H2020-SFS-2018-2020

DECIDE

Data-driven control and prioritisation of non-EU-regulated contagious animal diseases

Deliverable D2.2

Open-source software tools to perform multi-level monitoring of time-series data

WP2 – Methods for data analysis and modelling

Authors Carolina Merca (UCPH),
Leonardo Víctor de Knegt (UCPH),
Dan Børge Jensen (UCPH),
Anders Kristensen (UCPH)
Lead participant UCPH
Delivery date 27 December 2023
Dissemination level Public
Type Other

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 101000494.

Revision History

Author Name (Partner short name)	Description	Date
Carolina Merca (UCPH), Leonardo Víctor de Knegt (UCPH), Dan Børge Jensen (UCPH), Anders Kristensen (UCPH)	Draft deliverable report	05.12.2023
Leonardo de Knegt (UCPH)	Upload of code	05.12.2023
Céline Faverjon (AUSVET)	Revision 1	13.12.2023
Anders Kristensen (UCPH)	Version 2	19.12.2023
Johannes Ripperger (accelCH)	Final checks and formatting of deliverable report	27.12.2023
Leonardo Victor de Knegt	Upload of revised version	

Contents

Abbreviations

Abbreviation	Description
CI	Credible interval
DLM	Dynamic linear model
EU	European Union
WP	Work package

Partner short names

Short name	Organisation
UCPH	Københavns Universitet
accelCH	accelopment Schweiz AG
AUSVET	Ausvet Europe

Executive Summary

Data collection and its use to detect trends and changes are a vital part of any animal health surveillance system. In that sense, time-series models provide a framework that allows for a better understanding of different situations, and supports informed decision-making. Such models are typically based on estimating an expected underlying value for a parameter (e.g. milk yield or calf mortality), as well as the expected variation around it under normal circumstances, so the system can be monitored over time, and deviations can be quickly detected. State space models like Dynamic Linear Models (DLM) are a particular type of time series models building on a Bayesian framework to estimate underlying parameters from current observations, while incorporating measurement errors, systematic fluctuations and prior information from data. Software scripts for dealing with DLMs (univariate as well as multivariate) were presented previously as Deliverable D2.1. The present deliverable extends the framework to also cover multi-level models where a hierarchical structure is taken into account.

Objectives of the Deliverable

With the help of this deliverable, the reader will walk through an example approach to create a multi-level DLM in R, with the final objective of detecting increased mortality levels in salmon. Pre-programmed functions and R scripts needed to run such models are also provided, and can be found in the same repositories as this report.

Activities

The deliverable defines the concept of multi-level DLMs and illustrates how the necessary matrices and variance components are specified so that they reflect the hierarchy. Furthermore, it uses anonymised data for Salmon mortality to illustrate how software scripts can be applied for estimation of variance components and running the multi-level DLM.

The present deliverable was based on R codes written by Carolina Merca.

Outcome

With the R scripts containing the walkthrough and the functions used to define a multi-level DLM for salmon mortality, the reader should be able to develop their own DLMs for monitoring of their own hierarchical systems.

1 Introduction

1.1 Background

Animal health surveillance requires a continuous stream of data, as well as a framework that allows knowledge from information previously acquired to accumulate, leading to an improved understanding of the present situation and the detection of apparent trends or unexpected changes. State-space models offer such a framework, in which relevant prior knowledge and current information are combined to detect changes in an observed process, thus allowing for a better understanding of the situation, and resulting in better-informed decisions.

Classical monitoring methods like the Shewhart Control Chart, Moving Average or Exponentially Weighted Moving Averages assume that the true underlying mean is constant over time, but this assumption is often dubious, since values can often suffer some random fluctuations, or vary over the day or according to seasons, or just systematically increase or decrease over time. Thus, a more elaborate time series approach is needed.

A type of system is required, which allows the estimation of an observation to be based on the true underlying mean and its measurement or observation errors, but also on the dynamic aspects of the parameter, meaning its systematic fluctuations and changes over time. Therefore, a state-space model is normally defined by two equations: an observation equation describing the data observed by a vector of parameters, and a system equation describing the dynamics of the parameter vector as a first order autocorrelation model.

Through the two equations, domain specific knowledge about the system can be taken into account. Thus, components of the system equation may include, for example, fluctuations in water drinking during the day, lactation curves, disease seasonality or positive weight gain linear trends. Specifically, for multi-level models, the hierarchical structure of the problem can be directly represented in the matrices of the two equations as elaborated below.

