Computer experiments with big n: Has Gaussian process computation been tamed?

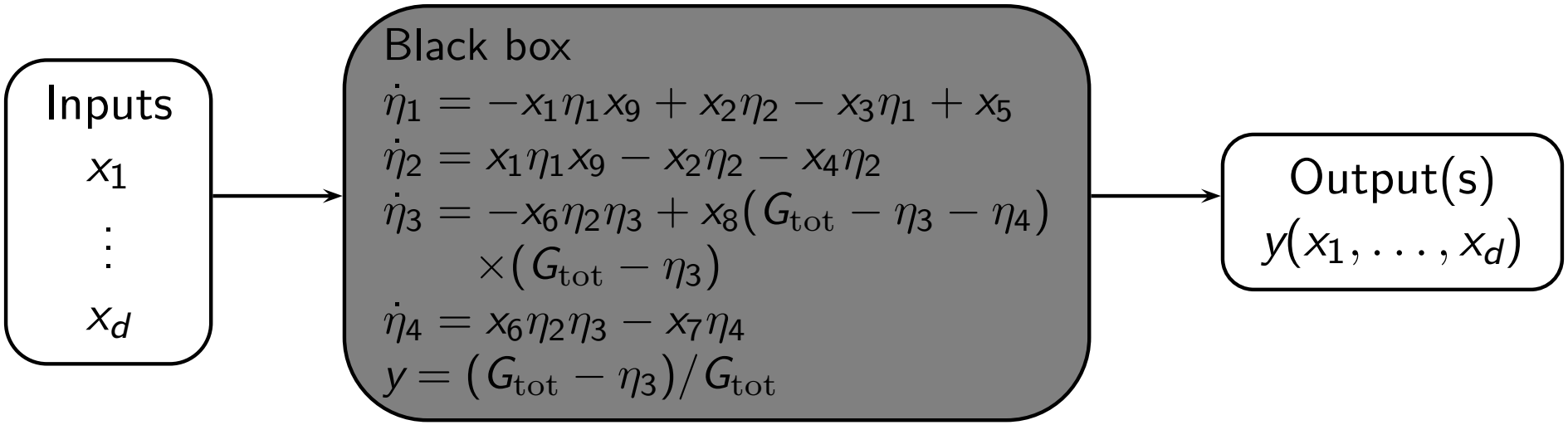
Sonja Surjanovic and William J. Welch

University of British Columbia

Workshop on Design of Experiments, April 30 – May 4, 2018
CIRM, Marseilles, France

Outline

Computer Code



Standard Gaussian Process (GP) Model

- d-dimensional vector of inputs $\mathbf{x}$
- Output $y(\mathbf{x})$
- Treat $y(\mathbf{x})$ as a realization of

$$Y(\mathbf{x}) = \text{regression model} + Z(\mathbf{x})$$

- $Z(\mathbf{x})$ hence $Y(\mathbf{x})$ is a correlated process
- The correlation function $R(\mathbf{x}, \mathbf{x}') \equiv R(Y(\mathbf{x}), Y(\mathbf{x}'))$ is the workhorse of the GP model

Standard Gaussian Process (GP) Model

- d-dimensional vector of inputs $\mathbf{x}$
- Output $y(\mathbf{x})$
- Treat $y(\mathbf{x})$ as a realization of

$$Y(\mathbf{x}) = \text{regression model} + Z(\mathbf{x})$$

- $Z(\mathbf{x})$ hence $Y(\mathbf{x})$ is a correlated process
- The correlation function $R(\mathbf{x}, \mathbf{x}') \equiv R(Y(\mathbf{x}), Y(\mathbf{x}'))$ is the workhorse of the GP model
- Sacks et al. (1989)



All the methods discussed today use this model in some form.

What's the Problem?

Computational complexity

- Training data of n runs at $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$
  - Key is the $n \times n$ correlation matrix

$$\mathbf{R} = R(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \quad \text{for } 1 \leq i, j \leq n$$

  - Maximum likelihood or Bayes MCMC needs $\mathbf{R}^{-1}$ and $\det(\mathbf{R})$, or the Cholesky decomposition
  - Computational time for one likelihood calculation is $O(n^3)$
  - Need 1000's or 10 000's likelihood calculations
- Prediction at N test points
  - Point prediction: $O(n)$ computation per prediction; $O(nN)$ for all test points
  - Predictive variance: $O(n^2)$ per prediction; $O(n^2 N)$ for all test points

Big n? Disclaimer

Dennis pointed out
big n to statisticians is not
so big. For GPs big is 1000's or 10 000's.

Big n? Disclaimer

Dennis pointed out
big n to statisticians is not
so big. For GPs big is 1000's or 10 000's.

- To illustrate methods, n will be really small: n
- Then some results for larger n

~~Student~~ Audience Participation

UBC Faculty of Science has an initiative in active learning

~~Student~~ Audience Participation

UBC Faculty of Science has an initiative in active learning

Has Gaussian process computation been tamed?

A No

B Yes

C I don't know

D Nobody knows

~~Student~~ Audience Participation

UBC Faculty of Science has an initiative in active learning

Has Gaussian process computation been tamed?

A No

B Yes

C I don't know

D Nobody knows

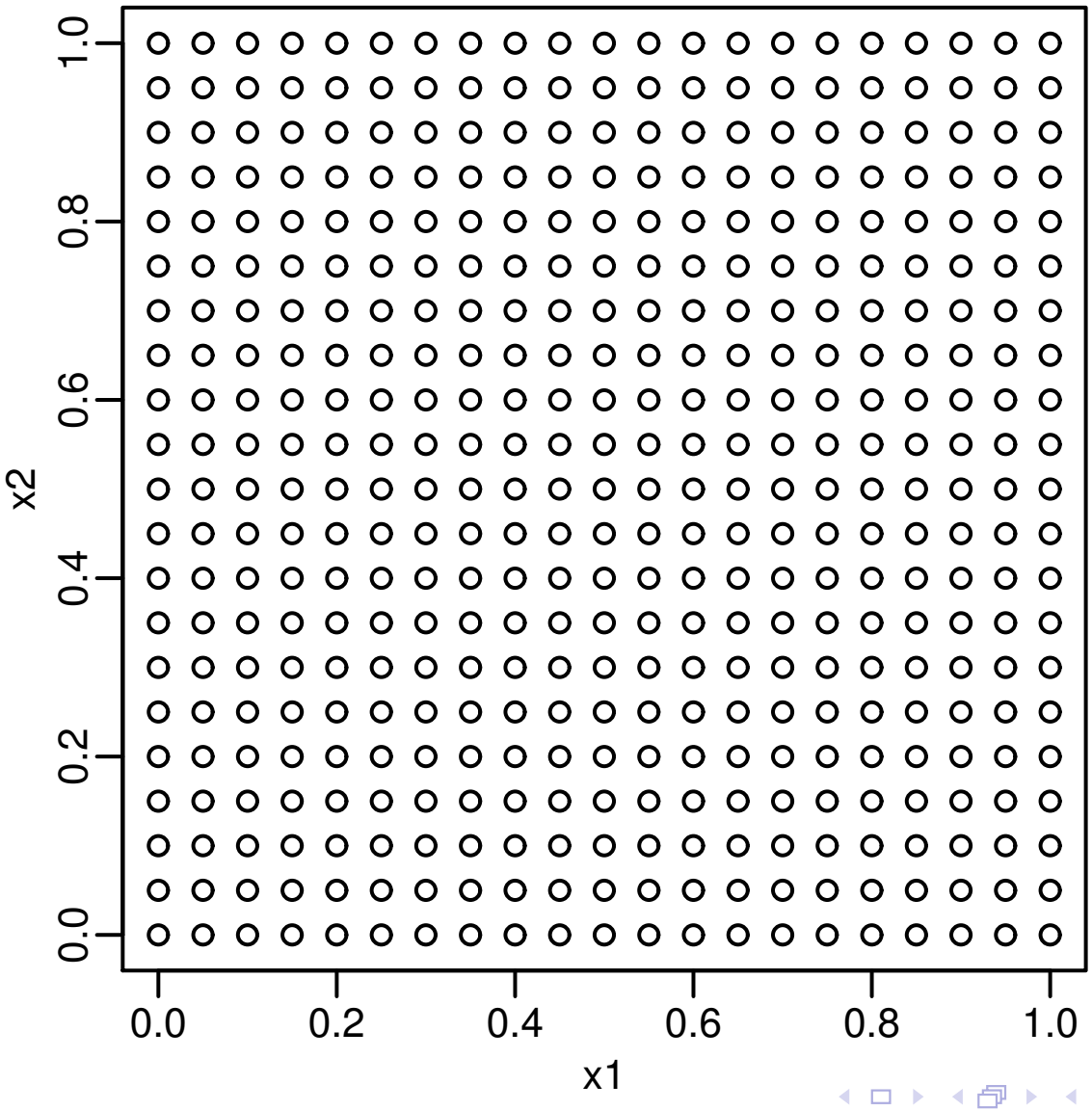
E Who cares? Isn't it dinner time yet?

