Chapter 4: More on ML-based Inference

S352: Data Modeling and Inference

Department of Statistics, Indiana University

A Review on Tests and Intervals

Chapter 3 and Chapter 4.3

MLE and inference

There are two classical ways to do inference using maximum likelihood estimates.

- **Wald inference:** Approximate the distribution of the MLE with a Normal distribution.

$$\hat{\theta} \approx N(\theta, \mathrm{V}(\hat{\theta})), \qquad \hat{\mathrm{V}}(\hat{\theta}) = -\frac{1}{\mathcal{I}(\hat{\theta})} \quad (\text{or } \frac{1}{\textit{Hessian}})$$

$$\begin{pmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \end{pmatrix} \approx N\left(\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}, \boldsymbol{V}(\hat{\boldsymbol{\theta}}) \right), \quad \hat{\boldsymbol{V}}(\hat{\boldsymbol{\theta}}) = -\mathcal{I}(\hat{\boldsymbol{\theta}})^{-1} = \begin{pmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{pmatrix} \quad (\text{or Hessian}^{-1})$$

- **Likelihood ratio inference:** Use something called the“likelihood ratio test statistic.”

Tests so far

- *z*-**test and** *t*-**test** for μ (from S350)
- **Wald tests:** See if your parameter estimate is close to its null values relative to the size of the standard error. We reject $H_0 : \theta = \theta_0$ when W is big.

$$Z = \frac{\hat{\theta} - \theta_0}{\sqrt{\hat{\mathrm{V}}(\hat{\theta})}} \approx N(0, 1^2), \qquad W = Z^2 = \frac{(\hat{\theta} - \theta_0)^2}{\hat{\mathrm{V}}(\hat{\theta})} \approx \chi^2(1)$$

- **Likelihood ratio tests:** See if the maximum log-likelihood under the null comes sufficiently close to the unrestricted maximum log-likelihood.

$$X = 2\left(l(\hat{\boldsymbol{\theta}}) - l(\hat{\boldsymbol{\theta}}_0)\right) \approx \chi^2(r)$$

  - G-test for multinomial data:

$$G = 2\sum_{i=1}^{m} o_i \cdot \log\left(\frac{o_i}{e_i}\right) \approx \chi^2(m-1)$$

We reject $H_0 : \theta = \theta_0$ when X is big; reject $H_0 : \mathbf{p} = \mathbf{p_0}$ when G is big.

Intervals so far

- *z*-**CI or** *t*-**CI** for μ (from S350)

$$\bar{x} \pm \texttt{qnorm}(1 - \frac{\alpha}{2}) \cdot \frac{\sigma}{\sqrt{n}} \qquad \text{or} \qquad \bar{x} \pm \texttt{qt}(1 - \frac{\alpha}{2}, \ \texttt{df = n-1}) \cdot \frac{s}{\sqrt{n}}$$

- **Wald CI** for θ

$$\hat{\theta} \pm \texttt{qnorm}(1 - \frac{\alpha}{2}) \cdot \hat{se}(\hat{\theta}), \qquad \hat{se}(\hat{\theta}) = \sqrt{\hat{\mathbb{V}}(\hat{\theta})} \quad (\text{or } \frac{1}{\sqrt{Hessian}})$$

- **Profile Likelihood CI** for θ: find values that give a log-likelihood within half of qchisq(1-alpha, df) (1.92 for a scalar parameter at 95%) of the peak.
  - Binomial data: library(binom) and binom.profile(y, n)
  - Other data: library(Bhat)

```
NLL <- function(par){
    lik <- ...
    return(-sum(log(lik)))
}
control.list = list(label, est, low, upp)
plkhci(control.list, NLL, label, prob)
```

Wald vs. LR

- Both kinds of tests rely on having sufficiently large sample sizes.
- The Wald test is arguably more intuitive, and certainly easier to explain to non-statisticians. But things can go horribly wrong more easily. . .
- The Wald statistic is **not invariant to transformation**. That is, different transformations can give very different test results.
- In contrast, the likelihood ratio test *is* invariant to transformation: the underlying statistical model remains the same.

Wald and transformations

Suppose we're testing $H_0 : \theta = 0$ and we get an MLE $\hat{\theta} = 1$ with $\mathrm{Var}(\hat{\theta}) = 0.01$. The Wald statistic is

$$W = \frac{(1-0)^2}{0.01} = 100.$$

The P-value is

```
> 1 - pchisq(100, df = 1)
[1] 0
```

Wald and transformations

Now let $\xi = \theta^{25}$. The null is $H_0 : \xi = 0^{25} = 0$. The MLE is $\hat{\xi} = 1^{25} = 1$. The variance of $\hat{\xi}$ can be found using the delta method (Millar 4.2). It turns out that

$$\mathrm{Var}(\hat{\xi}) = 25^2 \times 0.01 = 6.25.$$

The Wald statistic is now

$$W = \frac{(1-0)^2}{6.25} = 0.16.$$

The *P*-value is

```
> 1 - pchisq(0.16, df = 1)
[1] 0.6891565
```

We're testing the same hypothesis as before, but we're getting a completely different *P*-value. Seems bad!

