Urban Transportation Planning

Gregory Macfarlane, PhD, PE

2020-06-24

Contents

Foreword

This book contains course notes and assignments for a senior / graduate class in transportation planning and elementary travel modeling. A description for this course is:

> An advanced course in urban transportation planning. Urban transportation as the outcome of an economic system, details and techniques for four-step travel model development, applications of travel models within a legal and regulatory context.

The book is organized into six units:

1. Building Blocks
2. Trip Generation
3. Trip Distribution
4. Mode and Destination Choice
5. Network Assignment and Validation
6. The Planning Process

It may seem strange to put the chapter covering the planning process at the end of the course, after students have learned the details of quantitative travel modeling. The purpose for this is that I assign a term project where the students build and calibrate a four-step model as they learn the techniques to do so, and then complete an alternatives analysis using their models. To create the time and space to do this project, we cover "softer" and conceptual topics in the second half of the course.

The demonstration model the students calibrate and study is a model built in the Cube travel modeling software for the Roanoke, Virginia, metropolitan region. The model is a relatively advanced four-step, trip-based model with only 250 zones. The limited zone size means that the entire model system runs in approximately 15 minutes on a laptop computer. I am grateful to Virginia DOT for allowing my students the use of this model. Directions on how to use the Roanoke model are given in the Appendices.

A handful of assignments require the students to write numerical programs or estimate statistical models. Some guidance on using R and RStudio to accomplish these assignments is also given in the Appendices.

Acknowledgements

Photographs in the textbook are the work of the author unless otherwise attributed. The vector art in the textbook uses icons from FontAwesome and the Noun Project distributed under creative commons licenses. Specific attributions are below:

- training wheels by Marco Fleseri from the Noun Project
- Cover image: TRAX by Ashton Bingham on Unsplash

Chapter 1

Building Blocks

This chapter contains concepts, definitions, and mathematical techniques that will be used throughout the semester. Critical terms to understand are given in **bold**.

1.1 Planning for Human Systems

If you look out on any sufficiently busy road, you will see a steady stream of vehicles passing by. Each vehicle is largely indistinguishable from the others, and it is easy as an engineer responsible for that road to see the cars driving by as little more than an input to a problem. But the *people* inside the cars should not be indistinguishable from each other. Each person who is driving or riding in each of those cars has their own reasons to be driving on that road. One person might be driving to work; one person might be trying to get home to his or her family. Another car might hold a family going on vacation, or a group of friends heading to a movie.

If you don't recognize that each person who travels is different, with different needs and purposes, then it is easy to look only at the **supply** of transportation infrastructure. Is the road wide enough? Is the traffic signal timed appropriately? But as with anything in the economy transportation is a function of both supply and **demand**. Why are so many people trying to get down this one road *right now*? Why didn't more people take transit? Why didn't some people choose a different destination? Or why didn't some people just stay home in the first place?

Transportation planning therefore must be concerned with both the supply of infrastructure and the demand for travel. For the most part, economists consider travel a **derived demand**, which means people only go to the hassle of travelling somewhere if they have some other reason to be there. No one typically just drives around (with the possible exception of teenagers on a weekend

night); they are going to work, or school, or a social engagement, or *something*.

Travel demand has not been stable over time. The availability of inexpensive automobiles in the 20th Century created demand for inter-city and intra-urban roads that did not exist before. Rising labor force participation rates for women radically changed the number and types of trips the average household makes in an average day. Technological developments like teleconferencing and smartphone-enabled ridehailing could generate different trends. At the same time, populations in most regions continue to grow. **Planning** for future transportation infrastructure is difficult because of the uncertainty of the future, but it is necessary to keep economies rolling and preseve or improve quality of life.

In the United States and most societies with some democratic process, **decisions** about what transportation facilities to build, which policies to implement, and how to build a city generally fall to **decision makers**. These decision makers consist of mayors, city councils, planning commissions, state legislatures, Congress, state and federal agencies, and innumerable others who are elected by the public, or who are accountable to others who have been. In making decisions about how to spend public money on civil infrastructure or enact tax or other policies, decision makers consult **plans** developed by professional engineers and planners.

As engineers and planners, we are rarely in a position to *make* decisions, but we have a responsibility to provide accurate data and technical analysis to support decision makers. There is a misconception that transportation planners must accurately predict the future to be relevant. The purpose of transportation planning is not to perfectly envision what will happen under every scenario, it is to provide information that will help make good decisions now so that the future is at least as pleasant as the present. We all have hopes for what our lives and community will look like ten or twenty years from now; it may not be possible for anyone to provide analysis entirely free of all personal bias. But as you conduct your work as an engineer and planner, you owe the public your integrity and competence as you provide information to their representatives.

1.2 The Four-Step Process

1.3 Travel Model Building Blocks

In this section, we present some of the terms used in transportation planning and modeling, as well as some of the data objects used in constructing travel demand models.

1.3.1 Household Travel Surveys

Travel demand models try to represent individual behavior. How many trips does the average household make per day? How do people respond to changes

in transit fare? And how can a modeler know if the model accurately reflects total traffic?

Household travel surveys are a critical component of much travel modeling practice and research, and are a primary way to answer some of these questions. In a travel survey, a regional planning agency[1] will recruit households to participate in the survey. Often there is some kind of reward to encourage participation, like a gift card or raffle. Once recruited, household members fill out a diary of their activities on an assigned day; Figure 1.1 shows an example of one activity from a survey diary. From the example, you can see the kinds of data that are available: where the person traveled, which travel mode they used, and what was their reason for making the trip.

Not all travel surveys are filled in on forms; nowadays telephone interviews or mobile applications are more common (more on that below). But for decades, paper travel surveys were the basis of almost all transportation behavior science.

Once the surveys are collected, the data is usually processed into several tables stored in different files or a database.

- A **Households** table has one row for each household in the dataset, including information about the number of people in the household, the number of vehicles, and the household income.
- A **Persons** table has one row for each person in the dataset — including which household they are a part of (to link with the households table) — and personal attributes like age, student or worker status.
- A **Vehicles** table has one row for each vehicle owned by the households in in the dataset, including attributes like model year, vehicle class, and fuel efficiency.
- A **Trips** table has one row for each trip taken by each person in the dataset. This table can be linked against the other tables if necessary, and contains information like the trip purpose and many other elements collected with the form in Figure 1.1.

Tables 1.1 through 1.4 show data collected from one household in the 2017 National Household Travel Survey. The household contains four people, two of whom are working adults in their late thirties. (the other two are children, and the NHTS did not collect their trip data). The household has two vehicles, and on the survey travel day person 2 appeared to make a few very long trips. It's impossible to know if this is a typical day for this person or not, but that's the data that was collected.

Note that that the households data in Table 1.1 contains a numeric column called `wthhfin`. This is a survey *weight*. Because it is impossible to sample everyone in a population, there needs to be a way to *expand* the survey to the population. What this number means is that the selected household carries the same *weight* in this survey as approximately 1100 households in the general

[1] Like a Metropolitan Planning Organization (MPO).

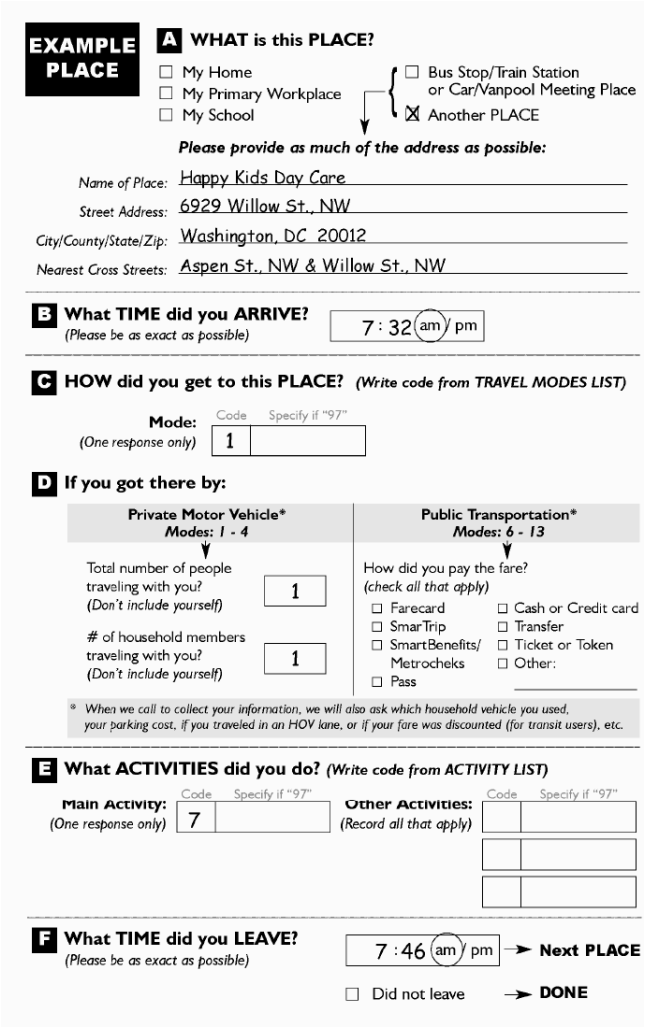


Figure 1.1: Example travel survey diary entry.

