

# Package ‘wqTools’

September 20, 2021

**Title** A Collection of R Tools for Utah Division of Water Quality

**Version** 0.0.0.9000

**Authors**

Jake Vander Laan, Utah Division of Water Quality, [jvander@utah.gov](mailto:jvander@utah.gov) [aut, cre], Elise Hinman, Utah Division of Water Quality, [ehinman@utah.gov](mailto:ehinman@utah.gov) [aut]

**Description** This package is intended to house R tools developed and for use by UDWQ staff as well as support the UDWQ irTools package.

**Depends** R ( $\geq$  3.5.0),

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.2

**Roxygen** list(markdown = TRUE)

**Imports** shinybusy,  
data.table,  
dplyr,  
glue,  
leaflet,  
RColorBrewer,  
akima,  
jsonlite,  
leaflet.extras,  
lubridate,  
mapedit,  
plyr,  
rLakeAnalyzer,  
reshape2,  
sf,  
shiny,  
shinyjs,  
shinyBS,  
shinycssloaders,  
tidyr,  
plotly,  
miniUI

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

## R topics documented:

addMapResetButton . . . . .	2
assignAUs . . . . .	3
assignHUCs . . . . .	3
assignPolys . . . . .	4
assignUses . . . . .	5
au_poly . . . . .	5
au_poly_1820 . . . . .	6
baseMap . . . . .	6
buildLDC . . . . .	7
buildMap . . . . .	8
bu_poly . . . . .	9
calcAICrit . . . . .	9
calcTSI . . . . .	10
convertUnits . . . . .	11
county_poly . . . . .	12
downloadWQP . . . . .	12
facToNum . . . . .	13
findSites . . . . .	13
hnnc_poly . . . . .	14
huc12_poly . . . . .	14
huc8_12_poly . . . . .	14
huc8_poly . . . . .	15
muni_poly . . . . .	15
profileHeatMap . . . . .	15
profilePlot . . . . .	16
readAttains . . . . .	17
readAWQMS . . . . .	18
readECHO_ec . . . . .	19
readECHO_fac . . . . .	20
readWQP . . . . .	21
readWQPbySite . . . . .	22
selectPolys . . . . .	23
ss_poly . . . . .	24
ul_trophic . . . . .	24
ut_poly . . . . .	25
wmu_poly . . . . .	25
wqDD . . . . .	25
<b>Index</b>	<b>26</b>

---

addMapResetButton	<i>Add a map reset button to a Leaflet map object</i>
-------------------	---

---

### Description

Add a map reset button to a Leaflet map object

### Usage

```
addMapResetButton(leaf)
```

**Arguments**

leaf                      A Leaflet map object

**Examples**

```
reset_map=wqTools::buildMap() %>% addMapResetButton()
```

---

assignAUs	<i>Assign Utah assessment units to sites</i>
-----------	--

---

**Description**

This function assigns assessment units to water quality portal type site objects (or data with site information attached). This can be done before or after assigning beneficial uses.

**Usage**

```
assignAUs(data, lat = "LatitudeMeasure", long = "LongitudeMeasure")
```

**Arguments**

data                      Input dataset. Must include latitude & longitude columns.  
 lat                        Name of latitude column. Default matches WQP objects.  
 long                      Name of longitude column. Default matches WQP objects.

**Value**

Returns the input data frame with assessment unit information appended.

**Examples**

```
# Read a couple of sites (Mantua Reservoir and a tributary)
sites=readWQP(type="sites", siteid=c("UTAHDWQ_WQX-4900440", "UTAHDWQ_WQX-4900470"))
sites_AUs=assignAUs(sites)
```

---

assignHUCs	<i>Assign HUC 8 &amp; 12 values to sites</i>
------------	--

---

**Description**

This function assigns HUC 8 & 12 values to water quality portal type site objects (or data with site information attached).

**Usage**

```
assignHUCs(data, lat = "LatitudeMeasure", long = "LongitudeMeasure")
```

**Arguments**

<code>data</code>	Input dataset. Must include latitude & longitude columns.
<code>lat</code>	Name of latitude column. Default matches WQP objects.
<code>long</code>	Name of longitude column. Default matches WQP objects.

**Value**

Returns the input data frame with HUC 8 & 12 information appended.

**Examples**

```
# Read a couple of sites from Mantua Reservoir
sites=readWQP(type="sites", siteid=c("UTAHDWQ_WQX-4900440", "UTAHDWQ_WQX-4900470"))
sites_HUCs=assignHUCs(sites)
```

---

<code>assignPolys</code>	<i>Assign spatial polygons to sites</i>
--------------------------	---

---

**Description**

This function assigns spatial polygon data to water quality portal type site objects (or data with site information attached).

