



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  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: 2024-12-09 Last modified: 2024-12-09

Abstract

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

¹Corresponding author:

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	5
5	3.1 Installation	5
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	6
10	3.3.3 Forecast	8
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)	12
15	4 Avanced case-study: Covid-19 hospitalizations forecast	13
16	4.1 Introduction	13
17	4.2 Methods	15
18	4.2.1 Data	15
19	4.2.2 Evaluation framework	15
20	4.2.3 Models	16
21	4.2.4 Hyperparameter optimisation using random search	16
22	4.3 Results	17
23	4.3.1 Hyperparameter selection	17
24	4.3.2 Forecast performance	21
25	4.3.3 Number of model to aggregate	21
26	4.3.4 Input feature importance	21
27	4.4 Discussion	25
28	5 Discussion and conclusion	25
29	References	27

30 1 Introduction

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

40 Originally, the RC paradigm was implemented in artificial firing-rate neurons (“Echo State Networks”,
41 Jaeger (2001)) and spiking neurons (“Liquid State Machine”, Maass, Natschläger, and Markram (2002))
42 as a recurrent neural network (RNN) where the internal recurrent connections, denoted as the
43 reservoir, are randomly generated and only the output layer (named “read-out”) is trained. The
44 reservoir projects temporal input signals onto a high-dimensional feature space, facilitating the
45 learning of non-linear and temporal interactions. Thus, this recurrent layer contains high-dimensional
46 non-linear recombination of the inputs and past states: it is a “reservoir of computations” from
47 which useful information can be linearly extracted (or “read-out”) to provide the desired outputs.
48 This offers the advantage of decreasing the computing time compared to conventional RNNs while
49 consistently maintaining performance (Vlachas et al. 2020). Besides, this RC paradigm fostered
50 increasing interest thanks to its ability to be implemented on classical computers, as the hidden
51 recurrent layer can be kept untrained. A wide range of physical media can be also used to replace
52 it and Tanaka et al. (2019) recently reviewed this prolific field: from FPGA hardware (Penkovsky,
53 Larger, and Brunner 2018), to spin waves using magnetic properties (Nakane, Tanaka, and Hirose
54 2018), skrymions (Prychynenko et al. 2018) or optical implementations (Rafayelyan et al. 2020).
55 This provides interesting and potentially more efficient alternative to traditional machine learning
56 computing.

57 RC leverages various hyperparameters to introduce prior knowledge about the relationship between
58 input variables and output targets. But because the connections within the reservoir are randomly
59 initialized, the same set of hyperparameters may exhibit diverse behaviors across different instances
60 of the reservoir connections. This unpredictability makes it challenging to anticipate the performance
61 of a particular hyperparameter setting, as identical settings may produce varying outcomes when
62 applied to distinct instances of the reservoir. Moreover, selecting the most suitable hyperparameters
63 often requires researchers to experiment with multiple combinations on a training dataset and
64 evaluate their performance on a separate test set². Although this approach can be resource-intensive
65 and time-consuming, it is a compromise that is acceptable considering the rapid simulation capabilities
66 offered by RC. Furthermore, there is a current absence of implementation in R, rendering the method
67 challenging for users unfamiliar with Python (Trouvain and Hinaut 2022) or Julia (Martinuzzi et al.
68 2022).

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

²In 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.

75 exploration includes detailed guidance on the modeling strategy, the selection of hyperparameters,
 76 and the implementation process.

77 2 RC presentation

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

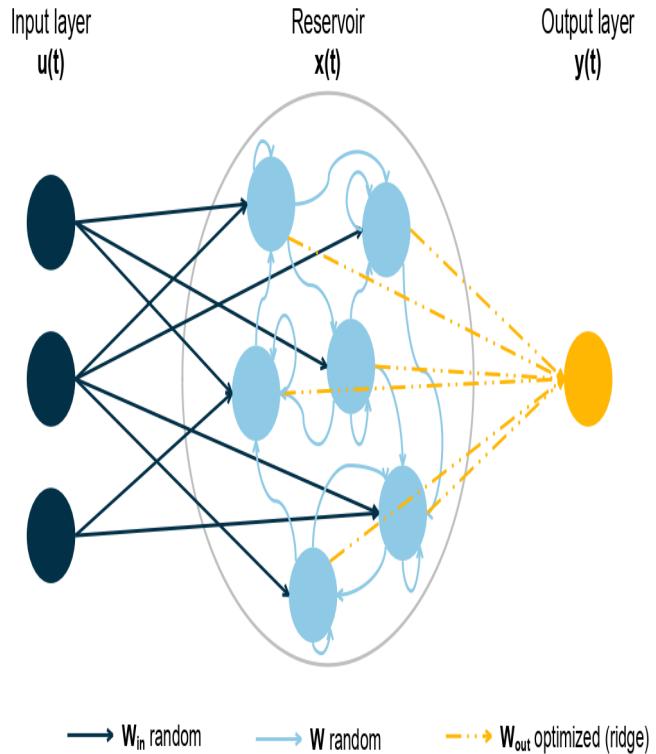


Figure 1: Reservoir computing is composed of an input layer, a reservoir and an output layer. Connection between input layer and reservoir and inside reservoir are random. Only the output layer is optimized based on a ridge regression. Adapted from Trouvain et al. (2020)

87 The input layer $u(t)$ is an M -dimension vector, where M is the number of input time series, which
 88 corresponds to the values of the input time series at time t where $t = 1, \dots, T$. The reservoir layer $x(t)$
 89 is an N_{res} -dimensional vector where N_{res} is the number of nodes in the reservoir. The value $x(t)$ is
 90 defined as follow:

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

91 The leaking rate $\alpha \in [0, 1]$ defines the update rate of the nodes. The closer α is to 1, the more the
 92 reservoir is sensitive to new inputs $u(t)$. Therefore, the reservoir state at time $t + 1$ denoted $x(t + 1)$
 93 depends on the reservoir state at the previous time $x(t)$ and the new inputs $u(t + 1)$. Both W_{in} and W
 94 are random matrices of size $N_{res} \times M$ and $N_{res} \times N_{res}$ respectively.

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

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

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

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

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

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

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

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

115 3 Usage workflow

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

118 3.1 Installation

119 `reservoirnet` is an R package making the Python module `ReservoirPy` easily callable from R using
 120 `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:

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

124 **3.2 Package workflow overview**

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

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

133 **3.3 Basic regression use-case**

134 **3.3.1 Covid-19 data**

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

143 We can proceed by loading useful packages - namely `ggplot2` Wickham, Navarro, and Pedersen
144 (2018) and `dplyr` Wickham et al. (2023), data and define the task:

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

152 **3.3.2 First reservoir**

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

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

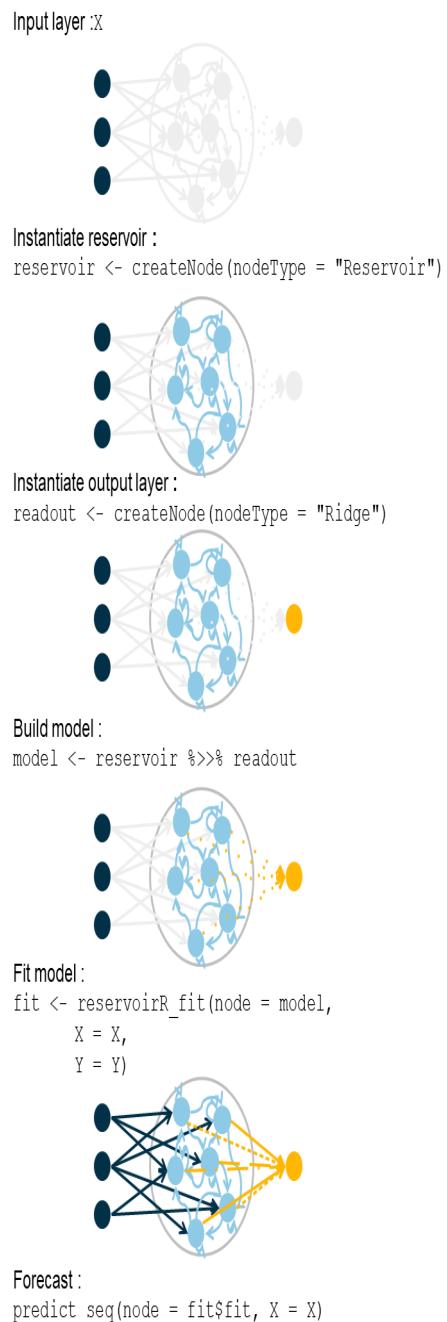


Figure 2: Workflow of `reservoirnet`.

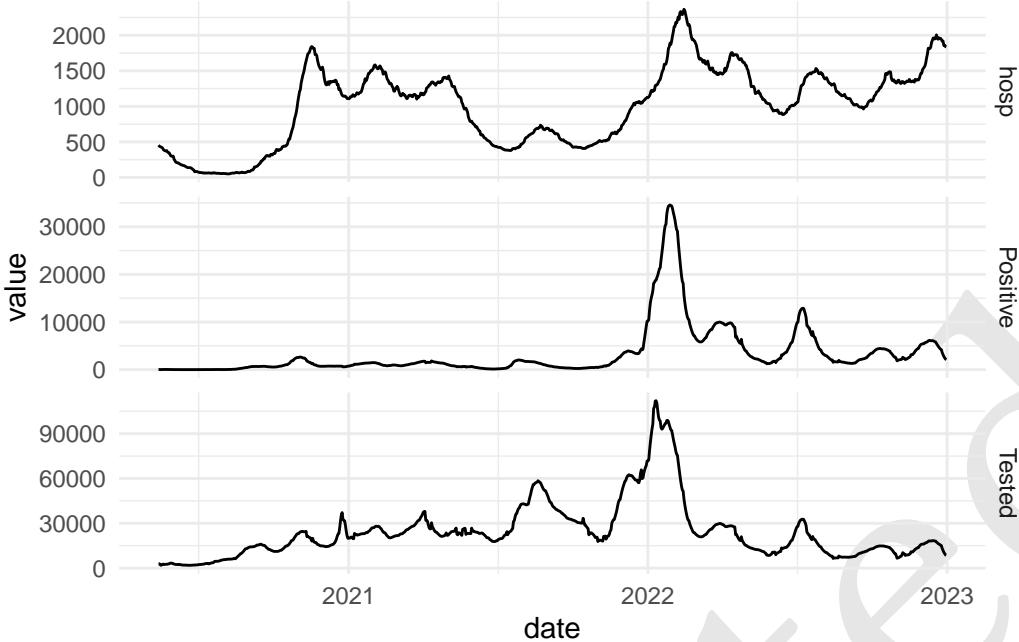


Figure 3: Hospitalizations, IPTCC and positive PCR of Bordeaux University Hospital.

