Comparing VAST and sdmTMB GOA indices

Contents

```
#remotes::install_github("pbs-assess/sdmTMB", dependencies = TRUE)
library(VAST)
library(sp)
library(sdmTMB)
library(dplyr)
library(ggplot2)
library(here)

species <- "Gadus_macrocephalus"
```

We will fit geostatistical spatiotemporal models with VAST and sdmTMB for the purposes of index standardization and compare the outputs given the same data. We will use data from the GOA AFSC GAP bottom trawl survey for the species specified above. The density units are kg/km^2.

```
dat_ll <- readRDS(here("species_specific_code", "GOA", species,
                       "data", paste0("Data_Geostat_", species, ".rds")))

dat_ll <-  dplyr::transmute(dat_ll,
                        cpue_kg_km2 = Catch_KG,
                        year = as.integer(Year),
                        vessel = "missing",
                        effort = AreaSwept_km2,
                        lat = Lat,
                        lon = Lon,
                        pass = 0) %>%
              as.data.frame() # ensure not a tibble
```

We begin by specifying the VAST model. To specify the mesh used to approximate the spatial process, which is used in the SPDE calculations, we use the k-means method in VAST. Rather than specifying the cutoff distance, meshes in VAST are typically generated by specifying only the number of knots, which we will later pass, along with other model settings to the function make_settings. We will use 750 knots, the same number in the mesh created in the existing production VAST index for this stock and region.

We will include a factor predictor that represents the mean estimate for each time slice. Settings used for index standardization are applied by specifying `purpose = "index2"`.

Unlike in sdmTMB, the fitting and predicting steps are all accomplished with the function `fit_model()` and thus we need to specify the prediction grid (referred to as the "extrapolation grid" in VAST). Here, X and Y are coordinates in UTM zone 3.

```
GOAgrid <- read.csv(here("extrapolation_grids", "GOAThorsonGrid_Less700m.csv"))
input_grid <- cbind(Lat=GOAgrid$Latitude,
                    Lon=GOAgrid$Longitude,
```

```
                    Area_km2=GOAgrid$Shape_Area/1000000)

settings <- make_settings(
  n_x = 750, # number of vertices in the SPDE mesh
  Region = "user",
  purpose = "index2", # index of abundance with Gamma for positive catches
  fine_scale = TRUE, # use bilinear interpolation from the INLA 'A' matrix
  zone = 3,
  Options = c("Calculate_Range" = TRUE, "Calculate_effective_area" = TRUE,
              "treat_nonencounter_as_zero" = FALSE),
  ObsModel = c(2, 1), # conventional logit-linked delta-Gamma; (2,4) if there are years with 100% encou
  bias.correct = TRUE,
  use_anisotropy = TRUE,
  max_cells = Inf, # use all grid cells from the extrapolation grid, production model used 2000
  knot_method = "grid", # or "samples"
  strata.limits = data.frame(STRATA = as.factor('All_areas')) # customize to sp.
)
```

Next we will fit a GLMM (generalized linear mixed effects model).

```
# create folder for saved output:
dir.create(paste0(here("species_specific_code", "GOA", species,
                       "index_comparison")), showWarnings = FALSE)

f <- here("species_specific_code", "GOA", species, "index_comparison", "VASTfit.RDS")
if (!file.exists(f)) {
  fit <- fit_model(
    settings = settings,
    Lat_i = dat_ll[, "lat"],
    Lon_i = dat_ll[, "lon"],
    t_i = dat_ll[, "year"],
    b_i = dat_ll[, "cpue_kg_km2"],
    a_i = dat_ll[, "effort"],
    input_grid = input_grid,
    working_dir = paste0(here("species_specific_code", "GOA",
                              species, "index_comparison"), "/")
  )
  saveRDS(fit, file = f)
} else {
  fit <- readRDS(f)
  fit <- reload_model(fit)
}
#> Maximum absolute gradient of 1.99e-06: No evidence of non-convergence
```

