1 Introduction & Model Specification

The goal of this analysis is to asses if and where the distribution of extreme streamflow has changed over the past 72 years. Let Y_{st} be the observation at location $s \in \{1, \ldots, 702\}$ and time $t \in \{1, \ldots, 72\}$. We define $X_t = (year_t - 1985)/10$ to be the linear trend covariate and restrict our analysis sites without missing values, $n = 236$. The proposed hierarchical model is as follows,

$$
\begin{aligned}
Y_{st}|\theta_s &\overset{\text{ind}}{\sim} \text{GEV}(\mu_{st}, \sigma_s, \xi_s) & \mu_{st} &= \theta_{1s} + \theta_{2s}X_t, \quad \theta_{3s} = \log(\sigma_s), \quad \theta_{4s} = \xi_s \\
\theta_s|\mu, \Sigma &\overset{\text{ind}}{\sim} N_4(\mu, \Sigma) & \mu &\sim N(0, \Omega), \quad \Sigma \sim \text{InvWish}(\nu, R)
\end{aligned}
$$

The prior distributions for μ and Σ are chosen for conjugacy. In this way the draws from the full conditional distributions of μ and Σ fall under Gibbs sampling. Their hyper-parameters are set such that the priors are uninformative, $\Omega = \text{diag}(\text{rep}(10000, 4)^2)$, $\nu = 3.001$, and $R = I_4$.

We "prime" this model with a simpler model described in the ST740 course code file "Extreme value analysis using Metropolis sampling." The simple model is identical, but instead replaces the prior distributions for μ and Σ with fixed values, $\mu = (0, 0, 0, 0)^T$ and $\Sigma = \text{diag}(c(10000, 10000, 10, 0.5)^2)$ effectively eliminating the Gibbs sampling steps.

The sampler was not found to be sensitive to the choice of hyper-parameters as long as they respected the notion of "uninformativeness." So long as Ω, the prior covariance matrix for μ, was initialized with a diagonal of sufficiently large magnitude, the prior remained uninformative. Likewise for the Inverse Wishart prior. So long as ν, the degrees of freedom parameter was small, in our case $\nu = 4 - 1 + \epsilon$, changes in the magnitude of the diagonal R matrix were observed to have minimal effect.

2 MCMC Algorithm Description & Convergence Diagnostics

2.1 MCMC Description

We implemented a hybrid sampling approach (Gibbs for μ and Σ, Metropolis for θ_{ps}) to draw from the joint posterior of the parameters. In the Metropolis sampling phase, drawing from the full conditional of each element of θ_s requires a symmetric candidate distribution, $q(\cdot)$, in our case chosen to be $\theta^c_{ps} \sim N\left(\theta^*_{ps}, \tau^2_{ps}\right)$, where θ^*_{ps} indicates the current value for the pth element of $\boldsymbol{\theta}$ at location s. The candidate

standard deviations, τ_{ps}, are tuned individually for all combinations of p and s. In the Metropolis update for θ_{ps} we draw a value from the candidate distribution, evaluate the ratio of the posterior distribution at the candidate and current value, and flip an $\min(1, R)$-probability coin to accept or reject the candidate value. The Gibbs steps follow the typical updates following the conjugacy rules for Normal-InvWishart priors.

Selecting initial values for θ_s and τ_{ps} proved particularly challenging. Poor choices would often result in an initial candidate distribution that failed to propose any sensible values a marred the MCMC in its tracks. This challenge is overcome using the simple model on a site-by-site basis to identify suitable initial values. The simple model is initialized using $\theta_s^{(0)} = \widehat{\theta}_{s,MLE}$ and $\tau_s = SE(\widehat{\theta}_{s,MLE})$ and run for a suitable number of iterations to achieve convergence (typically $< 20,000$). The posterior means and candidate std devs from the simple model are then transferred as initial values in the complex model. In this sense the complex model is "primed" for faster convergence. Additional tuning is applied in the complex model through a 5,000 iteration burn-in phase. The acceptance rate is assessed every 50 iterations and the candidate std devs are scaled 1.2x or 0.8x to decrease or increase acceptance, respectively, until an acceptance rate of 0.3-0.5 is achieved. This approach achieved suitable convergence in 234 of the 236 sites.

