Predicting first-year calculus marks with k nearest neighbour regression[1]

```
> # Heading for prediction of the replication data
>
> setwd("/Users/brunner/Dropbox/2018-19Teaching/2101f18/2101f18Lectures/25-2101f18-
knnMathReg")
> source("readmath1.txt") # Defines data frames datta and training2
Warning messages:
1: NAs introduced by coercion
2: NAs introduced by coercion
3: NAs introduced by coercion
4: NAs introduced by coercion
5: NAs introduced by coercion
6: NAs introduced by coercion
> # source("readmath2.txt") # Defines data frames replic and test2
> # Load knn packages
>
> # install.packages("DMwR") #   For  knnImputation
> library(DMwR)
Loading required package: lattice
Loading required package: grid
> # install.packages("kknn")
> library(kknn)
>
> head(training2)
  id     course precalc calc  gpa calculus english mark    lang    sex
1  1 Mainstream       2    0 78.0       65      80   39 English Female
2  2 Mainstream       6    2 66.0       54      75   57 English Female
3  3 Mainstream       4    4 80.2       77      70   62 English   Male
4  4 Mainstream       5    2 81.7       80      67   76 English Female
5  5 Mainstream       4    4 86.8       87      80   86 English   Male
6  6 Mainstream       3    1 76.7       53      75   60 English   Male
                nation1               nation2 sample                nation diagtest
1 European not Eastern European not Eastern      1 European not Eastern        2
2 European not Eastern European not Eastern      1 European not Eastern        8
3          East Indian         Other and DK      1         Other and DK        8
4     Eastern European     Eastern European      1     Eastern European        7
5          East Indian          East Indian      1          East Indian        8
6 European not Eastern European not Eastern      1 European not Eastern        4

> summary(training2)
       id                course       precalc          calc             gpa
 Min.   :  1.0   Catch-up  : 59   Min.   :0.000   Min.   : 0.000   Min.   :65.00
 1st Qu.:145.5   Elite     : 39   1st Qu.:3.000   1st Qu.: 1.000   1st Qu.:75.20
 Median :290.0   Mainstream:373   Median :4.000   Median : 3.000   Median :78.30
 Mean   :290.0   NA's      :108   Mean   :4.402   Mean   : 3.319   Mean   :79.27
 3rd Qu.:434.5                    3rd Qu.:6.000   3rd Qu.: 5.000   3rd Qu.:82.50
 Max.   :579.0                    Max.   :9.000   Max.   :11.000   Max.   :97.30
                                  NA's   :99      NA's   :99       NA's   :113
    calculus         english           mark          lang         sex
 Min.   : 50.00   Min.   :50.00   Min.   : 1.00   English:402   Female:266
 1st Qu.: 67.00   1st Qu.:71.00   1st Qu.:50.00   Other  :149   Male  :285
 Median : 76.00   Median :76.00   Median :60.00   NA's   : 28   NA's  : 28
 Mean   : 75.44   Mean   :75.84   Mean   :58.74
 3rd Qu.: 85.00   3rd Qu.:82.00   3rd Qu.:72.00
 Max.   :100.00   Max.   :96.00   Max.   :99.00
 NA's   :131      NA's   :99      NA's   :186
```

