Multivariate Linear Regression

Max Turgeon

STAT 4690–Applied Multivariate Analysis

Multivariate Linear Regression model

- We are interested in the relationship between p outcomes $Y_1, \ldots, Y_p$ and q covariates $X_1, \ldots, X_q$.
  - We will write $\mathbf{Y} = (Y_1, \ldots, Y_p)$ and $\mathbf{X} = (1, X_1, \ldots, X_q)$.
- We will assume a **linear relationship**:
  - $E(\mathbf{Y} \mid \mathbf{X}) = B^T\mathbf{X}$, where B is a $(q+1) \times p$ matrix of *regression coefficients*.
- We will also assume **homoscedasticity**:
  - $\mathrm{Cov}(\mathbf{Y} \mid \mathbf{X}) = \Sigma$, where Σ is positive-definite.
  - In other words, the (conditional) covariance of $\mathbf{Y}$ does not depend on $\mathbf{X}$.

Relationship with Univariate regression i

- Let σ_i^2 be the i-th diagonal element of Σ.
- Let β_i be the i-th column of B.
- From the model above, we get p univariate regressions:
  - $E(Y_i \mid \mathbf{X}) = \mathbf{X}\beta_i$;
  - $\mathrm{Var}(Y_i \mid \mathbf{X}) = \sigma_i^2$.
- However, we will use the correlation between outcomes for hypothesis testing
- This follows from the assumption that each component Y_i is linearly associated with the *same* covariates $\mathbf{X}$.

Relationship with Univariate regression ii

- If we assumed a different set of covariates $\mathbf{X}_i$ for each outcome Y_i and still wanted to use the correlation between the outcomes, we would get the **Seemingly Unrelated Regressions** (SUR) model.
  - This model is sometimes used by econometricians.

Least-Squares Estimation i

- Let $\mathbf{Y}_1 \dots, \mathbf{Y}_n$ be a random sample of size n, and let $\mathbf{X}_1, \dots, \mathbf{X}_n$ be the corresponding sample of covariates.
- We will write $\mathbb{Y}$ and $\mathbb{X}$ for the matrices whose i-th row is $\mathbf{Y}_i$ and $\mathbf{X}_i$, respectively.
    - We can then write $E(\mathbb{Y} \mid \mathbb{X}) = \mathbb{X}B$.
- For Least-Squares Estimation, we will be looking for the estimator $\hat{B}$ of B that minimises a least-squares criterion:
    - $LS(B) = \mathrm{tr}\left[(\mathbb{Y} - \mathbb{X}B)^T(\mathbb{Y} - \mathbb{X}B)\right]$
    - **Note**: This criterion is also known as the (squared) *Frobenius norm*; i.e. $LS(B) = \|\mathbb{Y} - \mathbb{X}B\|_F^2$.

Least-Squares Estimation ii

- **Note 2**: If you expand the matrix product and look at the diagonal, you can see that the Frobenius norm is equivalent to the sum of the squared entries.

- To minimise $LS(B)$, we could use matrix derivatives...
- Or, we can expand the matrix product along the diagonal and compute the trace.
- Let $\mathbf{Y}_{(j)}$ be the j-th column of $\mathbb{Y}$.

Least-Squares Estimation iii

- In other words, $\mathbf{Y}_{(j)} = (Y_{1j}, \ldots, Y_{nj})$ contains the n values for the outcome Y_j. We then have

$$\begin{aligned} LS(B) &= \mathrm{tr}\left[(\mathbb{Y} - \mathbb{X}B)^T(\mathbb{Y} - \mathbb{X}B)\right] \\ &= \sum_{j=1}^p (\mathbf{Y}_{(j)} - \mathbb{X}\beta_j)^T(\mathbf{Y}_{(j)} - \mathbb{X}\beta_j) \\ &= \sum_{j=1}^p \sum_{i=1}^n (Y_{ij} - \beta_j^T\mathbf{X}_i)^2. \end{aligned}$$

