Dealing with complicated data in time series models

Eric Ward

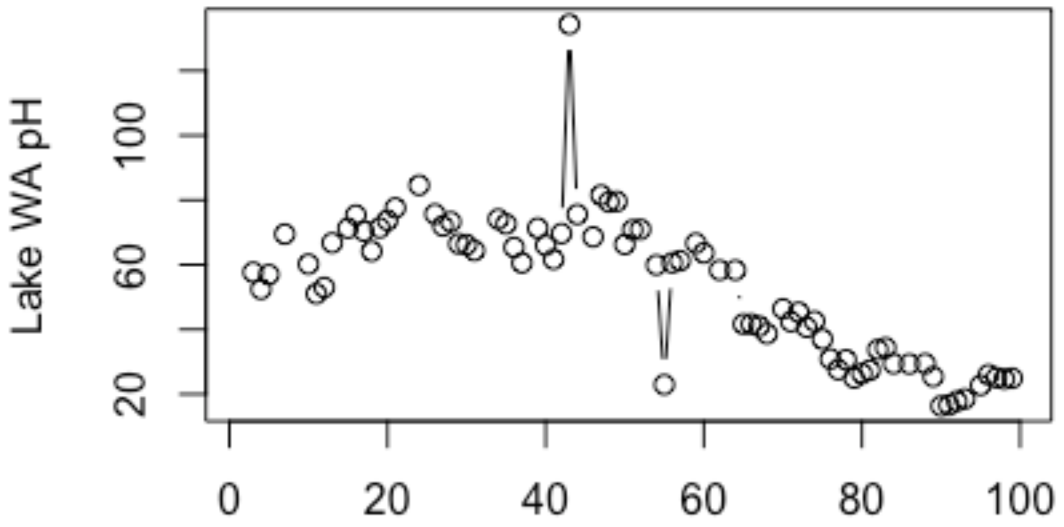
Feb 16 2017

Missing data

- Data may be missing (NAs) in response or predictors
- Some models (functions) require complete datasets
- Variety of interpolation methods

Example dataset

- Lake WA pH



Simple approach

- Linear interpolation

[3, NA, 4.3, 5.4, NA, 6.1]

- 2nd observation becomes (3+4.3)/2
- 5th observation becomes (5.4+6.1)/2

- Trickier when more data is missing,

[NA, NA, NA, 5.4, NA, 6.1]

Linear interpolation for longer gaps

library(zoo)

na.approx() function

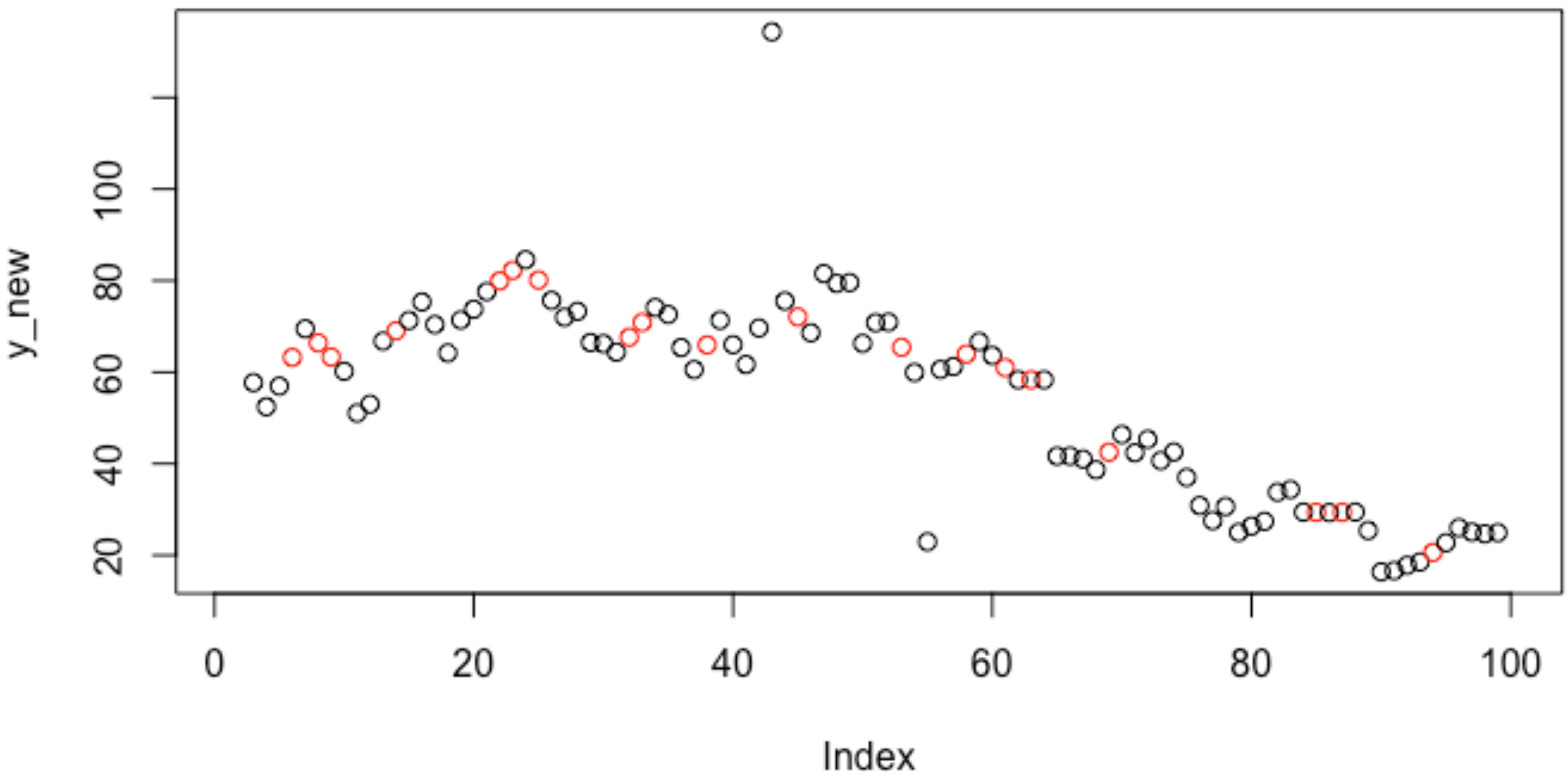
Example for our Lake WA data:

y_new = y

y_new[is.na(y)] = na.approx(y,na.rm=FALSE)
[is.na(y)]

* Missing values filled in via linear interpolation

Imputed values in red



Two useful extensions

maxgap

```
na.approx(y,na.rm=FALSE, maxgap = 3)
```

rule (inherited from approx()): should data be extrapolated (2) or not (1)

```
na.approx(y,na.rm=FALSE, maxgap = 3, rule=2)
```

Data point closest is used for extrapolation

Alternative linear interpolation

- Use rollmean() function in zoo library to customize the rolling window

```
rollmean(y, k=3, align = c("center", "left",
"right"))
```

Alternatives to linear interpolation

- Splines (think GAMs)

