Erasmus School of Economics

Statistics for Data Science Lecture 6

Dennis Fok (Econometric Institute)

September – October, 2025

Erasmus University Rotterdam

Before next time

- Reread Chapter 4 if needed
- Read from Chapter 5:
  - Logistic regression
  - Evaluating Classification Models
- Reconsider the in-class assignments of this week
- Take home assignment
- Ask questions on the discussion board
- Work on final assignment (next deadline October 12)

Take home assignment (see "starter code" on Canvas)

- Use the Murder rate data (Murder as dependent variable)
- Take four independent variables: Income, Population, Illiteracy, Frost
- Perform forward stepwise regression
- Test whether the optimal model obtained from forward stepwise regression is significantly different from the complete model (these are nested models)
- Also try backward selection starting from all four variables
- And try all subsets selection on AIC

Plan for today

Part I: Categorical explanatory variables

→ Working with "Dummy variables"

Part II: Generalized Linear Models

- Main idea
- Logit models

Part III: Bootstrap (perhaps make a start with)

- Main idea
- Practical application

Categorical variables

Can regression do more?

- What we did by regression so far
  - Independent variable y: quantitative variable
  - Dependent variables X: quantitative variable
- Quantitative versus qualitative variables
  - Quantitative (numerical): murder rate, population, price, etc.
  - Qualitative (categorical): gender, brand, postcode, etc.
- Handling qualitative variables
  - y is qualitative: Later today in part II
  - Now: X is qualitative

Simplest case

- Consider the simplest case
  - Only one independent variable x
  - Only two categories: e.g. male (k obs) and female (m obs.)
- The goal: Test whether mean of y differs across the groups
- We did this with the Two-sample t-test
  $\rightarrow$ We can also do this using regression!

Dummy variable regression

- Construct a "dummy" variable D
  $\rightarrow D = 1$ for male and $D = 0$ for female
- Run regression $y = \alpha + D\beta + \varepsilon$
- Python automatically treats a categorical variable/factor (with two levels) like this
- Testing whether $\beta = 0 \rightarrow$ test of equal means

The equivalence between the approaches

- *Two-sample t test* and *t-test in dummy regression* give identical results!
- Maintained assumptions also equivalent
  - Two-sample t-test: equal variance in two groups
  - Dummy variable regression: ε has a constant variance σ^2
  - Both need normality (or a large enough sample)

Extending to multiple categories

What about variables with $K > 2$ categories?

- Create a dummy for each of $K - 1$ levels
- Add the $K - 1$ dummies to the model
- Coefficients give difference relative to *omitted level*

Python does all of this automatically in `smf.ols()`

- If the variable is already categorical → Just add the factor to the model
- Otherwise: make categorical (safest) `df["var"] = df["var"].astype("category")`
- .. or use `+ C(varname)` in the formula
- Baseline is automatically chosen
- Change the baseline using `C(varname, Treatment('baselevel'))`
  eg. `smf.ols(formula = "price ~ lotsize + C(sizeclass, Treatment('small'))", data=df)`

Testing for differences

Use t-tests in standard output

- Test one level of the factor versus the baseline level
  (Corrected for potential other variables in the model)
- Be aware of multiple testing!
  (with many factor levels testing at 5% may not be smart)
- Test result will depend on chosen baseline!

Test for the overall significance of the factor

- Test whether the factor is important as a whole
- Does *not* depend on baseline
- Compare model with factor vs without using the F-test

Relation with the ANOVA method

Analysis of Variance method (ANOVA)

- ANOVA is also seen as an important method by itself
- ANOVA tests for differences in dependent variables across groups

Variants to know about:

- One-way ANOVA
  → One categorical independent variable (1 factor)
- Two-way ANOVA
  → Two categorical independent variables (2 factors)
  - Without interaction
  - With interaction (groups are defined on the combination of both factors)
- ANCOVA (Analysis of Covariance)
  → Also allow for an additional continuous variable

All these methods are equivalent to regression with dummy variables
→ Don't need to learn specific ANOVA functions!

