Lecture 7: Linear Regression

Cheng Lu

Overview

- Simple Linear Regression
- Multiple Regression
- Stepwise Regression
- Homework

Simple Linear Regression

Linear regression is an approach to modeling the relationship between response variable y and explanatory variable(s) x. y is modeled as a linear function of x:

Model

$$y = \alpha + \beta x + \epsilon$$

where α is intercept, β is slope, and $\epsilon \sim N(0, \sigma^2)$ is the error. Suppose we have n observations $(x_j, y_j), j = 1, \ldots, n$, then we can write

$$y_j = \alpha + \beta x_j + \epsilon_j, j = 1, \ldots, n$$

Simple Linear Regression

Parameters α and β are usually unknown, but we can estimate the parameters by minimizing the sum of squared errors.

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{\alpha,\beta} \sum_{j=1}^{n} (y_j - \alpha - \beta x_j)^2$$

which is given by solving the equations

$$\frac{\partial}{\partial \alpha} \sum_{j=1}^{n} (y_j - \alpha - \beta x_j)^2 = 0, \frac{\partial}{\partial \beta} \sum_{j=1}^{n} (y_j - \alpha - \beta x_j)^2 = 0$$

Simple Linear Regression

Here

$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}, \hat{\beta} = \frac{\sum_{j=1}^{n}(x_j - \bar{x})(y_j - \bar{y})}{\sum_{j=1}^{n}(x_i - \bar{x})^2}$$

are the estimators for the true parameters α and β respectively. Then for any given x, we have the prediction (regression line)

$$\hat{y} = \hat{\alpha} + \hat{\beta}x$$

and the fitted value of the jth observation is given by

$$\hat{y}_j = \hat{\alpha} + \hat{\beta}x_j, j = 1, \ldots, n$$

The residual for the jth observation is given by

$$\hat{\epsilon}_j = y_j - \hat{y}_j, j = 1, \ldots, n$$

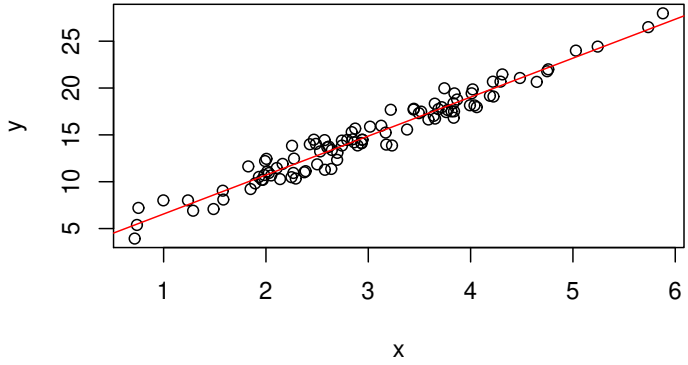
Simple Linear Regression

Example

```
> alpha <- 3
> beta <- 4
> x <- rnorm(100, mean = 3)
> epsilon <- rnorm(100)
> y <- alpha + beta * x + epsilon # model: y = 3 + 4x + epsilon
> plot(y ~ x) # scatter plot of the observations
> lm(y ~ x) # linear model y = alpha + beta*x + epsilon
Call:
lm(formula = y ~ x)
Coefficients:
(Intercept)            x
      2.400        4.157
> lm1 <- lm(y ~ x)
> abline(lm1, col = "red")  # add a red regression line for prediction
```

True model: $y = \alpha + \beta x + \epsilon = 3 + 4x + \epsilon$. Prediction: $\hat{y} = \hat{\alpha} + \hat{\beta} x = 2.4 + 4.157x$.