Sparse Grid Designs for GPs

Same designs as Henry's talk

Grid Designs: Intuition

Design for $d = 2$ inputs on a 21×21 grid ($n = 441$)



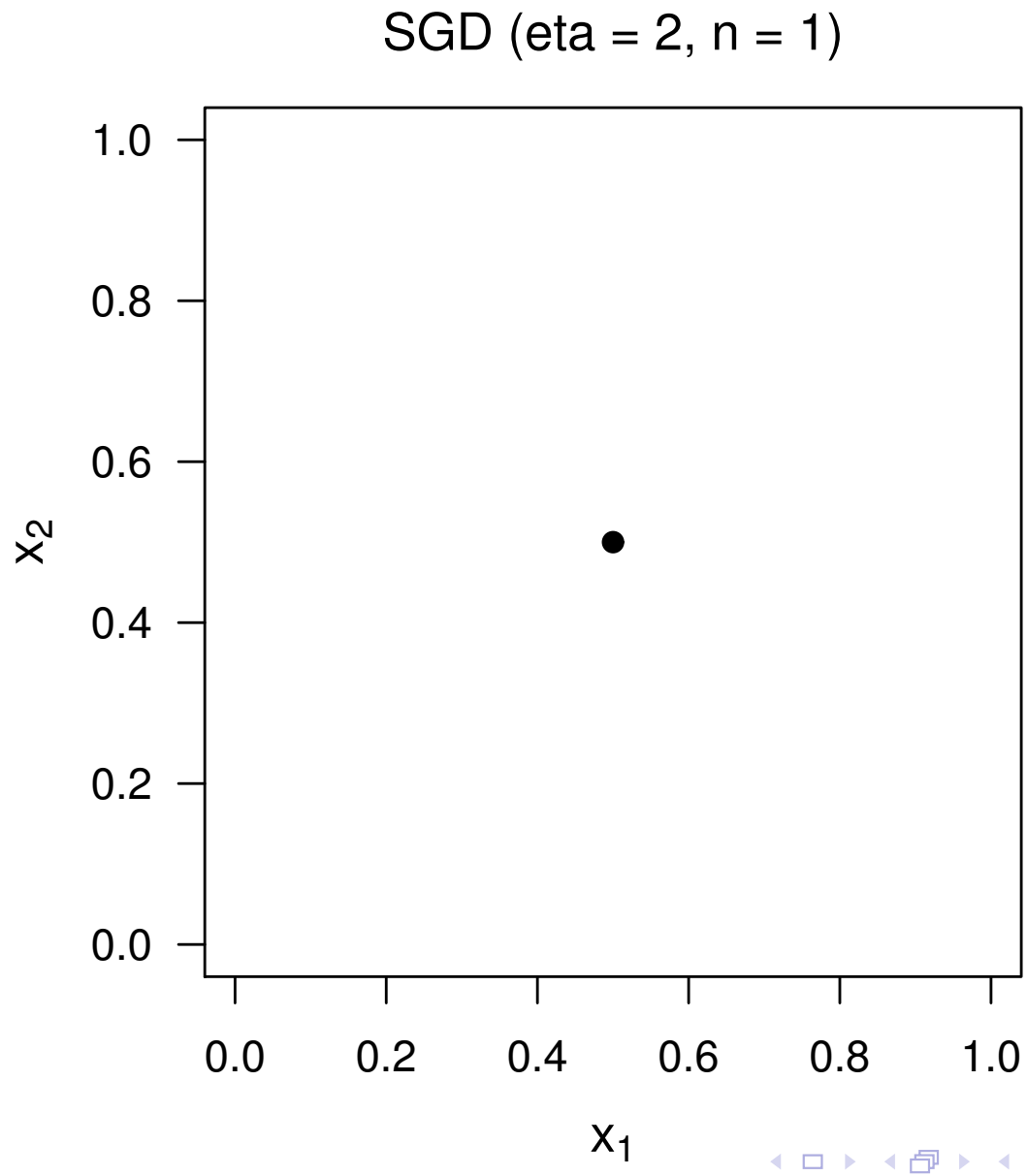
Grid Designs: Computational Complexity

- For the above 21×21 grid design

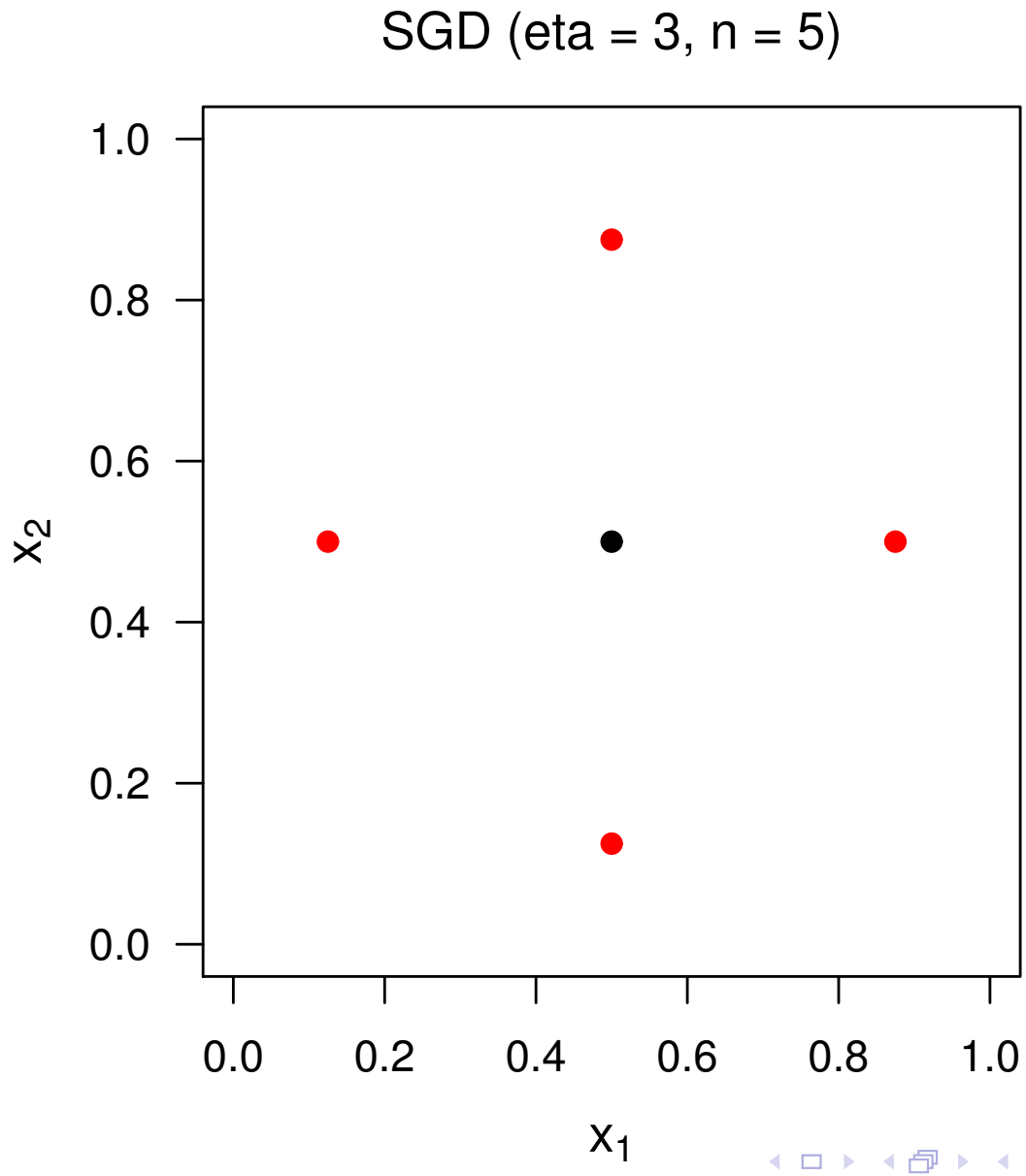
$$\underset{x_1 \text{ and } x_2}{\mathbf{R}_{441\times 441}} = \underset{x_1}{\mathbf{R}^{(1)}_{21\times 21}} \otimes \underset{x_2}{\mathbf{R}^{(2)}_{21\times 21}}$$

- $O(441^3)$ computation becomes $O(21^3) + O(21^3)$ computation
- In general, for $d = 2$ inputs $O(n^3)$ becomes $O(n^{3/2})$
- i.e., $O(n^{3/2})$ speed up
- For d inputs, $O(n^3)$ becomes $O(n^{3/d})$: even more relative speed-up
- But (dense) grid designs need too many computer-model runs, so ...

