1D Inverse problem class activity

Drew Remmenga

2022-11-29

Setup

```
library(viridis)
```

```
## Loading required package: viridisLite
```

```
source("naturalSplineBasis.R")
source("naturalCubicSplineR.R")
source("InverseExampleFunctions.R")


suppressMessages(library( fields))
suppressMessages(library( fda))
```

Create true function

True function is evaluated on a grid 1 to 200

```
set.seed(123)

M<- 200
sGrid<- 1:M


uGrid<- sGrid/ M
truef<- 9*uGrid*(1-uGrid)^3
```

Now generate the observations via the W matrix 80 observations

```
N<- 80
W<- matrix(0,N,M)
I<- matrix( NA, N,2)
center<- rep(NA, N)
set.seed(222)
for(  k in 1:N){
c0<- sample( 1:M, 1, replace=FALSE)
w0<- sample( 5:(M/2), 1, replace=FALSE)
```

```
i<-  c( max( c(c0-w0,1) ) , min(c(c0+w0,M) ))
I[k,]<- i
W[k, i[1]:i[2] ] <- 1/(i[2]- i[1])
center[k]<- (i[2] + i[1])/2
}
```

Stats on W matrix just to check distribution of widths.

```
dim( W)
```

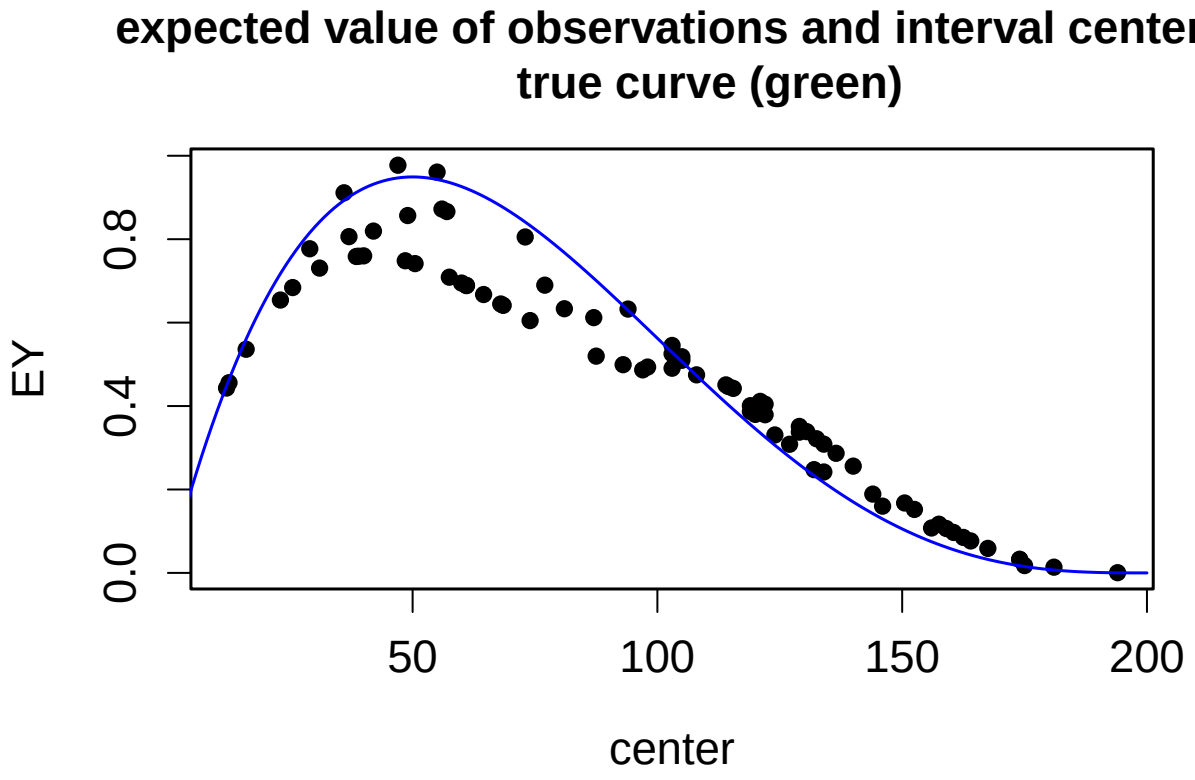
```
## [1]  80 200
```

```
WInd<- W>0
stats(  rowSums( WInd))
```

```
##                      [,1]
## N                80.00000
## mean             95.31250
## Std.Dev.         46.69768
## min              11.00000
## Q1               62.50000
## median           86.00000
## Q3              133.50000
## max             185.00000
## missing values    0.00000
```

Find expected value of the observations and create data. The plot below emphasizes the problem of trying to look at these data without using the W matrix correctly. Here the expected observation without the error added is plotted against the center of the interval. Note that without error we should see a smooth curve. The variation here is due the different sizes of the weight function applied at different locations.

```
EY<- W%*%truef
fields.style()
plot( center, EY, pch=16)
set.seed( 222)
error<- .05*rnorm( N)
z<- EY +error
lines( sGrid, truef, col="blue")
title("expected value of observations and interval centers
      true curve (green)")
```

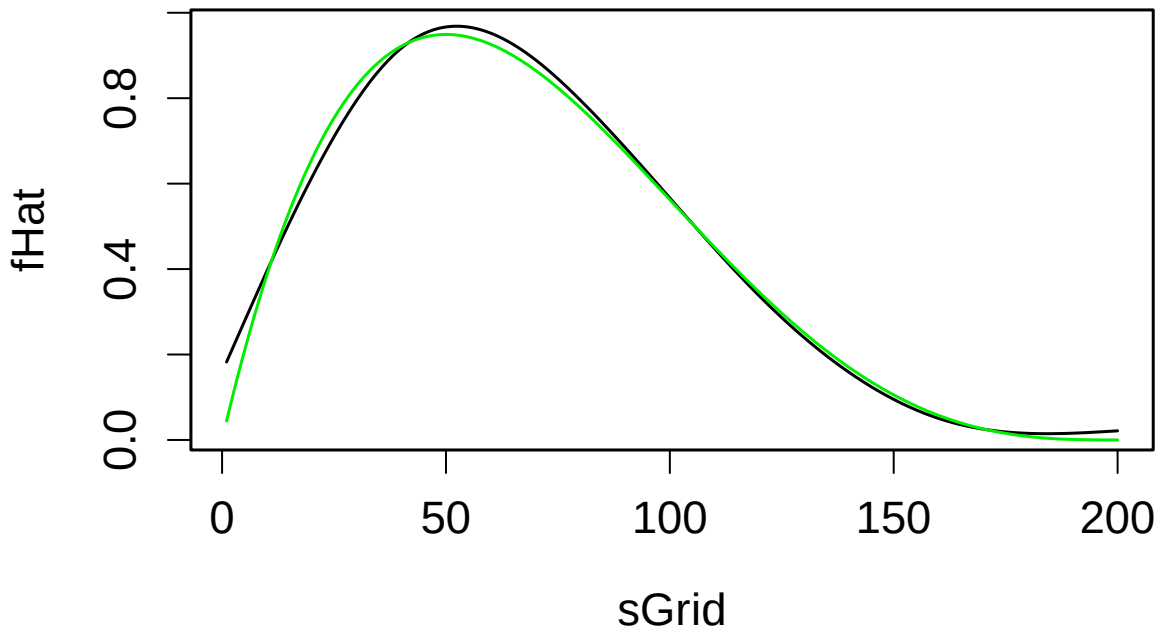


Functions for fitting

The function **fitWBS** fits a bspline to the the data correctly using the W weight matrix.

Example of fitting with 6 knots

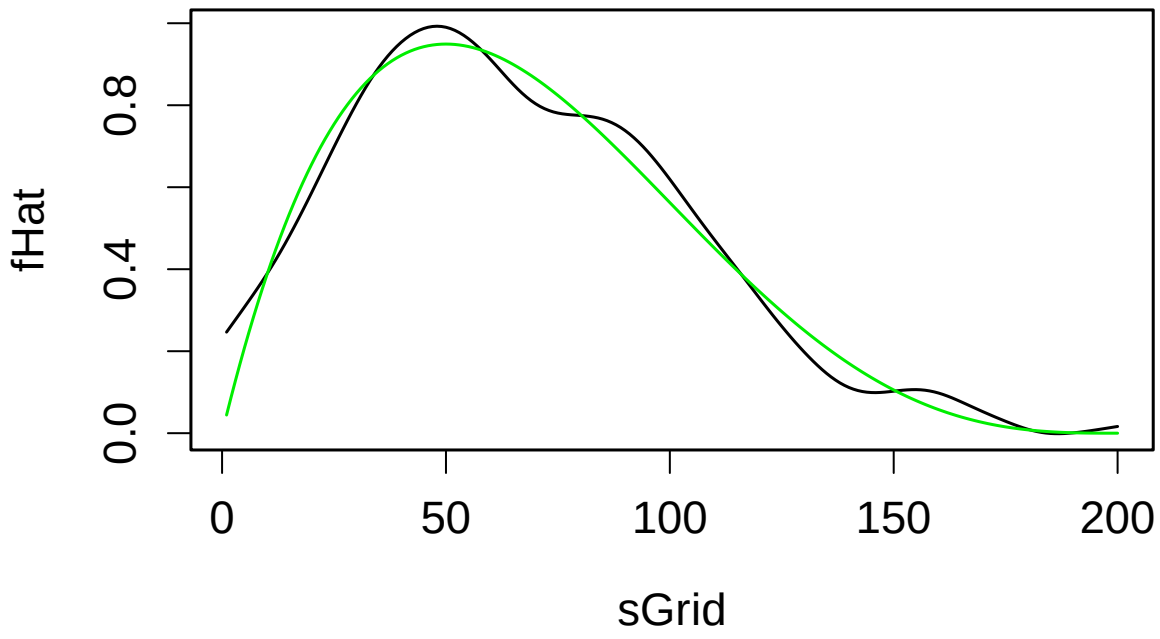
```
fHat<- fitWBS(z,W,6,sGrid)
fields.style()
plot( sGrid, fHat, type="l")
lines( sGrid, truef, col=2)
```



The function **fitWBS** fits a smoothing spline to the the data correctly using the W weight matrix. Here the smoothing parameter lambda takes over the role of choosing the number of knots. In fact this function simply puts a knot at every point of **sGrid** and relies on the roughness matrix/penalty and lambda to give a good fit.

Example fitting with lambda value equal to 50. (This is too small!)

```
fHat<- fitWSS(z,W, 50 ,sGrid)
fields.style()
plot( sGrid, fHat, type="l")
lines( sGrid, truef, col=2)
```

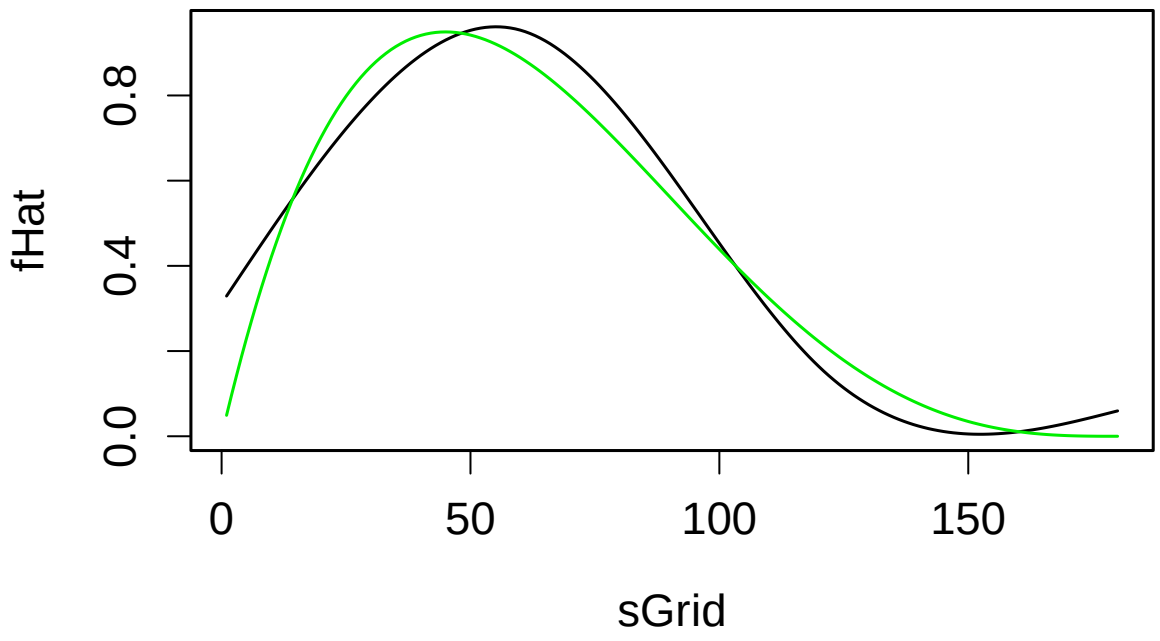


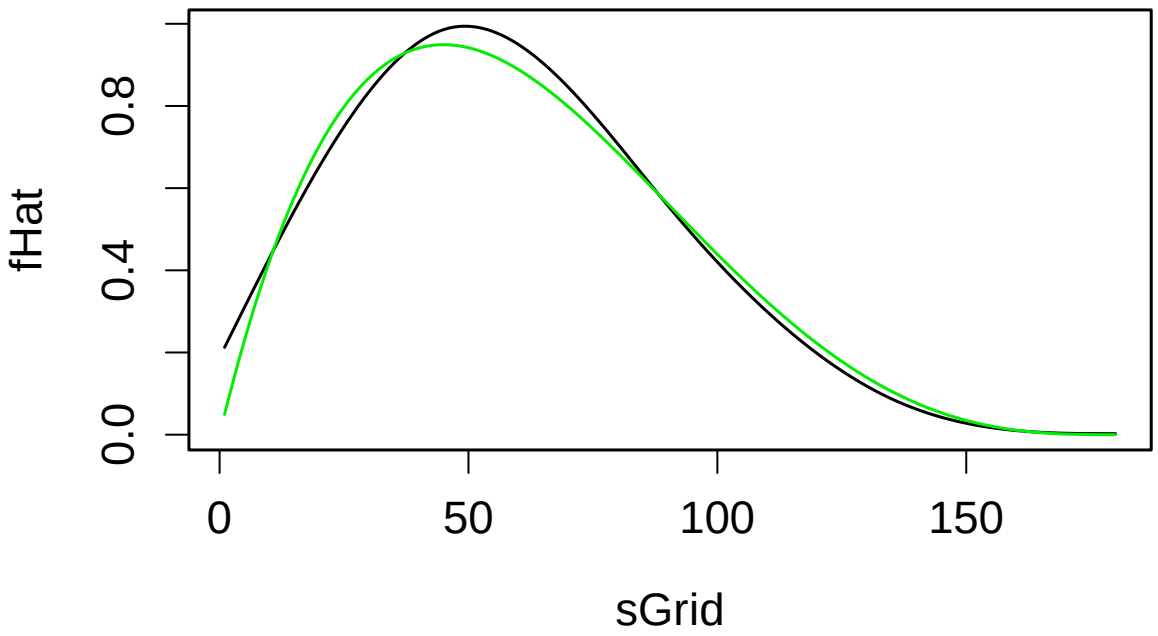
Task 1

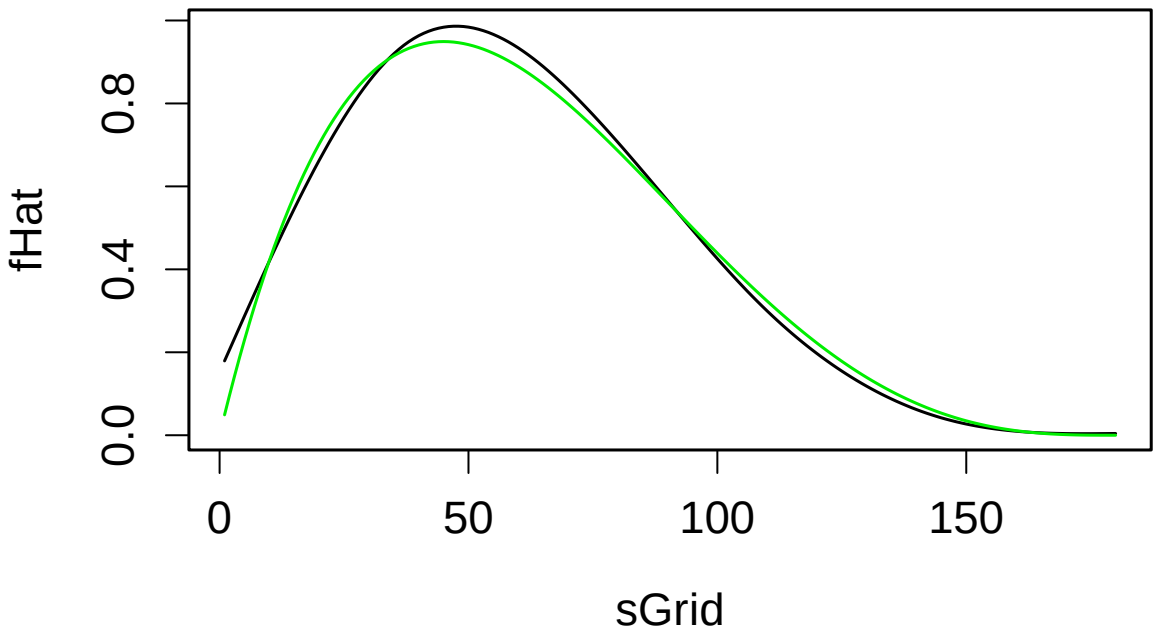
- How do you find the predicted values for the observations based on these estimates? Use sgrid and fitwss.
- How would you find residuals from the fit? fhat - fitwbs # Task 2 For the B spline model vary the knots from 4 to 15. and use 10/90 cross validation to estimate the best number of knots.

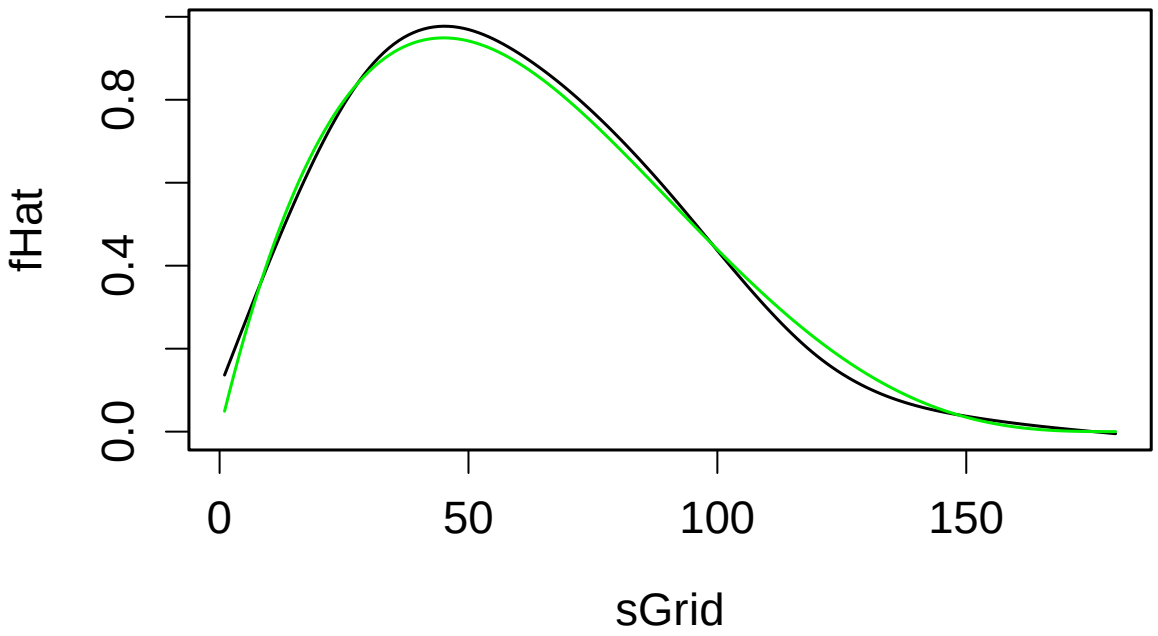
```
smp_size <- floor(0.9 * 200)
train_ind <- sample(200, size = smp_size)

train <- z[train_ind]
test <- z[-train_ind]
set.seed(123)

M<- 180
sGrid<- 1:M


uGrid<- sGrid/ M
truef<- 9*uGrid*(1-uGrid)^3
N<- 80
W<- matrix(0,N,M)
I<- matrix( NA, N,2)
center<- rep(NA, N)
```

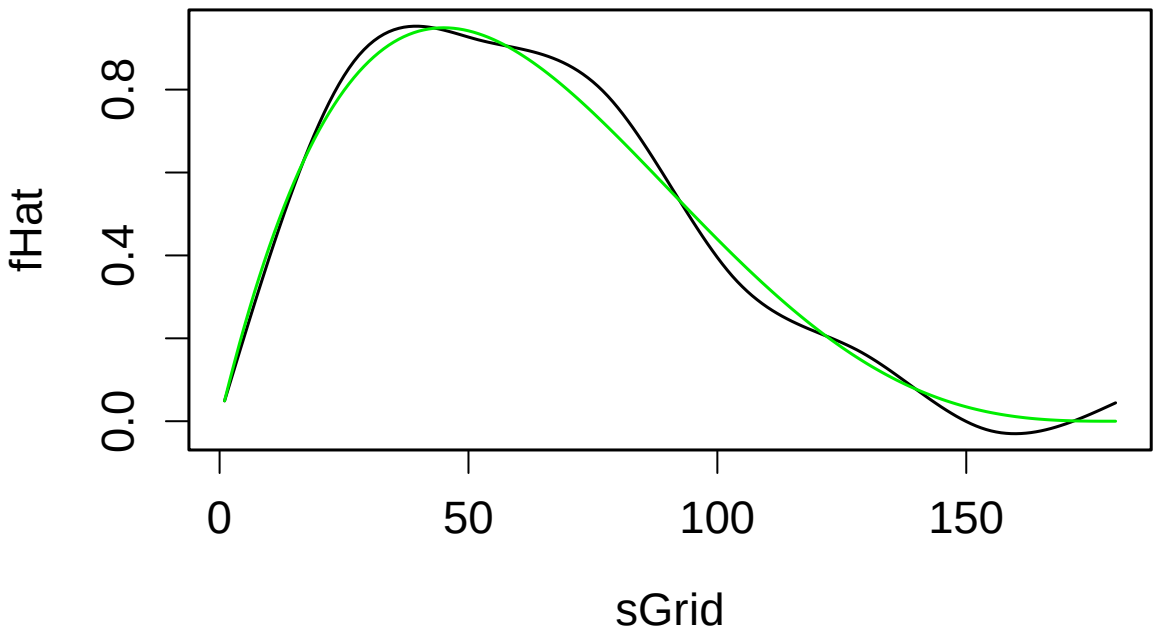
```
set.seed(222)
for(  k in 1:N){
c0<- sample( 1:M, 1, replace=FALSE)
w0<- sample( 5:(M/2), 1, replace=FALSE)
i<-  c( max( c(c0-w0,1) ) , min(c(c0+w0,M) ))
I[k,]<- i
W[k, i[1]:i[2] ] <- 1/(i[2]- i[1])
center[k]<- (i[2] + i[1])/2
}
EY<- W%*%truef
set.seed( 222)
error<- .05*rnorm( N)
z<- EY +error
val = 4:15
crossvalidation = c()
for (x in val){
fHat<- fitWBS(z,W,x,sGrid)
fields.style()
plot( sGrid, fHat, type="l")
lines( sGrid, truef, col=2)
append(crossvalidation,val=mean(fHat-truef)/x)
}
```