Simple Linear Regression



Simple Linear Regression

Example

```
> summary(lm1)
Call:
lm(formula = y ~ x)
Residuals:
     Min       1Q   Median       3Q      Max
-2.00288 -0.84300  0.03816  0.59530  2.06167
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.39953    0.29501   8.134 1.31e-12 ***
x            4.15740    0.09314  44.635  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.9856 on 98 degrees of freedom
Multiple R-squared:  0.9531,Adjusted R-squared:  0.9526
F-statistic:  1992 on 1 and 98 DF,  p-value: < 2.2e-16
```

Simple Linear Regression

Estimation

- The true parameters are $\alpha = 3, \beta = 4$
- Based on the summary, the estimators are $\hat{\alpha} = 2.39953, \hat{\beta} = 4.15740$.

P-values

- P-value helps you understand whether the factor is significant or not in your model
- The parameter is more significantly different from 0 when the P-value is smaller
- Usually we think the parameter is significant when its P-value is less than 0.05
- Based on the summary, the P-value for intercept α is 1.31e-12 (***), then we think the factor α is significant for the model (similar for β)

R-squared

- R-squared measures how well the linear model fits the data
- The model is better when R-squared is more close to 1

Simple Linear Regression

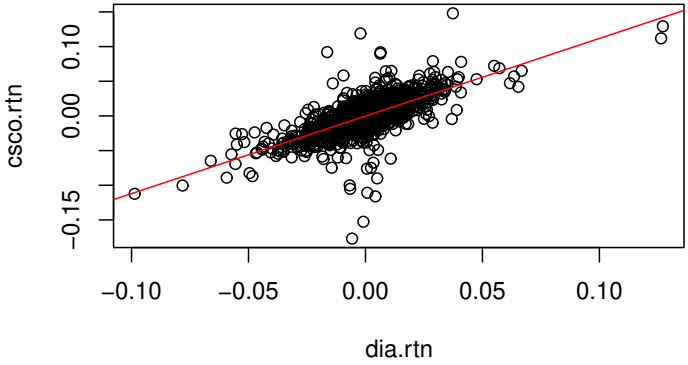
Linear regression with explanatory variable DIA return and response variable CSCO return

Example

```
> library(quantmod)
> getSymbols("CSCO")
> getSymbols("DIA")
> csco <- data.frame(CSCO)
> dia <- data.frame(DIA)
> # get adjusted close price
> csco.price <- csco$CSCO.Adjusted
> dia.price <- dia$DIA.Adjusted
> # get returns
> csco.rtn <- diff(log(csco.price))
> dia.rtn <- diff(log(dia.price))
> plot(csco.rtn ~ dia.rtn) # scatter plot of csco.rtn against dia.rtn
> lm2 <- lm(csco.rtn ~ dia.rtn) # model:csco.rtn = alpha + beta*dia.rtn + epsilon
> abline(lm2, col = "red") # add regression line
```

Assumption: $csco.rtn = \alpha + \beta * dia.rtn + \epsilon$. Prediction: $\hat{csco.rtn} = -0.0001719 + 1.0672768 * dia.rtn$

Simple Linear Regression



Multiple Regression

Model

In multiple regression, the response variable y depends on more than one explanatory variables, which are denoted by $x_1, x_2, \ldots, x_p$

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

Suppose we have n observations $(x_{j1}, x_{j2}, \ldots, x_{jp}, y_j), j = 1, \ldots, n$, then we can write

$$y_j = \alpha + \beta_1 x_{j1} + \beta_2 x_{j2} + \cdots + \beta_p x_{jp} + \epsilon_j, j = 1, \ldots, n$$

Multiple Regression

We can also solve for $\hat{\alpha}, \hat{\beta}_1, \ldots, \hat{\beta}_p$ by minimizing the sum of squared errors:

$$\min_{\alpha,\beta_1,\ldots,\beta_p} \sum_{j=1}^{n} \left(y_j - \alpha - \beta_1 x_{j1} - \beta_2 x_{j2} - \cdots - \beta_p x_{jp}\right)^2$$

Then the regression line is given by

$$\hat{y} = \hat{\alpha} + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$$

The fitted value of the jth observation is given by

$$\hat{y}_j = \hat{\alpha} + \hat{\beta}_1 x_{j1} + \hat{\beta}_2 x_{j2} + \cdots + \hat{\beta}_p x_{jp}, j = 1, \ldots, n$$

