



# Reservoir Computing in R: a Tutorial for Using reservoirnet to Predict Complex Time-Series

ISSN 2824-7795

- Thomas Ferté Inserm Bordeaux Population Health Research Center UMR 1219, Inria BSO, team SISTM, F-33000 Bordeaux, France, Inserm  
Inserm Bordeaux Population Health Research Center UMR 1219, Inria BSO, team SISTM, F-33000 Bordeaux, France, Inria  
Bordeaux Hospital University Center, Pôle de santé publique, Service d'information médicale, F-33000 Bordeaux, France, CHU de Bordeaux
- Kalidou Ba Inserm Bordeaux Population Health Research Center UMR 1219, Inria BSO, team SISTM, F-33000 Bordeaux, France, Inserm  
Inserm Bordeaux Population Health Research Center UMR 1219, Inria BSO, team SISTM, F-33000 Bordeaux, France, Inria  
Bordeaux Hospital University Center, Pôle de santé publique, Service d'information médicale, F-33000 Bordeaux, France, CHU de Bordeaux
- Dan Dutartre Inria BSO, Inria  
Pierrick Legrand Inria BSO, F-33000 Bordeaux, France, Inria  
IMB, Institut de Mathématiques de Bordeaux, UMR CNRS 5251, IMB
- Vianney Jouhet Inserm Bordeaux Population Health Research Center UMR 1219, team AHeAD, F-33000 Bordeaux, Inserm  
Bordeaux Hospital University Center, Pôle de santé publique, Service d'information médicale, F-33000 Bordeaux, France, CHU de Bordeaux
- Romain Griffier Inserm Bordeaux Population Health Research Center UMR 1219, team AHeAD, F-33000 Bordeaux, Inserm  
Bordeaux Hospital University Center, Pôle de santé publique, Service d'information médicale, F-33000 Bordeaux, France, CHU de Bordeaux
- Rodolphe Thiébaut Inserm Bordeaux Population Health Research Center UMR 1219, Inria BSO, team SISTM, F-33000 Bordeaux, France, Inserm  
Inserm Bordeaux Population Health Research Center UMR 1219, Inria BSO, team SISTM, F-33000 Bordeaux, France, Inria  
Bordeaux Hospital University Center, Pôle de santé publique, Service d'information médicale, F-33000 Bordeaux, France, CHU de Bordeaux
- Xavier Hinaut Inria BSO, F-33000 Bordeaux, France, Inria  
Univ. Bordeaux, CNRS, IMN, UMR 5293, Bordeaux, France, CNRS  
LabRI, Univ. Bordeaux, Bordeaux INP, CNRS UMR 5800., LaBRI
- Boris Hejblum <sup>1</sup>Inserm Bordeaux Population Health Research Center UMR 1219, Inria BSO, team SISTM, F-33000 Bordeaux, France, Inserm  
Inserm Bordeaux Population Health Research Center UMR 1219, Inria BSO, team SISTM, F-33000 Bordeaux, France, Inria

Date published: 2024-12-06 Last modified: 2024-12-06

## Abstract

<sup>1</sup>Corresponding author:

Reservoir Computing (RC) is a machine learning method based on neural networks that efficiently process information generated by dynamical systems. It has been successful in solving various tasks including time series forecasting, language processing or voice processing. RC is implemented in Python and Julia but not in R. This article introduces `reservoirnet`, an R package providing access to the Python API `ReservoirPy`, allowing R users to harness the power of reservoir computing. This article provides an introduction to the fundamentals of RC and showcases its real-world applicability through three distinct sections. First, we cover the foundational concepts of RC, setting the stage for understanding its capabilities. Next, we delve into the practical usage of `reservoirnet` through two illustrative examples. These examples demonstrate how it can be applied to real-world problems, specifically, regression of COVID-19 hospitalizations and classification of Japanese vowels. Finally, we present a comprehensive analysis of a real-world application of `reservoirnet`, where it was used to forecast COVID-19 hospitalizations at Bordeaux University Hospital using public data and electronic health records.

*Keywords:* Reservoir Computing, Covid-19, Electronic Health Records, Time series

## 1 Contents

2	1 Introduction	3
3	2 RC presentation	4
4	3 Usage workflow	6
5	3.1 Installation	6
6	3.2 Package workflow overview	6
7	3.3 Basic regression use-case	6
8	3.3.1 Covid-19 data	6
9	3.3.2 First reservoir	8
10	3.3.3 Forecast	9
11	3.4 Classification	11
12	3.4.1 The Japanese vowel dataset	11
13	3.4.2 Classification (sequence-to-vector model)	11
14	3.4.3 Transduction (sequence-to-sequence model)	13

15	<b>4 Avanced case-study: Covid-19 hospitalizations forecast</b>	14
16	4.1 Introduction . . . . .	14
17	4.2 Methods . . . . .	16
18	4.2.1 Data . . . . .	16
19	4.2.2 Evaluation framework . . . . .	16
20	4.2.3 Models . . . . .	17
21	4.2.4 Hyperparameter optimisation using random search . . . . .	17
22	4.3 Results . . . . .	18
23	4.3.1 Hyperparameter selection . . . . .	18
24	4.3.2 Forecast performance . . . . .	22
25	4.3.3 Number of model to aggregate . . . . .	22
26	4.3.4 Input feature importance . . . . .	22
27	4.4 Discussion . . . . .	26
28	<b>5 Discussion and conclusion</b>	26
29	<b>References</b>	28
30	<b>Session information</b>	31

## 31 **1 Introduction**

32 Reservoir Computing (RC) is a prominent machine learning method, proposed by Jaeger (2001),  
 33 Maass, Natschläger, and Markram (2002) and Lukoševičius and Jaeger (2009) that has gained signifi-  
 34 cant attention in recent years for its ability to efficiently process information generated by dynamical  
 35 systems. This innovative approach leverages the dynamics of a high-dimensional “reservoir” (we  
 36 define it below) to perform complex computations and solve various tasks based on the response of  
 37 this dynamical system to input signals. RC has demonstrated its efficacy in tackling various chal-  
 38 lenges, encompassing pattern classification and time series forecasting in applications ranging from  
 39 electrocardiogram analysis to bird calls Trouvain and Hinaut (2021), language processing Hinaut  
 40 and Dominey (2013), power plants, internet traffic, stock prices, and beyond Tanaka et al. (2019).

41 Originally, the RC paradigm was implemented in artificial firing-rate neurons (“Echo State  
 42 Networks”, Jaeger (2001)) and spiking neurons (“Liquid State Machine”, Maass, Natschläger, and  
 43 Markram (2002)) as a recurrent neural network (RNN) where the internal recurrent connections,  
 44 denoted as the reservoir, are randomly generated and only the output layer (named “read-out”)  
 45 is trained. The reservoir projects temporal input signals onto a high-dimensional feature space,  
 46 facilitating the learning of non-linear and temporal interactions. Thus, this recurrent layer contains  
 47 high-dimensional non-linear recombination of the inputs and past states: it is a “reservoir of  
 48 computations” from which useful information can be linearly extracted (or “read-out”) to provide  
 49 the desired outputs. This offers the advantage of decreasing the computing time compared to  
 50 conventional RNNs while consistently maintaining performance (Vlachas et al. 2020). Besides,  
 51 this RC paradigm fostered increasing interest thanks to its ability to be implemented on classical  
 52 computers, as the hidden recurrent layer can be kept untrained. A wide range of physical media

53 can be also used to replace it and Tanaka et al. (2019) recently reviewed this prolific field: from  
54 FPGA hardware (Penkovsky, Larger, and Brunner 2018), to spin waves using magnetic properties  
55 (Nakane, Tanaka, and Hirose 2018), skyrmions (Prychynenko et al. 2018) or optical implementations  
56 (Rafayelyan et al. 2020). This provides interesting and potentially more efficient alternative to  
57 traditional machine learning computing.

58 RC leverages various hyperparameters to introduce prior knowledge about the relationship between  
59 input variables and output targets. But because the connections within the reservoir are randomly  
60 initialized, the same set of hyperparameters may exhibit diverse behaviors across different instances  
61 of the reservoir connections. This unpredictability makes it challenging to anticipate the perfor-  
62 mance of a particular hyperparameter setting, as identical settings may produce varying outcomes  
63 when applied to distinct instances of the reservoir. Moreover, selecting the most suitable hyperpa-  
64 rameters often requires researchers to experiment with multiple combinations on a training dataset  
65 and evaluate their performance on a separate test set<sup>2</sup>. Although this approach can be resource-  
66 intensive and time-consuming, it is a compromise that is acceptable considering the rapid simu-  
67 lation capabilities offered by RC. Furthermore, there is a current absence of implementation in R,  
68 rendering the method challenging for users unfamiliar with Python (Trouvain and Hinaut 2022) or  
69 Julia (Martinuzzi et al. 2022).

70 Here, we offer comprehensive guidance to assist new users in maximizing the benefits of RC. Ini-  
71 tially, a broad introduction to reservoir computing is presented in Section 2, followed in Section 3  
72 by a tutorial on its application using reservoirnet, an R package built upon the ReservoirPy Python  
73 module Trouvain et al. (2020). Section 3 then introduces the workflow usage on reservoirnet for  
74 RC with two basic use-cases, and finally, in Section 4 we investigate the various challenges associ-  
75 ated with an advanced case-study leveraging RC for forecasting COVID-19 hospitalizations. This  
76 case-study exploration includes detailed guidance on the modeling strategy, the selection of hyper-  
77 parameters, and the implementation process.

## 78 2 RC presentation

79 RC is a machine learning paradigm which is most often implemented as Echo State Networks (ESNs),  
80 i.e. the firing-rate neuron version (Jaeger 2001). An ESN is described by three matrices of connec-  
81 tivity: an input layer  $W_{in}$ , a recurrent layer  $W$  and an output layer  $W_{out}$ . At each time step, the input  
82 vector  $u_t$  is projected into the reservoir which is also combined with reservoir past state  $x(t - 1)$   
83 through the recurrent connections. The output  $y(t)$  is linearly read-out from the reservoir. Input  
84  $W_{in}$  and recurrent  $W$  matrices are kept random; only the output matrix  $W_{out}$  is trained in an offline  
85 or online method. Often a ridge regression (i.e.~a regularized linear regression) is used to obtain the  
86 desired outputs  $y(t)$  from the reservoir states  $x(t)$ . Figure 1 depicts the architecture. For simplicity,  
87 we will use the term “reservoir computing” for “Echo State Network” in the remainder of the paper.  
88 The input layer  $u(t)$  is an  $M$ -dimension vector, where  $M$  is the number of input time series, which  
89 corresponds to the values of the input time series at time  $t$  where  $t = 1, \dots, T$ . The reservoir layer  
90  $x(t)$  is an  $N_{res}$ -dimensional vector where  $N_{res}$  is the number of nodes in the reservoir. The value  $x(t)$   
91 is defined as follow:

$$x(t + 1) = (1 - \alpha)x(t) + \alpha \tanh(Wx(t) + W_{in}u(t + 1)) \quad (1)$$

92 The leaking rate  $\alpha \in [0, 1]$  defines the update rate of the nodes. The closer  $\alpha$  is to 1, the more the  
93 reservoir is sensitive to new inputs  $u(t)$ . Therefore, the reservoir state at time  $t + 1$  denoted  $x(t + 1)$

<sup>2</sup>In this article, we employ the term “train set” to refer to the combined dataset consisting of both the training and validation sets, which are cycled through in a cross-validation manner.

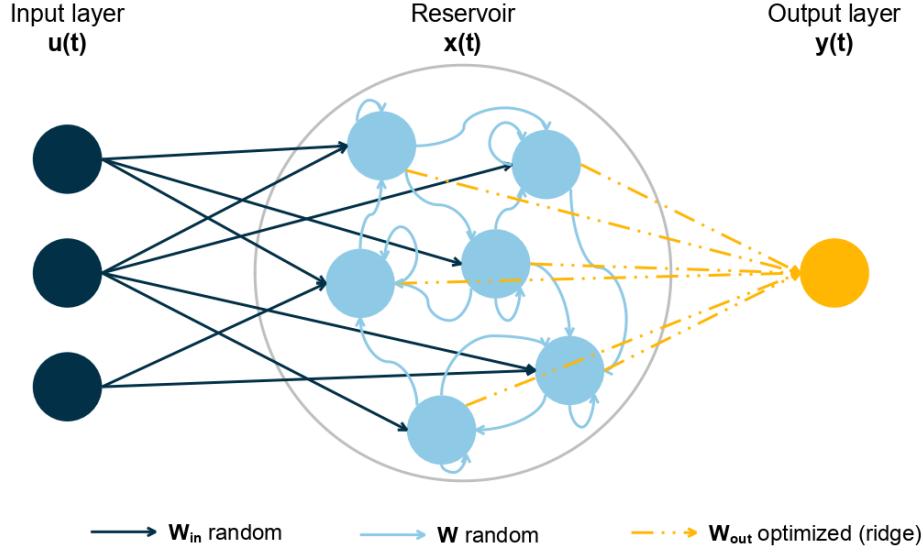


Figure 1: Reservoir computing is composed of an input layer, a reservoir and an output layer. Connection between input layer and reservoir and inside reservoir are random. Only the output layer is optimized based on a ridge regression. Adapted from Trouvain et al. (2020)

depends on the reservoir state at the previous time  $x(t)$  and the new inputs  $u(t + 1)$ . Both  $W_{in}$  and  $W$  are random matrices of size  $N_{res} \times M$  and  $N_{res} \times N_{res}$  respectively.

$W_{in}$  is a matrix (usually sparse) generated using a Bernoulli (bimodal) distribution where each value can be either  $-I_{scale}(m)$  or  $I_{scale}(m)$  with an equal probability where  $m = 1, \dots, M$  corresponds to a given feature in the input layer. The input scaling, denoted  $I_{scale}$ , is a hyperparameter coefficient common to all features from the input layer or specific to each feature  $m$ . In that case, the more important the feature is, the greater should be its input scaling.  $W$  is a matrix (usually sparse) where values are generated from a Gaussian distribution  $\mathcal{N}(0, 1)$ . Then, the  $W$  matrix is scaled according to the defined spectral radius, a hyperparameter defining the highest eigen value of  $W$ .

