ORDNANCE SURVEY GB

Getting Started with OS APIs in Power BI

**Copyright notice**

© Ordnance Survey Limited (2023). All rights reserved.

No part of this work may be reproduced, stored, or transmitted in any form, by electronic means or otherwise, without the written permission of Ordnance Survey Limited.

Disclaimer: Unless otherwise stated, the content in this work is provided for general information only and it is not intended to amount to advice on which the reader should rely. Ordnance Survey therefore disclaims all liability and responsibility arising from any reliance placed on such content. Professional or specialist advice should be obtained before taking, or refraining from, any action on the basis of the content herein.

**Version history**

| Version | Date | Description |
| --- | --- | --- |
| 1.0 |  | Initial release. |
|  |  |  |

**Contact details**

[OS website ‘Contact us’ page](https://www.ordnancesurvey.co.uk/contact-us) (<https://www.ordnancesurvey.co.uk/contact-us>).

Or contact [ollie.bowden@os.uk](mailto:ollie.bowden@os.uk) for technical queries.

Contents

[1. Introduction 4](#_Toc126681176)

[2. Mapbox 5](#_Toc126681177)

[3. Icon Map 6](#_Toc126681178)

[4. R – Custom Visualisations 8](#_Toc126681179)

[5. ArcGIS Maps for Power BI 13](#_Toc126681180)

[6. Cadcorp SIS WebMap for Power BI 15](#_Toc126681181)

[7. Power Query- OS Places API 16](#_Toc126681182)

[8. Python – OS Places API 20](#_Toc126681183)

[9. R – OS Places API 22](#_Toc126681184)

1. Introduction

**Overview**

This is the Getting Started Guide for using OS APIs within Power BI. There are a number of ways to get the OS APIs in Power BI and the best will depend on your own skills and goals. The options broadly fall into three categories: use a coding language such as R or Python, use power query within Power BI or use a custom visual from the visual store. Visuals are simply add-ons for Power BI that give you a custom visualisation or the ability to bring in other data types more easily.

The APIs covered in this guide are the OS Maps API & OS Vector Tile API for the purposes of background mapping and the OS Places API. The following ways of integrating the APIs will be covered:

* Mapbox Visual - a visual used to get the OS Vector Tile API integrated in Power BI as background mapping.
* Icon Map - a visual used to get the OS Maps API integrated in Power BI as background mapping.
* R Custom Visuals - a method used to create a custom visual using OS Maps API and the programming language R.
* ArcGIS Maps for Power BI - an ESRI visual using ArcGIS to integrate OS Maps API with power BI.
* CadCorp - a CadCorp visual used to integrate OS APIs with Power BI.
* Power query – use the built in power query functions to bring in and page OS places API.
* Python – use a python script to bring in OS Places API.
* R- use an R script to bring in OS Places API.

1. Mapbox

A picture containing graphical user interface

Description automatically generatedMapbox works with the OS Vector Tile API. First, bring in some data into Power BI or the visual will not work. Then, click on import visual and find the Map Box Visual:

Graphical user interface, application, Word

Description automatically generated

The Mapbox Visual will now appear in your build visual options tab. Click on it to bring in the visual and settings page. Add your longitude, latitude and location (this can be any field) and the visual will outline instructions on getting your Mapbox access token. You may need to set your latitude and longitude to ‘do not summarise’.

A picture containing logo

Description automatically generated

Navigate to the format visualisations tab and add in the access code. Change the Map Style to custom. Add in your OS Vector Tile API URL into the style and Mapbox API URL. An example of the URL is <https://api.os.uk/maps/vector/v1/vts/resources/styles?key=xxxxxxxsrs=27700>. This is obtained from the Data Hub (<https://osdatahub.os.uk/docs/vts/overview>).

You will now have a visual with the OS Vector Tile API in your dashboard:

1. Icon Map

Icon map works with the OS Maps API. First, bring in some data into Power BI or the visual will not work. Then, click on import visual and find the Icon Map Visual:

Graphical user interface, application, Word

Description automatically generated

A picture containing graphical user interface

Description automatically generated

The Icon Map Visual will now appear in your build visual options tab. Click on it to bring in the visual and settings page. Add your longitude, latitude and category. Then navigate to the format visualisation tab and add in the OS Maps API URL in the WMS Layer Tab. An example of the URL is [https://api.os.uk/maps/raster/v1/zxy/Road\_3857/{z}/{x}/{y}.png?key=xxxxxxxx](https://api.os.uk/maps/raster/v1/zxy/Road_3857/%7bz%7d/%7bx%7d/%7by%7d.png?key=xxxxxxxx). This is obtained from the Data Hub (https://osdatahub.os.uk/docs/wmts/overview).