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

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

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

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

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

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

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

185 3.3.3 Forecast

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

188 3.3.3.1 Set the ESN

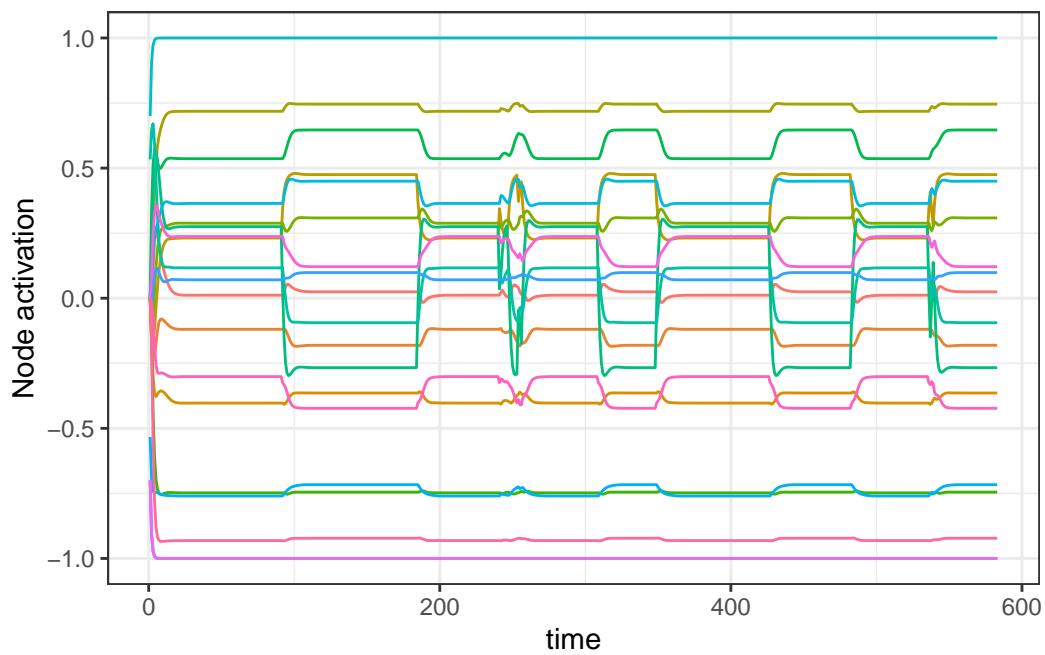


Figure 4: 20 random nodes activation over time.

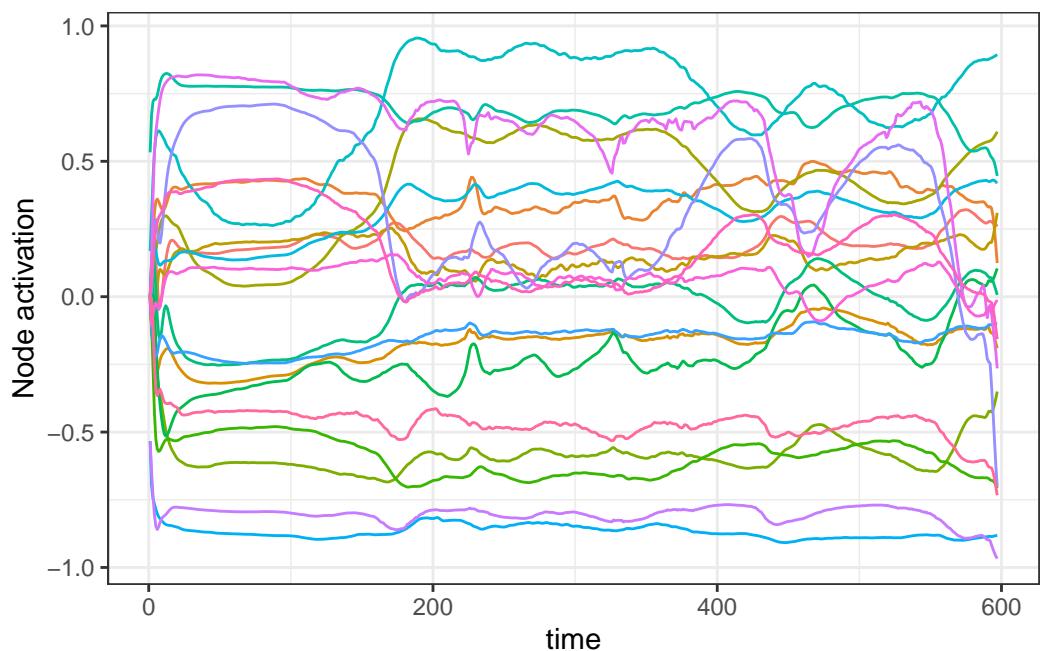


Figure 5: 20 random node activation over time. Scaled features.

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

194 **3.3.3.2 Set the data**

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

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

200 **3.3.3.3 Train the model and predict**

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

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

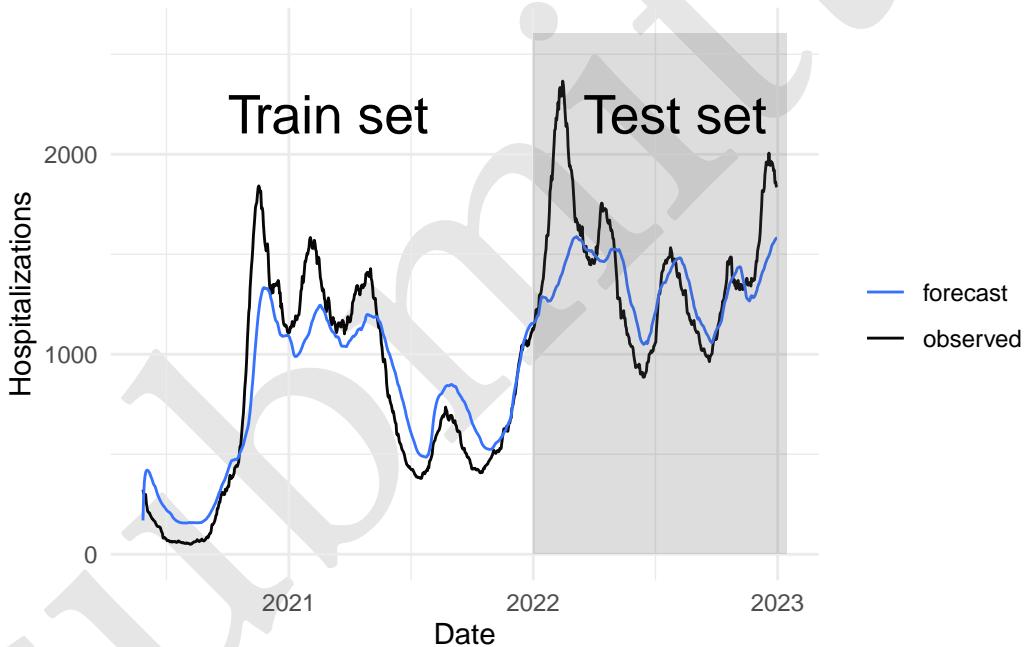


Figure 6: Forecast

205 We observe that the model forecast at Figure 6 is not fully accurate, both on the test set and the
 206 train set. In that case, one option could be to reduce ridge penalization to fit more closely the data,
 207 the optimization of ridge hyperparameter will be discussed at Section 4. Another possibility is to
 208 ease the learning of the algorithm by forecasting the variation of the hospitalization instead of
 209 the number of hospitalized patients. For that step, we will learn on the `outcome_deriv` contained
 210 in `yTrain_variation` data which is defined outcome as `outcome_deriv = outcome - hosp`. As
 211 depicted at Figure 7, this strategy improved the model forecast.

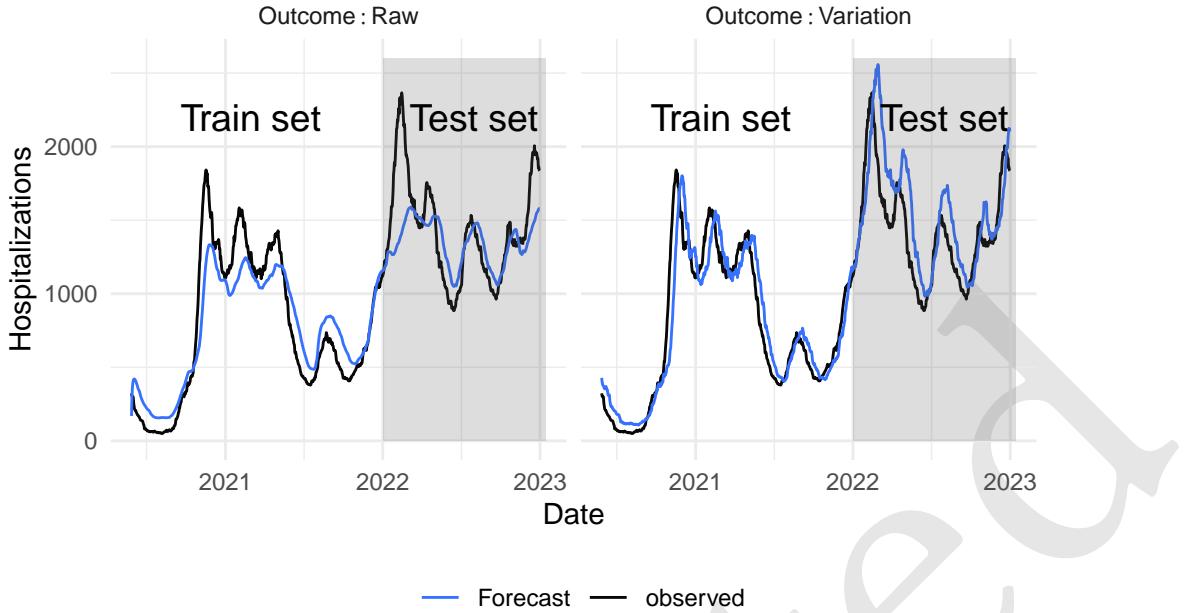


Figure 7: Covid-19 hospitalizations forecast. The model is either trained to forecast the number of hospitalizations (denoted Raw) or the variation of the hospitalizations compared to current level of hospitalisation (denoted Variation)

212 3.4 Classification

213 3.4.1 The Japanese vowel dataset

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

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

222 The primary objective involves the attribution of each utterance to its respective speaker, this is
 223 denoted as classification or sequence-to-vector encoding. The secondary objective involves the
 224 attribution of each time step of each utterance to its speaker, this is denoted as transduction or
 225 sequence-to-sequence encoding.