Assignment

In-class Assignment 6.1 (see "starter code" on Canvas)

Test the price difference for houses with respect to the number of bedrooms

- First make a factor for the number of bedrooms (see example code):
  - `df["catbed"] = ""` → create a new variable (containing strings)
  - `df.loc[df.bedrooms<=2, "catbed"] = "small"` → 1 or 2 bedrooms = small
  - `df.loc[df.bedrooms==3, "catbed"] = "medium"` → 3 bedrooms = medium
  - `df.loc[df.bedrooms>=4, "catbed"] = "large"` → 4 or more bedrooms = large
  - `df["catbed"] = df["catbed"].astype("category")`
- Test whether the price is different across the three categories:
  - Create a model with the factor
  - Use the reported F-statistic to test whether the categories matter

  → What is your conclusion?
- Interpret the coefficients of the regression model with the factor
- Change the baseline (`C(varname, Treatment("baselevel"))`) in the regression model and confirm that this does not affect your conclusion for the test (but does affect the coefficients)

Generalized Linear Models

What about "other" **dependent** variables?

Dependent variables are not always continuous

- Binary
- Categorical
- Counts
- Durations
- ...

→ Normal distribution cannot be correct for such variables (or errors)!

Generalized linear models can be used.
→ We focus on *binary* variables (aka logistic regression)

Binary dependent variables

Often the dependent variable (y_i) can only take two values
→ 0 or 1, "failure" or "success"

Examples

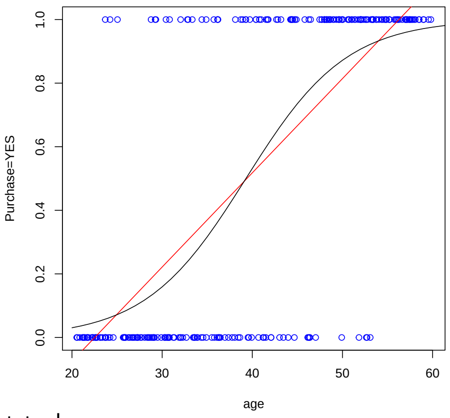
- Purchase decision (yes/no)
- Make an insurance claim (yes/no)
- Vote democrat or republican
- Respond to an email campaign (yes/no)
- Product returns (yes/no)

Research goals:

1. understand the decision making
2. predict (influence?) decision making

→ Need model to describe y_i using variables x_i

Example – "Has a house insurance"



Just applying OLS to $y_i = x_i'\beta + \varepsilon_i$

- *Linear Probability Model: LPM*
- ε_i cannot be considered to be normal
- ε_i has non-constant variance
- Predictions are strange

Want to have:

- Model that gives predictions within $[0, 1] \rightarrow$ probabilities
- A non-linear fit to the scatter.

Generalized linear models [GLM]

Can see this as a special case of a Generalized Linear Model (GLM)

Components of GLM for Y

1. A distribution function
2. Linear "predictor": $\eta = x'\beta = \beta_1 + \beta_2 x_2 + \ldots + \beta_k x_k$
3. Link function $g()$:

$$\eta = g(\mathrm{E}[Y])$$

   or conversely

$$\mathrm{E}[Y] = g^{-1}(x'\beta)$$

4. Variance is a usually some function of the expectation

GLM for binary data

For binary data we have

1. Bernoulli (or Binomial with n=1) distribution with $\mathrm{E}[Y] = \Pr[Y = 1] = \pi$
2. Standard linear predictor $\eta = x'\beta$
3. Link functions, choose from:

   Logit:

$$g(\pi) = \log(\frac{\pi}{1-\pi}) = x'\beta \qquad g^{-1}(x'\beta) = \frac{\exp(x'\beta)}{1+\exp(x'\beta)} = \pi$$

   Probit:

$$g(\pi) = \Phi^{-1}(\pi) = x'\beta \qquad g^{-1}(x'\beta) = \Phi(x'\beta) = \pi$$

