H2020-SFS-2018-2020

DECIDE

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

Deliverable D2.1

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

WP2 – Methods for data analysis and modelling

Authors Leonardo de Knegt (UCPH), Carolina Merca (UCPH), Dan Børge Jensen (UCPH), Anders Kristensen (UCPH)
Lead participant UCPH
Delivery date 18 March 2025
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
Leonardo de Knegt (UCPH), Carolina Merca (UCPH), Dan Børge Jensen (UCPH), Anders Kristensen (UCPH)	Draft deliverable report	26.06.2023
Leonardo de Knegt (UCPH), Dan Børge Jensen	Upload of code	28.06.2023
Gerdien van Schaik (UU)	Revision 1	03.07.2023
Johannes Ripperger (accelCH)	Final checks and formatting of deliverable report	05.07.2023
Leonardo Victor de Knegt (UU), Gerdien van Schaik (UU), Johannes Ripperger (accelCH)	Final version and submission (v1)	11.07.2023
Carolina Merca (UCPH)	Updated with new example (v2)	26.02.2025
Johannes Ripperger (accelCH)	Final checks and formatting of v2	04.03.2025

Content

Abbreviations

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

Partner short names

Short name	Organisation
UCPH	Københavns Universitet
UU	Universiteit Utrecht
accelCH	accelopment Schweiz AG

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. However, the assumption of a constant underlying mean is often questionable, as values can fluctuate randomly, vary throughout the day or seasons, or exhibit systematic trends. In those situations, state-space models like Dynamic Linear Models (DLM) are used, since they build on a Bayesian framework to estimate underlying parameters from current observations, while incorporating measurement errors, systematic fluctuations and prior information from data. Values for the monitored parameters are forecasted for each time-step, based on the underlying mean and prior knowledge. They are then updated with each new observation, by correcting them according to observation errors and variance components.

Objectives of the Deliverable

With the help of this deliverable, the reader will walk through an example approach to create a DLM in R, with the objective of monitoring and 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

An anonymized data with Scottish salmon data is provided to illustrate how software scripts can be applied for estimation of variance components and running the DLMs.

Two broad methods of DLM are described:

- The univariate DLM, where a DLM is defined and applied to each variable separately.
- The multivariate DLM, where a single DLM is defined to describe all the relevant variables simultaneously.

The present deliverable was based on R codes written by Dan Børge Jensen.

Outcome

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

Next steps

The next step for WP2 is to prepare a similar document for multilevel monitoring of time-series data.

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.

A basic model to describe a time series comprises an expected underlying value (or "true mean"), a sample error and an observation error. The fundamental assumption behind this type of model is 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.

To overcome that challenge, 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. 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.

In animal production and health monitoring, the most commonly used type of state space models are 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.

1.2 Previous work leading to this deliverable

In 2022, the University of Copenhagen held a one-week workshop on Dynamic Linear Models for partners of the DECIDE Project, as part of WP2. The workshop focused on the development of univariate DLMs in R, for monitoring and detection of state changes in data series. Theoretical lectures were held in the morning, and in the afternoon, the participants worked writing models for their own data, with support from the UCPH DECIDE partners. Some of the course work continued in partnership after the workshop was finished, with the Cattle Case Study in collaboration with Lely as an example. In September 2023, a second workshop will be held in Copenhagen, with focus on multilevel models, which will also be the subject of the next WP2 deliverable.

1.3 Objectives and content of the deliverable

The example situation, used here for teaching purposes, is the monitoring and detection of increased mortality levels in farmed salmon in Scotland's aquaculture industry.

The present deliverable includes a report, an anonymized dataset, and three R scripts produced by Leonardo Víctor de Knegt, Carolina Merca, and Dan Børge Jensen. The scripts walk the user through a scenario where univariate and multivariate DLMs can be applied, with functions and code lines written in a generalized format that allows them to be applied to any other dataset, provided that the data is in the correct format and the user has a working knowledge of the methods used. The anonymized dataset provided ("df_salmon.RData") consists of monthly mortality and environmental information for each farm (in aquaculture called site). The univariate DLM (R script "*Univariate DLM example documentation and walkthrough – DECIDE_Resubmission .R*") monitors one variable at a time as an independent parameter, while the multivariate version (R script "*Multivariate DLM example documentation and walkthrough – DECIDE_Resubmission.R*") monitors several variables, incorporating their co-variances into the model. Both use functions contained in the script "*DLM functions - DECIDE deliverable_Resubmission.R*". All four (3 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.

