Energetics

Load the data

```
library(rgdal)
library(plotly)
library(ggplot2)
library(dismo)
library(dplyr)
library(insol)
load("/home/rstudio/morph/data/test.rob")
Lat<-median(sites$lat) ## Get lat and long which are used in some functions such as day length
Lon<-median(sites$lon)
sites$mean_biomass[is.na(sites$mean_biomass)]<-0 ## Set not available to zero
map<-gmap(grat,type="satellite")
```

A function to make a time stamp

```
FMakeTime<-function(year=2016,month=1,day=1,hr=1){
  tm<-sprintf("%04d-%02d-%02d %02d:00:00",year,month,day,hr)
  tm<-as.POSIXct(tm)
  tm
}
FMakeDate<-function(ftm=tm)format(ftm,"%Y-%m-%d")
tm<-FMakeTime()
tm
```

```
## [1] "2016-01-01 01:00:00 UTC"
```

```
FMakeDate(tm)
```

```
## [1] "2016-01-01"
```

```
tm+60*60
```

```
## [1] "2016-01-01 02:00:00 UTC"
```

A Utility function to calculate day or night

Use the insol package. This produces an object with sunrise and sunset times, so if the hour falls between them it is day.

```
FIsDay<-function(tm,Lat=55.32,Lon=-162.8)
{
hr<-as.numeric(format(tm, format='%H'))
day_len<-data.frame(daylength(Lat, Lon,JD(tm), tmz=-10))
isday<-ifelse(hr>day_len$sunrise & hr< day_len$sunset,"Day","Night")
isday}

tm<-FMakeTime(2016,1,1,10)
FIsDay(tm)
```

```
## [1] "Day"
```

```
tm<-FMakeTime(2016,1,1,7)
FIsDay(tm)
```

```
## [1] "Night"
```

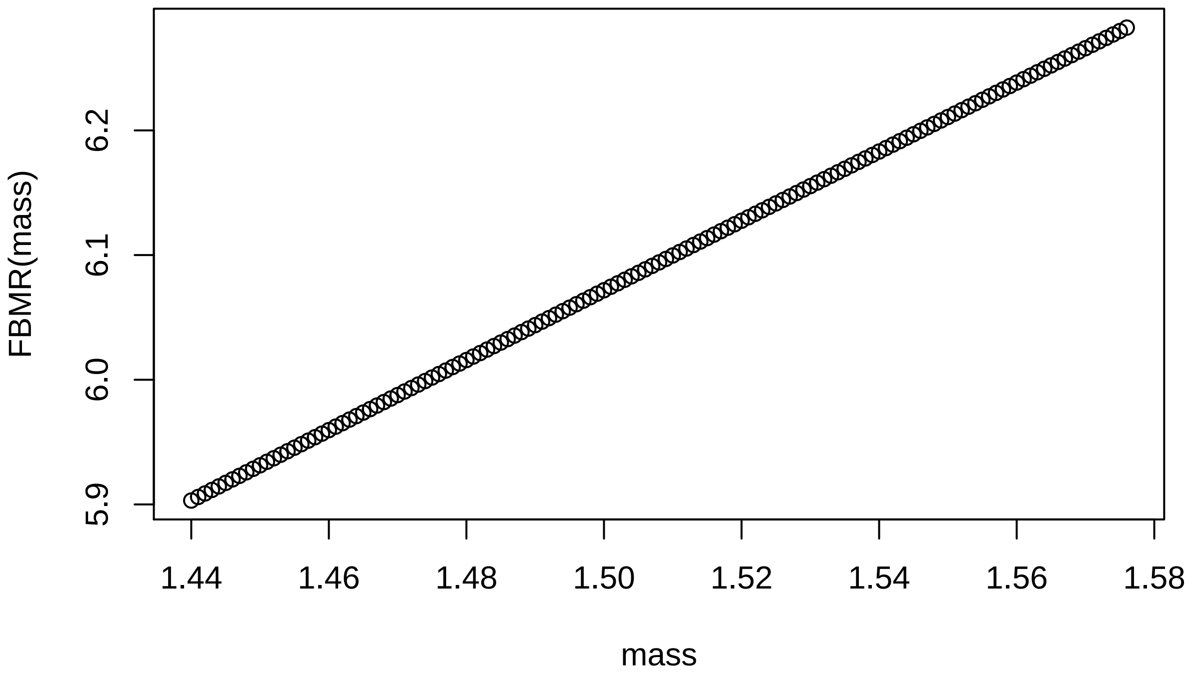
Calculate basal metabolic rate (BMR)

This is the function taken from Humbolt paper

```
FBMR <- function(mass)4.59*mass^0.69
FBMR(1.4)
```

```
## [1] 5.7895
```

```
mass<-(1440:1576)/1000
plot(FBMR(mass)~mass)
```



Conversion of energy to fat

```
FFat2Energy<-function(fat)34.3*fat # g fat to KJoules
FEnergy2Fat<-function(energy)energy/34.3 # KJoules to g fat
```

Using the spreadsheet equation taking into account temperature and windspeed to calculate BMR

As I understand it from the spreadsheet this set of equation should calculate the metabolic rate in the same units ()

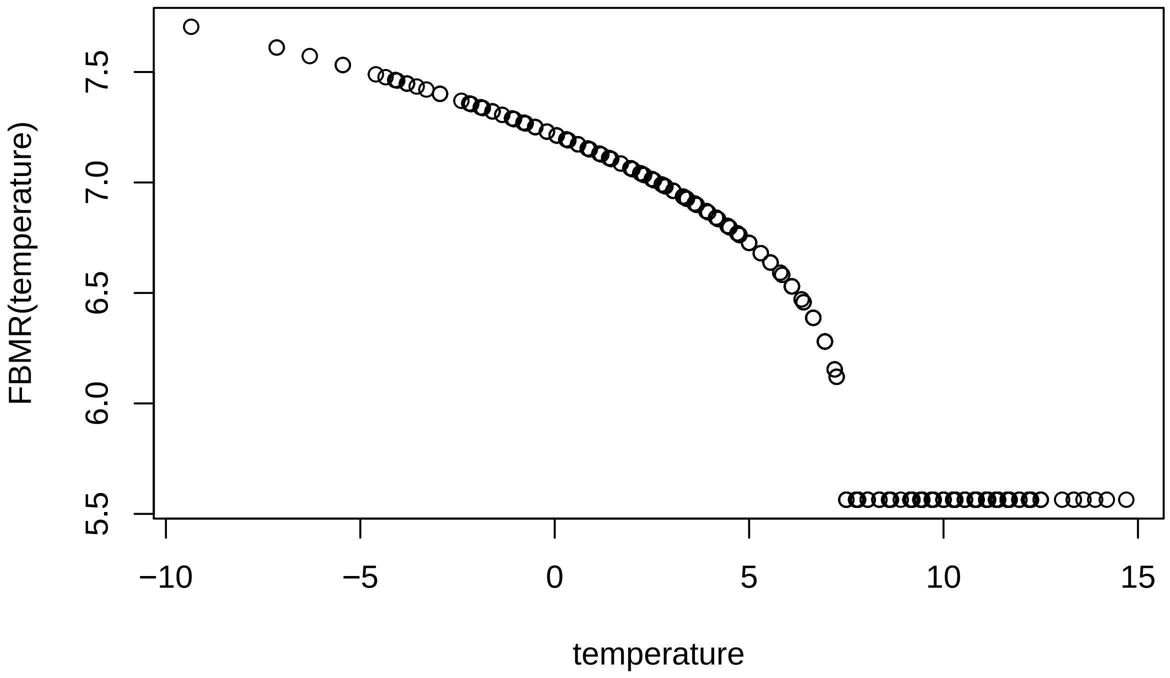
```
FBMR<-function(ftemperature=-10,fwindspeed=2,fmass=1500)
{
TBrant<-7.5
ftemperature[ftemperature>TBrant]<-TBrant
fwindspeed[fwindspeed<0.5]<-0.5
DeltaT<-TBrant-ftemperature
b<-0.0092*fmass^0.66*DeltaT^0.32
a<-4.15-b*sqrt(0.06)
a+b+sqrt(fwindspeed)
}
FBMR(10)
```

```
## [1] 5.564214
```

