# Day 7: Code examples

Link to the code examples created in the AM session

[+Day 7: AM](https://paper.dropbox.com/doc/NVPhNQxXYlch9TTSxvT2X)

Preparing our workspace:

library(readr)

library(dplyr)

library(tidyr)

housing\_data <- read\_csv("train.csv")

housing\_data$logSalePrice <- log10(housing\_data$SalePrice)

housing\_data <- select\_if(housing\_data, is.numeric)

set.seed(1)

n <- nrow(housing\_data)

random\_row\_ids <- sample(1:n,size=n,replace=TRUE)

housing\_data\_rs <- housing\_data[random\_row\_ids,]

mean(housing\_data$logSalePrice)

mean(housing\_data\_rs$logSalePrice)

plot(logSalePrice~GrLivArea,data=housing\_data, pch=1)

points(logSalePrice~GrLivArea,data=housing\_data\_rs, col="red", pch=2)

lm1 <- lm(logSalePrice~GrLivArea, data=housing\_data)

lm2 <- lm(logSalePrice~GrLivArea, data=housing\_data\_rs)

abline(lm1)

abline(lm2, col="red")

gradient1 <- lm1$coefficients[2]

gradient1

gradient2 <- lm2$coefficients[2]

gradient2

R=100

results <- data.frame(i=rep(0,R),

                      gradient = rep(0,R),

                      rmse = rep(0,R))

for (i in 1:R){

  n <- nrow(housing\_data)

  random\_row\_ids <- sample(1:n,size=n,replace=TRUE)

  housing\_data\_rs <- housing\_data[random\_row\_ids,]

  lm2 <- lm(logSalePrice~GrLivArea, data=housing\_data\_rs)

  gradient <- lm2$coefficients[2]

  results$gradient[i] <- gradient

  results$i[i] <- i

  RSS <- sum(lm2$redisuals^2)

  MSE <- RSS/n

  RMSE <- sqrt(MSE)

  # RMSE <- sqrt(sum(lm2$redisuals^2)/n)

  results$rmse[i] <- RMSE

}

Non-linear fit methods

# Use the example code to explore a relationship between

# logSalePrice and one of the four predictors we used

# in the model yesterday

library(ISLR)

library(splines)

data(Wage)

plot(wage~age, data=Wage)

Non-linear fits

bs() ns() s()  are different spline functions that can be used for fitting. Different functions use different mathematical methods to draw curves between the linear fits in each region

# fit using polynomial function 3rd order polynomial

# lm\_poly <- lm(wage~poly(age,3), data=Wage)

# fit using bs spline function 5 degrees of freedom

# lm\_spline <- lm(wage~bs(age,df=5), data=Wage)

# scatter plot + smoothed line

# scatter.smooth(Wage$age,Wage$wage,pch='.')

# make grid of age values for plotting predictions

agelims <- range(Wage$age)

age\_grid <- seq(from=agelims[1],to=agelims[2], length.out=100)

age\_grid\_df <- data.frame(age = age\_grid)

# simple linear fit

fit\_lin <- lm(wage~age, data=Wage)

pred <- predict(fit\_lin,newdata=age\_grid\_df)

plot(wage~age, pch='.', data=Wage)

lines(age\_grid,pred,lwd=2,col="blue")

# fitting using splines at defined knot points

library(splines)

# can specify knot points

fit\_spline1=lm(wage~bs(age,knots=c(25,40,60)),data=Wage)

pred <- predict(fit\_spline1,newdata=age\_grid\_df)

plot(wage~age, pch='.', data=Wage)

lines(age\_grid,pred,lwd=2,col="red")

# can specify degrees of freedom (more=more flexible)

fit\_spline2=lm(wage~bs(age,df=6),data=Wage)

pred <- predict(fit\_spline2,newdata=age\_grid\_df)

plot(wage~age, pch='.', data=Wage)

lines(age\_grid,pred,lwd=2,lty=2,col="red")