The final layer is a linear regression with ridge penalization where the explanatory features are the reservoir state and the variable to be explained is the outcome to predict such that:

$$W_{out} = YX^T(XX^T + \lambda I)^{-1}$$

Where  $x(t)$  and  $y(t)$  are accumulated in  $X$  and  $Y$  respectively such that:

$$X = \begin{bmatrix} x(1) \\ x(2) \\ \vdots \\ x(T) \end{bmatrix} \text{ and } Y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(T) \end{bmatrix}$$

The parameter  $\lambda$  is the ridge penalization which aims to prevent overfitting. Additionally, one can also connect the input layer to the output layer to the reservoir nodes. In that case,  $X$  is the accumulation of both such that :

$$X = \begin{bmatrix} x(1), u(1) \\ x(2), u(2) \\ \vdots \\ x(T), u(T) \end{bmatrix} \text{ and } Y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(T) \end{bmatrix}$$

109 Overall, there are four main hyperparameters to be chosen by the user: i) the leaking rate which  
110 defines the memory of the RC, ii) the input scaling which defines the relative importance of the  
111 features, iii) the spectral radius which defines the connections of the neurons inside the reservoir  
112 which in turn defines the degree of non-linear combination of features, and iv) the ridge penalization  
113 which controls the degree of overfitting. The choice of hyperparameters often requires the user to  
114 evaluate the performance of different combinations of hyperparameters on a validation set before  
115 selecting the optimal combination to forecast on the test set.

## 116 **3 Usage workflow**

117 In this section, we will cover the basics of `reservoirnet` use including installation, classification  
118 and regression. A more in depth description is provided in Section 4 with the covid-19 forecast use  
119 case.

### 120 **3.1 Installation**

121 `reservoirnet` is an R package making the Python module `ReservoirPy` easily callable from R using  
122 `reticulate` R package Ushey, Allaire, and Tang (2024). It is available on CRAN (see <https://cran.r-project.org/package=reservoirnet>) and can be installed using:

124 Reservoir Computing (RC) is well suited to both regression and classification tasks. We will introduce  
125 a simple example for both task.

### 126 **3.2 Package workflow overview**

127 The workflow of `reservoirnet` is described in Figure 2. A reservoir model is created by the association  
128 of an input layer (a matrix), a reservoir, and an output layer. Both the reservoir and the output  
129 layer are created using the function `reservoirnet::createNode()` by specifying the node type (i.e.,  
130 either `Reservoir` or `Ridge`).

131 This function accepts several arguments to specify the hyperparameters of the reservoir and will be  
132 detailed in future sections. After the reservoir and output layer are created, they can be connected  
133 using the `%>>%` operator, a specific pipe operator dedicated to `reservoirnet`. The model can then be  
134 fitted using `reservoirR_fit()` and used to make predictions on a new dataset using `predict_seq()`.

### 135 **3.3 Basic regression use-case**

#### 136 **3.3.1 Covid-19 data**

137 In this first use-case, we will introduce the fundamental usage of the `reservoirnet` package. This  
138 demonstration will be conducted using the COVID-19 dataset that is included within the package.  
139 These data encompass hospitalization, positive RT-PCR (Reverse Transcription Polymerase Chain  
140 Reaction) results, and overall RT-PCR data sourced from Santé Publique France, which are publicly  
141 available on [data.gouv.fr](http://data.gouv.fr) (for further details, refer to `help(dfCovid)`). Our primary objective is to  
142 predict the number of hospitalized patients 14 days into the future. To accomplish this, we will  
143 initially train our model on data preceding the date of January 1, 2022, and subsequently apply it to  
144 forecast values using the subsequent dataset.

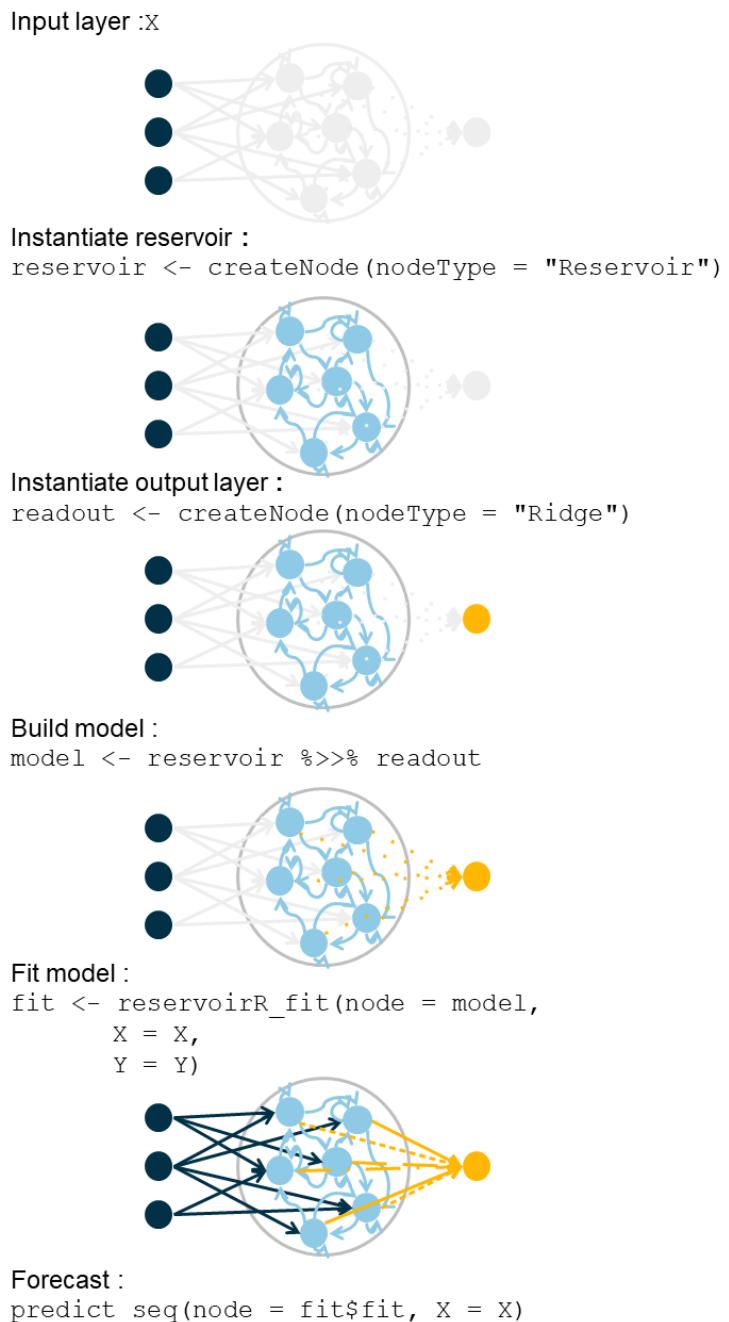


Figure 2: Workflow of `reservoirnet`.

145 We can proceed by loading useful packages - namely `ggplot2` Wickham, Navarro, and Pedersen  
146 (2018) and `dplyr` Wickham et al. (2023), data and define the task:

147 Due to the substantial fluctuations observed in both RT-PCR metrics, our initial step involves ap-  
148 plying a moving average computation over the most recent 7-day periods for these features. Addi-  
149 tionally, we augment the dataset by introducing an `outcome` column and an `outcomeDate` column,  
150 which will serve as valuable inputs for model training. Moreover, we calculate the `outcome_deriv`  
151 as the difference between the `outcome` and the number of hospitalized patients (`hosp`), represent-  
152 ing the variation in hospitalization in relation to the current count of hospitalized individuals. The  
153 resulting smoothed data is visualized in Figure 3.

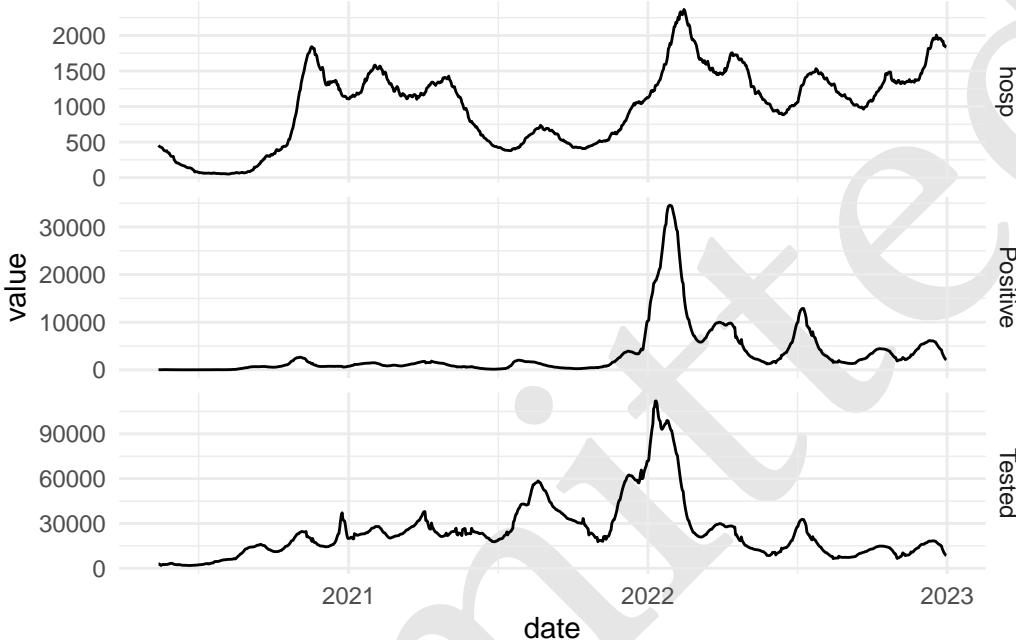


Figure 3: Hospitalizations, IPTCC and positive PCR of Bordeaux University Hospital.

### 154 3.3.2 First reservoir

155 Setting a reservoir is done with the `createNode()` function. The important hyperparameters are the  
156 following :

- 157 • Number of nodes (`units`) : it corresponds to the number of nodes inside the reservoir. Usually,  
158 the more the better, but more nodes increases the computation time.
- 159 • Leaking rate (`lr`) : the leaking rate corresponds to the balance between the new inputs and  
160 the previous state. A leaking rate of 1 only consider information from new inputs.
- 161 • Spectral radius (`sr`): the spectral radius is the maximum absolute eigenvalue of the reservoir  
162 connectivity matrix. A small spectral radius induces stable dynamics inside the reservoir, a  
163 high spectral radius induces a chaotic regime inside the reservoir.
- 164 • Input scaling (`input_scaling`): the input scaling is a gain applied to the input features of the  
165 reservoir.
- 166 • Warmup (`warmup`) : it corresponds to the number of time step during which the data are  
167 propagating into the reservoir but not used to fit the output layer. This hyperparameter is set  
168 in the `reservoirR_fit()` function.

169 In addition, we can set the seed (`seed`). Because the reservoir connections are set at random, setting  
170 the seed is a good approach to ensure reproducibility.

171 For this part of the tutorial, we will set the hyperparameter at a given value. Hyperparameter  
172 optimization will be detailed at Section 4.

173 Then we can feed the data to the reservoir and see the activation state of the reservoir  $x(t)$ . To do  
174 so, we first prepare the data and transform it to a matrix:

175 Then we run the `predict_seq()` function. It takes as input a node (i.e a reservoir or a reservoir  
176 associated with an output layer) and the feature matrix.

177 Now we can visualize node activation using the `plot()` function presented at Figure 4 .

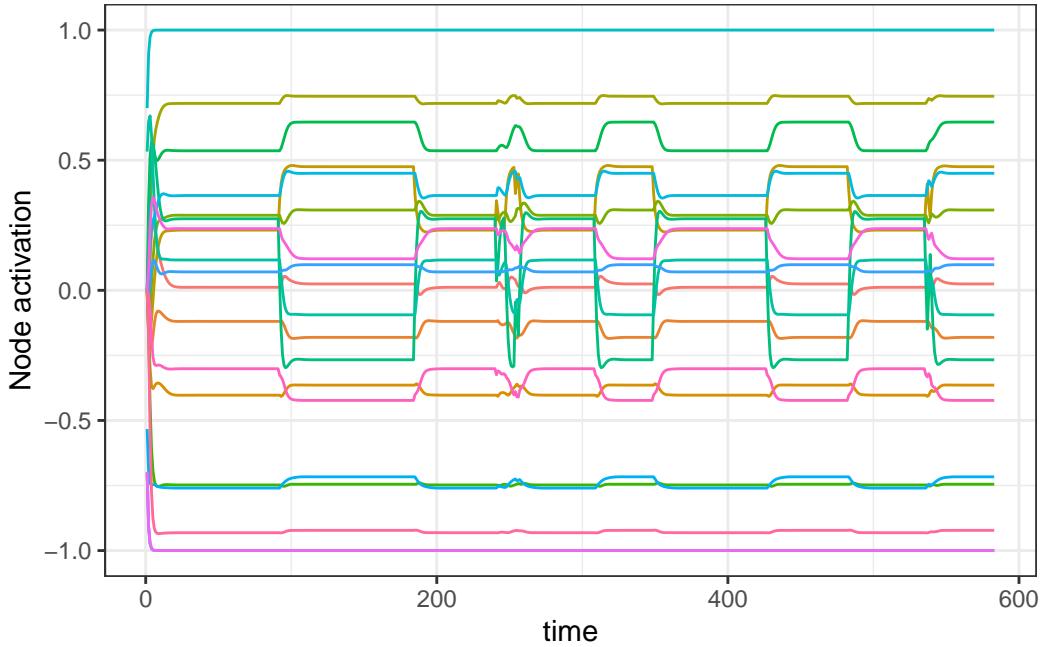


Figure 4: 20 random nodes activation over time.

178 Numerous nodes within the system exhibit a consistent equilibrium state. The challenge arises when  
179 the output layer attempts to extract knowledge from these nodes, as they do not convey meaningful  
180 information. This issue can be attributed to the disparate scales of the features. To address this  
181 concern, a practical approach involves normalizing the features by dividing each of them by their  
182 respective maximum values, thereby scaling them within the range of  $-1$  to  $1$  by dividing by the  
183 maximum of the absolute value. Of note, here the features will be scaled between  $0$  and  $1$  because  
184 all features are positive.

185 We then feed them to the reservoir and plot the node activation again. Compared to Figure 4, the  
186 obtained node activation at Figure 5 shows interesting trend outputs as no node seems saturated.

