Lab 4 - Advanced Machine Learning

Gustav Sternelöv

October 11, 2016

Assignment 1

a) - Simulate the model

A linear Guassian state space model is implemented in line with the given instructions. 100 observations are simulated from this model and the obtained states(x) and observations (y) are plotted in figure 1.

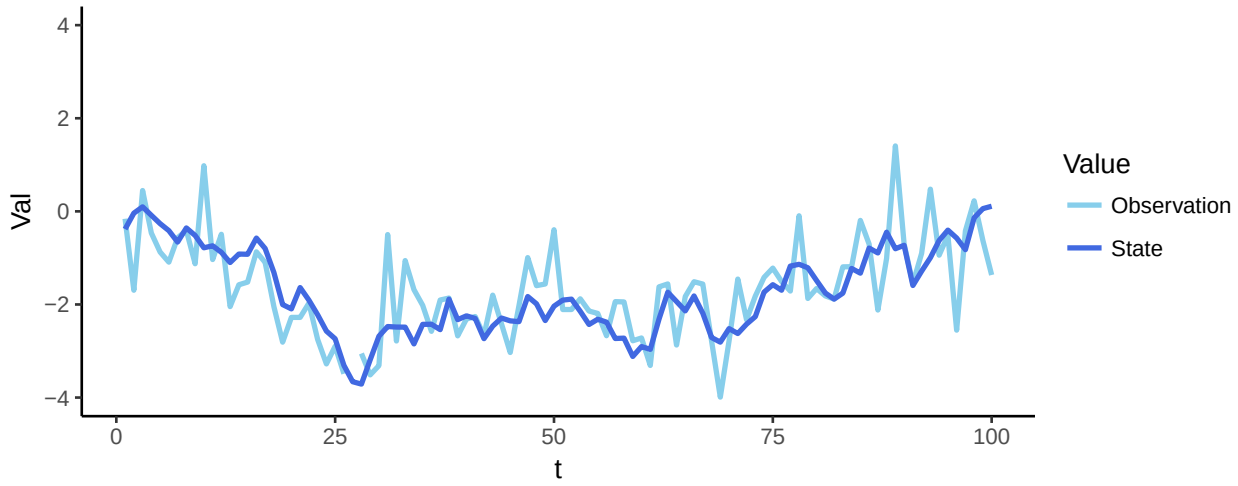

Figure 1

b) - Filtering: Sequential state inference

The point estimates for the Kalman filter and the 0.95 probability bands for the point estimates are plotted together with the states and observations simulated in *a)*.

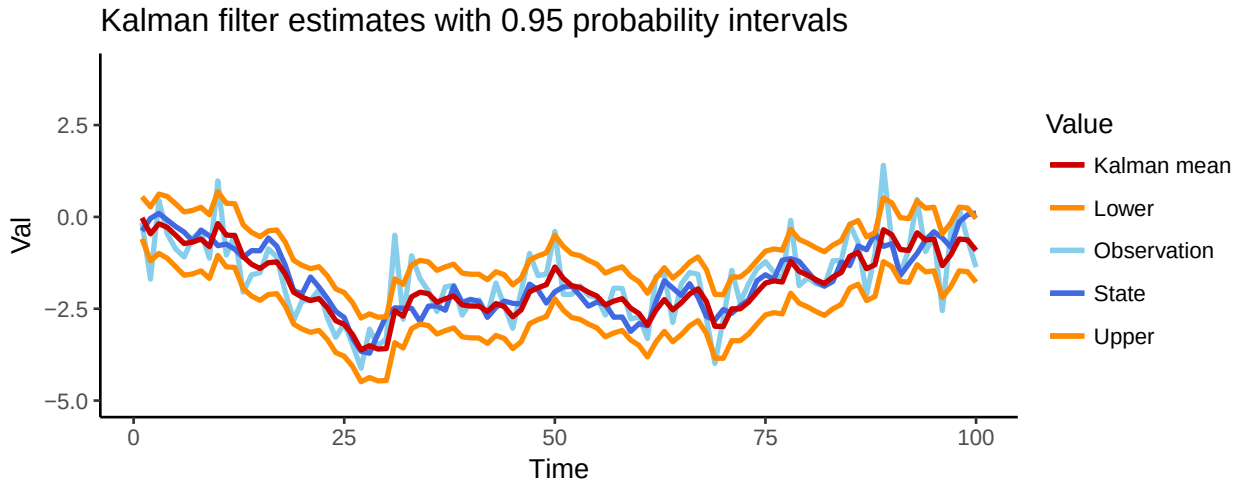

Figure 2

In figure 3 are the Kalman filter estimates and probability intervals obtained by the implemented Kalman algorithm compared to the estimates returned by the *dlmFilter* function from the *dlm* package.

