Mark-Recapture Distance Sampling

Workshop, 28, 31 October 2022

Centre for Research into Ecological and Environmental Modelling

Hidden Markov Line Transect Model Demonstration

1 The `hmltm` package

This document takes you through an example using the `hmltm` package to fit a hidden Markov line transect model to line transect survey data. The package implements the method of Borchers *et al.* (2013), with some extensions, and is available on GitHub here: https://github.com/david-borchers/hmltm. It was not written as a user-friendly general package and has very little in the way of error traps and data checks. Examples of its use can be found in Rekdal *et al.* (2015) and Heide-Jorgensen *et al.* (2017)

2 The Data

The data used in this exercise were gathered on a shipboard survey of beaked whales in the Alboran Sea in 2008, from a vessel moving at about 3.5 m/s. Beaked whales have long dives and spend a lot of time underwater and unobservable. Focal follows of whales gave estimates of mean times available (at surface) and unavailable (diving) in a single dive cycle, of 122 seconds (about 2 minutes) and 1,580 seconds (about 26 minutes), respecively, with standard errors of these means of 9.62 seconds and 134.9 seconds, respectively. With these mean times available and unavailable, animals are available only about 7% of the time!

A total of 81 groups were detected on the survey. The sightings data contain forward distance (y) as well as perpendicular distance (x), and also group size ($size$), and the height of the observer who made the detection (ht). We are going to focus on x and y only here. The survey was not a MRDS survey and so there is no duplicate data.You can load and look at the data like this:

```
library(hmltm)  # load the library
data("beaked.ship") # load the data (contained in the library)
knitr::kable(head(beaked.ship))
```

stratum	area	transect	L	x	y	size	object	ht
1	1000	1	100	88	351	3	1	3.60
1	1000	1	100	798	1228	7	2	17.20
1	1000	1	100	0	1212	2	3	17.20
1	1000	1	100	327	146	2	4	4.75

stratum	area	transect	L	x	y	size	object	ht
1	1000	1	100	2188	1263	4	5	17.20
1	1000	1	100	75	279	5	6	4.75

3 Conventional Distance Sampling Analysis

Let's start by fitting a conventional disance sampling model to the data. To do that we first have to rename some columns in the data frame `beaked.ship` to comply with the conventions of the `Distance` package. We will put the data frame with the new column names in something called `bkdsdat`:

```
bkdsdat = beaked.ship
names(bkdsdat)[1:4] = c("Region.Label", "Area", "Sample.Label", "Effort")
names(bkdsdat)[which(names(bkdsdat)=="x")] = "distance"

knitr::kable(head(bkdsdat))
```

Region.Label	Area	Sample.Label	Effort	distance	y	size	object	ht
1	1000	1	100	88	351	3	1	3.60
1	1000	1	100	798	1228	7	2	17.20
1	1000	1	100	0	1212	2	3	17.20
1	1000	1	100	327	146	2	4	4.75
1	1000	1	100	2188	1263	4	5	17.20
1	1000	1	100	75	279	5	6	4.75

To fit a CDS model, we need the `Distance` package and we need to specify the units of the distances, transect lengths and areas, which are as follows:

```
library(Distance)
conversion.factor <- convert_units("meter", "kilometer", "Square kilometer")
```

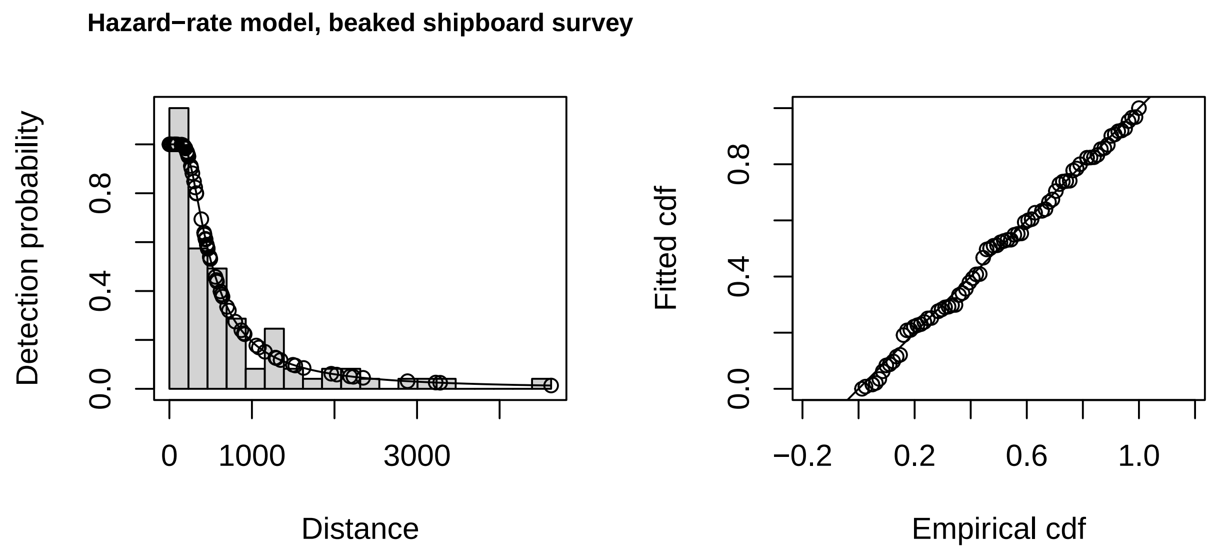
Let's try a half-normal and a hazard-rate model, and compare their AICs:

```
bk.hn <- ds(data=bkdsdat, key="hn", adjustment=NULL,convert_units=conversion.factor)
bk.hr <- ds(data=bkdsdat, key="hr", adjustment=NULL,convert_units=conversion.factor)
knitr::kable(AIC(bk.hn,bk.hr))
```

	df	AIC
bk.hn	1	1259.314
bk.hr	2	1232.746

The hazard-rate model is clearly preferable, so let's check that it seems adequate before proceeding:

```
cutpoints = seq(0,max(na.omit(bkdsdat$distance)),length=21)
par(mfrow=c(1,2))
plot(bk.hr, breaks=cutpoints, main="Hazard-rate model, beaked shipboard survey")
gof_ds(bk.hr)
```



```
##
## Goodness of fit results for ddf object
##
## Distance sampling Cramer-von Mises test (unweighted)
## Test statistic = 0.0291703 p-value = 0.978813
```

