Advanced density surface models

Solution 5, Intermediate Distance Sampling workshop, CREEM, 2018

Here is a "solution" for practical 5. As with any data analysis, there is no correct answer, but this shows how I would approach this analysis. The analysis here is conditional on selecting a detection function in the previous exercises; I've chosen `df_hn` and `df_hr_ss_size` that we saved previously (see detection function practical).

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.

Load data and packages

```
library(Distance)
```

```
## Loading required package: mrds

## This is mrds 2.2.0
## Built: R 3.5.1; ; 2018-08-03 20:22:18 UTC; windows

##
## Attaching package: 'Distance'

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

```
library(dsm)
```

```
## Loading required package: mgcv

## Loading required package: nlme

## This is mgcv 1.8-24. For overview type 'help("mgcv-package")'.

## Loading required package: numDeriv

## This is dsm 2.2.16
## Built: R 3.4.2; ; 2017-11-02 15:05:30 UTC; windows
```

```
library(ggplot2)
library(knitr)
library(plyr)
library(reshape2)
```

Loading the data processed from GIS and the fitted detection function objects from the previous exercises:

```
load("sperm-data.RData")
load("df-models.RData")
```

Exploratory analysis

We can do some exploratory analysis by aggregating the counts to each cell and plotting what's going on.

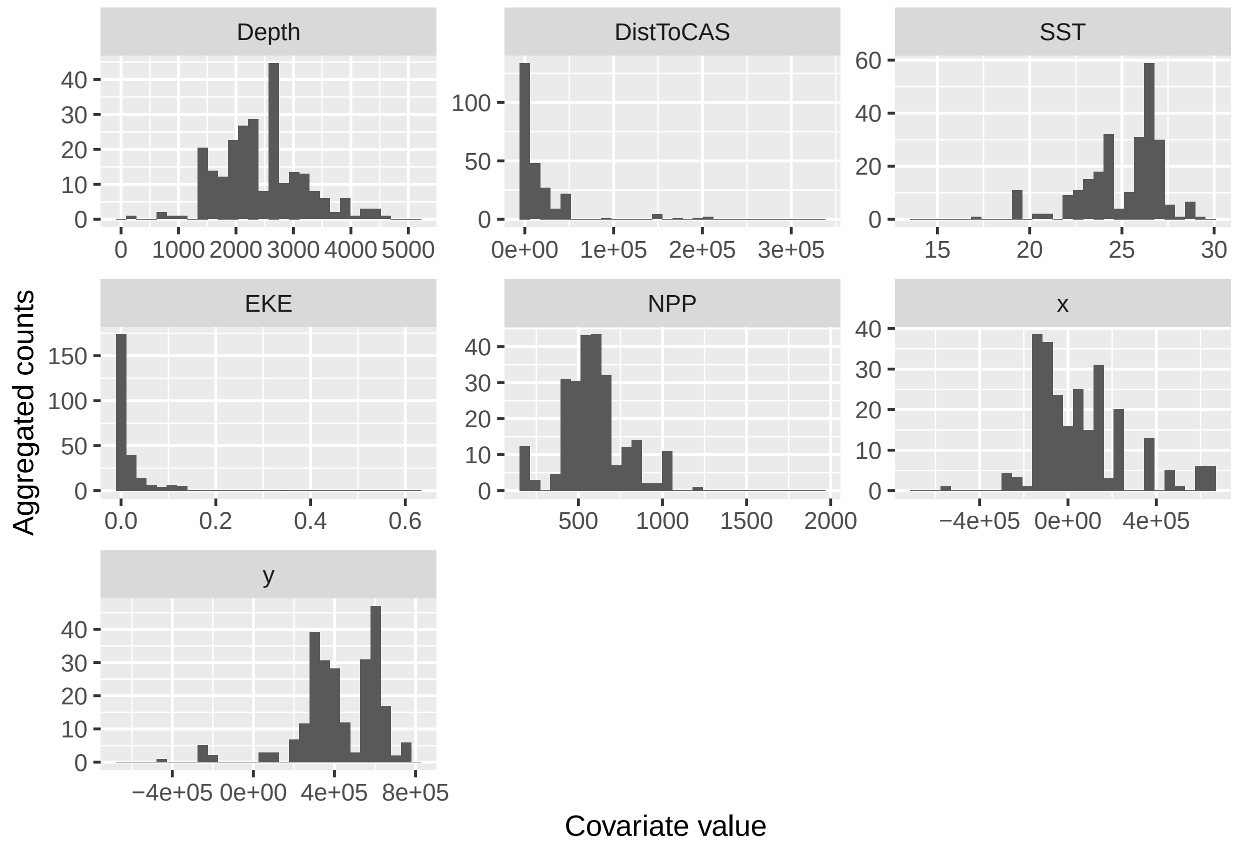
Don't worry about understanding what this code is doing at the moment.

```r
# join the observations onto the segments
join_dat <- join(segs, obs, by="Sample.Label", type="full")
# sum up the observations per segment
n <- ddply(join_dat, .(Sample.Label), summarise, n=sum(size), .drop = FALSE)
# sort the segments by their labsl
segs_eda <- segs[sort(segs$Sample.Label),]
# make a new column for the counts
segs_eda$n <- n$n

# remove the columns we don't need,
segs_eda$CentreTime <- NULL
segs_eda$POINT_X <- NULL
segs_eda$POINT_Y <- NULL
segs_eda$segment.area <- NULL
segs_eda$off.set <- NULL
segs_eda$CenterTime <- NULL
segs_eda$Effort <- NULL
segs_eda$Length <- NULL
segs_eda$SegmentID <- NULL
segs_eda$coords.x1 <- NULL
segs_eda$coords.x2 <- NULL

# "melt" the data so we have four columns:
#   Sample.Label, n (number of observations),
#   variable (which variable), value (its value)
segs_eda <- melt(segs_eda, id.vars=c("Sample.Label", "n"))
# try head(segs_eda)
```

Finally, we can plot histograms of counts for different values of the covariates:

```r
p <- ggplot(segs_eda) +
       geom_histogram(aes(value, weight=n)) +
       facet_wrap(~variable, scale="free") +
       xlab("Covariate value") +
       ylab("Aggregated counts")
print(p)
```

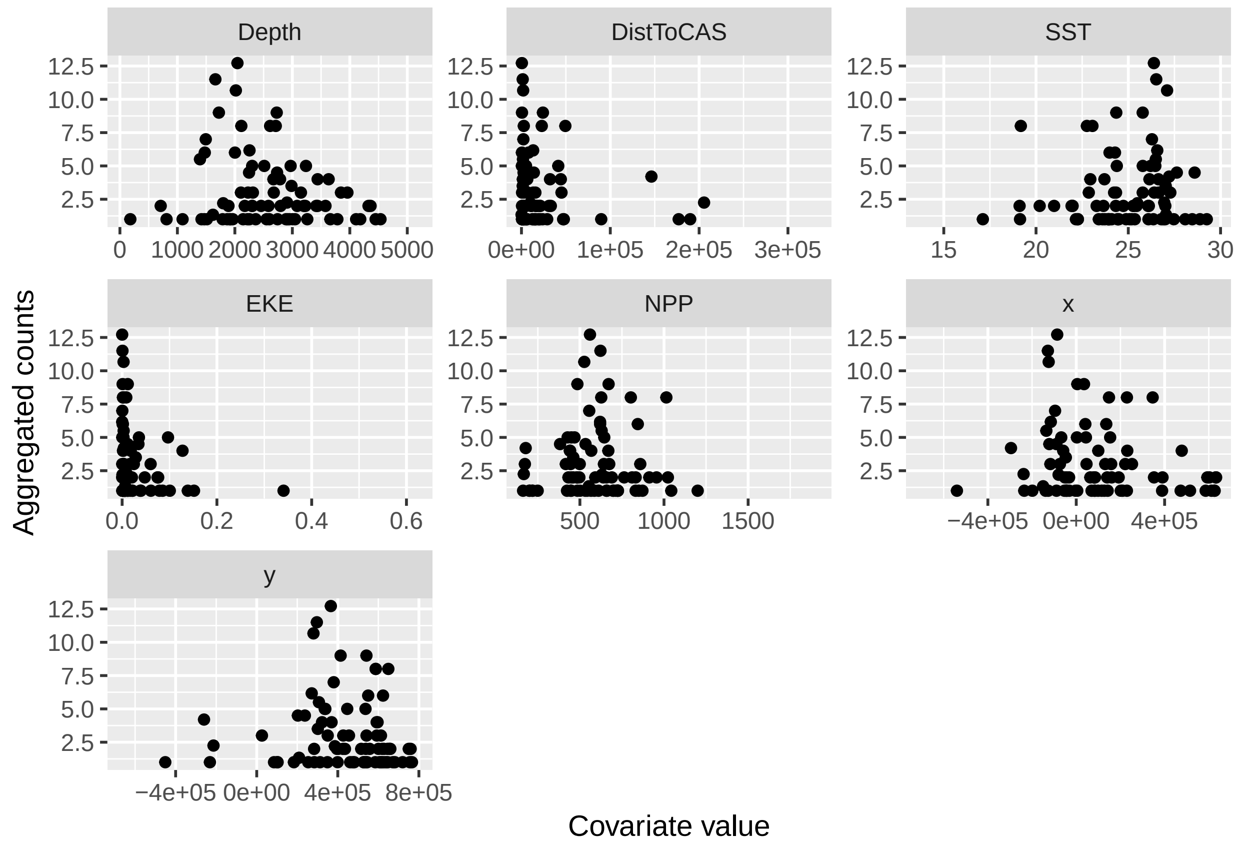
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can also just plot the counts against the covariates, note the high number of zeros (but still some interesting patterns):

```
p <- ggplot(segs_eda) +
       geom_point(aes(value, n)) +
       facet_wrap(~variable, scale="free") +
       xlab("Covariate value") +
       ylab("Aggregated counts")
print(p)
```

```
## Warning: Removed 6076 rows containing missing values (geom_point).
```



These plots give a very rough idea of the relationships we can expect in the model. Notably these plots don't take into account interactions between the variables and potential correlations between the terms, as well as detectability.

Pre-model fitting

As we did in the previous exercise we must remove the observations from the spatial data that we excluded when we fitted the detection function – those observations at distances greater than the truncation.

