[1] "Getting started in Bayesian modelling with Stan and rstan"

> bill.dixon at alumni.unimelb.edu.au

$$Y = a + b * X + \varepsilon$$

> ?uncertainty

Aleatory:

- true ‘random’ variability,
- irreducible but often better characterised

Incertitude / Epistemic:

- Model uncertainty
- abstraction, ignorance
- Often ignored

$$Y \sim \text{normal}(mu, sigma)$$
$$mu = \text{a} + \text{b}*X$$

> ?bayesian

prior

likelihood

posterior

$$p(\theta|y) = \frac{p(\theta)p(y|\theta)}{p(y)}$$

average likelihood /
normalising constant

$$p(y) \;=\; \textstyle\sum_{\theta} p(\theta)p(y|\theta)$$

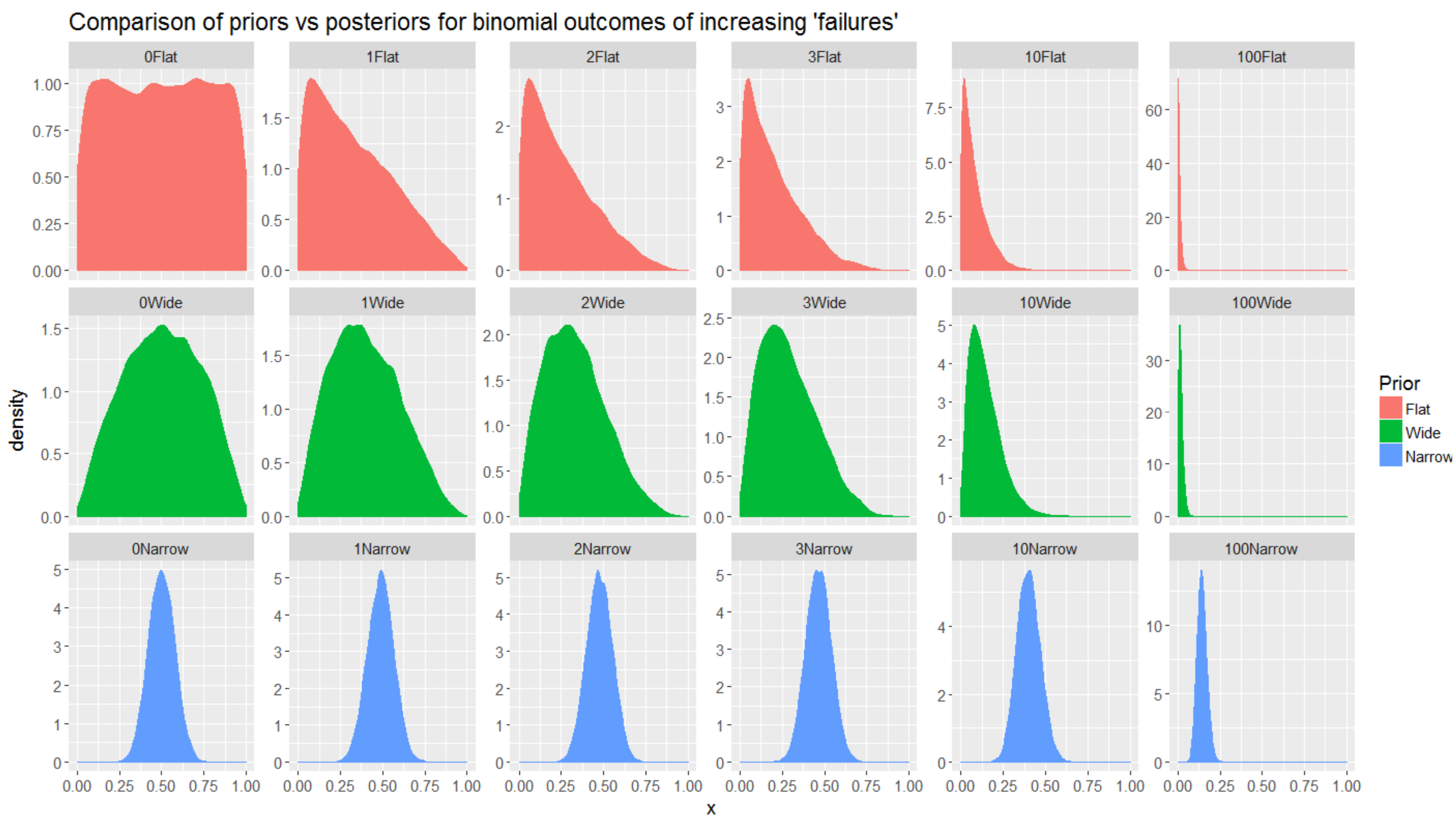
$$\int p(\theta)p(y|\theta)d\theta$$ (continuous)

