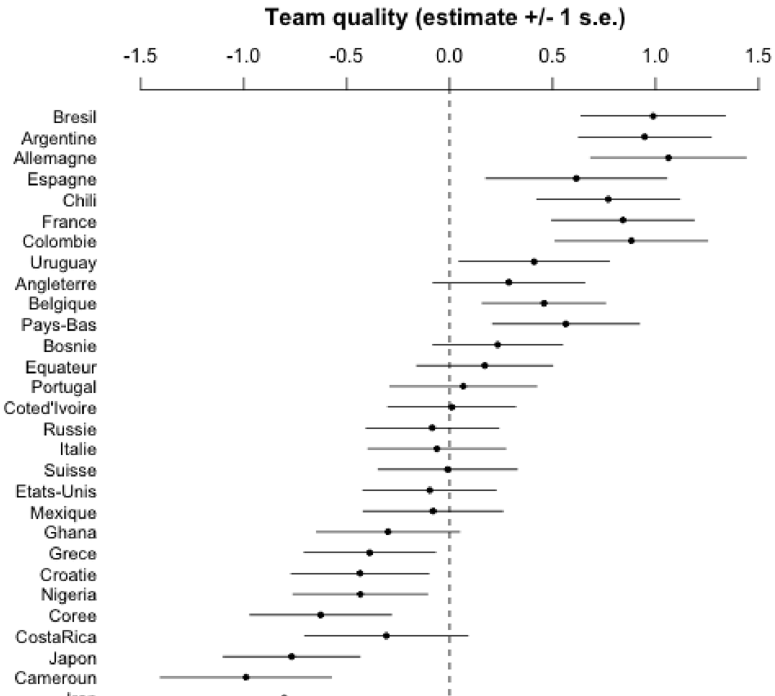
Bayesian data analysis and Stan

Andrew Gelman
Dept of Statistics and Dept of Political Science
Columbia University, New York

New York, 18–20 July 2016

1. Real-life Stan

- Soccer
- Golf
- “Global climate challenge”



Team quality (estimate +/- 1 s.e.)
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5
Bresil
Argentine
Allemagne
Espagne
Chili
France
Colombie
Uruguay
Angleterre
Belgique
Pays-Bas
Bosnie
Equateur
Portugal
Coted'Ivoire
Russie
Italie
Suisse
Etats-Unis
Mexique
Ghana
Grece
Croatie
Nigeria
Coree
CostaRica
Japon
Cameroun

worldcup2012.txt

```
Bresil 3 Croatie 1
Mexique 1 Cameroun 0
Bresil 0 Mexique 0
Cameroun 0 Croatie 4
Cameroun 1 Bresil 4
Croatie 1 Mexique 3
Espagne 1 Pays-Bas 5
Chili 3 Australie 1
Espagne 0 Chili 2
Australie 2 Pays-Bas 3
Australie 0 Espagne 3
Pays-Bas 2 Chili 0
Colombie 3 Grece 0
Coted'Ivoire 2 Japon 1
Colombie 2 Coted'Ivoire 1
Japon 0 Grece 0
Japon 1 Colombie 4
Grece 2 Coted'Ivoire 1
Uruguay 1 CostaRica 3
Angleterre 1 Italie 2
Uruguay 2 Angleterre 1
```

soccerpowerindex.txt

```
Bresil
Argentine
Allemagne
Espagne
Chili
France
Colombie
Uruguay
Angleterre
Belgique
Pays-Bas
Bosnie
Equateur
Portugal
Coted'Ivoire
Russie
Italie
Suisse
Etats-Unis
Mexique
```

The model in Stan

```
parameters {
  real b;
  real<lower=0> sigma_a;
  real<lower=0> sigma_y;
  vector[nteams] eta_a;
}
transformed parameters {
  vector[nteams] a;
  a = b*prior_score + sigma_a*eta_a;
}
model {
  eta_a ~ normal(0,1);
  sqrt_dif = student_t(df, a[team1]-a[team2],sigma_y);
}
```

Stan program (part 1)

```
data {
  int nteams;
  int ngames;
  vector[nteams] prior_score;
  int team1[ngames];
  int team2[ngames];
  vector[ngames] score1;
  vector[ngames] score2;
  real df;
}
transformed data {
  vector[ngames] dif;
  vector[ngames] sqrt_dif;
  dif = score1 - score2;
  sqrt_dif = (step(dif)-.5)*sqrt(fabs(dif));
}
```

Stan program (part 2)

```
parameters {
  real b;
  real<lower=0> sigma_a;
  real<lower=0> sigma_y;
  vector[nteams] eta_a;
}
transformed parameters {
  vector[nteams] a;
  a = b*prior_score + sigma_a*eta_a;
}
model {
  eta_a ~ normal(0,1);
  sqrt_dif ~ student_t(df, a[team1]-a[team2],sigma_y);
}
```

Fitting the model

- Go into R
- Read in the data
- Fit the Stan model
- Check convergence
- Graph the estimated team abilities
- Re-fit without prior information
- Compare to model with prior information

Load Stan into R

```
> setwd("~/AndrewFiles/teaching/stan_short_course/worldcup")
> library ("rstan")
Loading required package: ggplot2
rstan (Version 2.9.0-3, packaged: 2016-02-11 15:54:41 UTC, GitRe
For execution on a local, multicore CPU with excess RAM we recom
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
> rstan_options(auto_write = TRUE)
> options(mc.cores = parallel::detectCores())
```

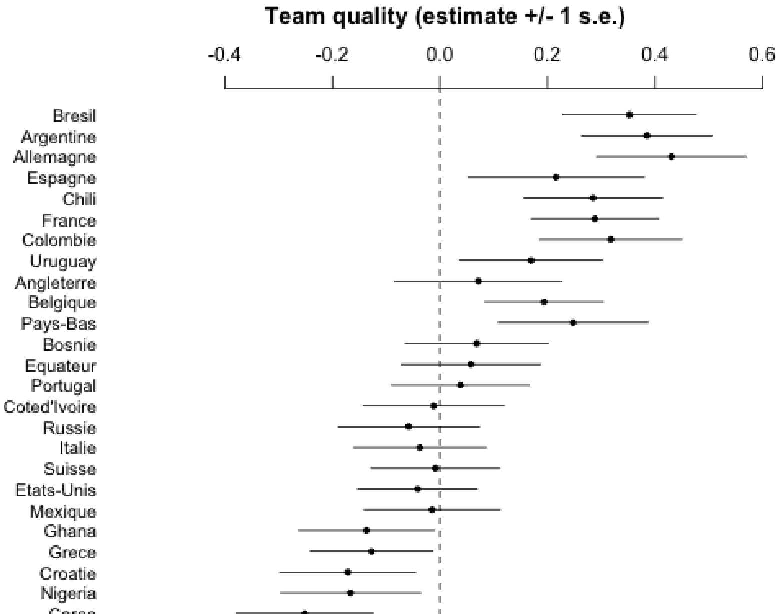
Fit the model

```
teams <- as.vector(unlist(read.table("soccerpowerindex.txt",
  header=FALSE)))
nteams <- length(teams)
prior_score <- rev(1:nteams)
prior_score <- (prior_score - mean(prior_score))/
  (2*sd(prior_score))

data2014 <- read.table("worldcup2014.txt", header=FALSE)
ngames <- nrow (data2014)

team1 <- match (as.vector(data2014[[1]]), teams)
score1 <- as.vector(data2014[[2]])
team2 <- match (as.vector(data2014[[3]]), teams)
score2 <- as.vector(data2014[[4]])

df <- 7
fit <- stan("worldcup_first_try.stan", iter=100, chains=4)
```