   Complementary log-log (cloglog):

$$g(p) = \log(-\log(1-\pi)) = x'\beta \qquad g^{-1}(x'\beta) = 1 - \exp(-\exp(x'\beta)) = \pi$$

4. $\mathrm{Var}[Y] = \pi(1-\pi) = \mathrm{E}[Y](1-\mathrm{E}[Y])$ (= property of Bernoulli distribution)

Binary models in Python

Core function: `smf.glm()`

Example for binary data with logit link

```
smf.glm(formula="y~x",
        family = sm.families.Binomial(link=sm.families.links.Logit()),
        data=dataframe)
```

where

- `import statsmodels.api as sm` and
- `import statsmodels.formula.api as smf`

Other link functions using

- `sm.families.Binomial(link=sm.families.links.Probit())`
- `sm.families.Binomial(link=sm.families.links.CLogLog())`

Usage of GLM

As before `m = smf.glm(...)`

- `fit = m.fit()`: estimate the parameters of model m
- `fit.summary()`: give estimated coefficients, standard errors and t-tests
- `fit.params`: return the coefficients
- `fit.conf_int()`: return confidence intervals
- `fit.predict()`: give in-sample predicted values
- `fit.predict(exog=newdf)`: give predicted values
- Can also use backward and forward selection and all subset regression from `model_selection.py` (with updated version on Canvas)

Also very useful `ploteffect(fit, "varname")` from `ploteffect.py` (see Canvas)
→ Create plot of probabilities versus an explanatory variable

In-class assignment

In-class assignment 6.2 (see "starter code" on Canvas)

Use the data in website.RData

- The variable `min` contains the number of minutes an individual has spent on a particular website, `active` is a 0/1 indicator for active or not active
- Create a logit model to explain `active` using the other variables
  → Do not include `min` (why not?)
- Which variables explain the probability of being active?
- Create a plot of the predicted probabilities versus age for all datapoints
- Use the `ploteffect` function to create a plot of the predicted probabilities versus age for someone with the average income and the "average region".
- (Extra: Experiment with transformations of age and/or income)

Estimation

Estimation theory

To estimate β

- Minimizing $\sum_i e_i^2 = \sum_i (y_i - \hat{y}_i)^2$ is not very useful anymore
- Measure fit by "log likelihood" (high is good)
- In GLM terminology: Deviance = constant − 2×log-likelihood (low is good)

→ Maximum likelihood estimation

Likelihood function

Likelihood function

- Input: parameter values for a model
- Output: "probability" of observing the available data given these parameters

→ Joint density function of data seen as function of model parameters

Given

- observations on binary data $y_1, \ldots, y_n$
- model with parameters β

Likelihood function:

random variable *observed value*

$$L(\beta) = f(y_1, \ldots, y_n|\beta) = \Pr[Y_1 = y_1] \times \Pr[Y_2 = y_2] \times \ldots \times \Pr[Y_n = y_n]$$
$$= \prod_i \Pr[Y_i = y_i] = \prod_i p_i^{y_i}(1 - p_i)^{1-y_i}$$

where p_i is a function of β (follows from the chosen link function)

Maximum likelihood estimation

- Log-likelihood function:

$$\log L(\beta) = \sum_i y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

- So, deviance equals

$$D = -2 \sum_i (y_i \log(p_i) + (1 - y_i) \log(1 - p_i))$$

- Min deviance → max likelihood
  → Find β such that *data is as likely as possible*
- Distributions of estimators follow from max. likelihood theory
- For normal distribution: $D = \sum_i (y_i - \hat{y}_i)^2$ → Equivalent to OLS!

Model selection

- Deviance can directly be used to compare models

$$AIC = D + 2p,$$

  where p equals the number of parameters.
- Difference in deviance for nested models can be used for testing.
  Under H_0 : both models equally good:

$$D_{\text{small}} - D_{\text{large}} \sim \chi^2(k_{\text{large}} - k_{\text{small}})$$

  → likelihood ratio test
  p-value: `1 - stats.chi2(`$k_{\text{large}} - k_{\text{small}}$`).cdf(small.deviance - large.deviance)`
  where `from scipy import stats` and `small` and `large` are two fitted models
- Can apply *elimination* and *selection* methods for model selection

Other GLMs

There are various other GLMs

Different distributions

- Poisson (for count data)
- Exponential (for survival data)
- ...

