Comparing VAST and sdmTMB Bering indices

Contents

```r
#pak::pak("pbs-assess/sdmTMB")
#pak::pak("pbs-assess/sdmTMB@index-split")
#pak::pak("afsc-gap-products/coldpool")
library(VAST)
library(sp)
library(sdmTMB)
library(dplyr)
library(ggplot2)
library(here)

species <- "pollock"
#pollock pacific_cod yellowfin_sole
```

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 EBS/NBS AFSC GAP bottom trawl surveys. The density units we will work in are either kg/km^2 or n/km^2, for biomass or numerical abundance.

```r
# TODO: standardize names better
if(species == "pacific_cod"){
  dat_ll <- readRDS(here("species_specific_code", "BS", species, "production",
                        "data", "data_geostat_numerical_index.RDS"))
  dat_ll <-  dplyr::transmute(dat_ll,
                         cpue = Catch_N / AreaSwept_km2, #note last cod model used catch ~ effort
                         year = as.integer(Year),
                         vessel = "missing",
                         effort = 1, # area swept is 1 when using CPUE instead of observed weight
                         lat = Lat,
                         lon = Lon,
                         pass = 0) %>%
                as.data.frame() # ensure not a tibble
}
if(species == "pollock"){
  dat_ll <- read.csv(here("species_specific_code", "BS", species, "VAST_ddc_all_2023.csv"))
  dat_ll <-  dplyr::transmute(dat_ll,
                         cpue = ddc_cpue_kg_ha * 100, # converts cpue from kg/ha to kg/km^2
                         year = as.integer(year),
                         vessel = "missing",
                         effort = 1, # area swept is 1 when using CPUE instead of observed weight
                         lat = start_latitude,
                         lon = start_longitude,
                         pass = 0) %>%
                as.data.frame() # ensure not a tibble
```

```r
}
if(species == "yellowfin_sole"){
  dat_ll <- readRDS(here("species_specific_code", "BS", species, "production",
                         "data", "data_geostat_biomass_index.RDS"))
  dat_ll <-  dplyr::transmute(dat_ll,
                              cpue = Catch_KG / AreaSwept_km2,
                              year = as.integer(Year),
                              vessel = "missing",
                              effort = 1, # area swept is 1 when using CPUE instead of observed weight
                              lat = Lat,
                              lon = Lon,
                              pass = 0) %>%
                    as.data.frame() # ensure not a tibble
}
```

We also need to pull in the appropriate environmental covariate as this is used as a spatially varying covariate in AFSC GAP Bering Sea indices. For yellowfin sole, this is the mean bottom temperature in waters less than 100m. For other species, this is the cold pool extent. The cold pool is defined here as the areal extent (in km^2) of seawater equal to or colder than 2 degrees Celsius near the seafloor, calculated from observations from the AFSC GAP EBS/NBS bottom trawl survey. We center and scale this for inclusion as a covariate.

```r
if(species == "yellowfin_sole"){
  env <- scale(coldpool:::cold_pool_index$MEAN_BT_LT100M)
} else {
  env <- scale(coldpool:::cold_pool_index$AREA_LTE2_KM2)
}

# TODO: drop below inclusion of missing year for sdmTMB? Or remove extra_year argument in sdmTMB()
covariate_data <- data.frame(Year = as.integer(c(coldpool:::cold_pool_index$YEAR, 2020)),
                             Lat = mean(dat_ll$lat),
                             Lon = mean(dat_ll$lon),
                             env = c(env, 0)) %>%
  filter(Year != 2024)

# TODO: replace "select(covariate_data, year, env)" with just "env" for sdmTMB
env_join <- covariate_data %>% select(year = Year, env)
dat_ll <- left_join(dat_ll, env_join, by = "year")
```

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 specified partially by specifying `purpose = "index2"` but we also explicitly provide arguments for these and other key settings here.

```r
FieldConfig <- c("Omega1"="IID", "Epsilon1"="IID", "Omega2"="IID", "Epsilon2"="IID")
RhoConfig <- c("Beta1" = 0, "Beta2" = 0, "Epsilon1" = 4, "Epsilon2" = 4)
OverdispersionConfig <- c("Eta1"=0, "Eta2"=0)
```

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 2.

```r
if(species == "pacific_cod"){
  ObsModel = c(2, 4)
} else {
  ObsModel = c(2, 1)
}

settings <- make_settings(
  n_x = 750, # number of vertices in the SPDE mesh
  Region = c("Eastern_Bering_Sea", "Northern_Bering_Sea"),
  purpose = "index2", # index of abundance with Gamma for positive catches
  fine_scale = TRUE, # use bilinear interpolation from the INLA 'A' matrix
  zone = 2,
  FieldConfig = FieldConfig,
  RhoConfig = RhoConfig,
  OverdispersionConfig = c("Eta1" = 0, "Eta2" = 0),
  Options = c("Calculate_Range" = TRUE, "Calculate_effective_area" = TRUE,
              "treat_nonencounter_as_zero" = FALSE),
  ObsModel = ObsModel, #delta-Gamma; (2,4) if there are years with 100% encounter rate; (10, 2) for Twe
  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'))
)
```

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

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

f <- here("species_specific_code", "BS", species, "index_comparison", "VASTfit_full.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"],
    a_i = dat_ll[, "effort"],
    create_strata_per_region = TRUE,
    getJointPrecision = TRUE,
    getReportCovariance = TRUE,
    X1_formula = ~ env,
    X2_formula = ~ env,
    X1config_cp = as.matrix(2),
    X2config_cp = as.matrix(2),
    covariate_data = covariate_data,
    working_dir = paste0(here("species_specific_code", "BS", species, "index_comparison"), "/")
  )
  saveRDS(fit, file = f)
} else {
  fit <- readRDS(f)
  fit <- reload_model(fit)
```