**Usage**

```
assignPolys(
  data,
  polygon,
  lat = "LatitudeMeasure",
  long = "LongitudeMeasure",
  columns
)
```

**Arguments**

<code>data</code>	Input dataset. Must include latitude & longitude columns.
<code>polygon</code>	Polygon to assign information to sites. Must be an sf type polygon shapefile, WGS84 projection recommended.
<code>lat</code>	Name of latitude column. Default matches WQP objects.
<code>long</code>	Name of longitude column. Default matches WQP objects.
<code>columns</code>	Vector of columns to retain from polygon object. If columns not specified, function will retain all columns.

**Value**

Returns the input data frame with assessment unit information appended.

**Examples**

```
# Read a couple of sites (Mantua Reservoir and a tributary)
sites=readWQP(type="sites", siteid=c("UTAHDWQ_WQX-4900440", "UTAHDWQ_WQX-4900470"))
sites_AUs=assignPolys(sites, wqTools::au_poly)
sites_BUs=assignPolys(sites, wqTools::bu_poly)
```

---

assignUses

*Assign Utah beneficial use classes to sites*


---

### Description

This function assigns beneficial use classes to water quality portal type site objects (or data with site information attached).

### Usage

```
assignUses(
  data,
  lat = "LatitudeMeasure",
  long = "LongitudeMeasure",
  flatten = FALSE
)
```

### Arguments

data	Input dataset. Must include latitude & longitude columns.
lat	Name of latitude column. Default matches WQP objects.
long	Name of longitude column. Default matches WQP objects.
flatten	Logical. If FALSE (default), maintain use categories as single comma separated column. If TRUE, use column and data are flattened by expanded use column.

### Examples

```
# Read a couple of sites (Mantua Reservoir and a tributary)
sites=readWQP(type="sites", siteid=c("UTAHDWQ_WQX-4900440", "UTAHDWQ_WQX-4900470"))
sites_uses=assignUses(sites)
sites_uses_flat=assignUses(sites, flatten=TRUE)
```

---

au\_poly

*Utah's IR-specific assessment unit polygons*


---

### Description

Polygons containing assessment unit designations for the current IR cycle.

### Usage

```
au_poly
```

### Format

An sf type polygon shapefile

---

au_poly_1820	<i>IR-specific assessment unit polygons as used in the 2018/2020 IR</i>
--------------	---

---

**Description**

Polygons containing assessment unit designations as used in the 2018/2020 IR.

**Usage**

```
au_poly_1820
```

**Format**

An sf type polygon shapefile

---

baseMap	<i>Water quality base map</i>
---------	-------------------------------

---

**Description**

This function generates a base map for use in water quality analyses. It includes useful base layers, Utah assessment unit polygons, and Utah beneficial use designations.

**Usage**

```
baseMap(layers_control = TRUE)
```

**Arguments**

**layers\_control** Logical. If TRUE (default), a layers control widget is added to the map. Set to FALSE to generate a custom layers control widget.

**Value**

A leaflet map object

---

**buildLDC***Calculate parameter loading capacity*

---

**Description**

Uses flow, parameter concentration, and standard values to determine observed loading and loading capacity

**Usage**

```
buildLDC(  
  x,  
  flow,  
  date,  
  value,  
  location,  
  parameter,  
  crit,  
  loading_units,  
  mos = 0.1,  
  cf,  
  plot_it = TRUE  
)
```

**Arguments**

<b>x</b>	A data frame containing columns indicating flow and parameter concentration for given dates (required for plotting: location and parameter name). Note that flow data for which no parameter values exist are still used to construct LDC plot.
<b>flow</b>	String. Column name containing flow data.
<b>date</b>	String. Column name containing date data.
<b>value</b>	String. Column name containing parameter concentration data.
<b>location</b>	String. Column name containing location name(s).
<b>parameter</b>	String. Column name containing parameter name.
<b>crit</b>	Numeric. Represents the standard criterion (in same concentration units as parameter) against which observed loadings are compared.
<b>loading_units</b>	String. Indicates the loading units (amount/time) to be plotted on the y-axis of the LDC (if plot_it = TRUE).
<b>mos</b>	Numeric. A decimal representing the percent margin of safety to apply to the loading capacity for management decisions.
<b>cf</b>	Numeric. A value representing the correction factor linking flow and parameter concentration to desired unit (load per time).
<b>plot_it</b>	Logical. If TRUE, plots observed capacity and loading capacity in one figure in a load duration curve framework.

**Value**

A data frame containing original columns supplied to function plus observed loading, loading capacity, loading capacity plus margin of safety, season, and flow percentile.

---

**buildMap***Build a site map of WQP sites or ECHO facilities*

---

**Description**

Build a map of sample sites, facilities, or both. Map includes sites, beneficial use and assessment unit polygons, and satellite and topo baselayers. This is designed to work with column names as extracted from WQP or AWQMS & ECHO via `udwqTools` functions `readWQP()`, `readAWQMS()` and `readECHO_fac()`. Map will launch in default browser (or R-Studio's browser if using R-Studio). Site and assessment unit features are searchable by identifier and name via the search button on the left side of the map. The most recently turned on layer is "on top" of the map. Only features on top will show their pop-up on click.