- Fit the spline yourself

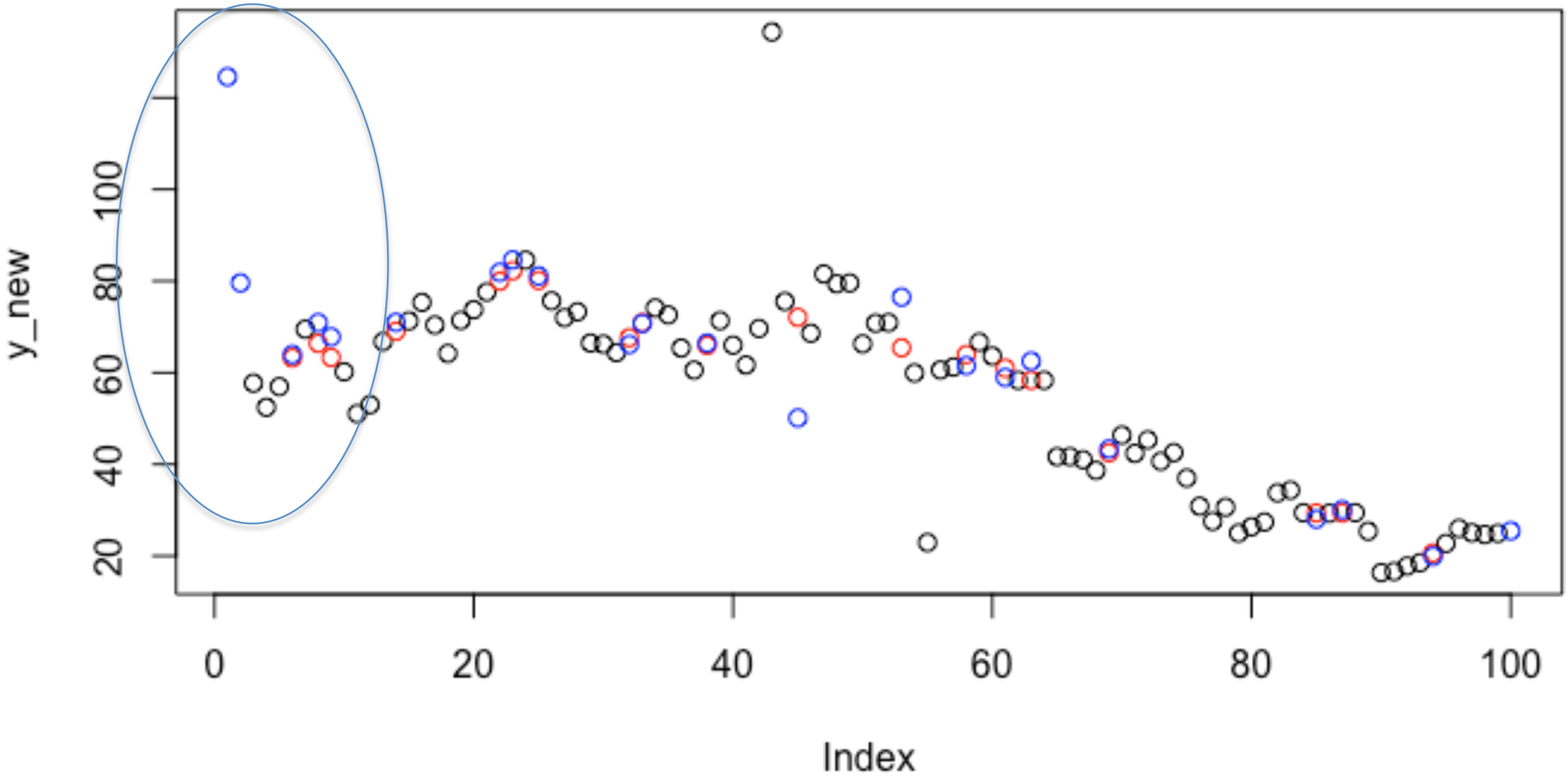
- Or use other 'na' functions

na.locf() – last observation carried forward

na.spline() – interpolate with spline

Spline(blue), linear (red)

- Beware near end of time series / near lots of missing data



Other interpolation approaches

- imputeTS: wrapper for approx(), contains other splines

- Splines:

smooth.spline()

gam (mcv)

Is interpolating a good idea?

- One consequence is falsely increased precision

- As example we can fit a univariate state space model (with MARSS) to the three datasets:
- Raw (including missing values)
- Interpolated (linear)
- Interpolated (spline)

How do variances compare?

- Linear interpolation = low observation error

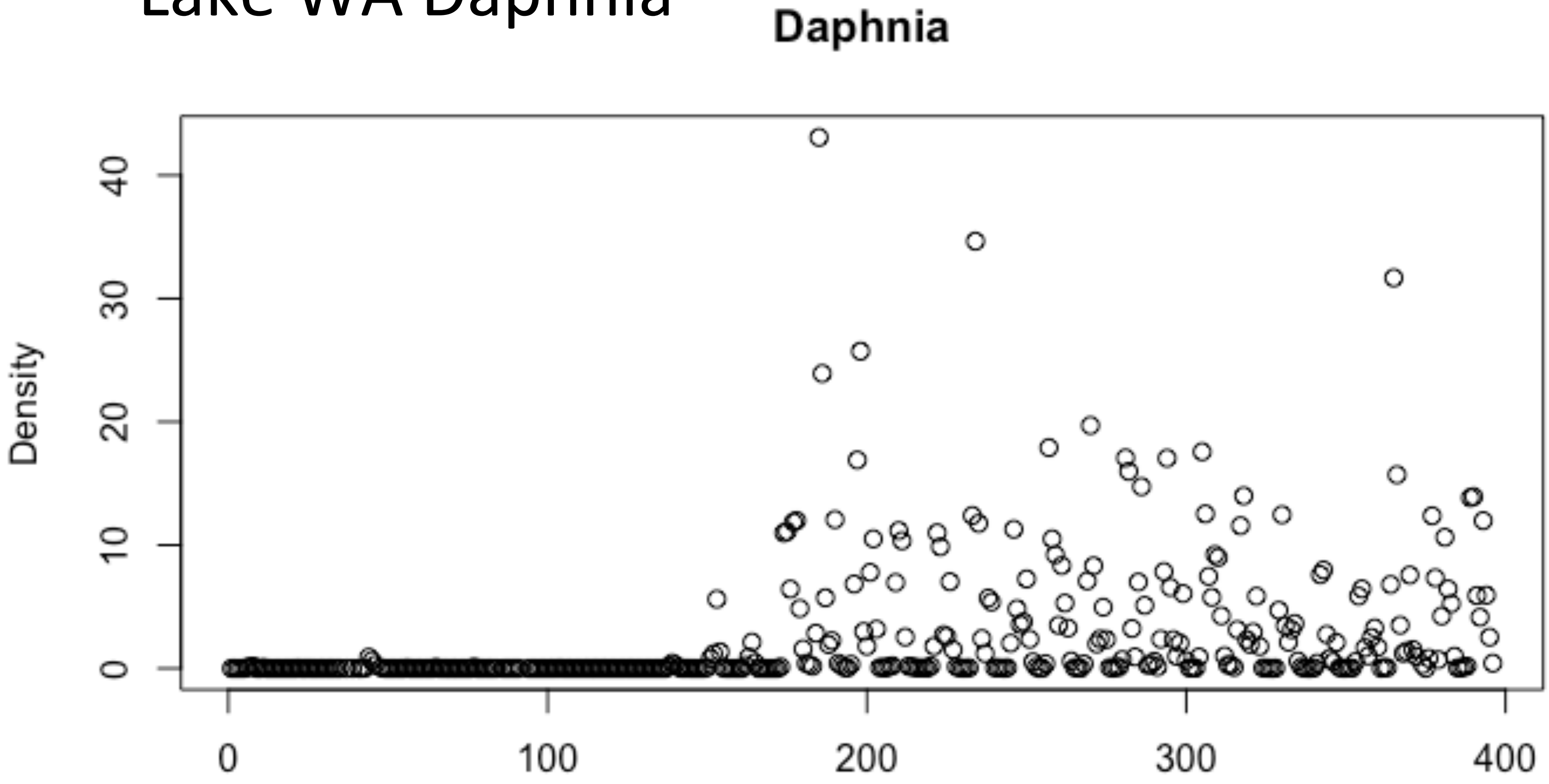
Model	Q	R
Raw	0.0161	0.0148
Linear interp	0.01079	0.01389
Spline interp	0.0161	0.0148

Missing covariates

- Generally, missing covariates can be more of a problem (e.g. state space models, DFAs, etc)
- Same approaches can be used for interpolation
- Or in a Bayesian framework, the missing values can be assigned priors

