Lab 3: More Linear Regression in R

Yifan Jin

9/23/2020

Today's Objectives

- Fitting multiple regression and fitting regression with transformed predictors(higher order pattern of the predictors)
- Including categorical variables in linear regression
- Brief introduction to interactions

```
library(ISLR)

data(Auto)
```

Fitting multiple regression with `lm()`

Suppose we also think vehicle weight will influence fuel efficiency. To fit a multiple linear regression, simply add more variables to the formula in `lm()`.

```
attach(Auto)

# fit a multiple linear regression
mlm.fit <- lm(mpg~horsepower+weight)

# view summary output
summary(mlm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower + weight)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0762  -2.7340  -0.3312   2.1752  16.2601
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 45.6402108  0.7931958  57.540  < 2e-16 ***
## horsepower  -0.0473029  0.0110851  -4.267 2.49e-05 ***
## weight      -0.0057942  0.0005023 -11.535  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.24 on 389 degrees of freedom
## Multiple R-squared:  0.7064, Adjusted R-squared:  0.7049
## F-statistic: 467.9 on 2 and 389 DF,  p-value: < 2.2e-16
```

A few things to note:

- Both predictors appear to have an influence on fuel efficiency
- The R-squared value has increased after adding another predictor

- The estimate of the coefficient for horsepower has changed

Suppose we want to add even more parameters to the model. The formula shortcut to add all variables is '.' (**Warning**: the '.' command is not compatible with 'attach', you'll have to specify the dataset in the lm call)

However in this case, the data contains non-numeric variables 'origin' and 'name' which we can remove from the model by subtracting them in the formula.

```
# fit a multiple linear regression
mlm.fit.2 <- lm(mpg~.-origin-name,data=Auto)

# view summary output
summary(mlm.fit.2)
```

```
##
## Call:
## lm(formula = mpg ~ . - origin - name, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6927 -2.3864 -0.0801  2.0291 14.3607
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.454e+01  4.764e+00  -3.051  0.00244 **
## cylinders    -3.299e-01  3.321e-01  -0.993  0.32122
## displacement  7.678e-03  7.358e-03   1.044  0.29733
## horsepower   -3.914e-04  1.384e-02  -0.028  0.97745
## weight       -6.795e-03  6.700e-04 -10.141  < 2e-16 ***
## acceleration  8.527e-02  1.020e-01   0.836  0.40383
## year          7.534e-01  5.262e-02  14.318  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.435 on 385 degrees of freedom
## Multiple R-squared:  0.8093, Adjusted R-squared:  0.8063
## F-statistic: 272.2 on 6 and 385 DF,  p-value: < 2.2e-16
```

Note also the p value for the horsepower increases, it is due to the phenomenon called colinearity(https://en.wikipedia.org/wiki/Multicollinearity).

Fitting regression with transformed predictors with `lm()`

Recall the plot we produced of residuals vs fitted values in our simple linear regression (y=mpg, x=horsepower). We can see a clear quadratic pattern in this plot which implies we could improve the fit by adding a quadratic term to our model.

There are three ways to do this:

(1) Create a new column in the data frame

```
# create a column containing the squared horsepower
Auto$horsepower2 <- horsepower^2

# fit a multiple linear regression on horsepower and horsepower^2
qm.fit <- lm(mpg~horsepower + horsepower2,data=Auto)
```

```r
# view basic output
qm.fit
```

```
##
## Call:
## lm(formula = mpg ~ horsepower + horsepower2, data = Auto)
##
## Coefficients:
## (Intercept)   horsepower  horsepower2
##   56.900100    -0.466190     0.001231
```

(2) Do the transformation directly in `lm()`

```r
# fit a multiple linear regression
qm.fit2 <- lm(mpg~horsepower+I(horsepower^2))

# view basic output
qm.fit2
```

```
##
## Call:
## lm(formula = mpg ~ horsepower + I(horsepower^2))
##
## Coefficients:
##     (Intercept)       horsepower  I(horsepower^2)
##       56.900100        -0.466190         0.001231
```