### 187 3.3.3 Forecast

188 In order to train the reservoir, we should train the last layer which linearly combines the neuron's  
189 output.

#### 190 3.3.3.1 Set the ESN

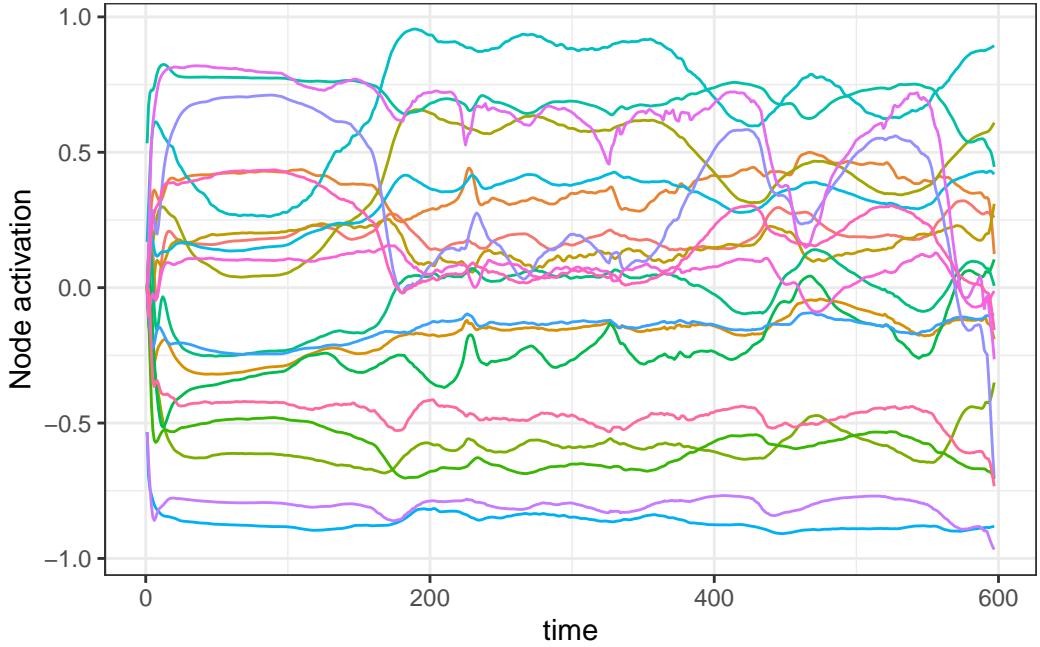


Figure 5: 20 random node activation over time. Scaled features.

191 Initially, we establish the output layer, incorporating a ridge penalty set at  $1e3$ . It's important to note  
 192 that this hyperparameter can be subject to optimization, a topic that will be explored in Section 4.  
 193 This parameter plays a pivotal role in fine-tuning the model's conformity to the data. When set  
 194 excessively high, the risk of underfitting arises, whereas setting it too low can lead to overfitting.  
 195 We connect the output layer to the reservoir making the model ready to be trained.

196 **3.3.3.2 Set the data**

197 First we separate the train set on which we will learn the ridge coefficients and the test set on which  
 198 we will make the forecast. We define the train set to be all the data before 2022-01-01 and the test  
 199 data to be all the data to have forecast both on train and test sets.

200 We standardize with the same formula as seen before. We learn the standardization on the training  
 201 set and apply it on the test set. Then we convert the dataframe to matrix.

202 **3.3.3.3 Train the model and predict**

203 We then feed the reservoir with the train set. To do so, we set a warmup of 30 days during which the  
 204 data are propagating into the reservoir but not used to fit the output layer.

205 Fitting node Ridge-0...

206 Now that the ridge layer is trained, we can forecast. We set the parameter `reset` to `TRUE` in order to  
 207 clean the reservoir from the data used by the training set.

208 We observe that the model forecast at Figure 6 is not fully accurate, both on the test set and the  
 209 train set. In that case, one option could be to reduce ridge penalization to fit more closely the  
 210 data, the optimization of ridge hyperparameter will be discussed at Section 4. Another possibility  
 211 is to ease the learning of the algorithm by forecasting the variation of the hospitalization instead of

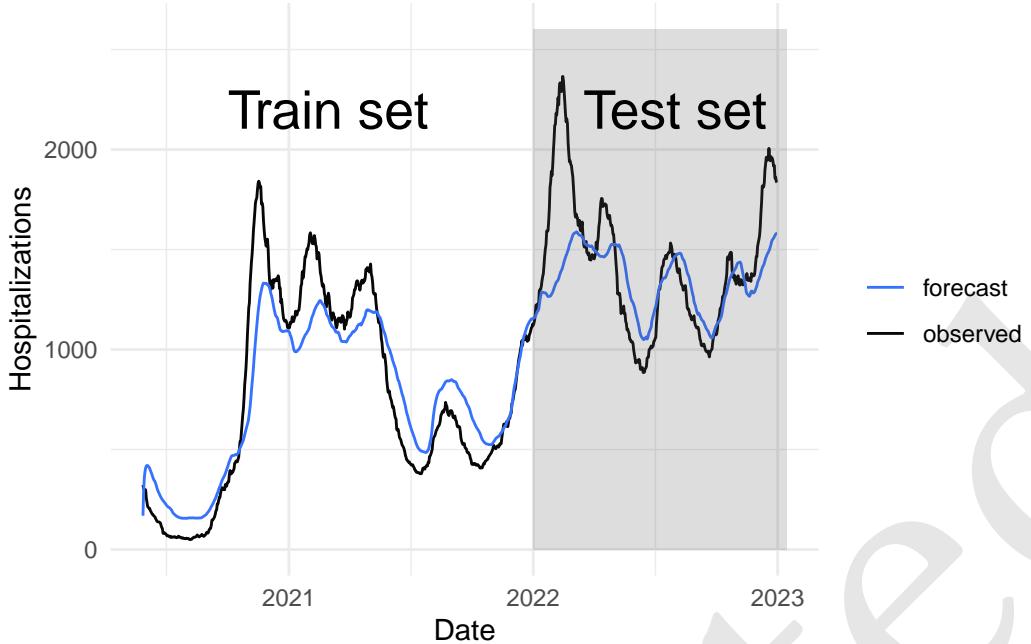


Figure 6: Forecast

212 the number of hospitalized patients. For that step, we will learn on the `outcome_deriv` contained  
 213 in `yTrain_variation` data which is defined outcome as `outcome_deriv = outcome - hosp.`. As  
 214 depicted at Figure 7, this strategy improved the model forecast.

215 Fitting node Ridge-0...

216 **3.4 Classification**

217 **3.4.1 The Japanese vowel dataset**

218 This example is largely inspired from the [classification tutorial of reservoirpy](#). To illustrate the  
 219 classification task, we will use the Japanese vowel dataset (Kudo, Toyama, and Shimbo (1999)). The  
 220 data can be loaded from `reservoirnet` as follow :

221 The dataset comprises 640 vocalizations of the Japanese vowel æ, contributed by nine distinct speak-  
 222 ers. Each vocalization represents a time series spanning between 7 and 29 time steps, encoded as  
 223 a 12-dimensional vector denoting the Linear Prediction Coefficients (LPC). A visual representation  
 224 of six distinct utterances from the test set, originating from three different speakers, is depicted in  
 225 Figure 8.

226 The primary objective involves the attribution of each utterance to its respective speaker, this is  
 227 denoted as classification or sequence-to-vector encoding. The secondary objective involves the  
 228 attribution of each time step of each utterance to its speaker, this is denoted as transduction or  
 229 sequence-to-sequence encoding.

230 **3.4.2 Classification (sequence-to-vector model)**

231 The first approach is the sequence-to-vector encoding. For this task we aim to predict the speaker  
 232 of the whole utterance (i.e the label is assigned to the whole sequence). We first start by creating

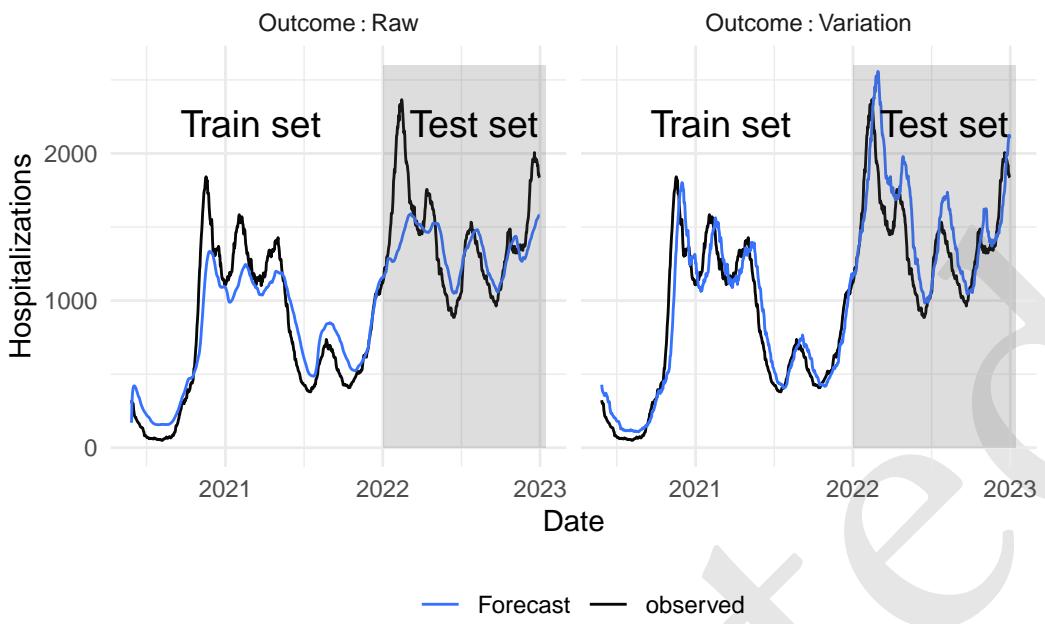


Figure 7: Covid-19 hospitalizations forecast. The model is either trained to forecast the number of hospitalizations (denoted Raw) or the variation of the hospitalizations compared to current level of hospitalisation (denoted Variation)

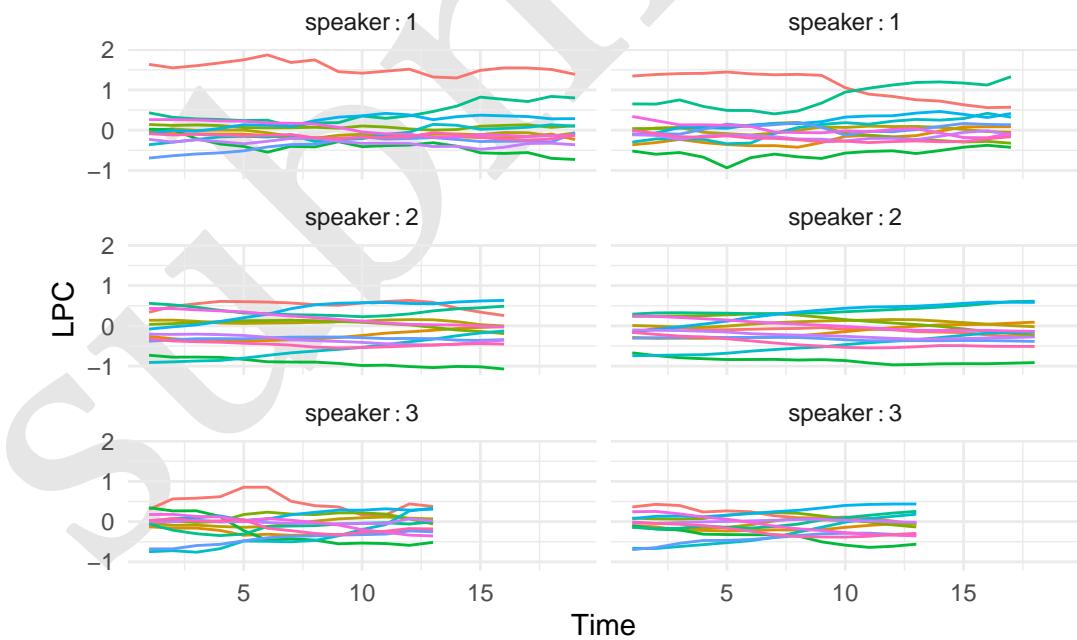


Figure 8: Vowel dataset, sample with 3 speakers and 2 utterance each.

233 the reservoir and the output layer.

234 To perform this task, we need to modify the training and testing processes. Leveraging the inherent  
235 inertia of the reservoir, information from preceding time steps is preserved, effectively endowing  
236 the RC with a form of memory. Consequently, the final state vector encapsulates insights gathered  
237 from all antecedent states. In the context of the sequence-to-vector encoding task, only this ultimate  
238 state is employed. This process is executed as follows:

239 Then we can train the readout based on this last state vector. In that case, Y\_train contains a single  
240 label for each utterance.

241 The prediction is also modified using only the final state :

242 Figure 9 shows the prediction for the 6 utterances depicted at Figure 8 where the model correctly  
243 identifies the speaker.

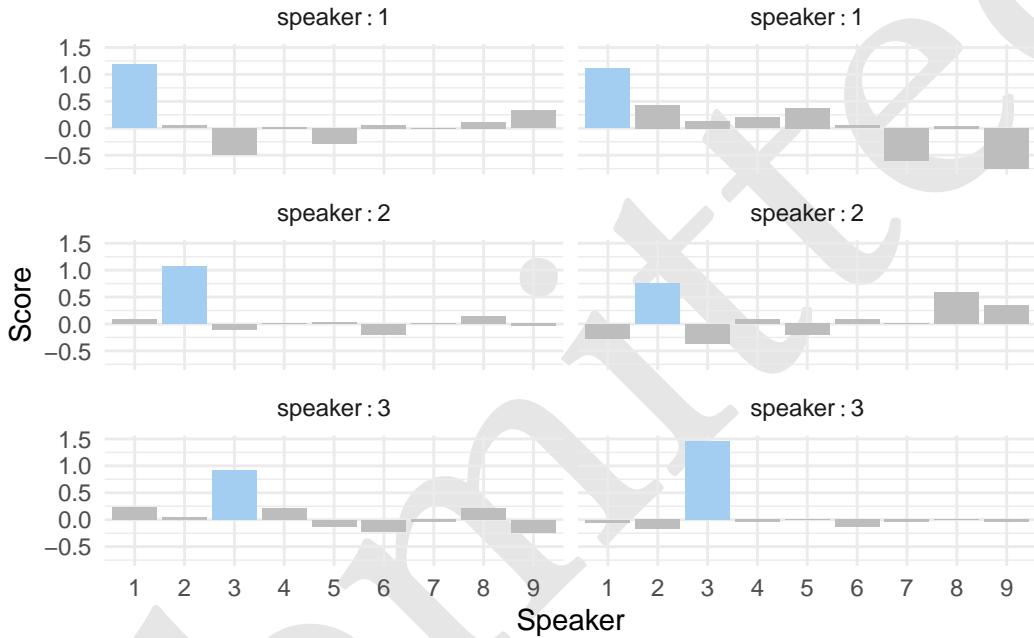


Figure 9: Prediction in a sequence-to-sequence approach 6 samples with 3 speakers and 2 utterance each. The speaker to predict is depicted in blue. For each of the 6 utterance, the model correctly identifies the speaker.

