CARLETON UNIVERSITY

SCHOOL OF
MATHEMATICS AND STATISTICS

HONOURS PROJECT

TITLE: Applying State Space Methods to Nonstationary Time Series

AUTHOR: Liam Bruce

SUPERVISOR: Dr. Mohamedou Ould Haye

DATE: April 28, 2020

Carleton UNIVERSITY

CONSENT FOR DISCLOSURE OF FOUR YEAR HONOURS PROJECT

I authorize the

School of Mathematics and Statistics
Office/Program/Individual

To use my Four Year Honours Project

Submitted on

May 4, 2020
Date submitted to Honours Coordinator

For the purpose of

Evaluation and publishing online
State specific purpose of information release

In the period

May 2020 - May 2021
State date range for which permission will exist

Full Name:	Liam Bruce
Student I.D. #:	101002643
Date:	May 4th 2020

Signature Liam Bruce

Protection of Privacy
The personal information requested on this form is collected under the authority of Section 42 (R.S.O.) 1990, c. F.31) of the *Freedom of Information and Protection of Privacy Act* and will be protected under Part 3 of that *Act*. It will be used for the purpose of managing the consent for disclosure of personal information process. Direct any questions about this collection to: [contact position, full address, and business telephone number].

Contents

1 Introduction

A *time series* is a sequence of observations $y_1, ..., y_n$ ordered in time. A common assumption in statistics is, when considering a set of $n > 1$ random variables $X_1, ..., X_n$, that these variables are independent and identically distributed (or i.i.d.). However, when considering time series data, it is important to consider *serial dependence*, which is dependence between observations at different points in time.

A famous modelling technique for time series is the *Box-Jenkins* approach, which is described in Section 2. This method requires the data to be sufficiently regular, or *stationary*, prior to modelling. It will be shown that this assumption can be restrictive when analyzing certain types of data.

A different modelling approach using *state space methods* requires no such stationarity assumption, and is described in Section 3. It offers several other advantages over Box-Jenkins, or *ARIMA* models, including flexibility and the ability to handle missing data. Forecasting future results and checking the goodness-of-fit of state space models will be highlighted. The final section of this report is a case study, which explores applying state space models to a time series of opioid-related harms in the province of Ontario.

All datasets used during the examples and the case study are included in Appendix D.

2 Background - time series analysis

2.1 Basic time series definitions

The most simple example of a time series assumes no serial dependence, and is called white noise. This series consists of a sequence of i.i.d. random variables with mean zero.

Definition 2.1. A time series $\varepsilon_1, ..., \varepsilon_n$ is called *Gaussian white noise* if

$\varepsilon_1, ..., \varepsilon_n \overset{\text{i.i.d.}}{\sim} N(0, {\sigma_\varepsilon}^2)$, for some ${\sigma^2}_\varepsilon > 0$.

To understand the behaviour of random variables in a time series, we define their mean and variability. In the general case of a time series, the mean is allowed to vary with time.

Definition 2.2. The *mean function* of a time series at time t is defined as $\mu_{y,t} = E(y_t) = \int_{-\infty}^{\infty} y f_t(y)\, dy$ for $t = 1, 2, ..., n$. $f_t(y)$ is the marginal density function of the time series at time t.

To account for possible serial dependence in the time series, the variability of a time series is measured as a covariance between two random variables, y_t and y_{t+h} for some integer h. A similar idea can be established for correlation, which is just a function of the variance and covariance of two random variables.

Definition 2.3. The *autocovariance* of a time series at time s and t is defined as $\gamma_y(s,t) = Cov(y_s, y_t) = E[(y_s - \mu_s)(y_t - \mu_t)]$ for $s, t = 1, 2, ..., n$.

Definition 2.4. Define the *autocorrelation function (ACF)* at time s and t as $\rho_y(s,t) = \dfrac{\gamma_y(s,t)}{\sqrt{\gamma_y(s)\gamma_y(t)}}$ for $s, t = 1, 2, ..., n$

An important concept based on the mean and autocovariance of a time series is stationarity. As mentioned in the introduction, assuming that a series is stationary is essential for ARIMA modelling, but is not required when applying state space methods.

Definition 2.5. A time series is *(weakly) stationary* if the following conditions are satisfied:

1. The mean $\mu_{y,t}$ is constant for all $t = 1, ..., n$. That is, the mean may be written as $\mu_{y,t} = \mu_y$ for all t.
2. $\gamma_y(s,t)$ only depends on s and t through the *lag* between two points y_s and y_t, $|s - t|$.

To illustrate stationarity (or a lack thereof), consider the *varve* time series from Shumway and Stoffer (2016) [5]. This data contains the thickness of varves, which are sedimentary deposits resulting from glacial runoff, over a period of 634 years. Included in Figure 1, this series is clearly non-stationary; it has a non-constant mean and variability that changes depending on t.

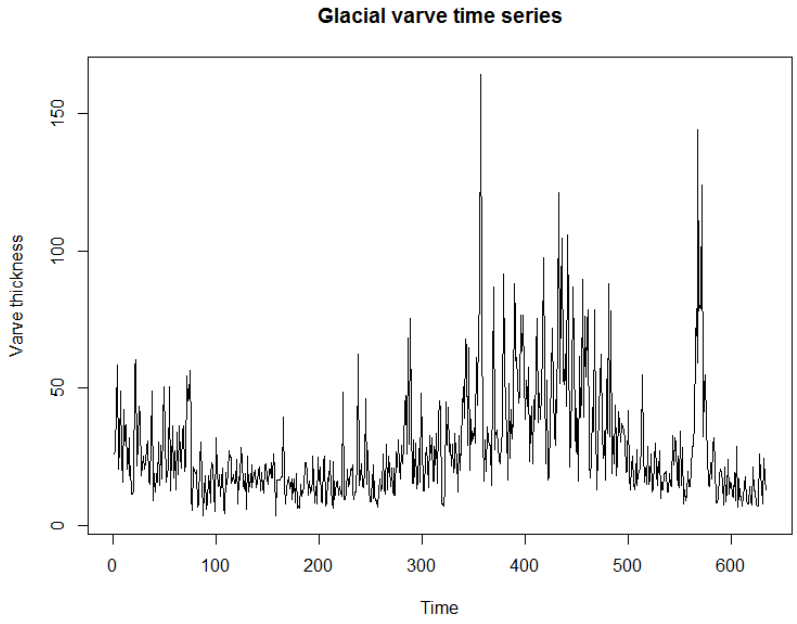

Figure 1: Glacial varve data. (Source: Shumway and Stoffer (2016) [5]).

Instead of interpreting a plot of the data for stationarity, it is easy to use the ACF to see if a time series is stationary. First, a definition of sample autocorrelation is required, which is very similar to the sample correlation that is commonly used in statistics.

Definition 2.6. The *sample autocovariance* of a time series at *lag* h is given by

$$\hat{\gamma}_y(h) = \frac{1}{n}\sum_{t=1}^{n-h}(y_{t+h} - \bar{y})(y_t - \bar{y})$$

for $h = 1, ..., n-1$, where $\bar{y} = n^{-1}\sum_{t=1}^{n} y_t$. Similarly, the *sample ACF* is given by $\hat{\rho}_y(h) = \dfrac{\hat{\gamma}_y(h)}{\hat{\gamma}_y(0)}$.

If the sample ACF "dies down" as h increases, then the series is likely stationary. That is, the sample ACF should converge fairly quickly towards zero. For example, using the varve data, the sample ACF is given in Figure 2. Even up to a maximum lag of 30, the sample ACF is still not close to zero. Therefore, we may conclude that the series is nonstationary.

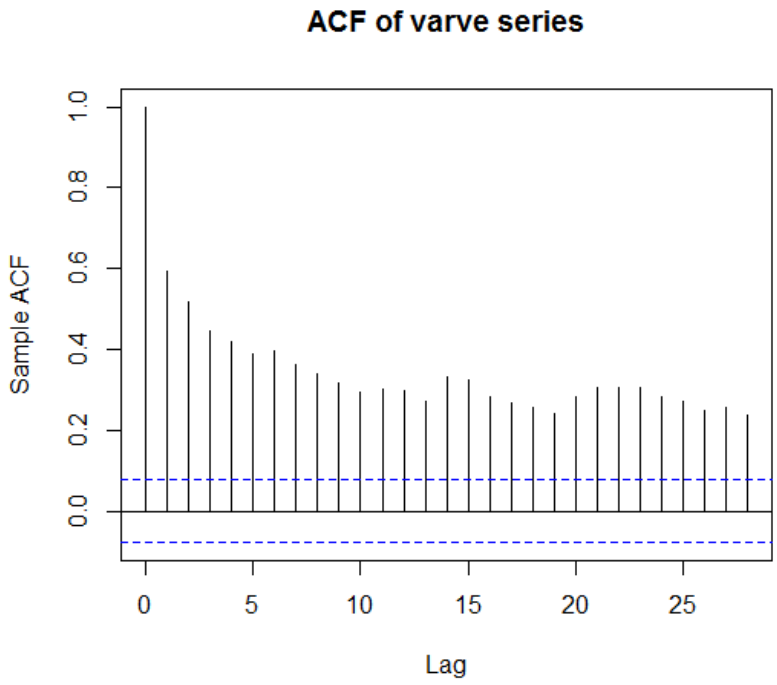

Figure 2: Sample ACF of the varve dataset evaluated at lags $h = 1, ..., 30$. The horizontal blue lines form a 95% confidence interval about 0. (Source: Shumway and Stoffer (2016) [5]).

Note that checking the sample ACF is not a rigorous test for stationarity. Several tests exist to check this, including the augmented Dickey-Fuller test,

mentioned in Chapter 5 of Shumway and Stoffer (2016) [5]. These tests are quite complex, so for the purposes of this report, analyzing the behaviour of the sample ACF will be sufficient.

2.2 ARIMA models

Now, consider some simple time series models. First, suppose each y_t, $t = 1, ..., n$, is a function of previous observations $y_{t-1}, ..., y_1$ and some random value ε_t, which is generated from a series of i.i.d. Gaussian white noise. This is the so-called *autoregressive* model.

Definition 2.7. A *p-order autoregressive* time series model (or AR(p)) with mean μ for $p \geq 1$ is given by

$$y_t - \mu = \phi_1(y_{t-1} - \mu) + ... + \phi_p(y_{t-p} - \mu) + \varepsilon_t$$

for $\phi_1, ..., \phi_p \in \mathbb{R}$, $\phi_p \neq 0$, and $t = 1, ..., n$.

It is also possible to define a model where the current value of the series depends entirely on previous observed white noise. This is the *moving average* model.

Definition 2.8. A *q-order moving average* time series model (or MA(q)) for $q \geq 1$ is given by

$$y_t = \varepsilon_t + \theta_1\varepsilon_{t-1} + ... + \theta_q\varepsilon_{t-q}$$

for $\theta_1, ..., \theta_p \in \mathbb{R}$, $\theta_q \neq 0$, and $t = 1, ..., n$.

We may also combine AR(p) models and MA(q) models to form *ARMA(p,q)* models.

Definition 2.9. An ARMA(p,q) time series, for $p, q \geq 1$, is stationary, and is defined by

$$y_t = \alpha + \phi_1 y_{t-1} + ... + \phi_p y_{t-p} + \varepsilon_t + \theta_1\varepsilon_{t-1} + ... + \theta_q\varepsilon_{t-q}$$

where $\phi_p > 0, \theta_q > 0$, $\varepsilon_t \overset{\text{i.i.d.}}{\sim} N(0, \sigma_\varepsilon^2)$, and $\sigma_\varepsilon^2 > 0$ for $t = 1, ..., n$. Here, $\alpha = \mu(1 - \phi_1 - ... - \phi_p)$.

Note that Definition 2.9 assumes stationarity in the data. In many cases, the data must be transformed into a stationary series prior to modelling. Certain transformations, such as taking the natural log of y_t can reduce the variability in the series. Another method is *differencing*, which considers analyzing the series $y_t - y_{t-1}$ instead of y_t.

The differenced, log-transformed varve data is included in Figure 3 a). The sample ACF of this series dies down very quickly (Figure 3 b)), indicating that the data is stationary.

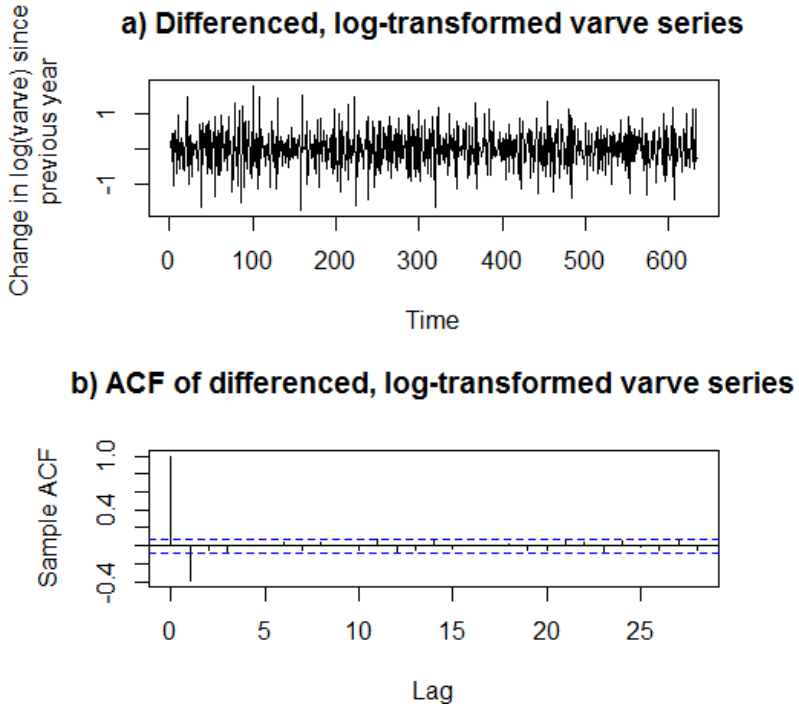

Figure 3: Differenced, log-transformed glacial varve data (a) and its sample ACF (b). (Source: Shumway and Stoffer (2016) [5]).

2.2.1 Example: Random Walk

To illustrate a case where differencing can eliminate information about a series, consider the *random walk* (RW) model:

$$y_t = y_{t-1} + \varepsilon_t$$

where $\varepsilon_t \overset{\text{i.i.d.}}{\sim} N(0, \sigma_\varepsilon^2)$, and $\sigma_\varepsilon^2 > 0$. This model is nonstationary, which can be seen in Figure 4. The differenced series, Δy_t, will be $\Delta y_t = y_t - y_{t-1} = \varepsilon_t$, which is just Gaussian white noise.

