Part 3 - Condition number of a matrix - Splines & Polynomial Regression

2023-03-14

- How to find the Condition Number of a matrix?
- Why is the condition number important?
- How to tune p? Given that changing p also changes the stability of the model.
- What is Splines and how to use Splines?
- How to do Spline?

Notes are adapted from Yu Zhao's DSC241 lab sessions at UCSD.

We are interested in the following linear model:

$$y = X\beta + \epsilon$$

- We aim to find the coefficient estimate $\hat{\beta}$, and $\hat{\beta}$ is affect by ϵ.
- The condition number of X measures how much can a change in ϵ affect the solution $\hat{\beta}$.
- If a matrix is singular, then its condition number is infinite. A finite large condition number means that the matrix is close to being singular.
- A problem with a low condition number is said to be well-conditioned, while a problem with a high condition number is said to be ill-conditioned.

How to find the Condition Number of a matrix?

condition number = max singular value / min singular value –> Use SVD to calculate singular values

```
X = matrix(c(3.2,-7.6,2.2,-5.2),nrow = 2)
svd_X = svd(X)
k_X = max(svd_X$d)/min(svd_X$d)
k_X
```

```
## [1] 1248.499
```

Check the sensitivity of the solution to changes in y

```
beta = matrix(c(1,2),nrow = 2)
y = X%*%beta
solve(X, y)
```

```
##      [,1]
## [1,]    1
## [2,]    2
```

```
solve(X, y + rnorm(2,0,0.5))
```

```
##            [,1]
## [1,]   9.539595
## [2,] -10.498366
```

```
solve(X, y + rnorm(2,0,0.5))
```

```
##           [,1]
## [1,]  11.40797
## [2,] -13.17606
```

```
solve(X, y + c(0,0.1))
```

```
##       [,1]
## [1,] -1.75
## [2,]  6.00
```

As we can see, the solution changes dramatically even there is only a small change in y.

How about the polynomial regression?

```
n = 100
p = 10
x = runif(100)
X = poly(x, degree = p, raw = TRUE)
X = cbind(rep(1, n), X) # get a complete desgin matrix, add a column of one
d = svd(X)$d  # svd outputs 3 matrices u v d --> we use d to extract the singular values

# d is a vector containing the singular values of x, of length min(n, p), sorted decreasingly.

max(d)/min(d) #condition number = max singular value / min singular value
```

```
## [1] 27406958
```

The condition number here is 2.7406958^{7}, which is very large, which means the model is not stable.

```
n = 100
p = 2
x = runif(100)
X = poly(x, degree = p, raw = TRUE)
X = cbind(rep(1, n), X)
d = svd(X)$d
max(d)/min(d)
```

```
## [1] 27.18113
```

Here we check with smaller p –> more stable, smaller cond number How small is small cond number? –> We can treat p as a tuning parameter

Why is the condition number important?

Wikipedia: In numerical analysis, the condition number of a function measures how much the output value of the function can change for a small change in the input argument. This is used to measure how sensitive a function is to changes or errors in the input, and how much error in the output results from an error in the input.

How to tune p? Given that changing p also changes the stability of the model.