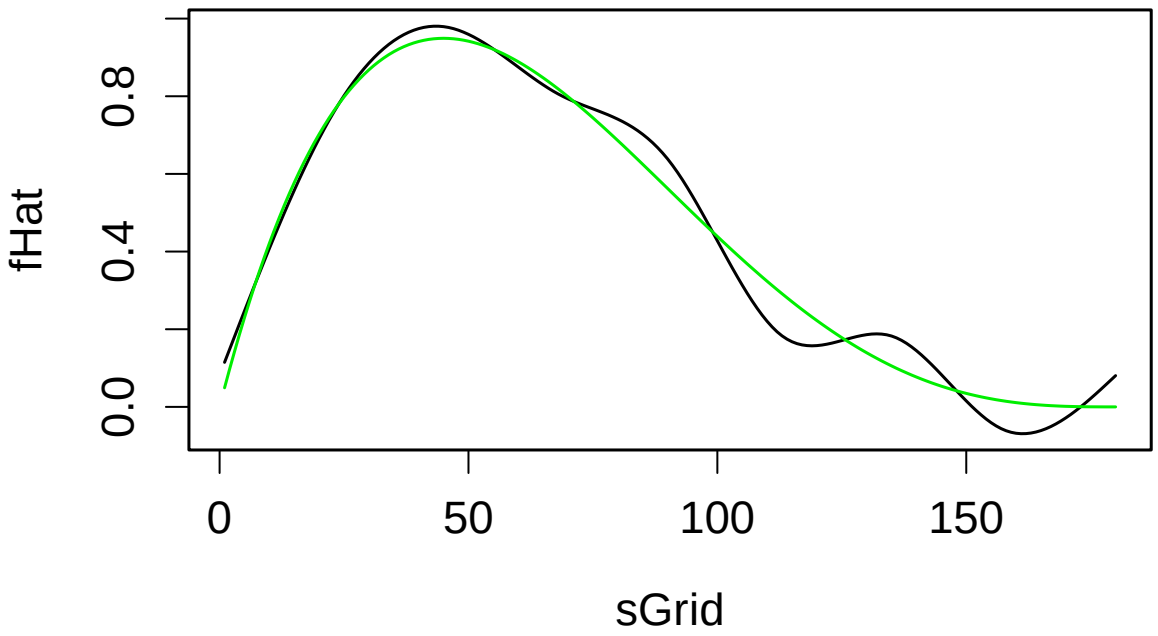


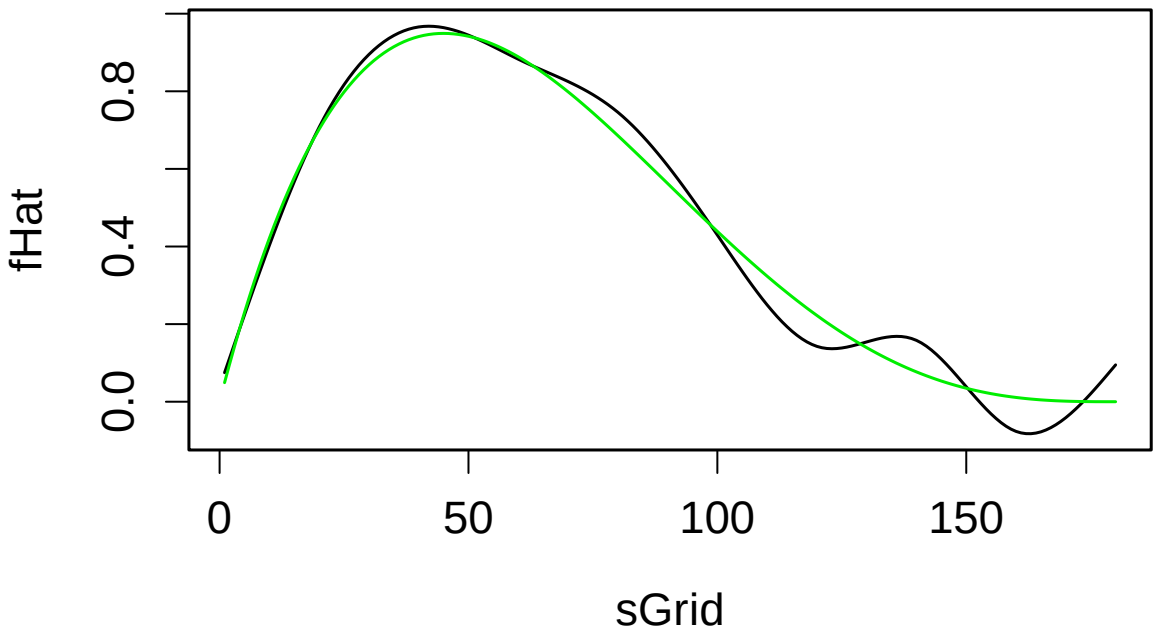

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

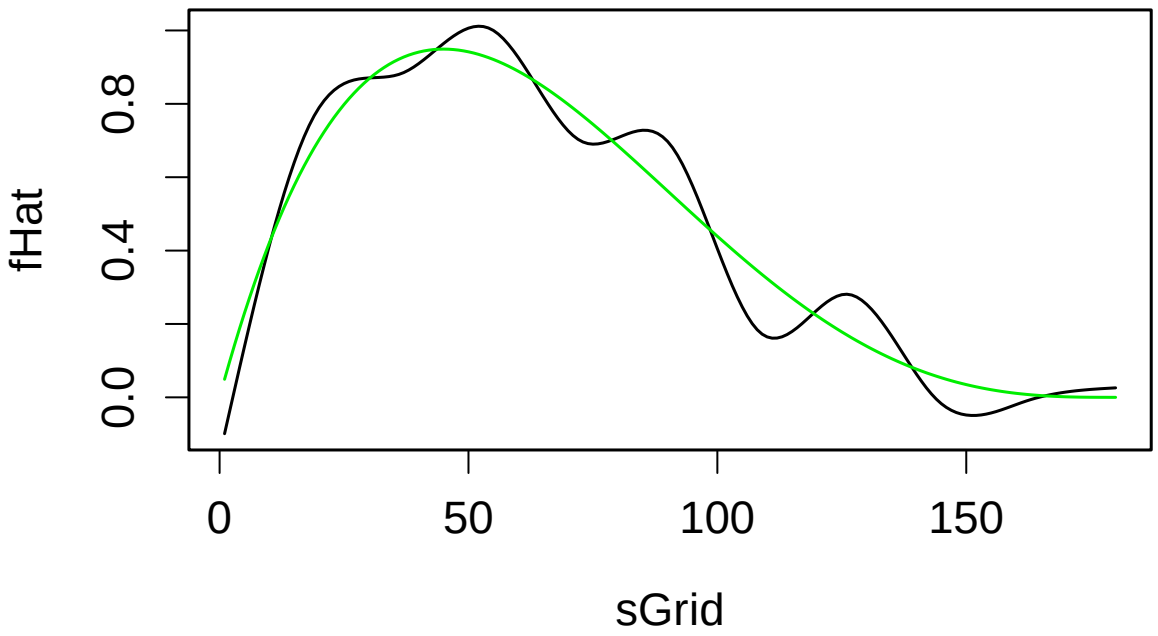

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

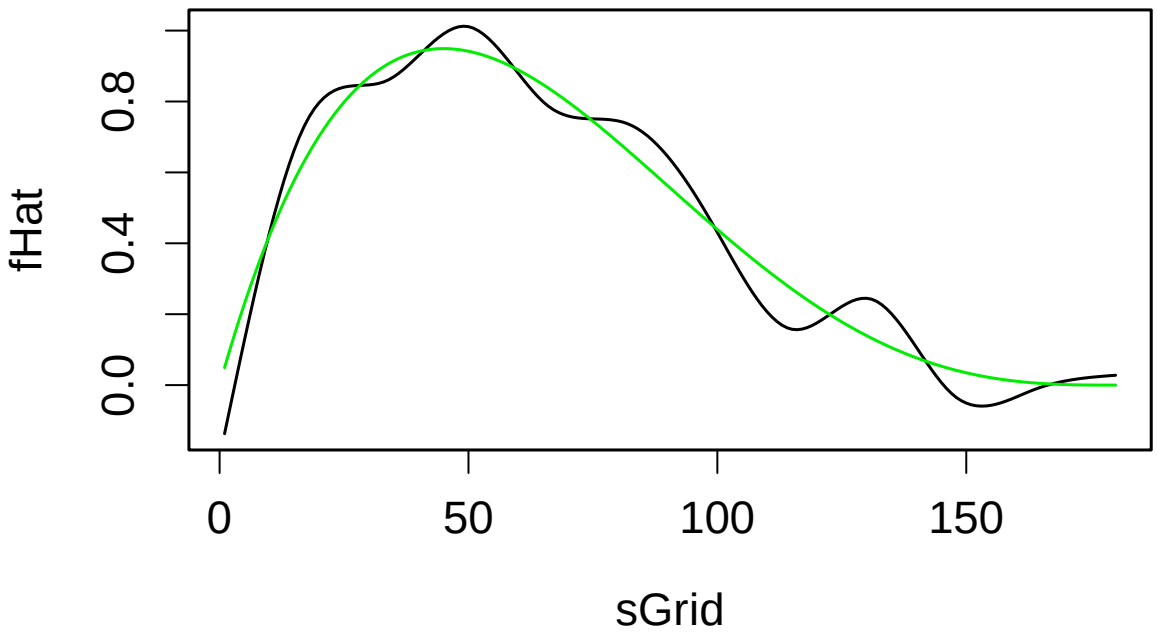

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

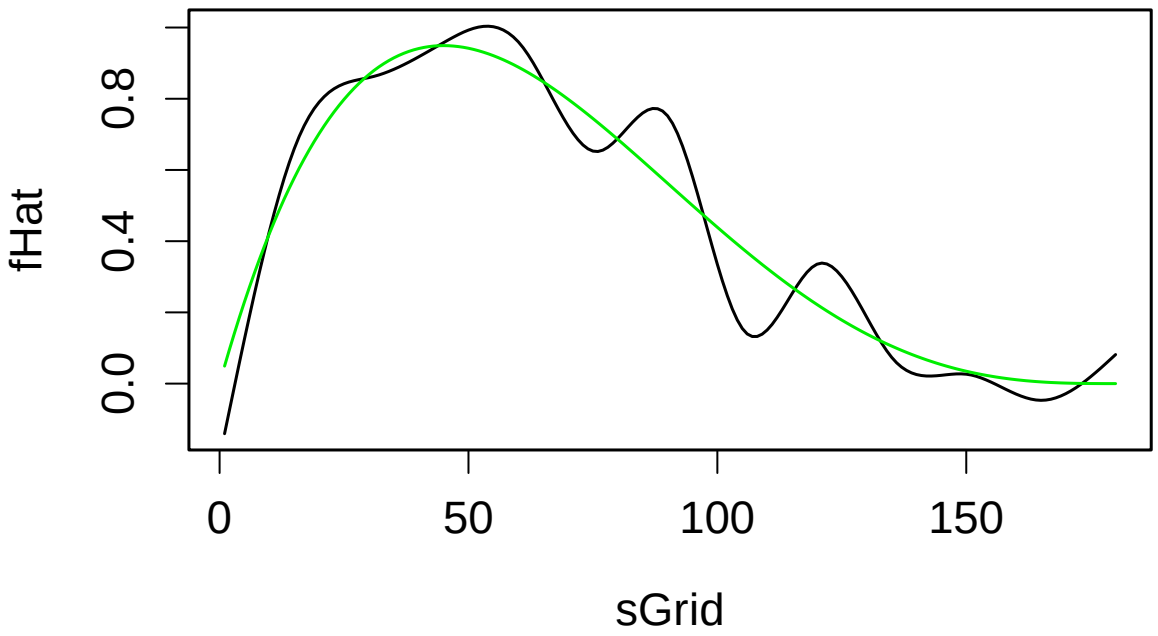

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

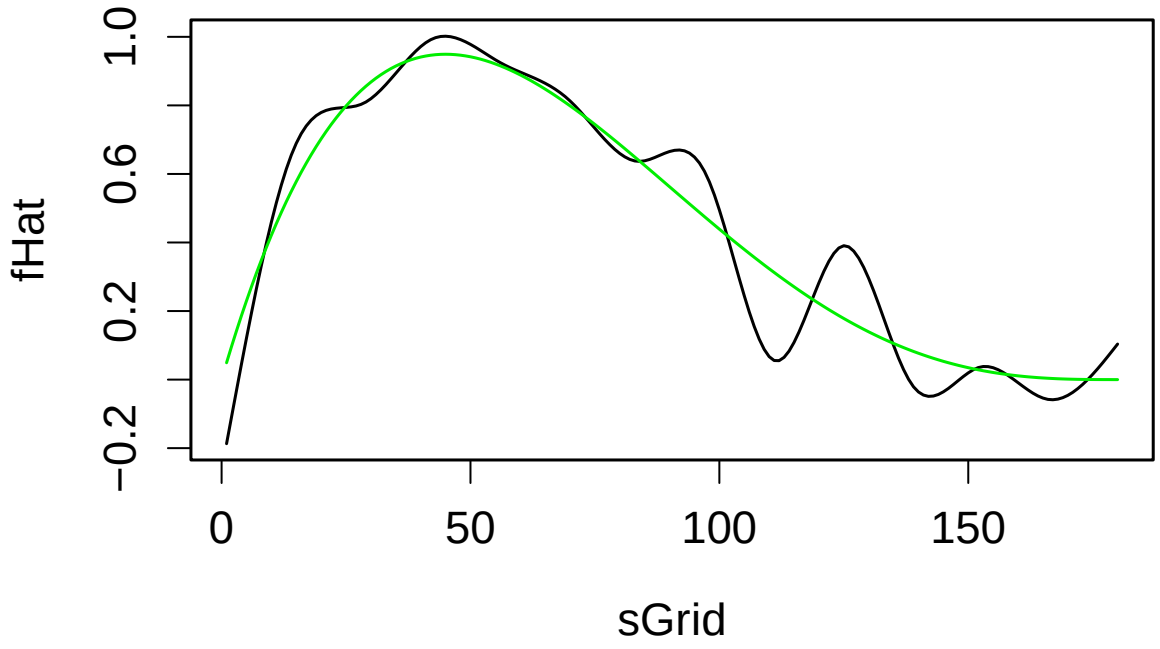

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

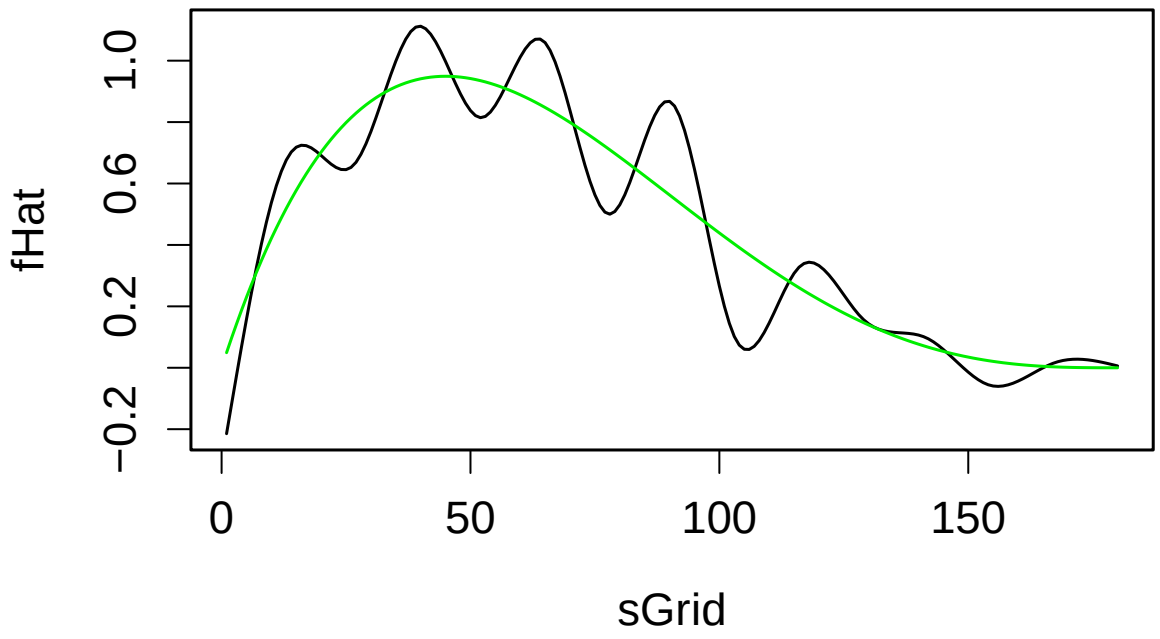

fHat
0.0
0.4
0.8
0
50
100
150
sGrid


fHat
0.0
0.4
0.8
0
50
100
150
sGrid


fHat
0.0
0.4
0.8
0
50
100
150
sGrid

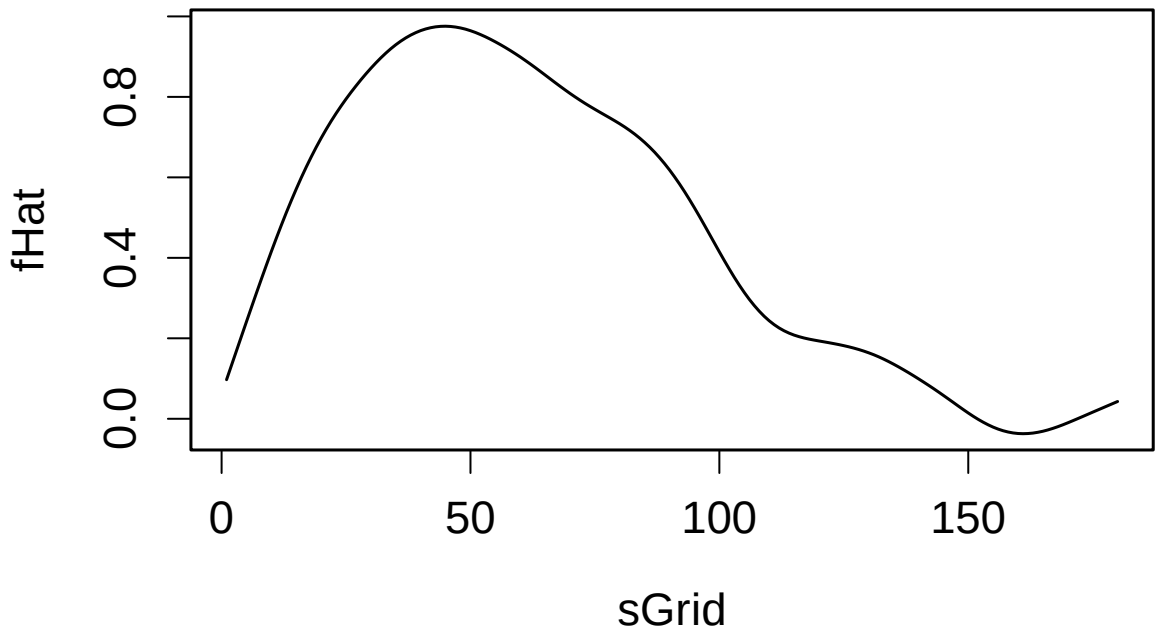

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

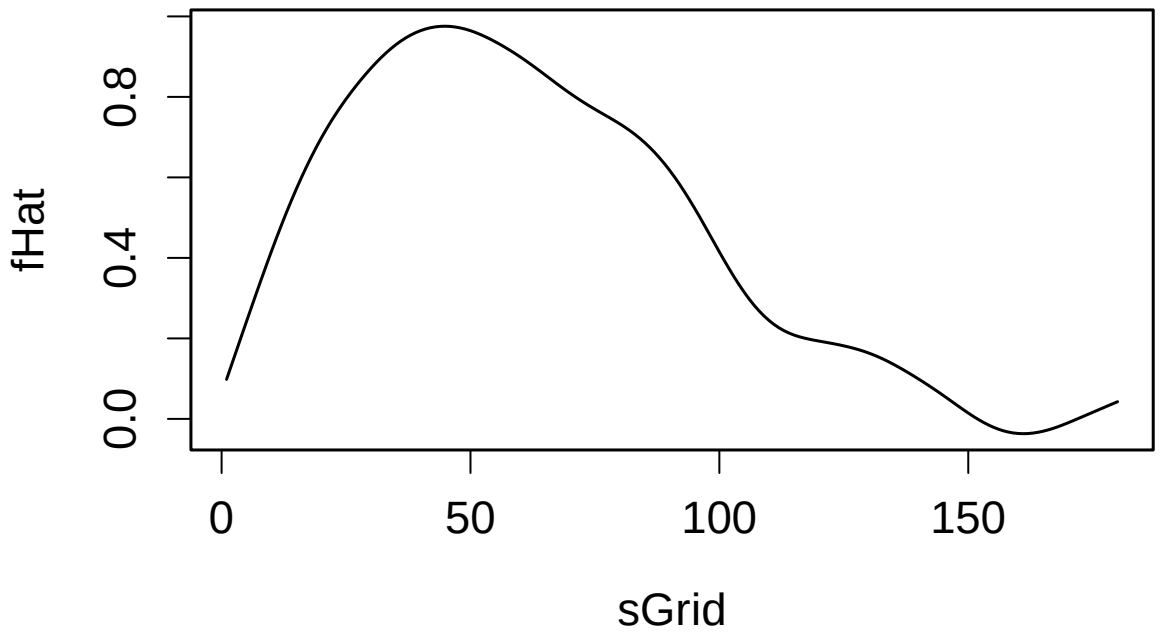

fHat
1.0
0.6
0.2
−0.2
0
50
100
150
sGrid

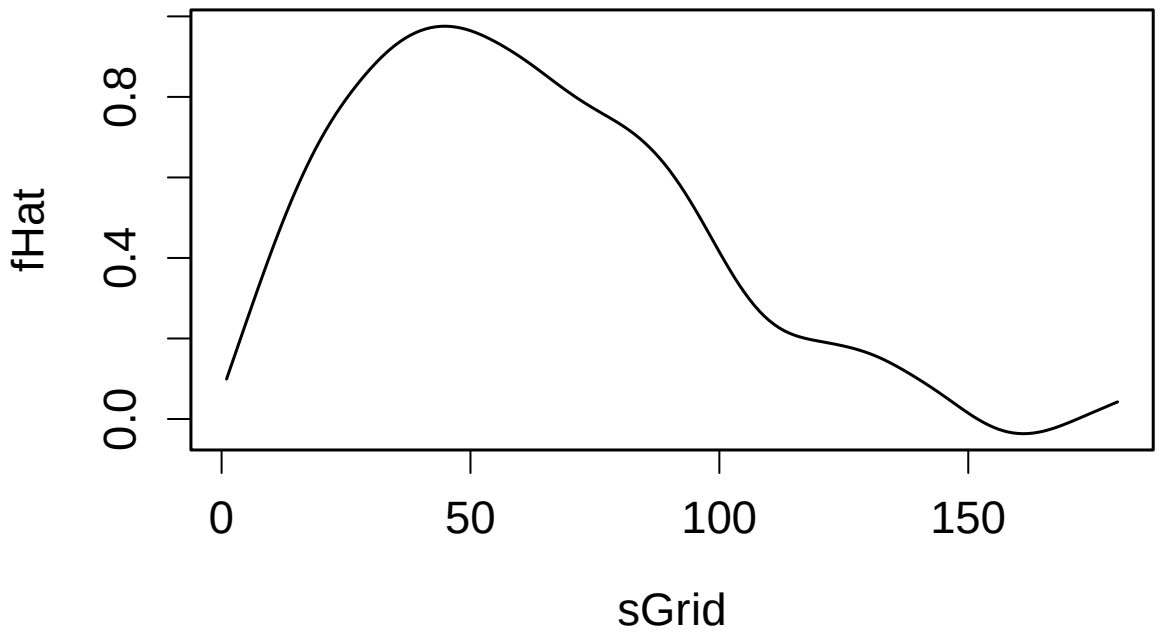


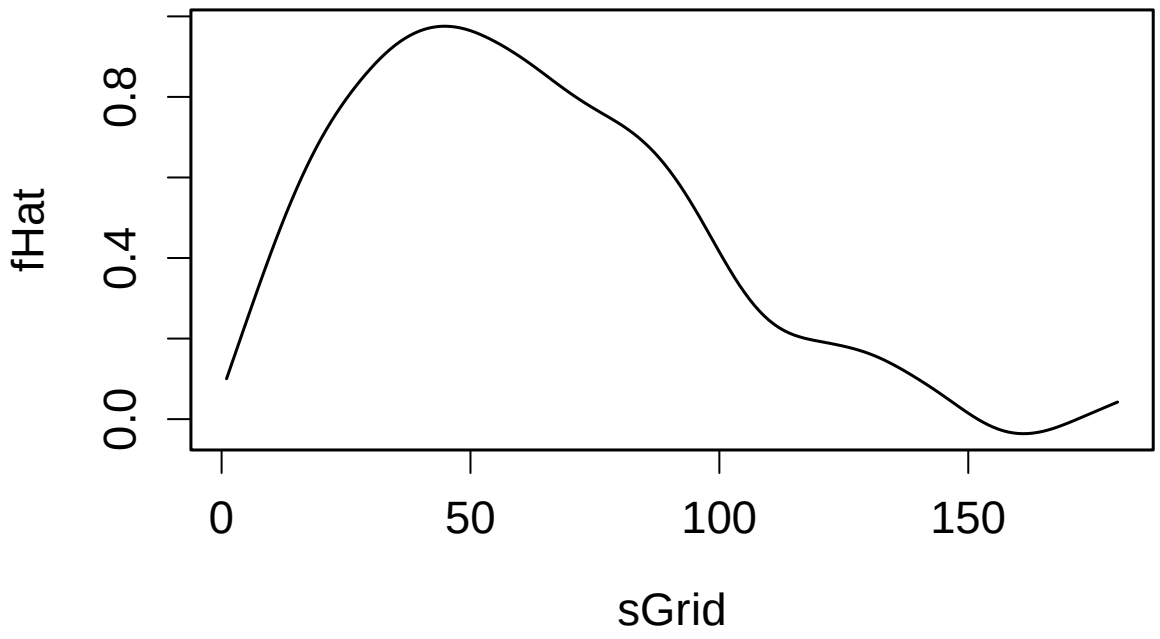
Task 3 For the smoothing spline model use 10/90 cross validation to estimate lambda.

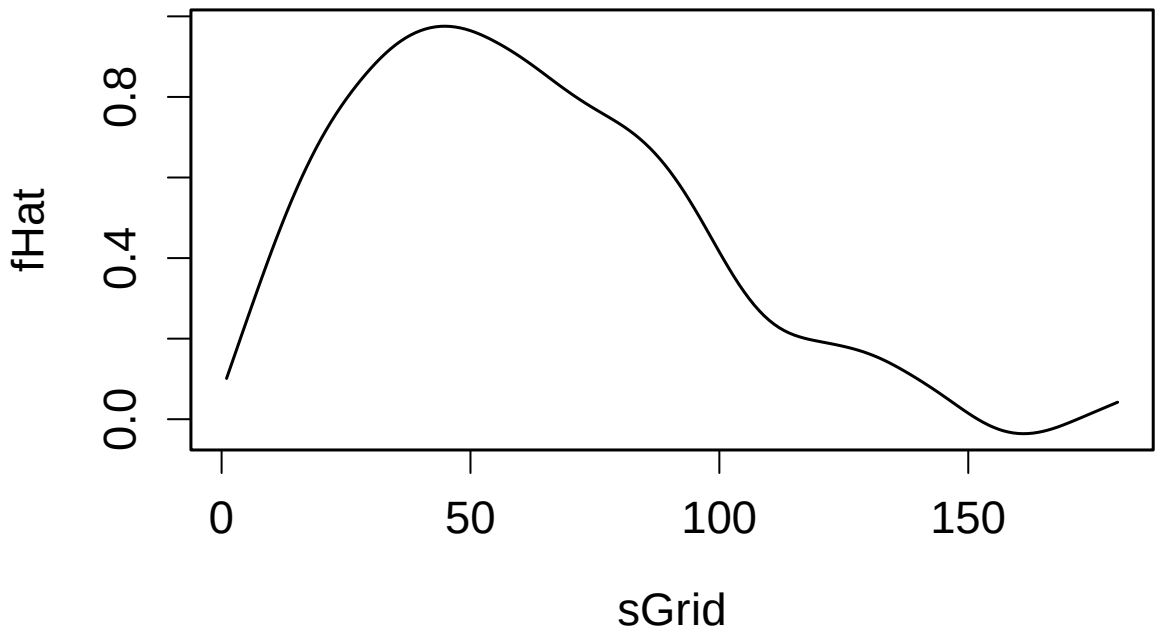
```
val= 50:150
for (x in val){
fHat<- fitWSS(z,W, x ,sGrid)
fields.style()
plot( sGrid, fHat, type="l")
}
```

