Linear regression 2

What we've seen so far (1/)

In linear regression, we have an interval dependent variable (*Y*) which we suspect may be modulated by independent variables *X*. We wish to

- predict the value *Y* will take on given a particular *X*, and/or
- infer whether changes in *X* are really associated with *Y*.

In other words, we are trying to infer a *correlation* between *X* and *Y*: much more complex and assumption-laden methods are required to infer causation (and the direction of the causal arrow).

What we've seen so far (2/)

X, the independent variable can be:

- Binomial (experimental condition vs. control condition, intervention vs. no intervention, etc.)
  - this is the same as computing the *point-biserial correlation*, and
  - this in turn is the same as doing Student's ("equal variance") two-sample t-test.
- Interval (height, age, hours of instruction, etc.)
  - this is the same as computing *Pearson's r*.

Outline

Today we'll see a few extensions:

- Multiple independent variables in a single analysis
  - which demand *centering* and *standardization*, and
  - which require tests and corrections for *(multi)collinearity.*
- Categorical variables with more than two levels
  - which require novel *coding schemes*, and
  - which require *post hoc* tests.

Quick review

The equation for the *i*th point

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

where

- y_i is a *i*th value of the DV,
- β_0 is the *y*-intercept of the line,
- β_1 is the slope of the line,
- x_i is the *i*th value of the IV, and
- ε_i is the deviance/error of the *i*th observation.

The equation in general

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

where

- Y is an n-length vector of DV values,
- β_0 is a scalar, the y-intercept of the line,
- β_1 is a scalar, the slope of the line,
- X is an n-length vector of IV values, and
- ε is an n-length vector of deviances.

Centering and standardizing

Interpretation is hard

Given:

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

What is a "typical" value for Y? E.g.:

- what is $\bar{Y}$? or equivalently,
- what is $\hat{y}$ when $X = \bar{X}$?

A method called *centering* allows us to quickly answer both these questions as a side-effect of fitting the linear model.

Centering (1/)

What if we rewrote $Y = \beta_0 + \beta_1 X + \varepsilon$ as

$$Y = \beta_0 + \beta_1 X' + \varepsilon$$

where

$$X' = X - \bar{X}?$$

Then $\beta_0 = \bar{Y}$. (That is, the intercept gives the sample mean of Y.)

Centering (2/)

This is actually easy to do in R:

```
> x.prime <- scale(x, center = TRUE, scale = FALSE)
```

Or (and I prefer this), just:

```
> x.prime <- x - mean(x)
```

```
> n <- 25
> y <- runif(n, 0, 100)
> y.bar <- mean(y)
> y.bar
[1] 55.75366
> x <- rnorm(n, 5, 3)
> x.bar <- mean(x)
> x.bar
[1] 5.326931
```

```
> lm(y ~ x)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     67.883       -2.277
> y.hat <- 67.883 + -2.277 * x.bar
[1] 55.75358
> all.equal(y.hat, y.bar)
[1] "Mean relative difference: 1.462302e-06"
```

```
> x.prime <- x - x.bar
> lm(y ~ x.prime)

Call:
lm(formula = y ~ x.prime)

Coefficients:
(Intercept)      x.prime
     55.754       -2.277
```

NB:

- $\beta_0 = \bar{Y}$, and
- β_1 is unchanged.

Interpretation is hard (2/)

β_1 in $Y = \beta_0 + \beta_1 X$ is a measure of units *Y changed* per unit of *X*.

Therefore βs from different scales will have different magnitudes (e.g., β_1 when *X* is kilometers vs. β_1 is in miles) and can't easily be compared.

Consider the following example from the `cars` data set (circa 1920s).

NB: *X* is speed (in miles/hour); *Y* is stopping distance (in feet).

```
> speed.prime <- with(cars, speed - mean(speed))
> range(speed.prime)
[1] -11.4   9.6
> lm(dist ~ speed.prime, data = cars)

Call:
lm(formula = dist ~ speed.prime, data = cars)

Coefficients:
(Intercept)  speed.prime
     42.980        3.932
```

```
> speed.km <- 1.61 * cars$speed
> speed.km.prime <- speed.km - mean(speed.km)
> range(speed.km.prime)
[1] -18.354  15.456
> lm(dist ~ speed.prime.km, data = cars)

Call:
lm(formula = dist ~ speed.prime.km, data = cars)

Coefficients:
(Intercept)  speed.prime.km
     42.980           2.442
```

Standardization

What if we rewrote $Y = \beta_0 + \beta_1 X + \varepsilon$ as

$$Y = \beta_0 + \beta_1 X' + \varepsilon$$

where

$$X' = (X - \bar{X}) / s_X ?$$

Then β_1 is the change in Y associated with a one-σ increase in X.

