Example: San Francisco

David Holstius

December 2, 2011

1 Introduction

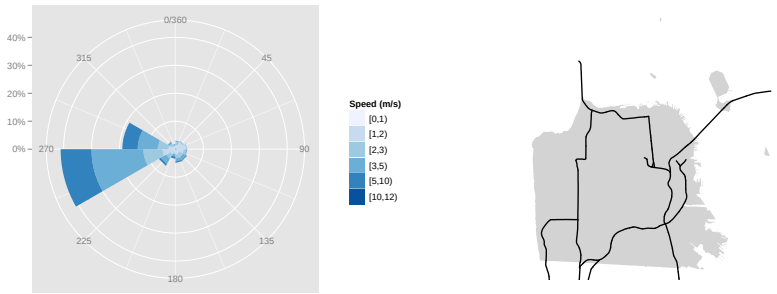
Here we apply `Rcaline` to San Francisco data provided by the Bay Area Air Quality Management District (BAAQMD).

Both the traffic data and the meteorological data have already been imported (from ESRI shapefile and ISC formats, respectively). To learn about importing spatial data and meteorological data for use with Rcaline, please consult the accompanying vignette, "Importing Data for use with Rcaline".

2 Constructing the model

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

```
> require(Rcaline)
> data(SanFrancisco, package='Rcaline')
> ls()
[1] "met_5801.isc"        "SF_county.shp"
[3] "STRte_BAAQMD_v2.shp"
```



(a) Meteorology (b) Highway network

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

2.1 Meteorology

The conventional way to provide hourly meteorology 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.isc, use='urban')
> names(met)
[1] "windSpeed"      "windBearing"    "stabilityClass"
[4] "mixingHeight"
> nrow(met)
[1] 8760
> 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 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 2b shows the San Franciso highway. This network is composed of multiple *polylines*, each of which is composed of one or more segments or *links*.

In this case, each polyline has an attribute, `TRVol2009`, that represents the Annual Average Daily Traffic for those links.

```
> lnk <- FreeFlowLinks(STRte_BAAQMD_v2.shp,
          vehiclesPerHour = TRVol2009 / 24,
          emissionFactor = 1.0,
          width = 30.0)
> plot(SF_county.shp, col="light gray", border=NA)
> lines(lnk, lwd=2)
```

We need to divide the AADT by 24 to convert to vehicles per hour. We also need to specify the road width (iun meters), and an emission factor[1], in grams per vehicle per mile. For the sake of this example, we'll use a "unit emission factor" of 1.0.

In California, fleet-level emission factors for specific pollutants can be obtained from models like EMFAC (TODO: cite). Imputing link-level factors from fleet-level factors can induce biases (CITE), but link-level data are often scarce.

[1]The source strength (total emissions, in grams per mile per hour) modeled by CALINE3 for any given link are exactly equal to the emission factors multiplies by the traffic volumes. This means that, if you use a unit emissions factor, the resulting dispersion surface can be linearly scaled to reflect different emissions scenarios, without needing to run the dispersion step again.

(a) Grid-based (b) Ring-based

Figure 2: Receptor locations sampled using two different methods.

2.3 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 receptor grid, and it need not be regular.

For example, you might want to compute predicted concentrations at a set of geocoded street address locations. As long as you have these in a format that you can import into R (as a SpatialPoints object), you can use these. Or, you can statistically sample locations within defined regions (like ZIP codes), possibly using the `spatstat` package.

Modelers often use a regular Cartesian grid (Figure 2a), but this can induce artificial "hot spots" where the grid happens to fall closer to the road. A better approach (Figure **??**) is to create receptors at specific distances from the road network. This avoids the problem of apparent "hot spots", and in addition allows receptors to be packed more densely close to the roadway, where higher concentrations and higher spatial variability are expected.

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

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

2.4 Other parameters

Additional parameters, including terrain and pollutant characteristics, also need to be specified. For detailed information, consult the CALINE3 User's Guide [1]. Here we use some reasonable default values.

```
> ter <- Terrain(surfaceRoughness=80.0)
> CO <- Pollutant("Carbon monoxide", molecularWeight=28.0, settlingVelocity=0.0)
> mod <- Caline3Model(lnk, met, rcp, ter, CO)
```

3 Modeling

3.1 Running the model

We use the `predict` method to run the model. It can be quickest to do a first pass using only 1% of the meteorology, sampled at random:

```
> mod <- Caline3Model(lnk, sample.rows(met, p=0.001), rcp, ter, CO)
> pred <- predict(mod)
```

If you're using R on a console (not, say, the Mac R.app GUI), `Rcaline` will use the `multicore` package to do the computations in parallel, using however many cores you have (minus one). The speedup should be nearly linear in the number of cores.

