Bayesian estimation for time series models

Feb 14 2017

Eric Ward

Re-cap following models using Bayesian code

- Regression
- ARMA models
- State Space Models
- Dynamic Factor Analysis
- Dynamic Linear Models
- MARSS models (multivariate time series models)

Why Bayesian?

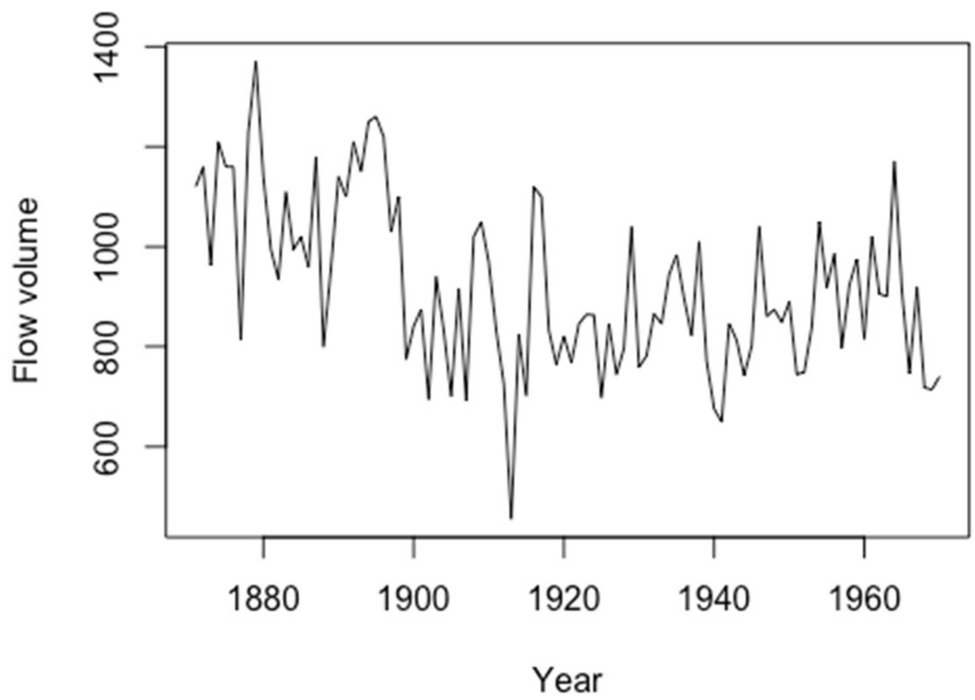
- Complex hierarchical models
- Inference: what's the probability that the data are less than some threshold?
- No bootstrapping!
  - We get credible intervals for parameters and states simultaneously

Getting started

```
library(rstan)
library(devtools)
devtools::install_github("eric-ward/safs-timeseries/statss")
library(statss)
```

Using STAN objects

- Flow volume from Nile River



Fitting linear regression

- We’ll use wrapper function '**fit_stan**’

```
x = model.matrix(lm(Nile~1))

lm_intercept = fit_stan(y = as.numeric(Nile), x =
rep(1, length(Nile)), model_name = "regression")
```

Output of fitted object

```
> lm_intercept
Inference for Stan model: regression.
3 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=1500.

           mean se_mean    sd   2.5%    25%    50%    75%   97.5% n_eff Rhat
beta[1]    0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
sigma    936.34    1.79 66.16 819.09 891.53 929.75 978.62 1074.10  1365    1
pred[1]    0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[2]    0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[3]    0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[4]    0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[5]    0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[6]    0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[7]    0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[8]    0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[9]    0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[10]   0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[11]   0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
pred[12]   0.56    0.05  1.97  -3.25  -0.78   0.51   1.87    4.53  1321    1
```

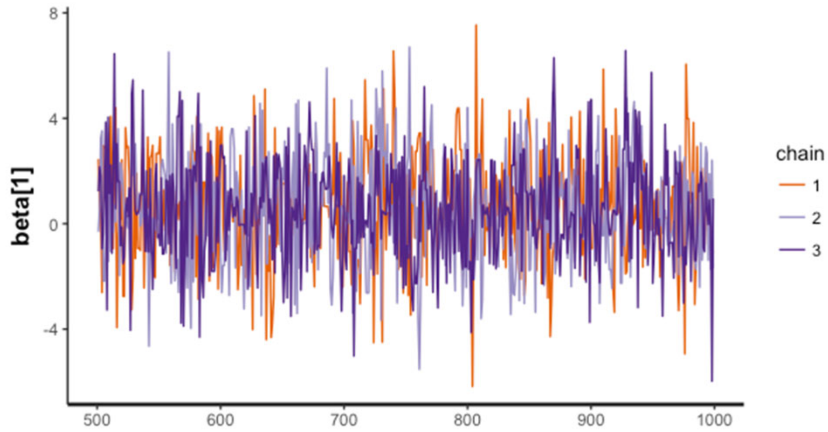
Output from model

- Can be plotted using base graphics in R

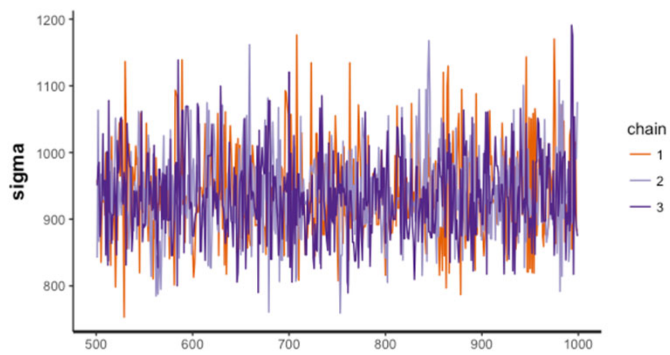
```
pars = extract(lm_intercept)
hist(pars$beta, 40, col="grey", xlab="Intercept", main="")
quantile(pars$beta, c(0.025,0.5,0.975))
```

Traceplots of parameters

- traceplot(lm_intercept, pars = "beta[1]")

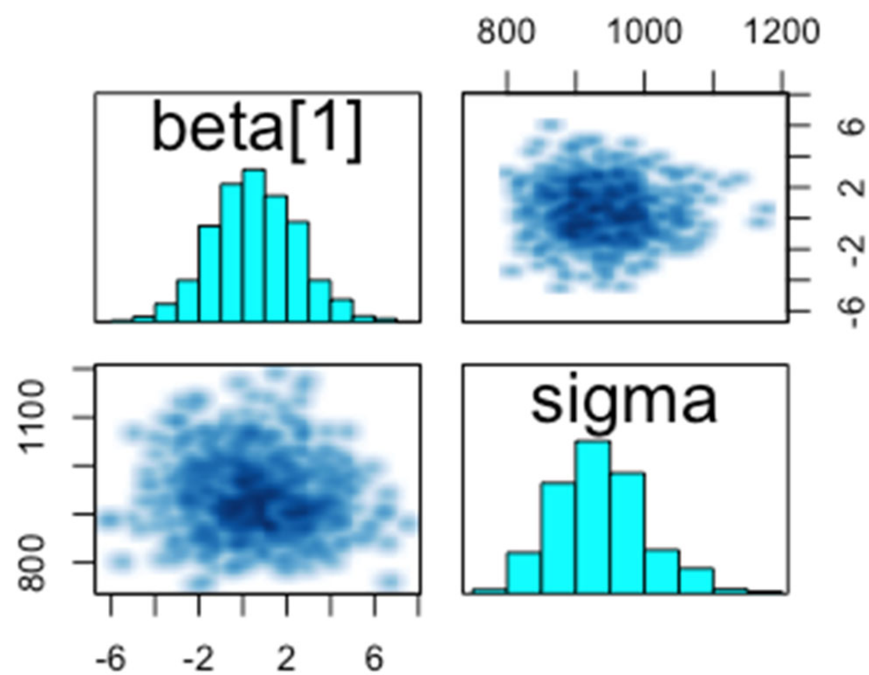


- traceplot(lm_intercept, pars = "sigma")