Users can get support when using the scripts by contacting the UCPH group at lvdk@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 Case study: Dynamic Linear Models applied to Scottish Salmon data

2.1 Presentation

As explained in the introduction, the contents of the present report refer to the scripts named "*Univariate DLM example documentation and walkthrough - DECIDE_Resubmission.R*", "*Multivariate DLM example documentation and walkthrough - DECIDE_Resubmission.R*", and "*DLM functions - DECIDE deliverable_Resubmission.R*". An anonymized dataset called "*df_salmon.RData*" is also provided. The processes described here can be followed step by step by running or adapting those codes.

The scripts will be applied to Scottish Salmon data, with anonymized site/farm names and regions. The dataset contains mortality and environmental data. Our goal is to monitor and detect increased mortality levels in farmed salmon in Scotland's aquaculture industry. For that, we will develop a univariate DLM, using only the mortality data, and a multivariate DLM including mortality and sea salinity related variables. We will only use mortality and salinity related variables as it was shown in previous work that these were the only relevant variables to monitor salmon mortality in Scotland.[1]

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 of 4 weeks.

[1] Merca C., Boerlage A.S., Kristensen A.R., Jensen D.B. 2024. Monitoring monthly mortality of maricultured Atlantic salmon (*Salmo salar* L.) in Scotland I. Dynamic linear models at production cycle level. Front. Mar. Sci. 11:1436755. doi:10.3389/fmars.2024.1436755

2.2 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 some 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 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 Learning set.
- To standardize 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 standardization involves subtracting the sample mean and, afterwards, dividing by the sample standard deviation.

The anonymized dataset provided already contains log-transformed variables. We will used log-transformed instead of the raw values because in DLMs the error terms have to follow a normal distribution. That was not the case for most variables in this dataset, therefore, we log-transformed them to be closer to a normal distribution (*e.g.* "log0.mortality.rel.20" corresponds to the log transformation of the relative mortality). Also, the original environmental data was reported on a daily basis while the mortality data on a monthly basis. To have all the data within the same time span, we aggregated the environmental data per month. Three aggregation methods were applied, including the 1st decile (*e.g.* "log.d1.sal" corresponds to the log transformation of the 1st decile of salinity), 9th decile (*e.g.* "log.d9.sal" corresponds to the log transformation of the 9th decile of salinity), and the maximum daily variation out of a month (*e.g.* "log.max.daily.range.sal" corresponds to the log transformation of the maximum daily variation of salinity out of a month.

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

```
load("df_salmon.RData")
df <- df_salmon
```

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

```
source DLM("functions - DECIDE deliverable_Resubmission.R")
```

And we will need the following libraries:

```
library(dplyr)
```

Then, 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" and "Test.set", respectively.

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

Both Learning and Test sets were then standardized (forced to be standard normal distributed). The DLMs will be applied to the standardized data. It is not strictly necessary to standardize the data used in a univariate DLM, but there is no harm in it either.

```
SDs <- c()
Means <- c()
for(name in relevant.names){
  SDs <- c(SDs, sd(na.omit(Learning.set[,name])))
  Means <- c(Means, mean(na.omit(Learning.set[,name])))
}
for(name in relevant.names){
  Mean.name <- Means[which(relevant.names == name)]
  SD.name <- SDs[which(relevant.names == name)]
  Learning.set[,name] <- (Learning.set[,name] - Mean.name)/SD.name
  Test.set[,name] <- (Test.set[,name] - Mean.name)/SD.name
}
Standarized.factors <- cbind(as.data.frame(Means), as.data.frame(SDs))
rownames(Standarized.factors) <- relevant.names
```

This completes the data preparation.

2.3 Univariate DLM

Here, we will create a univariate DLM, which is defined and applied to each variable separately. In this case, we will apply it to salmon mortality data.