1 This is an open-source document. Copyright information is on the last page.

```
                          nation1                              nation2        sample
 Asian                        :126   Asian                        :131   Min.   :1
 Eastern European             : 61   Eastern European             : 65   1st Qu.:1
 European not Eastern         :204   European not Eastern         :196   Median :1
 Middle-Eastern and Pakistani: 66   Middle-Eastern and Pakistani: 59   Mean   :1
 East Indian                  : 72   East Indian                  : 86   3rd Qu.:1
 Other and DK                 : 50   Other and DK                 : 42   Max.   :1

                         nation      diagtest
 Asian                       :131   Min.   : 0.000
 Eastern European            : 63   1st Qu.: 5.000
 European not Eastern        :195   Median : 7.000
 Middle-Eastern and Pakistani: 72   Mean   : 7.721
 East Indian                 : 78   3rd Qu.:10.000
 Other and DK                : 40   Max.   :20.000
                                    NA's   :99
> is.numeric(training2$nation2) # Just checking
[1] FALSE

>
> explanvars = training2[,c(2:7,9:12,14,15)]# Omitting id, mark,sample)
> summary(explanvars)
       course       precalc          calc             gpa           calculus
 Catch-up  : 59   Min.   :0.000   Min.   : 0.000   Min.   :65.00   Min.   : 50.00
 Elite     : 39   1st Qu.:3.000   1st Qu.: 1.000   1st Qu.:75.20   1st Qu.: 67.00
 Mainstream:373   Median :4.000   Median : 3.000   Median :78.30   Median : 76.00
 NA's      :108   Mean   :4.402   Mean   : 3.319   Mean   :79.27   Mean   : 75.44
                  3rd Qu.:6.000   3rd Qu.: 5.000   3rd Qu.:82.50   3rd Qu.: 85.00
                  Max.   :9.000   Max.   :11.000   Max.   :97.30   Max.   :100.00
                  NA's   :99      NA's   :99       NA's   :113     NA's   :131
   english          lang         sex                                nation1
 Min.   :50.00   English:402   Female:266   Asian                        :126
 1st Qu.:71.00   Other  :149   Male  :285   Eastern European             : 61
 Median :76.00   NA's   : 28   NA's  : 28   European not Eastern         :204
 Mean   :75.84                              Middle-Eastern and Pakistani: 66
 3rd Qu.:82.00                              East Indian                  : 72
 Max.   :96.00                              Other and DK                 : 50
 NA's   :99
                      nation2                              nation      diagtest
 Asian                    :131   Asian                        :131   Min.   : 0.000
 Eastern European         : 65   Eastern European             : 63   1st Qu.: 5.000
 European not Eastern     :196   European not Eastern         :195   Median : 7.000
 Middle-Eastern and Pakistani: 59   Middle-Eastern and Pakistani: 72   Mean   : 7.721
 East Indian              : 86   East Indian                  : 78   3rd Qu.:10.000
 Other and DK             : 42   Other and DK                 : 40   Max.   :20.000
                                                                     NA's   :99
> exvars = knnImputation(explanvars)
> summary(exvars)
       course       precalc          calc             gpa           calculus
 Catch-up  :100   Min.   :0.000   Min.   : 0.000   Min.   :65.00   Min.   : 50.00
 Elite     : 93   1st Qu.:3.000   1st Qu.: 2.000   1st Qu.:75.80   1st Qu.: 67.92
 Mainstream:386   Median :4.284   Median : 3.000   Median :78.30   Median : 74.77
                  Mean   :4.399   Mean   : 3.366   Mean   :79.14   Mean   : 74.81
                  3rd Qu.:5.000   3rd Qu.: 4.971   3rd Qu.:81.90   3rd Qu.: 82.00
                  Max.   :9.000   Max.   :11.000   Max.   :97.30   Max.   :100.00
   english          lang         sex                                nation1
 Min.   :50.00   English:429   Female:270   Asian                        :126
 1st Qu.:71.24   Other  :150   Male  :309   Eastern European             : 61
 Median :76.00                              European not Eastern         :204
 Mean   :75.83                              Middle-Eastern and Pakistani: 66
 3rd Qu.:81.00                              East Indian                  : 72
 Max.   :96.00                              Other and DK                 : 50
```

```
                          nation2                                 nation       diagtest
 Asian                        :131   Asian                        :131   Min.   : 0.000
 Eastern European             : 65   Eastern European             : 63   1st Qu.: 5.075
 European not Eastern         :196   European not Eastern         :195   Median : 7.422
 Middle-Eastern and Pakistani: 59   Middle-Eastern and Pakistani: 72   Mean   : 7.765
 East Indian                  : 86   East Indian                  : 78   3rd Qu.:10.000
 Other and DK                 : 42   Other and DK                 : 40   Max.   :20.000

> dim(exvars) # n = 579, no missing.
[1] 579  12
>
> mark = training2$mark
> imputed1 = cbind(mark,exvars); # is.data.frame(imputed1)
>
> # Staying within the exploratory sample, randomly divide the data into a training
subset and a validation subset.
>
> nvalid = 100 # Number of observations in the validation sample(s).
> n = dim(imputed1)[1]; n
[1] 579
>
> set.seed(9999)
> val = sample(1:n,size=nvalid) # Select lines for the validation sample
> mathlearn = imputed1[-val,] # Imputed data set without the validation cases
> mathvalid = imputed1[val,]  # Just the validation cases
> # All the potential predictors, with defaults
> kknnout1 <- kknn(mark ~ ., train=mathlearn, test=mathvalid)
> y = mark[val] # The numbers we are trying to predict.