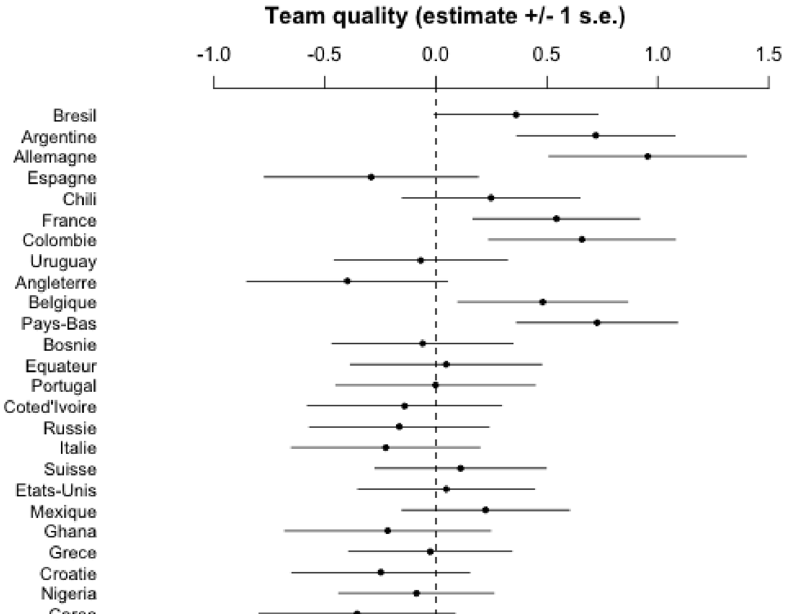
Check convergence

```
> print(fit)
Inference for Stan model: worldcup_first_try.
4 chains, each with iter=100; warmup=50; thin=1;
post-warmup draws per chain=50, total post-warmup draws=200.

           mean se_mean   sd   25%   50%  75% n_eff Rhat
b          0.47    0.01 0.09  0.42  0.47 0.54   200 0.99
sigma_a    0.12    0.01 0.07  0.07  0.12 0.16    67 1.05
sigma_y    0.43    0.00 0.05  0.39  0.43 0.46   126 1.03
eta_a[1]  -0.17    0.06 0.90 -0.84 -0.19 0.48   200 0.99
eta_a[2]   0.10    0.06 0.91 -0.53  0.14 0.75   200 0.99
eta_a[3]   0.48    0.06 0.83  0.00  0.53 0.98   200 1.00
eta_a[4]  -0.57    0.06 0.86 -1.19 -0.57 0.05   200 1.01
eta_a[5]   0.02    0.07 0.98 -0.69 -0.02 0.79   200 0.98
. . .
```

