Simulating tracks with the availability package

Ben Raymond, Simon Wotherspoon, Ryan Reisinger

2016-01-21

Background

This package provides methods for estimating the geographic space available to an animal, based on an observed track collected by a GPS tag or similar. Broadly, these functions generate a simulated track that preserves certain properties of the original track. The simulated tracks can optionally be constrained by a land mask (or other mask that prevents the simulated track from visiting unwanted locations). It can also be forced to pass through fixed points at certain times, allowing (for example) simulated tracks to finish at the same point from which they started.

The simulated tracks only attempt to preserve basic properties of the observed track, such as the speeds exhibited by the animal and the overall track duration. The intention is not to explicitly reproduce behavioural states such as area-restricted searching. Rather, the goal is to estimate a plausible track that the animal might have produced, if it did not have any environmental or other preferences governing its movements. The geographic and environmental properties of the actual track can then be compared to those of simulated tracks, giving insights into the preferred habitats of the animal.

Usage examples

Load the availability package, and some others that we'll use:

```
library(availability)
library(geosphere)
library(ggplot2)
set.seed(30)
```

Some helper functions used in this demo:

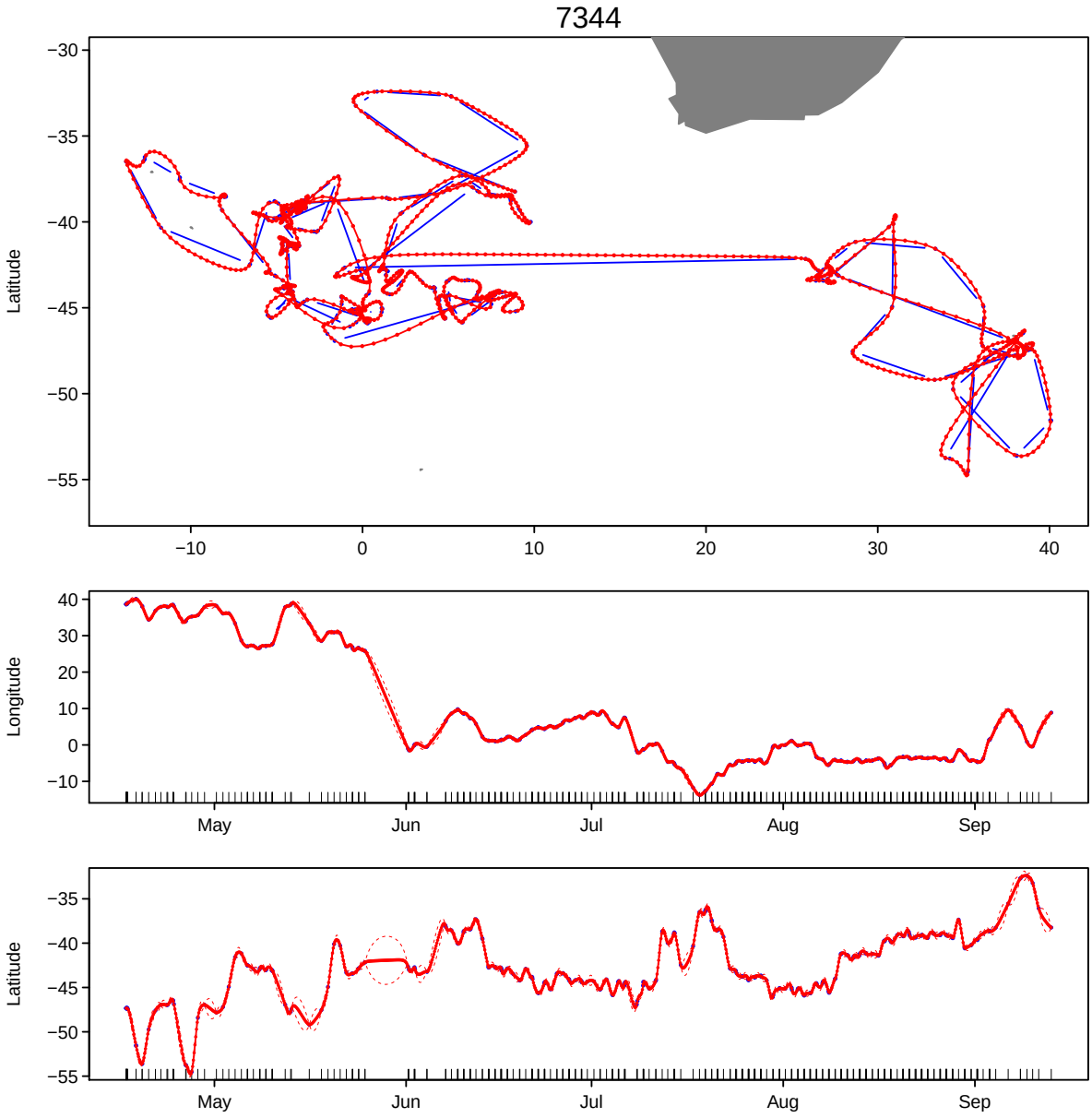
```
## helper function for crawl filtering, not part of this package
source("CRAWL_Filter.R")

## this function converts the fit object returned by crawl into the "template track"
## format used by the availability package routines
templateTrack <- function(fit) {
  pr <- fit$pred
  pr <- pr[pr$locType=="p",]
  list(ts=pr$date,xs=as.matrix(pr[,c("mu.x","mu.y","nu.x","nu.y")]))
}

## simple helper to plot tracks
plotTracks <- function(...) {
  require(maptools)
  data(wrld_simpl)
  pal <- c("#1B9E77", "#D95F02", "#7570B3", "#E7298A", "#66A61E", "#E6AB02" )
  trks <- list(...)
  ## if tracks are 4-column data.frames (e.g. from crawl, keep cols 1 and 3)
```

```
  ##trks <- lapply(trks,function(trk){trk$xs=trk$xs[,c(1,min(3,ncol(trk$xs)))]; trk})
  xlim <- do.call(range,lapply(trks,function(trk) trk$xs[,1]))
  ylim <- do.call(range,lapply(trks,function(trk) trk$xs[,2]))
  plot(xlim,ylim,type="n",xlim=xlim,ylim=ylim,xlab="longitude",ylab="latitude")
  for(k in floor((min(xlim)+180)/360):floor((max(xlim)+180)/360))
    plot(elide(wrld_simpl,shift=c(k*360,0)),col="grey80",border="grey80",add=TRUE,xlim=xlim,ylim=ylim)
  for(k in seq_along(trks)) lines(trks[[k]]$xs,col=pal[k])
}
```

Sooty Albatross

Sooty albatross data from Ryan R. Read the data file and pass it through crawl to produce a filtered track with evenly-sampled 3-hourly time steps:

```
d <- read.csv("MAR2015_DMS_PTT_7344_7344.csv",header=T)
time_step <- 3
fits <- crawlFilter(d,polar.coord=FALSE,tstep=time_step)
```



Extract the filtered track and show the first few rows:

```
tp <- templateTrack(fits[[1]])
head(tp$xs)
```

```
##        mu.x      mu.y       nu.x         nu.y
## 1  38.64381 -47.34339 0.01167768  0.005384574
## 8  38.73443 -47.32570 0.05686997 -0.014112818
## 13 38.97827 -47.41823 0.05482254 -0.020934651
## 15 39.13442 -47.55135 0.04966302 -0.065388495
## 16 39.27745 -47.79732 0.04585003 -0.096738789
## 17 39.41025 -48.12235 0.04276988 -0.118620438
```

The columns are in the order longitude mu, latitude mu, longitude nu, and latitude nu.