When missing data are zeros

- Lake WA Daphnia



Options

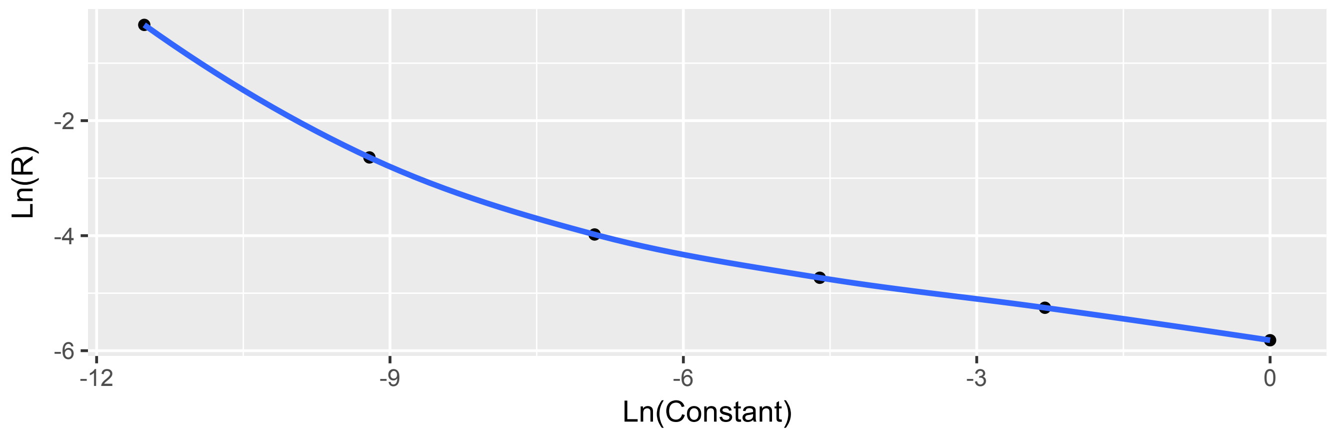
- Throw out observations
- Transform your data
- Work with more complicated statistical models

Data transformations

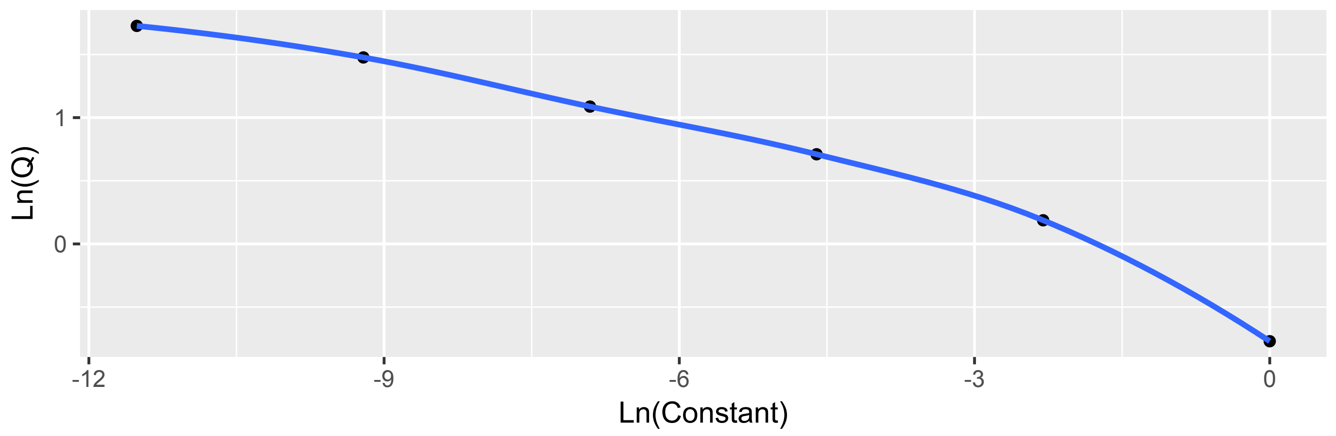
- Ln(y + small number) is one of the more common approaches
- BUT choice of small number has impact on results
- What is adding a constant going to do to observation or process variances?

Adding constants to Daphnia

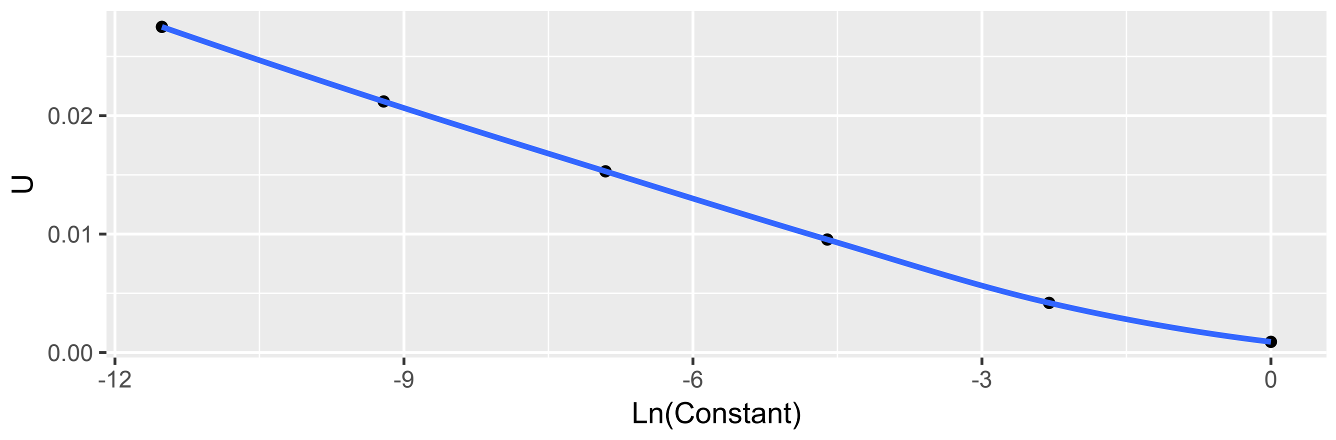
Constant	**R**	**Q**	**U**
1	0.002973	0.462834	0.000901
0.1	0.00523	1.20592	0.00419
0.01	0.00881	2.03384	0.00954
0.001	0.0187	2.9671	0.0153
0.0001	0.0714	4.3777	0.0212
0.00001	0.7155	5.62	0.0275



Ln(R)
-2
-4
-6
-12
-9
-6
-3
0
Ln(Constant)



Ln(Q)
1
0
-12
-9
-6
-3
0
Ln(Constant)



U
0.02
0.01
0.00
-12
-9
-6
-3
0
Ln(Constant)

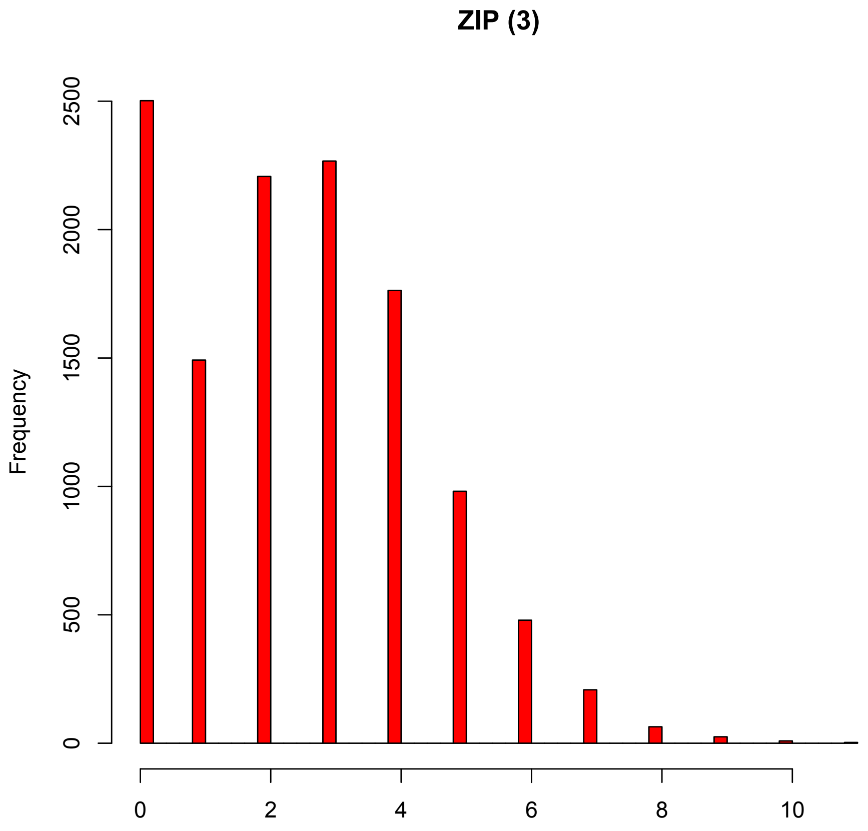
Alternate statistical distributions

- Use time series (MARSS) or other statistical model for positive values > 0

- Apply logistic regression (or more complicated model) to model zeros

Delta-GLMs

- Density of marine fishes almost always fits this pattern (zero inflated)



Delta-GLM or ‘hurdle models’

