Generalized linear model validation

Bert van der Veen

Department of Mathematical Sciences, NTNU

The model

Writing the linear model:

$$y_i = \alpha + \mathbf{x}_i\boldsymbol{\beta} + \epsilon_i \sim \mathcal{N}(0, \sigma^2) \tag{1}$$

Is the same as:

$$\mathbb{E}(y_i|\mathbf{x}_i) = \alpha + \mathbf{x}_i\beta \tag{2}$$

as long as $\mathbb{E}(\epsilon_i) = 0$.

Generalised linear model

$$
\begin{aligned}
g\{\mathbb{E}(y_i|x_i)\} &= \eta_i = \alpha + x_i\beta \\
\mathbb{E}(y_i|x_i) &= g^{-1}(\eta_i) = g^{-1}(\alpha + x_i\beta)
\end{aligned}
\tag{3}
$$

GLMs do not have a residual term

GLM Assumptions

- No outliers
- Independence
- Correct distribution
- Correct link function
- Correct variance function (implied by previous two)

Outline

- Residual plots for checking GLM assumptions
- Issues with residuals in GLMs
- Types of residuals
- Residual diagnostics
- Dharma/quantile residuals
- Prediction error

Residual diagnostics in GLMs

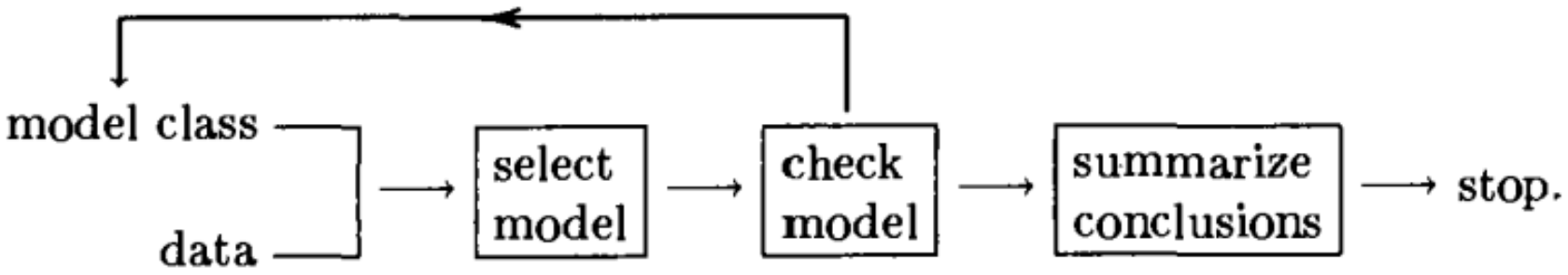


Figure 1: McCullagh and Nelder (1989) workflow

Methods for checking GLM assumptions

- ~~Tests~~
- Outliers: Cook's distance, Residual vs. fitted
- Independence: Partial residual plots, Residuals vs. fitted, Rresiduals vs. lagged residuals
- Correct distribution: QQ-plot
- Correct link function: residuals vs. fitted, include η_i in the model (Hinkley 1985) , linear predictor against transformed response
- Correct variance function: Abs(residuals) vs. fitted

Response residuals

We could use the same residual as in linear regression:

$$\epsilon_i = y_i - \hat{\mu}_i$$

But we do not expect these to look nice in GLMs.

- Mean depends on variance
- We would rather have nice looking residuals (when assumptions are not violated)

Defining GLM residuals: Pearson's

We do not have a ϵ_i term in GLMs, but we still calculate the residual similarly:

$$\epsilon_{i,pearson} = \frac{y_i - \hat{\mu}_i}{\sqrt{\text{var}(y_i; \mu_i, \phi)}} \tag{4}$$

as the scale difference between data and mean.

Approximately normally distributed in large samples

Recall deviance?

We do not have a ϵ_i term in GLMs, but we do have the deviance function:

$$D(\mathbf{y}; \hat{\boldsymbol{\mu}}) = \sum_{i=1}^{n} 2y_i\{g(y_i) - g(\hat{\mu}_i)\} - y_i + \hat{\mu}_i \tag{5}$$

Which represents distance to the saturated model.

Defining GLM residuals: deviance

$$D(\mathbf{y}; \hat{\boldsymbol{\mu}}) = \sum_{i=1}^{n} 2y_i\{g(y_i) - g(\hat{\mu}_i)\} - y_i + \hat{\mu}_i \tag{6}$$

- Note the summation over the observations
- We can split this per observation!
- $d_i = 2y_i\{g(y_i) - g(\hat{\mu}_i)\} - y_i + \hat{\mu}_i$

Deviance residuals

$$\epsilon_{i,deviance} = \text{sign}(y_i - \hat{\mu}_i)\sqrt{d_i}, \qquad \text{so that } \sum_{i=1}^{n} \epsilon^2_{deviance,i} \tag{7}$$

Approximately normally distributed in large samples

Why deviance residuals

- Converges faster to approximate normality

  (Dunn and Smyth 2018, Cox and Snell 1968)
- Otherwise, adjusted deviance residual
- Good for small samples (Pierce and Schafer 1986)
  - e.g. Poisson $\mu_i^{-0.5}/6$

