Package `SSsimple`

Dave Zes

April 1, 2013

1 Intro

Our objective with `SSsimple` is a lean, easily implemented suite of functions to serve both didactically and as a practical means to perform meaningful analyses on real data. Our approach here will be mostly deductive. We will think generally about features of the state space system, and simulate and visualize the resulting manufactured observations.

As an author of this sort of document there is a great temptation to pontificate on the subject matter behind the code; however, this is a tutorial, and I will keep digressions to a minimum. There exist countless sources detailing theory, history, etc. of dynamic systems. For comprehensive treatments, see [4]; [2]; [7]. For brief exposition, consider [5]; [3]. For mean function estimation (for creation of the soon-to-be-introduced matrix, $\mathbf{H}$), see [6]; [1].

2 What's a State Space System?

A state space system is an idealized mathematical construct used to describe certain types of action in time (and in potentially other domains, like space). In particular,

$$\boldsymbol{\beta}_t = \mathbf{F}\boldsymbol{\beta}_{t-1} + \boldsymbol{\nu}_t\,, \qquad \boldsymbol{\nu} \sim \mathcal{N}[\mathbf{0}, \mathbf{Q}] \tag{1}$$

$$\mathbf{z}_t^T = \mathbf{H}\boldsymbol{\beta}_t + \boldsymbol{\varepsilon}_t\,, \qquad \boldsymbol{\varepsilon} \sim \mathcal{N}[\mathbf{0}, \mathbf{R}] \tag{2}$$

The latent state at time t, $\boldsymbol{\beta}_t$, is $d \times 1$; the system function, $\mathbf{F}$, and the state variance, $\mathbf{Q}$, are $d \times d$; the measurement function, $\mathbf{H}$, is $n \times d$; the measurement variance, $\mathbf{R}$, is $n \times n$. A more general variant allows the system *hyperparameters*, $\mathbf{F}, \mathbf{Q}, \mathbf{H}, \mathbf{R}$, to be time-varying. Our general philosophy here is that great volumes of important statistical modeling can be done without this generalization.

There are certainly situations, though, where it will be sensible to allow the measurement function, **H**, to be a function of time, or a function of exogenous variables that are themselves functions of time.

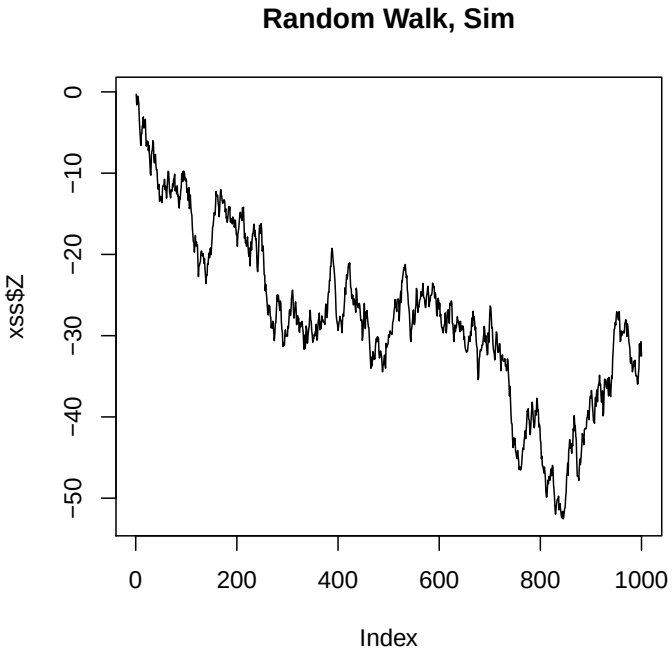
3 The Temporal Interpretation

The knee-jerk reaction to (1) and (2) is to imagine the observations, $\mathbf{z}_t$, as being dependent only upon the time domain, the simplest embodiment of which would be a "random walk," attained by setting $H = F = 1$, and $R = 0$.

3.1 Example, Local Level

```
> library(SSsimple)
> F <- 1
> H <- matrix(1) ### a 1 x 1 matrix, implying to SS.sim() that n=1 and d=1
> Q <- 1
> R <- 0
> tt <- 1000
> set.seed(999)
> xss <- SS.sim(F=F, Q=Q, H=H, R=R, length.out=tt, beta0=0)
```

```
> plot( xss$Z, type="l", main="Random Walk, Sim" )
```



3.2 Example, Smiles & Frowns

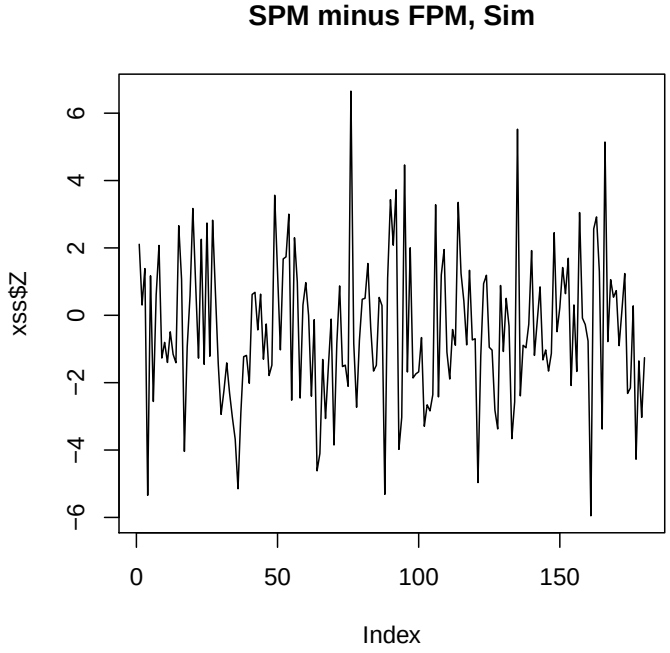
Let us imagine, just for the sake of illustration, we are studying the behavior of a friend. Our response (the observation) will be measured each minute for 3 hours as the number of smiles minus the number of frowns recorded over the prior minute (SMFPM). The underlying *state* that gives rise to this observation contains two elements, one we'll call the "humor index," the other we'll call the "happiness index." We decide that the response is deterministically equal 0.4 times the humor index plus 0.3 times the happiness index. We furthermore suppose that the magnitude of the states at any minute is in part a deterministic admixture of the magnitude of the two states during the prior minute, say the happiness index at time t equals 0.65 times the happiness index at $t-1$ plus 0.3 times the humor index at $t-1$, and analogously, the humor index at time t equals 0.65 times the humor index at $t-1$ plus 0.3 times the happiness index at $t-1$. FInally suppose that at each minute the response has a zero-mean gaussian stochastic component with variance 5, and the state possesses a zero-mean stochastic component with covariance $\mathsf{diag}[(0.2, 0.2)]$.

Let's simulate such a system:

```
> F <- matrix( c( 0.65, 0.3, 0.3, 0.65 ), 2, 2 ) ;
> H <- matrix( c(0.4, 0.3), 1, 2 ) ### a 1 x 2 matrix, so SS.sim() knows n=1 and d=2 ;
```

```
> Q <- 0.2 ;
> R <- 5 ;
> tt <- 180 ;
> set.seed(999) ;
> xss <- SS.sim(F=F, Q=Q, H=H, R=R, length.out=tt, beta0=0) ;

> plot( xss$Z, type="l", main="SPM minus FPM, Sim" )
```



Let's now use known hyperparameter values to produce posterior estimates of the true latent states:

```
> P0 <- diag(Q, 2) %*% solve(diag(1, 2) - t(F) %*% F)
> xslv <- SS.solve(Z=xss$Z, F=F, Q=Q, H=H, R=R, length.out=tt, P0=P0, beta0=0)
> Z.hat <- t(H %*% t(xslv$B.apri))
> sqrt( mean( (xss$Z - Z.hat)^2 ) )

[1] 2.213435
```

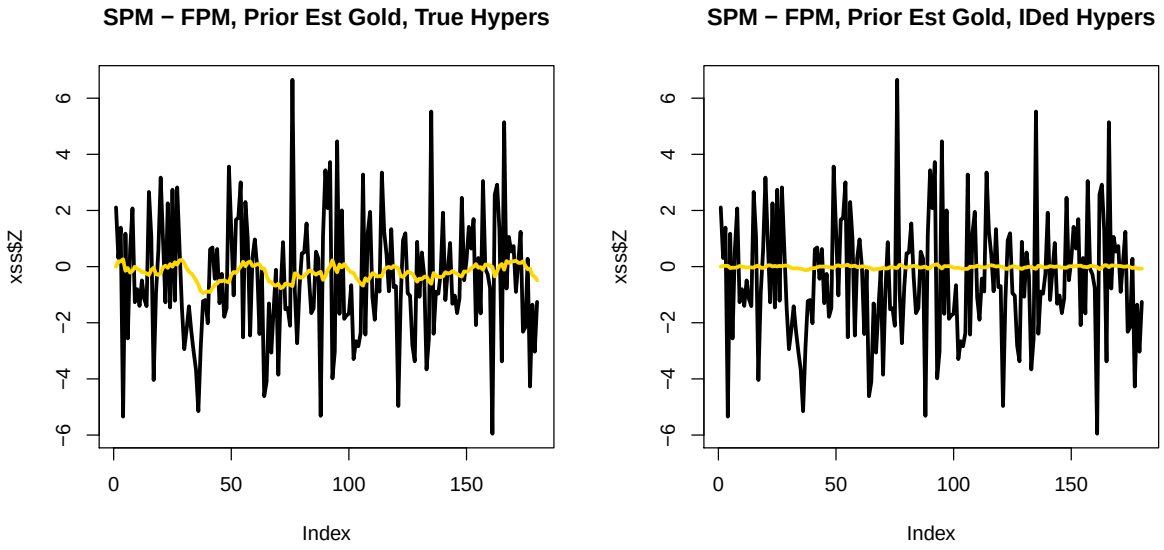
```
> par( mfrow=c(1,2) )
> plot(xss$Beta[ , 1], type="l", ylim=range(xss$Beta), col="red",
+    ylab="Humor Index (True is Heavy Line)",
+    main="Humor: True State and Posterior Est State", lwd=4)
> points(xslv$B.apos[ , 1], type="l", ylim=range(xslv$B.apos), col="red")
> plot(xss$Beta[ , 2], type="l", ylim=range(xss$Beta), col="blue",
+    ylab="Happiness Index (True is Heavy Line)",
+    main="Happiness: True State and Posterior Est State", lwd=4)
> points(xslv$B.apos[ , 2], type="l", ylim=range(xslv$B.apos), col="blue")
```

ID system.

```
> xid <- SS.ID( xss$Z, d=2 )
> xslv <- SS.solve(Z=xss$Z, F=xid$F, Q=xid$Q, H=xid$H, R=xid$R,
+    length.out=tt, P0=P0, beta0=0)
> Z.hat.2 <- t(xid$H %*% t(xslv$B.apri))
> sqrt( mean( (xss$Z - Z.hat.2)^2 ) )
```

```
[1] 2.222754
```

```
> par( mfrow=c(1,2) )
> plot( xss$Z, type="l", main="SPM - FPM, Prior Est Gold, True Hypers", lwd=3 )
> points( Z.hat, type="l", lwd=3, col="gold" )
> plot( xss$Z, type="l", main="SPM - FPM, Prior Est Gold, IDed Hypers", lwd=3 )
> points( Z.hat.2, type="l", lwd=3, col="gold" )
```



3.3 Example, Smiles & Frowns again, with Visual Stimulous

Let's now add an exogenous variable to our model of our friend's SMFPM. From minute 60 to minute 90, we are going to play for our friend choice excerpts from the delightful, though very dark Belgian comedy, *Man Bites Dog*. From minute 120 to minute 150, we shall deliberately assail our friend with excepts from the television catastrophe, *Full House*.

