UW FISH 572 Survey Science

Practical introduction to model-based index standardization in sdmTMB

Lewis Barnett & Eric Ward

Alaska Fisheries Science Center

February 2, 2026

Outline

- Benefits of model-based index approaches
- Intro to spatiotemporal models and sdmTMB
- Model-based index standardization with sdmTMB
- Self-guided practical coding exercises

Why model-based index standardization?

- Design-based methods (stratified means) are straightforward but assume perfect survey implementation
- Model-based approaches can account for aspects of imperfect implementation
- Model-based approaches can improve precision (and accuracy in some cases)
- Model-based approaches can integrate additional data sources, multiple gear types, accounting for gear calibration, etc.

Model-based approaches range in complexity and benefits

- Model-based approaches can be simple like a multiple linear regression
- Include auxiliary information in the form of covariates (e.g., habitat, vessel)
    - Modeled as a simple difference in mean
    - Or as a linear effect
    - Or more complex nonlinear parametric and nonparametric forms of responses

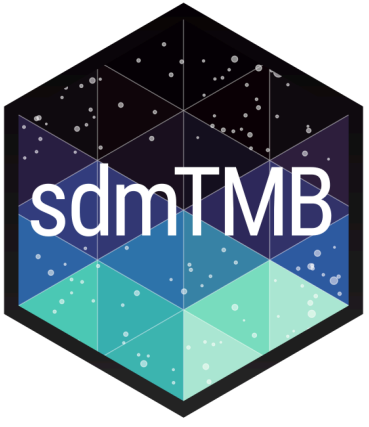
Model-based approaches range in complexity and benefits

- GLMs and delta/hurdle models allow handling of bimodal and skewed data typical of fish observations
- GLMMs and GAMs incorporate finer scale spatial relationships through random effects or smoothers
- Spatiotemporal models can leverage spatial and temporal correlation

sdmTMB highlights

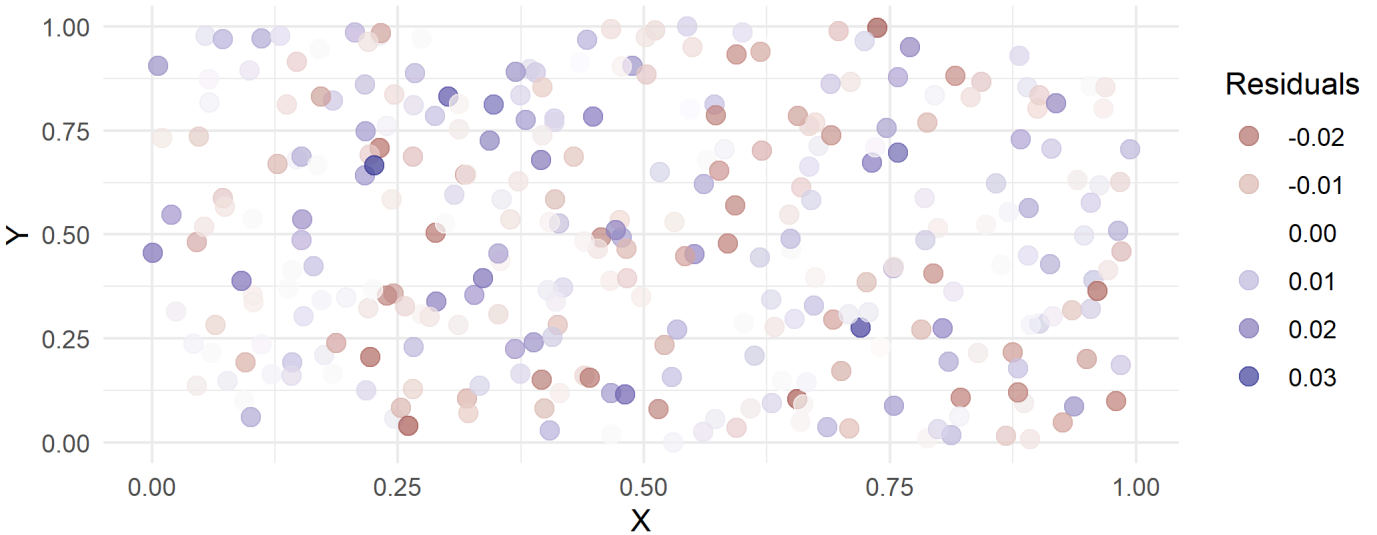
An R package for spatially explicit predictive-process GLMMs (Generalized Linear Mixed Effect Models)

- Can model spatial and spatiotemporal variation with Gaussian random fields
- Provides spatial covariance statistics that can be useful (e.g. spatial scale of correlation)