Still inappropriate for discrete data and small samples

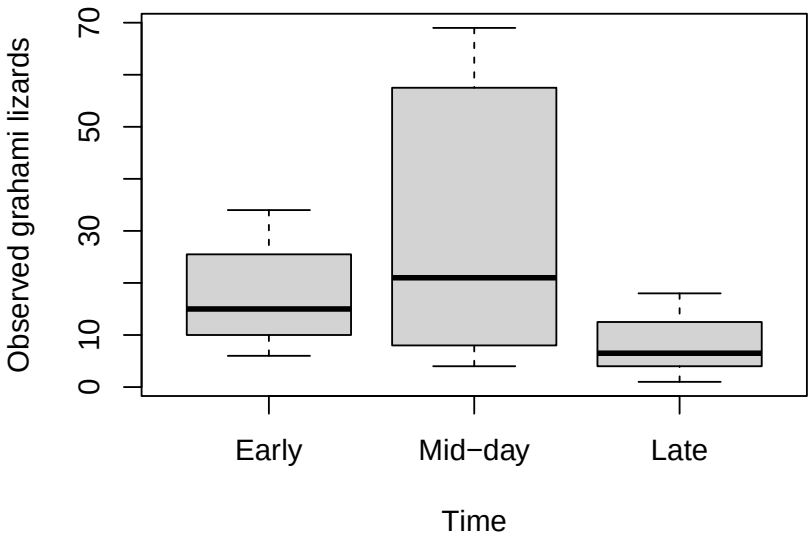
In practice

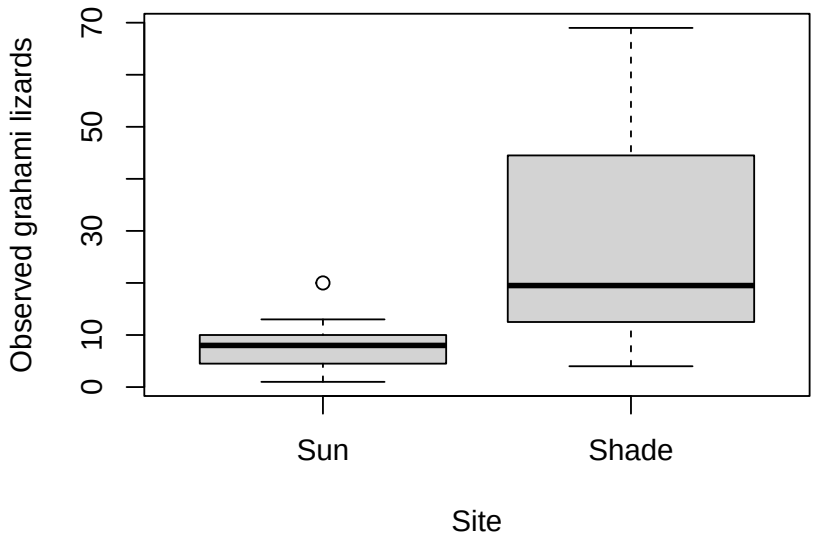
In practice, both Pearson's and Deviance residuals are often non-normally distributed.

