Inference Tutorial 4

These questions cover causal inference and inference for linear models

1. The text file `confound.txt` can be found on the course web page. It can be read into R using
`conf <-  read.table("https://people.maths.bris.ac.uk/~sw15190/TOI/confound.txt")`
The resulting data frame contains variables `y`, `x`, `z`, `v` and `w`. I generated `y` from a linear model of the structure $y_i = \beta_0 + \beta_1 x_i + \beta_2 z_i +$ other terms $+ \epsilon_i$. `conf` contains `x` and `z` but not the 'other stuff' variables. However `v` and `w` are independent of the 'other stuff' variables and were not used in generating `y`. I used integer values for β_1 and β_2. What is your best estimate of what they were?

   Hint: it should not be necessary to form any model matrices explicitly. The *fitted* values extracted from an `lm` model are the least squares projections of the response data onto the columns of the model matrix...

   Solution

   Here is the wrong answer

```
> lm(y~x+z,data=dat)

Call:
lm(formula = y ~ x + z, data = dat)

Coefficients:
(Intercept)            x            z
    -0.3196       5.7767       3.7345
```

   It has made no attempt to account for the effects of confounding. Instead try treating `v` and `w` as instrumental variables. We were told that they are uncorrelated with the confounders and not causally related to the response, but we need to check that they are correlated with `x` and `z`

```
> cor(dat)
          y         x         z         v         w
y 1.0000000 0.6081628 0.5665199 0.1753082 0.1532012
x 0.6081628 1.0000000 0.7043185 0.6023741 0.8020260
z 0.5665199 0.7043185 1.0000000 0.9053478 0.6110504
v 0.1753082 0.6023741 0.9053478 1.0000000 0.7025802
w 0.1532012 0.8020260 0.6110504 0.7025802 1.0000000
```

   ...fairly strong correlation. So now project `x` and `z` onto the column space of `v` and `w`, using `lm`, in order to orthogonalize them to the confounders. Then estimate the model with the new predictors

```
> dat$x1 <- fitted(lm(x~v+w,data=dat))
> dat$z1 <- fitted(lm(z~v+w,data=dat))
> lm(y~x1+z1,data=dat)

Call:
lm(formula = y ~ x1 + z1, data = dat)

Coefficients:
(Intercept)           x1           z1
    -0.5506       1.2276       1.8380
```

   So, given that integers were used in the simulation, the best estimate is that the values used were 1 and 2.

2. A government statistician under pressure to provide an explanation for the value of the pound more politically expedient than the obvious one, is trying to build a linear model in terms of a large number of economic variables that he has available, alongside a decade's worth of exchange rate data. Being scrupulous, he is acutely aware of the problems of inferring causation from association, and equally aware that the well known genius heading his department is unlikely to grasp the distinction. Reading up on instrumental variables late one night, he can not think of any suitable ones, but

then hits upon an idea. Why not randomly generate instrumental variables to be correlated with the known explanatory variables: because they are randomly generated they will be independent of any hidden confounder variables. What is wrong with this idea?

Solution

While it is possible to generate the random data that are correlated with the observed covariates (just add noise to the observed covariates, for example), the mere fact that this involves some randomness will not ensure that they are uncorrelated with the confounders, given that the observed covariates are correlated with the confounders.

3. A statistician has fitted two alternative models to response data y_i. The first is

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \tag{1}$$

and the second is

$$y_i = \beta_0 + \beta_1 x_i + \gamma_j + \epsilon_i \quad \text{if } y_i \text{ from group } j. \tag{2}$$

In R the factor variable containing the group labels is `trt`. The statistician wants to test the null hypothesis that model (1) is correct against the alternative that model (2) is correct. To do this, both models are fitted in R, and a fragment of the summary for each is shown below.

```
> summary(b0)
lm(formula = y ~ x)
...
Residual standard error: 0.3009 on 98 degrees of freedom

> summary(b1)
lm(formula = y ~ x + trt)
...
Residual standard error: 0.3031 on 95 degrees of freedom
```

In R this test could be conducted via `anova(b0,b1)`, but instead perform the test using just the information given (with reference to section 3.4.4. of the notes).

(a) Compute the residual sum of squares for each model.

(b) Compute the F ratio statistic for the test.

(c) Evaluate the p-value for the test in R, and interpret it.

Solution

(a) The residual standard error is the square root of the estimated residual variance, which is the residual sum of squares over the residual degrees of freedom, so…

```
> rss0 <- .3009^2*98
> rss1 <- .3031^2*95
> rss0;rss1
[1] 8.872999
[1] 8.727613
```

(b) Directly from section 3.4.4. of the notes…

```
> F <- ((rss0-rss1)/3)/(rss1/95);F
[1] 0.5275101
```

(c) We need the probability of getting this large an F if the null model is true…

```
> pf(F,3,95,lower.tail=F)
[1] 0.6644552
```

…so no reason to doubt the null model.

