Ball Drop Example, Bias

Grant Hutchings

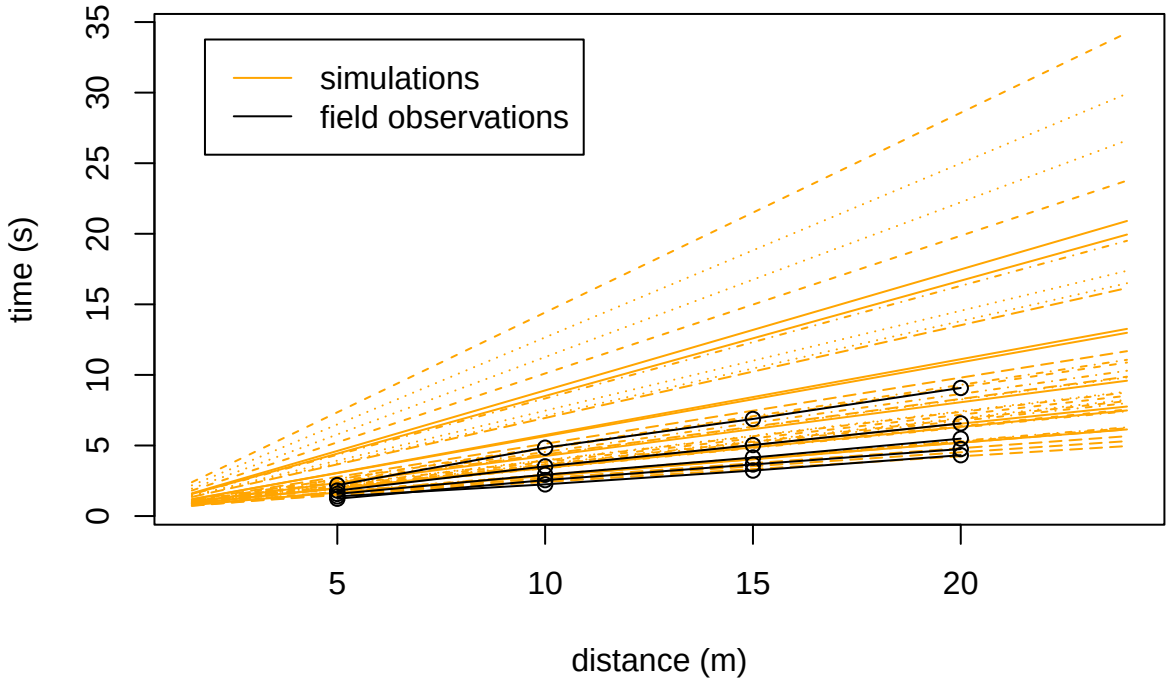
2/23/2022

This example is an extension of the unbiased ball drop example where we now work with a biased simulator. We will generate our observations from the same function as before

