Classical statistical models

Abhijit Dasgupta

BIOF 339

Statistical models

> *All models are wrong, but some are useful*

G.E.P. Box

Models

Models are our way of understanding nature, usually using some sort of mathematical expression

Famous mathematical models include Newton's second law of motion, the laws of thermodynamics, the ideal gas law

All probability distributions, like Gaussian, Binomial, Poisson, Gamma, are models

Mendel's laws are models that result in particular mathematical models for inheritance and population prevalence

Models

We use models all the time to describe our understanding of different processes

- Cause-and-effect relationships
- Supply-demand curves
- Financial planning
- Optimizing travel plans (perhaps including traffic like *Google Maps*)
- Understanding the effects of change
    - Climate change
    - Rule changes via Congress or companies
    - Effect of a drug on disease outcomes
    - Effect of education and behavioral patterns on future earnings

Data-driven models

> Can we use data collected on various aspects of a particular context to understand the relationships between the different aspects?

- How does increased smoking affect your risk of getting lung cancer? (causality/association)
    - Does genetics matter?
    - Does the kind of smoking matter?
    - Does gender matter?

Data-driven models

> Can we use data collected on various aspects of a particular context to understand the relationships between the different aspects?

- What is your lifetime risk of breast cancer? (prediction)
  - What if you have a sister with breast cancer?
  - What if you had early menarche?
  - What if you are of Ashkenazi Jewish heritage?

The Gail Model from NCI

Association models

These are more traditional, highly interpretative models that look at how different predictors affect outcome.

- Linear regression
- Logistic regression
- Cox proportional hazards regression
- Decision trees

Since these models have a particular known structure determined by the modeler, they can be used on relatively small datasets

You can easily understand which predictors have more "weight" in influencing the outcome

You can literally write down how a prediction would be made

Predictive models

These are more recent models that primarily look to provide good predictions of an outcome, and the way the predictions are made is left opaque (often called a *black box*)

- Deep Learning (or Neural Networks)
- Random Forests
- Support Vector Machines
- Gradient Boosting Machines

These models require data to both determine the structure of the model as well as make the predictions, so they require lots of data to *train* on

The relative "weight" of predictors in influencing the predictions can be obtained

The effect of individual predictors is not easily interpretable, though this is changing

They require a different philosophic perspective than traditional association models

R for statistical models

We've seen that R is great for data munging and data visualizations

R also can fit a wide variety of statistical models to data.

In fact, most new models first are implemented in R (see CRAN and GitHub)

Today we'll describe some standard popular models. Fitting most models follow the same pattern of code.

Datasets

We will use the `pbc` data from the `survival` package, and the in-built `mtcars` dataset.

```r
library(survival)
str(pbc)
```

```
'data.frame':	418 obs. of  20 variables:
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ time    : int  400 4500 1012 1925 1504 2503 1832 2466 2400 51 ...
 $ status  : int  2 0 2 2 1 2 0 2 2 2 ...
 $ trt     : int  1 1 1 1 2 2 2 2 1 2 ...
 $ age     : num  58.8 56.4 70.1 54.7 38.1 ...
 $ sex     : Factor w/ 2 levels "m","f": 2 2 1 2 2 2 2 2 2 2 ...
 $ ascites : int  1 0 0 0 0 0 0 0 0 1 ...
 $ hepato  : int  1 1 0 1 1 1 1 0 0 0 ...
 $ spiders : int  1 1 0 1 1 0 0 0 1 1 ...
 $ edema   : num  1 0 0.5 0.5 0 0 0 0 0 1 ...
 $ bili    : num  14.5 1.1 1.4 1.8 3.4 0.8 1 0.3 3.2 12.6 ...
 $ chol    : int  261 302 176 244 279 248 322 280 562 200 ...
 $ albumin : num  2.6 4.14 3.48 2.54 3.53 3.98 4.09 4 3.08 2.74 ...
 $ copper  : int  156 54 210 64 143 50 52 52 79 140 ...
 $ alk.phos: num  1718 7395 516 6122 671 ...
 $ ast     : num  137.9 113.5 96.1 60.6 113.2 ...
 $ trig    : int  172 88 55 92 72 63 213 189 88 143 ...
 $ platelet: int  190 221 151 183 136 NA 204 373 251 302 ...
 $ protime : num  12.2 10.6 12 10.3 10.9 11 9.7 11 11 11.5 ...
 $ stage   : int  4 3 4 4 3 3 3 3 2 4 ...
```

