Advanced topics in distance sampling

Workshop, 26-30 August 2019

Centre for Research into Ecological and Environmental Modelling

Solution 3. Detection function fitting

Here is a "solution" for exercise 3. As with any data analysis, there is no correct answer, but this shows how I would approach this analysis.

Much of the text below is as in the exercise itself, so it should be relatively easy to navigate.

Additional text and code is highlighted using boxes like this.

Preamble

First need to load the requisite R libraries

```
library(rgdal)
```

```
## Loading required package: sp
```

```
## rgdal: version: 1.4-4, (SVN revision 833)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
##  Path to GDAL shared files: C:/Users/louise/Documents/R/win-library/3.6/rgdal/gdal
##  GDAL binary built with GEOS: TRUE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: C:/Users/louise/Documents/R/win-library/3.6/rgdal/pr
##  Linking to sp version: 1.3-1
```

```
library(ggplot2)
library(Distance)
```

```
## Loading required package: mrds
```

```
## This is mrds 2.2.1
## Built: R 3.6.1; ; 2019-07-17 13:15:38 UTC; windows
```

```
##
## Attaching package: 'Distance'
```

```
## The following object is masked from 'package:mrds':
##
##     create.bins
```

```
library(knitr)
```

Load the data

The observation data here has all we need here for fitting a detection function…

```
load("sperm-data.RData")
distdata <- obs
```

We can check it has the correct format using `head`:

```
head(distdata)
```

```
##    Survey GroupSize SeaState  Distance        SightingTime SegmentID
## 1 en04395         2      3.0  246.0173 2004/06/28 10:22:21        48
## 2 en04395         2      2.5 1632.3934 2004/06/28 13:18:14        50
## 3 en04395         1      3.0 2368.9941 2004/06/28 14:13:34        51
## 4 en04395         1      3.5  244.6977 2004/06/28 15:06:01        52
## 5 en04395         1      4.0 2081.3468 2004/06/29 10:48:31        56
## 6 en04395         1      2.4 1149.2632 2004/06/29 14:35:34        59
##   SightingID coords.x1 coords.x2  distance object Sample.Label size
## 1          1   -65.636    39.576  246.0173      1           48    2
## 2          2   -65.648    39.746 1632.3934      2           50    2
## 3          3   -65.692    39.843 2368.9941      3           51    1
## 4          4   -65.717    39.967  244.6977      4           52    1
## 5          5   -65.820    40.279 2081.3468      5           56    1
## 6          6   -65.938    40.612 1149.2632      6           59    1
```

The `Distance` package expects certain column names to be used. Renaming is much easier to do in R than ArcGIS, so we do it here.

```
distdata$distance <- distdata$Distance
distdata$object <- distdata$SightingID
distdata$size <- distdata$GroupSize
```

Let's see what we did:

```
head(distdata)
```

```
##    Survey GroupSize SeaState  Distance        SightingTime SegmentID
## 1 en04395         2      3.0  246.0173 2004/06/28 10:22:21        48
## 2 en04395         2      2.5 1632.3934 2004/06/28 13:18:14        50
## 3 en04395         1      3.0 2368.9941 2004/06/28 14:13:34        51
## 4 en04395         1      3.5  244.6977 2004/06/28 15:06:01        52
## 5 en04395         1      4.0 2081.3468 2004/06/29 10:48:31        56
## 6 en04395         1      2.4 1149.2632 2004/06/29 14:35:34        59
##   SightingID coords.x1 coords.x2  distance object Sample.Label size
## 1          1   -65.636    39.576  246.0173      1           48    2
## 2          2   -65.648    39.746 1632.3934      2           50    2
## 3          3   -65.692    39.843 2368.9941      3           51    1
## 4          4   -65.717    39.967  244.6977      4           52    1
## 5          5   -65.820    40.279 2081.3468      5           56    1
## 6          6   -65.938    40.612 1149.2632      6           59    1
```

We now have four “extra” columns.

Exploratory analysis

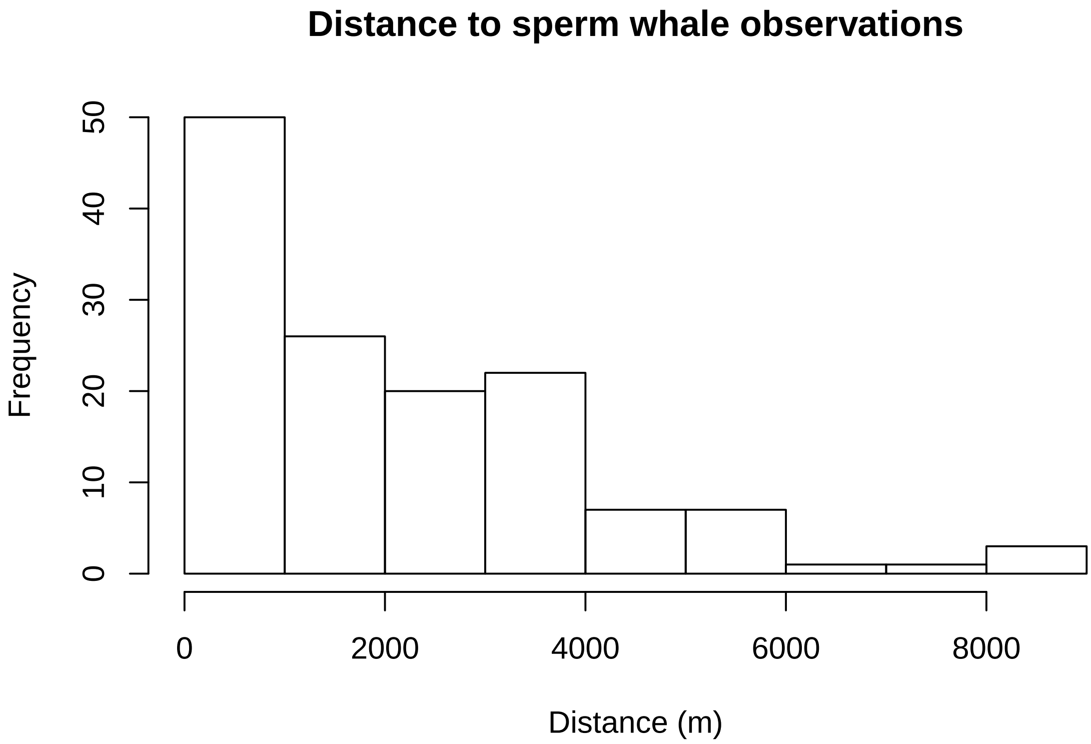
Before setting off fitting detection functions, let’s look at the relationship of various variables in the data.