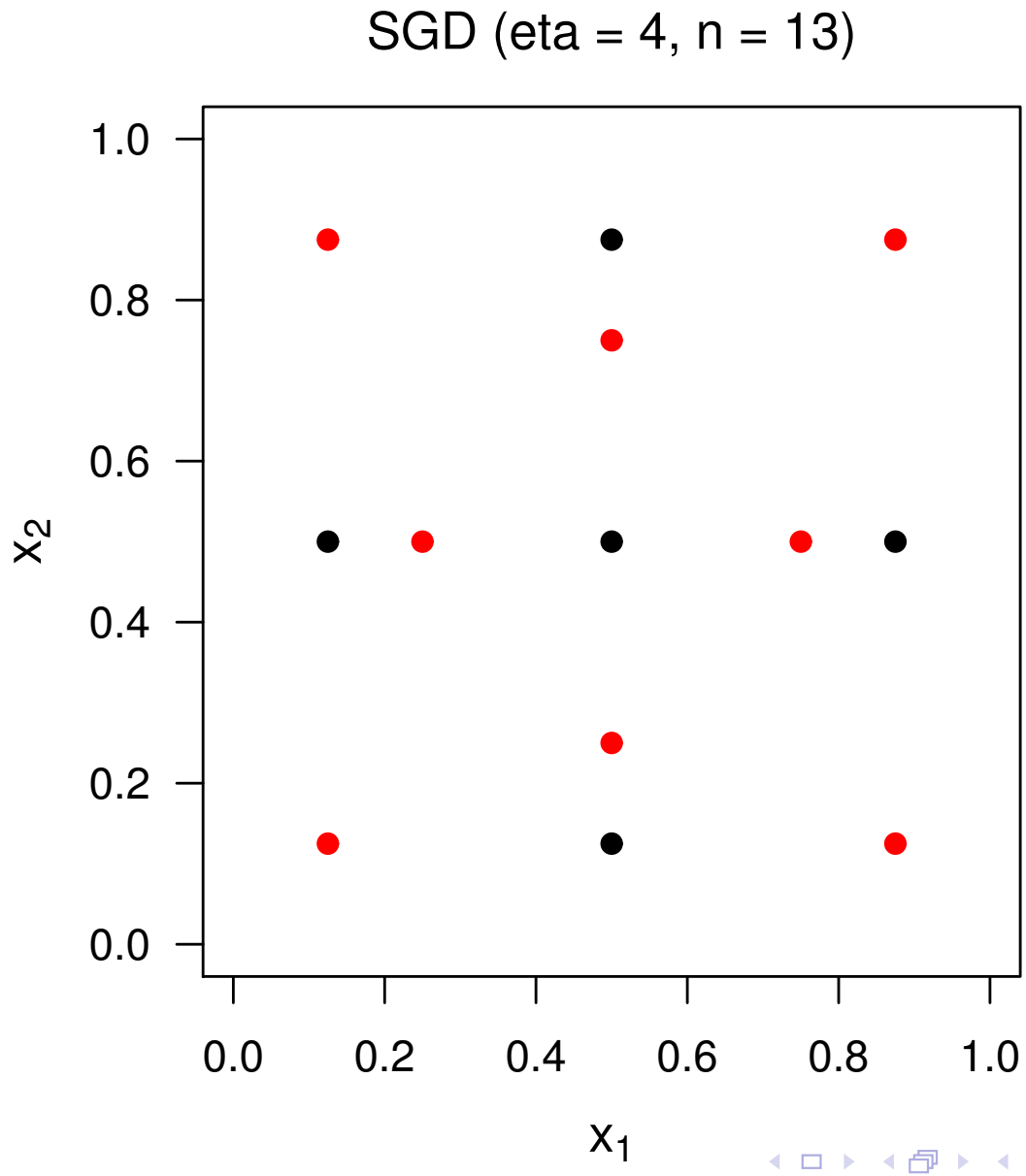
Sparse Grid Designs for GPs (SGDs, Plumlee, 2014)



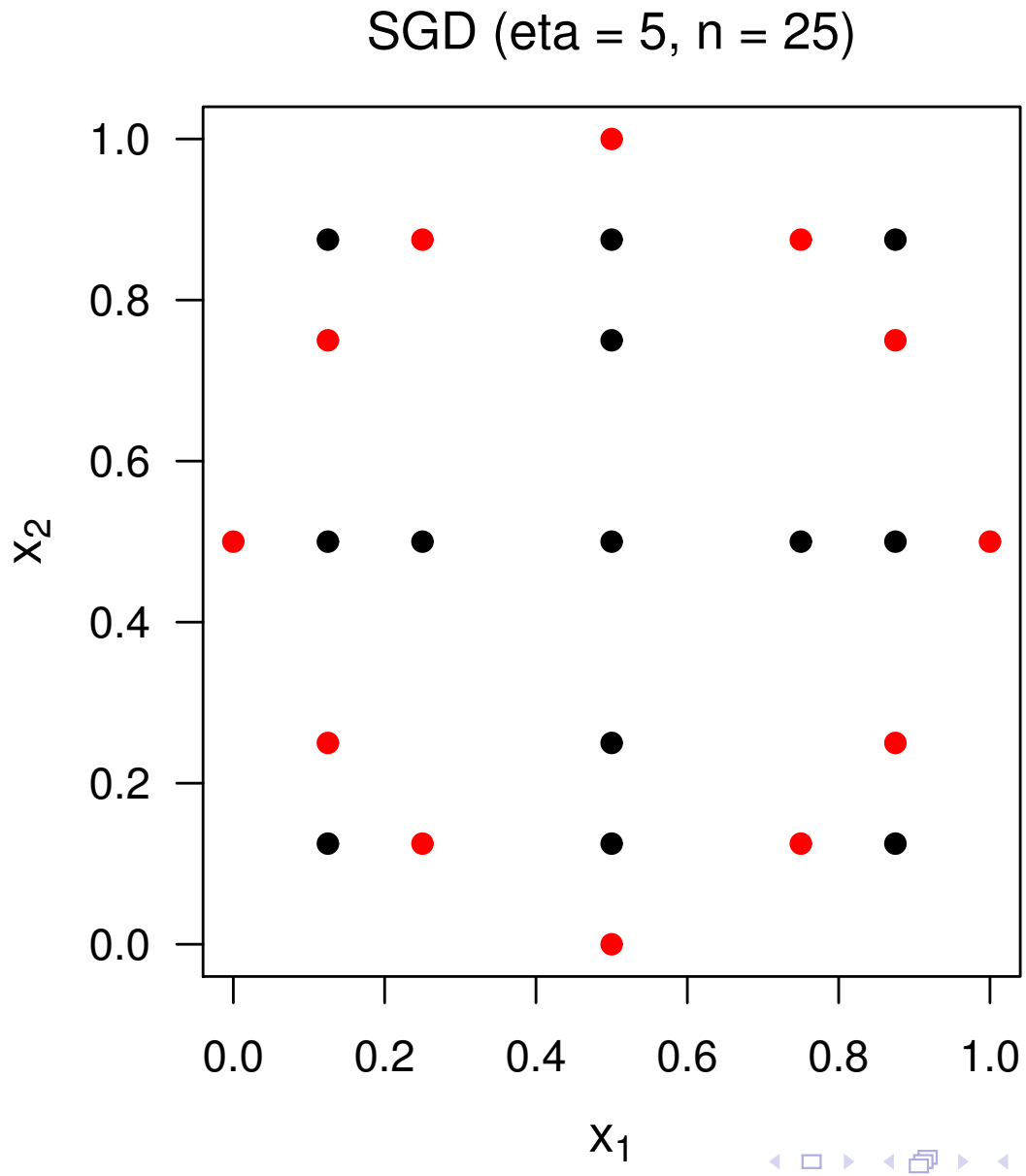
Sparse Grid Designs for GPs (SGDs, Plumlee, 2014)