Graph the estimates



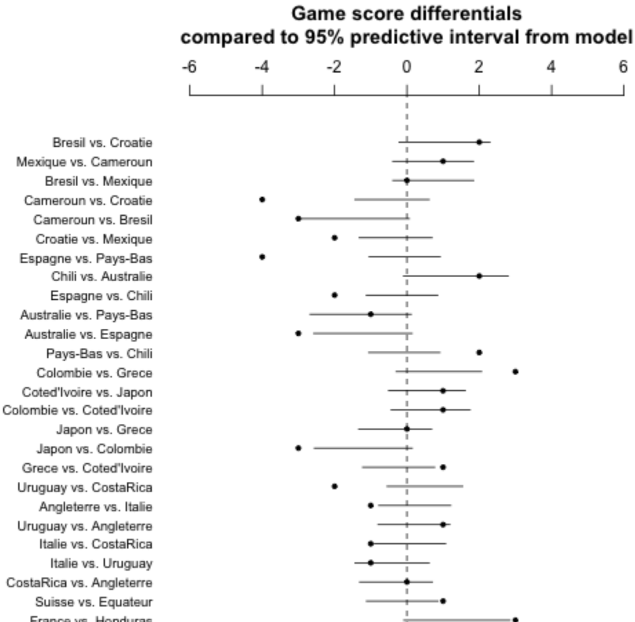
Re-fit the model without prior rankings

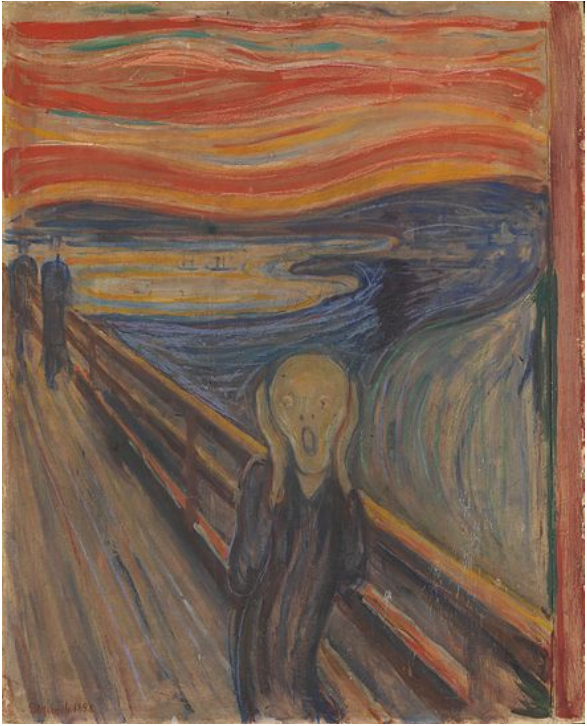


Checking model fit

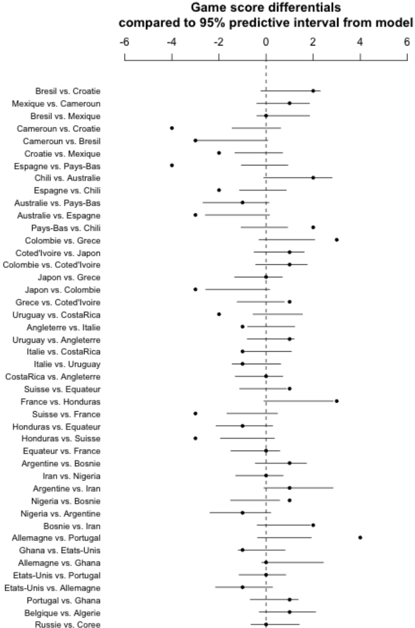
- Still inside R
- For each game, plot actual score differential and 95% predictive intervals
  - Not cross-validated but no big deal in this case because n is large
- The predictions don't fit the data!!
- Redoing the predictive intervals
- Re-plot, still a problem!

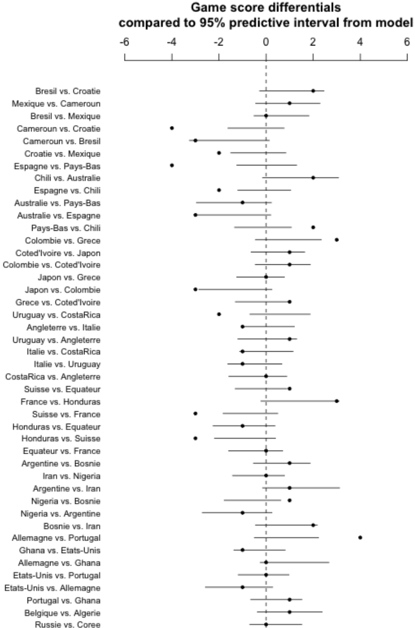
Compare model to predictions



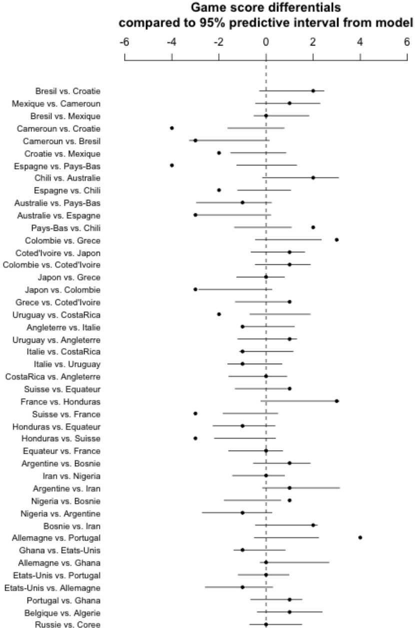


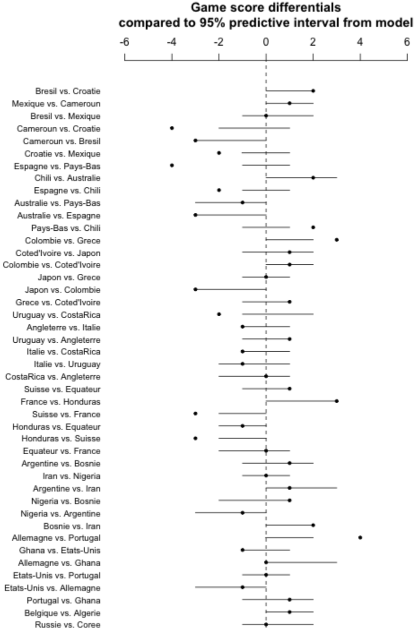
Use posterior simulations rather than point estimates



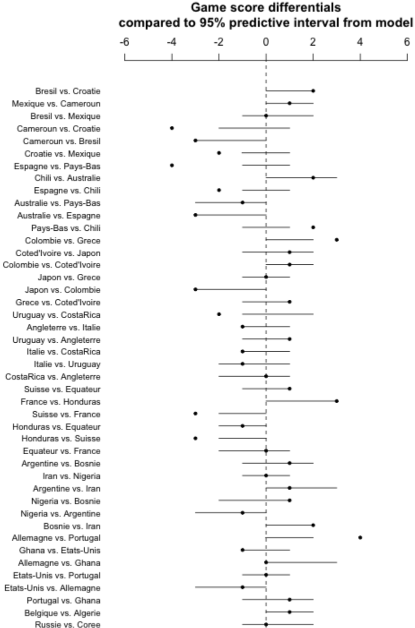


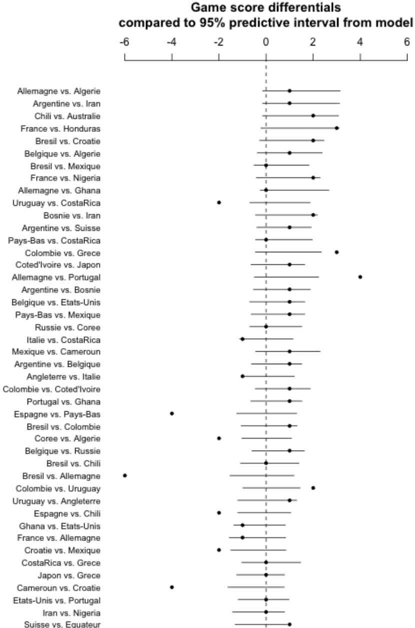
Round the predictions to integer score differentials



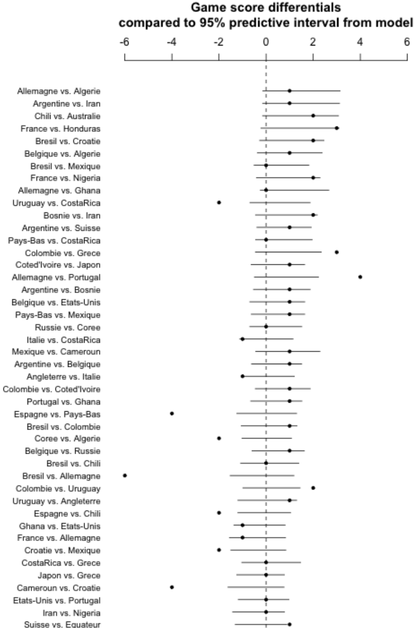


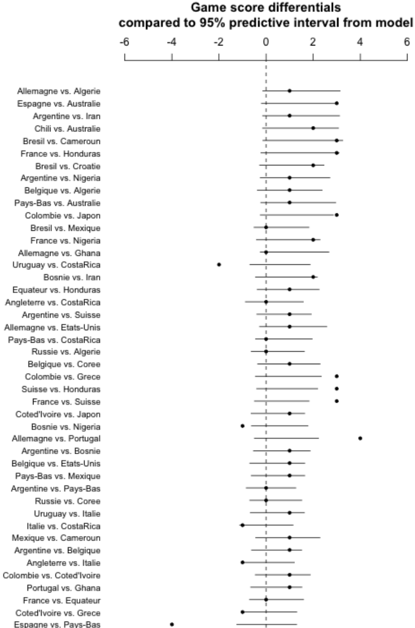
Plot in order of predicted score differential





