Ball Drop Example, Bias

Grant Hutchings

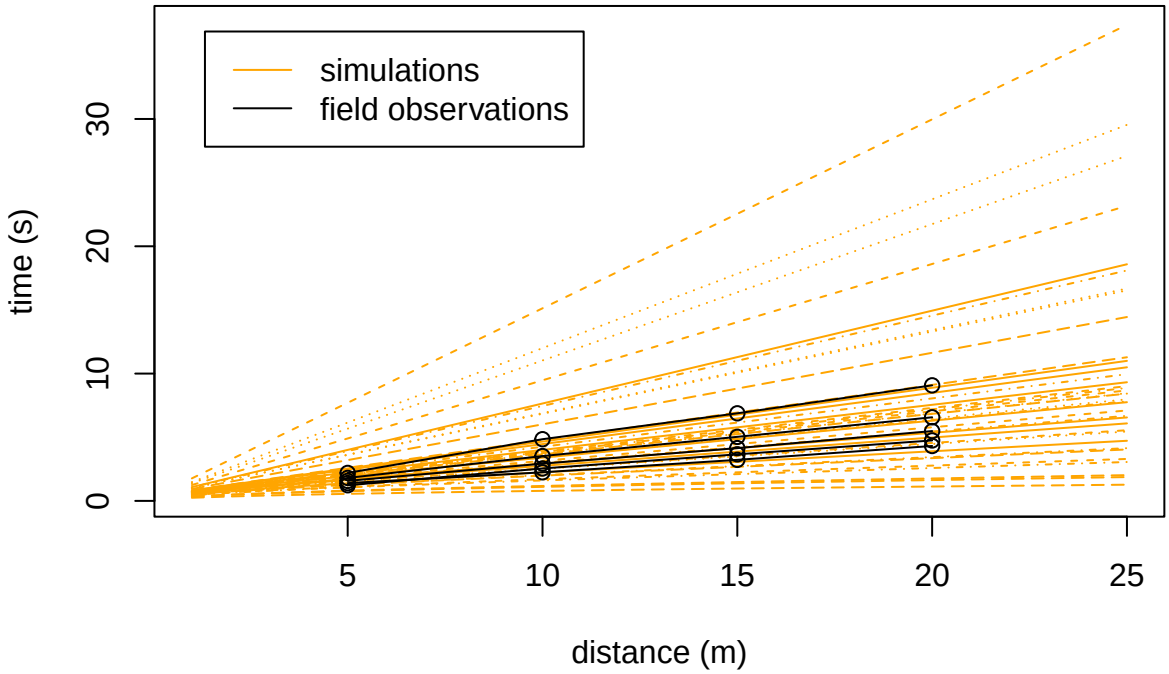
2/23/2022

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

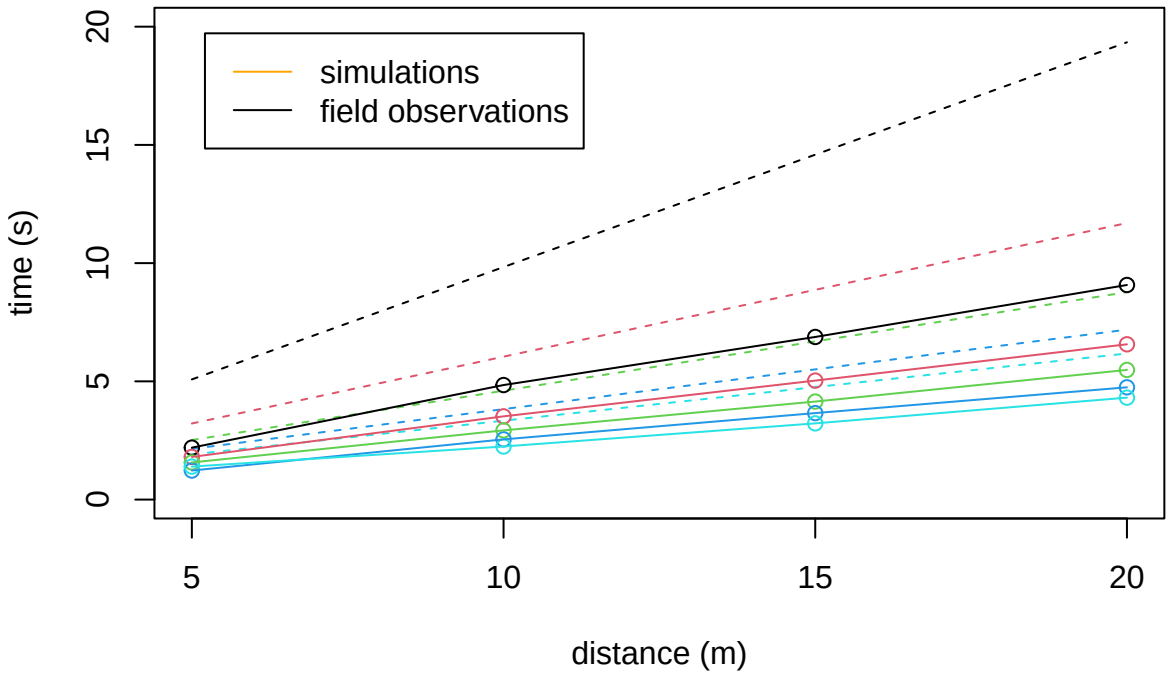
$$t(d) = \frac{acosh(\exp\{Cd/R\})}{\sqrt{Cg/R}}$$

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.

