

Generating a Table and 3D Plot in R with Data from a JSON-LD file

A Tutorial by Daniel O’Neil

May 15th, 2018

Introduction

Several thousand code packages are available for the R programming language. A few packages provide functions for manipulating JavaScript Object Notation Linked-Data (JSON-LD), generating HyperText Markup Language (HTML) tables, and generating interactive 3D models. This tutorial provides a procedure for importing data into the Protégé ontology editor, exporting an expanded JSON-LD document, using an R package to convert the file format from expanded to compact, and using data from the compact JSON-LD file to generate an HTML table of planetary orbit parameters and interactive 3D plot the planetary orbits.

Applicable Documents

Protégé 5 Documentation, Stanford Center for Biomedical Informatics Research, Stanford University School of Medicine. <http://protegeproject.github.io/protege/>

Jeroen Ooms (2014). The jsonlite Package: A Practical and Consistent Mapping Between JSON Data and R Objects. arXiv:1403.2805 [stat.CO] URL <https://arxiv.org/abs/1403.2805>.

Jeroen Ooms (April 11, 2017). The jsonld Package: JSON for Linking Data. <https://cran.r-project.org/web/packages/jsonld/jsonld.pdf>

Daniel Adler, Duncan Murdoch (March 28, 2018). The rgl Package: 3D Visualization using OpenGL. <https://cran.r-project.org/web/packages/rgl/rgl.pdf>

Background

The tutorial about creating an expert system with the Protégé ontology editor explained how to configure Protégé for Individual Identifiers. As described in that tutorial, set the new entity International Resource Indicators (IRI) and Rendering settings to use the `rdfs:label`. These configuration settings ensure that NamedIndividuals receive the names from the imported data.

Johannes Kepler identified orbital parameters that define an elliptical trajectory of an object moving around another object. These orbital parameters include:

- *Semi-major Axis* – Half of the distance between the nearest and further points of the elliptical orbit.
- *Eccentricity* – This value is a ratio of the smaller width over the larger width of the ellipse.
- *Inclination* – The tilt of the orbit with respect to a reference plane.
- *Longitude of Ascending Node* – An angular measurement from a reference line to the ascending node. The ascending node is a point where the orbiting object ascends through the reference plane.
- *Argument of Periapsis* – An angular measurement from the Ascending Node to the point on the ellipse that is closest to the primary body.
- *True Anomaly* – The current position of the secondary body in orbit around the primary body.

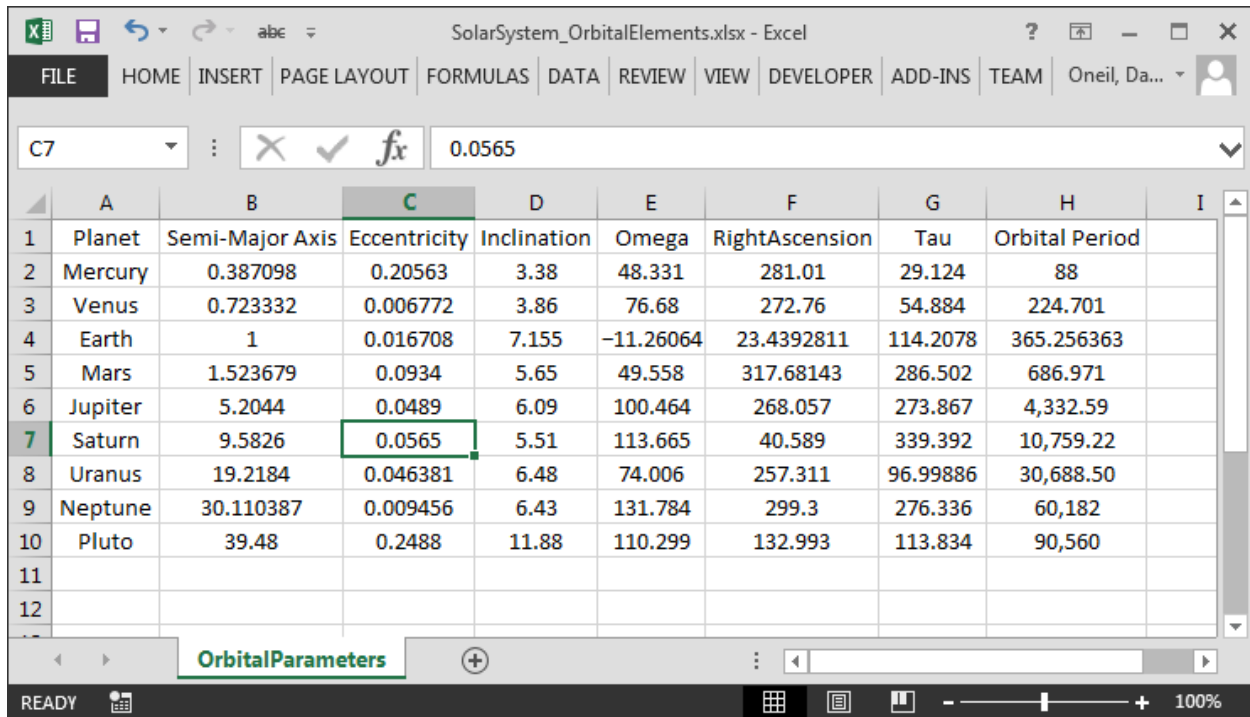
The following sections present a procedure for reading an Excel spreadsheet of orbital parameters into an ontology, exporting an expanded JSON-LD document, compacting the JSON-LD document, and writing R code to generate an HTML table and plot the orbital trajectories. Importing Orbital Parameters into Protégé.