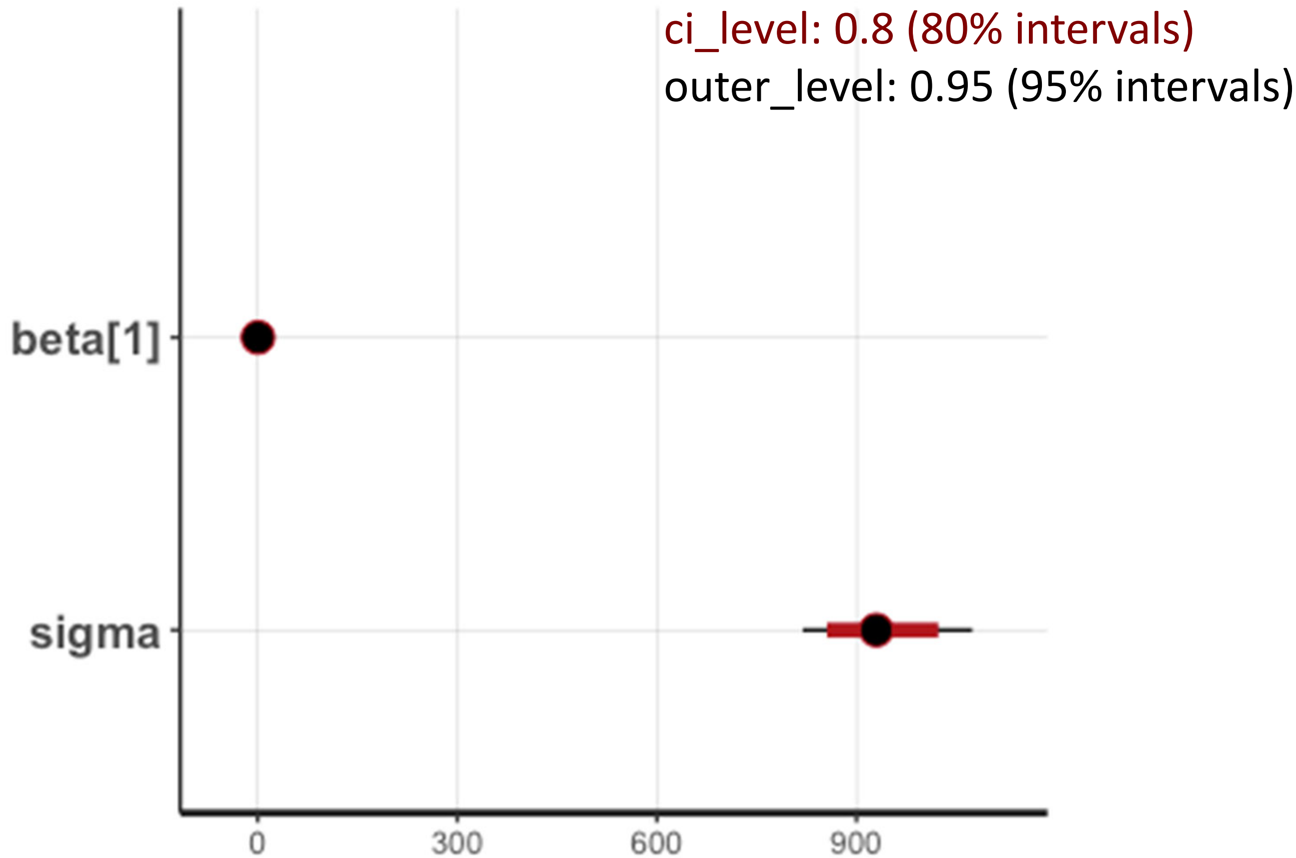
Pairs plots between parameters

- pairs(lm_intercept, pars=c("beta[1]","sigma"))



Plots with credible intervals

- plot(lm_intercept, pars=c("beta[1]","sigma"))



