UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

Uso de la transformada de *Wavelet* para la reducción del ruido en series de tiempo y su aplicación en el pronóstico de los niveles de partículas suspendidas en la ZMVM

TESINA Y EXAMEN PROFESONAL

QUE PARA OBTENER EL TÍTULO DE

ACTUARIA

PRESENTA

Eréndira Teresa Navarro García

Asesor: Dra. María del Carmen González Videgaray

Agosto 2015

Ciudad Universitaria, D. F.

Agradecimientos

Quiero agradecer de manera muy especial a mis padres. A mi mamá, mi gran compañera, juntas hemos superado muchas dificultades y tu apoyo ha sido fundamental para que pueda estar presentando este trabajo. A mi papá, durante toda mi vida, cada día, me has enseñado la importancia de leer, estudiar y aprender. Sin eso que me inculcaste, nada de esto habría sido posible.

También quisiera agradecer a Rodolfo Maya, en cada momento de este largo camino siempre estuviste ahí para que alcanzara el objetivo, muchas gracias.

Por otra parte quisiera agradecer al Dr. Víctor Palencia, un experto en Wavelets, por su apoyo e interés para la realización de este trabajo.

Y por supuesto un enorme agradecimiento a la Dra. MariCarmen por su tiempo, paciencia y gran apoyo, valoro muchos sus palabras y sus consejos, muchas gracias.

Índice general

Introducción

Al tomar una serie de tiempo para formular un modelo que pueda predecir su comportamiento en el futuro, se puede enfrentar un difícil reto si esta serie tiene datos altamente perturbados, pues los resultados del modelo no van a tener la exactitud deseada. En estos casos se utilizan métodos de suavizamiento para reducir el ruido presente en la serie; algunos pueden ser las medias móviles, medias móviles ponderadas o suavizamiento exponencial, existen otros métodos como la transformada de *Wavelet* que ha mostrado gran eficiencia con series tan perturbadas como los valores históricos de los contaminantes.

Si se requiere describir la transformada de *Wavelet* es conveniente conocer la transformada de Fourier, la cual divide una serie en las frecuencias que la componen; si tenemos una función f que depende del tiempo, por poner un ejemplo, la transformada de Fourier de esta función dependerá de la frecuencia.

Una función y su transformada de Fourier nos mostrarán dos versiones de la misma información. La transformada de Fourier muestra la información acerca de las frecuencias, pero la parte del tiempo se esconde en las fases de la transformada; por ejemplo, la transformada de Fourier de una melodía muestra que notas son tocadas, a partir de las frecuencias, pero no podría decir el momento en que son tocadas.

Existente ocasiones en las que no se puede elegir entre la frecuencia y el tiempo de ocurrencia pues para el análisis necesitamos de las dos, en estos casos se puede utilizar la transformada de Fourier de ventana (*Windowed Fourier Transform*) cuya idea es analizar la frecuencia de una señal segmento por segmento de tiempo.

Con la transformada de Fourier clásica se compara la señal entera de senos y cosenos de diferentes frecuencias para saber cuanto de cada frecuencia esta contenida en la señal; con en análisis por ventana, se compara una parte de las oscilaciones, primero de una frecuencia, después de otra y así sucesivamente.

La transformada de Fourier por ventana implica ciertas limitantes, mientras más corta es la ventana se pueden localizar mejor las discontinuidades o picos, pero disminuye la capacidad de analizar las frecuencias bajas, mientras que con una ventana grande se pueden analizar mejor las frecuencias bajas pero se pierde la capacidad de localizar en el tiempo.

La transformada *Wavelet* permite descomponer la señal en tiempo y frecuencia simultáneamente; por lo tanto, cuando se analiza una serie de tiempo con picos frecuentes, donde ambos componentes son esenciales, es conveniente utilizar esta transformada. En el caso de la reducción del ruido bajo este método, el proceso o idea central que se sigue empieza con obtener la transformada de *Wavelet* de la señal, identificar los picos o discontinuidades en las frecuencias y utilizar alguna regla para disminuir sus contrastes sin perder la tendencia que tiene y por último aplicar la transformada inversa, como resultado se obtiene una serie suavizada.

El objetivo del presente trabajo es comprobar que se obtiene un mejor resultado y ajuste en los pronósticos al utilizar la transformada de *wavelet* como método de reducción del ruido en una serie histórica de los niveles de contaminación por partículas menores a 10 micrómetros (PM_{10}).

Como parte de la investigación se puede citar *Wavelet analysis of ecological time series* (Cazelles *et al.*, 2008) donde se realizó una revisión sobre las ventajas de utilizar la transformada de *Wavelet* en el análisis de sistemas no estacionarios como los sistemas ecológicos. De manera más aterrizada, en *Predicting time series using neural networks with wavelet-based denoising layers* (Lotric y Dobnikar, 2005) se propone que el análisis multiresolución *wavelet* para la eliminación del rudio puede ser integrado para formular una red neuronal artificial, ya que esto minimiza el error, dado que de esta manera se logra captar la estructura caótica que llegan a tener las series de tiempo. También en *Wavelet transform-based artificial neural networks (WT-ANN) in* PM_{10}

pollution level estimation, based on circular variables (Shekarrizfard *et al.*, 2011) se propone un modelo donde se utiliza la transformada de *Wavelet* para la reducción de la perturbación en la serie de tiempo del contaminante y otras series meteorológicas, posteriormente se realiza un análisis de componentes principales y finalmente se incluyen los resultados en una red neuronal artificial. Como conclusión se puede observar que la precisión aumenta en este modelo a comparación de uno en donde no se utiliza la transformada.

Entonces, con datos obtenidos del Sistema de Monitorio Atmosférico (SIMAT) se buscará implementar un modelo para la reducción del ruido presente en la serie de tiempo de los niveles de partículas suspendidas en el Noroeste de la Zona Metropolitana del Valle de México (ZMVM) por medio de la transformada de *Wavelet* y utilizar las observaciones resultantes para establecer un pronóstico a corto plazo.

El interés de analizar los niveles de contaminación recae en mi participación en el Concurso Nacional de Estadísticos Jóvenes de 2011 en el que se observaron las complicaciones que representa el manejo de estos datos; además del problema de salud pública que representa la contaminación en una zona como el área conurbada de la Ciudad de México. Por otra parte también se pretende mostrar una de las aplicaciones en el campo de la Estadística que tiene la transformada de *Wavelet*.

Capítulo 1

Niveles de contaminación en la Zona Metropolitana del Valle de México

Para el presente trabajo se utilizará la información de los niveles de contaminación por partículas suspendidas en la Zona Metropolitana del Valle de México. En este capítulo se presentarán la entidad responsable del monitoreo de la calidad del aire, los contaminantes y variables que mide, además se observará el comportamiento histórico con el fin de conocer el tipo de datos que se manejarán. También se conocerán los valores de referencia contenidos en las Normas Oficiales Mexicanas que indican qué nivel de cada contaminante representa un riesgo para la salud.

1.1. ¿Cómo se mide la contaminación en la Zona Metropolitana del Valle de México?

La entidad encargada del monitoreo de la calidad del aire en la Ciudad de México es el SIMAT (Sistema de Monitoreo Atmosférico), es una entidad dependiente de la Secretaría del Medio Ambiente en la Ciudad de México y su fin es proteger la salud de la población mediante el monitoreo y difusión de la calidad del aire. El SIMAT

tiene 40 estaciones de monitoreo tanto en delegaciones de la Ciudad de México como en municipios aledaños pertenecientes al Estado de México.

Las estaciones de monitoreo no se encuentran en la totalidad de la Zona Metropolitana del Valle de México, en adelante ZMVM; que en el última delimitación realizada por INEGI (Instituto Nacional de Estadística y Geografía), en base al censo de 2010, está integrada por las 16 delegaciones de la Ciudad de México, 59 Municipios del Estado de México y 1 municipio de Hidalgo. Las estaciones de monitoreo se encuentran distribuidas en una extensión de la ZMVM que nos permite conocer el estado de la calidad del aire en al menos la totalidad de los municipios centrales de esta zona.

El SIMAT cuenta con 4 redes de estaciones, las cuales se dedican a recolectar la información necesaria para medir la calidad del aire:

- Red Automática de Monitoreo Atmosférico (RAMA), cuenta con 29 estaciones de monitoreo y con equipos para medir dióxido de azufre, monóxido de carbono, dióxido de nitrógeno, ozono, partículas menores a 10 y a 2.5 micrómetros.
- Red Manual de Monitoreo Atmosférico (REDMA), cuenta con 11 estaciones que realizan un muestreo cada 6 días de partículas suspendidas.
- Red de Meteorología y Radiación Solar (REDMET), se integra por 19 estaciones de monitoreo para variables meteorológicas, temperatura, humedad relativa, dirección y velocidad de viento, radiación solar y presión barométrica.
- Red de Depósito Atmosférico (REDDA), cuenta con 16 estaciones que realizan un muestreo cada 6 días de polvo sedimentable, lluvia, granizo, nieve y rocío.

Para completar sus objetivos, el SIMAT cuenta además con un Laboratorio de Análisis Ambiental (LAA) y con un Centro de Información de la Calidad del Aire (CICA).

Cada contaminante, debido a su naturaleza, tiene distinto equipo y método de medición. Para definir los equipos a utilizar el SIMAT sigue la Norma Oficial de cada contaminante y/o *el método de referencia o equivalente* establecidos por la Agencia de protección Ambiental de los Estados Unidos.

También existen Normas Oficiales tanto para la elaboración del índice de la calidad del aire como para los límites máximos permisibles de los contaminantes. Por ejemplo, para las PM_{10} la Norma Oficial Mexicana vigente NOM-025-SSA1-2014 (Secretaría de Salud, 2014b) establece un límite máximo para el promedio de 24 horas de 75 $\mu g/m^3$ y para el promedio anual de 40 $\mu g/m^3$. Para este contaminante, el SIMAT reporta información por hora y de acuerdo a los lineamientos de la NOM para realizar el promedio de 24 horas de las PM_{10}, se requiere un mínimo de 75 % de concentraciones horarias válidas, es decir, 18 horas. Para calcular el promedio anual se necesitan un 75 % de muestras de 24 horas válidas para cada trimestre, en primer lugar se deben calcular los promedios trimestral para después en base a esto obtener el promedio anual. Después de realizar los promedios con estas limitantes se puede establecer si el contaminante sobrepaso algún límite máximo y por lo tanto la salud de la población se encuentra en riesgo.

Como ya se mencionó, se utilizaran los datos de partículas suspendidas, en específico las partículas menores a 10 micrómetros PM_{10}. Las partículas suspendidas son "cualquier material sólido o liquido que es capaz de mantenerse suspendido en el aire por medios físicos o mecánicos" (Granados *et al.*, 2012). Al estar suspendidas en el aire el ser humano las respira y dependiendo de su tamaño logran alojarse en distintos niveles del sistema respiratorio.

Las fuentes de contaminación por partículas suspendidas son diversas como la combustión de vehículos automotores, procesos industriales, los incendios, el polen e incluso la agricultura.

Se utilizan dos medidas de partículas para la regulación de los niveles máximos permitidos, partículas con diámetro menor a 2.5 μm que puede llegar hasta los pulmones ($PM_{2,5}$) y partículas con diámetro menor a 10 μm que se localizan a distintos niveles del sistema respiratorio, desde las fosas nasales hasta la traquea o los bronquios. Debido a esto, al inhalar diariamente estas partículas en concentraciones altas se altera el funcionamiento del sistema respiratorio así como la coagulación de la sangre (Brook *et al.*, 2003).

Un estudio hecho por el Instituto Nacional de Salud Pública en 2009 muestra

una estimación de las muertes evitables por año, en la ZMVM, si se cumpliera la Norma Europea para las PM_{10} de 40 $\mu g/m^3$ y son alrededor de 1,000 muertes al año (Comisión Ambiental Metropolitana, 2011, pp. 17-26), actualmente este es el valor de referencia anual que tiene la NOM Mexicana. Este mismo estudio también da a notar que cuando los niveles de partículas son elevados loe efectos negativos a la salud son mayores a comparación de cuando ocurren concentraciones altas de Ozono.

Por estos motivos se utilizarán datos de PM_{10} para este trabajo, ya que, además de buscar establecer el beneficio de utilizar la Transformada de *Wavelet* para el manejo de series de este tiepo, se quiere realizar un modelo estadístico para poder obtener un pronóstico que pueda ser útil a corto plazo en el monitoreo de este contaminante.

1.2. Análisis descriptivo de las partículas suspendidas y otros contaminantes

Los contaminantes que se monitorean en la ZMVM, y tienen un límite máximo establecido por la Normas Oficiales Mexicanas, se llaman contaminantes criterio, estos son: ozono (O_3), monóxido de carbono (CO), bióxido de azufre (SO_2), bióxido de nitrógeno (NO_2), plomo (Pb), partículas suspendidas menores a 10 micrómetros (PM_{10}) y partículas suspendidas menores a 2.5 micrómetros ($PM_{2,5}$). En esta sección se mostrarán los niveles históricos en un periodo de 10 años ($2003 - 2014$) de los contaminantes que tengan una NOM vigente y que hayan sido monitoreados la mayor parte de este periodo en forma horaria.

En cuanto a las estaciones de las que se extrajeron los datos de las variables de estudio, son 5 estaciones y cada una de ellas pertenece a cada región de la ZMVM (Noreste, noroeste, centro, sureste y suroeste). Se eligieron estas estaciones debido a que contaban con datos de todos los contaminantes criterio incluidos en el presente trabajo, para todo o casi todo el periodo de tiempo que se esta analizando y además todas siguen activas.

Esto se realiza con el fin de observar el distinto comportamiento que cada

contaminante tiene con respecto a su región de recolección. En la tabla 1.1 se muestra la relación Región - Estación y la abreviatura que se utilizará en adelante (es la misma que emplea SIMAT).

Región	Estación	Abreviatura
Noreste	San Agustín	SAG
Noroeste	FES Acatlán	FAC
Centro	Merced	MER
SUreste	Tláhuac	TAH
Suroeste	Pedregal	PED

Tabla 1.1: Estaciones incluidas

Los contaminantes criterio a analizar son ozono (O_3), monóxido de carbono (CO), bióxido de azufre (SO_2), bióxido de nitrógeno (NO_2) y partículas suspendidas menores a 10 micrómetros (PM_{10}). Se descartaron 2 contaminantes criterio, el plomo (Pb) porque no se reporta en forma horaria y partículas suspendidas menores a 2.5 micrómetros ($PM_{2,5}$) debido a que no se tiene información para todo el periodo de estudio, ya que se empezó a monitorear después de 2003 y solo en algunas estaciones y no de forma regular, sobre todo en los primeros años del periodo de tiempo que se esta considerando.

La tabla 1.2 muestra los parámetros de referencia para estos contaminantes, los niveles históricos se presentarán de acuerdo a estos parámetros, ya sea en promedio 24 horas, promedio móvil 8 horas, por hora y/o promedio anual. En particular para las PM_{10} se podrá ver la información de cada una de la estaciones mencionadas en la 1.2, para el resto de los contaminantes se tendrá la información de la estación FAC y en el Anexo en formato digital se puede encontrar la información para las estaciones restantes.

Es necesario mencionar que al momento de realizar este análisis todas las NOM a las que se hace referencia se encuentran vigentes, es decir, si en algún momento se sobrepasan los niveles permitidos se puede decir que la estación no cumple la NOM. Para los contaminantes O_3, PM_{10} y SO_2, cuyas NOM comenzaron a ser vigentes en el

Tabla 1.2: Parámetros en las NOM vigentes para las variables de estudio

Contaminante	Descripción	Publicación	Unidad de Monitoreo
SO_2	0.110 ppm, máximo promedio de 24 horas 0.200 ppm, promedio móvil de 8 horas 0.025 ppm, promedio anual	08-09-2010	ppb
CO	11.0 ppm, máximo anual como promedio móvil de 8 horas	23-12-1994	ppm
NO_2	0.210 ppm, promedio horario	23-12-1994	ppb
O_3	0.095 ppm, promedio horario 0.070 ppm, máximo anual del promedio móvil de 8 horas	19-08-2014	ppb
PM_{10}	75 $\mu g/m_3$, promedio 24 horas 40 $\mu g/m_3$, promedio anual	20-08-2014	$\mu g/m_3$

transcurso del periodo de estudio, no se puede afirmar incumplimiento de alguna norma para el periodo anterior al inicio de su vigencia.

En el periodo anterior al inicio de su vigencia los contaminantes tenían otros parámetros de referencia, otras NOM vigentes[1]. En este trabajo no se incluyeron esos limites de referencia ya que se consideró mejor opción homologar todo el periodo para poder comparar los niveles de todos los años de estudio con un mismo parámetro y se consideró más adecuado utilizar el parámetro actual, debido a que estos nuevos parámetros están homologados con los utilizados a nivel mundial para prevenir los riesgos a la salud.

1.2.1. Comportamiento histórico

Como se mencionó anteriormente se necesitan un mínimo de concentraciones horarias válidas para poder obtener información diaria o anual de los contaminantes de acuerdo a las NOM, por esta razón, como primer paso se realizó un análisis de las

[1]Para el O_3 la NOM anterior era la NOM-020-SSA1-1993, para las PM_{10} la NOM-025-SSA1-1993 y para SO_2 la NOM-022-SSA1-1993

variables de estudio para conocer la calidad de los datos.

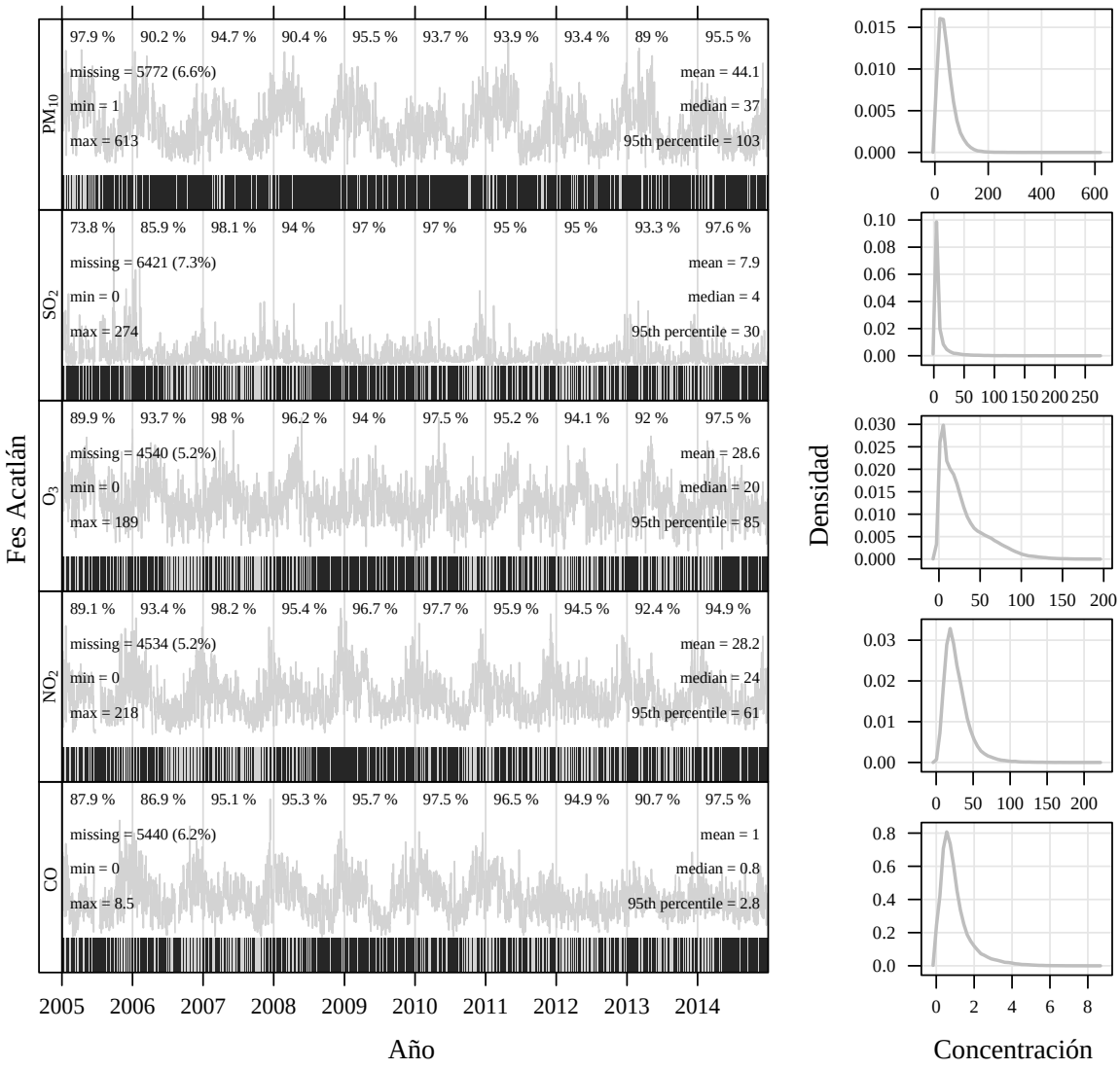

Figura 1.1: FES ACATLÁN - Gráfico de resumen

En la figura1.1 se muestran las concentraciones horarias para la estación FAC en los últimos 10 años. Se ha tenido un porcentaje anual de concentraciones horarias válidas por encima del 85 %, esto da una idea general de que en todos los contaminantes se tiene un porcentaje aceptable en los datos válidos, en específico para las PM_{10} se

tiene un porcentaje de datos válidos mayor al 90 % excepto para el año 2013.

Hablando de los niveles mostrados en las gráficas, se puede ver que en el caso de las PM_{10} los niveles más altos se dieron a final e inicio de cada año. Para el NO_2, también los niveles máximos, ocurrieron en los mismos momentos aunque el rango días donde se dan estos niveles es menor. Con el CO ocurría lo mismo hasta 2010, en los últimos años los máximos no han alcanzado los niveles de los primeros años. Para O_3 los máximos se establecieron en la mitad del primer semestre del año, con el SO_2 en el primer año es cuando se dio el máximo, en los años posteriores descendió y se estabilizó.

En la figura 1.2 están los niveles de PM_{10} para las 5 estaciones. Esto es útil para comparar el comportamiento que el contaminante ha tenido históricamente en cada estación a lo largo de estos 10 años de estudio. La estación FAC es la que tiene el menor porcentaje de datos no válidos con 6.6 %, el mayor pertenece a TAH con 19 %, esto se debe a que en los primeros meses del 2005 no estaba en funcionamiento, esta estación, para la medición de PM_{10}. En la estación SAG se observa, en los primeros meses de 2006, un periodo de datos fuera de los niveles mostrados normalmente, alcanzando un máximo de 1570 $\mu g/m^3$, al mismo tiempo se da el valor máximo para TAH que es de 929 $\mu g/m^3$. Ambos fuera del rango $[576, 717]$ de las otras 3 estaciones.

En las 5 estaciones, de manera general, se tienen el mismo comportamiento. Niveles más altos a inicio y/o finales de año, niveles más bajos a mediados de año.

La estación PED tiene los estadísticos más bajos en los 10 años de estudio, incluso en la gráfica de densidad se puede ver que sus datos se concentran por debajo de 100 $\mu g/m^3$. Su mediana es de 34 $\mu g/m^3$, su media de 38.5 $\mu g/m^3$ y su percentil 95 de 83 $\mu g/m^3$. La siguen FAC, TAH, SAG y MER, en orden de menor a mayor, los niveles de MER alcanzan una mediana de 48 $\mu g/m^3$, media de 54.3 $\mu g/m^3$ y percentil 95 de 114 $\mu g/m^3$. Para el percentil 95 SAG tiene el nivel más alto con 122 $\mu g/m^3$.

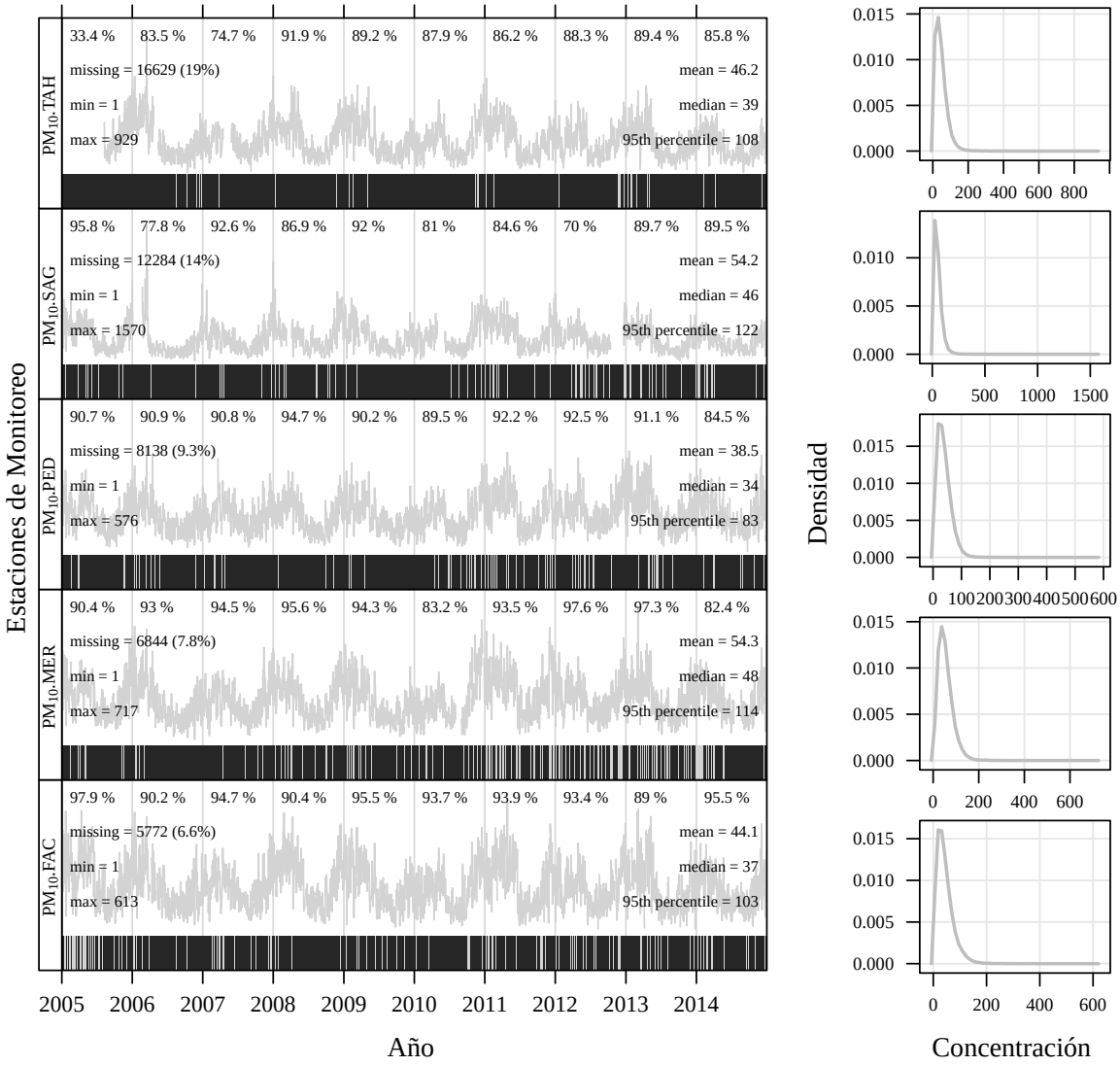

Figura 1.2: PM_{10} - Gráfico de resumen

1.2.2. Comportamiento de acuerdo a las NOM

Los datos de la sección anterior son útiles porque brindan una idea de los niveles que alcanzan los datos, el comportamiento de su serie de tiempo, *outliers* o cuántos *missing* existen. Pero no proporcionan información para definir si estos niveles están bien o mal, si causan daños a la salud o son normales. Para conocer esto se compararan los niveles con los valores de referencia de la NOM correspondiente (tabla 1.2) y para esto se aplicará el procedimiento para calcular los indicadores como lo indique la NOM.

Los contaminantes NO_2, O_3 y SO_2 se publican en partículas por billón (ppb) de modo que los valores de referencia, en partículas por millón, se transformaron.

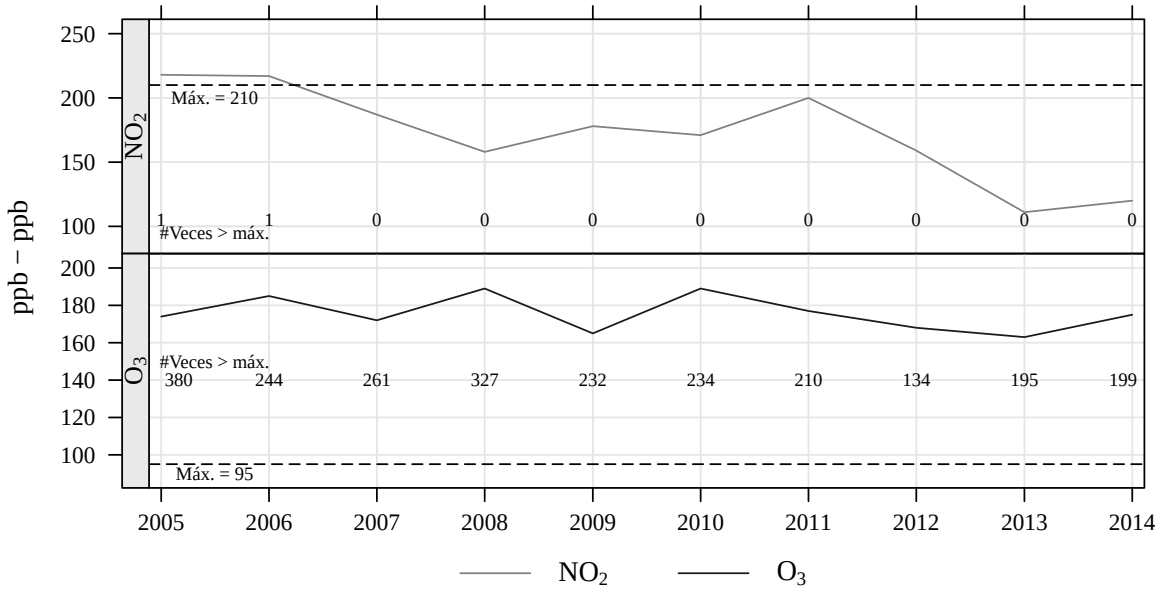

Figura 1.3: FAC - Referencia Horaria

Referencia horaria. En la figura 1.3 se tienen los dos contaminantes con un valor de referencia a nivel horario, estos son NO_2 y O_3. Con línea punteada se colocó el valor

de referencia, en este caso 0.210 y 0.095 ppm respectivamente. Ambas gráficas están en ppb ya que los datos se reportan en esta unidad, por lo tanto los valores de referencia transformados son 210 y 95 ppb.

El NO_2 en todos los años ha mantenido un nivel dentro de los valores de referencia, en 2005 y 2006 se excedió el nivel permitido pero en ambos casos fue una sola hora y por lo tanto, de acuerdo su NOM se encuentra en el límite de referencia.

En el caso de O_3 la NOM no permite que en ninguna hora se exceda el valor de referencia y en todos los años se ha estado por encima de este nivel, en cada año el máximo promedio horario a estado por encima de las 0.160 ppm (160 ppb). Cabe aclarar que la NOM vigente fue publicada el 19 de agosto de 2014 y este análisis se está haciendo en retrospectiva; la NOM anterior (NOM-020-SSA1-1993), tenía como valor máximo promedio horario 0.110 ppm, se puede observar que aún así se excede el límite pero las horas en que sucede disminuiría al comparar los niveles con este valor de referencia.

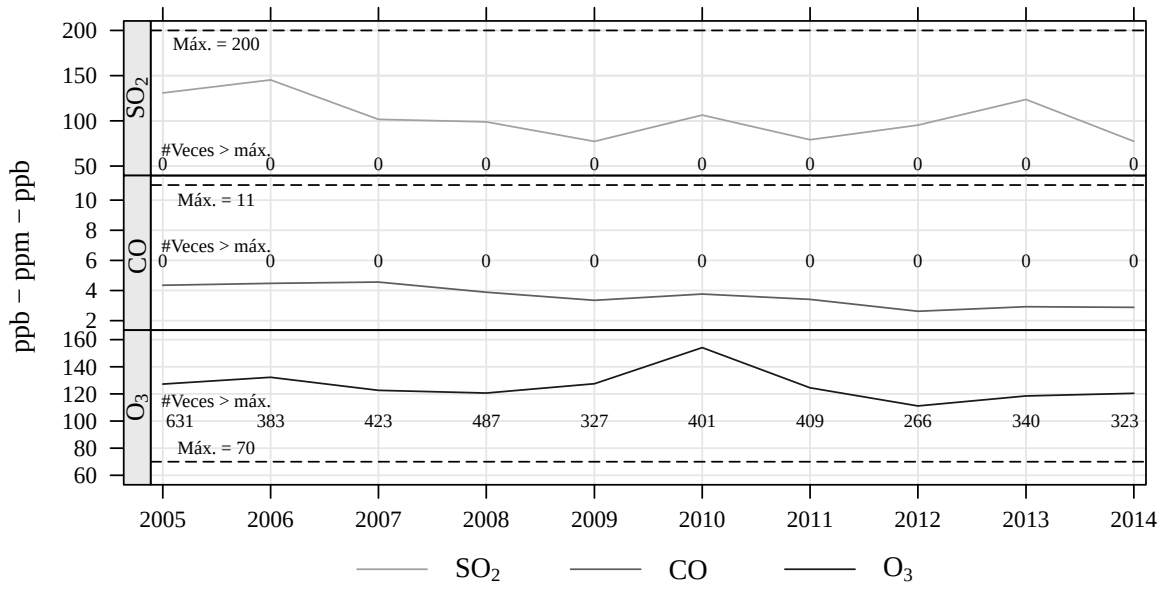

Figura 1.4: FAC - Referencia Promedio Móvil 8 Horas

Referencia promedio móvil 8 horas. La figura 1.4 presenta los contaminantes con valor de referencia por promedio móvil 8 horas. Estos son SO_2, CO y O_3, para los dos primeros el máximo de los promedio móviles ha estado muy por debajo del valor de referencia (0.200 ppm y 11.0 ppm), en ninguna ocasión alcanzaron este nivel aun que la NOM permite 1 y 2 veces al año respectivamente. Para O_3 el valor de referencia es 70 ppm y en todos los años se ha incumplido la norma, los máximos anuales se localizan por encima de las 0.100 ppm (en la NOM anterior el valor era 80 ppm).

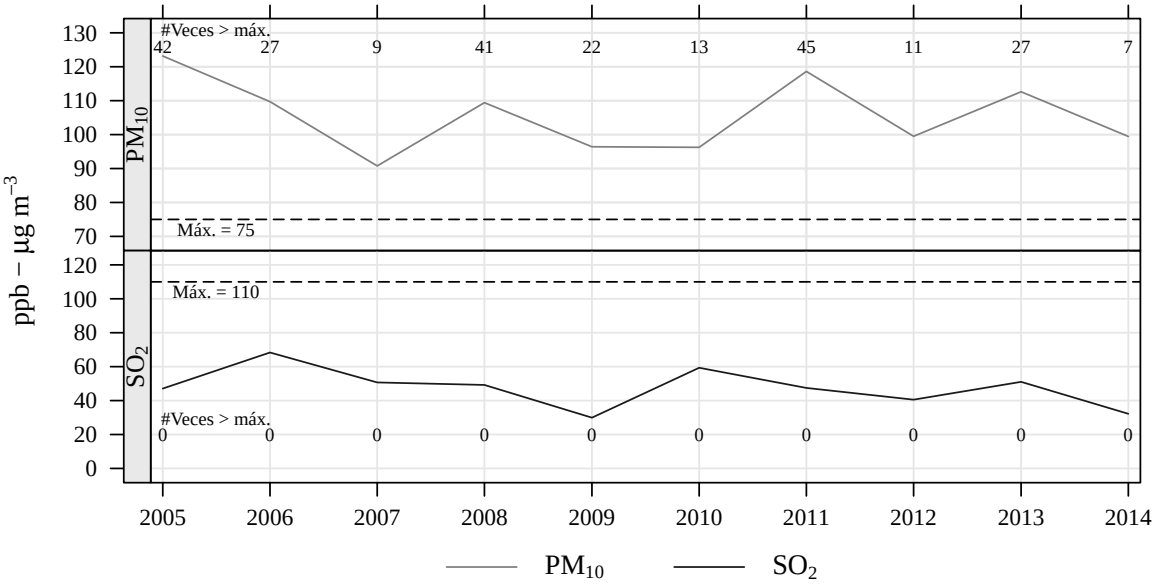

Figura 1.5: FAC - Referencia Promedio Diario

Referencia diaria. Para el promedio diario se muestran los resultados en la figura 1.5. Los contaminantes que se miden bajo este criterio son PM_{10} y SO_2, sus referencias son $75\mu g/m^3$ y 0,110 ppm respectivamente. Las PM_{10} tienen valores máximos anuales de promedio diario que excedente su nivel de referencia en todos los años de estudio, aunque vale la pena resaltar que el año 2014 tiene el mínimo de veces que se excedió este valor, solo 7 días; que si se comparan con los 45 días del años 2011, es una reducción importante. La NOM anterior a la vigente (NOM-025-SSA1-1993), tenía una referencia

de $120\mu g/m^3$, promedio de 24 horas (evaluado como el percentil 98 anual).

Para el SO_2 los niveles en ningún día de año excedieron el valor de referencia, en estos años siempre se han tenido valores por debajo de las 0,080 ppm.

Referencia anual. Por último la figura 1.6 presenta los resultado para promedio anual. Los contaminantes que tienen estas referencias son los mismos con referencia diaria, PM_{10} y SO_2, sus valores de referencia son $40\mu g/m^3$ y 0,025 ppm respectivamente.

Para las PM_{10} el valor de referencia se excede en casi todos los años, excepto en el 2010 y en el 2014. En ambos años el promedio anual se encuentra por debajo de los $40\mu g/m^3$. Los valores de SO_2 para los años 2005 y 2006 no se encuentran disponibles ya que no se cubre el porcentaje mínimo requerido de datos válidos, pero en los años donde se tiene información válida el promedio anual se encuentra muy por debajo del valor de referencia.

