Environmental Setting Toolkit

Inventory and Monitoring Division
March 2018

User Guide - Version 2.1.1

The Environmental Setting Toolkit (ESToolkit) functions request weather stations and gridded data (PRISM) around NPS parks and weather observation data for metrics used in the national IMD Environmental Setting protocol from Applied Climate Information System (ACIS) webservices. The generic functions `findStation()`, `getWxObservations()`, and `getGrids()` can be used independently of the protocol to get stations and data for non-protocol use. In other words, the data responses are not limited to the metrics of the protocol.

This toolkit began as the IM_Climate toolkit (2016-2017). The version 2.1.1 release includes functions to generate station and gridded (non-index) metrics, plus bug fixes and performance enhancements. These capabilities will be added to the Python section of the Toolkit in the next release.

For examples of protocol metrics, see ES_StationMetricExamples.pdf and ES_GriddedMetricExamples.pdf files in the ES_Toolkit_R folder. The R version of the toolkit includes extensive RDoc documentation.

Known Caveats - Station Data

- toolkit does not provide access to all ACIS webservice capabilities
- error trapping for requests is very limited and several warnings are suppressed
- toolkit does not allow departure from normal requests for yearly durations
- can combine a “normal’ request with a summary (reduce) request; however, protocol metrics do not use this direct summary technique
- requests of counts return count and number of missing observations in specified duration
- R does not support the date format YYYY-MM therefore, for monthly observations, the R data frame date vector is in character format.
- All missing values are given NA if none exists. This applies on download as well.
- ACIS web service errors will cause the modules to terminate and report back the error. The one exception to this is when the error message is ‘no data available’.
- In some cases when the tenths elevation is zero, the returned value is a whole number (e.g., 100.0 is being returned as 100).

Known Caveats - Grid Data (ACIS)

- Toolkit does not provide access to all ACIS webservice capabilities
- Grid requests can be made for each source vended by ACIS; however, protocol uses NPS' copy of 800m PRISM, not ACIS' 4km data
- ACIS web service errors will cause the modules to terminate and report back the error.
- error trapping for requests is very limited and several warnings are suppressed

Known Caveats - Gridded Metrics

- The protocol uses NPS' copy of 800m PRISM, which requires an NPS intranet connection either directly or via VPN hence response time will be slow.
- 800m PRISM is available only for CONUS and the source rasters are not shareable outside of the NPS. The NPS copy is updated each summer. Currently, the temporal range is 1895-2016.
- The getMetricGrids() function accepts a Spatial Polygon object. Standard area of analysis polygons can be generated by the getAOAFeature() function. However, any SpatialPolygon with a geographic coordinate reference system (CRS) based on the NAD83 or WGS84 datum can be used.

Finding Stations

Weather stations can be requested by using park codes which translates into the park bounding box or by using a custom bounding box (in geographic coordinates). In either case, climate parameters and start and end dates can be specified. Optionally, the bounding box can be buffered in kilometers. From the `findStation()` function, the request is sent to the ACIS `StnMeta` web service, returning a data frame (and optional CSV file) with the following vectors/columns:

uid name longitude latitude sid1 sid1_type sid2 sid2_type sid3 sid3_type state elev isHCNStation minDate maxDate climateParameter unitCode

Getting Weather Observation Data - Station Protocol Metrics

Protocol stations are requested by park using `findStation()` to get stations that collected data within the last 60 years through the end of the previous calendar year and that fall within a custom bounding box of the park 30km area of analysis. These stations are imported into the master protocol database. The final station set used is created by querying the master database of all stations that intersect the 30km areas of analysis polygons for environmental setting parks using the `getProtocolStations()` function.