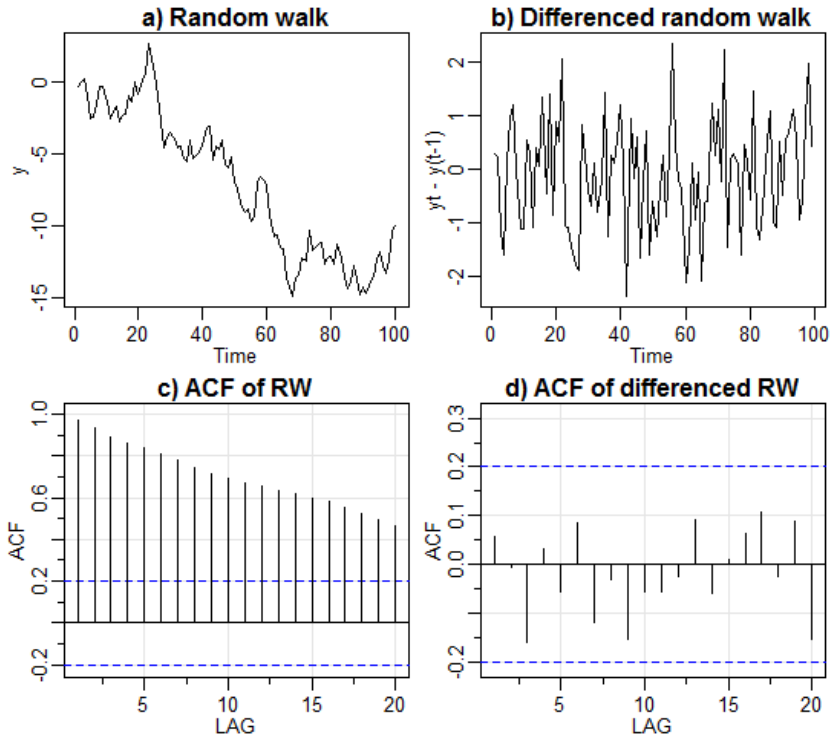

Figure 4: Plots of a random walk and a differenced random walk series ((a) and (b)) with their sample ACFs ((c) and (d)). The blue lines in plots (c) and (d) are 95% confidence intervals around 0.

From Figure 4 (d), the differenced series is stationary, since the sample ACF dies down. However, there are no significant peaks in the ACF, which in-

dicates that there is no serial dependence in the data (which is not true). Ideally, we would work with the original data to prevent the loss of this characteristic.

For notational simplicity, define B as the *backshift operator*, where $By_t = y_{t-1}, B^2y_t = y_{t-2}, ..., B^dy_t = y_{t-d}$ for $1 \leq d \leq t$. Also, define the *AR polynomial* as $\phi(B) = 1 - \phi_1 B - ... - \phi_p B^p$, and the *MA polynomial* as $\theta(B) = 1 + \theta_1 B - ... - \theta_q B^q$. An equivalent way of representing an ARMA(p, q) model using this notation is $\phi(B)y_t = \alpha + \theta(B)\varepsilon_t$, $t = 1, ..., n$.

If a series is subject to differencing, a different notation is used to describe the model. Instead of ARMA, *ARIMA*, for autoregressive integrated moving average, is used. *Integrated* refers to the differencing that occurs in the series.

Definition 2.10. A time series is ARIMA(p, d, q) if

$$\phi(B)(\Delta^d y_t) = \alpha + \theta(B)\varepsilon_t$$

where $\varepsilon_t \overset{\text{i.i.d.}}{\sim} N(0, \sigma_\varepsilon^2)$, and $\sigma_\varepsilon^2 > 0$ for $t = 1, ..., n$. Here, $\Delta^d y_t = (1 - B^d)y_t = y_t - y_{t-d}$, and $\alpha = \mu_d(1 - \phi_1 - ... - \phi_p)$, where $\mu_d = E(\Delta^d y_t)$.

Another source of nonstationarity is *seasonality*, or periodic fluctuations, in the data. To allow for this type of model, consider applying *seasonal differencing*. In the case of monthly data, the data is transformed from y_t into $\Delta^{12} y_t = y_t - y_{t-12}$.

Definition 2.11. A time series is SARIMA(p, d, q, P, D, Q, S) (i.e. seasonal ARIMA) if

$$\Phi_P(B^S)\phi_p(B)(\Delta_S^D \Delta^d y_t) = \alpha + \Theta_Q(B^S)\theta_q(B)\varepsilon_t$$

where S is the seasonal frequency ($S = 12$ in the case of monthly data) and $\Delta_S^D y_t = y_t - y_{t-D*S}$. As before, $\varepsilon_t \overset{\text{i.i.d.}}{\sim} N(0, \sigma_\varepsilon^2)$, and $\sigma_\varepsilon^2 > 0$ for $t = 1, ..., n$. In this case, $\alpha = \mu_S(1 - \Phi_1 - ... - \Phi_P)(1 - \phi_1 - ... - \phi_p)$, where $\mu_S = E(\Delta_S^D \Delta^d y_t)$.

In Definition 2.11, $\Phi(B)$ and $\Theta(B)$ are similar to the AR and MA polynomials used in the ARIMA model. Specifically, they are called the *seasonal* AR and MA polynomials respectively, where: $\Phi(B^S) = 1 - \Phi_1(B^S) - ... - \Phi_P(B^{P*S})$ and $\Theta(B^S) = 1 + \Theta_1(B^S) - ... - \Theta_Q(B^{Q*S})$.

2.3 The Box-Jenkins approach

In this section, the Box-Jenkins approach will be described by fitting a model to the dataset with the monthly number of injury-causing automobile collisions in Canada from 1999 to 2015, in Figure 5 (a). Data from 2016 and 2017 are also available, but were withheld during the fitting process to compare with forecasts generated by the model.

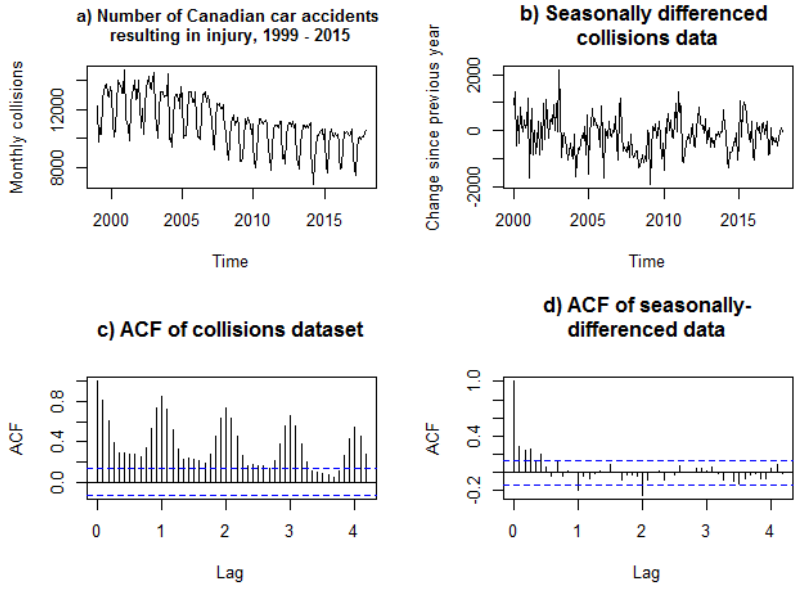

Figure 5: Monthly number of injury-causing automobile collisions in Canada from 1999 to 2018. (Source: Transport Canada [8]).

Modelling was done using the *astsa* package in R from Shumway and Stoffer (2016) [5], and all code is included in Appendix A. The first step in building

the model is to transform the data into a stationary series. Due to apparent seasonality in the data, seasonal differencing is required to make the data stationary. The sample ACFs are included in Figure 5 (c) and (d).

The next step is to determine the number of AR and MA parameters (both seasonal and non-seasonal) to include in the model. A concept that is important in this step of ARIMA modelling is the *partial autocorrelation function* (PACF). Similarly to the ACF, this considers the correlation between y_t and y_{t+h}, but removes the linear effect of $y_{t+1}, ..., y_{t+h-1}$ on both points. Further details about the PACF are beyond the scope of this report, and may be found in Chapter 3 of Shumway and Stoffer (2016) [5].

The following table is similar to the table on page 101 of Shumway and Stoffer (2016) [5], and it provides a good starting point for fitting ARIMA models. The difference between "cutting off" and "dying down" in the table is that cutting off is more abrupt, and the change in the ACF or PACF between consecutive lags is quite different.

Correlation Function	AR(p)	MA(q)	Seasonal AR(P)	Seasonal MA(Q)
ACF	Dies down	Cuts off after lag q	Dies down	Cuts off after seasonal lag Q
PACF	Cuts off after lag p	Dies down	Cuts off after seasonal lag P	Dies down

If the data has both AR and MA parameters, then both the ACF and PACF will die down gradually. The above table should be considered as a guideline only, and several tentative models should be tested prior to obtaining an "optimal" model. The model with the lowest Akaike Information Criterion (AIC) was chosen as optimal for the data.

After checking the ACF and the PACF of the series, and minimizing the AIC, the degrees of non-seasonal AR and MA polynomials should be $p = 1$

and $q = 1$. The seasonal AR and MA polynomials have degrees $P = 1$ and $Q = 2$ respectively. The parameters are included in the output below.

```
Coefficients:
          ar1      ma1     sar1     sma1     sma2  constant
      0.9250  -0.6735  -0.6300  -0.0120  -0.7585  -16.3762
s.e.  0.0518   0.0925   0.3249   0.3099   0.2124    3.2247

sigma^2 estimated as 249316:  log likelihood = -1474.41,
aic = 2962.82
```

To forecast using this model, the *sarima.for* function from the astsa package may be used. Detailed information about forecasting using ARIMA models is omitted from this report. The number of injury-causing collisions has been forecasted 2 years into the future, and compared to the real data from 2016-17.

A plot of the forecasted values is included in Figure 6. The Box-Jenkins approach is highly effective for modelling the collisions data, since all points in 2016-17 fall very close to the forecasts.

Additional model verification, or *diagnostic checking*, is required to ensure goodness-of-fit. However, this will be left until the next section, since it is similar for both Box-Jenkins and state space approaches. The sum of squared forecast errors is equal to $SSE_1 = 3,212,387$, which will be used for comparison with a state space model in the following section.

3 State Space Models

3.1 State space model definition

An alternative approach to ARIMA modelling is the state space approach. The general idea of a state space model is that the observed time series $y_1, ..., y_n$ is generated by an unobserved process of underlying *states* $x_1, ..., x_n$,

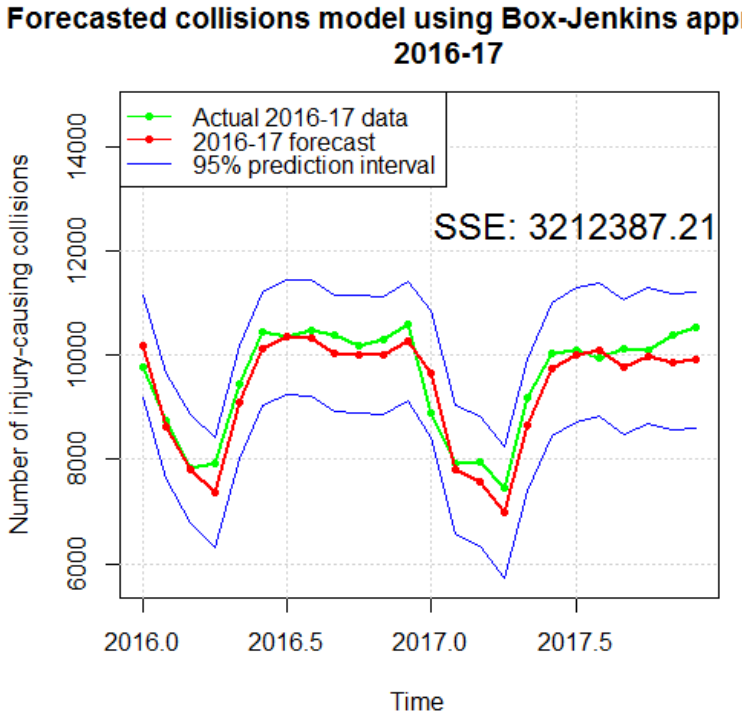

Figure 6: 24 forecasted values of the collisions dataset, with the actual data from 2016-17. (Source: Transport Canada [8]).

where y_t is p-dimensional and x_t is m-dimensional ($p, m \geq 1$) for $t = 1, ..., n$.

Definition 3.1. The *linear Gaussian state space model* (SSM) is written as

$$y_t = Z_t x_t + \varepsilon_t, \qquad \varepsilon_t \sim N_p(0, R_t)$$

$$x_{t+1} = T_t x_t + \eta_t, \qquad \eta_t \sim N_m(0, Q_t),$$

for $t = 1, ..., n$, where $N_z(\mu, \Sigma)$ represents the z-dimensional normal distribution, with mean μ and nonnegative definite covariance matrix Σ. Assume that ε_t and η_t are independent, and that the initial state x_0 is normally distributed with mean μ_0 and covariance Σ_0.

In Definition 3.1, the first equation is called the *observation equation* and the

second is called the *state equation*. Z_t is the $p \times m$ *observation matrix*, which determines which linear of combination of states will be used to represent y_t. ε_t and η_t represent the random noises in the observation and state processes, respectively.

T_t is the $m \times m$ *transition matrix*, which iteratively computes the following state vector given the current information. In this way, the linear state space model is *Markovian*. That is, given the current information about the state x_t, the future value x_{t+1} is independent of the past information $x_1, ..., x_{t-1}$. This property allows for very simple forecasting, as will be seen in Section 3.4.

The SSM in Definition 3.1 can be re-written to accommodate an r-dimensional vector of deterministic inputs u_t in the model ($r > 0$). In Definition 3.2, Υ and Γ are coefficient matrices for the input u_t, where Υ is $p \times r$ and Γ is $m \times r$. For simplicity, the next sections consider an SSM without inputs, but the results can be easily modified to include inputs.

Definition 3.2. The *linear Gaussian state space model with inputs* is written as

$$y_t = Z_t x_t + \Upsilon_t u_t + \varepsilon_t, \qquad \varepsilon_t \sim N_p(0, R_t)$$
$$x_{t+1} = T_t x_t + \Gamma_t u_t + \eta_t, \qquad \eta_t \sim N_m(0, Q_t),$$

for $t = 1, ..., n$, where ε_t and η_t are independent for all t and the initial state is $x_0 \sim N_m(\mu_0, \Sigma_0)$.

