Introduction to Gaussian processes

Matthew Stephens
University of Chicago
February 11, 2026

See here for a PDF version of this vignette.

Introduction

You need to be familiar with the multivariate normal distribution.

```
library(mvtnorm)
library(geoR)
```

Motivation

Suppose you want to model the variation of a continuous variable T — say, temperature — across a spatial region (think in two dimensions for now). Let $T(x)$ denote the temperature at any location x (so think of x as a position in a space $\mathcal{X}$). So T is a function that maps the space $\mathcal{X}$ to the real value line $\mathbb{R}$, which we can write $T : \mathcal{X} \to \mathbb{R}$.

Suppose you measured the temperature at just one location, say $x = x_1$. The value $T(x_1)$ would be a scalar, so you could imagine modeling it using a univariate normal distribution.

Similarly, if you measured the temperature at two locations, say x_1 and x_2, then you would obtain two temperatures $T(x_1), T(x_2)$. If the locations x_1 and x_2 were near to one another, then we might expect the two temperatures $T(x_1)$ and $T(x_2)$ to be similar to one another. If the locations were further away from one another then maybe the temperatures would be less similar. This suggests one might model the pair $T(x_1), T(x_2)$ using a bivariate normal distribution whose covariance depends on the distance between x_1 and x_2 (decaying with distance).

Now, suppose we measure the temperature at r locations $x_1, \ldots, x_r$. The natural generalization of the above is to model the temperatures $T(x_1), \ldots, T(x_r)$ as a multivariate (r-variate) normal distribution with the covariance of $T(x_i), T(x_j)$ depending on the distance between x_i and x_j (decaying with distance). Notice that, conceptually, the function T is defined everywhere in space, but we are measuring it at a finite number of points — and we assume that at any finite number of points the values we get will follow a multivariate normal distribution. This idea motivates the definition of a Gaussian process — it is defined in a continous space, but at any finite number of points it has a multivariate normal distribution.

Definition

A stochastic process $T(x) : x \in \mathcal{X}$ is said to be a *Gaussian Process* if $(T(x_1), \ldots, T(x_r))$ is multivariate normal for all $x_1, \ldots, x_r \in \mathcal{X}$.

Defining a Gaussian process

Just as a multivariate normal distribution is defined by a mean and a covariance matrix, a Gaussian process is defined by a *mean function* $\mu : \mathcal{X} \to \mathbb{R}$ and a *covariance function* $\Sigma : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Think of $\mu(x)$ as specifying the mean of $T(x)$ for any point x, and $\Sigma(x_1, x_2)$ as specifying the covariance for any pair of points x_1, x_2. That is,

$$E[T(x)] = \mu(x),$$

and

$$\text{Cov}(T(x_1), T(x_2)) = \Sigma(x_1, x_2).$$

Given suitable functions μ and Σ, we could simulate the values $T(x_1), \ldots, T(x_r)$ from a multivariate normal distribution by first computing the mean (by applying the function $\mu(x)$ to every point $x_1, \ldots, x_r$) and the covariance matrix (by appyling the function $\Sigma(x, y)$ to every pair of points), and then using a standard method for simulating from a multivariate normal. We will see an example of this below.

Note that the covariance function Σ has to be carefully chosen, so that the covariance matrix it gives for any set of points is a valid covariance matrix. Specifically, the covariance matrix must always be positive semi-definite (PSD), meaning all its eigenvalues are greater than or equal to zero. So you can't just choose any function you like for Σ. We will give some common choices below after we have introduced a common simplification.

Stationary and isotropic Gaussian processes

Gaussian processes become simpler to define and work with if we make two additional simplifying assumptions:

- The mean function μ is a constant, $\mu(x) = \mu$ for all x.
- The covariance function $\Sigma(x_1, x_2)$ depends only on the distance between the two points, $d(x_1, x_2)$. That is, $\Sigma(x_1, x_2) = K(d(x_1, x_2))$ for some function K that maps distance to covariance.

A Gaussian process that satisfies these two assumptions is said to be *stationary and isotropic.*

This simplifies defining a Gaussian process because we just have to specify the mean μ (a number) and the function $K(d)$, which says how covariance declines with distance d. Again, because we need the covariance matrix to be PSD, only certain choices of K are valid. Three simple common choices of K that lead to valid PSD covariances are:

- *Squared exponential:* $K(d; l) = \exp(-\frac{|d|^2}{2\ell^2})$ where ℓ is a parameter that determines the length scale over which the covariance decays (e.g., does covariance of temperature decay over meters, or kilometers, or hundreds of kilometers?).
- *Ornstein-Uhlenbeck:* $K(d; l) = \exp(-\frac{|d|}{\ell})$, where again ℓ is a parameter that controls the length scale.
- *Matérn:* whose form is a bit more complicated, but is easily computed using the geoR package. The Matérn covariance function is widely used in geostatistical applications.