The residual of the jth observation is given by

$$\hat{\epsilon}_j = y_j - \hat{y}_j, j = 1, \ldots, n$$

Multiple Regression

Example

```
> setwd("C:/Users/demonew/Documents/Stevens/Graduate Courses/FE515/20s")
> bone <- read.csv("biomark.csv")
> # Variables:
> # VO+: a measure of bone formation
> # VO-: a measure of bone resorption
> # OC: a biomarker of bone formation
> # TRAP: a biomarker of bone resorption
> bone.model <- lm(voplus ~ vominus + oc + trap, data = bone)
> summary(bone.model)
```

In this example, we assume the model is given by

$$voplus = \alpha + \beta_1 * vominus + \beta_2 * oc + \beta_3 * trap + \epsilon$$

Multiple Regression

Example

```
Call:
lm(formula = voplus ~ vominus + oc + trap, data = bone)
Residuals:
    Min      1Q  Median      3Q     Max
-364.19 -158.57  -15.13  120.08  441.11
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -243.4877    94.2183  -2.584  0.01549 *
vominus        0.9746     0.1211   8.048  1.2e-08 ***
oc             8.2349     2.8397   2.900  0.00733 **
trap           6.6071    10.3340   0.639  0.52797
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 207.8 on 27 degrees of freedom
Multiple R-squared:  0.8844,Adjusted R-squared:  0.8715
F-statistic: 68.84 on 3 and 27 DF,  p-value: 9.031e-13
```

Prediction: $\hat{voplus} = -243.4877 + 0.9746 * vominus + 8.2349 * oc + 6.6071 * trap$.

Multiple Regression

Syntax	Model
$y \sim x$	$y = \alpha + \beta x$
$y \sim -1 + x$	$y = \beta x$
$y \sim x_1 : x_2$	$y = \alpha + \beta x_1 x_2$
$y \sim x + I(x^2)$	$y = \alpha + \beta_1 x + \beta_2 x^2$
$y \sim x_1 + x_2$	$y = \alpha + \beta_1 x_1 + \beta_2 x_2$
$y \sim x_1 * x_2$	$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$

Table: Different types of linear models

Multiple Regression

Example

```
> setwd("C:/Users/demonew/Documents/Stevens/Graduate Courses/FE515/20s")
> crops <- read.csv("grape crops.csv", header=T)
> yield <- crops$yield
> cluster <- crops$cluster.count
> plot(yield ~ cluster)
> lm(yield ~ cluster)
Call:
lm(formula = yield ~ cluster)
Coefficients:
(Intercept)      cluster
   -1.02790      0.05138
> lm.r <- lm(yield ~ cluster)# model:yield = alpha + beta*cluster + epsilon
> abline(lm.r, col='red')
```

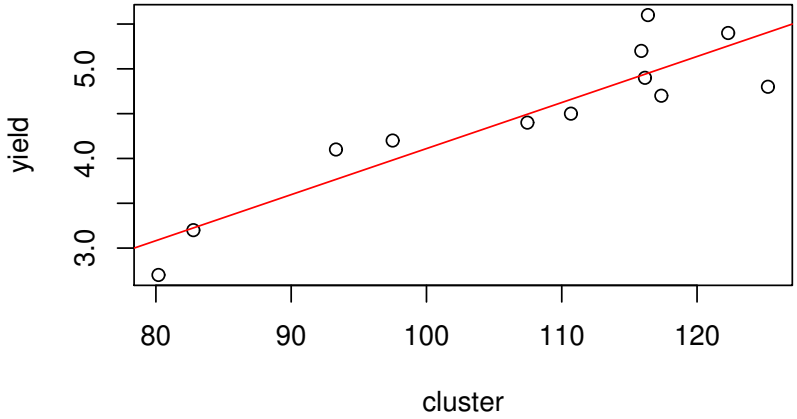
Model: $yield = \alpha + \beta * cluster + \epsilon$. Prediction: $\hat{yield} = -1.02790 + 0.05138 * cluster$

