Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

TUTORIAL: BASICS OF METAMODELING

Russell R. Barton

Department of Supply Chain and Information Systems
Smeal College of Business
The Pennsylvania State University
University Park, PA 16802, USA

ABSTRACT

Metamodels are fast-to-compute mathematical models that are designed to mimic the input-output behavior of discrete-event or other complex simulation models. Linear regression metamodels have the longest history, but other model forms include Gaussian process regression and neural networks. This introductory tutorial highlights basic issues in choosing a metamodel type and specific form, and making simulation runs to fit the metamodel. The tutorial ends with a warning on potential pitfalls, and suggestions on further reading to expand your knowledge of metamodeling.

1 INTRODUCTION

Discrete-event simulation (DES) allows one to explore the dynamic behavior of complex stochastic systems. In many cases exercise of the real system is not possible, either due to practical constraints or because the system does not yet exist. Although there has been work on modeling dynamic aspects of system behavior (Fishman and Kiviat 1967; Schruben and Cogliano 1987; Sargent and Som 1992), this introductory tutorial focuses on DES for the case where a single numerical characteristic is of interest. This could be the mean of some overall performance measure, perhaps statistical uncertainty about the mean; or another characterization such as a quantile. When the DES model is complex, the computational effort in system simulation can be substantial. Metamodels are useful in this context: a computationally efficient approximation to the simulation model input-output function. The term *metamodel* was popularized and developed by Jack Kleijnen (for example, Kleijnen 1975), but the term and concept were both originated by Robert Blanning (1974, 1975). Metamodels are also called surrogate and response surface models.

Metamodels, once fitted, can be used as a proxy, to evaluate instead of making (computationally expensive and stochastic) DES runs. Further, because of their explicit form, they can be used in many computationally intensive simulation operations, such as optimization (Barton and Meckesheimer 2006), input model uncertainty, quantiles and conditional value at risk, and robust design (Dellino et al. 2009).

Simulation metamodeling remains an active area of research. While advances in metamodeling continue to appear, the introductory body of knowledge is not greatly changed over the past few years, and so this tutorial borrows heavily from its predecessors (Barton 2015; Barton 2020). The emphasis is on ordinary regression with or without transformed variables, and Gaussian process (GP) metamodels, both of which are straightforward to implement. Difficulties with neural networks for beginners are explained. Aspects relevant in an introductory setting are presented in Section 3.

This tutorial aims to provide an accessible introduction to the beginner. This does not obviate a need for a familiarity with mathematical models and basic statistics. The intended learning outcome is for participants to understand the key tasks and tools needed to build, validate and use simple metamodels using tools like Excel, Minitab, SAS/JMP or R. Metamodeling process and metamodel types have greater emphasis than the experiment designs used to fit them. For more detail on the design of experiments (DOE) for choosing DES runs to fit a metamodel. see Barton (2021), Kleijnen (2008), Law (2017), and Sanchez et al. (2021), each of which provide different, complementary views. The tutorial focuses on common uses for metamodels, not on sophisticated applications such as quantile estimation (Batur et al. 2018) or input uncertainty (Barton et al. 2014).

The next section places the metamodeling activity in the overall process of building and using discrete-event simulation models. That is followed by a section reviewing and illustrating the basic metamodeling process for predicting average waiting time in a queue as a function of mean service time. The next section describes selecting and scaling the predictor variables (x) for the model and the selection and possible transformations of output variables (Y) to be predicted, again illustrated for the queueing example. An introduction to metamodel types and associated experiment designs follows. The next sections describe metamodeling software and the basics of metamodel validation.

2 METAMODELING PRINCIPLES

2.1 The Role of Metamodels in DES

There is general agreement about the major activities that are part of a simulation study; for examples see Banks et al. (2009) and Law (2014). Below is a summary in ten key steps:

1. Formulate the problem and objectives of the study.
2. Collect data, formalize assumptions and conceptual model.
3. Test validity of the conceptual model against objectives.
4. Identify design parameters and outputs; program the model.
5. Verify that the program output matches the conceptual model expectations.
6. Validate program: comparing against real system data or expected behavior.
7. Create a DOE for decision support.
8. Run simulation program using the decision support DOE.
9. Analyze results of the experiment.
10. Use *simulation* results to inform decisions: through prediction, sensitivity analysis, optimization.

The role of metamodels fits in the last four steps of this process. The key assumption is that there is interest in predicting system performance under a large number of conditions; too many to explore directly using the DES model, or because the response for a particular condition must be determined quickly, more quickly than by conducting the simulation. Two characteristics make metamodeling distinct from generic ‘machine learning’: the stochastic output typically has heterogeneous variance, and data is from a designed experiment, rather than observational data. While machine learning is often used to predict a class, metamodels predict some numerical performance, not a class. In the next section we expand the description of the metamodel process in the above context.

2.2 Basics of Metamodeling

A metamodel is a function, say f, that takes some simulation model design parameters as inputs, represented here by a vector x, to an output, $f(x)$. Examples of model design parameters include input probability distribution parameters, such as arrival rate and mean service time; and system configuration parameters, such as the number of servers, service priority, operational protocols, and buffer capacity. For now, assume that any design parameter can be coded numerically, even if only as a 0-1 variable. There will be more about how to do this later in the tutorial. The metamodel $f(x)$ produces an approximation to some characteristic of a simulation output Y, e.g., mean of Y, standard deviation of Y, 0.9-quantile of Y. Because our focus is on dynamic simulation, these quantities change over the duration of the simulated time period, which is called a *run*. Examples of simulation outputs are time in the system for a set of jobs or customers, utilization of a particular resource (e.g., operator, machine), or perhaps net revenue over a specific time period. Generally, these outputs are averaged over the length of the simulation run and vary randomly from run to run. If the value of the characteristic is $Y(x)$ for an actual simulation run with design parameters set to the values in x, then we represent the fitted metamodel approximation $f(x)$, as:

$$h(Y(x)) \approx f(x), \tag{1}$$

where h represents some function of the random variable Y such as its mean, standard deviation, or a quantile. Note that the distribution of Y depends on the values of the design parameters. The simplest and most common metamodel type is linear regression. The term "linear" refers to the way the unknown coefficients come into the model. Linear regression can capture curvilinear relationships.

To illustrate the basics of metamodeling, we will fit two linear regression metamodels to data from a simple queueing simulation. Although simulation is not needed for an M/M/1 queue, such a system is easy to understand and has behavior that is frequently seen in simulations. In this case we want to explore how the average waiting time (not including service; the *output*) varies with mean time to service a customer (the *input*, or *design parameter*), assuming an arrival rate of 1 customer per unit time (in some scaled units of time). Figure 1, made using the basic R linear model function `lm()`, shows the results of simulations of 5000 customers in systems with mean service times of 0.7, 0.75. 0.8, 0.85, 0.9 and 0.95. There are three replications for each of the run conditions.

