Published on *American Geosciences Institute* (https://www.americangeosciences.org)

The Hydro Network-Linked Data Index

USGS Office of Water Information

Introduction

The Hydro Network-Linked Data Index (NLDI) is a system that can index data to NHDPlus V2 catchments and offers a search service to discover indexed information. Data linked to the NLDI includes active NWIS stream gages, water quality portal sites, and outlets of HUC12 watersheds. The NLDI is a core product of the Open Water Data Initiative and is being developed as an open source project. coordinated through the Advisory Committee on Water Information (ACWI) Subcommittee on Spatial Water Data.
In this blog post, we introduce the basic functions of the NLDI and show how to use it as a data discovery and access tool in R. The first section describes the operations available from the NLDI's Web API. The second section shows how to map NLDI data and how to use the NLDI to discover data to be accessed with the dataRetrieval R- package.
Below, text highlighting is used in six ways:
1) The names of API parameters such as {featureSource}
2) Example values of API parameters such as: USGS-08279500
3) API operation names: ***navigation*** and ***basin***
4) API request names such as: *getDataSources*
5) R functions such as: **readOGR**
6) Other specific strings such as "siteNumber"

The NLDI Web API

The NLDI's Web API follows a loosely RESTful design and is documented with swagger documentation which can be found here. Every request to get data from the NLDI starts from a given network linked feature.

Feature Sources

Available network linked feature sources ({featureSource}s) can be found from the *getDataSources* request. (hint: Click the "try it out" button on the swagger page!) These are the collections of network linked features the NLDI knows about. Think of them as watershed outlets that can be used as a starting point. For this demo, we'll use NWIS Stream Gages as the {featureSource}. As a note for later, {featureSource} here is the same as the {dataSource} described below.

Feature IDs

For now, the particular {featureID} to be accessed from a given {featureSource} needs to be found outside the NLDI, so the *getFeatures* request doesn’t return anything. So let’s use a well known NWIS Streamgage on the Rio Grande at Embudo NM as our starting point.

Indexed Features

We can use the *getRegisteredFeature* request to see this feature. Enter nwissite and USGS-08279500 in the {featureSource} and {featureID}, respectively, in the swagger demo page. You can also see this in your browser at this url: https://cida.usgs.gov/nldi/nwissite/USGS-08279500 The response contains the location of the feature, is in geojson, and looks like:
{ "type": "FeatureCollection", "features": [{ "type": "Feature", "geometry": { "type": "Point", "coordinates": [-105.9639722, 36.20555556] }, "properties": { "source": "nwissite", "sourceName": "NWIS Sites", "identifier": "USGS-08279500", "name": "RIO GRANDE AT EMBUDO, NM", "uri": "https://waterdata.usgs.gov/nwis/inventory?agency_code=USGS&site_no=08279500", "comid": "17864756", "navigation": "https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate" } }] }

Navigation

The ***navigation*** property of the returned feature is a url for the *getNavigationTypes* request. This request provides four ***navigation*** options as shown below. Each of these URLs returns the NHDPlus flowlines for the navigation type.
{ "upstreamMain": "https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/UM", "upstreamTributaries": "https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/UT", "downstreamMain": "https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/DM", "downstreamDiversions": "https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/DD" }

Get Flowlines from Navigation

Each of the URLs found via the *getNavigationTypes* request is a complete *getFlowlines* request. This request has some optional input parameters. The most useful being {distance}, which allows specification of a distance to navigate in km. So, for example, we can use this to retrieve 150km of upstream mainstem flowlines from the NWIS gage 08279500 with a request like:
https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/UM?distance=150
Notice that the flowline goes downstream of the gage because the NLDI is referenced to whole NHDPlus catchments, not to precise network locations.

Get Linked Data from Navigation

Now that we have a {featureSource} = nwissite, a {featureID} = USGS-082795001, the ***navigate*** operation on the feature, and the {navigationMode} = UM with {distance} = 150km, we can use the *getFeatures* request to discover features from any {featureSource} which, in the context of a *getFeatures* request, is called a {dataSource}. Setting the {dataSource} = nwissite, we can see if there are any active NWIS streamgages 150km upstream on the main stem with a request that looks like: https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/UM/nwissite?distance=150 Note that we could enter wqp in place of nwissite after UM here to get water quality portal sites instead of NWIS sites. An example of this is shown later in this post.
Note: Click the black NWIS gage points to see a pop up and link!

