Multiple Linear Regression

The Bootstrap, MVN

Math 392

Resampling methods: two methods

There are two general approaches to getting bootstrap intervals for your regression estimates (for β, $\hat{E}(Y|X)$, $[Y|X]$):

1. Bootstrap the **cases**.
2. Bootstrap the **residuals**.

Aside:

Bootstrap is not a good idea when your you have few observations, let's simulate a data set of moderate size.

```
set.seed(8134)
n <- 35
x0 <- 1
x1 <- rnorm(n)
x2 <- rnorm(n)
X <- as.matrix(data.frame(x0, x1, x2))
B <- c(-3, .5, 2)
sig_sq <- 2.25
epsilon <- rnorm(n, mean = 0, sd = sqrt(sig_sq))
Y <- X %*% B + epsilon
df <- data.frame(X, Y)
```

Bootstrap the cases

```
b1 <- sample_frac(df, replace = TRUE)
head(b1)
```

```
##   x0          x1          x2         Y
## 1  1  0.83535656  0.1209323 -2.536903
## 2  1  0.07239619 -1.8961143 -7.910420
## 3  1 -0.25058242 -1.3210614 -3.608330
## 4  1 -0.74282899 -1.1589374 -3.755188
## 5  1 -0.96462888 -0.1406838 -3.856865
## 6  1  0.57466246 -1.1905117 -4.296790
```

```
b2 <- sample_frac(df, replace = TRUE)
head(b2)
```

```
##   x0         x1          x2         Y
## 1  1  0.5631423 -0.82051067 -3.4623495
## 2  1  2.0582999 -1.24876981 -4.4920737
## 3  1 -0.2023860  0.06540560 -0.8649449
## 4  1  2.0582999 -1.24876981 -4.4920737
## 5  1  0.8353566  0.12093227 -2.5369027
## 6  1  0.6973097  0.04931632 -2.6266855
```

Compute estimates for each b

```
mb1 <- lm(Y ~ x1 + x2, data = b1)
coef(mb1)
```

```
## (Intercept)          x1          x2
## -2.88886682 -0.05040406  1.57162276
```

```
mb2 <- lm(Y ~ x1 + x2, data = b2)
coef(mb2)
```

```
## (Intercept)          x1          x2
##  -2.8203529   0.5830797   1.9925660
```

Full bootstrap

```
it <- 5000
beta_hp <- rep(NA, it)
for (i in 1:it) {
  b <- sample_frac(df, replace = TRUE)
  beta_hp[i] <- lm(Y ~ x1 + x2, data = b)$coef[2]
}
sd(beta_hp)
```

```
## [1] 0.1955991
```

```
m1 <- lm(Y ~ x1 + x2, data = df)
summary(m1)$coef
```

```
##                Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -3.0149299  0.2057590 -14.652724 9.591452e-16
## x1           0.4959268  0.2034211   2.437932 2.050589e-02
## x2           1.5608675  0.1950372   8.002922 3.894623e-09
```

Thoughts on that...

It seemed to work fine but it is a bit odd because the bootstrap procedure suggests that *both* the X and the Y are random variables.

Can we devise a procedure that is in closer accordance with our idea of regression as conditioning on the values of X?

After conditioning on the values of X, Y is only random through the random vector ϵ. Why don't we bootstrap that?

$$\mathbf{Y} = X\beta + \epsilon$$

Bootstrap Residuals visualized

Bootstrap residuals

```
it <- 5000
beta_hp <- rep(NA, it)
m1 <- lm(Y ~ x1 + x2, data = df)
b <- df
for (i in 1:it) {
  b$Y <- m1$fit + sample(m1$res, replace = TRUE)
  beta_hp[i] <- lm(Y ~ x1 + x2, data = b)$coef[2]
}
sd(beta_hp)
```

```
## [1] 0.1930897
```

```
summary(m1)$coef
```

```
##                Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -3.0149299  0.2057590 -14.652724 9.591452e-16
## x1           0.4959268  0.2034211   2.437932 2.050589e-02
## x2           1.5608675  0.1950372   8.002922 3.894623e-09
```

Guidance on using the bootstrap

- If you are confident that you have the correct *mean function*, bootstrapping the residuals is more appropriate.
- If you have no idea about the form of the mean function, bootstrapping the cases is the safer / more conservative approach.

The Multivariate Normal Distribution

We can write *any* linear estimator as

$$\hat{\beta} = CY$$

Therefore $\hat{\beta}$ is a linear combination of the Y. If we assume

$$Y \sim N(X\beta, \epsilon)$$

Then we know (through MGFs) that $\hat{\beta}$ is also normally distributed. More specifically, when $C = (X'X)^{-1}X'$,

$$\hat{\beta} \sim N(\beta, \sigma^2(X'X)^{-1})$$

The Multivariate Normal Distribution, cont.

The general form of the multivariate Normal distribution is

$$\boldsymbol{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Where $\boldsymbol{X}$ is a $n \times p$ matrix, $\boldsymbol{\mu}$ is a $p \times 1$ mean vector, and $\boldsymbol{\Sigma}$ is a $p \times p$ variance/covariance matrix.