Table 1.1: NHTS Households File

houseid	hhsize	numadlt	wrkcount	hhvehcnt	hhfaminc	wthhfin
30000082	4	2	2	2	$100,000 to $124,999	1148.809

Table 1.2: NHTS Persons File

houseid	personid	r_age	educ	r_sex
30000082	01	39	Graduate degree or professional degree	Female
30000082	02	38	Bachelor's degree	Male

Table 1.3: NHTS Vehicles File

houseid	vehid	vehyear	make	model	fueltype	od_read
30000082	01	2011	Mazda	Mazda3	Gas	83644
30000082	02	2007	Toyota	Yaris	Gas	120615

Table 1.4: NHTS Trips File

houseid	personid	strttime	endtime	trpmiles	trptrans	trippurp
30000082	01	2017-10-10 07:45:00	2017-10-10 07:52:00	2.710	Car	Home-based trip (other)
30000082	01	2017-10-10 08:09:00	2017-10-10 08:13:00	1.432	Car	Not a home-based trip
30000082	01	2017-10-10 08:24:00	2017-10-10 08:28:00	0.777	Car	Not a home-based trip
30000082	01	2017-10-10 16:53:00	2017-10-10 16:57:00	1.075	Car	Not a home-based trip
30000082	01	2017-10-10 17:18:00	2017-10-10 17:26:00	2.727	Car	Home-based trip (other)
30000082	02	2017-10-10 07:30:00	2017-10-10 07:33:00	2.136	Car	Home-based trip (shoppi
30000082	02	2017-10-10 07:38:00	2017-10-10 08:50:00	88.581	Car	Not a home-based trip
30000082	02	2017-10-10 08:58:00	2017-10-10 09:49:00	45.341	Car	Not a home-based trip
30000082	02	2017-10-10 10:51:00	2017-10-10 12:24:00	28.208	Car	Not a home-based trip
30000082	02	2017-10-10 17:00:00	2017-10-10 17:05:00	0.239	Walk	Not a home-based trip
30000082	02	2017-10-10 19:15:00	2017-10-10 19:26:00	0.267	Walk	Not a home-based trip
30000082	02	2017-10-10 19:30:00	2017-10-10 20:43:00	29.293	Car	Not a home-based trip

population. Also note that not every household's weight will be equal; because some population groups have different survey response weights, some households will need to be weighted more heavily so that the survey reflects the general population. Most software packages have functions that allow you to calculate statistics or estimate models including weighted values. The code chunk below shows how to calculate the average number of workers per household with and without weights in R; as you can see, omitting the weights leads to a substantial change in the survey analysis.

```
# Average workers per household with no weights
mean(nhts_households$wrkcount)
```

```
## [1] 0.9891438
```

```
# Average workers per household, weighted
weighted.mean(nhts_households$wrkcount, nhts_households$wthhfin)
```

```
## [1] 1.173206
```

Travel survey methodology is changing rapidly as a result of mobile devices with location capabilities. First, most travel surveys are now administered through a mobile application: respondents are invited to install an app on their smartphone that tracks the respondent's position and occasionally asks questions about trip purpose or mode. This makes collecting and cleaning data consider-

ably easier than traditional paper surveys, and it also lowers the response burden for the survey participants. Another change that mobile data has brought to travel surveys is the introduction of large datasets of location information that planners can purchase directly from cellular providers or third-party providers. Though these data do not have all the information on demographics and preferences a survey would provide, they provide a considerably larger and more detailed sample on things like overall trip flows. As a result, it may be possible to collect surveys less frequently, or to reduce survey sample sizes.

1.3.2 Travel Analysis Zones

Activities in travel demand models happen in **Travel Analysis Zones** (TAZs), and the model tries to represent trips between the TAZs. Because trips inside a TAZ — called intrazonal trips — are not included in the travel model, each TAZ should be sufficiently small such that these trips do not affect the models' ability to forecast travel on roadways. The following rules are helpful when drawing TAZ's:

- The TAZ should not stretch across major roadways
- The TAZ should contain principally one land use, though in some areas this is not possible.
- In areas with more dense population, the TAZ should be smaller.

Each TAZ is associated with **socioeconomic** (SE) data, or information about the people, businesses, and other activities that are located in the TAZ.

Households are a basic unit of analysis in many economic and statistical analyses. A household typically consists of one or more **persons** who reside in the same dwelling. Individuals living in the same dwelling can make up a family or other group of individuals; that is, a group of roommates is considered a household. Not everyone lives in households, however; some people live in what are called group quarters: military barracks, college dormitories, monasteries, prisons, etc. Travel models need to handle these people as well, but in this class we will focus on people who live in households.

Firms are another basic unit of analysis in many economic and statistical analyses. A firm is a profit-seeking person or entity that provides goods or services in exchange for monetary transactions. A firm can provide raw resources, manufactured resources, other services, or be a place of employment. In some cases, a firm may be another household. Each firm will have an *industry* type. Examples of industry types include office, service, manufacturing, retail, etc. In many SE data files, firms are simply represented as the total number of jobs in a TAZ belonging to each industry. Other **Institutions** including academic, government, and non-profit entities will also be represented in the SE data in terms of their jobs.

It is important to be precise in our definitions when put all of these different things into a single file. A typical SE table for a small region is given in Table

Table 1.5: Example SE Table

taz	persons	hh	workers	retail	office	manufacturing
1	32	37	9	111	73	2
2	36	36	16	132	54	11
3	47	42	15	138	60	0

1.5. Note the following relationships:

- Persons live in Households
- Workers are Persons who have a Job
- Firms have employees who work at a Jobs

When we talk about "how many jobs" are in a TAZ, we mean "How many people do the firms located in that TAZ employ," and not "how many people who live in that TAZ are workers."

1.3.3 Highway Networks

Nodes Centroids are special nodes that indicate where the activities of a TAZ are located on average.

Links Centroid connectors are special links that connect centroids to a network.

Functional Types or **Functional Classes** are used to describe each road in a system, and its importance to that system. The types include: freeway, principal arterial, minor arterial, major collector, minor collector, local street, and cul-de-sac.

- Freeways are provided almost exclusively to enhance mobility for through traffic. Access to freeways is provided only at specific grade-separated interhcanges, with no direct access to the freeway from adjacent land except by way of those interachanges.
- Major and minor arterials primary function is to provide mobility of through traffic. However, arterials also connect to both collectors and local roads and streets and many arteirals provide direct access to adjacent development.
- Major and minor collectors connect arterials to local roads and provide access to adjacent development. Mobility for through traffic is less important.
- Local streets exist primarily to serve adjacent development. Mobility for thorough traffic is not important.
- Cul-de-sacs only serve adjacent development.

*See (2018) for more information.

Streets of a functional class below collector are almost never included in travel models, unless they provide essential connectivity between other roads. Entire neighborhoods of local streets may be represented by just a few centroid connectors.

Free-flow speed, or **FFS**, is the speed vehicles travel when the road is empty.

Link capacity is the maximum number of vehicles a *link*, or section of road, can optimally transport between two *nodes*. The capacity is a function of functional type, lanes, free-flow speed, area type, etc.

1.3.4 Matrices

Skim matrices, or skims, are matrices of impedance estimates between zones, used to estimate zone to zone travel demand. Impedance typically measures a travel time, distance, and travel costs. These measures can be combined to create generalized costs. Skims often distinguish between single occupancy and shared ride vehicles, though this ability is not always beneficial.

OD matrices, or origin-destination matrices, represent the number of trips from the origin i (row) to the destination j (column). The number in the corresponding cell T_{ij} is the total number of trips made, and represents the demand between two zones in a network.