Get the Upstream Basin Boundary

So far, we’ve covered four parameters of the NLDI Web API. The two base parameters, {featureSource} and {featureID}, and two that apply to the ***navigate*** option, {navigationMode} and {distance}. In addition to the ***navigate*** option, the NLDI offers a ***basin*** option for any {featureSource}/{featureID}. The *getBasin* operation doesn’t require any additional parameters, so a request to get the basin for our stream gage looks like: https://cida.usgs.gov/nldi/nwissite/USGS-08279500/basin.

NLDI API Summary

There are a few other options available from the NLDI that are not covered here. One, that is coming soon, will make catchment (local incremental NHDPlus catchment) and basin (upstream accumulation) landscape characteristics available. This functionality and data is available but is preliminary and subject to change.
Bringing together all the operations summarized above, we can get:
1) The NWIS site:
https://cida.usgs.gov/nldi/nwissite/USGS-08279500
2) The basin upstream of the site:
https://cida.usgs.gov/nldi/nwissite/USGS-08279500/basin

3) All upstream with tributaries flowlines:
https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/UT
4) The upstream mainstem flowlines:
https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/UM
5) The downstream mainstem flowlines:
https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/DM
6) The water quality observation sites in upstream catchments:
https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/UT/wqp
7) The water quality observations in downstream catchments:
https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/DM/wqp
For QGIS users, you can use the NLDI URLs directly in the “Add Vector Layer” dialogue. The following two screenshots were rendered by loading the data into QGIS, turning on a base map with the OpenLayers Plugin, and applying a little styling to the NLDI layers. No local files needed!

Screenshots of NLDI data loaded into QGIS.

Using the NLDI in R.

Starting from the recent blog post showing how to use National Map basemaps with leaflet in R, we can map some data retrieved from the NLDI in R. In the code below, the leaflet object map created in the National Map basemap blog post was wrapped up into the get_base_map() function called in the code shown below.

```
GetURL <- function(service, host = "basemap.nationalmap.gov") {
sprintf("https://%s/arcgis/services/%s/MapServer/WmsServer", host, service) } get_base_map <- function(options =
leaflet::leafletOptions()) { map <- leaflet::leaflet(options = options) grp <- c("USGS Topo", "USGS Imagery Only", "USGS
Imagery Topo", "USGS Shaded Relief", "Hydrography") att <- paste0("<a href='https://www.usgs.gov/'>", "U.S. Geological
Survey</a> | ", "<a href='https://www.usgs.gov/laws/policies_notices.html'>", "Policies</a>") map <- leaflet::addWMSTiles(map,
GetURL("USGSTopo"), group = grp[1], attribution = att, layers = "0") map <- leaflet::addWMSTiles(map,
GetURL("USGSImageryOnly"), group = grp[2], attribution = att, layers = "0") map <- leaflet::addWMSTiles(map,
GetURL("USGSImageryTopo"), group = grp[3], attribution = att, layers = "0") map <- leaflet::addWMSTiles(map,
GetURL("USGSShadedReliefOnly"), group = grp[4], attribution = att, layers = "0") opt <- leaflet::WMSTileOptions(format =
"image/png", transparent = TRUE) map <- leaflet::addWMSTiles(map, GetURL("USGSHydroCached"), group = grp[5], options
= opt, layers = "0") map <- leaflet::hideGroup(map, grp[5]) opt <- leaflet::layersControlOptions(collapsed = FALSE) map <-
leaflet::addLayersControl(map, baseGroups = grp[1:4], overlayGroups = grp[5], options = opt) }
```

