Univariate Bayesian time series models

FISH 550 – Applied Time Series Analysis

Eric Ward

2 May 2023

Overview of today's material

- Bayesian estimation
- Overview of Stan
- Manipulating and plotting Stan output
- Examples of time series models

Review of models we've used so far

Models

- Regression
- ARMA models
- State Space Models
- Dynamic Linear Models
- Dynamic Factor Analysis
- Multivariate time series models)

Why Bayesian?

- Complex hierarchical models
  - Non-linear models
  - Hierarchical or shared parameters
  - Non-normal data
  - Prior information
- Inference: what's the probability that the data are less than some threshold?
- No bootstrapping!
  - We get credible intervals for parameters and states simultaneously

Bayesian logic

- Conditional probability

$$P(\theta|\mathbf{y})P(\mathbf{y}) = P(\theta)P(\mathbf{y}|\theta)$$

$$P(\theta|\mathbf{y}) = \frac{P(\theta)P(\mathbf{y}|\theta)}{P(\mathbf{y})}$$

- $P(\mathbf{y})$ is a normalizing constant that we often don't have to worry about

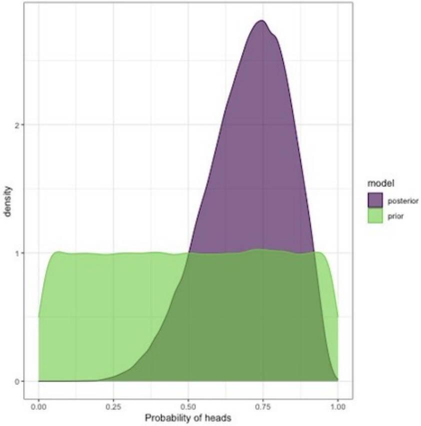
Bayesian logic

- Parameters are random, data are fixed
-
$$P(\theta|\mathbf{y}) = P(\theta)P(\mathbf{y}|\theta)$$
- $P(\theta|\mathbf{y})$ is the posterior
- $P(\mathbf{y}|\theta)$ is the likelihood
- $P(\theta)$ is the prior

Bayesian logic

Bayesian logic

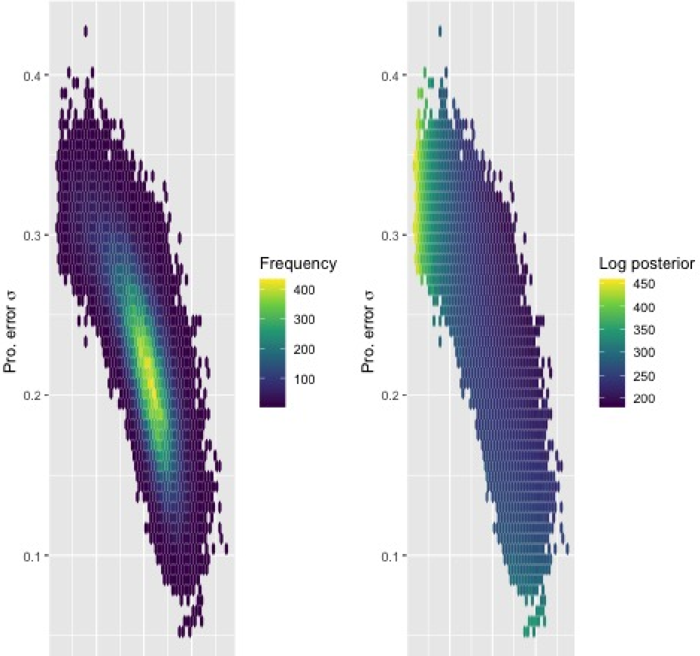
- Difference between posterior and prior represents how much we learn by collecting data
- Experiment {H, H, T, H, H, T, H, H}



Bayesian mechanics

- MLE seeks to find the combination of parameters that maximize the likelihood (e.g. find absolute best point)
- Bayesian estimation uses integration to find the combination of parameters that are best *on average*