Don’t worry too much about understanding the code that generates these plots at the moment.

Distances

Obviously, the most important covariate in a distance sampling analysis is distance itself. We can plot a histogram of the distances to check that (1) we imported the data correctly and (2) it conforms to the usual shape for line transect data.

```
hist(distdata$distance, xlab="Distance (m)",
     main="Distance to sperm whale observations")
```



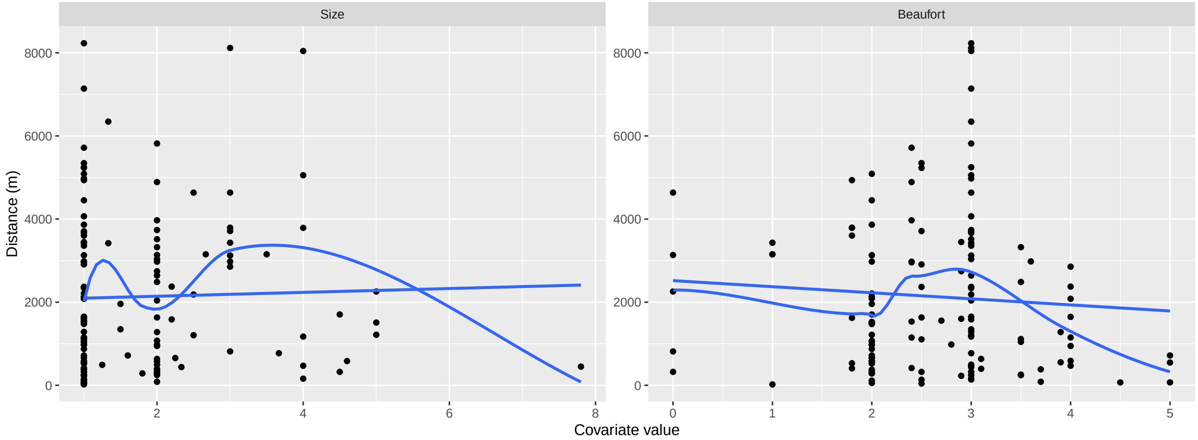
Size and distance

We might expect that there will be a relationship between the distance at whcih we see animals and the size of the groups observed (larger groups are easier to see at larger distances), so let's plot that to help us visualise the relationship.

```
# plot of size versus distance and sea state vs distance,
# - linear model and LOESS smoother overlay

# put the data into a simple format, only selecting what we need
distplot <- distdata[,c("distance","size","SeaState")]
names(distplot) <- c("Distance", "Size", "Beaufort")
library(reshape2)
# "melt" the data to have only three columns (try head(distplot))
distplot <- melt(distplot, id.vars="Distance", value.name="covariate")

# make the plot
p <- ggplot(distplot, aes(x=covariate, y=Distance)) +
      geom_point() +
      facet_wrap(~variable, scale="free") +
      geom_smooth(method="loess", se=FALSE) +
      geom_smooth(method="lm", se=FALSE) +
      labs(x="Covariate value", y="Distance (m)")
print(p)
```