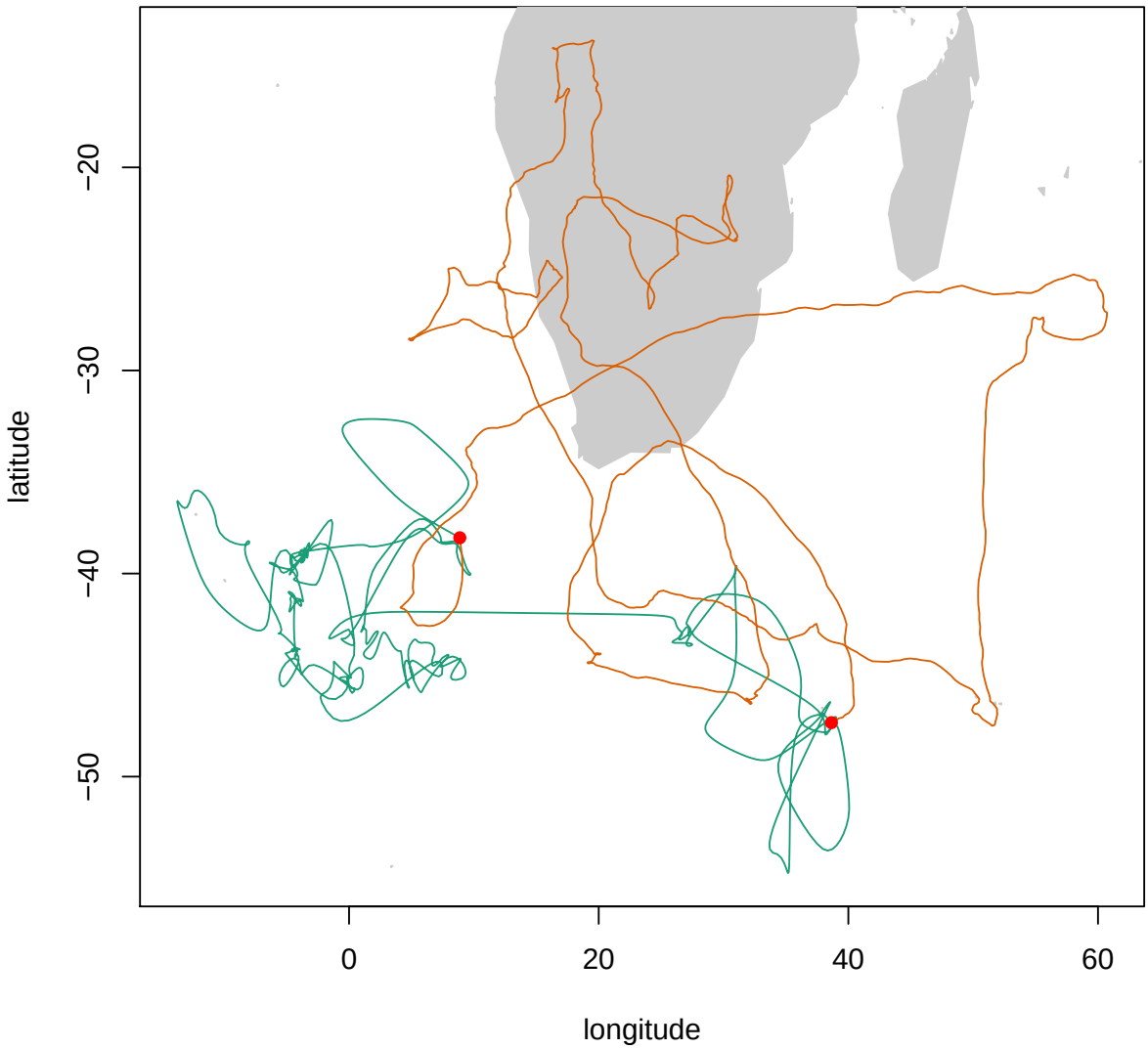
Simulating a track by the VAR method

Fit a vector-autoregressive movement model to this filtered track. This model assumes that the x- and y-speeds at time *t* are a linear function of the speeds at time *t-1*, plus random noise.

```
arfit <- surrogateARModel(tp$xs)
```

Use the fitted model to generate new tracks. By default, the end points of the simulated track are fixed to the same start and end points as the original track, and no land mask is applied:

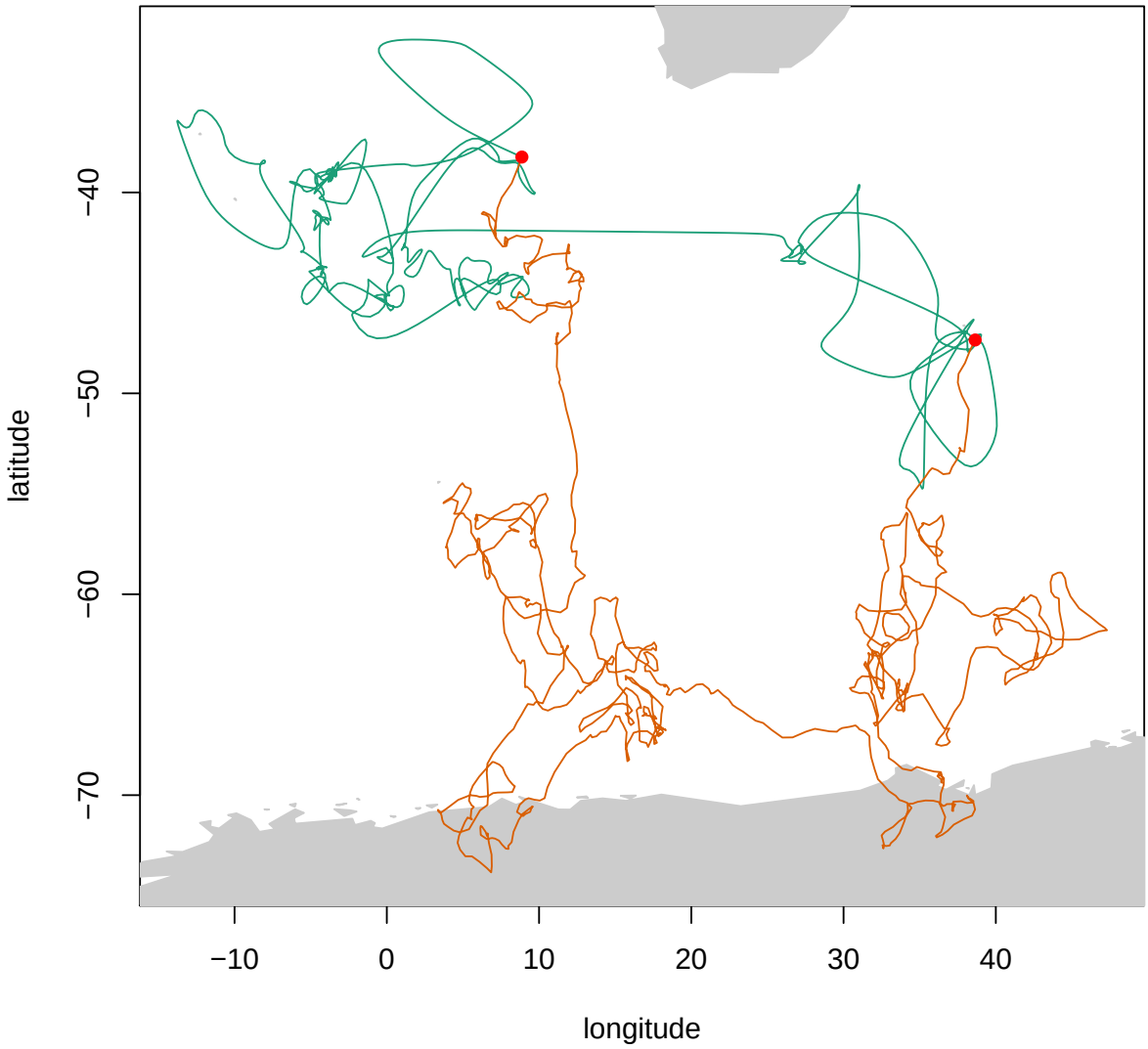
```
trkar <- surrogateAR(arfit,tp$xs,tp$ts)
plotTracks(tp,trkar)
points(tp$xs[c(1,nrow(tp$xs)),1:2],pch=16,col="red")
```