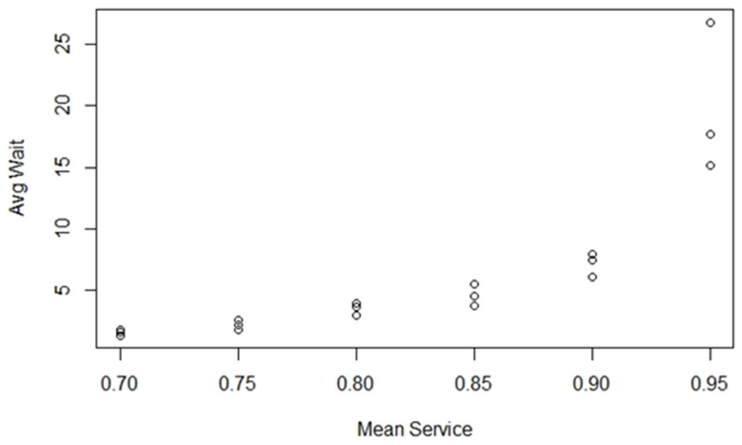

Figure 1: Output of M/M/1 experiments, three replications each at six mean service time settings.

If we fit a linear regression model to this data, the metamodel produces the fit seen in Figure 2. If we fit a quadratic metamodel to these data, it produces the fit shown in Figure 3. Clearly, there are problems with the fidelity of both of these fitted metamodels. We will return to this example throughout the tutorial to discuss each aspect of the metamodeling process.

For Figure 2, the metamodel corresponding to (1) has $Y \equiv$ average waiting time for the 5000 customers in a simulation run, $h(Y) \equiv$ the expected value of this run average, i.e., E(Y); $x \equiv$ mean service time (the x vector has only one component), and $f(x) = -78.10 + 61.42x$. For Figure 3, the fitted metamodel is $f(x) = 269.9 - 708.3x + 466.5x^2$.

Now that we have a basic understanding of a metamodel, we can discuss the process of selecting a metamodel form and choosing simulation experiment designs to fit a chosen metamodel form.

2.3 The Metamodeling Process

The rest of this introductory tutorial is organized around the steps of the metamodeling process. An early formal description was given by Burdick and Naylor (1966). Table 1 shows a similar process.

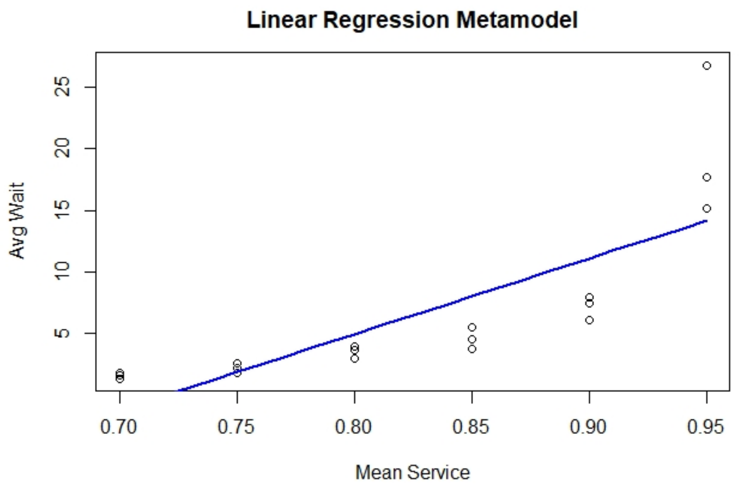

Figure 2: A regression metamodel fitted to the data, with intercept and linear term only.

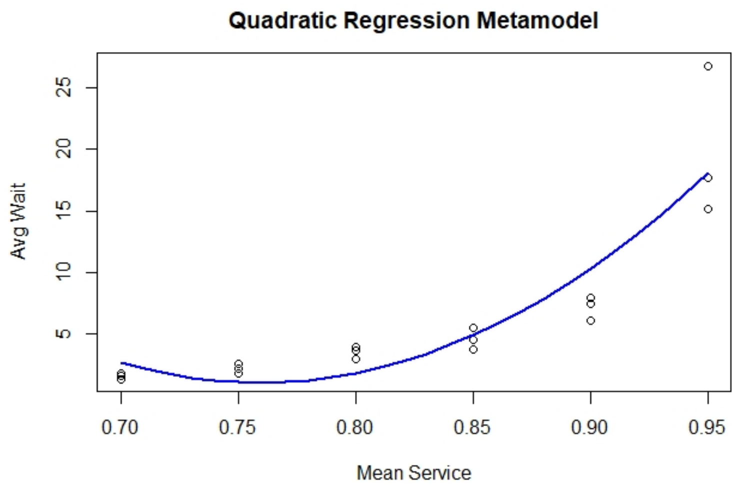

Figure 3: The fitted quadratic regression metamodel is an improvement.

Table 1: The metamodeling process.

Step	**Activity**
1	Determine Purpose(s) for Metamodeling
2	Identify Design Parameter(s) and Output(s)
3	Choose Metamodel Type
4	Based on Metamodel Type and on Purpose, Choose Experiment Design to Fit Metamodel
5	Conduct Simulation Runs Specified by the Experiment Design; Fit Metamodel
6	Validate Metamodel Adequacy: If Unsatisfactory Return (usually) to Step 3
7	Use Metamodel for Intended Purposes

3 IDENTIFYING DESIGN PARAMETERS AND OUTPUTS

3.1 Design Parameter and Output Selection

Typically, simulation models have many design parameters that might be included in a metamodel. Each parametric probability distribution used in the model has parameters. These are called input distribution parameters. They characterize randomly varying interarrival times, service times, times until machine breakdown, message lengths, routing choices, transport times, number of workers out sick, uncontrollable environmental factors, and other characteristics of the simulated system that have a stochastic nature.

In addition, there are design parameters that are not associated with probability models: the number of workers scheduled to work at a particular time, the number of available machines, processing protocols and other characteristics of the system being modeled that can be changed, either for a currently operating system or some system to be built in the future.

The particular parameters to include in the metamodel depend on three things: i) a desire to include parameters that the decision maker would like to explore changing; ii) the need to model the impact of any uncontrollable environmental factors that affect system performance; and iii) the recognition that more metamodel parameters generally means a larger fitting experiment, and more computational effort to fit the metamodel. Further, for regression metamodels, the inclusion of higher order terms to capture interaction effects and nonlinearity (x^2 for the M/M/1 example) also adds to the size of the experiment design. Cause-effect diagrams and a-priori plots can help identify the design parameters and any expected interactions or nonlinearity (Barton 2021). For the M/M/1 example above, the experiment design was presented as given. In actuality, it would be selected after considering whether to include x^2 and perhaps x^3 in the metamodel.