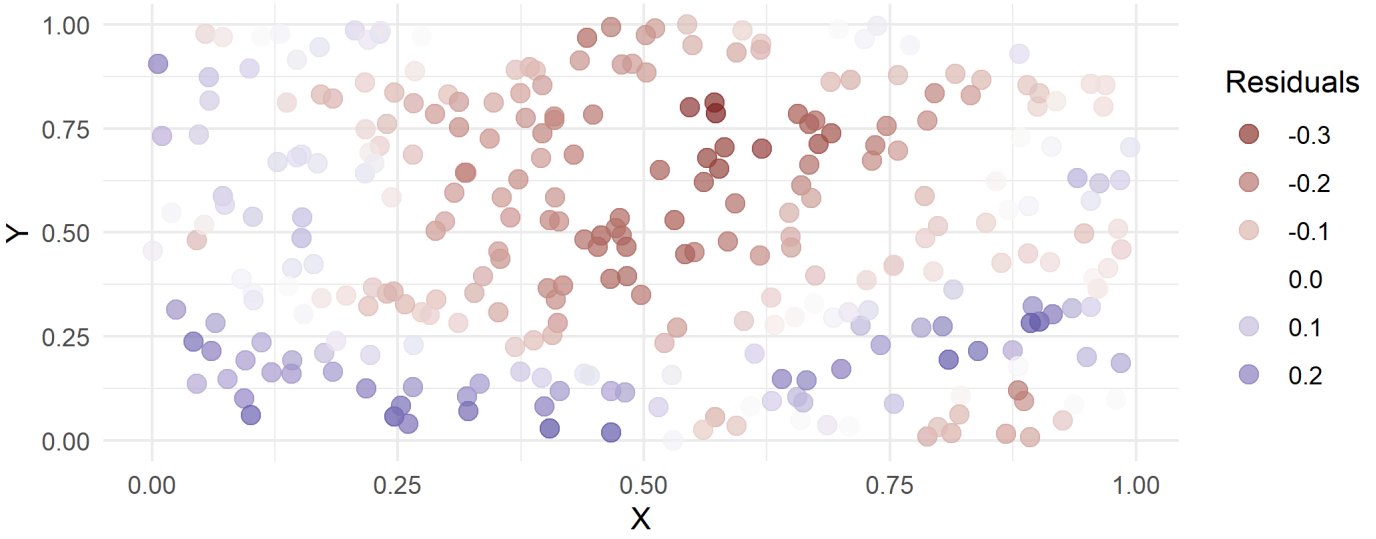
Why use a spatial model?

- Data often has spatial attributes
- Ideal world:
  - Plug spatial covariates into a GLM / GLMM
  - Residuals are uncorrelated



Reality

- Residual spatial autocorrelation



Modeling spatial autocorrelation

- Need 'wiggly'/smooth surface for approximating all spatial variables missing from model ('latent' variables)
- Several equivalent approaches exist
  - Smooths in `mgcv()`
  - Random fields and the Stochastic Partial Differential Equation (SPDE)
- SPDE differs in that it explicitly estimates parameters for spatial covariance function