226 3.4.2 Classification (sequence-to-vector model)

227 The first approach is the sequence-to-vector encoding. For this task we aim to predict the speaker of
 228 the whole utterance (i.e the label is assigned to the whole sequence). We first start by creating the
 229 reservoir and the output layer.

230 To perform this task, we need to modify the training and testing processes. Leveraging the inherent
 231 inertia of the reservoir, information from preceding time steps is preserved, effectively endowing the
 232 RC with a form of memory. Consequently, the final state vector encapsulates insights gathered from
 233 all antecedent states. In the context of the sequence-to-vector encoding task, only this ultimate state

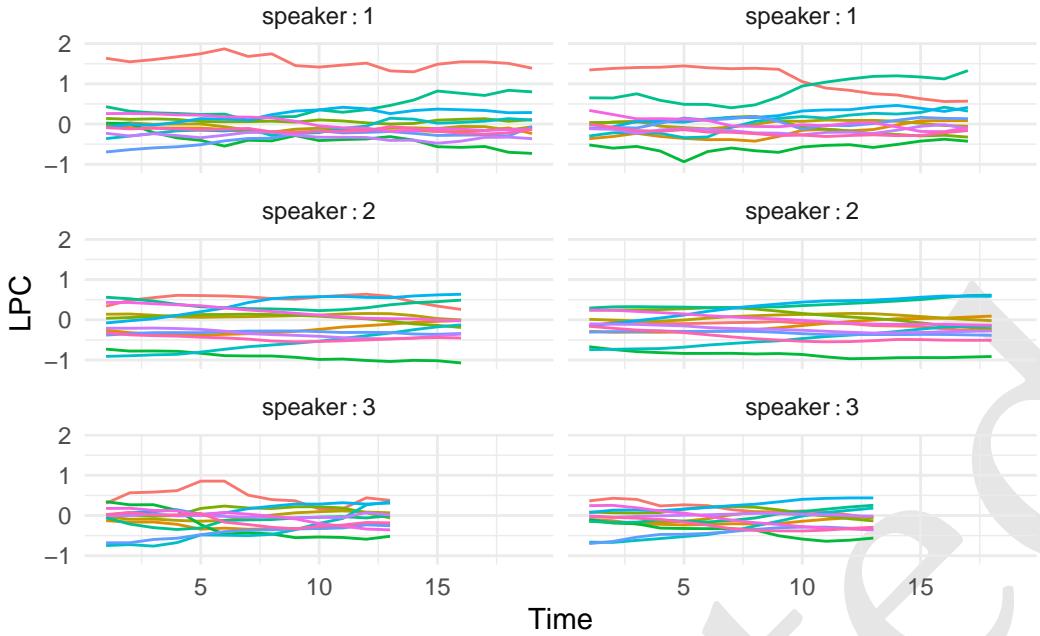


Figure 8: Vowel dataset, sample with 3 speakers and 2 utterance each.

234 is employed. This process is executed as follows:

235 Then we can train the readout based on this last state vector. In that case, `Y_train` contains a single
236 label for each utterance.

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

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

240 Then, we can also compute the overall accuracy :

241 [1] "Accuracy: 92.703%"

242 3.4.3 Transduction (sequence-to-sequence model)

243 For this task, the goal is to predict the speaker for each time step of each utterance. The first
244 step is to get the data where the label is repeated for each time step. This is easily done with the
245 `repeat_targets` argument as follow :

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

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

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

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

255 [1] "Accuracy: 92.456%"

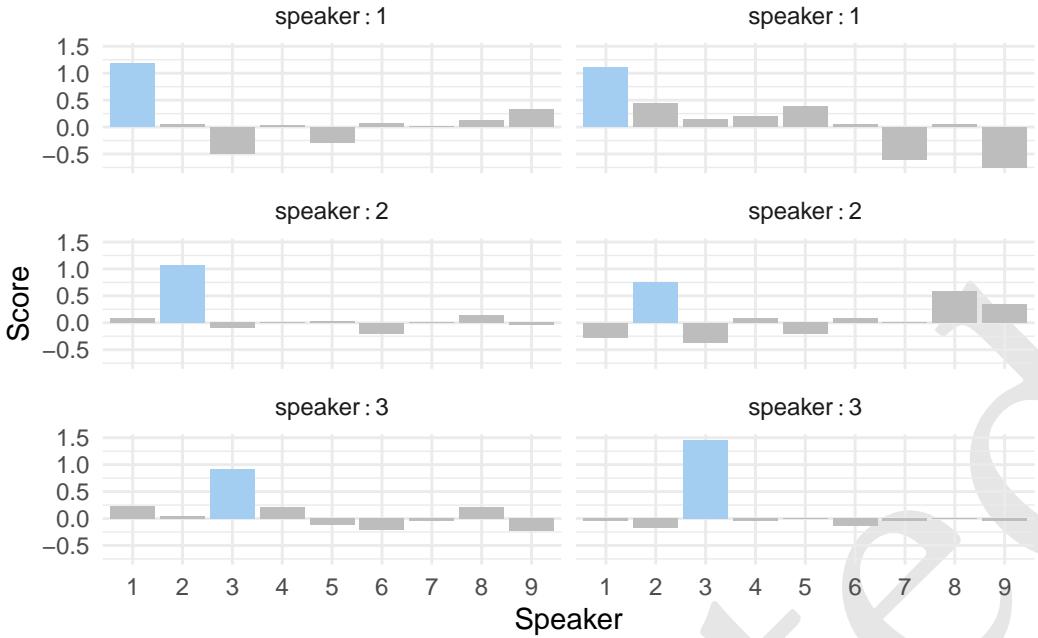


Figure 9: Prediction in a sequence-to-sequence approach 6 samples with 3 speakers and 2 utterance each. The speaker to predict is depicted in blue. For each of the 6 utterance, the model correctly identifies the speaker.

256 4 Avanced case-study: Covid-19 hospitalizations forecast

257 4.1 Introduction

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

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

272 RC has been used in the context of covid-19 epidemic forecast Ghosh et al. (2021). Among them,
 273 Ghosh et al. (2021), Liu et al. (2023) and Ray, Chakraborty, and Ghosh (2021) used it to forecast
 274 epidemic, Zhang et al. (2023) performed sentiment analysis and Kmet and Kmetova (2019) used
 275 it to solve optimal control related to vaccine. The evaluation of RC for epidemic forecast showed
 276 promising results in all approaches, being competitive with Long-Short Term Memory (LSTM) and
 277 Feed-Forward Neural Network (FFNN) in Ray, Chakraborty, and Ghosh (2021). However, the test
 278 period was short for Ghosh et al. (2021) (21 and 14 days) and Ray, Chakraborty, and Ghosh (2021)
 279 (86 days) making it difficult to evaluate the behavior of the methods during epidemic dynamic shift.

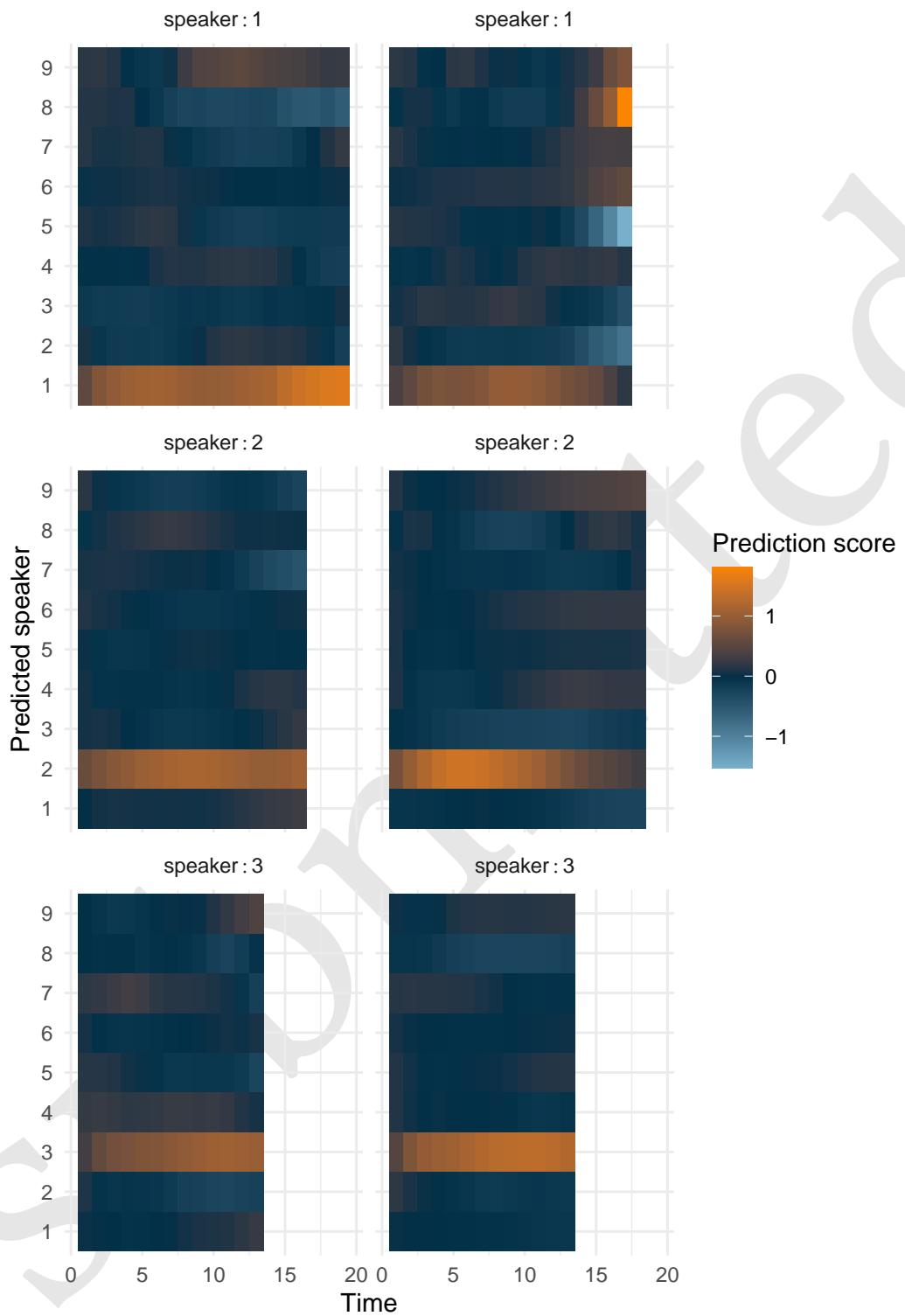


Figure 10: Prediction in a sequence-to-sequence approach 6 samples with 3 speakers and 2 utterance each. The higher the score of the speaker, the lighter the color.

