Inference

David L Miller

What do we want to know?

- Don't just fit models for the sake of it!
- What are our questions?
  - Relationship to covariates
  - Abundance
  - Distribution
  - Response to disturbance
  - Temporal changes
  - Other stuff?

Prediction

What is a prediction?

- Evaluate the model, at a particular covariate combination
- Answering (e.g.) the question “at a given depth, how many dolphins?”
- Steps:

1. evaluate the $s(\ldots)$ terms
2. move to the response scale (exponentiate? Do nothing?)
3. (multiply any offset etc)

Example of prediction

- in maths:
  - Model: $\text{count}_i = A_i \exp\left(\beta_0 + s(x_i, y_i) + s(\text{Depth}_i)\right)$
  - Drop in the values of x, y, Depth (and A)
- in R:
  - build a `data.frame` with x, y, Depth, A
  - use `predict()`

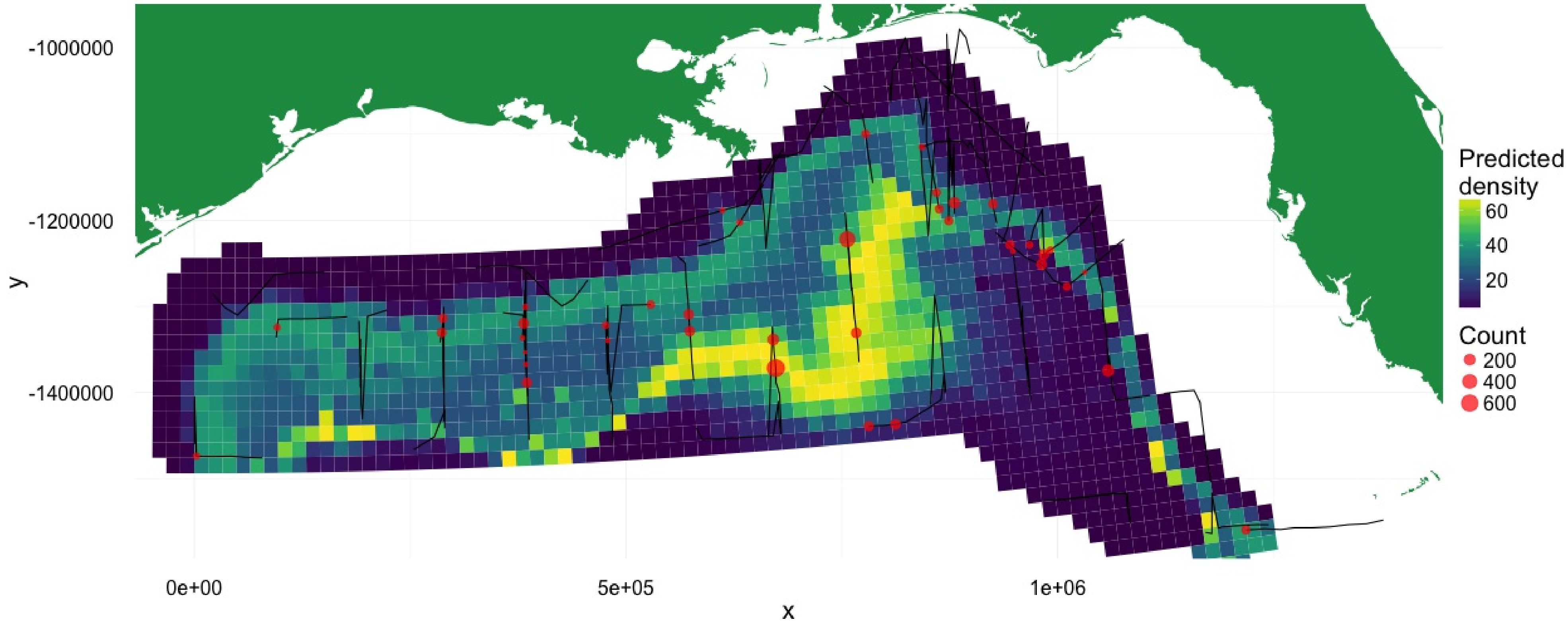
```
preds <- predict(my_model, newdat=my_data, type="response")
```

(`se.fit`=TRUE gives a standard error for each prediction)

Back to the dolphins…

Where are the dolphins?

```
dolphin_preds <- predict(dolphins_depth, newdata=preddata,
                         type="response")
```



(`ggplot2` code included in the slide source)

Prediction summary

- Evaluate the fitted model at a given point
- Can evaluate many at once (`data.frame`)
- Don't forget the `type=...` argument!
- Obtain per-prediction standard error with `se.fit`

Without uncertainty, we're not doing statistics

Where does uncertainty come from?

- $\boldsymbol{\beta}$: uncertainty in the spline parameters
- $\boldsymbol{\lambda}$: uncertainty in the smoothing parameter

- (Traditionally we've only addressed the former)

- (New tools let us address the latter…)

Parameter uncertainty

From theory:

$$\boldsymbol{\beta} \sim \mathrm{N}(\hat{\boldsymbol{\beta}}, \mathbf{V}_{\boldsymbol{\beta}})$$

(*caveat: the normality is only* **approximate** *for non-normal response*)

What does this mean? Variance for each parameter.