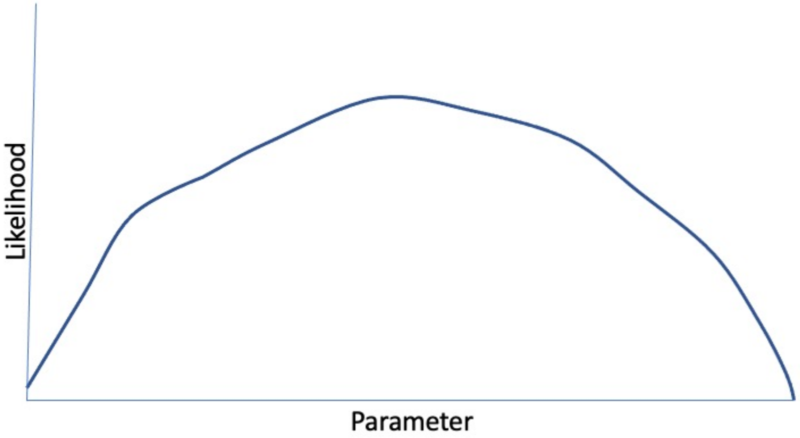
Bayesian mechanics in practice



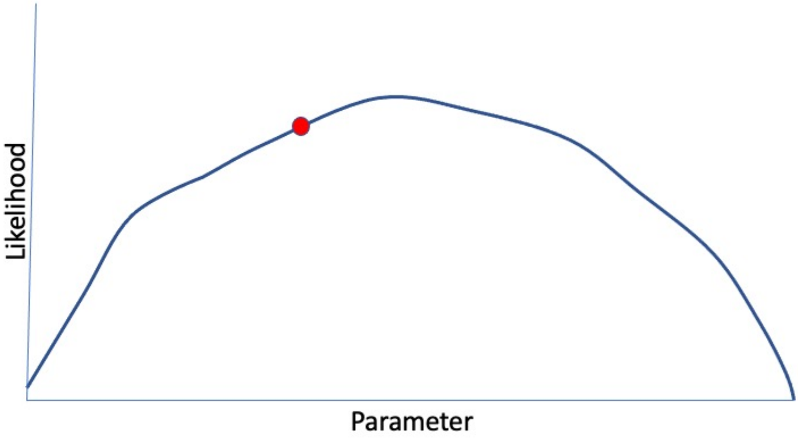
Estimation

- Goal of Bayesian estimation in drawing samples from the posterior $P(\theta|\mathbf{y})$
- For very simple models, we can write the analytical solution for the posterior
- But for 99% of the problems we work on, need to generate samples via simulation
- Markov chain Monte Carlo

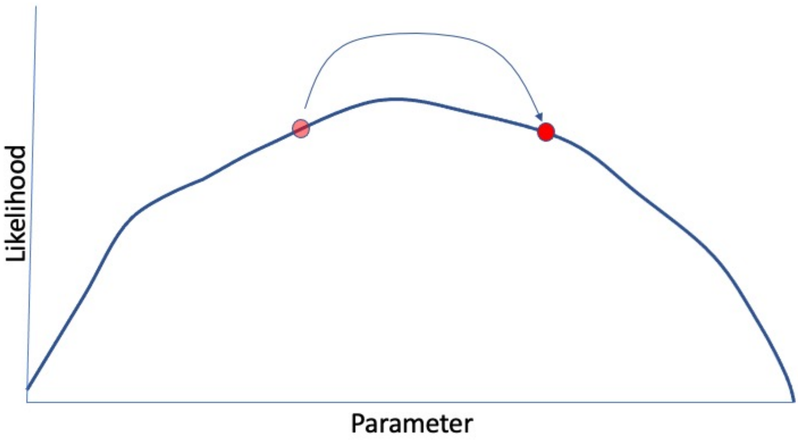
Estimation



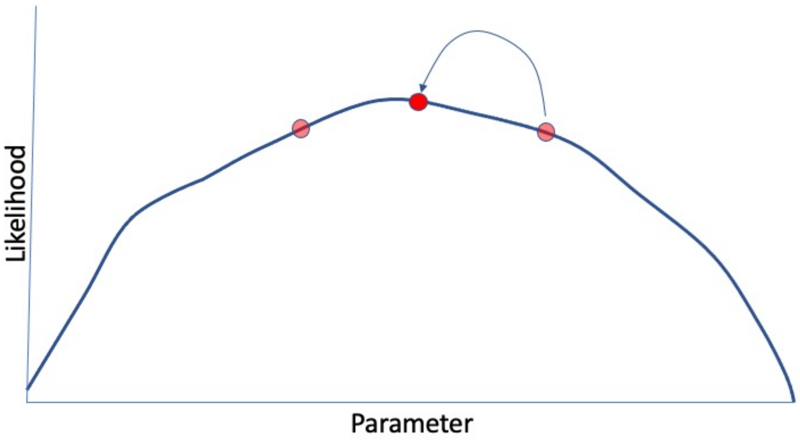
Estimation



Estimation

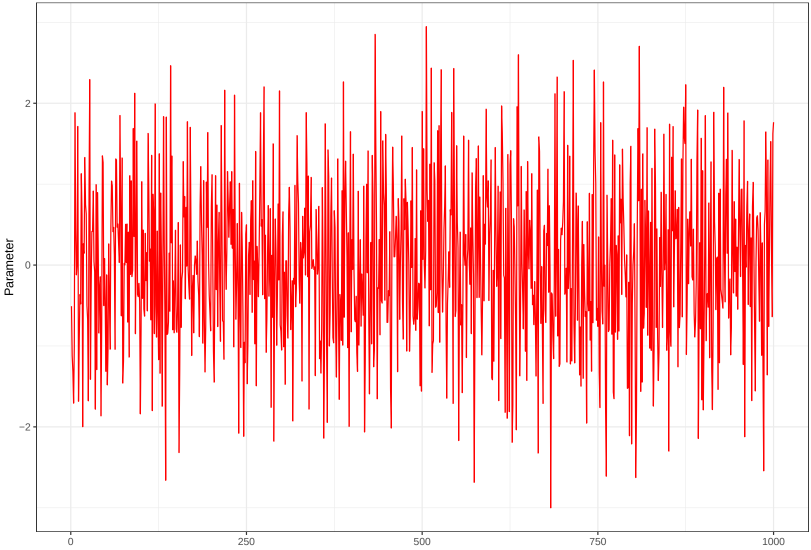


Estimation



Estimation

- Thousands of proposals later, we have a MCMC chain



Estimation: best practices

- Run 3-4 MCMC chains in parallel
- Discard first $\sim$ 10-50% of each MCMC chain as a 'burn-in'
- Optionally apply MCMC thinning to reduce autocorrelation

Lots of algorithms for sampling

- Metropolis, Metropolis-Hastings
- Sampling - Imporance - Resampling (SIR)
- No-U-Turn Sampler (NUTS)
- Monahan et al. 2016, Faster estimation of Bayesian models in ecology using Hamiltonian Monte Carlo

What is Stan?

- Powerful, cross-platform and cross-language (R, Julia, Matlab, etc) that allows users to write custom code that can be called directly from R
- Estimation can be fully or approximate Bayesian inference, or maximum a posteriori optimization (BFGS)
- Useful links:
  - Stan homepage
  - Stan manual
  - rstan

Options for using Stan in this class

- Write your own code (based on examples in the manual, etc)
- Use an existing package
- Use our bundled code to get started with simple models (we'll start here)

Existing packages: rstanarm and brms

- Both packages very flexible, and allow same syntax as basic lm/glm or lmer models, e.g.

```
rstan::stan_lm
rstan::stan_glm
rstan::stan_glmer
```

- Vignettes brms rstanarm

Existing packages: rstanarm and brms

- Very flexible `brms` includes autocorrelated errors, non-normal data, non-linear smooths (GAMs), etc.
- Limitations related to this class:
- allows multivariate data, but not multivariate time series models brms example

Existing packages: rstanarm and brms

`brms` offers notation that should be very familiar to run many classes of models,

```
brms::brm(y ~ x * z + (1|group), data=d)
brms::brm(y01 ~ x * z + (1|group), data=d,
          family = binomial("logit"))
brms::brm(bf(y ~ s(x)), data=d)
```

- smooths can also be of 2-d models (e.g. spatial models)