We can use Hypothesis testing and anova to compare different model, each time we increase p, we add a column!

```
load("datasets/04cars.rda")
tmp = dat[,c(13,15,16,18,19)]
tmp = tmp[complete.cases(tmp),]
tmp = as.data.frame(tmp)
names(tmp) = c("hp", "mpg", "wt", "len", "wd")
dat = tmp
attach(dat)
fit1 <- lm(mpg ~ wt + len + wd + hp,data = dat)
fit2 <- lm(mpg ~ wt + len + wd + poly(hp, 2, raw = TRUE))
fit3 <- lm(mpg ~ wt + len + wd + poly(hp, 3, raw = TRUE))
fit4 <- lm(mpg ~ wt + len + wd + poly(hp, 4, raw = TRUE))
fit5 <- lm(mpg ~ wt + len + wd + poly(hp, 5, raw = TRUE))
fit6 <- lm(mpg ~ wt + len + wd + poly(hp, 6, raw = TRUE))
fit7 <- lm(mpg ~ wt + len + wd + poly(hp, 7, raw = TRUE))
anova(fit1,fit2,fit3,fit4,fit5,fit6,fit7)
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ wt + len + wd + hp
## Model 2: mpg ~ wt + len + wd + poly(hp, 2, raw = TRUE)
## Model 3: mpg ~ wt + len + wd + poly(hp, 3, raw = TRUE)
## Model 4: mpg ~ wt + len + wd + poly(hp, 4, raw = TRUE)
## Model 5: mpg ~ wt + len + wd + poly(hp, 5, raw = TRUE)
## Model 6: mpg ~ wt + len + wd + poly(hp, 6, raw = TRUE)
## Model 7: mpg ~ wt + len + wd + poly(hp, 7, raw = TRUE)
##   Res.Df    RSS Df Sum of Sq       F    Pr(>F)
## 1    382 3860.7
## 2    381 3527.6  1    333.08 50.4702 6.103e-12 ***
## 3    380 3100.6  1    427.08 64.7132 1.143e-14 ***
## 4    379 2714.1  1    386.47 58.5595 1.676e-13 ***
## 5    378 2557.3  1    156.74 23.7503 1.620e-06 ***
## 6    377 2504.3  1     52.99  8.0299  0.004849 **
## 7    376 2481.5  1     22.88  3.4667  0.063396 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we can see here, p values are all small ($p < 0.05$) until we go from model 6 to model 7. So we might want to stop at the model 6.

How to interpret Pr(>F) in F test?

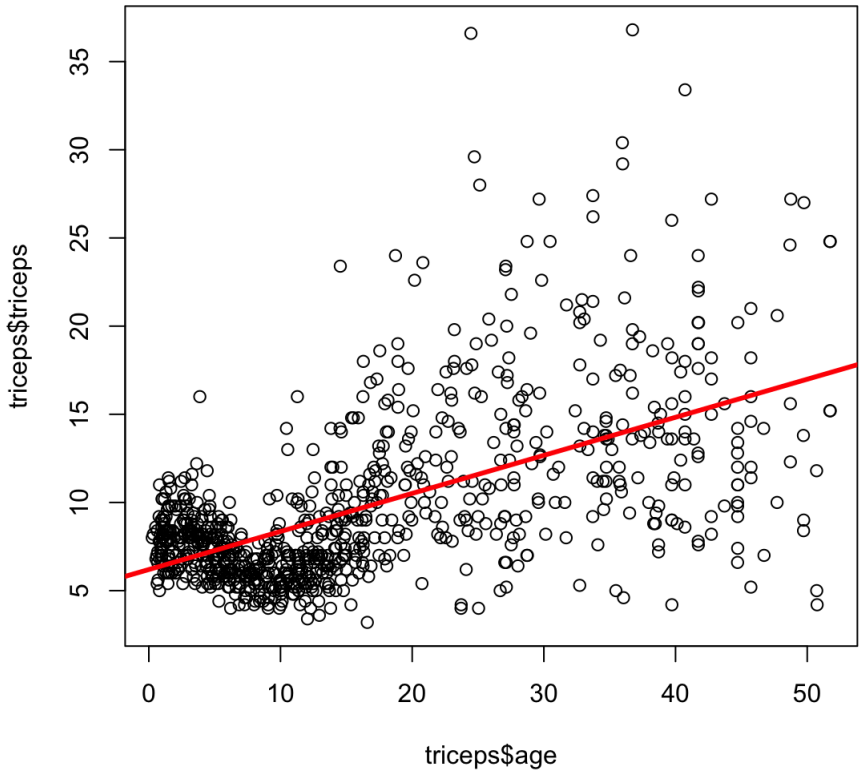
m1: $y = x + \epsilon$ m2: $y = x + x^2 + \epsilon$

–> F test p is small, reject null hypo (null hypo: adding column is not helpful) –> add one column is useful

What is Splines and how to use Splines?

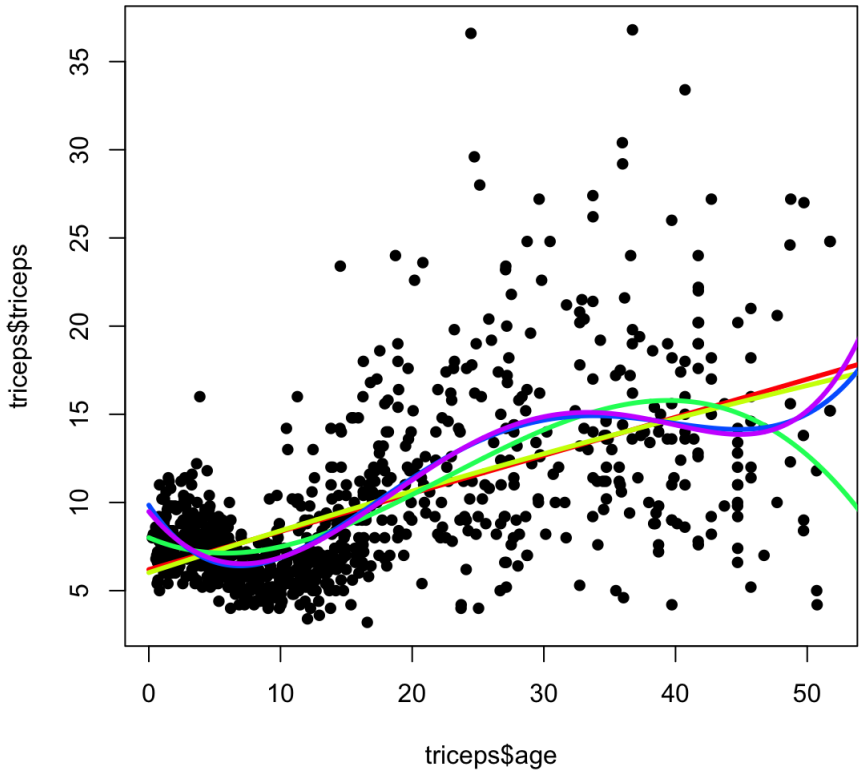
Consider data on a set of 892 females under 50 years collected in three villages in West Africa. We would like to explore the relationship between age (in years) and a crude measure of body fat, which is triceps skinfold thickness.