In animal production and health monitoring, the most commonly used type of state space models is Dynamic Linear Models (DLM). A DLM uses a Bayesian framework to estimate the underlying parameter vector from the observed data, while taking into account any prior information available before the observations are done. Values are forecast at each time-step, based on the theoretical true mean and prior knowledge on error and variance around the system and the data, and are then updated on the next time-step, being "corrected" according to each new observation, as well as to the error and variance components already mentioned.

Multi-level models (also referred to as hierarchical models) are used when several units (animals, pens, herds) organised in a hierarchical structure are being monitored simultaneously. The motivation for using a multi-level approach is to utilise the hierarchical structure directly in order to account for correlations between units and/or to enable monitoring at multiple levels. Examples of such hierarchical structures are country/region/herd or herd/section/pen.

Since multi-level models simultaneously handle responses (e.g., performance or disease prevalence) from multiple units, they are inherently multivariate. Still, such models are defined within the usual framework of a DLM with an observation equation and a system equation. The hierarchical structure of the model is just reflected in the structure of the matrices of those equations. Thus, software scripts developed for multivariate models can also be applied directly to multi-level models.

1.2 Previous work leading to this deliverable

In 2022, the University of Copenhagen (UCPH) held a one-week workshop on Dynamic Linear Models for partners of the DECIDE Project, as part of WP2 and as a precursor for the first deliverable (D2.1) of WP2. The workshop focused on the development of multivariate DLMs in R, for monitoring and detection of state changes in data series. The subsequent deliverable was submitted in the summer of 2023.

In September 2023, a second one-week workshop was held in Copenhagen as a precursor for the present second deliverable (D2.2) with focus on multi-level monitoring of time series data using DLMs. Theoretical lectures were held in the morning, and in the afternoon, the participants worked on writing models for their own data, with support from the UCPH DECIDE partner.

1.3 Objectives and content of the deliverable

Since general software scripts developed for multivariate models (i.e., Deliverable D2.1) can be used also for multi-level models, the focus is on how to take the hierarchical structure of the problem into account when the multi-level model is specified as a DLM. This is illustrated as a case study, where the example situation used is the monitoring of Scottish Salmon mortality.

The present deliverable includes a report, an anonymised dataset and two R scripts produced by Carolina Merca.

The script "*Multi-level DLM walkthrough.R*" walks the user through a scenario where a multi-level DLM can be applied, with functions and code lines written in a generalized format facilitating their adaptation to diverse datasets. It is needed that the data is in the correct format (data example provided "df_salmon.RData") and the user has a working knowledge of the methods used. The functions used are contained in the script "*Functions - Multi-level DLM.R*".

All three (2 R scripts + dataset) can be found on the DECIDE GitHub repository, at:

https://github.com/decide-project-eu/WP2-Deliverables

and also on the project internal document storage system (accelCLOUD) at:

https://cloud.accelopment.com/index.php/apps/files/?dir=/DECIDE/Deliverables/Submitted/D2.2_SoftwareHierarchicalMonitoring

Users can get support when using the scripts by contacting the UCPH group at cmgm@sund.ku.dk, and if any relevant developments and updates to the codes occur, the most recent versions will be uploaded to both repositories.

2 Mathematical description of a simplified hierarchical model

Consider a simplified framework, where a country consists of only two regions (A and B) each having two production sites (1 and 2). Monthly data (e.g., mortality or performance) at site level are available. We will build a model where all three levels (country, region and site) are directly represented. For the country and region levels, no time trend is assumed, whereas the site level may change according to a trend accounting for the stage of the production cycle (i.e., the age of the animals at the site). To model the trend, a spline function is used to provide the desired shape over the production cycle. The magnitude of the trend is estimated dynamically as a *trend factor*.

Now, let y_{rit} denote the observed value (e.g. mortality or performance) in month t for site i in region r. A direct way of modelling the hierarchy (country/region/site) is through a model like the following:

$$y_{rit} = \mu_t + \alpha_{rt} + \beta_{rit} + v_{rit}, \qquad v_{rit} \sim N(0, \sigma^2),$$

where μ_t is a dynamic country level, α_{rt} is a dynamic region level, β_{rit} is a dynamic site level, and v_{rit} is a random observation error. Thus, changes over time can happen at any level (country, region or site).