Make a vector called "relevant.names", which includes the column name that corresponds to salmon mortality:

```
relevant.names <- c("log0.mortality.rel.20")
```

2.3.1 Parametrization and training the univariate DLM

Obtain the initial parameter vector (mu0) and the initial variance matrix (C0) for salmon mortality, by using the functions *get.mu0* and *get.C0* from the sourced functions file. If you can't understand what the input parameters for the functions are, find the description directly in the provided script file.

```
outmu <- get.mu0(Data = Learning.set,
                 stratify.by=NA,
                 time.var = 'months.since.start',
                 expected.start.time = 0,
                 relevant.names = c('log0.mortality.rel.20'),
                 simple.linear = FALSE)
```

```
outc <- get.C0(Data = Learning.set,
               stratify.by=NA,
               time.var = 'months.since.start',
               expected.start.time = 0,
               relevant.names = c('log0.mortality.rel.20'))
mu0 <- outmu$mu0
C0 <- outc$C0
```

Obtain the system matrix Gt and the design matrix Ft, by using functions *get.Gt* and *get.Ft*. When running the two functions, set relevant.names = c('log0.mortality.rel.20'), because this is a univariate model. If your DLM contains a non-linear trend component, you need to first define a spline for the data series to build the Gt matrix, as explained in the code file. You can do that by using function *get.spline*.

```
Spline.list <- get.spline(Data = Learning.set,
                          stratify.by=NA,
                          time.var = 'months.since.start',
                          relevant.names = c('log0.mortality.rel.20'),
                          plot.it = FALSE)

Gt <- get.Gt(Data.A = Learning.set,
             i.A=NA,
             time.var.A='months.since.start',
             stratify.by.A=NA,
             Spline.list.A=Spline.list,
             relevant.names.A=c('log0.mortality.rel.20'))

Ft <- get.Ft(relevant.names.A = c('log0.mortality.rel.20'))
```

Obtain the observational variance (V) for mortality using function *get.V*.

```
V <- get.V(Data = Learning.set,
           identifyer='nseq',
           stratify.by=NA,
           time.var='months.since.start',
           relevant.names=c('log0.mortality.rel.20'))
```

After obtaining a data-based V from the function above, it is possible to directly estimate W and update/optimize the V initially obtained with *get.V*, by using an Estimation Maximization (EM) algorithm. The EM function uses several functions that have been sourced from the functions file, such as *getVSumElement*, *runDLM* (we will come back to this one later), *runSmoother*, and the EM algorithm for a specific number of steps (*runEM*), since we are going to run a version of the same algorithm with an early-stopping feature that stops running when the values of V and W yielding the best model performance are found.

```
varcom <- runEM_earlyStopping(Data=Learning.set,
                              stratify.by=NA,
                              Spline.list=Spline.list,
                              identifyer='nseq',
                              V0=V$V_1,
                              W0=NA,
                              C0.list=outc,
                              mu0.list=outmu,
                              no.better.limit=1,
                              time.var='months.since.start',
                              relevant.names=c('log0.mortality.rel.20'),
                              round.by=4)
```

The output "varcom" should be a list containing the following objects: V.list, W.list, mu0.list and C0.list. Those, in turn, contain V_1 (which is the EM-optimised version of the V_1 initially obtained with *get.V*), W_1 (the value of W estimated by the EM algorithm), mu0_1 (which is the EM-optimized version of the mu0 initially obtained with *get.mu0*) and C0_1 (which is the EM-optimized version of the C0 initially obtained using *get.C0* and stored in "outc"). Here, only the optimized V and W will be used as input for your DLM on the next step, keeping the original mu0 and C0 obtained using *get.mu0* and *get.C0*. However, all variance components given by the EM algorithm can be used when applying the DLM to the Test.set.

