Using the climateR and AOI package

Kenneth Pomeyie, Scout Jarman, Paul Gaona-Partida

Introduction

A climatic data element is a measured parameter that helps to specify the climate information of a specific location or region, such as precipitation, temperature, wind speed, and humidity. Raster data models(files) are one way of storing climate data in pixels. The size of these raster files tends to get large as dimensionality increases with time, multiple variables, and multiple models. In most cases, only a small amount of information is needed from these files. The hard way to extract site-specific climate information from raster files involves downloading the files and extracting the required information individually to reduce size. However, this data extraction method is inefficient and memory-limited because R loads everything in RAM. This vignette shows how the climateR and AOI packages can be used to call specific climate data more efficiently.

Introduction to AOI and climateR package Scout

Advantages of AOI and climateR package

1. climateR package offers support for sf, sfc, and bbox objects
2. climateR package provides access to 11 climate resources centralized under a common access pattern (PRISM, Daymet, GridMET, etc)
3. climateR package allows users to call specific climate data rather than downloading terabytes of data not needed
4. AOI package allows users to program traditional and non-traditional boundaries for analysis and mapping workflow

AOI Usage

Paul and Scout start ### 1. Load libraries

How to install AOI and climateR from github and load library packages that will be used in examples.

```
#remotes::install_github("mikejohnson51/AOI")
#remotes::install_github("mikejohnson51/climateR")

#load library
library(raster)
```

```
## Loading required package: sp
```

```
library(sf)
```

```
## Linking to GEOS 3.9.1, GDAL 3.4.0, PROJ 8.1.1; sf_use_s2() is TRUE
```

```
library(climateR)
library(AOI)
library(ggplot2)
library(leaflet.extras)
```

```
## Loading required package: leaflet
```