Let $y_t = (y_{A1t}, y_{A2t}, y_{B1t}, y_{B2t})'$ and $v_t = (v_{A1t}, v_{A2t}, v_{B1t}, v_{B2t})'$ be vectors of all site level observations and observation errors, respectively, at month t. Then the observation equation can be written in matrix notation as

$$y_t = F_t'\theta_t + v_t, \qquad v_t \sim N(\underline{0}, V),$$

where

$$F_t' = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix},$$

and

$$\theta_t = (\mu_t, \alpha_{At}, \alpha_{Bt}, \beta_{A1t}, \delta_{A1t}, \beta_{A2t}, \delta_{A2t}, \beta_{B1t}, \delta_{B1t}, \beta_{B2t}, \delta_{B2t})$$

where δ_{rit} denotes the trend factor for site i in region r at month t (depending of the stage of the production cycle, i.e., the age of the animals at the site).

This model already takes the hierarchy into account by seeing an observation as result of effects at all three levels. It may, however, also be natural to assume that the error terms $v_{A1t}, v_{A2t}, v_{B1t}, v_{B2t}$ are mutually correlated. This is achieved if the individual error term is seen as the sum of three underlying independent errors at country, region and site level, respectively, so that (where indexes refer to country (c), region (r) and site (i))

$$v_{rit} = v_{ct} + v_{rt} + v_{it}, \quad v_{ct} \sim N(0, \sigma_C^2), \quad v_{rt} \sim N(0, \sigma_R^2), \quad v_{it} \sim N(0, \sigma_I^2).$$

Assuming independence, the variance-covariance matrix V of v_t has the following form:

$$V = \begin{bmatrix} \sigma_C^2 + \sigma_R^2 + \sigma_I^2 & \sigma_C^2 + \sigma_R^2 & \sigma_C^2 & \sigma_C^2 \\ \sigma_C^2 + \sigma_R^2 & \sigma_C^2 + \sigma_R^2 + \sigma_I^2 & \sigma_C^2 & \sigma_C^2 \\ \sigma_C^2 & \sigma_C^2 & \sigma_C^2 + \sigma_R^2 + \sigma_I^2 & \sigma_C^2 + \sigma_R^2 \\ \sigma_C^2 & \sigma_C^2 & \sigma_C^2 + \sigma_R^2 & \sigma_C^2 + \sigma_R^2 + \sigma_I^2 \end{bmatrix}.$$

The system equation of the hierarchical DLM is

$$\theta_t = G_t\theta_{t-1} + w_t, \qquad w_t \sim N(\underline{0}, W_t),$$

where

$$G_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta_{A1t} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & \Delta_{A2t} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta_{B1t} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta_{B2t} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

In the above matrix, Δ_{Rit} denotes the increase of the spline function from month $t-1$ to t.

A component discounting approach is used to calculate W_t, the system variance-covariance matrix, for month t using a technique described by West & Harrison (1997)[1].

A discount factor is a number between 0 and 1. Values close to 0 reflect large system variance and values close to 1 reflect small variance. It is called component discounting because it allows each level (component) to have its own discount factor. Thus, the model specification allows for instance the country and regional levels to fluctuate less than the site level.

Thus, for a full specification of the variance components of this hierarchical model, six parameters are needed: the three observational variances $\sigma_C^2, \sigma_R^2, \sigma_I^2$ and the three discount factors ρ_C, ρ_R, ρ_I.

The six parameters are estimated from a learning data set as the values minimising the sum of squares of forecast errors. The built-in function `optim` in R can be used for this estimation as illustrated in the following section using the real anonymised data from the Scottish Salmon sites.

In addition to the variance components, also the initial distribution $\theta_0 \sim N(m_0, C_0)$ of the parameter vector θ_0 before any observations are done needs to be specified through the initial mean m_0 and the initial variance-covariance matrix C_0.

3 Case study: Scottish Salmon mortality

3.1 Presentation

As explained in the introduction, the contents of the present case study section refer to the scripts named "*Multi-level DLM walkthrough.R*" and "*Functions - Multi-level DLM.R*". An anonymised dataset called "*df_salmon.RData*" is provided. The processes described here can be followed step by step by running or adapting those codes.

The script will be applied to Scottish Salmon data, with anonymised site/farm names and regions. The dataset contains mortality and environmental data. Our goal is to develop a multi-level model using only the mortality data.

The dataset contains monthly information between 2002 and 2020. It consists of data about 5 regions (column "local.authority"), 293 sites (column "site") and 1610 production cycles (column "nseq"). A production cycle refers to a site-level period in which at least one pen on a site is occupied consecutively. Between different production cycles is usual to fallow a site (empty period) for around 4 weeks.