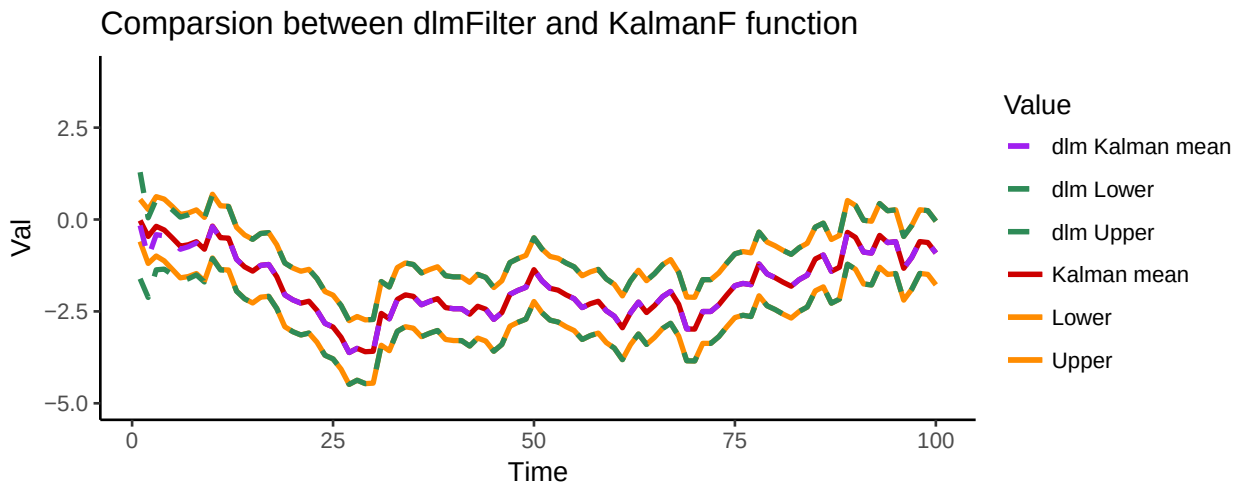

Figure 3

Except for the first values are the point estimates and the probability intervals identical.

c) - Prediction of state and data by simulation

The probability intervals for 5 future states and observations are plotted in figure 2 together with the data plotted in figure 2. The 5 future time steps are simulated 10 000 times and a 0.95 probabilty interval is given by the quantiles from the sample.

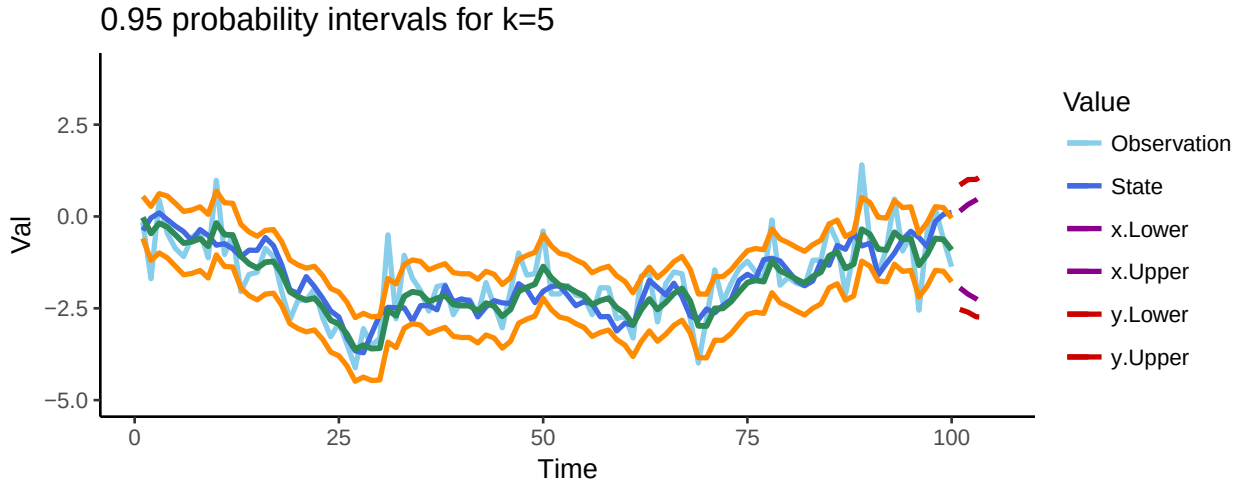

Figure 4

Even though it only is 5 time steps that are predicted it can be seen that the uncertainty grows as the number of steps forward in time increases.

d) - State and model parameter inference

By running a Gibbs sampler and updating θ 5000 times the following results are obtained. The starting value was set to 1 and as seen in both the trace plot and the density plot does not the sampler appears to have been particularly successful.

