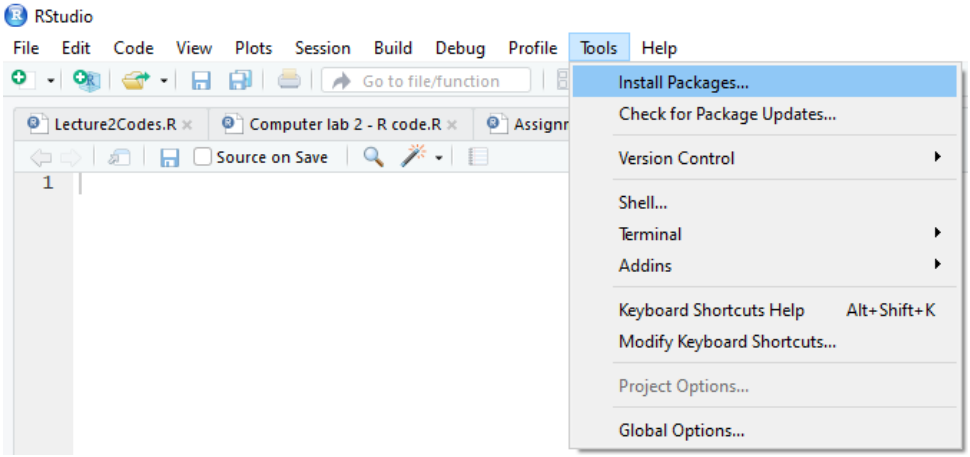
Computer lab 3 in MMS075, Feb 6, 2020

1. In this computer lab, we will work with the Carseats dataset that is used in Section 3.6.6 of the ISL book. Several datasets from ISL are available in R; to access these, type:
   library(ISLR)

   Did you get an error message? That's probably because this library is not installed on your computer. To install it, either use the command **install.packages("ISLR")** or use the Install Packages function of RStudio (see picture below)
   Note: Unfortunately, in the instructions for computer lab 3, this command was misspelled (it was written as "install.package", i.e. there was no "s" at the end), this is why it did not work for some people to install the ISLR package this way.



   Now that you have installed ISLR, you need to tell RStudio that you want to use it, and hopefully, the corresponding command will not cause an error message this time:
   library(ISLR)

   To see the content and description of the ISLR library, type:
   help(package=ISLR)

   Now you can check the Carseats dataset:
   fix(Carseats)
   Don't forget to close the editor before you continue with other commands!

   If you just wanted to see the names of variables included in the data, you could type:
   names(Carseats)

   To see the size of the dataset, type:
   dim(Carseats)
   The output shows the dimensions of the dataset, i.e. number of rows and number of columns.

   To learn what the different variables mean, type:
   ?Carseats

2. How good are the different variables in predicting the sales? Fit a linear model that includes all variables to learn about this. Perform backward variable selection: in each step, remove the variable with the highest p-value until you get a model where all variables are significant. Recall: the relevant function for fitting linear regression models is "lm(…)" and you get a summary of the model by typing "summary(…)"

   Code example:

```
attach(Carseats)
model=lm(Sales~CompPrice+Income+Advertising+Population+Price+ShelveLoc+Age+Education+Urban+US)
summary(model)
model=lm(Sales~CompPrice+Income+Advertising+Price+ShelveLoc+Age+Education+Urban+US)
summary(model)
model=lm(Sales~CompPrice+Income+Advertising+Price+ShelveLoc+Age+Education+US)
summary(model)
model=lm(Sales~CompPrice+Income+Advertising+Price+ShelveLoc+Age+US)
summary(model)
model=lm(Sales~CompPrice+Income+Advertising+Price+ShelveLoc+Age)
summary(model)
```

   Note: If you copied the names from the summary to the next line, it was important that you did not copy the level values of qualitative variables to the new command lines, but only the name of the variable. For example, including "USYes" in one of the command lines above would have resulted in an error, while "US" could be used as a variable name.

3. Was it boring to type all variable names to get the all variable model? Replace CompPrice+Income+Advertising+…+US in your code with a single dot (yes, just the "." symbol without the quotation signs!). What to do if we need almost all variables except Population and Education? Use ".-Population-Education".

