Some background

Flower-leaf sequences (the order and spacing of flowering and leaf phenological activity) seem to be an important adaptive trait for temperate trees. While there has been interest in this pattern for over a century, only recently have researchers begun to quantify the intra-specific variation in FLS to attempt to better understand the fitness consequences of FLS variation. And by researchers I mean mostly me. There are a number of long term phenological datasets where one can look at inter-annual variaition in FLS, variation amoung indivials, and variations amoung populations, however, to date there haven't been attempt to understand whether this variaiton is mostly controlled by platic response to environment or genetic local adaptation. A common garden, like the one growing at Weld Hill, provides the perfect opportunity to do this.

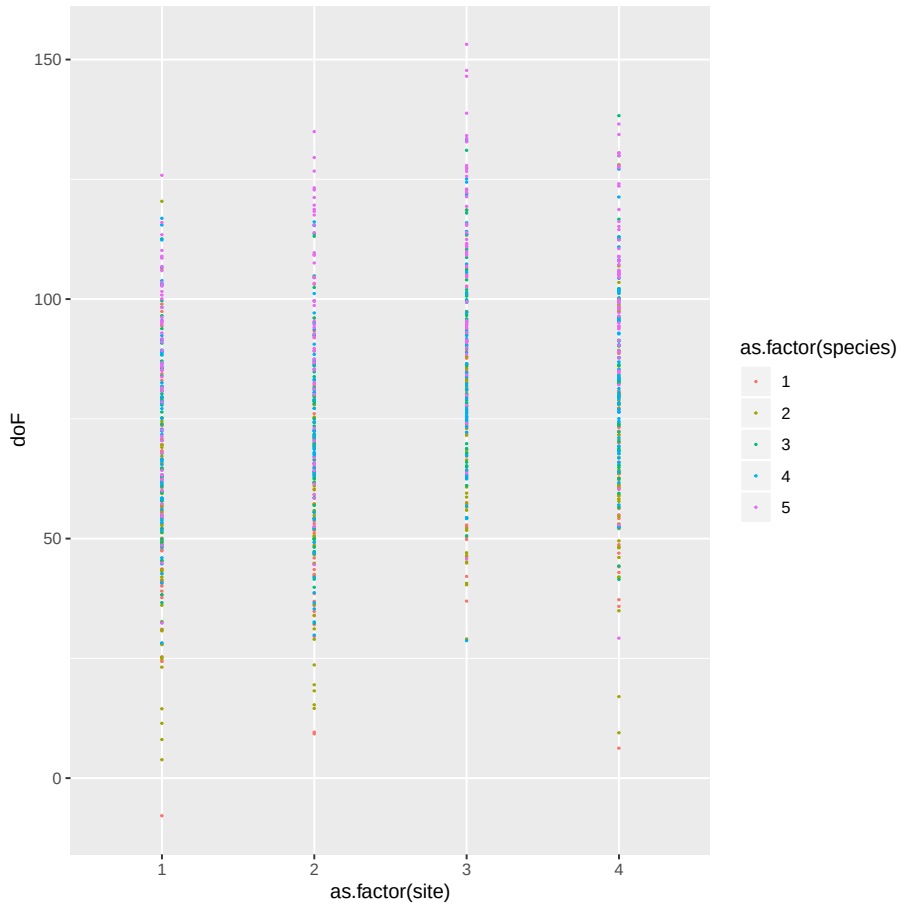
There are many quesitons I could ask with these data. To start simple, I will simply ask are there differences in FLS between the four source populations at Weld Hill. If yes, we will be able to infer there is some genetic control and local adaptation at play for FLS.

To do this, I think it might be best to run an intercept only model with species and site as random effects, but I am open to better ideas.