Simulating a track by the crawl method

Alternatively, we can use the movement model fitted by crawl to directly simulate a new track with the same movement properties:

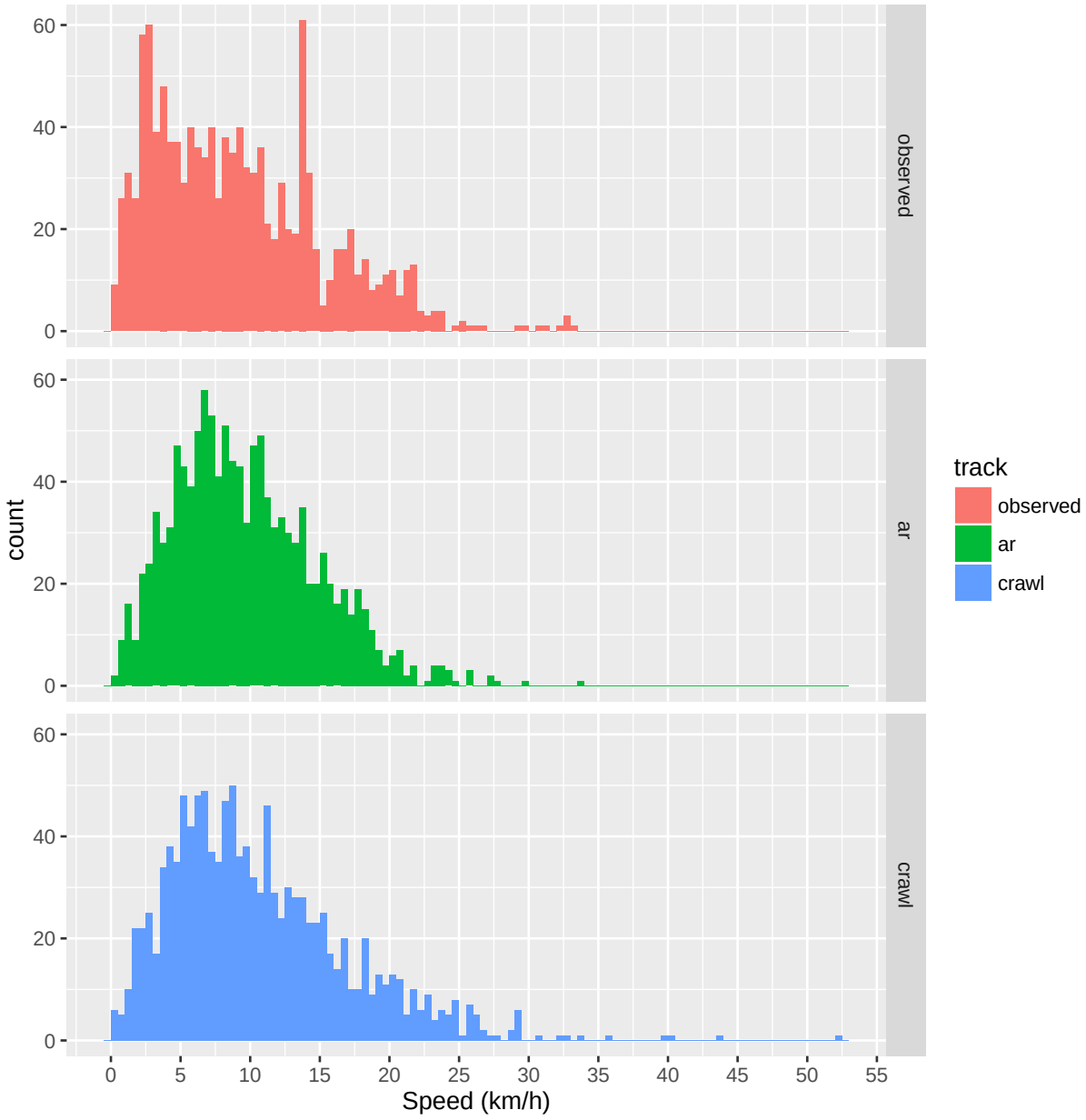
```
model <- surrogateCrawlModel(fits[[1]]$crawl,180/60)
trkcrw <- surrogateCrawl(model,tp$xs,tp$ts)
plotTracks(tp,trkcrw)
points(tp$xs[c(1,nrow(tp$xs)),1:2],pch=16,col="red")
```



Comparison of simulated tracks to the observed track

The speed distributions of the three tracks can be compared directly:

```
temp=data.frame(speed=distVincentyEllipsoid(tp$xs[-nrow(tp$xs),1:2],tp$xs[-1,1:2])/1e3/time_step,track=
temp=rbind(temp,data.frame(speed=distVincentyEllipsoid(trkar$xs[-nrow(trkar$xs),],trkar$xs[-1,])/1e3/ti
temp=rbind(temp,data.frame(speed=distVincentyEllipsoid(trkcrw$xs[-nrow(trkcrw$xs),1:2],trkcrw$xs[-1,1:2
ggplot(data=temp,aes(x=speed,fill=track))+geom_histogram(binwidth=0.5)+facet_grid(track~.)+scale_x_conti
```

