Bycatch Estimation and Expansion in STAN

Eric Ward

June 04, 2019

Load library

```
require(devtools)
#devtools::install_github("eric-ward/bycatch")
library(bycatch)
set.seed(123)
```

Load data

```
# replace this with your own data frame
d = data.frame("Year"= 2002:2014,
  "Takes" = c(0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0),
  "expansionRate" = c(24, 22, 14, 32, 28, 25, 30,  7, 26, 21, 22, 23, 27),
  "Sets" = c(391, 340, 330, 660, 470, 500, 330, 287, 756, 673, 532, 351, 486))
```

Simple model with constant bycatch, no covariates

We'll start by fitting a model with constant bycatch rate,

```
fit = fit_bycatch(Takes ~ 1, data=d, time="Year", effort="Sets", family="poisson",
  time_varying = FALSE)
```

If divergent transition warnings or other issues indicating lack of convergence are a problem, we can try changing some of the control arguments, e.g.

```
fit = fit_bycatch(Takes ~ 1, data=d, time="Year", effort="Sets", family="poisson",
  time_varying = FALSE, control=list(adapt_delta=0.99,max_treedepth=20))
```

We can also increase the iterations and number of chains from the defaults (1000 and 3),

```
fit = fit_bycatch(Takes ~ 1, data=d, time="Year", effort="Sets", family="poisson",
  time_varying = FALSE, iter=3000, chains=4)
```

Make plots

```
plot_fitted(fit, xlab="Year", ylab = "Fleet-level bycatch")
```

We can include points also,

```
plot_fitted(fit, xlab="Year", ylab = "Fleet-level bycatch", include_points = TRUE)
```