1.4 Statistical and Mathematical Techniques

Many elements of travel modeling and forecasting require complex numerical and quantitative techniques. In this section we will present some of these techniques.

1.4.1 Continuous and Discrete Distributions

In general, statistical variables can fall into one of two categories:

- *Continuous* variables can take any numeric value along some range
- *Discrete* variables can take some limited set of predetermined values

A simplistic definition would be to say that continuous variables are numeric and discrete variables are non-numeric. A continuous variable has statistics such as a *mean*, but these statistics do not make sense on discrete variables. In the NHTS trips dataset, we can compute a mean trip miles, but we cannot compute a mean trip purpose. Or we can't compute a mean that makes sense.

```
# mean of continuous variable: trip length
weighted.mean(nhts_trips$trpmiles, nhts_trips$wttrdfin)
```

```
## [1] 10.69119
```

```
# mean of categorical variable: trip purpose
weighted.mean(nhts_trips$trippurp, nhts_trips$wttrdfin)
```

```
## Error in x * w: non-numeric argument to binary operator
```

What we can do, however, is we can print a summary table showing the number of observations that fit in each trip purpose category. Note that sometimes there will be a category devoted to data that is missing or otherwise invalid.

```
table(nhts_trips$trippurp)
```

```
##
##       -9      HBO    HBSHOP HBSOCREC      HBW      NHB
##       32   190022   195188   110235   117368   310727
```

Sometimes it is handy to split a continuous variable into categories so that you can treat it as a discrete variable.

```
nhts_trips$miles_cat <- cut(nhts_trips$trpmiles, breaks = c(0, 10, 20, 30, 50, 100, Inf))
table(nhts_trips$miles_cat)
```

```
##
##    (0,10]   (10,20]   (20,30]   (30,50]  (50,100] (100,Inf]
##    719812    113383     38724     25064     14388     11060
```

When we visualize the distribution of a continuous variable, we might use a histogram or density plot, but with a discrete variable we would use a bar chart.

```
ggplot(nhts_trips, aes(x = trpmiles, weight = wttrdfin)) +
  geom_histogram() + xlab("Trip Distance [Miles]") + ylab("Weighted Trips") +
  scale_x_continuous(limits = c(0, 50))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
ggplot(nhts_trips, aes(x = as_factor(trippurp, levels = "labels"),
                       weight = wttrdfin)) +
  geom_bar() + xlab("Trip Purpose") + ylab("Weighted Trips")
```

To this point we've only looked at the distribution of one variable at a time. There are lots of cases where someone might want to consider the *joint* distribution of two variables. This joint distribution tells you what is happening with one variable while the other variable changes. In a table like the one below, the margins of the table (the row and column sums) contain the single variable distribution. So sometimes we call these the *marginal* distributions.

```
##
##                -9    HBO HBSHOP HBSOCREC    HBW    NHB
##   (0,10]       23 156315 162602    84980  67162 248730
##   (10,20]       6  20856  19881    13054  28018  31568
```

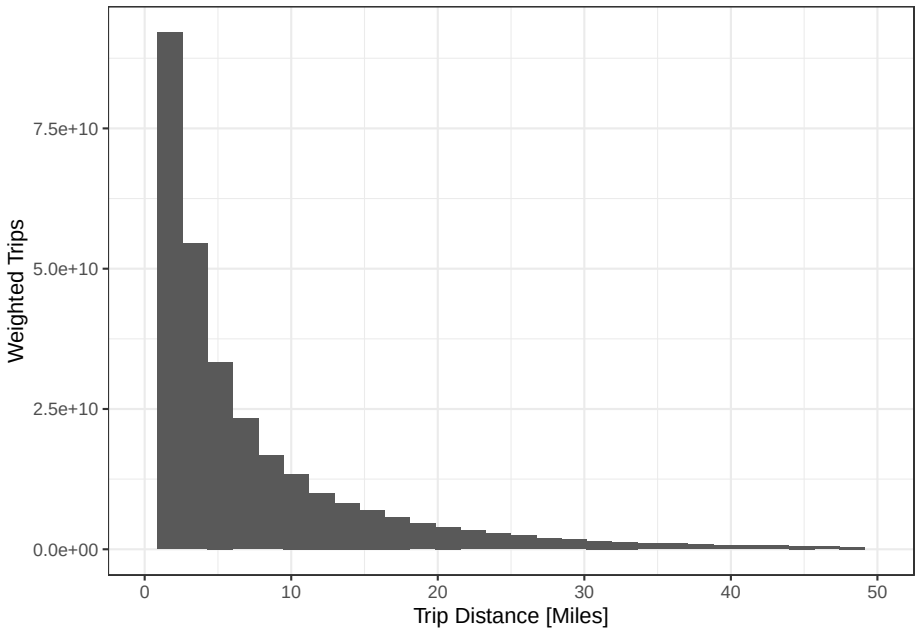


Figure 1.2: Visualizing a continuous distribution with a histogram.

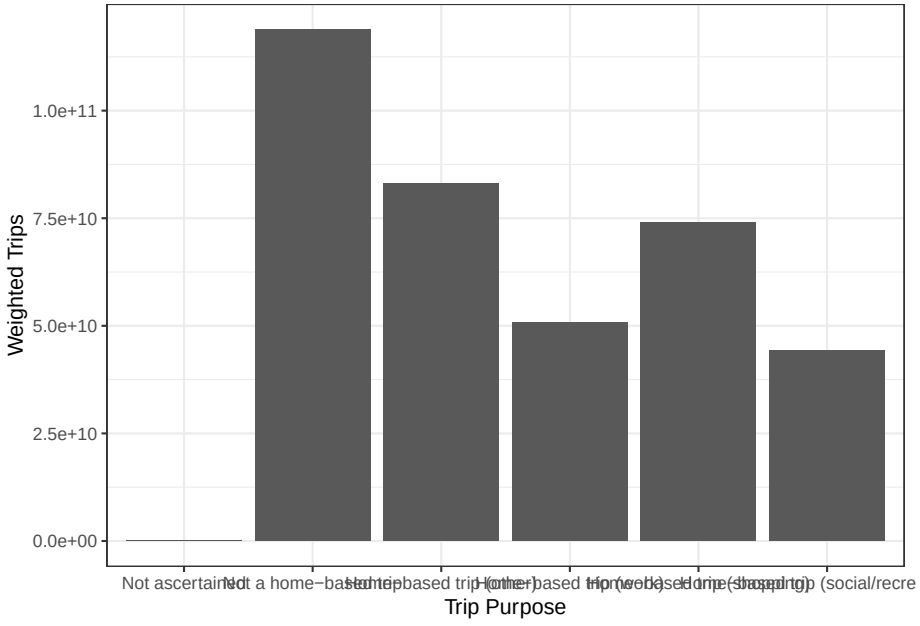
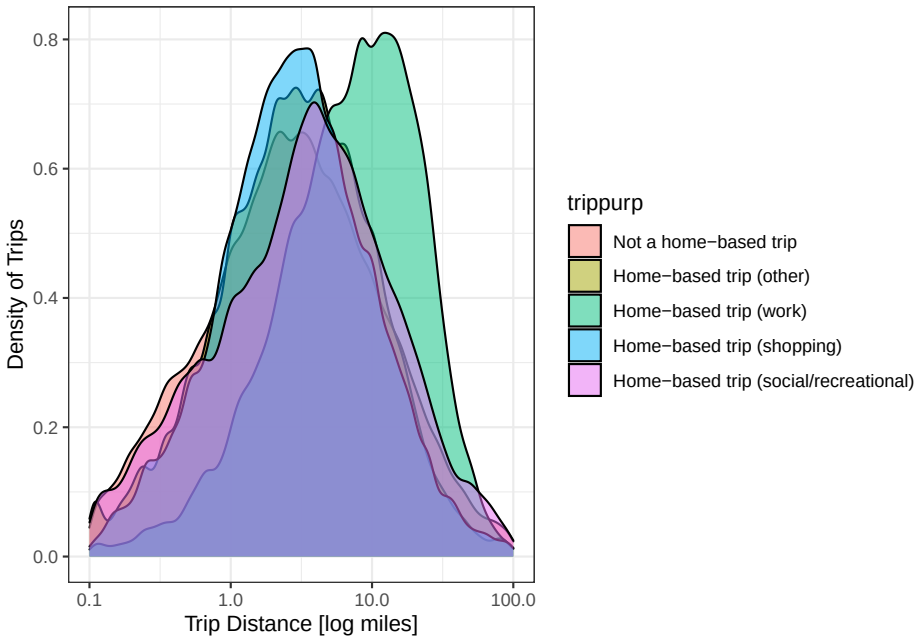


Figure 1.3: Visualizing a discrete distribution with a bar chart.

```
##    (20,30]         1   5635   5469     4361  12087  11171
##    (30,50]         0   3592   3427     3267   7117   7661
##    (50,100]        2   1943   2150     2504   2332   5457
##    (100,Inf]       0   1231   1634     1844    597   5754
```

We can visualize joint distributions as well, and sometimes the results are quite nice.