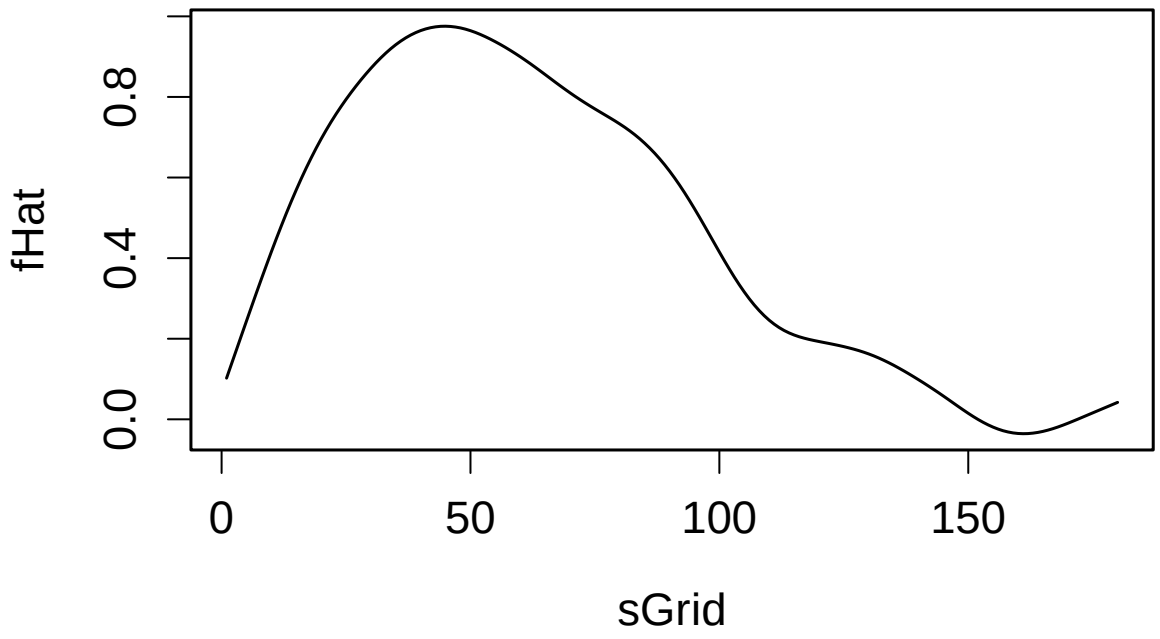

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

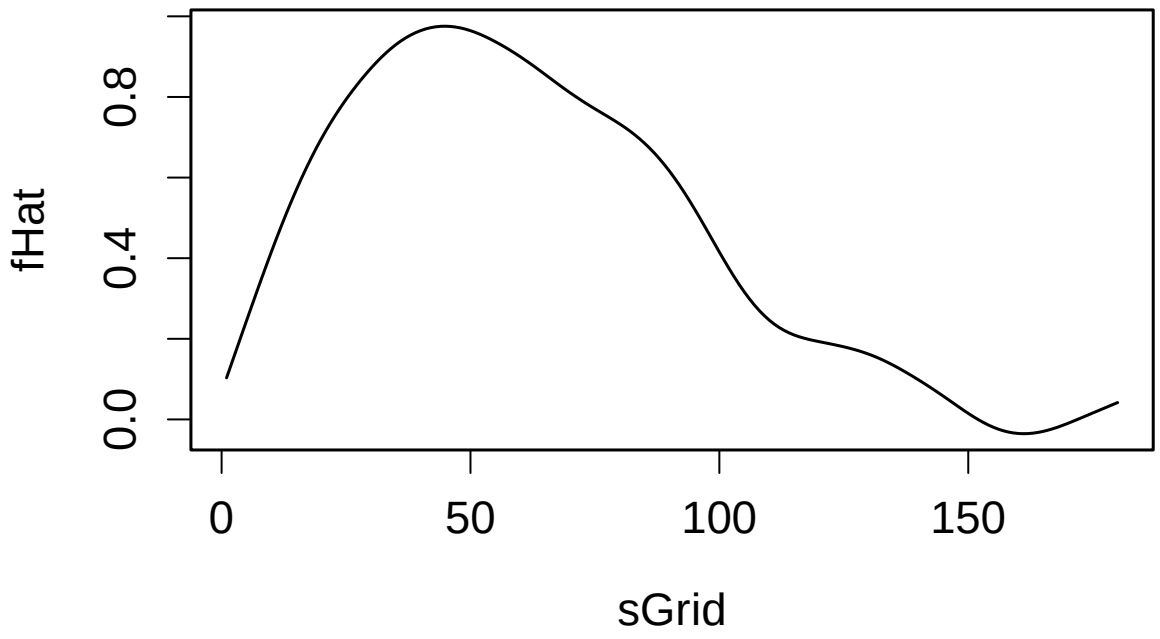

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

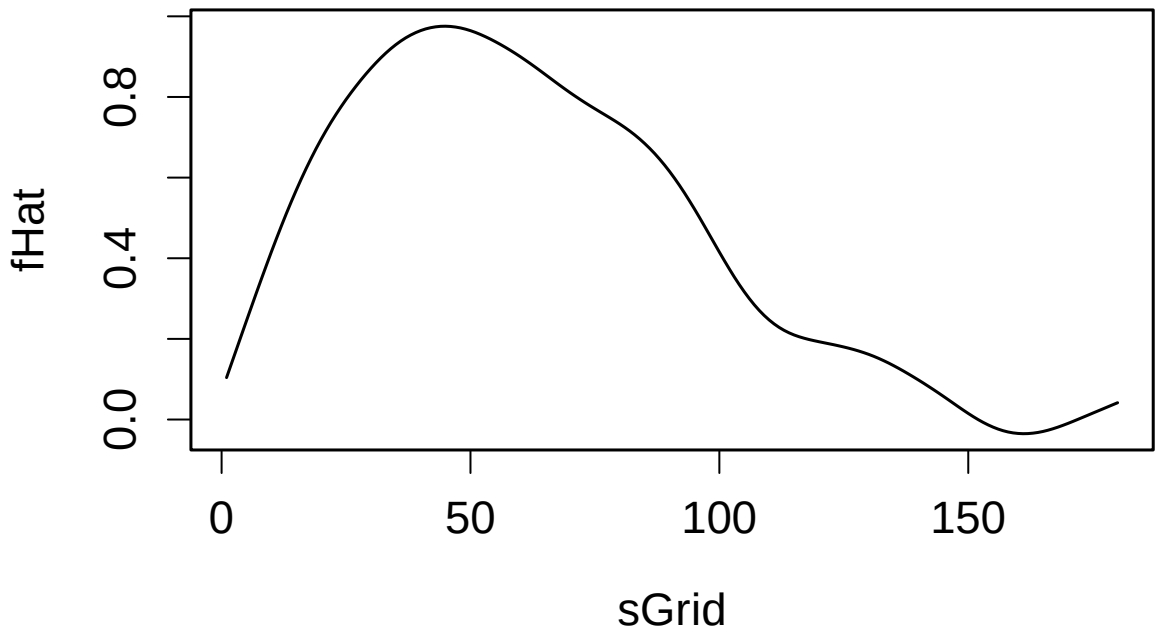

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

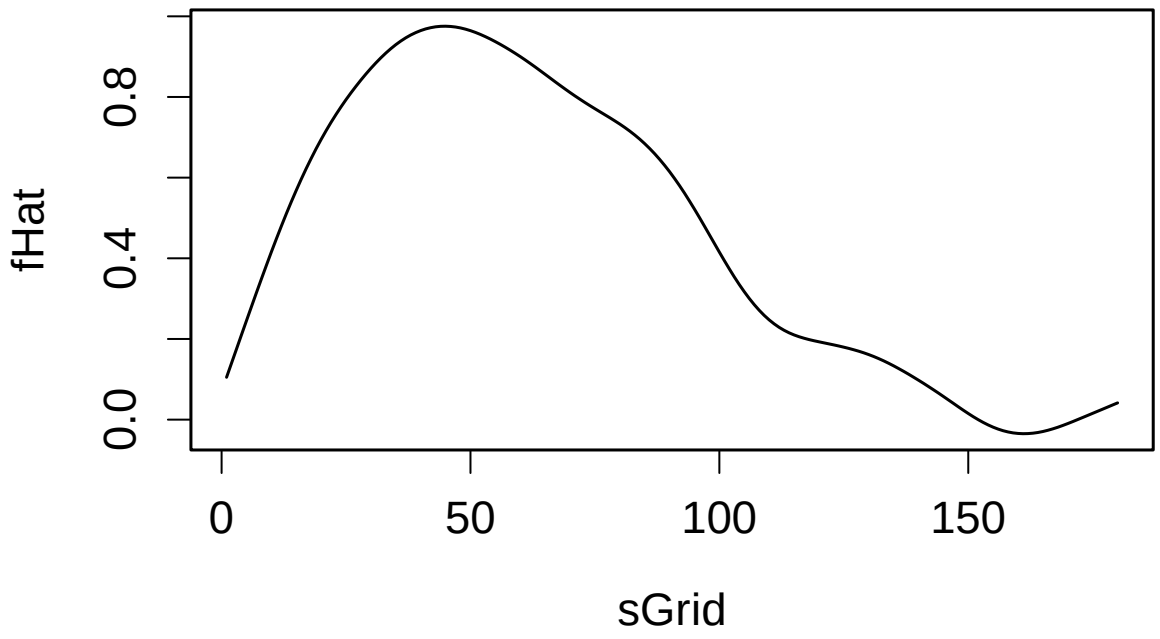

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

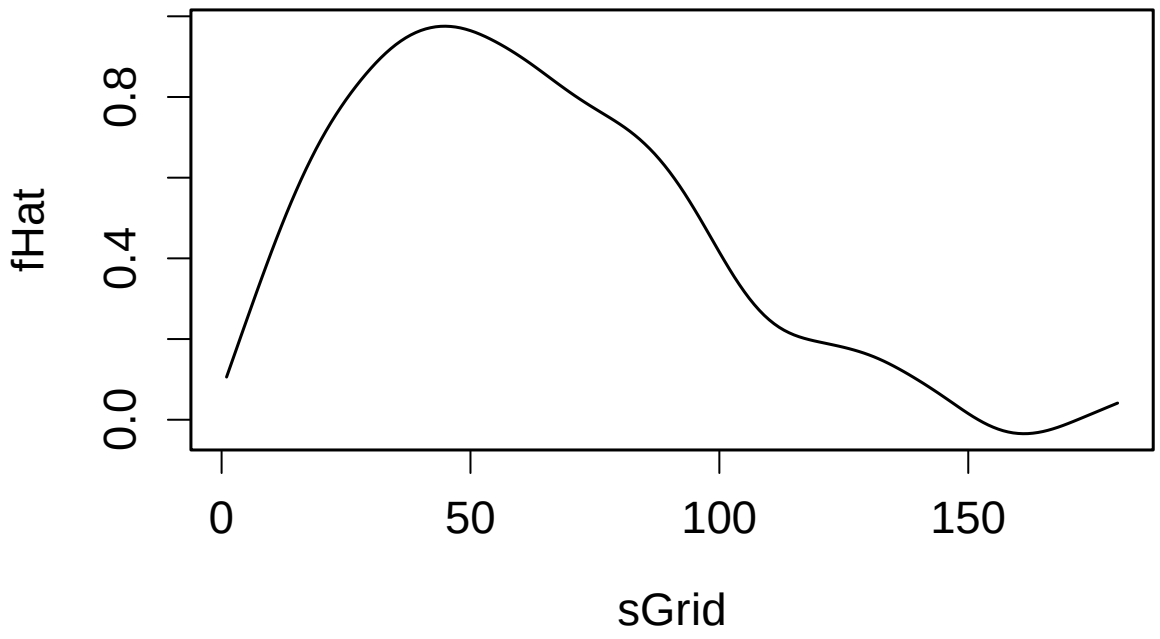

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

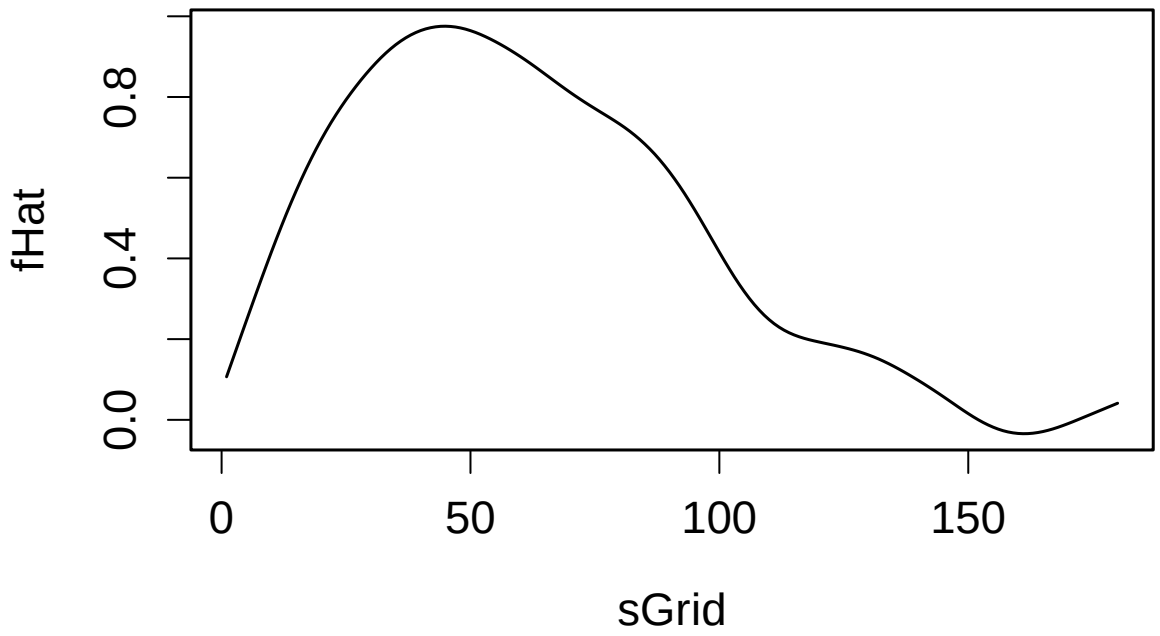

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

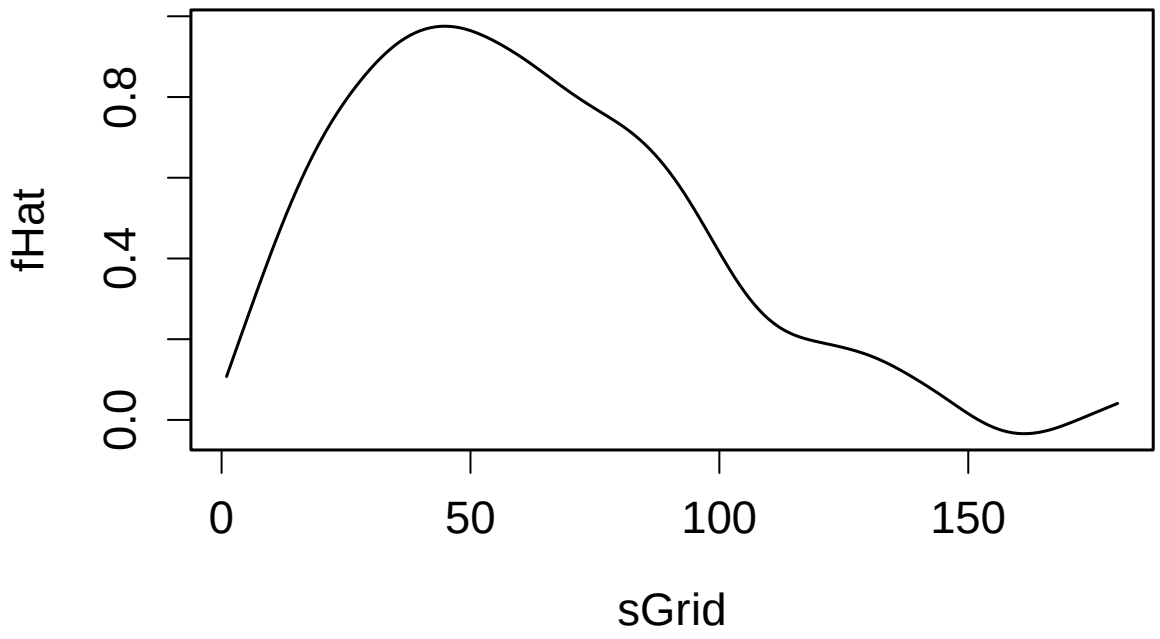

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

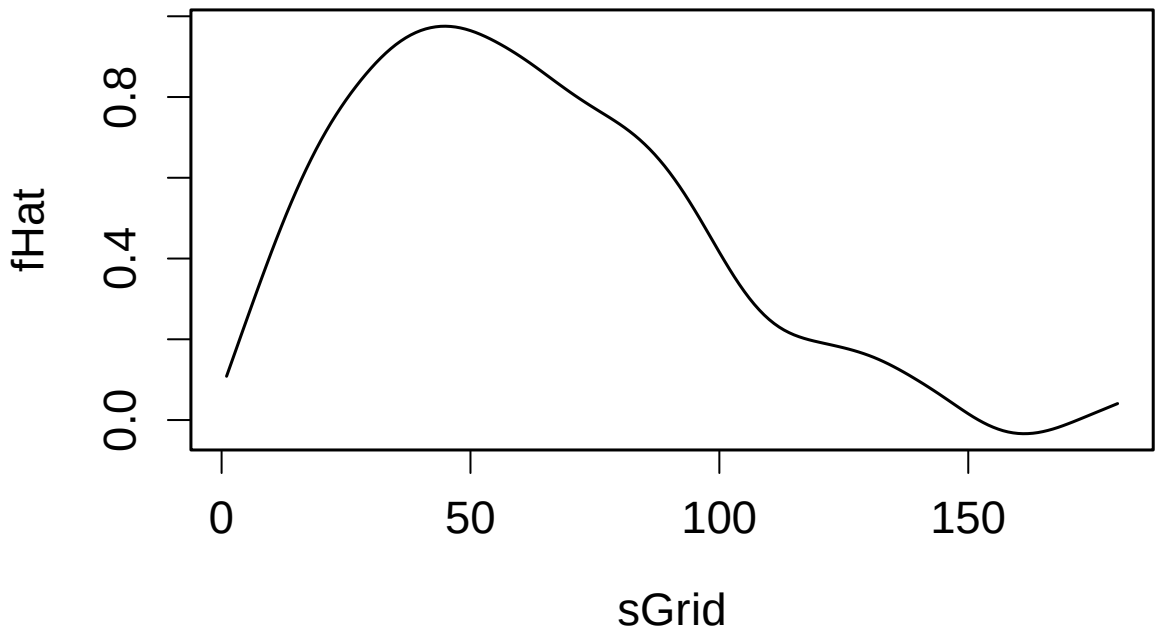

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

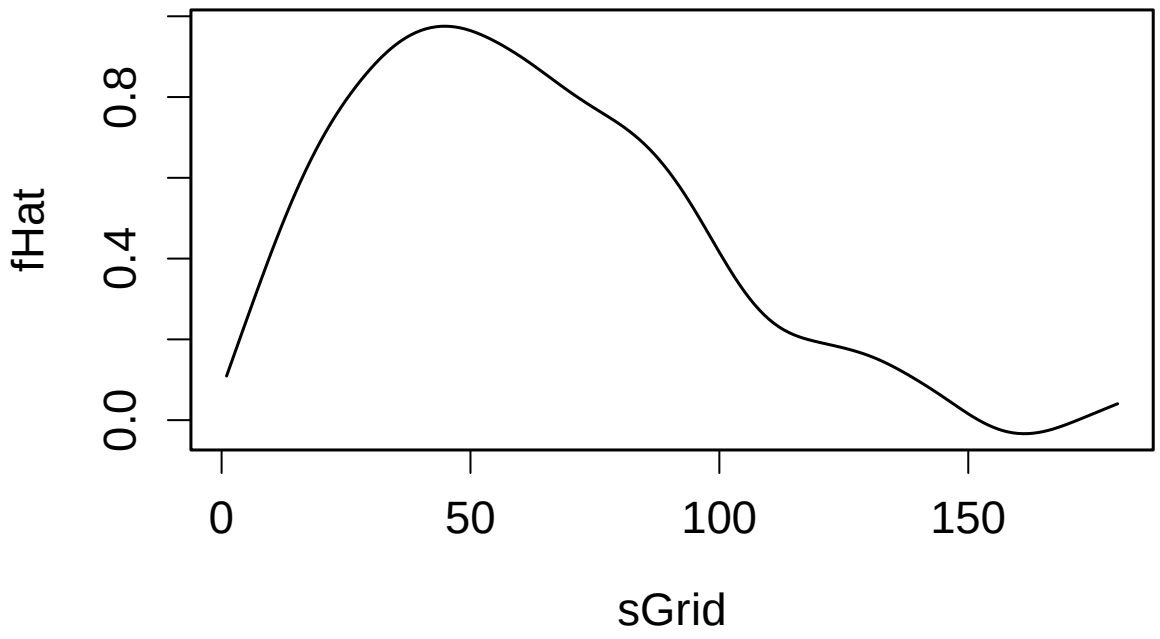

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

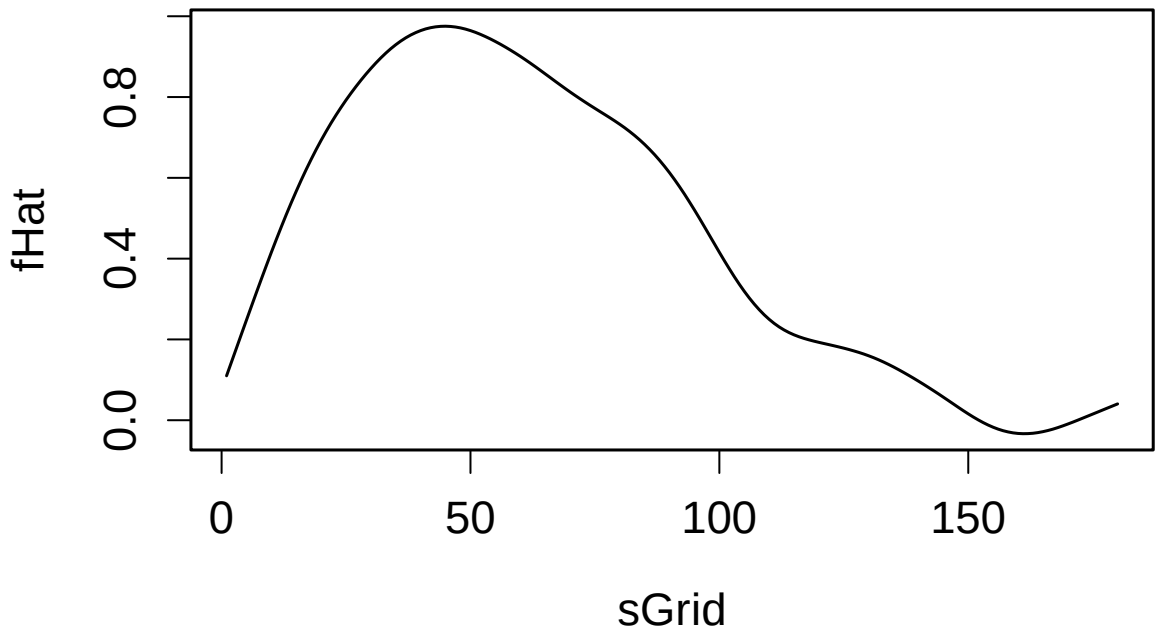

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

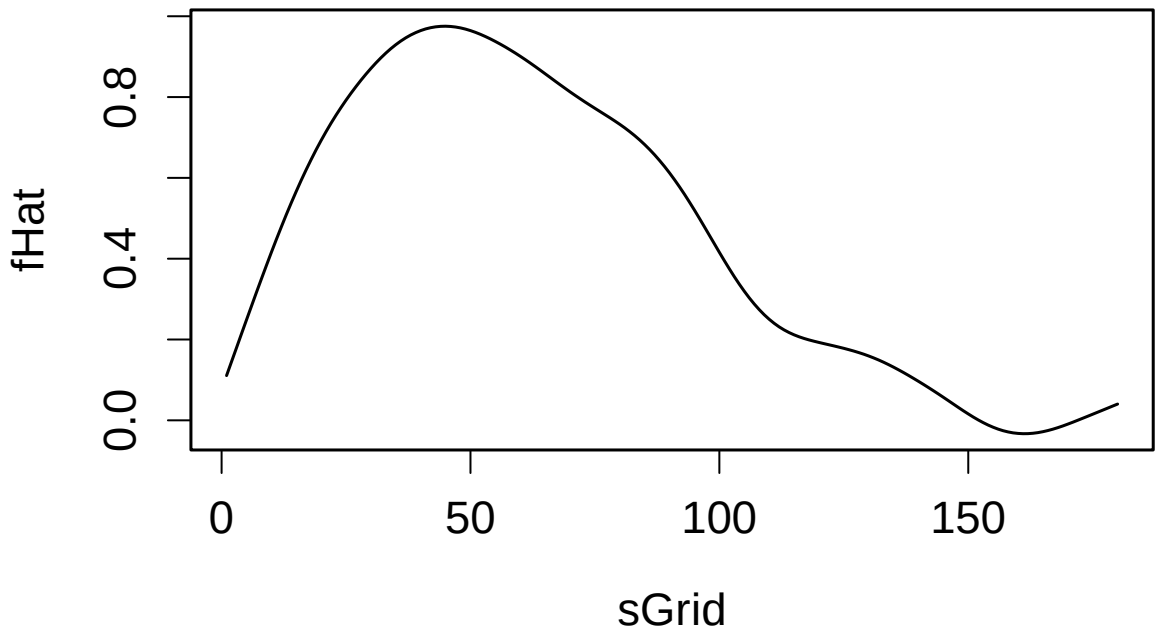

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

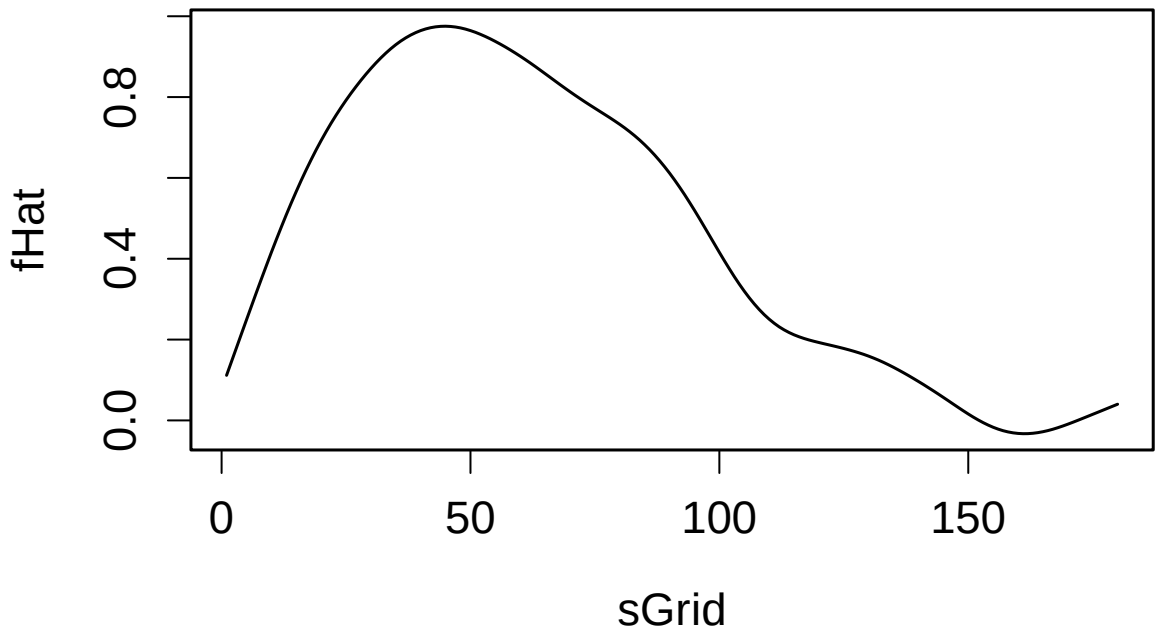

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

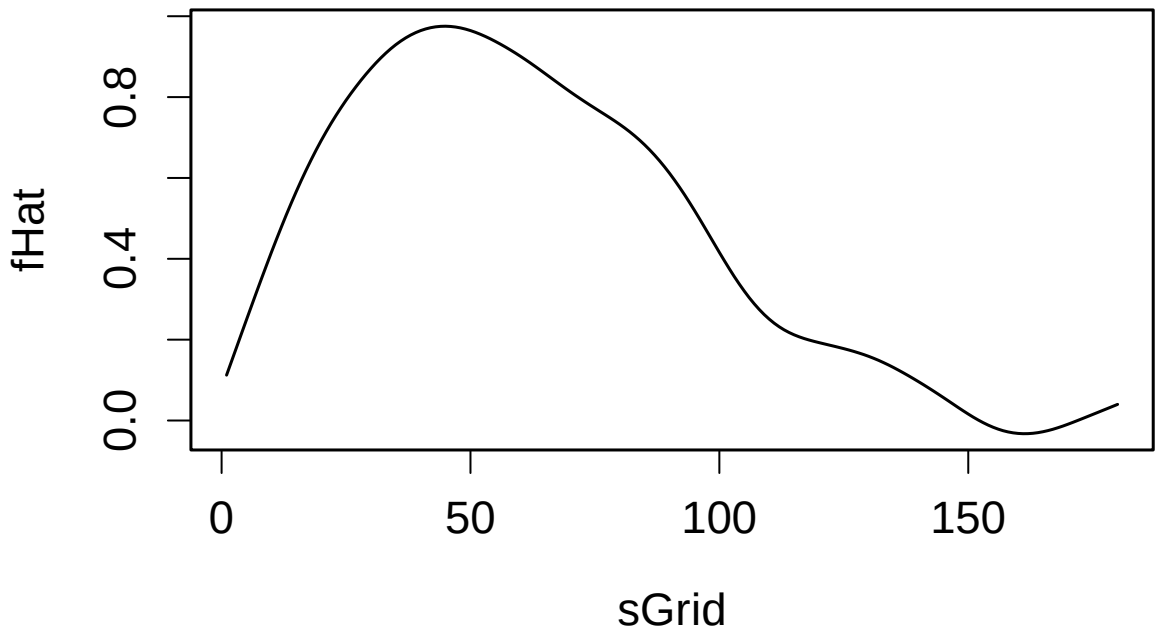

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

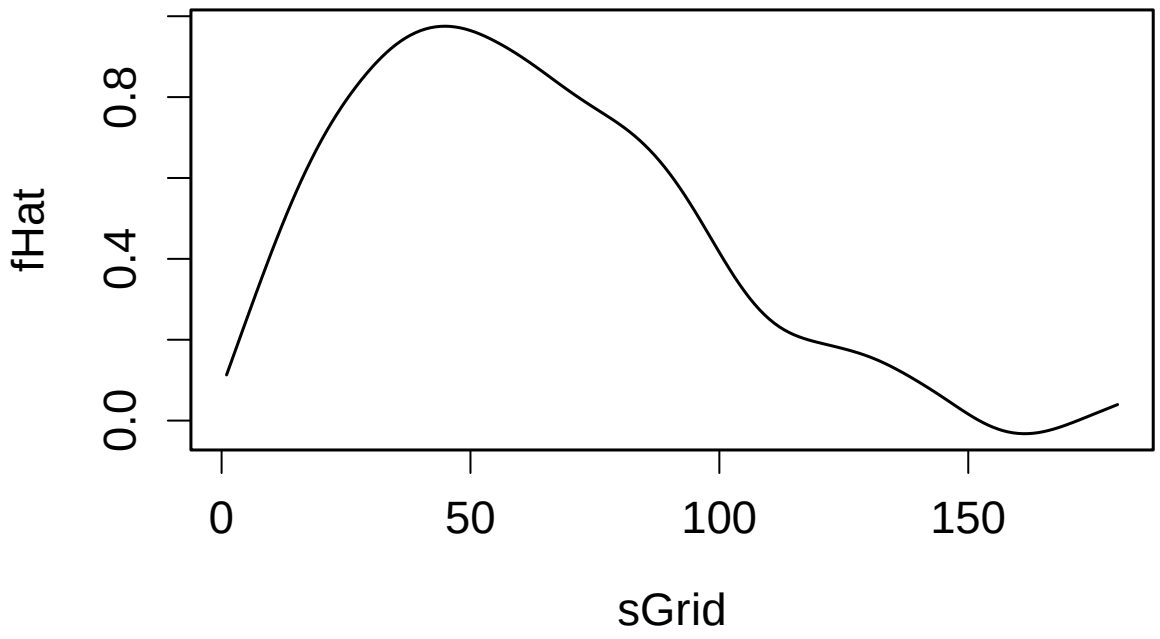

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

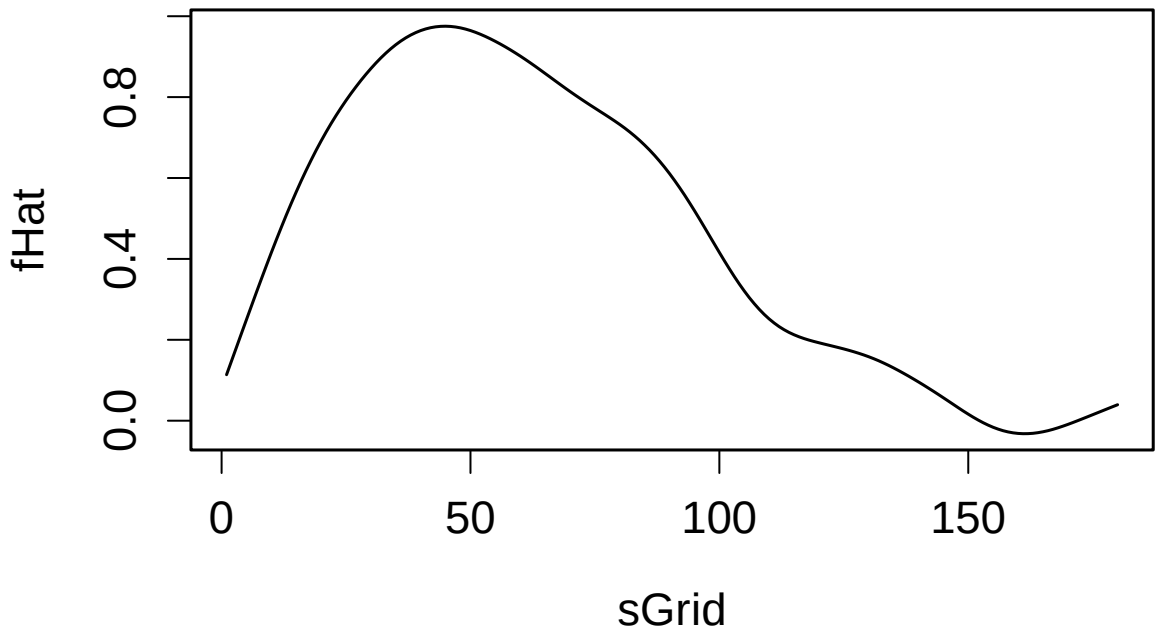

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

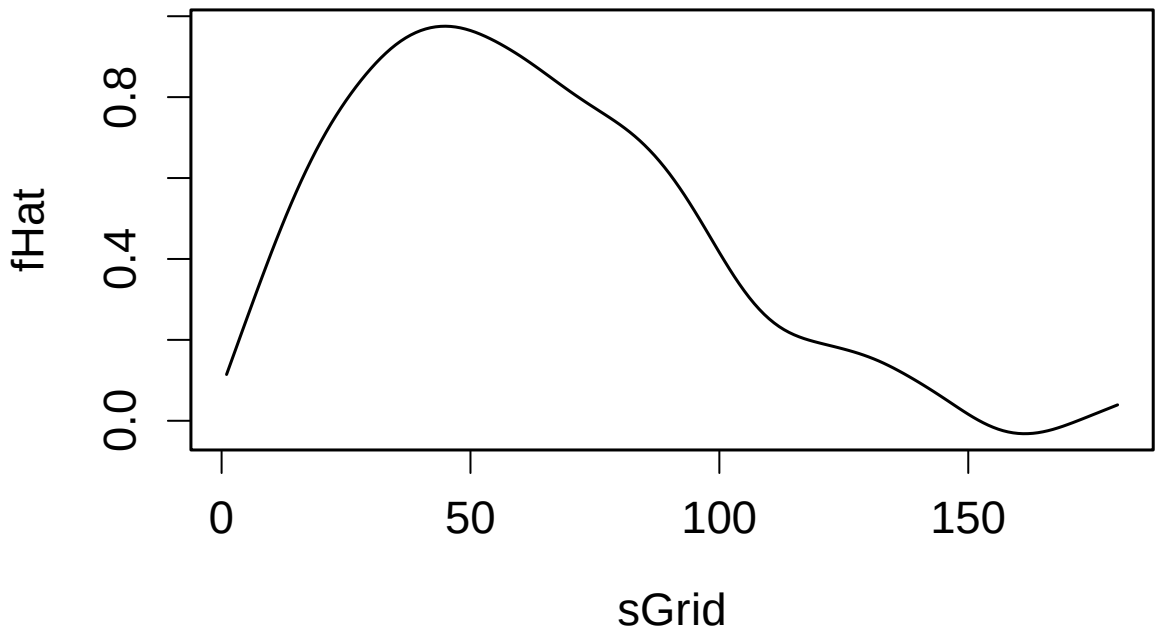

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

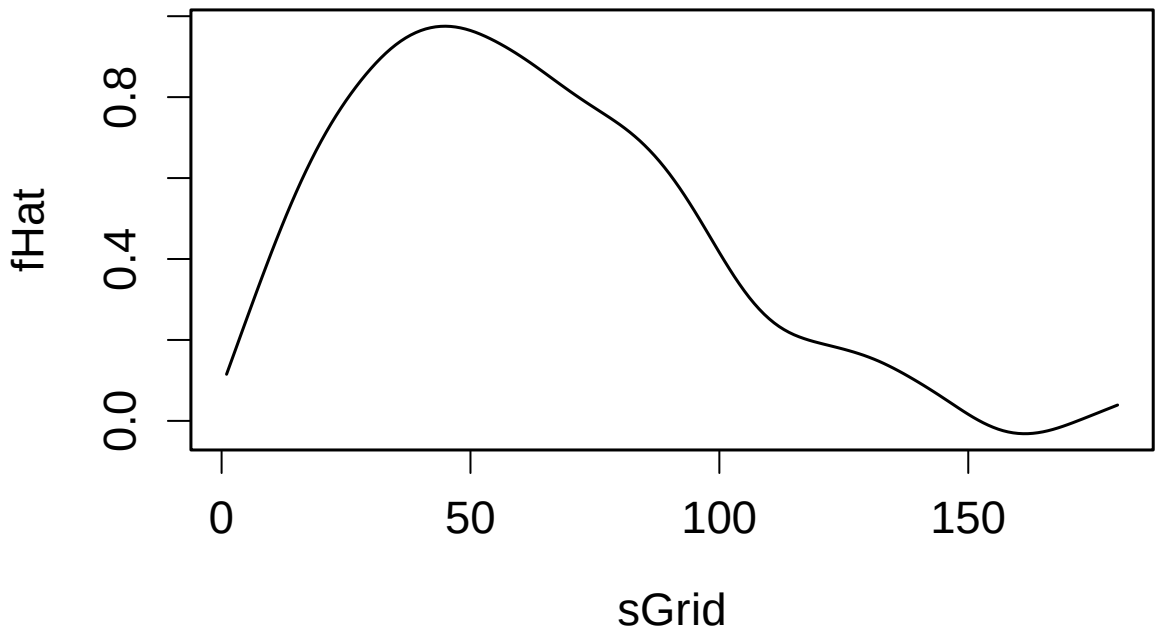