In situations for which there is not enough data, not enough time or not enough computing power to use the EM algorithm to optimise V and obtain W, it is possible to use a discount factor (delta) as the percentage of the total variance C that corresponds to W. As in the EM algorithm, several values of delta can be tested and optimised. If you need to find the optimal delta value for modelling mortality with the DLM, you can use the function *optimize.delta*:

```
delta <- optimize.delta(deltas=seq(from=0.5, to=1, by=0.01),
                        Learningset=Learning.set,
                        identifyer='nseq',
                        mu0.list=outmu,
                        C0.list=outc,
                        V.list=V$V_1,
                        relevant.names= c('log0.mortality.rel.20'),
                        et.name="log0.mortality.rel.20",
                        Spline.list=Spline.list,
                        time.var='months.since.start',
                        stratify.by=NA)
```

Now, we are going to validate the model. Apply the optimized DLM to the mortality of each production cycle (nseq) in the Learning.set, to check whether the standardized forecast errors follow a standard normal distribution. Use the *extract.res* function to extract the output of the DLM (which is returned as lists) as a data frame and call the resulting data frame "extracted". From this data frame, get the standardised forecast errors from the column called "ut_log0.mortality.rel.20" and call it "ut". Add the standardised forecast errors to the "ut.all" vector.

```
extracted.all <- data.frame()
identifyer <- 'nseq'

for(ID in unique(Learning.set[,identifyer])){

  ID.set <- subset(Learning.set, Learning.set[,identifyer] == ID)

  res <- runDLM(ID.set,
                mu0=outmu$mu0,
                C0=outc$C0,
                V=varcom$V.list$V_1,
                W=varcom$W.list$W_1,
                adjust.W=FALSE,
                delta=NA,
                relevant.names=c('log0.mortality.rel.20'),
                Spline.list=Spline.list,
                time.var='months.since.start',
                stratify.by=NA)

  extracted <- extract.res(res = res, smot = NULL,
  relevant.names=c('log0.mortality.rel.20'))
  extracted.all <- rbind(extracted.all, extracted)
}
```

Validate the DLM by assessing whether the standardized forecast errors follow a standard normal distribution, plotting it and getting the percentage outside of the 95 % CI. Use function *assess.ut* for that.

```
assess.ut(extracted.all)
```

2.3.2 Applying the trained univariate DLM to the Test set

Now we wish to apply the univariate DLM to the Test.set with the optimized parameters: mu0, C0, V, and W (V and W from the EM algorithm).

Here, we will also apply the Smoother to the DLM results (res), using the function *runSmoother*. The retrospective smoothing analyses the data backwards, providing the best possible estimate of the true underlying mortality level given all available information prior to and after a given time step. Therefore, it gives the best estimates for mortality.

```
extracted.all.test <- data.frame()

for(ID in unique(Test.set[,identifyer])){

   ID.set <- subset(Test.set, Test.set[,identifyer] == ID)

   res <- runDLM(ID.set,
                 mu0=outmu$mu0,
                 C0=outc$C0,
                 V=varcom$V.list$V_1,
                 W=varcom$W.list$W_1,
                 adjust.W=FALSE,
                 delta=NA,
                 relevant.names=c('log0.mortality.rel.20'),
                 Spline.list=Spline.list,
                 time.var='months.since.start',
                 stratify.by=NA)

   smot <- runSmoother(res)

   extracted <- extract.res(res = res, smot = smot,
                            relevant.names=c('log0.mortality.rel.20'))
   extracted.all.test <- rbind(extracted.all.test, extracted)
}
```

2.3.3 Using the univariate DLM for monitoring health indicators

The code's final section is dedicated to plotting relevant information for each production cycle, which includes standardized and log-transformed mortality, as well as the non-standardized and non-log-transformed.

The goal in this case is to monitor salmon mortality, and to this end, the 95% credible intervals were added to the plots. The filtered mean (green plot) can be interpreted as the best possible estimate of the true underlying mortality level given all previous information at each time step, while the smoothed mean (blue plot) can be interpreted as the best possible estimate of the true underlying mortality level given all available information prior to and after a given time step. 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 production cycle where everything is fine ("in control") and Figure 2 illustrates a production cycle where mortality exceeds the 95% credible interval in month 13 after stocking, being considered "out of control".