Creating an Orbit ontology with Protégé

After configuring Protégé as described in the background section of this tutorial, create a class named Orbit. Click on the Data Properties tab and create the following data properties or variables: eccentricity, inclination, omega, period, raan, sma, and tau. Variable inclination defines the pitch orientation of an orbit. Variable omega will represent longitude of the ascending node, which will define the yaw orientation of the orbit. Variable raan will represent the right ascension of the ascending node, which define the roll orientation of the orbit. Variable sma represents the semi-major axis of an orbit. Variable tau represents the argument of periapsis, which can serve as a starting point for simulating an orbit. The next step of the procedure involves creating an Excel workbook with orbital parameter data and importing the data into the orbit ontology.

Importing Orbital Parameter Data into Protégé

Create an Excel workbook with orbital parameter data. Figure 1 depicts an Excel spreadsheet with orbital parameters for the planets in our solar system. Wikipedia articles about each planet provided the data for the spreadsheet.



| | A | B | C | D | E | F | G | H | I |
|----|---------|-----------------|--------------|-------------|-----------|----------------|----------|----------------|---|
| | Planet | Semi-Major Axis | Eccentricity | Inclination | Omega | RightAscension | Tau | Orbital Period | |
| 1 | Mercury | 0.387098 | 0.20563 | 3.38 | 48.331 | 281.01 | 29.124 | 88 | |
| 2 | Venus | 0.723332 | 0.006772 | 3.86 | 76.68 | 272.76 | 54.884 | 224.701 | |
| 3 | Earth | 1 | 0.016708 | 7.155 | -11.26064 | 23.4392811 | 114.2078 | 365.256363 | |
| 4 | Mars | 1.523679 | 0.0934 | 5.65 | 49.558 | 317.68143 | 286.502 | 686.971 | |
| 5 | Jupiter | 5.2044 | 0.0489 | 6.09 | 100.464 | 268.057 | 273.867 | 4,332.59 | |
| 6 | Saturn | 9.5826 | 0.0565 | 5.51 | 113.665 | 40.589 | 339.392 | 10,759.22 | |
| 7 | Uranus | 19.2184 | 0.046381 | 6.48 | 74.006 | 257.311 | 96.99886 | 30,688.50 | |
| 8 | Neptune | 30.110387 | 0.009456 | 6.43 | 131.784 | 299.3 | 276.336 | 60,182 | |
| 9 | Pluto | 39.48 | 0.2488 | 11.88 | 110.299 | 132.993 | 113.834 | 90,560 | |
| 10 | | | | | | | | | |
| 11 | | | | | | | | | |
| 12 | | | | | | | | | |

Figure 1 Excel spreadsheet with planetary orbit parameters

The tutorial about creating a simple expert system with the Protégé editor included a section that explained how to write rules to import data using the transformation rule editor. Figure 2 depicts the Cellfie dialog box, which appears upon selection of the option Create Axioms from Excel workbook option on the Tools menu in Protégé. The transformation rule editor appears upon clicking the Add button on the Cellfie dialog box. When a spreadsheet contains headers in the first row, ensure that the starting field in the transformation rule editor specifies row two, so the headers will be skipped when importing the data. Figure 2 depicts the Cellfie dialog box after the transformation rules have been written and loaded. In the rule, the first line, Individual: @A* indicates to create a NamedIndividual for each row of column A. Line two of the rule specifies the Type, which for this tutorial is the Orbit class created in the previous section. The Facts section of the rule maps the columns to the data properties and specifies a data type. The last line of the rule specifies column A as containing the label for the NamedIndividuals. If the Renderor is configured to use labels, the NamedIndividuals will have the labels as their names. Figure 3 depicts the Individuals by Class tabbed window after importing the orbital parameter data. The next section explains how to compact an expanded JSON-LD file via R code.

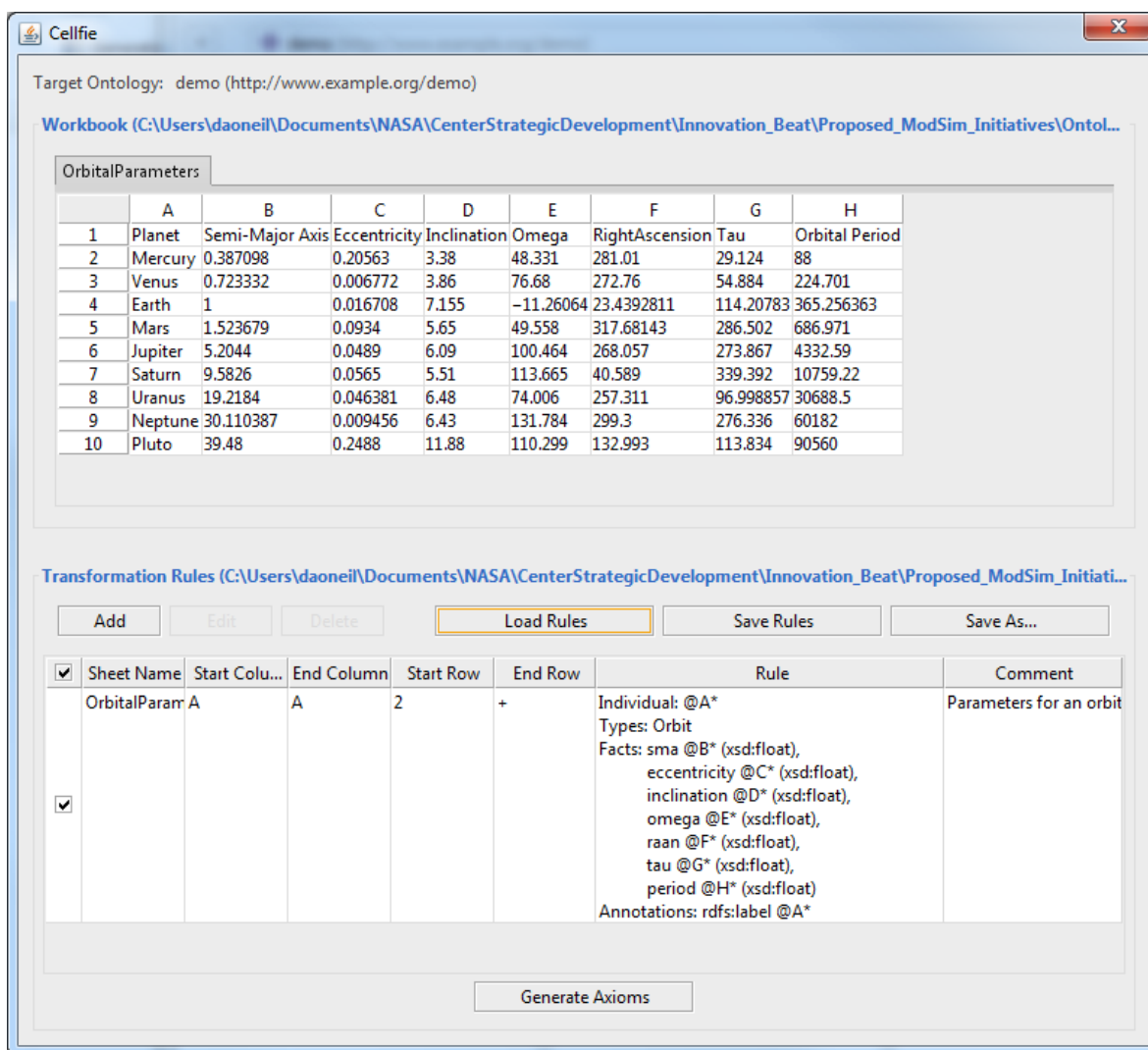


Figure 2 Cellfie dialog box, which appears upon selection of Create Axioms from Excel workbook

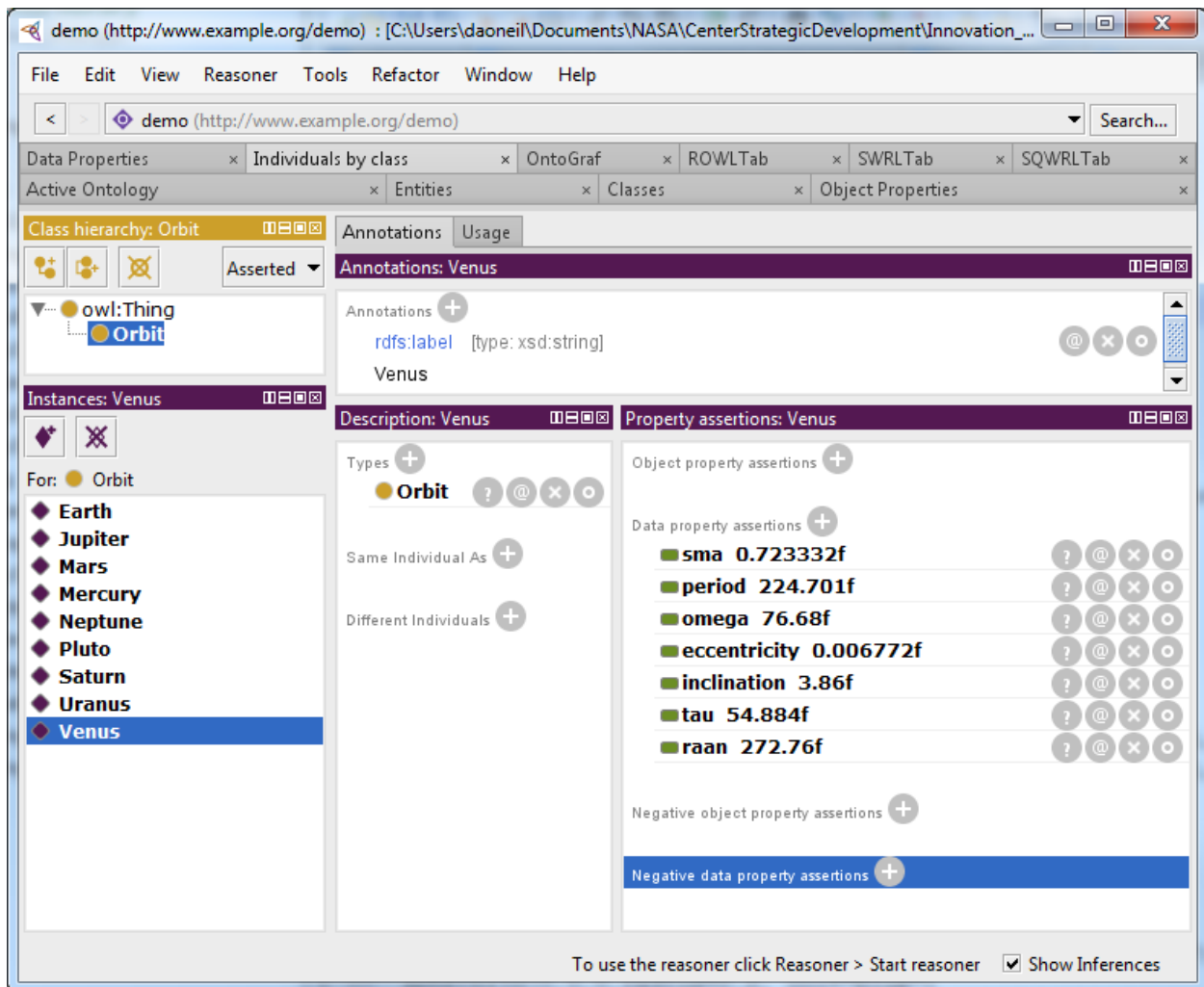


Figure 3 NamedIndividuals with populated DataProperties after importing the data

Compacting an Expanded JSON-LD Document

Protégé can save a file in the expanded JSON-LD format. Under the File menu, select the option to Save the file. When the Save dialog box appears, click the drop-down list button and select JSON-LD. Code Snippet 1 presents a few lines from a JSON-LD file saved from Protégé. Notice that each @type is followed by an explicit IRI. During compaction, these IRI will be replaced with a brief term. Another detail to notice is the value for omega, this value for Earth is supposed to be a negative number; evidently, there exists a problem with negative values in the JSON-LD export. In the next section, some R code will fix this problem.

```
{
  "@graph": [
    {
      "@id": "http://example.org/demo#Earth",
      "@type": [
        "http://example.org/demo#Orbit",
        "http://www.w3.org/2002/07/owl#NamedIndividual"
      ],
      "http://example.org/demo#eccentricity": {
```

```

    "@type": "http://www.w3.org/2001/XMLSchema#float",
    "@value": "0.016708"
  },
  "http://example.org/demo#inclination": {
    "@type": "http://www.w3.org/2001/XMLSchema#float",
    "@value": "7.155"
  },
  "http://example.org/demo#omega": {
    "@type": "http://www.w3.org/2001/XMLSchema#float",
    "@value": "\\u221211.26064"
  },
  "http://example.org/demo#period": {
    "@type": "http://www.w3.org/2001/XMLSchema#float",
    "@value": "365.25638"
  },
  "http://example.org/demo#raan": {
    "@type": "http://www.w3.org/2001/XMLSchema#float",
    "@value": "23.439281"
  },
  "http://example.org/demo#sma": {
    "@type": "http://www.w3.org/2001/XMLSchema#float",
    "@value": "1.0"
  },
  "http://example.org/demo#tau": {
    "@type": "http://www.w3.org/2001/XMLSchema#float",
    "@value": "114.20783"
  },
  "http://www.w3.org/2000/01/rdf-schema#label": "Earth"
}, ...

```

Code Snippet 1- A Few Lines from the Expanded JSON-LD Exported from Protege

A link in the Applicable Documents section leads to the `jsonld` package documentation. In that document, an example for the `jsonld_compact` function. Input values to the `jsonld_compact` function include an expanded JSON-LD document and a variable containing a character string that specifies a context that defines terms to substitute for each IRI. Code Snippet 2 presents content for a context variable. Notice that when a term is defined in a context, it can be used later in the context, e.g. notice how `demo` and `float` appear in term definitions. Another detail to notice in this snippet, is the defined term graph, which will replace the `@graph` in the compact JSON-LD format. Eliminating the `@` symbol will enable the R code to access the data.

```

{
  "demo": "http://example.org/demo#",
  "float": "http://www.w3.org/2001/XMLSchema#float",
  "owl": "http://www.w3.org/2002/07/owl#",
  "label": "http://www.w3.org/2000/01/rdf-schema#label",
  "comment": "http://www.w3.org/2000/01/rdf-schema#comment",
  "eccentricity": {
    "@id": "demo:eccentricity",
    "@type": "float"
  },
  "inclination": {
    "@id": "demo:inclination",
    "@type": "float"
  }
}

