Rcaline: Modeling traffic-related pollution with R and the CALINE3 dispersion model

David Holstius

January 12, 2011

1 Introduction

Rcaline provides an interface to Fortran implementations of the CALINE family of line-source atmospheric dispersion models [1, 2]. These steady-state, Gaussian dispersion models are used to predict aerosol concentrations downwind from mobile emission source(s) such as highway traffic.

1.1 Features

This release of `Rcaline` provides an interface to a Fortran implementation, based on the original CALTRANS implementation, of the CALINE3 model. se.

Given the same inputs, Rcaline been tested to produce identical outputs. However, `Rcaline` removes significant limitations found in previous implementations of CALINE. For example, `Rcaline` can be used to model an arbitrary number of roadway links and an arbitrary number of receptors[1].

Because it is an R package, Rcaline also makes it much easier to work with contemporary data sources, such as ESRI shapefiles. You can use the included features of the R environment, or other third-party R packages, to visualize and export model results. The R environment also provides useful scripting capabilities for automating model runs. Some of these niceties are illustrated in the remainder of this document.

1.2 Limitations

The CALINE3 model is most appropriately used for modeling dispersion of carbon monoxide (CO) attributable to free-flow traffic with wind speeds greater than 1.0 m/s. [2] As with any model, care should be exercised to ensure that the practical application is theoretically well founded. For more on the theoretical scope and limitations of the CALINE model family, including terrain and other considerations, see [2].

[1] arbitray in the sense that these are bound only by available memory and CPU time

[2] The improved CALINE4 model includes adjustments for atmospheric chemistry that enable the modeling of nitrogen oxides (NOx). Support for the CALINE4 model is planned for a future release of Rcaline.

2 A Brief Example

In this section, we illustrate the use of `Rcaline` by applying it to actual highway data sourced from the OpenStreetMaps project [3] (Figure 1).

2.1 Importing roadway geometry

Because the CALINE3 model expects coordinates to be in a Cartesian reference frame, we need to reproject this data using `spTransform` . If your data are in geodetic coordinates (long-lat or lat-long), you'll need to reproject it too.

```
> library(rgdal)
> WestOakland <- system.file("extdata", "WestOakland",
      package = "Rcaline")
> highways <- spTransform(readOGR(WestOakland, layer = "highways"),
      CRS("+proj=utm +zone=10"))
```

```
OGR data source with driver: ESRI Shapefile
Source: "/var/folders/Qk/QkJL6MbEElqU2jIi9twTP++++TI/-Tmp-//Rtmp58sS3m/Rinst5c22aa5f/Rcali
with 175 features and 3 fields
Feature type: wkbLineString with 2 dimensions
```

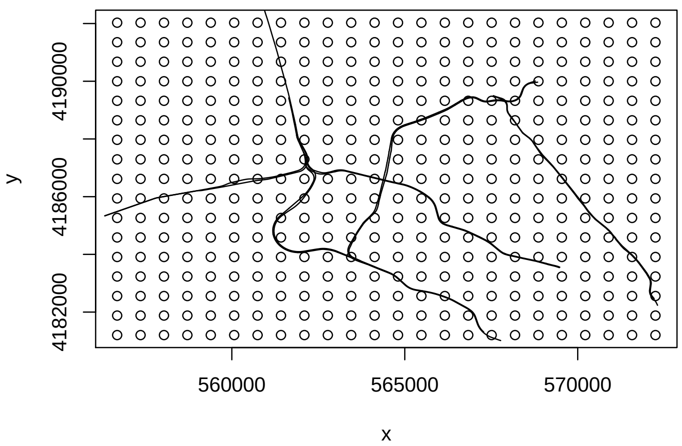

Figure 1: Example roadway geometry and receptor grid.

For this example, we're using `sp::sample.Spatial` to establish a grid of locations for which we want to predict traffic-attributable emissions. Receptor locations need not be regularly spaced. You might have a CSV file or other source of geographic locations if you are interested in estimating population-based exposures. If you do, make sure it is in the same (projected) coordinates as your roadway geometry. See the documentation for `caline3::receptor_totals` for more.

2.2 Decomposing roadway geometry into segments

In preparation for a CALINE model run, we need to break the roadway geometry up into individual segments. Polyline segmentation can be accomplished by the use of the `maptools::as.psp` function. This bit of code preserves the mapping between roadway segments and their parent polylines by way of the 'marks' attribute in the resulting dataframe. Segment coordinates are stored as x0, y0, x1, y1.

```
> library(spatstat)
> library(maptools)
> patterns <- lapply(highways@lines, function(x) as.psp(SpatialLines(list(x))))
> links <- do.call(rbind, lapply(patterns, as.data.frame))
```

Note that we haven't simplified the geometry: no vertices have been added, and none have been taken away. [3]

2.3 Traffic emissions and other attributes

We can use the `marks` attribute to assign to each segment the attributes associated with the polyline feature from which the segment was derived. This allows us to retain attributes like traffic volume, link elevation, number of lanes, or other data that might have been present in the shapefile.

```
> links <- merge(links, highways, by.x = "marks",
     by.y = "row.names")
> summary(links)
```

```
     marks            x0                 y0
 165    : 72   Min.   :556327   Min.   :4181042
 166    : 64   1st Qu.:562203   1st Qu.:4184190
 3      : 53   Median :564695   Median :4186210
 123    : 53   Mean   :564967   Mean   :4185986
 124    : 42   3rd Qu.:567213   3rd Qu.:4187399
 2      : 34   Max.   :572283   Max.   :4192376
 (Other):984
       x1                y1
 Min.   :556326   Min.   :4181013
 1st Qu.:562203   1st Qu.:4184190
 Median :564695   Median :4186210
 Mean   :564968   Mean   :4185989
 3rd Qu.:567213   3rd Qu.:4187399
 Max.   :572308   Max.   :4192473

                     NAME          LANES
 Nimitz Freeway        :308   Min.   :  2.000
 I 580                 :257   1st Qu.:  2.000
 Warren Freeway        :178   Median :  4.000
```

