Assignment2_MODELLING IN PHY GEO

Oyedayo Oyelowo

15 November 2017

Question 1:

Using bootstrap sampling with 1000 repeats, calculate sample mean and associated 95 % confidence intervals for mean air temperature (as in week 1 "day2_exercises.pdf"). Plot the result as a histogram with confidence intervals indicated as dashed lines. Save the figure.

```
data<-
read.csv("C:/Users/oyeda/Desktop/MODELLING_PHYSICAL_GEOGRAPHY/assignment2/Dat
a-20171114/AirTemperatureData.csv", header=T, sep = ";")
#the mean
mean(data$temp)
```

```
## [1] 9.70929
```

Boostrapping.

replicate the sampling of the mean temperature for 999 times

```
bsa<-replicate(999, expr=mean(sample(data$temp, replace = T)))

#combine the above with the original mean
bsa<- c(bsa, mean(data$temp))
confi<- quantile(bsa, probs = c(0.25, 0.975))

#save the plot in the file path
png(filename="C:/Users/oyeda/Desktop/MODELLING_PHYSICAL_GEOGRAPHY/assignment2
/Data-20171114/name1.png")
hist(bsa)
abline(v=confi, lty=2, col="red")
dev.off()
```

```
## png
##   2
```

Here the mean I obtained before the replicated sampling was 9.70929. which still falls within the confidence interval. Thus, I can say with 95% confidence that the mean is in between the red lines.

Question 2:

Model air temperatures using first order polynomial terms of elevation and sea as predictors. By the means of bootstrap-sampling (1000 repeats), quantify the 95 % confidence interval for estimated regression slope for elevation.

- Plot the results as a histogram with confidence intervals indicated as dashed lines. Save the figure. You may use and further modify the function below:

```
#   b <- function(){
#     sam <- sample(nrow(d), replace = TRUE) # draw a bootstrap sample
#     m <- lm(y~x,data=d[sam,]) # fit a linear regression model
#     return(coef(m)) # return estimated coefficients. Use "[]" -syntax to
extract specific model
#     # coefficient

coef_elev <- function(d){
    sam <- sample(nrow(d), replace = TRUE) # draw a bootstrap sample
    lm_temp <- lm(temp~elev+sea,data=d[sam,]) # fit a linear regression model
    summary(lm_temp)
     return(coef(lm_temp)[2]) # return estimated coefficients. Use "[]" -
syntax to extract specific model  coefficient
}

rep_coef_elev<-replicate(1000, coef_elev(d=data))
png(filename="C:/Users/oyeda/Desktop/MODELLING_PHYSICAL_GEOGRAPHY/assignment2
/Data-20171114/hist_elev_coef.png")
hist(rep_coef_elev, xlab = "Elevation Slope")
conf<- quantile(rep_coef_elev, probs = c(0.25,0.975))
abline(v=conf, lty=4, lwd=3, col="red")
dev.off()

## png
##   2
```

Question 3:

- Write a for-loop to perform a leave-one-out cross-validation of a model, where air temperatures are being predicted using elevation (first and second order polynomial terms).
- Create a scatterplot, where observed air temperatures are on y-axis and predicted air temperatures on x-axis. Quantify the agreement between observed and predicted values by the means of mean difference and Pearson's correlation coefficient. Add both measures to the scatterplot.

Tips for LOOCV:

- to define the for-loop, you need to know the number of rows in the data; this can be obtained using a function nrow()

- at each iteration (=loop-round), you need to set aside one row of the whole data for evaluation in turn; other are used for fitting the model
- you need to collect the predicted values of each iteration round to a result vector; before initiating the for-loop, create empty vector for this purpose
- inside the loop use c() -function to collect the predicted values to the result vector

```
{
temp_pred<-c()
for (i in 1:nrow(data)){
  lm_temp2<- lm(temp~elev+I(elev^2), data = data[-i,])
  eval_data<- data[i,]
  p<- predict(lm_temp2, eval_data)
  temp_pred[i]<-(p)

}
#combine the predicted temperature and observed into a dataframe
obs_pred_temp<-cbind.data.frame(temp_pred, data$temp)
colnames(obs_pred_temp)<-c("predicted_temp", "observed_temp")
}
```

creating function to calculate mean error

```
#function to calculate mean error
mean_error<- function(obs, pred, data){
            me<-sum(abs(obs-pred))/nrow(data)
            return(me)
}
```