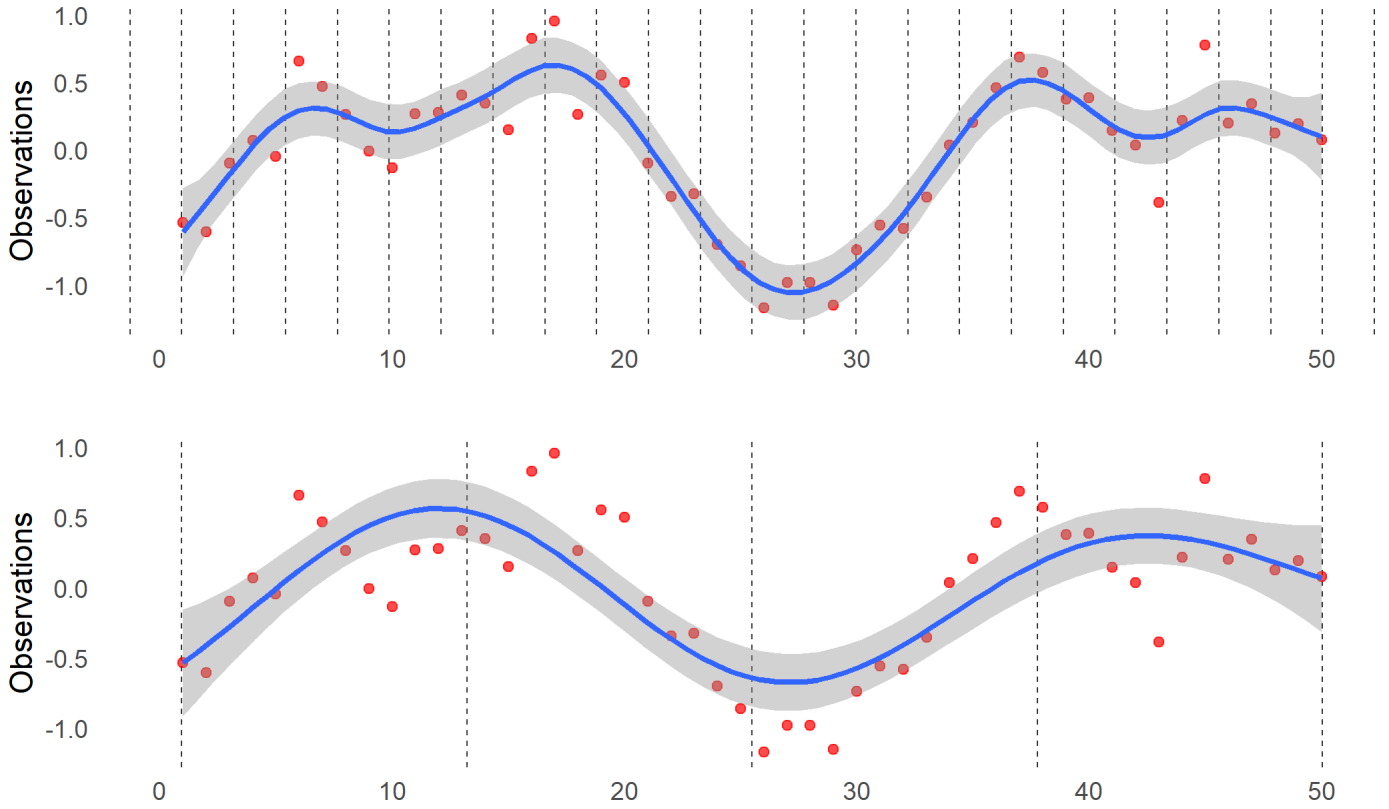
Miller, D.L., Glennie, R. & Seaton, A.E. Understanding the Stochastic Partial Differential Equation Approach to Smoothing. JABES 25, 1–16 (2020)

Predictive process models

- Estimate spatial field as random effects (random field)
- Gaussian process predictive process models:
  - Estimate values at a subset of locations ('knots')
  - Use covariance function to interpolate from knots to locations of observations

Predictive process models

- More knots (vertical dashed lines) = more wiggliness & parameters to estimate



What is a random field in practical terms?

- A way of estimating a wiggly surface to account for spatial and/or spatiotemporal correlation in data
- Alternatively, a way of estimating a wiggly surface to account for "latent" or unobserved variables
- As a bonus, it provides useful covariance parameter estimates: spatial variance and the distance at data points are effectively uncorrelated ("range")

sdmTMB workflow

1. Prepare data (convert to UTMs, scale covariates, ...)
2. Construct a mesh
3. Fit the model
4. Inspect the model (and possibly refit the model)
5. Predict from the model
6. Calculate index

sdmTMB workflow

1. Prepare data: `add_utm_columns()`
2. Construct a mesh: `make_mesh()`
3. Fit the model: `sdmTMB()`
4. Inspect the model: `print()`, `sanity()`, `tidy()`, `residuals()`
5. Predict from the model: `predict()`
6. Calculate index: `get_index()`

Preparing data: getting into constant distance coordinates

Projecting to UTMs to preserve distance

- Helper function: `sdmTMB::add_utm_columns()`
- Guesses UTM zone and uses the sf package: `sf::st_as_sf()`, `sf::st_transform()`, and `sf::st_coordinates()`
- Note default `units = "km"` because:
  - Range parameter estimated in units of X and Y
  - Values not too big or small for estimation

Constructing a mesh

Constructing a mesh

`make_mesh()` has 2 shortcuts to mesh construction

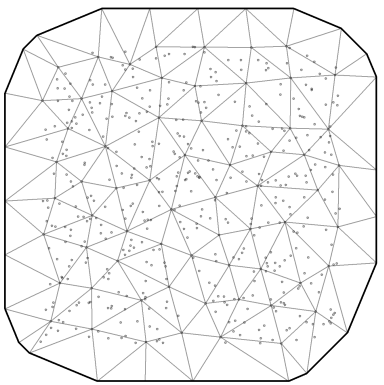
1. K-means algorithm: used to cluster points (e.g., `n_knots = 100`); approach taken in VAST; sensitive to random `seed` argument!