2.2 Convergence Diagnostics

The MCMC algorithm was executed with 50,000 iterations and an initial burn-in of 5,000 iterations (10%). Three diagnostics were used to assess the covergence of the chains. They are the effective sample size, lag 1 autocorrelation, and the Geweke convergence test. The sampler is sensitive to initial values so we did not run parallel chains or apply the Gelman diagnostic. Ultimately, the burn-in was extended to 20,000 iterations in order to produce chains with suitable convergence behavior. The diagnostics for μ and Σ parameters are presented in Table 1.

Convergence diagnostics for the θ_s parameter vectors are presented in boxplots. Figure 1 displays a panel of four plots each of which corresponds to a different convergence diagnostic. Each boxplot corresponds to the collection of convergence diagnostic values for an element of θ_s across all spatial locations. Most all convergence diagnostics are satisfactory, though there are cases where the effective sample size or the Geweke diagnostic statistics my be considered "questionable". The Geweke diagnostic

Table 1: Convergence diagnostics for μ and Σ.

Parameter	μ_1	μ_2	μ_3	μ_4
Eff Sample (x 10^3)	30	13.6	30	9.4
Lag 1 Autocorr	0.01	0.11	0.01	0.19
Geweke diagnostic	0.02	-0.93	1.09	0.15

Parameter	σ_{11}	σ_{12}	σ_{13}	σ_{14}	σ_{22}	σ_{23}	σ_{24}	σ_{33}	σ_{34}	σ_{44}
Eff Sample (x 10^3)	23.9	9.9	26.9	13.5	13.3	11.4	17.6	19.3	5.8	3.6
Lag 1 Autocorr	0.02	0.16	0.01	0.11	0.15	0.13	0.09	0.02	0.21	0.40
Geweke diagnostic	1.23	-0.17	0.80	-0.33	0.01	-0.68	2.03	-0.41	0.71	-2.80

is essentially a Z-score and there are many sites so we should expect some tests stats, Z, to be such that $2 < |Z| < 3$. The autocorrelations are generally high, but the chain is long enough to produce a suitable effective sample size in almost all cases. Lastly, the sampling acceptance rates are all well within the desired range.

3 Results & Analysis

The θ_{2s} parameter is associated with a decadal effect on the mean extreme stream flow. The posterior of this parameter will help us understand if and how the mean extreme stream flow changes over time. In Figure 2 we present histograms for the θ_{2s} posterior means and std devs. Many posterior means are near 0 and most std devs are less than 200. There are a small number of sites with substantial negative effect size.

Figure 2 is a panel of 4 maps, each of which display information regarding the posterior for θ_{2s}. The top left and right maps are same the posterior means and std devs from the histograms. The bottom left map depicts the posterior probability that θ_{2s} is greater than 0. The bottom right map highlights locations where the 95% credible interval, centered on the posterior mean, indicate a significant decadal effect (i.e. the credible interval does not include 0).

The Oregon coast and the border between Idaho and Montana are found to have a significant negative decadal effect on mean stream flow. A similar effect is observed for 5 locations peppered across the

Figure 1: Convergence diagnostics for θ_{2s}.

