Lecture 12:

R is like a fresh hair cut because...

Last time

We considered common manipulations with matrices and vectors; we reviewed the simple rules for extracting and assigning values to subsets elements of these structures.

The logical data type (TRUE and FALSE) emerged in this discussion; we also discussed another common statistical object, a factor.

Finally, we started to look at objects that were a little less rigid than matrices and vectors; we looked into lists and found that they are the basis for many “classes” of objects in R.

Today

We start with your homework; we will temporarily de-emphasize the classification problem and consider what we’re ambling toward when we fit polynomial “pieces” as we’ve been doing.

This will give us an opportunity to talk about some of the functions in R that make it convenient for common operations like fitting a linear model.

We then look at an endpoint in the evolution of the data table in S/R, the data frame; we will see how this structure combines the regular structure of a matrix with the “grab bag” properties of a list.

Thermal mapper experiment

Recall that the data from this experiment are spread between (now) five objects (four vectors and a matrix).

- groundcover: A character vector of length 307 indicating either a kind of plant, a patch of sand (small rocks) or a boulder (um, a big rock)
- positions: A numeric vector of length 307, describing the distance of the spot in cm from a fixed reference
- scantimes: A vector of length 148 representing the times each scan ended
- temp: A 307x148 matrix representing the temperatures collected by the robotic node
- quality: A logical vector of length 307 indicating which classifications were “clean”
Functional properties of the data

One of the nice characteristics of our data is that they arrange themselves nicely in a rectangle; rows index positions along the transect and columns indicate times.

As such, we have been thinking of the temperature values at a single position as a “response” to ambient conditions (temperature, wind, sun exposure) and have looked to characteristics of this response to differentiate the ground cover.

This led us to considering differences in curvature or slopes, locations of (and the value of) the minimum temperature and so on; these might be thought of as “functional” properties of our response.

Your homework

While I know that we don’t have the formal statistical machinery to treat a classification problem in earnest, investigating which properties “separate” the different groups in an ad hoc fashion really requires only subsetting, a linear model or two, and simple plotting.

Our goal is not to come up with an idea “classifier,” but instead to investigate some of the structural properties of the temperature responses and how they differ for the varieties of ground cover.
What do we see?

The temperature response for boulder has less of a drop in this region; it cools off more slowly than the other types of ground cover.

Poa is a grassy plant and it is much more influenced by the ambient temperature; hence the slopes associated with Poa are among the largest (steepest decline).

This kind of approach reduces each response (148 temperature values) to a single number and then forms groups by ground cover class; boxplots let us evaluate how distinct the populations are.
Two features at once

It is probably too much to expect that a single “derived feature,” like the slope associated with the drop in ambient temperature after midday, might allow us to tell the groups apart.

So for homework, you were to examine pairs and see what might produce patterns in scatterplots of the derived features that help us understand the differences between the temperature response of the different ground cover types.

But let’s not get carried away...

There are only 40 cases of Eriogonum in our data set (after we focused on the “quality” measurements) and 84 examples of sand.

We don’t want to get too carried away with our interpretation; this goal is to kick the tires on R.
R and statistical computing

Hopefully, what we’ve seen is an iterative process; some computation, some graphics, more computation, more graphics and so on.

These are cycles of exploratory data analysis as articulated by John Tukey; with luck, R is sufficiently expressive to let us cycle back and forth.

R is also extensible, in the sense that new statistical procedures find their way into the language and, with luck, behave in ways that we come to expect.

Linear models in R

We’ve already seen a simple recipe for working with linear models in R.

Many models (including linear, generalized linear and even nonlinear) begin with a formula to specify the relationship between inputs and outputs.

\[ y \sim x \]  # simple linear model in x, implicit intercept
\[ y \sim \text{poly}(x,2) \]  # polynomial of degree 2, orthogonal basis
\[ \log(y) \sim x \]  # model a transformation of the data

After you’ve fit a model, there are a series of functions that help you manipulate the fit; we’ve already seen examples of “extractor” functions:

- `plot`
- `coefficients`
- `predict`
- `residuals`
- `print`
- `summary`
- `formula`
- `add1`
- `drop1`

These are known as “generic” functions; depending on the kind of model fit you provide them with, they do different things.

We will talk about how R structures model-specific or, more generally, object-specific computations toward the end of the lecture.
But first...

The generic functions described are certainly convenient; they mean that you can work with models with a common set of language constructions.

This is, in part, what is meant by the expressiveness of the language: reasoning with statistical models, performing common computations and building graphical displays, and doing all that quickly, is what R does well.

R also captures less classical and more modern statistical procedures; as an example, let's consider our response data again for the sample of sand we started with.

Fitting polynomials, locally

Consider the sand sample; as a function of time, how do we describe the trend present in the data?

Is it (line + noise)? or (polynomial + noise)? What degree polynomial would we need here?
Alternatives

High degree polynomials are (somewhat) flexible and we have a little numerical analysis to back us up; we can approximate functions well with polynomials.

But they can be unstable; to fit the data, we end up constraining them a lot.

Thinking of approximation theory, we might appeal to a Taylor expansion; that is, fit low-degree polynomials locally.

This is essentially what we’ve been doing in our hunt for good features to separate the classes...

Local polynomial fitting

R captures this idea of fitting quadratics (or any degree polynomial) locally in a function called `loess`.

Local polynomials are a class of smoothers that behave more reliably than high-degree polynomials; whereas we increase the flexibility of polynomials by increasing their degree, we increase the flexibility of a local polynomial by decreasing its span.
The point?

R is rapidly becoming the repository for current statistical methodology; from smoothing (like loess) to imputation and beyond

Now, let's go back to the basics...
One last data structure: data frames

We have seen vectors and matrices and arrays; they consist of one basic data type (either numeric or logical or character or...)

We have also seen lists; these tend to be grab bags that hold just about anything, including other lists

In many statistical applications, we need a hybrid structure; a data table in which each column can be of a different data type, but not quite as loose as a list in that we want the notion of rows in the table to represent “cases”

Next time

We will cover data frames in a bit more detail next time; for our current data set, they are not so compelling since we can hold the data easily in a matrix

One last topic for today, again a teaser of sorts...

Generic functions

A few slides ago, we saw some examples of so-called generic functions; in these cases, the class of the object we are working with determines what kind of computation takes place

For example, we have seen plot() used to generate a scatterplot or a boxplot depending on whether the first argument is numeric or a factor

We have also seen predict() and summary() produce sensible but different results for different kinds of statistical fits (local regressions or ordinary linear models)
Methods

Functions that are designed to work on a specific class of objects are referred to as methods for that class.

We can find the methods associated with any class by asking R; we can specify either a class or a generic function.

Recognizing a generic

If you type out the name of any function in R, just like any object, it prints or exhibits the object in some way.

We recognize a generic because of the tell-tale `UseMethod`; this acts like a kind of dispatch service finding which method to use depending on the class of the object involved in the function call.

For the record, any time you look at or “print” an object, you are invoking a generic...