Graphical user interface, application

Description automatically generated

You will now have a visual with the OS Maps API in your dashboard:

Map

Description automatically generated

1. R – Custom Visualisations

A custom visual for Power BI can be create using the R programming language. This requires the installation of node.js and will produce a html visual that is customisable with R, offering varying degrees of interactivity within powerBI. The following method was created by the Welsh Ambulance Service NHS Trust.

Install the JavaScript server node.js from [here](https://nodejs.org/). This will require administrator permissions on your machine.

Once installed, launch cmd prompt and navigate to a new folder (e.g. **cd C:\Users\obowden\OneDrive - Ordnance Survey\PowerBI**). From this location run this command in cmd prompt in your project directory (**npm install -g powerbi-visuals-tools**).

Check your installation by typing **pbiviz**within command line, which should return information on your install.

Text

Description automatically generated

Navigate to your desired working directory within command line (**e.g cd C:\Users\obowden\OneDrive - Ordnance Survey\Documents\PowerBI­**). From here create a new rhtml template with the command **pbiviz new testproject -t rhtml.**

Text

Description automatically generated

Within the newly created project folder, open the pbviz.json file in the template folder and populate the author, description, support URL & GitHub URL. If you do not have this information populate the fields with a dash.



The **script** source file can be altered to use different R scripts and create different visuals. An example developed by the Welsh Ambulance Service NHS Trust of a script that brings in OS Maps API and allows for switching between different styles. Update the script with your OS Maps API key obtained via the OS Data Hub and dataset with X Y coordinates, then and overwrite the contents of the script file within your project folder to test this.

#Built Using Microsoft R Open 4.0.2

#if("XML"            %in% rownames(installed.packages()) == FALSE) {install.packages("XML",)}

#if("leaflet"        %in% rownames(installed.packages()) == FALSE) {install.packages("leaflet",)}

#if("leaflet.extras" %in% rownames(installed.packages()) == FALSE) {install.packages("leaflet.extras",)}

#if("htmlwidgets"    %in% rownames(installed.packages()) == FALSE) {install.packages("htmlwidgets",)}

#if("htmltools"      %in% rownames(installed.packages()) == FALSE) {install.packages("htmltools",)}

#if("magrittr"       %in% rownames(installed.packages()) == FALSE) {install.packages("magrittr",)}

#if("sf"             %in% rownames(installed.packages()) == FALSE) {install.packages("sf",)}

library(XML)

library(leaflet)

library(leaflet.extras)

library(htmlwidgets)

library(htmltools)

library(magrittr)

library(sf)

library(rgdal)

#get my working directory

my\_dir <- getwd()

#import wast data from csv

dataset       <- read.csv("C:/Users/obowden/OneDrive - Ordnance Survey/PowerBI/R/sampleRHTMLVis/wmc\_hospital\_locations\_2021\_11\_18.csv")

#create a list of labels

marker\_labels <- lapply(seq(nrow(dataset)),

function(i) {paste0( '<p>',

"Name: ",dataset[i, "Name"], '<br>',    #Name of hospital taken from Name column

"Type: ",dataset[i, "Service"], '</p>' )#Type of hospital taken from Service column

})

marker\_icon <-  icons(iconUrl = ifelse(dataset$Service == "Major A&E",#Create a list of icons dependant on the Service column

"C:/Users/obowden/OneDrive - Ordnance Survey/PowerBI/R/sampleRHTMLVis/hospital-fill.png",

"C:/Users/obowden/OneDrive - Ordnance Survey/PowerBI/R/sampleRHTMLVis/hospital-line.png"),

iconWidth = 20, iconHeight = 20)

Sys.setenv(OS\_PROJECT\_API\_KEY = 'xxxxxxxxxxxxxxx') #API key for Ordnance Survey maps- INSERT OWN KEY

#Define url templates for the different tile styles

OSRoad\_3857     <- paste0("https://api.os.uk/maps/raster/v1/zxy/Road\_3857/{z}/{x}/{y}.png?key=",    Sys.getenv("OS\_PROJECT\_API\_KEY"))

OSLight\_3857    <- paste0("https://api.os.uk/maps/raster/v1/zxy/Light\_3857/{z}/{x}/{y}.png?key=",   Sys.getenv("OS\_PROJECT\_API\_KEY"))