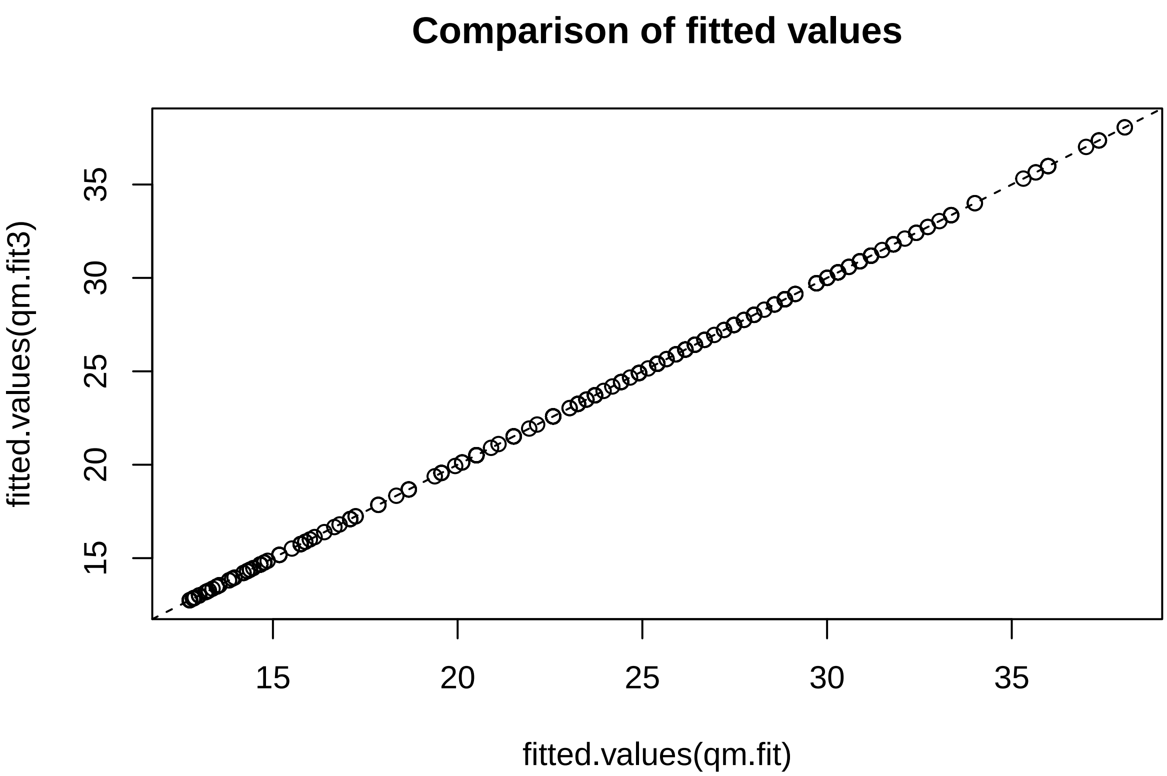
(3) Use the `poly()` command to fit a higher order polynomial model

```r
# fit a multiple linear regression
qm.fit3 <- lm(mpg~poly(horsepower,2))

# view basic output
qm.fit3
```

```
##
## Call:
## lm(formula = mpg ~ poly(horsepower, 2))
##
## Coefficients:
##          (Intercept)  poly(horsepower, 2)1  poly(horsepower, 2)2
##                23.45               -120.14                 44.09
```