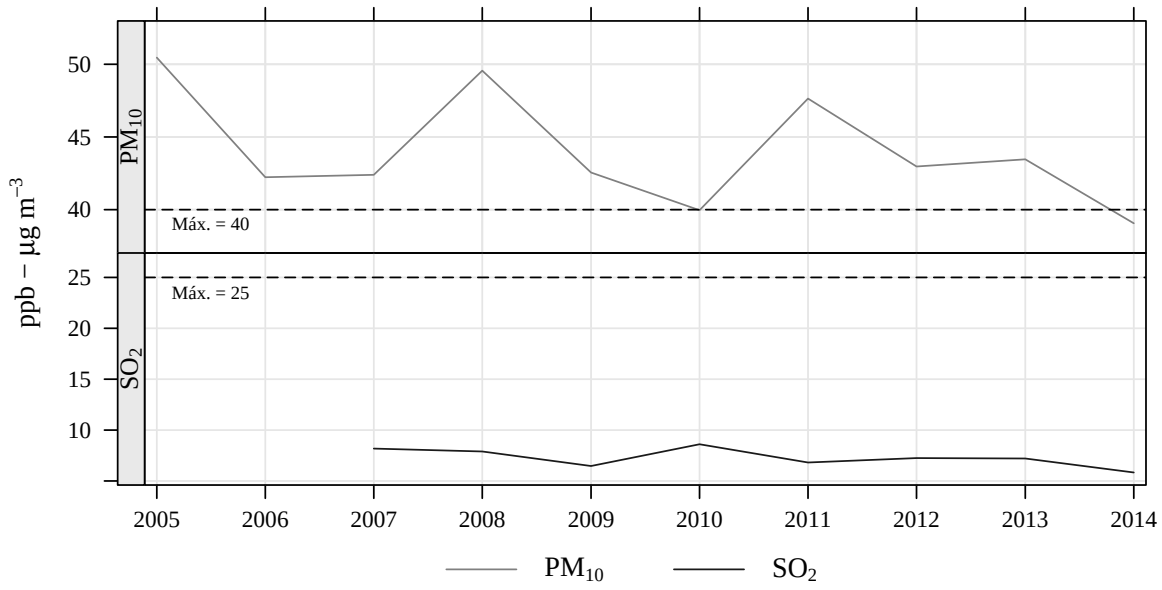

Figura 1.6: FAC - Referencia Promedio Anual

Después de presentar todos los contaminantes para la estación FAC, se puede concluir que los 2 contaminantes que pueden haber causado mayor problema a la salud son O_3 y PM_{10}, son los contaminantes que al aplicar la norma vigente en retrospectiva no cubren los parámetros de referencia en la mayoría y/o todos los años de estudio. Aunque esto no quiera decir que en el momento de ocurrencia hayan incumplido la norma que estaba vigente en ese momento.

Referencia PM_{10}. Para el contaminante PM_{10} se presentan los niveles tanto para máximo promedio diario como promedio anual para las 5 estaciones que se incluyen en este trabajo.

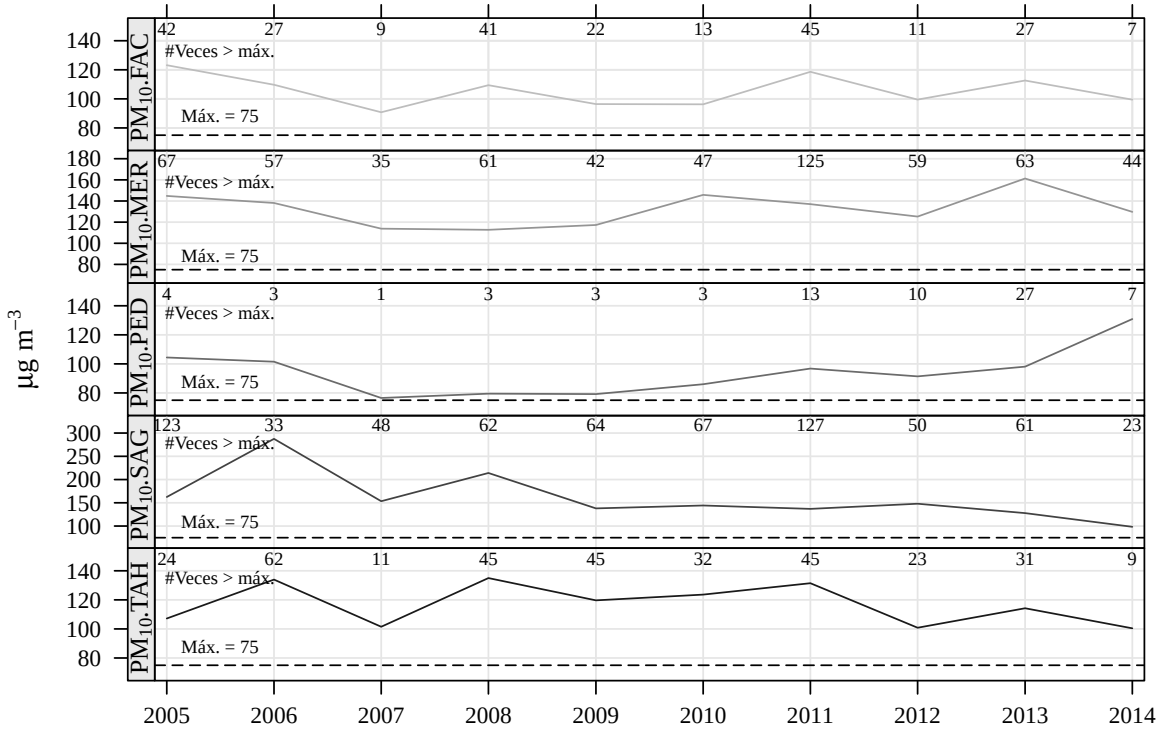

Figura 1.7: PM_{10} - Referencia Promedio Diario

En la figura 1.7 están los valores que corresponde al máximo promedio diario,

se observa que en el año 2014 descendió el nivel en casi todas las estaciones excepto en PED, en particular esta estación tuvo los niveles más bajos durante el periodo observado y justo en el último año el máximo se elevó a comparación de su histórico, aunque las veces que superó el valor de referencia $75\mu g/m^3$ no fue mayor que en las demás estaciones. En general en los últimos 10 años el nivel del indicador no estuvo por debajo del valor de referencia en ningún momento. SAG y MER son las estaciones que han sobrepasado más veces el valor de referencia. PED es la estación que menos veces ha estado por encima, pero en los últimos 4 años ha aumentado pero sin llegar a los valores de las estaciones antes mencionadas. Lo niveles del indicador no siguen un patrón por año claramente definido y la estación que muestra más estabilidad es FAC.

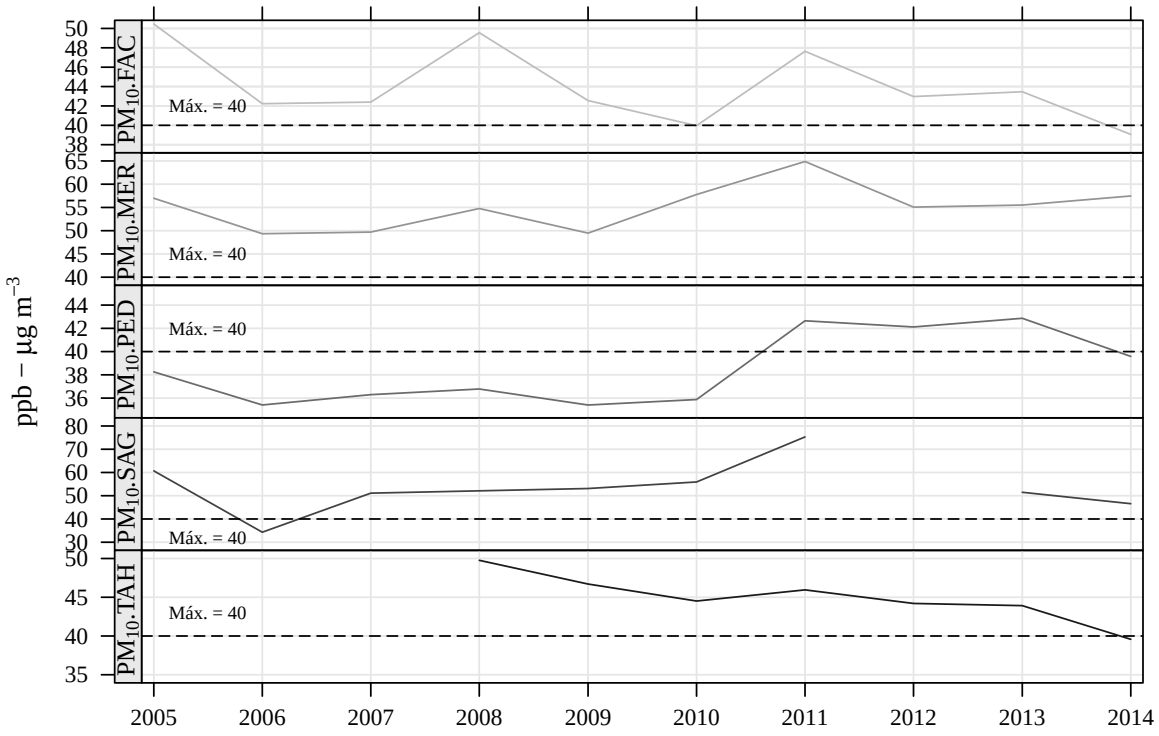

Figura 1.8: PM_{10} - Referencia Promedio Anual

En la figura 1.8 se encuentra los niveles para el promedio anual de las PM_{10}.

Aquí también se puede distinguir claramente el aumento de nivel en la estación PED, es la única estación que se mantuvo los primeros 6 años por debajo del valor de referencia $40\mu g/m^3$, al contrario que en la figura 1.7, aquí se muestra que en el año 2014 el promedio de la estación descendió, incluso por debajo del valor de referencia. La única estación que elevó su nivel de promedio anual fue MER y aún se encuentra muy por encima del valor de referencia, considerando los niveles de las demás estaciones. La estación TAH no tiene el porcentaje mínimo de datos válidas para poder calcular el promedio anual de 2005 a 2007, lo mismo ocurre con SAG en 2012.Al igual que con el máximo promedio 24 horas, no se puede distinguir un patrón en el comportamiento de promedio anual en las estaciones de estudio.

Capítulo 2

Wavelets y la reducción del ruido

2.1. ¿Qué es una *wavelet*?

En español el término *wavelet* se traduce como onda pequeña, esto quiere decir que es una onda que crece y decrece en un periodo de tiempo corto y limitado. El ejemplo de un caso contrario es la función $\text{sen}(x)$, esta función crece y decrece en todos los puntos cuando $x \in \mathbb{R}$.

Una *wavelet* debe tener dos propiedades básicas que se definen de la siguiente manera. Sea $\psi(\cdot)$ una función definida en $\mathbb{R}$ donde

1. La integral de $\psi(\cdot)$ es cero
$$\int_{-\infty}^{\infty} \psi(u)\, du = 0 \tag{2.1.1}$$
2. La integral de $\psi(\cdot)^2$ es la unidad
$$\int_{-\infty}^{\infty} \psi(u)^2\, du = 1 \tag{2.1.2}$$

El término "onda pequeña" es explicado en las ecuaciones 2.1.1 y 2.1.2, con la ecuación 2.1.1 se puede observar que cualquier crecimiento en la función por encima de

cero será cancelado por un decrecimiento de la misma, por lo tanto se considera una onda a la representación gráfica de $\psi(\cdot)$. Por otro lado, si la ecuación 2.1.2 se cumple entonces para cualquier ϵ que cumpla que $0 < \epsilon < 1$, existe un intervalo $[-T, T]$ tal que

$$\int_{-T}^{T} \psi(u)^2 \, du > 1 - \epsilon$$

Si se considera a un ϵ pequeño, entonces $\psi(\cdot)$ evaluada fuera del intervalo $[-T, T]$ tomará valores muy cercanos a cero, los valores diferentes de cero solo serán tomados cuando $\psi(\cdot)$ es evaluada en $[-T, T]$ y si comparamos el intervalo $[-\infty, \infty]$ con $[-T, T]$ se puede ver que los valores sólo serán tomados en un intervalo relativamente corto.

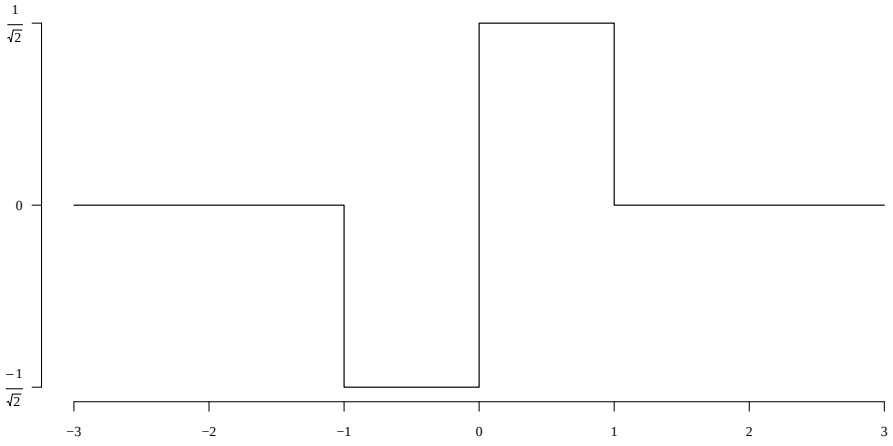

Figura 2.1: *Wavelet* Haar

Ejemplo 2.1.1. *Wavelet* Haar.

$$\psi(u) = \begin{cases} \frac{-1}{\sqrt{2}}, & -1 < u \leq 0; \\ \frac{1}{\sqrt{2}}, & 0 < u \leq 1; \\ 0, & \text{o.c.} \end{cases} \tag{2.1.3}$$

Para probar 2.1.1

$$\begin{aligned}\int_{-\infty}^{\infty}\psi(u)\,du &= \int_{-1}^{0}\frac{-1}{\sqrt{2}}\,du+\int_{0}^{1}\frac{1}{\sqrt{2}}\,du\\ &= \frac{-u}{\sqrt{2}}\Big|_{-1}^{0}+\frac{u}{\sqrt{2}}\Big|_{0}^{1}\\ &= \left(\frac{0}{\sqrt{2}}-\frac{1}{\sqrt{2}}\right)+\left(\frac{1}{\sqrt{2}}-\frac{0}{\sqrt{2}}\right)\\ &= 0\end{aligned}$$

Para probar 2.1.2

$$\begin{aligned}\int_{-\infty}^{\infty}\psi(u)^2\,du &= \int_{-1}^{0}\left(\frac{-1}{\sqrt{2}}\right)^2du+\int_{0}^{1}\left(\frac{1}{\sqrt{2}}\right)^2du\\ &= \int_{-1}^{0}\frac{1}{2}\,du+\int_{0}^{1}\frac{1}{2}\,du\\ &= \frac{u}{2}\Big|_{-1}^{0}+\frac{u}{2}\Big|_{0}^{1}\\ &= \left(\frac{0}{2}-\left(\frac{-1}{2}\right)\right)+\left(\frac{1}{2}-\frac{0}{2}\right)\\ &= 1\end{aligned}$$

2.2. Introducción a la transformada continua de *Wavelet*

A partir de la transformada de *Wavelet* se puede saber la variación de un periodo de tiempo a otro, sea $x(\cdot)$ una función real valuada de una variable independiente t, que para estos fines se tomará como "tiempo" y $x(\cdot)$ como una "señal". Considerando la integral

$$\frac{1}{a-b}\int_{a}^{b}x(u)\,du\equiv\alpha(a,b). \tag{2.2.1}$$

donde se asume que $a<b$ y que la integral de $x(\cdot)$ está bien definida. Se puede considerar a $\alpha(a,b)$ como el promedio de valores de $x(\cdot)$ en el intervalo $[a,b]$, esto debido a que

la media muestral de un conjunto de N observaciones donde $x(\cdot)$ es una función de la forma

$$x(t) = x_j, \text{ para } a + \frac{j}{N}(b-a) < t \leq a + \frac{j+1}{N}(b-a) \text{ y } j = 0, \ldots, N-1$$

y por la definición de la integral de Riemann, se sigue que

$$\frac{1}{b-a}\int_a^b x(u)\,du = \frac{1}{b-a}\sum_{j=0}^{N-1} x_j \frac{b-a}{N} = \frac{1}{N}\sum_{j=0}^{N-1} x_j. \tag{2.2.2}$$

Considerando el largo del intervalo como $\lambda = b-a$ y el centro en el intervalo de tiempo como $t = (a+b)/2$, se define

$$A(\lambda, t) \equiv \alpha\left(t - \frac{\lambda}{2}, t + \frac{\lambda}{2}\right) = \frac{1}{\lambda}\int_{t-\frac{\lambda}{2}}^{t+\frac{\lambda}{2}} x(u)\,du. \tag{2.2.3}$$

Por lo tanto $A(\lambda, t)$ es el promedio de valores de la señal $x(\cdot)$ bajo la escala λ centrado en el tiempo t.

Para ejemplificar este concepto y tomando como referencia (Percival y Walden, 2000), supongamos una serie de datos que contienen el promedio de desviaciones que tiene un reloj de pared comparado con un reloj maestro. Los promedios de la señal $x(\cdot)$ bajo intervalos de la forma $[t - \frac{1}{2}, t + \frac{1}{2}]$ tomando a t como días, nos dicen que, si $A(1,t) = 0$ entonces el reloj no tiene ninguna desviación respecto al reloj maestro en el periodo $[t - \frac{1}{2}, t + \frac{1}{2}]$. Si $A(1,t) < 0$ entonces el reloj presenta una pérdida de tiempo. Para conocer la certeza en la medición de el reloj atómico sería necesario conocer cuanto cambia $A(1,t)$ de un periodo de tiempo a otro, definido de la siguiente manera

$$D\left(1, t - \frac{1}{2}\right) \equiv A(1,t) - A(1,t-1) = \int_{t-\frac{1}{2}}^{t+\frac{1}{2}} x(u)\,du - \int_{t-\frac{3}{2}}^{t-\frac{1}{2}} x(u)\,du,$$

de manera equivalente

$$D(1,t) \equiv A\left(1, t + \frac{1}{2}\right) - A\left(1, t - \frac{1}{2}\right) = \int_t^{t+1} x(u)\,du - \int_{t-1}^{t} x(u)\,du. \tag{2.2.4}$$

Si $|D(1,t)|$ es cercano a cero, entonces la certeza en la medición es buena, para examinar la medición en la escala λ se reescribe la ecuación 2.2.1 como

$$D(\lambda,t) \equiv A\left(\lambda, t+\frac{\lambda}{2}\right) - A\left(\lambda, t-\frac{\lambda}{2}\right) = \frac{1}{\lambda}\int_t^{t+\lambda} x(u)\,du - \frac{1}{\lambda}\int_{t-\lambda}^{t} x(u)\,du. \quad (2.2.5)$$

Asociar las *wavelets* con estos cambios en los promedios es muy sencillo, dado que en la ecuación 2.2.4 las dos integrales tienen intervalos adyacentes y no sobrepuestos entonces se obtiene una integral bajo el eje de los reales.

$$D(\lambda,t) = \int_{\infty}^{\infty} \tilde{\psi}_{\lambda,t}(u)x(u)\,du, \quad (2.2.6)$$

donde

$$\tilde{\psi} = \begin{cases} \frac{-1}{\lambda}, & t-\lambda < u \leq t; \\ \frac{1}{\lambda}, & t < u \leq t+\lambda; \\ 0, & \text{o.c.} \end{cases} \quad (2.2.7)$$

Cuando $\lambda = 1$ y $t = 0$ se tiene que

$$\tilde{\psi} = \begin{cases} -1, & -1 < u \leq 0; \\ 1, & 0 < u \leq 1; \\ 0, & \text{o.c.} \end{cases} \quad (2.2.8)$$

Si se compara la ecuación 2.2.8 con la ecuación 2.1.3 (*Wavelet* Haar), se puede ver que

$$\tilde{\psi}_{1,0}(u) = \sqrt{2}\psi(u).$$

Entonces, la *Wavelet* Haar representa la información de la diferencia existente entre dos promedios al tiempo t. Se puede ajustar la *Wavelet* Haar a otros valores de t, solo es necesario cambiar la localización de $\psi(\cdot)$ de la siguiente manera $\psi_{1,t}(u) \equiv \psi(u-t)$

$$\psi_{1,t}(u) = \begin{cases} \frac{-1}{\sqrt{2}}, & t-1 < u \leq t; \\ \frac{1}{\sqrt{2}}, & t < u \leq t+1 \\ 0, & \text{o.c.} \end{cases} \quad (2.2.9)$$

Dado que $\psi_{1,t}(u)$ es la versión trasladada de $\psi(u)$ satisface las dos propiedades básicas de las *wavelets*. También se puede obtener información para distintas escalas λ considerando lo siguiente

$$\psi_{\lambda,t}(u) \equiv \frac{1}{\sqrt{\lambda}}\psi\left(\frac{u-t}{\lambda}\right) = \begin{cases} -\frac{1}{\sqrt{2\lambda}}, & t-\lambda < u \leq t; \\ \frac{1}{\sqrt{2\lambda}}, & t < u \leq t+\lambda; \\ 0, & \text{o.c} \end{cases} \tag{2.2.10}$$

Las propiedades de las ecuaciones 2.1.1 y 2.1.2 también se cumple en este caso dado que todo crecimiento por encima de cero es compensado con un decrecimiento. Utilizando esta *wavelet*, se obtiene que

$$W(\lambda,t) \equiv \int_{\infty}^{\infty} \psi_{\lambda,t}(u)x(u)\,du \propto D(\lambda,t). \tag{2.2.11}$$

El conjunto de variables $\{W(\lambda,t) : \lambda > 0, -\infty < t < \infty\}$ es conocido como la transformada continua de *Wavelet* Haar de $x(\cdot)$ y para su interpretación, como se ha visto, $W(\lambda,t)$ es proporcional a la diferencia entre dos promedios adyacentes de escala λ; existen distintos tipos de *wavelets* y se puede lograr realizar una interpretación similar.

El principal propósito de la transformada continua de *Wavelet* (TCW) es preservar toda la información de $x(\cdot)$. Si $\psi(\cdot)$ satisface la condición de admisibilidad, es decir, si su transformada de Fourier

$$\Psi(f) = \int_{-\infty}^{\infty} \psi(u)e^{-i2\pi fu}\,du, \tag{2.2.12}$$

es tal que

$$c_\psi \equiv \int_0^{\infty} \frac{|\Psi(f)|^2}{f}\,df \tag{2.2.13}$$

satisface que $0 < c_\psi < \infty$ y si la señal de $x(\cdot)$

$$\int_{-\infty}^{\infty} x(u)^2\,du < \infty. \tag{2.2.14}$$

Entonces se podrá reconstruir la función $x(\cdot)$ a partir de su transformada continua de

Wavelet

$$x(u) = \frac{1}{C_\psi} \int_0^\infty \left[\int_{-\infty}^\infty W(\lambda, t) \frac{1}{\sqrt{\lambda}} \psi \left(\frac{u-t}{\lambda} \right) \, dt \right] \frac{d\lambda}{\lambda^2}. \tag{2.2.15}$$

La TCW permite presentar la información acerca de una señal $x(\cdot)$ de una manera diferente, da la posibilidad de conocer detalles sobre la estructura de una serie, la cual no es posible ver de inmediato en su representación habitual.

2.3. Transformada discreta de *Wavelet*

En distintas áreas y principalmente en la que se centra este trabajo, los datos son presentados por un número finito de valores y por lo tanto es muy útil considerar la versión discreta de la transformada de *Wavelet*. La función *wavelet* está definida por una escala λ y una variable t que indica la ubicación, en el caso de este estudio, el tiempo.

$$\psi_{\lambda,t}(u) = \frac{1}{\sqrt{\lambda}} \psi \left(\frac{u-t}{\lambda} \right) \tag{2.3.1}$$

Los parámetros λ y t tienen cambios de manera continua, para discretizar estos valores se puede utilizar una discretización algorítmica de la escala λ y unir esto con el tamaño de pasos entre las localizaciones t. A partir de este tipo de discretización se obtiene

$$\psi_{m,n}(u) = \frac{1}{\sqrt{\lambda_0^m}} \psi \left(\frac{u - n t_0 \lambda_0^m}{\lambda_0^m} \right), \tag{2.3.2}$$

donde los enteros m y n controlan la dilatación (los cambios de escala) y traslación (la ubicación) de la *wavelet*; λ_0 es un parámetro fijo de dilatación (escala) y t_0 de localización. En la ecuación 2.3.2, el tamaño de los pasos de traslación, $\Delta t = t_0 \lambda_0^m$, es directamente proporcional a la escala *wavelet*, λ_0^m.

Comúnmente los parámetros que se eligen para la *wavelet* discreta son $1/2$ para λ_0 y 1 para t_0. Este proceso de escalamiento y dilatación logarítmica de potencia de 2 es conocida como "arreglo de red diádica". Sustituyendo $\lambda_0 = 1/2$ y $t_0 = 1$ en 2.3.2

se obtiene la *wavelet* de red diádica representada por

$$\psi_{m,n}(u) = 2^{m/2}\psi(2^m u - n). \tag{2.3.3}$$

Utilizando la *wavelet* de 2.3.3, se puede obtener la transformada discreta de *Wavelet* de una señal $x(\cdot)$

$$W_{m,n} = \int \psi_{m,n}(u)x(u)\,du = \int 2^{m/2}\psi(2^m u - n)x(u)\,du. \tag{2.3.4}$$

La cuestión fundamental es la construcción de $x(\cdot)$ a partir de la transformada discreta de *Wavelet* $W_{m,n}$ a través de la relación

$$x(u) = \sum_m \sum_n W_{m,n}\psi_{m,n}(u). \tag{2.3.5}$$

La ecuación 2.3.5 es justificada si la *wavelet* discreta $\psi_{m,n}(\cdot)$ es ortonormal y completa. La completitud de $\psi_{m,n}(\cdot)$ implica que cualquier función $x(\cdot) \in L^2(\mathbb{R})$ puede ser expresada por

$$x(u) = \sum_m \sum_n c_{m,n}\psi_{m,n}(u) \tag{2.3.6}$$

con los coeficiente apropiados $c_{m,n}$. Por lo tanto, la ortormalidad

$$\int \psi_{m,n}(u)\psi_{m',n'}(u)\,du = \delta_{m,n} \tag{2.3.7}$$

da como resultado $c_{m,n} = W_{m,n}$ en 2.3.6 porque

$$\begin{aligned} W_{m,n} &= \int x(u)\psi_{m,n}(u)\,du \\ &= \int \left[\sum_{m'}\sum_{n'} c_{m',n'}\psi_{m',n'}(u)\right]\psi_{m,n}(u)\,du \\ &= \sum_{m'}\sum_{n'} c_{m',n'}\int \psi_{m,n}(u)\psi_{m',n'}(u)\,du \\ &= \sum_{m'}\sum_{n'} c_{m',n'}\delta_{m,n}\delta_{m',n'} \\ &= c_{m,n} \end{aligned}$$

La transformada inversa de la fórmula 2.3.5 es válida para la clase de conjuntos de *wavelets* discretas $\{\psi_{m,n}(\cdot)\}$ que sean ortonormales y completos.

2.4. Análisis de multiresolución

La construcción de un conjunto apropiado de *wavelets* discretas $\{\psi_{m,n}(\cdot)\}$ ortonormal y completo, se basa en el concepto de análisis de multiresolución.

El análisis de multiresolución involucra una particular clase de espacios de funciones que genera una estructura anidada de subespacios de $L^2(\mathbb{R})$, lo cual permite construir un conjunto de funciones ortonormal completo; es decir, base ortonormal para $L^2(\mathbb{R})$. La base ortonormal resultante es la *wavelet* discreta $\psi_{m,n}(\cdot)$ que genera de la ecuación 2.3.5. El análisis de multiresolución implica un conjunto de espacios de funciones, una sucesión $\{V_j : j \in \mathbb{Z}\}$ de subespacios cerrados de $L^2(\mathbb{R})$, donde los subespacios V_j satisface las siguientes condiciones:

1. $V_0 \subset V_1 \subset V_2 \cdots \subset L^2(\mathbb{R})$.
2. $\cap_j V_j = \{0\}$.
3. $f(u) \in V_j$ si y solo si $f(2u) \in V_{j+1} \quad \forall j \in \mathbb{Z}$.

4. Existe una función $\phi(u) \in V_0$ tal que el conjunto $\{\phi(u-n),\ n \in \mathbb{Z}\}$ es una base ortonormal para V_0.

La función $\phi(\cdot)$ es llamada función escala (o *wavelet* padre), con la función $\phi(u)$ se puede establecer un análisis de multiresolución $\{V_j\}$, por definición el espacio de funciones V_0 generado por la base ortonormal $\{\phi(u-n),\ n \in \mathbb{Z}\}$ y después se forman otros subespacios V_j $(j \neq 0)$ utilizando la propiedad 3. Si ocurre esto la función $\phi(\cdot)$ genera el análisis de multiresolución $\{V_j\}$.

Ejemplo 2.4.1. Sea V_m el espacio de todas las funciones en $L^2(R)$ que son constantes en cada intervalo $[2^{-m}n, 2^{-m}(n+1)]$ para todo $n \in \mathbb{Z}$. Bajo está definición las propiedades de la $1-3$ se cumplen; el conjunto $\{\phi(u-n),\ n \in \mathbb{Z}\}$ definido por

$$\phi(u) = \begin{cases} 1, & 0 \leq u \leq 1; \\ 0, & \text{o.c.} \end{cases} \tag{2.4.1}$$

satisface la condición 4. Por lo tanto, cualquier función $f \in V_0$ puede ser expresada por

$$f(u) = \sum_n c_n \phi(u-n),$$

con los escalares adecuados c_n, entonces, los espacios V_m integran el análisis de multiresolución generado por la función escala 2.4.1.

2.4.1. Descomposición ortogonal

La importancia del análisis multiresolución reside en su capacidad de construir una base ortonormal para $L^2(\mathbb{R})$. Para probar esto primero se debe tener en cuenta que un análisis multiresolución $\{V_j\}$ satisface la relación

$$V_0 \subset V_1 \subset V_2 \subset \cdots \subset L^2.$$

Sea un espacio W_0 definidos como el complemento ortogonal de V_0 y V_1

$$V_1 = V_0 \oplus W_0. \tag{2.4.2}$$

El espacio W_0 es llamado espacio *wavelet* de orden cero, la relación 2.4.2 se puede extender a

$$V_2 = V_1 \oplus W_1 = V_0 \oplus W_0 \oplus W_1 \tag{2.4.3}$$

o generalizando, se tiene

$$L^2 = V_\infty = V_0 \oplus W_0 \oplus W_1 \oplus W_2 \oplus \cdots, \tag{2.4.4}$$

donde V_0 es el espacio inicial generado por el conjunto de funciones $\{\phi(u-n),\ n \in \mathbb{Z}\}$. La expresión 2.4.4 es definida como la descomposición ortogonal del espacio L^2 e indica que cualquier función $x \in L^2(\mathbb{R})$ se puede descomponer en una suma infinita de $g_j \in W_j$ con $j \in \mathbb{Z}$

$$x(u) = g_0(u) + g_1(u) + g_2(u) + \cdots \tag{2.4.5}$$

Por 2.4.2 y 2.4.3) se tiene que

$$W_0 \subset V_1 \text{ y } W_1 \subset V_2. \tag{2.4.6}$$

Por la definición del análisis de multiresolución $\{V_j\}$, se sigue que

$$f(u) \in V_1 \Longleftrightarrow f(2u) \in V_2,$$

por lo tanto

$$f(u) \in W_0 \Longleftrightarrow f(2u) \in W_1. \tag{2.4.7}$$

Además, la condición 4 muestra que

$$f(u) \in W_0 \Longleftrightarrow f(u-n) \in W_0 \text{ para cualquier } n \in \mathbb{Z} \tag{2.4.8}$$

Si se asume que existe una función $\psi(u)$ con la que se forma una base ortonormal

$\{\psi(u-n),\ n \in \mathbb{Z}\}$ para el espacio W_0. A partir de la notación

$$\psi_{0,n}(u) \equiv \psi(u-n) \in W_0,$$

y de 2.4.7 y 2.4.8, la ecuación

$$\psi_{1,n}(u) = \sqrt{2}\psi(2u-n)$$

funciona como una base ortonormal para W_1. El término $\sqrt{2}$ se agrega para conservar la condición de normalidad

$$\int \psi_{0,n}(u)^2\, du = \int_{-\infty}^{\infty} \psi_{1,n}(u)^2\, du = 1$$

Si se repite este procedimiento se obtiene la ecuación generalizada

$$\psi_{m,n}(u) = 2^{m/2}\psi(2^m u - n) \tag{2.4.9}$$

que forma una base ortonormal para el espacio W_m. Aplicando estos resultados a la expresión 2.4.5 se tiene para cualquier $x \in L^2(\mathbb{R})$,

$$\begin{aligned} x(u) &= g_0(u) + g_1(u) + \cdots \\ &= \sum_n c_{0,n}\psi_{0,n}(u) + \sum_n c_{1,n}\psi_{1,n}(u) + \cdots \\ &= \sum_m \sum_n c_{m,n}\psi_{m,n}(u). \end{aligned} \tag{2.4.10}$$

Entonces, $\psi_{m,n}(u)$ representa una base ortonormal para $L^2(\mathbb{R})$. En este sentido, cada W_m es llamado como espacio *wavelet* y la función $\psi(\cdot)$ es llamada *wavelet* madre. Las funciones en V_m son obtenidos a partir de las existentes en V_0 a través del escalamiento 2^m. Este resultado genera que a partir de la función

$$\phi_{0,n} \equiv \phi(u-n) \in V_0$$

se obtiene

$$\phi_{m,n}(u) = 2^{m/2}\phi(2^m u - n),\ m \in \mathbb{Z}$$

que es una base ortonormal para V_m.

Por medio de los conceptos anteriores se ha obtenido una base ortonormal que consiste en funciones escala $\psi_{m,n}(u)$ y *wavelets* $\phi_{m,n}(u)$ que generan a $L^2(\mathbb{R})$, dado que

$$L^2 = V_{m_0} \oplus W_{m_0} \oplus W_{m_0+1} \oplus \cdots ,$$

cualquier función $x(u) \in L^2(\mathbb{R})$ se puede representar como la siguiente serie

$$x(u) = \sum_n V_{m_o,n}\phi_{m_0,n}(u) + \sum_n \sum_m W_{m,n}\psi_{m,n}(u). \tag{2.4.11}$$

2.4.2. Momentos de desvanecimiento

Una propiedad importante de las *wavelets* son lo momentos de desvanecimiento, una función $\psi \in L^2(\mathbb{R})$ posee m momentos de desvanecimiento si satisface la función 2.4.12

$$\int_{\mathbb{R}} u^l \psi(u) du = 0, \tag{2.4.12}$$

para $l = 0, \ldots, m-1$.

Esto significa que si una *wavelet* tiene m momentos de desvanecimiento, entonces, para este fin, si una serie se compone por algunas discontinuidades y en los demás casos es "regular" la descomposición se centrará en las discontinuidades y en los demás casos los coeficientes de las *wavelet* serán pequeños o exactamente cero si en ese punto el polinomio es de orden inferior a m.

2.5. Representación matricial

La ecuación 2.4.11 se puede representar por medio de matrices, sea $\mathbf{X}$ un vector de dimensión N cuyos elementos pertenecen a una serie de tiempo real valuada ("señal") $\{X_t : t = 0, \ldots, N-1\}$, donde $N = 2^J$. La transformada de *Wavelet* discreta de nivel J de $\mathbf{X}$ es una transformación ortonormal dada por $\mathbf{W} = \mathcal{W}\mathbf{X}$, $\mathbf{W}$ es un

vector de dimensión N que contiene los coeficientes de la TDW y $\mathcal{W}$ es una matriz real valuada de $N \times N$ que define la TDW. Los coeficientes TDW $\mathbf{W}$ y la matriz $\mathcal{W}$ puede ser dividido de la siguiente manera

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_J \\ \mathbf{V}_J \end{bmatrix}, \qquad \mathcal{W} = \begin{bmatrix} \mathcal{W}_1 \\ \mathcal{W}_2 \\ \vdots \\ \mathcal{W}_J \\ \mathcal{V}_J \end{bmatrix}. \tag{2.5.1}$$

Por lo tanto, $\mathbf{W}_j = \mathcal{W}_j\mathbf{X}$ y también $\mathbf{V}_J = \mathcal{V}_J\mathbf{X}$. El vector $\mathbf{W}_j$ es de dimensión $N_j \equiv N/2^j$; $\mathcal{W}_j$ es una matriz de dimensión $N_j \times N$; $\mathbf{V}_J$ es un vector de dimensión N_J asociado a coeficientes escala y $\mathcal{W}_J$ es una matriz de dimensión $N_j \times N$. Por lo tanto la señal $\mathbf{X}$ puede ser expresada en términos de $\mathbf{W}$ y de esta manera obtener la representación matricial de la ecuación 2.4.11

$$\mathbf{X} = \mathcal{W}^T\mathbf{W} = \sum_{j=1}^{J} \mathcal{W}_j^T\mathbf{W}_j + \mathcal{V}_J^T\mathbf{V}_J = \sum_{j=1}^{J} \mathcal{D}_j + \mathcal{S}_J \tag{2.5.2}$$

Entonces 2.5.2 se define como un análisis de multiresolución de $\mathbf{X}$, es decir, una descomposición aditiva en términos de vectores $D_j = \mathcal{D}_j = \mathcal{W}_j^T\mathbf{W}_j$ y $\mathcal{S}_J = \mathcal{V}_J\mathbf{V}_J$ de dimensión N. En la práctica la matriz $\mathcal{W}$ no se forma explícitamente, pero $\mathbf{W}$ es calculado utilizando un algoritmo "piramidal" que utiliza un filtro *wavelet* y un filtro de escala. Por definición, un filtro $\{h_l : l = 0, \ldots, L-1\}$ con una dimensión fija $L \in \mathbb{Z}$ (implica que $h_0 \neq 0$ y $h_{L-1} \neq 0$) es llamado filtro *wavelet* si

$$\sum_{l=0}^{L-1} h_l = 0 \text{ y } \sum_{l=0}^{L-1} h_l h_{l+2n} = \begin{cases} 1, & \text{si } n = 0, \\ 0, & \text{si } n \neq 0. \end{cases} \tag{2.5.3}$$

La segunda suma muestra la propiedad ortonormal de un filtro *wavelet*. El filtro escala es definido en términos de el filtro *wavelet* por medio de la relación

$$g_l = (-1)^{l+1} h_{L-1-l}.$$

Este filtro satisface las condiciones

$$\sum_{l=0}^{L-1} g_l g_{l+2n} = \begin{cases} 1, & \text{si } n = 0, \\ 0, & \text{o.c. ;} \end{cases} \qquad \text{y} \qquad \sum_{l=0}^{L-1} g_l h_{l+2n} = 0 \quad \forall n. \tag{2.5.4}$$

Sin perder generalidad , se puede asumir que $\sum_l g_l = \sqrt{2}$. El filtro *wavelet* $\{h_l\}$ también es nombrado filtro de paso alto con una banda de paso $\frac{1}{4} \leq |f| \leq \frac{1}{2}$, mientra que el filtro escala $\{g_l\}$ como filtro de paso bajo con una banda de paso $0 \leq |f| \leq \frac{1}{4}$. Esto es porque cada filtro tiene una banda de paso que cubre la mitad y toda la banda de frecuencias.