Different link functions

- identity
- log
- reciprocal

(make sure that the range of the link function makes sense)

Prediction

Prediction

The logit or probit model gives predictions of $\Pr[y_i = 1]$: $\hat{p}_i$
$\rightarrow$ Often we want 0/1 predictions.

Choose a threshold c and transform $\hat{p}_i$ to $\hat{y}_i$ using

$$\hat{y}_i = \begin{cases} 1 & \text{if } \hat{p}_i > c \\ 0 & \text{if } \hat{p}_i \leq c, \end{cases}$$

Choices for c:

- Set $c = 0.5$ (default choice/maximizes hit rate)
- Set c equal to the fraction of observations with $y_i = 1$: (0/1 predictions more balanced)
- Set c optimally given *costs* of correct or wrong predictions

Evaluation of predictions

"Prediction-Realization" table (or "Contingency table" or "Confusion matrix").

	Realization		
	$y_i = 0$	$y_i = 1$	
Prediction			
$\hat{y}_i = 0$	p_{00}	p_{01}	$p_{0\cdot}$
$\hat{y}_i = 1$	p_{10}	p_{11}	$p_{1\cdot}$
	$p_{\cdot 0}$	$p_{\cdot 1}$	1

- Table can also be represented in number of observations
- `cm = skm.confusion_matrix(truevals, prediction)` from `import sklearn.metrics as skm`
- Prediction should be binary!
- More graphical: `skm.ConfusionMatrixDisplay(cm).plot()`
- Various statistics can be calculated based on this table
  - Hit rate (or accuracy) $h = p_{00} + p_{11}$
  - Sensitivity $p_{11}/p_{\cdot 1}$

In-class assignment

In-class assignment 6.2 – continued

- Compare the logit to the probit model using the deviance. Use `model.deviance` Which is better?
- Generate the confusion matrix to study the prediction quality of the logit
- Do you think the logit predicts well?
- (Optional: try to visualize the differences in predicted probabilities for logit vs. probit)

Parameter interpretation

Comparison logit/probit

- Difference between logit and probit is small
- Differences are larger when events are rare
- Choice depends on researcher's preference (derivations are easier for the logit model)

But estimated coefficients are really different!
$\rightarrow$ How can this be?

Parameter interpretation in non-linear models

Parameter interpretation is a bit difficult (nonlinear relations)

- Parameter deals with change in *linear predictor*
- Just looking at parameters is not enough
  $\rightarrow$ only gives *direction* of effect

To assign meaning to size of parameters:

- Compare probabilities under some scenarios
  $\rightarrow$ Make predictions for specific hypothetical cases, eg. using `m.predict(exog=...)`
  Special case:
  - Set continuous variables at their mean
  - Average over predictions for the levels of the categorical
  - `ploteffect(m, "varname")` from `ploteffect.py` (see Canvas)
- Translate parameters into something meaningful
  $\rightarrow$ (Average) marginal effects

Marginal effects

How does $E[y_i] = \Pr[y_i = 1]$ change if one of the x-variables changes?

- Effect of x_{ji} on $\Pr[y_i = 1]$ (keeping other variables fixed)
  $\rightarrow$ take derivative!
- This derivative is called marginal effect

Marginal effects for logit:

$$\Pr[y_i = 1] = \frac{\exp(x_i'\beta)}{1 + \exp(x_i'\beta)}$$

So

$$\frac{\partial \Pr[y_i = 1]}{\partial x_{ji}} = ... = \frac{\exp(x_i'\beta)}{(1 + \exp(x_i'\beta))^2}\beta_j = \Pr[y_i = 1]\Pr[y_i = 0]\beta_j.$$