The formula interface

Representing model relationships

In R, there is a particularly convenient way to express models, where you have

- one dependent variable
- one or more independent variables, with possible transformations and interactions

```
y ~ x1 + x2 + x1:x2 + I(x3^2) + x4*x5
```

`y` depends on …

- `x1` and `x2` linearly
- the interaction of `x1` and `x2` (represented as `x1:x2`)
- the square of `x3` (the `I()` notation ensures that the `^` symbol is interpreted correctly)
- `x4`, `x5` and their interaction (same as `x4 + x5 + x4:x5`)

Representing model relationships

```
y ~ x1 + x2 + x1:x2 + I(x3^2) + x4*x5
```

This interpretation holds for the vast majority of statistical models in R

- For decision trees and random forests and neural networks, don't add interactions or transformations, since the model will try to figure those out on their own

Our first model

```r
myLinearModel <- lm(chol ~ bili + albumin + copper + sex, data = pbc)
```

Note that everything in R is an object, so you can store a model in a variable name.

This statement runs the model and stored the fitted model in `myLinearModel`

R does not interpret the model, evaluate the adequacy or appropriateness of the model, or comment on whether looking at the relationship between cholesterol and bilirubin makes any kind of sense.

It just fits the model it is given

Our first model

```
myLinearModel
```

```

Call:
lm(formula = chol ~ bili + albumin + copper + sex, data = pbc)

Coefficients:
(Intercept)         bili      albumin       copper         sexf
   221.0571      22.7113      28.9076      -0.1888      -9.7605
```

> Not very informative, is it?

Our first model

```
summary(myLinearModel)
```

```
Call:
lm(formula = chol ~ bili + albumin + copper + sex, data = pbc)

Residuals:
    Min      1Q  Median      3Q     Max 
-580.83  -90.62  -34.79   37.96 1297.16 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 221.0571   135.6962   1.629    0.104    
bili         22.7113     3.2821   6.920 3.14e-11 ***
albumin      28.9076    33.8309   0.854    0.394    
copper       -0.1888     0.1743  -1.083    0.280    
sexf         -9.7605    40.8253  -0.239    0.811    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 214.3 on 277 degrees of freedom
  (136 observations deleted due to missingness)
Multiple R-squared:  0.1638,	Adjusted R-squared:  0.1517 
F-statistic: 13.56 on 4 and 277 DF,  p-value: 4.147e-10
```

> A little better

Our first model

```r
broom::tidy(myLinearModel)
```

```
# A tibble: 5 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)  221.      136.        1.63  1.04e- 1
2 bili          22.7       3.28      6.92  3.14e-11
3 albumin       28.9      33.8       0.854 3.94e- 1
4 copper        -0.189     0.174    -1.08  2.80e- 1
5 sexf          -9.76     40.8      -0.239 8.11e- 1
```

```r
broom::glance(myLinearModel)
```

```
# A tibble: 1 × 12
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.164         0.152  214.      13.6 4.15e-10     4 -1911. 3834. 3856.
# … with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Our first model

```
library(gtsummary)
tbl_regression(myLinearModel)
```

Characteristic	Beta	95% CI[1]	p-value
bili	23	16, 29	<0.001
albumin	29	-38, 96	0.4
copper	-0.19	-0.53, 0.15	0.3
sex			
m	—	—	
f	-9.8	-90, 71	0.8

[1] CI = Confidence Interval

```
library(stargazer)
stargazer(myLinearModel, type='html')
```

	Dependent variable:
	chol
bili	22.711^{***}
	(3.282)
albumin	28.908
	(33.831)
copper	-0.189
	(0.174)
sexf	-9.760
	(40.825)
Constant	221.057

Our first model

We do need some sense as to how well this model fit the data