1.4.2 Iterative Proportional Fitting

Iterative Proportional Fitting

1.4.3 Regression Analysis

We often want to know what will happen

1.4.4 Numerical Optimization

Let's say you have a function with a

Homework

> Some of these questions require a completed run of the demonstration model. For instructions on accessing and running the model, see the Appendix

1. How does recreational transportation — i.e., going for a bike ride — fit into the theory of derived demand for travel? Write a short paragraph explaining your thoughts based on what we covered in lecture and the text.

2. Think about a recent transportation-related construction project you have seen in your community. Find an article in a local newspaper discussing the project. Why was the project built (or why is it being built)? Who supports the project? Does anyone oppose the project? Write a short paragraph including a link and citation to the article.

3. Download the rmove mobile application, and log in with the password given you by the instructor. Track your daily activies and trips for *three days*. You may include at most one weekend day. Write a short summary of your activities, including:

   - How many trips you took each day
   - The mode split of all your trips

4. With the TAZ layer and socioeconomic data in the demonstration model, make a set of choropleth maps showing: total households; household density; total jobs; job density; density of manufacturing vs office vs retail employment. Compare your maps with aerial imagery from Google Maps or OpenStreetMap. Describe the spatial patterns of the socioeconomic data in the model region. Identify which zones constitute the central business district, and identify any outlying employment centers.

5. With the highway network layer in the demonstration model, create maps showing: link functional type; link free flow speed; and link hourly capacity. Compare your maps with aerial imagery from Google Maps or OpenStreetMap. Identify the major freeways and principal arterials in the model region. *Note*: you will need to run the demonstration model through the network setup step to calculate the capacities and append them to the link.

6. Find the shortest free-flow speed path along the network between two zones. Find the shortest distance path between the same two zones. Are the paths the same? Do the paths match what an online mapping service shows for a trip in the middle of the night?

7. Open the highway assignment report, which shows vehicle hours and miles traveled by facility type. What percent of the region's VMT occurs on freeways? What percent of the region's lane-miles are freeways?

8. Open the output highway network. Create a map of the highway links showing PM period level of service based on the volume to capacity ratios in the table below. How would you characterize traffic in Roanoke? Which is the worst-performing major facility?

Chapter 2

Trip Generation

2.1 Trip Production

2.2 Trip Attraction

Homework

Chapter 3

Trip Distribution

Chapter 4

Mode and Destination Choice

Chapter 5

Network Assignment and Validation

Chapter 6

The Planning Process

Appendix A

Demonstration Model

A.1 Running the Model

A.2 Files and Reports

A.3 Cube Tips and Tricks

A.3.1 Shortest Paths

A.3.2 Working with Matrices

A.3.3 Writing Custom Scripts

Appendix B

R and RStudio Help

R is a powerful, open-source statistical programming language used by both professional and academic data scientists. It is among the computer languages most suited to modern data science, and is growing rapidly in its user base and available packages.

Some students may not feel comfortable working in a programming language like R or a console-based application like RStudio, especially if they have used applications primarily through a GUI. This appendix provides a basic bootcamp for R and Rstudio, but cannot be a comprehensive manual on RStudio, and it certainly cannot be one for R. Good places to get more detailed help include:

- R help manuals
- Stack Overflow

Some of the sections in this appendix are text-based, and some contain little more than links to YouTube videos created by me or someone else.

B.1 Installing

There are two pieces of software you should install:

- **R** https://cran.r-project.org/: this contains the system libraries necessary to run R commands in a terminal on your computer, and contains a few additional helper applications. Install the most recent stable release for your operating system.
- **RStudio** https://rstudio.com/products/rstudio/download/ is an integrated application that makes using R considerably easier with text completion, file managment, and some GUI features.

Both software are available for Windows, MacOS, and Linux. The videos and screenshots of the application I post will use MacOS; the R code for all systems

is the same, and the RStudio interface all systems is very similar with minor differences.

B.2 RStudio Orientation

The video below gives a very basic introduction to RStudio. There is also a very useful cheat sheet for working with RStudio on the Rstudio website.

B.3 R Packages

One of the strengths of R is the ability for anyone to write packages. These packages make it easier to read manipulate, and vizualize data; to estimate statistical models; or to communicate results.

There are a number of ways to install additional packages. The most straightforward is to use the `install.packages()` function in the console. The problems in this book are solved with two additional packages[1]:

```
install.packages("tidyverse") # a suite of tools for data manipulation
install.packages("mlogit") # discrete choice modeling
```

RStudio also contains a GUI interface to install and update packages.

[1] `tidyverse` is actually a collection of very useful packages, and many R users just load them all at once.

Sometimes you want to use a package that has not yet been pushed to CRAN, the international repository of "approved" R packages. This may be because the package is in development, or for one reason or another does not meet CRAN's standards for completeness, etc. Oftentimes, the package has been made available on GitHub. You can install a package directly from GitHub with the `remotes` library. One package you will want for the problems in the book is the `nhts2017` package on the BYU Transportation GitHub account. This package contains datasets from the 2017 National Household Travel Survey.

```
install.packages("remotes") # tools for installing development packages
remotes::install_github("byu-transpolab/nhts2017")
```

You only need to **install** a package once on your computer. But every time you want to **use** a function in a package, you need to load the package with the `library()` function. To load the `tidyverse` packages, for instance,

```
library(tidyverse)
```

If you get errors when you run the command above, it means that for some reason you did not install the package correctly. And if you ever get an error like

```
kable(tibble(x = 1:2, y = c("blue", "red")))
```

x	y
1	blue
2	red

It often means you didn't load the library. In this case, the `kable()` function to make pretty tables is part of the `knitr` package.

```
library(knitr)
kable(tibble(x = 1:2, y = c("blue", "red")))
```

x	y
1	blue
2	red

You can also use a function from a package without loading the library if you use the `::` operator, like you did in the `remotes::install_github()` command earlier. This is handy if you only want to use one function from a package, or if you have two functions from different packages with the same name. For example, when you loaded the `tidyverse` package, R told you that `dplyr::filter()` would mask `stats::filter()`. So if for some reason you wanted to use the `filter` function from the `stats` package, you would need to use `stats::filter()`.

B.4 Working with Tables

Most data you will work with comes in a *tabular* form, meaning that the data is formatted in columns of variables and rows of observations.

B.4.1 Reading Data

Tabular data is often stored in a comma-separated values `.csv` file. To read a data file like this in R, you can use the `read_csv()` function included in `tidyverse`.

```
trips <- read_csv("data/demo_trips.csv")
```

```
## Parsed with column specification:
## cols(
##   houseid = col_double(),
##   personid = col_character(),
##   trpmiles = col_double(),
##   trippurp = col_character()
## )
```

```
print(trips)
```

```
## # A tibble: 924 x 4
##     houseid personid trpmiles trippurp
##       <dbl> <chr>       <dbl> <chr>
##  1 30182694 01          13.6   HBW
##  2 40532989 02           9.38  HBSOCREC
##  3 40729475 01           5.06  HBSHOP
##  4 40290784 01           0.509 HBSOCREC
##  5 30118876 02           0.599 NHB
##  6 30352119 01          20.3   HBO
##  7 30085077 01          22.5   NHB
##  8 30180962 02           0.581 HBSOCREC
##  9 40155356 01           2.74  HBO
## 10 30069734 01           4.65  NHB
## # ... with 914 more rows
```

This function will make a guess as to what the columns types should be. Often we want to keep ID values as characters, even if they are numeric (this preserves leading `0` values, etc.). We can tell `read_csv()` what types we expect with the `col_types` argument.

```
trips <- read_csv("data/demo_trips.csv", col_types = list(houseid = col_character()))
print(trips)
```

```
## # A tibble: 924 x 4
##    houseid  personid trpmiles trippurp
##    <chr>    <chr>       <dbl> <chr>
```

```
##  1 30182694 01         13.6   HBW
##  2 40532989 02          9.38  HBSOCREC
##  3 40729475 01          5.06  HBSHOP
##  4 40290784 01          0.509 HBSOCREC
##  5 30118876 02          0.599 NHB
##  6 30352119 01         20.3   HBO
##  7 30085077 01         22.5   NHB
##  8 30180962 02          0.581 HBSOCREC
##  9 40155356 01          2.74  HBO
## 10 30069734 01          4.65  NHB
## # ... with 914 more rows
```

You can also write tables back to `.csv` with the `write_csv()` command.