   Code example:

```
model=lm(Sales~.,data=Carseats)
summary(model)
model=lm(Sales~.-Population,data=Carseats)
summary(model)
model=lm(Sales~.-Population-Urban,data=Carseats)
summary(model)
model=lm(Sales~.-Population-Urban-Education,data=Carseats)
summary(model)
model=lm(Sales~.-Population-Urban-Education-US,data=Carseats)
summary(model)
```

   Note: When using ~. as a shorthand for using all predictors, it is essential to specify the dataset that you are using, i.e. to add that data=Carseats, otherwise you just get an error message. This way, the instructions were misleading – if you did not have data=Carseats in your code before, then you needed to add that part to make it work.

4. According to this dataset, what is the contribution of a good shelf location compared to a bad one? What is the contribution of a medium shelf location? How is the qualitative variable ShelveLoc represented in this model?
   The contribution can be quantified from the final model from backward variable selection. Looking at the summary of that model, we see that the coefficient estimates for ShelveLocGood and ShelveLocMedium are 4.836 and 1.952 respectively. This means that a good quality of the shelving location for the car seats compared to a bad quality shelving location is estimated to result in 4836 units difference in sales (because the variable "Sales" represented thousand sold units), while a medium location compared to a bad location is predicted to yield a difference of 1952 sold units.
   The qualitative variable ShelveLoc has 3 levels: bad, medium and good, and it is represented by 2 dummy variables: ShelveLocGood and ShelveLocMedium in the models. As usual, the baseline level, which was chosen "Bad" by R, does not need a separate dummy variable – we know that if all other dummy variables take value 0, then we are in the baseline level. However, these variables only appear in the model summaries, R does not create actual variable objects called like this (and therefore, if you just write "ShelveLocGood" as a command, you will get an error message).

5. To see the dummy variables that R defined for this variable, type:
   contrasts(ShelveLoc)

6. We will now learn how to do best subset variable selection in R. The relevant function is called "regsubsets" and is included in a library called "leaps". Therefore, type:
   library(leaps)
   If you get an error message, install the relevant package and type the above command again
   R code example:
   install.packages("leaps")
   library(leaps)

7. To tell R to do the model selection and display a summary, type:
   BestSSModel=regsubsets(Sales~.,data=Carseats)
   summary(BestSSModel)

   This will select the best models up to 8 variables.
   Note: The output should be understood as follows: row 1 corresponds to the best possible model among all models that use only one predictor, and the star is in the column of the predictor that should be included for this model. Row 2 corresponds to the best possible model among all models that use exactly two predictors, and the star is in the columns of the two predictors that should be included for this model. Row 3 corresponds to the best possible model among all models that use exactly three predictors, and the star is in the columns of the three predictors that should be included for this model. And so on, until row 8 showing the best possible model among all models that use exactly 8 predictors.

   We would like to consider all possible models, so we use the "nvmax" argument to increase the number of considered variables to the number of variables in the dataset (**Note:** including the dummy variables that are defined for different levels of the qualitative variables) and look at the summary:
   BestSSModel=regsubsets(Sales~.,data=Carseats,nvmax=11)

summary(BestSSModel)

We will soon see that it makes sense to define an object that contains the relevant data in the summary, so let's do that and check what kind of information it contains:
BestSSsummary=summary(BestSSModel)
names(BestSSsummary)

For example, we can see the R^2 values of the best models of different sizes:
BestSSsummary$rsq

Note: The output here describes the R^2 values of models of different sizes; for example, the best model among all models with 1 variable has R^2 value of 0.2505104, the best 2-variable model has R^2 value of 0.4699029, and so on, until the best 11-variable model that has R^2 value of 0.8734133. Of course, the best 11-variable model is the same as the model with all variables, because in this dataset, there can only be one model with this many variables.

We will now choose the best model of *any* size from these models (which is the point with the best subset selection algorithm). Recall, however, that there are different possibilities for measuring the quality of the model, e.g. Mallow's C_p, AIC, BIC and adjusted R^2. We will now see that they do not always give the same best model. For BIC, the best model is the one with the lowest value. For adjusted R^2, the best model is the one with the highest value. Use the "which.max" and "which.min" functions to check which models these are:
which.max(BestSSsummary$adjr2)
which.min(BestSSsummary$bic)

Note: The first output means that the best 10-variable model is the one that gives the highest adjusted R^2 value while the second output means that the best 7-variable model gives the best BIC value.

