Final Project

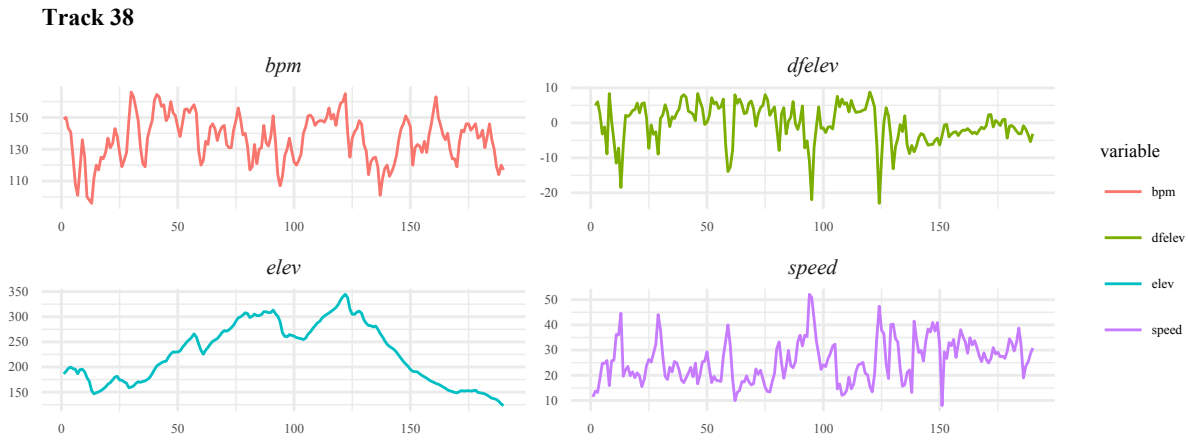
Ludovico Genovese, Luca Laringe, Alessandro Pistoni, Francesco Maria Varchetta

June 1st 2018

Exercise 1

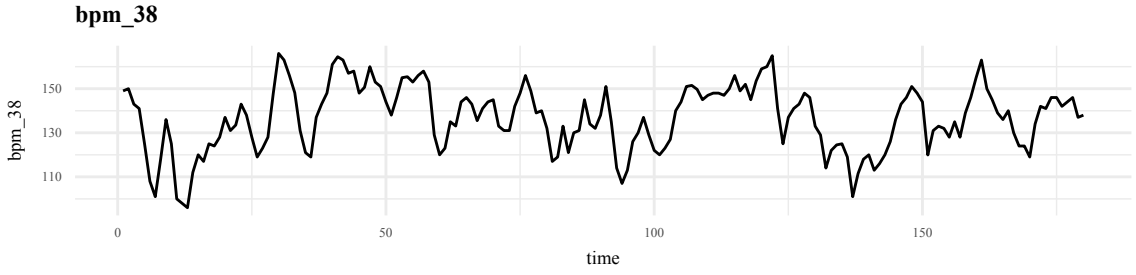
Point 1

Before starting the analysis, a preliminary remark is required. For plotting purposes, we only considered BPM, elevation, difference in elevation, and speed, thus excluding from the representation uninformative variables like "s" and "track_id_seq". Moreover, to facilitate the reading, we normalized the time variable in order to have on the horizontal axis the periods ranging from 1 to 190.



The patterns that appear from the graphs are coherent with what we intuitively expected. First, speed and change in elevation seem strongly negatively correlated (almost specular), meaning that peaks in speed are registered in correspondence of steep downhills and vice versa. Similarly, the highest heart rates correspond to upward slopes. As for trends and seasonalities, the variable that shows a clear trend is elevation. The trend is positive until about the 125th period, before becoming steadily negative. Focusing on BPM, instead, we could not detect trends or seasonalities, which would hinder stationarity.

Point 2



Even after excluding the last five minutes (10 periods), it is hard to notice relevant trends, cycles or seasonalitites. Therefore, neither classical time series decomposition nor differencing seem necessary in order to achieve stationarity. Moreover, we believe reasonable to assume that heart rate have a constant mean, since it is naturally bounded by physiological constraints. The same holds for variance and autocovariance.

(i) Checking for stationarity

```
    Augmented Dickey-Fuller Test

data:  bpm_38
Dickey-Fuller = -3.3826, Lag order = 5, p-value = 0.05947
alternative hypothesis: stationary
```

The test we are implementing is an Augmented Dickey-Fuller (ADF) test, which is aimed at testing the existence of unit roots in an ARMA model with K paramters. For instance, assuming that $(Y_t, t \geq 0)$ is distributed as AR(2) we can write the process as follows:

$$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \epsilon_t$$

$$y_t - y_{t-1} = -y_{t-1} + \alpha_2 y_{t-1} - \alpha_2 y_{t-1} + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \epsilon_t$$

readapting the equation, we obtain:

$$\Delta y_t = \gamma y_{t-1} - \alpha_2 \Delta y_{t-1} + \epsilon_t \text{ where } \epsilon_t \sim WN(0, \sigma^2) \text{ and } -1 + \alpha_1 + \alpha_2 = \gamma$$

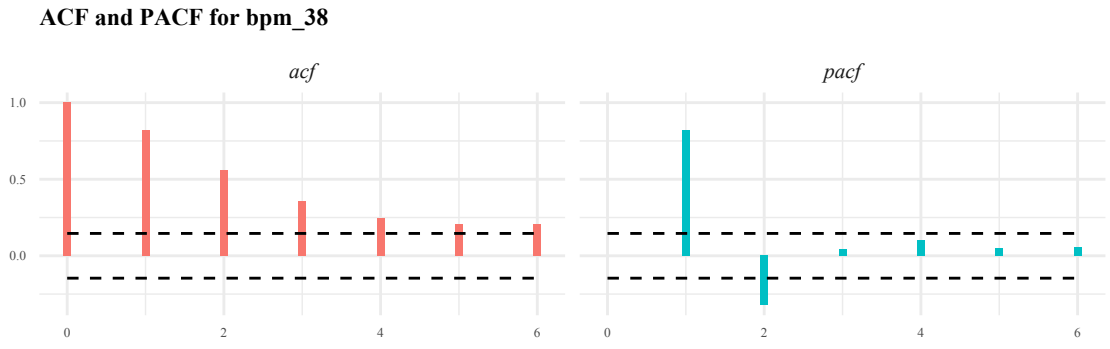
Accordingly, if one of the roots is 1, $L^* = 1$, when estimating the last model we should get $\hat{\gamma} = 0$ from the equation $\alpha(B) = 1 - \alpha_1 L - \alpha_2 L^2 = 0$. Hence, we could run a T-test with a null hypothesis $\gamma = 0$. However, when the model is non stationary or close to non-stationarity it can be shown that the distribution of the t-test is not the standard one anymore. Dickey and Fuller (1979) provided the critical values for this non standard distribution. Thus, in the case of an AR(2), the ADF command runs a T-test on γ under the null $H_0 : \gamma = 0$, where the test-statistic is $T = \frac{\hat{\gamma} - \gamma}{se(\hat{\gamma})} \sim^{H_0} ADF$

Moving back to our case, the null hypothesis of non-stationarity can be rejected at the 10% level, suggesting that the time series is stationary. Nonetheless, let us try to differentiate the model in order to detect whether we can reject non-stationarity at a lower p-value.

```
    Augmented Dickey-Fuller Test

data:  delta_bpm
Dickey-Fuller = -7.7538, Lag order = 5, p-value = 0.01
alternative hypothesis: stationary
```

As expected, the p-value for the stationarity test in the differenced model is lower. Still, differencing the data to make them even more stationary would not be advisable in our case. Quoting Chatfield, "my personal preference is to avoid transformations whenever possible except where the transformed variable has a direct physical interpretation". In this sense, the change in BPM over a 30 seconds period would not be a meaningful variable. Therefore, we decide to proceed the analysis using the original time-series for BPM.