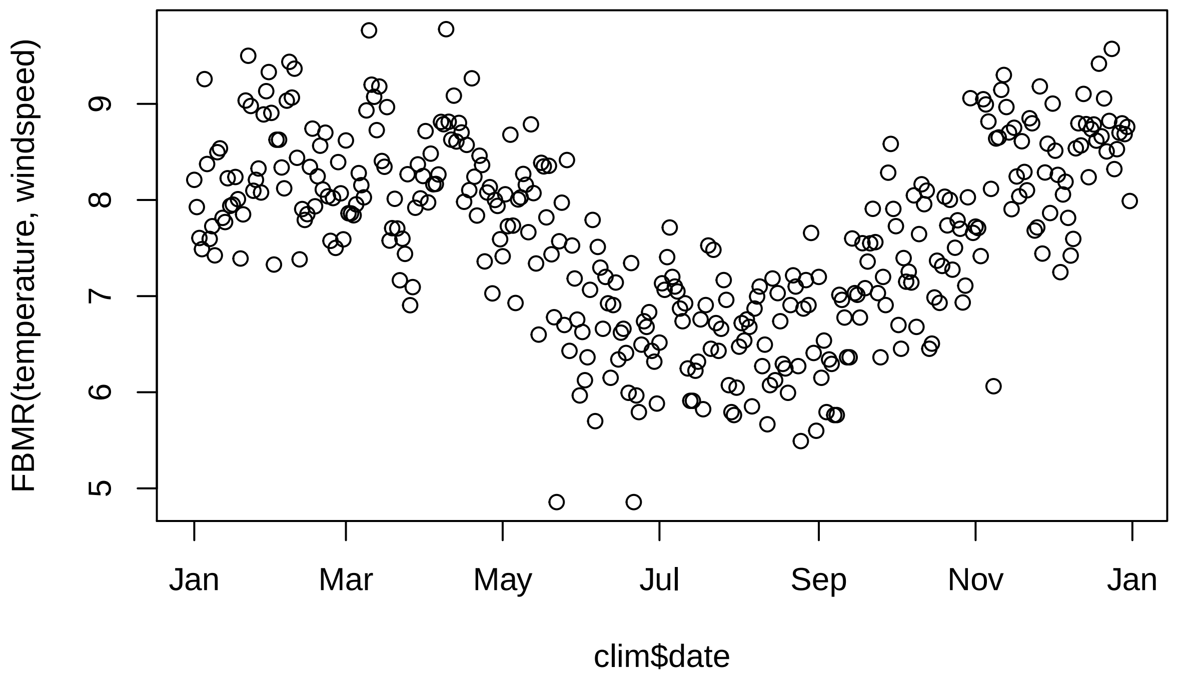
Testing against a year's climate data

I haven't got data for the whole of 2016.

```
clim<-subset(clim,as.numeric(format(clim$date,'%Y'))==2015)
temperature<-(clim$tmin+clim$tmax)/20
windspeed<-clim$avwind/10
plot(temperature,FBMR(temperature))
```

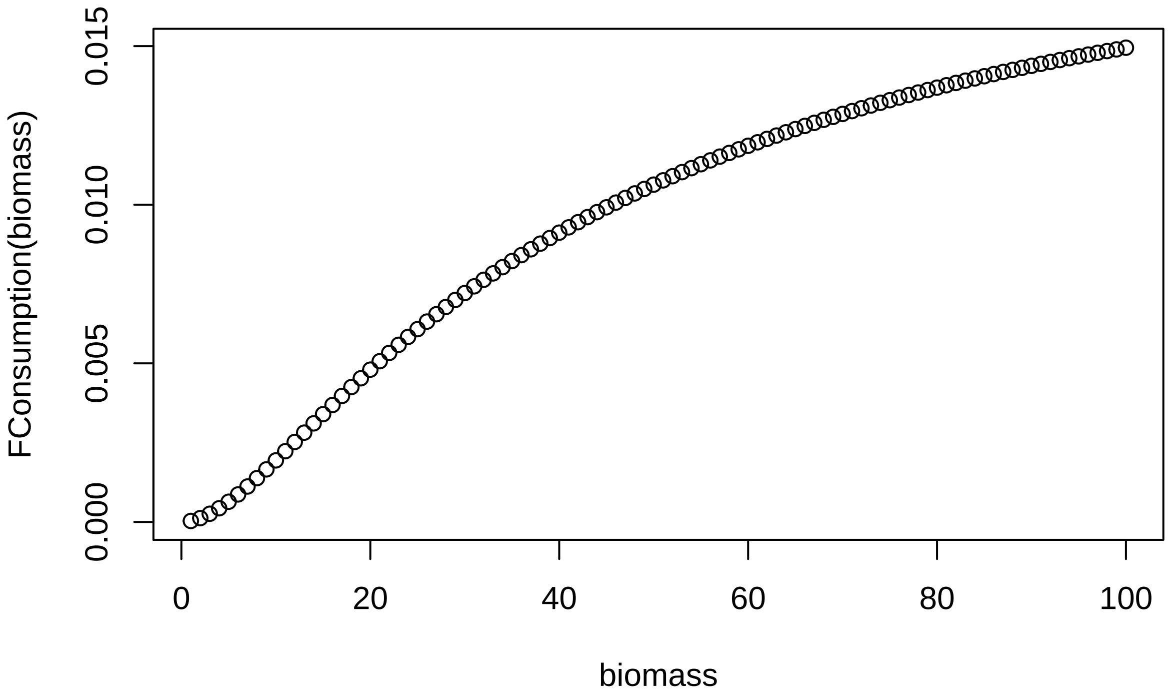


```
plot(clim$date,FBMR(temperature,windspeed))
```



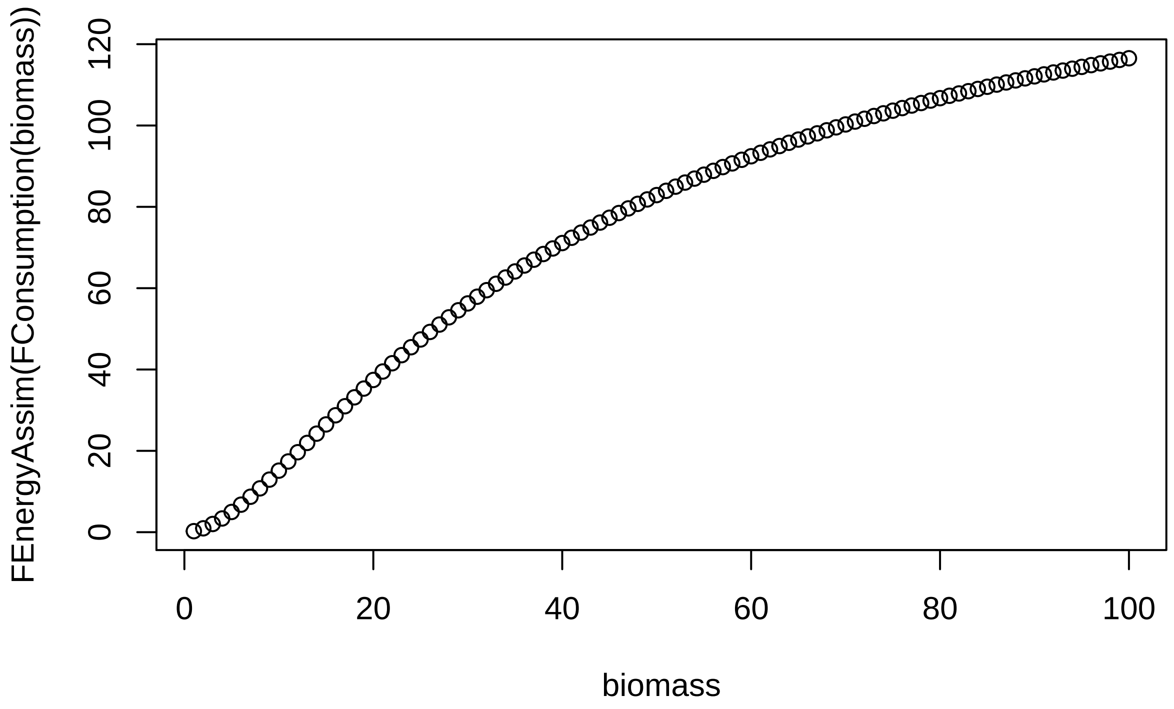
Consumption function from the Humbolt paper

```
biomass<-1:100
FConsumption<-function(fbiomass)100*0.01028*(1-exp(-0.105*fbiomass))*(1.0373*(1-exp(-0.0184*fbiomass))/
plot(FConsumption(biomass)~biomass)
```