> yhat = kknnout1$fitted.values
> cbind(y,yhat)[1:20,] # Look at first 20 rows
       y     yhat
 [1,] NA 62.90402
 [2,] 50 56.27338
 [3,] 30 36.20508
 [4,] NA 33.42653
 [5,] 50 62.51140
 [6,] 50 63.47820
 [7,] NA 63.68476
 [8,] 73 60.95401
 [9,] 80 56.70517
[10,] NA 30.09953
[11,] NA 51.46786
[12,] NA 52.91483
[13,] NA 59.38858
[14,] NA 43.87111
[15,] 54 46.78625
[16,] 62 49.26554
[17,] 80 54.21268
[18,] 53 68.36073
[19,] 75 51.51637
[20,] 71 64.17174
>
> cor(y,yhat, use="complete.obs")^2 # R-squared
[1] 0.1455476

> mean(abs(y-yhat), na.rm = TRUE)
[1] 14.30286
>
> # For least-squares regression, I got r^2 = 0.339 and mean abs error = 11.43 when
predicting replication data from exploratory data. Try tuning the knn prediction
model to minimize mean absolute error.
```

```
>
> # Variable selection
>
> # For this phase I will stay with the knn default values of k=10 for the
imputation and k=7 for the prediction.
>
> # Also, randomly select a validation sub-sample several times, and take the
average mean absolute error.
> # Do the full model this way for comparison.
>
> set.seed(9999)
>
> ntries = 100
> mae = numeric(ntries) # Will hold Mean Absolute Errors
> for(j in 1:ntries)
+ {
+ val = sample(1:n,size=nvalid) # Select lines for the validation sample
+ mathlearn = imputed1[-val,] # Imputed data set without the validation cases
+ mathvalid = imputed1[val,]  # Just the validation cases
+ # All the potential predictors, with defaults
+ kknnout1 <- kknn(mark ~ ., train=mathlearn, test=mathvalid)
+ y = mark[val] # The numbers we are trying to predict.
+ yhat = kknnout1$fitted.values
+ mae[j] = mean(abs(y-yhat), na.rm = TRUE)
+ } # next try
> sort(mae)
  [1]  9.255565  9.381547 10.583116 11.079411 11.201669 11.528330 11.714042 11.799500
  [9] 11.801646 11.876463 11.897931 11.912993 12.035962 12.048219 12.269701 12.320594
 [17] 12.333565 12.398875 12.401848 12.444073 12.455209 12.458527 12.462479 12.485699
 [25] 12.548694 12.570193 12.646148 12.650095 12.659063 12.663603 12.674207 12.680255
 [33] 12.687385 12.692069 12.726197 12.747187 12.760015 12.777094 12.783345 12.791656
 [41] 12.793750 12.825503 12.868223 12.972738 12.995085 13.032895 13.083878 13.155629
 [49] 13.217816 13.229993 13.259987 13.287804 13.311604 13.323955 13.364128 13.373639
 [57] 13.417954 13.436781 13.479253 13.498243 13.511870 13.541573 13.644931 13.683436
 [65] 13.697995 13.746954 13.755545 13.758785 13.793142 13.816336 13.846637 13.877155
 [73] 13.909107 13.939362 13.961144 14.059501 14.078641 14.093528 14.192152 14.236403
 [81] 14.250419 14.256342 14.298168 14.302049 14.302863 14.338504 14.424860 14.442856
 [89] 14.444254 14.456893 14.521771 14.607375 14.722588 14.818939 14.853476 14.992327
 [97] 15.033273 15.783372 15.931602 16.208923

> mean(mae)
[1] 13.21268
>
> # Forward selection. The Step 0 is to predict using the mean of the training
sample (median would be okay too).
>
> set.seed(9999)
>
> ntries = 100
> mae = numeric(ntries) # Will hold Mean Absolute Errors
> for(j in 1:ntries)
+ {
+ val = sample(1:n,size=nvalid) # Select lines for the validation sample
+ mathlearn = imputed1[-val,] # Imputed data set without the validation cases
+ mathvalid = imputed1[val,]  # Just the validation cases
+ y = mark[val] # The numbers we are trying to predict.