4 Analyzing results

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

4.1 Aggregation

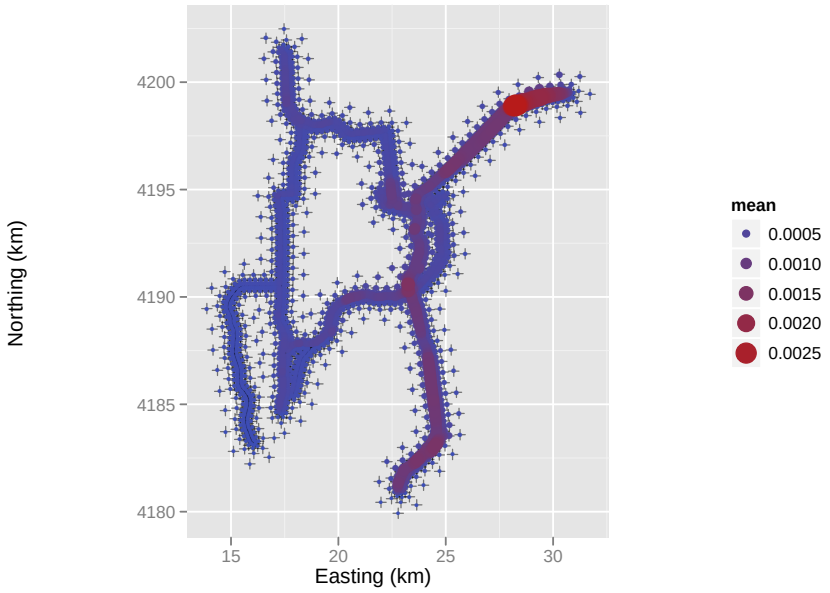
It is generally advisable to treat the result as a sample from an annual distribution (even if all hours from a single year are used). The `aggregate` function can be used to estimate the properties of that distribution, by computing summary statistics such as the mean or maximum.

```
> result <- aggregate(pred)
> head(result, 5)
     min     mean   median GM      max       sd
[1,]   0 0.000112 5.30e-05  0 0.000308 0.000129
[2,]   0 0.000109 4.75e-05  0 0.000306 0.000128
[3,]   0 0.000110 5.07e-05  0 0.000310 0.000129
[4,]   0 0.000118 8.84e-05  0 0.000300 0.000124
[5,]   0 0.000128 1.31e-04  0 0.000296 0.000126
```

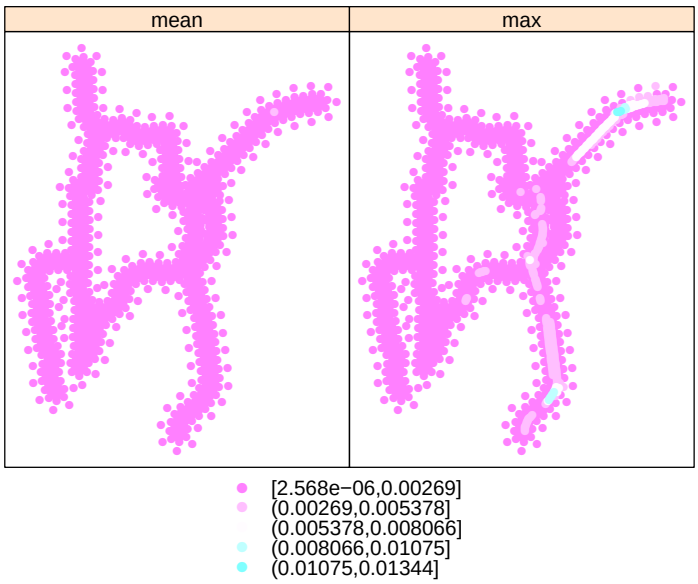
4.2 Plotting

Summary statistics can be visualized one at a time, or several at a time, using the `ggplot2` or `sp` packages:

```
> show(ggplot(result))
```




```
> show(spplot(result, zcol=c("mean", "max")))
```



4.3 Interpolation

As an alternative to the bubble plot, it's possible to construct a raster image by interpolating the summary statistics back to a regular grid. For example:

```
> grd <- ReceptorGrid(lnk, resolution=250, maxDistance=1e3)
```

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.