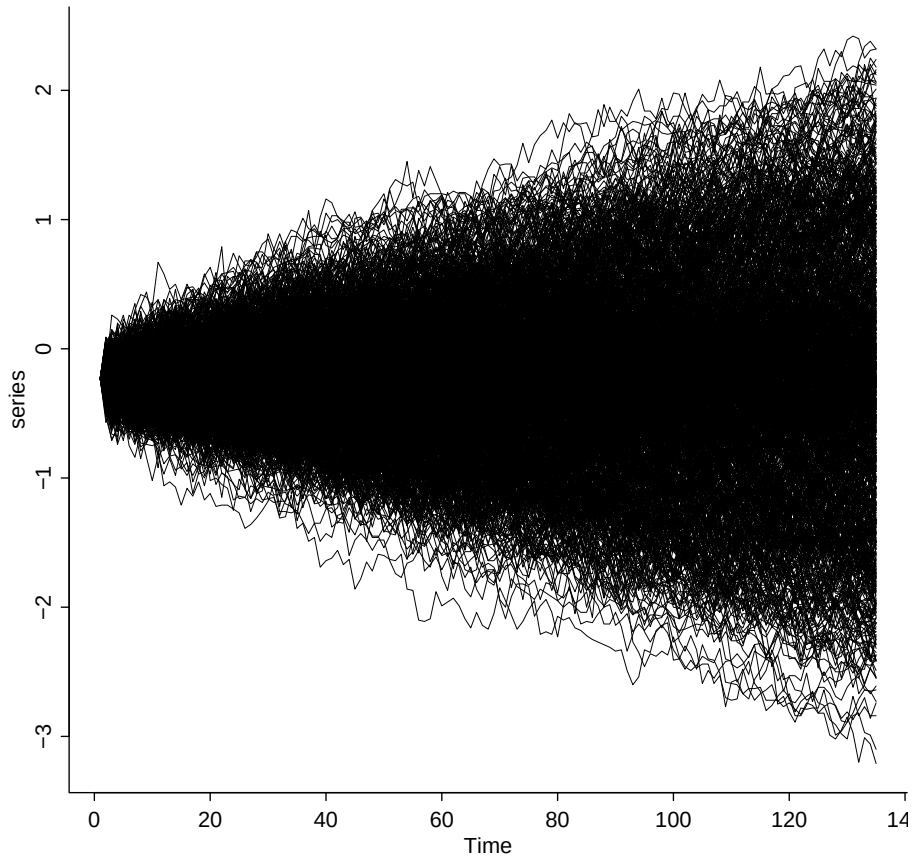
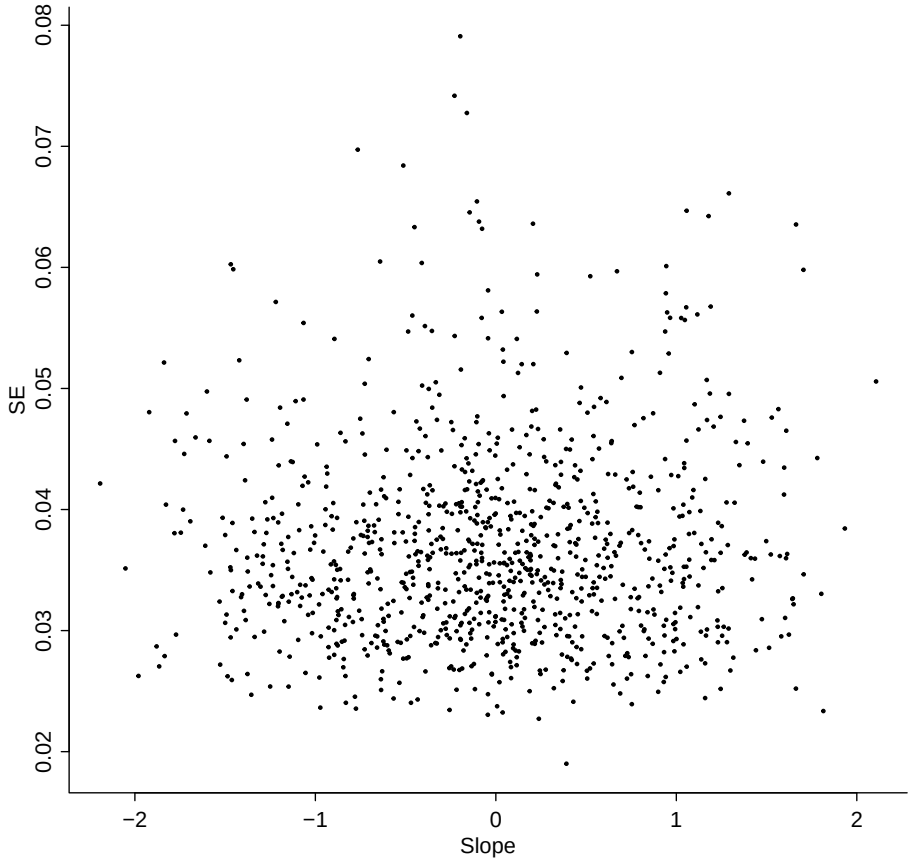
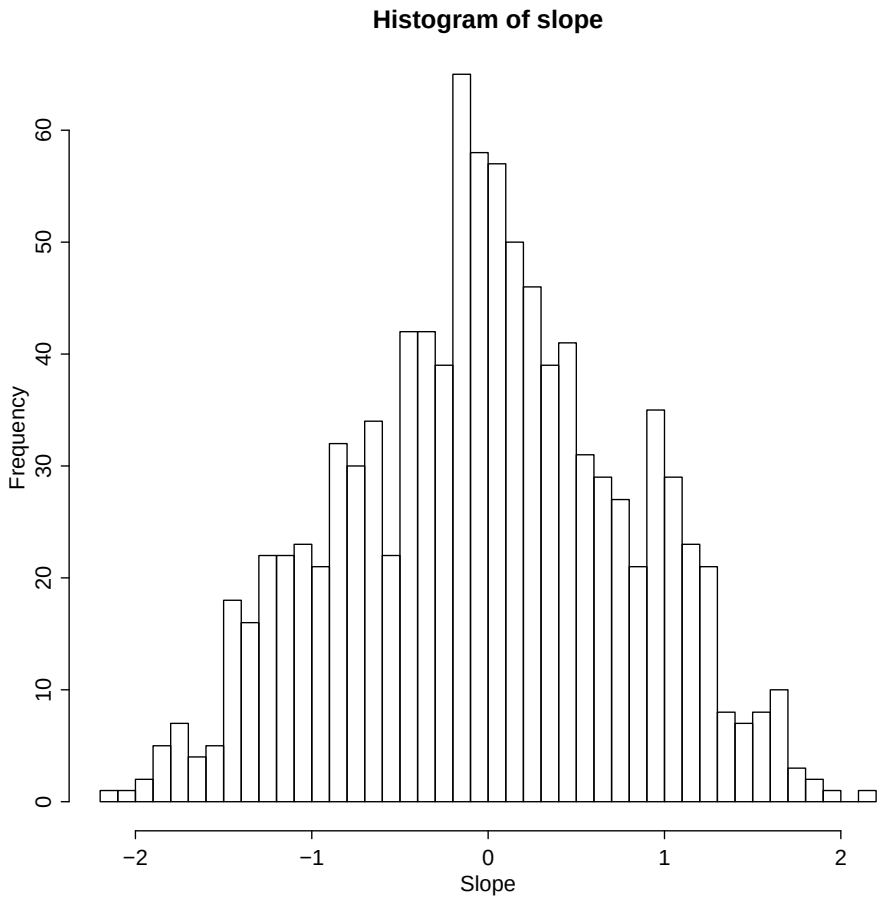
```
library(shinystan)

## Loading required package:  shiny
##
## This is shinystan version 2.0.1

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()

library("arm")

## Loading required package:  MASS
## Loading required package:  Matrix
## Loading required package:  lme4
##
## arm (Version 1.8-6, built:  2015-7-7)
##
## Working directory is /home/gquinn/Desktop/html/stonehill/math/stan/examples/series1000/s

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()

#pdf("series_3.pdf", height=5, width=6)
par(mar=c(3,3,2,0), tck=-.01, mgp=c(1.5,.5,0))
hist(slope, xlab="Slope", breaks=seq(floor(10*min(slope)), ceiling(10*max(slope)))/10)
```




```
#dev.off()

library("rstan")

## Loading required package:  Rcpp
## Loading required package:  ggplot2
## rstan (Version 2.8.0, packaged:  2015-09-19 14:48:38 UTC, GitRev:
05c3d0058b6a)
## For execution on a local, multicore CPU with excess RAM we recommend
calling
## rstan_options(auto_write = TRUE)
## options(mc.cores = parallel::detectCores())
##
## Attaching package:  'rstan'
##
## The following object is masked from 'package:arm':
```

```
##
##    traceplot
```

```
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

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