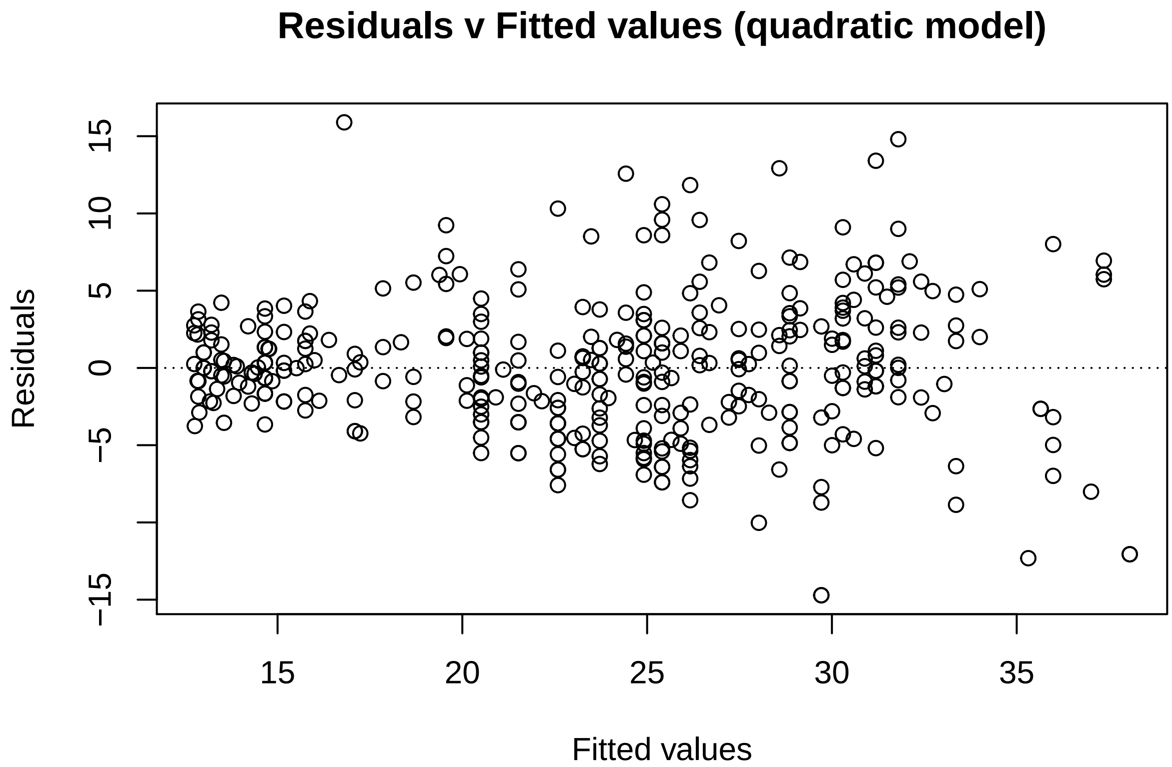
```r
# the poly command uses orthonormal polynomials, so it produces
# different coefficients but the same fitted values
plot(fitted.values(qm.fit),fitted.values(qm.fit3),
     main="Comparison of fitted values")
abline(a=0,b=1,lty=2)
```



The poly command can be used to fit even higher order polynomial models.

After adding a quadratic term, we have removed the pattern in the residual vs fitted value plot.

```
# quadratic model residuals v fitted values plot
y.hat <- predict(qm.fit)
e.hat <- residuals(qm.fit)
plot(y.hat,e.hat,
     xlab="Fitted values",ylab="Residuals",
     main="Residuals v Fitted values (quadratic model)")
abline(h=0,lty=3)
```



Categorical predictors

2-level categorical predictors

We first consider a binary categorical predictor into simple linear regression.

- A typical appearance for such predictor is the answer from 'Yes'/'No' questionnaire.
- For example, let 'x' be a binary predictor indicating whether you are from Michigan or not.
- The convention is to treat them as 0/1 indicators. That is, we label 'yes' as 1, while 'no' as 0. (Sometimes the other way around..)

Consider the following model

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i,$$

where x_i only takes 0 or 1, indicating the binary predictor. This means

$$y_i = \begin{cases} \beta_0 + \beta_1 + \varepsilon_i, & \text{if } x_i = 1 \\ \beta_0 + \varepsilon_i, & \text{if } x_i = 0 \end{cases}$$

- What's the interpretation of β_0 and β_1?
- β_1: The average difference in Y between the two categories.
- β_0: The average of Y for the baseline group (Here baseline is $x_i = 0$).

Following the same strategy in lecture, we can derive the expression for the estimator $\widehat{\beta}_0$ and $\widehat{\beta}_1$. Let $\bar{Y}_1$ be the sample average of y_i among those $x_i = 1$, while $\bar{Y}_0$ be the sample average of y_i among $x_i = 0$.

$$\begin{aligned} \widehat{\beta}_0 &= \bar{Y}_0 \\ \widehat{\beta}_1 &= \bar{Y}_1 - \bar{Y}_0 \end{aligned}$$

- Does those expressions seems intuitive?
- What would happen if we label `Yes` as 2 and `No` as 0?

