TFM-LUCIA

TFM Máster "Visual Analytics and Big Data"

Contenido

Introducción

Este proyecto consiste en la evaluación de la calidad del aire en la ciudad de Madrid. Para ello, el proyecto está dividido en cuatro apartados claramente diferenciados:

- **Captura, almacenamiento y procesado de datos** relacionados con la calidad del aire, como son por ejemplo la concentración de contaminantes, las condiciones meteorológicas o diferentes estadísticos socio-demográficos.
- **Análisis descriptivo** de la información recogida, con el objetivo de presentar adecuadamente la evolución de la calidad del aire a lo largo de los últimos años.

- **Modelado predictivo** de la calidad del aire, lo que permite anticipar su evolución en un futuro a corto-medio plazo. El modelado predictivo será la base para la toma de decisiones que permitan mantener unos niveles de calidad del aire aceptables.
- **Representación de la información** mediante una aplicación web interactiva. Los resultados del proyecto serán accesibles de manera online y gratuita a través de una URL.

Captura, almacenamiento y procesado de los datos

Calidad del aire

El Ayuntamiento de Madrid dispone de un portal web de datos abiertos (*open data*) en donde comparte un amplio catálogo de datos, así como una API para su descarga. Este catálogo de datos se puede consultar en el siguiente enlace: https://datos.madrid.es/portal/site/egob/menuitem.9e1e2f6404558187cf35cf3584f1a5a0/?vgnextoid=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default

Para la realización de este proyecto se han utilizado los datos ofrecidos por el Ayuntamiento de Madrid en relación a la calidad del aire. A continuación se describen sus principales características:

Estaciones de control

El Sistema de Vigilancia del Ayto. de Madrid está formado por 24 estaciones remotas automáticas que recogen la información básica para la vigilancia atmosférica. Poseen los analizadores necesarios para la medida correcta de los niveles de gases y de partículas. Las estaciones remotas son de varios tipos:

- **Urbanas de fondo (UF)**: Representativas de la exposición de la población urbana en general.
- **Urbanas de tráfico (UT)**: Situadas de tal manera que su nivel de contaminación está influido principalmente por las emisiones procedentes de una calle o carretera próxima, pero se ha de evitar que se midan microambientes muy pequeños en sus proximidades.
- **Suburbanas (S)**: Están situadas a las afueras de la ciudad, en los lugares donde se encuentran los mayores niveles de ozono.

De cada una de las 24 estaciones de control, el Ayto. de Madrid ofrece la siguiente información:

- Identificador numérico
- Nombre de la estación
- Dirección postal
- Coordenadas latitud y longitud
- Altitud
- Tipo de estación (UF/UT/S)
- Selección de los contaminante medidos
- Selección de los sensores meteorológicos instalados

SITUACIÓN DE LAS ESTACIONES DE LA RED DE VIGILANCIA DE LA CALIDAD DEL AIRE							CONTAMINANTE MEDIDO								SENSORES METEOROLÓGICOS							
NÚMERO	ESTACIÓN	DIRECCIÓN	LONGITUD	LATITUD	ALTITUD	TIPO ESTACION *	NO2	SO2	CO	PM10	PM2,5	O3	BTX	HC	UV	VV	DV	TMP	HR	PRB	RS	LL
4	Pza. de España	Plaza de España	3º 42' 44,09"O	40º 25' 25,87"N	635	UT	x	x	x							x	x	x	x			x
8	Escuelas Aguirre	Entre C/ Alcalá y C/ O' Donell	3º 40' 56,35"O	40º 25' 17,63"N	670	UT	x	x	x	x	x	x	x	x								
11	Avda. Ramón y Cajal	Avda. Ramón y Cajal esq. C/ Príncipe de Vergara	3º 40' 38,48"O	40º 27' 05,31"N	708	UT	x						x									x
16	Arturo Soria	C/ Arturo Soria esq. C/ Vizconde de los Asilos	3º 38' 21,24"O	40º 26' 24,17"N	693	UF	x		x			x										x
17	Villaverde	C/. Juan Peñalver	3º 42' 47,96"O	40º 20' 49,70"N	604	UF	x	x				x										
18	Farolillo	Calle Farolillo - C/Ervigio	3º 43' 54,67"O	40º 23' 41,21"N	630	UF	x	x	x	x		x	x					x				x
24	Casa de Campo	Casa de Campo (Terminal del Teleférico)	3º 44' 50,45"O	40º 25' 09,68"N	642	S	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
27	Barajas Pueblo	C/. Júpiter, 21 (Barajas)	3º 34' 48,11"O	40º 28' 36,94"N	621	UF	x					x										
35	Pza. del Carmen	Plaza del Carmen esq. Tres Cruces.	3º 42' 11,42"O	40º 25' 9,15"N	659	UF	x	x	x			x										
36	Moratalaz	Avd. Moratalaz esq. Camino de los Vinateros	3º 38' 43,10"O	40º 24' 28,61"N	685	UT	x	x	x	x												x
38	Cuatro Caminos	Avda. Pablo Iglesias esq. C/ Marqués de Lema	3º 42' 25,66"O	40º 26' 43,96"N	698	UT	x	x		x	x		x					x				x
39	Barrio del Pilar	Avd. Betanzos esq. C/ Monforte de Lemos	3º 42' 41,55"O	40º 28' 41,62"N	674	UT	x		x			x										x
40	Vallecas	C/ Arroyo del Olivar esq. C/ Río Grande.	3º 39' 05,48"O	40º 23' 17,35"N	677	UF	x	x		x												x
47	Mendez Alvaro	C/ Juan de Mariana / Pza. Amanecer Mendez Alvaro	3º 41' 12,57'O	40º 23' 53,21"N	599	UF	x			x	x											
48	Castellana	C/ Jose Gutierrez Abascal	3º 41' 25,32'O	40º 26' 23,63"N	676	UT	x			x	x											
49	Parque del Retiro	Paseo Venezuela- Casa de Vacas	3º 40' 57,3"O	40º 24' 52"N	662	UF	x					x										
50	Plaza Castilla	Plaza Castilla (Canal)	3º 41' 19,57"O	40º 27' 56,06"N	728	UT	x			x	x							x				
54	Ensanche de Vallecas	Avda La Gavia / Avda. Las Suertes	3º 36' 43,62'O	40º 22' 22,56"N	627	UF	x					x			x	x	x	x				x
55	Urb. Embajada	C/ Riaño (Barajas)	3º 34' 50,69"O	40º 27' 45,11"N	618	UF	x			x			x	x								
56	Pza. Fernández Ladreda	Pza. Fernández Ladreda - Avda. Oporto	3º 43' 07,42"O	40º 23' 05,87"N	604	UT	x		x			x				x	x	x	x			x
57	Sanchinarro	C/ Princesa de Eboli esq C/ Maria Tudor	3º 39' 37,81'O	40º 29' 39,15"N	700	UF	x	x	x	x								x				
58	El Pardo	Avda. La Guardia	3º 46' 28,60"O	40º 31' 05,01"N	615	S	x					x										
59	Juan Carlos I	Parque Juan Carlos I (frente oficinas mantenimiento)	3º 36' 32,66"O	40º 27' 54,90"N	660	S	x					x			x	x	x	x	x	x	x	x
60	Tres Olivos	Plaza Tres Olivos	3º 41' 23,14"O	40º 30' 02,12"N	715	UF	x			x		x								x		x

En este proyecto vamos a realizar el análisis, tanto descriptivo como predictivo, de una estación de control de cada uno de los tipos existentes. En particular, hemos seleccionado las siguientes estaciones de control:

- Estación "Escueltas Aguirre". Tipo UT
- Estación "Casa de Campo". Tipo S
- Estación "Farolillo". Tipo UF

La razón principal de la elección anterior se debe a que son, de cada tipo, las estaciones de control que más contaminantes miden. En este sentido, las estaciones de "Escuelas Aguirre" (UT) y "Casa de Campo" (S) proporcionan mediciones de todos los contaminantes; mientas que la estación "Farolillo" (UF) no recoge información del contaminante PM2.5, pero como se verá en la sección "Cálculo del ICA", este contaminante no se utilizará para el cálculo del ICA.

Enlace: https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=9e42c176313eb410VgnVCM1000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD

Histórico de datos

El Sistema Integral de la Calidad del Aire del Ayto. de Madrid obtener la información recogida por las estaciones de control de calidad del aire, con los datos horarios por anualidades de 2001 a 2018 (en el año en curso la información se actualiza mensualmente).

Enlace: https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=f3c0f7d512273410VgnVCM2000000c205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default

Los datos se descargan comprimidos en un archivo .zip. En su interior se encuentra un archivo (.txt o .csv) por cada uno de los 12 meses del año. En cada uno de estos ficheros, nos encontraremos todas las filas del documento de texto con la siguiente estructura:
"28079004013802150401000 8V00009V00008V00007V00006V....."

Cada fila contiene los 24 valores horarios de un día, así como 30/31 filas contiguas correspondientes a los valores de los días del mes, repitiéndose con cada magnitud de todas las estaciones que lo miden.

ESTACIÓN	MAGNITUD	TÉCNICA	DATO HORARIO	AÑO	MES	DIA	HORA 1	HORA 2	HORA 3	HORA 4
28079004	01	38	02	15	04	01	00008V	00009V	00008V	00007V
28079004	01	38	02	15	04	02	00005V	00005V	00005V	00005V
28079004	01	38	02	15	04	03	00006V	00007V	00006V	00005V
28079004	01	38	02	15	04	04	00006V	00006V	00005V	00005V

Para el procesado de cada uno de los ficheros de texto se ha desarrollado el script *prepro_hist_ca.R*.

To do:

- Describir lo que se hace *prepro_hist_ca.R*.

Datos en tiempo real

El Sistema Integral de la Calidad del Aire del Ayuntamiento de Madrid permite conocer en cada momento los niveles de contaminación atmosférica en el municipio. En este conjunto de datos puede obtener la información actualizada en tiempo real, actualizándose estos datos cada hora (entre los minutos 20 y 30).

Enlace: https://datos.madrid.es/portal/site/egob/menuitem.c05c1f754a33a9fbe4b2e4b284f1a5a0/?vgnextoid=41e01e007c9db410VgnVCM2000000c205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM100000171f5a0aRCRD&vgnextfmt=default

Para la descarga de datos en tiempo real se ha desarrollado el script *download_RT.R*.

To do:

- ¿Formato de los datos en tiempo real? -> Mismo que en el histórico
- Incorporación de los datos en tiempo real al histórico de datos
- Automatización del proceso de descarga

Cálculo del ICA

El índice de calidad de aire (ICA) es un indicador que sirve para informar de la calidad del aire a la población de una manera clara y sencilla. Para obtener el ICA es necesario medir en tiempo real una serie de parámetros, tales como los contaminantes SO2, NOx, CO, PM10 y O3, en estaciones distribuidas en distintas zonas.

El conjunto de valores que el ICA puede tomar se agrupa en intervalos a los que se les asocia una trama o color característico de la calidad del aire de una zona determinada. Generalmente la representación del ICA se dividide en cinco categorías, que definen los estados de calidad de aire según sea "muy buena", "buena", "mejorable", "mala" o "muy mala".

A partir de los datos de concentración de contaminantes descargados del portal del Ayto. de Madrid, se ha desarrollado el script *calcular_ICA.R* que calcula el ICA correspondiente a cada estación de control de manera horaria.

Metodología de cálculo del ICA

Para el cálculo del ICA se han tenido en cuenta los siguientes cinco contaminantes (según las directivas existentes en materia de contaminación atmosférica) para los cuales la Comunidad Europea ha establecido niveles de concentración que no se deben superar para la protección de la salud humana y, en algunos casos, de los ecosistemas:

1. Dióxido de azufre (SO2)
2. Dióxido de nitrógeno (NO2)
3. Partículas (PM10)
4. Ozono (O3)
5. Monóxido de carbono (CO)

Factores de cálculo del ICA

Los factores a aplicar en el cálculo del ICA se obtienen teniendo en cuenta los valores límite establecidos para cada contaminante en el RD 102/2011.

Dióxido de Azufre (SO2)

Se establece un valor límite diario para la concentración de SO2 de 125 µg/m3 que no puede superarse en más de 24 ocasiones por año civil, y un valor límite horario de 350 µg/m3 que no puede superarse en más de 3 ocasiones por año civil. Obtenemos por tanto, dos factores para el cálculo del ICA parcial:

Valores límite	Factor de cálculo
VL diario= 125 µg/m3	100/125=0.800
VL horario=350 µg/m3	100/350=0.286

Dióxido de Nitrógeno (NO2)

Se establece un valor límite horario para este contaminante de 200 µg/m3 que no puede superarse en más de 18 ocasiones por año civil, quedando el factor de cálculo del ICA:

Valores límite	Factores de cálculo
VL horario=350 µg/m3	100/200=0.500

Partículas (PM10)

Se establece un valor límite diario para este contaminante de 50 µg/m3 que no puede superarse en más de 35 ocasiones por cada año civil. En ausencia de valor de referencia horario para PM10 en la legislación igente, se ha considerado como tal el valor de 100 µg/m3, que es el valor tetrahorario indicado según el documento de 1999 *Health Guidelines for Vegetation Fire Events*, de la OMS, para definir el estado 1 de aviso

en las *Guidelines of wildfire emergency action plan of the Western States Air Resources Council*, de EEUU, obteniéndose los siguientes factores:

Valores límite	Factor de cálculo
VL diario= 50 µg/m3	100/50=2
VL horario=150 µg/m3	100/150=0.67

Ozono (O3)

Se establece un valor objetivo para la protección de la salud humana como una máxima diaria de las medias móviles octohorarias de 120 µg/m3 que no deberá superarse en más de 25 días por cada año civil de promedio en un periodo de 3 años. El máximo de las medias octohorarias de día deberá seleccionarse examinando promedios móviles de 8 horas, calculados a partir de datos horarios y actualizados cada hora. Cada promedio octohorario así calculado se asignará al día en que dicho promedio termina, es decir, el primer periodo de cálculo para un día cualquiera será el periodo a partir de las 17:00 del día anterior hasta la 1:00 de dicho día; el último periodo de cálculo para un día cualquiera será el periodo a partir de las 16:00 hasta las 23:59 de dicho día.

En el caso de este contaminante también es necesario tener en cuenta el umbral de información a la población, establecido en 180 µg/m3 de promedio horario, por lo que obtenemos los siguientes factores de cálculo:

Valores límite	Factor de cálculo
VL octohorario= 120 µg/m3	100/120=0.833
VL horario=180 µg/m3	100/180=0.556

Monóxido de Carbono (CO)

Para este contaminante se establece una máxima diaria de las medias móviles octohorarias de 10 mg/m3. El máximo de las medias móviles octohorarias del día deberá seleccionarse examinando promedios móviles de 8 horas, calculados a partir de datos horarios y actualizados cada hora. Cada promedio octohorario así calculado se asignará al día en que dicho promedio terminó, es decir, el primer periodo de cálculo para un día cualquiera será el periodo a partir de las 17:00 del día anterior hasta la 1:00 de dicho día, el último periodo se cálculo para un día cualquiera será el período a partir de las 16:00 hasta las 23:59 de dicho día. Obtenemos entonces el siguiente factor de cálculo:

Valores límite	Factor de cálculo
VL octohorario= 10 mg/m3	100/10=10

Cálculo del ICA de cada contaminante y cada estación a nivel horario

El cálculo del ICA de cada contaminante y cada estación se realiza multiplicando los factores calculados anteriormente con la concentración propia de cada contaminante.

Cálculo del ICA global

Una vez calculados los ICAs de cada contaminante, el ICA de cada estación se corresponderá con el mayor de ellos. Finalmente, el ICA representativo de una ciudad corresponderá al máximo ICA de cada una de las estaciones de control.

To do:

- Hablar del formato de salida de los datos preprocesados (almacenamiento?)

Información meteorológica

Está demostrado científicamente que las condiciones climáticas condicionan en gran parte el grado de disipación de contaminantes. Además, el clima influye directamente en la formación de algunos compuestos y sus concentraciones. Por tanto, los efectos del cambio climático, con condiciones climáticas extremas y un calentamiento general de la atmósfera, afectan a la calidad del aire.