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

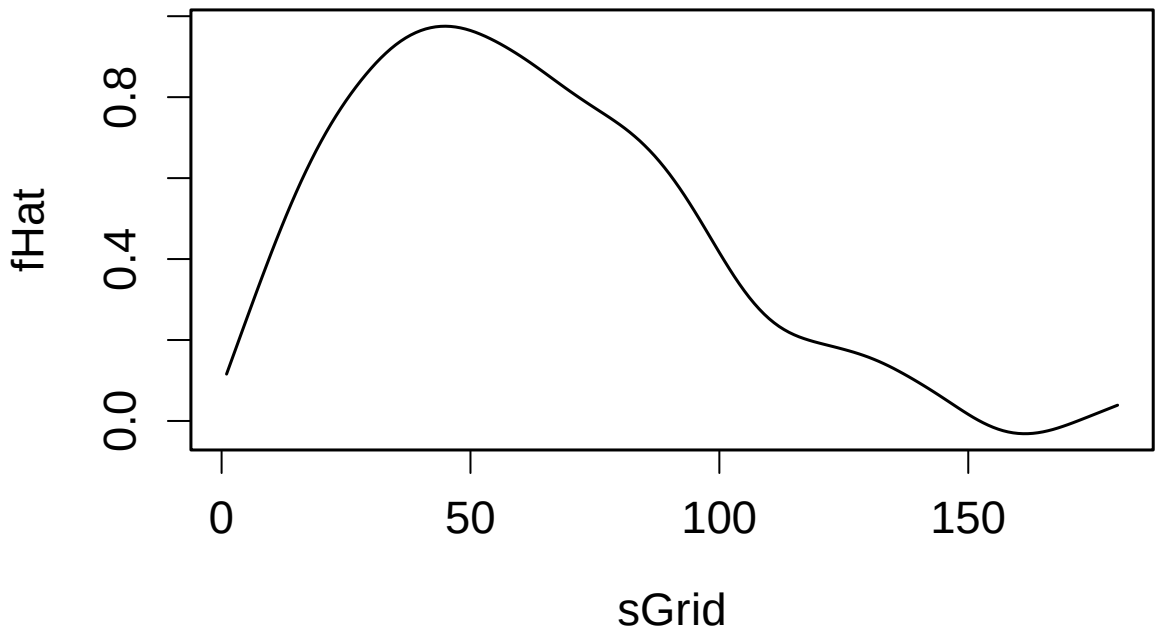

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

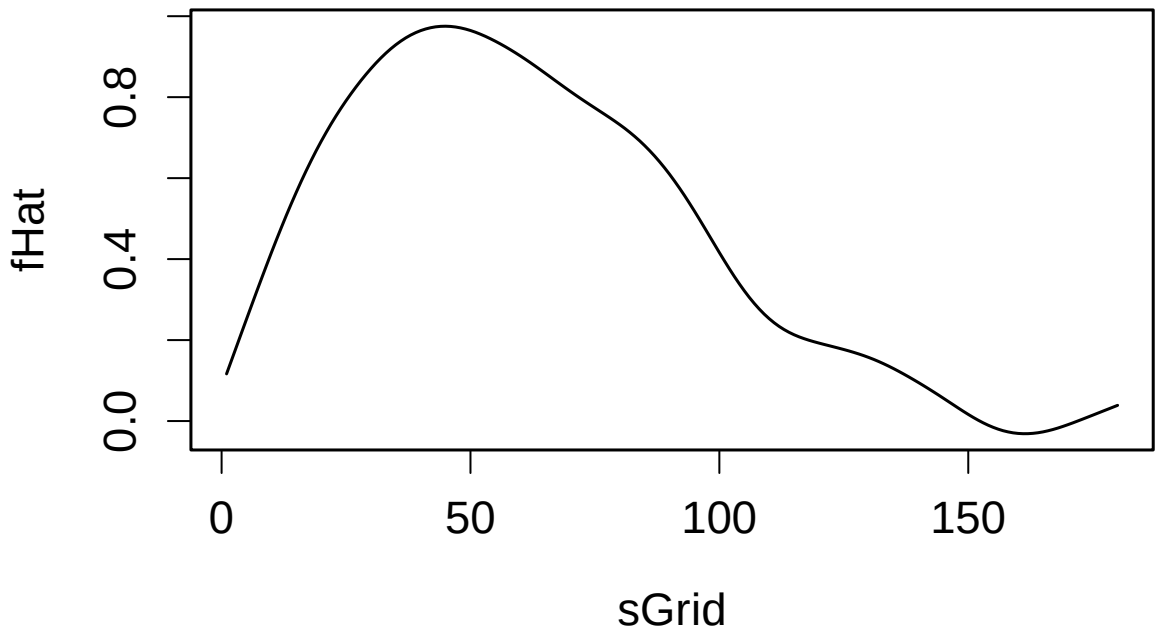

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

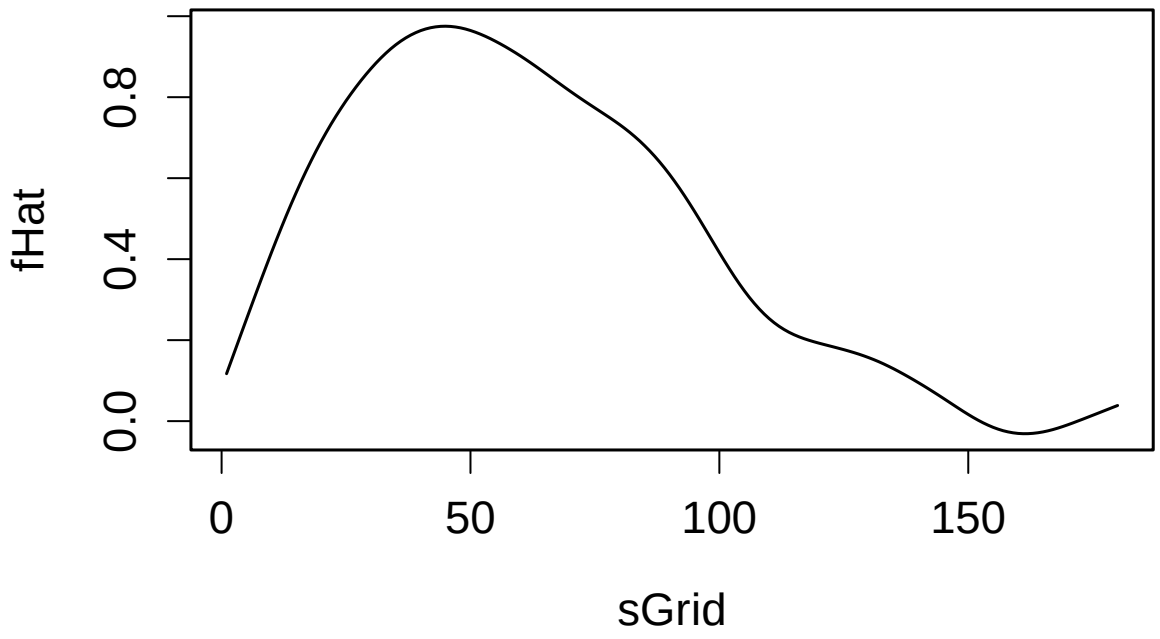

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

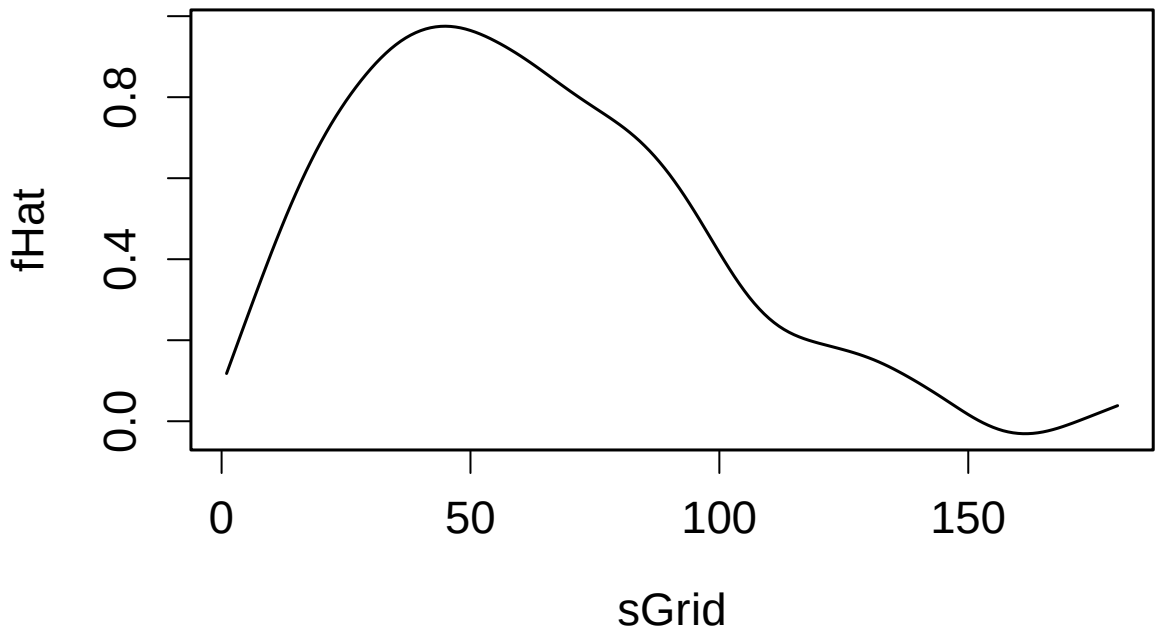

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

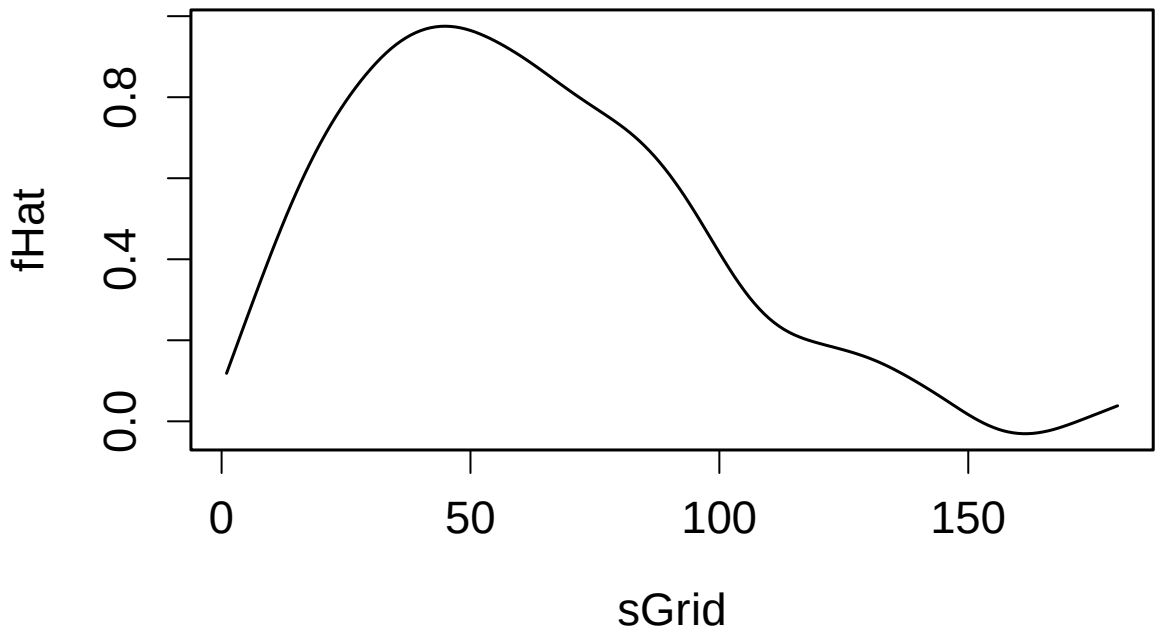

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

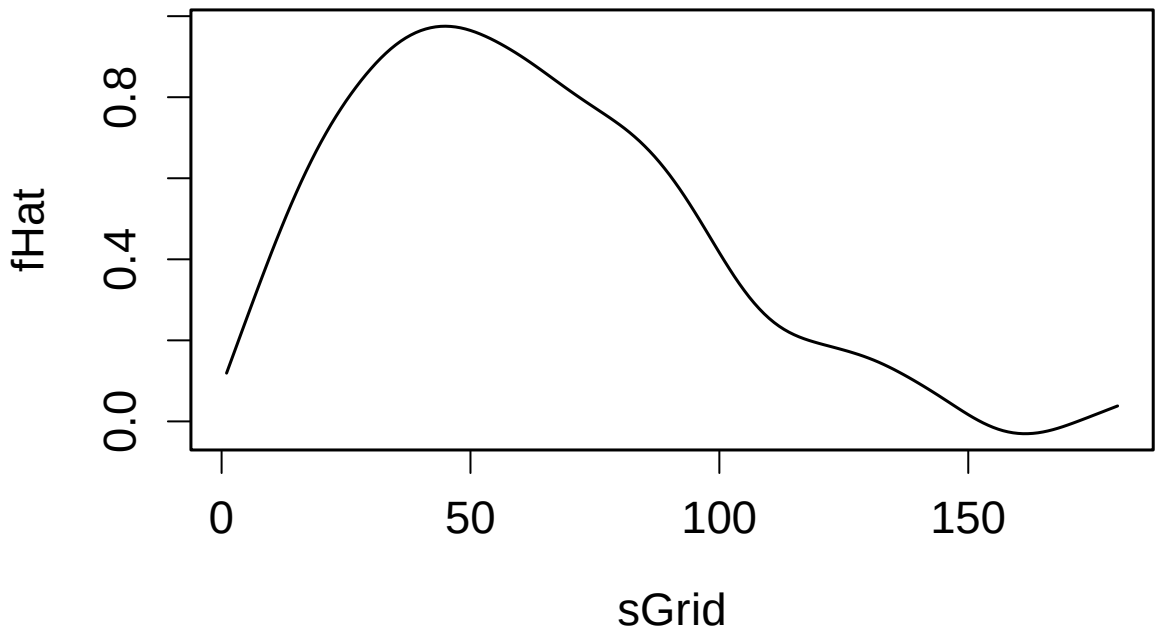

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

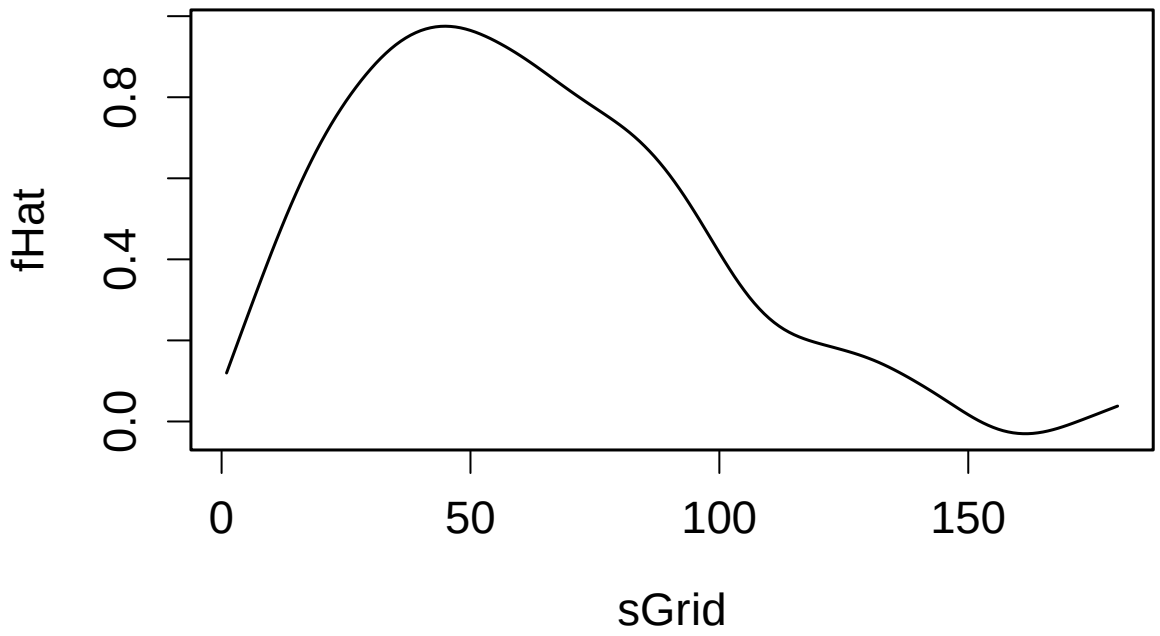

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

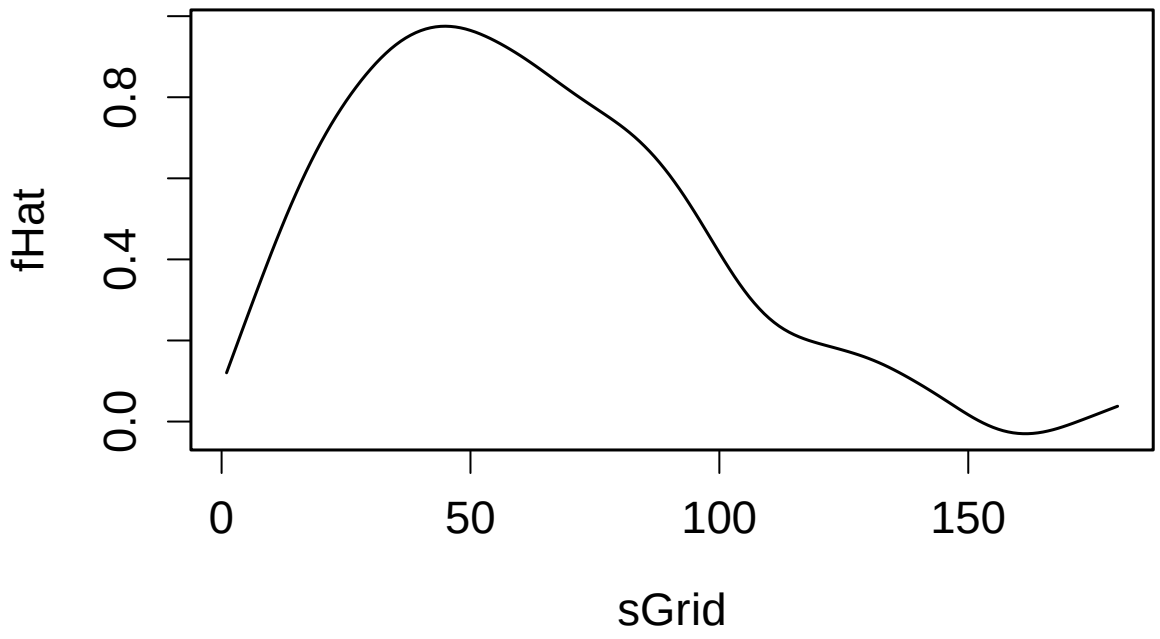

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

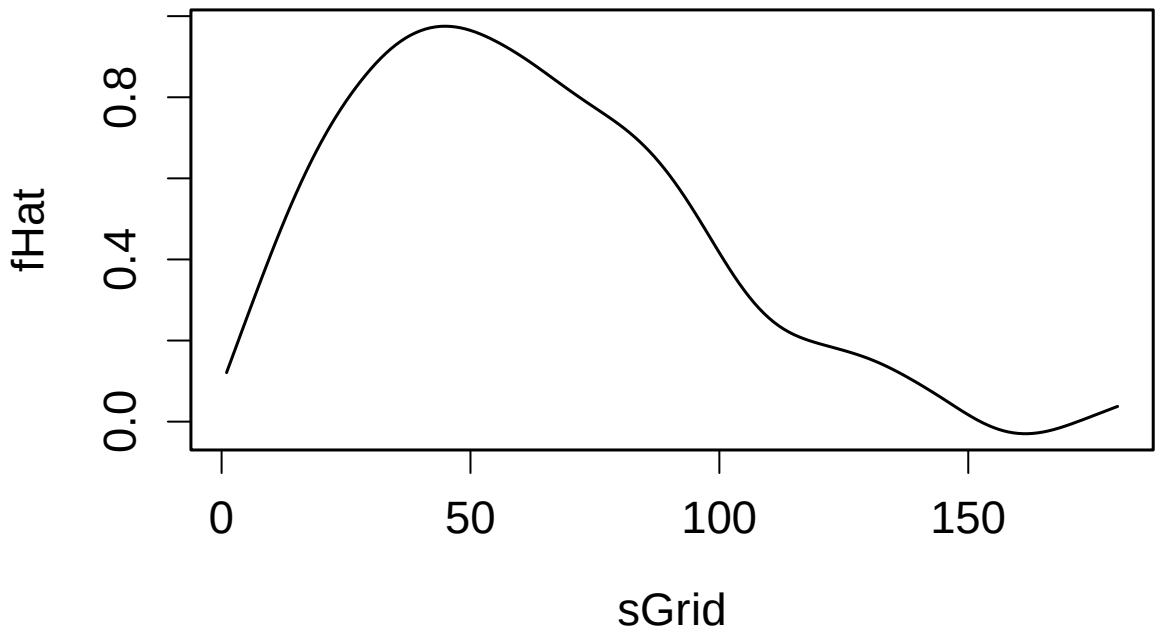

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

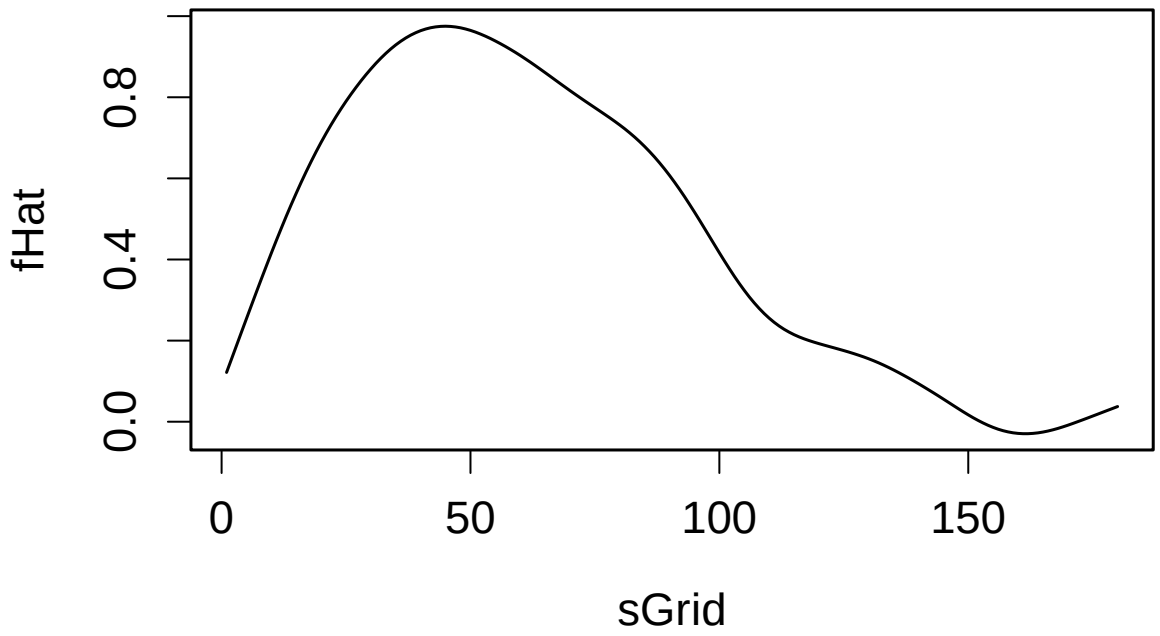

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

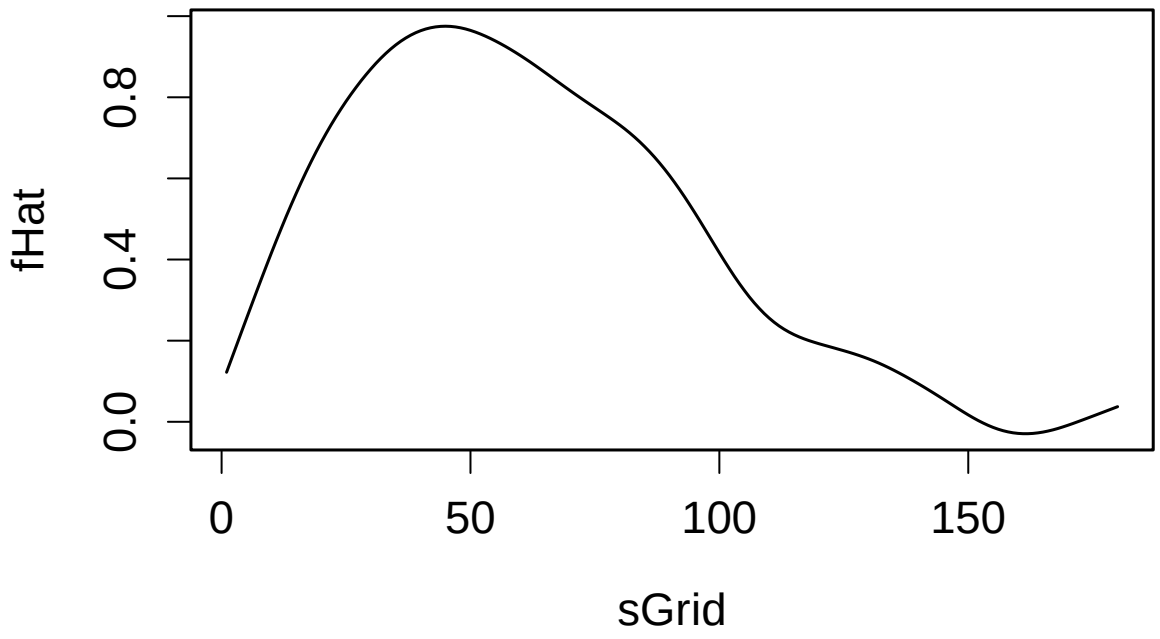

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

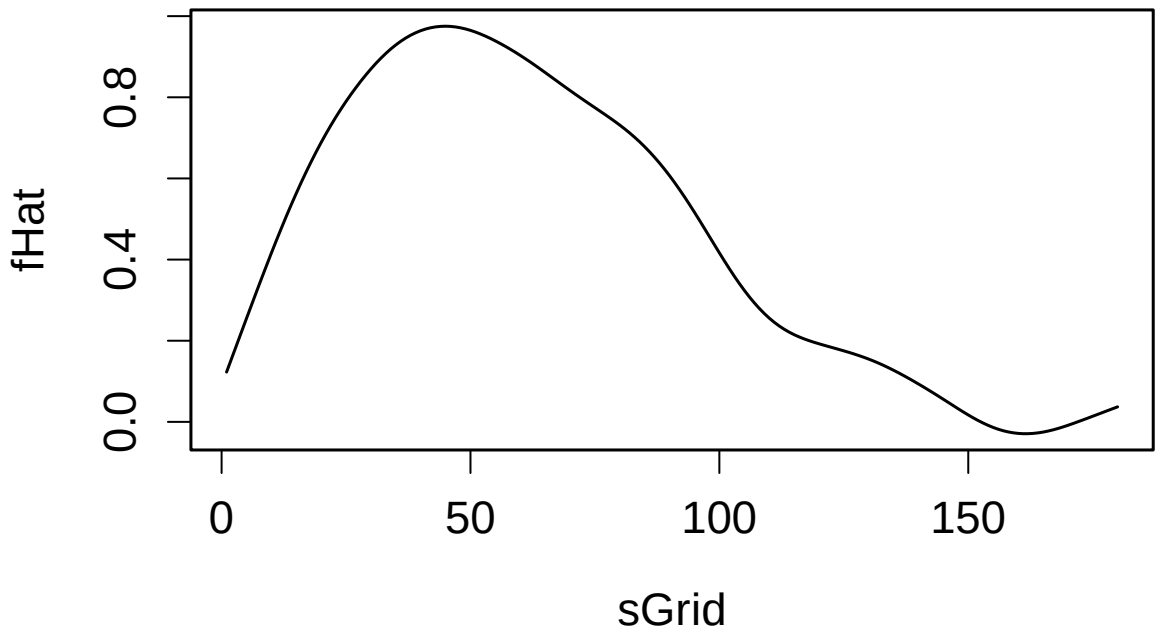

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

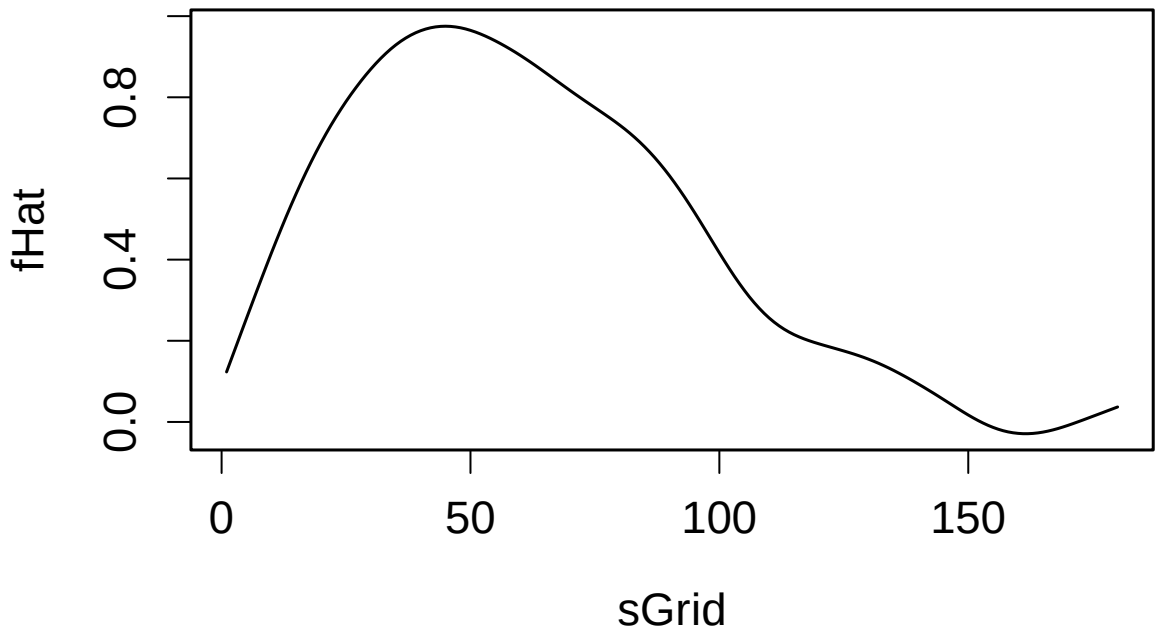

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

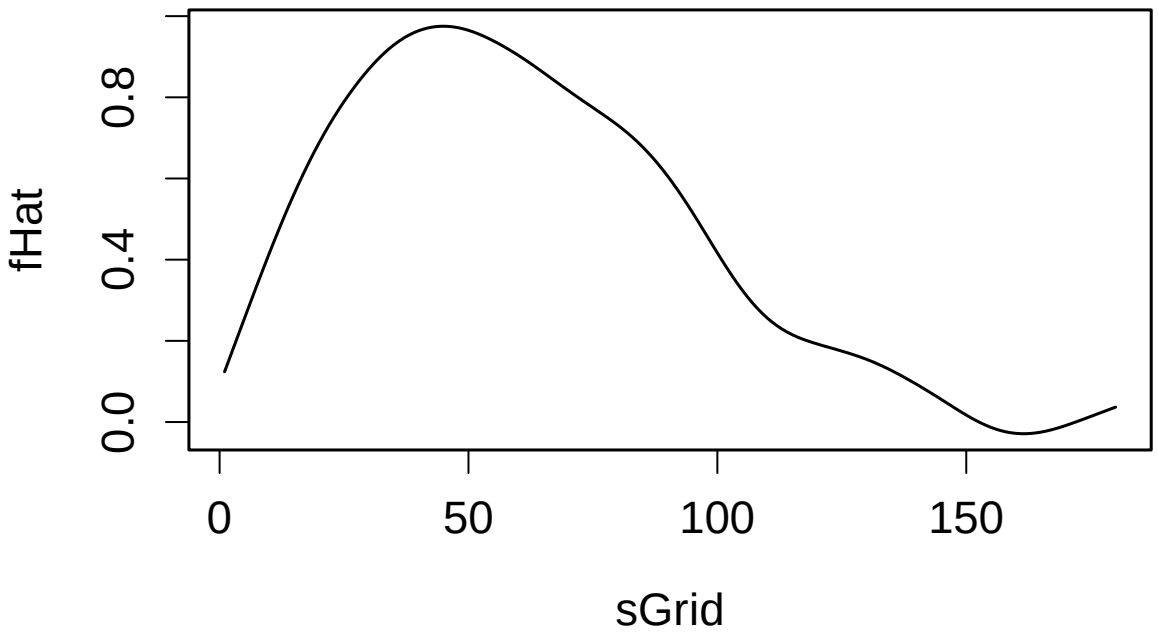

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

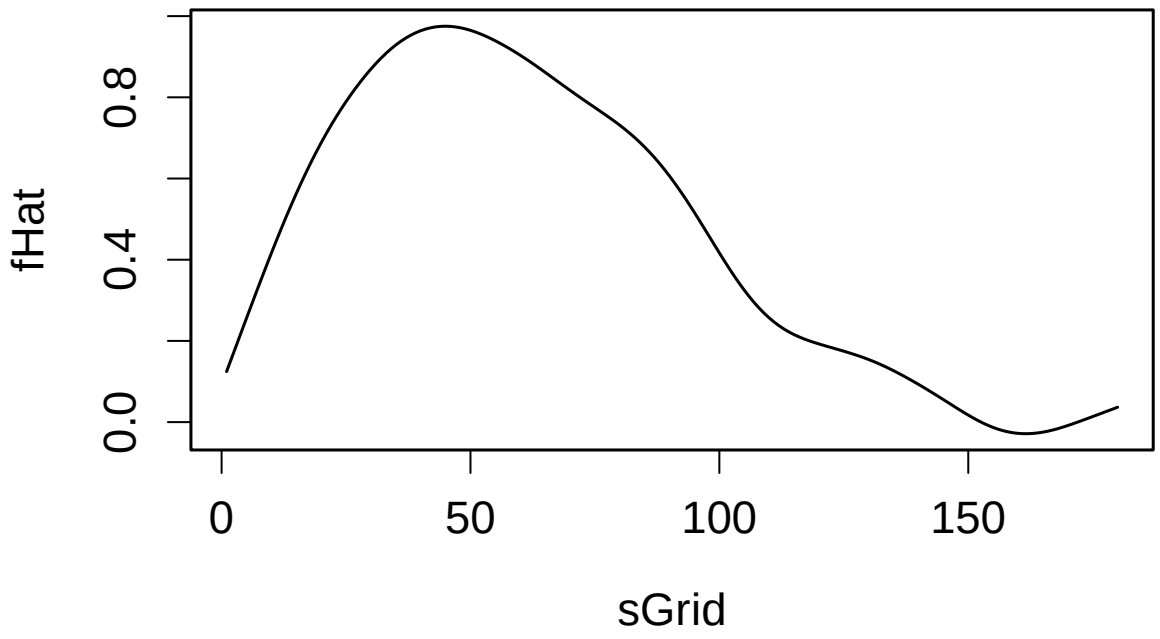

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

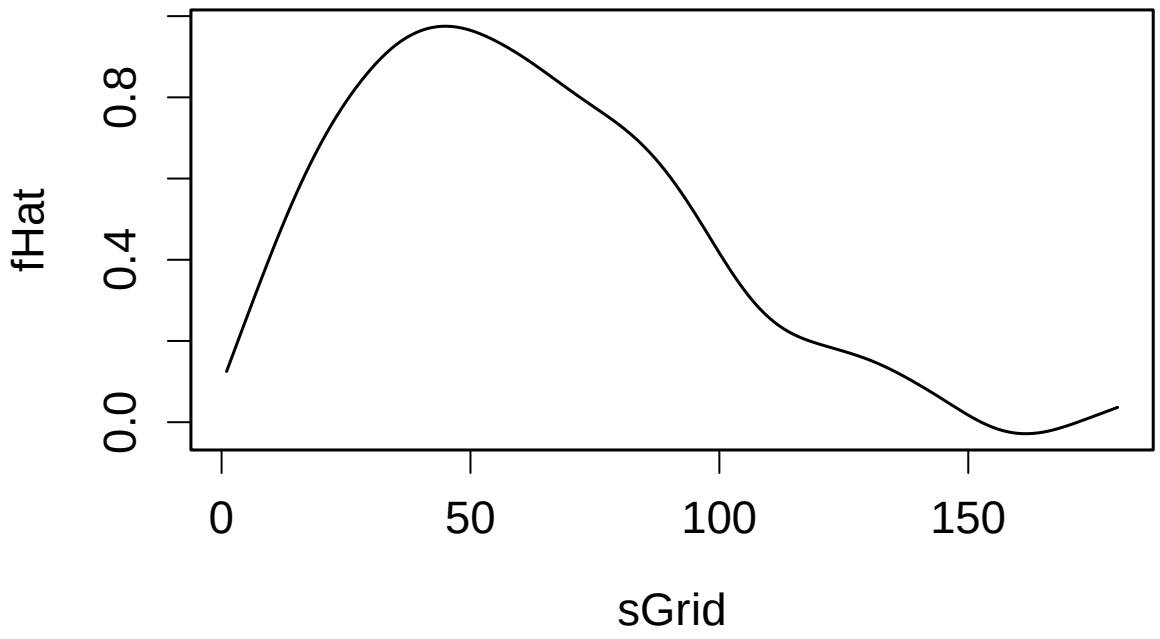