We can look at parameter estimates. First we see estimates from the binomial component and second we see estimates from the positive Gamma component.

```
fit$parameter_estimates$diagnostics
#>           Param starting_value     Lower         MLE     Upper final_gradient
#> 1    ln_H_input     0.13823879 -5.000000  0.13824940  5.000000  -1.588223e-08
#> 2    ln_H_input     0.33857991 -5.000000  0.33860864  5.000000   1.378718e-08
#> 3      beta1_ft     0.06843785      -Inf  0.06836707       Inf   1.175008e-08
#> 4      beta1_ft     0.09829280      -Inf  0.09822163       Inf   1.116844e-08
```

```
#> 5      beta1_ft    -0.13737195      -Inf -0.13744627       Inf   2.106849e-08
#> 6      beta1_ft    -0.41610908      -Inf -0.41617999       Inf   7.024088e-09
#> 7      beta1_ft    -0.86561410      -Inf -0.86568080       Inf  -4.111496e-09
#> 8      beta1_ft    -0.60680136      -Inf -0.60686825       Inf   3.616183e-10
#> 9      beta1_ft    -0.52833246      -Inf -0.52839178       Inf  -1.136970e-08
#> 10     beta1_ft    -0.52371308      -Inf -0.52378791       Inf   1.363214e-08
#> 11     beta1_ft    -0.06886954      -Inf -0.06894752       Inf   2.718483e-08
#> 12     beta1_ft    -0.10725440      -Inf -0.10731783       Inf  -4.161173e-09
#> 13     beta1_ft    -0.14658108      -Inf -0.14664722       Inf   2.995797e-09
#> 14     beta1_ft    -0.17628867      -Inf -0.17635919       Inf   1.312260e-08
#> 15     beta1_ft    -0.85903896      -Inf -0.85909940       Inf  -7.138457e-09
#> 16     beta1_ft    -0.48940291      -Inf -0.48947738       Inf   1.267775e-08
#> 17     beta1_ft    -0.39973269      -Inf -0.39979863       Inf   4.727486e-09
#> 18     beta1_ft    -0.19563090      -Inf -0.19571924       Inf   2.745117e-08
#> 19   L_omega1_z     1.47796228      -Inf  1.47799878       Inf  -1.961502e-07
#> 20 L_epsilon1_z     0.42822785      -Inf  0.42822211       Inf  -1.129275e-07
#> 21    logkappa1    -3.35089844 -6.775053 -3.35092745 -1.659693   1.292130e-07
#> 22     beta2_ft     6.12943638      -Inf  6.12942110       Inf  -1.148680e-08
#> 23     beta2_ft     5.99092362      -Inf  5.99092096       Inf  -2.107811e-09
#> 24     beta2_ft     6.27896794      -Inf  6.27895822       Inf  -5.640430e-09
#> 25     beta2_ft     6.01748260      -Inf  6.01748113       Inf  -2.021665e-09
#> 26     beta2_ft     6.09860256      -Inf  6.09859603       Inf  -7.945577e-09
#> 27     beta2_ft     6.09623363      -Inf  6.09622247       Inf  -9.587538e-09
#> 28     beta2_ft     6.10138812      -Inf  6.10139039       Inf   1.841990e-09
#> 29     beta2_ft     5.96815885      -Inf  5.96815653       Inf  -4.209006e-09
#> 30     beta2_ft     6.13056239      -Inf  6.13054226       Inf  -5.789687e-09
#> 31     beta2_ft     6.05838577      -Inf  6.05838092       Inf  -2.186603e-09
#> 32     beta2_ft     6.19480966      -Inf  6.19479510       Inf  -8.245181e-09
#> 33     beta2_ft     5.88873440      -Inf  5.88872365       Inf  -7.459455e-09
#> 34     beta2_ft     5.75774984      -Inf  5.75775432       Inf  -9.871286e-10
#> 35     beta2_ft     5.79318368      -Inf  5.79315651       Inf  -9.237205e-09
#> 36     beta2_ft     5.85757291      -Inf  5.85755499       Inf  -4.306962e-09
#> 37     beta2_ft     5.79324679      -Inf  5.79323700       Inf  -9.049188e-10
#> 38   L_omega2_z     0.77379451      -Inf  0.77381015       Inf  -2.035232e-07
#> 39 L_epsilon2_z     1.13385313      -Inf  1.13387752       Inf  -5.867277e-07
#> 40    logkappa2    -2.33130964 -6.775053 -2.33122157 -1.659693   9.943082e-08
#> 41    logSigmaM     0.05376951      -Inf  0.05376824 10.000000  -8.372568e-07
```