Sign so expected outcomes are always positive





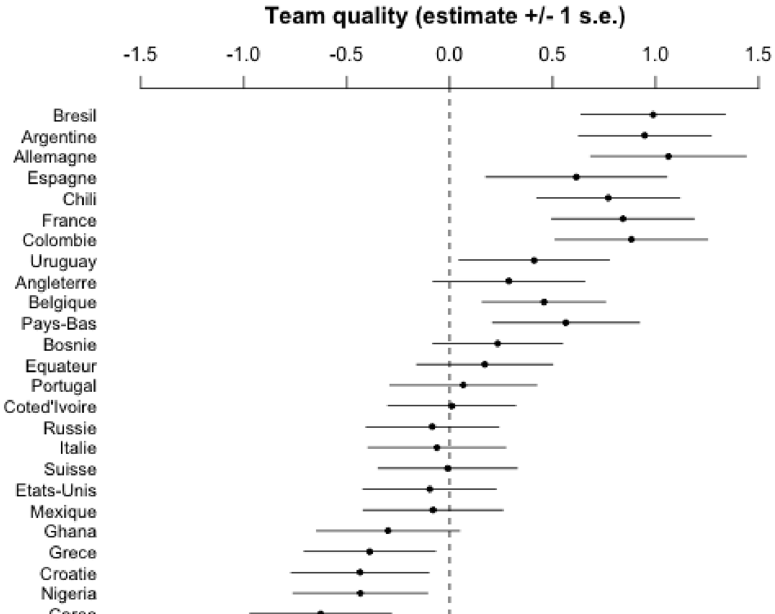
I found the bug!

- Still inside R
- Re-fit the model on the original scale
- Display the estimated team abilities
- Updated plot of data with predictive intervals—now it's ok!
- Go back and find the bug in the square-root-scale model
- Re-fit the debugged model

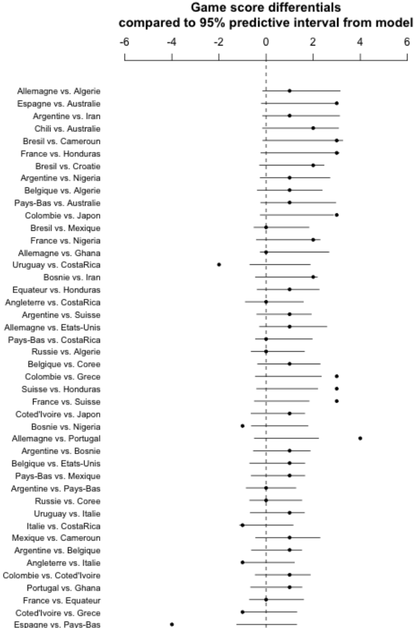
Our original model in Stan

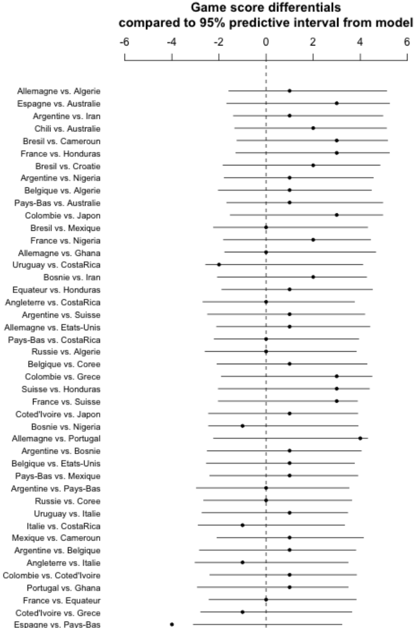
```
parameters {
  real b;
  real<lower=0> sigma_a;
  real<lower=0> sigma_y;
  vector[nteams] eta_a;
}
transformed parameters {
  vector[nteams] a;
  a = b*prior_score + sigma_a*eta_a;
}
model {
  eta_a ~ normal(0,1);
  sqrt_dif ~ student_t(df, a[team1]-a[team2],sigma_y);
}
```

