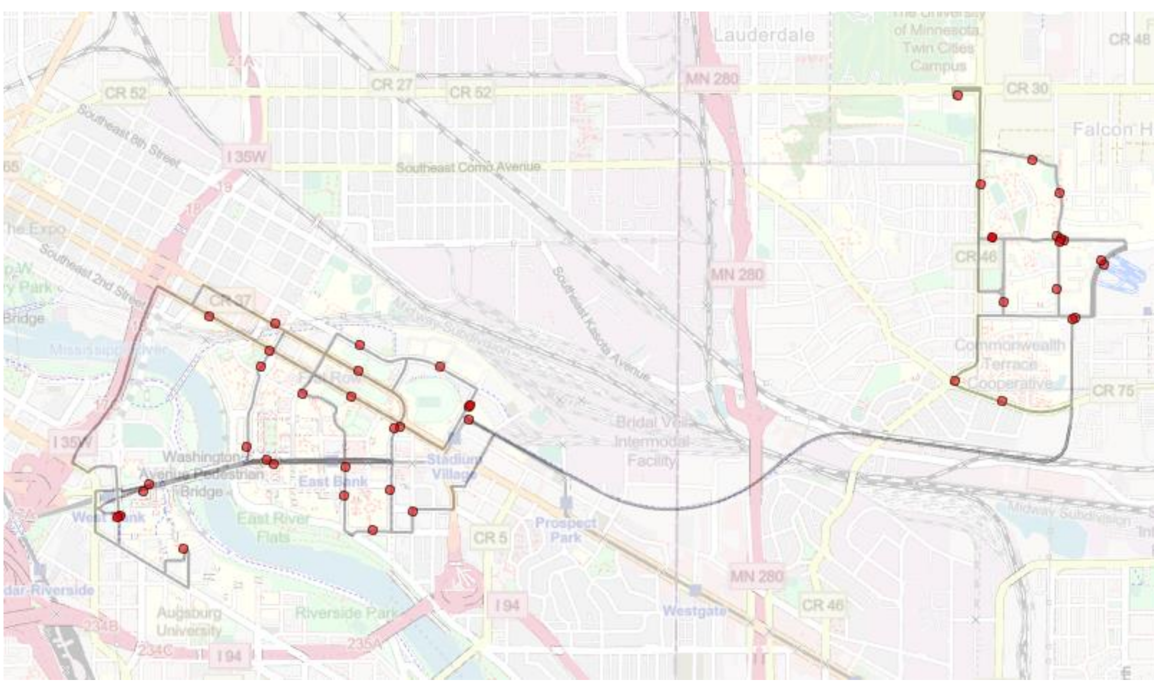
As we can see below, the routes are not linked to the road segments.



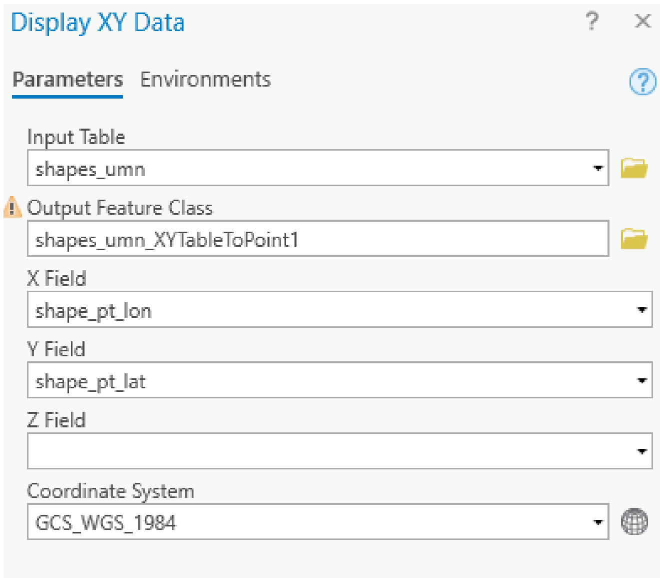
However, this would allow us to create a transit-integrated network to be used in ArcGIS network analysis packages. (if time allows, we will create and use a network dataset with public transit data (https://pro.arcgis.com/en/pro-app/help/analysis/networks/create-and-use-a-network-dataset-with-public-transit-data.htm).

(3) Use GTFS Shapes/Stops to Features

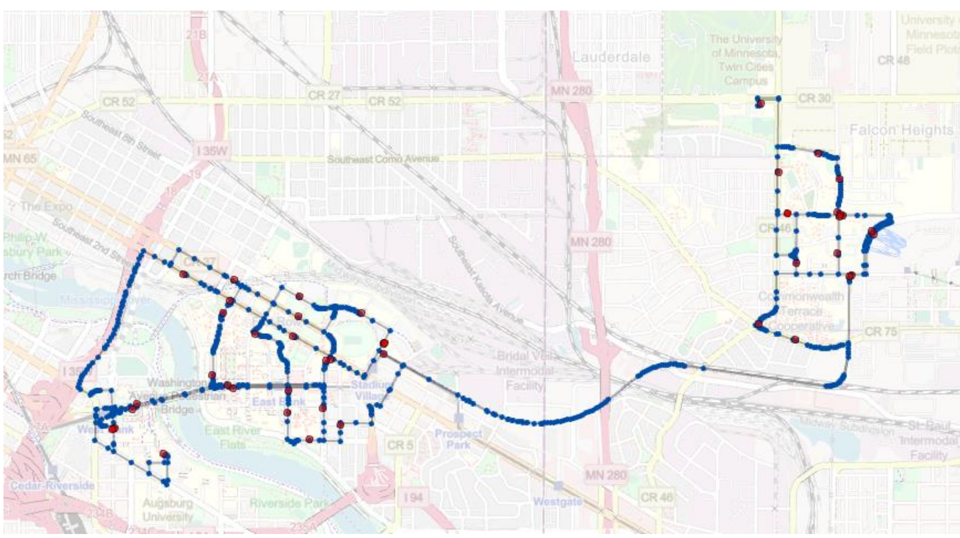
These would allow us to generate the routes and stop shapes that are best fit with the reality.

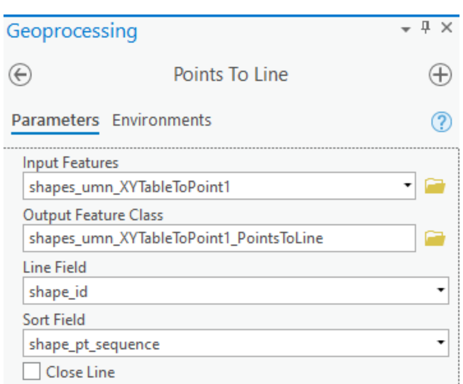


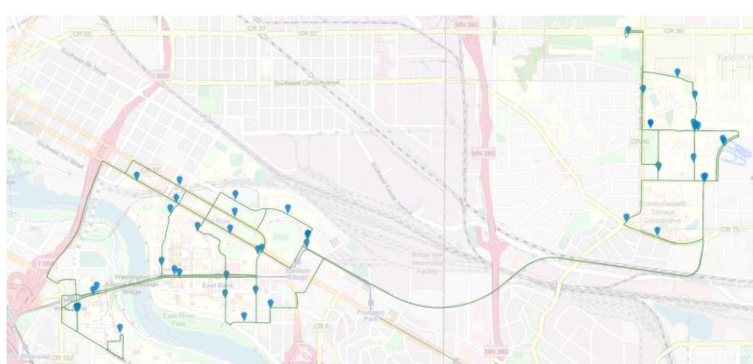
(4) Displace X, Y coordinates to create events data