244 Then, we can also compute the overall accuracy :

245 [1] "Accuracy: 92.703%"

#### 246 3.4.3 Transduction (sequence-to-sequence model)

247 For this task, the goal is to predict the speaker for each time step of each utterance. The first  
248 step is to get the data where the label is repeated for each time step. This is easily done with the  
249 repeat\_targets argument as follow :

250 Then we can train a simple Echo State Network to solve this task. For this example we will connect  
251 both the input layer and the reservoir layer to the readout layer which is performed by the %>>%  
252 operator :

253 We can then fit the model and predict the labels for the test data. The `reset` parameter is set to TRUE  
254 to remove information from the reservoir from the training process.

255 Fitting node Ridge-2...

256 From the `Y_pred` and `Y_test` we represent at Figure 10 the predictions for the same patients as in  
257 Figure 8.

258 For those 6 utterances, the model correctly identify the speaker for most of the time steps. We can  
259 then evaluate the overall accuracy of the model :

260 [1] "Accuracy: 92.456%"

## 261 4 Avanced case-study: Covid-19 hospitalizations forecast

### 262 4.1 Introduction

263 Since late 2020, millions of cases of SARS-CoV-2 infection have been documented across the globe  
264 Carrat et al. (2021). This ongoing pandemic has exerted significant strain on healthcare systems,  
265 resulting in a surge in hospitalizations. This surge, in turn, necessitated modifications to the health-  
266 care infrastructure and gave rise to population-wide lockdown measures aimed at preventing the  
267 saturation of healthcare facilities Kim et al. (2020). The capacity to predict the trajectory of the epi-  
268 demic on a regional scale is of paramount importance for effective healthcare system management.

269 Numerous COVID-19 forecasting algorithms have been proposed using different methods (e.g en-  
270 semble, deep learning, compartmental), yet none has proven entirely satisfactory Rahimi, Chen, and  
271 Gandomi (2021). In France, short-term forecasts with different methods have been evaluated with  
272 similar results Pottier (2021). In this context a machine learning algorithm based on linear regres-  
273 sion with elastic-net penalization, leveraging both Electronic Health Records (EHRs) and public data,  
274 was implemented at Bordeaux University Hospital (Ferté et al. 2022). This model, which aimed at  
275 forecasting the number of hospitalized patients at 14 days, showed good performance but struggled  
276 to accurately anticipate dynamic shifts of the epidemic.

277 RC has been used in the context of covid-19 epidemic forecast Ghosh et al. (2021). Among them,  
278 Ghosh et al. (2021), Liu et al. (2023) and Ray, Chakraborty, and Ghosh (2021) used it to forecast  
279 epidemic, Zhang et al. (2023) performed sentiment analysis and Kmet and Kmetova (2019) used  
280 it to solve optimal control related to vaccine. The evaluation of RC for epidemic forecast showed  
281 promising results in all approaches, being competitive with Long-Short Term Memory (LSTM) and  
282 Feed-Forward Neural Network (FFNN) in Ray, Chakraborty, and Ghosh (2021). However, the test  
283 period was short for Ghosh et al. (2021) (21 and 14 days) and Ray, Chakraborty, and Ghosh (2021)  
284 (86 days) making it difficult to evaluate the behavior of the methods during epidemic dynamic shift.  
285 This was not the case for Liu et al. (2023) (6 months) but they implemented daily ahead forecast  
286 which would be difficult to use to manage a hospital. Finally, all three implementations used only  
287 one time series as input whereas it has been shown that using different data sources could improve  
288 forecast Ferté et al. (2022). Therefore, it is still difficult to assess the usefulness of RC over a large  
289 period and using many time series as inputs.

290 RC can be viewed as an extension of penalized linear regression, where inputs undergo processing  
291 by a reservoir, introducing the capacity for memory and non-linear combinations. Given the effec-  
292 tiveness of penalized linear regression in COVID-19 forecasting, as highlighted in Ferté et al. (2022),

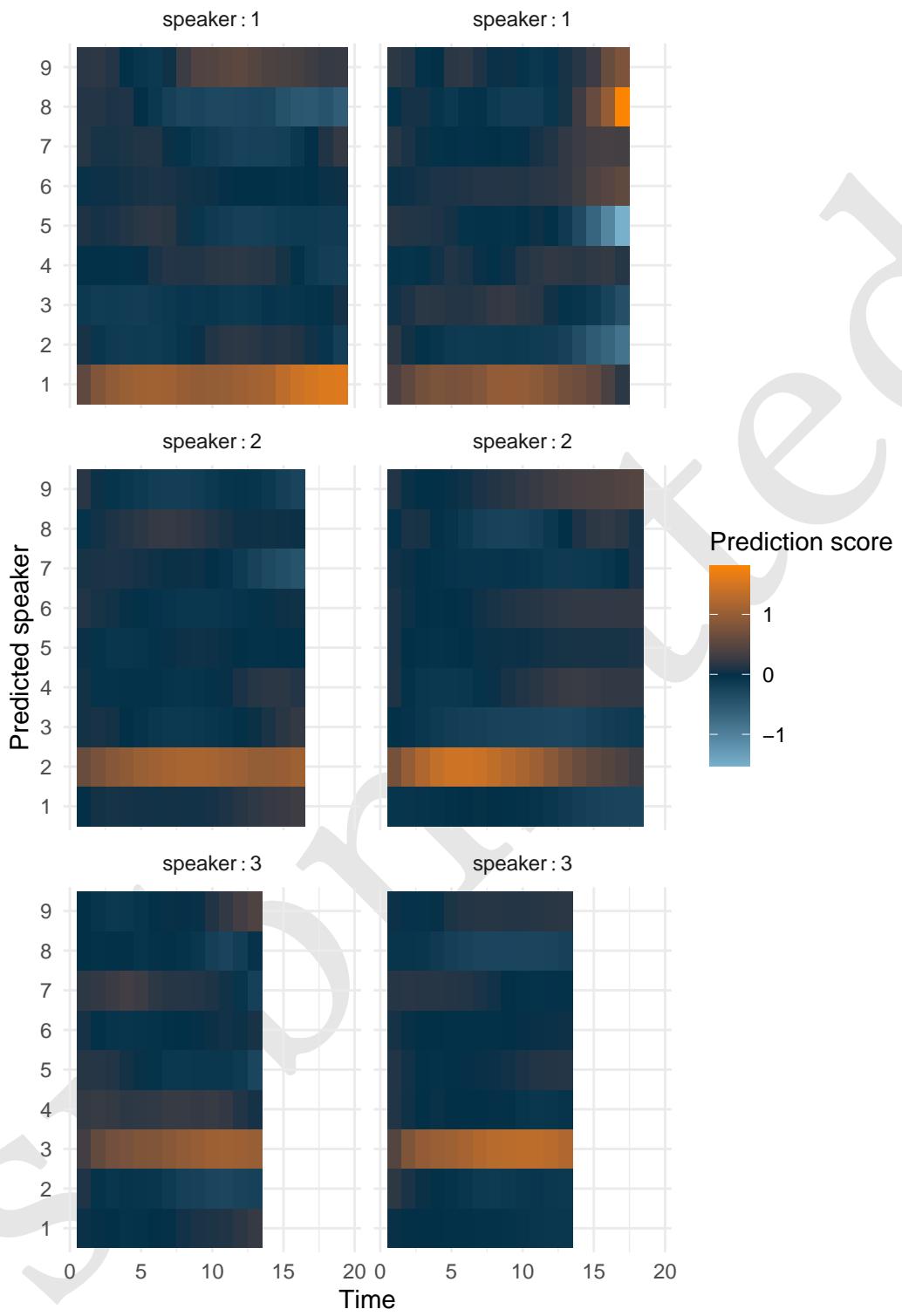


Figure 10: Prediction in a sequence-to-sequence approach 6 samples with 3 speakers and 2 utterance each. The higher the score of the speaker, the lighter the color.

and the promising results exhibited by RC in epidemic forecasting, as demonstrated in studies such as Ghosh et al. (2021), Liu et al. (2023), and Ray, Chakraborty, and Ghosh (2021), we have opted to employ RC for the prediction of hospitalizations at 14 days at the University Hospital of Bordeaux. The primary aim of this study is to assess the performance of RC in this forecasting task. Secondary objectives include (i) comparing the performance of RC with that of elastic-net penalized regression (identified as the optimal model in Ferté et al. (2022)) and (ii) evaluating variations in RC performance based on different architectural choices, such as the connection between the input layer and the output layer, and the use of one input scaling per feature versus a common input scaling.

## 4.2 Methods

### 4.2.1 Data

The study utilized aggregated data spanning from May 16, 2020, to January 17, 2022, regarding the COVID-19 epidemic in France, drawing from various sources to enhance forecasting accuracy. These sources encompassed epidemiological statistics from Santé Publique France, weather data from the National Oceanic and Atmospheric Administration (NOAA), both providing department-level data Etalab (2020) and Electronic Health Record (EHR) data from the Bordeaux Hospital providing hospital-level data. All data were daily updated. Santé Publique France data included information on hospitalizations, RT-PCR tests, positive RT-PCR results, variant prevalence, and vaccination data, categorized by age groups. NOAA data contributed temperature, wind speed, humidity, and dew point data, allowing for the computation of the COVID-19 Climate Transmissibility Predict Index (Roumagnac et al. 2021). EHRs data included hospitalizations, ICU admissions, ambulance service records, and emergency unit notes, with relevant COVID-19-related concepts extracted from the notes. Data are discussed more in depth in Ferté et al. (2022).

First derivative over the last 7 days were computed to enrich model information. To take into account measurement error and daily noise variation, data were smoothed using a local polynomial regression with a span of 21 days. As previously described, input features were scaled between -1 and 1 by dividing the observed value by the maximum of the absolute value of the given input feature.

All data are publicly available. Weather data can be obtained from Smith, Lott, and Vose (2011) using R package `worldmet` (Carslaw 2023). [Vaccine data](#) can be downloaded from Etalab (2020). EHRs data can be downloaded on dryad (Ferté et al. 2023). For privacy issues, publicly available EHRs data below 10 patients were obfuscated to 0. For convenience, all data were downloaded, merged and provided as replication material.

### 4.2.2 Evaluation framework

The task was to forecast 14 days ahead the number of hospitalized patients. As seen at Section 3.3, we will train the model to predict the variation of hospitalization, denoted as  $hosp$ , defined as  $outcome_{t+14} = hosp_{t+14} - hosp_t$ . Metrics computation and visualizations will be performed on the predicted number of hospitalizations denoted as  $\widehat{hosp}_{t+14} = \widehat{outcome}_{t+14} + hosp_t$ .

The dataset was separated into two periods. First period from May 16, 2020 to March 1, 2021 served to identify relevant hyperparameters. Remaining data was used to evaluate the model performance.

The performance of the model was evaluated according to several metrics:

- the mean absolute error :  $MAE = \text{mean}(|\widehat{hosp}_{t+14} - hosp_{t+14}|)$ .

- 334 • the median relative error :  $MRE = \text{median}(|\frac{\widehat{\text{hospt}_{t+14}} - \text{hospt}_{t+14}}{\text{hospt}_{t+14}}|)$ .
- 335 • the mean absolute error to baseline :  $MAEB = \text{mean}(|\widehat{\text{hospt}_{t+14}} - \text{hospt}_{t+14}| - |\text{hospt}_t - \text{hospt}_{t+14}|)$ .
- 336 • the median relative error to baseline :  $MREB = \text{median}(|\frac{\widehat{\text{hospt}_{t+14}} - \text{hospt}_{t+14}}{\text{hospt}_t - \text{hospt}_{t+14}}|)$

337 Median was chosen over mean for  $MRE$  and  $MREB$  because those metrics tend to have extremely  
 338 high values when the denominator is close to 0 (i.e when the number of hospitalized patients is close  
 339 to 0 or the number of patients hospitalized at 14 days is close to the current number of hospitalized  
 340 patients respectively).  $MAEB$  and  $MREB$  compare model performance to a baseline model which  
 341 predicts the current number of hospitalized patients at 14 days. Those metrics help to determine  
 342 the information added by the model and is a good baseline as covid-19 forecast model do not always  
 343 outperform this basic forecast (Cramer et al. (2022)).

344 Because the outcome is obfuscated below 10 hospitalizations for privacy reason, we set both the  
 345 outcome and the forecast to 10 when the observed value was 0 or the forecasted value was below  
 346 10 when evaluating the model performance.

#### 347 4.2.3 Models

348 We compared RC to elastic-net penalized regression (denoted as Enet). Furthermore we evaluated  
 349 RC based on several architectures. First we compared RC with a single input scaling common to all  
 350 features and a RC with on specific input scaling per feature. Second we compared RC where the  
 351 input layer is connected to the output layer in addition to the connection between reservoir and  
 352 output layer. Therefore, five models were evaluated :

- 353 • Elastic-net penalized regression denoted *Enet*
- 354 • RC with a single input scaling and no connection between input and ouput layers denoted  
*Common IS R %»% O*
- 355 • RC with a single input scaling and connection between input and ouput layers denoted *Com-*  
*mon IS I+R %»% O*
- 356 • RC with multiple input scaling and no connection between input and ouput layers denoted  
*Multiple IS R %»% O*
- 357 • RC with multiple input scaling and connection between input and ouput layers denoted *Mul-*  
*tiple IS I+R %»% O*

362 Because of the randomness of the reservoir, we took the median forecast of 10 reservoir on the train  
 363 set to evaluate the performance of a given hyperparameter set. On the test set we aggregated the  
 364 forecast of 40 reservoirs, each of them having one of the 40 best hyperparameter sets found on the  
 365 train set. In addition, because covid-19 hospitalization is a non-stationary process, models were re-  
 366 trained everyday using all previous days. To ease computation burden, only one day over two was  
 367 used to find hyperparameters on the training set.