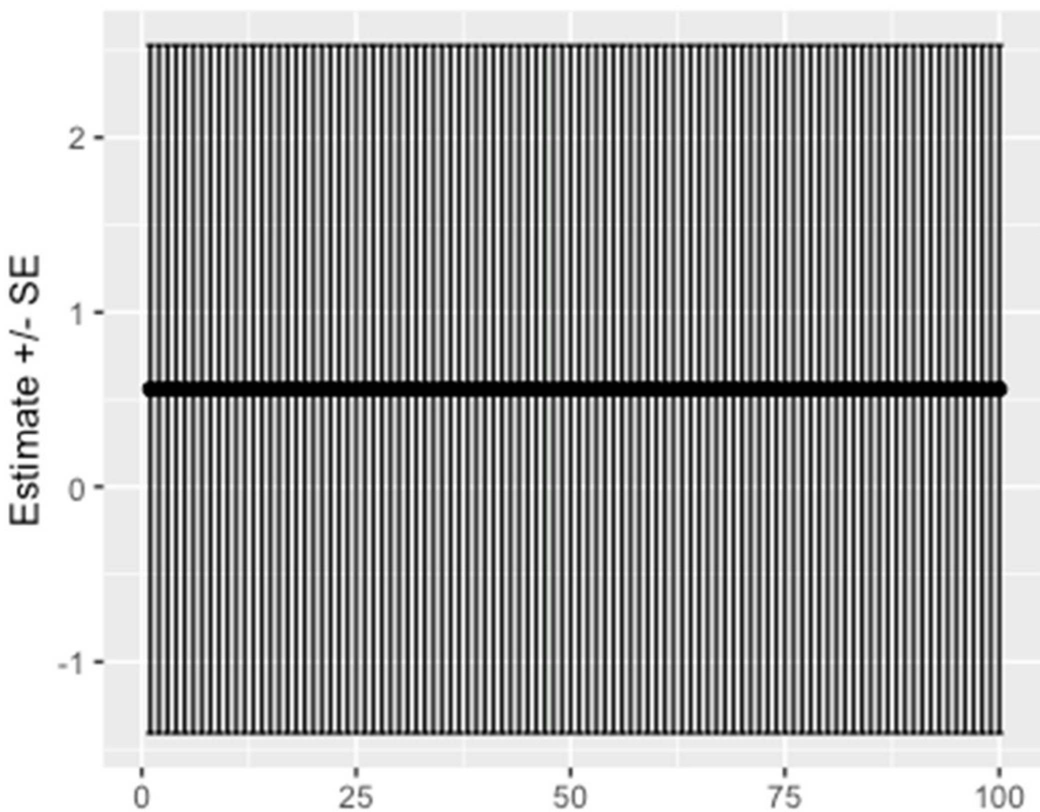
Getting tidy summaries

coef = broom::tidy(lm_intercept)

```
> broom::tidy(lm_intercept)
          term    estimate std.error
1      beta[1]   0.5602528  1.966926
2        sigma 936.3438793 66.160572
3      pred[1]   0.5602528  1.966926
4      pred[2]   0.5602528  1.966926
5      pred[3]   0.5602528  1.966926
6      pred[4]   0.5602528  1.966926
7      pred[5]   0.5602528  1.966926
8      pred[6]   0.5602528  1.966926
9      pred[7]   0.5602528  1.966926
10     pred[8]   0.5602528  1.966926
11     pred[9]   0.5602528  1.966926
12    pred[10]   0.5602528  1.966926
13    pred[11]   0.5602528  1.966926
14    pred[12]   0.5602528  1.966926
15    pred[13]   0.5602528  1.966926
16    pred[14]   0.5602528  1.966926
17    pred[15]   0.5602528  1.966926
18    pred[16]   0.5602528  1.966926
19    pred[17]   0.5602528  1.966926
```

Useful in ggplot, for example

- ggplot(coef[grep("pred",coef$term),], aes(x = 1:100,y=estimate)) + geom_point() + ylab("Estimate +/- SE")+ geom_errorbar(aes(ymin=estimate-std.error, ymax=estimate+std.error)) + xlab("")



Not interesting – all values are the same!

Preserving MCMC chains

- Each chain is independent
- Defaults to merging samples from all chains together

```
extract(object, pars, permuted = TRUE)
```

- But summaries can be generated for each combination of parameters-chains by setting

```
extract(object, pars, permuted = FALSE)
```

Random walk

- Formula: $E[Y_t] = Y_{t-1} + e_{t-1}$

- We'll fit model to temperature data

```
data(airquality)
Temp = airquality$Temp # air temperature
rw = fit_stan(y = Temp, est_drift = FALSE,
model_name = "rw")
```

Model convergence?

```
> rw
Inference for Stan model: rw.
3 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=1500.

          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
sigma     5.77    0.01 0.33  5.19  5.53  5.76  5.99  6.44  1125    1
pred[1]   0.00    0.00 0.00  0.00  0.00  0.00  0.00  0.00  1500  NaN
pred[2]  67.00    0.00 0.00 67.00 67.00 67.00 67.00 67.00  1500  NaN
pred[3]  72.00    0.00 0.00 72.00 72.00 72.00 72.00 72.00  1500  NaN
pred[4]  74.00    0.00 0.00 74.00 74.00 74.00 74.00 74.00  1500  NaN
pred[5]  62.00    0.00 0.00 62.00 62.00 62.00 62.00 62.00  1500  NaN
pred[6]  56.00    0.00 0.00 56.00 56.00 56.00 56.00 56.00  1500  NaN
pred[7]  66.00    0.00 0.00 66.00 66.00 66.00 66.00 66.00  1500  NaN
pred[8]  65.00    0.00 0.00 65.00 65.00 65.00 65.00 65.00  1500  NaN
pred[9]  59.00    0.00 0.00 59.00 59.00 59.00 59.00 59.00  1500  NaN
pred[10] 61.00    0.00 0.00 61.00 61.00 61.00 61.00 61.00  1500  NaN
pred[11] 69.00    0.00 0.00 69.00 69.00 69.00 69.00 69.00  1500  NaN
```

Univariate state space models

- State equation

$$x_t = \phi * x_{t-1} + e_{t-1}; e_{t-1} \sim Normal(0, q)$$

- Observation equation

$$Y_t \sim Normal(x_t, r)$$

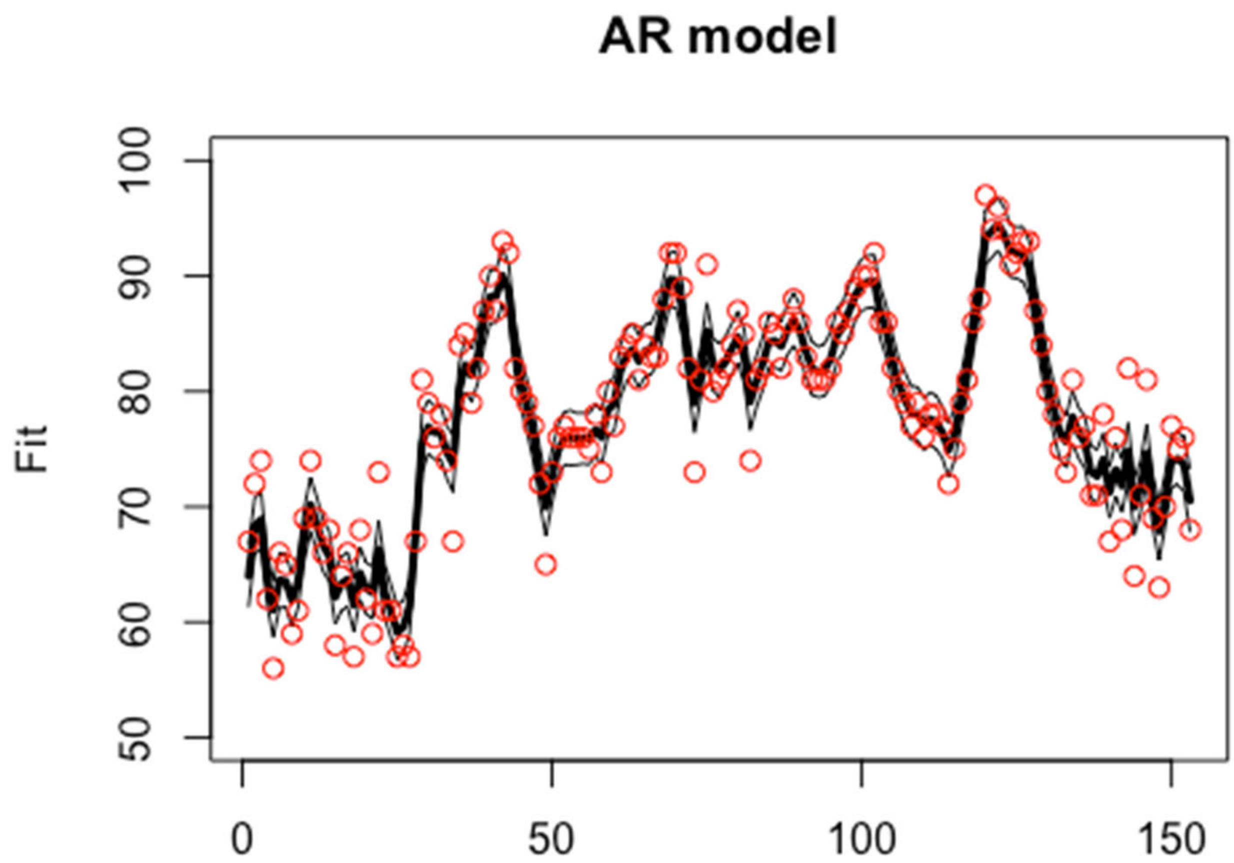
- Let's compare models with and without the ar parameter phi in the process model