(ii) Model specification



The estimated *acf* is monotonically decreasing toward zero, suggesting the presence of an autoregressive component with positive coefficients. Since the estimated partial acf shows a significantly negative value for

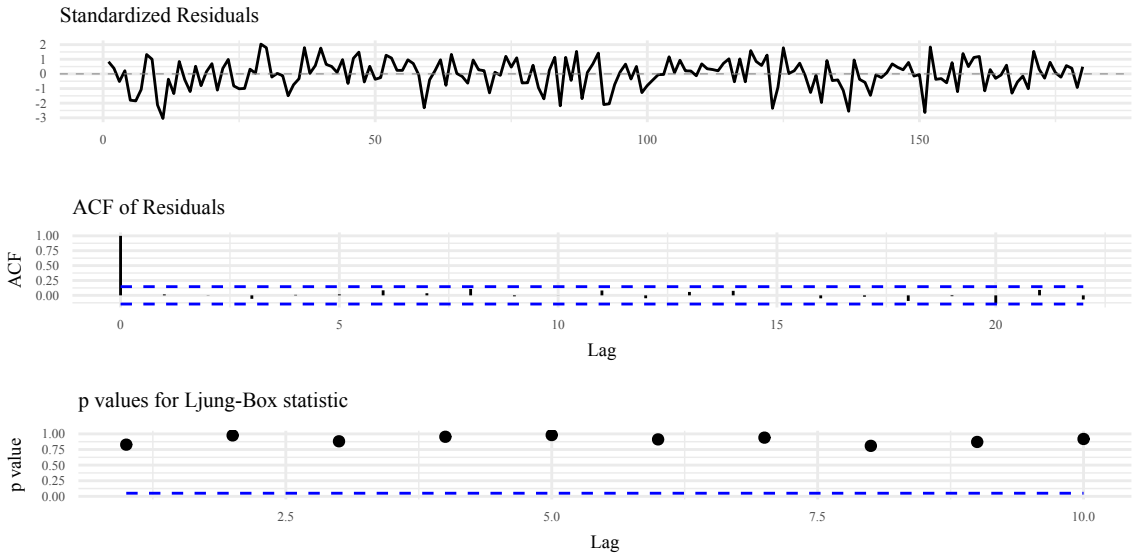
the second lag, we exclude the possibility of an AR(1) model. This feature of the *pacf* can be explained by the presence either of a second autoregressive component or of a moving average component. For these reasons, we will compare the performances of three alternative models: an AR(2), an ARMA(1,1) and an ARMA(1,2). First, we estimate the parameters for each case.

(iii) Estimation of the parameters

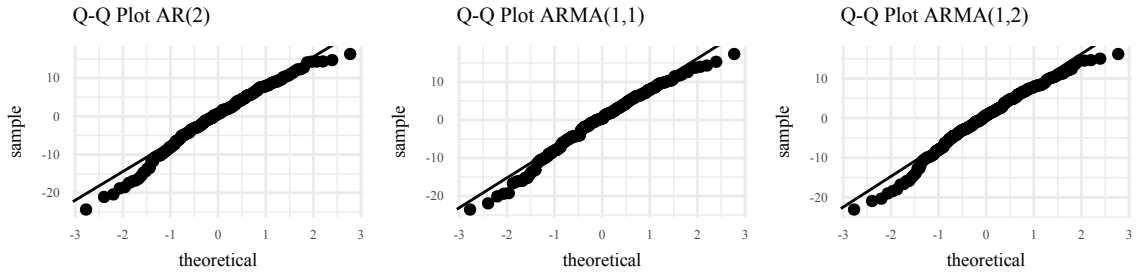
	ar1	ar2	ma1	ma2	drift
AR(2)	1.0784	-0.3201			136.8531
ARMA(1,1)	0.7199		0.33		136.9281
ARMA(1,2)	0.6174		0.4705	0.1862	136.8822

(iv) Model checking

We now perform diagnostic checking for the three models. After acknowledging that the results are broadly analogous in the three cases, for brevity reasons we only report the diagnostic output of the AR(2) model.



All models seem to fit very well the data: the residuals look random and show no autocorrelation. The p-values for the Ljung-Box test are all very large.



Shapiro-Wilk normality test

	AR(2)	ARMA(1,1)	ARMA(1,2)
W:	0.979657246	0.981708648	0.979428335
p-value:	0.009868475	0.018396811	0.009213612

Both the Q-Q Plot and the Shapiro test provide evidence to reject normality of the residuals. All considered, although clearly non-correlated, the residuals cannot be thought of as a gaussian white noise. This has major implications as regards estimation of the model's parameters, which will ultimately affect also model identification. Except for numerical optimization methods, the estimation of all models including a moving

average component - in our case, ARMA(1,1) and ARMA(1,2) - is done through MLE or conditional-sum-of-squares (Box and Jenkins, 1976). Both methods rely on the assumption of normality of the errors. This assumption implies that also the forecast errors distribute normally, which, in our case, is clearly at odds with the results of the Shapiro test. Conversely, an AR(2) model can be estimated via classical OLS, which is based on the less restrictive assumption of zero-mean and constant-variance uncorrelated errors. In conclusion, in order to prevent any estimation failure due to incorrect identification assumptions, it seems safer to us to stick to an AR(2) model. The AR(2) estimates obtained via OLS are displayed below.

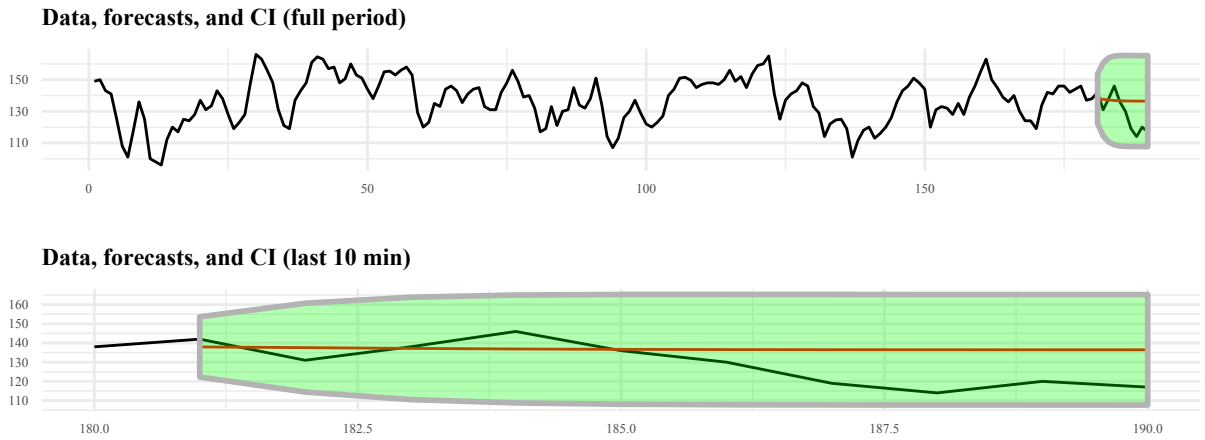
```
AR(2), OLS estimates

                  ar1     ar2   drift
Coefficients:   1.0803 -0.3221 -0.0641
```

Finally, in order to make a definitive choice between the models, we weigh the contrasting principles of explanatory power and parsimony. We therefore compute the AIC and BIC for the three models.

```
               AIC      BIC
AR(2)     1268.215 1280.987
ARMA(1,1) 1270.380 1283.152
ARMA(1,2) 1269.090 1285.054
```

According to both information criteria, the best model still appears to be an AR(2).