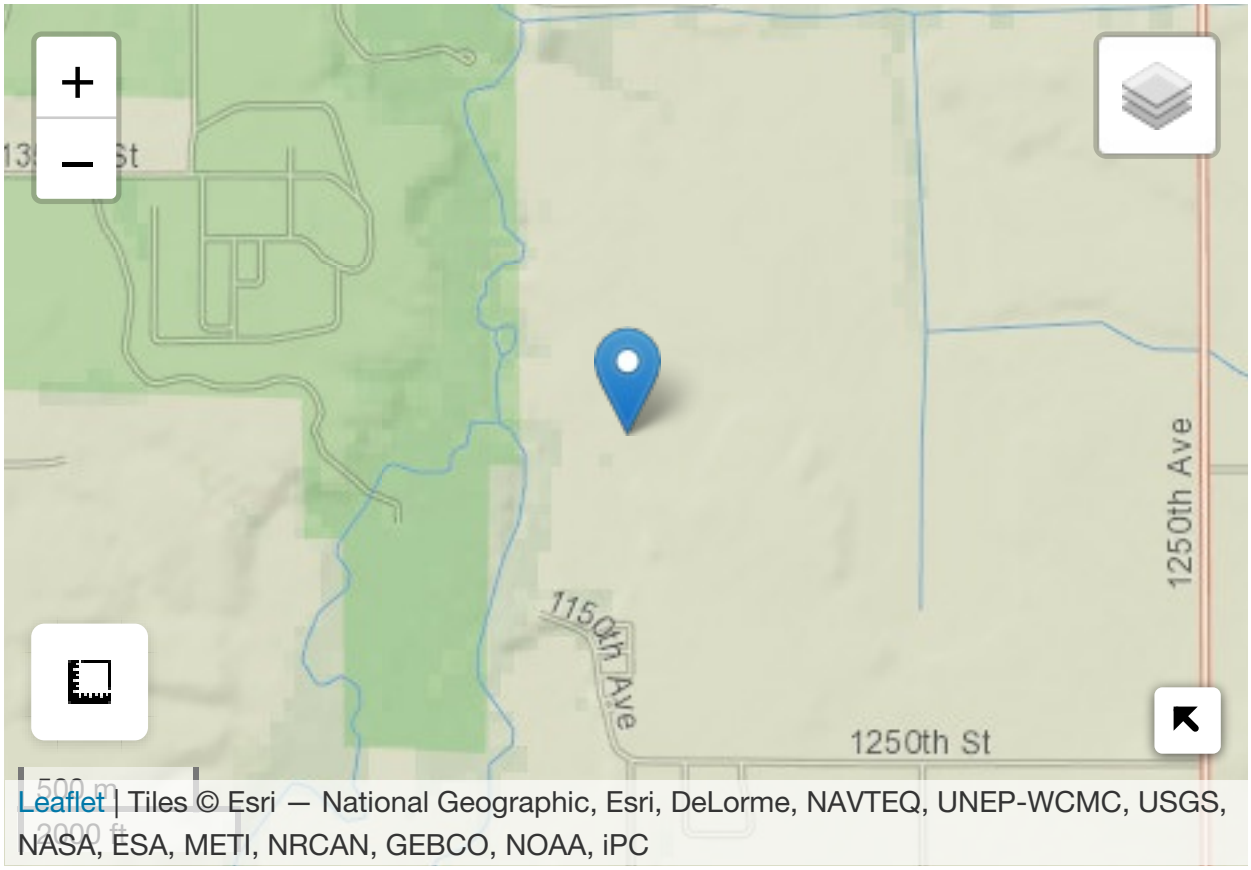
2. Geocode locations

How to geocode a point geometry. 'pt = True', will make sure that the returned object will be a 'sf' object. 'pt = FALSE' will return a 'data.frame' object. All other parameters were set to 'NULL' or 'FALSE, as an example to specify parameters or location for accurate results.

```
# get point geometry of Logan
logan <- geocode(
  location = "Logan",
  zipcode = NULL, event = NULL, pt = TRUE, bb = FALSE
)
```

An example to check if returned object is the correct location via a map. (This example shows that it is not)

```
# check if the location is right
aoi_map(AOI = logan, returnMap = TRUE)
```