To get an even better understanding, we make some plots to show how the quality of models of different sizes changes. We want to show 4 plots together, so it is good to divide the plotting window as 2x2:
par(mfrow=c(2,2))

We plot the different values for adjusted R^2 in the first quarter:
plot(BestSSsummary$adjr2,xlab="Number of variables",ylab="Adjusted R2")
After this, plots will fill up the quarters of this window. If you make a mistake and want to start again with the plot, type the following command:
dev.off()

We mark the point with the highest value:
BestModelSizeAdjR2=which.max(BestSSsummary$adjr2)
points(BestModelSizeAdjR2,BestSSsummary$adjr2[BestModelSizeAdjR2],col="red",cex=2, pch=20)

Note: in the second line, we use "points" which adds points to existing plots. Therefore, if you have tried dev.off() above, that has removed the plot and now the "points" command

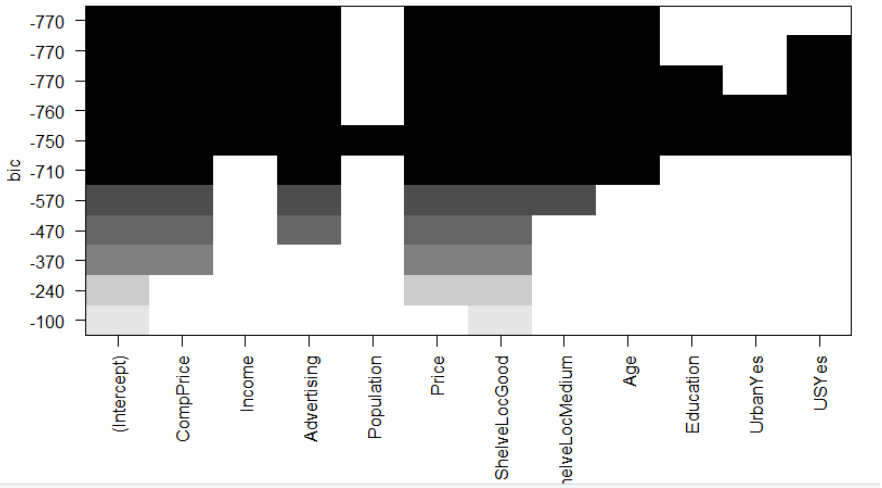
will give an error message. You need to use the above plot command first and only then "points".

Note2: in the first line above, we define a variable that corresponds to the number of variables where the best model has maximum adjusted R^2 value. As we have seen in an earlier output that the best 10-variable model is the one with maximum R^2 value, an alternative would be to avoid defining that variable and writing 10 instead, as follows:
points(10,BestSSsummary$adjr2[10],col="red",cex=2,pch=20)

We do the same using BIC and mark the point with the lowest value:
plot(BestSSsummary$bic,xlab="Number of variables",ylab="BIC")
BestModelSizeBIC=which.min(BestSSsummary$bic)
points(BestModelSizeBIC,BestSSsummary$bic[BestModelSizeBIC],col="red",cex=2,pch=20)

To avoid the manual work with plotting, we can use a built-in function in R:
plot(BestSSModel,scale="adjr2")
plot(BestSSModel,scale="bic")

Note: These are actually very informative plots. Take the second one, for example:



The top row has 8 black marks, namely in the columns of the intercept plus the 7 predictors that should be used in the best 7-variable model. We have seen earlier that the best 7-variable model gives the lowest BIC value – this plot confirms this in that it has that 7-variable model marked in the top row, shows which the 7 variables are by marking them black, and it even shows the minimum BIC value of -770 on the y-axis that can be achieved by any model. Additionally, the second row from the top corresponds to the second best model in terms of BIC – we see from the figure that this is an 8-variable model, containing the 7 variables that were in the best model plus the dummy variable representing level Yes of the qualitative variable US.

Similarly, the other plot created by the other command specifies that the best model according to adjusted R^2 value will be the best 10-variable model, specifies what those 10 variables are, and also shows the maximum possible adjusted R^2 value on the y-axis.

To get the coefficients of the best models, type:

coef(BestSSModel,7)
coef(BestSSModel,10)

Note: These are the actual outcomes of the best subset selection procedure. For example, considering BIC, we have seen from the plot or from the **which.min(BestSSsummary$bic)** command that the best 7-variable model will give the minimum BIC value. Therefore, we can check the coefficients using the above **coef(BestSSModel,7)** command and get that in the best model, we estimate sales in thousand units as Sales = 5.475 + 0.093 x CompPrice + 0.016 x Income + 0.116 x Advertising - 0.095 x Price + 4.836 x ShelveLocGood + 1.952 x ShelveLocMedium - 0.046 x Age.