Debido a ello, en este proyecto se ha decidido analizar la evolución de las condiciones climatológicas de la ciudad de Madrid durante los últimos años, confrontando los resultados obtenidos con la evolución de la calidad del aire en esta ciudad. Para ello, se han evaluado diferentes portales o páginas web que ofrecen información climática (histórica y en tiempo real) para su descarga. En particular se han analizado tres portales web: AEMET (http://www.aemet.es), OpenWeatherMap (https://openweathermap.org) y WeatherUnderground (https://www.wunderground.com). Al igual que ocurre con los datos relativos a la calidad del aire, en este caso se ha dividido la descarga en dos subapartados descritos a continuación: histórico de datos y predicción.

Histórico de datos

Para la descarga de datos históricos se ha desarrollado el script *getHistWeather.R*, el cual se centra en la descarga de los datos históricos necesarios para evaluar la evolución de la calidad del aire con las condiciones climatológicas desde el año 2001.

Las principales características encontradas de cada uno de los portales analizados se describen brevemente a continuación:

- **AEMET**:
  - Granularidad de la información: agregada en intervalos de 6 horas
  - Profundidad: Mayo 2013
  - Formato: .xls
  - Licencia: gratuito

- **OpenWeatherMap**:
  - Granularidad de la información: horaria
  - Profundidad: 5 últimos años
  - Formato: JSON

    - Licencia: de pago

- **WeatherUnderground**:
    - Granularidad de la información: horaria
    - Profundidad: Enero 2001
    - Formato: JSON
    - Licencia: gratuito

Según lo anterior, se ha optado por la descarga de los datos climatológicos a través del portal de WeatherUnderground. Para ello ha sido necesario solicitar una API key, cuya principal limitación es la realización máxima de 500 peticiones/día y 10 peticiones/min.

Entre otras, algunas de las mediciones horarias que aporta este portal son la temperatura, humedad, velocidad y dirección del viento, visibilidad, probabilidad de precipitación y presión atmosférica.

El procesado de los datos descargados se ha realizado a través del script *prepro_tiempo.R*.

To do:

- Describir lo que se hace *prepro_tiempo.R*.
- Hablar del formato de salida de los datos preprocesados (almacenamiento?)

Predicción

Para la descarga de la predicción meteorológica se ha desarrollado el script *getYestForeWeather.R*. Al igual que en el caso anterior, se ha realizado un estudio comparativo de los datos ofrecidos por cada uno de los tres portales mencionados anteriormente. La elección final ha estado condicionada por la selección del portal escogido para la descarga de información histórica, ya que el formato y los campos de los datos de predicción son los mismos, lo cual simplifica enormemente su adaptación. Por lo tanto, el procesado de los datos descargados se realiza también a través del script *prepro_tiempo.R*.

En este caso, el portal de WeatherUnderground proporciona la predicción meteorológica a nivel horario con una profundidad de 10 días.

Análisis descriptivo

Uno de los objetivos de este proyecto, a parte de predecir el comportamiento del ICA, es analizar el comportamiento histórico de este índice, así como de los contaminantes y condiciones meteorológicas que pueden estar relacionados con el mismo. Precisamente del estudio y análisis de estos factores, se puede observar la correlación, directa o inversa, que presentan entre sí.

Este análisis descriptivo servirá como antesala para la preparación y selección de atributos que participarán en el análisis predictivo, explicado en la siguiente sección.

Según lo anterior, se ha dividido el análisis descriptivo en dos apartados claramente diferenciados:

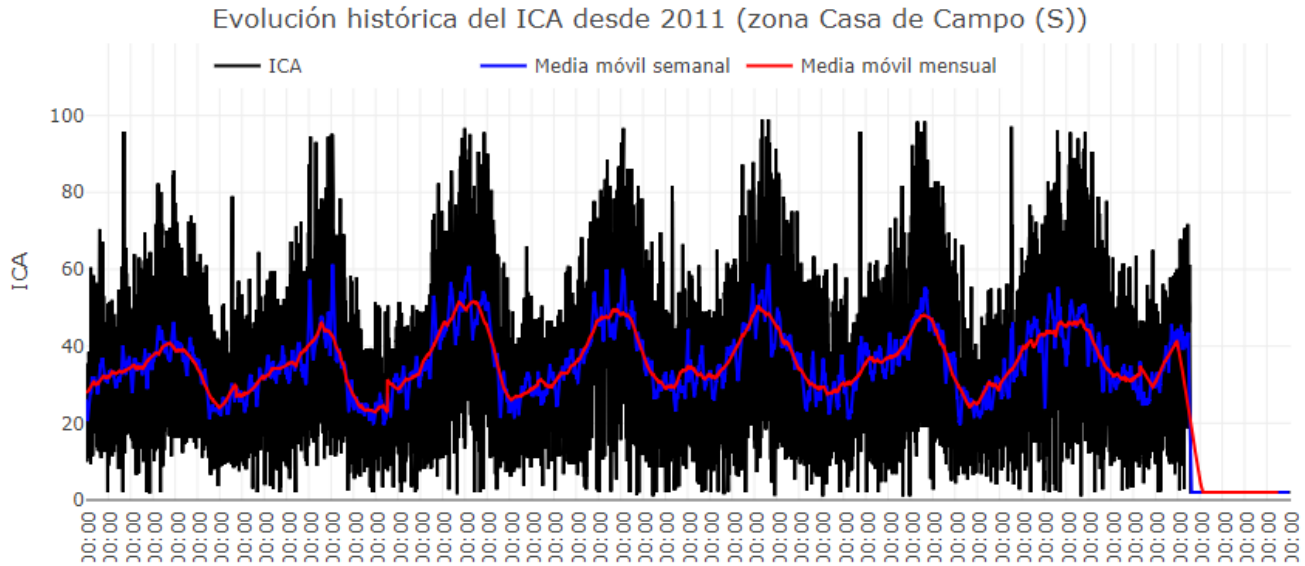
- Evolución de la calidad del aire (ICA y principales contaminantes)
- Evolución de las condiciones climatológicas

A continuación se detallan los principales resultados obtenidos:

Evolución interanual de la calidad del aire (ICA y contaminantes)

Este apartado se ha centrado en el estudio de la evolución del parámetro fundamental de este proyecto, es decir, el ICA. Además de el estudio de este índice, se ha estudiado la evolución de cada uno de los cinco contaminantes que influyen de manera más directa en su determinación (SO2, NOx, CO, PM10 y O3).

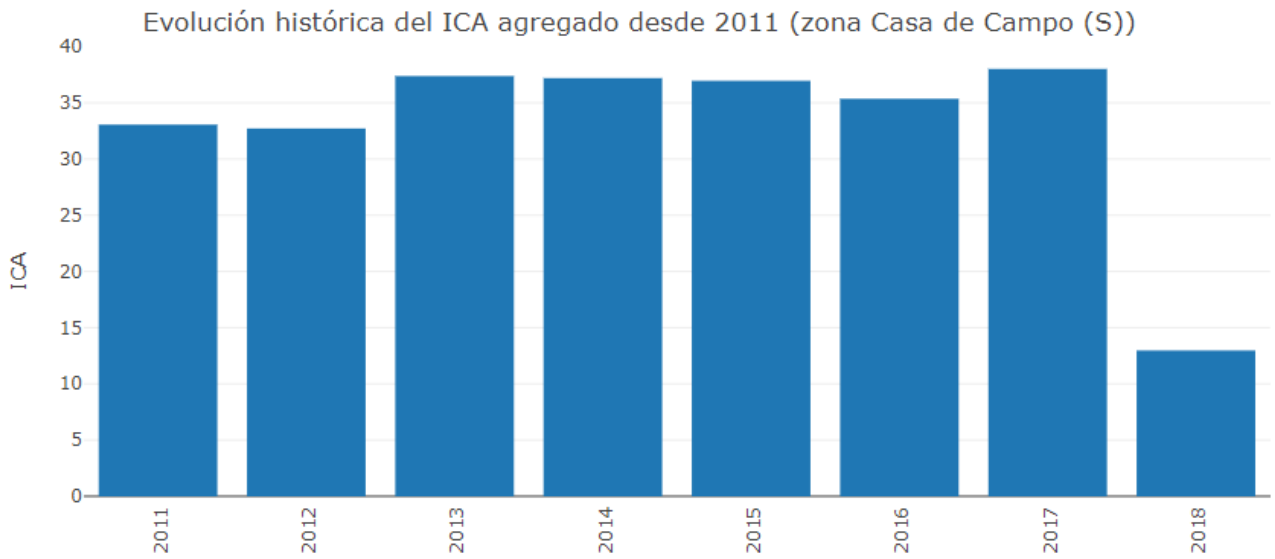
La siguiente figura muestra la evolución anual del ICA desde el año 2011 para una de las estaciones de control seleccionadas, en particular para la estación de Casa de Campo.



En color negro se observa la representación gráfica de los datos en bruto, sin ningún tipo de filtrado. Debido al alto volumen de información representada (una muestra cada hora, representando un total de 70128 instancias), hemos filtrado la señal. Para ello hemos utilizado un filtro basado en ventanas móviles; en primer lugar hemos filtrado la señal a nivel semanal (color azul) y en segundo lugar hemos filtrado la señal a nivel mensual (color rojo). Como se puede observar, el filtrado a nivel mensual es más agresivo, por lo que la señal representada muestra la evolución media del ICA a lo largo de los años, pero omite la información puntual que desvela su evolución interanual.

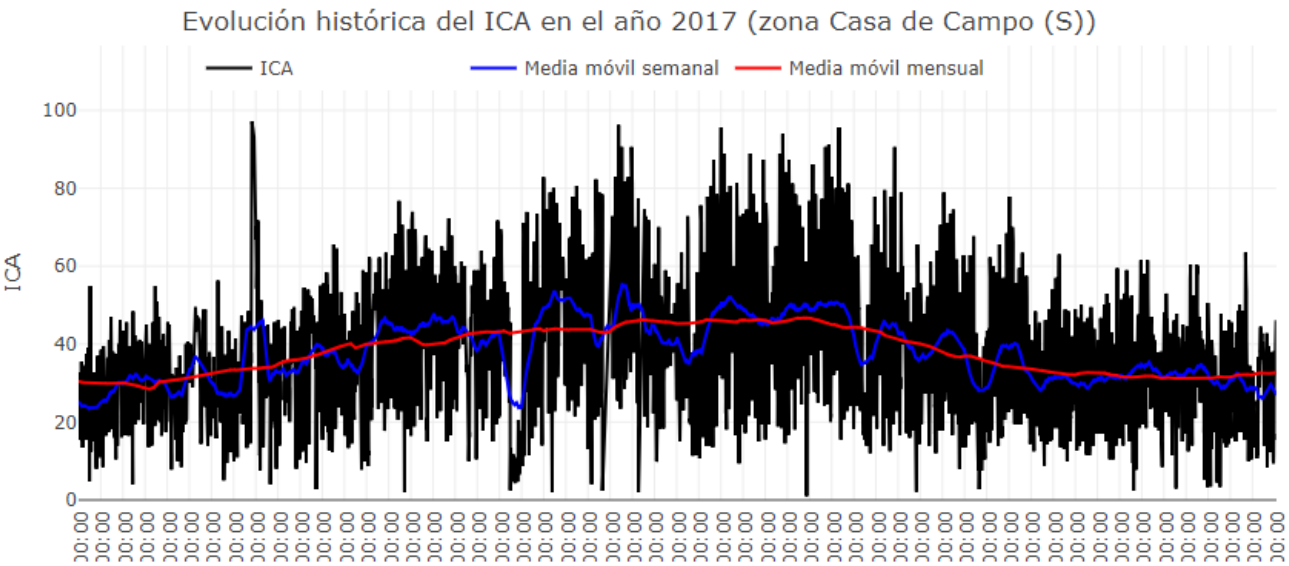
Tal y como se puede apreciar en la imagen anterior, la evolución del ICA es completamente periódica, lo que facilitará su aprendizaje posterior en la etapa de modelado predictivo.

Además de la representación anterior, la siguiente imagen muestra la evolución anual del ICA tomando como referencia el valor medio de este índice para cada uno de los años. El resultado se muestra a través de un diagrama de barras:



A pesar de que, en términos medios, la evolución del ICA es bastante estable a lo largo de los años, cabe destacar el aumento significativo producido en el año 2013. En los tres años siguientes, la tendencia fue decreciente, pero en el año 2017 esta volvió a subir hasta niveles del año 2013. En lo que va de año, 2018 muestra unos niveles que hacen pensar en un descenso importante en el nivel medio del ICA.

Con el objetivo de ver la evolución del ICA a lo largo de un año en particular, la siguiente imagen muestra el resultado obtenido para el último año entero registrado, es decir, para el año 2017. En esta imagen se observa claramente el movimiento oscilatorio del ICA, alcanzando sus valores mínimos en épocas frías (enero y diciembre) y sus valores máximos en épocas cálidas (julio y agosto). A priori estos resultados pueden parecer incoherentes o contrarios a que se cabría esperar de ellos, pero es en este punto donde radica la grandeza del análisis descriptivo. Los resultados muestran que, o bien las condiciones climatológicas típicas del verano o las características propias de la ciudad de Madrid en esta época (un volumen menor de personas y por lo tanto un volumen menor de tráfico, una menor producción industrial, etc.), o una combinación de las dos, podrían influir directamente sobre el ICA.



A continuación nos planteamos analizar la evolución del ICA con respecto a cada uno de los cinco contaminantes estudiados. Los resultados se muestan gráficamente en las siguientes imágenes:

To do: Preparar los plots de tal manera que se dibujen dos gráficas, una encima de la otra, compartiendo eje temporal (eje x). De esta manera se podrá visualizar fácilmente si existe alguna correlación entre el ICA y cada contaminante. Hacer a nivel de valores medios

Raw data

Raw data

Raw data

Raw data

Raw data

Evolución interanual de las condiciones climátológicas

To do: Preparar los plots de tal manera que se dibujen dos gráficas, una encima de la otra, compartiendo eje temporal (eje x). De esta manera se podrá visualizar fácilmente si existe alguna correlación entre el ICA y cada contaminante. Hacer a nivel de valores medios

Modelado predictivo

El principal objetivo de este proyecto consiste en el desarrollo de un modelo que permita predecir la evolución del ICA. Al no disponer de mediciones en tiempo real de los principales contaminantes que determinan este índice, el problema se plantea como una serie temporal, cuyo único parámetro de aprendizaje es la propia serie histórica del ICA. Además de este dato, disponemos de información meteorológica en tiempo real, pero no es una información que tenga entidad propia para servir como entrada de los modelos.

Como se ha explicado en el apartado "Estaciones de control", en este proyecto nos hemos centrado en el análisis de una estación de control de cada uno de los tipos existentes. En particular, hemos seleccionado las siguientes estaciones de control:

- Estación "Escueltas Aguirre". Tipo UT
- Estación "Casa de Campo". Tipo S
- Estación "Farolillo". Tipo UF

A continuación se explica brevemente el algoritmo seleccionado para entrenar la serie temporal del ICA para cada una de las tres estaciones anteriores.

Series temporales

Una serie temporal se define como una colección de observaciones de una variable recogidas secuencialmente en el tiempo. Estas observaciones se suelen recoger en instantes de tiempo equiespaciados.

La característica fundamental de las series temporales es que las observaciones sucesivas no son independientes entre sí, y el análisis debe llevarse a cabo teniendo en cuenta el orden temporal de las observaciones. Los métodos estadísticos basados en la independencia de las observaciones no son válidos para el análisis de series temporales porque las observaciones en un instante de tiempo dependen de los valores de la serie en el pasado.

Una serie temporal puede ser discreta o continua dependiendo de cómo sean las observaciones. Si se pueden predecir exactamente los valores, se dice que las series son determinísticas. Si el futuro sólo se puede determinar de modo parcial por las observaciones pasadas y no se pueden determinar exactamente, se considera que los futuros valores tienen una distribución de probabilidad que está condicionada a los valores pasados (series estocásticas). En nuestro caso, este proyecto trabaja con la evolución temporal del ICA, y su predicción solo se puede determinar de manera aproximada en base a los datos históricos, por lo que estamos hablando de una serie temporal estocástica.