Multiple Regression

Example

```
> summary(lm.r)
Call:
lm(formula = yield ~ cluster)
Residuals:
     Min       1Q   Median       3Q      Max
-0.60700 -0.19471 -0.03241  0.23220  0.64874
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.02790    0.78355  -1.312    0.219
cluster      0.05138    0.00725   7.087 3.35e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.3641 on 10 degrees of freedom
Multiple R-squared:  0.834,Adjusted R-squared:  0.8174
F-statistic: 50.23 on 1 and 10 DF,  p-value: 3.347e-05
```

Multiple Regression



Multiple Regression

Example

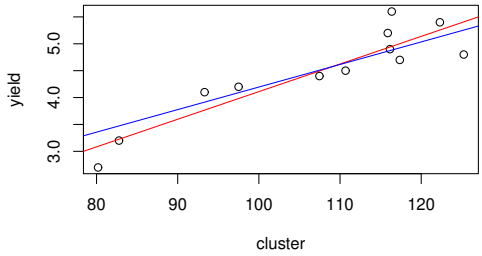
```
> newlm.r <- lm(yield ~ -1 + cluster)# model:yield = beta*cluster + epsilon
> summary(newlm.r)
Call:
lm(formula = yield ~ -1 + cluster)
Residuals:
     Min       1Q   Median       3Q      Max
-0.66443 -0.23611 -0.04086  0.20604  0.71761
Coefficients:
        Estimate Std. Error t value Pr(>|t|)
cluster 0.041956   0.001004    41.8 1.79e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.3758 on 11 degrees of freedom
Multiple R-squared:  0.9937,Adjusted R-squared:  0.9932
F-statistic:  1747 on 1 and 11 DF,  p-value: 1.789e-13
```

Model: $yield = beta * cluster + \epsilon$. Prediction: $\hat{yield} = 0.041956 * cluster$

Multiple Regression

Example

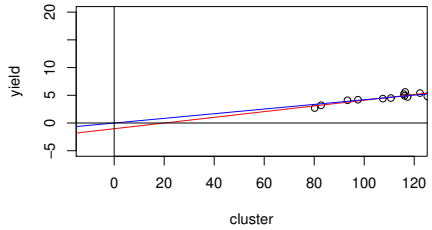
```
> plot(yield ~ cluster)
> abline(lm.r, col = 'red')
> abline(newlm.r, col = 'blue')
```



Multiple Regression

Example

```
> plot(yield ~ cluster, xlim = c(-10, 120), ylim = c(-5, 20))
> abline(lm.r, col = 'red')
> abline(newlm.r, col = 'blue')
> abline(h = 0)
> abline(v = 0)
```



Multiple Regression

There are also many other functions for the linear model objects

Example

```
> coef(newlm.r) # coefficients: beta_hat
   cluster
0.04195572
> resid(newlm.r)  # similar to "residuals(newlm.r)": epsilon_hat
         1          2          3          4          5          6          7
 0.7176133 -0.2726746 -0.1436586  0.1093177  0.3381716 -0.6644289 -0.4545339
         8          9         10         11         12
 0.0268436 -0.2239228  0.1851121 -0.1085612  0.2688159
> fitted(newlm.r) # fitted values: y_hat
       1        2        3        4        5        6        7        8        9
4.882387 3.472675 4.643659 4.090682 4.861828 3.364429 5.254534 4.873156 4.923923
      10       11       12
3.914888 4.508561 5.131184
```

Multiple Regression

Example

```
> lm.q <- lm(yield ~ cluster + I(cluster^2))#yield = alpha + beta1*cluster + beta2*cluster^2
> summary(lm.q)
Call:
lm(formula = yield ~ cluster + I(cluster^2))
Residuals:
     Min       1Q   Median       3Q      Max
-0.32087 -0.30127 -0.03626  0.19138  0.61498
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.121e+01  5.813e+00  -1.929   0.0858 .
cluster       2.552e-01  1.157e-01   2.207   0.0547 .
I(cluster^2) -9.971e-04  5.647e-04  -1.766   0.1113
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.3307 on 9 degrees of freedom
Multiple R-squared:  0.8767,Adjusted R-squared:  0.8493
F-statistic: 31.99 on 2 and 9 DF,  p-value: 8.123e-05
```