280 This was not the case for Liu et al. (2023) (6 months) but they implemented daily ahead forecast
281 which would be difficult to use to manage a hospital. Finally, all three implementations used only
282 one time series as input whereas it has been shown that using different data sources could improve
283 forecast Ferté et al. (2022). Therefore, it is still difficult to assess the usefulness of RC over a large
284 period and using many time series as inputs.

285 RC can be viewed as an extension of penalized linear regression, where inputs undergo processing by a
286 reservoir, introducing the capacity for memory and non-linear combinations. Given the effectiveness
287 of penalized linear regression in COVID-19 forecasting, as highlighted in Ferté et al. (2022), and the
288 promising results exhibited by RC in epidemic forecasting, as demonstrated in studies such as Ghosh
289 et al. (2021), Liu et al. (2023), and Ray, Chakraborty, and Ghosh (2021), we have opted to employ RC
290 for the prediction of hospitalizations at 14 days at the University Hospital of Bordeaux.

291 The primary aim of this study is to assess the performance of RC in this forecasting task. Secondary
292 objectives include (i) comparing the performance of RC with that of elastic-net penalized regression
293 (identified as the optimal model in Ferté et al. (2022)) and (ii) evaluating variations in RC performance
294 based on different architectural choices, such as the connection between the input layer and the
295 output layer, and the use of one input scaling per feature versus a common input scaling.

296 4.2 Methods

297 4.2.1 Data

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

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

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

319 4.2.2 Evaluation framework

320 The task was to forecast 14 days ahead the number of hospitalized patients. As seen at Section 3.3,
321 we will train the model to predict the variation of hospitalization, denoted as $hosp$, defined as
322 $outcome_{t+14} = hosp_{t+14} - hosp_t$. Metrics computation and visualizations will be performed on the
323 predicted number of hospitalizations denoted as $\widehat{hosp}_{t+14} = \widehat{outcome}_{t+14} + hosp_t$.

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

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

- 327 • the mean absolute error : $MAE = \text{mean}(\widehat{|hosp_{t+14} - hosp_{t+14}|})$.
- 328 • the median relative error : $MRE = \text{median}\left(\frac{|hosp_{t+14} - hosp_{t+14}|}{hosp_{t+14}}\right)$.
- 329 • the mean absolute error to baseline : $MAEB = \text{mean}(\widehat{|hosp_{t+14} - hosp_{t+14}|} - |hosp_t - hosp_{t+14}|)$.
- 330 • the median relative error to baseline : $MREB = \text{median}\left(\frac{|hosp_{t+14} - hosp_{t+14}|}{hosp_t - hosp_{t+14}}\right)$

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

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

341 4.2.3 Models

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

- 347 • Elastic-net penalized regression denoted *Enet*
- 348 • RC with a single input scaling and no connection between input and ouput layers denoted
Common IS R %»% O
- 349 • RC with a single input scaling and connection between input and ouput layers denoted *Common*
IS I+R %»% O
- 350 • RC with multiple input scaling and no connection between input and ouput layers denoted
Multiple IS R %»% O
- 351 • RC with multiple input scaling and connection between input and ouput layers denoted *Multiple*
IS I+R %»% O

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

362 4.2.4 Hyperparameter optimisation using random search

363 RC relies mainly on 4 hyperparameters including the leaking rate (i.e “memory” parameter), spectral
364 radius (i.e “chaoticity” parameter), input scaling (i.e “feature gain” parameter) and ridge (i.e penaliza-
365 tion parameter). Input scaling can be either, common to all features or specific to each feature which

366 increases the number of hyperparameter by the number of features.

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

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

- 373 • (RC) ridge and (Enet) lambda : log-uniform law defined between 1e-10 and 1e5
- 374 • (RC) input scaling and spectral radius : log-uniform law defined between 1e-5 and 1e5
- 375 • (RC) leaking rate : log-uniform law defined between 1e-3 and 1
- 376 • (Enet) alpha : uniform defined between 0 and 1

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

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

384 4.3 Results

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

393 4.3.1 Hyperparameter selection

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

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

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

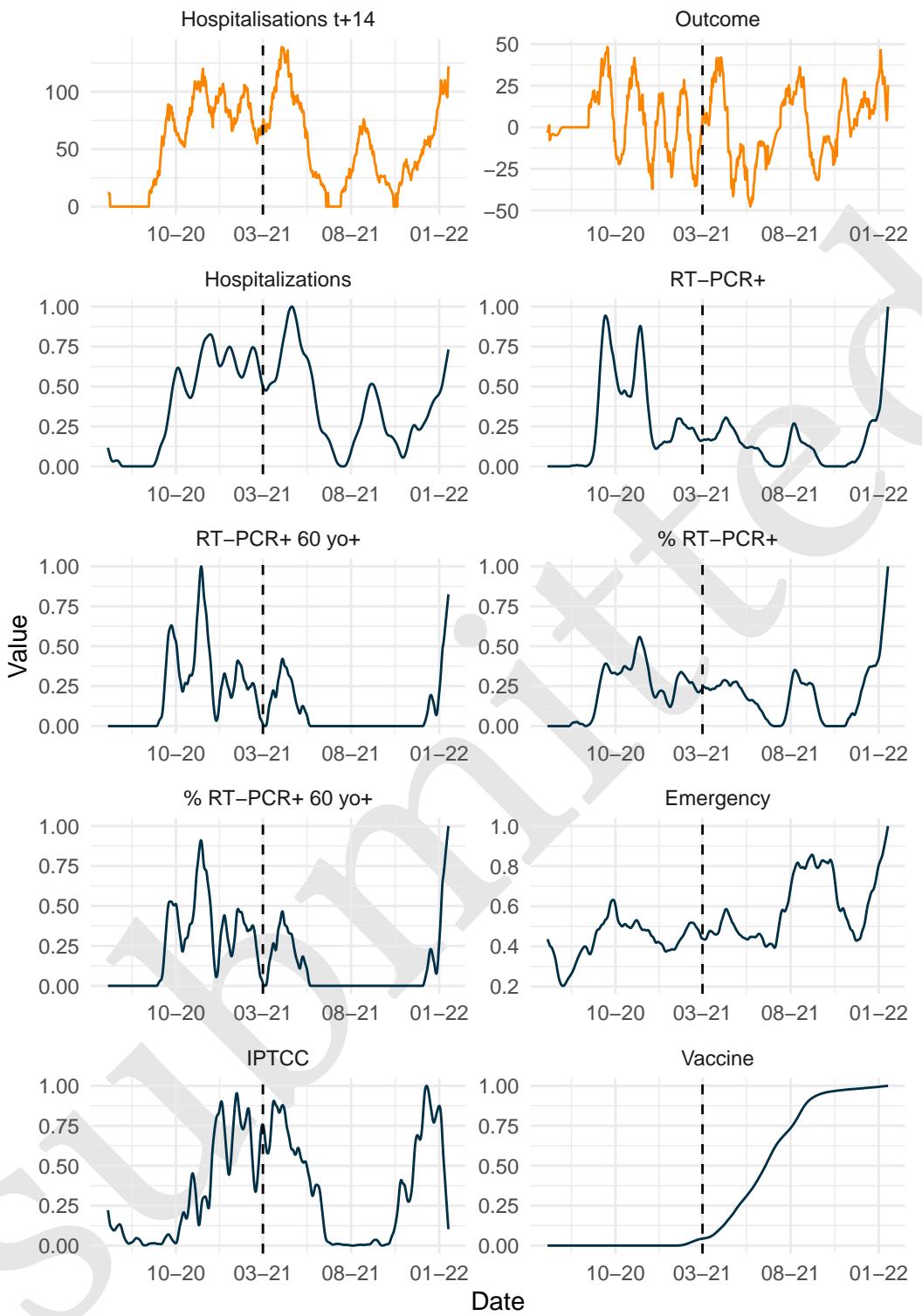


Figure 11: Covid-19 epidemic at BUH. Outcome of interest is presented in orange. Model is trained to forecast Outcome curve which corresponds to the difference between Hospitalisations at 14 days and current hospitalisations. Other features are scaled (divide by the maximum of the feature) represented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