3.2 Additive time series components

This section provides several simple examples of univariate state space models. A time series is often defined in terms of different *components*. The most common form of a time series is the *additive model* below.

$$y_t = \mu_t + \gamma_t + \varepsilon_t \qquad t = 1, ..., n$$

Here, μ_t represents a *level* component, which changes slowly over time. γ_t is the *seasonal* component that accounts for regular, periodic variations in the data. Lastly, ε_t is called the *irregular* component, which is usually represented by a white noise series. It accounts for random errors that cannot be explained by other parts of the time series.

The trend component may be represented by the following SSM:

$$y_t = \mu_t + \varepsilon_t$$

$$\mu_{t+1} = \mu_t + \eta_t,$$

where ε_t and η_t are independent Gaussian white noise. Durbin and Koopman (2012) [2] refer to this as the *local level* or *random walk* model. Definition 3.1 is flexible enough to allow for deterministic observations or states. For example, if $\eta_t = 0$ in the local level model, then y_t is just a constant $\mu = \mu_t$ plus some random noise ε_t.

Another example of a level component in a time series is the *local linear trend model*, which introduces a trend component that shifts the level over time.

$$y_t = \mu_t + \varepsilon_t$$
$$\begin{pmatrix} \mu_{t+1} \\ \nu_{t+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mu_t \\ \nu_t \end{pmatrix} + \begin{pmatrix} \eta_t \\ \xi_t \end{pmatrix},$$

where ε_t, η_t and ξ_t are Gaussian white noise. The state noise variables η_t and ξ_t may be dependent. Similarly to the random walk, any one of the noise variables may be set equal to zero to create a deterministic variable in the model.

To add a seasonal component to the model, suppose the seasonal frequency is s. In the case of monthly data, $s = 12$ represents the number of months per year. A common form for a seasonal component is the *dummy seasonal model*, where a white noise term is added to the seasonal state, allowing its effect to change over time.

$$y_t = \gamma_t + \varepsilon_t$$

$$\gamma_{t+1} = -\sum_{j=1}^{s-1} \gamma_{t+1-j} + \omega_t, \quad \omega_t \sim N(0, \sigma_{\omega^2}),$$

where $\sigma_\omega^2 \geq 0$. Here, $E(\sum_{j=1}^{s} \gamma_j) = 0$, or for any $t = 1, ..., n$, $E(\sum_{j=0}^{s-1} \gamma_{t+1-j}) = 0$.

To include all of these components in one SSM, consider an additive model from Section 3.2.3 of Durbin and Koopman (2012) [2], with $y_t = \mu_t + \gamma_t + \varepsilon_t$. The state may be written as: $x_t = (\mu_t, \nu_t, \gamma_t, \ldots, \gamma_{t-s+2})^T$, with $Z_t = (Z_\mu, Z_\gamma)$, $T_t = diag(T_\mu, T_\gamma)$, $Q_t = diag(Q_\mu, Q_\gamma)$. Here, $diag()$ refers to the matrix constructed using the component submatrices as diagonal blocks. This model will be fitted to the collisions data in Section 3.8.

$$Z_\mu = (1, 0), \quad Z_\gamma = (1, 0, \ldots, 0)$$

$$T_\mu = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad T_\gamma = \begin{pmatrix} -1 & -1 & \ldots & -1 & -1 \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \end{pmatrix}$$

$$Q_\mu = \begin{pmatrix} \sigma_\eta^2 & 0 \\ 0 & \sigma_\xi^2 \end{pmatrix}, \quad Q_\gamma = \begin{pmatrix} \sigma_\omega^2 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{pmatrix}$$

3.3 Filtering and smoothing

This subsection focuses on iteratively computing state estimates based on previous information in the series. The results in this section take advantage of the Markovian property of the SSM. Proofs of the results in this section are omitted, and may be found in Chapter 4 of Durbin and Koopman (2012) [2] or Chapter 6 of Shumway and Stoffer (2016) [5].

The best estimate of the state at time $t = 1, ...n$ is the conditional expectation of the state, given all previous observations $y_{t-1}, ..., y_1$. This is called *forecasting*. After the forecast is computed, the estimate can be updated for each new observation y_t. This process is called *filtering*. Both of these estimations are done by the *Kalman filter*.

Result 1. Let $\hat{x}_t = E(x_t|y_1, ..., y_{t-1})$ be the forecasted state estimate and let $\hat{x}_{t|t} = E(x_t|y_1, ..., y_t)$ be the filtered state estimate, $t = 1, ..., n$. The *Kalman filter* recursion for the SSM in Definition 3.1 is given by:

$$v_t = y_t - Z_t\hat{x}_t, \qquad F_t = Z_tP_tZ_t^T + R_t$$

$$\hat{x}_{t|t} = \hat{x}_t + P_tZ_t^TF_t^{-1}v_t, \qquad P_{t|t} = P_t - P_tZ_t^TF_t^{-1}Z_tP_t$$

$$\hat{x}_{t+1} = T_t\hat{x}_t + K_tv_t, \qquad P_{t+1} = T_tP_{t|t}(T_t - K_tZ_t)^T + Q_t$$

for $t = 1, ..., n$. $K_t = T_tP_tZ_t^TF_t^{-1}$ is called the *Kalman gain*. Note that the initial state estimate is $\hat{x}_0 = \mu_0$ and initial state variance $P_0 = \Sigma_0$.

P_t and $P_{t|t}$ are the covariance matrices for the forecasted and filtered state estimates respectively. The values v_t in Result 1 are the one-step-ahead prediction errors or *innovations* in the Kalman filter, with covariance matrix F_t. These values will be used during diagnostic checking for the SSM.

After computing the filtered states, the estimation can be refined by computing the expectation of the state given all observations $y_1, ..., y_n$. This process is called *smoothing* and $\hat{x}_t^n = E(x_t|y_1, ..., y_n)$ is the *smoothed state estimate*. A smoothing algorithm from Shumway and Stoffer (2016) [5] is used to calculate these values from the Kalman filter output.

Result 2. The *Kalman smoother* with initial conditions $\hat{x}_{n|n}$ and $P_{n|n}$ from Result 1 is given by:

$$\hat{x}_{t-1}^n = \hat{x}_{t|t} + J_{t-1}(\hat{x}_t^n - \hat{x}_t)$$

$$P_{t-1}^n = P_{t|t} + J_{t-1}(P_t^n - P_t)J_{t-1}^T,$$

where $J_{t-1} = P_{t-1|t-1}T_t^TP_t^{-1}$, for $t = n, n-1, ..., 1$.

To summarize Results 1 and 2, the Kalman filter computes forecasted and filtered states by moving forwards in time through observations $y_1, ..., y_n$. As a second step, the smoothing algorithm runs backwards through the data and updates the states to provide the best possible estimates.

3.4 Forecasting and handling missing data

One of the main advantages of the state space approach is the simplicity of forecasting and the ability to handle missing or irregularly-spaced observations, as mentioned in Chapter 4 of Durbin and Koopman (2012) [2]. First, consider the case where observations y_t, $t = \tau, \tau + 1..., \tau + k$ are partially or entirely missing for some $k > 0$ and $\tau \in \{1, ..., n - k\}$.

To calculate the missing state estimates, set the rows of Z_t corresponding to the missing observations equal to the zero vector in the Kalman filter recursion for $t = \tau, \tau + 1..., \tau + k$. For example, if the second row of y_t is missing, set the second row of Z_t equal to the zero vector in the Kalman filter.

If all p observations are missing, then Z_t is the zero matrix. Using Result 1, this gives $\hat{x}_{t|t} = \hat{x}_t$, $P_{t|t} = P_t$, $\hat{x}_{t+1} = T_t\hat{x}_t$, and $P_{t+1} = T_tP_tT_t^T + Q_t$. The smoothed estimates can be calculated as usual, since the inputs for Result 2 (forecasted and filtered state estimates) are computed in this way.

Forecasting is done in a similar way to missing values, and is discussed in Section 4.11 of Durbin and Koopman (2012) [2]. In this section, they mention that the forecast that minimizes the mean squared error of the j-step ahead forecast ($j = 1, ..., J$) is the conditional expectation $\hat{y}_{t+j} = E(y_{n+j}|y_1, ..., y_n)$.

Using Result 1, $\hat{y}_{t+1} = Z_{n+1}\hat{x}_{t+1}$. The covariance matrix for this forecast is given by $F_{n+1} = Var(\hat{y}_{n+1}|y_1, ..., y_n) = Z_{n+1}P_{n+1}Z_{n+1}^T + R_{n+1}$. In general, to compute forecasts for $j = 1, ..., J$,

$$\hat{y}_{n+j} = Z_{n+j}\hat{x}_{n+j}, \quad F_{n+j} = Z_{n+j}P_{n+j}Z_{n+j}^T + R_{n+j}$$

$$\hat{x}_{n+j+1} = T_{n+j}\hat{x}_{n+j}, \quad P_{n+j+1} = T_{n+j}P_{n+j}T_{n+j}^T + Q_{n+j}$$

This is the same as running the Kalman filter with $Z_{n+j} = 0$ for $j = 1, ..., J$, similarly to what is done when the entire observation y_t is missing.

It is important to note that the Kalman filter and smoother only compute state estimates. Any other unknowns in the model, including initial state parameters μ_0 and Σ_0, state disturbance covariance matrix Q_t, and observation disturbance covariance matrix R_t require additional estimation techniques.

Depending on the model specifications, it may also be desirable to find estimates for T_t. In the simple additive model in Section 3.2, this was not necessary since the values in the transition matrix were fixed. Some additional models will be seen in the case study that will require estimation of T_t.

3.5 Maximum likelihood estimation

Parameter estimation in this report was done by method of maximum likelihood. Two R packages - KFAS and MARSS - will be used to fit SSMs in this report, and each of them performs a different type of maximum likelihood estimation. The MARSS package uses the expectation maximization (EM) algorithm from Holmes (2012) [3].

The EM algorithm takes an input vector of initial parameter values, denoted as Θ_0. The expected value of the log-likelihood from the Kalman filter, given all observations $y_1, ..., y_n$ and initial parameters Θ_0, is computed. After this is done, this expectation is maximized with respect to parameter vector Θ, which returns a new "optimal" set of parameters Θ_1.

These two steps are called the *expectation* step and the *maximization* step respectively. After Θ_1 is computed, the two steps are repeated using Θ_1 as the initial parameter vector. In general, let Θ_j be the estimate at iteration $j \geq 0$. The EM algorithm keeps updating Θ_j until convergence.

Let L be the joint likelihood function of $y_1, ..., y_n$ and $x_1, ..., x_n$. Iteration $j > 0$ of the EM algorithm may be represented by the following equation calculation:

$$\Theta_j = \arg\max_{\Theta} \int_{\mathbf{x}} \int_{\mathbf{y}} \log L(\mathbf{x}, \mathbf{y}) f(\mathbf{x}, \mathbf{y} | \mathbf{Y} = \mathbf{y}, \Theta_{j-1}) d\mathbf{y} d\mathbf{x}$$

Due to the complexity of finding maximum likelihood estimates for time series, technical details are omitted from this report. Holmes (2012) [3] gives information on the derivation of the EM algorithm used in the MARSS package. Also, Shumway and Stoffer (2016) [5] give a simpler version of the EM algorithm for SSMs that is implemented in the astsa R package.

The maximum likelihood estimation done by the KFAS package relies on the *optim* function in R (Helske (2017) [6]); the likelihood computed from running the Kalman filter is maximized relative to the model's unknown parameters. However, as the number of parameters becomes larger, this optimization method often does not reach a global maximum, making the KFAS package better for simpler SSMs. Further details regarding fitting SSMs in R will be discussed in the next section.

3.6 State space modelling in R

There are many different packages, including KFAS and MARSS, that handle fitting SSMs in R. It is important to note that each package is ideal for different SSM specifications. For example, the structural seasonal model in Section 3.2 cannot be fitted using the MARSS package. Instead, a different seasonal model in Section 13.6 of Holmes et al.(2020) [7] would have to be used.

During the course of the analysis in this report, it was found that KFAS package does well with fitting these simple additive models. MARSS performs well for more complex, multivariate models. Using these observations, KFAS

was used for the collisions data example, and MARSS was used in the analysis of the case study data.

The initial state parameters μ_0 and Σ_0 are unknown in general. Handling this issue is called the *initialization* of the Kalman filter. μ_0 and Σ_0 can be treated as unknown parameters that need to be estimated, which is done in the EM algorithm. Another approach is to let $\Sigma_0 = \kappa I_m$ with $\kappa \to \infty$, where I_m is the $m \times m$ identity matrix. This is called *diffuse initialization* and it is described in Chapter 5 of Durbin and Koopman (2012) [2].

It is important to note that KFAS uses diffuse initialization, which is a good way of handling the data when nothing is known about the distribution of the initial state x_0. A different Kalman filter, called the *exact initial Kalman filter*, is used by the KFAS package. The exact initial filtering and smoothing recursions are omitted from this report, but are included in Appendix A of Helske (2017) [6].

One final note concerns the *identifiability* of SSMs. Holmes (2012) [3] notes on page 2 that it is essential for a SSM to be identifiable, or to have a unique solution. It will be assumed that the simple seasonal model from Durbin and Koopman (2012) [2] is identifiable. Also, according to Casals et al. (2016) [4] on page 138, vector autoregressive models are identifiable, which permits the use of the models in the case study.

3.7 Diagnostics checking

To check the goodness-of-fit for time series models, it is important to test the underlying assumptions that generate the model. In the case of the SSM, the key assumptions were that the disturbances ε_t and η_t are i.i.d normal random vectors. In other words, these variables should be serially independent and normally distributed with constant variance.

Using a note from Section 2.12 of Durbin and Koopman (2012) [2], these

assumptions may be checked by seeing if these properties hold for the Kalman filter innovations, v_t. If these vectors are serially independent and normally distributed, then so are ε_t and η_t.

To begin, the innovations must be standardized. If y_t is one-dimensional, this is given by $v_t^s = \frac{v_t}{\sqrt{F_t}}$. If diffuse initialization is used, v_t^s is ignored for $t = 1, ..., d$ during diagnostics checking, where d is the number *diffuse elements*. This is equal to the number of states in x_0.

To check for normality, a QQ-plot may be used, which plots sample quantiles from the standardized innovations against the theoretical quantiles from the standard normal distribution. If both quantiles are approximately equal (the points lie near the 45 degree line on the QQ-plot), then the data should be approximately normal.