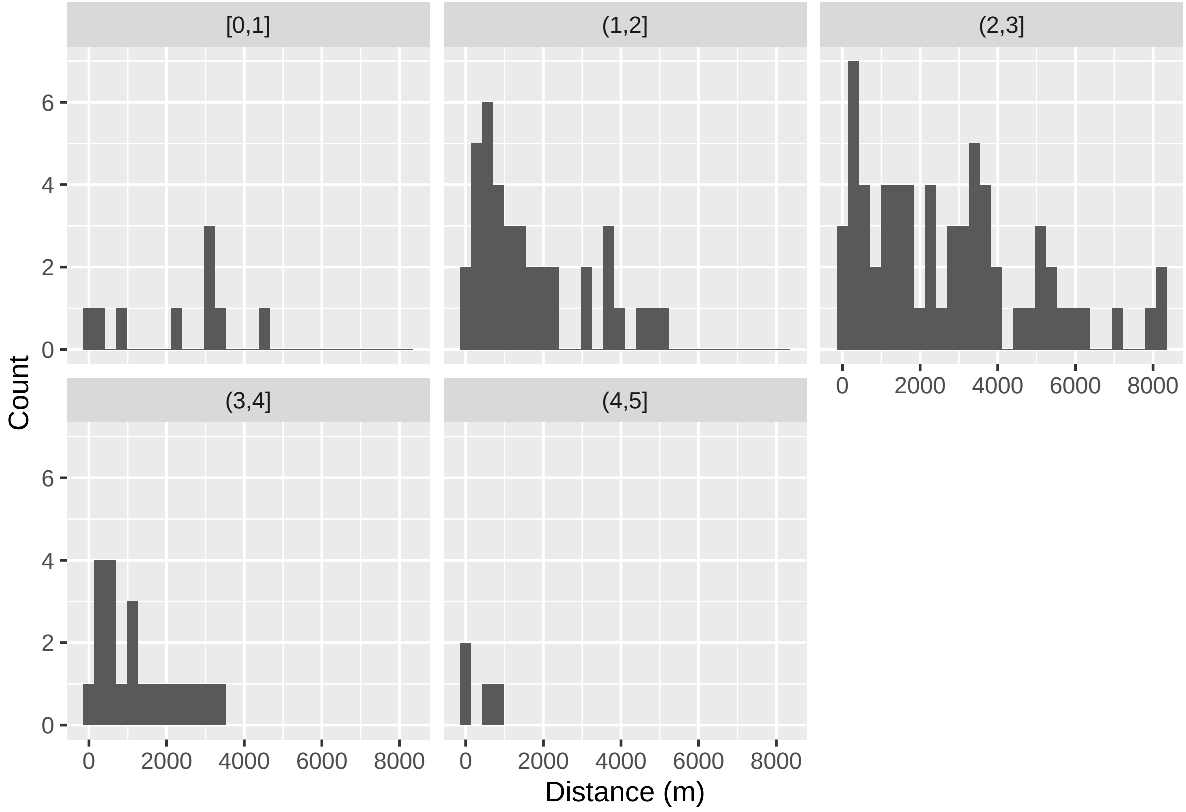


Distance and sea state

We might also expect that increaing sea state would result in a drop in observations. We can plot histograms of distance for each sea state level (making the sea state take only values 0,1,2,4,5 for this).

```
distdata$SeaStateCut <- cut(distdata$SeaState,seq(0,5,by=1), include.lowest=TRUE)
p <- ggplot(distdata) +
      geom_histogram(aes(distance)) +
      facet_wrap(~SeaStateCut) +
      labs(x="Distance (m)", y="Count")
print(p)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

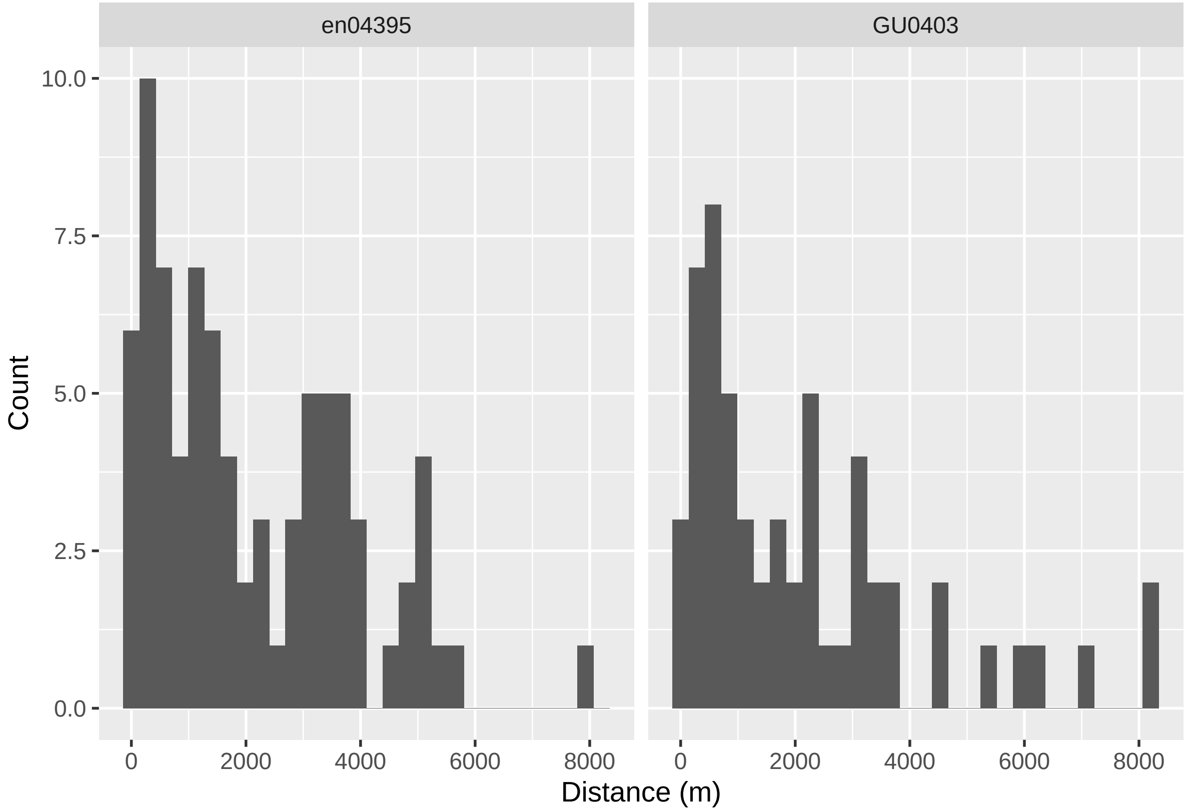


Survey effect

Given we are including data from two different surveys we can also investigate the relationship between survey and distances observed.

```
p <- ggplot(distdata) +
      geom_histogram(aes(distance)) +
      facet_wrap(~Survey) +
      labs(x="Distance (m)", y="Count")
print(p)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Fitting detection functions

It's now time to fit some detection function models. We'll use the `ds()` function from the `Distance` package to fit the detection function. You can access the help file for the `ds()` function by typing `?ds` – this will give you information about what the different arguments to the function are and what they do.