[3]Line simplification is a problem for which no globally optimal solutions exist, but if you have a technique that is compatible with your needs, go ahead and apply it here. It will speed up subsequent modeling computations.

```
Grove Shafter Freeway:120   Mean   :  3.289
I-980                : 91   3rd Qu.:  4.000
(Other)              :243   Max.   :  5.000
NA's                 :105   NA's   :942.000
     AADT
Min.   : 1700
1st Qu.: 6500
Median : 7200
Mean   : 7497
3rd Qu.: 8700
Max.   :13700
```

In this example, we're provided with an estimate of traffic volume: AADT (Annual Average Daily Traffic). However, we'll need to impute or assume default values for a few other variables.

```
> links[is.na(links$LANES), "LANES"] <- median(links$LANES,
     na.rm = TRUE)
> links <- transform(links, flow = AADT/24, emissions = 30,
     width = LANES * 10, classification = "AG",
     height = 0)
> summary(links)
```

```
    marks           x0                y0
165     : 72   Min.   :556327   Min.   :4181042
166     : 64   1st Qu.:562203   1st Qu.:4184190
3       : 53   Median :564695   Median :4186210
123     : 53   Mean   :564967   Mean   :4185986
124     : 42   3rd Qu.:567213   3rd Qu.:4187399
2       : 34   Max.   :572283   Max.   :4192376
(Other):984
      x1               y1
Min.   :556326   Min.   :4181013
1st Qu.:562203   1st Qu.:4184190
Median :564695   Median :4186210
Mean   :564968   Mean   :4185989
3rd Qu.:567213   3rd Qu.:4187399
Max.   :572308   Max.   :4192473

                   NAME         LANES
Nimitz Freeway       :308   Min.   :2.000
I 580                :257   1st Qu.:4.000
Warren Freeway       :178   Median :4.000
Grove Shafter Freeway:120   Mean   :3.803
I-980                : 91   3rd Qu.:4.000
(Other)              :243   Max.   :5.000
NA's                 :105
     AADT            flow            emissions
Min.   : 1700   Min.   : 70.83   Min.   :30
1st Qu.: 6500   1st Qu.:270.83   1st Qu.:30
Median : 7200   Median :300.00   Median :30
```

```
 Mean   : 7497   Mean   :312.38   Mean   :30
 3rd Qu.: 8700   3rd Qu.:362.50   3rd Qu.:30
 Max.   :13700   Max.   :570.83   Max.   :30

     width       classification     height
 Min.   :20.00   AG:1302        Min.   :0
 1st Qu.:40.00                  1st Qu.:0
 Median :40.00                  Median :0
 Mean   :38.03                  Mean   :0
 3rd Qu.:40.00                  3rd Qu.:0
 Max.   :50.00                  Max.   :0
```

We could use more sophisticated methods, of course. This example merely serves to illustrate the required inputs.

3 Meteorology

CALINE3 requires four variables corresponding to the prevailing meteorology:

- wind bearing;
- wind speed;
- Pasquill stability class; and
- mixing height.

Hourly values for these are sometimes available in the form of an "ISC-ready" input file, often with a .MET file extension. For this example, we'll just assume some reasonable values for a single hour. If you are interested in computing annual averages, 8-hour maximum concentrations, etc., it's quite easy to use the scripting capabilities of R to process many different scenarios and summarize the results.

```
> meteorology <- list(wind.bearing = 330, wind.speed = 1.5,
     stability.class = 4, mixing.height = 1000)
```

4 Running the model

Now that the roadway geometry, emissions, receptor locations, and prevailing meteorology have been established, we can run the model.

4.1 Model parameters

CALINE3 also requires several "job parameters", including:

- averaging time;
- surface roughness;
- pollutant settling velocity; and
- pollutant deposition velocity.

We'll use some common values for averaging time (60 min) and surface roughness (100 cm). When modeling carbon monoxide, it's conventional to specify the settling velocity and deposition velocity as 0.0 m/s. For more on these parameters, including reasonable ranges and representative values, see [1, 2] or the documentation for `receptor_totals` .

4.2 Computing receptor totals

The following code computes the predicted aerosol concentration, given the prevailing conditions and the parameters we've chosen.

```
> library(Rcaline)
> predicted <- CALINE3.predict(receptors, links,
     meteorology, averaging.time = 60, surface.roughness = 100)
> head(predicted)
```

```
           1            2            3            4
1.980743e-07 4.783360e-05 8.223038e-04 3.128347e-03
           5            6
5.060010e-03 5.624432e-03
```

References

[1] P.E. Benson. CALINE3: a versatile dispersion model for predicting air pollutant levels near highways and arterial streets. Interim report. Technical report, PB-80-220841, California State Dept. of Transportation, Sacramento (USA). Transportation Lab., 1979.

[2] P.E. Benson. A review of the development and application of the CALINE3 and 4 models. *Atmospheric Environment. Part B. Urban Atmosphere*, 26(3):379–390, 1992.

[3] M.M. Haklay and P. Weber. OpenStreetMap: user-generated street maps. *IEEE Pervasive Computing*, pages 12–18, 2008.