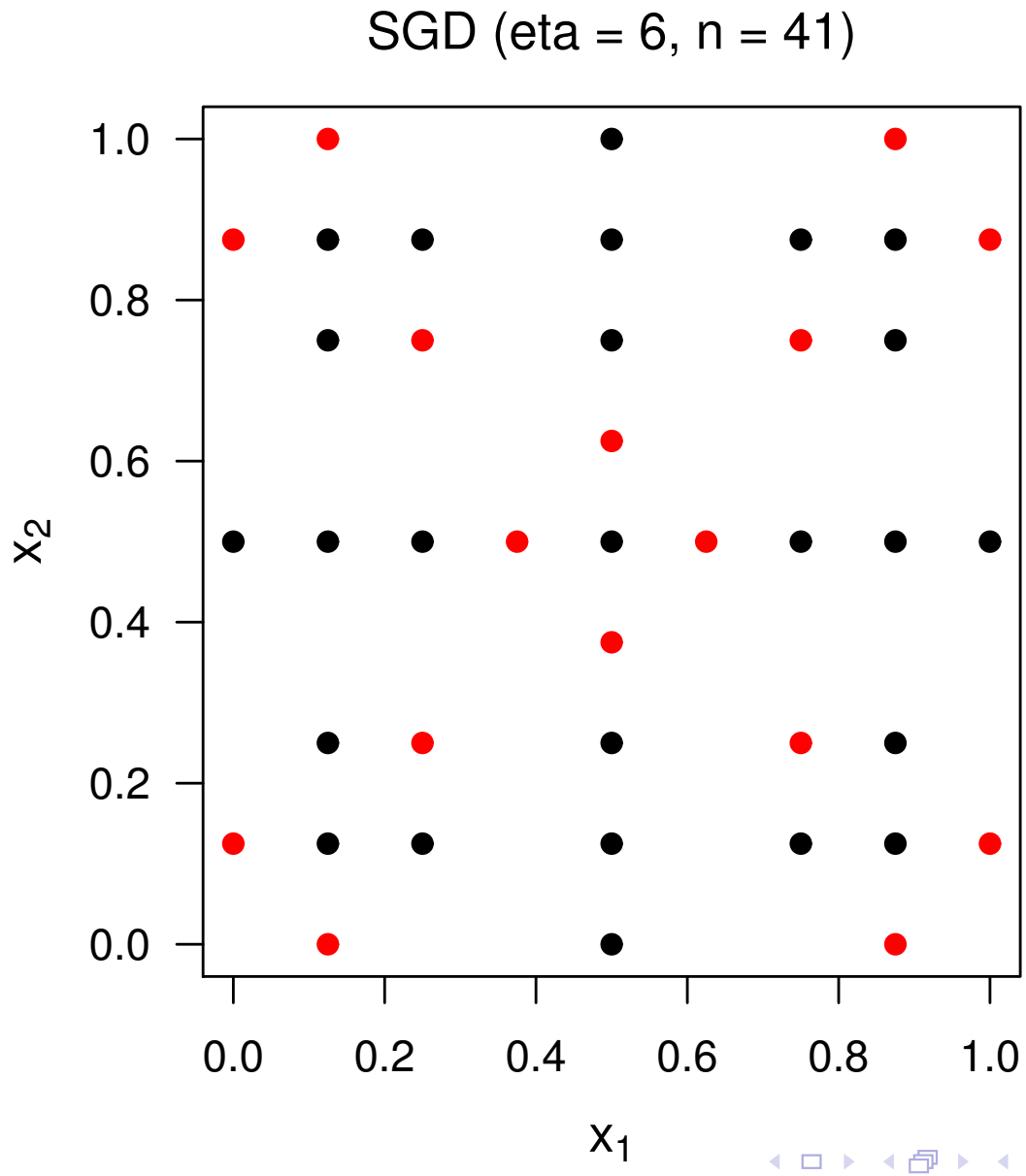
Sparse Grid Designs for GPs (SGDs, Plumlee, 2014)



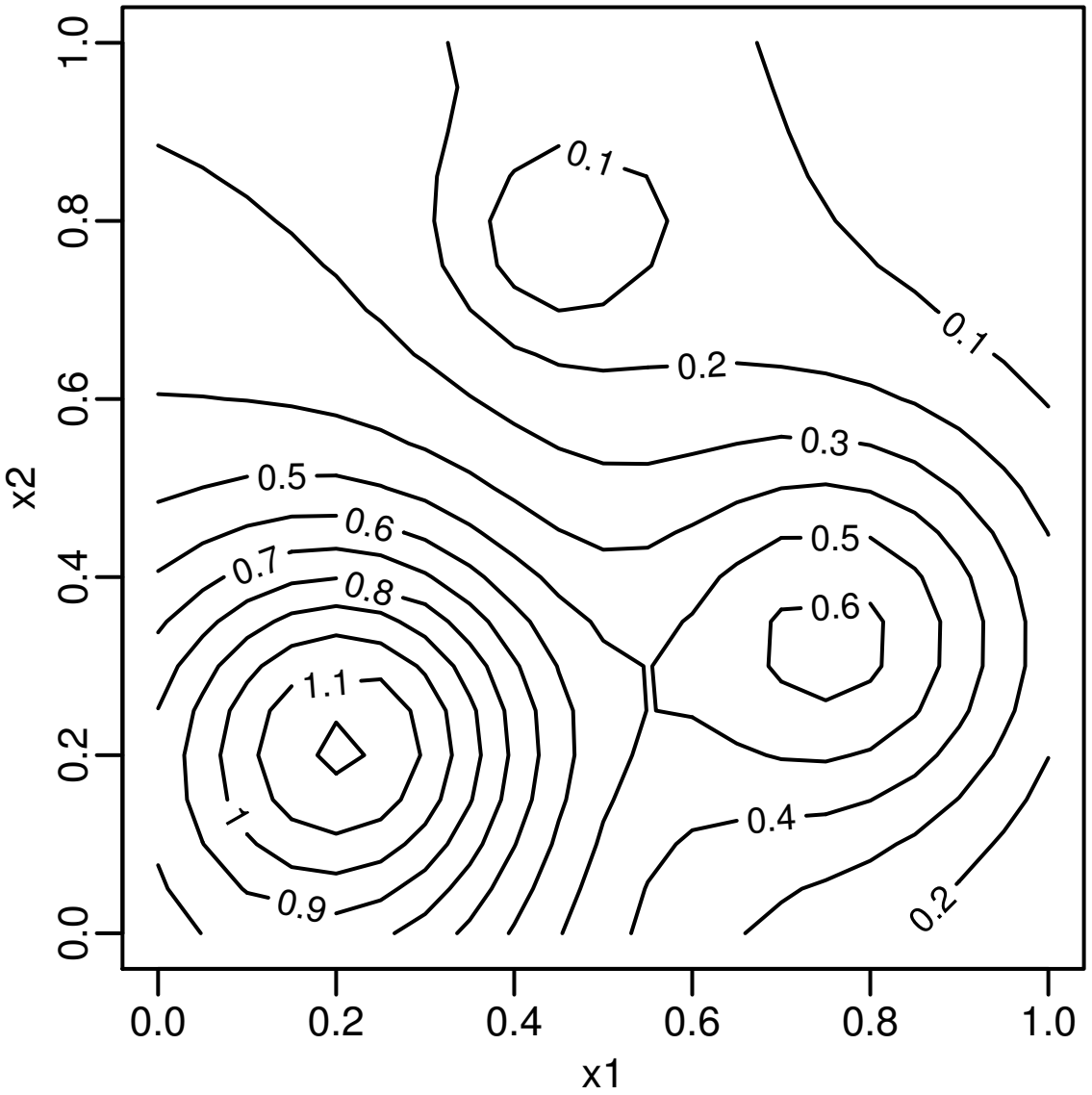
Sparse Grid Designs for GPs (SGDs, Plumlee, 2014)