```

```

},
"omega": {
  "@id": "demo:omega",
  "@type": "float"
},
  "omega": {
    "@id": "demo:omega",
    "@type": "float"
  },
"sma": {
  "@id": "demo:sma",
  "@type": "float"
},
"raan": {
  "@id": "demo:raan",
  "@type": "float"
},
"tau": {
  "@id": "demo:tau",
  "@type": "float"
},
"period": {
  "@id": "demo:period",
  "@type": "float"
},
"graph": "@graph"
}

```

Code Snippet 2 Text for reading into a context object in R

Code Snippet 3 presents R code to load the `jsonld`, `jsonlite`, and `htmltable` libraries. Before these libraries can be loaded the packages must be installed. The command `install.packages()` will open the R documentation about installing packages. After the libraries are loaded the line `fname <- "justContext.txt"` provides the name of the text file containing the content presented in Code Snippet 2. Notice there is no pathname, which means that the file exists within the working directory used by the R interpreter. The R command, `setwd()`, changes the working directory to a specified folder. The next line in Code Snippet 3 reads the contents of the specified file as a character string. The second input parameter of the `readChar` command gets the size of the file to determine the length of the character string. With the Code Snippet 2 content stored in the variable named `orbContext`, the R code in Code Snippet 3 can call the `jsonld_compact` function to apply the context to compact the expanded JSON-LD document.

```

library(jsonlite)
library(jsonld)
library(htmlTable)

fname <- "justContext.txt"
orbContext <- readChar(fname, file.info(fname)$size)

compactOrbits <- jsonld_compact("PlanetaryOrbits.owl", orbContext)

```

Code Snippet 3 R code that reads the context character string and compacts the JSON-LD document

The JSON-LD Playground web application enables the development of a context section to compact an expanded JSON-LD file. If you use the JSON-LD Playground to compact the file then the last three lines of Code Snippet 3 can be skipped and the fromJSON function from the jsonlite library can read the compact JSON-LD file into a variable.

