On Using Data Models in Travel Forecasting

David Ory
Presentation to Ohio Model User's Group
April 5, 2024

Agenda

- What?
- Why?
- How? (and Where?)
- Example: Ohio's Roadway Network Standard
- Questions/Discussions

What?

From Wikipedia:

An abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities.

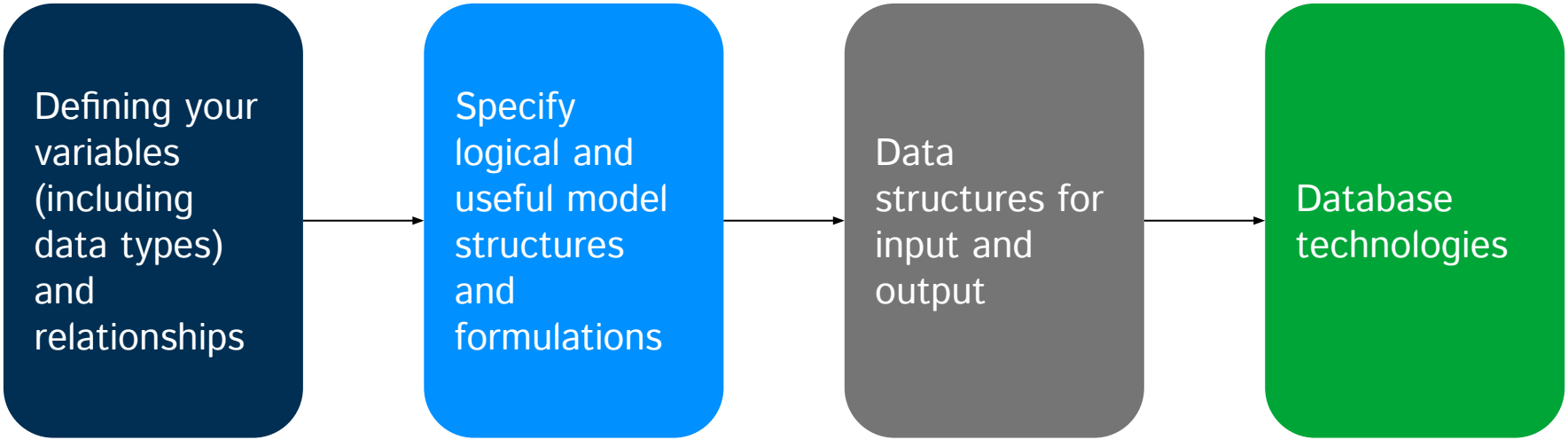
What?

```python
class TravelAnalysisZoneData(PydanticModel):
    """
    Space is segmented into discrete segments for use in travel model analysis. A travel analysis zone, or TAZ, is a unit of space.
    """

    id: PositiveInt
    """ ID
    Unique Identifier
    """

    private_dwellings: NonNegativeInt
    """ Private Dwellings
    A physical space where a single household resides. May be a standalone physical structure or part of a multi-unit structure.
    """

    employment_agriculture: NonNegativeFloat
    """ Employment in Agriculture Industry
    The number of people who work in a firm engaged in an agriculture-related business at this location on a typical weekday.
    """
```

What?

—

Data Model



What?

Why?

Documentation

Why?

Employment

Which is correct?

A. A firm-operated physical place identified in tax records as the employer of one or more worker.

B. A physical place identified as the “work location” in a survey response of one or more worker.

C. A firm-operated physical place to which one or more workers travels on a regular basis.

D. Something else?

2

Relationships

Why?

```python
@computed_field
@property
def employment_total(self) -> NonNegativeFloat:
    """ Total Employment
    The sum of employment across industry categories.
    """
    return (
        self.employment_agriculture
        + self.employment_mining
        + self.employment_construction
        + self.employment_manufacturing
        + self.employment_wholesale
        + self.employment_retail
        + self.employment_transport
        + self.employment_communication
        + self.employment_finance
        + self.employment_rental
        + self.employment_professional
        + self.employment_administrative
        + self.employment_education
        + self.employment_health
        + self.employment_social
        + self.employment_accommodation
        + self.employment_public_administration
        + self.employment_other
    )
```

Requiring users to sum fields is error prone, vague, and unnecessary.

Why?

—

Calculations that derive variables from other variables (e.g., total employment, household density) should be done in one and only one place.

Why?

```python
class Tour(TravelActivity):
    """ Tour
    A tour is a round-trip movement, with or without stops, between home or work and a primary destination.
    """

    purpose: e.Purpose
    """ Purpose
     A label defining the purpose of the tour, which is the activity that takes place
     at the primary destination.
    """

    return_to_origin: e.ModelTime
    """ Model Time
     The time of day category at which the traveler returns to the tour origin.
    """

    trips: List[Trip]
    """ Trips on the Tour
     A trip is a movement between two of the tour origin, primary tour destination, or an intermediate stop.
    """
```

Numerous relationships in activity-based models are implied by the model structures rather than explicitly defined.

