Using Rcaline: An Illustrated Example

David Holstius

January 8, 2012

1 Introduction

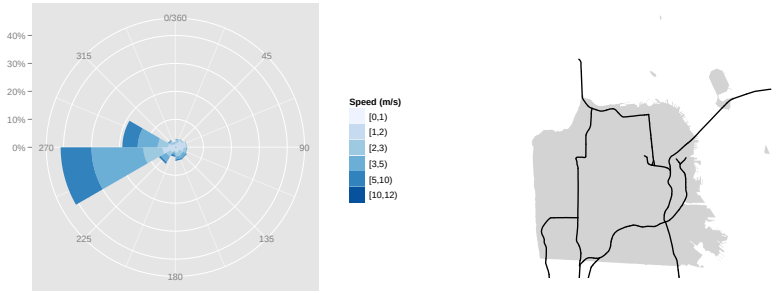
Here we apply `Rcaline` to San Francisco data provided by the Bay Area Air Quality Management District (BAAQMD). This example illustrates the basic steps of constructing, running, and visualizing a model with Rcaline.

2 Model construction

In this example, the traffic data and the meteorological data have already been imported. If you're unfamiliar with this step, please consult the accompanying vignette, *Importing Data for use with Rcaline*.

There are three objects in the imported data: an `ISCFile`, containing hourly meteorological records; a `SpatialPolygonsDataFrame`, representing the extent of San Francisco county; and a `SpatialLinesDataFrame`, which describes the roadway geometry and the Annual Average Daily Traffic (AADT) counts.

```
> require(Rcaline)
> data(SanFrancisco, package='Rcaline')
> ls()
[1] "met_5801"        "SF_county"       "STRte_BAAQMD_v3"
```



(a) Meteorology (b) Highway network

Figure 1: San Francisco data included with the `Rcaline` package.

2.1 Meteorology

The conventional way to supply hourly meteorological data is to use an "ISC-ready" meteorology file containing records of wind speed, wind bearing, atmospheric stability, and mixing height. We need only specify whether to use urban or rural mixing heights, as follows:

```
> met <- Meteorology(met_5801, use='urban')
> require(ggplot2)
> show(ggplot(met))
```

The hourly resolution ensures that we can adequately model the seasonal and diurnal variability. Figure 1a illustrates the joint distribution of wind speed and wind bearing; the wind here is predominantly coming from the south-west.

2.2 Roadways and traffic

Importing shapefiles (with a package like `maptools` or `rgdal`) is usually the easiest way to get GIS data into R. Figure 1b shows the San Franciso highway network after importing a shapefile with `rgdal`. The relevant data are contained in the variable `STRte_BAAQMD_v3`, which is an object of class `SpatialLinesDataFrame`.[1]

In CALINE terminology, a segment connecting two points is referred to as a *link*—although in other domains, sometimes entire polylines are referred to as links. Attributes in a `SpatialLinesDataFrame` correspond to entire polylines, but we need to use them to express attributes at the link (i.e., segment) level. That is what this does:

```
> lnk <- FreeFlowLinks(STRte_BAAQMD_v3,
          vehiclesPerHour = TRVol2009 / 24,
          width = 3.7 * LaneNum + 3.0 + 3.0,
          emissionFactor = 1.0)
> plot(SF_county, col="light gray", border=NA)
> lines(lnk, lwd=2)
```

We're also doing some basic transforms on the supplied data. The estimated Annual Average Daily Traffic (AADT) volume is supplied by the attribute `TRVol2009`. In order to convert it to *vehicles per hour*, we divide by 24. Likewise, we use the `LaneNum` attribute to impute road widths, assuming 3.7 meters per lane plus 3.0 meter shoulders.

2.3 Emission factors

Emission factors can also vary by link, just like traffic volume or road width. Slower stretches of road, for example, might merit a higher emission factor (since emissions vary with speed). However, since we don't have link-level data on average emission factors, we just supply a constant value.

[1] A `SpatialLinesDataFrame` represents a number of (possibly disjoint) *polylines*, each of which is composed of one or more connected segments. For more on common representations of spatial data in R, see the `sp` package documentation.

(a) Grid-based (b) Distance-based

Figure 2: Receptor locations sampled using two different methods.

Emission factors are always in *grams per vehicle per mile*. For the sake of this example, we use a constant "unit emission factor" of 1.0 across the entire highway network.