These would allow us to displace points along the trips to stores their shapes. However, further steps need to be taken to generate the shape for each trip (e.g. using Points to Line)







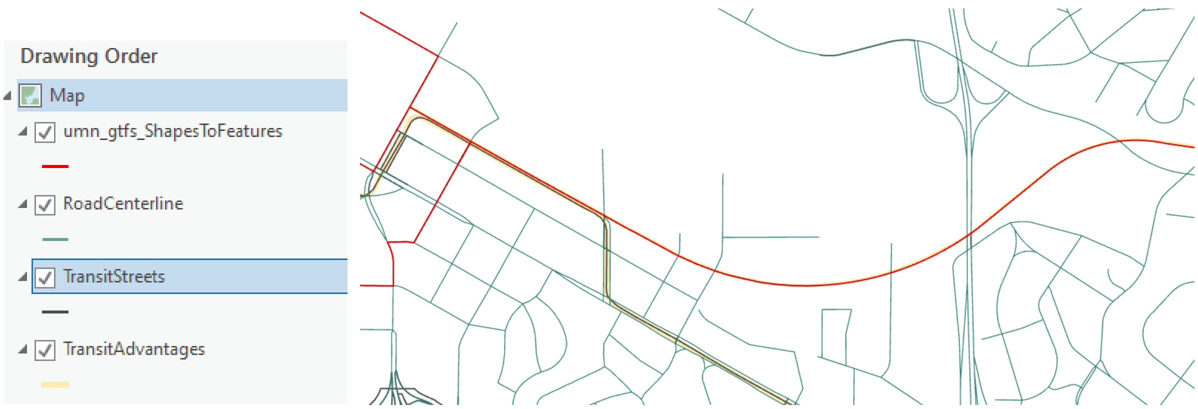
Part Three: Create network dataset for transit service analysis

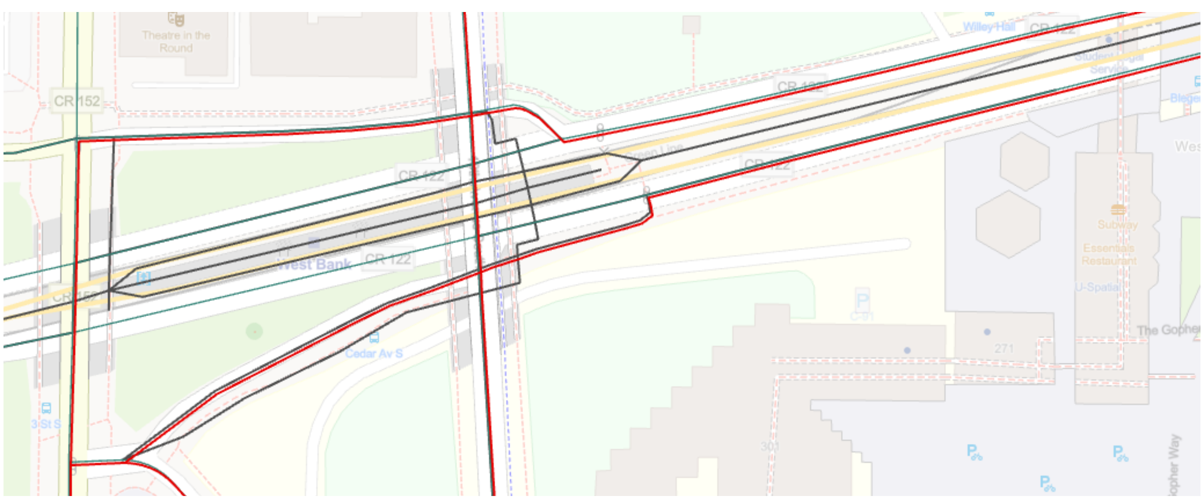
4. Create a Network Dataset integrated with the GTFS transit network

(1) **Import shapefiles**

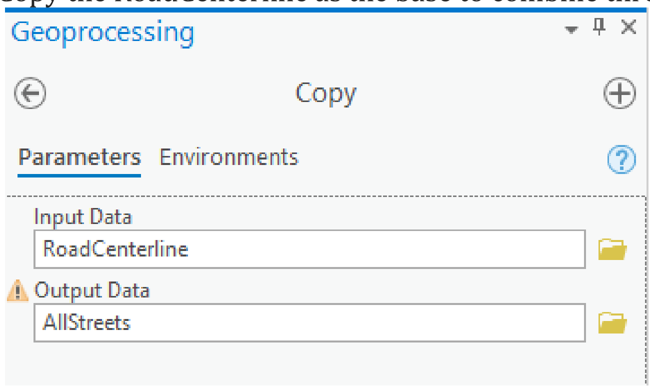
- Import three road centerline datasets into the geodatabase
  - ✓ RoadCenterline: centerlines of all roads for regular driving, biking and walking purposes. https://gisdata.mn.gov/dataset/us-mn-state-metrogis-trans-road-centerlines-gac
  - ✓ TransitStreets: added small segments that for transit routing purposes, such as light rail and some small segments cross roads https://gisdata.mn.gov/dataset/us-mn-state-metc-trans-transit-streets
  - ✓ TransitAdvantages: rights-of-way that for transit uses only, such as light rail and bus only roads https://gisdata.mn.gov/dataset/us-mn-state-metc-trans-transit-row-segments

The images below showed that we need to combine three datasets together to create a road network that are valid for transit analysis. The image above shows the rights-of-way that only in the TransitAdvantages dataset and the image below shows small road segments that are only in the TransitStreets dataset.