Protocol metrics are requested using the `getWxObservations()` function with the final station set and metric-specific parameters. The request is sent to the ACIS `StnData` web

service, returning a data frame and CSV file with the following vectors/columns plus one or more metric-specific data columns:

uid name longitude latitude sid1 sid1_type sid2 sid2_type sid3 sid3_type state elev date <dataColumn> metric

The data column(s) use a naming convention of:

climate parameter_unit of measure_(reduceCode)

where reduceCode is included only if used. An example of the CSP1 - Hot Days data column is:

maxt_F_cnt_ge_90

Getting Weather Observation Data

Weather data are requested using the `getWxObservations()` function with the one or more stations and one or more climate parameters. Stations can be specified individually or by using the output of the `findStation()` function. The request is sent to the ACIS `StnData` web service, returning a data frame and optional CSV file with the following vectors/columns plus one or more data columns:

uid name longitude latitude sid1 sid1_type sid2 sid2_type sid3 sid3_type state elev date <dataColumn>

The data column(s) use a naming convention of:

climate parameter_unit of measure_(reduceCode)

where reduceCode is included only if used. An example of the precipitation data column is:

pcpn_in

Getting Gridded Weather Data

Gridded data are requested using the `getGrids()` function with one park code and one or more climate parameters. In version 2.0, any of the grid sources in ACIS can be requested, some of which may cover only certain areas of the U.S. The request is sent to the ACIS `GridData` web service, returning an in-memory raster stack and an optional ASCII formatted grid file by date. Grids use the following projection and CRS:

```
GEOGCS[\"NAD83\",DATUM[\"D_North_American_1983\",SPHEROID[\"GRS_1980\
",6378137.0,298.257222101]],PRIMEM[\"Greenwich\",0.0],UNIT[\"Degree\"
,0.0174532925199433]]", "cellSize":"0.04166666666667
+proj=longlat +ellps=GRS80 +datum=NAD83
```

Examples using R

See additional examples in the toolkit RDocs. Stations and data observations are requested using ACIS' element codes (Table 1) as the climateParameter value. If no codes are specified, the request will default to all codes except degree days.

Table 1. ACIS station climate parameter codes.

Parameter Code	**Description**
maxt	Maximum temperature (°F)
mint	Minimum temperature (°F)
avgt	Average temperature (°F)
obst	Obs time temperature (°F)
pcpn	Precipitation (inches)
snow	Snowfall (inches)
snwd	Snow depth (inches)
gdd or gddXX	Growing degree days; XX is a base temperature other than the default (50°F)
hdd or hddXX	Heating degree days; XX is a base temperature other than the default (65°F)
cdd or cddXX	Cooling degree days; XX is a base temperature other than the default (65°F)

Stations for a park

Get stations around AGFO:

```
agfoStations20km <- findStation(unitCode = "AGFO", distance = 20,
climateParameters = list("maxt","mint","gdd"))
```

Filter to get stations collecting data within the last 20 years:

```
agfoStationsCurrent <- agfoStations20km [agfoStations20km$minDate >=
"1998-02-10" & agfoStations20km$maxDate >= "2018-02-10" ,]
```

Stations within a custom bounding box:

```
agfoStationsBBox <- findStation(unitCode = "AGFO", customBBox =
"-104.19609745831781, 42.112886184389545, -103.2970796760243,
42.7314612519983")
```

Daily weather data:

Pass the filtered list of stations to the getWxObservations() function.

```
agfoData <- getWxObservations(climateStations = agfoStationsCurrent,
climateParameters = list("maxt","mint","gdd"), sdate = "1998-02-10", edate =
"2018-02-10")
str(agfoData)
'data.frame':     43836 obs. of  22 variables
```

Monthly/Yearly weather data:

```
agfoMonthlyData <- getWxObservations(climateStations = agfoStationsCurrent,
climateParameters = list("maxt","mint","gdd"), duration = "mly", sdate =
"1998-02", edate = "2018-02")
str(agfoMonthlyData)
'data.frame':     1446 obs. of  37 variables

agfoYearlyData <- getWxObservations(climateStations = agfoStationsCurrent,
climateParameters = list("maxt","mint","gdd"), duration = "yly", sdate =
"1998", edate = "2018")
str(agfoYearlyData)
'data.frame':     126 obs. of  37 variables
```

Gridded data for a park:

Different elements are used for gridded datasets (Tables 2 and 3).