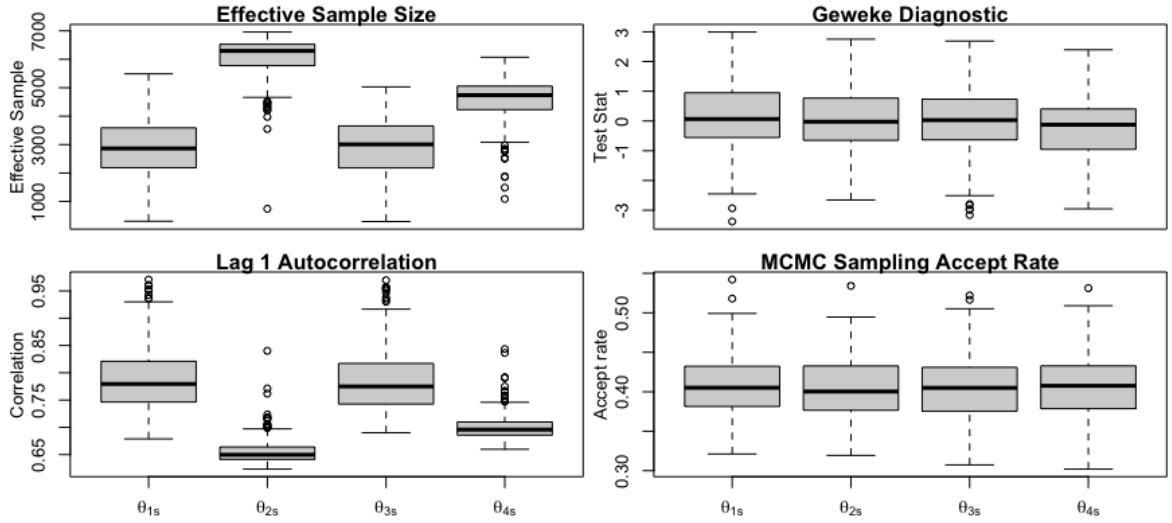


Figure 2: Posterior summary for θ_{2s}.

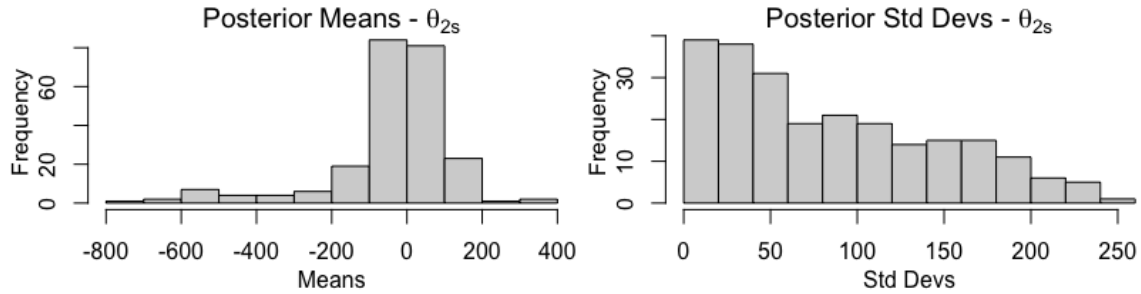
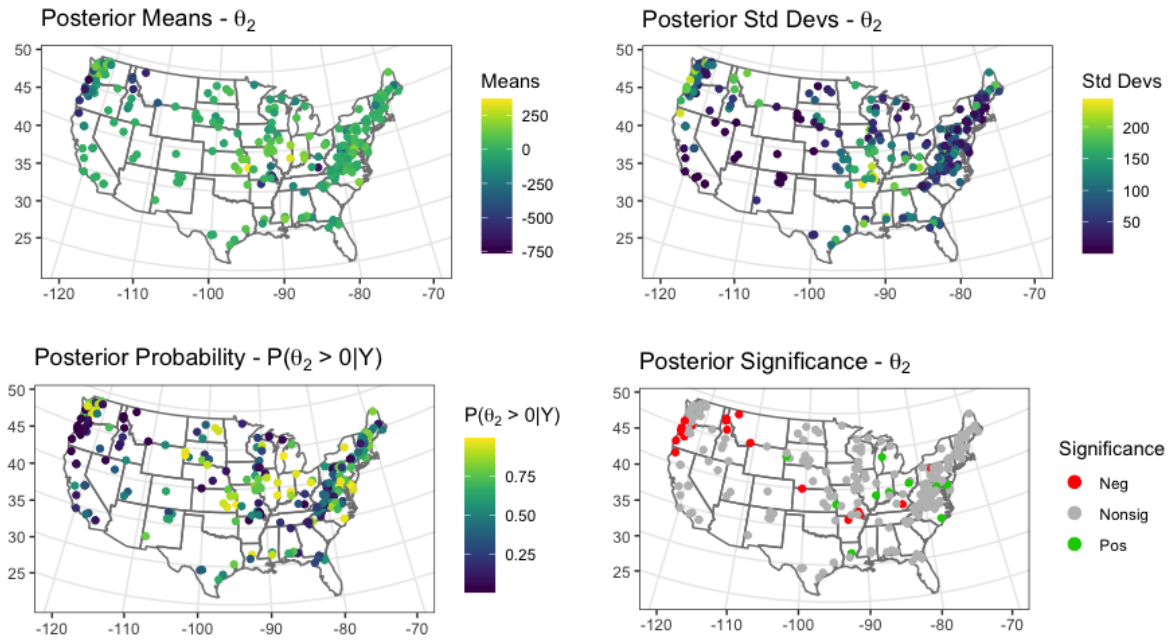


