ITAM

Aprendizaje de Máquina

Luis Alpízar - 121369
Maestría en Ciencias en Computación

Salvador García - 119718
Maestría en Ciencias en Computación

Alan Córdova - 114712
Maestría en Ciencias en Computación

Conciliación de pronósticos para series de tiempo jerárquicas con Machine Learning

Proyecto Final

Otoño 2020

Profesor:
L. Felipe González Perez

Índice general

Capítulo 1

Introducción

En este capítulo exponemos el problema que buscamos resolver y mencionamos brevemente los acercamientos más populares.

1.1. Identificación del problema

Es común encontrar datos de series de tiempo cuya organización refleja una estructura de jerarquía. Cualquier cadena u institución con presencia nacional o internacional debe considerar las delimitaciones geográficas en su planeación. Por ejemplo, la venta de un artículo se puede visualizar como su venta en una tienda específica, en un estado o en una región. Similarmente, dicho artículo puede ser parte de una categoría específica que a su vez es parte de una línea de producto, por lo que su venta debe estar reflejada en estos niveles.

Una muestra de lo anterior se observa en la figura **??**. A este tipo de datos cuyo valor puede ser agregado y que refleja un orden temporal se le conoce como series de tiempo jerárquicas [**?**].

Realizar predicciones para datos de series de tiempo es un problema bien conocido y con distintas técnicas y modelos según el caso al que nos enfrentemos. Nos interesan estas predicciones, pues ayudan en la toma de decisiones complejas ante mucha incertidumbre (por ejemplo, resulta complicado pronosticar la demanda futura de turistas en el próximo verano en una zona o la demanda de SKUs en las distintas tiendas de una cadena empresarial) [**?**].

En particular, las series de tiempo jerárquicas tienen la dificultad adicional de conciliar los pronósticos, esto es, de garantizar que la suma de las predicciones individuales es equivalente a realizar una predicción del total general (podemos pronosticar la demanda de turistas en un hotel en una región y así para todo el país. La suma de estas debe ser equivalente al pronóstico individual de

Figura 1.1: Ejemplo visual para las ventas nacionales de un producto con estructura de jerarquía

.

la demanda de la temporada de verano completa). El proceso de conciliación típicamente degrada la exactitud del pronóstico [**?**].

Para resolver lo anterior, se tienen las técnicas de "abajo-arriba", "arriba-abajo" y "punto medio" donde usamos como punto pivote la predicción en algún nivel de la jerarquía. Dado que la conciliación se hace como un paso posterior al pronóstico, existen trabajos dedicados al relajamiento del cómputo requerido para pronosticar y conciliar en un mismo paso como el de [**?**].

También está el acercamiento combinatorio que no se basa en un nodo pivote sino que utiliza toda la información de la jerarquía para conciliar las predicciones. La información es codificada con un sistema de pesos que dependen principalmente de la jerarquía y no los datos. Este acercamiento ha visto mucha actividad reciente, pues tecnologías y técnicas como las de Machine Learning (ML) intentan descifrar la mejor manera de presentar el sistema de pesos [**?**, **?**, **?**].

Este documento sigue la siguiente estructura: en el capítulo 2 analizamos el marco teórico para la predicción con series de tiempo jerárquicas; en el capítulo 3 exponemos un modelo que realiza predicciones con ayuda de ML y exhibimos los resultados obtenidos, y en el último capítulo, se indican nuestras conclusiones.

Capítulo 2

Marco Teórico y herramientas de desarrollo

Como mencionamos anteriormente, uno de los principales obstáculos para realizar predicciones a series de tiempo con niveles de agregación o jerarquía es garantizar su coherencia en todos los niveles. Esto es, que al agregar las predicciones, estas sean consistentes con la propia jerarquía. En este capítulo analizaremos algunas de las formas más utilizadas para conciliar predicciones de series de tiempo jerárquicas.