Usually, a separate metamodel is fitted for each function of the simulation outputs needed for the purpose(s) identified in Step 1. Multiple response surface models were considered in the early paper by Burdick and Naylor (1966) and many subsequent authors. True multivariate metamodels are a recent development (Liu et al. 2018). For the M/M/1 example above, only one function of one simulation output was considered: h corresponds to the expected value, and Y is the average waiting time of 5000 simulated customers. Occasionally, h will be chosen to reduce the nonlinearity of the output with respect to the design parameters, or to achieve equal (homogeneous) output variance across the design space. Methods for choosing such transformations are summarized in Barton and Meckesheimer (2006).

3.2 Continuous and Discrete Design Parameters and Outputs

Generally metamodeling assumes that all design parameters and outputs can take on continuously varying values. But many design parameters are discrete. Examples include numbers of servers, machines or other resources, buffer sizes, number of products, type of processing protocol, system configuration alternatives, and so forth. When the parameter has a numerical value that is restricted to a discrete set of values, then assuming that the parameter can take on a continuous set of values is often practical. The discrete nature places a restriction on the experiment design that is used for fitting, but the fitted metamodel can be evaluated on the discrete set of allowed values. When there are only two values for the parameter (e.g., *Protocol A* and *Protocol B*), a discrete numerical characterization can be assigned, for example, $x = 0$ for *Protocol A* and $x =1$ for *Protocol B*. If the parameter does not have numerical value and there are more than two levels, numerical conversion is still possible, but requires additional x components. For example, for three protocols, let $x_1 = 1$ if *Protocol A* is used, and = 0 otherwise. Let $x_2 = 1$ if *Protocol B* is used, and = 0 otherwise. If *Protocol C* is used, then both x_1 and $x_2 = 0$. Then metamodel terms for x_1 (or x_2) will indicate the differences of *Protocol A* (or *Protocol B*) from *Protocol C*.

When the simulation output is discrete (e.g., success or failure), there is a restriction on the type of metamodel. These are generally called classification or discriminant metamodels. It is also possible to construct metamodels when the response is piecewise continuous (Meckesheimer et al. 2001).

3.3 Scaling/Coding Parameters and Outputs

In addition to the output transformations and qualitative variable coding mentioned previously, the ability of the metamodel to provide insight depends on careful scaling and coding of all design parameters. Generally, scale all numerical design parameters so that -1 is the smallest value taken, and +1 is the largest value. This scaling is accomplished by the following:

$$x_{new} = 2[x- ((x_{max} + x_{min})/2)/(x_{max} - x_{min})]. \qquad (2)$$

Compare the insight from the coefficients for the fitted metamodel with unscaled mean service time:

```
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         269.92      76.85   3.512 0.003144 **
Mean_Service       -708.30     187.75  -3.773 0.001844 **
I(Mean_Service^2)   466.50     113.68   4.104 0.000939 ***
```

with the results below, with mean service time scaled to +/-1.

```
(Intercept)           3.076      1.091   2.820 0.012927 *
sMean_Service         7.677      1.038   7.398 2.23e-06 ***
I(sMean_Service^2)    7.289      1.776   4.104 0.000939 ***
```

First, the coefficients for the unscaled regression are hard to interpret. What does a linear effect of -708 mean? Remember, the units are in minutes. For the scaled coefficient model, the value 7.677 means the linear increase in average wait time when moving from the mid-level mean service time value (.7+.95/2 = .825) to the high value (.95) is about 7.7 minutes; added to that is the quadratic portion of increase: another 7.3 minutes. Clearly, basic insights from the regression model coefficients fail to materialize without careful coding of the design parameter value(s). Second, the linear coefficient changed drastically from the model with only a linear term described at the end of Section 2.2: 61.4 vs. -708.30. The regression model with only a linear term but using scaled *x* would produce the same linear coefficient as the scaled quadratic model: 7.677. Why are they so different for the unscaled model? This difference is caused by the same issue as this third problem: the p-values are larger for the unscaled model - because the linear and quadratic effects are confounded. These effects are not confounded in the scaled model. The quadratic effect *is* confounded with the intercept in the scaled model. Fixing that requires an orthogonal polynomial model structure. The topic of orthogonal polynomials is beyond this tutorial. See for example Montgomery (2012).

4 CHOOSE A METAMODEL TYPE

For this introductory tutorial we focus on two types of metamodels: linear regression and stochastic Kriging (spatial correlation) models, and briefly touch on a third: neural network regression. There are descriptions of other metamodel types in Barton (2009). We identify the metamodel type before selecting an experiment design for practical reasons: factorial and fractional-factorial designs, appropriate for linear regression, can cause significant numerical difficulties when used to fit Kriging models. Further, the complexity of the linear regression model places (minimum) requirements on the type of factorial or fractional-factorial design that can be employed for fitting.

4.1 Linear Regression Metamodels

The form of the linear regression probability model that characterizes the simulation output is:

$$Y(x) = \beta_0 + \beta_1 g_1(x_1, x_2, \cdots, x_d) + \cdots + \beta_p g_p(x_1, x_2, \cdots, x_d) + \varepsilon \qquad (3)$$

where ε are independent, normal random quantities with mean zero and unknown variance and there are d design parameters. Again, the model is *linear* because the unknown coefficients (β's) appear linearly (as multipliers) in the model. The g functions can be nonlinear in the x's, for example, $g_5(x_1, x_2, ..., x_d) = x_1^2$, or $g_7(x_1, x_2, ..., x_d) = x_1x_5$. There are p terms in the model (not counting the intercept). The assumption is that the variance of $Y(x)$ does not change depending on the values of $(x_1, x_2, ..., x_d)$. This model (3) implies that $E(Y)$ has the same form as (3) but without the ε term. Further, the regression metamodel $f(x)$ will match (3) but with estimated values $b_0, b_1, \dots, b_p$ for the unknown β coefficients. Since the estimated values $b_0, b_1, \dots, b_p$ will vary randomly from one experiment to the next, conceptually the fitted metamodel depends randomly on the data. Given a set of data $\{x_i, y_i\}$ where $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ is the vector of design parameter values for the i^{th} simulation run, and y_i is the corresponding output, let X be the matrix whose i^{th} row is x_i and let y be the column vector consisting of the elements $\{y_i\}$. Then the estimated coefficient vector $b = (b_0, b_1, \dots b_p)$ is computed by:

$$b = (X'X)^{-1}X'y, \tag{4}$$

where the prime symbol denotes matrix transpose. The solution (4) minimizes the average squared deviation of the metamodel prediction from the observed simulation outputs. The intercept term b_0 has a 'dummy' g function $g_0(x_1, x_2, ..., x_d) \equiv 1$. The fitted quadratic regression metamodel using scaled x values at the end of Section 3 has estimated coefficients $b_0 = 3.076$, $b_1 = 7.677$, and $b_2 = 7.289$.