Least-Squares Estimation iv

- For each j, the sum $\sum_{i=1}^{n}(Y_{ij} - \beta_j^T \mathbf{X}_i)^2$ is simply the least-squares criterion for the corresponding univariate linear regression.
- $\hat{\beta}_j = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbf{Y}_{(j)}$
- But since $LS(B)$ is a sum of p positive terms, each minimised at $\hat{\beta}_j$, the whole is sum is minimised at

$$\hat{B} = \begin{pmatrix} \hat{\beta}_1 & \cdots & \hat{\beta}_p \end{pmatrix}.$$

- Or put another way:

$$\hat{B} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}.$$

Comments i

- We still have not made any distributional assumptions on $\mathbf{Y}$.
  - We do not need to assume normality to derive the least-squares estimator.
- The least-squares estimator is *unbiased*:

$$\begin{aligned} E(\hat{B} \mid \mathbb{X}) &= (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}E(\mathbb{Y} \mid \mathbb{X}) \\ &= (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{X}B \\ &= B. \end{aligned}$$

Comments ii

- We did not use the covariance matrix Σ anywhere in the estimation process. But note that:

$$\begin{aligned}
\mathrm{Cov}(\hat{\beta}_i, \hat{\beta}_j) &= \mathrm{Cov}\left((\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbf{Y}_{(i)}, (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbf{Y}_{(j)}\right) \\
&= (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathrm{Cov}\left(\mathbf{Y}_{(i)}, \mathbf{Y}_{(j)}\right)\left((\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\right)^T \\
&= (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\left(\sigma_{ij}I_n\right)\mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1} \\
&= \sigma_{ij}(\mathbb{X}^T\mathbb{X})^{-1},
\end{aligned}$$

where σ_{ij} is the (i, j)-th entry of Σ.

Example i

```r
# Let's revisit the plastic film data
library(heplots)
library(tidyverse)

Y <- Plastic %>%
  select(tear, gloss, opacity) %>%
  as.matrix

X <- model.matrix(~ rate, data = Plastic)
head(X)
```

Example ii

```
##   (Intercept) rateHigh
## 1           1        0
## 2           1        0
## 3           1        0
## 4           1        0
## 5           1        0
## 6           1        0
```

```
(B_hat <- solve(crossprod(X)) %*% t(X) %*% Y)
```

Example iii

```
##               tear gloss opacity
## (Intercept) 6.49  9.57    3.79
## rateHigh    0.59 -0.51    0.29
```

```r
# Compare with lm output
fit <- lm(cbind(tear, gloss, opacity) ~ rate,
          data = Plastic)
coef(fit)
```

```
##               tear gloss opacity
## (Intercept) 6.49  9.57    3.79
## rateHigh    0.59 -0.51    0.29
```

Geometry of LS i

- Let $P = \mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T$.
- P is symmetric and *idempotent*:

$$P^2 = \mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T = \mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T = P.$$

- Let $\hat{\mathbb{Y}} = \mathbb{X}\hat{B}$ be the fitted values, and $\hat{\mathbb{E}} = \mathbb{Y} - \hat{\mathbb{Y}}$, the residuals.
    - We have $\hat{\mathbb{Y}} = P\mathbb{Y}$.
    - We also have $\hat{\mathbb{E}} = (I - P)\mathbb{Y}$.

Geometry of LS ii

- Putting all this together, we get

$$\begin{aligned}\hat{\mathbb{Y}}^T\hat{\mathbb{E}} &= (P\mathbb{Y})^T(I-P)\mathbb{Y}\\ &= \mathbb{Y}^T P(I-P)\mathbb{Y}\\ &= \mathbb{Y}^T(P-P^2)\mathbb{Y}\\ &= 0.\end{aligned}$$

- In other words, the fitted values and the residuals are **orthogonal**.
- Similarly, we can see that $\mathbb{X}^T\hat{\mathbb{E}} = 0$ and $P\mathbb{X} = \mathbb{X}$.

Geometry of LS iii

- **Interpretation**: $\hat{\mathbb{Y}}$ is the orthogonal projection of $\mathbb{Y}$ onto the column space of $\mathbb{X}$.