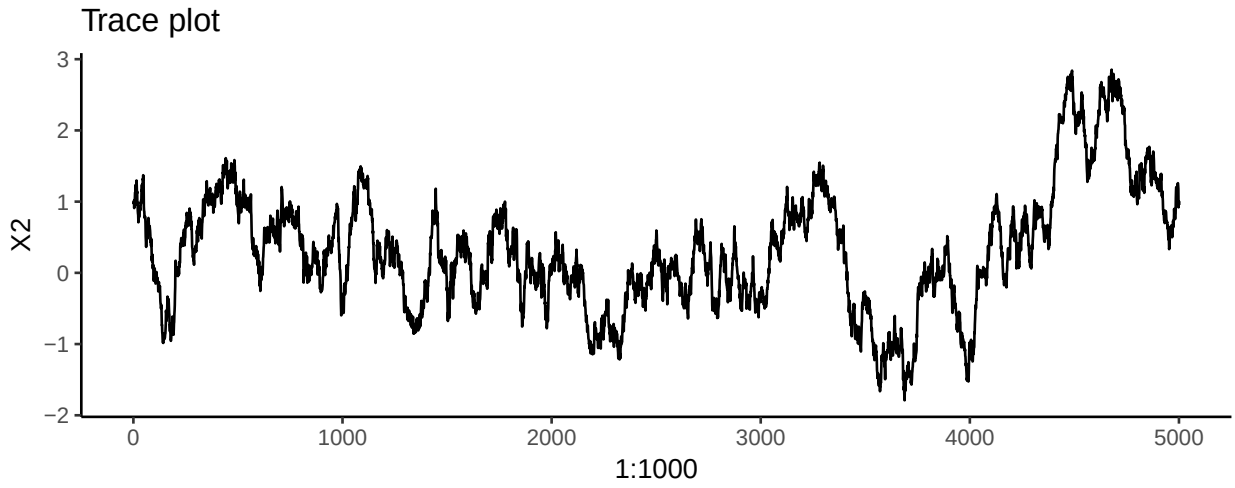

Figure 5

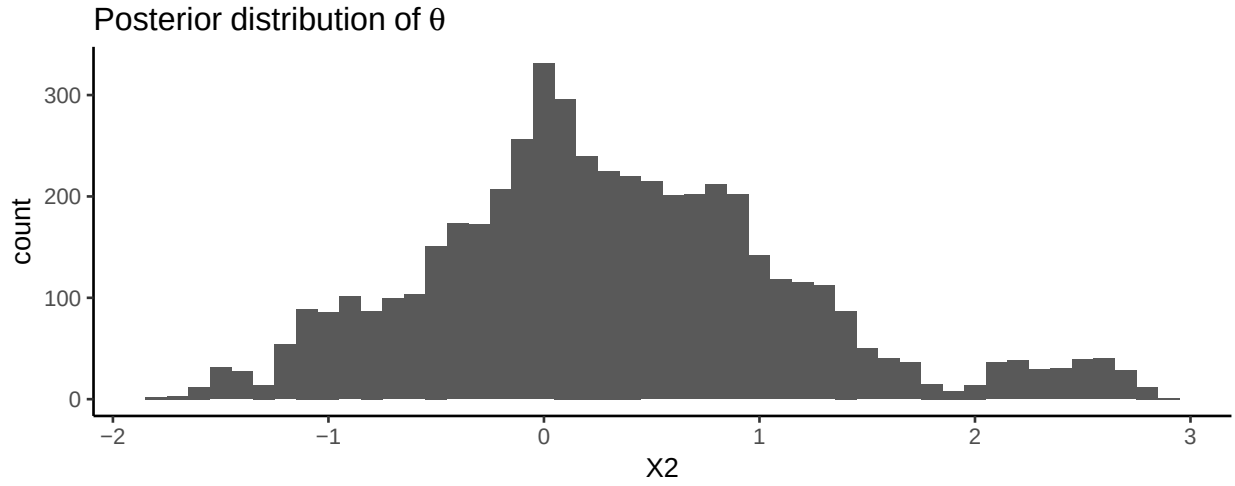

Figure 6

```
## [1] 0.3151664
```

The mean value is 0.3151664 so with the Gibbs sampler implemented here is the true value of θ not estimated accurately. By looking at the plots above you could perhaps suspect that something is wrong in the implementation.

Assignment 2

a) - Estimating the variance components by MLE.

By MLE the following estimates are given for the variance components.

$\sigma^2_\epsilon = 0.0023275$,

$\sigma^2_{v^{(1)}} = 1.1230781 \times 10^{-5}$,

$\sigma^2_{v^{(2)}} = 6.3383481 \times 10^{-4}$

b) - Filtering and smoothing

The filtering and smoothing distribution is computed for α and β. In the figures below are these distributions plotted together with 0.95 probability intervals.

The respective distributions for the alpha parameter are shown in the figures below. As expected, the same pattern is seen in both figures and a more smooth curve is returned for the smoothig distribution.

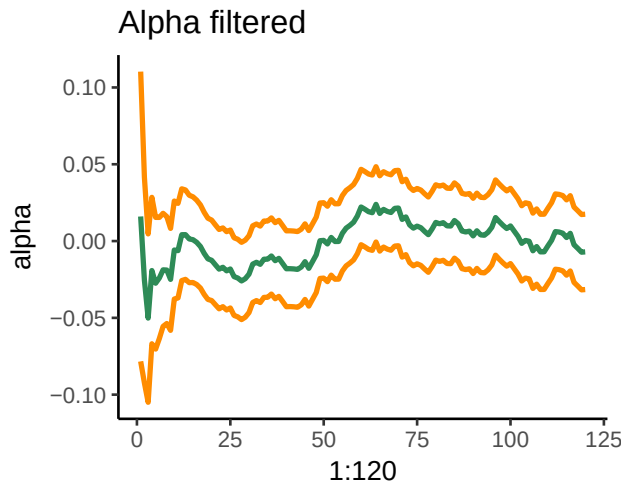

Figure 7

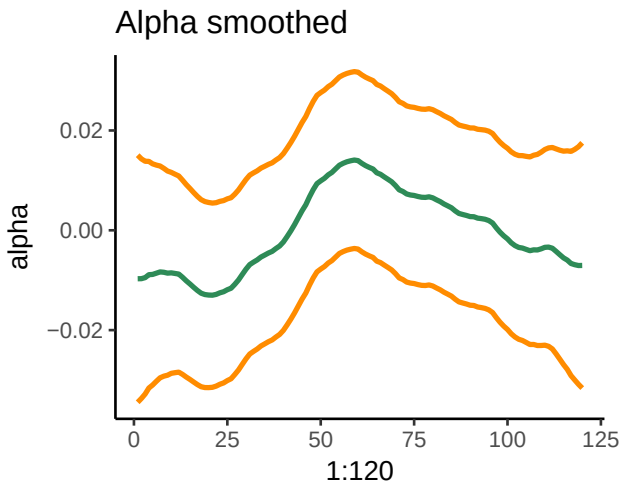

Figure 8

For the beta parameter is the pattern a little bit different for the respective distribution. In figure 10, for the smooth distribution, the value is increasing from around time point 50 and onwards. This pattern was not seen in the figure for the filter distribution.

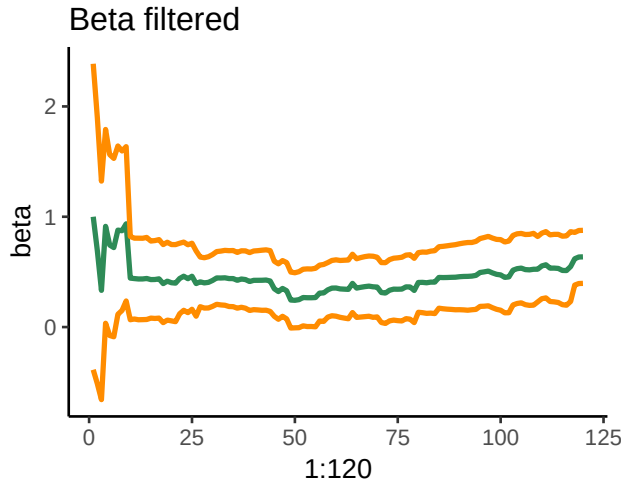

Figure 9

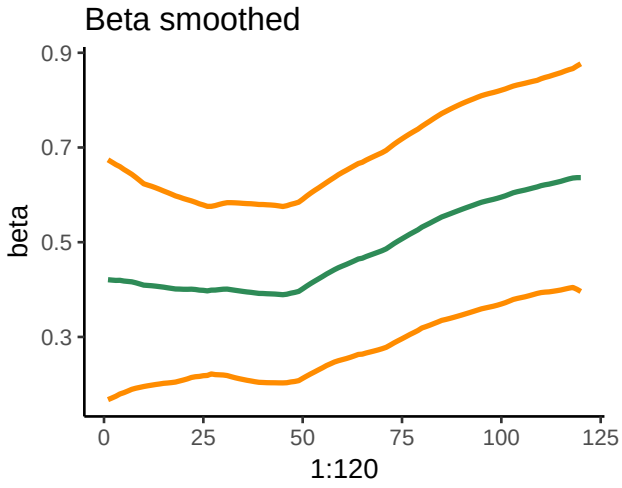

Figure 10

c) - Average sensitivity pre and post 1982 - a retrospective analysis

The retrospective analysis is performed by first computing the smoothing distribution for the time points pre 1982. This is done by only using data for the first 48 observed time steps. Then, the same thing is done once again but for all the data points.

For both smoothing distributions is then the mean and the standard variance calculated and inserted into the *rnorm* function. With this function is a sample of size 10 000 simulated from the specified distribution.

The resulting densities from the samples are plotted in figure 11.

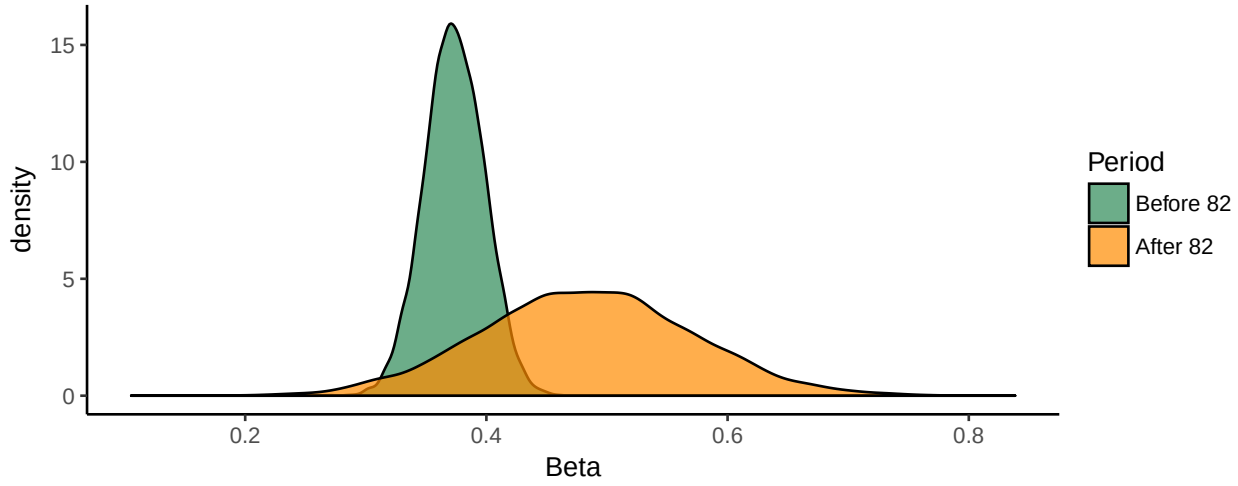

Figure 11

It can easily be seen that on average the sensitivity was higher post 1982 than pre 1982. The majority of the probability mass for the sensitivty post 1982 lies to the right of the probability mass for the sensitivty pre 1982. Therefore, on average, this retrospective analysis gives the conclusion that the sensitivity increased after 1982.

d) - Bayesian inference for the variance components

All of the variance components are unknown parameters that we want to estimate. In order to do bayesian inference on the variance components a possible approach is to set a prior for each component. Then, the next step would be to analyse the point estimates for each component together with the uncertainty for each point estimate. The result of this approach would be an analysis that consider both the estimated value for each component and the uncertainty for these estimations.

Appendix - R-code

```
library(ggplot2)
library(dlm)
library(gridExtra)
library(mvtnorm)
# Assignment 1

# a) - Simulate the model
ssm1 <- function(C = matrix(0), A = matrix(1), B = matrix(0), u = matrix(0),
    x0 = 0, omegaE = 0, omegaV = 0, TimeS = 0) {
    xt <- 0
    yt <- 0
    vt <- rnorm(1, 0, omegaV)
    xt[1] <- A %*% x0 + vt
    et <- rnorm(1, 0, omegaE)
    yt[1] <- C %*% xt + B %*% u + et
    for (i in 2:TimeS) {
```

```
        vt <- rnorm(1, 0, omegaV)
        xt[i] <- A %*% xt[i - 1] + vt
        et <- rnorm(1, 0, omegaE)
        yt[i] <- C %*% xt[i] + B %*% u + et
    }
    res <- data.frame(Val = c(xt, yt), t = rep(1:TimeS, 2), Value = rep(c("State",
        "Observation"), each = TimeS))
    return(res)
}
set.seed(1234)
resA <- ssm1(C = matrix(0.93), x0 = 0, omegaE = sqrt(0.5), omegaV = sqrt(0.1),
    TimeS = 100)
plotA <- ggplot(resA, aes(x = t, y = Val, col = Value)) + geom_line(size = 1) +
    theme_classic() + ylim(-4, 4) + scale_color_manual(values = c("skyblue",
    "royalblue"))
plotA + labs(caption = "Figure 1")
KalmanF <- function(yt, C = matrix(0), A = matrix(1), B = matrix(0), u = matrix(0),
    omegaE = 0, omegaV = 0, TimeS = 0) {
    KalmanMat <- matrix(c(0, 0), ncol = 2, nrow = 101)
    for (i in 1:TimeS) {
        # Prediction update
        muBar_t <- A %*% KalmanMat[i, 1] + B %*% u
        sigmaBar_t <- A %*% KalmanMat[i, 2] %*% t(A) + omegaV
        # Measurement update
        K_t <- sigmaBar_t %*% t(C) %*% solve(C %*% sigmaBar_t %*% t(C) + omegaE)
        KalmanMat[i + 1, 1] <- muBar_t + K_t %*% (yt[i] - C %*% muBar_t)
        KalmanMat[i + 1, 2] <- (diag(ncol(K_t)) - K_t %*% C) %*% sigmaBar_t
    }
    KalmanFrame <- data.frame(KalmanMat[-1, ], Time = 1:TimeS)
    return(KalmanFrame)
}

resB <- KalmanF(yt = resA[101:200, 1], C = matrix(0.93), omegaE = 0.5, omegaV = 0.1,
    TimeS = 100)
resB_1 <- data.frame(Val = c(resB[, 1], resB[, 1] + 1.96 * sqrt(resB[, 2]),
    resB[, 1] - 1.96 * sqrt(resB[, 2])), Time = rep(1:100, 3), Value = rep(c("Kalman mean",
    "Upper", "Lower"), each = 100))

plotA + geom_line(data = resB_1, aes(x = Time, y = Val, col = Value), size = 1) +
    ylim(-5, 4) + scale_colour_manual(values = c("red3", "darkorange", "skyblue",
    "royalblue", "darkorange")) + labs(x = "Time", title = "Kalman filter estimates with 0.95 probabili
    caption = "Figure 2")
filteringB <- dlmFilter(y = resA[101:200, 1], dlm(m0 = 0, V = 0.5, W = 0.1,
    C0 = 10, FF = 0.93, GG = 1))

filterB <- data.frame(y = (filteringB$m)[-1])
filterVar <- unlist(dlmSvd2var(u = filteringB$U.C, d = filteringB$D.C))
filterB[101:300, 1] <- c(filterB$y + 1.96 * sqrt(filterVar[-1]), filterB$y -
    1.96 * sqrt(filterVar[-1]))
filterB$Time <- rep(1:100, 3)
filterB$Value <- rep(c("dlm Kalman mean", "dlm Upper", "dlm Lower"), each = 100)

ggplot(resB_1, aes(x = Time, y = Val, col = Value, linetype = Value)) + geom_line(size = 1) +
```

```
    theme_classic() + ylim(-5, 4) + geom_line(data = filterB, aes(x = Time,
    y = y, col = Value, linetype = Value), size = 1) + scale_colour_manual(values = c("purple",
    "seagreen", "seagreen", "red3", "darkorange", "darkorange")) + scale_linetype_manual(values = c("da
    "dashed", "dashed", "solid", "solid", "solid")) + labs(title = "Comparsion between dlmFilter and Ka
    caption = "Figure 3")
PredFunc <- function(C = matrix(0), A = matrix(1), B = matrix(0), u = matrix(0),
    x0 = 0, omegaE = 0, omegaV = 0, TimeS = 0) {
    xt <- 0
    yt <- 0
    vt <- rnorm(1, 0, omegaV)
    xt[1] <- x0 + vt
    et <- rnorm(1, 0, omegaE)
    yt[1] <- C %*% xt + B %*% u + et
    for (i in 2:TimeS) {
        vt <- rnorm(1, 0, omegaV)
        xt[i] <- A %*% xt[i - 1] + vt
        et <- rnorm(1, 0, omegaE)
        yt[i] <- C %*% xt[i] + B %*% u + et
    }
    res <- data.frame(x = c(xt, y = yt))
    return(res)
}

PredC <- as.data.frame(matrix(nrow = 10, ncol = 10000))
for (i in 1:10000) {
    PredC[, i] <- PredFunc(C = matrix(0.93), x0 = rnorm(1, mean = resB$X1[100],
        sd = sqrt(resB$X2[100])), omegaE = sqrt(0.5), omegaV = sqrt(0.1), TimeS = 5)
}
quanTs <- t(apply(PredC, 1, quantile, probs = c(0.025, 0.975)))
quanTsFrame <- data.frame(x = c(quanTs[, 1], quanTs[, 2]), Time = 100 + rep(1:5,