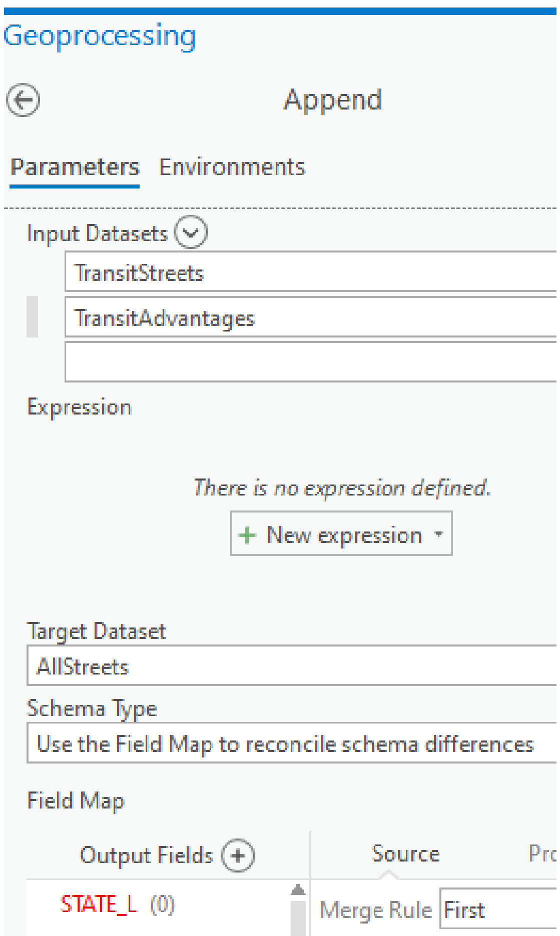


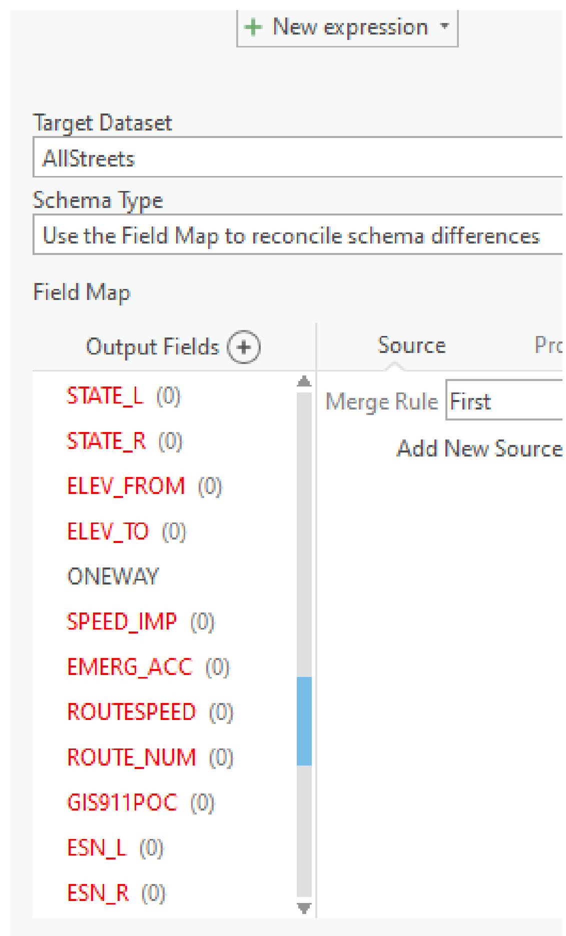


- Copy the RoadCenterline as the base to combine three datasets as “AllStreets”



- Append the other two datasets to it. Note that they have different schema, so we keep all fields in the output and will further explore the attributes. The only common fields for these three datasets are ONEWAY.



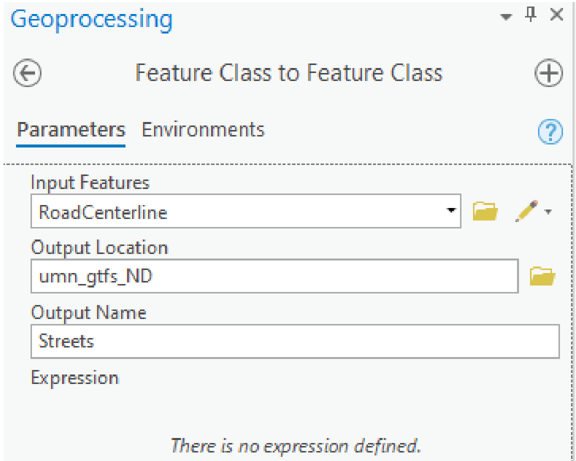


After examining the attributes, it seems to be quite time-consuming to create our own network dataset with network embedded in. So, we will compromise it and resort for a different way.

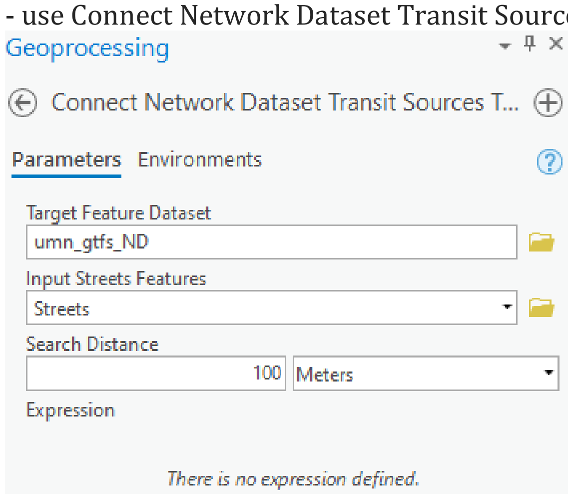
5. Combine regular network dataset with the GTFS transit, no geometry preserved.

(1) **Continue with the GTFS network dataset** in previous step 4.(2)

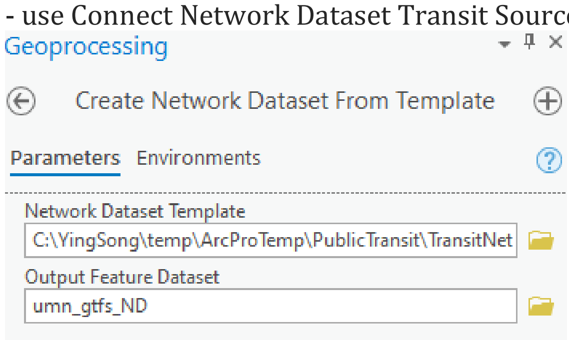
- add road centerlines to the feature dataset



- use Connect Network Dataset Transit Source to Streets to map-match stops to roads



- use Connect Network Dataset Transit Source to Streets to map-match stops to roads



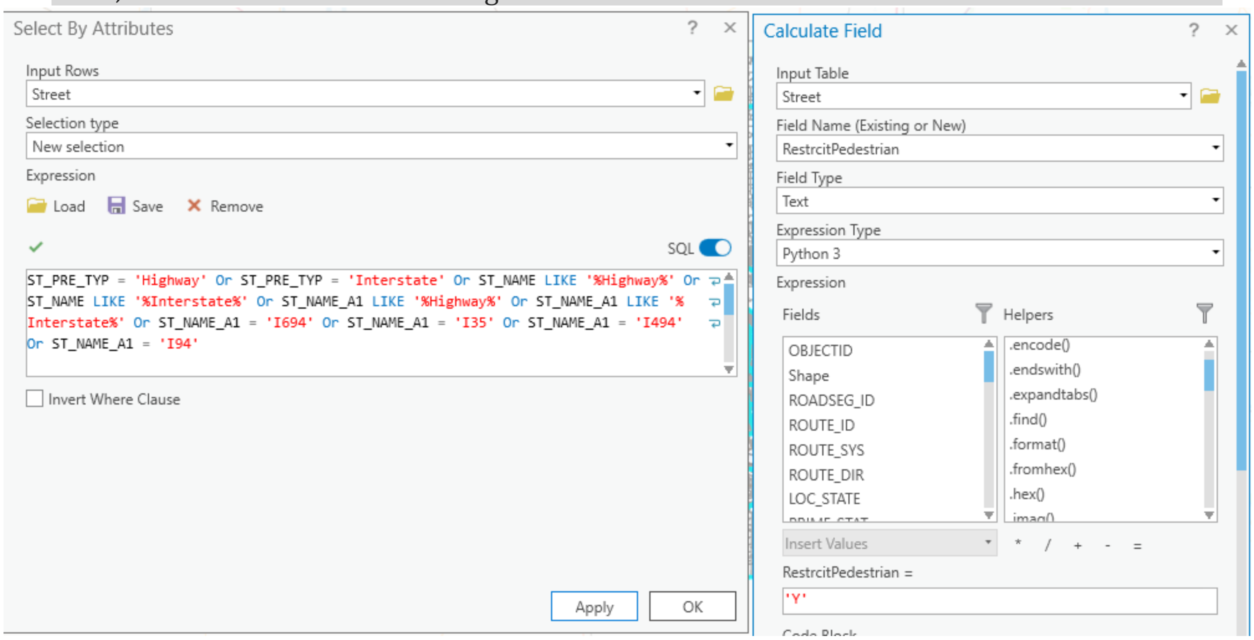
(2) **Modify the attribute** of the centerlines of roads

We will add a field to show whether the roads are walkable or not. For this workshop, we will use the type of the roads and the name of the road to narrow down our selection.

