21_analyzing_quantitative_data_intro_and_dataset

May 2, 2021

1 Brief introduction to Bayesian methods and `pymc3` for linguists

1.1 Introduction

In this notebook, we start introducing the basics of Bayesian statistical modeling.

- Bayesian methods are not specific to ACT-R, or to cognitive modeling
- they are a general framework for doing plausible inference over data, both categorical ('symbolic') and numerical ('subsymbolic') data

Why do we need statistical modeling at this point?

- very soon, we will start introducing the subsymbolic components of ACT-R, which come with a good number of numerical parameters / 'knobs'
- these 'knobs' need to be dialed in to specific settings based on (numerical) experimental data
- we don't want to pull the correct settings and values out of thin air, and hand-wave in the general direction of statistical inference for the proper way to obtain these specific values

The Bayesian inference framework introduced in this chapter will enable us to:

1. learn the best settings for numerical parameters from the data
2. explicitly quantify our uncertainty about these settings, and
3. do empirically-driven theory comparison

We will then be able to:

- introduce the subsymbolic components of ACT-R
- set the values of the numerical parameters associated with these components based on linguistic data, and
- numerically compare and evaluate different linguistic theories

1.2 The `Python` libraries we need

We first load the relevant `Python` libraries:

- `numpy` provides fast numerical and vectorial operations
- `matplotlib` and `seaborn` provide plotting facilities
- `pandas` provides data frames, i.e., data structures well suited for data analysis, basically Excel sheets on steroids; similar to R data frames

- finally, `pymc3` is the library for Bayesian modeling: Monte Carlo (MC) methods for Python3

[1]:
```
# uncomment the lines below to install the correct version of pymc3 and␣
 ↪dependencies
# !pip3 install --upgrade 'arviz==0.11.1'
# !pip3 install --upgrade 'pymc3==3.9.3'
```

[2]:
```
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn')
import seaborn as sns

import pandas as pd
import pymc3 as pm
```

1.3 The data

We will introduce Bayesian estimation methods by using a very simple data set from Brasoveanu, Adrian, and Jakub Dotlačil. 2015c. Strategies for scope taking. *Natural Language Semantics* 23:1–19.

We load the file using the `read_csv` method provided by `pandas` on line 1 below. Line 2 specifies that the `"quant"` (quantifier) variable should be considered categorical: we're interested in reading times (RTs) associated with the two quantifiers *every* and *each*. We can look at the shape of our data (line 3) and we can list first 3 rows of the data (line 5): we see that the data consists of 347 observations (rows) with respect to two variables (columns): `"logRTresid"` (residualized log-transformed RTs) and `"quant"`. We can also select several different rows / observations by using the `iloc` (integer-based location) method (line 10): we select rows `[0, 8, 18, 31]`, and we display the values in all the columns (`:`).

[3]:
```
url = 'https://github.com/abrsvn/pyactr-book/blob/master/data/every_each.csv?
 ↪raw=true'
every_each = pd.read_csv(url)
every_each["quant"] = every_each["quant"].astype('category')
every_each.shape
```

[3]:
```
(347, 2)
```

The data has 347 rows (observations) and 2 columns (variables). Let's take a look at the first 3 rows / observations:

[4]:
```
every_each.head(n=3)
```

[4]:

	logRTresid	quant
0	0.056128	each
1	0.241384	each
2	0.056128	every

Let's take a look at several non-contiguous rows (rows `0, 8, 18, 31`):

[5]:
```
every_each.iloc[[0, 8, 18, 31], :]
```

[5]:

	logRTresid	quant
0	0.056128	each
8	0.869077	every
18	-0.073706	every
31	-0.187536	each

This data is part of the results of Experiment 2 (a self-paced reading experiment) reported in Brasoveanu, Adrian, and Jakub Dotlačil. 2015c. Strategies for scope taking. *Natural Language Semantics* 23:1–19.

The experiment investigated a hypothesis formulated in Tunstall, Susanne. 1998. *The interpretation of quantifiers: Semantics and processing*. Doctoral Dissertation, University of Massachusetts, Amherst.