2. Cutoff: minimum allowed distance between vertices (e.g., `cutoff = 10`)

Constructing a mesh

Size of mesh has the single largest impact on fitting speed
`cutoff` (minimum triangle side length) is in units of x and y

```
d <- data.frame(x = runif(500), y = runif(500))
mesh <- make_mesh(d, xy_cols = c("x", "y"), cutoff = 0.1)
mesh$mesh$n
#> [1] 91
plot(mesh)
```



Fitting the model: sdmTMB()

sdmTMB()

Common arguments:

```
fit <- sdmTMB(
  formula,
  data,
  mesh,
  time = NULL,
  family = gaussian(link = "identity"),
  spatial = c("on", "off"),
  spatiotemporal = c("iid", "ar1", "rw", "off"),
  silent = TRUE,
  ...
)
```

See `?sdmTMB`

Fitting the model: non-spatial model components

Formula interface

sdmTMB uses a similar formula interface to widely used R packages

A formula is used to specify fixed effects and (optionally) random intercepts

```
# linear effect of x1:
formula = y ~ x1

# add smoother effect of x2:
formula = y ~ x1 + s(x2)

# add random intercept by group g:
formula = y ~ x1 + s(x2) + (1 | g)
```

Other common R formula options

Polynomials and omitting the intercept:

```
# polynomial functions using `poly`
y ~ poly(depth, degree = 2)

# omit intercept
y ~ -1 + as.factor(year)
y ~ 0 + as.factor(year)
```

Families and links

Many of the same families used in `glm()`, `glmmTMB()`, `mgcv::gam()` can be used here

Includes: `gaussian()`, `Gamma()`, `binomial()`, `poisson()`, `Beta()`, `student()`, `tweedie()`, `nbinom1()`, `nbinom2()`, `truncated_nbinom1()`, `truncated_nbinom2()`, `delta_gamma()`, `delta_lognormal()`, `delta_beta()`, and more...

All have `link` arguments

See `?sdmTMB::Families`

Fitting the model: spatial model components

Spatial vs. spatiotemporal fields

- A spatial field can be thought of as a spatial intercept
    - a wiggly spatial process that is constant in time
- Spatiotemporal variation represents separate fields estimated for each time slice (possibly correlated)
    - wiggly spatial processes that change through time

Spatial fields can be turned on/off

- By default `sdmTMB()` estimates a spatial field

```
fit <- sdmTMB(
  y ~ x,
  family = gaussian(),
  data = dat,
  mesh = mesh,
  spatial = "on",
  ...
)
```

Why *not* estimate a spatial field?

- If shared process across time slices isn't of interest
- If magnitude of spatiotemporal variability >> spatial variation
- If confounded with other parameters

Spatiotemporal fields

- Represents missing variables that vary through time
- Why include spatiotemporal fields?
    - If the data are collected in both space and time *and* there are 'latent' spatial processes that vary through time
        - e.g., effect of water temperature on abundance, if temperature wasn't in the model

Types of spatiotemporal fields

- None (`spatiotemporal = "off"`)
- Independent (`spatiotemporal = "iid"`)
- Random walk (`spatiotemporal = "rw"`)
- Autoregressive (`spatiotemporal = "ar1"`)

Types of spatiotemporal fields

- The type of field to use depends on this question:
    - Do you expect hotspots to be independent with each time slice (IID) or adapt slowly over time (AR1/RW)?
- For stock assessments, indices often use IID fields so that all temporal correlation is handled inside the assessment model itself

Fit a model for index standardization of cod data

Omit intercept and include an annual mean in formula to derive annual index from coefficients

```
mesh <- make_mesh(pcod, xy_cols=c("X", "Y"), cutoff=10)
fit <- sdmTMB(
  density ~ 0 + as.factor(year),
  data = pcod,
  mesh = mesh,
  family = tweedie(link = "log"),
  spatial = "on",
  time = "year",
  spatiotemporal = "iid"
)
```

Fit a model for index standardization of cod data in Queen Charlotte Sound

Include a covariate (depth) and specify how this should be modeled

```
fit <- sdmTMB(
  density ~ s(depth) + 0 + as.factor(year),
  data = pcod,
  mesh = mesh,
  family = tweedie(link = "log"),
  spatial = "on",
  time = "year",
  spatiotemporal = "iid"
)
```