Point 3



The AR(2) model with drift can be expressed as follows:$y_t = c + \alpha_1 y_{t-1} + \alpha_2 y_{t-2}$ Accordingly, the resulting forecasts are:

$$\hat{y}_{t+1|t} = \hat{c} + \hat{\alpha}_1 y_t + \hat{\alpha}_2 y_{t-1}$$

$$\hat{y}_{t+2|t} = \hat{c} + \hat{\alpha}_1 \hat{y}_{t+1} + \hat{\alpha}_2 y_t$$

$$\hat{y}_{t+3|t} = \hat{c} + \hat{\alpha}_1 \hat{y}_{t+2} + \hat{\alpha}_2 \hat{y}_{t+1}$$

...

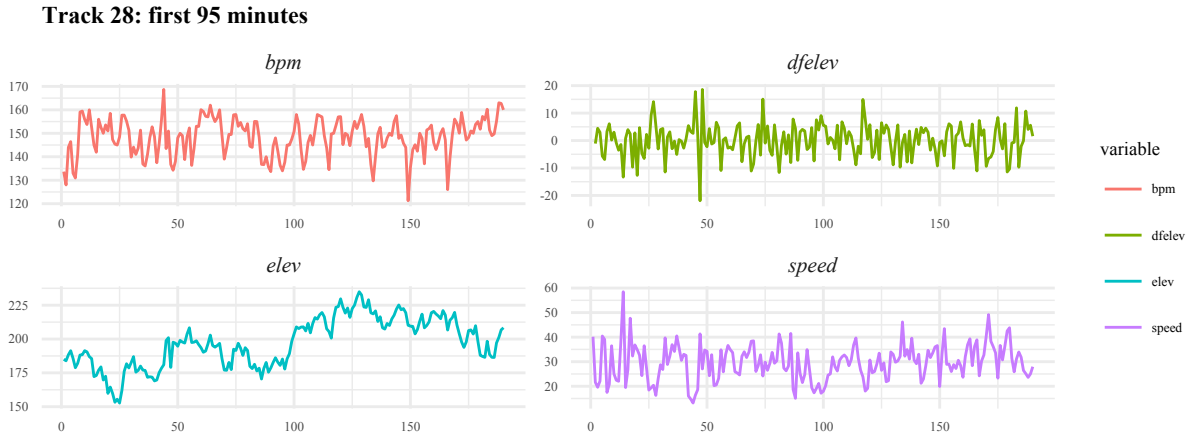
$$\hat{y}_{t+h|t} = \hat{c} + \hat{\alpha}_1 \hat{y}_{t+h-1} + \hat{\alpha}_2 \hat{y}_{t+h-2}$$

As regards the shape of the confidence intervals, we notice that they tend to widen as h increases. This feature is consistent with the properties of dynamic forecasting.

After all, the forecast performance of our model is good. Indeed, all the observed values of bpm over the last 10 periods fall within the 95% confidence interval built around our forecast.

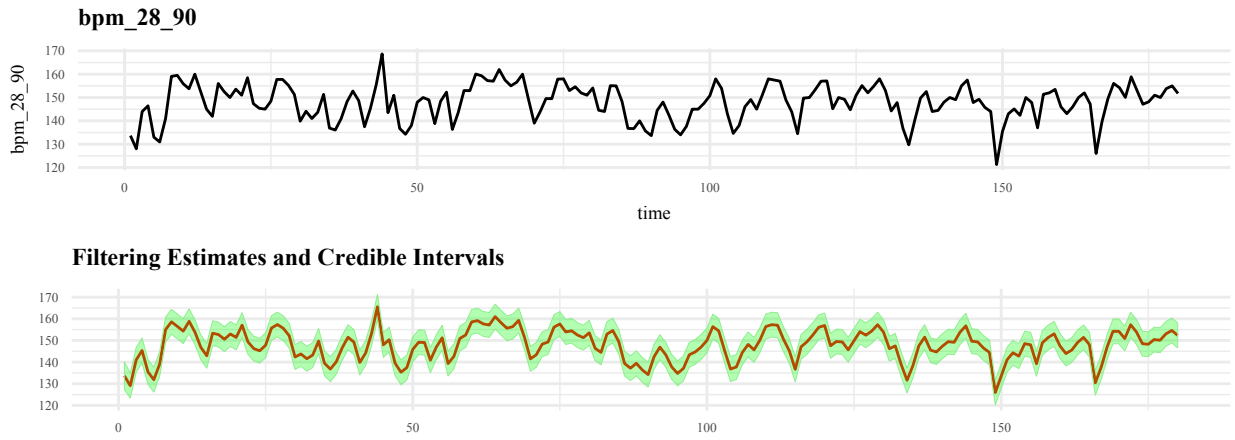
Exercise 2

Point 1



The plot of track_28 shows both similarities and discrepancies with respect to that of track_38. As before, we notice a strong negative correlation between speed and change in elevation. However, dealing with the variable elevation, it is now harder to detect a clear trend component. Again, BPM can be safely assumed to be stationary.

Point 2



The Dynamic Linear Model we are defining (i.e. Random Walk plus Noise) is composed of a bivariate process with only one state equation and one observation equation.

$$y_t = \theta_t + v_t \qquad v_t \sim^{i.i.d} N(0, \sigma_v^2) \qquad\qquad \theta_t = \theta_{t-1} + w_t \qquad w_t \sim^{i.i.d} N(o, \sigma_w^2)$$

As observed in class, the filtering estimates can be computed recursively according to the following formula:

$$m_t = m_{t-1} + \frac{C_{t-1}+\sigma_w^2}{C_{t-1}+\sigma_w^2+\sigma_v^2}(y_t - m_{t-1}) \qquad \text{and} \qquad C_t = (C_{t-1} + \sigma_w^2) - \frac{(C_{t-1}+\sigma_w^2)^2}{C_{t-1}+\sigma_w^2+\sigma_v^2}$$

A particular characteristic of the model comes from the analysis of the signal-to-noise ratio, defined as: $r = \frac{\sigma_w^2}{\sigma_v^2}$. This has an huge impact on the Kalman Filter: $K_t = \frac{C_{n-1}+\sigma_w^2}{C_{t-1}+\sigma_w^2+\sigma_v^2}$. In other words, the signal-to-noise ratio r quantifies, through k_t, the gain in terms of information carried by the current observation y_t with respect to the previous filtering state estimate m_{t-1}. In this case, the estimated signal-to-noise ratio is 3.16. Although y_t is carrying a smaller variance than θ_t, our filter m_t is relying both on the new information y_t and on the

filter of the previous period m_{t-1}, without disproportionate weight being given to one with respect to the other.

Point 3

Generally, we can describe a Dynamic Linear Model as a polynomial model if some conditions are satisfied. Accordingly, we will prove that these conditions hold for the Random Walk plus Noise, defining it as a Polynomial DLM of order one. Hence, according to the definition, given the forecast function: $f_t(h) = E(Y_{t+h}|Y_{1:t}) = \alpha_{t,0} + \alpha_{t,1}h + \alpha_{t,2}h^2 + ... + \alpha_{t,n-1}h^{n-1}$ (where the $\alpha_{t,j}$s are linear functions of $m_t = E(\theta_t|Y_{1:t})$), those models that present such forecast function are called Polynomial DLMs of order n.

