# Day 7: AM

**Bootstrapping example**

#housing\_data

n <- nrow(housing\_data)

resampled\_rows <- sample(1:n,n,replace=TRUE)

housing\_data\_rs <- housing\_data[resampled\_rows, ]

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

B1\_lm1 = lm1$coefficients[2]

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

B1\_lm2 = lm2$coefficients[2]

plot(SalePrice~GrLivArea, data=housing\_data, pch='.',col='red')

points(SalePrice~GrLivArea, data=housing\_data\_rs, pch='.',col='blue')

set.seed(1)

R = 10

B1\_vals <- rep(0,R)

RMSE\_vals <- rep(0,R)

for (i in 1:R){

  n <- nrow(housing\_data)

  resampled\_rows <- sample(1:n,n,replace=TRUE)

  housing\_data\_rs <- housing\_data[resampled\_rows, ]

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

  B1\_vals[i] <- lm2$coefficients[2]

  RSS <- sum((lm2$residuals)^2)

  MSE <- RSS/n

  RMSE <- sqrt(MSE)

  RMSE\_vals[i] <- sqrt((sum((lm2$residuals)^2))/n)

}

hist(RMSE\_vals)

# mean value calculated by bootstrap

mean(RMSE\_vals)

# standard error calculated by bootstrap

sd(RMSE\_vals)

## 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)

# Use the example code to explore a relationship between

# logSalePrice and one of the four predictors we used

# in the model yesterday

library(ISLR)

data(Wage)

# scatter plot

plot(wage~age, pch='.', 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(splines)

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

fit\_spline2=lm(logSalePrice~bs(GrLivArea,df=6),data=housing\_data)

GrLivArea\_grid <- seq(from=350,to=5500, length.out=100)

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

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

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

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

RMSE <- sqrt((sum((fit\_spline2$residuals)^2))/n)

library(cvTools)

n\_df\_vals = 6

df\_vals <- c(4,6,8,10,12,14)

cv\_results <- rep(0,6)

for (i in 1:6){

  df\_i = df\_vals[i]

  fit\_spline = lm(logSalePrice~bs(GrLivArea,df=df\_i),data=housing\_data)

  cv\_result = cvFit(fit\_spline, data = housing\_data, y = housing\_data$logSalePrice, K = 10)

  cv\_results[i] <- cv\_result$cv

}

 Transformation and normalisation

**Take the 4 numerical values we had yesterday**

Transform - mean 0 standard deviation 1

Perform the linear fit to logSalesPrice using these rescaled variables

lm.selected <- lm(logSalePrice~OverallQual + OverallCond + GrLivArea + YearBuilt, data=house\_data\_num)

OverallQual\_norm <- (housing\_data$OverallQual-mean(housing\_data$OverallQual))/sd(housing\_data$OverallQual)

housing\_data$OverallQual\_norm <- OverallQual\_norm

OverallCond\_norm <- (housing\_data$OverallCond-mean(housing\_data$OverallCond))/sd(housing\_data$OverallCond)

housing\_data$OverallCond\_norm <- OverallCond\_norm

GrLivArea\_norm <- (housing\_data$GrLivArea-mean(housing\_data$GrLivArea))/sd(housing\_data$GrLivArea)

housing\_data$GrLivArea\_norm <- GrLivArea\_norm

YearBuilt\_norm <- (housing\_data$YearBuilt-mean(housing\_data$YearBuilt))/sd(housing\_data$YearBuilt)

housing\_data$YearBuilt\_norm <- YearBuilt\_norm

**Simple Ridge regression example (glmnet model is more flexible)**

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

library(MASS)

# example of ridge regression with lambda = 1

lm\_ridge\_1 <- lm.ridge(logSalePrice~.-SalePrice, lambda = 1, data=housing\_data[])

#penalty is on the sum of the absolute values of

# the coefficients

# we can check this with

sum(abs(lm\_ridge\_1$coef))

**glmnet Ridge Regression / 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()