BIOS7345 Lab 2

Least-squares regression

Sarah Lotspeich

September 14, 2018

Prologue: Pipes

Before we begin, there are two basic pipes functions I will be using.

- Single pipe: `%>%`
- Double pipe: `%<>%`

The `%>%` operator "pipes" an argument into a function. For example, the following two lines of code are equivalent.

```
# without pipes
mean(seq(1, 10))
```

```
## [1] 5.5
```

```
# with pipes
seq(1, 10) %>% mean()
```

```
## [1] 5.5
```

The single pipe is super useful when you have lots of nested arguments, and it saves you a lot of confusing parentheses.

The `%<>%` operator "pipes" the object on the left-hand side into a function, and then reassigns the new value to the original name. Some simple code:

```
# create a vector
x <- seq(1, 10)

# say I want to squareroot transform the whole vector without pipes
x <- sqrt(x)
x
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751
##  [8] 2.828427 3.000000 3.162278
```

```
# create a vector
x <- seq(1, 10)
# with pipes
x %<>% sqrt()
x
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751
##  [8] 2.828427 3.000000 3.162278
```

Data: Entertainer Career Arcs

These data come from a super cool social media site for data people called data.world. The link to the documentation can be found here, but the TLDR is:

1. we have a sample of 63 notable entertainers (actors, actresses, musicians, etc.) from the last hundred years, and
2. for each we know gender (`gender_traditional`), the age when they first "broke through" (`breakthrough_age`), and the age when they won their first award (`first_award_age`).

Spoiler alert: you won't know all of the names. Darn millenials.

```
stars <- read.csv(file = "https://query.data.world/s/xsweyho62srvyorzoly3w2uctn76rz",
    header = TRUE, stringsAsFactors = FALSE)
```

Objective

We are interested in the following research questions:

1. What is the association between the age at which an entertainer broke through and the age at which they won their first award?
2. Do men and women have different expected ages at which they would win their first award?
3. Is the association between ages of breakthrough and first award different for men and women?

We can answer them using three separate, pre-specified regression models.

Models

Simple linear regression with one continuous predictor

The first model we will fit is for `first_award_age` on `breakthrough_age`. Specify the model, using proper LaTex, below:

$$E(\text{first award age}|\text{breakthrough age}) = \beta_0 + \beta_1(\text{breakthrough age}).$$

Now, begin fitting the model by restructuring these data into your response vector (`y`) and design matrix (`X`). I recommend using `data.matrix()` for this.

```
# create response vector

# create design matrix using data.matrix()
```

Begin by estimating the model coefficients, $\hat{\boldsymbol{\beta}} = \begin{bmatrix}\hat{\beta}_0 & \hat{\beta}_1\end{bmatrix}^T$ according to the Rencher Theorem 7.3a:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

```
# use matrix operations to calculate least-squares estimates beta0, beta1

# for later, go ahead and round these to 3 digits
```

Now we need an unbiased estimate for the variance of the error terms (σ^2), which we obtain using Rencher Equation (7.23):

$$s^2 = \frac{1}{n-k-1}(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}),$$

where $k = $ the number of predictors in your model and $n = $ sample size.

```
# sample size

# number of predictors (excl. intercept)

# use (7.23) to estimate the variance of the error terms
```

The last thing this model needs are measures of uncertainty (e.g. standard errors) for the coefficient estimates. Using our estimate s^2 above, we can calculate these directly according to Rencher (7.27):

$$\hat{Cov}(\hat{\boldsymbol{\beta}}) = s^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}.$$

```r
# estimate the SEs for the estimated coefficients

# round these off
```

To wrap up this model let's make a nice table and print it using `xtable()` (which we totally remember from last week's lab). Recall: what do we need to add in the chunk header to make this print nicely?

```r
# create a table for the model
```

Try using the `dplyr::mutate()` function to append columns for the Wald Z test statistic and p-value for significance tests for $H_0 : \beta_0 = 0$ and $H_0 : \beta_1 = 1$. Consider using `ifelse()` to clean up the p-values if they're really small.

```r
# append Z test statistics and p-values

# print model table
```

Do future you a favor and put this into a function called `my_ls()` with parameters for the response vectors, `y`, the design matrix, `X`, and a caption for the model table to be printed.

```r
# write a function to take in y, X and print the model table
my_ls <- function(y, X, cap) {

}
```

For comparison, you may now fit this model using `rms::ols()`.

```r
# fit the same model using ols()
```

Simple linear regression with one categorical predictor

The next model we are going to fit is for `first_award_age` on `traditional_gender`. Once again, Specify the model, using proper LaTex, below:

$$E(\text{first award age}|\text{gender}) = \beta_0 + \beta_1(\text{gender}).$$

Our response vector (`y`) is the same as above, but we code categorical design matrices differently. Treating males as referrent, create the design matrix for this model now.

```r
# create design matrix using data.matrix()
```

Use `my_ls()` to calculate the same three values for this model, and create a table of the same format.

```r
# use my_ls() to fit this model
```