Linear regression models have simple form and so provide direct insight on the behavior of the simulation. When design parameters are coded over [-1, +1], then the magnitude of the linear coefficients indicate the relative sensitivities of the simulation output to all design parameters (over the defined ranges of parameter values). Similarly, quadratic coefficients can indicate nonlinearity, and convexity/concavity. Coefficients for cross-product terms indicate interaction effects – the sensitivity of the output to changes in one design parameter may vary, depending on the setting of another design parameter. Also, linear regression models are readily available in commercial and free statistical software, and direct calculation via (4) is straightforward.

Linear regression models using polynomial functions of the design parameters have limited flexibility, however. Figure 4 shows an attempt to find a better-fitting metamodel for the M/M/1 example by adding a cubic term. While the curve comes closer to the observed waiting times overall, it is no longer monotonically increasing, something we expect in a metamodel of mean waiting time versus mean service time. Adding more terms improves the fit near the six experiment design points but increases the excursions of the metamodel away from the design points. For the classic illustration of this "excursions" shortcoming of polynomial models see Figure 1 in Barton (1992).

Looking at Figures 1-4 it is apparent that the variance of the response is larger at higher average waiting times. This is common for models where the output is some function of a queueing system, as is often the case in discrete-event simulation. One approach to reducing this *heteroscedasticity* is to take different numbers of replications at different design points. By taking more replications at high-variance points, the variance of the average response at such points is reduced. This is an expensive proposition though: the spread you see is related to the standard deviation, which is only reduced as the square root, so to reduce the spread by a factor of two requires 4 times the number of replications. For our M/M/1 data we would need approximately 120 additional replications for the x = .95 setting!

Often problems with heteroscedasticity *and* fitting can be reduced by transforming the dependent variable(s). A typical transformation for queueing output data is the logarithmic transformation. The left graph in Figure 5 shows the same model as Figure 2 but using $\ln(Y)$ as the dependent variable. Note that the large difference in spread across the design points is reduced. In addition, the fit, as measured by R^2, is better. And it is monotonically increasing, as required. When viewed in the untransformed scale, the metamodel still fails to capture the extreme nonlinearity at the right. Using a quadratic regression improves the fit, but still fails to capture the extreme nonlinearity. For more on transformations, see Montgomery (2012).

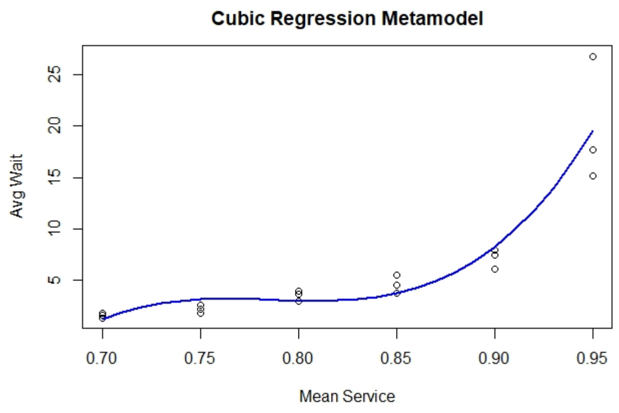

Figure 4: The fitted cubic regression metamodel is not monotonic.

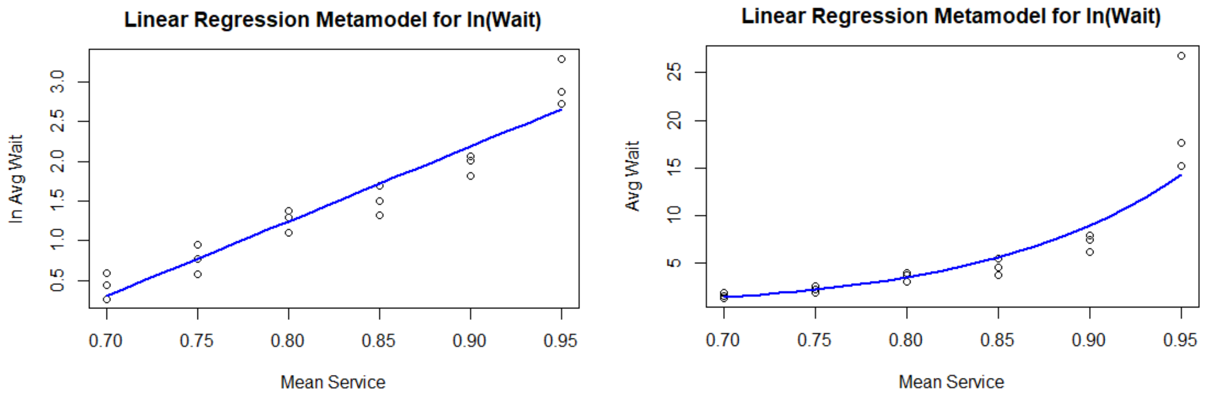

Figure 5: Transforming Y to ln(Avg Wait) reduces heteroscedasticity and improves fit.

4.2 Gaussian Process (Kriging) Metamodels

The simplest Gaussian Process (sometimes identified as Kriging) model is:

$$Y(x) = \beta_0 + M(x), \tag{5}$$

where M is the realization of a mean zero random field. That means it is a function drawn at random from the set of all functions whose nearby values are correlated according to a prespecified spatial correlation function. For that reason, these models are also called spatial correlation models. As in (3), the realization can be interpreted as $g_1(x) = M(x)$ and $\beta_1 = 1$. Other regression-type g terms might be added, but typically the intercept β_0 and $M(x)$ are all that is needed for a good fit. For stochastic GP models, a term $\varepsilon(x)$ is added to (5). When the randomness is assumed to be Gaussian, the models are called Gaussian Process (GP) regression models. GP models approximate deterministic response functions, since once the realization occurs, the model (5) has no intrinsic randomness. Ankenman et al. (2010) modeled $\varepsilon(x)$ added to (5), by a second spatial correlation model. The fitted stochastic GP metamodel for the mean response is:

$$\hat{Y}(x) = b_0 + [t^2 R_M(\hat{\theta}) + \hat{\Sigma}_\varepsilon]^{-1}(\bar{Y} - b_0 1_k) \tag{6}$$

where t^2 and $\hat{\theta}$ are spatial correlation parameters estimated from the experimental data, $\hat{\Sigma}_\varepsilon$ is the sample intrinsic error covariance matrix (in their experiments no covariances were included), $R_M(\hat{\theta})$ is an approximate spatial correlation matrix computed using $\hat{\theta}$, and 1_k is a k-dimensional vector of ones. Popular stochastic GP modeling software also implements a version of (6), with diagonal $\hat{\Sigma}_\varepsilon$, characterized as the *nugget*.