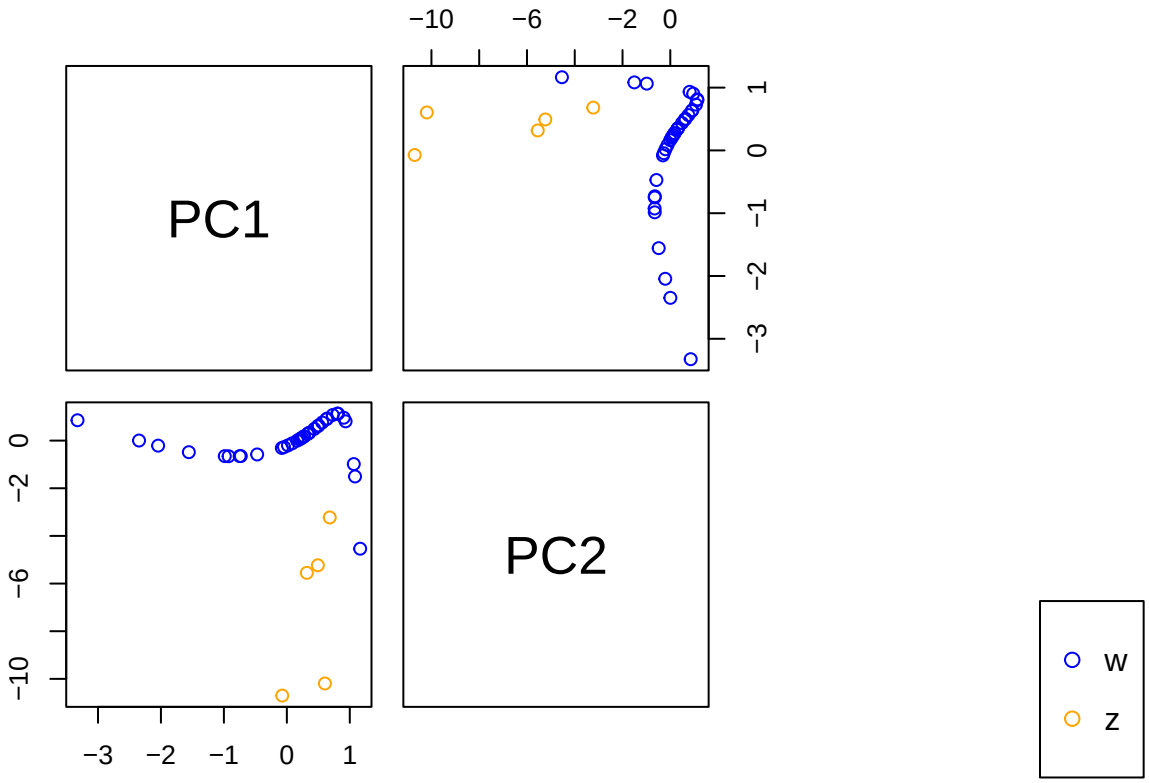


Just like in the unbiased case we begin by first pre-scaling the inputs to the unit hyper-cube, and standardizing the outputs to mean zero and unit variance. We then generate basis representations of our outputs, and stretch and compress inputs based on length-scale estimates from a full GP on a subset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

```
# step 0: Pre-scale data
XTdata = transform_xt(Xsim,Tsim,Xobs,Tobs)
Ydata = transform_y(Ysim,YindSim,Yobs,YindObs,center = T,scale = T, scaletype = 'scalar')

# step 1: get basis
simBasis = get_basis(Ydata$sim$trans,nPC)
obsBasis = get_obs_basis(simBasis,Ydata$obs$trans,YindSim,YindObs)

# Add linear discrepancy
Dsim = matrix(c(rep(1,nrow(YindSim)),YindSim),ncol=2)
Dobs = matrix(c(rep(1,nrow(YindObs)),YindObs),ncol=2)
Dobs = Dobs/sqrt(max(t(Dsim)%*%Dsim))
Dsim = Dsim/sqrt(max(t(Dsim)%*%Dsim))

simBasis$D = Dsim
obsBasis$D = Dobs

# 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,
               YindObs=YindObs,
               YindSim=YindSim)
```

