Package 'yhatr'

July 29, 2015

Type Package

Title R Binder for the Yhat API

Version 0.13.7

Date 2015-07-29

Author Greg Lamp <greg@yhathq.com>, Eric Chiang <eric@yhathq.com>

Maintainer Greg Lamp <greg@yhathq.com>

Description Deploy, maintain, and invoke models via the Yhat REST API.

Depends R (>= 2.12.0)

URL https://github.com/yhat/yhatr

Imports httr, RCurl, rjson, plyr, jsonlite, stringr

License FreeBSD

NeedsCompilation no

Repository CRAN

Date/Publication 2015-07-29 17:51:24

R topics documented:

`capture.src` *Private function for catpuring the source code of model*

Description

Private function for catpuring the source code of model

Usage

```
capture.src(funcs)
```

Arguments

`funcs`	functions to caputre, defaults to required yhat model functions

---

`check.dependencies` *Checks dependencies and makes sure all are installed.*

Description

Checks dependencies and makes sure all are installed.

Usage

```
check.dependencies()
```

`check.image.size` *Private function for checking the size of the user's image.*

Description

Private function for checking the size of the user's image.

Usage

```
check.image.size()
```

is.https *Private predicate function that checks if the protocol of a url is https.*

Description

Private predicate function that checks if the protocol of a url is https.

Usage

```
is.https(x)
```

Arguments

x	is a url string

yhat.deploy *Deploy a model to Yhat's servers*

Description

This function takes model.transform and model.predict and creates a model on Yhat's servers which can be called from any programming language via Yhat's REST API (see yhat.predict).

Usage

```
yhat.deploy(model_name, packages = c())
```

Arguments

model_name	name of your model
packages	list of packages to install using apt-get

Examples

```
yhat.config <- c(
 username = "your username",
 apikey = "your apikey",
 env = "http://sandbox.yhathq.com/"
)
iris$Sepal.Width_sq <- iris$Sepal.Width^2
fit <- glm(I(Species)=="virginica" ~ ., data=iris)

model.require <- function() {
 # require("randomForest")
}

model.transform <- function(df) {
```

```
 df$Sepal.Width_sq <- df$Sepal.Width^2
 df
}
model.predict <- function(df) {
 data.frame("prediction"=predict(fit, df, type="response"))
}
## Not run:
yhat.deploy("irisModel")

## End(Not run)
```

`yhat.deploy.to.file` *Deploy a model to a file that you can then upload via the browser.*

Description

This function creates a .yhat file which can be deployed via the browser. This is useful for larger models (>20 MB).

Usage

```
yhat.deploy.to.file(model_name)
```

Arguments

`model_name`	name of your model

Examples

```
yhat.config <- c(
 username = "your username",
 apikey = "your apikey",
 env = "http://sandbox.yhathq.com/"
)
iris$Sepal.Width_sq <- iris$Sepal.Width^2
fit <- glm(I(Species)=="virginica" ~ ., data=iris)

model.require <- function() {
 # require("randomForest")
}

model.transform <- function(df) {
 df$Sepal.Width_sq <- df$Sepal.Width^2
 df
}
model.predict <- function(df) {
 data.frame("prediction"=predict(fit, df, type="response"))
}
## Not run:
 yhat.deploy.to.file("irisModel")
```

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

`yhat.deploy.with.scp` *Deploy a model via SCP. For when you want to automate large model uploads.*

Description

For when you have a really big model file and you don't want to mess with uploading it via the admin console. This is useful for larger models (>20 MB).

Usage

```
yhat.deploy.with.scp(model_name, pem_path)
```

Arguments

`model_name`	name of your model
`pem_path`	path to your pemfile (for AWS)

Examples

```
yhat.config <- c(
 username = "your username",
 apikey = "your apikey",
 env = "http://google.yhathq.com/"
)
iris$Sepal.Width_sq <- iris$Sepal.Width^2
fit <- glm(I(Species)=="virginica" ~ ., data=iris)

model.require <- function() {
 # require("randomForest")
}

model.transform <- function(df) {
 df$Sepal.Width_sq <- df$Sepal.Width^2
 df
}
model.predict <- function(df) {
 data.frame("prediction"=predict(fit, df, type="response"))
}
## Not run:
 yhat.deploy.with.scp("irisModel", "~/path/to/pemfile.pem")

## End(Not run)
```

yhat.get *Private function for performing a GET request*

Description

Private function for performing a GET request

Usage

```
yhat.get(endpoint, query = c())
```

Arguments

endpoint	/path for REST request
query	url parameters for request

yhat.ls *Private function for determining model dependencies*

Description

List all object names which are dependencies of ‘model.transform‘ and ‘model.predict‘

Usage

```
yhat.ls()
```

yhat.post *Private function for performing a POST request*

Description

Private function for performing a POST request

Usage

```
yhat.post(endpoint, query = c(), data, silent = TRUE)
```

Arguments

endpoint	/path for REST request
query	url parameters for request
data	payload to be converted to raw JSON
silent	should output of url to console be silenced? Default is FALSE.

yhat.predict — *Make a prediction using Yhat.*

Description

This function calls Yhat's REST API and returns a response formatted as a data frame.

