The Election Analytics Glossary

Sun Young Park and Soubhik Barari
Gov 1347, Harvard University

October 7, 2020

Analyzing elections involves a difficult (but learnable) mixture of both applied statistics and political science knowledge. We will update this glossary as needed so that it serves as a common reference for some of the terminology, concepts, or vocabulary pertaining to applied statistics or political science that we use throughout the semester. Some of these are simply points made in section (sometimes quickly) articulated in written form for reference; others are supplementary and serve to deepen your understanding of certain materials.

For a practical treatment of some of these concepts in R, refer to `ModernDive`. If you have a specific question or are seeking more intuition, we highly recommend you try searching on `Cross Validated`, the largest stats Q&A forum on the web – answers are often highly specific and helpful!

Contents

1 Variables

In election analytics, we distinguish between two different types of variables (things that vary in the world!) – **independent variables (IVs)**, or variables we tend to think of as capturing different or "independent" variables in the real world, and **dependent variables (DVs)**, or variables we are trying to predict that are "dependent" on these IVs.

You'll often hear us (the instructors) use the term independent variable interchangeably with the following:

- predictor
- covariate
- regressor
- explanatory variable
- input

Similarly, we use dependent variable interchangeably with the following:

- outcome
- response
- target
- output

2 MSE (Mean Squared Error)

Let's say we want to predict a **dependent variable** y (ex. popular vote share), with two **independent variables**, x_1 (ex. GDP growth) and x_2 (ex. presidential approval rating). We may specify a simple linear regression model as below:

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2$$

Interpretation of the parameters in the model:

- $\boldsymbol{\alpha}$ **(intercept parameter)** can be interpreted as the value of y when all the independent variables are 0. (When $x_1 = 0$ and $x_2 = 0$, $y = \alpha + \beta_1 0 + \beta_2 0 = \alpha$)
- $\boldsymbol{\beta_1}$ **(slope parameter)** denotes how much y increases when x_1 changes by 1. Similarly, β_2 indicates how much y increases when x_2 changes by 1.

In R, by running `lm(y ~ x1 + x2, data = df)`, we can estimate the values of α, β_1, and β_2. `lm` commands returns parameters estimates that fit our data the best (ex. US presidential election between 1948 and 2016). These estimated values of parameters (*regression coefficients*) are denoted with a hat sign: $\hat{\alpha}$, $\hat{\beta}_1$, and $\hat{\beta}_2$.

Using these estimates, we can now predict the dependent variable only by using our independent variables. Note that $\hat{y}$ is the value of y predicted as such (*predicted value* or, when the corresponding independent variables were already fit on, *fitted value* of y):

$$\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2.$$

For the elections between 1948 and 2016, we have true value of y in our data. So by comparing y and $\hat{y}$, we can see the **in-sample** predictive performance of our model. Here, 'sample' is a term that refers to the data we used to fit the model. For each observation i in $1, ..., N$ in our data (in our case, each election year), we can calculate error (or **residual**) in the model prediction for each data point:

$$\mathsf{error}_i = y_i - \hat{y}_i$$

And to summarize the predictive power of the model, we can square the error and get the mean error across observation.

$$\mathsf{error}_i^2 = (y_i - \hat{y}_i)^2$$

$$\mathsf{MSE} = \frac{1}{N}\sum_i^N \mathsf{error}_i^2 = \frac{1}{N}\sum (y_i - \hat{y}_i)^2$$

Why do we square the error? We do this primarily to get rid of the *sign* of the error, so we treat over- and under-performance errors equally. This is what makes it a simple summary of model performance. But sometimes it is useful to differentiate these, in which case we might want to visualize a histogram of the errors rather than summarise it with one number.