Data simulation

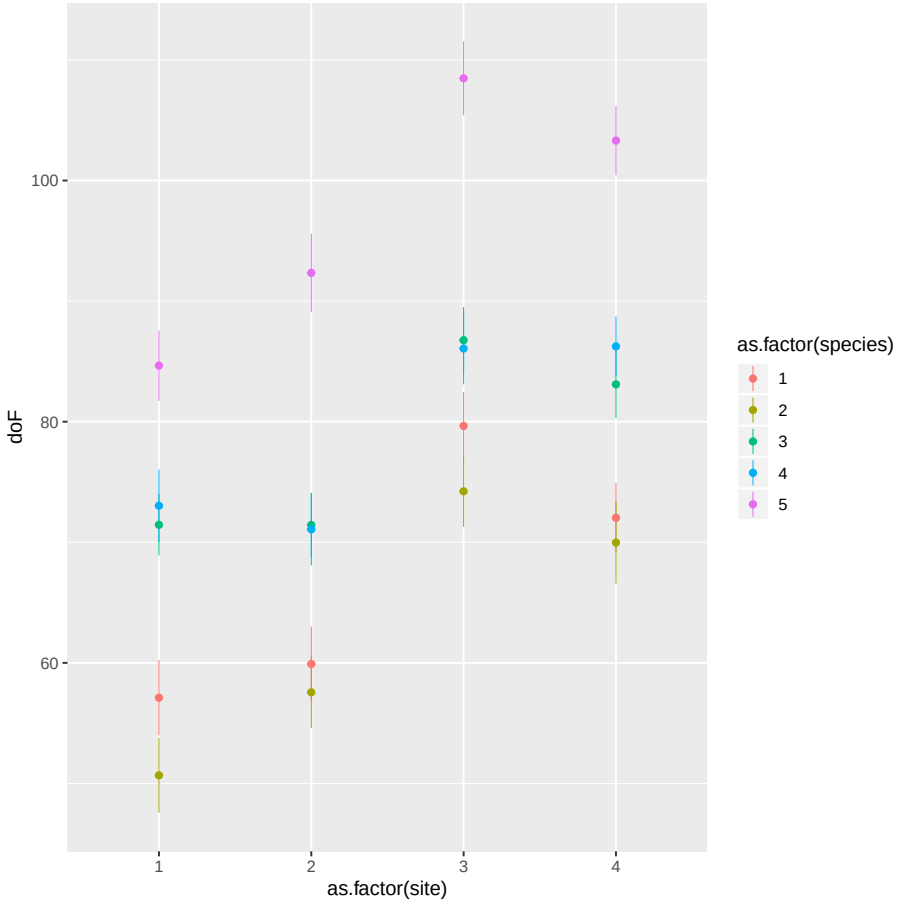
0.1 Simulate the data

```
library(ggplot2)
###first fake data, only sigma on
base<-75 # This is the refernce intercept for  all species and site
spmean<-c(-10,-15,-1, 4, 20) ## species difference in f date
sitemean<-c(-5,-3,11,12) ## site diffrences
sigma<-20
sample<-50 # number of individuals per species per site

df<-data.frame(species=numeric(),site=numeric(),doF=numeric())

for (k in  (1:length(sitemean))){
for (i in (1:length(spmean))){
dfhere<-data.frame(species=seq(length(spmean))[i],site=seq(length(sitemean))[k],doF=rnorm(sa
df<-rbind(df,dfhere)
 }
}

ggplot(df, aes(as.factor(site),doF))+geom_point(aes(color=as.factor(species)),size=0.2)
```




```
ggplot(df, aes(as.factor(site),doF))+stat_summary(aes(color=as.factor(species)),size=0.2)
```

```
## No summary function supplied, defaulting to 'mean_se()
```