```
# install.packages('ggfortify')
library(ggfortify)
autoplot(myLinearModel)
```

Our first model

Let's see if we have some strangeness going on

```
ggplot(pbc, aes(x = bili))+geom_density()
```

We'd like this to be a bit more "Gaussian" for better behavior

Our first model

Let's see if we have some strangeness going on

```r
ggplot(pbc, aes(x = log(bili)))+geom_density()
```

Our first model

```
myLinearModel2 <- lm(chol~log(bili) + albumin + copper + sex, data = pbc)
summary(myLinearModel2)
```

```

Call:
lm(formula = chol ~ log(bili) + albumin + copper + sex, data = pbc)

Residuals:
    Min      1Q  Median      3Q     Max 
-448.77  -96.23  -26.77   40.76 1221.21 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 128.3685   132.9579   0.965   0.3351    
log(bili)   124.2339    14.8852   8.346 3.39e-15 ***
albumin      53.6093    33.2245   1.614   0.1078    
copper       -0.3775     0.1743  -2.166   0.0312 *  
sexf         19.6595    39.1715   0.502   0.6161    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 207.4 on 277 degrees of freedom
  (136 observations deleted due to missingness)
Multiple R-squared:  0.2163,	Adjusted R-squared:  0.205 
F-statistic: 19.11 on 4 and 277 DF,  p-value: 6.792e-14
```

Our first model

```
tbl_regression(myLinearModel2)
```

Characteristic	Beta	95% CI[1]	p-value
log(bili)	124	95, 154	<0.001
albumin	54	-12, 119	0.11
copper	-0.38	-0.72, -0.03	0.031
sex			
m	—	—	
f	20	-57, 97	0.6

[1] CI = Confidence Interval

Our first model

```
autoplot(myLinearModel2)
```

Just the residual plot, please

```
autoplot(myLinearModel2, which=1)
```

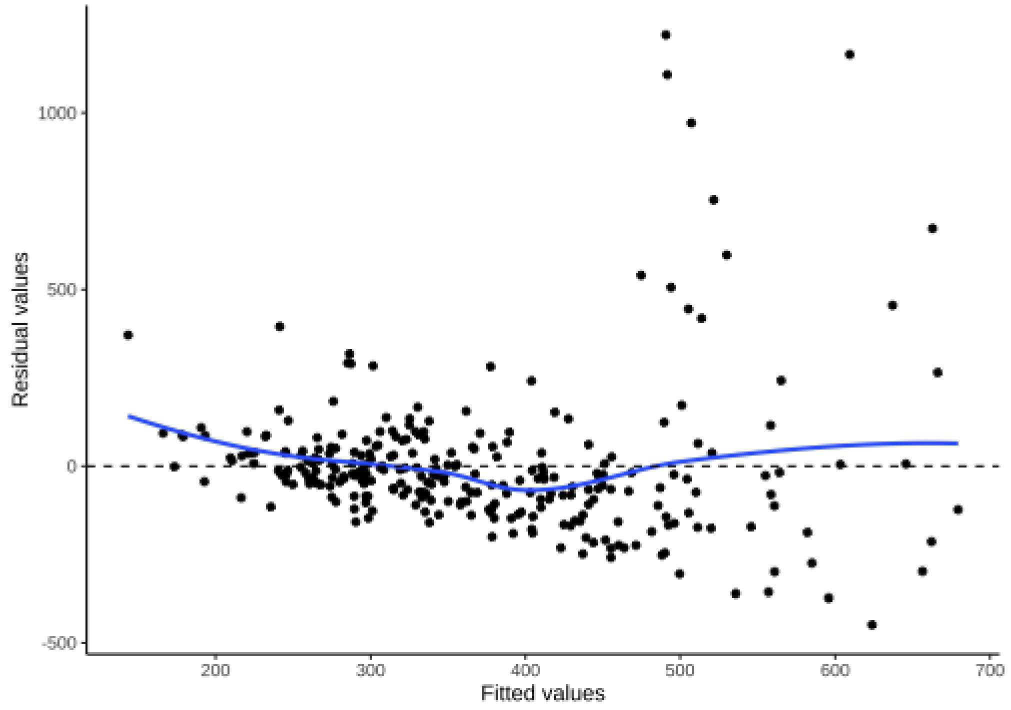
Just the residual plot, please