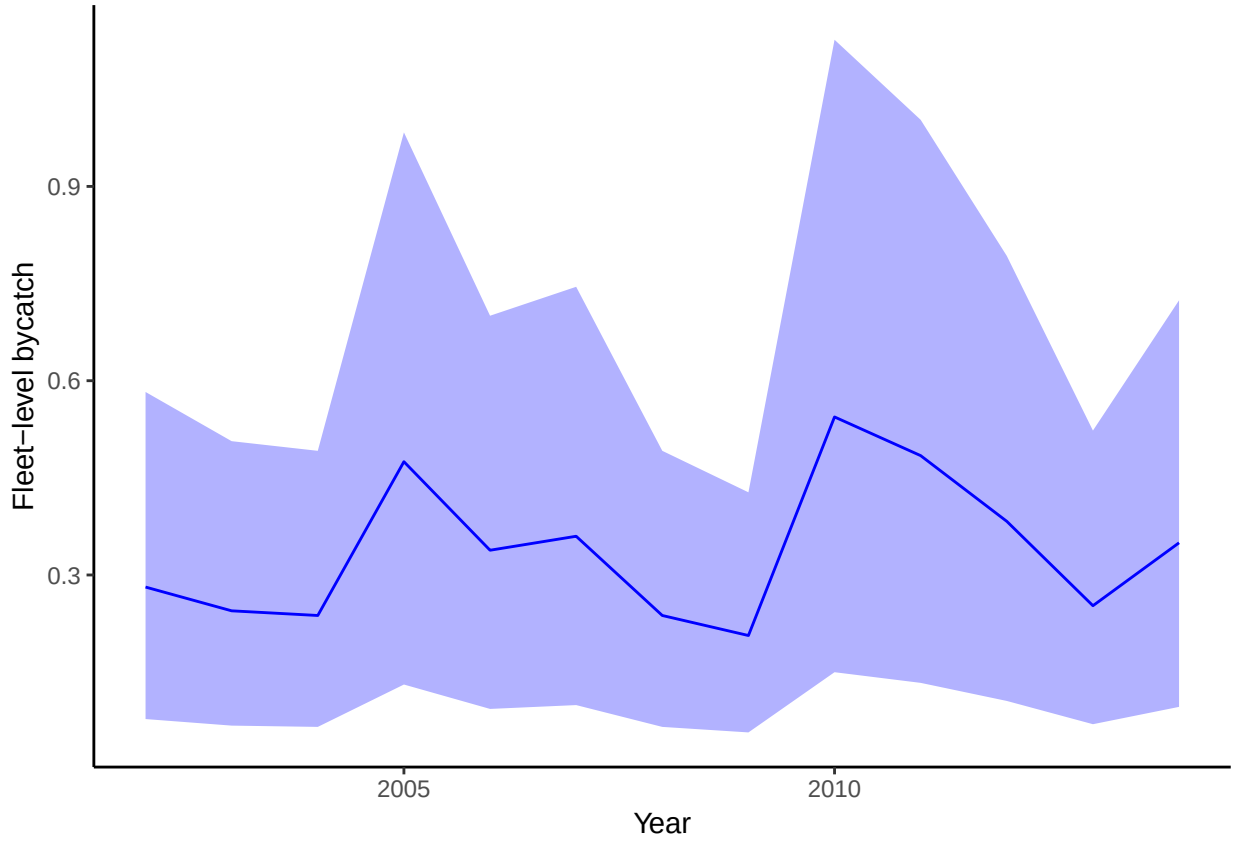

Figure 1: Estimated fleet-level bycatch (not expanded by observer coverage), incorporating data on takes, effort.

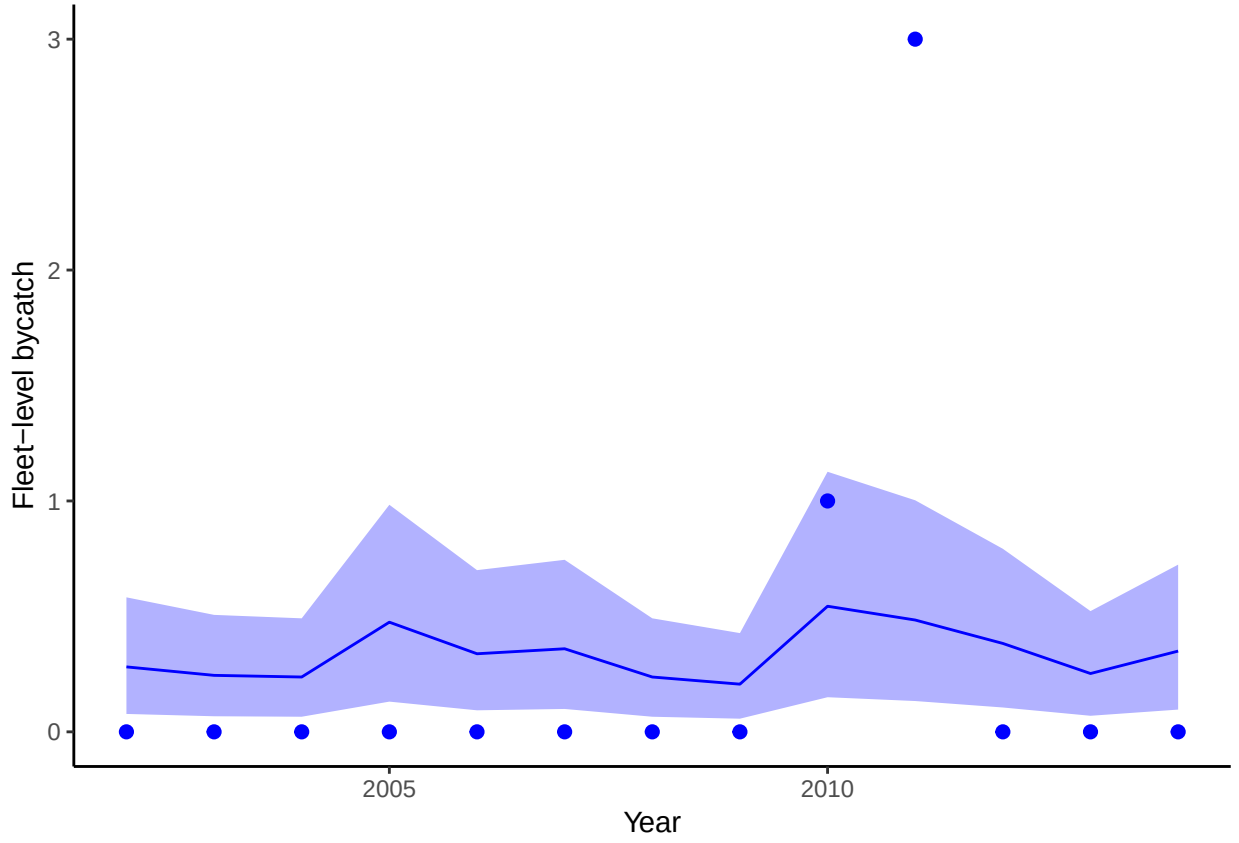

Figure 2: Estimated fleet-level bycatch (not expanded by observer coverage), incorporating data on takes, effort. Dots represent observed bycatch events.