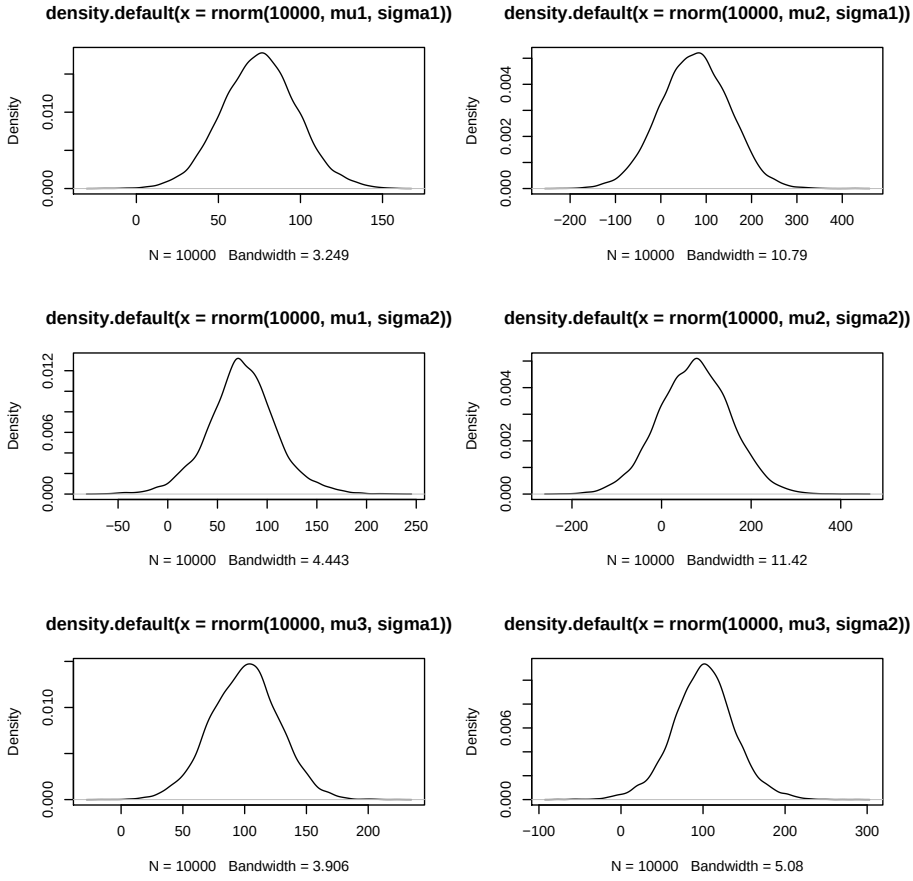
Questions:

1. Are there better ways to simulate the variation amoung sites and species?
2. Did I actually simulatee nested data?

Prior predictive checks

```
mu1<-rnorm(1e4,75,20)
mu2<-rnorm(1e4,75,75)
mu3<-rnorm(1e4,100,25)

sigma2<-runif(1e4,0,50)
sigma1<-runif(1e4,0,20)
```

```
###prior predictive check plots
par(mfrow = c(3,2))
plot(density(rnorm(1e4,mu1,sigma1)))
plot(density(rnorm(1e4,mu2,sigma1)))
plot(density(rnorm(1e4,mu1,sigma2)))
plot(density(rnorm(1e4,mu2,sigma2)))
plot(density(rnorm(1e4,mu3,sigma1)))
plot(density(rnorm(1e4,mu3,sigma2)))
```



model Fit

```
library(brms)

## Loading required package:  Rcpp
```

```
## Registered S3 method overwritten by 'xts':
##  method    from
##  as.zoo.xts zoo
## Loading 'brms' package (version 2.9.0).  Useful instructions
## can be found by typing help('brms').  A more detailed introduction
## to the package is available through vignette('brms_overview').
```

```
get_prior(doF~1+(1|species/site),data = df, family = gaussian())
```

```
##                  prior     class      coef        group resp dpar nlpar
## 1 student_t(3, 78, 26) Intercept
## 2  student_t(3, 0, 26)        sd
## 3                             sd                species
## 4                             sd Intercept      species
## 5                             sd           species:site
## 6                             sd Intercept species:site
## 7  student_t(3, 0, 26)     sigma
##   bound
## 1
## 2
## 3
## 4
## 5
## 6
## 7
```

```
###set py priors
priorz<- prior(normal(75,20), class="Intercept")+prior(normal(0, 20),class="sd")+
  prior(normal(0, 20),class="sigma")


test.mod<- brm(doF~1+(1|species/site), data = df,
               family =gaussian(),prior=priorz,iter=3000, warmup=2000,control=list(adapt_del
```

