MUBD

Màster Universitari en Enginyeria de Dades Massives (Big Data)

Estadística

laSalle ENG
Universitat Ramon Llull

Paquete data.table

Utilidad

- Es una alternativa a los *data.frames* convencionales

- Reduce el tiempo de programación: sintaxis más compacta
  - Menos llamadas a funciones, menos repeticiones de nombres

- Reduce el tiempo de computación
  - Agregación y cambios más rápidos

- Consumo de memoria ligeramente superior al inicio y menor a medida que se modifica el *data.table* en comparación con el *data.frame*

- Ventaja: un *data.table* también es un *data.frame*. Esto implica que es usable en todas las funciones que requieren un *data.frame*

Paquete data.table

Sintaxis

- Sintaxis de data.table
  - dt [i , j, by] → i = WHERE ; j = SELECT ; by = GROUP BY

```
iris.df <- as.data.frame(iris)
```

```
iris.dt <- as.data.table(iris)
```

- Media según especie
  - `with(iris, tapply(Sepal.Length,Species,mean))`
  - `iris.dt[,mean(Sepal.Length),by=Species]`
- Seleccionar Sepal.Width<3
  - `iris.df[iris.df$Sepal.Width<3,]`
  - `iris.dt[Sepal.Width<3]`
- Media según especie para Sepal.Width<3
  - `with(iris.df[iris.df$Sepal.Width<3,],tapply(Sepal.Length,Species,mean))`
  - `iris.dt[Sepal.Width<3,mean(Sepal.Length),by=Species]`

Paquete data.table

Tiempo de computación

- Lectura

```
url <- 'https://raw.githubusercontent.com/wiki/arunsrinivasan/flights/NYCflights14/flights14.csv'
system.time(fly.df <- read.csv(url(url)))
   user  system elapsed
  30.64    0.10   32.75
system.time(fly.dt <- fread(url))
   user  system elapsed
   0.42    0.10   15.07
```

- Selección

```
system.time(fly.df.AA <- fly.df[fly.df$carrier=='AA',])
   user  system elapsed
   0.11    0.00    0.11
system.time(fly.dt.AA <- fly.dt[carrier=='AA'])
   user  system elapsed
      0       0       0
```

Paquete data.table

Consumo de memoria similar

- Inicio

```
> format(object.size(fly.df),units = "MB")
[1] "16.6 Mb"
> format(object.size(fly.dt),units = "MB")
[1] "21.4 Mb"
```

- Una operación sencilla (incrementa en uno y disminuye en otro)

```
> format(object.size(fly.df[-1,]),units = "MB")
[1] "17.6 Mb"
> format(object.size(fly.dt[-1]),units = "MB")
[1] "20.5 Mb"
```

Paquetes Biglm, bigMemory y bigAnalytics

Ajuste de modelos

- *biglm*. Ajuste de modelo lineal y logístico (entre otros) de forma más eficiente
  - *biglm*. Modelo lineal
  - *bigglm*. Modelo logístico (y otros)

- *bigmemory*
  - *big.matrix*. Reduce el espacio de los datos

- biganalytics:
  - *bigkmeans*. Kmeans de forma más eficiente

Otros paquetes

Recopilatorio de paquetes

- Más de un procesador. R por defecto, trabaja con un solo procesador. Hay paquetes que lo pueden hacer trabajar en paralelo: *foreach*, *snowfall*
- Web Scrapping. Recuperar información de la web de forma sencilla: *rvest* y *Rcurl*
- Spark (http://spark.apache.org/): SparkR, sparklyr

Data mining

Reference card

R Reference Card for Data Mining

by Yanchang Zhao, yanchang@rdatamining.com, January 3, 2013
The latest version is available at `http://www.RDataMining.com`. Click the link also for document *R and Data Mining: Examples and Case Studies*.
The package names are in parentheses.

Association Rules & Frequent Itemsets

APRIORI Algorithm

a level-wise, breadth-first algorithm which counts transactions to find frequent itemsets

`apriori()` mine associations with APRIORI algorithm (*arules*)

ECLAT Algorithm

employs equivalence classes, depth-first search and set intersection instead of counting

`eclat()` mine frequent itemsets with the Eclat algorithm (*arules*)

Packages

arules mine frequent itemsets, maximal frequent itemsets, closed frequent itemsets and association rules. It includes two algorithms, Apriori and Eclat.

arulesViz visualizing association rules

Sequential Patterns

Functions

`cspade()` mining frequent sequential patterns with the cSPADE algorithm (*arulesSequences*)

`seqefsub()` searching for frequent subsequences (*TraMineR*)

Packages

arulesSequences add-on for *arules* to handle and mine frequent sequences

TraMineR mining, describing and visualizing sequences of states or events

Classification & Prediction

Decision Trees

`ctree()` conditional inference trees, recursive partitioning for continuous, censored, ordered, nominal and multivariate response variables in a conditional inference framework (*party*)

`rpart()` recursive partitioning and regression trees (*rpart*)

`mob()` model-based recursive partitioning, yielding a tree with fitted models associated with each terminal node (*party*)

Random Forest

`cforest()` random forest and bagging ensemble (*party*)

`randomForest()` random forest (*randomForest*)

`varimp()` variable importance (*party*)

`importance()` variable importance (*randomForest*)

Neural Networks

`nnet()` fit single-hidden-layer neural network (*nnet*)

Support Vector Machine (SVM)