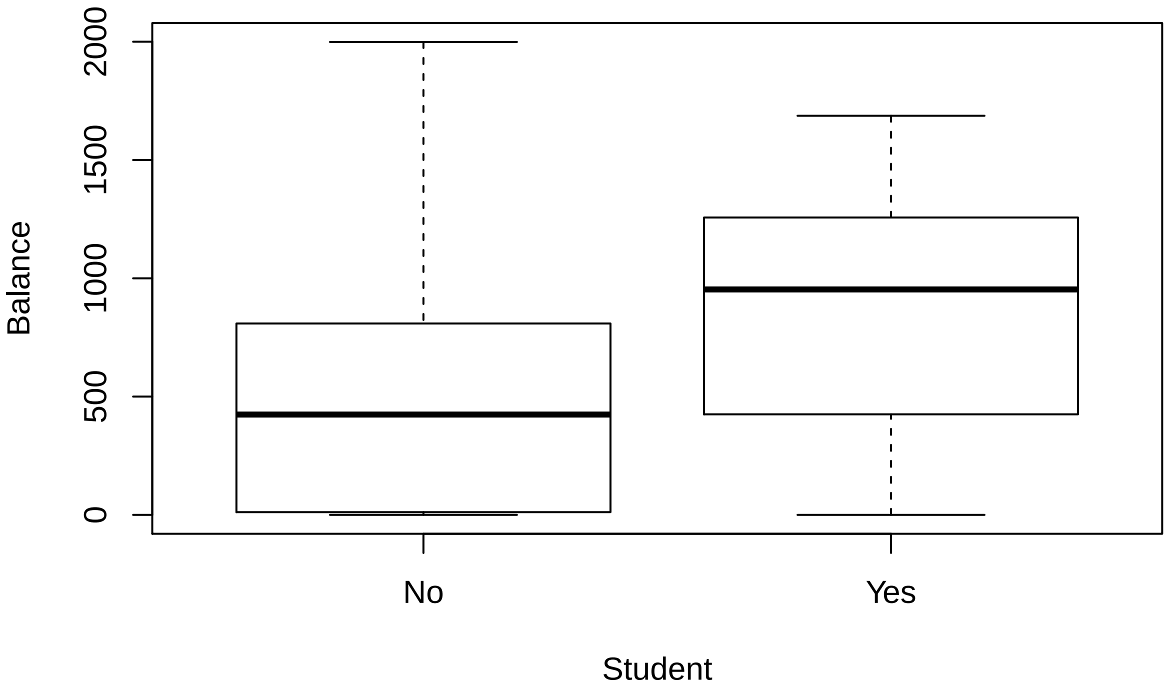
To see a real-world application, we turn to the `Credit` dataset from the `ISLR` package. The data set records information on credit card clients. We first try to model `Balance` using a binary predictor `Student`.

- `Balance` records the average credit balance for a client
- `Student` is binary, representing whether the user is a student or not.
- A summary of the variable `Student` shows it's indeed categorical.

```
library(ISLR)
data(Credit)
summary(Credit$Student)
```

```
##  No Yes
## 360  40
```

```
plot(Balance~Student, data = Credit)
```



Then we can fit a simple linear regression model of `Balance~Student`.

```
fit1 = lm(Balance~Student, data = Credit)
summary(fit1)
```

```
##
## Call:
## lm(formula = Balance ~ Student, data = Credit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -876.82 -458.82  -40.87  341.88 1518.63
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   480.37      23.43   20.50  < 2e-16 ***
```

```
## StudentYes    396.46      74.10    5.35 1.49e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 444.6 on 398 degrees of freedom
## Multiple R-squared:  0.06709,    Adjusted R-squared:  0.06475
## F-statistic: 28.62 on 1 and 398 DF,  p-value: 1.488e-07
```

- Is the coefficient for 'Student' significant?
- While R implement the regression above, how do we know if 'Yes' = 1 or 'No' = 1?