Trying it out

Figure 2: nwf.org

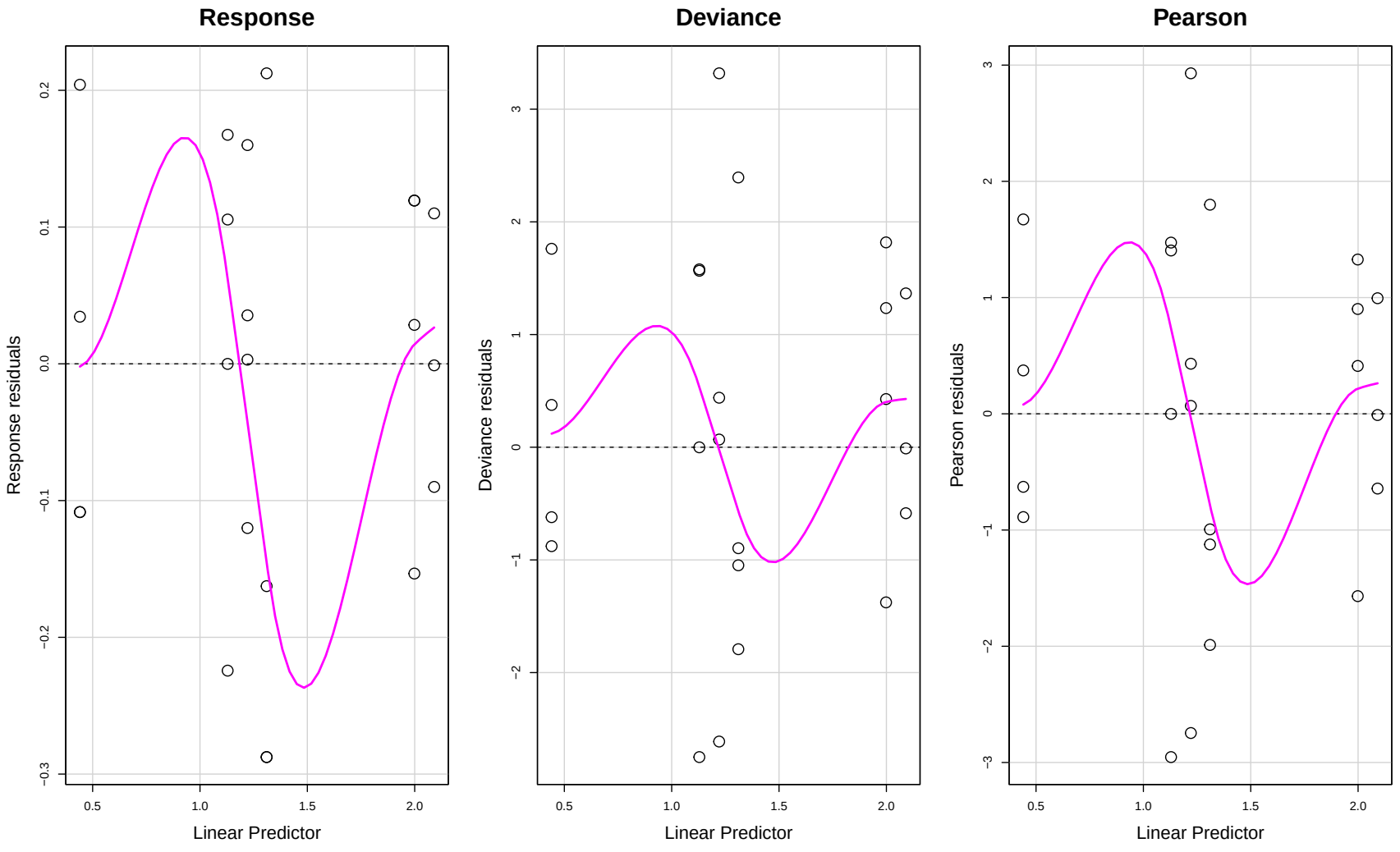
Lizards: recap



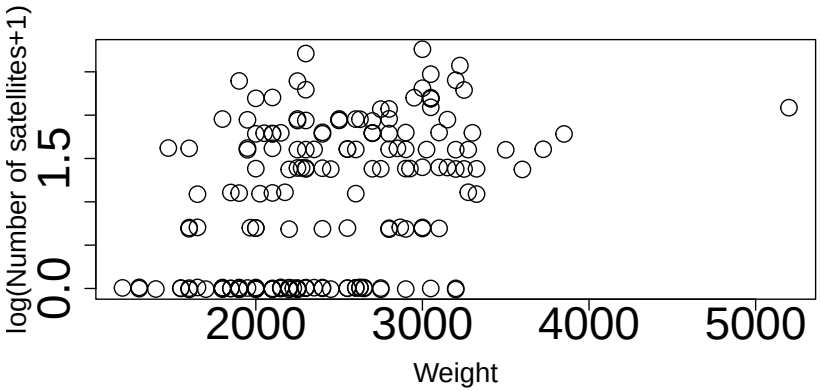


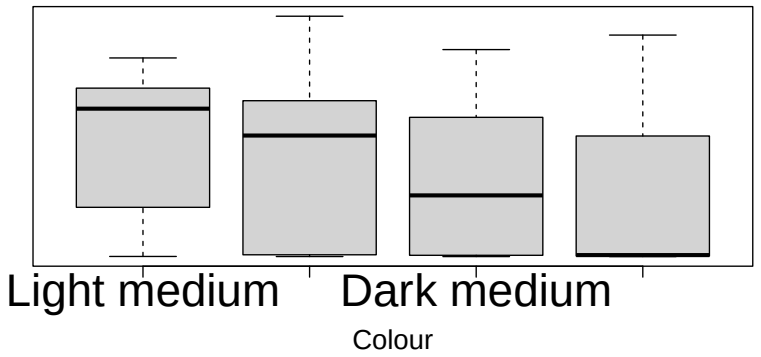

```
model1 <- glm(cbind(grahami, opalinus)~Time+Site,
              data = lizards, family="binomial")
```