OSOutdoor\_3857  <- paste0("https://api.os.uk/maps/raster/v1/zxy/Outdoor\_3857/{z}/{x}/{y}.png?key=", Sys.getenv("OS\_PROJECT\_API\_KEY"))

#Attribution for map tiles

OS\_Attribution  <- "© Crown copyright and database rights 2022"

#Create map

m <- leaflet(dataset) %>%

addTiles(OSRoad\_3857,

attribution = OS\_Attribution,

group ='OS Road') %>%

addTiles(OSLight\_3857,

attribution = OS\_Attribution,

group ='OS Light') %>%

addTiles(OSOutdoor\_3857,

attribution = OS\_Attribution,

group ='OS Outdoor') %>%

addMarkers(lng    = ~Longitude,                            #Taken from Longitude column

lat    = ~Latitude,                             #Taken from Latitude column

label  = lapply(marker\_labels, htmltools::HTML),#Label with information that appears when clicking on icon

popup  = lapply(marker\_labels, htmltools::HTML),#Label with information when you hover over icon

icon   = marker\_icon,                           #Icon displayed dependant on Service column

group  = "Icons") %>%

addCircleMarkers(lng    = ~Longitude,                            #Taken from Longitude column

lat    = ~Latitude,                             #Taken from Latitude column

radius = 10,                                    #Radius of circle

label  = lapply(marker\_labels, htmltools::HTML),#Label with information that appears when clicking on circle

popup  = lapply(marker\_labels, htmltools::HTML),#Label with information when you hover over circle

color  = ifelse(dataset$Service == "Major A&E", "red","orange"),#Circle colour dependant on value of Service column

group  = "Circles") %>%

addMarkers(lng   = ~Longitude,                            #Taken from Longitude column

lat   = ~Latitude,                             #Taken from Latitude column

label = lapply(marker\_labels, htmltools::HTML),#Label with information that appears when clicking on marker

popup = lapply(marker\_labels, htmltools::HTML),#Label with information when you hover over marker

group = "Markers") %>%

addLayersControl(baseGroups    = c("OS Road","OS Light","OS Outdoor"),

overlayGroups = c("Icons","Circles","Markers"),

position      = "bottomleft",

options       = layersControlOptions(collapsed = FALSE)) %>%

hideGroup("Circles")%>%

hideGroup("Markers") #Hides the markers so that it just defaults to circles

m

Navigate to the new project folder and run (**pbiviz package).**

Graphical user interface, text

Description automatically generated

This will create a custom visual within the dist folder of your project. Import this to PowerBI by selecting ‘Import’ and ‘Power BI Visual from File’ (you will have need to also go into the Power BI options and turn the R scripting on.)

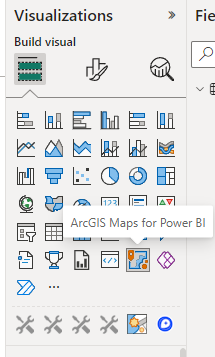
Map

Description automatically generated

1. ArcGIS Maps for Power BI

ArcGIS Maps for Power BI works with APIs due to its link to ArcGIS Online (AGOL). You will need an ESRI enterprise licence or an AGOL account to access this. Maps you have within your AGOL account can be imported into Power BI. This includes basemapping from OS Maps API if your organisation has set these up as an option with your enterprise account.

Add in an ArcGIS Maps for Power BI visualization.



This will display a sign in portal where you need to input your credentials.

Graphical user interface, application

Description automatically generated

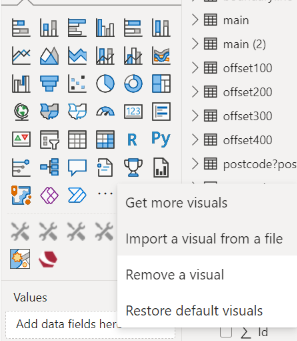
Once this is done you will be able to bring in basemaps and content from your account, including OS APIs you have as basemaps within AGOL



1. Cadcorp SIS WebMap for Power BI

The CadCorp custom visual can be used with the CadCorp SIS Webmap software to create pull in OS APIs. The data from your webmap can be pulled in using the custom visual, including the OS APIs. Information on adding the APIs to your webmap can be found here <https://help.cadcorp.com/en/9.0/webmapadmin/layers-osdatahub-wmts.htm?Highlight=OS%20maps%20api>.

To show the webmap in to Power BI click on import a visual from a file in the visualisation pane.