+ yhat = mean(mathlearn$mark,na.rm = TRUE)
+ mae[j] = mean(abs(y-yhat), na.rm = TRUE)
+ } # next try
> # sort(mae)
> mean(mae) #
[1] 14.61009
```

```
>
> # Now need explanatory variable names
> evname = colnames(exvars); nevs = length(evname); evno = 1:nevs
> data.frame(evname)
     evname
1    course
2   precalc
3      calc
4       gpa
5  calculus
6   english
7      lang
8       sex
9   nation1
10  nation2
11   nation
12 diagtest

>
> # Step 1: 1-variable knn prediction
>
> ntries = 100
> mae = numeric(ntries) # Will hold Mean Absolute Errors
> for(v in evno)
+ {
+ call = paste("mark ~ ",evname[v], sep = ' ') # Use expl var v
+ for(j in 1:ntries)
+     {
+     val = sample(1:n,size=nvalid) # Select lines for the validation sample
+     mathlearn = imputed1[-val,] # Imputed data set without the validation cases
+     mathvalid = imputed1[val,]  # Just the validation cases
+     kknnout <- kknn(call, train=mathlearn, test=mathvalid)
+     y = mark[val] # The numbers we are trying to predict.
+     yhat = kknnout$fitted.values
+     mae[j] = mean(abs(y-yhat), na.rm = TRUE)
+     } # Next random validation sample
+ cat(v,call, '      Mean absolute error = ',mean(mae), '\n')
+ } # Next explanatory variable
1 mark ~  course       Mean absolute error =  15.34691
2 mark ~  precalc       Mean absolute error =  15.13105
3 mark ~  calc       Mean absolute error =  15.69824
4 mark ~  gpa       Mean absolute error =  13.8262
5 mark ~  calculus       Mean absolute error =  14.05751
6 mark ~  english       Mean absolute error =  16.20216
7 mark ~  lang       Mean absolute error =  15.3083
8 mark ~  sex       Mean absolute error =  15.50992
9 mark ~  nation1       Mean absolute error =  15.28292
10 mark ~  nation2       Mean absolute error =  15.97894
11 mark ~  nation       Mean absolute error =  15.83062
12 mark ~  diagtest       Mean absolute error =  14.81018

> # Next time find the best one automatically
> # gpa is number 4

> oldcall = paste("mark ~ ",evname[4], sep = ' ')
> bestcall = paste("mark ~ ",evname[4], sep = ' ')
> bestmae = 13.59605
> bestno = 4
> evname = evname[-bestno] # Remove it from the list
> nevs = length(evname); evno = 1:nevs
```

```
> data.frame(evname)
     evname
1    course
2   precalc
3      calc
4  calculus
5   english
6      lang
7       sex
8   nation1
9   nation2
10   nation
11 diagtest
>
> # Step 2: 2-variable knn prediction
>
> ntries = 100
> mae = numeric(ntries) # Will hold Mean Absolute Errors
> better = FALSE
> for(v in evno)
+ {
+ call = paste(oldcall, '+', evname[v], sep = ' ') # Use expl var v
+ for(j in 1:ntries)
+     {
+     val = sample(1:n,size=nvalid) # Select lines for the validation sample
+     mathlearn = imputed1[-val,] # Imputed data set without the validation cases
+     mathvalid = imputed1[val,]  # Just the validation cases
+     kknnout <- kknn(call, train=mathlearn, test=mathvalid)
+     y = mark[val] # The numbers we are trying to predict.
+     yhat = kknnout$fitted.values
+     mae[j] = mean(abs(y-yhat), na.rm = TRUE)
+     } # Next random validation sample
+ cat(call, '      Mean absolute error = ',mean(mae), '\n')
+ if(mean(mae) < bestmae)
+     {
+     better = TRUE # There was improvement
+     bestmae = mean(mae)
+     bestcall = call
+     bestno = v
+     } # End if there's a new winner
+ } # Next explanatory variable
mark ~  gpa + course       Mean absolute error =  14.24707
mark ~  gpa + precalc       Mean absolute error =  13.22014
mark ~  gpa + calc       Mean absolute error =  13.31644
mark ~  gpa + calculus       Mean absolute error =  13.05677
mark ~  gpa + english       Mean absolute error =  13.40953
mark ~  gpa + lang       Mean absolute error =  13.72525
mark ~  gpa + sex       Mean absolute error =  14.28381
mark ~  gpa + nation1       Mean absolute error =  14.13402
mark ~  gpa + nation2       Mean absolute error =  13.95634
mark ~  gpa + nation       Mean absolute error =  13.88885
mark ~  gpa + diagtest       Mean absolute error =  12.93901

> cat("\n\nImprovement?",better, "  Best so far is ",bestcall, ", yielding mae
=",bestmae, "\n\n")


Improvement? TRUE   Best so far is  mark ~  gpa + diagtest , yielding mae =
12.93901
```

```
>
>
> # Step 3: 3-variable knn prediction
>
> # Re-initialize
> oldcall = bestcall
> evname = evname[-bestno] # Remove it from the list
> nevs = length(evname); evno = 1:nevs
>
> ntries = 100
> mae = numeric(ntries) # Will hold Mean Absolute Errors
> better = FALSE
> for(v in evno)
+ {
+ call = paste(oldcall, '+', evname[v], sep = ' ') # Use expl var v
+ for(j in 1:ntries)
+     {
+     val = sample(1:n,size=nvalid) # Select lines for the validation sample
+     mathlearn = imputed1[-val,] # Imputed data set without the validation cases
+     mathvalid = imputed1[val,]  # Just the validation cases
+     kknnout <- kknn(call, train=mathlearn, test=mathvalid)
+     y = mark[val] # The numbers we are trying to predict.