**Usage**

```
buildMap(
  fac,
  sites,
  au_poly,
  bu_poly,
  ss_poly,
  search = c("sites", "aus"),
  plot_polys = TRUE,
  dragging = T,
  ...
)
```

**Arguments**

<code>fac</code>	Facility locations queried via <code>readECHO_fac</code> .
<code>sites</code>	Site locations queried via <code>readWQP(type="sites")</code> . May also be a data file with WQP site information merged to it.
<code>au_poly</code>	Optional. Polygon file to be mapped as assessment units. Useful for mapping a subset of specific assessment units. If missing, the default state wide AU polygon is used.
<code>bu_poly</code>	Optional. Polygon file to be mapped as beneficial uses. Useful for mapping a subset of beneficial uses. If missing, the default state wide uses polygon is used.
<code>ss_poly</code>	Optional. Polygon file to be mapped as site specific standards. Useful for mapping a subset of ss polygons. If missing, the default state wide ss polygon is used.
<code>search</code>	Vector of objects to be made searchable. One or both of "sites" and "aus". Defaults to <code>c("sites", "aus")</code> . Any other inputs are ignored.

**Examples**

```
# Read sites & facility locations
jr_sites=readWQP(type="sites",
siteid=c("UTAHDWQ_WQX-4994100", "UTAHDWQ_WQX-4994120", "UTAHDWQ_WQX-4991860",
```



```

    "UTAHDWQ_WQX-4994190", "UTAHDWQ_WQX-4994172", "UTAHDWQ_WQX-4994090",
    "UTAHDWQ_WQX-4992890", "UTAHDWQ_WQX-4992880", "UTAHDWQ_WQX-4992480",
    "UTAHDWQ_WQX-4992055", "UTAHDWQ_WQX-4991940", "UTAHDWQ_WQX-4991880"))
jr_fac=readECHO_fac(p_pid=c("UT0024392", "UT0024384", "UT0025852", "UT0021725"))
#Build some maps
map1=buildMap(sites=jr_sites, fac=jr_fac) #define new object for use later
map1 #call generated map object to launch in browser
buildMap(sites=jr_sites) #just sites, launch w/o generating map object in workspace
buildMap(fac=jr_fac) #just facilities
buildMap() #Build an empty map w/ just AU, BU, and SS std polys
#html maps can be saved via htmlwidgets package saveWidget(map1, file="your/path/map1.html")

```

bu\_poly

*Utah's beneficial use polygon shapes*

## Description

Polygons containing beneficial use designations and water body type information. Used to assign uses or standards to site locations.

## Usage

```
bu_poly
```

## Format

An sf type polygon shapefile

calcAlCrit

*Calculate freshwater chronic and acute aluminum criteria*

## Description

Calculates chronic and acute aluminum criteria according to the EPA 2018 update as documented here: <https://www.epa.gov/wqc/aquatic-life-criteria-aluminum#2018>. The function is a generalized wrapper for the criterion calculator provided within Aluminum Criteria Calculator R Code and Data V2.0 (Zip). Which allows the user to directly input hardness, pH, and DOC data as a data frame and specify column names as appropriate. The function code also includes a dput() version of the toxicity tables, which, although annoying to look at, allows the tables to follow the function without needing to read separate CSVs to the workspace. This is the final object, all, as on line 71 of the original calculator. This could be stored separately in package data if function is included in a package.

## Usage

```

calcAlCrit(
  x,
  pH = "pH",
  hardness = "Hardness_mgL",
  DOC = "DOC_mgL",
  inc_ranks = T,
  tox_table = NULL
)

```

**Arguments**

x	Input dataset including pH, DOC, and hardness in a wide format. DOC and hardness must be in mg/L
pH	Column name (in quotes) for pH data.
hardness	Column name (in quotes) for hardness data.
DOC	Column name (in quotes) for DOC data.
inc_ranks	If TRUE (default), include genus toxicity ranking information in output.
tox_table	Optional. If desired, specify custom toxicity table for calculations. If not provided, default tables will be used.

**Examples**

```
# Build some test data
pH=seq(6.5, 8.4, 0.1)
Hardness_mgL=seq(25, 500, 25)
DOC_mgL=seq(0.1, 2, 0.1)
sampID=paste0(rep('samp', 20), seq(1,20,1))
test_data=data.frame(sampID, pH, Hardness_mgL, DOC_mgL)
head(test_data)
# Apply function to test_data
criteria=calcAlCrit(test_data, 'pH', 'Hardness_mgL', 'DOC_mgL')
```

calcTSI

*Calculate TSI values from input data***Description**

This function calculates TSI values according to Utah's IR methods from input data containing values for of chlorophyll, total phosphorus, and secchi disk depth. Note that inputs for these parameters must be specified in units of ug/L, mg/L, and meters, respectively.