That looks OK, so we will proceed using this model. We know that $g(0)$ is less than 1 (because of how long the whales dive), so the CDS estimates will be negatively biased, but lets look at the total abundance estimate anyway:

```
cds.N = c(est=bk.hr$dht$individuals$N$Estimate[3],
          lcl=bk.hr$dht$individuals$N$lcl[3],
          ucl=bk.hr$dht$individuals$N$ucl[3])
caption="CDS Beaked whale abundance estimates"
knitr::kable(t(round(as.data.frame(cds.N))),caption=caption)
```

Table 4: CDS Beaked whale abundance estimates

	est	lcl	ucl
cds.N	361	232	560

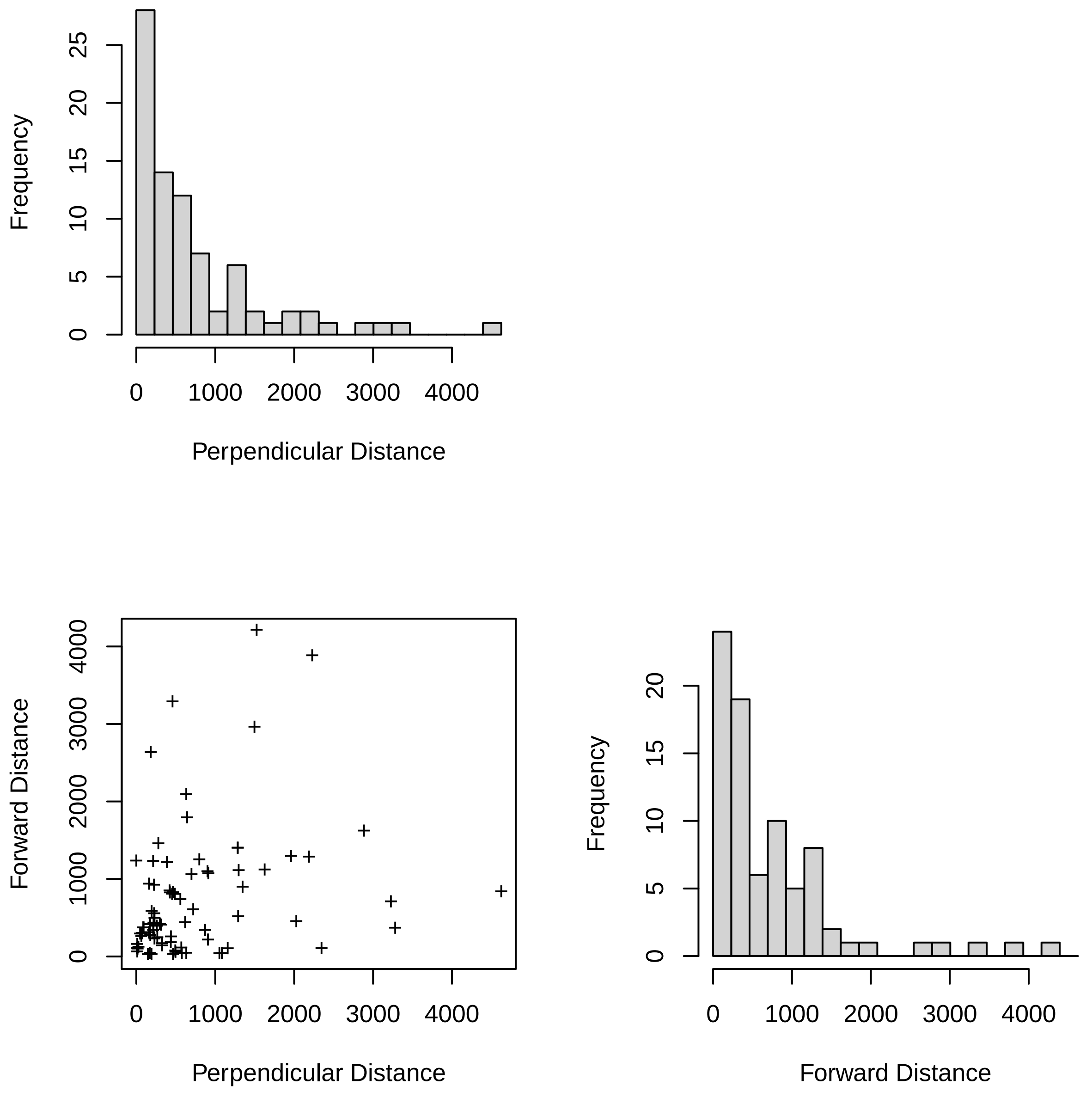
4 CDS with Correction Factors

Let's now calculate various correction factors and use them to "correct'' for $g(0) < 1$. We will do this for the simple (instantaneous) correction factor (which is just the proportion of time available), McLaren's factor, and Laake's factor. The `hmltm` package has functions that allow you to do this, given the mean times available and unavailable, and (optionally) estimates of the CVs of these times. McLaren's and Laake's factors require us to specify a forward distance at which animals are first detectable if they are available (we will call this `ymax`).

