Package 'RESS'

January 20, 2025

Type Package

Title Integrates R and Essentia

Version 1.3

Date 2015-10-26

Author Ben Waxer

Maintainer Ben Waxer <bwaxer@auriq.com>

Description Contains three functions that query AuriQ Systems' Essentia Database and return the results in R. 'essQuery' takes a single Essentia command and captures the output in R, where you can save the output to a dataframe or stream it directly into additional analysis. 'read.essentia' takes an Essentia script and captures the output csv data into R, where you can save the output to a dataframe or stream it directly into additional analysis. 'capture.essentia' takes a file containing any number of Essentia commands and captures the output of the specified statements into R dataframes. Essentia can be downloaded for free at http://www.auriq.com/documentation/source/install/index.html.

License LGPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-28 08:44:54

Imports utils

Contents

RESS-package *Integrates R and Essentia.*

Description

Contains three functions that query AuriQ Systems' Essentia Database and return the results in R.

'essQuery' takes a single Essentia command and captures the output in R, where you can save the output to a dataframe or stream it directly into additional analysis.

'read.essentia' takes an Essentia script and captures the output csv data into R, where you can save the output to a dataframe or stream it directly into additional analysis.

'capture.essentia' takes a file containing any number of Essentia commands and captures the output of the specified statements into R dataframes.

Essentia can be downloaded for free at http://www.auriq.com/documentation/source/install/index.html.

Details

Package:	RESS
Type:	Package
Version:	1.2
Date:	2015-10-26
License:	LGPL-3

Author(s)

Ben Waxer, Data Scientist with Auriq Systems.

Maintainer: Ben Waxer <bwaxer@auriq.com>

capture.essentia *capture.essentia*

Description

Read the essentia commands in the stated file, query the Essentia database, and save the results into R as dataframes.

Usage

```
capture.essentia(scriptcall = "", linenumber = "all", separator = " ")
```

Arguments

`scriptcall`	The file containing the essentia commands you want to run and any arguments you want to pass into the bash script. 'ess stream' and 'ess query' statements will have their output ignored by R unless there is a '#Rinclude' flag somehwere in the statement line. 'ess exec' statements will have their output included unless there is a '#Rignore' flag somewhere in the statement line. The file should contain primarily the query commands but can contain the entire essentia script including loading the Essentia database if desired. Any command that you want to capture the output from must have its output in csv format.
`linenumber`	This is set to default to all line numbers so that every command in the file is executed. You can specify the line number of the command you wish to run if you dont want to run the entire set of commands in the file. If your statement spans multiple lines, you can specify each line using the syntax c(first_line_number, second_line_number, ...).
`separator`	The character that should be used to split the script name and arguments you are passing into that script. The default is set to a space in the style of the linux command line; however, you can set it to whatever you want. This can be useful if one or more of the arguments you are trying to pass into your script contains an unquoted space.

Details

capture.essentia reads all of the statements in a file (unless linenumber is specified, see above) and captures the output of the specified commands into R dataframes. By default only the output of 'ess exec' statements is captured and it's stored in R dataframes command1 to commandN, where N is the number of captured statements.

You can include 'ess stream' or 'ess query' statements by adding a '#Rinclude' flag. This method can be used to stream multiple files into R for data exploration or analysis. If you plan to run multiple statements that may be somewhat related to each other, it is recommended that you use capture.essentia.

Value

If there is only one statement that is having its output captured and this statement does not contain a `#R#name#R#` flag, capture.essentia will simply return the data in R so you can save it however you want or stream it directly into additional analysis.

If there is more than one statement that is having its output captured, there is no value returned. This command creates a set of R dataframes containing the output from the specified essentia commands in the file specified in scriptcall.

Note

The flags added to the essentia commands in file can include:

`#Rignore` : Ignore an 'ess exec' statement. Do not capture the output of the statement into R.

`#Rinclude` : Include an 'ess stream' or 'ess query' statement. Capture the output of the statement into R.

`#-notitle` : Tell R not to use the first line of the output as the header.

`#Rseparate` : Can be used when saving multiple files into an R dataframe using an 'ess stream' command. Saves each file into a different R dataframe.

`#filelist` : Causes an extra dataframe to be stored in R that saves the list of files streamed into R when streaming multiple files.

