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 <- "Sebastes_polyspinis" #Gadus_macrocephalus Sebastes_alutus Sebastes_polyspinis Sebastes_vari
```

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, "production",
                       "data", paste0("Data_Geostat_", species, ".rds")))

dat_ll <-  dplyr::transmute(dat_ll,
                            cpue_kg_km2 = Catch_KG / AreaSwept_km2,
                            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 5.

```
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 = NA, # detects automatically
  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)
}
#> Warning in .local(x, logarithm, ...): the default value of argument 'sqrt' of
#> method 'determinant(<CHMfactor>, <logical>)' may change from TRUE to FALSE as
#> soon as the next release of Matrix; set 'sqrt' when programming
#> Maximum absolute gradient of 1.93e-07: 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      0.5721852  -Inf  0.5721895   Inf  -1.981853e-09
```

```
#> 2    ln_H_input      0.1176879  -Inf  0.1176928   Inf  -1.434646e-09
#> 3      beta1_ft     -0.5044709  -Inf -0.5044855   Inf   8.378613e-10
#> 4      beta1_ft     -0.5313257  -Inf -0.5313267   Inf  -2.457661e-10
#> 5      beta1_ft     -0.2631710  -Inf -0.2631670   Inf  -3.059668e-10
#> 6      beta1_ft     -0.1343462  -Inf -0.1343374   Inf  -1.769873e-10
#> 7      beta1_ft     -0.1961073  -Inf -0.1960960   Inf   1.026290e-11