Marginal effects

Marginal effect of x_{ji} in the logit model:

$$\Pr[y_i = 1]\Pr[y_i = 0]\beta_j$$

Note:

- Effect of x_{ji} on $\Pr[y_i = 1]$ depends on all elements of x_i.
- Effect equals 0 if $\Pr[y_i = 1]$ equals 0 or 1.
- Compare this to the LPM $y_i = x_i'\beta + \varepsilon_i$, where

$$\frac{\partial \Pr[y_i = 1]}{\partial x_{ji}} = \frac{\partial x_i'\beta}{\partial x_{ji}} = \beta_j.$$

- Marginal effect in the probit model:

$$\frac{\partial \Pr[y_i = 1]}{\partial x_{ji}} = \frac{\partial \Phi(x_i'\beta)}{\partial x_{ji}} = \phi(x_i'\beta)\beta_j.$$

- Often we consider the average marginal effect over all observations

Avg. marginal effects in example

Dependent variable: Has a house insurance

	Logit	avg. marg. effect	Probit	avg. marg. effect	LPM
C	-7.3409		-4.2670		-0.70183
AGE	0.17808	0.025637	0.10339	0.025758	0.029261

$\rightarrow$ Very small differences in marginal effects!

Calculating marginal effects in Python

Given a model `m = smf.glm(..).fit()` use `m.get_margeff().summary()`

- Gives average marginal effects
- + tests & confidence intervals
- `m.get_margeff(atexog=data)`: calculate marginal effects for specific observation

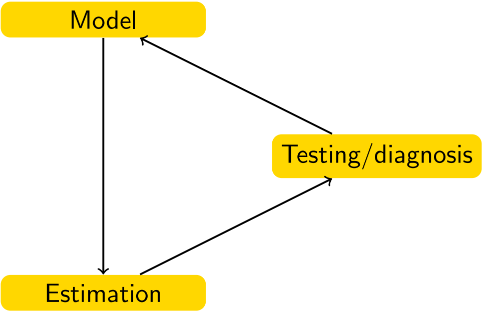
In-class assignment

In-class assignment 6.2 – continued

- Use the logit model with age, income and region as variables
- Calculate the average marginal effects of income over the observations
- What does this marginal effect mean?

Bootstrap

Summary of statistics so far



- Which component one should look at?
- Where do you need a mathematical statistician?

Estimation uncertainty

<u>Every</u> estimate has its uncertainty

How to obtain estimation uncertainty?

- Derive the (asymptotic) variance of the estimator!!
- Estimate the "asymptotic variance"
- Construct confidence interval and tests

$\rightarrow$ New derivations for new models!

Statisticians will never lose their job.... until... Bootstrap!

Drawbacks of asymptotic theory

Classical statistics is based on asymptotic theory

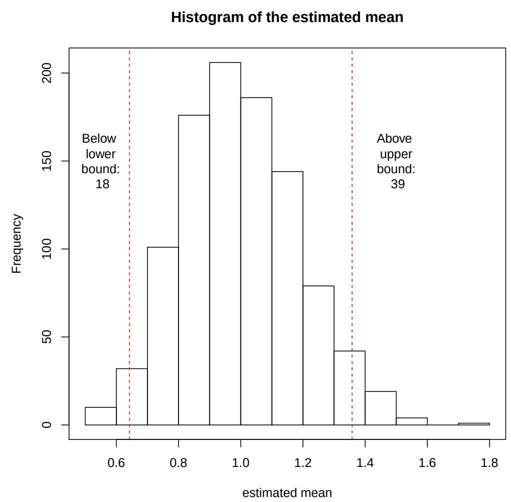
- Assuming large number of observations
- In practice n never goes to infinity
- Asymptotic theory may (sometimes) capture reality very slowly

Example: Estimate mean from simulated data

- DGP: 30 obs simulated from a standard *exponential* distribution (mean=1 and std=1)
- CLT says distribution of sample mean $\approx \mathrm{N}(1, \frac{1}{30})$
- Check CLT by repeating 1000 times:
  - Generate data & construct an point estimate

