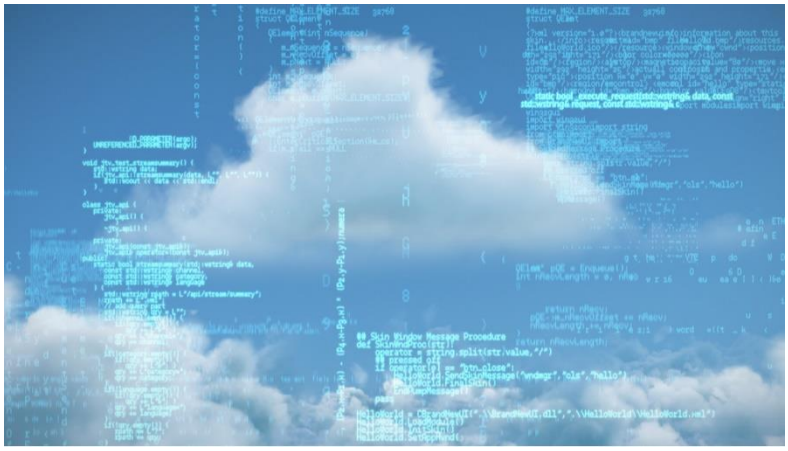
FIRST ANALYTICS®

# Retrieving NOAA Weather Data through the NOAA API in SAS



February 2024

FIRST ANALYTICS®

## Contents



2500 Regency Parkway
Cary, NC 27518
P: (919) 521-5519 F: (202) 318-4485

FIRST ANALYTICS®

# Introduction

Analyzing weather patterns can provide valuable insights into how certain events, such as temperature changes or precipitation, may affect business operations such as sales at retail stores.

Explore how First Analytics leverages weather analysis in a variety of ways through these case studies[1] on our website.

In this technical paper, we aim to analyze how weather conditions, such as temperature and precipitation, influenced sales at the stores of one of our clients. To achieve this, we retrieve the coordinates of NOAA weather stations, identify the stations closest to the client's stores, and download the corresponding weather data.

We will guide you through the process of retrieving NOAA (National Oceanic and Atmospheric Administration) weather data using the NOAA API in SAS 9.4.

[1] https://firstanalytics.com/portfolio-item/weather-data-in-analytics/



FIRST ANALYTICS®

# Background

NOAA provides a comprehensive dataset of weather information through its API. You can find information regarding the Climate Data Online (CDO) web services that you can access through the API here[2]. However, accessing this data efficiently can be challenging due to limitations on the volume of data that can be retrieved at once. To overcome this challenge, we have developed SAS code that enables iterative downloading of weather data, optimizing the retrieval process.

[2] https://www.ncdc.noaa.gov/cdo-web/webservices/v2/?ref=

FIRST ANALYTICS®

# Step-by-Step Code

1. Setup and Initialization
   - Establish a SAS session with necessary configurations.
   - Import required SAS libraries and macros.
   - Obtain authorization to use the NOAA API by generating an API token here[3]. %let token=*your_token*;

2. Retrieving NOAA Stations
   - Define parameters for retrieving NOAA weather stations based on location and data type (e.g., temperature). You can retrieve the stations to a certain geographical location using a FIPS (Federal Information Processing System) code; see all USA State codes here[4].
   - Utilize the PROC HTTP procedure to make GET requests to the NOAA API.
     - Notice that we are using the CATS() function in a data step to have a better control of the parameters in the API request.
   - Parse the JSON response to extract station data.
   - Finally, store the station data in the WORK.STATIONS dataset. Repeat the station data retrieval for the States needed and append all results in the WORK.STATIONS dataset.

```
filename resp temp;

/* Retrieve the first 1000 stations in North Carolina*/
data _null_;
      length url $20000;
      url=cats('https://www.ncdc.noaa.gov/cdo-web/api/v2/stations'
                  , '?datasetid=GHCND'
                  , '&locationid=FIPS:37'
                  , '&datacategoryid=TEMP'
                  , '&limit=1000'
                  );
      call symputx('url', quote(trim(url), "'"));
run;

proc http method="GET" url=&url. out=resp;
      headers "token"="&token.";
run;
```

[3] https://www.ncdc.noaa.gov/cdo-web/token
[4] https://www.census.gov/geographies/reference-files/2016/demo/popest/2016-fips.html