SQL for highways links -> RestrictPedestrian = "Y"

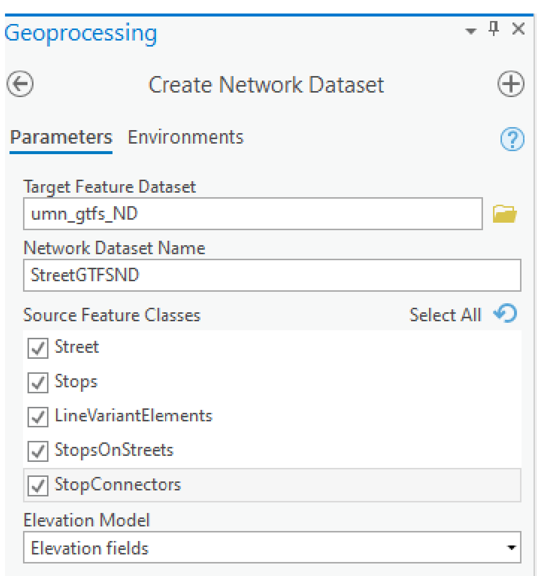
ST_PRE_TYP = 'Highway' Or ST_PRE_TYP = 'Interstate' Or
ST_NAME LIKE '%Highway%' Or ST_NAME LIKE '%Interstate%' Or
ST_NAME_A1 LIKE '%Highway%' Or ST_NAME_A1 LIKE '%Interstate%' Or ST_NAME_A1 = 'I694' Or
ST_NAME_A1 = 'I35' Or ST_NAME_A1 = 'I494' Or ST_NAME_A1 = 'I94'

Then, switch the selection and assign value to the rest of the links -> RestrictPedestrian = "N"

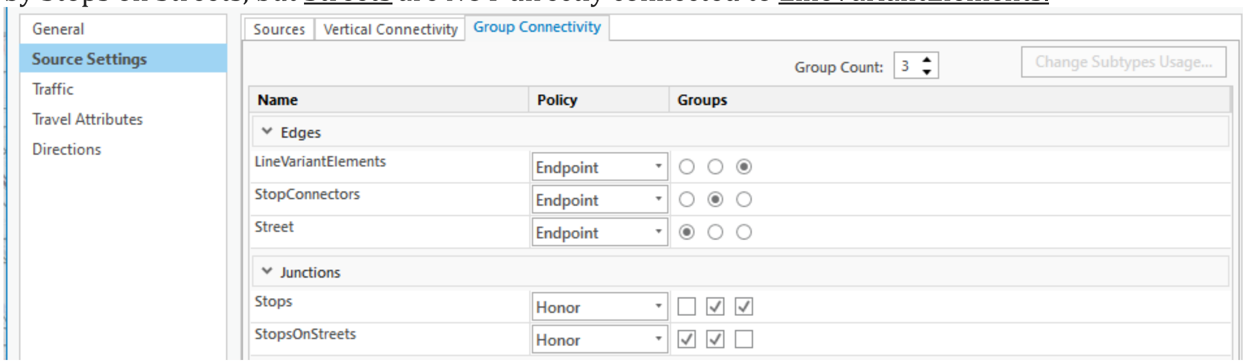


(3) Create a Network Dataset (Continue in Workshop #3 if needed)

- We will use the transit schedules and roads to develop a walk, transit, and drive network. See here of the specification of transit network dataset in ArcGIS Pro: https://pro.arcgis.com/en/pro-app/help/analysis/networks/transit-data-model.htm

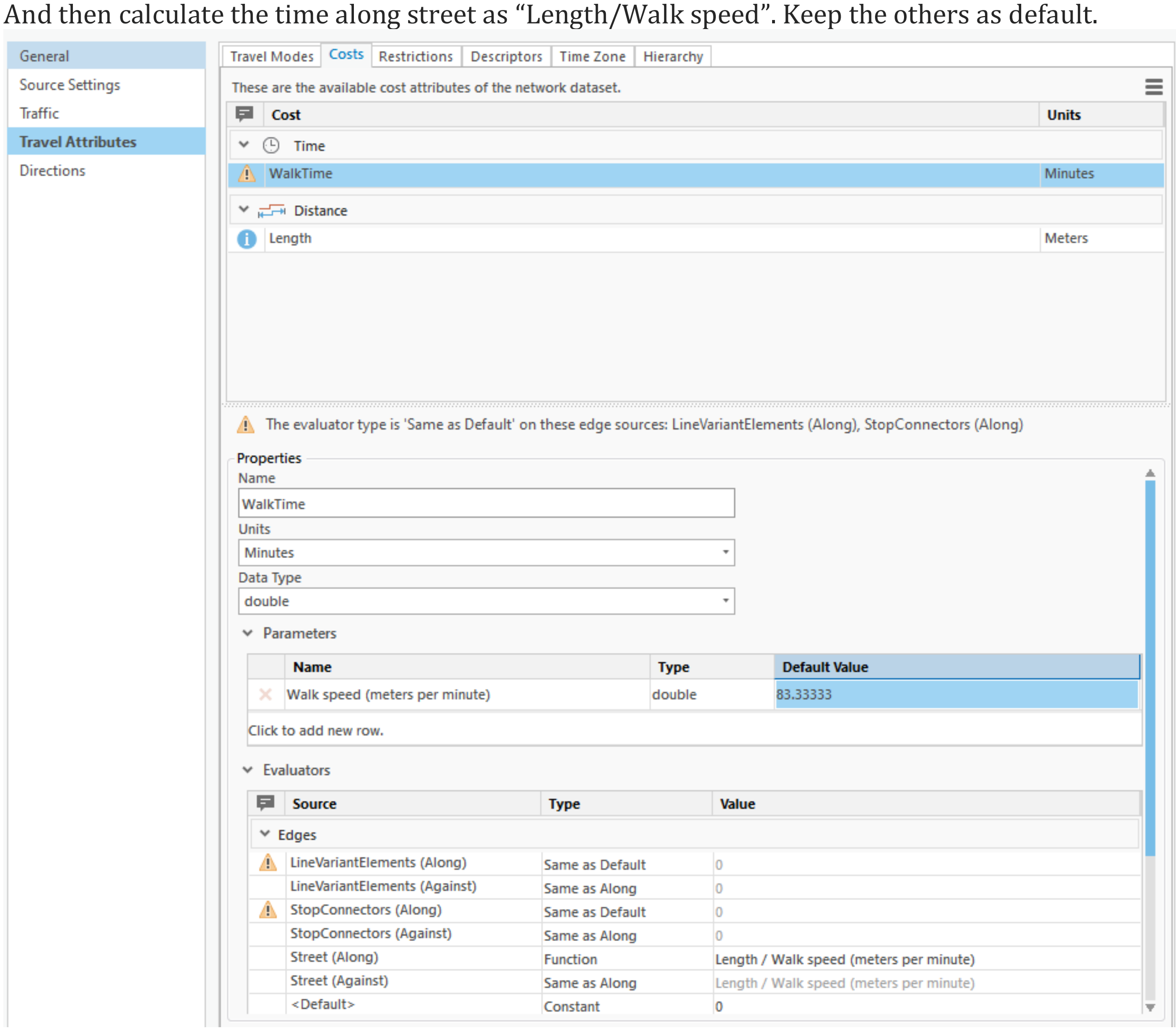


- Then, we set the connectivity by groups. This setting groups edges and junctions so that links and nodes can better match to each other. For instance, the Streets and StopConnectors are connected by Stops on Streets, but Streets are NOT directly connected to LineVariantElements.