```
}
#> Maximum absolute gradient of 0.000104: 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
#> 1      ln_H_input     0.09368142 -5.000000  0.09367727  5.000000
#> 2      ln_H_input    -0.42203177 -5.000000 -0.42203033  5.000000
#> 3        beta1_ft     0.30170208      -Inf  0.30166625       Inf
#> 4        beta1_ft     0.33837843      -Inf  0.33835976       Inf
#> 5        beta1_ft     0.43131191      -Inf  0.43128063       Inf
#> 6        beta1_ft     0.60580346      -Inf  0.60583287       Inf
#> 7        beta1_ft     1.16796996      -Inf  1.16801190       Inf
#> 8        beta1_ft     0.17451778      -Inf  0.17455755       Inf
#> 9        beta1_ft     1.07184515      -Inf  1.07184739       Inf
#> 10       beta1_ft     0.33847038      -Inf  0.33843465       Inf
#> 11       beta1_ft     0.30101791      -Inf  0.30098903       Inf
#> 12       beta1_ft     0.71431596      -Inf  0.71428788       Inf
#> 13       beta1_ft     0.58690508      -Inf  0.58684931       Inf
#> 14       beta1_ft     0.73834071      -Inf  0.73834665       Inf
#> 15       beta1_ft     1.18335522      -Inf  1.18328263       Inf
#> 16       beta1_ft     0.90190133      -Inf  0.90185533       Inf
#> 17       beta1_ft     0.91692073      -Inf  0.91683705       Inf
#> 18       beta1_ft     0.88081861      -Inf  0.88078865       Inf
#> 19       beta1_ft     1.04015840      -Inf  1.04012640       Inf
#> 20       beta1_ft     1.88233789      -Inf  1.88233009       Inf
#> 21       beta1_ft     1.31442606      -Inf  1.31441054       Inf
#> 22       beta1_ft     1.49167771      -Inf  1.49159599       Inf
#> 23       beta1_ft     1.06306471      -Inf  1.06301635       Inf
#> 24       beta1_ft     1.06163304      -Inf  1.06160466       Inf
#> 25       beta1_ft     1.51991178      -Inf  1.51992544       Inf
#> 26       beta1_ft     1.45497894      -Inf  1.45495691       Inf
#> 27       beta1_ft     1.24496417      -Inf  1.24491544       Inf
#> 28       beta1_ft     1.08817526      -Inf  1.08810577       Inf
#> 29       beta1_ft     0.57621930      -Inf  0.57623016       Inf
#> 30       beta1_ft     0.81126198      -Inf  0.81121881       Inf
#> 31       beta1_ft     0.68710040      -Inf  0.68704750       Inf
#> 32       beta1_ft     1.13393587      -Inf  1.13381313       Inf
#> 33       beta1_ft     1.53298882      -Inf  1.53295582       Inf
#> 34       beta1_ft     1.54829312      -Inf  1.54825073       Inf
#> 35       beta1_ft     1.92310470      -Inf  1.92306032       Inf
#> 36       beta1_ft     2.14952619      -Inf  2.14949117       Inf
#> 37       beta1_ft     2.07561957      -Inf  2.07556434       Inf
#> 38       beta1_ft     2.64423178      -Inf  2.64416488       Inf
#> 39       beta1_ft     0.73392039      -Inf  0.73379074       Inf
#> 40       beta1_ft     2.87185531      -Inf  2.87177500       Inf
#> 41       beta1_ft     3.07717118      -Inf  3.07709067       Inf
#> 42       beta1_ft     3.02883344      -Inf  3.02873850       Inf
#> 43       beta1_ft     2.86211219      -Inf  2.86198114       Inf
#> 44      L_omega1_z     1.38198578      -Inf  1.38201211       Inf
#> 45    L_epsilon1_z     0.37783132      -Inf  0.37782607       Inf
#> 46       logkappa1    -4.82305167 -6.061886 -4.82306045 -1.727504
```

```
#> 47  Epsilon_rho1_f     0.93146056 -0.990000  0.93146173  0.990000
#> 48 log_sigmaXi1_cp    -1.46067286      -Inf -1.46068017       Inf
#> 49         beta2_ft     5.49662471      -Inf  5.49669840       Inf
#> 50         beta2_ft     6.62714780      -Inf  6.62711580       Inf
#> 51         beta2_ft     5.70636508      -Inf  5.70643707       Inf
#> 52         beta2_ft     6.11019670      -Inf  6.11012671       Inf
#> 53         beta2_ft     5.29443585      -Inf  5.29453030       Inf
#> 54         beta2_ft     6.54880961      -Inf  6.54878020       Inf
#> 55         beta2_ft     5.71517581      -Inf  5.71531871       Inf
#> 56         beta2_ft     6.24684678      -Inf  6.24685041       Inf
#> 57         beta2_ft     5.98715461      -Inf  5.98709012       Inf
#> 58         beta2_ft     5.54440505      -Inf  5.54435104       Inf
#> 59         beta2_ft     5.59688654      -Inf  5.59695951       Inf
#> 60         beta2_ft     5.81474017      -Inf  5.81469664       Inf
#> 61         beta2_ft     5.19002529      -Inf  5.19005247       Inf
#> 62         beta2_ft     4.98345615      -Inf  4.98352657       Inf
#> 63         beta2_ft     5.02872993      -Inf  5.02870264       Inf
#> 64         beta2_ft     5.18775794      -Inf  5.18775123       Inf
#> 65         beta2_ft     5.14716603      -Inf  5.14709366       Inf
#> 66         beta2_ft     4.21573331      -Inf  4.21582454       Inf
#> 67         beta2_ft     5.41428700      -Inf  5.41430972       Inf
#> 68         beta2_ft     5.17132879      -Inf  5.17129896       Inf
#> 69         beta2_ft     5.45264196      -Inf  5.45264700       Inf
#> 70         beta2_ft     6.13038868      -Inf  6.13039487       Inf
#> 71         beta2_ft     5.19635151      -Inf  5.19633898       Inf
#> 72         beta2_ft     5.15421130      -Inf  5.15418014       Inf
#> 73         beta2_ft     4.32435436      -Inf  4.32430566       Inf
#> 74         beta2_ft     4.68618410      -Inf  4.68621148       Inf
#> 75         beta2_ft     4.69369581      -Inf  4.69370105       Inf
#> 76         beta2_ft     3.93204227      -Inf  3.93213920       Inf
#> 77         beta2_ft     4.76349572      -Inf  4.76348451       Inf
#> 78         beta2_ft     5.23854428      -Inf  5.23856349       Inf
#> 79         beta2_ft     4.80807912      -Inf  4.80803826       Inf
#> 80         beta2_ft     5.00598563      -Inf  5.00607695       Inf
#> 81         beta2_ft     5.74069395      -Inf  5.74073378       Inf
#> 82         beta2_ft     5.97492669      -Inf  5.97490182       Inf
#> 83         beta2_ft     5.90283332      -Inf  5.90279768       Inf
#> 84         beta2_ft     5.61851765      -Inf  5.61848539       Inf
#> 85         beta2_ft     6.36602488      -Inf  6.36602475       Inf
#> 86         beta2_ft     4.96082347      -Inf  4.96076687       Inf
#> 87         beta2_ft     4.26241346      -Inf  4.26242876       Inf
#> 88         beta2_ft     4.40867773      -Inf  4.40872612       Inf
#> 89         beta2_ft     4.56072500      -Inf  4.56077918       Inf
#> 90       L_omega2_z     1.03678604      -Inf  1.03679459       Inf
#> 91     L_epsilon2_z     1.17438212      -Inf  1.17438731       Inf
#> 92        logkappa2    -4.03459907 -6.061886 -4.03459935 -1.727504
#> 93  Epsilon_rho2_f     0.26162598 -0.990000  0.26162559  0.990000
#> 94 log_sigmaXi2_cp    -1.15280394      -Inf -1.15281538       Inf
#> 95        logSigmaM     0.01717104      -Inf  0.01717057 10.000000
#>     final_gradient
#> 1    -1.812314e-08
#> 2    -4.622542e-09
#> 3    -1.168168e-09
#> 4     2.098403e-09
```

