Mark-recapture Distance Sampling

Workshop, 24-25 August 2020

Centre for Research into Ecological and Environmental Modelling

Hidden Markol Line Transect Model Exercise

1 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 and 1,580 seconds, respecively, with standard errors of these means of 9.62 seconds and 134.9 seconds, respectively.

A total of 81 groups were detected. The sightings data contain forward distance (y) as well as perpendicular distance (x), and also group size ($size$), Beaufort sea state (bf) 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 the data contains no duplicate data.

```
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	bf	object	ht
1	1000	1	100	88	351	3	3	1	3.60
1	1000	1	100	798	1228	7	7	2	17.20
1	1000	1	100	0	1212	2	2	3	17.20
1	1000	1	100	327	146	2	2	4	4.75
1	1000	1	100	2188	1263	4	4	5	17.20
1	1000	1	100	75	279	5	5	6	4.75

2 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	bf	object	ht
1	1000	1	100	88	351	3	3	1	3.60
1	1000	1	100	798	1228	7	7	2	17.20
1	1000	1	100	0	1212	2	2	3	17.20
1	1000	1	100	327	146	2	2	4	4.75
1	1000	1	100	2188	1263	4	4	5	17.20
1	1000	1	100	75	279	5	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")
```

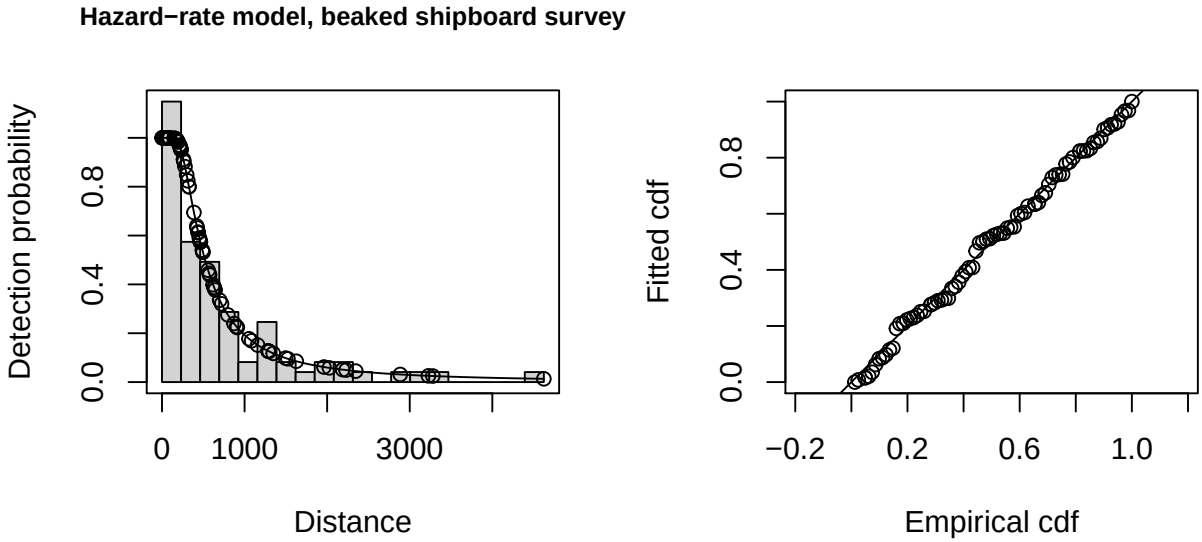
Lets try a half-normal and a hazard-rate model, and compare their AICs:

```
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 lets 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 = 0.0291797 p-value = 0.978784
```