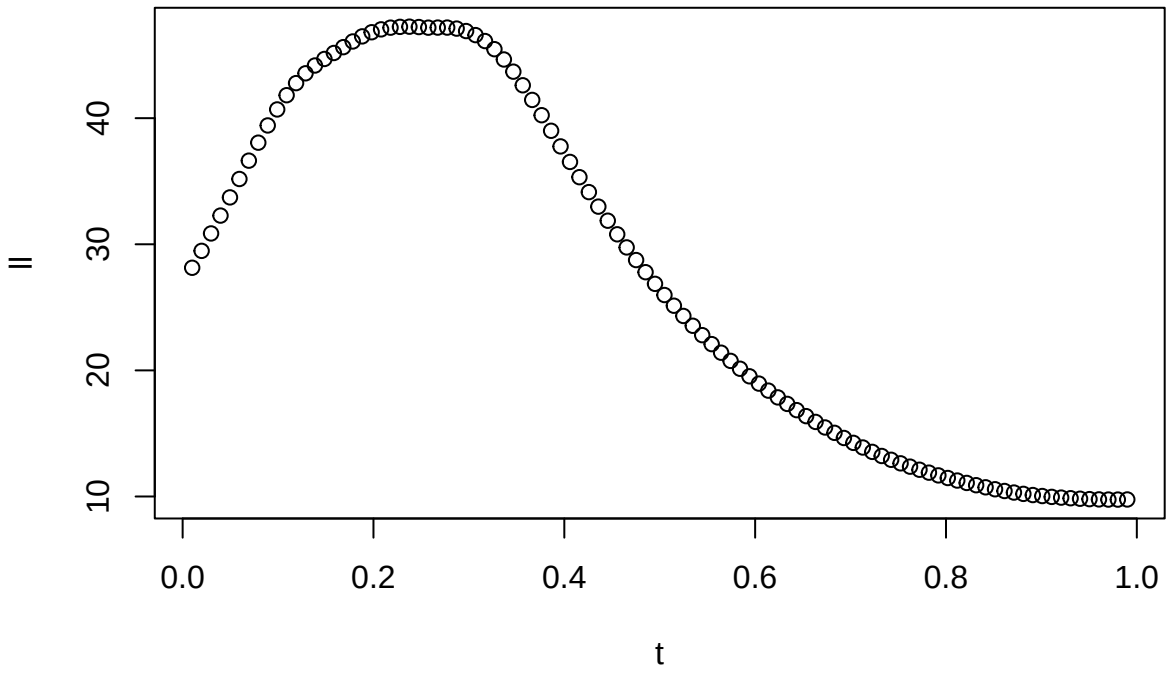
```
py_save_object(mvcData,'bd_bias_C_data.pkl')
```

```
plot_wz_pairs(simBasis$Vt,obsBasis$Vt,legend=T)
```



We start by evaluating the likelihood on a sparse grid, then use the best results as initial values of t for the snomadr optimization algorithm.

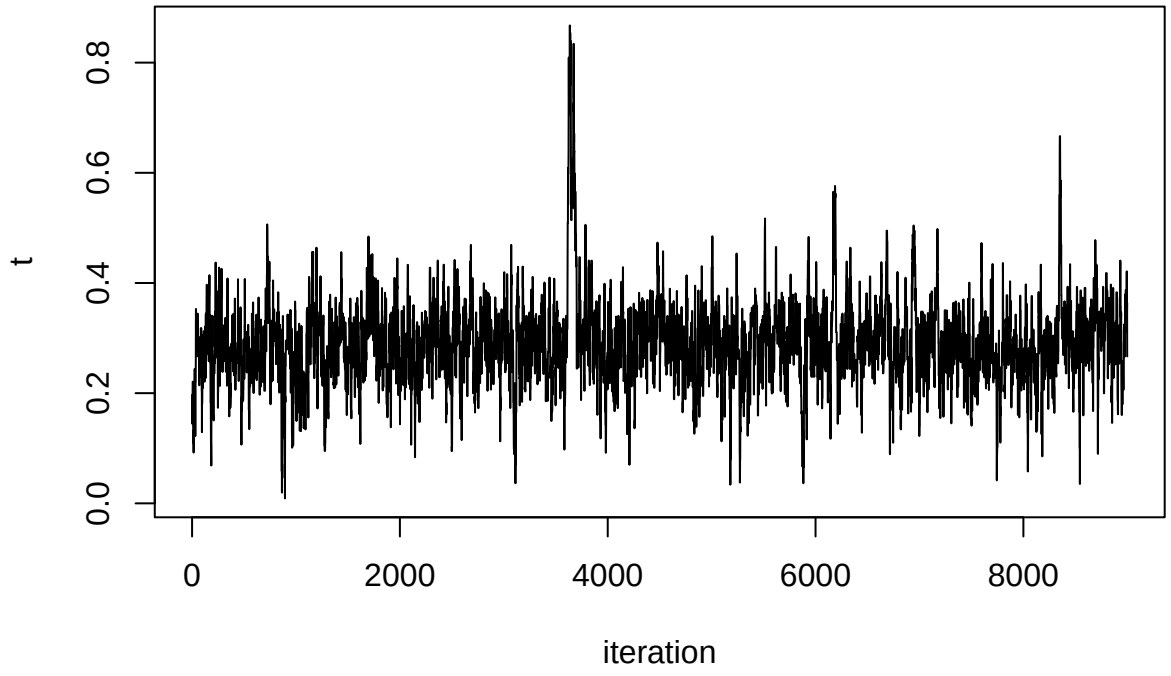
```
nT = 100
tCalib = matrix(seq(.01,.99,length.out=100))
init = calib.init(tCalib,mvcData,bias=T,sample=F)
plot(init$theta1,init$ll,ylab='ll',xlab='t')
```