Con $\{h_l\}$ y $\{g_l\}$ definidos, el algoritmo piramidal para la etapa j consiste de filtros de N_{j-1} elementos de

$$\mathbf{V}_{j-1} = [V_{j-1,0}, V_{j-1,1}, \ldots, V_{j-1,N_{j-1}-1}]^T$$

y reteniendo los valores filtrados con indices impares, se produce el $j-$ésimo nivel de coeficientes *wavelet* y escala

$$W_{j,t} = \sum_{l=0}^{L-1} h_l V_{j-1,2t+l-1 \bmod N_{j-1}}, \quad V_{j,t} = \sum_{l=0}^{L-1} g_l V_{j-1,2t+1-l \bmod N_{j-1}},$$

$$t \in \{0, \ldots, N_j - 1\}.$$

Estos son elementos de $\mathbf{W_j}$ y $\mathbf{V_J}$, el procedimiento de retener los elementos impares es llamado "disminución de resolución por dos". Con $\mathbf{V}_0 = \mathbf{X}$, se inicia el algoritmo piramidal con $j = 1$ y, después repitiendo el algoritmo con $j = 2, 3, \ldots, J$, se obtienen todo los coeficiente necesarios para formar $\mathbf{W}$, es decir, los J vectores de coeficiente *wavelet* $\mathbf{W}_1, \ldots, \mathbf{W}_J$ y un único vector de coeficientes escala $\mathbf{V}_J$ ya que los otros vectores $\mathbf{V}_1, \ldots, \mathbf{V}_{J-1}$ son únicamente considerados en los cálculos intermedios.

Al utilizar el algoritmo piramidal, $\mathbf{W_j}$ y $\mathbf{V}_j$ se pueden obtener directamente a

partir de $\mathbf{X}$ por medio de lo siguiente

$$W_{j,t} = \sum_{l=0}^{L_j-1} h_{j,l} X_{2^j(t+1)-1-l \bmod N}, \quad V_{j,t} = \sum_{l=0}^{L_j-1} g_{j,l} X_{2^j(t+1)-1-l \bmod N},$$

donde $\{h_{j,l}\}$ y $\{g_{j,l}\}$ son el equivalente al $j-$ésimo nivel de filtros *wavelet* y escala, cada uno con un largo $L_j \equiv (2^j - 1)(L-1) + 1$. El filtro $\{h_{j,l}\}$ es un filtro paso alto con un paso de banda dado por $1/2^{j+1} \leq |f| \leq 1/2^j$, mientras $\{g_{j,l}\}$ es un filtro de paso bajo con un paso de banda de $0 \leq |f| \leq 1/2^{j+1}$.

2.6. Reducción del ruido vía *wavelets*

Para explicar los pasos para la reducción del ruido, sean N muestras con ruido de una función f tal que

$$y_k = f(t_k) + \epsilon_k \quad k = 1, \ldots, N$$

$$\text{con } t_k = \frac{k-1}{N}, \epsilon_k \sim N(0, \sigma^2).$$

El procedimiento para la reducción del ruido por medio de la transformada de *Wavelet* se puede resumir de la siguiente manera:

1. Calcular la transformada discreta de *Wavelet* $\mathbf{y} = \mathcal{W}y$ de la señal y.

2. Aplicar una regla de reducción no lineal $(\delta_\lambda(\cdot))$ a los coeficiente *wavelet* $\mathbf{y}$ para obtener:

$$\hat{\mathbf{w}}_k = \delta_{\lambda_k} \hat{\sigma}(\mathbf{y_k}),$$

   donde $\hat{\sigma}$ es un estimador de la desviación estándar del ruido ϵ.

3. Calcular la inversa de la transformada de *Wavelet*

$$\hat{f} = \mathcal{W}^{-1} \hat{\mathbf{w}}$$

La ortogonalidad de la TDW tiene una consecuencia estadística fundamental, $\mathcal{W}$ transforma ruido blanco, entonces si $(\mathbf{y}_k)$ son los coeficientes *wavelet* de $(y_k)_{k=0}^{N-1}$ y $\mathbf{w}_k$ son los coeficientes de $(f(t_k))$, entonces

$$\mathbf{y}_k = \mathbf{w}_k + \epsilon_k,$$

donde $\epsilon_k \sim N(0, \sigma^2)$ *i.i.d.* es una sucesión de ruido, esto implica que los coeficientes *wavelet* de una muestra con ruido son en sí mismas versiones con ruido de coeficientes *wavelet*.

El objetivo final es estimar el vector $f = (f(t_i))_{i=1}^{N}$ con el mínimo error cuadrático medio, es decir, encontrar un estimador $\hat{f}$ dependiente de $y_1, y_2, \ldots, y_n$ con el mínimo error. Para cumplir este objetivo se parte de que $\mathcal{W}^T\mathcal{W} = \mathcal{W}\mathcal{W}^T = I$, se sigue entonces

$$\|y\|^2 = \mathbf{y}^T\mathbf{y} = (\mathcal{W}y)^T\mathcal{W}y = y^T(\mathcal{W}^T\mathcal{W})y = y^Ty = \|y\|^2.$$

Esta es la relación de Parseval y a partir de lo anterior la siguiente relación también es válida

$$\|\hat{\mathbf{w}} - \mathbf{w}\|^2 = \left\|\hat{f} - f\right\|^2.$$

Por lo tanto $\mathcal{W}$ transforma estimadores de un dominio en estimadores de otro dominio. Si $\hat{\mathbf{w}}_k$ son estimadores de los coeficientes *wavelet*, entonces existe un estimador $\hat{f}$ de $f = f(t_k)$ en el otro dominio obtenido por $\hat{f} = \mathcal{W}^T\mathbf{w}$ y los errores siguen la relación de Parseval.

Antes de definir la función de contracción no lineal se necesita tomar en cuenta que la transformada de *Wavelet* descompone una señal en tiempo y frecuencia, por definición los coeficientes de la transformada serán más cercanos a cero si presentan ruido, entonces el problema de estimar $\hat{f}$ consiste en conservar los coeficientes significativamente diferentes de cero.

La idea anterior también se puede explicar diciendo que los coeficientes de mayor magnitud están integrados en la mayoría de los casos por señal y ruido, mientras que los de menor magnitud solo están integrados por ruido. Entonces, para estimar $\hat{f}$ se crea un estimador $\hat{\mathbf{w}}$ removiendo los coeficientes $\hat{\mathbf{y}}$ que son más pequeños que algún

límite (*threshold*) y conservando los que son de mayor magnitud. A este procedimiento se le llama en inglés "Thresholding". Para este procedimiento existen diversas reglas para conservar o eliminar los coeficientes, en (Donoho y Johnstone, 1994) se tienen dos definiciones *Hard* y *Soft*:

$$\hat{\mathbf{w}}_H = \delta_{\delta_H}(\mathbf{y}) = \mathbf{y}\mathbb{I}_{\{|\mathbf{y}|>\lambda\}} \tag{2.6.1}$$

$$\hat{\mathbf{w}}_S = \delta_{\lambda_S}(\mathbf{y}) = sgn(\mathbf{y})(|\mathbf{y}| - \lambda)\mathbb{I}_{\{|\mathbf{y}|>\lambda\}} \tag{2.6.2}$$

En el caso de *Hard* se tiende a tener una varianza grande debido a la discontinuidad de la función, para el caso *Soft* se tiende a tener un sesgo grande dada la reducción a cero de todos los coeficientes grandes. Para superar esos inconvenientes también en (Donoho y Johnstone, 1994) se tiene la función *Semisoft*:

$$\hat{\mathbf{w}}_{SS} = \delta_{\lambda_1,\lambda_2}(\mathbf{y}) = \begin{cases} 0, & \text{si } |\mathbf{y}| \leq \lambda_1, \\ sgn(\mathbf{y})\frac{\lambda_2(|\mathbf{y}|-\lambda_1)}{\lambda_2-\lambda_1}, & \text{si } \lambda_1 \leq |\mathbf{y}| \leq \lambda_2, \\ \mathbf{y}, & \text{si } |\mathbf{y}| > \lambda_2. \end{cases} \tag{2.6.3}$$

Con esta función se logra tener un equilibrio, para el modelo que se presentará más adelante no se consideró esta última función ya que el paquete de `R wavethresh` no la contenía, pero dada la naturaleza de los datos con la función *Hard* se lograron buenos resultados.

Capítulo 3

Técnicas de pronóstico

Los pronósticos son estimaciones de una o más variables que conforman un evento en el futuro por medio de su información actual o del pasado. Las estimaciones pueden ser de largo, mediano o corto plazo, en este trabajo se busca realizar un pronóstico de corto plazo. Las técnicas de pronóstico se dividen en dos, cualitativas y cuantitativas, en las primeras el método es subjetivo, la opinión o conocimiento de un experto; en las segundas se utilizan herramientas matemáticas y estadísticas para obtener una estimación.

Para los datos de contaminantes, se pueden encontrar referencias sobre distintos modelos para el pronóstico de los niveles, se pueden utilizar variables exógenas como otros contaminantes o variables meteorológicas.

Para la realización del modelo, donde se pretende probar las eficacia de la transformada de Wavelet, se van a utilizar las Redes Neuronales Artificiales, se debe considerar como una referencia indispensable para la realización de este trabajo el articulo *Wavelet transform-based artificial neural networks (WT-ANN) in* PM_{10} *pollution level estimation, based on circular variables*(Shekarrizfard *et al.*, 2011).

En este capítulo se presentará una breve y muy general descripción acerca del tema y en particular del tipo de Red Neuronal que se utilizará.

3.1. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales, en adelante RNA, se podrían definir como técnicas de procesamiento de datos que a través de modelos computacionales intentan imitar cierto funcionamiento del cerebro útil para el reconocimiento y clasificación de patrones. Desde luego de una forma limitada tanto en tamaño, habilidad como poder, pero en particular las RNA buscan replicar las siguientes características:

- Procesamiento paralelo: La totalidad de la neuronas intervienen al mismo tiempo en el cerebro, el procesamiento que realiza el cerebro para resolver un problema, caminar o encontrar la ruta adecuada lo distribuye en cada célula que lo compone y se transmite por medio de las complejas interconexiones entre cada célula. Por está razón cualquiera de estas actividades son muy complejas para un sistema artificial pero el cerebro lo procesa y ejecuta de una forma veloz.

- Adaptabilidad y Autoorganización: El cerebro se adapta y se organiza en base al aprendizaje que recibe, a las nueva experiencias o simplemente al paso del tiempo. El cerebro es capaz de funcionar a pesar de que diariamente se mueren neuronas, las iterconexiones tan importantes en el procesamiento, guardan paralelamente, al igual que como procesan, la información necesaria para que todo el sistema siga funcionando. Además no necesita una guía o un orden o algo preestablecido, en base a toda la información, experiencias y aprendizaje continuo. El cerebro sabe como realizar una actividad y cuando algo cambia sabe de que manera reorganizarse para buscar una solución.

Esto es lo que intenta imitar del cerebro una RNA, que el procesamiento sea paralelo, que se adapte y a partir de esto se autoorganice. Todo esto como se mencionó de una forma limitada en habilidad, tamaño y poder. Porque si se hablan de las aplicaciones estadísticas, se puede tener una red neuronal que haga un pronóstico, como en este caso, pero esa misma red no servirá para realizar un agrupamiento.

En definiciones más concretas una red neuronal es un sistema compuesto por dos conjuntos N y V y una función w, donde N es un conjunto de neuronas y V es un

conjunto $\{(i,j)|i,j \in N\}$ donde los elementos son las conexiones entre la neurona i y la neurona j. La función $w : V \longrightarrow \mathbb{R}$ define los pesos de conexión de la neurona i a la neurona j ($w_{i,j}$). Si los pesos son indefinidos o igual a 0 entonces la red no existe.

Los datos se transfieren entre neuronas por medio de las conexiones con los pesos que pueden activar o inhibir la señal. La neurona j puede tener varias neuronas conectadas a ella, es decir, que le transfieren su resultado. Al dirigirse a la neurona j, estos resultados $o_{i_1}, o_{1_2}, \ldots, o_{i_n}$ que las neuronas $i_1, i_2, \ldots, i_n$ transfiere a j, son recibidos por una función, llamada función de propagación, que convierte este vector de resultados junto con el vector de pesos para las conexiónes entre las neuronas $i_1, i_2, \ldots, i_n$ a la neurona j, en el dato de entrada ($nnet_j$) de la neurona j. La función de propagación utilizada comúnmente es la suma ponderada

$$nnet_j = \sum_{i \in I} (o_i \cdot w_{i,j}) \tag{3.1.1}$$

Una vez que los datos de entrada han llegado a la neurona j la neurona tiene una reacción, está reacción está definida por una función llamada función de activación $a_j(t)$. En la función de activación intervienen 3 elementos, los datos de entrada a la neurona j ($nnet_j$), el estado de activación anterior de la neurona j ($a_j(t-1)$) y un valor umbral (Θ_j) que establece un parámetro para definir el estado de activación. Entre algunas de las funciones que comúnmente se utilizan como función de activación en una RNA están la función logística, la función hiperbólica y la función binaria.

Por último la neurona j tiene una función de salida, por medio de la cual transfiere los datos procesados a otras neuronas que están conectadas a ella. Comúnmente se utiliza la función identidad como función de salida.

Estos son los componentes básicos de procesamiento de una RNA. Pero para esto se necesita un vector con los valores de entrada, una red con n neuronas de entrada tiene n valores de entrada y de todo esto se obtienen datos de salida, m neuronas de salida dan m valores de salida. Por otra parte, ninguna red funciona sin una estrategia de aprendizaje, esta es la forma por medio de la cual se entrena la red para que un dato de entrada dado produzca un dato de salida esperado.

3.1.1. Clasificación de acuerdo a su diseño

Las RNA se pueden clasificar de acuerdo a su topología o diseño.

Feedforward. Las redes *feedforward*, son también nombradas en español como redes hacia adelante o unidireccionales, están conformadas por capas separadas. Una capa de entrada, una capa de salida y una o varias capas de procesamiento, que son llamadas capas ocultas ya que los únicos valores “visibles"son los datos que se introducen y salen de la RNA. Las únicas conexiones permitidas en este diseño son hacia las neuronas de la siguiente capa. Dentro de este diseño se tiene una vertiente, las redes feedforward con conexiones de acceso directo, estas redes además de ser *feedforward*, pueden tener conexiones que se salten una o más capas, incluso saltarse las capas de procesamiento y llegar directo a la capa de salida.

Recurrentes. En términos de RNA, la recurrencia es un proceso donde una neurona es capaz de que su resultado influya sobre si misma. Que una red sea recurrente no significa que deje de establecer conexiones con la siguiente capa, como las *feedforward*, pero no solo tiene estas conexiones. Existen distintos tipo de diseños para las RNA recurrentes:

- Recurrente directa: Una neurona es capaz de hacer conexión consigo misma.
- Recurrente indirecta: Basándose en el diseño *feedforward* se agregan las conexiones de la capa posterior a la capa anterior y de está manera la neurona se conecta consigo misma de manera indirecta, es decir, si la neurona i de la capa 1 está conectada hacia adelante a la neurona j de la capa 2, entonces la neurona j de la capa 2 se conectará en sentido contrario a la neurona i de la capa 1.
- Recurrente lateral: También se establece una conexión indirecta, pero en este caso es con las neuronas que pertenecen a la misma capa.
- Con completa interconexión. Permite conexiones entre todas las neuronas, pero no consigo misma, es decir, no se permite la recurrencia directa.

3.1.2. Clasificación de acuerdo a su tipo de aprendizaje

El tipo de aprendizaje que tiene una red es la clave para que la RNA pueda adaptarse y autoorganizarse con el fin de obtener el valor de salida deseado, el tipo de aprendizaje está definido por el logaritmo que la RNA sigue.

Aprendizaje no supervisado. Es el método que se puede llegar a acerca más al proceso que sigue el cerebro, pero no es el más adecuado en todos los casos. Solo se proveen valores de entrada y la red intenta encontrar patrones similares para clasificar estos datos.

Aprendizaje supervisado. En este método se tienen datos de entrada, así como los datos que se espera obtenga la red. Entonces los datos de entrada se procesan y al final se comparan los resultados, con los resultados esperados y la red puede cambiar sus pesos para acercarse al objetivo. Este método es el más práctico.

A continuación se presentará una de las RNA más utilizadas, el perceprón multicapa, posteriormente se describirá un versión de esta RNA que es la que se utilizará en el modelo realizado en este trabajo.

3.1.3. Perceptrón multicapa

Un perceptrón multicapa (MLP) es una red *feedforward* de aprendizaje supervisado en capas, donde cada neurona esta integrada por una función de activación no lineal que comúnmente es continuamente diferenciable. Las funciones de activación más utilizados para este modelo son la función sigmoide y la función tangente hiperbólica. Las neuronas están organizadas en capas la capa de entrada, una o más capas ocultas y la capa de salida. Un modelo MLP ofrece un mapeo no lineal entre los datos de entrada y la salida.

Para transformar este modelo en un modelo de predicción, se podría repre-

sentar al modelo MLP como una función no lineal f, en donde los valores de entrada se toman como el vector independiente $\bar{x} = (x_1, \ldots, x_n)$ y el valor de salida como la variable dependiente y, entonces se tendría la forma

$$y = f(x_1, \ldots, x_n). \tag{3.1.2}$$

A partir de esto si el problema se traslada a la predicción de una serie de tiempo, se puede tomar al vector de entrada como las observaciones anteriores y considerar la predicción al tiempo t como el dato de salida la ecuación 3.1.2. Entonces se tendría a la siguiente ecuación como la representación de un modelo de pronóstico de series de tiempo

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-n}). \tag{3.1.3}$$

El modelo se convierte en un equivalente de un modelo no lineal autorregresivo y la RNA pasa a ser un RNA autorregresiva.

Capítulo 4

Desarrollo del modelo

A partir de la transformada de *Wavelet* y las RNA se busca establecer un pronóstico para las PM_{10} en la estación FAC, situada en el Noroeste de la ZMVM. El pronóstico que se busca es a corto plazo, en especifico del promedio 24 horas del día siguiente al que se estén tomando los datos.

Datos. Todas las tabla de datos se obtuvieron de la página de internet del SIMAT [1], los datos son públicos y para la información horaria se tienen dos distintos formatos. En el primer caso se tienen archivos separados por año, en donde de forma apilada se presentan los valores horarios de cada contaminante para cada estación. Cuentan con datos desde el año 1986 y los nombres de las estaciones están homologados al último nombre establecido. Por otra parte el SIMAT cuenta con un aplicativo de consulta donde se puede elegir el contaminante, estación y forma de reporte que se desea.

Para realizar este trabajo se utilizaron los datos apilados de todas las estaciones y todos los contaminantes del periodo 2005-2014, los datos fueron manejados en el programa `R`, se utilizó el paquete `reshape2` para transponer los valores y posteriormente se eligieron las estaciones cuyos datos se iban a utilizar para el análisis previo (FAC, MER, SAG, TAH y PED). Los datos de cada estación se colocaron en un archivo

[1] http://www.aire.df.gob.mx

diferente.

El SIMAT reporta datos de 1 a 24 horas, esto es una dificultad porque la hora 24 el programa la toma como la primer hora del día siguiente. Por esta razón se realizó una modificación para que se reflejaran de 0 a 23 horas, también es importante asegurarse que al colocar la fecha, día y hora en el formato que `R` lo requiere se coloque la Zona Horaria del meridiano de Greenwich, de lo contrario se perderán las horas donde se realizan los cambios de horario de verano e invierno.

Como se mencionó los datos se obtuvieron de forma horaria, para realizar los análisis previos y la construcción del modelo se tuvieron que calcular los promedios 24 horas, trimestral, anual y/o móvil 8 horas. Esto fue muy sencillo gracias al paquete `openair`[2], este paquete tiene muchas funciones indispensables para el análisis de datos acerca de la calidad de aire. Por está razón en la siguientes lineas se detallaran algunas funciones que fueron utilizadas.

Para utilizar este paquete es indispensable tener una variable de tiempo, es conveniente tener como nombre de las columnas de los contaminare la abreviación que es utilizada comúnmente (O3, SO2, etc.) ya que el paquete reconoce el nombre y cuándo se realizan las gráficas coloca los subíndices de forma automática. Aunque sino se tiene de esta manera los gráfico tienen opciones para solucionarlo.

Para el promedio móvil se utilizó la función `rollingMean`, con está función se crea una nueva variable en la tabla de datos que contiene el promedio móvil del contaminante seleccionado, se puede elegir el número de horas que se quieren utilizar para el promedio móvil, el nuevo nombre que se desea dar a la variable y el mínimo porcentaje de datos que se necesita para realizar el promedio.

Para los promedios 24 horas, trimestral y anual se utilizó la función `timeAverage`, esta función además de obtener el promedio puede obtener el máximo, el mínimo, percentiles, la frecuencia, la mediana o la desviación estándar; se puede elegir entre distintos periodos: día, semana, mes, trimestre, año o incluso estación del año. Si existen datos de la dirección del viento, que es un dato circular, se le puede indicar al programa que

[2]Ver http://www.openair-project.org/ para mayor referencia.

para este caso no calcule el estadístico escalar sino el estadístico circular. También se tiene la opción de establecer el mínimo porcentaje de datos requeridos para calcular los valores resultantes, esto entre otras opciones.

El paquete `openair` facilita mucho el tratamiento, análisis y visualización de datos acerca de la calidad del aire, está hecho específicamente para este tipo de datos y por lo tanto tiene todas las opciones requeridas. Incluso tiene funciones particulares para importar datos directamente de repositorios en Reino Unido de la calidad del aire de este país, de donde es originario este proyecto.

En este trabajo, por no ser el tema central, no se explotaron todas las posibilidades que `openair` ofrece, en el capítulo 1 se muestran algunos gráficos básicos que se pueden realizar con las funciones `timePlot` y `summaryPlot` con algunas adecuaciones obtenidas del paquete `latticeExtra` y para ampliar el panorama, en las figuras 4.1, 4.2 y 4.3 se muestran los promedios diarios en formato calendario de los años 2005, 2013 y 2014 de las PM_{10}.

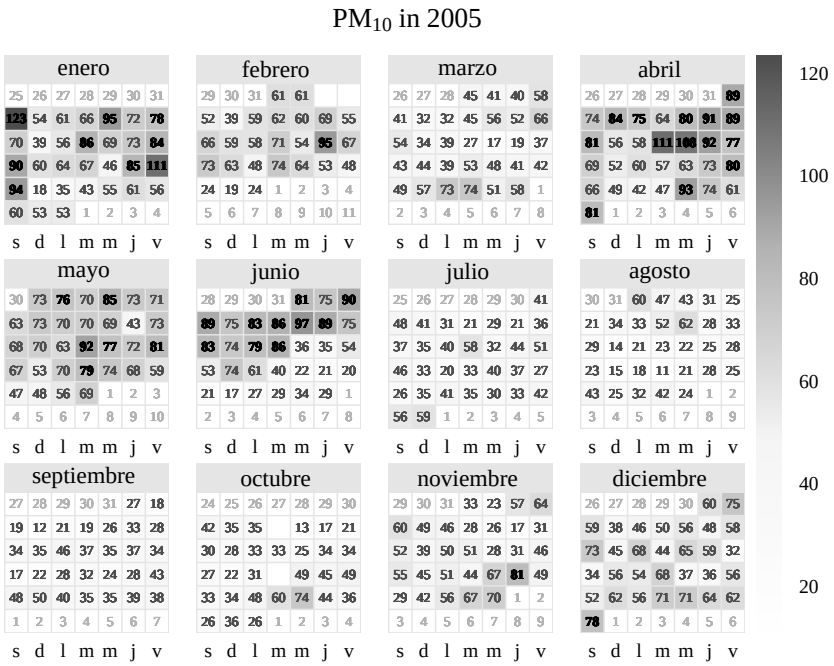

Figura 4.1: Calendario 2005 - Concentraciones PM_{10}

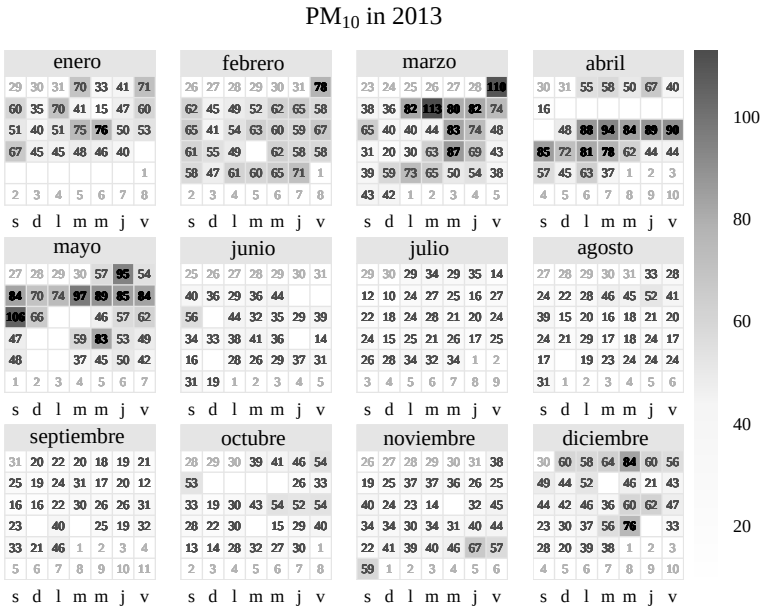

Figura 4.2: Calendario 2013 - Concentraciones PM_{10}

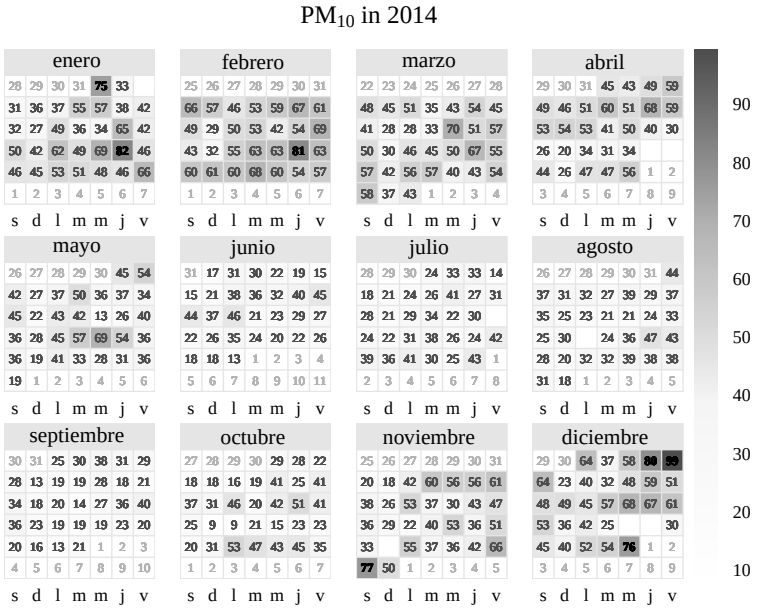

Figura 4.3: Calendario 2014 - Concentraciones PM_{10}

Se muestra en blanco y sin número los días de los que no es posible obtener información ya que no tienen al menos el 75 % de datos válidos, en negritas el valor del día si sobrepasa o es igual al máximo permitido en la NOM vigente. Esto es posible con la función `calendarPlot` y gracias a esto con solo un vistazo se puede detectar la evolución de los niveles, las temporadas donde se encuentran los máximos valores para este contaminantes y los máximos que se han alcanzado en cada año. Es solo una muestra de muchas otras opciones que se pueden realizar con este paquete.

4.1. Serie de tiempo a pronosticar

La Serie de Tiempo que se decidió pronosticar es el promedio 24 horas de las PM_{10} en la estación FAC. Debido a que se aplicó la transformada de *Wavelet*, el número de registros requeridos tuvo una limitante, debió ser igual a una potencia de dos.

Se eligieron los últimos dos años (2013-2014) de los datos obtenidos en promedio 24 horas. Posterior a esto se contó el número de datos no válidos que tenía la serie, estos significaban del 6 % del total. Por ser un valor inferior al 10 % se consideró viable continuar con el procedimiento excluyendo estos valores. Entonces, de las 685 observaciones restantes se seleccionaron las últimas 512 ($n = 2^9$) y esto dio como resultado la serie de tiempo a pronosticar que se muestra en la figura 4.4

4.2. Selección del modelo

La meta fue obtener dos modelos, uno basado en Redes Neuronales Artificiales y otro donde primero se redujo el ruido de la serie por medio de la Transformada de *Wavelet* y posteriormente se modelaron estos datos con una Red Neuronal Artificial.

Los paquetes de `R` que se utilizaron fueron `wavethresh` para las *Wavelets* y `tsDyn` para las RNA autorregresivas.

Los datos de entrada de las RNA se dividieron en 2 conjuntos, uno de entre-

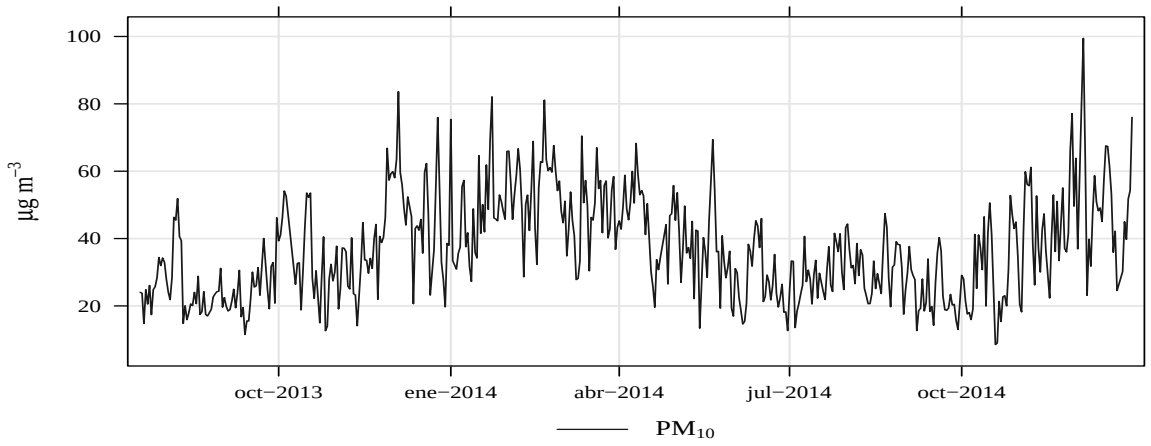

Figura 4.4: Serie de tiempo a pronosticar - Concentraciones PM_{10} estación FAC

namiento y otro de prueba, la división fue de 70 % - 30 % respectivamente. Es decir, los primeros 307 datos se utilizaron para elegir la Red Neuronal y los últimos 205 para probarla. Para todos los casos se escaló la serie, esto con el fin de que los pesos de la red se ajustaran de una forma más veloz, esto no afectó en algo la estructura de los datos ni agregó supuestos.

Para las RNA no existe una técnica o procedimiento preestablecido para que con solo saber las características de los datos de entrada se pueda definir la mejor arquitectura de la red, de igual manera ocurre con las *Wavelets*; se puede suponer cuales son los mejores parámetros que se deben utilizar pero es necesario probar y comparar. Por esta razón se realizaron varios modelos, en ambos casos, al final se comparó su coeficiente de determinación (R^2) y así se obtuvo el mejor modelo.

Se puede llegar a tener un modelo con un coeficiente de determinación "bueno" en el segmento de entrenamiento y un resultado muy diferente en el segmento de prueba, esto ocurre cuando la red se sobreajusta, por un número elevado de neuronas en capa oculta, o se sobreentrena. Cuando esto ocurre la red pierde capacidad de generalización, esto quiere decir que se "ajusta" tanto a los datos de entrada que no logra guardar su estructura general. Por esta razón para elegir los modelos se consideró ambos coeficientes

de determinación.

Para el caso Wavelet+RNA, es un modelo compuesto y se eligió el mejor modelo en función al resultado final. Es decir, se eligió la *wavelet* que nos permitiera tener el mejor modelo de pronóstico, cuidando que está *wavelet* mantuviera la estructura de los datos.

4.2.1. Modelos RNA

Para seleccionar el modelo de pronóstico de la serie sin reducción del ruido se utilizó como primer filtro la función `selectNNET` del paquete `tsDyn`, con el cual se realizaron todas las RNA autorregresivas para ambos modelos. Está función muestra los resultados de los criterios BIC y AIC para los modelos incluidos en los parámetros de la función con el fin de compararlos y ordena de acuerdo al criterio AIC, en esta función se elige el número de neuronas de entrada, en este caso atrasos o *lags*, que se desean utilizar y el número de neuronas ocultas que se desea comparar.

Las redes neuronales que se pueden hacer con este paquete son del tipo Perceptrón Multicapa incluidas en el paquete más popular de `R` de redes neuronales `nnet`, `tsDyn` solo modifica algunas opciones con el fin de realizar una RNA autorregresiva. Solo tiene una neurona de salida, por esta razón solo se establece un pronóstico a corto plazo y en específico solo de 1 día.

En este caso se realizaron pruebas de 1 a 8 *lags*, no se incluyeron más *lags* porque los resultados no mejoraron al aumentar este valor. Para cada valor de número de neuronas de entrada se probaron modelos de 1 a 17 neuronas en capa oculta por medio de la función antes mencionada.

En el segundo filtro se probaron 12 modelos, primero se seleccionó el mejor modelo, de acuerdo al criterio AIC, para cada uno de los *lags* propuestos. Además para esta prueba se incluyeron 4 modelos más para 8 neuronas de entrada con 17, 15, 8 y 1 neuronas en capa oculta, el mejor modelo para este valor de *lags*, de acuerdo a AIC, fue el 8-16-1.

En este filtro se comparó el coeficiente de determinación para observar su factor de ajuste tanto en la etapa de entrenamiento como en la de prueba, los resultados obtenidos para cada modelo se observan en la tabla 4.1.

# Modelo	Descripción	Entrenamiento	Prueba
1	1-3-1	0.497	0.414
2	2-14-1	0.571	0.364
3	3-17-1	0.625	0.232
4	4-16-1	0.674	0.245
5	5-17-1	0.799	0.128
6	6-15-1	0.823	0.244
7	7-17-1	0.881	0.116
8	8-17-1	0.928	0.060
9	8-1-1	0.511	0.489
10	8-8-1	0.780	0.049
11	8-15-1	0.897	0.284
12	8-16-1	0.921	0.135

Tabla 4.1: Coeficientes de determinación RNA

La selección del mejor modelo RNA fue complicada debido a que en casi todos los modelos se ve una pérdida de generalización por parte de la RNA, es decir, se obtiene un valor de R^2 muy superior en el entrenamiento contra la prueba. En casos como el modelo 8 incluso alcanzó un coeficiente de determinación de 0.921 en el entrenamiento pero para la prueba se redujo hasta 0.060. Por esta razón se eligió el modelo 9 (8 neuronas de entrada, una en capa oculta y una de salida) como "mejor" modelo ya que su valor entre entrenamiento y prueba únicamente se redujo 0.022 pero el valor de ajuste del modelo en ambos casos es inferior a 0.75, que es lo mínimo que se podría aceptar.

4.2.2. Modelos Wavelet+RNA

Para este caso se realizaron 7 modelos, se utilizaron 2 familias de *wavelets*: *Daubechies* y *Symlets*. El máximo número de desvanecimientos que permite el paquete son 10, las pruebas se hicieron con 9 y 10 desvanecimientos. Se utilizó el máximo posible ya que como se menciona en el capítulo 2, entre mayor número de desvanecimientos

más suavizada es la *wavelet* y la descomposición se centra más en las discontinuidades.

Solo se realizó una prueba con *Daubechies* con el fin de ejemplificar la prioridad de selección que para estos datos tuvo la familia *Symlets*. En las gráficas 4.5 y 4.6 se tiene la serie original vs la serie con reducción del ruido. Se observa un comportamiento similar en ambas pero en el caso *Daubechies* la serie reduce su nivel, también en el segmento anterior a octubre 2013 se tiene un decaimiento que no va acorde con el comportamiento de la serie y además las discontinuidades en las frecuencias más bajas se pierden casi por completo.

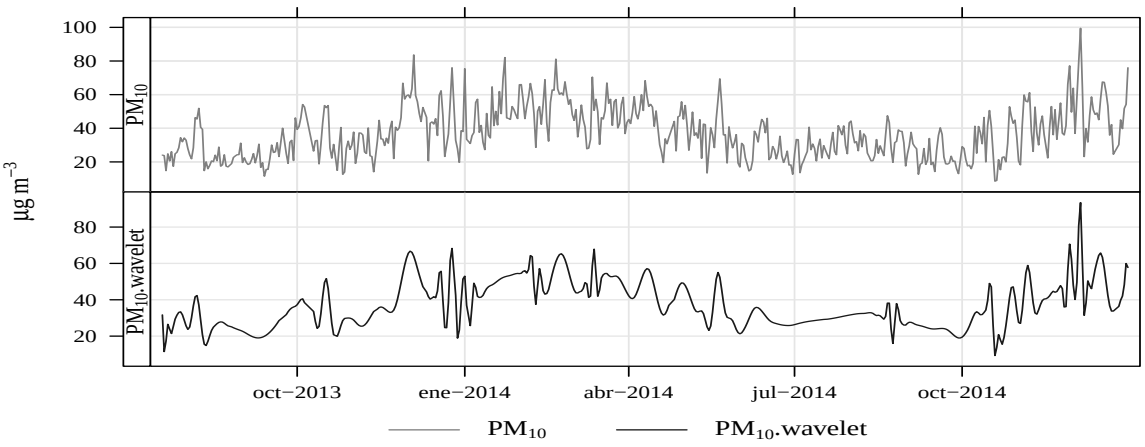

Figura 4.5: Serie con reducción del ruido utilizando Symlet 10

Para ambas se realizó el mismo procedimiento para reducción del ruido, esto se detallará posteriormente. Pero de acuerdo al comportamiento de las series resultantes, se decidió dar prioridad para la selección del modelo a la familia *Symlet*. En la figura 4.7 se observa la *wavelet* madre y la función escala (o *wavelet* padre) de un "miembro" de la familia *Symlet*, la *Symlet* 10.