Point it to your CadCorp SIS Webmap for Power BI visual. This will appear in your visual tab. Bring this in to your report and add X Y data to activate the visual. Follow the instructions in the visual.

Graphical user interface, application

Description automatically generated

After adding your in service URL for your webmap, the map will appear in the report with any OS APIs you have included.

1. Power Query- OS Places API

There are multiple ways to bring OS Places API into Power BI. Each way must be able to page through the API results or only the first 100 features will be returned by the API call. This section will run through how to get OS Places API in Power BI using power query.

Bring in the data using your API URL. Include the offset parameter in your URL so it can be parameterised by Power Query at a later stage:

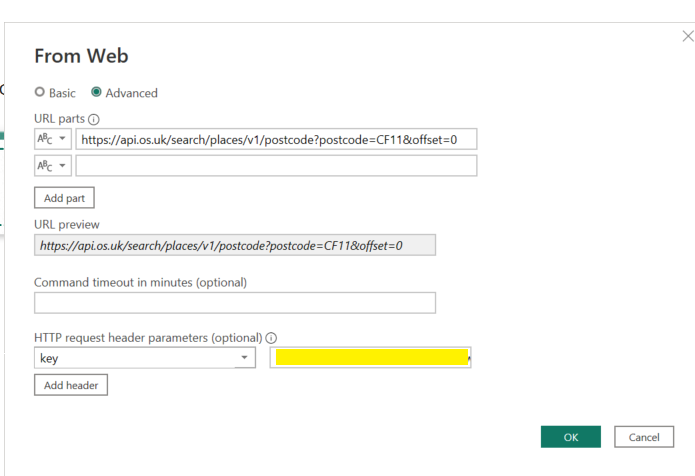
[*https://api.os.uk/search/places/v1/postcode?postcode=CF11&offset=0*](https://api.os.uk/search/places/v1/postcode?postcode=CF11&offset=0)

To do this use the get data option and choose web:

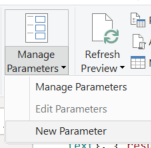
Graphical user interface, application

Description automatically generated

Choose the advanced option. Include your API key as a header with ‘key’ as the header request:



This will pull the first 100 results of the API call into Power Query. From here we want to parameterise the offset part of the URL. Under the home tab choose manage parameters and select new.

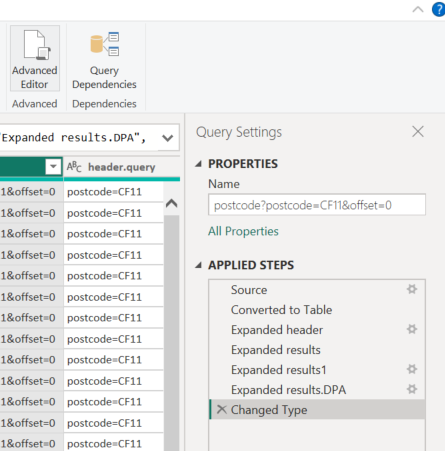


Call the new parameter offset, give it a type of text and set the current value to zero.

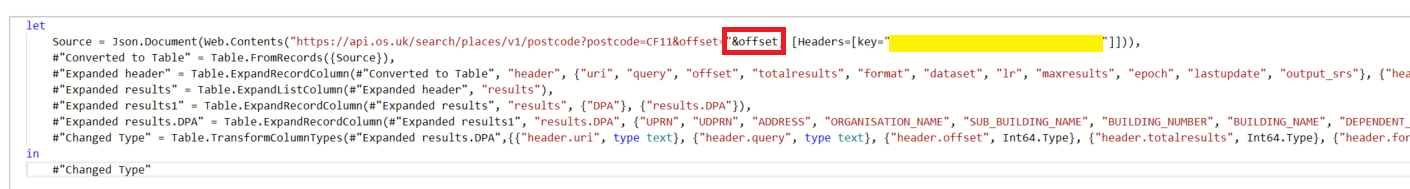
Graphical user interface, application

Description automatically generated

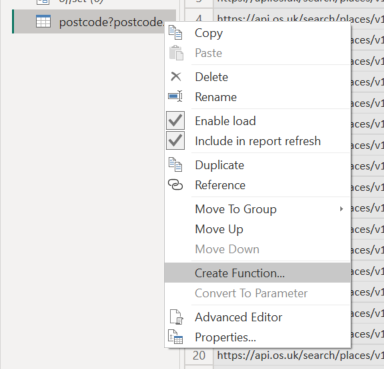
Your offset parameter will now appear in other queries. Next, navigate to the query that is calling and processing the OS Places API, click the view tab and choose advanced editor.



