Getting Ready for the Next Election

Daniel Kaplan

November 12, 2014

There is an extensive and sophisticated industry of data analysis for politics. For fund-raising, it's best to target efforts on people likely to give. For devising approaches to issues, it helps to know who are the "swing" groups: categories of people whose views are not completely fixed. For get-out-the-vote drives, it's good to identify areas where there is a substantial number of registered non-voters.

The data to support such exporations is sometimes available from public sources, e.g. voter registration information at a county-by-county level and campaign donation listings from the Federal Election Commission.

In this activity, you will work with `registeredVoters`, voter registration data from Wake County, North Carolina, USA. There are more than a half-million registered voters in Wake County. Among other things, for many of these voters a party affiliation is listed along with a flag to indicate whether the person voted in each of several recent elections. Documentation is available at http://msweb03.co.wake.nc.us/bordelec/Waves/boedatadescription.pdf.

The `registeredVoters` data table contains only 10,000 records. If, after you've developed an analysis, you want to test it on the complete dataset, you can access it with

```
load( url(
  "http://www.mosaic-web.org/go/Repository/DannyKaplan/WakeCountyBOE.Rdata"
  ))
```

But start with the smaller dataset for developing your approaches.

Looking at the Data

For simplicity, focus on the political party affiliation: `party`. Working with your team members, generate three or four hypotheses about what factors might shape party affiliation.

For these hypotheses, construct glyph-ready data and a summary table or an informative graph. Interpret the table or graph to check whether the hypothesis is supported by the data. Don't be disappointed if a hypothesis doesn't check out as you imagined. Finding that a hypothesis is **not** supported by the data is very important and worth reporting.

A Specific Question

Are there zip codes where the voters are registered more heavily than usual Democratic or Republican (or unaffiliated)?

Use `ZipGeography` to make a geographic scatterplot of the zipcodes, showing latitude and longitude, number of voters, and strength of affiliation.

Machine Learning

There is certainly important information to be had from simple graphs, tables, and maps summarizing the data. When you have more than three or four variables to consider, however, it becomes hard to relate the variables to each other.

This is where machine learning — techniques for identifying patterns that are not accessible by eye — comes in.

You're going to use one specific technique for pattern learning: recursive partitioning. This is a form of *supervised* learning to find a function that predicts an outcome (often called a "response variable").

To start, make sure that you have the `partykit` package installed on your system. Then load it.

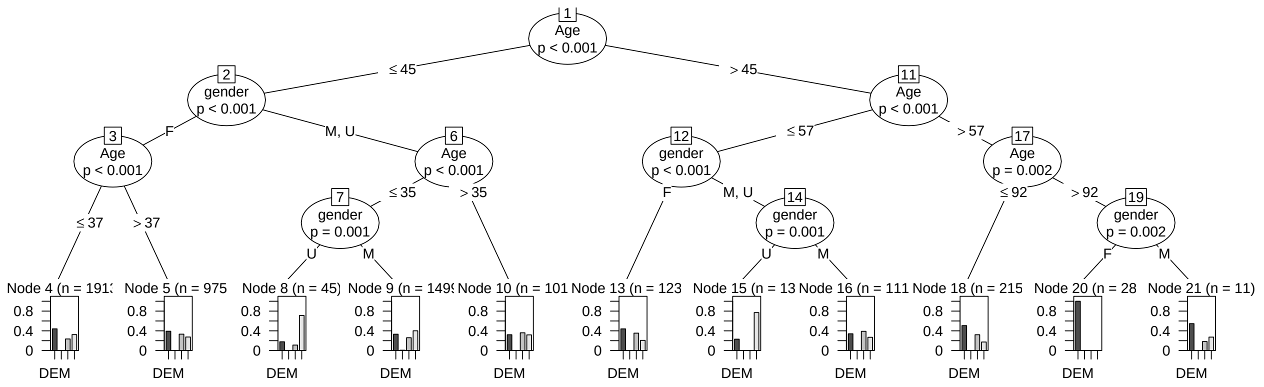
```
library( partykit )
```

To use recursive partitioning, you choose an outcome variable (e.g. `party`) and other variables that you think may be predictive. Then, simple enough, construct the model. For instance, this model uses age and sex to predict party affliation:

```
mod1 <- ctree( party ~ gender + Age, data=registeredVoters )
```

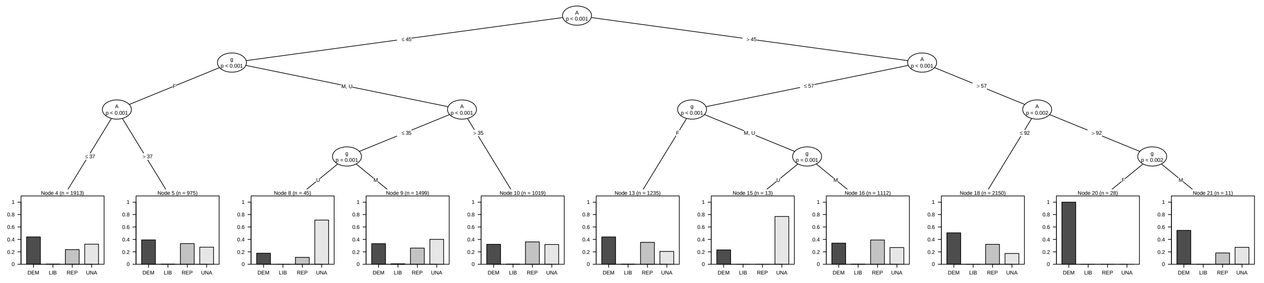
One of the easiest ways to interpret such models is to graph them out as a tree. You can do this simply with `plot( mod1 )`. Like this:

```
plot( mod1 )
```