BDA3, p7

> pirors

noninformative vs informative

conjugacy



Model:

Y ~ normal(*mu*, *sigma*);

mu = a + b***X*;

Priors:

sigma ~ cauchy(0, 5);

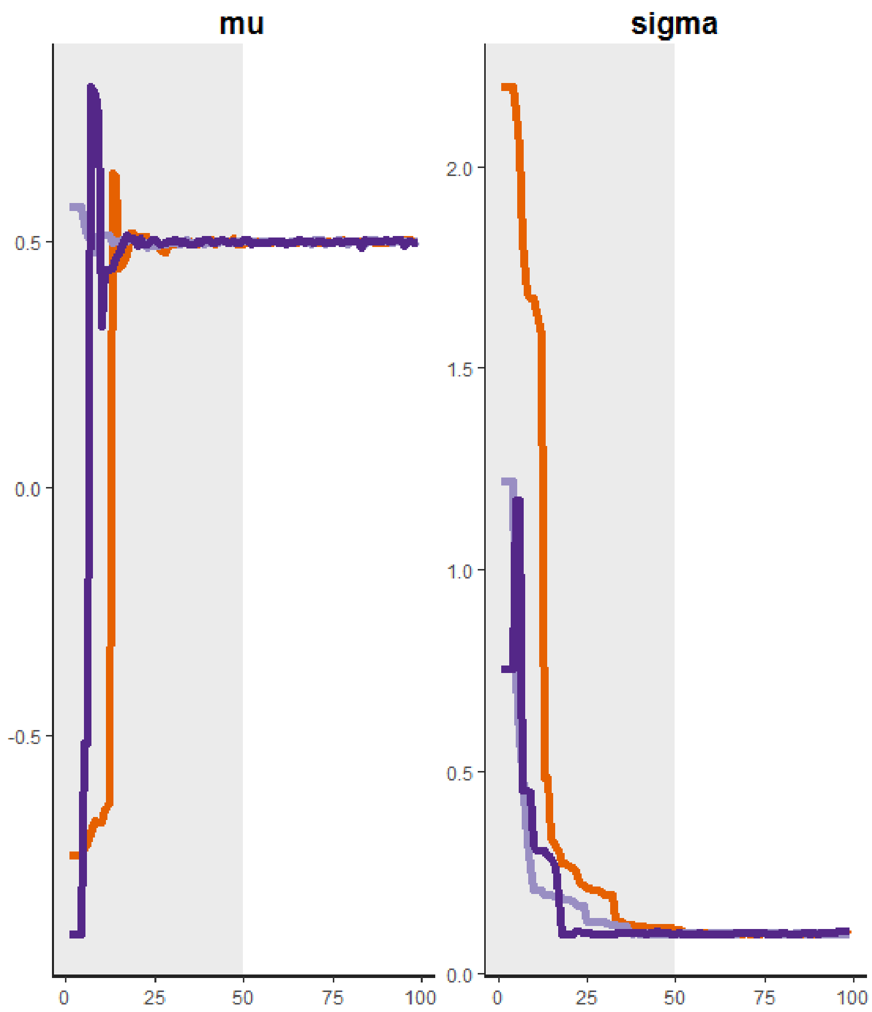
a ~ normal(0, 10);

b ~ normal(0, 10);

- Numeric integration (MCMC)
- Multiple chains

> MCMC

- Numeric integration
- Multiple chains



> Why?

- Philosophical reasons
- Be explicit about prior info / state of knowledge
- Decision making (optimisation, utility, etc)
- Updating
- Model structures
- Characterise and propagate uncertainty
- Combine (causal) inference, prediction and assessment

rstan> installation

- Stan is a probabilistic programming language
- C++ library for Bayesian modeling and inference
- Can be run from R (rstan)
- Primarily uses the No-U-Turn sampler (NUTS)
- Needs rtools
- install.packages("rstan", dependencies=TRUE)

```
devtools::find_rtools()
Sys.getenv('PATH')
```

https://github.com/stan-dev/rstan/wiki/Installing-RStan-on-Windows

> workflow

(write model)

- Data (list)
- Model (save as .stan file best)
- Compile model
- Run model (pass data and model spec. to stan)
- Analyse output and ‘extract’ samples.