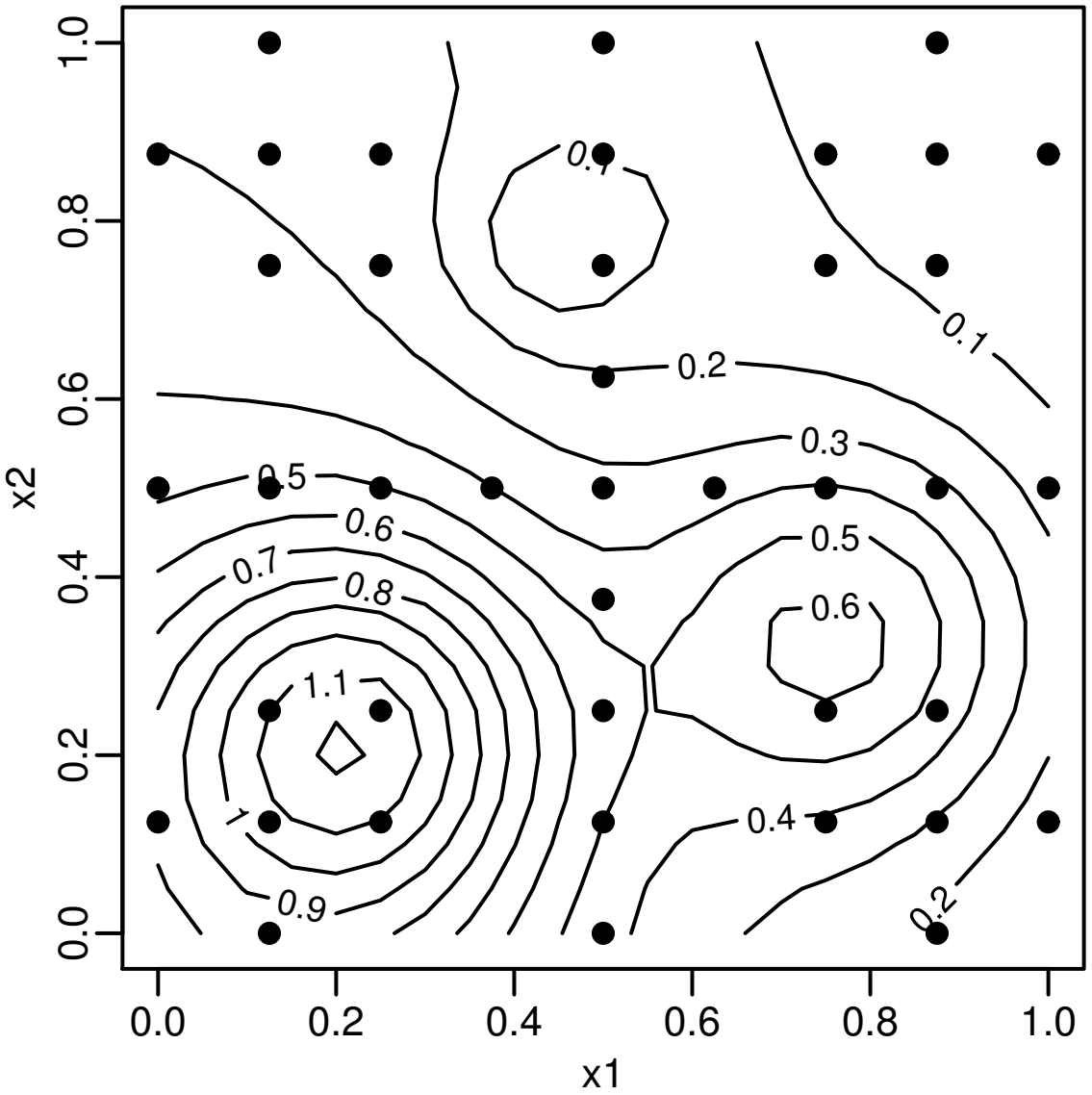
Sparse Grid Designs for GPs (SGDs, Plumlee, 2014)



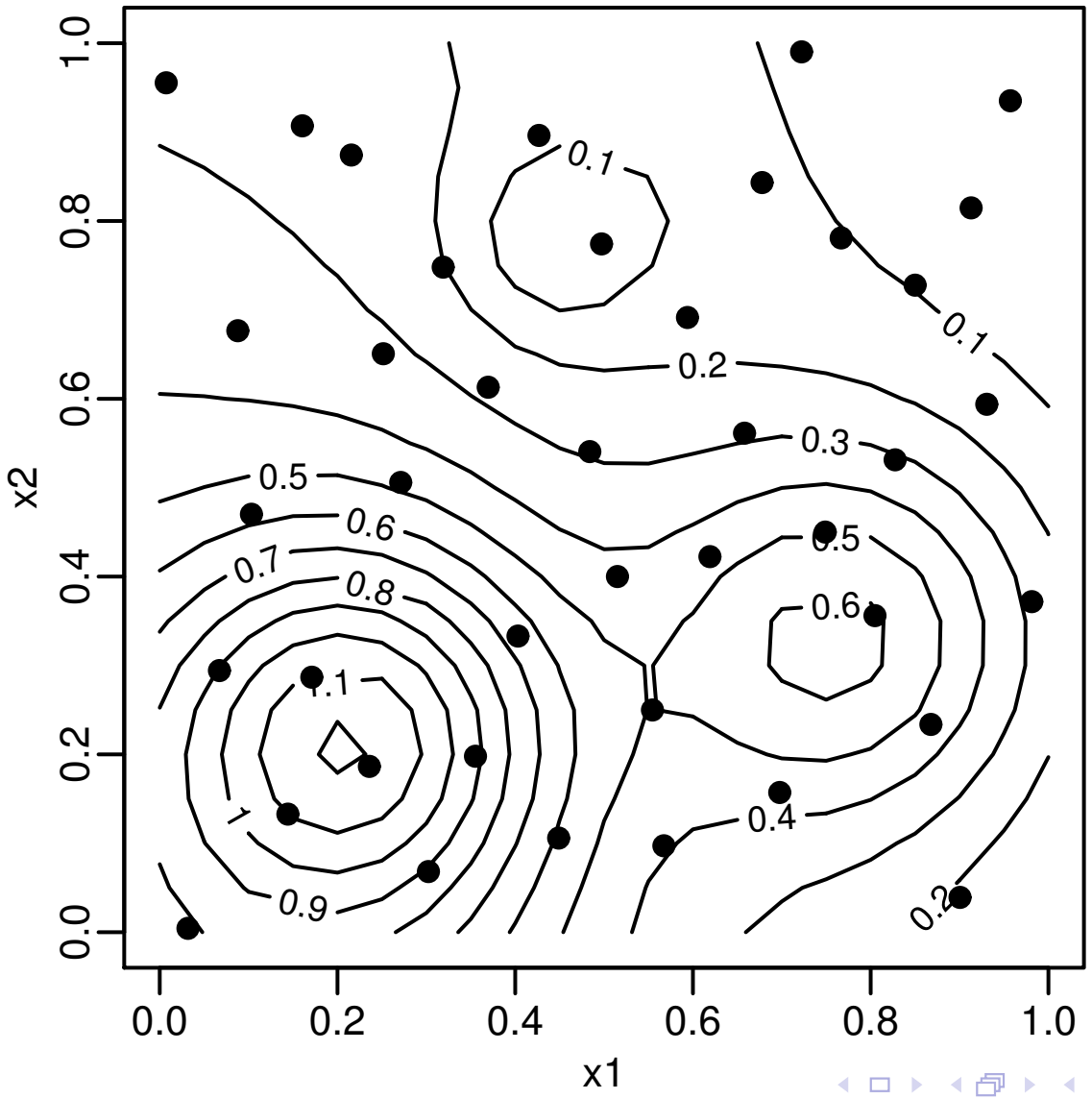
Franke's Function



Franke's Function and Sparse Grid Design ($n = 41$)



Franke's Function and Maximin Design ($n = 41$) for Comparison



Measures of Prediction Accuracy

- Prediction accuracy measured using "gold standard" hold-out test set
- $N = 1000$ or $10\,000$ random points $\mathbf{x}$ in the input space with y known
- Average error: Normalized root mean squared prediction error

$$\frac{\sqrt{\frac{1}{N}\sum_{\text{test points}}(y-\hat{y})^2}}{\text{test set standard deviation of } y}$$

- Worst error: Normalized max absolute prediction error

$$\frac{\max_{\text{test points}}|y-\hat{y}|}{\max_{\text{test points}}|y-\bar{y}|}$$

- Normalization: 0 = perfect, 1 = no better than predicting using $\bar{y}$

Franke's Function: Prediction Accuracy

(1000 test points from a random Latin hypercube)