A test statistic for normality is mentioned in Section 2.12 of Durbin and Koopman (2012) [2], which is not used in this report. Under the null hypothesis of normality, the test statistic N has an asymptotic chi-square distribution with 2 degrees of freedom, where S and K are the sample skewness and kurtosis respectively:

$$N = n\left(\frac{S^2}{6} + \frac{(K-3)^2}{24}\right)$$

To test for serial independence in the innovations, the *Ljung-Box statistic* was used. For a pre-specified integer k, this is given in Section 8.5 of Commandeur and Koopman (2007) [1]:

$$Q(k) = n(n+2)\sum_{j=1}^{k}\frac{\hat{\rho}^2(j)}{n-j}$$

where $\hat{\rho}(j)$ is the sample ACF of the innovations calculated at lag j. Under the null hypothesis of independence, Q has an asymptotic chi-square distribution with $k - w + 1$ degrees of freedom. Here, w is the dimension of the estimated parameter vector Θ for the SSM.

Finally, there must be *homoscedasticity* in the standardized one-step-ahead Kalman filter residuals. That is, the variance of these innovations should be constant over time. A simple way to check this is by analyzing a plot of the standardized residuals. If the variability seems to be constant over time, then the assumption of homoscedasticity should hold.

Commandeur and Koopman (2007) [1] give a test statistic for this:

$$H(h) = \frac{\sum_{t=n-h+1}^{n} (v_t^s)^2}{\sum_{t=d+1}^{d+h} (v_t^s)^2}$$

Here, $H(h)$ has an asymptotic $F_{h,h}$ distribution under the null hypothesis of constant variance. Commandeur and Koopman (2007) [1] recommend dividing the dataset into three parts, and using the oldest and the most recent thirds of the data to run this test (or $h = \frac{(n-d)}{3}$).

In the multi-dimensional case, the standardized innovations are:

$$v_t^s = B_t v_t,$$

where B_t is the matrix such that $B_t^T B_t = F_t$. Once this transformation is done, the elements of v_t^s are uncorrelated, and it is appropriate to apply the tests in this section to each element of v_t^s (Section 7.5, Durbin and Koopman (2012) [2]).

3.8 Example: Fitting a state space model to the collisions data

Recall the collisions data from Section 2.3. A SSM will now be fitted to this data using the methods discussed in previous sections. The sum of squared errors from this model will be compared to the Box-Jenkins SARIMA model from earlier.

Since there is seasonality ($s = 12$) in the collisions data, the additive seasonal model from Section 3.2 will be used. This model was fitted using the KFAS

package in R, and all code for this section is included in Appendix B. The system matrices that will be used for modelling are included below.

$$x_t = (\mu_t, \nu_t, \gamma_t, \ldots, \gamma_{t-10})^T, \qquad Z = (1, 0, 1, 0, ..., 0)$$

$$T = \begin{pmatrix} 1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & -1 & \ldots & -1 & -1 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \end{pmatrix}, \quad R = \sigma_\varepsilon^2, \quad Q = \begin{pmatrix} \sigma_\eta^2 & 0 & 0 & \ldots & 0 \\ 0 & \sigma_\xi^2 & 0 & \ldots & 0 \\ 0 & 0 & \sigma_\omega^2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

As mentioned in the previous sections, diffuse initialization will be used for this dataset which means that the initial state mean and variance, μ_0 and Σ_0, do not need to be estimated. Therefore, only four parameters need to be estimated: $\Theta = \{\sigma_\varepsilon, \sigma_\eta, \sigma_\xi, \sigma_\omega\}$. The maximum likelihood estimates returned from R are as follows:

$$\sigma_\varepsilon^2 = 225,068.81, \quad \sigma_\eta^2 = 0.74, \quad \sigma_\xi^2 = 59.53, \quad \sigma_\omega^2 = 22.51$$

The plots of the smoothed states are included in Figure 14, which reveal some interesting information about the collisions dataset. Firstly, the level drops quickly between around 2006 and 2009. In addition, the stochastic slope component is negative starting after around 2003. Both of these indicate a steady drop in the number of injury-causing collisions (corrected for seasonality) over the course of the data.

The seasonal component shows the variations that are strictly related to the month of the year in which the collisions occur. Since the data starts in January 1999, the number of collisions drops in February of that year. In general, August and December tend to have the highest number of injury-causing collisions, whereas February through May have the lowest.

The irregular component gives the estimated values of ε_t in the SSM. This component may be used for outlier detection, as in Section 2.12 of Durbin

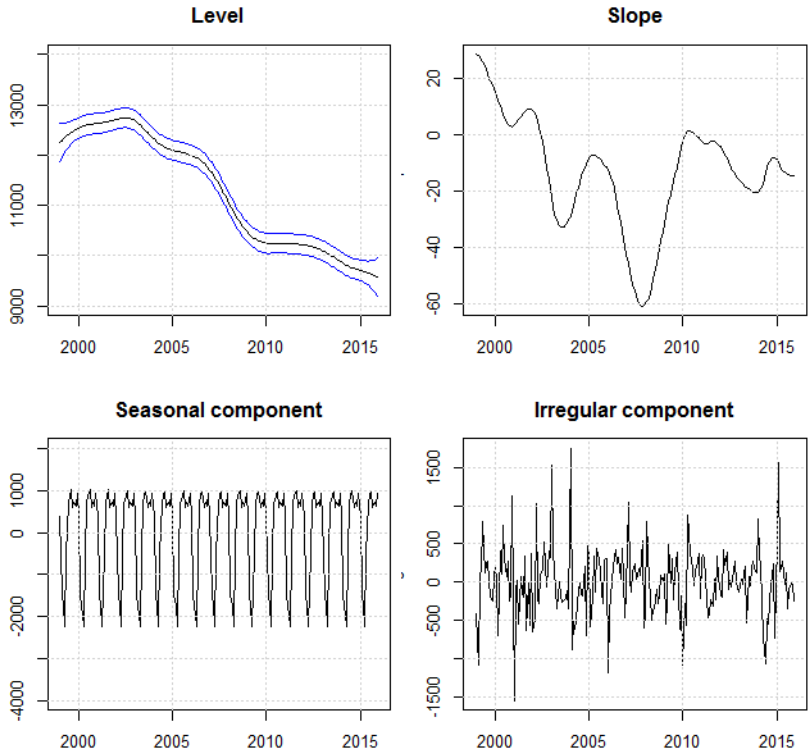

Figure 7: The smoothed states for the collisions data. The level component includes 95% confidence intervals in blue.

and Koopman (2012) [2]. To compensate for large peaks in this series, an *intervention variable* may be added as a deterministic input. This will be explored in the case study.

Next, diagnostic checking was done on the SSM model fit. Figure 8 contains four plots: a plot of the standardized residuals (a), a QQ-plot (b), p-values for the Ljung-Box statistic for lags $k = 5, ..., 25$ (c), and the sample ACFs for lags $k = 1, ..., 25$ (d). Each plot indicates a good fit; the sample quantiles are close to the 45 degree line in the QQ-plot, all p-values for the Ljung-Box statistic are insignificant at the 95% confidence level, and there are few significant peaks in the ACF of the residuals.

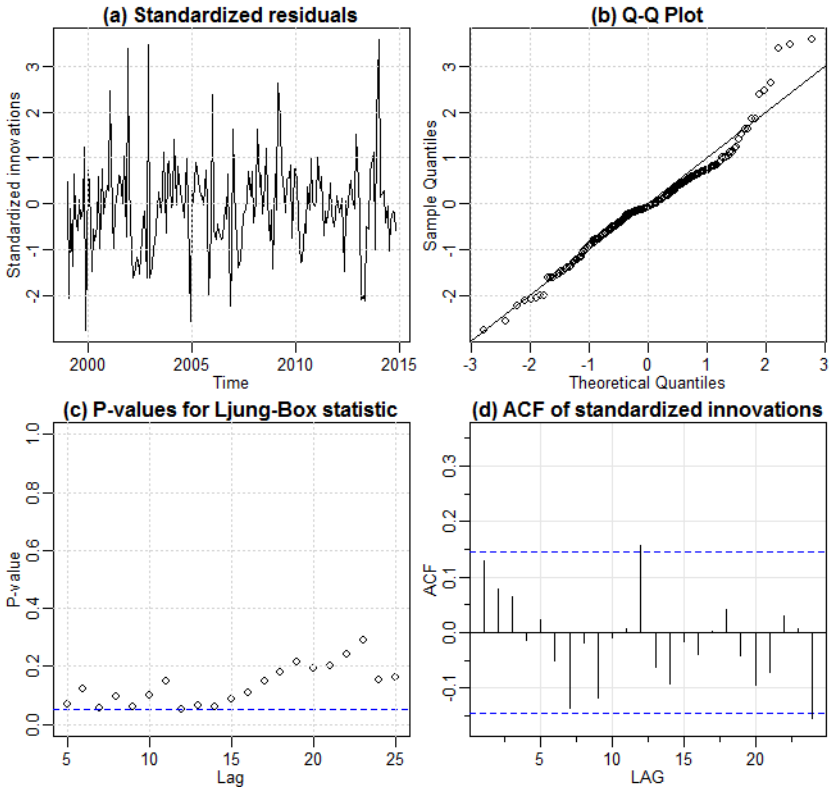

Figure 8: The diagnostic plots for the fitted seasonal SSM to the collisions data.

Therefore, it may be concluded that the residuals are serially independent and approximately normal. The final check that must be made is for homoscedasticity in the residuals. Using the H statistic from Section 3.7, with $h = \dfrac{(n-d)}{3} \doteq 64$,

$$H(64) = 0.6331448$$

Using a note from Section 8.5 of Koopman and Commandeur (2007) [1], the null hypothesis of homoscedasticity will be rejected at the 95% confidence level if one of the following are true:

1. If $H(h) > 1$ and $H(h) < F_{h,h}(0.025)$, or

2. $H(h) < 1$ and $\dfrac{1}{H(h)} < F_{h,h}(0.025)$.

where $F_{h,h}(x)$ is the upper-tail probability of the F distribution with degrees of freedom (h, h). Since $H(64) < 1$, and $\dfrac{1}{H(h)} = 1.579418 < 1.639485 = F_{64,64}(0.025)$, it may be concluded that the innovations have constant variance. Thus, all diagnostics checks indicate that this model is a good fit.

Now that the goodness-of-fit has been verified, the model may now be applied to predict the values for the collisions dataset in 2016-17. The predictions are included in Figure 9, and the sum of squared errors is equal to $SSE_2 = 3,110,139$, which is less than the error in the SARIMA model for the collisions data. While both models provide good forecasts for 2016-17, the SSM seems to be slightly more accurate.

4 Case Study: Opioid Harms Model

The case study that was chosen in this report uses data from the Public Health Ontario website [9], and contains monthly counts of opioid-related harms from January 2003 until September 2019. All data from 2019 was withheld during the model fitting, and will be compared to the forecasts generated by the model.

The data includes monthly emergency department (ED) visits ($y_{t,1}$), hospitalizations ($y_{t,2}$), and deaths ($y_{t,3}$) from opioid poisonings. In terms of geography, the data is aggregated over four different Local Health Integration Networks (LHINs) in Ontario: Central, Central West, Missisauga Halton, and Toronto Central.

The three different series are plotted in Figure 10. Note that the death data for 2003 and 2004 is missing. These years do not need to be ignored, and they will be estimated by the fitted SSM. All relevant code for this case study is in Appendix C.

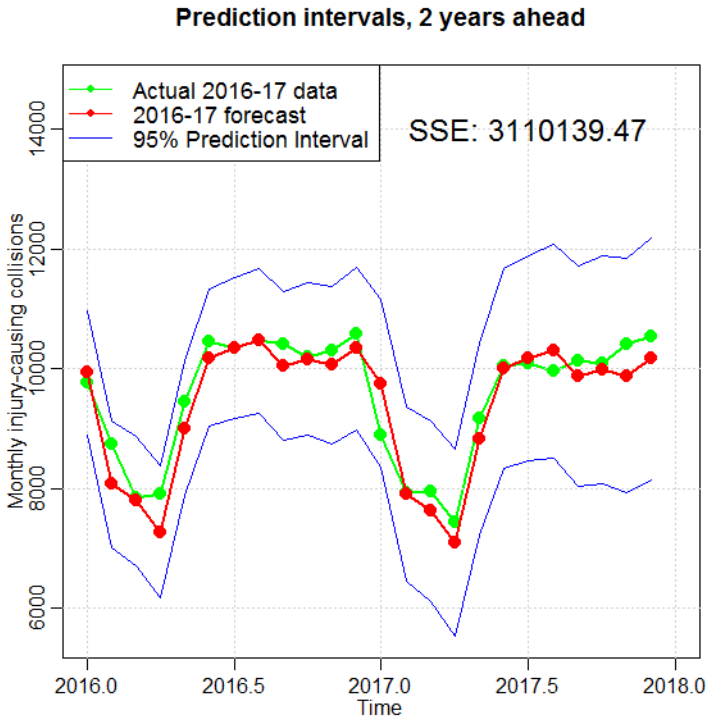

Figure 9: 24 forecasted values of the collisions dataset, with the actual data from 2016-17. (Source: Transport Canada [8]).