In our case, let us try to define our forecast function and, thus, let us define recursively the h-step ahead forecasts. We can start from the $t-th$ period assuming the classic filtering distribution, defined as follows

$$\theta_t|Y_{1:t} \sim N(m_t, C_t)$$

thus, we can proceed by forecasting both for state and observation processes

$$\theta_{t+1}|Y_{1:t} \sim N(a_{t+1}, R_{t+1}) \quad \text{where} \quad a_{t+1} = m_t \quad \text{and} \quad R_{t+1} = C_t + \sigma_w^2$$

$$Y_{t+1}|Y_{1:t} \sim N(f_{t+1}, Q_{t+1}) \quad \text{where} \quad f_{t+1} = a_{t+1} = m_t \quad \text{and} \quad Q_{t+1} = R_{t+1} + \sigma_v^2 = C_t + \sigma_w^2 + \sigma_v^2$$

Proceeding until h-step ahead, we can obtain

$$\theta_{t+h}|Y_{1:t} \sim N(a_{t+h}, R_{t+h}) \quad \text{where} \quad a_{t+h} = m_t \quad \text{and} \quad R_{t+h} = C_t + h\sigma_w^2$$

$$Y_{t+h}|Y_{1:t} \sim N(f_{t+h}, Q_{t+h}) \quad \text{where} \quad f_{t+h} = a_{t+h} \quad \text{and} \quad Q_{t+h} = R_{t+h} + \sigma_v^2 = C_t + h\sigma_w^2 + \sigma_v^2$$

Finally, as mentioned above, the process can be defined as a polynomial DLM of order 1, since

$$f_t(h) = \alpha_{t,0} = E(\theta_t|Y_{1:t}) = m_t$$

At this point, it seems clear that the h-step ahead state and observation forecasts are evolving constantly based on the last available filtered value m_t. Since we are not able to update our information period by period, uncertainty connected to each successive prediction increases. Thus, credible intervals, for both state and observation forecasts, get larger as h increases.

A different scenario arises in the case of the 1-step ahead model, obtained by fitting a DLM model on the overall sample of 95 minutes. Now, the set of observation and state forecasts is more flexible, since we are updating them every time a new observation becomes available. Besides, their confidence interval remain stable, in terms of width, over time. Before proceeding with a comparison between 1-step ahead and h-step ahead predictions, we can recover the more general forecasting procedure for the one-step ahead model, starting, as before, from the last filtering distribution at time t. Thus,

$$\theta_t|Y_{1:t} \sim N(m_t, C_t)$$

the forecasting process can be expressed in the following recursive way:

$$\theta_{t+1}|Y_{1:t} \sim N(a_{t+1}, R_{t+1}) \qquad a_{t+1} = G_{t+1}m_t = m_t \qquad R_{t+1} = G_{t+1}C_tG_{t+1}^{\intercal} + W_{t+1} = C_t + \sigma_w^2$$
$$Y_{t+1}|Y_{1:t} \sim N(f_{t+1}, Q_{t+1}) \qquad f_{t+1} = F_{t+1}a_t = a_t \qquad Q_{t+1} = F_{t+1}R_{t+1}F_{t+1}^{\intercal} + V_{t+1} = R_{t+1} + \sigma_v^2$$

until

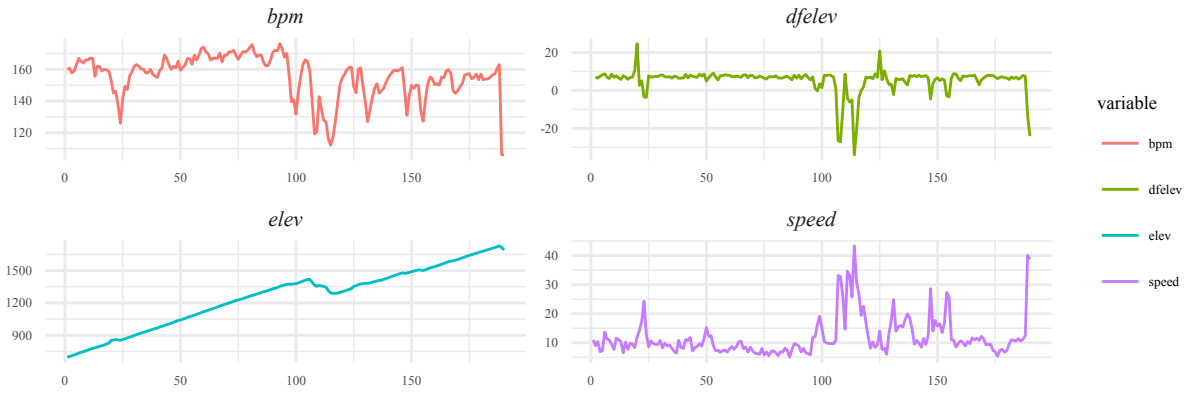
$$\theta_{t+h}|Y_{1:t+h-1} \sim N(a_{t+h}, R_{t+h}) \qquad a_{t+h} = m_{t+h-1} \qquad R_{t+h} = C_{t+h-1} + \sigma_w^2$$
$$Y_{t+h}|Y_{1:t+h-1} \sim N(f_{t+h}, Q_{t+h}) \qquad f_{t+h} = F_{t+1}a_t = a_{t+h} = m_{t+h-1} \qquad Q_{t+h} = R_{t+h} + \sigma_v^2$$

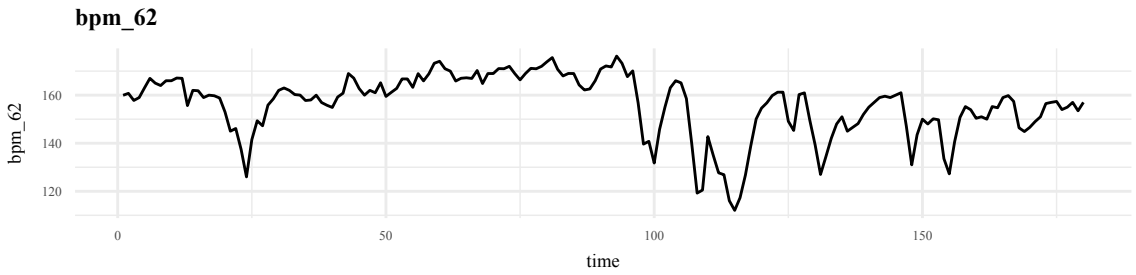
As we can observe, the updating process proposes an evolution in both point forecasts and credible intervals. The upcoming information at every new period $t + j$, gives the possibility of computing a new Kalman Filter for that particular period. Updating the filtering period, we are enlarging the set of information that Kalman Filter is recursively considering, which allows to ameliorate our prediction. Based on the formulae and motivation showed, we can observe how the one-step ahead model proposes state point-forecasts and observation point-forecasts that are identical to each other, period by period (i.e. from 181 to 190 observation). That is to say, point forecasts for both state and observation processes are equal to the previous-time filtering estimate. At this point, we can proceed with the comparison between the forecasts of the one and h-step ahead models. First of all, the graph comparing observation forecasts between the two models is identical to the one comparing state forecasts. Accordingly, for the first model, forecasts - both state and observation - start by defining $f_{t+1} = a_{t+1} = m_t$ (with t=180) and proceed until $f_{t+10} = a_{t+10} = m_{t+9}$ (with t+10=190). By contrast, for the second model, we are starting from $f_{t+1} = a_{t+1} = m_t$ (i.e. t=180) and forecasts (both state and observation) remain constant at the same level for the next 10-step ahead periods.