Model fit to data on raw scale



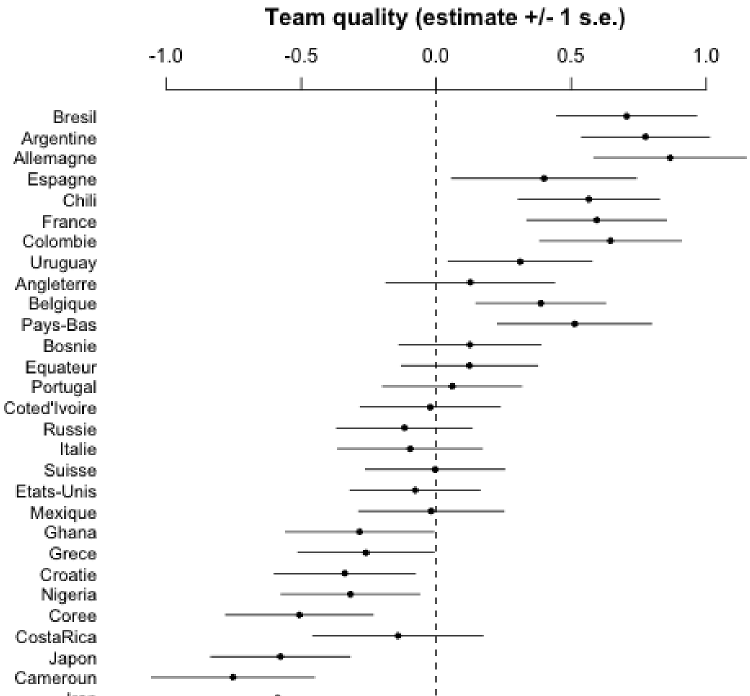
Compare data to predictive intervals



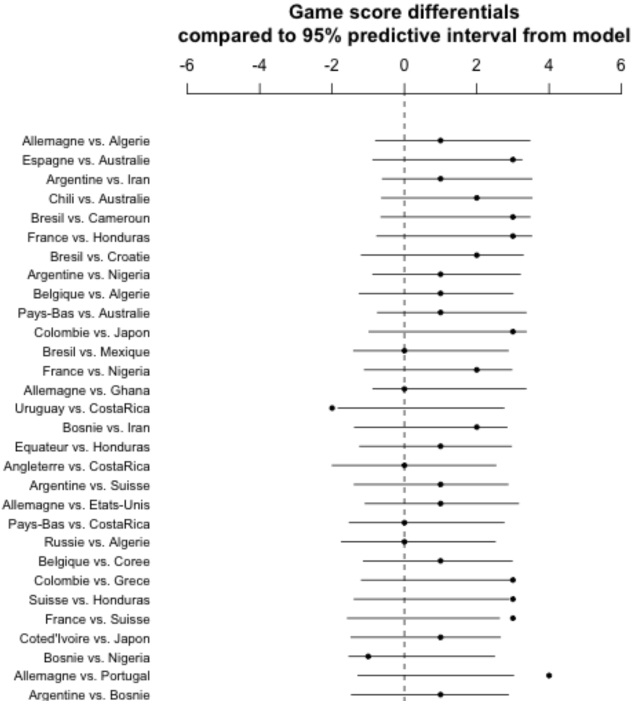


Our original Stan program (part 1)

```
data {
  int nteams;
  int ngames;
  vector[nteams] prior_score;
  int team1[ngames];
  int team2[ngames];
  vector[ngames] score1;
  vector[ngames] score2;
  real df;
}
transformed data {
  vector[ngames] dif;
  vector[ngames] sqrt_dif;
  dif = score1 - score2;
  sqrt_dif <- (step(dif)-.5)*sqrt(fabs(dif));
}
```

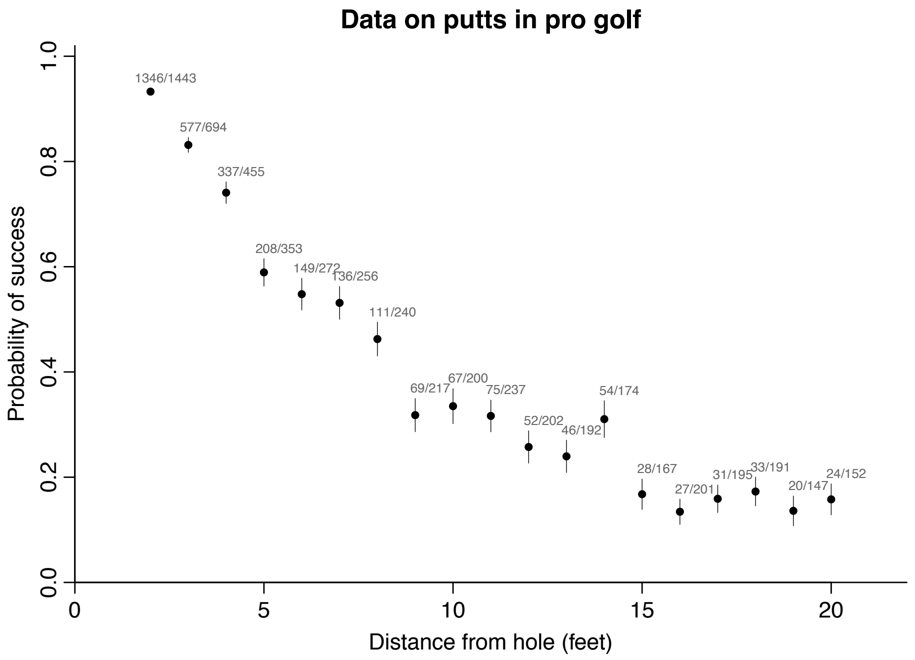


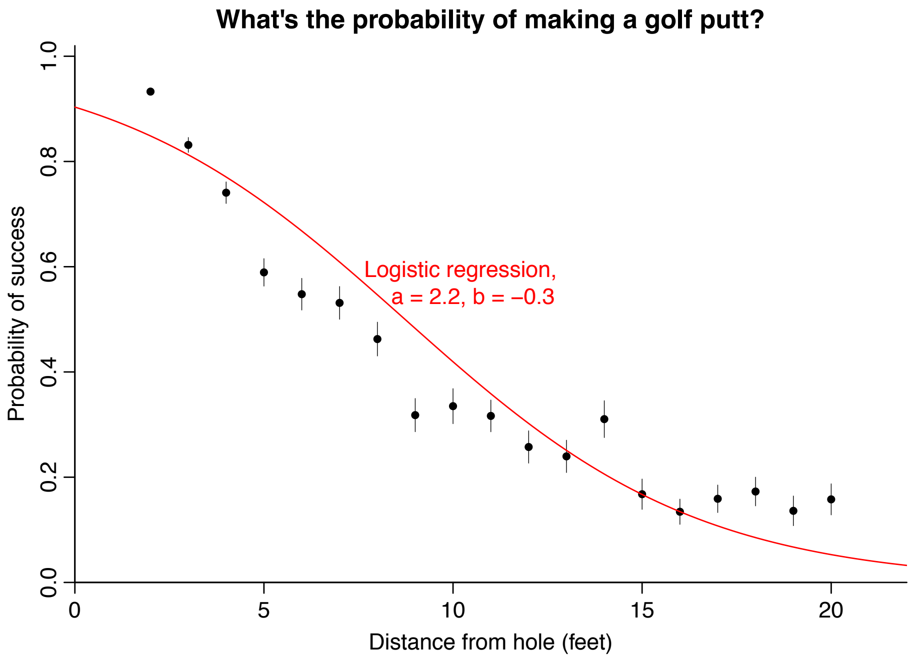
Team quality (estimate +/- 1 s.e.)
-1.0
-0.5
0.0
0.5
1.0
Bresil
Argentine
Allemagne
Espagne
Chili
France
Colombie
Uruguay
Angleterre
Belgique
Pays-Bas
Bosnie
Equateur
Portugal
Coted'Ivoire
Russie
Italie
Suisse
Etats-Unis
Mexique
Ghana
Grece
Croatie
Nigeria
Coree
CostaRica
Japon
Cameroun