`svm()` train a support vector machine for regression, classification or density-estimation (*e1071*)

`ksvm()` support vector machines (*kernlab*)

Performance Evaluation

`performance()` provide various measures for evaluating performance of prediction and classification models (*ROCR*)

`roc()` build a ROC curve (*pROC*)

`auc()` compute the area under the ROC curve (*pROC*)

`ROC()` draw a ROC curve (*DiagnosisMed*)

`PRcurve()` precision-recall curves (*DMwR*)

`CRchart()` cumulative recall charts (*DMwR*)

Packages

rpart recursive partitioning and regression trees

party recursive partitioning

randomForest classification and regression based on a forest of trees using random inputs

rpartOrdinal ordinal classification trees, deriving a classification tree when the response to be predicted is ordinal

rpart.plot plots rpart models with an enhanced version of `plot.rpart` in the *rpart* package

ROCR visualize the performance of scoring classifiers

pROC display and analyze ROC curves

Regression

Functions

`lm()` linear regression

`glm()` generalized linear regression

`nls()` non-linear regression

`predict()` predict with models

`residuals()` residuals, the difference between observed values and fitted values

`gls()` fit a linear model using generalized least squares (*nlme*)

`gnls()` fit a nonlinear model using generalized least squares (*nlme*)

Packages

nlme linear and nonlinear mixed effects models

Clustering

Partitioning based Clustering

partition the data into k groups first and then try to improve the quality of clustering by moving objects from one group to another

`kmeans()` perform k-means clustering on a data matrix

`kmeansCBI()` interface function for `kmeans` (*fpc*)

`kmeansruns()` call `kmeans` for the k-means clustering method and includes estimation of the number of clusters and finding an optimal solution from several starting points (*fpc*)

`pam()` the Partitioning Around Medoids (PAM) clustering method (*cluster*)

`pamk()` the Partitioning Around Medoids (PAM) clustering method with estimation of number of clusters (*fpc*)

`cluster.optimal()` search for the optimal k-clustering of the dataset (*bayesclust*)

`clara()` Clustering Large Applications (*cluster*)

`fanny(x,k,...)` compute a fuzzy clustering of the data into `k` clusters (*cluster*)

`kcca()` k-centroids clustering (*flexclust*)

`ccfkms()` clustering with Conjugate Convex Functions (*cba*)

`apcluster()` affinity propagation clustering for a given similarity matrix (*apcluster*)

`apclusterK()` affinity propagation clustering to get K clusters (*apcluster*)

`cclust()` Convex Clustering, incl. k-means and two other clustering algorithms (*cclust*)

`KMeansSparseCluster()` sparse k-means clustering (*sparcl*)

`tclust(x,k,alpha,...)` trimmed k-means with which a proportion `alpha` of observations may be trimmed (*tclust*)

Hierarchical Clustering

a hierarchical decomposition of data in either bottom-up (agglomerative) or top-down (divisive) way

`hclust(d, method, ...)` hierarchical cluster analysis on a set of dissimilarities `d` using the `method` for agglomeration

`birch()` the BIRCH algorithm that clusters very large data with a CF-tree (*birch*)

`pvclust()` hierarchical clustering with p-values via multi-scale bootstrap re-sampling (*pvclust*)

`agnes()` agglomerative hierarchical clustering (*cluster*)

`diana()` divisive hierarchical clustering (*cluster*)

`mona()` divisive hierarchical clustering of a dataset with binary variables only (*cluster*)

`rockCluster()` cluster a data matrix using the Rock algorithm (*cba*)

`proximus()` cluster the rows of a logical matrix using the Proximus algorithm (*cba*)

`isopam()` Isopam clustering algorithm (*isopam*)

`LLAhclust()` hierarchical clustering based on likelihood linkage analysis (*LLAhclust*)

`flashClust()` optimal hierarchical clustering (*flashClust*)

`fastcluster()` fast hierarchical clustering (*fastcluster*)

`cutreeDynamic()`, `cutreeHybrid()` detection of clusters in hierarchical clustering dendrograms (*dynamicTreeCut*)

`HierarchicalSparseCluster()` hierarchical sparse clustering (*sparcl*)

Model based Clustering

`Mclust()` model-based clustering (*mclust*)

`HDDC()` a model-based method for high dimensional data clustering (*HDclassif*)

`fixmahal()` Mahalanobis Fixed Point Clustering (*fpc*)

`fixreg()` Regression Fixed Point Clustering (*fpc*)

`mergenormals()` clustering by merging Gaussian mixture components (*fpc*)

Density based Clustering

generate clusters by connecting dense regions

`dbscan(data,eps,MinPts,...)` generate a density based clustering of arbitrary shapes, with neighborhood radius set as `eps` and density threshold as `MinPts` (*fpc*)

`pdfCluster()` clustering via kernel density estimation (*pdfCluster*)

Other Clustering Techniques

`mixer()` random graph clustering (*mixer*)

`nncluster()` fast clustering with restarted minimum spanning tree (*nnclust*)

`orclus()` ORCLUS subspace clustering (*orclus*)

Plotting Clustering Solutions

`plotcluster()` visualisation of a clustering or grouping in data (*fpc*)

`bannerplot()` a horizontal barplot visualizing a hierarchical clustering (*cluster*)

MUBD

Màster Universitari en Enginyeria de Dades Massives (Big Data)

Estadística

laSalle ENG
Universitat Ramon Llull