Technical Aside: When the trees become complicated, you may want to have finer control over the graphic.

```
plot(mod1, gp = gpar(fontsize = 6),
  inner_panel=node_inner,
  ip_args=list(
        abbreviate = TRUE,
        id = FALSE)
  )
```



Your job as the human is to interpret the results. Leave it to the computer to create the models.

Try several different explanatory variables in your model. Report the best at dividing the electorate up into identifiable subgroups, so called marketing "demographics".

The models can be used for prediction as well as classification. The `predict()` function will return both the prediction itself, as well as the node to which each case belongs.

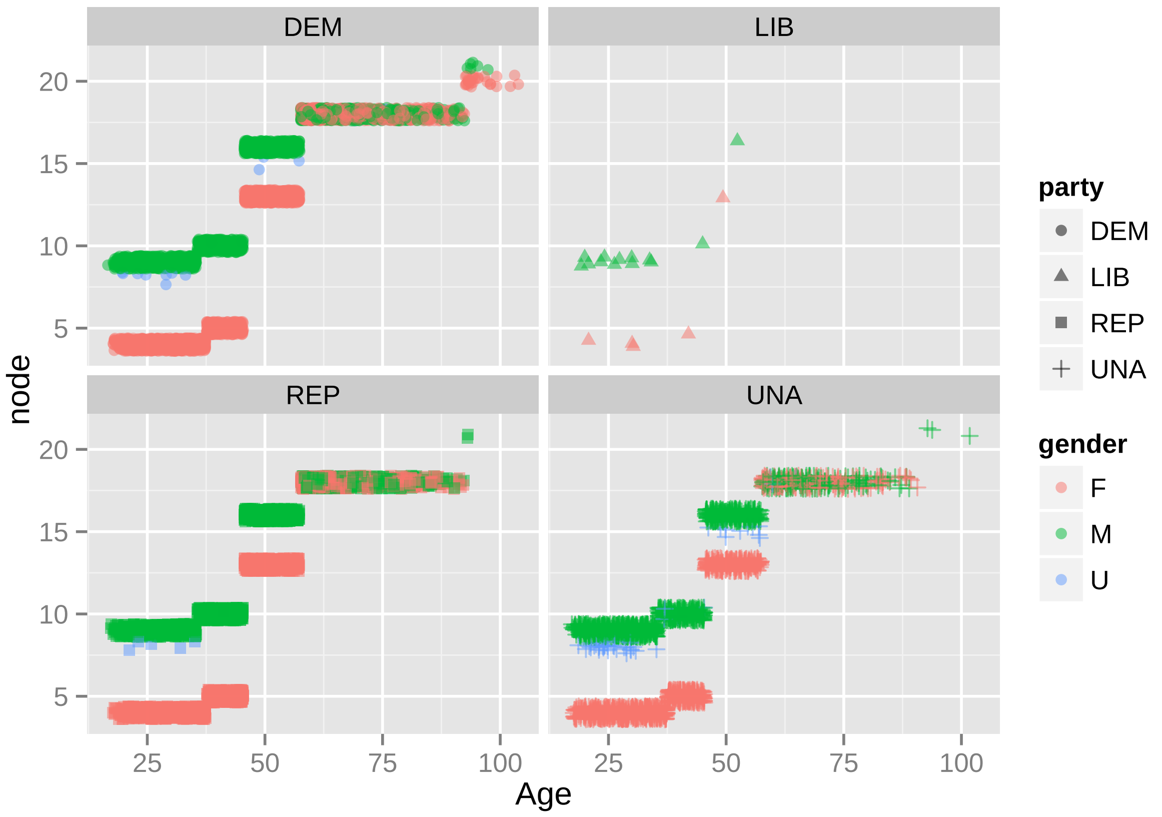
```
results <- predict( mod1 )
nodes   <- names( results ) %>% as.numeric
```

The prediction is rather simple for a categorical response variable: everybody is predicted to be in the dominant category for that node.

You can use `mutate()` to add the nodes or prediction along with the input data.

Sometimes it's helpful to make a graphic showing how the node relates to other variables.

```
ggplot( data=registeredVoters, aes(x=Age, y=node, color=gender, shape=party)) +
  geom_point(alpha=.5, position="jitter") + facet_wrap(~party)
```


DEM
LIB
REP
UNA
node
Age
party
DEM
LIB
REP
UNA
gender
F
M
U