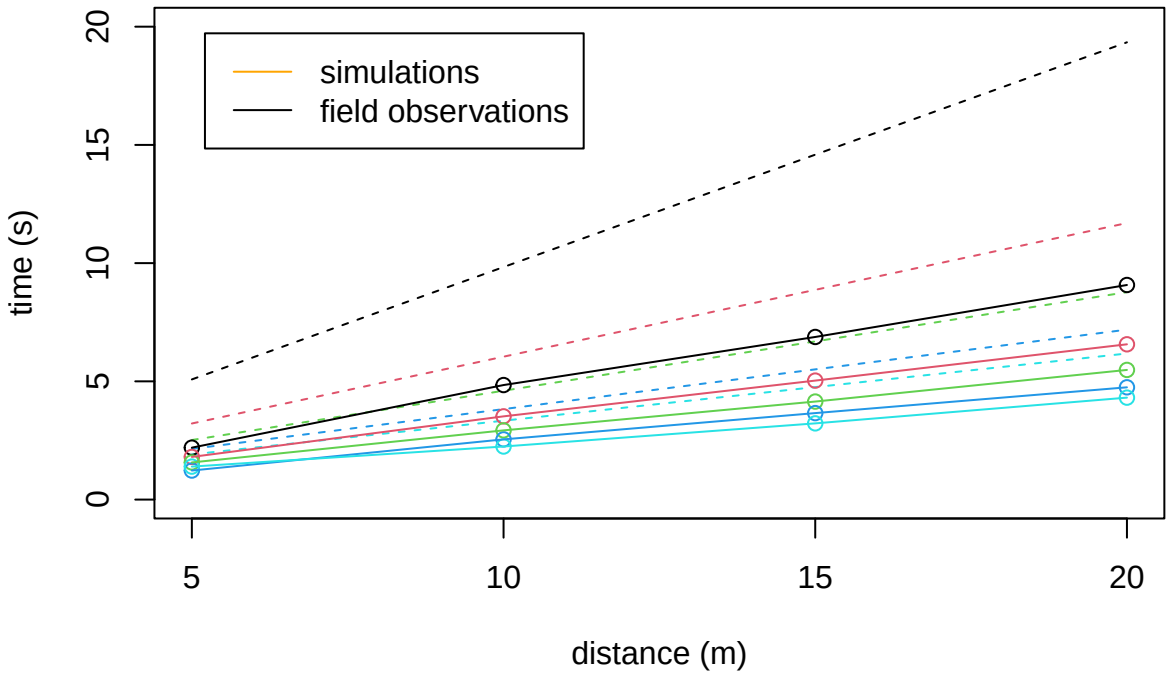
$$t(d) = \frac{acosh(\exp\{Cd/R\})}{\sqrt{Cg/R}}$$

and our simulations from

$$t(d) = \frac{acosh(\exp\{Cd/R\})}{(Cg/R)^{1/4}}.$$



Lets visualize the discrepancy by comparing simulation data and observed data at the same inputs. Observed data is shown as solid lines and simulation data as dashed lines, colors correspond to the 5 different input settings.



Just like in the unbiased case we begin by first pre-scaling the inputs to the unit hyper-cube, and standardizing the outputs to mean zero and unit variance. We then generate basis representations of our outputs, and stretch and compress inputs based on length-scale estimates from a full GP on a subset of the simulations.

```
# step 0: Pre-scale data
XTdata = transform_xt(Xsim,Tsim,Xobs)
Ydata = transform_y(Ysim,YindSim,Yobs,YindObs,center = T,scale = T)

# step 1: get basis
simBasis = get_basis(Ydata$sim$trans,nPC)
obsBasis = get_obs_basis(simBasis,Ydata$obs$trans,YindSim,YindObs,sigY=ifelse(sd_true>0,sd_true^2,1))

# step 2: get length-scale estimates and stretch + compress input space
estLS = mv_lengthscales(XTdata,simBasis$Vt,g=1e-7)
SCinputs = get_SC_inputs(estLS,XTdata,nPC)

mvcData = list(XTdata=XTdata,
               Ydata=Ydata,
               simBasis=simBasis,
               obsBasis=obsBasis,
               estLS=estLS,
               SCinputs=SCinputs)
```

Calibration proceeds as described in the document laGP_calibration.pdf. The calibration likelihood has some interesting gradients which may be due to different number of PC's being chosen for the discrepancy basis. We use the likelihoods evaluated on this space-filling grid to determine suitable initial values of t for the snomadr optimization algorithm.