#> 8      beta1_ft     -0.2665193  -Inf -0.2665204   Inf  -7.456791e-11
#> 9      beta1_ft     -0.1173524  -Inf -0.1173542   Inf  -1.487592e-10
#> 10     beta1_ft     -0.4327071  -Inf -0.4326989   Inf  -2.883649e-10
#> 11     beta1_ft     -0.5017272  -Inf -0.5017233   Inf  -1.166125e-10
#> 12     beta1_ft     -0.5742793  -Inf -0.5742759   Inf  -1.970397e-10
#> 13     beta1_ft     -0.4942690  -Inf -0.4942708   Inf  -1.153717e-10
#> 14     beta1_ft     -0.7926880  -Inf -0.7926908   Inf  -7.689849e-11
#> 15     beta1_ft     -0.3001855  -Inf -0.3001972   Inf   4.669847e-10
#> 16     beta1_ft     -0.5206570  -Inf -0.5206497   Inf  -7.161205e-11
#> 17     beta1_ft     -0.6424059  -Inf -0.6424096   Inf  -9.841017e-13
#> 18     beta1_ft     -0.9173144  -Inf -0.9173170   Inf  -8.177548e-11
#> 19   L_omega1_z      2.5825097  -Inf  2.5825167   Inf  -1.971273e-09
#> 20 L_epsilon1_z      0.4405871  -Inf  0.4405863   Inf  -1.492986e-08
#> 21    logkappa1     -3.7040392  -Inf -3.7040468   Inf   5.228884e-09
#> 22     beta2_ft      1.8006812  -Inf  1.8006750   Inf   1.858957e-10
#> 23     beta2_ft      1.4748041  -Inf  1.4748286   Inf  -4.932303e-10
#> 24     beta2_ft      1.8290249  -Inf  1.8290449   Inf  -4.904734e-10
#> 25     beta2_ft      1.2945370  -Inf  1.2945294   Inf   8.540013e-11
#> 26     beta2_ft      2.1621478  -Inf  2.1621189   Inf   4.192700e-10
#> 27     beta2_ft      1.3100369  -Inf  1.3100283   Inf   1.695746e-10
#> 28     beta2_ft      1.9213010  -Inf  1.9213069   Inf  -1.275993e-10