Dado que el tamaño de los datos $n = 2^m$ fue igual a 512 entonces $m = 9$. Este es el número de niveles de descomposición que se obtuvieron al aplicar la transformada, por lo tanto, en el nivel de la escala más "fina" se obtuvieron $n/2 = 2^{n-1}$ datos. Esto independientemente de la *wavelet* utilizada, en la gráfica 4.8 se puede observar la

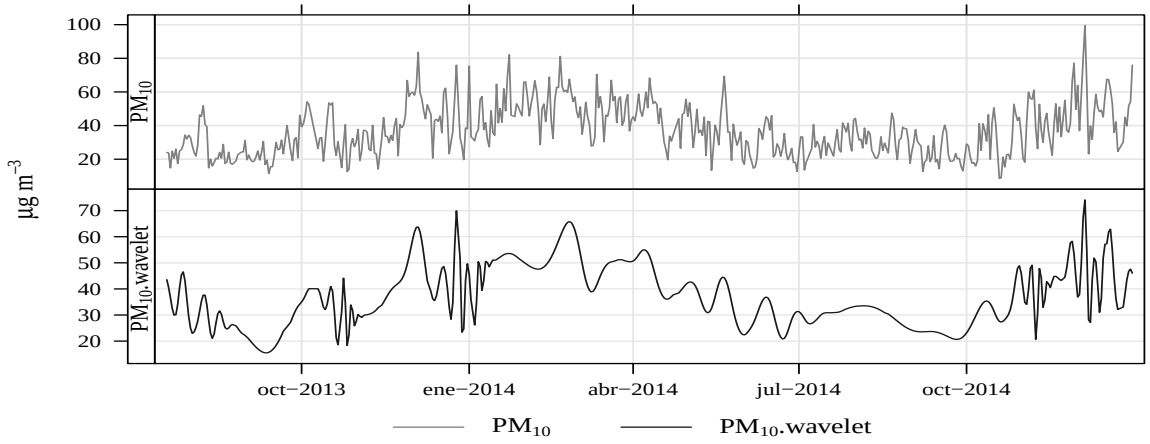

Figura 4.6: Serie con reducción del ruido utilizando Daubechies 10

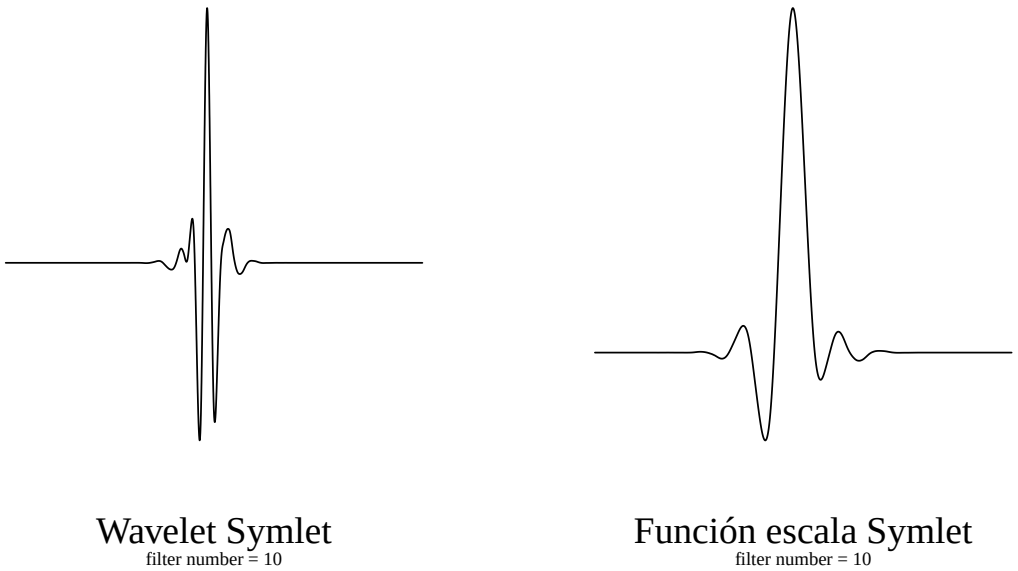

Figura 4.7: *Symlet* 10

descomposición con la *Symlet* 10.

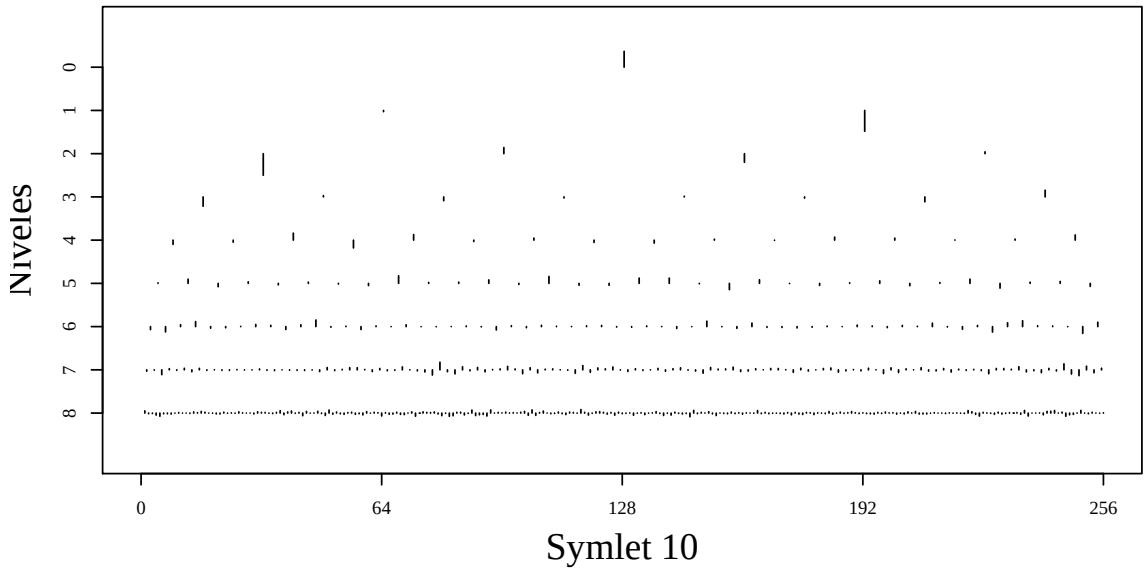

Figura 4.8: Descomposición *Wavelet* al aplicar *Symlet* 10

Para reducir el ruido de la serie se aplicó la regla de reducción no lineal y se utilizó el tipo de regla *Hard*, el paquete únicamente tiene dos opciones: *Hard* y *Soft*, que fueron las que se expusieron en el capítulo 2; se decidió elegir el tipo *Hard* ya que por la naturaleza de los datos se tiene una varianza grande y al ocupar la otra opción se eliminaría esta propiedad que es la base de la estructura de los datos. Para calcular λ se utilizo la función de umbral universal propuesto por Donoho y Johnstone (1994) cuya función es:

$$\lambda = \sigma\sqrt{2\log n}.$$

En este caso, σ no significa desviación estándar de los datos originales sino de los niveles de ruido, por esta razón los mismos autores proponen utilizar la desviación media absoluta como estimador en el dominio de las *wavelets*, también proponen que se aplique a los coeficientes de la escala más "fina" ya que por ser los valores más pequeños son los que se componen mayormente por ruido. Estos parámetros se ocuparon en todos los modelos, únicamente se cambio el tipo de *wavelet*.

Para seleccionar el modelo, se utilizó como primer filtro lo obtenido de la función `selectNNET` que ofrece el paquete. Por medio de está función se probaron redes de 1 a 8 neuronas en la capa de entrada y de 1 a 8 neuronas en la capa oculta. El modelo con mejor AIC fue el que tuvo 8 neuronas de entrada y 8 neuronas en capa oculta, esto ocurrió tanto con la serie transformada por la *wavelet Symlet* 9 como con la *Symlet* 10.

# Modelo	Descripción	Entrenamiento	Prueba
1	Symlet10+8-8-1	0.996	0.960
2	Symlet10+8-6-1	0.995	0.934
3	Symlet10+8-10-1	0.997	0.919
4	Symlet9+8-8-1	0.993	0.896
5	Symlet9+8-6-1	0.981	0.753
6	Symlet9+8-10-1	0.994	0.780
7	Daubechies10+8-8-1	0.998	0.979

Tabla 4.2: Coeficientes de determinación Wavelet+RNA

Se modificaron estos dos modelos, cambiando en número de neuronas en capa oculta por 6 y 10, con esto se obtuvieron 6 modelos para comparar. El número de *lags* no se modificó porque se consideró suficiente información para tener una tendencia y se obtuvieron resultados adecuados en la etapa de entrenamiento.

Los resultados de los coeficiente de determinación para estos modelos para la etapa de entrenamiento y prueba se muestran en la tabla 4.2. Debido a que el modelo 7 se descartó por las inconsistencias que el tipo de *wavelet* mostró, se seleccionó el modelo 1 porque el coeficiente de determinación en la etapa de prueba tuvo una diferencia de 0.036 respecto a la etapa de entrenamiento y este valor fue el menor en los modelos probados, además de mostrar un buen ajuste.

Capítulo 5

Resultados

Modelo RNA. Para el modelo RNA se eligió una red autorregresiva con 8 neuronas en la capa de entrada, 1 neurona en la capa oculta y 1 neurona en la capa de salida. Su coeficiente de determinación en la etapa de entrenamiento fue de 0.511 y en la etapa de prueba de 0.489. Es un modelo que no se ajusta a los datos y aunque se compararon diversos modelos con distinta estructura, no se encontró alguno que bajo el procesamiento que ofrece el paquete `tsDyn` tuviera un ajuste aceptable tanto en la etapa de entrenamiento como en la de prueba.

En el gráfico de dispersión 5.1 se muestran los resultados obtenidos en este modelo contra su conjunto de entrenamiento y en la gráfica 5.2 se tiene la comparación de las series de tiempo.

Se observa en la gráfica 5.2 que el principal problema del modelo es en las frecuencias más altas. Son tantas discontinuidades que presenta la serie que el modelo no es capaz de cubrirlas, al utilizar más neuronas en capa oculta, para este caso, se logró mejorar el nivel de ajuste en la etapa de entrenamiento pero de esta manera perdía su capacidad de generalización y conforme aumentaba el número de neuronas en capa oculta aumentaba la diferencia entre el ajuste obtenido en el entrenamiento y en la prueba.

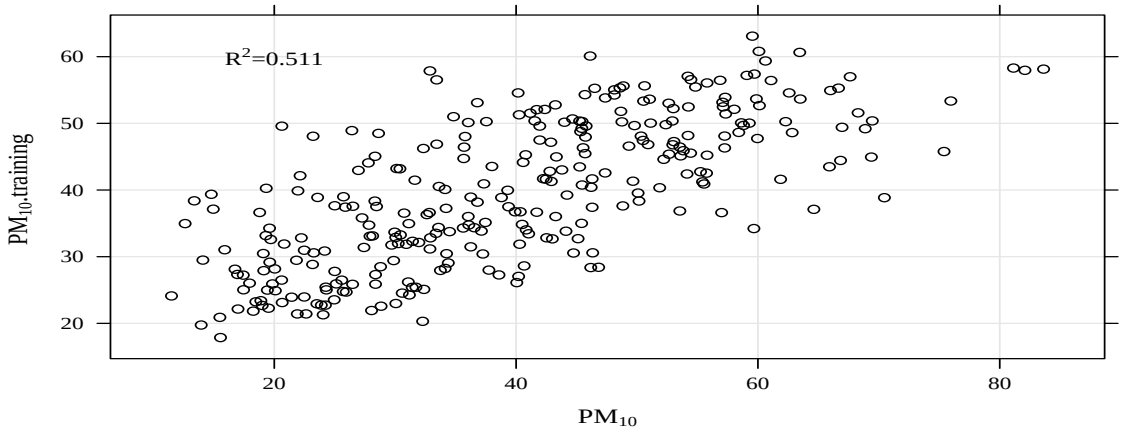

Figura 5.1: Gráfico de dispersión modelo RNA

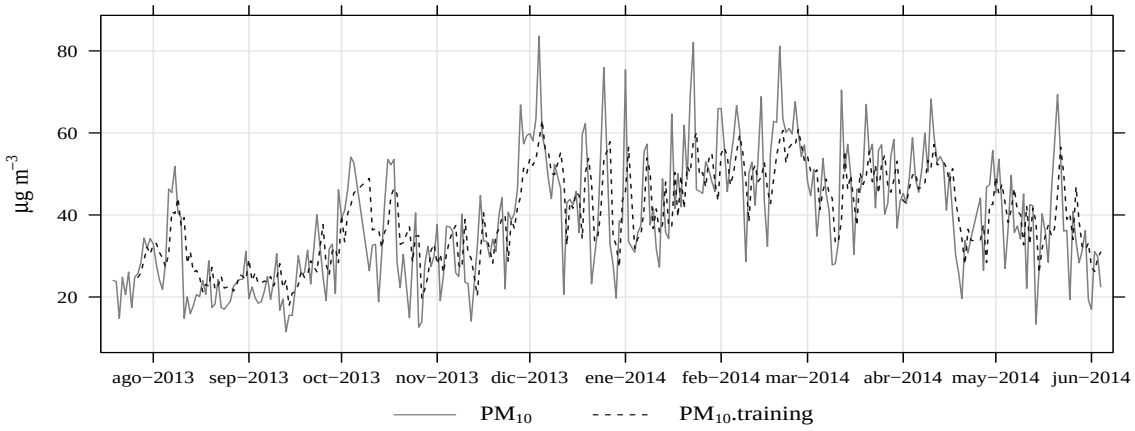

Figura 5.2: Serie de tiempo del pronóstico obtenido por modelo RNA y Serie de tiempo del conjunto de entrenamiento

En la gráfica 5.3 se muestran los resultados obtenidos en la etapa de prueba. En este conjunto los niveles de frecuencia son más bajos que en el conjunto de entrenamiento y la serie de pronóstico muestra un desface y además se mantiene la baja capacidad de alcanzar los niveles de las altas frecuencias.

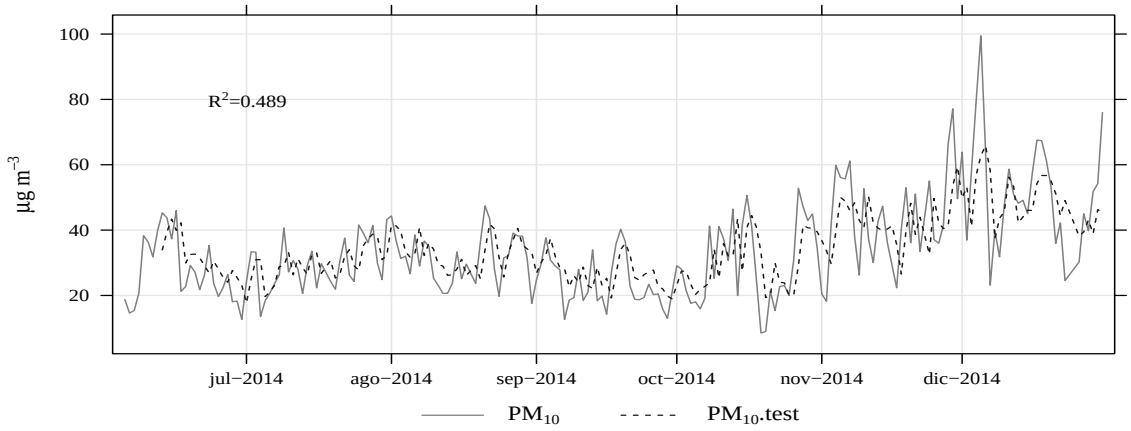

Figura 5.3: Serie de tiempo del pronóstico obtenido por modelo RNA y Serie de tiempo del conjunto de prueba

Modelo Wavelet+RNA. Para el caso del modelo con reducción del ruido por medio de la transformada de *Wavelet*, se utilizó la *wavelet Symlet* 10 y la regla de reducción no lineal *Hard*. En la gráfica 5.4 se muestra el resultado obtenido al aplicar este método para reducción del ruido, se observa en línea punteada el resultado y en línea solida la serie original. Se puede observar que se conserva la estructura general de la serie, por la naturaleza de la regla aplicada se conservan de mejor forma las frecuencia altas que las frecuencias bajas.

La serie resultante, al tener menos discontinuidades, facilita que el modelo de pronóstico pueda captar la estructura general de la serie y se pueda llegar a un pronóstico más acertado. Lógicamente al modificar la serie original, no se tendrá un pronostico certero de la serie original pero si de su estructura general y por lo tanto de su tendencia.

La red autorregresiva que se utilizó posteriormente fue una red con 8 neuronas en la capa de entrada, 8 neuronas en la capa oculta y 1 neurona en la capa de salida. El coeficiente de determinación en la etapa de entrenamiento fue de 0.996 y en la etapa de prueba de 0.960, se consiguió un buen nivel de ajuste y además en la etapa de prueba a pesar de que los datos poseían distinta estructura, debido a que se tienen más lapsos

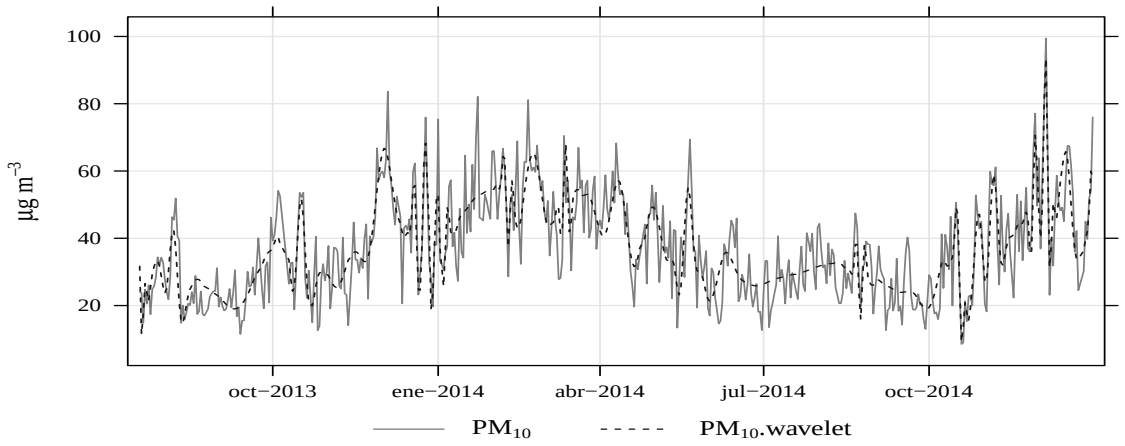

Figura 5.4: Comparación serie original vs Serie con Reducción del Ruido

de frecuencia bajas, se tuvo un resultado muy similar al de entrenamiento.

En el gráfico 5.5 se tiene la dispersión del pronóstico obtenido contra su conjunto de entrenamiento. En la gráfica 5.6 se pueden ver ambas series de tiempo, en línea punteada se observa el pronóstico. No se observan diferencias importantes más que en algunos puntos donde se tiene un cambio hacia una frecuencia baja.

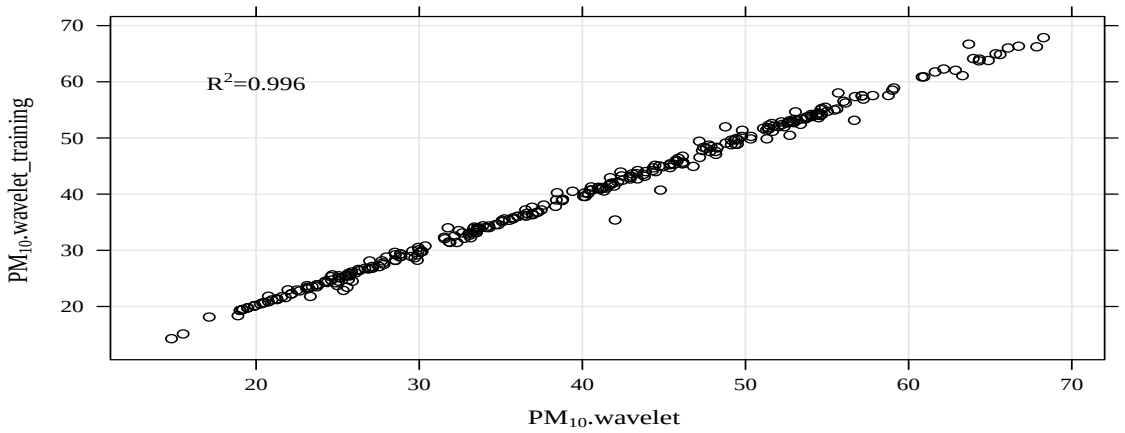

Figura 5.5: Gráfico de dispersión modelo Wavelet+RNA

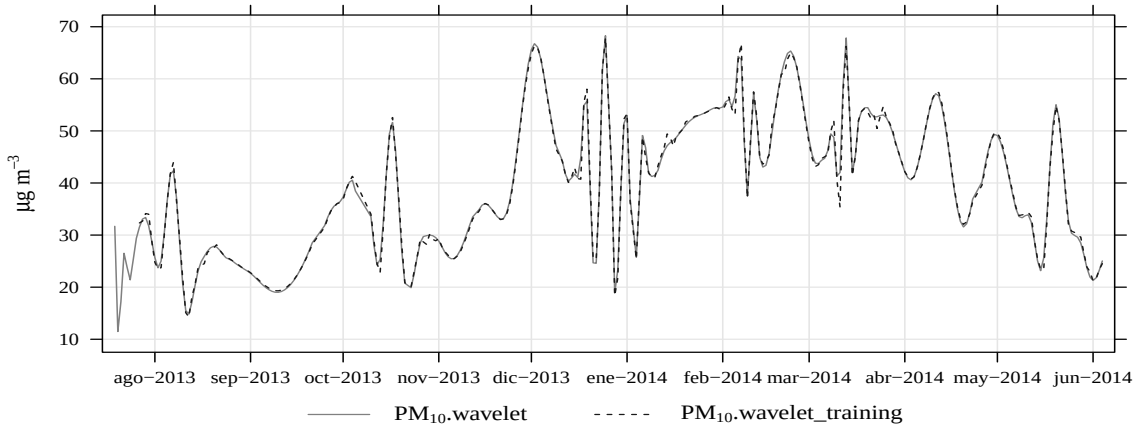

Figura 5.6: Serie de tiempo del pronóstico obtenido por modelo Wavelet+RNA y Serie de tiempo del conjunto de entrenamiento

Para la etapa de prueba en la gráfica 5.7 se encuentran los resultados obtenidos. Se observa que la estructura es muy diferente a la mostrada en la etapa anterior donde se tienen más discontinuidades y a pesar de esto la red siguió pronosticando de manera eficiente los niveles, tuvo algunos problemas en el último mes donde se tuvieron las frecuencias más altas.

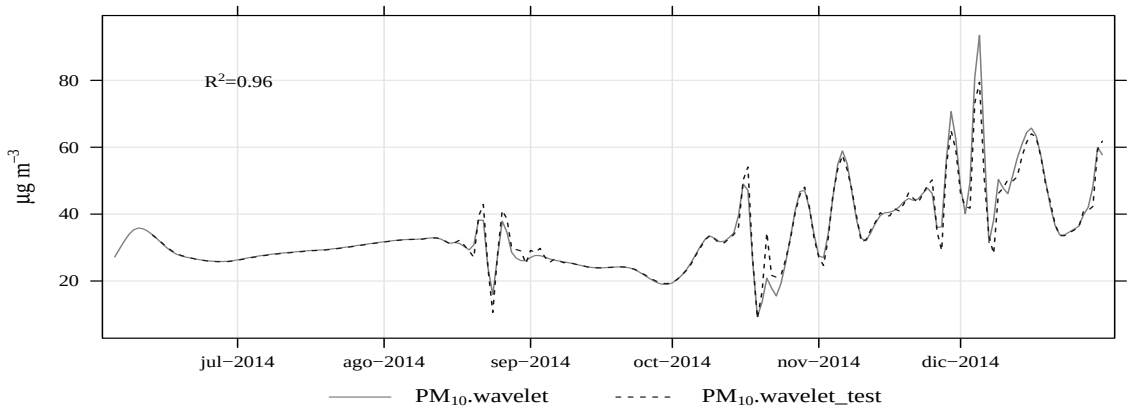

Figura 5.7: Serie de tiempo del pronóstico obtenido por modelo Wavelet+RNA y Serie de tiempo del conjunto de prueba

Conclusiones

Con el modelo de pronóstico con una red neuronal artificial donde se utilizó la transformada de *Wavelet* como método de reducción del ruido, en una serie histórica de los niveles de contaminación por partículas menores a 10 micrómetros (PM_{10}), se obtuvo un mejor ajuste que al aplicar únicamente la red neuronal sobre los datos originales. Con este modelo no se está realizando un pronóstico sobre los datos originales pero si se está realizando un pronostico sobre la estructura general de los datos y con esto, si bien no se puede dar un valor cercano al nivel que podría llegar el contaminante al día siguiente, si se puede tener una tendencia que los datos podrían tener.

Con este modelo solo se están considerando los valores anteriores que la misma serie tuvo para establecer el pronostico, por lo tanto no se incluyen pronósticos ni históricos adicionales para las variables exógenas que se pudieran considerar en otros modelos, ya sean variables meteorológicas o de otros contaminantes. Esto tiene una ventaja pues el modelo no esta limitado por la disponibilidad de valores externos y además el resultado fue muy satisfactorio, también tiene una limitante debido al funcionamiento de la transformada ya que solo se puede trabajar con datos donde el tamaño sea igual a una potencia de 2.

En futuros estudios se buscaría considerar otras reglas de reducción no lineal en donde se pueda conservar mayor estructura de los datos originales y también incluir variables exógenas, si es que estas ayudan a mejorar el nivel de ajuste con estructuras más complejas.

Fuera del resultado que se obtuvo, también con este trabajo se tiene una base

tanto teórica como práctica para desarrollar modelos no convencionales para pronósticos con otro tipo de variables, como las económicas y financieras, que en muchos casos tienen una estructura compleja. Este es un reto que he encontrado personalmente en el mercado laboral, ya que muchas veces los modelos convencionales no se adaptan al comportamiento de los datos que se ven en el día a día y espero que con esto, tanto estudiantes como profesionistas de mi carrera y de carreras afines, puedan utilizar nuevas técnicas para su labor. También a partir de esto se puede ver que la teoría no está peleada con la práctica, es muy importante seguir investigando sobre nuevas técnicas y modelos que además de tener una base teórica muy importante son totalmente aplicables, a veces esto se olvida o carece de importancia cuando se está en el mercado laboral pero realmente es muy valorado.

Por otra parte este trabajo se hizo completamente con datos públicos y *software* libre, es decir, está al alcance de todos. Además aquí se muestra el desarrollo completo de un modelo, esto es muy útil debido a que muchas veces tanto en artículos como en libros se presenta una descripción general y los resultados, pocas veces se puede obtener una idea clara de los pasos que se siguieron para conseguir el resultado, esta es una aportación que puede ser aprovechada por alguna persona interesada en el tema.

Anexos

Anexo A

Código Modelo RNA

```
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
```

```
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n <- nnetTs(pm10.n, m=8, size=1,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,8)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:197){
```

```
  test.n[i]<-(predict(mod.nnet.n,newdata=pm10.n.test[i:(i+7)],n.ahead=1)*
                sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,8)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
summary(test.n.lm)$r.squared

#Crear Gráficas
library(openair)

colnames(wt_pm10)[3]<-c("pm10.training")
colnames(wt_pm10.test)[3]<-c("pm10.test")

round(summary(train.n.lm)$r.squared ,3)

library(latticeExtra)
pm10.train<-scatterPlot(wt_pm10[-c(1:8),], x = "pm10", y = "pm10.training",
                        cols="black")
pm10.train
pm10.train<-trellis.last.object() +
  layer(ltext(x = 20, y = 60,
              labels =
                expression("R"^2* "=0.511"),
              cex=0.9), rows = 1)
pm10.train
pdf(file="pm10_training.pdf",height=4,width=7,
    family="Times",pointsize=5,colormodel="grey")
print(pm10.train)
dev.off()
dev.off()
pm10.train.serie<-timePlot(wt_pm10,pollutant=c("pm10","pm10.training"),
                           date.format= "%b-%Y",cols="greyscale",
                           group = TRUE,
                           ylab = expression(mu * "g m" ^-3))
```

```
pdf(file="ts_pm10_training.pdf",height=4,width=8,
    family="Times",pointsize=5,colormodel="grey")
print(pm10.train.serie)
dev.off()

round(summary(test.n.lm)$r.squared ,3)

pm10.test.serie<-timePlot(wt_pm10.test,pollutant=c("pm10","pm10.test"),date.format
                          = "%b-%Y",cols="greyscale",group = TRUE,
                          ylab = expression(mu * "g m" ^-3))
pm10.test.serie
pm10.test.serie<-trellis.last.object() +
  layer(ltext(x = as.POSIXct("2014-07-01"), y = 80,
              labels =
                expression("R"^2* "=0.489"),
              cex=0.8), rows = 1)
pm10.test.serie

pdf(file="ts_pm10_testing.pdf",height=4,width=8,
    family="Times",pointsize=5,colormodel="grey")
print(pm10.test.serie)
dev.off()

#Guardar Modelo
save.image("rna.RData")
```

Anexo B

Código Modelo Wavelet+RNA

```
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
library(wavethresh)

#Carga de todas las tablas
load("tablas_procesadas.RData")

rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]
```

```
#Eliminar temporales
rm(list=c("temp","temp2"))

#Gráfica Wavelet Symlet 10 y Wavelet Padre
pdf(file="wavelet.pdf",height=4,width=7,
    family="Times",pointsize=8)
oldpar<-par(mfrow=c(1,2))#To plot one fig above the other
draw.default(filter.number=10, family="DaubLeAsymm",
               enhance=FALSE, main="",
             xlab="Wavelet Symlet",
             sub="filter number = 10",
             ylab="",axes=F,cex.lab=2)
draw.default(filter.number=10, family="DaubLeAsymm",
              enhance=FALSE, scaling.function=TRUE, main="",
             xlab="Función escala Symlet",
             sub="filter number = 10",
             ylab="",axes=F,cex.lab=2)
par(oldpar)
dev.off()

#Serie de datos para wavelet
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")

#Transformada Wavelet Discreta Symlet 10
pm_wd<-wd(pm,filter.number=10, family="DaubLeAsymm")

#Gráfica descomposición wavelet
pdf(file="wavelet_desc.pdf",height=4,width=7,
    family="Times",pointsize=8)
plot(pm_wd,main="",ylab="Niveles",
     xlab="Symlet 10",sub="",cex.lab=2,cex.main=0.1)
dev.off()

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)
```

```
#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))

#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)

#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"

#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]
wt_pm10<-wt_pm10.com[1:307,]

#Escalar la serie
pm10.w<-scale(wt_pm10.w)

#Obtener los datos de prueba
wt_pm10.w.test<-wt_pm10.com[308:512,3]
wt_pm10.test<-wt_pm10.com[308:512,]

#Cargar paquete para RNA
library(tsDyn)

#Entrenar la RED
mod.nnet.w <- nnetTs(pm10.w, m=8, size=8,step=1)

#obtener los valores ajustados y quitar el escalamiento
fit.w<-mod.nnet.w$fitted.values
fit.w<-(fit.w*sd(wt_pm10.w))+mean(wt_pm10.w)
colnames(fit.w)[1]<-"fit.w"
fit.w<-rbind(data.frame(fit.w=rep(NA,8)),fit.w)

#Agregar a la base general
wt_pm10<-cbind(wt_pm10,fit.w)

#obtener R cuadrada
```

```
train.w.lm<-lm(pm10.wavelet ~ fit.w, data=wt_pm10)
summary(train.w.lm)$r.squared

#Guardar datos de escalamiento de la serie de entrenamiento
sd.w<-sd(wt_pm10.w)
mean.w<-mean(wt_pm10.w)

#Escalar la serie de prueba
pm10.w.test<-(wt_pm10.w.test-mean.w)/sd.w

#Aplicar el modelo a la serie de prueba
test.w<-0
for(i in 1:197){
  test.w[i]<-(predict(mod.nnet.w,newdata=pm10.w.test[i:(i+7)],n.ahead=1)*
                sd.w)+mean.w
}
#Agregar los resultados a la serie de prueba
test.w<-data.frame(test.w=test.w)
test.w<-rbind(data.frame(test.w=rep(NA,8)),test.w)
wt_pm10.test<-cbind(wt_pm10.test,test.w)

#Obtener R cuadrada
test.w.lm<-lm(pm10.wavelet ~ test.w, data=wt_pm10.test)
summary(test.w.lm)$r.squared

#Gráficar los resultados wavelet y wavelet+RNA
library(openair)
pdf(file="completedata.pdf",height=4,width=7,
    family="Times",pointsize=5,colormodel="grey")
timePlot(wt_pm10.com,pollutant=c("pm10"),date.format
         = "%b-%Y",cols="greyscale",
         ylab = expression(mu * "g m" ^-3))
dev.off()
pdf(file="completedata_wavelet.pdf",height=4,width=7,
    family="Times",pointsize=5,colormodel="grey")
timePlot(wt_pm10.com,pollutant=c("pm10","pm10.wavelet"),date.format
         = "%b-%Y",cols="greyscale",
         ylab = expression(mu * "g m" ^-3))
dev.off()
pdf(file="completedata_waveletc.pdf",height=4,width=7,
```

```
    family="Times",pointsize=5,colormodel="grey")
timePlot(wt_pm10.com,pollutant=c("pm10","pm10.wavelet"),date.format
         = "%b-%Y",cols="greyscale",group=TRUE,
         ylab = expression(mu * "g m" ^-3))
dev.off()
rm(oldpar)

colnames(wt_pm10)[4]<-c("pm10.wavelet_training")
colnames(wt_pm10.test)[4]<-c("pm10.wavelet_test")

round(summary(train.w.lm)$r.squared ,3)

library(latticeExtra)
wavelet.train<-scatterPlot(wt_pm10[-c(1:8),], x = "pm10.wavelet",
y = "pm10.wavelet_training",
                           cols="black")
wavelet.train
wavelet.train<-trellis.last.object() +
  layer(ltext(x = 20, y = 60,
              labels =
                expression("R"^2* "=0.996"),
              cex=0.9), rows = 1)
wavelet.train
pdf(file="wavelet_training.pdf",height=4,width=7,
    family="Times",pointsize=5,colormodel="grey")
print(wavelet.train)
dev.off()
dev.off()


wavelet.train.serie<-timePlot(wt_pm10,pollutant=c("pm10.wavelet",
"pm10.wavelet_training"),
date.format= "%b-%Y",cols="greyscale",group = TRUE,
ylab = expression(mu * "g m" ^-3))
pdf(file="ts_wavelet_training.pdf",height=4,width=8,
    family="Times",pointsize=5,colormodel="grey")
print(wavelet.train.serie)
dev.off()

round(summary(test.w.lm)$r.squared ,3)
```

```
wavelet.test.serie<-timePlot(wt_pm10.test,pollutant=c("pm10.wavelet",
"pm10.wavelet_test"),
date.format= "%b-%Y",cols="greyscale",group = TRUE,
ylab = expression(mu * "g m" ^-3))
wavelet.test.serie
wavelet.test.serie<-trellis.last.object() +
  layer(ltext(x = as.POSIXct("2014-07-01"), y = 80,
              labels =
                expression("R"^2* "=0.96"),
              cex=0.8), rows = 1)
wavelet.test.serie
pdf(file="ts_wavelet_testing.pdf",height=4,width=8,
    family="Times",pointsize=5,colormodel="grey")
print(wavelet.test.serie)
dev.off()

#Guardar el modelo
save.image("wavelet+rna.RData")
```

Anexo C

Anexos Digitales

Gráficas resumen

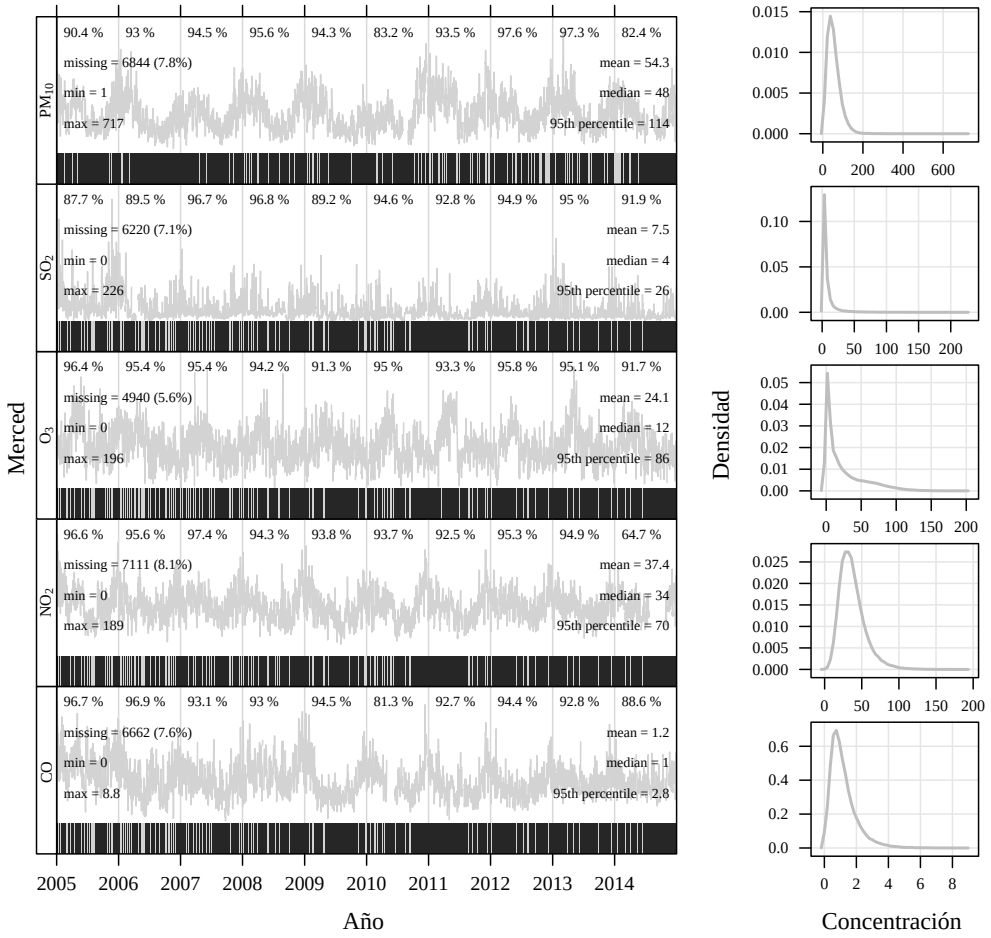

Figura C.1: MERCED - Gráfico de resumen (concentraciones horarias)