Conveniently, the emission rate calculated by CALINE3 for any given link (in grams per mile per hour) is exactly equal to that link's emission factor multiplied by its traffic volume.[2] Further, the contribution of any given link (to a downwind receptor) is directly proportional to that link's emission rate. This means that a dispersion surface calculated with a unit emissions factor can be linearly scaled to reflect different emissions scenarios—without needing to run the dispersion step again—as long as the emission factors and/or traffic volumes are scaled uniformly.

In California, trip-based emission factors for specific pollutants can be obtained from models like EMFAC [3, 4, 5]. There are subtleties in applying trip-based factors to a link-level model; we do not cover them here, but see [1] for a relevant discussion.

2.4 Receptors

Receptors are the locations at which the dispersed pollutant concentration will be calculated. CALINE3 is a steady-state Gaussian plume model, rather than a numerical grid-based model. Hence, it is up to us to determine the spatial distribution of the receptor locations. Essentially, we can do one of two things: (1) generate a set of locations algorithmically; or (2) supply a predefined set of locations.

2.4.1 Constructing a receptor grid algorithmically

Modelers often use a planar Cartesian grid (Figure 2a), so here we show how to generate one.[3] By default, receptor locations have an elevation (z-coordinate) of 1.8 m, corresponding to an "average" human height.

[2] This is not true of CALINE4.

[3] To inspect the code used by the `ReceptorGrid` function, just type `ReceptorGrid` and hit Enter.

```
> rcp <- ReceptorGrid(lnk, resolution=250, maxDistance=1e3)
> plot(SF_county, col="light gray", border=NA)
> lines(lnk)
> points(rcp, pch='+', cex=0.5)
```

Properly conceived, a set of receptor locations is a sample taken from a larger sampling frame. Because we do not expect significant concentrations beyond 1 km, here we are sampling only locations within 1 km of the roadways. However, sampling locations from a Cartesian grid can still result in apparent "hot spots" where grid locations happen to fall closer to some part of the roadway.

A different approach is to use the distance from the road network as a basis for sampling (Figure 2b). Conveniently, this approach also allows receptors to be packed more densely in the more interesting parts of the dispersion field—close to the roadway, where there are both higher concentrations and more variance. In practice, this results in a more efficient use of computational resources to produce a map with sufficient (not necessarily uniform) detail.

```
> rcp <- ReceptorRings(lnk, distances=c(100, 250, 500, 1000))
> plot(SF_county, col="light gray", border=NA)
> lines(lnk)
> points(rcp, pch='+', cex=0.5)
```

If one were interested in estimating aggregate exposures at a population level, we could also construct an irregular grid by sampling locations from predefined spatial units, like ZIP codes or census tracts. Sampling multiple locations could provide a useful way to estimate variance within such units.

2.4.2 Using pre-specified locations

Of course, receptor locations need not be constructed algorithmically. For example, you might want to compute predicted concentrations at a specific set of geocoded street addresses, corresponding to some cohort of interest. As long as you have these in a format that you can import into R, you can use them—just replace the `rcp` variable in this example with your `SpatialPointsDataFrame` or `SpatialPoints` object.

2.5 Other model parameters

Additional model parameters, including terrain and pollutant characteristics, also need to be specified. For detailed information, consult the CALINE3 User's Guide [2]. Here we supply some reasonable default values:

```
> ter <- Terrain(surfaceRoughness=80.0)
> CO <- Pollutant("Carbon monoxide",
          molecularWeight = 28.0,
          settlingVelocity = 0.0)
```

3 Predicting concentrations

The CALINE3 algorithm is CPU-intensive. Although it is beyond the scope of this example, you can use the `foreach` package to do the computations in parallel, using multiple cores or networked hosts. (A future vignette will illustrate this technique.)

3.1 Running the model

We use the `predict` method to actually run the model. Since the model will actually be run once for every meteorological condition we supply (N=8760), it can be quickest to use only a small sample for the first pass. Here we use 1% of the meteorology, sampled at random:

```
> mod <- Caline3Model(lnk, sample.rows(met, p=0.01), rcp, ter, CO)
> pred <- predict(mod, units='ppm')
```

The result of running the model is an $M \times N$ array, where M is the number of meteorological conditions and N is the number of receptors. Each cell indicates the concentration, in g/m^3, at that receptor during those conditions.