```
obs <- obs[obs$distance <= df_hn$ddf$meta.data$width, ]
```

Here we've used the value of the truncation stored in the detection function object, but we could also use the numeric value (which we can also find by checking the model's `summary()`).

Again note that if you want to fit DSMs using detection functions with different truncation distances, then you'll need to reload the `sperm-data.RData` and do the truncation again for that detection function.

Our new friend +

We can build a really big model using `+` to include all the terms that we want in the model. We can check what's available to us by using `head()` to look at the segment table:

```
head(segs)
```

```
##              CenterTime SegmentID   Length  POINT_X  POINT_Y      Depth
## 1 2004/06/24 07:27:04         1 10288.91 214544.0 689074.3  118.5027
## 2 2004/06/24 08:08:04         2 10288.91 222654.3 682781.0  119.4853
## 3 2004/06/24 09:03:18         3 10288.91 230279.9 675473.3  177.2779
## 4 2004/06/24 09:51:27         4 10288.91 239328.9 666646.3  527.9562
## 5 2004/06/24 10:25:39         5 10288.91 246686.5 659459.2  602.6378
## 6 2004/06/24 11:00:22         6 10288.91 254307.0 652547.2 1094.4402
##    DistToCAS      SST          EKE      NPP  coords.x1 coords.x2        x
## 1 14468.1533 15.54390 0.0014442616 1908.129  214544.0  689074.3 214544.0
## 2 10262.9648 15.88358 0.0014198086 1889.540  222654.3  682781.0 222654.3
## 3  6900.9829 16.21920 0.0011704842 1842.057  230279.9  675473.3 230279.9
## 4  1055.4124 16.45468 0.0004101589 1823.942  239328.9  666646.3 239328.9
## 5  1112.6293 16.62554 0.0002553244 1721.949  246686.5  659459.2 246686.5
## 6   707.5795 16.83725 0.0006556266 1400.281  254307.0  652547.2 254307.0
##          y   Effort Sample.Label
## 1 689074.3 10288.91            1
## 2 682781.0 10288.91            2
## 3 675473.3 10288.91            3
## 4 666646.3 10288.91            4
## 5 659459.2 10288.91            5
## 6 652547.2 10288.91            6
```

We can then fit a model with the available covariates in it, each as an `s()` term.

```
dsm_nb_xy_ms <- dsm(count~s(x,y, bs="ts") +
                       s(Depth, bs="ts") +
                       s(DistToCAS, bs="ts") +
                       s(SST, bs="ts") +
                       s(EKE, bs="ts") +
                       s(NPP, bs="ts"),
                 df_hn, segs, obs,
                 family=nb())
summary(dsm_nb_xy_ms)
```

```
##
## Family: Negative Binomial(0.11)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + s(DistToCAS,
##     bs = "ts") + s(SST, bs = "ts") + s(EKE, bs = "ts") + s(NPP,
##     bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.7278     0.2219  -93.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                    edf Ref.df Chi.sq  p-value
## s(x,y)       1.8069778     29 19.155 2.24e-05 ***
## s(Depth)     3.1224156      9 47.384 3.60e-12 ***
## s(DistToCAS) 0.0001369      9  0.000    0.741
## s(SST)       0.0001613      9  0.000    0.517
## s(EKE)       0.8126762      9  5.185    0.011 *
## s(NPP)       0.0001217      9  0.000    0.813
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.093   Deviance explained = 39.1%
## -REML = 276.88  Scale est. = 1         n = 949
```

Notes:

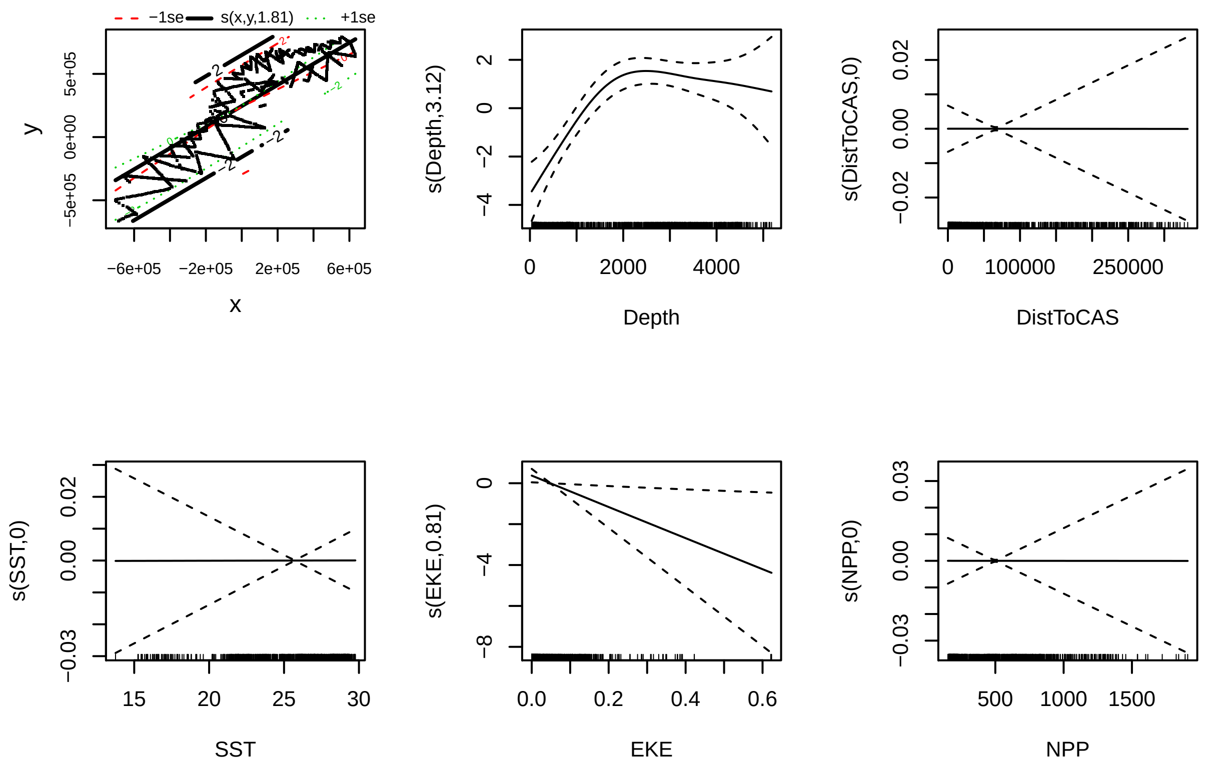
1. We're using `bs="ts"` to use the shrinkage thin plate regression spline. More technical detail on these smooths can be found on their manual page `?smooth.construct.ts.smooth.spec`.
2. We've not specified basis complexity (`k`) at the moment. Note that if you want to specify the same complexity for multiple terms, it's often easier to make a variable that can then be given as `k` (for example, setting `k1<-15` and then setting `k=k1` in the required `s()` terms).

Looking at the `summary()` here we can see that many of the terms have been shrunk out of the model. We could simply use this model to make predictions at this point since those shrunken terms will have little effect on the results (since their EDFs are so small), but it makes sense to remove them (if only because then predictions will be faster, and we might avoid building many prediction grids for those covariates – more on that later).

Plot

Let's plot the smooths from this model:

```
plot(dsm_nb_xy_ms, pages=1, scale=0)
```

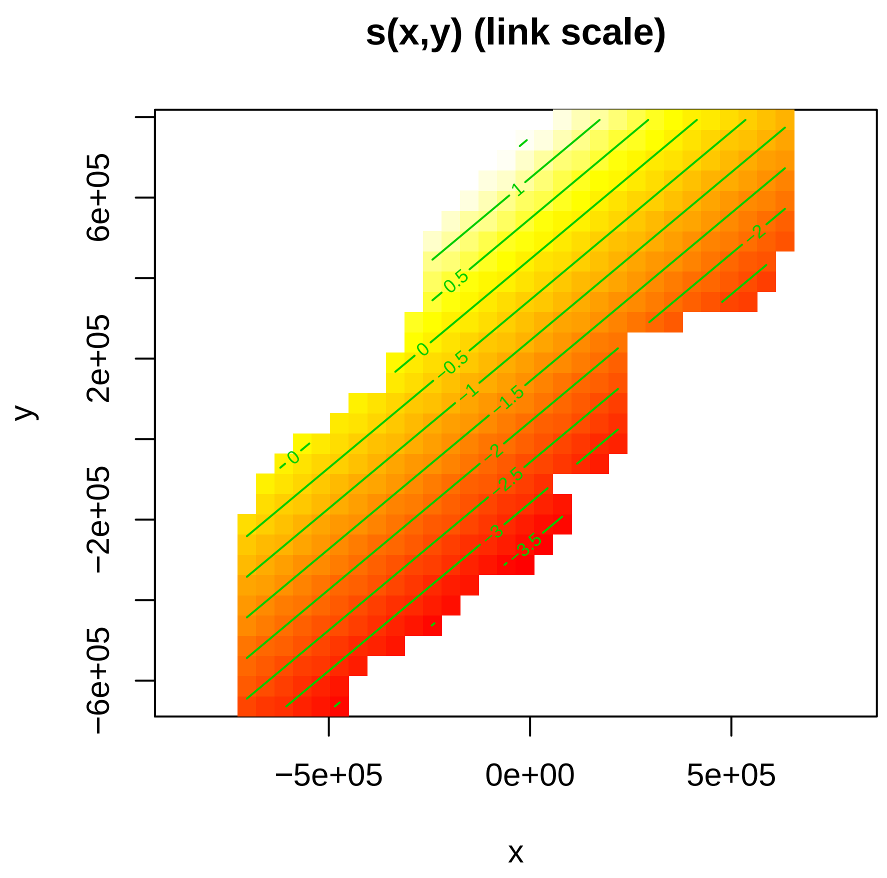


Notes:

1. Setting `shade=TRUE` gives prettier confidence bands.
2. As with `vis.gam()` the response is on the link scale.
3. `scale=0` puts each plot on a different y-axis scale, making it easier to see the effects. Setting `scale=-1` will put the plots on a common y-axis scale

We can also plot the bivariate smooth of `x` and `y` as we did before, using `vis.gam()`:

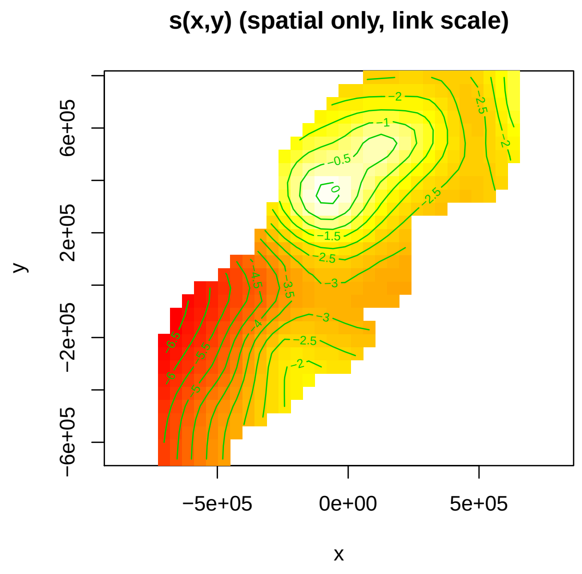
```
vis.gam(dsm_nb_xy_ms, view=c("x","y"), plot.type="contour", too.far=0.1, main="s(x,y) (link scale)", asp
```

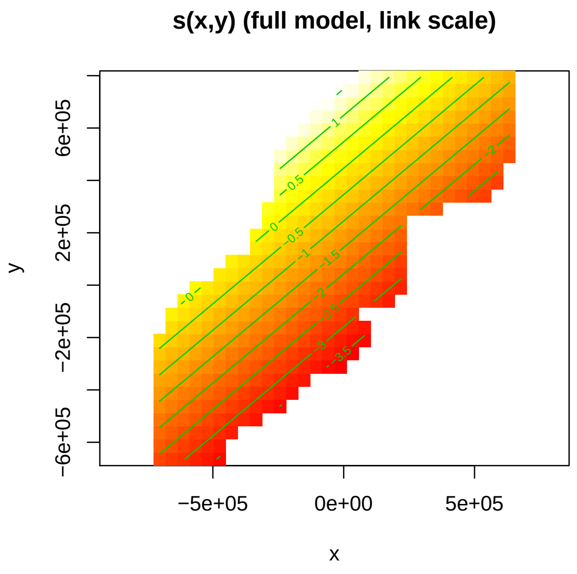


Compare this plot to the one that was generated in the previous practical when only `x` and `y` were included in the model.

We can load up that saved model and compare the bivariate smooth of location:

```
load("dsms-xy.RData")
par(mfrow=c(1,2))
vis.gam(dsm_nb_xy, view=c("x","y"), plot.type="contour", too.far=0.1, main="s(x,y) (spatial only, link s
vis.gam(dsm_nb_xy_ms, view=c("x","y"), plot.type="contour", too.far=0.1, main="s(x,y) (full model, link
```



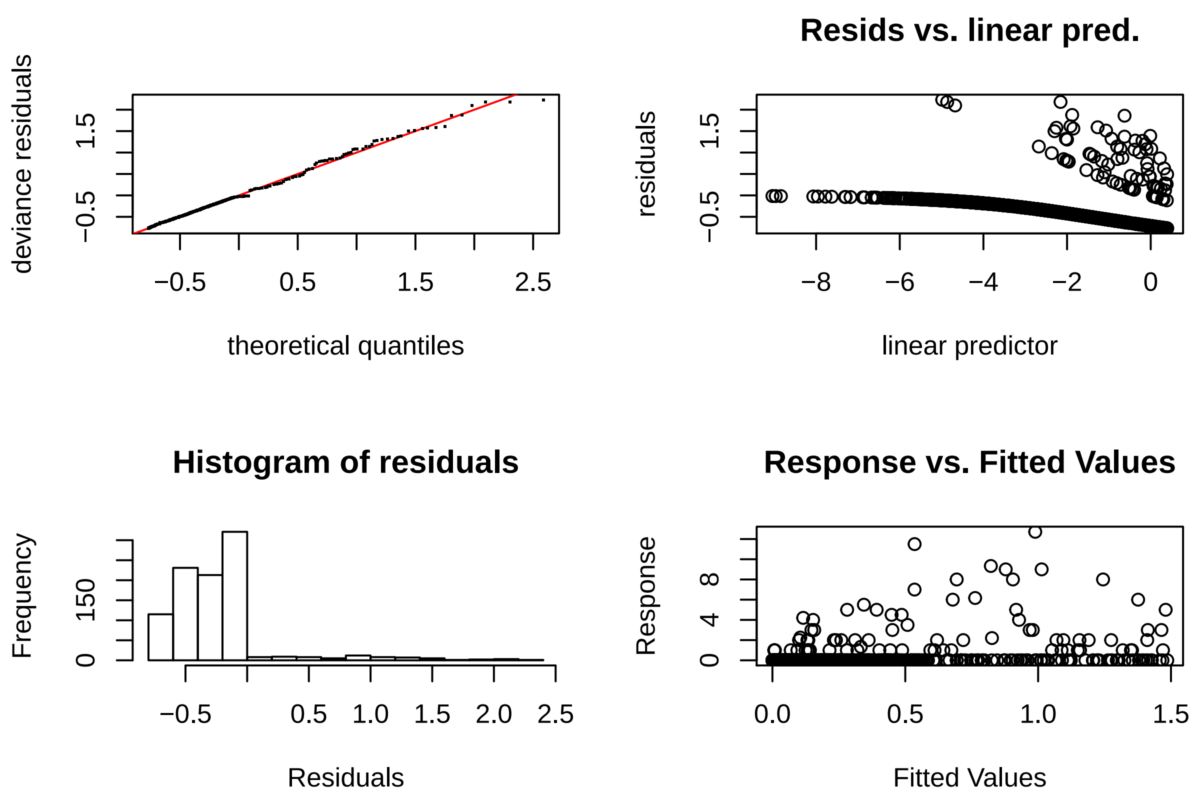


Once additional covariates are included in the model the spatial term is much simpler, as those terms are explaining variation that we'd previously only been able to describe via spatial variation.

Check

As before, we can use `gam.check()` and `rqgam.check()` to look at the residual check plots for this model. Do this in the below gaps and comment on the resulting plots and diagnostics.

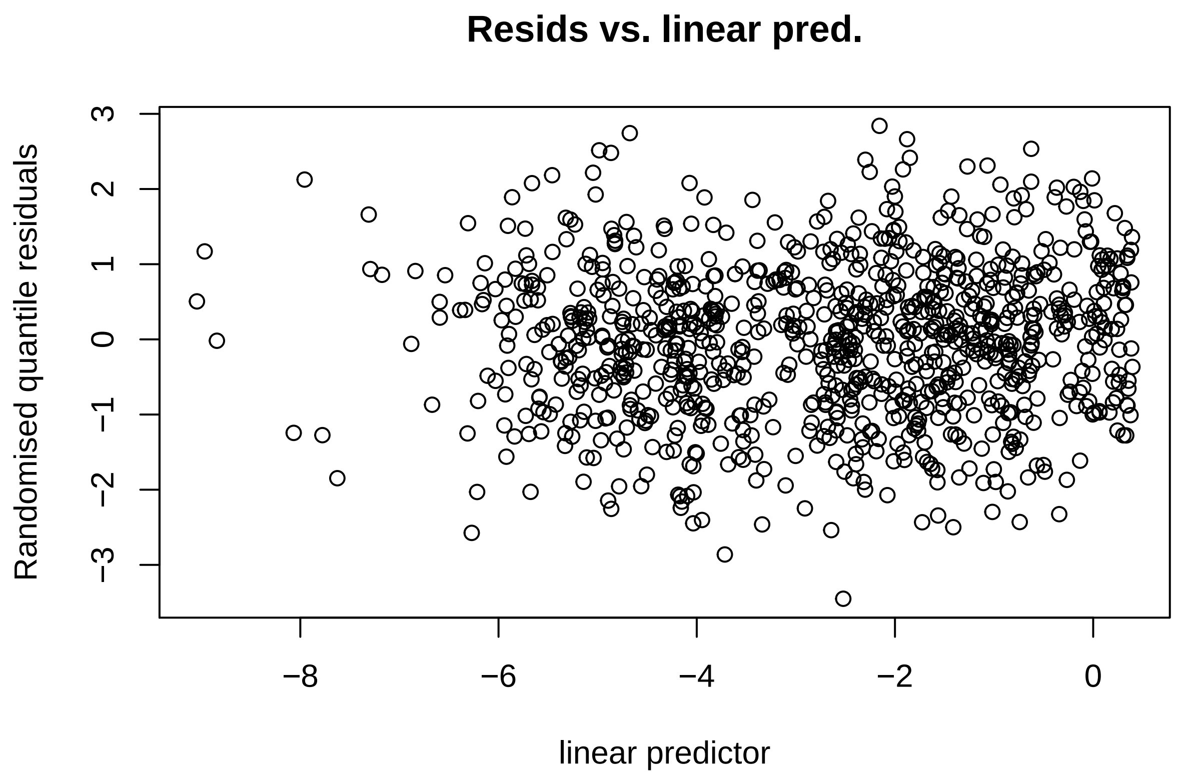
```r
gam.check(dsm_nb_xy_ms)
```



```
##
## Method: REML   Optimizer: outer newton
## full convergence after 10 iterations.
## Gradient range [-6.084924e-05,5.259734e-05]
## (score 276.8754 & scale 1).
## Hessian positive definite, eigenvalue range [3.814945e-05,21.78745].
## Model rank =  75 / 75
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                    k'      edf k-index p-value
## s(x,y)       2.90e+01 1.81e+00    0.55  <2e-16 ***
## s(Depth)     9.00e+00 3.12e+00    0.68   0.035 *
## s(DistToCAS) 9.00e+00 1.37e-04    0.66   0.020 *
## s(SST)       9.00e+00 1.61e-04    0.71   0.240
## s(EKE)       9.00e+00 8.13e-01    0.74   0.725
## s(NPP)       9.00e+00 1.22e-04    0.68   0.020 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The left side of the `gam.check()` plot shows the Q-Q plot looks okay, the histogram less so.

```
rqgam.check(dsm_nb_xy_ms)
```



`rqgam.check()` shows we don't need to be concerned with non-constant variance.

You might decide from the diagnostics that you need to increase `k` for some of the terms in the model. Do this and re-run the above code to ensure that the smooths are flexible enough. The `?choose.k` manual page can offer some guidance. Generally if teh EDF is close to the value of `k` you supplied, it's worth doubling `k` and refitting to see what happens. You can always switch back to the smaller `k` if there is little difference.

It doesn't seem like `k` needs to be increased for any of the terms.

Select terms

As was covered in the lectures, we can select terms by (approximate) *p*-values and by looking for terms that have EDFs significantly less than 1 (those which have been shrunk).

Decide on a significance level that you'll use to discard terms in the model. Remove the terms that are non-significant at this level and re-run the above checks, summaries and plots to see what happens. It's helpful to make notes to yourself as you go

It's easiest to either comment out the terms that are to be removed (using `#`) or by copying the code chunk above and pasting it below.

Having removed a smooth and reviewed your model, you may decide you wish to remove another. Follow the process again, removing a term, looking at plots and diagnostics.