Check yourself. Since we want males to be our reference group, we need to recode `gender_traditional` as a factor to make sure that `ols()` will know.

```r
# make gender_traditional a factor

# fit the same model using ols()
```

Multiple linear regression with interacting continuous and categorical predictors

Our last model seeks to incorporate separate effects of `breakthrough_age` for male and female entertainers, which we accomplish via an interaction between predictors `breakthrough_age` and `traditional_gender`. Thus, we specify the following model:

$E(\text{first award age}|\text{breakthrough age, gender}) = \beta_0+\beta_1(\text{breakthrough age})+\beta_2(\text{gender})+\beta_3(\text{breakthrough age})\times(\text{gender}).$

Once more, we can recycle `y` from the first model, but our design matrix `X` now needs:

1. an intercept,
2. the continuous `breakthrough_age` values,
3. the indicator for female (as in model 2), and
4. the interaction between `breakthrough_age` and the female indicator.

```
# create design matrix using data.matrix() for model: first_award_age ~
# breakthrough_age*gender_traditional
```

Use `my_ls()` again to calculate the same three values for this model, and create a table of the same format.

```
# fit this model using my_ls()
```

And check yourself against `rms::ols()`.

```
# fit the same model using ols()
```

Predicting based on models

Manually calculating predictions

Begin by plotting `breakthrough_age` against `first_award_age`.

```
# make a scatterplot of breakthrough_age vs. first_award_age
```

Manually calculate the predicted values for `first_award_age` based on your `beta_hat1` estimates for `breakthrough_age` from 10 to 55 years old.

```
# create design matrix to predict age at first award for breakthrough ages
# 10-55

# use matrix operations to predict y
```

Add a line to your `mod1_plot` with the predicted values from the model.

```
# create a dataframe with columns for x and predicted y


# add geom_line() layer to mod1_plot with the model predictions for
# first_award_age
```

Lastly, let's add the model equation to the plot (because we're fancy). Before we put it into the plot, let's make sure we're comfortable with `paste()`.

```
# test your paste() statement to make sure you like how it looks

# now add an annotate() layer to mod1_plot with the model equation

# and print your plot
```

Invariance of predicted values to linear transformation

Often, we like to transform predictors so that the coefficients can be interpreted more meaningfully. Say we're more interested in 5 year changes in `breakthrough_age` than 1 year changes. Let's transform `breakthrough_age`, refit model 1 (recall: `first_award_age ~ breakthrough_age`), and compare predicted values between the original and linearly transformed models.

```
# add a column for breakthrough_age_centered by subtracting the median

# create the design matrix again

# use matrix operations to calculate least-squares estimates beta0, beta1

# create design matrix to predict age at first award for breakthrough ages
# 10-55

# and predict first_award_age for ages 10-55 again using matrix operations

# create a dataframe with columns for x and predicted y
```

Create the same plot as before with `breakthrough_age_5yrs` against `first_award_age`, overlaid with your transformed model predictions. Use `ggpubr:ggarrange()` to print this plot side-by-side with the original.

```
# test your paste() statement to make sure you like how it looks

# create the same plot using the transformed model

# print this plot next to the original
```

This is follows from **Rencher Theorem 7.3e.**

Full rank transformation on predictor

Take any full-rank 2×2 matrix $\boldsymbol{K}_1$ (e.g. sampling the elements independently from a Normal distribution).

```
# generate full-rank 2x2 transformation matrix K
```

Just for fun, let's make sure that this matrix is full-rank.

You: "But Sarah, didn't we just design it to be?"

Sarah: "Yes, but wouldn't you like to know how to calculate the rank of a matrix in R anyway?"

```
# check Rank(K)
```

There are some other helpful matrix operations in `R` that I would recommend for checking homework, spending a Friday night at home, etc. that can be found here.

Now, consider a model adjusting for `breakthrough_age` and `tranditional_gender` without the interaction. Construct this design matrix,

```
# create the design matrix for model: first_award_age ~ breakthrough_age +
# traditional_gender
```

and fit this model.

```r
# fit the model for y ~ X5
```

Next, let's do a full-rank linear transformation of our design matrix `X5`.

```r
# full-rank linear transformation of X5 be careful: should you be
# transforming the intercept column?
```

What do we know about Rank($\boldsymbol{Z}$)? First, check that `X5` is full-rank.

```r
# check rank(X5)
```

Now, since `X5` is full-rank and `Z` is a full-rank, square matrix, we have that multiplying `X5` by `Z` does not change the rank (by Rencher Theorem 2.4.). Check that the rank of `Z` equals the rank of `X5`.

```r
# check that Rank(Z) = Rank(X5)
```

Fit the model for `first_award_age` on the transformed design matrix `Z` now.

```r
# fit the model for y ~ Z
```

Last, we compare the fitted values for the 63 entertainers based on the model before and after full-rank linear transformation.

```r
# check that predicted values for the models are the same
```

This is Rencher Theorem 7.3e Corollary 1.