We can fit a very simple model with the following code:

```
df_hn <- ds(data=distdata, truncation=6000, key="hn", adjustment=NULL)
```

```
## Fitting half-normal key function
```

```
## Key only model: not constraining for monotonicity.
```

```
## AIC= 2252.06
```

```
## No survey area information supplied, only estimating detection function.
```

Let's dissect the call and see what each argument means:

- `data=distdata`: the data to use to fit the model, as we prepared above.
- `truncation=6000`: set the truncation distance. Here, observations at distances greater than 6000m will be discarded before fitting the detection function.
- `key="hn"`: the key function to use for the detection function, in this case half-normal (`?ds` lists the other options).

- `adjustment=NULL`: adjustment term series to fit. `NULL` here means that no adjustments should be fitted (again `?ds` lists all options).

Other useful arguments for this practical are:

- `formula=`: gives the formula to use for the scale parameter. By default it takes the value `~1`, meaning the scale parameter is constant and not a function of covariates.
- `order=`: specifies the "order" of the adjustments to be used. This is a number (or vector of numbers) specifying the order of the terms. For example `order=2` fits order 2 adjustments, `order=c(2,3)` will fit a model with order 2 and 3 adjustments (mathematically, it only makes sense to include order 3 with order 2). By default the value is `NULL` which has `ds()` select the number of adjustments by AIC.

Summaries

We can look at the summary of the fitted detection function using the `summary()` function:

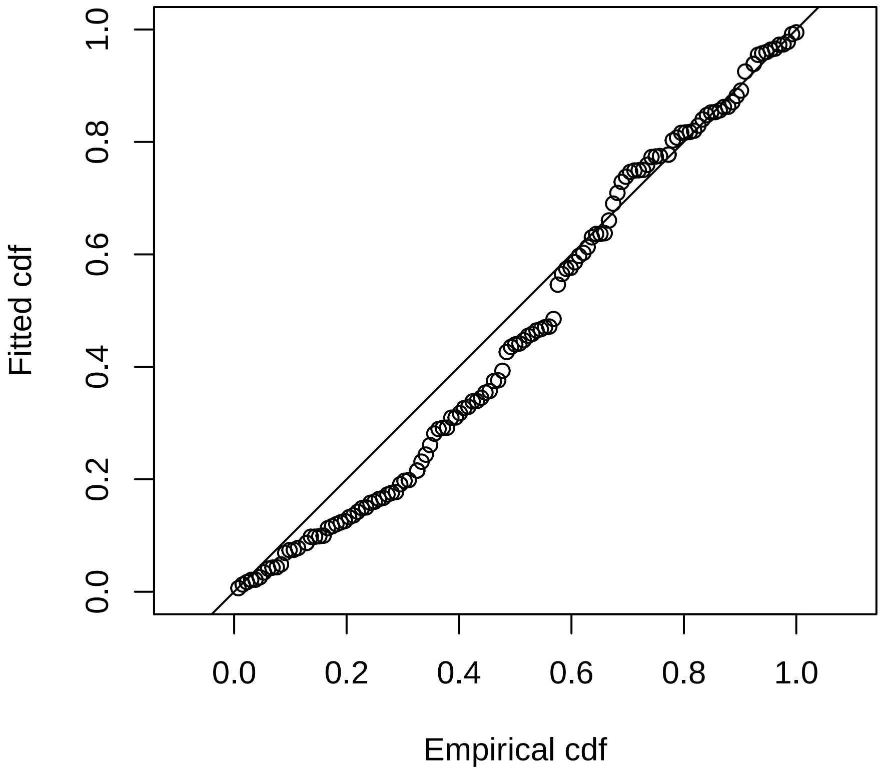
```
summary(df_hn)
```

```
##
## Summary for distance analysis
## Number of observations :  132
## Distance range         :  0  -  6000
##
## Model : Half-normal key function
## AIC   : 2252.06
##
## Detection function parameters
## Scale coefficient(s):
##             estimate         se
## (Intercept) 7.900732 0.07884776
##
##                        Estimate          SE         CV
## Average p             0.5490484  0.03662569 0.06670757
## N in covered region 240.4159539 21.32287580 0.08869160
```

Goodness of fit

Goodness of fit quantile-quantile plot and test results can be accessed using the `gof_ds()` function:

```
gof_ds(df_hn)
```