Why?

—

3

~~Strings~~

Why?

```
class Mode(IntEnum):
    """ Mode
    Provides an integer mapping for travel mode.
    """

    DRIVE_ALONE = 1
    SHARED_RIDE_2 = 2
    SHARED_RIDE_3 = 3
    WALK = 4
    BICYCLE = 5
    WALK_TO_TRANSIT = 6
    PARK_AND_RIDE_TRANSIT= 7
    KISS_AND_RIDE_TRANSIT= 8
    SCHOOL_BUS = 9
```

Strings should never be used in travel models: they are error prone and have relatively large memory footprints.

Why?

—

4

Validation (a.k.a. Verification)

Why?

Verification

Data model software has verification tools that can be used to:

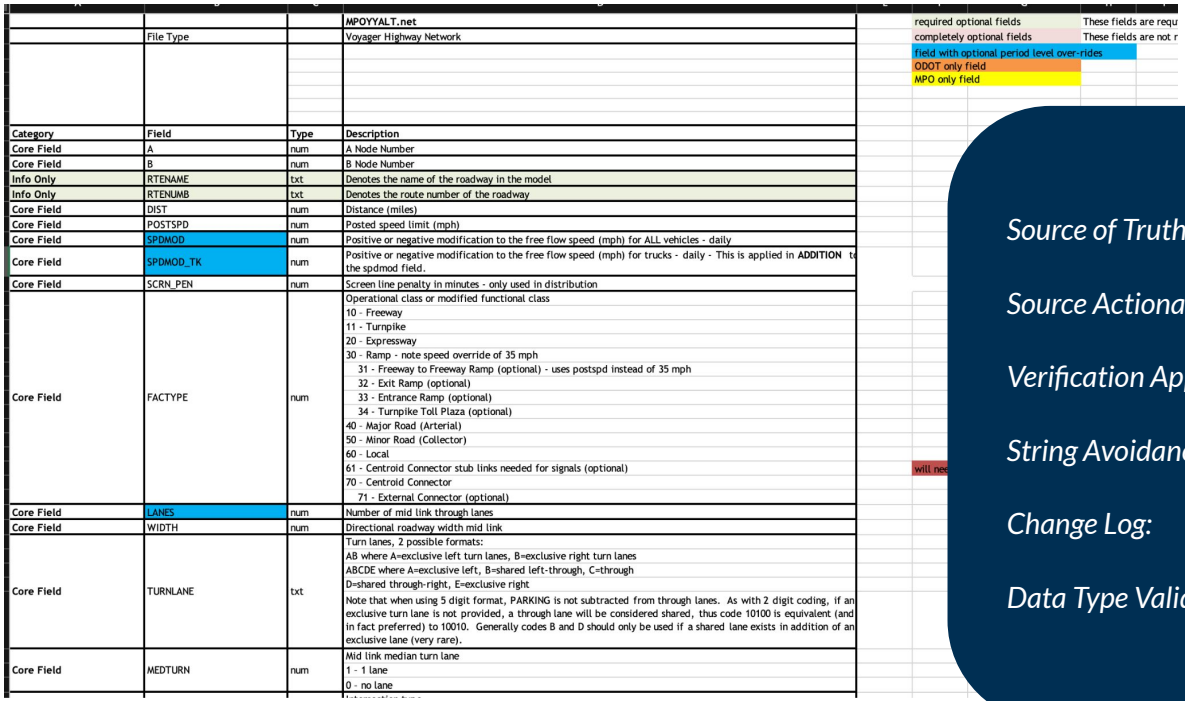
- Check variable names
- Check data types
- Check relationships, e.g., if households > 0, then household population > 0
- Compute “derived” variables, e.g., total employment, as needed

How? (and Where?)

Examples

- Overture Maps (JSON Schema)
- General Modeling Network Specification (JSON Schema)
- Main Roads W.A. PTM Prototype (Pydantic)
- Chandra Bot Project (Google Protocol Buffers)
- Agent (internal data model or “schema”)