+     yhat = kknnout$fitted.values
+     mae[j] = mean(abs(y-yhat), na.rm = TRUE)
+     } # Next random validation sample
+ cat(call, '      Mean absolute error = ',mean(mae), '\n')
+ if(mean(mae) < bestmae)
+     {
+     better = TRUE # There was improvement
+     bestmae = mean(mae)
+     bestcall = call
+     bestno = v
+     } # End if there's a new winner
+ } # Next explanatory variable
mark ~  gpa + diagtest + course       Mean absolute error =  13.13778
mark ~  gpa + diagtest + precalc       Mean absolute error =  12.89075
mark ~  gpa + diagtest + calc       Mean absolute error =  12.56295
mark ~  gpa + diagtest + calculus       Mean absolute error =  12.66313
mark ~  gpa + diagtest + english       Mean absolute error =  13.08659
mark ~  gpa + diagtest + lang       Mean absolute error =  12.81824
mark ~  gpa + diagtest + sex       Mean absolute error =  12.97395
mark ~  gpa + diagtest + nation1       Mean absolute error =  13.3226
mark ~  gpa + diagtest + nation2       Mean absolute error =  13.38946
mark ~  gpa + diagtest + nation       Mean absolute error =  13.15419
> cat("\n\nImprovement?",better, "  Best so far is ",bestcall, ", yielding mae
=",bestmae, "\n\n")


Improvement? TRUE   Best so far is  mark ~  gpa + diagtest + calc , yielding mae =
12.56295
```

```
>
> # Step 4: 4-variable knn prediction
>
> # Re-initialize
> oldcall = bestcall
> evname = evname[-bestno] # Remove it from the list
> nevs = length(evname); evno = 1:nevs

>
> ntries = 100
> mae = numeric(ntries) # Will hold Mean Absolute Errors
> better = FALSE
> for(v in evno)
+ {
+ call = paste(oldcall, '+', evname[v], sep = ' ') # Use expl var v
+ for(j in 1:ntries)
+     {
+     val = sample(1:n,size=nvalid) # Select lines for the validation sample
+     mathlearn = imputed1[-val,] # Imputed data set without the validation cases
+     mathvalid = imputed1[val,]  # Just the validation cases
+     kknnout <- kknn(call, train=mathlearn, test=mathvalid)
+     y = mark[val] # The numbers we are trying to predict.