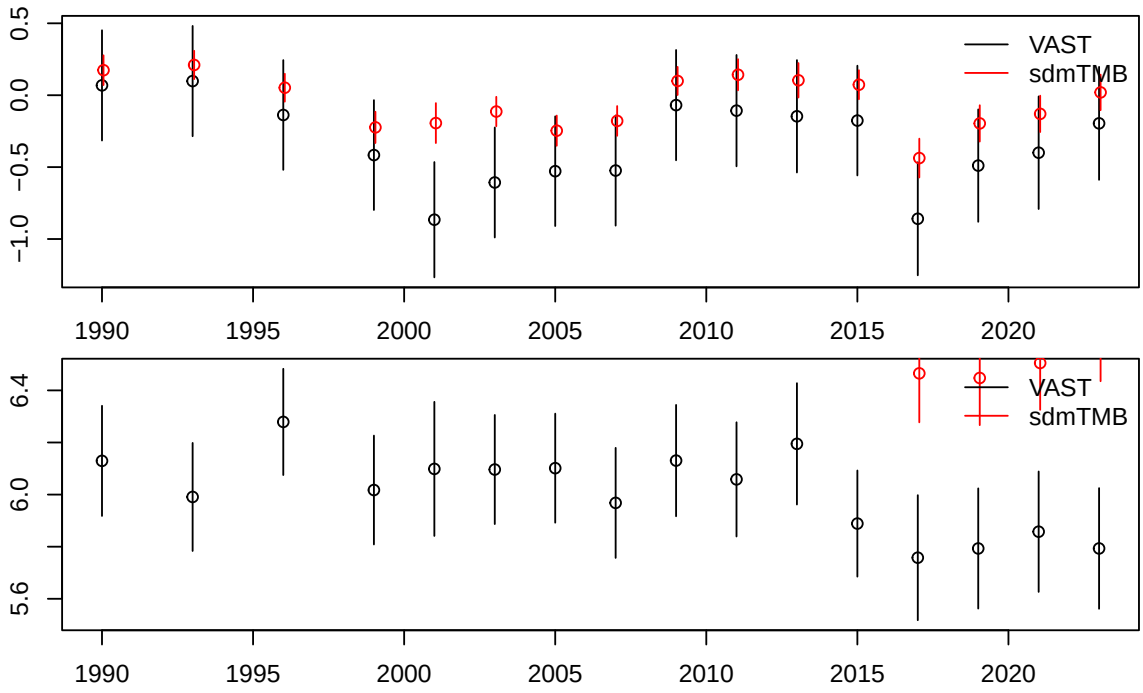
Now we fit the same model in sdmTMB:

```
dat <- dat_ll %>%
  rename(X = lon, Y = lat)

dat$year_f <- as.factor(dat$year)

coordinates(dat) <- ~ X + Y
proj4string(dat) <- CRS("+proj=longlat +datum=WGS84")
dat <- as.data.frame(spTransform(dat, CRS("+proj=utm +zone=3")))
# scale to km so values don't get too large
dat$X <- dat$coords.x1 / 1000
dat$Y <- dat$coords.x2 / 1000

f1 <- here("species_specific_code", "GOA", species,
           "index_comparison", "fit_sdmTMB.RDS")
```

```r
if (!file.exists(f1)) {
# make mesh and fit model
mesh <-  make_mesh(dat, xy_cols = c("X", "Y"), mesh = fit$spatial_list$MeshList$anisotropic_mesh) #pass
#mesh <-  make_mesh(dat, xy_cols = c("X", "Y"), n_knots = 50, type = "kmeans") #coarser mesh for experi

fit_sdmTMB <- sdmTMB(
  cpue_kg_km2 ~ 0 + year_f,
  data = dat,
  mesh = mesh,
  family = delta_gamma(type = "poisson-link"),
  time = "year",
  spatial = "on",
  spatiotemporal = "iid",
  silent = FALSE,
  anisotropy = TRUE,
  do_fit = TRUE
  #, do_index = TRUE (to compute index at same time, requires passing args)
)
fit_sdmTMB
saveRDS(fit_sdmTMB, file = here("species_specific_code", "GOA",
                                species, "index_comparison",
                                "fit_sdmTMB.RDS"))
} else {
fit_sdmTMB <- readRDS(f1)
}

# diagnose estimation issues due to model structure
#TMBhelper::check_estimability(fit_sdmTMB$tmb_obj)
```