Figure 3: Spatial results for θ_{2s}



midwest, though there are also several locations in the midwest that were found to have a positive decadal effect.

4 Limitations & Further Research

There are several limitations of the analysis and potential areas for further research. These include, but are not limited to, discarded data for nearly two thirds of the sites due to missing values, assumed spatial independence, choice of naive symmetric candidate distribution for the MCMC sampling, and an ad hoc tuning routine that is not successful in all cases. It may also be noteworthy to revisit the distributional assumption. The support of the GEV distribution depends on the parameter values and can on occasion include negative values, a behaviour that does not match the physical fact that the yearly extreme stream flow must be non-negative. It may be prudent to fit the model with a distributional family that has matching support. The discarded data and spatial independence require substantial additional tools to incorporate into the analysis. Tackling issues with the candidate distribution or the tuning routine may be more approachable.

DA1 - Code Turnin

Matthew Shisler

2022-10-17

Packages

```
library(MCMCpack)
library(ggplot2)
library(viridis)
library(mapproj)
library(evd)
library(coda)
library(profvis)
```

Load data and process.

```
file <- "https://www4.stat.ncsu.edu/~bjreich/ST740/HCDN_annual_max.RData"
load(url(file))
sites <- s
sites <- sites[which(complete.cases(Y)),]

Xt <- (year - 1985)/10
X  <- matrix(c(rep(1,nyears), Xt), nrow = nyears, ncol = 2)
Yc  <- Y[complete.cases(Y),]
sites2 <- sites[complete.cases(Y),]

nsites <- nrow(Yc)
ntheta <- 4
```

Log posterior function

```
log_post <- function(theta,y,x,pri_mn,pri_prec){
   sum(dgev(y,theta[1]+x*theta[2],exp(theta[3]),theta[4],log=TRUE)) -
   0.5*t(theta-pri_mn)%*%pri_prec%*%(theta-pri_mn)
}
```