Las componentes o fuentes de variación que se consideran habitualmente son las siguientes:

1. **Tendencia**: Se puede definir como un cambio a largo plazo que se produce en relación al nivel medio, o el cambio a largo plazo de la media. La tendencia se identifica con un movimiento suave de la serie a largo plazo.
2. **Efecto estacional**: Muchas series temporales presentan cierta periodicidad o dicho de otro modo, variación de cierto periodo (anual, mensual ...). Por ejemplo, el paro laboral aumenta en general en invierno y disminuye en verano. Estos tipos de efectos son fáciles de entender y se pueden medir explícitamente o incluso se pueden eliminar del conjunto de los datos, desestacionalizando la serie original.
3. **Componente aleatoria**: Una vez identificados los componentes anteriores y después de haberlos eliminado, persisten unos valores que son aleatorios. Se pretende estudiar qué tipo de comportamiento aleatorio presentan estos residuos, utilizando algún tipo de modelo probabilístico que los describa.

De las tres componentes reseñadas, las dos primeras son componentes determinísticas, mientras que la última es aleatoria.

Las series temporales se pueden clasificar en:

Estacionarias: una serie es estacionaria cuando es estable, es decir, cuando la media y la variabilidad son constantes a lo largo del tiempo. Esto se refleja gráficamente en que los valores de la serie tienden a oscilar alrededor de una media constante y la variabilidad con respecto a esa media también permanece constante en el tiempo. Es una serie básicamente estable a lo largo del tiempo, sin que se aprecien aumentos o 5 disminuciones sistemáticos de sus valores.

No Estacionarias: son series en las cuales la media y/o variabilidad cambian en el tiempo. Los cambios en la media determinan una tendencia a crecer o decrecer a largo plazo, por lo que la serie no oscila alrededor de un valor constante.

Prophet

Facebook ha desarrollado un algoritmo de predicción de series temporales llamado Prophet, disponible de manera abierta en Python y R (https://github.com/facebook/prophet). Para Facebook, el *forecast* o predicción es fundamental para su actividad principal. Prophet ha sido una pieza clave para mejorar la capacidad de Facebook de crear una gran cantidad de pronósticos fiables utilizados para la toma de decisiones e incluso en las características del producto.

Prophet está optimizado para realizar tareas de predicción sobre conjuntos de datos que generalmente presentan alguna de las siguientes características:

- observaciones horarias, diarias o semanales con al menos algunos meses (preferiblemente un año) de histórico
- fuertemente estacionales a nivel "humano": día de la semana y época del año
- días festivos importantes que ocurren a intervalos irregulares que se conocen de antemano
- un número razonable de observaciones incompletas o con valores atípicos
- cambios de tendencia no lineales

La principal característica de Prophet es su facilidad de implementación, donde una persona "no experta" en el campo de las series temporales puede entrenar un algoritmo con relativa facilidad. A pesar de ello, los resultados que ofrece son generalmente muy buenos, ofreciendo una solución cuyo equilibrio esfuerzo-resultado es enorme.

Aunque la configuración básica de Prophet es muy sencilla, para aquellos usuarios con un nivel de conocimiento en el área superior, el algoritmo también ofrece la posibilidad de variar o *tunear* una serie de parámetros, lo cual permitirá alcanzar mejores resultados de predicción.

En esencia, el algoritmo Prophet es un modelo de regresión aditiva que trabaja con cuatro componentes principales:

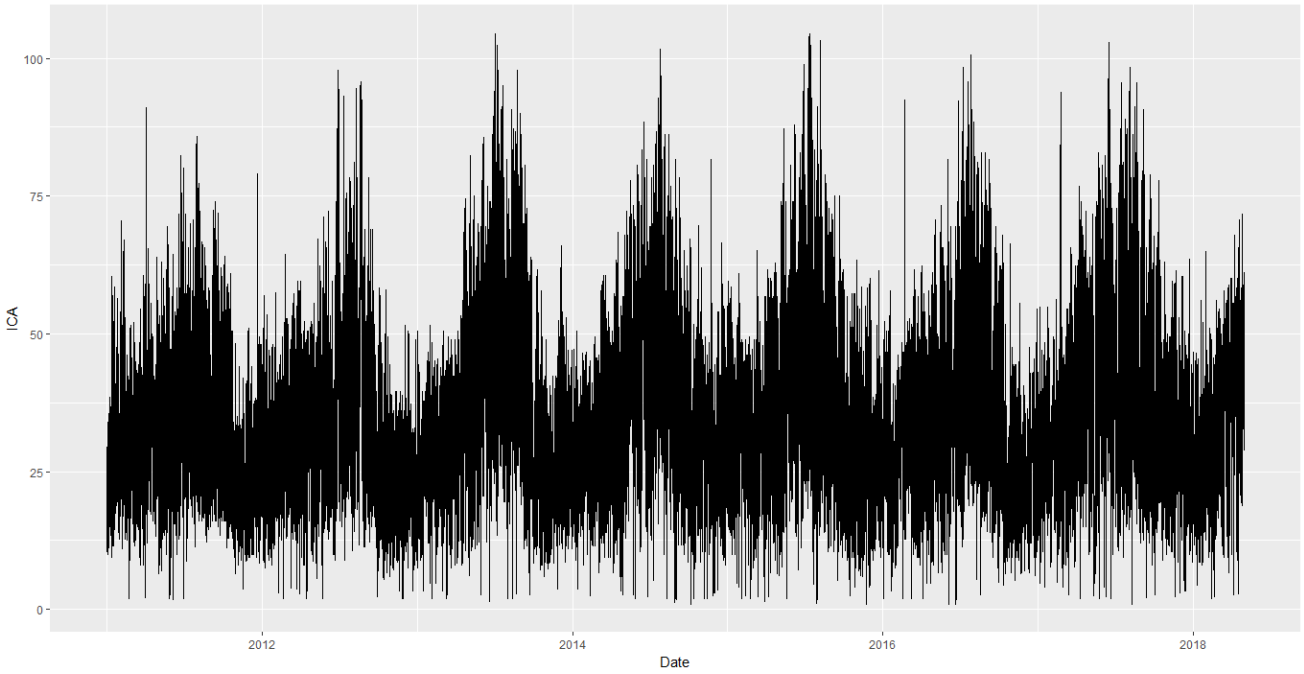
- La curva tendencia lineal o logística. Prophet detecta automáticamente los cambios en las tendencias al seleccionar los "puntos de cambio" o *changepoints* en los datos.
- La componente estacional anual modelado a partir de la serie de Fourier.
- La componente estacional semanal que usa variables ficticias.
- Una lista de festivos proporcionada por el usuario.

ESTACIÓN	MAGNITUD	TÉCNICA	DATO HORARIO	AÑO	MES	DIA	HORA 1	HORA 2	HORA 3	HORA 4
28079004	01	38	02	15	04	01	00008V	00009V	00008V	00007V
28079004	01	38	02	15	04	02	00005V	00005V	00005V	00005V
28079004	01	38	02	15	04	03	00006V	00007V	00006V	00005V
28079004	01	38	02	15	04	04	00006V	00006V	00005V	00005V

Metodología

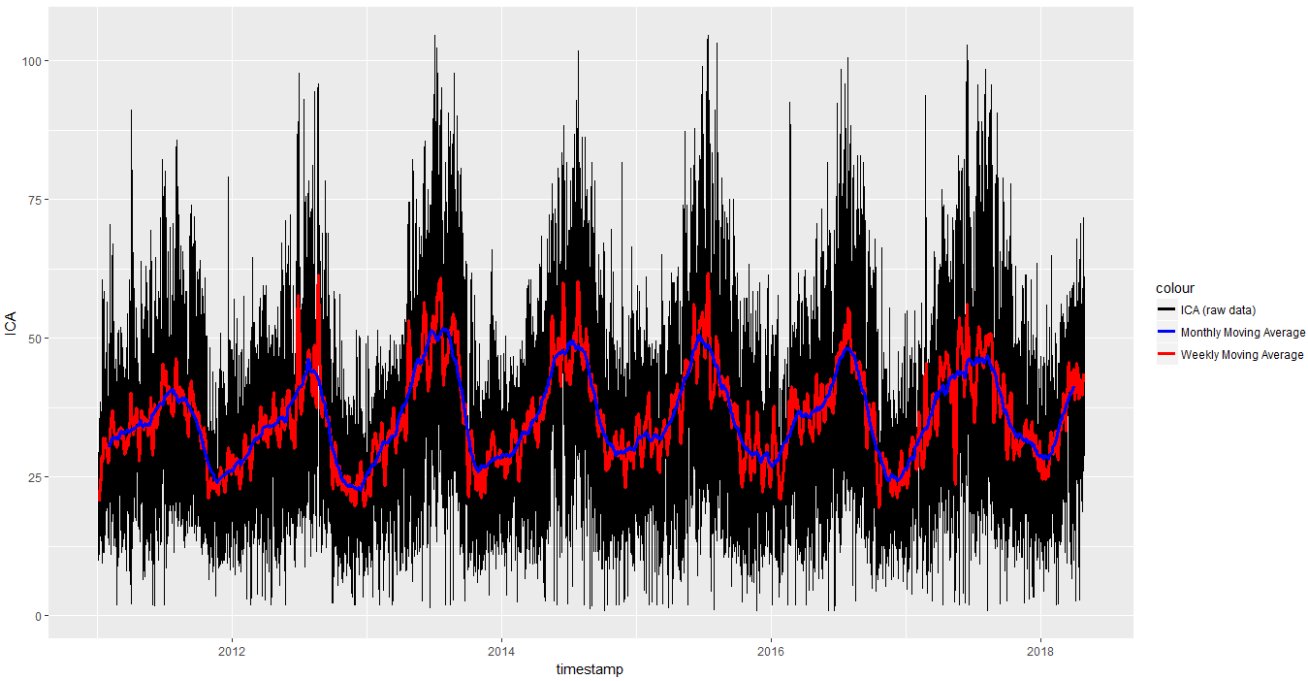
Para el desarrollo del modelo que permita predecir la evolución del ICA hemos utilizado los datos históricos recogidos y procesados según lo explicado en la sección "Calidad del aire". Estos datos contienen información histórica de la evolución del ICA desde el año 2001 hasta mediados de 2018 (momento de realización del presente proyecto). Debido al alto volumen de información (granularidad horaria del dato), se han utilizado los datos posteriores al año 2011 (inclusive).

En primer lugar se realizó una división del conjunto de datos (previamente filtrado por el código de estación) en dos subconjuntos: **entrenamiento** (datos comprendidos entre el 01/01/2011 y 31/12/2016) y **validación** (datos posteriores al 01/01/2017). A continuación se representa el conjunto de datos correspondiente a la estación de control de la Casa de Campo:



Tal y cómo se puede observar, debido a que disponemos de mediciones del ICA a nivel horario, su prepresentación para 8 años no excesivamente representativa y no permite ver correctamente la evolución detallada de este índice. Debido a que los datos presentan varias componentes estacionales (periodicidad diaria, semanal, estacional, anual...) decidimos filtrar los datos para que el algoritmo seleccionado aprenda la estacionalidad a dos niveles diferentes: anual y diaria. Con la primera podremos detectar la evolución interanual del ICA, mientras que la segunda nos permitirá ver, de manera aproximada, la evolución diaria del mismo.

Para realizar el aprendizaje de la periodicidad interanual, hemos decidido filtrar la información de entrada, ya que no necesitamos el detalle de la evolución a nivel diario. Para ello implementamos dos filtrados de ventana móvil: media móvil semanal y mensual. El resultado se puede observar a continuación:

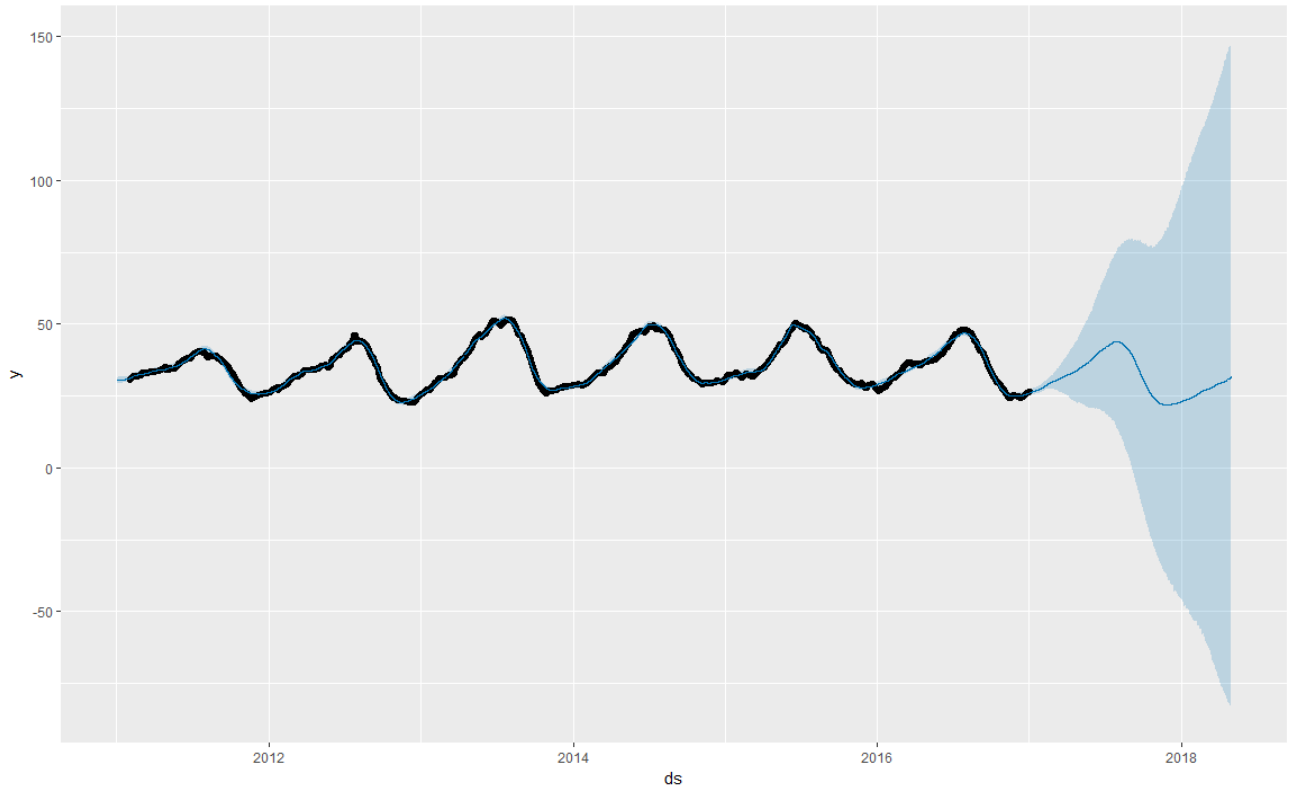


En la imagen anterior podemos ver claramente el caracter periódico del ICA a lo largo de los años. Será precisamente la señal filtrada a nivel mensual la que utilicemos para entrenar el algoritmo.

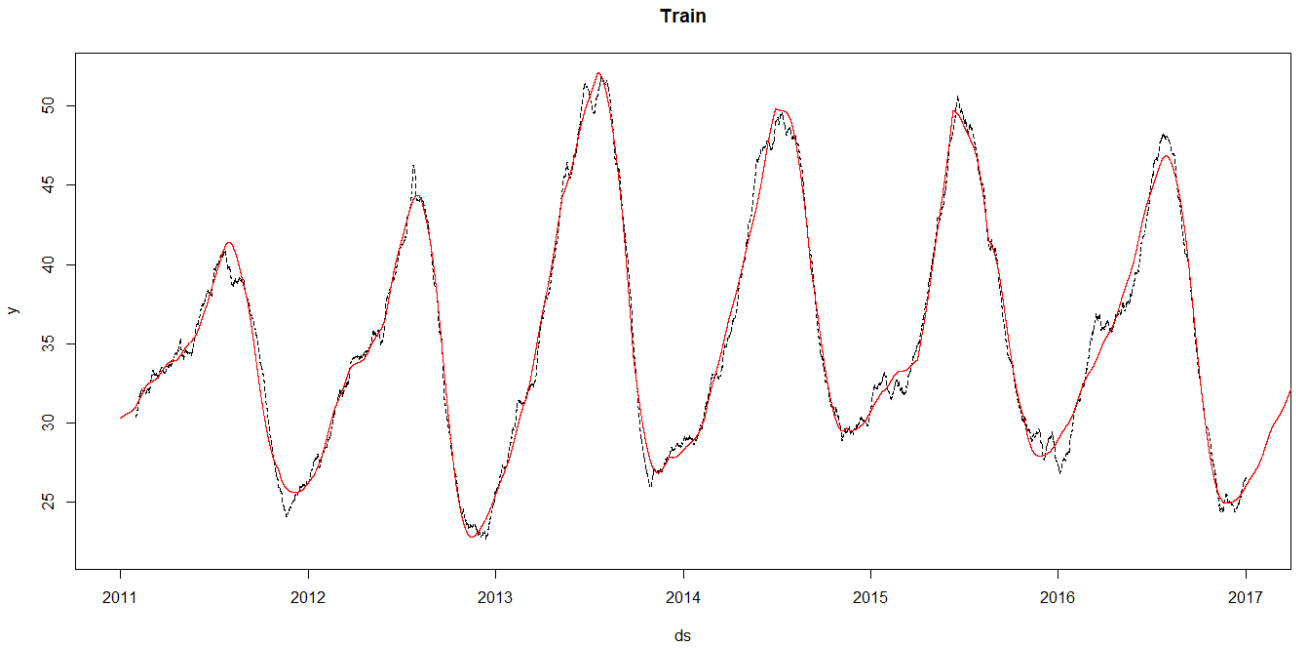
Prophet permite especificar eventos o fechas especiales, tales como festivos, fines de semana, o cualquier tipo de evento que pueda provocar una alteración en la variable que se quiere predecir. En nuestro caso hemos decidido especificar los días correspondientes a fines de semana, ya que consideramos que la distinción de días entre laborales y no laborales condiciona enormemente la evolución del ICA.

Una característica del algoritmo Prophet es que las variables de entrada deben responder a dos nombres: la variable *y* corresponde a la serie temporal que queremos predecir (en este caso el ICA) y la variable *ds* corresponde a la marca temporal (las fechas).

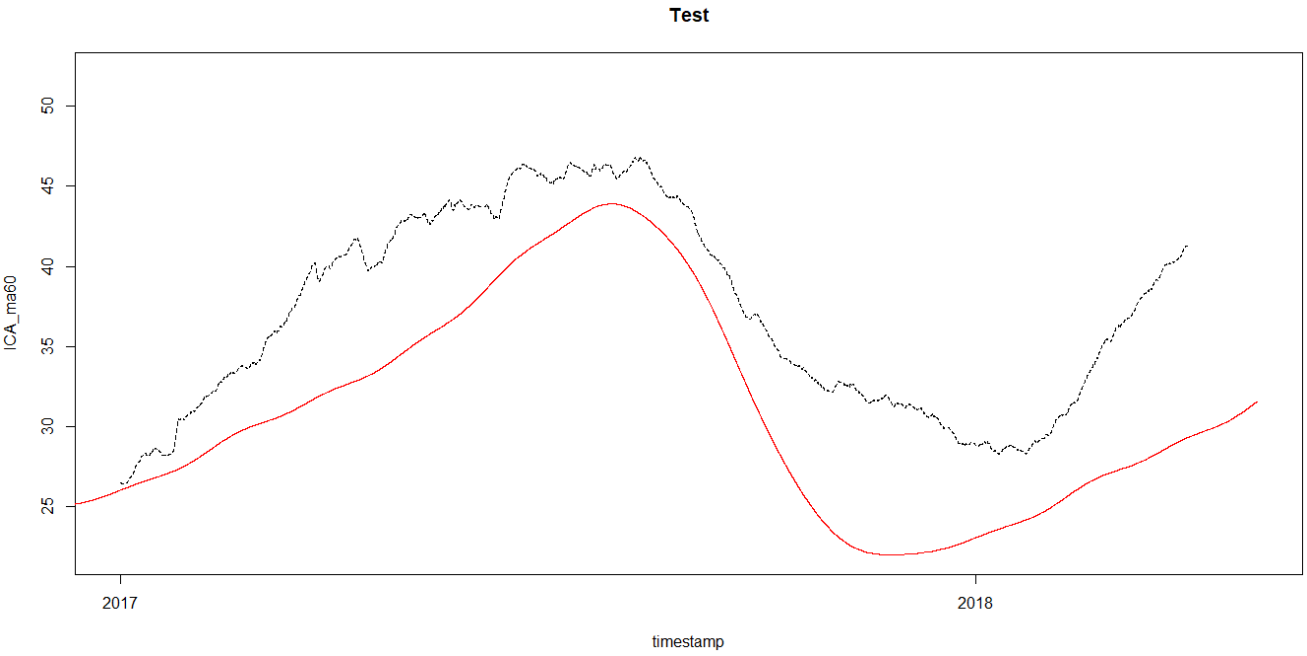
Para el entrenamiento del algoritmo se utilizan los datos reservados para tal fin, generando una representación gráfica de la evolución aprendida:



La siguiente imagen muestra de manera un poco más detallada la generalización que realiza el algoritmo con los datos de entrenamiento:

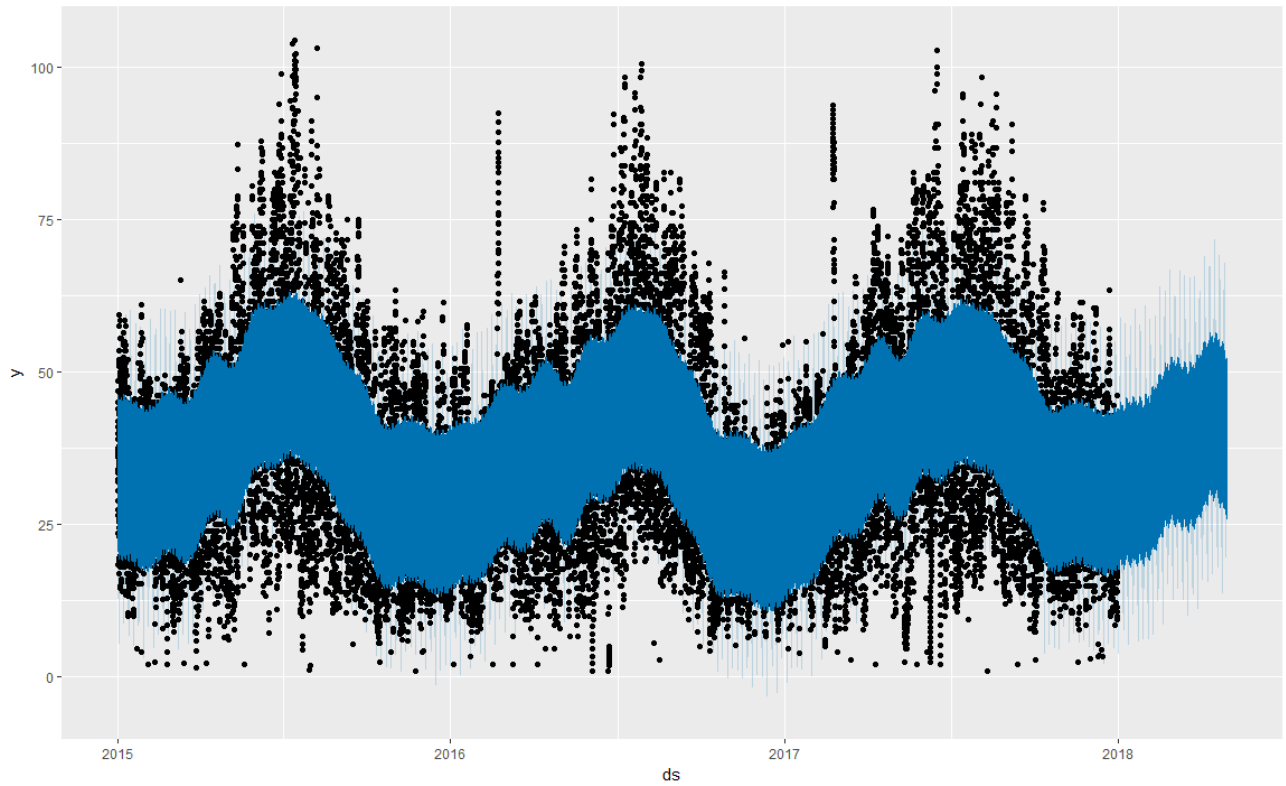


Una vez modelado el algoritmo, se evalua con el conjunto de datos reservados para validación. En este sentido, la representación gráfica de la predicción del algoritmo entrenado se muestra a continuación:



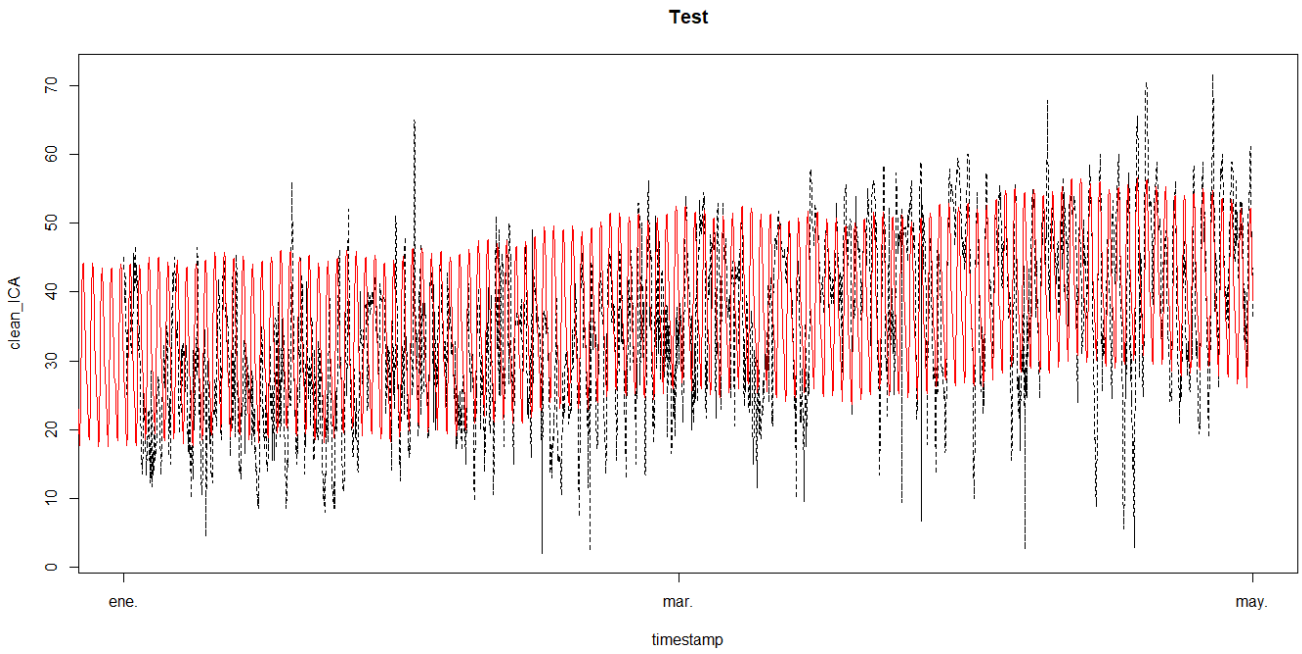
Tal y cómo se puede observar en la imagen anterior, el algoritmo ha aprendido correctamente la evolución general anual del ICA, pero los valores absolutos se alejan ligeramente de la realidad. La razón principal radica en que el algoritmo solo dispone de información histórica para predecir el comportamiento futuro, pero existen una serie de condicionantes que hacen que esta tendencia varie particularmente a lo largo de cada año. Por ejemplo, las condiciones climatológicas, el aumento o descenso en la venta de vehículos, las políticas reguladoras en materia de medio ambiente, etc. pueden afectar notablemente la evolución del ICA, hechos que no quedan reflejados en la evolución histórica del mismo.

Una vez aprendida la evolución anual del ICA, proponemos el aprendizaje de su evolución diaria. Para ello repetimos el proceso de entrenamiento, cuyos datos de partida corresponderán a los datos iniciales sin filtrar. En este caso, la generalización de la señal que propone Prophet se muestra a continuación:

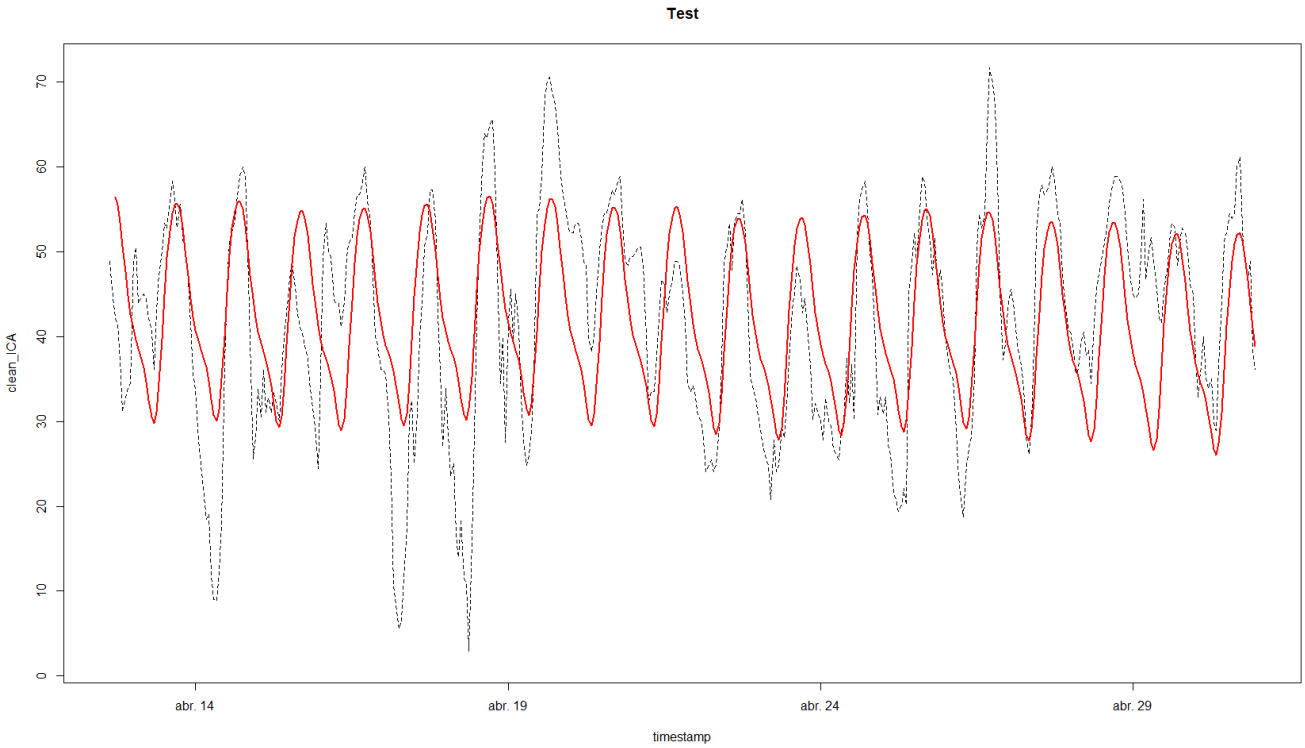


De nuevo, aunque la alta concentración de puntos no permita ver la evolución detallada del ICA, vemos la diferencia significativa entre esta imagen y la obtenida durante el entrenamiento de la evolución anual del ICA. En este caso, a parte de cambiar la información de entrada del algoritmo, se ha modificado el parámetro que especifica la periodicidad de la señal.

Una vez más, validamos el modelo generado con el conjunto de datos de validación, obteniendo la siguiente predicción:



La siguiente imagen muestra con más detalle la predicción realizada por el algoritmo, correspondiente a la última quincena de abril:



Si nos fijamos con detenimiento, vemos como el algoritmo ha detectado y aprendido correctamente la periodicidad de la señal, así como su tendencia. La diferencia el la predicción radica, una vez más, en los términos absolutos del ICA, cuyos valores dependen indudablemente de efectos externos no contemplados en el entrenamiento de la serie temporal.

Resultados

Recordamos que el desarrollo se ha realizado para tres estaciones de control (Escuelas Aguirre, Farolillo y Casa de Campo). A continuación, con el objetivo de no espesar el contenido de esta memoria, se mostrarán en detalle los resultados obtenidos para una de ellas (Casa de Campo), pero el procedimiento de ejecución seguido para las otras dos es análogo. El lector que lo desee, podrá consultar los resultados totales en la herramienta de visualización online desarrollada para mostrar los resultados del proyecto, que será explicada con detalle en el apartado "Representación de la información".

Con el objetivo de evaluar el proceso de entrenamiento realizado, se han calculado dos métricas sobre la predicción realizada para los datos de validación: el error absoluto medio (*mae*, por sus siglas en inglés) y el error cuadrático médio (*mse*, por sus siglas en inglés).

De esta forma, los errores de predicción obtenidos para la predicción anual de cada una de las tres estaciones de control analizadas son los siguientes:

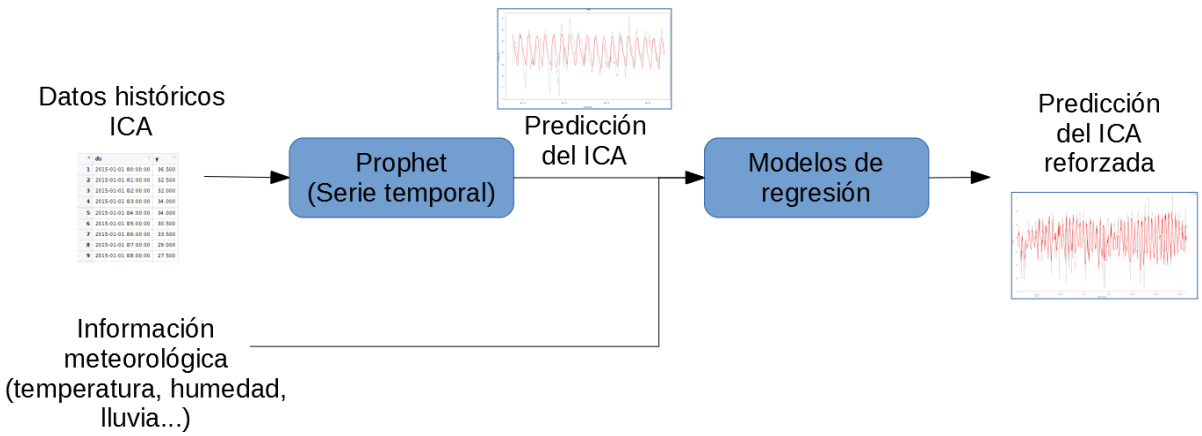
Estación	**MAE**	**MSE**
Casa de Campo	5.88	6.44
Escuelas Aguirre	2.51	3.05
Farolillo	6.76	7.64

Por su parte, los errores de predicción obtenidos para la predicción a nivel horaria de cada una de las tres estaciones de control analizadas son los siguientes:

Estación	MAE	MSE
Casa de Campo	9.03	11.34
Escuelas Aguirre	8.19	10.82
Farolillo	11.12	13.99

Modelos de regresión

Con el objetivo de mejorar la precisión de los resultados de predicción ofrecidos a nivel horario por el modelo anterior, se ha decidido entrenar un segundo modelo, que podríamos denominar "modelo de refuerzo", que tiene en cuenta tanto la salida proporcionada por el algoritmo Prophet, como las variables meteorológicas registaradas de manera horaria. De esta forma, el procedimiento seguido se explica gráficamente en la siguiente imagen:



El objetivo perseguido con este segundo entrenamiento es recoger la predicción ofrecida por el algoritmo Prophet, el cual ha aprendido la tendencia y estacionalidad del ICA a lo largo del tiempo, y modificarlo ligeramente de acuerdo a otros factores determinantes, como son los factores meteorológicos. De esta manera se obtiene una predicción global que mantiene la periodicidad de la serie temporal y a su vez se ajusta, en términos absolutos, a los valores reales del ICA.

Como se ha comentado anteriormente, el objetivo de este segundo modelado consiste en la inclusión de variables que puedan alterar la predicción del ICA, como son las variables meteorológicas. En nuestro caso, tal y cómo se ha explicado en la sección "Información meteorológica", en este proyecto se han descargado los datos meteorológicos históricos desde 2001. Las variables meteorológicas utilizadas para este segundo entrenamiento son las siguientes: *temperatura*, *humedad*, *velocidad del viento*, *presion*, *niebla* y *lluvia*. A parte de las variables anteriores, se ha generado una variable artificial que hemos llamado days_from_last_rain, que representa los días que han pasado, para cada marca temporal, desde el último día que ha llovido.

La API consultada para la descarga de datos meteorológicos solamente ofrece información en el intervalo 07:00 a 20:00, por lo que las mediciones que no pertenezcan a este intervalo serán eliminadas del estudio. Este hecho condiciona el marco temporal de las predicciones realizadas por el modelo final.

Además de la serie temporal predicha por el algoritmo Prophet y los datos meteorológicos, la marca temporal correspondiente a cada medida se ha dividido en las variables *mes*, *día* y *hora*.

Metodología

Una vez establecidas las variables de entrada que alimentarán los modelos, el procedimiento llevado a cabo ha consistido en la separación del conjunto total de datos en un subconjunto de entrenamiento (datos comprendidos entre las fechas 01/01/2015 y 31/12/2017) y un subconjunto de validación (datos posteriores al 01/01/2018). Se ha restringido el periodo de entrenamiento a los años 2015, 2016 y 2017 por considerarse que el volumen de datos es lo suficientemente elevado como para que los algoritmos de regresión generen un modelo preciso.

Se han evaluado tres algoritmos de aprendizaje automático: las redes neuronales, el algoritmo M5P y el algoritmo Random Forest. A continuación se detallan los resultados obtenidos para cada uno de ellos con respecto a la estación de Casa de Campo (los resultados obtenidos para cada una de las tres estaciones se podrán consultar en la herramienta de visualización online):

Redes Neuronales

Las redes neuronales son una técnica de aprendizaje y procesamiento automático inspirada en el funcionamiento del cerebro humano. Podemos definir las redes neuronales como una estructura de procesamiento paralelo masivo constituida por unas unidades muy sencillas (denominadas neuronas), que tienen la capacidad de almacenar conocimiento experimental y ponerla a disposición para su uso.

La arquitectura (o topología) de la red hace referencia a la disposición de las neuronas en la red. Las neuronas se organizan formando capas, de modo que la red neuronal puede consistir en una o más capas de neuronas.

Cada neurona recibe un conjunto de entradas multiplicadas por su interconexión (peso), que son sumados y operados por una función de transferencia (o función de activación) antes de transmitirse a la siguiente capa o como salida de la red.

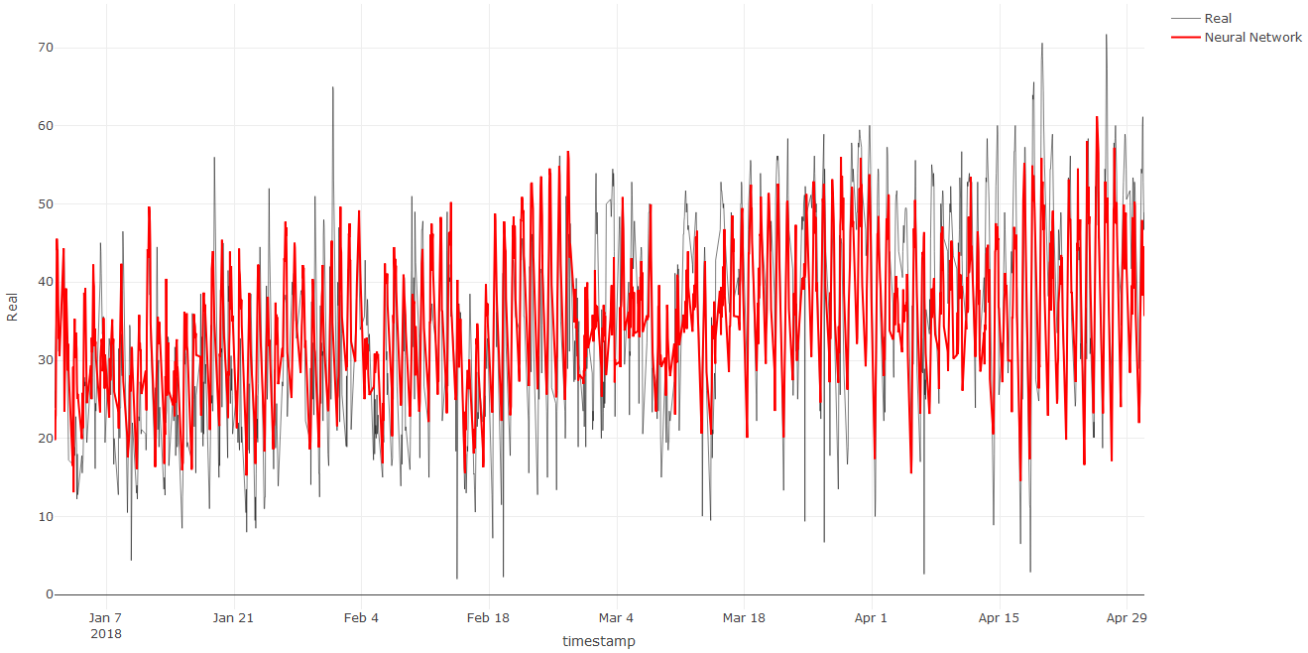
La capa que contiene las salidas de la red se conoce como *capa de salida* y el resto de capas como *capas ocultas.*

Es habitual clasificarlas por su arquitectura. Así nos encontramos con:

- Redes con propagación hacia adelante (feed-forward)
- Redes con retropropagación (back-forward)

En este proyecto se ha entrenado una red neuronal con propagación hacia adelante. En particular se ha entrenado una red neuronal con 15 capas ocultas y un número de iteraciones máximo limitado a 5000. Para ello hemos utilizado el paquete "nnet" de R.

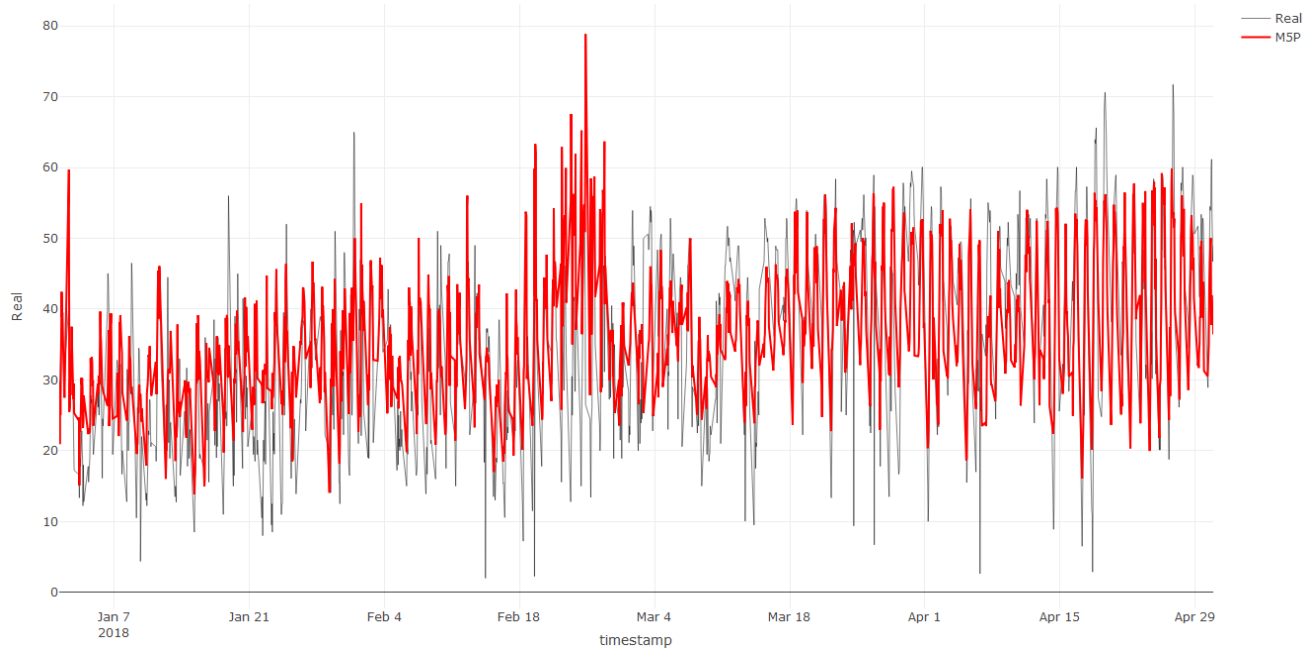
La predicción sobre el conjunto de validación se representa gráficamente en la siguiente imagen:



M5P

El algoritmo M5P se basa en el algoritmo M5 desarrollado por R. Quinlan, sobre el cual Yong Wang realiza una serie de mejoras. La principal característica del algoritmo M5P es que es un método que trata de encontrar un árbol de regresión a partir de instancias de entrenamiento, capaz de predecir correctamente el valor numérico de la clase de una instancia futura. La principal diferencia, y a su vez la principal mejora, con respecto al algoritmo M5 radica en la posibilidad de tener funciones de regresión lineal en cada uno de los nodos del árbol.

La predicción sobre el conjunto de validación se representa gráficamente en la siguiente imagen:



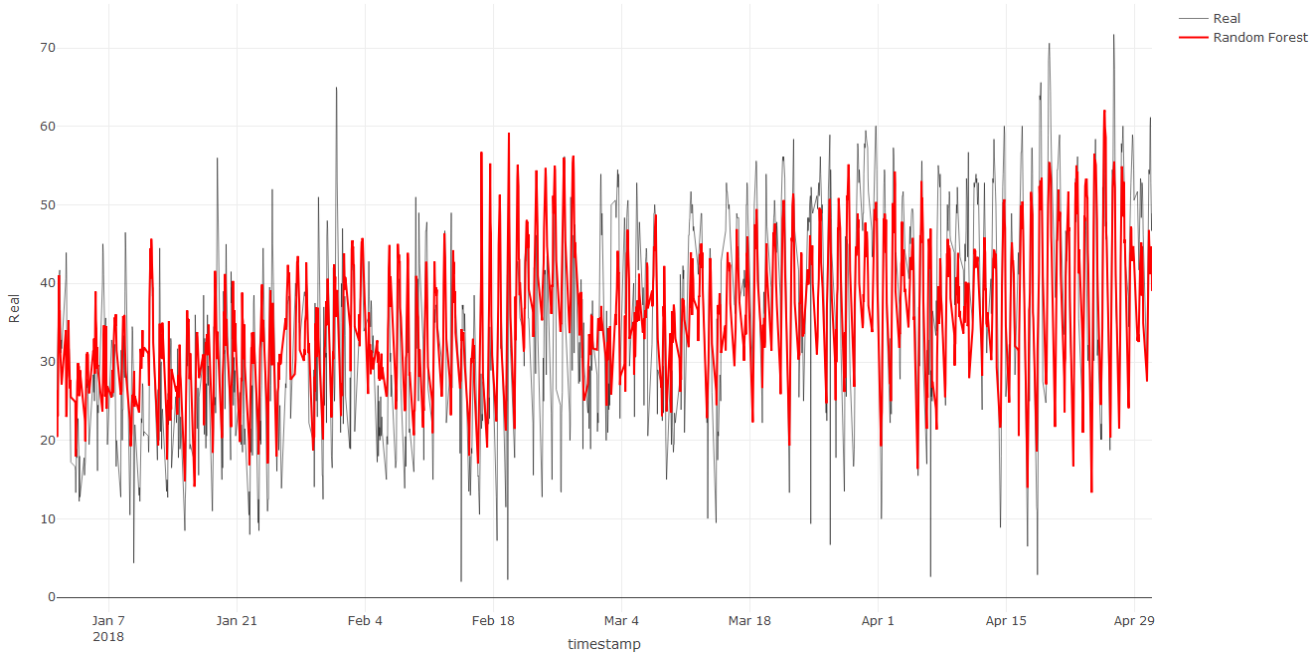
Random Forest

Random Forests es una combinación de árboles de decisión, de tal manera que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos.