To begin the analysis, a naive first guess was made for the model - a variation of a *vector autoregressive* model with noise, which is given on page 289 of Shumway and Stoffer (2016) [5]. The SSM equations are given below.

$$\boldsymbol{y_t} = Z_t \boldsymbol{x_t} + \varepsilon_t, \quad \boldsymbol{x_t} = T_t \boldsymbol{x_{t-1}} + \eta_t,$$

where $\varepsilon_t \sim N_3(0, R_t)$, $\eta_t \sim N_3(0, Q_t)$, and:

$$\boldsymbol{y_t} = \begin{pmatrix} y_{t,1} \\ y_{t,2} \\ y_{t,3} \end{pmatrix}, \boldsymbol{x_t} = \begin{pmatrix} x_{t,1} \\ x_{t,2} \\ x_{t,3} \end{pmatrix}, Z_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, R_t = \begin{pmatrix} R_{t,1} & 0 & 0 \\ 0 & R_{t,2} & 0 \\ 0 & 0 & R_{t,3} \end{pmatrix},$$

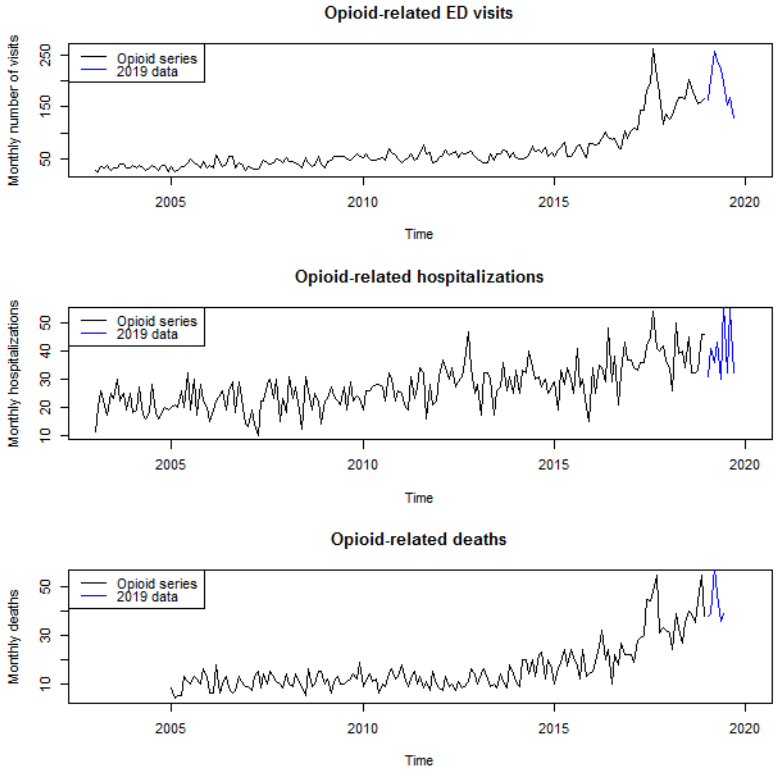

Figure 10: Monthly opioid-related harms from 2003-2019 in 4 LHINs: Central, Central West, Mississauga Halton, and Toronto Central. (Source: Public Health Ontario [9]).

$$T_t = \begin{pmatrix} \phi_{1,1} & \phi_{1,2} & \phi_{1,3} \\ \phi_{2,1} & \phi_{2,2} & \phi_{2,3} \\ \phi_{3,1} & \phi_{3,2} & \phi_{3,3} \end{pmatrix}, Q_t = \begin{pmatrix} Q_{1,1} & Q_{2,1} & Q_{3,1} \\ Q_{2,1} & Q_{2,2} & Q_{3,2} \\ Q_{3,1} & Q_{3,2} & Q_{3,3} \end{pmatrix}$$

This simple model was chosen due to model specification issues with state space models, and was fitted using the EM algorithm in the astsa R package. To check the model fit, see the plot of the standardized residuals in Figure 11. The residuals for hospitalizations indicate a good fit, but there are some extreme outliers for deaths and ED visits that indicate problems with this model.

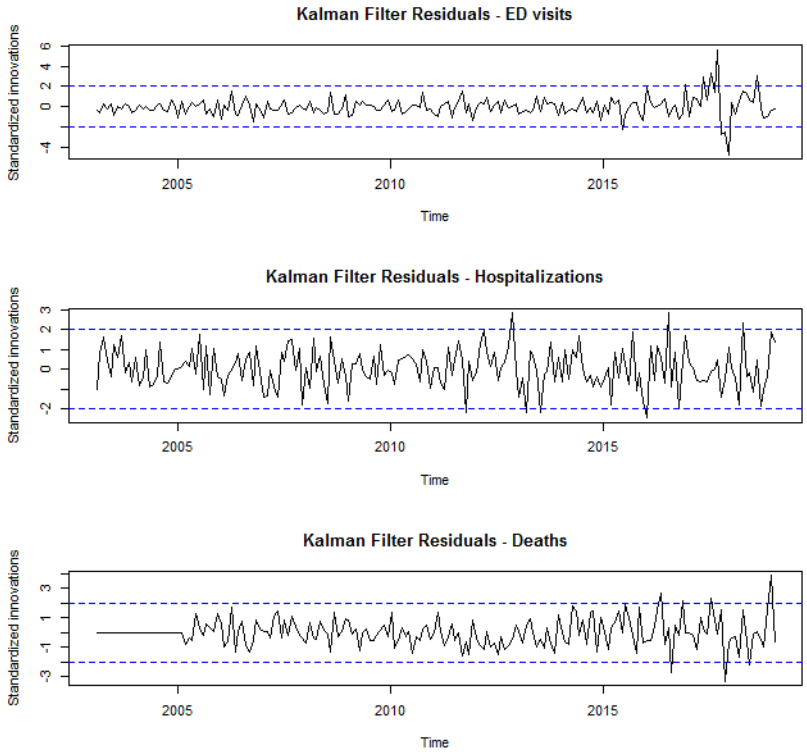

Figure 11: Standardized innovations for all three opioids series. The dotted blue lines indicate 95% confidence intervals for a standard normal distribution. The plots for ED visits and deaths indicate a poorly-fit model.

Since the volatility appears to increase greatly in the residuals around 2017, a time-varying covariance matrix, Q_t, was explored. That is, the model is given by:

$$\boldsymbol{y_t} = Z_t \boldsymbol{x_t} + \varepsilon_t, \quad \boldsymbol{x_t} = T_t \boldsymbol{x_{t-1}} + \eta_t,$$

where the system is the same as before, except for the covariance matrix for the state disturbance, Q_t, will be allowed to change starting at time $t = 172$,

which corresponds to a large outlier in the residuals from the first model.

$$Q_t = \begin{cases} (Q_{i,j}), & t < 172, \\ (Q'_{i,j}), & t \geq 172, \end{cases} \qquad i \geq j = 1, 2, 3$$

This model is much more complicated, so the MARSS package was used for fitting. Diagnostics checks for this model fit are included in Figure 12 below. Due to difficulties with the Ljung-Box statistic for large numbers of estimated parameters, this was omitted from the case study. Instead, the ACF for the standardized innovations will be used, to establish that they are serially uncorrelated.

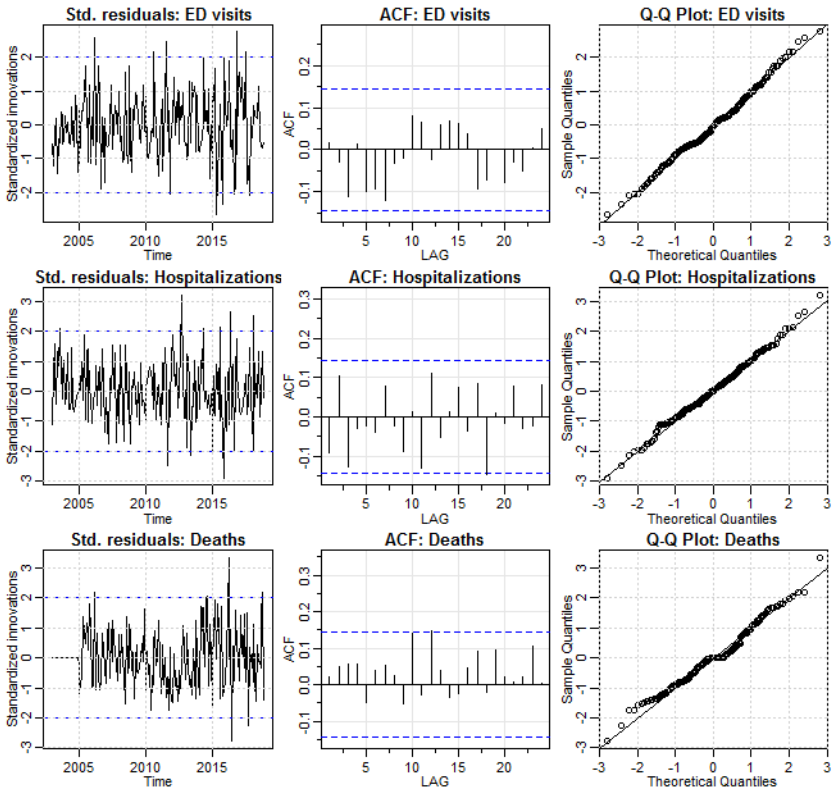

Figure 12: Diagnostics checking for model with time-varying Q_t.

The hypotheses of homoscedasticity were rejected for the ED visits and for

the deaths. The H statistics for each test are included below:

$$ED\,visits : H(64) = 2.002 > F_{64,64} = 1.639$$

$$Hospitalizations : H(64) = 1.356 < F_{64,64} = 1.639$$

$$Deaths : H(64) = 2.094 > F_{64,64} = 1.639$$

Note that for deaths, since the first 24 months of data are missing, these were be excluded from this test, since the innovations are equal to zero. Therefore,

$$H(h) = \frac{\sum_{t=n-h+1}^{n} (v_t^s)^2}{\sum_{t=25}^{25+h} (v_t^s)^2}$$

To explore fixing the heteroscedasticity in the residuals for ED visits and deaths, an input variable was added to the observation equation:

$$\boldsymbol{y_t} = Z_t \boldsymbol{x_t} + \Upsilon_t u_t + \varepsilon_t, \quad \boldsymbol{x_t} = T_t \boldsymbol{x_{t-1}} + \eta_t,$$

where all variables are the same as before, with Q_t changing at time 172. Here, a fixed input, $u_t = (I_t, 0, 0)^T$, was added, with

$$I_t = \begin{cases} 0, & t < 172 \\ 1, & t \geq 172 \end{cases}$$

This is called an *intervention variable* starting at $t = 172$, which will be applied to the ED visits.

Again, fitting this model in R using the MARSS package takes a long time. Diagnostic checking was conducted on this model fit, and a similar problem arises in the residuals. The results for the tests for homoscesasticity are included below:

$$ED\,visits : H(64) = 1.870 > F_{64,64} = 1.639$$

$$Hospitalizations : H(64) = 1.517 < F_{64,64} = 1.639$$

$$Deaths : H(64) = 2.200 > F_{64,64} = 1.639$$

Therefore, the variances observed in the Kalman filter innovations are not constant. Unfortunately, due to time constraints, a model was not found with homoscedastic innovations.

Other diagnostics for this model are included in Figure 13, which indicate a good model fit. Therefore, this model was used as a tentative model for the opioids data. Figure 14 contains plots of smoothed states for each variable. Note that the deaths prior to 2005 have been estimated - there appears to have been approximately 10 deaths from opioid poisonings per month.

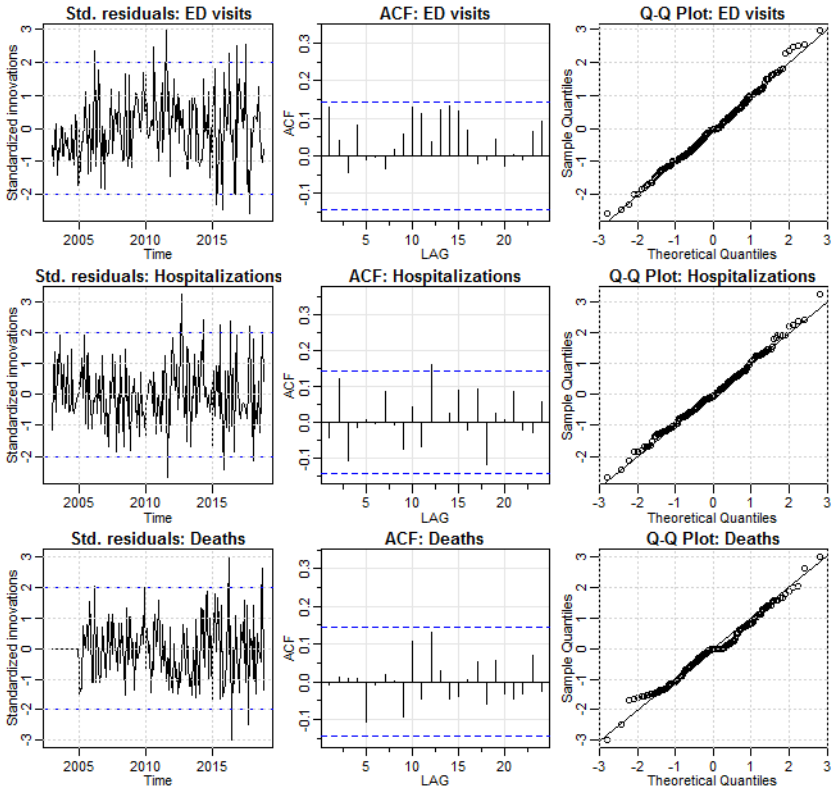

Figure 13: Diagnostics checking for model with time-varying Q_t and an intervention variable.

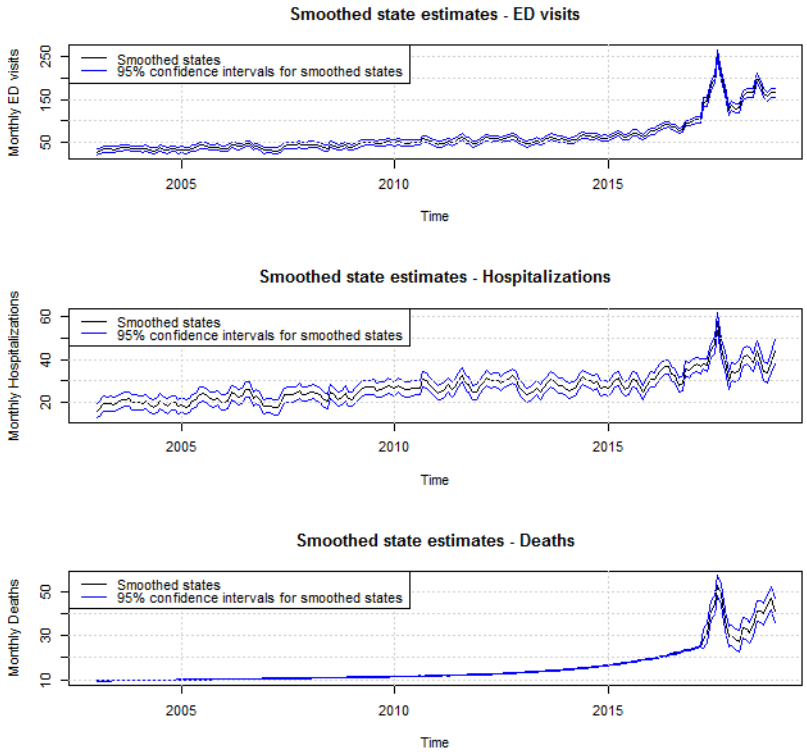

Figure 14: Smoothed state estimates for opioids data.

The estimated parameter matrices are included below:

$$R_t = \begin{pmatrix} 33.48 & 0 & 0 \\ 0 & 25.08 & 0 \\ 0 & 0 & 14.93 \end{pmatrix}, \quad T_t = \begin{pmatrix} 0.64 & 0.01 & 1.46 \\ -0.20 & 1.04 & 0.72 \\ 0.02 & -0.03 & 0.99 \end{pmatrix},$$

$$Q_{t:t<172} = \begin{pmatrix} 25.15 & 12.48 & -0.30 \\ 12.48 & 6.19 & -0.15 \\ -0.30 & -0.15 & 0.004 \end{pmatrix},$$

$$Q_{t:t\geq 172} = \begin{pmatrix} 727.04 & 137.54 & 132.23 \\ 137.54 & 26.36 & 22.93 \\ 132.23 & 22.93 & 36.90 \end{pmatrix}$$

$$\Upsilon_t = \begin{pmatrix} 29.68 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

By looking at Q_t, the state volatility increases greatly at $t = 172$. This indicates a change in the data at this time. In addition to the increase in volatility, the coefficient for the intervention variable suggests that ED visits increase by a fixed value of around 30 per month after $t = 172$.