GP and stochastic GP models have great flexibility. They can model more complex response function shapes than is possible with polynomial regression metamodels. If one requires a global approximation to a nonlinear response, GP metamodels are an attractive alternative to linear regression models that use polynomial *g* functions. This comes at a cost. First, the model is more complex to fit, and GP or stochastic GP modeling capability is not available in some statistical packages. Second, fitted model coefficients give some indication of how rapidly the response changes as components of *x* change, but the detailed insight available from a fitted linear regression model cannot be obtained. Further, if experimental run conditions are scarce, predictions in design parameter space between experimental runs can be significantly in error due to mean reversion (see Figure 1 in Erickson et al. (2018)). These models can be sensitive to parameter choices (Gramacy and Lee 2012) – the example below illustrates this. Because this is an introductory tutorial, the best advice here is to educate yourself thoroughly in these models, and their potential shortfalls.

For Gaussian process regression, the R package *mlegp* is used (Dancik 2018). For this software the stochastic modeling is similar to that in Ankenman et al. (2010). The stochastic variance terms can be estimated directly at each design point using the replication sample variance, provided to *mlegp*; otherwise *mlegp* uses a pooled estimate, identical at every design point.

Figure 6 shows the fitted GP model for the M/M/1 experimental data using the stochastic variance terms estimated locally from the replication variance at each point. Notice that the prediction is poor at high mean service time. This is because the variance is allowed to differ across mean service values, and the high variance at high mean service means the prediction relies heavily on the spatial correlation, thus a predicted wait that is close to the waits at neighboring mean service times. If one uses a constant variance nugget, one might expect a closer fit at the high end, since the variance will be less, and the response values observed for mean service time = .95 would be given more weight in the prediction.

Figure 7 shows the resulting fit. While the prediction at high mean service time is improved, the metamodel now exhibits undulation in its fit, for a function known to be monotonic. Monotonicity for Gaussian process regression cannot be enforced in any straightforward way, but see Kleijnen and van Beers (2013). *This serves as a warning on the use of Gaussian process models. Note that in high dimensions, nonmonotonic structure is difficult to detect.*

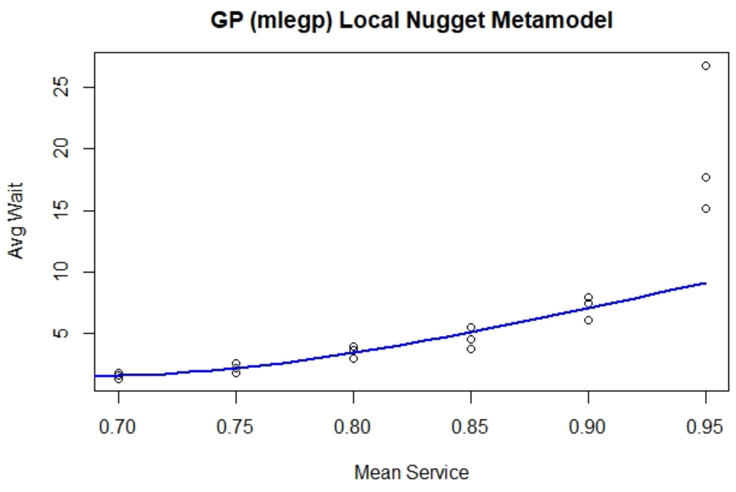

Figure 6. Gaussian process model fit to the M/M/1 simulation data, locally estimated variance.

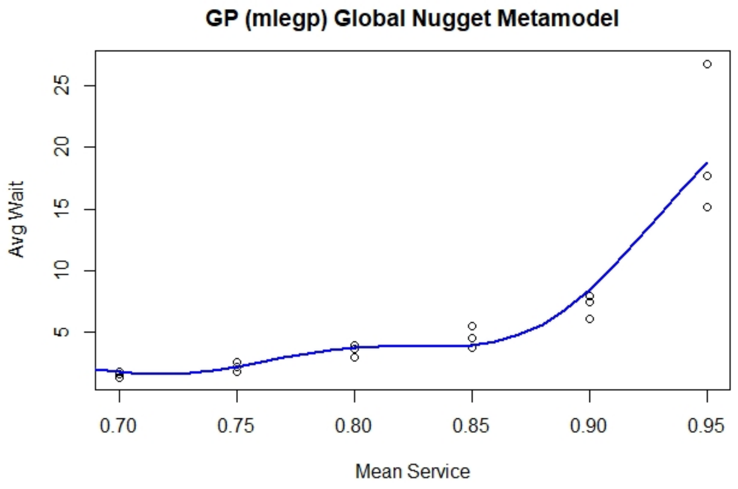

Figure 7: Gaussian process metamodel fit using common variance across all points.

4.3 Artificial Neural Network Metamodels

Artificial neural network (ANN) metamodels develop the approximation function *f*(*x*) as a composition of networked simple nonlinear functions with adjustable coefficients. A feedforward ANN is an acyclic network of simple nonlinear (neuron) functions, each using a linear combination of outputs from the direct predecessor nodes. The ANN model can be thought of as a nested form of equation (3), where the *Y* output is defined node-by-node with a *g* function, e.g., *g*(*s*) = = 1/(1+e^{-s}) applied to each weighted sum *s* of the immediate predecessor nodes. There is one input node for each component of *x*. Layers between the input and output nodes are called hidden layers. For example, an ANN with five input components and two hidden 3-node layers (with nodes indexed by *i* and *h*) would have mathematical form:

$$f(x) = g(w_{30} + \sum_{h=1}^{3} w_{3h} g(w_{2h0} + \sum_{i=1}^{3} w_{2hi} g(w_{1i0} + \sum_{j=1}^{5} w_{1ij} x_j))) \tag{7}$$

where the *w* values in (7), analogous to *b* values in (4), would be fitted via least squares. ANN fidelity and flexibility make them attractive for complex metamodeling and metamodel-based optimization (Dunke and Nickel 2020). They are often referred to as neural networks, although ANN is more formally correct.