3.2 Computing summary statistics

By computing summary statistics such as the mean or maximum, we can treat the result as a sample from a theoretical annual distribution, and estimate its properties. The `aggregate` function computes several statistics by default. Casting the result to a `SpatialPointsDataFrame` re-binds the statistics to the receptor locations:

```
> agg <- aggregate(pred)
> spdat <- as(agg, 'SpatialPointsDataFrame')
> spdat[1:3, c('distance', 'mean', 'max')]
                    coordinates distance         mean
RECP. 1 (15033.7, 4188610, 1.8)      100 6.228928e-05
RECP. 2 (14989.9, 4188700, 1.8)      100 6.272246e-05
RECP. 3 (14959.8, 4188800, 1.8)      100 5.966218e-05
                max
RECP. 1 0.001041414
RECP. 2 0.001057207
RECP. 3 0.001062583
```

4 Analyzing results

After aggregation, we can select a statistic of interest and explore the distribution. Here we focus on exploring results graphically, although they could also be tabulated or subjected to statistical tests.

4.1 Upwind vs. downwind concentrations

Within the results is a variable, `distance`, that contains the distance-to-roadway for each receptor. (Recall that we specified these distances when constructing the receptor grid.) We can use this to divide the receptors into specific classes and explore the distribution of predicted concentrations in each.

```
> ggplot(aes(x=mean), data=as(spdat, 'data.frame')) +
        geom_histogram(binwidth=5e-2) +
        stat_density(aes(y=5e-2*..count..), color='red', geom='path') +
        facet_wrap(~ distance)
```

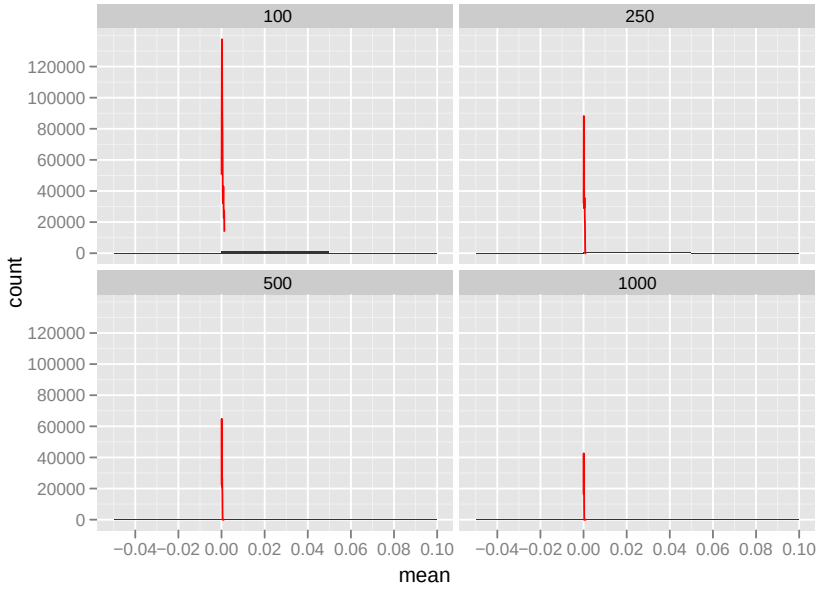
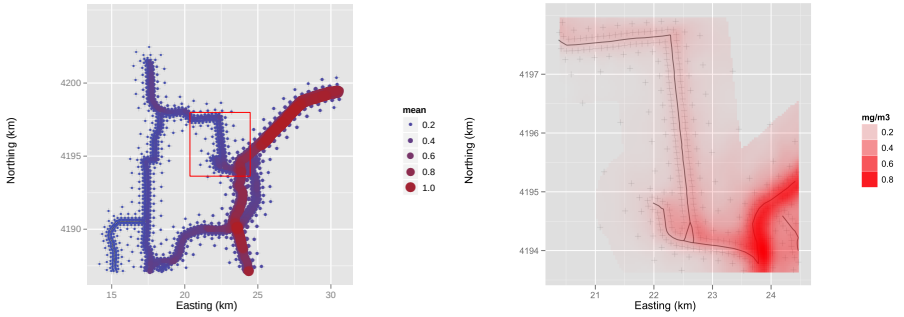


Figure 3: Mean predicted concentration, by distance-to-roadway.

4.2 Mapping



(a) Bubble plot

(b) Interpolated subregion

Figure 4: Estimated concentrations, with receptors shown as crosshairs.