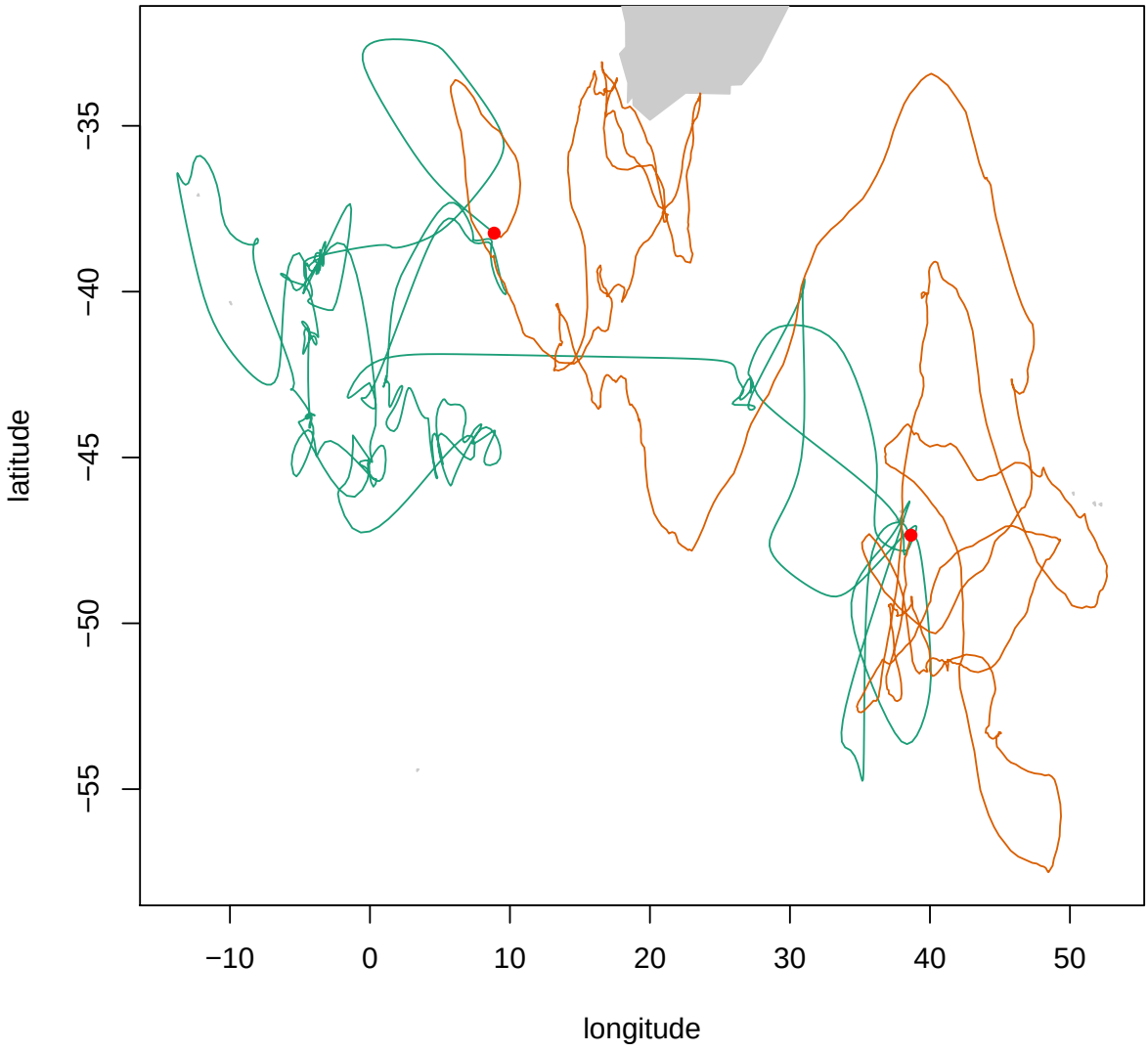


The observed track appears to have a number of segments with slow speeds (see the peak in the histogram at around 3 km/h, possibly related to area-restricted searching), superimposed on a broad distribution of speeds up to about 35 km/h. There is also a peak in the histogram at around 15 km/h, but this looks suspiciously like an artefact arising from the interpolation of the raw track positions.

Simulating tracks: land masking

By default, the end points of the simulated track are fixed to the same start and end points as the original track, and no land mask is applied. A land mask can be applied using the mask supplied with the package:

```
trkar <- surrogateAR(arfit,tp$xs,tp$ts,point.check=gshhsMask())
plotTracks(tp,trkar)
points(tp$xs[c(1,nrow(tp$xs)),1:2],pch=16,col="red")
```



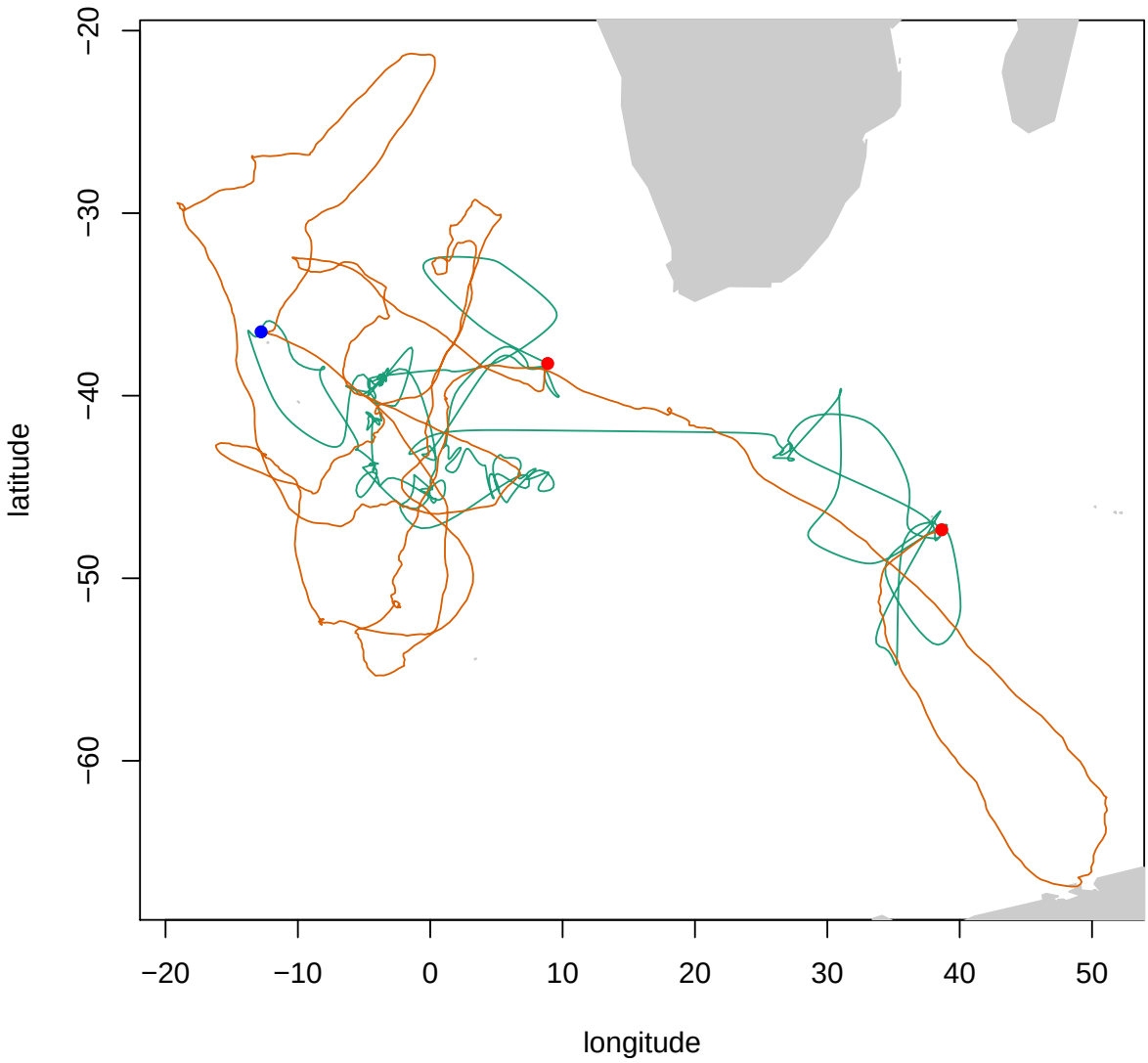
The land masking function can accept a time coordinate as well as a location, so it could be made time-varying (e.g. dynamically masking out areas covered by sea ice, for example).