Finally, using this model, the data for 2019 was forecasted, along with 90% prediction intervals. These are given in Figure 15. Despite not clearing all diagnostics checks, this model performs fairly well with the 2019 data. For ED visits and deaths, all data from 2019 fall within the prediction intervals. For hospitalizations, there are two points from 2019 that fall outside of the interval.

5 Conclusion

In conclusion, state space models provide an alternative to ARIMA approaches, and they do not rely on an assumption of stationarity. In this report, state space models were fitted to nonstationary data, and yielded fairly accurate forecasts.

While SSMs do not rely on stationarity, they rely greatly on model formulation. Since the SSM form is so broad, many different models could be specified for an individual problem. It is up to the investigator to carefully select an appropriate model for their data.

In the case study, it is likely that there is a more appropriate SSM that can be fitted to the data. Unfortunately, time constraints and the need for an identifiable model limited the number of options that were explored. A model that perfectly satisfies all diagnostics checks remains to be found.

Ideally, the forecasts in the case study would also be compared to the forecasts

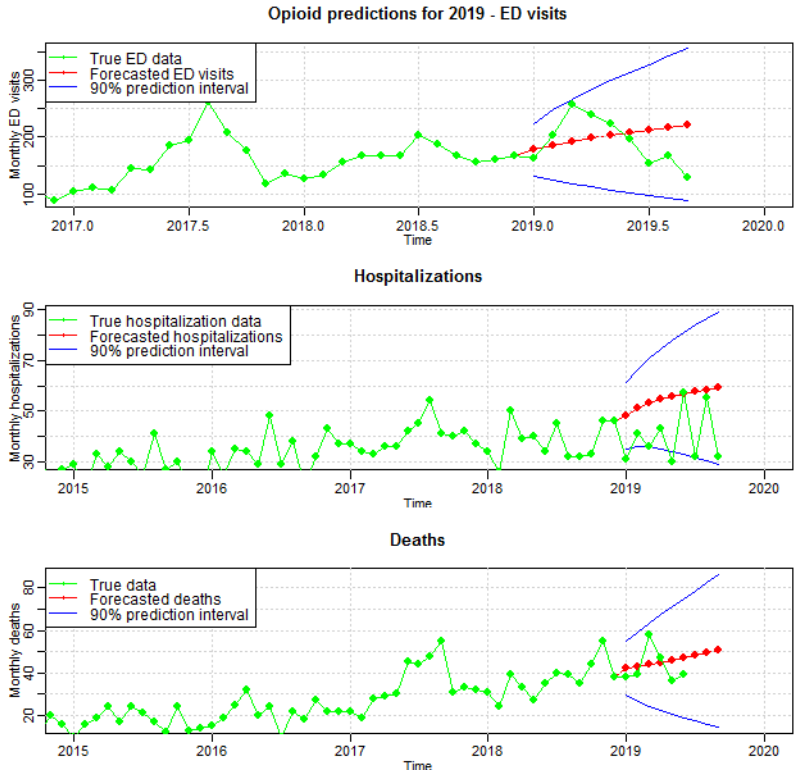

Figure 15: Forecasted 2019 data for opioids series.

for a vector-valued time series model fitted using an ARIMA approach. Such a comparison, however, relies on information beyond what was covered in this report.

Appendices

A R code from Section 2

```
###Use astsa for ARIMA
library(astsa)
library(stats)

###Illustration of stationarity - varve dataset
plot(varve, main="Glacial varve time series", ylab="
    Varve thickness")
acf(varve, main="ACF of varve series", ylab="Sample ACF
    ")
par(mfrow=c(2,1), mar=c(4,5,4,4))
plot(diff(log(varve)), main = "a) Differenced,
    log-transformed varve series", ylab = "Change in
    log(varve) since
    previous year")
acf(diff(log(varve)), main="b) ACF of differenced,
    log-transformed varve series", ylab="Sample ACF")
#=======================================

##Random walk example
par(mfrow=c(2,2))
set.seed(888777)
x<-rnorm(100)
y<-cumsum(x)
plot.ts(y, main="a) Random walk", xlab="Time", type="l"
    )
plot.ts(diff(y), main = "b) Differenced random walk",
    xlab = "Time",
          ylab = "yt - y(t-1)", type="l")

acf1(y, main="c) ACF of RW")
```

```
acf1(diff(y), main="d) ACF of differenced RW")
#=============================================================

graphics.off()
####Collisions example
###Set directory for .csv files
dir <- ""
##Monthly data from January 1999 to December 2018
coll_sev<-read.csv(file=paste(dir,"Coll_sev.csv",sep=""
    ))
coll.ts<-as.ts(ts(coll_sev,start=c(1999,1),frequency
    =12))
n=length(coll.ts)

###Plot the collisions data, with seasonally-
    differenced series
par(mfrow = c(2,2))
plot.ts(coll.ts[,"Num_inj"],main="a) Number of Canadian
    car accidents
resulting in injury, 1999 - 2018", cex.main=0.98,
         ylab="Monthly collisions")
plot.ts(diff(coll.ts[,"Num_inj"],12), main = "b)
    Seasonally differenced collisions data",
         ylab = "Change since previous year")

###Plot ACFs for both series
acf(coll.ts[,1], lag.max=50,
     main = "c) ACF of collisions dataset")
acf(diff(coll.ts[,1],12), lag.max = 50,
     main="d) ACF of seasonally-differenced data")
```

```
###Remove the data for 2016-17
inj_2017<-as.ts(ts(coll.ts[(n-23):n,1], start=c(2016,1)
    , frequency=12))
coll.ts<-as.ts(ts(coll.ts[1:(n-24),],class="mts",start=
    c(1999,1),frequency=12))
num=length(coll.ts[,"Num_inj"])
y1=coll.ts[,"Num_inj"]

###ARIMA method
par(mfrow=c(1,1))
acf2(diff(y1,12))

###Fit a SARIMA model to the data
sarma.1<-sarima(y1, p = 1, d = 0, q = 1, S=12, P=1, D
    =1, Q=2)
sarma.1$fit

##Forecast 2 years ahead, with 95% prediction intervals
for1<-sarima.for(y1, n.ahead = 24, p = 1, d = 0, q = 1,
     S=12, P=1, D=1, Q=2)
sarma.up<-for1$pred + 1.96 * for1$se
sarma.lo<-for1$pred - 1.96 * for1$se

###Plot the forecasts against the true 2016-17 data
par(mar=c(4,4,4,4))
plot.ts(inj_2017, main = "Forecasted collisions model
    using Box-Jenkins approach, 2016-17", col="green",
     lwd=2, ylim = c(min(sarma.lo), max(y1)),
     ylab="Number of injury-causing collisions")
```

```
grid()
points(inj_2017, col="green", pch=20)
lines(for1$pred, col="red", lwd=2)
points(for1$pred, col="red", pch=20)
lines(sarma.lo, col="blue")
lines(sarma.up, col="blue")
legend(x="topleft", legend = c("Actual 2016-17 data",
    "2016-17 forecast", "95% prediction interval"),
    col=c("green", "red", "blue"), lty=c(1,1,1),
    pch=c(20,20,NA))

##Compute the sum of squared forecast errors
SSE_diff<-sum((for1$pred-inj_2017)^2)
SSE_diff
###Print it on the plot
text(x = 2017.5, y=12500, labels = paste("SSE:",round(
   SSE_diff,2)), cex=1.5)
#=============================================
```

B R code from Section 3

```
#Section 3=====================================================
#Fitting a SSM to collisions data==============================
###State space model - collisions
library(KFAS)
library(astsa)
graphics.off()
```

```
##Set directory
dir <- ""

###Import the time series - injury-causing collisions
    in Canada
##Monthly data from January 1999 to December 2017
coll_sev<-read.csv(file=paste(dir,"Coll_sev.csv",sep=""
    ))
coll.ts<-as.ts(ts(coll_sev,start=c(1999,1),frequency
    =12))
n=length(coll.ts)

###Remove the data for 2016-17 - will be compared to
    model forecasts
inj_2017<-as.ts(ts(coll.ts[(n-23):n,1], start=c(2016,1)
    , frequency=12))
y1<-as.ts(ts(coll.ts[1:(n-24),],start=c(1999,1),
    frequency=12))
num=length(y1)

###Fit the seasonal model using KFAS
model_SS = SSModel(y1 ~ SSMseasonal(period=12, Q=NA,
    sea.type="dummy", a1=rep(0,11), index=1) +
                       SSMtrend(degree=2, Q=list(NA,NA),
    a1=c(0,0)), H=NA)

##Calculate the maximum likelihood estimates and print
    them
SS_coll<-fitSSM(model_SS, inits = c(1,1,1,1))$model
print(paste("Observation disturbance:", SS_coll$H))
```

```
print(paste("State disturbance (level):", SS_coll$Q
    [1,1,1]))
print(paste("State disturbance (trend):", SS_coll$Q
    [2,2,1]))
print(paste("State disturbance (seasonal):", SS_coll$Q
    [3,3,1]))

##Run the Kalman smoother
fit_coll<-KFS(SS_coll,filtering="state",smoothing="
    state")

###Plot the different components
par(mfrow=c(2,2), mai=c(.5,.5,.5,0.1))
###Level component
plot(fit_coll$alphahat[,"level"], main="Level", ylab="
    Level",
      lwd=1.5, ylim=c(9000,14000))
grid()
lines(fit_coll$alphahat[,"level"] + 1.96 * sqrt(fit_
    coll$V[1,1,]), col="blue")
lines(fit_coll$alphahat[,"level"] - 1.96 * sqrt(fit_
    coll$V[1,1,]), col="blue")

#Slope component
plot(fit_coll$alphahat[,"slope"], main="Slope", ylab="
    Slope")
grid()

##Seasonal component
plot(fit_coll$alphahat[,"sea_dummy1"],
```

```
      main="Seasonal component",ylab="Seasonal", ylim=c
    (-4000,2000))
grid()

#Irregular component
irr_1<-y1-fit_coll$alphahat[,"level"]-fit_coll$alphahat
    [,"sea_dummy1"]
plot(irr_1,main="Irregular component", ylab="Irregular"
    )
grid()

###Calculate the standardized innovations and do
    diagnostic checking
std_resid<-fit_coll$v[(fit_coll$d+1):num]/sqrt(fit_coll
    $F[(fit_coll$d+1):num])
###Number of estimated parameters
num_par<-4

par(mfrow=c(2,2))
###Plot of standardized residuals
plot(ts(std_resid,start=c(1999,1),frequency=12),main="(
    a) Standardized residuals",
      type="l", ylab="Standardized innovations")
grid()

##QQplot
qqnorm(std_resid, main="(b) Q-Q Plot")
grid()
abline(a=0,b=1)
```

```
##Test for independence - Ljung-Box statistics
lb <- rep(0,20)
for(i in 5:25){
    lb[i] <- Box.test(std_resid, lag=(i),fitdf=num_par
   -1)$p.value
}
plot(x=(5:25),y=lb[5:25], ylim=c(0,1),main="(c) P-
   values for Ljung-Box statistic",xlab="Lag",
     ylab="P-value")
abline(a=0.05,b=0, col="blue",lty=2)
grid()
##Sample ACF
acf1(std_resid, main = "(d) ACF of standardized
   innovations")

###Homoscedasticity
resid_all<-fit_coll$v[1:num]/sqrt(fit_coll$F[1:num])

h<-round((num-fit_coll$d)/3,0)
H_num<-sum(resid_all[(num-h+1):num]^2)
H_den<-sum(resid_all[(fit_coll$d+1):(fit_coll$d+h)]^2)
H <- H_num/H_den
F.hh <- qf(0.975, df1 = h, df2 = h)
if(H<1){
    print(paste("Reject null hypothesis of
   homoscedasticity:", (1/H)>=F.hh))
} else {
    print(paste("Reject null hypothesis of
   homoscedasticity:", H>=F.hh))
}
```

```
###Predict 2016-17 data
##Compute forecasts with confidence intervals
par(mfrow=c(1,1), mar=c(4,4,4,4))
y_hat<-predict(SS_coll, n.ahead = 24, interval = "
    prediction")

y_ahead<-as.ts(ts(y_hat[,"fit"], start = c(2016,1),
    frequency = 12))
y_up<-as.ts(ts(y_hat[,"upr"], start = c(2016,1),
    frequency = 12))
y_low<-as.ts(ts(y_hat[,"lwr"], start = c(2016,1),
    frequency = 12))

plot.ts(inj_2017, xlim = c(2016, 2018), ylim=c(min(y_
    low), max(y1)), main = "Prediction intervals, 2
    years ahead",
      ylab="Monthly injury-causing collisions", col="
    green", lwd=2)
grid()
points(inj_2017, col="green", cex=2, pch=20);
lines(y_up, col="blue", lwd=1)
lines(y_low, col="blue", lwd=1)
points(y_ahead, pch=20, cex=2, col="red"); lines(y_
    ahead, lwd=2, col="red")
legend(x="topleft", legend = c("Actual 2016-17 data",
                               "2016-17 forecast",
                               "95% Prediction Interval
    "),
       col=c("green", "red", "blue"),
```

```
        lty=c(1,1,1), pch=c(20,20,NA), cex=1.15)

###Compute the sum of squared forecast errors
SSE_ssm<-sum((y_ahead-inj_2017)^2)
text(x = 2017.5, y=14000, labels = paste("SSE:",round(
    SSE_ssm,2)), cex=1.5)
```