`#R#name#R#` : Allows any automatically saved dataframe to be renamed to whatever is entered in place of 'name'. When used with #Rseparate, saves the files as name1 to nameN, where N is the number of files. Since this still counts as a statement, the next default dataframe saved will be stored as command followed by the number of previous statements run plus one.

Author(s)

Ben Waxer, Data Scientist with Auriq Systems.

References

See our website at www.auriq.com or our documentation at www.auriq.com/documentation.

Examples

```
## Not run:
-----------------------------------------------------------------------------------------------

These examples require Essentia to be installed:

queryfile <- file("examplequery.sh","w")
cat("ess exec \"echo -e '11,12,13\n4,5,6\n7,8,9'\" #-notitle \n",file=queryfile)
cat("ess exec \"echo -e '11,12,13\n4,5,6\n7,8,9'\" \n", file=queryfile)
cat("ess exec \"echo -e '11,12,13\n4,5,6\n7,8,9'\" #Rignore \n", file=queryfile)
capture.essentia("examplequery.sh")
print(command1)
print(command2)
print("The last statement is ignored by R and just executed on the command line.")

-----------------------------------------------------------------------------------------------

This example requires Essentia to have selected a datastore containing purchase log data:

Store these lines as querypurchase.sh:

ess query "select count(refID) from purchase:2014-09-01:2014-09-15 \
where articleID>=46 group by price" #Rinclude
ess query "select count(distinct userID) from purchase:2014-09-01:2014-09-15 \
where articleID>=46" #Rinclude
ess query "select count(refID) from purchase:2014-09-01:2014-09-15 \
where articleID>=46 group by userID" #Rinclude
ess query "select * from purchase:*:* where articleID <= 20" #Rinclude #R#querystream#R# #-notitle

-----------------------------------------------

Then run these commands in R:
```

```
library(RESS)
capture.essentia("querypurchase.sh")
print(command1)
print(command2)
print(command3)
print(querystream)
```

The following example requires Essentia to be installed with apache log data stored in it.

Store the following lines as queryapache.sh:

```
# Query the Essentia database logsapache3 and save the contents of vector3 in R as command1.
ess exec "aq_udb -exp logsapache3:vector3" --debug

# Query the Essentia database logsapache1 and save the sorted contents of vector1 in R as command2.
ess exec "aq_udb -exp logsapache1:vector1 -sort pagecount -dec" --debug

# Stream the last five lines of the file in category 125accesslogs between dates 2014-12-07 and
# 2014-12-07, convert them to csv, return them to R, and store them into an R dataframe singlefile.
ess stream 125accesslogs '2014-12-07' '2014-12-07' "tail -5 \
| logcnv -f,eok - -d ip:ip sep:' ' s:rlog sep:' ' s:rusr sep:' [' i,tim:time sep:'] \"' \
s,clf:req_line1 sep:' ' s,clf:req_line2 sep:' ' s,clf:req_line3 sep:'\" ' i:res_status sep:' ' \
i:res_size sep:' \"' s,clf:referrer sep:'\" \"' \
s,clf:user_agent sep:'\"' X | cat -" #Rinclude #R#singlefile#R#

# Stream the last five lines of the files in category 125accesslogs between dates 2014-11-30 and
# 2014-12-07, convert them to csv, and save them into R dataframes apachefiles1 and apachefiles2.
ess stream 125accesslogs '2014-11-30' '2014-12-07' "tail -5 \
| logcnv -f,eok - -d ip:ip sep:' ' s:rlog sep:' ' s:rusr sep:' [' i,tim:time sep:'] \"' \
s,clf:req_line1 sep:' ' s,clf:req_line2 sep:' ' s,clf:req_line3 sep:'\" ' i:res_status sep:' ' \
i:res_size sep:' \"' s,clf:referrer sep:'\" \"' s,clf:user_agent sep:'\"' X -notitle | cat -" \
#Rinclude #R#apachefiles#R# #Rseparate
```

Then run these commands in R:

```
library(RESS)
capture.essentia("queryapache.sh")

print(command1)
print(command2)
print(singlefile)
print(apachefiles1)
print(apachefiles2)
```

The references contain more extensive examples that
fully walkthrough how to load and query the Essentia Database.

```
## End(Not run)
```

---

essQuery *essQuery*

Description

Query the Essentia database and return the results to R.