3.2 Data preparation

The data preparation step will typically be very case dependent because the format of the raw data typically differs between cases. Thus, the scripts presented below will typically need considerable modification in order to be used in another case.

Nevertheless, the end goals of data preparation are typically:

- To test the preconditions for using a DLM (error terms following a normal distribution) and, if necessary, transform the data to meet the conditions (e.g., log transform).
- To standardise data. If only one variable is observed at each site, this step can be skipped, but if several, very different, variables are observed, it is highly recommended. The standardisation involves subtracting the sample mean and, afterwards, dividing by the sample standard deviation.

[1] West, M. & J. Harrison. 1997. *Bayesian Forecasting and Dynamic Models.* Second Edition. Springer, New York.

- To split the data between a learning set for estimation of variance components and spline functions and a test set for evaluation of the performance of the model. Usually, it is recommended to include around 75% of the data in a learning set.
- To have data frames where each row corresponds to a vector of observations as described below.

The first step of the case is to load the anonymized dataset:

```
load("df_salmon.RData")
```

Source the script containing the pre-programmed functions for the DLM:

```
source("Functions - Multi-level DLM.R")
```

First, the dataset was split between the Learning set (to train the model) and the Test set (to apply the model). This division was done by dividing the dataset into four parts, being the first ¾ of the data the learning set and the rest the test set. As this division would compulsorily cut production cycles, it was needed to have different cut-off points for each site, ensuring the integrity of each production cycle. Therefore, an adjustment was made in order for each cut-off point in each site to be as close to the ¾ reference as possible. Only the sites present in both Learning and Test sets were kept. In this code, those files are called "Learning.set.i" and "Test.set.i", respectively.

```
N <- round(3*(length(unique(df[order(df[,"date"]), "date"]))/4))
sets <- get.learning.test.sets(df,
     relevant.names="log0.mortality.rel.20", hierarchical=TRUE, N=N)
Learning.set.i <- sets[["Learning.set"]]
Test.set.i <- sets[["Test.set"]]
```

Then, both Learning and Test sets structures had to be changed in order to be possible to apply the multi-level DLM. The new structure should be: each row corresponds to a date and each column corresponds to a site name. Each cell has the log-transformed value of mortality (variable "log0.mortality.rel.20" in the initial dataset) for the correspondent date-site. To not lose the region each site belongs to, the column names were saved as "SiteName_RegionName". We call it "Learning.set" and "Test.set", respectively.

Note: The log-transformed values of mortality were used instead of the raw mortality values because in DLMs the data has to follow a normal distribution. That was not the case for the mortality data, therefore, we log-transformed them to be closer to a normal distribution.

```
#Learning.set
col.needed <- Learning.set.i[,c(1,2,3,11)]
col.needed <- col.needed %>% arrange(local.authority)
col.needed[,"site_region"] <- paste(col.needed$site,
col.needed$local.authority, sep="_")
col.needed <- col.needed[,c(1,4,5)]
Learning.set <- as.data.frame(matrix(NA, length(unique(col.needed$date)),
length(unique(col.needed$site_region))+1))
colnames(Learning.set) <- c("date", unique(col.needed$site_region))
Learning.set$date <- unique(col.needed$date)
Learning.set <- Learning.set %>% arrange(date)
for (i in 1:dim(col.needed)[1]){
  Learning.set[which(Learning.set$date == col.needed[i,"date"]),
    col.needed[i,"site_region"]] <- col.needed[i,"log0.mortality.rel.20"]
}
```

```
#Test.set
col.needed <- Test.set.i[,c(1,2,3,11)]
col.needed <- col.needed %>% arrange(local.authority)
col.needed[,"site_region"] <- paste(col.needed$site,
col.needed$local.authority, sep="_")
col.needed <- col.needed[,c(1,4,5)]
Test.set <- as.data.frame(matrix(NA, length(unique(col.needed$date)),
length(unique(col.needed$site_region))+1))
colnames(Test.set) <- c("date", unique(col.needed$site_region))
Test.set$date <- unique(col.needed$date)
Test.set <- Test.set %>%  arrange(date)
for (i in 1:dim(col.needed)[1]){
  Test.set[which(Test.set$date == col.needed[i,"date"]),
    col.needed[i,"site_region"]] <- col.needed[i,"log0.mortality.rel.20"]
}
```

