# Cyclothone example

This is an example of how to use KRMr.

This is not an official release and is still in experimental phase.

## Reading the shape information

Cyclothone used here are non-swimbladdered fish. We have to shape files, one for the fish body and one for the swimbladder.
The shape file for the fish body contains one column named `x_fb` the fishbody (fb) along the x axis (Length), `w_fb` the width (seen from top) of the fish body at each position x (along the fish body length), `z_fbL` the lower height of the fish body along the x axis (lower (L) fish body (fb) extend in direction z) and `z_fbU` the upper height along the x axis (upper (U) fish body (fb) extend in direction z).

Analoguously, the swimbadder (sb) shape file contains `x_sb` - x axis (Length direction), `w_sb` - Width of the swimbladder along the x axis, `z_sbL` - lower swimbladder extend and `z_sbU` - upper swimbladder extend.

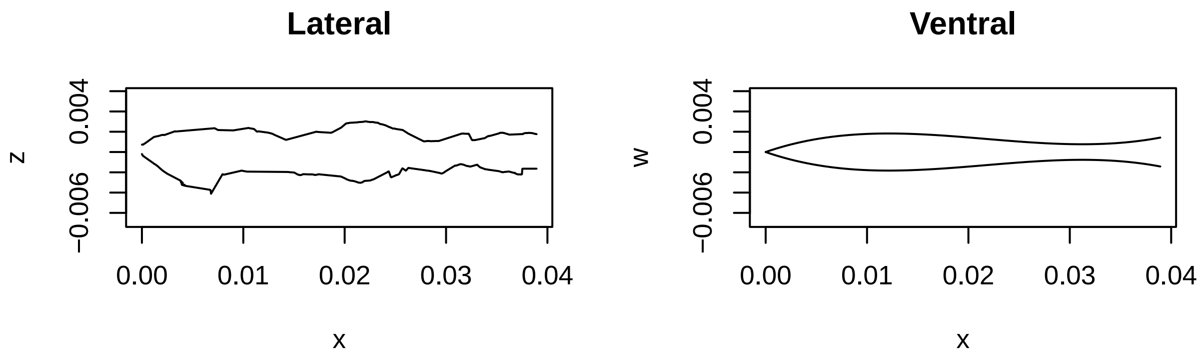
```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(KRMr)
stb_fb <- read.csv(paste0(dirname(getwd()),'/data/fb1contour.csv'))
```

Let's have a look at the shapes we loaded.
To plot the loaded shapes, we can use the `shplot(x_fb, w_fb, x_sb, w_sb, z_fbU, z_fbL, z_sbU, z_sbL)` function:

```
fb =stb_fb
layout(matrix(c(1,2,3,3), 2, 2, byrow = TRUE))
KRMr::shplot(x_fb = fb$x_fb, w_fb = fb$w_fb,
             x_sb = NA, w_sb = NA,
             z_fbU = fb$z_fbU, z_fbL = fb$z_fbL,
             z_sbU = NA, z_sbL = NA)
```




## Running a KRM simulation

Now that we have our shapes sorted, let's have a go at a KRM simulation.
We want to get TS at 38, 70, 120 and 200 kHz, with the below defined settings:

- c.w = ambient water sound speed
- rho.w = density ambient water

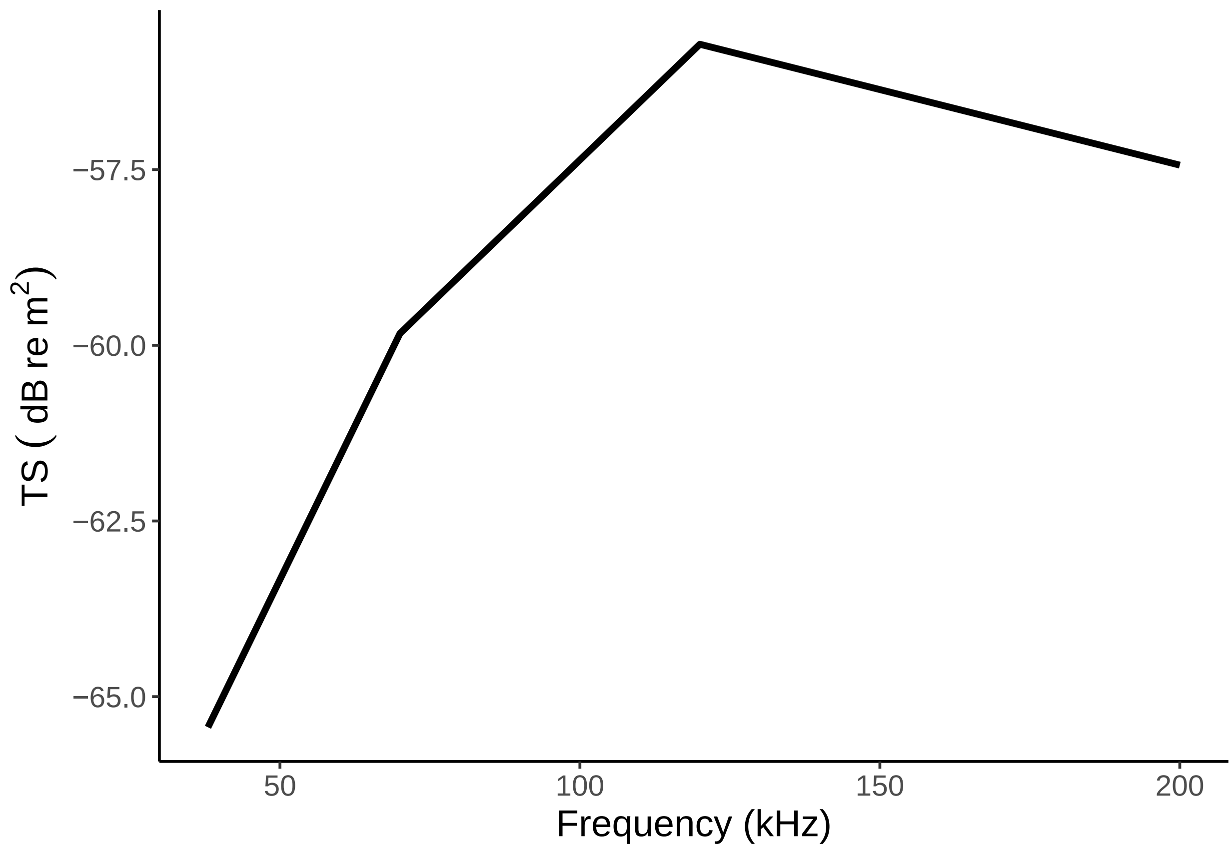
- theta = orientation in degrees
- c.fb = sound speed inside fish body
- c.sb = sound speed inside swimbladder
- rho.sb = density inside swimbladder
- rho.fb = density inside fish body = L = length of the fishbody in m
- x_fb = Fish body coordinates along x axis
- x_sb = Swimbladder coordinates along x axis
- w_fb = Fish body width along x axis
- w_sb = Swimbladder width coordinates along x axis
- z_fbU = Upper height of the fish body along the x axis
- z_fbL = Lower height of the fish body along the x axis
- z_fbU = Upper height of the swimbladder along the x axis
- z_fbL = Lower height of the swimbladder along the x axis

A description of the parameters is also available in the help files `?KRMr::krm` (runs a single krm simulation) or `?KRMr::krm.sim` (runs multiple simulations if any input parameter contains more than one value (except for coordinates)

```
TS = KRMr::krm.sim(frequency =c(38,70,120,200) * 1000,
                   c.w = 1490,
                   rho.w = 1030,
                   theta=90,
                   c.fb = 1570,
                   c.sb = 345,
                   rho.sb = 1.24,
                   rho.fb = 1070,
                   L=0.25,
                   x_fb = fb$x_fb,
                   x_sb = NULL,
                   w_fb = fb$w_fb,
                   w_sb = NULL,
                   z_fbU = fb$z_fbU,
                   z_fbL = fb$z_fbL,
                   z_sbU = NULL,
                   z_sbL = NULL)
```

Let's have a look at the outcome:

```
library(ggplot2)
ggplot(data=TS,
       aes(x=frequency/1000, y=TS))+
  geom_line(size=1.2)+
  ylab(expression(TS~(~dB~re~m^2)))+
  xlab('Frequency (kHz)')+
  theme_classic()+
  theme(text=element_text(size=14),
        legend.position='top')
```



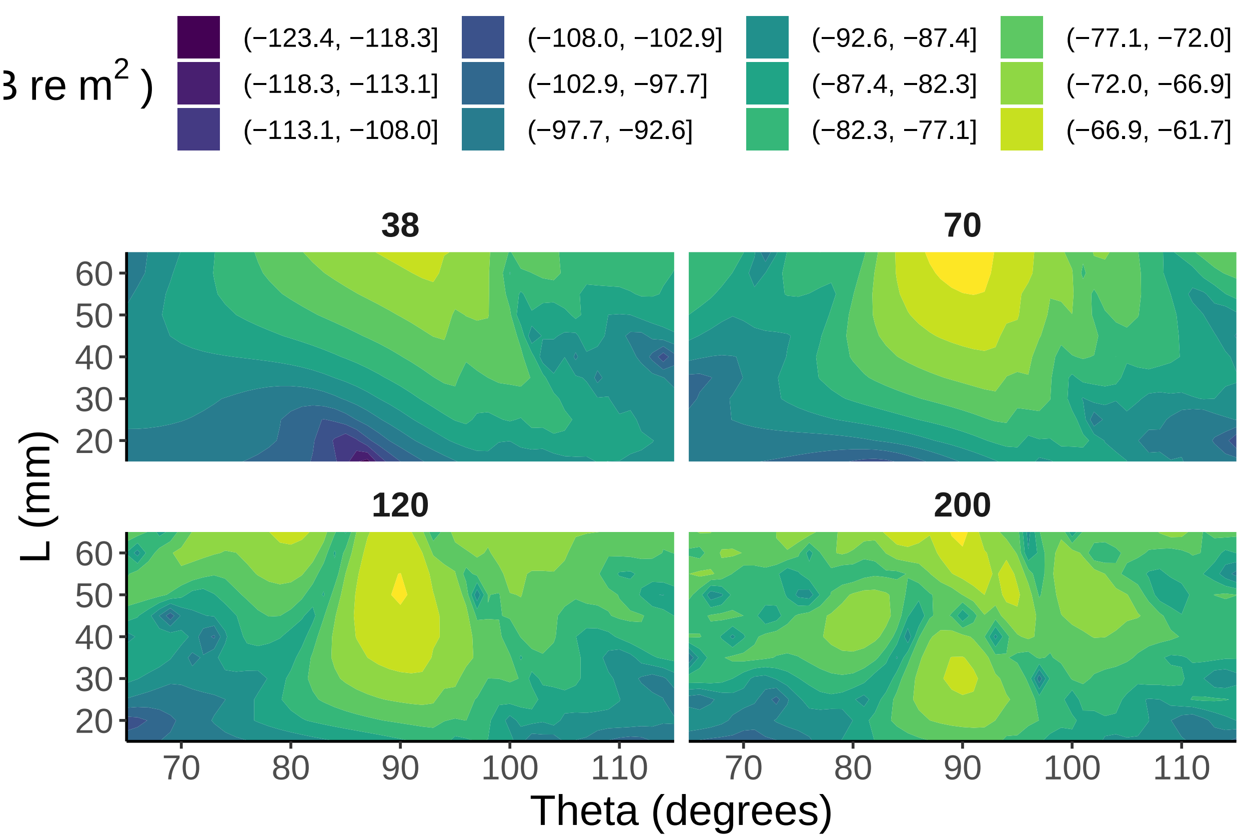

We can also compute slightly more sophisticated simulations of a range of frequencies, lengths and orientations (remermbe the KRM model is only valid for orientations, close to 90 degrees ~65-115 degrees):

```
TS = krm.sim(frequency =c(38,70,120,200) * 1000,
                   c.w = 1490,
                   rho.w = 1030,
                   theta=seq(65,115),
                   c.fb = 1570,
                   c.sb = 345,
                   rho.sb = 1.24,
                   rho.fb = 1070,
                   L=seq(0.015,0.065,by=0.005),
                   x_fb = fb$x_fb,
                   x_sb = NULL,
                   w_fb = fb$w_fb,
                   w_sb =  NULL,
                   z_fbU = fb$z_fbU,
                   z_fbL = fb$z_fbL,
                   z_sbU = NULL,
                   z_sbL = NULL)
```

Let's plot the results:

```
ggplot2::ggplot(data=TS, ggplot2::aes(x= theta, y=L*1000 , z=TS))+
  facet_wrap(.~frequency/1000)+
  geom_contour_filled(bins=15)+
```

```
  ggplot2::scale_fill_viridis_d(expression(TS~'('~dB~re~ m^2~')'))+
  ggplot2::xlab('Theta (degrees)')+
  ggplot2::ylab(expression(L~'(mm)'))+
  ggplot2::scale_x_continuous(expand=c(0,0))+
  ggplot2::scale_y_continuous(expand=c(0,0))+
  ggplot2::theme_classic()+
  ggplot2::theme(legend.position='top',
                 legend.text=element_text(size=10),
                 text=element_text(size=16),
                 strip.background = element_blank(),
                 strip.text = element_text(face='bold'))
```



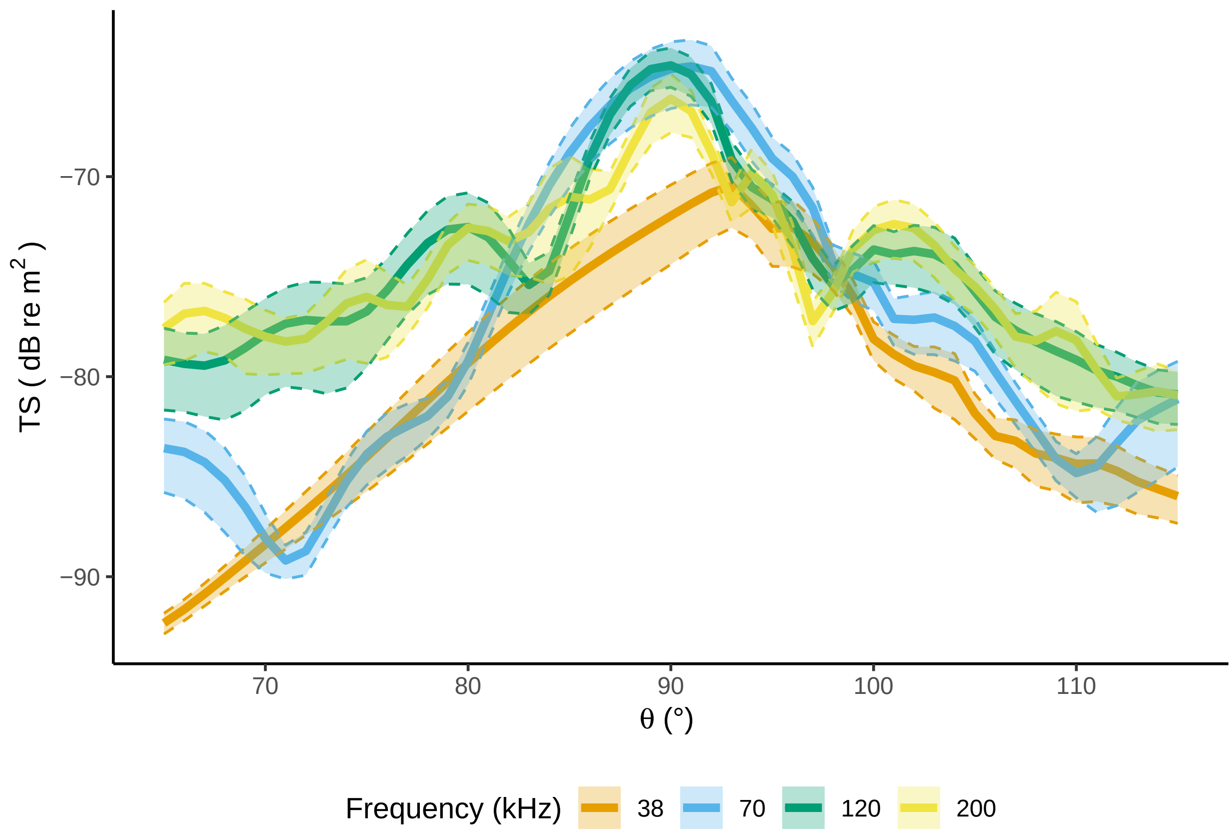

Or produce another plot:

```
TS%>%
  mutate(frequency=frequency/1000)%>%
  mutate(sigma=10^(TS/10))%>%
  group_by(frequency,theta) %>%
  summarise(mean.sigma = mean(sigma, na.rm=T),
            sd.sigma = sd(sigma, na.rm=T),
            n.TS = n()) %>%
  mutate(se.sigma = sd.sigma / sqrt(n.TS),
         lower.ci.sigma = mean.sigma - qt(1 - (0.05 / 2), n.TS - 1) * se.sigma,
         upper.ci.sigma = mean.sigma + qt(1 - (0.05 / 2), n.TS - 1) * se.sigma,
         mean.TS=10*log10(mean.sigma),
         sd.TS=10*log10(sd.sigma),
```

```r
         se.TS=10*log10(se.sigma),
         lower.ci.TS=10*log10(lower.ci.sigma),
         upper.ci.TS=10*log10(upper.ci.sigma))->TS.ci
```

```
## `summarise()` has grouped output by 'frequency'. You can override using the `.groups` argument.

## Warning in mask$eval_all_mutate(quo): NaNs produced

## Warning in mask$eval_all_mutate(quo): NaNs produced

## Warning in mask$eval_all_mutate(quo): NaNs produced

## Warning in mask$eval_all_mutate(quo): NaNs produced
```

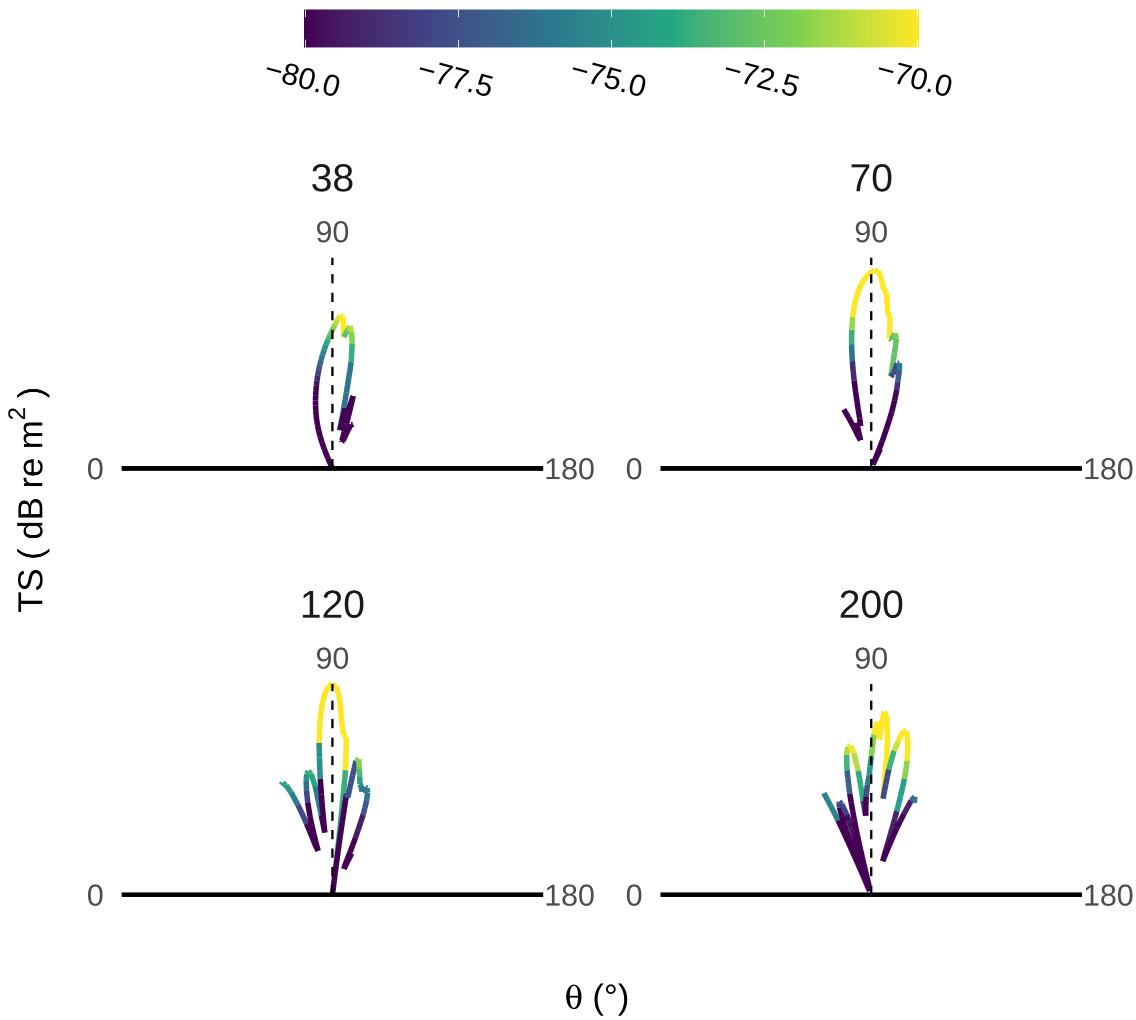
```r
#TS.ci=TS
#TS.ci$sigma=10^(TS.ci$TS/10)
okabe <- c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

ggplot(data=TS.ci, aes(x=theta, group=factor(frequency)))+
  #geom_smooth(aes(x= theta, y=sigma, group=frequency, col=factor(frequency),
  #                method = lm, formula=y~x,
  #                    method.args = list(family = gaussian(link = 'log'))), lwd=1.5)

  geom_line(aes(x= theta, y=mean.TS, group=frequency, col=factor(frequency)), lwd=1.5)+
  geom_line(aes(x= theta, y=10*log10(mean.sigma+se.sigma), group=frequency, col=factor(frequency)),lty=
    geom_line(aes(x= theta, y=10*log10(mean.sigma-se.sigma), group=frequency, col=factor(frequency)),lt
  geom_ribbon(
          aes(ymin=10*log10(mean.sigma-se.sigma),
              ymax=10*log10(mean.sigma+se.sigma), fill=factor(frequency)), alpha=0.3)+
  theme_classic()+
  scale_color_manual(values=okabe, name='Frequency (kHz)')+
  scale_fill_manual(values=okabe, name='Frequency (kHz)')+
  theme(legend.position = 'bottom')+
  ylab(expression(TS~'('~dB~re~m^2~')' ))+
  xlab(expression(paste(theta,' (\u00B0) ')))
```




