```
rainf<-function(site,dm,dr){
  rainsel<-paste("rain_",site,sep=’’)
  print(rainsel)
  raincol<-match(rainsel,names(rainfall))
  print(raincol)
  tempdf<-rainfall[!is.na(rainfall[[raincol]]),]
  raincol<-match(rainsel,names(tempdf))
  measurable_rain<-as.numeric(tempdf[[raincol]]>0)
  sitename<-rep(site,nrow(tempdf))
  dseq<-(as.numeric(tempdf$date)-dm)/(dr/4)
  rdf<-data.frame(sitename,measurable_rain,dseq)
  rm(tempdf)
  return(rdf)
}
```

```
library(lubridate)

##
## Attaching package:  ’lubridate’
## The following object is masked from ’package:base’:
##
##     date

load("marain.Rdata")
str(rainfall)

## ’data.frame': 44529 obs. of  13 variables:
##  $ date         : Date, format: "1893-01-02" "1893-01-03" ...
##  $ rain_amherst : num  215 0 0 0 4 0 0 NaN 6 NaN ...
##  $ rain_bedford : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ rain_bluehill: num  22 73 0 0 0 0 0 0 0 0 ...
##  $ rain_gb      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ rain_lawrence: num  76 0 0 0 31 0 0 0 42 0 ...
##  $ rain_nb      : num  NaN 40 NaN NaN NaN NaN 6 NaN NaN NaN ...
##  $ rain_pk      : num  117 0 0 0 45 0 0 0 38 0 ...
##  $ rain_ptown   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ rain_reading : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ rain_taunton : num  123 0 0 0 143 0 0 0 24 0 ...
##  $ rain_walpole : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ rain_wm      : num  NA NA NA NA NA NA NA NA NA NA ...

seqdate<-as.numeric(rainfall$date)
str(seqdate)

##  num [1:44529] -28122 -28121 -28120 -28119 -28118 ...
```

```
date_range<-max(seqdate)-min(seqdate)
date_mean=mean(seqdate)

ldf<-rainf("amherst",date_mean,date_range)  #start with amherst

## [1] "rain_amherst"
## [1] 2

df<-rainf("bluehill",date_mean,date_range)  #add bluehill

## [1] "rain_bluehill"
## [1] 4

ldf<-rbind(ldf,df)

df<-rainf("taunton",date_mean,date_range)  #add taunton

## [1] "rain_taunton"
## [1] 11

ldf<-rbind(ldf,df)

df<-rainf("pk",date_mean,date_range)  #add pk

## [1] "rain_pk"
## [1] 8

ldf<-rbind(ldf,df)

df<-rainf("lawrence",date_mean,date_range)  #add lawrence

## [1] "rain_lawrence"
## [1] 6

ldf<-rbind(ldf,df)

df<-rainf("bedford",date_mean,date_range)  #add bedford

## [1] "rain_bedford"
## [1] 3

ldf<-rbind(ldf,df)

df<-rainf("gb",date_mean,date_range)  #add gb

## [1] "rain_gb"
## [1] 5
```

```
ldf<-rbind(ldf,df)

df<-rainf("ptown",date_mean,date_range)  #add ptown
```

```
## [1] "rain_ptown"
## [1] 9
```

```
ldf<-rbind(ldf,df)

df<-rainf("nb",date_mean,date_range)  #add nb
```

```
## [1] "rain_nb"
## [1] 7
```

```
ldf<-rbind(ldf,df)

df<-rainf("reading",date_mean,date_range)  #add reading
```

```
## [1] "rain_reading"
## [1] 10
```

```
ldf<-rbind(ldf,df)

df<-rainf("walpole",date_mean,date_range)  #add walpole
```

```
## [1] "rain_walpole"
## [1] 12
```

```
ldf<-rbind(ldf,df)

df<-rainf("wm",date_mean,date_range)  #add wm
```

```
## [1] "rain_wm"
## [1] 13
```

```
ldf<-rbind(ldf,df)

str(ldf)
```

```
## 'data.frame': 338824 obs. of  3 variables:
##  $ sitename       : Factor w/ 12 levels "amherst","bluehill",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ measurable_rain: num  1 0 0 0 1 0 0 1 0 1 ...