3 Linear Regression Model `summary()`

Some of you might be wondering more about what various components of the `summary` of a linear regression means. For instance, here is the summary of the regression of popular vote share on poll support for the incumbent candidate:

```
Call:
lm(formula = pv ~ avg_support, data = dat_poll_inc)
Residuals:
    Min      1Q  Median      3Q     Max
-5.3613 -0.9303  0.1666  2.6013  4.1467
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  15.1876     4.8647   3.122  0.00972 **
avg_support   0.7294     0.1047   6.966 2.37e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 2.931 on 11 degrees of freedom
Multiple R-squared:  0.8152,Adjusted R-squared:  0.7984
F-statistic: 48.53 on 1 and 11 DF,  p-value: 2.372e-05
```

Before we proceed, we should mention that all models, of course, will have error in both the data its fitted (in-sample) and the data its not fitted on (out-of-sample).

What's special about linear regression is an *assumption* that it makes on its errors (for data point i, $\texttt{pv}_i - 15.1876 + (0.7294 \times \texttt{avg\_support}_i)$): that, altogether, they will be distributed according to a **normal (Gaussian) distribution**, or colloquially a "bell curve".

- **Residuals** describes the observed distribution of the errors for our sample. For example, above, we see that the smallest error between the true popular vote and predicted popular vote within the sample is `-5.3613`. The median is only `0.1666`. Maybe, not so bad!

- **Coefficients** describe the *point estimates* for each of parameters (`15.1876` and `0.7294` here) which are estimated (by default) using Ordinary Least Squares (OLS). The *standard error* of each coefficients is the uncertainty of our estimate given our errors – this actually comes from a nice formula since we're assuming normality in our errors.

- t**-values** tell us how far our parameter estimates are from zero, adjusting for their standard error. Why is this called t? It turns out that since we assume our errors are normally distributed and given how the linear regression equation is set up, the error between our coefficient estimate ($\hat{\beta}$) and the true coefficient value (β) will be distributed according to a t**-distribution**.

- `Pr(>|t|)` or the p**-value** tells us *the probability we would observe a t-value as high as we did if the <u>true</u> parameter estimate were 0*. This hypothetical scenario of the true paramter being 0 is called the **null hypothesis**. A small p-value indicates that it is unlikely we will observe a relationship between the predictor by chance alone – which means we reject this null hypothesis.

  Typically, a p-value of 5% or less is a good cut-off point. In our model example, the p-values are very close to zero. Note the `'Signif. codes'` associated to each estimate. Three stars (`***`) represent a highly significant p-value. Consequently, a small p-value indicates that we can reject the null hypothesis which allows us to conclude that there is a relationship between `avg_support` and `pv`.

- **Residual standard error (RSE)** is the average distance that each true response value y_i will deviate from the regression prediction $\hat{y}_i$. It's *almost* the like the square root of the MSE (see above), except instead of dividing by N, we divide by the **degrees of freedom** $N - k$ where k is the number of parameters (2 in this case).[1]

[1]Intuitively, the `degrees of freedom` tells us how much of the data we're actually using to estimate <u>each</u> parameter – if it's close to N (e.g., just one predictor) then each parameter is drawing on a lot of data, but if it's close to 1 (e.g., a silly model where we have a dummy predictor for literally every data point), each parameter is being estimated with very little data.

- **Multiple R-Squared** tells us how much variation in the outcome our selected predictors explains and **Adjusted R-Squared** is the former but with a penalty if we add too many IVs that don't help explain variation in the outcome. See slides in the economy and poll weeks for more on these.
- F**-statistic** is the most conservative of tests of our model's quality. It compares the error of the fitted model to the error of a model with *just* an intercept (called a reduced model).

  The related statistical test uses our assumption of error normality to answer the question: *is our model's error small enough to be statistically distinct from a reduced model?* If the `p-value` is small, then yes: if it were not statistically distinct from a reduced model (the null hypothesis), there's only a tiny chance we'd see this F statistic ($\approx$ the ratio of the reduced model's RSE to your model's RSE).

In this class, we'll mostly be focusing on the residuals as a measure of model fit – either by plotting them or looking at the MSE.

4 Interactions

Let's say now we want to predict popular vote share (DV), with increase in unemployment rate and the partisanship of the incumbent president (IVs).