Note2: The above procedure gave an understanding of best models according to different quality measures, shown the second best, third best, etc. models and how much worse they are than the best one, produced nice figures, etc. However, if we do not want to know all this but just want to quickly determine the coefficients of the best model according to, for example, BIC, we can do that in only two lines:
BestSSModel=regsubsets(Sales~.,data=Carseats,nvmax=11)
coef(BestSSModel, which.min(summary(BestSSModel)$bic))

8. Polynomial regression is quite easy to do in R; we will demonstrate this with the Auto dataset in the (already loaded) ISLR library. We can attach Auto to avoid having to write Auto$ all the time:
attach(Auto)

We can then define polynomial regression models of different degrees:
hpmodel=lm(mpg~horsepower)
summary(hpmodel)

Note: the above step fit a usual simple linear regression model, regressing mpg on horsepower. The next step will define a quadratic model, i.e. a linear model which contains a quadratic term: horsepower*horsepower and a usual linear term: horsepower as predictors. The capital I letter in the command around horsepower^2 is necessary – it tells R that we indeed want to use the usual power operation (like we do in x^2), otherwise it would use the ^ in the formula differently and return some weird output.

hp2model=lm(mpg~I(horsepower^2)+horsepower)
summary(hp2model)

Note: how to understand this summary? It means that miles per gallon can be predicted as mpg = 56.9 + 0.0012 x horsepower x horsepower – 0.466 x horsepower. For example, for a car with 98 horsepower we would predict an mpg of 56.9 + 0.0012 * 98 * 98 -0.466*98, i.e. 22.76 miles per gallon.

hp3model=lm(mpg~I(horsepower^3)+I(horsepower^2)+horsepower)
summary(hp3model)
hp4model=lm(mpg~I(horsepower^4)+I(horsepower^3)+I(horsepower^2)+horsepower)
summary(hp4model)

hp5model=lm(mpg~I(horsepower^5)+I(horsepower^4)+I(horsepower^3)+I(horsepower^2)+horsepower)
summary(hp5model)

Note: In hp5model, we included powers up to degree 5, i.e. we included 5 predictors, namely horsepower, horsepower*horsepower, horsepower*horsepower*horsepower, horsepower*horsepower*horsepower*horsepower, and horsepower*horsepower*horsepower* horsepower*horsepower.

For higher degrees, it is even more convenient to use the poly function:
hp10model=lm(mpg~poly(horsepower, 10))
summary(hp10model)

Note: this model included all degrees of horsepower up to degree 10.

Alternatively, we can start with a linear model:
lin=lm(mpg~horsepower)
summary(lin)
and update it step by step:
quad=update(lin,~.+I(horsepower^2))
summary(quad)
cub=update(quad,~.+I(horsepower^3))
summary(cub)

Note: the meaning of, for example, **cub=update(quad,~.+I(horsepower^3))** is: update the model called quad, take all predictors that it contains, and add the term I(horsepower^3).

We can also create the plots shown during the lecture:
plot(horsepower,mpg)
abline(hpmodel$coefficients[1],hpmodel$coefficients[2],col="orange",lwd=2)
lines(seq(40,250,0.01), predict(hp2model,data.frame(horsepower=seq(40,250,0.01)))
,lwd=2, col="blue")
lines(seq(40,250,0.01), predict(hp5model,data.frame(horsepower=seq(40,250,0.01)))
,lwd=2, col="green")
lines(seq(40,250,0.01), predict(hp10model,data.frame(horsepower=seq(40,250,0.01)))
,lwd=2, col="red")

Note: These commands define a sequence from 40 to 250 by steps of 0.01 where the relevant models (e.g. hp10model in the last line) make predictions for all these values and these predictions are plotted as a red curve having line width 2.

9. If you are done with the previous exercises, do exercises 8 and 10 in Section 2.4 of ISL (pages 54-56). Recall that the book can be downloaded for free at http://faculty.marshall.usc.edu/gareth-james/ISL/index.html. Note that the College dataset is included in the ISLR library, so it does not need to be downloaded from the internet.

The solutions of these two exercises are not provided here, but if you attempt solving the exercises and want to get hints or discuss your solution with me, please feel very welcome to contact me!

10. If you would like to ask something or give feedback about this computer lab, feel free to talk to me or enter your question/feedback at www.menti.com, using the code 35 97 09.