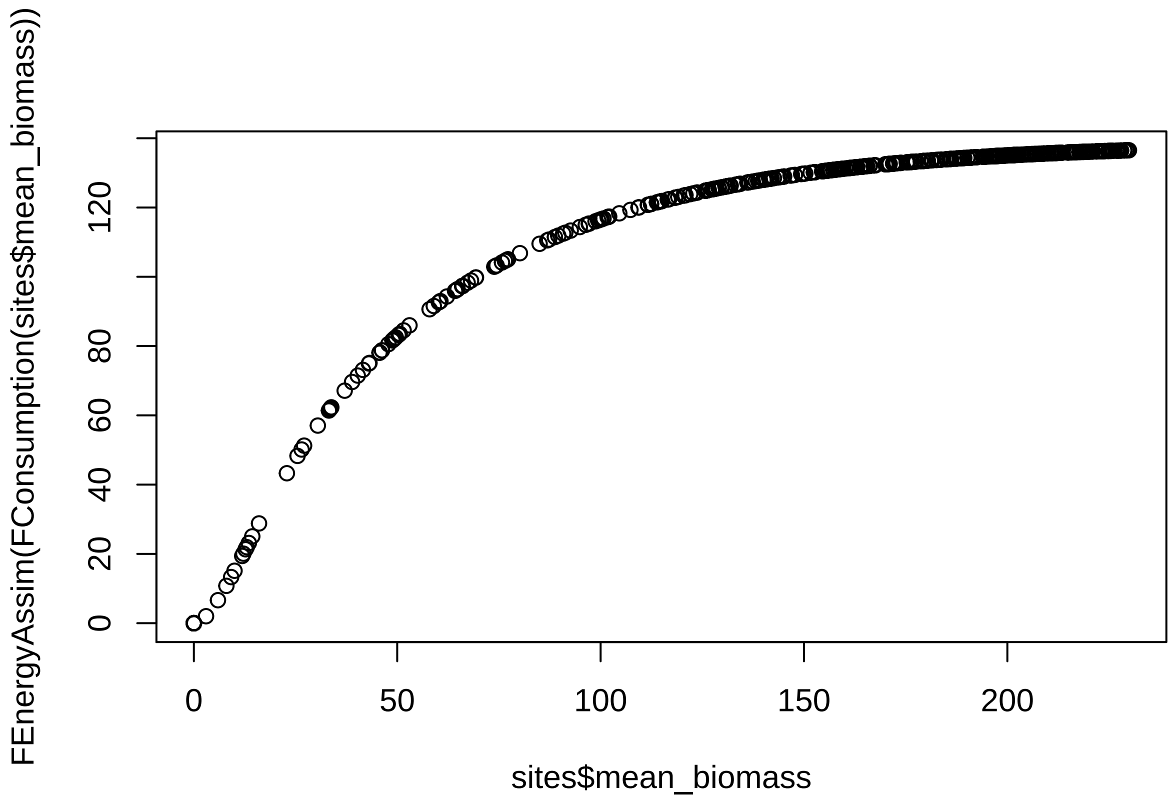


Converting into energy assimilated

```r
FEnergyAssim<-function(consumption, a=0.464, E= 16.8)consumption*a*E*1000
plot(FEnergyAssim(FConsumption(biomass))~biomass)
```



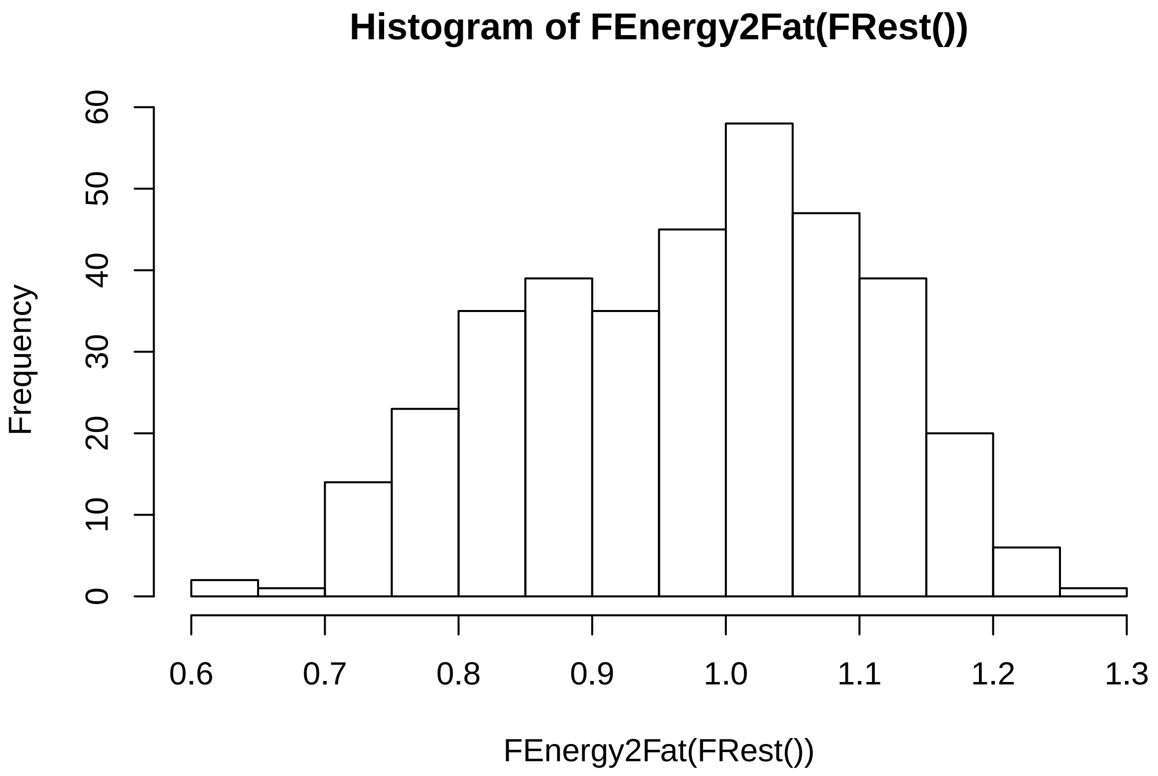
```r
plot(FEnergyAssim(FConsumption(sites$mean_biomass))~sites$mean_biomass)
```



Resting

Assume 20% over BMR. Add time argument in seconds. Returns energy used in KJoules.

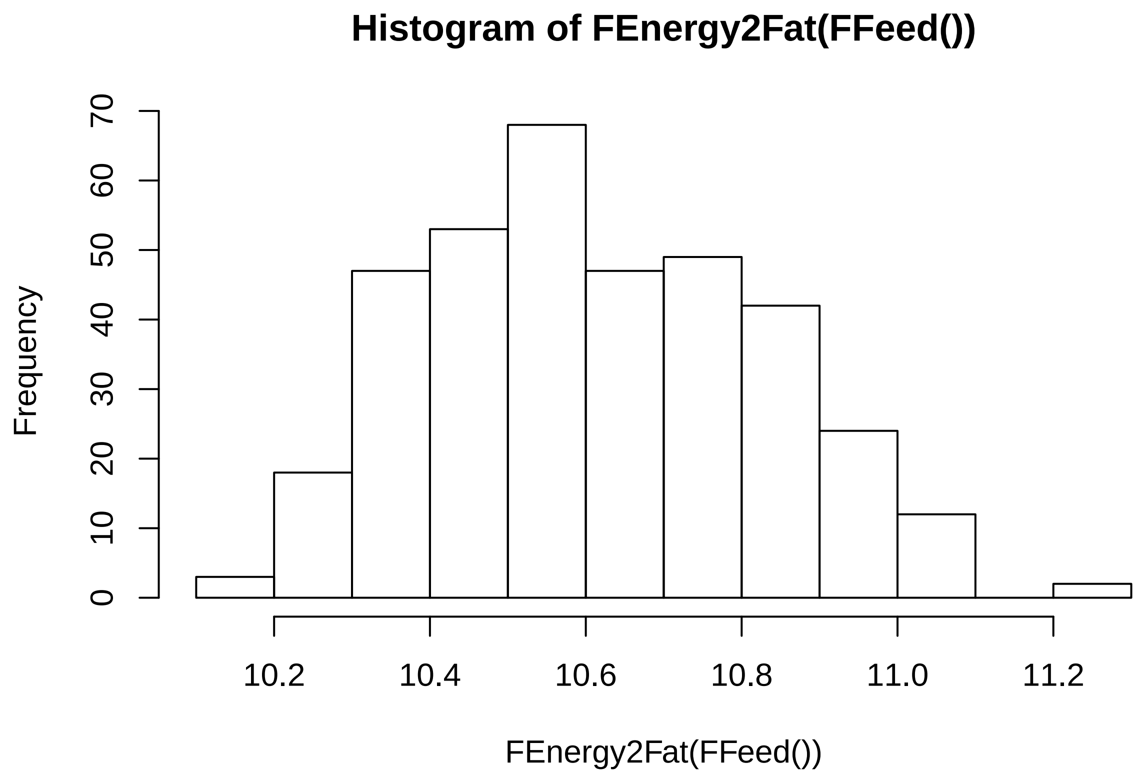
```
BMR<-FBMR(temperature,windspeed,fmass=1700)

FRest<-function(fbmr=BMR,ftime=3600)
{1.2*fbmr*ftime/1000}

hist(FEnergy2Fat(FRest()))
```