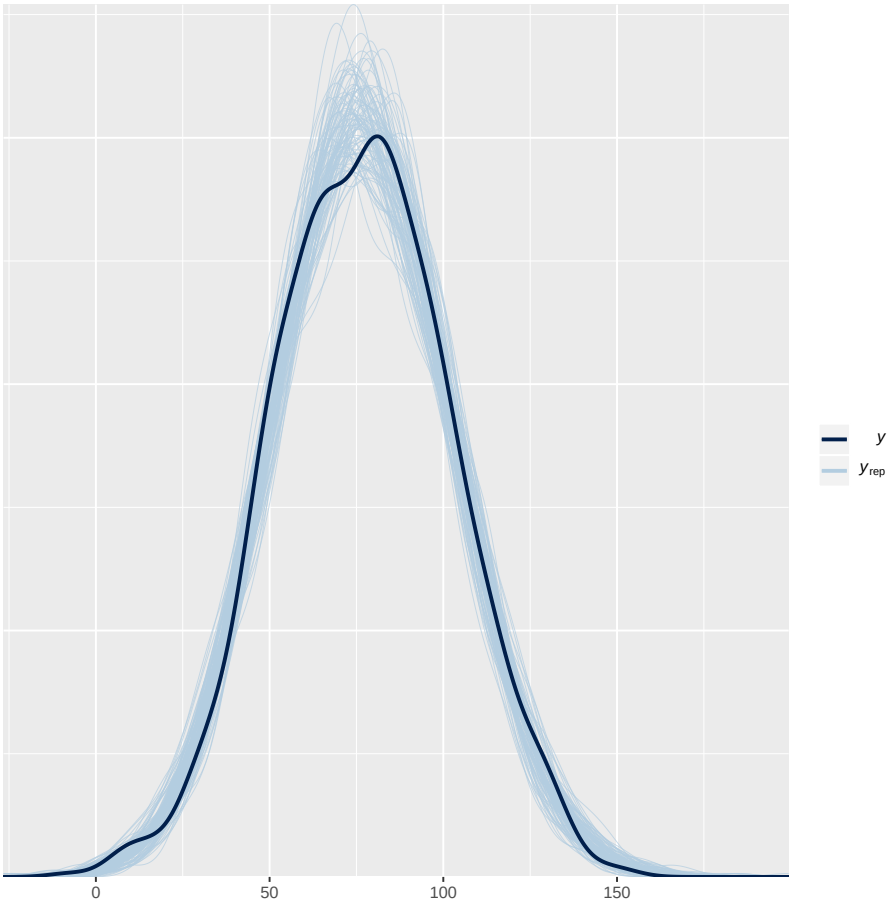
```
## Compiling the C++ model
## Start sampling
```

```
##
## SAMPLING FOR MODEL 'd65512d186e39b04c5893c2b6aa8eb79' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000296 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.96 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 1: Iteration:  300 / 3000 [ 10%]  (Warmup)
```