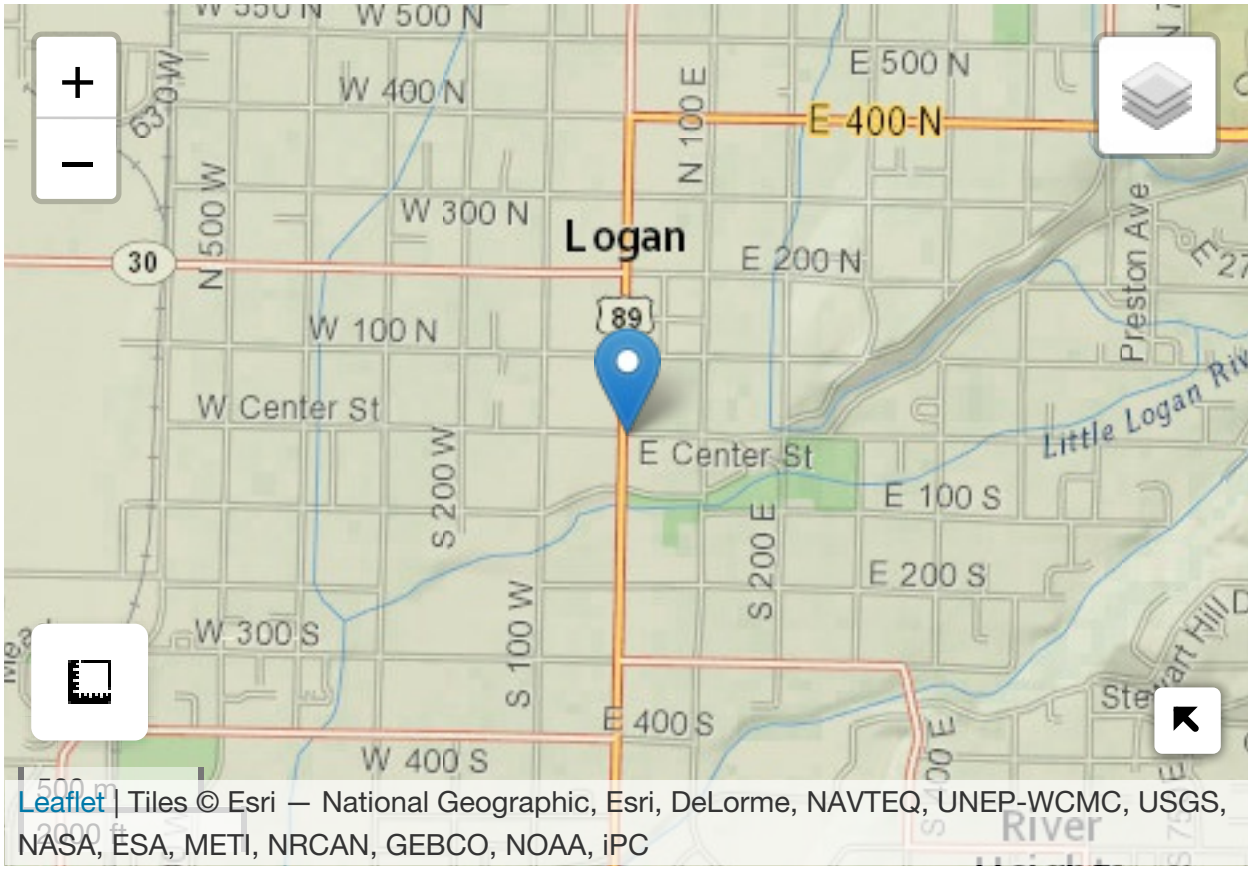


A correcetion to the previous example, this time with the correct specifications for the area of interest.

```
# use correct location name
logan <- geocode(location = "Logan UT", pt = TRUE)
```

A visual check of the correctly specified object's map.

```
# check if the location is right
aoi_map(AOI = logan, returnMap = TRUE)
```



A interactive shiny app, that allows for user drawn maps, will return an multipolygon. There are difficulties in saving directly to envionment, A suggestiong of saving as a file and following code below to read file of object and visually see object.

```
# Interactively draw an Area of Interest
aoi_draw()
```