#### 368 4.2.4 Hyperparameter optimisation using random search

369 RC relies mainly on 4 hyperparameters including the leaking rate (i.e “memory” parameter), spectral  
 370 radius (i.e “chaoticity” parameter), input scaling (i.e “feature gain” parameter) and ridge (i.e penal-  
 371 ization parameter). Input scaling can be either, common to all features or specific to each feature  
 372 which increases the number of hyperparameter by the number of features.

373 Following the notation from `glmnet` package (Friedman, Hastie, and Tibshirani 2010), elastic-net  
374 penalized linear regression relies on two hyperparameters, `lambda` (i.e the penalization parameter)  
375 and `alpha` (i.e the compromise between lasso and ridge penalty)

376 Hyperparameter were selected in the training set (i.e before March 1, 2021) using a wrapper approach  
377 and a random search sampler using 2000 samples for each model. The sampling distribution were  
378 defined as follow :

- 379     • (RC) ridge and (Enet) `lambda` : log-uniform law defined between 1e-10 and 1e5  
380     • (RC) input scaling and spectral radius : log-uniform law defined between 1e-5 and 1e5  
381     • (RC) leaking rate : log-uniform law defined between 1e-3 and 1  
382     • (Enet) `alpha` : uniform defined between 0 and 1

383 We provided large search space for all hyper-parameters. Search space was slightly reduced for  
384 leaking rate based on previous results and because a leaking rate of 1e-3 already imply that new  
385 inputs make the reservoir change really slowly which is not inline with the dynamic of covid-19 but  
386 would be appropriate for an application where the phenomena to forecast has a slow dynamic.

387 Finally, we provided an additional Enet model similar to the one in Ferté et al. (2022) where `alpha`  
388 was set to 0.5 and `lambda` was re-evaluated everyday in the test set based on previous data using  
389 the cross-validation procedure provided by `glmnet`.

390 **4.3 Results**

391 The goal of this task is to predict 14 days ahead the hospitalization. Figure Figure 11 shows both the  
392 training set (i.e before 2021-03-01) and the test set where the blue curve correspond to the input fea-  
393 tures (first derivatives are not shown) and the orange curves correspond to the outcome the model is  
394 trained on (i.e the hospitalization variation) and the hospitalizations at 14 days on which the per-  
395 formance metrics are computed. The figures outline that the relation between the input features and  
396 the outcome evolve over time and that the time series is not stationary. For instance IPTCC (*Index*  
397 *PREDICT de Transmissivité Climatique de la COVID-19*) seems correlated to the outcome except that  
398 it completely miss the summer 2021 increase.

399 **4.3.1 Hyperparameter selection**

400 Figure Figure 12 shows the hyperparameter optimisation using random search for the different RC  
401 architectures. We observe that model with multiple input scaling achieved better performance on  
402 the train set compared to model with single input scaling which is expected as they can adapt more  
403 closely to the data thanks to specific input scaling for each feature.

404 As expected, we observe that the optimal leaking rate is above 1e-2 for all RC which is coherent  
405 with the short term dynamic of covid-19 epidemic. Trend for other hyperparameter are less clear  
406 even though best hyperparameter sets were close for RC with common input scaling and for RC  
407 with multiple input scaling.

408 Figure Figure 13 shows the hyperparameter search for RC with multiple input scaling and connected  
409 input layer. We observe that the random search tend to favor high importance given to derivative  
410 of positive RT-PCR (including the elderly) and the derivative of IPTCC. The rest of the feature do  
411 not show clear pattern.

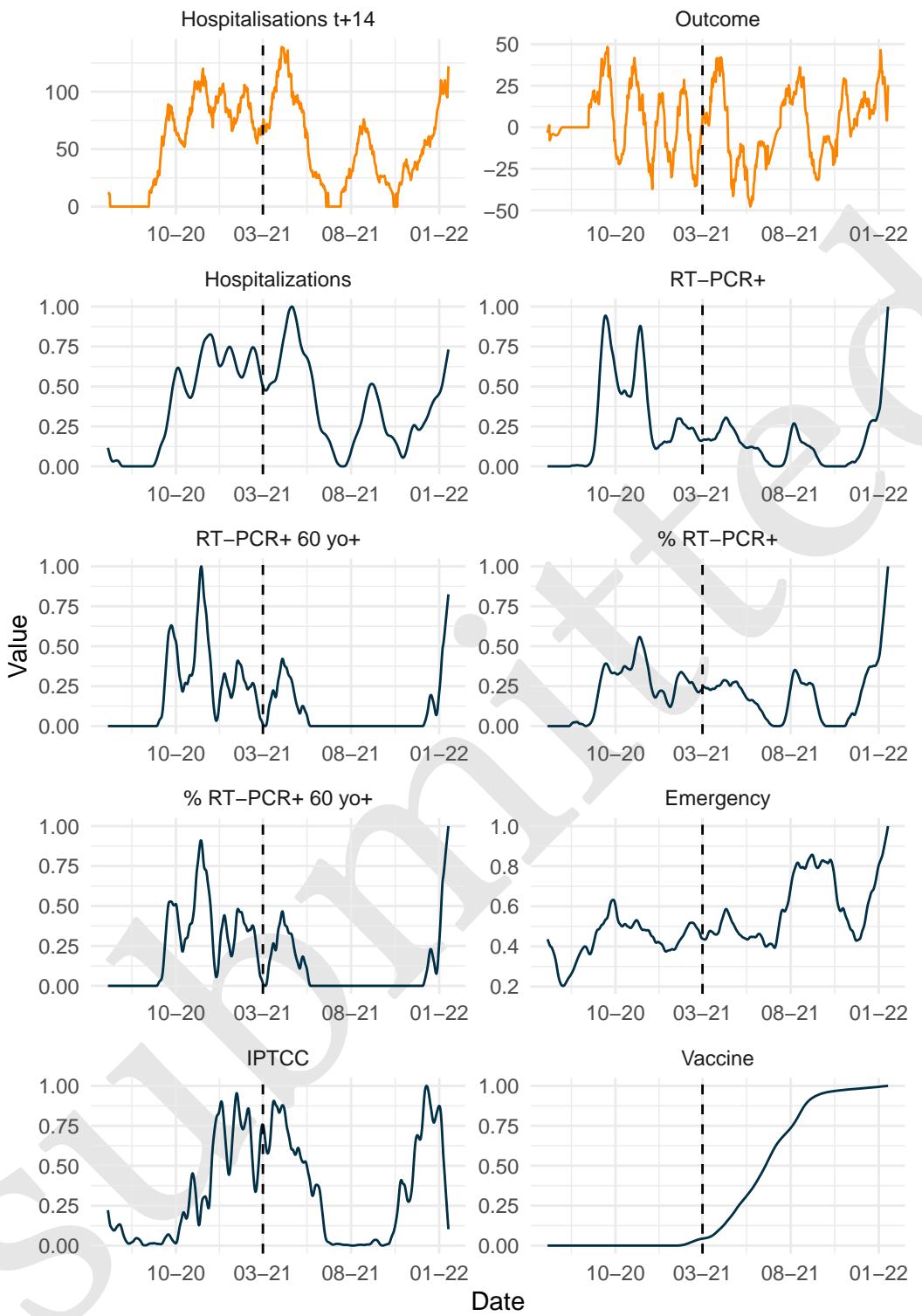


Figure 11: Covid-19 epidemic at BUH. Outcome of interest is presented in orange. Model is trained to forecast Outcome curve which corresponds to the difference between Hospitalisations at 14 days and current hospitalisations. Other features are scaled (divide by the maximum of the feature) represented in darkblue.

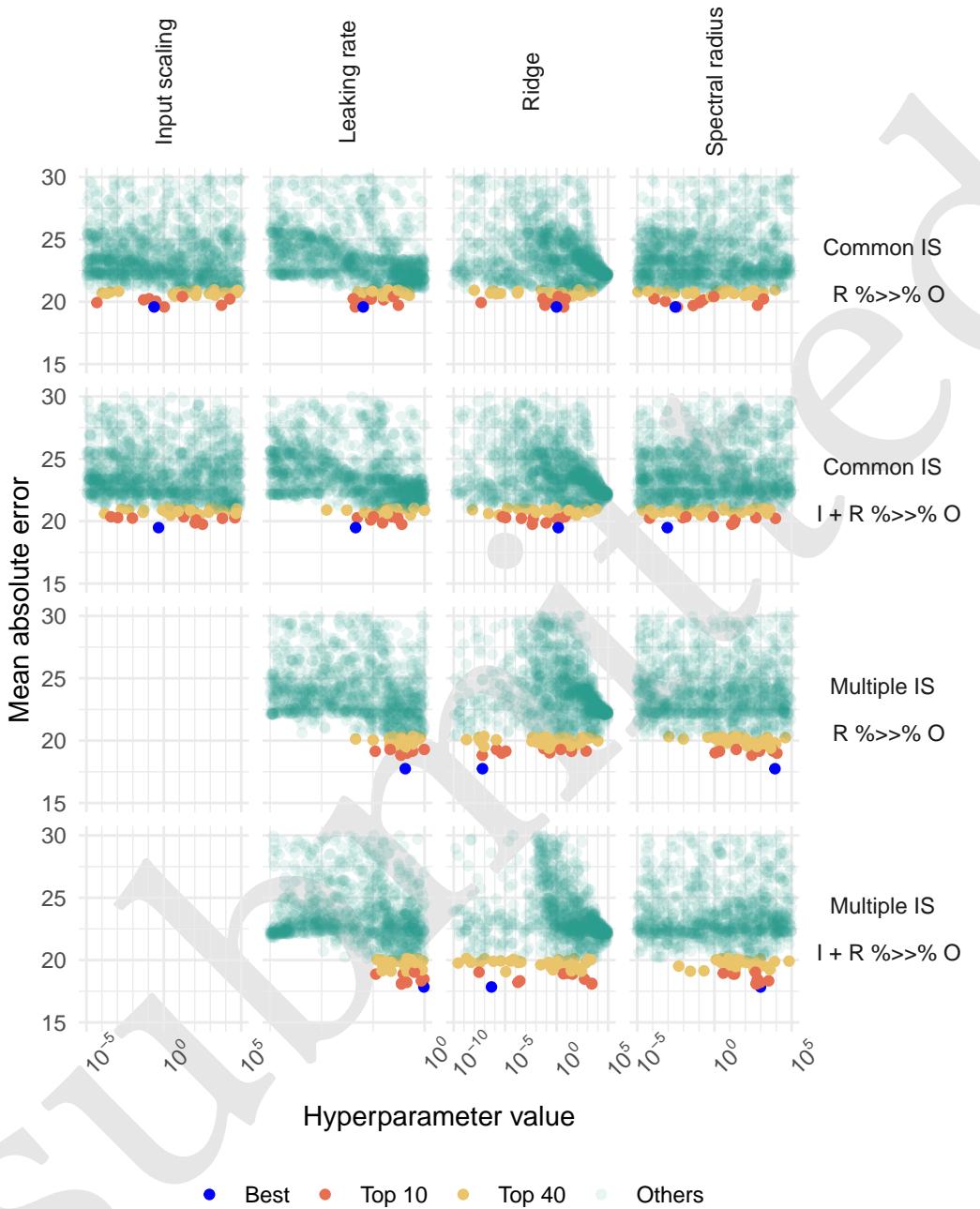


Figure 12: Hyperparameter evaluation on training set by random search. Hp sets with MAE above 30 were removed for clarity of visualisation.

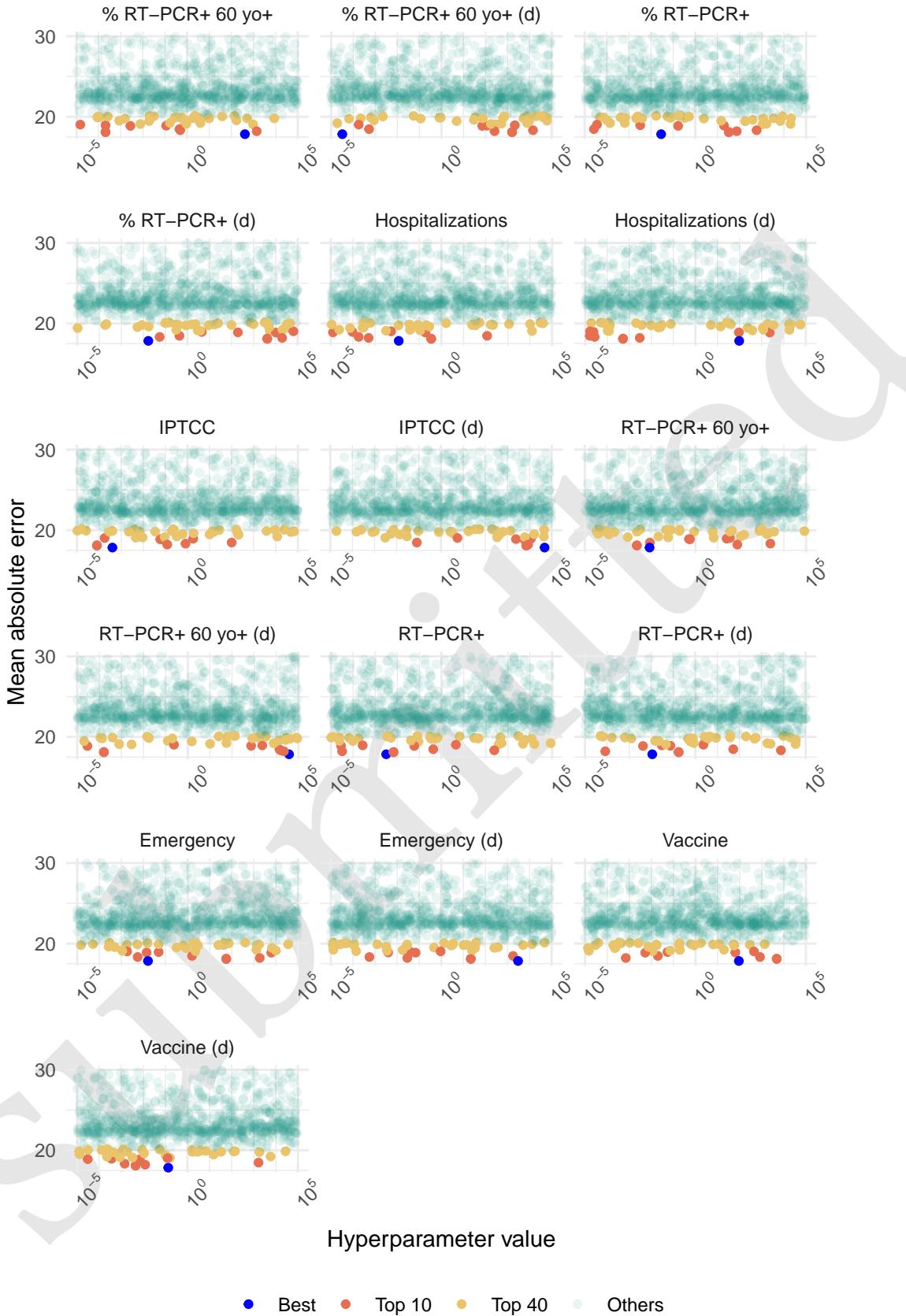


Figure 13: Hyperparameter evaluation on training set by random search of the model with multiple input scaling and no connection between input layer and output layer. Hp sets with MAE above 30 were removed for clarity of visualisation.