Inspect the model

Checking convergence

- Learning curve of TMB can be steep
- Models can be very complex (hard to diagnose issues)
- `sanity()` function tries to help

```
sanity(fit)
#> ✓ Non-linear minimizer suggests successful convergence
#> ✓ Hessian matrix is positive definite
#> ✓ No extreme or very small eigenvalues detected
#> ✓ No gradients with respect to fixed effects are >= 0.001
#> ✓ No fixed-effect standard errors are NA
#> ✓ No standard errors look unreasonably large
#> ✓ No sigma parameters are < 0.01
#> ✓ No sigma parameters are > 100
#> ✓ Range parameter doesn't look unreasonably large
```

Inspecting the model fit

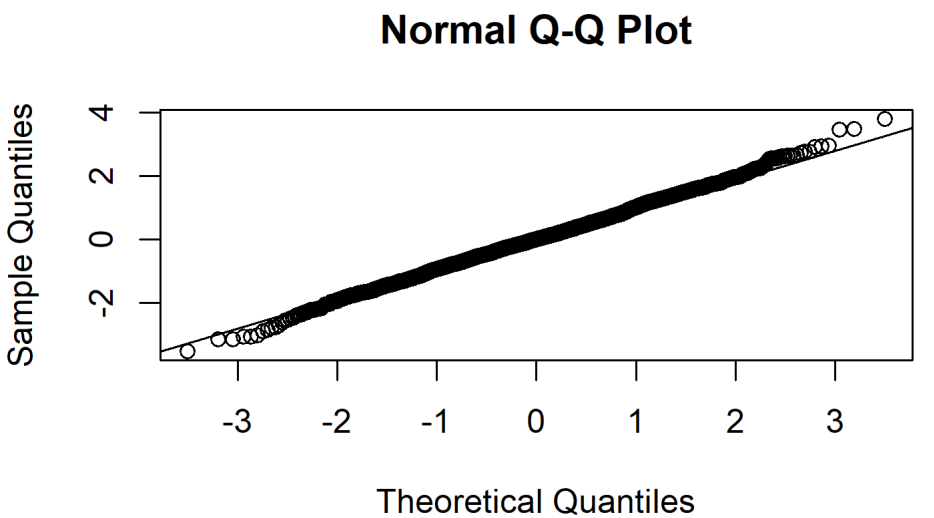
```
fit
```

```
#>                     coef.est coef.se
# as.factor(year)2003     1.80    0.28
# as.factor(year)2004     2.40    0.27
# as.factor(year)2005     2.12    0.27
# as.factor(year)2007     0.92    0.29
# as.factor(year)2009     1.43    0.28
# as.factor(year)2011     1.88    0.28
# as.factor(year)2013     2.17    0.27
# as.factor(year)2015     2.11    0.27
# as.factor(year)2017     1.39    0.28
# sdepth                  3.95   26.46
#
# Dispersion parameter: 10.73
# Tweedie p: 1.49
# Matern range: 12.23
# Spatial SD: 1.85
# Spatiotemporal SD: 1.84
# ML criterion at convergence: 6242.801
```

Model residuals

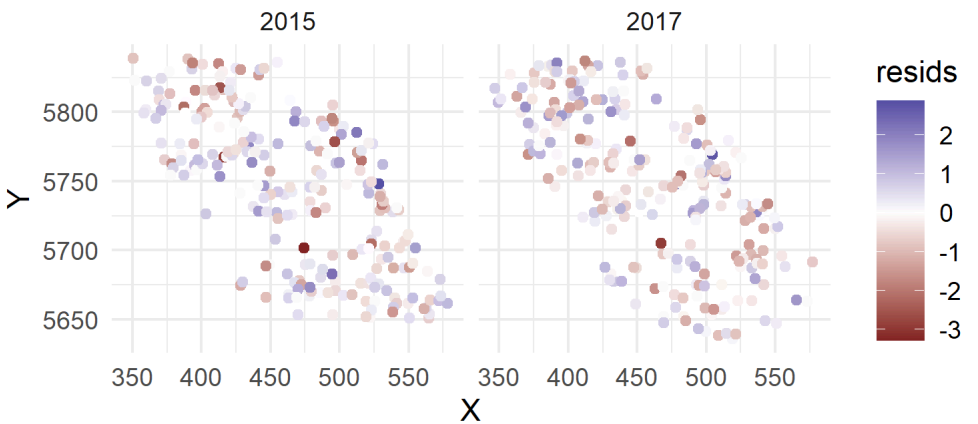
Warning: these residuals are fast but might look off even if the model is fine. Also see MCMC residuals. See the 'Residual checking' vignette.

```
set.seed(1)
rq_res <- residuals(fit) # randomized quantile residuals
qqnorm(rq_res);qqline(rq_res)
```