```
#> 5   -2.971156e-09
#> 6    8.811512e-10
#> 7    3.450751e-10
#> 8    4.521610e-09
#> 9   -5.657398e-09
#> 10   8.128609e-10
#> 11   2.008676e-09
#> 12   1.422357e-09
#> 13  -2.094936e-09
#> 14  -9.464358e-10
#> 15  -1.552692e-09
#> 16   1.022684e-10
#> 17  -5.217586e-10
#> 18   1.797517e-09
#> 19   4.506475e-10
#> 20  -6.026966e-10
#> 21   9.189236e-10
#> 22  -1.752824e-09
#> 23   1.164224e-10
#> 24   1.124270e-09
#> 25  -8.912266e-10
#> 26   1.258861e-09
#> 27   1.079748e-09
#> 28  -3.730875e-09
#> 29   4.253572e-09
#> 30  -1.263146e-09
#> 31   2.900343e-09
#> 32  -4.801016e-09
#> 33   1.697714e-09
#> 34  -2.410872e-11
#> 35   9.374830e-10
#> 36  -1.065551e-09
#> 37   8.023164e-10
#> 38   1.262478e-09
#> 39   2.889990e-10
#> 40   6.623253e-10
#> 41   1.347104e-09
#> 42   9.092957e-10
#> 43  -2.388148e-09
#> 44  -2.784986e-08
#> 45  -9.013840e-07
#> 46   1.948304e-07
#> 47  -1.229346e-06
#> 48  -1.787330e-08
#> 49  -1.851923e-09
#> 50  -4.672458e-10
#> 51  -4.904237e-10
#> 52   1.546539e-09
#> 53  -1.003976e-09
#> 54  -1.535341e-09
#> 55  -3.762338e-09
#> 56  -1.164679e-09
#> 57   1.955662e-09
#> 58   1.543825e-09
```

```
#> 59   3.885248e-11
#> 60   1.684775e-09
#> 61   9.194139e-10
#> 62  -6.211138e-10
#> 63  -1.986464e-10
#> 64   6.174403e-10
#> 65   7.163194e-10
#> 66  -6.499334e-11
#> 67  -3.723386e-10
#> 68   1.055781e-09
#> 69  -6.222365e-10
#> 70  -1.677499e-09
#> 71  -3.223661e-10
#> 72   1.750635e-10
#> 73   1.775490e-09
#> 74   1.045059e-09
#> 75   3.942375e-10
#> 76  -1.850673e-09
#> 77   2.237854e-10
#> 78   7.650129e-10
#> 79   2.324391e-09
#> 80  -1.731564e-09
#> 81  -2.106617e-09
#> 82   1.859917e-10
#> 83   2.181295e-10
#> 84   1.215398e-09
#> 85  -1.067029e-09
#> 86   9.525536e-10
#> 87  -7.673719e-10
#> 88   1.760299e-10
#> 89   2.521247e-09
#> 90  -1.789113e-08
#> 91  -5.245446e-07
#> 92   5.909958e-07
#> 93   1.845728e-07
#> 94  -3.279697e-08
#> 95  -6.009969e-08
```

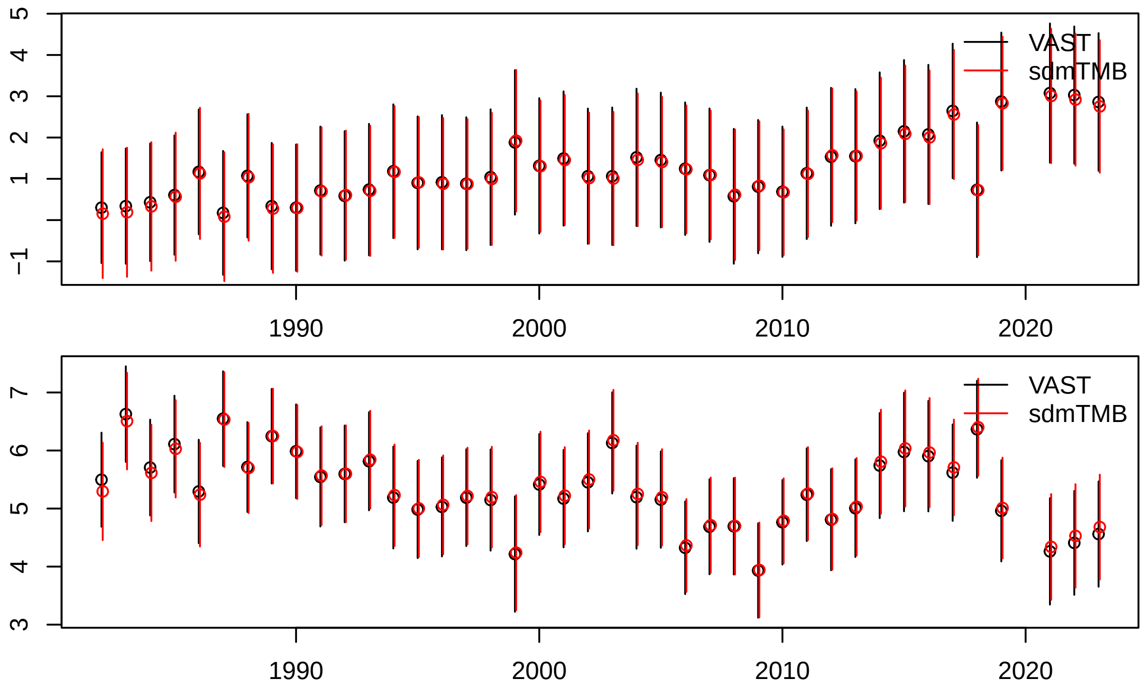
Now we fit the same model in sdmTMB:

```
dat <- dat_ll %>%
  rename(X = lon, Y = lat) #%>% filter(year != 2020) #drop dummy 2020 data

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=2")))
# 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", "BS", species, "index_comparison", "fit_sdmTMB.RDS")
if (!file.exists(f1)) {
```

```r
  # make mesh and fit model
  mesh <-  make_mesh(dat, xy_cols = c("X", "Y"), mesh = fit$spatial_list$MeshList$anisotropic_mesh) #pa
  #mesh <-  make_mesh(dat, xy_cols = c("X", "Y"), n_knots = 50, type = "kmeans") #coarser mesh for expe

  fit_sdmTMB <- sdmTMB(
    cpue ~ 0 + year_f,
    spatial_varying = ~ env,
    data = dat,
    mesh = mesh,
    family = delta_gamma(type = "poisson-link"),
    time = "year",
    spatial = "on",
    spatiotemporal = "ar1",
    extra_time = 2020L, #omit if dummy 2020 included in data
    silent = FALSE,
    anisotropy = TRUE,
    do_fit = TRUE
  )
  fit_sdmTMB
  saveRDS(fit_sdmTMB, file = here("species_specific_code", "BS", species, "index_comparison", "fit_sdmT
} 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)
```



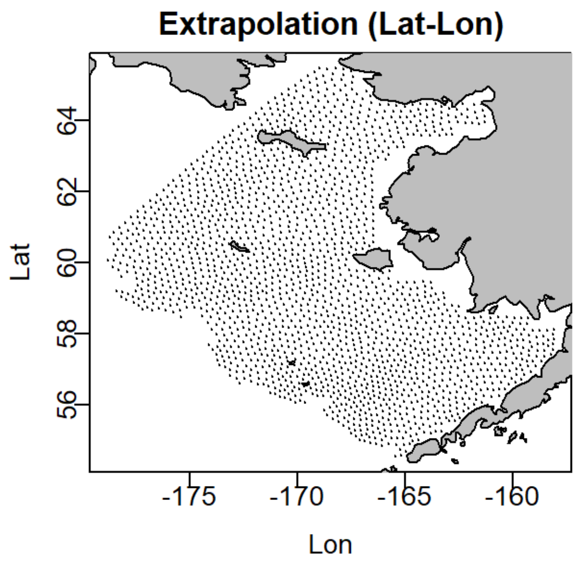