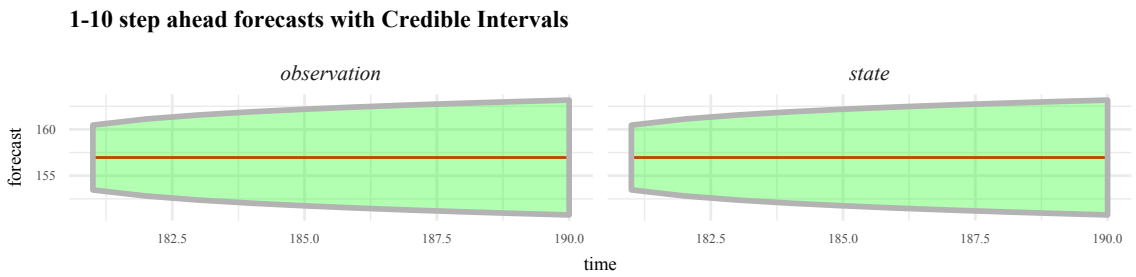
Point 4

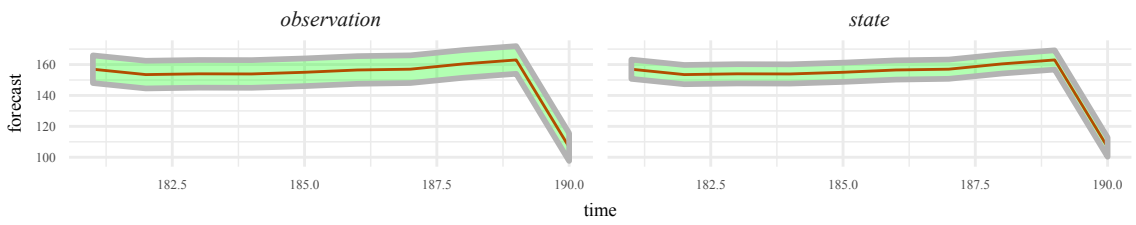
We repeat points 1 and 3 for track #62.

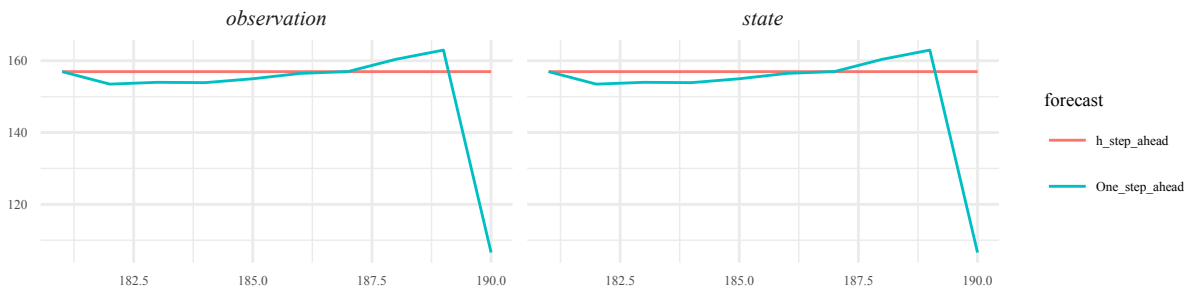
Track 62: first 95 minutes











Finally, based on the results of points 2 and 3, we can draw some general remarks on h-step ahead vs recursive 1-step ahead forecasts in the case univariate models. As already shown in detail, the h-step ahead forecasts (both for states and observations) evolve constantly based on the last filtered value:

$$f_{t+h} = a_{t+h} = m_t \text{ with } h = 1, ..., 10$$

Therefore, differently from recursive 1-step ahead forecasting, any changes in parameters (e.g. θ_t and potentially also σ_v^2 and σ_w^2) that might occur within the time interval $t+1, ..., t+10$ will not be taken into account by the correseponding h-step ahead forecasts, i.e. $f_{t+2}, ..., f_{t+10}$. This feature, together with the fact that credible intervals for h-step ahead forecasts increase with h, lead us to the conclusion that, whenever available, recursive 1-step ahead forecasts should be preferred to their h-step ahead counterparts.

Having said that, if we are in a case other than streaming data, and recursive forecasting is not feasible, h-step ahead forecasting remains the one and only method to be applied. In this last scenario (which, yet, is

not the case in this project), it is essential to check for the stability of the parameters.

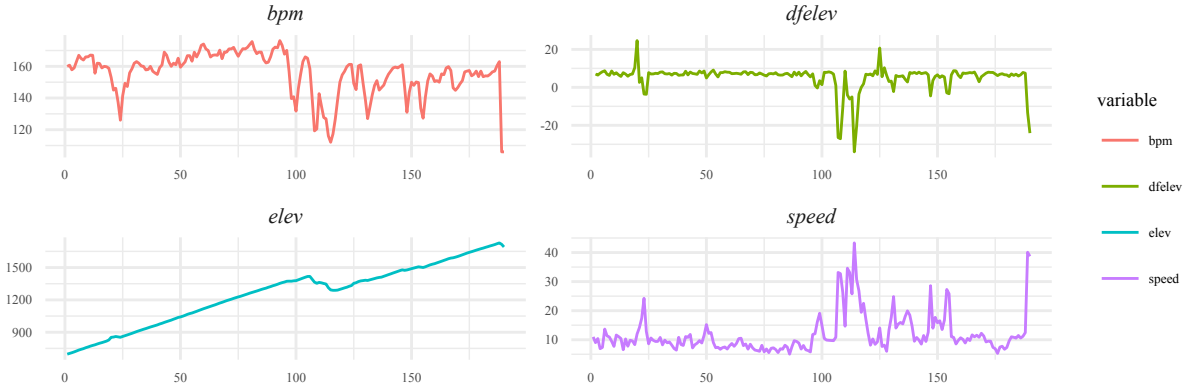
In conclusion, it is worth mentioning that running an ADF test on the stationarity of the last 5 minutes of time-series BPM for track 28, we do not reject the null hypothesis of non-stationarity at the 5% significance level. In a sense, this result confirms the risk of forecast failure due to parameter instability in the case of h-step ahead forecasting.

```
    Augmented Dickey-Fuller Test

data:  bpm_28_last_5min
Dickey-Fuller = -3.4541, Lag order = 2, p-value = 0.07027
alternative hypothesis: stationary
```

Exercise 3

Point 1

Track 62



OLS estimates properties of unbiasedness and consistency require three assumptions to hold:

- LRM (1) linearity of the model
- LRM (2) full column rank of the explanatory variable matrix
- LRM (3) the expected value of errors conditioned of the explanatory variable is 0.