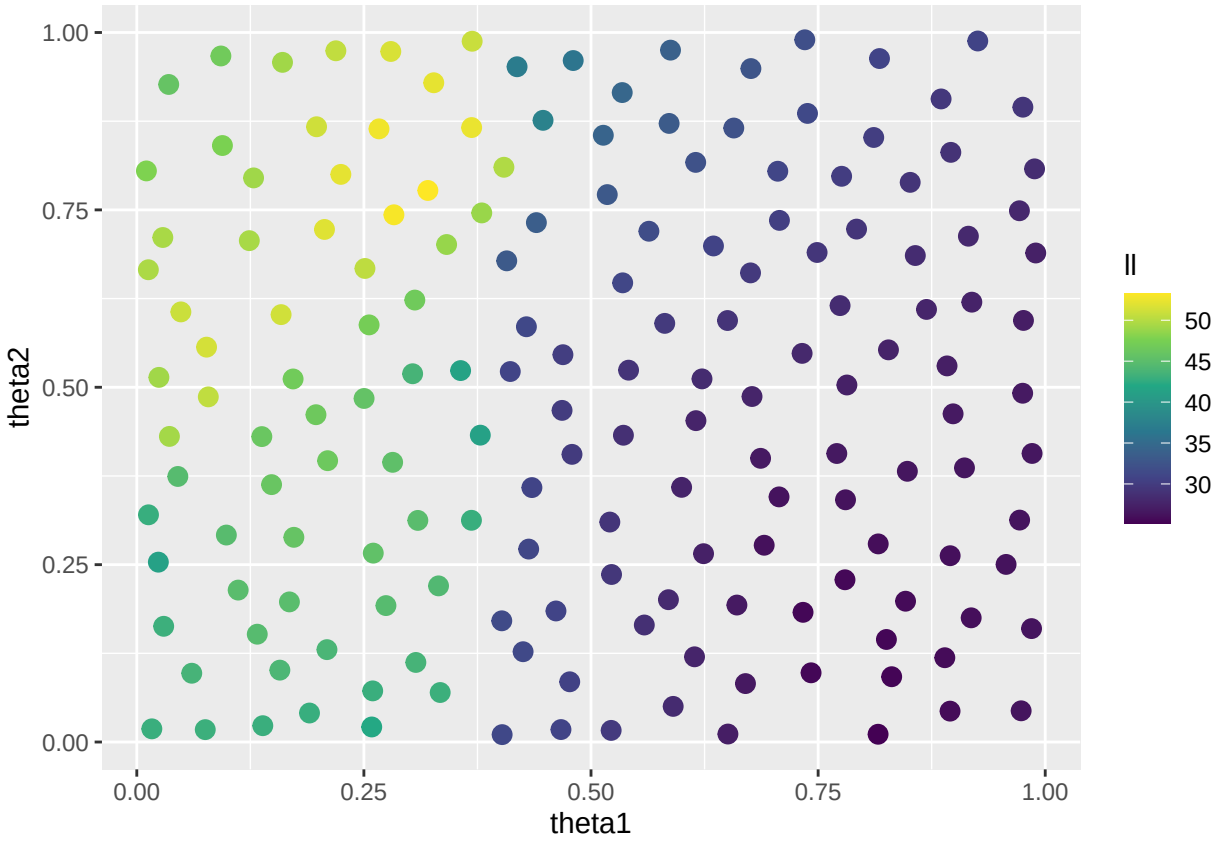
```
nT = 13^2
tInit = dopt.gp(nT, Xcand = lhs(nT*100, matrix(rep(c(0.01,.99),SCinputs$pT),nrow=2,byrow=T)))$XX

calib = mv.calib(mvcData,tInit,nrestarts=3,bias=T)

##
```

```
## iterations: 23
## time:       13
##
## iterations: 26
## time:       12
##
## iterations: 26
## time:       13
```

```
# plot of initial calibration results
ggplot(calib$ll_df, aes(x = theta1, y = theta2, col = ll)) +
  geom_point(size = 3) +  scale_color_viridis_c()
```