Now, let's assume we expect a *differential effect* of unemployment rate depending on the party of the incumbent president. We can capture this differential effect in our regression model with **an interaction term** as below.

$$\text{(popular vote share)} = \alpha + \beta_1\text{(increase in unemployment)} + \beta_2\text{(incumbent is Republican)} \tag{1}$$

$$+\beta_3\Big(\text{(increase in unemployment)} \times \text{(incumbent is Republican)}\Big) \tag{2}$$

To clarify, these are the independent variables in the model:

- (incumbent is Republican): a dummy variable that takes 1 if the incumbent is Republican and 0 if the incumbent is Democrat.
- (increase in unemployment): the percentage point increase in unemployment from the previous quarter.

To get a sense of why there's a multiplicative term, let's see what happens to various inputs to our model:

- incumbent is Democrat, 0% unemployment increase:

$$\text{(popular vote share)} = \boldsymbol{\alpha} + \beta_1 0 + \beta_2 0 + \beta_3(0 \times 0).$$

- incumbent is Republican, 0% unemployment increase:

$$\text{(popular vote share)} = \alpha + \beta_1 0 + \beta_2 1 + \beta_3 (0 \times 1).$$

- incumbent is Republican, 2% employment increase:

$$\text{(popular vote share)} = \alpha + \beta_1 2 + \beta_2 1 + \beta_3 (2 \times 1).$$

- incumbent is Democrat, 2% employment increase:

$$\text{(popular vote share)} = \alpha + \beta_1 2 + \beta_2 0 + \beta_3 (2 \times 0).$$

If we squint at these examples, we can see that β_1 is *the effect of an increase in unemployment for Democratic incumbents*. More precisely, β_1 can be interpreted as the predicted difference in popular vote share corresponding to 1 percentage point increase in unemployment, if (incumbent is Republican) is 0.

β_2 can be interpreted as *the difference in baseline* popular vote share between the Democrat and Republican incumbents when (increase in unemployment) is 0. To see this, plug in 0 to all (increase in unemployment) into the model.

Now, it is clear that $\beta_1 + \beta_3$ is *the effect of unemployment on Republican incumbents' vote share* (as compared to a Democrat). β_1 was the effect of unemployment for Democratic incumbents. So we can interpret β_3 as the additional (potentially negative) effect of unemployment for a Republican incumbents' vote share. β_3, in other words, is the interaction term.

Thus the multiplicative term captures the interaction between party and unemployment, or the differential or added effect of unemployment for a Republican. We can arrange the parameter interpretations corresponding to this interaction in the following table:

	incumbent is Democrat	incumbent is Republican
unemployment increases by x	$\beta_1 x$	$(\beta_1 + \beta_3)x + \beta_2$

You can estimate all the parameters above in R by running `lm(y ~ x1 + x2 + x1:x2, data = df)` or `lm(y ~ x1*x2, data = df)`. Note that ':' here is unrelated to the : operator used to create sequences like '`1:10`' in R. Here, ':' is just asking the regression formula to multiply x1 and x2 and pass that value to `lm` as another independent variable to the regression. We, in fact, ran the regression for the example above in class! The results were:

	parameter	Estimate Std.	Error	t value	Pr(>|t|)
(Intercept)	α	49.259	0.923	53.367	<2e-16 ***
unemployment increase	β_1	-1.964	1.532	-1.282	0.2004
incumbent is Republican.	β_2	2.104	1.147	1.834	0.0673 .
incumbent is Republican : unemployment increase	β_3	-1.734	1.635	-1.061	0.2894

Quiz to test your understanding:

- What is the effect of 1 percentage point increase in unemployment for Democratic incumbents' popular vote share?
- What is the effect of 1 percentage point increase in unemployment for Republican incumbents' popular vote share?
- Write down the interpretation of the third coefficient (2.104).

Answers:

- The popular vote share will decrease by 1.964 percentage point.
- The popular vote share will decrease by 3.698 percentage point.
- When unemployment does not change from the previous quarter, Republican incumbents will on average have 2.104 percentage point higher popular vote share than Democratic incumbents.

5 Ensembles

An **ensemble** is a collection of models that are combined in some principled way. More colloquially, election forecasters call this a *combined forecast*. One principled way is taking a simple average where each model is weighted equally, or of equal importance. Another way is weight each model according to some criteria – we usually call these **weighted ensembles**.