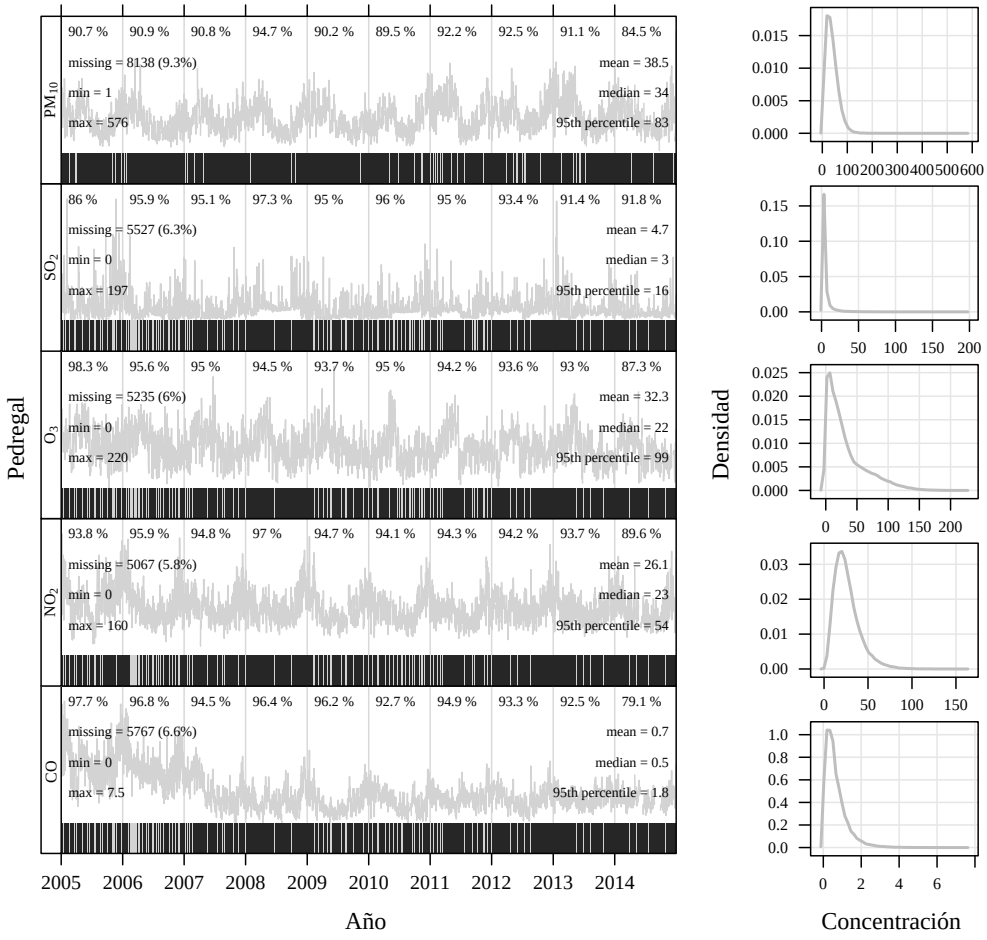

Figura C.2: PEDREGAL - Gráfico de resumen (concentraciones horarias)

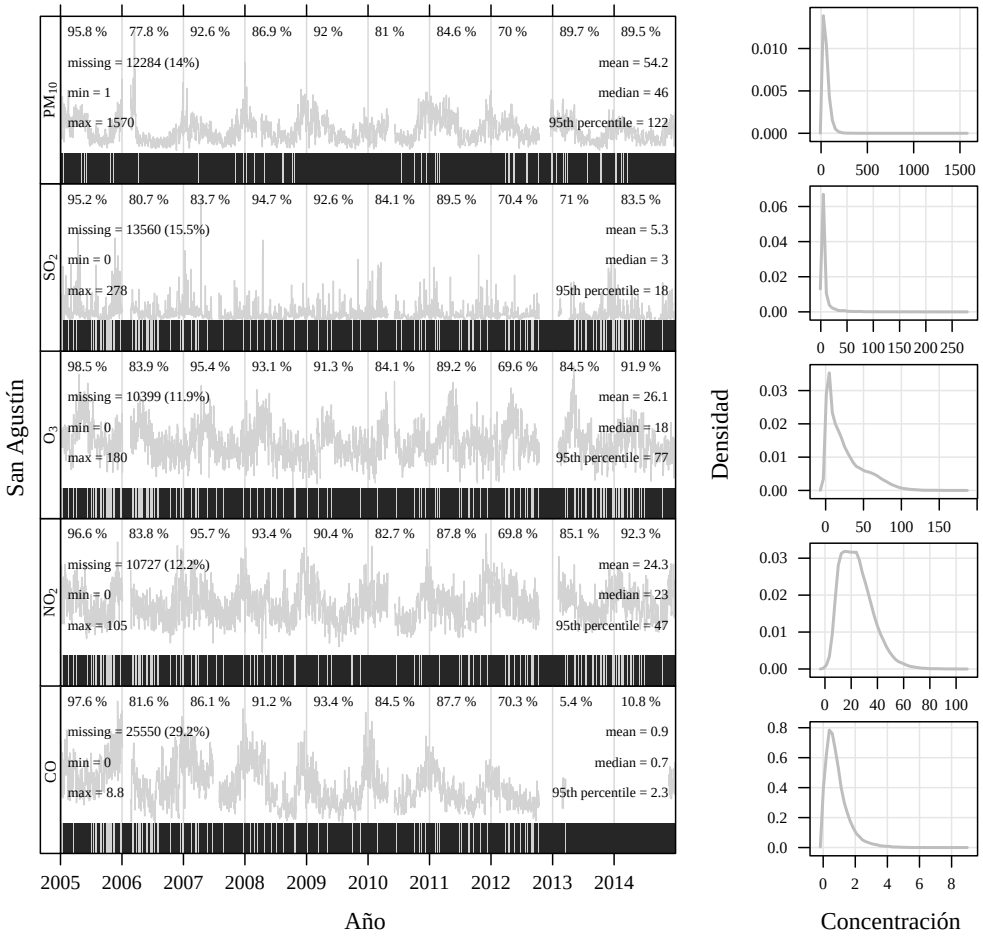

Figura C.3: SAN AGUSTÍN - Gráfico de resumen (concentraciones horarias)

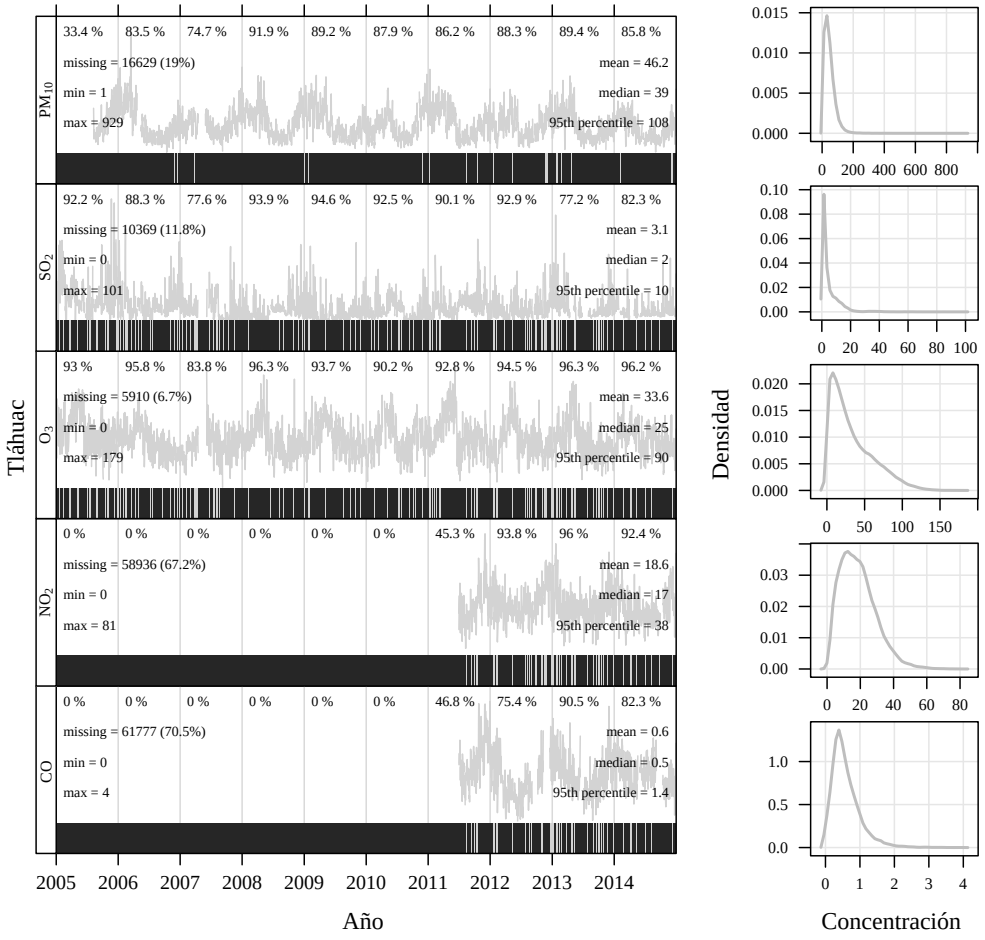

Figura C.4: TLÁHUAC - Gráfico de resumen (concentraciones horarias)