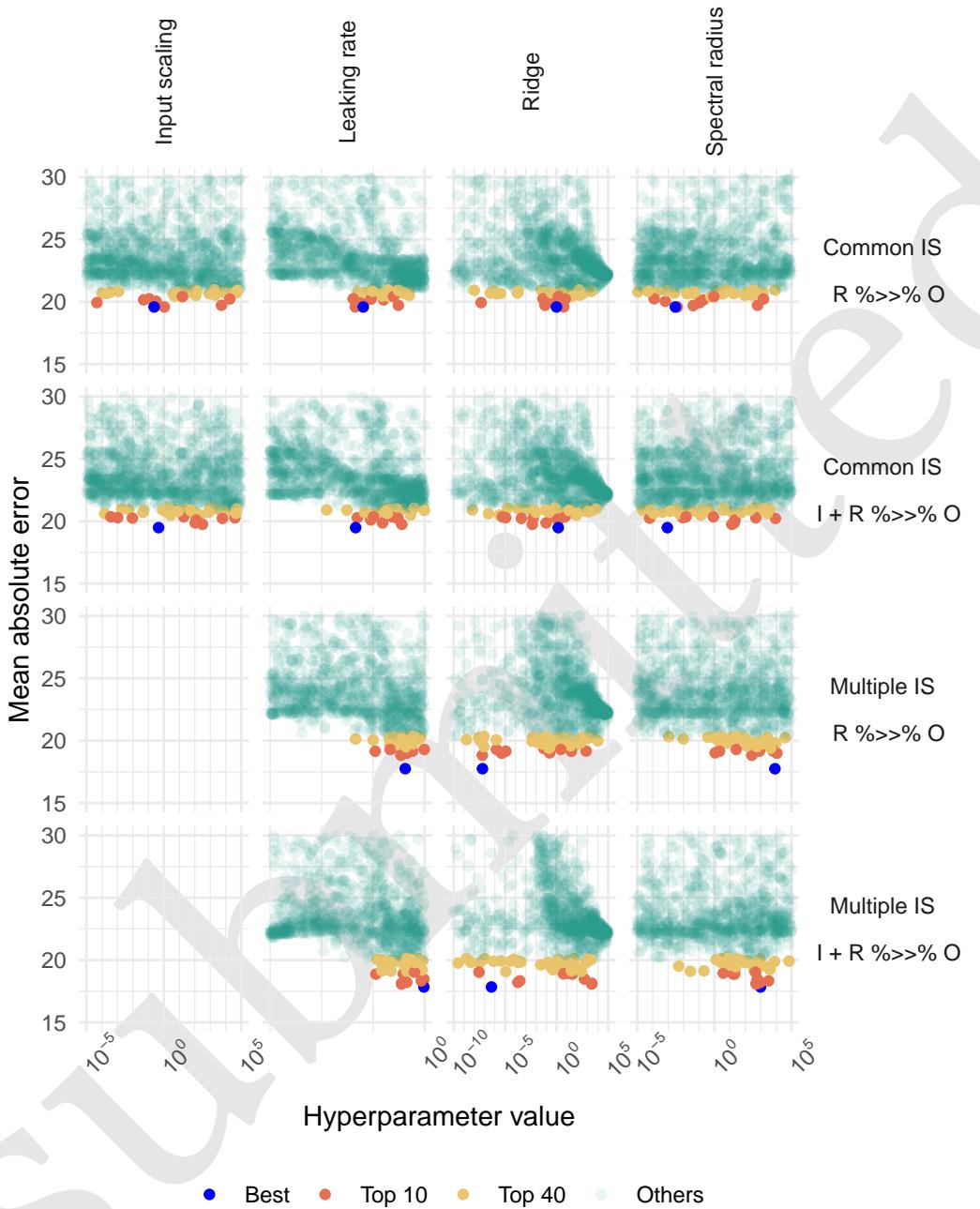


Figure 12: Hyperparameter evaluation on training set by random search. Hp sets with MAE above 30 were removed for clarity of visualisation.

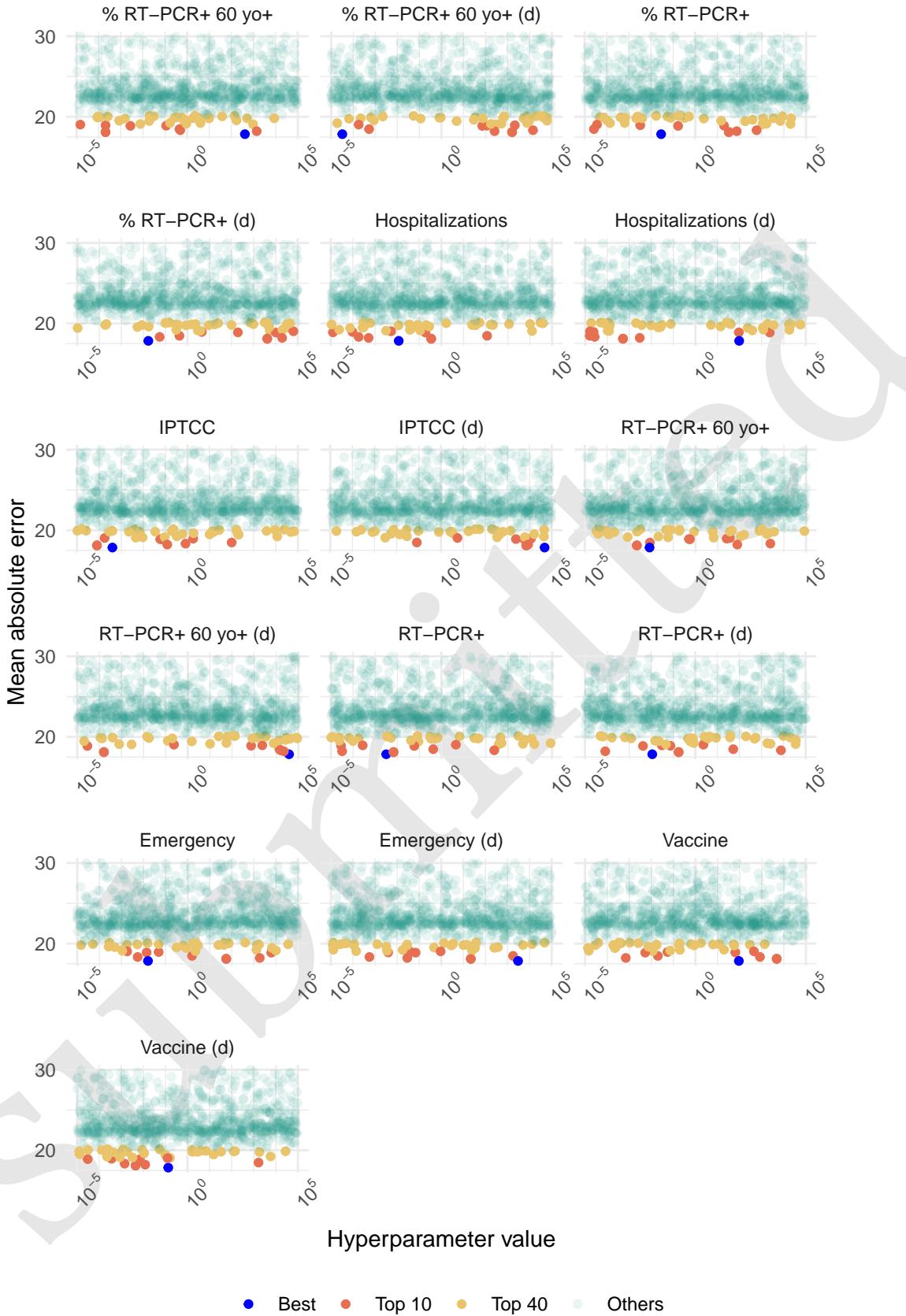


Figure 13: Hyperparameter evaluation on training set by random search of the model with multiple input scaling and no connection between input layer and output layer. Hp sets with MAE above 30 were removed for clarity of visualisation.

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

406 **4.3.2 Forecast performance**

407 Table 1 shows the performance on the test set. Best model according to all metrics was RC with
 408 common input scaling and connection between input and output layers. Having one input scaling per
 409 feature did not improve the model which might be due to low generalisability of the hyperparameter
 410 of the training set to the test set due to non-stationarity. Additionaly, connecting input layer to
 411 output layer improved the model forecast. All RC models performed better than the elastic-net
 412 model.

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

417 **4.3.3 Number of model to aggregate**

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

423 **4.3.4 Input feature importance**

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

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

433 Furthermore, Figure 18 compares the coefficients assigned to input features by the elastic-net model
 434 and the RC models. While the coefficients are generally consistent across RC models, there are
 435 some notable differences with elastic-net. Specifically, certain features deemed important by the

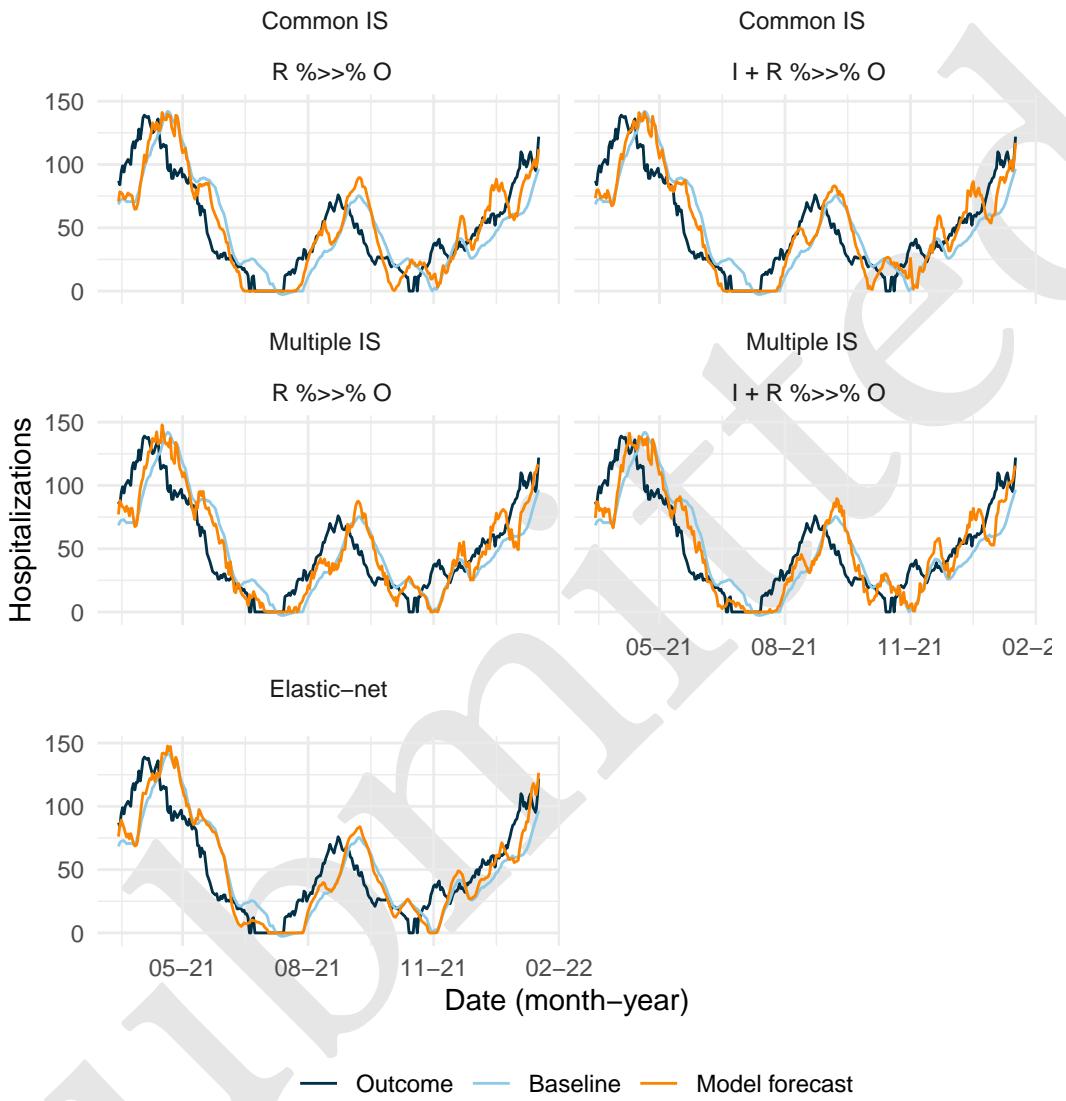


Figure 14: Reservoir computing forecast depending on the setting with and without monthly update. Red line is the median forecast of 40 reservoirs. Grey lines are individual forecast of each of the 40 reservoirs.