Feeding

Assume use twice BMR while feeding.

```
FFeed<-function(fbmr=BMR,ftime=3600,fbiomass=100){
EGain<-FEnergyAssim(FConsumption(fbiomass))*ftime/1000
EUse<-2*fbmr*ftime/1000
EGain-EUse
}
hist(FEnergy2Fat(FFeed()))
```



Maximum energy per day

According to the Humbolt paper this is given by

```
FMaxDaily<-function(mass)1713*mass^0.72
FMaxDaily(1500)/1000
```

```
## [1] 331.5454
```

```
FFeed()[1]
```

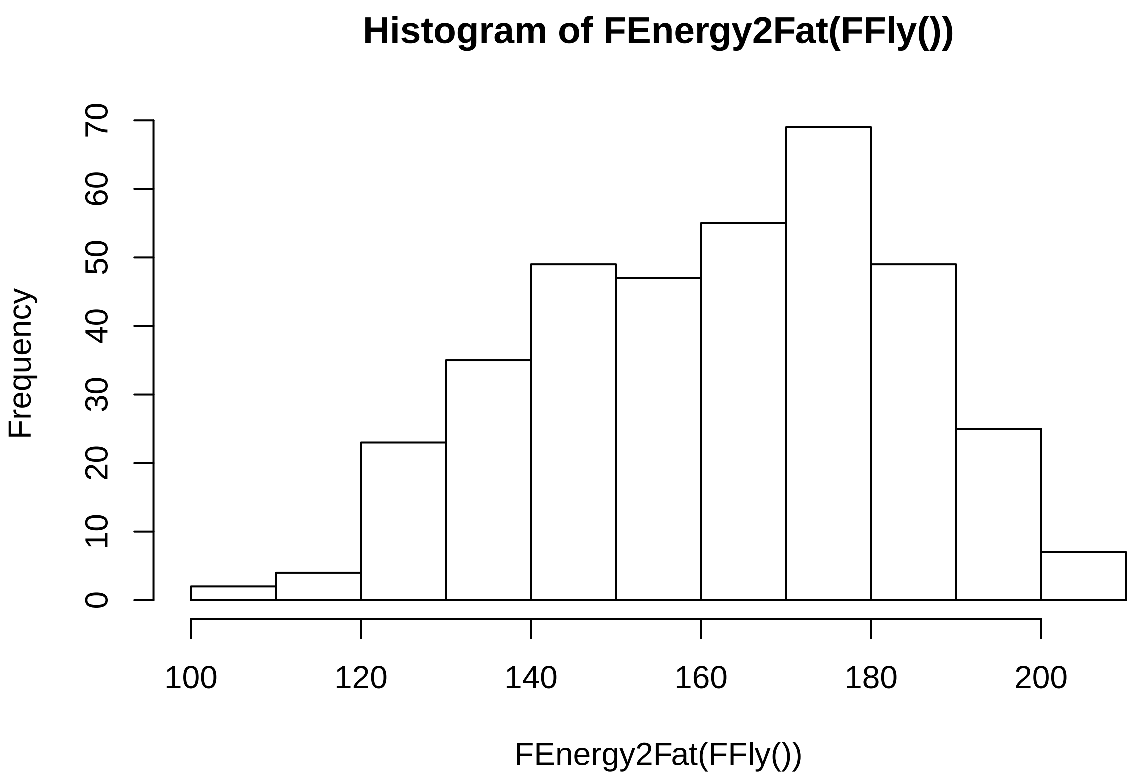
```
## [1] 359.3228
```

So using these formulae the birds can get enough energy for a whole day from just one hour's intensive feeding. This seems much too high. However this assumes that an hour feeding is completely dedicated to intensive feeding activity, which is unrealistic.

Flying

Assume velocity of 60km per second and overhead for take off for all flights equivalent to a minute's flight time. Use 12 times BMR.

```
FFlightTime<-function(fspeed=60,fdistance=1000000)
{
  speedms<-fspeed/3.6
  ftime<-fdistance/speedms+60
}
FFly<-function(fbmr=BMR,ftime=FFlightTime()){
  EUse<-12*fbmr*ftime/1000
  EUse
  }

hist(FEnergy2Fat(FFly()))
```



Setting the percentage area of each site which are suitable for feeding

This function is the first in the set to receive two dataframes as arguments. The sites data frame is merged with the tides data frame after filtering the tides data frame to the rows representing the height of the tides in each bay. The proportion of the area in each patch lying between a minimum depth and a maximum height after changing the water level at each site accordingly is calculated, added to the sites data frame and the whole data frame is returned. So we need to assign the result to the sites data frame.

```
FSuitable<-function(fsites=sites,ftm=tm,ftides=tides,depth=-0.4,height=0){
  current_tide<-subset(ftides,ftides$time==ftm)
  d<-merge(fsites,current_tide)
  tide<-d$ht
  depth<-depth+tide
  height<-height+tide
  dd<-cbind(d$min,d$q10,d$q25,d$median,d$q75,d$q90,d$max,depth,height)
  f<-function(x)
  {
    q<-c(0,10,25,50,75,90,100)
    qs<-x[1:7]
    depth<-x[8]
    height<-x[9]
    x2<-q[qs>=depth&qs<=height]
    # The supress warnings are needed as the vector may be of zero
    # length. This also leads to results of -inf instead of zero
    suppressWarnings(x2<-max(x2,na.rm=TRUE)-min(x2,na.rm=TRUE))
    if(is.na(x2))x2<-0
    if(x2==-Inf)x2<-0
    x2
  }
  fsites$psuitable<-apply(dd,1,f)
  fsites
}
```

```
tm<-FMakeTime(2015,7,15)
dt<-FMakeDate(tm)
sites<-FSuitable(sites,tm)
str(sites)
```

```
## 'data.frame':    553 obs. of  21 variables:
##  $ x                 : num  -18134280 -18118072 -18114830 -18118072 -18129417 ...
##  $ y                 : num  7424290 7435016 7435016 7422758 7424290 ...
##  $ geom              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ area_m2           : num  805904 803687 803687 806221 805904 ...
##  $ lon               : num  -163 -163 -163 -163 -163 ...
##  $ lat               : num  55.3 55.4 55.4 55.3 55.3 ...
##  $ min               : num  -0.652 -7.435 -5.449 -0.75 -1.5 ...
##  $ q10               : num  0.825 -4.164 -2.604 -0.75 -1.5 ...
##  $ q25               : num  1.22 -1.876 -0.671 -0.585 -0.513 ...
##  $ median            : num  1.22 -0.715 -0.293 -0.105 0.553 ...
##  $ mean              : num  3.125 -1.445 -0.681 0.012 0.109 ...
##  $ q75               : num  5.7275 -0.4179 -0.0661 0.5023 0.8234 ...
##  $ q90               : num  7.22 -0.2411 0.0759 0.8652 0.8924 ...
##  $ max               : num  9.22 0.084 0.9 4.763 0.91 ...
##  $ rid               : int  251 106 120 285 264 497 452 496 455 209 ...
##  $ psuitable         : num  0 15 0 0 0 0 25 0 25 10 ...
##  $ median_biomass    : num  134 227 214 215 126 ...
##  $ mean_biomass      : num  131 213 201 183 131 ...
##  $ median_shootlength: num  18 63.5 56 52 35 96 61 49 55 68 ...
##  $ mean_shootlength  : num  13.2 65.6 63.6 49.1 46 ...
##  $ station           : int  1 1 1 1 1 4 1 4 1 1 ...
```