**Usage**

```
calcTSI(
  x,
  in_format = "wide",
  chl_ugL = "Chlorophyll a",
  TP_mgL = "Phosphate-phosphorus",
  SD_m = "Depth, Secchi disk depth",
  value_var = "ResultMeasureValue",
  param_var = "CharacteristicName"
)
```

**Arguments**

x	Input dataset
in_format	One of "wide" or "flat" to specify data input format. Note that only wide format inputs are currently supported.
chl_ugL	Name of chlorophyll-a variable in ug/L
TP_mgL	Name of total phosphorus variable in mg/L
SD_m	Name of secchi disk depth variable in m

## Examples

```
data(ul_trophic)
head(ul_trophic)
tsi=calcTSI(ul_trophic,chl_uGL="ChlA",TP_mgL="Phosphate.phosphorus.Total",SD_m="Depth.Secchi.disk.depth")
head(tsi)
plot(TSIchl~ChlA,tsi)
```

---

convertUnits	<i>Convert data to target units</i>
--------------	-------------------------------------

---

## Description

This converts flat data to target units as defined in the dataset. It creates three new columns in your dataset: 1. The converted unit value, 3. the conversion factor used, & 4. a recommended acceptance/rejection based on pairs of input & target units. All original columns are retained.

## Usage

```
convertUnits(
  x,
  input_units,
  target_units = "target_unit",
  value_var,
  conv_val_col = "converted_value"
)
```

## Arguments

x	Input dataset.
value_var	Column name of x containing result values (in quotes).
conv_val_col	Name of new values column for converted values (in quotes).
input_unit	Column name of x with units to convert from (in quotes).
target_unit	Column name of x with units to convert to (in quotes).

## Examples

```
# Make example data
value=c(abs(rnorm(5,1, 0.1)), rnorm(5, 1000, 100), rnorm(5, 20, 1))
raw_units=c(rep('mg/l',5), rep('ug/L', 5), rep('deg C',5))
preferred_units=c(rep('mg/L', 10), rep('deg C',5))
data=data.frame(value,raw_units,preferred_units)
data

# Unit conversion
conv_data=convertUnits(data, input_units='raw_units', target_units='preferred_units', value_var='value', conv_val_col='conv_val_col')
conv_data
```

---

county_poly	<i>Utah's Counties</i>
-------------	------------------------

---

**Description**

Polygons containing county boundaries from gis.utah.gov.

**Usage**

```
county_poly
```

**Format**

An sf type polygon shapefile

---

downloadWQP	<i>Download water quality, station, and other data from USEPA Water Quality Portal</i>
-------------	--

---

### Description

Download data from EPA's Water Quality Portal (WQP) as .csv files to outfile\_path folder.

### Usage

```
downloadWQP(
  outfile_path = getwd(),
  retrieve = c("narrowresult", "activity", "sites", "detquantlim"),
  siteType = c("Aggregate surface-water-use", "Lake, Reservoir, Impoundment", "Spring",
    "Stream"),
  statecode = "US:49",
  zip = FALSE,
  unzip = FALSE,
  ...
)
```

### Arguments

outfile_path	Path for file outputs. Defaults to current working directory.
retrieve	Vector of data type names to retrieve from WQP. One or more of: "result", "narrowresult", "activity", "sites", "detquantlim". Defaults to query narrowresult, activity, sites, and detquantlim.
zip	Logical. If FALSE (default) files downloaded as zip folders are not automatically unzipped. If TRUE, the function will unzip all downloaded zip files. Only used if zip==TRUE.
...	Other arguments to be passed to readWQP when generating a query URL.
start_date	Query start date. "mm/dd/yyyy" format.
end_date	Query end date. "mm/dd/yyyy" format.

**Value**

Exports .csv files for all selected data types during selected date period in specified output path.

**Examples**

```
# Read 2018 download narrow result & sites
downloadWQP(outfile_path='C:\\Your\\Folder\\Path', start_date="01/01/2018", end_date="12/31/2018", retrieve=
```

---

facToNum	<i>Convert factors to numeric equivalents</i>
----------	---

---

**Description**

Converts input object to number if class=="factor". If class!="factor", input object is returned un-altered.

**Usage**

```
facToNum(x)
```

**Arguments**

x                      Input vector object

---

findSites	<i>Interactively find and select water quality sites</i>
-----------	--

---

**Description**

A Shiny gadget designed for interactive selection of water quality sites including permit locations, ambient monitoring locations, and streamflow gauges. This is intended to help users find sites for querying data from ECHO, WQP, or NWIS.

**Usage**

```
findSites()
```

**Value**

A dataframe of site information

---

hnnc_poly	<i>Utah's headwater numeric nutrient criteria polygons</i>
-----------	--

---

**Description**

A spatial polygons representation of Utah's headwater numeric nutrient criteria.