Utilizaremos la jerarquía de la figura **??** y la siguiente notación tomada de [**?**]:

- m: total de nodos de la serie
- m_i: total de nodos en el nivel i
- k: niveles de la jerarquía
- n: observaciones en una serie
- y_t: t-ésima observación
- $\hat{y_t}$: predicción de la t-ésima observación
- $\mathbf{Y}_{i,t}$: vector de todas las obervaciones del nivel i
- $\hat{\mathbf{Y}}_{i,t}(h)$: vector de predicciones con h saltos por delante para el nivel i
- $\tilde{Y}_n(h)$: vector de predicciones conciliadas con h saltos por delante para el nivel i

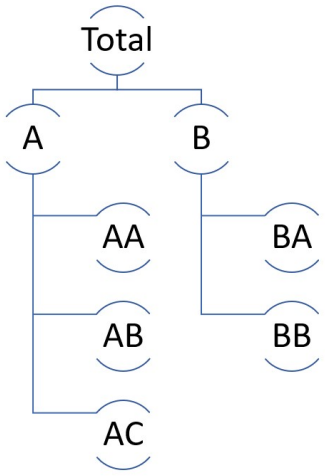


Figura 2.1: Jerarquía ejemplo

2.1. Técnicas de conciliación en una jerarquía

Tradicionalmente, las predicciones en series de tiempo de jerarquía requieren de seleccionar un nivel en la estructura para realizar la predicción y de ahí agregarla (niveles superiores) o desagregarla (niveles inferiores) para los otros niveles.

2.1.1. Buttom-up

El método de "abajo-arriba" es uno de los más simples pues únicamente implica realizar la predicción en la base de la jerarquía. Una vez hecho esto, las predicciones se suman para conformar la información de los niveles superiores.

La principal ventaja de esta técnica es que considera el nivel más bajo de la serie por lo que no perdemos información atómica. No obstante, esta técnica tiene la desventaja de ser muy ruidosa.

Por ejemplo, con la jerarquía de la figura **??** calcularíamos primero las predicciones (con h saltos por delante) de la base: $\hat{Y}_{AA,h}$, $\hat{Y}_{AB,h}$, $\hat{Y}_{AC,h}$, $\hat{Y}_{BA,h}$ y $\hat{Y}_{BB,h}$. Luego las sumaríamos para obtener la predicción conciliada del total de diferentes maneras como:

$$\tilde{Y}_h = \hat{Y}_{AA,h} + \hat{Y}_{AB,h} + \hat{Y}_{AC,h} + \hat{Y}_{BA,h} + \hat{Y}_{BB,h} \tag{2.1}$$

$$\tilde{Y}_h = \tilde{Y}_{A,h} + \tilde{Y}_{B,h} \tag{2.2}$$

Donde las predicciones conciliadas para los niveles A y B se calculan al sumar las predicciones base de sus elementos.

2.1.2. Top-down

Por su parte, el método de "arriba-abajo" genera una única predicción en el nivel más agregado la cual se desagrega a los niveles inferiores. Para realizar lo anterior, se utilizan proporciones que ayudan a determinar cómo manejar la predicción en los distintos niveles.

La principal ventaja de este método es su simplicidad al usar una sola predicción, la del total. Sin embargo, su defecto es la pérdida de información de los niveles inferiores como patrones de temporada o eventos especiales por mencionar algunos.

En este caso, para la jerarquía de la figura **??**, primero calculamos la predicción en la cima, luego calculamos las predicciones en la base usando las proporciones p_1, p_2, p_3, p_4 y p_5 de la siguiente manera: $\tilde{Y}_{AA,t} = p_1\hat{Y}_t$, $\tilde{Y}_{AB,t} = p_2\hat{Y}_t$, $\tilde{Y}_{AC,t} = p_3\hat{Y}_t$, $\tilde{Y}_{BA,t} = p_4\hat{Y}_t$, $\tilde{Y}_{BB,t} = p_5\hat{Y}_t$.