Add the birds

The dataframes obtained from the data base include site data, distances between sites and climate data. However there are no birds. This function forms an initial birds data frame. Other derived characteristics can be added when the model runs.

```
FArriveBirds<-function(ftm=tm,nbirds= 10,fsites=sites,male_wt=1500,female_wt=1400,fat=300)
  {
  sex<-sample(c("M","F"),nbirds,replace=T)
  lean_wt<-numeric(nbirds)
  lean_wt[sex=="M"]<-male_wt
  lean_wt[sex=="F"]<-female_wt
  lean_wt<-lean_wt+rnorm(nbirds,0,sd=20)
  fat<-rlnorm(nbirds,mean=log(fat),sd=0.2)
  wt<-lean_wt+fat
  energy_store<-FFat2Energy(fat)
  ## Add other properties here
  ##
  rid<-sample(fsites$rid,nbirds,replace=TRUE) ## Place them at random
  bid<-1:nbirds ## ID number
  birds<-data.frame(bid,arrive_time=ftm,sex=sex,weight=wt,fat=fat,energy_store=energy_store,rid=rid)
  birds
}
```

Add more birds

If we want to add birds after the first arrival time then we need to pass the birds dataframe that was created to the function in order to keep the ids sequential. The same function is used again to create more birds at the site.

```
FAddBirds<-function(ftm=tm,nbirds= 10,fsites=sites,fbirds=birds)
  {
  newbirds<-FArriveBirds(ftm,nbirds,fsites)
  newbirds$bid<-newbirds$bid+max(fbirds$bid)
  rbind(fbirds,newbirds)
}
```

```
birds<-FArriveBirds(tm,10,sites)
birds<-FAddBirds(tm,10000,sites,birds)
str(birds)
```

```
## 'data.frame':    10010 obs. of  7 variables:
##  $ bid         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ arrive_time : POSIXct, format: "2015-07-15 01:00:00" "2015-07-15 01:00:00" ...
##  $ sex         : Factor w/ 2 levels "F","M": 1 1 2 1 2 1 1 1 2 2 ...
##  $ weight      : num  1636 1667 1872 1759 1773 ...
##  $ fat         : num  210 239 372 382 261 ...
##  $ energy_store: num  7212 8199 12749 13119 8940 ...
##  $ rid         : int  440 542 460 322 204 414 337 57 66 160 ...
```

Calculate value of moving

This is a tricky function to get right. The “dist”" data frame looks like this …

```
str(dist)
```

```
## 'data.frame':    305809 obs. of  3 variables:
##  $ rid   : int  251 251 251 251 251 251 251 251 251 251 ...
##  $ rid2  : int  251 106 120 285 264 497 452 496 455 209 ...
##  $ dist_m: num  0 11069 12651 9285 2773 ...
```

We have two rids, one representing the source site and the second representing the potential destinations. This contains all possible moves. To speed things up we only want to look at distances between sites with birds and only within a search distance.

```
FFilterMoves<-function(fdist=dist,fbirds=birds,search_distance=10000)
{
moves<-fdist[fdist$rid %in% unique(fbirds$rid),]
moves<-moves[moves$dist_m<search_distance,]
moves}
```

```
moves<-FFilterMoves()
head(moves)
```

```
##    rid rid2    dist_m
## 1  251  251    0.0000
## 4  251  285 9284.9133
## 5  251  264 2772.8898
## 10 251  209 8717.6461
## 11 251  276 6527.9176
## 17 251  253  871.9961
```

Scoring the moves

We need to look at eelgrass biomass and the area that is suitable on the destination sites (psuitable). If we merge the sites data frame by default the matched column is rid, not rid2. I.e. the source rather than the destination. We should only look at sites with a suitability of greater than zero.

We can calculate the energy gained feeding and lost by moving using a fixed basal metabolic rate for comparative purposes to simplify a bit. There's nothing to be gained by calculating the actual BMR on the day as all we really need at the end is a rank. The ranks are grouped by source rid using tapply.

```
FScoreMoves<-function(fmoves=moves,fsites=sites,flight_cost=TRUE)
{
fsites<-fsites[fsites$psuitable>0,]
fmoves<-merge(fmoves,fsites, by.x="rid2",by.y="rid")
fmoves$flight_time<-FFlightTime(fdistance=fmoves$dist_m)
fmoves$fly<-FFly(fbmr=9,ftime=fmoves$flight_time)
biomass<-fmoves$mean_biomass#*(fmoves$psuitable/100)

if (flight_cost==TRUE)
  {fmoves$feed<-FFeed(fbmr=9,ftime=3600-fmoves$flight_time,fbiomass=biomass) ## Subtract the time neeed
fmoves$energy<-fmoves$feed -fmoves$fly}

if (flight_cost==FALSE)
  {fmoves$feed<-FFeed(fbmr=9,ftime=3600,fbiomass=biomass) ## Subtract the time neeed to fly there
fmoves$energy<-fmoves$feed}


fmoves<-fmoves %>%
          arrange(rid, -energy) %>%
          group_by(rid) %>%
          mutate(rank=row_number()) %>%
          filter(rank==1)

fmoves<-fmoves[c("rid","rid2","dist_m","flight_time","feed","fly","energy","rank")]
}

moves<-FScoreMoves()
```

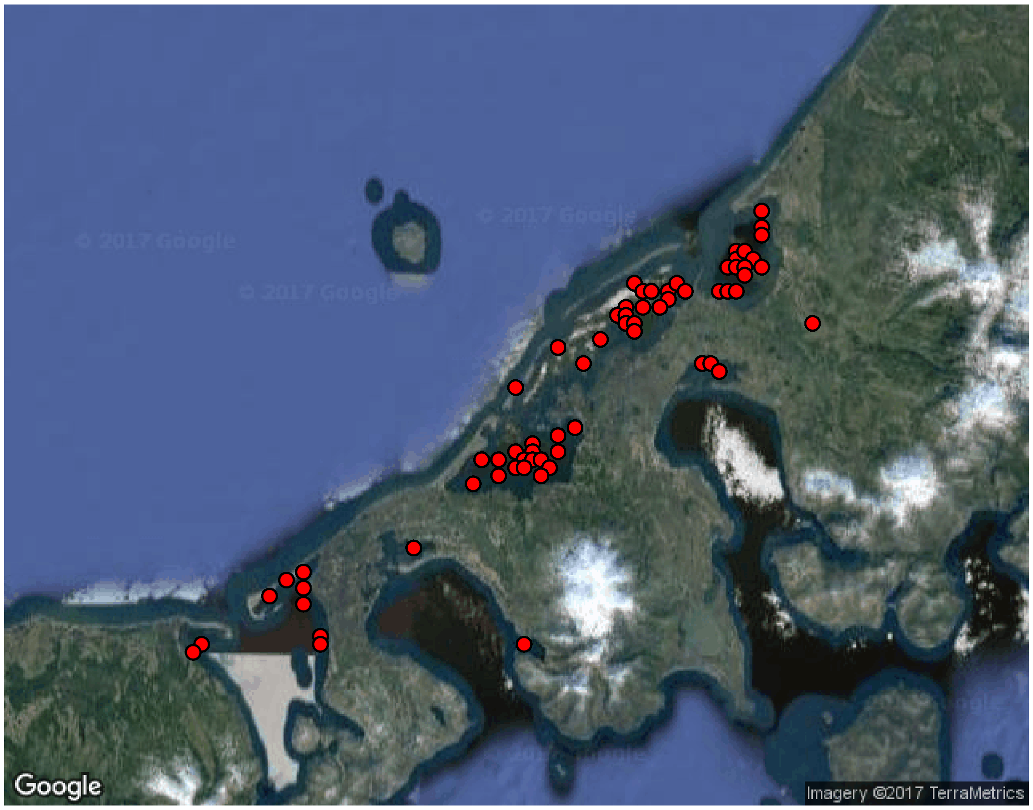
Move the birds

There will be some cases in which there are no suitable sites in range as they are all covered by the tide. In this case the birds stay put. Change the rid to the new value and add only the flight time and distance to the data frame.