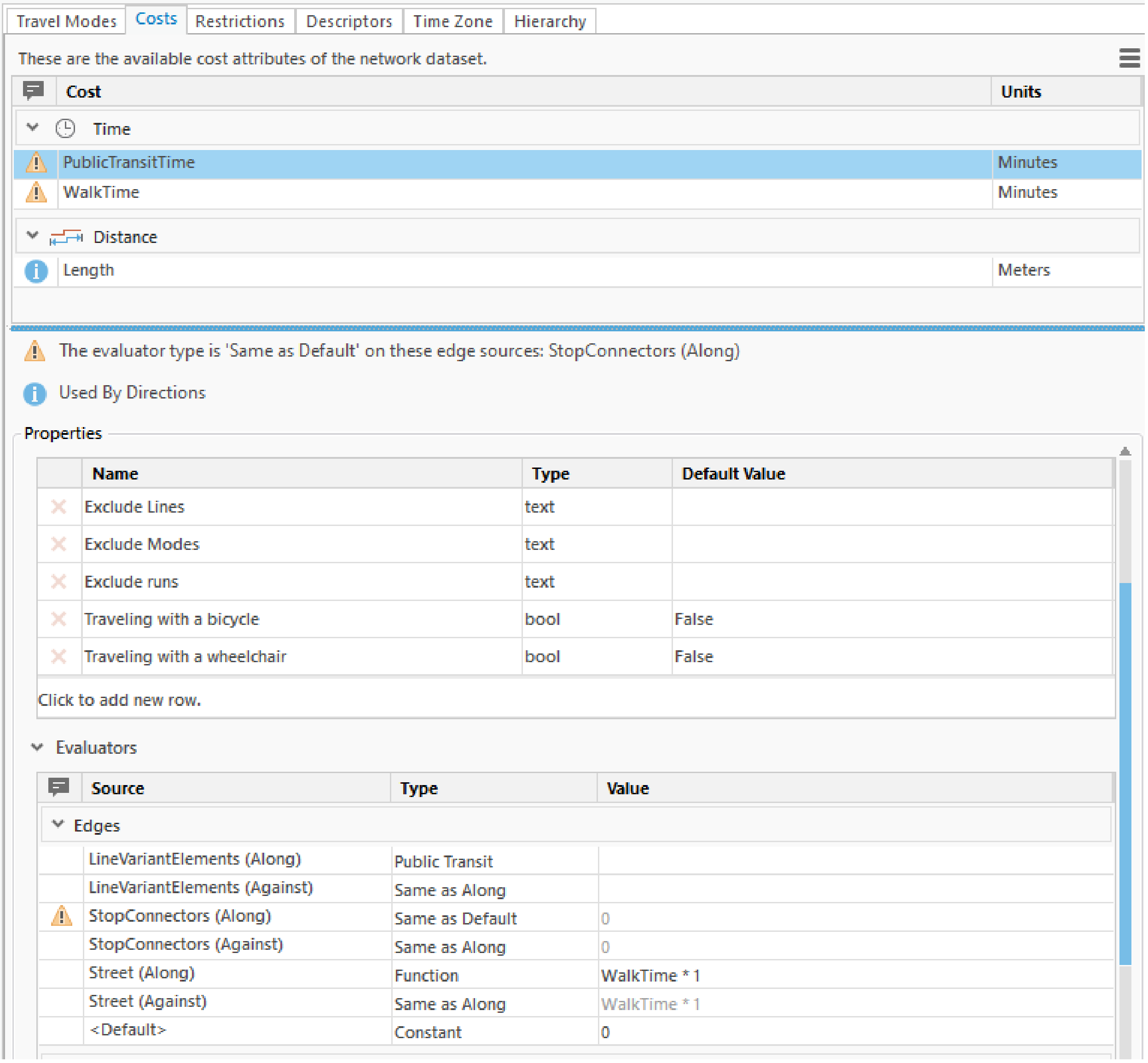


- Set the Costs

(a) add a cost for WalkTime. Add a new parameter for walking speed (83.3333 meter per minute). And then calculate the time along street as "Length/Walk speed". Keep the others as default.