We wrote some custom code to extract comparable parameters (not shown above). Here are the annual mean estimates in link space with 95% confidence intervals for the two components to the delta model:

```r
par(mfrow = c(2, 1), cex = 0.8, mar = c(1.5, 1, 1, 1), oma = c(2, 3, 1, 1))
plot_betas(fit, fit_sdmTMB, "beta1_ft", sdmTMB_pars = 1)
plot_betas(fit, fit_sdmTMB, "beta2_ft", sdmTMB_pars = 2)
```



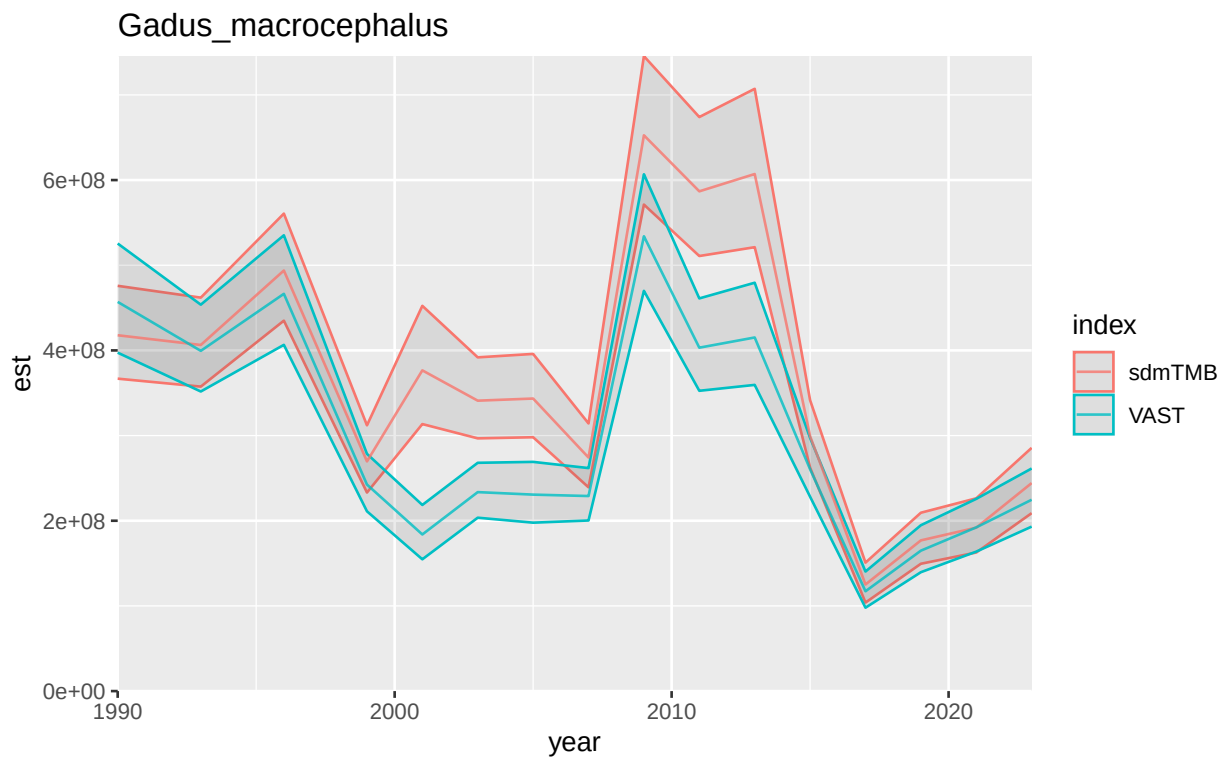
We can compare the index we would get using sdmTMB.

```
# prep prediction grid and transform to UTM projection
grid_ll <- as.data.frame(input_grid)
names(grid_ll) <- tolower(names(grid_ll))
coordinates(grid_ll) <- ~ lon + lat
proj4string(grid_ll) <- CRS("+proj=longlat +datum=WGS84")
grid <- as.data.frame(spTransform(grid_ll, CRS("+proj=utm +zone=3")))

# rename and scale to km so values don't get too large
grid$X <- grid$coords.x1 / 1000
grid$Y <- grid$coords.x2 / 1000

# or with sf:
# grid_ll <- sf::st_as_sf(
#   x = grid_ll,
#   coords = c("lon", "lat"),
#   crs = "+proj=longlat +datum=WGS84"
# )
# grid <- sf::st_transform(grid_ll, crs = "+proj=utm +zone=3")

# replicate extrapolation grid for each year in data
pred_grid <- replicate_df(grid, "year_f", unique(dat$year_f))
pred_grid$year <- as.integer(as.character(factor(pred_grid$year_f)))

# make predictions and get index
f2 <- here("species_specific_code", "GOA", species,
           "index_comparison", "predictions.RDS")
if (!file.exists(f2)) {
```