Para calcular las proporciones existen 2 métodos populares: promedio de las proporciones históricas y proporciones de los promedios históricos. El primero tiene la fórmula indicada en la ecación **??**, mientras que el segundo se expresa como la ecuación **??**. Donde j va de 1 a m_k.

$$p_j = \frac{1}{n}\sum_{t=1}^{n}\frac{Y_{j,t}}{Y_t} \tag{2.3}$$

$$p_j = \frac{\sum_{t=1}^{n} Y_{j,t}}{\sum_{t=1}^{n} Y_t} \tag{2.4}$$

2.1.3. Middle-out

Esta técnica junta las dos anteriores a partir de un nivel pivote. Los niveles superiores se calculan con la técnica "abajo-arriba" mientras que los niveles inferiores con la técnica "arriba-abajo ". Su principal desventaja es que el analista debe elegir el nivel pivote y sustentar su elección.

Como podemos observar, el camino sugerido para la conciliación de pronósticos es primero realizar la predicción ($\hat{Y}_t$) y luego conciliar por alguno de los métodos anteriores ($\tilde{Y}_t$)

Sin embargo, la 3 técnicas descritas tienen el detalle de que se enfocan en un nivel particular de agregación para hacer el pronóstico. Esto tiene la consecuencia de ignorar información relevante de los otros niveles. Por ello, las tecnicas que combinan las bondades de las 3 anteriores y consideran la información en todos los niveles de la jerarquía son superiores. Estas técnicas son llamadas de

combinación óptima, pues mediante pesos calculados, combinan las predicciones obtenidas garantizando la coherencia requerida. Estos pesos dependen solo de la jerarquía y no de los datos.

Es importante destacar que se facilita la visualización de la agregación de series de tiempo jerárquicas con el uso de notación matricial.

La figura **??** en el tiempo t se puede representar de la siguiente manera:

$$\begin{bmatrix} Y_t \\ Y_{A,t} \\ Y_{B,t} \\ Y_{AA,t} \\ Y_{AB,t} \\ Y_{AC,t} \\ Y_{BA,t} \\ Y_{BB,t} \end{bmatrix} = \begin{bmatrix} 1,1,1,1,1 \\ 1,1,1,0,0 \\ 0,0,0,1,1 \\ 1,0,0,0,0 \\ 0,1,0,0,0 \\ 0,0,1,0,0 \\ 0,0,0,1,0 \\ 0,0,0,0,1 \end{bmatrix} \begin{bmatrix} Y_{AA,t} \\ Y_{AB,t} \\ Y_{AC,t} \\ Y_{BA,t} \\ Y_{BB,t} \end{bmatrix} \tag{2.5}$$

De manera compacta, la serie $Y_t = SY_{k,t}$ se calcula con la matriz S de agregación de m x m_k y el vector $Y_{k,t}$ con las observaciones en el nivel k de la jerarquía en el tiempo t. En este caso, las del último nivel. Notemos que estas últimas se expresan con la matriz identidad.

2.2. ML Framework

El framework para realizar pronósticos de series de tiempo con ML es similar al visto en el curso. Destacamos el camino sugerido por [**?**] el cual comienza con la preparación de los datos, su visualización y su descripción con un modelo; posteriormente el modelo se entrena con los datos de entrenamiento; finalmente, se evalúa su desempeño con una muestra separada de prueba. Si el desempeño no es satisfactorio, realizamos modificaciones en los primeros pasos. Con un modelo apropiado, entonces se realizan las predicciones.

2.2.1. Evaluación de residuales

Cuando hacemos predicciones de series de tiempo, comúnmente usamos toda la información previa al punto de predicción. Es decir, un pronóstico $\hat{y}_t$ involucra toda la serie hasta t-1. Aunque existen diversas formas de construir la predicción, se utiliza el acercamiento naive como punto de comparación. Este acercamiento propone como predicción $\hat{y}_t = y_{t-1}$. A los pronósticos les llamamos valores ajustados, pues involucran esa información previa.

Con los valores ajustados de una serie, podemos checar el ajuste al compararlo contra el valor real en el mismo tiempo t. La diferencia entre estos valores se llama residual y nos permite determinar si el modelo está capturando información relevante de los datos.