Esta técnica de agregación, desarrollada por Leo Breiman, mejora la precisión en la clasificación mediante la incorporación de aleatoriedad en la construcción de cada clasificador (árbol de decisión) individual. Esta aleatorización puede introducirse en la partición del espacio (construcción del árbol), así como en la muestra de entrenamiento. Es una modificación sustancial de bagging que construye una larga colección de árboles no correlacionados y luego los promedia.

En este proyecto se ha utilizado el paquete "randomForest" de R para entrenar el algoritmo, el cual se ha limitado a un máximo de 100 árboles.

La predicción sobre el conjunto de validación se representa gráficamente en la siguiente imagen:



Resultados

Los resultados obtenidos por cada modelo para cada una de las tres estaciones de control analizadas son los siguientes:

Casa de Campo

Los errores obtenidos son los siguientes:

Algoritmo	Métrica	Entrenamiento	Validación
NN	MAE	2.68	7.27
NN	MSE	0.11	0.21
NN	Correlación	-	71%
M5P	MAE	6.33	**7.21**
M5P	MSE	0.25	**0.20**
M5P	Correlación	-	68%
RF	MAE	6.88	7.49
RF	MSE	0.27	0.22
RF	Correlación	-	69%

Tal y como se puede observar en la tabla anterior, a pesar de que el algoritmo M5P no es el algoritmo que mejores resultados de entrenamiento ofrece, si es el algoritmo que mejor generaliza, pues es el que genera los errores de validación más bajos. Por lo tanto, el modelado de predicción para la estación de la Casa de Campo estará formado por el algoritmo M5P entrenado.

Escuelas Aguirre

Los errores obtenidos son los siguientes:

Algoritmo	Métrica	Entrenamiento	Validación
NN	MAE	2.64	**6.67**
NN	MSE	0.08	**0.19**
NN	Correlación	-	46%
M5P	MAE	6.53	7.21
M5P	MSE	0.19	0.21
M5P	Correlación	-	33%
RF	MAE	6.81	6.75
RF	MSE	0.20	0.20
RF	Correlación	-	46%

En este caso, el algoritmo que mejor se ajusta a los datos es la red neuronal, la cual obtiene los mejores resultados tanto en entrenamiento como en validación. Por lo tanto, el modelado de predicción para la estación de la Escuelas Aguirre estará formado por la red neuronal entrenada.

Farolillo

Los errores obtenidos son los siguientes:

Algoritmo	Métrica	Entrenamiento	Validación
NN	MAE	2.65	**7.70**
NN	MSE	0.09	**0.23**
NN	Correlación	-	47%
M5P	MAE	6.57	10.24
M5P	MSE	0.21	0.41
M5P	Correlación	-	18%
RF	MAE	6.84	9.51
RF	MSE	0.22	0.36
RF	Correlación	-	41%

En este último caso, una vez más el algoritmo que mejor se ajusta a los datos es la red neuronal, la cual vuelve a obtener los mejores resultados tanto en entrenamiento como en validación. Por lo tanto, el modelado de predicción para la estación de la Farolillo estará formado por la red neuronal entrenada.

Representación de la información

La representación de los resultados obtenidos en este proyecto se realiza a través de una aplicación web interactiva, accesible de manera online y gratuita, desarrollada con la herramienta *Shiny* (https://shiny.rstudio.com/). *Shiny* es un paquete de RStudio que proporciona un marco para el desarrollo de aplicaciones web interactivas, conocidas como "Shiny apps". Además, esta herramienta proporciona una capacidad reactiva de interacción automática entre las entradas y las salidas.

Una aplicación de Shiny está formada principalmente por dos componentes:

- **UI.R**: este archivo define la interfaz de usuario de la aplicación. Proporciona interactividad tomando como entrada las acciones del usuario y mostrando dinámicamente la salida generada.
- **Server.R**: este archivo define la metodología necesaria para convertir la entrada del usuario en la salida deseada.

Estos dos archivos se pueden fusionar en un archivo común. En este caso, se han desarrollado los componentes ui y server dentro del archivo **app.R**.

En esta sección se detalla la estructura de la aplicación desarrollada, la cual se divide principalmente en tres partes:

- INFO:

Esta pestaña está principalmente diseñada para proporcionar información básica sobre el proyecto a aquellas personas que accedan por primera vez a la herramienta de visualización desarrollada. En ella se especificarán, brevemente, las principales características de este proyecto, como son el objetivo, los orígenes de datos consultados y el contenido fundamental de la aplicación.

- Información histórica:

  Esta pestaña está diseñada para mostrar la evolución histórica de los datos recogidos para la realización de este proyecto. En particular, se muestra gráficamente la evolución temporal del ICA, de los principales contaminantes relacionados con este índice y de las condiciones climatológicas (temperatura, humedad...). Por lo tanto, esta pestaña se centra en la parte de analítica descriptiva de este proyecto.

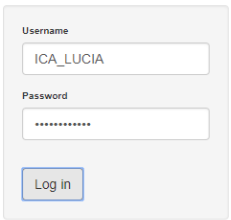
- Predicción del ICA:

  Esta pestaña está diseñada con el objetivo de mostrar la predicción del ICA devuelta por los modelos desarrollados y explicados en la sección "Modelado predictivo". En particular, muestra la predicción del ICA para las próximas 24 horas, para los próximos 7 días y para el resto del año vigente. Por lo tanto, esta pestaña se centra en la parte de analítica predictiva de este proyecto.

La aplicación desarrollada se encuentra desplegada en la máquina virtual del proyecto, accesible a través de la URL: http://visa-lfs.westeurope.cloudapp.azure.com:8080/. Las principales características de esta máquina virtual se explicarán de manera más detallada en la siguiente sección "Arquitectura de producción".

Con el objetivo de controlar el acceso a la aplicación se ha desarrollado un módulo de "login", en el cuál se deben introducir las credenciales adecuadas para poder acceder al contenido. En nuestro caso, las credenciales establecidas son las siguientes:

- **Username**: ICA_LUCIA
- **Password**: VISA_tfm2018



El esquema fundamental del script **app.R** se se detalla a continuación:

- Declaración de dependencias: inclusión de los paquetes necesarios para la ejecución del script. En nuestro caso los paquetes necesarios son los siguientes:
    - shiny y shinyjs: paquetes para el desarrollo de aplicaciones Shiny.
    - data.table: paquete para la importación eficiente de archivos de gran extensión.
    - readr: paquete para la lectura de ficheros .csv
    - lubridate: paquete que permite la manipulación y transformación de fechas.
    - ggplot2: paquete principal de representación gráfica.
    - plotly: paquete que extiende los gráficos de ggplot2 a gráficos web interactivos.
    - plotrix: paquete con diferentes funciones adicionales de representación gráfica.
    - RColorBrewer: paquete con funciones de coloreado para representaciones gráficas.
    - plyr y dplyr: paquetes con funciones para el procesado de data frames.
    - forecast: paquete empleado para el suavizado de series temporales.
    - markdown: paquete empleado para añadir a la aplicación archivos en formato markdown.

      Para una información más detallada sobre los paquetes anteriores, el lector puede acudir al repositorio principal de paquetes de R, a través de la URL: https://cran.r-project.org.

- Inicialización de variables: declaración de variables que se van a utilizar en determinados puntos del programa, como por ejemplo las credenciales, así como la lectura de archivos cuyo contenido se va a visualizar a través la herramienta. En particular, en este punto se cargan los ficheros que contienen información histórica (ICA, contaminantes y condiciones meteorológicas) y la predicción anual del ICA.

  Debido al caracter dinámico de la aplicación, la información relativa a la predicción del ICA para las próximas 24 horas y para los próximos 7 días se cargará en la parte reactiva del programa, lo cual permitirá actualizar el contenido de manera continua.

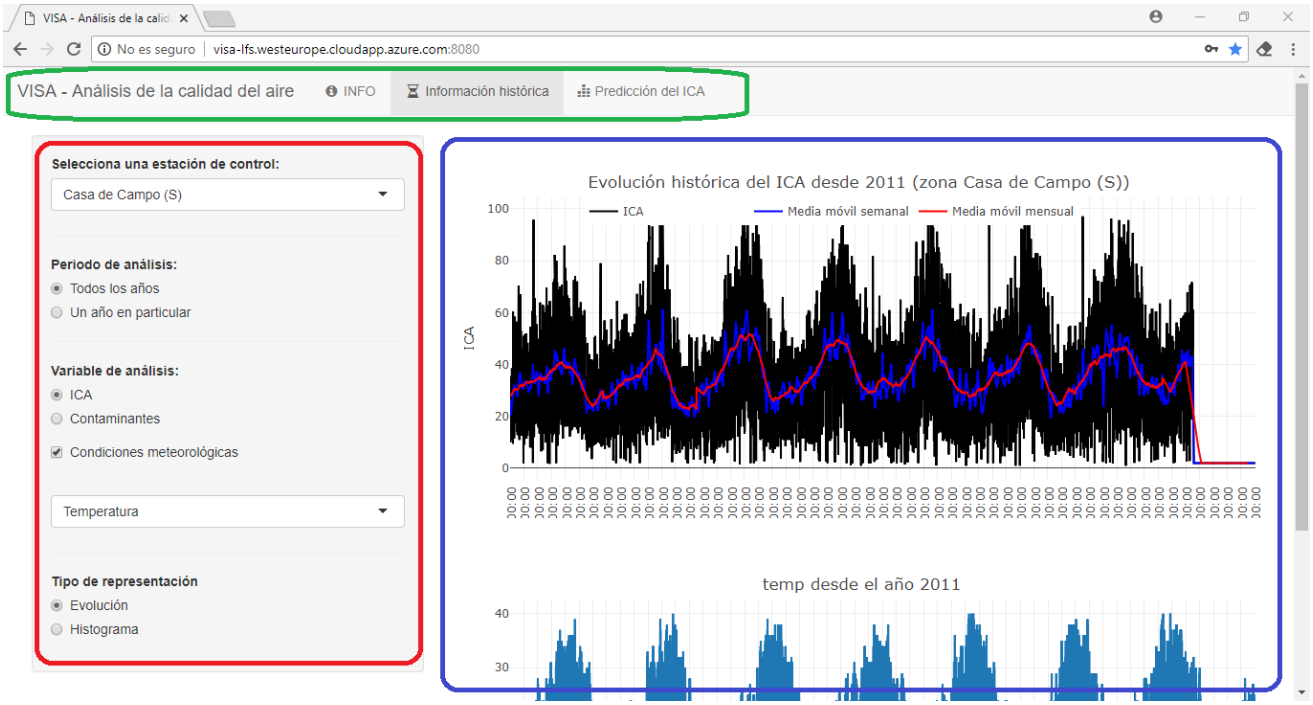
- User interface (UI): tal y como se adelantó anteriormente, en esta sección se define la interfaz de usuario de la aplicación. En particular, se define la interfaz de usuario de las tres pestañas desarrolladas: INFO, Información histórica y Predicción del ICA. Además de lo anterior, se define la interfaz gráfica de la ventana de "login", en donde el usuario debe introducir las credenciales que le permitan acceder a la aplicación. Debido a lo anterior, el UI del script se divide en dos partes: ui1 y ui2, definidas directamente sobre el código a continuación:

```
ui1 <- function(){
  tagList(
    div(id = "login",
        wellPanel(textInput("userName", "Username"),
                  passwordInput("passwd", "Password"),
                  br(),actionButton("Login", "Log in"))),
    tags$style(type="text/css", "#login {font-size:10px;   text-align:
left;position:absolute;top: 40%;left: 50%;margin-top: -100px;margin-left: -150px;}")
)}
```

```
ui2 <- function(){
  navbarPage("VISA - Análisis de la calidad del aire",
             # Pestaña INFO:
             tabPanel("INFO", icon = icon("info-circle"),
                      fluidRow(
                        includeMarkdown("info.md")
                      )
             ),
             # Pestaña Información histórica
             tabPanel("Información histórica", icon = icon("hourglass-half"),
                      fluidPage(
                        sidebarLayout(
                          sidebarPanel(

                            # Desplegable con las estaciones de control
                            selectInput(inputId = "estacion_historico",
                                        label = "Selecciona una estación de control:",
                                        choices = c("Casa de Campo (S)",
                                                    "Escuelas Aguirre (UT)",
                                                    "Farolillo (UF)"),
                                        selected = "Casa de Campo (S)"),
                            hr(),

                            # Selección del periodo de análisis
                            radioButtons("periodo_seleccionado", "Periodo de análisis:",
                                         c("Todos los años" = "todos",
                                           "Un año en particular" = "año_particular"),
                                         selected = "todos"),

                            # Si se selecciona "Un año en particular", desplegable con años
                            uiOutput(outputId = "años"),
                            hr(),

                            # Selección de la variable de análisis
                            radioButtons("hist_variable", "Variable de análisis:",
                                         c("ICA" = "hist_ica",
                                           "Contaminantes" = "hist_cont")),

                            # Si se selecciona "Contaminantes", desplegable con listado
                            uiOutput(outputId = "contaminante"),

                            # Opción de visualización de condiciones climatológicas
                            checkboxInput("hist_tiempo", "Condiciones meteorológicas", TRUE),

                            # Si se selecciona la opción anterior, listado de condiciones meteo.
                            uiOutput(outputId = "meteorologia"),
                            hr(),

                            # Selección del tipo de representación
                            radioButtons("hist_representacion", "Tipo de representación",
                                         c("Evolución" = "hist_med",
                                           "Histograma" = "hist_hist"))
                          ),
```

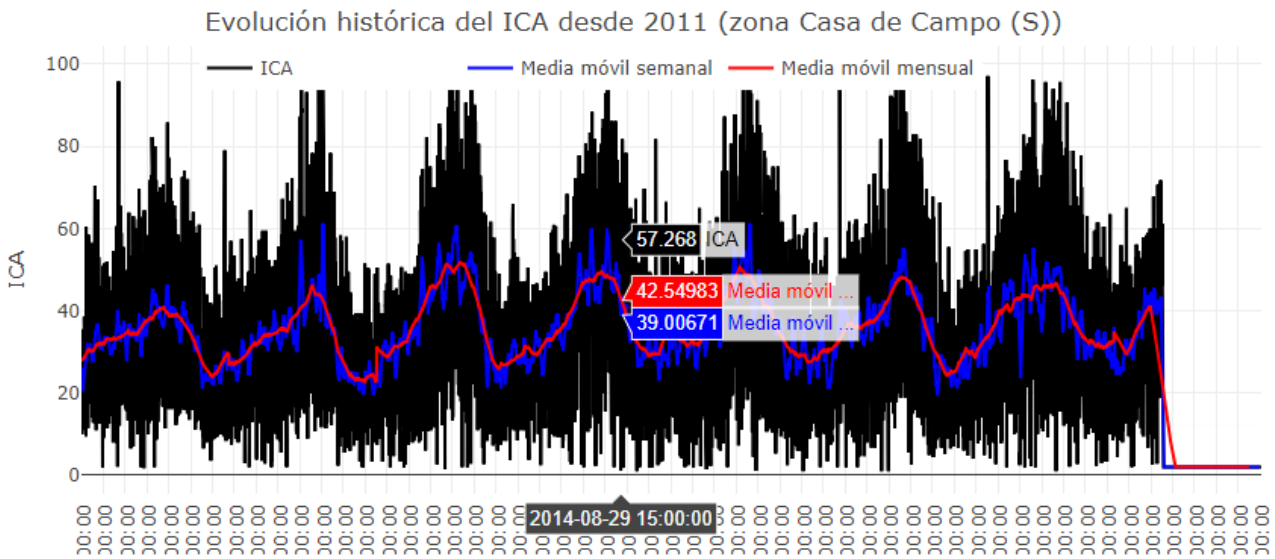
```
                        mainPanel(
                          plotlyOutput("plot_hist"),   # Gráfico superior
                          br(),
                          plotlyOutput("plot_tiempo")  # Gráfico inferior
                        )
                      )
                    )),

            # Pestaña Predicción del ICA
            tabPanel("Predicción del ICA ", icon = icon("equalizer", lib = "glyphicon"),
                     fluidPage(
                       sidebarLayout(
                         sidebarPanel(

                           # Desplegable con las estaciones de control
                           h4("Estación de control:"),
                           selectInput(inputId = "estacion_prediccion",
                                       label = "Selecciona una estación en particular:",
                                       choices = c("Casa de Campo (S)",
                                                   "Escuelas Aguirre (UT)",
                                                   "Farolillo (UF)"),
                                       selected = "Casa de Campo (S)"),
                           hr(),

                           # Selección de la predicción que se desea consultar
                           h4("Predicción precisa (modelo reforzado):"),
                           radioButtons("forecast_reforzado", "",
                                        c("24 horas" = "forecast_24",
                                          "7 dias" = "forecast_7")),
                           hr(),

                           # Opción de visualización de la predicción para el resto del año
                           h4("Predicción estimada (serie temporal):"),
                           checkboxInput("forecast_año", "Año completo", FALSE)
                         ),
                         mainPanel(
                           plotlyOutput("plot_forecast")   # Gráfico con la predicción
                         )
                       )
                     )
            )
       )
}

ui = (htmlOutput("page"))
```