Function for the simplified model (from course code)

```
Bayes_GEV <- function(y,x,
                      pri_mn  = rep(0,4),
                      pri_cov = 1000*diag(4),
                      iters = 10000, burn = 1000){
 library(evd)
 tik <- proc.time()

 # Set initial values using MLE

  mle      <- fgev(y,nsloc=x,warn.inf=FALSE)
  theta    <- mle$estimate
  theta[3] <- log(theta[3])
  init     <- theta
  p        <- length(theta)
  pri_prec <- solve(pri_cov)
  curlp    <- log_post(theta,y,x,pri_mn,pri_prec)


 # Store the samples
  keepers     <- matrix(0,iters,p)
  keepers[1,] <- theta
  colnames(keepers) <- c("Intercept","Slope","Log scale","Shape")
  canlp_save    <- matrix(0,iters,p)

 # Set up Metropolis steps

  att   <- acc <- rep(0,p)
  MH    <- mle$std.err# Standard errors as initial guesses
  MH[3] <- MH[3]/mle$estimate[3]  # Delta method for log(scale)
  init_MH <- MH

 for(iter in 2:iters){

   # Update the GEV parameters

    for(j in 1:p){
      att[j] <- att[j] + 1
      can    <- theta
      can[j] <- rnorm(1,theta[j],MH[j])
      canlp  <- log_post(can,y,x,pri_mn,pri_prec)
      canlp_save[iter, j] <- canlp
      if(!is.na(canlp)){if(log(runif(1))<canlp-curlp){
        acc[j] <- acc[j] + 1
        theta  <- can
        curlp  <- canlp
      }}
    }

   # Tune the Metropolis algorithm

    if(iter<burn){for(j in 1:length(att)){if(att[j]>50){
      if(acc[j]/att[j] < 0.3){MH[j] <- MH[j]*0.8}
```

```
      if(acc[j]/att[j] > 0.5){MH[j] <- MH[j]*1.2}
      acc[j] <- att[j] <- 0
    }}}

   # Store the output

    keepers[iter,] <- theta
 }

 tok <- proc.time()
 out <- list(theta=keepers,acc_rate=acc/att,time=tok-tik, canlp = canlp_save, init = init, init_
MH = init_MH, final_MH = MH)

return(out)}
```

Run the simplified model to prime the complex model. Sites with singular observed info matrices are initialized on a case-by-case basis. (code ommitted)

```
# store initial values and candidate std devs for complex model
stage2_init     <- matrix(0, nrow=nsites, ncol=4)
stage2_MH       <- matrix(0, nrow=nsites, ncol=4)
stage1_acc_rate <- matrix(0, nrow=nsites, ncol=4)

burn     <- 5000
iters    <- 10000
pri_mn   <- rep(0,4)
pri_cov  <- diag(c(10000,10000,10,0.5)^2)

for (site_iter in 1:nsites){
  y <- Yc[site_iter,]
  fit      <- Bayes_GEV(y,x,pri_mn=pri_mn,pri_cov=pri_cov,
                        burn=burn,iters=iters)

  par(mfrow=c(2,2))
  for(j in 1:4){
    plot(fit$theta[,j],type="l",
         xlab="Iteration",ylab=colnames(fit$theta)[j])
    abline(fit$theta[1,j],0,col=2) # MLE/inital value in red
  }

  stage2_init[site_iter, ] <- colMeans(fit$theta[burn:iters,])
  stage2_MH[site_iter, ] <- fit$final_MH
  stage1_acc_rate[site_iter,] <- fit$acc_rate
}
```

Set-up the complex model.

```
# For mu
delta     <- rep(0,4)
omega_inv <- solve(diag(c(10000,10000,10000,10000)^2)) # may need to adjust this

# For capsigma
nu <- ntheta - 1 + 0.001
R  <- diag(ntheta)


MH <- MH_init
theta <- theta_init

mu <- rep(0,ntheta)
capsigma     <- diag(c(10000,10000,10,0.5)^2)
capsigma_inv <- solve(capsigma)

# compute the initial (current) log posterior values
curlp <- rep(0, nsites)
for (s in 1:nsites){
  curlp[s]  <- log_post(theta[s,], Yc[s,], Xt, pri_mn = mu, pri_prec = capsigma_inv)
}
```

More Metro set-up set-up.

```
iters <- 50000
burn  <- 5000
att <- acc <- matrix(0, nrow = nsites, ncol = ntheta)

# storage
keep_theta    <- array(0, dim = c(iters, ntheta, nsites))
keep_mu       <- matrix(0, nrow = iters, ncol = ntheta)
keep_capsigma <- array(0, dim = c(ntheta, ntheta, iters))
```