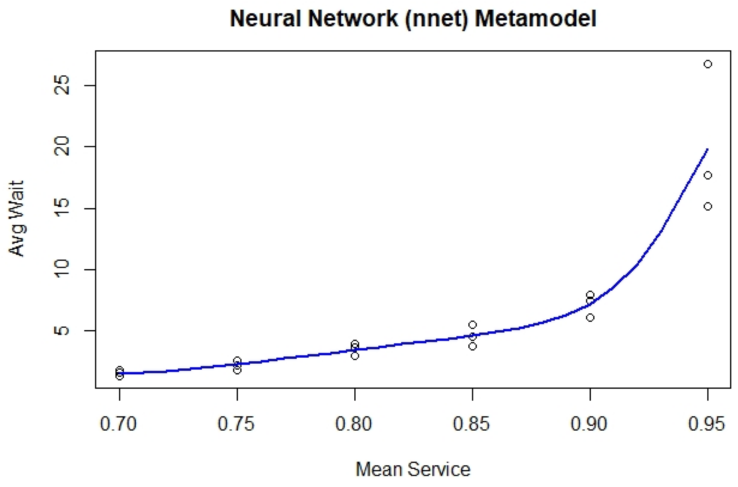

Figure 8: ANN metamodel fit using `nnet` in R.

ANNs can provide very good fit to nonlinear simulation response functions. Figure 8 shows the result using the package `nnet` (nnet 2022) and specifying one hidden layer with two nodes for the network. Although ANN software is readily accessible in R and other programming languages, implementation of ANN metamodels must be undertaken with caution. Coding the independent variables and choosing an ANN structure require care and sophistication. In an exploratory study (Fonseca et al. 2003) the authors note "the ANNs capabilities of effectively learning were highly restricted by the selected codification scheme." This assessment is further supported by Al-Hindi (2004) and Can and Heavey (2012), who explored the predictive ability of artificial neural networks for simple (s, S) simulation models.

ANNs are viewed as black boxes. While insight is possible from either linear regression coefficients and to a lesser extent from spatial correlation parameters, ANN coefficients generally do not provide insight on the impact of independent variables on simulation output. Graphical presentations of the neural network edge weights are available, but these provide little understanding of the nature of the response.

5 CHOOSE AN EXPERIMENT DESIGN

Metamodels are fitted using a designed simulation experiment. Experiment design types include random designs, optimal designs, combinatorial designs, Latin hypercube designs, orthogonal arrays, uniform designs, mixture designs, sequential designs, and factorial and fractional-factorial designs. The number and kinds of terms to be fitted place constraints on the minimum number of runs and minimum number of design parameter levels tested. Barton (2021) and Sanchez et al. (2021) are references for this activity.

5.1 Experiment Designs for Linear Regression

Experiment designs for regression are well-developed. There is a clear link between the form of the model being fitted and the kind of experiment design that is preferred. Typically, regression designs are either factorial designs or fractional factorial designs. Factorial designs are based on a grid, with each factor tested in combination with every level of every other factor. Factorial designs are attractive for three reasons: i) the number of levels that are required for each factor is one greater than the highest-order power of that variable in the model, and the resulting design permits the estimation of coefficients for all cross-product terms ii) they are probably the most commonly used class of designs, and iii) the resulting set of run conditions are easy to visualize graphically for as many as nine factors.

The disadvantage of factorial designs is that they require a large number of distinct runs when the number of factors and/or the number of levels of the factors are large. In this case, fractional-factorials are often employed. Table 2 gives some guidance on experiment designs appropriate for regression modeling, depending on the purpose and nature of the mode, but is not comprehensive. For example, it does not include sequential designs, e.g., Wan et al. (2009).

Table 2: Experiment designs for linear regression metamodels.

Objective	**Minimum Size Factorial Designs**	**References**
Find Influential Design Params	Saturated and supersaturated fractional factorial	Lin (1993), Li and Lin (2003)
Sensitivity	Saturated and resolution III Plackett-Burman fractional factorial	Sanchez and Sanchez (2005), Kleijnen (2008b)
Insight	3-level full or fractional factorial or central composite (more than 3 levels needed to check lack of fit)	Montgomery (2012), Kleijnen (2017), Sanchez et al. (2021)
Optimization	3- or more level fractional factorial	Montgomery (2012), Law (2014)

5.2 Experiment Designs for Gaussian Process Regression

Optimal experiment designs for GP metamodels remains an active field. Factorial designs have been found to work poorly with deterministic Kriging models. Latin hypercube designs are often used for GP models, but it is important to sample many Latin hypercube designs and choose one with good properties; for example, the design that maximizes the minimum distance between any two points (maximin). Alternatives are Hammersley sampling sequences, orthogonal arrays, and uniform designs. There are many references for such designs, but see Morris and Mitchell (1995) Sanchez and Sanchez (2019) for examples.

5.3 Experiment Designs for Artificial Neural Network Regression

ANN regression is less sensitive to the experiment design than GP regression. Experiment designs described above for linear regression and for GP regression have been used. The critical aspect of experiment design for neural network models is to scale the independent variables to [0,1] or +/- 1. While neural network fitting software typically allows the option of keeping the dependent variable in the original units, I have found the fitting to be more reliable when the dependent variable is scaled to [0,1] as well. For the M/M/1 example, the x variable was already scaled between 0 and 1 (.7 to .95), and the Y variable was divided by 30 to ensure values between 0 and 1.

6 CONDUCT RUNS AND FIT MODEL

Unlike physical experiments, external environmental factors generally do not affect simulation results. Once the model type and design selection steps are complete, running the experimental conditions is straightforward, provided the experimenter keeps the simulation output results attached to the correct values of the design parameter settings for each run.

Reusing random number streams for runs with different design parameter settings can induce correlation. Correlation can be hard to achieve, but it can result in better fits for regression metamodels (Schruben and Margolin 1978; Tew and Wilson 1992). Work by Chen et al. (2012) found that correlation induction using common random numbers was not effective for GP modeling, but more recent work seems to indicate an advantage for GP metamodel-assisted optimization (Pearce et al. 2019).

6.1 Software for Metamodeling

Virtually all DES software allows input of a spreadsheet of simulation run parameters (created using your favorite DOE software, such as Matlab, Minitab, R, SAS/JMP or at the SEED Center for Data Farming) and spreadsheet output of results, for analysis using your favorite metamodeling software. Built-in metamodeling capabilities seem rare in DES software, based on the survey by Swain (2019). The survey had no specific entry for metamodeling, but software identifying DOE capability includes AnyLogic, Flexsim, Frontline Solver, INCONTROL, ProModel, Siemens PLM, SIPmath Modeler. SAS/JMP and Arena were not participants in the 2019 survey, but both SAS and JMP software identify the ability to link design of experiments capability with simulation and back to analysis/fitting metamodels.