**Usage**

hnnc\_poly

**Format**

An sf type polygon shapefile

---

huc12_poly	<i>HUC 12 polygons for the state of Utah</i>
------------	--

---

**Description**

HUC 12 polygons for the state of Utah.

**Usage**

huc12\_poly

**Format**

An sf type polygon shapefile

---

huc8_12_poly	<i>Unioned HUC 8 &amp; 12 polygons for the state of Utah</i>
--------------	--

---

**Description**

Unioned HUC 8 & 12 polygons for the state of Utah.

**Usage**

huc8\_12\_poly

**Format**

An sf type polygon shapefile

---

huc8_poly	<i>HUC 8 polygons for the state of Utah</i>
-----------	---

---

**Description**

HUC 8 polygons for the state of Utah.

**Usage**

```
huc8_poly
```

**Format**

An sf type polygon shapefile

---

muni_poly	<i>Utah's Municipalities</i>
-----------	------------------------------

---

**Description**

Polygons containing municipal boundaries from [gis.utah.gov](http://gis.utah.gov).

**Usage**

```
muni_poly
```

**Format**

An sf type polygon shapefile

---

profileHeatMap	<i>Profile heat map plot</i>
----------------	------------------------------

---

**Description**

Plots a lake profile heatmap for a single site and parameter.

**Usage**

```
profileHeatMap(
  data,
  parameter,
  depth = "Depth_m",
  param_units,
  depth_units = "m",
  date = "ActivityStartDate",
  show_dates = TRUE,
  min_date = min(data[, date], na.rm = T),
  max_date = max(data[, date], na.rm = T),
  param_lab = "pH",
  criteria
)
```

**Arguments**

<code>data</code>	Lake profile data (wide format)
<code>parameter</code>	Column name for parameter to be used as z-values.
<code>depth</code>	Column name for depth column.
<code>param_units</code>	Character. Parameter units. Used to plot build label.
<code>depth_units</code>	Character. Depth units. Used to plot build label.
<code>date</code>	Date column name. Must be in 'YYYY-mm-dd' format.
<code>show_dates</code>	Logical. If TRUE (default), show individual profile dates on plot x-axis.
<code>min_date</code>	Minimum plot date. 'YYYY-mm-dd' format.
<code>max_date</code>	Maximum plot date. 'YYYY-mm-dd' format.
<code>param_lab</code>	Character. Label to be used for parameter name. Used to build plot label.
<code>criteria</code>	Vector of criteria values to be used as contours on heatmap plot. If not specified, contours from 0-30 at increments of 5 are drawn.

---

profilePlot

---

*Plot an individual lake profile*


---

**Description**

Plots an individual lake profile provided in long data format. If provided, dashed lines representing water quality criteria are also plotted. Default arguments are set to take data from the water quality portal, however, they may be updated as needed to reflect different data sources.

**Usage**

```
profilePlot(
  data,
  parameter = "CharacteristicName",
  units = "ResultMeasure.MeasureUnitCode",
  depth = "Depth, data-logger (ported)",
  do = "Dissolved oxygen (DO)",
  temp = "Temperature, water",
  pH = "pH",
  value_var = "ResultMeasureValue",
  line_no = "DataLoggerLine",
  do_crit,
  temp_crit,
  pH_crit
)
```



**Arguments**

data	Lake profile data (long format)
parameter	Column name containing parameter names.
units	Column name containing data units.
depth	Name of depth variable in input data.
do	Name of dissolved oxygen variable in input data.
temp	Name of temperature variable in input data.
pH	Name of pH variable in input data.
value_var	Column name of value variable.
line_no	Column name containing line number.
do_crit	Optional. Dissolved oxygen criterion to display on plot.
temp_crit	Optional. Temperature criterion to display on plot.
pH_crit	Optional. Vector of two pH criteria to display on plot.

**Examples**

```
# Read in some profile data
nr=readWQP(type="narrowresult", siteid="UTAHDWQ_WQX-4938550", print=F)
act=readWQP(type="activity", siteid="UTAHDWQ_WQX-4938550", print=F)
nr_act=merge(nr, act, all.x=T)
profiles=nr_act[!is.na(nr_act$DataLoggerLine),] #Subset to profile data
table(droplevels(profiles$ActivityIdentifier)) #Find activity IDs associated w/ profiles
profilePlot(subset(profiles,ActivityIdentifier=="UTAHDWQ_WQX-BORFG051909-4938550-0519-Pr-F"))
profilePlot(subset(profiles,ActivityIdentifier=="UTAHDWQ_WQX-LC082807-210255-PR3855083007"), do_crit=4, tem
```

readAttains

*Read data from EPA's ATAINS database***Description**

This function pulls information from EPA ATAINS database based on submitted arguments via ATAINS web service. Any ATAINS web service compatible argument can be submitted to this function. Depending on type, at least one argument may be required. See <https://www.epa.gov/waterdata/how-access-and-use-attains-web-services> for a list of required and compatible arguments. The function is essentially an ATAINS specific wrapper for `jsonlite::fromJSON`. It generates the appropriate web path to connect to ATAINS web service and converts JSON to R object.