By creating the Learning and Test sets with this different structure, some important data was lost. This is specific for this case study, because we intend to use age dependent spline functions. Therefore, four other data frames with information about when each production cycle started and the identification number of each production cycle were created with the same structure as the Learning and Test sets. In this code, those files are called "months.learning.set", "months.test.set", "nseq.learning.set" and "nseq.test.set".

```
### Create a table with information about when each production cycle
### started
#Learning.set
months.learning.set <- data.frame(matrix(NA, nrow = dim(Learning.set)[1],
      ncol = dim(Learning.set)[2]))
months.learning.set[,1] <- Learning.set$date
colnames(months.learning.set) <- colnames(Learning.set)
for (farm in unique(Learning.set.i$site)){
  df.farm <- subset(Learning.set.i, Learning.set.i$site == farm)
  for(date in unique(df.farm$date)){
    if(is.na(df.farm[which(df.farm$date==date), "months.since.start"])){
      #if fallow period (NaN)
       months.learning.set[which(months.learning.set$date==date),
            paste(farm, unique(df.farm$local.authority), sep="_")] <- NaN
    }else{
      months.learning.set[which(months.learning.set$date==date),
          paste(farm, unique(df.farm$local.authority), sep="_")] <-
          df.farm[which(df.farm$date==date), "months.since.start"]
    } } }

#Test.set
months.test.set <- data.frame(matrix(NA, nrow = dim(Test.set)[1],
      ncol = dim(Test.set)[2]))
months.test.set[,1] <- Test.set$date
colnames(months.test.set) <- colnames(Test.set)
for (farm in unique(Test.set.i$site)){
  df.farm <- subset(Test.set.i, Test.set.i$site == farm)
  for(date in unique(df.farm$date)){
```

```
    if(is.na(df.farm[which(df.farm$date==date), "months.since.start"])){
      #if fallow period (NaN)
      months.test.set[which(months.test.set$date==date),
          paste(farm, unique(df.farm$local.authority), sep="_")] <- NaN
    }else{
      months.test.set[which(months.test.set$date==date),
          paste(farm, unique(df.farm$local.authority), sep="_")] <-
          df.farm[which(df.farm$date==date), "months.since.start"]
    } } }

### Create a table with information about the identification number of
### the production cycle
#Learning.set
nseq.learning.set <- data.frame(matrix(NA, nrow = dim(Learning.set)[1],
ncol = dim(Learning.set)[2]))
nseq.learning.set[,1] <- Learning.set$date
colnames(nseq.learning.set) <- colnames(Learning.set)
for (farm in unique(Learning.set.i$site)){
  df.farm <- subset(Learning.set.i, Learning.set.i$site == farm)
  for(date in unique(df.farm$date)){
    if(is.na(df.farm[which(df.farm$date==date), "months.since.start"])){
      #if fallow period (NaN)
      nseq.learning.set[which(nseq.learning.set$date==date),
            paste(farm, unique(df.farm$local.authority), sep="_")] <- NaN
    }else{
      nseq.learning.set[which(nseq.learning.set$date==date),
          paste(farm, unique(df.farm$local.authority), sep="_")] <-
          df.farm[which(df.farm$date==date), "nseq"]
    } } }

#Test.set
nseq.test.set <- data.frame(matrix(NA, nrow = dim(Test.set)[1], ncol =
dim(Test.set)[2]))
nseq.test.set[,1] <- Test.set$date
colnames(nseq.test.set) <- colnames(Test.set)
for (farm in unique(Test.set.i$site)){
  df.farm <- subset(Test.set.i, Test.set.i$site == farm)
  for(date in unique(df.farm$date)){
    if(is.na(df.farm[which(df.farm$date==date), "months.since.start"])){
      #if fallow period (NaN)
      nseq.test.set[which(nseq.test.set$date==date),
          paste(farm, unique(df.farm$local.authority), sep="_")] <- NaN
    }else{
      nseq.test.set[which(nseq.test.set$date==date),
         paste(farm, unique(df.farm$local.authority), sep="_")] <-
         df.farm[which(df.farm$date==date), "nseq"]
    } } }
```