- the hypothesis that the distinct scopal properties of *each* and *every* are (at least to some extent) the consequence of an event-differentiation requirement contributed by *each*
- by scopal properties, we mean the preference of these quantifiers to take wide scope over another quantifier in the same sentence

Consider the examples below:

1. A helper dyed every shirt without thinking about it.
2. A helper dyed each shirt without thinking about it.

The quantifier phrases *every/each shirt* can take wide or narrow scope relative to the indefinite *a helper* in subject position:

- on the wide scope reading, the sentences above are taken to mean that every/each shirt was dyed by a possibly different helper
  - we also call this reading the *inverse scope* reading because the scope of the quantifiers is the inverse of their surface order
- on the narrow scope reading, also known as the *surface scope* reading (for obvious reasons), the sentences above are taken to mean that one and the same helper dyed every/each shirt

On the face of it, both *every* and *each* have the same meaning:

- they contribute so-called universal quantificational force
  - as opposed to indefinites like *a* or *some*, which contribute existential quantificational force

However, Tunstall (and others before her) notices that *each*, but not *every*, require a separate event for each element it quantifies over.

For example: - the sentence *Jake photographed every student in the class, but not separately* is perfectly acceptable - but the minimally different sentence *Jake photographed each student in the class, but not separately*, where *each* is substituted for *every*, is definitely less acceptable

Based on contrasts like this, Tunstall (1998, p. 100) proposes that *each* contributes a **differentiation condition** to the effect that:

> ''[e]ach individual object in the restrictor set of the quantified phrase must be associated with its own subevent, in which the predicate applies to that object, and which can be differentiated in some way from the other subevents."

There are many ways in which events can be differentiated from one another, but one way, relevant for our sentences above, is for *each* to take inverse scope.

- in that case, each shirt is dyed by a (possibly) different helper, which ensures that each shirt-dyeing event is differentiated from all others because of the different person doing the dyeing
- if *each* contributes an event-differentiation requirement, but not *every*, we expect it to have a higher preference for inverse scope than *every*
- and since inverse scope is known to lead to processing difficulties, which manifest themselves as increased reading times (RTs), we expect to see higher RTs for the *each* sentence relative to *every*
- for more discussion of the processing of inverse scope, see for example:
  - Kurtzman, Howard S., and Maryellen C. MacDonald. 1993. Resolution of quantifier scope ambiguities. *Cognition* 48:243–279
  - Tunstall (1998); see full reference above
  - Anderson, Catherine. 2004. *The structure and real-time comprehension of quantifier scope ambiguity*. Doctoral Dissertation, Northwestern University, Evanston, Illinois
  - Pylkkänen, Liina, and Brian McElree. 2006. The syntax-semantic interface: On-line composition of sentence meaning. In *Handbook of psycholinguistics*, ed. Matthew Traxler and Morton Ann Gernsbacher, 537–577. New York: Elsevier

In their Experiment 2, Brasoveanu & Dotlačil tested this prediction using a moving-window self-paced reading task.

- because the experiment included a separate manipulation, the most important regions of interest (ROIs) were the spillover words immediately following the universal quantifier phrase
- in the examples above, these ROIs were the words *without*, *thinking* and *about*
- the data set we have just loaded in Python and assigned to the `every_each` variable contains measurements collected for the third ROI *about*
- the RTs collected for the ROI *about* were transformed in a couple of ways
  - raw reading times in self-paced reading experiments are roughly between 300 and 600 ms per word
  - these raw reading times were first log-transformed, which yields log RTs roughly between 5 and 7; see the code cell immediately below
    - we will discuss log transformation / log compression in more detail when we discuss the subsymbolic components associated with declarative memory in ACT-R
  - in addition, following Trueswell, John, Michael Tanenhaus, and Susan Garnsey. 1994. Semantic influences on parsing: Use of thematic role information in syntactic ambiguity resolution. *Journal of Memory and Language* 33:285–318, Brasoveanu & Dotlačil factored out the influence of word length and word position out of log RTs, which yields residualized log RTs that are roughly between -3 and 3
  - in fact, in this particular case, they fall just between -1 and 2, as we can see when we inspect the minimum and maximum of the residualized log RTs in our data; see below

[6]:
```
np.log(300), np.log(600)
```

[6]: (5.703782474656201, 6.396929655216146)

