Modifying MED for Model Selection

Kristyn Pantoja

11/06/2019

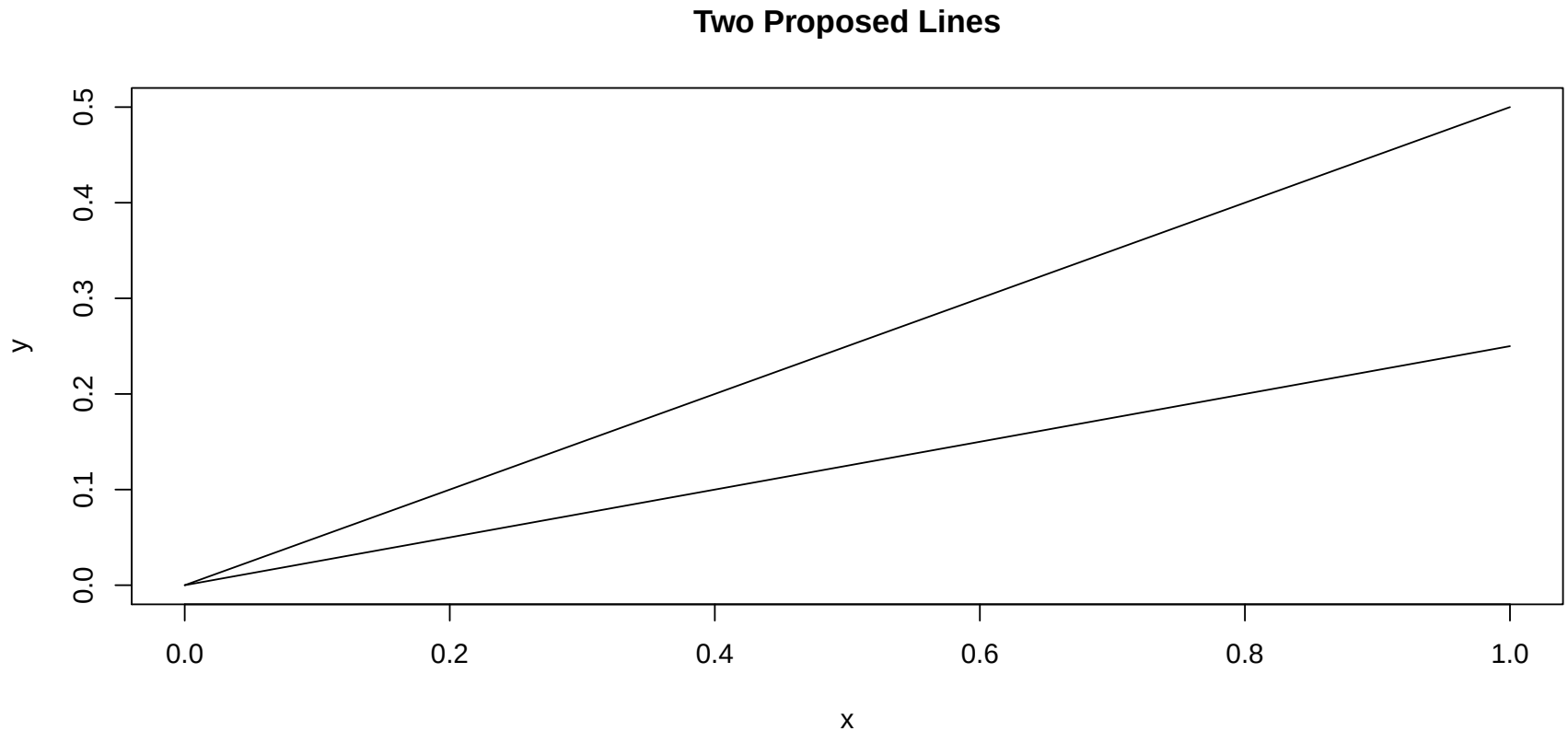
Gaussian Process Application

Gaussian vs. Matern, $l = 0.5$

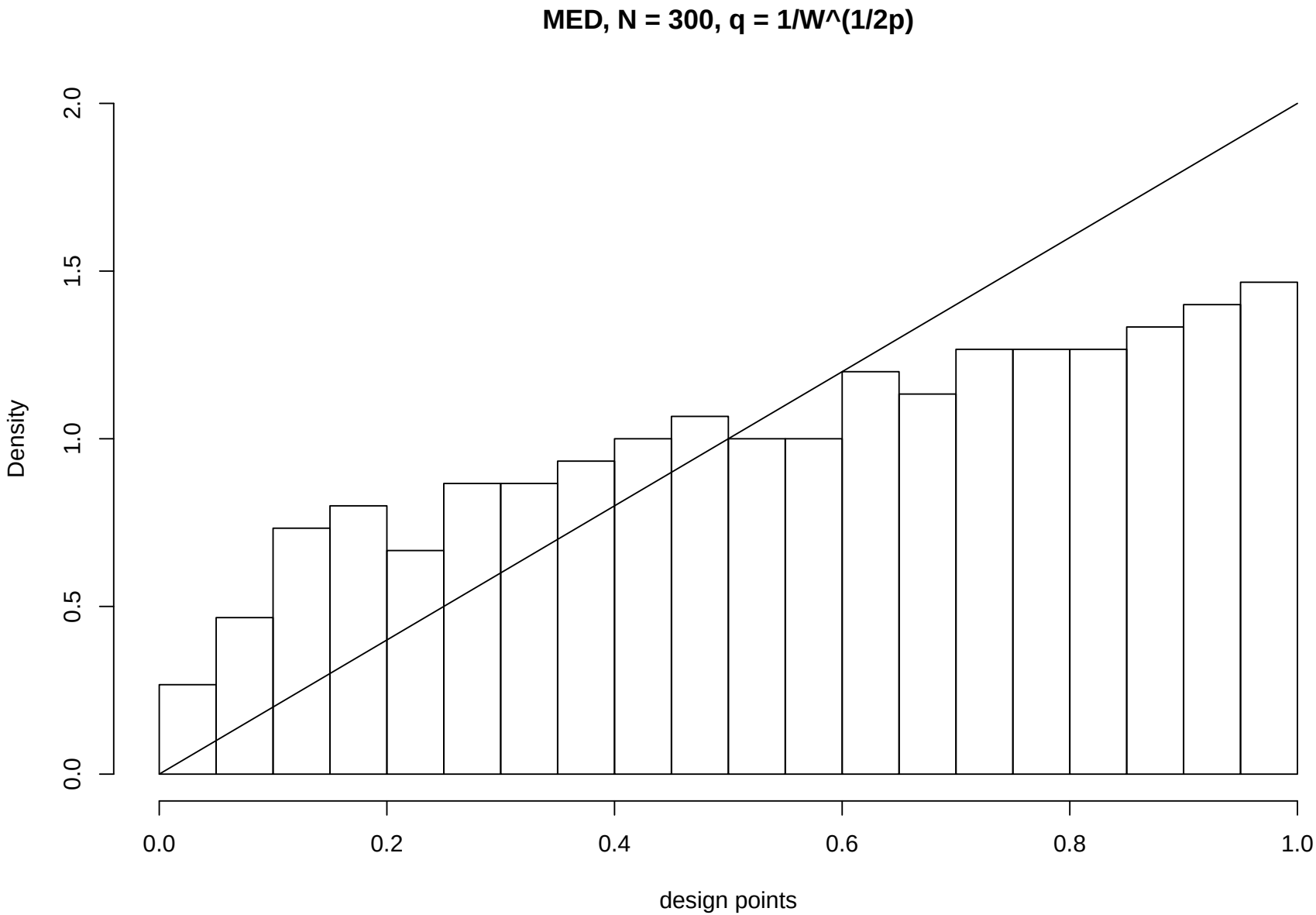
Gaussian vs. Matern, $l = 0.1$

Gaussian vs. Periodic Kernel

Original Motivating Example



Getting an idea of limiting distribution (N = 300, k = 4)

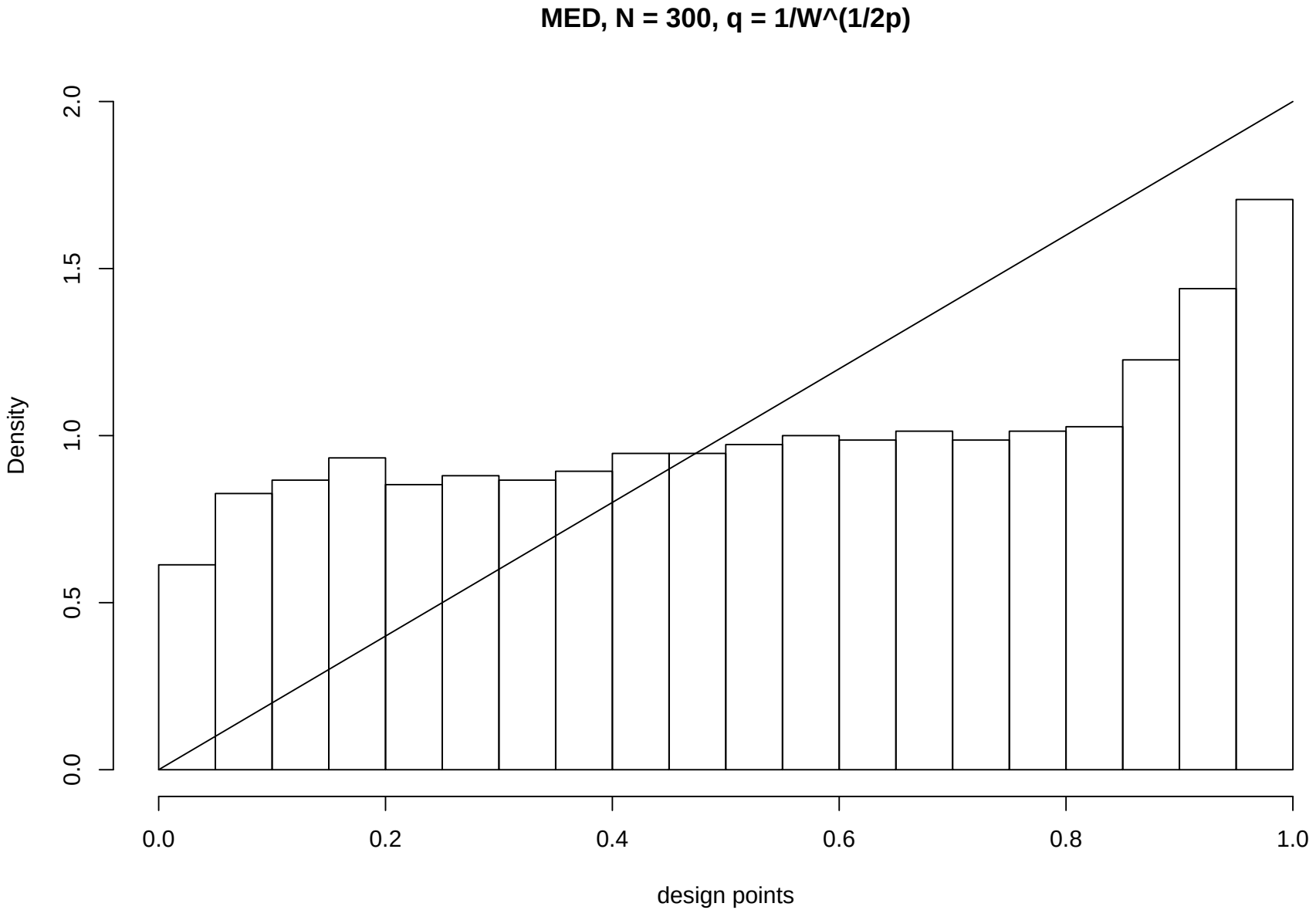


When N is large

```
mean_beta0 = c(0, 1 / 2) # slope of null model
mean_beta1 = c(0, 1 / 4) # slope of alternative model
var_beta0 = diag(c(0.005, 0.005)); var_beta1 = var_beta0 #
var_e = 0.025 # variance on error
xmin = 0
xmax = 1
f0 = function(x) mean_beta0[1] + mean_beta0[2] * x # null
f1 = function(x) mean_beta1[1] + mean_beta1[2] * x # alter
N = 300
type = c(2, 2)
p = 2
# for fast algorithm:
S = 5
# for one-at-a-time algorithm:
numCandidates = 10^4
k = 4
```

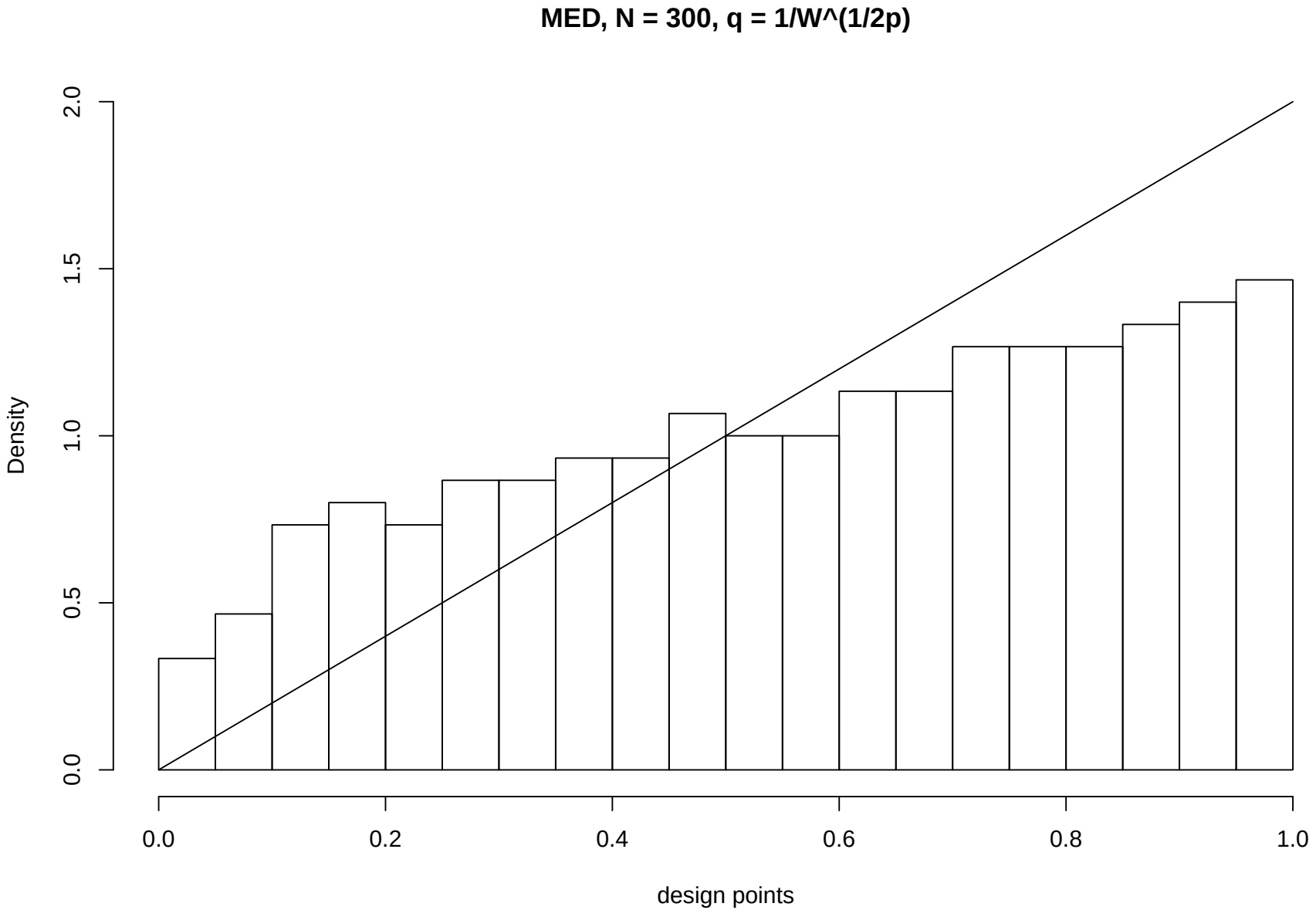
Fast (N = 300, S = 5)

The other algorithm isn't much better, somehow. Maybe it's more suited for dealing with higher dimensions.