Metropolis loop for the complex model.

```
tik <- proc.time()
pb <- txtProgressBar(min=0, max=iters, initial=0, style=3)
num_canlp_na <- 0
for (iter in 1:iters){

  setTxtProgressBar(pb, iter)
  # capsigma_inv <- solve(capsigma)
  capsigma_inv <- chol2inv(chol(capsigma))

  # update theta for each site
  for(s in 1:nsites){
    for(j in 1:ntheta){
      att[s,j] <- att[s,j] + 1
      can      <- theta[s,]
      can[j]   <- rnorm(1,theta[s,j],MH[s,j])
      canlp    <- log_post(can, Yc[s,], Xt, pri_mn = mu, pri_prec = capsigma_inv)

      if(!is.na(canlp)){
        if(log(runif(1))<canlp-curlp[s]){
          acc[s,j]  <- acc[s,j] + 1
          theta[s,] <- can
          curlp[s]  <- canlp
        }
      } else {
        num_canlp_na <- num_canlp_na + 1
        #print(paste("canlp NA", s,j))
      }
    }
  }

  # update mu
  M     <- capsigma_inv%*%colSums(theta)
  inv_V <- solve(nsites*capsigma_inv + omega_inv)
  #inv_V <- chol2inv(chol(nsites*capsigma_inv + omega_inv))
  mu    <- as.vector(inv_V%*%M+t(chol(inv_V))%*%rnorm(length(M)))

  # update capsigma
  temp <- sweep(theta,2,mu)
  W <- t(temp)%*%temp
  capsigma <- MCMCpack::riwish(nsites + nu, W + R)

  # tune
  if(iter<burn){
    for(s in 1:nsites){
      for(j in 1:ntheta){
        if(att[s,j]>50){
          if(acc[s,j]/att[s,j] < 0.3){MH[s,j] <- MH[s,j]*0.8}
          if(acc[s,j]/att[s,j] > 0.5){MH[s,j] <- MH[s,j]*1.2}
          acc[s,j] <- att[s,j] <- 0
        }
      }
    }
```

```
  }


  # store the current iteration
  keep_theta[iter,,]    <- t(theta)
  keep_mu[iter,]        <- mu
  keep_capsigma[,,iter] <- capsigma

}
tok <- proc.time()
(tok-tik)/60
```

Investigate some convergence diagnostics.

```
burn <- 20000
iters <- 50000

# theta
theta_neff <- matrix(0,nrow = nsites, ncol = ntheta)
theta_geweke <- matrix(0,nrow = nsites, ncol = ntheta)
theta_autocorr <- matrix(0,nrow = nsites, ncol = ntheta)
theta_arate    <- theta_geweke <- matrix(0,nrow = nsites, ncol = ntheta)

for(s in 1:nsites){
  if(!(s %in% dead_sites)){
    samps <- mcmc(keep_theta[burn:iters,,s], start=1, end=iters-burn+1, thin=1)
    theta_neff[s,]    <- effectiveSize(samps)
    theta_geweke[s,]  <- geweke.diag(samps)$z
    theta_autocorr[s,] <- diag(autocorr(samps)[2,,])
    theta_arate[s,]   <- 1-rejectionRate(samps)
  }
}

# for mu
samps <- mcmc(keep_mu[x.axis,], start=1, end=iters-burn+1, thin=1)
geweke.diag(samps)
summary(samps)
plot(samps)
1-rejectionRate(samps)
diag(autocorr(samps)[2,,])
effectiveSize(samps)

# for capsigma
capsigma_neff <- matrix(0, nrow=10,ncol=8)
idx <- 1
for(i in 1:4){
  for(j in i:4){
    samps <- mcmc(matrix(keep_capsigma[i,j,burn:iters],nrow=iters-burn+1,ncol=1), start=1, end=i
ters-burn+1, thin=1)
    capsigma_neff[idx, 1] = i
    capsigma_neff[idx, 2] = j
    capsigma_neff[idx, 3] = quantile(samps, 0.025)
    capsigma_neff[idx, 4] = mean(samps)
    capsigma_neff[idx, 5] = quantile(samps, 0.975)
    capsigma_neff[idx, 6] = floor(effectiveSize(samps))
    capsigma_neff[idx, 7] = geweke.diag(samps)$z
    capsigma_neff[idx, 8] = autocorr(samps)[2,,]
    idx <- idx + 1
  }
}
capsigma_neff
```