Model Assumptions vs Model Checking

more formal ways to assess the model

another use of G-test: test discrete distributions

What you've learned so far

- s350 data and distribution (Trosset ch7): histogram: `hist(x)`, kernel density estimate: `plot(density(x))`, QQ plot: `qqnorm(x)`, ...
- s350 t-tests (Trosset ch10-11): check normality using QQ plot
- s350 ANOVA (Trosset ch12): check normality (QQ plot) and homoscedasticity
- s350 regression (Trosset ch15): check linearity between X and Y, independence, homoscedasticity and normality of errors
- s352 ch2: fit a model and check with simulation, compare histograms
  1. Count data: Poisson vs Negative Binomial vs Zero-Inflated Poisson vs Zero-Inflated Negative Binomial
  2. Right skewed continuous data: Exponential vs Gamma
- s352 ch3: Wald test, LR test, G-test
  1. Data from standard normal, $N(\mu = 0, \sigma^2 = 1^2)$?
  2. Exp(λ) or Gamma(k, λ), or is $k = 1$?
  3. Test of fully-specified multinomial distribution

Uses of G-tests

	Categories	1	2	3	$\cdots$	m
$H_0 \rightarrow$	$\underline{p_0}$	p_1	p_2	p_3	$\cdots$	p_m
data $\rightarrow$	O_i	O_1	O_2	O_3	$\cdots$	O_m
$n\underline{p_0} \rightarrow$	e_i	e_1	e_2	e_3	$\cdots$	e_m

$\sum P_i = 1$

H_0: $\underline{p_0}$ is given.

- Tests of fully-specified probability models: "Do the colors of these M&Ms follow these probabilities?" We've covered these in ch3. $G \approx \chi^2(m-1)$
- Tests of independence of categorical variables: "Are handedness and gender independent?" These are useful, but the traditional (Pearson's) chi-squared test is far more popular. (S350 Ch13)
- Tests of discrete distributions. "Does this data follow a Binomial distribution?" or "Does this data follow a Poisson distribution?" We'll focus on these in this chapter.

Partially specified probability models

In H_0, $\underline{P_0}$ is not fully known.

With a **partially-specified probability model**, you *cannot* write down the null hypothesis probability of being in each category before collecting the data.

Instead, you need to *estimate* parameters from the data.

e.g. Does this data X follow a Binomial distribution?

- Null hypothesis: X is Binomial(n, p)
- Alternative hypothesis: X follows some other distribution

You need to know n and p to get the PMF of X. You might know n before looking at the data, but you don't know p...

Example: Is this Binomial?

We observe the genders of the first three children in families with 3 or more kids in Denmark.

- 0 girls: 23,236
- 1 girl: 58,529
- 2 girls: 53,908
- 3 girls: 18,770

Categories

	0 girl	1 girl	2 girls	3 girls
$H_0 \to \hat{p}_0$	$\hat{p}_i = f(x_i) = \binom{n}{x_i}\hat{p}^{x_i}(1-\hat{p})^{n-x_i} = dbinom(x_i, n, \hat{p})$			
O_i				
$N\hat{p}_0 = e_i$				

$m = 4$

Does the number of girls follow a binomial distribution? $H_0 : X_i \sim Binomial(n, p)$. $X_i = 0, 1, 2, 3.$

- We know $n = 3$. (Note the sample size is $N =$ 154, 443; $i = 1, \ldots, 154, 443$)
- To estimate p, observe that there are 222,655 girls out of 463,329 children, giving a sample proportion of about 0.4806. (MLE of p under H_0) $\hat{p}_{MLE} = \frac{\# \text{ of girls}}{\# \text{ of children}}$
- Then you can use `dbinom()` to find the probabilities of 0, 1, 2, and 3 girls.

G-test for partially specified probability models

Recall that for G-tests, the degrees of freedom is the difference in the number of free parameters under the null relative to the alternative.

With a partially-specified probability model, then under the null, the G-statistic has a chi-squared distribution with $\mathbf{m-1-d}$ degrees of freedom, where

- m is the number of categories (the -1 is because probabilities sum to 1)
- d is the minimum number of parameters estimated from the data

If the null is that X is Binomial$(3, p)$, then $m = 4$ (since the categories are 0, 1, 2, 3) and $d = 1$ (we needed to estimate one parameter, p).

Degrees of freedom will be $4 - 1 - 1 = 2$. (Under the alt, we can vary 3 probabilities freely, whereas under the null, we can only vary one parameter, p.)

Example: Are girls Binomial?

Observed counts:

o_i: 23236, 58529, 53908, 18770

Expected counts:

e_i: 21646.5, 60077.5, 55579.6, 17139.5

G-test statistic:

natural log !

$$2 \sum o_i \log(o_i / e_i) \approx 355.$$

P-value: `1 - pchisq(355, df = 2)` ≈ 0.

The data is *not* consistent with a binomial distribution. There are too many 0-girl and 3-girl families.

Example: Are goals Poisson?

Do the goals scored in games in the 2018-19 EPL soccer season follow a Poisson distribution?

```
> epl1819 <- read.csv("epl1819.csv")
> TotalGoals <- epl1819$FTHG + epl1819$FTAG
> table(TotalGoals)
TotalGoals
 0  1  2  3  4  5  6  7  8
22 55 99 84 65 32 16  5  2
```

$m = 8$!

categories

7 or more

	0	1	2	3	4	5	6	7 8 9 10 ...
$H_0 \to \hat{p}_0$	$\hat{p}_i = e^{-\hat{\lambda}} \cdot \frac{\hat{\lambda}^x}{x!} = \text{dpois}(x, \hat{\lambda})$							
O_i								
$n\hat{p}_0 = e_i$								

We previously did simulations which looked okay. Can we do a more formal test?

$$H_0 : X_i \sim Poisson(\lambda)$$

$\hat{\lambda}_{MLE} = \bar{X}$

PMF: $f(x) = e^{-\lambda} \cdot \frac{\lambda^x}{x!}$,

Example: Are goals Poisson?