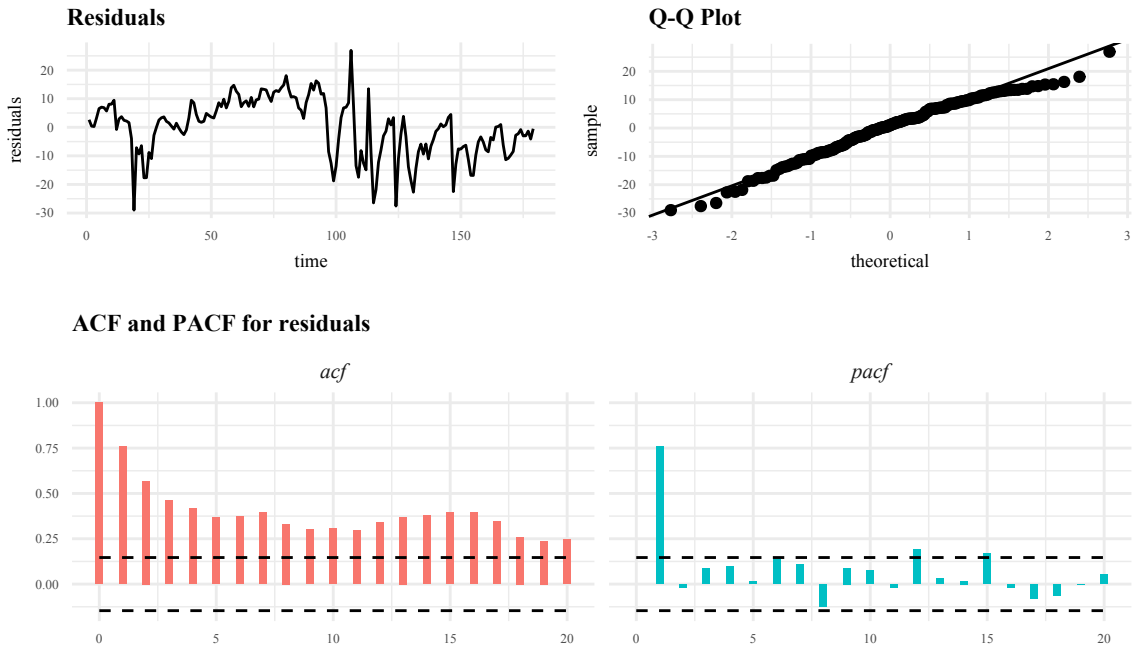
We may safely assume that the first two hypotheses are respected, given the theoretical formulation of the model. We have only one regressor that is varying over time and thus not perfectly collinear with the constant term. As for the third hypothesis, we cannot totally exclude the risk of endogeneity due to omitted varable bias. Nonetheless, we proceed with the analysis assuming that LRM(3) holds.

Furthermore, it is worth checking if also other two properties of the linear regression model hold, namely:

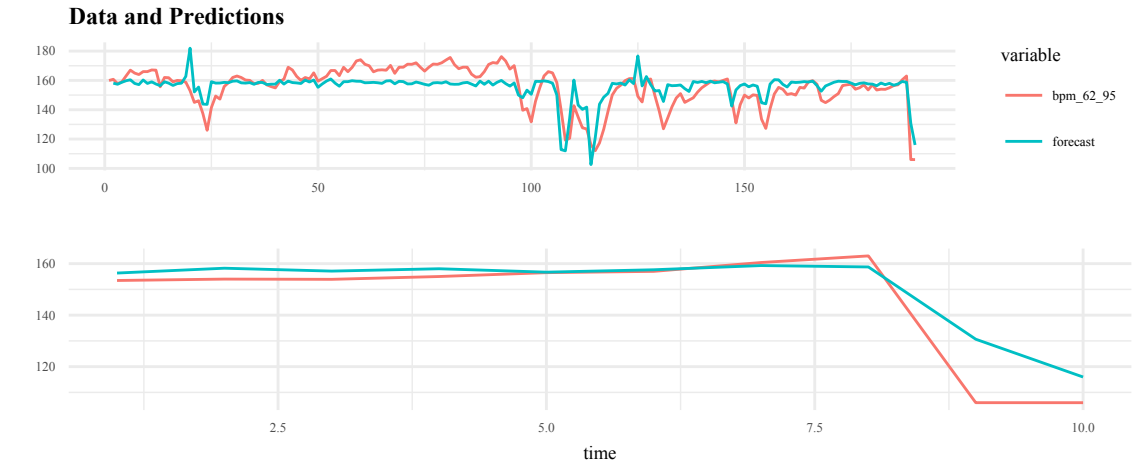
- LRM (4) Homoskedasticity of errors given explanatory variables
- LRM (5) Normality of errors given explanatory variables.

These two properties are essential in order to define the finite sample distribution properties of the estimators.

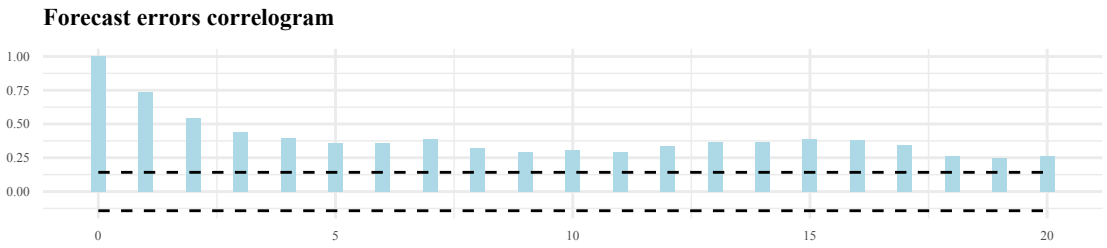
```
    Shapiro-Wilk normality test

data:  res
W = 0.98086, p-value = 0.01464
```



As we can observe, residuals seem to be correlated over the sample, which does not allow us to easily estimate coefficient variances through OLS approach. Instead, a more sophisticated analysis, via F-GLS or HAC, is required. In addition, also LRM (5) is violated, meaning that the finite sample distribution of the OLS coefficients is apparently not normal. In order to implement significance tests (e.g. T-test, F-test) we must rely on asymptotic distributions. At a later point, we can properly solve previous concerns through HAC or F-GLS estimation techniques. However, for the purposes of point forecasting we can continue to rely just on the LRM-1)-LRM-3) properties.



In order to evaluate the predictive performance of the model we check whether a white noise process can be fitted to the residuals.



Looking at the correlogram of forecast errors, we notice that forecast errors are correlated over time, suggesting that the model is not performing well.

β_1 can be defined as the variation of bpm associated to a one-meter variation in altitude. A static linear regression model assumes the stability of these parameters all over the sample. However, we are led to approach the idea that the mentioned assumption might be too restrictive.

The model we have fitted measures a linear relationship between change in elevation and bpm. Particularly, let us consider the first of the two. Obviously, some major determinants of the fatigue of a cyclist are the slope of the route and the length of the slope itself. For instance, we can assume that the longer a cyclist is subject to a positive slope, the greater the fatigue she accumulates. This first example is suggesting the presence of a non-linear relationship between the variables. In other words, suppose that from $t-1$ to t the individual climbs 1 meter in elevation. In this case we are assuming that the impact on bpm is β_1. Let us also suppose that the same individual, travels 3 meters in elevation from period t to $t+3$. The individual has climbed 3 times the elevation she had climbed from $t-1$ to t, but now she has done it in 4 successive periods. This is likely to add extra burden on the individual's fatigue. However, applying the model in question, we would expect heart rate at period $t+3$ (i.e. y_{t+3}) to be exactly the same as the heart rate at period t (i.e.y_t). Thus

$$y_t = \beta_0 + \beta_1 x_t \text{ supposing } x_t = 1 \Longrightarrow y_t = \beta_0 + \beta_1$$

Now let us predict y_t from t to $t+3$ recursively, supposing $x_j = 1 \ \forall j \in [t+1, t+3]$

$$y_{t+1} = \beta_0 + \beta_1 x_{t+1} = \beta_0 + \beta_1$$
$$y_{t+2} = \beta_0 + \beta_1 x_{t+2} = \beta_0 + \beta_1$$
$$y_{t+3} = \beta_0 + \beta_1 x_{t+3} = \beta_0 + \beta_1$$

This is clearly unrealistic since heart rate, even if eventually converging to a maximum value, should reach higher frequencies after prolonged climb than after a simple 30-second slope. Nevertheless, a static regression model is not able to capture this difference.