Note: this is different from a multivariate regression model! Although we can think of estimated coefficients as "weighting" different IVs, a multivariate regression is one single model (function) with one single procedure (typically OLS) for estimating model parameters.

Weights in an ensemble can be picked according to *many* different criteria so that we can exploit useful features of different models. For instance,

- We might include one model that has strong in-sample fit but less-than optimal out-of-sample performance (from cross-validation) and another model with weak in-sample fit, but stronger out-of-sample performance.
- We might include one model that uses a set of independent variables that is available for 80 years (e.g., economic data) and another model that uses a set of independent varaibles available for 60 years (e.g., polling data).

This mix-and-match flexibility is what makes ensembles appealing!

One question you might have: why would we want to do instead of just incorporating independent variables (IVs) together into a <u>single</u> multivariate model (like multivariate linear regression)? As we know, one major feature of linear regression is its linear-ness: our model assumes that predictors x_1 and x_2 combine in a linear way $\beta_1 x_1 + \beta_2 x_2$ to contribute to the outcome y. An artifact of this is that univariate models individually might have better out-of-sample performances than a multivariate regression that combines them together in this (unrealistic) way. Instead, we can weight the two univariate models on their own to maximize out-of-sample performance.

Some ways that forecasters weight separately fitted models:

1. Weight a bunch of models that each have theoretical merit (PollyVote does this)
2. Weight a bunch of models based on their in-sample performance (R^2) (Nate Silver's 2020 forecast does this)
3. Weight a bunch of models based on some other variable that we theoretically believe increases predictiveness, e.g. days left til election (if we're trying to weigh a poll model, this turns out to have the same effect as doing 2)
4. Weight on a bunch of models using their cross-validation error (in statistics, this technique is called "Super Learning"; this has the nice property of never performing worse out-of-sample (on average) than any of its constituent models!)

6 Probabilistic Models

A **probabilistic model** is a model where the outcome (DV) or predictors (IVs) have an explicitly probabilistic interpretation – as in they can be interpreted as random variables drawn from some probability distribution rather than deterministic variables with fixed values.

Consider our vanilla linear regression model:

$$Y = A + BX$$

Recall in Section 3 that we implicitly make an assumption that the errors, $y_i - (\hat{A} + \hat{B}x_i)$ are distributed according to a normal distribution.

In fact, this actually implies that y_i itself is distributed according to a normal distribution denoted as $y_i \sim \mathsf{Normal}(A + Bx_i, \sigma^2)$, meaning that y_i has mean $A+Bx_i$ and some variance σ^2 (which we can estimate from the residuals). This *technically* make simple linear regression a probabilistic model. However, as we explain in lab, this doesn't always line up with the actual probabilistic process in our data. It is more like this is the default probabilistic assumption that linear regression makes which is not always right (we discuss in our lab section on the "air war").

Instead we can choose a probabilistic model like binomial logistic regression, which *can* closely model the probabilistic process we're analyzing: turnout of voters in an election.

6.1 Binomial logistic regression

The vote total for any particular candidate in any election has a fixed set of possible values: $0 - \mathsf{VEP}$, where VEP is the voter eligible population for the election in question. Let's say we're interested in the popular vote total for Democrat candidates in some state s, DemPV_s. Thus, the election outcome for Democrats in state s is some draw of voters from the voter-eligible population (VEP_s) turning out to vote Democrat. We call this process of draws from a population (often called successes from a number of trials) a **binomial process**.