Looking again at the model summary:

```r
summary(dsm_nb_xy_ms)
```

```
##
```

```
## Family: Negative Binomial(0.11)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + s(DistToCAS,
##     bs = "ts") + s(SST, bs = "ts") + s(EKE, bs = "ts") + s(NPP,
##     bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.7278     0.2219  -93.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                    edf Ref.df Chi.sq  p-value
## s(x,y)       1.8069778     29 19.155 2.24e-05 ***
## s(Depth)     3.1224156      9 47.384 3.60e-12 ***
## s(DistToCAS) 0.0001369      9  0.000    0.741
## s(SST)       0.0001613      9  0.000    0.517
## s(EKE)       0.8126762      9  5.185    0.011 *
## s(NPP)       0.0001217      9  0.000    0.813
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.093   Deviance explained = 39.1%
## -REML = 276.88  Scale est. = 1         n = 949
```

Terms DistToCAS, SST and NPP seem to have been shrunk to be very small. We can start by removing DistToCAS from the model (as it has been shrunk the most):

```
dsm_nb_xy_ms <- dsm(count~s(x,y, bs="ts") +
                       s(Depth, bs="ts") +
                       s(SST, bs="ts") +
                       s(EKE, bs="ts") +
                       s(NPP, bs="ts"),
                  df_hn, segs, obs,
                  family=nb())
```

Looking at the summary of our new model:

```
summary(dsm_nb_xy_ms)
```

```
##
## Family: Negative Binomial(0.11)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + s(SST, bs = "ts") +
##     s(EKE, bs = "ts") + s(NPP, bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.7278     0.2219  -93.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Approximate significance of smooth terms:
##                 edf Ref.df Chi.sq  p-value
## s(x,y)   1.8069822     29 19.156 2.24e-05 ***
## s(Depth) 3.1224146      9 47.384 3.60e-12 ***
## s(SST)   0.0001605      9  0.000    0.517
## s(EKE)   0.8126758      9  5.185    0.011 *
## s(NPP)   0.0002167      9  0.000    0.813
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.093   Deviance explained = 39.1%
## -REML = 276.88  Scale est. = 1         n = 949
```

Let's now remove NPP:

```
dsm_nb_xy_ms <- dsm(count~s(x,y, bs="ts") +
                       s(Depth, bs="ts") +
                       s(SST, bs="ts") +
                       s(EKE, bs="ts"),
                 df_hn, segs, obs,
                 family=nb())
summary(dsm_nb_xy_ms)
```

```
##
## Family: Negative Binomial(0.11)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + s(SST, bs = "ts") +
##     s(EKE, bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.7278     0.2219  -93.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df Chi.sq  p-value
## s(x,y)   1.806981     29 19.158 2.23e-05 ***
## s(Depth) 3.122411      9 47.384 3.60e-12 ***
## s(SST)   0.000449      9  0.000    0.517
## s(EKE)   0.812673      9  5.185    0.011 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.093   Deviance explained = 39.1%
## -REML = 276.88  Scale est. = 1         n = 949
```

And now removing SST:

```
dsm_nb_xy_ms <- dsm(count~s(x,y, bs="ts") +
                       s(Depth, bs="ts") +
                       s(EKE, bs="ts"),
                 df_hn, segs, obs,
                 family=nb())
summary(dsm_nb_xy_ms)
```

```
##
## Family: Negative Binomial(0.11)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + s(EKE, bs = "ts") +
##     offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.7278     0.2219  -93.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df Chi.sq  p-value
## s(x,y)   1.8070     29 19.161 2.24e-05 ***
## s(Depth) 3.1224      9 47.387 3.60e-12 ***
## s(EKE)   0.8127      9  5.185    0.011 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.093   Deviance explained = 39.1%
## -REML = 276.88  Scale est. = 1         n = 949
```

At this point we could leave `EKE` in the model, but I've decided that my cutoff for removing terms is that they are `***` significant, so I'll remove that too:

```
dsm_nb_xy_ms <- dsm(count~s(x,y, bs="ts") +
                       s(Depth, bs="ts"),
                 df_hn, segs, obs,
                 family=nb())
summary(dsm_nb_xy_ms)
```

```
##
## Family: Negative Binomial(0.105)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.6341     0.2178  -94.72   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df Chi.sq  p-value
## s(x,y)   1.794     29  21.18 7.09e-06 ***
## s(Depth) 3.316      9  46.85 7.27e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0909   Deviance explained = 37.4%
## -REML = 278.16  Scale est. = 1         n = 949
```

Again note no major differences in the EDFs of the model. This model is also appealing as it contains only static covariates (we'll talk about this more in the lectures).

Compare response distributions

Use the `gam.check()` to compare quantile-quantile plots between negative binomial and Tweedie distributions for the response.

First we need to perform model selection again, when Tweedie is used as the response distribution:

```
dsm_tw_xy_ms <- dsm(count~s(x,y, bs="ts") +
                          s(Depth, bs="ts") +
                          s(DistToCAS, bs="ts") +
                          s(SST, bs="ts") +
                          s(EKE, bs="ts") +
                          s(NPP, bs="ts"),
                    df_hn, segs, obs,
                    family=tw())
summary(dsm_tw_xy_ms)
```

```
##
## Family: Tweedie(p=1.281)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + s(DistToCAS,
##     bs = "ts") + s(SST, bs = "ts") + s(EKE, bs = "ts") + s(NPP,
##     bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.6217     0.2254  -91.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                    edf Ref.df     F  p-value
## s(x,y)       1.851e+00     29 0.719 4.11e-06 ***
## s(Depth)     3.335e+00      9 5.089 2.72e-11 ***
## s(DistToCAS) 1.063e-04      9 0.000   0.5213
## s(SST)       1.057e-01      9 0.016   0.2294
## s(EKE)       7.573e-01      9 0.481   0.0158 *
## s(NPP)       9.799e-05      9 0.000   0.7570
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.105   Deviance explained = 34.7%
## -REML = 278.73  Scale est. = 4.6302    n = 949
```

Again it doesn't seem like we need to increase `k` from the default values for any of the terms in the model, and again it looks like we should remove `DistToCAS` first:

```r
dsm_tw_xy_ms <- dsm(count~s(x,y, bs="ts") +
                          s(Depth, bs="ts") +
                          s(SST, bs="ts") +
                          s(EKE, bs="ts") +
                          s(NPP, bs="ts"),
                    df_hn, segs, obs,
                    family=tw())
summary(dsm_tw_xy_ms)
```

```
##
## Family: Tweedie(p=1.281)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + s(SST, bs = "ts") +
##     s(EKE, bs = "ts") + s(NPP, bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.6217     0.2254  -91.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                 edf Ref.df     F  p-value
## s(x,y)    1.8510977     29 0.719 4.14e-06 ***
## s(Depth)  3.3347881      9 5.089 2.72e-11 ***
## s(SST)    0.1045061      9 0.016   0.2295
## s(EKE)    0.7572623      9 0.481   0.0158 *
## s(NPP)    0.0001823      9 0.000   0.7566
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.105   Deviance explained = 34.7%
## -REML = 278.73  Scale est. = 4.6302    n = 949
```

Now we see a bit of a change from the negative binomial model. Rather than having the EDFs stay roughly the same, we see instead that the p-value for `s(x,y)` has increased and `NPP` has become significant. Let's now remove `SST` and see what happens:

```r
dsm_tw_xy_ms <- dsm(count~s(x,y, bs="ts") +
                          s(Depth, bs="ts") +
                          s(EKE, bs="ts") +
                          s(NPP, bs="ts"),
                    df_hn, segs, obs,
                    family=tw())
summary(dsm_tw_xy_ms)
```

```
##
## Family: Tweedie(p=1.281)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + s(EKE, bs = "ts") +
##     s(NPP, bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.6211     0.2254   -91.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                 edf Ref.df     F  p-value
## s(x,y)    1.8543737     29 0.721 1.15e-05 ***
## s(Depth)  3.3371879      9 5.156 2.35e-11 ***
## s(EKE)    0.7542728      9 0.473   0.0168 *
## s(NPP)    0.0003163      9 0.000   0.7166
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.103   Deviance explained = 34.7%
## -REML = 278.73  Scale est. = 4.6369    n = 949
```

Removing SST has moved us back to roughly the smooth terms EDF/p -values we has previously. Let's remove NPP. . .

```
dsm_tw_xy_ms <- dsm(count~s(x,y, bs="ts") +
                          s(Depth, bs="ts") +
                          s(EKE, bs="ts"),
                    df_hn, segs, obs,
                    family=tw())
summary(dsm_tw_xy_ms)
```

```
##
## Family: Tweedie(p=1.281)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + s(EKE, bs = "ts") +
##     offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.6211     0.2254   -91.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df     F  p-value
## s(x,y)    1.8544     29 0.721 1.15e-05 ***
## s(Depth)  3.3372      9 5.157 2.35e-11 ***
## s(EKE)    0.7543      9 0.473   0.0168 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.103   Deviance explained = 34.7%
## -REML = 278.73  Scale est. = 4.6369    n = 949
```

Again, we could leave EKE in the model but let's remove it:

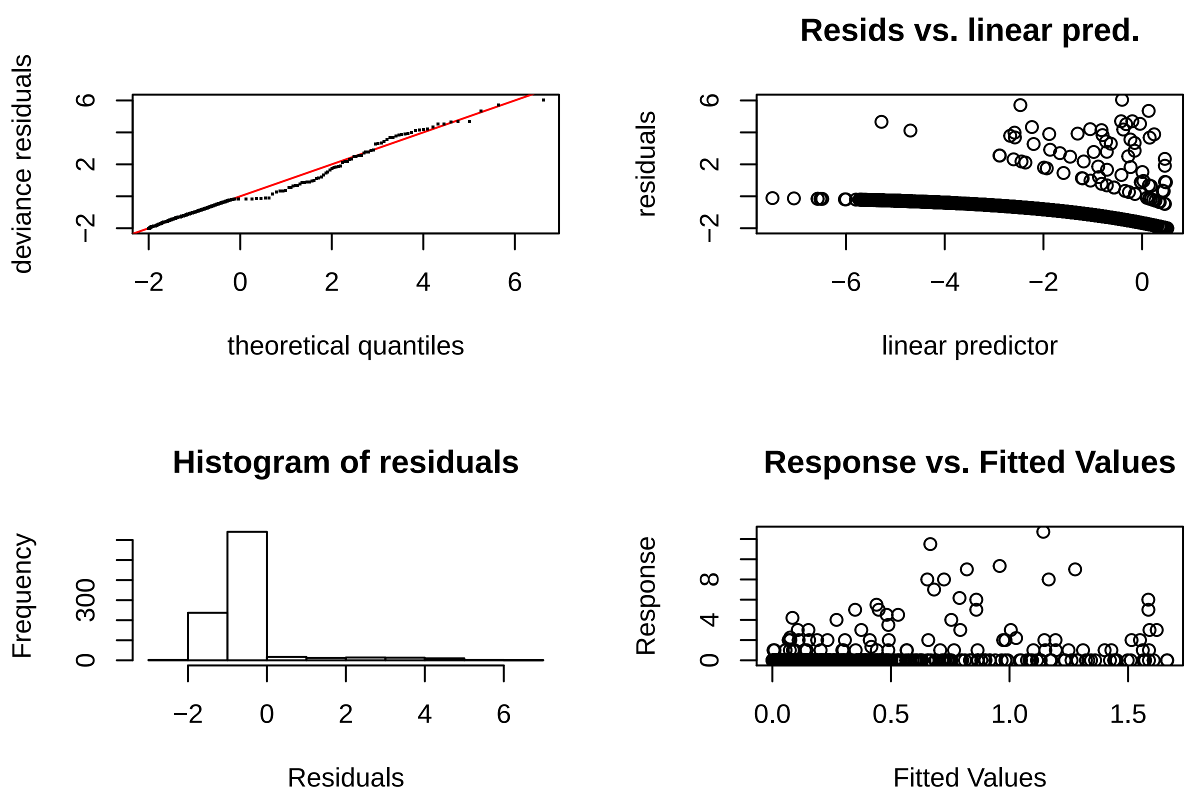
```r
dsm_tw_xy_ms <- dsm(count~s(x,y, bs="ts") +
                          s(Depth, bs="ts"),
                    df_hn, segs, obs,
                    family=tw())
summary(dsm_tw_xy_ms)
```

```
##
## Family: Tweedie(p=1.283)
## Link function: log
##
## Formula:
## count ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.5571     0.2216  -92.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df     F  p-value
## s(x,y)   1.869     29 0.877 1.22e-06 ***
## s(Depth) 3.434      9 5.226 2.23e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.101   Deviance explained = 33.4%
## -REML = 279.76  Scale est. = 4.7061    n = 949
```

We're now left with the same model structure as `dsm_nb_xy_ms`, but by rather a different path. This demonstrates some potential instabilities in the model and the effect of using a different response distribution (and different mean-variance relationship).