B.4.2 Modifying and Summarizing Tables

In much of this section, we will work with the `nhts_trips` dataset of trips from the 2017 National Household Travel Survey in the `nhts2017` package you installed from GitHub above.

```
library(nhts2017)
trips <- nhts_trips
trips
```

```
## # A tibble: 923,572 x 62
##    houseid personid tdtrpnum strttime            endtime
##    <chr>   <chr>       <dbl> <dttm>              <dttm>
##  1 300000~ 01              1 2017-10-10 10:00:00 2017-10-10 10:15:00
##  2 300000~ 01              2 2017-10-10 15:10:00 2017-10-10 15:30:00
##  3 300000~ 02              1 2017-10-10 07:00:00 2017-10-10 09:00:00
##  4 300000~ 02              2 2017-10-10 18:00:00 2017-10-10 20:30:00
##  5 300000~ 03              1 2017-10-10 08:45:00 2017-10-10 09:00:00
##  6 300000~ 03              2 2017-10-10 14:30:00 2017-10-10 14:45:00
##  7 300000~ 01              1 2017-10-10 11:15:00 2017-10-10 11:30:00
##  8 300000~ 01              2 2017-10-10 23:30:00 2017-10-10 23:40:00
##  9 300000~ 01              1 2017-10-10 05:50:00 2017-10-10 06:05:00
## 10 300000~ 01              2 2017-10-10 07:00:00 2017-10-10 07:15:00
## # ... with 923,562 more rows, and 57 more variables: trvlcmin <dbl+lbl>,
## #   trpmiles <dbl+lbl>, trptrans <chr+lbl>, trpaccmp <dbl+lbl>,
## #   trphhacc <dbl+lbl>, vehid <chr+lbl>, trwaittm <dbl+lbl>,
## #   numtrans <dbl+lbl>, tracctm <dbl+lbl>, drop_prk <chr+lbl>,
## #   tregrtm <dbl+lbl>, whodrove <chr+lbl>, whyfrom <chr+lbl>,
## #   loop_trip <chr+lbl>, trphhveh <chr+lbl>, hhmemdrv <chr+lbl>,
## #   hh_ontd <dbl+lbl>, nonhhcnt <dbl+lbl>, numontrp <dbl+lbl>,
## #   psgr_flg <chr+lbl>, pubtrans <chr+lbl>, trippurp <chr+lbl>,
## #   dweltime <dbl+lbl>, tdwknd <chr+lbl>, vmt_mile <dbl+lbl>,
## #   drvr_flg <chr+lbl>, whytrp1s <chr+lbl>, ontd_p1 <chr+lbl>,
```

```
## #   ontd_p2 <chr+lbl>, ontd_p3 <chr+lbl>, ontd_p4 <chr+lbl>,
## #   ontd_p5 <chr+lbl>, ontd_p6 <chr+lbl>, ontd_p7 <chr+lbl>,
## #   ontd_p8 <chr+lbl>, ontd_p9 <chr+lbl>, ontd_p10 <chr+lbl>,
## #   ontd_p11 <chr+lbl>, ontd_p12 <chr+lbl>, ontd_p13 <chr+lbl>,
## #   tdcaseid <chr>, tracc_wlk <chr+lbl>, tracc_pov <chr+lbl>,
## #   tracc_bus <chr+lbl>, tracc_crl <chr+lbl>, tracc_sub <chr+lbl>,
## #   tracc_oth <chr+lbl>, tregr_wlk <chr+lbl>, tregr_pov <chr+lbl>,
## #   tregr_bus <chr+lbl>, tregr_crl <chr+lbl>, tregr_sub <chr+lbl>,
## #   tregr_oth <chr+lbl>, whyto <chr+lbl>, gasprice <chr>, wttrdfin <dbl>,
## #   whytrp90 <chr+lbl>
```

B.4.2.1 Select, Filter, and Chains

This table is pretty overwhelming. But there are two functions that can help us pare it down:

- `select()` lets you select columns in a table using the names of the columns.
- `filter()` lets you select rows in a table that meet a certain condition.

Let's practice this by selecting our `trips` dataset to only include the id columns, the trip length, and the trip purpose.

```
select(trips, houseid, personid, trpmiles, trippurp)
```

```
## # A tibble: 923,572 x 4
##    houseid  personid  trpmiles trippurp
##    <chr>    <chr>    <dbl+lbl> <chr+lbl>
##  1 30000007 01            5.24 HBO [Home-based trip (other)]
##  2 30000007 01            5.15 HBO [Home-based trip (other)]
##  3 30000007 02           84.0  HBW [Home-based trip (work)]
##  4 30000007 02           81.6  HBW [Home-based trip (work)]
##  5 30000007 03            2.25 HBO [Home-based trip (other)]
##  6 30000007 03            2.24 HBO [Home-based trip (other)]
##  7 30000008 01            8.02 HBW [Home-based trip (work)]
##  8 30000008 01            8.02 HBW [Home-based trip (work)]
##  9 30000012 01            3.40 HBSOCREC [Home-based trip (social/recreatio~
## 10 30000012 01            3.40 HBSOCREC [Home-based trip (social/recreatio~
## # ... with 923,562 more rows
```

Let's also practice filtering the `trips` dataset to only include trips of the purpose "HBO" (home-based other). Notice how the number of rows in the table trips is much smaller.

```
filter(trips, trippurp == "HBW") # use double equals as comparison
```

```
## # A tibble: 117,368 x 62
##    houseid personid tdtrpnum strttime           endtime
##    <chr>   <chr>       <dbl> <dttm>             <dttm>
```

```
##  1 300000~ 02              1 2017-10-10 07:00:00 2017-10-10 09:00:00
##  2 300000~ 02              2 2017-10-10 18:00:00 2017-10-10 20:30:00
##  3 300000~ 01              1 2017-10-10 11:15:00 2017-10-10 11:30:00
##  4 300000~ 01              2 2017-10-10 23:30:00 2017-10-10 23:40:00
##  5 300000~ 01              5 2017-10-10 09:00:00 2017-10-10 09:20:00
##  6 300000~ 01              7 2017-10-10 15:30:00 2017-10-10 16:05:00
##  7 300000~ 01              1 2017-10-10 08:00:00 2017-10-10 08:20:00
##  8 300000~ 01              2 2017-10-10 18:00:00 2017-10-10 20:00:00
##  9 300000~ 02              3 2017-10-10 09:00:00 2017-10-10 11:00:00
## 10 300000~ 02              4 2017-10-10 18:30:00 2017-10-10 20:30:00
## # ... with 117,358 more rows, and 57 more variables: trvlcmin <dbl+lbl>,
## #   trpmiles <dbl+lbl>, trptrans <chr+lbl>, trpaccmp <dbl+lbl>,
## #   trphhacc <dbl+lbl>, vehid <chr+lbl>, trwaittm <dbl+lbl>,
## #   numtrans <dbl+lbl>, tracctm <dbl+lbl>, drop_prk <chr+lbl>,
## #   tregrtm <dbl+lbl>, whodrove <chr+lbl>, whyfrom <chr+lbl>,
## #   loop_trip <chr+lbl>, trphhveh <chr+lbl>, hhmemdrv <chr+lbl>,
## #   hh_ontd <dbl+lbl>, nonhhcnt <dbl+lbl>, numontrp <dbl+lbl>,
## #   psgr_flg <chr+lbl>, pubtrans <chr+lbl>, trippurp <chr+lbl>,
## #   dweltime <dbl+lbl>, tdwknd <chr+lbl>, vmt_mile <dbl+lbl>,
## #   drvr_flg <chr+lbl>, whytrp1s <chr+lbl>, ontd_p1 <chr+lbl>,
## #   ontd_p2 <chr+lbl>, ontd_p3 <chr+lbl>, ontd_p4 <chr+lbl>,
## #   ontd_p5 <chr+lbl>, ontd_p6 <chr+lbl>, ontd_p7 <chr+lbl>,
## #   ontd_p8 <chr+lbl>, ontd_p9 <chr+lbl>, ontd_p10 <chr+lbl>,
## #   ontd_p11 <chr+lbl>, ontd_p12 <chr+lbl>, ontd_p13 <chr+lbl>,
## #   tdcaseid <chr>, tracc_wlk <chr+lbl>, tracc_pov <chr+lbl>,
## #   tracc_bus <chr+lbl>, tracc_crl <chr+lbl>, tracc_sub <chr+lbl>,
## #   tracc_oth <chr+lbl>, tregr_wlk <chr+lbl>, tregr_pov <chr+lbl>,
## #   tregr_bus <chr+lbl>, tregr_crl <chr+lbl>, tregr_sub <chr+lbl>,
## #   tregr_oth <chr+lbl>, whyto <chr+lbl>, gasprice <chr>, wttrdfin <dbl>,
## #   whytrp90 <chr+lbl>
```