We can also look at orientation plots for a fish of 5 cm:

```
TSrot = krm.sim(frequency =c(38,70,120,200) * 1000,
                   c.w = 1490,
                   rho.w = 1030,
                   theta=65:115,
                   c.fb = 1570,
                   c.sb = 345,
                   rho.sb = 1.24,
                   rho.fb = 1070,
                   L=0.05,
                   x_fb = fb$x_fb,
                   x_sb = NA,#sb$x_sb,
                   w_fb = fb$w_fb,
                   w_sb = NA,#sb$w_sb,
                   z_fbU = fb$z_fbU,
                   z_fbL = fb$z_fbL,
                   z_sbU = NA,#sb$z_sbU,
                   z_sbL = NA)#sb$z_sbL)
ggplot(TSrot, aes(x = theta, y = TS, group=frequency/1000, col=TS)) +
  geom_path(size=1.2) +
  facet_wrap(.~frequency/1000)+
  scale_x_continuous(limits=c(0,360), breaks=c(0,90,180))+
  coord_polar(start=-pi/2,direction=1)+
  scale_colour_viridis_c(name='', limits=c(-80,-70), oob=scales::squish)+
```

```
ylab(expression(TS~'('~dB~re~m^2~')' ))+
xlab(expression(paste(theta,' (\u00B0) ')))+

geom_vline(xintercept=90, lty=2)+
geom_vline(xintercept=0,size=1)+
geom_vline(xintercept=180,size=1)+

theme_classic()+
theme(strip.background = element_blank(),
      strip.text=element_text(size=18),
      legend.position='top',
      legend.text=element_text(angle=-15),
      legend.key.width = unit(2,'cm'),
      axis.line.y = element_blank(),
      axis.text.y = element_blank(),
      axis.ticks.y = element_blank(),
      axis.line.x = element_blank(),
      panel.spacing.y = unit(-5,'lines'),
      axis.title.x = element_text(vjust=27, size=16),
      axis.title.y = element_text(hjust=0.75, size=16),
      text=element_text(size=18))
```



-80.0
-77.5
-75.0
-72.5
-70.0
38
90
70
90
TS ( dB re m2 )
0
180
0
180
120
90
200
90
0
180
0
180
θ (°)