Standardization

This is also actually easy to do in R:

```
> x.prime <- scale(x)
```

Or:

```
> x.prime <- (x - mean(x)) / sd(x)
```

```
> speed.prime <- scale(cars$speed)
> summary(speed.prime)
       V1
 Min.   :-2.15597
 1st Qu.:-0.64301
 Median :-0.07565
 Mean   : 0.00000
 3rd Qu.: 0.68083
 Max.   : 1.81555
```

```
> lm(dist ~ speed.prime, data = cars)

Call:
lm(formula = dist ~ speed.prime, data = cars)

Coefficients:
(Intercept)  speed.prime
      42.98        20.79
```

```
> speed.km.prime <- scale(speed.km)
> summary(speed.km.prime)
       V1
 Min.   :-2.15597
 1st Qu.:-0.64301
 Median :-0.07565
 Mean   : 0.00000
 3rd Qu.: 0.68083
 Max.   : 1.81555
```

```
> lm(dist ~ speed.prime.km, data = cars)

Call:
lm(formula = dist ~ speed.prime.km, data = cars)

Coefficients:
(Intercept)  speed.prime.km
     42.980          20.79
```

How many standard deviations?

While we normally divide by *s*, Gelman & Hill (2007:57) advise instead to divide by 2*s*, because this makes it easier to compare interval and binomial predictors.

```
> x.prime <- (x - mean(x)) / (2 * sd(x))
```

When a binomial predictor has $p = .5$, it has the roughly the range and sample standard deviation of an interval predictor standardized by 2*s*.

This is purely interpretative: it doesn't change the underlying model or inferences.

Local summary

- *Centering* (subtracting the mean from an independent variable) makes it easier to interpret the intercept (e.g., as $\bar{Y}$, and as the predicted value of y when $x = \bar{X}$).
  - It does not change the underlying inference, however, just the interpretation.
- *Standardization* (centering followed by dividing by the sample standard deviation) makes it easier to interpret coefficients (other than the intercept) as the magnitude of change in Y for each s_X.
  - This preserves the interpretation across any linear transformation of X.
  - But it can impact the inference when applying non-linear transformations to X (e.g., Hz vs. mel).
  - This can change the inference when there are multiple independent variables, because they may (and probably do) have different standard deviations.

Looking forward...

Naturally, this ability to compare variables on different scales will be useful when we perform *multiple regression* (i.e., regression with several independent variables).

Categorical variable coding

Simple dummy coding

When we have a *binomial* categorical variable (i.e. one that can take on at most two values), we use dummy coding, e.g.:

"dialect A": 0
"dialect B": 1

In fact, it doesn't really matter that much how we code binomial categorical variables, so long as the codes are two different real values.

But what about categorical variables in general?

This won't work correctly when we have three or more categories, e.g.:

Typical development (TD): 0
Specific language impairment (SLI): 1
Autism spectrum disorder (ASD): ??

If we say "2", then we are essentially treating a *categorical* variable like an *interval* variable. Not good!

Treatment coding

One straightforward extension of the binary dummy coding strategy is known as *treatment coding*. For a categorical variable with *n* levels it codes each observation using *n* − 1 binary values.

The "first" level of a categorical variable is coded as a sequence of *n* − 1 '0's, and then applies *one-hot* encoding for the remaining levels, with a '1' in the (*n* − 1)th slot.

FYI: R automatically picks one of the levels as the "first" one; you can change the order of the levels, but it's annoying.

To see how R constructs this, you can use `contr.treatment`.

```
> contr.treatment(2)
  2
1 0
2 1
> contr.treatment(3)
  2 3
1 0 0
2 1 0
3 0 1
```

```
> contr.treatment(4)
  2 3 4
1 0 0 0
2 1 0 0
3 0 1 0
4 0 0 1
> contr.treatment(5)
  2 3 4 5
1 0 0 0 0
2 1 0 0 0
3 0 1 0 0
4 0 0 1 0
5 0 0 0 1
```

Example

Thus, for our earlier example, we might use:

Typical development (TD):	[0, 0]
Specific language impairment (SLI):	[1, 0]
Autism spectrum disorder (ASD):	[0, 1]

Interpretation

Thus for a 3-way categorical variable:

$$Y = \beta_0 + \beta_1 X' + \beta_2 X'' + \varepsilon$$

then

- β_0 gives $\bar{Y}_{TD}$, the sample mean of *Y* for the TD group,
- $\beta_0 + \beta_1$ gives $\bar{Y}_{SLI}$, the sample mean of *Y* for the SLI group, and
- $\beta_0 + \beta_2$gives $\bar{Y}_{ASD}$, the sample mean of *Y* for the ASD group.