4.3 Bubble plots

The `ggplot` package makes it easy to generate bubble plots (Figure 4a). First we define the bounds of the map (San Francisco county); then we add a `geom_point` layer, binding the size, color, and stacking order of the points to our variable of interest. (For more information on using `ggplot`, consult the `ggplot` documentation.) We also add a rectangle to highlight a sub-region of interest.

```
> bounds <- Rectangle(SF_county)
> map <- ggplot(agg, bounds=bounds)
> bubbles <- geom_point(aes(x, y, size=mean, color=mean, order=mean))
> region <- resize(bounds, 0.25)
> box <- geom_rect(aes(xmin=xmin, ymin=ymin, xmax=xmax, ymax=ymax),
          fill=NA, color='red', data=as(region, 'data.frame'))
> show(map + bubbles + box)
```

4.4 Interpolating back to a raster

As an alternative to the bubble plot, it's possible to construct a raster image. If we had started with receptor locations on a planar grid, this would be obvious. But even though we didn't, we can still interpolate a summary statistic from our receptor locations back to a planar grid (Figure 4b).

Here we intersect the region of interest defined above (red box, Figure 4a) with a 1 km buffer constructed around the highways. Then we sample the resulting sub-region with `spsample`, such that we obtain a regular grid having 10,000 points. This is our planar grid.

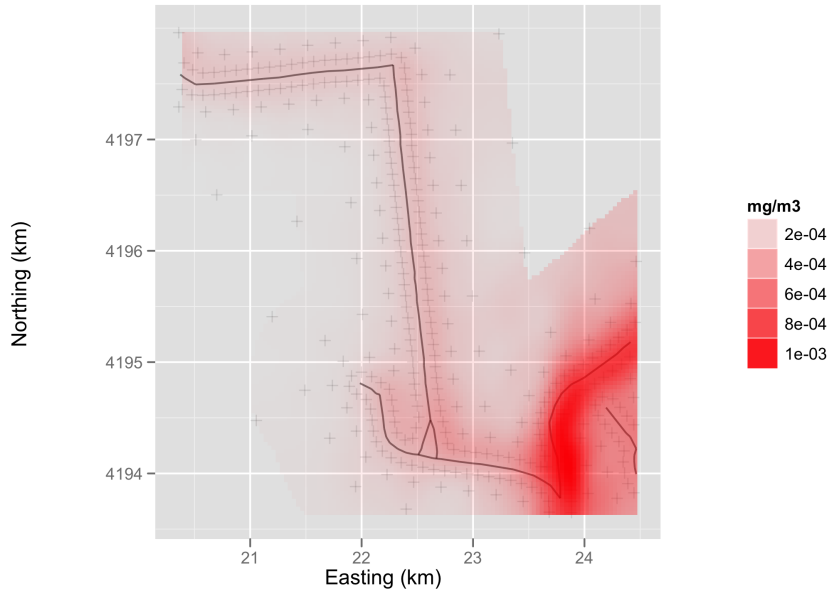
```
> buf <- gBuffer(centerlines(lnk), width=1e3)
> buf <- gIntersection(buf, as(region, 'SpatialPolygons'))
> grd <- spsample(buf, n=1e4, type='regular', offset=c(0.5, 0.5))
> coordnames(grd) <- c('x', 'y')
```

After the grid has been constructed, a method for interpolation must be selected. Here, we use multilevel B-splines [6], as implemented by the `MBA` package.

```
> require(MBA)
> obs <- cbind(coordinates(spdat)[,1:2], z=spdat$mean)
> fit <- mba.points(obs, coordinates(grd), verbose=FALSE)
> srf <- with(fit, as.data.frame(xyz.est))
```

Finally, we replace the primary plot data, using the `%+%` operator. Since our data are now regularly spaced, we substitute `geom_tile` for `geom_point`. We also change the extent of the map, so that we show the region of interest more clearly.

```
> map <- ggplot(agg, bounds=region) %+% srf
> show(map + geom_tile(aes(x, y, alpha=z), fill='red') +
         scale_alpha('mg/m3', to=c(0, 1)))
```



References

[1] S. Bai, Y.C. Chiu, and D.A. Niemeier. A comparative analysis of using trip-based versus link-based traffic data for regional mobile source emissions estimation. *Atmospheric Environment*, 41(35):7512 – 7523, 2007.

[2] 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., 1979.

[3] California Air Resources Board. Estimated Annual Average Emissions, 2005.

[4] CARB. EMFAC2007 Version 2.30 User Guide: Calculating Emission Inventories for Vehicles in California. Technical report, California Air Resources Board, 2006.

[5] CARB. Motor Vehicle Emission Factor / Emission Inventory Model, EMFAC, 2007.

[6] S. Lee, G. Wolberg, and S. Y. Shin. Scattered data interpolation with multilevel B-splines. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):229–244, 1997.