VO2016

Virtual Observations 2016

WERKCOLLEGE 4: STATISTIC IN DATA MINING

Document identifier:	**VO2016-W4-01**
Date:	**May 5, 2016**
Activity:	
Document status:	
Document link:	

Abstract: This werkcollege gives examples of catalog classification, work with R and MySQL.

1. USING R

We need to use R as our main statistical programming language and as well as a software for MySQL data.

1.1. FIRST STEPS

For the first exercize retrieve data from Vizier (2MASS) for 1 deg area in csv format, file `twomass.csv`.

1. start R:

```
R version 2.9.0 (2009-04-17)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

2. load data from the file

```
> t = read.csv('twomass.csv', header=TRUE,sep="|")
```

Note, that `sep` is used only if you have non-standart separator. The file should be adjusted (all leading strings with format should be removed, but leave a string with headers).

```
_r|_RAJ2000|_DEJ2000|RAJ2000|DEJ2000|2MASS|Jmag|e_Jmag|Hmag|e_Hmag|Kmag|e_Kmag|Qflg|Rflg|Bflg|Cflg|Xflg|
0.003535|020.000195|+29.996469|020.000195|+29.996469|01200004+2959472 |15.650| 0.068|15.295| 0.120|15.17
```

t is what is called a data frame. A data frame is pretty much like a table with named columns and the names of the columns can be printed out with the names function.

```
> names(t)
 [1] "X_r"       "X_RAJ2000" "X_DEJ2000" "RAJ2000"   "DEJ2000"   "X2MASS"
 [7] "Jmag"      "e_Jmag"    "Hmag"      "e_Hmag"    "Kmag"      "e_Kmag"
[13] "Qflg"      "Rflg"      "Bflg"      "Cflg"      "Xflg"      "Aflg"
```

3. The next is to calculate some of the statistical quantities we saw in the lecture and to do that we need to access the individual columns of t:

```
> mean(t$RAJ2000)
[1] 19.99512
> median(t$RAJ2000)
[1] 19.98377
> sd(t$RAJ2000)
[1] 0.5732393
```

This will go wrong sometimes:

```
> mean(t$e_Jmag)
[1] NA
```

- there are NA (not available) values in this column. It is possible to remove them:

```
> mean(t$e_Jmag,na.rm=TRUE)
[1] 0.07895541
```

NB: data frames are usiful analog oftables but sometimes you need just arrays:

```
>x<-array(1:20,dim=c(1,20))
```

array of 20 elements of $1x20$ dimensions.

```
>x<-c(1,2,3,4,5,6,7,8,9,10)
```

vector of 10 elements

```
>df=data.frame(x,0.5*x)
```

and a new data frame.

1.2. VISUALIZATION

Simple plots:

```
>plot(t$RAJ2000,t$DEJ2000)
>plot(t$RAJ2000,t$DEJ2000,xlab="RA, deg",ylab="DEC, deg")
>plot(t$RAJ2000,t$DEJ2000,main="2MASS",xlab="RA, deg",ylab="DEC, deg")
>plot(t$RAJ2000,t$DEJ2000,main="2MASS",xlab="RA, deg",ylab="DEC, deg",pch=20,col="red")
```

Additional abilities come with the lattice library

```
>library(lattice)
>xyplot(t$Jmag ~ t$Jmag-t$Hmag)
>xyplot(t$Jmag ~ t$Jmag-t$Hmag)
>xyplot(t$Jmag ~ t$Jmag-t$Hmag, xlim=c(-2,4), ylim=c(18,5), xlab="J-H",ylab="J")
>histogram(t$Jmag)
>densityplot(t$Jmag)
```

Now let us have some fun:

```
xyplot(t$Jmag ~ t$Jmag-t$Hmag | Qflg, data=t)
```

And now seriosly:

```
>xyplot(t$Jmag[t$Qflg=='AAA'] ~ t$Jmag[t$Qflg=='AAA']-t$Hmag[t$Qflg=='AAA'],
               xlim=c(-2,4), ylim=c(18,5), xlab="J-H",ylab="J")
```

Compare this plot to the previous CMD, without selection.

1.3. TASK - [FE/H] DISTRIBUTION

1. Repeat step 2 from Werkcollege 3, but this time select all objects with there galactic coordinates and metallicities (drop all objects without metallicities), export the result in csv file and load in R
2. Build histograms for distribution of objects in galactic latitude for different intervals of metallicity
3. Build coordinate plot for the distribution of objects on coordinate plane,use different symbols and colors for different metallicity bins.

2. R AND MYSQL

2.1. PATCH MYSQL CONFIGURATION FILE AND START MYSQL

In your home directory add to the file `/etc/my.cnf` record for the connection of a client application:

```
[client]
socket =/Users/users/belikov/mysql/mysql.sock
```

with your socket.

Launch MySQL server with command

```
>mysqld_safe --defaults-file=/Users/users/belikov/mysql/my.cnf &
```

2.2. CONNECT TO MYSQL SERVER FROM R

Start R and in R environment load library RMySQL

```
R version 2.9.0 (2009-04-17)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(RMySQL)
Loading required package: DBI
```

Make a connection with database "prob" as a default database

```
con <- dbConnect(MySQL(), default.file='/Users/users/belikov/mysql/my.cnf',dbname='prob',user='root')
```

put in `default.file` a full path to your default file.

Check this connection - list tables

```
> dbListTables(con)
[1] "CROSSID" "TWOMASS" "USNOA2"
```

List columns of a table:

```
dbListFields(con,"USNOA2")
[1] "ID"      "RA2000"  "DEC2000" "USNOID"  "ACTFLAG" "MFLAG"   "BMAG"
[8] "RMAG"    "EPOCH"
```

2.3. RETRIEVE DATA FROM THE DATABASE

Select coordinates from TWOMASS table:

```
que=dbSendQuery(con, "select RA2000, DEC2000 from TWOMASS")
```

Retrieve this data into R structure:

```
data1<-fetch(que,n=-1)
```

With n you can specify how many records should be selected, n=-1 retrieve all records.

The same can be done with the direct execution of SQL statement:

```
>data2 <- dbGetQuery(con, "select RA2000, DEC2000 from TWOMASS")
```

which give you all results.

2.4. LOAD/UNLOAD DATA FROM R TO/FROM DATABASE

Write a result of data selection to a new table in the database

```
> dbWriteTable(con,"TWOMASSC",data2)
[1] TRUE
```

You wrote all output of the previous request table to the new table TWOMASSC which was created on-the-fly.

```
> dbListFields(con,"TWOMASSC")
[1] "row_names" "RA2000"    "DEC2000"
```

The same goes for csv-files:

```
> dbWriteTable(con,"TWOMASSBIS","2mass.csv")
[1] TRUE
```

Note that the first argument is a path to the file, and the file should be true comma-separated file, overvise you will dump all in a single column.

Check the table

```
dbListFields(con,"TWOMASSBIS")
 [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12"
[13] "V13" "V14" "V15" "V16" "V17" "V18"
```

2.5. PLOT THE DATA

Read the whole table into R data frame:

```
twomass <- dbReadTable(con, "TWOMASS")
```

Check names:

```
> names(twomass)
 [1] "ID"         "RA2000"     "DEC2000"    "TWOMASSID" "JMAG"       "EJMAG"
 [7] "HMAG"       "EHMAG"      "KMAG"       "EKMAG"      "Qflg"       "Rflg"
[13] "Bflg"       "Cflg"       "Xflg"       "Aflg"
```

Plot CMD:

```
> plot(twomass$HMAG-twomass$KMAG,twomass$HMAG,xlab="H-K",ylab="H",ylim=c(20,5),xlim=c(-2,4))
```

Plot histogram with 10 steps:

```
>hist(twomass$HMAG, breaks=10)
```

And, finally, search for a compliteness limit for H magnitude (original 2MASS) in color interval $(H-K) > 1.0$ mag:

```
> data2 <- dbGetQuery(con, "select HMAG from TWOMASS where HMAG-KMAG > 1.0")
> names(data2)
[1] "HMAG"
> hist(data2$HMAG, breaks=30)
```

Save the plot in postscript file:

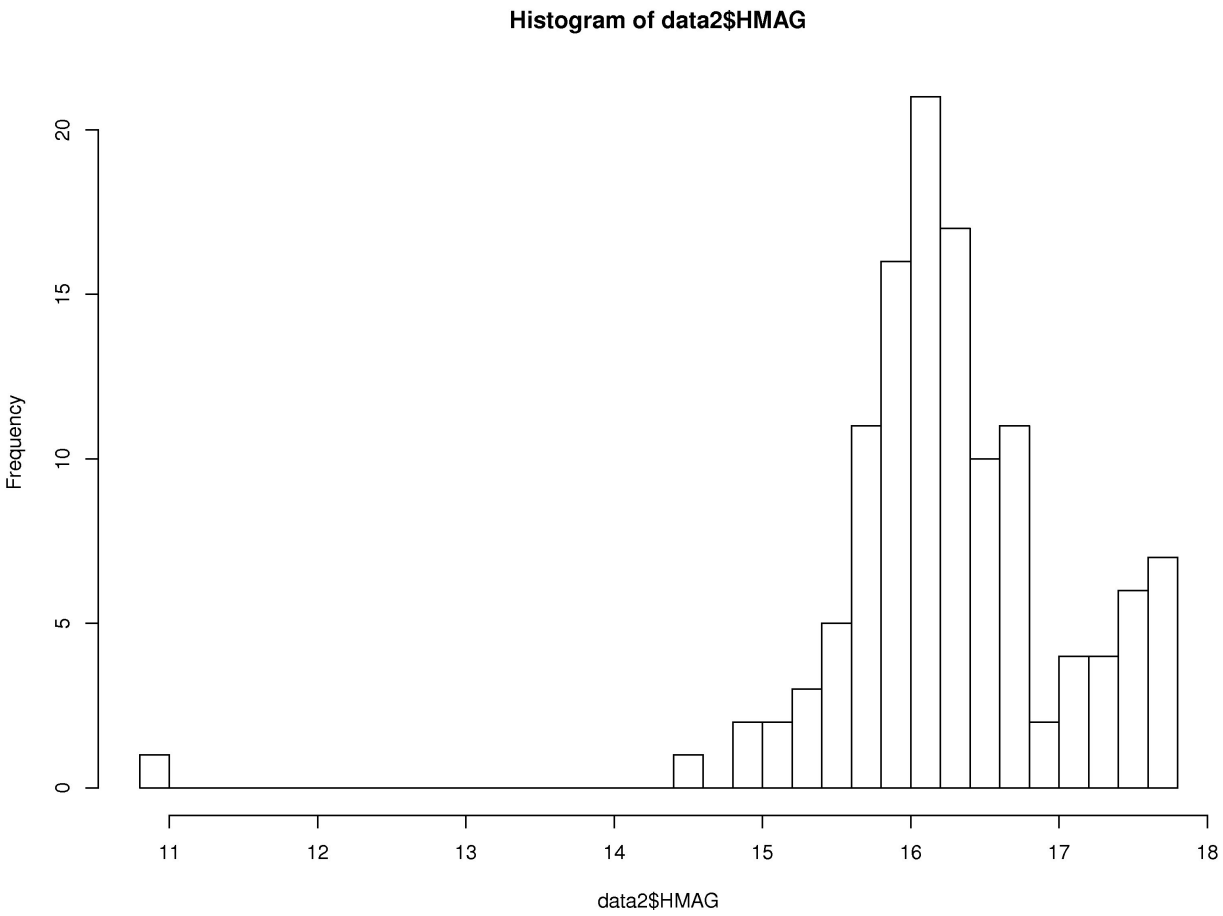

Figure 1: An example of histogram with R. Please, note that completeness list is 16.2 mag

```
> postscript('hist.ps')
> hist(data2$HMAG, breaks=30)
> dev.off()
X11cairo
       2
```

You can use `jpeg('hist.jpg')` instead of `postscript()`.

Do not forget to disconnect at the end:

```
>dbDisconnect(con)
```

3. KERNEL SMOOTHING

1. Connect to your MySQL database from R
2. select J magnitudes from 2MASS table

```
>jmag <- dbGetQuery(con, "select t1.JMAG from TWOMASS t1")
```

3. find brightest and faintest stars in the sample

```
> min(jmag$JMAG);max(jmag$JMAG)
[1] 4.9
[1] 18.657
```

4. plot a histogram for J magnitude
5. find a density estimation with epanechnikov kernel and bandwidth 0.1 - note, input is a vector not data frame

```
rd<-density(jmag$JMAG,kernel=c("epanechnikov"),bw=0.1)
```

6. plot the resulting distribution

```
>plot(rd)
```

7. find the completeness limit

```
> print(rd$x[rd$y==max(rd$y)])
[1] 16.54075
```

3.1. TASK - COMPLETENESS LIMIS IN ORIGINAL AND CROSS-IDENTIFIED DATA

1. Repeat previous tasks for all magnitudes from original and cross-identified catalogs (remember to write a correct SQL statement)
2. Plot all density functions
3. Write a table of completeness limits

4. DUMMY PCA

1. generate a sample of the random data

```
>x<-rnorm(100)
>y<-0.23*x+4.0+0.2*rnorm(100)
```

2. create a data frame

```
>xy<-data.frame(x,y)
```

3. perfome PCA with R function prcomp. Note, that the averages will be deduced automatically

```
> rp<-prcomp(xy)
```

   Compare deviations with original values:

```
> print(rp)
```

4. Use PCs to compute new coordinates and plot them.

5. DUMMY LINEAR DESCRIMINATION

1. Generate dummy dataset:

```
>x1<-5.0+0.5*rnorm(50)
>x2<-0.0+0.5*rnorm(50)
```

   Create an empty classificator and group all in data frame

```
>cl<-rnorm(50)
>d<-data.frame(cl,x1,x2)
>names(d)<-c("cl","x1","x2")
```

2. “Classify” it according to some combination of (x1,x2)

```
>d$cl[d$x1 > d$x2]=0
>d$cl[d$x1 <= d$x2]=1
```

   Plot 2 “samples”

```
>plot(d$x1[d$cl==0],d$x2[d$cl==0],col="blue")
>points(d$x1[d$cl==1],d$x2[d$cl==1])
```

3. Performe lda

```
>library(MASS)
>g<-lda(cl ~ x1+x2,data=d)
```

4. Find a line which divide 2 samples:

```
>gmean<-g$prior%*%g$means
>const<-as.numeric(gmean%*%g$scaling)
>a<- -g$scaling[1]/g$scaling[2]
>b<-const/g$scaling[2]
```

   Overplot this line

```
>xp<-c(-100,100)
>yp<-a*xp+b
>lines(xp,yp,col="red")
```

5. You can use g for classification of other samples

```
>pr<-predict(g,inp_sample)
```

6. CLASSIFICATION WITH R

The theme of this tutorial is a different methods of classification with R. We will briefly go through following ways to classify a sample:

1. k-nearest neighbour
2. linear discrimination
3. quadratic discrimination
4. neural network

6.1. TRAIN AND TEST SAMPLES

You have to read 2 samples: the data with the training sample and the data you are going to classify

```
>df_train=read.csv("train.csv",header=TRUE)
>df_test=read.csv("test.csv",header=TRUE)
```

6.2. K-NEAREST NEIGHBOURS

Load the package which contains knn method

```
>library(class)
>?knn
```

Try to classify your dataset:

```
k_res=knn(as.matrix(data.frame(df_train$x,df_train$y)),
          as.matrix(data.frame(df_test$x,df_test$y)),as.matrix(df_train$class),k=3)
```

In the case above you've used 3 nearst neighbours. Try to change `k` and compare result. Plot the train and test data.

6.2.1. FAILURE RATIO

What is the ratio of missclassification? Try to do the test on the training dataset itself:

```
k_t=knn(as.matrix(data.frame(df_train$x,df_train$y)),
        as.matrix(data.frame(df_train$x,df_train$y)),as.matrix(df_train$class),k=3)
```

Compare results:

```
>comp=as.numeric(as.matrix(k_t))-as.numeric(df_train$class)
>i_wrong=length(abs(comp[comp!=0]))/length(comp)
>print(i_wrong)
```

Try to change number of neighbours and trace `i_wrong`. When the method start to fail? Plot different cases to get the answer.

6.3. LINEAR DISCRIMINATION

Before creating classificator put both train and test samples in the same coordinate system to use in classificator:

```
>newtrain<-data.frame(df_train$x,df_train$y)
>newtest<-data.frame(df_test$x,df_test$y)
>names(newtrain)<-c("x","y")
>names(newtest)<-c("x","y")
```

Load library and create a classificator:

```
>library(MASS)
>lcl=lda(newtrain,df_train$class)
```

This classificator you can use on your data:

```
>cl=predict(lcl,newtest)
```

The result you will find in `cl$class`. Plot training dataset and test dataset with classification. Plot the discrimination line (see tasks to this werkcollege to find coefficients).

6.4. QUADRATIC DISCRIMINATION

Repeat the section above, this time with `qda`

```
>lcl=qda(newtrain,df_train$class)
>cl=predict(lcl,newtest)
```

It is easy to draw a partitioning line with the use of package `klaR` (should be installed first!) Load package from the page of werkcollege and install it in your local Rlib directory:

```
virgo01>R CMD INSTALL -l /Users/user/<your_name>/R_libs klaR_0.6-3.tar.gz
```

Draw a plot:

```
> drawparti(cl$class,df_test$x,df_test$y,method="qda")
```

6.5. NEURAL NETWORK

Repeat the training and classification, this time with neural network of 5 elements in the hidden layer and linear output units

```
>library(nnet)
>ncl=nnet(data.frame(df_train$x,df_train$y),df_train$class,size=5,linout=TRUE)
>cl=round(predict(ncl,df_test)
```

Plot the result with the training set.