[7]:
```
np.min(every_each["logRTresid"]), np.max(every_each["logRTresid"])
```

[7]: (-0.678407840683957, 1.19278354190761)

The main question we want to ask of this data set is:

- **are the reading times, specifically in the form of residualized log RTs, different for the two quantifiers *every* vs. *each*?**
- that is, we will model RTs as a function of quantifier

One way to model RTs as a function of quantifier is to estimate the two means for the two quantifiers:

- we can estimate the means and our uncertainty about them
- that is, we estimate two full probability distributions, one for each of the means

But estimating the mean RTs for the two quantifiers will not give us a direct answer to our question:

- is there a difference in RTs between the two quantifiers?
- in a Bayesian framework, we could still answer the question given a two-mean model
- but it is more straightforward (and closer to the way frequentist estimation would be done) to estimate the difference between the two quantifiers directly

Thus, in our model:

- we estimate the mean RTs for *every* (together with our uncertainty about it)
- instead of estimating the mean RTs for *each*, we estimate the mean difference between the *every* RTs and *each* RTs (together with our uncertainty about it)

We can still obtain our mean RTs for *each* by starting with the mean for *every* and adding to it the difference between the two RTs.

If we want to answer our question (are the RTs different for *every* vs. *each*?):

- we look at our probability distribution for the difference in RTs and check if enough of that probability distribution is away from 0
- 'enough' usually means 95% of the probability mass

Even if a lot of the details of what we just said do not make complete sense, it is important to pause at this point and realize the structure of the argument we are pursuing here.

- it is actually the opposite of what linguists are usually trying to do
- from very early on in our linguistic training:
  - we are presented with some data
  - *we automatically assume there is a pattern in the data*
  - we try to identify the pattern and build a theory to capture it

In contrast, our main job as empirically-driven statistical modelers is to ask:

- is there really a pattern in the data?

- how sure are we that we're not hallucinating regularities/signal in what is actually pure noise?
- how sure are we that we are not finding patterns in fleeting clouds?

Instead of assuming that there are patterns in the data, our job is to be skeptical and quantify our (un)certainty about the presence of such patterns.

- if we are certain enough that there is a pattern (usually, 'enough' is 95% certain), then we can proceed with the assumption that the pattern is real and can start building a theory for it

This kind of skepticism is actually familiar to linguists in other forms.

- for example, it is never clear at the outset whether a meaning-related phenomenon (e.g., licensing negative polarity items like *any* or *ever*) should receive:
  - a syntactic analysis (Klima, Edward. 1964. Negation in English. In *The Structure of Language: Readings in the Philosophy of Language*, ed. Jerry A. Fodor and Jerrold J. Katz, 246–323. Prentice-Hall, Englewood Cliffs), which might seem the 'obvious' way to go, or
  - a semantic analysis (Ladusaw, William. 1979. *Polarity sensitivity as inherent scope relations*. Doctoral Dissertation, University of Texas)
- as linguists, we know all too well that it is important to be skeptical about the assumptions we make as we build theories
- but it is equally important to be skeptical about the assumptions we make when we identify 'obvious' generalizations and patterns in the data
- all data (even introspective data) is ultimately behavioral data, i.e., a product of a performance system, never a direct expression of the unobservable competence system hypothesized to be at the core of the performance system
- so we need to be reasonably skeptical about all the generalizations and patterns we think we see in the behavior of the system

Therefore, our question about the quantifier data set is unpacked as follows:

- can we actually show with enough credibility that the RTs actually differ between the two quantifiers (*every* and *each*)?
- assuming we can, what is the magnitude of the change?
  - in residualized log ms, admittedly a non-intuitive temporal unit, which we will omit from now on
- also, what is our uncertainty about that magnitude?
- the answer to the two immediately preceding subquestions should be roughly of the form:
  - there was a change of x_{mean} on average, and
  - we're 95% certain that the actual value of the change is somewhere in the interval $(x_{\text{lower limit}}, x_{\text{upper limit}})$

Let's now turn to specifying the actual model. Officially, the model we are about to specify is called a t-test, or a linear regression with one binary categorical predictor.

[ ]: