Kalman Filter Experiments

Eric Bridgeford

August 19, 2016

Kalman Filter

The Kalman Filter is an efficient algorithm that uses a series of measurements over time (with known/approximated statistical noises and inaccuracies) and produces estimates of latent variables that tend to be quite precise.

Prediction

For each time step, we begin by making a prediction based on our previous state, as well as known noises/inaccuracies. We estimate our sample state vector, X_k, and our Process Covariance Matrix, P_k:

$$X_{k,p} = FX_{k-1} + Bu_k \tag{1}$$

$$P_{k,p} = AP_{k-1} + Q \tag{2}$$

Using our estimated values, we compute the Kalman Gain, K, which gives us an idea of how much of our estimate will be the actual measurement versus the predicted state. If we have highly inaccurate sensors, we might expect our prediction to matter more (can factor in previous observations) than our actual measure, and K will be lower. If we have accurate sensors, we would expect our prediction to matter less. Here, H represents our extraction matrix, where we theoretically can recover our measures from our latent variables through $Z = HX + \varepsilon$. R represents our measurement covariance matrix, which gives us a sense of the presence of errors in our measure.

$$K = P_{k,p}H^T(HP_{k,p}H^T + R_k)^{-1} \tag{3}$$

Taking our K into consideration, we can now update our predictions to reflect how accurate we think our predictions are versus how accurate our measurements are. We adjust our predictions, given actual measurements Z:

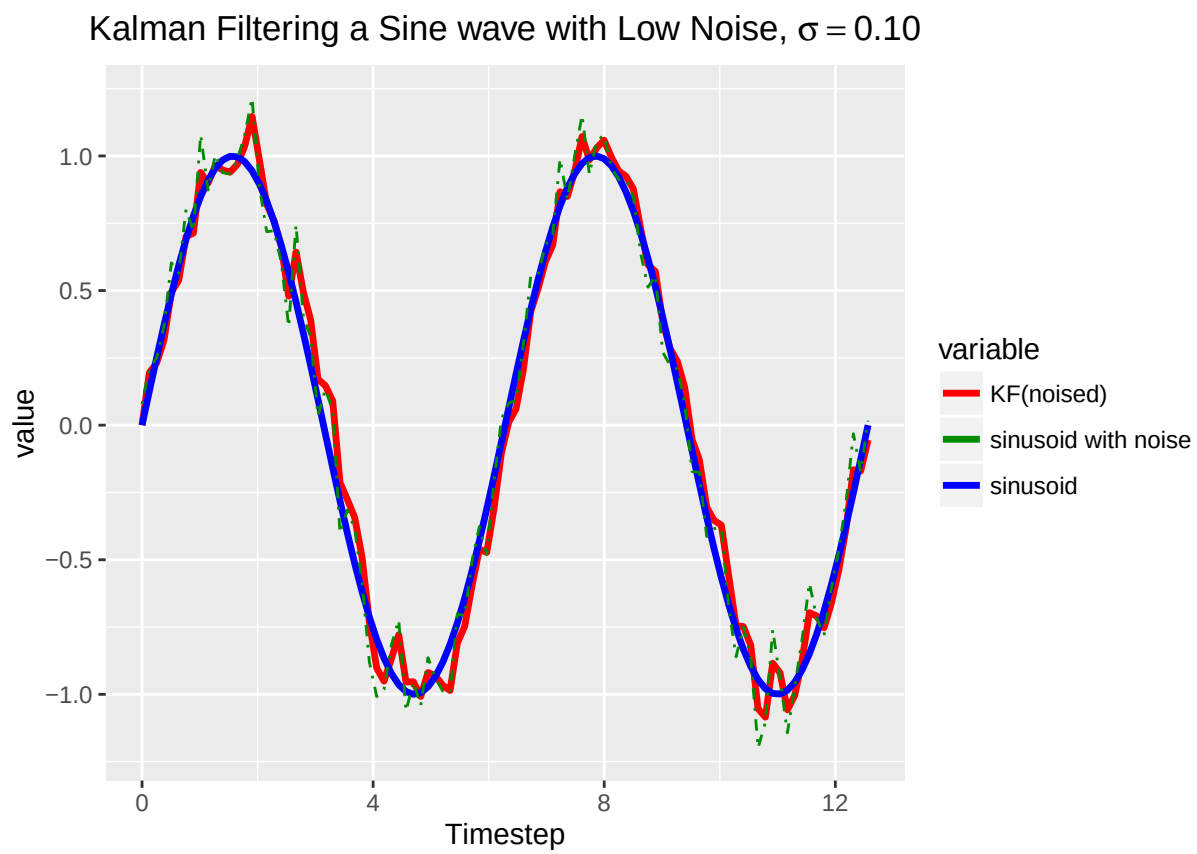
$$X_k = X_{k,p} + K(Z_k - HX_{k,p}) \tag{4}$$

$$P_k = P_{k,p} - KHP_{k,p} \tag{5}$$

Sine Wave with Gaussian Noise

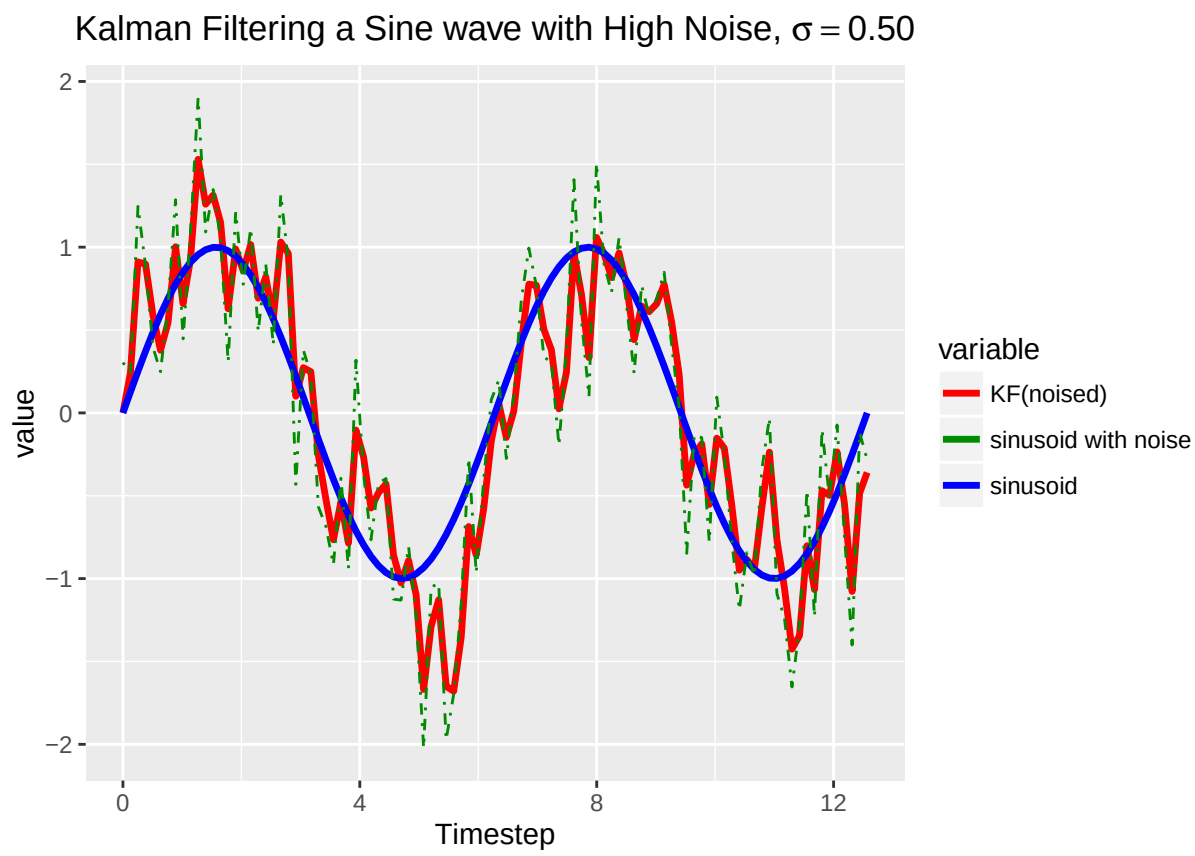
In this simple example, we consider a gaussian signal with underlying zero-mean gaussian noise, with $\sigma = .25$. We construct an input signal given our parameters, and then apply our kalman filter. Note that our kalman filter variables are passed as arrays; this is necessary for the function to be compatible with multivariate inputs.

```r
source('kalman_filter.R')
require('reshape2')
require('ggplot2')
require(latex2exp)
t <- seq(0, 4*pi,,100)  # simple vector
signal <- sin(t)
sd=.1
noise = rnorm(100, mean=0, sd=sd)
Z <- array(signal + noise, dim=c(100, 1))
filtered <- kalman_filter(Fm=array(1, dim=c(1,1)),Z=Z, H=array(1, dim=c(1,1)),
                          R=array(.001, dim=c(1,1,1)), u=array(0, dim=c(1,1)),
                          B=array(0, dim=c(1,1)), Q=array(.001, dim=c(1,1)),
                          p0=array(sd^2, dim=c(1,1)), x0=array(0, dim=c(1,1)))
data <- data.frame(Timestep = t, KF=filtered$state, sinusoid=signal, noised=Z)
mdata <- melt(data, id=c("Timestep"))
ggplot(data = mdata, aes(x=Timestep, y=value, group=variable, color=variable)) +
  geom_line(data = mdata[mdata$variable != "noised",], size=1.2) +
  geom_line(data = mdata[mdata$variable == "noised",], size=.5, linetype='dotdash') +
  scale_color_manual(values=c("KF"='red', "sinusoid"='blue', "noised"='green4'),
                     labels=c("KF"="KF(noised)", "sinusoid"="sinusoid", "noised"="sinusoid with noise
  ggtitle(TeX(sprintf('Kalman Filtering a Sine wave with Low Noise, $\\sigma = %.2f$', sd)))
```



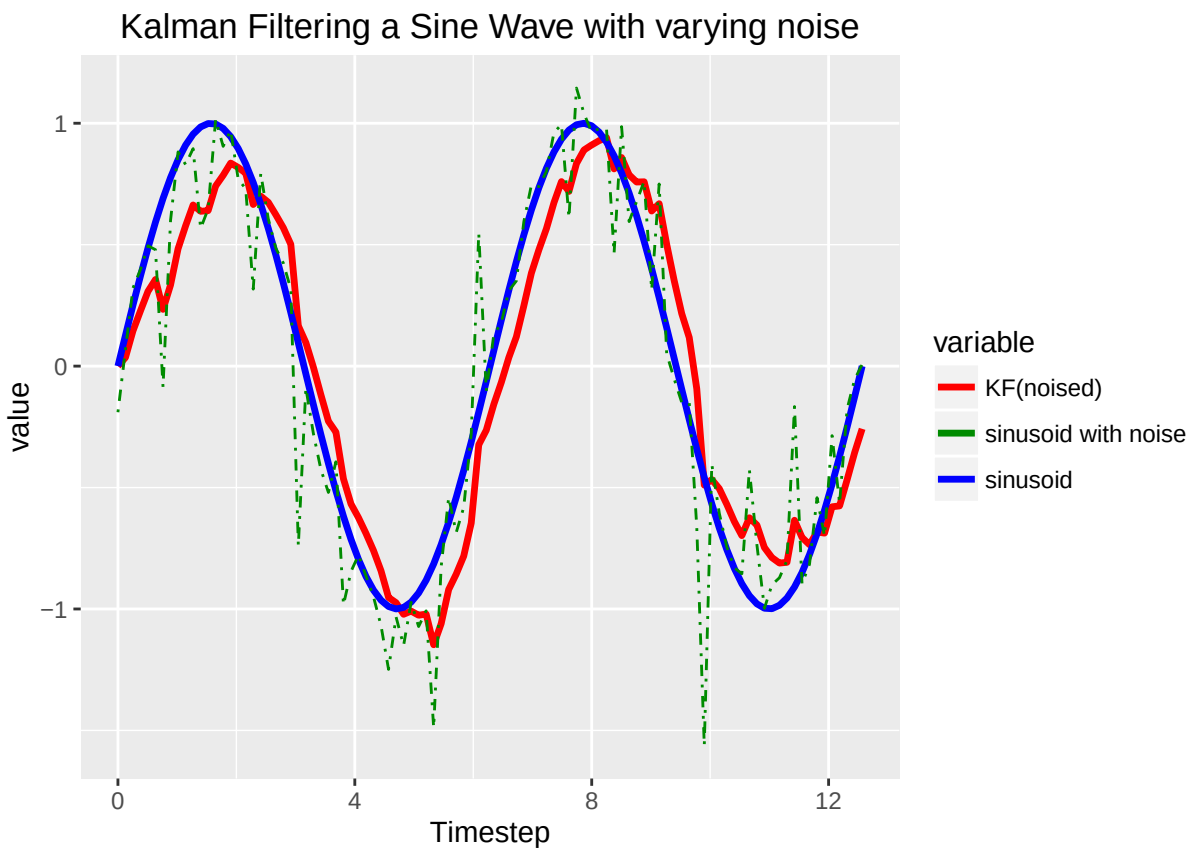
It is clear that the kalman filter does a very good job at determining the latent signal with a low noise component added. Let's try with a larger noise component:

```r
source('kalman_filter.R')
  require('reshape2')
  require('ggplot2')
  require('latex2exp')
  t <- seq(0, 4*pi,,100)  # simple vector
  signal <- sin(t)
  sd = 0.5
  noise = rnorm(100, mean=0, sd=0.5)
  Z <- array(signal + noise, dim=c(100, 1))
  filtered <- kalman_filter(Fm=array(1, dim=c(1,1)),Z=Z, H=array(1, dim=c(1,1)),
                            R=array(.001, dim=c(1,1,1)), u=array(0, dim=c(1,1)),
                            B=array(0, dim=c(1,1)), Q=array(.001, dim=c(1,1)),
                            p0=array(sd^2, dim=c(1,1)), x0=array(0, dim=c(1,1)))
  data <- data.frame(Timestep = t, KF=filtered$state, sinusoid=signal, noised=Z)
  mdata <- melt(data, id=c("Timestep"))
  ggplot(data = mdata, aes(x=Timestep, y=value, group=variable, color=variable)) +
    geom_line(data = mdata[mdata$variable != "noised",], size=1.2) +
    geom_line(data = mdata[mdata$variable == "noised",], size=.5, linetype='dotdash') +
    scale_color_manual(values=c("KF"='red', "sinusoid"='blue', "noised"='green4'),
                       labels=c("KF"="KF(noised)", "sinusoid"="sinusoid", "noised"="sinusoid with noise
    ggtitle(TeX(sprintf('Kalman Filtering a Sine wave with High Noise, $\\sigma = %.2f$', sd)))
```



It is clear that with high noise, the kalman filter does a poor job of determining the latent signal. A potential thought might be that the filter performs well when we have mostly good data, and a few bad data points, as below:

```r
source('kalman_filter.R')
require('reshape2')
require('ggplot2')
require(latex2exp)
t <- seq(0, 4*pi,,100)  # simple vector
signal <- sin(t)
noise = rnorm(100, mean=0, sd=0.1)
noise[seq(1, 100, by=6)] = rnorm(17, mean=0, sd=.5)
Z <- array(signal + noise, dim=c(100, 1))
filtered <- kalman_filter(Fm=array(1, dim=c(1,1)),Z=Z, H=array(1, dim=c(1,1)),
                          R=array(.01, dim=c(1,1,1)), u=array(0, dim=c(1,1)),
                          B=array(0, dim=c(1,1)), Q=array(.001, dim=c(1,1)),
                          p0=array(.1^2, dim=c(1,1)), x0=array(0, dim=c(1,1)))
data <- data.frame(Timestep = t, KF=filtered$state, sinusoid=signal, noised=Z)
mdata <- melt(data, id=c("Timestep"))
ggplot(data = mdata, aes(x=Timestep, y=value, group=variable, color=variable)) +
  geom_line(data = mdata[mdata$variable != "noised",], size=1.2) +
  geom_line(data = mdata[mdata$variable == "noised",], size=.5, linetype='dotdash') +
  scale_color_manual(values=c("KF"='red', "sinusoid"='blue', "noised"='green4'),
                     labels=c("KF"="KF(noised)", "sinusoid"="sinusoid", "noised"="sinusoid with noise
  ggtitle('Kalman Filtering a Sine Wave with varying noise')
```

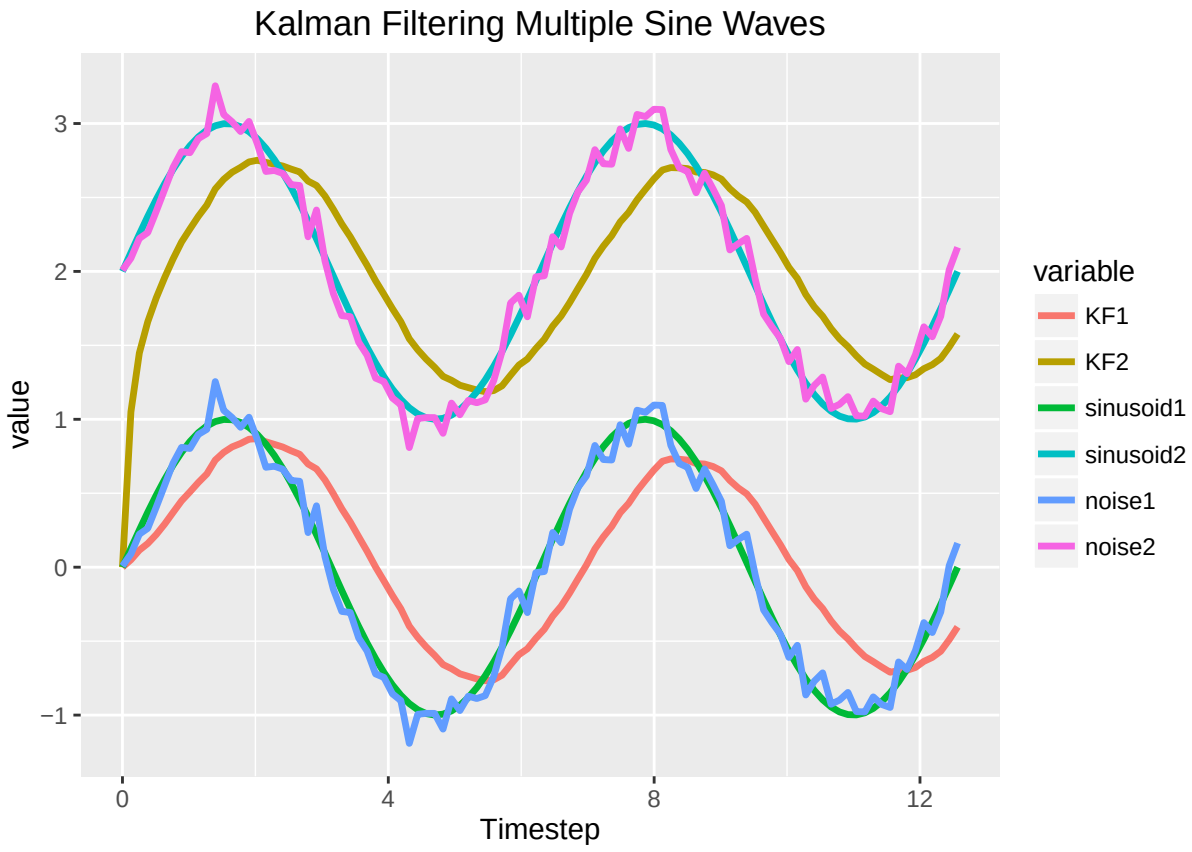


Even when we have a very large number of low deviation points and only a few bad points, it is clear that the Kalman filter still does not accurately predict the latent signal at these time points. Here, we have exposed a limitation of the Kalman filter; it does not respond well to substantial local fluctuations.

Multivariate Case

The below is just an example that our function works with multivariate inputs.

```
source('kalman_filter.R')
require('reshape2')
require('ggplot2')
require(latex2exp)
t <- seq(0, 4*pi,,100)  # simple vector
signal1 <- sin(t)
signal2 <- signal1 + 2
sd <- .1
noise <- rnorm(100, mean=0, sd=sd)
Z <- array(signal1 + noise, dim=c(100, 2))
Z[,2] <- signal2+noise
filtered <- kalman_filter(Fm=diag(2),Z=Z, H=diag(2),
                          R=array(c(sd^2, 0, 0, sd^2), dim=c(1,2,2)), u=array(0, dim=c(1,1)),
                          B=array(0, dim=c(2, 1)), Q=array(.0001, dim=c(1,2)),
                          p0=array(c(sd^2, 0, 0, sd^2), dim=c(2,2)), x0=array(0, dim=c(1,2)))
data <- data.frame(Timestep = t, KF1=filtered$state[,1], KF2=filtered$state[,2],
                   sinusoid1=signal1, sinusoid2=signal2, noise1=signal1 + noise, noise2 = signal2 + no
mdata <- melt(data, id=c("Timestep"))
ggplot(data = mdata, aes(x=Timestep, y=value, group=variable, color=variable)) +
  geom_line(data = mdata[mdata$variable != "noise1" | mdata$variable != "noise2",], size=1.2) +
  geom_line(data = mdata[mdata$variable == "noise1" | mdata$variable == "noise2",], size=.5, linetype=
  ggtitle('Kalman Filtering Multiple Sine Waves')
```

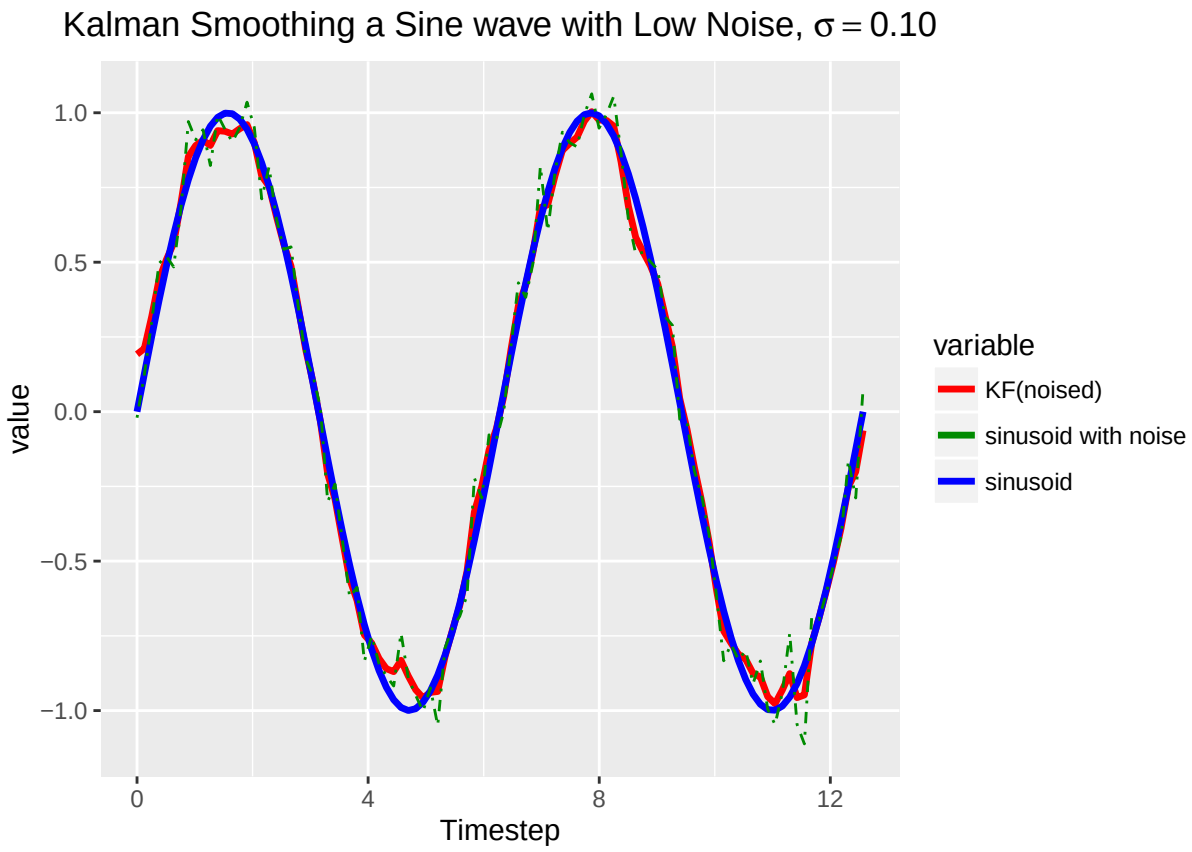


It is clear that this at least recovers something modestly looking like our desired signals. I do not yet understand the parameter choice (especially for multivariate case), and the results vary INCREDIBLY by parameter choice, so we will have to come back to this example after studying EM of the Kalman Filter.

Kalman Smoothing

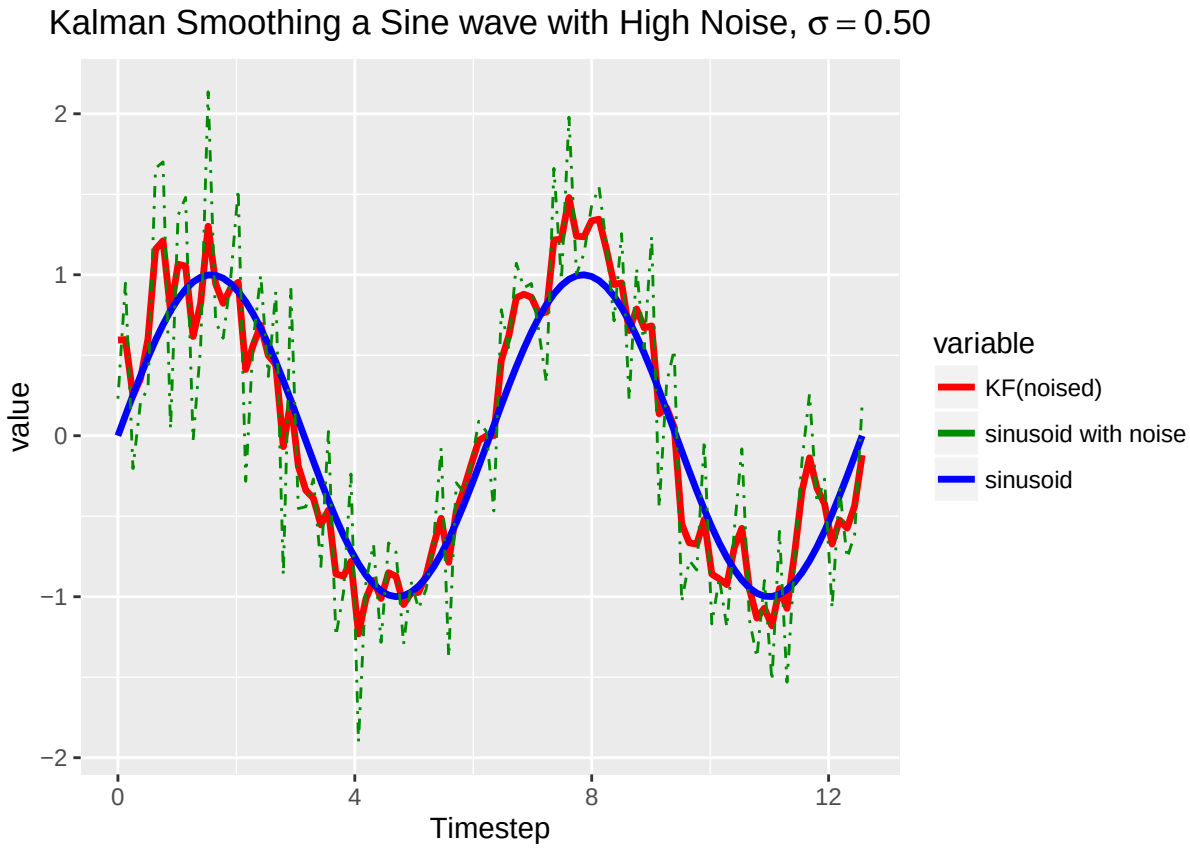
Examples

```
source('kalman_filter.R')
source('kalman_smoother.R')
require('reshape2')
require('ggplot2')
require(latex2exp)
t <- seq(0, 4*pi,,100)  # simple vector
signal <- sin(t)
sd=.1
noise = rnorm(100, mean=0, sd=sd)
Z <- array(signal + noise, dim=c(100, 1))
filtered <- kalman_smoother(Fm=array(1, dim=c(1,1)),Z=Z, H=array(1, dim=c(1,1)),
                            R=array(.001, dim=c(1,1,1)), u=array(0, dim=c(1,1)),
                            B=array(0, dim=c(1,1)), Q=array(.001, dim=c(1,1)),
                            p0=array(sd^2, dim=c(1,1)), x0=array(0, dim=c(1,1)))
data <- data.frame(Timestep = t, KF=filtered$state, sinusoid=signal, noised=Z)
mdata <- melt(data, id=c("Timestep"))
ggplot(data = mdata, aes(x=Timestep, y=value, group=variable, color=variable)) +
  geom_line(data = mdata[mdata$variable != "noised",], size=1.2) +
  geom_line(data = mdata[mdata$variable == "noised",], size=.5, linetype='dotdash') +
  scale_color_manual(values=c("KF"='red', "sinusoid"='blue', "noised"='green4'),
                     labels=c("KF"="KF(noised)", "sinusoid"="sinusoid", "noised"="sinusoid with noise
  ggtitle(TeX(sprintf('Kalman Smoothing a Sine wave with Low Noise, $\\sigma = %.2f$', sd)))
```



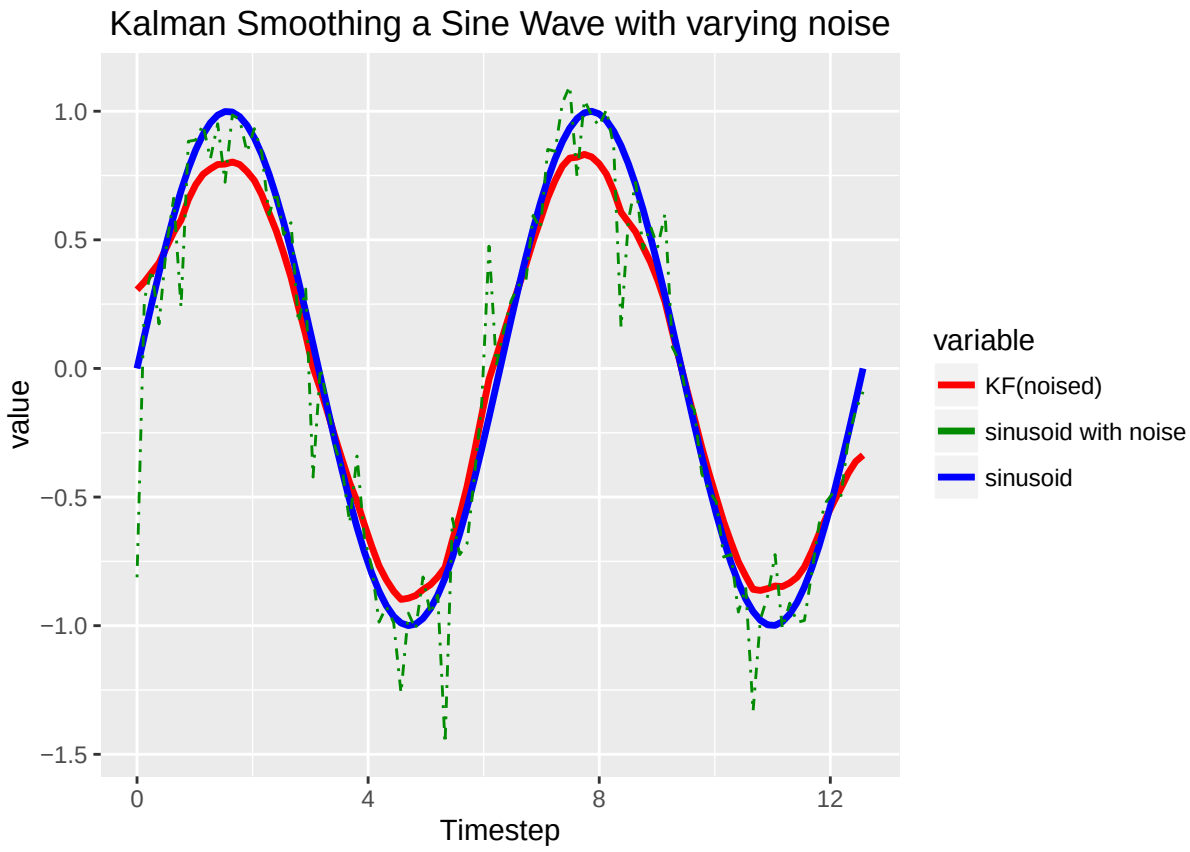
It is clear that the kalman smoothing again does a very good job at determining the latent signal with a low noise component added. Let's try with a larger noise component:

```
source('kalman_filter.R')
source('kalman_smoother.R')
require('reshape2')
require('ggplot2')
require('latex2exp')
t <- seq(0, 4*pi,,100)  # simple vector
signal <- sin(t)
sd = 0.5
noise = rnorm(100, mean=0, sd=0.5)
Z <- array(signal + noise, dim=c(100, 1))
filtered <- kalman_smoother(Fm=array(1, dim=c(1,1)),Z=Z, H=array(1, dim=c(1,1)),
                            R=array(.001, dim=c(1,1,1)), u=array(0, dim=c(1,1)),
                            B=array(0, dim=c(1,1)), Q=array(.001, dim=c(1,1)),
                            p0=array(sd^2, dim=c(1,1)), x0=array(0, dim=c(1,1)))
data <- data.frame(Timestep = t, KF=filtered$state, sinusoid=signal, noised=Z)
mdata <- melt(data, id=c("Timestep"))
ggplot(data = mdata, aes(x=Timestep, y=value, group=variable, color=variable)) +
  geom_line(data = mdata[mdata$variable != "noised",], size=1.2) +
  geom_line(data = mdata[mdata$variable == "noised",], size=.5, linetype='dotdash') +
  scale_color_manual(values=c("KF"='red', "sinusoid"='blue', "noised"='green4'),
                     labels=c("KF"="KF(noised)", "sinusoid"="sinusoid", "noised"="sinusoid with noise
  ggtitle(TeX(sprintf('Kalman Smoothing a Sine wave with High Noise, $\\sigma = %.2f$', sd)))
```



It is clear that with high noise, the kalman smoother doesn't do a terrific job identifying the latent signal, however, it still does a decent job. When we have a mixture of noise (with mostly good points, and some bad ones):

```
source('kalman_filter.R')
source('kalman_smoother.R')
require('reshape2')
require('ggplot2')
require(latex2exp)
t <- seq(0, 4*pi,,100)  # simple vector
signal <- sin(t)
noise = rnorm(100, mean=0, sd=0.1)
noise[seq(1, 100, by=6)] = rnorm(17, mean=0, sd=.5)
Z <- array(signal + noise, dim=c(100, 1))
filtered <- kalman_smoother(Fm=array(1, dim=c(1,1)),Z=Z, H=array(1, dim=c(1,1)),
                            R=array(.01, dim=c(1,1,1)), u=array(0, dim=c(1,1)),
                            B=array(0, dim=c(1,1)), Q=array(.001, dim=c(1,1)),
                            p0=array(.1^2, dim=c(1,1)), x0=array(0, dim=c(1,1)))
data <- data.frame(Timestep = t, KF=filtered$state, sinusoid=signal, noised=Z)
mdata <- melt(data, id=c("Timestep"))
ggplot(data = mdata, aes(x=Timestep, y=value, group=variable, color=variable)) +
  geom_line(data = mdata[mdata$variable != "noised",], size=1.2) +
  geom_line(data = mdata[mdata$variable == "noised",], size=.5, linetype='dotdash') +
  scale_color_manual(values=c("KF"='red', "sinusoid"='blue', "noised"='green4'),
                     labels=c("KF"="KF(noised)", "sinusoid"="sinusoid", "noised"="sinusoid with noise
  ggtitle('Kalman Smoothing a Sine Wave with varying noise')
```



We get a much more favorable result, and we can see that the latent signal is far better estimated.