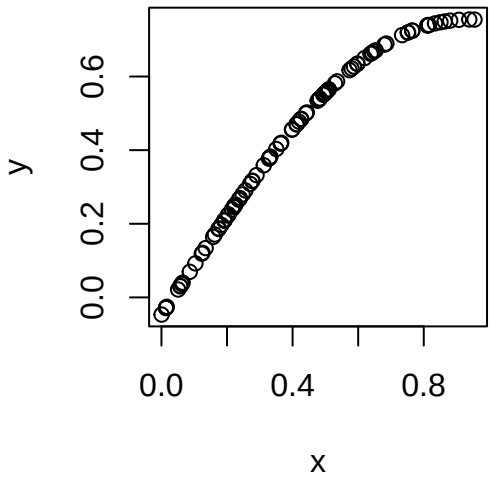
Simulation

To help understand a new distribution, it can often be helpful to simulate from it. Here we look at some simulations from a few different Gaussian processes. In these examples, we look at one-dimensional examples just because they are much easier to plot and visualize. One-dimensional GPs are also useful, say, to model the way that variables vary along a 1-d space (e.g., time, or along the human genome).

Squared exponential kernel

So here we generate 100 points $x_1, \ldots, x_{100}$ in the range $[0, 1]$ and simulate the value of a Gaussian process at those 100 points using the squared exponential kernel:

```
set.seed(11)
x <- runif(100)
l <- 1
d <- abs(outer(x,x,"-"))
Sigma_SE <- exp(-d^2/(2*l^2))
y <- rmvnorm(1,sigma = Sigma_SE)
plot(x,y)
```

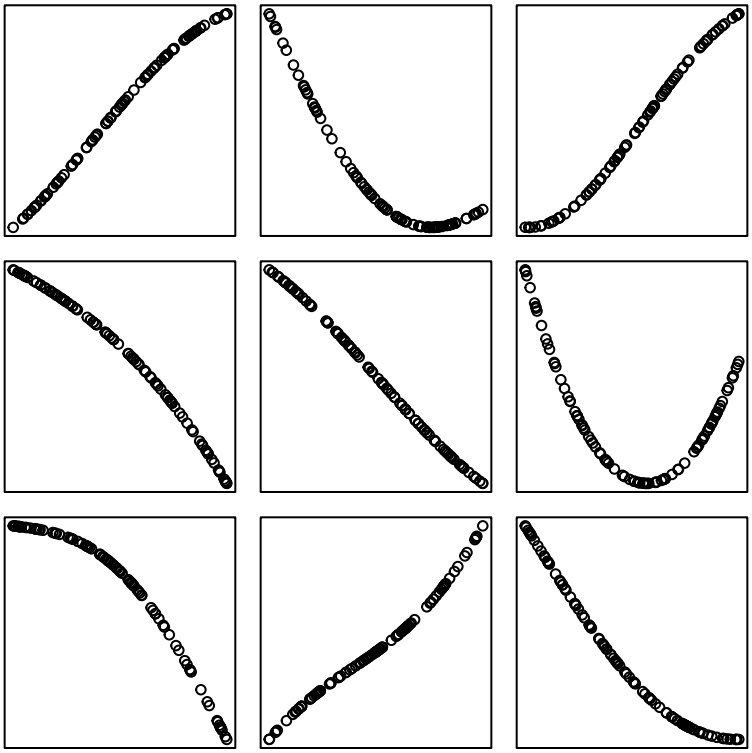


You can see that effectively this simulation has created a "curve" in which y varies smoothly with x. Indeed, a GP can be thought of as a distribution on curves (whose smoothness depends on the covariance function used, as we will see later).

Of course, this is just one random curve (and only measured at 100 points, although conceptually it is defined at all points in the space). Here we generate nine different simulations:

```
par(mfcol = c(3,3),mar = c(0.5,0.5,0.5,0.5))
for (i in 1:9) {
  x <- runif(100)
  d <- abs(outer(x,x,"-"))
  Sigma_SE <- exp(-d^2/(2*l^2))
```