```
d <- broom::augment(myLinearModel2, newdata=pbc)
d
```

```
# A tibble: 418 × 22
      id  time status   trt   age sex   ascites hepato spiders edema  bili  chol
   <int> <int>  <int> <int> <dbl> <fct>   <int>  <int>   <int> <dbl> <dbl> <int>
 1     1   400      2     1  58.8 f           1      1       1   1    14.5   261
 2     2  4500      0     1  56.4 f           0      1       1   0     1.1   302
 3     3  1012      2     1  70.1 m           0      0       0   0.5   1.4   176
 4     4  1925      2     1  54.7 f           0      1       1   0.5   1.8   244
 5     5  1504      1     2  38.1 f           0      1       1   0     3.4   279
 6     6  2503      2     2  66.3 f           0      1       0   0     0.8   248
 7     7  1832      0     2  55.5 f           0      1       0   0     1     322
 8     8  2466      2     2  53.1 f           0      0       0   0     0.3   280
 9     9  2400      2     1  42.5 f           0      0       1   0     3.2   562
10    10    51      2     2  70.6 f           1      0       1   1    12.6   200
# … with 408 more rows, and 10 more variables: albumin <dbl>, copper <int>,
#   alk.phos <dbl>, ast <dbl>, trig <int>, platelet <int>, protime <dbl>,
#   stage <int>, .fitted <dbl>, .resid <dbl>
```

Just the residual plot, please

```
ggplot(d, aes(x = .fitted, y = .resid))+geom_point()+ geom_smooth(se=F)+
  labs(x = 'Fitted values', y = 'Residual values')+
  geom_hline(yintercept=0, linetype=2) +
  theme_classic()
```



Predictions

```
head(predict(myLinearModel2, newdata = pbc))
```

```
       1        2        3        4        5        6 
560.7384 361.4248 277.4503 333.0571 435.3173 314.7947 
```

The `newdata` has to have the same format and components as the original data the model was trained on

Categorical predictors

```
myLM3 <- lm(chol ~ log(bili) + sex, data = pbc)
broom::tidy(myLM3)
```

```
# A tibble: 3 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    283.       36.6     7.71  2.14e-13
2 log(bili)       99.6      12.1     8.22  7.37e-15
3 sexf            32.5      37.8     0.858 3.92e- 1
```

R has a somewhat unfortunate notation for categorical varialbes here, as `{variable name}{level}`

Logistic regression

The logistic transformation

For an outcome which is binary (0/1), what is really modeled is the probability that the outcome is 1, usually denoted by *p*.

However, we know $0 \leq p \leq 1$, so what if the model gives a prediction outside this range!!

The logistic transform takes *p* to

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

and we model *logit(p)*, which has a range from $-\infty$ to ∞

Logistic regression

Logistic regression is a special case of a generalized linear model, so the function we use to run a logistic regression is `glm`

```r
myLR <- glm(spiders ~ albumin + bili + chol, data = pbc, family = binomial)
myLR
```

```

Call:  glm(formula = spiders ~ albumin + bili + chol, family = binomial, 
    data = pbc)

Coefficients:
(Intercept)      albumin         bili         chol  
  2.3326484   -0.9954927    0.0995915   -0.0003176  

Degrees of Freedom: 283 Total (i.e. Null);  280 Residual
  (134 observations deleted due to missingness)
Null Deviance:	    341.4 
Residual Deviance: 315.2 	AIC: 323.2
```

- We have to add the `family = binomial` as an argument, since this is a special kind of GLM
- All these models only use complete data; they kick out rows with missing data

Logistic regression