95% confidence interval should be $1 \pm 1.96\frac{1}{\sqrt{30}}$
$\rightarrow$ 25 cases should be left, and 25 right of this interval

Histogram of the estimated mean



From theory to bootstrap

Handling estimation uncertainty using (asymptotic) theory is not always useful

- You need a mathematical statistician all the time!
- Even if the theory is correct, in practice it may not work

Alternative: handle it by *bootstrap*

- Use "simulation"
- If we can "simulate" the estimator many times, we can get a confidence band (and do hypothesis testing)!
- However, we have only one set of data in practice
- Can we simulate from our data?
  → Bootstrap!

The idea of bootstrap

- An estimator is a function of all observations
- Randomly redraw a new sample from the observations (is called *bootstrap sample*)
  - Draw with replacement
    → the same observation may appear more than once!
  - Sample size is (in general) equal to original sample.
- Recalculate the estimator for each bootstrapped sample (=bootstrapped estimate)
- Pull all bootstrapped estimates together and calculate:
  1. Sample variance as the estimated variance of the original point estimate
  2. A bootstrapped confidence band (rank the bootstrapped estimates)

Bootstrap in Python

- Steps for using bootstrap
  1. Draw the bootstrap sample
  2. Calculate the statistic of interest (make sure you really get a single number!)
  3. Repeat... and collect all statistics
  4. Report distribution and percentiles of statistics
- Example steps 1–3 to bootstrap distribution of sample mean (given dataframe `df`)

```
collectstats = []
for i in range(10000):
    bootstrapsample = df.sample(frac=1, replace=True)
    stat = bootstrapsample["varname"].mean()
    collectstats.append(stat)
```


Display the bootstrap results

- Display the bootstrapped distribution
  ```
  plt.hist(collectstats)
  ```
- Display the bootstrapped confidence interval, eg.
  ```
  np.quantile(collectstats, [0.025, 0.975])
  ```

Notes:

- Above gives 95% interval → can of course get different interval
- This is the most straightforward implementation
- Other methods exist (eg. with bias correction)

Summary bootstrap

- Bootstrap is a powerful idea
- You can get rid of the mathematical statistician for
  - Construct confidence band or test for an estimate
  - Conduct a diagnosis test without knowing the limit distribution
- You still need a mathematical statistician (or yourself) for
  - Construct/Imagine an estimator that works
  - Construct/Imagine a diagnosis statistic that works
  - The "that works" part can also be checked by simulation

You bootstrapped yourself out of the mud!

Hang on! Bootstrap is not always working!

- Bootstrapping the maximum (as an estimator for the endpoint) does **not** work
- Standard bootstrap requires independence of observations (can be generalized)

Assignment 6.3/Take home assignment

- Use the website data
- Continue from In-class Assignment 6.3 and consider the logit model
- Predict the active probability for
  - `exog={'age': 40, 'income': 2000, 'region' : 1}`
  - `exog={'age': 40, 'income': 3000, 'region' : 1}`
- Calculate the difference in predicted probabilities
- Convert the difference into a single number by selecting the [0] element
- Construct the 95% confidence interval for this difference using bootstrap (at least 1000 times)

→ See also the example bootstrap code on Canvas

Before next time

- Nothing to read
- Reconsider/finish the in-class assignments of this week
- Look at (the code of) an additional example/exercise using binary data (next slide)
- Prepare questions for next time (final lecture!)
  - Theory
  - Applications
  - Exercises
  - Final assignment
  - Statistical challenges...
- You can already work on part 3 of the assignment

Additional exercise binary data

Consider the data in `brandchoice.csv` containing brand choice decisions of households together with prices (per ounce) and marketing actions: display (special positioning of product at retailer) and feature (product is mentioned in leaflet)

- Make a model to explain the brand choice (see this as a binary decision)
- What (transformations of) variables should be included?
  Also consider a model containing the price difference (next to other variables)
- What do the coefficients mean?
- Does price have a significant influence?
- How well does the model predict?