Extracting model selection information (LOOIC)

The `loo` package in R provides a nice interface for extracting leave one out information criterion (LOOIC) from `stanfit` objects. Like AIC, lower is better. Values of LOOIC can be used to compare models with the same response but different structure (covariates or not, time-varying bycatch or not, etc).

```
loo::loo(fit$fitted_model)$estimates
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
##            Estimate        SE
## elpd_loo -11.177868  6.043199
## p_loo      2.526928  2.162952
## looic     22.355736 12.086398
```

Expanding bycatch estimates

Using our example above, the observer coverage for that dataset was less than 100% and so our estimates need to be expanded to the fleetwide level. There are some important `control` arguments here that are left at defaults, but should maybe be changed for huge numbers of bycatch.

```
expanded = expand(fit, coverage = d$expansionRate)
```

And we can then plot these estimates. Like the previous function we can specify whether to include the raw points or not.

```
plot_expanded(fitted_model=fit, expanded_estimates = expanded, xlab="Year", ylab = "Fleet-level bycatch
```

Make table of expanded bycatch estimates

We can also do things like summarize the expanded estimates in table form

```
df = data.frame("time" = d[,"Year"],
  "mean" = apply(expanded_estimates, 2, mean),
  "median" = apply(expanded_estimates, 2, quantile, 0.5),
  "lower95" = apply(expanded_estimates, 2, quantile, 0.025),
  "upper95" = apply(expanded_estimates, 2, quantile, 0.975))

write.table(df, "estimated_bycatch.csv", row.names=F, col.names=T, sep=",")
```