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

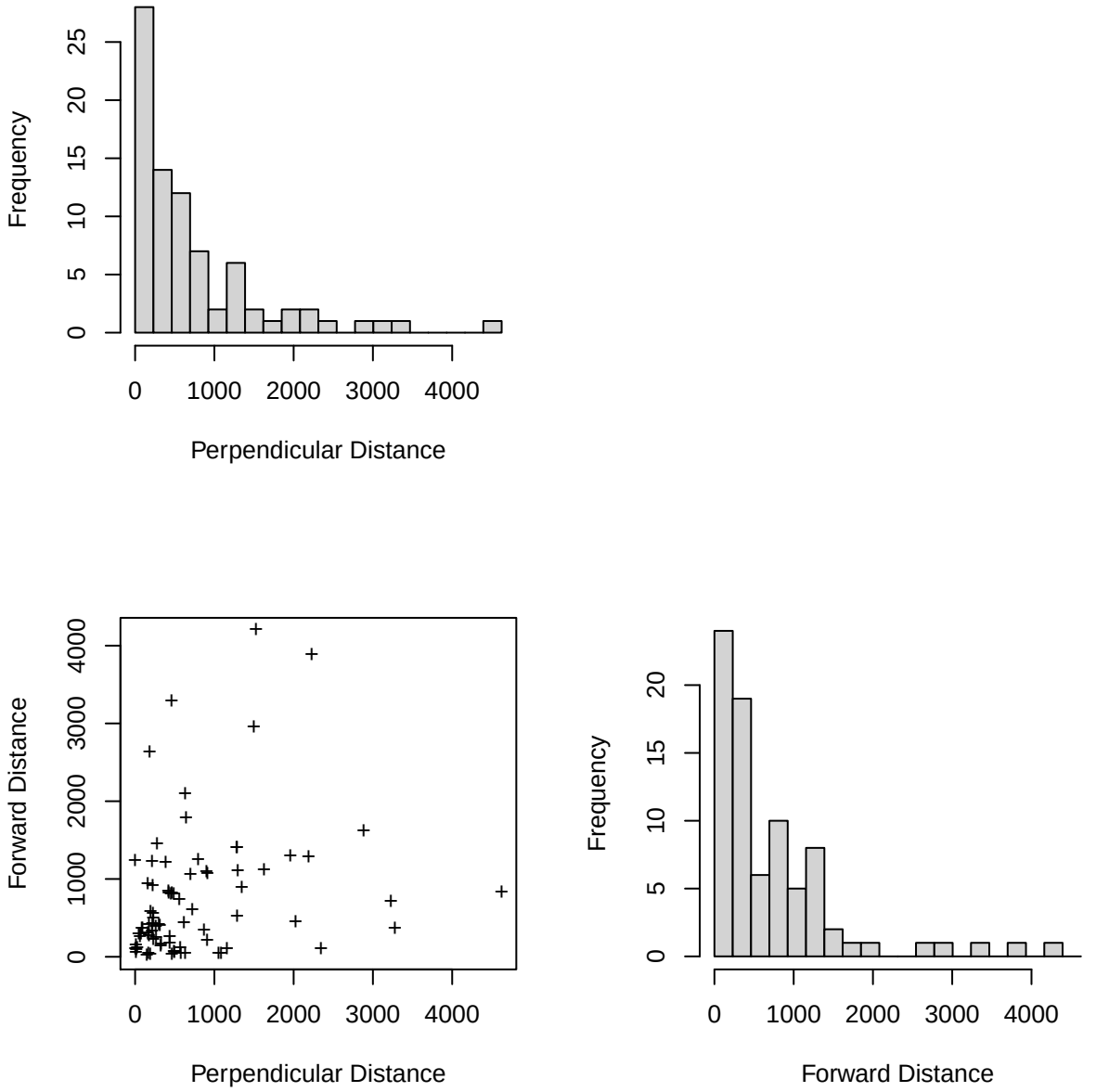
3 CDS with Correction Factors

Lets now calculate various correction factors and use them to "correct" for $g(0) < 1$. We will do this for the simple (instantaneous) correction factor (whihc 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`).