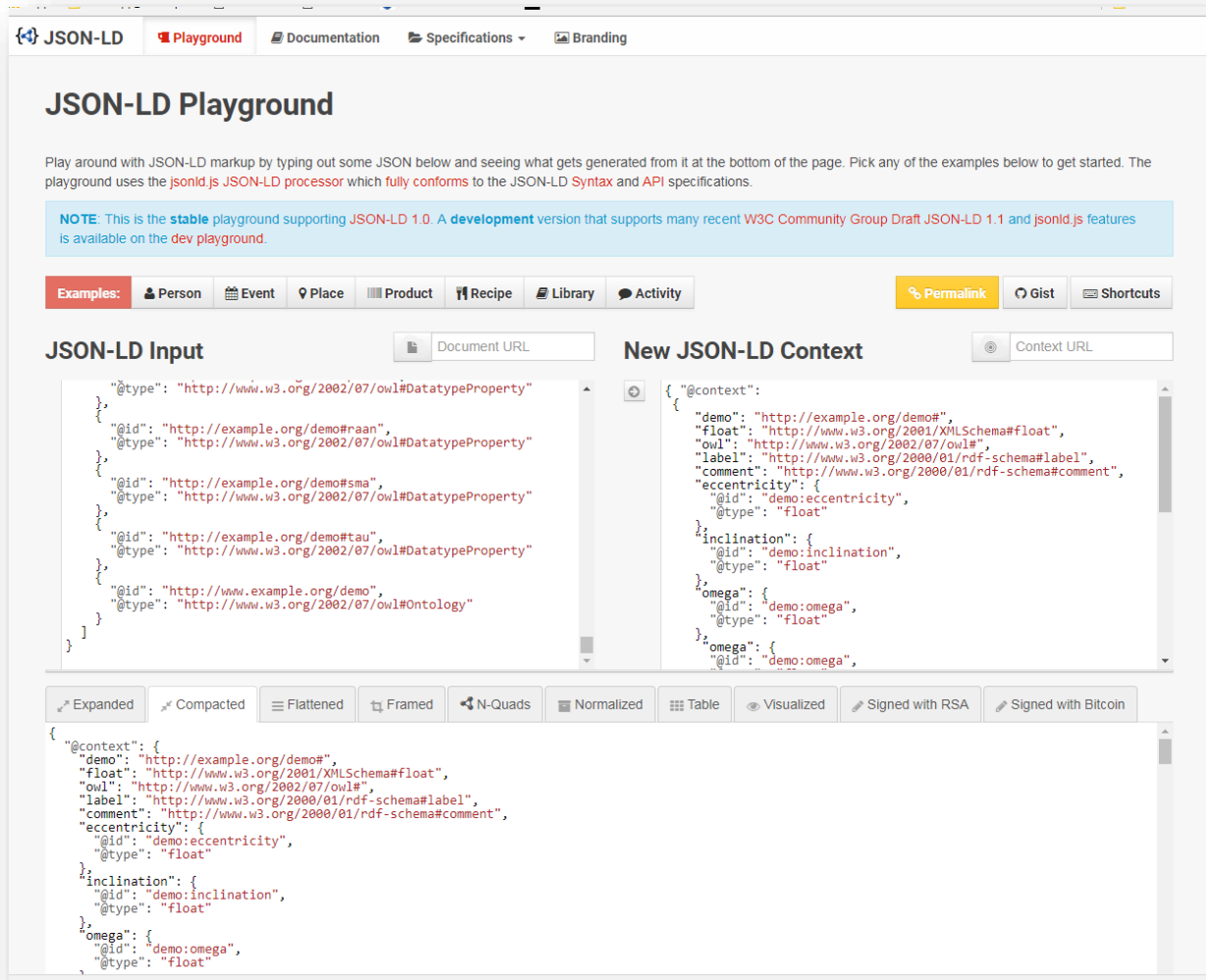


Figure 4 JSON-LD Playground with a New JSON-LD Context field

Figure 4 presents a screenshot of the JSON-LD Playground with the New JSON-LD Context field for defining terms to replace the IRI. Text in the New JSON-LD Context field is the same as the text in Code Snippet 2. A first line { "@context:" : has been added along with a corresponding closing curly bracket. As you develop the context, you can see the IRIs being replaced in the Compacted frame of the JSON-LD Playground web-page. With a compact JSON-LD document, R code can read and process the data. The next two sections present code that generates an HTML table and a 3D plot.

Generating an HTML Table

With the jsonlite and htmltable packages installed, it only takes a few lines to produce an HTML table from a compact JSON-LD document. Code Snippet 4 presents R code that uses the fromJSON function load the compact JSON-LD into a variable named orbits. Recall that the last line of Code Snippet 2 defined the term graph to eliminate the @ symbol. The second line of Code Snippet 4 extracts columns three through ten from orbits\$graph and stores the matrix in a variable named orbitparms. The na.omit function removes the Not Available (NA) data from the matrix.

Recall the problem with the negative number in Code Snippet 1; a line in Code Snippet 4 replaces the value in the matrix to fix that problem. Notice the comment about determining the coordinates. Typing the name of the variable orbitparms at the command prompt lists the table so you can determine the coordinates of the number that needs to be replaced. Values in the orbitparms matrix are characters, so an input parameter in the order function converts the characters to numbers via the as.numeric function. The specified column for the ordering is the semi-major axis (sma). The htmlTable function formats the matrix as HTML. The next to last line establishes a file connector and the writeLines function writes the HTML to a file. Table 1 was copied from the generated HTML file as it was rendered in a web browser and pasted in to this document.

```
orbits <- fromJSON(compactOrbits,flatten=TRUE)
orbitparms <- orbits$graph[,3:10]
orbitparms <- na.omit(orbitparms)
# Ensure that the columns are in the correct order.
orbitparms <- orbitparms[c("label","eccentricity","inclination",
                           "omega","period","raan","sma","tau")]
orbitparms # print the variable to the screen
orbitparms[1,4] <- -11.26064 # determine the coordinates (it may not be 1,4)
orbitparms <- orbitparms[order(as.numeric(orbitparms$sma)),]
orbparmTable <- htmlTable(orbitparms,rnames=FALSE)
fcon <- file("orbitingplanets.html")
writeLines(orbparmTable,fcon)
```

Code Snippet 4 R code for compacting the JSON-LD and generating an HTML table

Table 1HTML Table generated from the R code

| label | eccentricity | inclination | omega | period | raan | sma | tau |
|--------------|---------------------|--------------------|--------------|---------------|-------------|------------|------------|
| Mercury | 0.20563 | 3.38 | 48.331 | 88.0 | 281.01 | 0.387098 | 29.124 |
| Venus | 0.006772 | 3.86 | 76.68 | 224.701 | 272.76 | 0.723332 | 54.884 |
| Earth | 0.016708 | 7.155 | -11.26064 | 365.25638 | 23.439281 | 1.0 | 114.20783 |
| Mars | 0.0934 | 5.65 | 49.558 | 686.971 | 317.68143 | 1.523679 | 286.502 |
| Jupiter | 0.0489 | 6.09 | 100.464 | 4332.59 | 268.057 | 5.2044 | 273.867 |
| Saturn | 0.0565 | 5.51 | 113.665 | 10759.22 | 40.589 | 9.5826 | 339.392 |
| Uranus | 0.046381 | 6.48 | 74.006 | 30688.5 | 257.311 | 19.2184 | 96.998856 |
| Neptune | 0.009456 | 6.43 | 131.784 | 60182.0 | 299.3 | 30.110388 | 276.336 |
| Pluto | 0.2488 | 11.88 | 110.299 | 90560.0 | 132.993 | 39.48 | 113.834 |