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

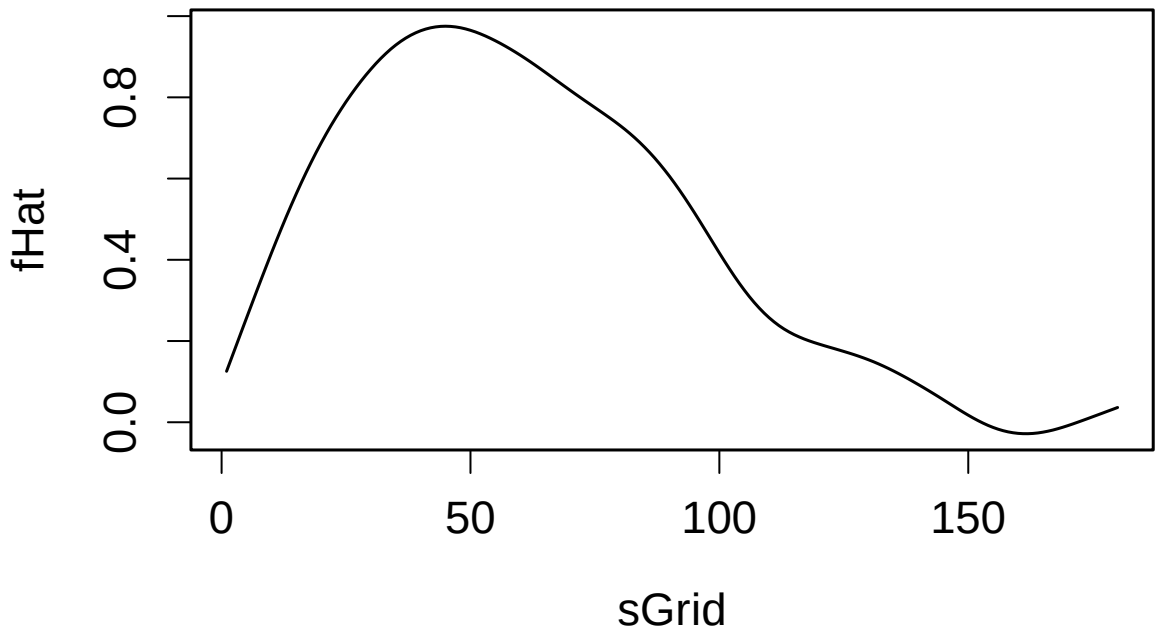

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

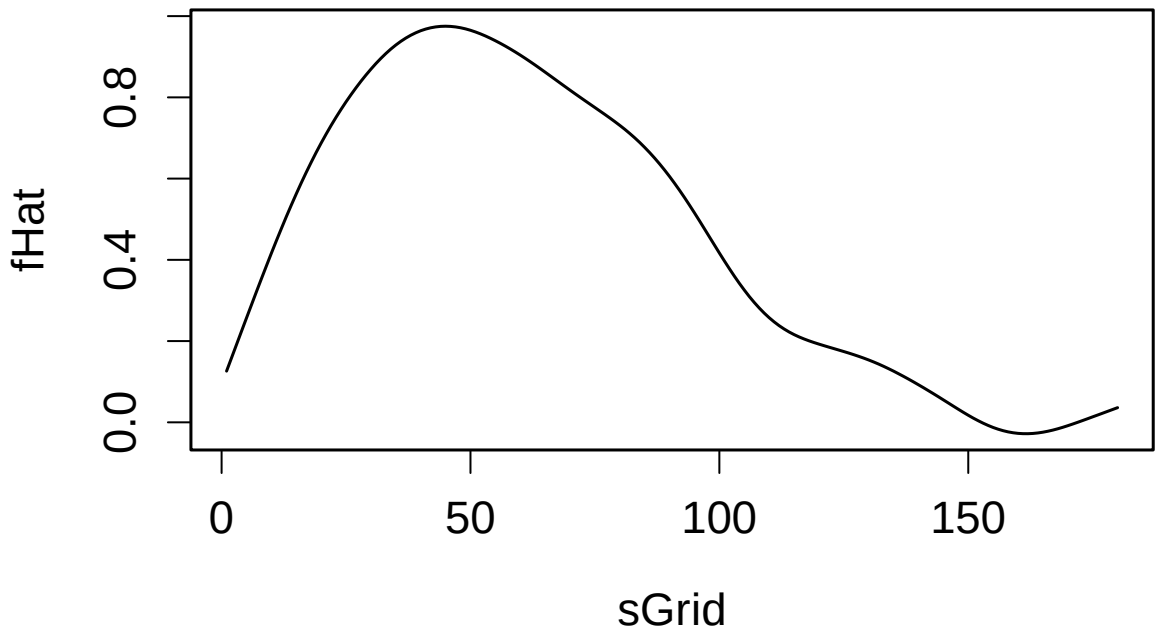

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

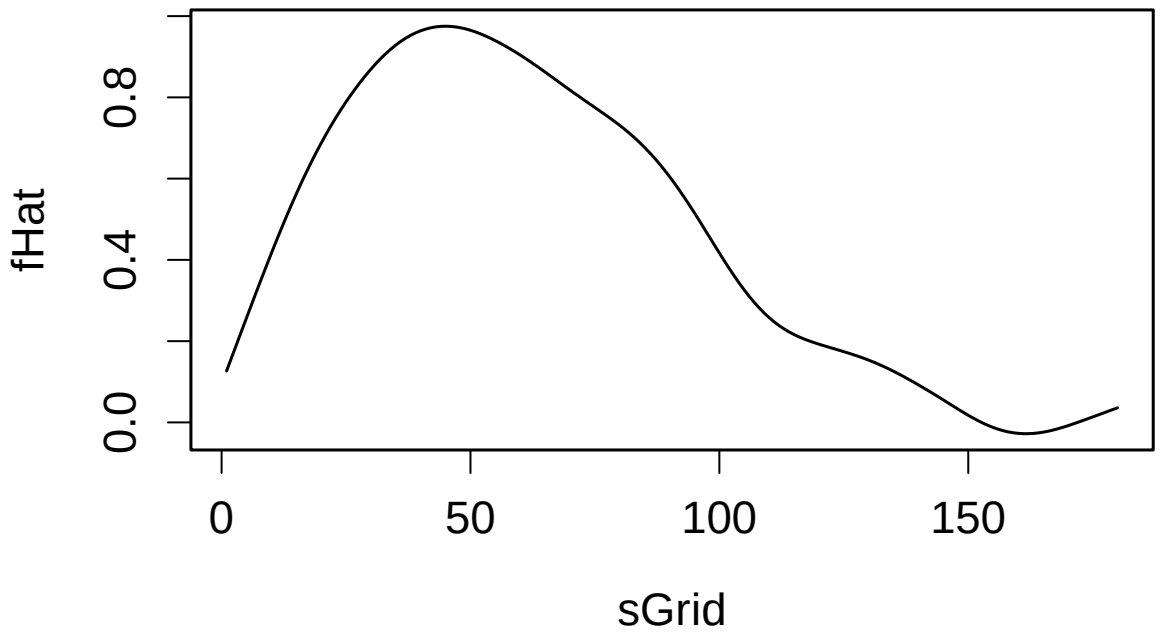

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

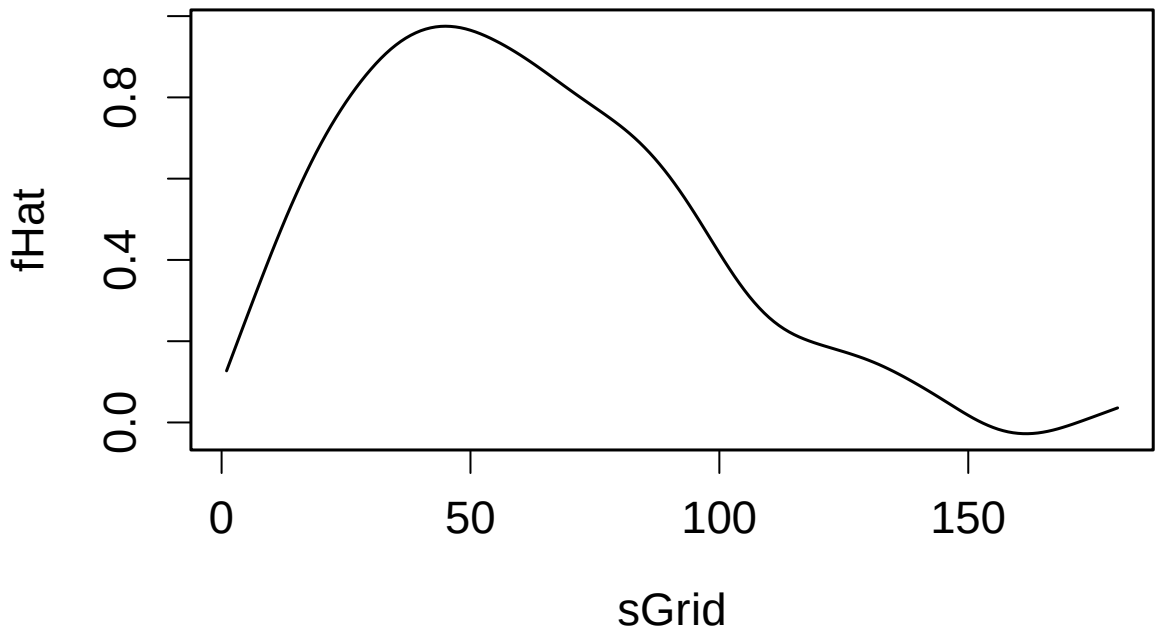

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

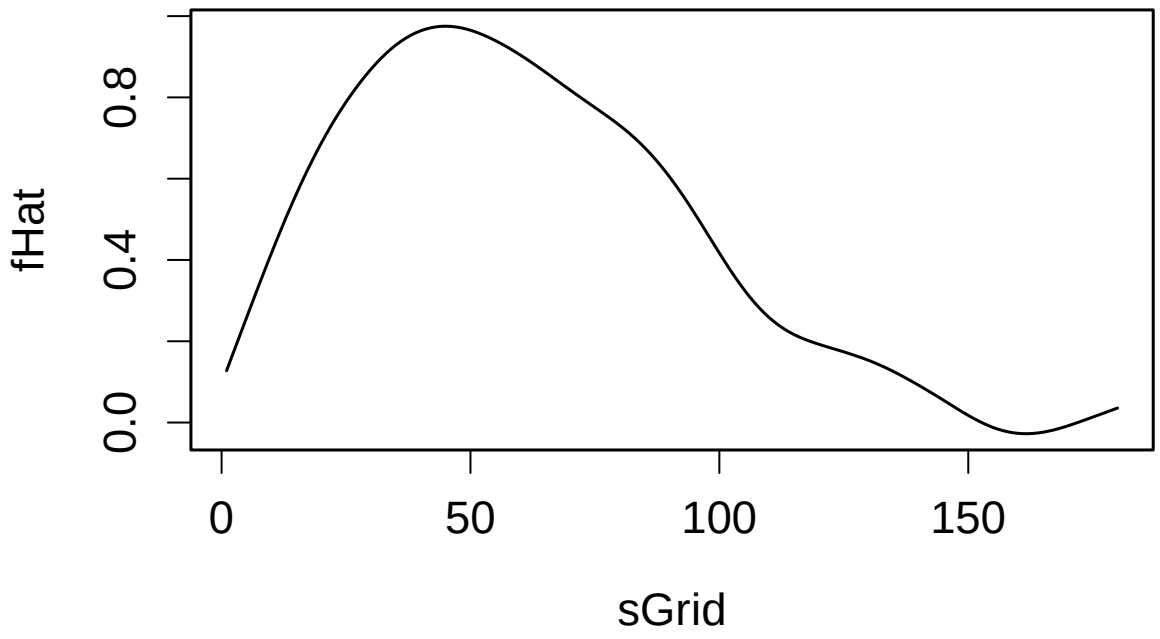

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

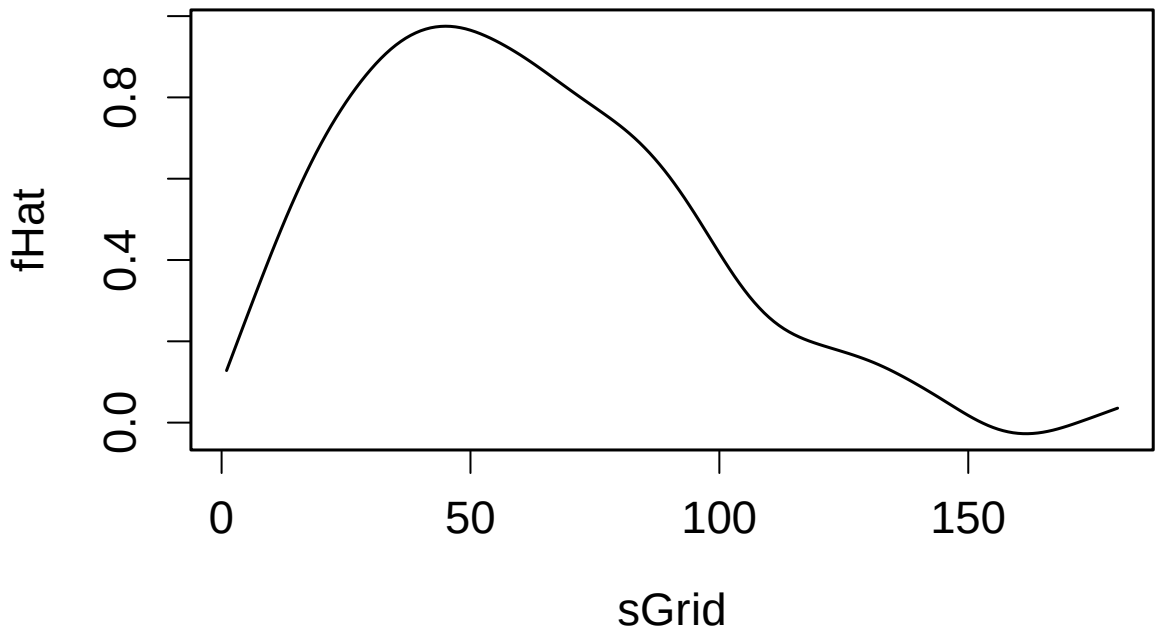

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

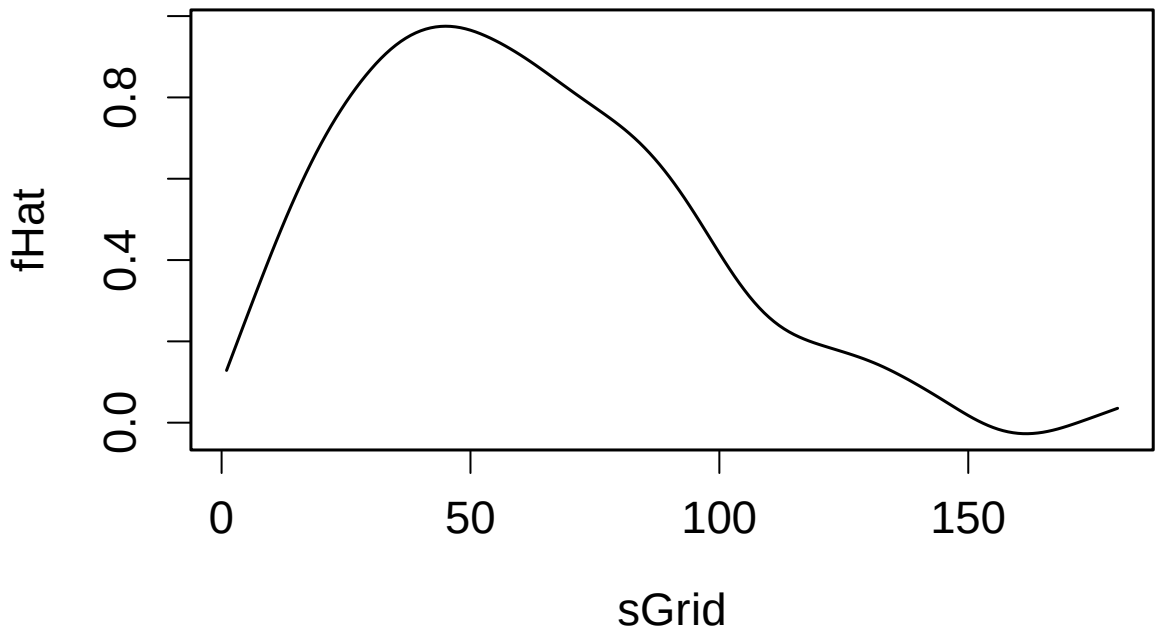

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

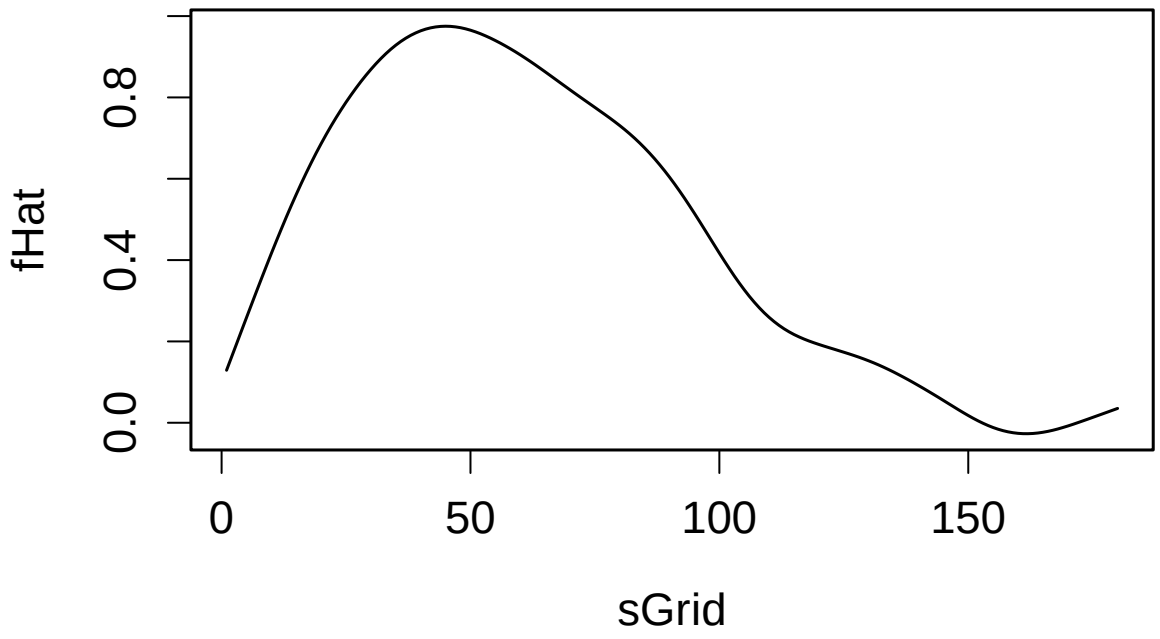

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

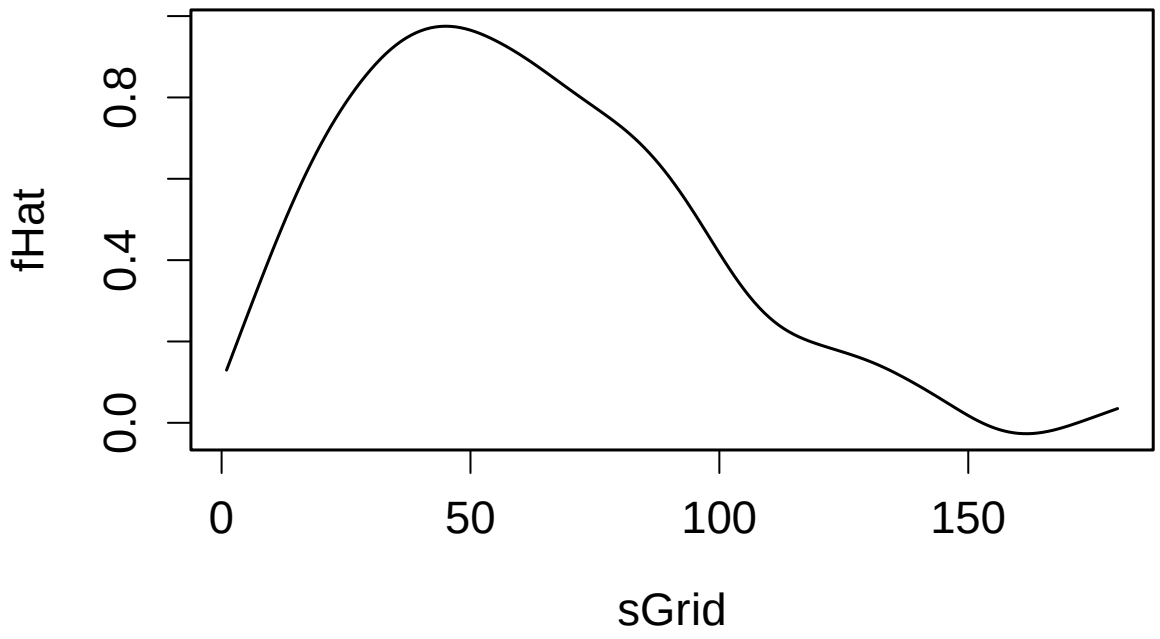

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

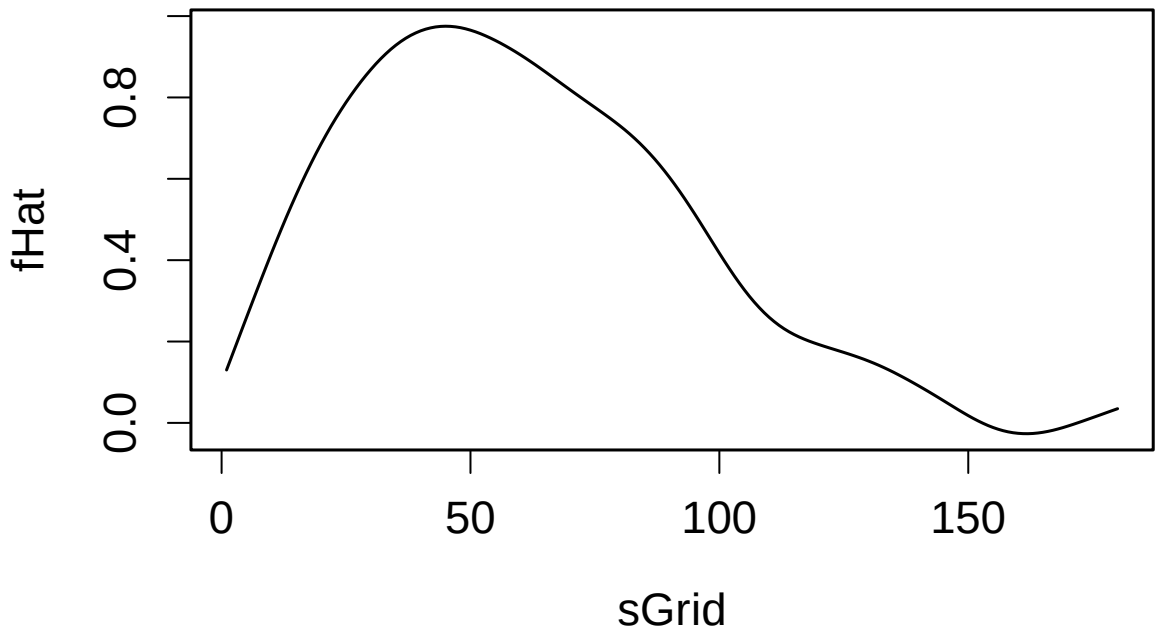

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

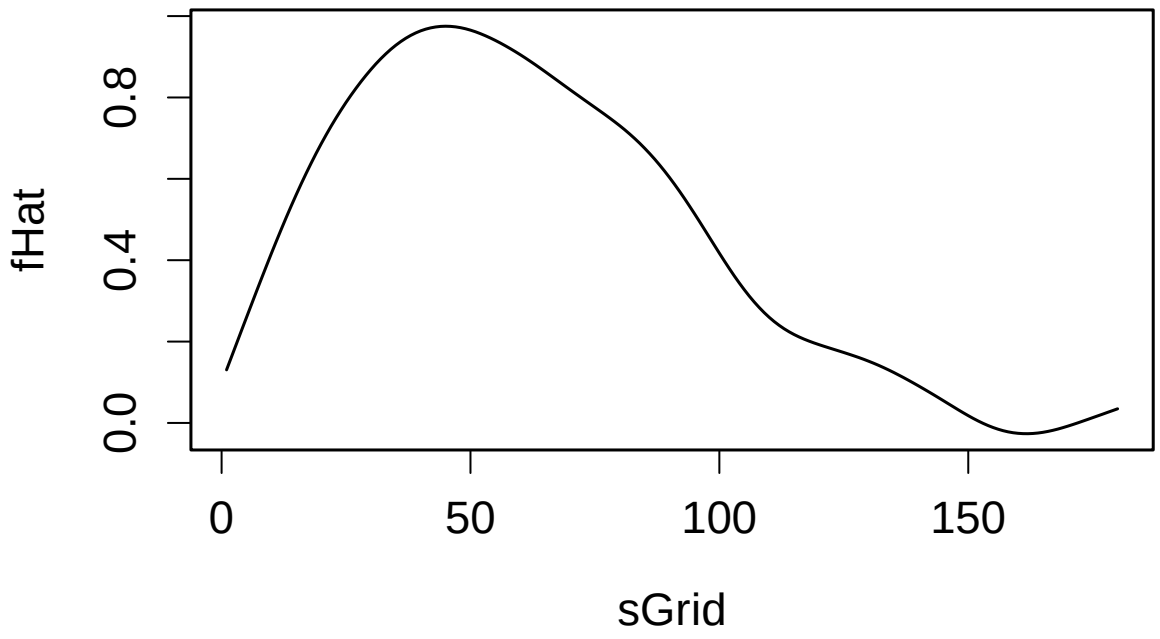

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

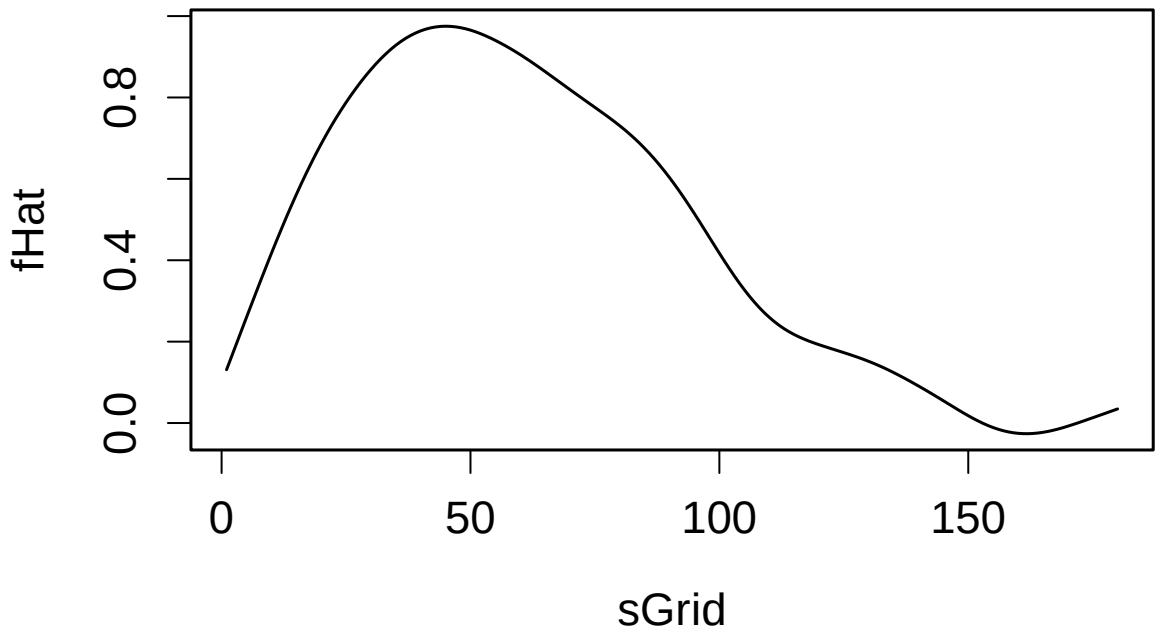

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

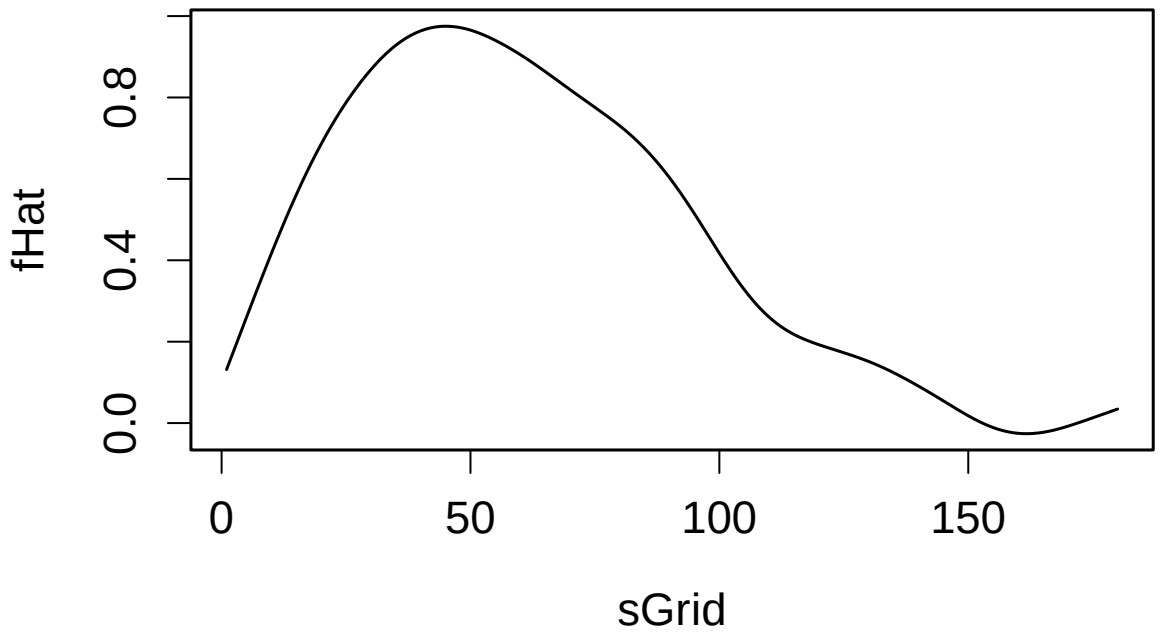

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

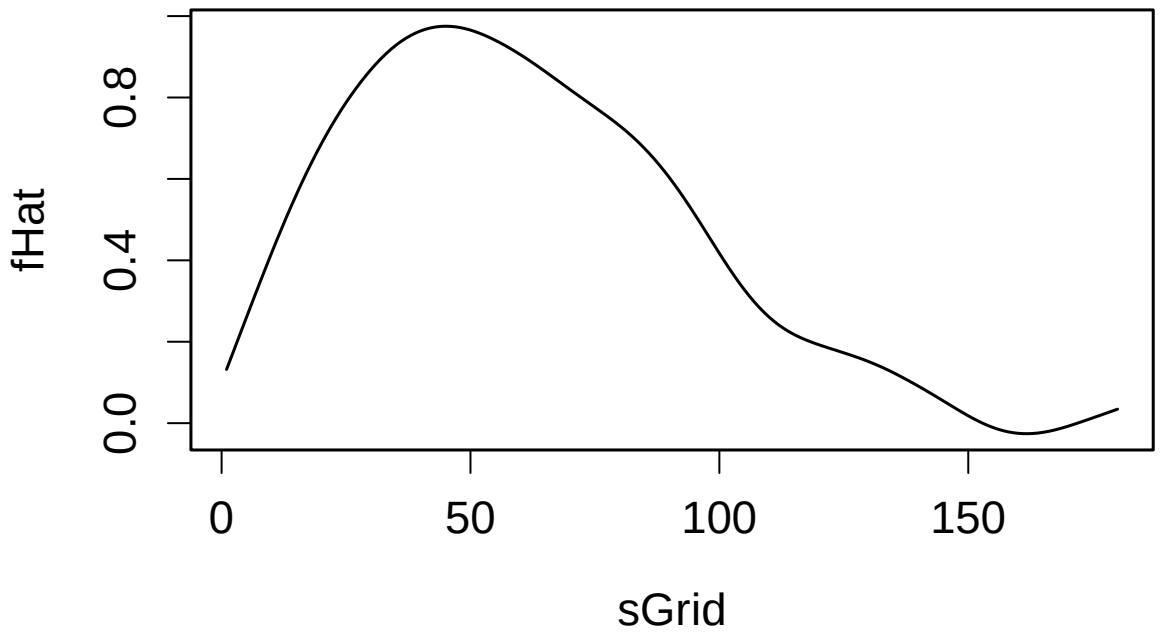

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

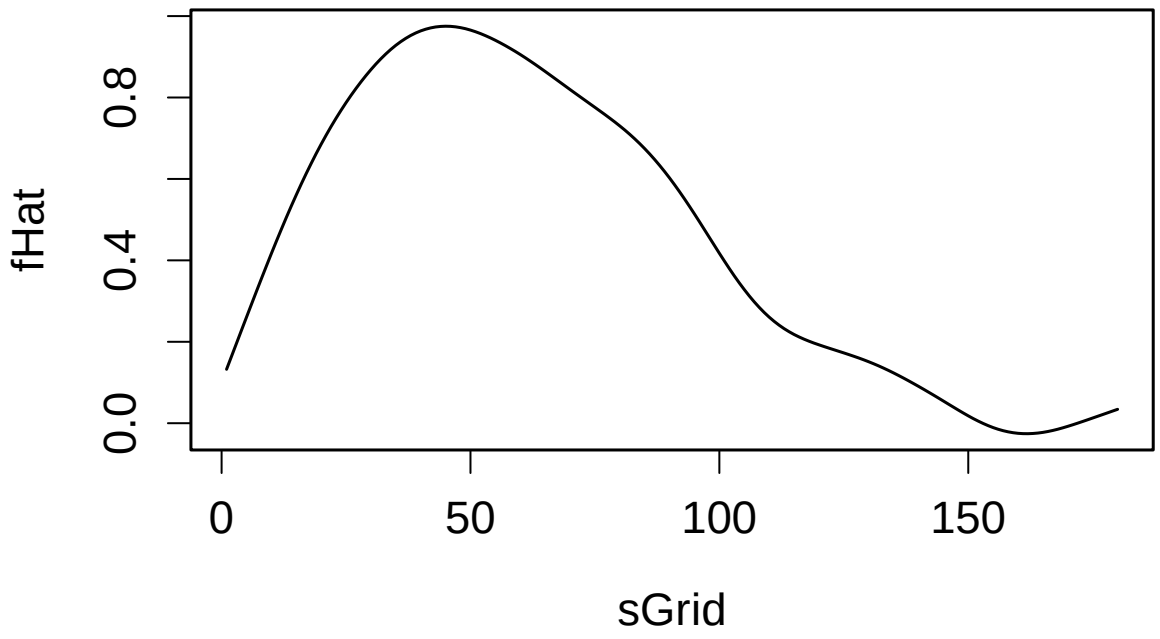

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

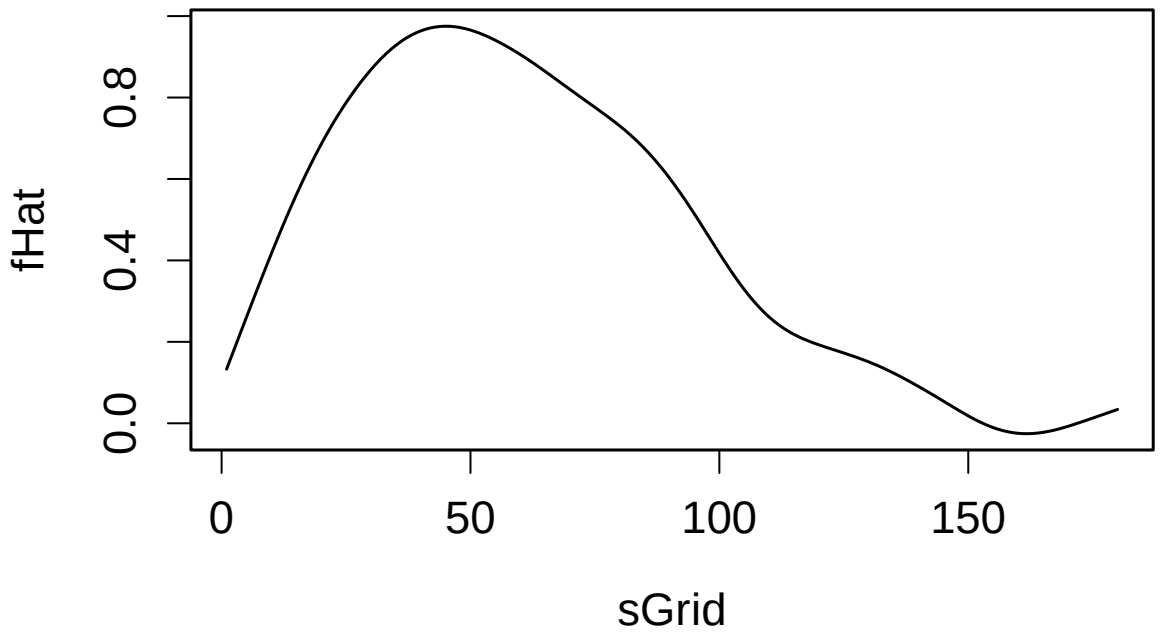