+     yhat = kknnout$fitted.values
+     mae[j] = mean(abs(y-yhat), na.rm = TRUE)
+     } # Next random validation sample
+ cat(call, '      Mean absolute error = ',mean(mae), '\n')
+ if(mean(mae) < bestmae)
+     {
+     better = TRUE # There was improvement
+     bestmae = mean(mae)
+     bestcall = call
+     bestno = v
+     } # End if there's a new winner
+ } # Next explanatory variable
mark ~  gpa + diagtest + calc + course       Mean absolute error =  12.86674
mark ~  gpa + diagtest + calc + precalc       Mean absolute error =  12.72714
mark ~  gpa + diagtest + calc + calculus       Mean absolute error =  12.69758
mark ~  gpa + diagtest + calc + english       Mean absolute error =  12.99643
mark ~  gpa + diagtest + calc + lang       Mean absolute error =  12.76977
mark ~  gpa + diagtest + calc + sex       Mean absolute error =  12.84169
mark ~  gpa + diagtest + calc + nation1       Mean absolute error =  13.23459
mark ~  gpa + diagtest + calc + nation2       Mean absolute error =  13.04671
mark ~  gpa + diagtest + calc + nation       Mean absolute error =  13.11949
> cat("\n\nImprovement?",better, "  Best so far is ",bestcall, ", yielding mae
=",bestmae, "\n\n")


Improvement? FALSE   Best so far is  mark ~  gpa + diagtest + calc , yielding mae =
12.56295

> # That is okay. Let's stop. Model is mark ~  gpa + diagtest + calc
> # Regression model was mark ~ gpa + diagtest + calculus + english + lang for mae
= 11.43
> # Let's try the variables from the regression model with knn
```

```
> # Let's try the variables from the regression model with knn
>
> call = 'mark ~ gpa + diagtest + calculus + english + lang'
> for(j in 1:ntries)
+     {
+     val = sample(1:n,size=nvalid) # Select lines for the validation sample
+     mathlearn = imputed1[-val,] # Imputed data set without the validation cases
+     mathvalid = imputed1[val,]  # Just the validation cases
+     kknnout <- kknn(call, train=mathlearn, test=mathvalid)
+     y = mark[val] # The numbers we are trying to predict.
+     yhat = kknnout$fitted.values
+     mae[j] = mean(abs(y-yhat), na.rm = TRUE)
+     } # Next random validation sample
> cat(call, '      Mean absolute error = ',mean(mae), '\n')
mark ~ gpa + diagtest + calculus + english + lang       Mean absolute error =
12.70284
```

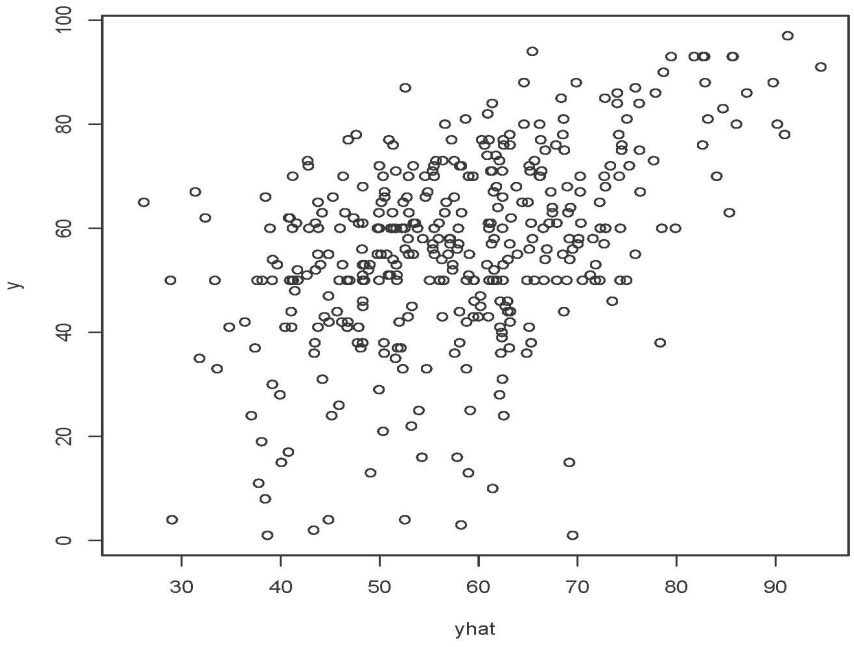
```
>
> # So let's keep mark ~  gpa + diagtest + calc and try to optimize over k.
> call = 'mark ~  gpa + diagtest + calc'
> # Examine k for the prediction
>
> cat("\nLooping over K for prediction: \n\n")

Looping over K for prediction:

> for(K in 1:30)
+ {
+ for(j in 1:ntries)
+     {
+     val = sample(1:n,size=nvalid) # Select lines for the validation sample
+     mathlearn = imputed1[-val,] # Imputed data set without the validation cases
+     mathvalid = imputed1[val,]  # Just the validation cases
+     kknnout <- kknn(call, train=mathlearn, test=mathvalid, k=K)
+     y = mark[val] # The numbers we are trying to predict.