Lets 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)
```

```
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 we will go with that. 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, `ymax`, and survey platform speed:

```
ymax = 4500 # max forward distance can detect animal

Eu=1580 # mean time UNavailable -per dive cycle
Ea=122 # mean time available per dive cycle
spd = 3.5 # ship speed in m/s
# Calculate time in view, in minutes
```

```
tiv = ymax/spd / 60 # minutes in view

# Make availability parameters object
availpars = make.hmm.pars.from.Et(Eu=Eu, Ea=Ea, seEu=134.9, seEa=9.62)
```

From this we see that 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? Lest 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 list 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

Lets 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")
```

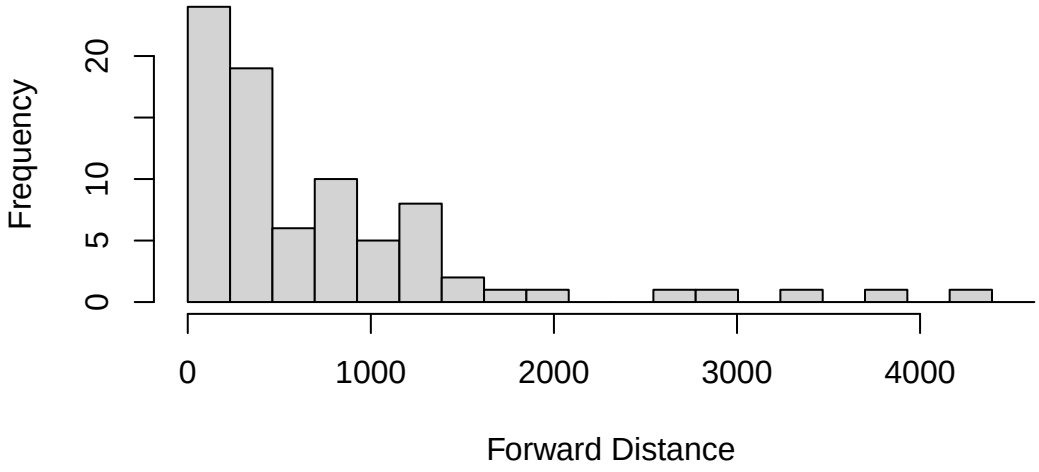
Table 6: Corrected abundance estimates

	est	lcl	ucl
simple	5031	3243	7806
mclaren	436	281	676
laake	613	395	951

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

Lets 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="")
```



3.1 The simple correction factor

The simple method estimates the probability that a whale is available at a randomly chosen *instant*. But here the whales are available (to some extent) 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.

3.2 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.

3.3 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.

4 The Hidden Markov Line Transect Abundance Estimator

4.1 The `hmltm` package

We will use the `hmltm` package to fit a hidden Markov line transect model to these survey data. The package 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. It has been used for a number of real analyses and so while I am pretty sure it is error free (or as error-free as a package with its level of use would typically be), I do not vouch for its ease of use, and nor do I guarantee that if you try it with new data that nothing will go wrong – something probably will go wrong!