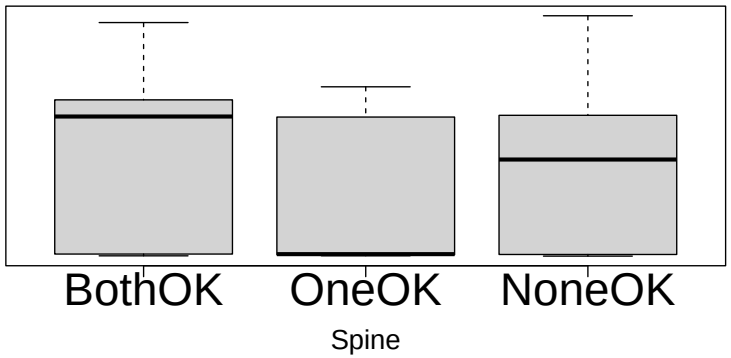
Lizards: residuals

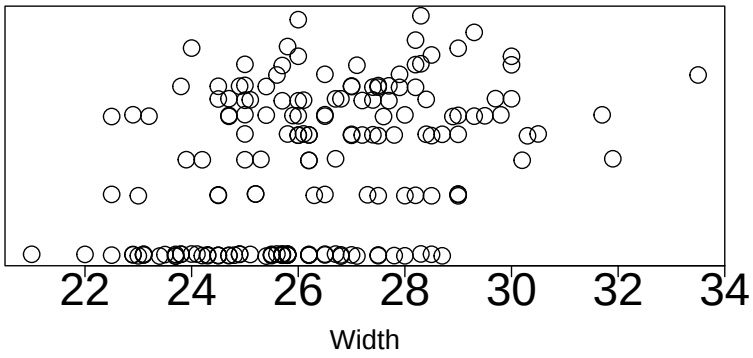


Horseshoe crabs: recap



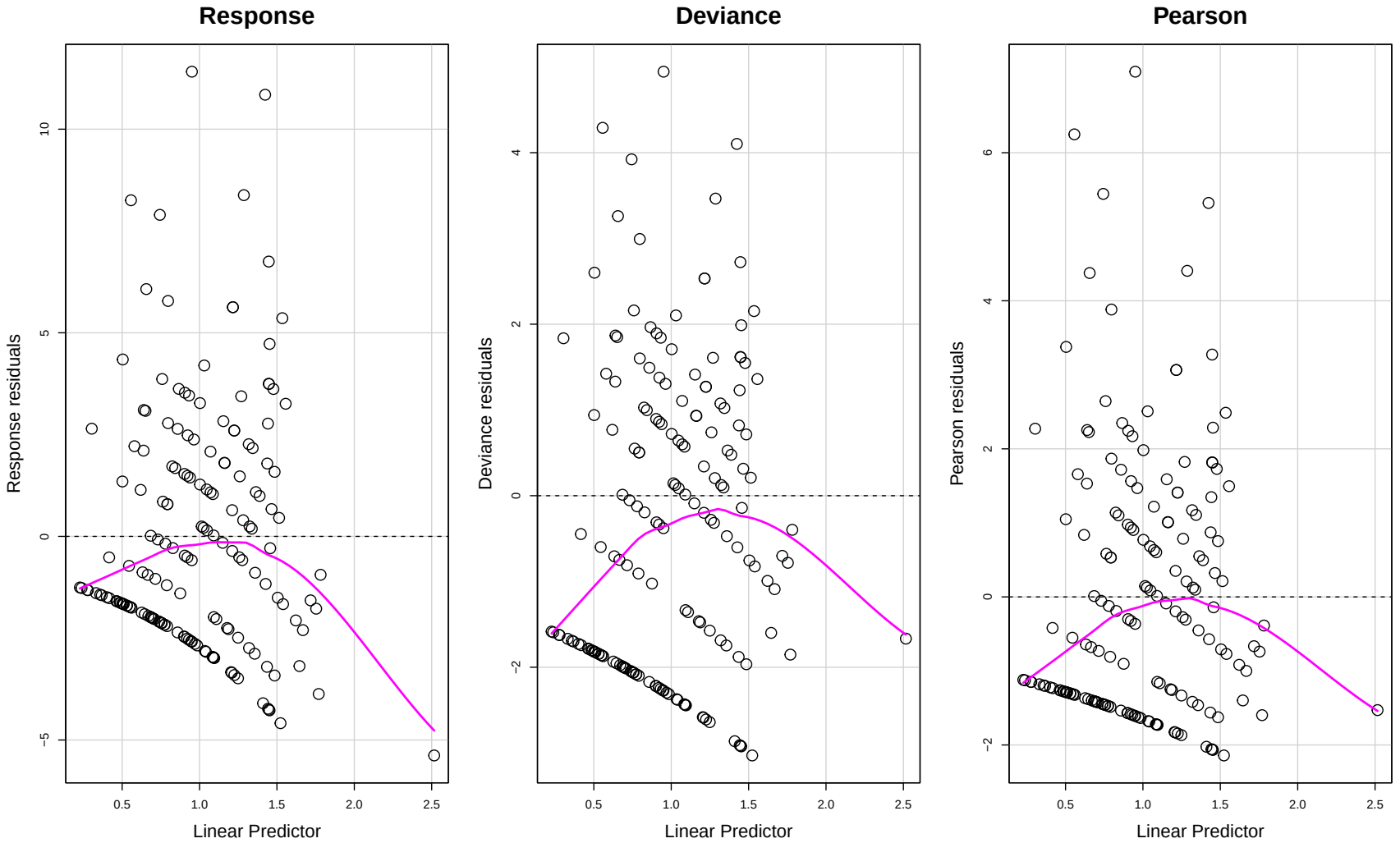




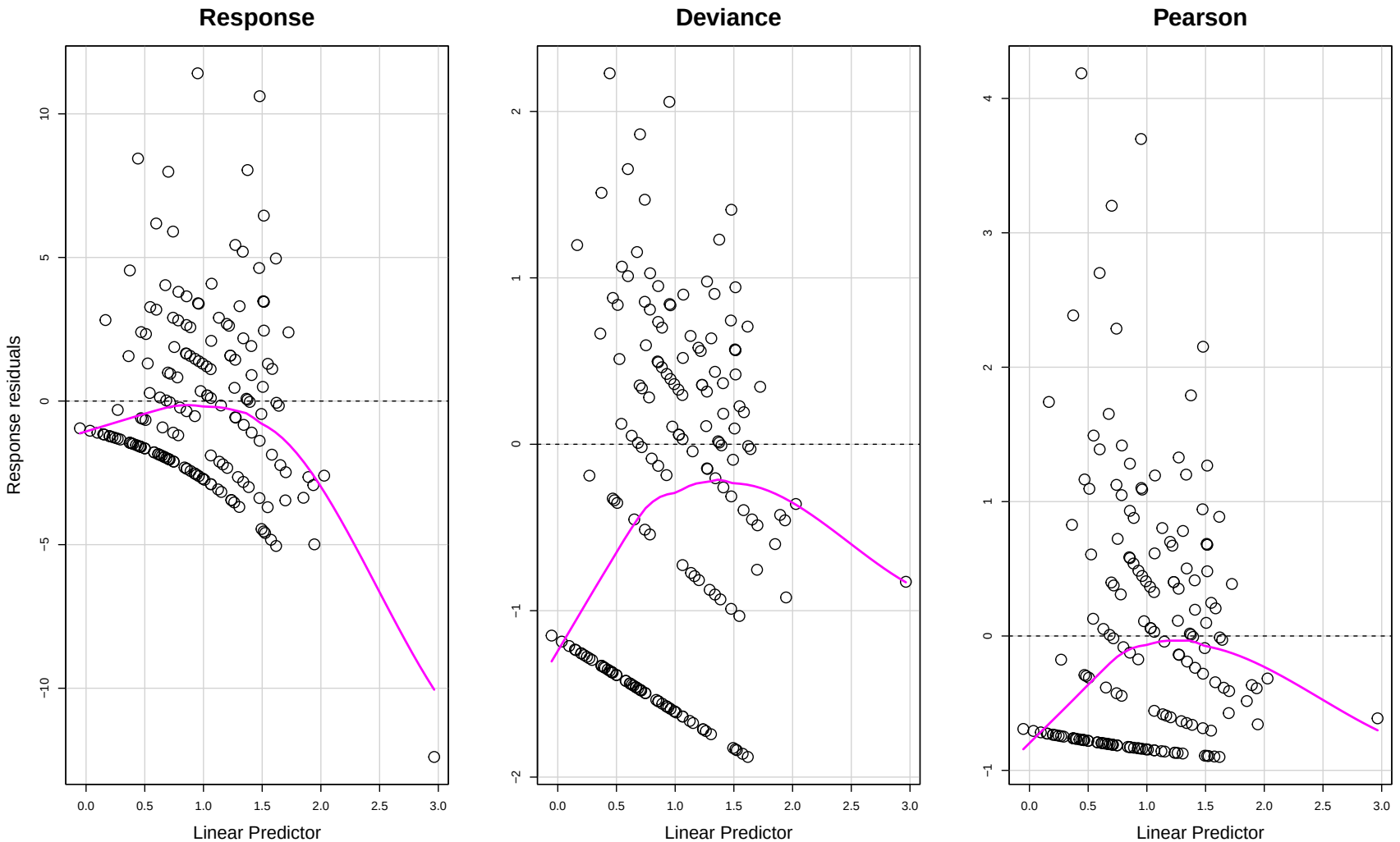



```
model2 <- glm(Sat ~ Spine + Colour + Width + Weight,
              family = "poisson", data = hcrabs)
```