Game score differentials
compared to 95% predictive interval from model
-6
-4
-2
0
2
4
6
Allemagne vs. Algerie
Espagne vs. Australie
Argentine vs. Iran
Chili vs. Australie
Bresil vs. Cameroun
France vs. Honduras
Bresil vs. Croatie
Argentine vs. Nigeria
Belgique vs. Algerie
Pays-Bas vs. Australie
Colombie vs. Japon
Bresil vs. Mexique
France vs. Nigeria
Allemagne vs. Ghana
Uruguay vs. CostaRica
Bosnie vs. Iran
Equateur vs. Honduras
Angleterre vs. CostaRica
Argentine vs. Suisse
Allemagne vs. Etats-Unis
Pays-Bas vs. CostaRica
Russie vs. Algerie
Belgique vs. Coree
Colombie vs. Grece
Suisse vs. Honduras
France vs. Suisse
Coted'Ivoire vs. Japon
Bosnie vs. Nigeria
Allemagne vs. Portugal
Argentine vs. Bosnie

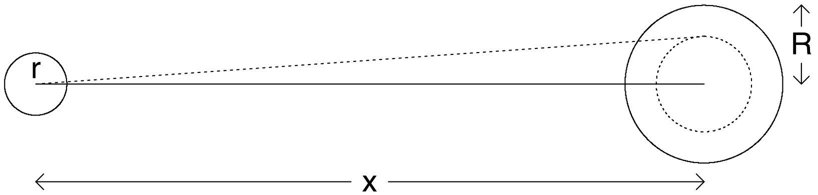
Lessons from World Cup example

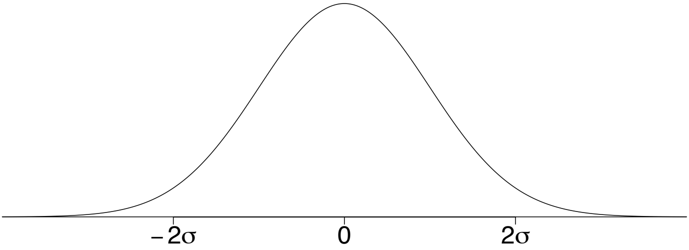
- Model score differential, not simple wins and losses—even if your only goal is to predict wins and losses
- Same thing in education (model test scores rather than pass/fail) and elections (model vote share not win/loss)
- Jump in and fit a model, then check its fit to data
- Combine sources of information
- Compare different fits graphically


Data on putts in pro golf
Probability of success
Distance from hole (feet)
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
1346/1443
577/694
337/455
208/353
149/272
136/256
111/240
69/217
67/200
75/237
52/202
46/192
54/174
28/167
27/201
31/195
33/191
20/147
24/152


What's the probability of making a golf putt?
Probability of success
0.0
0.2
0.4
0.6
0.8
1.0
Logistic regression,
a = 2.2, b = −0.3
0
5
10
15
20
Distance from hole (feet)

Geometry-based model





Stan code

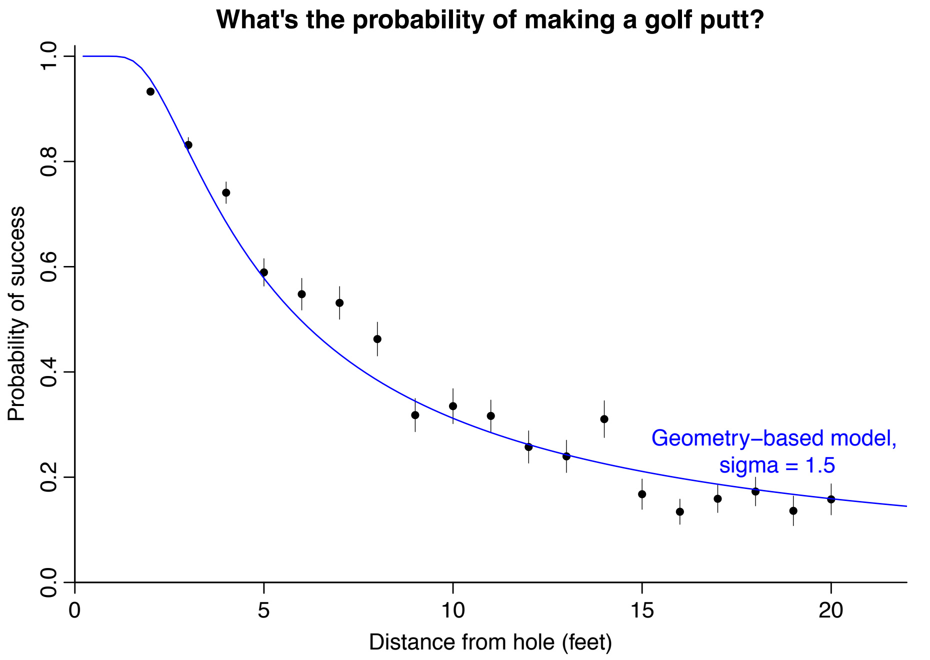
```
data {
  int J;
  int n[J];
  real x[J];
  int y[J];
  real r;
  real R;
}
parameters {
  real<lower=0> sigma;
}
model {
  real p[J];
  p = 2*Phi(asin((R-r)/x) / sigma) - 1;
  y ~ binomial(n, p);
}
```

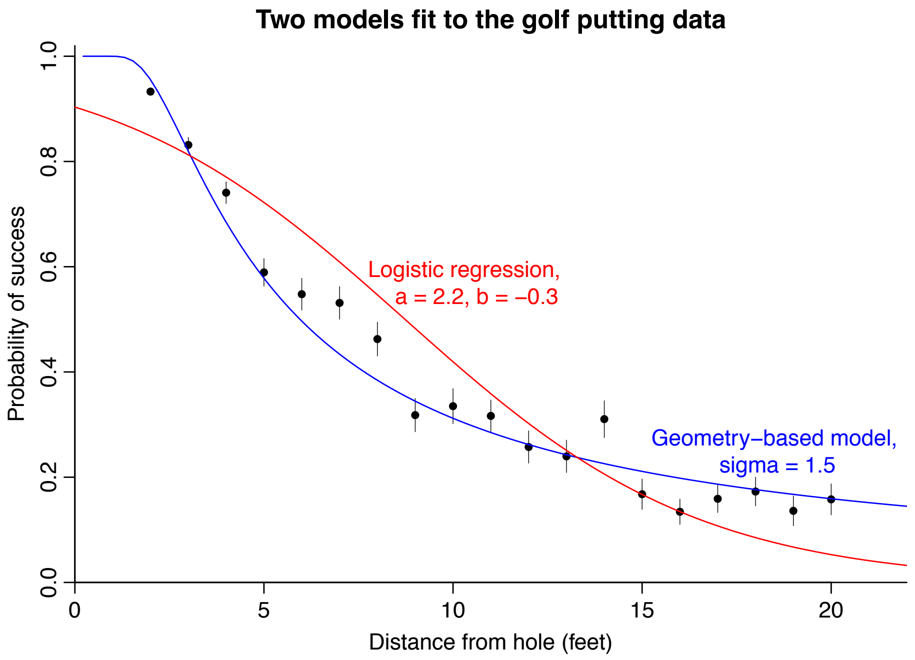
Fit the model

```
golf <- read.table("golf.txt", header=TRUE, skip=2)
x <- golf$x
y <- golf$y
n <- golf$n
J <- length(y)
r <- (1.68/2)/12
R <- (4.25/2)/12
se <- sqrt((y/n)*(1-y/n)/n)

fit1 <- stan("golf1.stan")
```