You can access densities and random number generated for this distribution via the `dmvnorm()` and `rmvnorm()` functions

```
library(mvtnorm)
rmvnorm(n, mean = mu, sigma = Sigma)
```

Simulating from MVN

First, specify parameters,

```
mu_vec <- c(1, 2, 3)
sigma_mat <- matrix(c(.5,  0, .7,
                      0, .5,  0,
                     .7,  0,  2),
                    ncol = 3, byrow = TRUE)
sigma_mat
```

```
##      [,1] [,2] [,3]
## [1,]  0.5  0.0  0.7
## [2,]  0.0  0.5  0.0
## [3,]  0.7  0.0  2.0
```

Simulating from MVN, cont.

then, generate random variables.

```
rmvnorm(1, mean = mu_vec, sigma = sigma_mat)
```

```
##             [,1]      [,2]     [,3]
## [1,] 0.2138873 0.4631653 2.079299
```

```
rmvnorm(1, mean = mu_vec, sigma = sigma_mat)
```

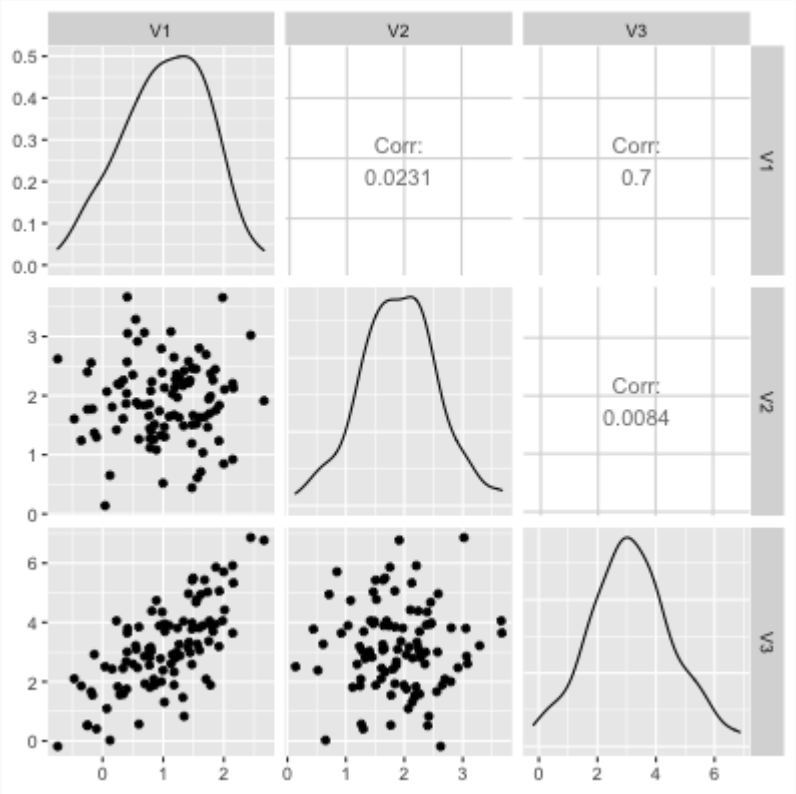
```
##             [,1]     [,2]     [,3]
## [1,] 1.223988 2.113755 3.117293
```

```
rmvnorm(10, mean = mu_vec, sigma = sigma_mat)
```

```
##              [,1]      [,2]     [,3]
##  [1,] 0.48087366 2.8899896 2.889368
##  [2,] 1.29128984 2.5380857 4.171362
##  [3,] 1.34164338 1.5339317 4.110056
##  [4,] 1.31996910 0.5135465 3.566813
##  [5,] 0.01468775 1.2799916 1.300745
```

Visualizing the MVN

```
X <- rmvnorm(100, mean = mu_vec, sigma = sigma_mat)
library(GGally)
ggpairs(as.data.frame(X))
```