Include the offset parameter in the URL. As the current value is set to one the query should run as normal for now.



To be able to change the parameter, change the whole query to a function. Right click your query and choose create function, call it *fnpaging* or similar.



The function can be invoked by changing the offset parameter and clicking invoke. For example if we change the offset value to 100 the next 100 results will appear as a new query.

Graphical user interface, application

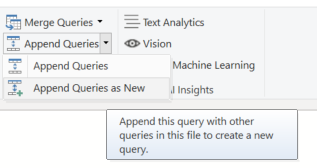
Description automatically generated

The function can also be called against a column with offset values (e.g 100, 200, 300, 400) to pull back all the results in one go. But for now, we will invoke the function three times with values 100, 200, 300 and 400 and rename the tables to something sensible.

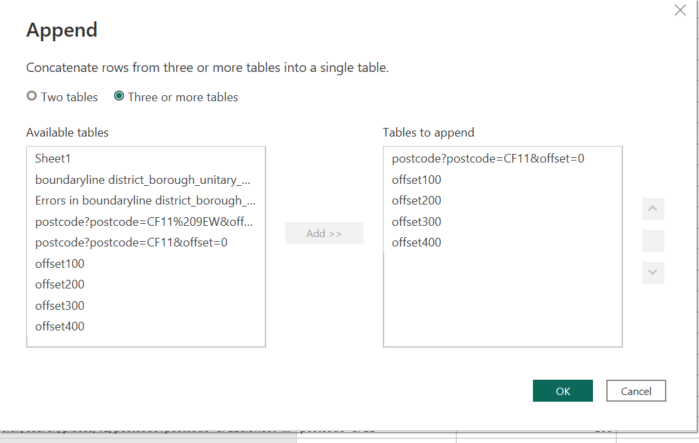
Graphical user interface, application, Word

Description automatically generated

Next, under the home tab choose append queries as new.



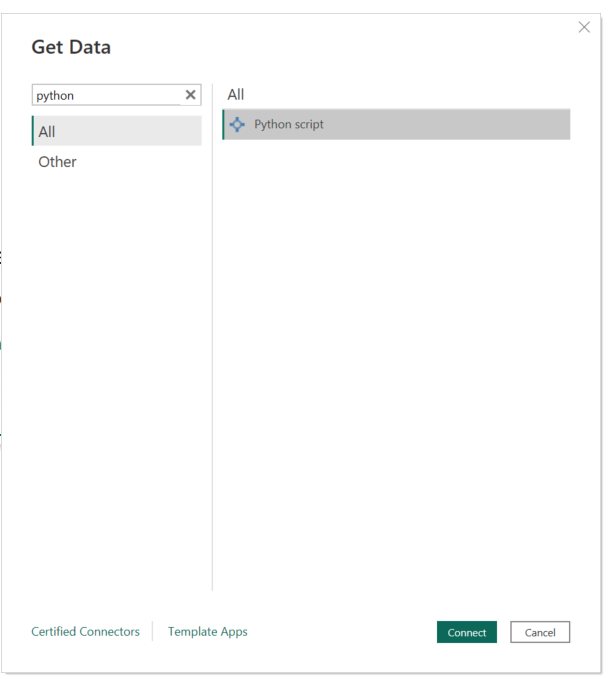
Select all of your offset queries and append them together.



Rename the appended query to ‘results’. In this case, the results table contains the first 500 features returned by the OS Places API for a particular postcode. More offset queries can be added to increase this amount.

1. Python – OS Places API

Choose get data and more. Search python script and select that option.



Paste the following script into the window, substituting your API key and your chosen parameters for the OS Places API call:

# Import packages

import pandas as pd

import requests as r

# Define API parameters

key = 'enter API key here'

query = 'enter OS Places query here (e.g EX4)'

url = f'https://api.os.uk/search/places/v1/find?query={query}&key={key}'

dataframe = pd.DataFrame()

# Make API requests with offsets and concatenate results

offsets = range(0, 1000, 100)

for offset in offsets:

    url = f'https://api.os.uk/search/places/v1/find?query={query}&offset={offset}&key={key}'

    response = r.get(url)

    response.raise\_for\_status()  # Check for errors

    results = response.json()['results']

    page = pd.json\_normalize(results)

    dataframe = pd.concat([dataframe, page])

# Delete page so only main dataframe is detected by PowerBI

del page

Load in the dataframe.