Using our stan_fit function

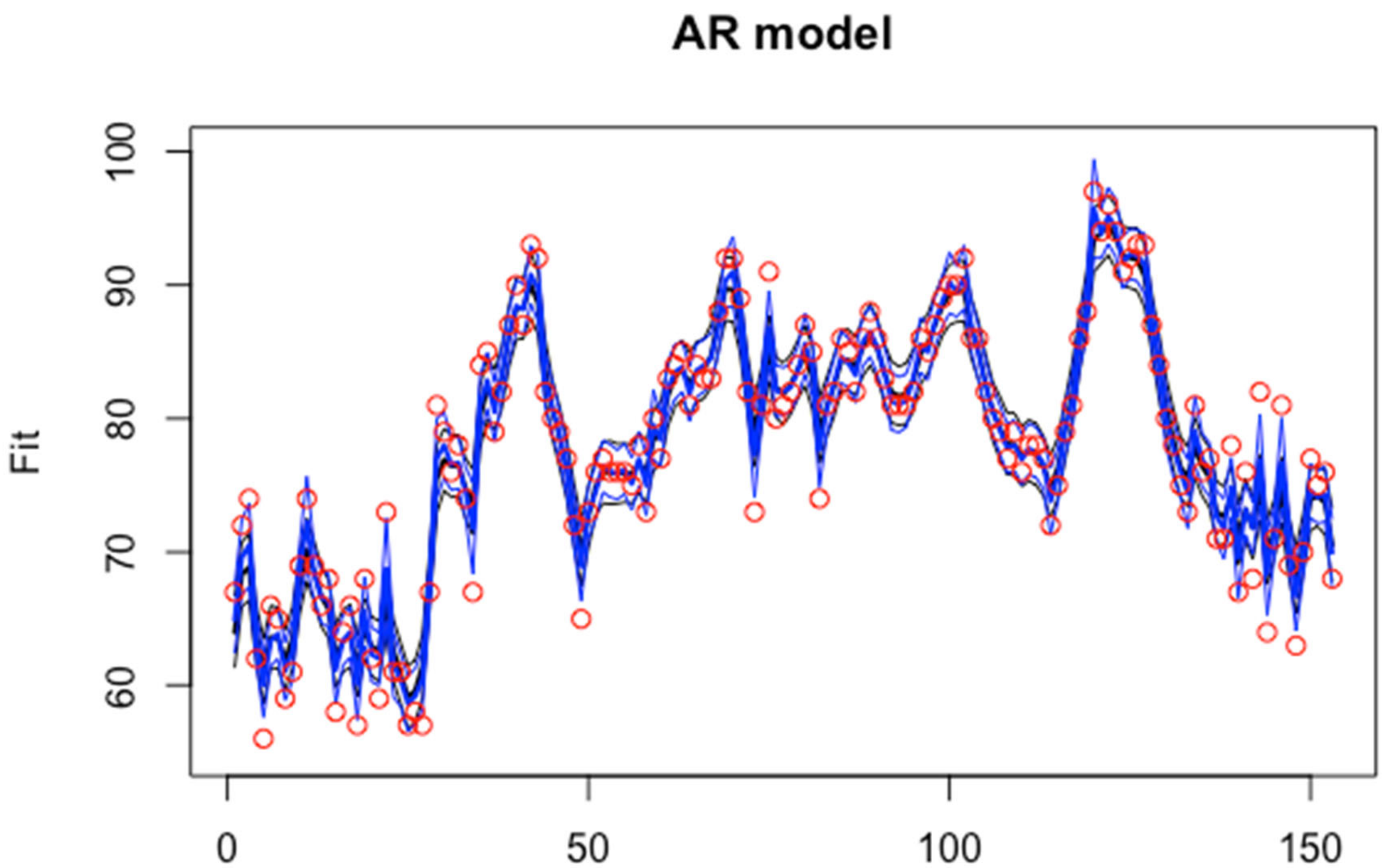
```
ss_ar = fit_stan(y = Temp, est_drift=FALSE,
model_name = "ss_ar")

ss_rw = fit_stan(y = Temp, est_drift=FALSE,
model_name = "ss_rw")
```

Estimates from AR SS model

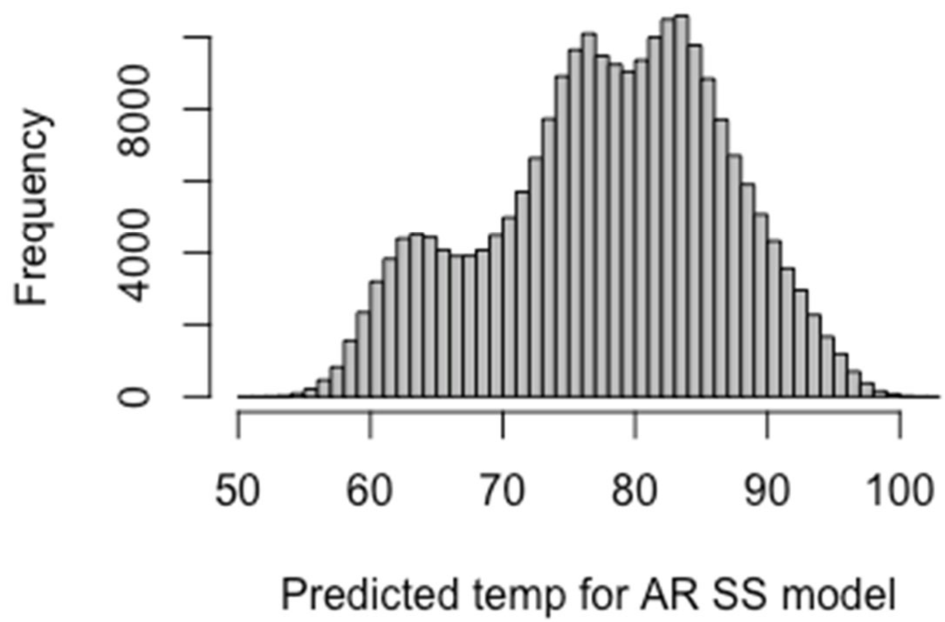


Estimates from both models



Posterior probability

- What's the probability that the temperature exceeds some threshold? 100 degrees?