FIRST ANALYTICS®

```
libname data JSON fileref=resp;

data work.stations;
      set data.results;
run;
```

| ordinal_root | ordinal_results | elevation | mindate | maxdate | latitude | name | datacoverage | id | elevationUnit | longitude |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 185.9 | 1911-06-20 | 2023-12-31 | 35.3999 | ALBEMARLE, NC US | 0.9766 | GHCND:USC00310090 | METERS | -80.1994 |
| 1 | 2 | 841.9 | 1910-11-01 | 1962-03-31 | 35.9000 | ALTAPASS, NC US | 0.8047 | GHCND:USC00310160 | METERS | -82.0167 |
| 1 | 3 | 533.1 | 1909-08-01 | 2005-09-30 | 35.2014 | ANDREWS, NC US | 0.9133 | GHCND:USC00310184 | METERS | -83.8386 |
| 1 | 4 | 137.2 | 1993-07-01 | 2024-02-06 | 35.7426 | APEX, NC US | 0.9699 | GHCND:USC00310212 | METERS | -78.8370 |

Figure 1: First rows in the WORK.STATIONS dataset for North Carolina.

3. Data Processing:
   - Cleanse and preprocess the station data.
   - Filter out irrelevant stations or outdated information.
   - Convert date formats and perform necessary data transformations.
   - Generate lookup files for stations' longitude and latitude; LUS stands for Lookup Station.

```
data work.stations;
      set work.stations;
      max_date=INPUT(maxdate, YYMMDD10.);
      if max_date<'01Jul2023'd then delete;
run;

proc sql noprint;
create table TBL_STATION_LONG_LAT_LOOKUP as
      select id   as LUS_Station
            , Longitude      as LUS_Longitude
            , Latitude       as LUS_Latitude
      from work.stations
      order by id;
quit;
run;
```

4. Find the nearest NOAA stations
   - Find the nearest weather stations for each ZIP code in the USA.
   - Retrieve the ZIP code of the client's stores and get the list of nearest weather stations.

FIRST ANALYTICS®

```
proc sql noprint;
create table TBL_ZIP_LONG_LAT as
      select put(Zip, z5.)  as ZipCode
            , X    as Longitude
            , Y    as Latitude
      from SASHELP.ZIPCODE
      order by ZipCode;
quit;
run;
proc sql noprint;
create table TBL_ZIP_LONG_LAT_LOOKUP as
      select ZipCode    as LUZ_ZipCode
            , Longitude      as LUZ_Longitude
            , Latitude       as LUZ_Latitude
      from TBL_ZIP_LONG_LAT
      order by ZipCode;
quit;
run;

%macro Get_Long_Lat_Distance(LONG1, LAT1, LONG2, LAT2);
((arcos(sin(&LAT1 * constant('PI') / 180) * sin(&LAT2 * constant('PI') / 180)
    + cos(&LAT1 * constant('PI') / 180) * cos(&LAT2 * constant('PI') / 180) *
cos((&LONG2 - &LONG1) * constant('PI') / 180))
    * 180 / constant('PI')) * 60 * 1.1515)
%mend Get_Long_Lat_Distance;

data TBL_ZIP_NEAREST_STATION
(keep =  ZipCode   Station   Miles   Station_2    Miles_2    Station_3    Miles_3
) ;
attrib ZipCode   length=$5 ;
attrib Miles     length= 8 ;
retain id ;
retain miles 0;
retain id_2 ;
retain miles_2 0;
retain id_3 ;
retain miles_3 0;

do while (zips_eof = 0);
      set TBL_ZIP_LONG_LAT_LOOKUP end=zips_eof;
      ZipCode = LUZ_Zipcode;
      id = .;
      miles = .;
      id_2 = .;
      miles_2 = .;
      id_3 = .;
      miles_3 = .;
      station_pt = 0;
      do while (station_pt < station_nobs);
            station_pt + 1;
            set TBL_STATION_LONG_LAT_LOOKUP point=station_pt nobs=station_nobs;
```