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

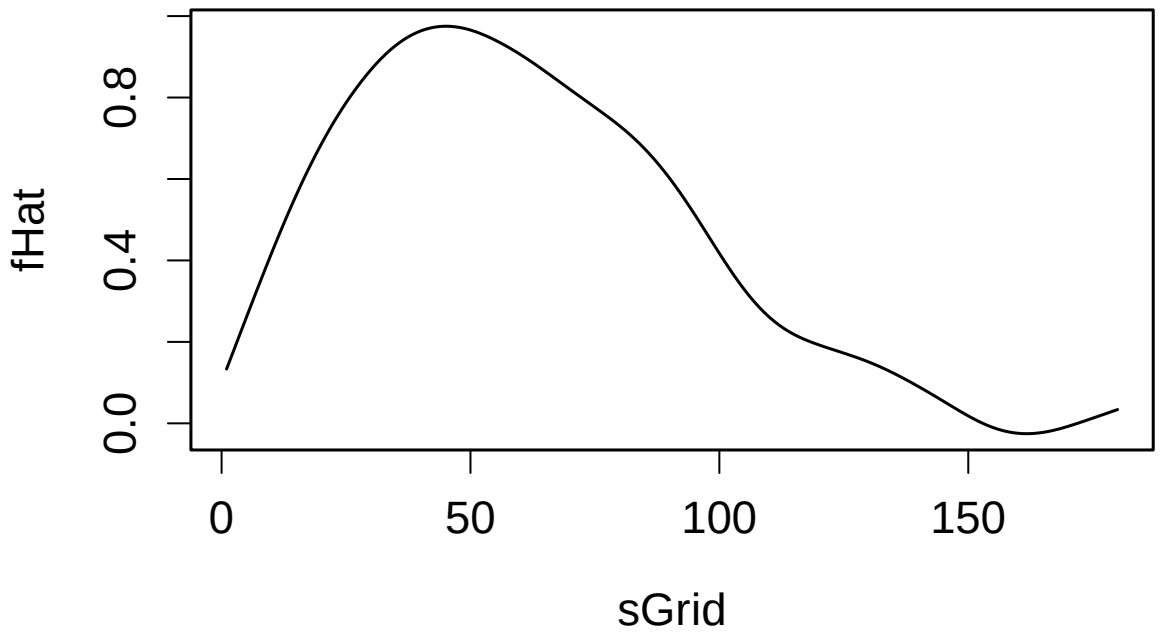

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

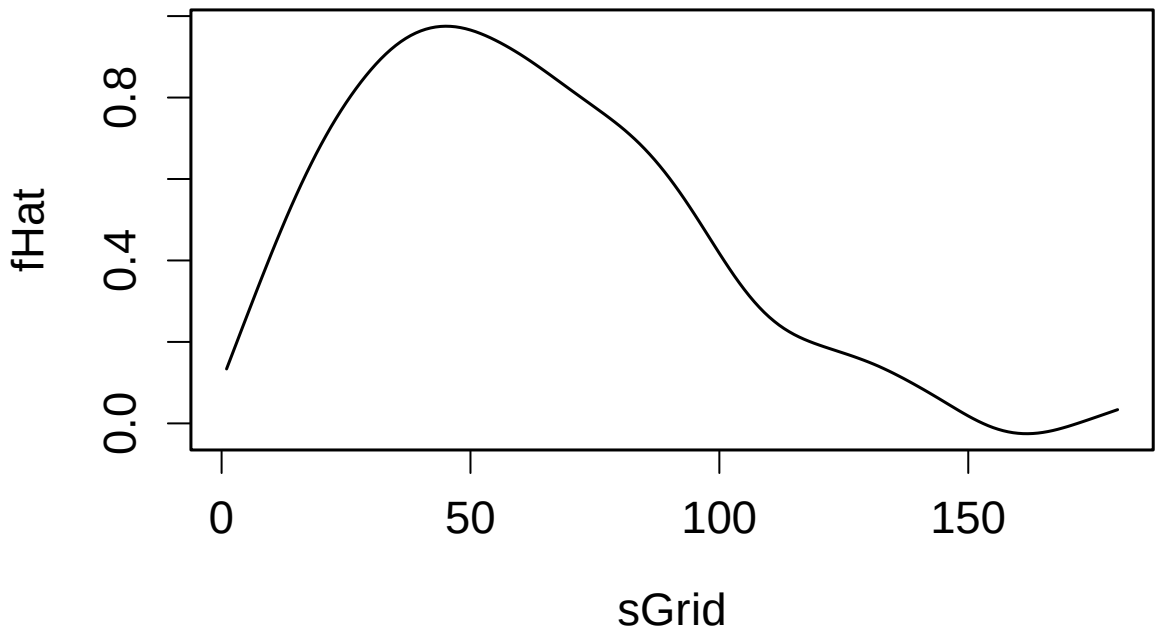

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

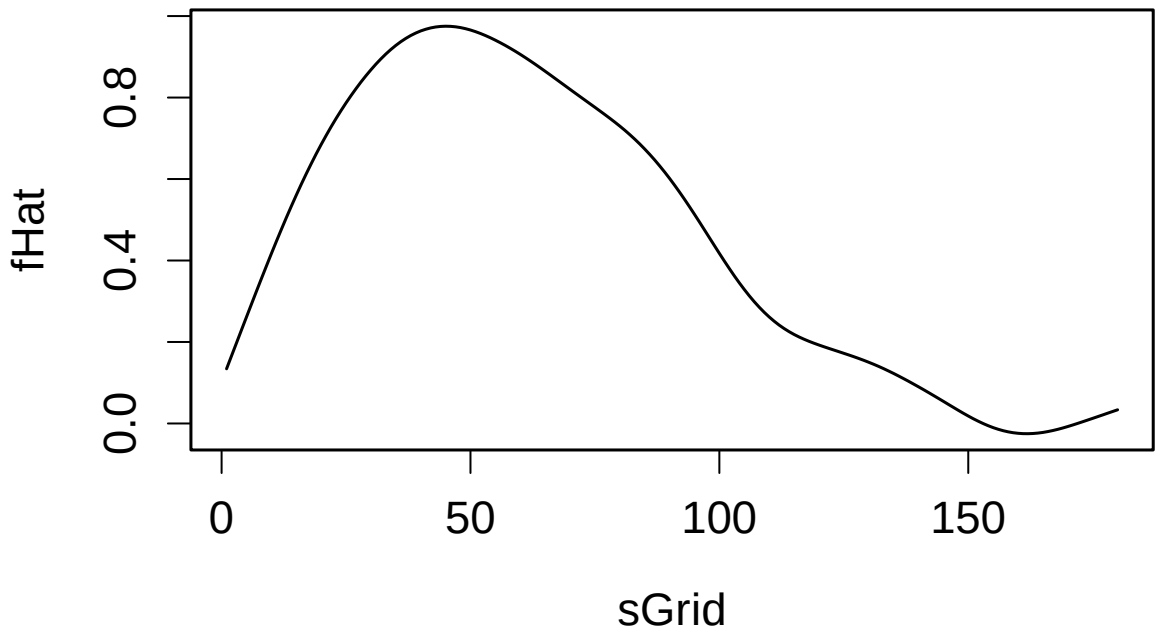

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

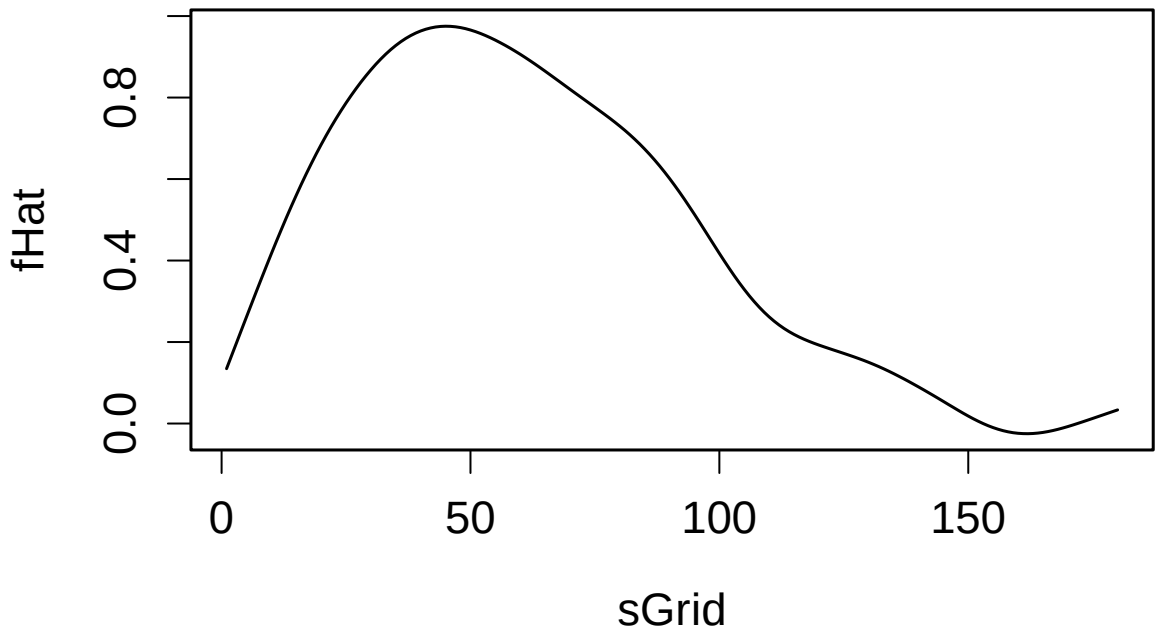

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

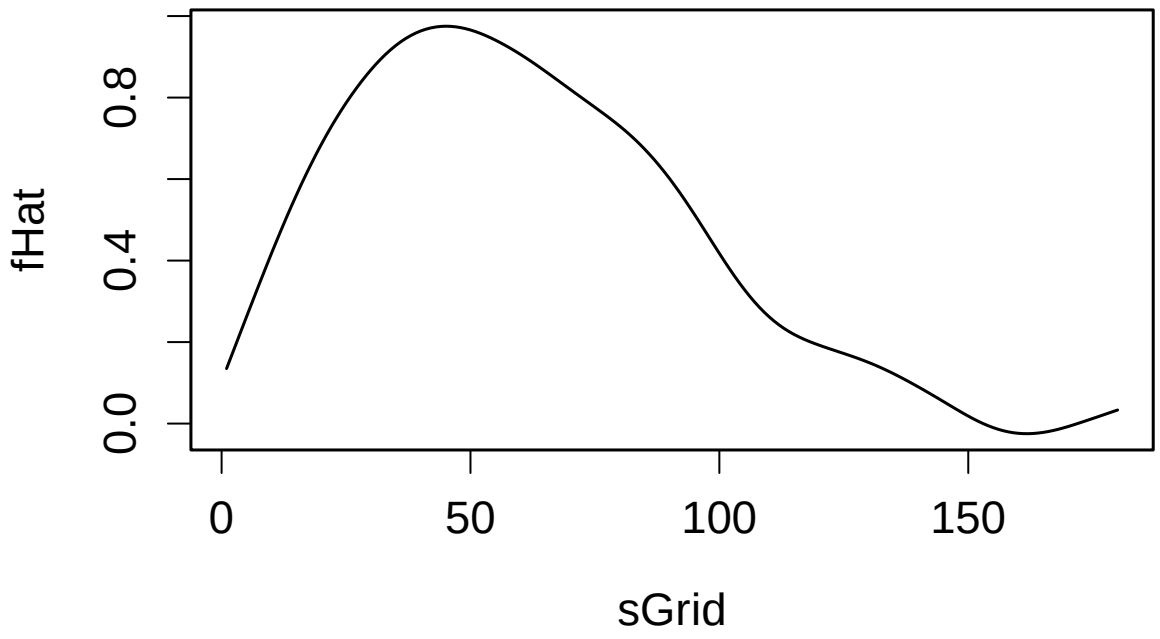

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

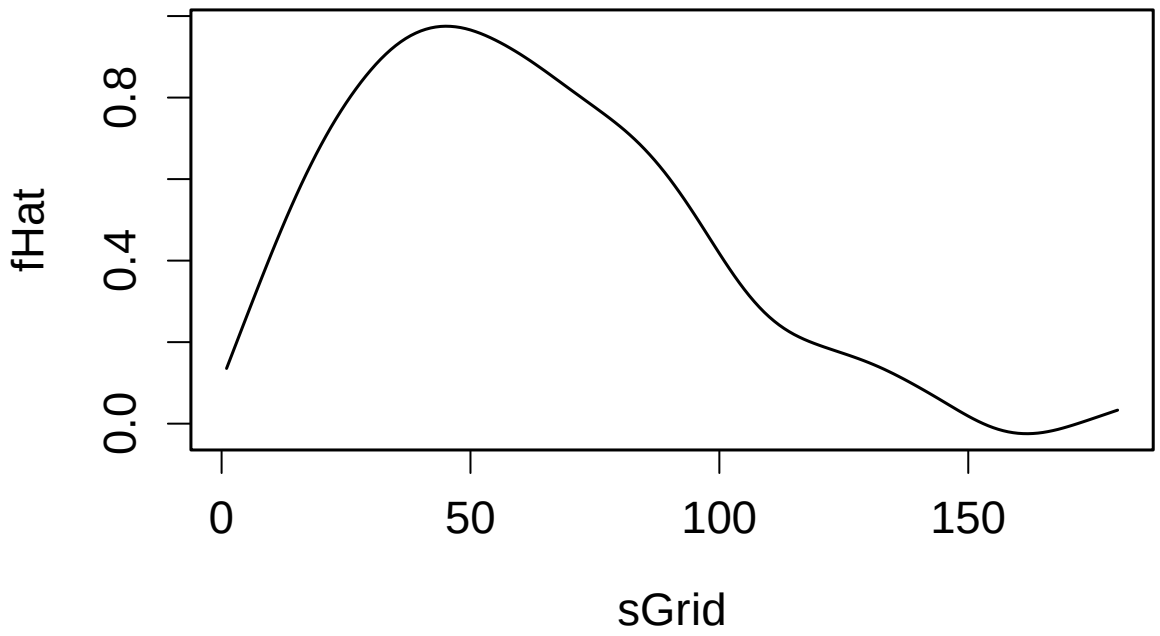

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

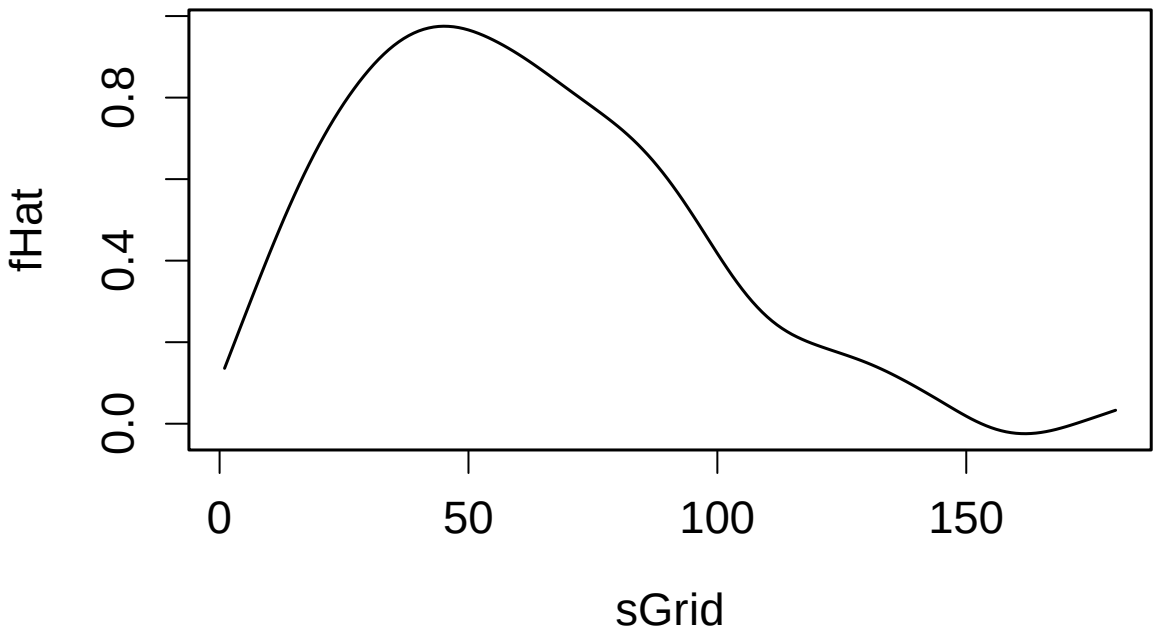

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

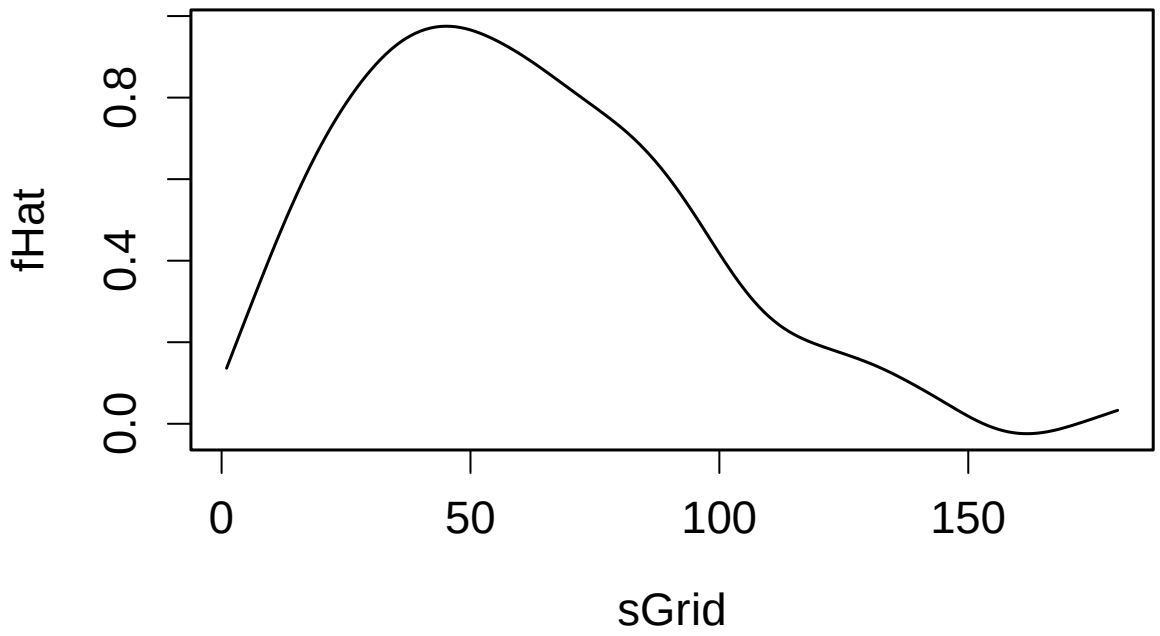

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

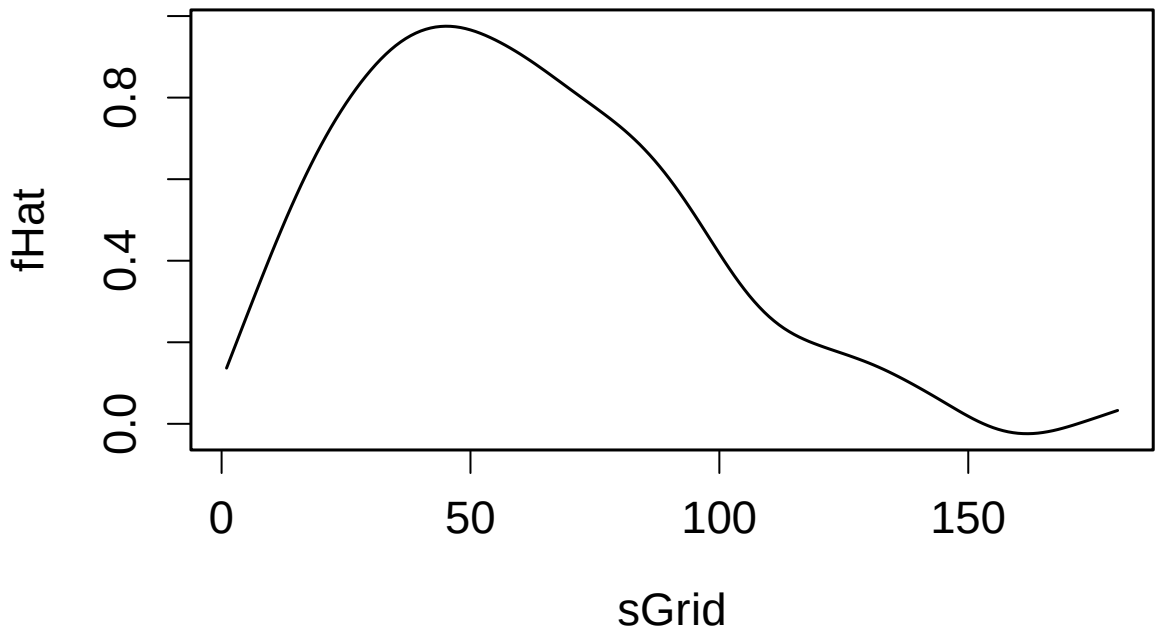

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

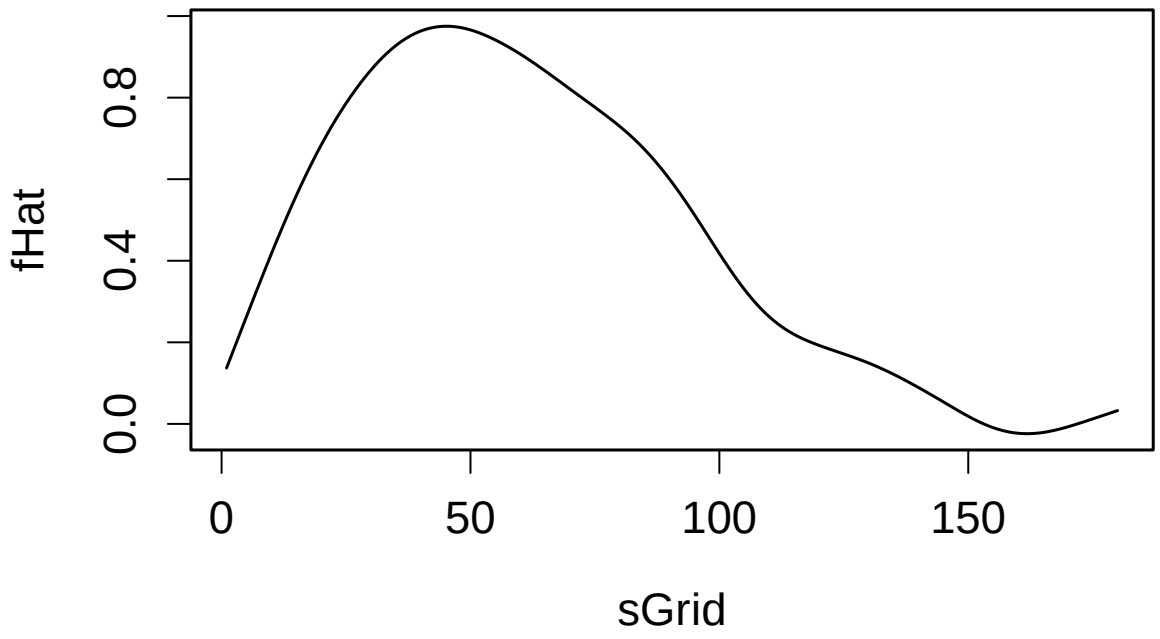

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

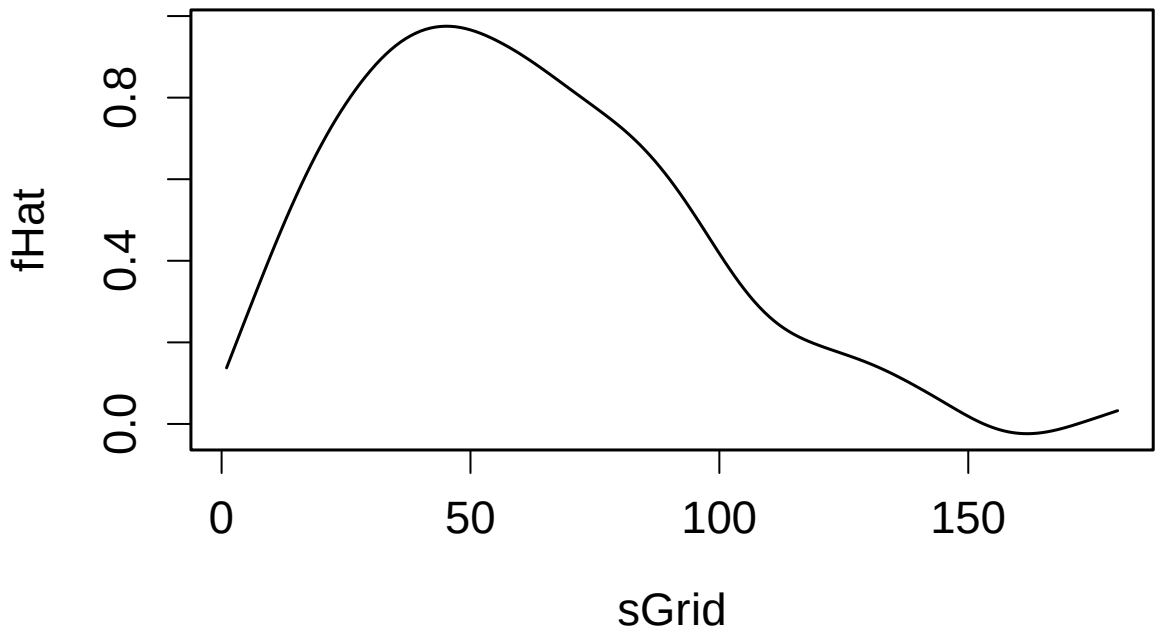

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

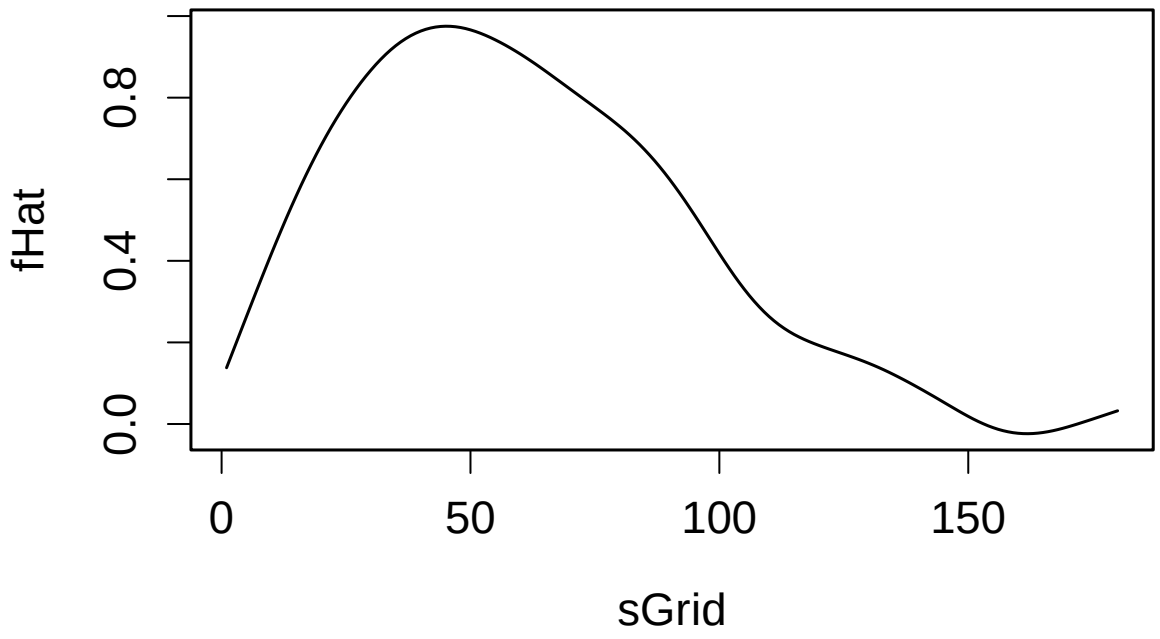

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

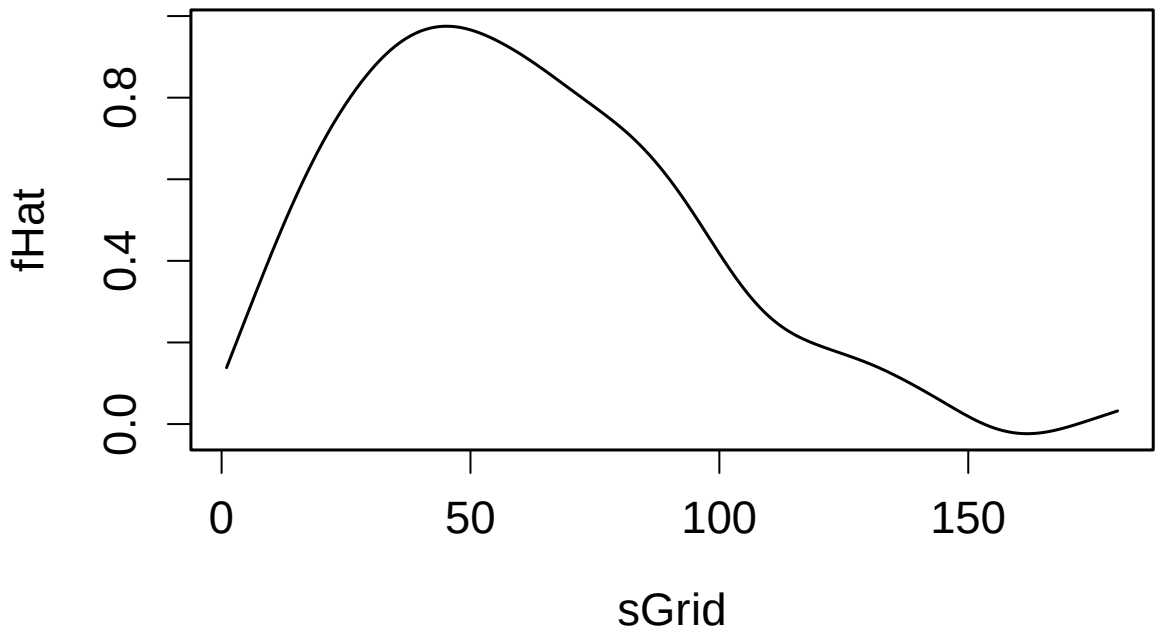

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

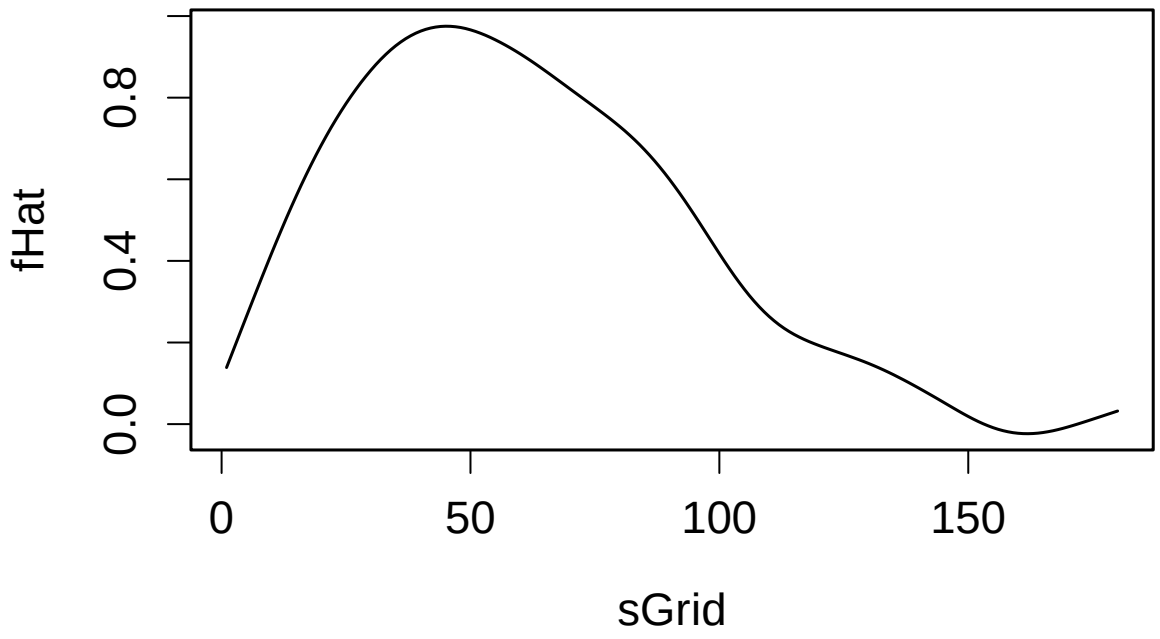

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

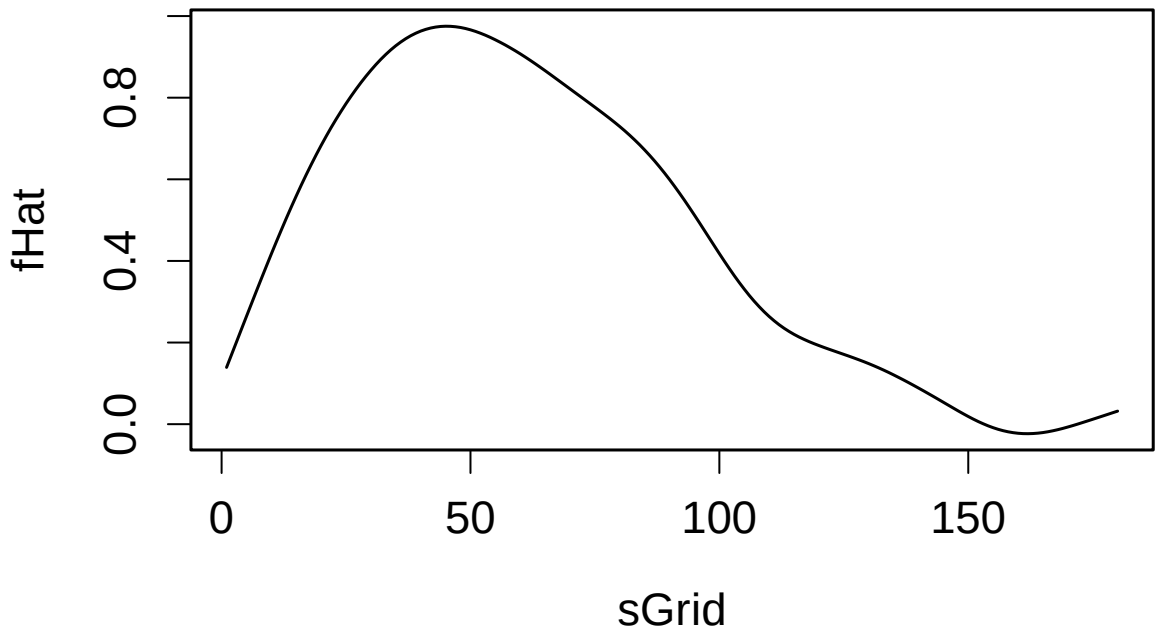

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

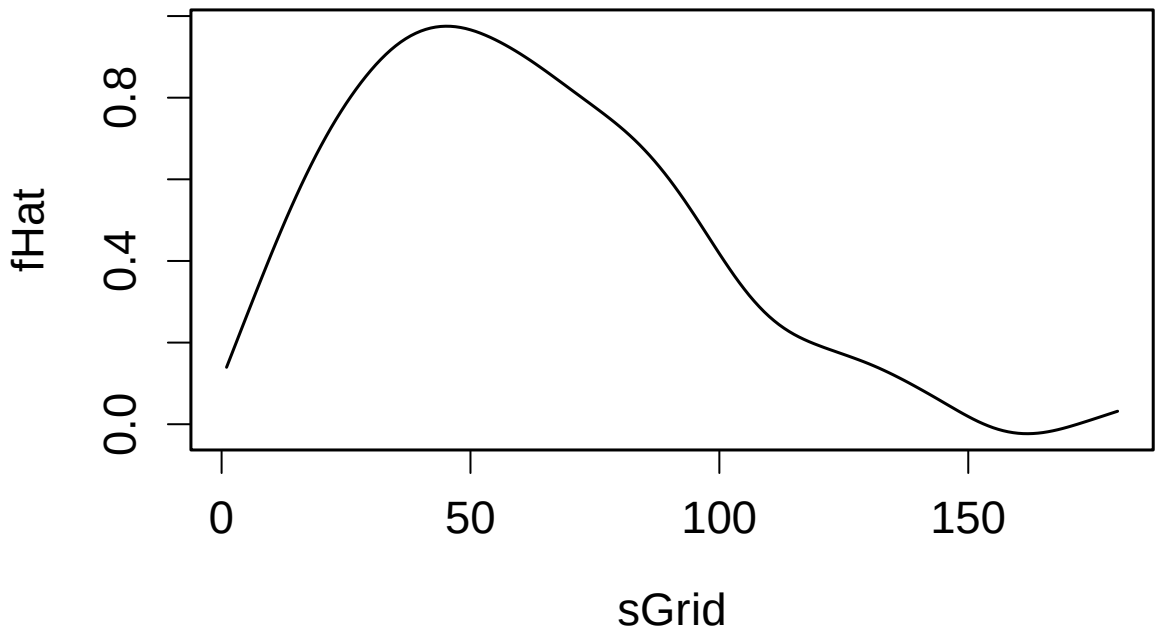

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