Example (cont'd) i

```r
Y_hat <- fitted(fit)
residuals <- residuals(fit)

crossprod(Y_hat, residuals)
```

```
##                  tear        gloss       opacity
## tear     -9.489298e-16 2.959810e-15 -4.720135e-15
## gloss    -1.424461e-15 1.109357e-15 -1.150262e-14
## opacity  -7.268852e-16 1.211209e-15  1.648459e-16
```

Example (cont'd) ii

```
crossprod(X, residuals)
```

```
##                tear        gloss      opacity
## (Intercept)       0 5.828671e-16 -4.440892e-16
## rateHigh          0 1.387779e-16  4.440892e-16
```

Example (cont'd) iii

```
# Is this really zero?
isZero <- function(mat) {
  all.equal(mat, matrix(0, ncol = ncol(mat),
                        nrow = nrow(mat)),
            check.attributes = FALSE)
}

isZero(crossprod(Y_hat, residuals))
```

```
## [1] TRUE
```

Example (cont'd) iv

```r
isZero(crossprod(X, residuals))
```

```
## [1] TRUE
```

Bootstrapped Confidence Intervals i

- We still have not made any assumption about the distribution of $\mathbf{Y}$, beyond the conditional mean and covariance function.
    - Let's see how much further we can go.
- We will use **bootstrap** to derive confidence intervals for our quantities of interest.
- Bootstrap is a resampling technique for estimating the sampling distribution of an estimator of interest.
    - Particularly useful when we think the usual assumptions may not hold, or when the sampling distribution would be difficult to derive.

Bootstrapped Confidence Intervals ii

- Let's say we want to estimate the sampling distribution of the correlation coefficient.
- We have a sample of pairs $(U_1, V_1), \ldots, (U_n, V_n)$, from which we estimated the correlation $\hat{\rho}$.
- The idea is to resample **with replacement** from our sample to mimic the process of "repeating the experiment".

Bootstrapped Confidence Intervals iii

- For each bootstrap sample $(U_1^{(b)}, V_1^{(b)}), \ldots, (U_n^{(b)}, V_n^{(b)})$, we compute the sample correlation $\hat{\rho}^{(b)}$.
- We now have a whole sample of *correlation coefficients* $\hat{\rho}^{(1)}, \ldots, \hat{\rho}^{(B)}$.
- From its quantiles, we can derive a confidence interval for $\hat{\rho}$.

Example i

```r
library(candisc)

dataset <- HSB[,c("math", "sci")]

(corr_est <- cor(dataset)[1,2])
```

```
## [1] 0.6495261
```

Example ii

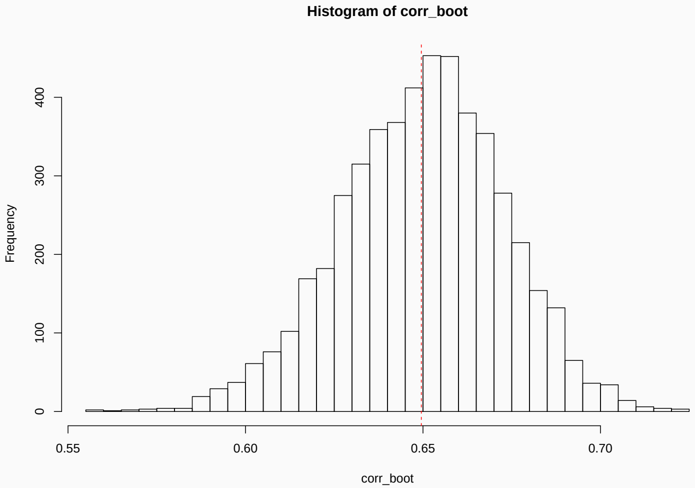
```r
# Choose a number of bootstrap samples
B <- 5000
corr_boot <- replicate(B, {
  samp_boot <- sample(nrow(dataset),
                      replace = TRUE)
  dataset_boot <- dataset[samp_boot,]
  cor(dataset_boot)[1,2]
})

quantile(corr_boot,
         probs = c(0.025, 0.975))
```

Example iii

```
##      2.5%     97.5%
## 0.6021182 0.6930400
```

```
hist(corr_boot, breaks = 50)
abline(v = corr_est, col = 'red',
       lty = 2)
```

Example iv


Histogram of corr_boot
Frequency
0
100
200
300
400
0.55
0.60
0.65
0.70
corr_boot

Bootstrapped Confidence Intervals (cont'd) i

- Going back to our multivariate linear regression setting, we can bootstrap our estimate of the matrix of regression coefficients!
- We will sample with replacement the rows of $\mathbb{Y}$ and $\mathbb{X}$
    - It's important to sample the **same** rows in both matrices. We want to keep the relationship between $\mathbf{Y}$ and $\mathbf{X}$ intact.
- For each bootstrap sample, we can compute the estimate $\hat{B}^{(b)}$.
- From these samples, we can compute confidence intervals for each entry in B.