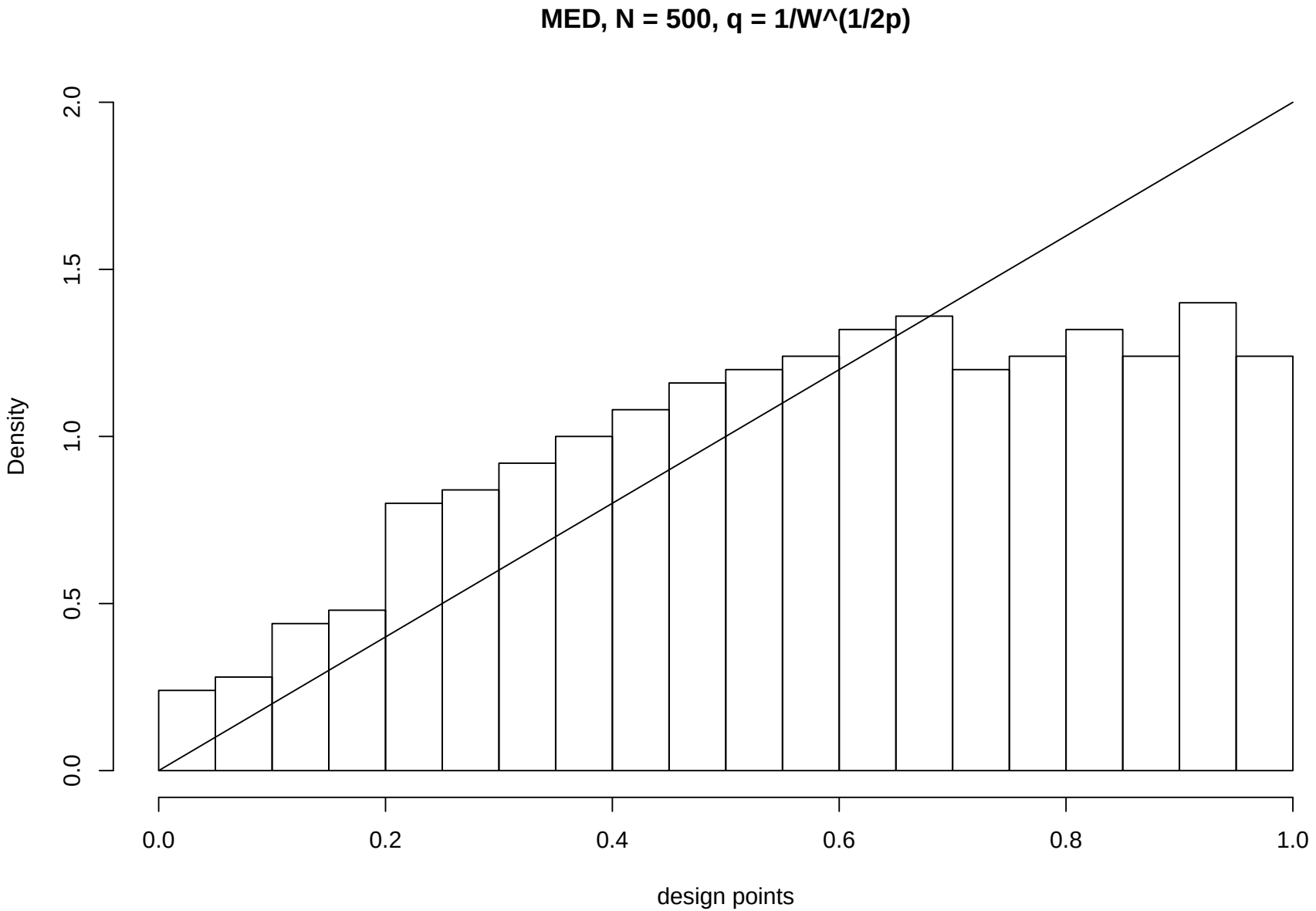


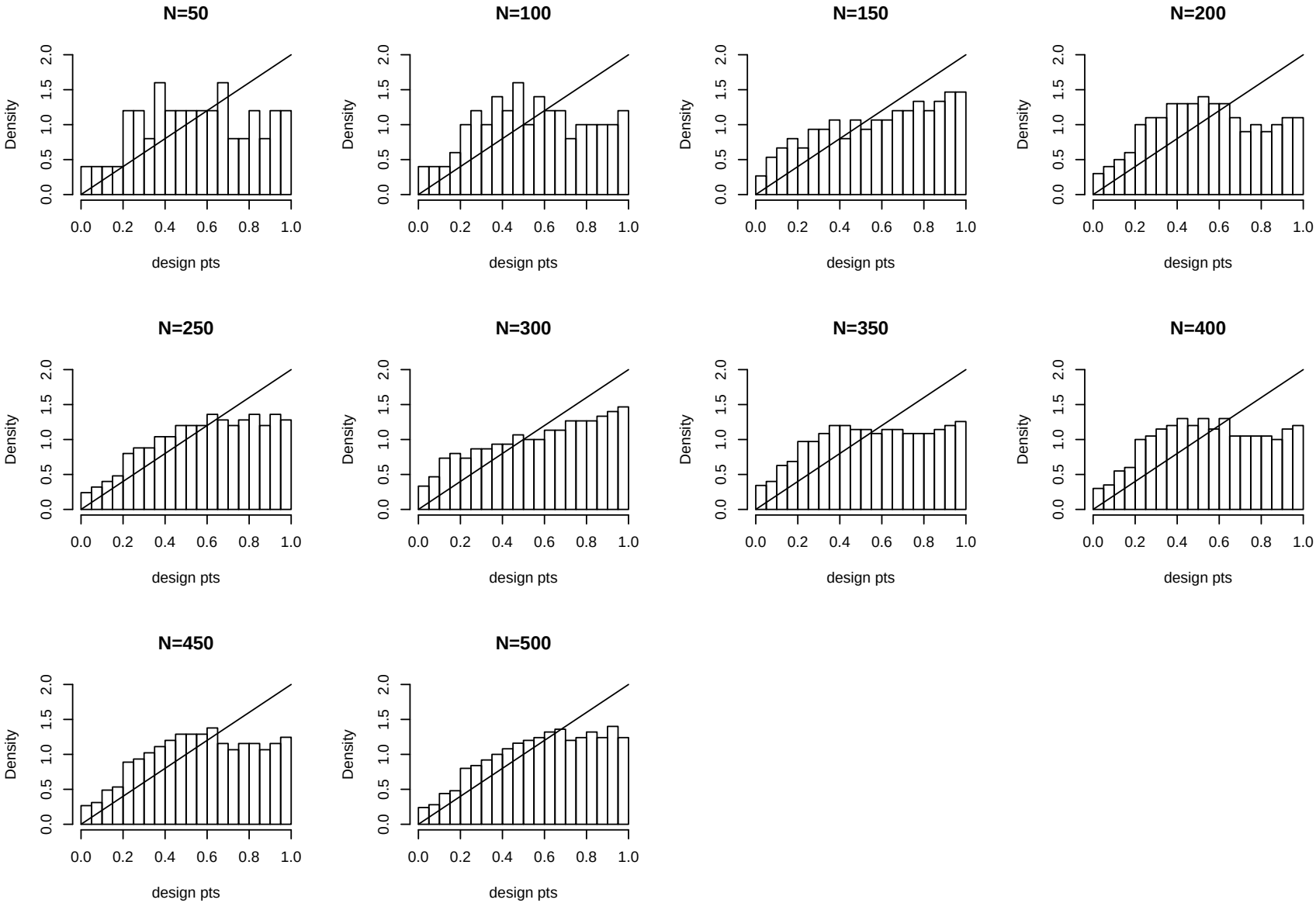
Greedy (N = 300, k = 4, numCandidates = 10^4)

The original method, for baseline comparison:



Greedy (N = 500, k = 4, numCandidates = 10^4)




N=50
N=100
N=150
N=200
N=250
N=300
N=350
N=400
N=450
N=500
Density
design pts
0.0
0.5
1.0
1.5
2.0
0.2
0.4
0.6
0.8

What is the algorithm trying to do?

It looks like the algorithm is trying to balance filling in the space between the high-importance region near 1 and the lesser important regions (closer to 0), but perhaps overcompensates that part and ending up with more points in the middle than necessary, then trying to re-balance that out somehow.

Look at TPE at certain points

```
tail(mmed_lim_N500_k4, 100)
```

```
##   [1] 0.7748775 0.7155716 0.5640564 0.6707671 0.3805381 0.6231623 0.5862586
##   [8] 0.5314531 0.9015902 0.8269827 0.6017602 0.4978498 0.4702470 0.9686969
##  [15] 0.6381638 0.7695770 0.7100710 0.8966897 0.4190419 0.8218822 0.9638964
##  [22] 0.5576558 0.7642764 0.5248525 0.5955596 0.9591959 0.7045705 0.8166817
##  [29] 0.6321632 0.4910491 0.8869887 0.5146515 0.9543954 0.7589759 0.4631463
##  [36] 0.8114811 0.5511551 0.6989699 0.6736674 0.5893589 0.9496950 0.6261626
##  [43] 0.8820882 0.7535754 0.8062806 0.9448945 0.4841484 0.6933693 0.8771877
##  [50] 0.6678668 0.5831583 0.7482748 0.6201620 0.8010801 0.9401940 0.8722872
##  [57] 0.9662966 0.6877688 0.9353935 0.6619662 0.7958796 0.7428743 0.8672867
##  [64] 0.8192819 0.5114511 0.6140614 0.5768577 0.9305931 0.9567957 0.8622862
##  [71] 0.7017702 0.6821682 0.8140814 0.7906791 0.6560656 0.7374737 0.9257926
##  [78] 0.7562756 0.8572857 0.8088809 0.9209921 0.9472947 0.6961696 0.8522852
##  [85] 0.6501650 0.8036804 0.7265727 0.8472847 0.6905691 0.9377938 0.7455746
##  [92] 0.9112911 0.8697870 0.7984798 0.9965997 0.8422842 0.9329933 0.6849685
##  [99] 0.8647865 0.7932793
```

we can look closer at how the point after

```
mmed_lim_N500_k4[402]
```

```
## [1] 0.7155716
```

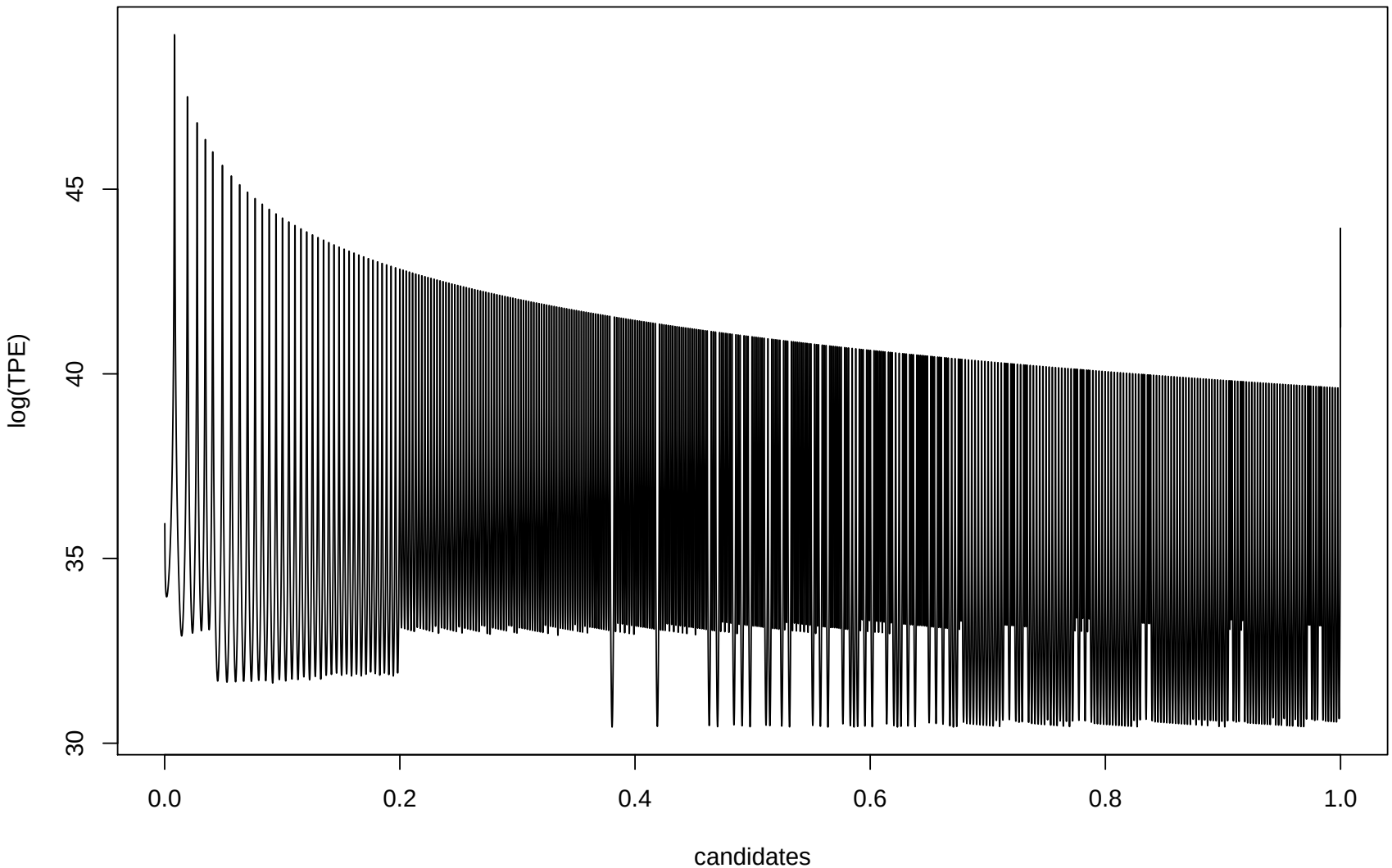
i.e.

```
mmed_lim_N500_k4[403]
```

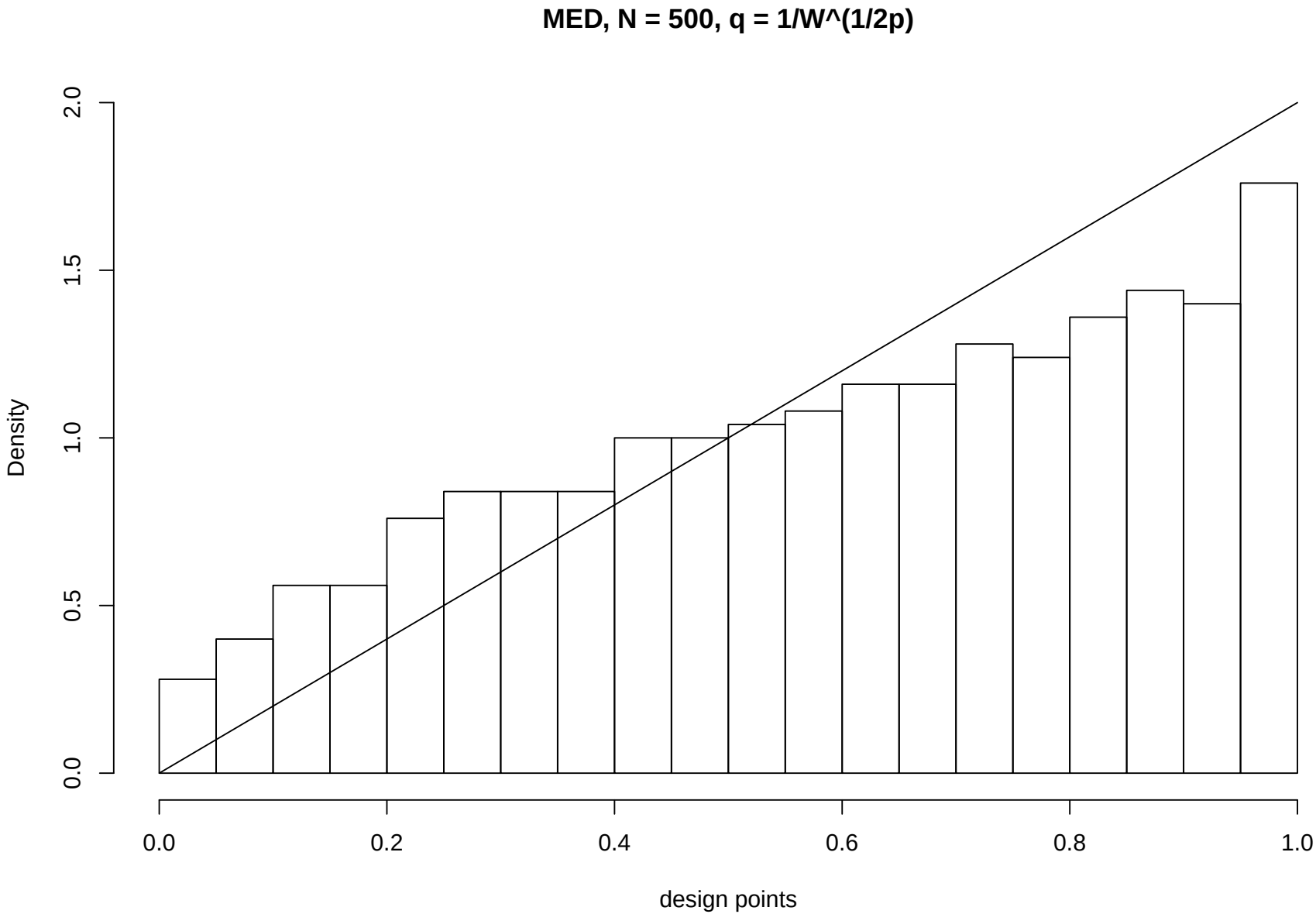
```
## [1] 0.5640564
```

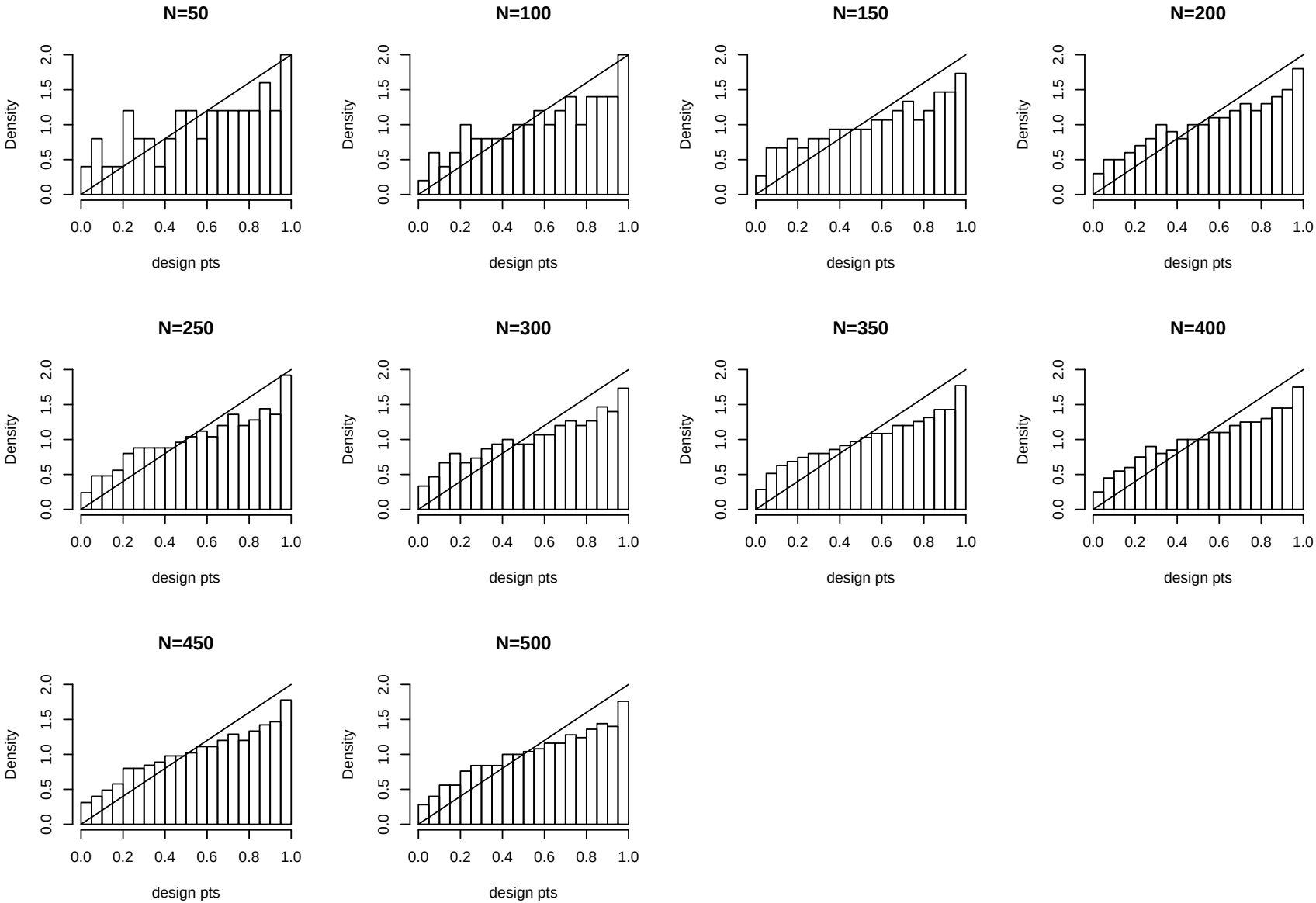
is chosen by calculating the TPE for $\mathbf{x}_i$ for $i = 1 : 402$

TPE



Greedy (N = 500, k = 1, numCandidates = 10^4)




N=50
N=100
N=150
N=200
N=250
N=300
N=350
N=400
N=450
N=500
Density
design pts
0.0
0.5
1.0
1.5
2.0
0.2
0.4
0.6
0.8

What is the algorithm trying to do?