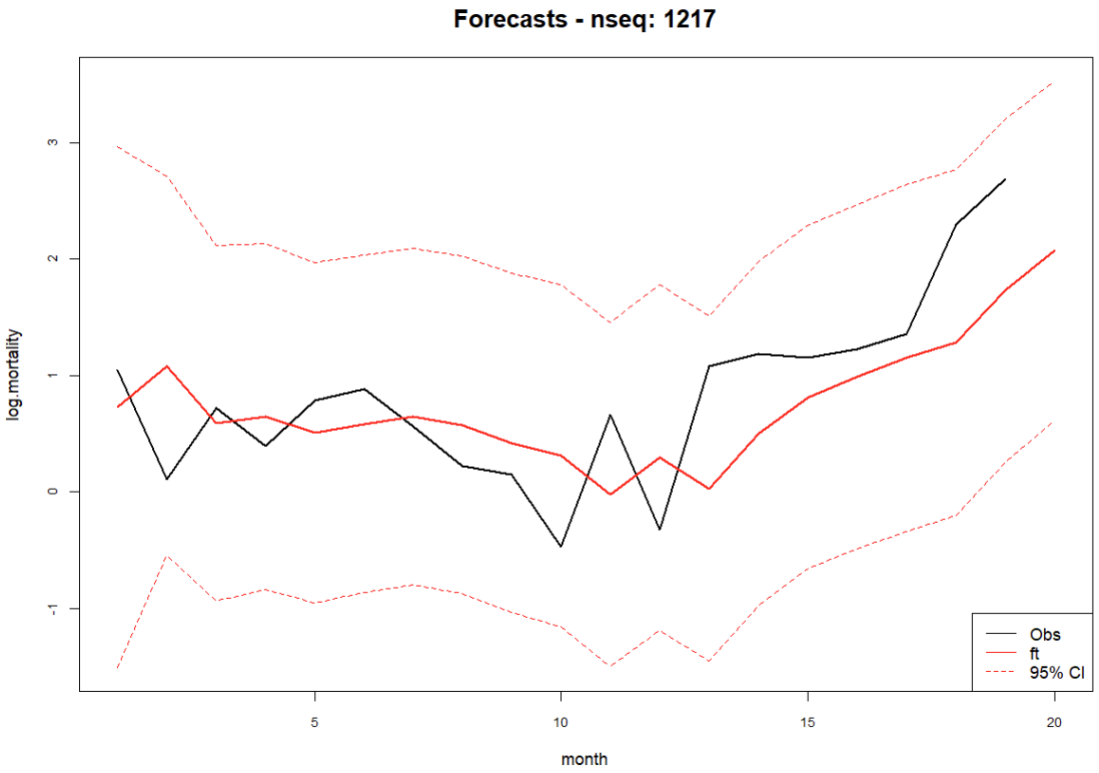

Figure 1: Outcomes from the univariate DLM applied to production cycle 1217. In black: standardized and log-transformed mortality observations; In red: forecasts (ft) and the respective 95% credible interval (CI).