Estimate λ:

```
lambda.hat <- mean(TotalGoals)
```

Find the Poisson probabilities and expected values:

```
> p <- dpois(0:11, lambda.hat)
> expected <- 380 * p
> round(expected, 2)
 [1] 22.63 63.83 90.03 84.66 59.71 33.69 15.84  6.38  2.25  0.71
      0.20  0.05
```

0 1 2 3 4 5 6

7 or more group!

too small! < 5

BUT high numbers of goals, the expected counts are too small and make the test noisy. A rule of thumb is that expected counts should be at least 5. . .

Example: Are goals Poisson?

Instead, create a "7 or more goals" category:

```
p <- dpois(0:6, lambda.hat)
expected <- 380 * p
expected <- c(expected, 380 - sum(expected))
```

Now you can do the *G*-test as usual. (With how many degrees of freedom?)

$df = m - 1 - d$

$= 8 - 1 - 1 = 6$

Model Selection

more formal ways to compare

Model selection

In many situations, we may have one particular null hypothesis in mind. Other times, we may want to do **model selection** to choose between two or more models:

- Should we model this count data using a Poisson or a zero-inflated Poisson or a Negative Binomial or a zero-inflated Negative Binomial or. . .
- Does the best regression model have predictors x_1 and x_2, or x_1, x_2 and x_3 (or x_1 and x_3, or. . .)
- Should we model the data with a linear or nonlinear regression?

Sometimes hypothesis testing is a principled way to do this model selection, but usually not.

Model selection methods

- Fitting models on a training set and evaluating them on a test set
- AIC (Akaike's Information Criterion)
- BIC (Bayesian Information Criterion)
- Lots of other things with "IC" in the name
- Cross-validation
- Model averaging
- Bayesian's way of variable/model selection

and many more...

AIC and maximum likelihood

For now, let's look at Akaike's Information Criterion (AIC), which is simple and widely used.

Recall that the main point of this course is that we want to maximize likelihood (or log-likelihood.) So you might think we can just choose the model that gives the highest maximum log-likelihood.

However, this isn't always a fair comparison. For example, for positive data, a gamma distribution will *always* give a maximum log-likelihood that's at least as big as an Exponential log-likelihood. But some things really are Exponential!

Overfitting

Fitting a model that's more complex than you need is called **overfitting**.

Generally it means predictions will be less accurate than those made with a simpler model.

To reduce the risk of overfitting, we want to penalize complex models with more parameters than simple models.

But how much should we penalize an extra parameter relative to the log-likelihood? Is there a magic number?

AIC: 2 is the magic number

The **AIC** of a model is

$$\mathrm{AIC} = -2l(\hat{\boldsymbol{\theta}}) + 2s$$

where:

- $\hat{\boldsymbol{\theta}}$ is the MLE for the model;
- $l(\hat{\boldsymbol{\theta}})$ is the maximum value of the log-likelihood;
- s is the number of parameters estimated from the data.

We want to choose the model with the *smallest* AIC.

Example: Roots

The file roots.txt contains the data from Millar p. 57 concerning the number of roots on apple cultivars.

```
> roots <- scan("roots.txt")
Read 40 items
> table(roots)
roots
0  1  2  3  4  5  6  7  9
19  2  2  4  3  1  4  3  2
```

Which of our favorite count data models would look best?

Roots: Poisson model

The MLE of a Poisson(λ) model is

```
lambda.hat <- mean(roots)
```

The log-likelihood at the MLE is

```
pois.ll <- sum(log(dpois(roots, lambda.hat)))
```

The AIC is

```
-2 * pois.ll + 2 * 1
```

giving 223.11.

Is that good? Bad? We don't know until we have something to compare it to. . .

Roots: Negative Binomial

It's about time that we wrote a general function to find the MLE of a Negative Binomial:

```
nbinom.nll <- function(params, data){
    size <- params[1]
    prob <- params[2]
    lik <- dnbinom(data, size, prob)
    loglik <- log(lik)
    nll <- -sum(log(lik))
    return(nll)
}
```

Roots: Negative Binomial

Get the MOMs for initial guesses:

```
x.bar <- mean(roots)
sigma2.hat <- mean(roots^2) - x.bar^2
p.mom <- x.bar / sigma2.hat
r.mom <- p.mom / (1 - p.mom) * x.bar
```

Use `optim()`:

```
roots.nb.mle <- optim(par = c(r.mom, p.mom),
                       fn = nbinom.nll,
                     data = roots)
```

Roots: Negative Binomial

`optim()` gives you both the MLEs and the negative log-likelihood at the MLEs:

```
> roots.nb.mle$par
[1] 0.4790240 0.1635703
> roots.nb.mle$value
[1] 81.00005
```

We estimated two parameters. Thus the AIC is

```
2 * roots.nb.mle$value + 2 * 2
```

which gives 166.00. That's a lot lower (i.e. better) than the Poisson.

Roots: Zero-inflated Poisson model

Write a function to find the MLE of a ZIP:

```
pois0.nll <- function(pars, data){
    p <- pars[1]
    lambda <- pars[2]
    pois.lik <- dpois(data, lambda)
    real.lik <- rep(NA, length(data))
    which0 <- (data == 0)
    real.lik[which0 == FALSE] = (1 - p) * pois.lik[which0 == FALSE]
    real.lik[which0 == TRUE] = p + (1 - p) * pois.lik[which0 == TRUE]
    loglik <- log(real.lik)
    return(-sum(loglik))
}
```

Roots: Zero-inflated Poisson model

Make some initial guesses and use optim():

```
roots.zip.mle <- optim(par = c(0.475, 2.45),
                        fn = pois0.nll,
                      data = roots)
```

Again, roots.zip.mle$value gives the negative log-likelihood.