First, we get all our URLs into a list and use **readOGR** from the rgdal package to download and read all the data into spatial data types. The end of this code block creates html for popup text with a web link tag that we’ll use later.

```
nldiURLs <- list(site_data = "https://cida.usgs.gov/nldi/nwissite/USGS-08279500", basin_boundary =
"https://cida.usgs.gov/nldi/nwissite/USGS-08279500/basin", UT = "https://cida.usgs.gov/nldi/nwissite/USGS-
08279500/navigate/UT", UM = "https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/UM", DM =
"https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/DM", UTwqp = "https://cida.usgs.gov/nldi/nwissite/USGS-
08279500/navigate/UT/wqp", DMwqp = "https://cida.usgs.gov/nldi/nwissite/USGS-08279500/navigate/DM/wqp") nldi_data <-
list() for(n in names(nldiURLs)) { nldi_data[n] <- rgdal::readOGR(dsn = nldiURLs[n][[1]], layer = "OGRGeoJSON", verbose =
FALSE) print(paste(n, "is of class", class(nldi_data[n][[1]])[1], "and has", length(nldi_data[n][[1]]), "features")) } ## [1]
"site_data is of class SpatialPointsDataFrame and has 1 features" ## [1] "basin_boundary is of class SpatialPolygonsDataFrame
and has 1 features" ## [1] "UT is of class SpatialLinesDataFrame and has 3371 features" ## [1] "UM is of class
SpatialLinesDataFrame and has 184 features" ## [1] "DM is of class SpatialLinesDataFrame and has 1367 features" ## [1]
"UTwqp is of class SpatialPointsDataFrame and has 1908 features" ## [1] "DMwqp is of class SpatialPointsDataFrame and has
2756 features" UTwqp_html <- paste('<a href="', nldi_data$UTwqp@data$uri, '" target="_blank">',
nldi_data$UTwqp@data$name, '</a>') DMwqp_html <- paste('<a href="', nldi_data$DMwqp@data$uri, '" target="_blank">',
nldi_data$DMwqp@data$name, '</a>')
```

Now that we have all our data, we can use leaflet functions to add the data to a map. First, let’s map just the upstream data. Note that the order we add them determines the order the layers are drawn. You can zoom in on the map shown and click the water quality sites to get a popup containing a link to the site’s landing page.

```
map <- get_base_map() # the function described above. map <- leaflet::addPolygons(map, data=nldi_data$basin_boundary, color
= "black", fill = FALSE, weight = 1, opacity = 1) map <- leaflet::addPolylines(map, data = nldi_data$UT, color = "blue", weight =
1, opacity = 1) map <- leaflet::addPolylines(map, data = nldi_data$UM, color = "blue", weight = 3, opacity = 0.5) map <-
leaflet::addCircleMarkers(map = map, data = nldi_data$UTwqp, radius = 1, color = "black", opacity = .5, fill = FALSE, popup =
UTwqp_html) map <- leaflet::addCircleMarkers(map, data = nldi_data$site_data, radius = 5, color = "red")
```

To complete the picture, we can add the downstream main stem and water quality sites. Now we have an interactive map of all the upstream tributaries, water quality sites, basin boundary, the entire main stem, and water quality sites downstream.

```
map <- leaflet::addPolylines(map, data = nldi_data$DM, color = "blue", weight = 3, opacity = 0.5) map <-
leaflet::addCircleMarkers(map = map, data = nldi_data$DMwqp, radius = 1, color = "black", opacity = .5, fill = FALSE, popup =
DMwqp_html)
```

This final map illustrates a very important detail about the NLDI if you zoom in on the downstream main stem. Notice that the sites are not all **on** the main stem flowpath. When the system indexes data sources that aren’t already indexed to particular reachcodes and measures along those reaches, it links sites (points) by looking at what local catchment polygon the site is in. This means that sites found through navigation may not be **on** the main flowpath of a catchment. In the future, we hope to improve the system such that it would know if indexed data are **on** or **off** the main flowpath of a catchment, but for now users need to be aware of this limitation.

Using the NLDI to discover linked observations data.