**Usage**

```
readAttains(type = "assessments", stateCode = NULL, ...)
```

**Arguments**

type	ATTAINS data type to read. One of: "assessmentUnits", "assessments", or "actions".
stateCode	Two letter state code
...	Additional arguments to be passed to ATAINS web service path.

## Value

Returns a data.frame or list of queried ATTAINS data. Note that some results may contain additional lists of results that can be further flattened.

## Examples

```
#Read Utah assessment unit information
UT_AUs=readAttains(type="assessmentUnits", stateCode="UT")

#Read Utah actions
UT_actions=readAttains(type="actions", stateCode="UT")

#Read Utah assessments
UT_assessments=readAttains(type="assessments", stateCode="UT", reportingCycle=2020)
```

## Examples

```
# Read a couple of sites
sites=readAWQMS(type='sites', MonitoringLocationIdentifiersCsv=c("4900440","4900460"))
# Make a map of those sites & assign assessment units & uses
buildMap(sites=sites)
sites_aus=assignAUs(sites)
sites_aus_uses=assignUses(sites_aus)
sites_aus_uses

# Read all results from those sites (2016-2018)
results=readAWQMS(type='results', MonitoringLocationIdentifiersCsv=c("4900440","4900460"), start_date="01/01/2016", end_date="12/31/2018")

# Read a subset of those results (just two parameters, 2016-2018)
results2=readAWQMS(type='results', MonitoringLocationIdentifiersCsv=c("4900440","4900460"), CharacteristicCodes=c("TDS","pH"), start_date="01/01/2016", end_date="12/31/2018")

# Read Utah DWQ projects
projects=readAWQMS(type="projects", OrganizationIdentifiersCsv="UTAHDWQ_WQX")
# Read state-wide TDS
ut_tds=readAWQMS(type='results', start_date="01/01/2016", end_date="12/31/2018", StateCode="UT", CharacteristicCodes=c("TDS"))
```

---

readECHO\_ec

*Read effluent chart data from EPA ECHO webservices*


---

## Description

This function extracts effluent chart data from EPA ECHO for multiple stations & combinations of parameters. All arguments are optional except p.id. At least one p.id must be specified.

## Usage

```
readECHO_ec(
  ...,
  print = TRUE,
  stringsAsFactors = default.stringsAsFactors(),
  progress = TRUE
)
```

## Arguments

...	additional arguments to be passed to ECHO query path. See optional arguments for effluent chart data reads.
print	Logical. If TRUE (default), print summary table of facilities & parameters returned.
stringsAsFactors	Logical. Passed to read.csv. See ?read.csv for more information.
p.id	Permitted facility ID. Either a single text value (in quotes) or a vector of text values.
parameter_code	Parameter code. Either a single text value (in quotes) or a vector of text values.
start_date	Query start date in "mm/dd/yyyy" format.
end_date	Query end date in "mm/dd/yyyy" format.

**Value**

A flat data frame of EPA ECHO effluent chart data

**Examples**

```
#Extract effluent chart data for facility UT0025241, all outfalls
UT0025241_ec=readECHO_ec(p_id="UT0025241", start_date="01/01/2010", end_date="01/15/2019")
head(UT0025241_ec)

# Extract effluent total phosphorus data from outfall 001 for facility UT0025241
UT0025241_tp_001=readECHO_ec(p_id="UT0025241", parameter_code="00665", outfall="001")
UT0025241_tp_001_effluent=UT0025241_tp_001[UT0025241_tp_001$monitoring_location_desc=="Effluent Gross",]
head(UT0025241_tp_001_effluent)

# Extract flow through facility from UT0021717
UT0021717_flow=readECHO_ec(p_id="UT0021717", parameter_code="50050")

# Extract flow & TP from UT0025241 & UT0021717
tp_flow=readECHO_ec(p_id=c("UT0025241","UT0021717"), parameter_code=c("50050","00665"))
```

---

readECHO\_fac

*Read facility information from EPA ECHO webservices*


---

**Description**

This function extracts facility information from EPA ECHO based on argument inputs.

**Usage**

```
readECHO_fac(..., df = FALSE)
```

**Arguments**

...	Additional arguments to be passed to ECHO query path. See <a href="https://echo.epa.gov/tools/webservices/facility-search-water#!/Facility_Information/get_cwa_rest_services_get_facility_info">https://echo.epa.gov/tools/webservices/facility-search-water#!/Facility_Information/get_cwa_rest_services_get_facility_info</a> for optional arguments for facilities. Note that arguments for output are ignored.
df	Logical. If TRUE, extract geometry coordinates from list format and return as columns in data frame.