We can easily verify that the estimated values have the desired interpreration.

```r
beta0.hat = mean(Credit$Balance[Credit$Student == 'No'])
beta1.hat = mean(Credit$Balance[Credit$Student == 'Yes']) - mean(Credit$Balance[Credit$Student == 'No']
c(beta0.hat,beta1.hat)
```

```
## [1] 480.3694 396.4556
```

```r
coef(fit1)
```

```
## (Intercept)  StudentYes
##    480.3694    396.4556
```

If we would like to put 'Yes' as our baseline instead, the function `relevel` comes handy.

```r
fit2 = lm(Balance~relevel(Student, ref = 'Yes'), data = Credit)
summary(fit2)
```

```
##
## Call:
## lm(formula = Balance ~ relevel(Student, ref = "Yes"), data = Credit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -876.83 -458.83  -40.87  341.88 1518.63
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         876.8       70.3   12.47  < 2e-16 ***
## relevel(Student, ref = "Yes")No   -396.5       74.1   -5.35 1.49e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 444.6 on 398 degrees of freedom
## Multiple R-squared:  0.06709,    Adjusted R-squared:  0.06475
## F-statistic: 28.62 on 1 and 398 DF,  p-value: 1.488e-07
```

Another way of doing this is adjusting the order of the levels first.

```r
levels(Credit$Student)
```

```
## [1] "No"  "Yes"
```

```r
Credit$MyStudent =  factor(Credit$Student, levels = c("Yes", "No"))
levels(Credit$MyStudent)
```

```
## [1] "Yes" "No"
```

```r
fit2.1 = lm(Balance~MyStudent , data = Credit)
summary(fit2.1)
```

```
##
## Call:
## lm(formula = Balance ~ MyStudent, data = Credit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -876.83 -458.83  -40.87  341.88 1518.63
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    876.8       70.3   12.47  < 2e-16 ***
## MyStudentNo   -396.5       74.1   -5.35 1.49e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 444.6 on 398 degrees of freedom
## Multiple R-squared:  0.06709,    Adjusted R-squared:  0.06475
## F-statistic: 28.62 on 1 and 398 DF,  p-value: 1.488e-07
```

- The `level` function reveals all the possible levels of a categorical variable **in order**.
- The `factor` function creates a categorical variable in `R` with the specified level names **in order**.

Multiple regression

Now let's try to include some other continuous variable in the model.

```r
fit3 = lm(Balance~Student + Income + Rating + Age, data = Credit)
summary(fit3)
```

```
##
## Call:
## lm(formula = Balance ~ Student + Income + Rating + Age, data = Credit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -217.606  -79.887   -8.163   62.680  292.009
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -547.30470   21.46064 -25.503   <2e-16 ***
## StudentYes   417.50564   17.17164  24.314   <2e-16 ***
## Income        -7.79773    0.24218 -32.198   <2e-16 ***
## Rating         3.98073    0.05458  72.927   <2e-16 ***
## Age           -0.62418    0.30407  -2.053   0.0408 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 102.9 on 395 degrees of freedom
## Multiple R-squared:  0.9504, Adjusted R-squared:  0.9499
## F-statistic:  1892 on 4 and 395 DF,  p-value: < 2.2e-16
```

- How would you interpret the coefficient for `Student` now?