One ***extremely*** useful feature of the `tidyverse` functions is the chain operator, `%>%`. This operator basically does the opposite of the assigment operator `<-`. While assignment says "take the thing on the right and put it in the thing on the left," chain says "take the thing on the left and pass it as the first argument of the function on the right." What this means in practice is we can chain R commands together. So we can do the `select` *and* the `filter` statements in sequence,

```
trips %>%
  select(houseid, personid, trpmiles, trippurp) %>%
  filter(trippurp == "HBW")
```

```
## # A tibble: 117,368 x 4
##    houseid  personid  trpmiles trippurp
##    <chr>    <chr>    <dbl+lbl> <chr+lbl>
##  1 30000007 02           84.0  HBW [Home-based trip (work)]
```

```
##  2 30000007 02           81.6  HBW [Home-based trip (work)]
##  3 30000008 01            8.02 HBW [Home-based trip (work)]
##  4 30000008 01            8.02 HBW [Home-based trip (work)]
##  5 30000012 01            4.29 HBW [Home-based trip (work)]
##  6 30000012 01            6.82 HBW [Home-based trip (work)]
##  7 30000039 01           11.5  HBW [Home-based trip (work)]
##  8 30000041 01           73.7  HBW [Home-based trip (work)]
##  9 30000041 02           77.9  HBW [Home-based trip (work)]
## 10 30000041 02           77.8  HBW [Home-based trip (work)]
## # ... with 117,358 more rows
```

Notice that we didn't have to tell the `select` and `filter` functions the name of the table we were selecting or filtering. The `%>%` chain operator did that for us.

Once we have the table we want, we can assign it to a new object called `mytrips` In this case, let's get `HBO` and `HBW` trips.

```
mytrips <- trips %>%
  select(houseid, personid, trpmiles, trippurp) %>%
  filter(trippurp %in% c("HBW", "HBO")) # use %in% for multiple comparisons.
```

B.4.3 Mutate, Summarize, and Group

Sometimes we want to calculate a new column in a table, or recompute an existing column. We can do that with the `mutate` function, and we can put more than one calculation in a single `mutate` statement.

```
mytrips %>%
  mutate(
    tripkm = trpmiles * 1.60934, # convert miles to km.
    longtrip = ifelse(tripkm > 50, TRUE, FALSE) # is trip longer than 50 km?
  )
```

```
## # A tibble: 307,390 x 6
##    houseid  personid  trpmiles trippurp                    tripkm longtrip
##    <chr>    <chr>    <dbl+lbl> <chr+lbl>                  <dbl+lb> <lgl>
##  1 30000007 01            5.24 HBO [Home-based trip (oth~     8.44 FALSE
##  2 30000007 01            5.15 HBO [Home-based trip (oth~     8.29 FALSE
##  3 30000007 02           84.0  HBW [Home-based trip (wor~   135.   TRUE
##  4 30000007 02           81.6  HBW [Home-based trip (wor~   131.   TRUE
##  5 30000007 03            2.25 HBO [Home-based trip (oth~     3.62 FALSE
##  6 30000007 03            2.24 HBO [Home-based trip (oth~     3.61 FALSE
##  7 30000008 01            8.02 HBW [Home-based trip (wor~    12.9  FALSE
##  8 30000008 01            8.02 HBW [Home-based trip (wor~    12.9  FALSE
##  9 30000012 01            4.29 HBW [Home-based trip (wor~     6.91 FALSE
## 10 30000012 01            6.82 HBW [Home-based trip (wor~    11.0  FALSE
## # ... with 307,380 more rows
```

Other times we want to calculate summary statistics like means. For this we can use the `summarize()` function.

```
mytrips %>%
  summarize(
    mean_trip = mean(trpmiles),
    sd_trip = sd(trpmiles),
    max_trip = max(trpmiles),
    min_trip = min(trpmiles)
  )
```

```
## # A tibble: 1 x 4
##   mean_trip sd_trip max_trip min_trip
##       <dbl>   <dbl>    <dbl>    <dbl>
## 1      9.81    32.1    5699.       -9
```

Finally, we sometimes want to calculate summary statistics for different groups. We can tell `tidyverse` to group our tables with the `group_by()` function.

```
mytrips %>%
  group_by(trippurp) %>%
  summarize(
    mean_trip = mean(trpmiles),
    sd_trip = sd(trpmiles),
    max_trip = max(trpmiles),
    min_trip = min(trpmiles)
  )
```

```
## # A tibble: 2 x 5
##   trippurp                       mean_trip sd_trip max_trip min_trip
## * <chr+lbl>                          <dbl>   <dbl>    <dbl>    <dbl>
## 1 HBO [Home-based trip (other)]       7.73    32.0    5699.       -9
## 2 HBW [Home-based trip (work)]       13.2     31.9    2927.       -9
```

> As you might expect, work trips are on average longer than other kinds of trips. But some people report very long trips! You might want to filter your data more carefully for real analyses.

B.5 Graphics with ggplot2

The `ggplot2` package included in the `tidyverse` is a very powerful graphics engine with a relatively easy-to-learn grammar. In fact, the gg stands for "grammar of graphics" as it implements the grammar defined by Wilkinson (2012).

The basic structure of a `ggplot2` call is constructed as follows:

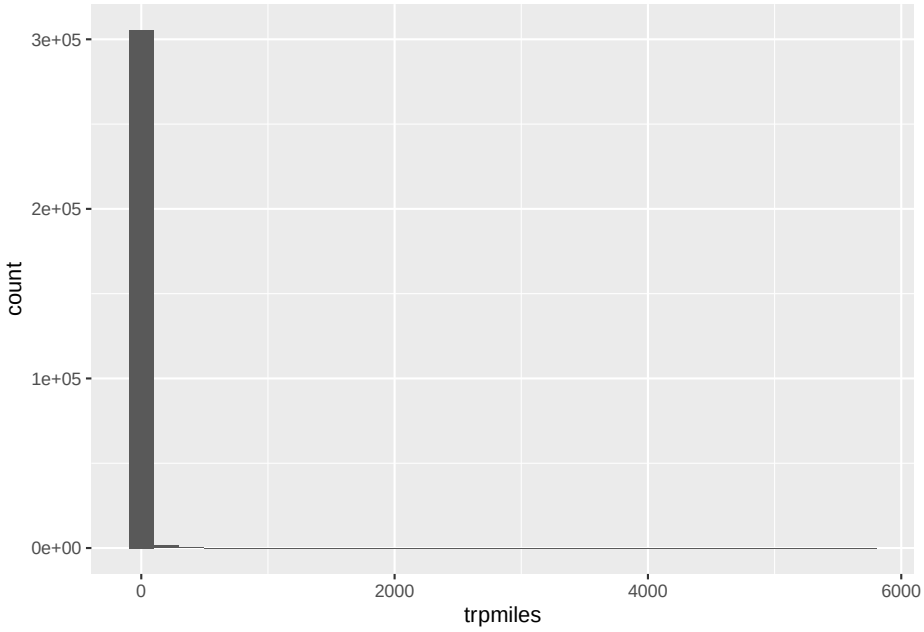
```
ggplot(data, aes(data aesthetics like x and y coordinates, fill color, etc.)) +
  geom_(geometry style like point, bar, or histogram) +
  other things like theme, color, and labels
```

For instance, we can create a histogram of trip lengths in the NHTS by giving the `x` aesthetic as the `trpmiles` column in the `mytrips` dataset.

```
ggplot(mytrips, aes(x = trpmiles)) +
  geom_histogram()
```

```
## Don't know how to automatically pick scale for object of type haven_labelled. Defaul
```