Let's look at check plots for this model

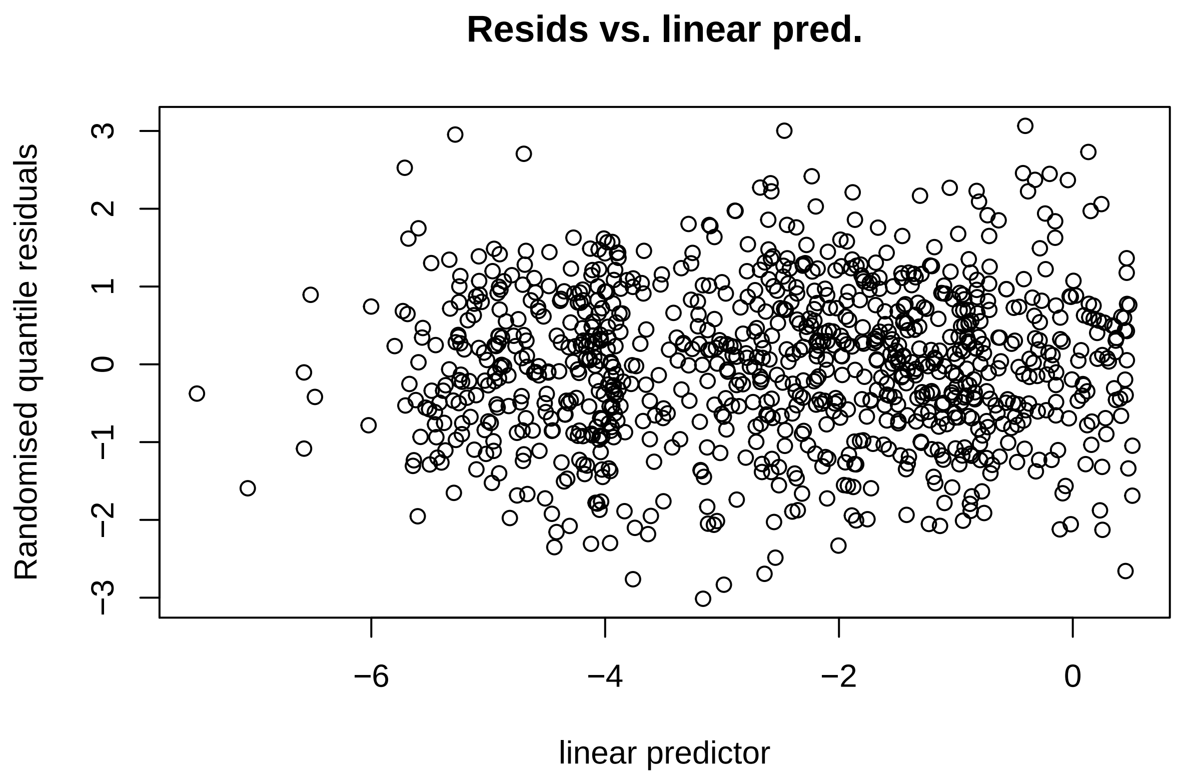
```r
gam.check(dsm_tw_xy_ms)
```



```
##
## Method: REML   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [-3.067794e-05,4.880858e-06]
## (score 279.7574 & scale 4.706123).
## Hessian positive definite, eigenvalue range [0.7122741,215.9676].
## Model rank =  39 / 39
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##              k'   edf k-index p-value
## s(x,y)    29.00  1.87    0.61  <2e-16 ***
## s(Depth)   9.00  3.43    0.80    0.41
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

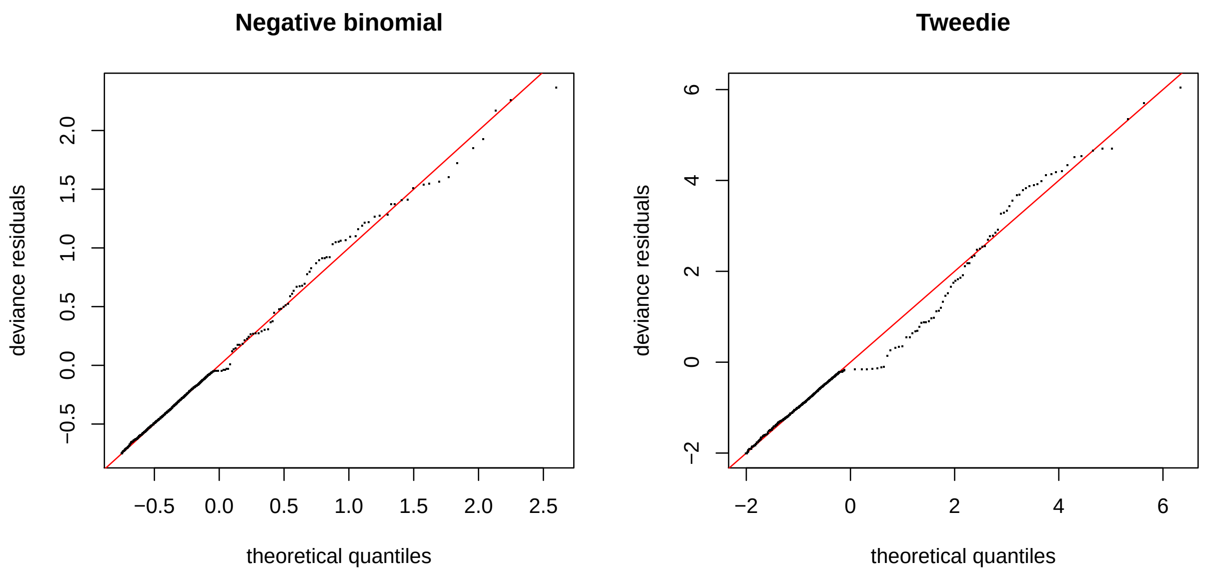
```
rqgam.check(dsm_tw_xy_ms)
```

```
## Loading required namespace: tweedie
```



These plots look okay and don't show any major issues. Let's compare the Q-Q plots for the negative binomial and Tweedie models side-by-side:

```r
par(mfrow=c(1,2))
qq.gam(dsm_nb_xy_ms, main="Negative binomial")
qq.gam(dsm_tw_xy_ms, main="Tweedie")
```



It looks like the negative binomial model has a better Q-Q plot here, so we're inclined to pick that model.

Estimated abundance as a response

Again, we've only looked at models with `count` as the response. Try using a detection function with covariates and the `abundance.est` response in the chunk below:

Let's test out both negative binomial and Tweedie distributions with estimated abundance as the response.

Starting with negative binomial. To save space I've commented the terms out as I've gone and annotated them with numbers to show the sequence.

```r
dsm_nb_xy_ms_ae <- dsm(abundance.est~s(x,y, bs="ts") +
                                     s(Depth, bs="ts"),# +
                                     # s(DistToCAS, bs="ts") + # 3
                                     #s(SST, bs="ts") + # 1
                                     #s(EKE, bs="ts"), #+ # 4
                                     #s(NPP, bs="ts"), # 2
                       df_hr_ss_size, segs, obs,
                       family=nb())
summary(dsm_nb_xy_ms_ae)
```

```
##
## Family: Negative Binomial(0.045)
## Link function: log
##
## Formula:
## abundance.est ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.2445     0.2044  -99.05   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df Chi.sq  p-value
## s(x,y)   1.188     29  14.22 7.19e-05 ***
## s(Depth) 3.319      9  57.02 8.61e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0555   Deviance explained = 31.4%
## -REML = 364.27  Scale est. = 1         n = 949
```

Again the model structure is the same at the end of the selection, but it takes a different route.

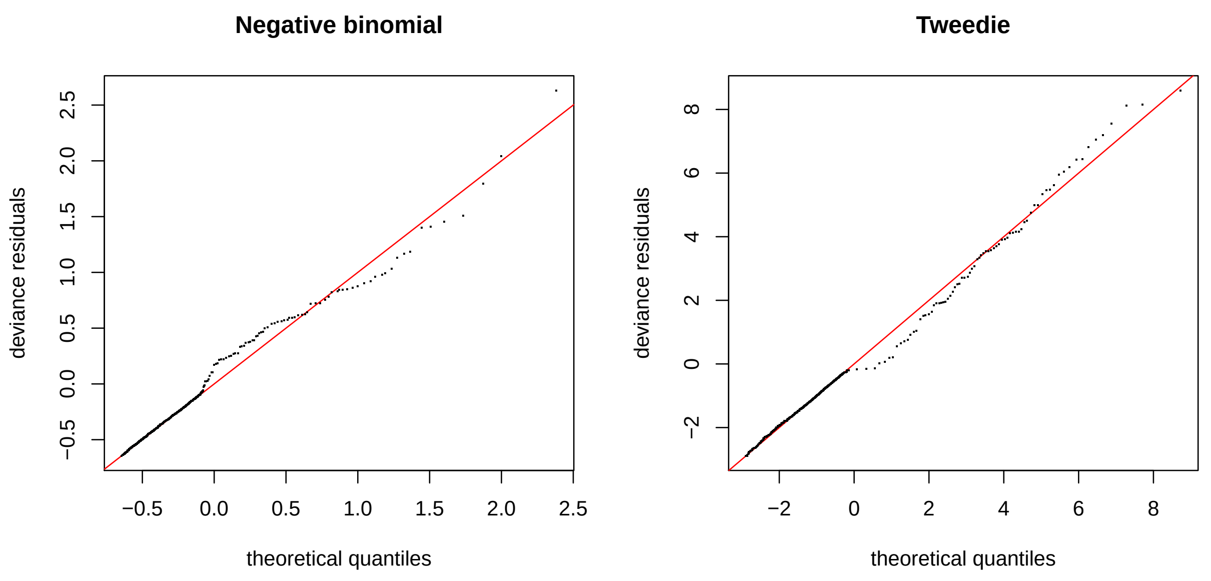
```r
dsm_tw_xy_ms_ae <- dsm(abundance.est~s(x,y, bs="ts") +
                                     s(Depth, bs="ts"), # +
                                     #s(DistToCAS, bs="ts") + # 2
                                     #s(SST, bs="ts") + # 3
                                     #s(EKE, bs="ts"),# +
                                     #s(NPP, bs="ts"), # 1
                       df_hr_ss_size, segs, obs,
                       family=tw())
summary(dsm_tw_xy_ms_ae)
```

```
##
## Family: Tweedie(p=1.232)
## Link function: log
##
## Formula:
## abundance.est ~ s(x, y, bs = "ts") + s(Depth, bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.1420     0.2245   -89.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df     F  p-value
## s(x,y)   1.877     29 0.896 9.58e-07 ***
## s(Depth) 3.501      9 5.056 5.94e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.112   Deviance explained = 33.4%
## -REML = 330.25  Scale est. = 8.8581    n = 949
```

We see the same terms get selected here too, reassuring!

Now quickly plotting the Q-Q plots to compare Tweedie and negative binomial models:

```
par(mfrow=c(1,2))
qq.gam(dsm_nb_xy_ms_ae, main="Negative binomial")
qq.gam(dsm_tw_xy_ms_ae, main="Tweedie")
```



This is a close call, neither distribution appears to perform considerably better than the other in Q-Q plot terms. We might perfer the Tweedie model as their is a smaller distance between point and line at larger values.

Concurvity

Checking concurvity of terms in the model can be accomplished using the `concurvity()` function.