Usage

```
essQuery(essentia, aq="", flags="")
```

Arguments

essentia	The essentia command to run. The options are "ess stream category startdate enddate", "ess exec", and "ess query". Each stream or query command can be used to stream any number of files directly into your R analysis. Alternatively, each stream command can save multiple files into separate R dataframes, one file per dataframe. The default value for the essentia argument is "ess exec".
aq	This can be any combination of the aq_tools and standard UNIX commands for "ess stream" and "ess exec" statements or an sql-like statement for "ess query" statements. However, the output MUST be in a csv format if you want R to capture the output. If you only want to run the command without R capturing the output, add "#Rignore" to the flags argument.
flags	Any of the essentia flags can be used here in addition to any of these RESS-specific flags: #Rignore : Ignore an 'ess exec' statement. Do not capture the output of the statement into R. #Rinclude : Include an 'ess stream' or 'ess query' statement. Capture the output of the statement into R. #-notitle : Tell R not to use the first line of the output as the header. #Rseparate : Can be used when saving multiple files into an R dataframe using an 'ess stream' command. Saves each file into a different R dataframe, entitled command1 to commandN, where N is the number of files. #filelist : Causes an extra dataframe to be stored in R that saves the list of files streamed into R when streaming multiple files. #R#name#R# : Allows any automatically saved dataframe to be renamed to whatever is entered in place of 'name'. This only applies in essQuery when streaming multiple files with #Rseparate.

Details

essQuery is used to directly query the database using a single statement. You can call essQuery multiple times to run different statements.

However, you can also use capture.essentia to read all of the statements in a file instead. Thus if you plan to run multiple statements that may be somewhat related to each other, it is recommended that you use capture.essentia.

Value

The value returned is the output from querying the database. This can be saved into an R dataframe or directly analyzed in R.

If you use essQuery to save multiple files into separate R dataframes using a single stream command, the files are stored automatically in R dataframes called command1 to commandN (where N is the number of files) and no value is returned. To change the name of the stored dataframes, use the #R#any_name#R# flag. The dataframes will then be stored as any_name1 to any_nameN.

With #filelist, the extra dataframe is saved as commandN+1 by default, or any_nameN+1 if #R#any_name#R# is also used.

Author(s)

Ben Waxer, Data Scientist with Auriq Systems.

References

See our website at www.auriq.com or our documentation at www.auriq.com/documentation