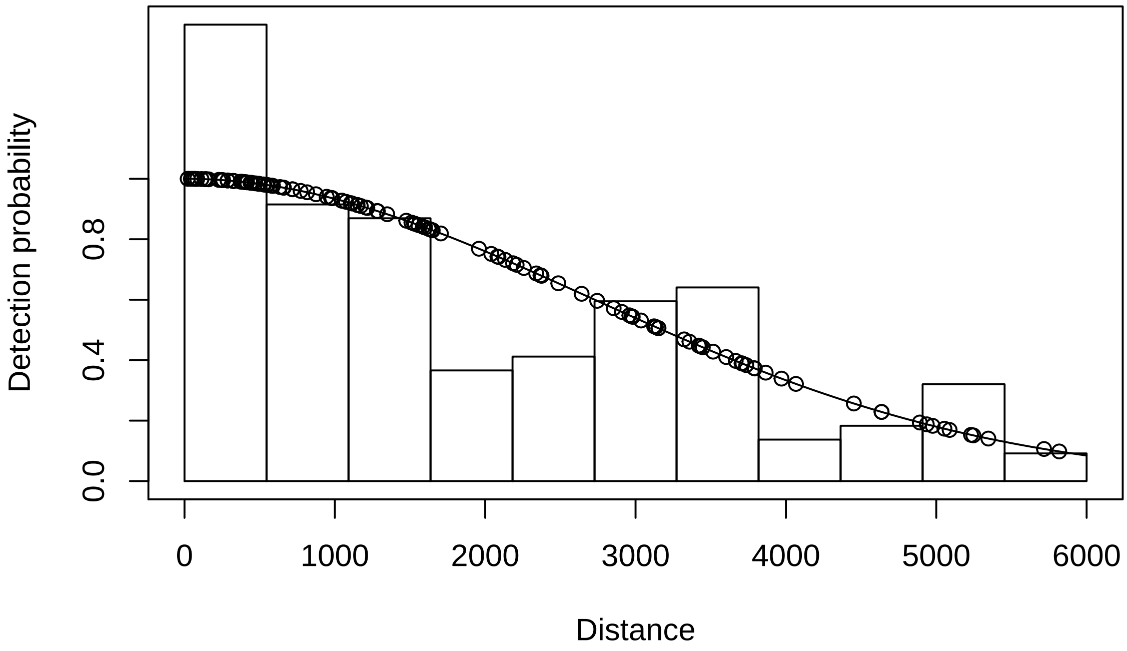
```
##
## Goodness of fit results for ddf object
##
## Distance sampling Cramer-von Mises test (unweighted)
## Test statistic = 0.396179 p-value = 0.0739475
```

Note we are ignoring the χ^2 test results, as they rely on binning the distances to calculate test statistics where as Cramer-von Mises and Kolmogorov-Smirnov tests do not (so they have more power).

Plotting

We can plot the models simply using the `plot()` function:

```
plot(df_hn)
```



The dots on the plot indicate the distances where observations are. We can remove them (particularly useful for a model without covariates) using the additional argument `showpoints=FALSE` (try this out!).

Now you try…

Now try fitting a few models and comparing them using AIC. Don't try to fit all possible models, just try a selection (say, a hazard-rate, a model with adjustments and a couple with different covariates). You can also try out changing the truncation distance.

Here's an example to work from. Some tips before you start:

- You can include as many lines as you like in a given chunk (though you may find it more managable to separate them up, remembering each time to give the chunk a unique name).
- You can run the current line of code in RStudio by hitting `Control+Enter` (on Windows/Linux; `Command+Enter` on Mac).
- Giving your models informative names will help later on! Here I'm using `df_` to indicate that this is a detection function, then shortened forms of the model form and covariates, separated by underscores, but use what makes sense to you (and future you!).

```
df_hr_ss_size <- ds(distdata, truncation=6000, adjustment=NULL, key="hr",
                    formula=~SeaState+size)
```

```
## Fitting hazard-rate key function
## AIC= 2249.327
```

```
## No survey area information supplied, only estimating detection function.
```

Once you've got the hang of writing models and looking at the differences between them, you should move onto the next section.

Since I wasn't constrained for time, here we can fit all possible detection function models with half-normal and hazard-rate functions using each combination of covariates. This is somewhat brute force.

Our covariates `size` and `SeaState` might be interesting…

```
table(distdata$SeaState)
```

```
##
##    0   1 1.8   2 2.4 2.5 2.7 2.8 2.9   3 3.1 3.5 3.6 3.7 3.9   4 4.5   5
##    5   4   7  31   8  10   1   1   4  41   2   6   1   2   2   8   1   3
```

Oh, yikes, the sea state probably was averaged from multiple observers… Let's chop it up before we start modelling.

```
distdata$SeaState <- cut(distdata$SeaState, breaks=0:5, include.lowest=TRUE)
table(distdata$SeaState)
```

```
##
## [0,1] (1,2] (2,3] (3,4] (4,5]
##     9    38    65    21     4
```