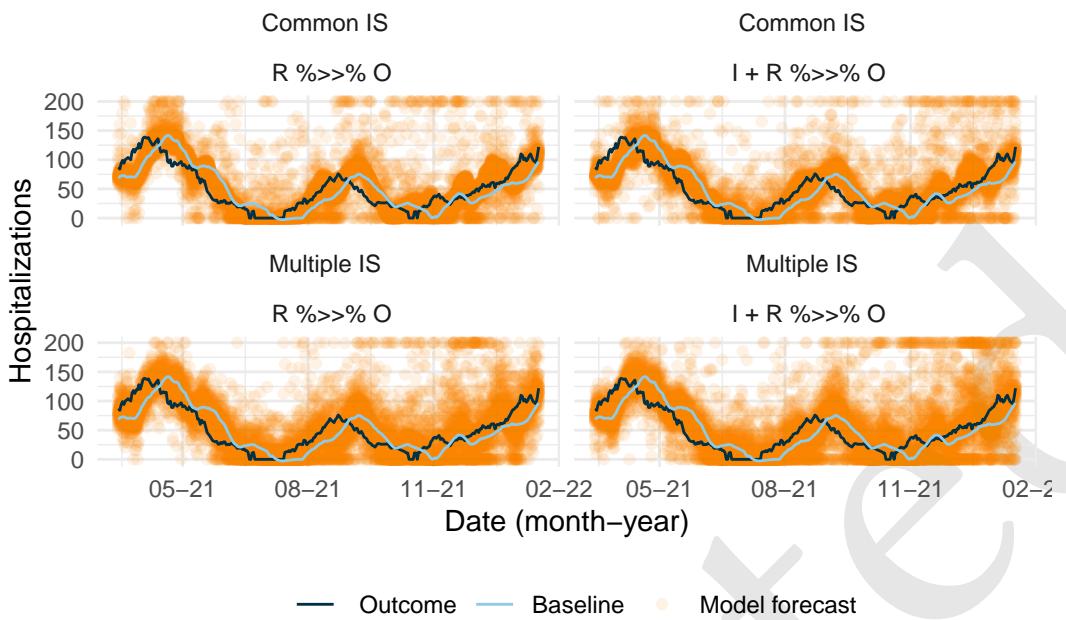


Figure 15: Individual forecast the 40 best hyperparameter sets for the different RC configuration. Forecast value above 200 were set to 200 for clarity.

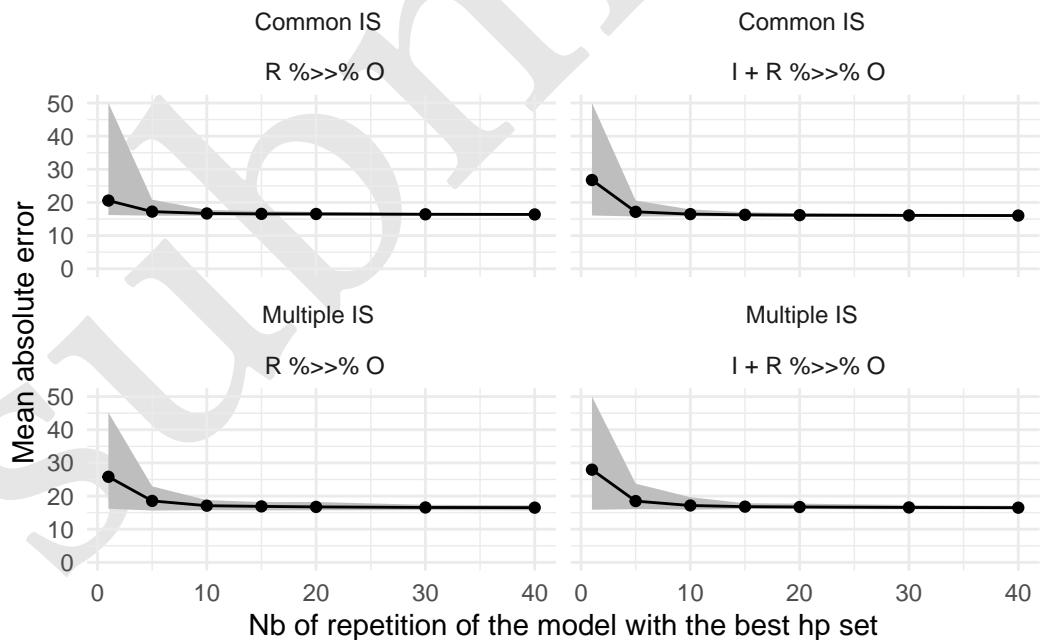


Figure 16: Mean absolute error depending on the number of aggregated reservoir.

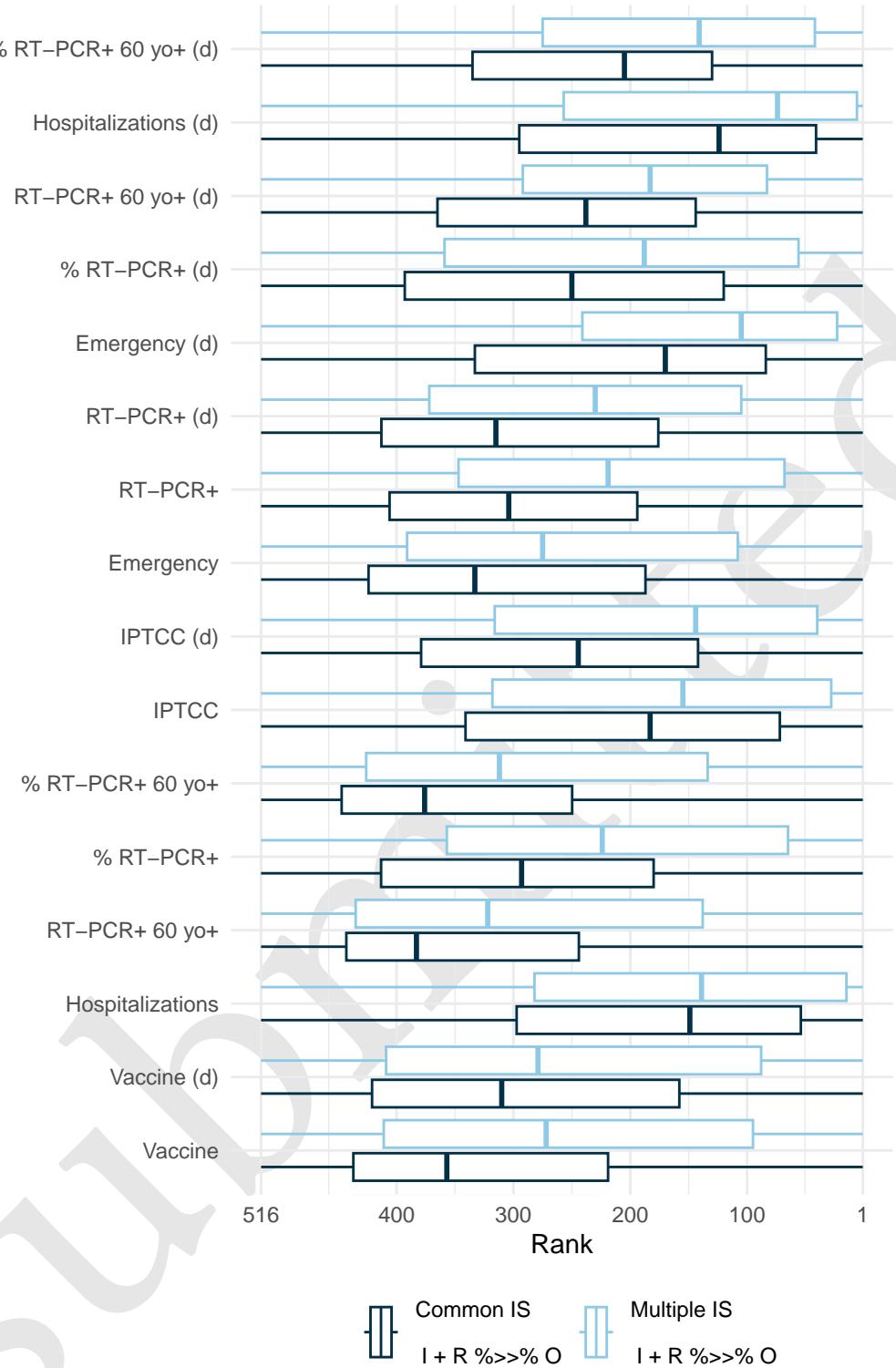


Figure 17: Mean feature importance of the 40 best hyperparameter sets by model, focus on the connection between the input and output layers. Models with direct connection between input and output layer are included. The rank is obtained by comparing the feature input layer and all other connection coefficients (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.