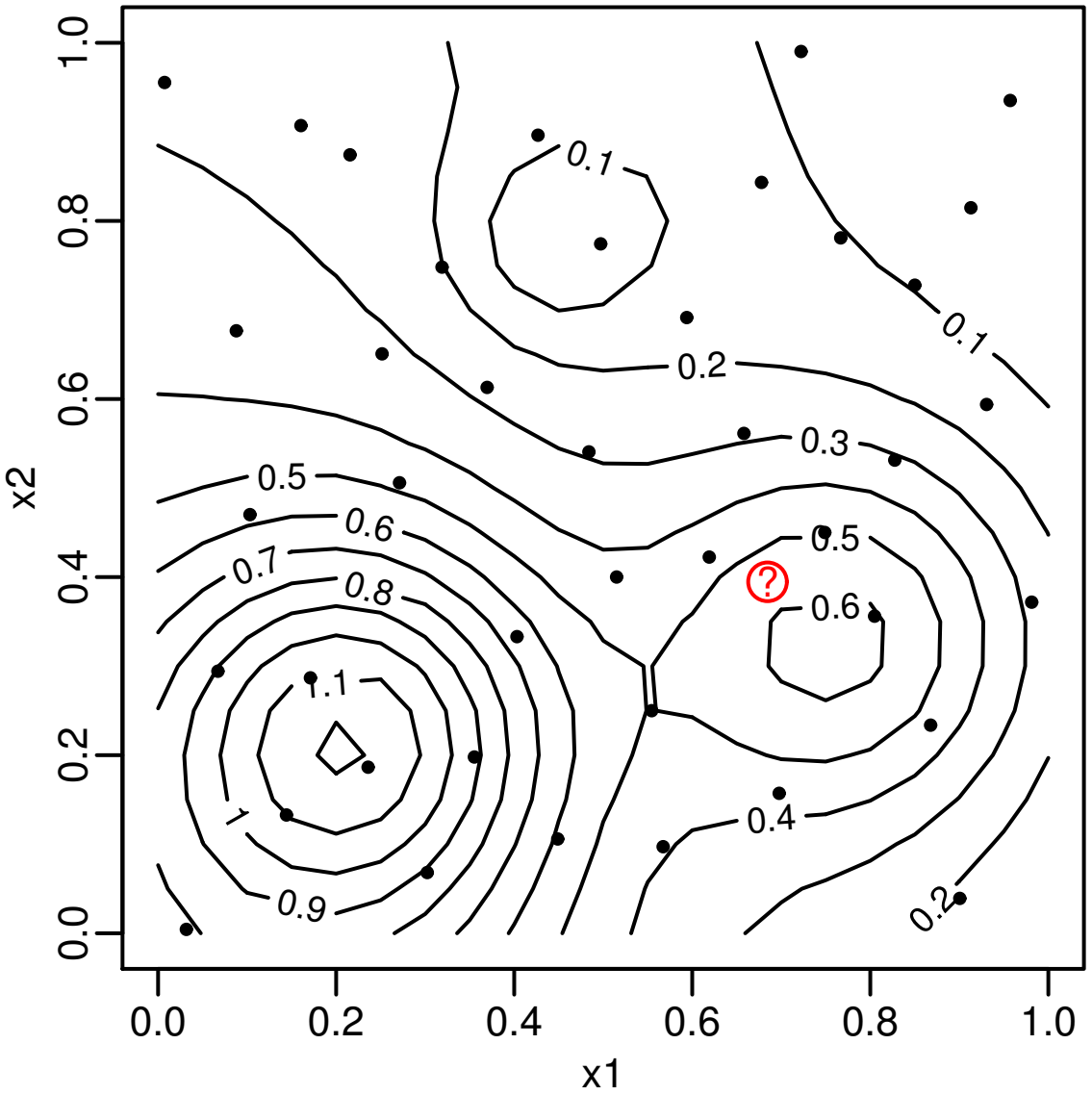
	Normalized	
Design	RMSPE	Max Error
Sparse grid	0.068	0.099
Maximin	0.047	0.110

Local Approximate Gaussian Processes (laGP, Gramacy and Apley, 2015; Gramacy, 2016)

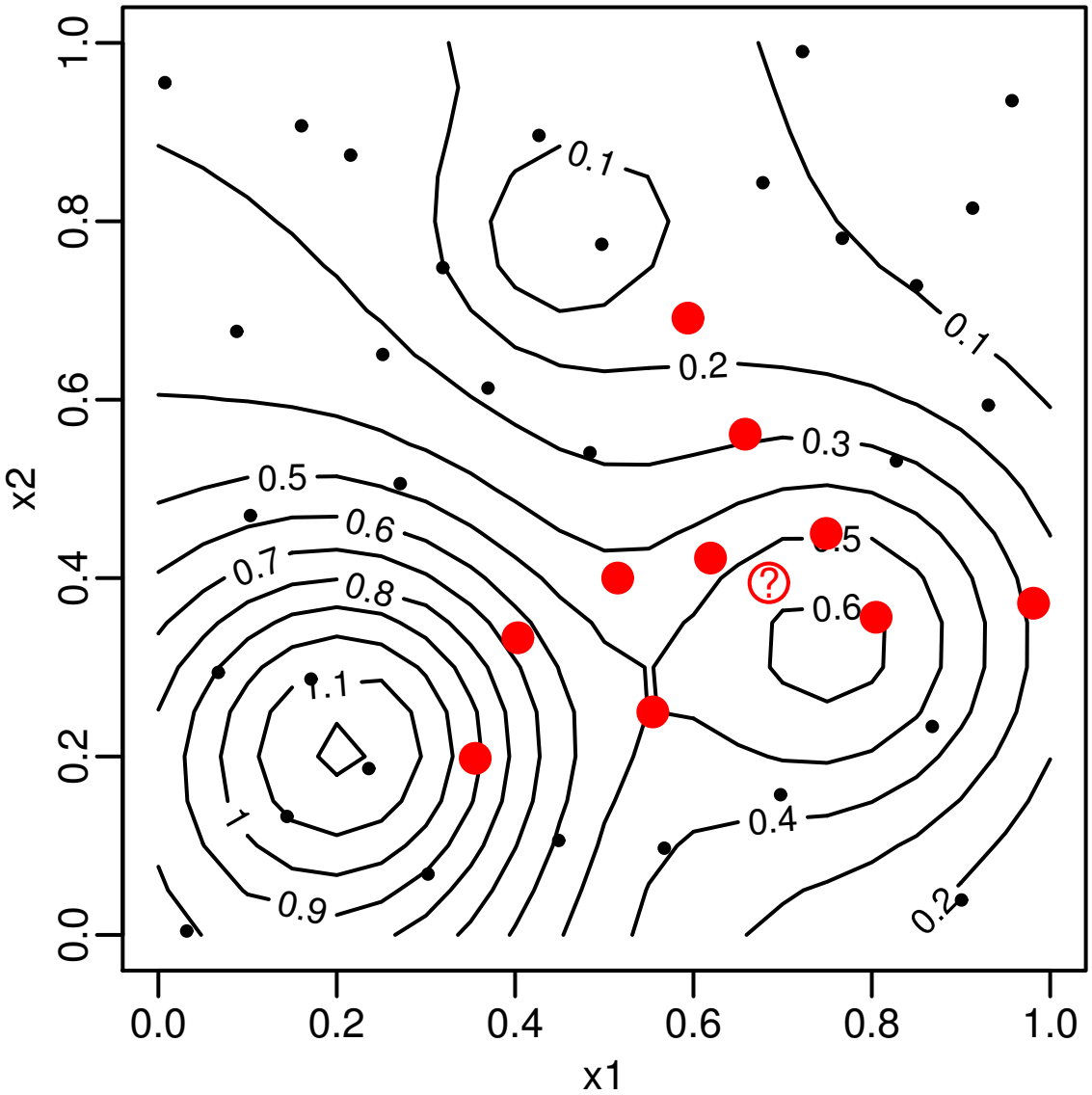
for each test point **do**
  Find $n_0 < n$ training neighbours of the test point
  Fit GP using only the n_0 neigbours of the test point
  Predict the test point using the GP
end for

- $O(n^3)$ training computation becomes $O(n_0^3)$, i.e., $(n/n_0)^3$ speed up
- Has to be repeated for each prediction

Franke's Function: Training Data and a Test Point



Franke's Function: 10 Local Training Points



Franke's Function: Prediction Accuracy

(1000 test points from a random Latin hypercube)