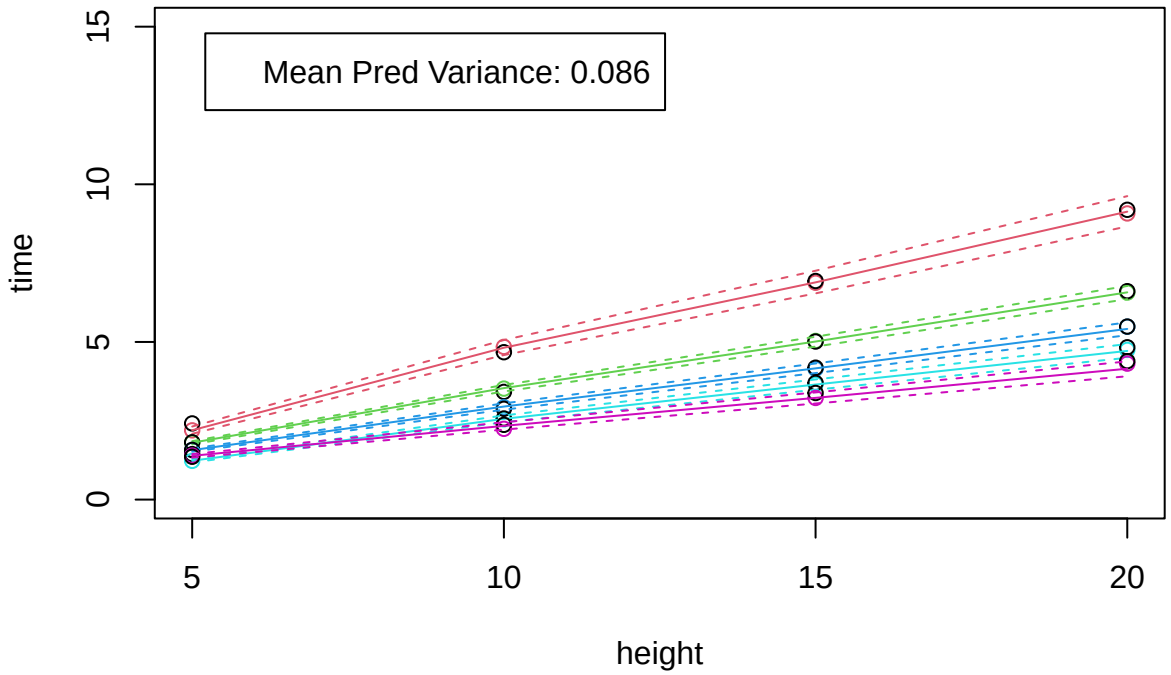



```
## t*: 0.3187801 0.7822348
```

We can see that our estimate of the error variance from the discrepancy GP is too small indicating that the discrepancy GP is over-fitting.

```
## Error Variance Estimation
## true sd: 0.1
## est sd : 0.0477
```

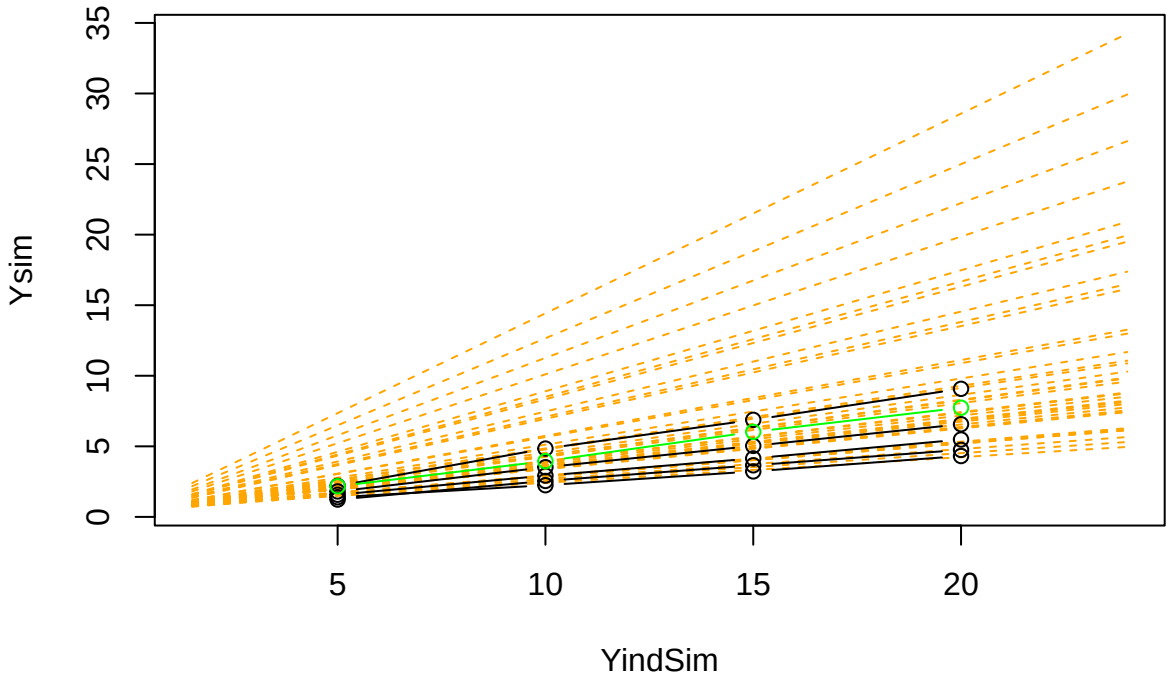
Unsurprisingly we get good prediction for the data points used to train the discrepancy model.