Here, the greedy algorithm is simpler - no raising the terms of the summation in TPE to any power (i.e. k = 1)

Look at TPE at certain points

```
tail(mmed_lim_N500_k1, 100)
```

```
##   [1] 0.84078408 0.98769877 0.64096410 0.33413341 0.73317332 0.94569457
##   [7] 0.55645565 0.38193819 0.89128913 0.45524552 0.69576958 0.50505051
##  [13] 0.92729273 0.03350335 0.24802480 0.65876588 0.78547855 0.96349635
##  [19] 0.59425943 0.84888489 0.43414341 0.21852185 0.72057206 0.34953495
##  [25] 0.86818682 0.57825783 0.97579758 0.82698270 0.51535154 0.16231623
##  [31] 0.61336134 0.30393039 0.73957396 0.41744174 0.99119912 0.92019202
##  [37] 0.46274627 0.88608861 0.81028103 0.36403640 0.70827083 0.63526353
##  [43] 0.93219322 0.54245425 0.77957796 0.20772077 0.95689569 0.39303930
##  [49] 0.04910491 0.68306831 0.75677568 0.48764876 0.99769977 0.14011401
##  [55] 0.87358736 0.25762576 0.79367937 0.85728573 0.56265627 0.32173217
##  [61] 0.72687269 0.89598960 0.62896290 0.98009801 0.29012901 0.83268327
##  [67] 0.48044804 0.68936894 0.91039104 0.23382338 0.60696070 0.37093709
##  [73] 0.52225223 0.17901790 0.76817682 0.95239524 0.44444444 0.81598160
##  [79] 0.10831083 0.64996500 0.40384038 0.54945495 0.86268627 0.27612761
##  [85] 0.94339434 0.67336734 0.71437144 0.31313131 0.14911491 0.83808381
##  [91] 0.42774277 0.80468047 0.62246225 0.50165017 0.75107511 0.58495850
##  [97] 0.96569657 0.06220622 0.70217022 0.47324732
```

we can look closer at how the point after

```
mmed_lim_N500_k1[488]
```

```
## [1] 0.3131313
```

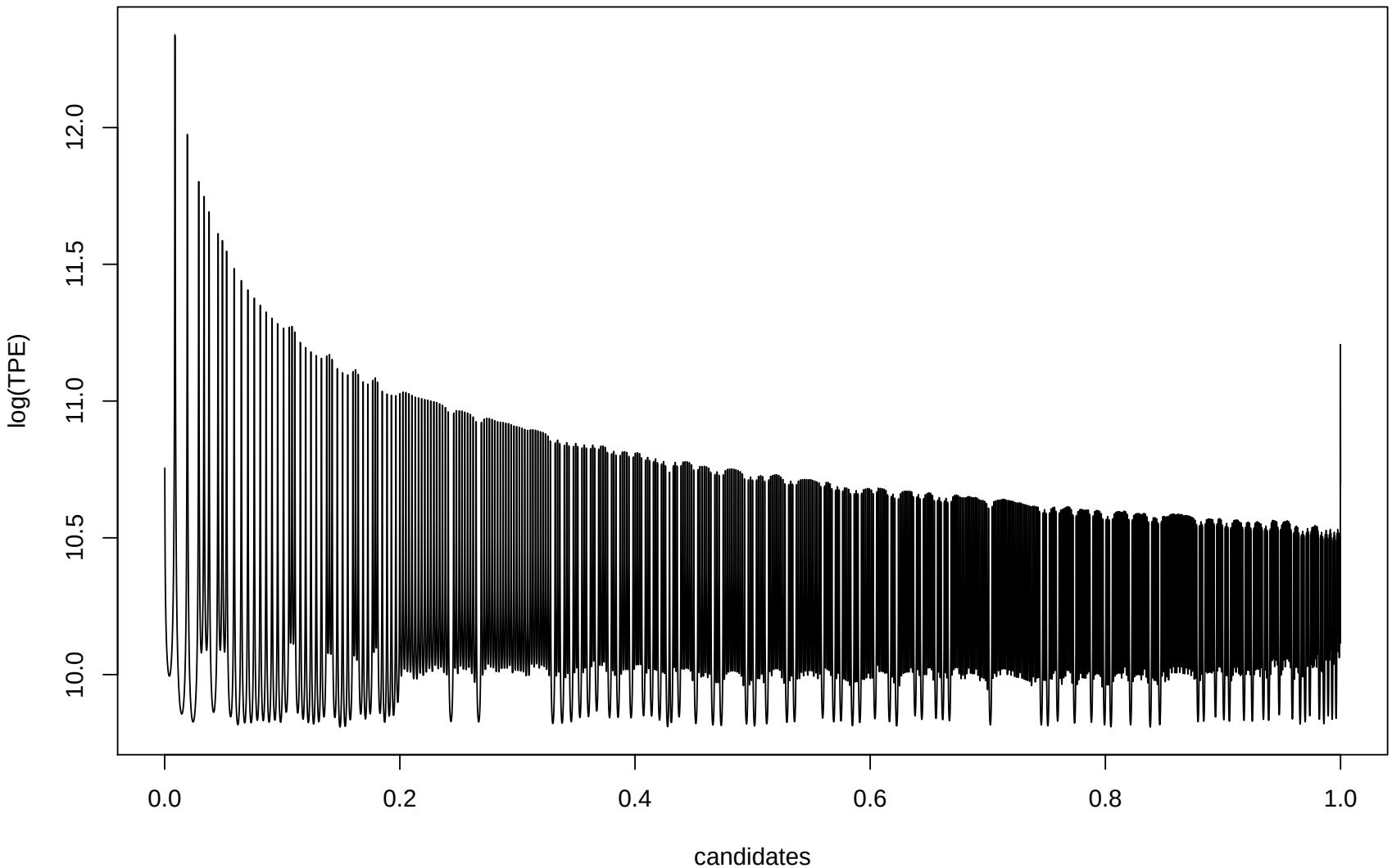
i.e.

```
mmed_lim_N500_k1[489]
```

```
## [1] 0.1491149
```

is chosen by calculating the TPE for $\mathbf{x}_i$ for $i = 1 : 488$

TPE



Gaussian Process Application

Applying MED to Gaussian Process Model Selection

- When there are two Gaussian Process Models that can be used to estimate a function, e.g. Matern vs. Squared Exponential covariance functions[1]
  - Squared Exponential: infinitely differentiable, standard choice
  - Matern: more reasonable smoothness assumptions

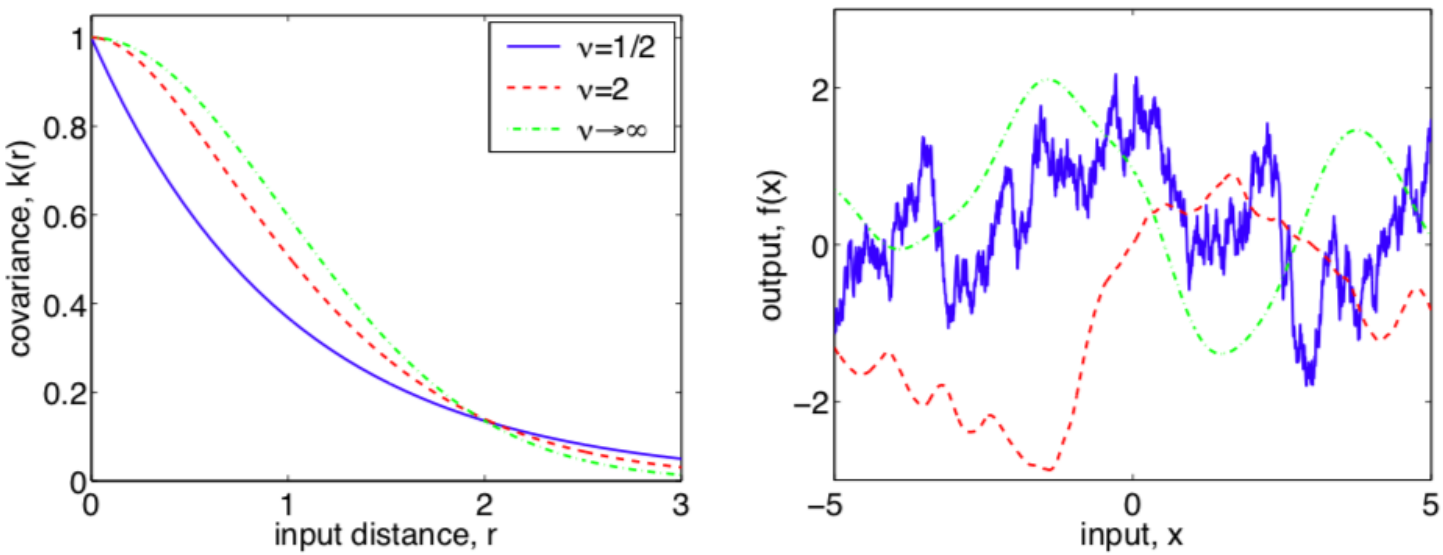


Figure 4.1: Panel (a): covariance functions, and (b): random functions drawn from Gaussian processes with Matérn covariance functions, eq. (4.14), for different values of ν, with $\ell = 1$. The sample functions on the right were obtained using a discretization of the x-axis of 2000 equally-spaced points.

[5] "Gaussian Processes for Machine Learning" Rasmussen et. al. 2005

Applying MED to Gaussian Process Model Selection

- Goal: Choose a design that will distinguish the two gaussian process models.
- Distinguishing functions vs. distributions over functions:
  - For regression models, we use $f_D(\mathbf{x}) = \text{Wasserstein}(\phi_{0,\mathbf{x}}, \phi_{1,\mathbf{x}})$. What is the distance function now? What are $\phi_{0,\mathbf{x}}, \phi_{0,\mathbf{x}}$?
  - Key Question: Do we need to consider the predictive distribution for each GP model?
    - Doing so would give us an option for $\phi_{0,\mathbf{x}}, \phi_{0,\mathbf{x}}$.
    - However, we will need some initial data.

One-at-a-Time Algorithm for GP

Suppose you have training data $\mathcal{T} = \{(\mathbf{x}_k, y_k)\}_{k=1}^{N_1}$.

1. Obtain candidate set C
2. Initialize the new set of design points $\mathbf{D}$ as the candidate point $\mathbf{x}_*$ that maximizes $f_D(\mathbf{x}) = \text{Wasserstein}(\phi_{0,\mathbf{x}}, \phi_{1,\mathbf{x}})$, where, here, $\phi_{\ell,\mathbf{x}}$ is the predictive distribution $f_*|\mathbf{x}_*, X, f \sim N(k_*^T(K+\tau^2 I)^{-1}Y, k(\mathbf{x},\mathbf{x}) - k_*^T(K+\tau^2 I)^{-1}k_*)$, where $k_* = k(\mathbf{x}, X), K = K(X, X)$, and k and K are determined by the hypothesis ℓ.
3. For subsequent design points, choose:

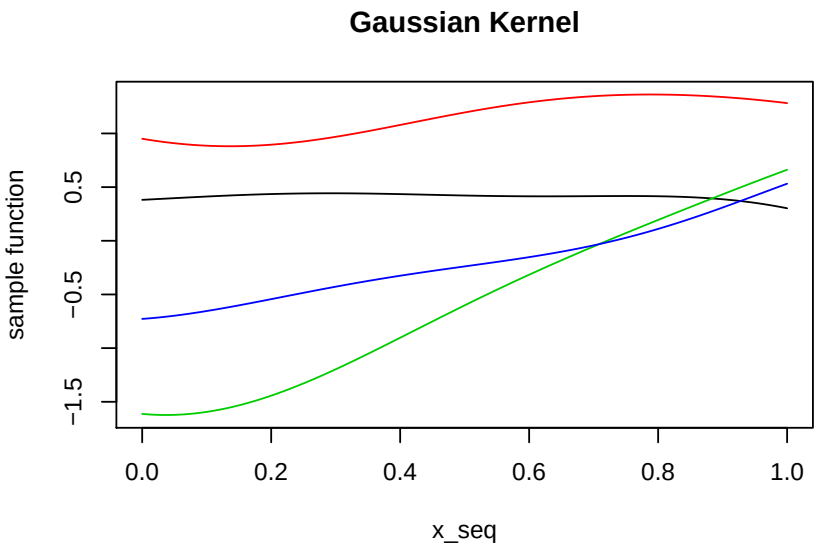
$$\mathbf{x}_{j+1} = \arg\min_{\mathbf{x}\in C} \sum_{\mathbf{x}_i \in \mathbf{D}}^{j} \left(\frac{q(\mathbf{x}_i)q(\mathbf{x})}{d(\mathbf{x}_i,\mathbf{x})}\right)^k + \sum_{\mathbf{x}_i \in \mathcal{T}} \left(\frac{q(\mathbf{x}_i)q(\mathbf{x})}{d(\mathbf{x}_i,\mathbf{x})}\right)^k$$

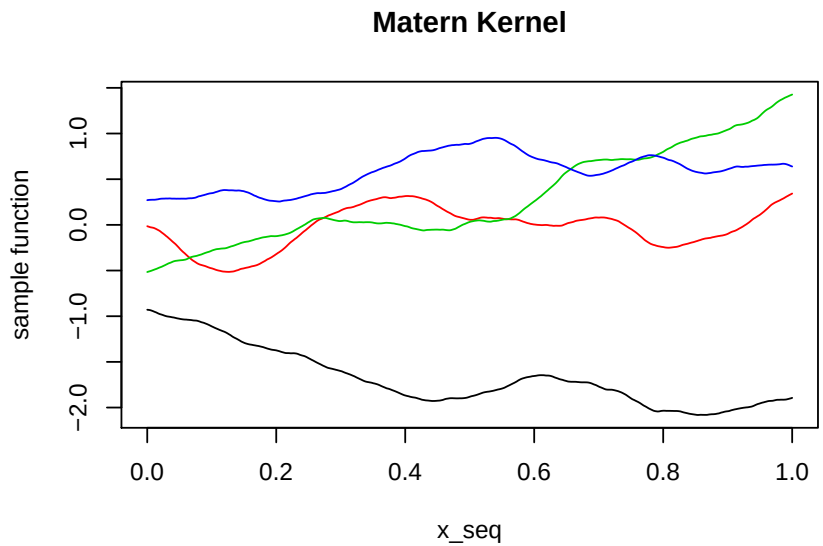
where $q = 1/f_D^{1/2p}$ and $k = 1$ for a greedy algorithm to minimize TPE.

Gaussian vs. Matern, $l = 0.5$

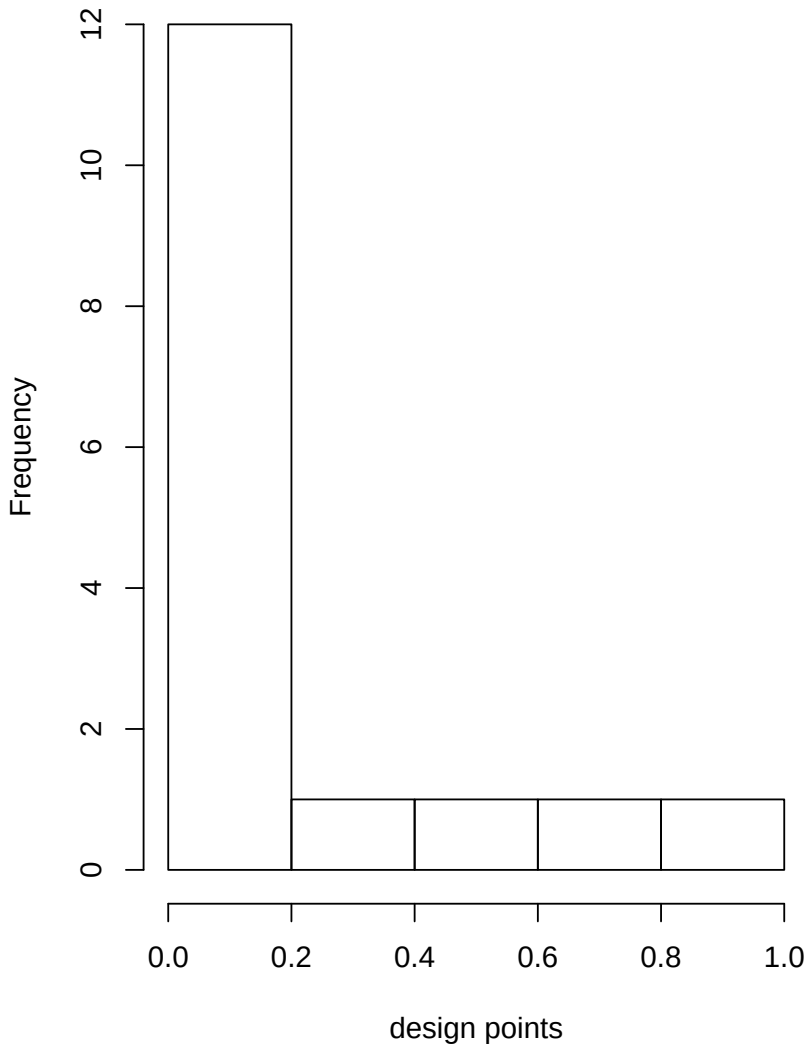
Gaussian vs. Matern with l = 0.5

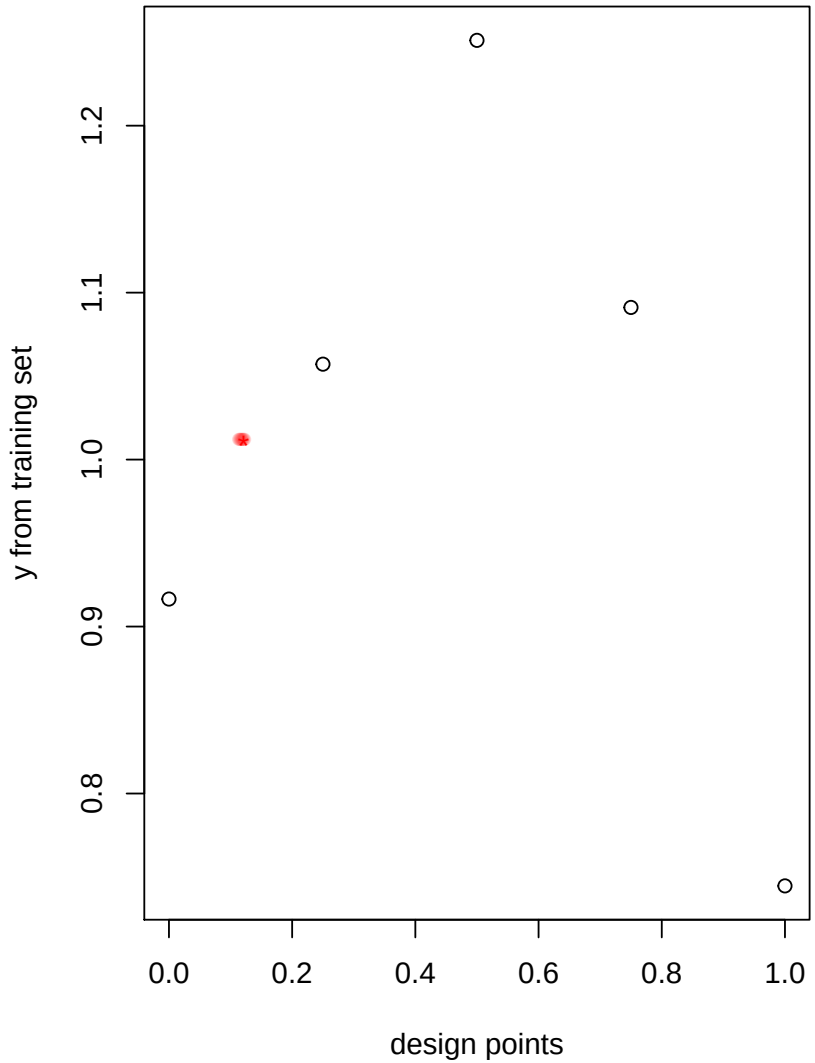
length-scale parameter $\ell = 0.5$





Including Data's Points in TPE (requires nugget term)