Network Standard Example



			MPOYYALT.net
	File Type		Voyager Highway Network

Category	Field	Type	Description
Core Field	A	num	A Node Number
Core Field	B	num	B Node Number
Info Only	RTENAME	txt	Denotes the name of the roadway in the model
Info Only	RTENUMB	txt	Denotes the route number of the roadway
Core Field	DIST	num	Distance (miles)
Core Field	POSTSPD	num	Posted speed limit (mph)
Core Field	SPDMOD	num	Positive or negative modification to the free flow speed (mph) for ALL vehicles - daily
Core Field	SPDMOD_TK	num	Positive or negative modification to the free flow speed (mph) for trucks - daily - This is applied in **ADDITION** t the spdmod field.
Core Field	SCRN_PEN	num	Screen line penalty in minutes - only used in distribution
Core Field	FACTYPE	num	Operational class or modified functional class 10 - Freeway 11 - Turnpike 20 - Expressway 30 - Ramp - note speed override of 35 mph 31 - Freeway to Freeway Ramp (optional) - uses postspd instead of 35 mph 32 - Exit Ramp (optional) 33 - Entrance Ramp (optional) 34 - Turnpike Toll Plaza (optional) 40 - Major Road (Arterial) 50 - Minor Road (Collector) 60 - Local 61 - Centroid Connector stub links needed for signals (optional) 70 - Centroid Connector 71 - External Connector (optional)
Core Field	LANES	num	Number of mid link through lanes
Core Field	WIDTH	num	Directional roadway width mid link
Core Field	TURNLANE	txt	Turn lanes, 2 possible formats: AB where A=exclusive left turn lanes, B=exclusive right turn lanes ABCDE where A=exclusive left, B=shared left-through, C=through D=shared through-right, E=exclusive right Note that when using 5 digit format, PARKING is not subtracted from through lanes. As with 2 digit coding, if an exclusive turn lane is not provided, a through lane will be considered shared, thus code 10100 is equivalent (and in fact preferred) to 10010. Generally codes B and D should only be used if a shared lane exists in addition of an exclusive lane (very rare).
Core Field	MEDTURN	num	Mid link median turn lane 1 - 1 lane 0 - no lane

required optional fields	These fields are requ
completely optional fields	These fields are not r
field with optional period level over-rides	
ODOT only field	
MPO only field	

Source of Truth:	Excel
Source Actionable:	No
Verification Approach:	Mark B.
String Avoidance:	Int or short-string codes
Change Log:	Excel versions, manual diffs?
Data Type Validation:	Model run errors?

Network Standard Example

Pydantic

What would this look like?

```python
class RoadwayNetworkLinkAttribute(BaseModel):
  """ Roadway Network Link Attribute
  Defines a standard network link
  """

  category: Category
  """ Category
   An enum defining whether the attribute is required, optional, for information, etc.
   See enum.py for complete details
  """

  eight_char_name: Annotated[str, StringConstraints(max_length=8)]
  """ Eight Character Name
   A variable name limited to eight characters to use in select software programs that
   require FORTRAN-like variable names.
  """

  description: Annotated[str, StringConstraints(max_length=255)]
  """ Description
  A description of the variable.
  """
  ...
```

Network Example

```
class Category(IntEnum):
    """
    Category
    Provides integer mapping to the category of variable type.

    """

    CORE_FIELD = 1
    INFORMATION_ONLY = 2
    AUXILLIARY_USE = 3
    OVER_RIDE = 4
    VALIDATION_ONLY = 5
    MPO_USE = 6
```

Network Example

```python
class a_node(RoadwayNetworkLinkAttribute):
  """ A Node
  The node identifier at the beginning end of the link.
  """
  value: ClassVar[PositiveInt]
  def __str__(self):
       return standard_name

  category = Category.CORE_FIELD
  eight_char_name = "A_NODE"
  standard_name = "a_node"
  description = "A Node Number"
  varies_by_model_time_period= False
  ...
```

Network Example

```python
class RoadwayNetworkLink(BaseModel):
    """ Roadway Network Link
    A collection of roadway network attributes
    """

    a_node: ANode
    b_node: BNode
    distance: Distance
    posted_speed_limit: PostedSpeedLimit
    city_name: Optional[City]

    ...

class RoadwayNetwork(BaseModel):
    """ RoadwayNetwork
    Defines a roadway network standard
    """

    links: List[RoadwayNetworkLink]
    nodes: List[RoadwayNetworkNode]
```

Network Example

```
try:
   RoadwayNetwork(links = input_links, nodes = input_nodes)
   print("Verification successful")
except ValidationError as e:
   print(e)
```

Network Example

Source of Truth:	Excel
Source Actionable:	No
Verification Approach:	Manual Review?
String Avoidance:	Int or string codes
Change Log:	Excel versions, manual diffs?
Data Type Validation:	Model run errors?

Source of Truth:	Python
Source Actionable:	Yes
Verification Approach:	Automated
String Avoidance:	Enumerated variables
Change Log:	GitHub
Data Type Validation:	Python

Network Standard Example

—

Pros

1. Python — Yay!
2. Verification cost → zero (verification is *the* killer feature of data models)
3. Inconsistencies (definitions, data types, derived calculations) → zero
4. Variable codes (e.g., intersection type) are defined in one and only one place

Cons

1. Python — Ugh!
2. Excel version becomes “read only”
3. Change
4. Python code requires *continuous*, usually minor, maintenance
5. Higher upfront cost, including becoming familiar with Pydantic (or something similar)

Network Standard Example

Questions/Discussion