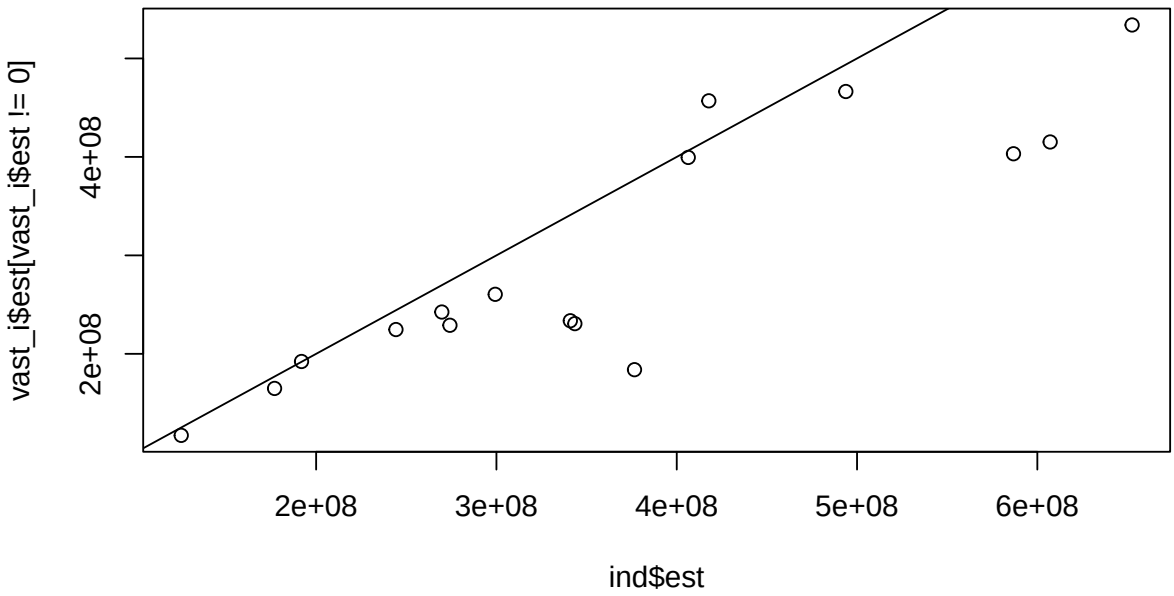
```
p <- predict(fit_sdmTMB, newdata = pred_grid, return_tmb_object = TRUE)
saveRDS(p, file = here("species_specific_code", "GOA", species, "index_comparison", "predictions.RDS"))
} else {
p <- readRDS(f2)
}

f3 <- here("species_specific_code", "GOA", species,
           "index_comparison", "index.RDS")
if (!file.exists(f3)) {
ind <- get_index(p, bias_correct = TRUE, area = p$data$area_km2)
saveRDS(ind, file = here("species_specific_code", "GOA", species, "index_comparison", "index.RDS"))
} else {
ind <- readRDS(f3)
}
```

Now, we can compare the indices.