Generating an Interactive 3D plot of the Planetary Orbits

With the orbital parameters stored in an R variable, additional code can generate sets of 3D coordinates and plot lines that represent the planetary orbital trajectories. Code snippets presented in this section generate sets of 3D coordinates for an ellipse and use functions from the `rgl` R package to rotate the ellipse by the angles specified in the variables inclination, omega, and raan, and plot the orbital trajectories.

```
library(rgl)          # Load the rgl library for 3D rotations and plotting.

makeOrbit <- function(a,e,oI,aN,aP) {
  oI <- oI * 0.01745329      # convert orbital inclination to radians
  aN <- aN * 0.01745329      # convert longitude of ascending node to radians
  aP <- aP * 0.01745329      # convert argument of perihelion to radians
  sLR = a * (1 - e^2)        # Compute Semi-Latus Rectum.

  theta <- seq(-pi,pi, length.out=80)
  r = sLR/(1 + e * cos(theta)) # Compute radial distance.
  x <- r * (cos(aP + theta) * cos(aN) - cos(oI) * sin(aP + theta) * sin(aN))
  y <- r * (cos(aP + theta) * sin(aN) + cos(oI) * sin(aP + theta) * cos(aN))
  z <- r * (sin(aP + theta) * sin(oI))

  points <- xyz.coords(x,y,z)

  helioorbit <- list(coordinates = points )
  return(helioorbit)
}

positSphere <- function (planet, orbitnum, size, color) {
  # Produce a sequence of potential true anomaly values.
  theta <- seq(-pi,pi, length.out=80)
  tau <- as.numeric(orbitparms[orbitnum,8]) * (pi/180)
  tau <- tau - pi # adjust tau to be in range of theta
  # Find the theta that is closest to tau.
  diffThetaTau <- abs(theta) - abs(tau)
  trueAnomaly <- which(diffThetaTau == min(diffThetaTau))
  taX <- planet$coordinates$x[trueAnomaly]
  taY <- planet$coordinates$y[trueAnomaly]
  taZ <- planet$coordinates$z[trueAnomaly]
  sphereid <- spheres3d(xyz.coords(x=taX,y=taY,z=taZ),
                        radius = size, col = color)
}

planetaryOrbit <- function(kp,n) { # Keplerian parameters and planet number
  a <- as.numeric(kp[n,7]) # semi-major axis
  e <- as.numeric(kp[n,2]) # eccentricity
  oI <- as.numeric(kp[n,3]) # orbital inclination
  aN <- as.numeric(kp[n,6]) # right ascension of ascending node
  aP <- as.numeric(kp[n,4]) # argument of perihelion

  helioOrbit <- makeOrbit(a,e,oI,aN,aP)
  return(helioOrbit)
}
```

Code Snippet 5 R code for generating and rotating an ellipse

Code Snippet 5 presents functions for generating coordinates for an ellipse and rotating the ellipse coordinates. Input variables for the function `makeOrbit` include the semi-major axis (`sma`), eccentricity, inclination, omega, and raan; the function generates 80 points and returns a set of 3D coordinates. The `positSphere` function generates 80 points between negative and positive Pi. The tau column in Table 1, represents a starting position for the planets within their orbital trajectories. After converting tau to radians and adjusting it to be in the same range as the theta values, the `positSphere` function finds the closest value of theta to tau and gets the index of the array. That index, `trueAnomaly` is used to get the 3D coordinates for the position of a sphere. Inputs to the `positSphere` function include the size and color for the planet. The `planetaryOrbit` function uses the Keplerian parameters from the `orbitparms` table and a planet number to get the data, convert it from characters to numbers, and calls the `makeOrbit` function.

```
# Open a 3D plotting device that has no labels, box, or axes.
decorate3d(xlab="", ylab="", zlab="", box=FALSE, axes=FALSE)
# Establish arrays of colors and sizes for the planets.
orbcolors <- c("brown", "gray", "blue", "red", "orange", "darkorchid",
              "cadetblue", "aquamarine4", "blue4")
orbsizes <- c(0.015, 0.04, 0.04, 0.03, 1, 0.8, 0.7, 0.6, 0.02)

planets <- planetaryOrbit(orbitparms, 1) # Generate trajectory of planet 1.

for(i in 2:9) { # Generate the other trajectories.
  planets <- c(planets, planetaryOrbit(orbitparms, i))
}
for(i in 1:9) { # Plot the trajectories and position the planets.
  plot3d(planets[i]$coordinates, add=TRUE, type="l", col=orbcolors[i])
  positSphere(planets[i], i, orbsizes[i], orbcolors[i])
}
library(htmltools) # load htmltools for the save_html function
orrery <- scene3d() # Store the plot as a scene.
orbwidget <- rglwidget(orrery) # Convert the scene to an rgl widget.
save_html(orbwidget, "orrery.html") # Save the widget as a web page.
```

Code Snippet 6 R code to generate 3D plot of the elliptical orbits

Code Snippet 6 opens a 3D plotting device with the `decorate3d` function from the `rgl` library that was loaded at the top of Code Snippet 5. Parameters of the `decorate3d` function turn off labels, a bounding box, and axes. The next two lines of Code Snippet 6 establish color and size arrays for the spheres that will represent the planets. A list object receives a generated planetary trajectory comprised of 3D coordinates. Subsequently, a for loop appends the other planetary orbital trajectories. A second for loop increments an index to plot the planetary orbital trajectories and the colorful spheres that represent the planets. Loading the `htmltools` library enables use of the `save_html` function at the end of the snippet. The `scene3d` and `rglwidget` functions, from the `rgl` library store the plot as a scene and converts that scene to a widget. The `save_html` function generates an HTML page that presents an interactive 3D plot. The next section explains how to integrate the table and the 3D plot into a single web page.