436 elastic-net model, such as the derivative of RT-PCR, and the derivative of Vaccine, are less important
437 for the reservoir computing model. This may indicate that these features predictive ability is better
438 conveyed by their relationship with other features, which is captured by the reservoir computing
439 model but might not be by the elastic-net model. Conversely, emergency, IPTCC, proportion of
440 positive RT-PCR, and hospitalizations are more important for the reservoir computing model than
441 for the elastic-net model.

442 **4.4 Discussion**

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

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

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

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

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

466 **5 Discussion and conclusion**

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

471 We provided a comprehensive overview of the basic usage of the `reservoirnet` package through
472 illustrative examples in regression and classification tasks. This introductory section serves as
473 a foundation for R users, offering step-by-step guidance on constructing and training reservoir
474 computing models using the package. By demonstrating the application of RC in both regression
475 and classification scenarios, we aim to equip users with the essential knowledge and skills needed to
476 harness the capabilities of reservoir computing for diverse tasks.

477 Drawing on the robust foundation of the ReservoirPy structure, a well-maintained Python library,
478 this package inherits its reliability and longevity. We have focused on providing access to the

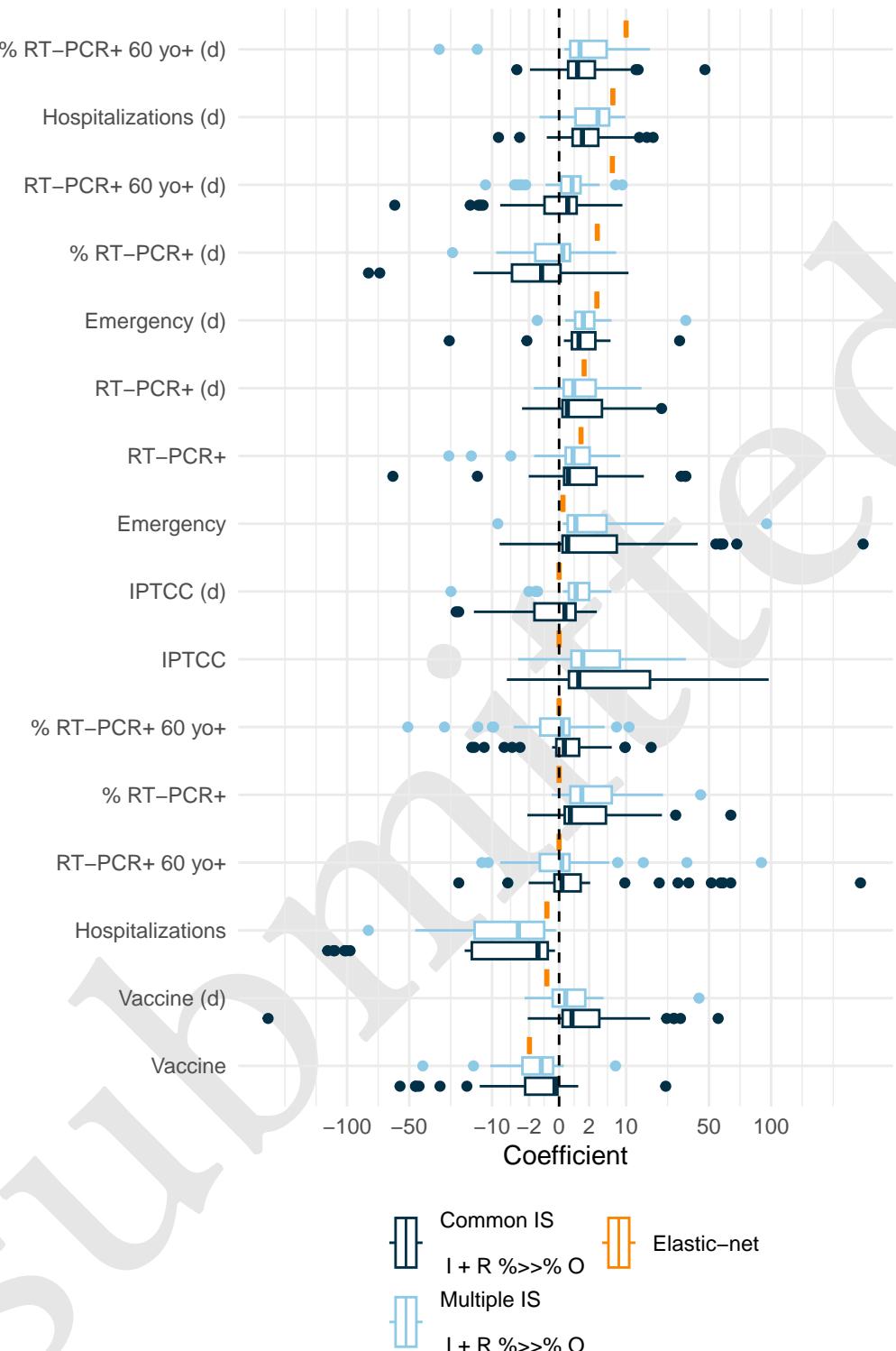


Figure 18: Mean feature coefficient of the 40 best hyperparameter sets by model and the elastic-net model. Only models with direct connection between input and output layer are included. The coefficients were calculated as the average value across all dates for each feature, model and hyperparameter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

479 fundamental features, building upon the strong base provided by ReservoirPy. Therefore, this initial
480 version of reservoirnet must evolve in tandem with the growing understanding and adoption of
481 RC within the R community.

482 References

- 483 Bala, Abubakar, Idris Ismail, Rosdiazli Ibrahim, and Sadiq M. Sait. 2018. “Applications of Meta-
484 heuristics in Reservoir Computing Techniques: A Review.” *IEEE Access* 6: 58012–29. <https://doi.org/10.1109/ACCESS.2018.2873770>.
- 485 Carrat, Fabrice, Julie Figoni, Joseph Henny, Jean-Claude Desenclos, Sofiane Kab, Xavier de Lamballerie,
486 and Marie Zins. 2021. “Evidence of Early Circulation of SARS-CoV-2 in France: Findings from
487 the Population-Based ‘CONSTANCES’ Cohort.” *European Journal of Epidemiology*, February, 1–4.
488 <https://doi.org/10.1007/s10654-020-00716-2>.
- 489 Carslaw, David. 2023. “Worldmet: Import Surface Meteorological Data from NOAA Integrated
490 Surface Database (ISD).” <https://cran.r-project.org/web/packages/worldmet/index.html>.
- 491 Carvalho, Kathleen, João Paulo Vicente, Mihajlo Jakovljevic, and João Paulo Ramos Teixeira. 2021.
492 “Analysis and Forecasting Incidence, Intensive Care Unit Admissions, and Projected Mortality
493 Attributable to COVID-19 in Portugal, the UK, Germany, Italy, and France: Predictions for 4
494 Weeks Ahead.” *Bioengineering* 8 (6): 84. <https://doi.org/10.3390/bioengineering8060084>.
- 495 COVID-19 Cumulative Infection Collaborators. 2022. “Estimating Global, Regional, and National
496 Daily and Cumulative Infections with SARS-CoV-2 Through Nov 14, 2021: A Statistical Analysis.”
497 *Lancet*. [https://doi.org/10.1016/S0140-6736\(22\)00484-6](https://doi.org/10.1016/S0140-6736(22)00484-6).
- 498 Cramer, Estee Y., Evan L. Ray, Velma K. Lopez, Johannes Bracher, Andrea Brennen, and et al.
499 2022. “Evaluation of Individual and Ensemble Probabilistic Forecasts of COVID-19 Mortality
500 in the United States.” *Proceedings of the National Academy of Sciences* 119 (15): e2113561119.
501 <https://doi.org/10.1073/pnas.2113561119>.
- 502 Etalab. 2020. “Les Données Relatives Au COVID-19 En France - Data.gouv.fr.” <https://www.data.gouv.fr/fr/pages/donnees-coronavirus/>.
- 503 Ferté, Thomas, Kalidou Ba, Dan Dutartre, Pierrick Legrand, Vianney Jouhet, Romain Griffier, Rodolphe
504 Thiébaut, Xavier Hinaut, and Boris P. Hejblum. 2024. “Thomasferte/Jss_reservoirnet: First
505 Release.” Zenodo. <https://doi.org/10.5281/ZENODO.11281341>.
- 506 Ferté, Thomas, Vianney Jouhet, Romain Griffier, Boris P. Hejblum, Rodolphe Thiébaut, and Bordeaux
507 University Hospital Covid-19 Crisis Task Force. 2022. “The Benefit of Augmenting Open Data
508 with Clinical Data-Warehouse EHR for Forecasting SARS-CoV-2 Hospitalizations in Bordeaux
509 Area, France.” *JAMIA Open* 5 (4): ooac086. <https://doi.org/10.1093/jamiaopen/ooac086>.
- 510 Ferté, Thomas, Vianney Jouhet, Romain Griffier, Boris Hejblum, Rodolphe Thiébaut, and Bordeaux
511 University Hospital Covid-19 Crisis Task Force. 2023. “The Benefit of Augmenting Open Data
512 with Clinical Data-Warehouse EHR for Forecasting SARS-CoV-2 Hospitalizations in Bordeaux
513 Area, France.” Dryad. <https://doi.org/10.5061/DRYAD.HHMGQNKX>.
- 514 Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. 2010. “Regularization Paths for Generalized
515 Linear Models via Coordinate Descent.” *Journal of Statistical Software* 33 (1): 1–22.
- 516 Ghosh, Subrata, Abhishek Senapati, Arindam Mishra, Joydev Chattopadhyay, Syamal K. Dana,
517 Chittaranjan Hens, and Dibakar Ghosh. 2021. “Reservoir Computing on Epidemic Spreading:
518 A Case Study on COVID-19 Cases.” *Physical Review E* 104 (1): 014308. <https://doi.org/10.1103/PhysRevE.104.014308>.
- 519 Hinaut, Xavier, and Peter Ford Dominey. 2013. “Real-Time Parallel Processing of Grammatical
520 Structure in the Fronto-Striatal System: A Recurrent Network Simulation Study Using Reservoir
521 Computing.” *PLOS ONE* 8 (2): e52946. <https://doi.org/10.1371/journal.pone.0052946>.
- 522 Hübner, Martin, Tobias Zingg, David Martin, Philippe Eckert, and Nicolas Demartines. 2020. “Surgery
523 for Non-Covid-19 Patients During the Pandemic.” *PLoS ONE* 15 (10): e0241331. <https://doi.org/10.5281/ZENODO.11281341>.
- 524
- 525
- 526