```
rm(fit)
```

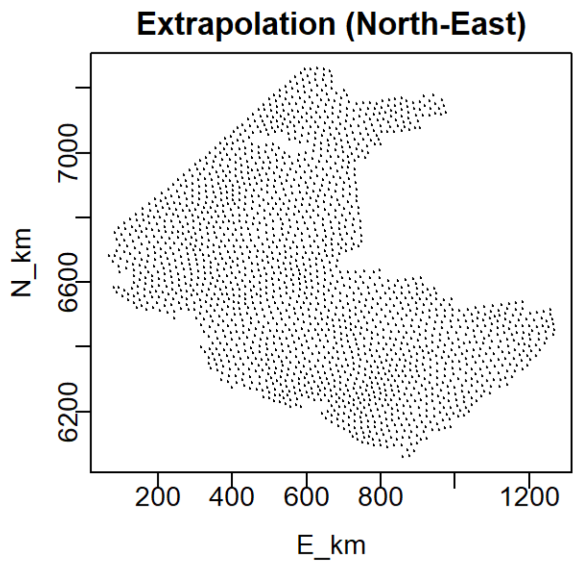
While making custom plots of individual elements would require considerable additional code to extract and reformat the necessary components of each output, VAST has a wrapper function that generates the typical plots one may want. Here we stick with the default set of plots (`plot_set = 3`); however, one can specify different standard plots to make by changing the setting of this argument (see `?FishStatsUtils::plot_maps` and `?FishStatsUtils::plot_results`).

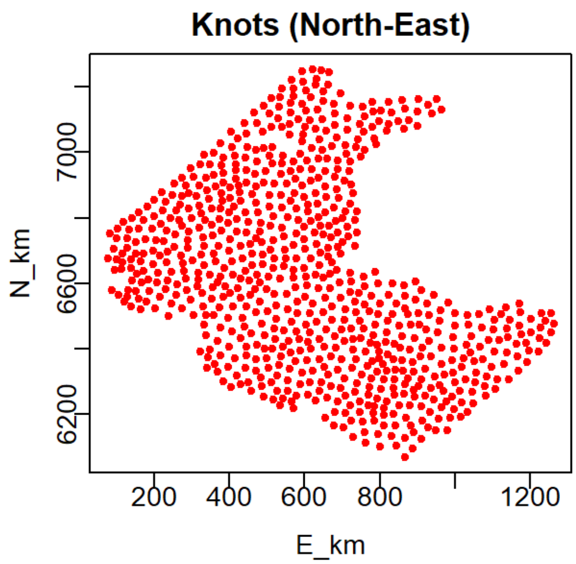
```
if(!file.exists(here("species_specific_code", "BS", species, "index_comparison", "plots", "Data_and_kno
  plot(
    fit,
    check_residuals = FALSE,
    working_dir = paste0(here("species_specific_code", "BS", species, "index_comparison", "plots"), "/")
  )
}
```

Here we will read in some key plots. We can start by looking at the location of samples and knots.

```
knitr::include_graphics(here("species_specific_code", "BS", species, "index_comparison", "plots", "Data_
```

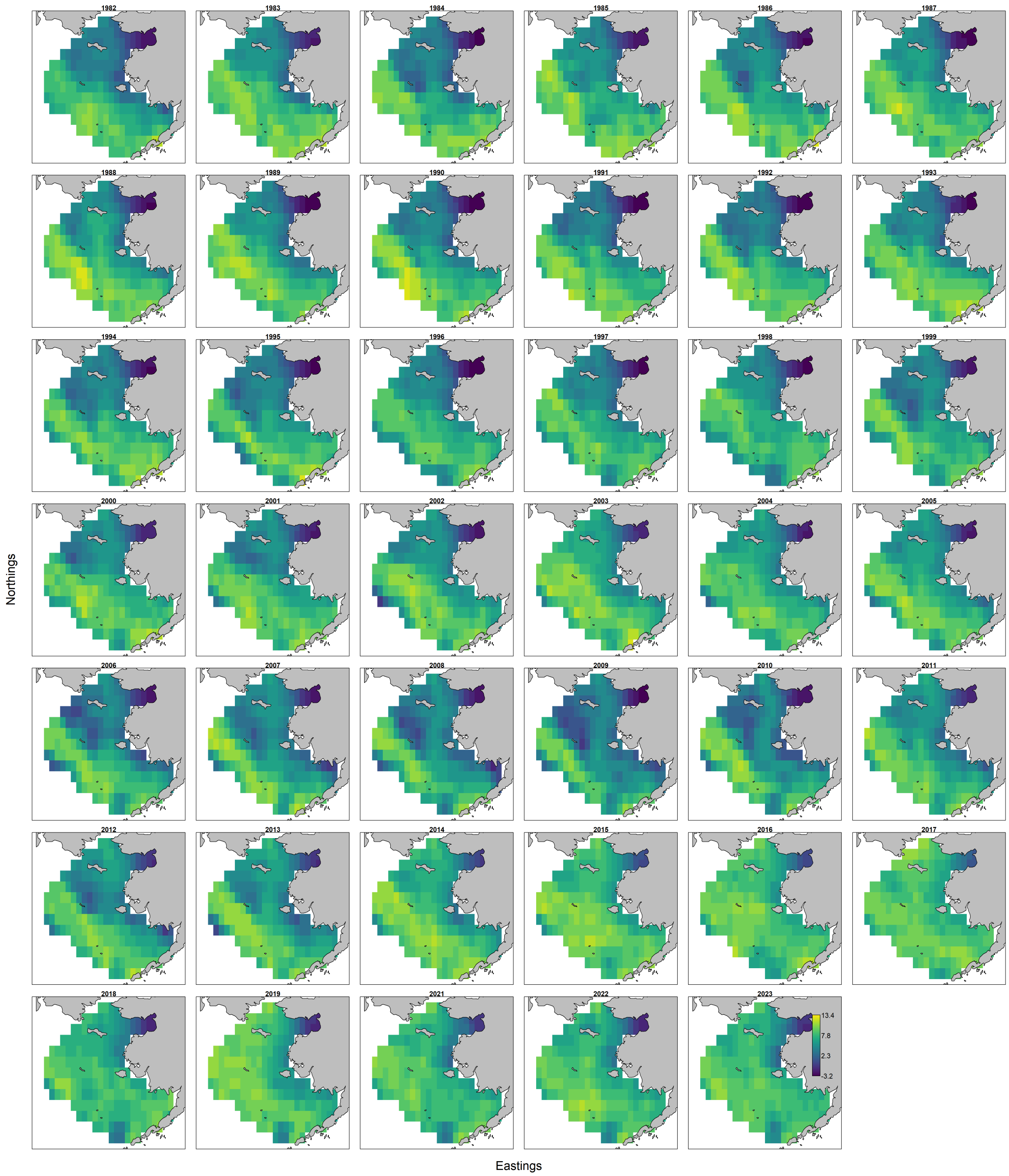






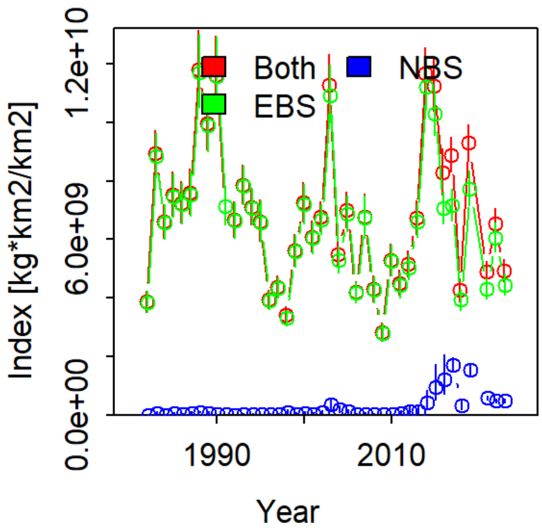
Then we can look at maps of the predicted population densities (here on the log scale).