FIRST ANALYTICS®

```
            Distance = %Get_Long_Lat_Distance(luz_longitude, luz_latitude,
lus_longitude, lus_latitude);
            if (Miles = . or Distance < Miles) then do;
                  Miles_3 = Miles_2;
                  id_3 = id_2;
                  Miles_2 = Miles;
                  id_2 = id;
                  Miles = Distance;
                  Station = LUS_Station;
            end;
            else if (Miles_2 = . or Distance < Miles_2) then do;
                  Miles_3 = Miles_2;
                  id_3 = id_2;
                  Miles_2 = Distance;
                  id_2 = LUS_Station;
            end;
            else if (Miles_3 = . or Distance < Miles_3) then do;
                  Miles_3 = Distance;
                  id_3 = LUS_Station;
      end;

   end;
   output TBL_ZIP_NEAREST_STATION;
end;
stop;
run;

* ------------------------------------------------------------------------ ;
*  Create the lookup file.
* ------------------------------------------------------------------------ ;
*  LUNS = Lookup Nearest Station
* ------------------------------------------------------------------------ ;

proc sql noprint;
create table TBL_ZIP_NEAREST_STATION_LOOKUP as
      select ZipCode   as LUNS_ZipCode
            , Station      as LUNS_Station
      from TBL_ZIP_NEAREST_STATION
      order by LUNS_ZipCode;
quit;
run;

/* Find the LAT and LONG of the Client's - directory of stores */
proc sql noprint;
create table CLIENT_STORES_NEAREST_STATION as
      select  Client.unit
            ,Z.LUNS_Station
      FROM        CLIENTSDATASET as Client
      left join   TBL_ZIP_NEAREST_STATION_LOOKUP as Z
      on Client.ZIP =  Z.LUNS_ZipCode
;
quit;
```



```
run;
```

| Unit | LUNS_Station |
|---|---|
| 1041 | GHCND:USC00311975 |
| 1090 | GHCND:USC00314970 |
| 1094 | GHCND:USW00093719 |
| 1111 | GHCND:USC00311975 |
| 1121 | GHCND:USC00317516 |
| 1141 | GHCND:USC00317319 |
| 1161 | GHCND:USC00314996 |

Figure 2: View of the client's stores (the column "unit") and nearest weather stations (the column "LUNS_Station")

```
/* Generate the list of the weather stations that are close to the client's stores*/
proc sql;
      select  distinct(Luns_station) into: nearest_station_list
 separated by ' ' from CLIENT_STORES_NEAREST_STATION;
quit;
run;
```

## 5. Optimizing Iterative Data Retrieval

- Use macro parameters to optimize iterative data retrieval.

1. The dataset GHCND contains one row for the following daily measurements at the specified weather station ID:

    - PRCP = Precipitation (tenths of mm)
    - SNOW = Snowfall (mm)
    - SNWD = Snow depth (mm)
    - TMAX = Maximum temperature (tenths of degrees C)
    - TMIN = Minimum temperature (tenths of degrees C)
    - TAGV = Average temperature (tenths of degrees C)

FIRST ANALYTICS®

2. Use the macro loop_retrieve_NOAA_data that retrieves the weather station measurements from a start to end date iteratively for each station in &nearest_station_list.
3. Compute the number of iterations as the number of days between the beginning of the client data, Jan 2018, and today multiplied by 6 (since we have at most 6 rows per day in the data) and divided by 1000 (which is the maximum number of rows retrievable by the API).
    - Use the CATS() function in the API request to address conflicts between macro parameters and API parameters, since both use the '&' symbol.

| date | datatype | station | attributes | value |
|---|---|---|---|---|
| 2018-01-02T00:00:00 | TAVG | GHCND:AM000037959 | H,,S, | 45 |
| 2018-01-02T00:00:00 | TMAX | GHCND:AM000037959 | ,,S, | 56 |
| 2018-01-02T00:00:00 | PRCP | GHCND:AMM00037717 | ,,S, | 51 |
| 2018-01-02T00:00:00 | TAVG | GHCND:AMM00037717 | H,,S, | -9 |
| 2018-01-02T00:00:00 | TMAX | GHCND:AMM00037717 | ,,S, | 14 |
| 2018-01-02T00:00:00 | TAVG | GHCND:AO000066160 | H,,S, | 258 |
| 2018-01-02T00:00:00 | TAVG | GHCND:AO000066390 | H,,S, | 207 |
| 2018-01-02T00:00:00 | ADPT | GHCND:AQW00061705 | ,,W, | 250 |
| 2018-01-02T00:00:00 | ASLP | GHCND:AQW00061705 | ,,W, | 10105 |
| 2018-01-02T00:00:00 | ASTP | GHCND:AQW00061705 | ,,W, | 10098 |
| 2018-01-02T00:00:00 | AWBT | GHCND:AQW00061705 | ,,W, | 261 |
| 2018-01-02T00:00:00 | AWND | GHCND:AQW00061705 | ,,W, | 21 |