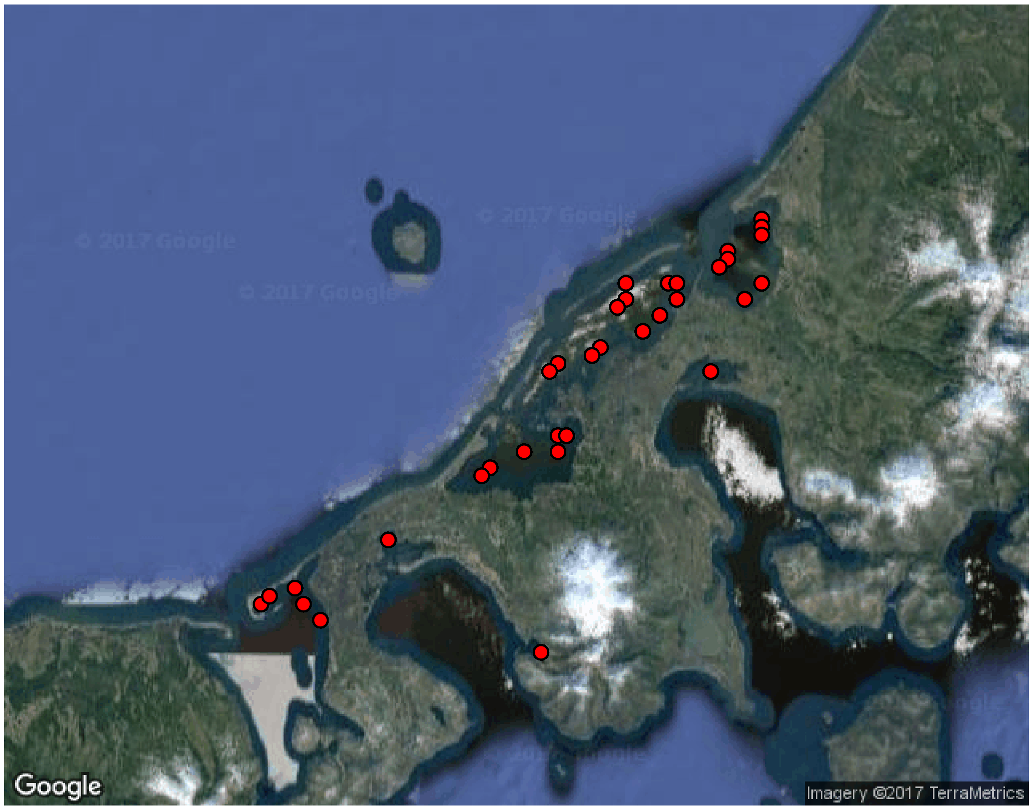
```
FMoveBirds<-function(fbirds=birds,fmoves=moves)
{
  bird_moves<-merge(fbirds,fmoves,all.x=TRUE)
  bird_moves$rid2[is.na(bird_moves$rid2)]<-bird_moves$rid[is.na(bird_moves$rid2)]
  fbirds$rid<-bird_moves$rid2
  fbirds$distance<-bird_moves$dist_m
  fbirds
}
```

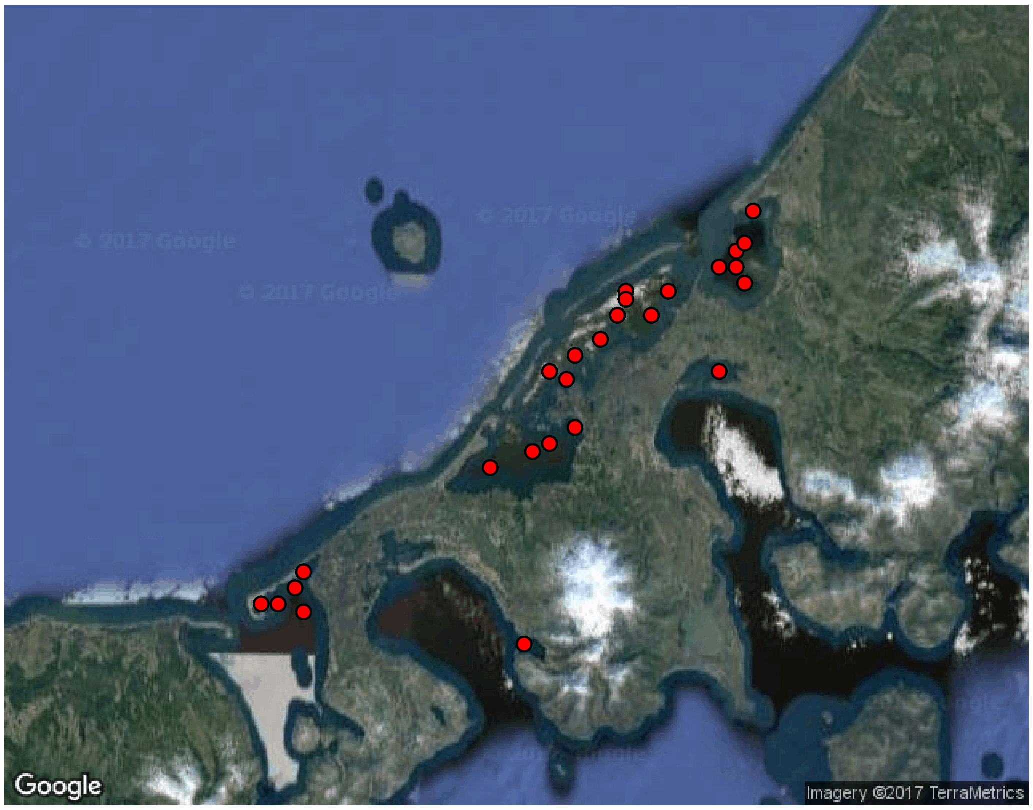
```
move<-FFilterMoves(search_distance=20000)
moves<-FScoreMoves()
birds<-FMoveBirds()

plot(map)
```

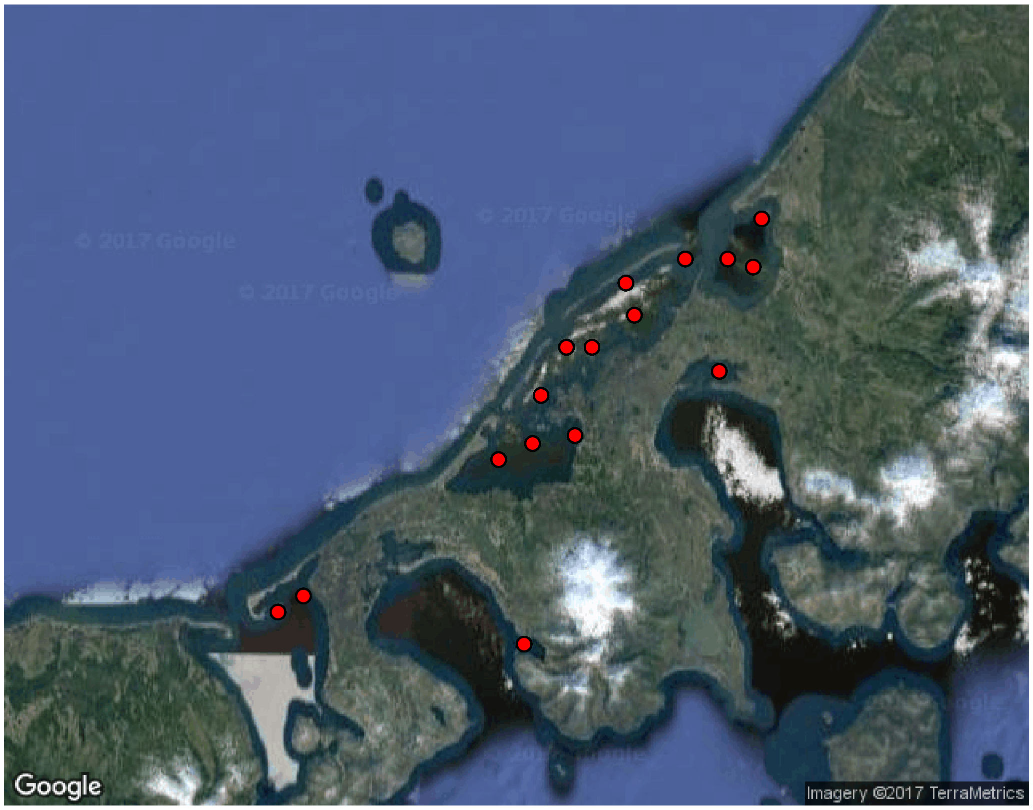
```
birds_sites<-merge(birds,sites)
points(birds_sites$x,birds_sites$y,pch=21,bg="red")
```