The two sources of linked data shown above, nwissite and wqp, are both queryable from the dataRetrieval package. The “siteNumber” input of the dataRetrieval functions that start with “readNWIS” can be found by removing “USGS-” from the “identifier” attribute of features found using nwissite as the {dataSource} input. The following code shows the NLDI identifiers and how to use them with the dataRetrieval function **readNWISdv**.

```
nwis_ids <- as.character(nwis_gages@data$identifier) print(paste("The NLDI ID for:", nwis_gages@data$name, "is", nwis_ids))
## [1] "The NLDI ID for: RIO GRANDE NEAR LOBATOS, CO is USGS-08251500" ## [2] "The NLDI ID for: RIO GRANDE
NEAR CERRO, NM is USGS-08263500" ## [3] "The NLDI ID for: RIO GRANDE BLW TAOS JUNCTION BRIDGE NEAR
TAOS, NM is USGS-08276500" ## [4] "The NLDI ID for: RIO GRANDE AT EMBUDO, NM is USGS-08279500" nwis_ids <-
gsub(pattern = "USGS-", replacement = "", nwis_ids) print(paste(nwis_gages@data$name, "has id", nwis_ids)) ## [1] "RIO
GRANDE NEAR LOBATOS, CO has id 08251500" ## [2] "RIO GRANDE NEAR CERRO, NM has id 08263500" ## [3] "RIO
GRANDE BLW TAOS JUNCTION BRIDGE NEAR TAOS, NM has id 08276500" ## [4] "RIO GRANDE AT EMBUDO, NM
has id 08279500" # Now we can use these IDs with dataRetrieval. dv_data <- dataRetrieval::readNWISdv(siteNumber =
nwis_ids[1], parameterCd = '00060') plot(dv_data$Date, dv_data$X_00060_00003, main = paste("Daily Streamflow for",
nwis_gages@data$name[1]), xlab = "", ylab = "Daily Streamflow (CFS)")
```

Similarly, we can use identifiers returned using wqp as the {dataSource} with the **readWQPqw** function. In this case, the identifiers can be used without modification as shown below. Note that the NLDI query for downstream mainstem found 2756 sites and upstream tributaries found 1908 sites. The query below gets data from just one! The NLDI is used as a spatial pre-filter in the Water Quality Portal user interface, which has a rich set of filter options in addition to network navigation.

```
wqp_site <- list(names = as.character(nldi_data$DMwqp@data$name), ids = as.character(nldi_data$DMwqp@data$identifier))
print(paste(wqp_site$names[1:10], "has id", wqp_site$ids[1:10])) ## [1] "Unm Resaca nr Hidalgo, TX has id USGS-
260708098164501" ## [2] "Rio Grande at Talley Cpgd, BBNP, TX has id USGS-285858103110000" ## [3] "Rio Grande at
Boquillas Crsg, BBNP, TX has id USGS-291119102564400" ## [4] "Lk Amistad Site AMS nr Del Rio, TX has id USGS-
292816101061801" ## [5] "Rio Grande at Soldado Ck, TX (2011 Gain-Loss) has id USGS-294616101394200" ## [6] "JL-49-13-
823 has id USGS-314516106251401" ## [7] "JL-49-13-945 has id USGS-314607106244701" ## [8] "JL-49-14-415 has id USGS-
314930106221201" ## [9] "JL-49-05-630 has id USGS-315659106241101" ## [10] "JL-49-04-106 has id USGS-
315733106364501" wqp_data <- dataRetrieval::readWQPqw(siteNumbers = wqp_site$ids[1:10], parameterCd = "")
print(paste0("Got ", ncol(wqp_data), " samples beween ", min(wqp_data$ActivityStartDate), " and ",
max(wqp_data$ActivityStartDate), " for characteristics: ", paste(unique(wqp_data$CharacteristicName), collapse = ", "))) ## [1]
"Got 65 samples beween 1975-06-02 and 2014-05-21 for characteristics: Oxygen, Hydrogen ion, Temperature, water, Barometric
pressure, Specific conductance, pH, Depth, Hexachlorocyclopentadiene, Isophorone, Nitrobenzene, 1,2-Dichloropropane,
Isopropyl acetate, HCFC-22, HFC-152a, CFC-114, HCFC-123, 1-Methoxy-4-(2-propenyl)benzene, 1,3-Butadiene, Ethylene
glycol monoethyl ether acetate, 2-Propen-1-ol, .alpha.-Terpineol, Butyraldehyde, trans-Crotonaldehyde, Stream flow,
instantaneous, Hardness, Ca, Mg, Hardness, non-carbonate, Sodium adsorption ratio [(Na)/(sq root of 1/2 Ca + Mg)], Sodium,
percent total cations, Total dissolved solids, Ammonia and ammonium, Carbonate, Acetonitrile, Methyl acetate, 1-Butanol, tert-
Butanol, Pentanal, 2-Nitropropane, 4-Methyl-2-pentanol, HCFC-21, 5-Methyl-2-hexanone, Isobutyl acetate, 1-Octanol, 1,2,3,4-
```