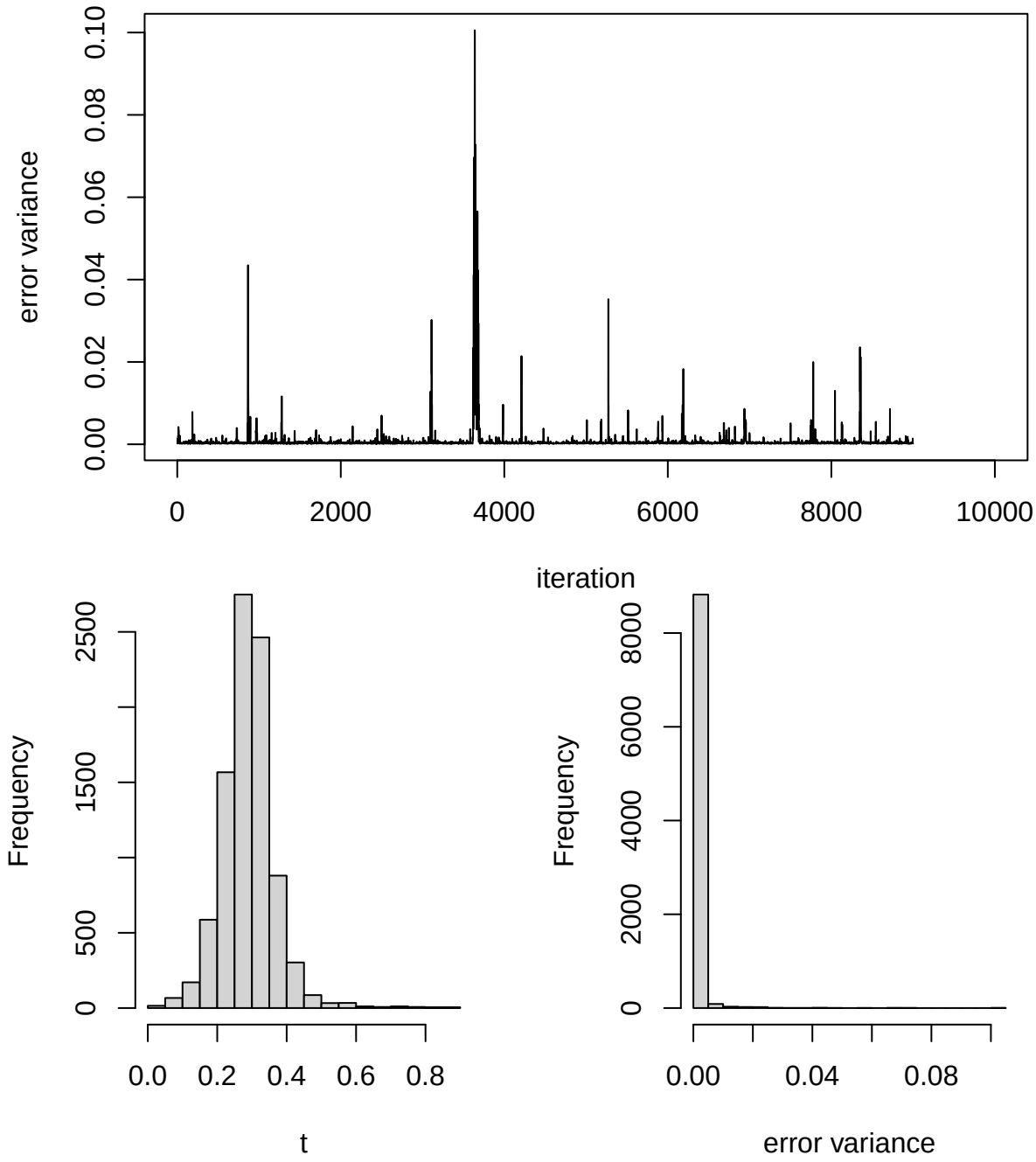



```
calib = mv.calib(mvcData,init,nrestarts=10,bias=T,sample=F)
```