Let's plot the data in x- and y- dimensions to figure out what a reasonable `ymax` would be

```
xcuts = seq(0,max(na.omit(beaked.ship$x)),length=21)
ycuts = seq(0,max(na.omit(beaked.ship$y)),length=21)
```

```r
layout(matrix(1:4,nrow=2))
hist(beaked.ship$x,breaks=xcuts,xlab="Perpendicular Distance",main="")
plot(beaked.ship$x, beaked.ship$y, pch="+",xlab="Perpendicular Distance",
     ylab="Forward Distance")
plot(0,0,xaxt="n",yaxt="n",bty="n",xlab="",ylab="",type="n")
hist(beaked.ship$y,breaks=xcuts,xlab="Forward Distance",main="")
```



From this, a `ymax` equal to 4,500 seems reasonable, so let's go with that. We will also calculate the “time in view’’ – the time that that it takes a stationary animal to go from a distance `ymax` ahead of the observer, to abeam of the observer:

```r
ymax = 4500 # distance beyond which cannot detect animal
# Calculate time in view, in minutes
spd = 3.5 # ship speed in m/s
tiv = ymax/spd / 60 # minutes in view
```

The `hmltm` package has a funtion `make.hmm.pars.from.Et` that constructs the HMM parameter object needed for inference from the mean times unavailable, mean time available, and their standard errors, so let's make this object. (By default these mean times are assumed to be independent, although the function also allows you to specify a covariance.)

```
library(hmltm)  # load the library

Eu=1580 # mean time UNavailable -per dive cycle
Ea=122 # mean time available per dive cycle
seEu=134.9 # std error of mean time Unavailable
seEa=9.62 # std error of mean time available

# Make availability parameters object
availpars = make.hmm.pars.from.Et(Eu=Eu, Ea=Ea, seEu=seEu, seEa=seEa)
```

The time an animal is within detectable distance is 21.43 minutes, while the mean time UNavailable is 26.33 minutes and the mean time available is 2.03 minutes.

Clearly $g(0)$ is going to be a lot less than 1 - but how much less? Let's calculate the naive estimate and ask McLaren and Laake. Package `hmltm` provides the functions `simple.a`, `mclaren.a`, and `laake.a` to do the calculations, using objects like `availpars`, which we created above, as input:

```
simple.cf = simple.a(availpars)
laake.cf = laake.a(availpars,spd=spd, ymax=ymax)
mclaren.cf = mclaren.a(availpars,spd=spd, ymax=ymax)
# put them in a data frame for convenience
cf = data.frame(simple=simple.cf$mean,
                mclaren=mclaren.cf$mean,
                laake=laake.cf$mean)
knitr::kable(round(cf,4),caption="Availability correction factors")
```

Table 5: Availability correction factors

simple	mclaren	laake
0.0717	0.8271	0.5886

Let's use these to "correct'' the CDS abundance estimates:

```
# Point estimates of total abundance and 95% CI with each correction method
simple.N = cds.N/cf[,"simple"]
mclaren.N = cds.N/cf[,"mclaren"]
laake.N = cds.N/cf[,"laake"]
Nests = t(data.frame(simple=round(simple.N),
                     mclaren=round(mclaren.N),
                     laake=round(laake.N)))
knitr::kable(Nests, caption="Corrected abundance estimates")
```

Table 6: Corrected abundance estimates

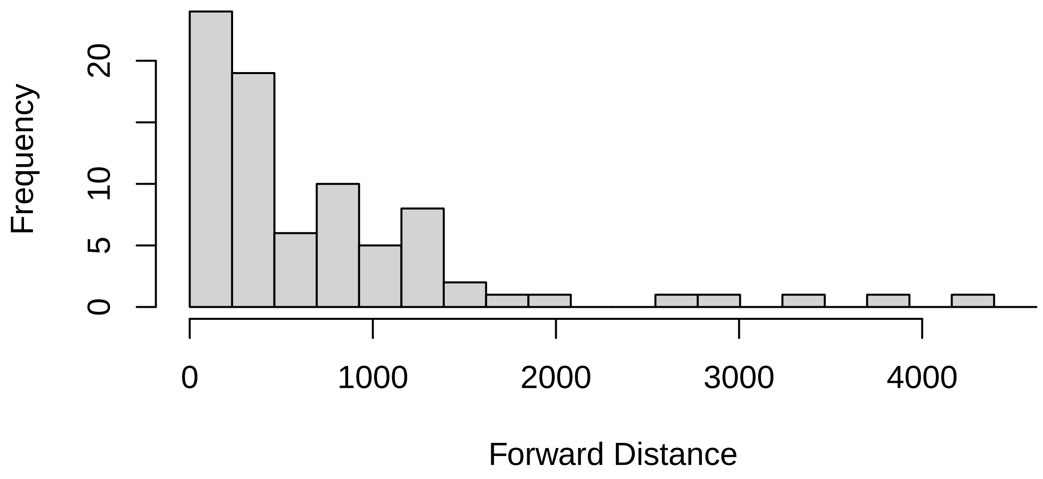
	est	lcl	ucl
simple	5032	3243	7809
mclaren	436	281	677

	est	lcl	ucl
laake	613	395	951

There are big differences between the methods! Which one to believe?

Let's take another look at the forward distance data and think about what this implies for the validity of each of the above correction methods:

```
hist(beaked.ship$y,breaks=xcuts,xlab="Forward Distance",main="")
```



4.1 The simple correction factor

The simple method estimates the probability that a whale is available at a randomly chosen *instant*. But here we look for 21.43 minutes, which is very far from an instant, and so this simple method estimate is going to be way too small, and the corresponding "corrected'' abundance estimate way too big.

4.2 McLaren's correction factor

McLaren's method estimates the probability that a whale is available `at least once` while within detectable distance, but it does this in a way that can allow the probability to go above 1, and Laake's method fixes this problem, so Laake's method is better than McLaren's and we should prefer it to McLaren's.

4.3 Laake's correction factor

Laake's method estimates the probability that a whale is available `at least once` while within detectable distance, and does so in a sensible way, but is this the appropriate correction factor?

Look at the forward distance plot and consider the fact that *where* the animal becomes available between forward distance 0 and 4500 m. If it becomes available anywhere farther ahead than about 1,500 m, there is only a very small chance that it will be detected (we know this because very few detections occurred beyond about 1,500 m). So you would expect that an appropriate correction factor should be smaller than the probability that a whale is available `at least once`, and hence that Laake's correction factor is too big, and the associated abundance estimate too small.