In `mgcv`: `vcov(model)` returns $\mathbf{V}_{\boldsymbol{\beta}}$.

What can we do this this?

- confidence intervals in `plot`
- standard errors using `se.fit`
- derived quantities? (see bibliography)


Huh. Well, there's only
one way to solve this.
Well, there's probably,
like, a lot of different ways.

The lpmatrix, magic, etc

For regular predictions:

$$\hat{\boldsymbol{\eta}}_{\mathrm{p}} = \mathrm{L_p}\hat{\boldsymbol{\beta}}$$

form $\mathrm{L_p}$ using the prediction data, evaluating basis functions as we go.

(Need to apply the link function to $\hat{\boldsymbol{\eta}}_{\mathrm{p}}$)

But the $\mathrm{L_p}$ fun doesn't stop there…

[[mathematics intensifies]]

Variance and lpmatrix

To get variance on the scale of the linear predictor:

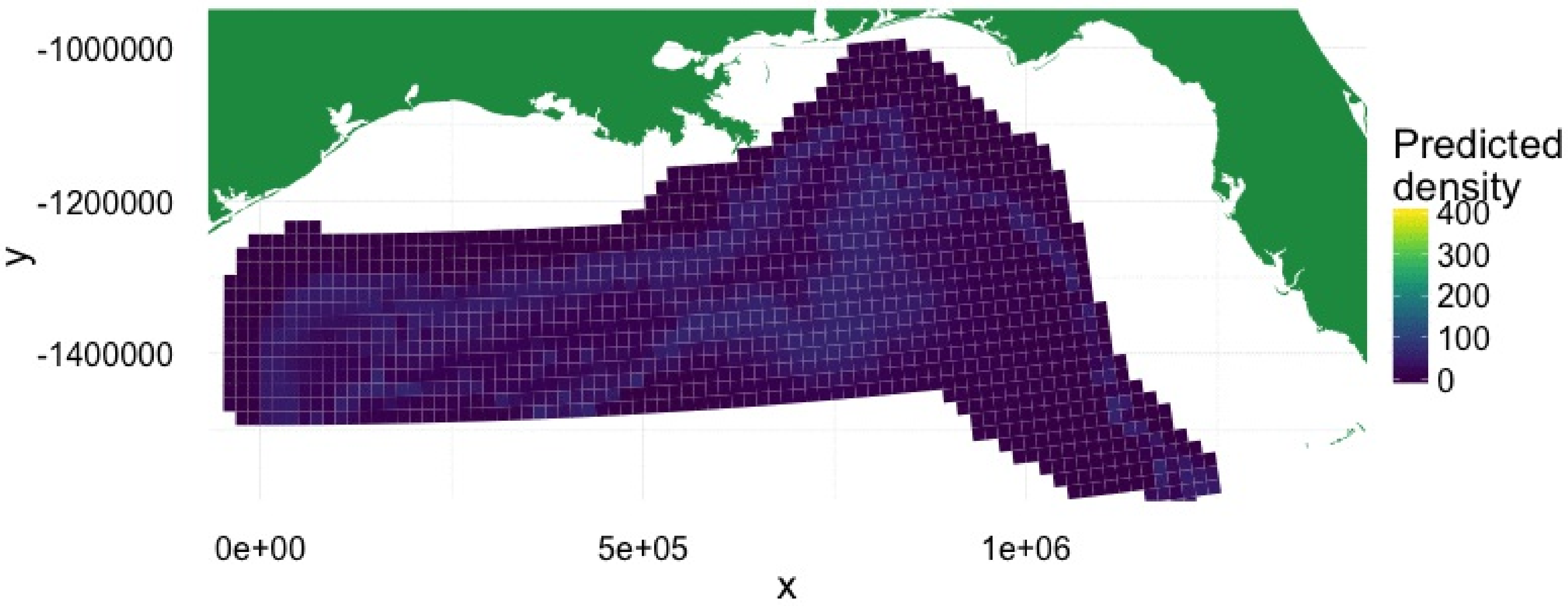
$$\mathrm{V}_{\hat{\boldsymbol{\eta}}} = \mathrm{L}_\mathrm{p}^\mathrm{T}\mathrm{V}_{\hat{\boldsymbol{\beta}}}\mathrm{L}_\mathrm{p}$$

pre-/post-multiplication shifts the variance matrix from parameter space to linear predictor-space.

(Can then pre-/post-multiply by derivatives of the link to put variance on response scale)

Simulating parameters

- $\boldsymbol{\beta}$ has a distribution, we can simulate



Uncertainty in smoothing parameter

- Recent work by Simon Wood
- "smoothing parameter uncertainty corrected" version of $V_{\hat{\boldsymbol{\beta}}}$
- In a fitted model, we have:
  - `$Vp` what we got with `vcov`
  - `$Vc` the corrected version

Variance summary

- Everything comes from variance of parameters
- Need to re-project/scale them to get the quantities we need
- `mgcv` does most of the hard work for us
- Fancy stuff possible with a little maths
- Can include uncertainty in the smoothing parameter too

Summary

- `predict` is your friend
- Most stuff comes down to matrix algebra, that `mgcv` sheilds you from
  - To do fancy stuff, get inside the matrices