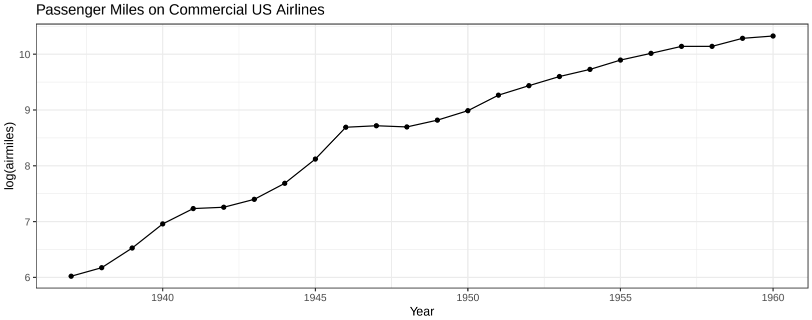
Existing packages: rstanarm and brms

`brms` allows ARMA correlation structures that we're familiar with,

```
data("LakeHuron")
LakeHuron <- as.data.frame(LakeHuron)
fit <- brm(x ~ arma(p = 2, q = 1),
           data = LakeHuron)
```

- also includes spatial models (car, sar)
- does not include these in the context of state space models

Example: linear regression in brms



Example: linear regression and AR(1) models in brms

- Regression

```
lm_fit = brms::brm(log(airmiles) ~ year, data=df)
```

- Question: how would we change the code to be an AR(1) model?

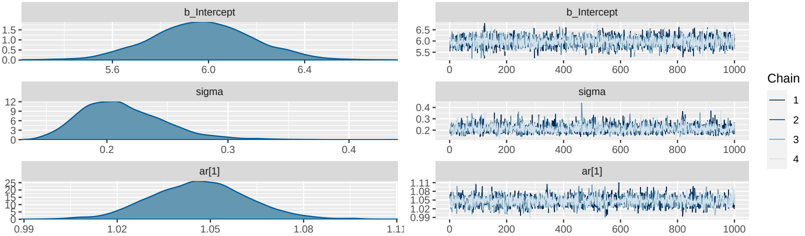
```
lm_ar= brms::brm(log(airmiles) ~ arma(p = 1, q = 0),
                  data=df)
```