```
concurvity(dsm_nb_xy_ms)
```

```
##                  para    s(x,y)  s(Depth)
## worst    1.606125e-23 0.9780982 0.9780982
## observed 1.606125e-23 0.3668759 0.7101532
## estimate 1.606125e-23 0.2247389 0.9067347
```

By default the function returns a matrix of a measure of concurvity between one of the terms and the rest of the model.

Compare the output of the models before and after removing terms.

We can see from the above that by the "worst" measure of concurvity, `s(x,y)` and `s(Depth)` are highly concurve with the rest of the model, respectively. We expect to see this in our models in general, since depth can be expressed as a function of location pretty easily (thinking about how bathymetry changes as we move away from the shore).

Looking at how bad things could be, let's re-fit out model with all the terms in and look at those results:

```
dsm_nb_xy_all <- dsm(count~s(x,y, bs="ts") +
                       s(Depth, bs="ts") +
                       s(DistToCAS, bs="ts") +
                       s(SST, bs="ts") +
                       s(EKE, bs="ts") +
                       s(NPP, bs="ts"),
                 df_hn, segs, obs,
                 family=nb())
concurvity(dsm_nb_xy_all)
```

```
##                  para    s(x,y)  s(Depth) s(DistToCAS)    s(SST)    s(EKE)
## worst    4.530024e-23 0.9963493 0.9836597    0.9959057 0.9772853 0.7702479
## observed 4.530024e-23 0.8646807 0.8274429    0.9882052 0.9523370 0.6745515
## estimate 4.530024e-23 0.7580838 0.9272203    0.9642030 0.8978412 0.4906765
##             s(NPP)
## worst    0.9727752
## observed 0.9515884
## estimate 0.8694619
```

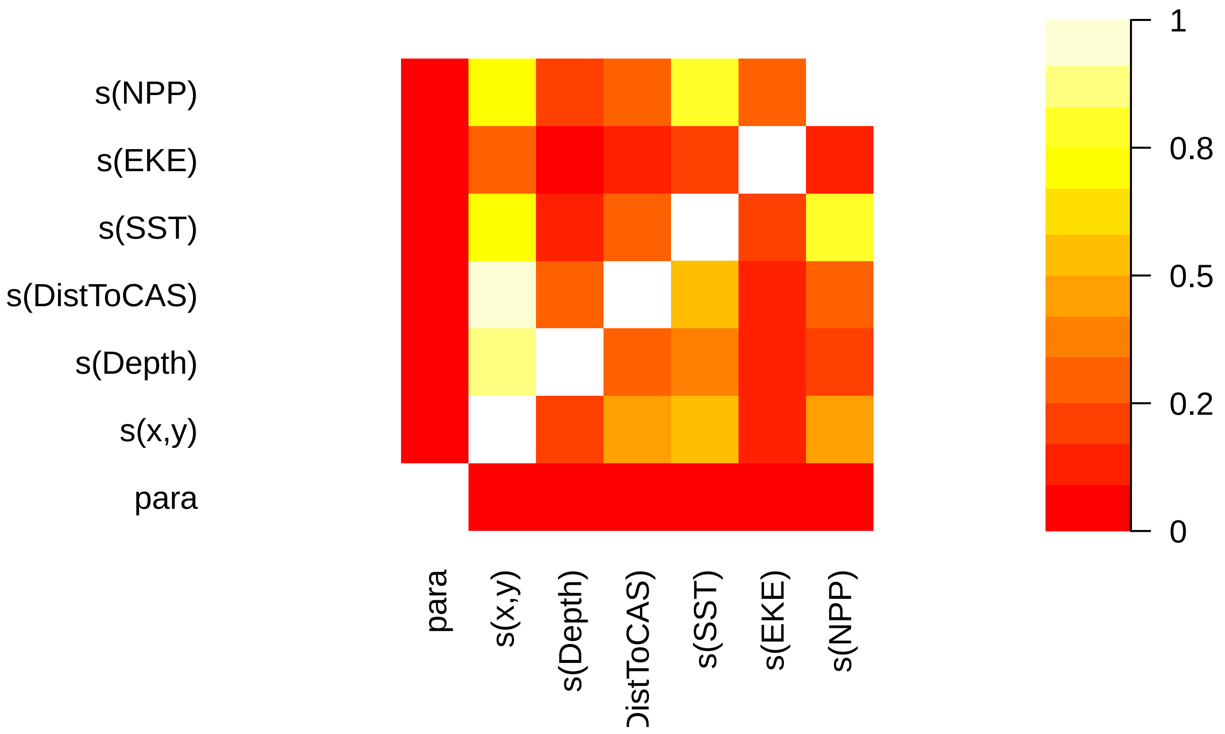
Here we see (generally speaking) most of the terms are highly concurve with the rest of the model using most of the measures of concurvity. We'll look at term-term concurvity next. . .

Reading these matrices can be a little laborious and not very fun. Here is a little bit of code to visualise the concurvity *between terms* in a model by colour coding the matrix.

Again compare the results of plotting for models with different terms.

Visualising the term-term concurvity in the model with all terms:

```
vis.concurvity(dsm_nb_xy_all)
```



Here we can see that all terms are strongly concurve with the spatial smooth – to be expected. We also see that `NPP` and `SST` are highly concurve, which seems biologically plausible.

Sensitivity

Compare bivariate and additive spatial effects

If we replace the bivariate smooth of location (`s(x, y)`) with an additive terms (`s(x)+s(y)`), we may see a difference in the final model (different covariates selected). For the negative binomial distribution:

```
dsm_nb_x_y_ms <- dsm(count~s(x, bs="ts") +
                           s(y, bs="ts") +
                           s(Depth, bs="ts") +
                           s(DistToCAS, bs="ts") +
                           s(SST, bs="ts") +
                           s(EKE, bs="ts") +
                           s(NPP, bs="ts"),
                  df_hn, segs, obs,
                  family=nb())
summary(dsm_nb_x_y_ms)
```

```
##
## Family: Negative Binomial(0.111)
## Link function: log
##
```

```
## Formula:
## count ~ s(x, bs = "ts") + s(y, bs = "ts") + s(Depth, bs = "ts") +
##     s(DistToCAS, bs = "ts") + s(SST, bs = "ts") + s(EKE, bs = "ts") +
##     s(NPP, bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.7261     0.2199  -94.24   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                    edf Ref.df Chi.sq  p-value
## s(x)         5.684e-04      9  0.000   0.3401
## s(y)         3.005e-05      9  0.000   0.6614
## s(Depth)     3.128e+00      9 38.197 1.48e-09 ***
## s(DistToCAS) 6.335e-05      9  0.000   0.4189
## s(SST)       1.225e-04      9  0.000   0.6320
## s(EKE)       8.232e-01      9  5.517   0.0089 **
## s(NPP)       2.537e+00      9 25.751 8.12e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0918   Deviance explained = 40.3%
## -REML = 277.72  Scale est. = 1         n = 949
```

Try performing model selection as before from this base model and compare the resulting models.

Again saving space by doing this in one code chunk, I've numbered the order in which the terms were removed:

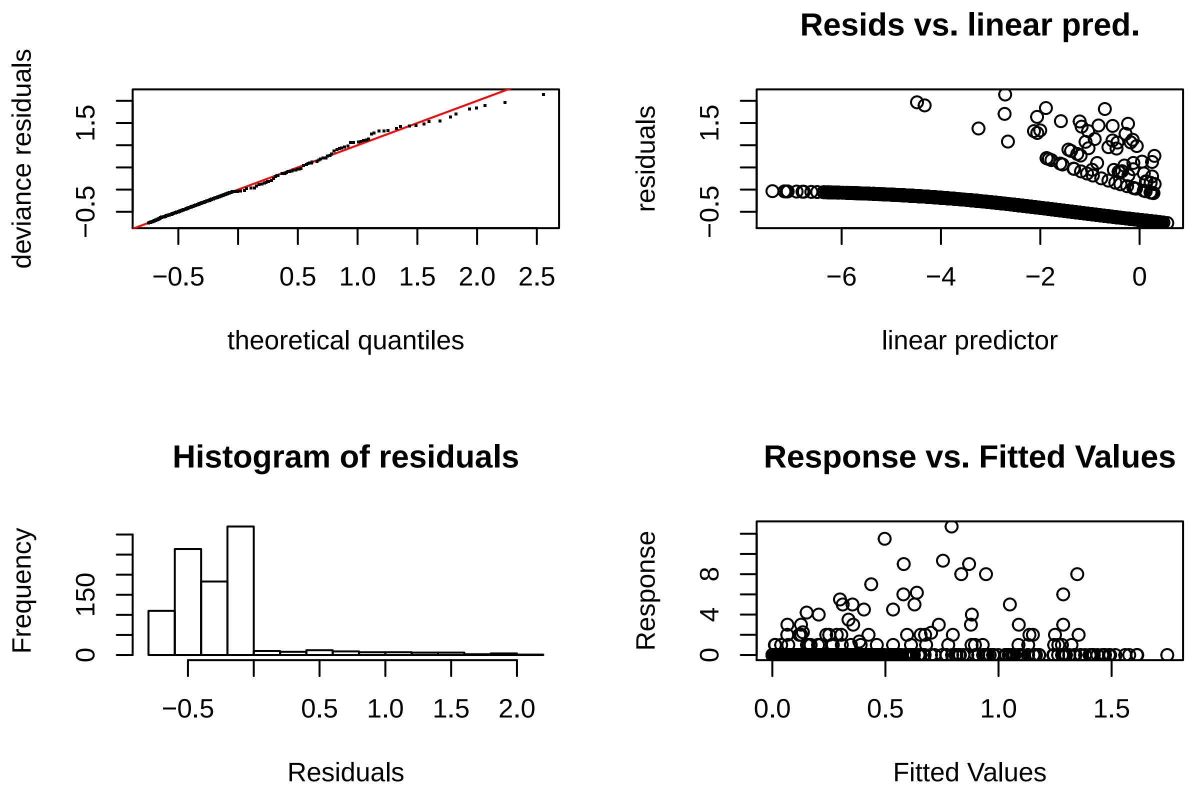
```
dsm_nb_x_y_ms <- dsm(count~ #s(x, bs="ts") + # 5
                           s(y, bs="ts") +
                           s(Depth, bs="ts"),# +
                           #s(DistToCAS, bs="ts") + # 1
                           #s(SST, bs="ts") + # 2
                           #s(EKE, bs="ts"),# + # 4
                           #s(NPP, bs="ts"), # 3
                     df_hn, segs, obs,
                     family=nb())
summary(dsm_nb_x_y_ms)
```