(Compare models etc.)

> model.blocks

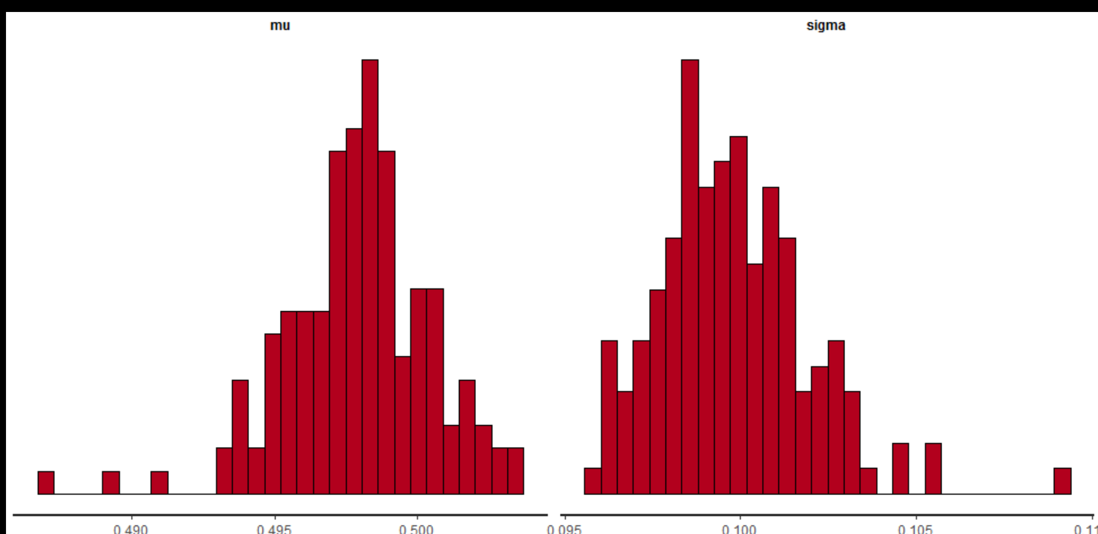
```
functions {
}
data {
}
transformed data {
}
parameters {
}
transformed parameters {
}
model {
}
generated quantities {
}
```

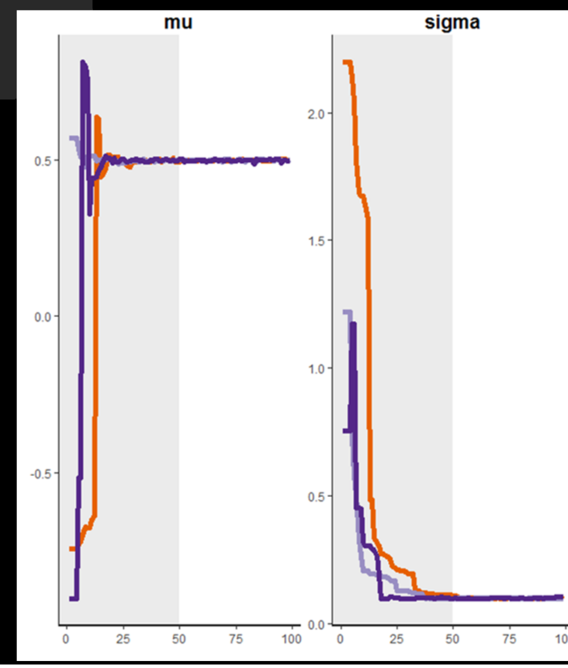
- Line must end in " ; "
- Comments " // "
- Models support vectors

> baisc.model(dnorm)

```
data{
 int<lower=0>          N;  // No. obs
 vector<lower=0>[N]  x;  // data
}
parameters{
  real                         mu;   // mean
  real<lower=0>    sigma;   // standard deviation
}
model{
  x ~ normal(mu, sigma);    // univariate distribution
}
```

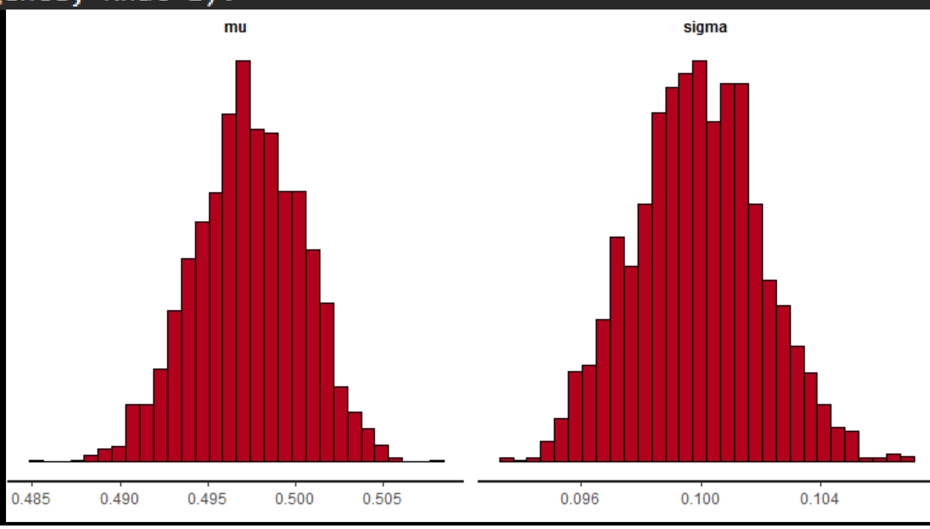
```
Inference for Stan model: 85831eef6fda8921202a26d9d49640ee.
3 chains, each with iter=100; warmup=50; thin=1;
post-warmup draws per chain=50, total post-warmup draws=150.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
mu       0.50    0.00 0.00    0.49    0.50    0.50    0.50    0.50   123 1.01