Integrate the HTML files with a Text Editor

Code Snippet 4 presented R script that generated an HTML table. Code Snippet 6 presented R script that generated an HTML file with embedded scripts that call library functions to generate an interactive 3D model of the solar system. Integrating the two HTML files is simply a matter of opening both of them and copying and pasting the `<table>...</table>` HTML code into the body of the other HTML file between the `<script>` and `</body>` tags. Figure 5 presents the integrated web-page with the interactive 3D model and HTML table.

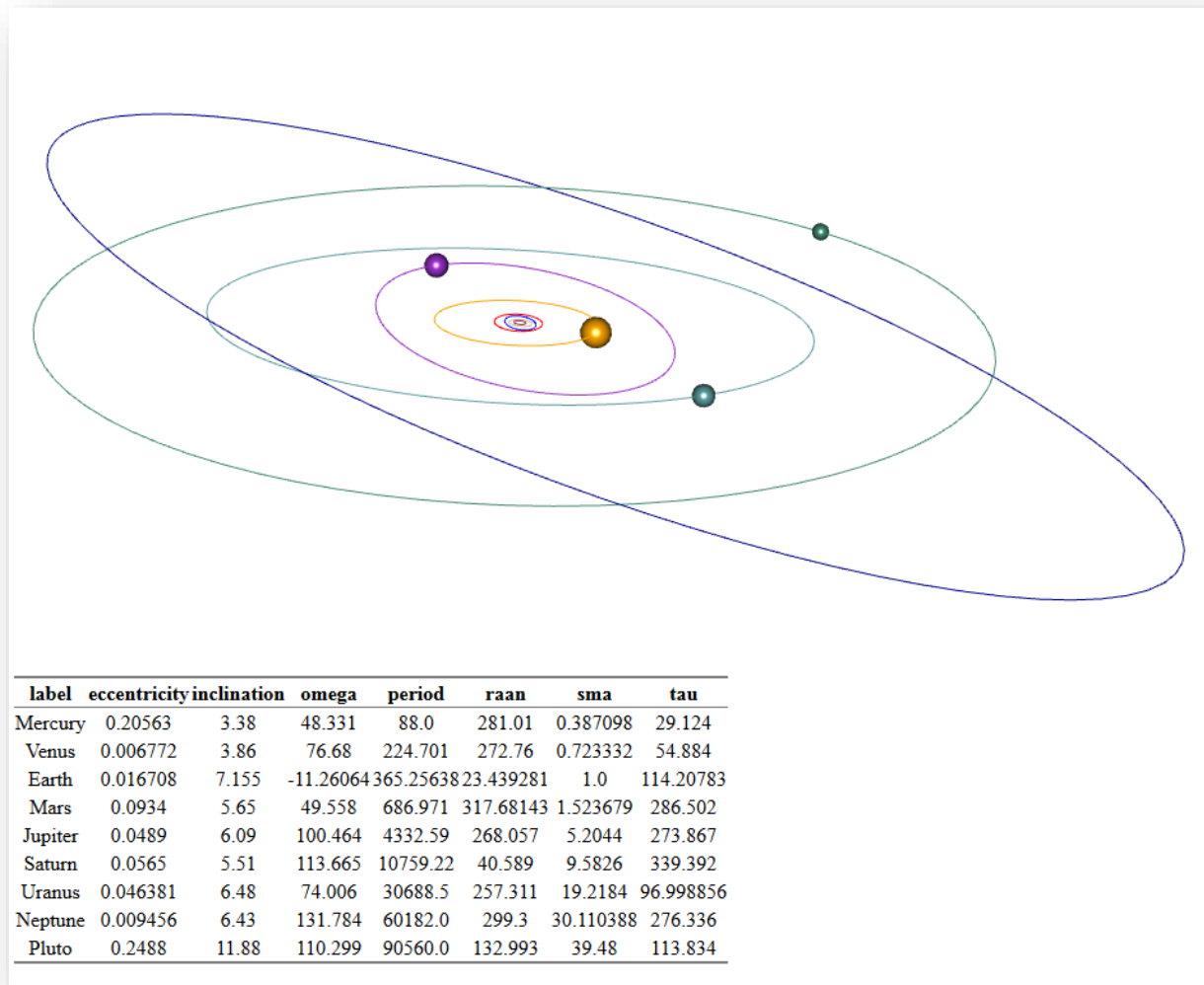


Figure 5 Integrated HTML table and interactive 3D model

Conclusion

This tutorial explained a procedure for importing data from Excel into Protégé, generating an expanded JSON-LD document from Protégé, two methods for compacting the JSON-LD file, and generating a HTML and an interactive 3D plot. With a capability read JSON-LD and post process the data into web pages with tables and interactive 3D plots, an ontology can manage mission data.