Finally, both Learning and Test sets were standardised (forced to be standard normal distributed). The DLMs will be applied to the standardised data. We call it "Learning.set.stand" and "Test.set.stand", respectively.

```
#Learning.set
all.values <- unlist(Learning.set[,c(2:ncol(Learning.set))])
Mean <- mean(na.omit(all.values))
SD <- sd(na.omit(all.values))
Standarized.factors <- c(Mean=Mean, SD=SD)

Learning.set.stand <- data.frame(matrix(NA, nrow = dim(Learning.set)[1],
     ncol = dim(Learning.set)[2]))
Learning.set.stand[,1] <- Learning.set$date
colnames(Learning.set.stand) <- colnames(Learning.set)
Learning.set.stand[1:dim(Learning.set.stand)[1],
     2:dim(Learning.set.stand)[2]] <-
     unlist(Learning.set[,c(2:ncol(Learning.set))]) - Mean)/SD

#Test.set
Test.set.stand <- data.frame(matrix(NA, nrow = dim(Test.set)[1],
     ncol = dim(Test.set)[2]))
Test.set.stand[,1] <- Test.set$date
colnames(Test.set.stand) <- colnames(Test.set)
Test.set.stand[1:dim(Test.set.stand)[1], 2:dim(Test.set.stand)[2]] <-
    (unlist(Test.set[,c(2:ncol(Test.set))]) -
    Standarized.factors["Mean"])/Standarized.factors["SD"]
```

This completes the data preparation.

4 Application of the multi-level dynamic linear model for the case

4.1 Overview

Having prepared the data, the next steps are far more general. Therefore, a number of useful R functions have been developed. The functions have all been written as part of the DECIDE project, and they can easily be adapted to other cases than the one presented here.

Basically, the purpose of applying a DLM is to (dynamically) estimate the underlying parameter vector θ_t as observations are done. This is done by use of the Kalman filter which uses the most recent observation y_t to update the previous estimates $(\theta_{t-1}|D_{t-1}) \sim N(m_{t-1}, C_{t-1})$ to $(\theta_t|D_t) \sim N(m_t, C_t)$ where $D_t = \{m_0, C_0, y_1, \ldots, y_t\}$ is the set consisting of the initial information and all observations until time t. The values $m_1 \ldots, m_t$ are referred to as the *filtered* means.

In addition to the filtered values, several other values are calculated by the Kalman filter and the associated functions. Thus, the one-step forecasts distribution $(y_t|D_{t-1}) \sim N(f_t, Q_t)$ is estimated, and the *Kalman smoother* provides the so-called smoothed means and variances $(\theta_t|D_T) \sim N(\tilde{m}_t, \tilde{C}_t)$ where *all* observations (also those after time t) are used to compute the best possible estimates of θ_t given the model.

For monitoring purposes, the forecast errors $e_t = y_t - e_t$ play an important role. Obviously, as long as the process is in control, the forecast errors tend to be "small" whereas "large" errors signal that something has happened and a warning should be given.

Before the Kalman filter can be applied, the initial information and the variance components need to be estimated as illustrated in the following subsection. Also the spline functions accounting for the systematic effects of age of the Salmon must be estimated.

4.2 Parametrisation and training the multi-level DLM

Obtain the initial parameter vector (mu0 corresponding to m_0) and the spline functions (Spline.list) for the standardised and log-transformed mortality data available on the Learning set, by using the function get.m0 from the sourced functions file. Since this data contains a non-linear trend component, we used Spline functions to estimate the trend factor. If you cannot understand what the input parameters for the functions are, find the description directly in the functions script.

```
mu <- get.m0(D=Learning.set.stand, D.months=months.learning.set,
     expected.start.time=0, regions=c("A", "B", "C", "D", "E"))
mu0 <- mu$mu0
Spline.list  <- mu$Spline.list
```

Get the initial variance matrix (C0 corresponding to C_0) by using the function get.C0 from the sourced functions file:

```
C <- get.C0(D=Learning.set.stand, D.months=months.learning.set,
     expected.start.time=0, regions=c("A", "B", "C", "D", "E"))
C0 <- C$C0
```

Run the *estimateDiscountModel()* function with the multi-level DLM to get 3 variance components and 3 discount factors (one for Country, one for Region and one for Farm) to be used in observational variance (V) and system variance (W_t), respectively. Running this code can take some time, so when already run consider to save it.

```
Vs.deltas <- estimateDiscountModel(priorVs=c(3, 2, 1),
     priorDeltas=c(0.99, 0.99, 0.99),
     D=Learning.set.stand, D.months=months.learning.set, mu0, C0,
     Spline.list, regions=c("A", "B", "C", "D", "E"))
```

The function *estimateDiscountModel()* (code above) gives the optimised 3 variance components on the logarithmic scale and the optimised 3 discount factors on logistic scale. We use the function *transformResults()* to get the transformed back values.

```
Vs.deltas.t <- transformResults(Vs.deltas, vars=3)
```