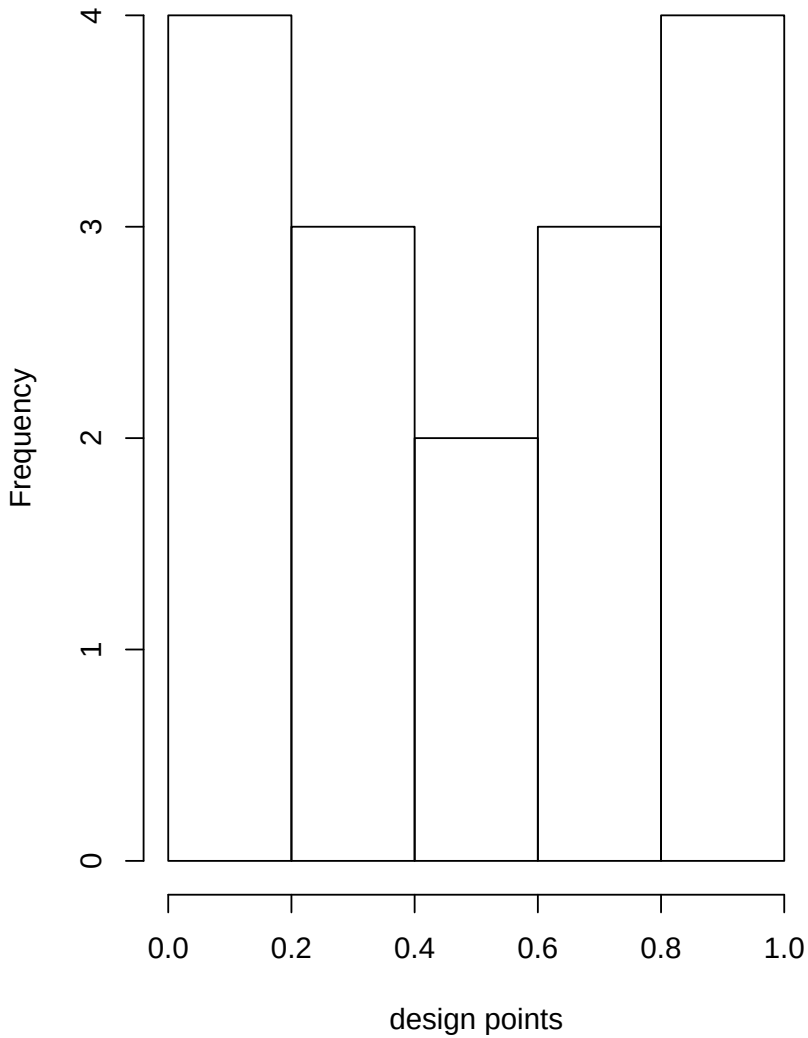


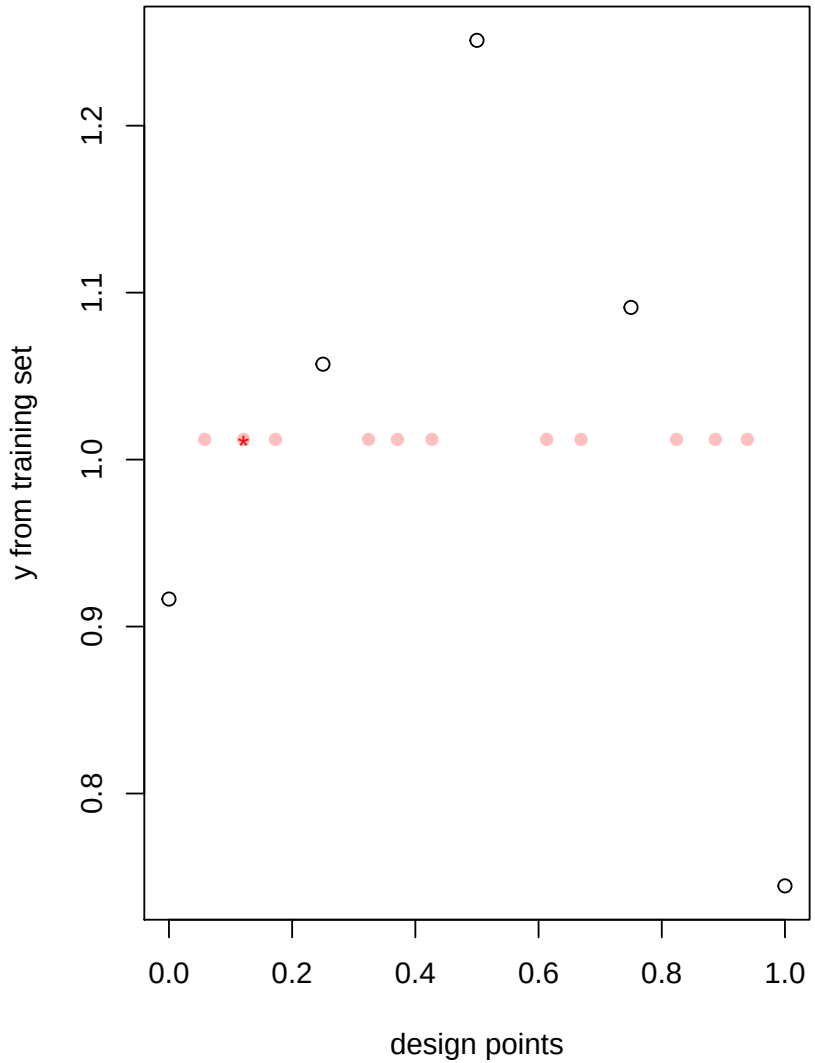
Training Points and q

- The noise parameter, i.e. nugget term, is necessary for computing $q(\mathbf{x})$ for $\mathbf{x}$ in the training set, $\mathcal{T}$:
  - if $\mathbf{x} \in \mathcal{T}$, then $\phi_{0,\mathbf{x}} = \phi_{1,\mathbf{x}}$ due to GP interpolation, and $f_D(\mathbf{x}) = 0$, which implies that $q(\mathbf{x}) = \infty$ and TPE cannot be evaluated.

Alternatively, we can leave out these points $\mathbf{x} \in \mathcal{T}$ in the evaluation of TPE, which we do next. Leaving out these points from the design can be justified by the fact that these points necessarily parameterize the function before we can begin choosing a design for model selection.

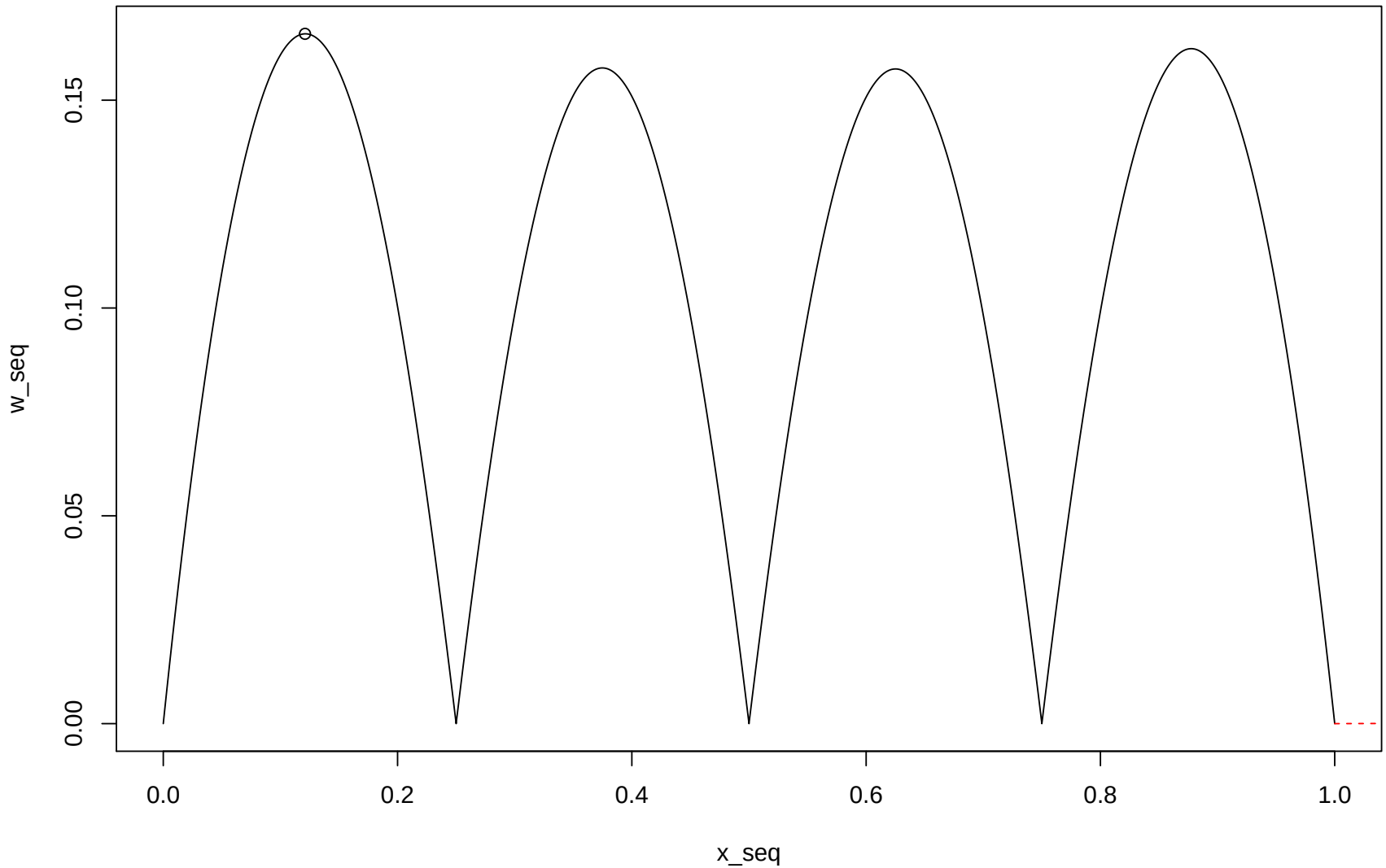
Not Including Training Pts (with no nugget term)





Wasserstein Distance between Points

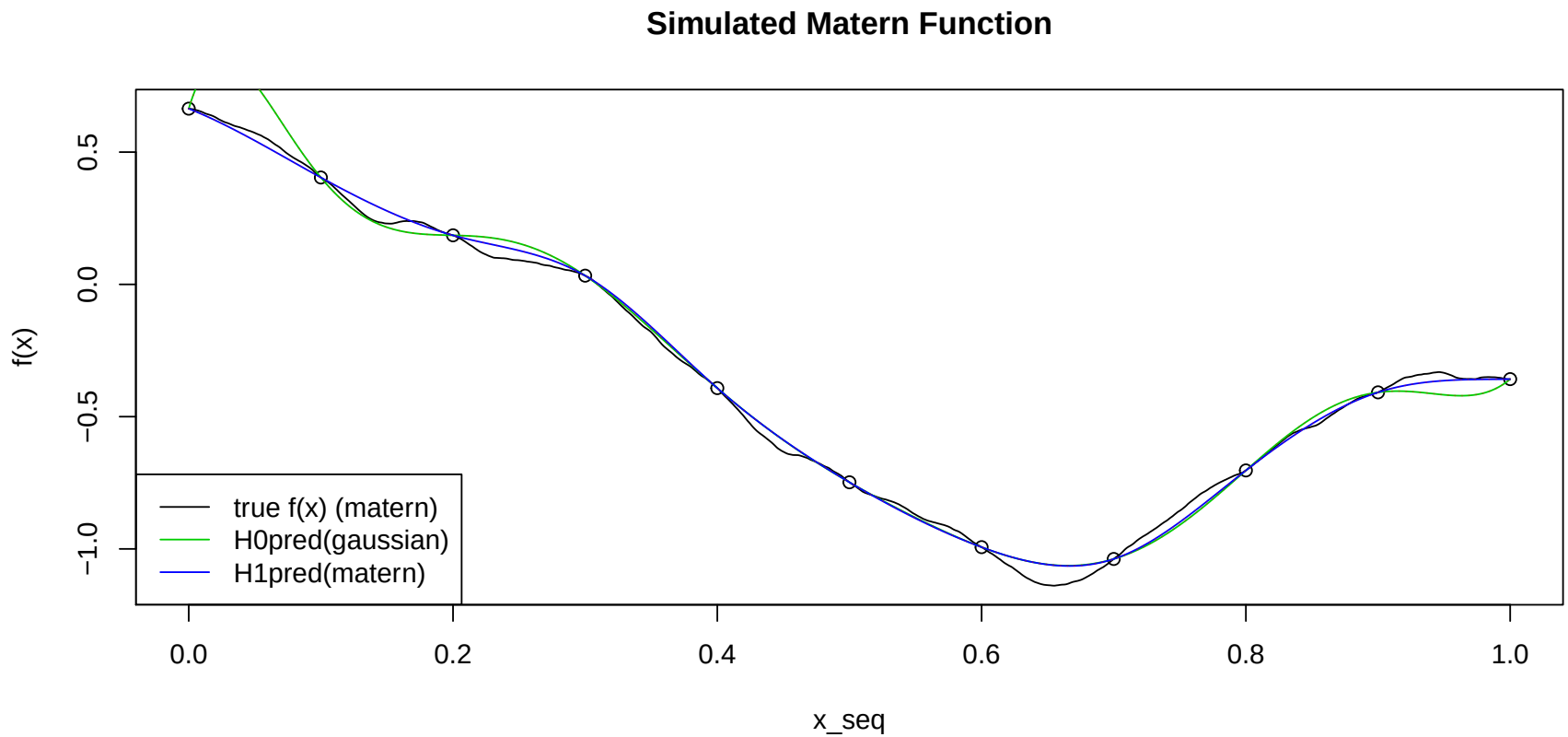
(It turns out my matern function was parameterized differently! Hopefully this behavior makes more sense.)

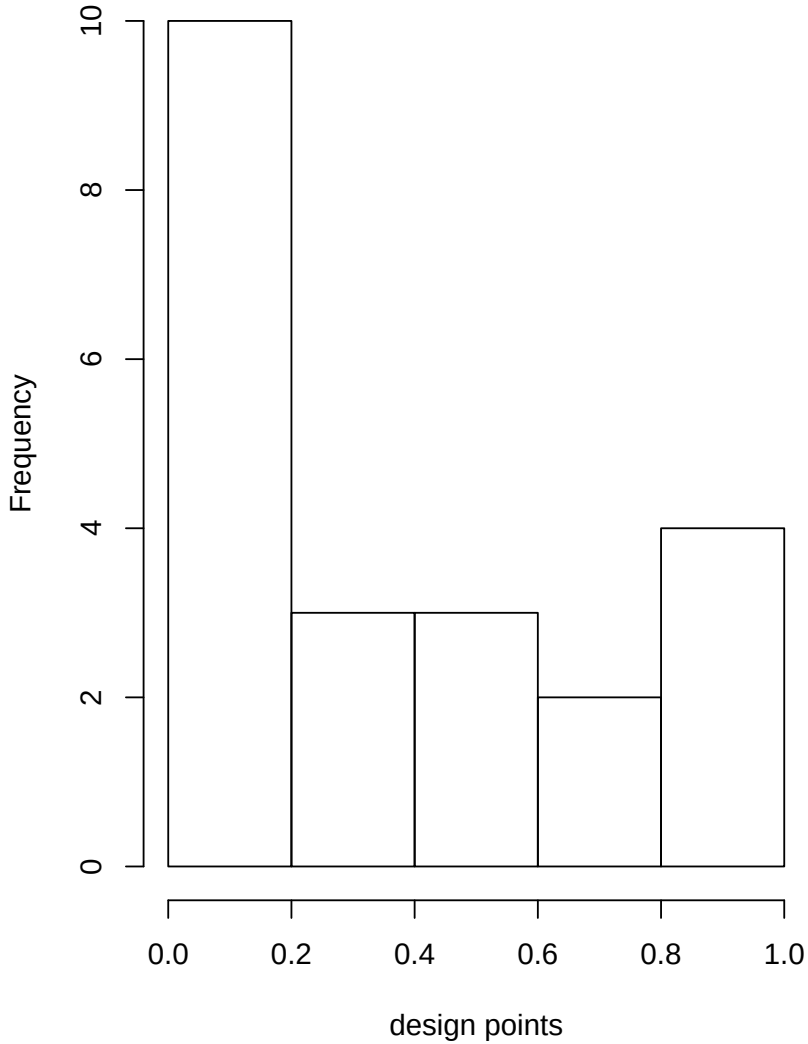


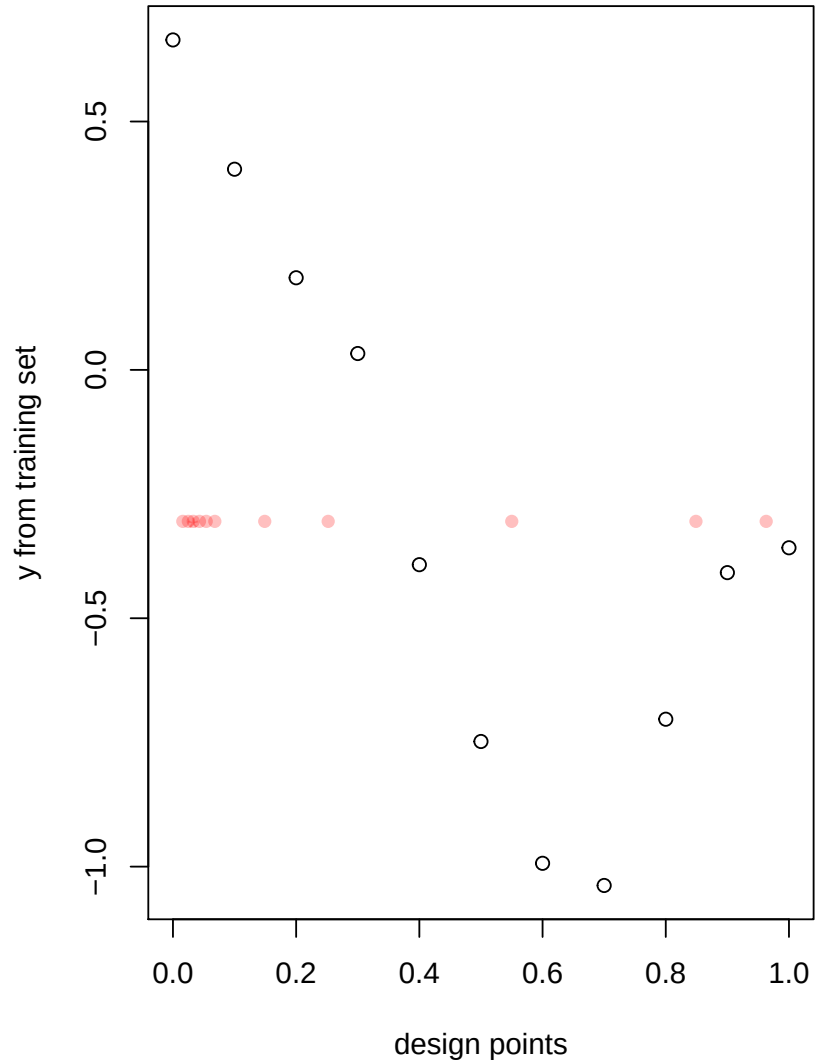

```
x_seq[which.max(w_seq)] == updateDesign2$addD[1]
```

- It seems to me almost like the distance term sort of dominates in this case, considering that the points are so spread out.
- Next, suppose I generate a function from a Matern kernel and test the design points by their RSS to see which hypothesized kernel is better - the true, Matern kernel, or the Gaussian kernel, which is also being considered.
- Due to the strong spreading out tendency, I let $\alpha = 2p$ to get a better idea of which areas of the support are favored.