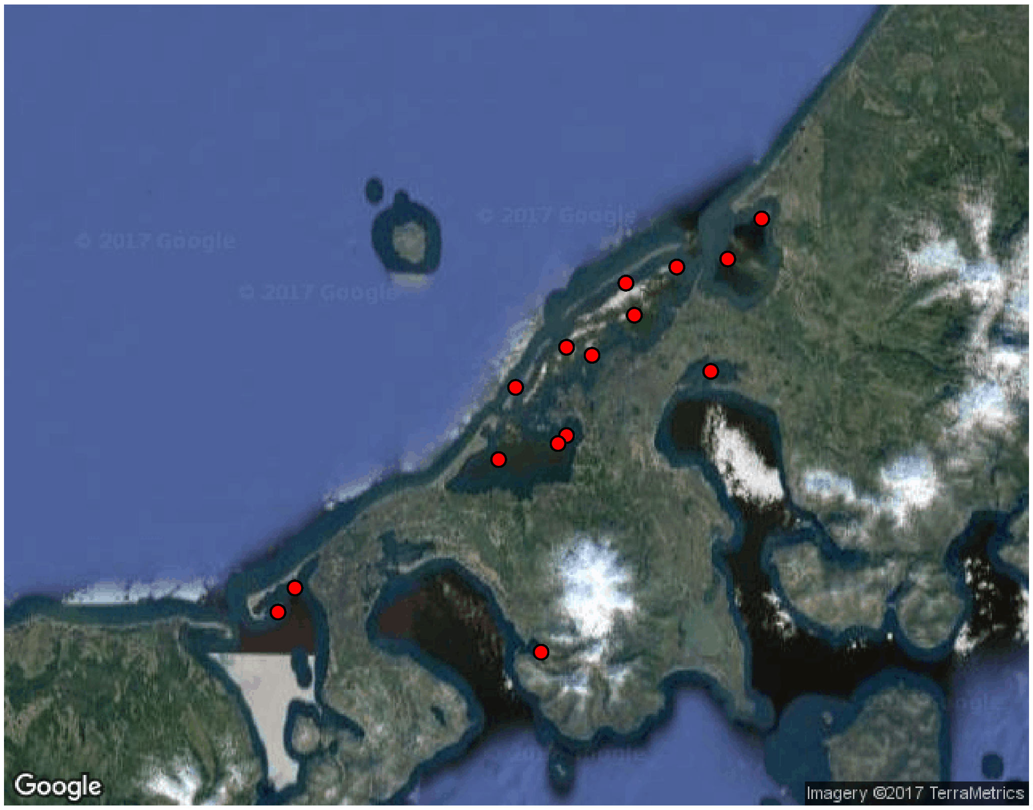


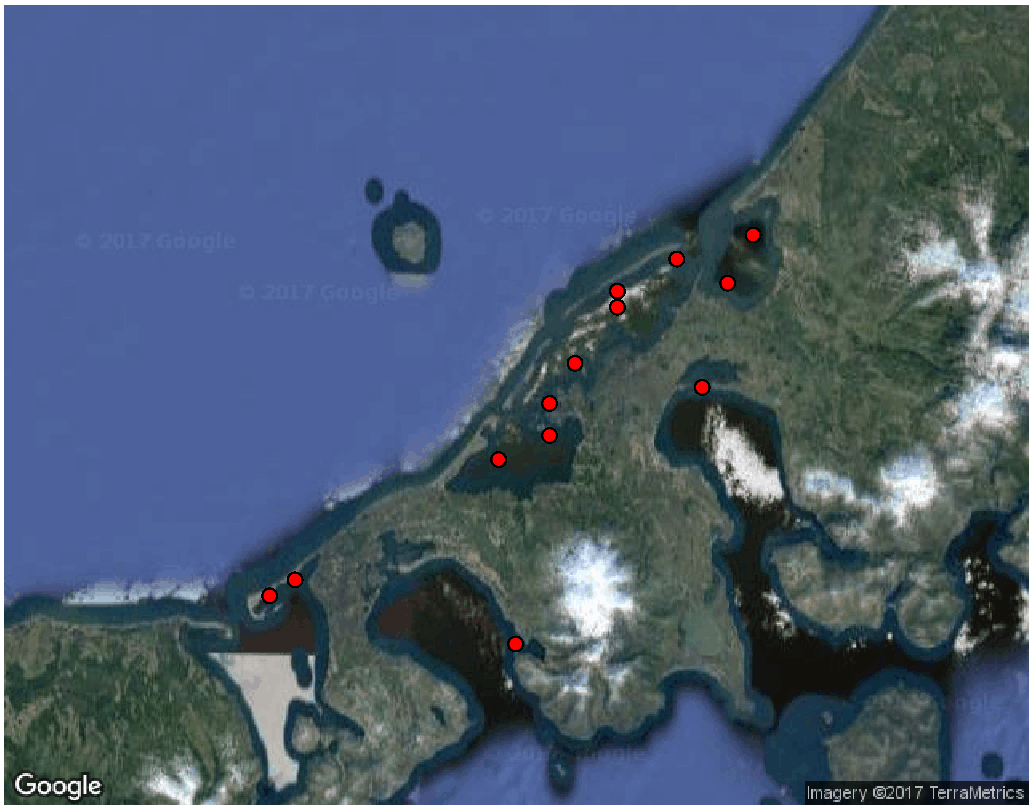
```
for (i in 1:10)
{
tm<<-tm+(60*60*60)
sites<<-FSuitable(sites,tm)
moves<<-FFilterMoves(fbirds=birds)
moves<<-FScoreMoves()
birds<<-FMoveBirds(birds,moves)
plot(map)
birds_sites<-merge(birds,sites)
points(birds_sites$x,birds_sites$y,pch=21,bg="red")
}
```