#> 29     beta2_ft      1.8774797  -Inf  1.8774654   Inf   1.768363e-10
#> 30     beta2_ft      1.4751018  -Inf  1.4750725   Inf   4.479297e-10
#> 31     beta2_ft      1.5871041  -Inf  1.5871089   Inf  -1.020251e-10
#> 32     beta2_ft      2.1917282  -Inf  2.1917379   Inf  -1.558007e-10
#> 33     beta2_ft      1.4986774  -Inf  1.4986768   Inf   1.538680e-11
#> 34     beta2_ft      1.6215733  -Inf  1.6215752   Inf  -1.940137e-11
#> 35     beta2_ft      1.5505635  -Inf  1.5505565   Inf   5.026379e-11
#> 36     beta2_ft      1.4437080  -Inf  1.4437097   Inf  -3.759482e-11
#> 37     beta2_ft      1.1580751  -Inf  1.1580906   Inf  -2.590301e-10
#> 38   L_omega2_z      1.9734047  -Inf  1.9734067   Inf  -1.052079e-08
#> 39 L_epsilon2_z      1.7639187  -Inf  1.7639229   Inf  -6.643020e-09
#> 40    logkappa2     -2.7134917  -Inf -2.7134908   Inf  -5.832128e-09
#> 41    logSigmaM      0.2392693  -Inf  0.2392668   Inf   6.796128e-09
```

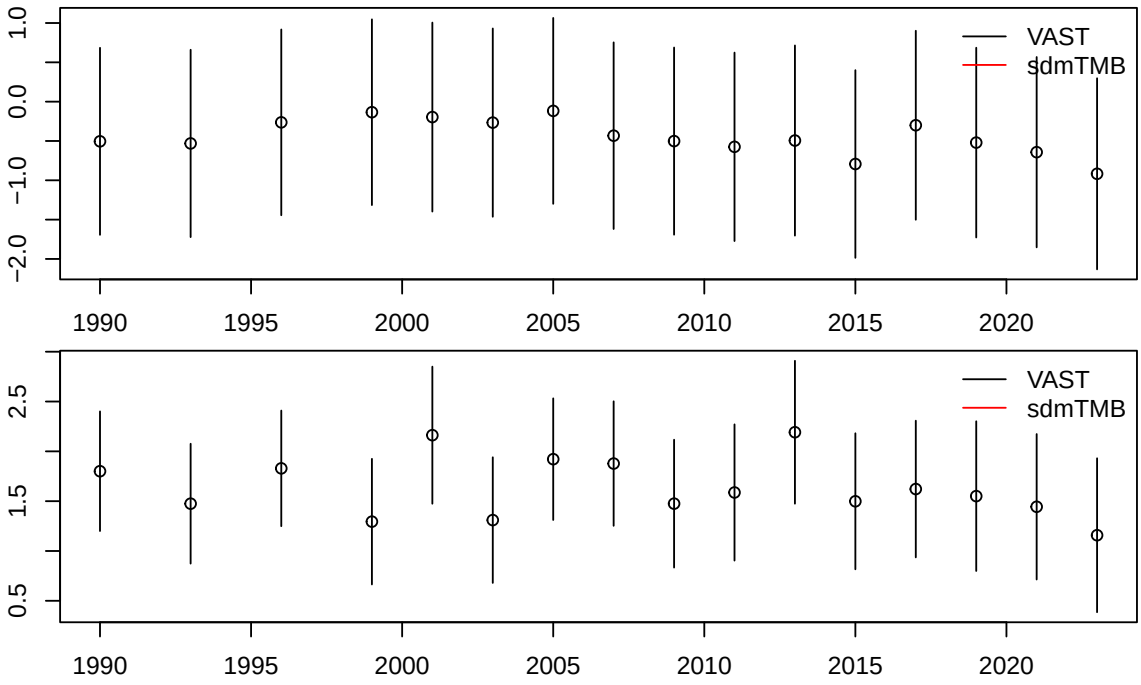
Now we fit the same model in sdmTMB:

```
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=5")))
# 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")
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)
}
#> attempting to improve convergence with optimHess
#> running TMB sdreport

# 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:

```
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)
```



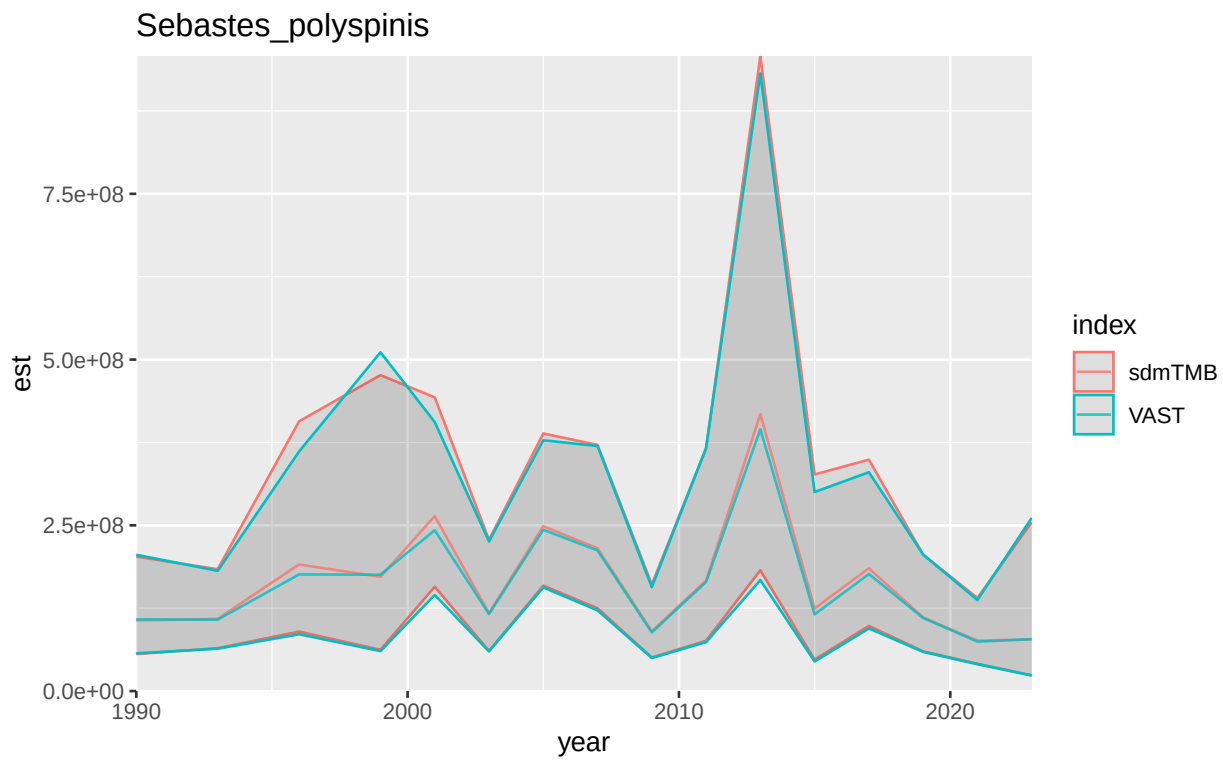
We can compare the index we would get using sdmTMB.