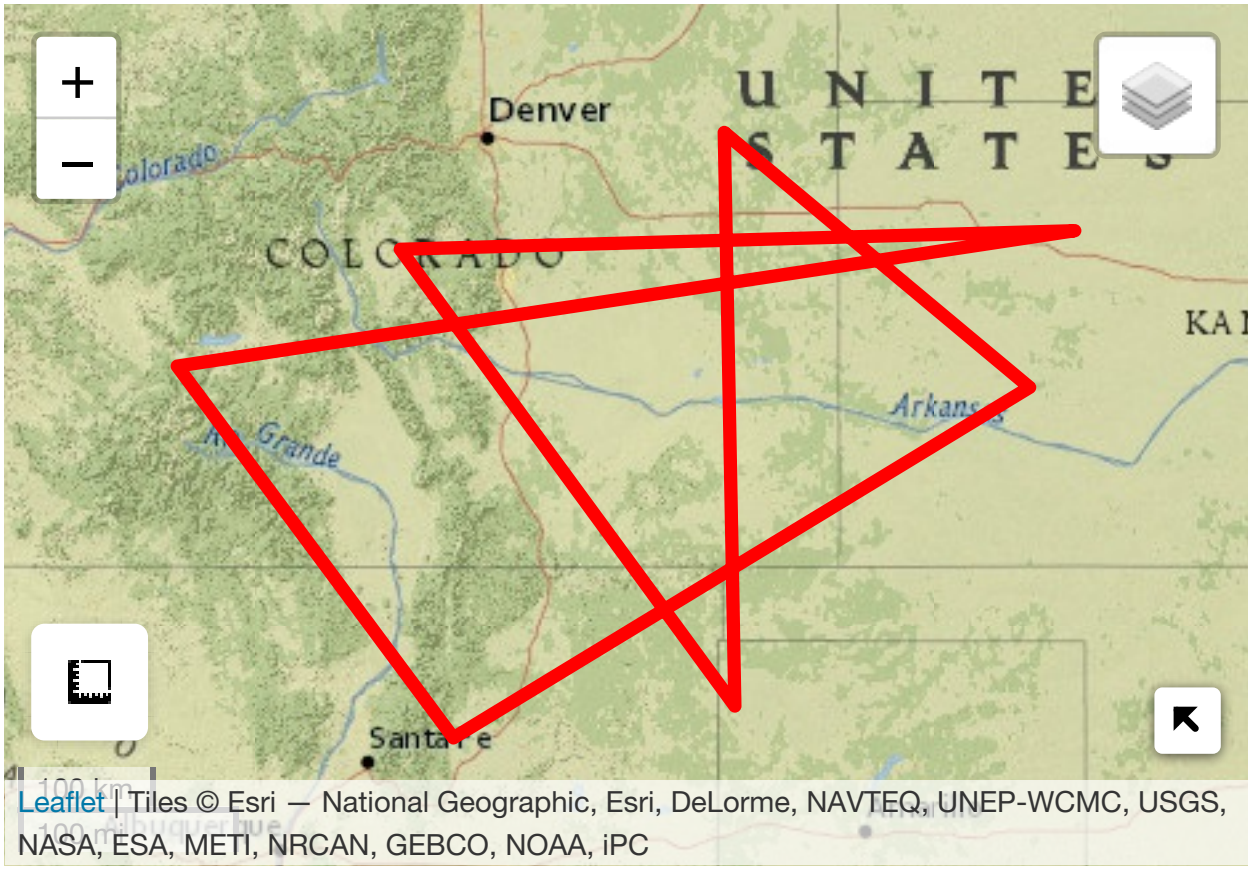
```
## Loading required package: shiny
```

```
##
## Listening on http://127.0.0.1:7661
```


```
test <- sf::st_read("aoi_draw/test.gpkg")
```

```
## Reading layer `test' from data source
##   `/Users/kenkin/Documents/USU/Classes/SPRING 2022/APPLIED SPATIAL/project/stat6410_software_project/
##   using driver `GPKG'
## Simple feature collection with 1 feature and 0 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: -107.5034 ymin: 35.8519 xmax: -100.0854 ymax: 39.80818
## Geodetic CRS:  WGS 84
```

```
aoi_map(test, returnMap = TRUE)
```