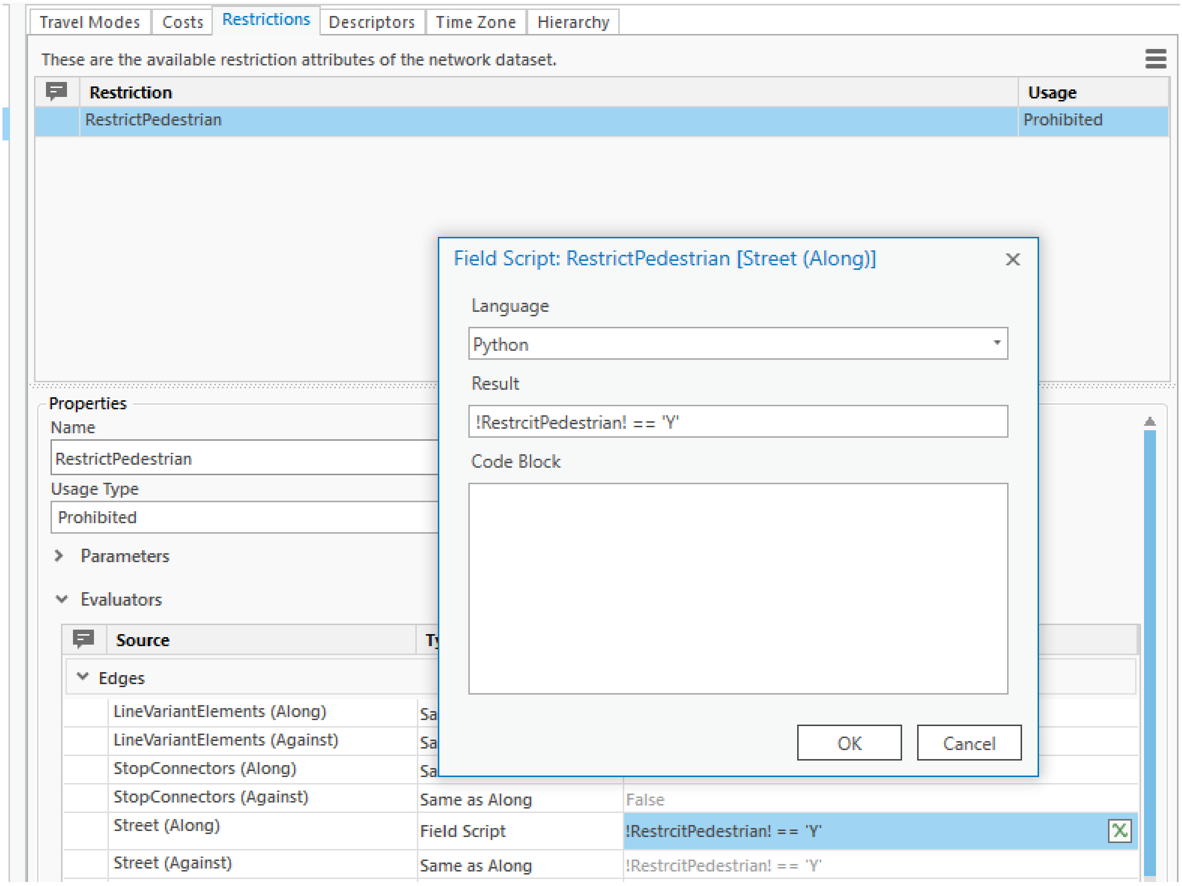


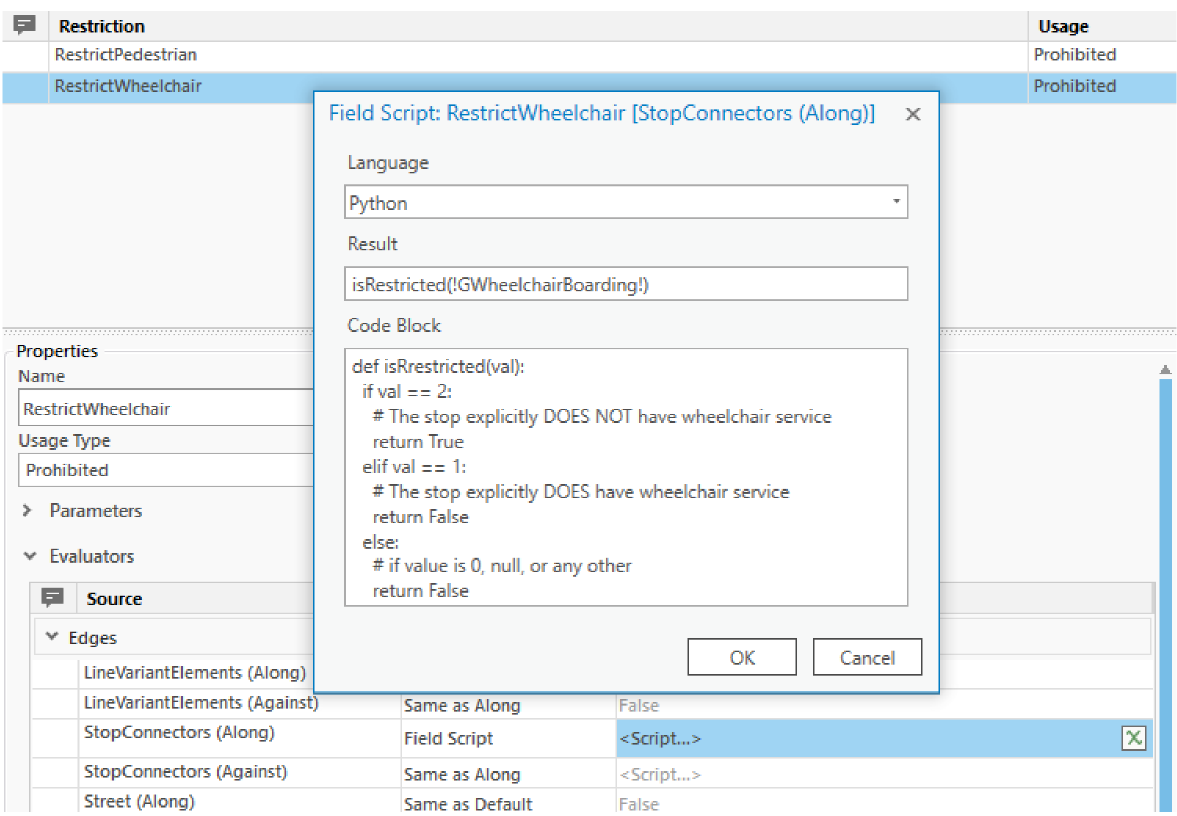
(b) add a cost for PublicTransitTime. Change the LineVariantElements (Along) as PublicTransit and the use WalkTime we just calculated for time (Along) Street.



(c) Add restrictions for pedestrian access based on the field "RestrictPedestrian": if a road link has "Y" for the field, then the restriction will be "TRUE", and the link will not be traversed. Similarly, add restrictions for wheelchair based on the field "GWheelchairBoarding" in the Stops points.

```
def isRrestricted(val):
  if val == 2:
    # The stop explicitly DOES NOT have wheelchair service
    return True
  elif val == 1:
    # The stop explicitly DOES have wheelchair service
    return False
  else:
    # if value is 0, null, or any other
    return False
```

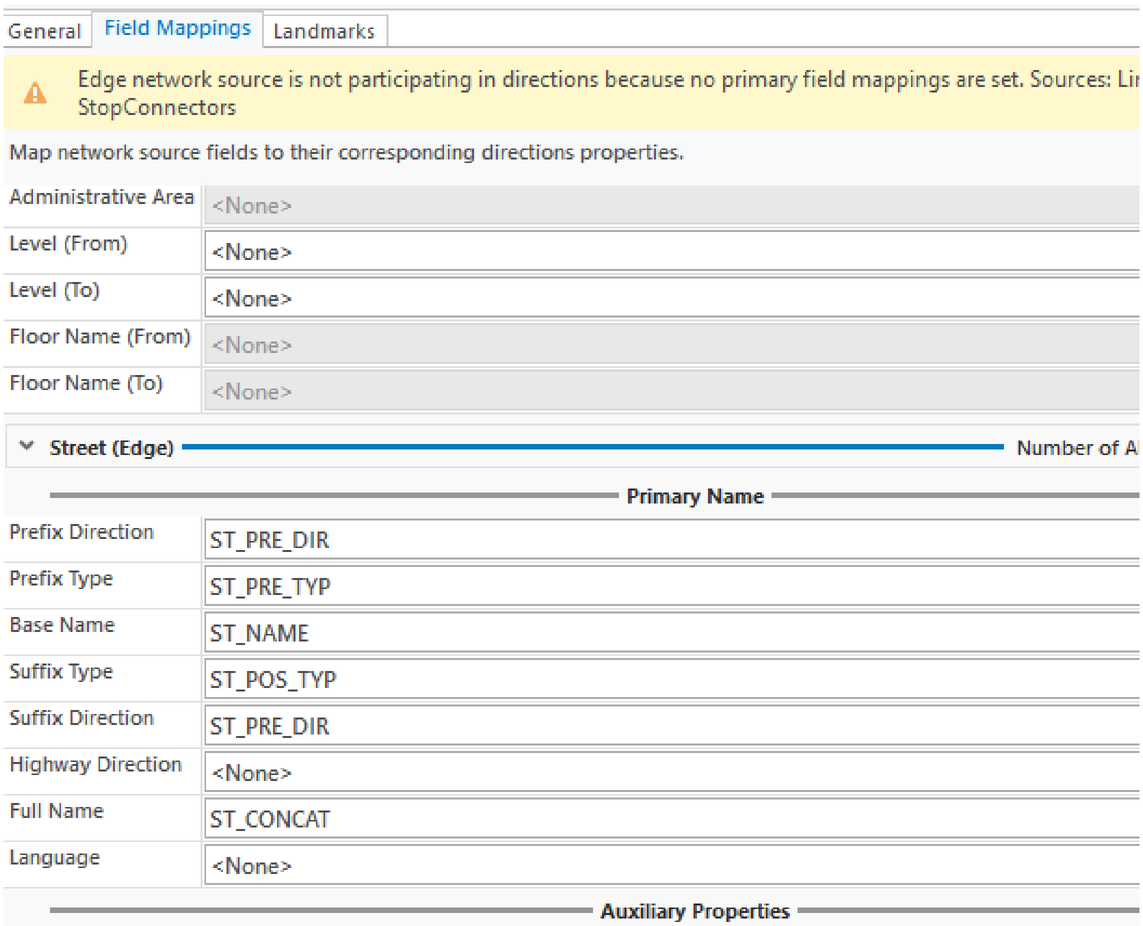

Travel Modes
Costs
Restrictions
Descriptors
Time Zone
Hierarchy
These are the available restriction attributes of the network dataset.
Restriction
Usage
RestrictPedestrian
Prohibited
Properties
Name
RestrictPedestrian
Usage Type
Prohibited
Parameters
Evaluators
Source
Edges
LineVariantElements (Along)
LineVariantElements (Against)
StopConnectors (Along)
StopConnectors (Against)
Same as Along
False
Street (Along)
Field Script
!RestrcitPedestrian! == 'Y'
Street (Against)
Same as Along
!RestrcitPedestrian! == 'Y'
Field Script: RestrictPedestrian [Street (Along)]
Language
Python
Result
!RestrcitPedestrian! == 'Y'
Code Block
OK
Cancel