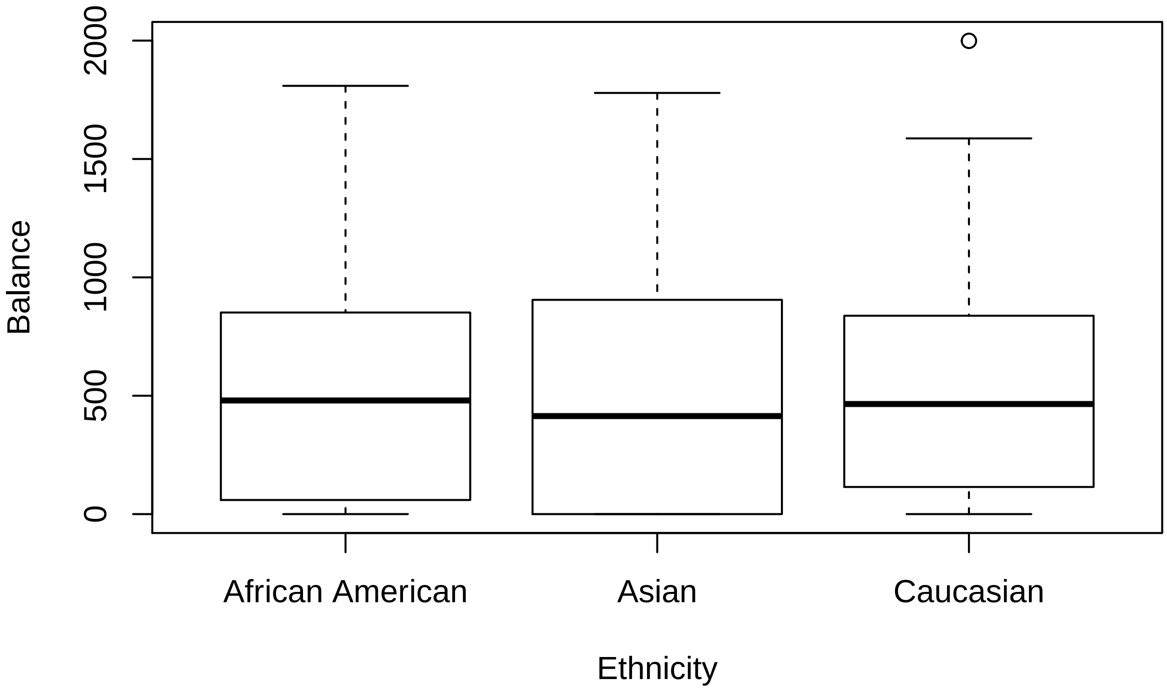
Multi-level categorical predictor

Now let's consider the case with more than two levels. First we take a look at the variable `Ethnicity`.

```
summary(Credit$Ethnicity)
```

```
## African American            Asian        Caucasian 
##               99              102              199
```

```
plot(Balance ~ Ethnicity , data = Credit)
```



- There are three levels.
- There does not seem to be a significant relationship between the balance and ethnicity.
- Would indexing `Ethnicity` as a integer of 0/1/2 work like the binary case?

To incorporate this, we introduce a set of 'dummy' variables. Suppose x has d distinct levels denoted as $f_1, \ldots, f_d$ (They are not numeric!). Let $z_1, \ldots, z_{d-1}$ be $d-1$ binary (dummy) variables that are:

$$z_k = \begin{cases} 1, & \text{if } x = f_k \\ 0, & \text{Otherwise} \end{cases}$$

Then we fit the regression as if we use those $d-1$ dummy variables $z_1, \ldots, z_{d-1}$ instead of the original categorical variable x in the linear regression.

- The dummy z_k indicates whether x takes the k-th value f_k or not.
- Why are there only $d-1$ dummy variables?
-
- At most one of the dummies z_k will equal to 1.
- When all the $d-1$ dummies are 0, we immediately know $x = f_k$.
- The level f_d serves as the baseline.

Now let's see how that works in practice.

```r
fit4 = lm(Balance~ Ethnicity, data = Credit)
summary(fit4)
```

```
##
## Call:
## lm(formula = Balance ~ Ethnicity, data = Credit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -531.00 -457.08  -63.25  339.25 1480.50
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          531.00      46.32  11.464   <2e-16 ***
## EthnicityAsian       -18.69      65.02  -0.287    0.774
## EthnicityCaucasian   -12.50      56.68  -0.221    0.826
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 460.9 on 397 degrees of freedom
## Multiple R-squared:  0.0002188,  Adjusted R-squared:  -0.004818
## F-statistic: 0.04344 on 2 and 397 DF,  p-value: 0.9575
```

- Which level is the baseline?
- Now, the coefficients represents the average difference with the **baseline** level.
- Does it appear to be a significant difference between each ethnicity?

Alternatively, we can create our own 0/1 valued dummy variables.

```r
Credit$AfAmer <- as.numeric(Credit$Ethnicity == 'African American')
Credit$Asian <- as.numeric(Credit$Ethnicity == 'Asian')
Credit$Cauc <- as.numeric(Credit$Ethnicity == 'Caucasian')
table(Credit$AfAmer)
```

```
##
##   0   1
## 301  99
```

- The `==` operator in `R` compares the left and right hand side, returning a logical value TRUE/FALSE.
- In our case, it returns a vector of the same length as `Credit$Ethnicity`.
- The `as.numeric` convert those logical variable to 1/0.