Table 1: Model performance with several reservoir configuration. For each setting, 40 reservoirs are computed and the forecast is the median of the 40 forecasts. Results show the performance metrics : MAE = Mean Absolute Error, MRE = Median Relative Error, MAEB = Mean Absolute Error to Baseline, MREB = Median Relative Error to Baseline.

### Model Performance

Model	MAE	MRE	MAEB	MREB
Common IS R %»% O	15.23	0.26	-3.50	0.85
Common IS I + R %»% O	14.84	0.26	-3.89	0.83
Multiple IS R %»% O	15.38	0.28	-3.35	0.82
Multiple IS I + R %»% O	15.25	0.28	-3.49	0.83
Elastic-net	16.40	0.29	-2.34	0.93

412 **4.3.2 Forecast performance**

413 Table 1 shows the performance on the test set. Best model according to all metrics was RC with  
414 common input scaling and connection between input and output layers. Having one input scaling  
415 per feature did not improve the model which might be due to low generalisability of the hyperpa-  
416 rameter of the training set to the test set due to non-stationarity. Additionally, connecting input layer  
417 to output layer improved the model forecast. All RC models performed better than the elastic-net  
418 model.

419 Figure 14 shows the forecast of the different models. We note that models struggle to accurately  
420 forecast slope shifts. For instance, summer 2021 initial increase is partially predicted by all models  
421 but its decrease is not well predicted. Winter 2021 increase is anticipated by all models but they  
422 tend to overestimate it because of the rise of vaccine effect.

423 **4.3.3 Number of model to aggregate**

424 Figure 15 show the individual forecast for the 40 best sets of hyperparameters of each RC architec-  
425 ture. Due to the internal random connection of the reservoir, we observe forecast stochasticity and  
426 relying on only one forecast is unreliable. We explored the number of model needed at Figure 16  
427 which shows that after 10 models, forecast is stable and even 5 models for the simpler model with  
428 common input scaling which rely on less hyperparamters.

429 **4.3.4 Input feature importance**

430 We compared the coefficients of the output layer estimated for the input layer and the reservoir  
431 nodes. Additionally, we compared the coefficient given to the input layer by the output layer in the  
432 reservoir and the coefficient estimated by the elastic-net model.

433 Figure 17 illustrates the ranking of input layer compared to all connections to the output layer,  
434 including the 500 reservoir nodes and the 16 features of the input layer (excluding bias). The figure  
435 shows that the model with common input scaling tends to assign less weight to input layer compared  
436 to the model with multiple input scaling. This suggests that the reservoir with common input scaling  
437 provides more information than the reservoir with multiple input scaling, which aligns with its  
438 better performance, as shown in Table Table 1.

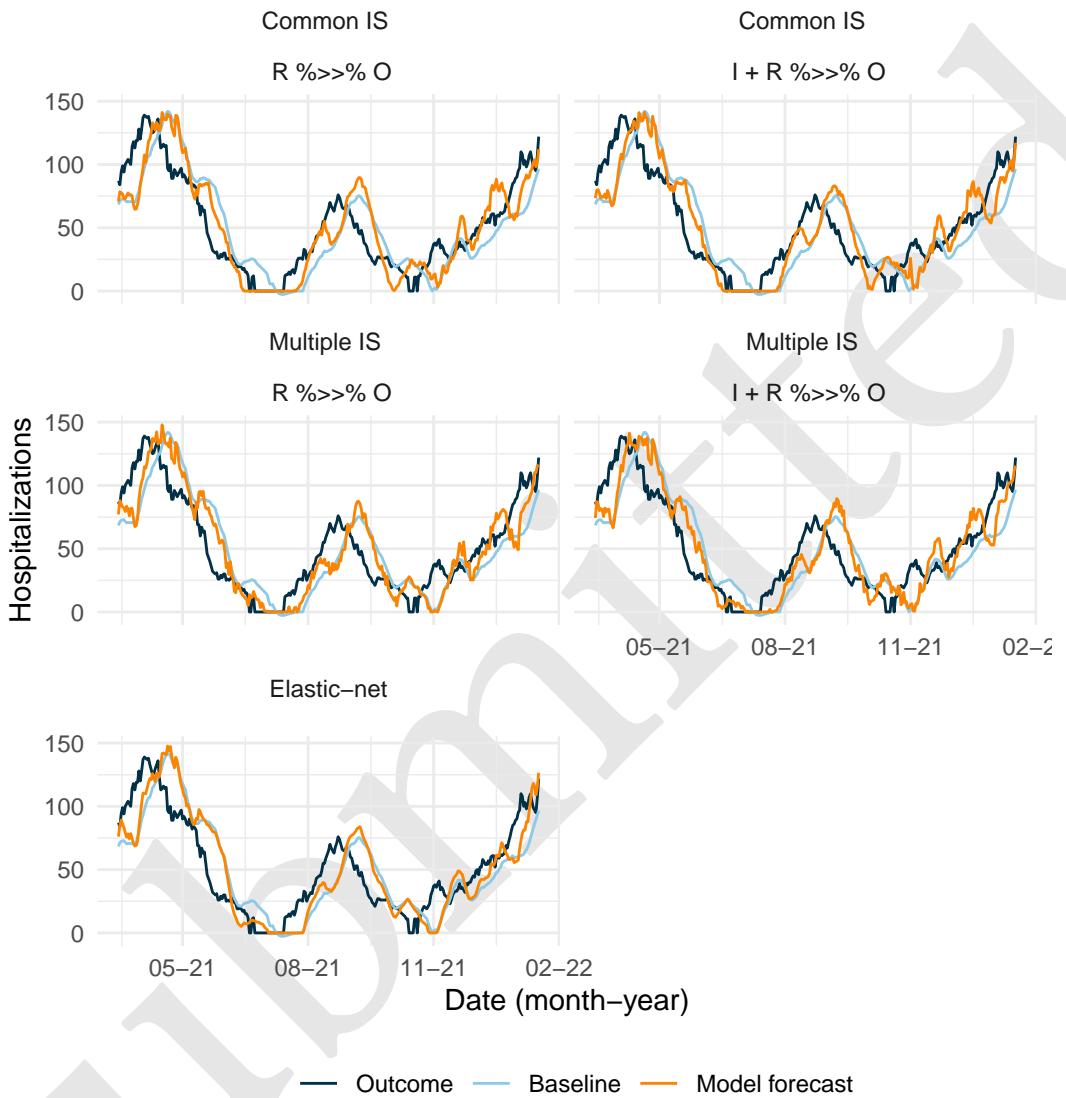


Figure 14: Reservoir computing forecast depending on the setting with and without monthly update. Red line is the median forecast of 40 reservoirs. Grey lines are individual forecast of each of the 40 reservoirs.

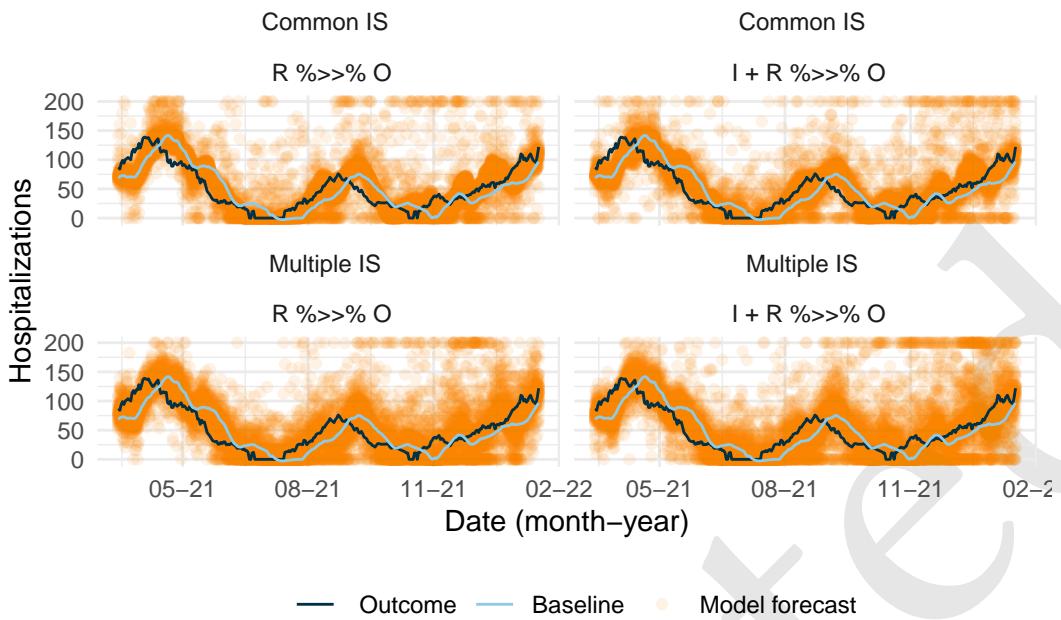


Figure 15: Individual forecast the 40 best hyperparameter sets for the different RC configuration. Forecast value above 200 were set to 200 for clarity.

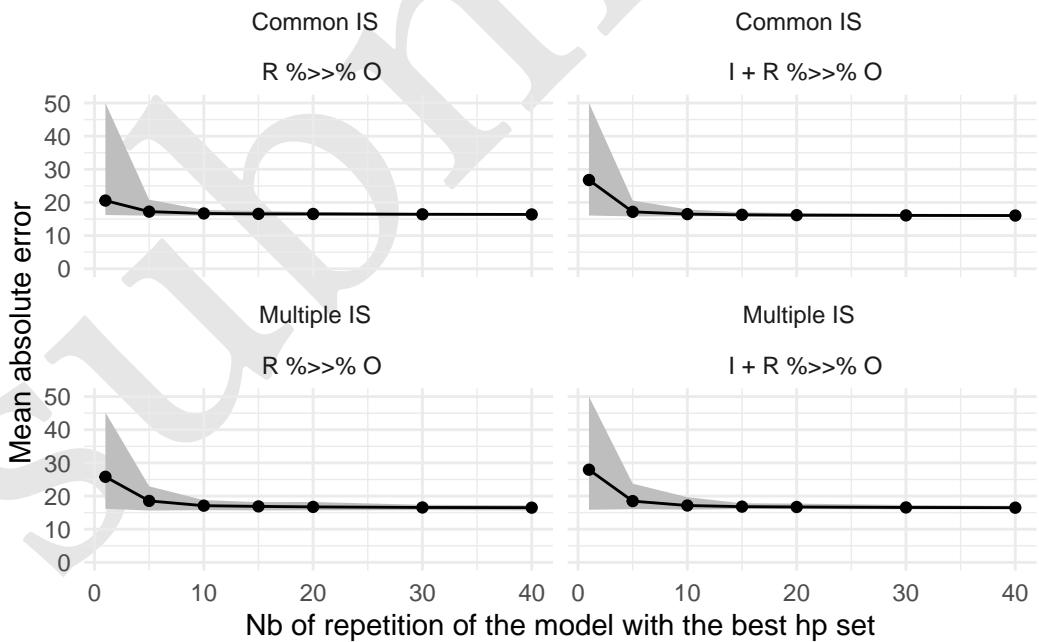


Figure 16: Mean absolute error depending on the number of aggregated reservoir.

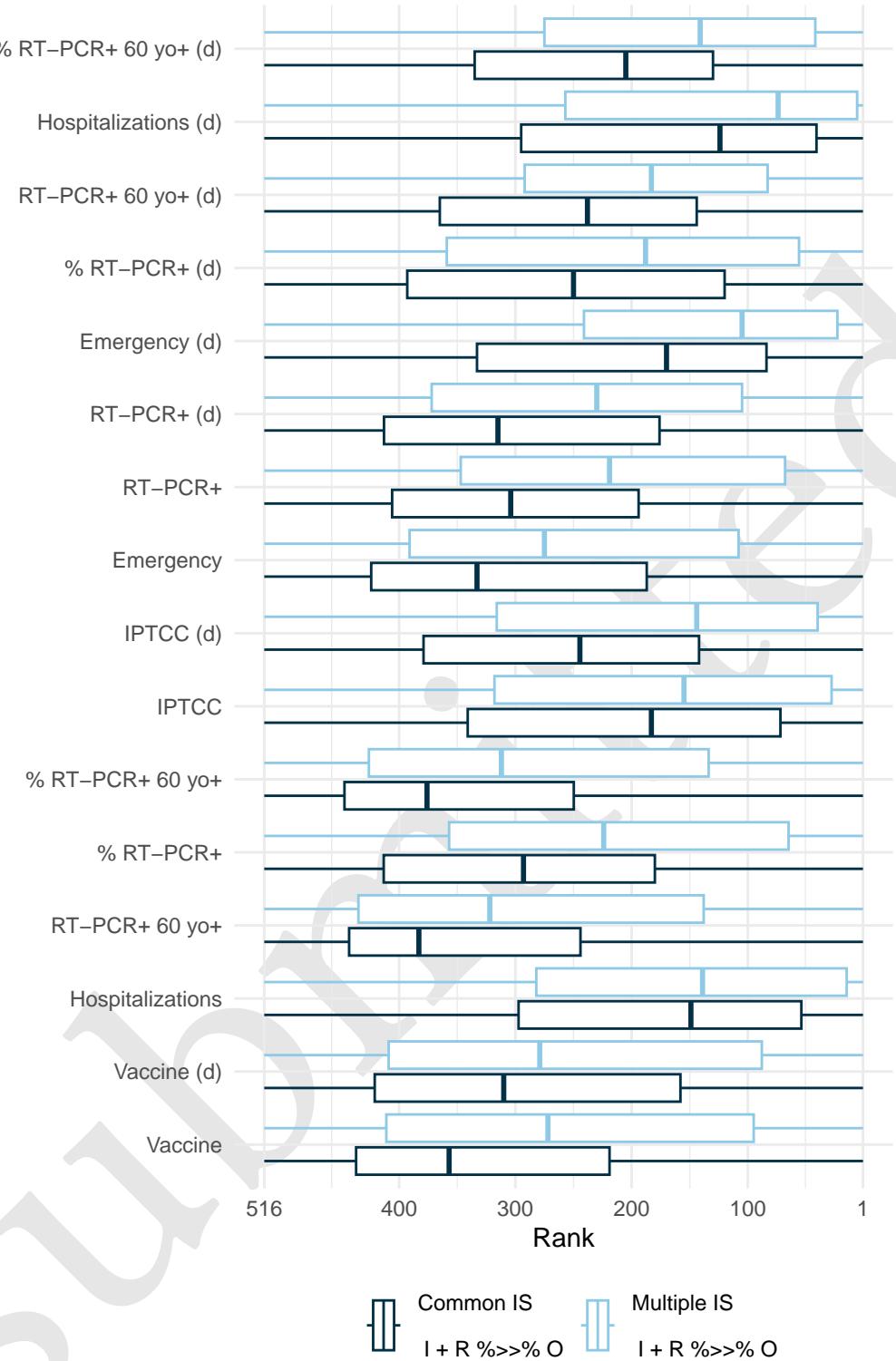


Figure 17: Mean feature importance of the 40 best hyperparameter sets by model, focus on the connection between the input and output layers. Models with direct connection between input and output layer are included. The rank is obtained by comparing the feature input layer and all other connection coefficients (both input and reservoir corresponding coefficients) attributed by the output layer at each date for each hyperparameter set. The higher the output layer's coefficient for the input layer, the closer its rank will be to 1 and the more important the feature is.