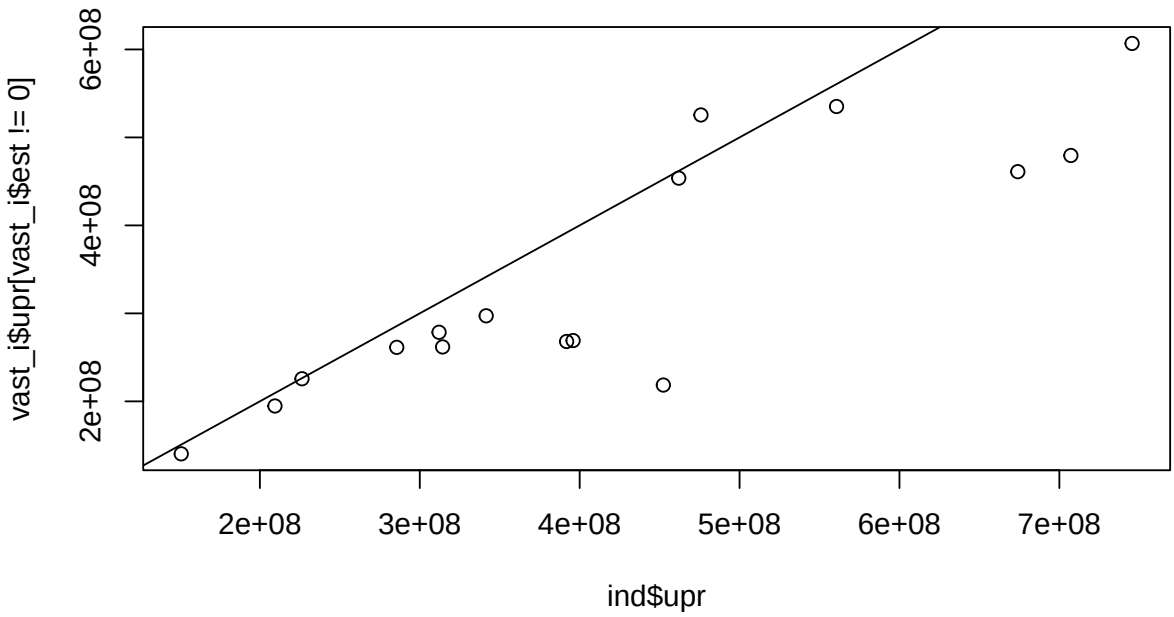
```
vast_i <- read.csv(here("species_specific_code", "GOA", species, "index_comparison", "Index.csv")) %>%
  mutate(index = "VAST", year = as.numeric(Time), est = Estimate,
    se = Std..Error.for.ln.Estimate.) %>%
  select(index, year, est, se) %>%
  mutate(lwr = exp(log(est) + qnorm(0.025) * se)) %>%
  mutate(upr = exp(log(est) + qnorm(0.975) * se))
sdm_i <- ind %>% mutate(index = "sdmTMB")
both_i <- bind_rows(sdm_i, vast_i) %>% filter(est > 0)

ggplot(both_i, aes(x = year, y = est, ymin = lwr, ymax = upr, colour = index)) +
  geom_ribbon(alpha = 0.1) +
  geom_line(alpha = 0.8) +
  ylim(0, max(both_i$upr)) +
  ggtitle(species) +
  coord_cartesian(expand = FALSE)
```