    4), Var = rep(c("x", "y"), 2, each = 5), InterV = rep(c("Lower", "Upper"),
    each = 10))
quanTsFrame$Interval <- interaction(quanTsFrame$Var, quanTsFrame$InterV)
quanTsFrame$Interval <- factor(quanTsFrame$Interval, levels = rev(levels(quanTsFrame$Interval)))

ggplot(resA, aes(x = t, y = Val, col = Value)) + geom_line(size = 1) + theme_classic() +
    ylim(-5, 4) + geom_line(data = resB_1[1:100, ], aes(x = Time, y = Val),
    col = "seagreen", size = 1.1) + geom_line(data = resB_1[101:200, ], aes(x = Time,
    y = Val), col = "darkorange", size = 1) + geom_line(data = resB_1[201:300,
    ], aes(x = Time, y = Val), col = "darkorange", size = 1) + geom_line(data = quanTsFrame,
    aes(x = Time, y = x, col = Interval), size = 1, linetype = "dashed") + scale_color_manual(values =
    "royalblue", "darkmagenta", "darkmagenta", "red3", "red3")) + labs(x = "Time",
    title = "0.95 probability intervals for k=5", caption = "Figure 4")
KalmanF_2 <- function(yt, C = matrix(0), A = matrix(1), B = matrix(0), u = matrix(0),
    omegaE = 0, omegaV = 0, TimeS = 0) {
    KalmanMat <- matrix(c(0, 0, 0, 0), ncol = 4, nrow = 101)
    for (i in 1:TimeS) {
        # Prediction update
        KalmanMat[i + 1, 3] <- A %*% KalmanMat[i, 1] + B %*% u  #muBar_t
        KalmanMat[i + 1, 4] <- A %*% KalmanMat[i, 2] %*% t(A) + omegaV  #sigmaBar_t
        # Measurement update
        K_t <- KalmanMat[i + 1, 4] %*% t(C) %*% solve(C %*% KalmanMat[i + 1,
            4] %*% t(C) + omegaE)
```

```
        KalmanMat[i + 1, 1] <- KalmanMat[i + 1, 3] + K_t %*% (yt[i] - C %*%
            KalmanMat[i + 1, 3])  #mu_t
        KalmanMat[i + 1, 2] <- (diag(ncol(K_t)) - K_t %*% C) %*% KalmanMat[i +
            1, 4]  # sigma_t
    }
    KalmanFrame <- data.frame(KalmanMat[-1, ], Time = 1:TimeS)
    return(KalmanFrame)
}

BS <- function(A, Kalman, x) {
    x_t <- data.frame(Val = 0)
    for (k in (nrow(Kalman) - 1):1) {
        ht <- Kalman[k, 1] + Kalman[k, 2] %*% t(A) %*% solve(Kalman[k + 1, 4]) %*%
            (x[k + 1] - Kalman[k + 1, 3])
        Ht <- Kalman[k, 2] - Kalman[k, 2] %*% t(A) %*% solve(Kalman[k + 1, 4]) %*%
            A %*% Kalman[k, 2]
        x_t[k, 1] <- rnorm(1, mean = ht, sd = sqrt(Ht))
    }
    return(x_t)
}

bayesReg <- function(beta0, Xt, y, omega0, cov0) {
    betaHat <- solve(t(Xt) %*% Xt) %*% t(Xt) %*% as.matrix(y)
    sigma0 <- diag(omega0/cov0, 2)
    betaNew <- solve(t(Xt) %*% Xt + sigma0) %*% (t(Xt) %*% Xt %*% betaHat +
        sigma0 %*% beta0)
    sigmaNew <- t(Xt) %*% Xt + sigma0
    Vari <- omega0 * solve(sigmaNew)
    coef <- rmvnorm(n = length(y), mean = betaNew, sigma = sqrt(diag(sqrt(diag(Vari)),
        length(betaNew))))
    return(coef)
}

# Gibbs sampling
set.seed(311015)
theta <- data.frame(matrix(ncol = 2, nrow = 5000))  #nrow sets number of sims
theta[1, 2] <- 1  #start value for theta
Xt = matrix(data = c(rep(1, 100), resA[1:100, 1]), ncol = 2)
for (j in 1:nrow(theta)) {
    LGSS <- ssm1(C = matrix(theta[j, 2]), x0 = 0, omegaE = sqrt(0.5), omegaV = sqrt(0.1),
        TimeS = 100)
    y = LGSS[101:200, 1]
    x = LGSS[1:100, 1]
    KalmanRes <- KalmanF_2(yt = y, C = matrix(theta[j, 2]), omegaE = 0.5, omegaV = 0.1,
        TimeS = 100)
    NewX <- (BS(A = matrix(1), Kalman = KalmanRes, x = x))
    Xt[1:99, 2] <- as.numeric(NewX[, 1])
    Xt[100, 2] <- KalmanRes[100, 1]
    coefs <- bayesReg(beta0 = matrix(data = c(0, 0), ncol = 1), Xt = Xt, y = y,
        omega0 = 0.5, cov0 = 100)
    theta[j + 1, ] <- colMeans(coefs)
}
ggplot(theta, aes(y = X2, x = 1:nrow(theta))) + geom_line() + theme_classic() +
```

```
    labs(caption = "Figure 5", title = "Trace plot", x = "1:1000")
ggplot(theta, aes(x = X2)) + geom_histogram(binwidth = 0.1) + labs(caption = "Figure 6",
    title = expression(Posterior ~ distribution ~ of ~ theta)) + theme_classic()
mean(theta$X2)
# Assignment 2
load("C:\\Users\\Gustav\\Documents\\AdvML\\Lab4\\CAPM.Rda")

# a) - Estimating the variance components by MLE

CAPMv <- CAPM_data[, c(10, 17, 18)]
Zt <- matrix(CAPM_data$MARKET - CAPM_data$RKFREE)
Yt <- matrix(CAPMv$IBM - CAPMv$RKFREE)

buildLocalTrend <- function(x, data) {
    V = exp(x[1])