Probability of > 100 degrees

```
pars = extract(ss_ar)
length(which(pars$pred > 100))
```

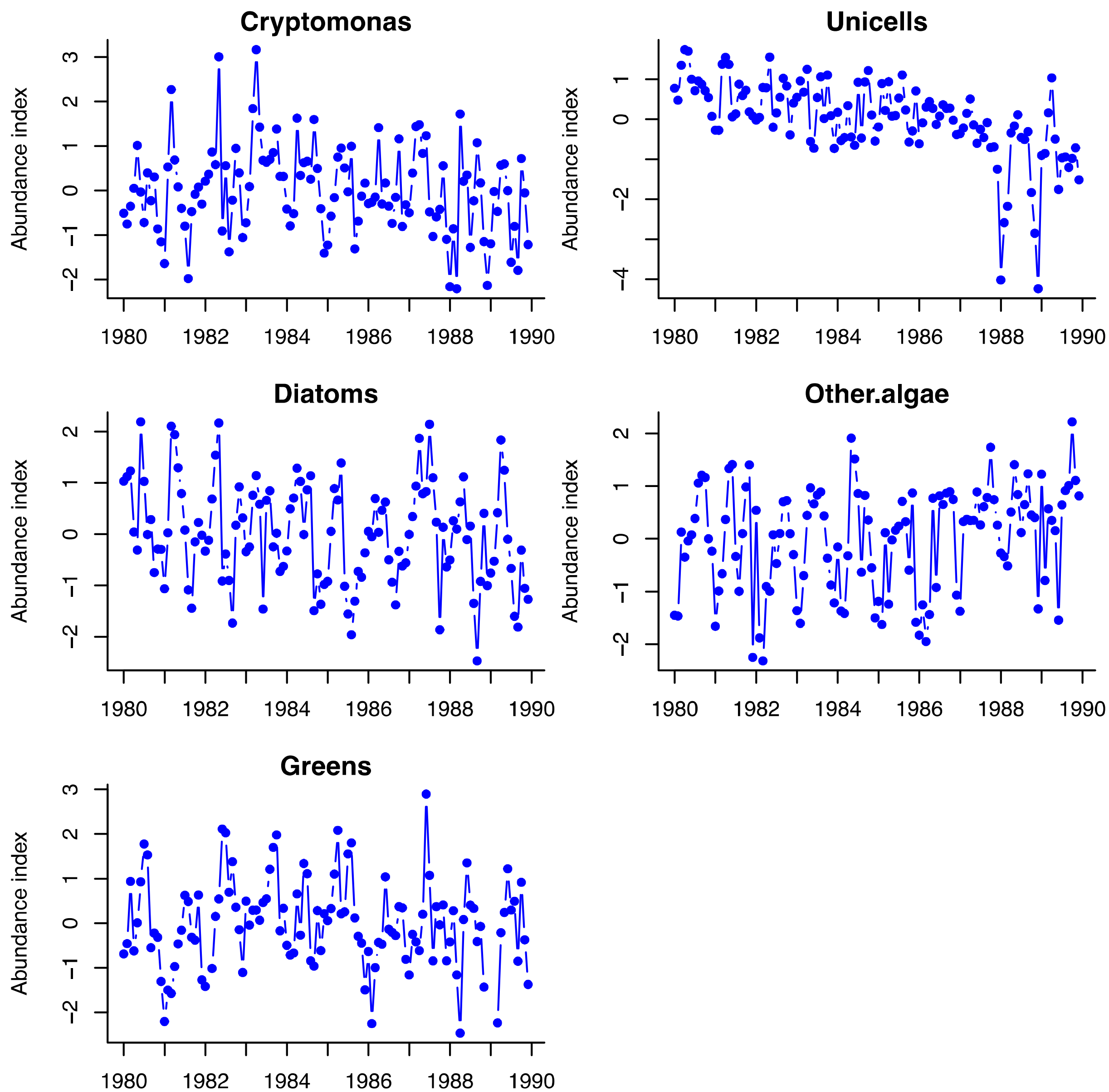
Low probability: ~ 20 / 229500

Dynamic Factor Analysis

- Lake WA plankton example used in manual

```
# load the data (there are 3 datasets contained here)
data(lakeWAplankton)
# we want lakeWAplanktonTrans, which has been transformed
# so the 0s are replaced with NAs and the data z-scored
dat = lakeWAplanktonTrans
# use only the 10 years from 1980-1989
plankdat = dat[dat[,"Year"]>=1980 & dat[,"Year"]<1990,]
# create vector of phytoplankton group names
phytoplankton = c("Cryptomonas", "Diatoms", "Greens",
                  "Unicells", "Other.algae")
# get only the phytoplankton
dat.spp.1980 = plankdat[,phytoplankton]
```

Plankton data



Running the model

- 3 trend model to start

```
mod_3 = fit_dfa(y = t(dat.spp.1980),
num_trends=3)
```

Trends need to be rotated (like MARSS)

- Again we'll use varimax rotation
- Use function we've written, **rotate_trends**

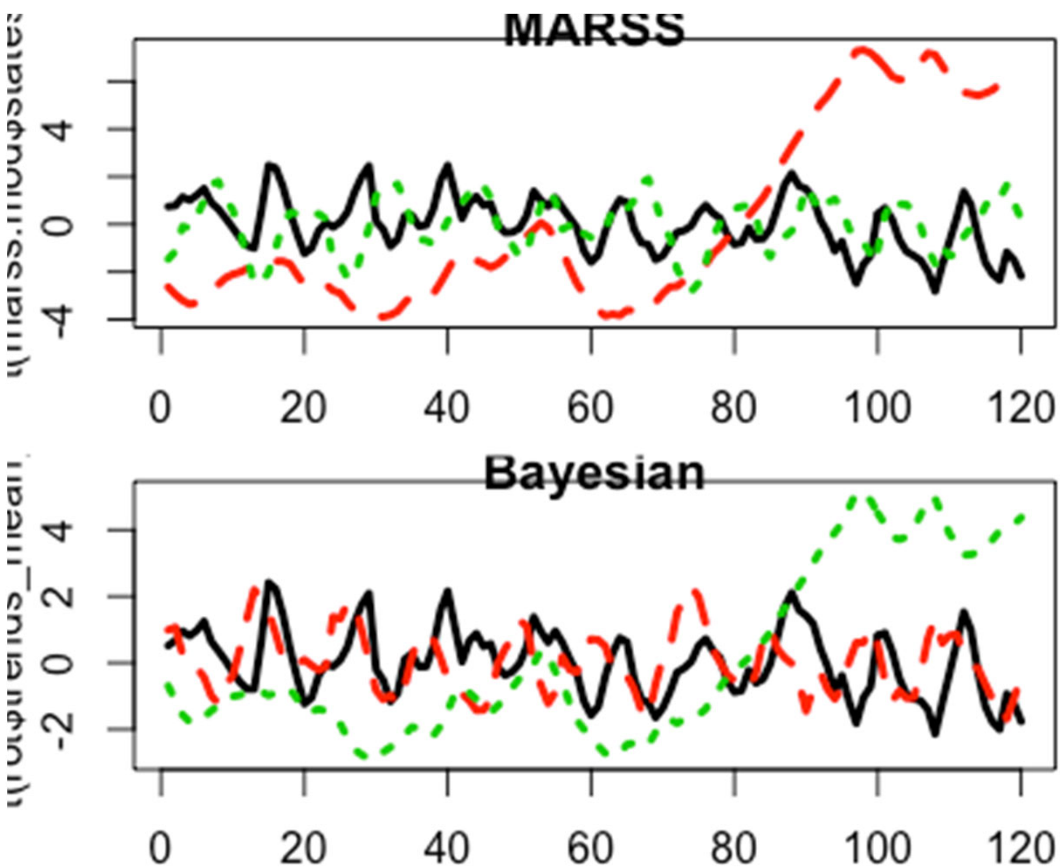
```
rot = rotate_trends(mod_3)
```

names(rot)