##  $ dseq           : num  -2 -2 -2 -2 -2 ...
```

```
measurable_rain<-ldf$measurable_rain
dseq<-ldf$dseq
site<-as.numeric(ldf$sitename)
Nsites<-length(levels(ldf$sitename))
N<-nrow(ldf)
st<-table(site)
str(st)
```

```
##  'table' int [1:12(1d)] 42803 43998 40182 42817 41848 18839 10904 10462 34297 19920 ...
##  - attr(*, "dimnames")=List of 1
##   ..$ site: chr [1:12] "1" "2" "3" "4" ...

levels(ldf$sitename)

##  [1] "amherst"  "bluehill" "taunton"  "pk"       "lawrence" "bedford"
##  [7] "gb"       "ptown"    "nb"       "reading"  "walpole"  "wm"

s<-st[1:length(st)]
```

Call STAN

```
library(rstan)

## Loading required package:  ggplot2
## rstan (Version 2.9.0-3, packaged:  2016-02-11 15:54:41 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())

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

stanfit<-stan("logit_frequency.stan") #call STAN
print(stanfit)                        #print a summary of the results

## Inference for Stan model: logit_frequency.
## 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       2.5%        25%        50%
## beta0[1]         -0.68    0.00 0.01      -0.70      -0.68      -0.68
## beta0[2]         -0.53    0.00 0.01      -0.55      -0.53      -0.53
## beta0[3]         -0.85    0.00 0.01      -0.87      -0.86      -0.85
## beta0[4]         -0.83    0.00 0.01      -0.85      -0.84      -0.83
## beta0[5]         -0.93    0.00 0.01      -0.95      -0.94      -0.93
## beta0[6]         -0.80    0.00 0.03      -0.86      -0.82      -0.80
## beta0[7]         -0.58    0.00 0.07      -0.71      -0.62      -0.58
## beta0[8]         -0.78    0.00 0.05      -0.88      -0.81      -0.78
## beta0[9]         -0.67    0.00 0.01      -0.69      -0.68      -0.67
## beta0[10]        -0.62    0.00 0.03      -0.69      -0.65      -0.62
## beta0[11]        -0.92    0.00 0.06      -1.03      -0.96      -0.92
## beta0[12]        -1.18    0.00 0.04      -1.26      -1.21      -1.18
## beta1[1]         -0.02    0.00 0.01      -0.04      -0.02      -0.02
## beta1[2]        -0.02    0.00 0.01      -0.04      -0.03      -0.02
## beta1[3]         0.13    0.00 0.01       0.11       0.13       0.13
## beta1[4]         0.19    0.00 0.01       0.18       0.19       0.19
## beta1[5]         0.07    0.00 0.01       0.05       0.06       0.07
## beta1[6]         0.23    0.00 0.03       0.17       0.21       0.23
## beta1[7]         0.06    0.00 0.06      -0.05       0.02       0.06
## beta1[8]        -0.26    0.00 0.07      -0.39      -0.31      -0.26
## beta1[9]        -0.05    0.00 0.01      -0.07      -0.06      -0.05
## beta1[10]        0.08    0.00 0.03       0.02       0.06       0.08
## beta1[11]        0.21    0.00 0.04       0.12       0.18       0.21
## beta1[12]        0.27    0.00 0.04       0.20       0.25       0.27
## lp__      -212748.55    0.08 3.42 -212756.16 -212750.70 -212748.20
##                  75%      97.5% n_eff Rhat
## beta0[1]        -0.67      -0.66  4000    1
## beta0[2]        -0.52      -0.51  4000    1
## beta0[3]        -0.84      -0.83  4000    1
## beta0[4]        -0.83      -0.81  4000    1
## beta0[5]        -0.92      -0.91  4000    1
## beta0[6]        -0.77      -0.73  2756    1
## beta0[7]        -0.53      -0.44  2594    1
## beta0[8]        -0.74      -0.68  2653    1
## beta0[9]        -0.66      -0.65  4000    1
## beta0[10]       -0.60      -0.56  2877    1
## beta0[11]       -0.88      -0.80  2467    1
## beta0[12]       -1.16      -1.11  2888    1
## beta1[1]        -0.01       0.00  4000    1
## beta1[2]        -0.01       0.00  4000    1
## beta1[3]         0.14       0.15  4000    1
## beta1[4]         0.20       0.21  4000    1
## beta1[5]         0.08       0.09  4000    1
## beta1[6]         0.25       0.29  2744    1
## beta1[7]         0.10       0.16  2558    1
## beta1[8]        -0.22      -0.12  2610    1
## beta1[9]        -0.04      -0.03  4000    1
## beta1[10]        0.10       0.14  2816    1
## beta1[11]        0.24       0.29  2541    1
## beta1[12]        0.30       0.35  2924    1
## lp__      -212746.07 -212742.82  1693    1
##
## Samples were drawn using NUTS(diag_e) at Wed Jun  8 01:34:24 2016.
## 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).
```

```
save(stanfit,file="logit_frequency.Rdata")
print(get_stanmodel(stanfit))    #print the model file
```

```
## S4 class stanmodel 'logit_frequency' coded as follows:
## //logistic regression for probability of measurable rain
## //
## data {
##   int<lower=1> N;                            //number of observations
##   int<lower=1> Nsites;                       //number of sites
##   int<lower=0,upper=1> measurable_rain[N];   //measurable rain or not
##   int<lower=1,upper=Nsites> site[N];         //site
##   vector[N] dseq;                            //day index
##   int s[Nsites];                             //number of obs for each site
## }
## parameters {
##   real beta0[Nsites];                        //logistic intercept
##   real beta1[Nsites];                        //logistic slope
## }
## model {
##   int pos;
##   beta0 ~ normal(0,5);
##   beta1 ~ normal(0,1);
##
##   pos<-1;
##   for(i in 1:Nsites){
##      segment(measurable_rain, pos, s[i]) ~ bernoulli_logit(beta0[i]+beta1[i]*segment(dseq
##      pos<-pos+s[i];
##   }
## }
```

Launch shinystan

```
library(shinystan)               #downloaded from cran

## Loading required package:  shiny
##
## This is shinystan version 2.1.0

launch_shinystan(stanfit)

##
## Loading...
## Note:  for large models ShinyStan may take a few moments to launch.
##
## Listening on http://127.0.0.1:5652
## Warning in file.copy(extPath(swf), dest, overwrite = TRUE): problem
copying /usr/lib64/R/library/DT/htmlwidgets/lib/datatables-extensions/copy_csv_xls_pdf.swf
to www/copy_csv_xls_pdf.swf:  Permission denied
```

```
## Warning in file.copy(extPath(swf), dest, overwrite = TRUE): problem
copying /usr/lib64/R/library/DT/htmlwidgets/lib/datatables-extensions/copy_csv_xls_pdf.swf
to www/copy_csv_xls_pdf.swf:  Permission denied
## Warning in file.copy(extPath(swf), dest, overwrite = TRUE): problem
copying /usr/lib64/R/library/DT/htmlwidgets/lib/datatables-extensions/copy_csv_xls_pdf.swf
to www/copy_csv_xls_pdf.swf:  Permission denied
## Warning in file.copy(extPath(swf), dest, overwrite = TRUE): problem
copying /usr/lib64/R/library/DT/htmlwidgets/lib/datatables-extensions/copy_csv_xls_pdf.swf
to www/copy_csv_xls_pdf.swf:  Permission denied
```