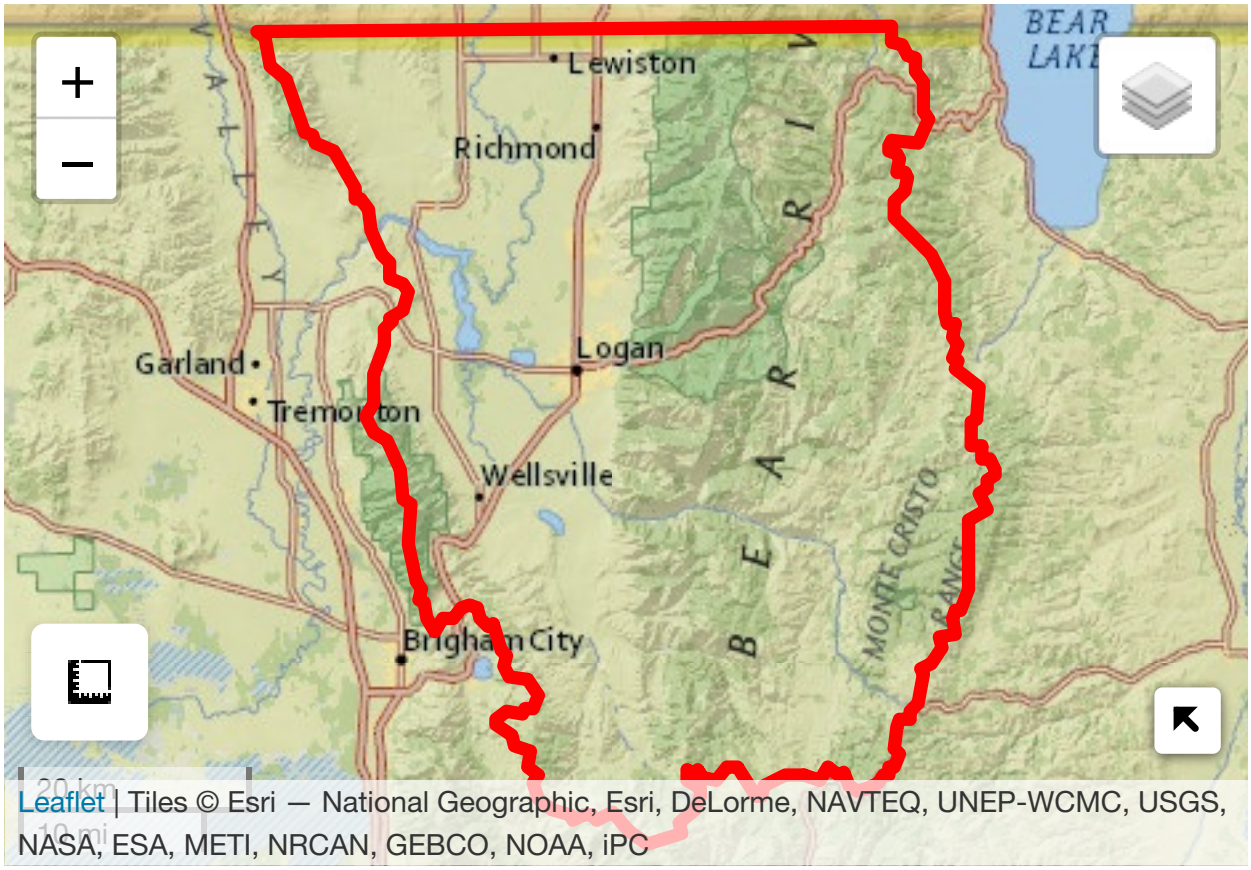
3. Queries for state and county boundaries

Easily able to return polygon of Utah and Cache County by specifing parameters. A Map is returned to inspect that the specified object is correct.

```
# get Cache county boundaries
cache_aoi <- aoi_get(state = "UT", county = "Cache")
cache_aoi
```

```
## Simple feature collection with 1 feature and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -112.1658 ymin: 41.36882 xmax: -111.4016 ymax: 41.99955
## Geodetic CRS:  WGS 84
##   state_region state_division feature_code state_name state_abbr  name
## 1            4              8      1448017       Utah         UT Cache
##   fip_class tiger_class combined_area_code metropolitan_area_code
## 1        H1       G4020                 NA                  <NA>
##   functional_status  land_area water_area fip_code
## 1                 A 3016627500   21097698    49005
##                       geometry
## 1 MULTIPOLYGON (((-111.5078 4...
```