Model residuals in space

```
pcod$resids <- residuals(fit)
filter(pcod, year %in% c(2015, 2017)) %>%
  ggplot(aes(X, Y, colour = resids)) +
  geom_point() +
  facet_wrap(~year) +
  scale_colour_gradient2() +
  coord_fixed()
```

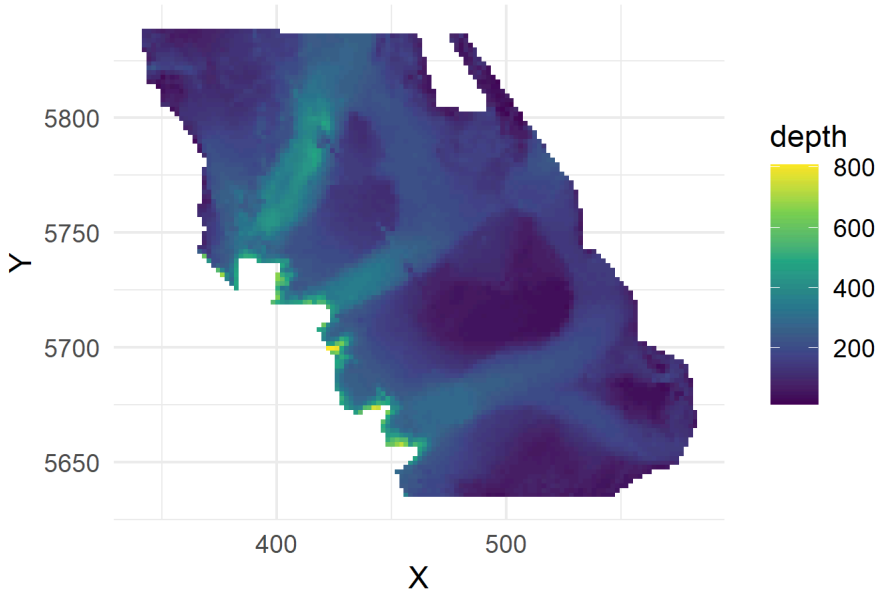


Predict from the model

Predict over the survey domain

- `qcs_grid`: a 2x2 km grid extending over the full survey domain

```
ggplot(qcs_grid, aes(X, Y, fill = depth)) +
  geom_raster() +
  coord_fixed()
```



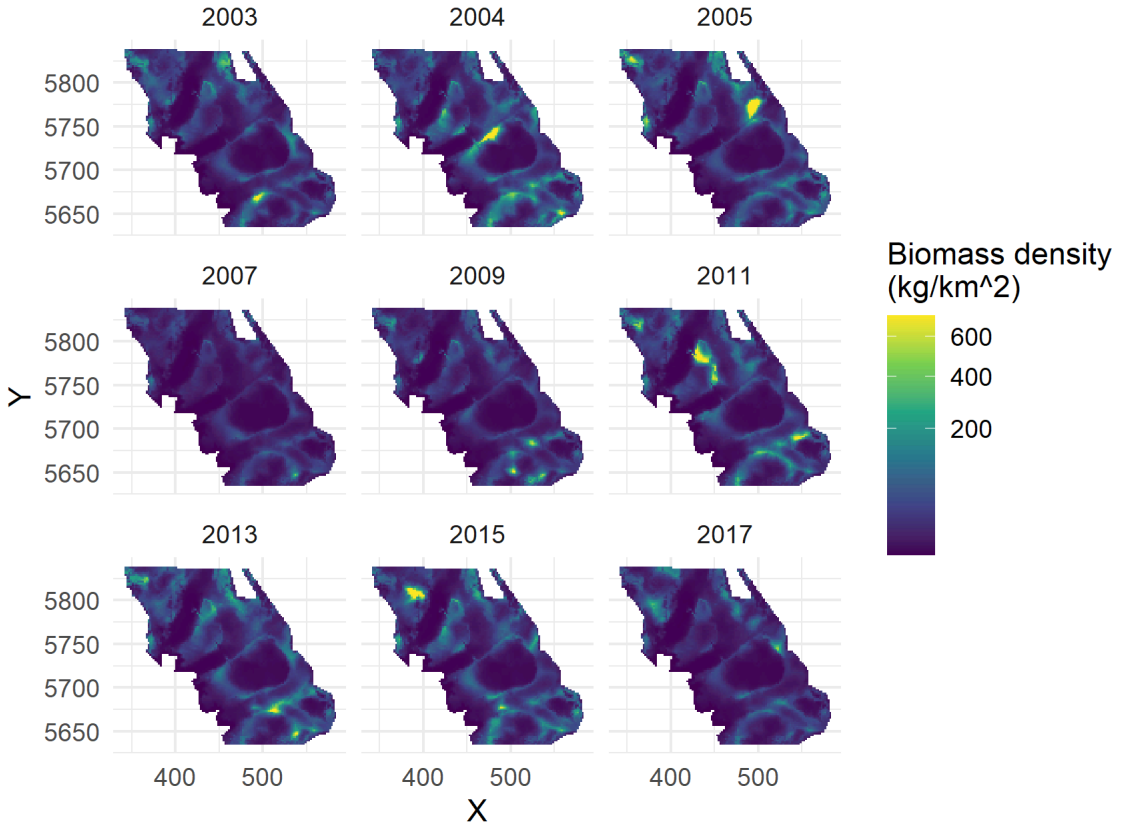
Predict over the survey domain

```
# replicate grid over all years:
survey_grid <- replicate_df(qcs_grid, "year", unique(pcod$year))
pred <- predict(
  fit, newdata = survey_grid, return_tmb_object = TRUE
)
select(pred$data, year, X, Y, est, est_non_rf, est_rf, omega_s, epsilon_st) |>
  head()
#>   year   X    Y       est est_non_rf     est_rf   omega_s  epsilon_st
#> 1 2003 456 5636 -3.730875  -3.803677 0.07280115 0.1008410 -0.02803982
#> 2 2003 458 5636  1.624400   1.529580 0.09482001 0.1232472 -0.02842721
#> 3 2003 460 5636  2.362616   2.245777 0.11683886 0.1456534 -0.02881459
#> 4 2003 462 5636  3.236758   3.097901 0.13885771 0.1680597 -0.02920197
#> 5 2003 464 5636  3.297821   3.110504 0.18731776 0.2060681 -0.01875031
#> 6 2003 466 5636  3.317910   2.966754 0.35115614 0.3121577  0.03899844
```