Simulating tracks: fixing points

By default, the first and last points of the simulated track are fixed to match those of the template track. These fixed points can be removed, or other fixed points enforced, via the `fixed` parameter. (For the AR method, the first point must always be fixed.)

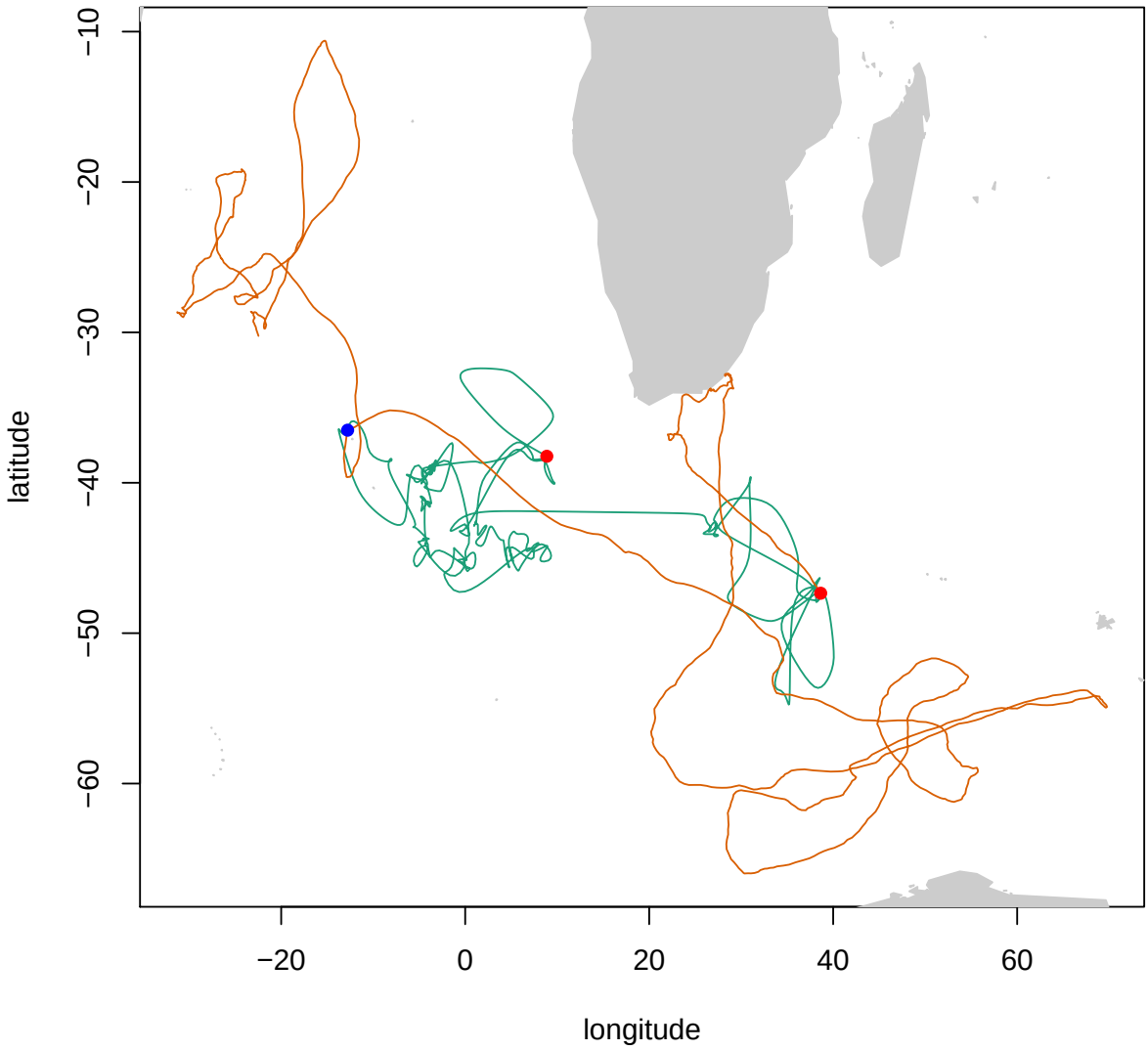
Fix an additional arbitrary point on the path (indicated by the blue point on the plot):

```
fixed <- logical(nrow(tp$xs))
fixed[1] <- fixed[nrow(tp$xs)] <- fixed[750] <- T
trkar <- surrogateAR(arfit,tp$xs,tp$ts,fixed,point.check=gshhsMask())
plotTracks(tp,trkar)
points(tp$xs[c(1,nrow(tp$xs)),1:2],pch=16,col="red")
points(tp$xs[750,1],tp$xs[750,2],pch=16,col="blue")
```



Repeat but do not fix final endpoint:

```
fixed <- logical(nrow(tp$xs))
fixed[1] <- fixed[750] <- T
trkar <- surrogateAR(arfit,tp$xs,tp$ts,fixed,point.check=gshhsMask())
plotTracks(tp,trkar)
points(tp$xs[c(1,nrow(tp$xs)),1:2],pch=16,col="red")
points(tp$xs[750,1],tp$xs[750,2],pch=16,col="blue")
```



Elephant Seal

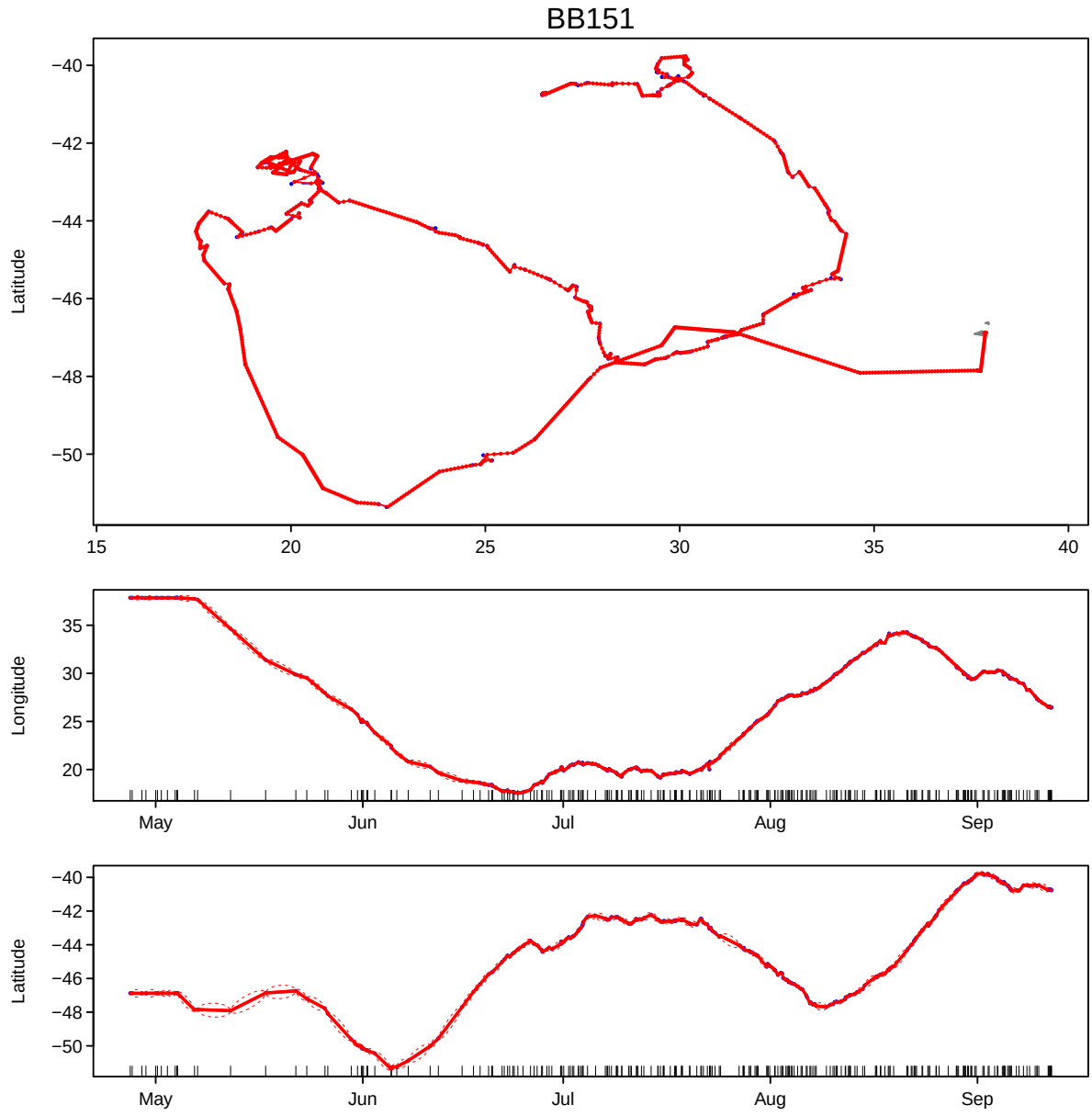
Elephant seal data from Ryan R.

```
d <- read.csv("MAR2015_SES_PTT_BB151_24647.csv",header=T)
```