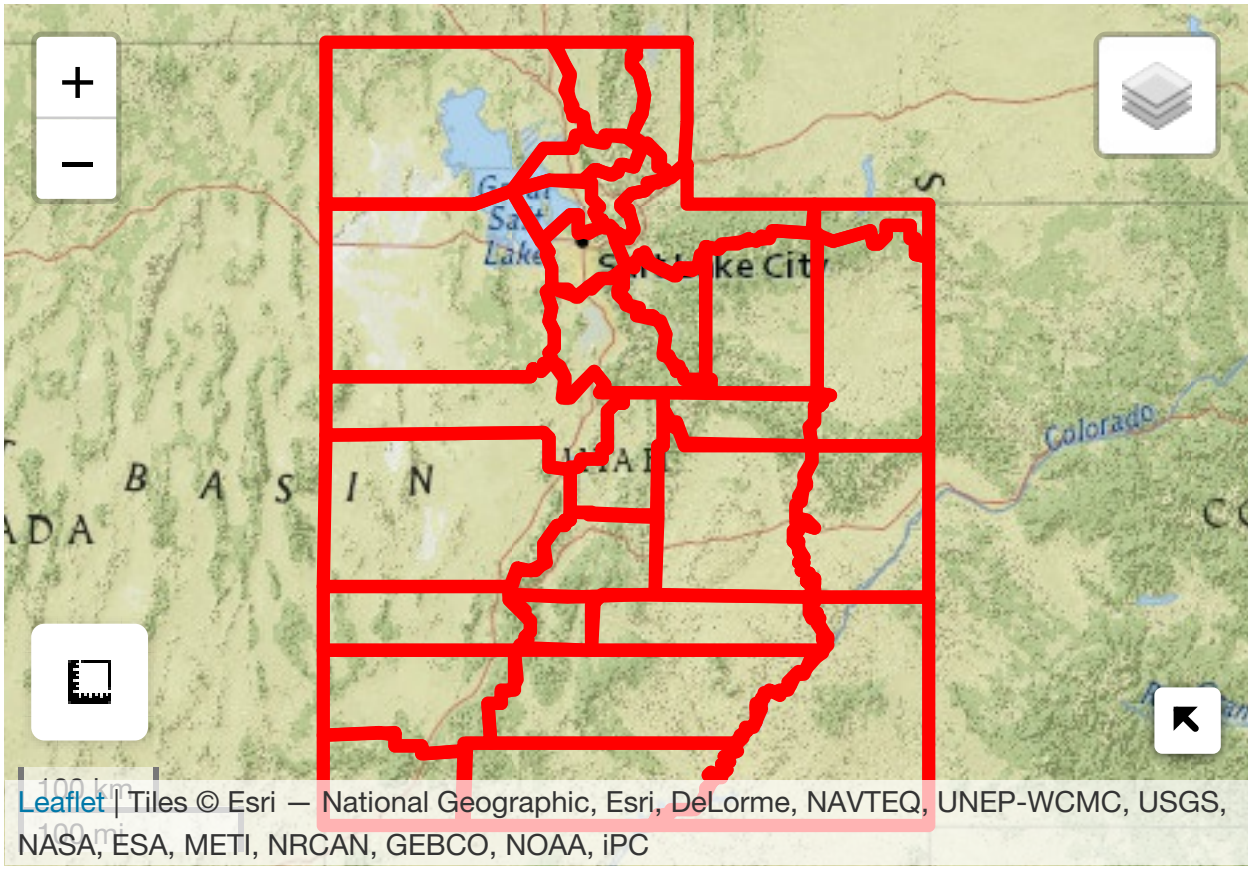
```
# map area of interest
aoi_map(AOI = cache_aoi, returnMap = TRUE)
```



Easily abe to return polygon of Utah and every county associated with Utah. A Map is returned to inspect that the specified object is correct.

```
# get Utah state boundaries
ut_aoi <- aoi_get(state = "UT", county = "all")
```

```
# map area of interest
aoi_map(AOI = ut_aoi, returnMap = TRUE)
```