```
knitr::include_graphics(here("species_specific_code", "BS", species, "index_comparison", "plots", "ln_d
```



And finally the index.

```
if (file.exists(here("species_specific_code", "BS", species, "index_comparison", "plots", "Index.png")))
  knitr::include_graphics(here("species_specific_code", "BS", species, "index_comparison", "plots", "In
}
```



We can compare the index we would get using sdmTMB.

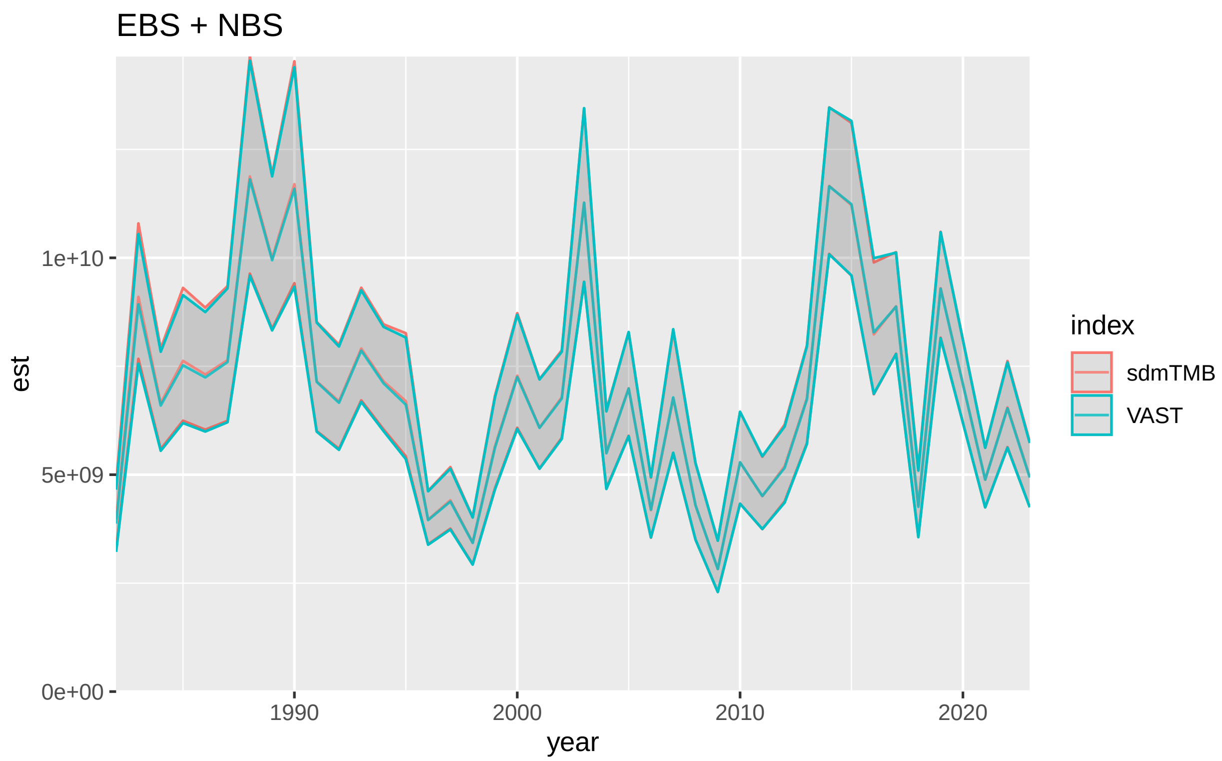
```
# TODO: save grid so it can be loaded to make prediction grid replicated for each year with covariate
# prep prediction grids (all, EBS, NBS) and transform to UTM projection
load(here("extrapolation_grids", "eastern_bering_sea_grid.rda"))
load(here("extrapolation_grids", "northern_bering_sea_grid.rda"))

# EBS grid
grid_ll_ebs <- as.data.frame(eastern_bering_sea_grid)
names(grid_ll_ebs) <- tolower(names(grid_ll_ebs))
grid_ll_ebs <- grid_ll_ebs %>%
  rename(X = lon, Y = lat)
coordinates(grid_ll_ebs) <- ~ X + Y
proj4string(grid_ll_ebs) <- CRS("+proj=longlat +datum=WGS84")
grid_ebs <- as.data.frame(spTransform(grid_ll_ebs, CRS("+proj=utm +zone=2")))
grid_ebs$X <- grid_ebs$coords.x1 / 1000 # scale to km to work with smaller numbers
grid_ebs$Y <- grid_ebs$coords.x2 / 1000

# NBS grid
grid_ll_nbs <- as.data.frame(northern_bering_sea_grid)
names(grid_ll_nbs) <- tolower(names(grid_ll_nbs))
grid_ll_nbs <- grid_ll_nbs %>%
  rename(X = lon, Y = lat)
coordinates(grid_ll_nbs) <- ~ X + Y
proj4string(grid_ll_nbs) <- CRS("+proj=longlat +datum=WGS84")
grid_nbs <- as.data.frame(spTransform(grid_ll_nbs, CRS("+proj=utm +zone=2")))
grid_nbs$X <- grid_nbs$coords.x1 / 1000 # scale to km to work with smaller numbers
grid_nbs$Y <- grid_nbs$coords.x2 / 1000

# Combined grid
grid <- bind_rows(grid_nbs, grid_ebs)

# replicate extrapolation grids for each year in data
pred_grid_ebs <- replicate_df(grid_ebs, "year_f", unique(dat$year_f))
pred_grid_nbs <- replicate_df(grid_nbs, "year_f", unique(dat$year_f))
```

```r
pred_grid <- replicate_df(grid, "year_f", unique(dat$year_f))
pred_grid_ebs$year <- as.integer(as.character(factor(pred_grid_ebs$year_f)))
pred_grid_nbs$year <- as.integer(as.character(factor(pred_grid_nbs$year_f)))
pred_grid$year <- as.integer(as.character(factor(pred_grid$year_f)))

# join in environmental covariate (cold pool or mean bottom temperature)
pred_grid_ebs <- left_join(pred_grid_ebs, rename(env_join, cpe = env), by = "year")
pred_grid_nbs <- left_join(pred_grid_nbs, rename(env_join, cpe = env), by = "year")
pred_grid <- left_join(pred_grid, rename(env_join, cpe = env), by = "year")
# TODO: update in new fits, as prior pollock model covariate was "cpe" rather than "env",
# so now rename if needed: from env_join to rename(env_join, cpe = env)