Horseshoe crabs: Poisson model residuals



Horseshoe crabs: NB model residuals

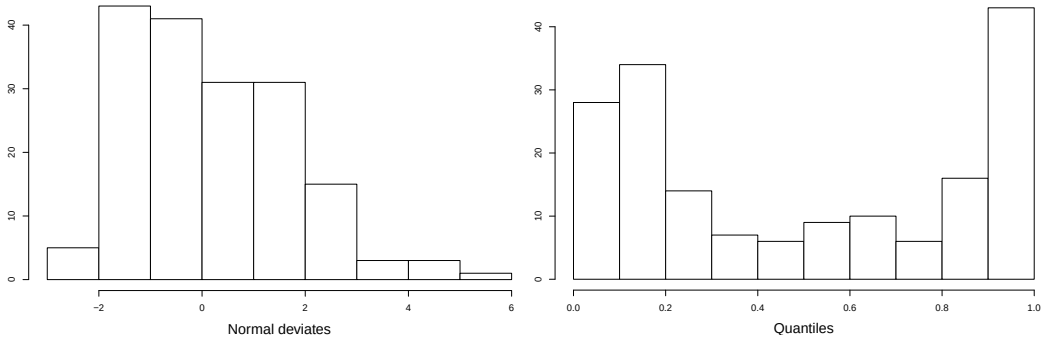


Randomized Quantile residual (Dunn and Smyth 1996)

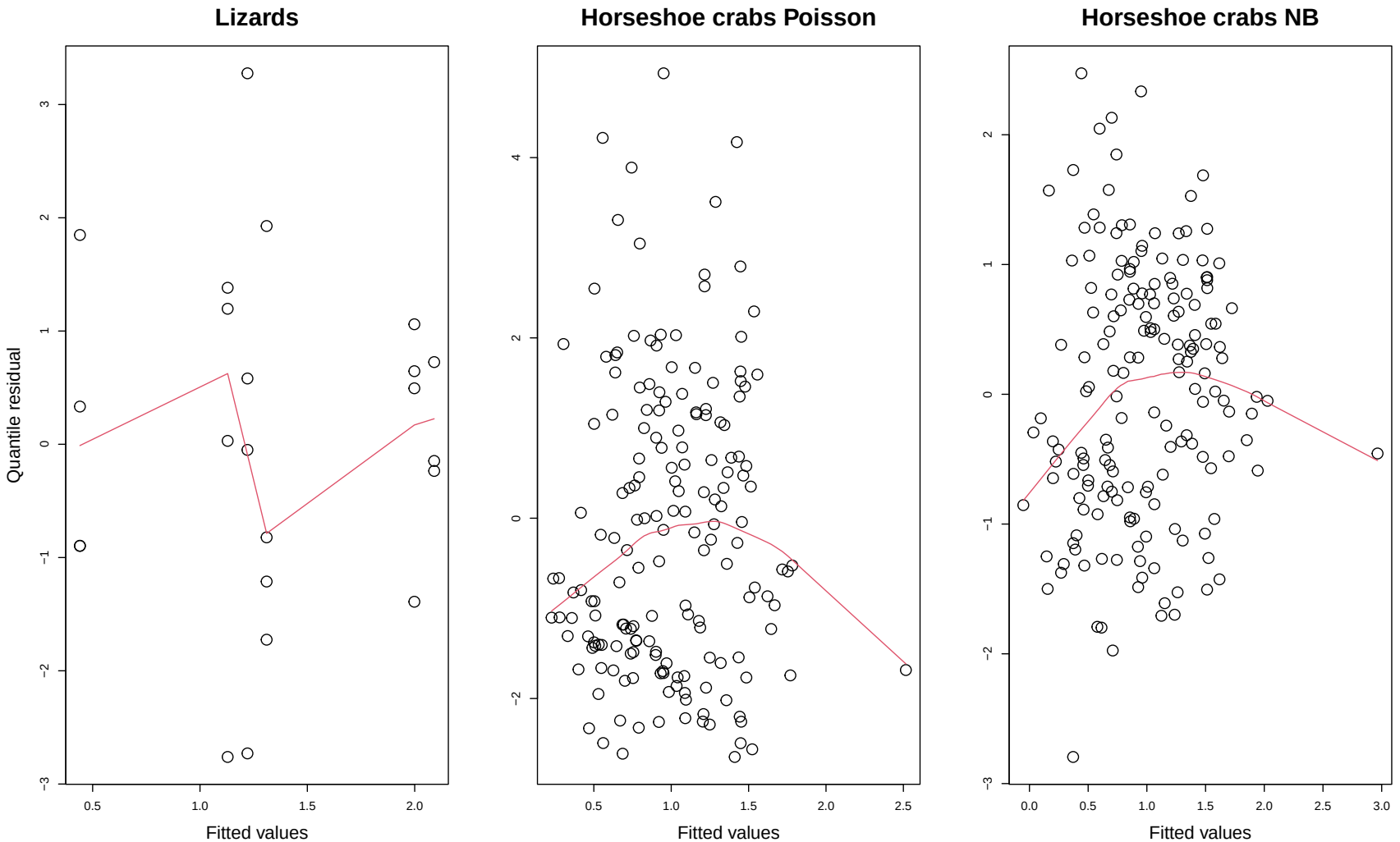
- Gold standard residual
- Better suited for small samples and discrete data types
- Exactly normally distributed
- Suitable for all kinds of models