Example

Let:

- $\beta_0 = 0.191$,
- $\beta_1 = -0.208$, and
- $\beta_2 = 0.217$.

Then:

- $\bar{Y}_{TD} = 0.191$
- $\bar{Y}_{SLI} = 0.191 + -0.208 = -0.017$
- $\bar{Y}_{ASD} = 0.191 + 0.217 = 0.408$

Pros and cons

Pros:

- It's easy to compute the category mean.

Cons:

- It's not easy to compute the overall mean.
- We have to arbitrarily choose one level as a baseline.

Sum coding

Sum coding is an alternative which compares each level to the global mean.

The first $n - 1$ levels are all one-hot encoded; the nth level is coded as a sequence of −1s.

To see how R constructs this, you can use `contr.sum`.

```
> contr.sum(2)
  [,1]
1    1
2   -1
> contr.sum(3)
  [,1] [,2]
1    1    0
2    0    1
3   -1   -1
```

```
> contr.sum(4)
  [,1] [,2] [,3]
1    1    0    0
2    0    1    0
3    0    0    1
4   -1   -1   -1
> contr.sum(5)
  [,1] [,2] [,3] [,4]
1    1    0    0    0
2    0    1    0    0
3    0    0    1    0
4    0    0    0    1
5   -1   -1   -1   -1
```

Example

Thus, for our earlier example, we might use:

Typical development (TD): [+1, +0]
Specific language impairment (SLI): [+0, +1]
Autism spectrum disorder (ASD): [−1, −1]

Interpretation

Thus for a 3-way categorical variable:

$$Y = \beta_0 + \beta_1 X' + \beta_2 X'' + \varepsilon$$

then

- β_0 gives $\bar{Y}$, the overall sample mean,
- $\beta_0 + \beta_1$ gives $\bar{Y}_{TD}$, the sample mean of Y for the TD group,
- $\beta_0 + \beta_2$ gives $\bar{Y}_{SLI}$, the sample mean of Y for the SLI group, and
- $\beta_0 - \beta_1 - \beta_2$ gives $\bar{Y}_{ASD}$, the sample mean of Y for the ASD group.

Pros and cons

Pros:

- β_0 is once again the overall mean,
- and it's easy to compute the category mean.

Cons:

- The subtractions make getting category means a bit more challenging.

Local summary

- *Treatment coding* compares the first level of a categorical variable to each of the remaining $n - 1$ levels.
- *Sum coding* compares each of the n levels of a categorical variable to the global mean.

Of course, there are many other categorical coding systems in R, including

- *contrast coding*, a variant of sum coding, and
- *Helmert coding*, which makes it easy to compute the category means, and
- *(orthogonal) polynomial coding*, which is sometimes used for ordinal IVs.

How to apply in R (1/)

```
x <- factor(c("TD", "TD", "TD", "ASD", "ASD", "ASD", "ASD",
              "SLI", "SLI"))
> contrasts(x)
  SLI TD
ASD   0  0
SLI   1  0
TD    0  1
```

How to apply in R (2/)

```
> contrasts(x) <- contr.sum
> contrasts(x)
    [,1] [,2]
ASD    1    0
SLI    0    1
TD    -1   -1
```

In other words, treatment coding is the default, but a categorical variable (a "factor" in R lingo) stores the contrast matrix: it's "stateful".

How to report (sum coding example)

	Coef.	S.E.	χ^2	$p(\chi^2)$
(Intercept)	−5.462	0.11		
Group:			2.27	<.001
TD	0.191	0.13		
ASD	−0.208	0.19		
SLI	0.017	0.12		

```
> contrasts(iris$Species) <- contr.sum
> r <- lm(Petal.Length ~ Species, data = iris)
> summary(r)
Call:
lm(formula = Petal.Length ~ Species, data = iris)

...

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.75800    0.03514  106.95   <2e-16 ***
Species1    -2.29600    0.04969  -46.21   <2e-16 ***
Species2     0.50200    0.04969   10.10   <2e-16 ***
```

Post-hoc tests for categorical variables

Category vs. group significance

With a binomial categorical IV, we test the null hypothesis that the two samples are drawn from a single population, i.e., that they have the same population means.