Moreover, with regard to the stability of the parameters, while assuming the linearity of the model, we can advance another simple counterexample to refute the initial hypothesis. Suppose you can face the same climb, with the same speed from time 0 to time 1, or from time $t+k-1$ to time $t+k$ with $k > 1$ and $t > 0$. It seems obvious that the effort required to the subject to face the climb at the beginning of the ride is less than the effort required for the next interval. This is simply due to the fact that from period 1 to period $t+k-1$ the subject has has already been riding, therefore has already accumulated fatigue. In this way the effect on bpm for the climb from 0 to 1 will be lower than the effect of facing the same climb at the same speed from $t+k-1$ to $t+k$.

Lastly, it is worth noticing that even the very relation between fatigue and bpm is unlikely to be linear. Conversely, it would be more plausible to imagine bpm as a function of fatigue, increasing but at a decreasing rate.

As for the model's pros, one advantage of implementing a simple linear regression model is the simplicity of the model. Compared to a DLM, in fact, the level of complexity, intended primarily as the number of parameters to be estimated, is significantly lower. In this terms, linear models are approached to be very

consistent and easily manageable (Angrist and Pishke, 2008). Contrastingly, we need to estimate the exact Conditional Expectation Function and not approximate it wrongly (Sims, 2010).

Point 2

$$\widehat{W} = \begin{bmatrix} 23.18661 & 0 \\ 0 & 0.04470154 \end{bmatrix} \quad \text{and} \quad \hat{\sigma}^2 = 1.92971e-08$$

As we can observe, the two processes of Smoothing and Forecasting for observations, are diametrically opposed. The first one is reling on Backward Smothing and the second one on Foreward Forecast. As we have described in previous assignments, we can observe how forecasts for a Random Walk plus noise model and their variances relies on the following formula:

$$E(Y_t|Y_{1:t-1}) = f_t = m_{t-1}$$
$$Var(Y_t|Y_{1:t-1}) = Q_t = C_{t-1} + \sigma_w^2 + \sigma_v^2$$

At this point, we can focus on the smoothing process, that starting at T, the last period of our dataset, with

$$E(\theta_T|Y_{1:T}) = s_T = m_T$$
$$E(\theta_s|Y_{1:T}) = s_t = m_t + C_t G'_{t+1} R_{t+1}^{-1}(s_{t+1} - a_{t+1})$$

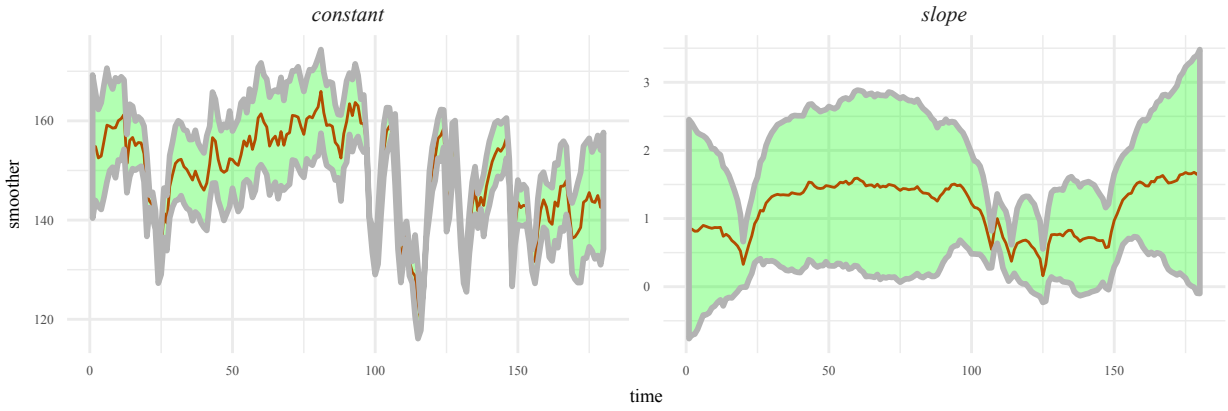
where its variances are

$$Var(\theta_T|Y_{1:T}) = S_T = C_T$$
$$Var(\theta_s|Y_{1:T}) = S_t = C_t - C_t G'_{t+1} R_{t+1}^{-1}(R_{t+1} - S_{t+1}) R_{t+1}^{-1} G'_{t+1} C_t$$

where, in this model $G_t = 1$

Finally, it is interesting to observe that smoothing estimates are actually smoother than filtering estimates. This is generally true, since the smoothing process is relying on a wider set of information.

Smoothing Estimates and Credible Intervals



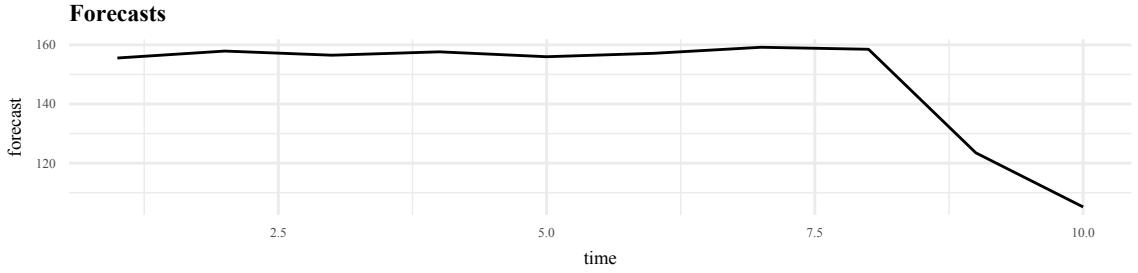
The process we are fitting presents some important chatacteristics. Firstly, when we fit a dlm regression model, the variance for the observation process is almost 0. This is suggesting that everything of the obervation process is perfectly explained by the bivariate state process we are taking into consideration. Thus, every observation y_t and so the BPM level at time t is perfectly predicted by the process $\beta_{0t} + \beta_{1t}x_t$. The first of the two state processes is defined for the constant β_{0t}. Observing its smoothing estimates, and so the overall state trajectory from $t = 1$ to $t = 180$, we can denote how the credible interval is alternatively widening and restricting with respect to the smoothing point estimate. Specifically, we can notice how credible intervals shrink in correspondence of very low values of BPM. In this case the uncertainty relative to the smooting estimate is really low, both in statistical and in relevance terms. Interestingly, we can compare the β_{0t} trajectory with the BPM trajectory from $t = 1$ to $t = 180$. We can notice how the two trends are very similar, especially in case of low peaks. This is suggested by the fact that every time the individual is on

a downhill, the effect that the change in elevation has on bpm becomes less relevant, as we will see later, and more weight is given to the authomatic resting and recovering process carried by the constant term. As previously mentioned, in this dynamic dimension the constant can assume the role of an accumulator of fatigue, increasing during prolonged upward slopes. Thus, in some periods, like the 25th or from 100th to 110th, and so in correspondence of steep downhils, the constant decreases dramatically with respect to its mean value of 149. Finally, the overall trajectory of β_{0t} results to be strongly relying on the y_t path. Accordingly, the Kalman Smoother is giving very large weight to the observation at each time.

Differently, for the slope β_{1t}, we can claim different results in terms of stability and uncertainty. As matter of fact, β_{1t} seems to predict relatively badly the effect that the difference in elevation has on bpm. In more practical terms, we can focus on the credible intervals affecting smoothing point estimates of the slope. In some periods the credible interval shrinks on the point estimates, while in other periods it becomes as large as a variation of 100% (up or down) of the point estimate. Hence, those periods in which β_{1t} seems to be less uncertain, correspond to more volatile $dfelev$. In other words, β_{1t} better succeeds in capturing the effect that x_t has on y_t during the periods in which the difference in elevation changes continuosly and with consistent slope variation. In other words, we are suggesting that our "slope process" is more certain in those cases in which dfelev varies more. Contrastingly, in those periods during which dfelev seems to be constant at a fixed value (e.g. from $t = 25$ to $t = 90$) and shows incosistent and small elevation changes, the credible interval of β_{1t} smoothing process enlarges.