- Z_rot, rotated Z matrix for each MCMC draw
- trends, rotated trends for each MCMC draw
- **Z_rot_mean, mean Z across draws**
- **trends_mean, mean trends across draws**
- trends_lower, lower 2.5% on trends
- trends_upper, upper 97.5% on trends

Predicted values from Bayesian DFA

- Same results as MARSS (**trends_mean**)



Other variance structures

```
mod_3 = fit_dfa(y = t(dat.spp.1980),
num_trends=3)
```

- By default, this is modeling 'diagonal and equal'

- Diagonal and unequal or shared variances can also be specified using 'varIndx' argument

```
mod_3 = fit_dfa(y = t(dat.spp.1980),
num_trends=3, varIndx = rep(1,5))
```

Model selection: how to select best number of trends?

- First run multiple models with varying trends

```
mod_1 = fit_dfa(y = t(dat.spp.1980), num_trends=1)
mod_2 = fit_dfa(y = t(dat.spp.1980), num_trends=2)
mod_3 = fit_dfa(y = t(dat.spp.1980), num_trends=3)
mod_4 = fit_dfa(y = t(dat.spp.1980), num_trends=4)
mod_5 = fit_dfa(y = t(dat.spp.1980), num_trends=5)
```

- 3 minutes to fit all models (4000 iterations) – probably could be at least cut in half

Leave One Out Information Criterion (LOOIC)

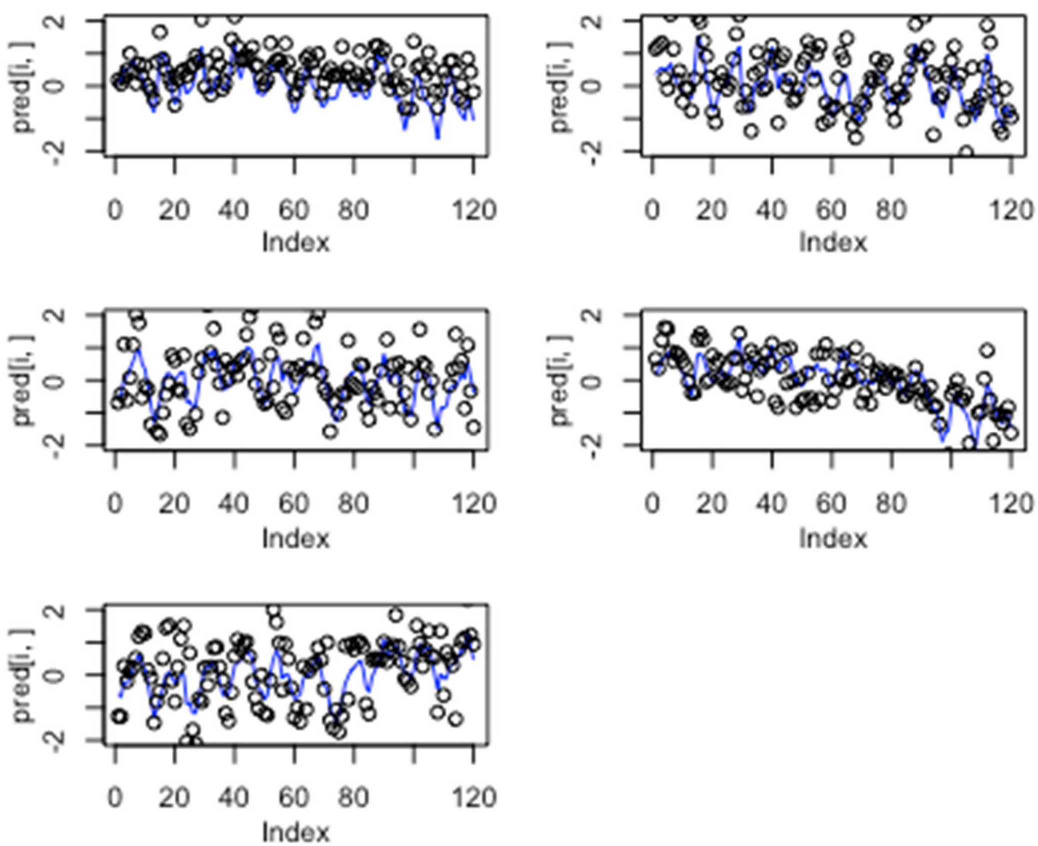
- Like AIC, lower is better
- Simple function in **library(loo)**