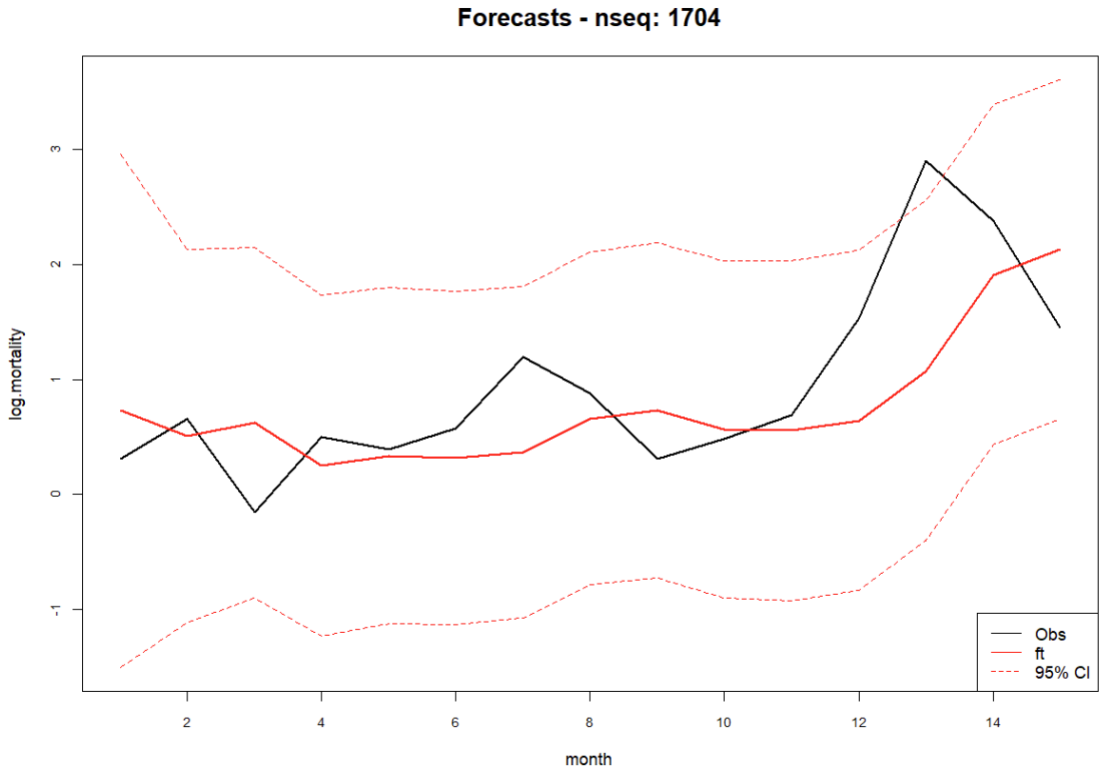

Figure 2: Outcomes from the univariate DLM applied to production cycle 1704. In black: standardized and log-transformed mortality observations; In red: forecasts (ft) and the respective 95% credible interval (CI).

2.4 Multivariate DLM

Now, we wish to make a single multivariate DLM, which simultaneously describes all variables, while taking their mutual co-variances into account.

Make a vector called "relevant.names", which includes the column names that correspond to salmon mortality and to the salinity related variables:

```
relevant.names <- c("log.d9.sal", "log.max.daily.range.sal",
                    "log0.mortality.rel.20")
```

2.4.1 Parametrization and training the multivariate DLM

First, we create the initial parameters, mu0 and C0, for the multivariate DLM. Again, if you can't understand what the input parameters for the functions are, find the description directly in the provided script file.

```
outmu <- get.mu0(Data = Learning.set,
                 stratify.by=NA,
                 time.var = c("n.months", "n.months", "months.since.start"),
                 expected.start.time = 0,
                 relevant.names = c("log.d9.sal","log.max.daily.range.sal",
                                    "log0.mortality.rel.20"),
                 simple.linear = FALSE)

outc <- get.C0(Data = Learning.set,
               stratify.by=NA,
               time.var = c("n.months", "n.months", "months.since.start"),
               expected.start.time = 0,
               relevant.names = c("log.d9.sal","log.max.daily.range.sal",
                                  "log0.mortality.rel.20"))
mu0 <- outmu$mu0
C0 <- outc$C0
```

Now, create Gt and Ft matrices for the multivariate model:

```
Spline.list <- get.spline(Data = Learning.set,
                          stratify.by=NA,
                          time.var = c("n.months", "n.months",
                                       "months.since.start"),
                          relevant.names = c("log.d9.sal",
                                             "log.max.daily.range.sal",
                                             "log0.mortality.rel.20"),
                          plot.it = FALSE)
```

```
Gt <- get.Gt(Data.A=Learning.set,
             i.A=NA,
             time.var.A=c("n.months", "n.months",
                          "months.since.start"),
             stratify.by.A=NA,
             Spline.list.A=Spline.list,
             relevant.names.A=c("log.d9.sal", "log.max.daily.range.sal",
                                "log0.mortality.rel.20"))

Ft <- get.Ft(relevant.names.A = c("log.d9.sal", "log.max.daily.range.sal",
                                  "log0.mortality.rel.20"))
```

Calculate a data-based initial value for V:

```
V <- get.V(Data = Learning.set,
           identifyer='nseq',
           stratify.by=NA,
           time.var=c("n.months", "n.months",
                      "months.since.start"),
           relevant.names=c("log.d9.sal", "log.max.daily.range.sal",
                            "log0.mortality.rel.20"))
```