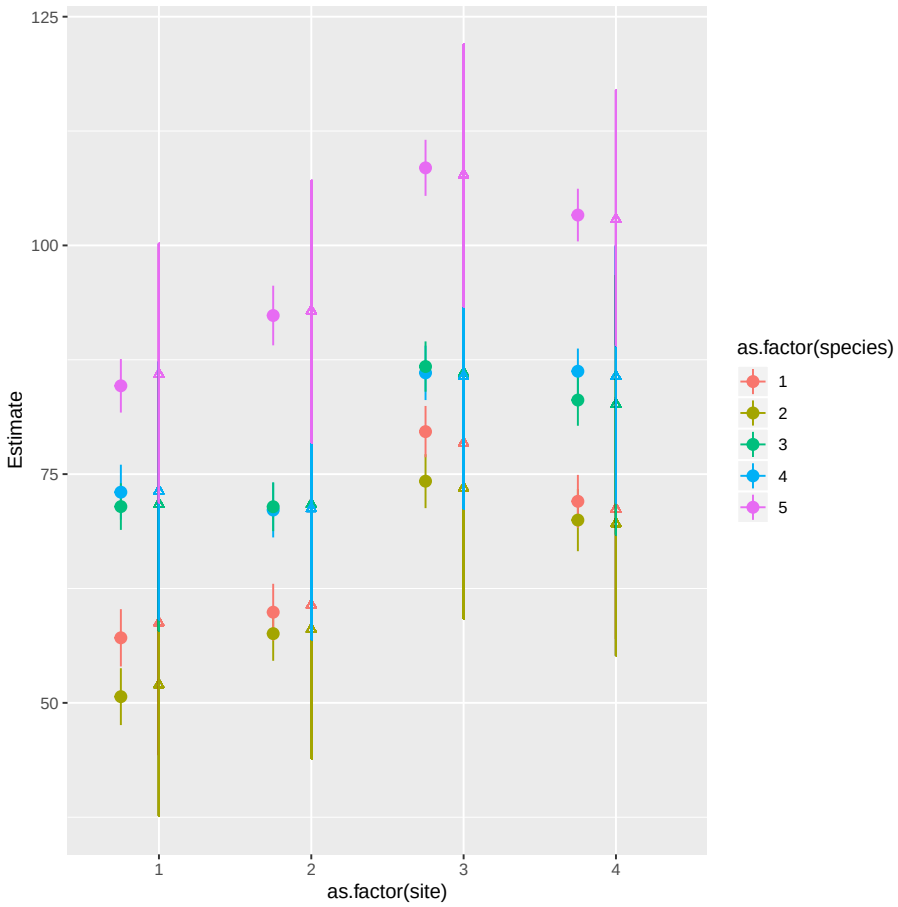
```
## Chain 1: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 1: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1200 / 3000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1500 / 3000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1800 / 3000 [ 60%]  (Warmup)
## Chain 1: Iteration: 2001 / 3000 [ 66%]  (Sampling)
## Chain 1: Iteration: 2300 / 3000 [ 76%]  (Sampling)
## Chain 1: Iteration: 2600 / 3000 [ 86%]  (Sampling)
## Chain 1: Iteration: 2900 / 3000 [ 96%]  (Sampling)
## Chain 1: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 15.9325 seconds (Warm-up)
## Chain 1:                6.67006 seconds (Sampling)
## Chain 1:                22.6026 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'd65512d186e39b04c5893c2b6aa8eb79' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 7e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.7 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 2: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 2: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 2: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1200 / 3000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1500 / 3000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1800 / 3000 [ 60%]  (Warmup)
## Chain 2: Iteration: 2001 / 3000 [ 66%]  (Sampling)
## Chain 2: Iteration: 2300 / 3000 [ 76%]  (Sampling)
## Chain 2: Iteration: 2600 / 3000 [ 86%]  (Sampling)
## Chain 2: Iteration: 2900 / 3000 [ 96%]  (Sampling)
## Chain 2: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 15.3111 seconds (Warm-up)
## Chain 2:                6.93498 seconds (Sampling)
## Chain 2:                22.2461 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'd65512d186e39b04c5893c2b6aa8eb79' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 7.4e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.74 seconds.
```

```
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 3: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 3: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 3: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1200 / 3000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1500 / 3000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1800 / 3000 [ 60%]  (Warmup)
## Chain 3: Iteration: 2001 / 3000 [ 66%]  (Sampling)
## Chain 3: Iteration: 2300 / 3000 [ 76%]  (Sampling)
## Chain 3: Iteration: 2600 / 3000 [ 86%]  (Sampling)
## Chain 3: Iteration: 2900 / 3000 [ 96%]  (Sampling)
## Chain 3: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 15.889 seconds (Warm-up)
## Chain 3:                7.04252 seconds (Sampling)
## Chain 3:                22.9315 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'd65512d186e39b04c5893c2b6aa8eb79' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 7.1e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.71 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 4: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 4: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 4: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 4: Iteration: 1200 / 3000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1500 / 3000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1800 / 3000 [ 60%]  (Warmup)
## Chain 4: Iteration: 2001 / 3000 [ 66%]  (Sampling)
## Chain 4: Iteration: 2300 / 3000 [ 76%]  (Sampling)
## Chain 4: Iteration: 2600 / 3000 [ 86%]  (Sampling)
## Chain 4: Iteration: 2900 / 3000 [ 96%]  (Sampling)
## Chain 4: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 16.3745 seconds (Warm-up)
## Chain 4:                7.89052 seconds (Sampling)
## Chain 4:                24.265 seconds (Total)
## Chain 4:
```

```
pp_check(test.mod,nsamples=100)
```




```
new.data<-data.frame(species=rep(1:5,each=4),site=rep(1:4,by=5))
predy<-predict(test.mod,probs =c(0.25,.75),newdata=new.data)
new.data<-cbind(new.data,predy)

new.data<-dplyr::left_join(new.data,df)

## Joining, by = c("species", "site")

p1<-ggplot(new.data, aes(as.factor(site),Estimate))+geom_point(aes(color=as.factor(species))
p1+stat_summary(data=df, aes(as.factor(site),doF,color=as.factor(species)),position = positi

## No summary function supplied, defaulting to `mean_se()
## Warning in max(table(panel$xmin)):  no non-missing arguments to
max; returning -Inf
```


125
100
75
50
Estimate
1
2
3
4
as.factor(site)
as.factor(species)
1
2
3
4
5