5 The Hidden Markov Line Transect Abundance Estimator

5.1 Setting things up

To use the hidden Markov line transect (HMLT) method, we need first to specify some key survey parameters:

```
# set survey and fitting parameters
spd=3.5 # observer speed
Wl=0 # left-truncation for perpendicular distance (none here)
W=4000 # right-truncatino for perpendicular distance
W.est=W # set perpendicular truncation distance for Horvitz-Thompson-like
#         abundance estimator
ymax=4500 # maximum forward distance beyond which nothing could be detected

# specify survey parameters and put them in an object called `survey.pars`,
# using the hmltm funciton `make.survey.parsz
survey.pars=make.survey.pars(spd=3.5,Wl=0,W=4000,ymax=4500)

# set some fitting parameters
# hessian=FALSE says don't estimate Hessian matrix, and
# nx=64 says use 64 cutpoints when integrating over the perp distance dimension
control.fit=list(hessian=FALSE,nx=64)

# set some parameters for the optimiser
# trace specifies how much to report as function is optimised
# maxit is maximum number of iterations when optimising
# see help(optim) for more
control.opt=list(trace=0,maxit=1000)
```

5.2 Fitting a model

Now we fit the model with the function `est.hmltm`. You can choose from a number of detection hazard model forms (see Table~10) and depending on which model you use, you can specify dependence of the parameters in the x- and/or y-dimension on covariates. All the detection hazard forms assume that an available animal that is at ***radial*** distance zero (not perpendicular distance zero) will be detected. Here we don't use any covariates. You also have to specify paramerter starting values, which is a bit of a pain, but the vignette that comes with the `hmltm` package gives a little guidance on this.

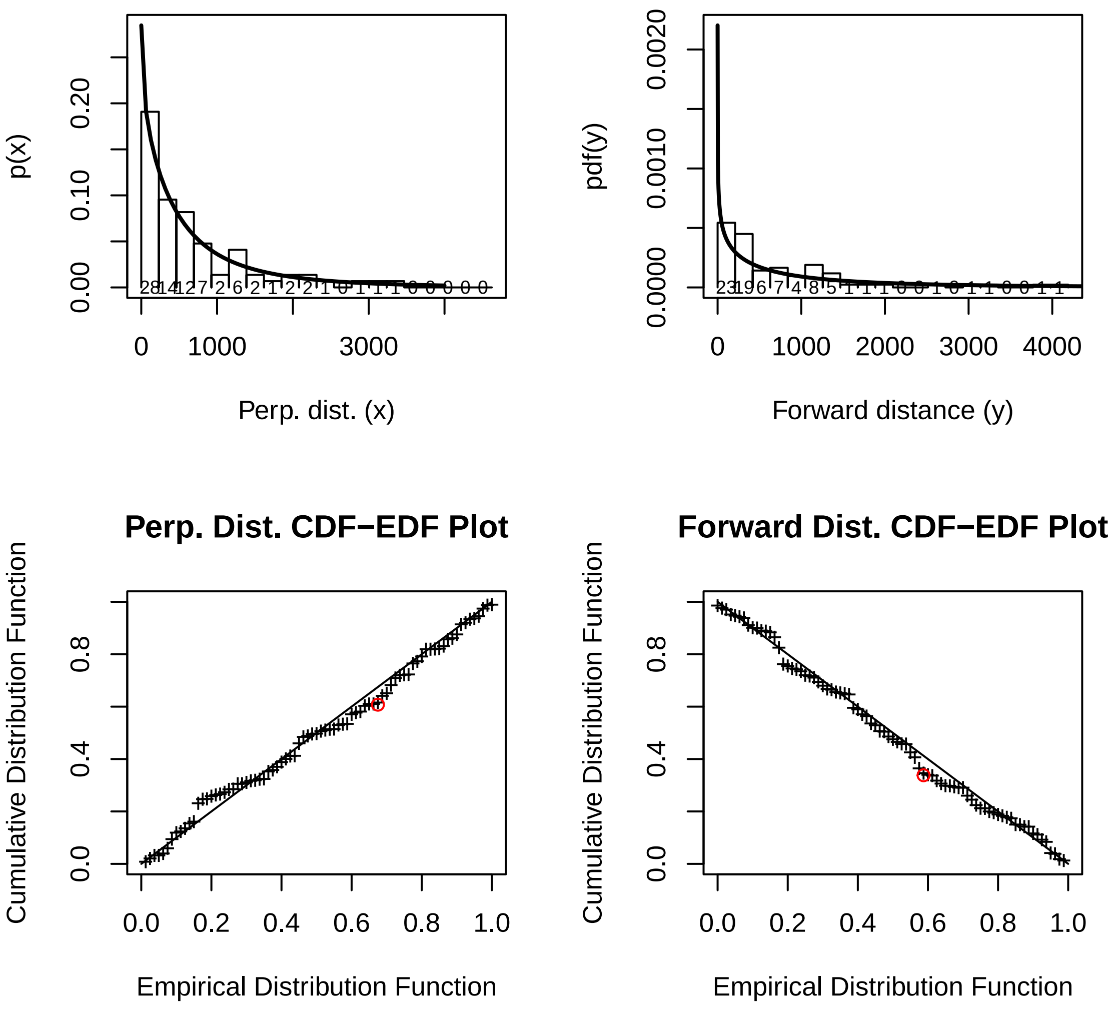
```
# specify model and starting parameter values and fit model
hfun="h.EP2x.0" # detection hazard model to use
models=list(y=NULL,x=NULL) # specify detection hazard model form (no covariates)

# You need to give starting values to the optimiser:
pars=c(0.45, 0.31, 7.7, 62.8) # detection hazard parameter starting values
```

```
# Point estimation:
bkEP2x.null=est.hmltm(beaked.ship,pars,hfun,models,survey.pars,availpars,
                      control.fit,control.opt,W.est=W.est)
```