Continuous

$$r_Q = \Phi^{-1}\left\{\mathcal{F}\left(y_i; \hat{\mu}_i, \hat{\phi}\right)\right\} \tag{8}$$



Lizards and Horseshoe crabs



Prediction

When you are only interested in predictions, e.g.:

- 20 years from now
- on a map

Some assumptions might not matter. Generally, we still do not want structural deviations from the model.

Finding a model that predicts well

Usual procedure:

1) Fit model to a part of the data ("train" and "test")
2) Predict for the remaining part ("test")
3) Quantify prediction "error" (e.g., RMSE)

Statistically leaving out data is weird. We want as much information for our model as possible.

Quantifying prediction error

MSE: $\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_i)^2$

RMSE: $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \mu_i)^2}$ MAD: median$(|\mathbf{y} - \boldsymbol{\mu}|)^2$

Or another metric. "AUC" is often used in binomial models.

For prediction, we want a model that performs well on one of these metrics.

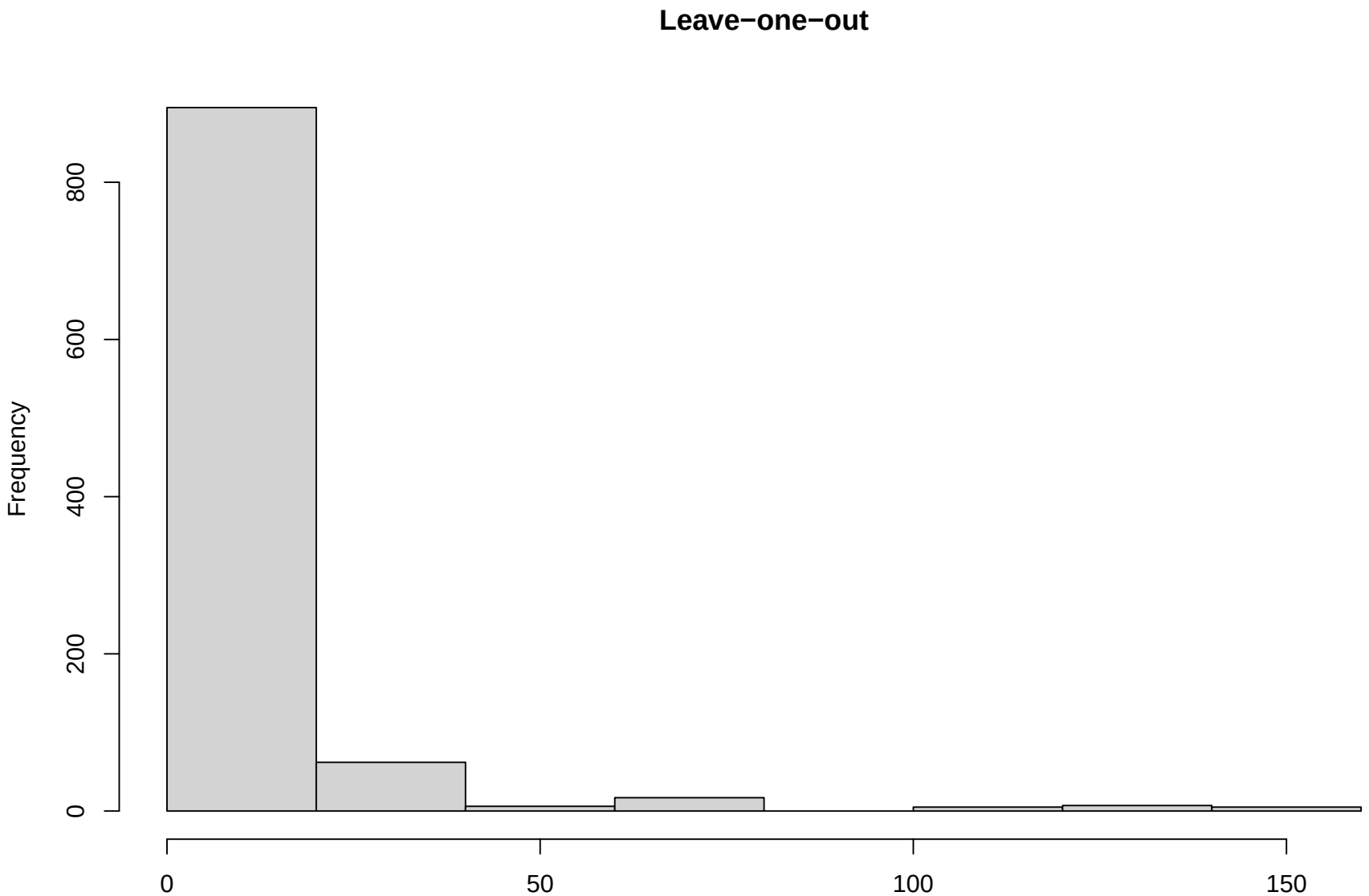
"Test" data is not external/independent

- Test data is usually collected under the same circumstances
- It is not really external!
- Better procedure: fit model to all data, collect new data for testing

Leave-one-out cross-validation Horseshoe crabs

```
SE <- NULL
for(i in 1:1000){
set.seed(i)
test <- sample(1:nrow(hcrabs), size = 1)
train <- hcrabs[-test,]
model <- update(model3, data = train)
SE <- c(SE,(train[test, "Sat"]-
  predict(model,newdata = train[test, ], type="response"))^2)
}
```

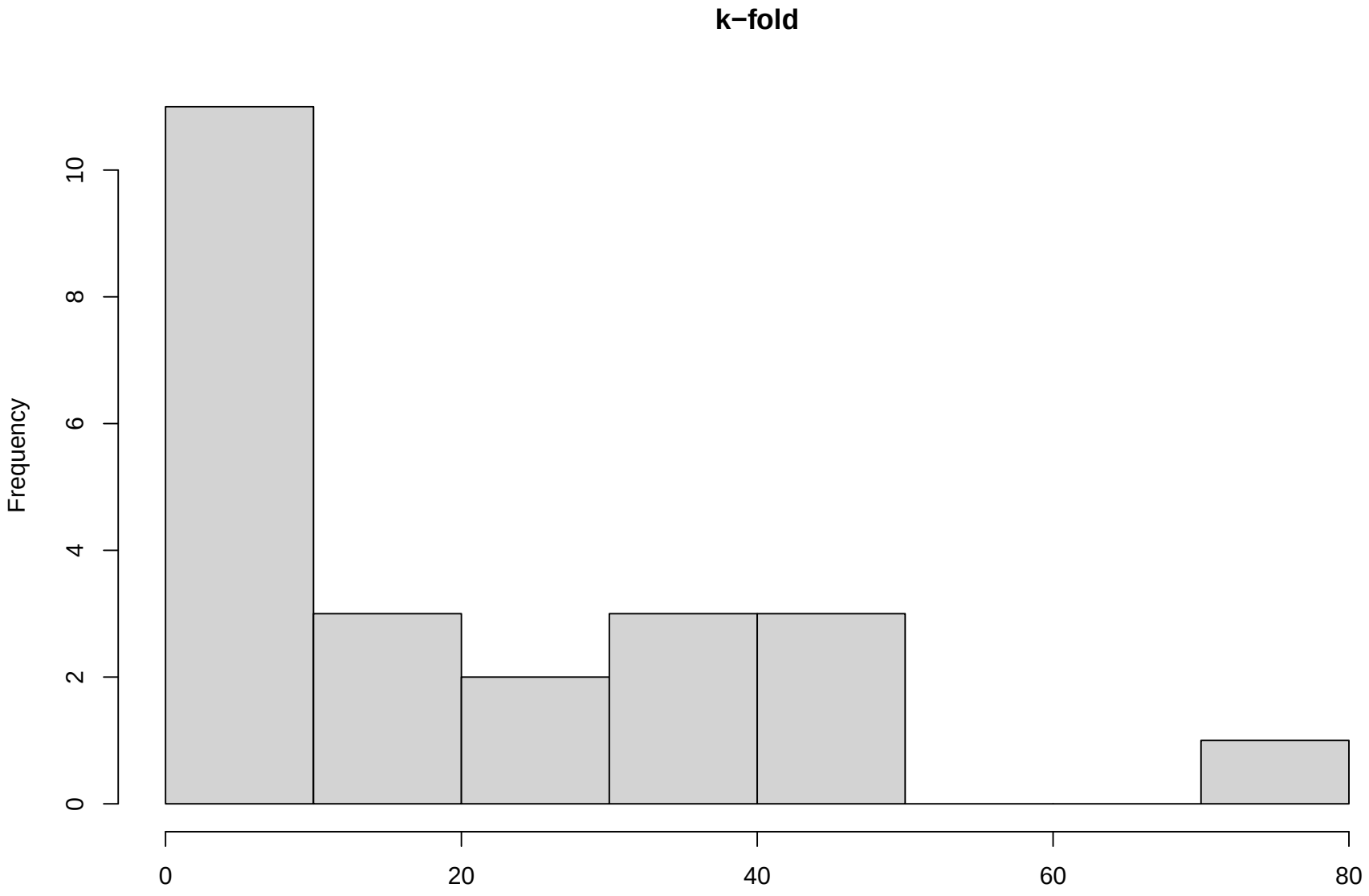
Result



K-fold cross-validation Horseshoe crabs

```
k = 5
SE <- NULL
shuffle <- sample(1:nrow(hcrabs), nrow(lizards))
for(i in seq(1,nrow(hcrabs),by=k)){
set.seed(i)
test <- i:min(i+k-1,nrow(hcrabs))
train <- hcrabs[shuffle, ][-test,]
model <- update(model3, data = train)
SE <- c(SE,(hcrabs[shuffle, ][test, "Sat"]-
  predict(model,newdata = hcrabs[shuffle, ][test, ], type="re
}
```