+     yhat = kknnout$fitted.values
+     mae[j] = mean(abs(y-yhat), na.rm = TRUE)
+     } # Next random validation sample
+ cat('k =',K,':','Mean absolute error = ',mean(mae), '\n')
+ } # Next K

k = 1 : Mean absolute error =  16.31598
k = 2 : Mean absolute error =  14.9671
k = 3 : Mean absolute error =  13.82439
k = 4 : Mean absolute error =  13.23168
k = 5 : Mean absolute error =  13.09073
k = 6 : Mean absolute error =  12.93962
k = 7 : Mean absolute error =  12.66451
k = 8 : Mean absolute error =  12.67249
k = 9 : Mean absolute error =  12.62788
k = 10 : Mean absolute error =  12.40136
k = 11 : Mean absolute error =  12.35727
k = 12 : Mean absolute error =  12.31339
k = 13 : Mean absolute error =  12.37528
k = 14 : Mean absolute error =  12.11125
k = 15 : Mean absolute error =  12.36065
k = 16 : Mean absolute error =  12.30189
k = 17 : Mean absolute error =  12.40247
k = 18 : Mean absolute error =  12.34466
k = 19 : Mean absolute error =  12.25026
k = 20 : Mean absolute error =  12.05031
k = 21 : Mean absolute error =  12.0884
k = 22 : Mean absolute error =  12.04424
k = 23 : Mean absolute error =  12.08723
k = 24 : Mean absolute error =  12.15132
k = 25 : Mean absolute error =  11.87051
k = 26 : Mean absolute error =  11.95292
k = 27 : Mean absolute error =  12.02884
k = 28 : Mean absolute error =  12.15453
k = 29 : Mean absolute error =  12.0348
k = 30 : Mean absolute error =  12.08185

> # We will go with k=25 even though the seemingly optimal k will change a bit by
random error.
# It's definitely bigger than the default k=7
```

```
> # Examine k for the imputation
>
> cat("\nLooping over K for imputation: \n\n")

Looping over K for imputation:

> for(K in 1:20)
+ {
+ exvars = knnImputation(explanvars,k=K)
+ imputed1 = cbind(mark,exvars)
+ for(j in 1:ntries)
+     {
+     val = sample(1:n,size=nvalid) # Select lines for the validation sample
+     mathlearn = imputed1[-val,] # Imputed data set without the validation cases
+     mathvalid = imputed1[val,]  # Just the validation cases
+     kknnout <- kknn(call, train=mathlearn, test=mathvalid, k=25)
+     y = mark[val] # The numbers we are trying to predict.
+     yhat = kknnout$fitted.values
+     mae[j] = mean(abs(y-yhat), na.rm = TRUE)
+     } # Next random validation sample
+ cat('k =',K,':','Mean absolute error = ',mean(mae), '\n')
+ } # Next K
k = 1 : Mean absolute error =  12.11717
k = 2 : Mean absolute error =  12.05429
k = 3 : Mean absolute error =  12.09187
k = 4 : Mean absolute error =  12.16548
k = 5 : Mean absolute error =  12.16723
k = 6 : Mean absolute error =  12.00279
k = 7 : Mean absolute error =  12.21391
k = 8 : Mean absolute error =  12.08391
k = 9 : Mean absolute error =  12.0734
k = 10 : Mean absolute error =  11.93713
k = 11 : Mean absolute error =  12.06125
k = 12 : Mean absolute error =  12.01128
k = 13 : Mean absolute error =  12.10128
k = 14 : Mean absolute error =  12.04733
k = 15 : Mean absolute error =  12.12593
k = 16 : Mean absolute error =  12.09096
k = 17 : Mean absolute error =  12.11413
k = 18 : Mean absolute error =  12.26911
k = 19 : Mean absolute error =  12.1884
k = 20 : Mean absolute error =  12.13171
```

```
> # Try median instead of weighted average
> cat("\nLooping over K for imputation: \n\n")

Looping over K for imputation:

> for(K in 1:20)
+ {
+ exvars = knnImputation(explanvars,k=K,meth='median')
+ imputed1 = cbind(mark,exvars)
+ for(j in 1:ntries)
+     {
+     val = sample(1:n,size=nvalid) # Select lines for the validation sample
+     mathlearn = imputed1[-val,] # Imputed data set without the validation cases
+     mathvalid = imputed1[val,]  # Just the validation cases
+     kknnout <- kknn(call, train=mathlearn, test=mathvalid, k=25)
+     y = mark[val] # The numbers we are trying to predict.