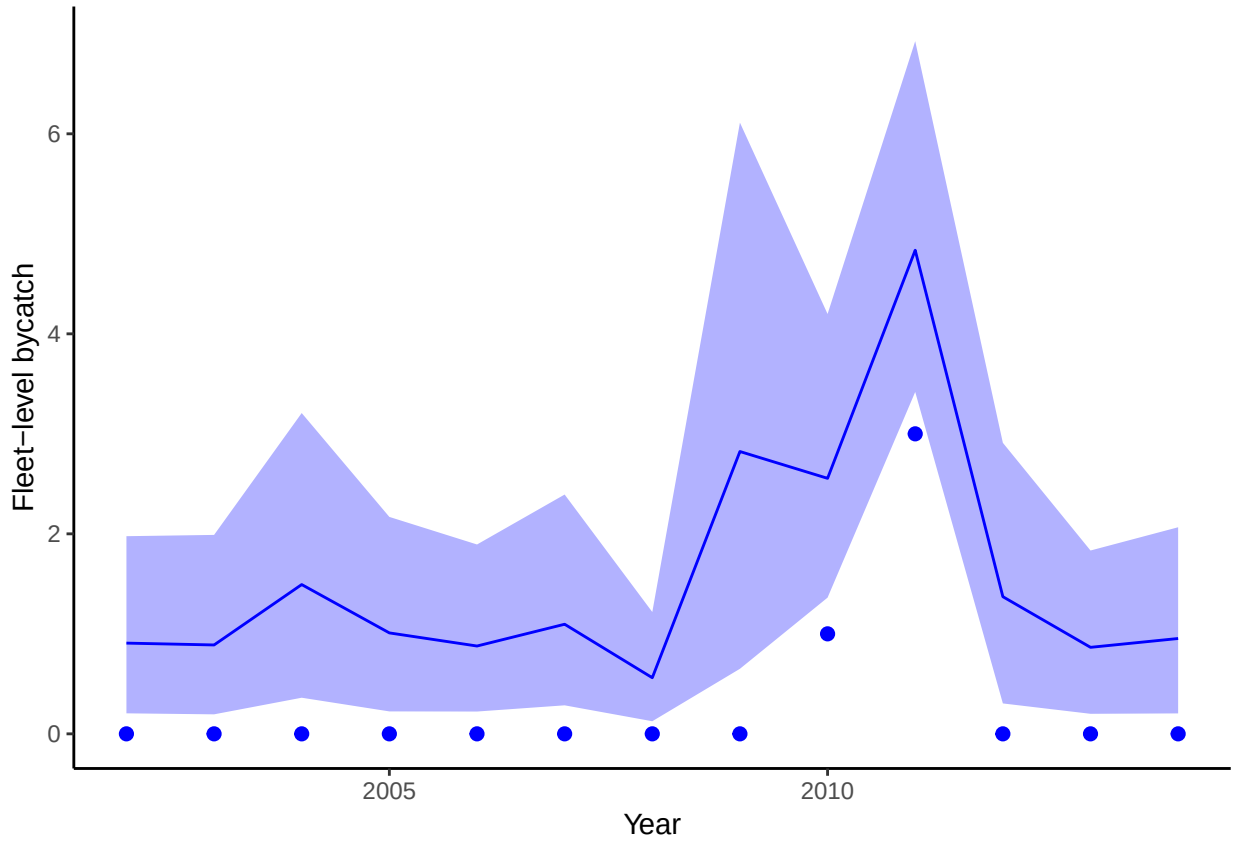

Figure 3: Estimated fleet-level expanded bycatch, incorporating data on takes, effort, and observer coverage. Dots represent observed bycatch events.

Negative binomial example

Using our example dataset above, we can also switch from the Poisson model to Negative Binomial.

```
fit = fit_bycatch(Takes ~ 1, data=d, time="Year", effort="Sets", family="nbinom2",
  time_varying = FALSE)
```

The degree of overdispersion here is stored in the variable `nb2_phi`, which we can get with

```
phi = rstan::extract(fit$fitted_model)$nb2_phi
```

Example with covariates

Following Martin et al. 2015 we can include fixed or continuous covariates.

For example, we could include a julian day, and a break point (representing a regulatory change for example) in the data. We could model the first variable as a continuous predictor and the second as a factor.

```
d$Day = sample(seq(220,280),size=nrow(d),replace=T)
d$Reg = ifelse(d$Year < 2008, 0, 1)
```

Using the formula interface makes it easy to include covariates,

```
fit = fit_bycatch(Takes ~ Day + Reg, data=d, time="Year", effort="Sets", family="poisson",
  time_varying = FALSE)
```

```
##
## SAMPLING FOR MODEL 'bycatch' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.02637 seconds (Warm-up)
## Chain 1:                0.181707 seconds (Sampling)
## Chain 1:                0.208077 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bycatch' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 8e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
```

```
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.027413 seconds (Warm-up)
## Chain 2:                0.030374 seconds (Sampling)
## Chain 2:                0.057787 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bycatch' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 8e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
## Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
## Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
## Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
## Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
## Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
## Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
## Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
## Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
## Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
## Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
## Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.04538 seconds (Warm-up)
## Chain 3:                0.078236 seconds (Sampling)
## Chain 3:                3.12362 seconds (Total)
## Chain 3:

## Warning: There were 2 chains where the estimated Bayesian Fraction of Missing Information was low. S
## http://mc-stan.org/misc/warnings.html#bfmi-low

## Warning: Examine the pairs() plot to diagnose sampling problems
```

We can get the 3 covariate effects out with the following call:

```
betas = rstan::extract(fit$fitted_model)$beta
```

Note that 'betas' has 3 columns. These correspond to (1) the intercept, (2) continuous predictor, and (3)

factor variable above. If we didn't include the covariates, we'd still estimate beta[1] as the intercept.