Model: $yield = \alpha + \beta_1 * cluster + \beta_2 * cluster^2 + \epsilon$. Prediction: $\hat{yield} = -11.21 + 0.2552 * cluster - 0.0009971 * cluster^2$

Multiple Regression

We can see the explanatory variable x (which is called cluster), fitted value $\hat{y}$, and y for each observations by creating a data frame

Example

```
> head(tab <- data.frame(cluster, fitted(lm.q), yield))
   cluster fitted.lm.q. yield
1   116.37     4.985023   5.6
2    82.77     3.080610   3.2
3   110.68     4.820867   4.5
4    97.50     4.192640   4.2
5   115.88     4.973427   5.2
6    80.19     2.841304   2.7
```

They are not well sorted.

Multiple Regression

We can use **order()** function to calculate the order of explanatory variable and sort each column of the data frame.

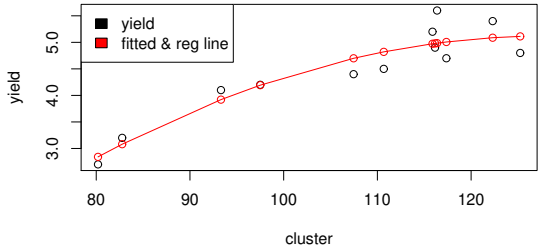
Example

```
> (order.cluster <- order(cluster))
 [1]  6  2 10  4 11  3  5  8  1  9 12  7
> head(tab[order.cluster,])
   cluster fitted.lm.q. yield
6    80.19     2.841304   2.7
2    82.77     3.080610   3.2
10   93.31     3.920353   4.1
4    97.50     4.192640   4.2
11  107.46     4.699363   4.4
3   110.68     4.820867   4.5
> tab <- tab[order.cluster,]
```

Multiple Regression

Example

```
plot(yield ~ cluster)
lines(tab$cluster, tab$fitted.lm.q., type = "o", col='red')
legend("topleft", legend = c("yield", "fitted & reg line"),
    fill = c("black", "red"))
```



Stepwise Regression

Sometimes we may not sure about what variables should be used in the multiple linear regression. In such cases, we can use stepwise regression to determine whether to add or delete variables from the model

- Forward selection: Adding variables from model with no explanatory variable
- Backward selection: Deleting variables from model with all explanatory variables

We can do forward selection or backward selection or both, given a measure of goodness of fit.

Stepwise Regression

Besides adjusted R-squared value, there are some other measures of goodness of fit.

AIC (Akaike information criterion)

- Measures the goodness of fit and penalize the number of parameters
- Small value indicate that the model is good

BIC (Bayesian information criterion)

- Similar to AIC, but penalize more on number of parameters
- Small value indicate that the model is good

Remember, you can use any kind of goodness criteria when evaluating the model performance. However, you can only use one and stick with it for each step of the model selection process.

Stepwise Regression

We can use **step()** function to do stepwise regression based on AIC

Example

```
bone <- read.csv("biomark.csv")
# Variables:
#   VO+: a measure of bone formation
#   VO-: a measure of bone resorption
#   OC: a biomarker of bone formation
#   TRAP: a biomarker of bone resorption
bone.model <- lm(voplus ~ vominus + oc + trap, data = bone)# full model
null.model <- lm(voplus ~ 1, data = bone)# model with no factor
full.model.formula <- voplus ~ vominus + oc + trap# scope for searching
```