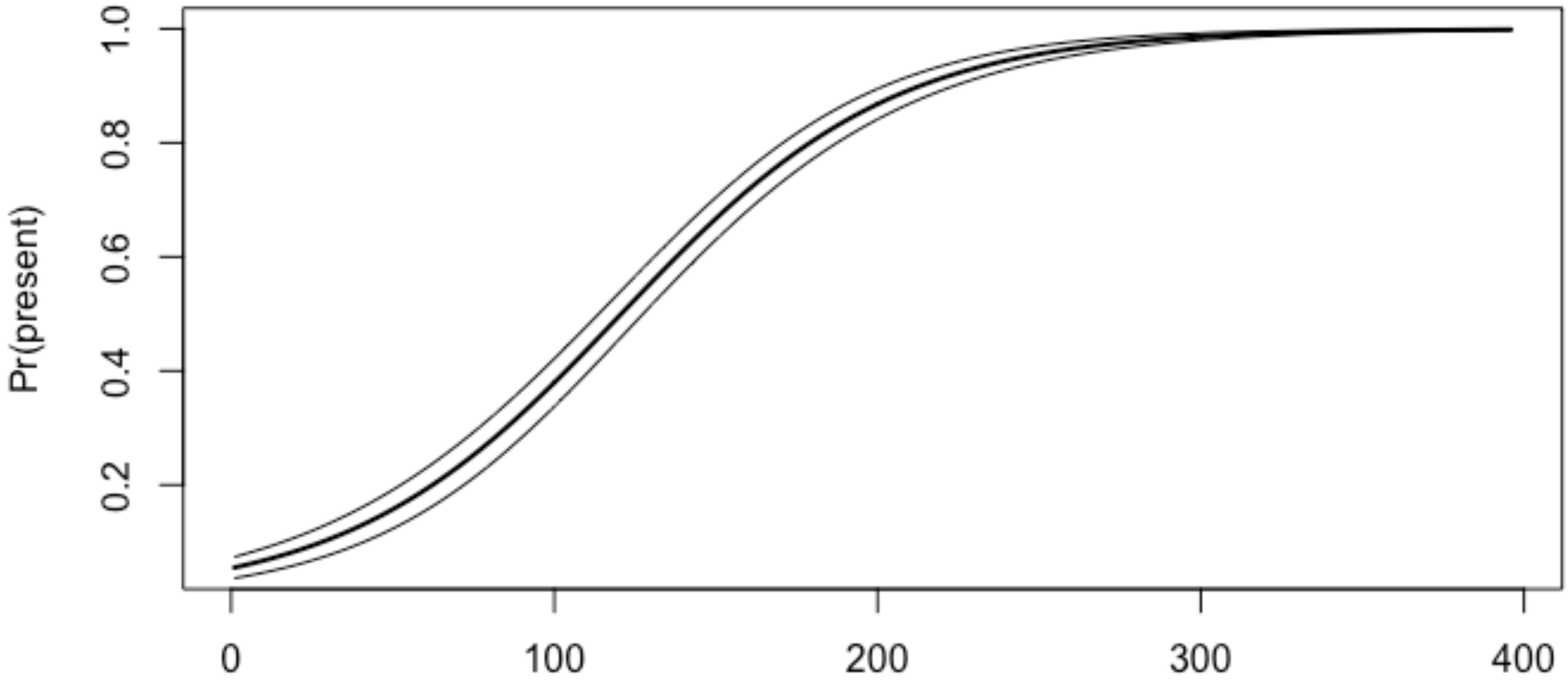
- Breaks the response into 2 parts
  - Presence / absence
  - Positive density
- 2 separate GLMs
  - May include different covariates
- If we include random effects / shared terms, they usually aren’t correlated across models
  - Different data + different link functions = weird interpretation
  - Results from both models combined for estimates of total density

Example with Daphnia

Example code for GLM

```
y_int = ifelse(y > 0, 1, 0)
mod = glm(y_int~seq(1,length(y_int)),
family="binomial")
pred = predict(mod,
newdata=data.frame(1:396), type="response",
se.fit=T)
```

Probability of Daphnia



Positive model

- Convert 0s to NAs

```
y[which(y==0)]=NA
```

- Fit MARSS model

```
mod = MARSS(log(y))
```

- Predictions (log space)