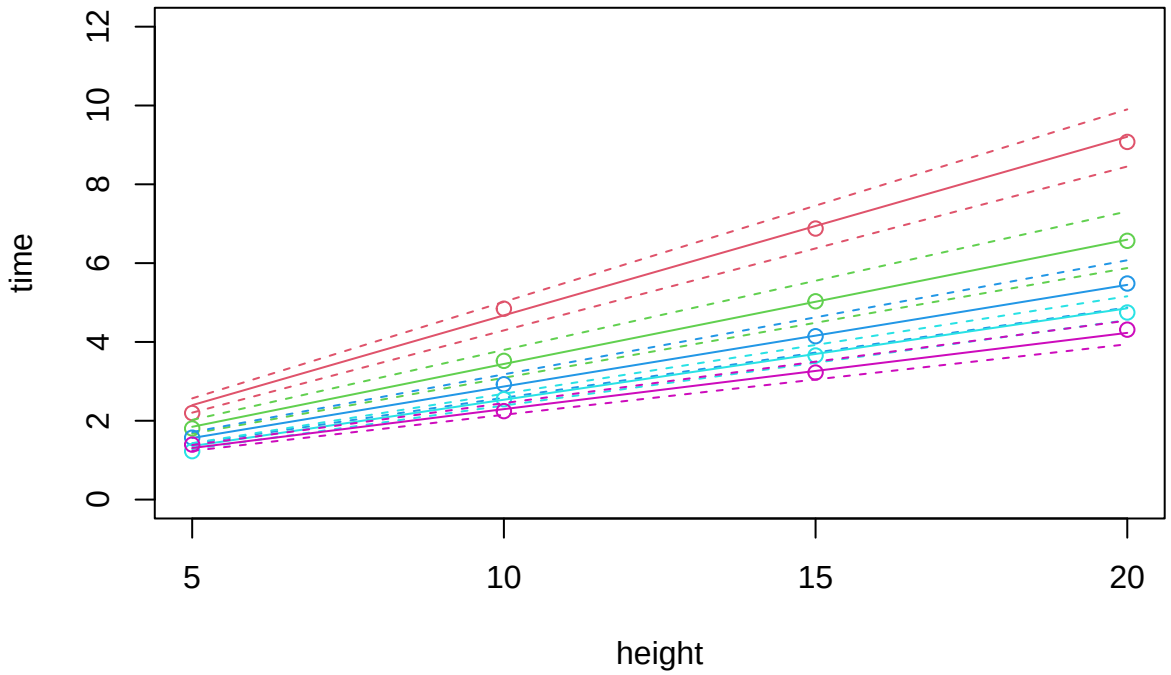
```
## t*: 0.2358277
```

```
## estimated error variance: 0.007324748
```



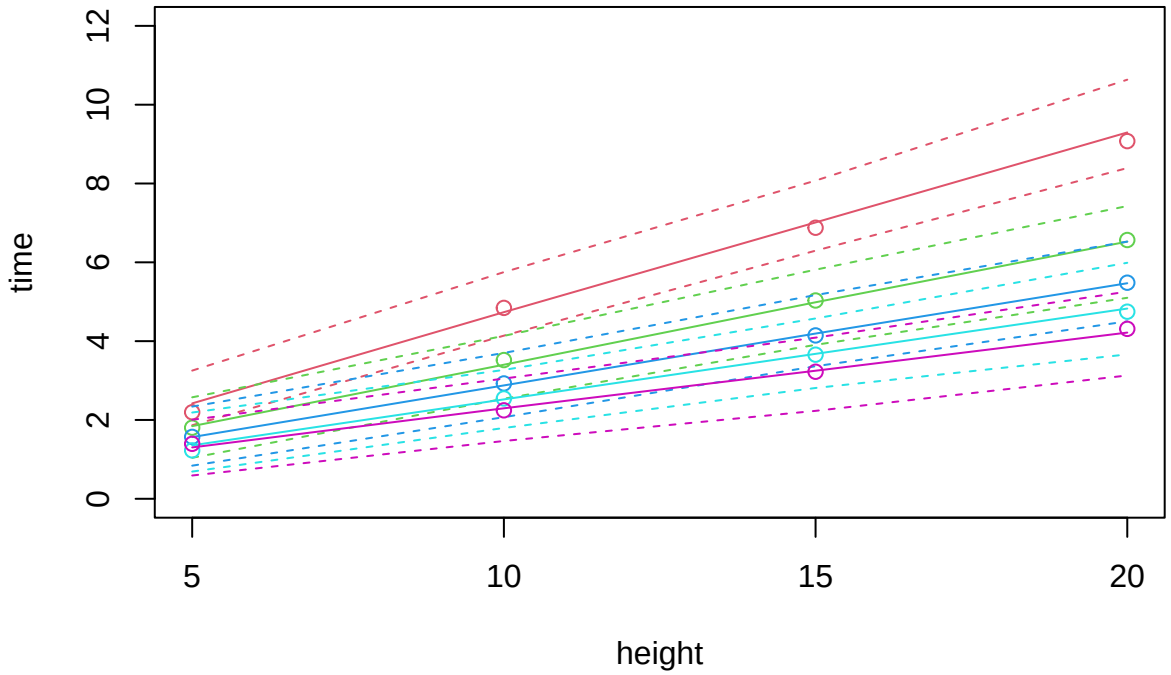


And similarly to `balldrop_nobias_C.Rmd` we compare predictions from the point estimate calibration and MCMC calibration. Prediction from the point estimate mostly covers the data, but there may be some issues with overconfidence at the smallest heights.