Filter with crawl (currently cannot use polar correction or drift) and extract sequence of states from a crawl prediction.

```
time_step <- 3
fits <- crawlFilter(d,polar.coord=FALSE,tstep=time_step)
```

```
## Warning in sqrt(diag(Cmat)): NaNs produced
```



```
tp <- templateTrack(fits[[1]])
head(tp$xs)
```

```
##       mu.x      mu.y         nu.x          nu.y
## 1 37.86300 -46.87500 1.331929e-05 -1.994669e-05
## 3 37.86382 -46.87623 2.994201e-04 -4.484053e-04
## 4 37.86472 -46.87758 2.973756e-04 -4.457114e-04
## 6 37.86521 -46.87838 8.980337e-05 -1.721616e-04
## 7 37.86547 -46.87889 8.975486e-05 -1.720977e-04
## 8 37.86574 -46.87941 8.975486e-05 -1.720977e-04
```

Use the fitted model to determine parameters and generate new path fixing both ends, and using a land mask:

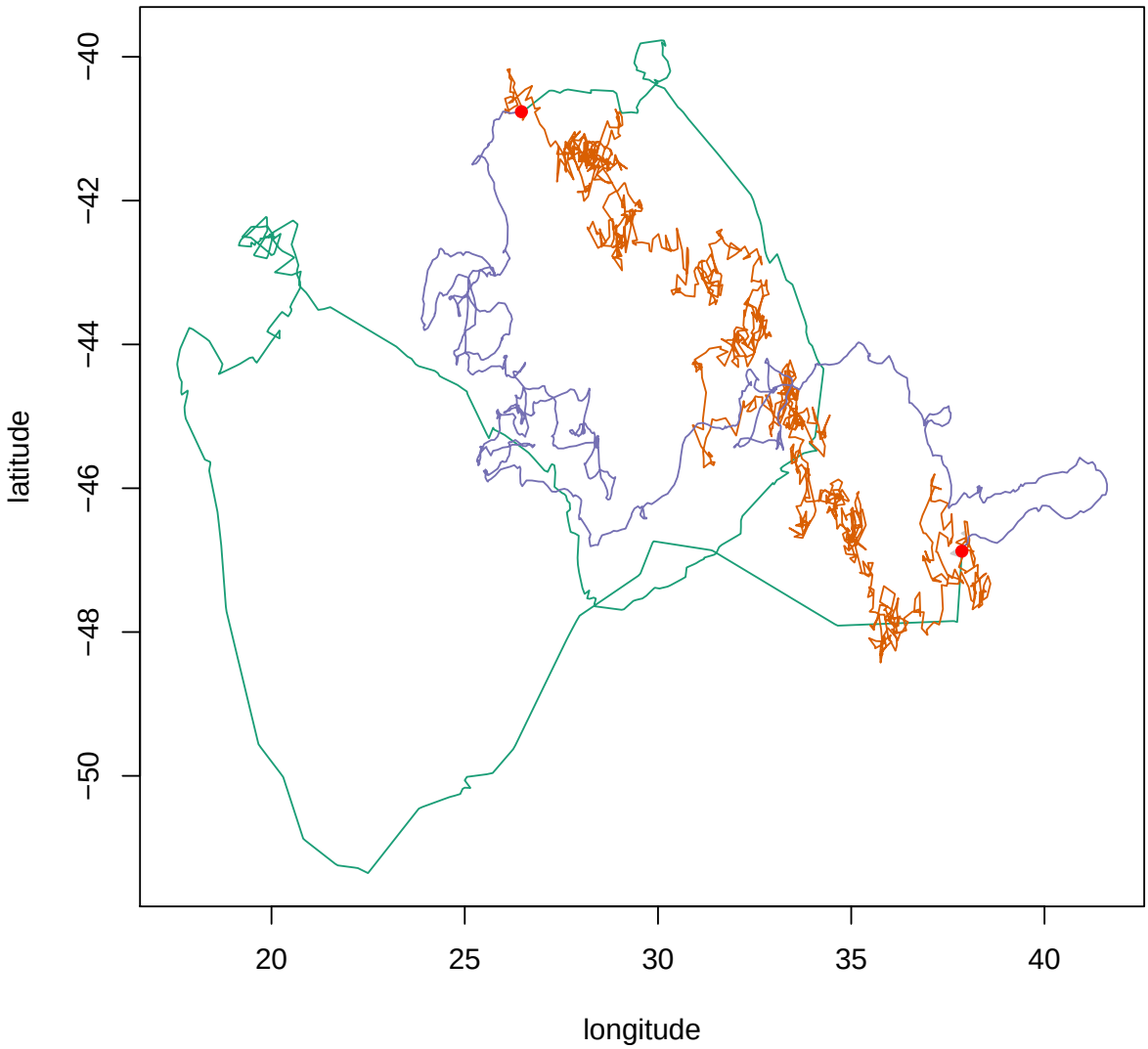
```
model <- surrogateCrawlModel(fits[[1]]$crawl,180/60)
trkcrw <- surrogateCrawl(model,tp$xs,tp$ts,point.check=gshhsMask())
```

Also simulate using the AR method:

```
model <- surrogateARModel(tp$xs)
trkar <- surrogateAR(model,tp$xs,tp$ts,point.check=gshhsMask())
```