Tetrahydronaphthalene, 2,6-Dimethyl-4-heptanone, 1,2,3-Trichloropropane, Chloropicrin, Ethylene dibromide, Methyl tert-butyl ether, N-Nitrosodiethylamine, Bicarbonate, Nitrite, Inorganic nitrogen (nitrate and nitrite), Orthophosphate, Calcium, Magnesium, Sodium, Potassium, Chloride, Sulfate, Fluoride, Silica, Arsenic, Boron, Iron, Strontium, Vanadium, Lithium, 1,1-Dichloropropanone, Butane, Dimethoxymethane, 1,4-Dioxane, 1,3-Dioxolane, Ethyl acetate, Hexane, Pentane, 1,2-Dibromo-3-chloropropane, 1-Chloro-1,1-difluoroethane, 1-Bromo-3-chloropropane-d6, Tetrahydrofuran-d8, Carbon dioxide, Trihalomethanes, 2,4-D-d3, Selenium, Alkalinity, total, Bromide, Deuterium, Oxygen-18, Diuron-d6, cis-Permethrin-13C6, Butachlor ESA, Dimethachlor sulfonic acid, Tebuconazole-d6, Thiobencarb-d10, Metolachlor-d6, Nicosulfuron-d6, Hexazinone-d6, Linuron-d6, Malathion-D10, Diflubenzuron-d4, Carbendazim-d4, Carbofuran-D3, Deethylatrazine-d6, Diazinon-D10, Acetochlor-d11, Alachlor-d13, Carbaryl-d7, 3-Phenoxybenzoic acid-13C6, 3-Phenoxybenzoic acid, Chlorimuron-ethyl, Alachlor ESA, 1rs Cis-Permethrin, Triclopyr, Triallate, Tribufos, Transpermethrin, Terbufos sulfone, Terbufos sulfoxide, Terbufos oxon sulfoxide, Terbufos oxygen analog sulfone, Terbufos, Terbufos oxon, Terbacil, Tebuthiuron TP 109, Tebuthiuron, Tebupirimfos oxon, Tebufenozide, Phostebupirim, 2,3,3-Trichloro-2-propene-1-sulfonic acid (sodium salt), Sulfosulfuron ethyl sulfone, Sulfosulfuron, Sulfentrazone, Sulfometuron methyl, sec-Alachlor oxanilic acid, Siduron, sec-Acetochlor oxanilic acid, Pyriproxyfen, Pyridaben, Propoxur, Pymetrozine, Propazine, Propargite, Profenofos, Phthalazinone, Phorate sulfoxide, Phorate sulfone, Phorate oxygen analog sulfone, Phorate oxon sulfoxide, Phorate, Phorate O.A., Oxamyl oxime, Paraoxon, Oryzalin, Oxamyl, Orthosulfamuron, 2-Hydroxy-4-isopropylamino-6-amino-s-triazine, 2-Hydroxyatrazine, Omethoate, O-Ethyl S-methyl S-propyl phosphorodithioate, O-Ethyl S-propyl phosphorothioate, 2-Hydroxy-6-ethylamino-4-amino-s-triazine, Novaluron, Naled, Metribuzin DK, Metolachlor ESA, Metribuzin, Metolachlor OA, Methyl paraoxon, Metolachlor hydroxy morpholinone, Methomyl oxime, Methoxyfenozide, MCPA, Methamidophos, Methomyl, Linuron, Lactofen, Isoxaflutole, Hydroxysimazine, Imazamox, Indoxacarb, Hydroxyphthalazinone, Hydroxytebuthiuron, Hydroxymetolachlor, Hydroxyfluometuron, Hydroxy didemethyl fluometuron, Hydroxy monodemethyl fluometuron, Hydroxydiazinon, Hydroxyalachlor, Hydroxyacetochlor, Fluometuron, 2-(1-Hydroxyethyl)-6-methylaniline, Fipronil amide, Fipronil sulfonate, Flubendiamide, Fentin, Fenamiphos sulfoxide, Fenbutatin-oxide, Fenamiphos Sulfone, Fenamiphos, O-Ethyl O-methyl S-propyl phosphorothioate, Etoxazole, Ethoprop, 2-[(2-Ethyl-6-methylphenyl)amino]-1-propanol, Disulfoton sulfone, Oxydisulfoton, Demethyl fluometuron, Disulfoton oxon sulfone, Demeton-S, Disulfoton oxon sulfoxide, Dimethenamid ESA, Dimethenamid sulfinylacetic acid, Dimethenamid oxanilic acid, Diflufenzopyr, Benzenepropanenitrile, .alpha.-(cyclopropylcarbonyl)- 2-(methylsulfonyl)-.beta.