De este modo, a partir del código anterior, la aplicación tendrá la interfaz gráfica que muestra la siguiente imagen:

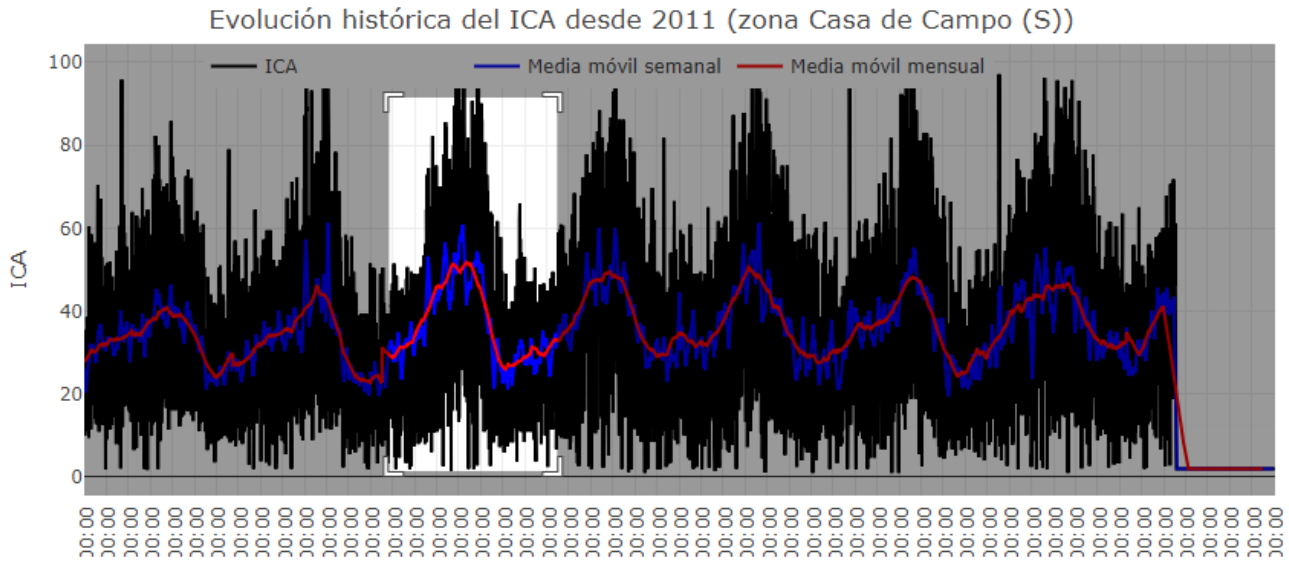


En la parte superior (en verde) podemos ver el menú de pestañas de la aplicación; en la parte izquierda (en rojo) se encuentra el *sidebar*, que es el panel en el que el usuario encuentra toda la gama de opciones con las que interactuar; y finalmente, en la parte de la derecha (en azul) se encuentra el *mainPanel* que es la zona en la que se muestran gráficamente los resultados. Cabe destacar que todas las gráficas representadas en la herramienta, además de reactivas, son interactivas, es decir, permiten una interacción directa con el usuario. Para ello basta con situar el cursor del ratón encima de ellas, mostrándose los valores correspondientes a ese punto de la gráfica, se puede hacer zoom, quitar o añadir variables, etc. A modo de ejemplo, las siguientes imágenes muestran algunas variantes de interacción que nos permiten los gráficos desarrollados:

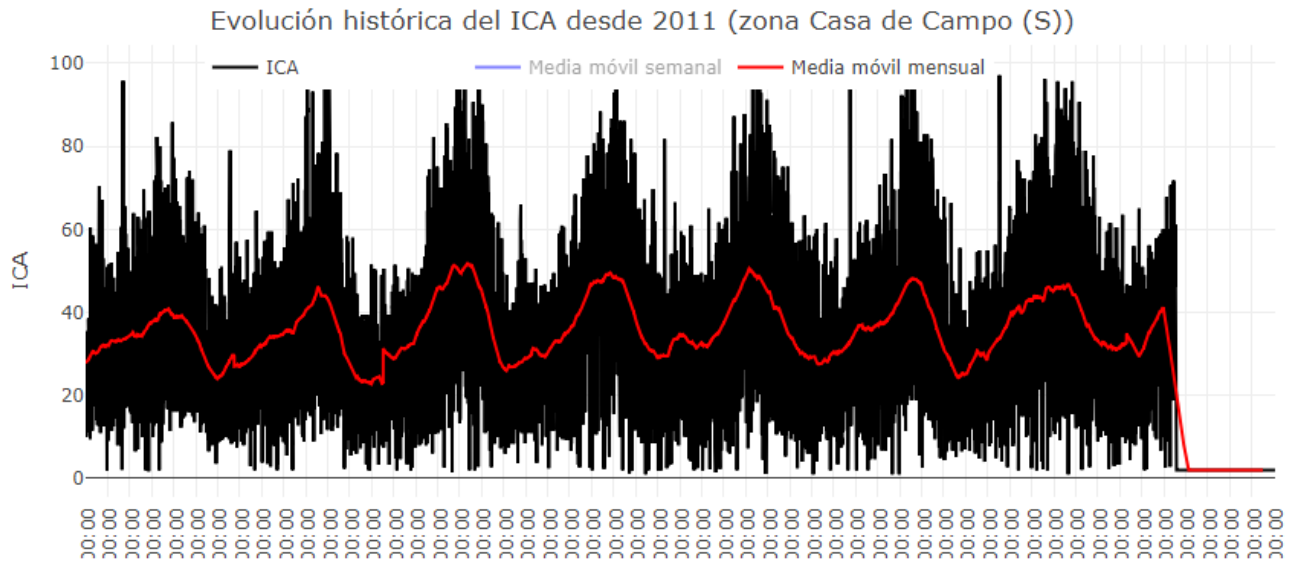
a) Detalle de los valores numéricos representados en el gráfico:



b) Zoom de un periodo en particular:



c) Reducción de variables:



- Server: en esta sección se define la metodología necesaria para convertir la entrada del usuario en la salida deseada. En particular, se define el contenido relacionado con cada una de las variables definidas en el interfaz de usuario (UI).

```
server = (function (input, output, session) {

  USER <- reactiveValues(Logged = Logged)
  # Aqui se declaran las variables globales
  width_images = 1200
  height_images = 600

  # Esperando el login...
  observe({
    if(USER$Logged == FALSE) {
      if (!is.null(input$Login)) {
        if (input$Login > 0) {
```

```
          Username <- isolate(input$userName)
          Password <- isolate(input$passwd)
          Id.username <- which(my_username == Username)
          Id.password <- which(my_password == Password)
          if (length(Id.username) > 0 & length(Id.password) > 0) {
            if (Id.username == Id.password) {
              USER$Logged <- TRUE
            }
          }
        }
      }
    }
  })

  observe({
    if(USER$Logged == FALSE) {
      output$page <- renderUI({
        div(class="outer",do.call(bootstrapPage,c(ui1())))  # Pantalla de login
      })
    }

    # Si se ha producido un login correcto:
    if(USER$Logged == TRUE){
      output$page <- renderUI({
        div(class="outer",do.call(bootstrapPage,c(ui2())))  # Pantalla de la aplicación
      })


      # Contenido de la aplicacion:
      #----------------------------

      # Desplegable listado contaminantes
      output$contaminante <- renderUI({
        if(input$hist_variable == "hist_cont"){
          selectInput(inputId = "sel_contaminante",
                      label = "Selecciona un contaminante en particular:",
                      choices = c("SO2", "CO", "NO2", "PM10", "O3"),
                      selected = "SO2", multiple=FALSE)
        }else{
          return(NULL)
        }
      })

      # Desplegable listado condiciones climatologicas
      output$meteorologia <- renderUI({
        if(input$hist_tiempo == TRUE){
          selectInput(inputId = "sel_meteorologia",
                      label = "",
                      choices = c("Temperatura" = "temp", "Humedad" = "hum", "Presión" =
"presion", "Lluvia" = "rain"),
                      selected = "temp", multiple=FALSE)
        }else{
          return(NULL)
```

```
    }
  })

  # Desplegable listado de años
  output$años <- renderUI({
    if(input$periodo_seleccionado == "año_particular"){
      selectInput(inputId = "años",
                  label = "Selecciona un año en particular:",
                  choices = seq(2001,2018,1),
                  selected = 2018, multiple=FALSE)
    }else{
      return(NULL)
    }
  })

  # Historico
  #-----------------------------------------
  observeEvent(input$periodo_seleccionado,{
    observeEvent(input$hist_variable,{
      observeEvent(input$estacion_historico,{
        observeEvent(input$hist_representacion,{
          observeEvent(input$hist_tiempo,{

            if(input$estacion_historico == "Casa de Campo (S)"){
              est_hist = "Casa de campo_28079024"
              est_hist_code = 28079024
            }
            if(input$estacion_historico == "Escuelas Aguirre (UT)"){
              est_hist = "Escuelas Aguirre_28079008"
              est_hist_code = 28079008
            }
            if(input$estacion_historico == "Farolillo (UF)"){
              est_hist = "Farolillo_28079018"
              est_hist_code = 28079018
            }

            #######
            # ICA #
            #######
            if(input$hist_variable == "hist_ica"){
              if(input$periodo_seleccionado == "todos"){   # Todos los años
                hist_ica_plot = hist_ica[which(hist_ica$cod_est==est_hist_code),

c("timestamp","anyo","clean_ICA","ICA_ma","ICA_ma60")]

                if(input$hist_representacion == "hist_med"){   # Curva media
                  output$plot_hist = renderPlotly({
                    plot_ly(hist_ica_plot, x = ~timestamp, y = ~clean_ICA, type = "scatter",
mode = "lines",
                            name = "ICA", color = I('black')) %>%
                      add_trace(y = ~ICA_ma, mode = 'lines', name = "Media móvil semanal",
                                color = I('blue')) %>%
                      add_trace(y = ~ICA_ma60, mode = 'lines', name = "Media móvil mensual",
```

```
                                    color = I('red')) %>%
                        layout(margin = list(t=65, pad=0),
                               title = paste0("\nEvolución histórica del ICA desde 2011 (zona
",input$estacion_historico,")"),
                               xaxis = list(title = "", tickangle = 270),
                               yaxis = list(title = "ICA", range =
c(0,max(hist_ica_plot$clean_ICA)*1.2)),
                               hovermode = "FALSE",
                               legend = list(x = 0.1, y = 1, orientation = 'h'))
                      })
                    }
                    if(input$hist_representacion == "hist_hist"){   # Histograma
                      hist_ica_hist = aggregate(hist_ica_plot$clean_ICA, by =
list(hist_ica_plot$anyo), mean, na.rm=T)
                      colnames(hist_ica_hist) = c("timestamp","clean_ICA")
                      output$plot_hist = renderPlotly({
                        plot_ly(hist_ica_hist, x = ~timestamp, y = ~clean_ICA, type = "bar") %>%
                          layout(margin = list(t=65, pad=0),
                                 title = paste0("\nEvolución histórica del ICA agregado desde
2011 (zona ",input$estacion_historico,")"),
                                 xaxis = list(title = "", tickangle = 270),
                                 yaxis = list(title = "ICA"),
                                 hovermode = "FALSE")
                      })
                    }
                  }else{   # Un año en particular
                    observeEvent(input$años,{
                      if(!is.null(input$años)){
                        hist_ica_plot = hist_ica[which(hist_ica$cod_est==est_hist_code &
hist_ica$anyo==input$años),

c("timestamp","anyo","clean_ICA","ICA_ma","ICA_ma60")]

                        if(input$hist_representacion == "hist_med"){   # Curva media
                          output$plot_hist = renderPlotly({
                            plot_ly(hist_ica_plot, x = ~timestamp, y = ~clean_ICA, type =
"scatter", mode = "lines",
                                    name = "ICA", color = I('black')) %>%
                              add_trace(y = ~ICA_ma, mode = 'lines', name = "Media móvil
semanal",
                                        color = I('blue')) %>%
                              add_trace(y = ~ICA_ma60, mode = 'lines', name = "Media móvil
mensual",
                                        color = I('red')) %>%
                              layout(margin = list(t=65, pad=0),
                                     title = paste0("\nEvolución histórica del ICA en el año
",input$años," (zona ",input$estacion_historico,")"),
                                     xaxis = list(title = "", tickangle = 270),
                                     yaxis = list(title = "ICA", range =
c(0,max(hist_ica_plot$clean_ICA)*1.2)),
                                     hovermode = "FALSE",
                                     legend = list(x = 0.1, y = 1, orientation = 'h'))
                          })
```

```
                        }
                        if(input$hist_representacion == "hist_hist"){   # Histograma
                          hist_ica_hist = aggregate(hist_ica_plot$clean_ICA, by =
list(hist_ica_plot$anyo), mean, na.rm=T)
                          colnames(hist_ica_hist) = c("timestamp","clean_ICA")
                          output$plot_hist = renderPlotly({
                            plot_ly(hist_ica_hist, x = ~timestamp, y = ~clean_ICA, type = "bar")
%>%
                              layout(margin = list(t=65, pad=0),
                                     title = paste0("\nEvolución histórica del ICA en el año
",input$años," (zona ",input$estacion_historico,")"),
                                     xaxis = list(title = "", tickangle = 270),
                                     yaxis = list(title = "ICA"),
                                     hovermode = "FALSE")
                          })
                        }
                      }
                    })
                  }
                }

                ################
                # CONTAMINANTE #
                ################
                if(input$hist_variable == "hist_cont"){
                  observeEvent(input$sel_contaminante,{
                    if(!is.null(input$sel_contaminante)){
                      col_sel_contaminante =
as.numeric(which(colnames(hist_ica)==input$sel_contaminante))

                      if(input$periodo_seleccionado == "todos"){   # Todos los años
                        hist_cont_plot =
hist_ica[which(hist_ica$cod_est==est_hist_code),c(1,11,col_sel_contaminante)]
                        colnames(hist_cont_plot) = c("timestamp","anyo","contaminante")

                        if(input$hist_representacion == "hist_med"){   # Curva media
                          output$plot_hist = renderPlotly({
                            plot_ly(hist_cont_plot, x = ~timestamp, y = ~contaminante, type =
"scatter", mode = "lines") %>%
                              layout(margin = list(t=65, pad=0),
                                     title = paste0("\nEvolución histórica del
",input$sel_contaminante," desde 2011 (zona ",input$estacion_historico,")"),
                                     xaxis = list(title = "", tickangle = 270),
                                     yaxis = list(title = input$sel_contaminante),
                                     hovermode = "FALSE")
                          })
                        }
                        if(input$hist_representacion == "hist_hist"){   # Histograma
                          hist_cont_hist = aggregate(hist_cont_plot$contaminante, by =
list(hist_cont_plot$anyo), mean, na.rm=T)
                          colnames(hist_cont_hist) = c("timestamp","contaminante")
                          output$plot_hist = renderPlotly({
```

```
                            plot_ly(hist_cont_hist, x = ~timestamp, y = ~contaminante, type =
"bar") %>%
                              layout(margin = list(t=65, pad=0),
                                     title = paste0("\nEvolución histórica del
",input$sel_contaminante," desde 2011 (zona ",input$estacion_historico,")"),
                                     xaxis = list(title = "", tickangle = 270),
                                     yaxis = list(title = input$sel_contaminante),
                                     hovermode = "FALSE")
                          })
                        }
                      }else{   # Un año en particular
                        observeEvent(input$años,{
                          if(!is.null(input$años)){
                            hist_cont_plot = hist_ica[which(hist_ica$cod_est==est_hist_code &
hist_ica$anyo==input$años),

 c("timestamp","anyo",input$sel_contaminante)]
                            colnames(hist_cont_plot) = c("timestamp","anyo","contaminante")

                            if(input$hist_representacion == "hist_med"){   # Curva media
                              output$plot_hist = renderPlotly({
                                plot_ly(hist_cont_plot, x = ~timestamp, y = ~contaminante, type
= "scatter", mode = "lines") %>%
                                  layout(margin = list(t=65, pad=0),
                                         title = paste0("\nEvolución histórica del
",input$sel_contaminante," en el año ",input$años," (zona ",input$estacion_historico,")"),
                                         xaxis = list(title = "", tickangle = 270),
                                         yaxis = list(title = input$sel_contaminante),
                                         hovermode = "FALSE")
                              })
                            }
                            if(input$hist_representacion == "hist_hist"){   # Histograma
                              hist_cont_hist = aggregate(hist_cont_plot$contaminante, by =
list(hist_cont_plot$anyo), mean, na.rm=T)
                              colnames(hist_cont_hist) = c("timestamp","contaminante")
                              output$plot_hist = renderPlotly({
                                plot_ly(hist_cont_hist, x = ~timestamp, y = ~contaminante, type
= "bar") %>%
                                  layout(margin = list(t=65, pad=0),
                                         title = paste0("\nEvolución histórica del
",input$sel_contaminante," en el año ",input$años," (zona ",input$estacion_historico,")"),
                                         xaxis = list(title = "", tickangle = 270),
                                         yaxis = list(title = input$sel_contaminante),
                                         hovermode = "FALSE")
                              })
                            }
                          }
                        })
                      }
                    }
                  })
                }
```

```
                ########################
                # DATOS METEOROLOGICOS #
                ########################
                if(input$hist_tiempo == TRUE){
                  observeEvent(input$sel_meteorologia,{
                    if(!is.null(input$sel_meteorologia)){

                      if(input$periodo_seleccionado == "todos"){   # Todos los años
                        hist_tiempo_plot = hist_tiempo[,
c("timestamp","anyo",input$sel_meteorologia)]
                        colnames(hist_tiempo_plot) = c("timestamp","anyo","meteorologia")
                        hist_tiempo_plot$meteorologia =
as.numeric(hist_tiempo_plot$meteorologia)

                        if(input$hist_representacion == "hist_med"){   # Curva media
                          output$plot_tiempo = renderPlotly({
                            plot_ly(hist_tiempo_plot, x = ~timestamp, y = ~meteorologia, type =
"scatter", mode = "lines") %>%
                              layout(margin = list(t=65, pad=0),
                                     title = paste0("\n ",input$sel_meteorologia," desde el año
2011"),
                                     xaxis = list(title = "", tickangle = 270),
                                     yaxis = list(title = input$sel_meteorologia),
                                     hovermode = "FALSE")
                          })
                        }
                        if(input$hist_representacion == "hist_hist"){   # Histograma
                          hist_meteo_hist = aggregate(hist_tiempo_plot$meteorologia, by =
list(hist_tiempo_plot$anyo), mean, na.rm=T)
                          colnames(hist_meteo_hist) = c("timestamp","meteorologia")
                          output$plot_tiempo = renderPlotly({
                            plot_ly(hist_meteo_hist, x = ~timestamp, y = ~meteorologia, type =
"bar") %>%
                              layout(margin = list(t=65, pad=0),
                                     title = paste0("\nEvolución histórica del
",input$sel_meteorologia,"  desde el año 2011"),
                                     xaxis = list(title = "", tickangle = 270),
                                     yaxis = list(title = input$sel_meteorologia),
                                     hovermode = "FALSE")
                          })
                        }
                      }else{   # Un año en particular
                        observeEvent(input$años,{
                          if(!is.null(input$años)){
                            hist_tiempo_plot = hist_tiempo[which(hist_tiempo$anyo ==
input$años), c("timestamp","anyo",input$sel_meteorologia)]
                            colnames(hist_tiempo_plot) = c("timestamp","anyo","meteorologia")
                            hist_tiempo_plot$meteorologia =
as.numeric(hist_tiempo_plot$meteorologia)

                            if(input$hist_representacion == "hist_med"){   # Curva media
                              output$plot_tiempo = renderPlotly({
```

```
                                plot_ly(hist_tiempo_plot, x = ~timestamp, y = ~meteorologia,
type = "scatter", mode = "lines") %>%
                                  layout(margin = list(t=65, pad=0),
                                         title = paste0("\n ",input$sel_meteorologia," en el año
",input$años),
                                         xaxis = list(title = "", tickangle = 270),
                                         yaxis = list(title = input$sel_meteorologia),
                                         hovermode = "FALSE")
                              })
                            }
                            if(input$hist_representacion == "hist_hist"){   # Histograma
                              hist_meteo_hist = aggregate(hist_tiempo_plot$meteorologia, by =
list(hist_tiempo_plot$anyo), mean, na.rm=T)
                              colnames(hist_meteo_hist) = c("timestamp","meteorologia")
                              output$plot_tiempo = renderPlotly({
                                plot_ly(hist_meteo_hist, x = ~timestamp, y = ~meteorologia, type
= "bar") %>%
                                  layout(margin = list(t=65, pad=0),
                                         title = paste0("\nEvolución histórica del
",input$sel_meteorologia," en el año ",input$años),
                                         xaxis = list(title = "", tickangle = 270),
                                         yaxis = list(title = input$sel_meteorologia),
                                         hovermode = "FALSE")
                              })
                            }
                          }
                        })
                      }
                    }
                  })
                }


              })  # input$hist_tiempo
            })  # input$hist_representacion
          })  # input$estacion_historico
        })  # input$hist_variable
      })  # input$periodo_seleccionado


      # Prediccion:
      #----------------------------------------

      observeEvent(input$forecast_año,{
        observeEvent(input$estacion_prediccion,{
          if(input$estacion_prediccion == "Escuelas Aguirre (UT)"){
            est_pred = "Escuelas Aguirre_28079008"
            est_pred_code = 28079008
          }else if(input$estacion_prediccion == "Farolillo (UF)"){
            est_pred = "Farolillo_28079018"
            est_pred_code = 28079018
          }else{
            est_pred = "Casa de campo_28079024"
```

```
            est_pred_code = 28079024
          }

          if(input$forecast_año==TRUE){   # Plot prediccion anual
              if(est_pred_code == 28079024){
                output$plot_forecast = renderPlotly({
                  plot_ly(forecast_anyo_28079024, x = ~ds, y = ~yhat, type = "scatter", mode =
"lines") %>%
                    layout(margin = list(t=65, pad=0),
                           title = "\nPredicción del ICA para el resto del año (zona Casa de
Campo (S))",
                           xaxis = list(title = "", tickangle = 270),
                           yaxis = list(title = "ICA", range =
c(0,max(forecast_anyo_28079024$yhat)*1.5)),
                           hovermode = "FALSE")
                })
              }
              if(est_pred_code == 28079008){
                output$plot_forecast = renderPlotly({
                  plot_ly(forecast_anyo_28079008, x = ~ds, y = ~yhat, type = "scatter", mode =
"lines") %>%
                    layout(margin = list(t=65, pad=0),
                           title = "\nPredicción del ICA para el resto del año (zona Escuelas
Aguirre (UT))",
                           xaxis = list(title = "", tickangle = 270),
                           yaxis = list(title = "ICA", range =
c(0,max(forecast_anyo_28079008$yhat)*1.5)),
                           hovermode = "FALSE")
                })
              }
              if(est_pred_code == 28079018){
                output$plot_forecast = renderPlotly({
                  plot_ly(forecast_anyo_28079018, x = ~ds, y = ~yhat, type = "scatter", mode =
"lines") %>%
                    layout(margin = list(t=65, pad=0),
                           title = "\nPredicción del ICA para el resto del año (zona Farolillo
(UF))",
                           xaxis = list(title = "", tickangle = 270),
                           yaxis = list(title = "ICA", range =
c(0,max(forecast_anyo_28079018$yhat)*1.5)),
                           hovermode = "FALSE")
                })
              }
          }else{   # Prediccion precisa
            observeEvent(input$forecast_reforzado,{
              if(input$forecast_reforzado == "forecast_24"){   # Prediccion 24h
                forecast = fread(paste0("../Calidad del aire/Análisis
predictivo/Resultados/",est_pred,"/forecast_24.csv"))

                output$plot_forecast = renderPlotly({
                  plot_ly(forecast, x = ~timestamp, y = ~forecast, type = "scatter", mode =
"lines") %>%
                    layout(width = 800, height = 500, margin = list(t=65, pad=0),
```

```
                            title = paste0("\nPredicción del ICA para las próximas 24 horas (zona
",input$estacion_prediccion,")"),
                            xaxis = list(title = "", tickangle = 270),
                            yaxis = list(title = "ICA", range = c(0,max(forecast$forecast)*1.5)),
                            hovermode = "FALSE")
                  })
                }
                if(input$forecast_reforzado == "forecast_7"){   # Prediccion 7d
                  forecast = fread(paste0("../Calidad del aire/Análisis
predictivo/Resultados/",est_pred,"/forecast_7.csv"))

                  output$plot_forecast = renderPlotly({
                    plot_ly(forecast, x = ~timestamp, y = ~forecast, type = "scatter", mode =
"lines") %>%
                      layout(width = 800, height = 500, margin = list(t=65, pad=0),
                            title = paste0("\nPredicción del ICA para los próximos 7 días  (zona
",input$estacion_prediccion,")"),
                            xaxis = list(title = "", tickangle = 270),
                            yaxis = list(title = "ICA", range = c(0,max(forecast$forecast)*1.5)),
                            hovermode = "FALSE")
                  })
                }
              })
            }
          })  # input$estacion_prediccion
        })  # input$forecast_año


    }  # USER$Logged
  })
})
```

- Finalmente, para desplegar la aplicación en el servidor seleccionado, debemos especificar el host donde la queremos desplegar (en Shiny se debe especificar el host 0.0.0.0 para permitir el acceso remoto desde cualquier equipo) y el puerto de salida (en nuestro caso hemos habilitado el puerto 8080). Para ello se ejecuta la siguiente instrucción:

```
runApp(list(ui = ui, server = server), port = 8080, host = "0.0.0.0", launch.browser=TRUE)
```

Arquitectura de producción

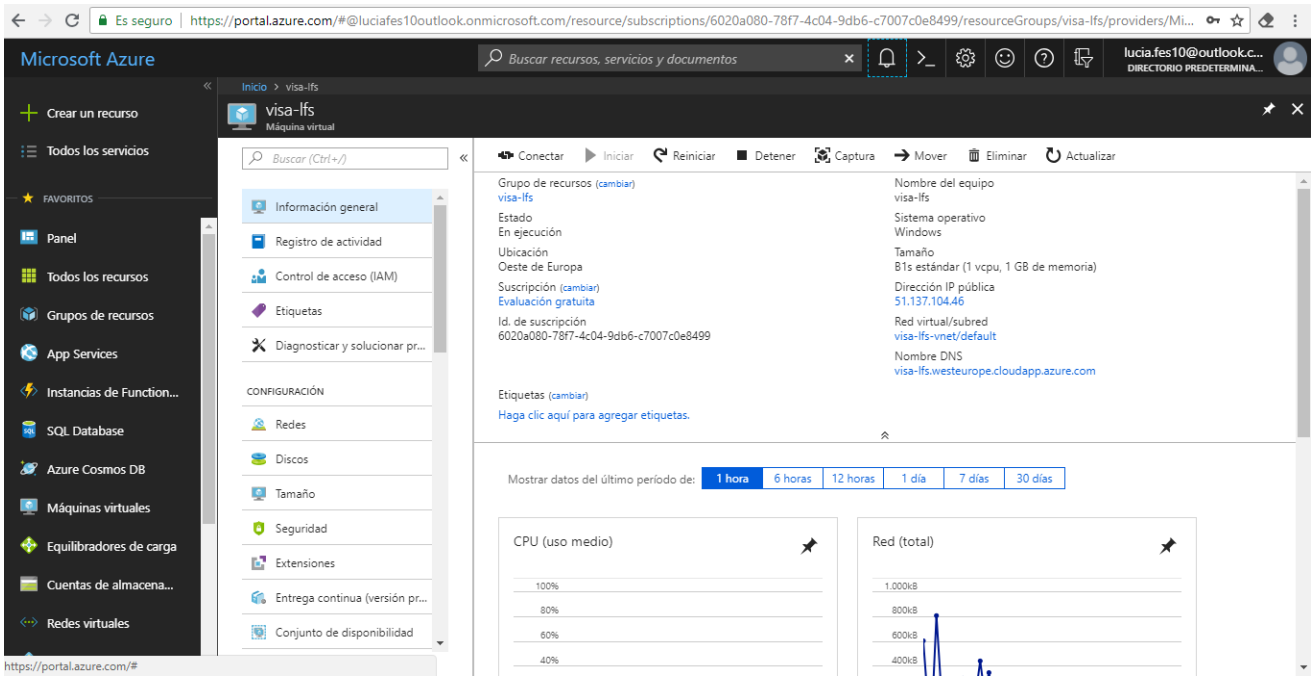
El desarrollo de este proyecto conlleva la ejecución diária de una serie de procesos que permiten la descarga de datos necesarios para la predicción del ICA (por ejemplo los datos meteorológicos). Además, los datos descargados deben procesarse adecuadamente antes de su utilización, como se ha explicado en a lo largo de esta memoria. De este modo, con el objetivo de ejecutar de manera automatizada cada uno de los scripts de descarga y preprocesado desarrollados, así como para mantener ejecutada de manera

continua la aplicación web de visualización desarrollada, se ha contratado una máquina virtual de Microsoft Azure.

Microsoft ofrece una amplia gama de servicios, entre los que se encuentran las máquinas virtuales. Permiten la elección de software y sistema operativo , así como características de requisitos de almacenamiento, RAM y vCPU (núcleo).

Para este proyecto se optó por la adquisición de una máquina virtual sencilla, ya que no se requiere de una gran infrastructura para ejecutar los scripts anteriormente mencionados. En particular, las principales características de la máquina virtual contratada son las siguientes:

- Modelo BS1: Pensada para cargas de trabajo que normalmente no usan mucha CPU pero que, a veces, necesitan repentinamente un rendimiento de CPU mucho más alto cuando la demanda aumenta. Estas cargas de trabajo no necesitan usar toda la CPU siempre, sino que en algunas ocasiones tienen que aumentar el rendimiento para completar algunas tareas rápidamente. Muchas aplicaciones, como los servidores de desarrollo y pruebas, los servidores web con poco tráfico, las bases de datos pequeñas, los servidores para pruebas de concepto, los servidores de compilación y los repositorios de código, encajan con este modelo.
- Sistema operativo: Windows
- 1 core CPU.
- 1 GB de RAM.
- Coste: 0.015 €/h.



Desde el portal de Azure se pueden controlar multitud de aspectos relacionados con la suscripción adquirida, como por ejemplo crear recursos, ver la información general de los recursos ya existentes, monitorizar parámetros característicos de la VM (uso de CPU, memoria disponible, etc.), controlar el encendido/apagado, ...

El nombre asociado a la máquina virtual adquirida es "visa-lfs". Se ha mantenido el mismo alias como DNS de la URL en la cual está desplegada la aplicación Shiny, descrita en la sección anterior: *visa-lfs.westeurope.cloudapp.azure.com*. De este modo, tal y cómo se adelantó en la sección "Representación de la información", la herramienta de visualización interactiva desarrollada en este proyecto se encuentra accesible a través de la URL: http://visa-lfs.westeurope.cloudapp.azure.com:8080/.