The ZIP has two parameters, so the AIC is

```
2 * roots.zip.mle$value + 2 * 2
```

This gives 153.76. Better still!

An alternative: BIC

AIC imposes a penalty of 2 units per parameter (on the log-likelihood scale.)
This is pretty light, so AIC sometimes overfits.

The **Bayesian information criterion** imposes (for reasonable sample sizes) a heavier penalty:

$$\mathrm{BIC} = -2l(\hat{\theta}) + s \log n$$

where n is the sample size. (This sometimes underfits. `log(30) ≈ 3.40`)

- If the models you're comparing have the same number of parameters, then AIC and BIC will select the same model.
- If the models you're comparing use different distributions, pretty often the differences in log-likelihood will be big and AIC and BIC will often select the same model.

Roots: BICs

```
> n <- length(roots)
> -2 * pois.ll + 1 * log(n)
[1] 224.7979
> 2 * roots.nb.mle$value + 2 * log(n)
[1] 169.3779
> 2 * roots.zip.mle$value + 2 * log(n)
[1] 157.1385
```

Regression models

The simple linear regression model

We have measurements of variables x and Y. Suppose we're willing to treat the x's as fixed while the Y's are random (for example, because we want to predict the Y's from the x's.)

The **simple linear regression** model, sometimes called the **ordinary least squares (OLS)** model is

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

where the errors ϵ_i are IID Normal$(0, \sigma^2)$.

Simple linear regression assumptions

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

where the errors ϵ_i are IID Normal$(0, \sigma^2)$.

We can break down the assumptions of this regression model into parts, and put them in order of importance:

- **Linear trend:** $E(Y_i) = \beta_0 + \beta_1 x_i$.
- **Independence of errors:** The ϵ_is are independent.
- **Equal variance of errors (homoskedasticity):** The ϵ_is all have the same variance, σ^2.
- **Normality of errors:** The ϵ_is follow a normal distribution.

(Does it matter that these assumptions are rarely if ever literally true? We'll worry about that some other time. . .)

Example: Brother-sister heights

Pearson and Lee (1903) collected data on brother-sister heights (in inches):

Bro's height (x)	71	68	66	67	70	71	70	73	72	65	66
Sis's height (y)	69	64	65	63	65	62	65	64	66	59	62

Let the x's be the brothers' heights and the y's be the sisters' heights.

How do we fit a regression model?

Regression likelihood

According to our model, each Y_i comes from a Normal distribution with mean $\beta_0 + \beta_1 x_i$ and variance σ^2. For given $\beta_0, \beta_1, \sigma^2$, we can find the likelihood/log-likelihood/negative log-likelihood:

```
bros <- c(71, 68, 66, 67, 70, 71, 70, 73, 72, 65, 66)
sis <- c(69, 64, 65, 63, 65, 62, 65, 64, 66, 59, 62)
siblings.nll <- function(pars){
    beta0 <- pars[1]
    beta1 <- pars[2]
    sigma2 <- pars[3]
    lik <- dnorm(sis, beta0 + beta1 * bros, sqrt(sigma2))
    loglik <- log(lik)
    return(-sum(loglik))
}
```

Regression MLEs

To find MLEs, we can use `optim()` as usual:

```
> optim(c(0, 1, 5), siblings.nll)
$par
[1] 27.628087  0.527125  4.131119

$value
[1] 23.41078
```

So our regression model is:

$$\text{Sister's height} = 27.6 + 0.527 \times \text{Brother's height} + \epsilon_i$$

where ϵ_i is Normal with mean 0 and variance 4.13.

Least squares

Suppose we're primarily interested in $\hat{\beta}_0$ and $\hat{\beta}_1$.
With a bit of calculus, it can be shown that these MLEs are the values that minimize

$$\sum_{i=1}^{n} [Y_i - (\beta_0 + \beta_1 x_i)]^2$$

i.e. the sum of squared (vertical) distances from the observations to the line.

We call this the method of **least squares**.

It turns out that this problem is *much* easier computationally than using `optim()`. So we can (for now) forget about using `optim()` for regression problems.
(For somewhat more subtle reasons, it also means we can be a lot more relaxed about the normality of errors assumption, which is the one that's least likely to be close to true.)

lm()

In R, the `lm()` function finds the least squares solution quickly for sample sizes into the tens of millions.

```
> lm(sis ~ bros)

Call:
lm(formula = sis ~ bros)

Coefficients:
(Intercept)         bros
27.635        0.527
```

summary(lm())

We can get inference for the parameters using the summary function.

```
> siblings.lm <- lm(sis ~ bros)
> summary(siblings.lm)
     (...)
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)  27.6351    18.0371   1.532   0.1599
bros          0.5270     0.2612   2.018   0.0744 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.247 on 9 degrees of freedom
Multiple R-squared:  0.3114,    Adjusted R-squared:  0.2349
F-statistic:  4.07 on 1 and 9 DF,  p-value: 0.07442
```

Danger! You need linearity and independence and equal variance to be pretty close to true for this inference to be valid!

predict()

To use the predict() function to make predictions from the regression model, input the *x*-values for the predictions as a data frame.

```
> some.bros <- data.frame(bros = c(66, 68, 70, 72))
> predict(siblings.lm, newdata = some.bros)
1        2        3        4
62.41892 63.47297 64.52703 65.58108
```

The multiple linear regression model

Now suppose we have measurements of variables x_1, x_2, and Y, and we're willing to treat the x_1's and x_2's as fixed while the Y's are random.
The **multiple linear regression** model is

$$Y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \epsilon_i$$

where $x_{1,i}$ denotes the value of x_1 for the ith observation and the errors ϵ_i are IID Normal$(0, \sigma^2)$.