Apply the EM algorithm:

```
varcom <- runEM_earlyStopping(Data=Learning.set,
                              stratify.by=NA,
                              Spline.list=Spline.list,
                              identifyer='nseq',
                              V0=V$V_1,
                              W0=NA,
                              C0.list=outc,
                              mu0.list=outmu,
                              no.better.limit=1,
                              time.var=c("n.months", "n.months",
                                         "months.since.start"),
                              relevant.names=c("log.d9.sal",
                                               "log.max.daily.range.sal",
                                               "log0.mortality.rel.20"),
                              round.by=4)
```

Now, we are going to validate the model. Apply the optimized DLM to each production cycle (nseq) in the Learning.set, to check whether the standardized forecast errors follow a standard normal distribution.

```
extracted.all <- data.frame()
identifyer <- 'nseq'

for(ID in unique(Learning.set[,identifyer])){

  ID.set <- subset(Learning.set, Learning.set[,identifyer] == ID)

  res <- runDLM(ID.set,
                mu0=outmu$mu0,
                C0=outc$C0,
                V=varcom$V.list$V_1,
                W=varcom$W.list$W_1,
                adjust.W=FALSE,
                delta=NA,
                relevant.names=c("log.d9.sal", "log.max.daily.range.sal",
                                 "log0.mortality.rel.20"),
                Spline.list=Spline.list,
                time.var=c("n.months", "n.months", "months.since.start"),
                stratify.by=NA)

  extracted <- extract.res(res = res, smot = NULL,
                           relevant.names=c("log.d9.sal",
                                            "log.max.daily.range.sal",
                                            "log0.mortality.rel.20"))
  extracted.all <- rbind(extracted.all, extracted)
}
```

Validate the DLM by assessing whether the standardized forecast errors follow a standard normal distribution, plotting it and getting the percentage outside of the 95 % CI. Use function *assess.ut* for that.

```
assess.ut(extracted.all)
```

2.4.2 Applying the trained multivariate DLM to the Test set

Now we wish to apply the multivariate DLM to the Test.set with the optimized parameters: mu0, C0, V, and W (V and W from the EM algorithm). Here, we will also apply the Smoother to the DLM results (res).

```
extracted.all.test <- data.frame()

for(ID in unique(Test.set[,identifyer])){

  ID.set <- subset(Test.set, Test.set[,identifyer] == ID)

  res <- runDLM(ID.set,
                mu0=outmu$mu0,
                C0=outc$C0,
                V=varcom$V.list$V_1,
                W=varcom$W.list$W_1,
                adjust.W=FALSE,
                delta=NA,
                relevant.names=c("log.d9.sal", "log.max.daily.range.sal",
                                 "log0.mortality.rel.20"),
                Spline.list=Spline.list,
                time.var=c("n.months", "n.months", "months.since.start"),
                stratify.by=NA)

  smot <- runSmoother(res)

  extracted <- extract.res(res = res, smot = smot,
                           relevant.names=c("log.d9.sal",
                                            "log.max.daily.range.sal",
                                            "log0.mortality.rel.20"))
  extracted.all.test <- rbind(extracted.all.test, extracted)
}
```

2.4.3 Using the multivariate DLM for monitoring health indicators

The code's final section is dedicated to plotting relevant information for each production cycle and for each variable, including the standardized and log-transformed data, as well as the non-standardized and non-log-transformed.

The goal is to monitor salmon mortality by using a multivariate DLM that incorporates variables expected to influence the variable of interest, such as sea salinity, which is known as a key factor in salmon mortality. In the multivariate DLM, all relevant information is simultaneously monitored, and the inclusion of relevant variables is expected to improve the estimates of the variable of interest.

In Figure 3 you can see the three different types of plots for salmon mortality applied to production cycle number 1217. The filtered mean (green plot) can be interpreted as the best possible estimate of the true underlying mortality level given all previous information at each time step, while the smoothed mean (blue plot) can be interpreted as the best possible estimate of the true underlying mortality level given all available

information prior to and after a given time step. 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".

The DLM also provides information about the salinity related variables. The plots for the salinity related variables are not included in this report since the goal here is to monitor salmon mortality; however, the same plots for the salinity related variables can be seen by running the script.