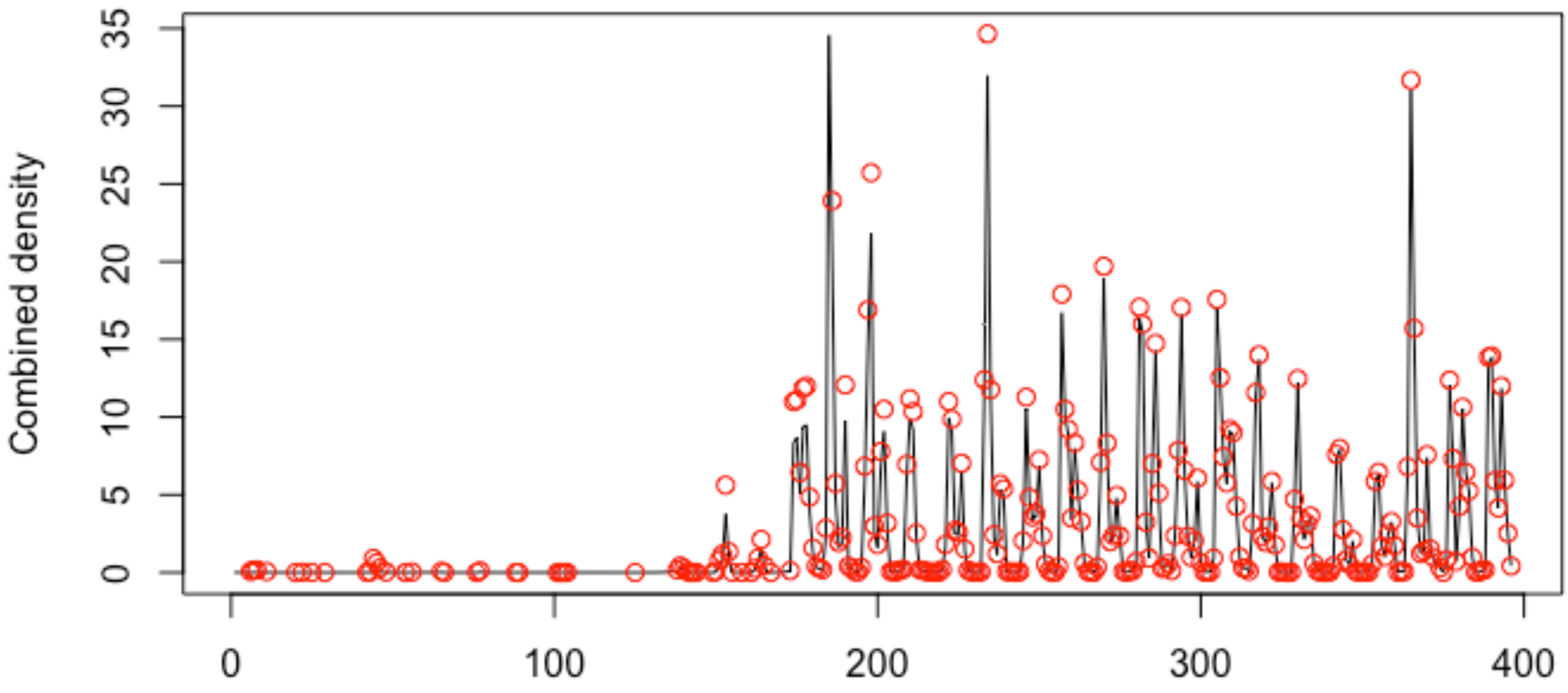
```
exp(c(mod$states))
```

Estimating total density

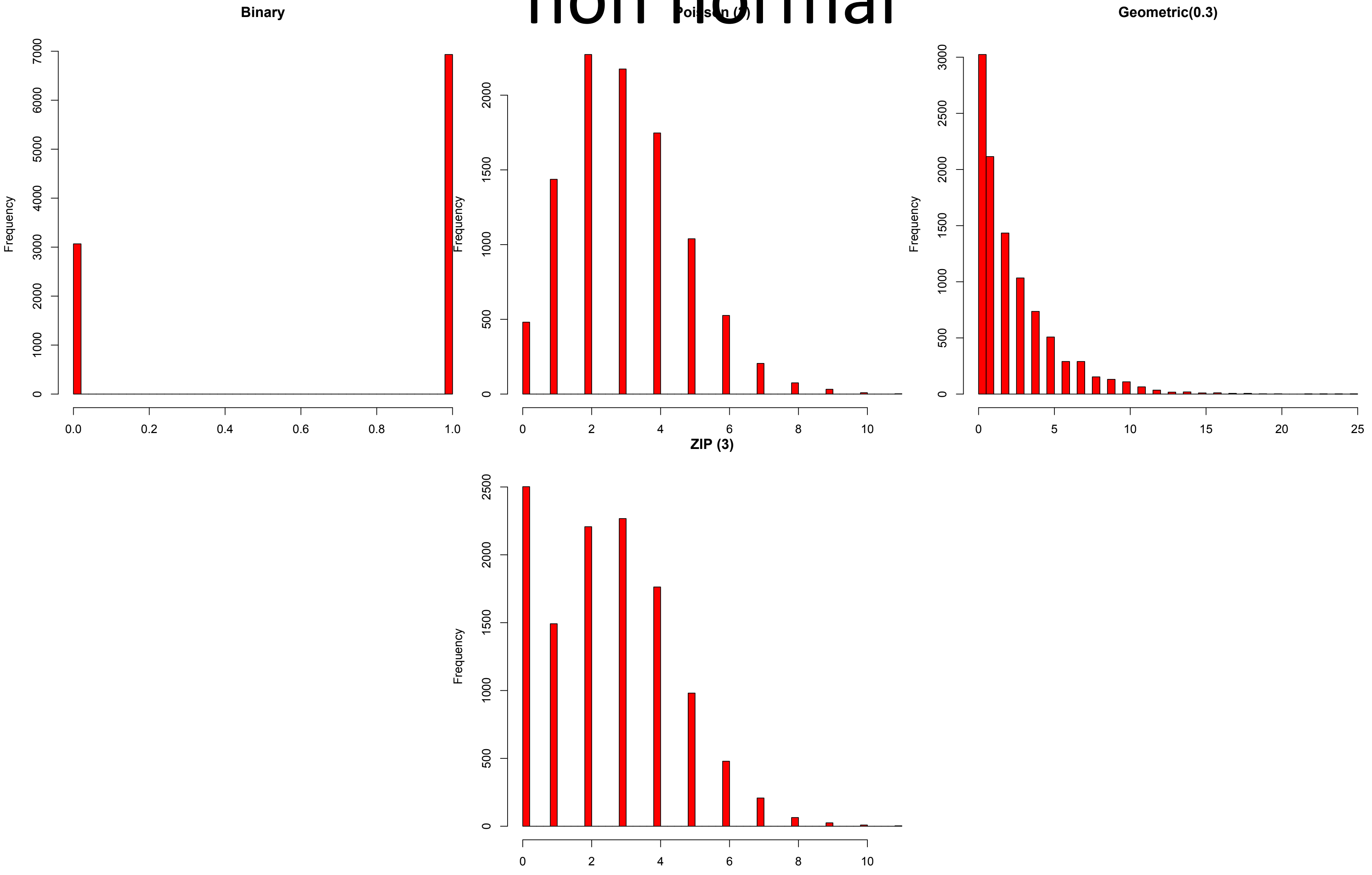
- Multiply Pr(present) * E[density|present]

- Standard errors more complicated
  - Delta method
  - Monte Carlo simulation, etc

Combined model: E(present) * E(pos|present)



Other types of response data may be non normal



We can model the response as function of predictors using link functions

- Defaults in GLMs

- Binomial data (logit link)

$$logit(p_i) = \log\left(p_i/(1-p_i)\right) = BX_i$$

- Poisson, Negative Binomial, Gamma, Lognormal (log link)

$$log(u_i) = BX_i$$

- Note that these formulas don't include additional error (like regression)

GLMMs

- Including additional variation turns GLMs -> GLMMs

$$logit(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = BX_i + e_i$$

$$log(u_i) = BX_i + e_i$$

$$e_i \sim Normal(0, \sigma)$$

- More data hungry, but flexible
  – Random effects allow us to turn ordinary GLMMs into time series models or models with spatial effects

Where have we seen this before?

$$X_{t+1} = BX_t + e_t$$

$$e_t \sim Normal(0, q)$$

$$logit(p_t) = X_t$$

$$Y_t \sim Bernoulli(p_t)$$

- We could construct a DLM with binomial response (or any other distribution)

Univariate -> multivariate

- For population *i*

$$X_{i,t+1} = BX_{i,t} + e_{i,t}$$

- As in MARSS models, we need to think about how to model the deviations
  - Independent and shared variance across pops?
  - Independent and unique variance across pops?
  - “equalvarcov”
  - Unconstrained
  - Model covariance as spatially correlated

Powerful functions for estimating non-normal response data

- rstanarm
  - Extension of Bayesian regression, GLMs, GLMMs, etc
- glmmTMB (or lme4, bbmle, etc)
  - Fast maximum likelihood estimation
- Same formula syntax as glm(), lm(), etc

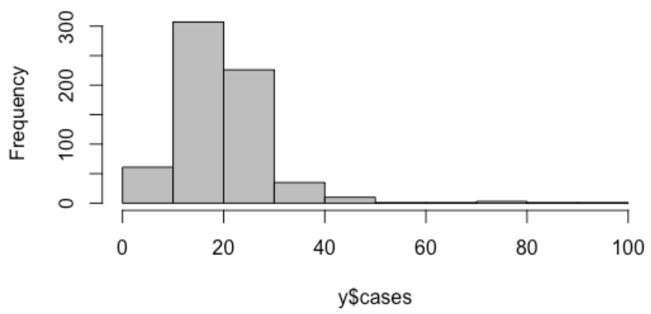
BUT

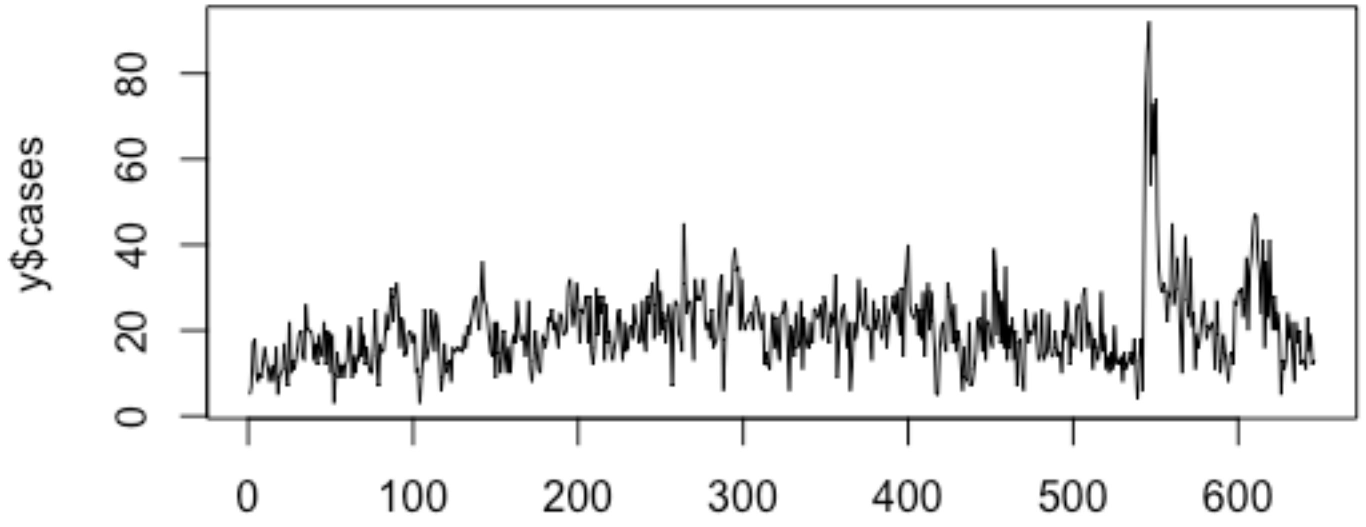
- These approaches don’t incorporate time series aspect of our data
- Other packages: tscount, etc
- And can be included in our function fit_stan()