4.2 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=FALSDE 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)
```

4.3 Fitting a model

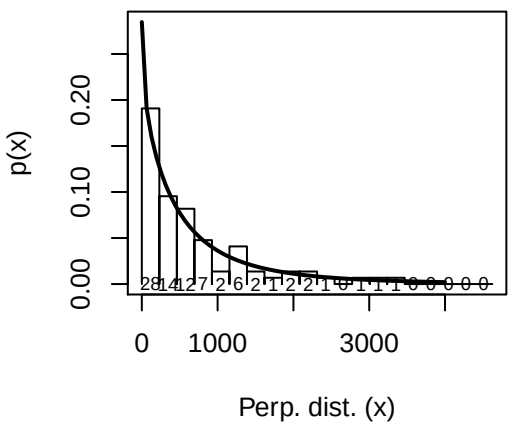
Now we fit the model with the function `est.hmltm`. You can choose from a number of detection hazard model forms (see the vignette that comes with the `hmltm` package) and depending on which model you use, you can specirfy dependence of the parameters in the x- and/or y-dimension on covariates. All the detection hazard forms assume that detection of an available animal that is at *radial* distance zero (not perpendicular distance zero) are 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 give a little guidance on this.

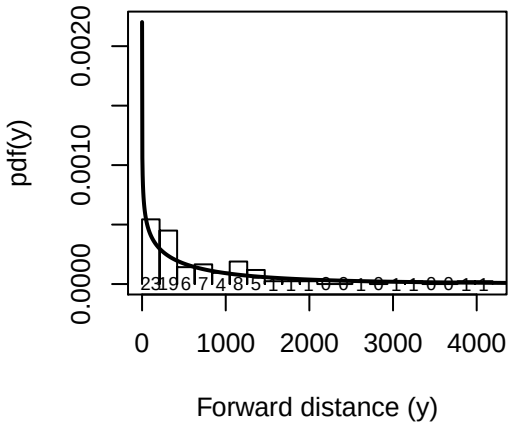
```
# specify model and starting parameter values and fit model
hfun="h.EP2x.0" # detection hazard model to use
models=list(y=NULL,x=NULL) # specify detection hazard model form (no covariates)

# 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)
```

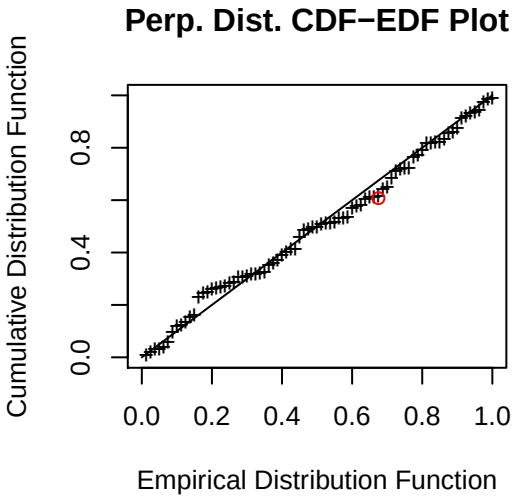
4.4 Checking goodness-of-fit

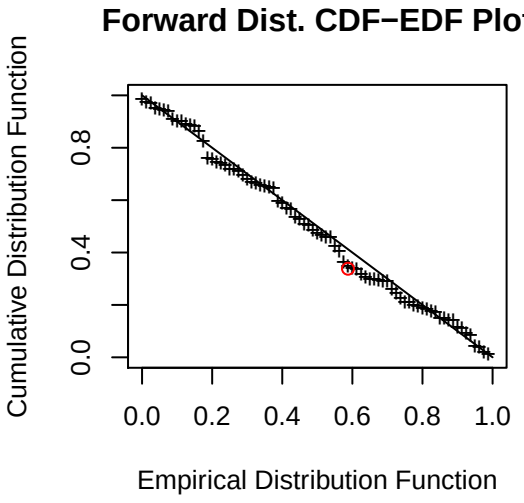
We can look at diagnostic plots similarly to the way we do for CDS models, but now we need to consider the goodness-of-fit in both the x- and the y-dimensions

```
# 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

4.5 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=50,hmm.pars.bs=availpars,bs.trace=0,report.by=10,
                 fixed.avail=TRUE,bsfile="./objects/beaked.bsests.RData")
```

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

```
load("./objects/beaked.bsests.RData") # get bootstrap estimates file
bsum = bootsum(bests)
```

```
## Results from 50 bootstrap replicates:
## ---------------------------------------
```

```
estable = strat.estable(bkEP2x.null$point$ests,bsum$cv)
hmltmNests = estable[3,c("N","N.lo","N.hi")]
names(hmltmNests) = colnames(Nests)
Nests = rbind(Nests,hmltm=hmltmNests)
knitr::kable(round(Nests),caption="Abundance estimates")
```

Table 8: Abundance estimates

	est	lcl	ucl
simple	5031	3243	7806
mclaren	436	281	676
laake	613	395	951
hmltm	2184	1562	3069

4.6 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.