- Defaults to 4 MCMC chains, 2000 iterations, 1000 burn-in

Example: linear regression and AR(1) models in brms

- `lm_ar` is a "brmsfit" object and has a bunch of convenient plotting functions

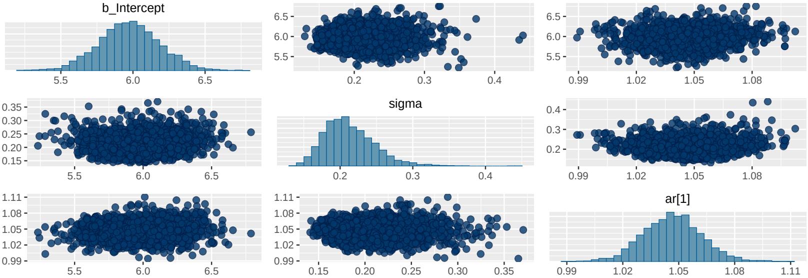
```
plot(lm_ar)
```



Example: linear regression and AR(1) models in brms

- Pairs plots

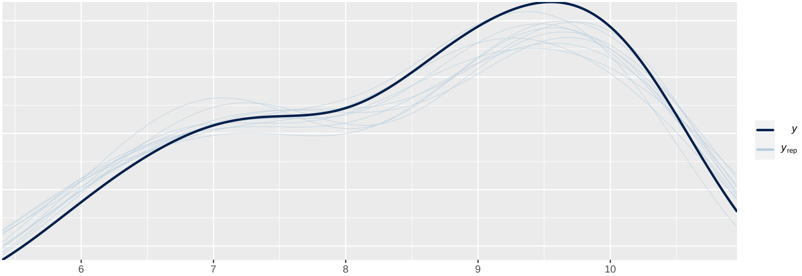
```
pairs(lm_ar)
```



Example: linear regression and AR(1) models in brms

- Posterior predictive checks

```
pp_check(lm_ar)
```



Example: linear regression and AR(1) models in brms

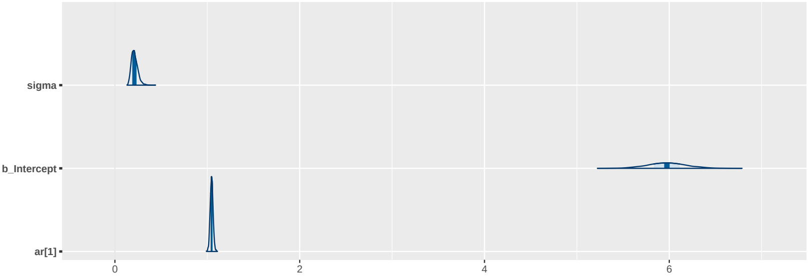
- Shinystan

```
shinystan::launch_shinystan(lm_ar)
```

Example: linear regression and AR(1) models in brms

- Additional functionality / diagnostics in bayesplot

```
mcmc_areas(lm_ar,c("sigma","b_Intercept","ar[1]"))
```



Plotting with Stan output

These plots only the tip of the iceberg for plotting. For more great examples of the kinds of plots avaialable, see these vignettes:

- Examples on Stan
- Jonah Gabry's introduction to bayesplot
- Matthew Kay's introduction to bayesplot and tidybayes

Customized models and code for this class

- We'll need to install these packages to run Stan,

```
install.packages("rstan",
                 repos = "https://cloud.r-project.org")
install.packages("devtools",
                 repos = "https://cloud.r-project.org")
```

- And then we can install our custom package for the class with bundled Stan time series models

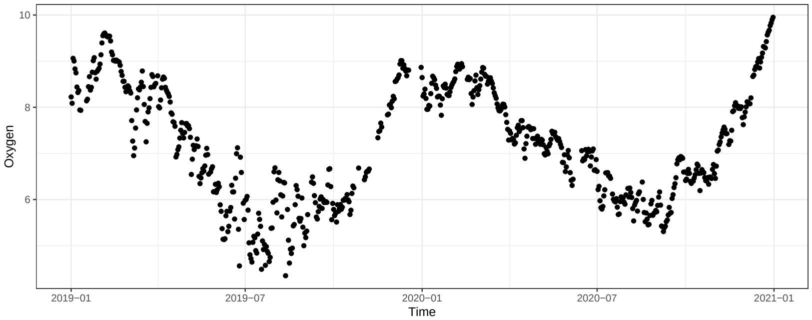
```
devtools::install_github(repo="atsa-es/atsar")
library("atsar")
```

Models included

- `atsar` package includes:
- RW, AR and MA models (with and without drift)
- DLMs (intercept, slope, both)
- State space RW and AR models
- Flexible families for each model

More time series models: application to NEON EFI Aquatics challenge

- Daily temperature and oxygen data available from Barco Lake in Florida



'atsar' package: random walk and AR(1) models

This model should be familiar,

$$E\left[Y_t\right] = E\left[Y_{t-1}\right] + e_{t-1}$$

* Note that the use of the argument `model_name` and `est_drift`. By not estimating drift, we assume the process is stationary with respect to the mean

```
rw = fit_stan(y = neon$oxygen,
              est_drift = FALSE, model_name = "rw")
```

‘atsar’ package: univariate state space models

- Specify the MCMC parameters

```
rw = fit_stan(y = neon$oxygen,
              est_drift = FALSE,
              model_name = "rw",
              mcmc_list = list(n_mcmc = 2000, n_burn = 500,
                               n_chain = 3, n_thin = 1))
```

'atsar' package: univariate state space models

State equation:

$$x_t = \phi x_{t-1} + \varepsilon_{t-1}$$

where $\varepsilon_{t-1} \sim Normal(0, q)$

Observation equation:

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

- Let's compare models with and without the AR parameter ϕ in the process model

'atsar' package: univariate state space models

We can first run the model with ϕ,

```
ss_ar = fit_stan(y = neon$oxygen,
        est_drift=FALSE, model_name = "ss_ar",
        mcmc_list = list(n_mcmc = 2000, n_chain = 1, n_thin
```

then without,

```
ss_rw = fit_stan(y = neon$oxygen,
        est_drift=FALSE, model_name = "ss_rw",
        mcmc_list = list(n_mcmc = 2000, n_chain = 1, n_thin
```

'atsar' package: univariate state space models

Did the models converge?