sigma    0.10    0.00 0.00    0.10    0.10    0.10    0.10    0.10    38 1.02
lp__  1802.66    0.15 1.06 1800.11 1802.49 1803.04 1803.28 1803.43    51 1.06

Samples were drawn using NUTS(diag_e) at Mon Sep 11 16:22:37 2017.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

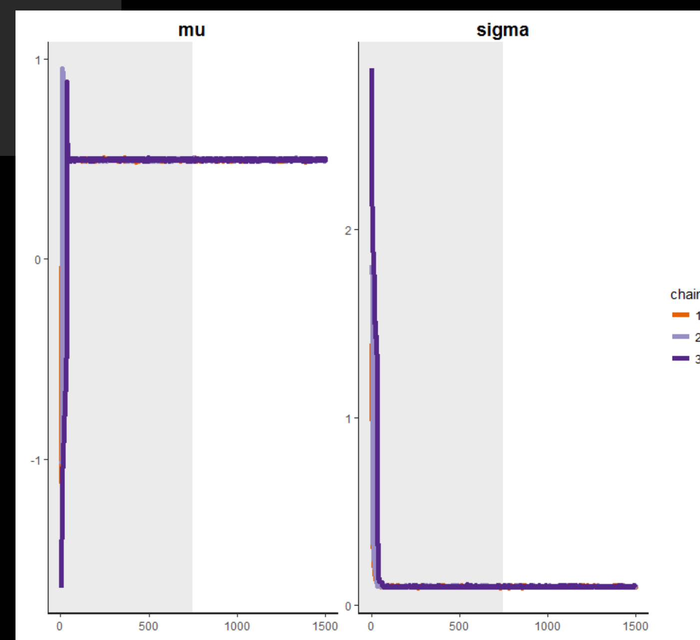




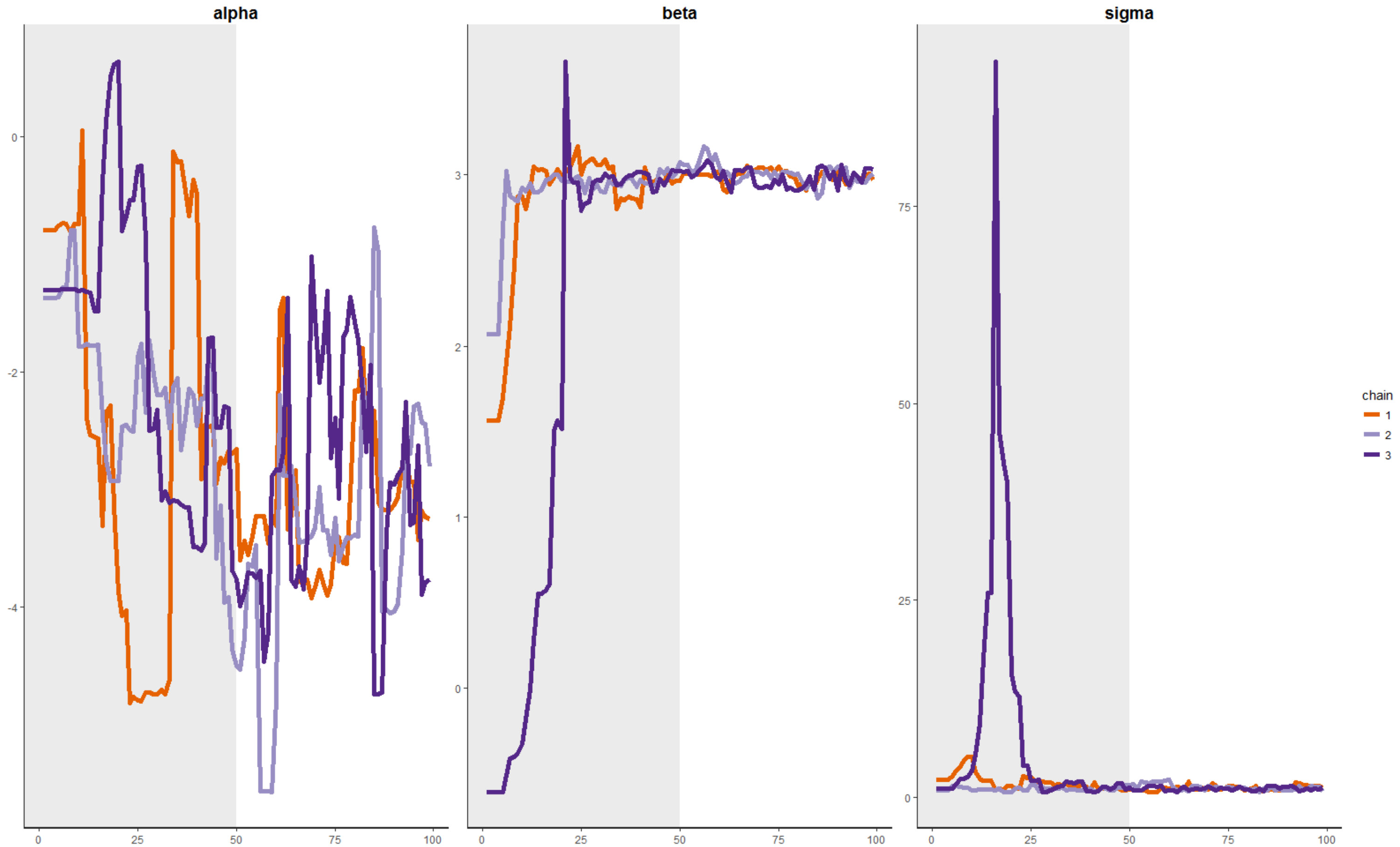
More samples: iter=1500