Stepwise Regression

Example: Forward Selection

```
> step(null.model, full.model.formula, direction = "forward")
Start:  AIC=395.48
voplus ~ 1
          Df Sum of Sq      RSS    AIC
+ vominus  1   8093909  1993152 347.21
+ trap     1   5901115  4185945 370.21
+ oc       1   4388785  5698276 379.77
<none>                 10087061 395.48
Step:  AIC=347.21
voplus ~ vominus
       Df Sum of Sq     RSS    AIC
+ oc    1    809205 1183947 333.06
+ trap  1    463595 1529557 341.00
<none>              1993152 347.21
```

Stepwise Regression

Example Continued

```
Step:  AIC=333.06
voplus ~ vominus + oc
       Df Sum of Sq     RSS    AIC
<none>              1183947 333.06
+ trap  1     17658 1166289 334.60
Call:
lm(formula = voplus ~ vominus + oc, data = bone)
Coefficients:
(Intercept)      vominus           oc
   -234.144        1.019        9.404
```

When using forward selection, we calculate AIC value for all the models which adding one variable to the current model.

Stepwise Regression

In Backward selection, we need to start at full model

Example: Backward Selection

```
> step(bone.model, full.model.formula, direction = "backward")
Start:  AIC=334.6
voplus ~ vominus + oc + trap
          Df Sum of Sq     RSS    AIC
- trap     1     17658 1183947 333.06
<none>                 1166289 334.60
- oc       1    363268 1529557 341.00
- vominus  1   2797825 3964114 370.52
```

Stepwise Regression

Example Continued

```
Step:  AIC=333.06
voplus ~ vominus + oc
          Df Sum of Sq     RSS    AIC
<none>                  1183947 333.06
- oc       1    809205 1993152 347.21
- vominus  1   4514329 5698276 379.77
Call:
lm(formula = voplus ~ vominus + oc, data = bone)
Coefficients:
(Intercept)      vominus           oc
   -234.144        1.019        9.404
```

When using backward selection, we calculate AIC value for all the models which deleting one variable to the current model.

Stepwise Regression

Example: Both Forward and Backward

```
> step(null.model, full.model.formula, direction = "both")
Start:  AIC=395.48
voplus ~ 1
          Df Sum of Sq      RSS    AIC
+ vominus  1   8093909  1993152 347.21
+ trap     1   5901115  4185945 370.21
+ oc       1   4388785  5698276 379.77
<none>                 10087061 395.48
Step:  AIC=347.21
voplus ~ vominus
          Df Sum of Sq      RSS    AIC
+ oc       1    809205  1183947 333.06
+ trap     1    463595  1529557 341.00
<none>                  1993152 347.21
- vominus  1   8093909 10087061 395.48
```

Stepwise Regression

Example Continued

```
Step:  AIC=333.06
voplus ~ vominus + oc
          Df Sum of Sq     RSS    AIC
<none>                  1183947 333.06
+ trap     1     17658 1166289 334.60
- oc       1    809205 1993152 347.21
- vominus  1   4514329 5698276 379.77
Call:
lm(formula = voplus ~ vominus + oc, data = bone)
Coefficients:
(Intercept)      vominus           oc
   -234.144        1.019        9.404
```

When using both forward and backward selection, we calculate AIC value for all the models which adding or deleting one variable to the current model.

Stepwise Regression

Conclusion: The three methods forward regression, backward regression, and both forward and backward regression gives the same model

$$voplus = \alpha + \beta_1 * vominus + \beta_2 * oc + \epsilon$$

and the corresponding regression line (or fitted model) is given by

$$voplus = -234.144 + 1.019 * vominus + 9.404 * oc$$

Homework

- Download daily equity data of "JPM" and "WFC" (from 2007-01-01 to now), and calculate their daily log return
- Build a simple linear regression model using the WFC return as explanatory variable and the JPM return as response variable, summarize (**summary()**) the model
- Draw a scatter plot of JPM return against WFC return (i.e. WFC return in x-axis and JPM return in y-axis), and add a red regression line
- Download "cheese.csv" from canvas, build a multiple regression model to find the factors for cheese taste. Summarize (**summary()**) the model with all 3 factors and the model with no factor (null model)
- Use forward selection, backward selection, and both forward and backward selection to find the best model for cheese taste