# get predictions for total area, and the two subareas of interest (EBS, NBS)
# f2 <- here("species_specific_code", "BS", species,
#             "index_comparison", "predictions.RData")
# if (!file.exists(f2)) {
#   p <- predict(fit_sdmTMB, newdata = pred_grid, return_tmb_object = TRUE)
#   p_ebs <- predict(fit_sdmTMB, newdata = pred_grid_ebs, return_tmb_object = TRUE)
#   p_nbs <- predict(fit_sdmTMB, newdata = pred_grid_nbs, return_tmb_object = TRUE)
#     save(p, p_ebs, p_nbs, file = f2)
# } else {
#   load(f2)
# }

# get indices for total area, and the two subareas of interest (EBS, NBS)
# f3 <- here("species_specific_code", "BS", species,
#             "index_comparison", "indices.RData")
# if (!file.exists(f3)) {
#   gc()
#   ind <- get_index(p, bias_correct = FALSE, area = pred_grid$area_in_survey_km2)
#   ind$stratum <- "Both"
#
#   ind_ebs <- get_index(p_ebs, bias_correct = FALSE, area = pred_grid_ebs$area_in_survey_km2)
#   ind_ebs$stratum <- "EBS"
#
#   ind_nbs <- get_index(p_nbs, bias_correct = FALSE, area = pred_grid_nbs$area_in_survey_km2)
#   ind_nbs$stratum <- "NBS"
#   save(ind, ind_ebs, ind_nbs, file = f3)
# } else {
# load(f3)
# }

# NOTE: if using get_index_split() rather than get_index() you need to pass the
# model object and new data (not prediction object, you can bypass that step) and run:
f3 <- here("species_specific_code", "BS", species,
           "index_comparison", "indices.RData")
if (!file.exists(f3)) {
  gc()
  ind <- get_index_split(fit_sdmTMB, newdata = pred_grid, nsplit = 2, # may need 6 if have 64GB RAM
                         bias_correct = TRUE, area = pred_grid$area_in_survey_km2)
  #if using offsets also include the argument predict_args = list(offset = fake_offset)
  ind$stratum <- "Both"

  ind_ebs <- get_index_split(fit_sdmTMB, newdata = pred_grid_ebs, nsplit = 2,
```

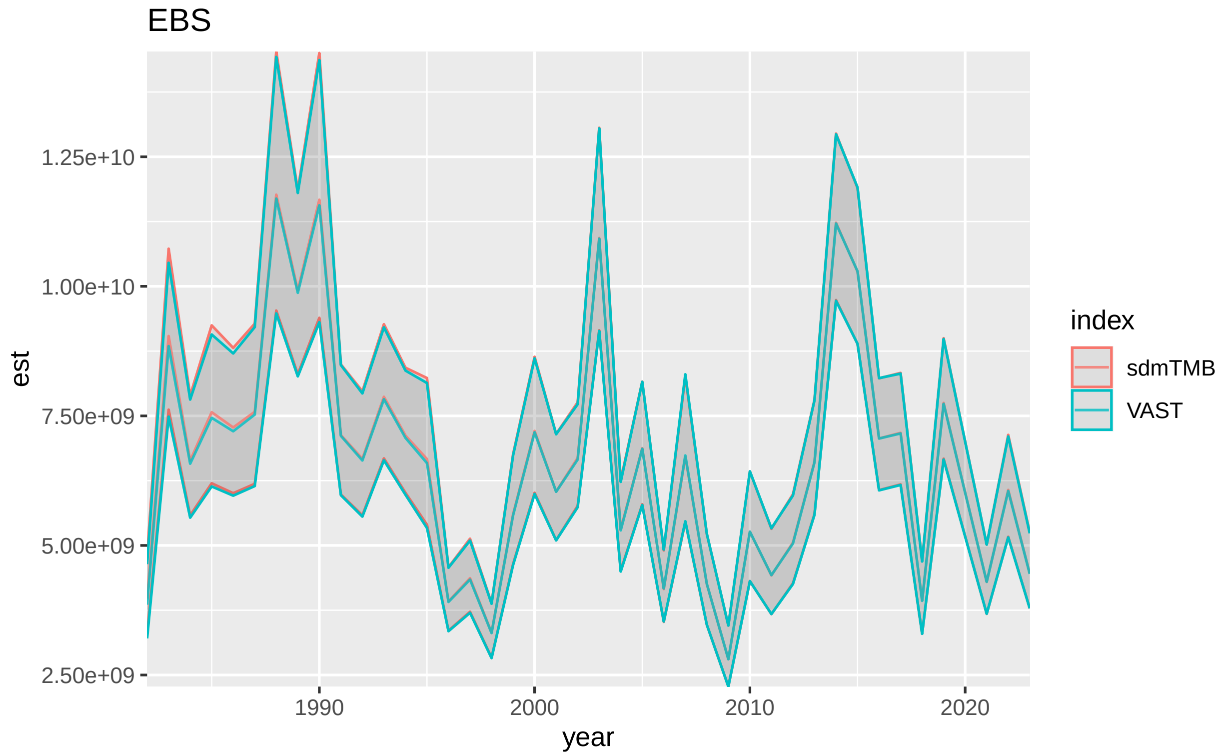
```
                              bias_correct = TRUE, area = pred_grid_ebs$area_in_survey_km2)
  ind_ebs$stratum <- "EBS"