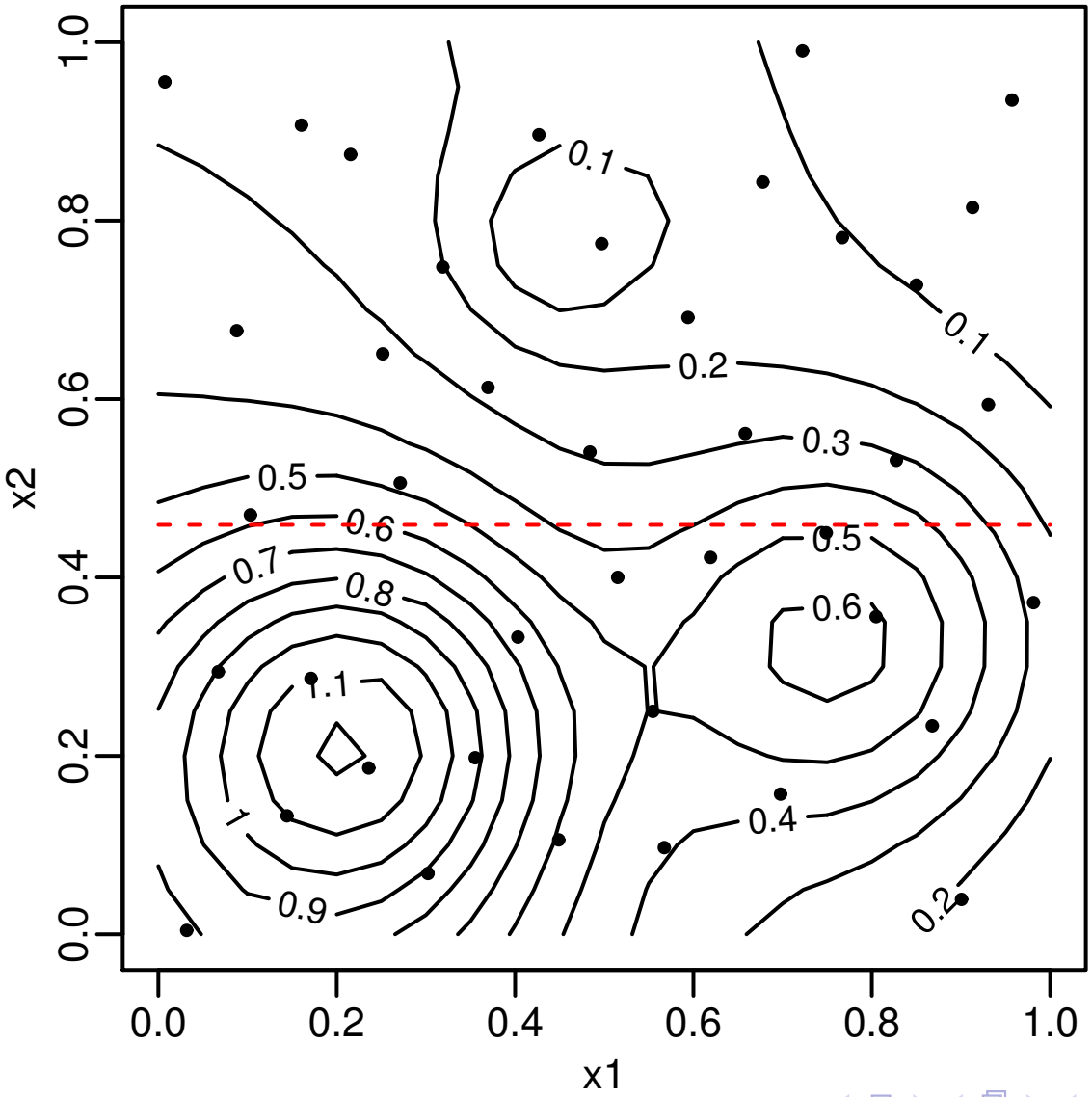
	Normalized	
Design	RMSPE	Max Error
Sparse grid	0.068	0.099
Maximin	0.047	0.110
laGP	0.061	0.127

Treed GPs (tgp, Gramacy and Lee, 2008)

- Partition the input space with a binary tree
- For each leaf (terminal node) fit a GP using the leaf's data
- Actually builds many trees and averages them for prediction

Franke's Function: Treed GP

Tree with 2 leaves: $x_2 \leq 0.44$ and $x_2 > 0.44$



Franke's Function: Prediction Accuracy

(1000 test points from a random Latin hypercube)

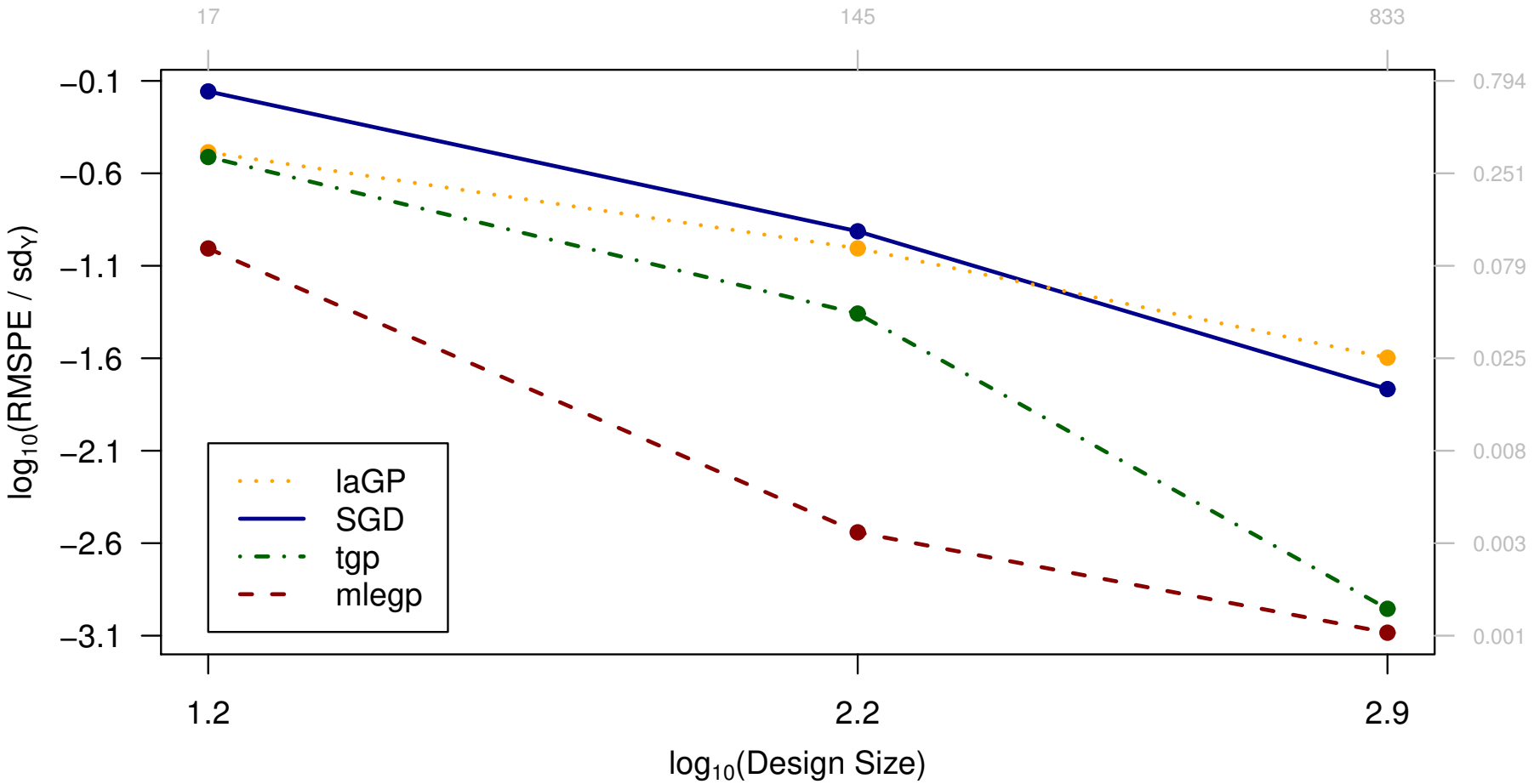
	Normalized	
Design	RMSPE	Max Error
Sparse grid	0.068	0.099
Maximin	0.047	0.110
laGP	0.061	0.127
Treed GP	0.259	0.425