```
##
## Family: Negative Binomial(0.096)
## Link function: log
##
## Formula:
## count ~ s(y, bs = "ts") + s(Depth, bs = "ts") + offset(off.set)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -20.6016     0.2239     -92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df Chi.sq  p-value
## s(y)     0.9829      9  17.43 1.37e-05 ***
## s(Depth) 3.5419      9  53.08 1.03e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.0502   Deviance explained = 35.6%
## -REML = 278.72  Scale est. = 1         n = 949
```

Here we actually remove the `x` term (as it's p-value was 0.00269, which was larger than the significance I'd decided on).

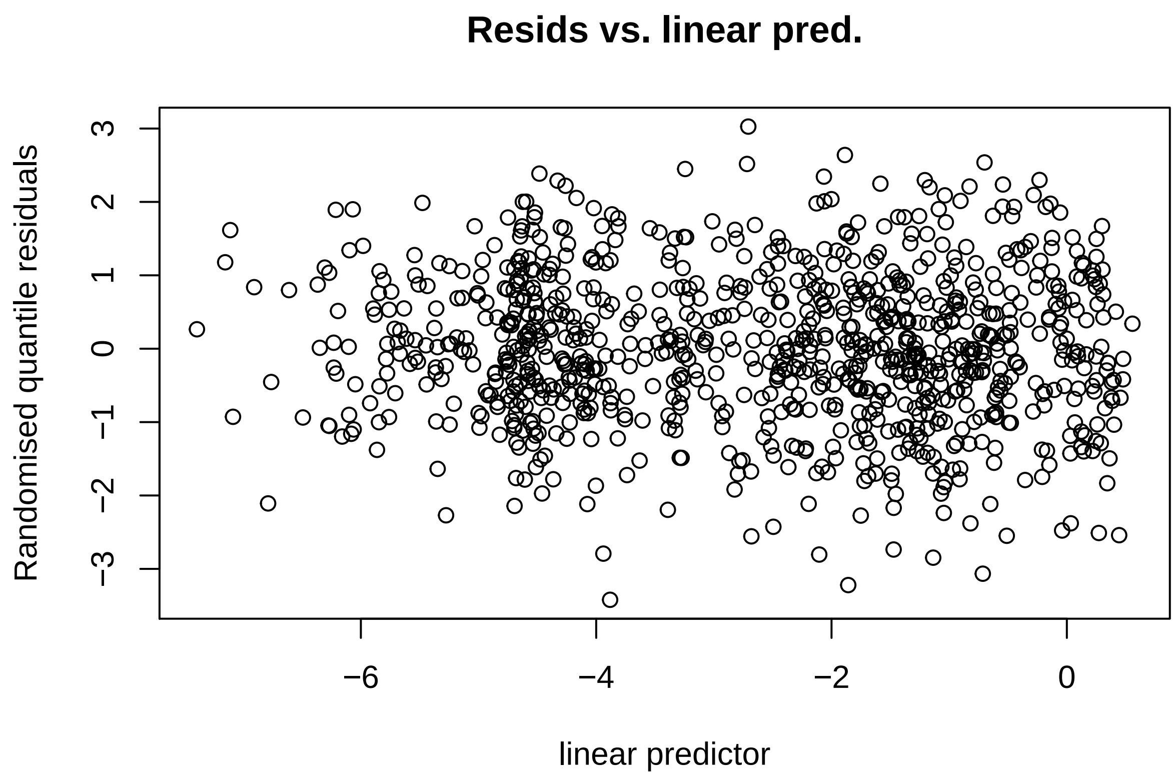
Looking at check plots for this model

```
gam.check(dsm_nb_x_y_ms)
```



```
##
## Method: REML   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [1.2514e-08,7.56025e-07]
## (score 278.7176 & scale 1).
## Hessian positive definite, eigenvalue range [0.3345744,23.95691].
## Model rank =  19 / 19
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##             k'   edf k-index p-value
## s(y)     9.000 0.983    0.64   0.010 **
## s(Depth) 9.000 3.542    0.65   0.015 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
rqgam.check(dsm_nb_x_y_ms)
```



All looks fine here.

Let's do the same with the Tweedie distribution for the sake of completeness.

```
dsm_tw_x_y_ms <- dsm(count~ #s(x, bs="ts") +
                          #s(y, bs="ts") +
                          s(Depth, bs="ts") +
                          #s(DistToCAS, bs="ts") + # 2
                          #s(SST, bs="ts") + # 1
                          #s(EKE, bs="ts") +
                          s(NPP, bs="ts"),
                  df_hn, segs, obs,
                  family=tw())
summary(dsm_tw_x_y_ms)
```

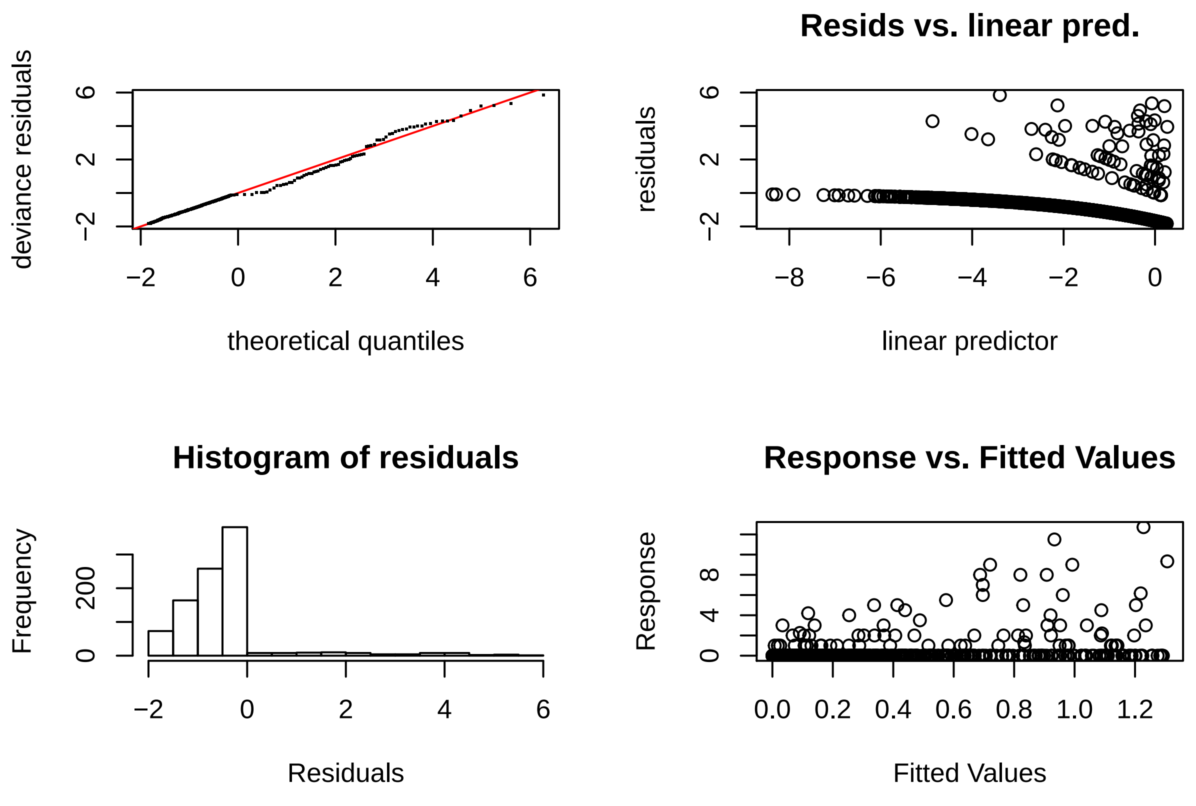
```
##
## Family: Tweedie(p=1.285)
## Link function: log
##
## Formula:
## count ~ s(Depth, bs = "ts") + s(NPP, bs = "ts") + offset(off.set)
##
```

```
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.6155     0.2294  -89.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df     F  p-value
## s(Depth) 3.545      9 5.326 2.89e-11 ***
## s(NPP)   2.780      9 2.977 8.14e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.103   Deviance explained = 34.3%
## -REML = 281.26  Scale est. = 4.7141    n = 949
```

Now we pick rather different covariates in the model…

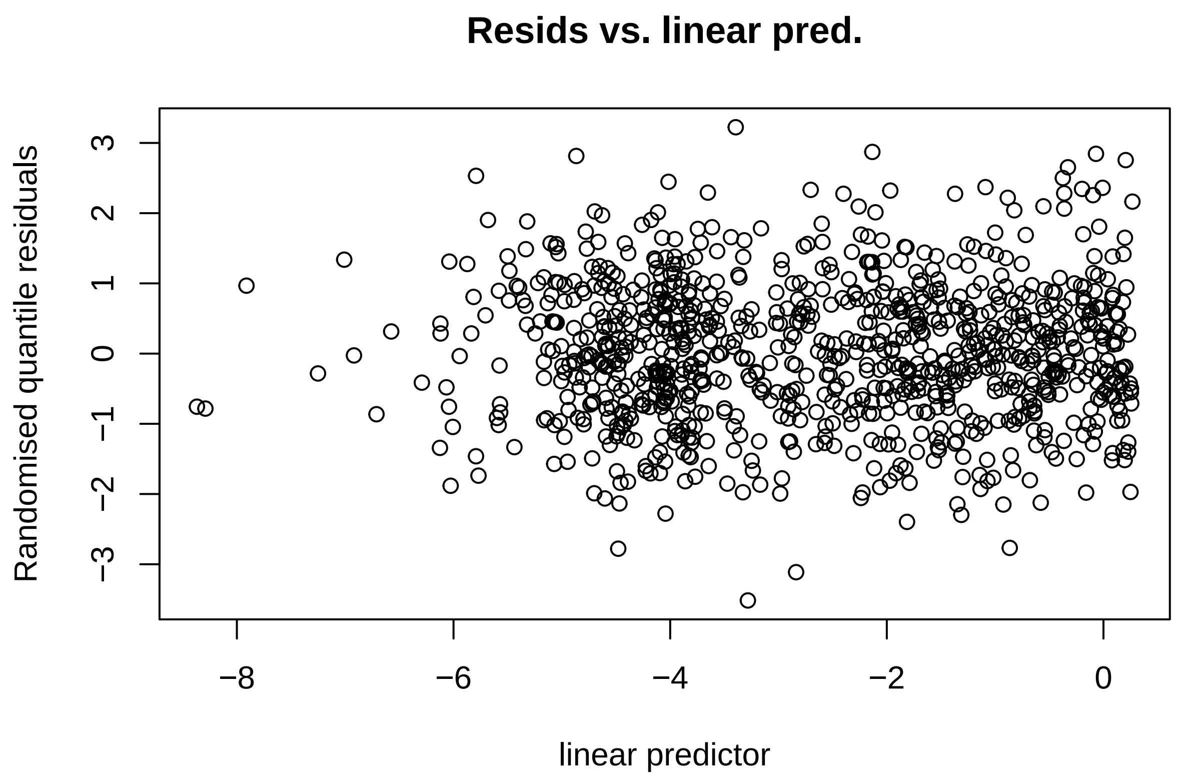
Looking at check plots for this model