loo(extract_log_lik(mod_1))$looic

Trends	LOOIC
1	1598.889
2	1535.885
3	1469.439
4	1455.7
5	1449.304

Predicted values

- Also like MARSS, use $pred parameter



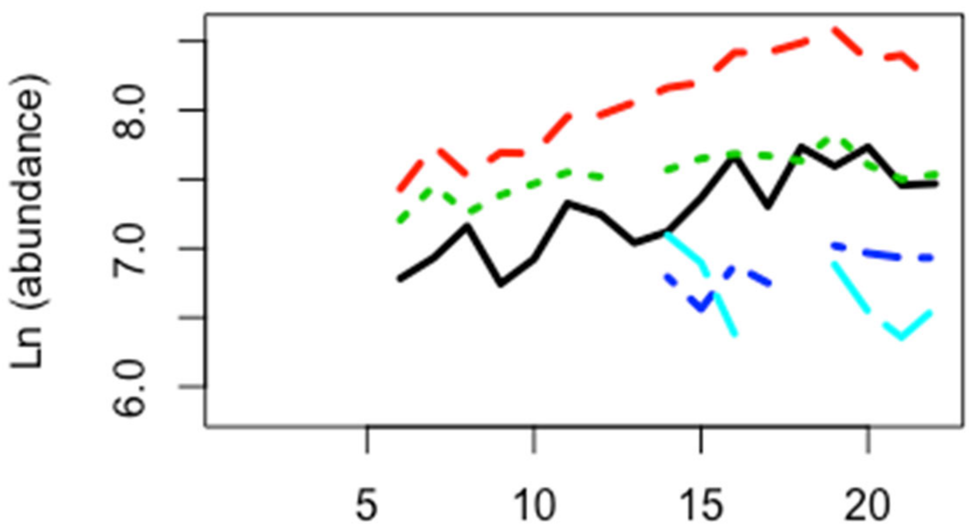
Uncertainty intervals on states

- We often just present the trend estimates for DFA – but not uncertainty

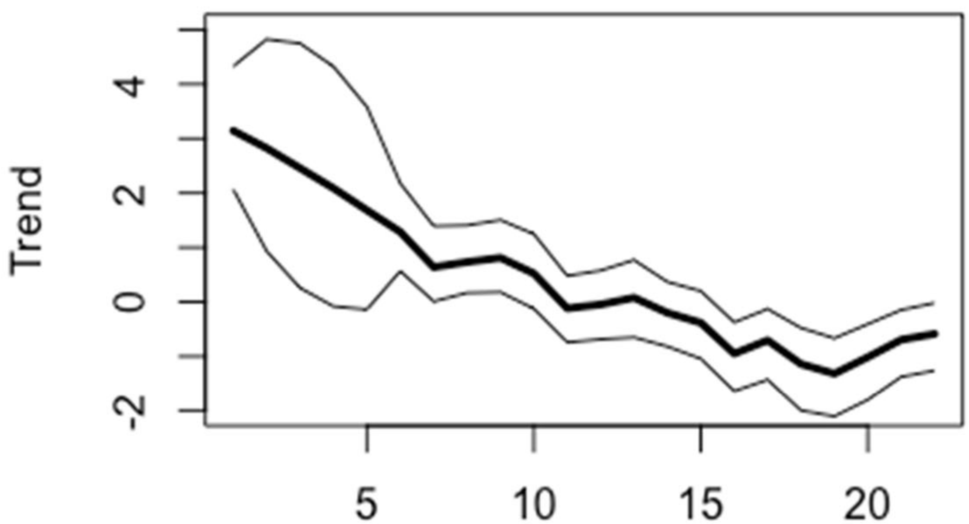
- Let's look at effect of missing data on DFA for the harbor seal dataset

```
data("harborSealWA")
```

- Assume single trend for the population

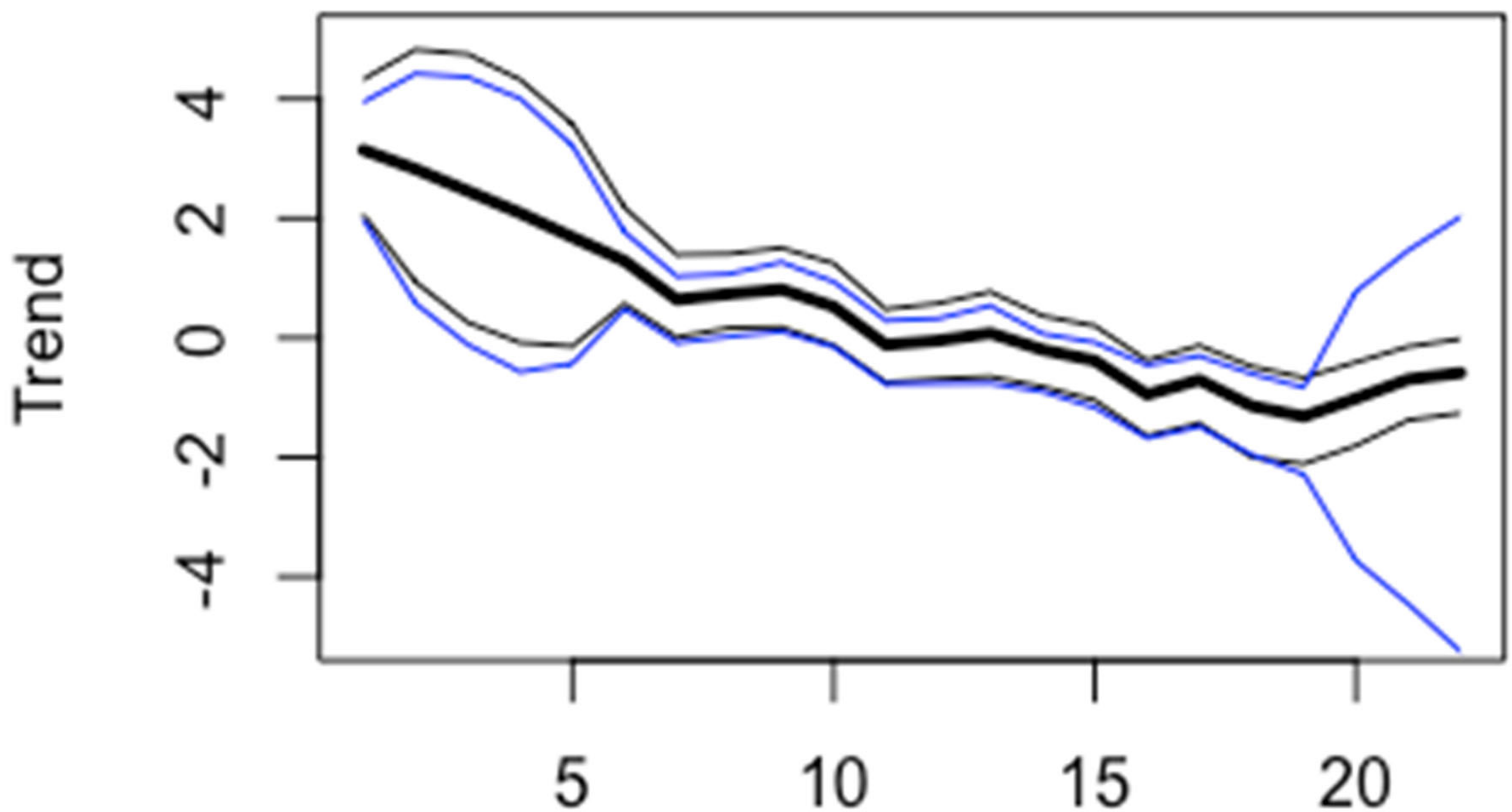


- pars = extract(fit_dfa(y = t(harborSealWA[,-1]), num_trends = 1))