You could check for sensitivity in choppings too…

We'll store all the models in a list that we can then iterate over later.

```
models <- list()

# half-normal models
models$hn <- ds(distdata, truncation=6000, adjustment=NULL)
```

```
## Fitting half-normal key function
```

```
## Key only model: not constraining for monotonicity.
```

```
## AIC= 2252.06
```

```
## No survey area information supplied, only estimating detection function.
```

```
models$hn.ss <- ds(distdata, truncation=6000, adjustment=NULL, formula=~SeaState)
```

```
## Fitting half-normal key function
```

```
## AIC= 2236.726
```

```
## No survey area information supplied, only estimating detection function.
```

```
models$hn.size <- ds(distdata, truncation=6000, adjustment=NULL, formula=~size)
```

```
## Fitting half-normal key function
## AIC= 2253.892
```

```
## No survey area information supplied, only estimating detection function.
```

```
models$hn.ss.size <- ds(distdata, truncation=6000, adjustment=NULL,
                        formula=~size+SeaState)
```

```
## Fitting half-normal key function
```

```
## AIC= 2238.411
```

```
## No survey area information supplied, only estimating detection function.
```

```
models$hn.survey <- ds(distdata, truncation=6000, adjustment=NULL,
                       formula=~as.factor(Survey))
```

```
## Fitting half-normal key function
```

```
## AIC= 2252.35
```

```
## No survey area information supplied, only estimating detection function.
```

```
models$hn.survey.ss <- ds(distdata, truncation=6000, adjustment=NULL,
                          formula=~as.factor(Survey)+SeaState)
```

```
## Fitting half-normal key function
```

```
## AIC= 2235.885
```

```
## No survey area information supplied, only estimating detection function.
```

```
models$hn.survey.size <- ds(distdata, truncation=6000, adjustment=NULL,
                            formula=~as.factor(Survey)+size)
```

```
## Fitting half-normal key function
```

```
## AIC= 2254.339
```

```
## No survey area information supplied, only estimating detection function.
```

```
models$hn.survey.size.ss <- ds(distdata, truncation=6000, adjustment=NULL,
                               formula=~as.factor(Survey)+size+SeaState)
```

```
## Fitting half-normal key function
```

```
## AIC= 2237.818
```

```
## No survey area information supplied, only estimating detection function.
```

```
# hazard-rate models
models$hr <- ds(distdata, truncation=6000, adjustment=NULL, key="hr")
```

```
## Fitting hazard-rate key function
```

```
## Key only model: not constraining for monotonicity.
```

```
## AIC= 2247.594
```

```
## No survey area information supplied, only estimating detection function.
```

```
models$hr.ss <- ds(distdata, truncation=6000, adjustment=NULL, formula=~SeaState,
                   key="hr")
```

```
## Fitting hazard-rate key function

## AIC= 2240.354

## No survey area information supplied, only estimating detection function.
```

```
models$hr.size <- ds(distdata, truncation=6000, adjustment=NULL, formula=~size,
                     key="hr")
```

```
## Fitting hazard-rate key function

## AIC= 2249.427

## No survey area information supplied, only estimating detection function.
```

```
models$hr.ss.size <- ds(distdata, truncation=6000, adjustment=NULL,
                        formula=~size+SeaState, key="hr")
```

```
## Fitting hazard-rate key function

## AIC= 2242.175

## No survey area information supplied, only estimating detection function.
```

```
models$hr.survey <- ds(distdata, truncation=6000, adjustment=NULL,
                       formula=~as.factor(Survey), key="hr")
```

```
## Fitting hazard-rate key function

## AIC= 2248.871

## No survey area information supplied, only estimating detection function.
```

```
models$hr.survey.ss <- ds(distdata, truncation=6000, adjustment=NULL,
                          formula=~as.factor(Survey)+SeaState, key="hr")
```

```
## Fitting hazard-rate key function

## AIC= 2241.45

## No survey area information supplied, only estimating detection function.
```

```
models$hr.survey.size <- ds(distdata, truncation=6000, adjustment=NULL,
                            formula=~as.factor(Survey)+size, key="hr")
```

```
## Fitting hazard-rate key function

## AIC= 2250.146

## No survey area information supplied, only estimating detection function.
```

```
models$hr.survey.size.ss <- ds(distdata, truncation=6000, adjustment=NULL,
                               formula=~as.factor(Survey)+size+SeaState, key="hr")
```

```
## Fitting hazard-rate key function
## AIC= 2243.434
```

```
## No survey area information supplied, only estimating detection function.
```

Phew!

Model selection

Looking at the models individually can be a bit unwieldy – it's nicer to put that data into a table and sort the table by the relevant statistic.

The code below will make a results table with relevant statistics for model selection in it. Don't worry about how this code exactly works at the moment.

```
make_table <- function(models){

  # this function extracts the model data for a single model (row)
  extract_model_data <- function(model){
    c(summary(model)$ds$key,
      model$ddf$ds$aux$ddfobj$scale$formula,
      model$ddf$criterion,
      ddf.gof(model$ddf, qq=FALSE)$dsgof$CvM$p,
      summary(model)$ds$average.p,
      summary(model)$ds$average.p.se
    )
  }

  # applying that to all the models then putting it into a data.frame
  res <- as.data.frame(t(as.data.frame(lapply(models, extract_model_data))),
                        stringsAsFactors=FALSE)

  # making sure the correct columns are numeric
  res[,3] <- as.numeric(res[,3])
  res[,4] <- as.numeric(res[,4])
  res[,5] <- as.numeric(res[,5])
  res[,6] <- as.numeric(res[,6])

  # giving the columns names
  colnames(res) <- c("Key", "Formula", "AIC", "CvMises $p$-value",
                     "$\\hat{P_a}$", "se($\\hat{P_a}$)")

  # creating a new column for the AIC difference to the best model
  res[["$\\Delta$AIC"]] <- res$AIC - min(res$AIC, na.rm=TRUE)
  # ordering the model by AIC score
  res <- res[order(res$AIC),]

  # returning the data.frame
  return(res)
}
```

The `make_table()` function expects a `list` of models as it's input. We can do that with the two models that I fitted like so:

```
#models <- list()
#models$df_hn <- df_hn
#models$df_hr_ss_size <- df_hr_ss_size
```

(You can add the models you fitted above into this list.)

We already have everything in a list from the above, so we comment out the above.

Here is the resulting table from the code above, made using the `kable` function from `knitr`:

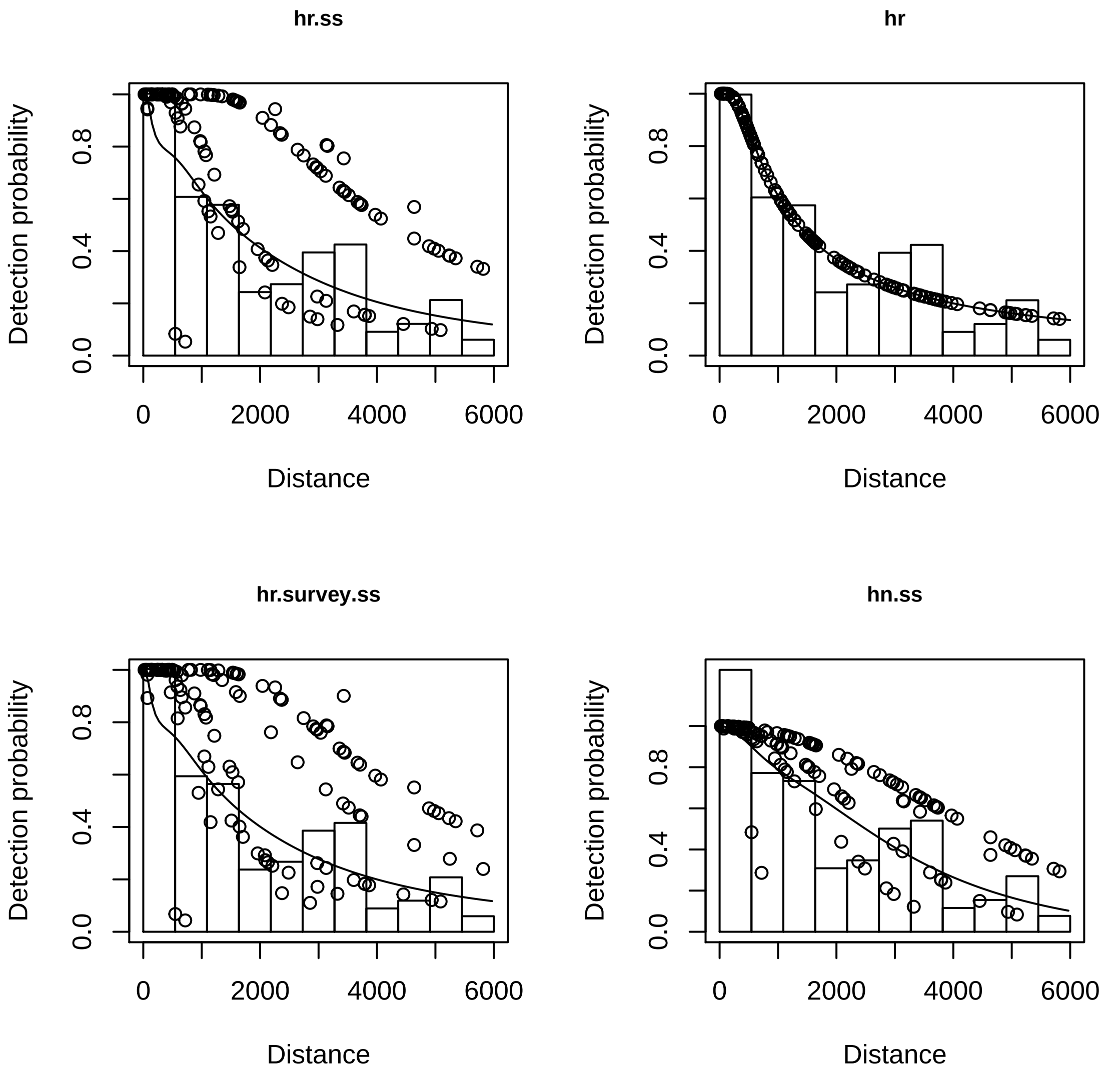
```
model_table <- make_table(models)
kable(model_table, digits=2)
```

	Key	Formula	AIC	CvMises p-value	$\hat{P}_a$	se(
hn.survey.ss	hn	~as.factor(Survey) + SeaState	2235.88	0.36	0.46	0
hn.ss	hn	~SeaState	2236.73	0.33	0.46	0
hn.survey.size.ss	hn	~as.factor(Survey) + size + SeaState	2237.82	0.35	0.46	0
hn.ss.size	hn	~size + SeaState	2238.41	0.34	0.46	0
hr.ss	hr	~SeaState	2240.35	0.91	0.36	0
hr.survey.ss	hr	~as.factor(Survey) + SeaState	2241.45	0.95	0.36	0
hr.ss.size	hr	~size + SeaState	2242.17	0.89	0.36	0
hr.survey.size.ss	hr	~as.factor(Survey) + size + SeaState	2243.43	0.94	0.36	0
hr	hr	~1	2247.59	0.90	0.36	0
hr.survey	hr	~as.factor(Survey)	2248.87	0.88	0.38	0
hr.size	hr	~size	2249.43	0.90	0.36	0
hr.survey.size	hr	~as.factor(Survey) + size	2250.15	0.86	0.37	0
hn	hn	~1	2252.06	0.07	0.55	0
hn.survey	hn	~as.factor(Survey)	2252.35	0.08	0.55	0
hn.size	hn	~size	2253.89	0.08	0.55	0
hn.survey.size	hn	~as.factor(Survey) + size	2254.34	0.08	0.55	0