Examples

```
## Not run:
-----------------------------------------------------------------------------------------------

These examples require Essentia to be installed:

fullexec <- essQuery("ess exec", "echo -e '11,12,13\n4,5,6\n7,8,9' ","#-notitle")
print(fullexec)
defaultexec <- essQuery("echo -e '11,12,13\n4,5,6\n7,8,9' ","#-notitle")
print(defaultexec)
essQuery("echo -e '11,12,13\n4,5,6\n7,8,9' ","#Rignore")
print("This last statement is ignored by R and just executed on the command line.")

-----------------------------------------------------------------------------------------------

This example requires Essentia to have selected a datastore containing purchase log data:

command1 <- essQuery("ess query","select count(refID) from purchase:2014-09-01:2014-09-15 \
where articleID>=46 group by price","#Rinclude")
command2 <- essQuery("ess query", "select count(distinct userID) from \
purchase:2014-09-01:2014-09-15 where articleID>=46", "#Rinclude")
command3 <- essQuery("ess query", "select count(refID) from \
purchase:2014-09-01:2014-09-15 where articleID>=46 group by userID", "#Rinclude")
```

```
querystream <- essQuery("ess query", "select * from purchase:*:* where articleID <= 20", "\
#Rinclude #-notitle")

Then run these commands to view the saved dataframes:

print(command1)
print(command2)
print(command3)
print(querystream)

-----------------------------------------------------------------------------------------------

The following example requires Essentia to be installed with apache log data stored in it:

# Query the Essentia database logsapache3 and return the contents of vector3 into R.
command1 <- essQuery("aq_udb -exp logsapache3:vector3", "--debug")

# Query the Essentia database logsapache1 and return the sorted contents of vector1 into R.
command2 <- essQuery("ess exec", "aq_udb -exp logsapache1:vector1 -sort pagecount -dec", "\
--debug")

# Stream the last five lines of the file in category 125accesslogs between dates 2014-12-07 and
# 2014-12-07, convert them to csv, return them to R, and store them into an R dataframe singlefile.
singlefile <- essQuery("ess stream 125accesslogs '2014-12-07' '2014-12-07'","tail -5 \
| logcnv -f,eok - -d ip:ip sep:' ' s:rlog sep:' ' s:rusr sep:' [' i,tim:time sep:'] \\"' \
s,clf:req_line1 sep:' ' s,clf:req_line2 sep:' ' s,clf:req_line3 sep:'\\" ' i:res_status sep:' ' \
i:res_size sep:' \\"' s,clf:referrer sep:'\\" \\"' \
s,clf:user_agent sep:'\\"' X | cat -","#Rinclude")

# Stream the last five lines of the files in category 125accesslogs between dates 2014-11-30 and
# 2014-12-07, convert them to csv, and save them into R dataframes apachefiles1 and apachefiles2.
essQuery("ess stream 125accesslogs '2014-11-30' '2014-12-07'","tail -5 \
| logcnv -f,eok - -d ip:ip sep:' ' s:rlog sep:' ' s:rusr sep:' [' i,tim:time sep:'] \\"' \
s,clf:req_line1 sep:' ' s,clf:req_line2 sep:' ' s,clf:req_line3 sep:'\\" ' i:res_status sep:' ' \
i:res_size sep:' \\"' s,clf:referrer sep:'\\" \\"' \
s,clf:user_agent sep:'\\"' X -notitle | cat -","\
#Rinclude #R#apachefiles#R# #Rseparate")

print(command1)
print(command2)
print(singlefile)
print(apachefiles1)
print(apachefiles2)

The references contain more extensive examples that
fully walkthrough how to load and query the Essentia Database.

## End(Not run)
```

---

read.essentia *read.essentia*

Description

Read the essentia commands in the stated file, and save the csv output into R. This command is primarily intended to capture the output of a query to the Essentia database.

Usage

```
read.essentia(file)
```

Arguments

`file`	The essentia script to run and any arguments you want to pass into the bash script.

Details

read.essentia is used to directly query the database using a script with a single csv formatted output. You can call read.essentia multiple times to run statements that produce different outputs and capture their respective output into R.

However, you can also use capture.essentia to read all of the statements in a file instead and capture the output of any or all of the statements. Thus if you plan to run multiple statements that may be somewhat related to each other, you may want to use capture.essentia.

Value

The value returned is the output from querying the database. This can be saved into an R dataframe or directly analyzed in R.

Note

The argument, file, can also include the following flag:

`#-notitle` : Tell R not to use the first line of the output as the header.

Author(s)

Ben Waxer, Data Scientist with Auriq Systems.

References

See our website at www.auriq.com or our documentation at www.auriq.com/documentation

Examples

```
## Not run:
-----------------------------------------------------------------------------------------------

These examples require Essentia to be installed:

queryfile <- file("examplequery.sh","w")
cat("ess exec \"echo -e '11,12,13\n4,5,6\n7,8,9'\" \n", file=queryfile)
simpleecho <- read.essentia("examplequery.sh")
```

```
print(simpleecho)

-----------------------------------------------------------------------------------------------

This example requires Essentia to have selected a datastore containing purchase log data:

Store these lines as querypurchase.sh:

ess query "select count(refID) from purchase:2014-09-01:2014-09-15 \
where articleID>=46 group by userID"

-----------------------------------------------

Then run these commands in R:

library(RESS)
print(read.essentia("querypurchase.sh"))


-----------------------------------------------------------------------------------------------

The following example requires Essentia to be installed with apache log data stored in it.

Store the following lines as queryapache.sh:

# Query the Essentia database logsapache1 and save the sorted contents of vector1 in R as command2.
ess exec "aq_udb -exp logsapache1:vector1 -sort pagecount -dec"


-----------------------------------------------

Then run these commands in R:

library(RESS)
mydata <- read.essentia("queryapache.sh")

print(mydata)


The references contain more extensive examples that
fully walkthrough how to load and query the Essentia Database.


## End(Not run)
```

Index