When DOE and metamodeling are not supported by your simulation environment, you should be prepared to provide spreadsheet input for the simulation experiment design and receive spreadsheet output from the runs to use in building your metamodel. The DOE and metamodel fitting can be managed by statistical software. Commercial software such as Excel-based packages, Minitab, or SAS/JMP provides an easy, menu-driven interface, while the freely available R software requires programming ability.

R provides free, high-quality statistical tools and includes many metamodel options. What makes R so powerful are its packages; there are over 5200 on the CRAN download site https://cran.r-project.org/. The free package RStudio (RStudio 2022) provides an integrated software development environment (IDE) that minimizes the R programming effort. All of the examples in this tutorial were programmed in R using RStudio on a PC. Posts on my personal page https://sites.psu.edu/russellbarton/ have the R code and Excel data file available for download.

7 VALIDATING METAMODEL ADEQUACY

The fitted model must be checked to see if the fidelity is adequate for the intended use. See Kleijnen and Sargent (2000) for more details on this process. For a regression metamodel for screening, simple statistical significance checks may be sufficient. For purposes where fidelity is important, additional goodness-of-fit tests are employed.

There are well-developed goodness-of-fit tests for regression that appear in all commercial packages. To use them you will need to include more than the minimum number of design parameter levels. Further, confidence intervals give uncertainty about the predicted mean simulation output, and prediction intervals characterize uncertainty for possible results of a single simulation run. For regression models, mean squared error (MSE) or its square root (RMSE) is provided automatically with the fit, as is R^2. High R^2 values can occur when there are just a few extreme values of x and correspondingly high or low values of the response. MSE or RMSE will give a better assessment of fit in this case.

These concepts exist for GP models, but their theoretical justification is weaker. A general-purpose measure of fit that can be used outside the regression setting is to leave some design conditions and corresponding responses out of the simulation runs used to fit the model and then check the error of the fitted model against the output of simulation runs left out of the fitting process. This *cross-validation* process can be computationally expensive if it is repeated for each possible omission. Meckesheimer et al. (2002) provide some efficient and effective assessment methods of this sort. Repeated instances of a cross-validation process is called *k-fold cross-validation*. Cross-validation is built into the `train()` function in the caret package in R (caret 2022), but beyond the scope of this tutorial.

8 BEWARE OF PITFALLS

While metamodels provide a powerful tool in the simulationist's toolkit, care is required to gain the potential benefits. One needs to be alert to the following:

1. Failing to scale predictors to +/-1.
2. Using too-high order for polynomial regression.
3. Using a variance stabilizing transformation when there is no heteroscedasticity.
4. Using a GP model when simplifying response surface properties are known.
5. Trusting the universal fit properties of neural networks when fitting stochastic responses with replications.

For more information on the nature of these pitfalls and how to avoid them, see Barton (2023).

9 PUT THE METAMODEL TO USE – AND FURTHER STUDY

Assuming the fitted model passed the validation checks, it is ready to be used. Congratulations on successfully developing a metamodel! Remember though, that uses beyond the original purpose (e.g., using a screening metamodel for prediction or, worse, optimization) are not appropriate. To expand your understanding, many books on simulation have a chapter on the design of experiments, which usually covers metamodeling. Three books with comprehensive coverage are Friedman (1996), Kleijnen (2008a - ebook available online through some university libraries), and Law (2014).

ACKNOWLEDGMENTS

I have had the good fortune to benefit from the wisdom of the authors cited in this paper. I also thank the comments of one anonymous reviewer that helped improve the paper.

REFERENCES

Al-Hindi, H. 2004. “Approximation of a Discrete Event Stochastic Simulation Using an Evolutionary Artificial Neural Network”. *Journal of King Abdulaziz University-Engineering Sciences* 15: 125-138.

Ankenman, B., B. L. Nelson, and J. Staum. 2010. “Stochastic Kriging for Simulation Metamodeling”. *Operations Research* 58(2): 371-382.

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2009. *Discrete-Event System Simulation*. 5th ed. Englewood Cliffs, NJ: Prentice Hall.

Barton, R. R. 1992. “Metamodels for Simulation Input-Output Relations”. In *Proceedings of the 1992 Winter Simulation Conference*, edited by J. J. Swain, D. Goldsman, R. C. Crain and J. R. Wilson, 289-299. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Barton, R. R. 2009. “Simulation Optimization Using Metamodels”. In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 230-238. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Barton, R. R. 2015. “Tutorial: Simulation Metamodeling”. In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti,1765–1779. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Barton, R. R. 2020. “Tutorial: Metamodeling for Simulation”. In *Proceedings of the 2021 Winter Simulation Conference*, edited by K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, 1102-1116. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Barton, R. R. 2021. “Tutorial: Graphical Methods for the Design and Analysis of Experiments”. In *Proceedings of the 2021 Winter Simulation Conference*, edited by S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, 342-353. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Barton, R. R. 2023. “Metamodelling: Power, Pitfalls, and Model-Free Interpretation”. In *Proceedings of the Operational Research Society Simulation Workshop 2023 (SW23)*, edited by C. Currie and L. Rhodes-Leader, 48-62. Birmingham, UK: The Operational Research Society.

Barton, R. R., and M. Meckesheimer. 2006. “Metamodel-Based Simulation Optimization”. In *Simulation*, ed. S. G. Henderson and B. L. Nelson. Vol. 13, 535-574. Handbooks in Operations Research and Management Science. New York: Elsevier.

Barton, R. R., B. L. Nelson, and W. Xie. 2014. “Quantifying Input Uncertainty via Simulation Confidence Intervals”. *INFORMS Journal on Computing* 26(1): 74-87.

Batur, D., J. M. Bekki, and X. Chen. 2018. “Quantile Regression Metamodeling: Toward Improved Responsiveness in the High-Tech Electronics Manufacturing Industry”. *European Journal of Operational Research* 264(1): 212-224.

Blanning, R. W. 1974. “The Sources and Uses of Sensitivity Information”. *INFORMS Journal on Applied Analytics* 4(4): 32-38.

Blanning, R. W. 1975. “The Construction and Implementation of Metamodels”. *Simulation* 24: 177-184.

Burdick, D. S., and T. H. Naylor. 1966. “Design of Computer Simulation Experiments for Industrial Systems”. *Communications of the ACM* 9(5): 329-339.

Can, B., and C. Heavy. 2012. “A Comparison of Genetic Programming and Artificial Neural Networks in Metamodeling of Discrete-Event Simulation Models”. *Computers & Operations Research* 39:424-436.

caret. 2022. caret: Classification and Regression Training. https://cran.r-project.org/web/packages/caret, accessed 21st April 2022.

Chen, X., B. E. Ankenman, and B. L. Nelson. 2012. “The Effects of Common Random Numbers on Stochastic Kriging Metamodels”. *ACM Transactions on Modeling and Computer Simulation* 22(2): 1-20.