+     yhat = kknnout$fitted.values
+     mae[j] = mean(abs(y-yhat), na.rm = TRUE)
+     } # Next random validation sample
+ cat('k =',K,':','Mean absolute error = ',mean(mae), '\n')
+ } # Next K
k = 1 : Mean absolute error =  12.3207
k = 2 : Mean absolute error =  12.0784
k = 3 : Mean absolute error =  12.28319
k = 4 : Mean absolute error =  12.12815
k = 5 : Mean absolute error =  12.28349
k = 6 : Mean absolute error =  12.06268
k = 7 : Mean absolute error =  12.08708
k = 8 : Mean absolute error =  11.97813
k = 9 : Mean absolute error =  11.86776
k = 10 : Mean absolute error =  12.01476
k = 11 : Mean absolute error =  11.96726
k = 12 : Mean absolute error =  12.0112
k = 13 : Mean absolute error =  12.26049
k = 14 : Mean absolute error =  12.00439
k = 15 : Mean absolute error =  11.99715
k = 16 : Mean absolute error =  12.15004
k = 17 : Mean absolute error =  11.86476
k = 18 : Mean absolute error =  12.00897
k = 19 : Mean absolute error =  12.09783
k = 20 : Mean absolute error =  12.06179
>
> # To me, k for imputation does not seem to matter much, so stay with the default
> # k=10
```

```
> # Now we will try predicting the replication data. Again, for least-squares
regression, I got r^2 = 0.339 and mean abs error = 11.43. Re-start R.
>
> setwd("/Users/brunner/Dropbox/2018-19Teaching/2101f18/2101f18Lectures/25-2101f18-
knnMathReg")
> source("readmath1.txt") # Defines data frames datta and training2
Warning messages:
1: NAs introduced by coercion
2: NAs introduced by coercion
3: NAs introduced by coercion
4: NAs introduced by coercion
5: NAs introduced by coercion
6: NAs introduced by coercion
> source("readmath2.txt") # Defines data frames replic and test2
The following objects are masked from training:

    calc, calculus, course, english, gpa, id, lang, mark, nation1, nation2,
    precalc, sample, sex

Warning messages:
1: NAs introduced by coercion
2: NAs introduced by coercion
3: NAs introduced by coercion
4: NAs introduced by coercion
5: NAs introduced by coercion
6: NAs introduced by coercion
>
> # Load knn packages
> # install.packages("DMwR") #   For  knnImputation
> library(DMwR)
Loading required package: lattice
Loading required package: grid
> # install.packages("kknn")
> library(kknn)
>
> # Training (exploratory) data: The complete set
> explanvars1 = training2[,c(2:7,9:12,14,15)]# Omitting id, mark,sample)
> exvars1 = knnImputation(explanvars1)
> dim(exvars1) # n = 579, no missing.
[1] 579  12
> mark = training2$mark
> imputed1 = cbind(mark,exvars1)
>
> explanvars2 = test2[,c(2:7,9:12,14,15)]# Omitting id, mark,sample)
> exvars2 = knnImputation(explanvars2)
> dim(exvars2) # n = 579, no missing.
[1] 579  12
> mark = test2$mark
> imputed2 = cbind(mark,exvars2)
> rm(mark) # Just to be safe
```

```
> kknnout <- kknn(mark ~  gpa + diagtest + calc, train=imputed1, test=imputed2)
> y = imputed2$mark # The numbers we are trying to predict.
> yhat = kknnout$fitted.values
> cbind(y,yhat)[1:20,] # Look at first 20 rows
       y     yhat
 [1,] 56 66.89795
 [2,] 97 91.24131
 [3,] NA 47.64181
 [4,] NA 43.20700
 [5,] 50 62.46234
 [6,] 67 50.48620
 [7,] 70 84.06025
 [8,] 50 70.49123
 [9,]  8 38.43242
[10,] 44 63.21172
[11,] 60 49.96654
[12,] NA 50.34312
[13,] NA 60.56075
[14,] 45 53.25641
[15,] 50 64.88214
[16,] 66 57.53618
[17,] NA 66.02812
[18,] NA 53.00130
[19,] NA 61.68852
[20,] 41 41.10245
>
> cor(y,yhat, use="complete.obs")^2 # R-squared
[1] 0.2203939
> plot(yhat,y)
```




```
> mean(abs(y-yhat), na.rm = TRUE)
[1] 12.93827
>
> # For least-squares regression, I got r^2 = 0.339 and mean abs error = 11.43 when
predicting replication data from exploratory data.
```