Paul and Scout end

climateR Usage

4. Point Based Time Series

In this example, we will show how to download a point based time series with a single climate data for a specified location. For this example, we want to get the minimum temperature of Logan, UT from the PRISM climate source. PRISM is a dataset of daily high-spatial resolution (4-km) surface meteorological data covering the contiguous US from 1981-present. Using the getPRISM function, we specify our area of interest(Logan, UT), the start and end time, and the minimum temperature as the parameter.

```
# get minimum temperature from PRSIM
logan_temp <- climateR::getPRISM(
  AOI = AOI::geocode(location = "Logan UT", pt = TRUE),
  param = "tmin", startDate = "2021-01-01",
  endDate = "2021-12-31"
)
```

```
## Spherical geometry (s2) switched off
```

```
## Spherical geometry (s2) switched on
```

```
head(logan_temp)
```

```
##   source   lat       lon       date    tmin
## 1  prism 41.75 -111.8333 2021-01-01  -7.538
## 2  prism 41.75 -111.8333 2021-01-02  -6.804
## 3  prism 41.75 -111.8333 2021-01-03  -8.001
```

```
## 4  prism 41.75 -111.8333 2021-01-04  -8.065
## 5  prism 41.75 -111.8333 2021-01-05  -7.940
## 6  prism 41.75 -111.8333 2021-01-06 -11.286
```

5. Multi Point Time Series

In this section of the workflow, we will show how to download multi point times series of climate data. For this example, we will download precipitation and minim temperature for Logan, Salt Lake, and Brigham in Utah from the GridMET climate source. GridMET is a dataset of daily high-spatial resolution (4-km) surface meteorological data covering the contiguous US from 1979-yesterday. First, we will use the **geocode** function to create an sf object of point geometry for the specified locations. Next, we will feed the **getGridMET** with our defined sf object and specify our climate type along with the date range. The **getGridMET** will return a list of two raster files representing precipitation and minimum temperature. Finally, we use the **extract_sites** to extract the climate data for the locations, which returns a list.

```
# get point of 3 sites
sites <- AOI::geocode(location = c(
  "Logan UT", "Salt Lake UT", "Brigham UT"
), pt = TRUE)


# get precipitation and min temp from GridMET
ut_ppt_tmin <- getGridMET(
  AOI = sites,
  param = c("prcp", "tmin"), startDate = "2021-01-01",
  endDate = "2021-01-31"
)
```

```
## Spherical geometry (s2) switched off
```

```
## Spherical geometry (s2) switched on
```

```
# extract precipitation and temperature for specified sites from the raster stack
site_extract <- extract_sites(ut_ppt_tmin, sites, "request")

head(site_extract$gridmet_prcp, n=5)
```

```
##         date site_Logan UT site_Salt Lake UT site_Brigham UT
## 1 2021-01-01             0                 0               0
## 2 2021-01-02             0                 0               0
## 3 2021-01-03           1.2                 0             1.6
## 4 2021-01-04             3               0.4             1.8
## 5 2021-01-05             0                 0             2.6
```

```
head(site_extract$gridmet_tmin, n=5)
```

```
##         date site_Logan UT site_Salt Lake UT site_Brigham UT
## 1 2021-01-01         265.6             266.8           263.2
## 2 2021-01-02         264.4             268.1           265.1
## 3 2021-01-03         264.3             269.4           268.2
## 4 2021-01-04         264.7             269.6           269.7
## 5 2021-01-05         261.7               267             267
```

6. Multi-layer raster

In this section of the workflow, we show how to download a multi-layer raster file from the GridMET climate source. Using **getGridMET** function we will define the area of interest, the climate parameter and the date range. The difference example 4 and this example is the type of sf object fed to the **getGridMET**. In this

example since we want a raster file we defined the our of interest as a polygon. Below is the plot of the rasters of minimum and maximum temperature for the state of Utah for January 1, 2000.

```
# get max and min temp  rasters from GridMET
grids <- getGridMET(aoi_get(state = "UT"),
  param = c("tmax", "tmin"),
  startDate = "2000-01-01"
)
```

```
## Spherical geometry (s2) switched off
```

```
## Spherical geometry (s2) switched on
```

```
raster::plot(raster::stack(grids), main = c("Min Temperature", "Max Temperture"))
```