```
Inference for Stan model: 85831eef6fda8921202a26d9d49640ee.
3 chains, each with iter=1500; warmup=750; thin=1;
post-warmup draws per chain=750, total post-warmup draws=2250.

         mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
mu       0.50    0.00 0.00    0.49    0.50    0.50    0.50    0.50  2250    1
sigma    0.10    0.00 0.00    0.10    0.10    0.10    0.10    0.10  1204    1
lp__  1802.52    0.03 0.89 1800.16 1802.13 1802.77 1803.16 1803.42  1227    1

Samples were drawn using NUTS(diag_e) at Mon Sep 11 16:33:31 2017.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```





>model.checking



```
> baisc.model(dnorm)
model{
    x ~ normal(mu, sigma);            // vector notation

    target+= normal_lpdf(x | mu2, sigma2); // increment

for(i in 1:N){
     x[i] ~ normal(mu3, sigma3);   // loop style
     }
}
```

> linear.model

```
data {
  int<lower=0> N;   // Num obs
  vector[N]   Y;      // Obs times
  vector[N]   X;      // Obs Xances
}
```

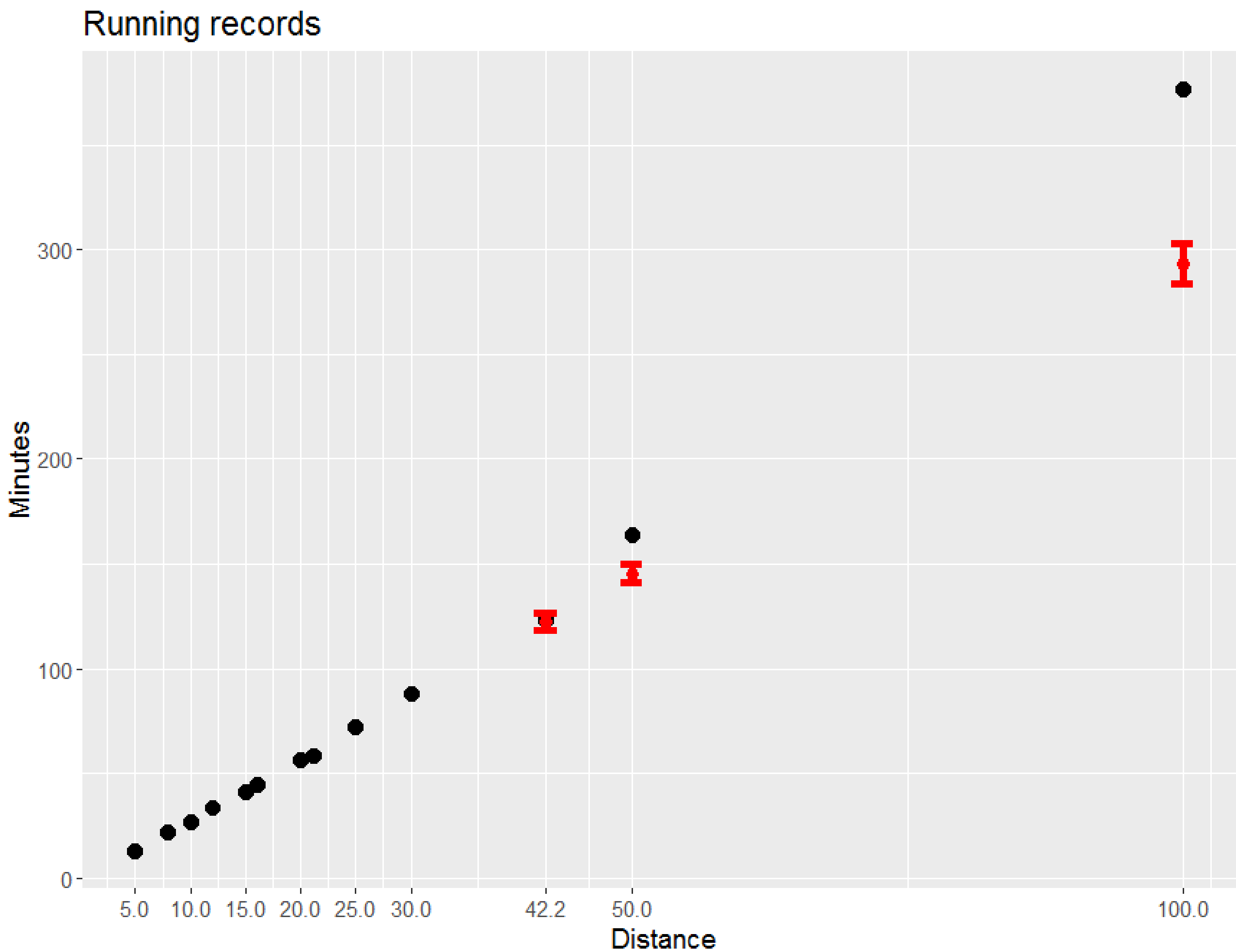
```
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
}
model {
  // priors -------
  alpha ~ normal(0,10);
  beta ~ normal(0,10);
  sigma ~ cauchy(0,5);
  // model --------
  Y ~ normal(alpha + beta * X, sigma);
}
```