Gráficas por referencia

Merced - MER

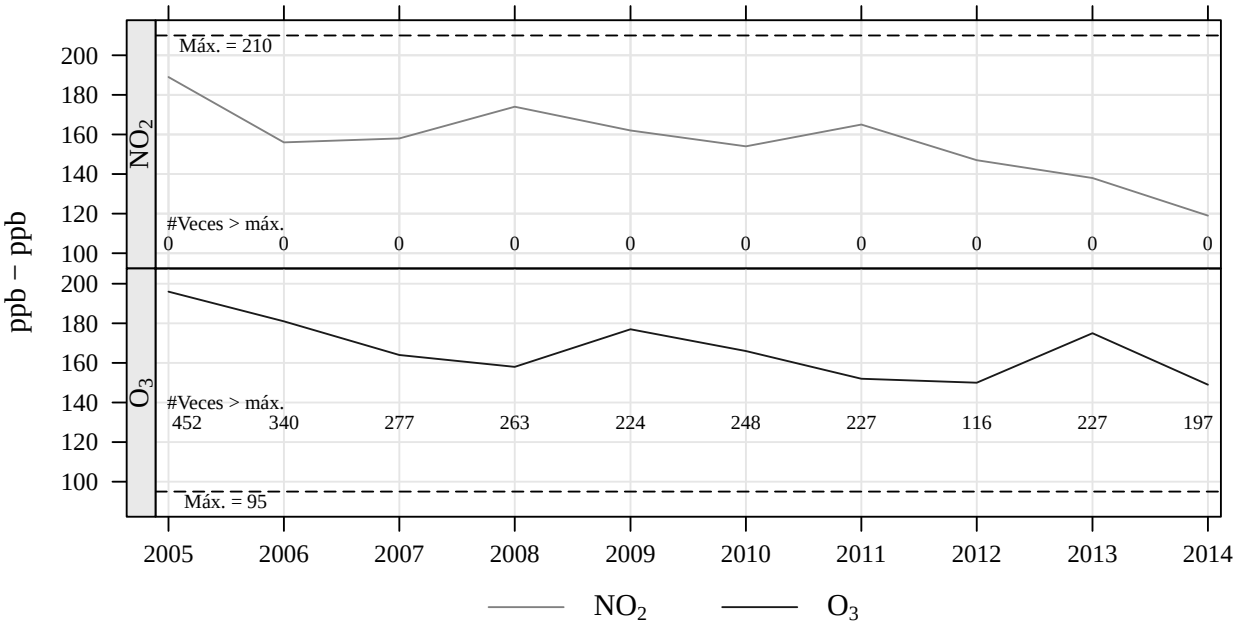

Figura C.5: MER - Referencia Horaria

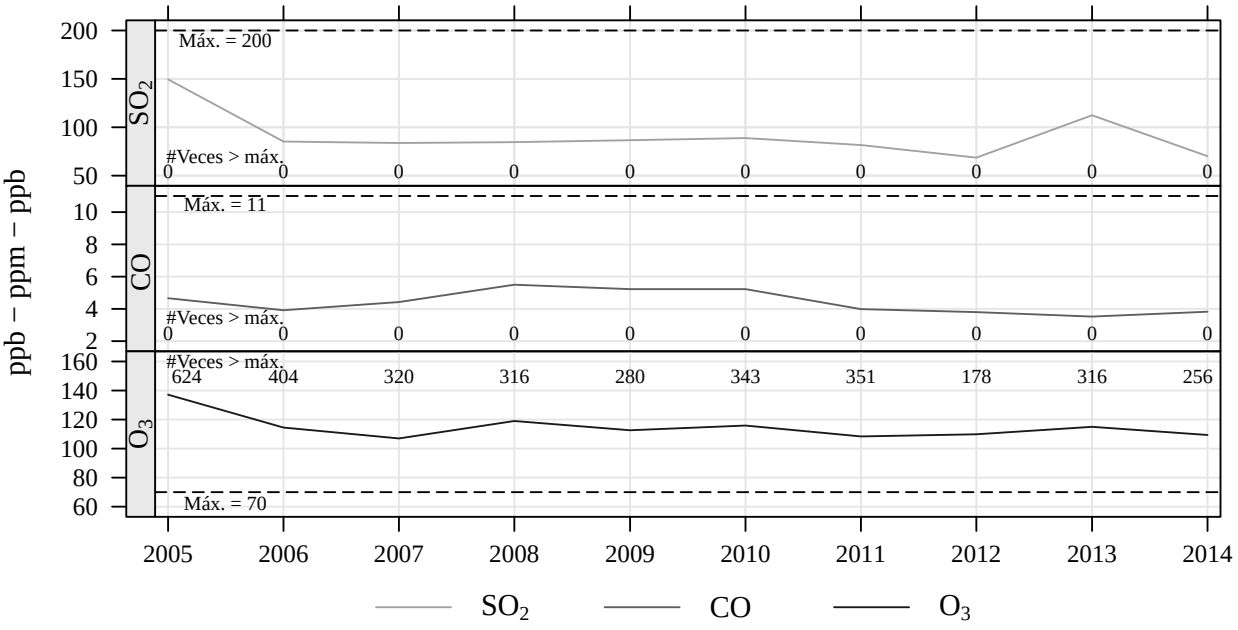

Figura C.6: MER - Referencia Promedio Móvil 8 Horas

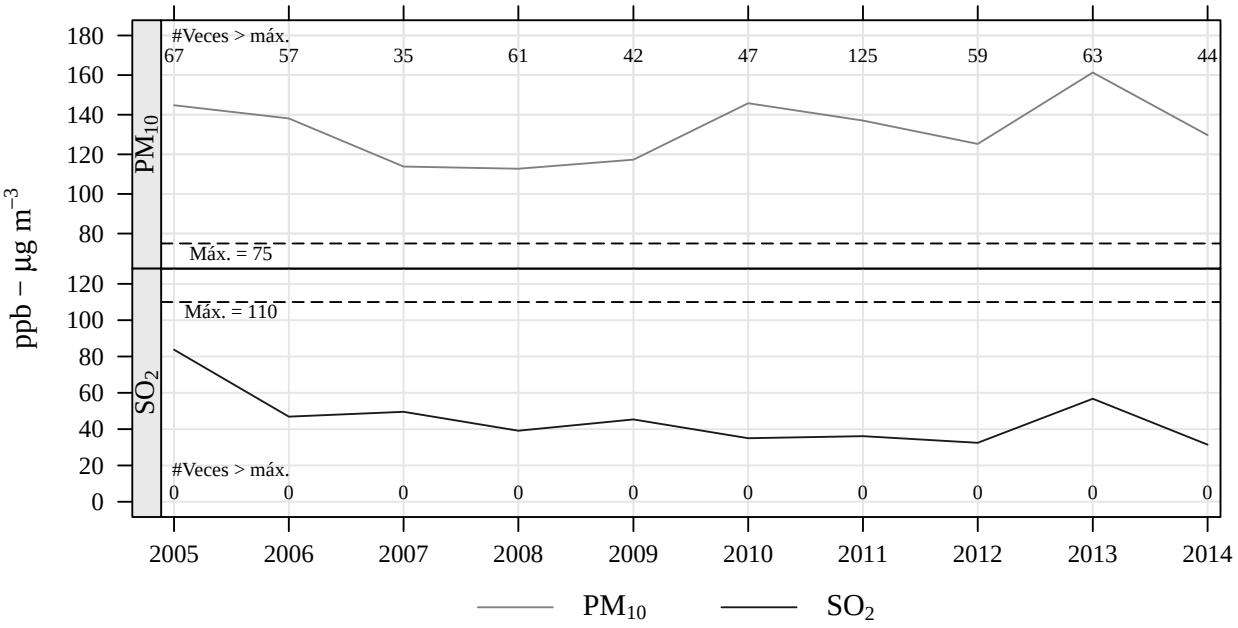

Figura C.7: MER - Referencia Promedio Diario

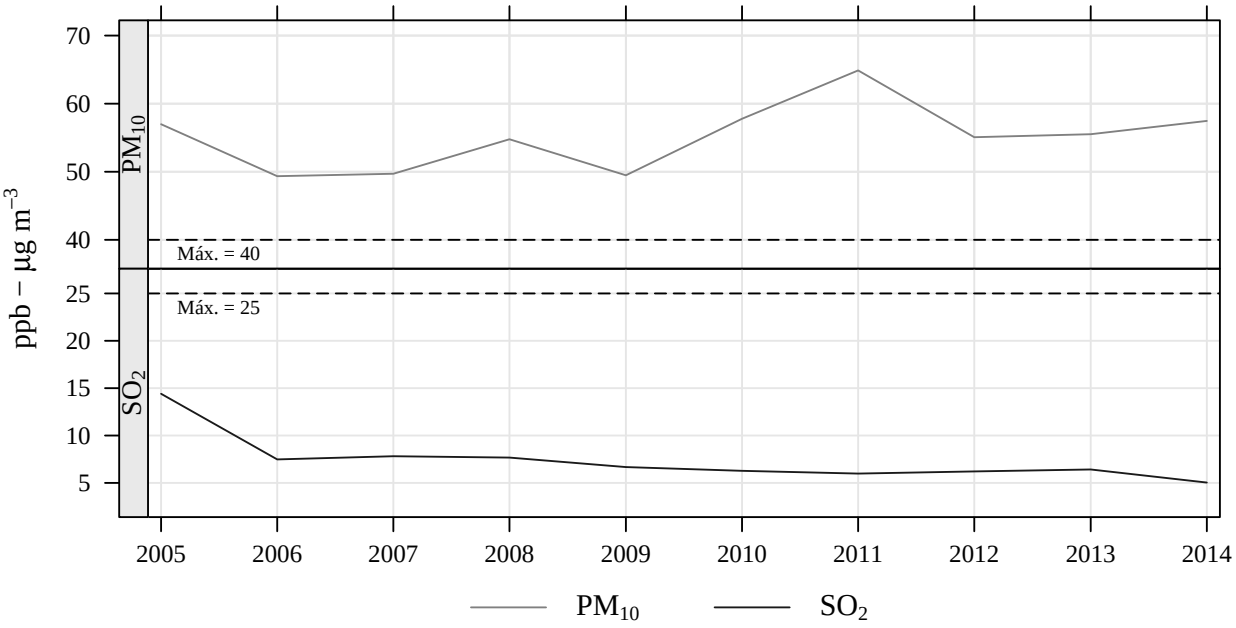

Figura C.8: MER - Referencia Promedio Anual

Pedregal - PED

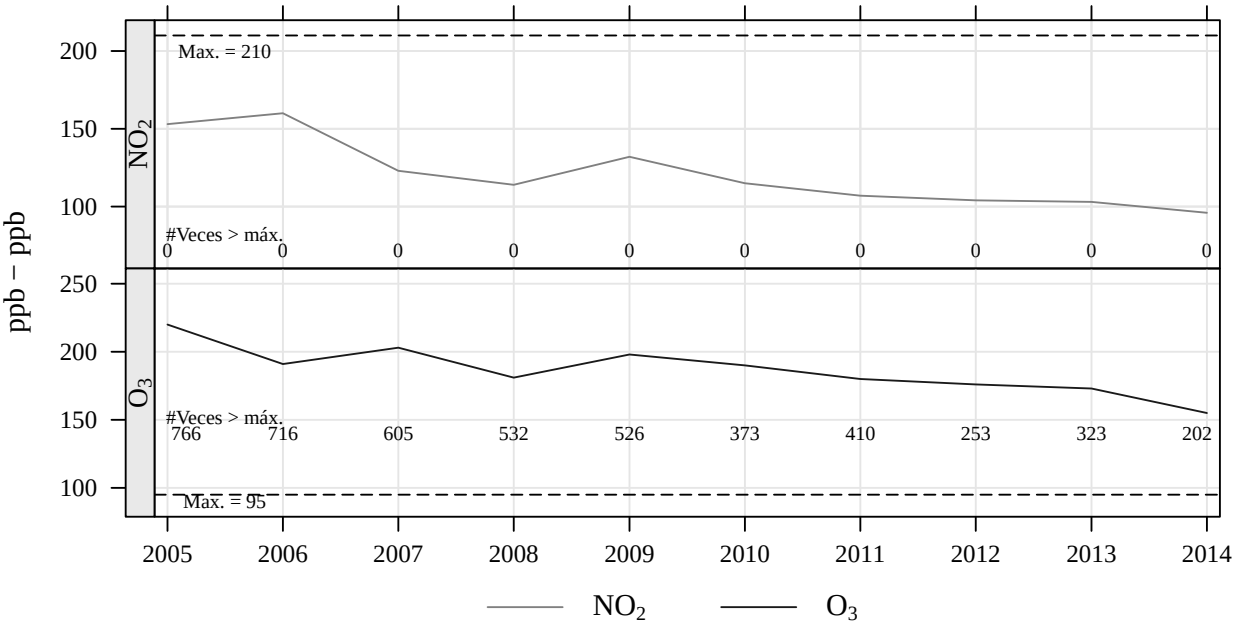

Figura C.9: PED - Referencia Horaria

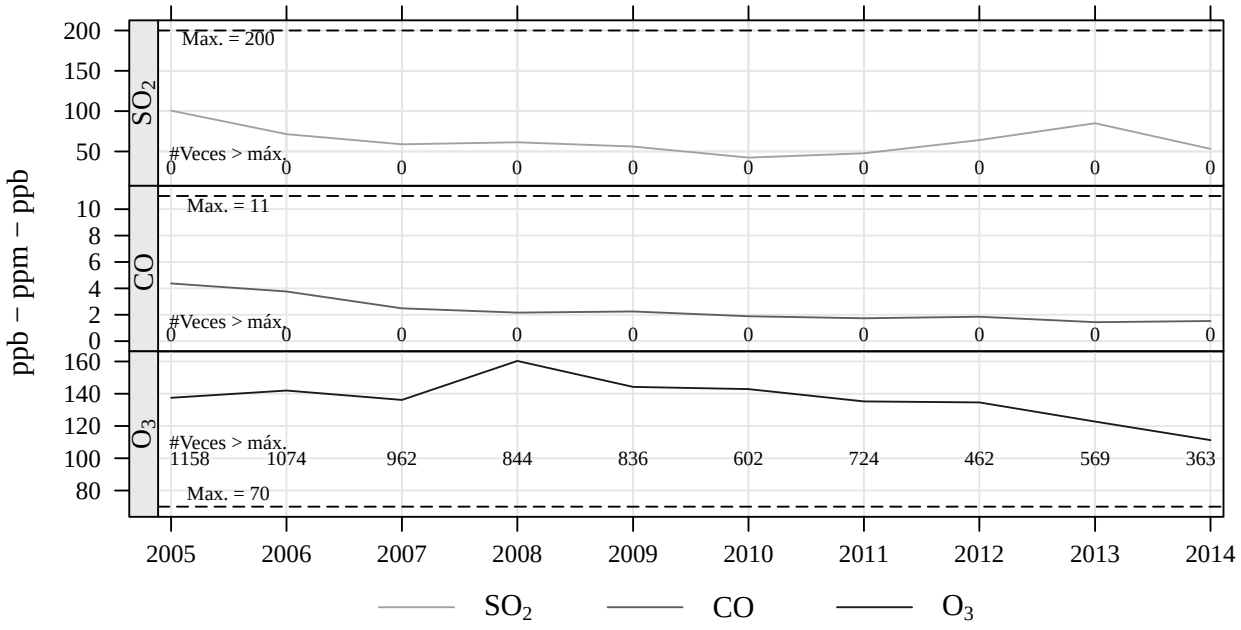

Figura C.10: PED - Referencia Promedio Móvil 8 Horas

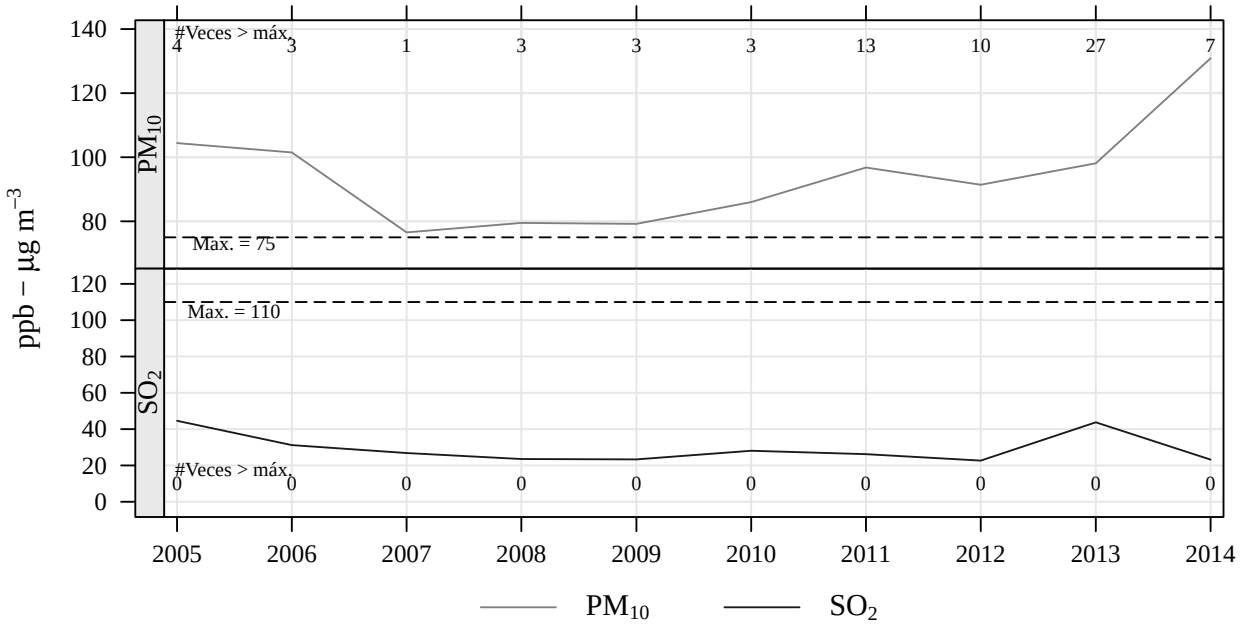

Figura C.11: PED - Referencia Promedio Diario

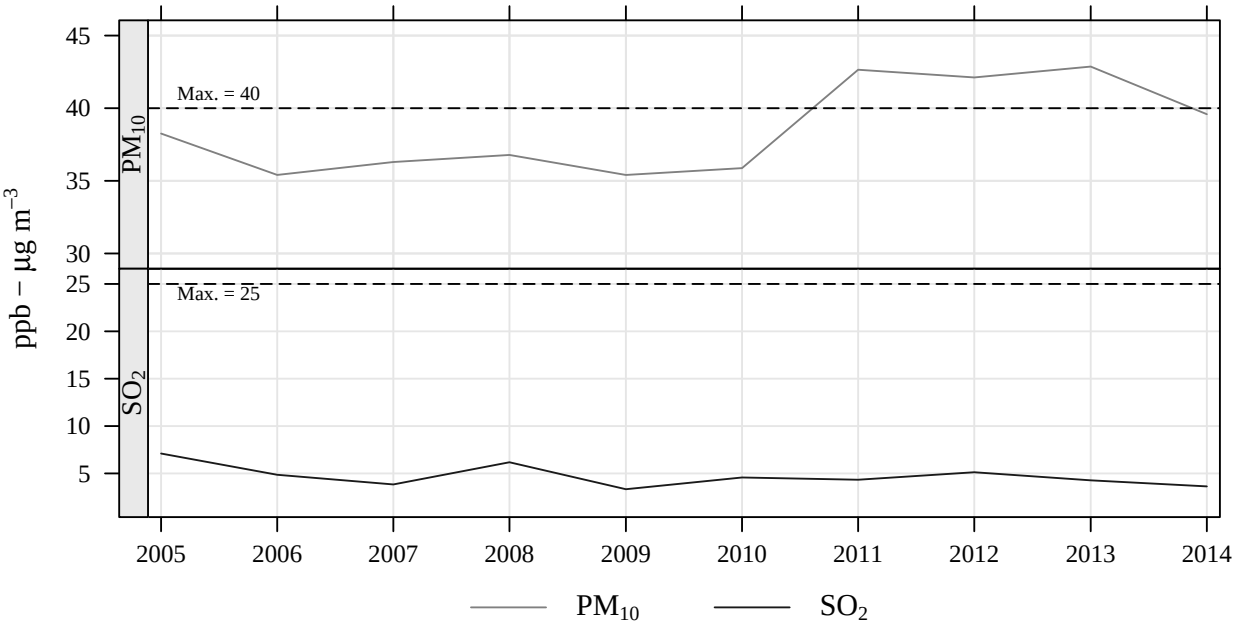

Figura C.12: PED - Referencia Promedio Anual

San Agustín - SAG

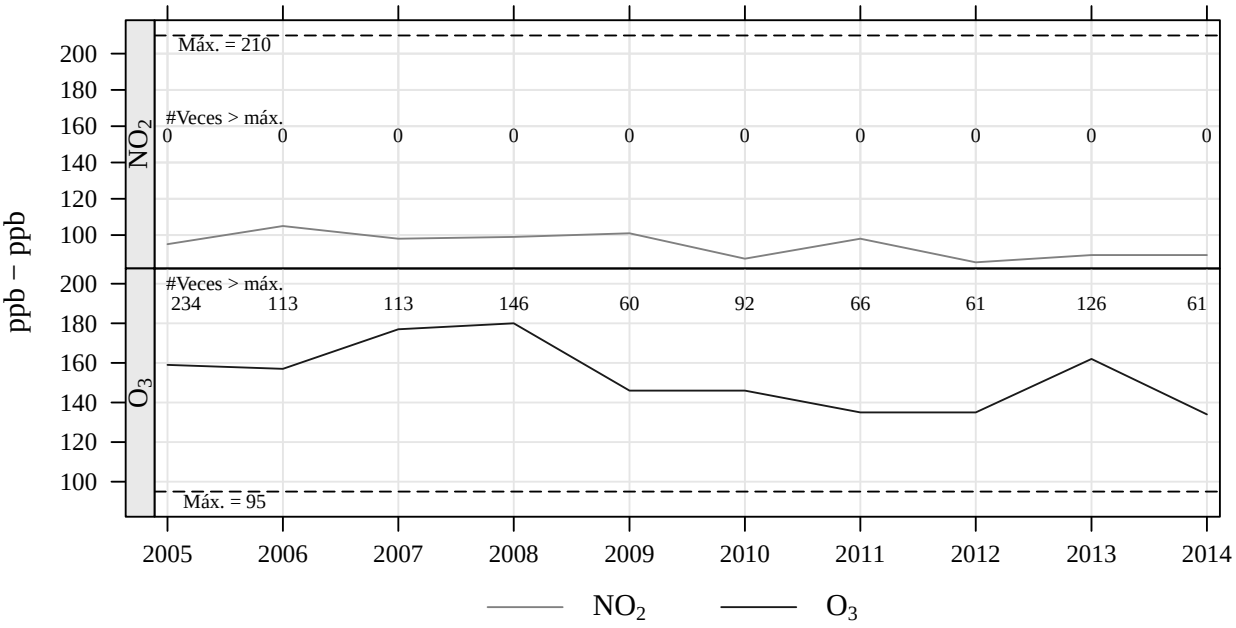

Figura C.13: SAG - Referencia Horaria

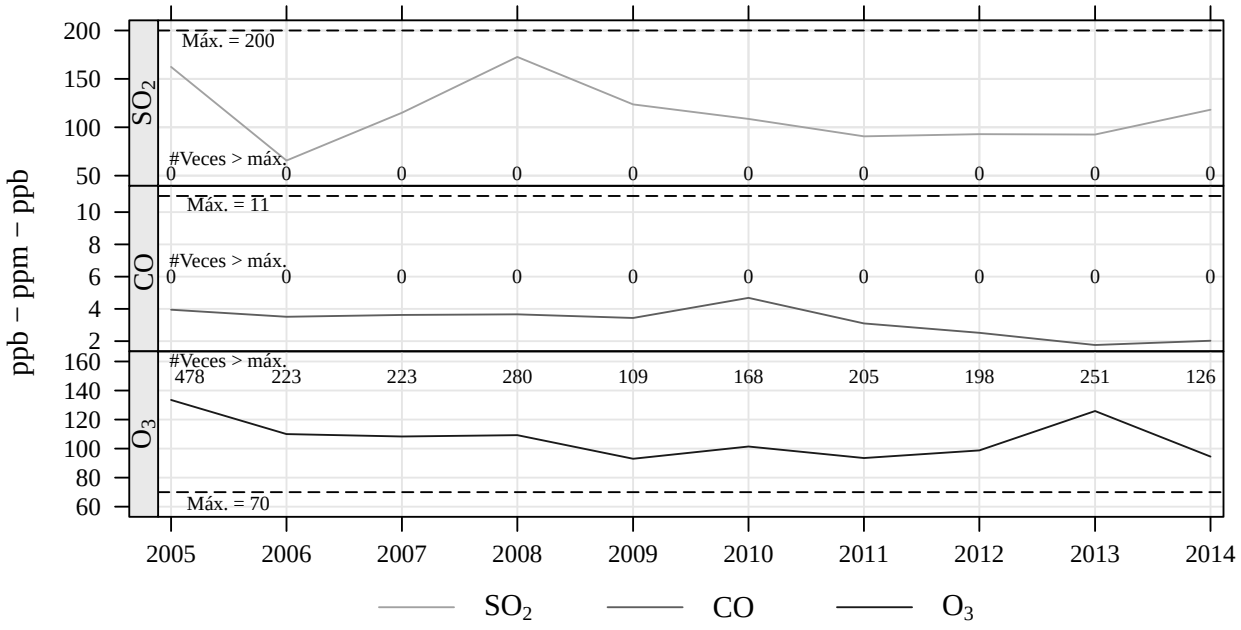

Figura C.14: SAG - Referencia Promedio Móvil 8 Horas

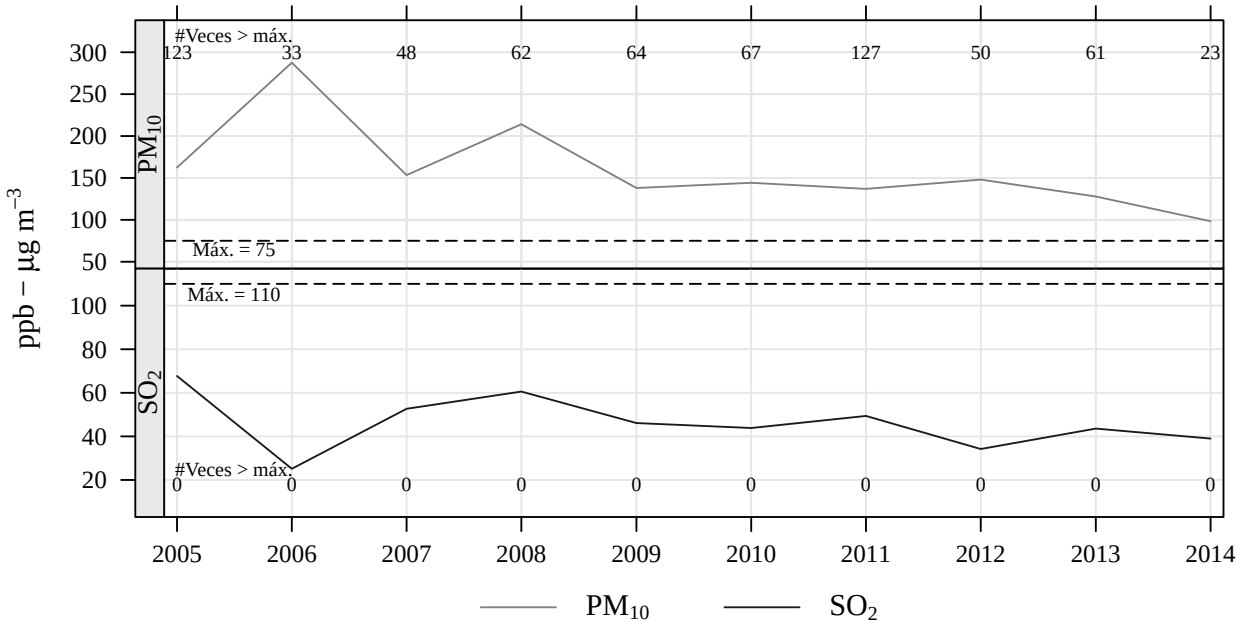

Figura C.15: SAG - Referencia Promedio Diario

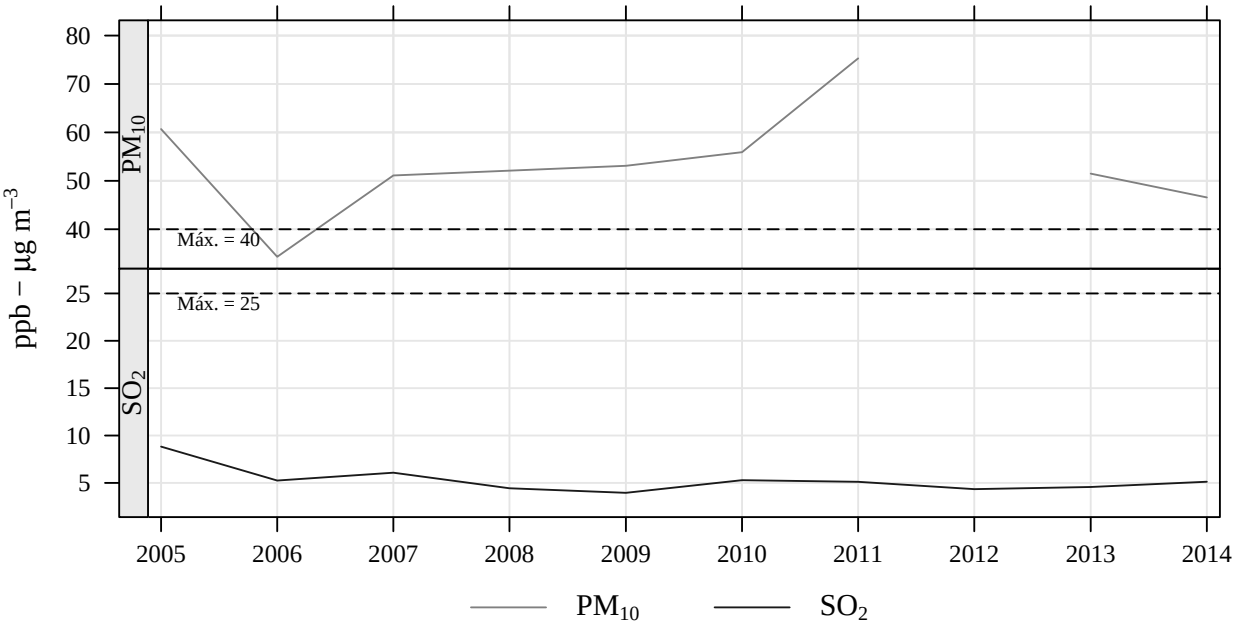

Figura C.16: SAG - Referencia Promedio Anual

Tláhuac - TAH

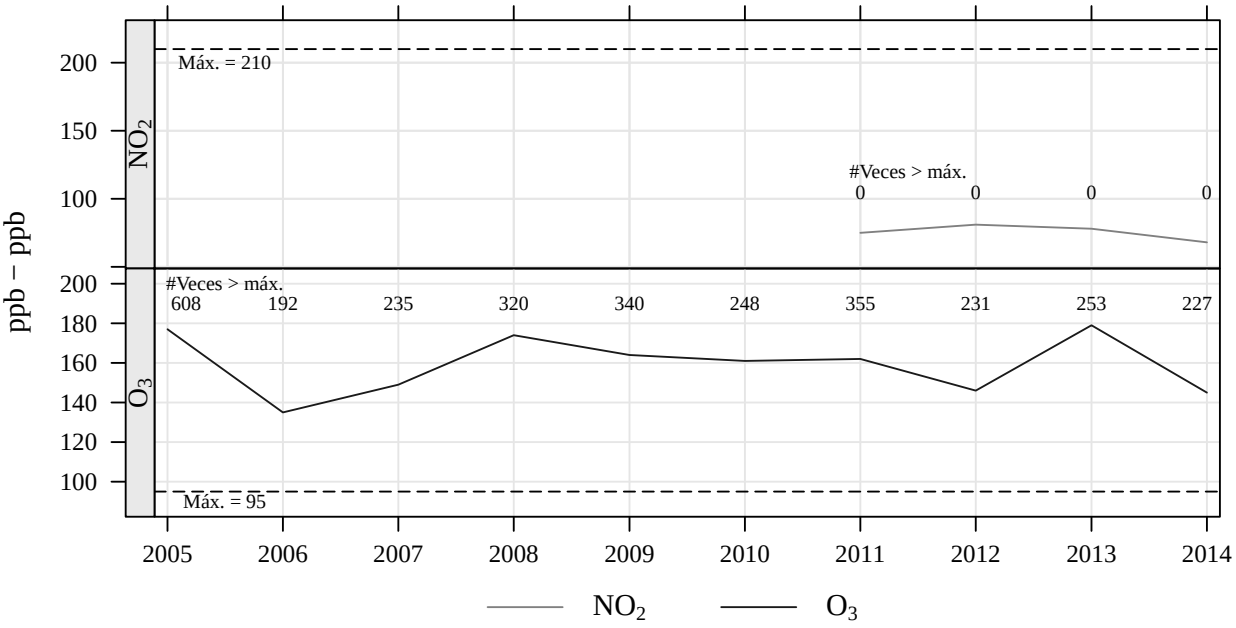

Figura C.17: TAH - Referencia Horaria

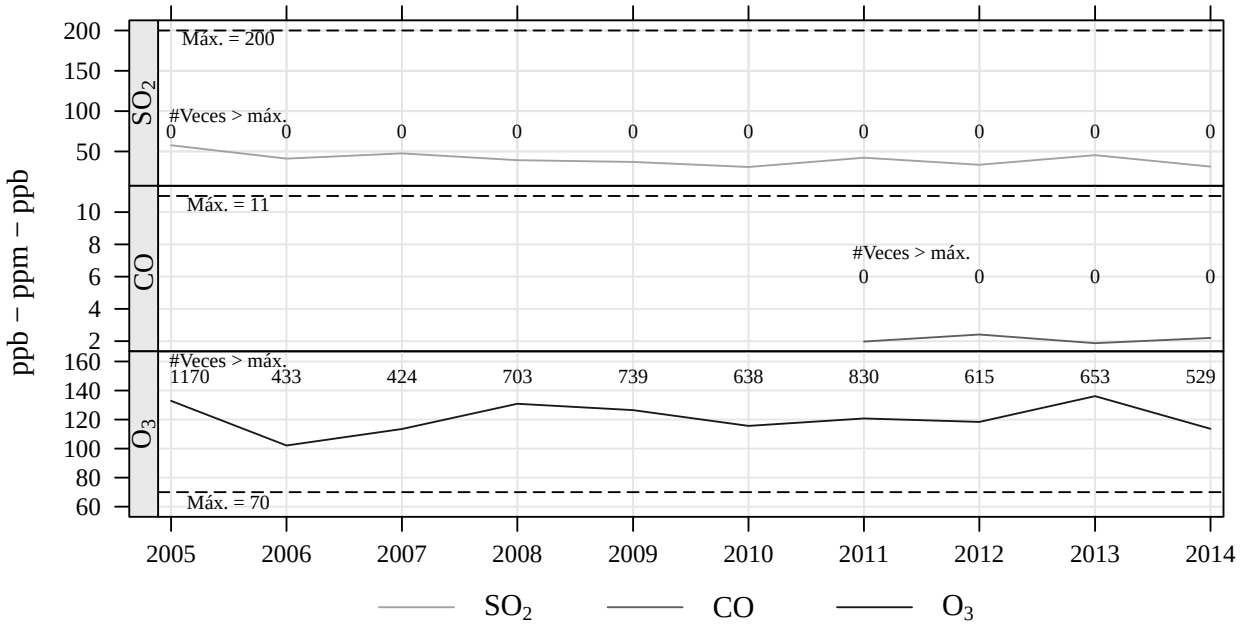

Figura C.18: TAH - Referencia Promedio Móvil 8 Horas

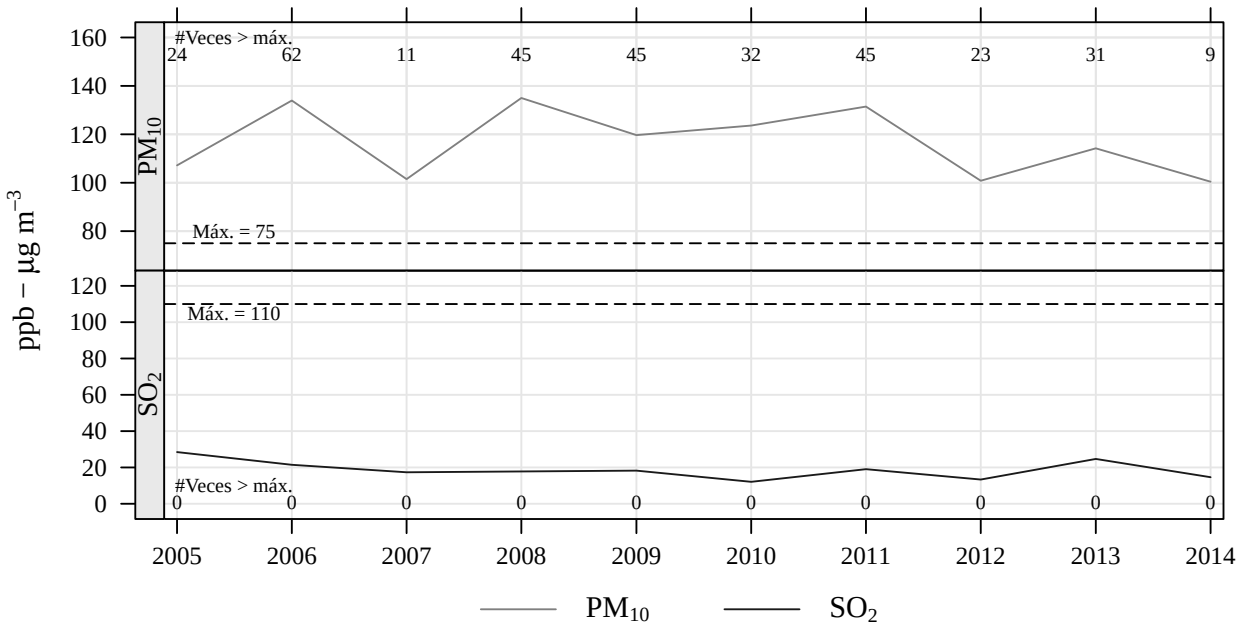

Figura C.19: TAH - Referencia Promedio Diario

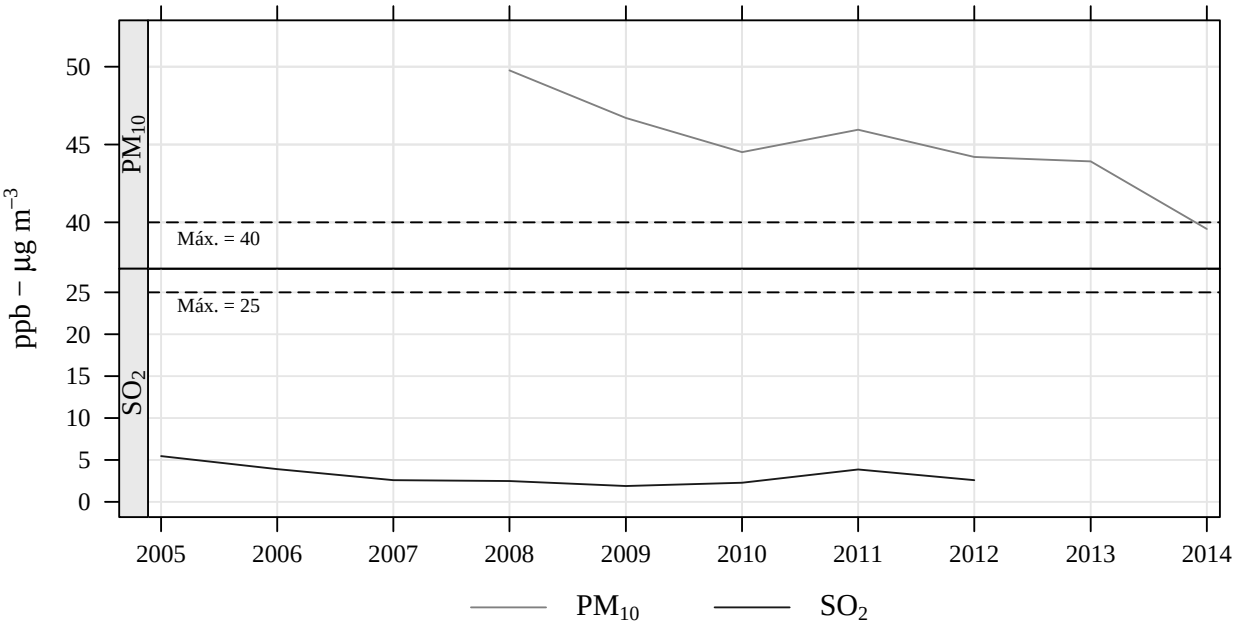

Figura C.20: TAH - Referencia Promedio Anual

[

Procesamiento inicial]Códigos para el procesamiento inicial de los datos

Extracción inicial de los archivos origen

```
#Cargar workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")
#Cargar paquete
library(reshape2)

#vectores con nombre
vector<-paste("base_",c(2005:2011,2014),sep="")
vector2<-paste("base_",2012:2013,sep="")
#Loop para extraer los contaminantes y estciones requerido
for(i in vector) {
#Extraer año
  j<-substr(i,6,9)
#Leer datos del año
  base<-read.csv(paste(paste("contaminantes",j,sep="_"),"csv",sep="."),
    colClasses = c("character","character","character",
          "numeric","numeric"),
    skip=8)
#Renombrar variables
  colnames(base)<-c("fecha","estacion","parametro","nivel","unidad")
#Extraer día, mes, año y hora
  base$d<-substr(base$fecha,1,2)
  base$m<-substr(base$fecha,4,5)
  base$y<-substr(base$fecha,7,10)
```

```
#Se resta 1 hora
  base$h<-(as.numeric(substr(base$fecha,12,13))*1)-1
#Formar la fecha en el formato adecuado y eliminar variables puente
  base$dia_hora<-as.POSIXct(paste(paste(base$y,base$m,
   base$d,sep="-"),paste(base$h,"00",sep=":"),sep=" "),
   format = "%Y-%m-%d %H:%M",tz="GMT")
  base$dia<-as.Date(paste(base$d,base$m,base$y,sep="-"),
   format = "%d-%m-%Y")
  base$dia_cad<-paste(base$y,base$m,base$d,sep="")
  base<-base[,-c(1,5:9)]
#Transponer datos
  p<-dcast(base, dia_hora + dia + dia_cad + estacion ~ parametro,
    value.var="nivel")
  attach(p)
#Elegir estaciones
  a<-p[which(estacion == "FAC" ),]
  b<-p[which(estacion == "MER" ),]
  c<-p[which(estacion == "PED" ),]
  d<-p[which(estacion == "TAH" ),]
  e<-p[which(estacion == "SAG" ),]
  v<-paste("fac_",i,sep="")
  w<-paste("mer_",i,sep="")
  x<-paste("ped_",i,sep="")
  y<-paste("tah_",i,sep="")
  z<-paste("sag_",i,sep="")
#Crear tabla para cada estación
  assign(v,a)
  assign(w,b)
  assign(x,c)
  assign(y,d)
  assign(z,e)
  detach(p)
  rm(base,p,a,b,c,d,e,v,w,x,y,z)
}

#Mismo loop pero con variaciones para 2012 y 2013 debido a diferencias
#en formto de fecha
for(i in vector2) {
#Extraer año
  j<-substr(i,6,9)
```

```
#Leer datos del año
  base<-read.csv(paste(paste("contaminantes",j,sep="_"),"csv",sep="."),
   colClasses = c("character","character","character",
      "numeric","numeric"),
    skip=8)
#Renombrar variables
  colnames(base)<-c("fecha","estacion","parametro","nivel","unidad")
#Extraer día, mes, año y hora
  base$d<-substr(base$fecha,9,10)
  base$m<-substr(base$fecha,6,7)
  base$y<-substr(base$fecha,1,4)
#Se resta 1 hora
  base$h<-(as.numeric(substr(base$fecha,12,13))*1)-1
#Formar la fecha en el formato adecuado y eliminar variables puente
  base$dia_hora<-as.POSIXct(paste(paste(base$y,base$m,base$d,sep="-"),
   paste(base$h,"00",sep=":"),sep=" "),
   format = "%Y-%m-%d %H:%M",tz="GMT")
  base$dia<-as.Date(paste(base$d,base$m,base$y,sep="-"),
   format = "%d-%m-%Y")
  base$dia_cad<-paste(base$y,base$m,base$d,sep="")
  base<-base[,-c(1,5:9)]
#Transponer datos
  p<-dcast(base, dia_hora + dia + dia_cad + estacion ~ parametro,
    value.var="nivel")
  attach(p)
#Elegir estaciones
  a<-p[which(estacion == "FAC" ),]
  b<-p[which(estacion == "MER" ),]
  c<-p[which(estacion == "PED" ),]
  d<-p[which(estacion == "TAH" ),]
  e<-p[which(estacion == "SAG" ),]
  v<-paste("fac_",i,sep="")
  w<-paste("mer_",i,sep="")
  x<-paste("ped_",i,sep="")
  y<-paste("tah_",i,sep="")
  z<-paste("sag_",i,sep="")
#Crear tabla para cada estación
  assign(v,a)
  assign(w,b)
  assign(x,c)
```

```
  assign(y,d)
  assign(z,e)
  detach(p)
  rm(base,p,a,b,c,d,e,v,w,x,y,z)
}

#Apilar todos los años para cada estación y eliminar bases por año
base_total_fac<-rbind(fac_base_2005,fac_base_2006,fac_base_2007,
fac_base_2008,fac_base_2009,fac_base_2010,fac_base_2011,
fac_base_2012,fac_base_2013,fac_base_2014)

rm(i,j,fac_base_2005,fac_base_2006,fac_base_2007,fac_base_2008,
   fac_base_2009,fac_base_2010,fac_base_2011,fac_base_2012,
   fac_base_2013,fac_base_2014)

base_total_mer<-rbind(mer_base_2005,mer_base_2006,mer_base_2007,
mer_base_2008,mer_base_2009,mer_base_2010,mer_base_2011,
mer_base_2012,mer_base_2013,mer_base_2014)

rm(mer_base_2005,mer_base_2006,mer_base_2007,mer_base_2008,
   mer_base_2009,mer_base_2010,mer_base_2011,mer_base_2012,
   mer_base_2013,mer_base_2014)

base_total_ped<-rbind(ped_base_2005,ped_base_2006,ped_base_2007,ped_base_2008,
ped_base_2009,ped_base_2010,ped_base_2011,
ped_base_2012,ped_base_2013,ped_base_2014)

rm(ped_base_2005,ped_base_2006,ped_base_2007,ped_base_2008,
   ped_base_2009,ped_base_2010,ped_base_2011,ped_base_2012,
   ped_base_2013,ped_base_2014)

base_total_tah<-rbind(tah_base_2005,tah_base_2006,tah_base_2007,tah_base_2008,
tah_base_2009,tah_base_2010,tah_base_2011,
tah_base_2012,tah_base_2013,tah_base_2014)

rm(tah_base_2005,tah_base_2006,tah_base_2007,tah_base_2008,
   tah_base_2009,tah_base_2010,tah_base_2011,tah_base_2012,
   tah_base_2013,tah_base_2014)

base_total_sag<-rbind(sag_base_2005,sag_base_2006,sag_base_2007,sag_base_2008,
```

```
sag_base_2009,sag_base_2010,sag_base_2011,
sag_base_2012,sag_base_2013,sag_base_2014)

rm(sag_base_2005,sag_base_2006,sag_base_2007,sag_base_2008,
   sag_base_2009,sag_base_2010,sag_base_2011,sag_base_2012,
   sag_base_2013,sag_base_2014)
rm(vector,vector2)

#Cambiar nombre de tablas y variables
fac<-base_total_fac
mer<-base_total_mer
ped<-base_total_ped
tah<-base_total_tah
sag<-base_total_sag

rm(base_total_fac,base_total_mer,base_total_ped,
   base_total_tah,base_total_sag)

names(fac)[1]<-"date"
names(mer)[1]<-"date"
names(ped)[1]<-"date"
names(sag)[1]<-"date"
names(tah)[1]<-"date"

#Guardar información
save.image("informacion_estaciones.RData")
```

Transformación de datos horarios

```
#Cargar workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar datos
load("informacion_estaciones.RData")

#Cargar paquetes
library("openair")
library("psych")
```

```
#Cambiar nombre a variables
fac_com<-fac[,-c(2,3,4)]
colnames(fac_com)<-c("date","co","no","no2","nox","o3","pm10","pm25","so2")
mer_com<-mer[,-c(2,3,4)]
colnames(mer_com)<-c("date","co","no","no2","nox","o3","pm10","pm25","so2")
ped_com<-ped[,-c(2,3,4)]
colnames(ped_com)<-c("date","co","no","no2","nox","o3","pm10","pm25","so2")
sag_com<-sag[,-c(2,3,4)]
colnames(sag_com)<-c("date","co","no","no2","nox","o3","pm10","pm25","so2")
tah_com<-tah[,-c(2,3,4)]
colnames(tah_com)<-c("date","co","no","no2","nox","o3","pm10","pm25","so2")

#ROLLING AVERAGE PARA
#FES ACATLÁN
fac_com <- rollingMean(fac_com, pollutant = "so2", hours = 8,
                       new.name = "so2_rolling", data.thresh = 75)
fac_com <- rollingMean(fac_com, pollutant = "co", hours = 8,
                       new.name = "co_rolling", data.thresh = 75)
fac_com <- rollingMean(fac_com, pollutant = "o3", hours = 8,
                       new.name = "o3_rolling", data.thresh = 75)
#MERCED
mer_com <- rollingMean(mer_com, pollutant = "so2", hours = 8,
                       new.name = "so2_rolling", data.thresh = 75)
mer_com <- rollingMean(mer_com, pollutant = "co", hours = 8,
                       new.name = "co_rolling", data.thresh = 75)
mer_com <- rollingMean(mer_com, pollutant = "o3", hours = 8,
                       new.name = "o3_rolling", data.thresh = 75)

#PEDREGAL
ped_com <- rollingMean(ped_com, pollutant = "so2", hours = 8,
                       new.name = "so2_rolling", data.thresh = 75)
ped_com <- rollingMean(ped_com, pollutant = "co", hours = 8,
                       new.name = "co_rolling", data.thresh = 75)
ped_com <- rollingMean(ped_com, pollutant = "o3", hours = 8,
                       new.name = "o3_rolling", data.thresh = 75)

#SAN AGUSÍN
sag_com <- rollingMean(sag_com, pollutant = "so2", hours = 8,
                       new.name = "so2_rolling", data.thresh = 75)
```

```
sag_com <- rollingMean(sag_com, pollutant = "co", hours = 8,
                       new.name = "co_rolling", data.thresh = 75)
sag_com <- rollingMean(sag_com, pollutant = "o3", hours = 8,
                       new.name = "o3_rolling", data.thresh = 75)

#TLAHUAC
tah_com <- rollingMean(tah_com, pollutant = "so2", hours = 8,
                       new.name = "so2_rolling", data.thresh = 75)
tah_com <- rollingMean(tah_com, pollutant = "co", hours = 8,
                       new.name = "co_rolling", data.thresh = 75)
tah_com <- rollingMean(tah_com, pollutant = "o3", hours = 8,
                       new.name = "o3_rolling", data.thresh = 75)

#PARA SO2 SE CALCULA EL PROMEDIO DIARIO
#SE CALCULA  PROMEDIOS MÓVILES DE 8 HORAS
#SE CALCULA EL PROMEDIO ANUAL
#PARA CO SE CALCULA EL PROMEDIO MOVIL DE 8 HORAS
#PARA NO2 NO SE CALCULA PUES ES POR HORA
#PARA 03 ES POR HORA Y SE CALCULA EL PROMEDIO MOVIL DE 8 HORAS
#PARA PM10 SE CALCULA EL PROMEDIO DIARIO Y EL PROMEDIO ANUAL
#PARA PM25 SE CALCULA EL PROMEDIO DIARIO Y EL PROMEDIO ANUAL

#EL PROMEDIO DIARIO SE CALCULA PARA SO2 PM10 PM25 PEO SE ESTABLECE
#PARA TODOS PARA SABER SI CUMPLEN EL 75%
#AGREGAR POR DÍA, MES, TRIMESTRE Y AÑO PARA TODAS LAS ESTACIONES
#DÍA
fac_d <- timeAverage(fac_com, avg.time = "day", data.thresh = 75,
                     statistic = "mean",vector.ws=T)
mer_d <- timeAverage(mer_com, avg.time = "day", data.thresh = 75,
                     statistic = "mean",vector.ws=T)
ped_d <- timeAverage(ped_com, avg.time = "day", data.thresh = 75,
                     statistic = "mean",vector.ws=T)
sag_d <- timeAverage(sag_com, avg.time = "day", data.thresh = 75,
                     statistic = "mean",vector.ws=T)
tah_d <- timeAverage(tah_com, avg.time = "day", data.thresh = 75,
                     statistic = "mean",vector.ws=T)
#MES
fac_m <- timeAverage(fac_d, avg.time = "month", data.thresh = 75,
                     statistic = "mean",vector.ws=T)
mer_m <- timeAverage(mer_d, avg.time = "month", data.thresh = 75,
```

```
                      statistic = "mean",vector.ws=T)
ped_m <- timeAverage(ped_d, avg.time = "month", data.thresh = 75,
                      statistic = "mean",vector.ws=T)
sag_m <- timeAverage(sag_d, avg.time = "month", data.thresh = 75,
                      statistic = "mean",vector.ws=T)
tah_m <- timeAverage(tah_d, avg.time = "month", data.thresh = 75,
                      statistic = "mean",vector.ws=T)
#TRIMESTRE
fac_q <- timeAverage(fac_d, avg.time = "quarter", data.thresh = 75,
                      statistic = "mean",vector.ws=T)
mer_q <- timeAverage(mer_d, avg.time = "quarter", data.thresh = 75,
                      statistic = "mean",vector.ws=T)
ped_q <- timeAverage(ped_d, avg.time = "quarter", data.thresh = 75,
                      statistic = "mean",vector.ws=T)
sag_q <- timeAverage(sag_d, avg.time = "quarter", data.thresh = 75,
                      statistic = "mean",vector.ws=T)
tah_q <- timeAverage(tah_d, avg.time = "quarter", data.thresh = 75,
                      statistic = "mean",vector.ws=T)
#AÑO
fac_y <- timeAverage(fac_q, avg.time = "year",
                      statistic = "mean",vector.ws=T)
mer_y <- timeAverage(mer_q, avg.time = "year",
                      statistic = "mean",vector.ws=T)
ped_y <- timeAverage(ped_q, avg.time = "year",
                      statistic = "mean",vector.ws=T)
sag_y <- timeAverage(sag_q, avg.time = "year",
                      statistic = "mean",vector.ws=T)
tah_y <- timeAverage(tah_q, avg.time = "year",
                      statistic = "mean",vector.ws=T)

#AGREGAR VARIABLE DE AÑO
fac_y$year <- format(fac_y$date,"%Y")
mer_y$year <- format(mer_y$date,"%Y")
ped_y$year <- format(ped_y$date,"%Y")
sag_y$year <- format(sag_y$date,"%Y")
tah_y$year <- format(tah_y$date,"%Y")

fac_com$year <- format(fac_com$date, "%Y")
mer_com$year <- format(mer_com$date, "%Y")
ped_com$year <- format(ped_com$date, "%Y")
```

```
tah_com$year <- format(tah_com$date, "%Y")
sag_com$year <- format(sag_com$date, "%Y")

#BORRAR BASES COMPLETAS
rm(fac,mer,ped,sag,tah)

#FAC-FES ACATLÁN
#DATOS NO VÁLIDOS
temp_facq<-fac_q
temp_facq$year<-format(temp_facq$date,"%Y")
des_facq<-describeBy(temp_facq[,c(1:8,16)],temp_facq$year,mat=TRUE)
des_facq<-des_facq[des_facq$n<3,c(2,4)]
#2005 2006 SO2
fac_y[fac_y$year %in% c("2005","2006"),"so2"]<-NA

#MER-MERCED
#DATOS NO VÁLIDOS
temp_merq<-mer_q
temp_merq$year<-format(temp_merq$date,"%Y")
des_merq<-describeBy(temp_merq[,c(1:8,16)],
                     temp_merq$year,mat=TRUE)
des_merq<-des_merq[des_merq$n<3,c(2,4)]
#2014 no2 no nox
mer_y[mer_y$year %in% c("2014"),c("no2","no","nox")]<-NA

#PED-PEDREGAL
#DATOS NO VÁLIDOS
temp_pedq<-ped_q
temp_pedq$year<-format(temp_pedq$date,"%Y")
des_pedq<-describeBy(temp_pedq[,c(1:8,16)],
                     temp_pedq$year,mat=TRUE)
des_pedq<-des_pedq[des_pedq$n<3,c(2,4)]

#2014 co 2005-2011 pm25
ped_y[ped_y$year %in% c("2014"),c("co")]<-NA
ped_y[ped_y$year %in% c("2011"),c("pm25")]<-NA

#SAG - SAN AGUSTIN
#DATOS NO VÁLIDOS
temp_sagq<-sag_q
```

```
temp_sagq$year<-format(temp_sagq$date,"%Y")
des_sagq<-describeBy(temp_sagq[,c(1:8,16)],
                     temp_sagq$year,mat=TRUE)
des_sagq<-des_sagq[des_sagq$n<3,c(2,4)]

#2014 2013 2012 co
#2012 no
#2012 no2
#2012 nox
#2012 03
#2012 pm10
#2012 pm25
#2012 o3_rolling

sag_y[sag_y$year %in% c("2012"),c("co","no","no2","nox",
 "o3","pm10","pm25","o3_rolling")]<-NA
sag_y[sag_y$year %in% c("2013","2014"),c("co")]<-NA

#TAH - TLAHUAC
#DATOS NO VÁLIDOS
temp_tahq<-tah_q
temp_tahq$year<-format(temp_tahq$date,"%Y")
des_tahq<-describeBy(temp_tahq[,c(1:8,16)],
                     temp_tahq$year,mat=TRUE)
des_tahq<-des_tahq[des_tahq$n<3,c(2,4)]

#2012-2005 co
#2011-2005 no
#2011-2005 no2
#2011-2005 nox
#2005 2007 pm10
#2013 so2

tah_y[tah_y$year %in% c("2011","2010","2009","2008","2007","2006","2005"),
      c("co","no","no2","nox")]<-NA
tah_y[tah_y$year %in% c("2012"),c("co")]<-NA
tah_y[tah_y$year %in% c("2005","2007"),c("pm10")]<-NA
tah_y[tah_y$year %in% c("2013"),c("so2")]<-NA

rm(des_facq,des_merq,des_pedq,des_sagq,des_tahq,
```

```
   temp_facq,temp_merq,temp_pedq,temp_sagq,temp_tahq)

#CONSTRUCCIÓN DE TABLA PM10 CON LAS 5 ESTACIONES CONCETRACIÓN HORARIA
fac_pm10<-fac_com[,c("date","pm10")]
colnames(fac_pm10)[2]<-"pm10.FAC"

mer_pm10<-mer_com[,c("date","pm10")]
colnames(mer_pm10)[2]<-"pm10.MER"

ped_pm10<-ped_com[,c("date","pm10")]
colnames(ped_pm10)[2]<-"pm10.PED"

sag_pm10<-sag_com[,c("date","pm10")]
colnames(sag_pm10)[2]<-"pm10.SAG"

tah_pm10<-tah_com[,c("date","pm10")]
colnames(tah_pm10)[2]<-"pm10.TAH"

pm.10<-merge(fac_pm10,mer_pm10,by="date")
pm.10<-merge(pm.10,ped_pm10,by="date")
pm.10<-merge(pm.10,sag_pm10,by="date")
pm.10<-merge(pm.10,tah_pm10,by="date")

#CONSTRUCCIÓN DE TABLA PM10 CON LAS 5 ESTACIONES CONCENTRACIÓN DIARIA
facd_pm10<-fac_d[,c("date","pm10")]
colnames(facd_pm10)[2]<-"pm10.FAC"

merd_pm10<-mer_d[,c("date","pm10")]
colnames(merd_pm10)[2]<-"pm10.MER"

pedd_pm10<-ped_d[,c("date","pm10")]
colnames(pedd_pm10)[2]<-"pm10.PED"

sagd_pm10<-sag_d[,c("date","pm10")]
colnames(sagd_pm10)[2]<-"pm10.SAG"

tahd_pm10<-tah_d[,c("date","pm10")]
colnames(tahd_pm10)[2]<-"pm10.TAH"

pmd.10<-merge(facd_pm10,merd_pm10,by="date")
```

```
pmd.10<-merge(pmd.10,pedd_pm10,by="date")
pmd.10<-merge(pmd.10,sagd_pm10,by="date")
pmd.10<-merge(pmd.10,tahd_pm10,by="date")

#CONSTRUCCIÓN DE TABLA PM10 CON LAS 5 ESTACIONES CONCENTRACIÓN ANUAL
facy_pm10<-fac_y[,c("date","pm10")]
colnames(facy_pm10)[2]<-"pm10.FAC"

mery_pm10<-mer_y[,c("date","pm10")]
colnames(mery_pm10)[2]<-"pm10.MER"

pedy_pm10<-ped_y[,c("date","pm10")]
colnames(pedy_pm10)[2]<-"pm10.PED"

sagy_pm10<-sag_y[,c("date","pm10")]
colnames(sagy_pm10)[2]<-"pm10.SAG"

tahy_pm10<-tah_y[,c("date","pm10")]
colnames(tahy_pm10)[2]<-"pm10.TAH"

pmy.10<-merge(facy_pm10,mery_pm10,by="date")
pmy.10<-merge(pmy.10,pedy_pm10,by="date")
pmy.10<-merge(pmy.10,sagy_pm10,by="date")
pmy.10<-merge(pmy.10,tahy_pm10,by="date")

#GUARDAR DATOS
save.image("tablas_procesadas.RData")
```

[

Creación de gráficos]Códigos para la creación de gráficos

RESUMEN

```
#Cargar workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar datos
load("tablas_procesadas.RData")

#Cargar paquete
library("openair")

#SUMMARYPLOTS
#CALENDAR PLOT PARA FAC-PM10 AÑOS 2014, 2013 Y 2005
#2014
pdf(file="calendar_2014.pdf",height=5,width=10,family="Times",
pointsize=8)
calendarPlot(fac_com, pollutant = "pm10", year="2014",statisic="mean",
             annotate="value",lim=75,data.thresh=75,
             cols = c("white", "whitesmoke", "grey70","grey30"),
             cex.lim=c(0.6,0.6))
dev.off()

#2013
pdf(file="calendar_2013.pdf",height=5,width=10,family="Times",
pointsize=8)
```

```
calendarPlot(fac_com, pollutant = "pm10", year="2013",statisic="mean",
             annotate="value",lim=75,data.thresh=75,
             cols = c("white", "whitesmoke", "grey70","grey30"),
             cex.lim=c(0.6,0.6))
dev.off()

#2005
pdf(file="calendar_2005.pdf",height=5,width=10,family="Times",pointsize=8)
calendarPlot(fac_com, pollutant = "pm10", year="2005",statisic="mean",
             annotate="value",lim=75,data.thresh=75,
             cols = c("white", "whitesmoke", "grey70","grey30"),
             cex.lim=c(0.6,0.6))
dev.off()

#SUMMARYPLOT PM10
pdf(file="pm10_summaryPlot.pdf",family="Times",pointsize=8)
summaryPlot(pm.10,na.le=1,clip=FALSE,type="density",cols="greyscale",
            xlab=c("Año","Concentración"),
            ylab=c("Estaciones de Monitoreo","Densidad"))
dev.off()

#SUMMARYPLOT PARA TODAS LAS ESTACIONES (DATOS HORARIOS)
#FAC
pdf(file="summaryPlot_fac.pdf",family="Times",pointsize=10)
summaryPlot(fac_com[,c("date","co","no2","o3","so2","pm10")],na.le=1,
            clip=FALSE,type="density",cols="greyscale",
            xlab=c("Año","Concentración"),ylab=c("Fes Acatlán","Densidad"))
dev.off()

#MER
pdf(file="summaryPlot_mer.pdf",family="Times",pointsize=10)
summaryPlot(mer_com[,c("date","co","no2","o3","so2","pm10")],na.le=1,
            clip=FALSE,type="density",cols="greyscale",
            xlab=c("Año","Concentración"),ylab=c("Merced","Densidad"))
dev.off()

#PED
pdf(file="summaryPlot_ped.pdf",family="Times",pointsize=10)
summaryPlot(ped_com[,c("date","co","no2","o3","so2","pm10")],na.le=1,
            clip=FALSE,type="density",cols="greyscale",
```

```
            xlab=c("Año","Concentración"),ylab=c("Pedregal","Densidad"))
dev.off()

#SAG
pdf(file="summaryPlot_sag.pdf",family="Times",pointsize=10)
summaryPlot(sag_com[,c("date","co","no2","o3","so2","pm10")],na.le=1,
            clip=FALSE,type="density",cols="greyscale",
            xlab=c("Año","Concentración"),
            ylab=c("San Agustín","Densidad"))
dev.off()

#TAH
pdf(file="summaryPlot_tah.pdf",family="Times",pointsize=10)
summaryPlot(tah_com[,c("date","co","no2","o3","so2","pm10")],na.le=1,
            clip=FALSE,type="density",cols="greyscale",
            xlab=c("Año","Concentración"),ylab=c("Tláhuac","Densidad"))
dev.off()
```

FES ACATLÁN

```
#Cargar workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar datos
load("tablas_procesadas.RData")

#Cargar paquete
library("openair")
library(latticeExtra)

#Obtener y guardar el número de días que se excedió
#el valor de referencia

fd_pm10<-group_by(fac_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
75)), var = as.name("pm10")))
fd_pm10<-as.character(as.vector(fd_pm10$days))
```

```
fd_so2<-group_by(fac_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
110)), var = as.name("so2")))
fd_so2<-as.character(as.vector(fd_so2$days))

fr_so2<-group_by(fac_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
200)), var = as.name("so2_rolling")))
fr_so2<-as.character(as.vector(fr_so2$days))

fr_co<-group_by(fac_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
11)), var = as.name("co_rolling")))
fr_co<-as.character(as.vector(fr_co$days))

fr_o3<-group_by(fac_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
70)), var = as.name("o3_rolling")))
fr_o3<-as.character(as.vector(fr_o3$days))

fh_no2<-group_by(fac_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
210)), var = as.name("no2")))
fh_no2<-as.character(as.vector(fh_no2$days))

fh_o3<-group_by(fac_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
95)), var = as.name("o3")))
fh_o3<-as.character(as.vector(fh_o3$days))

#Lugar en el gráfico en donde se colocarán los valores externos
x<-c(as.POSIXct("2005-01-01"),as.POSIXct("2006-01-01"),
     as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
```

```
     as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
     as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
     as.POSIXct("2013-01-01"),as.POSIXct("2014-01-01"))
x1<-c(as.POSIXct("2005-03-01"),as.POSIXct("2006-01-01"),
     as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
     as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
     as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
     as.POSIXct("2013-01-01"),as.POSIXct("2013-12-01"))

xlabel<-as.POSIXct("2005-07-01")
label<-"#Veces > máx."

#Gráficos de valores diarios
prom24_fac<-timePlot(fac_d, pollutant = c("pm10","so2"),
 avg.time="year",
                     statistic="max",
                     cols="greyscale",
                     ylab = expression("ppb - " * mu * "g m" ^-3),
                     y.relation="free",
                     ylim = list(c(70, 130), c(0, 120)))
prom24_fac<-trellis.last.object() +
  layer(panel.abline(h=75, lty = 5),rows = 1)
prom24_fac
prom24_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 72, labels = "Máx. = 75", cex=0.65),
        rows = 1)
prom24_fac
prom24_fac<-trellis.last.object() +
  layer(ltext(x = x, y = 126, labels = fd_pm10, cex=0.65), rows = 1)
prom24_fac
prom24_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 131,
              labels = label, cex=0.65), rows = 1)
prom24_fac
prom24_fac<-trellis.last.object() +
  layer(panel.abline(h=110, lty = 5),rows = 2)
prom24_fac
prom24_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 104, labels = "Máx. = 110", cex=0.65),
        rows = 2)
```

```
prom24_fac
prom24_fac<-trellis.last.object() +
  layer(ltext(x = x, y = 20, labels = fd_so2, cex=0.65), rows = 2)
prom24_fac
prom24_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 29, labels = label, cex=0.65), rows = 2)
prom24_fac
pdf(file="prom24_fac.pdf",height=4,width=7,family="Times",
pointsize=10,colormodel="grey")
print(prom24_fac)
dev.off()

#Gráficos de valores anuales
prom365_fac<-timePlot(fac_y, pollutant = c("pm10","so2"),
  cols="greyscale",
                      ylab = expression("ppb - " * mu * "g m" ^-3),
                      y.relation="free",
                      ylim = list(c(38, 52), c(6, 26)))
prom365_fac<-trellis.last.object() +
  layer(panel.abline(h=40, lty = 5),rows = 1)
prom365_fac
prom365_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 39, labels = "Máx. = 40", cex=0.65),
        rows = 1)
prom365_fac
prom365_fac<-trellis.last.object() +
  layer(panel.abline(h=25, lty = 5),rows = 2)
prom365_fac
prom365_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 23.5, labels = "Máx. = 25", cex=0.65),
        rows = 2)
prom365_fac
pdf(file="prom365_fac.pdf",height=4,width=7,family="Times",
pointsize=10,colormodel="grey")
print(prom365_fac)
dev.off()

#Gráficos de valores promedio móvil
promroll_fac<-timePlot(fac_com, pollutant = c("so2_rolling",
   "co_rolling","o3_rolling"),
```

```
                        avg.time="year",
                        statistic="max",
                        cols="greyscale",
                        ylab="ppb - ppm - ppb",
                        y.relation="free",
                        ylim = list(c(50, 200), c(2, 11),c(60,160)),
                        name.pol=c("so2","co","o3"))
promroll_fac<-trellis.last.object() +
  layer(panel.abline(h=200, lty = 5),rows = 1)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 185, labels = "Máx. = 200", cex=0.65),
        rows = 1)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(ltext(x = x, y = 52, labels = fr_so2, cex=0.65), rows = 1)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 68,
              labels = label, cex=0.65), rows = 1)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(panel.abline(h=11, lty = 5),rows = 2)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 10, labels = "Máx. = 11", cex=0.65),
        rows = 2)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(ltext(x = x, y = 6, labels = fr_co, cex=0.65), rows = 2)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 7,
              labels = label, cex=0.65), rows = 2)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(panel.abline(h=70, lty = 5),rows = 3)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 80, labels = "Máx. = 70", cex=0.65),
```

```
        rows = 3)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(ltext(x = x1, y = 100, labels = fr_o3, cex=0.65), rows = 3)
promroll_fac
promroll_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 115, labels = label, cex=0.65), rows = 3)
promroll_fac
pdf(file="rolling_fac.pdf",height=4,width=7,family="Times",
pointsize=5,colormodel="grey")
print(promroll_fac)
dev.off()

#Gráficos de valores horarios
promh_fac<-timePlot(fac_com, pollutant = c("no2","o3"),
                    avg.time="year",
                    statistic="max",
                    cols="greyscale",
                    ylab="ppb - ppb",
                    y.relation="free",
                    ylim = list(c(90, 250),c(90,200)))
promh_fac<-trellis.last.object() +
  layer(panel.abline(h=210, lty = 5),rows = 1)
promh_fac
promh_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 200, labels = "Máx. = 210", cex=0.65),
        rows = 1)
promh_fac
promh_fac<-trellis.last.object() +
  layer(ltext(x = x, y = 105, labels = fh_no2, cex=0.65), rows = 1)
promh_fac
promh_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 95, labels = label, cex=0.65), rows = 1)
promh_fac
promh_fac<-trellis.last.object() +
  layer(panel.abline(h=95, lty = 5),rows = 2)
promh_fac
promh_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 90, labels = "Máx. = 95", cex=0.65),
        rows = 2)
```

```
promh_fac
promh_fac<-trellis.last.object() +
  layer(ltext(x = x1, y = 140, labels = fh_o3, cex=0.65), rows = 2)
promh_fac
promh_fac<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 150, labels = label, cex=0.65), rows = 2)
promh_fac
pdf(file="prom_fac.pdf",height=4,width=7,family="Times",
pointsize=10,colormodel="grey")
print(promh_fac)
dev.off()
```