- One quick check is to look at the value of R-hat for each parameter (generally should be small, < 1.1 or < 1.05)

```
rw_summary <- summary(ss_rw, pars = c("sigma_process","sign
                      probs = c(0.1, 0.9))$summary
print(rw_summary)
```

```
##                       mean     se_mean          sd
## sigma_process 0.25328836 0.001908073 0.008570219 0.2418
## sigma_obs     0.04480232 0.005540708 0.014138820 0.0291
##                    n_eff      Rhat
## sigma_process 20.174086 0.9990057
## sigma_obs      6.511721 1.0170924
```

‘atsar’ package: univariate state space models

- Calculate maximum Rhat across all parameters,

```
rhats <- summary(ss_rw)$summary[,"Rhat"]
print(max(rhats))
```

```
## [1] 1.017997
```

- Reminder: we only ran one chain / 2000 iterations, so overall not bad!

'atsar' package: univariate state space models

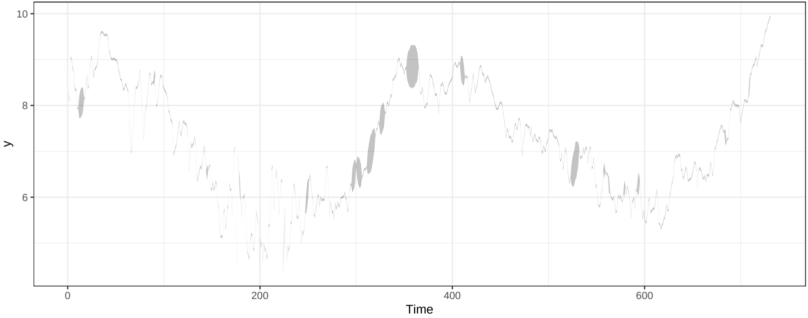
- Tidy summaries from Stan output: Using the `broom.mixed` package, we can also extract some tidy summaries of the output

```
coef = broom.mixed::tidy(ss_ar)
head(coef)
```

```
## # A tibble: 6 x 3
##   term          estimate std.error
##   <chr>            <dbl>     <dbl>
## 1 sigma_process    0.261   0.00731
## 2 pred[1]          8.22    0.0228
## 3 pred[2]          8.10    0.0263
## 4 pred[3]          9.05    0.0247
## 5 pred[4]          9.00    0.0230
## 6 pred[5]          8.83    0.0237
```

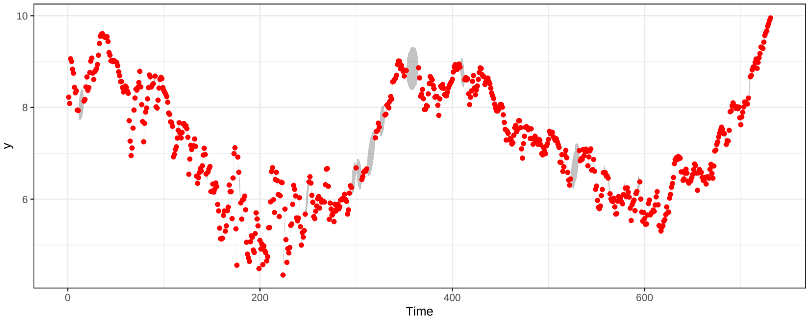
‘atsar’ package: univariate state space models

- We can use this to look at predictions versus our data



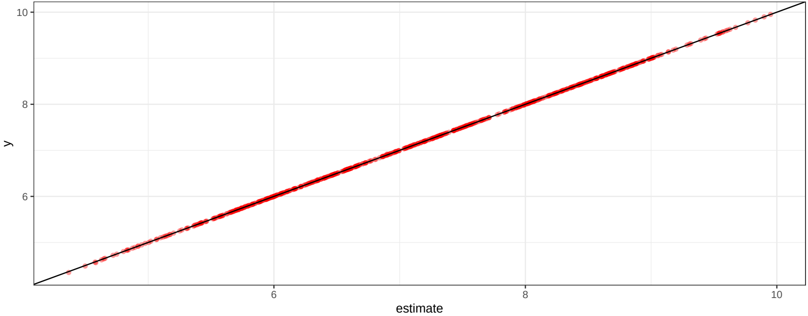
'atsar' package: univariate state space models