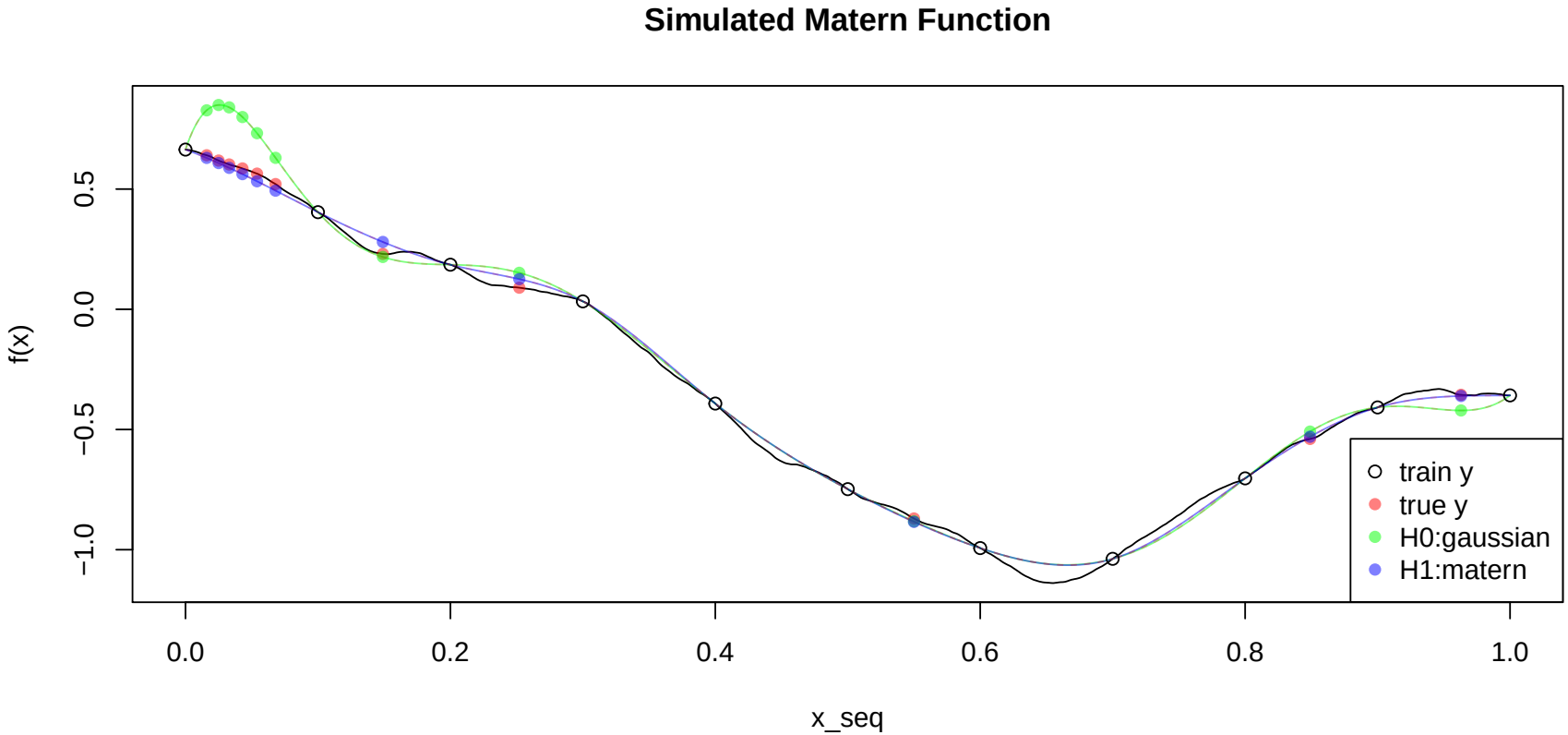
Simulated Matern Function




Frequency
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0
design points


y from training set
−1.0
−0.5
0.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
design points

Evaluating MMED Design




```
RSS0 # gaussian
```

```
## [1] 0.24002
```

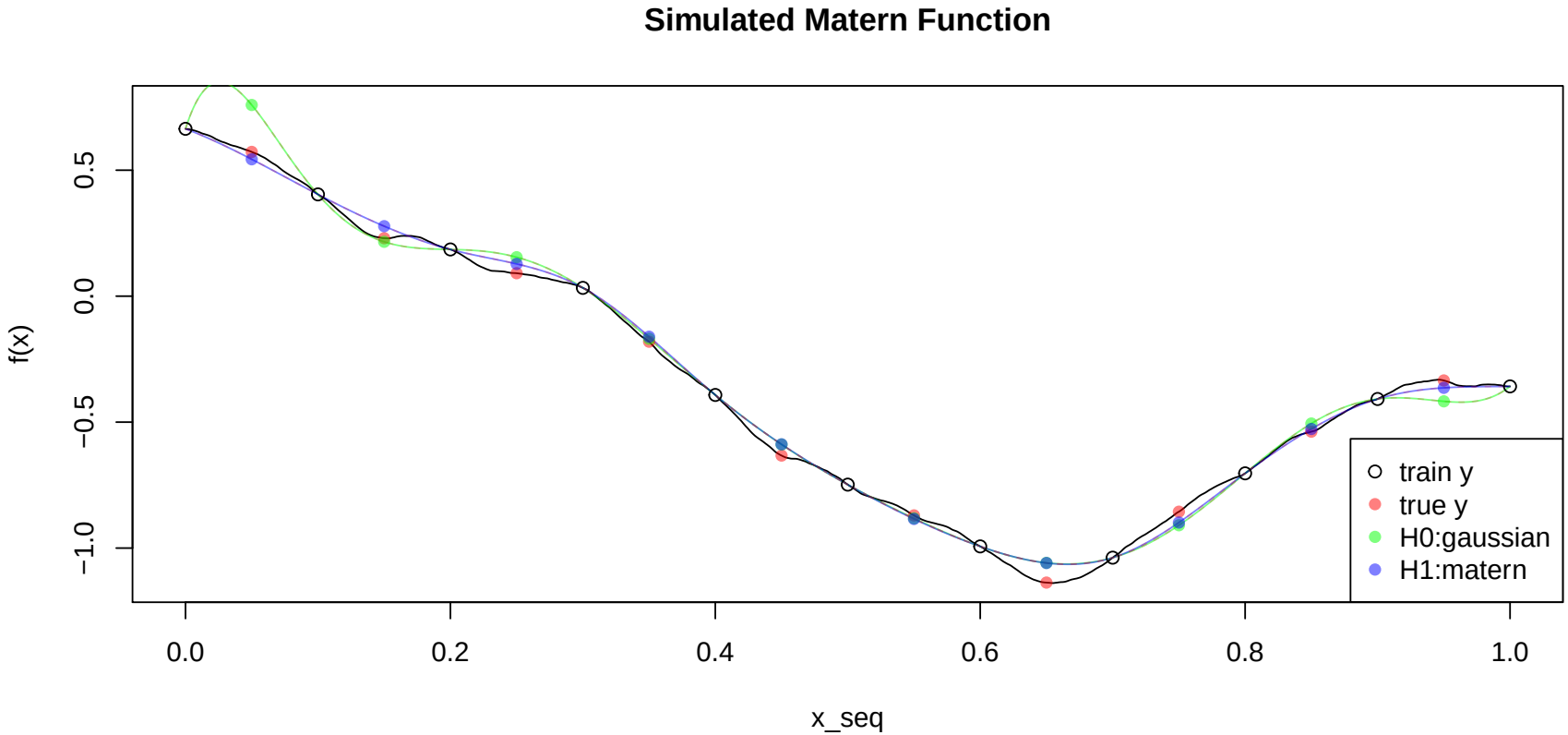
```
RSS1 # matern (true)
```

```
## [1] 0.006781766
```

```
log(RSS0/RSS1)
```

```
## [1] 3.566485
```

Compare to Space-filling




```
RSS0 # gaussian
```

```
## [1] 0.05803846
```

```
RSS1 # matern (true)
```

```
## [1] 0.01592616
```

```
log(RSS0/RSS1)
```

```
## [1] 1.293143
```

More Simulated Functions

Typically, the RSS of the incorrect hypothesis (H_0 : Gaussian Kernel) is higher than that of the correct hypothesis. We can see how their log ratio is distributed (should be greater than 0 if the correct model is favored, i.e. RSS of the incorrect model is greater) and also see what proportion of the simulations has a log ratio larger than 0.

```
summary(log(RSS01mmed_vec))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.09734 3.56649 4.26862 4.43133 5.42193 7.72948
```

```
sum(RSS01mmed_vec>1)/length(RSS01mmed_vec)
```

```
## [1] 1
```

```
summary(log(RSS01sf_vec))
```

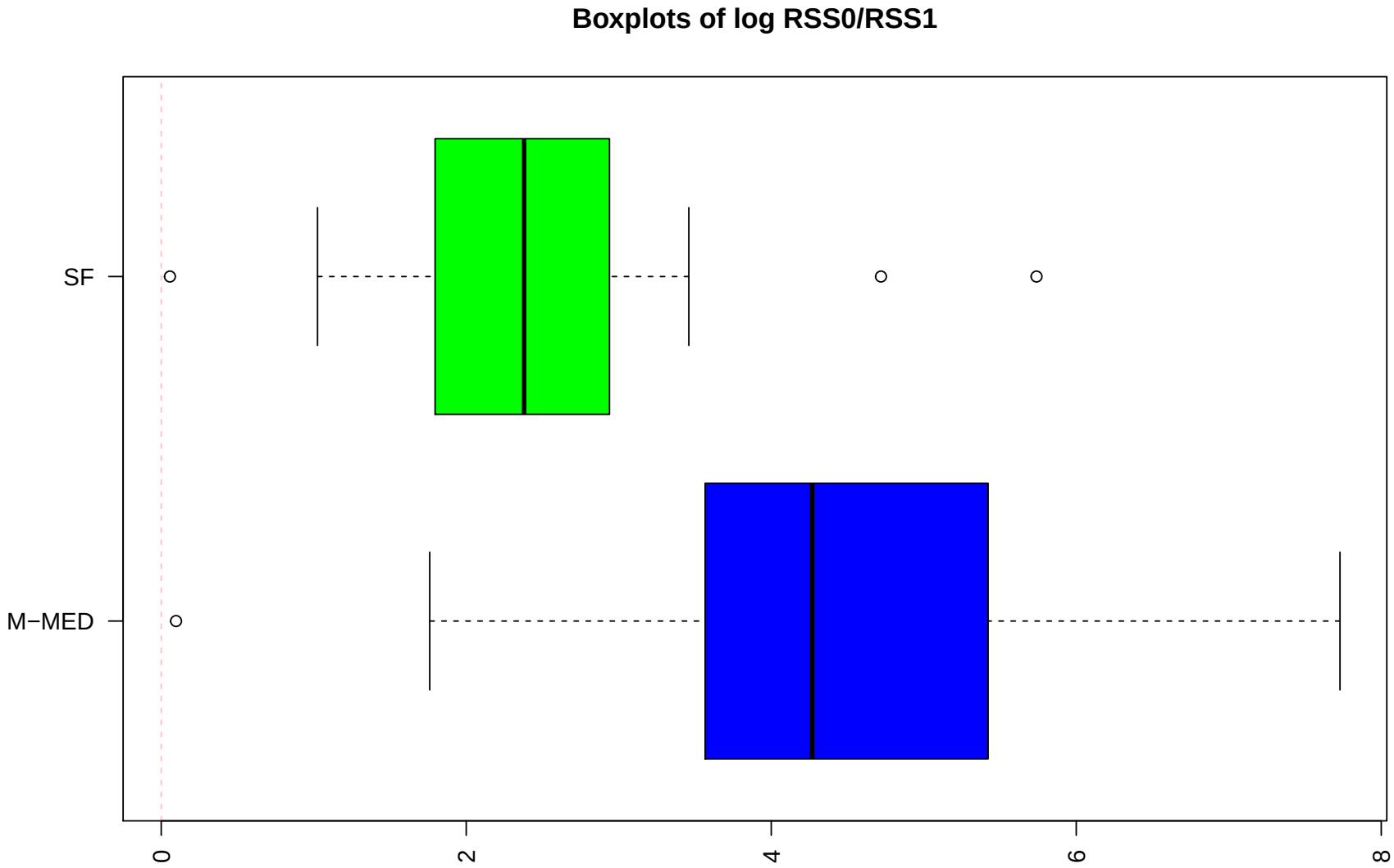
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.05701 1.79592 2.37963 2.47080 2.93953 5.73969
```

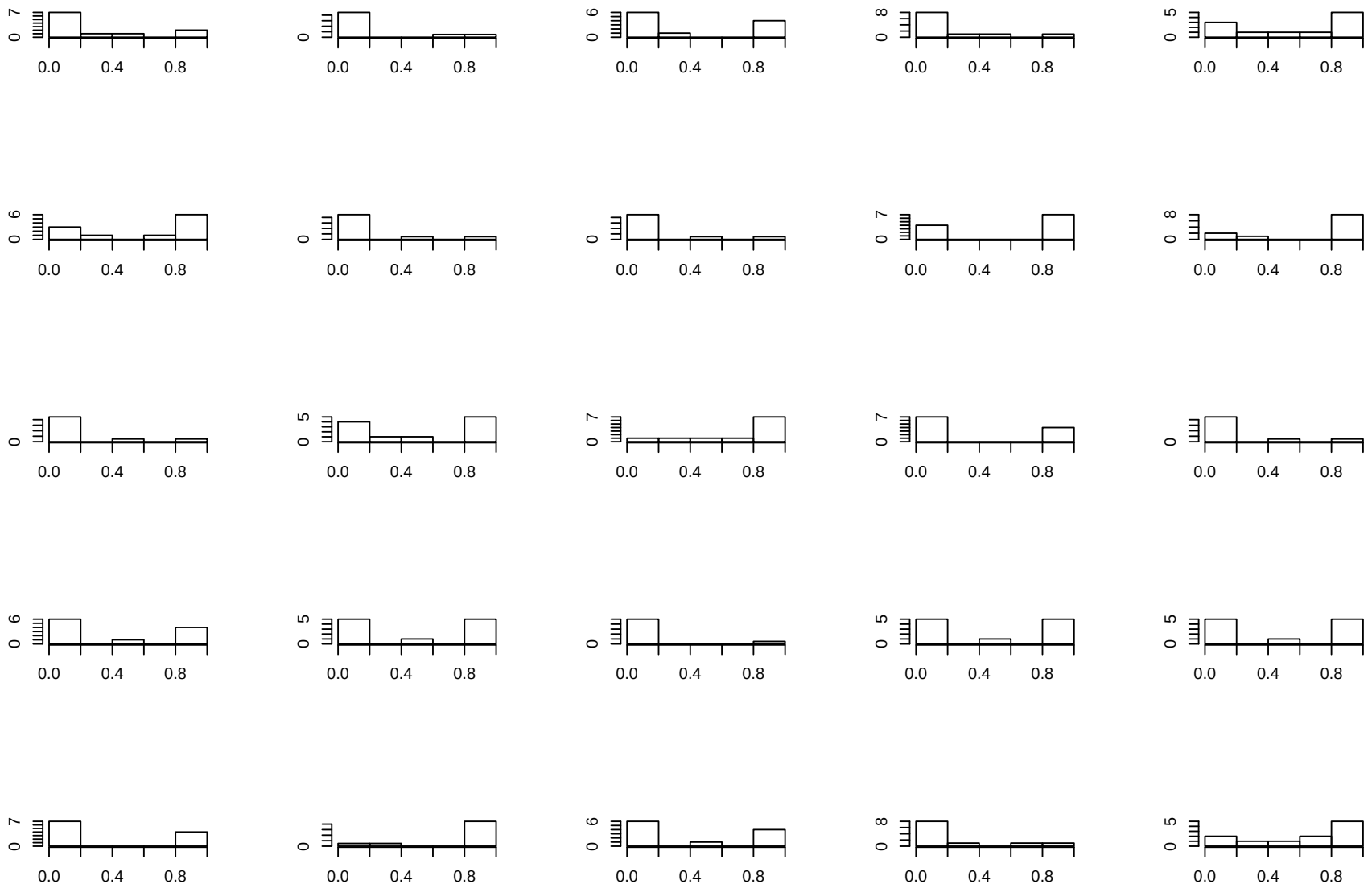
```
sum(RSS01sf_vec>1)/length(RSS01sf_vec)
```