Usage

```
yhat.predict(model_name, data, model_owner, raw_input = FALSE,
  silent = TRUE)
```

Arguments

model_name	the name of the model you want to call
data	input data for the model
model_owner	the owner of the model [optional]
raw_input	when true, incoming data will NOT be coerced into data.frame
silent	should output of url to console (via yhat.post) be silenced? Default is FALSE.

Examples

```
yhat.config <- c(
 username = "your username",
 apikey = "your apikey",
 env = "http://sandbox.yhathq.com/"
)
## Not run:
yhat.predict("irisModel", iris)

## End(Not run)
```

yhat.predict_raw — *Calls Yhat's REST API and returns a JSON document containing both the prediction and associated metadata.*

Description

Calls Yhat's REST API and returns a JSON document containing both the prediction and associated metadata.

Usage

```
yhat.predict_raw(model_name, data, model_owner, raw_input = FALSE,
  silent = TRUE)
```

Arguments

model_name	the name of the model you want to call
data	input data for the model
model_owner	the owner of the model [optional]
raw_input	when true, incoming data will NOT be coerced into data.frame
silent	should output of url to console (via yhat.post) be silenced? Default is FALSE.

Examples

```
yhat.config <- c(
 username = "your username",
 apikey = "your apikey"
)
## Not run:
yhat.predict_raw("irisModel", iris)

## End(Not run)
```

---

yhat.scaffolding — *Quick function for setting up a basic scaffolding of functions for deploying on Yhat.*

Description

Quick function for setting up a basic scaffolding of functions for deploying on Yhat.

Usage

```
yhat.scaffolding()
```

Examples

```
yhat.scaffolding()
```

---

yhat.show_models — *Shows which models you have deployed on Yhat.*

Description

This function queries the Yhat API and finds the models that have been deployed for your account.

Usage

```
yhat.show_models()
```

Examples

```
yhat.config <- c(
 username = "your username",
 apikey = "your apikey",
 env = "http://sandbox.yhathq.com/"
)
## Not run:
yhat.show_models()

## End(Not run)
# some output here
#    username className                 name version
# 1      greg                MySMSClassifier       1
# 2      greg                MySMSClassifier       2
# 3      greg                MySMSClassifier       3
# 4      greg                MySMSClassifier       4
```

`yhat.spider.block` *Private function for recursively looking for variables*

Description

Private function for recursively looking for variables

Usage

```
yhat.spider.block(block, defined.vars = c())
```

Arguments

`block`	code block to spider
`defined.vars`	variables which have already been defined within the scope of the block. e.g. function argument

---

`yhat.spider.func` *Private function for spidering function source code*

Description

Private function for spidering function source code

Usage

```
yhat.spider.func(func.name)
```

Arguments

`func.name`	name of function you want to spider

yhat.test_predict *Test a prediction through the JSONification process*

Description

This function tests model.transform and model.predict on new data by sending it through a JSONification process before the two stated functions. This allows users to test their model locally in conditions that are similar to those after a deployment.

Usage

```
yhat.test_predict(data, verbose = FALSE)
```

Arguments

data	Data to envoke the model with
verbose	Whether or not to print intermediate results

Examples

```
model.transform <- function(df) {
 df$Sepal.Width_sq <- df$Sepal.Width^2
 df
}
model.predict <- function(df) {
 data.frame("prediction"=predict(fit, df, type="response"))
}
## Not run:
model.test_predict(iris)

## End(Not run)
```

yhat.transform_from_example *Generates a model.transform function from an example input data.frame. Handles columns which need to be type casted further after the initial JSON to Robject such as factors and ints.*

Description

Generates a model.transform function from an example input data.frame. Handles columns which need to be type casted further after the initial JSON to Robject such as factors and ints.

Usage

```
yhat.transform_from_example(df)
```

Arguments

df	A data.frame object which mirrors the kind of input to the model.

Examples

```
## Not run:
model.transform <- yhat.transform_from_example(iris)

## End(Not run)
```

---

yhat.verify *Private function for verifying username and apikey*

Description

Private function for verifying username and apikey

Usage

```
yhat.verify()
```

Index