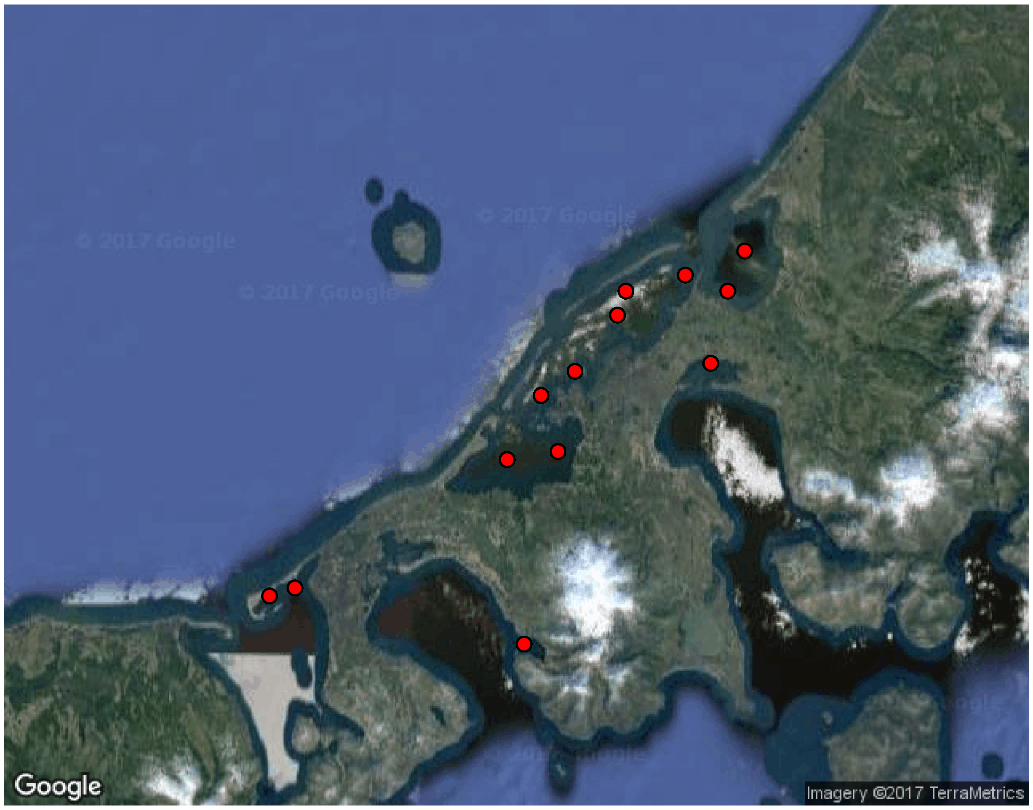

© 2017 Google
© 2017 Google
© 2017 Google
Google
Imagery ©2017 TerraMetrics


© 2017 Google
© 2017 Google
© 2017 Google
Google
Imagery ©2017 TerraMetrics

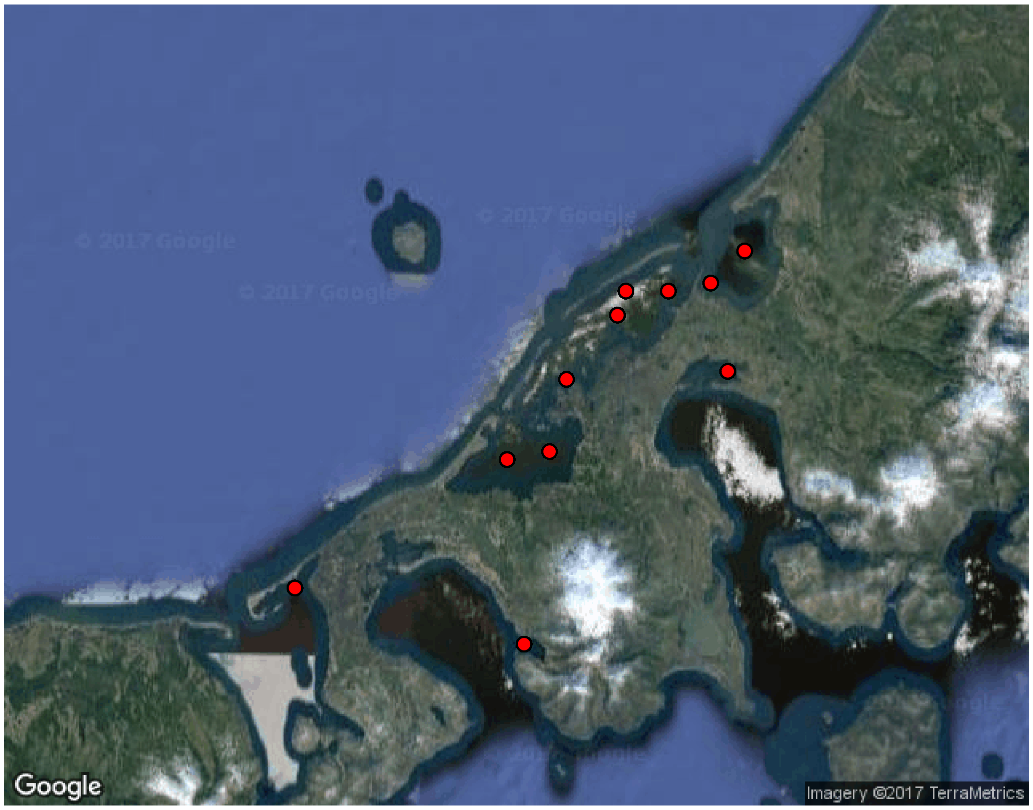

Google
Imagery ©2017 TerraMetrics

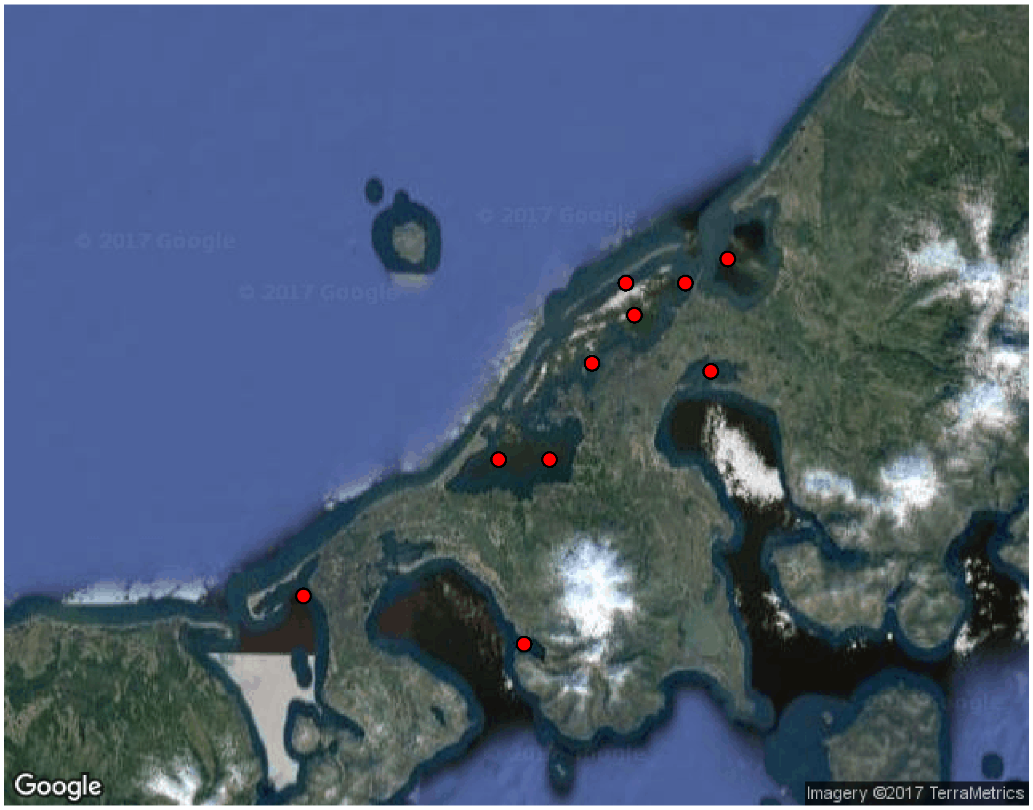

© 2017 Google
© 2017 Google
© 2017 Google
Google
Imagery ©2017 TerraMetrics

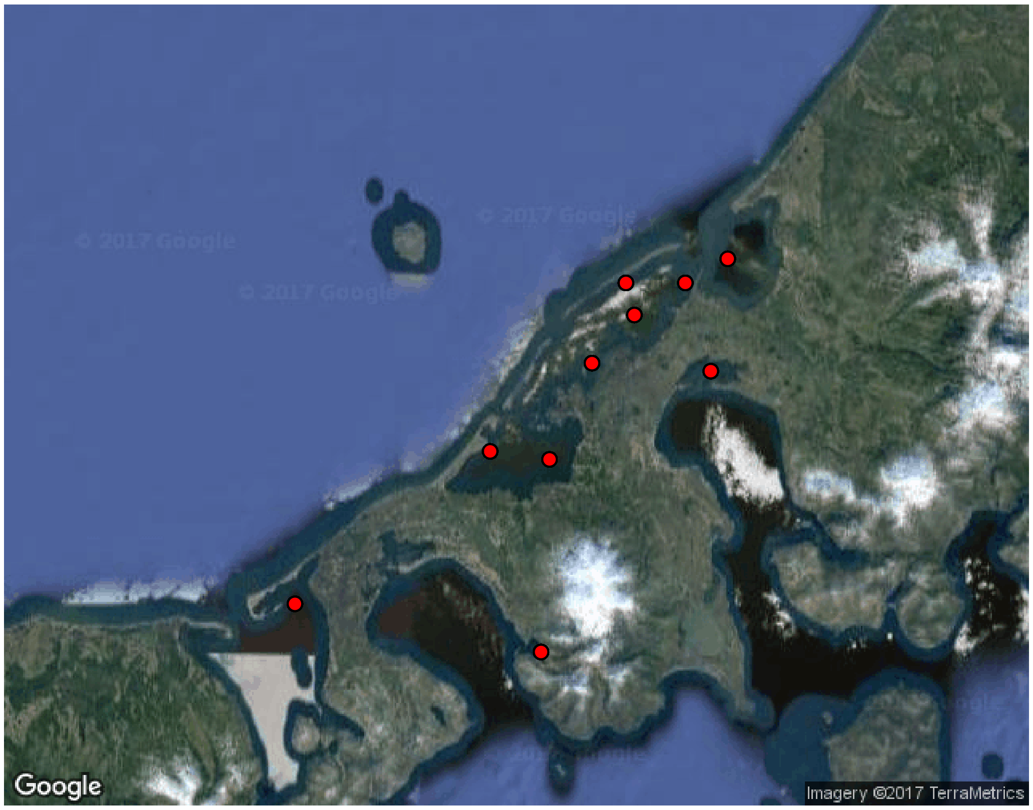

Google
Imagery ©2017 TerraMetrics


© 2017 Google
© 2017 Google
© 2017 Google
Google
Imagery ©2017 TerraMetrics

Google
Imagery ©2017 TerraMetrics


© 2017 Google
© 2017 Google
© 2017 Google
Google
Imagery ©2017 TerraMetrics


© 2017 Google
Google
Imagery ©2017 TerraMetrics


© 2017 Google
Google
Imagery ©2017 TerraMetrics

```r
save(list=ls(),file="/home/rstudio/morph/scripts/functions.rob")
```