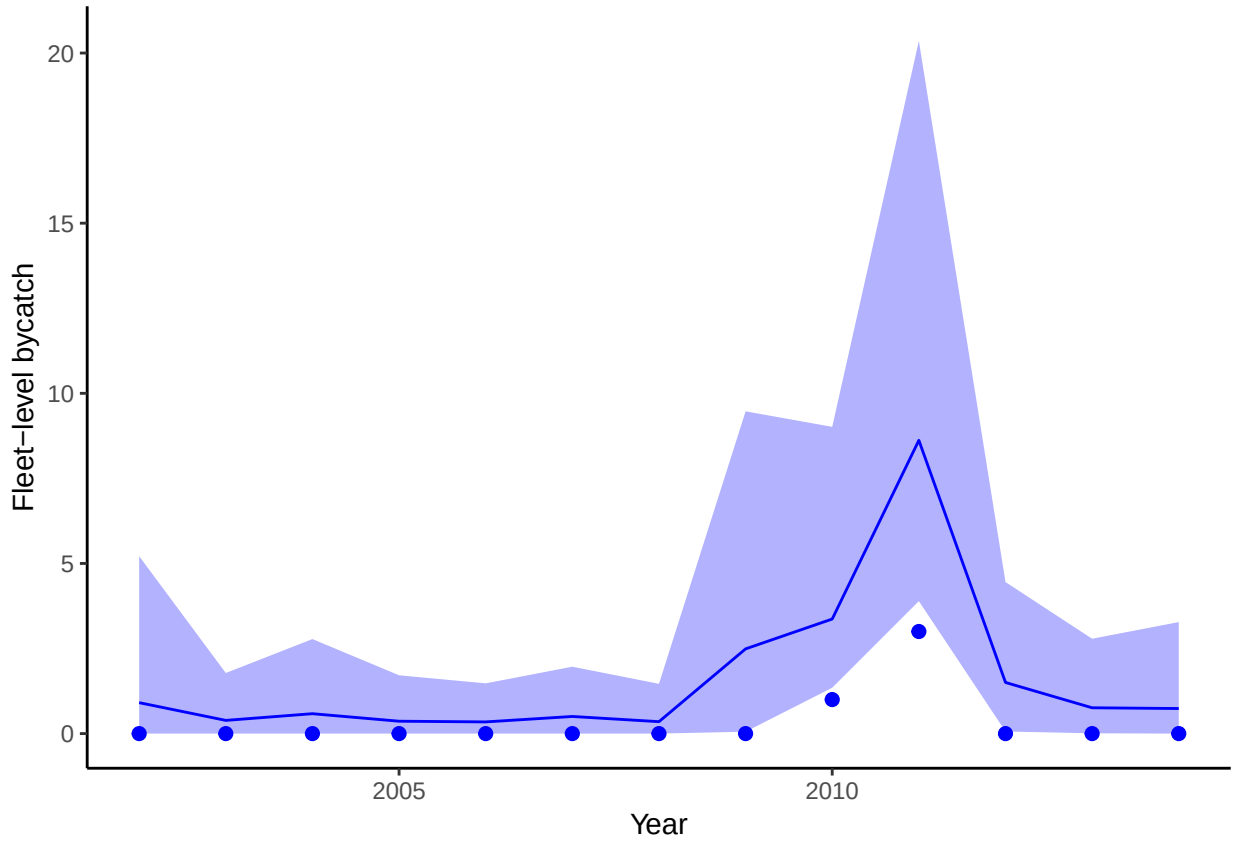

Figure 4: Estimated fleet-level expanded bycatch from the model with time-varying effects, incorporating data on takes, effort, and observer coverage. Dots represent observed bycatch events.

Fit model with time-varying effects

To incorporate potential autocorrelation, we can fit a model with time-varying random effects. This is equivalent to a dynamic linear model with time varying intercept in a Poisson GLM.

```
fit = fit_bycatch(Takes ~ 1, data=d, time="Year", effort="Sets", family="poisson",
  time_varying = TRUE)
```

```
expanded = expand(fit, coverage=d$expansionRate)
plot_expanded(fitted_model=fit, expanded_estimates = expanded, xlab="Year", ylab = "Fleet-level bycatch
```