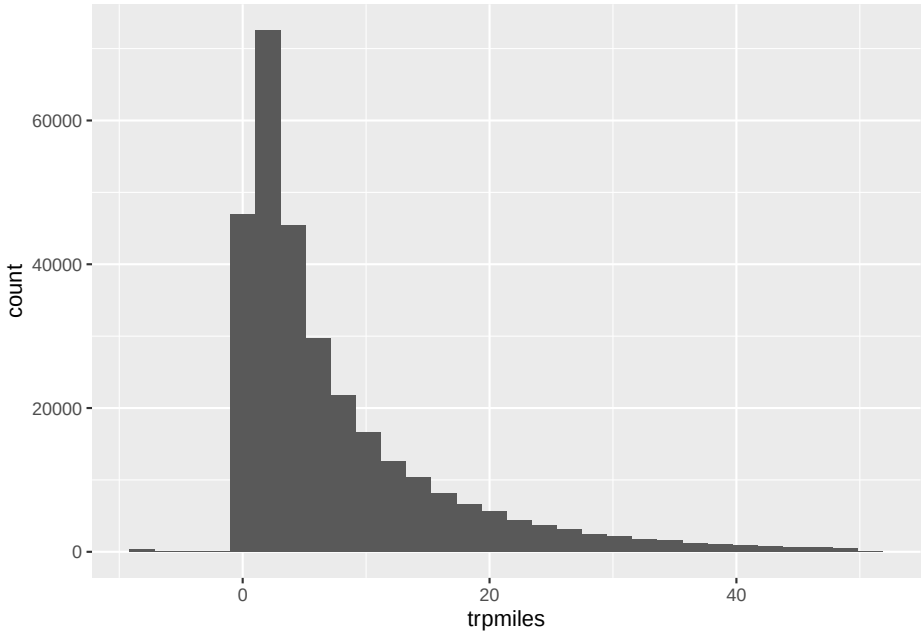
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



This ends up not being very informative because some trips are very long. We could filter out the long trips within the data argument (Note that we still have the `-9` values from the missing information).

```
ggplot(mytrips %>% filter(trpmiles < 50), aes(x = trpmiles)) +
  geom_histogram()
```

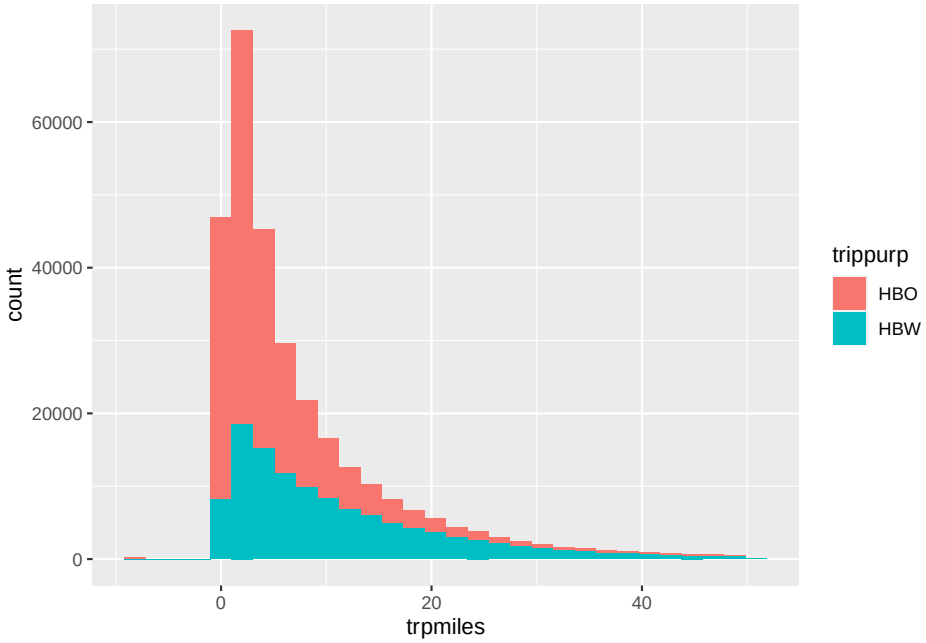
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



If we wanted to see the difference between lengths of different trip purposes, we could add a color aesthetic to the plot. By default this stacks the two categories on top of each other.

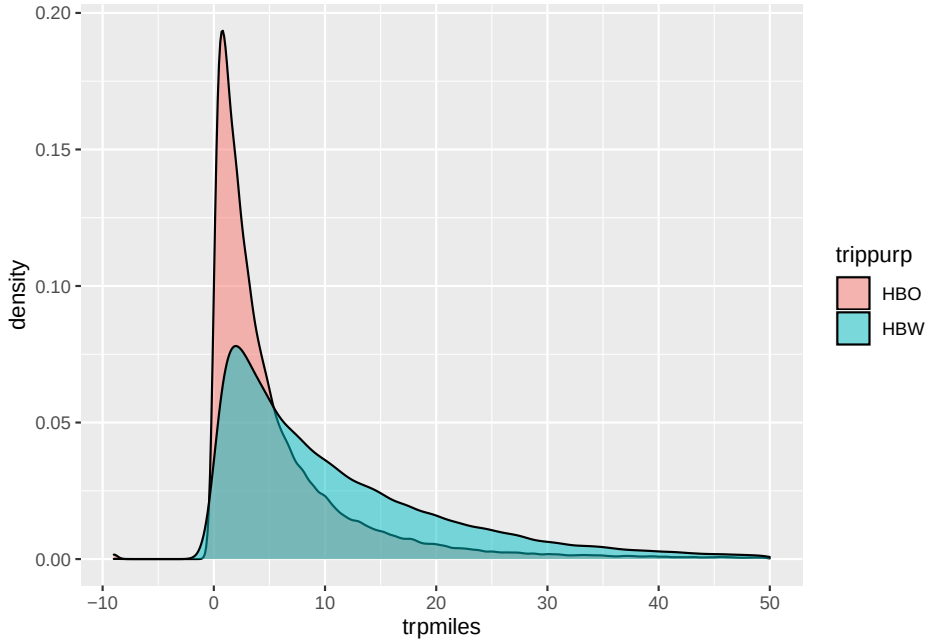
```
ggplot(mytrips %>% filter(trpmiles < 50), aes(x = trpmiles, fill = trippurp)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



You could also show this with a statistical density (the integral of a density function is 1). Note that the `alpha` statement for fill opacity is not included as an aesthetic, because it doesn't vary based on any data elements in the way that the `x` and `fill` variables do.

```
ggplot(mytrips %>% filter(trpmiles < 50), aes(x = trpmiles, fill = trippurp)) +
  geom_density(alpha = 0.5)
```

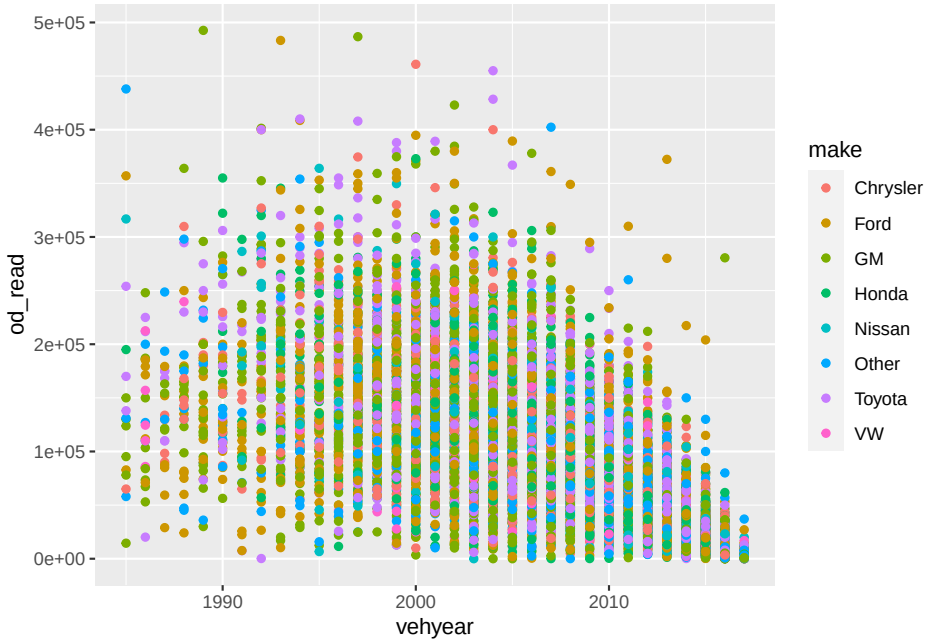


ggplot2 also excels at building statistical analysis on top of visualization. For example, we can see the odometer reading for cars still on the road in 2017 by make.