5.3 Checking goodness-of-fit

We can look at diagnostic plots similarly to the way we do for CDS models, but now we need to consider the goodness-of-fit in both the x- and the y-dimensions

```
# Look at goodness of fit
par(mfrow=c(2,2))
bkEP2x.null.fx=fxfit.plot(bkEP2x.null,breaks=xcuts,type="prob",allx=FALSE)
bkEP2x.null.fy=fyfit.plot(bkEP2x.null,breaks=ycuts,allx=FALSE,nys=250)
bkEP2x.null.gof.x=hmmlt.gof.x(bkEP2x.null)
bkEP2x.null.gof.y=hmmlt.gof.y(bkEP2x.null,breaks=ycuts)
```



These plots and the associated Kolmogorov-Smirnov test statistics for the fits in the x- and the y-dimensions suggest that the model fits pretty well:

```
gof_hmltm = data.frame(perp_dist=bkEP2x.null.gof.x$p.ks,
                       forward_dist=bkEP2x.null.gof.y$p.ks)
knitr::kable(round(gof_hmltm,3),
             caption="Kolmogorov-Smirnov test statistics")
```

Table 7: Kolmogorov-Smirnov test statistics

perp_dist	forward_dist
0.858	0.768

The point estimates are stored in the element `$point$ests` and we can look at them thus:

```
knitr::kable(bkEP2x.null$point$ests,caption="Point estimates")
```

Table 8: Point estimates

stratum	n	k	L	stratum.Area	Dgroups	Ngroups	mean.size	D	N
1	40	5	500	1000	0.317	317	2.2	0.69027	690.3
2	40	6	600	2000	0.264	529	2.8	0.74713	1494.3
Total	80	11	1100	3000	0.282	846	2.6	0.72818	2184.5

We need to bootstrap to get interval estimates.

5.4 Obtaining confidence intervals by bootstrap

Confidence intervals can be obtained by bootstrapping. For comparison with the correction factor methods used above, we treat the avaiability model as fixed and known here (`fixed.avail=TRUE`), but we need not do so in general. Similarly to other `hmltm` functions, you can get help on the use of the boostrap function `bs.hmltm` by typing `help(bs.hmltm)`.

```
set.seed(1)
bests = bs.hmltm(bkEP2x.null,B=99,hmm.pars.bs=availpars,bs.trace=0,report.by=10,
                 fixed.avail=TRUE,bsfile="./objects/beaked.bsests.Rds")
```

The functions `bootsum` and `strat.estable` help you summarise the estimates from bootstraps in a convenient way:

```
bests = readRDS("./objects/beaked.bsests.Rds") # get bootstrap estimates file
bsum = bootsum(bests,bkEP2x.null)
```

```
## Results from 99 bootstrap replicates:
## ----------------------------------------
```

```
hmltmNests = bsum[3,c("N","N.lo","N.hi")]
names(hmltmNests) = colnames(Nests)
Nests = rbind(Nests,hmltm=hmltmNests)
knitr::kable(round(Nests),caption="Abundance estimates")
```

Table 9: Abundance estimates

	est	lcl	ucl
simple	5032	3243	7809
mclaren	436	281	677
laake	613	395	951

	est	lcl	ucl
hmltm	2185	1371	3512

5.5 Conclusion

The HMLT method abundance estimate falls between that from the simple (instantaneous) correction method and that from Laake's correction method – as expected. Moreover, in scenarios in which the survey is not instantaneous (when the simple method is valid) and in which available animals are not certain to be detected when within detectable range (when Laake's method is valid), the correction factor methods are not appropriate.

In such cases the HMLT method provides a valid and rigorous method for estimating density and abundance. It uses more information than any of the correction factor methods (namely the forward distances), and it uses this with an assumption that available animals that are at *radial* distance zero are detected with certainty - a much weaker assumption than that of CDS line transect methods. It comes with the usual maximum likelihood-based model selection and diagnostic tools. It does require forward distance data.

6 Other Detection Hazard Forms and Covariates

Covariates always affect the scale parameter, not the shape parameter. The reason for this is historical: line transect estimators tend to be implemented with covariates affecting the scale parameter. The only rationale I have ever found for this is a line transect study of beer cans reported in Pollock et al. (1990) in which having covariates affect the scale parameter of detection functions was found to be more effective than having them affect the shape parameter.

Table 10: Forms and parameters of the detection hazard function, $h(x, y)$. Here x and y are perpendicular and forward distances, respectively, $\mathbf{X}$ and $\mathbf{X}_x$ are design matrices for covariates affecting the scale parameters and $\boldsymbol{\beta}$ and $\boldsymbol{\beta}_x$ are parameter vectors associated with the covariates affecting the scale parameters.

		Shape	Scale parameter(s)	
Name	Form	parameter(s)	No covariates	With covariates
h.IP.0	$\sigma^\gamma / (\sigma^2 + x^2 + y^2)^{\gamma/2}$	γ	σ	$\sigma \exp(\mathbf{X}\boldsymbol{\beta})$
h.EP1.0	$\exp\left\{-\left(\frac{x}{\sigma}\right)^\gamma - \left(\frac{y}{\sigma}\right)^\gamma\right\}$	γ	σ	$\sigma \exp(\mathbf{X}\boldsymbol{\beta})$
h.EP2.0	$\exp\left\{-\left(\frac{x}{\sigma}\right)^{\gamma_x} - \left(\frac{y}{\sigma}\right)^{\gamma_y}\right\}$	γ_x, γ_y	σ	$\sigma \exp(\mathbf{X}\boldsymbol{\beta})$
h.EP1x.0	$\exp\left\{-\left(\frac{x}{\sigma_x}\right)^\gamma - \left(\frac{y}{\sigma}\right)^\gamma\right\}$	γ	σ, σ_x	$\sigma \exp(\mathbf{X}\boldsymbol{\beta})$, $\sigma_x \exp(\mathbf{X}_x\boldsymbol{\beta}_x)$
h.EP2x.0	$\exp\left\{-\left(\frac{x}{\sigma_x}\right)^{\gamma_x} - \left(\frac{y}{\sigma}\right)^\gamma\right\}$	γ, γ_x	σ, σ_x	$\sigma \exp(\mathbf{X}\boldsymbol{\beta})$, $\sigma_x \exp(\mathbf{X}_x\boldsymbol{\beta}_x)$

Deciding on starting parameter values involves some trial and error. A reasonable starting value for σ is to make it similar size to the average x and y (when the model does not include σ_x), average y (when the model includes σ_x). A reasonable value for σ_x is something around the size of the average x. Shape parameter values between about 0.5 and 2 have worked for cases I've looked at, so a value around 1 might be a reasonable starting value guess.