    W = diag(exp(x[2:3]), 2, 2)
    return(dlm(m0 = c(0, 1), C0 = diag(c(100, 0.5), 2), FF = matrix(c(1, 1),
        1, 2), GG = diag(2), JFF = matrix(c(0, 1), 1, 2), V = V, W = W, X = data))
}

initVal <- c(1, 1, 1)  # Initial values for optim on the estimated parameters
MLEs <- dlmMLE(Yt, parm = initVal, build = buildLocalTrend, data = Zt)
dlmWithMLEs <- buildLocalTrend(MLEs$par, data = Zt)

## Filtering ##
Filter_2b <- dlmFilter(y = Yt, dlmWithMLEs)

# Variances
u1 <- matrix((t(data.frame(Filter_2b$U.C)))[c(seq(1, 242, 2)), 1])
u2 <- matrix((t(data.frame(Filter_2b$U.C)))[c(seq(2, 242, 2)), 2])
u_1 <- list()
u_2 <- list()
for (i in 1:120) {
    u_1[i] <- c(u1[i + 1])
    u_2[i] <- c(u2[i + 1])
}
Filter_y <- data.frame(alpha = c(Filter_2b$m)[2:121], beta = c(Filter_2b$m)[123:242])
Filter_y$alphaVar <- unlist(dlmSvd2var(u = u_1, d = matrix(Filter_2b$D.C[-1,
    1])))
Filter_y$betaVar <- unlist(dlmSvd2var(u = u_2, d = matrix(Filter_2b$D.C[-1,
    2])))
# Intervals
Filter_y$alphaLower <- Filter_y$alpha - 1.96 * sqrt(Filter_y$alphaVar)
Filter_y$alphaUpper <- Filter_y$alpha + 1.96 * sqrt(Filter_y$alphaVar)
Filter_y$betaLower <- Filter_y$beta - 1.96 * sqrt(Filter_y$betaVar)
Filter_y$betaUpper <- Filter_y$beta + 1.96 * sqrt(Filter_y$betaVar)

alphaFilter <- ggplot(Filter_y, aes(x = 1:120, y = alpha)) + geom_line(col = "seagreen",
    size = 1) + theme_classic() + labs(title = "Alpha filtered", caption = "Figure 7") +
    geom_line(data = Filter_y, aes(x = 1:120, y = alphaLower), col = "darkorange",
        size = 1) + geom_line(data = Filter_y, aes(x = 1:120, y = alphaUpper),
    col = "darkorange", size = 1)
```

```
betaFilter <- ggplot(Filter_y, aes(x = 1:120, y = beta)) + geom_line(col = "seagreen",
    size = 1) + theme_classic() + labs(title = "Beta filtered", caption = "Figure 9") +
    geom_line(data = Filter_y, aes(x = 1:120, y = betaLower), col = "darkorange",
        size = 1) + geom_line(data = Filter_y, aes(x = 1:120, y = betaUpper),
    col = "darkorange", size = 1)

#### Smoothed ####
Smooth_2b <- dlmSmooth(y = Yt, dlmWithMLEs)
Smooth_frame <- data.frame(alpha = Smooth_2b$s[-1, 1], beta = Smooth_2b$s[-1,
    2])

covList = dlmSvd2var(Smooth_2b$U.S, Smooth_2b$D.S)
varMat = t(sapply(covList, FUN = function(x) sqrt(diag(x))))

Smooth_frame$alphaSd <- varMat[-1, 1]
Smooth_frame$betaSd <- varMat[-1, 2]
# Intervals
Smooth_frame$alphaLower <- Smooth_frame$alpha - 1.96 * Smooth_frame$alphaSd
Smooth_frame$alphaUpper <- Smooth_frame$alpha + 1.96 * Smooth_frame$alphaSd
Smooth_frame$betaLower <- Smooth_frame$beta - 1.96 * Smooth_frame$betaSd
Smooth_frame$betaUpper <- Smooth_frame$beta + 1.96 * Smooth_frame$betaSd

alphaSmooth <- ggplot(Smooth_frame, aes(x = 1:120, y = alpha)) + geom_line(size = 1,
    col = "seagreen") + theme_classic() + labs(title = "Alpha smoothed", caption = "Figure 8") +
    geom_line(data = Smooth_frame, aes(x = 1:120, y = alphaLower), size = 1,
        col = "darkorange") + geom_line(data = Smooth_frame, aes(x = 1:120,
    y = alphaUpper), size = 1, col = "darkorange")

betaSmooth <- ggplot(Smooth_frame, aes(x = 1:120, y = beta)) + geom_line(size = 1,
    col = "seagreen") + theme_classic() + labs(title = "Beta smoothed", caption = "Figure 10") +
    geom_line(data = Smooth_frame, aes(x = 1:120, y = betaLower), size = 1,
        col = "darkorange") + geom_line(data = Smooth_frame, aes(x = 1:120,
    y = betaUpper), size = 1, col = "darkorange")

grid.arrange(alphaFilter, alphaSmooth, ncol = 2)
grid.arrange(betaFilter, betaSmooth, ncol = 2)
# c)
TimeB <- data.frame(Beta = (dlmSmooth(y = Yt[1:48], dlmWithMLEs)$s)[, 2])
TimeA <- data.frame(Beta = (dlmSmooth(y = Yt[1:120], dlmWithMLEs)$s)[, 2])

Sensitivity <- data.frame(Beta = c(rnorm(10000, mean(TimeB$Beta), sd(TimeB$Beta)),
    rnorm(10000, mean(TimeA$Beta), sd(TimeA$Beta))), Period = rep(c("Before 82",
    "After 82"), each = 10000))
Sensitivity$Period <- factor(Sensitivity$Period, levels = rev(levels(Sensitivity$Period)))

ggplot(Sensitivity, aes(x = Beta)) + geom_density(aes(y = ..density.., fill = Period),
    alpha = 0.7) + scale_fill_manual(values = c("seagreen", "darkorange")) +
    theme_classic() + labs(caption = "Figure 11")
## NA
```