Of course we can extend this to 3 or 4 or more x-variables (as long as the number of x-variables doesn't approach the sample size.)

Example: Teaching evaluations

The data set `https://www.openintro.org/book/statdata/evals.csv` contains a bunch of variables concerning 463 UT Austin professors,[1] but for now we'll look at:

- `score`: Average teaching evaluation score for a professor, on a 1 to 5 scale.
- `age`: Age of the professor.
- `bty_avg`: The average "beauty rating" of the professor on a 1 to 10 scale, as assigned by a panel of six students.

Do age and beauty help predict a professor's evaluation score?

[1] See `https://www.openintro.org/book/statdata/?data=evals` for more details.

Draw some pictures

```r
evals <- read.csv("https://www.openintro.org/book/statdata/evals.csv")
plot(evals$age, evals$score)
plot(evals$bty_avg, evals$score)
plot(evals$age, evals$bty_avg)
```

If you want to download and learn the `ggplot2` package, you can draw nicer graphs. (Take STAT-S 470 to learn more.)

Teaching evaluations: In R

Fitting a model with both age and beauty as predictors is easy enough:

```
> lm(score ~ age + bty_avg, data = evals)

Call:
lm(formula = score ~ age + bty_avg, data = evals)

Coefficients:
(Intercept)          age      bty_avg
4.054732    -0.003059     0.060656
```

The model, as estimated by least squares, is

$$\text{Score} = 4.05 - 0.00306 \times \text{Age} + 0.0607 \times \text{Beauty} + \text{error}$$

Model selection

Is this model better than one with one predictor? One with zero predictors?
There are lots of ways to compare models. Here are two:

- Use AIC.
- Split the data into a training set and a test set. Fit models on the training set. See which of the models performs best on the test set, as measured by e.g. mean squared error.

AIC for regression

Fortunately there's a chunky old `AIC()` function. As before, low `AIC()` is good:

```
> AIC(lm(score ~ 1, data = evals))
[1] 752.951
> AIC(lm(score ~ age, data = evals))
[1] 749.6163
> AIC(lm(score ~ bty_avg, data = evals))
[1] 738.4449
> AIC(lm(score ~ age + bty_avg, data = evals))
[1] 739.1194
```

(There's also a `BIC()` function if you prefer that.)

Data splitting: Training and test sets

Put the data in random order:

```
set.seed(352)
n <- nrow(evals)
evals.rand <- evals[sample(n),]
```

For regression-type model, a 70-30 split between the training and test sets is about right:

```
evals.train <- evals.rand[1:324,]
evals.test <- evals.rand[325:463,]
```

Fitting and predicting

Fit some models:

```
lm.train.null <- lm(score ~ 1, data = evals.train)
lm.train.age <- lm(score ~ age, data = evals.train)
lm.train.bty <- lm(score ~ bty_avg, data = evals.train)
lm.train.both <- lm(score ~ age + bty_avg, data = evals.train)
```

Make some predictions:

```
pred.null <- predict(lm.train.null, newdata = evals.test)
pred.age <- predict(lm.train.age, newdata = evals.test)
pred.bty <- predict(lm.train.bty, newdata = evals.test)
pred.both <- predict(lm.train.both, newdata = evals.test)
```

Mean squared error

Use **mean squared error** to see how well our models did (low MSE is good):

$$\mathrm{MSE} = \frac{1}{n_{\mathrm{test}}} \sum_i (y_i - \hat{y}_i)^2$$

where the $\hat{y}_i$'s are the predictions and the sum is taken over the test set. Our "correct" y_i's are in `evals.test$score`.

```
mean((evals.test$score - pred.null)^2)
mean((evals.test$score - pred.age)^2)
mean((evals.test$score - pred.bty)^2)
mean((evals.test$score - pred.both)^2)
```

I again found that using beauty alone did the best. (It can depend on the random seed; it would be better to repeat this a hundred times and take the average.) Once you've decided which model you like best, re-fit it on the *full* data set.

Regression with a categorical predictor

Teaching evaluations again

In the `evals` data set, there's a `rank` variable that categorizes instructors by academic rank:

- `tenured`
- `tenure track`
- `teaching`

On average, are there differences in evaluation `score` between these three different categories of instructor?

Using aggregate()

aggregate() can calculate summary statistics by group:

```
> evals <- read.csv("https://www.openintro.org/book/statdata/evals.csv")
> aggregate(score ~ rank, FUN = mean, data = evals)
```

```
rank    score
1     teaching 4.284314
2 tenure track 4.154630
3      tenured 4.139130
```

Using lm()

With one categorical predictor, lm() will just predict the group means, phrased in a different (weird) way. One group (by default, the first in alphabetical order) is taken to be **the baseline**. Each of the other groups has a coefficient that give that group's mean *relative* to that baseline.

```
> rank.lm <- lm(score ~ rank, data = evals)
> rank.lm

Call:
lm(formula = score ~ rank, data = evals)

Coefficients:
(Intercept)  ranktenure track       ranktenured
4.2843           -0.1297           -0.1452
```

Using `predict()`

To use `predict()`, create a new data frame with categories corresponding to those in the original data set.

```
> rank.df <- data.frame(rank = c("teaching", "tenure track", "tenured"))
> predict(rank.lm, newdata = rank.df)
1        2        3
4.284314 4.154630 4.139130
```

Inference: anova()

If the *k*-sample testing (Chapter 12) was covered in STAT-S 350, you'll remember that we can test the null hypothesis that all the group have the same population mean using an *analysis of variance*. This is straightforward to do once you've fitted the LM:

```
> anova(rank.lm)
Analysis of Variance Table

Response: score
Df  Sum Sq Mean Sq F value  Pr(>F)
rank        2   1.589 0.79457  2.7061 0.06786 .
Residuals 460 135.065 0.29362
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

(though recall the ANOVA makes assumptions. . .)

AIC()

We can compare the model using rank to a null model using AIC:

```
> AIC(lm(score ~ 1, data = evals))
[1] 752.951
> AIC(rank.lm)
[1] 751.5352
```

Remember! Choosing a model for prediction and choosing a model based on "significance" may not give the same results.

Predicting teaching evaluations, so far...

Based on the data at `https://www.openintro.org/book/statdata/evals.csv` :

- There's strong evidence that the numeric variable beauty rating (`bty_avg`) helps to predict average evaluation (`score`.)
- There's good evidence that the numeric variable age helps to predict `score`.
- There's weakish? evidence that the categorical variable `rank` helps to predict `score`.

A categorical x and a numeric x

Now suppose we allow ourselves to use the beauty measure `bty_avg` as an additional predictor. There are lots of ways this could turn out:

- The model with just `bty_avg` might turn out best: perhaps students give teaching faculty higher evaluations solely because they're better looking?
- The model with both `rank` and `bty_avg` might turn out best: perhaps students give teaching faculty higher evaluations because they're better at teaching than tenure-track faculty, and students give higher evaluations to better-looking teaching faculty than less good-looking teaching faculty, higher evaluations to better-looking tenure-track faculty than less good-looking tenure-track faculty, etc.
- Or maybe teaching profs are ugly but happen to teach better. Or maybe tenured professors teach the best but are old so get lower beauty evaluations. Or. . .

We can't be sure which of these *causal* stories is correct based on this observational data alone. But we can try to find as good a *predictive* model as we can.

Fit a multiple regression

We can fit a model that adds separate terms in `rank` and `score`:

```
> btyRank.lm <- lm(score ~ rank + bty_avg, data = evals)
> btyRank.lm

Call:
lm(formula = score ~ rank + bty_avg, data = evals)

Coefficients:
(Intercept)  ranktenure track       ranktenured
3.98155          -0.16070          -0.12623
bty_avg
0.06783
```

Interpreting the regression

```
> coefficients(btyRank.lm)
(Intercept) ranktenure track      ranktenured          bty_avg
3.9815462       -0.1607023      -0.1262267        0.0678259
```

To make a prediction:

- Start from 3.982
- Subtract -0.161 if they're tenure-track
- Subtract -0.127 if they're tenured
- Add 0.0678 times their beauty score

Three regression lines

```
> coefficients(btyRank.lm)
(Intercept) ranktenure track      ranktenured          bty_avg
3.9815462        -0.1607023       -0.1262267        0.0678259
```

We have three parallel regression lines. For teaching faculty:

$$\text{Score} = 3.982 + 0.0678 \times \text{beauty} + \text{error}$$

For tenure-track faculty:

$$\text{Score} = 3.821 + 0.0678 \times \text{beauty} + \text{error}$$

For tenured faculty:

$$\text{Score} = 3.855 + 0.0678 \times \text{beauty} + \text{error}$$

Visualization

```
plot(evals$bty_avg, evals$score,
        pch = ".",
       main = "Blue: teaching;
             orange: tenure-track;
               gold: tenured")
abline(c(3.982, 0.0678), col = "blue")
abline(c(3.821, 0.0678), col = "orange")
abline(c(3.855, 0.0678), col = "gold")
```

Model with a categorical predictor

Our model can be written as

$$\text{Score} = \beta_0 + \beta_1 \times \text{tenure track} + \beta_2 \times \text{tenured} + \beta_3 \times \text{beauty} + \text{error}$$

where “tenure track” and “tenured” are 1 if you’re in that group and 0 otherwise.
For teaching faculty:

$$\text{Predicted score} = \hat{\beta}_0 + \hat{\beta}_3 \times \text{beauty}$$

For tenure-track faculty:

$$\text{Predicted score} = (\hat{\beta}_0 + \hat{\beta}_1) + \hat{\beta}_3 \times \text{beauty}$$

For tenured faculty:

$$\text{Predicted score} = (\hat{\beta}_0 + \hat{\beta}_2) + \hat{\beta}_3 \times \text{beauty}$$

Model comparison

Compare the AIC of a few models:

```
AIC(lm(score ~ rank, data = evals))
AIC(lm(score ~ bty_avg, data = evals))
AIC(btyRank.lm)
```

Does BIC give the same result?

Model comparisons

Compare the AIC and BIC of the models we've fit so far:

Predictors	AIC	BIC
Age		
Rank		
Beauty rating		lowest
(Age + Rank)		
Age + Beauty rating		
Rank + Beauty rating	lowest	

Different slopes?

Rather than three regression lines with the same slope, it might be predictively better to fit the three lines with *different* regression slopes.

It's hard to draw a plot that lets you assess if this is a good idea in base R, but you can use `ggplot()`:

```
# install.packages("ggplot2")
library(ggplot2)
ggplot(evals, aes(x = bty_avg, y = score, group = rank, color = rank)) +
              geom_point(alpha = 0.5) +
              geom_smooth(method = "lm", se = FALSE) +
              scale_color_viridis_d()
```

Regression with interactions

Interaction model

Our previous model was

$$\text{Score} = \beta_0 + \beta_1 \times \text{tenure track} + \beta_2 \times \text{tenured} + \beta_3 \times \text{beauty} + \text{error}$$

where "tenure track" and "tenured" are 1 if you're in that group and 0 otherwise. We can now try fitting a model with an interaction:

$$\begin{aligned} \text{Score} = \beta_0 + \beta_1 \times \text{tenure track} + \beta_2 \times \text{tenured} + \beta_3 \times \text{beauty} \\ {\color{red} + \beta_4 \times \text{beauty} \times \text{tenure track}} \\ {\color{red} + \beta_5 \times \text{beauty} \times \text{tenured}} \\ + \text{error} \end{aligned}$$

Interaction model in R

This is again easy to fit in R:

```
> btyRank.int.lm <- lm(score ~ rank + bty_avg + rank:bty_avg, data=evals)
> coefficients(btyRank.int.lm)
(Intercept)         ranktenure track              ranktenured
4.09811000              -0.01884686              -0.40910460
bty_avg ranktenure track:bty_avg      ranktenured:bty_avg
0.04171331              -0.02639870               0.06585879
```

For teaching faculty (the baseline), the regression prediction is

$$\text{Predicted score} = 4.098 + 0.0417 \times \text{beauty}$$

Interaction model in R

```
(Intercept)          ranktenure track             ranktenured
4.09811000               -0.01884686              -0.40910460
bty_avg ranktenure track:bty_avg      ranktenured:bty_avg
0.04171331               -0.02639870               0.06585879
```

For tenure-track faculty, the regression line is

$$\begin{aligned}\text{Predicted score} &= 4.098 - 0.019 + 0.0417 \times \text{beauty} - 0.0264 \times \text{beauty} \\ &= 4.079 + 0.0153 \times \mathbf{beauty}\end{aligned}$$

For tenured faculty, the regression line is

$$\begin{aligned}\text{Predicted score} &= 4.098 - 0.409 + 0.0417 \times \text{beauty} + 0.0659 \times \text{beauty} \\ &= 3.689 + 0.1076 \times \mathbf{beauty}\end{aligned}$$

Another graph

```
plot(evals$bty_avg, evals$score, pch = ".",
main = "Blue: teaching; orange: tenure-track; gold: tenured")
abline(c(4.098, 0.0417), col = "blue")
abline(c(4.079, 0.0153), col = "orange")
abline(c(3.689, 0.1076), col = "gold")
```

AIC

```
> AIC(btyRank.lm)
[1] 736.8954
> AIC(btyRank.int.lm)
[1] 734.9368
```

It's tempting to now say something like "Beauty doesn't matter much for tenure-track faculty but matters a lot for tenured faculty." BUT:

- Causal statements are dangerous! There are still lots of confounding variables! Tenured faculty tend to be older than tenure-track faculty!
- Even if you only wanted to make a descriptive statement, we haven't proven significance. In fact, inference for regression interactions is non-trivial and we'll leave it for STAT-S 431.

You *can* say "According to our model, which might be wrong. . . "

Warning: Don't fit this model!

```
lm(score ~ bty_avg + rank:bty_avg, data = evals)
```

This fits regression lines with the same y-intercept but different slopes. But there's no reason to think that the regression lines should magically meet at zero.

- In general, only include an interaction if you're also including the variables that go into that interaction,[2] regardless of what AIC says.

[2] Unless you have a good theoretical reason why you can omit one of the variables going into the interaction.

Can we fit an interaction between numeric variables?

Yeah, sure.

```
> btyAge.int.lm <- lm(score ~ bty_avg + age + bty_avg:age, data = evals)

> coefficients(btyAgek.int.lm)
(Intercept)      bty_avg          age  bty_avg:age
5.156076720 -0.187800306 -0.026128326  0.005317673

> AIC(btyAge.int.lm)
[1] 729.829
```

. . . so what does this mean?

The model

$$\text{Score} = 5.156 - 0.1878 \times \text{bty\_avg} - 0.0261 \times \text{age} + 0.00532 \times \text{bty\_avg} \times \text{age} + \text{error}$$

So if you're 50 and you have a middling beauty rating of 5, your prediction is

$$5.156 - 0.1878 \times 5 - 0.0261 \times 50 + 0.00532 \times 5 \times 50 = 4.24$$

It might be easier to understand the model if we plugged in a few different ages and got the resulting regression lines.

Regression lines by age

Age 30:

$$\begin{aligned}\text{Score} &= (5.156 - 0.0261 \times 30) + (-0.1878 + 0.00532 \times 30) \times \text{bty\_avg} + \text{error} \\ &= 4.372 - 0.0283 \times \text{bty\_avg} + \text{error}\end{aligned}$$

Age 40:

$$\begin{aligned}\text{Score} &= (5.156 - 0.0261 \times 40) + (-0.1878 + 0.00532 \times 40) \times \text{bty\_avg} + \text{error} \\ &= 4.111 + 0.0249 \times \text{bty\_avg} + \text{error}\end{aligned}$$

Age 50:

$$\text{Score} = 3.850 + 0.0781 \times \text{bty\_avg} + \text{error}$$

Age 60:

$$\text{Score} = 3.588 + 0.1313 \times \text{bty\_avg} + \text{error}$$

A graph

```
plot(evals$bty_avg, evals$score, pch = ".",
main = "Blue: age 30; orange: age 40; gold: age 50; green: age 60")
abline(4.372, -0.0283, col = "blue")
abline(4.111, 0.0249, col = "orange")
abline(3.850, 0.0781, col = "gold")
abline(3.588, 0.1313, col = "green")
```

Three variables

Let's now add rank back into the model:

```
> btyAgeRank.int.lm <- lm(score ~ bty_avg + age + rank + bty_avg:age, data = evals)
> coefficients(btyAgeRank.int.lm)
(Intercept)          bty_avg              age
5.21428571      -0.15168394      -0.02528329
ranktenure track      ranktenured      bty_avg:age
-0.18137331      -0.09206255       0.00452376
> AIC(btyAgeRank.int.lm)
[1] 728.8777 # lowest AIC
```

Again, plug in a few numbers to get a sense of what's happening.

Understanding three variables

Let's just look at teaching faculty.