Bootstrapped Confidence Intervals (cont'd) ii

- We can also technically compute confidence regions for multiple entries in B
  - E.g. a whole column or a whole row
  - But multivariate quantiles are tricky...

Example (cont'd) i

```r
B_boot <- replicate(B, {
  samp_boot <- sample(nrow(Y),
                      replace = TRUE)
  X_boot <- X[samp_boot,]
  Y_boot <- Y[samp_boot,]

  solve(crossprod(X_boot)) %*% t(X_boot) %*% Y_boot
})

# The output is a 3-dim array
dim(B_boot)
```

Example (cont'd) ii

```
## [1]    2    3 5000
```

```r
B_boot[,,1]
```

```
##             tear gloss opacity
## (Intercept) 6.56  9.73    3.73
## rateHigh    0.55 -0.81   -0.63
```

```r
# CI for effect of rate on tear
quantile(B_boot["rateHigh", "tear",],
         probs = c(0.025, 0.975))
```

Example (cont'd) iii

```
##       2.5%     97.5%
## 0.2637088 0.9022186
```

```r
# CI for effect of rate on gloss
quantile(B_boot["rateHigh", "gloss",],
         probs = c(0.025, 0.975))
```

```
##        2.5%      97.5%
## -0.8912202 -0.1219643
```

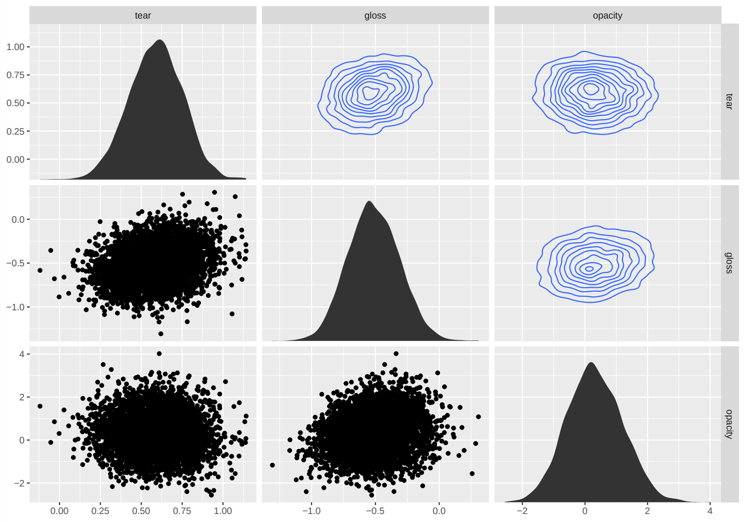
Example (cont'd) iv

```r
# CI for effect of rate on opacity
quantile(B_boot["rateHigh", "opacity",],
         probs = c(0.025, 0.975))
```

```
##      2.5%     97.5%
## -1.354167  2.071758
```

Example (cont'd) v

```
library(ggforce)

B_boot["rateHigh",,] %>%
  t() %>%
  as.data.frame() %>%
  ggplot(aes(x = .panel_x, y = .panel_y)) +
  geom_point() +
  geom_autodensity() +
  geom_density2d() +
  facet_matrix(vars(everything()),
               layer.diag = 2,
               layer.upper = 3)
```


tear
gloss
opacity
tear
gloss
opacity
1.00
0.75
0.50
0.25
0.00
0.0
−0.5
−1.0
4
2
0
−2
0.00
0.25
0.50
0.75
1.00
−1.0
−0.5
0.0
−2
0
2
4

```r
# There is some correlation, but not much
B_boot["rateHigh",,] %>%
  t() %>%
  cor()
```

```
##                tear     gloss     opacity
## tear     1.00000000 0.2386486 -0.06418527
## gloss    0.23864865 1.0000000  0.15195001
## opacity -0.06418527 0.1519500  1.00000000
```