Calculating mean error

```
me_obs_pred<-mean_error(obs = obs_pred_temp$observed_temp, pred =
obs_pred_temp$predicted_temp, data=obs_pred_temp)
#########################################################################
#this was done to test the function
# obs_pred_temp$test<-abs((obs_pred_temp$predicted_temp)-
(obs_pred_temp$observed_temp))
# mean(obs_pred_temp$test)
#########################################################################
```

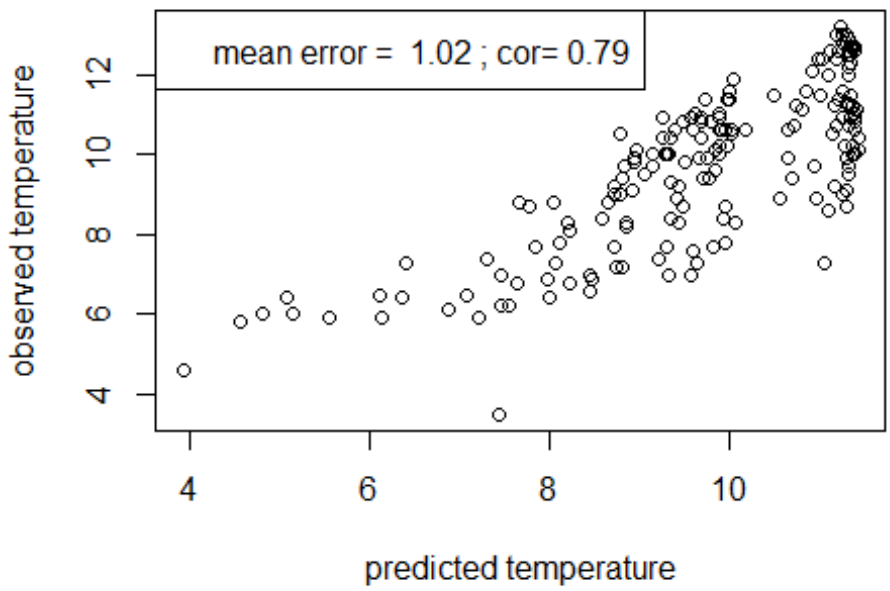
correlation

```
cor_obs_pred<-cor(obs_pred_temp$predicted_temp,obs_pred_temp$observed_temp,
method = "pearson")
```

plotting the observed against the predicted

```
plot(observed_temp~predicted_temp, data=obs_pred_temp,
     xlab="predicted temperature", ylab = "observed temperature")

legend("topleft", paste(paste("mean error = ", round((me_obs_pred),2)),";",
        paste("cor=", round((cor_obs_pred),2))) )
```



Question 4

Does the predictive performance increase after including latitude (y) and longitude (x) to the previous model? Consider their first and second order polynomial terms & and their interaction. Create a scatterplot, where observed air temperatures are on y-axis and predicted air temperatures on x-axis. Quantify the agreement between observed and predicted values by the means of mean difference and Pearson's correlation coefficient. Add both measures to the scatterplot.

```
{
  temp_pred<-c()
  for (i in 1:nrow(data)){
    #print(i)
    lm_temp2<- lm(temp~elev+y+ x+ I(elev^2)+ I(y^2) + I(x^2) + y:x, data =
data[-i,])
    eval_data<- data[i,]
    p<- predict(lm_temp2, eval_data)
    temp_pred[i]<-(p)

  }
  #combine the predicted temperature and observed into a dataframe
  obs_pred_temp<-cbind.data.frame(temp_pred, data$temp)
  colnames(obs_pred_temp)<-c("predicted_temp", "observed_temp")
}
```

Calculating mean error

```
me_obs_pred<-mean_error(obs = obs_pred_temp$observed_temp, pred =
obs_pred_temp$predicted_temp, data=obs_pred_temp)
```

correlation

```
cor_obs_pred<-cor(obs_pred_temp$predicted_temp,obs_pred_temp$observed_temp,
method = "pearson")
```

plotting the observed against the predicted

```
plot(observed_temp~predicted_temp, data=obs_pred_temp,
     xlab="predicted temperature", ylab = "observed temperature")

legend("topleft", paste(paste("mean error = ", round((me_obs_pred),2)),";",
                        paste("cor=", round((cor_obs_pred),2))) )
```