We see notable wider confidence intervals, especially for small heights.

```
tsamp = matrix(sample(mcmc.out$t.samp,1000))
yPred = ypred_mcmc(mvcData$XTdata$obs$X$orig[1:n,,drop=F],mvcData,tsamp,support='obs',bias=T)
```

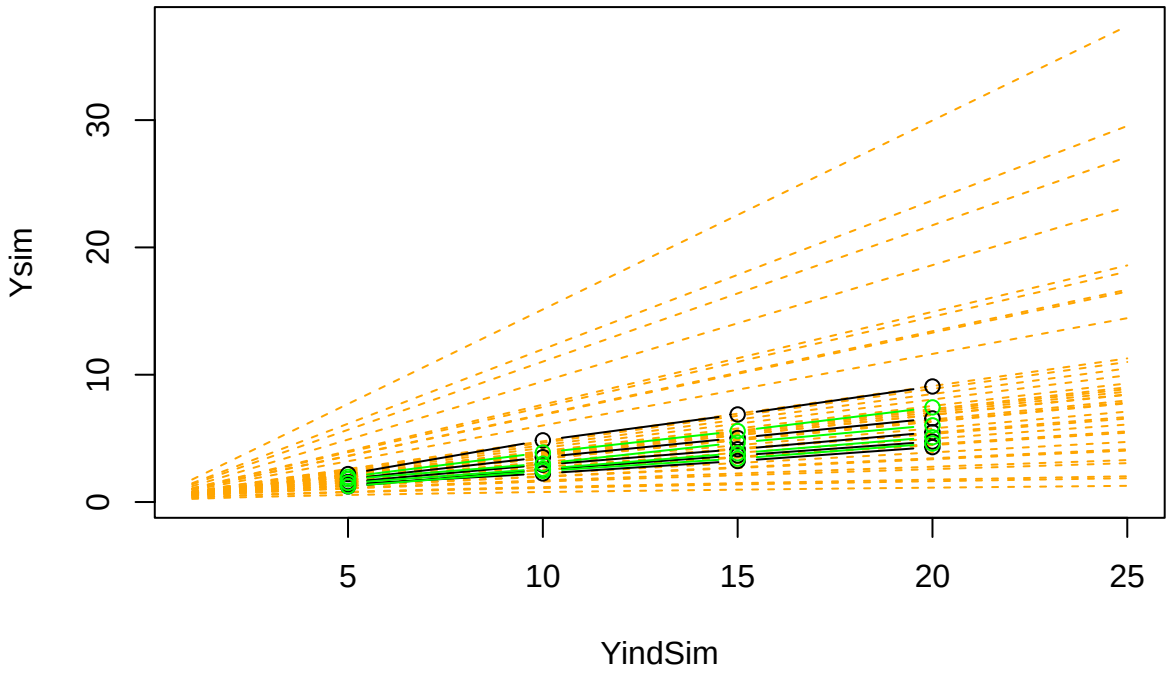


Prediction at new location

The out-of-sample data we would like to predict is shown in green on the fist plot. The second plot shows predictions from MCMC calibration in red and point estimate calibration in green. The point estimate predictions are overconfident at small heights, and in the case of $x = .075$ the mean misses the true values. Mean predictions from the MCMC calibration are good for all x's and we are not overconfident for small