C R code from Case Study

```
library(astsa)
library(MARSS)
##For square-rooting matrices
library(expm)

###Set directory for opioids data
dir <- ""

###Import the data
opi<-read.csv(file=paste(dir,"opioids.csv",sep=""))

###Remove the 2019 data - this will be forecasted later
y<-as.ts(ts(opi[1:192,],start=c(2003,1),frequency=12))
ahead<-as.ts(ts(opi[193:201,], start=c(2019,1),
    frequency = 12))

###Create plots of the opioids data
par(mfrow=c(3,1))
plot.ts(y[,1], main="Opioid-related ED visits", ylab="
    Monthly number of visits",
```

```
        xlim=c(2003,2020))
lines(ahead[,1], col="blue")
legend(x="topleft", legend = c("Opioid series", "2019
    data"),
       col=c("black", "blue"), lty=c(1,1))

plot.ts(y[,2], main="Opioid-related hospitalizations",
        ylab="Monthly hospitalizations", xlim=c
    (2003,2020))
lines(ahead[,2], col="blue")
legend(x="topleft", legend = c("Opioid series", "2019
    data"),
       col=c("black", "blue"), lty=c(1,1))

plot.ts(y[,3], main="Opioid-related deaths",
        ylab="Monthly deaths", xlim=c(2003,2020))
lines(ahead[,3], col="blue")
legend(x="topleft", legend = c("Opioid series", "2019
    data"),
       col=c("black", "blue"), lty=c(1,1))

##Count the remaining number of data points
num=nrow(y)

###Missing data in astsa package is represented by
    zeros
for(i in 1:num){ if(is.na(y[i,3])) y[i,3]<-0}

###Set the observation matriix equal to zero where the
    death data is missing
```

```
Z = array(0, dim=c(3,3,num))
for(k in 1:num) {
  if (y[k,3]==0) {Z[,,k]= diag(c(1,1,0))}
  else {Z[,,k]= diag(c(1,1,1))}
}

###Initialize parameters for SSM
T = diag(1,3)
a1 = c(mean(y[1:50,1]), mean(y[1:50,2]), mean(y
    [25:75,3]))
P1 = diag(c(0.1, 0.1, 0.1), 3)
cQ = diag(c(.1, .1, .1), 3)
cH = diag(c(0.1,0.1,0.1))

##Run astsa EM algorithm
(em = EM1(num, y, A = Z, mu0 = a1, Sigma0 = P1, Phi = T
    , cQ = cQ, cR = cH, 300, .0001))

###Run Kalman filter and smoother
kf_gta<-Kfilter1(num, y, A = Z, mu0 = em$mu0, Sigma0 =
    em$Sigma0,
                 Phi = em$Phi, cQ = chol(em$Q), cR =
    chol(em$R),
                 input=0, Ups = 0, Gam = 0)

ks_gta<-Ksmooth1(num, y, A = Z, mu0 = em$mu0, Sigma0 =
    em$Sigma0,
                 Phi = em$Phi, cQ = chol(em$Q), cR =
    chol(em$R),
                 input=0, Ups = 0, Gam = 0)
```

```
###Diagnostics checking
std_innov<-array(0, dim=c(4,1,num))

###Standardize the residuals
R<-array(0,dim=c(3,3,num))
z<-array(0,dim=c(3,1,num))
for(t in 1:num){
  R[,,t]<-sqrtm(solve(kf_gta$sig[,,t]))
  z[,,t]<-R[,,t]%*%kf_gta$innov[,,t]
}

par(mfrow=c(3,1))
###Plot residuals
plot(ts(z[1,1,],start=c(2003.1),frequency=12), main="
    Kalman Filter Residuals - ED visits",
      ylab = "Standardized innovations", cex=2)
abline(a=2, b=0, col="blue", lty=2); abline(a=-2, b=0,
    col="blue", lty=2)
plot(ts(z[2,1,],start=c(2003.1),frequency=12), main="
    Kalman Filter Residuals - Hospitalizations",
      ylab = "Standardized innovations")
abline(a=2, b=0, col="blue", lty=2); abline(a=-2, b=0,
    col="blue", lty=2)
plot(ts(z[3,1,],start=c(2003.1),frequency=12), main="
    Kalman Filter Residuals - Deaths",
      ylab = "Standardized innovations")
abline(a=2, b=0, col="blue", lty=2); abline(a=-2, b=0,
    col="blue", lty=2)
```

```
#=============================================================
###Trying different model - time-varying Q
##Need to use the MARSS package to run the EM algorithm
#Format the data for MARSS
y<-as.ts(ts(opi[1:192,],start=c(2003,1),frequency=12))
dat = t(y)

###Initialize the parameters
H=matrix(list(1,0,0,1,0,0,0,1),3,3)
A=matrix(0,3,1)
R=matrix(list(0),3,3); diag(R)=c("r1","r2","r3")
B = matrix(list("b11","b21","b31","b12","b22","b32","
    b13","b23","b33"),3,3)
U=matrix(0,3,1)
Q= array(0, dim = c(3,3,num))
###Time-varying Q with shift at time = 172
for(i in 1:num){
  if(i < 172){
    Q[,,i]<-matrix(c("q11","q21","q31","q21","q22","q32
    ","q31","q32","q33"),3,3)
  } else {
    Q[,,i]<-matrix(c("Q11","Q21","Q31","Q21","Q22","Q32
    ","Q31","Q32","Q33"),3,3)
  }
}

###Use the initial state information from previous run
    of EM
##To reduce the number of parameters that need to be
    estimated
```

```
x0=em$mu0
V0=matrix(list(0),3,3)

model.gen=list(Z=H, A=A, R=R, B=B, U=U, Q=Q, x0=x0, V0=
    V0, tinitx=0)
###Time-consuming step - run the EM algorithm
kem = MARSS(dat, model=model.gen, control=list(maxit =
    10000, trace=1, safe=TRUE))

##Import fitted information from MARSS function
coef1<-kem$coef; states1<-kem$states
se1<-kem$states.se; innov1<-kem$Innov
sigma1<-kem$Sigma; xtt1<-kem$xtt; Ptt1<-kem$Vtt

###Standardize the innovations from the KF
R<-array(0,dim=c(3,3,num))
z<-array(0,dim=c(3,1,num))
for(t in 1:num){
  R[,,t]<-sqrtm(solve(sigma1[,,t]))
  z[,,t]<-R[,,t]%*%(innov1[,t])
}

###Diagnostics checking
par(mfrow=c(3,3))
for(j in 1:3){
  if(j==1){lab="ED visits"} else if(j==2){lab="
    Hospitalizations"} else{lab="Deaths"}
  ###Plot of standardized residuals
  plot(ts(z[j,1,],start=c(2003,1),frequency=12),main=
    paste("Std. residuals:",lab),
```

```
        type="l", ylab="Standardized innovations")
  abline(a=2, b=0, col="blue", lty=2); abline(a=-2, b
   =0, col="blue", lty=2)
  grid()

  acf1(z[j,1,], main=paste("ACF:",lab))

  ##QQplot
  qqnorm(z[j,1,], main=paste("Q-Q Plot:",lab))
  grid()
  abline(a=0,b=1)

  ###Homoscedasticity
  h<-round((num)/3,0)
  H_num<-sum(z[j,1,(num-h+1):num]^2)
  ###For deaths, ignore innovations for missing cases
  if(j!=3){H_den<-sum(z[j,1,(1:h)]^2)} else{H_den<-sum(
   z[j,1,(25:(h+25))]^2)}
  H <- H_num/H_den
  F.hh <- qf(0.975, df1 = h, df2 = h)
  if(H<1){
    print(paste("Reject null hypothesis of
   homoscedasticity:", (1/H)>=F.hh, H))
  } else {
    print(paste("Reject null hypothesis of
   homoscedasticity:", H>=F.hh, H))
  }
}

###Intervention variable and time-varying Q
```

```
###Initialize the parameters
H=matrix(list(1,0,0,1,0,0,0,1),3,3)
A=matrix(0,3,1)
R=matrix(list(0),3,3); diag(R)=c("r1","r2","r3")
B = matrix(list("b11","b21","b31","b12","b22","b32","
    b13","b23","b33"),3,3)
U=matrix(0,3,1)
Q= array(0, dim = c(3,3,num))
###Time-varying Q with shift at time = 172
for(i in 1:num){
  if(i < 172){
    Q[,,i]<-matrix(c("q11","q21","q31","q21","q22","q32
    ","q31","q32","q33"),3,3)
  } else {
    Q[,,i]<-matrix(c("Q11","Q21","Q31","Q21","Q22","Q32
    ","Q31","Q32","Q33"),3,3)
  }
}

###Use the initial state information from previous run
    of EM
##To reduce the number of parameters that need to be
    estimated
x0=em$mu0
V0=matrix(list(0),3,3)

###INTERVENTION VARIABLE
inp = cbind(c(rep(0,171),rep(1,num-171)),rep(0,num),
    rep(0,num))
d = t(inp)
```

```
D = matrix(list(0),3,3); D[1,1] = "D1"

model.input=list(Z=H, A=A, R=R, B=B, U=U, Q=Q, d=d, D=D
    , x0=x0, V0=V0, tinitx=0)
###Time-consuming step - run the EM algorithm
kem.i = MARSS(dat, model=model.input, control=list(
    maxit = 10000, trace=1, safe=TRUE))

coef2<-kem.i$coef; states2<-kem.i$states
se2<-kem.i$states.se; innov2<-kem.i$Innov
sigma2<-kem.i$Sigma; xtt2<-kem.i$xtt; Ptt2<-kem.i$Vtt

###Diagnostics checking
par(mfrow=c(3,3))
for(j in 1:3){
  if(j==1){lab="ED visits"} else if(j==2){lab="
   Hospitalizations"} else{lab="Deaths"}
  ###Plot of standardized residuals
  plot(ts(z[j,1,],start=c(2003,1),frequency=12),main=
   paste("Std. residuals:",lab),
        type="l", ylab="Standardized innovations")
  abline(a=2, b=0, col="blue", lty=2); abline(a=-2, b
   =0, col="blue", lty=2)
  grid()

  acf1(z[j,1,], main=paste("ACF:",lab))

  ##QQplot
  qqnorm(z[j,1,], main=paste("Q-Q Plot:",lab))
  grid()
```

```
  abline(a=0,b=1)

  ###Homoscedasticity
  h<-round((num)/3,0)
  H_num<-sum(z[j,1,(num-h+1):num]^2)
  ###For deaths, ignore innovations for missing cases
  if(j!=3){H_den<-sum(z[j,1,(1:h)]^2)} else{H_den<-sum(
   z[j,1,(25:(h+25))]^2)}
  H <- H_num/H_den
  F.hh <- qf(0.975, df1 = h, df2 = h)
  if(H<1){
    print(paste("Reject null hypothesis of
   homoscedasticity:", (1/H)>=F.hh, H))
  } else {
    print(paste("Reject null hypothesis of
   homoscedasticity:", H>=F.hh, H))
  }
}
###Smoothed states with confidence intervals
###95% confidence intervals
CI_mult=1.96
par(mfrow = c(3,1))

#ED visits
e_upper<-ts(states2[1,]+(coef2["D.D1",]*inp[,1]) + CI_
    mult * se2[1,], start=c(2003,1), frequency=12)
e_lower<-ts(states2[1,]+(coef2["D.D1",]*inp[,1]) - CI_
    mult * se2[1,], start=c(2003,1), frequency=12)
plot(ts(states2[1,]+(coef2["D.D1",]*inp[,1]), start=c
    (2003,1), frequency=12),
```

```
      type="l", ylim=c(min(e_lower),max(e_upper)),
      main="Smoothed state estimates - ED visits", ylab=
   "Monthly ED visits")
lines(e_upper, col="blue")
lines(e_lower, col="blue")
grid()
legend(x="topleft", legend = list("Smoothed states", "
   95% confidence intervals for smoothed states"),
        col=c("black", "blue"), lty=c(1,1))

#Hospitalizations
h_upper<-ts(states2[2,] + CI_mult * se2[2,],start=c
   (2003,1), frequency=12)
h_lower<-ts(states2[2,] - CI_mult * se2[2,],start=c
   (2003,1), frequency=12)
plot(ts(states2[2,],start=c(2003,1), frequency=12),
   type="l", ylim=c(min(h_lower),max(h_upper)),
      main="Smoothed state estimates - Hospitalizations"
   , ylab="Monthly Hospitalizations")
lines(h_upper, col="blue")
lines(h_lower, col="blue")
grid()
legend(x="topleft", legend = list("Smoothed states", "
   95% confidence intervals for smoothed states"),
        col=c("black", "blue"), lty=c(1,1))

#Deaths
d_upper<-ts(states2[3,] + CI_mult * se2[3,],start=c
   (2003,1), frequency=12)
d_lower<-ts(states2[3,] - CI_mult * se2[3,],start=c
```

```
    (2003,1), frequency=12)
plot(ts(states2[3,],start=c(2003,1), frequency=12),
    type="l", ylim=c(min(d_lower),max(d_upper)),
      main="Smoothed state estimates - Deaths", ylab="
    Monthly Deaths")
lines(d_upper, col="blue")
lines(d_lower, col="blue")
grid()
legend(x="topleft", legend = list("Smoothed states", "
    95% confidence intervals for smoothed states"),
        col=c("black", "blue"), lty=c(1,1))

# Forecast 2 years ahead
n.ahead = 9;

###Using output from EM to forecast
Q= array(0, dim = c(3,3,num))
for(i in 1:num){
  if(i < 172){
    Q[,,i]<-matrix(c(coef2["Q.q11",],coef2["Q.q21",],
    coef2["Q.q31",],
                     coef2["Q.q21",], coef2["Q.q22",],
    coef2["Q.q32",],
                     coef2["Q.q31",], coef2["Q.q32",],
    coef2["Q.q33",]),3,3)
  } else {
    Q[,,i]<-matrix(c(coef2["Q.Q11",],coef2["Q.Q21",],
    coef2["Q.Q31",],
                     coef2["Q.Q21",], coef2["Q.Q22",],
    coef2["Q.Q32",],
```

```
                        coef2["Q.Q31",], coef2["Q.Q32",],
    coef2["Q.Q33",]),3,3)
  }
}
R = diag(c(coef2["R.r1",], coef2["R.r2",], coef2["R.r3"
    ,]))
##Transition matrix
Phi = matrix(c(coef2["B.b11",], coef2["B.b21",], coef2[
    "B.b31",],
                coef2["B.b12",], coef2["B.b22",], coef2[
    "B.b32",],
                coef2["B.b13",], coef2["B.b23",], coef2[
    "B.b33",]), nrow=3)
Z<-diag(1,3)

###Starting point for forecast - Kalman filter output
x00 = xtt2[,num]; P00 = Ptt2[,,num]
###Prediction matrix
pred<-matrix(c(rep(0,n.ahead),rep(0,n.ahead),rep(0,n.
    ahead)), nrow=3)
###Array for square-rooted prediction errors
P = array(0, dim=c(3,3,n.ahead))
for (m in 1:n.ahead){
  xp = Phi%*%x00 ; Pp = Phi%*%P00%*%t(Phi)+Q[,,num]
  P[,,m] = sqrt(Z%*%Pp%*%t(Z)+R)
  pred[,m] = Z%*%xp
  x00 = xp; P00 = Pp
}

###Plot the forecasts
```

```
par(mfrow = c(3,1), mai=c(.3,.6,.5,.1))
#ED visits
x1_ahead<-ts(append(y[,1],t(pred[1,]+coef2["D.D1",])),
    start=c(2003,1), frequency = 12)
x1_true<-ts(append(y[,1],ahead[,1]), start=c(2003,1),
    frequency = 12)
x1_upper<-ts(pred[1,]+(coef2["D.D1",]) + 1.64 * (P
    [1,1,]), start=c(2019,1), frequency = 12)
x1_lower<-ts(pred[1,] +(coef2["D.D1",]) - 1.64 * (P
    [1,1,]), start=c(2019,1), frequency = 12)
plot.ts(x1_ahead, xlim=c(2017,2020), ylim=c(min(x1_
    lower),max(x1_upper)),
         main = "Opioid predictions for 2019 - ED visits
    ", ylab="Monthly ED visits", col="red")
grid()
points(x1_ahead, col="red", cex=2, pch=20)
lines(x1_upper, col="blue"); lines(x1_lower, col="blue"
    )
##True values
lines(x1_true, col="green")
points(x1_true, col="green", pch=20, cex=2)
legend(x="topleft", legend = list("True ED data", "
    Forecasted ED visits", "90% prediction interval"),
        col=c("green", "red", "blue"), lty=c(1,1,1), pch
    =c(20,20,NA), cex=1.1)

###Hospitalizations
x2_ahead<-ts(append(y[,2],t(pred[2,])), start=c(2003,1)
    , frequency = 12)
x2_true<-ts(append(y[,2],ahead[,2]), start=c(2003,1),
```

```
    frequency = 12)
x2_upper<-ts(pred[2,] + 1.64 * (P[2,2,]), start=c
    (2019,1), frequency = 12)
x2_lower<-ts(pred[2,] - 1.64 * (P[2,2,]), start=c
    (2019,1), frequency = 12)
plot.ts(x2_ahead, xlim=c(2015,2020),ylim=c(min(x2_lower
    ),max(x2_upper)),
         main="Hospitalizations", ylab="Monthly␣
    hospitalizations", col="red")
grid()
points(x2_ahead, col="red", cex=2, pch=20)
lines(x2_upper, col="blue")
lines(x2_lower, col="blue")
##True values
lines(x2_true, col="green")
points(x2_true, col="green", pch=20, cex=2)
legend(x="topleft", legend = list("True␣hospitalization
    ␣data", "Forecasted␣hospitalizations", "90%␣
    prediction␣interval"),
        col=c("green", "red", "blue"), lty=c(1,1,1), pch
    =c(20,20,NA), cex=1.1)

###Deaths
x3_ahead<-ts(append(dat[3,],t(pred[3,])), start=c
    (2003,1), frequency = 12)
x3_true<-ts(append(y[,3],ahead[,3]), start=c(2003,1),
    frequency = 12)
x3_upper<-ts(pred[3,] + 1.64 * (P[3,3,]), start=c
    (2019,1), frequency = 12)
x3_lower<-ts(pred[3,] - 1.64 * (P[3,3,]), start=c
```

```
    (2019,1), frequency = 12)
plot.ts(x3_ahead, xlim=c(2015,2020), ylim=c(min(x3_
    lower),max(x3_upper)),
         ylab="Monthly deaths", col='red', main="Deaths"
    )
grid()
points(x3_ahead, col="red", cex=2, pch=20)
lines(x3_upper, col="blue")
lines(x3_lower, col="blue")
##True values
lines(x3_true, col="green")
points(x3_true, col="green", pch=20, cex=2)
legend(x="topleft", legend = list("True data", "
    Forecasted deaths", "90% prediction interval"),
        col=c("green", "red", "blue"), lty=c(1,1,1), pch
    =c(20,20,NA), cex=1.1)
```