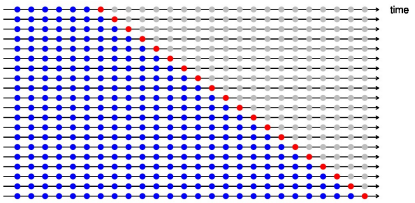


Figura 2.2: Validación cruzada a 1 tiempo en el futuro. Los cortes de entrenamiento comprenden las observaciones azules. Los cortes de prueba se construyen con la observación roja.

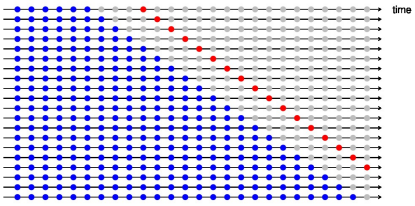


Figura 2.3: Validación cruzada a h tiempos en el futuro. Los cortes de entrenamiento comprenden las observaciones azules. Los cortes de prueba se construyen con la observación roja.

Un buen modelo de predicción de series de tiempo tiene las siguientes características en sus residuales: no están correlacionados y tienen media 0. Cuando esto no sucede, entonces el modelo puede ser mejorado. Adicionalmente, resulta beneficioso que los residuales tengan varianza constante y que su distribución sea normal, pues facilitan el cálculo de intervalos de predicción [**?**].

2.2.2. Separación de datos

Para evaluar las predicciones de un modelo bien ajustado necesitamos valores reales, pues la información de los residuales subestima el peso de un error predictivo. Por ello, utilizamos una muestra de entrenamiento para ajustar el modelo y una muestra separada para probar su capacidad predictora.

Comunmente, la separación en series de tiempo utiliza la siguiente forma: la muestra de entrenamiento comprende las observaciones $\{y_1, ..., y_t\}$, mientras que la muestra de prueba va de $\{y_{t+1}, ...\}$ en adelante. De esta manera, podemos hablar del error de predicción de un pronóstico como la diferencia entre el valor observado y el propio pronóstico $e_t = y_t - \hat{y}_t$.

Dicho lo anterior, la manera de implementar validación cruzada en series de tiempo implica garantizar que los cortes de entrenamiento y prueba tengan la estructura previamente descrita: la muestra entrenamiento siempre tiene información previa a la muestra de prueba. Normalmente construimos la muestra de prueba con una sola observación en el tiempo t mientras que la muestra de

entrenamiento tiene las observaciones hasta t-1. Esto se puede ver en la figuras **??**. Por su parte la figura **??** indica la misma idea cuando el pronóstico es h tiempos adelante.

Destacamos la diferencia entre residuales y errores de predicción ya que los primeros solo se calculan en la muestra de entrenamiento con diferencia de 1 tiempo t, pero los segundos se calculan en la muestra de prueba con diferencia de h tiempos, pues reflejan el pronóstico en el futuro deseado.

Finalmente, utilizamos algunas medidas tradicionales de ML para medir el desempeño de las predicciones, en particular, cuando las observaciones están en la misma escala podemos usar MAE o RMSE. Cabe destacar que es común la ausencia de misma escala en los datos por lo que ajustes como MASE o RMSSE, donde se toma en cuenta qué tan alejada fue la predicción respecto al modelo naive, son preferibles [**?**].

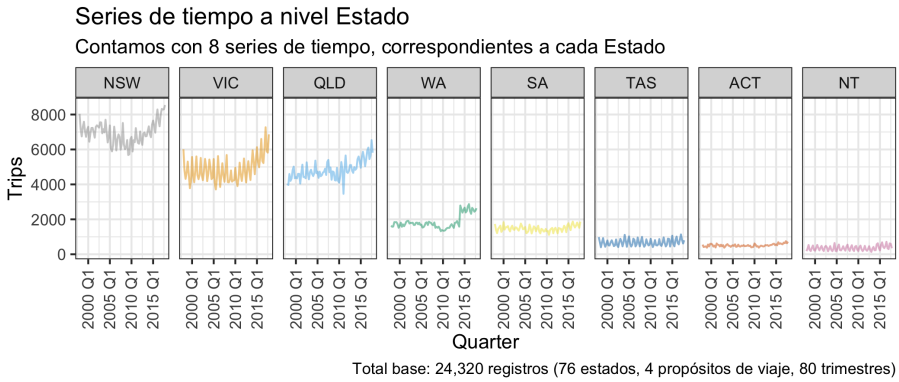
2.3. Base de datos

Para ejemplificar la conciliación de series de tiempo jerárquica, se usará la base de datos (**tourism**), que contiene los viajes nocturnos trimestrales desde el primer trimestre de 1998 hasta el cuarto trimestre de 2016 en Australia. Las variables de esta base de datos son:

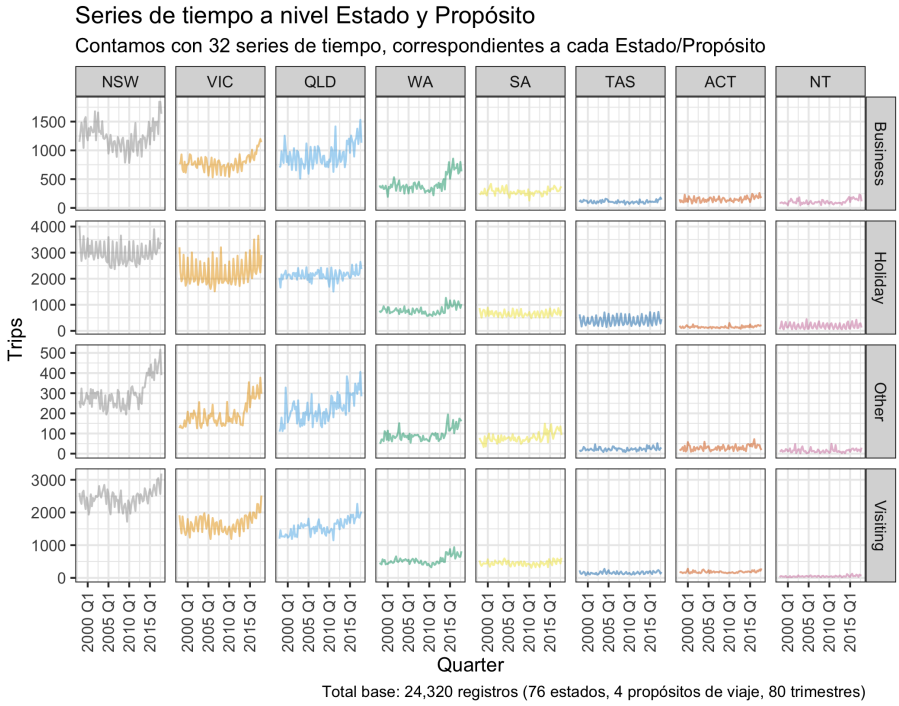
- **Quarter:** Trimestre
- **State:** 8 Estados y territorios de Australia.
- **Region:** 76 Regiones formadas a partir de la agregación de Áreas Locales Estadísticas (SLA), definnidas por autoridades de turismo estatales.
- **Purpose:** 4 Propósitos de visita ("Holiday", "Visiting friends and relatives", "Business", .ºther reason").
- **Trips:** Viajes nocturnos en miles.

En *tourism* se tiene una jerarquía dentro de sus variables, para cada Estado, se pueden tener distintas Regiones contenidas. En R, la base de datos es parte del paquete *tsibble* que se explicará más adelante. Como un breve EDA, tenemos que la base contiene 24,320 registros. Al ver todas las series de tiempo que se pueden generar hay que considerar los distintos niveles de agregación:

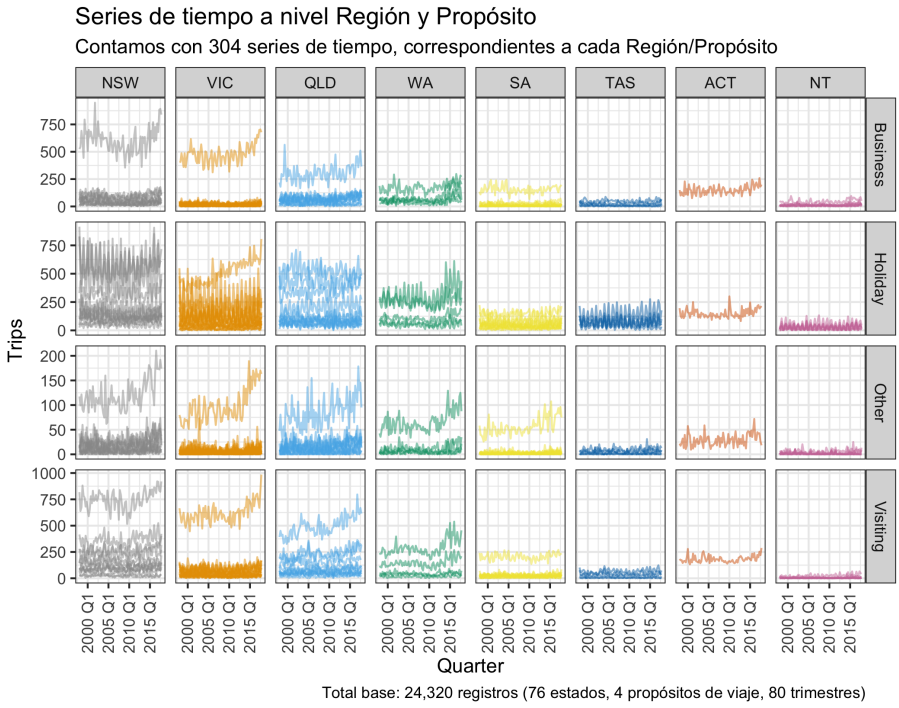
Al considerar las estados podemos contar 8 series de tiempo a un nivel estado:



Si añadimos un nivel extra a la serie de tiempo, es decir consideramos todas las posibles series de tiempo a nivel estado y propósitos podemos tener 32 series de tiempo:



Finalmente, cuando consideramos el nivel de agregación más pequeño podemos llegar a un nivel región - propósito donde podemos tener 304 series de tiempo:



2.4. Paquetería empleada de R:

Para resolver el problema tomamos la colección de paquetes de *Tidyverts*, que es un simil de *Tidyverse* y *Tidymodels* enfocado en series de tiempo. Esta colección de paquetes tiene contribuciones de Rob Hyndman autor del libro *Forecasting: Principles and Practice* [**?**], donde se plantean temas de series de tiempo desde un punto de vista aplicado y enfocado a R.

Los paquetes de *tidyverts* permiten predicciones de series de tiempo con grupos. Por ejemplo *tourism* tiene establecida por default la llave: {Estado, Region, Propósito} el cual nos lleva a contar los 304 grupos indicados anteriormente. La base de datos de *tourism* empleada en este trabajo consiste en un objeto de tipo *tsibble*, donde:

- **Indice (index):** es una variable con un orden inherente del pasado al presente (En la base corresponde a la variable Quarter). Las posibles opciones que se pueden utilizar son:
  - Anual
  - Trimestral
  - Mensual
  - Semanal
  - Diario
  - Subdiario (A varios niveles como hora y minuto)
- **Clave (key):** es un conjunto de variables que definen las unidades de observación a lo largo del tiempo. (En la base corresponde a las variables {Estado, Region, Propósito})
- **Mediciones:** corresponden a los valores (o unidades de medida) que vamos a pronosticar.

La colección contiene los siguientes paquetes:

tsibble: Provee una clase *tbl_ts* para datos temporales. Adicional contiene funciones que permiten analizar estos objetos

fable: Contiene una colección de modelos para realizar time series forecasting. Lo interesante es que fucionan con el framework de *fable* proveido por el paquete *fabletools*

fable.prophet: Contiene wrapers para el paquete de prophet que permite que sean utilizados en el mismo flujo del framework

feasts: Provee una colección de características, métricas estadísticas y funciones para graficar que es compatible con el mismo framework

fable.tools: Contiene las funcionalidades y herramientas core del *fable* framework.