```
launch_shinystan(mix)
```

```
##
## Loading...
## Note:  for large models ShinyStan may take a few moments to launch.
##
## Listening on http://127.0.0.1:6539
```

```
prob_sims <- extract(mix, "p")$p
prob <- array(NA, c(N,K))
for (n in 1:N){
  for (k in 1:K){
    prob[n,k] <- mean(prob_sims[,n,k])
  }
}
max_prob <- apply(prob, 1, max)
choice <- apply(prob, 1, which.max)
expected_correct <- sum(max_prob)
sd_correct <- sqrt(sum(max_prob*(1-max_prob)))
print(table(choice))
```

```
## choice
##   1   2   3
## 559 232 209

pfround(c(expected_correct, sd_correct), 1)

## [1] 854.3  10.3

pnorm(expected_correct, 900, sd_correct)

## [1] 4.638375e-06

1/pnorm(expected_correct, 900, sd_correct)

## [1] 215592.8

1/pnorm(expected_correct, 899.5, sd_correct)

## [1] 172335.6

theta_true <- c(.5, .25, .25)
sigma_true <- .4
prob_simple <- array(NA, c(N,K))
for (n in 1:N){
  prob_0 <- theta_true*dnorm(y[n], mu, sigma_true)
  prob_simple[n,] <- prob_0/sum(prob_0)
}
pfround(cbind(prob, prob_simple)[1:10,], 2)

##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,] 0.08 0.00 0.92 0.06 0.00 0.94
##  [2,] 0.40 0.60 0.00 0.37 0.63 0.00
##  [3,] 0.93 0.01 0.06 0.92 0.01 0.07
##  [4,] 0.83 0.17 0.00 0.81 0.19 0.00
##  [5,] 0.82 0.17 0.00 0.81 0.19 0.00
##  [6,] 0.95 0.01 0.05 0.94 0.01 0.05
##  [7,] 0.74 0.00 0.26 0.70 0.00 0.30
##  [8,] 0.86 0.14 0.00 0.85 0.15 0.00
##  [9,] 0.11 0.00 0.89 0.08 0.00 0.92
## [10,] 0.87 0.00 0.13 0.85 0.00 0.15

max_prob_simple <- apply(prob_simple, 1, max)
choice_simple <- apply(prob_simple, 1, which.max)
expected_correct_simple <- sum(max_prob_simple)
sd_correct_simple <- sqrt(sum(max_prob_simple*(1 - max_prob_simple)))
print(table(choice_simple))
```

```
## choice_simple
##   1   2   3
## 540 239 221

pfround(c(expected_correct_simple, sd_correct_simple), 1)

## [1] 855.3  10.3

1/pnorm(expected_correct_simple, 899.5, sd_correct_simple)

## [1] 120744.4

prob_simple_binary <- cbind(prob_simple[,1], prob_simple[,2] + prob_simple[,3])

max_prob_simple_binary <- apply(prob_simple_binary, 1, max)
choice_simple_binary <- apply(prob_simple_binary, 1, which.max)
expected_correct_simple_binary <- sum(max_prob_simple_binary)
sd_correct_simple_binary <- sqrt(sum(max_prob_simple_binary*(1 - max_prob_simple_binary)))
print(table(choice_simple_binary))

## choice_simple_binary
##   1   2
## 540 460

pfround(c(expected_correct_simple_binary, sd_correct_simple_binary), 1)

## [1] 855.3  10.3
```