Figure 3: View on the daily measurements (as seen in datatype) at different weather stations (as seen in station)

```
%macro loop_retrieve_NOAA_data(nearest_station_list);

data _null_;
      date=put(today(), YYMMDD10.);
      call symput('today', date);
      start_date=put('01Jan2018'd, YYMMDD10.);
      call symput('start_date', start_date);
      n_iterations=int(intck('day', '01Jan2018'd, today())*6/1000) +1;
      call symput('n_iterations', n_iterations);
run;

%let it=1;
%do %while( %scan(&nearest_station_list, &it.,' ') ne );
      %let station= %scan(&nearest_station_list, &it.,' ');

            %put today=&today;
            %do it2=1 %to &n_iterations;
```

FIRST ANALYTICS®

```
            data _null_;
             length url $20000 ;
            new_start=put(intnx( 'day',  '01Jan2018'd, %eval(( &it2.-1)
*166)),YYMMDD10.);
            new_end=put(intnx( 'day',  '01Jan2018'd, %eval(&it2.
*166)),YYMMDD10.);
             url = cats
            ('https://www.ncdc.noaa.gov/cdo-web/api/v2/data'
            ,'?stationid=', "&station."
            ,'&datasetid=GHCND'
            ,'&startdate=', new_start
            ,'&enddate=',  new_end
            ,'&limit=1000'
            );
             call symputx('url',quote(trim(url),"'"));
            run;

            proc http  method="GET"  url=&url. out=resp;
             headers   "token" = "&token.";
            run;

            libname data JSON fileref=resp;
            %if %sysfunc(exist(data.results)) %then %do;
                  %if %sysfunc(exist(NOAA_DATA))  %then %do;
                  data NOAA_DATA;
                        set NOAA_DATA  data.results;
                  run;
                  %end;

            %else %do;
                  data NOAA_DATA;
                        set data.results;
                  run;
            %end;
      %end;
      %end;

%let it=%eval(&it.+1);
%end;

%mend loop_retrieve_NOAA_data;


%loop_retrieve_NOAA_data(&nearest_station_list);
proc sort data=NOAA_DATA nodupkey;
      by station date datatype value;
run;
```



FIRST ANALYTICS®

## 6. Saving Data

- Transpose the NOAA data to get the weather measurements as columns of the table.
- Finally, merge the transposed NOAA_DATA dataset with the client's data at store, daily level.

```
proc transpose data=NOAA_DATA out=NOAA_DATA_t;
      id datatype;
      by station date ;
      var value;
run;

data casuser.NOAA_DATA_t;
      set NOAA_DATA_t (keep=station date PRCP SNOW SNWD  TAVG TMAX TMIN);
      format date2 date9.;
      date2= input(substr(left(date),1,10), YYMMDD10.);
      drop date;
      rename date2=date;
      if PRCP           >  99       THEN PRCP    =.;
      if SNDP           > 999       THEN SNDP    =.;
run;


proc sql noprint;
create table CLIENTSDAILYDATA_withWeather as
      select  client.Unit
            ,client.Sales,
            ,ST.LUNS_Station
            ,ND.*
      FROM        CLIENTSDAILYDATA as client
      left join   CLIENT_STORES_NEAREST_STATION as ST
      on  client.StoreNum =  ST.UNIT
      left join NOAA_DATA_t as ND
      on  ST.Luns_station =  ND.STATION
      and datepart(client.Date)=ND.date
;
quit;
run;
```

FIRST ANALYTICS®

# Conclusion

Retrieving NOAA weather data through the NOAA API in SAS is a crucial step in leveraging weather information for analytics and decision-making. By following the steps outlined in this paper users can efficiently access and process weather data for various analytical purposes, such as understanding the impact of weather on sales, optimizing supply chain logistics, or predicting consumer behavior. Additionally, the use of SAS Base, SAS Macro, data steps, and PROC HTTP facilitates integration of weather data into existing SAS workflows, empowering organizations to derive actionable insights from weather analytics.

## Disclaimer

The code snippets provided in this paper are for illustrative purposes only and will require customization based on specific requirements and environments. Users are advised to refer to official documentation and consult with SAS experts for optimal implementation. Additionally, users must ensure proper authorization and compliance with NOAA API usage policies by obtaining a valid token for accessing the API.

Key Words: #NOAA-API #SAS #firstanalytics



FIRST ANALYTICS®

# About First Analytics

First Analytics designs and implements predictive analytics and machine learning solutions. We span multiple industries and applications. With an enabling engagement model, we team up with our clients to build their in-house capabilities and systems.

We have a long and successful history of helping our clients with analytics, always with a data driven approach. Visit our website to view our offerings and case studies and to learn more about how we can help your company leverage the power of advanced analytics with statistical modeling, machine learning and artificial intelligence.