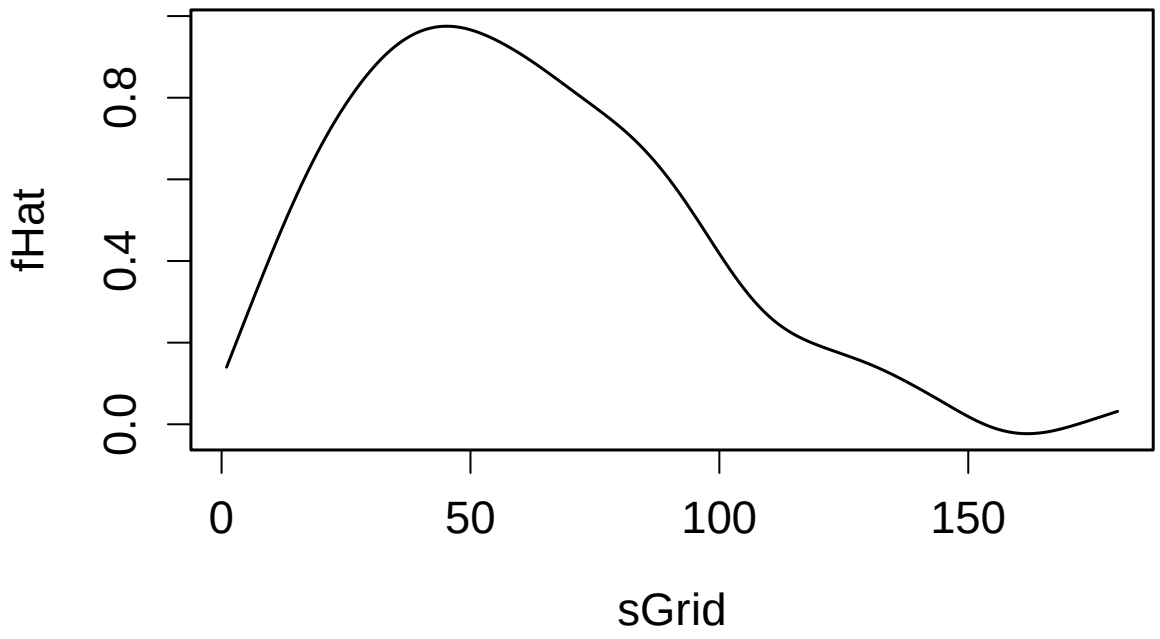

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

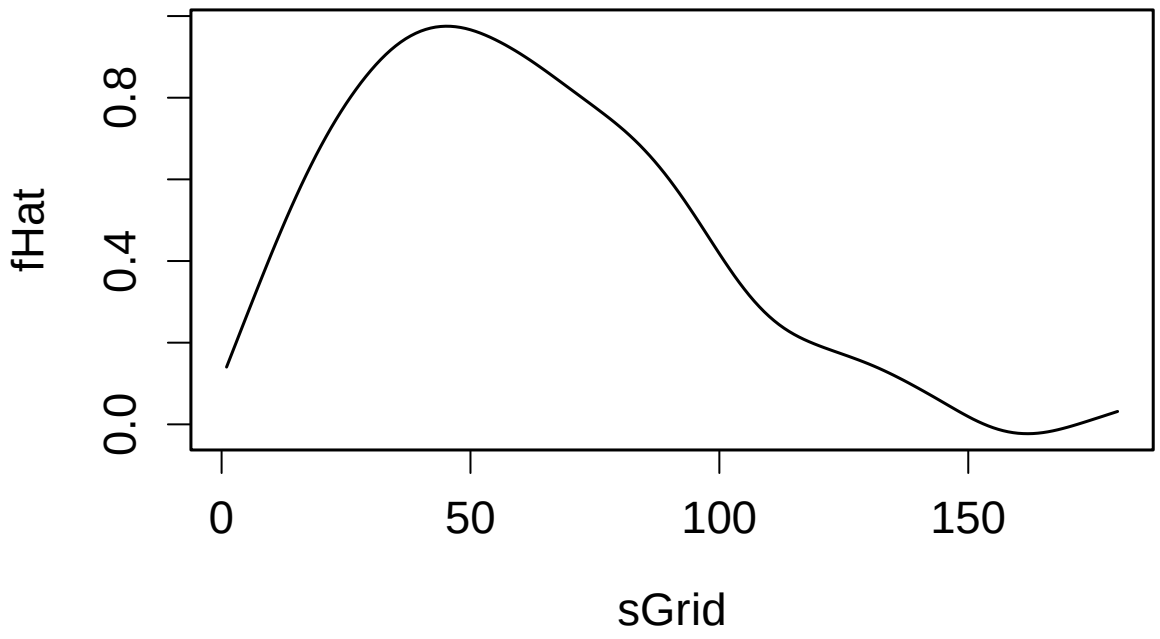

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

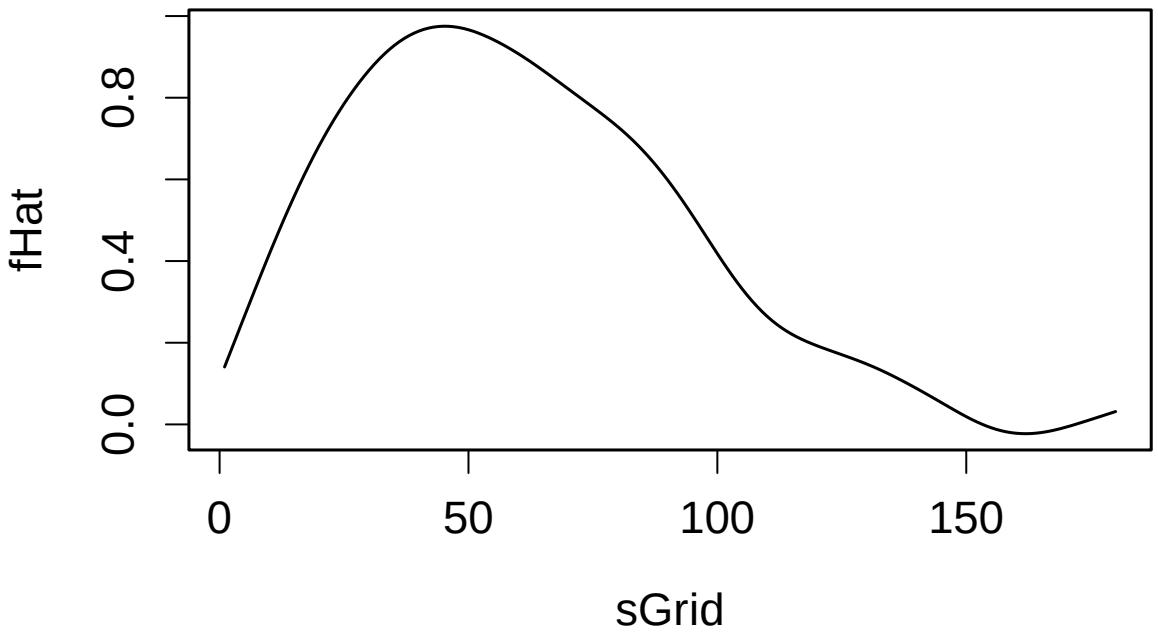

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

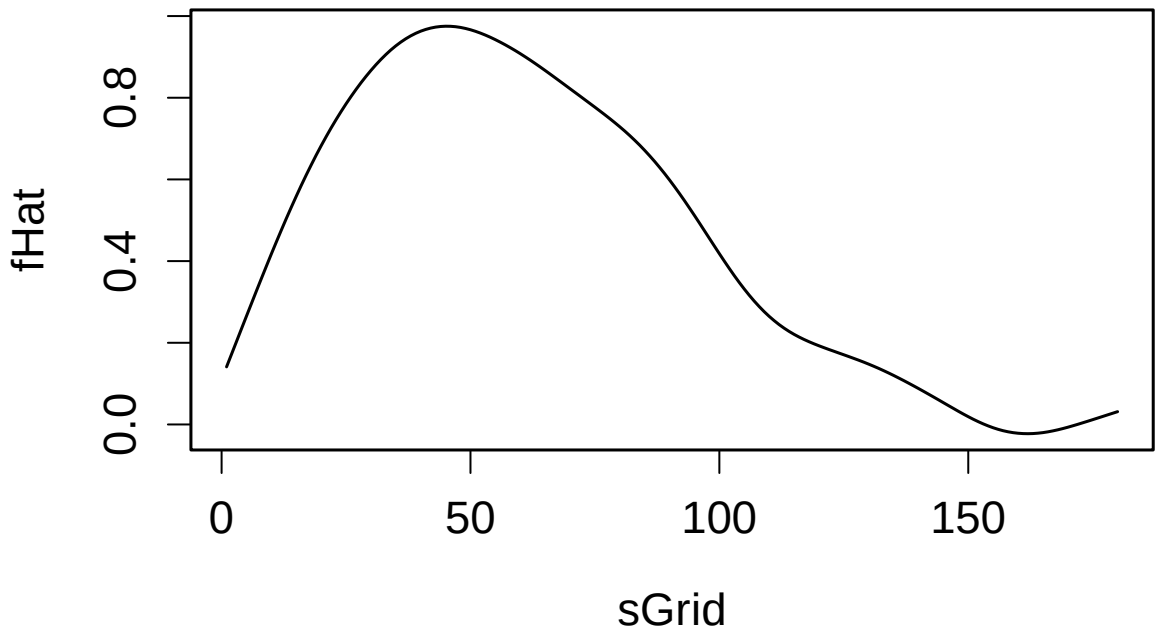

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

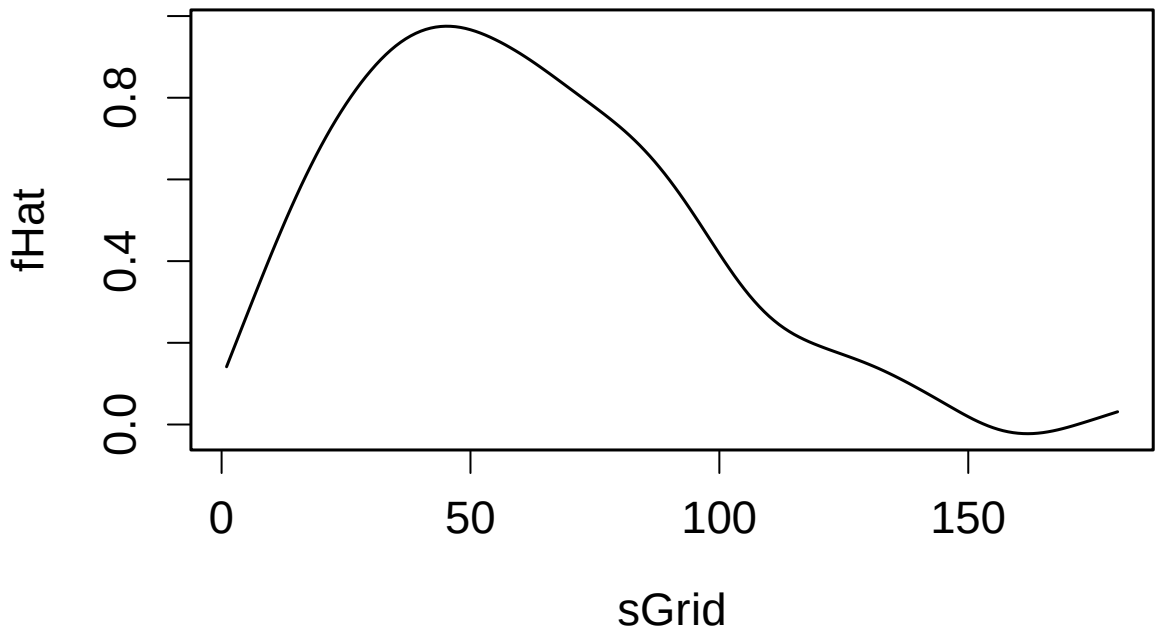

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

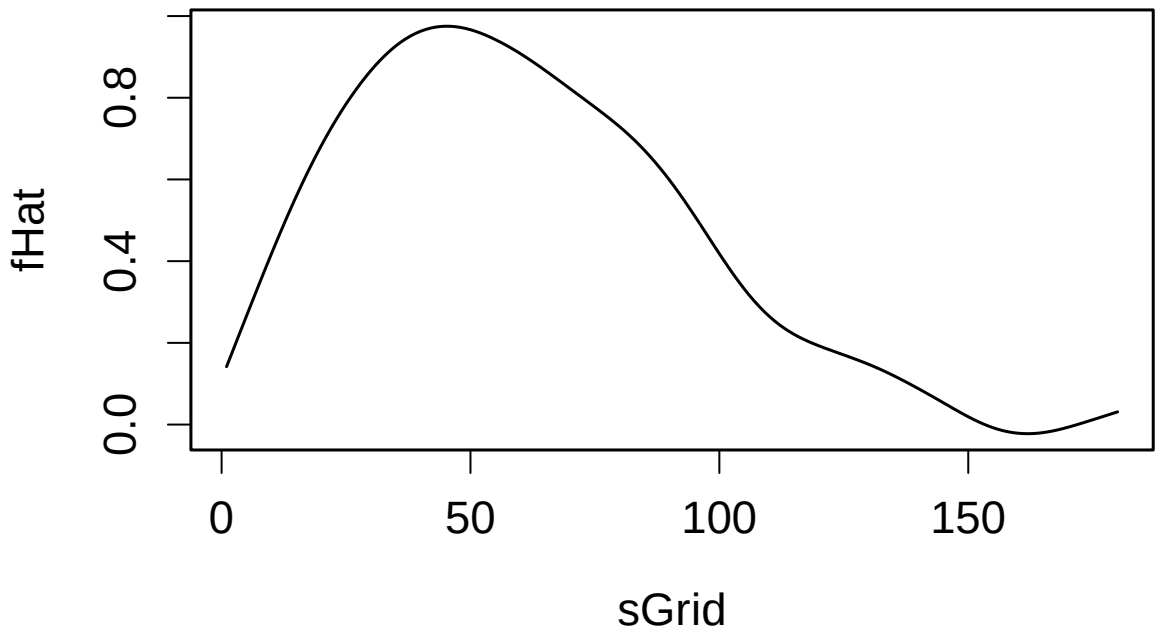

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

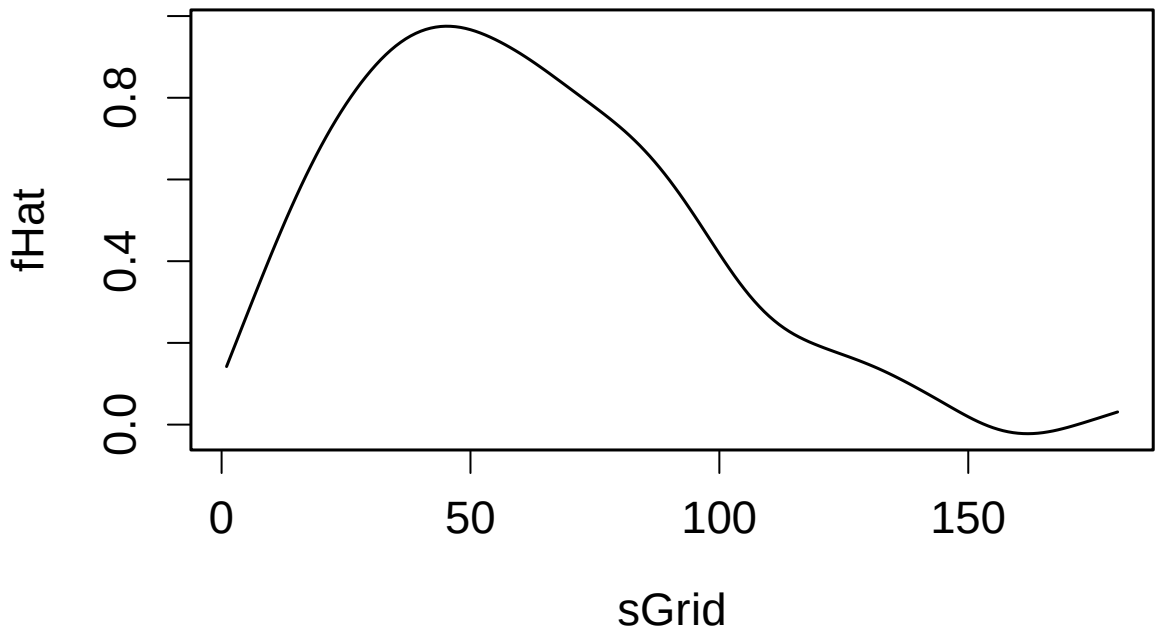

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

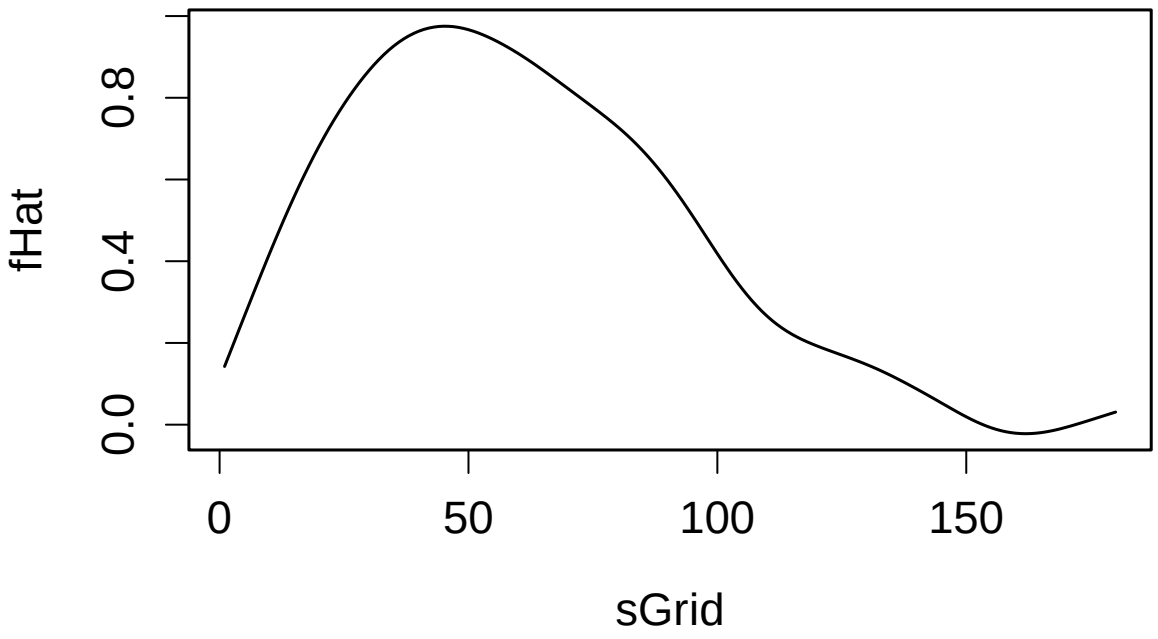

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

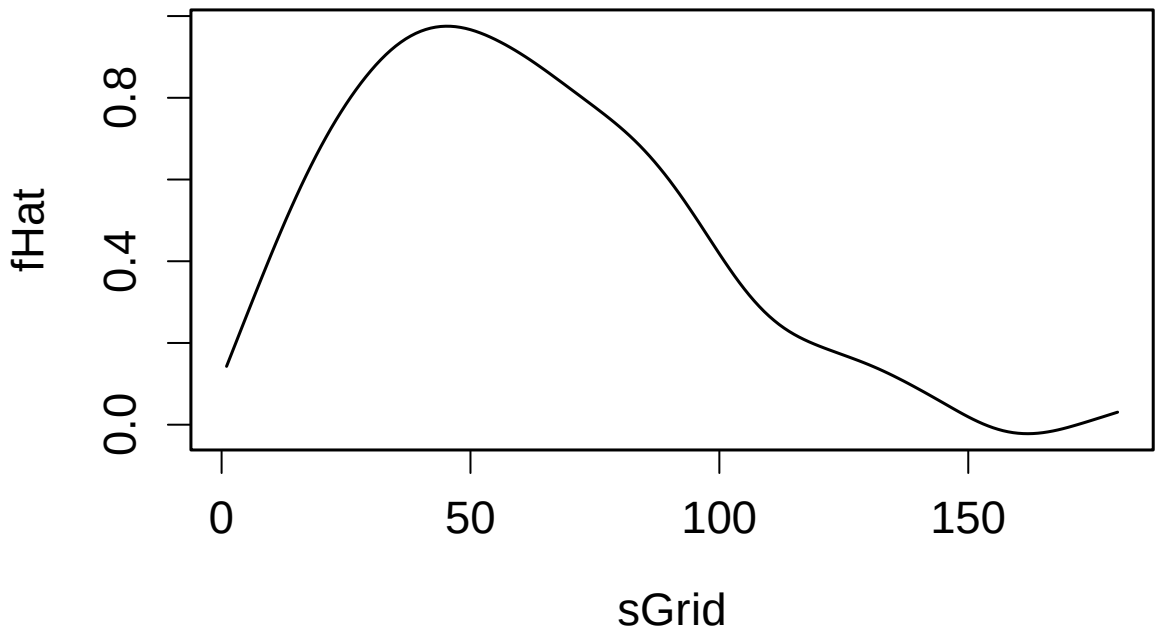

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

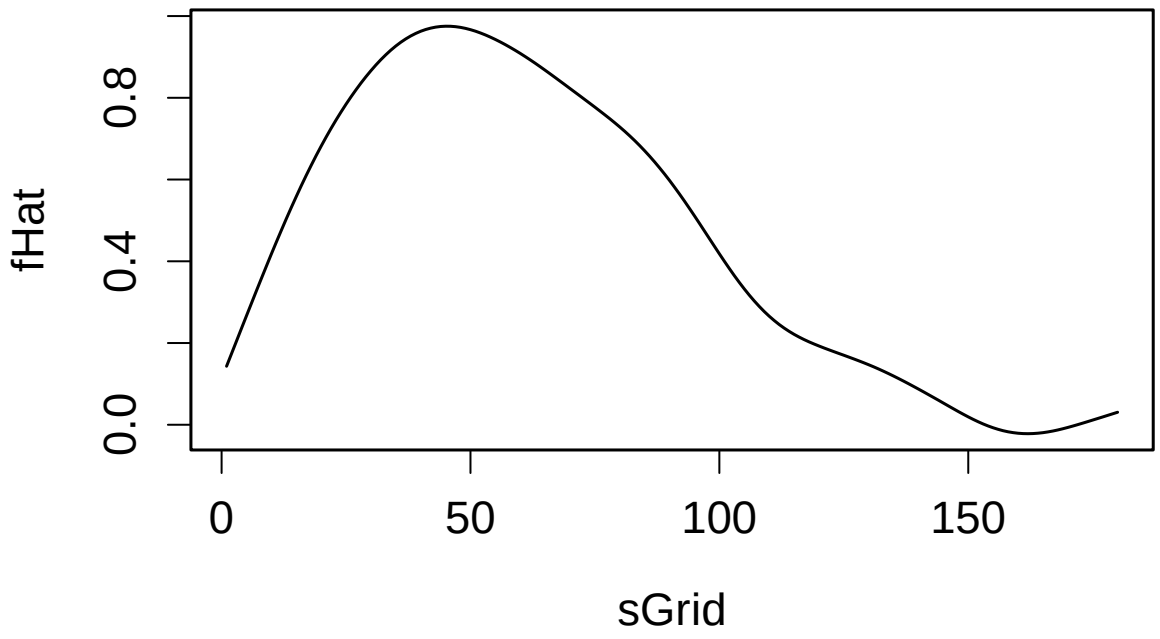

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

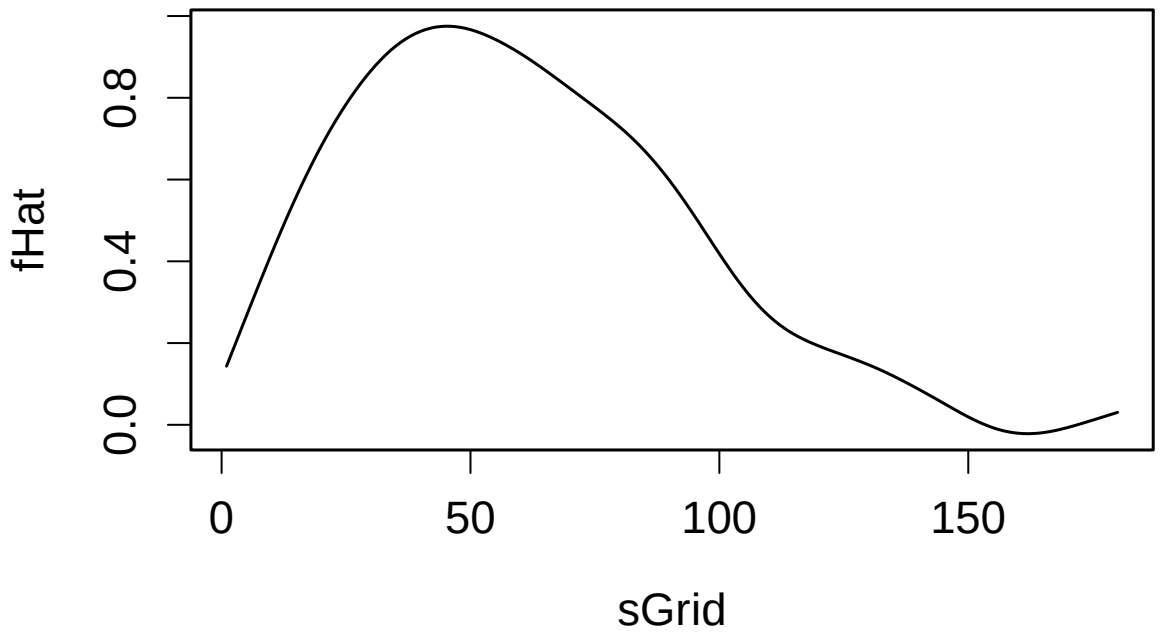

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

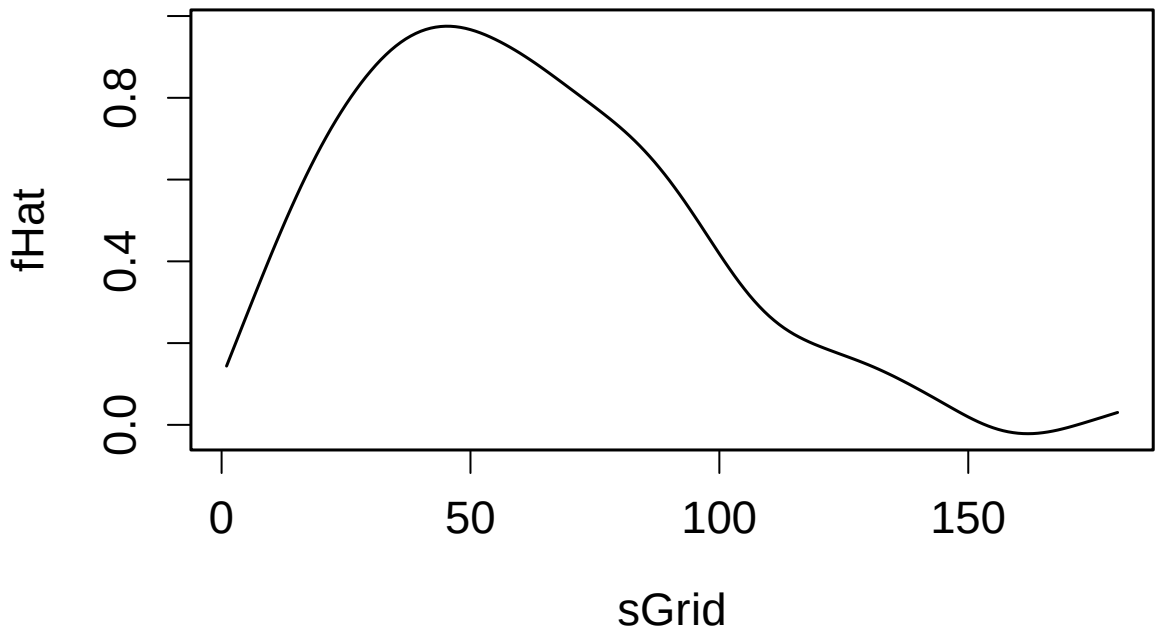

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

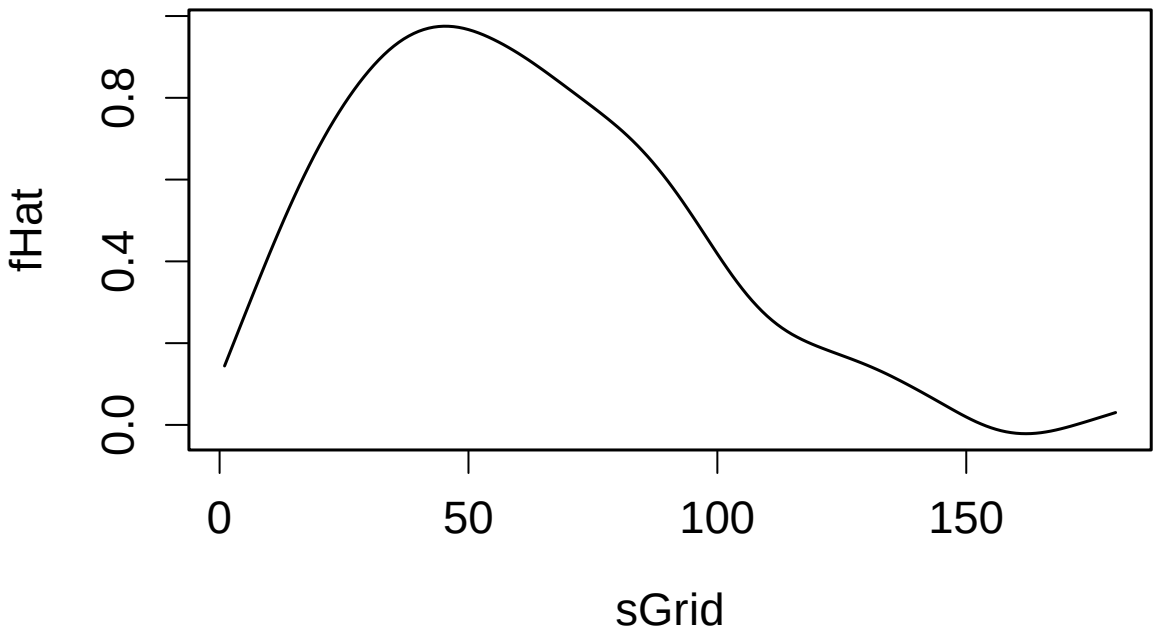

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

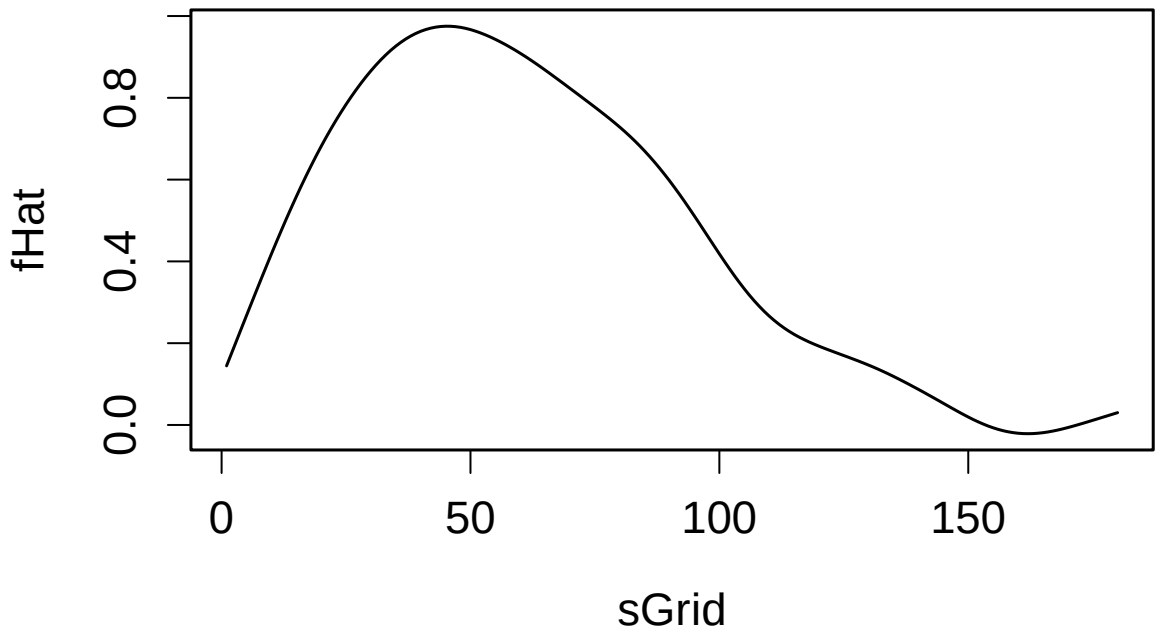

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

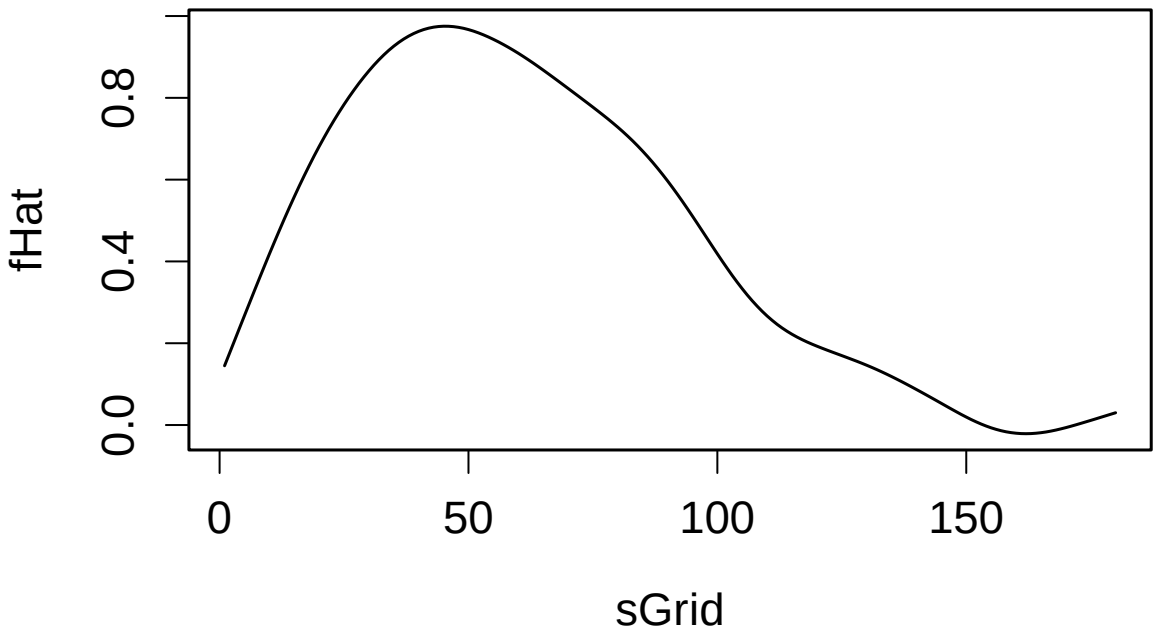

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

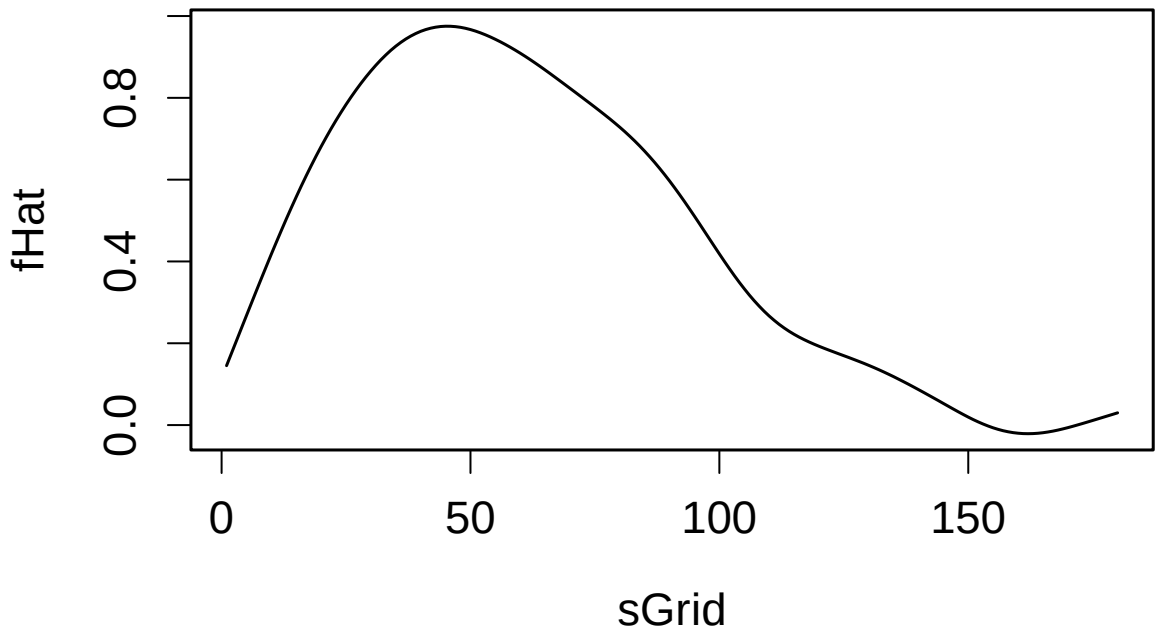

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