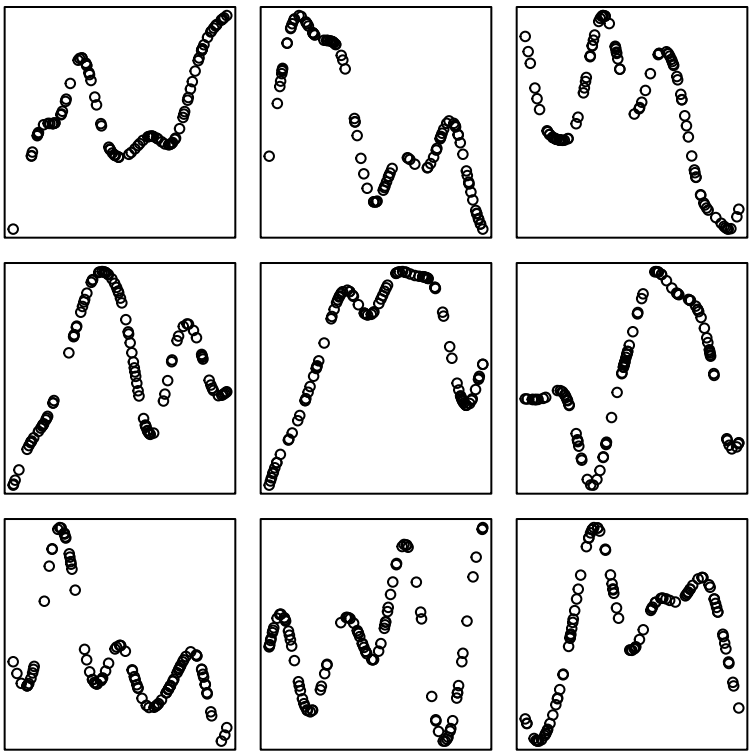
```
  y <- rmvnorm(1,sigma = Sigma_SE)
  plot(x,y,axes = FALSE,frame.plot = TRUE)
}
```



Change length scale

Notice that these curves do not change very quickly with distance. We can get curves that change more quickly by using a smaller length scale parameter ℓ in the covariance function:

```
par(mfcol = c(3,3),mar = c(0.5,0.5,0.5,0.5))
l <- 0.1
for (i in 1:9) {
  x <- runif(100)
  d <- abs(outer(x,x,"-"))
  Sigma_SE <- exp(-d^2/(2*l^2))
  y = rmvnorm(1,sigma = Sigma_SE)
  plot(x,y,axes = FALSE,frame.plot = TRUE)
}
```

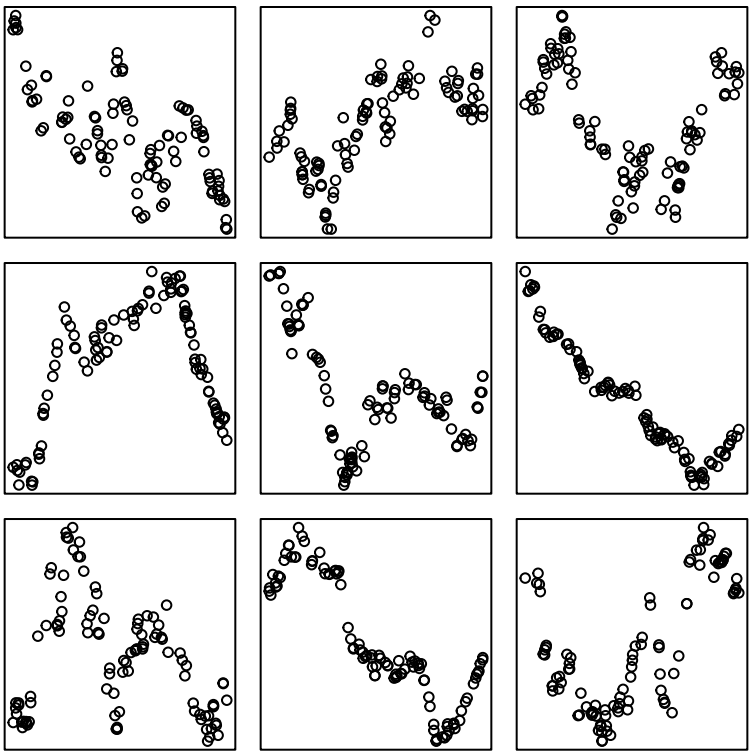


The Ornstein-Uhlenbeck covariance function

Here we use the covariance function for what is known as the *Ornstein-Uhlenbeck process*, which you can think of as a modified Brownian motion, where the modification tends to pull the process back towards zero. (Unmodified Brownian motion tends to wander progressively further from zero.)

Notice that visually these functions seem much "rougher" than those produced by the squared exponential function. And indeed this visual impression is correct: the functions produced by the OU covariance are not differentiable, although they are continuous.

```
par(mfcol = c(3,3),mar = c(0.5,0.5,0.5,0.5))
l <- 1
for (i in 1:9) {
  x <- runif(100)
  d <- abs(outer(x,x,"-"))
  Sigma_OU <- exp(-d/l)
  y <- rmvnorm(1,sigma = Sigma_OU)
  plot(x,y,axes = FALSE,frame.plot = TRUE)
}
```



The Matérn covariance function

The Matérn covariance function produces curves that are a smoother than OU, but not as smooth as the squared exponential:

```
par(mfcol = c(3,3),mar = c(0.5,0.5,0.5,0.5))
l <- 0.1
for (i in 1:9) {
  x <- runif(100)
  d <- abs(outer(x,x,"-"))
  Sigma_M <- matern(d,phi=l,kappa=1)
  y <- rmvnorm(1,sigma = Sigma_M)
  plot(x,y,axes = FALSE,frame.plot = TRUE)
}
```