- oxo-4-(trifluoromethyl)-, Dimethenamid, Diflubenzuron, Didemethyl tebuthiuron, Dichlorvos, Dicrotophos, Dicamba, Metribuzin DA, Metribuzin DADK, Desulfinylfipronil amide, Desmethylnorflurazon, Deisopropyl prometryn, Deiodo flubendiamide, Dechlorometolachlor, Dechlorofipronil, cis-Cyhalothric acid, Chlorthal-Monomethyl, 2-Chloro-4-isopropylamino-6-amino-s-triazine, Desisopropyl atrazine, Chlorosulfonamide acid, Carboxy molinate, Carbendazim, 2-Chloro-4,6-diamino-s-triazine, Bromoxynil, Butralin, Bentazon, Bromacil, Ametryn, Asulam, Aldicarb sulfoxide, Aldicarb sulfone, Alachlor sulfinylacetic acid, Aldicarb, Alachlor oxanilic acid, Acetochlor sulfinylacetic acid, 2-Chloro-2',6'-diethylacetanilide, Acetochlor OA, Acetochlor ESA, 2-Chloro-N-(2-ethyl-6-methylphenyl)acetamide, Acetochlor, 4-Hydroxy molinate, Acephate, 4-Chlorobenzylmethyl sulfoxide, 4-(Hydroxymethyl) pendimethalin, 2-Isopropyl-6-methyl-4-pyrimidinol, 3-Hydroxycarbofuran, 2-Aminobenzimidazole, 2-Amino-N-isopropylbenzamide, Metalaxyl, 1H-1,2,4-Triazole, 2,4-D, Imidacloprid, Malaoxon, 4-Hydroxychlorothalonil, 1-(3,4-dichlorophenyl)-3-methyl urea, Diazoxon, Chlorpyrifos O.A., 3,4-Dichlorophenylurea, Azinphos-methyl oxygen analog, Prometon, Pronamide, Kresoxim-methyl, Norflurazon, Disulfoton, Famoxadone, Trifloxystrobin, Tetraconazole, Chlorsulfuron, Flumetsulam, Halosulfuron-methyl, Imazaquin, Imazethapyr, Nicosulfuron, Prosulfuron, Alachlor, Azinphos-methyl, Bifenthrin, Butylate, Carbaryl, Carbofuran, Chlorpyrifos, Diazinon, S-Ethyl dipropylthiocarbamate, Fonofos, Hexazinone, Malathion, Methidathion, Metolachlor, Molinate, Oxyfluorfen, Pendimethalin, Piperonyl butoxide, Prometryn, Simazine, Thiobencarb, Azoxystrobin, Cyanazine, Dimethoate, Diuron, Fipronil, 1H-Pyrazole-3-carbonitrile, 5-amino-1-[2,6-dichloro-4-(trifluoromethyl)phenyl]-4-(trifluoromethyl)-, 5-Amino-1-[2,6-dichloro-4-(trifluoromethyl)phenyl]-4-[(trifluoromethyl)thio]pyrazole-3-carbonitrile, Fipronil Sulfone, Metconazole, Myclobutanil, Propanil, Propiconazole, Pyraclostrobin, Tebuconazole, Terbuthylazine, Barium, Beryllium, Cadmium, Chromium, Cobalt, Copper, Lead, Manganese, Thallium, Molybdenum, Nickel, Silver, Zinc, Antimony, Aluminum, Gross-Uranium, Perchlorate, Nitrate, Organic Nitrogen, Radium-224, Radium-226, Lead-210, Polonium-210, Arsenate, Monomethylarsonate, Cacodylic acid, Alpha particle, Beta particle, Arsenite, Radium-228, Temperature, air, deg C, Turbidity, Dichlorobromomethane, Carbon tetrachloride, 1,2-Dichloroethane, Tribromomethane, Chlorodibromomethane, Chloroform, Toluene, Benzene, Chlorobenzene, Ethylbenzene, Methyl bromide, Methylene chloride, Tetrachloroethene, 1,1-Dichloroethane, 1,1-Dichloroethylene, 1,1,1-Trichloroethane, 1,1,2-Trichloroethane, o-Dichlorobenzene, Nitrogen, mixed forms (NH3), (NH4), organic, (NO2) and (NO3), Kjeldahl nitrogen, Phosphorus, trans-1,2-Dichloroethylene, 1,2,4-Trichlorobenzene, p-Dichlorobenzene, Naphthalene, trans-1,3-Dichloropropene, cis-1,3-Dichloropropene, Vinyl chloride, Trichloroethene (TCE), Isobutyl alcohol-d6, Carbon disulfide, cis-1,2-Dichloroethylene, Styrene, o-Xylene, 1,2,4-Trimethylbenzene, n-Propylbenzene, Halon 1011, sec-Butylbenzene, 1,1,1,2-Tetrachloroethane, m,p-Xylene, 1,2-Dichloroethane-d4, Chloromethane, p-Bromofluorobenzene, Toluene-d8, Organic carbon, Radon-222, Escherichia coli, Total Coliform, Enterococcus, Coliphage, Male Specific (F+) all Groups, cis-Androsterone-2,2,3,4,4-d5, Carbon-14, Carbon, isotope of mass 13, Tritium"