Select the right variance components and discount factors from the vector "Vs.deltas.t". Call them: "countryVar", "regionVar", "farmVar" and "deltas".

```
countryVar = Vs.deltas.t[1]
regionVar = Vs.deltas.t[2]
farmVar = Vs.deltas.t[3]
deltas = Vs.deltas.t[4:6]
```

4.3 Applying the trained multi-level DLM to the Test set

Run the multi-level DLM on the standardised and log-transformed mortality data available on the Test set. The `runDiscountDLM` function runs the Kalman filter.

```
out <- runDiscountDLM(D=Test.set.stand, D.months=months.test.set, mu0,
```

```
    C0, countryVar, regionVar, farmVar, deltas, Spline.list,
    regions=c("A", "B", "C", "D", "E"))
```

Run the Kalman smoother on the DLM outcomes (out). The smoother is a retrospective analysis where data are analyzed from the latest update and working backwards to the initial point.

```
smot <- runSmoother(out)
```

Use the *extract.res()* function to extract the outcomes of both the DLM and the Smoother as a list of eight data frames: mt (filtered mean), ft (forecasts), et (forecast errors), ut (standardized forecast errors), Ct (filtered variance, Qt (one-step forecast variance), mts (smoothed mean) and Cts (smoothed variance).

```
results <- extract.res(out, smot, D=Test.set.stand, H.w.country=NULL,
    regions=c("A", "B", "C", "D", "E"))
```

Validate the DLM by assessing whether the standardized forecast errors follow a standard normal distribution.

```
ut.res <- results[["ut"]]
ut <- unlist(ut.res[,c(8:ncol(ut.res))])
hist(ut)
```

4.4 Using the DLM for monitoring health indicators

The code's final section is dedicated to plotting relevant information for the country, regions, and sites, which includes standardised and log-transformed mortality, as well as non-standardised and non log-transformed mortality.

The goal of this case is to monitor Salmon mortality, and to this end, the 95% credible intervals were added to the plots. If the mortality values exceed the 95% credible interval when applied to the forecasts (red plot), it indicates that mortality has significantly deviated from expected levels, and is considered to be "out of control". Figure 1 shows an example of a site where everything is fine ("in control") and Figure 2 illustrates a site where mortality exceeds the 95% credible interval in several time steps, being considered "out of control".

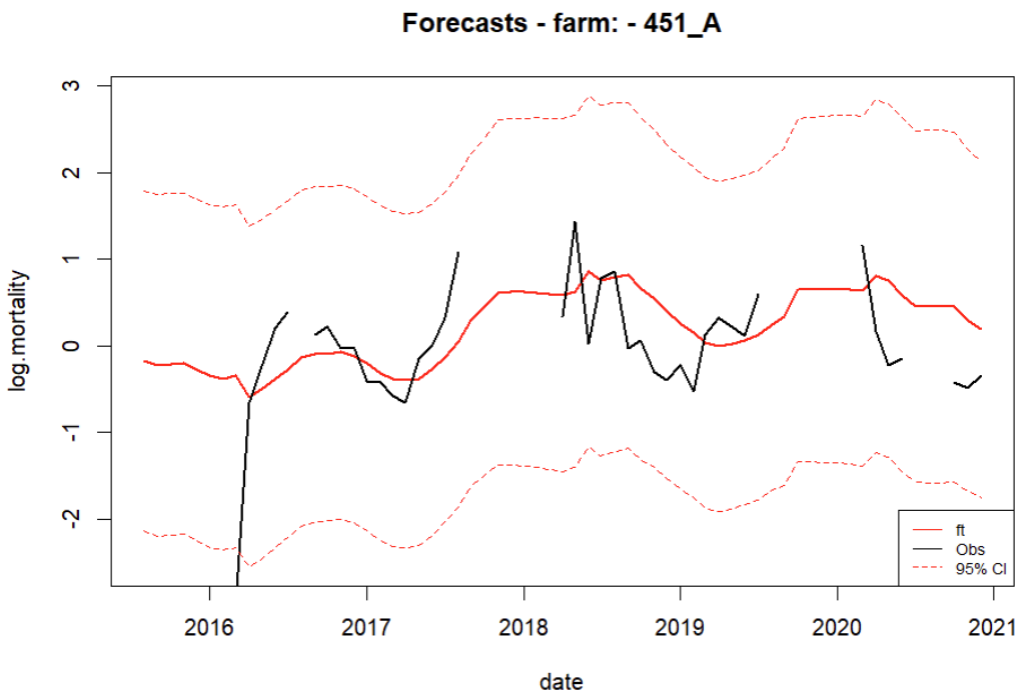

Figure 1: Outcomes from the multi-level DLM applied to farm 451 located in Region A. In black: standardised and log-transformed mortality observations; In red: forecasts (ft) and the respective 95% credible interval (CI). Time periods where no observations are plotted correspond to fallow periods and months where data are missing for other reasons.