Example: time series of ecoli

- y = tscount::ecoli

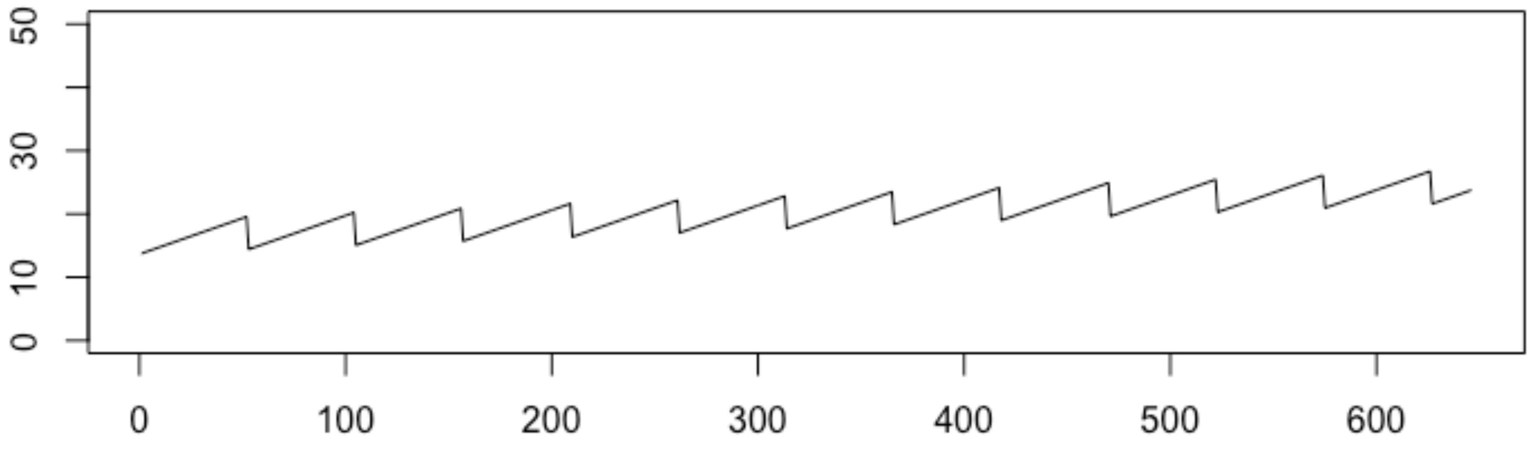
```
> head(y)
  year week cases
1 2001    1     5
2 2001    2     7
3 2001    3    17
4 2001    4    18
5 2001    5    10
6 2001    6     8
```

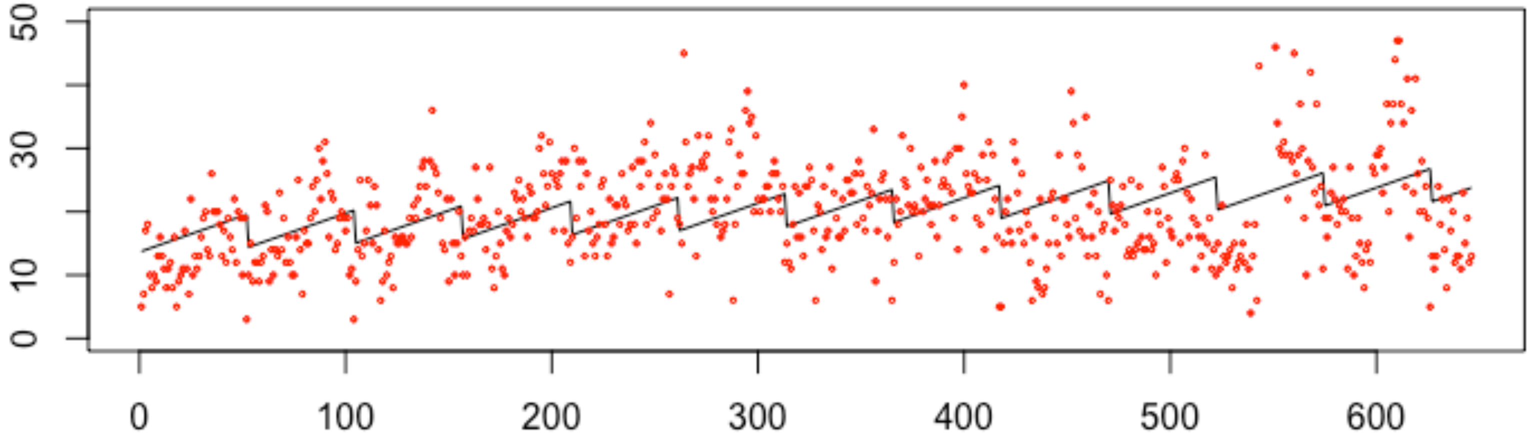




glmmTMB

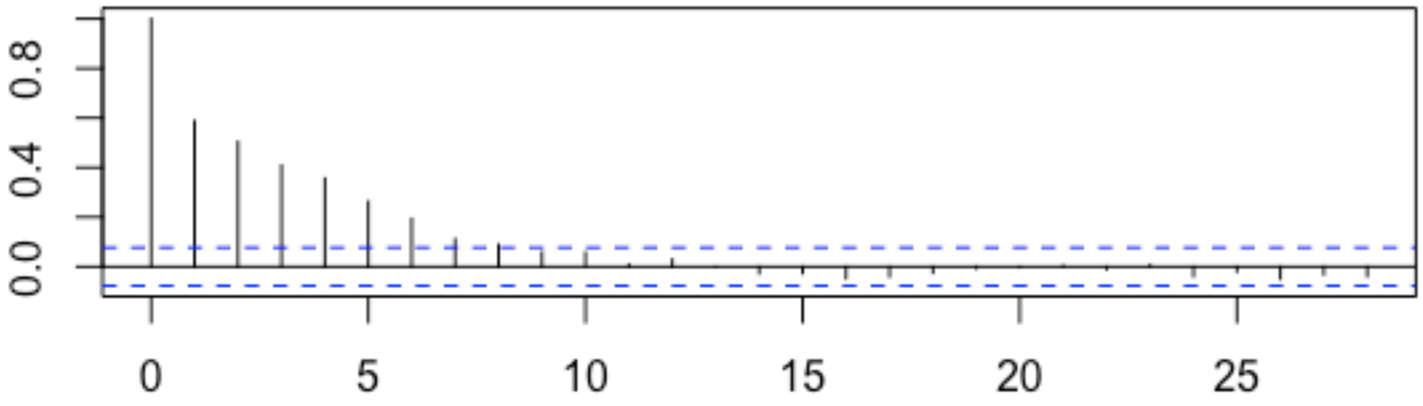
- glmmTMB(cases ~ week + year, data=y, family=“poisson”)