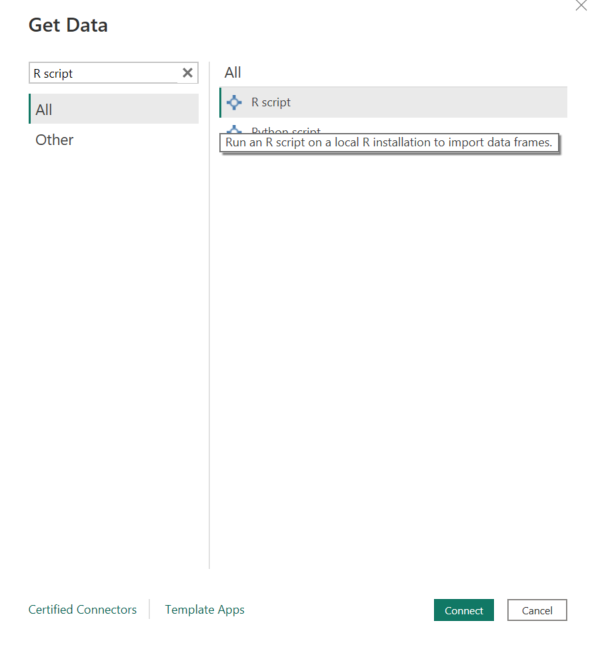
Graphical user interface, application

Description automatically generated

This will have loaded in the first 1000 results from the API call. The number of results can be adjusted by changing the maximum range value in the script.

1. R – OS Places API

Choose get data and more. Search R script and select that option.



Paste the following script into the window, substituting your API key and your chosen parameters for the OS Places API call:

# Import packages

library("httr")

library("jsonlite")

library('gtools')

# Define API parameters

key <- 'FzR8BJRJvs5f1w0FaGOQk5mdwQrOwBhe'

query <- 'CF11'

# Define empty dataframe and offset. Change second value to increase number of results.

offset= seq(0, 900, by=100)

dataframe = data.frame()

# Make API requests with offsets and concatenate results

for (value in offset) {

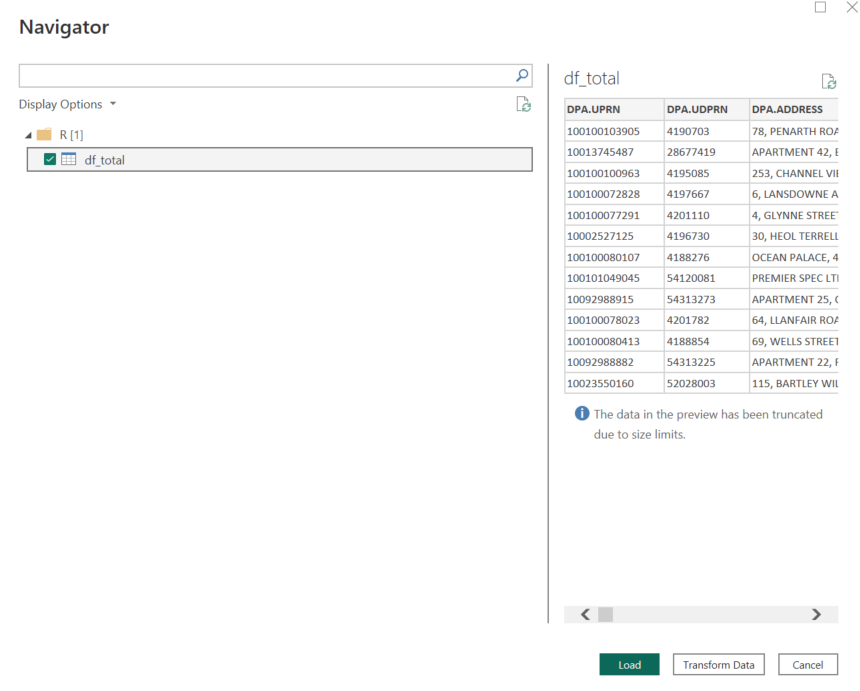
  url <- paste0('https://api.os.uk/search/places/v1/find?query=',query,'&offset=',value)

  data <- GET(url,add\_headers(key=key))

  json <- fromJSON(rawToChar(data$content), flatten=TRUE)

  dataframe <- smartbind(dataframe,json$results)

}



This will have loaded in the first 1000 results from the API call. The number of results can be adjusted by changing the maximum range value for offset in the script.