When you've got covariates you also need to specify starting values for the parameters associated with the covariates. You do this by specifying starting values for $\exp(\beta)$ for each β in the vector $\boldsymbol{\beta}$. Some guidance can be got from the facts that

$\exp(\beta) = 1$ implies the covariate has no effect,

$0 < \exp(\beta) < 1$ implies that increasing values of the covariate decrease σ,

$1 < \exp(\beta)$ implies that increasing values of the covariate increase σ.

In the absence of information about the effect of the covariate on σ a starting value of $\exp(\beta) = 1$ seems sensible.

If we let $e^{\beta_1}, \ldots, e^{\beta_m}$ be the starting values for the m covariates affecting σ and $e^{\beta_{x1}}, \ldots, e^{\beta_{xm_x}}$ be the starting values for the m_x covariates affecting σ_x, then the starting values are put in a vector (named `pars` here) in the order shown below, omitting values

for any parameters that are not relevant for the hazard function and model being used:\
`pars=c(`γ,γ_x,σ,$e^{\beta_1}, \ldots, e^{\beta_m}$, σ_x,$e^{\beta_{x1}}, \ldots, e^{\beta_{xm_x}}$`)`.

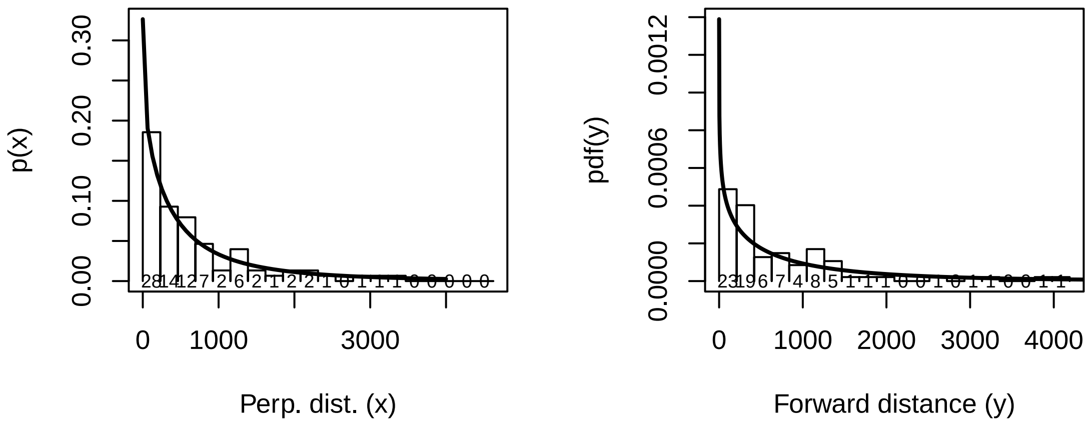
6.1 Example: Model `h.EP1x.0` without covariates

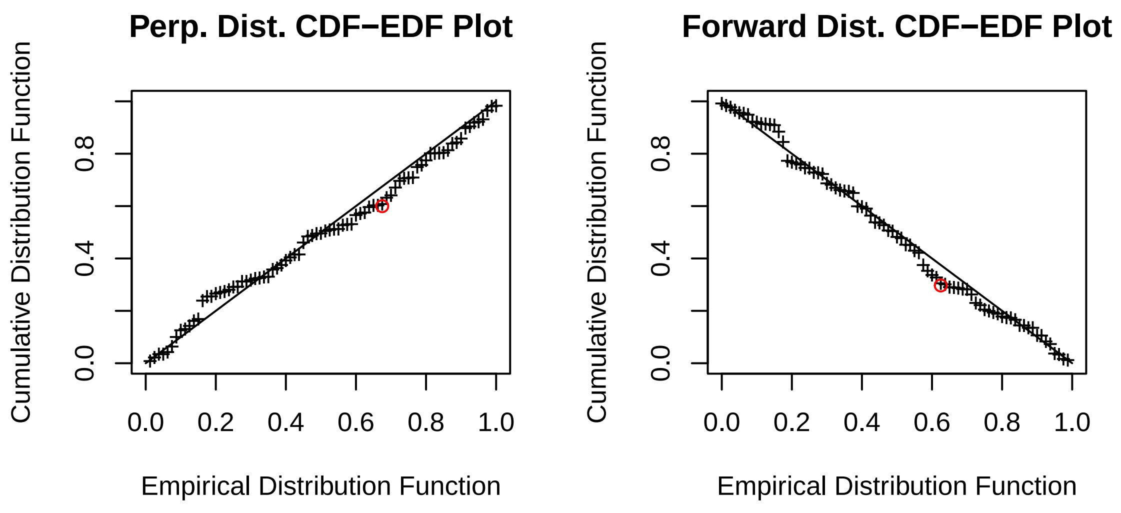
We fit the model as before, we just now have to (a) specify model 'h.EP1x.0' instead of 'h.EP2x.0', and (b) becausee model 'h.EP1x.0' has only one shape parameter, γ, we drop the second parameter from the starting parameter vector `pars` that we used for model 'h.EP2x.0' (which has two shape parameters, γ and γ_x).

```
# specify model and starting parameter values and fit model
hfun1="h.EP1x.0" # detection hazard model to use
models=list(y=NULL,x=NULL) # specify detection hazard model form (no covariates)

# You need to give starting values to the optimiser:
pars=c(0.45, 7.7, 62.8) # detection hazard parameter starting values

# Point estimation:
bkEP1x.null=est.hmltm(beaked.ship,pars,hfun1,models,survey.pars,availpars,
                      control.fit,control.opt,W.est=W.est)
```