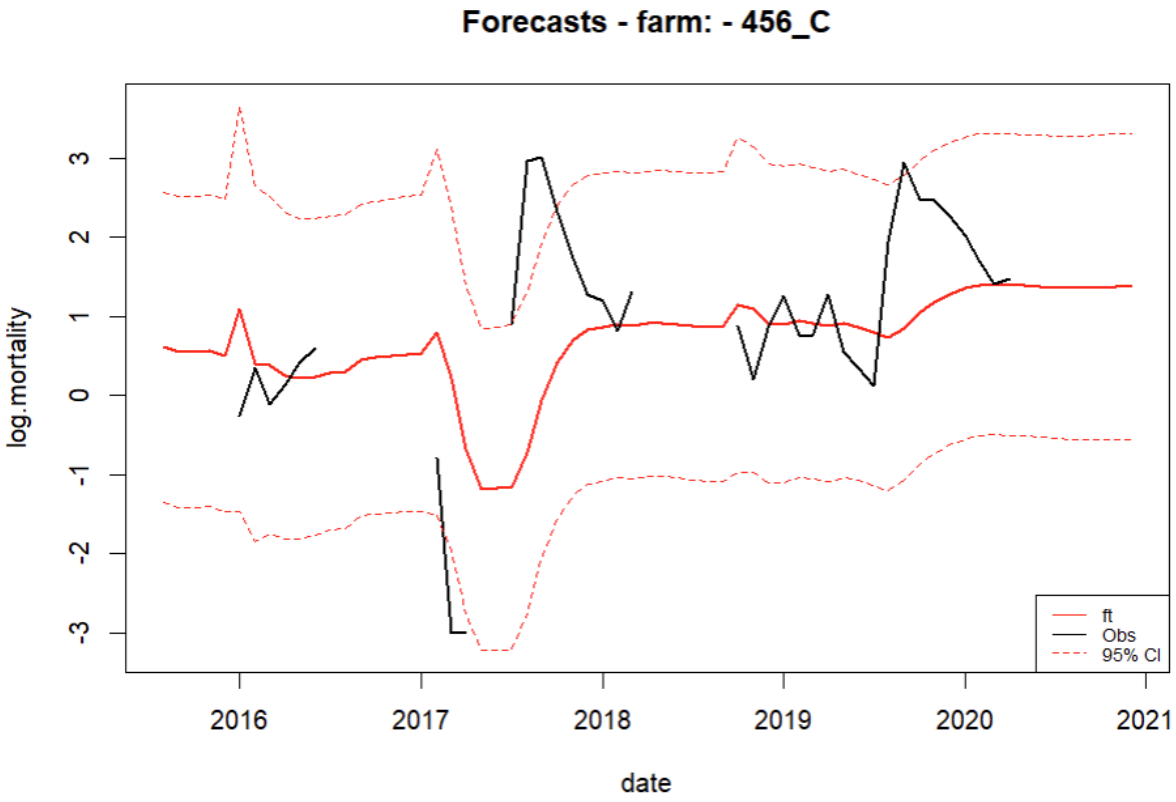

Figure 2: Outcomes from the multi-level DLM applied to farm 456 located in region C. In black: standardised and log-transformed mortality observations; In red: forecasts (ft) and the respective 95% credible interval (CI). Time periods where no observations are plotted correspond to fallow periods and months where data are missing for other reasons.

5 Concluding remarks

Particularly in health monitoring, the hierarchical organisation of the units being monitored seems to be important. It is natural to assume that the observations from different units are correlated according to a pattern reflecting the hierarchy, and furthermore, it is often of interest to monitor at more than one level. Therefore, the concept of multi-level monitoring plays an important role in the DECIDE project as described in the Grant Agreement, and it is expected that many individual cases of the project can benefit from the hierarchical approach.

With the help of this report and the scripts accompanying (together with Deliverable 2.1) it will be possible for other project partners to develop their own multi-level models. The annual workshop series organized by the UCPH partner represent another opportunity for getting support for development of monitoring models. It is also possible for DECIDE partners to get support directly from the UCPH group as mentioned in the introduction.

The UCPH group will also be grateful to be notified of any bugs found by users.