Table 2. ACIS climate parameter codes for daily climate grids.

Parameter Code	**Description**
maxt	Maximum temperature (°F)
mint	Minimum temperature (°F)
avgt	Average temperature (°F)
pcpn	Precipitation (inches)

Table 3. ACIS climate parameter codes for monthly and yearly climate grids.

Parameter Code	**Description**
mly_maxt	Monthly mean maximum temperature (°F)
mly_mint	Monthly mean minimum temperature (°F)
mly_avgt	Monthly mean average temperature (°F)

mly_pcpn	Monthly precipitation sum (inches)
yly_maxt	Yearly mean maximum temperature (°F)
yly_mint	Yearly mean minimum temperature (°F)
yly_avgt	Yearly mean average temperature (°F)
yly_pcpn	Yearly precipitation sum (inches)

```
agfoMonthlyGrids2017 <- getGrids(unitCode = "AGFO", climateParameters =
list("mly_mint","mly_maxt"), duration = "mly", sdate = "2017-01", edate =
"2017-12")
nlayers(agfoMonthlyGrids2017)
[1] 24
```

Metric rasters and summary statistics CSV for a park:

A spatial polygon is used to crop the NPS 800m PRISM source rasters, producing monthly GeoTIFF metric and (optionally) monthly normals (1981-2010) rasters. Also output are a CSV of monthly summary statistics and plots by year of source, metric, and optionally, normals monthly rasters. Metrics produced are percent normal precipitation and departure from normal in degrees Fahrenheit. For the latter, the departure is calculated in degrees Celsius then converted to Fahrenheit.

```
# Get the standard 30km AOA feature for AGFO
agfoAOA <- getAOAFeature(unitCode = "AGFO")

# Request CGT1, the monthly departure of mean temperature from 30-year climate
normal (1981-2010), for 2016.
agfoTemperatureMetricStack <- getMetricGrids(featurePolygon = agfoAOA, metric =
"CGT1", unitCode = "AGFO", sdate = "2016", edate = "2016", includeNormals =
TRUE)
str(agfoTemperatureMetricStack)

# Request CGP1, the monthly percent of 30-year climate normal (1981-2010) mean
temperature, for 2016.
agfoPrecipitationMetricStack <- getMetricGrids(featurePolygon = agfoAOA, metric
= "CGP1", unitCode = "AGFO", sdate = "2016", edate = "2016", includeNormals =
TRUE)
str(agfoPrecipitationMetricStack)
```

Appendices

General Logic

- All missing values are given NA if none exists. This applies on download as well.
- ACIS web service errors will cause the modules to terminate and report back the error. The one exception to this is when the error message is 'no data available'.
- Known Issues
    - In some cases when the tenths elevation is zero, the returned value is a whole number (e.g., 100.0 is being returned as 100).

StationFinder

```
findStation()
```

- Invalid unitCode not error trapped. In other words, someone passes an an invalid code or a bounding box does not exist for a particular unit code or more than one unit code is specified, there is no logic to catch/warn the user and prevent the the request from failing
- All missing properties (e.g., station missing an elevation, sid1 does not exist, etc.) are converted to 'NA'.
- ACIS terminates requests returning thousands of stations (not sure of exact number). There is no attempt to trap for this. Because all arguments are optional, a user can potentially search for all stations in ACIS get no results.
- Climate parameters should be comma-delimited. Failure to do so may result in errors.
- The unit code supplied is used to request a bounding box (longitude and latitude) from the IRMA Unit service. This bounding box is buffered by a user-supplied distance (kilometers) which is translated into decimal degrees. The formula used to calculate the buffer in decimal degrees is : $Buffer_{DD} = Buffer_{KM} * 0.011$. Buffer distance is ignored if parkCode is not specified.
- When exporting the text file, there is no attempt to add extension and/or check for valid extensions. File delimiter is a comma.
- There is no attempt to error trap if only dDate or eDate is provided. ACIS will ignore the date request if only one is provided. Invalid date formats will also results in failure.
- ValidDateRange is caluclated using the following logic:
    - If no climate parameter specified, return max date range of all supported climate parameters (see table 1) (via background request)
    - If climate parameters are specified, return max combined date range for a station for specified set of parameters
- Station Type is assigned for sid1, sid2, sid3 only. Additional station IDs (sometimes up to 7) are ignored

