dbhydroR: An R package to access the DBHYDRO Environmental Database

Joseph Stachelek

September 15, 2016

1 Introduction

This document introduces the `dbhydroR` package and its associated functions. These functions are aimed at improving programmatic workflows that query the DBHYDRO Environmental Database which holds over 35 million hydrologic and water quality records from the Florida Everglades and surrounding areas.

2 Package installation

Computers running the Windows operating system can only install binary package archive files unless they have additional compiler software installed. Without this software, `dbhydroR` can be installed from CRAN by running the following command in the `R` console:

2.1 Stable version from CRAN

```
install.packages("dbhydroR")
```

Otherwise, the `dbhydroR` can be installed by running the following command in the `R` console:

2.2 or development version from Github

```
devtools::install_github("SFWMD/dbhydroR")
```

Once installed, the package can be loaded using the following command:

```
library(dbhydroR)
```

3 Composing database queries

3.1 Water quality data

Water quality data can be retrieved using the `get_wq` function which takes four required arguments. The user must specify a station ID, a test name, and a date range. Station IDs can be located on the ArcGIS Online Station Map or the Google Earth kmz file. An abbreviated list of available test names can be found in the Appendix to this document while a full listing can be found at the DBHYDRO metadata page. Dates must be specified in YYYY-MM-DD format (e.g. 2015-02-26). The following set of examples retrieve measurements between March 2011 and May 2012. They can be run from the R console by issuing the command:

```
example(get_wq)
```

- One variable at one station

```
get_wq(station_id = "FLAB08", date_min = "2011-03-01",
      date_max = "2012-05-01", test_name = "CHLOROPHYLLA-SALINE")
```

- One variable at multiple stations

```
get_wq(station_id = c("FLAB08","FLAB09"), date_min = "2011-03-01",
      date_max = "2012-05-01", test_name = "CHLOROPHYLLA-SALINE")
```

- One variable at a wildcard station

```
get_wq(station_id = c("FLAB0%"), date_min = "2011-03-01",
      date_max = "2012-05-01", test_name = "CHLOROPHYLLA-SALINE")
```

- Multiple variables at multiple stations

```
get_wq(station_id = c("FLAB08","FLAB09"), date_min = "2011-03-01",
      date_max = "2012-05-01", test_name = c("CHLOROPHYLLA-SALINE",
      "SALINITY"))
```

By default, `get_wq` returns a *cleaned output*. First, the cleaning function `clean_wq` converts the raw output from native DBHYDRO *long* format (each piece of data on its own row) to *wide* format (each site x variable combination in its own column) using the reshape2 package (Wickham 2007). Next, the extra columns associated with QA flags, LIMS, and District receiving are removed. Finally, row entries associated with QA field *blanks*, which are used to check on potential sources of contamination, are removed. Setting the `raw` flag to `TRUE` will force `get_wq` to retain information on QA field blanks as well as the other QA fields. An example query that retains this information and the original *long* formatting is shown below.

```
raw_wq <- get_wq(station_id = "FLAB08", date_min = "2011-03-01",
      date_max = "2011-05-01", test_name = "CHLOROPHYLLA-SALINE",
      raw = TRUE)
```

This raw data can then be cleaned using the `clean_wq` function:

```
clean_wq(raw_wq)
```

3.2 Hydrologic data

Hydrologic time series data can be retrieved using the `get_hydro` function. The first task to accomplish prior to running `get_hydro` is to identify one or more dbkeys which correspond to unique site x variable time-series. This can be done before-hand using the `get_dbkey` function, the ArGIS Online Station Map or the DBHYDRO Browser. One useful strategy for finding desired dbkeys is to run the `get_dbkey` function interactively using progressively narrower search terms. For example, suppose we are interested in daily average wind data at Joe Bay but we have no alphanumeric `dbkey`. Initially we could run `get_dbkey` with the `detail.level` set to "summary".

```
get_dbkey(stationid = "JBTS", category = "WEATHER", param = "WNDS",
         detail.level = "summary")
```