**Value**

A data frame of EPA ECHO facility information

**Examples**

```
# Read facility locations in Utah
ut_fac=readECHO_fac(p_st="ut", p_act="y")
head(ut_fac)
# Read facility locations for two permit IDs
two_fac=readECHO_fac(p_pid=c("UT0021717","UT0025241"))
two_fac
```

readWQP

*Read EPA Water Quality Portal Data***Description**

This function extracts water quality data from EPA's Water Quality Portal based on user argument inputs. Note that connections to the WQP occasionally time out during download. This function tries to download requested files up to 10 times before exiting. All arguments except type are optional, but at least one should be provided to limit download size and prevent errors connecting to WQP. Note that some, but not all, special characters in characteristic names have been accounted for. If in doubt, use the WQP web interface to determine the appropriate syntax for odd characteristic names. This function coerces non-numeric values in ResultMeasureValue column (for result & narrowresult type queries). This may generate NA values with a warning for special characters.

**Usage**

```
readWQP(
  type = "result",
  ...,
  print = FALSE,
  coerce_num = FALSE,
  url_only = FALSE,
  auid = NULL
)
```

**Arguments**

type	Data type to read. One of "result", "narrowresult", "sites", "activity", "detquantlim", or "project".
...	additional arguments to be passed to WQP query path. See <a href="https://www.waterqualitydata.us">https://www.waterqualitydata.us</a> for optional arguments.
print	Logical. Print summary table of sites & characteristics (only for result or narrowresult types).
coerce_num	Logical. If TRUE the ResultMeasureValue column in result and narrowresult type reads is coerced to numeric values. This will generate NAs in the ResultMeasureValue column for non-numeric values. Defaults to FALSE.
url_only	Logical. If FALSE (default) read and return data. If TRUE, return just the query url.
auid	Optional. A vector of Utah DWQ assessment unit identifiers for which to query data. Note that siteid is ignored if auid is specified.
start_date	Query start date in "mm/dd/yyyy" format.
end_date	Query end date in "mm/dd/yyyy" format.

**Value**

A data frame of WQP data

## Examples

```
# Read some data from Mantua Reservoir (2016-2018)
nr=readWQP(type="narrowresult", siteid=c("UTAHDWQ_WQX-4900440", "UTAHDWQ_WQX-4900460"),
  start_date="01/01/2016", end_date="12/31/2018")

# Read just Arsenic, Cadmium, and DO, all dates
nr=readWQP(type="narrowresult",
  siteid=c("UTAHDWQ_WQX-4900440", "UTAHDWQ_WQX-4900460"),
  characteristicName=c("Arsenic", "Cadmium", "Dissolved oxygen (DO)"))

# Read all Total dissolved solids statewide (2016-2018) (& note statecode for Utah)
tds_sw=readWQP(type="result",
  statecode="US:49",
  characteristicName="Total dissolved solids",
  start_date="01/01/2016", end_date="12/31/2018",
  print=F)

# Read data by assessment unit identifiers
utah_lake_nr=readWQP(type="narrowresult",
  auid=c('UT-L-16020201-004_01', 'UT-L-16020201-004_02'),
  start_date="01/01/2016", end_date="12/31/2018",
  siteType=c("Lake, Reservoir, Impoundment", "Stream"),
  print=F)
utah_lake_sites=readWQP(type="sites",
  auid=c('UT-L-16020201-004_01', 'UT-L-16020201-004_02'),
  siteType=c("Lake, Reservoir, Impoundment", "Stream"),
  print=F)
buildMap(sites=utah_lake_sites)

# Read DWQ's sites
sites=readWQP(type="sites", statecode="US:49", organization="UTAHDWQ_WQX", siteType=c("Lake, Reservoir, Impo
plot(LatitudeMeasure~LongitudeMeasure, sites[sites$LatitudeMeasure>0 & sites$LongitudeMeasure<0,])
```

---

readWQPbySite

*Read WQP data by selecting sites in an interactive map*


---

## Description

This function allows the user to read WQP data by selecting desired sites in an interactive map and specifying desired types of data and output. When map is launched, either click (mode="click") or draw polygons (mode="draw") around desired sites, then click "Done" button at bottom right of map. If edit==TRUE, an edit dialog will open in R console. Update "keep" column to anything other than "Y" to reject selected sites, then close dialog to proceed.

## Usage

```
readWQPbySite(
  sites,
  map,
  mode = "click",
  types = c("sites", "narrowresult", "activity"),
```

```

merge = T,
edit = T,
sitetypes = c("Canal Drainage", "Canal Irrigation", "Canal Transport", "Lake",
  "Lake, Reservoir, Impoundment", "Reservoir", "River/Stream",
  "River/Stream Intermittent", "River/Stream Perennial", "Seep", "Spring", "Stream",
  "Stream: Canal", "Stream: Ditch", "Wetland", "Wetland Undifferentiated"),
  ...
)