We may also want to formulate additional null hypotheses such as "no two pairs of group means are different". E.g., spelling this out a bit:

- There are no differences between the TD and SLI groups.
- There are no differences between the TD and ASD groups.
- There are no differences between the SLI and ASD groups.

Significance at the category level does not imply significance in comparisons between individual pairs of groups!

Motivation (1/)

- Our null hypothesis is that all group means are drawn from a single population so they should be normally distributed according to the central limit theorem.
- We could just repeatedly do binary tests, but if a category has *j* levels there are

$$j\,(j - 1) / 2$$

such combinations. For a category with 10 levels, that's 45 tests.

Motivation (2/)

- And what happens when we run many, many similar tests? The chance of type I error is

$$1 - (1 - \alpha)^j$$

where *j* is the number of tests. E.g., with α = .05 and a category with 10 levels, α_{fw} = .901.

Correction for multiple comparison

Let X be a Bernoulli random variable such that $P(X = \text{true}) = .5$.

Then, for each of n "trials":

> sample X repeatedly and test $H_0 : P(X = \text{true}) = .5$ at α.

How many times out of n will we (falsely) reject H_0?

Answer: $\alpha \times n$.

Our *familywise error rate* (our chance of getting any Type I error) is much larger than our nominal α.

Bonferroni correction: implementation

One simple solution is the Bonferroni correction:

> Reject H_0 when $P < \alpha / n$.

The Bonferroni procedure is maximally *conservative*, i.e., it has low type I error (error of rejecting a true null hypothesis).

There are more sophisticated, less conservative procedures (I quite like the Benjamini-Hochberg one) and many of them are implemented by `p.adjust` in R.

One of them, specific to *multiple pairwise comparisons*, is the Tukey procedure.

Tukey's Honestly Significant Difference (HSD) method

(Also sometimes known as Tukey's Range test.)

Given two category means $\bar{Y}_1$ *and* $\bar{Y}_2$ such that $\bar{Y}_1 < \bar{Y}_2$, and n samples:

$$q = (\bar{Y}_2 - \bar{Y}_1) / (s_{\text{pooled}} + \sqrt{[2 / n]})$$

then q has a *Studentized* t-distribution parameterized by d.f. and number of comparisons: essentially a t-distribution corrected for multiple comparisons.

In R, the cumulative distribution function is `ptukey` and the quantile function is `qtukey`.

```
> r <- lm(Petal.Length ~ Species, data = iris)
> summary(r)

Call:
lm(formula = Petal.Length ~ Species, data = iris)

...

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        1.46200    0.06086   24.02   <2e-16 ***
Speciesversicolor  2.79800    0.08607   32.51   <2e-16 ***
Speciesvirginica   4.09000    0.08607   47.52   <2e-16 ***
```

One commonly-used way to test for group effects is to use the *log-likelihood ratio test*.

```
> drop1(r, test = "Chisq")
Single term deletions

Model:
Petal.Length ~ Species
        Df Sum of Sq    RSS     AIC  Pr(>Chi)
<none>                27.22 -249.99
Species  2     437.1 464.33  171.49  <2.2e-16 ***
```

"There is a significant effect of species at α = .05 according to the log-likelihood ratio test ($\chi(2) = 437.1$, $p < .001$)."

```
> TukeyHSD(aov(r))
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = r)

$Species
                       diff     lwr     upr p adj
versicolor-setosa    2.798 2.59422 3.00178     0
virginica-setosa     4.090 3.88622 4.29378     0
virginica-versicolor 1.292 1.08822 1.49578     0
```

"All pairwise comparisons are significant at α_{fw} = .05 (setosa < versicolor < virginica) according to the Tukey HSD test."

Alternatives

The R library `multcomp` gives you more control over the definition of the HSD tests; for instance, you can choose not to run post-hoc tests on all categorical independent variables.

The printout is also a bit cleaner.

Assumptions

The Tukey HSD test assumes homoscedasticity across levels of a group.

Levene's test (`levene.test` in R) tests the null hypothesis of homoscedasticity across two or more levels.

Questions? Please take them to email, or Slack.