4. This question is about using R to get a ‘hands on’ understanding of the linear model fitting theory in the notes as well as brushing up some practical matrix skills. You will need to use some basic matrix operators and functions for working with matrices and vectors:

`t(A)` returns the transpose of any matrix `A`.

`A%*%B` performs a matrix multiplication of matrices `A` and `B`.

`solve(A,b)` returns $\mathbf{A}^{-1}\mathbf{b}$.

`solve(A)` returns $\mathbf{A}^{-1}$.

`b[3:7]` returns elements 3 to 7 of vector `b` (obviously any range can be supplied).

`as.numeric(t(b)%*%b)` turns the results of a matrix-vector computation (evaluating $\mathbf{b}^T\mathbf{b}$ in this example) into (an) ordinary number(s): can be useful in part h.

Should you require an arbitrary vector of length `n`, for some reason, then `runif(n)` is one possibility.

The aim of the question is to use the theory in section 3.2 of the notes to calculate least squares estimates, and a corresponding estimator covariance matrix, for the linear model for the `cars` data, which we met in section 3.1.1 of the notes (and in a previous tutorial). Letting y_i be stopping distance for the i^{th} trial, and x_i the corresponding initial speed, then the model is:

$$\mu_i = \beta_1 + \beta_2 x_i + \beta_3 x_i^2, \quad y_i \sim N(\mu_i, \sigma^2)$$

where the y_i are independent.

It is a good idea to build up the R code answering these questions in a text file, so that you can cut and paste it into R, and end up with a complete solution to the practical.

(a) Create a model matrix $\mathbf{X}$, for the above model, using the data in the `cars` data frame and the `model.matrix` function e.g. `X <- model.matrix(~speed+I(speed^2))` will do the trick.

(b) Now form the QR decomposition of $\mathbf{X}$ as follows

```
qrx <- qr(X) ## returns a QR decomposition object
Q <- qr.Q(qrx,complete=TRUE) ## extract Q
R <- qr.R(qrx)  ## extract R
```

(c) Look at `R` to confirm its structure. Confirm that `Q` is and orthogonal matrix. Confirm that $\|\mathbf{Qx}\|^2 = \|\mathbf{x}\|^2$ for any $\mathbf{x}$ of appropriate dimension, by trying some example $\mathbf{x}$'s (why does this happen?).

(d) Obtain $\mathbf{f}$ and $\mathbf{r}$ (in the notation of section 3.2).

(e) Evaluate $\hat{\boldsymbol{\beta}}$ using $\mathbf{R}$ and $\mathbf{f}$.

(f) Confirm that $\|\mathbf{r}\|^2 = \|\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2$, for this model.

(g) Estimate σ^2 as $\hat{\sigma}^2 = \|\mathbf{r}\|^2/(n-p)$.

(h) Using $\hat{\sigma}^2$ and $\mathbf{R}$, obtain an estimate of the estimator covariance matrix $\mathbf{V}_{\hat{\boldsymbol{\beta}}}$ corresponding to $\hat{\boldsymbol{\beta}}$.

(i) The following code obtains $\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{V}}_{\hat{\boldsymbol{\beta}}}$ using `lm`.

```
lm.fit <- lm(dist~speed+I(speed^2),data=cars)
beta.lm <- coef(lm.fit)
V.beta.lm <- vcov(lm.fit)
```

Compare these results to the ones that you have obtained.

Solution

```
## 1.
X <- model.matrix(~speed+I(speed^2),data=cars)

## 2.
qrx <- qr(X)
Q <- qr.Q(qrx,complete=TRUE)
R <- qr.R(qrx)

## 3.
```

```
R ## ooh! it's upper triangular
## look at properties of Q
Q%*%t(Q) ## it's orthogonal!
n <- nrow(cars)
x <- runif(n)
x.norm <- t(x)%*%x
Qx <- Q%*%x
Qx.norm <- t(Qx)%*%Qx
x.norm;Qx.norm
## Qx.norm = t(x)%*%t(Q)%*%Q%*%x = t(x)%*%x,
## by orthogonality of Q

## 4.
Qty <- t(Q)%*%cars$dist
f <- Qty[1:3]
r <- Qty[-(1:3)]

## 5.
beta.hat <- solve(R,f) ## backsolve more efficient

## 6.
r.norm <- as.numeric(t(r)%*%r)
rss <- sum((cars$dist-X%*%beta.hat)^2)
r.norm;rss

## 7.
sigma2.hat <- r.norm/(n-3)

## 8.
R.inv <- solve(R)
V.beta <- R.inv%*%t(R.inv)*sigma2.hat

## 9.
lm.fit <- lm(dist~speed+I(speed^2),data=cars)
beta.lm <- coef(lm.fit)
V.beta.lm <- vcov(lm.fit)

beta.lm;beta.hat
V.beta.lm;V.beta
```