  ind_nbs <- get_index_split(fit_sdmTMB, newdata = pred_grid_nbs, nsplit = 2,
                              bias_correct = TRUE, area = pred_grid_nbs$area_in_survey_km2)
  ind_nbs$stratum <- "NBS"
  save(ind, ind_ebs, ind_nbs, file = here("species_specific_code", "BS", species, "index_comparison", "
} else {
load(f3)
}
#> Calculating index in 2 chunks ===============>--------------- 50% | ETA: 0s
#> Calculating index in 2 chunks ===============>--------------- 50% | ETA: 0s
#> Calculating index in 2 chunks ===============>--------------- 50% | ETA: 0s
```

Now, we can compare the indices.

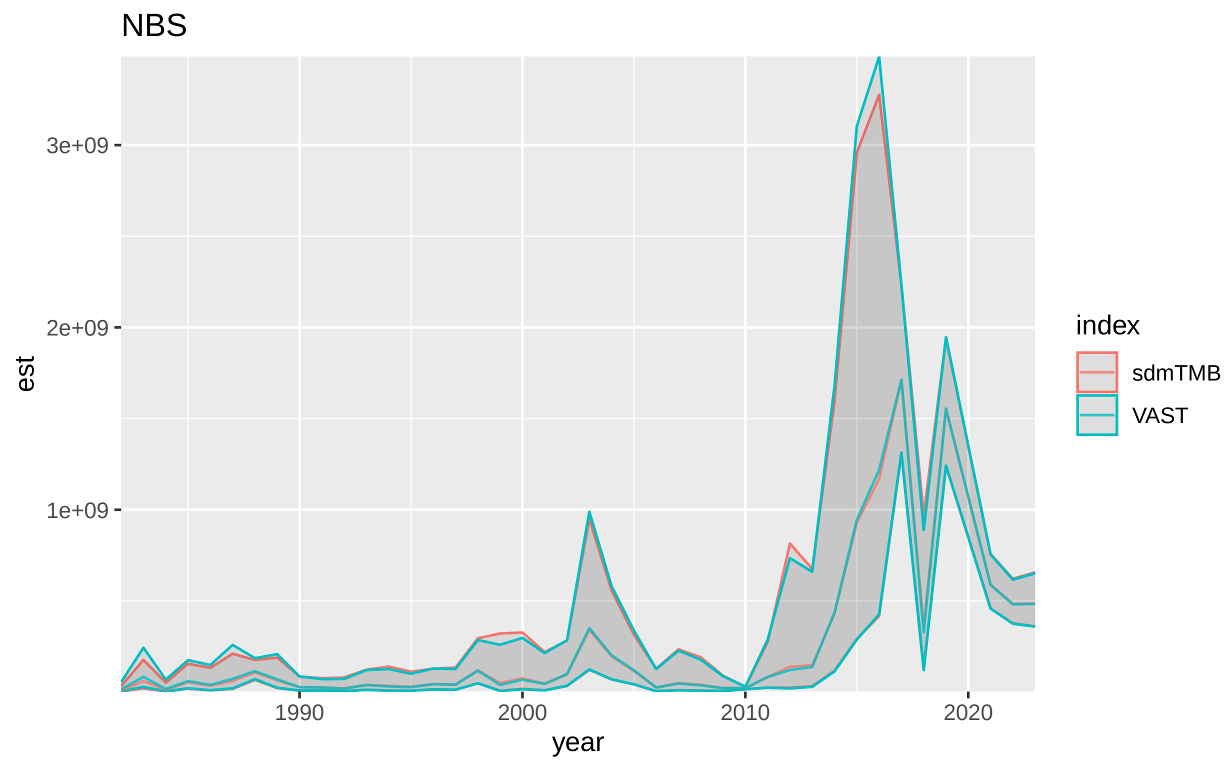
```
vast_i <- read.csv(here("species_specific_code", "BS", species, "index_comparison", "Index.csv")) %>%
  mutate(index = "VAST", year = as.numeric(Time), est = Estimate,
    se = Std..Error.for.ln.Estimate.) %>%
  select(index, year, est, se, stratum = Stratum) %>%
  filter(year != 2020) %>%
  mutate(lwr = exp(log(est) + qnorm(0.025) * se)) %>%
  mutate(upr = exp(log(est) + qnorm(0.975) * se))
sdm_i <- bind_rows(ind, ind_ebs, ind_nbs) %>% mutate(index = "sdmTMB")
both_i <- bind_rows(sdm_i, vast_i) %>% filter(est > 0)

ggplot(filter(both_i, stratum == "Both"), 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("EBS + NBS") +
  coord_cartesian(expand = FALSE)
```



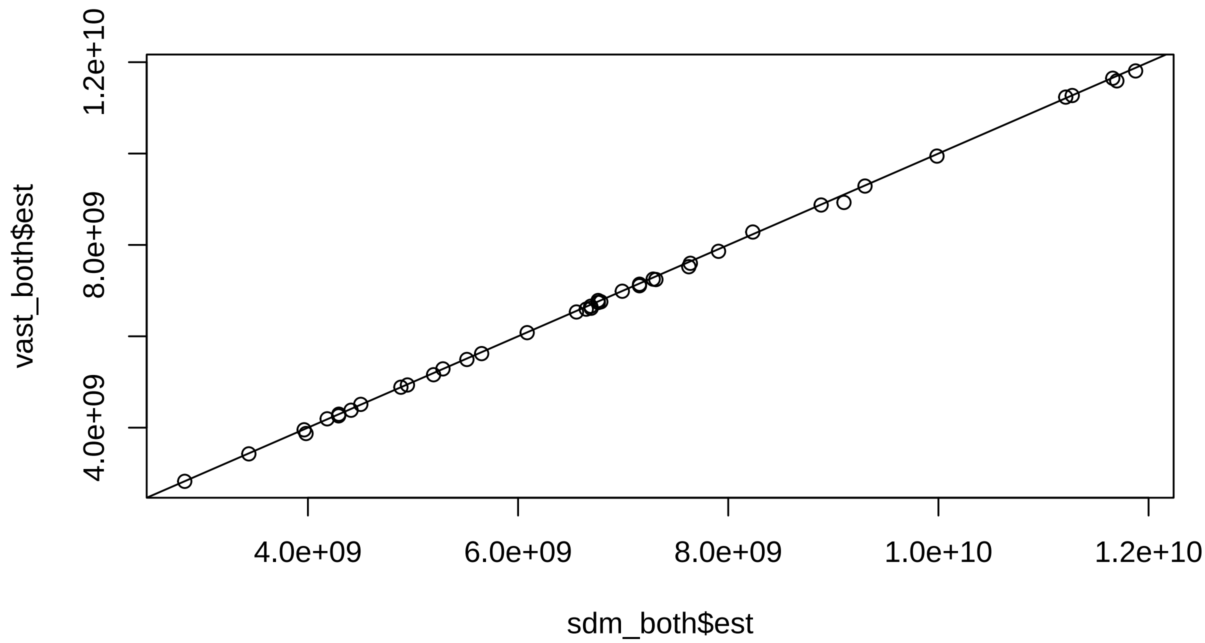

```r
ggplot(filter(both_i, stratum == "EBS"), aes(x = year, y = est, ymin = lwr, ymax = upr, colour = index))
  geom_ribbon(alpha = 0.1) +
  geom_line(alpha = 0.8) +
  ggtitle("EBS") +
  coord_cartesian(expand = FALSE)
```



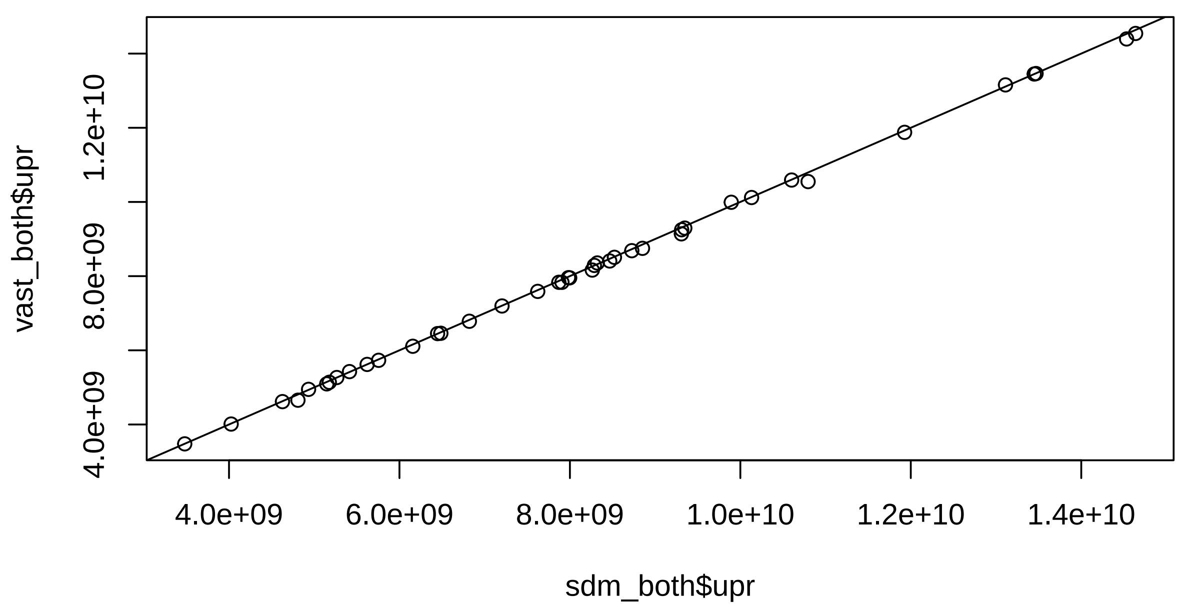