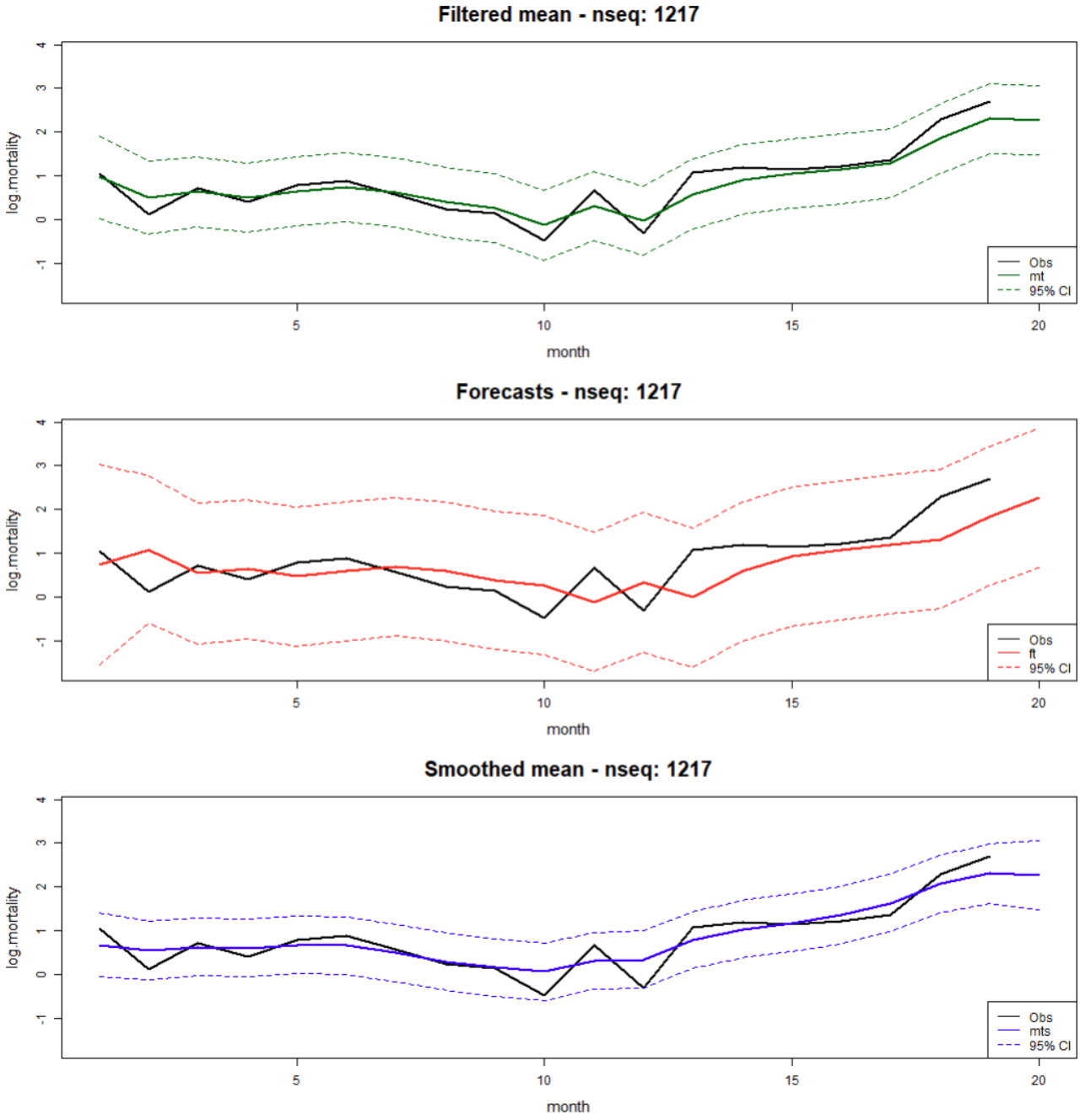

Figure 3: Outcomes from the multivariate DLM for salmon mortality applied to production cycle 1217. In black: standardized and log-transformed mortality observations; In green: filtered mean (mt) and the respective 95% credible interval (CI); In red: forecasts (ft) and the respective 95% credible interval (CI); In red: smoothed mean (mts) and the respective 95% credible interval (CI).