We check goodness-of-fit as before, with plots and Kolmogorov-Smirnov test statistics:

```
# Look at goodness of fit
par(mfrow=c(2,2))
bkEP1x.null.fx=fxfit.plot(bkEP1x.null,breaks=xcuts,type="prob",allx=FALSE)
bkEP1x.null.fy=fyfit.plot(bkEP1x.null,breaks=ycuts,allx=FALSE,nys=250)
bkEP1x.null.gof.x=hmmlt.gof.x(bkEP1x.null)
bkEP1x.null.gof.y=hmmlt.gof.y(bkEP1x.null,breaks=ycuts)
```





```r
gof1_hmltm = data.frame(perp_dist=bkEP1x.null.gof.x$p.ks,
                        forward_dist=bkEP1x.null.gof.y$p.ks)
knitr::kable(round(gof1_hmltm,3),
             caption="Kolmogorov-Smirnov test statistics")
```

Table 11: Kolmogorov-Smirnov test statistics

perp_dist	forward_dist
0.754	0.715

The fit is adequate although not quite as good as was the ‘h.EP2x.0’ – which is to be expected since model ‘h.EP1x.0’ has one less parameter than model ‘h.EP2x.0’. We can use AIC to choose between them:

```r
knitr::kable(AIC.hmltm(bkEP2x.null,bkEP1x.null,
                        criterion="AIC"),caption="AIC Table")
```

Table 12: AIC Table

	y.model	x.model	npar	logLik	AIC	AICc	dAIC	AICwt
bkEP1x.null	~1	~1	3	454.67	915.344	915.66	0.000	0.6606
bkEP2x.null	~1	~1	4	454.34	916.676	917.21	1.332	0.3394

From this we'd prefer model 'h.EP1x.0'.

6.2 Example: Model `h.EP1x.0` with covariates

```
load("hmltmDemoFits.RData") # load objects fitted below - to save time
```

A model with σ depending on group size:

```
hfun1="h.EP1x.0"
model.ssy=list(y="~size",x=NULL)
pars=c(0.45, 7.7, 1, 62.8) # put 1 in slot for sigma size parameter
bkEP1x.ssy=est.hmltm(beaked.ship,pars,hfun1,model.ssy,survey.pars,availpars,
                     control.fit,control.opt,W.est=W.est)
```

A model with σ_x depending on group size:

```
hfun1="h.EP1x.0"
model.ssx=list(y=NULL,x="~size")
pars=c(0.45, 7.7, 62.8, 1) # put 1 in slot for sigma_x size parameter
bkEP1x.ssx=est.hmltm(beaked.ship,pars,hfun1,model.ssx,survey.pars,availpars,
                     control.fit,control.opt,W.est=W.est)
```

A model with both σ and σ_x depending on group size:

```
hfun1="h.EP1x.0"
model.ssx.ssy=list(y="~size",x="~size")
pars=c(0.45, 7.7, 1, 62.8, 1) # put 1 in slot for sigma size parameter
#                               and for sigma_x size parameter
bkEP1x.ssx.ssy=est.hmltm(beaked.ship,pars,hfun1,model.ssx.ssy,survey.pars,availpars,
                     control.fit,control.opt,W.est=W.est)
```

A model with σ depending on group size and σ_x depending on platform height:

```
hfun1="h.EP1x.0"
model.htx.ssy=list(y="~size",x="~ht")
pars=c(0.45, 7.7, 1, 62.8, 1) # put 1 in slot for sigma size parameter
#                               and for sigma_x size parameter
bkEP1x.htx.ssy=est.hmltm(beaked.ship,pars,hfun1,model.htx.ssy,survey.pars,availpars,
                     control.fit,control.opt,W.est=W.est)
# When first ran models, saved them thus:
# save(bkEP2x.null,bkEP1x.null,bkEP1x.ssy,bkEP1x.ssx,bkEP1x.ssx.ssy,bkEP1x.htx.ssy,
#      file="hmltmDemoFits.RData")
```

Compare AICs:

```
knitr::kable(AIC.hmltm(bkEP2x.null,bkEP1x.null,bkEP1x.ssy,bkEP1x.ssx,
                       bkEP1x.ssx.ssy,bkEP1x.htx.ssy,
                       criterion="AIC"),caption="AIC Table")
```

