### Indicator variables

#### New example

```
Data = data.frame(Y = c(.24, .21, .22, .32, .51, .56, .56, .67, .89, .92),
                  X1 = c(0, 0, 0, 0, 0, 0, 1, 1, 1, 1),
                  X2 = c(1, 1, 1, 2, 2, 2, 3, 3, 4, 4),
                  X3 = c("low", "low", "low", "low", "med", "med", "med", "high", "high", "high"))

#X1 has 2 levels
#X2 has 4 levels, quantitative categorical variables,
#X3 has 3 levels, qualitative categorical variables
```

#### Data

#### Indicators in practice:

- **THE FOLLOWING CORRESPONDS TO THE CODING CALLED “OPTION 1” IN CLASS:**
  - If variable is \{0,1\} only, you do NOT need to set any additional contrast options
  - Just use the variable name by itself or factor()
    ```
    Fit = lm(Y ~ X1, data=Data)
    Fit = lm(Y ~ factor(X1), data=Data)
    summary( Fit )
    ```
  - If variable is NOT in the form \{0,1\}, and you want the last level to be the base level:
    - Set options(contrasts()) to set the base level to be the LAST level of the factor, by typing:
      ```
      options(contrasts = c("contr.SAS", "contr.SAS"))
      #now, anytime factor() function is used, the base level will be the LAST level of the factor
      #(highest Number, or highest Letter in the alphabet)
      Fit = lm(Y ~ factor(X2) + factor(X3), data=Data)
      summary( Fit )
      ```
      - Alternatively, you may create a 'factor'/indicator variable and store it in your dataset:
        ```
        Data$X2ind = factor(Data$X2)
        Data$X3ind = factor(Data$X3)
        Data
        Fit = lm(Y ~ X2ind + X3ind, data=Data)
        summary( Fit )
        ```
  - If the variable is categorical, i.e. \{text\},
    - Use option 'contr.treatment' with base level set to desired level number, by typing:
      ```
      Data$X3factor = C( factor(Data$X3), contr.treatment(n=3, base=2) )
      #this creates column of [X3factor] inside your dataset Data,
      #which represents indicator variables with base level: 'low'
      #here, base level is chosen from [ 'high', 'low', 'med' ] factor levels in alphabetical order
      Fit = lm(Y ~ X3factor, data=Data)
      summary( Fit )
      ```
# The following part of code is for LEARNING about contrast function \( C() \).
# I advise you to run the code in R and see the results for yourself.
# You will rarely need to use these.

# create a categorical variable (with levels) from a numerical column
# can be used when only TWO levels/categories are present
factor(Data$X1)
   # here, base level is FIRST level of factor, SECOND level will be fitted by model
summary( lm(Y ~ factor(X1), data=Data) )

# create indicators with constrain: sum to zero (OPTION 2 IN CLASS NOTES), see (8.44)
alternative coding
\( C( \text{factor(Data$X1)}, \text{contr.sum} ) \)
\( C( \text{factor(Data$X2)}, \text{contr.sum} ) \)
\( C( \text{factor(Data$X3)}, \text{contr.sum} ) \)

# indicators that contrasts each level with base level (specified by 'base')
# by default, base level is the FIRST level, or FIRST letter in alphabet, seen in dataset:
\( C( \text{factor(Data$X1)}, \text{contr.treatment} ) \)
\( C( \text{factor(Data$X2)}, \text{contr.treatment} ) \)
\( C( \text{factor(Data$X3)}, \text{contr.treatment} ) \)
# to set baseline: to SECOND level seen in the dataset
\( C( \text{factor(Data$X3)}, \text{contr.treatment}(n=3, \text{base}=2) ) \)
   # 'n' is the total number of levels present in X
   # 'base' is the specified baseline level
# to create baseline to be the LAST level, do {one} of the following, see (8.35):
   # 1: change 'base' in 'contr.treatment'
   # 2: use 'contr.SAS
\( C( \text{factor(Data$X2)}, \text{contr.treatment}(n=4, \text{base}=4) ) \)
\( C( \text{factor(Data$X3)}, \text{contr.treatment}(n=3, \text{base}=3) ) \)
\( C( \text{factor(Data$X1)}, \text{contr.SAS} ) \)
\( C( \text{factor(Data$X2)}, \text{contr.SAS} ) \)
\( C( \text{factor(Data$X3)}, \text{contr.SAS} ) \)