> linear.model

```
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
alpha -2.47    0.04 1.00 -4.49 -3.04 -2.46 -1.85 -0.56   514 1.01
beta   2.96    0.00 0.06  2.86  2.93  2.96  2.99  3.07   534 1.01
sigma  1.21    0.02 0.39  0.74  0.96  1.14  1.35  2.25   494 1.01
```

```
> runf <- run[1:10,] %>% select(Distance, Minutes)
> fit <- lm(Minutes~Distance, data=runf)
> fit

Call:
lm(formula = Minutes ~ Distance, data = runf)

Coefficients:
(Intercept)     Distance
     -2.488        2.963
```

```
> linear.model
```



> log.rules

- Calculations often involve product of small probabilities
- Smaller than machine epsilon/precision
- Use logs to avoid under/overflow and floating point errors:

log(a*b) == log(a) + log(b)
log(a/b) == log(a) - log(b)

>loo("leave one out")

- approximate LOO using Pareto Smoothed Importance Sampling (PSIS).

    elpd_loo - (expected log predictive density),

    p_loo - (effective number of parameters), and

    looic - (the LOO information criterion).

- compare(model1, model2)
- WAIC, DIC, BIC etc.
- Model comparison/averaging

https://github.com/stan-dev/loo/blob/master/vignettes/loo-example.Rmd

> predictive.evaluation(lppd)

*__Out of Sample predictive evaluation__

```
generated quantities {
real  log_p_new;     // posterior predictive log density test data
 log_p_new = 0;
  for(i in 1:N_nu){
log_p_new = log_p_new + normal_lpdf(Y_nu[i] | alpha + X_nu[i]*beta, sigma);
}}
```

- NB: this gives posterior expectation of the log predictive density:

  (**expectation of the log**)

- We want: log of posterior predictive density (lppd):

  (**log of the expectation**)

Therefore needs to be 'corrected'

(-log(m)+log_sum_exp(log *p(y_nu*| *theta*)))

http://mc-stan.org/users/documentation/case-studies/pool-binary-trials.html (also McElreath, p193)

> stan.advantages

- Support, resources / momentum
- Development
- algorithms (HMC/NUTS)
- faster convergence & better explore parameter space
- vectorisation
- supports parallelisation
- functions, distributions
- saving compiled models

> bayes.challenges

- Complicated
- Time consuming
- Unresolved issues / developing field
- thinning, h-centering, priors, goodness-of-fit etc.
- Computational complexity / large models
- (Latent) discrete unknown parameters (stan – can be done but not directly)

> other.packages

- shinystan
- ggmcmc
- loo
- rstanarm*
- brms*
- Rethinking (github)*
- Prophet

*Do some of the model building for you

> resources.bayes

Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, Donald B. Rubin, 2013, **Bayesian Data Analysis, Third Edition,** CRC press.

John Kruschke, 2014, **Doing Bayesian Data Analysis, 2nd Edition:** A Tutorial with R, JAGS, and Stan, Elsevier.

Richard McElreath, 2015, **Statistical Rethinking**: A Bayesian Course with Examples in R and Stan, CRC press.

Adam Branscum, Ronald Christensen, Timothy E Hanson, and Wesley O. Johnson, 2010, **Bayesian Ideas and Data Analysis**: An Introduction for Scientists and Statisticians, CRC press

John Kruschke & Liddell, T., 2017, **Bayesian data analysis for newcomers**, *Psychonomic Bulletin & Review, https://psyarxiv.com/nqfr5*

 # Philosophy:

John K. Kruschke and Liddell, T.M., 2017, **The Bayesian New Statistics**: Hypothesis testing, estimation, meta-analysis, and power analysis from a Bayesian perspective*, Psychonomic Bulletin & Review*

Andrew Gelman and Shalizi, R.S., 2012, **Philosophy and the practice of Bayesian statistics**, *British Journal of Mathematical and Statistical Psychology*, 66:8-38

> resources.stan

Stan reference manual
http://mc-stan.org/
http://mc-stan.org/users/**documentation**/index.html
http://**discourse**.mc-stan.org/
https://github.com/stan-dev/rstan
https://cran.r-project.org/web/packages/rstan/**vignettes**/rstan.html
http://**modernstatisticalworkflow**.blogspot.com.au/
http://**doingbayesiandataanalysis**.blogspot.com

http://www.mcmchandbook.net/

Kaggle:
http://blog.kaggle.com/2017/05/19/march-machine-learning-mania-1st-place-winners-interview-andrew-landgraf/