Restriction
Usage
RestrictPedestrian
Prohibited
RestrictWheelchair
Prohibited
Properties
Name
RestrictWheelchair
Usage Type
Prohibited
Parameters
Evaluators
Source
Edges
LineVariantElements (Along)
LineVariantElements (Against)
Same as Along
False
StopConnectors (Along)
Field Script
<Script...>
StopConnectors (Against)
Same as Along
<Script...>
Street (Along)
Same as Default
False
Field Script: RestrictWheelchair [StopConnectors (Along)]
Language
Python
Result
isRestricted(!GWheelchairBoarding!)
Code Block
def isRrestricted(val):
if val == 2:
# The stop explicitly DOES NOT have wheelchair service
return True
elif val == 1:
# The stop explicitly DOES have wheelchair service
return False
else:
# if value is 0, null, or any other
return False
OK
Cancel

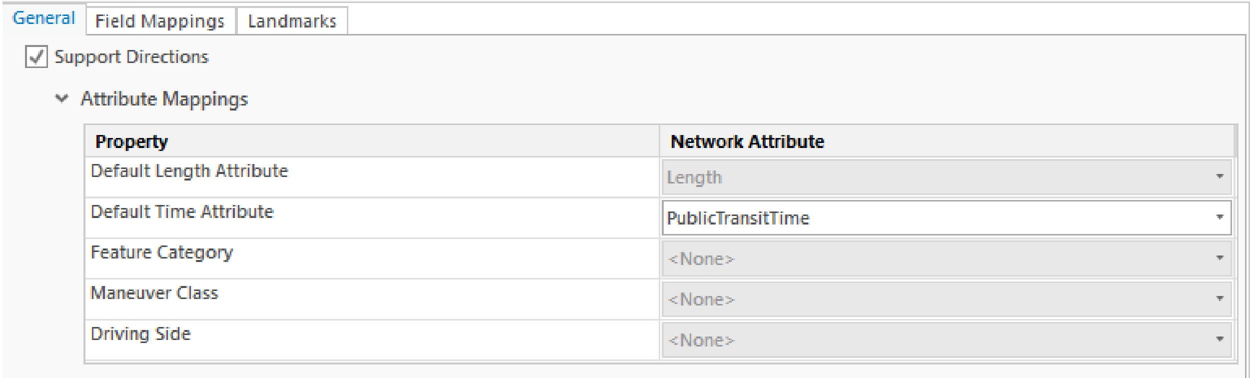
- set directions

(a) assign corresponding fields for streets. We use the road centerline data standard for Minnesota as the basis to determine the appropriate fields. The standard can be accessed via the website: https://www.mngeo.state.mn.us/committee/standards/roadcenterline/index.html

Geocoding Elements	3.1	Street Name Pre Modifier	ST_PRE_MOD	Text	15	Conditional
	3.2	Street Name Pre Directional	ST_PRE_DIR	Text	9	Conditional
	3.3	Street Name Pre Type	ST_PRE_TYP	Text	35	Conditional
	3.4	Street Name Pre Separator	ST_PRE_SEP	Text	20	Conditional
	3.5	Street Name	ST_NAME	Text	60	Mandatory
	3.6	Street Name Post Type	ST_POS_TYP	Text	15	Conditional
	3.7	Street Name Post Directional	ST_POS_DIR	Text	9	Conditional
	3.8	Street Name Post Modifier	ST_POS_MOD	Text	15	Conditional
	3.9	Street Name Full	ST_CONCAT	Text	150	Optional
	3.10	Alternate Street Name 1	ST_NAME_A1	Text	150	Conditional
	3.11	Alt 1 Legitimate MSAG Value	A1_MSAG_V	Text	7	Conditional
	3.12	Alternate Street Name 2	ST_NAME_A2	Text	150	Conditional
	3.13	Alt 2 Legitimate MSAG Value	A2_MSAG_V	Text	7	Conditional
	3.14	Alternate Street Name 3	ST_NAME_A3	Text	150	Conditional
	3.15	Alt 3 Legitimate MSAG Value	A3_MSAG_V	Text	7	Conditional

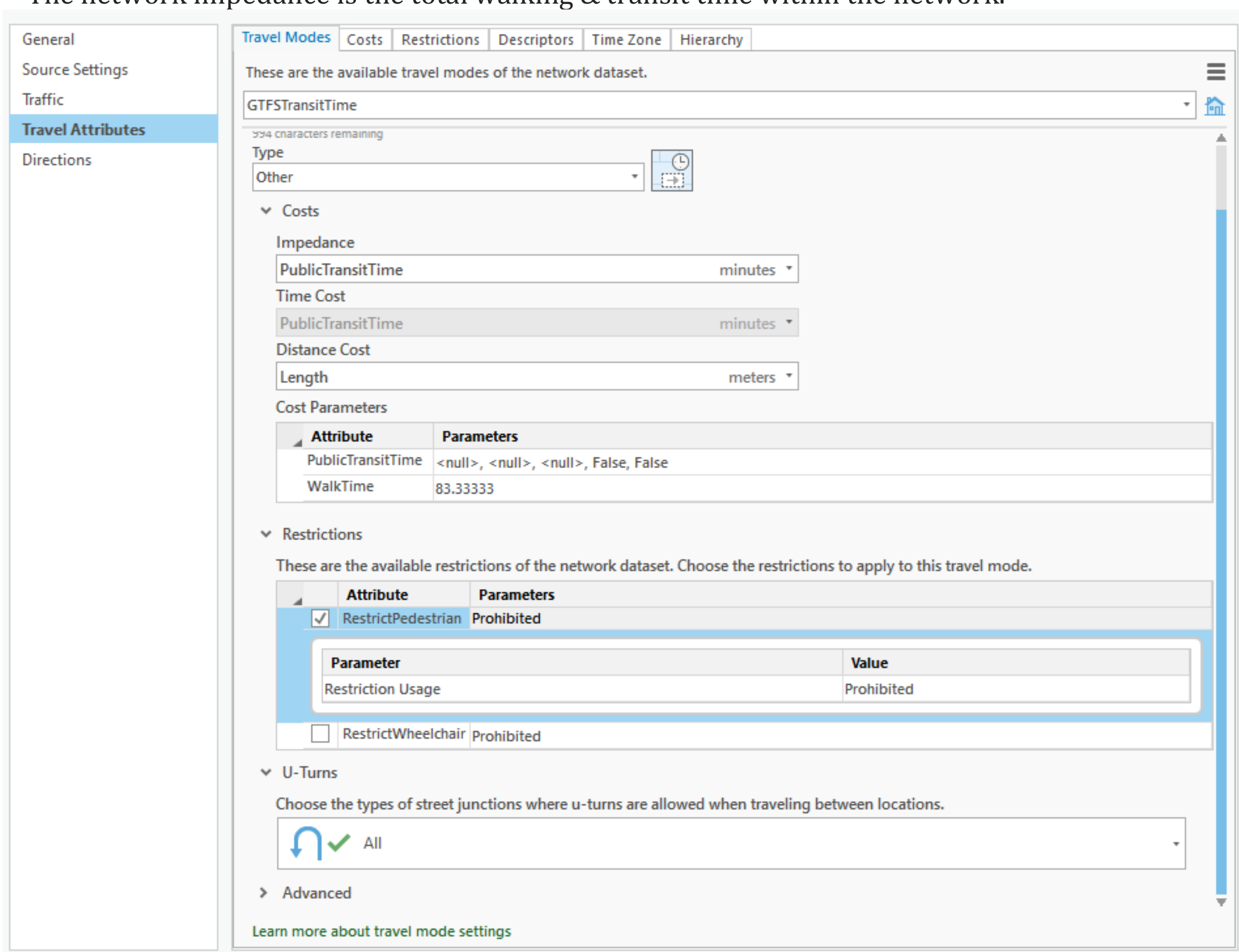


(b) Set the default time attribute as "PublicTransitTime" (created in the travel attribute -> costs)