439 Furthermore, Figure 18 compares the coefficients assigned to input features by the elastic-net model  
440 and the RC models. While the coefficients are generally consistent across RC models, there are some  
441 notable differences with elastic-net. Specifically, certain features deemed important by the elastic-  
442 net model, such as the derivative of RT-PCR, and the derivative of Vaccine, are less important for  
443 the reservoir computing model. This may indicate that these features predictive ability is better  
444 conveyed by their relationship with other features, which is captured by the reservoir computing  
445 model but might not be by the elastic-net model. Conversely, emergency, IPTCC, proportion of  
446 positive RT-PCR, and hospitalizations are more important for the reservoir computing model than  
447 for the elastic-net model.

#### 448 4.4 Discussion

449 In this specific application, we have demonstrated that RC exhibits commendable performance in  
450 comparison to Elastic-net, which serves as the reference model. Furthermore, we highlight the  
451 inherent challenges in forecasting within this context, primarily stemming from the non-stationarity  
452 of the time series.

453 All computations in this study were conducted using the `reservoirnet` package, and the entire  
454 codebase is accessible on Zenodo (Ferté et al. 2024). This R package demonstrates its efficacy in  
455 implementing various reservoir architectures, including connection between the input layer and  
456 the output layer, as well as the utilization of several input scaling, all within the context of a real-  
457 world use case.

458 Given the substantial number of hyperparameters involved, we acknowledge that random search  
459 may not be the most efficient optimization algorithm. We have retained this approach for the sake  
460 of simplicity in this tutorial paper; however, meta-heuristic approaches, particularly those utilizing  
461 evolutionary algorithms, may prove more efficient, especially when employing multiple input  
462 scaling (Bala et al. 2018).

463 This study represents a novel contribution to epidemic forecasting utilizing RC. Notably, previous lit-  
464 erature predominantly focused on simpler problems characterized by fewer input features or shorter  
465 evaluation periods Ghosh et al. (2021). Our findings underscore the potential of this approach for  
466 future epidemics, suggesting its potential to surpass more traditional epidemiological tools while  
467 maintaining a lightweight model structure compared to other RNNs.

468 It is worth noting that all models, including the Ferté et al. (2022) models, encounter challenges  
469 in accurately anticipating slope shifts, indicating the need for further investigation. Specifically,  
470 additional work is warranted to extend the application of RC to high-dimensional settings, building  
471 upon the insights gained from models based on a more extensive set of features.

### 472 5 Discussion and conclusion

473 In this paper, we introduce the R package `reservoirnet`, which serves as a versatile tool for imple-  
474 menting reservoir computing based on `ReservoirPy`'s Python library. It offers flexibility in defining  
475 the reservoir architecture, including options for specifying connections between the input layer and  
476 the output layer, as well as variations in input scaling as demonstrated on a real-world use case.

477 We provided a comprehensive overview of the basic usage of the `reservoirnet` package through  
478 illustrative examples in regression and classification tasks. This introductory section serves as a  
479 foundation for R users, offering step-by-step guidance on constructing and training reservoir com-  
480 puting models using the package. By demonstrating the application of RC in both regression and

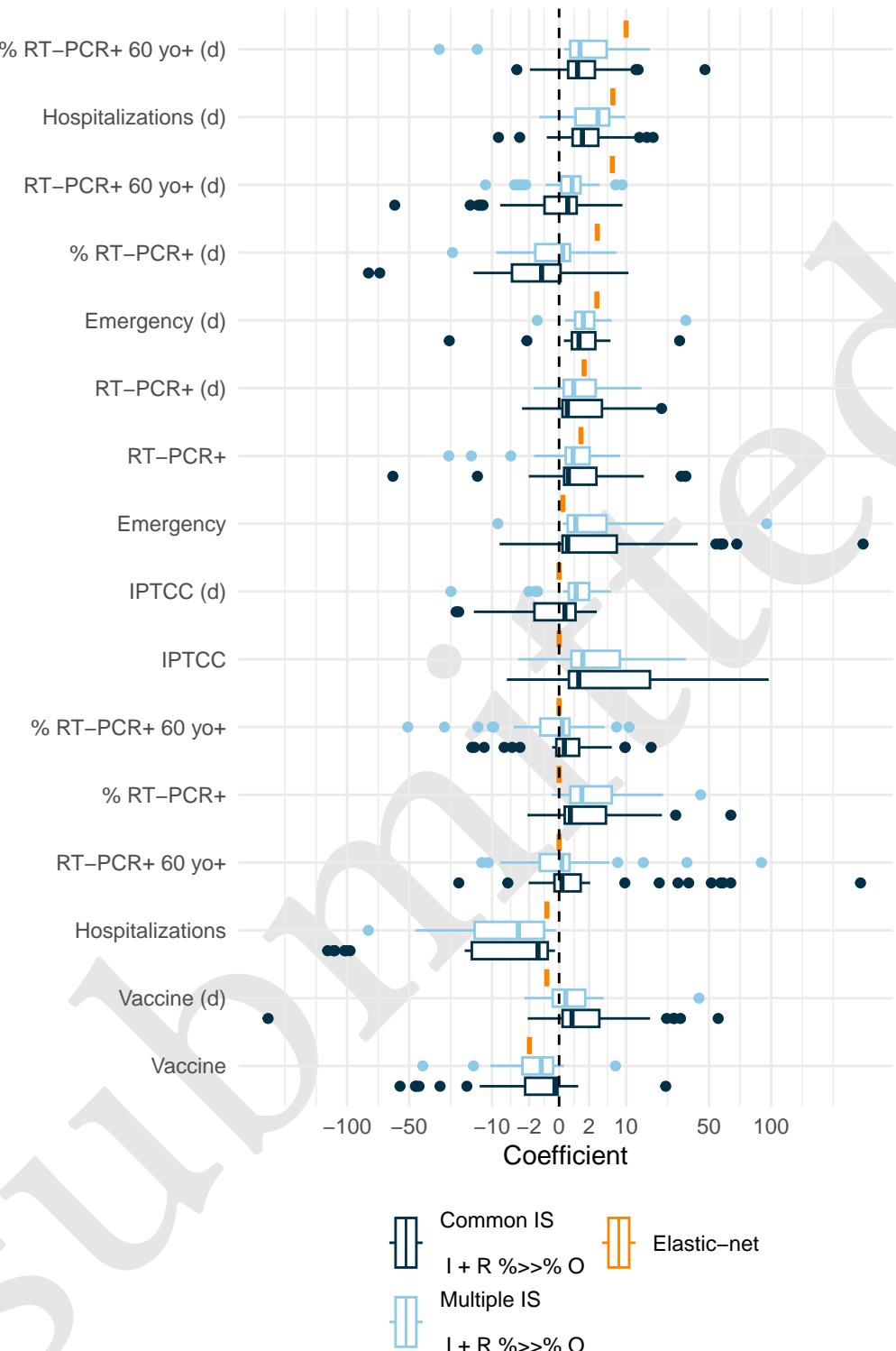


Figure 18: Mean feature coefficient of the 40 best hyperparameter sets by model and the elastic-net model. Only models with direct connection between input and output layer are included. The coefficients were calculated as the average value across all dates for each feature, model and hyperparameter set.

481 classification scenarios, we aim to equip users with the essential knowledge and skills needed to  
482 harness the capabilities of reservoir computing for diverse tasks.

483 Drawing on the robust foundation of the ReservoirPy structure, a well-maintained Python library,  
484 this package inherits its reliability and longevity. We have focused on providing access to the fun-  
485 damental features, building upon the strong base provided by ReservoirPy. Therefore, this initial  
486 version of reservoirnet must evolve in tandem with the growing understanding and adoption of  
487 RC within the R community.

## 488 References

- 489 Bala, Abubakar, Idris Ismail, Rosdiazli Ibrahim, and Sadiq M. Sait. 2018. “Applications of Meta-  
490 heuristics in Reservoir Computing Techniques: A Review.” *IEEE Access* 6: 58012–29. <https://doi.org/10.1109/ACCESS.2018.2873770>.
- 492 Carrat, Fabrice, Julie Figoni, Joseph Henny, Jean-Claude Desenclos, Sofiane Kab, Xavier de Lambal-  
493 lerie, and Marie Zins. 2021. “Evidence of Early Circulation of SARS-CoV-2 in France: Findings  
494 from the Population-Based ‘CONSTANCES’ Cohort.” *European Journal of Epidemiology*, Febru-  
495 ary, 1–4. <https://doi.org/10.1007/s10654-020-00716-2>.
- 496 Carslaw, David. 2023. “Worldmet: Import Surface Meteorological Data from NOAA Integrated  
497 Surface Database (ISD).” <https://cran.r-project.org/web/packages/worldmet/index.html>.
- 498 Carvalho, Kathleen, João Paulo Vicente, Mihajlo Jakovljevic, and João Paulo Ramos Teixeira. 2021.  
499 “Analysis and Forecasting Incidence, Intensive Care Unit Admissions, and Projected Mortality  
500 Attributable to COVID-19 in Portugal, the UK, Germany, Italy, and France: Predictions for 4  
501 Weeks Ahead.” *Bioengineering* 8 (6): 84. <https://doi.org/10.3390/bioengineering8060084>.
- 502 COVID-19 Cumulative Infection Collaborators. 2022. “Estimating Global, Regional, and National  
503 Daily and Cumulative Infections with SARS-CoV-2 Through Nov 14, 2021: A Statistical Analy-  
504 sis.” *Lancet*. [https://doi.org/10.1016/S0140-6736\(22\)00484-6](https://doi.org/10.1016/S0140-6736(22)00484-6).
- 505 Cramer, Estee Y., Evan L. Ray, Velma K. Lopez, Johannes Bracher, Andrea Brennen, and et al. 2022.  
506 “Evaluation of Individual and Ensemble Probabilistic Forecasts of COVID-19 Mortality in the  
507 United States.” *Proceedings of the National Academy of Sciences* 119 (15): e2113561119. <https://doi.org/10.1073/pnas.2113561119>.
- 509 Etalab. 2020. “Les Données Relatives Au COVID-19 En France - Data.gouv.fr.” <https://www.data.gouv.fr/fr/pages/donnees-coronavirus/>.
- 511 Ferté, Thomas, Kalidou Ba, Dan Dutartre, Pierrick Legrand, Vianney Jouhet, Romain Griffier,  
512 Rodolphe Thiébaut, Xavier Hinaut, and Boris P. Hejblum. 2024. “Thomasferte/Jss\_reservoirnet:  
513 First Release.” Zenodo. <https://doi.org/10.5281/ZENODO.11281341>.
- 514 Ferté, Thomas, Vianney Jouhet, Romain Griffier, Boris P. Hejblum, Rodolphe Thiébaut, and Bordeaux  
515 University Hospital Covid-19 Crisis Task Force. 2022. “The Benefit of Augmenting Open Data  
516 with Clinical Data-Warehouse EHR for Forecasting SARS-CoV-2 Hospitalizations in Bordeaux  
517 Area, France.” *JAMIA Open* 5 (4): ooac086. <https://doi.org/10.1093/jamiaopen/ooac086>.
- 518 Ferté, Thomas, Vianney Jouhet, Romain Griffier, Boris Hejblum, Rodolphe Thiébaut, and Bordeaux  
519 University Hospital Covid-19 Crisis Task Force. 2023. “The Benefit of Augmenting Open Data  
520 with Clinical Data-Warehouse EHR for Forecasting SARS-CoV-2 Hospitalizations in Bordeaux  
521 Area, France.” Dryad. <https://doi.org/10.5061/DRYAD.HHMGQNKX>.
- 522 Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. 2010. “Regularization Paths for Generalized  
523 Linear Models via Coordinate Descent.” *Journal of Statistical Software* 33 (1): 1–22.
- 524 Ghosh, Subrata, Abhishek Senapati, Arindam Mishra, Joydev Chattopadhyay, Syamal K. Dana, Chit-  
525 taranjan Hens, and Dibakar Ghosh. 2021. “Reservoir Computing on Epidemic Spreading: A Case  
526 Study on COVID-19 Cases.” *Physical Review E* 104 (1): 014308. <https://doi.org/10.1103/PhysRevE.104.014308>.