```
# sample 15k vehicles built after 1980 with 0 to 500k miles
vehicles <- nhts_vehicles %>%
  # convert numeric make to its labeled name, and then group into manufacturers
  mutate(
    make = as_factor(make, levels = "labels"),
    vehtype = as_factor(vehtype, levels = "labels"),
    make = case_when(
      make %in% c("Toyota", "Lexus", "Subaru") ~ "Toyota",
      make %in% c("Ford", "Lincoln", "Mercury") ~ "Ford",
      make %in% c("Chevrolet", "GMC", "Pontiac", "Buick", "Cadillac", "Saturn") ~ "GM",
      make %in% c("Volkswagen", "Audi", "Porsche") ~ "VW",
      grepl("Jeep", make) | grepl("Chrysler", make) | make %in% c("Ram", "Dodge", "Plymouth") ~ '
      make %in% c("Honda", "Acura") ~ "Honda",
      make %in% c("Nissan/Datsun", "Infiniti") ~ "Nissan",
      TRUE ~ "Other" # all other makes
    ) ,
    vehtype = case_when(
      grepl("Car", vehtype) ~ "Car",
      grepl("Van", vehtype) ~ "Van",
      grepl("SUV", vehtype) ~ "SUV",
      grepl("Pickup", vehtype) ~ "Pickup",
```

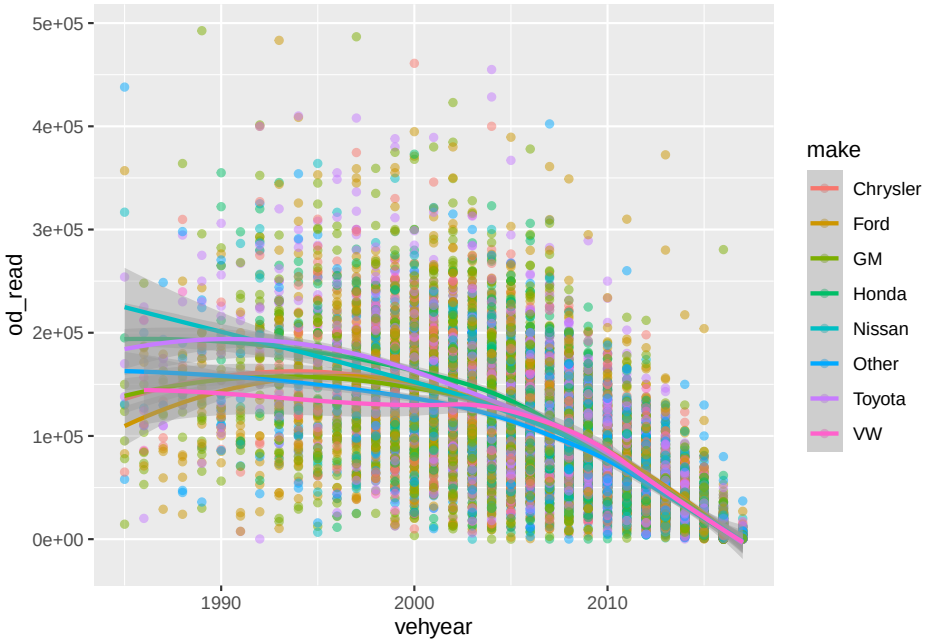
```
      TRUE ~ "Other",
    )
  ) %>%
  filter(vehtype != "Other") %>%
  filter(vehyear > 1980) %>%
  filter(od_read > 0, od_read < 500000) %>%
  sample_n(15000)

ggplot(vehicles, aes(x = vehyear, y = od_read, color = make)) +
  geom_point()
```



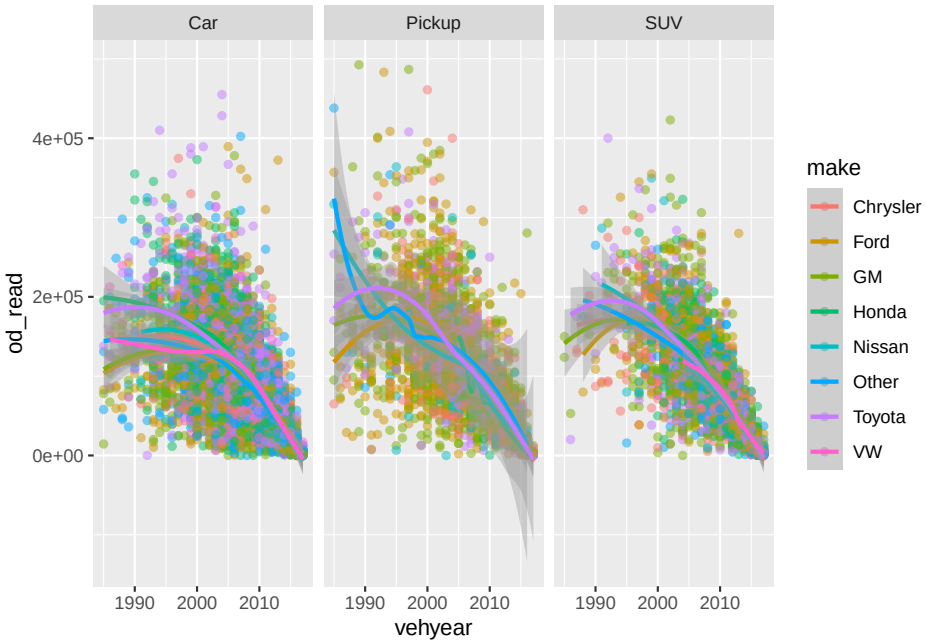
This is pretty unreadable. But we can add a few things to the figure to make it a little bit easier to understand, like smooth average lines and point transparency.

```
## `geom_smooth()` using formula 'y ~ x'
```



Let's break this out by vehicle type.

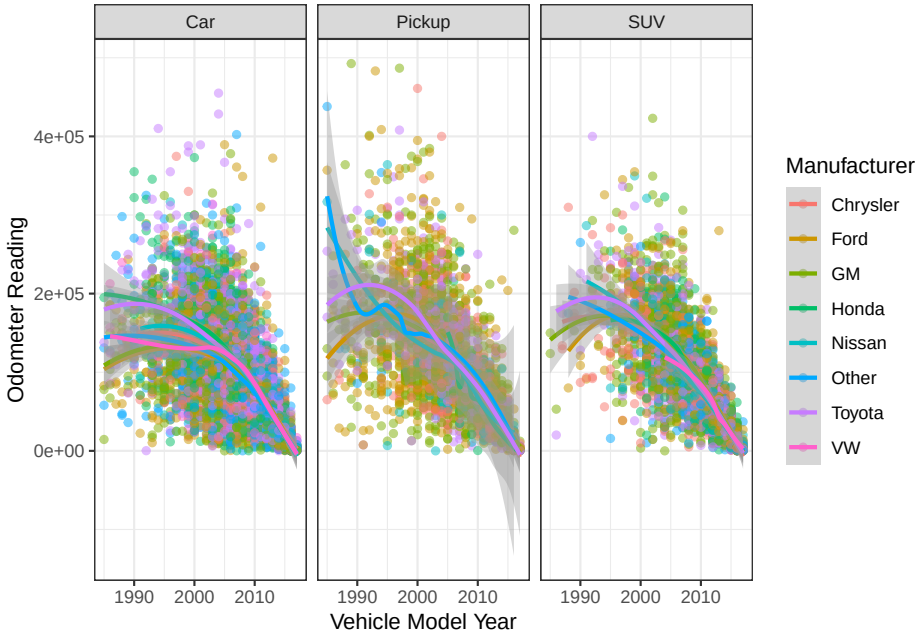
```
## `geom_smooth()` using formula 'y ~ x'
```



And let's clean it up a little bit. This is a figure that you could put in a published

journal article or thesis, if it showed something you cared to show.

```
## `geom_smooth()` using formula 'y ~ x'
```



Bibliography

(2018). *A Policy on Geometric Design of Highways and Streets, 7th Edition,*. The Green Book. American Association of State Highway and Transportation Officials.

Wilkinson, L. (2012). The grammar of graphics. In *Handbook of Computational Statistics*, pages 375–414. Springer.