- Add travel mode GTFSTransitTime

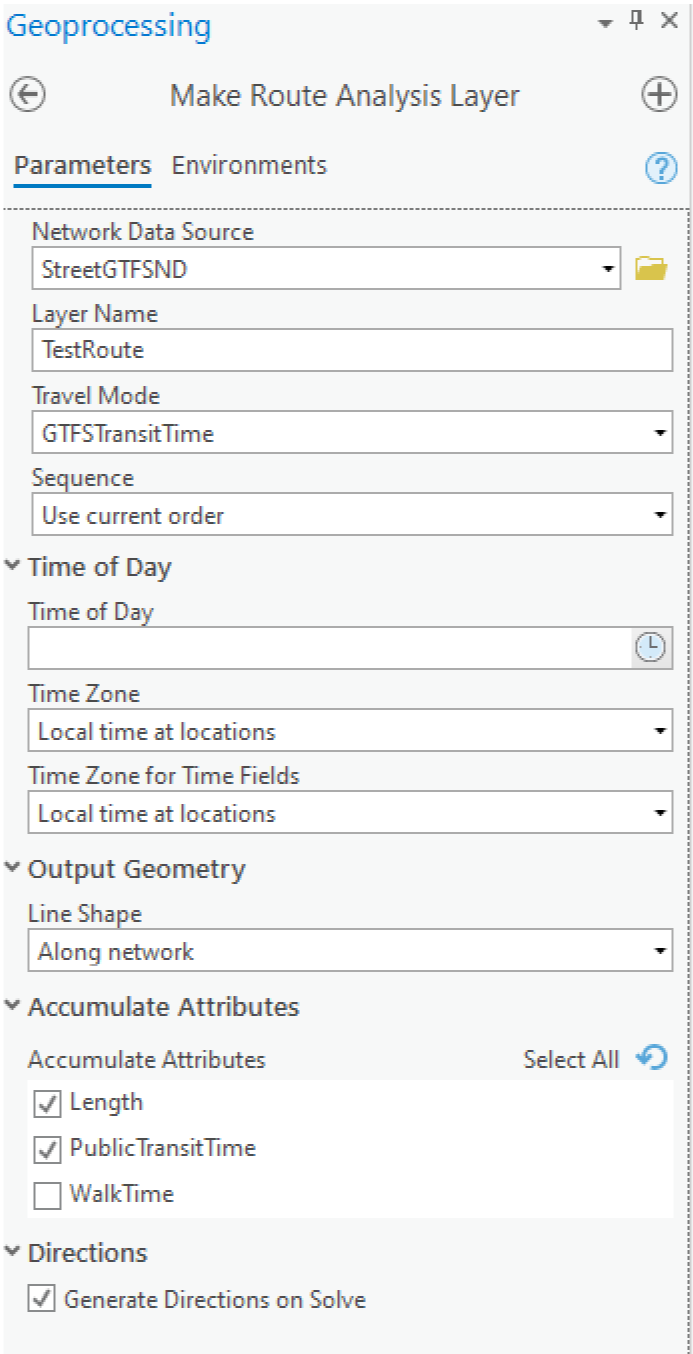
The network impedance is the total walking & transit time within the network.



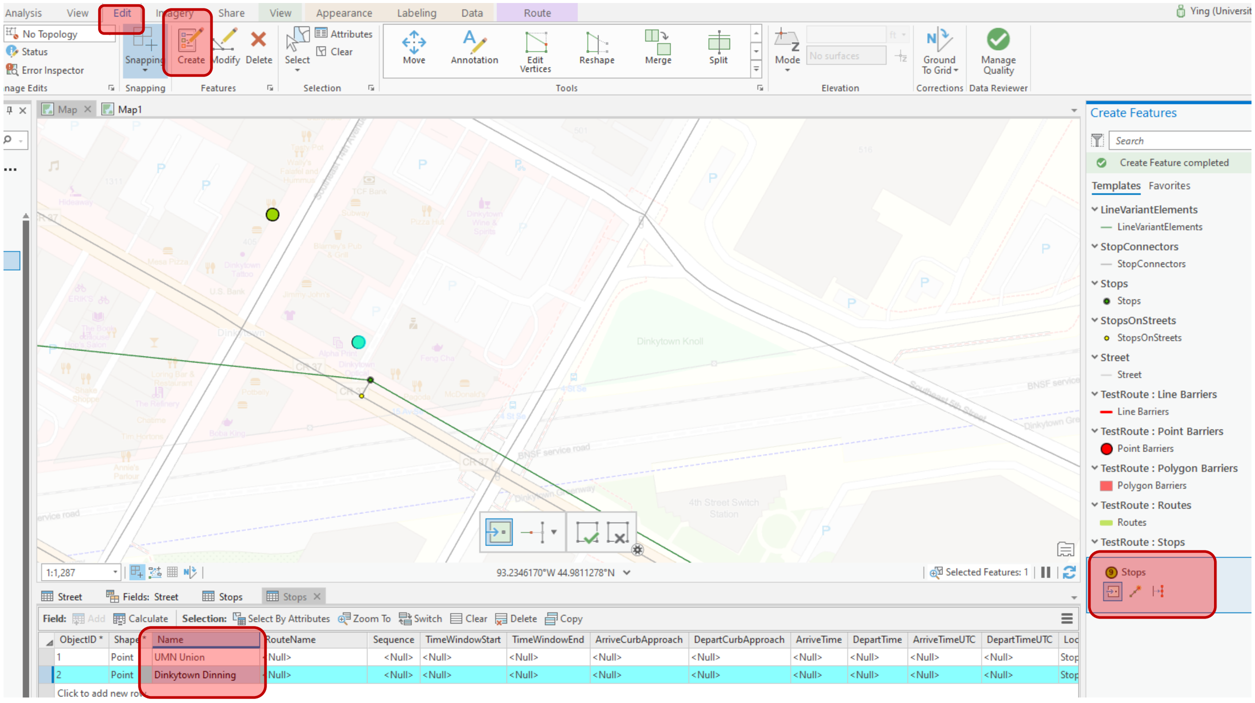
(4) Finally, right-click on "StreetGTFSND" in the feature dataset and choose "Build Network"

Part Four: Conduct a test analysis

(1) Make Route Analysis Layer using the StreetGTFSND instead of the ArcGIS online service.



(2) Create two points for origin and destination
- Origin: UMN Student Union
- Destination: Dinkytown Dinning



(2) Set analysis time as 11/17/2020, 11:00AM and get the travel time from Origin to Destination.