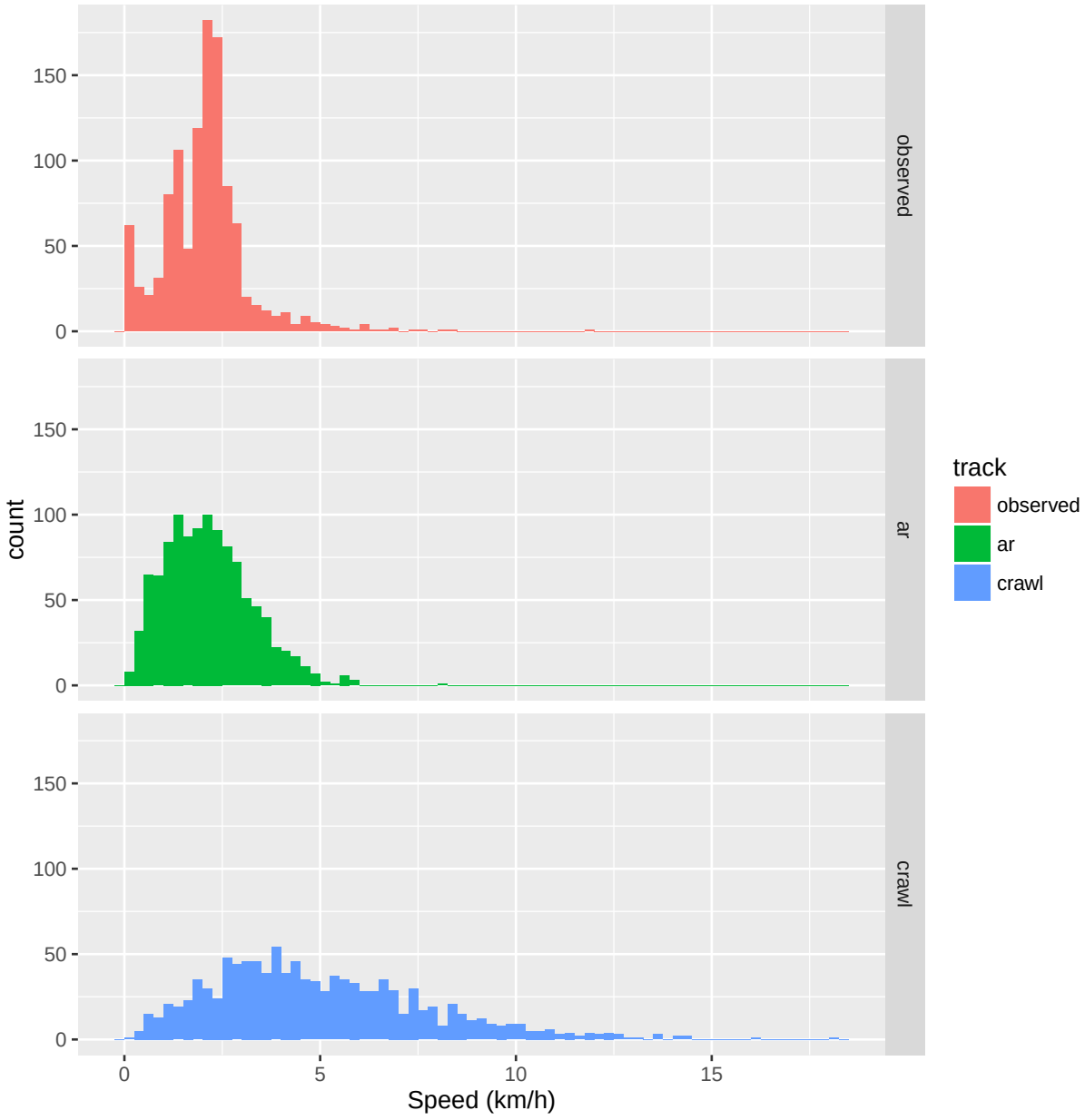
Compare the tracks (original track in green, crawl-simulated track in orange, AR-simulated track in purplish-indigo-whatever):

```
plotTracks(tp,trkcrw,trkar)
points(tp$xs[c(1,nrow(tp$xs)),1:2],pch=16,col="red")
```



Compare speed distributions as we did before:

```
temp=data.frame(speed=distVincentyEllipsoid(tp$xs[-nrow(tp$xs),1:2],tp$xs[-1,1:2])/1e3/time_step,track=
temp=rbind(temp,data.frame(speed=distVincentyEllipsoid(trkar$xs[-nrow(trkar$xs),],trkar$xs[-1,])/1e3/ti
temp=rbind(temp,data.frame(speed=distVincentyEllipsoid(trkcrw$xs[-nrow(trkcrw$xs),1:2],trkcrw$xs[-1,1:2
ggplot(data=temp,aes(x=speed,fill=track))+geom_histogram(binwidth=0.25)+facet_grid(track~.)+scale_x_con
```