The four best models are hn.survey.ss, hn.ss, hn.survey.size.ss, hn.ss.size

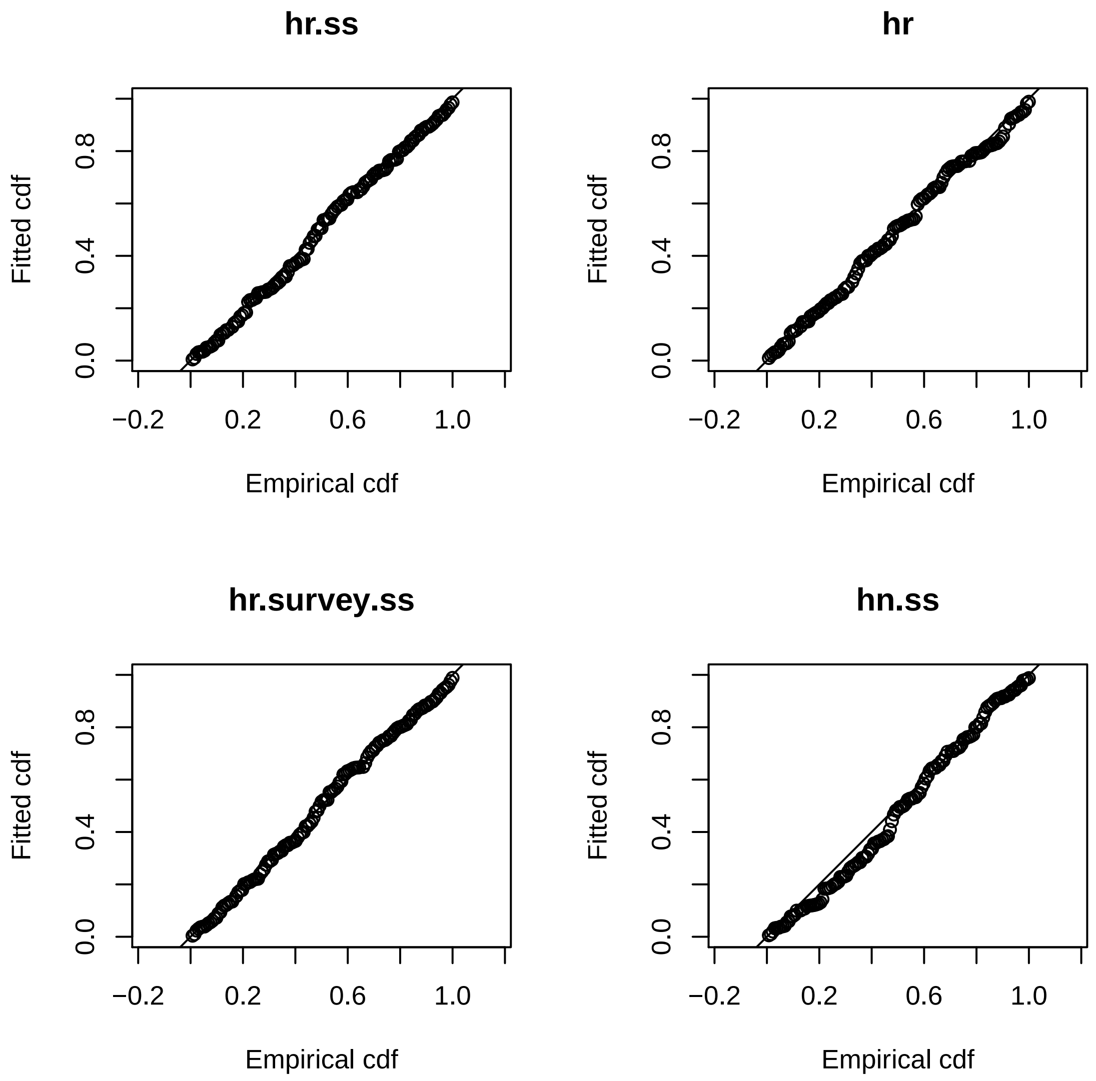
We can plot the best 4 models:

```
par(mfrow=c(2,2))
plot(models$hr.ss, main="hr.ss")
plot(models$hr, main="hr")
plot(models$hr.survey.ss, main="hr.survey.ss")
plot(models$hn.ss, main="hn.ss")
```



and produce the corresponding quantile-quantile plots and goodness of fit test results

```r
par(mfrow=c(2,2))
gof_ds(models$hr.ss, main="hr.ss")
gof_ds(models$hr, main="hr")
gof_ds(models$hr.survey.ss, main="hr.survey.ss")
gof_ds(models$hn.ss, main="hn.ss")
```



(that was a lot of output, we can get rid of that by setting `results="hide"` in the chunk above).

From that it looks like the “best” mode by AIC and by goodness of fit is a hazard-rate model with Beaufort sea state as a covariate. But note that there is a considerable spike in the distance data (see next comment).

A further note about model selection for the sperm whale data

Note that there is a considerable spike in our distance data. This may be down to observers guarding the trackline (spending too much time searching near zero distance). It’s therefore possible that the hazard-rate model is overfitting to this peak. So we’d like to investigate results from the half-normal model too and see what the effects are on the final spatial models.

Estimating abundance

Just for fun, let's estimate abundance from these models using a Horvtiz-Thompson-type estimator.

We know the Horvitz-Thompson estimator has the following form:

$$\hat{N} = \frac{A}{a} \sum_{i=1}^{n} \frac{s_i}{p_i}$$

we can calculate each part of this equation in R:

- `A` is the total area of the region we want to estimate abundance for. This was $A = 5.285e + 11m^2$.
- `a` is the total area we surveyed. We know that the total transect length was 9,498,474m and the truncation distance. Knowing that then $a = 2wL$ we can calculate a.
- s_i are the group sizes, they are stored in `df_hn$ddf$data$size`.
- p_i are the probabilities of detection, we can obtain them using `predict(df_hn$ddf)$fitted`.

We know that in general operations are vectorised in R, so calculating `c(1, 2, 3)/c(4, 5, 6)` will give `c(1/4, 2/5, 3/6)` so we can just divide the results of getting the s_i and p_i values and then use the `sum()` function to sum them up.

Try out estimating abundance using the formula below using both `df_hn` and your favourite model from above:

Just trying that out for `df_hn`:

```
N_hn <- (5.285e+11/(2*df_hn$ddf$meta.data$width*9498474)) *
          sum(df_hn$ddf$data$size/predict(df_hn$ddf)$fitted)
N_hn
```

```
## [1] 2015.82
```

Save model objects

Save your top few models in an `RData` file, so we can load them up later on. We'll also save the distance data we used to fit out models.

```
save(df_hn, df_hr_ss_size, models,
     distdata,
     file="df-models.RData")
```

You can check it worked by using the `load()` function to recover the models.