2.5. Ejemplo del Framework tidyverts

Con fines exploratorios se exhiben las principales funciones de tidyverts para series de tiempo. Principalmente, tenemos la funcion *Model* que permite construir distintos modelos como se muestra en el siguiente código. En este caso, generamos 3 modelos distintos y comunes en la literatura de series de tiempo (ARIMA, ETS, SNAIVE) para los 304 grupos:

```
1 fit <-
2   tourism %>%
3   filter(year(Quarter) <= 2014) %>%
4   model(
5     snaive = SNAIVE(Trips ~ lag("year")),
6     ets = ETS(Trips),
7     arima = ARIMA(Trips)
8   )
```

```
> fit
# A mable: 304 x 6
# Key:     Region, State, Purpose [304]
   Region         State Purpose    snaive          ets                            arima
   <chr>          <chr> <chr>      <model>      <model>                        <model>
 1 Adelaide       SA    Business <SNAIVE> <ETS(M,N,M)> <ARIMA(0,0,0)(1,0,1)[4] w/ mean>
 2 Adelaide       SA    Holiday  <SNAIVE> <ETS(A,N,A)> <ARIMA(0,0,0)(1,0,1)[4] w/ mean>
 3 Adelaide       SA    Other    <SNAIVE> <ETS(M,A,N)>          <ARIMA(0,1,1) w/ drift>
 4 Adelaide       SA    Visiting <SNAIVE> <ETS(A,N,A)> <ARIMA(0,0,0)(1,0,1)[4] w/ mean>
 5 Adelaide Hills SA    Business <SNAIVE> <ETS(A,N,N)>           <ARIMA(0,0,0) w/ mean>
 6 Adelaide Hills SA    Holiday  <SNAIVE> <ETS(A,A,N)>                   <ARIMA(0,1,1)>
 7 Adelaide Hills SA    Other    <SNAIVE> <ETS(A,N,N)>         <ARIMA(0,1,2)(0,0,2)[4]>
 8 Adelaide Hills SA    Visiting <SNAIVE> <ETS(M,A,M)>                   <ARIMA(0,1,1)>
 9 Alice Springs  NT    Business <SNAIVE> <ETS(M,N,M)>         <ARIMA(0,1,1)(0,0,1)[4]>
10 Alice Springs  NT    Holiday  <SNAIVE> <ETS(M,N,A)>         <ARIMA(0,0,0)(0,1,2)[4]>
# … with 294 more rows
```

Es importante notar en la línea 3 se aplica un filtro a los datos. Esto es reelevante por que es requerido para poder realizar predicciones y comparar con el dato verdadero

Una ventaja de tidyverts, es que se puede analizar la especificacion para cada modelo en particular. Por ejemplo: Se puede seleccionar el modelo ARIMA para el grupo de **Region = Snowy Mountains** y **Purpose = Holiday**. Con esto vemos la especificacion del modelo, asi como sus respectivos coeficientes con sus errores estándar. Adicionalmente, proporciona la log likelihood, AIC y BIC.

```
fit %>%
  filter(Region == "Snowy Mountains", Purpose == "Holiday") %>%
  select(arima) %>%
  report()
```

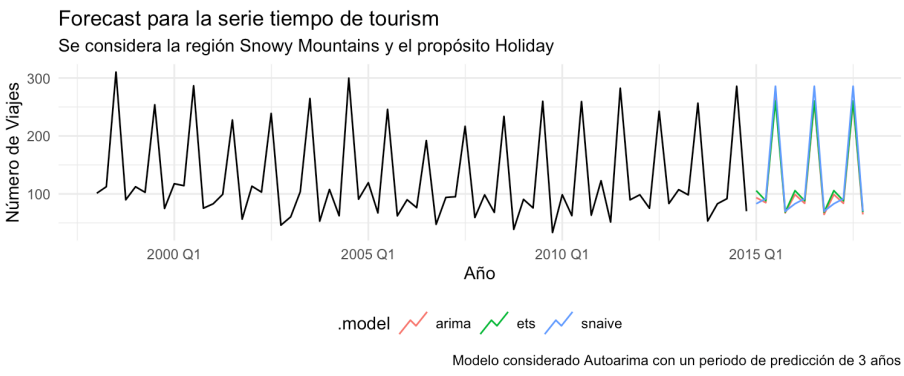
```
Series: Trips
Model: ARIMA(1,0,0)(0,1,2)[4]

Coefficients:
         ar1     sma1     sma2
      0.2162  -0.3711  -0.1897
s.e.  0.1156   0.1280   0.1161

sigma^2 estimated as 592.9:  log likelihood=-349.58
AIC=707.16   AICc=707.72   BIC=716.48
```