```
## [1] 1
```

```
sum(RSS01mmed_vec>=RSS01sf_vec)/length(RSS01sf_vec)
```

```
## [1] 1
```


Boxplots of log RSS0/RSS1
SF
M−MED
0
2
4
6
8

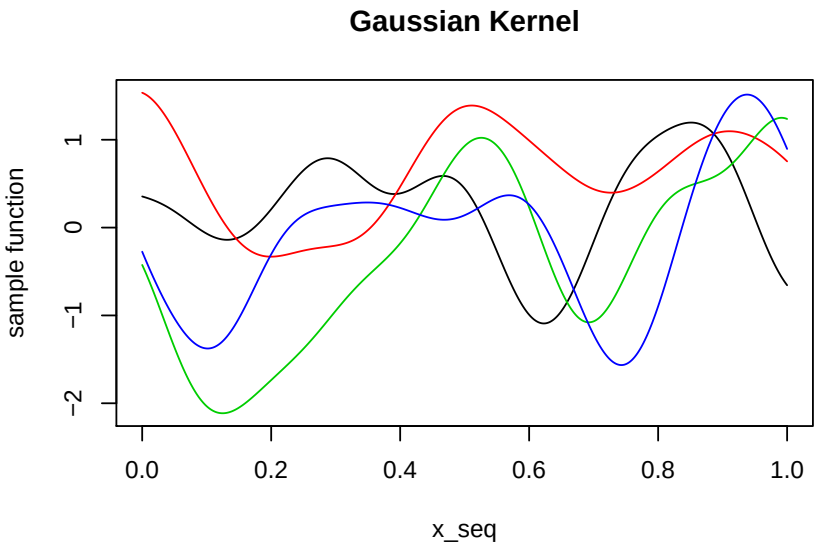

0.0
0.4
0.8

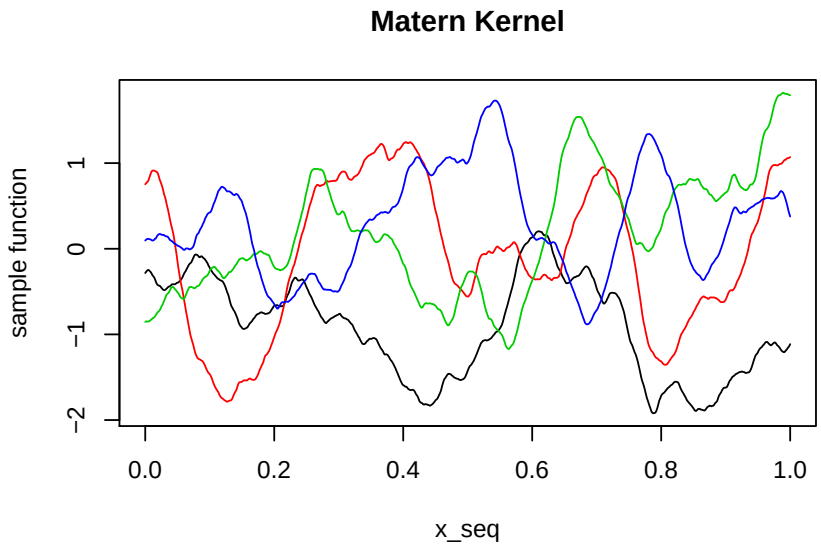
Gaussian vs. Matern, $l = 0.1$

Gaussian vs. Matern with l = 0.1

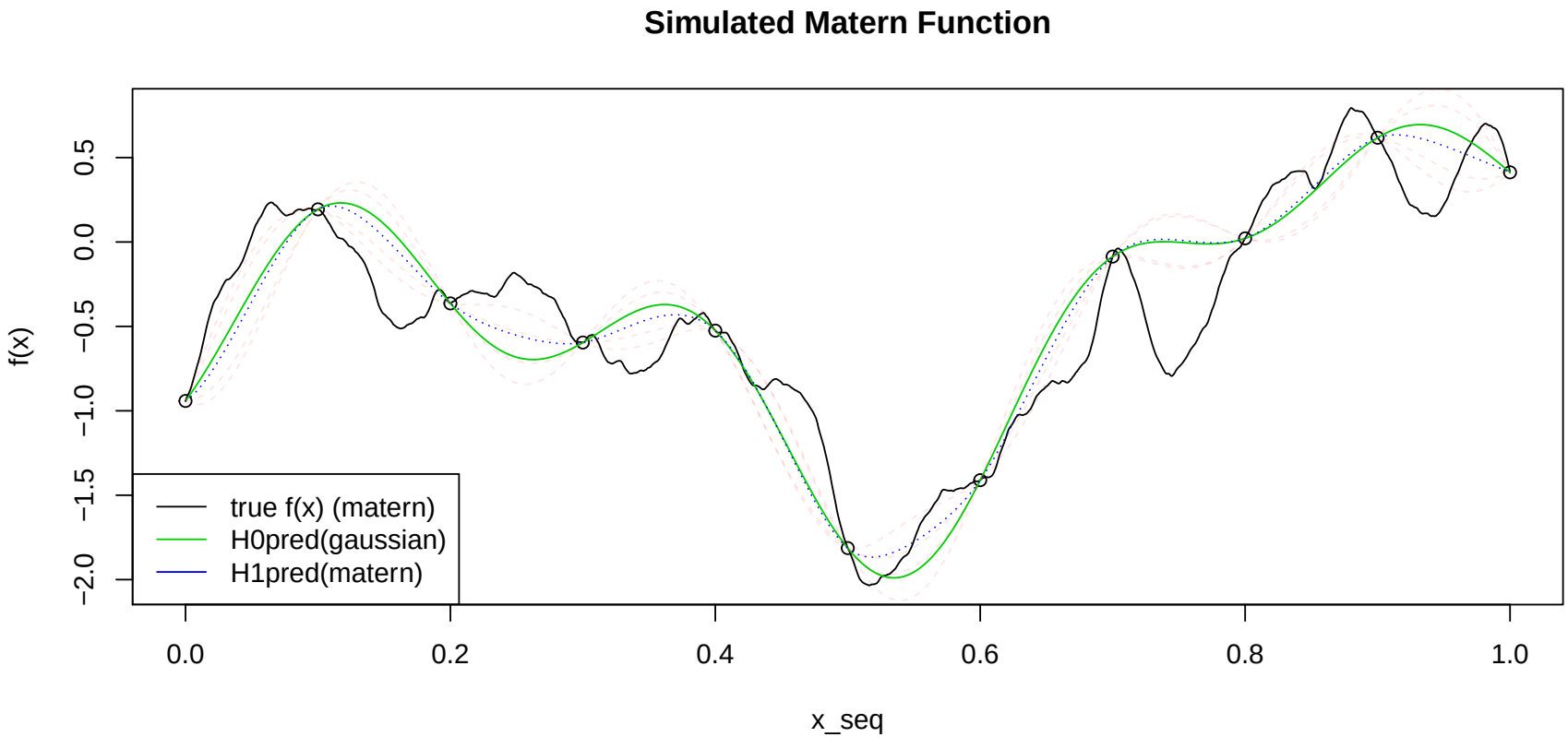
length-scale parameter $\ell = 0.1$

with length-scale parameters $\ell_G = 0.1$ and $\ell_P = 2$ respectively

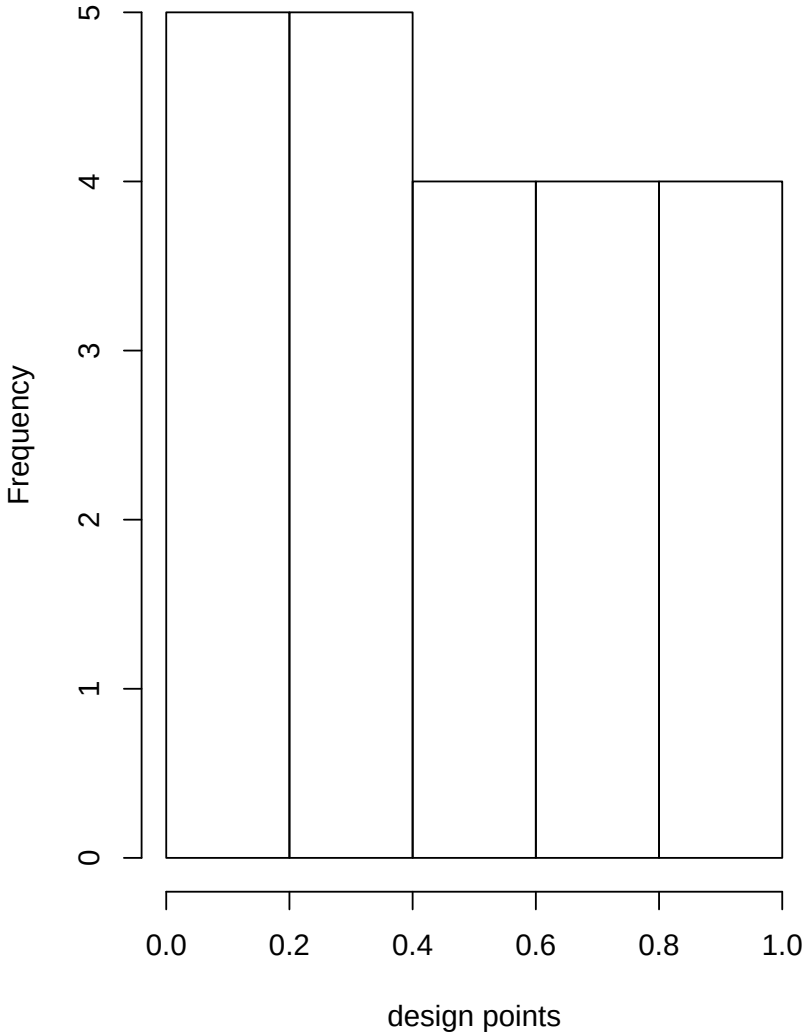


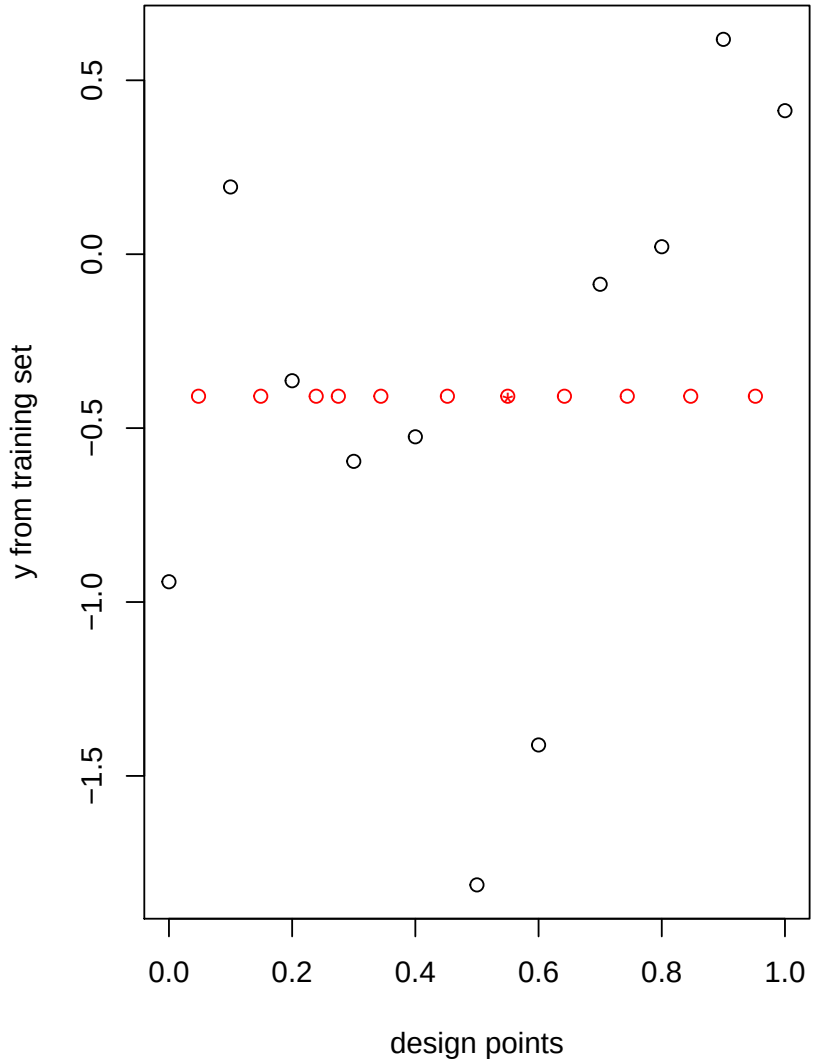


Simulated Matern Function

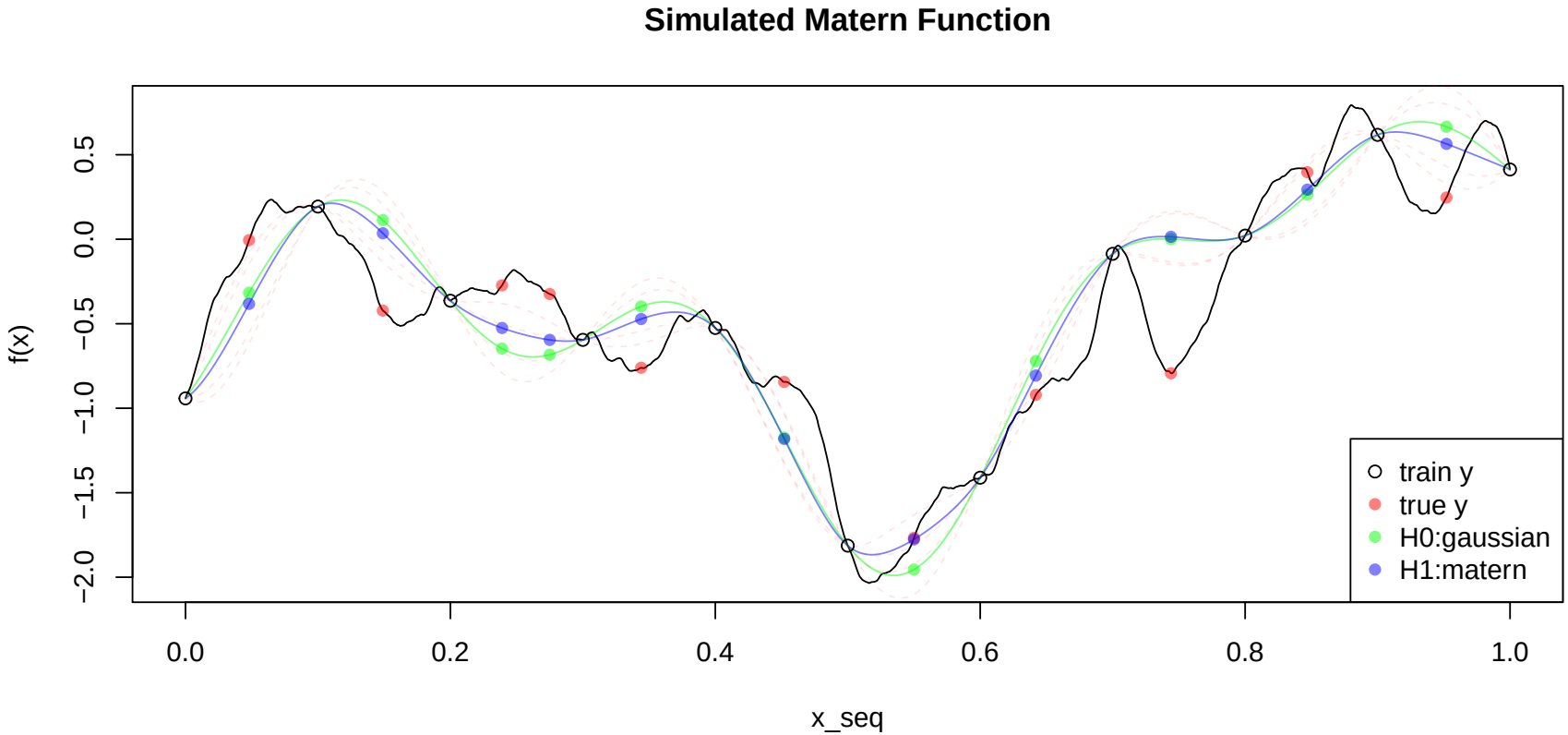


MMED


Frequency
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
design points


y from training set
−1.5
−1.0
−0.5
0.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
design points

Evaluating MMED Design




```
RSS0 # gaussian
```

```
## [1] 1.790802
```

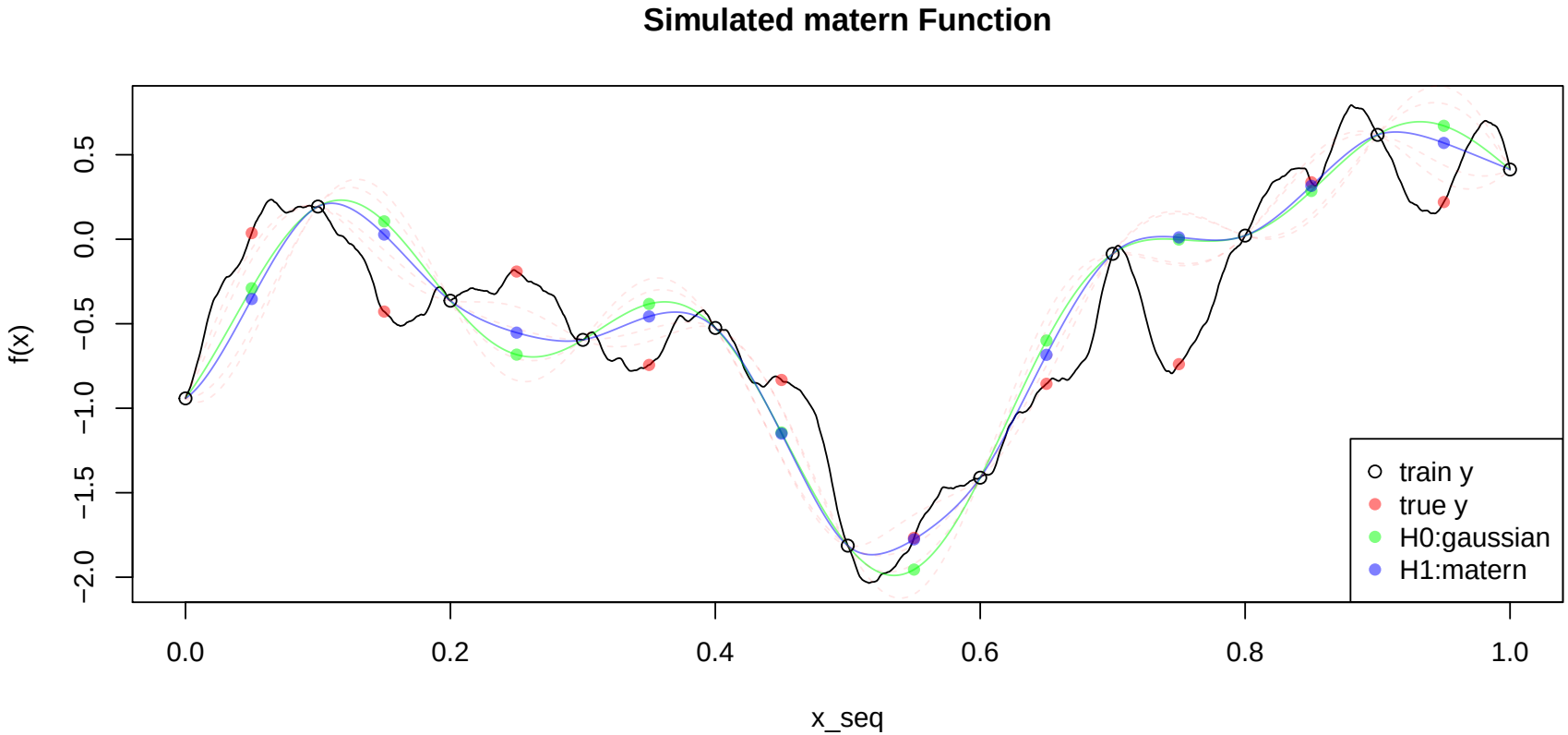
```
RSS1 # matern (true)
```

```
## [1] 1.462096
```

```
log(RSS0/RSS1)
```

```
## [1] 0.2027924
```

Compare to Space-filling




```
RSS0 # gaussian
```

```
## [1] 1.710809
```

```
RSS1 # matern (true)
```

```
## [1] 1.388908
```

```
log(RSS0/RSS1)
```

```
## [1] 0.2084482
```

More Simulated Functions

Typically, the RSS of the incorrect hypothesis (H_0 : Gaussian Kernel) is higher than that of the correct hypothesis. We can see how their ratio is distributed (should be greater than 1 if the correct model is favored, i.e. RSS of the incorrect model is greater) and also see what proportion of the simulations has a ratio larger than 1.