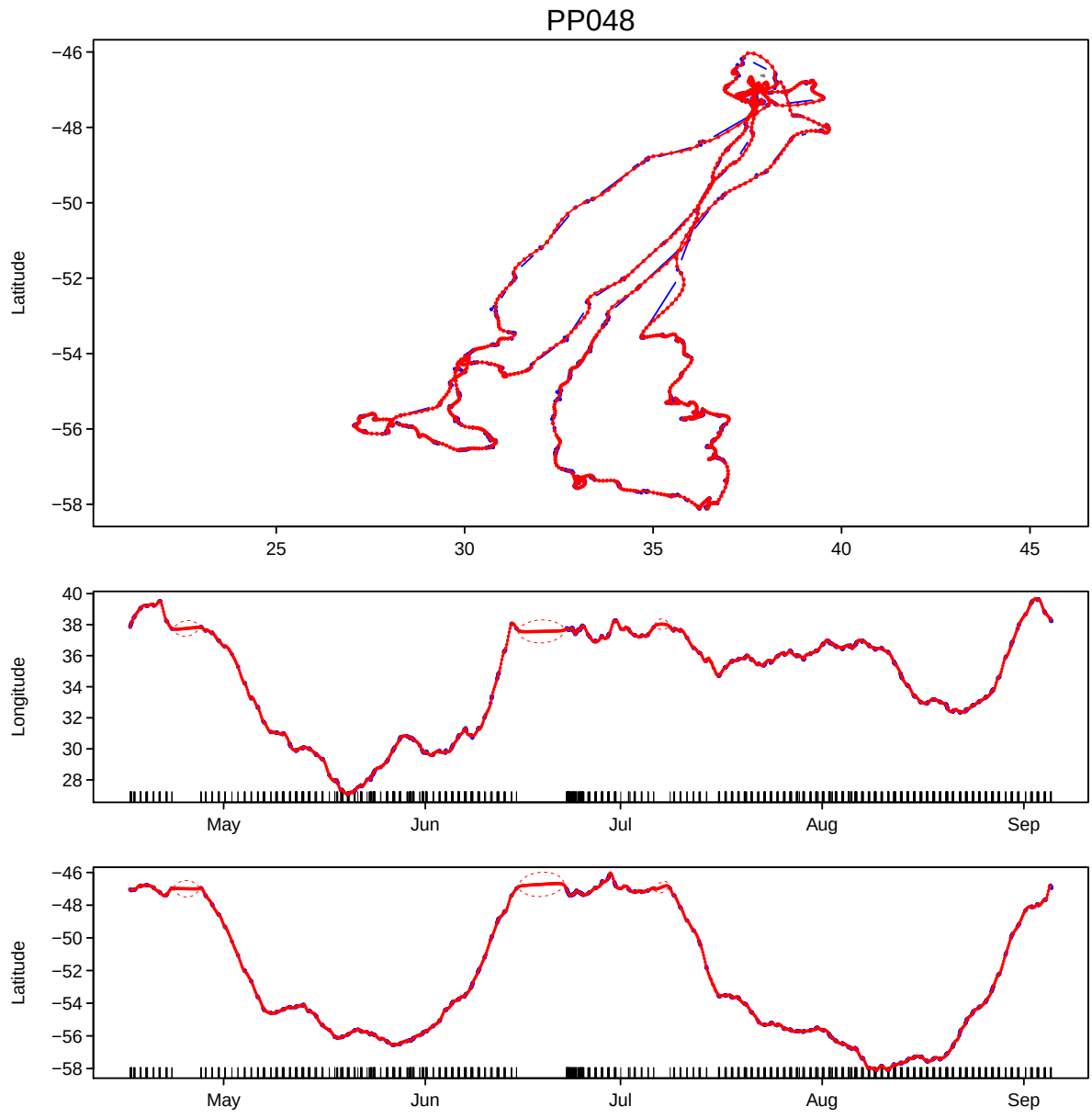


In this instance the crawl-based method doesn't do a particularly good job of reproducing the speeds of the observed track. The AR-based method similarly fails to reproduce the steeply-peaked shape of the distribution, but does produce a simulated track that is plausible in the sense that it has speeds in largely the same range as the observed track.

Antarctic Fur Seal

Fur seal data from Ryan R.

```
d <- read.csv("MAR2015_AFS_PTT_PP048_38622.csv",header=T)
time_step <- 3
fits <- crawlFilter(d,polar.coord=FALSE,tstep=time_step)
```

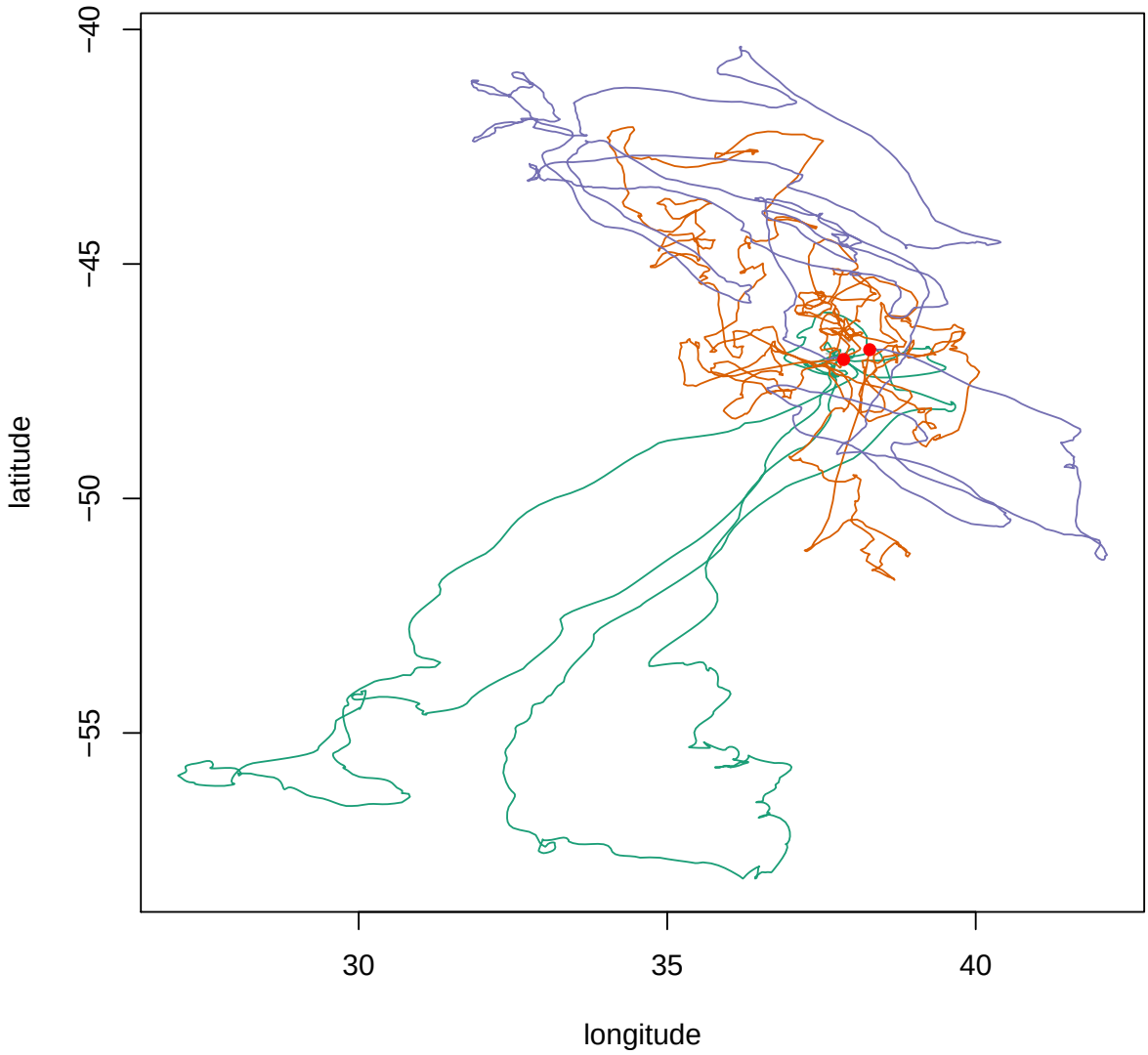



```
tp <- templateTrack(fits[[1]])
head(tp$xs)
```

```
##         mu.x      mu.y       nu.x          nu.y
## 1   37.86012 -47.03775 0.05818132 -0.0151815192
## 7   38.03530 -47.07609 0.05285685 -0.0113232659
## 12  38.17150 -47.06500 0.04350334  0.0089819245
## 13  38.30826 -47.05656 0.04494257 -0.0012839070
## 14  38.42462 -47.06068 0.02977513  0.0005178456
## 18  38.48704 -47.05221 0.01936535  0.0023243401
```

Simulate by both methods:

```
model <- surrogateCrawlModel(fits[[1]]$crawl,180/60)
trkcrw <- surrogateCrawl(model,tp$xs,tp$ts,point.check=gshhsMask())
model <- surrogateARModel(tp$xs)
trkar <- surrogateAR(model,tp$xs,tp$ts,point.check=gshhsMask())
plotTracks(tp,trkcrw,trkar)
points(tp$xs[c(1,nrow(tp$xs)),1:2],pch=16,col="red")
```



Compare speed distributions:

```
temp=data.frame(speed=distVincentyEllipsoid(tp$xs[-nrow(tp$xs),1:2],tp$xs[-1,1:2])/1e3/time_step,track=
temp=rbind(temp,data.frame(speed=distVincentyEllipsoid(trkar$xs[-nrow(trkar$xs),],trkar$xs[-1,])/1e3/ti
```

```
temp=rbind(temp,data.frame(speed=distVincentyEllipsoid(trkcrw$xs[-nrow(trkcrw$xs),1:2],trkcrw$xs[-1,1:2]
ggplot(data=temp,aes(x=speed,fill=track))+geom_histogram(binwidth=0.25)+facet_grid(track~.)+scale_x_cont
```