- 527 1371/journal.pone.0241331.
- 528 Jaeger, Herbert. 2001. "The" Echo State" Approach to Analysing and Training Recurrent Neu-
529 ral Networks-with an Erratum Note'." *Bonn, Germany: German National Research Center for*
530 *Information Technology GMD Technical Report* 148 (January).
- 531 Kim, Gina, Mengru Wang, Hanh Pan, Giana H. Davidson, Alison C. Roxby, Jen Neukirch, Danna Lei,
532 Elicia Hawken-Dennis, Louise Simpson, and Thuan D. Ong. 2020. "A Health System Response to
533 COVID-19 in Long-Term Care and Post-Acute Care: A Three-Phase Approach." *Journal of the*
534 *American Geriatrics Society* 68 (6): 1155–61. <https://doi.org/10.1111/jgs.16513>.
- 535 Kmet, Tibor, and Maria Kmetova. 2019. "Bézier Curve Parametrisation and Echo State Network
536 Methods for Solving Optimal Control Problems of SIR Model." *Biosystems* 186 (December): 104029.
537 <https://doi.org/10.1016/j.biosystems.2019.104029>.
- 538 Kudo, Mineichi, Jun Toyama, and Masaru Shimbo. 1999. "Multidimensional Curve Classification
539 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).
- 540 Liu, Bocheng, Yiyuan Xie, Weichen Liu, Xiao Jiang, Yichen Ye, Tingting Song, Junxiong Chai, Manying
541 Feng, and Haodong Yuan. 2023. "Nanophotonic Reservoir Computing for COVID-19 Pandemic
542 Forecasting." *Nonlinear Dynamics* 111 (7): 6895–6914. <https://doi.org/10.1007/s11071-022-08190-z>.
- 543 Lukoševičius, Mantas, and Herbert Jaeger. 2009. "Reservoir Computing Approaches to Recurrent
544 Neural Network Training." *Computer Science Review* 3 (3): 127–49. <https://doi.org/10.1016/j.cosrev.2009.03.005>.
- 545 Maass, Wolfgang, Thomas Natschläger, and Henry Markram. 2002. "Real-Time Computing Without
546 Stable States: A New Framework for Neural Computation Based on Perturbations." *Neural
547 Computation* 14 (11): 2531–60. <https://doi.org/10.1162/089976602760407955>.
- 548 Martinuzzi, Francesco, Chris Rackauckas, Anas Abdelrehim, Miguel D. Mahecha, and Karin Mora.
549 2022. "ReservoirComputing.jl: An Efficient and Modular Library for Reservoir Computing
550 Models." *Journal of Machine Learning Research* 23 (288): 1–8. <http://jmlr.org/papers/v23/22-0611.html>.
- 551 Mohimont, Lucas, Amine Chemchem, François Alin, Michaël Krajecki, and Luiz Angelo Steffenel.
552 2021. "Convolutional Neural Networks and Temporal CNNs for COVID-19 Forecasting in France."
553 *Applied Intelligence*, April. <https://doi.org/10.1007/s10489-021-02359-6>.
- 554 Nakane, Ryosho, Gouhei Tanaka, and Akira Hirose. 2018. "Reservoir Computing With Spin Waves
555 Excited in a Garnet Film." *IEEE Access* PP (January): 1–1. <https://doi.org/10.1109/ACCESS.2018.2794584>.
- 556 Paireau, Juliette, Alessio Andronico, Nathanaël Hozé, Maylis Layan, Pascal Crépey, Alix Roumagnac,
557 Marc Lavielle, Pierre-Yves Boëlle, and Simon Cauchemez. 2022. "An Ensemble Model Based
558 on Early Predictors to Forecast COVID-19 Health Care Demand in France." *Proceedings of the
559 National Academy of Sciences* 119 (18): e2103302119. <https://doi.org/10.1073/pnas.2103302119>.
- 560 Penkovsky, Bogdan, Laurent Larger, and Daniel Brunner. 2018. "Efficient Design of Hardware-
561 Enabled Reservoir Computing in FPGAs." *Journal of Applied Physics* 124 (16): 162101. <https://doi.org/10.1063/1.5039826>.
- 562 Pottier, Loïc. 2021. "Forecast of the Covid19 Epidemic in France." *medRxiv*. <https://doi.org/10.1101/2021.04.13.21255418>.
- 563 Prychynenko, Diana, Matthias Sitte, Kai Litzius, Benjamin Krüger, George Bourianoff, Mathias Kläui,
564 Jairo Sinova, and Karin Everschor-Sitte. 2018. "Magnetic Skyrmion as a Nonlinear Resistive
565 Element: A Potential Building Block for Reservoir Computing." *Physical Review Applied* 9 (1):
566 014034. <https://doi.org/10.1103/PhysRevApplied.9.014034>.
- 567 Rafayelyan, Mushegh, Jonathan Dong, Yongqi Tan, Florent Krzakala, and Sylvain Gigan. 2020. "Large-
568 Scale Optical Reservoir Computing for Spatiotemporal Chaotic Systems Prediction." *Physical
569 Review X* 10 (4): 041037. <https://doi.org/10.1103/PhysRevX.10.041037>.
- 570 Rahimi, Iman, Fang Chen, and Amir H. Gandomi. 2021. "A Review on COVID-19 Forecasting Models."

- 577 *Neural Computing & Applications*, February, 1–11. <https://doi.org/10.1007/s00521-020-05626-8>.
- 578 Ray, Arnob, Tanujit Chakraborty, and Dibakar Ghosh. 2021. “Optimized Ensemble Deep Learning
579 Framework for Scalable Forecasting of Dynamics Containing Extreme Events.” *Chaos (Woodbury,
580 N.Y.)* 31 (11): 111105. <https://doi.org/10.1063/5.0074213>.
- 581 Roumagnac, Alix, Eurico de Carvalho Filho, Raphaël Bertrand, Anne-Kim Banchereau, and Guillaume
582 Lahache. 2021. “Étude de l'influence Potentielle de l'humidité Et de La Température Dans La
583 Propagation de La Pandémie COVID-19.” *Médecine de Catastrophe - Urgences Collectives*, Douleur
584 et situations d'exceptionPandémie COVID-19, 5 (1): 87–102. <https://doi.org/10.1016/j.pxur.2021.01.002>.
- 585 Simões, Jorge, João Paulo Moreira Magalhães, André Biscaia, António da Luz Pereira, Gonçalo
586 Figueiredo Augusto, and Inês Fronteira. 2021. “Organisation of the State, Model of Health System
587 and COVID-19 Health Outcomes in Six European Countries, During the First Months of the
588 COVID-19 Epidemic in 2020.” *The International Journal of Health Planning and Management*, June,
589 10.1002/hpm.3271. <https://doi.org/10.1002/hpm.3271>.
- 590 Smith, Adam, Neal Lott, and Russ Vose. 2011. “The Integrated Surface Database: Recent Developments
591 and Partnerships.” *Bulletin of the American Meteorological Society* 92 (6): 704–8. <https://doi.org/10.1175/2011BAMS3015.1>.
- 592 Tanaka, Gouhei, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji
593 Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. 2019. “Recent Advances in Physical
594 Reservoir Computing: A Review.” *Neural Networks* 115 (July): 100–123. <https://doi.org/10.1016/j.neunet.2019.03.005>.
- 595 Trouvain, Nathan, and Xavier Hinaut. 2021. “Canary Song Decoder: Transduction and Implicit
596 Segmentation with ESNs and LTSMs.” In *Artificial Neural Networks and Machine Learning – ICANN*
597 2021, edited by Igor Farkaš, Paolo Masulli, Sebastian Otte, and Stefan Wermter, 71–82. Lecture
598 Notes in Computer Science. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-86383-8_6.
- 599 ———. 2022. “Reservoirpy: A Simple and Flexible Reservoir Computing Tool in Python.” <https://inria.hal.science/hal-03699931>.
- 600 Trouvain, Nathan, Luca Pedrelli, Thanh Trung Dinh, and Xavier Hinaut. 2020. “ReservoirPy:
601 An Efficient and User-Friendly Library to Design Echo State Networks.” In *Artificial Neural
602 Networks and Machine Learning – ICANN 2020*, 494–505. Springer International Publishing.
603 <https://inria.hal.science/hal-02595026>.
- 604 Trouvain, Nathan, Nicolas Rougier, and Xavier Hinaut. 2022. “Create Efficient and Complex Reservoir
605 Computing Architectures with ReservoirPy.” In *From Animals to Animats 16*, edited by Lola
606 Cañamero, Philippe Gaussier, Myra Wilson, Sofiane Boucenna, and Nicolas Cuperlier, 91–102.
607 Lecture Notes in Computer Science. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-16770-6_8.
- 608 Ushey, Kevin, JJ Allaire, and Yuan Tang. 2024. *Reticulate: Interface to 'Python'*. <https://rstudio.github.io/reticulate/>.
- 609 Vlachas, P. R., J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos. 2020.
610 “Backpropagation Algorithms and Reservoir Computing in Recurrent Neural Networks for the
611 Forecasting of Complex Spatiotemporal Dynamics.” *Neural Networks* 126 (June): 191–217. <https://doi.org/10.1016/j.neunet.2020.02.016>.
- 612 Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. “dplyr: A
613 Grammar of Data Manipulation.” <https://CRAN.R-project.org/package=dplyr>.
- 614 Wickham, Hadley, Danielle Navarro, and Thomas Lin Pedersen. 2018. *ggplot2: Elegant Graphics for
615 Data Analysis (3e)*. 3rd ed. Springer-Verlag New York. <https://ggplot2-book.org/>.
- 616 World Health Organisation. 2020. “WHO Coronavirus (COVID-19) Dashboard.” <https://covid19.who.int>.
- 617 Zhang, Qihuang, Grace Y. Yi, Li-Pang Chen, and Wenqing He. 2023. “Sentiment Analysis and

627 Causal Learning of COVID-19 Tweets Prior to the Rollout of Vaccines.” *PloS One* 18 (2): e0277878.
628 <https://doi.org/10.1371/journal.pone.0277878>.

Submitted