```r
gam.check(dsm_tw_x_y_ms)
```



```
##
## Method: REML   Optimizer: outer newton
## full convergence after 7 iterations.
## Gradient range [-4.47048e-08,1.671209e-08]
## (score 281.2606 & scale 4.714094).
## Hessian positive definite, eigenvalue range [0.8310851,213.7643].
## Model rank =  19 / 19
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##              k'  edf k-index p-value
## s(Depth) 9.00 3.55    0.79   0.315
## s(NPP)   9.00 2.78    0.77   0.065 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
rqgam.check(dsm_tw_x_y_ms)
```

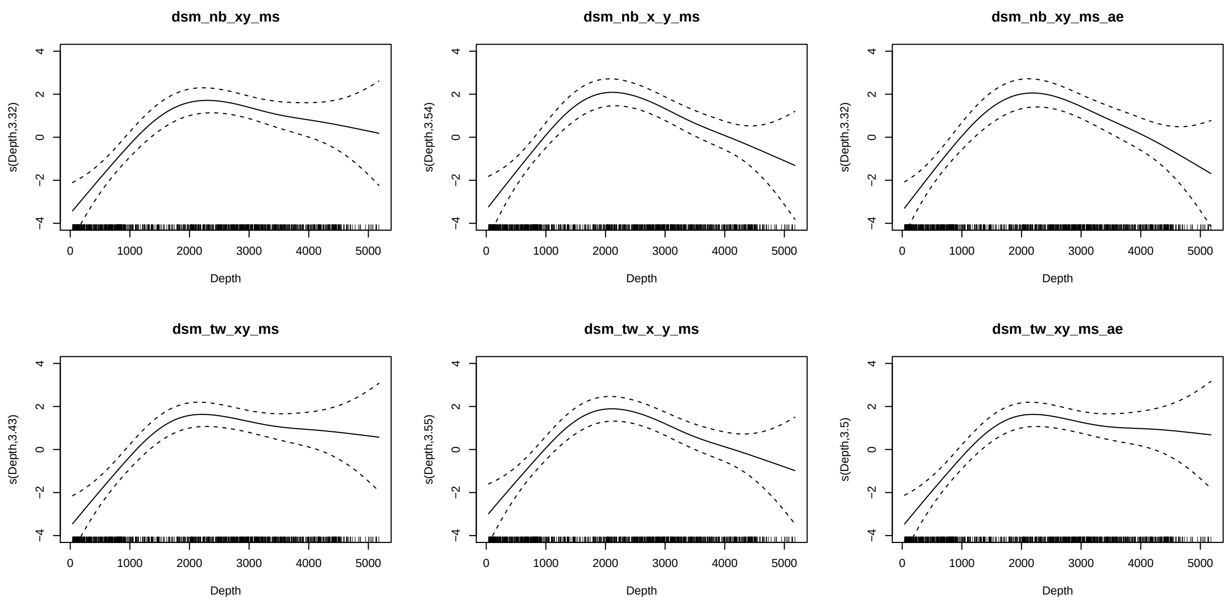


The plots look okay for this model, perhaps there is a slight issue with non-constant variance for the smaller values of the linear predictor.

Compare the resulting smooths from like terms in the model. For example, if depth were selected in both models, compare EDFs and plots, e.g.:

Plotting all of the `Depth` terms, which are common to all models:

```
par(mfrow=c(2,3))
plot(dsm_nb_xy_ms, main="dsm_nb_xy_ms", select=2, scale=0, ylim=c(-4,4))
plot(dsm_nb_x_y_ms, main="dsm_nb_x_y_ms", select=2, scale=0, ylim=c(-4,4))
plot(dsm_nb_xy_ms_ae, main="dsm_nb_xy_ms_ae", select=2, scale=0, ylim=c(-4,4))
plot(dsm_tw_xy_ms, main="dsm_tw_xy_ms", select=2, scale=0, ylim=c(-4,4))
plot(dsm_tw_x_y_ms, main="dsm_tw_x_y_ms", select=1, scale=0, ylim=c(-4,4))
plot(dsm_tw_xy_ms_ae, main="dsm_tw_xy_ms_ae", select=2, scale=0, ylim=c(-4,4))
```



As we can see, these are pretty similar between models. In general showing a peak at around 2000m.

Note that there `select=` picks just one term to plot. These are in the order in which the terms occur in the `summary()` output (so you may well need to adjust the above code).

Comparing models

As with the detection functions in the earlier exercises, here is a quick function to generate model results tables with appropriate summary statistics:

```
summarize_dsm <- function(model){

  summ <- summary(model)

  data.frame(response = model$family$family,
             terms    = paste(rownames(summ$s.table), collapse=", "),
             AIC      = AIC(model),
             REML     = model$gcv.ubre,
             "Deviance_explained" = paste0(round(summ$dev.expl*100,2),"%")
            )

}
```

We can, again, make a list of the models and give that to the above function

Note here we're just going to compare the models that have count as the response:

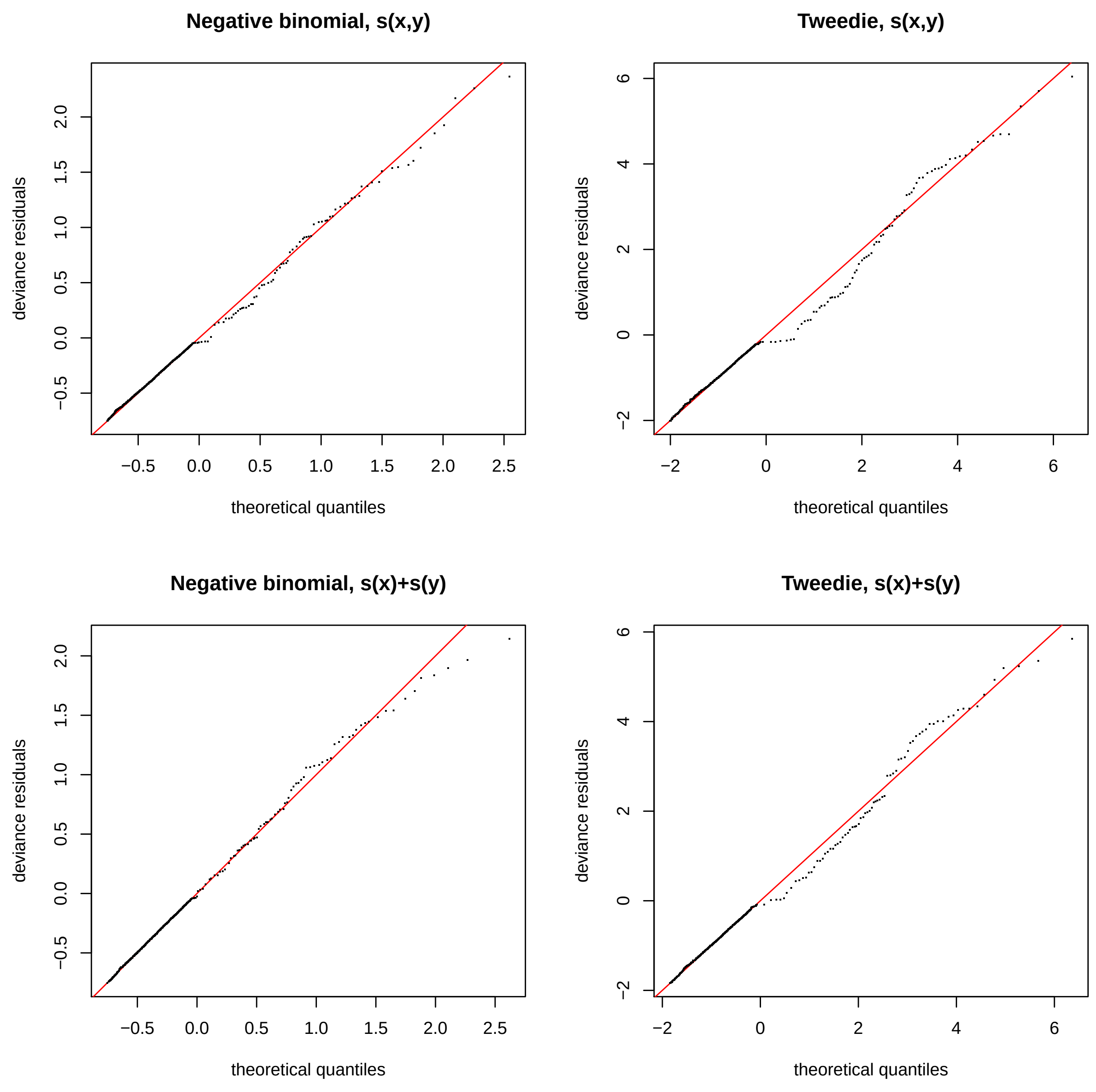
```
# add your models to this list!
model_list <- list(dsm_nb_xy, dsm_nb_x_y_ms, dsm_tw_xy_ms, dsm_tw_x_y_ms)
library(plyr)
summary_table <- ldply(model_list, summarize_dsm)
row.names(summary_table) <- c("dsm_nb_xy", "dsm_nb_x_y_ms", "dsm_tw_xy_ms", "dsm_tw_x_y_ms")
```

```
summary_table <- summary_table[order(summary_table$REML, decreasing=TRUE),]
kable(summary_table)
```

	response	terms	AIC	REML	Deviance_explained
dsm_nb_xy	Negative Binomial(0.089)	s(x,y)	787.8953	285.4549	34.69%
dsm_tw_x_y_ms	Tweedie(p=1.285)	s(Depth), s(NPP)	1227.8914	281.2606	34.29%
dsm_tw_xy_ms	Tweedie(p=1.283)	s(x,y), s(Depth)	1225.8925	279.7574	33.45%
dsm_nb_x_y_ms	Negative Binomial(0.096)	s(y), s(Depth)	762.9281	278.7176	35.64%

We can also plot the Q-Q plots for these models to compare their distributional assumptions:

```r
par(mfrow=c(2,2))
qq.gam(dsm_nb_xy_ms, main="Negative binomial, s(x,y)")
qq.gam(dsm_tw_xy_ms, main="Tweedie, s(x,y)")
qq.gam(dsm_nb_x_y_ms, main="Negative binomial, s(x)+s(y)")
qq.gam(dsm_tw_x_y_ms, main="Tweedie, s(x)+s(y)")
```



We'll save all four models below, but our selection were based on the deviance explained and the Q-Q plot would be for `dsm_nb_xy_ms`.

Saving models

Now save the models that you'd like to use to predict with later. I recommend saving as many models as you can so you can compare their results in the next practical.

```r
# add your models here
save(dsm_nb_xy_ms, dsm_nb_x_y_ms, dsm_tw_xy_ms, dsm_tw_x_y_ms,
     file="dsms.RData")
```