Suppose we use `African American` as the baseline level. Fitting a linear regression using our dummies gives the same result.

```r
fit5 = lm(Balance~ Asian + Cauc, data = Credit)
coef(fit5)
```

```
## (Intercept)       Asian        Cauc
##   531.00000   -18.68627   -12.50251
```

```r
coef(fit4)
```

```
##        (Intercept)     EthnicityAsian EthnicityCaucasian
##          531.00000          -18.68627          -12.50251
```

Of course we can involve more predictors in the model.

```r
fit6 = lm(Balance~ Ethnicity + Student + Income + Rating + Age, data = Credit)
summary(fit6)
```

```
##
## Call:
## lm(formula = Balance ~ Ethnicity + Student + Income + Rating +
##     Age, data = Credit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -218.14  -82.41  -11.24   64.17  291.85
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -558.91663   23.82767 -23.457   <2e-16 ***
## EthnicityAsian       19.11395   14.58032   1.311   0.1906
## EthnicityCaucasian    9.24769   12.68082   0.729   0.4663
## StudentYes          416.82254   17.20402  24.228   <2e-16 ***
## Income               -7.80138    0.24238 -32.186   <2e-16 ***
## Rating                3.98304    0.05465  72.889   <2e-16 ***
## Age                  -0.59631    0.30492  -1.956   0.0512 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 102.9 on 393 degrees of freedom
## Multiple R-squared:  0.9506, Adjusted R-squared:  0.9499
## F-statistic:  1261 on 6 and 393 DF,  p-value: < 2.2e-16
```

- Note the interpretation of the 'intercept' terms now involves the baseline of `Ethnicity` and `Student`.

Interactions

We will briefly introduce the interaction effect. An interaction term between two variables x_1 and x_2 means the product term $x_1 \cdot x_2$. For example, a linear regression with predictor x_1, x_2 and their interaction is:

$$y = \beta_0 + x_1\beta_1 + x_2\beta_2 + x_1x_2\beta_3 + \varepsilon$$

- This is particularly interesting when x_1 is categorical (binary).
- Intuitively, the marginal effect of x_2 depend on the level of x_1.

Let's look at the interaction between `Student` and `Rating`.

```r
fit7 = lm(Balance ~ Income + Student + Rating + Student:Rating , data = Credit)
summary(fit7)
```

```
##
## Call:
## lm(formula = Balance ~ Income + Student + Rating + Student:Rating,
##     data = Credit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -217.930  -78.225   -5.887   66.799  284.319
##
## Coefficients:
```

```
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -568.38426   14.07027 -40.396  < 2e-16 ***
## Income              -7.91290    0.23684 -33.410  < 2e-16 ***
## StudentYes         271.73432   43.97212   6.180  1.6e-09 ***
## Rating               3.95653    0.05456  72.518  < 2e-16 ***
## StudentYes:Rating    0.41548    0.11463   3.624 0.000327 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 101.8 on 395 degrees of freedom
## Multiple R-squared:  0.9515, Adjusted R-squared:  0.951
## F-statistic:  1937 on 4 and 395 DF,  p-value: < 2.2e-16
```

- The colon `:` represents the two-way interaction
- Alternatively we can use `Student*Rating` to represent both the main effect and the interaction: `Student + Rating + Student:Rating`.

```r
fit8 = lm(Balance ~ Income + Student*Rating , data = Credit)
summary(fit8)
```

```
##
## Call:
## lm(formula = Balance ~ Income + Student * Rating, data = Credit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -217.930  -78.225   -5.887   66.799  284.319
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -568.38426   14.07027 -40.396  < 2e-16 ***
## Income              -7.91290    0.23684 -33.410  < 2e-16 ***
## StudentYes         271.73432   43.97212   6.180  1.6e-09 ***
## Rating               3.95653    0.05456  72.518  < 2e-16 ***
## StudentYes:Rating    0.41548    0.11463   3.624 0.000327 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 101.8 on 395 degrees of freedom
## Multiple R-squared:  0.9515, Adjusted R-squared:  0.951
## F-statistic:  1937 on 4 and 395 DF,  p-value: < 2.2e-16
```