D Data used in report

Monthly opioid-related harms in 4 LHINs in Ontario (Central, Central West, Missisauga Halton, and Toronto Central). NA symbolizes a month with missing data (Source: Public Health Ontario [9]).

Year	Month	ED Visits	Hospitalizations	Deaths
2003	Jan	27	11	NA
2003	Feb	24	20	NA
2003	Mar	34	26	NA
2003	Apr	32	21	NA
2003	May	37	17	NA
2003	Jun	27	25	NA

Year	Month	ED Visits	Hospitalizations	Deaths
2003	Jul	32	23	NA
2003	Aug	33	30	NA
2003	Sep	39	22	NA
2003	Oct	40	25	NA
2003	Nov	33	19	NA
2003	Dec	32	25	NA
2004	Jan	36	18	NA
2004	Feb	33	19	NA
2004	Mar	36	27	NA
2004	Apr	31	17	NA
2004	May	28	16	NA
2004	Jun	31	18	NA
2004	Jul	38	28	NA
2004	Aug	33	18	NA
2004	Sep	28	16	NA
2004	Oct	38	18	NA
2004	Nov	38	20	NA
2004	Dec	25	19	NA
2005	Jan	35	20	8
2005	Feb	25	21	4
2005	Mar	28	20	5
2005	Apr	34	26	5
2005	May	35	20	13
2005	Jun	42	32	11
2005	Jul	50	19	10
2005	Aug	41	30	13
2005	Sep	40	17	12
2005	Oct	31	28	10

Year	Month	ED Visits	Hospitalizations	Deaths
2005	Nov	45	22	16
2005	Dec	31	20	13
2006	Jan	37	15	6
2006	Feb	32	19	6
2006	Mar	56	22	18
2006	Apr	44	24	6
2006	May	35	26	10
2006	Jun	40	19	13
2006	Jul	55	26	8
2006	Aug	54	29	6
2006	Sep	32	18	7
2006	Oct	42	29	13
2006	Nov	40	23	11
2006	Dec	26	14	9
2007	Jan	35	13	9
2007	Feb	32	19	7
2007	Mar	30	14	13
2007	Apr	29	10	15
2007	May	36	22	8
2007	Jun	48	22	14
2007	Jul	41	28	10
2007	Aug	39	30	15
2007	Sep	43	23	13
2007	Oct	49	30	11
2007	Nov	47	15	10
2007	Dec	43	23	8
2008	Jan	51	18	14
2008	Feb	45	31	10

Year	Month	ED Visits	Hospitalizations	Deaths
2008	Mar	45	23	9
2008	Apr	43	27	14
2008	May	36	19	11
2008	Jun	32	12	9
2008	Jul	53	31	5
2008	Aug	42	26	16
2008	Sep	35	19	9
2008	Oct	36	25	10
2008	Nov	54	22	15
2008	Dec	40	14	15
2009	Jan	33	22	10
2009	Feb	45	23	12
2009	Mar	47	27	6
2009	Apr	54	24	11
2009	May	55	22	13
2009	Jun	55	21	10
2009	Jul	55	27	10
2009	Aug	49	19	11
2009	Sep	47	29	12
2009	Oct	51	22	14
2009	Nov	60	24	12
2009	Dec	54	23	19
2010	Jan	51	19	9
2010	Feb	60	26	11
2010	Mar	50	26	14
2010	Apr	47	27	11
2010	May	47	28	12
2010	Jun	50	28	6

Year	Month	ED Visits	Hospitalizations	Deaths
2010	Jul	51	27	10
2010	Aug	48	22	9
2010	Sep	68	32	14
2010	Oct	62	30	16
2010	Nov	58	22	12
2010	Dec	50	26	13
2011	Jan	42	25	18
2011	Feb	48	21	13
2011	Mar	52	19	9
2011	Apr	60	31	12
2011	May	46	23	15
2011	Jun	49	27	10
2011	Jul	59	34	13
2011	Aug	77	32	8
2011	Sep	60	16	11
2011	Oct	62	28	7
2011	Nov	42	21	15
2011	Dec	45	22	10
2012	Jan	52	31	8
2012	Feb	56	37	7
2012	Mar	67	34	13
2012	Apr	57	30	9
2012	May	60	34	10
2012	Jun	64	27	7
2012	Jul	53	29	11
2012	Aug	62	31	8
2012	Sep	59	36	9
2012	Oct	61	47	11

Year	Month	ED Visits	Hospitalizations	Deaths
2012	Nov	65	34	16
2012	Dec	54	25	14
2013	Jan	49	28	10
2013	Feb	45	17	14
2013	Mar	42	32	16
2013	Apr	43	32	12
2013	May	59	30	9
2013	Jun	47	17	10
2013	Jul	59	26	8
2013	Aug	60	27	14
2013	Sep	66	36	11
2013	Oct	65	26	8
2013	Nov	53	31	18
2013	Dec	62	25	14
2014	Jan	53	33	11
2014	Feb	49	25	9
2014	Mar	50	33	20
2014	Apr	51	32	20
2014	May	58	40	14
2014	Jun	74	34	20
2014	Jul	65	30	13
2014	Aug	66	31	22
2014	Sep	62	27	23
2014	Oct	72	30	12
2014	Nov	54	25	20
2014	Dec	62	27	16
2015	Jan	54	29	10
2015	Feb	66	19	16

Year	Month	ED Visits	Hospitalizations	Deaths
2015	Mar	71	33	19
2015	Apr	82	28	24
2015	May	55	34	17
2015	Jun	55	30	24
2015	Jul	61	25	21
2015	Aug	73	41	17
2015	Sep	77	27	12
2015	Oct	67	30	24
2015	Nov	52	20	13
2015	Dec	78	15	14
2016	Jan	79	34	15
2016	Feb	76	25	19
2016	Mar	80	35	25
2016	Apr	90	34	32
2016	May	101	29	20
2016	Jun	91	48	24
2016	Jul	86	29	10
2016	Aug	88	38	22
2016	Sep	71	21	18
2016	Oct	68	32	27
2016	Nov	105	43	22
2016	Dec	89	37	22
2017	Jan	104	37	22
2017	Feb	110	34	19
2017	Mar	107	33	28
2017	Apr	144	36	29
2017	May	143	36	30
2017	Jun	184	42	45

Year	Month	ED Visits	Hospitalizations	Deaths
2017	Jul	195	45	44
2017	Aug	262	54	48
2017	Sep	207	41	55
2017	Oct	175	40	31
2017	Nov	117	42	33
2017	Dec	135	37	32
2018	Jan	127	34	31
2018	Feb	133	26	24
2018	Mar	155	50	39
2018	Apr	167	39	33
2018	May	168	40	27
2018	Jun	166	34	35
2018	Jul	202	45	40
2018	Aug	187	32	39
2018	Sep	167	32	35
2018	Oct	156	33	44
2018	Nov	161	46	55
2018	Dec	166	46	38
2019	Jan	163	31	38
2019	Feb	202	41	39
2019	Mar	256	36	58
2019	Apr	240	43	47
2019	May	223	30	36
2019	Jun	197	57	39
2019	Jul	154	32	NA
2019	Aug	168	55	NA
2019	Sep	129	32	NA

Table 1: Number of injury-causing collisions in Canada from 1999-2017. Data for January to August. (Source: Transport Canada [8]).

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
1999	12,218	9,741	10,703	10,251	12,669	13,229	13,546	13,738
2000	13,135	11,105	10,142	10,743	12,199	14,048	13,792	13,698
2001	11,454	11,073	10,937	9,858	12,225	13,221	13,348	14,023
2002	12,443	10,763	12,030	10,281	11,963	13,551	13,783	14,281
2003	14,590	11,596	11,065	10,011	11,898	13,244	13,211	13,235
2004	14,442	9,946	10,024	9,390	11,231	12,805	13,049	13,027
2005	12,889	10,035	10,159	10,184	11,439	13,206	13,219	13,199
2006	11,190	10,103	10,069	9,920	11,772	13,047	13,016	13,207
2007	11,967	11,241	9,932	9,203	11,163	12,398	12,320	12,435
2008	11,013	10,314	9,205	8,468	9,818	11,101	11,242	11,588
2009	10,949	8,408	8,417	8,625	9,843	11,333	10,960	11,449
2010	9,535	8,498	7,955	8,884	10,425	11,319	11,326	11,216
2011	10,930	9,137	8,838	7,791	9,275	10,525	10,884	10,931
2012	10,421	8,545	8,899	8,190	10,094	10,782	11,078	11,122
2013	10,337	8,653	8,318	7,912	9,737	10,172	10,939	10,940
2014	11,080	8,751	8,330	6,833	8,425	9,402	10,160	10,221
2015	10,651	9,799	8,109	7,637	9,445	10,321	10,652	10,291
2016	9,775	8,742	7,841	7,921	9,458	10,449	10,358	10,480
2017	8,894	7,935	7,954	7,445	9,182	10,050	10,100	9,958

Table 2: Collisions data from September to December (Source: Transport Canada [8]).

Sep	Oct	Nov	Dec
13,115	13,021	12,904	13,543
13,498	13,119	13,102	14,703
12,648	13,416	12,780	14,031
13,582	13,404	13,764	13,982
12,823	12,945	12,915	12,958
12,571	13,070	12,057	13,536
12,529	12,488	12,951	13,259
12,489	12,996	12,486	12,218
12,068	12,058	12,356	11,467
10,941	11,135	11,275	11,467
11,272	10,878	10,288	11,019
11,019	11,187	11,265	10,976
10,867	10,734	11,214	11,213
10,798	11,148	10,755	10,991
10,744	10,948	10,701	10,955
10,332	10,541	10,615	9,925
10,132	10,326	10,202	10,270
10,409	10,199	10,306	10,593
10,143	10,098	10,405	10,548

References

[1] J. Commandeur and S.J. Koopman. *An Introduction to State Space Time Series Analysis*. Oxford University Press, 2007.

[2] J. Durbin and S.J. Koopman. *Time Series Analysis by State Space Methods*. 2nd ed. Oxford University Press, 2012.

[3] E. E. Holmes. *Derivation of an EM algorithm for constrained and unconstrained multivariate autoregressive state-space (MARSS) models*. 2012. URL: https://arxiv.org/pdf/1302.3919.pdf.

[4] J. Casals et al. *State-space methods for time series analysis: Theory, applications and software*. CRC Press, Taylor Francis Group, 2016.

[5] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications With R Examples*. 4th ed. Springer, 2016.

[6] J. Helske. *KFAS: Exponential Family State Space Models in R. Version 3.10.12*. 2017. URL: https://cran.r-project.org/web/packages/KFAS/vignettes/KFAS.pdf. (accessed: 28.03.2020).

[7] E. E. Holmes, E. J. Ward, and M. D. Scheuerell. *Analysis of multivariate time-series using the MARSS package. Version 3.10.12*. Feb. 3, 2020. URL: https://cran.r-project.org/web/packages/MARSS/vignettes/UserGuide.pdf. (accessed: 28.03.2020).

[8] Transport Canada. *National Collision Database Online*. URL: https://wwwapps2.tc.gc.ca/Saf-Sec-Sur/7/NCDB-BNDC/p.aspx?c=100-0-0&l=en. (accessed: 27.02.2020).

[9] Ontario Agency for Health Protection and Promotion. *Interactive Opioid Tool. Opioid-related morbidity and mortality in Ontario*. URL: https://www.publichealthontario.ca/en/data-and-analysis/substance-use/interactive-opioid-tool#/trends. (accessed: 01.04.2020).