MERCED

```
#Cargar workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar datos
load("tablas_procesadas.RData")

#Cargar paquete
library("openair")
library(latticeExtra)

#Obtener y guardar el número de días que se excedió
#el valor de referencia

md_pm10<-group_by(mer_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
75)), var = as.name("pm10")))
md_pm10<-as.character(as.vector(md_pm10$days))

md_so2<-group_by(mer_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
110)), var = as.name("so2")))
md_so2<-as.character(as.vector(md_so2$days))
```

```
mr_so2<-group_by(mer_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
200)), var = as.name("so2_rolling")))
mr_so2<-as.character(as.vector(mr_so2$days))

mr_co<-group_by(mer_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
11)), var = as.name("co_rolling")))
mr_co<-as.character(as.vector(mr_co$days))

mr_o3<-group_by(mer_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
70)), var = as.name("o3_rolling")))
mr_o3<-as.character(as.vector(mr_o3$days))

mh_no2<-group_by(mer_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
210)), var = as.name("no2")))
mh_no2<-as.character(as.vector(mh_no2$days))

mh_o3<-group_by(mer_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
95)), var = as.name("o3")))
mh_o3<-as.character(as.vector(mh_o3$days))

#Lugar en el gráfico en donde se colocarán los valores externos
x<-c(as.POSIXct("2005-01-01"),as.POSIXct("2006-01-01"),
     as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
     as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
     as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
     as.POSIXct("2013-01-01"),as.POSIXct("2014-01-01"))
x1<-c(as.POSIXct("2005-03-01"),as.POSIXct("2006-01-01"),
      as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
      as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
```

```
      as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
      as.POSIXct("2013-01-01"),as.POSIXct("2014-01-01"))

xlabel<-as.POSIXct("2005-07-01")
label<-"#Veces > máx."

#Gráficos de valores diarios
prom24_mer<-timePlot(mer_d, pollutant = c("pm10","so2"),avg.time="year",
                     statistic="max",
                     cols="greyscale",
                     ylab = expression("ppb - " * mu * "g m" ^-3),
                     y.relation="free",
                     ylim = list(c(70, 180), c(0, 120)))
prom24_mer<-trellis.last.object() +
  layer(panel.abline(h=75, lty = 5),rows = 1)
prom24_mer
prom24_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 70, labels = "Máx. = 75", cex=0.65),
        rows = 1)
prom24_mer
prom24_mer<-trellis.last.object() +
  layer(ltext(x = x, y = 170, labels = md_pm10, cex=0.65), rows = 1)
prom24_mer
prom24_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 180,
              labels = label, cex=0.65), rows = 1)
prom24_mer
prom24_mer<-trellis.last.object() +
  layer(panel.abline(h=110, lty = 5),rows = 2)
prom24_mer
prom24_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 105, labels = "Máx. = 110", cex=0.65),
        rows = 2)
prom24_mer
prom24_mer<-trellis.last.object() +
  layer(ltext(x = x, y = 5, labels = md_so2, cex=0.65), rows = 2)
prom24_mer
prom24_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 18, labels = label, cex=0.65), rows = 2)
prom24_mer
```

```
pdf(file="prom24_mer.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(prom24_mer)
dev.off()

#Gráficos de valores anuales
prom365_mer<-timePlot(mer_y, pollutant = c("pm10","so2"),cols="greyscale",
                      ylab = expression("ppb - " * mu * "g m" ^-3),
                      y.relation="free",
                      ylim = list(c(38, 70), c(3, 26)))
prom365_mer<-trellis.last.object() +
  layer(panel.abline(h=40, lty = 5),rows = 1)
prom365_mer
prom365_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 38, labels = "Máx. = 40", cex=0.65),
        rows = 1)
prom365_mer
prom365_mer<-trellis.last.object() +
  layer(panel.abline(h=25, lty = 5),rows = 2)
prom365_mer
prom365_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 23.5, labels = "Máx. = 25", cex=0.65),
        rows = 2)
prom365_mer
pdf(file="prom365_mer.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(prom365_mer)
dev.off()

#Gráficos de valores promedio móvil
promroll_mer<-timePlot(mer_com, pollutant = c("so2_rolling",
   "co_rolling","o3_rolling"),
                       avg.time="year",
                       statistic="max",
                       cols="greyscale",
                       ylab="ppb - ppm - ppb",
                       y.relation="free",
                       ylim = list(c(50, 200), c(2, 11),c(60,160)),
                       name.pol=c("so2","co","o3"))
promroll_mer<-trellis.last.object() +
```

```
  layer(panel.abline(h=200, lty = 5),rows = 1)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 190, labels = "Máx. = 200", cex=0.65),
        rows = 1)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(ltext(x = x, y = 54, labels = mr_so2, cex=0.65), rows = 1)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 70,
              labels = label, cex=0.65), rows = 1)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(panel.abline(h=11, lty = 5),rows = 2)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 10.5, labels = "Máx. = 11", cex=0.65),
        rows = 2)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(ltext(x = x, y = 2.5, labels = mr_co, cex=0.65), rows = 2)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 3.5,
              labels = label, cex=0.65), rows = 2)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(panel.abline(h=70, lty = 5),rows = 3)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 62, labels = "Máx. = 70", cex=0.65),
        rows = 3)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(ltext(x = x1, y = 150, labels = mr_o3, cex=0.65), rows = 3)
promroll_mer
promroll_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 160, labels = label, cex=0.65), rows = 3)
promroll_mer
```

```
pdf(file="rolling_mer.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(promroll_mer)
dev.off()

#Gráficos de valores horarios
promh_mer<-timePlot(mer_com, pollutant = c("no2","o3"),
                    avg.time="year",
                    statistic="max",
                    cols="greyscale",
                    ylab="ppb - ppb",
                    y.relation="free",
                    ylim = list(c(100, 210),c(90,200)))
promh_mer<-trellis.last.object() +
  layer(panel.abline(h=210, lty = 5),rows = 1)
promh_mer
promh_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 205, labels = "Máx. = 210", cex=0.65),
        rows = 1)
promh_mer
promh_mer<-trellis.last.object() +
  layer(ltext(x = x, y = 105, labels = mh_no2, cex=0.65), rows = 1)
promh_mer
promh_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 115, labels = label, cex=0.65), rows = 1)
promh_mer
promh_mer<-trellis.last.object() +
  layer(panel.abline(h=95, lty = 5),rows = 2)
promh_mer
promh_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 90, labels = "Máx. = 95", cex=0.65),
        rows = 2)
promh_mer
promh_mer<-trellis.last.object() +
  layer(ltext(x = x1, y = 130, labels = mh_o3, cex=0.65), rows = 2)
promh_mer
promh_mer<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 140, labels = label, cex=0.65), rows = 2)
promh_mer
pdf(file="prom_mer.pdf",height=4,family="Times",
```

```
pointsize=10,colormodel="grey")
print(promh_mer)
dev.off()
```

PEDREGAL

```
#Cargar workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar datos
load("tablas_procesadas.RData")

#Cargar paquete
library("openair")
library(latticeExtra)

#Obtener y guardar el número de días que se excedió
#el valor de referencia

pd_pm10<-group_by(ped_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
75)), var = as.name("pm10")))
pd_pm10<-as.character(as.vector(pd_pm10$days))

pd_so2<-group_by(ped_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
110)), var = as.name("so2")))
pd_so2<-as.character(as.vector(pd_so2$days))

pr_so2<-group_by(ped_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
200)), var = as.name("so2_rolling")))
pr_so2<-as.character(as.vector(pr_so2$days))

pr_co<-group_by(ped_com, year) %>% do(timeAverage(.,
```

```
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
11)), var = as.name("co_rolling")))
pr_co<-as.character(as.vector(pr_co$days))

pr_o3<-group_by(ped_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
70)), var = as.name("o3_rolling")))
pr_o3<-as.character(as.vector(pr_o3$days))

ph_no2<-group_by(ped_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
210)), var = as.name("no2")))
ph_no2<-as.character(as.vector(ph_no2$days))

ph_o3<-group_by(ped_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
95)), var = as.name("o3")))
ph_o3<-as.character(as.vector(ph_o3$days))

#Lugar en el gráfico en donde se colocarán los valores externos
x<-c(as.POSIXct("2005-01-01"),as.POSIXct("2006-01-01"),
     as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
     as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
     as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
     as.POSIXct("2013-01-01"),as.POSIXct("2014-01-01"))
x1<-c(as.POSIXct("2005-03-01"),as.POSIXct("2006-01-01"),
      as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
      as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
      as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
      as.POSIXct("2013-01-01"),as.POSIXct("2014-01-01"))

xlabel<-as.POSIXct("2005-07-01")
label<-"#Veces > max."

#Gráficos de valores diarios)
prom24_ped<-timePlot(ped_d, pollutant = c("pm10","so2"),avg.time="year",
```

```
                    statistic="max",
                    cols="greyscale",
                    ylab = expression("ppb - " * mu * "g m" ^-3),
                    y.relation="free",
                    ylim = list(c(70, 138), c(0, 120)))
prom24_ped<-trellis.last.object() +
  layer(panel.abline(h=75, lty = 5),rows = 1)
prom24_ped
prom24_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 73, labels = "Max. = 75", cex=0.65),
        rows = 1)
prom24_ped
prom24_ped<-trellis.last.object() +
  layer(ltext(x = x, y = 135, labels = pd_pm10, cex=0.65), rows = 1)
prom24_ped
prom24_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 139,
              labels = label, cex=0.65), rows = 1)
prom24_ped
prom24_ped<-trellis.last.object() +
  layer(panel.abline(h=110, lty = 5),rows = 2)
prom24_ped
prom24_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 105, labels = "Max. = 110", cex=0.65),
        rows = 2)
prom24_ped
prom24_ped<-trellis.last.object() +
  layer(ltext(x = x, y = 10, labels = pd_so2, cex=0.65), rows = 2)
prom24_ped
prom24_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 18, labels = label, cex=0.65), rows = 2)
prom24_ped
pdf(file="prom24_ped.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(prom24_ped)
dev.off()

#Gráficos de valores anuales
prom365_ped<-timePlot(ped_y, pollutant = c("pm10","so2"),cols="greyscale",
                      ylab = expression("ppb - " * mu * "g m" ^-3),
```

```
                      y.relation="free",
                      ylim = list(c(30, 45), c(2, 25)))
prom365_ped<-trellis.last.object() +
  layer(panel.abline(h=40, lty = 5),rows = 1)
prom365_ped
prom365_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 41, labels = "Max. = 40", cex=0.65),
        rows = 1)
prom365_ped
prom365_ped<-trellis.last.object() +
  layer(panel.abline(h=25, lty = 5),rows = 2)
prom365_ped
prom365_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 24, labels = "Max. = 25", cex=0.65),
        rows = 2)
prom365_ped
pdf(file="prom365_ped.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(prom365_ped)
dev.off()

#Gráficos de valores promedio móvil
promroll_ped<-timePlot(ped_com, pollutant = c("so2_rolling",
   "co_rolling","o3_rolling"),
                       avg.time="year",
                       statistic="max",
                       cols="greyscale",
                       ylab="ppb - ppm - ppb",
                       y.relation="free",
                       ylim = list(c(20, 200), c(0, 11),c(70,160)),
                       name.pol=c("so2","co","o3"))
promroll_ped<-trellis.last.object() +
  layer(panel.abline(h=200, lty = 5),rows = 1)
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 190, labels = "Max. = 200", cex=0.65),
        rows = 1)
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(ltext(x = x, y = 30, labels = pr_so2, cex=0.65), rows = 1)
```

```
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 48,
              labels = label, cex=0.65), rows = 1)
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(panel.abline(h=11, lty = 5),rows = 2)
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 10.3, labels = "Max. = 11", cex=0.65),
        rows = 2)
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(ltext(x = x, y = 0.5, labels = pr_co, cex=0.65), rows = 2)
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 1.5,
              labels = label, cex=0.65), rows = 2)
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(panel.abline(h=70, lty = 5),rows = 3)
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 78, labels = "Max. = 70", cex=0.65),
        rows = 3)
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(ltext(x = x1, y = 100, labels = pr_o3, cex=0.65), rows = 3)
promroll_ped
promroll_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 110, labels = label, cex=0.65), rows = 3)
promroll_ped
pdf(file="rolling_ped.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(promroll_ped)
dev.off()

#Gráficos de valores horarios
promh_ped<-timePlot(ped_com, pollutant = c("no2","o3"),
                    avg.time="year",
```

```
                    statistic="max",
                    cols="greyscale",
                    ylab="ppb - ppb",
                    y.relation="free",
                    ylim = list(c(70, 210),c(90,250)))
promh_ped<-trellis.last.object() +
  layer(panel.abline(h=210, lty = 5),rows = 1)
promh_ped
promh_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 200, labels = "Max. = 210", cex=0.65),
        rows = 1)
promh_ped
promh_ped<-trellis.last.object() +
  layer(ltext(x = x, y = 72, labels = ph_no2, cex=0.65), rows = 1)
promh_ped
promh_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 82, labels = label, cex=0.65), rows = 1)
promh_ped
promh_ped<-trellis.last.object() +
  layer(panel.abline(h=95, lty = 5),rows = 2)
promh_ped
promh_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 90, labels = "Max. = 95", cex=0.65),
        rows = 2)
promh_ped
promh_ped<-trellis.last.object() +
  layer(ltext(x = x1, y = 140, labels = ph_o3, cex=0.65), rows = 2)
promh_ped
promh_ped<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 152, labels = label, cex=0.65), rows = 2)
promh_ped
pdf(file="prom_ped.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(promh_ped)
dev.off()
```

SAN AGUSTÍN

```
#Cargar workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar datos
load("tablas_procesadas.RData")

#Cargar paquete
library("openair")
library(latticeExtra)

#Obtener y guardar el número de días que se excedió
#el valor de referencia

sd_pm10<-group_by(sag_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
75)), var = as.name("pm10")))
sd_pm10<-as.character(as.vector(sd_pm10$days))

sd_so2<-group_by(sag_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
110)), var = as.name("so2")))
sd_so2<-as.character(as.vector(sd_so2$days))

sr_so2<-group_by(sag_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
200)), var = as.name("so2_rolling")))
sr_so2<-as.character(as.vector(sr_so2$days))

sr_co<-group_by(sag_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
11)), var = as.name("co_rolling")))
sr_co<-as.character(as.vector(sr_co$days))

sr_o3<-group_by(sag_com, year) %>% do(timeAverage(.,
```

```
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
70)), var = as.name("o3_rolling")))
sr_o3<-as.character(as.vector(sr_o3$days))

sh_no2<-group_by(sag_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
210)), var = as.name("no2")))
sh_no2<-as.character(as.vector(sh_no2$days))

sh_o3<-group_by(sag_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
95)), var = as.name("o3")))
sh_o3<-as.character(as.vector(sh_o3$days))

#Lugar en el gráfico en donde se colocarán los valores externos
x<-c(as.POSIXct("2005-01-01"),as.POSIXct("2006-01-01"),
     as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
     as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
     as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
     as.POSIXct("2013-01-01"),as.POSIXct("2014-01-01"))
x1<-c(as.POSIXct("2005-03-01"),as.POSIXct("2006-01-01"),
      as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
      as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
      as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
      as.POSIXct("2013-01-01"),as.POSIXct("2014-01-01"))

xlabel<-as.POSIXct("2005-07-01")
label<-"#Veces > máx."

#Gráficos de valores diarios
prom24_sag<-timePlot(sag_d, pollutant = c("pm10","so2"),avg.time="year",
                     statistic="max",
                     cols="greyscale",
                     ylab = expression("ppb - " * mu * "g m" ^-3),
                     y.relation="free",
                     ylim = list(c(60, 320), c(10, 110)))
prom24_sag<-trellis.last.object() +
```

```
  layer(panel.abline(h=75, lty = 5),rows = 1)
prom24_sag
prom24_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 65, labels = "Máx. = 75", cex=0.65),
        rows = 1)
prom24_sag
prom24_sag<-trellis.last.object() +
  layer(ltext(x = x, y = 300, labels = sd_pm10, cex=0.65), rows = 1)
prom24_sag
prom24_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 320,
              labels = label, cex=0.65), rows = 1)
prom24_sag
prom24_sag<-trellis.last.object() +
  layer(panel.abline(h=110, lty = 5),rows = 2)
prom24_sag
prom24_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 105, labels = "Máx. = 110", cex=0.65),
        rows = 2)
prom24_sag
prom24_sag<-trellis.last.object() +
  layer(ltext(x = x, y = 18, labels = sd_so2, cex=0.65), rows = 2)
prom24_sag
prom24_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 25, labels = label, cex=0.65), rows = 2)
prom24_sag
pdf(file="prom24_sag.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(prom24_sag)
dev.off()

#Gráficos de valores anuales
prom365_sag<-timePlot(sag_y, pollutant = c("pm10","so2"),cols="greyscale",
                      ylab = expression("ppb - " * mu * "g m" ^-3),
                      y.relation="free",
                      ylim = list(c(35, 80), c(3, 26)))
prom365_sag<-trellis.last.object() +
  layer(panel.abline(h=40, lty = 5),rows = 1)
prom365_sag
prom365_sag<-trellis.last.object() +
```

```
  layer(ltext(x = xlabel, y = 38, labels = "Máx. = 40", cex=0.65),
        rows = 1)
prom365_sag
prom365_sag<-trellis.last.object() +
  layer(panel.abline(h=25, lty = 5),rows = 2)
prom365_sag
prom365_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 24, labels = "Máx. = 25", cex=0.65),
        rows = 2)
prom365_sag
pdf(file="prom365_sag.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(prom365_sag)
dev.off()

#Gráficos de valores promedio móvil
promroll_sag<-timePlot(sag_com, pollutant = c("so2_rolling",
   "co_rolling","o3_rolling"),
                       avg.time="year",
                       statistic="max",
                       cols="greyscale",
                       ylab="ppb - ppm - ppb",
                       y.relation="free",
                       ylim = list(c(50, 200), c(2, 11),c(60,160)),
                       name.pol=c("so2","co","o3"))
promroll_sag<-trellis.last.object() +
  layer(panel.abline(h=200, lty = 5),rows = 1)
promroll_sag
promroll_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 190, labels = "Máx. = 200", cex=0.65),
        rows = 1)
promroll_sag
promroll_sag<-trellis.last.object() +
  layer(ltext(x = x, y = 52, labels = sr_so2, cex=0.65), rows = 1)
promroll_sag
promroll_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 68,
              labels = label, cex=0.65), rows = 1)
promroll_sag
promroll_sag<-trellis.last.object() +
```

```
  layer(panel.abline(h=11, lty = 5),rows = 2)
promroll_sag
promroll_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 10.5, labels = "Máx. = 11", cex=0.65),
        rows = 2)
promroll_sag
promroll_sag<-trellis.last.object() +
  layer(ltext(x = x, y = 6, labels = sr_co, cex=0.65), rows = 2)
promroll_sag
promroll_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 7,
              labels = label, cex=0.65), rows = 2)
promroll_sag
promroll_sag<-trellis.last.object() +
  layer(panel.abline(h=70, lty = 5),rows = 3)
promroll_sag
promroll_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 62, labels = "Máx. = 70", cex=0.65),
        rows = 3)
promroll_sag
promroll_sag<-trellis.last.object() +
  layer(ltext(x = x1, y = 150, labels = sr_o3, cex=0.65), rows = 3)
promroll_sag
promroll_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 161, labels = label, cex=0.65), rows = 3)
promroll_sag
pdf(file="rolling_sag.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(promroll_sag)
dev.off()

#Gráficos de valores horarios
promh_sag<-timePlot(sag_com, pollutant = c("no2","o3"),
                    avg.time="year",
                    statistic="max",
                    cols="greyscale",
                    ylab="ppb - ppb",
                    y.relation="free",
                    ylim = list(c(90, 210),c(90,200)))
promh_sag<-trellis.last.object() +
```

```
  layer(panel.abline(h=210, lty = 5),rows = 1)
promh_sag
promh_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 205, labels = "Máx. = 210", cex=0.65),
        rows = 1)
promh_sag
promh_sag<-trellis.last.object() +
  layer(ltext(x = x, y = 155, labels = sh_no2, cex=0.65), rows = 1)
promh_sag
promh_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 165, labels = label, cex=0.65), rows = 1)
promh_sag
promh_sag<-trellis.last.object() +
  layer(panel.abline(h=95, lty = 5),rows = 2)
promh_sag
promh_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 90, labels = "Máx. = 95", cex=0.65),
        rows = 2)
promh_sag
promh_sag<-trellis.last.object() +
  layer(ltext(x = x1, y = 190, labels = sh_o3, cex=0.65), rows = 2)
promh_sag
promh_sag<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 200, labels = label, cex=0.65), rows = 2)
promh_sag
pdf(file="prom_sag.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(promh_sag)
dev.off()
```

TLÁHUAC

```
#Cargar workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar datos
load("tablas_procesadas.RData")
```

```
#Cargar paquete
library("openair")
library(latticeExtra)

#Obtener y guardar el número de días que se excedió
#el valor de referencia

td_pm10<-group_by(tah_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
75)), var = as.name("pm10")))
td_pm10<-as.character(as.vector(td_pm10$days))

td_so2<-group_by(tah_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
110)), var = as.name("so2")))
td_so2<-as.character(as.vector(td_so2$days))

tr_so2<-group_by(tah_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
200)), var = as.name("so2_rolling")))
tr_so2<-as.character(as.vector(tr_so2$days))

tr_co<-group_by(tah_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
11)), var = as.name("co_rolling")))
tr_co<-as.character(as.vector(tr_co$days))

tr_o3<-group_by(tah_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
70)), var = as.name("o3_rolling")))
tr_o3<-as.character(as.vector(tr_o3$days))

th_no2<-group_by(tah_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
```

```
210)), var = as.name("no2")))
th_no2<-as.character(as.vector(th_no2$days))

th_o3<-group_by(tah_com, year) %>% do(timeAverage(.,
avg.time = "hour", statistic = "max", 0)) %>%
summarise_(days = interp(~length(which(var >
95)), var = as.name("o3")))
th_o3<-as.character(as.vector(th_o3$days))

#Lugar en el gráfico en donde se colocarán los valores externos
x<-c(as.POSIXct("2005-01-01"),as.POSIXct("2006-01-01"),
     as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
     as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
     as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
     as.POSIXct("2013-01-01"),as.POSIXct("2014-01-01"))
x1<-c(as.POSIXct("2005-03-01"),as.POSIXct("2006-01-01"),
      as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
      as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
      as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
      as.POSIXct("2013-01-01"),as.POSIXct("2014-01-01"))

xlabel<-as.POSIXct("2005-07-01")
label<-"#Veces > máx."

#Gráficos de valores diarios
prom24_tah<-timePlot(tah_d, pollutant = c("pm10","so2"),avg.time="year",
                     statistic="max",
                     cols="greyscale",
                     ylab = expression("ppb - " * mu * "g m" ^-3),
                     y.relation="free",
                     ylim = list(c(70, 160), c(0, 120)))
prom24_tah<-trellis.last.object() +
  layer(panel.abline(h=75, lty = 5),rows = 1)
prom24_tah
prom24_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 80, labels = "Máx. = 75", cex=0.65),
        rows = 1)
prom24_tah
prom24_tah<-trellis.last.object() +
  layer(ltext(x = x, y = 153, labels = td_pm10, cex=0.65), rows = 1)
```

```
prom24_tah
prom24_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 160,
              labels = label, cex=0.65), rows = 1)
prom24_tah
prom24_tah<-trellis.last.object() +
  layer(panel.abline(h=110, lty = 5),rows = 2)
prom24_tah
prom24_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 105, labels = "Máx. = 110", cex=0.65),
        rows = 2)
prom24_tah
prom24_tah<-trellis.last.object() +
  layer(ltext(x = x, y = 1, labels = td_so2, cex=0.65), rows = 2)
prom24_tah
prom24_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 10, labels = label, cex=0.65), rows = 2)
prom24_tah
pdf(file="prom24_tah.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(prom24_tah)
dev.off()

#Gráficos de valores anuales
prom365_tah<-timePlot(tah_y, pollutant = c("pm10","so2"),cols="greyscale",
                      ylab = expression("ppb - " * mu * "g m" ^-3),
                      y.relation="free",
                      ylim = list(c(38, 52), c(0, 26)))
prom365_tah<-trellis.last.object() +
  layer(panel.abline(h=40, lty = 5),rows = 1)
prom365_tah
prom365_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 39, labels = "Máx. = 40", cex=0.65),
        rows = 1)
prom365_tah
prom365_tah<-trellis.last.object() +
  layer(panel.abline(h=25, lty = 5),rows = 2)
prom365_tah
prom365_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 23.5, labels = "Máx. = 25", cex=0.65),
```

```
        rows = 2)
prom365_tah
pdf(file="prom365_tah.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(prom365_tah)
dev.off()

#Gráficos de valores promedio móvil
promroll_tah<-timePlot(tah_com, pollutant = c("so2_rolling",
   "co_rolling","o3_rolling"),
                       avg.time="year",
                       statistic="max",
                       cols="greyscale",
                       ylab="ppb - ppm - ppb",
                       y.relation="free",
                       ylim = list(c(20, 200), c(2, 11),c(60,160)),
                       name.pol=c("so2","co","o3"))
promroll_tah<-trellis.last.object() +
  layer(panel.abline(h=200, lty = 5),rows = 1)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 190, labels = "Máx. = 200", cex=0.65),
        rows = 1)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(ltext(x = x, y = 75, labels = tr_so2, cex=0.65), rows = 1)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 95,
              labels = label, cex=0.65), rows = 1)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(panel.abline(h=11, lty = 5),rows = 2)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 10.5, labels = "Máx. = 11", cex=0.65),
        rows = 2)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(ltext(x = x[-c(1:6)], y = 6, labels = tr_co[-c(1:6)],
```

```
  cex=0.65), rows = 2)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(ltext(x = x[7]+13042800, y = 7.5,
              labels = label, cex=0.65), rows = 2)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(panel.abline(h=70, lty = 5),rows = 3)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 63, labels = "Máx. = 70", cex=0.65),
        rows = 3)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(ltext(x = x1, y = 150, labels = tr_o3, cex=0.65), rows = 3)
promroll_tah
promroll_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 161, labels = label, cex=0.65), rows = 3)
promroll_tah
pdf(file="rolling_tah.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(promroll_tah)
dev.off()

#Gráficos de valores horarios
promh_tah<-timePlot(tah_com, pollutant = c("no2","o3"),
                    avg.time="year",
                    statistic="max",
                    cols="greyscale",
                    ylab="ppb - ppb",
                    y.relation="free",
                    ylim = list(c(60, 220),c(90,200)))
promh_tah<-trellis.last.object() +
  layer(panel.abline(h=210, lty = 5),rows = 1)
promh_tah
promh_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 200, labels = "Máx. = 210", cex=0.65),
        rows = 1)
promh_tah
promh_tah<-trellis.last.object() +
```

```
  layer(ltext(x = x[7:10], y = 105, labels = th_no2[7:10],
  cex=0.65), rows = 1)
promh_tah
promh_tah<-trellis.last.object() +
  layer(ltext(x = x[7]+13042800, y = 120, labels = label,
  cex=0.65), rows = 1)
promh_tah
promh_tah<-trellis.last.object() +
  layer(panel.abline(h=95, lty = 5),rows = 2)
promh_tah
promh_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 90, labels = "Máx. = 95", cex=0.65),
        rows = 2)
promh_tah
promh_tah<-trellis.last.object() +
  layer(ltext(x = x1, y = 190, labels = th_o3,
  cex=0.65), rows = 2)
promh_tah
promh_tah<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 200, labels = label,
  cex=0.65), rows = 2)
promh_tah
pdf(file="prom_tah.pdf",height=4,family="Times",
pointsize=10,colormodel="grey")
print(promh_tah)
dev.off()b
```

PM_{10}

```
#Cargar workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar datos
load("tablas_procesadas.RData")

#Cargar paquete
library("openair")
library(latticeExtra)
```

```
#Obtener y guardar el número de días que se excedió
#el valor de referencia

fd_pm10<-group_by(fac_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
75)), var = as.name("pm10")))
fd_pm10<-as.character(as.vector(fd_pm10$days))

md_pm10<-group_by(mer_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
75)), var = as.name("pm10")))
md_pm10<-as.character(as.vector(md_pm10$days))

sd_pm10<-group_by(sag_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
75)), var = as.name("pm10")))
sd_pm10<-as.character(as.vector(sd_pm10$days))

pd_pm10<-group_by(ped_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
75)), var = as.name("pm10")))
pd_pm10<-as.character(as.vector(pd_pm10$days))

td_pm10<-group_by(tah_com, year) %>% do(timeAverage(.,
avg.time = "day", statistic = "mean", 75)) %>%
summarise_(days = interp(~length(which(var >
75)), var = as.name("pm10")))
td_pm10<-as.character(as.vector(td_pm10$days))

#Lugar en el gráfico en donde se colocarán los valores externos
x<-c(as.POSIXct("2005-01-01"),as.POSIXct("2006-01-01"),
     as.POSIXct("2007-01-01"),as.POSIXct("2008-01-01"),
     as.POSIXct("2009-01-01"),as.POSIXct("2010-01-01"),
     as.POSIXct("2011-01-01"),as.POSIXct("2012-01-01"),
     as.POSIXct("2013-01-01"),as.POSIXct("2014-01-01"))
```

```
xlabel<-as.POSIXct("2005-07-01")
label<-"#Veces > máx."

#Gráficos de valores diarios
prom24_pm10<-timePlot(pmd.10,
                      pollutant = c("pm10.FAC","pm10.MER","pm10.PED",
                                    "pm10.SAG","pm10.TAH"),
                      avg.time="year",
                      statistic="max",
                      cols="greyscale",ylab=expression( mu * "g m" ^-3),
                      y.relation="free",
                      ylim = list(c(70, 150), c(70, 180)
                                  , c(70, 150), c(70, 320)
                                  , c(70, 150)),
                      key=FALSE)
prom24_pm10<-trellis.last.object() +
  layer(panel.abline(h=75, lty = 5),rows = 1)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(panel.abline(h=75, lty = 5),rows = 2)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(panel.abline(h=75, lty = 5),rows = 3)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(panel.abline(h=75, lty = 5),rows = 4)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(panel.abline(h=75, lty = 5),rows = 5)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 88, labels = "Máx. = 75", cex=0.65),
        rows = 1)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 88, labels = "Máx. = 75", cex=0.65),
        rows = 2)
prom24_pm10
prom24_pm10<-trellis.last.object() +
```

```
  layer(ltext(x = xlabel, y = 88, labels = "Máx. = 75", cex=0.65),
        rows = 3)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 110, labels = "Máx. = 75", cex=0.65),
        rows = 4)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 88, labels = "Máx. = 75", cex=0.65),
        rows = 5)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = x, y = 148, labels = fd_pm10, cex=0.65), rows = 1)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 133,
              labels = label, cex=0.65), rows = 1)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = x, y = 178, labels = md_pm10, cex=0.65), rows = 2)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 158,
              labels = label, cex=0.65), rows = 2)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = x, y = 148, labels = pd_pm10, cex=0.65), rows = 3)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 135,
              labels = label, cex=0.65), rows = 3)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = x, y = 318, labels = sd_pm10, cex=0.65), rows = 4)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 280,
              labels = label, cex=0.65), rows = 4)
prom24_pm10
prom24_pm10<-trellis.last.object() +
```

```
  layer(ltext(x = x, y = 148, labels = td_pm10, cex=0.65), rows = 5)
prom24_pm10
prom24_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 135,
              labels = label, cex=0.65), rows = 5)
prom24_pm10
pdf(file="prom24_pm10.pdf",family="Times",
height=5,width=7,pointsize=10,colormodel="grey")
print(prom24_pm10)
dev.off()

#Gráficos de valores anuales
prom365_pm10<-timePlot(pmy.10,
                       pollutant = c("pm10.FAC","pm10.MER","pm10.PED",
                                     "pm10.SAG","pm10.TAH"),
                       cols="greyscale",
                       ylab = expression("ppb - " * mu * "g m" ^-3),
                       y.relation="free",
                       ylim = list(c(38, 50), c(40, 65),c(35, 45),
                                   c(30, 80),c(35, 50)),
                       key=FALSE)
prom365_pm10<-trellis.last.object() +
  layer(panel.abline(h=40, lty = 5), rows = 1)
prom365_pm10
prom365_pm10<-trellis.last.object() +
  layer(panel.abline(h=40, lty = 5), rows = 2)
prom365_pm10
prom365_pm10<-trellis.last.object() +
  layer(panel.abline(h=40, lty = 5), rows = 3)
prom365_pm10
prom365_pm10<-trellis.last.object() +
  layer(panel.abline(h=40, lty = 5), rows = 4)
prom365_pm10
prom365_pm10<-trellis.last.object() +
  layer(panel.abline(h=40, lty = 5), rows = 5)
prom365_pm10
prom365_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 42, labels = "Máx. = 40", cex=0.65),
        rows = 1)
prom365_pm10
```

```
prom365_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 45, labels = "Máx. = 40", cex=0.65),
        rows = 2)
prom365_pm10
prom365_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 42, labels = "Máx. = 40", cex=0.65),
        rows = 3)
prom365_pm10
prom365_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 32, labels = "Máx. = 40", cex=0.65),
        rows = 4)
prom365_pm10
prom365_pm10<-trellis.last.object() +
  layer(ltext(x = xlabel, y = 43, labels = "Máx. = 40", cex=0.65),
        rows = 5)
prom365_pm10
pdf(file="prom365_pm10.pdf",family="Times",
height=5,width=7,pointsize=10,colormodel="grey")
print(prom365_pm10)
dev.off()
```

[

Selección de modelo RNA]Códigos para la selección de modelo RNA

selectNNET

```
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]
```

```
#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Cargar paquete de RNA
library(tsDyn)

#Utilizar la función para selección automática
#que tsDyn ofrece, de 1 a 8 lags (neuronas en capa en entrada)
#y de 1 a 17 neuronas en capa oculta
selectNNET(pm10.n, m=1, size=1:17)
selectNNET(pm10.n, m=2, size=1:17)
selectNNET(pm10.n, m=3, size=1:17)
selectNNET(pm10.n, m=4, size=1:17)
selectNNET(pm10.n, m=5, size=1:17)
selectNNET(pm10.n, m=6, size=1:17)
selectNNET(pm10.n, m=7, size=1:17)
selectNNET(pm10.n, m=8, size=1:17)
```

RNA 1-3-1

```
#Modelo RNA 1-3-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
```

```
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n <- nnetTs(pm10.n, m=1, size=3,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,1)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)
```

```
#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:204){
test.n[i]<-(predict(mod.nnet.n,newdata=pm10.n.test[i],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,1)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t1<-summary(train.n.lm)$r.squared
n1<-summary(test.n.lm)$r.squared
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
save.image("SELECCION_RNA.RData")
```

RNA 2-14-1

```
#Modelo RNA 2-14-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
```

```
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n2 <- nnetTs(pm10.n, m=2, size=14,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n2$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,2)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:203){
test.n[i]<-(predict(mod.nnet.n2,newdata=pm10.n.test[i:(i+1)],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,2)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t2<-summary(train.n.lm)$r.squared
```

```
n2<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
save.image("SELECCION_RNA.RData")
```

RNA 3-17-1

```
#Modelo RNA 3-17-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
```

```
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n3 <- nnetTs(pm10.n, m=3, size=17,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n3$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,3)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:202){
```

```
test.n[i]<-(predict(mod.nnet.n3,newdata=pm10.n.test[i:(i+2)],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,3)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t3<-summary(train.n.lm)$r.squared
n3<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
save.image("SELECCION_RNA.RData")
```

RNA 4-16-1

```
#Modelo RNA 4-16-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))
```

```
#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n4 <- nnetTs(pm10.n, m=4, size=16,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n4$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,4)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
```

```
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:201){
test.n[i]<-(predict(mod.nnet.n4,newdata=pm10.n.test[i:(i+3)],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,4)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t4<-summary(train.n.lm)$r.squared
n4<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
save.image("SELECCION_RNA.RData")
```

RNA 5-17-1

```
#Modelo RNA 5-17-1
#Cargar Workspace
```

```
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n5 <- nnetTs(pm10.n, m=5, size=17,step=1)
```

```
#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n5$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,5)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:200){
test.n[i]<-(predict(mod.nnet.n5,newdata=pm10.n.test[i:(i+4)],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,5)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t5<-summary(train.n.lm)$r.squared
n5<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
```

```
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
save.image("SELECCION_RNA.RData")
```