However, overall, our process shows improvements in terms of estimation and prediction performances. On avarage the β_{0t} process defines a smoothing estimates that is consistent with the one predicted by the linear model. This intuition relies on the fact that by its very econometrical nature, the constant of the linear model is estimated as the sample mean of the observations. Nevertheless, as previously explained, the β_{0t} smoothing values are undergoing the avarage value, suggesting resting periods. Differently, both the estimated slope in linear model and the β_{1t} process seem to have lower relevance in relative terms. However, even if, on avarage, β_{1t} smoothing estimates are lower than the slope estimated in the linear model, the former are very volatile, especially in presence of downhills, where the smoothing estimates become lower than 1.

Point 3




```
             MAPE
Model 1:  0.4518514
Model 2:  0.2851411
```

Finally, it seems clear that the dynamic linear model we are estimating is better in terms of forecasting precision. Accordingly, by computing the Mean Avarage Percentage Forecast Errors from $t = 181$ to $t = 190$ for both the linear model and the DLM one, the former is defining a greater MAPE than the latter, where MAPE is defined as

$$\sum_{h=1}^{H} \frac{|e_{t+h}|}{y_{t+h}}$$

Point 4

The first part of the code loads the usual libraries we used throughout the course and loads the raw dataset, tranforming it into tibble format. It then stores the value 43 in the which_track variable. This variable will let us select the track 62 in the next code chunck for further analysis.

Moving on, the code stores new variable called all_tracks_repli. It starts from the all_tracks_data dataframe, it then removes the "s" and the "track_id_seq" columns because they are useless for our purposes. Then it groups the dataset by the track_id: this is done so that the next operations will be performed "by group" and not by the whole dataset. After that, it creates the dfelev variable, which is a first difference of the elev variable, and modifies time variable, subtracting the minimum to each datapoint so that it starts off from zero. Then, the dataset is ungrouped and we keep only four variables: track_id, time, bpm and dfelev.

After that, the code creates a new dataframe called bpm_all. It starts from the all_tracks_repli dataframe, then it removes the dfelev column. The spread function creates a column for each track_id and for each track_id it stores in the relative column the bpm values (each one corresponding to the time of observation). tail(-1) removes the first observation, whereas head(190) keeps only the first 190 rows (track_id = 1). The time column is removed. At the end, we will have a matrix in which each column represents the time-series of bpm values of each track, without the first value of the series (this will allow the series to have the same lenght as the dfelev series).

Then, the code creates another new dataframe called dfe_all. The code basically does the same as before, with dfelev in place of bpm. At the end, we will have a matrix in which each column represents the time-series of dfelev values of each track.

After this data wrangling, the code creates two training and test sets for the models we are going to fit to the bpm and dfe datasets. In particular, we take the bpm_all and dfe_all dataframes and leave the last 10 rows aside for testing.

After the data preparation phase, the modelling phase starts.

First, the code creates a function called single_track_mod, with two parameters: FFx and pars. This function creates a DLM regression model as a function of the FFx parameter (which will be the regressors vector), and pars, a vector which stores transformed values for the V and W matrix. In particular, since the observation equation has only one dimension, V is actually a scalar, whereas W is a 2x2 matrix (in this model the state vector is made up of an intercept and a slope). We can notice that, as we usually assume, the covariance between the intercept and the slope shocks are by default 0, whereas the constant shock variance is set to zero: this means that the model will estimate a constant value that does not change across time (a constant constant). The initial estimates of the parameters are set to 140 for the constant and 0 for the intercept.

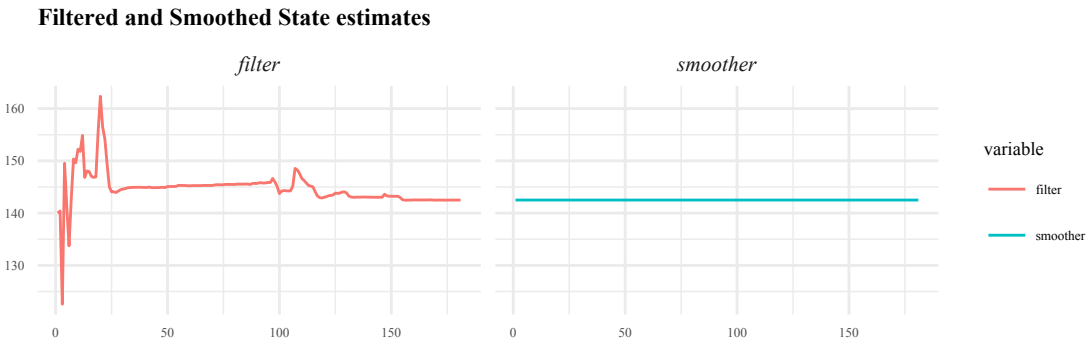
After the single_track_mod function, the code creates the replicates_LL function. The parameters of this function are: par, y_tab and x_tab. The last 2 parameters are set to default to bpm_all and dfe_all. m stores the number of columns of the y_tab parameter (which is going to be the bpm_all dataframe as default). LL_vec is initialized as an mx1 vector. After the initialization, we have a for loop that modify each component of LL_vec. In particular, the ith component of this vector is set to be the negative log-likelihood of the regression model we defined in the previous code chunk, with the i-th column of y_tab that plays the role of the dependent variable and the i-th column of the x_tab that plays the role of the regressor. The par parameter of the model is left unchanged to par. Finally, the function returns the sum of the components of LL_vec. Let us be clear on one point: what is usually done in practice, to estimate the parameters of a model, is minimizing the negative log-likelihood of one model, whereas in our case we are going to minimize the sum of the negative log likelihoods of different models, one for each track.

Once the code has defined these two function, it exploits them to compute maximum likelihood estimates of the unknown parameters. In particular, it stores in the replicates_vars variable the results of the optim function. The optim function if fed with a starting value of c(0,0) (it needs it in order to know where to start the iteration from), and then it is fed with the replicates_LL function, with par = x, y_tab = bpm_train and x_tab = dfe_train. optim will minimize replicates_LL by changing the par parameters (which is going to be made by the observation error and the evolution error of the slope, that are the parameters to be

esimated by mle). What is usually done in practice, is to estimate the parameters of a model using MLE's, is minimizing the negative log-likelihood of one model, whereas in our case we are going to minimize the sum of the negative log likelihoods of different models, each one fitted for each track. The reason is that we wish to estimate parameters that could "fit" each track, thus we have to use the information coming from each track.

Finally, the code creates a dlm object called track_model. The model is built using the single_track_mod function. The model is trained using the training set we defined before for the track we defined (track_seq = 62), using the parameters we estimated before for the variance-covariannce matrices of the random shocks. To recap, the whole code is written to estimate a single dynamic regression model, where bpm is the time-series we want to model, and dfelev is the regressor through which we want to explain bpm. Intuitively, we would say that the higher the difference in elevation the higher the bpm should be. Dynamic estimation should solve the problem of the non-linearity of the relationship between the two variables. The model could anyway suffer from omitted variables, such as lags of the dfelev variable or lags of the bpm variable itself.

After having created the dlm model, the code computes the the filtered and the smoothed dlm objects for the previously trained model. We are going to show such estmates on the next chart (only for the constant term).



The plot shows the filtered and smoothed estimates of the constant term of the dynamic regression model. In the filter chart, we can see that it first swings around and then converges to 140. As for the smoother chart, we can see that the value immediately settles a little below 143.