Predict over the survey domain

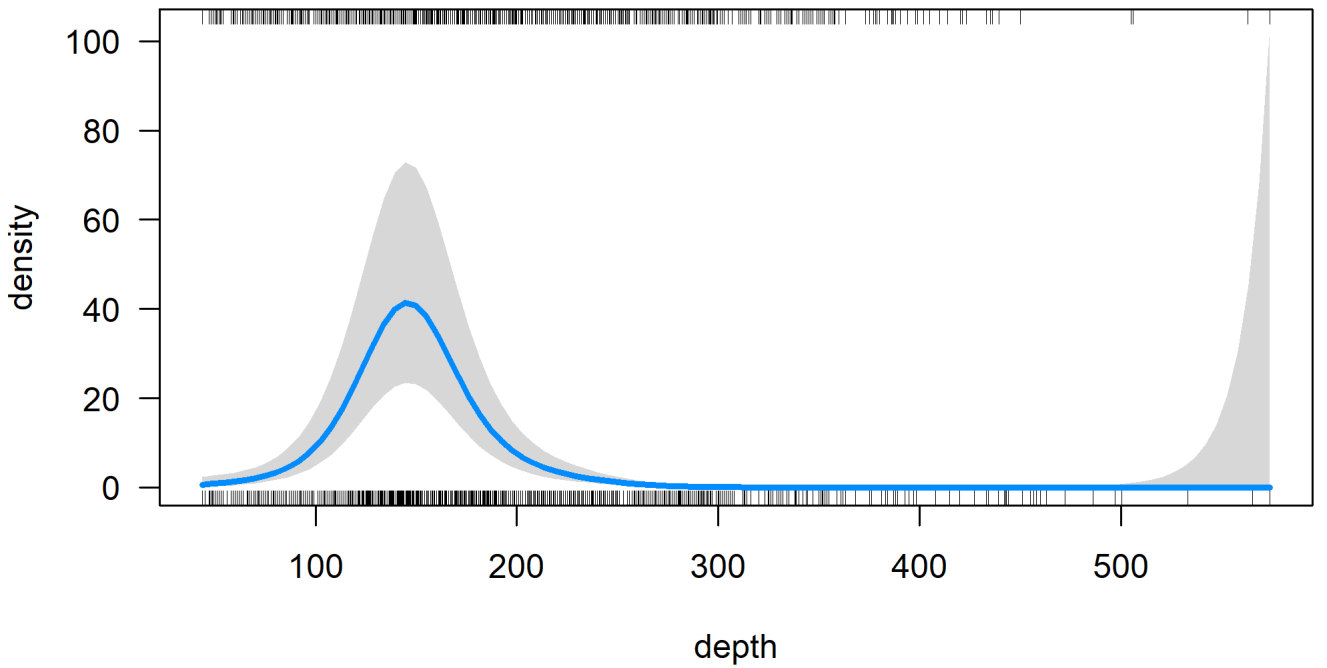
- `est`: Overall estimate in link space (log here)
- `est_non_rf`: Estimate of non-random-field components
- `est_rf`: Estimate of random-field components
- `omega_s`: Spatial random field
- `epsilon_st`: Spatiotemporal random field