```
triceps <- read.csv('datasets/triceps.csv')
plot(triceps$triceps~triceps$age)
abline(lm(triceps~age,data = triceps),col = 'red',lwd = 3)
```



First try polynomial fit:

```
pts = seq(0,55,by = 0.1)
plot(triceps$age, triceps$triceps, pch = 16)
for (d in 1:5){
    fit = lm(triceps ~ poly(age, d, raw = TRUE),data = triceps)
    val = predict(fit, data.frame(age = pts))
    lines(pts, val, col=rainbow(5)[d], lwd = 3)
    }
```

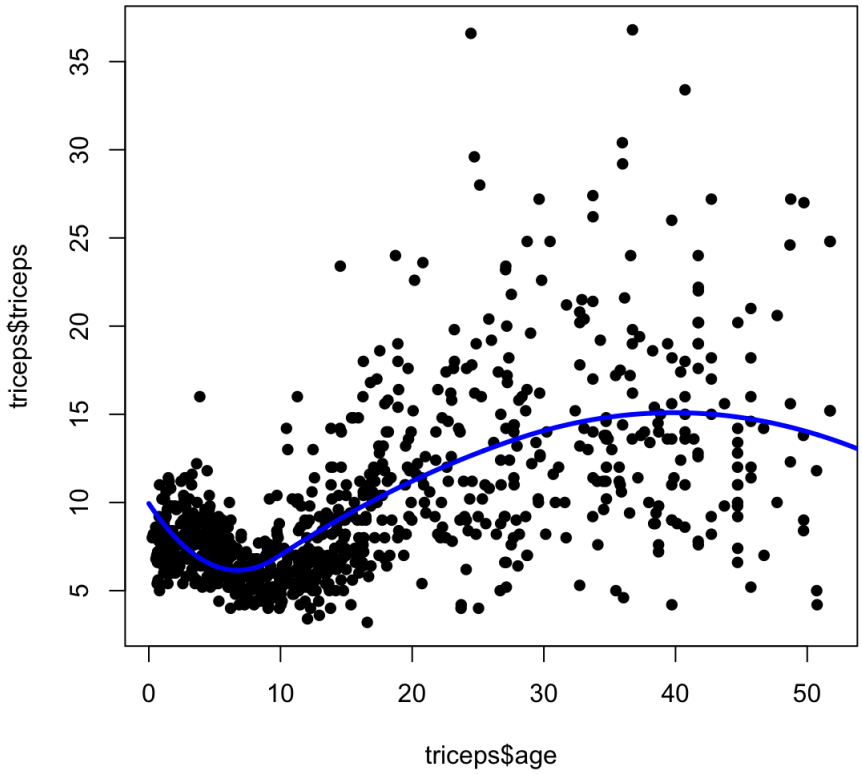


What is Spline? Spline Regression is one of the non-parametric regression technique. In this technique the dataset is divided into bins at intervals or points which we called as knots. Also this bin has its separate fit.

How to do Spline?

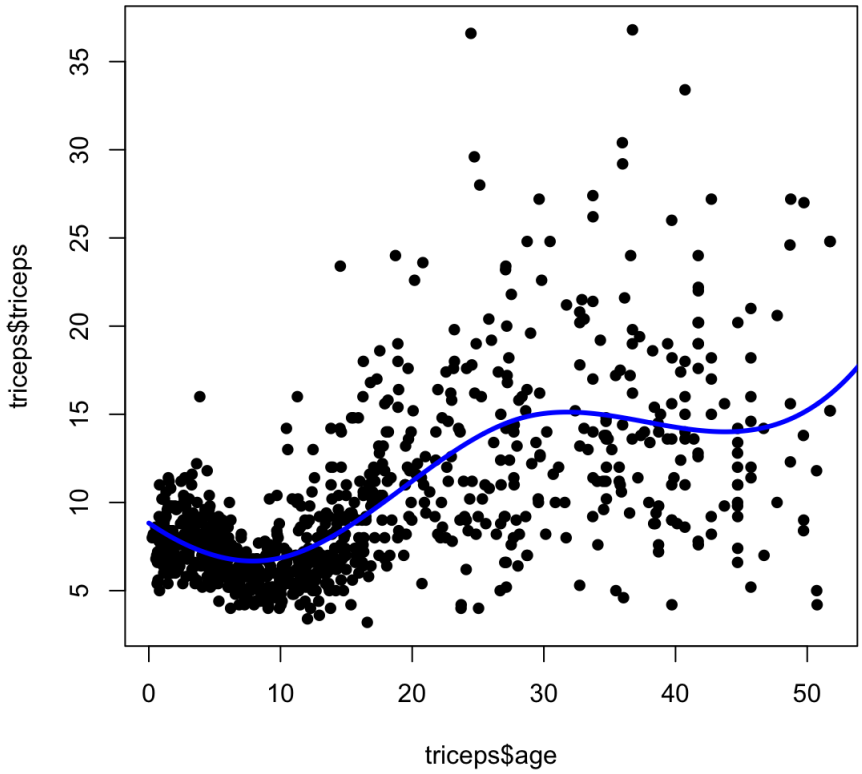
Spline of degree 2:

```
K = quantile(triceps$age, c(0.4,0.8), type=1)
plot(triceps$age, triceps$triceps, pch = 16)
fit = lm(triceps ~ bs(age,degree=2,knots=K),data = triceps)
val = predict(fit, data.frame(age = pts))
lines(pts, val, col="blue", lwd = 3)
```



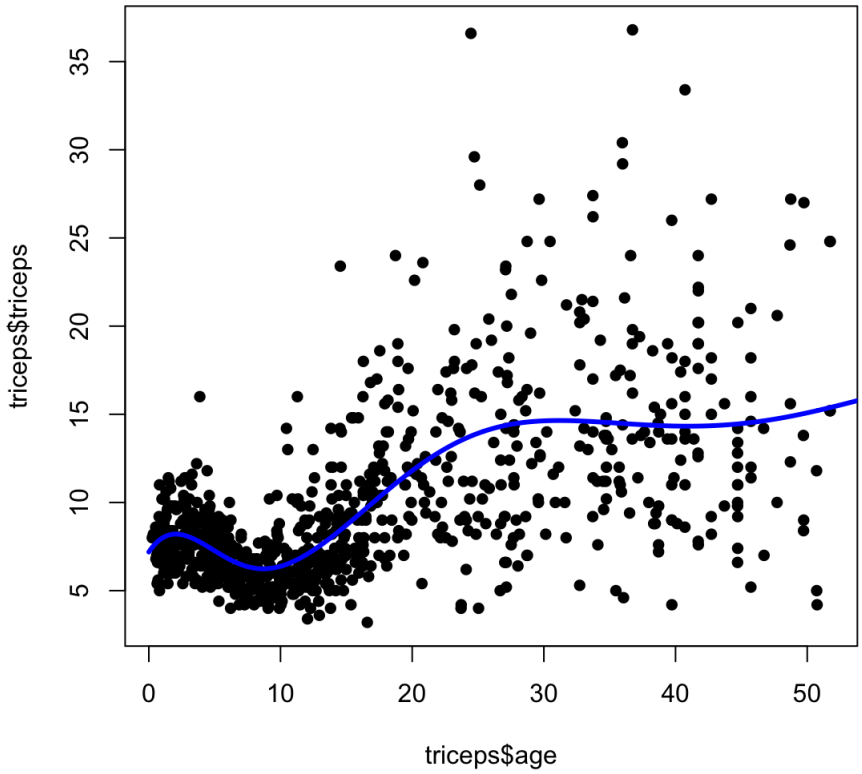
Spline of degree 3:

```
K = quantile(triceps$age, c(0.4,0.8), type=1)
plot(triceps$age, triceps$triceps, pch = 16)
fit = lm(triceps ~ bs(age,degree=3,knots=K),data = triceps)
val = predict(fit, data.frame(age = pts))
lines(pts, val, col="blue", lwd = 3)
```



Spline of degree 4:

```
K = quantile(triceps$age, c(0.4,0.8), type=1)
plot(triceps$age, triceps$triceps, pch = 16)
fit = lm(triceps ~ bs(age,degree=4,knots=K),data = triceps)
val = predict(fit, data.frame(age = pts))
lines(pts, val, col="blue", lwd = 3)
```



As we can see here, the higher degree give more smaller “curves”, which fit the polynomial regression in each bin. For example when `degree = 4` , there are 4 curves that fit the data in each bin.