RNA 6-15-1

```
#Modelo RNA 6-15-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]
```

```
#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n6 <- nnetTs(pm10.n, m=6, size=15,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n6$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,6)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:199){
test.n[i]<-(predict(mod.nnet.n6,newdata=pm10.n.test[i:(i+5)],n.ahead=1)*
              sd.n)+mean.n
}
```

```
#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,6)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t6<-summary(train.n.lm)$r.squared
n6<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
save.image("SELECCION_RNA.RData")
```

RNA 7-17-1

```
#Modelo RNA 7-17-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
```

```
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n7 <- nnetTs(pm10.n, m=7, size=17,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n7$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,7)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
```

```
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:198){
test.n[i]<-(predict(mod.nnet.n7,newdata=pm10.n.test[i:(i+6)],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,7)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t7<-summary(train.n.lm)$r.squared
n7<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
save.image("SELECCION_RNA.RData")
```

RNA 8-17-1

```
#Modelo RNA 8-17-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")
```

```
#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n8 <- nnetTs(pm10.n, m=8, size=17,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n8$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
```

```
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,8)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:197){
test.n[i]<-(predict(mod.nnet.n8,newdata=pm10.n.test[i:(i+7)],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,8)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t8<-summary(train.n.lm)$r.squared
n8<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
```

```
save.image("SELECCION_RNA.RData")
```

RNA 8-1-1

```
#Modelo RNA 8-1-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)
```

```
#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n9 <- nnetTs(pm10.n, m=8, size=1,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n9$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,8)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:197){
test.n[i]<-(predict(mod.nnet.n9,newdata=pm10.n.test[i:(i+7)],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,8)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)
```

```
#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t9<-summary(train.n.lm)$r.squared
n9<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
save.image("SELECCION_RNA.RData")
```

RNA 8-8-1

```
#Modelo RNA 8-8-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
```

```
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n10 <- nnetTs(pm10.n, m=8, size=8,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n10$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,8)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n
```

```
#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:197){
test.n[i]<-(predict(mod.nnet.n10,newdata=pm10.n.test[i:(i+7)],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,8)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t10<-summary(train.n.lm)$r.squared
n10<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
save.image("SELECCION_RNA.RData")
```

RNA 8-15-1

```
#Modelo RNA 8-15-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
```

```
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n11 <- nnetTs(pm10.n, m=8, size=15,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n11$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,8)),fit.n)

#Agregarlos a la tabla de origen
```

```
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:197){
test.n[i]<-(predict(mod.nnet.n11,newdata=pm10.n.test[i:(i+7)],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,8)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t11<-summary(train.n.lm)$r.squared
n11<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
save.image("SELECCION_RNA.RData")
```

RNA 8-16-1

```
#Modelo RNA 8-16-1
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Carga de todas las tablas
load("tablas_procesadas.RData")

#Eliminar tablas no necesarias
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets y se utiliza
#el mismo parámetro para el modelo RNA
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Seleccionar datos para conjunto de entrenamiento
wt_pm10.n<-wt_pm10.com[1:307,2]

#Escalar valores
pm10.n<-scale(wt_pm10.n)

#Seleccionar datos para conjunto de prueba
wt_pm10.n.test<-wt_pm10.com[308:512,2]

#Cargar paquete de RNA
```

```
library(tsDyn)

#"Entrenar" la RNA
mod.nnet.n12 <- nnetTs(pm10.n, m=8, size=16,step=1)

#Extraer los valores de ajuste y quitar el escalamiento
fit.n<-mod.nnet.n12$fitted.values
fit.n<-(fit.n*sd(wt_pm10.n))+mean(wt_pm10.n)
colnames(fit.n)[1]<-"fit.n"
fit.n<-rbind(data.frame(fit.n=rep(NA,8)),fit.n)

#Agregarlos a la tabla de origen
wt_pm10<-cbind(wt_pm10,fit.n)

#Obtener el valor de R cuadrada
train.n.lm<-lm(pm10 ~ fit.n, data=wt_pm10)
summary(train.n.lm)$r.squared

#Guardar valores de escalamiento en entrenamiento
sd.n<-sd(wt_pm10.n)
mean.n<-mean(wt_pm10.n)

#Escalar la serie de prueba
pm10.n.test<-(wt_pm10.n.test-mean.n)/sd.n

#Aplicar el modelo a los datos de prueba
test.n<-0
for(i in 1:197){
test.n[i]<-(predict(mod.nnet.n12,newdata=pm10.n.test[i:(i+7)],n.ahead=1)*
              sd.n)+mean.n
}

#Agregar los resultados a la tabla origen de prueba
test.n<-data.frame(test.n=test.n)
test.n<-rbind(data.frame(test.n=rep(NA,8)),test.n)
wt_pm10.test<-cbind(wt_pm10.test,test.n)

#Obtener R cuadrada
test.n.lm<-lm(pm10 ~ test.n, data=wt_pm10.test)
t12<-summary(train.n.lm)$r.squared
```

```
n12<-summary(test.n.lm)$r.squared
load("SELECCION_RNA.RData")
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))

testing<-rbind(data.frame(testing=n1),
               data.frame(testing=n2),
               data.frame(testing=n3),
               data.frame(testing=n4),
               data.frame(testing=n5),
               data.frame(testing=n6),
               data.frame(testing=n7),
               data.frame(testing=n8),
               data.frame(testing=n9),
               data.frame(testing=n10),
               data.frame(testing=n11),
               data.frame(testing=n12))
training<-rbind(data.frame(training=t1),
               data.frame(training=t2),
               data.frame(training=t3),
               data.frame(training=t4),
               data.frame(training=t5),
               data.frame(training=t6),
               data.frame(training=t7),
               data.frame(training=t8),
               data.frame(training=t9),
               data.frame(training=t10),
               data.frame(training=t11),
               data.frame(training=t12))
description<-data.frame(description=c("1-3-1","2-14-1","3-17-1",
                                      "4-16-1","5-17-1","6-15-1",
                                      "7-17-1","8-17-1","8-1-1",
                                      "8-8-1","8-15-1","8-16-1"))
rna_selection<-cbind(description,training,testing)
rna_selection$training<-round(rna_selection$training,3)
rna_selection$testing<-round(rna_selection$testing,3)
```

```
library(xtable)
xtable(rna_selection,digits=3)
save.image("SELECCION_RNA.RData")
```

[

Selección de modelo Wavelet+RNA]Códigos para la selección de modelo Wavelet+RNA

selectNNET *Symlet* 9

```
#Symlet 9
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
library(wavethresh)

#Carga de todas las tablas
load("tablas_procesadas.RData")

rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]
```

```
#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Serie de datos para wavelet
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")

#Transformada Wavelet Discreta Symlet 9
pm_wd<-wd(pm,filter.number=9, family="DaubLeAsymm")

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)

#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))

#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)

#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"

#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]

#Escalar la serie
pm10.w<-scale(wt_pm10.w)

#Cargar paquete para RNA
```

```
library(tsDyn)

selectNNET(pm10.w, m=1, size=1:8)
selectNNET(pm10.w, m=2, size=1:8)
selectNNET(pm10.w, m=3, size=1:8)
selectNNET(pm10.w, m=4, size=1:8)
selectNNET(pm10.w, m=5, size=1:8)
selectNNET(pm10.w, m=6, size=1:8)
selectNNET(pm10.w, m=7, size=1:8)
selectNNET(pm10.w, m=8, size=1:8)
```

selectNNET *Symlet* 10

```
#Symlet 10
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
library(wavethresh)

#Carga de todas las tablas
load("tablas_procesadas.RData")

rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]
```

```
#Eliminar temporales
rm(list=c("temp","temp2"))

#Serie de datos para wavelet
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")

#Transformada Wavelet Discreta Symlet 10
pm_wd<-wd(pm,filter.number=10, family="DaubLeAsymm")

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)

#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))

#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)

#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"

#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]

#Escalar la serie
pm10.w<-scale(wt_pm10.w)

#Cargar paquete para RNA
library(tsDyn)

selectNNET(pm10.w, m=1, size=1:8)
```

```
selectNNET(pm10.w, m=2, size=1:8)
selectNNET(pm10.w, m=3, size=1:8)
selectNNET(pm10.w, m=4, size=1:8)
selectNNET(pm10.w, m=5, size=1:8)
selectNNET(pm10.w, m=6, size=1:8)
selectNNET(pm10.w, m=7, size=1:8)
selectNNET(pm10.w, m=8, size=1:8)
```

selectNNET *Daubechies* 10

```
#Daubechies 10
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
library(wavethresh)

#Carga de todas las tabla
load("tablas_procesadas.RData")

rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))
```

```
#Serie de datos para wavelet
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")

#Transformada Wavelet Discreta Daubechies 10
pm_wd<-wd(pm,filter.number=10, family="DaubExPhase")

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)

#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))

#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)

#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"

#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]

#Escalar la serie
pm10.w<-scale(wt_pm10.w)

#Cargar paquete para RNA
library(tsDyn)

selectNNET(pm10.w, m=1, size=1:8)
selectNNET(pm10.w, m=2, size=1:8)
selectNNET(pm10.w, m=3, size=1:8)
selectNNET(pm10.w, m=4, size=1:8)
```

```
selectNNET(pm10.w, m=5, size=1:8)
selectNNET(pm10.w, m=6, size=1:8)
selectNNET(pm10.w, m=7, size=1:8)
selectNNET(pm10.w, m=8, size=1:8)
```

Wavelet+RNA Symlet10+8-8-1

```
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
library(wavethresh)

#Carga de todas las tablas
load("tablas_procesadas.RData")

rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Serie de datos para wavelet
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")
```

```
#Transformada Wavelet Discreta Symlet 10
pm_wd<-wd(pm,filter.number=10, family="DaubLeAsymm")

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)

#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))

#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)

#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"

#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]
wt_pm10<-wt_pm10.com[1:307,]

#Escalar la serie
pm10.w<-scale(wt_pm10.w)

#Obtener los datos de prueba
wt_pm10.w.test<-wt_pm10.com[308:512,3]
wt_pm10.test<-wt_pm10.com[308:512,]

#Cargar paquete para RNA
library(tsDyn)

#Entrenar la RED
mod.nnet.w1 <- nnetTs(pm10.w, m=8, size=8,step=1)
```

```
#obtener los valores ajustados y quitar el escalamiento
fit.w<-mod.nnet.w1$fitted.values
fit.w<-(fit.w*sd(wt_pm10.w))+mean(wt_pm10.w)
colnames(fit.w)[1]<-"fit.w"
fit.w<-rbind(data.frame(fit.w=rep(NA,8)),fit.w)

#Agregar a la base general
wt_pm10<-cbind(wt_pm10,fit.w)

#obtener R cuadrada
train.w.lm<-lm(pm10.wavelet ~ fit.w, data=wt_pm10)
t1<-summary(train.w.lm)$r.squared

#Guardar datos de escalamiento de la serie de entrenamiento
sd.w<-sd(wt_pm10.w)
mean.w<-mean(wt_pm10.w)

#Escalar la serie de prueba
pm10.w.test<-(wt_pm10.w.test-mean.w)/sd.w

#Aplicar el modelo a la serie de prueba
test.w<-0
for(i in 1:197){
  test.w[i]<-(predict(mod.nnet.w1,newdata=pm10.w.test[i:(i+7)],n.ahead=1)*
                sd.w)+mean.w
}

#Agregar los resultados a la serie de prueba
test.w<-data.frame(test.w=test.w)
test.w<-rbind(data.frame(test.w=rep(NA,8)),test.w)
wt_pm10.test<-cbind(wt_pm10.test,test.w)

#Obtener R cuadrada
test.w.lm<-lm(pm10.wavelet ~ test.w, data=wt_pm10.test)
n1<-summary(test.w.lm)$r.squared

#Borrar temporales
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
```

```
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))

#Guardar resultados
save.image("wavelet-selec.RData")
```

Wavelet+RNA Symlet10+8-6-1

```
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
library(wavethresh)

#Carga de todas las tablas
load("tablas_procesadas.RData")

rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Serie de datos para wavelet
```

```
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")

#Transformada Wavelet Discreta Symlet 10
pm_wd<-wd(pm,filter.number=10, family="DaubLeAsymm")

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)

#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))

#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)

#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"

#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]
wt_pm10<-wt_pm10.com[1:307,]

#Escalar la serie
pm10.w<-scale(wt_pm10.w)

#Obtener los datos de prueba
wt_pm10.w.test<-wt_pm10.com[308:512,3]
wt_pm10.test<-wt_pm10.com[308:512,]

#Cargar paquete para RNA
library(tsDyn)

#Entrenar la RED
```

```
mod.nnet.w2 <- nnetTs(pm10.w, m=8, size=6,step=1)

#obtener los valores ajustados y quitar el escalamiento
fit.w<-mod.nnet.w2$fitted.values
fit.w<-(fit.w*sd(wt_pm10.w))+mean(wt_pm10.w)
colnames(fit.w)[1]<-"fit.w"
fit.w<-rbind(data.frame(fit.w=rep(NA,8)),fit.w)

#Agregar a la base general
wt_pm10<-cbind(wt_pm10,fit.w)

#obtener R cuadrada
train.w.lm<-lm(pm10.wavelet ~ fit.w, data=wt_pm10)
t2<-summary(train.w.lm)$r.squared

#Guardar datos de escalamiento de la serie de entrenamiento
sd.w<-sd(wt_pm10.w)
mean.w<-mean(wt_pm10.w)

#Escalar la serie de prueba
pm10.w.test<-(wt_pm10.w.test-mean.w)/sd.w

#Aplicar el modelo a la serie de prueba
test.w<-0
for(i in 1:197){
  test.w[i]<-(predict(mod.nnet.w2,newdata=pm10.w.test[i:(i+7)],n.ahead=1)*
                sd.w)+mean.w
}

#Agregar los resultados a la serie de prueba
test.w<-data.frame(test.w=test.w)
test.w<-rbind(data.frame(test.w=rep(NA,8)),test.w)
wt_pm10.test<-cbind(wt_pm10.test,test.w)

#Obtener R cuadrada
test.w.lm<-lm(pm10.wavelet ~ test.w, data=wt_pm10.test)
n2<-summary(test.w.lm)$r.squared

#Borrar temporales
rm(list=c(ls(pattern="wt_"),
```

```
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))

#Cargar resultados
load("wavelet-selec.RData")

#Guardar resultados
save.image("wavelet-selec.RData")
```

Wavelet+RNA Symlet10+8-10-1

```
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
library(wavethresh)

#Carga de todas las tablas
load("tablas_procesadas.RData")

rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]
```

```
#Eliminar temporales
rm(list=c("temp","temp2"))

#Serie de datos para wavelet
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")

#Transformada Wavelet Discreta Symlet 10
pm_wd<-wd(pm,filter.number=10, family="DaubLeAsymm")

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)

#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))

#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)

#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"

#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]
wt_pm10<-wt_pm10.com[1:307,]

#Escalar la serie
pm10.w<-scale(wt_pm10.w)

#Obtener los datos de prueba
wt_pm10.w.test<-wt_pm10.com[308:512,3]
wt_pm10.test<-wt_pm10.com[308:512,]
```

```
#Cargar paquete para RNA
library(tsDyn)

#Entrenar la RED
mod.nnet.w3 <- nnetTs(pm10.w, m=8, size=10,step=1)

#obtener los valores ajustados y quitar el escalamiento
fit.w<-mod.nnet.w3$fitted.values
fit.w<-(fit.w*sd(wt_pm10.w))+mean(wt_pm10.w)
colnames(fit.w)[1]<-"fit.w"
fit.w<-rbind(data.frame(fit.w=rep(NA,8)),fit.w)

#Agregar a la base general
wt_pm10<-cbind(wt_pm10,fit.w)

#obtener R cuadrada
train.w.lm<-lm(pm10.wavelet ~ fit.w, data=wt_pm10)
t3<-summary(train.w.lm)$r.squared

#Guardar datos de escalamiento de la serie de entrenamiento
sd.w<-sd(wt_pm10.w)
mean.w<-mean(wt_pm10.w)

#Escalar la serie de prueba
pm10.w.test<-(wt_pm10.w.test-mean.w)/sd.w

#Aplicar el modelo a la serie de prueba
test.w<-0
for(i in 1:197){
  test.w[i]<-(predict(mod.nnet.w3,newdata=pm10.w.test[i:(i+7)],n.ahead=1)*
                sd.w)+mean.w
}

#Agregar los resultados a la serie de prueba
test.w<-data.frame(test.w=test.w)
test.w<-rbind(data.frame(test.w=rep(NA,8)),test.w)
wt_pm10.test<-cbind(wt_pm10.test,test.w)

#Obtener R cuadrada
```

```
test.w.lm<-lm(pm10.wavelet ~ test.w, data=wt_pm10.test)
n3<-summary(test.w.lm)$r.squared

#Borrar temporales
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))

#Cargar resultados
load("wavelet-selec.RData")

#Guardar resultados
save.image("wavelet-selec.RData")
```

Wavelet+RNA Symlet9+8-8-1

```
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
library(wavethresh)

#Carga de todas las tablas
load("tablas_procesadas.RData")

rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
```

```
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Serie de datos para wavelet
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")

#Transformada Wavelet Discreta Symlet 9
pm_wd<-wd(pm,filter.number=9, family="DaubLeAsymm")

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)

#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))

#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)

#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"

#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]
wt_pm10<-wt_pm10.com[1:307,]

#Escalar la serie
```

```
pm10.w<-scale(wt_pm10.w)

#Obtener los datos de prueba
wt_pm10.w.test<-wt_pm10.com[308:512,3]
wt_pm10.test<-wt_pm10.com[308:512,]

#Cargar paquete para RNA
library(tsDyn)

#Entrenar la RED
mod.nnet.w4 <- nnetTs(pm10.w, m=8, size=8,step=1)

#obtener los valores ajustados y quitar el escalamiento
fit.w<-mod.nnet.w4$fitted.values
fit.w<-(fit.w*sd(wt_pm10.w))+mean(wt_pm10.w)
colnames(fit.w)[1]<-"fit.w"
fit.w<-rbind(data.frame(fit.w=rep(NA,8)),fit.w)

#Agregar a la base general
wt_pm10<-cbind(wt_pm10,fit.w)

#obtener R cuadrada
train.w.lm<-lm(pm10.wavelet ~ fit.w, data=wt_pm10)
t4<-summary(train.w.lm)$r.squared

#Guardar datos de escalamiento de la serie de entrenamiento
sd.w<-sd(wt_pm10.w)
mean.w<-mean(wt_pm10.w)

#Escalar la serie de prueba
pm10.w.test<-(wt_pm10.w.test-mean.w)/sd.w

#Aplicar el modelo a la serie de prueba
test.w<-0
for(i in 1:197){
  test.w[i]<-(predict(mod.nnet.w4,newdata=pm10.w.test[i:(i+7)],n.ahead=1)*
                sd.w)+mean.w
}

#Agregar los resultados a la serie de prueba
```

```
test.w<-data.frame(test.w=test.w)
test.w<-rbind(data.frame(test.w=rep(NA,8)),test.w)
wt_pm10.test<-cbind(wt_pm10.test,test.w)

#Obtener R cuadrada
test.w.lm<-lm(pm10.wavelet ~ test.w, data=wt_pm10.test)
n4<-summary(test.w.lm)$r.squared

#Borrar temporales
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))

#Cargar resultados
load("wavelet-selec.RData")

#Guardar resultados
save.image("wavelet-selec.RData")
```

Wavelet+RNA Symlet9+8-6-1

```
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
library(wavethresh)

#Carga de todas las tablas
load("tablas_procesadas.RData")

rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
```

```
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Serie de datos para wavelet
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")

#Transformada Wavelet Discreta Symlet 9
pm_wd<-wd(pm,filter.number=9, family="DaubLeAsymm")

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)

#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))

#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)

#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"
```

```
#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]
wt_pm10<-wt_pm10.com[1:307,]

#Escalar la serie
pm10.w<-scale(wt_pm10.w)

#Obtener los datos de prueba
wt_pm10.w.test<-wt_pm10.com[308:512,3]
wt_pm10.test<-wt_pm10.com[308:512,]

#Cargar paquete para RNA
library(tsDyn)

#Entrenar la RED
mod.nnet.w5 <- nnetTs(pm10.w, m=8, size=6,step=1)

#obtener los valores ajustados y quitar el escalamiento
fit.w<-mod.nnet.w5$fitted.values
fit.w<-(fit.w*sd(wt_pm10.w))+mean(wt_pm10.w)
colnames(fit.w)[1]<-"fit.w"
fit.w<-rbind(data.frame(fit.w=rep(NA,8)),fit.w)

#Agregar a la base general
wt_pm10<-cbind(wt_pm10,fit.w)

#obtener R cuadrada
train.w.lm<-lm(pm10.wavelet ~ fit.w, data=wt_pm10)
t5<-summary(train.w.lm)$r.squared

#Guardar datos de escalamiento de la serie de entrenamiento
sd.w<-sd(wt_pm10.w)
mean.w<-mean(wt_pm10.w)

#Escalar la serie de prueba
pm10.w.test<-(wt_pm10.w.test-mean.w)/sd.w

#Aplicar el modelo a la serie de prueba
test.w<-0
for(i in 1:197){
```

```
  test.w[i]<-(predict(mod.nnet.w5,newdata=pm10.w.test[i:(i+7)],n.ahead=1)*
                sd.w)+mean.w
}

#Agregar los resultados a la serie de prueba
test.w<-data.frame(test.w=test.w)
test.w<-rbind(data.frame(test.w=rep(NA,8)),test.w)
wt_pm10.test<-cbind(wt_pm10.test,test.w)

#Obtener R cuadrada
test.w.lm<-lm(pm10.wavelet ~ test.w, data=wt_pm10.test)
n5<-summary(test.w.lm)$r.squared

#Borrar temporales
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))

#Cargar resultados
load("wavelet-selec.RData")

#Guardar resultados
save.image("wavelet-selec.RData")
```

Wavelet+RNA Symlet9+8-10-1

```
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
library(wavethresh)

#Carga de todas las tablas
load("tablas_procesadas.RData")
```

```
rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Serie de datos para wavelet
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")

#Transformada Wavelet Discreta Symlet 9
pm_wd<-wd(pm,filter.number=9, family="DaubLeAsymm")

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)

#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))

#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)
```

```
#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"

#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]
wt_pm10<-wt_pm10.com[1:307,]

#Escalar la serie
pm10.w<-scale(wt_pm10.w)

#Obtener los datos de prueba
wt_pm10.w.test<-wt_pm10.com[308:512,3]
wt_pm10.test<-wt_pm10.com[308:512,]

#Cargar paquete para RNA
library(tsDyn)

#Entrenar la RED
mod.nnet.w6 <- nnetTs(pm10.w, m=8, size=10,step=1)

#obtener los valores ajustados y quitar el escalamiento
fit.w<-mod.nnet.w6$fitted.values
fit.w<-(fit.w*sd(wt_pm10.w))+mean(wt_pm10.w)
colnames(fit.w)[1]<-"fit.w"
fit.w<-rbind(data.frame(fit.w=rep(NA,8)),fit.w)

#Agregar a la base general
wt_pm10<-cbind(wt_pm10,fit.w)

#obtener R cuadrada
train.w.lm<-lm(pm10.wavelet ~ fit.w, data=wt_pm10)
t6<-summary(train.w.lm)$r.squared

#Guardar datos de escalamiento de la serie de entrenamiento
sd.w<-sd(wt_pm10.w)
mean.w<-mean(wt_pm10.w)

#Escalar la serie de prueba
```

```
pm10.w.test<-(wt_pm10.w.test-mean.w)/sd.w

#Aplicar el modelo a la serie de prueba
test.w<-0
for(i in 1:197){
  test.w[i]<-(predict(mod.nnet.w6,newdata=pm10.w.test[i:(i+7)],n.ahead=1)*
                sd.w)+mean.w
}

#Agregar los resultados a la serie de prueba
test.w<-data.frame(test.w=test.w)
test.w<-rbind(data.frame(test.w=rep(NA,8)),test.w)
wt_pm10.test<-cbind(wt_pm10.test,test.w)

#Obtener R cuadrada
test.w.lm<-lm(pm10.wavelet ~ test.w, data=wt_pm10.test)
n6<-summary(test.w.lm)$r.squared

#Borrar temporales
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))

#Cargar resultados
load("wavelet-selec.RData")

#Guardar resultados
save.image("wavelet-selec.RData")
```

Wavelet+RNA Daubechies10+8-8-1

```
#Cargar Workspace
setwd("C:\\Users\\USUARIO\\Documents\\Ere\\TESINA_FINAL\\BASES_TESINA")

#Cargar paquete wavelets
```

```
library(wavethresh)

#Carga de todas las tablas
load("tablas_procesadas.RData")

rm(list=c(ls(pattern="f_"),
          ls(pattern="s"),
          ls(pattern="t"),
          ls(pattern="p"),
          ls(pattern="m")))
rm(list=c("fac_co","fac_no2","fac_o3","fac_q","fac_y"))

#Elegir últimos 2 años y quitar NA's.
temp<-fac_d[(fac_d$date > "2012-12-31"),c("date","pm10")]
wt_pm10<-fac_d[complete.cases(fac_d$pm10),c("date","pm10")]
temp2<-wt_pm10[(wt_pm10$date > "2012-12-31"),c("date","pm10")]

#Seleccionar unicamente los 512 últimos datos, se tiene que tener un
#número de datos igual a alguna potencia de 2 para wavelets
wt_pm10.com<-wt_pm10[2955:3466,]

#Eliminar temporales
rm(list=c("temp","temp2"))

#Serie de datos para wavelet
pm<-as.vector(wt_pm10.com$pm10,mode="numeric")

#Transformada Wavelet Discreta Daubechies 10
pm_wd<-wd(pm,filter.number=10, family="DaubExPhase")

#Extraer los coeficientes de la TDW aplicada en el paso anterior
#del nivel mas "fino"
pm_coef<-accessD(pm_wd,lev=nlevelsWT(pm_wd)-1)

#Obtener la desviación media absoluta de los coeficientes
pm_sigma<-mad(pm_coef)

#Obtener el threshold
pm_ut<-pm_sigma*sqrt(2*log(512))
```

```
#Aplicar el shrinkage hard
pm_wdT<-threshold(pm_wd,type="hard",policy="manual",value=pm_ut)

#Aplicar la inversa y obtener los datos "sin ruido"
pm_wr<-wr(pm_wdT)

#Agregar a la tabla los datos sin ruido
wt_pm10.com<-cbind(wt_pm10.com,pm_wr)
colnames(wt_pm10.com)[3]<-"pm10.wavelet"

#Obtener los datos para entrenamiento
wt_pm10.w<-wt_pm10.com[1:307,3]
wt_pm10<-wt_pm10.com[1:307,]

#Escalar la serie
pm10.w<-scale(wt_pm10.w)

#Obtener los datos de prueba
wt_pm10.w.test<-wt_pm10.com[308:512,3]
wt_pm10.test<-wt_pm10.com[308:512,]

#Cargar paquete para RNA
library(tsDyn)

#Entrenar la RED
mod.nnet.w7 <- nnetTs(pm10.w, m=8, size=8,step=1)

#obtener los valores ajustados y quitar el escalamiento
fit.w<-mod.nnet.w7$fitted.values
fit.w<-(fit.w*sd(wt_pm10.w))+mean(wt_pm10.w)
colnames(fit.w)[1]<-"fit.w"
fit.w<-rbind(data.frame(fit.w=rep(NA,8)),fit.w)

#Agregar a la base general
wt_pm10<-cbind(wt_pm10,fit.w)

#obtener R cuadrada
train.w.lm<-lm(pm10.wavelet ~ fit.w, data=wt_pm10)
t7<-summary(train.w.lm)$r.squared
```

```
#Guardar datos de escalamiento de la serie de entrenamiento
sd.w<-sd(wt_pm10.w)
mean.w<-mean(wt_pm10.w)

#Escalar la serie de prueba
pm10.w.test<-(wt_pm10.w.test-mean.w)/sd.w

#Aplicar el modelo a la serie de prueba
test.w<-0
for(i in 1:197){
  test.w[i]<-(predict(mod.nnet.w7,newdata=pm10.w.test[i:(i+7)],n.ahead=1)*
                sd.w)+mean.w
}

#Agregar los resultados a la serie de prueba
test.w<-data.frame(test.w=test.w)
test.w<-rbind(data.frame(test.w=rep(NA,8)),test.w)
wt_pm10.test<-cbind(wt_pm10.test,test.w)

#Obtener R cuadrada
test.w.lm<-lm(pm10.wavelet ~ test.w, data=wt_pm10.test)
n7<-summary(test.w.lm)$r.squared

#Gráfica Daubechies
library(openair)
pdf(file="completedata_wavelet_Daubechies.pdf",height=4,width=7,
    family="Times",pointsize=5,colormodel="grey")
timePlot(wt_pm10.com,pollutant=c("pm10","pm10.wavelet"),date.format
         = "%b-%Y",cols="greyscale",
         ylab = expression(mu * "g m" ^-3))
dev.off()

#Borrar temporales
rm(list=c(ls(pattern="wt_"),
          ls(pattern="test"),
          ls(pattern="train"),
          ls(pattern="sd"),
          ls(pattern="pm"),
          ls(pattern="f"),ls(pattern="mean"),"i"))
```

```
#Cargar resultados
load("wavelet-selec.RData")

#Tablas resumen
testing<-rbind(data.frame(testing=n1),
               data.frame(testing=n2),
               data.frame(testing=n3),
               data.frame(testing=n4),
               data.frame(testing=n5),
               data.frame(testing=n6),
               data.frame(testing=n7))
training<-rbind(data.frame(training=t1),
                data.frame(training=t2),
                data.frame(training=t3),
                data.frame(training=t4),
                data.frame(training=t5),
                data.frame(training=t6),
                data.frame(training=t7))
description<-data.frame(description=c("Symlet10+8-8-1","Symlet10+8-6-1",
                                      "Symlet10+8-10-1","Symlet9+8-8-1",
                                      "Symlet9+8-6-1","Symlet9+8-10-1",
                                      "Daubechies10+8-8-1"))
rna_wavelet_selection<-cbind(description,training,testing)
rna_wavelet_selection$training<-round(rna_wavelet_selection$training,3)
rna_wavelet_selection$testing<-round(rna_wavelet_selection$testing,3)
library(xtable)
xtable(rna_wavelet_selection, digits=3)

#Guardar
save.image("wavelet-selec.RData")
```

Bibliografía

Alonso, L. (2013). *Perceptrón Multicapa*. Recuperado el 1 de mayo de 2015, del Sitio web del Departamento de Informatica y Automatica de la Universidad de Salamanca: `http://avellano.fis.usal.es/~lalonso/RNA/introMLP.htm`.

Brook, R., Brook, J., y Rajagopalan, S. (2003). Air pollution: The "heart" of the problem. *Current Hypertension Reports*, 5(1):32–39.

Carslaw, D. y Ropkins, K. (2012). openair — An R package for air quality data analysis. *Environmental Modelling & Software*, 27–28(0):52–61.

Carslaw, D. y Ropkins, K. (2015). *openair: Open-source tools for the analysis of air pollution data*. R package version 1.5.

Cazelles, B., Chavez, M., Berteaux, D., Ménard, F., Vik, J., Jenouvrier, S., y Stenseth, N. (2008). Wavelet analysis of ecological time series. *Oecologia*, 156:287–304.

Cheng, B. y Titterington, D. (1994). Neural networks: A review from a statistical perspective. *Statistical science*, pp. 2–30.

Comisión Ambiental Metropolitana (2011). *Programa para mejorar la calidad del aire de la Zona Metropolitana del Valle de México 2011-2020* [en línea]. México, D.F. [fecha de consulta: 25 de marzo de 2015] Disponible en: `http://www.sedema.df.gob.mx/flippingbook/proaire2011-2020/`.

Connor, J., Martin, R., y Atlas, L. (1994). Recurrent neural networks and robust time series prediction. *Neural Networks, IEEE Transactions on*, 5(2):240–254.

Dahl, D. (2014). *xtable: Export tables to LaTeX or HTML*. R package version 1.7-4.

Di Narzo, A., Di Narzo, F., Aznarte, J., y Stigler, M. (2009). *tsDyn: Time series analysis based on dynamical systems theory*. R package version 0.7.

Donoho, D. y Johnstone, J. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455.

Fernández, D. (2004). Reducción del ruido y prediccón de series temporales de alta frecuencia mediante sistemas dinámicos no lineales y técnicas neurales (Documento de trabajo No. 001-2014). *Banco Central del Uruguay*.

Gao, H. y Bruce, A. (1997). Waveshrink with firm shrinkage. *Statistica Sinica*, 7(4):855–874.

Gao, R. y Yan, R. (2011). Discrete wavelet transform. En *Wavelets*, pp. 49–68. Springer US.

Granados, G., Retama, A., Jaimes, M., y Becerra, A. (2012). *Calidad del Aire en la Ciudad de México; informe 2011* [en línea]. Secretaría del Medio Ambiente del Gobierno del Distrito Federal [fecha de consulta: 20 de marzo de 2015]. Disponible en: `http://www.sedema.df.gob.mx/flippingbook/informe_anual_calidad_aire_2011/`.

Hu, Y. y Hwang, J. (2001). *Handbook of neural network signal processing*. CRC press.

Hubbard, B. (1998). *The world according to wavelets: the story of a mathematical technique in the making*. Massachusetts:A K Peters, Ltd.

Kaiser, G. (1994). *A friendly guide to wavelets*. Boston: Birkhäuserr.

Kriesel, D. (2007). *A Brief Introduction to Neural Networks* [en línea]. [fecha de consulta: 12 de abril de 2015] Disponible en: `http://www.dkriesel.com`.

Lotric, U. y Dobnikar, A. (2005). Predicting time series using neural networks with wavelet-based denoising layers. *Neural Computing & Applications*, 14(1):11–17.

Moustris, K., Larissi, I., Nastos, P., Koukouletsos, K., y Paliatsos, A. (2013). Development and Application of Artificial Neural Network Modeling in Forecasting PM_{10} Levels in a Mediterranean City. *Water, Air, & Soil Pollution*, 224(8):1–11.

Nason, G. (2008). *Wavelet Methods in Statistics with R.* Use R! Springer New York.

Nason, G. (2013). *wavethresh: Wavelets statistics and transforms.* R package version 4.6.6.

Nason, G. y Silverman, B. (1994). The Discrete Wavelet Transform in S. *Journal of Computational and Graphical Statistics*, 3(2):163–191.

Percival, D. y Walden, A. (2000). *Wavelet Methods for Time Series Analysis.* Cambridge series in statistical and probabilistic mathematics. Cambridge University Press.

Prakash, A., Kumar, U., Kumar, K., y Jain, V. (2011). A Wavelet-based Neural Network Model to Predict Ambient Air Pollutants' Concentration. *Environmental Modeling and Assessment*, 16:503–517.

R Core Team (2015). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Roussou, T. y Behrakis, P. (2005). The respiratory effects of air pollution. En Nicolopoulou-Stamati, P., Hens, L., y Howard, C., editores, *Environmental Health Impacts of Transport and Mobility*, volumen 21 de *Environmental Science and Technology Library*, pp. 79–94. Springer Netherlands.

Sarkar, D. y Andrews, F. (2013). *latticeExtra: Extra Graphical Utilities Based on Lattice.* R package version 0.6-26.

Secretaría de Salud (1993a). NORMA Oficial Mexicana NOM-021-SSA1-1993, *Salud ambiental. Criterio para evaluar la calidad del aire ambiente con respecto al monóxido de carbono (CO). Valor permisible para la concentración de monóxido de carbono (CO) en el aire ambiente como medida de protección a la salud de la población.* Diario Oficial de la Federación, 18 de agosto de 1994.

Secretaría de Salud (1993b). NORMA Oficial Mexicana NOM-023-SSA1-1993, *Salud ambiental. Criterio para evaluar la calidad del aire ambiente con respecto al bióxido de nitrógeno (NO_2). Valor normado para la concentración de bióxido de nitrógeno (NO_2) en el aire ambiente como medida de protección a la salud de la población.* Diario Oficial de la Federación, 18 de agosto de 1994.

Secretaría de Salud (2010). NORMA Oficial Mexicana NOM-022-SSA1-2010, *Salud ambiental. Criterio para evaluar la calidad del aire ambiente, con respecto al dióxido de azufre (SO_2). Valor normado para la concentración de dióxido de azufre (SO_2) en el aire ambiente, como medida de protección a la salud de la población.* Diario Oficial de la Federación, 8 de septiembre de 2010.

Secretaría de Salud (2014a). NORMA Oficial Mexicana NOM-020-SSA1-2014, *Salud ambiental. Valor límite permisible para la concentración de ozono (O3) en el aire ambiente y criterios para su evaluación.* Diario Oficial de la Federación, 19 de agosto de 2014.

Secretaría de Salud (2014b). NORMA Oficial Mexicana NOM-025-SSA1-2014, *Salud ambiental. Valores límite permisibles para la concentración de partículas suspendidas PM10 y PM2.5 en el aire ambiente y criterios para su evaluación.* Diario Oficial de la Federación, 20 de agosto de 2014.

Shekarrizfard, M., Karimi-Jashni, A., y Hadad, K. (2011). Wavelet transform-based artificial neural networks (WT-ANN) in PM_{10} pollution level estimation, based on circular variables. *Environmental Science and Pollution Research*, pp. 1–13.

Shima, H. y Nakayama, T. (2010). Wavelet transformation. En *Higher Mathematics for Physics and Engineering*, pp. 449–480. Springer Berlin Heidelberg.

Solomon, P., Costantini, M., Grahame, T., Gerlofs-Nijland, M., Cassee, F., Russell, A., Brook, J., Hopke, P., Hidy, G., Phalen, R., Saldiva, P., Sarnat, S., Balmes, J., Tager, I., Özkaynak, H., Vedal, S., Wierman, S., y Costa, D. (2012). Air pollution and health: bridging the gap from sources to health outcomes: conference summary. *Air Quality, Atmosphere & Health*, 5(1):9–62.

Stigler, M. (2010). *Threshold cointegration: overview and implementation in R.* R package version 0.7-2.

Tong, W., Li, Y., y Q., Y. (2006). A Wavelet Analysis Based Data Processing for Time Series of Data Mining Predicting. En Ng, W.-K., Kitsuregawa, M., Li, J., y Chang, K., editores, *Advances in Knowledge Discovery and Data Mining*, volumen 3918 de *Lecture Notes in Computer Science*, pp. 780–789. Springer Berlin Heidelberg.

Wickham, H. (2007). Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20.

Xia, J., Wu, X., y Yuan, Y. (2007). Integration of wavelet transform with PCA and ANN for metabolomics data-mining. *Metabolomics*, 3(4):531–537.

Zhang, G., Patuwo, B., y Hu, M. (1998). Forecasting with artificial neural networks: The state of the art. *International journal of forecasting*, 14(1):35–62.