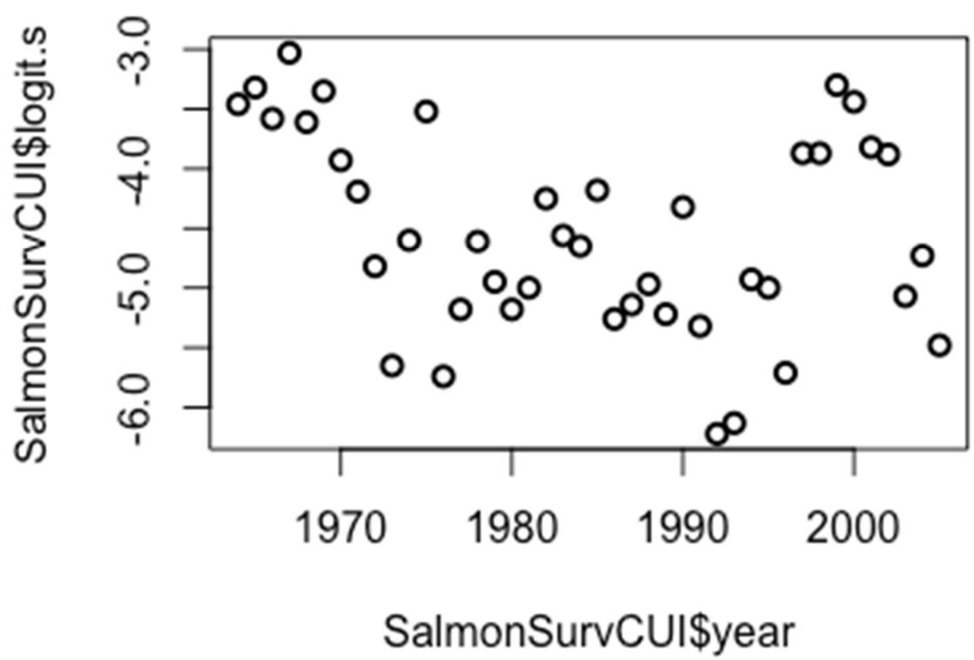

```
> harborSealWA
      Year      SJF      SJI    EBays     PSnd       HC
 [1,] 1978 6.033086 6.747587 6.626718 5.820083 6.595781
 [2,] 1979       NA       NA       NA       NA       NA
 [3,] 1980       NA       NA       NA       NA       NA
 [4,] 1981       NA       NA       NA       NA       NA
 [5,] 1982       NA       NA       NA       NA       NA
 [6,] 1983 6.783325 7.431300 7.205635       NA       NA
 [7,] 1984 6.932448 7.744137 7.454141       NA       NA
 [8,] 1985 7.160846 7.527794 7.255591 6.595781       NA
 [9,] 1986 6.744059 7.693026 7.385851       NA       NA
[10,] 1987 6.923629 7.686621 7.467942       NA       NA
[11,] 1988 7.325149 7.954021 7.550661       NA       NA
[12,] 1989 7.245655 7.966933 7.516977       NA       NA
[13,] 1990 7.040536 8.057377       NA       NA       NA
[14,] 1991 7.121252 8.163371 7.569928 6.792344 7.095064
[15,] 1992 7.365180 8.199739 7.650645 6.562444 6.896694
[16,] 1993 7.675082 8.417152 7.684784 6.879356 6.383507
[17,] 1994 7.305188 8.418256 7.670429 6.749931       NA
[18,] 1995 7.732369 8.487146 7.634337       NA       NA
[19,] 1996 7.594884 8.581107 7.832411 7.020191 6.882437
[20,] 1997 7.733684 8.361007 7.604894 6.966024 6.543912
[21,] 1998 7.458186 8.398635 7.501082 6.933423 6.357842
[22,] 1999 7.468513 8.185350 7.535297 6.932448 6.566672
```

What happens when we delete last 3 years of data?



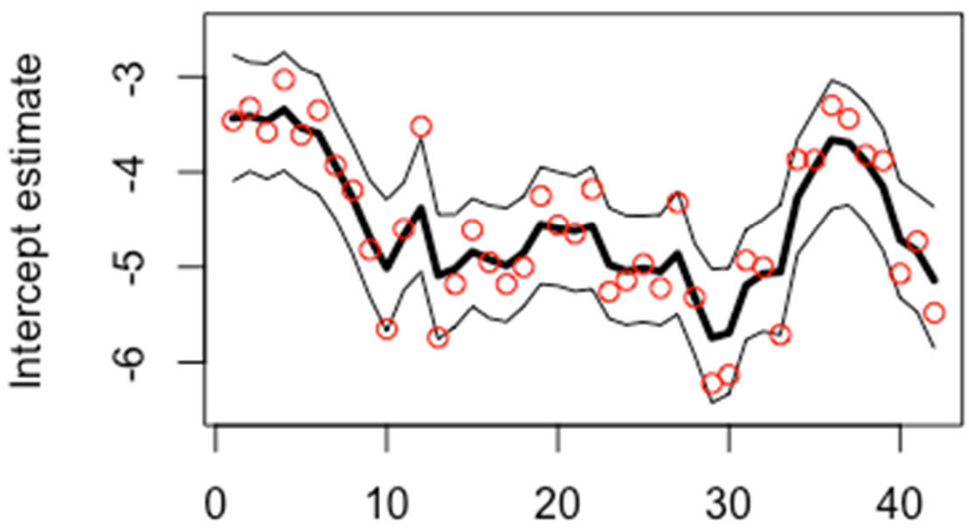
DLMs

- Mark's example of salmon survival
- Logit transformed data



Fitting a DLM with time varying intercept

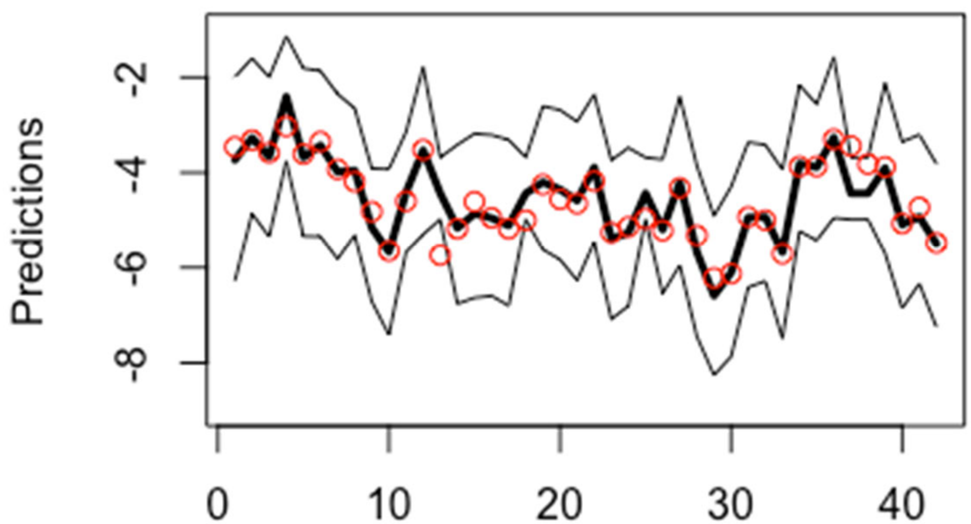
```
mod = fit_stan(y = SalmonSurvCUI$logit.s,
model_name="dlm-intercept")
```



Constant intercept, time – varying slope

```
mod = fit_stan(y = SalmonSurvCUI$logit.s, x =
SalmonSurvCUI$CUI.apr, model_name="dlm-
slope")
```

Wider CIs than
time varying intercept
model



Time varying intercept and slope

- Use model.matrix() to specify 'x'

- x = model.matrix(lm(SalmonSurvCUI$logit.s ~ SalmonSurvCUI$CUI.apr))

```
mod = fit_stan(y = SalmonSurvCUI$logit.s, x = model.matrix(lm(SalmonSurvCUI$logit.s ~ SalmonSurvCUI$CUI.apr)), model_name="dlm")
```

Summary

- Additional models available, e.g. 'marss'
- Very flexible
- Easy to add custom features on existing code