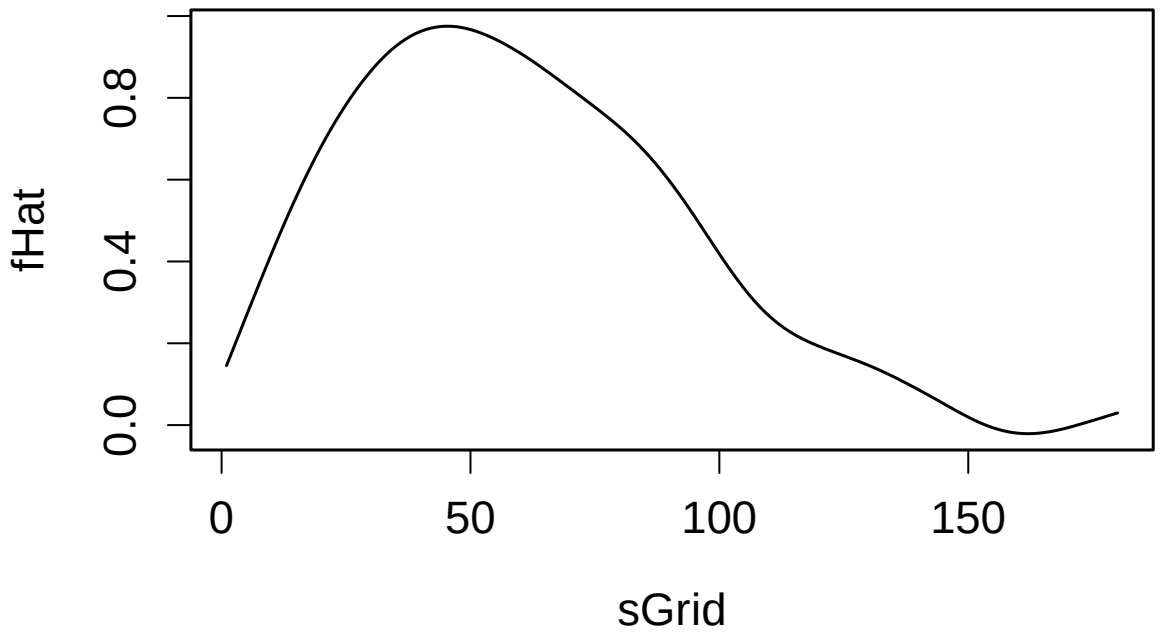

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

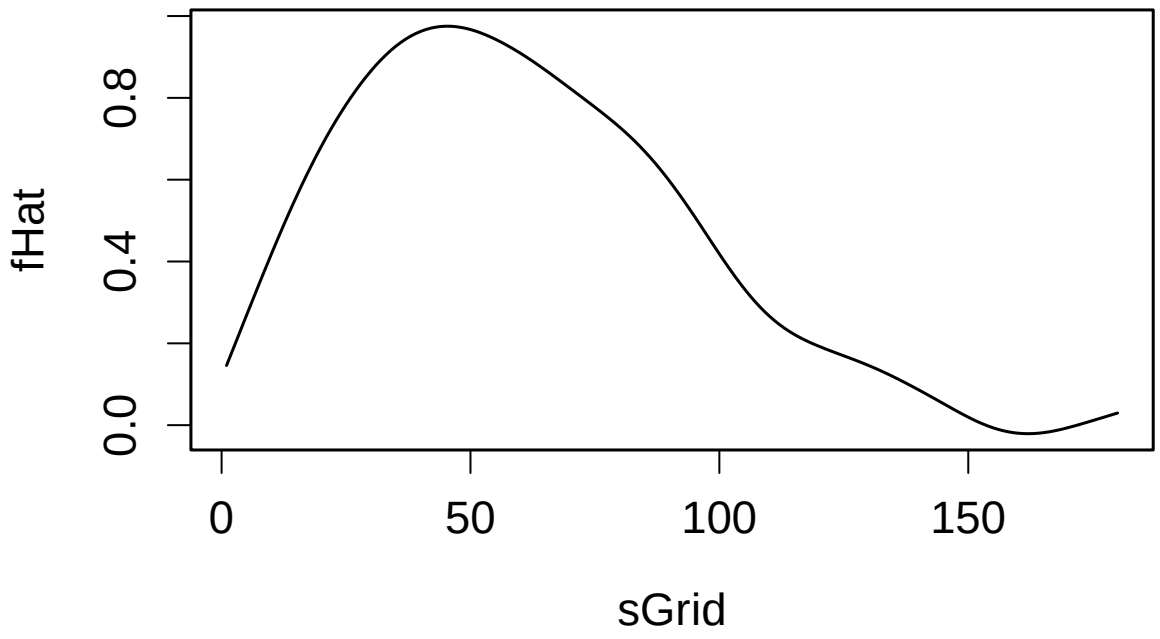

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

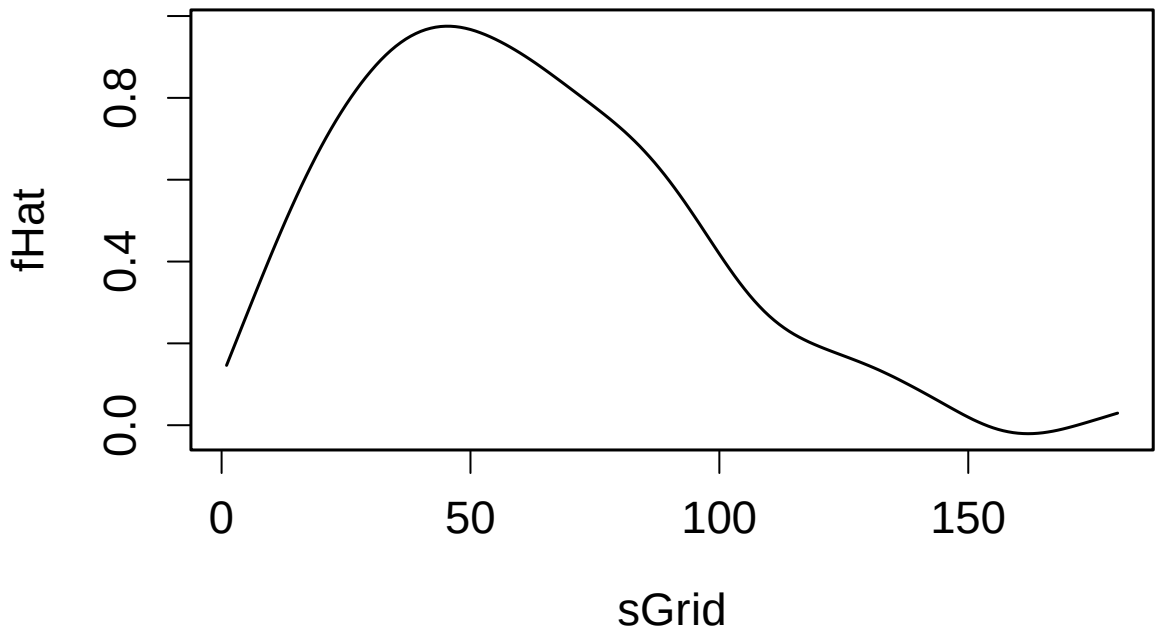

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

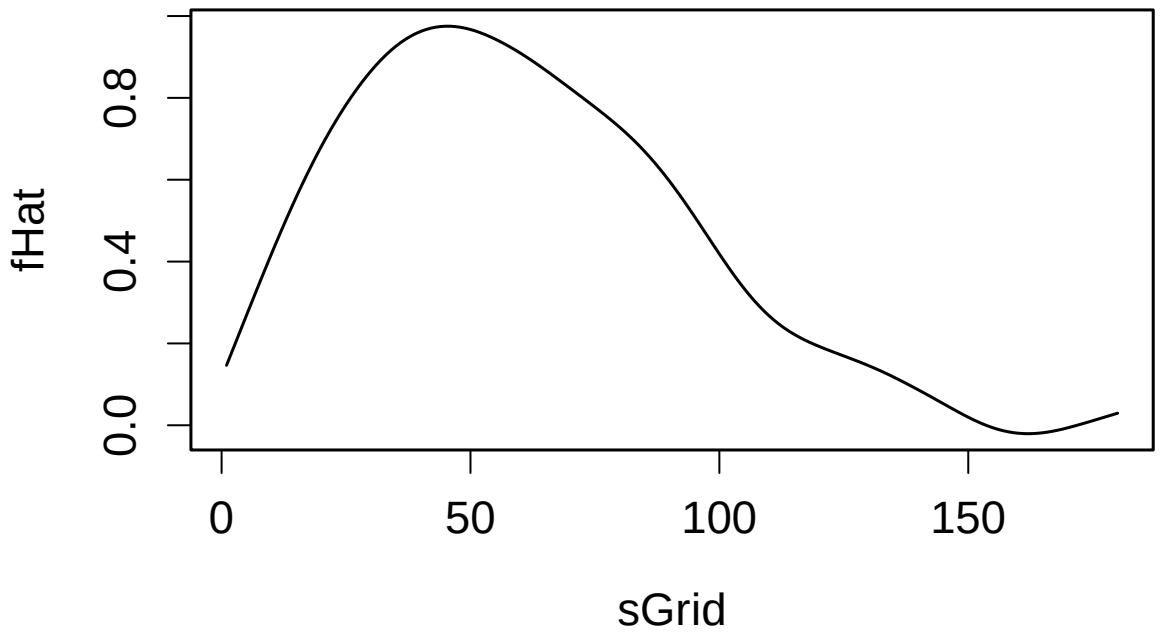

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

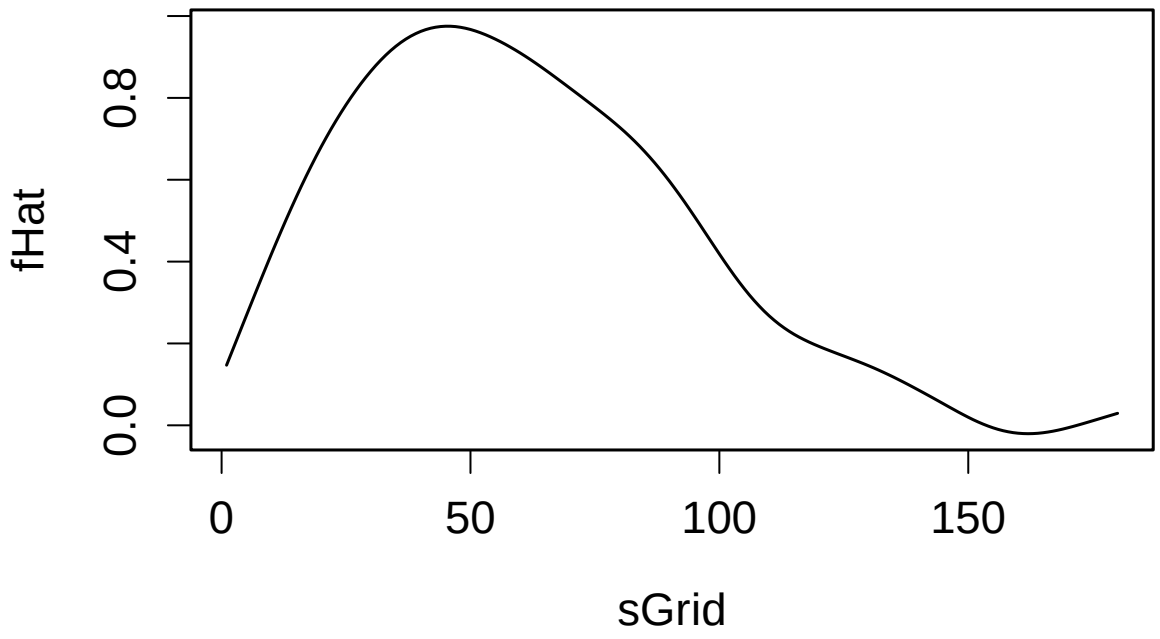

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

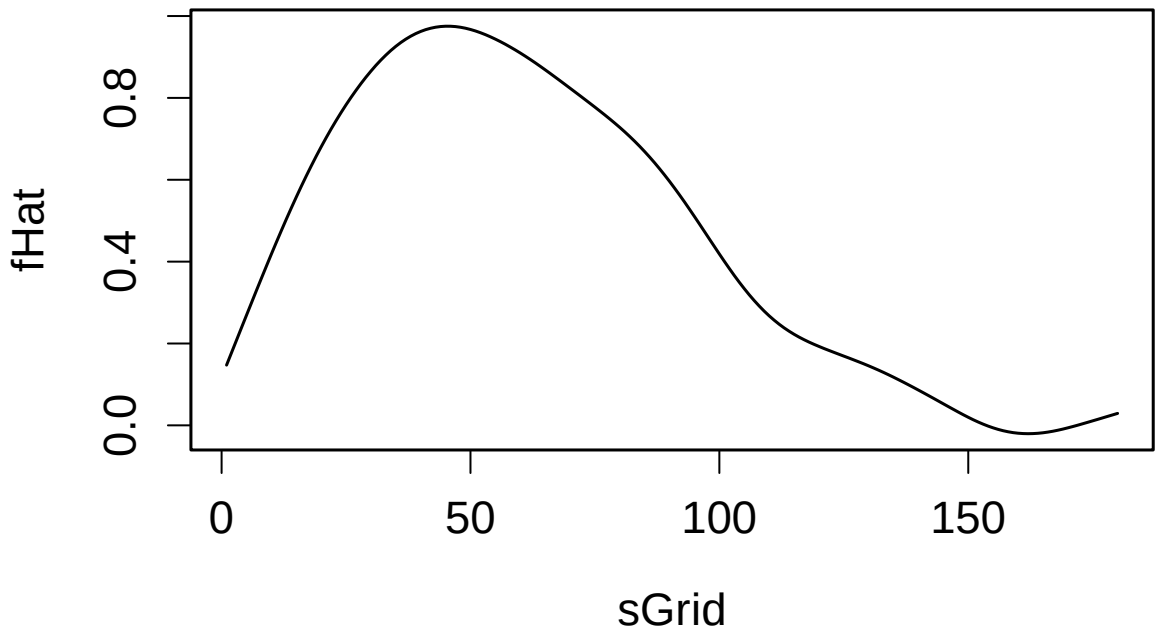

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

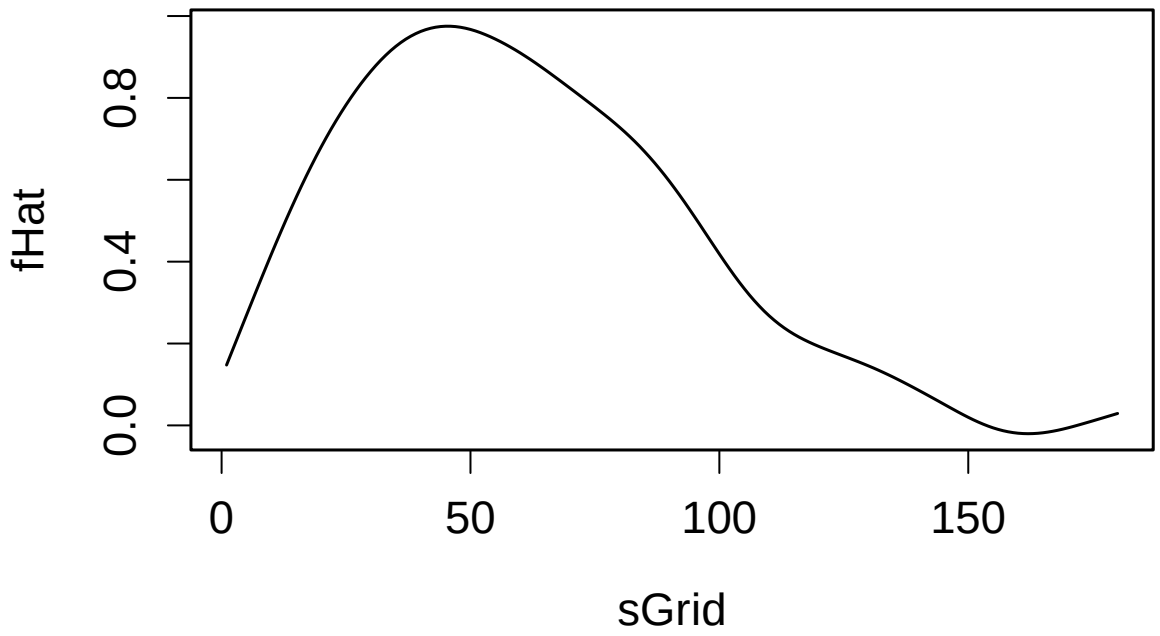

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

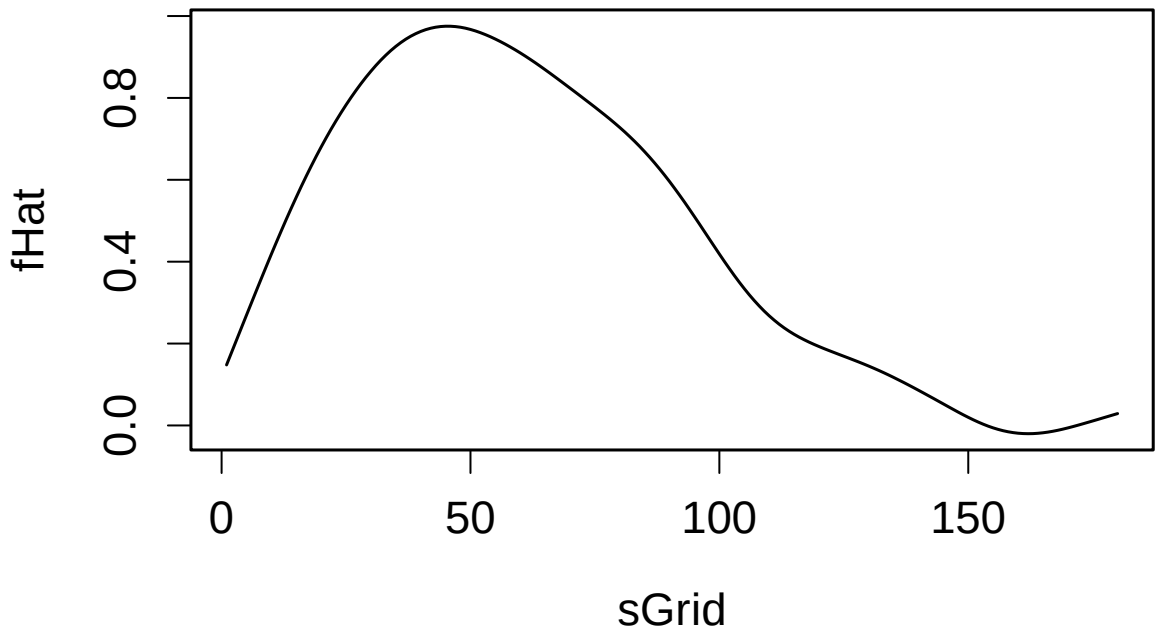

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

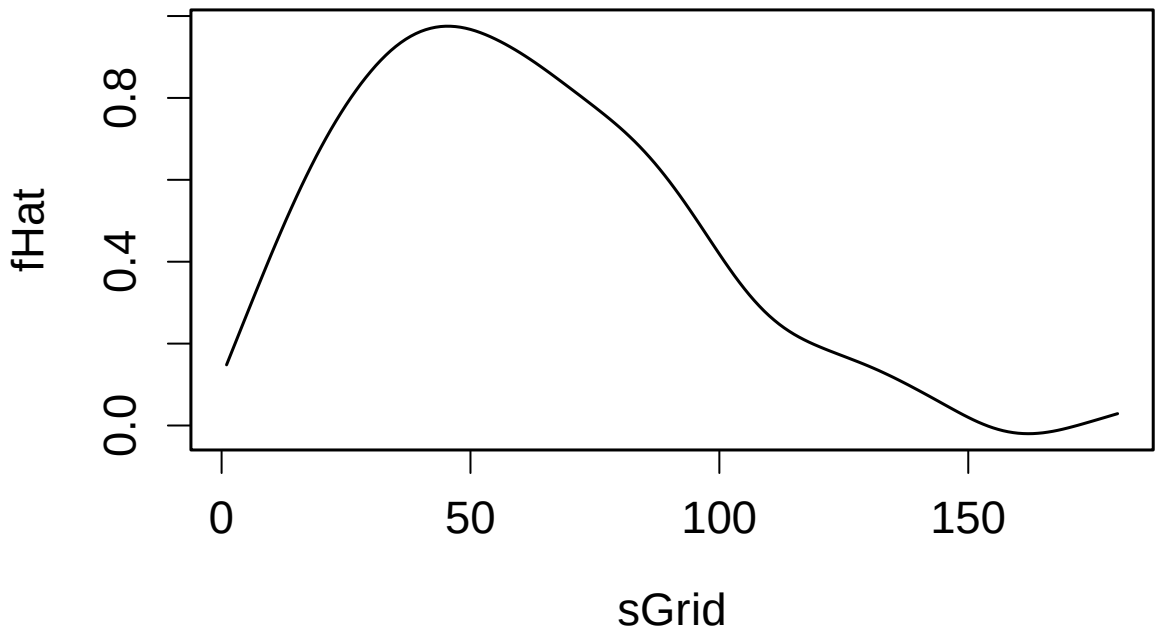

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

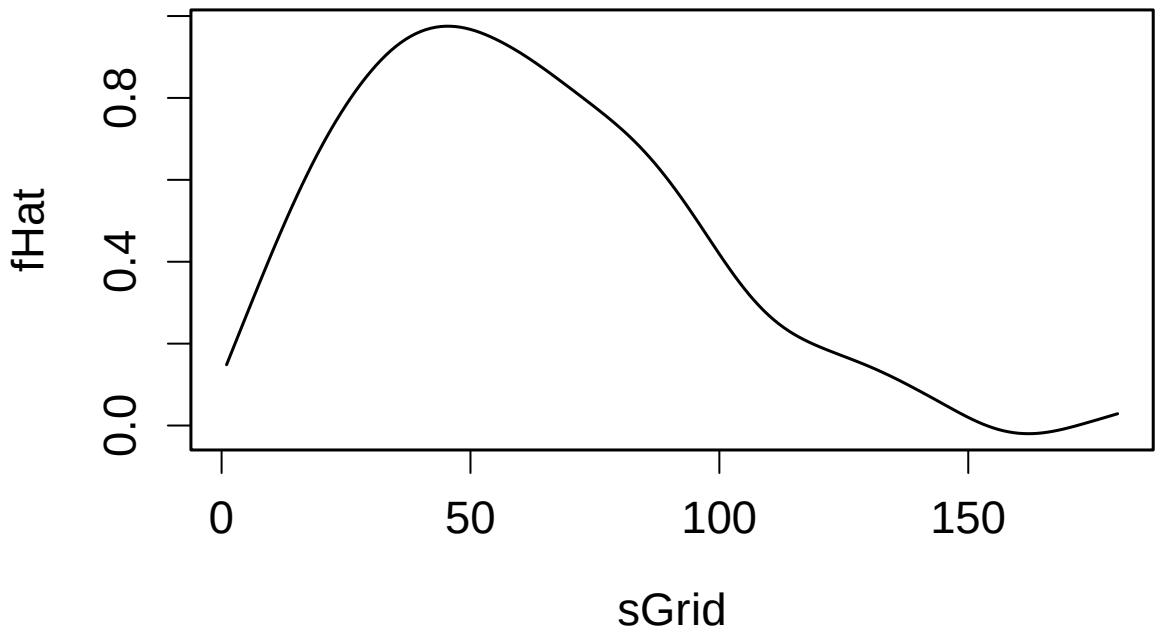

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

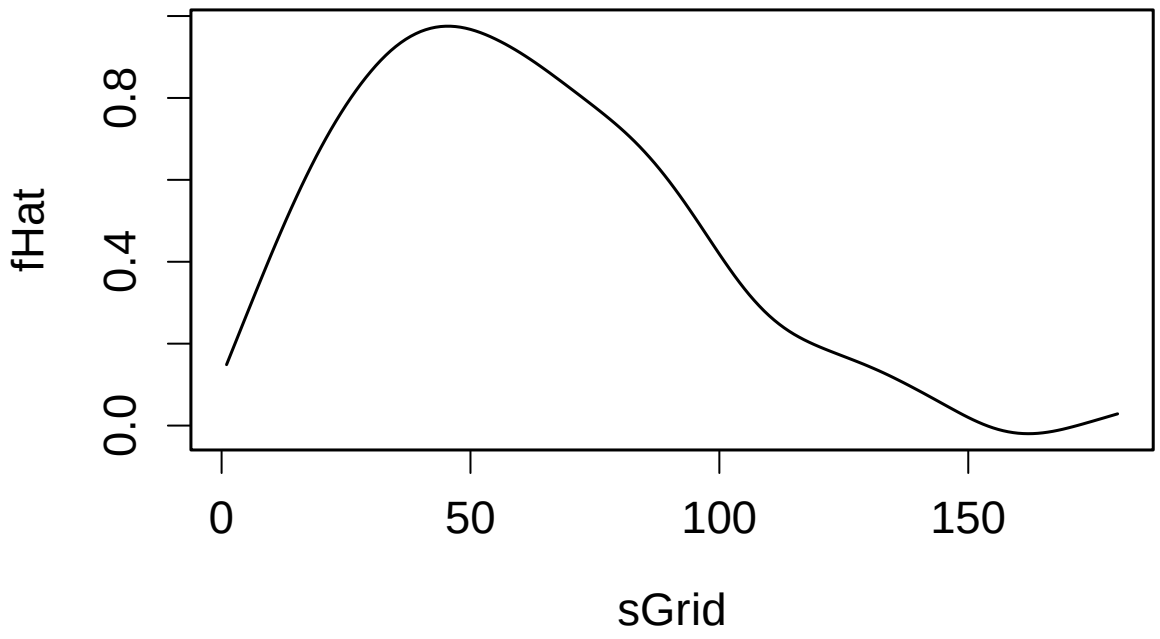

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

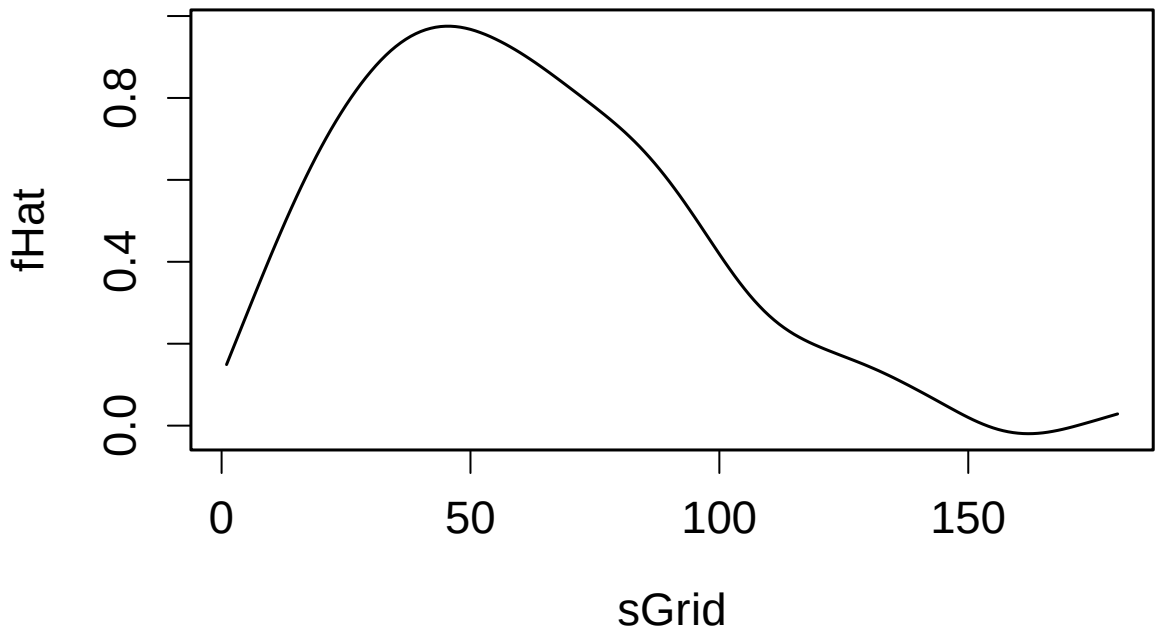

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

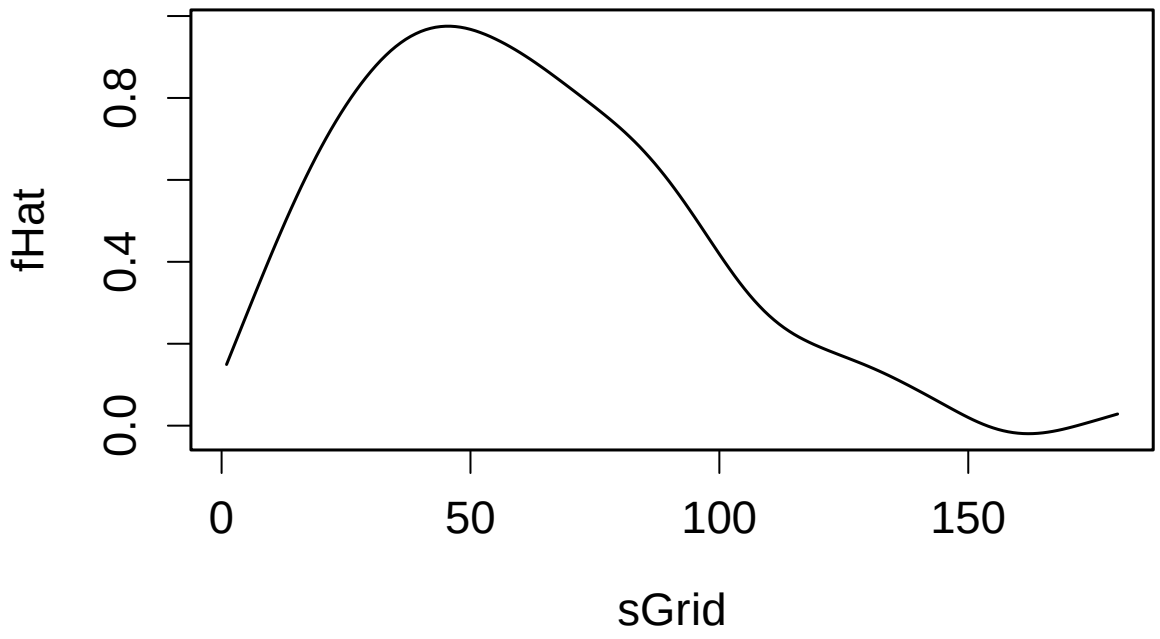

fHat
0.0
0.4
0.8
0
50
100
150
sGrid

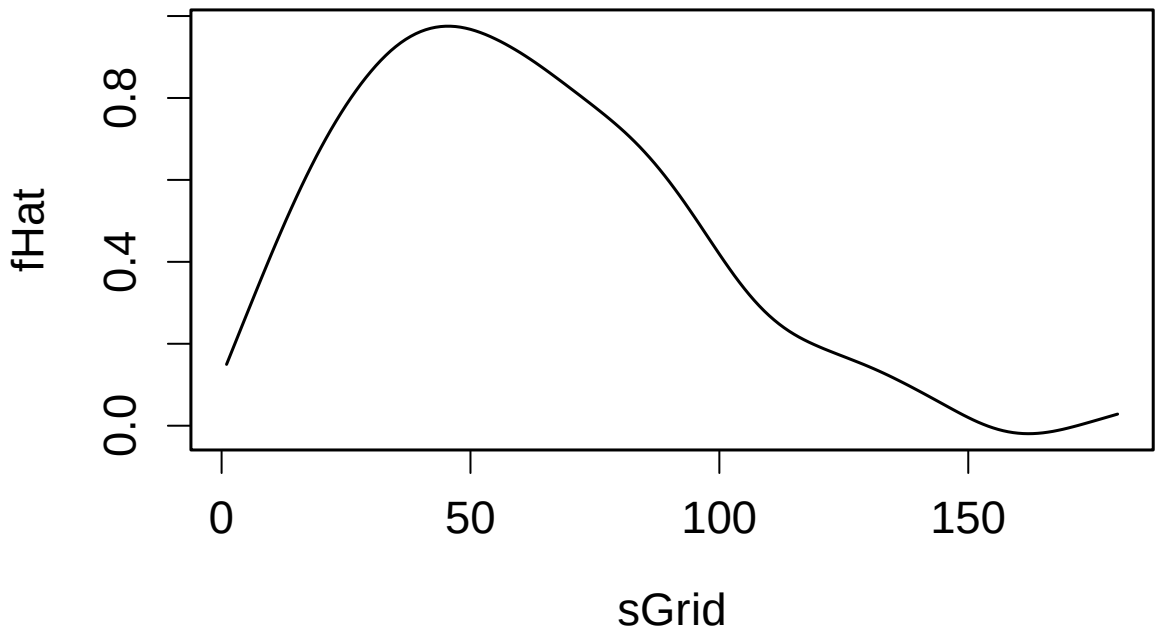

fHat
0.0
0.4
0.8
0
50
100
150
sGrid


fHat
0.0
0.4
0.8
0
50
100
150
sGrid

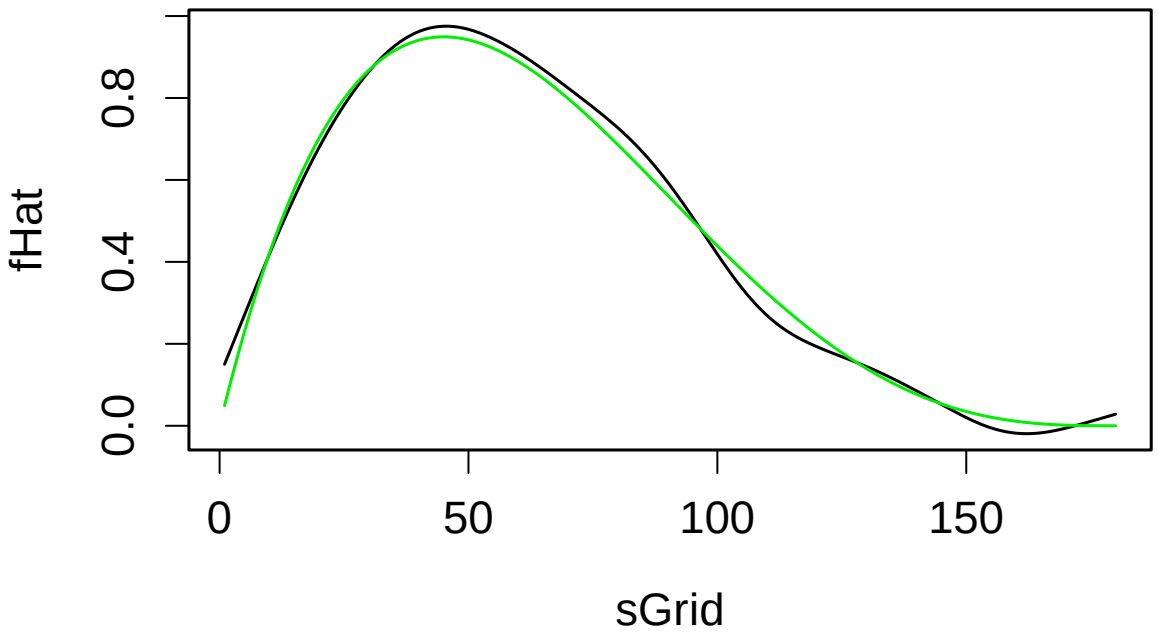

fHat
0.0
0.4
0.8
0
50
100
150
sGrid


fHat
0.0
0.4
0.8
0
50
100
150
sGrid


fHat
0.0
0.4
0.8
0
50
100
150
sGrid




```
lines( sGrid, truef, col=2)
```


fHat
0.0
0.4
0.8
0
50
100
150
sGrid