# loess to build a LOcal regrESSion

fit\_loess1=loess(wage~age,span=.2,data=Wage)

fit\_loess2=loess(wage~age,span=1,data=Wage)

plot(wage~age, pch='.', data=Wage)

lines(age\_grid,predict(fit\_loess1,newdata=age\_grid\_df),col="red",lwd=2)

lines(age\_grid,predict(fit\_loess2,newdata=age\_grid\_df),col="blue",lwd=2)

# fit with step function

cut\_points <- quantile(Wage$age,c(0,0.25,0.5,0.75,1.0))

Wage$age\_groups <- cut(Wage$age, breaks = cut\_points)

age\_grid\_df <- data.frame(age = age\_grid)q

age\_grid\_df$age\_groups <- cut(age\_grid\_df$age, breaks = cut\_points)

fit\_step <- lm(wage~age\_groups, data=Wage)

pred <- predict(fit\_step,newdata=age\_grid\_df)

plot(wage~age, pch='.', data=Wage)

lines(age\_grid,pred,lwd=2,col="blue")

**Challenge: Use k-fold validation (10 folds) to measure the performance of a spline fit between logSalePrice and GrLivArea (or a numerical predictor of your choice) for df = 4, 6, 8, 10, 12, 14**

library(cvTools)

df\_vals <- c(4,,8,12,16,20,24,28,32,36,40)

n\_vals <- length(df\_vals)

results <- 0

results <- data.frame(df = rep(0,n\_vals),

                      rmse = rep(0,n\_vals),

                      se = rep(0,n\_vals))

                      )

for (i in 1:n\_vals){

  df\_i <- df\_vals[i]

  fit\_spline2=lm(wage~bs(age,df=df\_i),data=Wage)

  cv\_result <- cvFit(fit\_spline2, data = Wage, y = Wage$wage, K = 10, R = 10)

  results$df[i] <- df\_i

  results$rmse[i] <- cv\_result$cv

  results$se[i] <- cv\_result$se

}

head(results)

plot(results$rmse~results$df, pch=1, ylim=c(39.8,40.9))

points(results$df, results$rmse + 1.96\*results$se, pch=2)

points(results$df, results$rmse - 1.96\*results$se, pch=2)

**glmnet Ridge Regression and Lasso method**

# In order to use these methods we need to use a different fitting package

# glmnet requires us to pass arguments

# x: a matrix containing predictor columns

# y: a vector containing the response column

# alpha: the type of penalty or "net" used to constrain fit coefficients

#        alpha = 0 means ridge regression (penalty is summed B\_i^2)

#        alpha = 1 means lasso (penalty is summed |B\_i|)

# glmnet does not like NA values...

# either remove problematic rows/columns

# in this case e.g. MiscFeature=NA means number of misc features = 0

# so we replace the NAs with 0 using the following code:

housing\_data[is.na(housing\_data)] <- 0

library(glmnet)

housing\_data.x <- dplyr::select(housing\_data,  -SalePrice, -logSalePrice)

housing\_data.x <- data.matrix(housing\_data.x)

housing\_data.y <- housing\_data$logSalePrice

# glmnet can do ridge regression alpha=0

#            or lasso regression alpha=1

# we can also use values between 0 and 1

# which uses a hybrid of the two penalty types

# (called elastic net)

fit\_lasso <- glmnet(housing\_data.x, housing\_data.y,

               alpha = 1, lambda = 1)

fit\_ridge <- glmnet(housing\_data.x, housing\_data.y,

               alpha = 0, lambda = 1)

# view fit coefficients

coef(fit\_lasso)

# to predict new points based on model

pred <- predict(fit\_lasso,newx=housing\_data.x)

# if we do not specify lambda glmnet will test out

# a range of penalties and we can view how coefficents

# behave by plotting the result

fit\_lasso <- glmnet(housing\_data.x, housing\_data.y,

               alpha = 1)

plot(fit\_lasso, xvar = "lambda") # lambda is on log-scale

Homework challenge:

Use cross validation to explore how fit performance (MSE) changes as lambda is changed.

Hint: look at function

cv.glmnet()