# Day 5: Notes

# To plot a correlation plot:

# install.packages("corrplot")

library(corrplot)

corrplot(cor(boston\_housing))

# correlation can only be calculated between numerical variables

credit\_card\_numeric <- select(credit\_card, Balance, Income, Limit, Rating, Cards, Age, Education )

## To take a numerical variable and turn it into a factor using bins

transport <- cut(boston\_housing$rad, breaks = c(0,15,30), labels=c("low","high")))

boston\_hosting$transport <- transport

#transport is now a factor vector - level labels "low" and "high"

# alternative

boston\_hosting <- transform(boston\_housing, transport = cut(rad, breaks = c(0,15,30), labels=c("low","high")))

## To calculate Rsquared of a fit manually

# method to calculate Rsquared by hand

# should match value from fit summary

model1 <- lm(Balance~Income,data=credit\_card)

credit\_card$Balance\_pred <- predict(model1, newdata = credit\_card)

credit\_card$Balance\_mean <- mean(credit\_card$Balance)

credit\_card$squared\_dev <- (credit\_card$Balance

                            - credit\_card$Balance\_mean)^2

credit\_card$squared\_resid <- (credit\_card$Balance

                            - credit\_card$Balance\_pred)^2

TSS <-  sum(credit\_card$squared\_dev)

RSS <-  sum(credit\_card$squared\_resid)

Rsq <- 1 - RSS/TSS

## To calculate Rsquared of a fit with respect to a test dataset

# method to calculate Rsquare by hand on new data

# fit to "training dataset"

model1 <- lm(Balance~Income,data=credit\_card)

# now make predictions on a "testing dataset" and measure Rsquared

credit\_card\_test$Balance\_pred<-predict(model1, newdata=credit\_card\_test)

credit\_card\_test$Balance\_mean <- mean(credit\_card\_test$Balance)

credit\_card\_test$squared\_dev <- (credit\_card\_test$Balance

                            -credit\_card\_test$Balance\_mean)^2

credit\_card\_test$squared\_resid <- (credit\_card\_test$Balance

                            -credit\_card\_test$Balance\_pred)^2

TSS <-  sum(credit\_card\_testd$squared\_dev)

RSS <-  sum(credit\_card\_test$squared\_resid)

Rsq <- 1 - RSS/TSS

**How to create an empty model / maximal model**

# when working with model selection its useful to know the following

# shortcuts to create an empty / full model

data.original = boston\_housing

# we create an empty model using ~ 1

# equivalent to using the average as

# the prediction

lm.no\_predictors = lm( medv ~ 1,data=data.original)

# we create a model based on all predictors in the dataframe

# using ~ .

# note that the response variable is excluded

# (we can specify individual columns to omit using a minus sign)

lm.full = lm(medv ~ ., data=data.original)

## how to use stepAIC to perform stepwise selection

library(MASS)

data.original = boston\_housing

# to perform stepwise selection

# first we have to specify the scope

# which defines the minimal and maximal models

# we first  define the minimum and maximum model sizes

lm.min <- lm(medv ~ 1, data=data.original)

lm.max <- lm(medv ~ ., data=data.original)

# and put them in a list that can be used as an argument

# when calling the stepAIC function

scp <- list(lower = lm.min, upper = lm.max)

# run 1 step forward

lm.selected <- stepAIC(lm.min,

                       direction = 'forward',

                       scope = scp,

                       steps = 1)

# change steps to change maximum number of steps

# function stops if no improvement is possible

# can change direction to "forward" "backwards" "both"#

# if going backwards start from lm.max not lm.min

# e.g. stepAIC(lm.max, ...

## Use leaps to perform exhaustive testing of every potential subset of predictors

# take care if applying to set of more that ~20 predictors or on large

# dataset

# Using leaps

#install.packages("leaps")

library(leaps)

regsubsets.out <- regsubsets( medv ~ .,

                              data = data.original,

                              nbest = 1,

                              nvmax = NULL,

                              force.in = NULL, force.out = NULL,

                              method = 'exhaustive')

# display best subsets

# (shows best 1-predictor model, best 2-predictor model etc...)

summary(regsubsets.out)

# converts this to dataframe

as.data.frame(summary(regsubsets.out)$outmat)

# create plot of results - this also includes fit performance information

plot(regsubsets.out, scale='adjr2', main='Adjusted Rsq')