We will add the effect of this visual stimulation into the *observation space*, i.e., build it into (2), more precisely, we shall insert it into $\mathbf{H}_t$. Our model assumption will be that the comedy will have a purely deterministic mean effect of +9 SMFPM, and *Full House*, −9. The effect of these stimuli will also be manifest through a state variable, "receptivity," that is independent (both through **F** and **Q**) of the other states, with a variance of 0.01, a system coefficient of 0.99, and a mapping into the observation space of unity.

```
> d <- 4
> n <- 1
> F <- matrix(0, d, d)
> F[ 1:2, 1:2 ] <- c(0.65, 0.3, 0.3, 0.65)
```

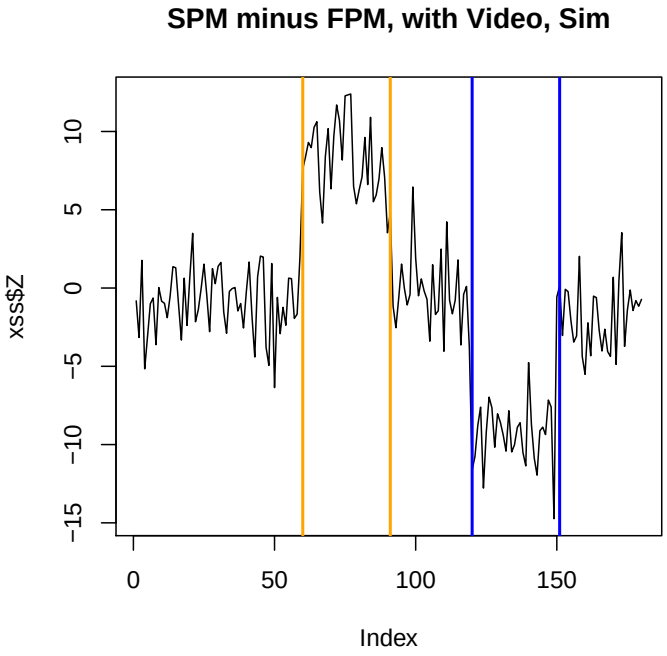
```
> F[ 3, 3 ] <- 1
> F[ 4, 4 ] <- 0.99
> eigen(F) #### check stability


$values
[1] 1.00 0.99 0.95 0.35

$vectors
     [,1] [,2]      [,3]       [,4]
[1,]    0    0 0.7071068  0.7071068
[2,]    0    0 0.7071068 -0.7071068
[3,]    1    0 0.0000000  0.0000000
[4,]    0    1 0.0000000  0.0000000


> tt <- 180
> H.tv <- list()
> for(i in 1:tt) {
+    H.tv[[i]] <- matrix( c(0.4, 0.3, 0, 1), n, d )
+    if( i >= 60 & i < 90 ) { H.tv[[i]][ , 3] <- 9 }
+    if( i >= 120 & i < 150 ) { H.tv[[i]][ , 3] <- -9 }
+ }
> Q <- diag( 0.2, d )
> Q[ 3, 3 ] <- 0
> Q[ 4, 4 ] <- 1/100
> R <- 5
> beta0 <- c(0, 0, 1, 0)
> set.seed(999)
> xss <- SS.sim.tv(F=F, Q=Q, H=H.tv, R=R, length.out=tt, beta0=beta0)
```

```
> plot( xss$Z, type="l", main="SPM minus FPM, with Video, Sim" ) ;
> abline(v=60, col="orange", lwd=2) ;
> abline(v=91, col="orange", lwd=2) ;
> abline(v=120, col="blue", lwd=2) ;
> abline(v=151, col="blue", lwd=2)
```



4 The Spacio-Temporal Interpretation

We can use our system, (1)-(2), to describe spacio-temporal phenomenon by making two key connections. First, we will call upon **H** to help define a smooth mean function over space, second, we'll have **R** describe covariance as a function of distance between spacial locations.

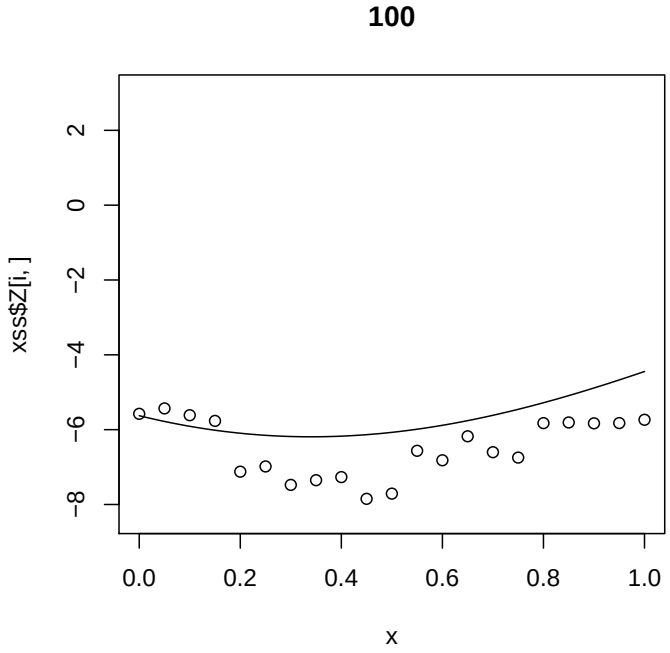
4.1 Example, ST over 1D

Let's say our spacial domain $\Omega = [0, 1] \subset \mathbb{R}^1$, and have 21 sites evenly spaced over Ω.

```
> x <- I(0:20) / 20
> n <- length(x)
> H <- H.omega.sincos(x, c(1,2))
> F <- 0.999 ; Q <- 0.1
```

```
> ## the following line constructs the matrix of Euclidean distances btwn locations
> D <- abs( tcrossprod(x, rep(1, n)) - tcrossprod(rep(1, n), x) )
> R <- exp(-3 * D)
> set.seed(999)
> xss <- SS.sim(F=F, Q=Q, H=H, R=R, length.out=100, beta0=0)
> xdom <- I(0:100) / 100
> Hdom <- H.omega.sincos(xdom, c(1,2))


> for(i in 1:tt) {
+    plot(x, xss$Z[i, ], ylim=range(xss$Z), main=i)
+    points( xdom, Hdom %*% xss$Beta[i,], type="l" )
+    Sys.sleep(0.1)
+ }
```



4.2 Example, ST over 2D

Here we'll simulate a system over $\Omega = [0,1] \times [0,1] \subset \mathbb{R}^2$ and have our locations be an evenly spaced 11 by 11 grid. We will also use true state values to create a raster map of (posterior) spacial

predictions. Recall the best $L2$ estimate of a response at a new location comes by way of

$$\widehat{\mathbf{z}}_t = \mathbf{H}\,\boldsymbol{\beta}_t \tag{3}$$

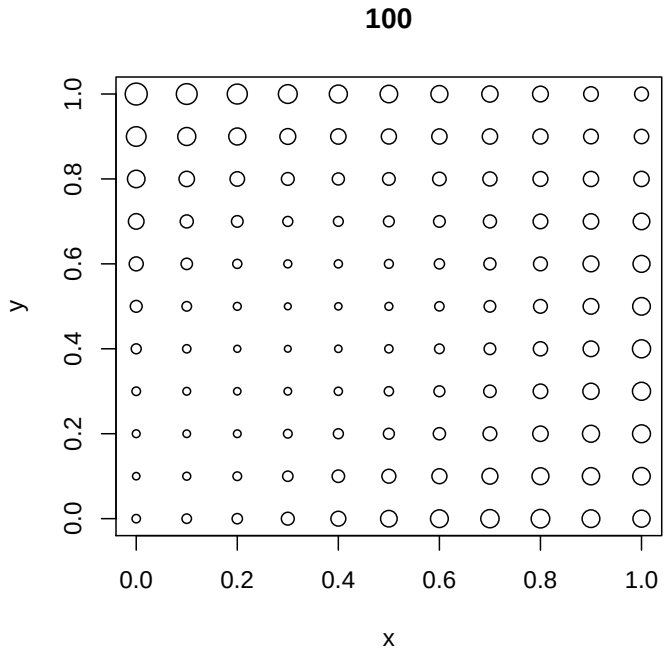
$$\widehat{z}_{0\,t} = \mathbf{h}_0\,\boldsymbol{\beta}_t \tag{4}$$

$$\widetilde{z}_{0\,t} = \widehat{z}_{0\,t} + (\mathbf{z}_t - \widehat{\mathbf{z}}_t)\ \mathbf{R}\,\mathbf{r}_0 \tag{5}$$

where $\mathbf{h}_0$ is the $1 \times d$ bases expansion at the new location (naturally, this expansion should always be the same as that used to create $\mathbf{H}$), and $\mathbf{r}_0$ is the $n \times 1$ covariance between the new location and the existing locations (and, naturally, this covariance should be created in the same vein as $\mathbf{R}$).

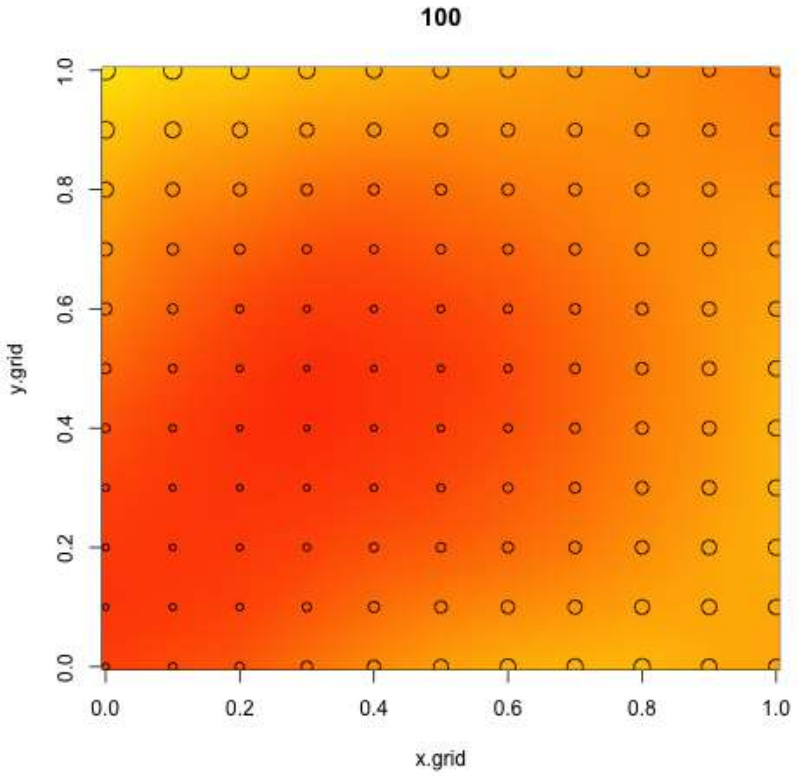
```
> x <- rep( 0:10 / 10, 11 )
> y <- rep( 0:10 / 10, each=11 )
> n <- length(x)
> Hx <- H.omega.sincos( x, c(1,2,3)*pi / 2 )
> Hy <- H.omega.sincos( y, c(1,2,3)*pi / 2 )
> ## Construct the tensor bases expansion over Omega
> H <- matrix(NA, nrow(Hx), ncol(Hx)*ncol(Hy))
> k <- 0
> for(i in 1:ncol(Hx)) {
+    for(j in 1:ncol(Hy)) {
+       k <- k+1
+       H[ , k]  <- Hx[ ,i ] * Hy[ , j]
+    }
+ }
> Dx <- tcrossprod(x, rep(1, n)) - tcrossprod(rep(1, n), x)
> Dy <- tcrossprod(y, rep(1, n)) - tcrossprod(rep(1, n), y)
> D <- sqrt( Dx^2 + Dy^2 ) ; ## the Euclidean distance matrix
> R <- exp(-3 * D)
> xss <- SS.sim( 0.99, H, 1/2, R,   500, rep(0, ncol(H)) )
```

```
> for(i in 1:nrow(xss$Z)) {
+    plot(x, y, cex=(xss$Z[i ,]-min(xss$Z))/30, main=i)
+    Sys.sleep(0.1)
+ }
```




```
> ###################### raster map, interpolation to arb locs
> x.grid <- 0:100 / 100
> y.grid <- 0:100 / 100
> xdom <- rep( x.grid, length(y.grid) )
> ydom <- rep( y.grid, each=length(x.grid) )
> Hx.dom <- H.omega.sincos( xdom, c(1,2,3)*pi / 2 )
> Hy.dom <- H.omega.sincos( ydom, c(1,2,3)*pi / 2 )
> Hdom <- matrix(NA, nrow(Hx.dom), ncol(Hx.dom)*ncol(Hy.dom))
> k <- 0
> for(i in 1:ncol(Hx.dom)) {
+    for(j in 1:ncol(Hy.dom)) {
+       k <- k+1
+       Hdom[ , k]  <- Hx.dom[ ,i ] * Hy.dom[ , j]
+    }
+ }
> Dx.dom <- tcrossprod(x, rep(1, length(xdom))) - tcrossprod(rep(1, n), xdom)
> Dy.dom <- tcrossprod(y, rep(1, length(ydom))) - tcrossprod(rep(1, n), ydom)
> D.dom <- sqrt( Dx.dom^2 + Dy.dom^2 )
> R0 <- exp(-3 * D.dom)
> bb <- solve(R) %*% R0
```

```
> for(i in 1:nrow(xss$Z)) {
+    z.hat <- H %*% xss$Beta[i, ]
+    z.0 <- Hdom %*% xss$Beta[i, ]
+    z.tilde <- z.0 + t( t(xss$Z[i, ] - z.hat) %*% bb )
+    Z.mx <- matrix( z.tilde, length(y.grid), length(x.grid) )
+    image(x.grid, y.grid, Z.mx, zlim=range(xss$Z), main=i, col=heat.colors(10000))
+    points(x, y, cex=(xss$Z[i ,]-min(xss$Z))/30)
+    Sys.sleep(0.1)
+ }
```



4.3 Ozone

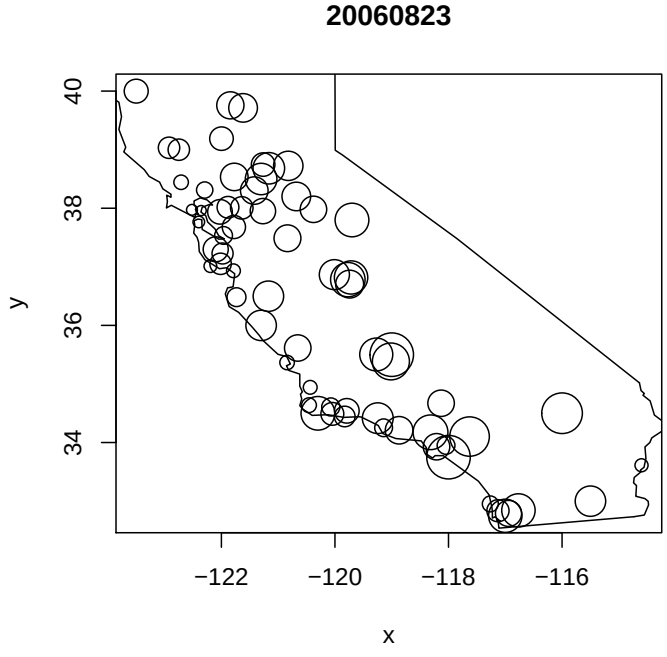
Here we'll use recorded ozone concentrations (ppb) from 68 sites, measured daily form 2005-2006 as the greatest hourly average.

Download data and view:

```
> ## library(maps)
```

```
> data(SS_O3) #### two dataframes, Z and locs
> Z <- SS_O3$Z
> locs <- SS_O3$locs
> xdate <- row.names(Z)
> x <- locs[ ,1]
> y <- locs[ ,2]
> Z <- as.matrix(Z)
> tt <- nrow(Z)
> n <- ncol(Z)
> Dx <- tcrossprod(x, rep(1, n)) - tcrossprod(rep(1, n), x)
> Dy <- tcrossprod(y, rep(1, n)) - tcrossprod(rep(1, n), y)
> D <- sqrt( Dx^2 + Dy^2 )


> for(i in 1:tt) {
+         plot(x, y, cex=(Z[i ,]-min(Z))/30, main=xdate[i])
+ ##        map("state", "california", add=TRUE)
+         Sys.sleep(1/5)
+ }
```



Assume a random walk model, check *a prior* RMSE. (Note: Only assess RMSE fit with the *a priori* state estimates; it is meaningless to fit in this way with the *a posteriori* state estimates.)

```
> Q <- 1
> F <- 1
> R <- 1
> H <- matrix( 1, n, 1 )
> xslv <- SS.solve(Z=Z, F=F, Q=Q, H=H, R=R, length.out=tt, P0=10^5, beta0=0)
> Z.hat <- t(H %*% t(xslv$B.apri))
> sqrt( mean( ( Z - Z.hat )[10:tt, ]^2 ) )
```

```
[1] 15.71792
```

Note that our RMSE here is 15.72.

Now let's try an additive sine-cosine expansion over California. We will also utilize an exponential covariance function.

```
> ux <- I(1:3)*pi / 20
> uy <- I(1:3)*pi / 20
> Hx <- H.omega.sincos( x, ux )
> Hy <- H.omega.sincos( y, uy )
> H <- cbind( rep(1,n), Hx, Hy )
> R <- exp( -0.11 * D )
> Q <- 1
> F <- 1
> xslv <- SS.solve(Z=Z, F=F, Q=Q, H=H, R=R, length.out=tt, P0=10^5, beta0=0)
> Z.hat <- t(H %*% t(xslv$B.apri))
> sqrt( mean( ( Z - Z.hat )[10:tt, ]^2 ) )
```

```
[1] 13.3834
```

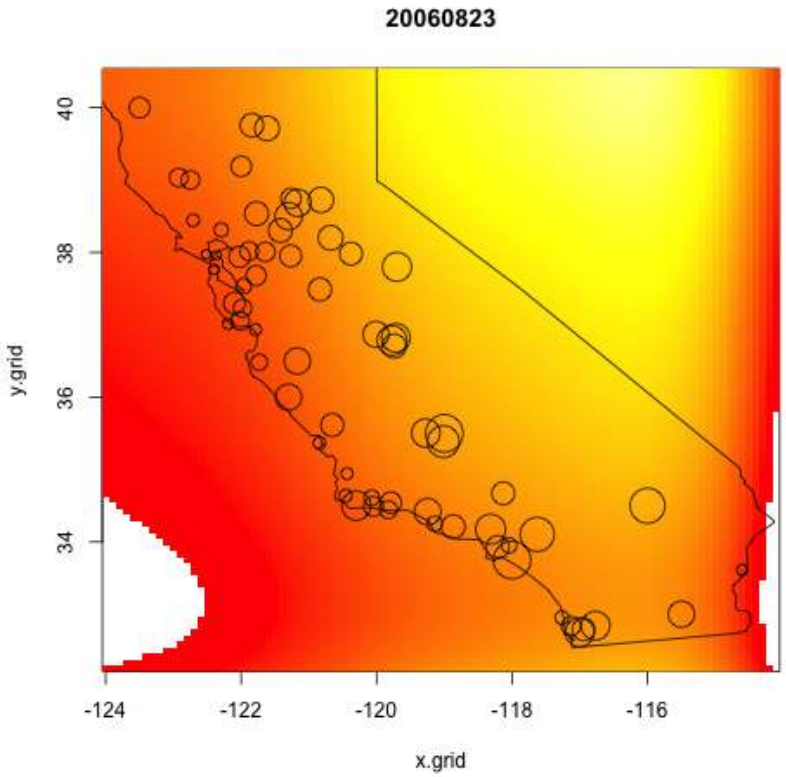
We've improved our RMSE to 13.38.

Plot spacial predictions:

```
> x.grid <- seq( min(x)-0.5, max(x)+0.5, length=100 )
> y.grid <- seq( min(y)-0.5, max(y)+0.5, length=100 )
> xdom <- rep( x.grid, length(y.grid) )
> ydom <- rep( y.grid, each=length(x.grid) )
```

```
> Hx.dom <- H.omega.sincos( xdom, ux )
> Hy.dom <- H.omega.sincos( ydom, uy )
> Hdom <- cbind( rep(1, length(xdom)), Hx.dom, Hy.dom )
```

```
> for(i in 1:nrow(Z)) {
+    Z.mx <- matrix( Hdom %*% xslv$B.apri[i, ], length(y.grid), length(x.grid) )
+    image(x.grid, y.grid, Z.mx, zlim=range(Z), main=xdate[i], col=heat.colors(10000))
+    points(x, y, cex=(Z[i ,]-min(Z))/30)
+    map("state", "california", add=TRUE)
+    Sys.sleep(0.1)
+ }
```



Let's try a bases tensor expansion over California using the same frequencies and plot spacial predictions:

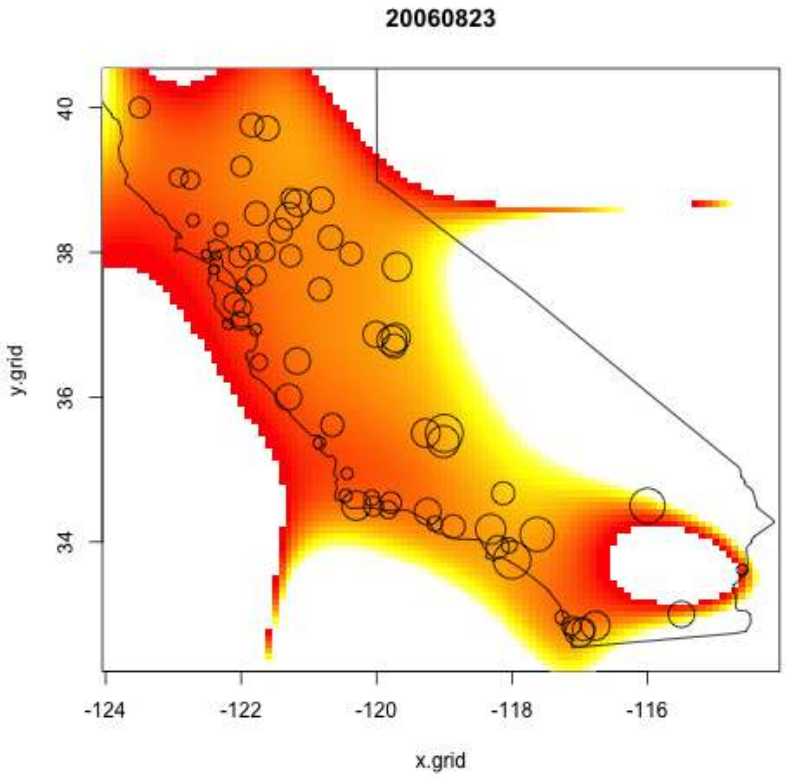
```
> ######################## tensor bases
> H <- matrix(NA, nrow(Hx), ncol(Hx)*ncol(Hy))
> k <- 0
```

```
> for(i in 1:ncol(Hx)) {
+    for(j in 1:ncol(Hy)) {
+       k <- k+1
+       H[ , k]  <- Hx[ ,i ] * Hy[ , j]
+    }
+ }
> H <- cbind( rep(1,n), H ) ### add intercept
> R <- exp( -0.11 * D )
> Q <- 1
> F <- 1
> xslv <- SS.solve(Z=Z, F=F, Q=Q, H=H, R=R, length.out=tt, P0=10^5, beta0=0)
> Z.hat <- t(H %*% t(xslv$B.apri))
> sqrt( mean( ( Z - Z.hat )[10:tt, ]^2 ) )

[1] 12.51818

> ####### new sites
> Hdom <- matrix(NA, nrow(Hx.dom), ncol(Hx.dom)*ncol(Hy.dom))
> k <- 0
> for(i in 1:ncol(Hx.dom)) {
+    for(j in 1:ncol(Hy.dom)) {
+       k <- k+1
+       Hdom[ , k]  <- Hx.dom[ ,i ] * Hy.dom[ , j]
+    }
+ }
> Hdom <- cbind( rep(1, length(xdom)), Hdom )

> for(i in 1:nrow(Z)) { ;
+    Z.mx <- matrix( Hdom %*% xslv$B.apri[i, ], length(y.grid), length(x.grid) ) ;
+    image(x.grid, y.grid, Z.mx, zlim=range(Z), main=xdate[i], col=heat.colors(10000)) ;
+    points(x, y, cex=(Z[i ,]-min(Z))/30) ;
+    map("state", "california", add=TRUE) ;
+    Sys.sleep(0.1) ;
+ }
```



Our RMSE has dropped to 12.90766, but exceedingly poor spacial predictions residing "outside" our inferential space are plainly evident.

Let's now consider a tessellation-style model that can be implied by setting $\mathbf{H} = \mathbf{I}$, and view in higher resolution:

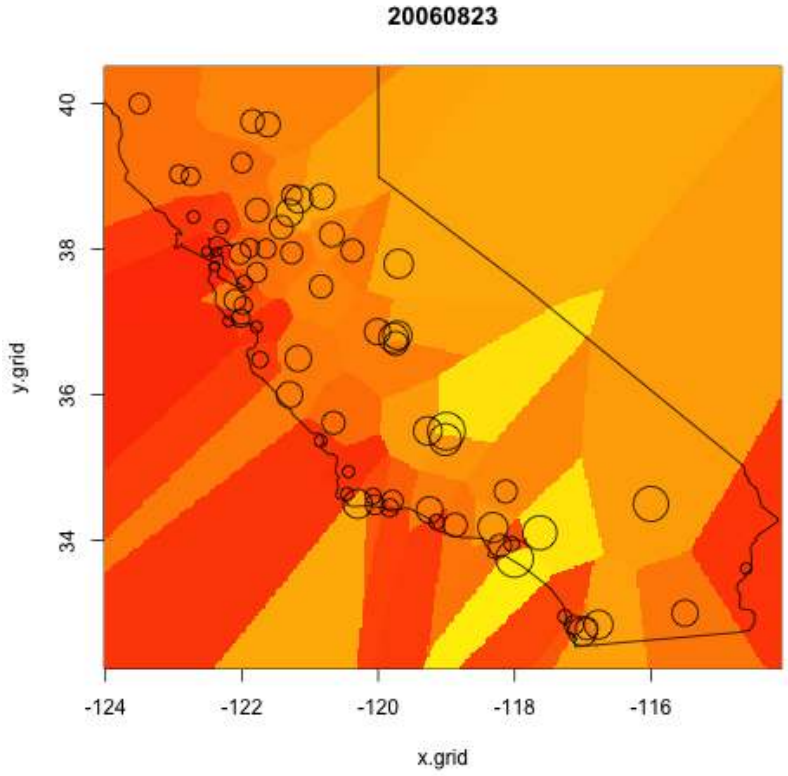
```
> H <- diag(1, n)
> R <- diag(1, n)
> d <- ncol(H)
> F <- diag(1, d)
> Q <- 1
> xslv <- SS.solve(Z=Z, F=F, Q=Q, H=H, R=R, length.out=tt, P0=10^5, beta0=0)
> Z.hat <- t(H %*% t(xslv$B.apri))
> sqrt( mean( ( Z - Z.hat )[10:tt, ]^2 ) )
```

```
[1] 10.25075
```

```
> x.grid <- seq( min(x)-0.5, max(x)+0.5, length=300 )
```

```
> y.grid <- seq( min(y)-0.5, max(y)+0.5, length=300 )
> xdom <- rep( x.grid, length(y.grid) )
> ydom <- rep( y.grid, each=length(x.grid) )
> Dx.dom <- tcrossprod(x, rep(1, length(xdom))) - tcrossprod(rep(1, n), xdom)
> Dy.dom <- tcrossprod(y, rep(1, length(ydom))) - tcrossprod(rep(1, n), ydom)
> D.dom <- t(  sqrt( Dx.dom^2 + Dy.dom^2 )  )
> xmin <- apply(D.dom, 1, min)
> xmin.mx <- matrix( xmin, nrow(D.dom), ncol(D.dom) )
> Hdom <- matrix( as.integer( D.dom == xmin.mx ), nrow(D.dom), ncol(D.dom) )
> rm( Dx.dom, Dy.dom, D.dom )


> for(i in 1:nrow(Z)) {
+    Z.mx <- matrix( Hdom %*% xslv$B.apri[i, ], length(y.grid), length(x.grid) )
+    image(x.grid, y.grid, Z.mx, zlim=range(Z), main=xdate[i], col=heat.colors(10000))
+    points(x, y, cex=(Z[i ,]-min(Z))/30)
+    map("state", "california", add=TRUE)
+    Sys.sleep(0.1)
+ }
```



Our RMSE looks much improved at 10.25.

Let's ID the system. Of course, our model assumptions are absurd (ozone concentrations are heavy right-skewed).

```
> xid <- SS.ID( Z + rnorm(tt*ncol(Z), 0, 0.1) , d=7, rsN <- c(3, 3, 350) ) ;
> xslv <- SS.solve(Z=Z, F=xid$F, Q=xid$Q, H=xid$H, R=xid$R, length.out=tt,
+    P0=10^5, beta0=0) ;
> Z.hat <- t(xid$H %*% t(xslv$B.apri)) ;
> sqrt( mean( ( Z - Z.hat )[10:tt, ]^2 ) )

[1] 10.42033
```

The RMSE is 10.42. Of course, when we ID the system, we find the observation function, $\mathbf{H}$, for the data. We cannot immediately use the IDed $\mathbf{H}$ to predict to new locations because it is *per se* not in a functional form we can use to extrapolate to unmonitored sites.

4.4 More O3

Let's examine a few more examples with the O3 data. Load and construct Euclidean distance matrix:

```
> data(SS_O3) #### two dataframes, Z and locs
> Z <- SS_O3$Z
> locs <- SS_O3$locs
> xdate <- row.names(Z)
> x <- locs[ ,1]
> y <- locs[ ,2]
> Z <- as.matrix(Z)
> tt <- nrow(Z)
> n <- ncol(Z)
> Dx <- tcrossprod(x, rep(1, n)) - tcrossprod(rep(1, n), x)
> Dy <- tcrossprod(y, rep(1, n)) - tcrossprod(rep(1, n), y)
> D <- sqrt( Dx^2 + Dy^2 )
```

Now, let's assume a very simple system with $Q = F = 1$ and $\mathbf{H} = (1, 1, ..., 1)^T$, (i.e., the measurement function is the mean function), and $\mathbf{R} = \exp[-\alpha \cdot \mathbf{D}]$ ($\mathbf{D}$ is our Euclidean distance matrix

between our 68 sites), and utilize *a posteriori* RMSE over cross validation to locate a suitable α.

```
> d <- 1
> H <- matrix(1, n, d)
> F <- 1
> Q <- 1
```

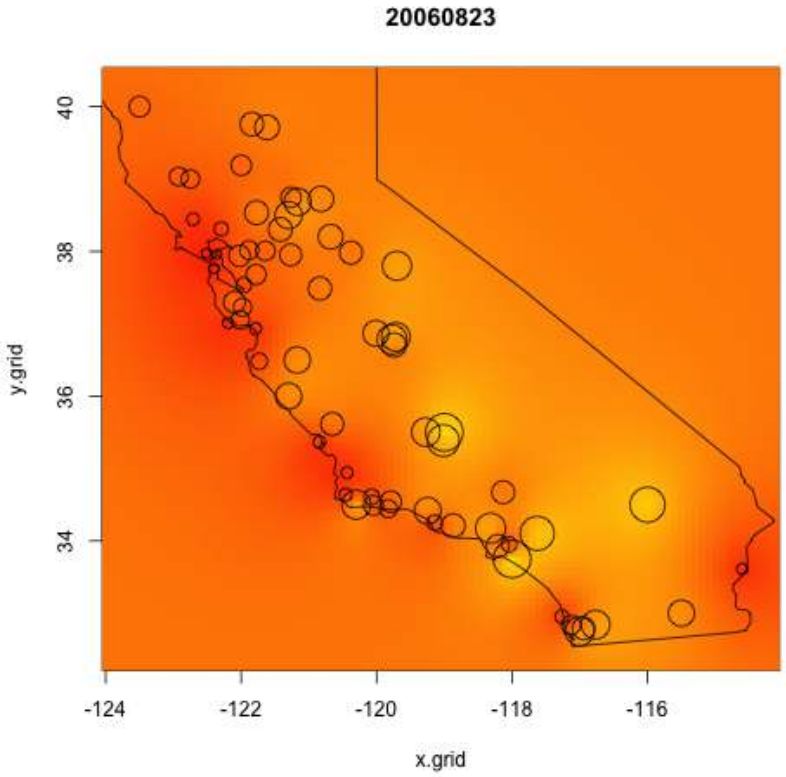
```
> for( alpha in I(2^(-5:4)) ) { ; #### this will take a while ...
+    Z.tilde <- matrix(NA, tt, n)
+    R <- exp( -alpha*D )
+    for( ii in 1:n ) {
+       cat(ii, " ")
+       bb <- solve( R[ -ii, -ii] ) %*% R[ -ii, ii]
+       xslv <- SS.solve(Z[ , -ii], F=F, Q=Q, H=H[-ii, , drop=FALSE],
+          R=R[ -ii, -ii], length.out=tt, P0=10^5, beta0=0)
+       Z.hat <- t(  H[-ii, , drop=FALSE] %*% t(xslv$B.apos)  )
+       z.0 <- H[ ii, , drop=FALSE ] %*% t(xslv$B.apos)
+       cov.adj <- (Z[ , -ii] - Z.hat) %*% bb
+       z.tilde <- t(z.0) + cov.adj
+       Z.tilde[ , ii] <- z.tilde[ , 1]
+    } ;
+    rmse <- sqrt( mean( ( Z - Z.tilde )[10:tt, ]^2 ) )
+    cat( alpha, rmse, "\n" )
+ }
```

It looks as if $\alpha = 1$ is a decent retrospective choice. Let's use this value to predict in space:

```
> alpha <- 1
> R <- exp( -alpha*D )
> xslv <- SS.solve(Z=Z, F=F, Q=Q, H=H, R=R, length.out=tt, P0=10^5, beta0=0)
> x.grid <- seq( min(x)-0.5, max(x)+0.5, length=100 )
> y.grid <- seq( min(y)-0.5, max(y)+0.5, length=100 )
> xdom <- rep( x.grid, length(y.grid) )
> ydom <- rep( y.grid, each=length(x.grid) )
> Dx.dom <- tcrossprod(x, rep(1, length(xdom))) - tcrossprod(rep(1, n), xdom)
> Dy.dom <- tcrossprod(y, rep(1, length(ydom))) - tcrossprod(rep(1, n), ydom)
> D.dom <- t(  sqrt( Dx.dom^2 + Dy.dom^2 )  )
```

```
> R0 <- exp(-alpha*D.dom)
> bb <- solve(R) %*% t(R0)
> Hdom <- matrix(1, length(xdom), 1)


> for(i in 1:nrow(Z)) {
+    z.hat <- H %*% xslv$B.apos[i, ]
+    z.0 <- Hdom %*% xslv$B.apos[i, ]
+    z.tilde <- z.0 + t( t(Z[i, ] - z.hat) %*% bb )
+    Z.mx <- matrix( z.tilde, length(y.grid), length(x.grid) )
+    image(x.grid, y.grid, Z.mx, zlim=range(Z), main=xdate[i], col=heat.colors(10000))
+    points(x, y, cex=(Z[i ,]-min(Z))/30)
+    map("state", "california", add=TRUE)
+    Sys.sleep(0.1)
+ }
```



5 Final Thoughts

The system of attention, (1) and (2), should be regarded as remarkably flexible, and can spawn in one's imagination countless potential applications.

When fitting to data, as a starting point consider setting $\mathbf{Q} = \mathbf{F} = \mathbf{I}$, have $\mathbf{H}$ be some function over important exogenous covariates, and have $\mathbf{R}$ reflect covariance in the response attributable to those exogenous variables. Recall that $\{\mathbf{Q}, \mathbf{R}\}$ forms a (sort of) equivalence class of solutions, e.g., the solution, $\widehat{\mathbf{Z}}$ or $\widetilde{\mathbf{Z}}$, of a system with $\mathbf{Q} = 1 \cdot \mathbf{I}, \mathbf{R} = 5 \cdot \mathbf{I}$ will be identical to that created using $\mathbf{Q} = 3 \cdot \mathbf{I}, \mathbf{R} = 15 \cdot \mathbf{I}$.

If desiring a good quality *a posteriori* estimate, do not judge a fit using a measure of distance (e.g., RMSE) between the observations and the *a posteriori* estimate using a solution made from the full data. Instead, use (leave-one-site-out-at-a-time) cross validation (just as we've done in the example above).

References

[1] S. Efromovich. *Nonparametric Curve Estimation (Springer Series in Statistics)*, volume 1862. Springer U.S., New York, 1999.

[2] S. Haykin. *Adaptive Filter Theory*. Prentice-Hall, Upper Saddle River, forth edition, 2002.

[3] J. H. McCulloch. The kalman foundations of adaptive least squares, with application to u.s. inflation. *Unpublished*, 2005.

[4] A. H. Sayed. *Fundamentals of Adaptive Filetering*. John Wiley & Sons, Hoboken, N.J., 2003.

[5] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications, With R Examples*. Springer, N.Y., second edition, 2006.

[6] L. Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[7] M. West and J. Harrison. *Bayesian Forecasting and Dynamic Models*. Springer-Verlag, New York, second edition, 1997.