Result



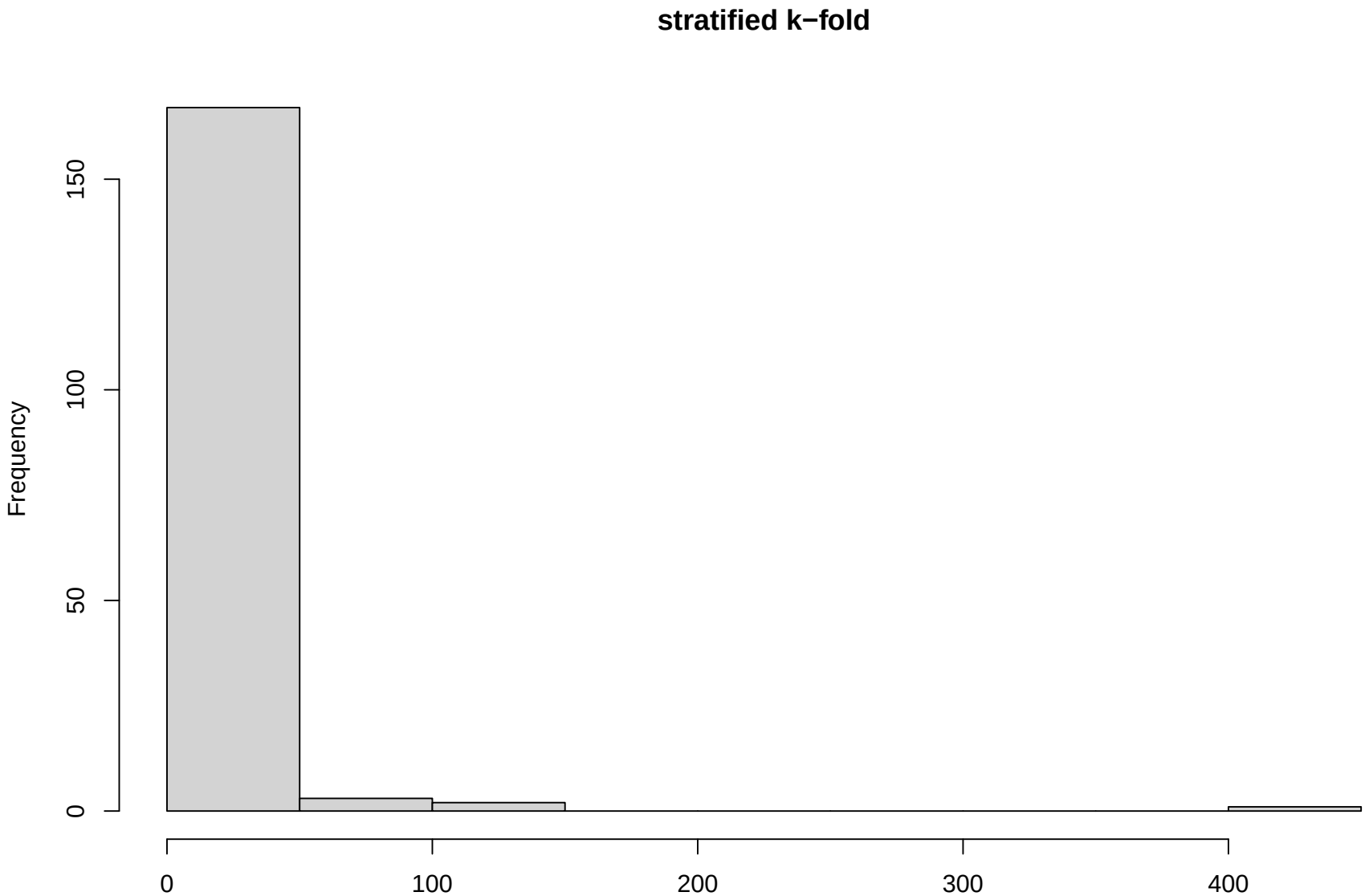
Stratified cross-validation

- Cross-validation in a structured fashion: time or space.
- Select roughly equal partitions
  - Time: fit to a few years per partition
  - Space: areas
- I.e., blocking for non-independent responses

Stratified k-fold cross-validation Horseshoecrabs

```
k = 5
SE <- NULL
for(i in seq(1,nrow(hcrabs),by=k)){
set.seed(i)
test <- i:min(i+k-1,nrow(hcrabs))
train <- hcrabs[order(hcrabs$Weight),][-test,]
model <- update(model3, data = train)
SE <- c(SE,(hcrabs[order(hcrabs$Weight),][test, "Sat"]-
predict(model,newdata = hcrabs[order(hcrabs$Weight),][test, ], type="respo
}
```

Result



Summary

- I did not mention Anscombe residuals, nor working residuals
- Do not draw conclusions before validating your model
- Use residual diagnostics/plots!
- If violated, adjust the model
- For prediction, look at cross-validation
  - Consider the structure of the data
  - Use independently collected data
  - There are many forms of cross-validation and quantifying error
  - Find model that minimizes the prediction error

Useful packages: `car`, `DHARMa`, `boot`