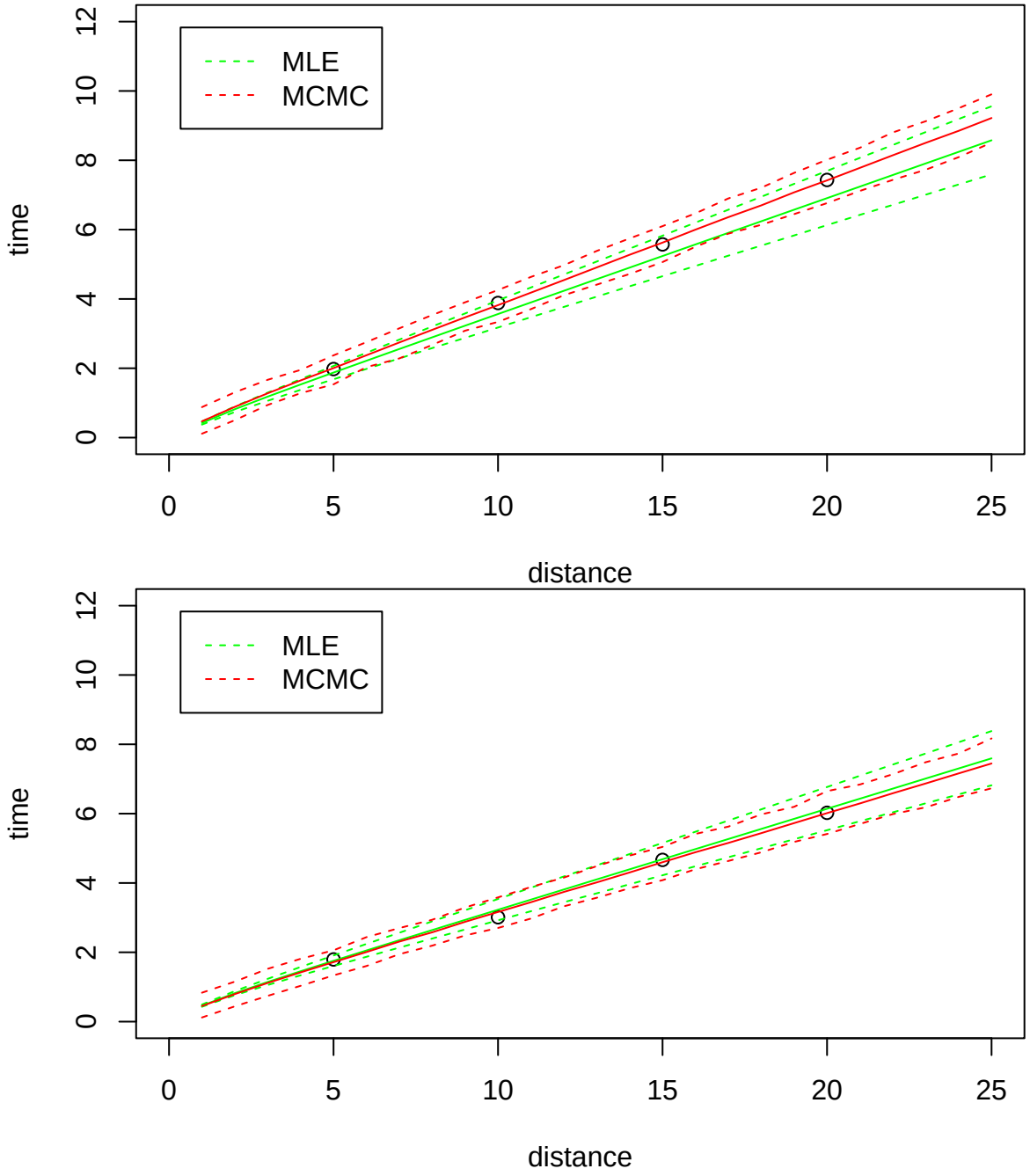
heights.

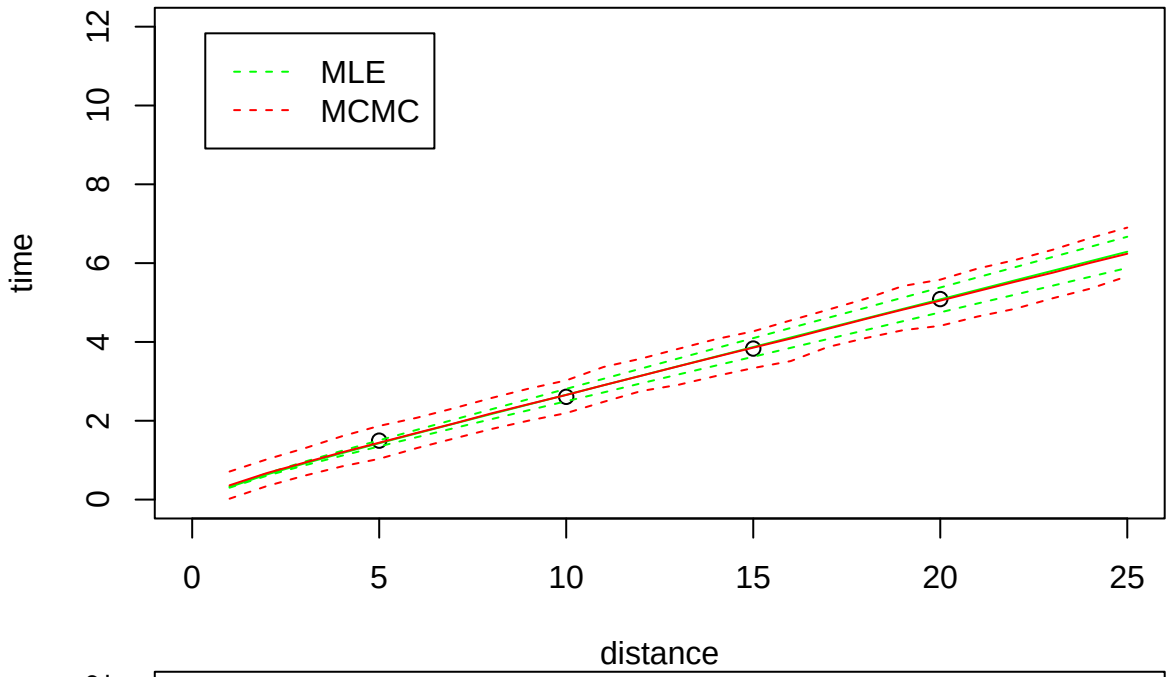
```
set.seed(11)
XobsNew = as.matrix(c(.075,.125,.175,.225))
YobsNew = matrix(nrow=length(YindObs),ncol=nrow(XobsNew))
for(j in 1:nrow(XobsNew)){
  YobsNew[,j] = t_at_h(C=C_true,h=YindObs,R=XobsNew[j],g=g_true) + rnorm(length(YindObs),0,sd_true)
}

matplot(YindSim,Ysim,col='orange',type='l',lty=2)
matplot(YindObs,Yobs,col='black',type='b',lty=1,pch=1,add=T)
matplot(YindObs,YobsNew,col='green',type='b',lty=1,pch=1,add=T)
```