Our search returns two results but only one of them has a daily average (DA) measurement frequency. We can verify the remaining attributes of our likely dbkey by setting the `freq` parameter to "DA" and the `detail.level` parameter to "full".

```
get_dbkey(stationid = "JBTS", category = "WEATHER", param = "WNDS",
         freq = "DA", detail.level = "full")
```

This exact dbkey can only be returned reliably by specifying all of the `get_dbkey` parameters applicable to the "WEATHER" category.

```
get_dbkey(stationid = "JBTS", category = "WEATHER", param = "WNDS",
         freq = "DA", stat = "MEAN", recorder = "CR10", agency = "WMD",
         detail.level = "dbkey")
```

Now that we have our dbkey in hand, we can use is as input to `get_hydro`. In addition to a dbkey, we must specify a date range. Dates must be entered in YYYY-MM-DD format (e.g. 2015-02-26).

```
get_hydro(dbkey = "15081",
         date_min = "2013-01-01", date_max = "2013-02-02")
```

Alternatively, we can specify a set of arguments in our call to `get_hydro` that will be passed to `get_dbkey` on-the-fly. Use caution when using this strategy as complex stationid/category/parameter combinations can easily cause errors or return unexpected results. It is good practice to pre-screen your parameter values using `get_dbkey`.

```
get_hydro(date_min = "2013-01-01", date_max = "2013-02-02",
          stationid = "JBTS", category = "WEATHER", param = "WNDS",
          freq = "DA", stat = "MEAN", recorder = "CR10", agency = "WMD")
```

The contents of multiple data streams can be returned by specifying multiple dbkeys or entering on-the-fly `get_dbkey` queries that return multiple dbkeys.

```
get_hydro(dbkey = c("15081", "15069"), date_min = "2013-01-01",
          date_max = "2013-02-02")

get_hydro(date_min = "2013-01-01", date_max = "2013-02-02",
          category = "WEATHER", stationid = c("JBTS", "MBTS"),
          param = "WNDS", freq = "DA", stat = "MEAN")
```

More `get_hydro` examples including queries of other `category` values ("SW", "GW", and "WQ") can be viewed by issuing the following commands from the `R` console:

```
example(get_dbkey)
example(get_hydro)
```

By default, `get_hydro` returns a *cleaned output*. First, the cleaning function `clean_hydro` converts the raw output from native DBHYDRO *long* format (each piece of data on its own row) to *wide* format (each site x variable combination in its own column) using the reshape2 package (Wickham 2007). Next, some extra columns are removed that are associated with measurement location (longitude/latitude), frequency, and QA flags are removed. Setting the `raw` flag to `TRUE` will force `get_hydro` to retain the original formatting and metadata fields. An example query that retains this information and the original *long* formatting is shown below.

```
raw_data <- get_hydro(date_min = "2013-01-01", date_max = "2013-02-02",
          stationid = "JBTS", category = "WEATHER", param = "WNDS",
          freq = "DA", stat = "MEAN", recorder = "CR10", agency = "WMD", raw = TRUE,
clean_hydro(raw_data)
```

4 Appendix

4.1 Test names

There are many test names available in DBHYDRO. A subset of these are detailed in the following table.

Code
AMMONIA-N
CARBON, TOTAL ORGANIC
CHLOROPHYLL-A(LC)
CHLOROPHYLL-B(LC)
CHLOROPHYLLA-SALINE
DISSOLVED OXYGEN
KJELDAHL NITROGEN,TOTAL
NITRATE+NITRITE-N
NITRITE-N
PHEOPHYTIN-A(LC)
PHOSPHATE,ORTHO AS P
PHOSPHATE,TOTAL AS P
SALINITY
SILICA
SP CONDUCTIVITY, FIELD
TEMP
TOTAL NITROGEN
TURBIDITY

4.2 Further reading

See section on URL-based data access in the DBHYDRO Browser User's Guide

References

Hadley Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20, 2007. URL http://www.jstatsoft.org/v21/i12/.