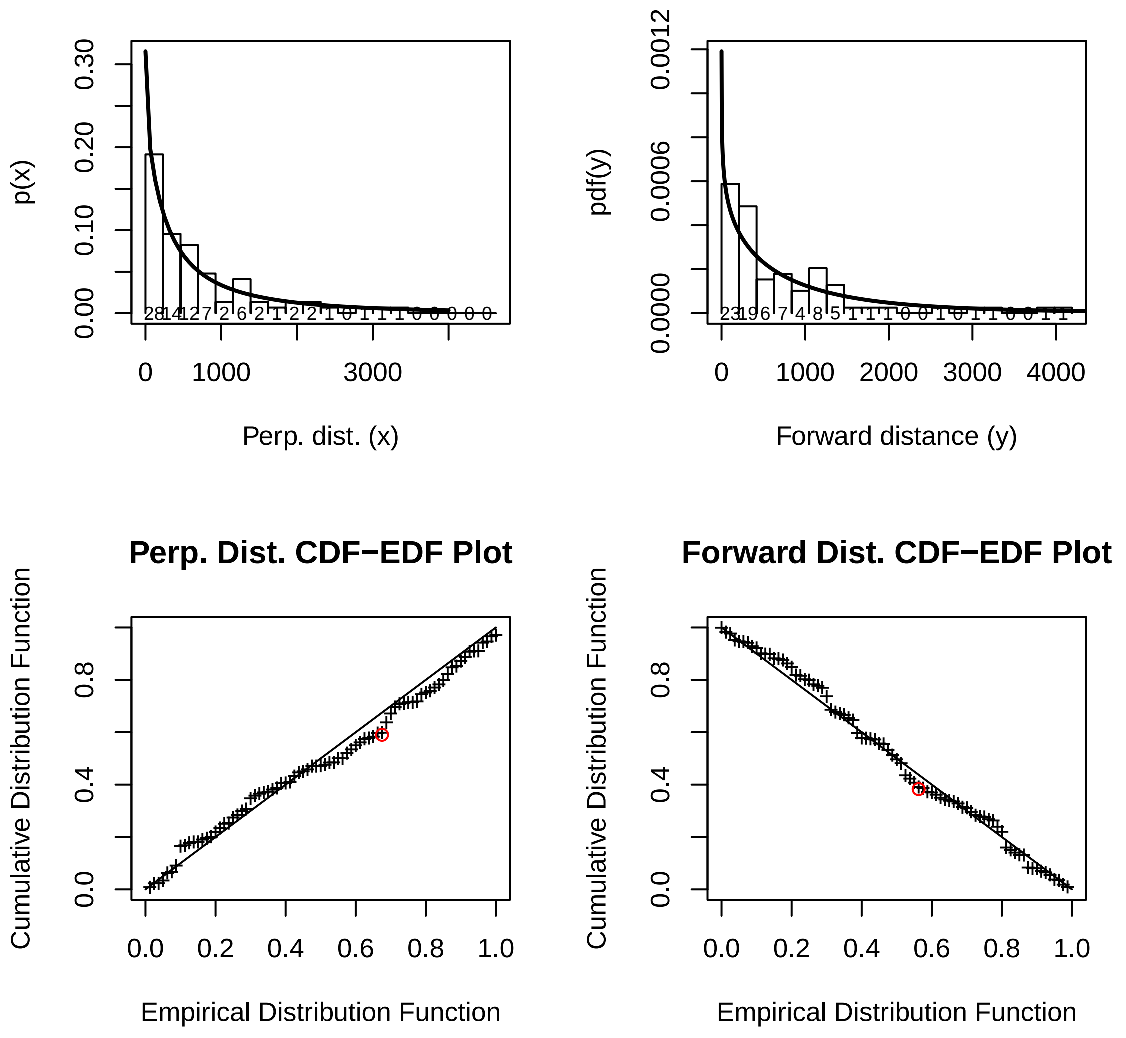
Table 13: AIC Table

	y.model	x.model	npar	logLik	AIC	AICc	dAIC	AICwt
bkEP1x.htx.ssy	~size	~ht	5	444.65	899.302	900.113	0.000	1
bkEP1x.ssy	~size	~1	4	452.79	913.584	914.117	14.282	0

	y.model	x.model	npar	logLik	AIC	AICc	dAIC	AICwt
bkEP1x.null	~1	~1	3	454.67	915.344	915.660	16.042	0
bkEP1x.ssx.ssy	~size	~size	5	452.77	915.544	916.355	16.242	0
bkEP1x.ssx	~1	~size	4	454.16	916.316	916.849	17.014	0
bkEP2x.null	~1	~1	4	454.34	916.676	917.210	17.374	0

The last-fitted model is the best of those considered, so lets look at its diagnostics:

```r
# Look at goodness of fit
par(mfrow=c(2,2))
bkEP1x.htx.ssy.fx=fxfit.plot(bkEP1x.htx.ssy,breaks=xcuts,type="prob",allx=FALSE)
bkEP1x.htx.ssy.fy=fyfit.plot(bkEP1x.htx.ssy,breaks=ycuts,allx=FALSE,nys=250)
bkEP1x.htx.ssy.gof.x=hmmlt.gof.x(bkEP1x.htx.ssy)
bkEP1x.htx.ssy.gof.y=hmmlt.gof.y(bkEP1x.htx.ssy,breaks=ycuts)
```



```r
gof1_hmltm = data.frame(perp_dist=bkEP1x.htx.ssy.gof.x$p.ks,
                        forward_dist=bkEP1x.htx.ssy.gof.y$p.ks)
knitr::kable(round(gof1_hmltm,3),
             caption="Kolmogorov-Smirnov test statistics")
```

Table 14: Kolmogorov-Smirnov test statistics

perp_dist	forward_dist
0.613	0.971

The point estimates are as follows:

```
knitr::kable(bkEP1x.htx.ssy$point$ests,
             caption="Point estimates for best model to date")
```

Table 15: Point estimates for best model to date

stratum	n	k	L	stratum.Area	Dgroups	Ngroups	mean.size	D	N
1	40	5	500	1000	0.295	295	1.8	0.52819	528.2
2	40	6	600	2000	0.282	564	2.7	0.74912	1498.2
Total	80	11	1100	3000	0.286	858	2.4	0.67548	2026.4

6.3 References:

Borchers, D.L., Zucchini, W., Heide-Jørgensen, M.P., Canadas, A. & Langrock, R. (2013). Using hidden Markov models to deal with availability bias on line transect surveys. *Biometrics* **69**, 703– 713.

Heide-Jørgensen M.P.,Hansen, R.G, Fossette, S., Nielsen, N.H., Borchers, D.L., Stern, H. and Witting, L. (2017). Rebuilding belugas stocks in West Greenland. *Animal Conservation* **20**, 282-293.

Pollock, K.H. and Otto, M.C. (1990). Size bias in line transect sampling: a field test. *Biometrics* **46**, 239-245.

Rekdal, S.L., Hansen, R.G., Borchers, D.L., Bachmann, L., Laidre, K.L., Wiig, Ø., Nielsen, N.H., Fossette, S., Tervo, O. & Heide-Jørgensen, M.P. (2015). Trends in bowhead whales in West Greenland: aerial surveys vs. genetic capture-recapture analyses. *Mar. Mamm. Sci.* **31**, 133– 154.