Check convergence

```
> print(fit1)
Inference for Stan model: golf1.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

                mean se_mean   sd  25%  50%  75% n_eff Rhat
sigma           0.03    0.00 0.00 0.03 0.03 0.03  1692    1
sigma_degrees   1.53    0.00 0.02 1.51 1.53 1.54  1692    1
```


What's the probability of making a golf putt?
Probability of success
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
Distance from hole (feet)
Geometry−based model,
sigma = 1.5


Two models fit to the golf putting data
Probability of success
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
Distance from hole (feet)
Logistic regression,
a = 2.2, b = −0.3
Geometry−based model,
sigma = 1.5

“Global climate challenge”

On Dec 7, 2015, at 11:16 AM, Tom Daula <***@***.com> wrote:

Interesting applied project for your students, or as a warning for decisions under uncertainty / statistical significance. Real money on the line so the length of time and number of entries required to get a winner may be an interesting dataset after this is all done.

http://www.informath.org/Contest1000.htm

Terms of the Contest

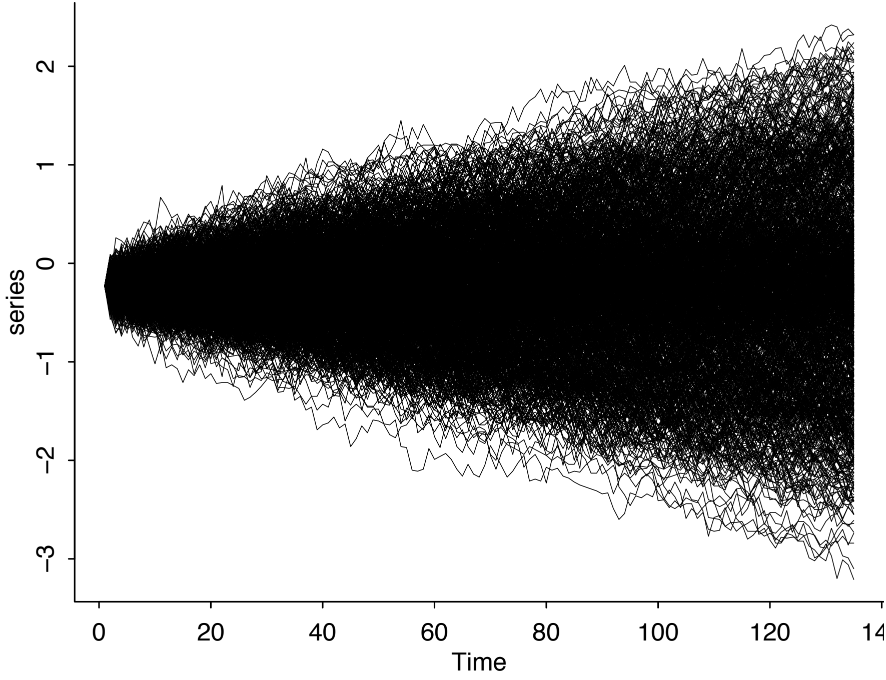
The file Series1000.txt contains 1000 simulated time series. Each series has length 135: the same length as that of the most commonly studied series of global temperatures (which span 1880–2014). The 1000 series were generated as follows. First, 1000 random series were obtained (for more details, see below). Then, some of those series were randomly selected and had a trend added to them. Each added trend was either 1°C/century or −1°C/century. For comparison, a trend of 1°C/century is greater than the trend that is claimed for global temperatures.

A prize of $100 000 (one hundred thousand U.S. dollars) will be awarded to the first person who submits an entry that correctly identifies at least 900 series: which series were generated without a trend and which were generated with a trend.

For instructions on how to submit an entry, see the Contest Entry page. Each entry must be accompanied by a payment of $10; this is being done to inhibit non-serious entries. There is a limit of one entry per person.

Download and graph the data

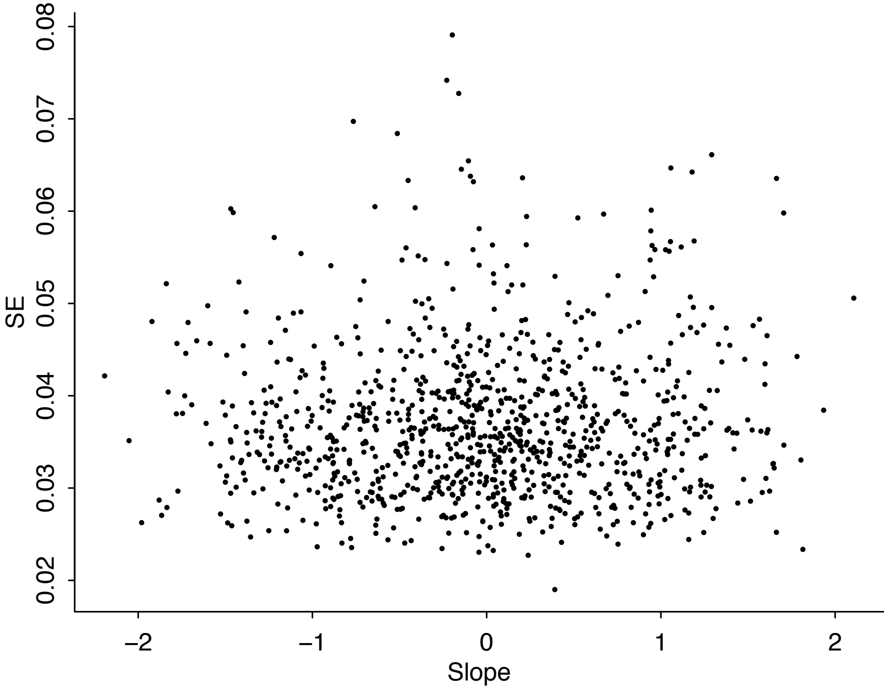
```
series <- matrix(scan("Series1000.txt"), nrow=1000, ncol=135,
  byrow=TRUE)
T <- 135
N <- 1000

pdf("series_1.pdf", height=5, width=6)
par(mar=c(3,3,2,0), tck=-.01, mgp=c(1.5,.5,0))
plot(c(1,T), range(series), bty="l", type="n",
  xlab="Time", ylab="series")
for (n in 1:N)
  lines(1:T, series[n,], lwd=.5)
dev.off()
```