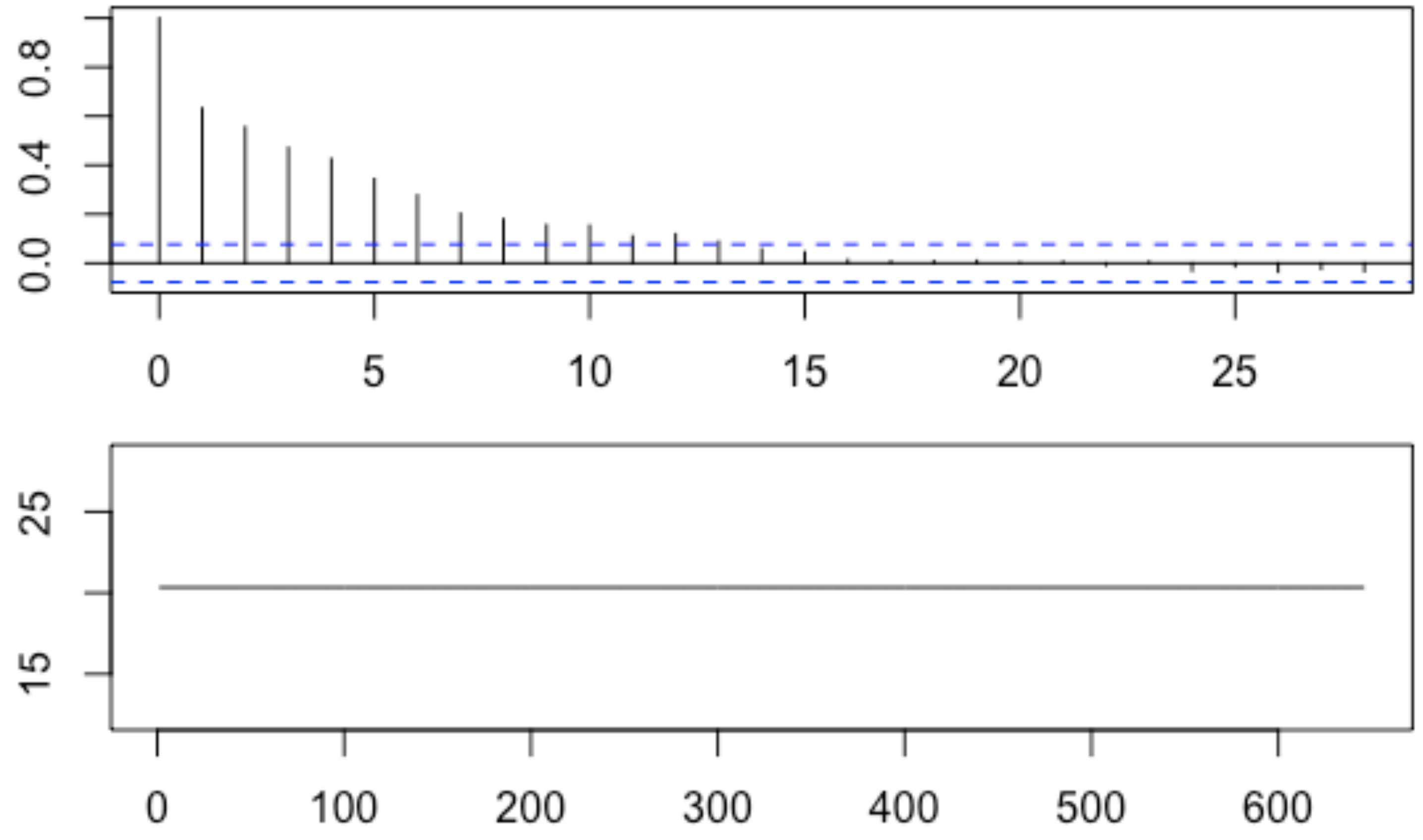
But residuals show problem

- Not independent, ACF shows they’re very correlated



tscount()

```
mod = tsglm(y, link="log", distr="poisson")
```



Added complexity

- Observations at time t can be made a function of predictions or observations in previous time steps
- model = list(past_obs = NULL, past_mean = NULL, …)
- Size of moving window also flexible
- Covariates can be included: time or external predictors

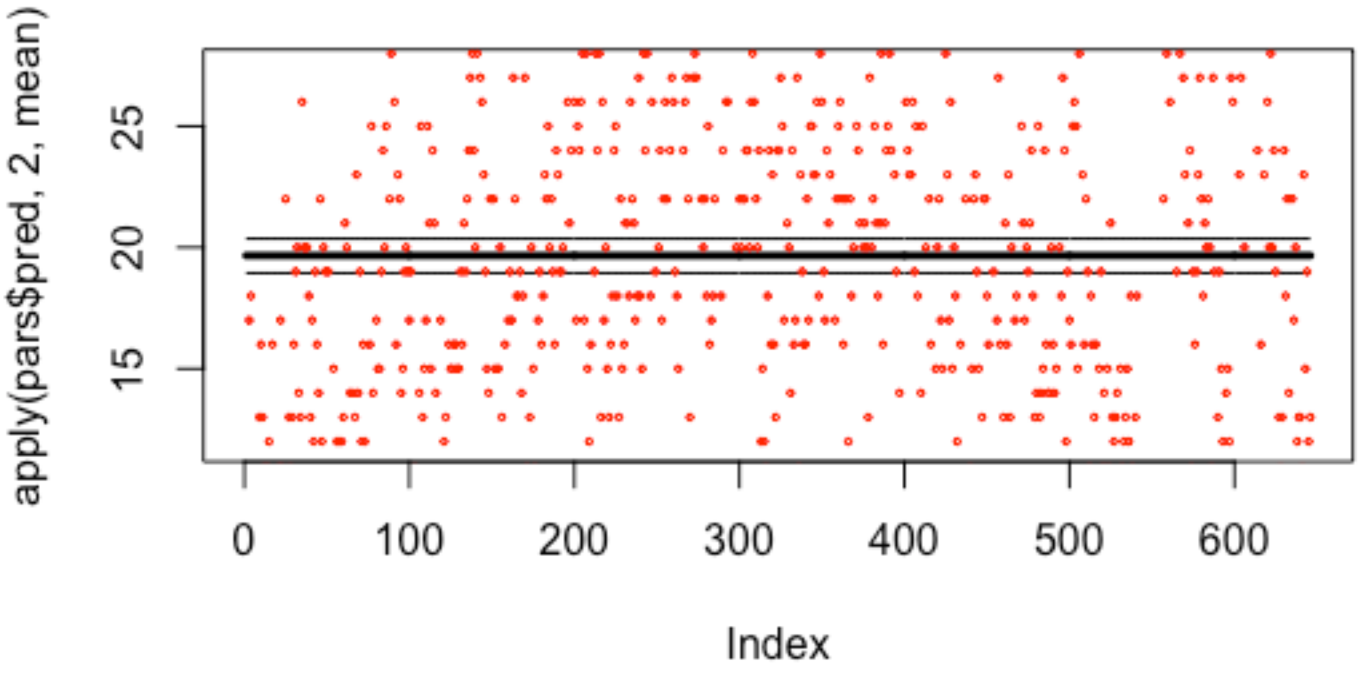
Implementation in stan

- fit_stan has a 'family' argument which can be specifed as
- “gaussian”
- “binomial”
- “poisson”
- “gamma”
- “lognormal”
- “negative-binomial”
- Etc

- Only for the following models: Regression, DLMs, 'MARSS’ models

Poisson regression

```
mod = fit_stan(y, x = model.matrix(lm(y~1)),
model="regression", family="poisson")
```



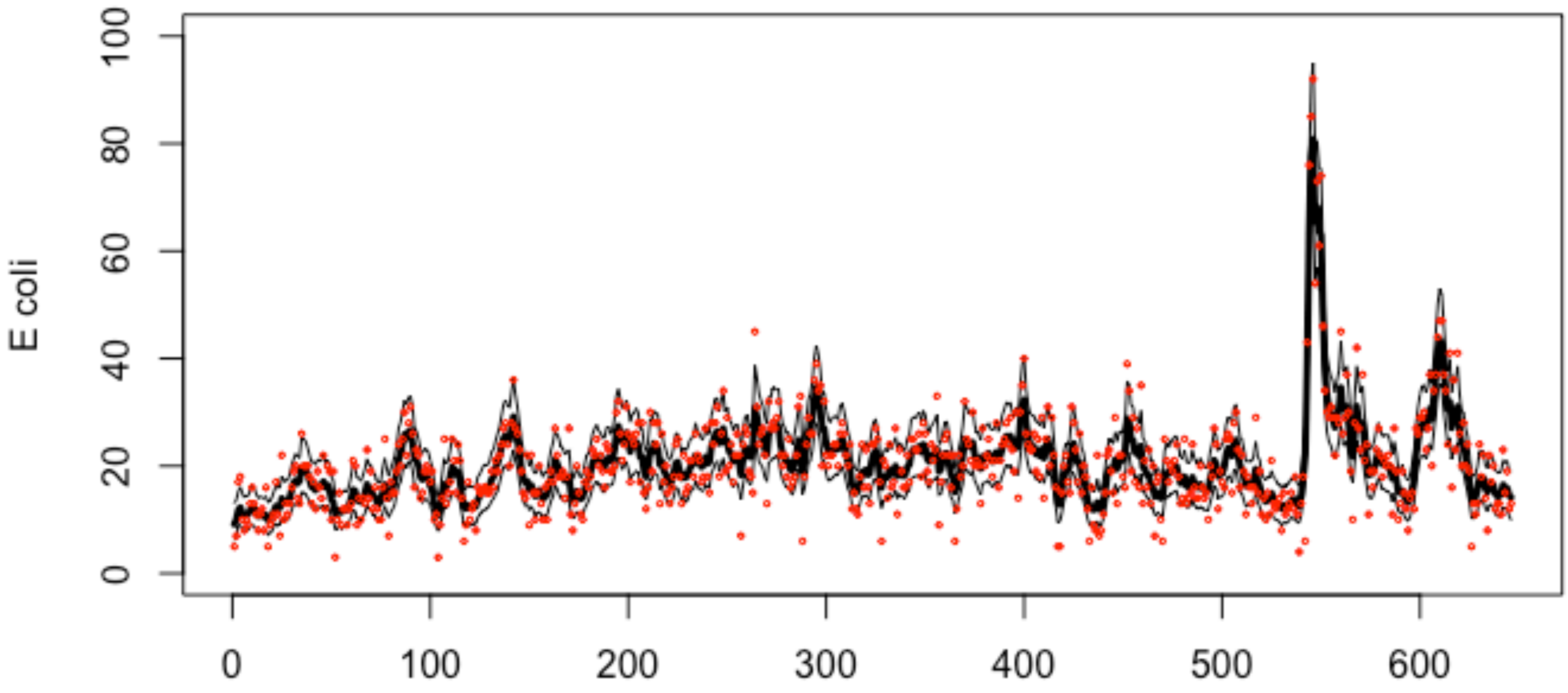
Implementation of DLM

We'll fit model with time-varying level (mean)