Station Data Requestor

```
getDailyWxObservations()
```

- Requested climate parameters and station ids are not verified to be valid.
- Start and end dates are not checked against period of record.
- No changes are made to ACIS flag values in response (e.g. T-trace, M-missing, etc.).
- Empty responses are changed to "NA".
- The application will automatically request the following codes from Table 7 in the ACIS documentation: Flags and source flags (add = 'f,s')

- Multiple parameters should be comma-delimited. The program does remove white-space to ensure a successful call to ACIS.
- Start and end dates will need to be in format YYYY-MM-DD or YYYYMMDD. There is no attempt to correct or catch other date formats.
- If there is no daily data to export, then a file is not created
- If sdate and edate or not specified, the entire period of record is returned
- If multiple stations are requested and data does not exist for some, those stations not having weather data will be ignored on export
- In R, for monthly data, value vector returned as character format to accommodate missing or trace records ("M", "T"). In Python, export to CSV returns character format outputs.
- R does not support the YYYY-MM format for a Date object. Hence, for monthly observations, the date vector is in character format.
- Currently, precision is not specified in the request.

Grid Requestor

```
getGrids()
```

- For PRISM monthly and yearly data, these data are aggregated by PRISM, not by ACIS.
- Since unitCode is optional yet ACIS requires a bounding box, the default bbox is '-130, 20,-50,60'
- Output grids are spatially registered to the centroid of the lower left pixel
- Grids use the following spatial reference: NAD83 (geographic)
    - .prj format (for ASCII outputs)
        - GEOGCS[\"NAD83\",DATUM[\"D_North_American_1983\",SPHEROID[\"GRS_1980\",6378137.0,298.257222101]],PRIMEM[\"Greenwich\",0.0],UNIT[\"Degree\",0.0174532925199433]]", "cellSize":"0.04166666666667"
    - crs format (for R raster stack outputs)
        - +proj=longlat +ellps=GRS80 +datum=NAD83 +no_defs
- For PRISM, ACIS returns the temperature grids in fahrenheit, which differs from the source grids found here: http://www.prism.oregonstate.edu/. In other words, PRISM natively uses degrees Celsius (temperature) and tenths of centimeters (precipitation) and ACIS performs the unit conversion calculations.
- When requesting grids, the default precision is to the nearest tenth (e.g., 10.1 degrees F). This precision results in about a 0.05 F difference in temperature.
- ACIS returns -999 for missing values
- Supported output formats are:
    - ASCII grid format along with PRJ file
    - In-memory raster stack

```
getMetricGrids()
```

- Output rasters use the following spatial reference: NAD83 (geographic)
    - .prj format (for ASCII outputs)
        - GEOGCS[\"NAD83\",DATUM[\"D_North_American_1983\",SPHEROID[\"GRS_1980\",6378137.0,298.257222101]],PRIMEM[\"Greenwich\",0.0],UNIT[\"Degree\",0.0174532925199433]]", "cellSize":"0.04166666666667"
    - crs format (for R raster stack outputs)
        - +proj=longlat +ellps=GRS80 +datum=NAD83 +no_defs
- NPS' 800m PRISM data units are mm for precipitation and degrees Celsius for temperature.

Bugs and Issues

- There is sometimes an inconsistency in the elevation precision (e.g., 1098.0 vs 1098). Whether this is a result of R and Python interpreting the precision differently or whether the precision changes depending on the ACIS web service call is to be determined.