```
summary(log(RSS01mmed_vec))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.633442 -0.149048 -0.031970  0.007411  0.162616  0.579114
```

```
sum(RSS01mmed_vec>1)/length(RSS01mmed_vec)
```

```
## [1] 0.44
```

```
summary(log(RSS01sf_vec))
```

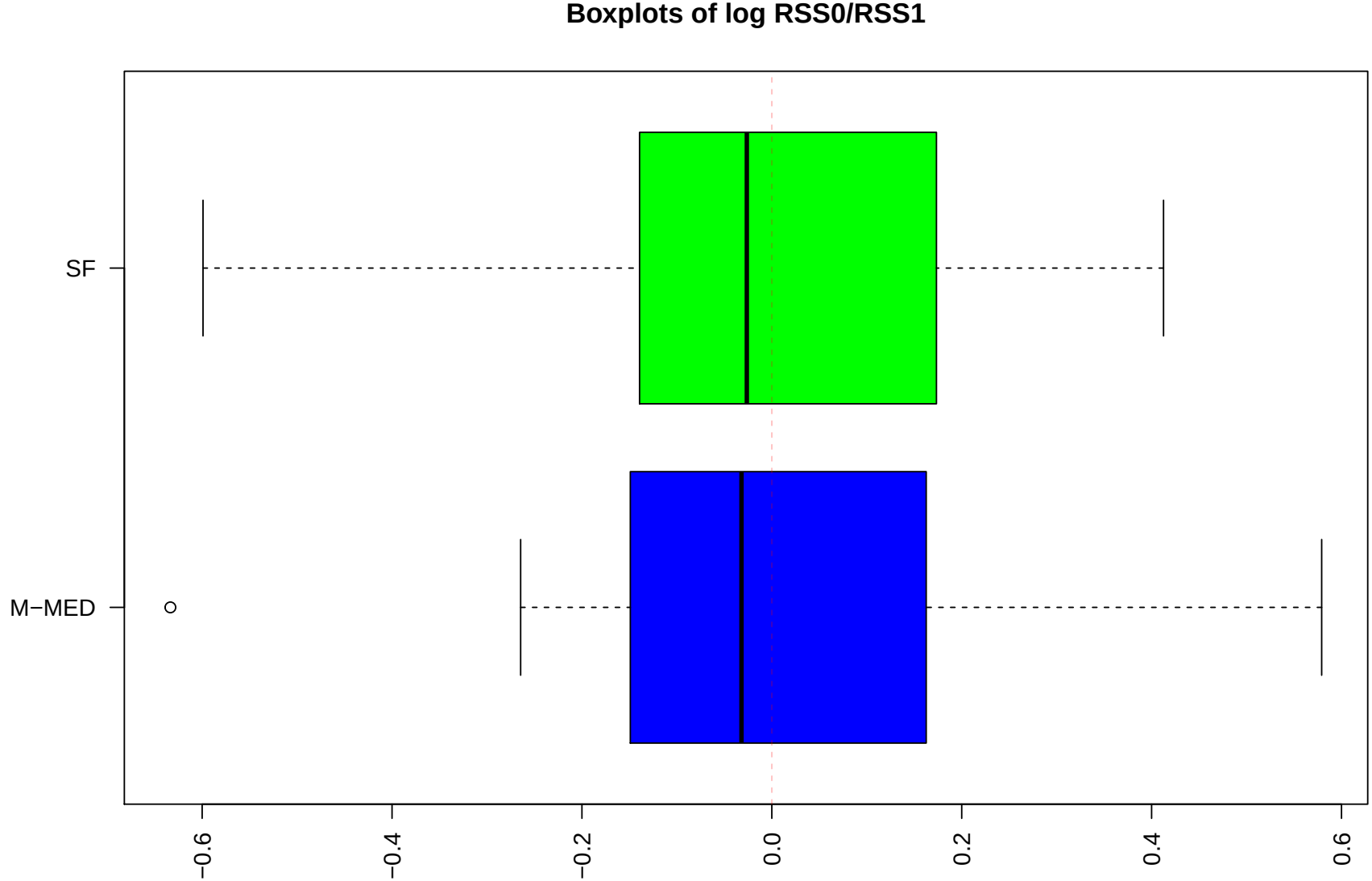
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.599082 -0.139300 -0.026337  0.003826  0.173384  0.412548
```

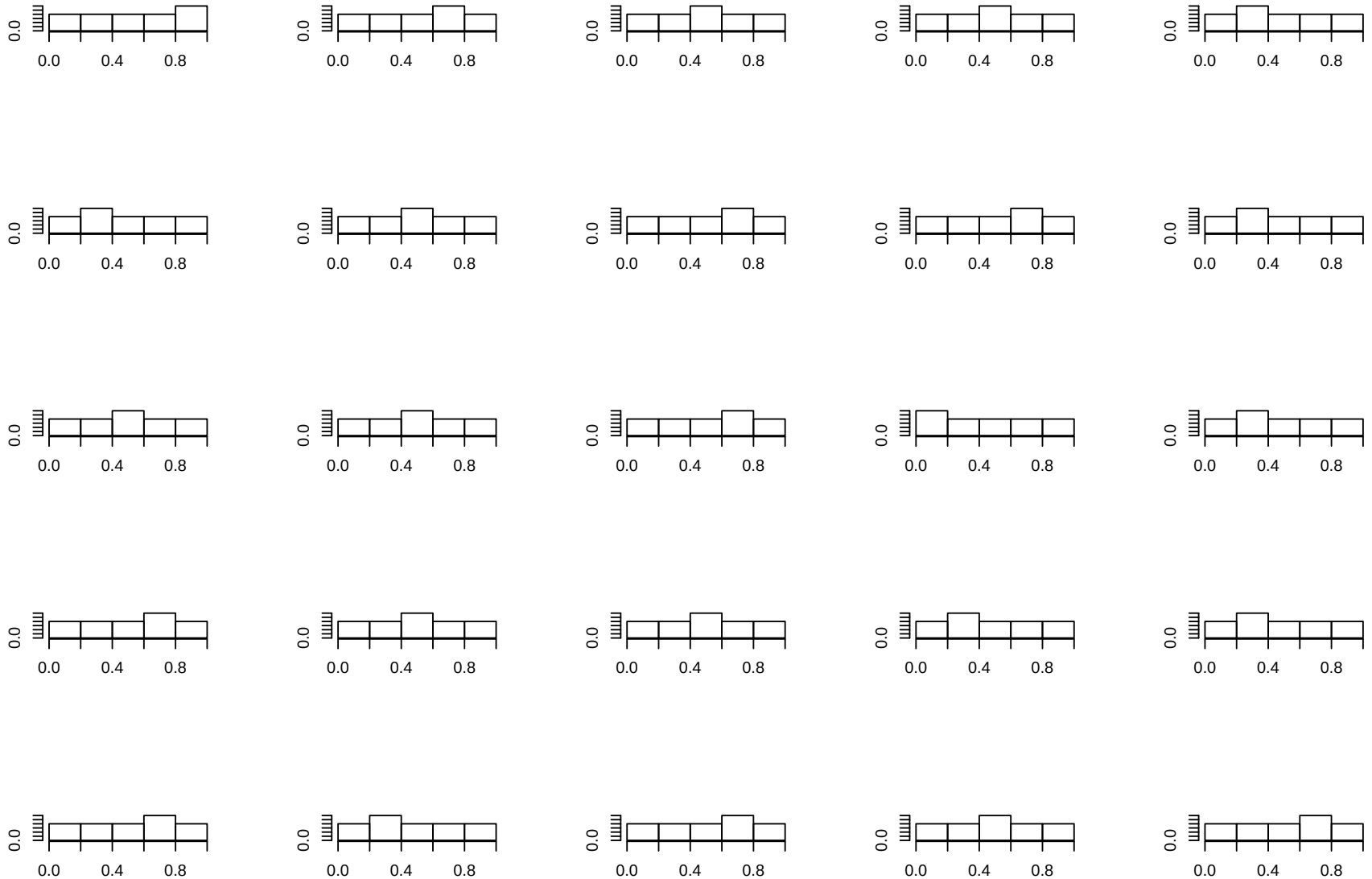
```
sum(RSS01sf_vec>1)/length(RSS01sf_vec)
```

```
## [1] 0.44
```

```
sum(RSS01mmed_vec>=RSS01sf_vec)/length(RSS01sf_vec)
```

```
## [1] 0.44
```

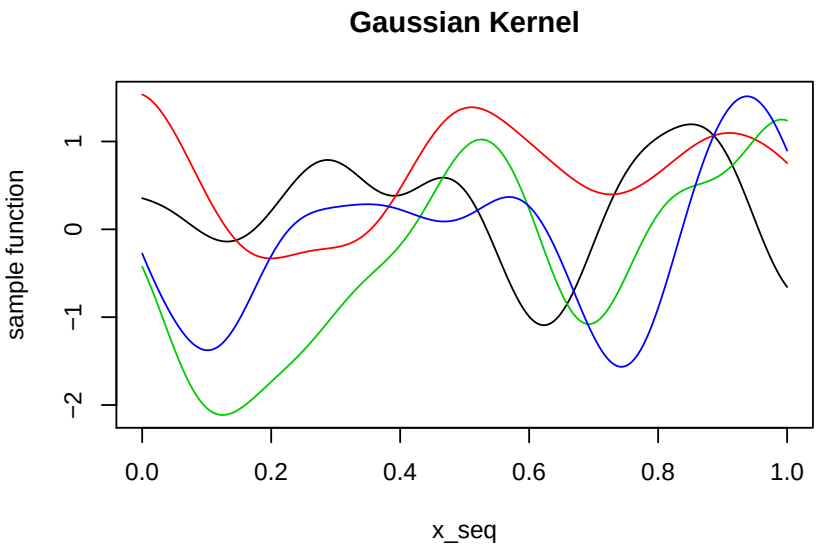

Boxplots of log RSS0/RSS1
SF
M−MED
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