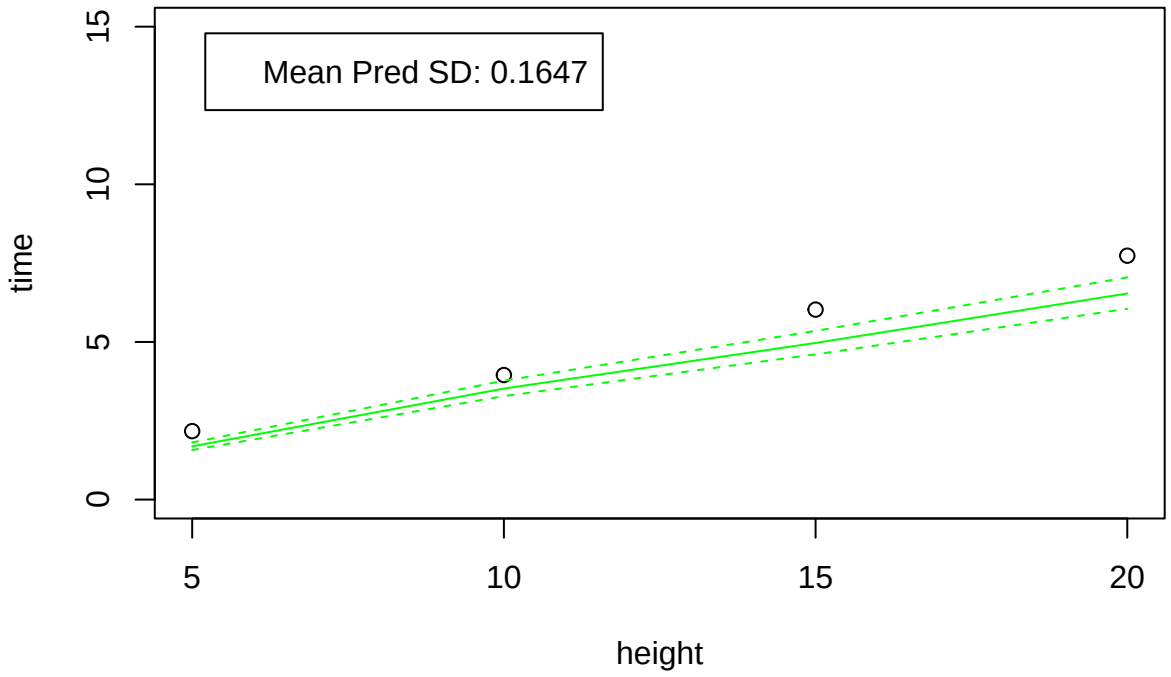


Prediction at new location

The out-of-sample data we would like to predict is shown in green on the plot below and is firmly within the range of both observations and simulations.

Prediction is much worse this new location, with our 95% prediction interval missing the data at all points. This may the the best we can hope for using a flexible discrepancy model like a GP trained on 5 data points.




Mean Pred SD: 0.1647
time
height
0
5
10
15
5
10
15
20