```
ggplot(filter(both_i, stratum == "NBS"), aes(x = year, y = est, ymin = lwr, ymax = upr, colour = index)
  geom_ribbon(alpha = 0.1) +
  geom_line(alpha = 0.8) +
  ggtitle("NBS") +
  coord_cartesian(expand = FALSE)
```



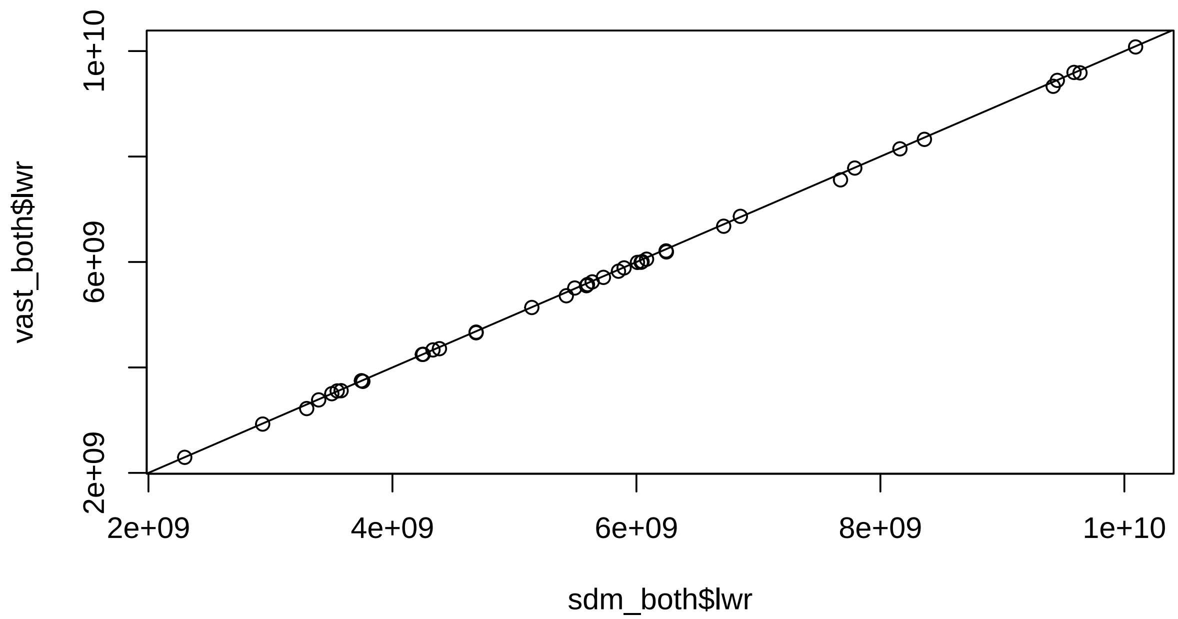

```r
vast_both <- filter(vast_i, stratum == "Both")
sdm_both <- filter(sdm_i, stratum == "Both")

plot(sdm_both$est, vast_both$est);abline(0, 1)
```




```r
plot(sdm_both$upr, vast_both$upr);abline(0, 1)
```




```r
plot(sdm_both$lwr, vast_both$lwr);abline(0, 1)
```




```
(sdm_both$est - vast_both$est) / vast_both$est
#>  [1]  2.844843e-02  1.923805e-02  8.259166e-03  1.363634e-02  9.520020e-03
#>  [6]  5.508149e-03  5.776039e-03  4.212889e-03  9.253388e-03  2.255428e-03
#> [11]  5.102604e-03  6.041121e-03  6.799978e-03  1.226418e-02  2.770062e-03
#> [16]  6.698355e-03  3.400976e-03  5.790334e-03  4.777753e-03  1.133783e-03
#> [21]  4.661272e-03  2.801184e-04  3.750900e-03  7.985234e-04 -2.380418e-03
#> [26] -3.104861e-03 -3.030112e-05  9.015368e-04 -2.805398e-04 -1.458466e-03
#> [31]  7.237483e-03  3.425515e-03  9.357016e-04 -2.129354e-03 -5.869361e-03
#> [36]  1.141385e-03  7.931521e-03  1.419053e-03 -1.287037e-04  3.604492e-03
#> [41]  2.617912e-03
(sdm_both$upr - vast_both$upr) / vast_both$upr
#>  [1]  0.0330466161  0.0232555893  0.0093036864  0.0183989423  0.0116944701
#>  [6]  0.0057071724  0.0065256339  0.0042232075  0.0094956208  0.0018145812
#> [11]  0.0054294576  0.0065101921  0.0069592723  0.0122233449  0.0022665035
#> [16]  0.0075392727  0.0032780528  0.0054352552  0.0046108765  0.0011407541
#> [21]  0.0046352631 -0.0000983470  0.0040890215 -0.0002431368 -0.0027819081
#> [26] -0.0039910256 -0.0003629763  0.0007791243 -0.0005753755 -0.0022665393
#> [31]  0.0077815211  0.0030261660  0.0005504194 -0.0034303350 -0.0097881771
#> [36]  0.0011751539  0.0101640458  0.0010588223  0.0001502297  0.0045224201
#> [41]  0.0043495616
(sdm_both$lwr - vast_both$lwr) / vast_both$lwr
#>  [1]  2.387072e-02  1.523628e-02  7.215726e-03  8.896012e-03  7.350243e-03
#>  [6]  5.309164e-03  5.027003e-03  4.202571e-03  9.011214e-03  2.696469e-03
#> [11]  4.775857e-03  5.572268e-03  6.640710e-03  1.230502e-02  3.273873e-03
#> [16]  5.858139e-03  3.523913e-03  6.145539e-03  4.944657e-03  1.126811e-03
#> [21]  4.687282e-03  6.587270e-04  3.412892e-03  1.841269e-03 -1.978765e-03
#> [26] -2.217909e-03  3.024847e-04  1.023964e-03  1.438291e-05 -6.497389e-04
#> [31]  6.693739e-03  3.825023e-03  1.321132e-03 -8.266755e-04 -1.935036e-03
```

```
#> [36]  1.107617e-03  5.703931e-03  1.779414e-03 -4.075593e-04  2.687403e-03
#> [41]  8.892485e-04
```

This document was built using:

```
R.Version()$version.string
#> [1] "R version 4.3.0 (2023-04-21 ucrt)"
packageVersion("VAST")
#> [1] '3.10.0'
packageVersion("FishStatsUtils")
#> [1] '2.12.0'
```