- We can use this to look at predictions versus our data



‘atsar’ package: univariate state space models

- We can use this to look at predictions versus our data



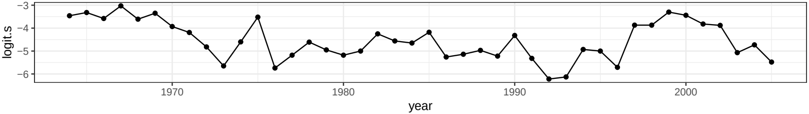
##‘atsar’ package: raw samples

- tidy() functions great at summarizing
- fit_stan() returns ‘stanfit’ object that we can use `rstan::extract()` on to get raw posterior draws, by chain

```
pars = extract(ss_ar)
```

'atsar' package: DLMs

- For comparison to MARSS, we'll use Mark's example of logit-transformed survival from the Columbia River. We can think about setting the DLM up in the slope or the intercept. For this first example, we'll do the latter.



'atsar' package: DLMs

- Fit DLM with random walk in intercept

```
mod = fit_stan(y = SalmonSurvCUI$logit.s,
               model_name="dlm-intercept",
               mcmc_list = list(n_mcmc = 2000,
                                n_chain = 1, n_thin = 1,n_b
```

- Fit DLM with random walk in slope

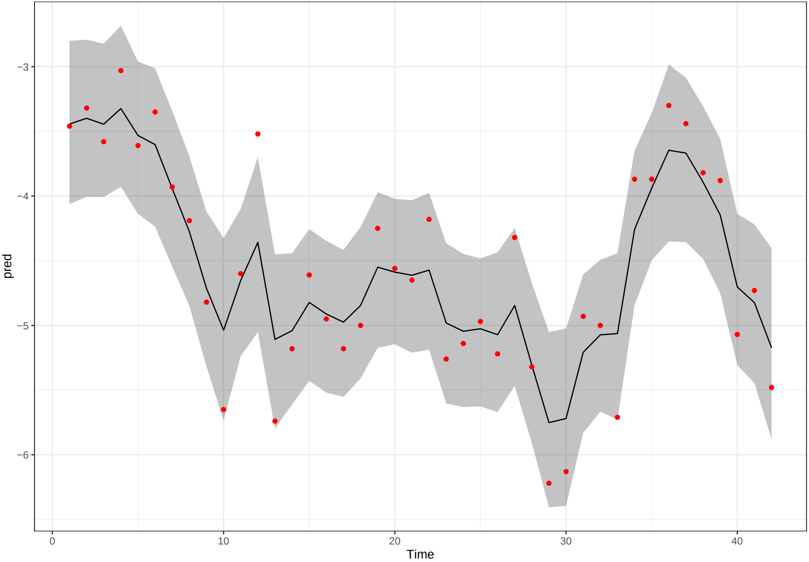
```
mod_slope = fit_stan(y = SalmonSurvCUI$logit.s,
        x = SalmonSurvCUI$CUI.apr,
        model_name="dlm-slope",
        mcmc_list = list(n_mcmc = 2000,
                                n_chain = 1, n_thin = 1,n_b
```

'atsar' package: DLMs

Let's look at predictions using the `rstan::extract()` function

'atsar' package: DLMs

Let's look at predictions using the `rstan::extract()` function



Extra extensions

- `family` argument in `fit_stan` allows to have flexible families
- e.g., fit a Poisson or binomial DLM with

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

Summary

- Bayesian implementation of time series models in Stan can do everything that MARSS can do and more!
- Very flexible language, great developer community
- Widely used by students in SAFS / UW / QERM / etc
- Please come to us with questions, modeling issues, or add to code in our packages to make them better!