6.1.1 Components of a binomial process

The core probabilistic phenomena driving a binomial process is whether or not a single voter, denoted by i, will turn out and vote for the party in question. Let's call this individual voter probability in state s. Let's denote the predicted probability of one voter draw for the Dems as:

$$p_{s,i} = \Pr(\mathsf{VoteforDem}_{s,i})$$

The question we can ask is: what, then, is the probability of n voters voting for the democratic party in this state? It turns out that there is a formula for this probability with some intuition given below:

$$\Pr(\mathsf{DemPV}_s = n) = \underbrace{p^n_{s,i}}_{\substack{\text{probability of}\\ \text{n turning out}\\ \text{for Dem}}} \times \underbrace{(1-p_{s,i})^{\mathsf{VEP}_s - n}}_{\substack{\text{probability of}\\ \text{VEP}_s - \text{ n not}\\ \text{turning out for Dem}}} \times \underbrace{\left(\frac{\mathsf{VEP}_s!}{n! \times (\mathsf{VEP}_s - n)!}\right)}_{\substack{\text{to account for}\\ \text{interchangeable orderings}\\ \text{of turned out voters}}}$$

As you might guess, this distribution is explicitly named the Binomial distribution.

We can simulate draws from 10,000 repeated Binomial process with a population of size `VEP` and a draw probability of `p` in R as follows:

```
rbinom(n = 100000, size = VEP, prob = p)
```

6.1.2 Modeling a binomial process

So once we have an estimate of $p^n_{s,i}$ for a particular state we can estimate a probability distribution for DemPV_s in that state. We can then predict an expected value for DemPV_s, or the mean value of this probability distribution. What's nice is that this is <u>guaranteed</u> to be in the range of $0 - \mathsf{VEP}_s$, since $\sum_{n=0}^{\mathsf{VEP}_s} \Pr(\mathsf{DemPV}_s = n) = 1$. Sweet!

So, how do we actually estimate $p^n_{s,i}$? In fact, we can estimate via a regression model ... but with a twist: we need a **link function** $f(\cdot)$ to make sure our outputs are in range. We can model the draw probability as follows:

$$\begin{aligned} p_{s,i} = \Pr(\mathsf{VoteforDem}_{s,i}) &= f(\alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k) \\ &= \frac{\exp(\alpha + \beta_1 x_1 + \ldots + \beta_k x_k)}{1 + \exp(\alpha + \beta_1 x_1 + \ldots + \beta_k x_k)} \end{aligned}$$

Here, the link function is called the **inverse logistic function (inverse logit)** which bounds values in the range $(-\infty, \infty)$ to the probability range $(0, 100)$.

Table 1 compares characteristics of linear regression with binomial logistic regression, as we see in lab.

The interpretation for each coefficient β_j: β_j is *the marginal increase in the odds*[2] *of a voter turning out for Dems given a unit increase in the jth predictor.*

[2]Technically, it's logged odds. The $\log(\cdot)$ is needed to, again, make sure that our predicted probability remains in the probability range and not $(-\infty, \infty)$

Supposing we have `x` (a single IV), `y` (a DV) as a % which is computed from `y = draws/vep`:

	Linear regression	**Binomial logistic regression (binomial logit)**
link function	$f(\alpha + \beta x) = \alpha + \beta x$	$f(\alpha + \beta x) = \frac{\exp(\alpha+\beta x)}{1+\exp(\alpha+\beta x)}$
link function name	identity	inverse logistic function
link function output	predicted outcome	predicted probability of one draw
`R` code	`lm(y∼x)`	`glm(cbind(draws, vep-draws)∼x, family=binomial)`
fitting intuition	"do OLS to find coefficients that minimize $\sum(y - \hat{y})^2$"	"find coefficients where fitted draw probabilities $f(\hat{\alpha} + \hat{\beta} x)$ best predict observed `draws` for all x"
prediction intuition	"plug in x_{new} and get (i) predicted outcome $\hat{y}_{new} = \hat{\alpha} + \beta x_{new}$ and (ii) prediction interval $\hat{y}_{new} \pm 1.96 \times \text{se}(\hat{y}_{new})$"	"plug in x_{new} and get (i) predicted probability of one draw, $f(\hat{\alpha} + \hat{\beta} x_{new})$; also plug in `vep` to get (ii) predicted expected number of draws, $\widehat{\texttt{draws}} \rightarrow \frac{\widehat{\texttt{draws}}}{\texttt{vep}}$ and (iii) predicted distribution of draws from repeated binomial process simulations"

Table 1: Comparison of characteristics of linear regression and binomial logit.