- 528 Hinaut, Xavier, and Peter Ford Dominey. 2013. “Real-Time Parallel Processing of Grammatical  
529 Structure in the Fronto-Striatal System: A Recurrent Network Simulation Study Using Reser-  
530 voir Computing.” *PLOS ONE* 8 (2): e52946. <https://doi.org/10.1371/journal.pone.0052946>.
- 531 Hübner, Martin, Tobias Zingg, David Martin, Philippe Eckert, and Nicolas Demartines. 2020.  
532 “Surgery for Non-Covid-19 Patients During the Pandemic.” *PLoS ONE* 15 (10): e0241331.  
533 <https://doi.org/10.1371/journal.pone.0241331>.
- 534 Jaeger, Herbert. 2001. “The” Echo State” Approach to Analysing and Training Recurrent Neural  
535 Networks-with an Erratum Note.” *Bonn, Germany: German National Research Center for Infor-  
536 mation Technology GMD Technical Report* 148 (January).
- 537 Kim, Gina, Mengru Wang, Hanh Pan, Giana H. Davidson, Alison C. Roxby, Jen Neukirch, Danna Lei,  
538 Elicia Hawken-Dennis, Louise Simpson, and Thuan D. Ong. 2020. “A Health System Response  
539 to COVID-19 in Long-Term Care and Post-Acute Care: A Three-Phase Approach.” *Journal of the  
540 American Geriatrics Society* 68 (6): 1155–61. <https://doi.org/10.1111/jgs.16513>.
- 541 Kmet, Tibor, and Maria Kmetova. 2019. “Bézier Curve Parametrisation and Echo State Network  
542 Methods for Solving Optimal Control Problems of SIR Model.” *Biosystems* 186 (December):  
543 104029. <https://doi.org/10.1016/j.biosystems.2019.104029>.
- 544 Kudo, Mineichi, Jun Toyama, and Masaru Shimbo. 1999. “Multidimensional Curve Classification  
545 Using Passing-Through Regions.” *Pattern Recognition Letters* 20 (11): 1103–11. [https://doi.org/10.1016/S0167-8655\(99\)00077-X](https://doi.org/10.1016/S0167-8655(99)00077-X).
- 546 Liu, Bocheng, Yiyuan Xie, Weichen Liu, Xiao Jiang, Yichen Ye, Tingting Song, Junxiong Chai, Many-  
547 ing Feng, and Haodong Yuan. 2023. “Nanophotonic Reservoir Computing for COVID-19 Pan-  
548 demic Forecasting.” *Nonlinear Dynamics* 111 (7): 6895–6914. [https://doi.org/10.1007/s11071-022-08190-z](https://doi.org/10.1007/s11071-022-<br/>549 08190-z).
- 550 Lukoševičius, Mantas, and Herbert Jaeger. 2009. “Reservoir Computing Approaches to Recurrent  
551 Neural Network Training.” *Computer Science Review* 3 (3): 127–49. [cosrev.2009.03.005](https://doi.org/10.1016/j.<br/>552 cosrev.2009.03.005).
- 553 Maass, Wolfgang, Thomas Natschläger, and Henry Markram. 2002. “Real-Time Computing With-  
554 out Stable States: A New Framework for Neural Computation Based on Perturbations.” *Neural  
555 Computation* 14 (11): 2531–60. <https://doi.org/10.1162/089976602760407955>.
- 556 Martinuzzi, Francesco, Chris Rackauckas, Anas Abdelrehim, Miguel D. Mahecha, and Karin Mora.  
557 2022. “ReservoirComputing.jl: An Efficient and Modular Library for Reservoir Computing Mod-  
558 els.” *Journal of Machine Learning Research* 23 (288): 1–8. [http://jmlr.org/papers/v23/22-0611.html](http://jmlr.org/papers/v23/22-0611.<br/>559 html).
- 560 Mohimont, Lucas, Amine Chemchem, François Alin, Michaël Krajecki, and Luiz Angelo Steffenel.  
561 2021. “Convolutional Neural Networks and Temporal CNNs for COVID-19 Forecasting in  
562 France.” *Applied Intelligence*, April. <https://doi.org/10.1007/s10489-021-02359-6>.
- 563 Nakane, Ryosho, Gouhei Tanaka, and Akira Hirose. 2018. “Reservoir Computing With Spin Waves  
564 Excited in a Garnet Film.” *IEEE Access PP* (January): 1–1. [https://doi.org/10.1109/ACCESS.2018.2794584](https://doi.org/10.1109/ACCESS.2018.<br/>565 2794584).
- 566 Paireau, Juliette, Alessio Andronico, Nathanaël Hozé, Maylis Layen, Pascal Crépey, Alix Roumagnac,  
567 Marc Lavielle, Pierre-Yves Boëlle, and Simon Cauchemez. 2022. “An Ensemble Model Based  
568 on Early Predictors to Forecast COVID-19 Health Care Demand in France.” *Proceedings of the  
569 National Academy of Sciences* 119 (18): e2103302119. <https://doi.org/10.1073/pnas.2103302119>.
- 570 Penkovsky, Bogdan, Laurent Larger, and Daniel Brunner. 2018. “Efficient Design of Hardware-  
571 Enabled Reservoir Computing in FPGAs.” *Journal of Applied Physics* 124 (16): 162101. [https://doi.org/10.1063/1.5039826](https://<br/>572 doi.org/10.1063/1.5039826).
- 573 Pottier, Loïc. 2021. “Forecast of the Covid19 Epidemic in France.” *medRxiv*. [https://doi.org/10.1101/2021.04.13.21255418](https://doi.org/10.1101/<br/>574 2021.04.13.21255418).
- 575 Prychynenko, Diana, Matthias Sitte, Kai Litzius, Benjamin Krüger, George Bourianoff, Mathias  
576 Kläui, Jairo Sinova, and Karin Everschor-Sitte. 2018. “Magnetic Skyrmion as a Nonlinear Re-

- 578 sistive Element: A Potential Building Block for Reservoir Computing.” *Physical Review Applied*  
579 9 (1): 014034. <https://doi.org/10.1103/PhysRevApplied.9.014034>.
- 580 Rafayelyan, Mushegh, Jonathan Dong, Yongqi Tan, Florent Krzakala, and Sylvain Gigan. 2020.  
581 “Large-Scale Optical Reservoir Computing for Spatiotemporal Chaotic Systems Prediction.”  
582 *Physical Review X* 10 (4): 041037. <https://doi.org/10.1103/PhysRevX.10.041037>.
- 583 Rahimi, Iman, Fang Chen, and Amir H. Gandomi. 2021. “A Review on COVID-19 Forecasting  
584 Models.” *Neural Computing & Applications*, February, 1–11. <https://doi.org/10.1007/s00521-020-05626-8>.
- 585 Ray, Arnob, Tanujit Chakraborty, and Dibakar Ghosh. 2021. “Optimized Ensemble Deep Learning  
586 Framework for Scalable Forecasting of Dynamics Containing Extreme Events.” *Chaos (Wood-  
587 bury, N.Y.)* 31 (11): 111105. <https://doi.org/10.1063/5.0074213>.
- 588 Roumagnac, Alix, Eurico de Carvalho Filho, Raphaël Bertrand, Anne-Kim Banchereau, and Guil-  
589 laume Lahache. 2021. “Étude de l’influence Potentielle de l’humidité Et de La Température  
590 Dans La Propagation de La Pandémie COVID-19.” *Médecine de Catastrophe - Urgences Collec-  
591 tives, Douleur et situations d’exceptionPandémie COVID-19*, 5 (1): 87–102. <https://doi.org/10.1016/j.pxur.2021.01.002>.
- 592 Simões, Jorge, João Paulo Moreira Magalhães, André Biscaia, António da Luz Pereira, Gonçalo  
593 Figueiredo Augusto, and Inês Fronteira. 2021. “Organisation of the State, Model of Health Sys-  
594 tem and COVID-19 Health Outcomes in Six European Countries, During the First Months of  
595 the COVID-19 Epidemic in 2020.” *The International Journal of Health Planning and Management*,  
596 June, 10.1002/hpm.3271. <https://doi.org/10.1002/hpm.3271>.
- 597 Smith, Adam, Neal Lott, and Russ Vose. 2011. “The Integrated Surface Database: Recent De-  
598 velopments and Partnerships.” *Bulletin of the American Meteorological Society* 92 (6): 704–8.  
599 <https://doi.org/10.1175/2011BAMS3015.1>.
- 600 Tanaka, Gouhei, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji  
601 Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. 2019. “Recent Advances in Physical  
602 Reservoir Computing: A Review.” *Neural Networks* 115 (July): 100–123. <https://doi.org/10.1016/j.neunet.2019.03.005>.
- 603 Trouvain, Nathan, and Xavier Hinaut. 2021. “Canary Song Decoder: Transduction and Implicit  
604 Segmentation with ESNs and LTSMs.” In *Artificial Neural Networks and Machine Learning –  
605 ICANN 2021*, edited by Igor Farkaš, Paolo Masulli, Sebastian Otte, and Stefan Wermter, 71–82.  
606 Lecture Notes in Computer Science. Cham: Springer International Publishing. [https://doi.org/10.1007/978-3-030-86383-8\\_6](https://doi.org/10.1007/978-3-030-86383-8_6).
- 607 ———. 2022. “Reservoirpy: A Simple and Flexible Reservoir Computing Tool in Python.” <https://inria.hal.science/hal-03699931>.
- 608 Trouvain, Nathan, Luca Pedrelli, Thanh Trung Dinh, and Xavier Hinaut. 2020. “ReservoirPy: An  
609 Efficient and User-Friendly Library to Design Echo State Networks.” In *Artificial Neural Networks  
610 and Machine Learning – ICANN 2020*, 494–505. Springer International Publishing. <https://inria.hal.science/hal-02595026>.
- 611 Trouvain, Nathan, Nicolas Rougier, and Xavier Hinaut. 2022. “Create Efficient and Complex Reser-  
612 voir Computing Architectures with ReservoirPy.” In *From Animals to Animats 16*, edited by Lola  
613 Cañamero, Philippe Gaussier, Myra Wilson, Sofiane Boucenna, and Nicolas Cuperlier, 91–102.  
614 Lecture Notes in Computer Science. Cham: Springer International Publishing. [https://doi.org/10.1007/978-3-031-16770-6\\_8](https://doi.org/10.1007/978-3-031-16770-6_8).
- 615 Ushey, Kevin, JJ Allaire, and Yuan Tang. 2024. *Reticulate: Interface to ‘Python’*. <https://rstudio.github.io/reticulate/>.
- 616 Vlachas, P. R., J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos. 2020.  
617 “Backpropagation Algorithms and Reservoir Computing in Recurrent Neural Networks for the  
618 Forecasting of Complex Spatiotemporal Dynamics.” *Neural Networks* 126 (June): 191–217. <https://doi.org/10.1016/j.neunet.2020.02.016>.

- 628 Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. “dplyr:  
 629     A Grammar of Data Manipulation.” <https://CRAN.R-project.org/package=dplyr>.
- 630 Wickham, Hadley, Danielle Navarro, and Thomas Lin Pedersen. 2018. *ggplot2: Elegant Graphics for  
 631     Data Analysis (3e)*. 3rd ed. Springer-Verlag New York. <https://ggplot2-book.org/>.
- 632 World Health Organisation. 2020. “WHO Coronavirus (COVID-19) Dashboard.” [https://covid19.  
 who.int](https://covid19.<br/>
  633     who.int).
- 634 Zhang, Qihuang, Grace Y. Yi, Li-Pang Chen, and Wenqing He. 2023. “Sentiment Analysis and Causal  
 635     Learning of COVID-19 Tweets Prior to the Rollout of Vaccines.” *PloS One* 18 (2): e0277878.  
 636 <https://doi.org/10.1371/journal.pone.0277878>.

## 637 Session information

```
638 R version 4.4.2 (2024-10-31)
639 Platform: x86_64-pc-linux-gnu
640 Running under: Ubuntu 20.04.6 LTS
641
642 Matrix products: default
643 BLAS:    /usr/lib/x86_64-linux-gnublas/libblas.so.3.9.0
644 LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
645
646 locale:
647 [1] LC_CTYPE=fr_FR.UTF-8      LC_NUMERIC=C
648 [3] LC_TIME=fr_FR.UTF-8      LC_COLLATE=fr_FR.UTF-8
649 [5] LC_MONETARY=fr_FR.UTF-8   LC_MESSAGES=fr_FR.UTF-8
650 [7] LC_PAPER=fr_FR.UTF-8     LC_NAME=C
651 [9] LC_ADDRESS=C            LC_TELEPHONE=C
652 [11] LC_MEASUREMENT=fr_FR.UTF-8 LC_IDENTIFICATION=C
653
654 time zone: Europe/Paris
655 tzcode source: system (glibc)
656
657 attached base packages:
658 [1] stats      graphics   grDevices datasets  utils      methods   base
659
660 other attached packages:
661 [1] reservoirnet_0.2.0 patchwork_1.3.0    ggplot2_3.5.1      dplyr_1.1.4
662
663 loaded via a namespace (and not attached):
664 [1] gt_0.11.1        utf8_1.2.4       generics_0.1.3    tidyverse_1.3.1
665 [5] renv_1.0.11      xml2_1.3.6       rstatix_0.7.2     lattice_0.20-45
666 [9] stringi_1.8.4    digest_0.6.37    magrittr_2.0.3    evaluate_1.0.1
667 [13] grid_4.4.2       timechange_0.3.0  fastmap_1.2.0    rprojroot_2.0.4
668 [17] Matrix_1.7-1     jsonlite_1.8.9    slider_0.3.2     backports_1.5.0
669 [21] brio_1.1.5      Formula_1.2-5    purrr_1.0.2      fansi_1.0.6
670 [25] scales_1.3.0     abind_1.4-8      cli_3.6.3       rlang_1.1.4
671 [29] munsell_0.5.1    withr_3.0.2      yaml_2.3.10     tools_4.4.2
672 [33] ggsignif_0.6.4   colorspace_2.1-1  ggpubr_0.6.0    here_1.0.1
673 [37] broom_1.0.7     reticulate_1.40.0 png_0.1-8     vctrs_0.6.5
674 [41] R6_2.5.1        lifecycle_1.0.4   lubridate_1.9.3  snakecase_0.11.1
```

```
675 [45] stringr_1.5.1      car_3.1-3        janitor_2.2.0    warp_0.2.1  
676 [49] pkgconfig_2.0.3    pillar_1.9.0      gtable_0.3.6     Rcpp_1.0.13-1  
677 [53] glue_1.8.0          xfun_0.49       tibble_3.2.1     tidyselect_1.2.1  
678 [57] knitr_1.49         farver_2.1.2     htmltools_0.5.8.1 labeling_0.4.3  
679 [61] rmarkdown_2.29      carData_3.0-5    testthat_3.2.1.1 compiler_4.4.2
```

Submitted