```
> coefficients(btyAgeRank.int.lm)
(Intercept)          bty_avg              age
5.21428571      -0.15168394      -0.02528329
ranktenure track      ranktenured      bty_avg:age
-0.18137331      -0.09206255       0.00452376
```

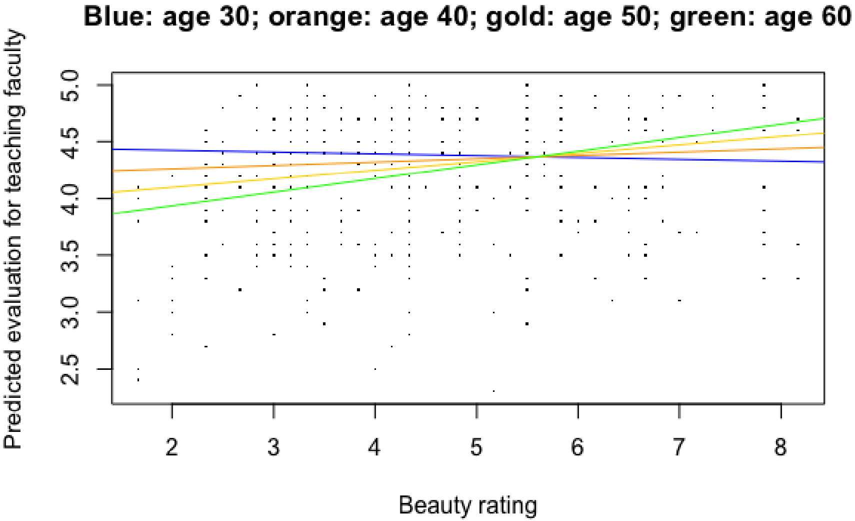
Intercepts at ages 30, 40, 50, and 60:

```
> ints <- 5.21428571 - 0.02528329 * c(30, 40, 50, 60)
> ints
[1] 4.455787 4.202954 3.950121 3.697288
```

Slopes (with respect to beauty) at ages 30, 40, 50, and 60:

```
> slopes <- -0.15168394 + 0.00452376 * c(30, 40, 50, 60)
> slopes
[1] -0.01597114  0.02926646  0.07450406  0.11974166
```

Plot the model



Things to worry about: Bad answers

Trying to minimize some numerical criterion is often a reasonable approach.

However, doing so doesn't guarantee you'll get a sensible answer.

- Check the estimates of the parameter values your model gives you are valid.
- Check the predictions your model gives you are sensible (for sensible values of the predictors.)

Variable selection in the linear regression

Things to worry about: Lots of variables/parameters

- If the number of parameters exceeds the number of observations, regression doesn't have a unique solution. (This might seem like much to worry out, but is very common in e.g. genetics problems.)
- More generally, the more variables there are, the more you have to worry about relationships due to chance. If a variable doesn't have any conceivable causal relationship to the response (e.g. it's an ID number), don't consider it. . .
- Pairs plots (scatterplots of pairs of numeric variables), e.g. using `pairs()` can give you a quick sense of what the relationships between the variables are. If two variables are measuring almost exactly the same thing, it might not be wise to include both of them in the model.

Things to worry about: Finding the optimum

- So far we've only considered up to three x-variables. If we wished, we could work through every reasonable combination of these three variables and ranked them by (for example) AIC.
- With more variables, this might not be practical. e.g. If there are 20 possible explanatory variables, there are 2^{20} possible models to consider even before worrying about interactions.
- A *stepwise* selection process (e.g. using the `step()` function in R) may help to search the space of models efficiently. Such methods do not guarantee that you'll converge to the model with the lowest AIC out of all possible models. This may or may not bother you.
  - In more complicated models, there may be multiple local maxima. You might have to try different starting values and see if you always come to the same answer.

Forward stepwise selection

1. Fit a regression with one x-variable, trying each of the predictors in turn. Keep the one that gives the lowest AIC (call this x_1.)
2. Now try a regression with two x-variables: x_1 and whichever of the remaining predictors gives the lowest AIC in conjunction with x_1.
3. Keep adding variables unless you can't improve AIC any more.

This doesn't necessarily converge to the absolute best model in terms of minimizing AIC (and doesn't consider interactions), but usually gives a pretty good one.

Example: Teaching evaluations

```
evals <- read.csv("https://www.openintro.org/book/statdata/evals.csv")

max.model <- lm(score ~ . , data = evals) # full model with all variables

step(lm(score ~ 1, data = evals), scope = formula(max.model),
      direction = "forward")
```

Backward stepwise selection

Or we could start with a “full” model with all predictors we wish to consider, then try to improve AIC by *dropping* variables one at a time.

This sometimes but not always results in a different final model than forward selection. You can always choose between them using AIC.

```
step(max.model, direction = "backward")
```

Stepwise selection: Both ways

Or at each step, we could either add or drop a variable, depending on what improves AIC the most. It might matter whether you start from the empty model or the full model.

```
step(lm(score ~ 1, data = evals), scope = formula(max.model),
      direction = "both")

step(max.model, direction = "both")
```

- If you really want, after selecting a model by a stepwise method, you can then consider interactions as well.

The usual warnings

- If a stepwise method converges to a p-variable model, it's not necessary the p-variable model with the lowest AIC. For low p it might be better to do an exhaustive search.
- AIC tends to overfit a bit. If simplicity is a goal, consider e.g. BIC.
- Regression, like correlation, is not causation.