```
# 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=5")))

# 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=5")

# 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)) {
```

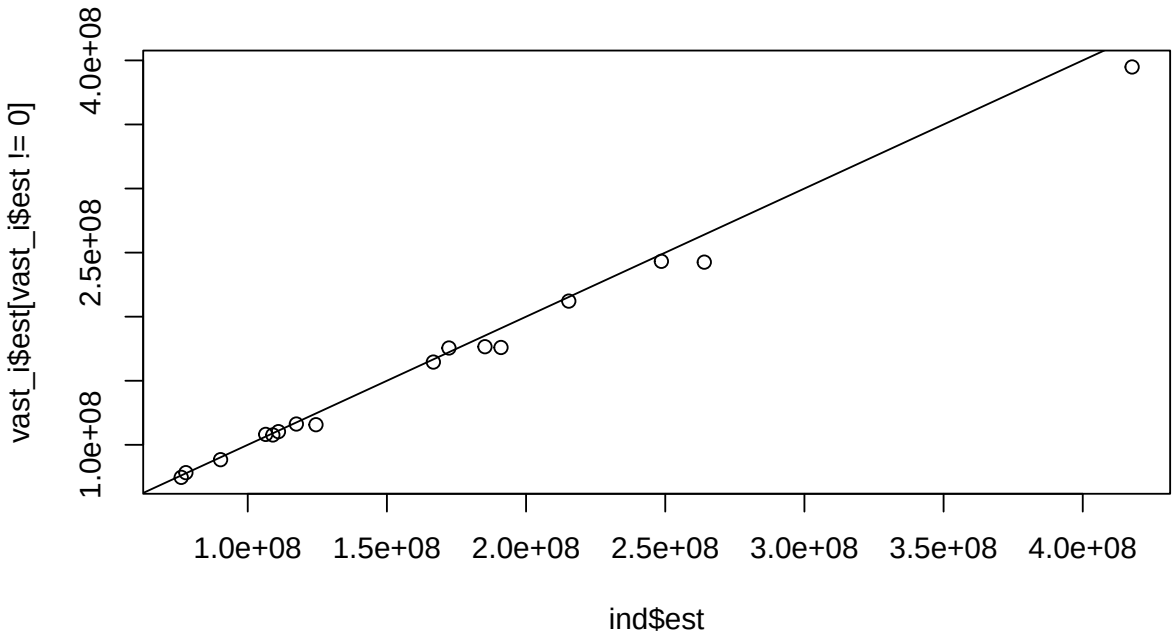
```
p <- predict(fit_sdmTMB, newdata = pred_grid, return_tmb_object = TRUE)
saveRDS(p, file = here("species_specific_code", "GOA", species, "index_comparison", "predictions.RDS"))
} else {
p <- readRDS(f2)
}

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.

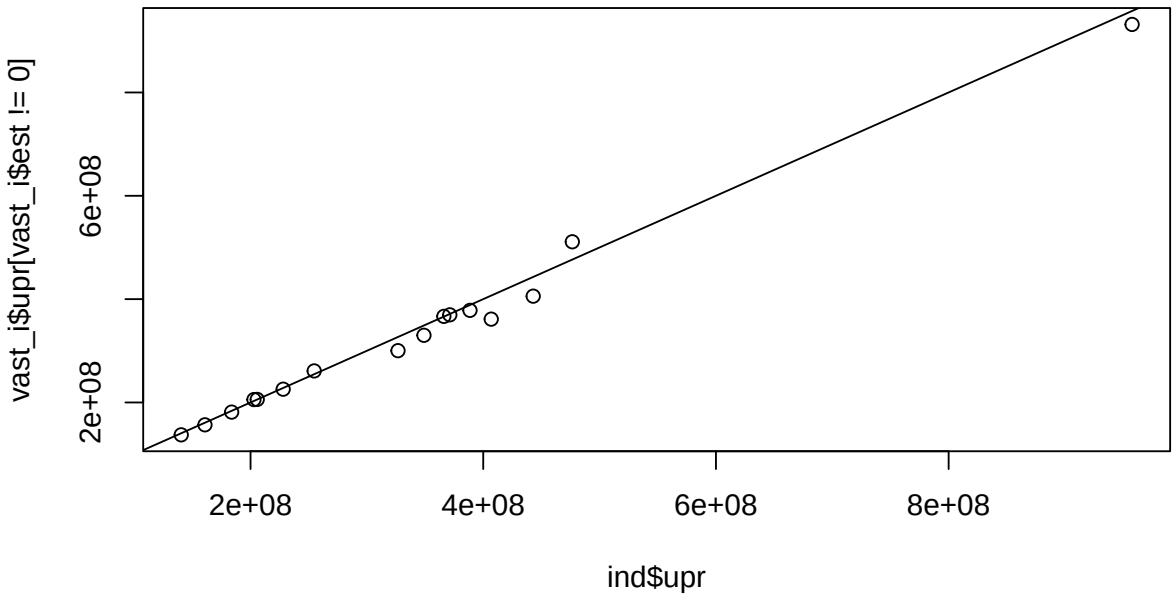
```
sdm_i <- ind %>% mutate(index = "sdmTMB")
vast_i <- read.csv(here("species_specific_code", "GOA", species, "index_comparison", "Index.csv")) %>%
  mutate(index = "VAST", year = as.numeric(Time), est = Estimate,
    se = Std..Error.for.ln.Estimate.) %>%
  select(index, year, est, se) %>%
  filter(year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



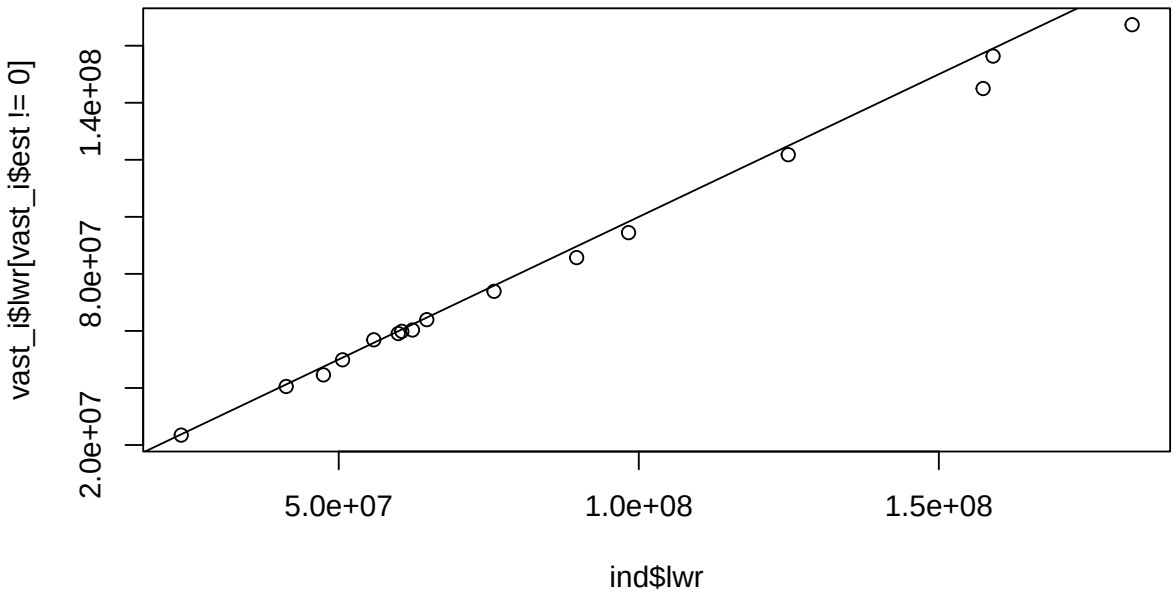

```
plot(ind$est, vast_i$est[vast_i$est != 0]);abline(0, 1)
```




```
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] -0.015627015  0.012290839  0.085358872 -0.018666149  0.088545004
#>  [6]  0.010567492  0.022176771  0.014974138  0.020629205  0.012733132
#> [11]  0.057871493  0.076182992  0.049038821  0.006670591  0.020006325
#> [16] -0.006419354
(ind$upr - vast_i$upr[vast_i$est != 0]) / vast_i$upr[vast_i$est != 0]
#>  [1] -0.0129666627  0.0133242289  0.1261751443 -0.0674587313  0.0916240426
#>  [6]  0.0104523385  0.0272361987  0.0044101572  0.0259050065 -0.0013319062
#> [11]  0.0279480041  0.0881737502  0.0575832984 -0.0006883773  0.0222929363
#> [16] -0.0244112876
(ind$lwr - vast_i$lwr[vast_i$est != 0]) / vast_i$lwr[vast_i$est != 0]
#>  [1] -0.01828020  0.01125850  0.04602191  0.03267937  0.08547465  0.01068266
#>  [7]  0.01714226  0.02564923  0.01538054  0.02699626  0.08866605  0.06432436
#> [13]  0.04056338  0.01408375  0.01772483  0.01190439
```

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'
```