```r
broom::tidy(myLR)
```

```
# A tibble: 4 × 5
  term         estimate std.error statistic p.value
  <chr>           <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)  2.33      1.30         1.80  0.0717
2 albumin     -0.995     0.362       -2.75  0.00595
3 bili         0.0996    0.0344       2.89  0.00381
4 chol        -0.000318  0.000615    -0.517 0.605
```

```r
broom::glance(myLR)
```

```
# A tibble: 1 × 8
  null.deviance df.null logLik   AIC   BIC deviance df.residual  nobs
          <dbl>   <int>  <dbl> <dbl> <dbl>    <dbl>       <int> <int>
1          341.     283  -158.  323.  338.     315.         280   284
```

Logistic regression

```
tbl_regression(myLR)
```

Characteristic	log(OR)[1]	95% CI[1]	p-value
albumin	-1.0	-1.7, -0.30	0.006
bili	0.10	0.04, 0.17	0.004
chol	0.00	0.00, 0.00	0.6

[1] OR = Odds Ratio, CI = Confidence Interval

Logistic regression

```r
tbl_regression(myLR, exponentiate = TRUE)
```

Characteristic	OR[1]	95% CI[1]	p-value
albumin	0.37	0.18, 0.74	0.006
bili	1.10	1.04, 1.19	0.004
chol	1.00	1.00, 1.00	0.6

[1] OR = Odds Ratio, CI = Confidence Interval

Predictions from logistic regression

```
head(predict(myLR))
```

```
          1           2           3           4           5           6 
 1.10554163 -1.77506554 -1.04814132 -0.09414055 -0.93144911 -1.62851203 
```

These are on the "wrong" scale. We would expect probabilities

```
head(predict(myLR, type='response'))
```

```
        1         2         3         4         5         6 
0.7512970 0.1449135 0.2595822 0.4764822 0.2826308 0.1640343 
```

or you can use `plogis(predict(myLR))` for the inverse logistic transform

Model selection

How to get the "best" model

Generally getting to the best model involves

- looking at a lot of graphs
- Fitting lots of models
- Comparing the model fits to see what seems good

Sometimes if you have two models that fit about the same, you take the smaller, less complex model (Occam's Razor)

Generally it is not recommended that you use automated model selection methods. It screws up your error rates and may not be the right end result for your objectives

> Model building and selection is an art

Clues to follow

You can look at the relative weights (size of coefficient and its p-value) of different predictors

- These weights will change once you change the model, so be aware of that

You can trim the number of variables based on collinearities

- If several variables are essentially measuring the same thing, use one of them

You can look at residuals for clues about transformations

You can look at graphs, as well as science, for clues about interactions (synergies and antagonisms)

Automated model selection

```
# install.packages('leaps')
library(leaps)
mtcars1 <- mtcars %>% mutate(across(c(cyl, vs:carb), as.factor))
all_subsets <- regsubsets(mpg~., data = mtcars1)
all_subsets
```

```
Subset selection object
Call: regsubsets.formula(mpg ~ ., data = mtcars1)
16 Variables  (and intercept)
      Forced in Forced out
cyl6      FALSE      FALSE
cyl8      FALSE      FALSE
disp      FALSE      FALSE
hp        FALSE      FALSE
drat      FALSE      FALSE
wt        FALSE      FALSE
qsec      FALSE      FALSE
vs1       FALSE      FALSE
am1       FALSE      FALSE
gear4     FALSE      FALSE
gear5     FALSE      FALSE
carb2     FALSE      FALSE
carb3     FALSE      FALSE
carb4     FALSE      FALSE
carb6     FALSE      FALSE
carb8     FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
```

Automated model selection

Which has the best R^2?

```
ind <- which.max(summary(all_subsets)$adjr2)
summary(all_subsets)$which[ind,]
```

```
(Intercept)        cyl6        cyl8        disp          hp        drat 
       TRUE        TRUE       FALSE       FALSE        TRUE       FALSE 
         wt        qsec         vs1         am1       gear4       gear5 
       TRUE       FALSE        TRUE        TRUE       FALSE       FALSE 
      carb2       carb3       carb4       carb6       carb8 
      FALSE       FALSE       FALSE       FALSE       FALSE 
```