Bigger n

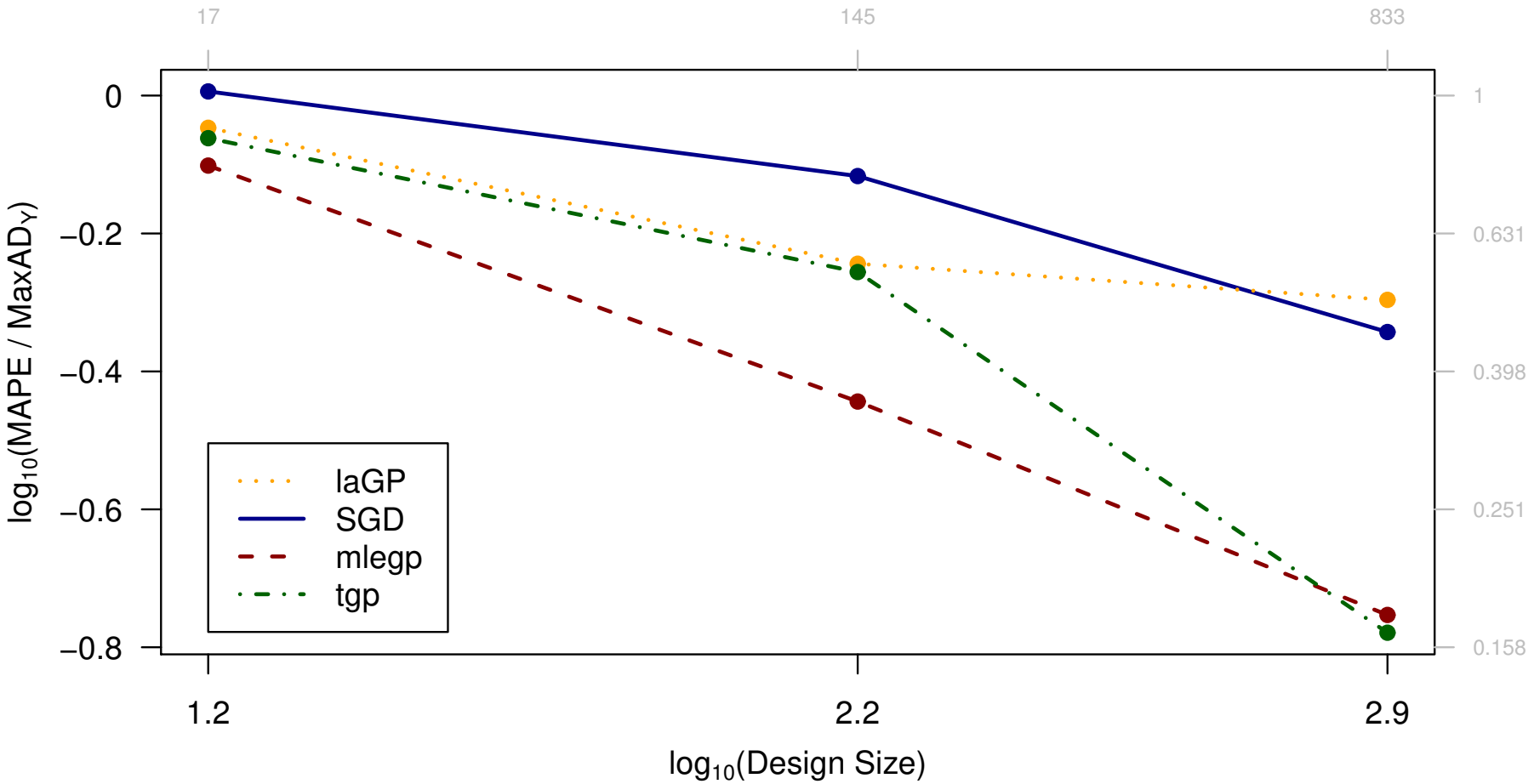
Two functions $y(\mathbf{x})$ with 8-dimensional $\mathbf{x}$

- Borehole function: easy to predict
- Corner peak function: difficult to predict (increases rapidly at the origin)

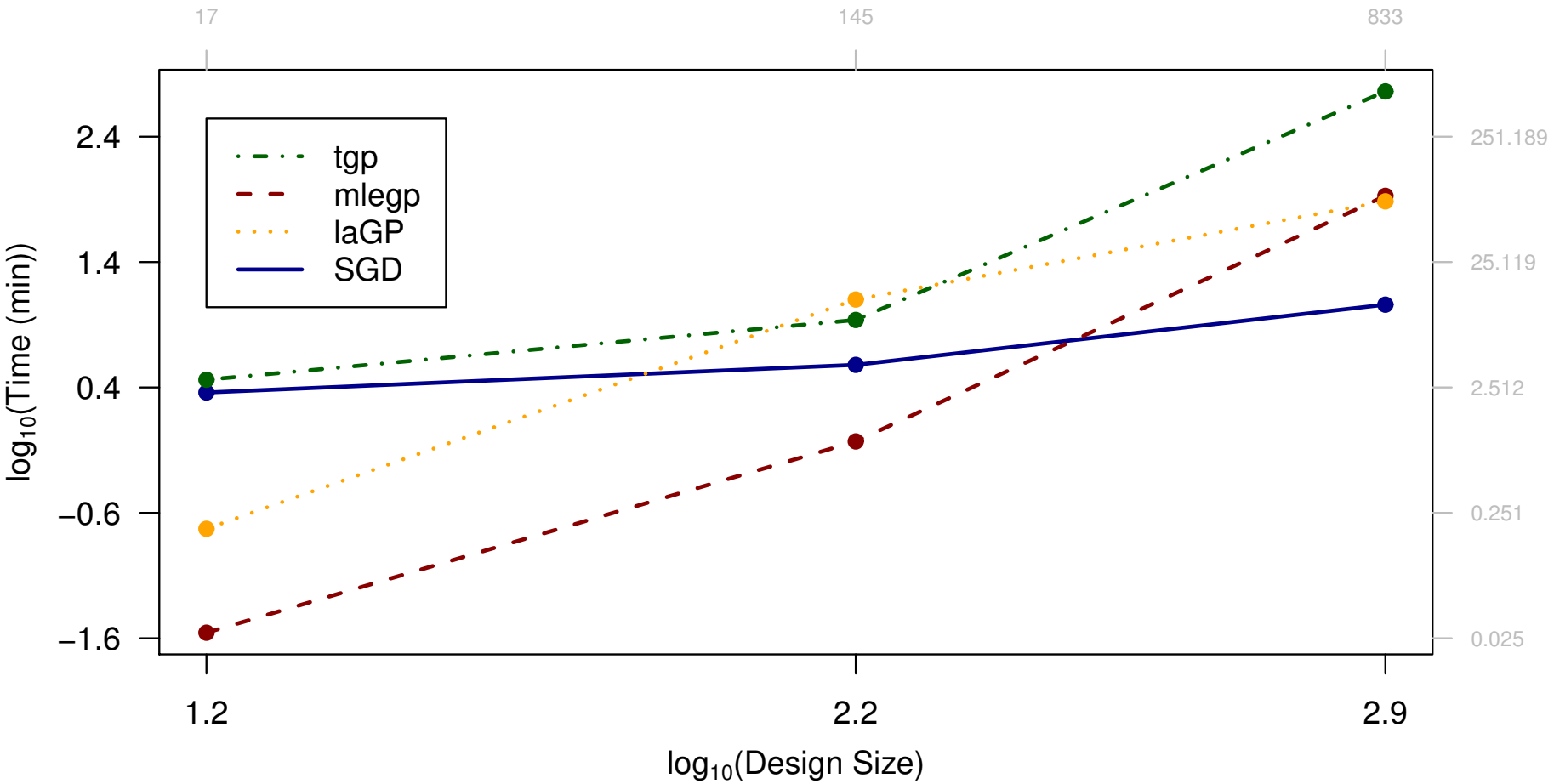
Borehole: Normalized RMS Prediction Error Versus n



Corner Peak: Normalized Max Absolute Error Versus n



Corner Peak: Computing Time Versus n



Other Methods

- Bayesian local kriging (Pronzato and Rendas, 2017): dynamically weighted combination of local GPs

- Compactly supported correlated functions (Kaufman, Bingham, Habib, Heitmann, and Frieman, 2011): induce sparse correlation matrix

Conclusion

Has Gaussian process computation been tamed?

A No

B Yes

C I don't know

D Nobody knows

E 3 hours to dinner

Unknowns

- Implementation of "standard" analysis is difficult for some local methods
- Any one of these methods is not one method:
  - How to choose a sparse grid?
  - How many points in a local region?
- Domain of practical problems?
  - Do these methods allow large enough n for a useful statistical model of a complex function?
  - Remember, we have to run the computer model n times

THANK YOU!

References

Gramacy, R. B. (2016), "laGP: Large-Scale Spatial Modeling via Local Approximate Gaussian Processes in R," *Journal of Statistical Software*, 72, 1–46.

Gramacy, R. B. and Apley, D. W. (2015), "Local Gaussian Process Approximation for Large Computer Experiments," *Journal of Computational and Graphical Statistics*, 24, 561–578.

Gramacy, R. B. and Lee, H. K. H. (2008), "Bayesian Treed Gaussian Process Models With an Application to Computer Modeling," *Journal of the American Statistical Association*, 103, 1119–1130.

Kaufman, C. G., Bingham, D., Habib, S., Heitmann, K., and Frieman, J. A. (2011), "Efficient Emulators of Computer Experiments Using Compactly Supported Correlation Functions, With an Application to Cosmology," *Annals of Applied Statistics*, 5, 2470—2492.

Plumlee, M. (2014), "Fast Prediction of Deterministic Functions Using Sparse Grid Experimental Designs," *Journal of the American Statistical Association*, 109, 1581–1591.

Pronzato, L. and Rendas, M. J. (2017), "Bayesian Local Kriging," *Technometrics*, 59, 293–304.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989), "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409–423.