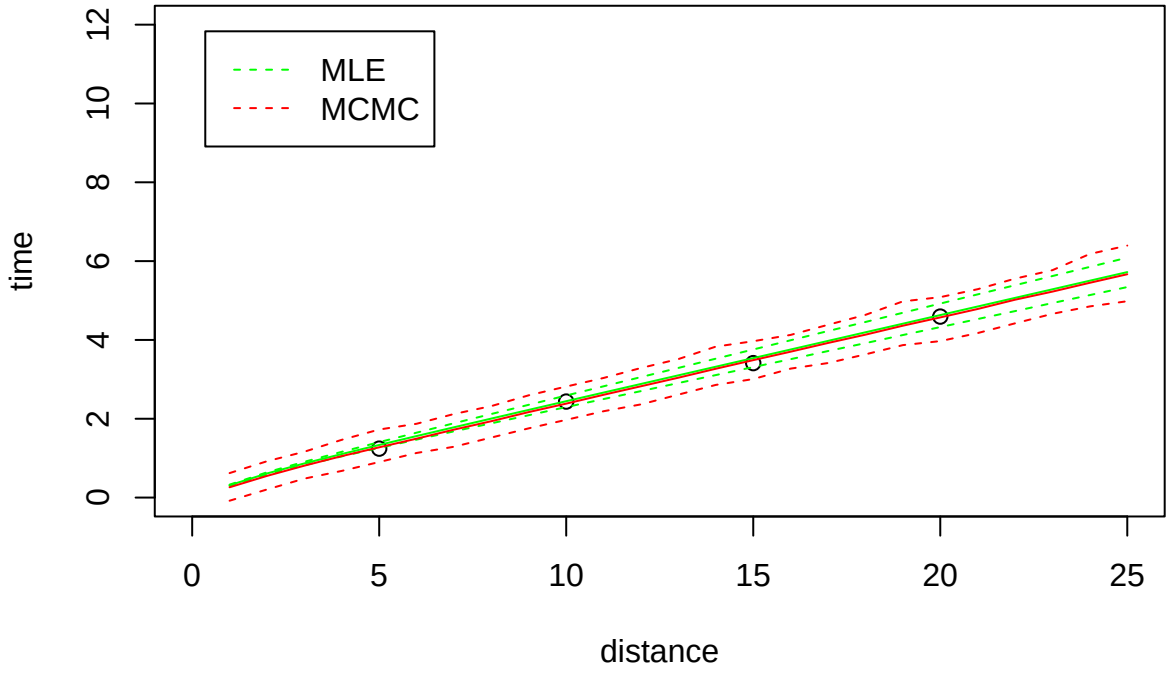



```
XTdataNew = transform_xt(Xsim,Tsim,XobsNew)
YdataNew = transform_y(Ysim,YindSim,YobsNew,YindObs,center = T,scale = T,scaletype = 'scalar')
obsBasisNew = get_obs_basis(simBasis,YdataNew$obs$trans,YindSim,YindObs,sigY=ifelse(sd_true>0,sd_true^2
obsBasisNew$D = Dobs
SCinputsNew = get_SC_inputs(estLS,XTdataNew,nPC)

mvcDataNew = list(XTdata=XTdataNew,
                  Ydata=YdataNew,
                  simBasis=simBasis,
                  obsBasis=obsBasisNew,
                  estLS=estLS,
                  SCinputs=SCinputsNew,
                  YindObs=YindObs,
                  YindSim=YindSim)
py_save_object(mvcDataNew,'bd_bias_oos_C.pkl')

support = 'sim'
if(support=='obs'){
  predInd = YindObs[,1]
} else if(support=='sim'){
```

```
  predInd = YindSim[,1]
}

calib = mv.calib(mvcData,init,nrestarts=10,bias=T,sample=F)
YpredMLE = ypred_mle(XobsNew,mvcDataNew,calib,1000,support = support,bias=T)
YpredMCMC = ypred_mcmc(XobsNew,mvcDataNew,tsamp,support = support,bias=T)
```




MLE
MCMC
time
distance
0
2
4
6
8
10
12
0
5
10
15
20
25


MLE
MCMC
time
distance
0
2
4
6
8
10
12
0
5
10
15
20
25