```

### Arguments

<code>sites</code>	Optional. A sites object containing lat/long columns named "Latitude-Measure" & "LongitudeMeasure". If no sites object is provided, a Utah state-wide query of WQP will be generated for you.
<code>map</code>	Optional. A map object to use as a background for site selection. If no map object is provided, a basic map will be generated for you.
<code>mode</code>	Mode for map selection. One of "click" or "draw". Click allows site selection by clicking on individual sites. Draw allows site selection by polygons.
<code>types</code>	Vector of data types to read from WQP for selected sites. See ?wqTools::readWQP for options.
<code>merge</code>	Logical. If TRUE (default), merge all selected data types to single data frame. Merges are performed as left joins in order they are provided. If FALSE, return list of individual data objects.
<code>edit</code>	Logical. If TRUE (default) open editor in R console to edit selected sites. To drop sites from the query, update the "keep" column to anything other than "Y", then close edit dialog.
<code>sitetypes</code>	Vector of site types to be included in query (only used if sites argument is not provided).
<code>...</code>	Other arguments to be passed to readWQP (e.g. start_date, end_date, characteristicName, etc.). See ?wqTools::readWQP for more info. This is passed to both the sites query (if sites are not provided) and the results queries.

### Examples

```
wqp_data=readWQPbySite(start_date="01/01/2016", end_date="12/31/2018")
```

---

<code>selectPolys</code>	<i>Interactively select spatial polygons to query or filter data</i>
--------------------------	--

---

### Description

A Shiny gadget designed for interactive selection of spatial polygons. This is intended to help users find AU and HUC IDs for querying or filtering data.

### Usage

```
selectPolys(type = "au")
```

Arguments

type                    One of 'au', 'huc8', or 'huc12' to select assessment unit, HUC 8, or HUC 12 polygons

Value

A vector of selected polygon identifiers (AU IDs or HUC numbers).

Examples

```
sites=readWQP(type='sites', auid=selectPolys('au'))
```

---

ss_poly	<i>Utah's site-specific standard polygon shapes</i>
---------	---

---

Description

Polygons containing site-specific standard designations and information.

Usage

ss\_poly

Format

An sf type polygon shapefile

---

ul_trophic	<i>Utah Lake trophic data</i>
------------	-------------------------------

---

Description

Utah Lake trophic data

Usage

data(ul\_trophic)

Format

A data.frame with 729 rows and 15 columns



---

ut_poly	<i>Utah state boundary, excluding tribal jurisdictions (unofficial)</i>
---------	---

---

**Description**

Polygon showing unofficial boundaries of the State of Utah, excluding tribal jurisdictions. This is an unofficial representation used to screen sites for the integrated report only.

**Usage**

ut\_poly

**Format**

An sf type polygon shapefile

---

wmu_poly	<i>Utah DWQ watershed management unit polygons</i>
----------	--

---

**Description**

Utah DWQ watershed management unit polygons.

**Usage**

wmu\_poly

**Format**

An sf type polygon shapefile

---

wqDD	<i>Run the Water Quality Data Dashboard Runs the Water Quality Data Dashboard app embedded in UT WQ tools package.</i>
------	--

---

**Description**

Run the Water Quality Data Dashboard Runs the Water Quality Data Dashboard app embedded in UT WQ tools package.

**Usage**

wqDD()

# Index

- \* **datasets**
  - au\_poly, [5](#)
  - au\_poly\_1820, [6](#)
  - bu\_poly, [9](#)
  - county\_poly, [12](#)
  - hnnc\_poly, [14](#)
  - huc12\_poly, [14](#)
  - huc8\_12\_poly, [14](#)
  - huc8\_poly, [15](#)
  - muni\_poly, [15](#)
  - ss\_poly, [24](#)
  - ul\_trophic, [24](#)
  - ut\_poly, [25](#)
  - wmu\_poly, [25](#)
- addMapResetButton, [2](#)
- assignAUs, [3](#)
- assignHUCs, [3](#)
- assignPolys, [4](#)
- assignUses, [5](#)
- au\_poly, [5](#)
- au\_poly\_1820, [6](#)
- baseMap, [6](#)
- bu\_poly, [9](#)
- buildLDC, [7](#)
- buildMap, [8](#)
- calcAlCrit, [9](#)
- calcTSI, [10](#)
- convertUnits, [11](#)
- county\_poly, [12](#)
- downloadWQP, [12](#)
- facToNum, [13](#)
- findSites, [13](#)
- hnnc\_poly, [14](#)
- huc12\_poly, [14](#)
- huc8\_12\_poly, [14](#)
- huc8\_poly, [15](#)
- muni\_poly, [15](#)
- profileHeatMap, [15](#)
- profilePlot, [16](#)
- readAttains, [17](#)
- readAWQMS, [18](#)
- readECHO\_ec, [19](#)
- readECHO\_fac, [20](#)
- readWQP, [21](#)
- readWQPbySite, [22](#)
- selectPolys, [23](#)
- ss\_poly, [24](#)
- ul\_trophic, [24](#)
- ut\_poly, [25](#)
- wmu\_poly, [25](#)
- wqDD, [25](#)