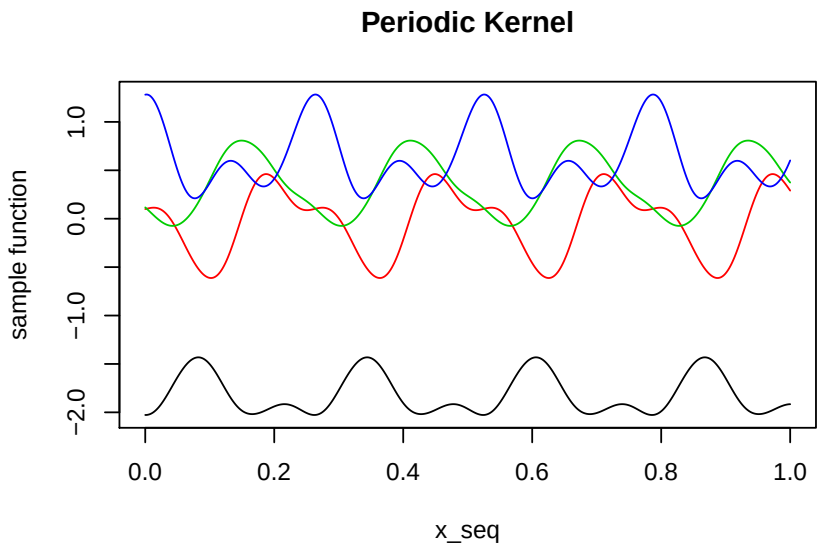

0.0
0.0
0.4
0.8

Gaussian vs. Periodic Kernel

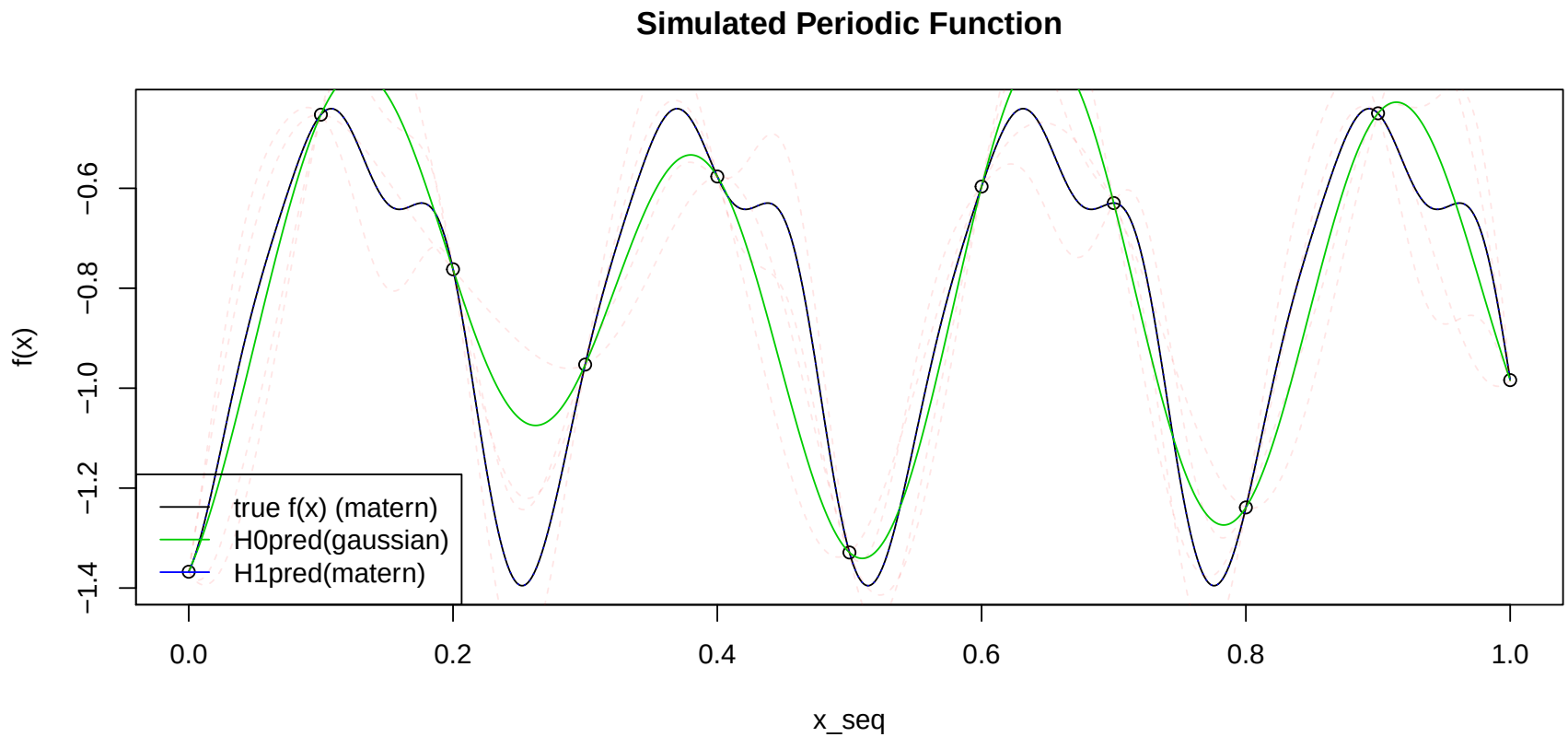
Gaussian vs. Periodic

with length-scale parameters $\ell_G = 0.1$ and $\ell_P = 2$ respectively

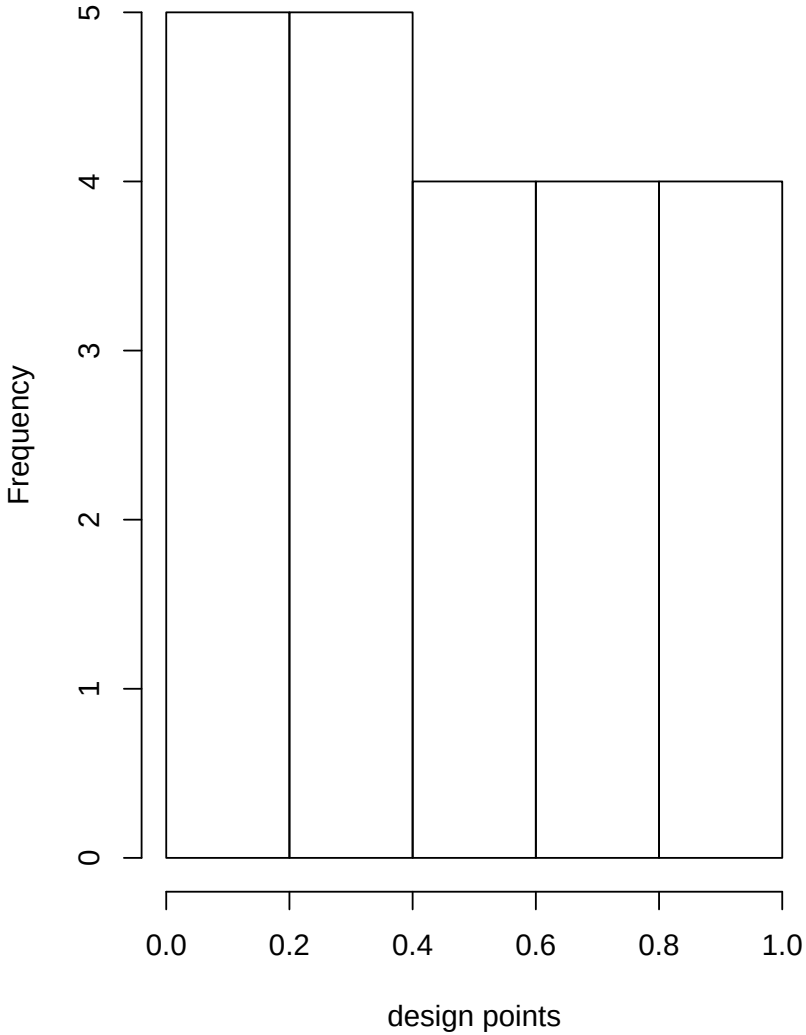


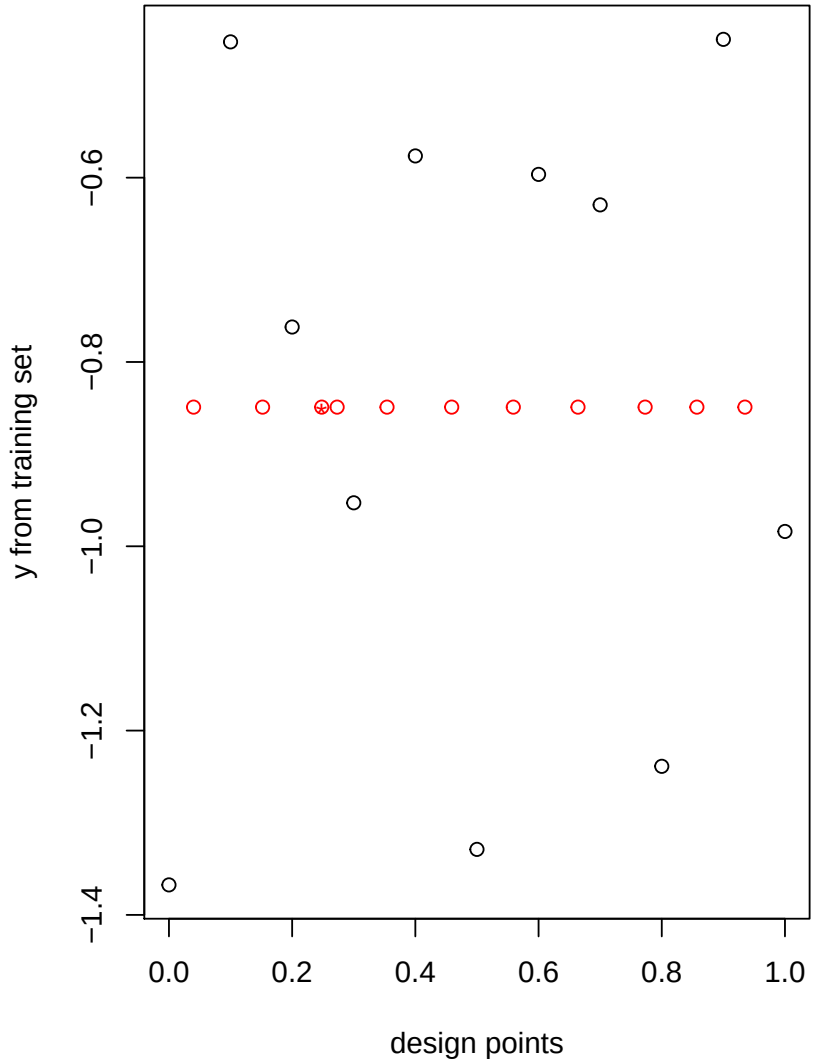


Simulated Periodic Function

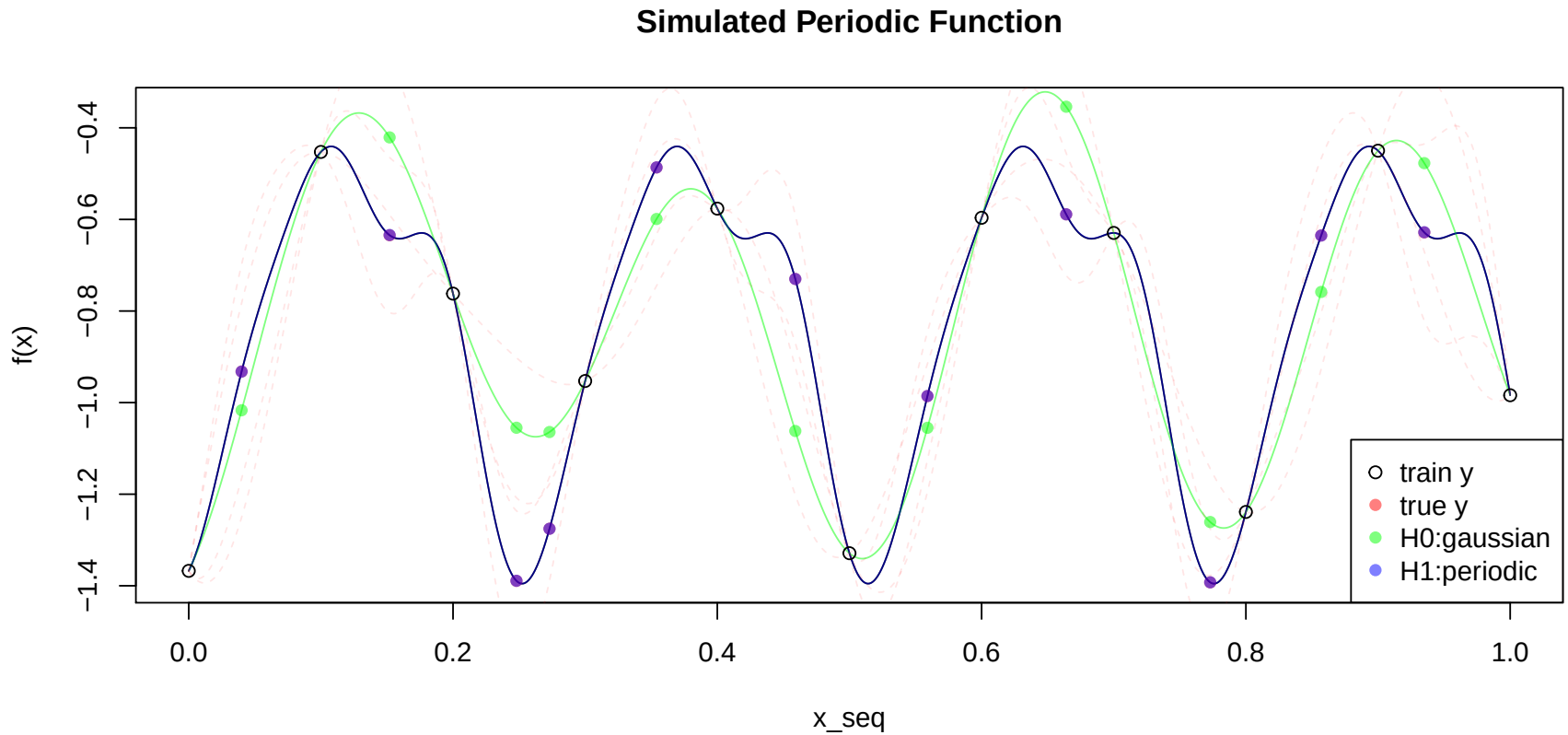


MMED


Frequency
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0
design points


y from training set
−1.4
−1.2
−1.0
−0.8
−0.6
0.0
0.2
0.4
0.6
0.8
1.0
design points

Evaluating MMED Design




```
RSS0 # gaussian
```

```
## [1] 0.4475884
```

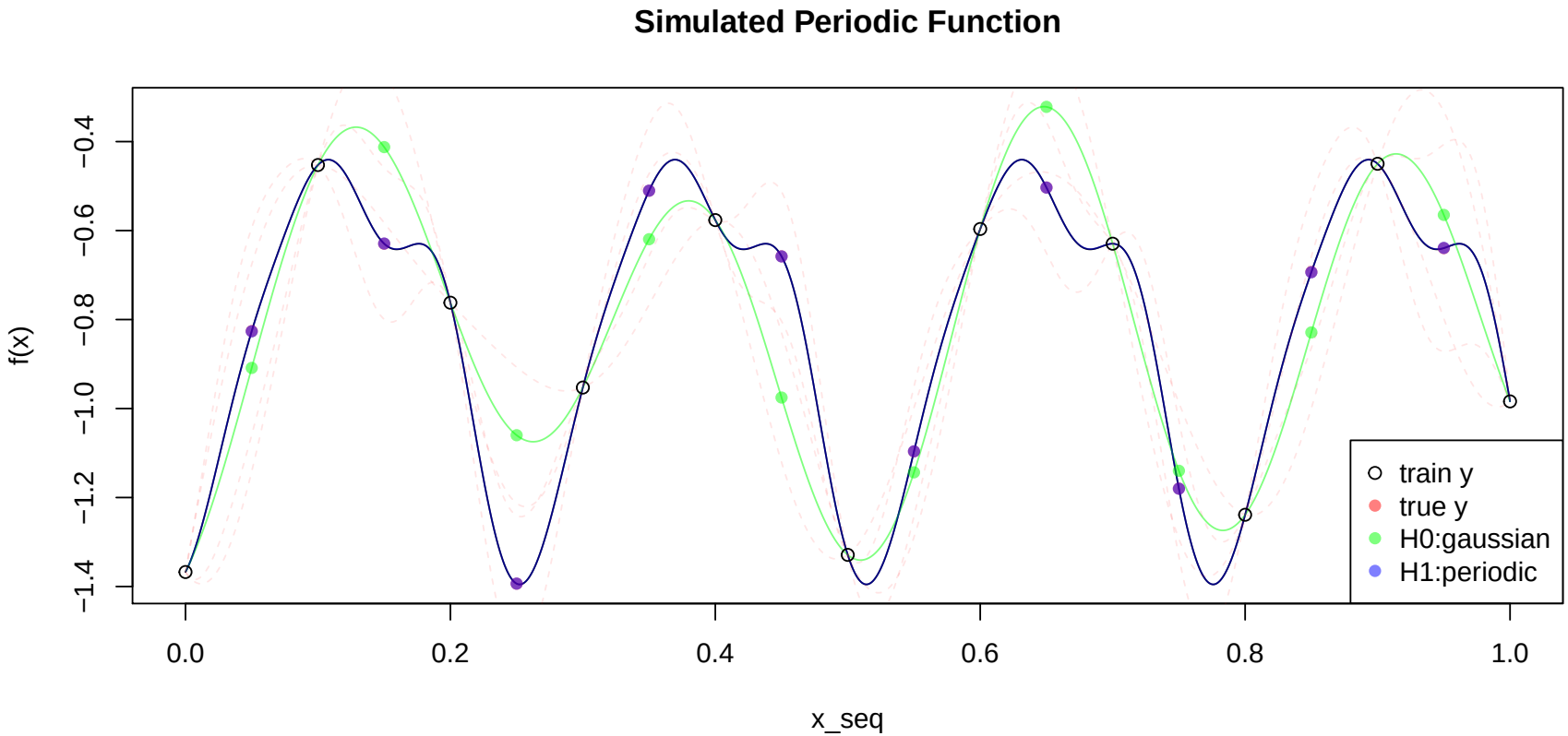
```
RSS1 # periodic (true)
```

```
## [1] 5.103724e-07
```

```
log(RSS0/RSS1)
```

```
## [1] 13.68424
```

Compare to Space-filling




```
RSS0 # gaussian
```

```
## [1] 0.3386493
```

```
RSS1 # periodic (true)
```

```
## [1] 1.102595e-06
```

```
log(RSS0/RSS1)
```

```
## [1] 12.63505
```

More Simulated Functions

Typically, the RSS of the incorrect hypothesis (H_0 : Gaussian Kernel) is higher than that of the correct hypothesis. We can see how their ratio is distributed (should be greater than 1 if the correct model is favored, i.e. RSS of the incorrect model is greater) and also see what proportion of the simulations has a ratio larger than 1.

```
summary(log(RSS01mmed_vec))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.10   12.99   13.57   14.05   14.95   19.15
```

```
sum(RSS01mmed_vec>1)/length(RSS01mmed_vec)
```

```
## [1] 1
```

```
summary(log(RSS01sf_vec))
```

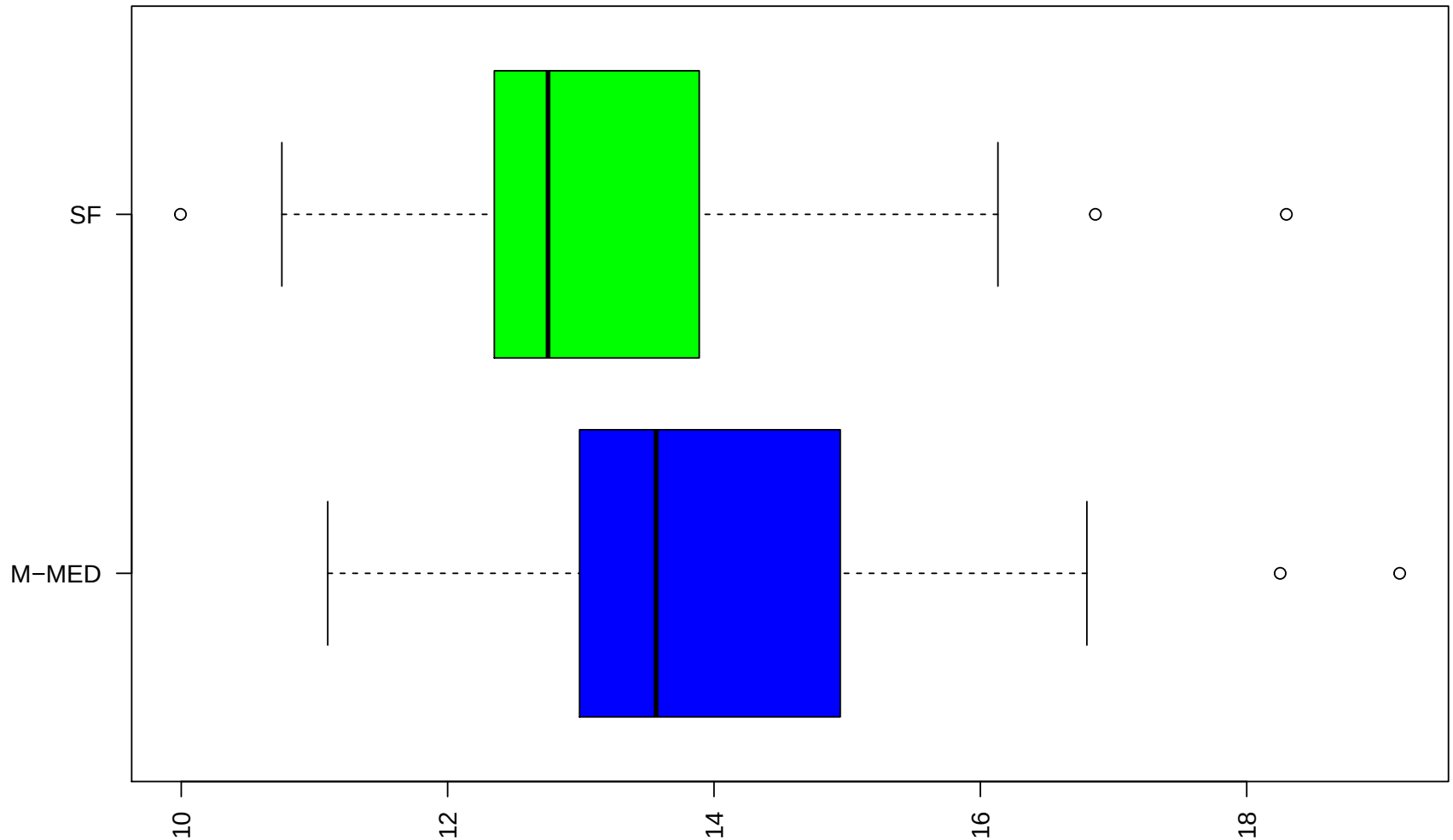
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   9.994  12.351  12.754  13.269  13.889  18.298
```

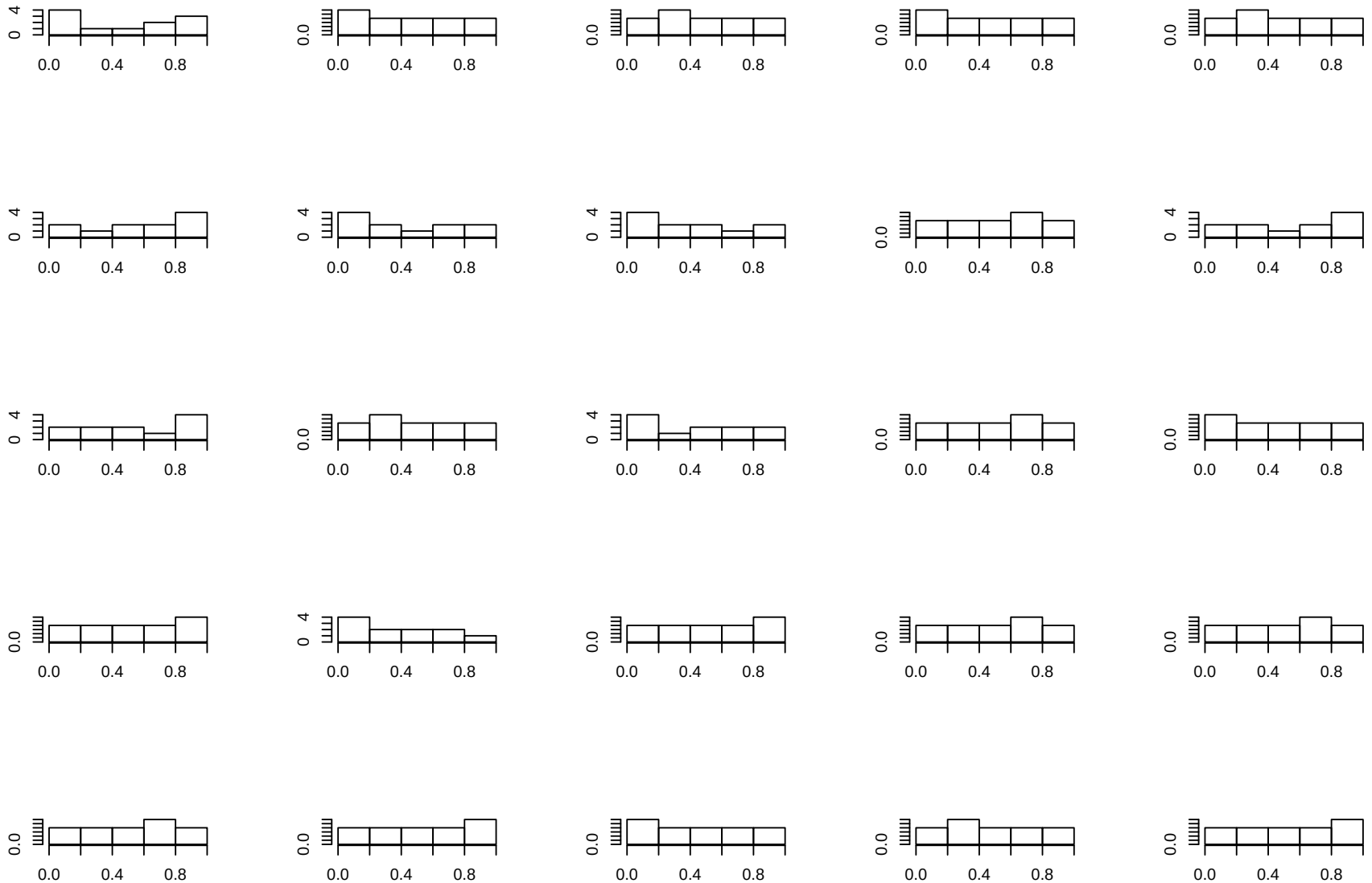
```
sum(RSS01sf_vec>1)/length(RSS01sf_vec)
```

```
## [1] 1
```

```
sum(RSS01mmed_vec>=RSS01sf_vec)/length(RSS01sf_vec)
```

```
## [1] 0.96
```


Boxplots of log RSS0/RSS1
SF
M−MED
10
12
14
16
18


0 4
0.0
0.4
0.8