Two ways to view a dataset Y

- n iid draws from $N(\mu, \sigma^2)$

- One draw from $N(\mu_n, \sum_{n \times n})$

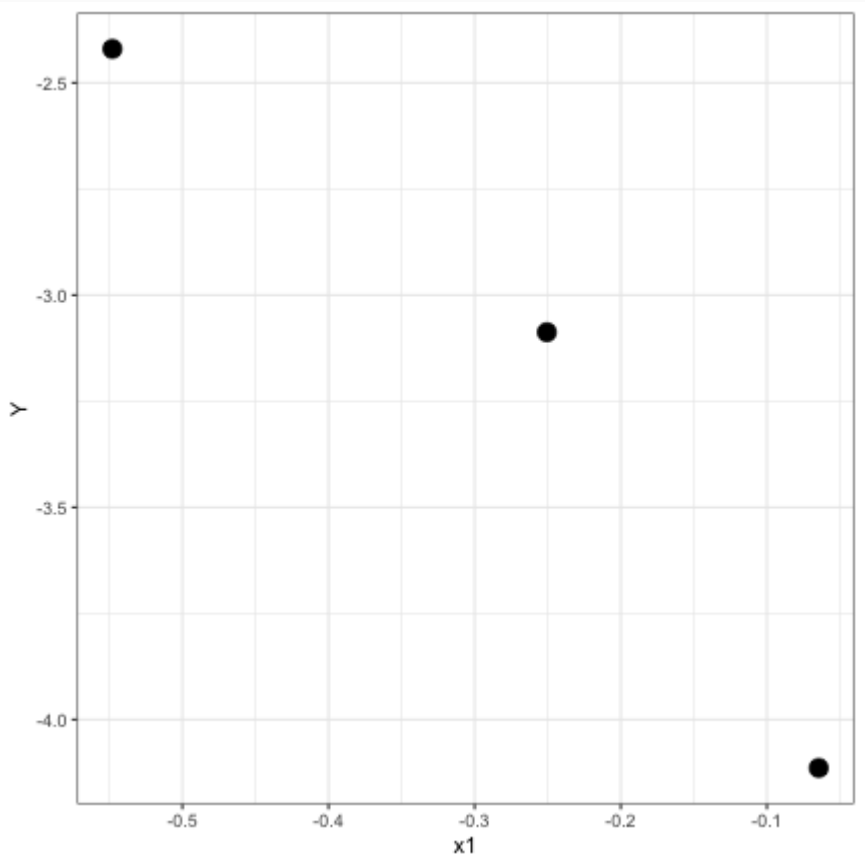
Small example $(n = 3)$

```
set.seed(8134)
n <- 3
x0 <- 1
x1 <- rnorm(n)
x2 <- rnorm(n)
x3 <- rnorm(n)
x4 <- rnorm(n)
X <- as.matrix(data.frame(x0, x1, x2, x3, x4))
B <- c(-3, .5, 2, 1, -.8)
sig_sq <- 2.25
epsilon <- rnorm(n, mean = 0, sd = sqrt(sig_sq))
Y <- X %*% B + epsilon
df <- data.frame(Y, X)
df
```

```
##           Y x0          x1         x2          x3          x4
## 1 -2.420069  1 -0.54789199  0.6245961 -0.07552716  0.02026105
## 2 -4.114054  1 -0.06456819 -1.4713007  0.49497462 -0.39986583
## 3 -3.087447  1 -0.25058242 -0.9935656  0.34607212 -1.02113706
```

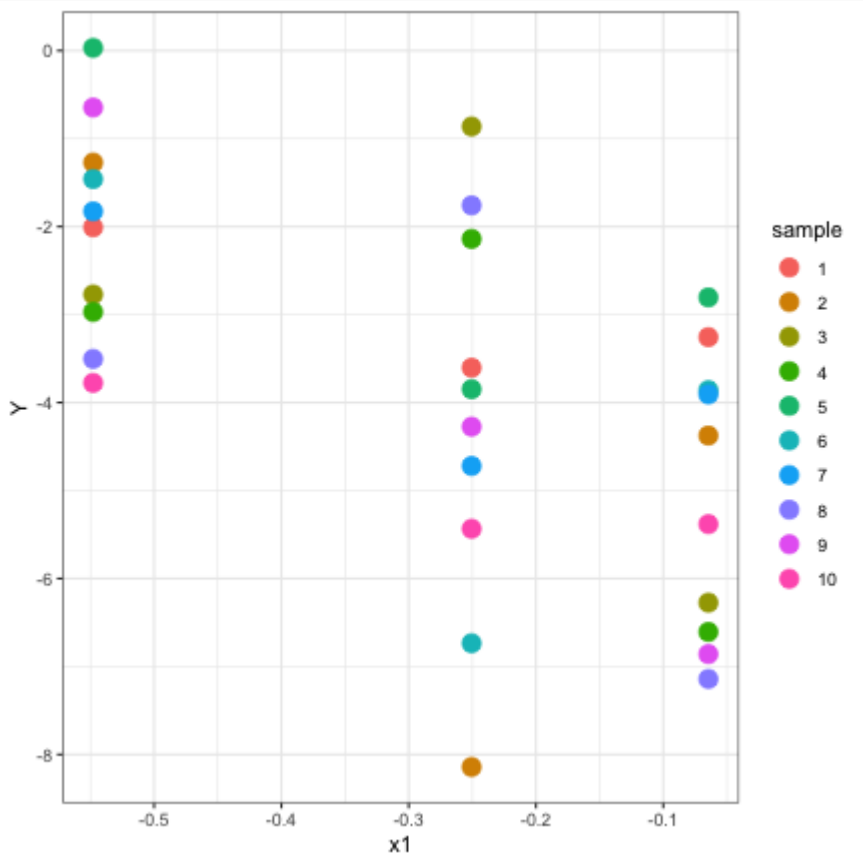
Three iid scalar draws

A partial view:



One vector draw

What does hypothetical resampling look like?



10 vector draws