Plotting overall predictions



Plotting depth effect with visreg

```
visreg::visreg(fit, xvar = "depth", scale = "response")
```



Calculate an area-weighted population index

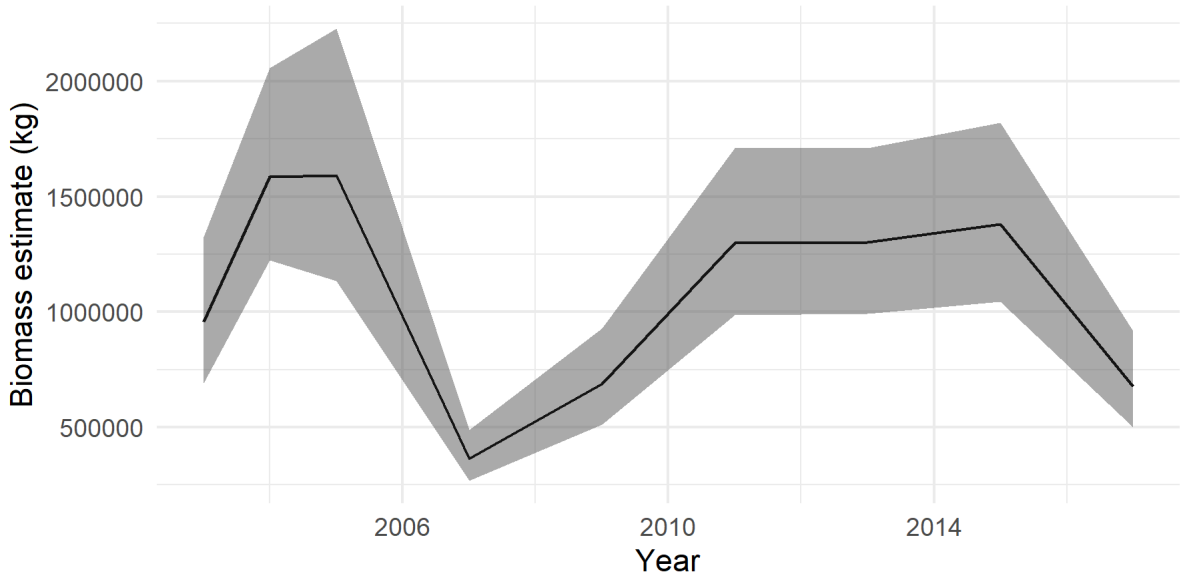
Sum densities multiplied by cell areas

(*and calculate standard errors)

```
index <- get_index(pred, area = 4, bias_correct = TRUE)
head(index)
#>   year       est       lwr       upr  log_est         se
#> 1 2003  957592.8  691686.7 1325721.5 13.77218 0.1659671
#> 2 2004 1585884.6 1223175.1 2056148.8 14.27665 0.1324985
#> 3 2005 1590357.5 1135313.1 2227788.1 14.27947 0.1719676
#> 4 2007  362802.1  269084.4  489160.2 12.80161 0.1524683
#> 5 2009  688792.4  511393.0  927730.7 13.44270 0.1519424
#> 6 2011 1298822.9  986437.6 1710134.3 14.07697 0.1403666
```

Plot the standardized index

```
ggplot(index, aes(year, est)) + geom_line() +
  geom_ribbon(aes(ymin = lwr, ymax = upr), alpha = 0.4) +
  xlab('Year') + ylab('Biomass estimate (kg)')
```



Zooming out for perspective on alternative indices

There is *one* design-based index.

There are *many* possible model-based indexes. Thankfully, results are often (but not always!) qualitatively similar across sensible configurations.

Commander, C.J.C, L.A.K. Barnett, E.J. Ward, S.C. Anderson, T.E. Essington. 2022. The shadow model: how and why small choices in spatially explicit species distribution models affect predictions. PeerJ. 10: e12783. https://doi.org/10.7717/peerj.12783

Model-based estimates of composition

See this vignette for estimating abundance or proportions at age

https://sdmtmb.github.io/sdmTMB/articles/age-composition.html#sdmtmb-vs--tinyvast-for-age-composition-models