# note, with qualitative variables, the order is chosen based on dictionary order
# so: level1 = "high", level2 = "low", level3 = "med", because of alphabetical ordering

### Model Selection
# Example: Grocery Retailer: Problem 6.9
Data = read.table("CH06PR09.txt")
names(Data) = c("Hours","Cases","Costs","Holiday")
Fit = lm(Hours~Cases+Costs+Holiday, data=Data)

# Sub-models are: only X1, or only X2, or only X3, or just X1 and X2, or just X1 and X3, or
# just X2 and X3, or all three variables; also models including powers of these variables
# (appropriately centered), or interactions like X1X2, or other transformations (square roots,
# logs, etc.)
#Method 1:

leaps() function: searches for the best subsets of predictors using specified criterion

#This is found in the package leaps, which must first be loaded:
library(leaps)

#If R can't find the package you will need to go to the R repository via
#the Packages menu and the Install package(s)... option to download it and install it.

leaps( x=Data[,]2:4, y=Data[,]1, names=names(Data)[2:4], method="Cp")

#input:
#x: matrix consisting of the predictor variables
#y: vector consisting of the response variable
#names: of the predictor variables
#method: criterion to use. Possible choices: "r2", "adjr2", "Cp"

#output:
#$which: each row is a sub-model, variables used are designated by TRUE
#$Cp: value of the Mallows' Cp criterion for each sub-model, in the same order

###Goal of model selection: Choose model that maximizes/minimizes a chosen criterion.
#1) Minimizes Mallows' Cp Criterion, or
#2) Maximizes R-Square, or Adjusted-R-Square

#In class we used 3 criterions at once, "r2", "adjr2", "Cp",
#however, leaps() can take one criterion at a time.
leaps( x=Data[,]2:4, y=Data[,]1, names=names(Data)[2:4], method="r2")
leaps( x=Data[,]2:4, y=Data[,]1, names=names(Data)[2:4], method="adjr2")

#Method 2:

#Make a list of each sub-model you wish to consider, then fit a linear model
#for each sub-model individually to obtain the selection criteria for that model.

#Start with the full model, then use:
#update() function: to remove and/or add predictors step-by-step, One-by-One.
NewMod = update( Fit, .~. - Costs )

#We started with full model Fit and deleted just one variable, Costs.
#Then fit a new model named NewMod with only the remaining predictors.

NewMod

#to modify NewMod to fit another model without Costs and Cases, delete Cases from NewMod
NewMod = update( NewMod, .~. - Cases)
NewMod

#to add Costs back into the model (but not Cases)
NewMod = update( NewMod, .~. + Costs)
NewMod

#In each Step,
#Retrieve R-Squared or Adjusted-R-Squared value from summary() output:
summary(NewMod)

#Calculate Cp criterion manually by formula (9.9) (see p.358): you need:
#MSE: MSE comes from the full model with all the potential predictor variables, Fit.
#SSEp: SSE for the sub-model in the ANOVA table for that sub-model.
#n: number of observations in the data set.
#p: number of parameters in the sub-model (with p-1 predictor variables).
MSE = anova(Fit)[4,3]
SSEP = anova(NewMod)[3,2]
n = nrow(Data)
p = 3
Cp = SSEp / MSE - (n - 2*p)
Cp

### Method 3:
Similar to Method 2, yet the re-fitting of new models is done through function lm().

### Few more useful tricks

#Example: Grocery Retailer: Problem 6.9
Data = read.table("CH06PR09.txt")
names(Data) = c("Hours","Cases","Costs","Holiday")
dim(Data)

#Useful for removing outliers, or for data-splitting (used in model validation):
#remove ONE row from the dataset, say row #23:
DataNew = Data[-23, ]
#remove THREE specific rows from the dataset, say rows #2, 5, and 19:
DataNew = Data[-c(2,19,5), ]  #order does not matter
#get part of the dataset, say rows #1-30
DataNew = Data[1:30, ]  #by subsetting wanted rows
DataNew = Data[-(31:52), ]  #by removing unwanted rows

#Problem 9.25 asks to consider observations 57-113 from your dataset,
#instead of the full dataset with rows 1-113.
Data = read.table("APPENCO1.txt")
dim(Data)
#zoom-in on observations (rows) 57-113:
DataNew = Data[57:113, ]
#then work with DataNew.