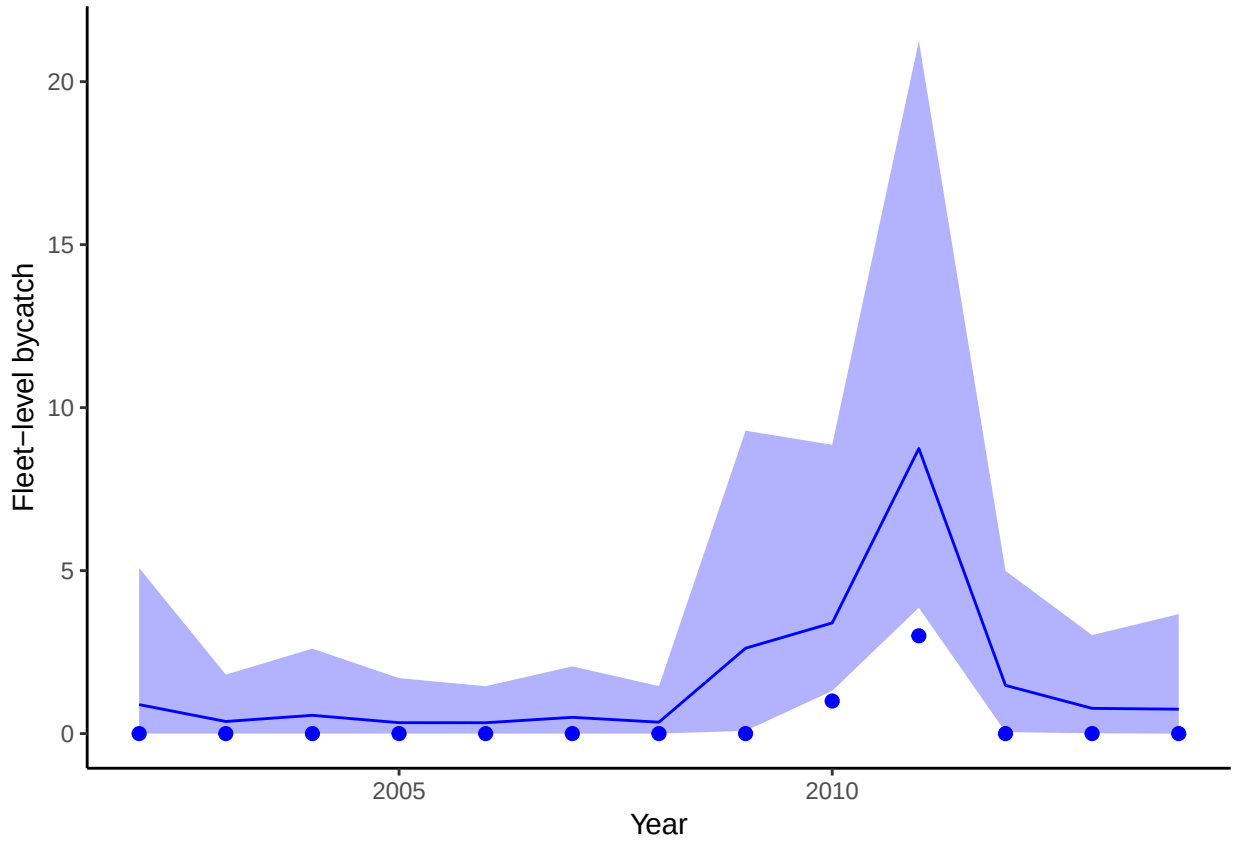

Figure 5: Estimated fleet-level expanded bycatch from the dataset with no events, incorporating effort, and observer coverage. Dots represent observed bycatch events.

Fit model with no bycatch events

```
# replace this with your own data frame
d = data.frame("Year"= 2002:2014,
  "Takes" = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
  "expansionRate" = c(24, 22, 14, 32, 28, 25, 30,  7, 26, 21, 22, 23, 27),
  "Sets" = c(391, 340, 330, 660, 470, 500, 330, 287, 756, 673, 532, 351, 486))
```

```
fit = fit_bycatch(Takes ~ 1, data=d, time="Year", effort="Sets", family="poisson",
  time_varying = FALSE)
```

```
expanded = expand(fit, coverage=d$expansionRate)
plot_expanded(fitted_model=fit, expanded_estimates = expanded, xlab="Year", ylab = "Fleet-level bycatch
```