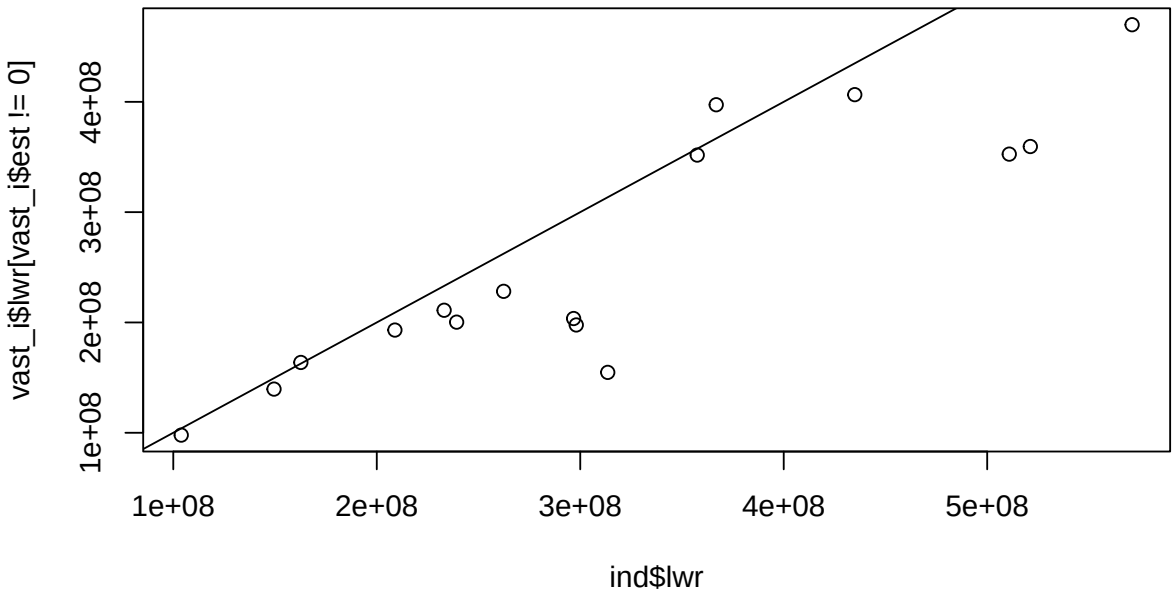

```
plot(ind$est, vast_i$est[vast_i$est != 0]);abline(0, 1)
```




```
plot(ind$upr, vast_i$upr[vast_i$est != 0]);abline(0, 1)
```




```
plot(ind$lwr, vast_i$lwr[vast_i$est != 0]);abline(0, 1)
```




```
(ind$est - vast_i$est[vast_i$est != 0]) / vast_i$est[vast_i$est != 0]
```

```
#>  [1] -0.085734329  0.017272839  0.058654506  0.112355963  1.047740498
#>  [6]  0.459709390  0.489071685  0.197380848  0.222014407  0.455317083
#> [11]  0.462295665  0.149277170  0.068018618  0.073242851 -0.001760198
#> [16]  0.087218410
(ind$upr - vast_i$upr[vast_i$est != 0]) / vast_i$upr[vast_i$est != 0]
#>  [1] -0.094540207  0.018210021  0.047595785  0.120388830  1.069702237
#>  [6]  0.461805388  0.471098394  0.200241187  0.228529651  0.461869089
#> [11]  0.474932887  0.148961352  0.074550092  0.075410053  0.002873784
#> [16]  0.092665826
(ind$lwr - vast_i$lwr[vast_i$est != 0]) / vast_i$lwr[vast_i$est != 0]
#>  [1] -0.076842811  0.016336520  0.069829966  0.104380689  1.026011797
#>  [6]  0.457616396  0.507264567  0.194527326  0.215533716  0.448794443
#> [11]  0.449766718  0.149593075  0.061526844  0.071080016 -0.006372767
#> [16]  0.081798152
```

This document was built using:

```
R.Version()$version.string
#> [1] "R version 4.3.0 (2023-04-21 ucrt)"
packageVersion("VAST")
#> [1] '3.11.2'
packageVersion("FishStatsUtils")
#> [1] '2.13.1'
```