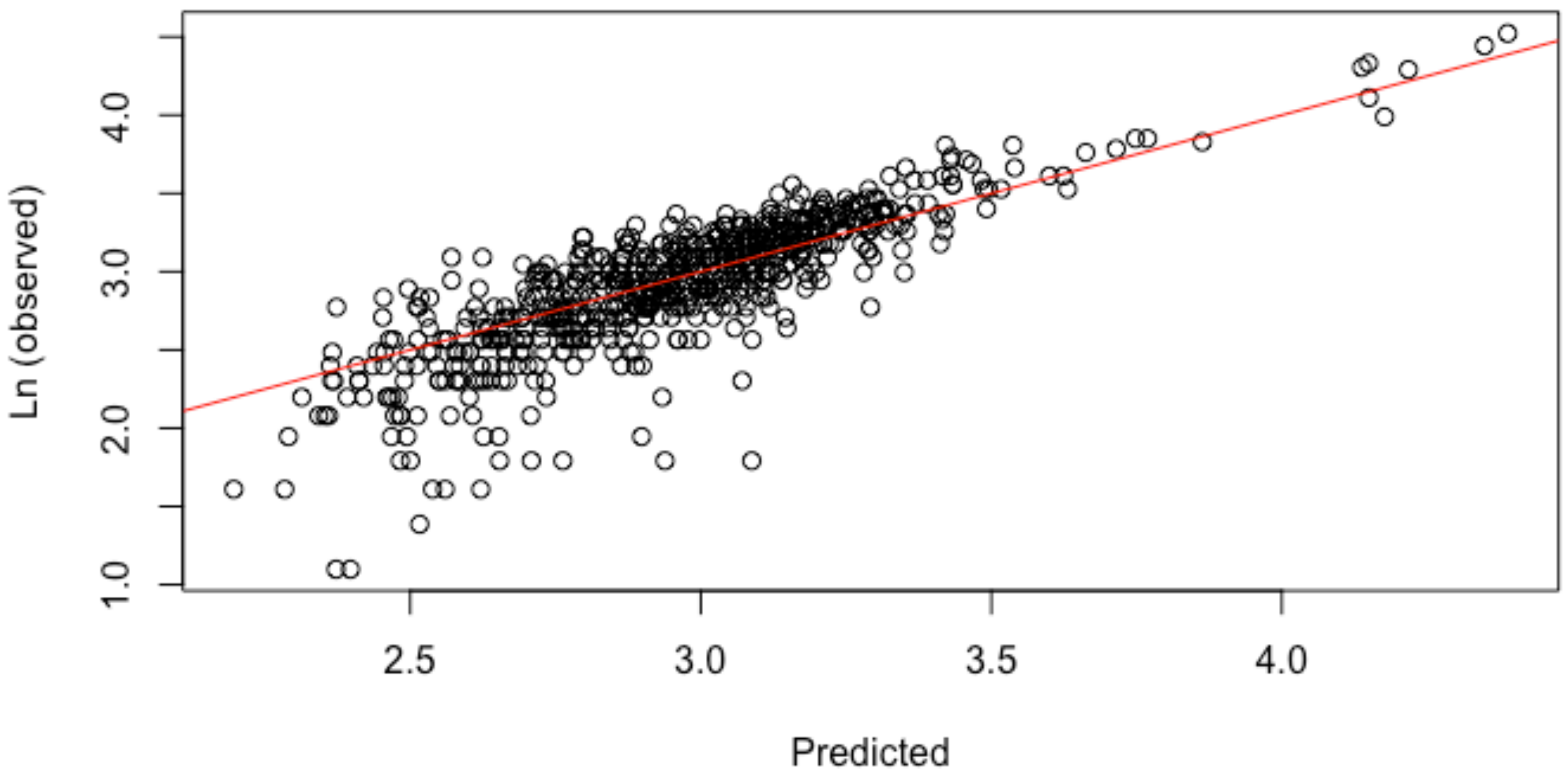
- No covariates included

```
mod = fit_stan(y, model="dlm-intercept",
family="poisson")
```

Now capturing data much better!

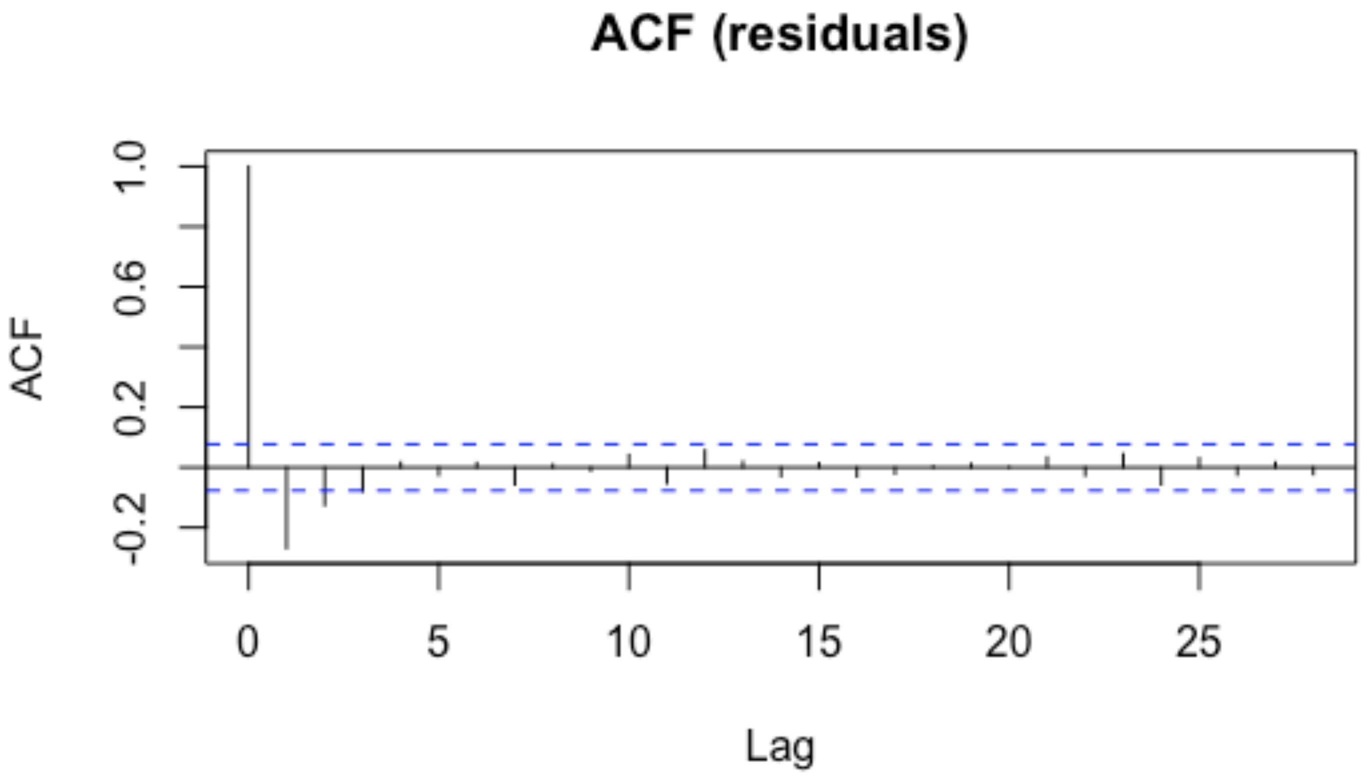


Predicted vs observed



Residuals look much better

- Still negative acf ~ lags 1-2


ACF (residuals)
ACF
Lag