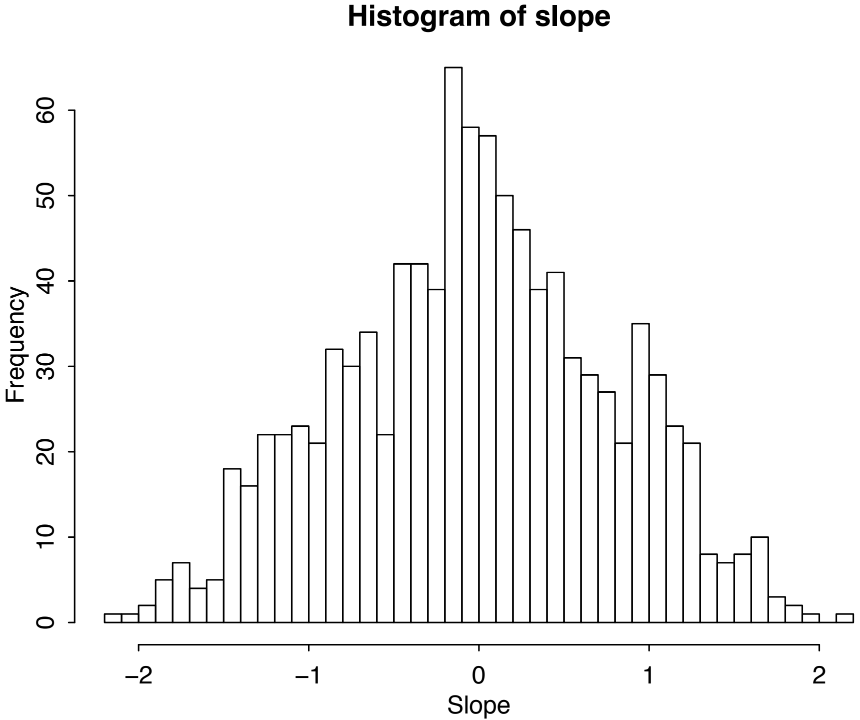
series
Time
−3
−2
−1
0
1
2
0
20
40
60
80
100
120
14

Fit a regression to each line and plot the estimated slopes

```
library("arm")
slope <- rep(NA, N)
se <- rep(NA, N)
for (n in 1:N){
  data <- series[n,]
  time <- 1:T
  fit <- lm(data ~ time)
  slope[n] <- 100*coef(fit)[2]
  se[n] <- 100*se.coef(fit)[2]
}
pdf("series_2.pdf", height=5, width=6)
par(mar=c(3,3,2,0), tck=-.01, mgp=c(1.5,.5,0))
plot(slope, se, bty="l", xlab="Slope", ylab="SE",pch=20,cex=.5)
dev.off()
```



0.08
0.07
0.06
0.05
0.04
0.03
0.02
SE
−2
−1
0
1
2
Slope


Histogram of slope
Frequency
0
10
20
30
40
50
60
−2
−1
0
1
2
Slope

Program a mixture model in Stan

```
data {
  int K;
  int N;
  real y[N];
  real mu[K];
}
parameters {
  simplex[K] theta;
  real<lower=0> sigma;
}
model {
  real ps[K];
  sigma ~ cauchy(0,2.5);
  mu ~ normal(0,10);
  for (n in 1:N) {
    for (k in 1:K)
      ps[k] = log(theta[k]) + normal_log(y[n], mu[k], sigma);
    increment_log_prob(log_sum_exp(ps));
  }
}
```

Fit the model in R

```
y <- slope
K <- 3
mu <- c(0,-1,1)
mix <- stan("mixture.stan")
print(mix)
```

```
Inference for Stan model: mixture.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

         mean se_mean   sd  25%  50%  75% n_eff Rhat
theta[1] 0.54       0 0.02 0.52 0.54 0.55  2449    1
theta[2] 0.24       0 0.02 0.23 0.24 0.25  2537    1
theta[3] 0.22       0 0.02 0.21 0.22 0.23  2444    1
sigma    0.40       0 0.02 0.39 0.40 0.42  2078    1
```

For each series, compute probability of it being in each component

```
generated quantities {
  matrix[N,K] p;
  for (n in 1:N){
    vector[K] p_raw;
    for (k in 1:K)
      p_raw[k] = theta[k]*exp(normal_log(y[n], mu[k], sigma));
    for (k in 1:K)
      p[n,k] = p_raw[k]/sum(p_raw);
  }
}
```

Results

```
      [,1] [,2] [,3]
 [1,] 0.09 0.00 0.91
 [2,] 0.41 0.59 0.00
 [3,] 0.93 0.01 0.06
 [4,] 0.83 0.17 0.00
 [5,] 0.82 0.17 0.00
 [6,] 0.95 0.01 0.05
 [7,] 0.74 0.00 0.26
 [8,] 0.86 0.14 0.00
 [9,] 0.11 0.00 0.89
[10,] 0.87 0.00 0.13
[11,] 0.94 0.01 0.06
[12,] 0.29 0.71 0.00
[13,] 0.09 0.91 0.00
[14,] 0.67 0.33 0.00
[15,] 0.93 0.01 0.06
[16,] 0.95 0.01 0.04
[17,] 0.16 0.84 0.00
[18,] 0.95 0.04 0.01
[19,] 0.77 0.23 0.00
```

Summaries

- Best guess for each series:

```
  1   2   3
559 232 209
```

- Expected # correct and sd:

```
854.1  10.3
```

- Probability of getting at least 900 correct:

```
> pnorm(expected_correct, 899.5, sd_correct)
[1] 5.421277e-06
```

- Ummmmm ...

```
> 1/pnorm(expected_correct, 899.5, sd_correct)
[1] 184458.4
```

Should I play the \$100,000 challenge?

- From the discussion thread:

Richard Tol (@RichardTol) *says:*

November 20, 2015 at 8:31 pm

Why don't you guys just pay £10 to win £100,000? You don't need to accept that the challenge has any bearing on climate change — it has not — but it is a great opportunity to make £99,990.

- Expected return on \$10 bet:

$$(5.4 * 10^{-6}) * 10^{5} = \$0.54$$

- What would *you* do?