Dancik, G. 2018. mlegp: An R Package for Gaussian Process Modeling and Sensitivity Analysis. https://cran.r-project.org/web/packages/mlegp/vignettes/mlegp.pdf, accessed 27th April 2020.

Dellino, G., J. P. C. Kleijnen, and C. Meloni. 2009. “Robust Simulation-Optimization Using Metamodels”. In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 540-549. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Dunke, F., and S. Nickel. 2020. “Neural Networks for the Metamodeling of Simulation Models with Online Decision Making”. *Simulation Modelling Practice and Theory* 99(102016): 1-18.

Erickson, C. B., B. E. Ankenman, and S. M. Sanchez. 2018. “Comparison of Gaussian Process Modeling Software”. *European Journal of Operational Research* 266(1): 179-192.

Fishman, G. S., and P. J. Kiviat. 1967. “The Analysis of Simulation-Generated Time Series”. *Management Science* 13: 525-557.

Fonseca, D. J., D. O. Navaresse, and G. P. Moynihan. 2003. “Simulation Metamodeling through Artificial Neural Networks”. *Engineering Applications of Artificial Intelligence* 16(3): 177-183.

Friedman, L. W. 1996. *The Simulation Metamodel*. New York: Springer.

Gramacy, R. B., and H. K. H. Lee. 2012. “Cases for the Nugget in Modeling Computer Experiments”. *Statistics and Computing* 22(3): 713-722.

Kleijnen, J. P. C. 1975. “A Comment on Blanning’s ‘Metamodel for Sensitivity Analysis: The Regression Metamodel in Simulation.’” *INFORMS Journal on Applied Analytics* 5(3): 21-23.

Kleijnen, J. P. C. 2008a. *Design and Analysis of Simulation Experiments*. Berlin: Springer.

Kleijnen, J. P. C. 2008b. “Design of Experiments: Overview”. In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R., R. Hill, L. Mönch, O. Rose, O. Jefferson, J. W. Fowler, 479-488. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kleijnen, J. P. C. 2017. “Regression and Kriging Metamodels with Their Experimental Designs in Simulation: A Review”. *European Journal of Operational Research* 256(1): 1-16.

Kleijnen, J. P. C., and R. G. Sargent. 2000. “A Methodology for Fitting and Validating Metamodels in Simulation”. *European Journal of Operational Research* 120(1): 14-29

Kleijnen, J. P. C., and W. C. M. van Beers. 2013. “Monotonicity-Preserving Bootstrapped Kriging Metamodels for Expensive Simulations”. *Journal of the Operational Research Society* 64(5): 708-717.

Law, A. M. 2014. *Simulation Modeling and Analysis*. 5th ed. New York: McGraw Hill Higher Education.

Law, A. M. 2017. “A Tutorial on Design of Experiments for Simulation Modeling”. In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 550-564. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Li, R., and D. K. J. Lin. 2003. “Analysis Methods for Supersaturated Design: Some Comparisons”. *Journal of Data Science* 1: 249-260.

Lin, D. K. J. 1993. “A New Class of Supersaturated Designs”. *Technometrics* 35(1): 28-31.

Liu, H., J. Cai, and Y.-S. Ong. 2018. “Remarks on Multi-Output Gaussian Process Regression”. *Knowledge-Based Systems* 144: 102-121.

Meckesheimer, M., R. R. Barton, F. Limayem, T. Simpson, and B. Yannou. 2001. “Metamodeling of Combined Discrete/Continuous Responses”. *AIAA Journal* 39(10): 1950-1959.

Meckesheimer, M., A. J. Booker, R. R. Barton, and T. W. Simpson. 2002. “Computationally Inexpensive Metamodel Assessment Strategies” *AIAA Journal* 40(10): 2053-2060.

Montgomery, D. C. 2012. *Design and Analysis of Experiments*. 8th ed. Hoboken, New Jersey: Wiley.

nnet. 2022. nnet: Feed-Forward Neural Networks and Multinomial Log-Linear Models. https://cran.r-project.org/web/packages/nnet, accessed 21st April 2022.

Morris, M. D., and Mitchell, T. J. 1995. “Exploratory Designs for Computational Experiments”. *Journal of Statistical Planning and Inference* 43(3): 381-402.

Pearce, M., M. Poloczek, and J. Branke. 2019. “Bayesian Optimization Allowing for Common Random Numbers”. http://arxiv.org/abs/1910.09259, accessed 27th April 2020.

RStudio. 2022. RStudio. https://rstudio.com/, accessed 21st April 2022.

Sanchez, S. M., and P. J. Sanchez. 2005. “Very Large Fractional Factorial and Central Composite Designs”. *ACM Transactions on Modeling and Computer Simulation* 15(4): 362-377.

Sanchez, P. J., and S. M. Sanchez. 2019. “Orthogonal Second-order Space-filling Designs with Insights from Simulation Experiments to Support Test Planning”. *Quality and Reliability Engineering International* 35(3): 854–867.

Sanchez, S. M., P. J. Sanchez, and H. Wan. 2021. “Work Smarter, Not Harder: A Tutorial on Designing and Conducting Simulation Experiments”. In *Proceedings of the 2021 Winter Simulation Conference*, edited by S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, 237-251. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sargent, R. G., and T. K. Som. 1992. “Current Issues in Frequency Domain Experimentation”. *Management Science* 38: 667-687.

Schruben, L. W., and V. J. Cogliano. 1987. “An Experimental Procedure for Simulation Response Surface Model Identification”. *Communications of the ACM* 30: 716-730.

Schruben, L. W., and B. H. Margolin. 1978. “Pseudorandom Number Assignment in Statistically Designed Simulation and Distribution Sampling Experiments”. *Journal of the American Statistical Association* 73(363): 504-520.

Swain, J. J. 2019. “2019 Simulation Software Survey”. *ORMS Today* 42(5): 36-49.

Tew, J. D., and J. R. Wilson. 1992. “Validation of Simulation Analysis Methods for the Schruben-Margolin Correlation-Induction Strategy”. *Operations Research* 40(1): 87-103.

Wan, H., B. E. Ankenman, and B. L. Nelson. 2009. “Improving the Efficiency and Efficacy of Controlled Sequential Bifurcation for Simulation Factor Screening”. *INFORMS Journal on Computing* 22(3): 482-492.

AUTHOR BIOGRAPHY

RUSSELL BARTON is Distinguished Professor of Supply Chain and Information Systems in the Smeal College of Business and Professor of Industrial Engineering at the Pennsylvania State University. His research interests include applications of statistical and simulation methods to system design and product design, manufacturing and delivery. His email address is rbarton@psu.edu and his homepage is https://sites.psu.edu/russellbarton/.