Compute post mean & sd of theta2_s as well as P(theta2_s > 0)

```
burn <- 20000
iters <- 50000

theta2_post_means <- rep(0,nsites)
theta2_post_sds   <- rep(0,nsites)
theta2_post_gr0   <- rep(0,nsites)
theta2_lower      <- rep(0,nsites)
theta2_upper      <- rep(0,nsites)
theta2_sig        <- rep(0,nsites)

for (s in 1:nsites){
  theta2_post_means[s] <- mean(keep_theta[burn:iters,2,s])
  theta2_post_sds[s]   <- sd(keep_theta[burn:iters,2,s])
  theta2_post_gr0[s]   <- sum(keep_theta[burn:iters,2,s] > 0)/(iters-burn+1)
  theta2_lower[s]      <- quantile(keep_theta[burn:iters,2,s], 0.025)
  theta2_upper[s]      <- quantile(keep_theta[burn:iters,2,s], 0.975)
  theta2_sig[s]        <- sign(theta2_post_means[s])*(theta2_lower[s]*theta2_upper[s]>0)
}
```

Construct the plots & maps.

```
dead_sites <- c(201, 213)

hist(theta2_post_means[-dead_sites])
hist(theta2_post_sds[-dead_sites])

dat <- data.frame(long = sites2[,1], lat = sites2[,2],
                  means = theta2_post_means,
                  stddevs   = theta2_post_sds,
                  gr0     = theta2_post_gr0,
                  sig   = factor(theta2_sig, labels = c("Neg", "Nonsig", "Pos")))


p1 <- ggplot(dat[-dead_sites,], aes(long, lat)) +
  borders("state") +
  geom_point(aes(colour = means)) +
  scale_colour_gradientn(name = "Means", colours = viridis(10)) +
  coord_map(projection = "albers", lat0 = 39, lat1 = 45, xlim = c(-120,-70)) +
  xlab("")+ylab("")+labs(title=expression(paste("Posterior Means - ", theta[2]))) +
  theme_bw()
  #theme(legend.position = c(0.9,0.3))

p2 <- ggplot(dat[-dead_sites,], aes(long, lat)) +
  borders("state") +
  geom_point(aes(colour = stddevs)) +
  scale_colour_gradientn(name = "Std Devs", colours = viridis(10)) +
  coord_map(projection = "albers", lat0 = 39, lat1 = 45, xlim = c(-120,-70)) +
  xlab("")+ylab("")+labs(title=expression(paste("Posterior Std Devs - ", theta[2]))) +
  theme_bw()
  #theme(legend.position = c(0.9,0.3))

p3 <- ggplot(dat[-dead_sites,], aes(long, lat)) +
  borders("state") +
  geom_point(aes(colour = gr0)) +
  scale_colour_gradientn(name = expression(paste("P(",theta[2]," > 0|Y)")), colours = viridis(10
)) +
  coord_map(projection = "albers", lat0 = 39, lat1 = 45,xlim = c(-120,-70)) +
  xlab("")+ylab("")+labs(title=expression(paste("Posterior Probability - P(",theta[2]," > 0|Y)"
))) +
  theme_bw()
  #theme(legend.position = c(0.9,0.3))

p4 <- ggplot(dat[-dead_sites,], aes(long, lat)) +
  borders("state") +
  geom_point(aes(color = sig)) +
  scale_color_brewer(name = "Significance",palette = "Set1")+
  coord_map(projection = "albers", lat0 = 39, lat1 = 45, xlim = c(-120,-70)) +
  xlab("")+ylab("")+labs(title=expression(paste("Posterior Significance - ", theta[2]))) +
  theme_bw() + guides(color = guide_legend(override.aes = list(size=3)))

require(gridExtra)
grid.arrange(p1,p2,p3,p4, ncol=2)
```