Summary

In this blog post, we summarized the NLDI's Web API through links to the system's Swagger documentation. The primary API parameters, {featureSource} and {featureID}, were described. Two functions that operate with any {featureID}, ***navigation*** (and it's optional {distance} parameter) and ***basin*** were demonstrated. The the ***navigation*** function's {dataSource} parameter, which can be any {featureSource}, was shown by retrieving NWIS (nwissite) and WQP (wqp) sites upstream and downstream of an NWIS site.

Building on the recent blog post showing how to use National Map basemaps with leaflet in R, downloading, parsing, and mapping NLDI data in R was demonstrated. This basic demonstration can be extended by using different NLDI inputs and there are many operations that are supported by the spatial data formats returned by the rgdal packge function **readOGR**.

The post finishes by showing how to use sites found with the NLDI to download data from the National Water Information System and Water Quality portal. The potential for extending this use of the NLDI is vast. As more feature/data sources are indexed and the system evolves, the NLDI should serve as a major new discovery service for many sources of observed and modeled data.

The NLDI is an exciting new service that is being implemented in an incremental and agile development process. Given that, the API will expand and new versions of the API may have somewhat different design. The intention is to keep the Web API described here the same, only changing it by introducing a version identifier as an API parameter. If you found this useful and plan on using the NLDI as a dependency in a project of application, we would greatly appreciate hearing about your use case and can answer any questions you have while implementing your application.

Please email dblodgett@usgs.gov with questions and feedback.