Además, podemos incluir predicciones (forecast) de la serie de tiempo para un periodo especificado:

```
fit %>%
  filter(Region == "Snowy Mountains", Purpose == "Holiday") %>%
  forecast(h = "3 years")
```



Adicionalmente, se puede determinar el error de esta predicción con la función *accuracy*:

```
accuracy(model\_for, tourism)
```

	.model	Region	State	Purpose	.type	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
	<chr>	<chr>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	arima	Snowy Mountains	NSW	Holiday	Test	26.3	44.6	29.3	12.3	16.3	1.40	-0.000738
2	ets	Snowy Mountains	NSW	Holiday	Test	22.4	43.7	29.9	8.07	17.2	1.43	-0.00752
3	snaive	Snowy Mountains	NSW	Holiday	Test	20.4	35.1	26.2	10.2	17.8	1.25	-0.0178

Algo interesante de este approach, es que se pueden utilizar funciones genéricas como *glance*, *coef* y *augment*. La primera nos permite tener un "vistazo"de nuestro modelo ajustado. En este caso podemos para cada Región, Estado y Propósito los modelos especificados, su AIC, BIC, MAE, MASE, etc.

```
fit %>%
  glance() %>%
  arrange(MSE)
```

```
# A tibble: 912 x 14
   Region          State Purpose  .model sigma2 log_lik   AIC  AICc   BIC   MSE  AMSE   MAE ar_roots ma_roots
   <chr>           <chr> <chr>    <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <list>   <list>
 1 Kangaroo Island SA    Other    ets     0.832   -136.  278.  279.  285. 0.808 0.819 0.604 <NULL>   <NULL>
 2 MacDonnell      NT    Other    ets     1.04    -143.  296.  297.  307. 0.983 0.984 0.676 <NULL>   <NULL>
 3 Adelaide Hills  SA    Other    ets     1.89    -164.  334.  335.  341. 1.83  1.79  1.00  <NULL>   <NULL>
 4 Wilderness West TAS   Other    ets     2.02    -166.  339.  339.  345. 1.96  1.99  0.719 <NULL>   <NULL>
 5 Barkly          NT    Other    ets     2.13    -168.  342.  343.  349. 2.06  2.09  0.923 <NULL>   <NULL>
 6 Yorke Peninsula SA    Other    ets     2.28    -170.  347.  347.  354. 2.21  2.15  1.20  <NULL>   <NULL>
 7 MacDonnell      NT    Visiting ets     2.30    -171.  347.  348.  354. 2.23  2.25  1.14  <NULL>   <NULL>
 8 East Coast      TAS   Other    ets     2.32    -171.  348.  348.  355. 2.25  2.27  1.23  <NULL>   <NULL>
 9 Clare Valley    SA    Other    ets     2.39    -172.  350.  351.  357. 2.32  2.33  1.05  <NULL>   <NULL>
10 Lasseter        NT    Visiting ets     2.65    -176.  357.  358.  364. 2.57  2.61  1.27  <NULL>   <NULL>
# … with 902 more rows
```

Adicionalmente, es posible obtener el valor de los coeficientes que se estiman, así como su error estándard y su respectivo p-value.

```
fit %>%
  select(Region, State, Purpose, arima) %>%
  coef()
```

```
# A tibble: 586 x 9
   Region   State Purpose  .model term     estimate std.error statistic  p.value
   <chr>    <chr> <chr>    <chr>  <chr>       <dbl>     <dbl>     <dbl>    <dbl>
 1 Adelaide SA    Business arima  sar1        0.802    0.184       4.36 4.56e- 5
 2 Adelaide SA    Business arima  sma1       -0.632    0.230      -2.74 7.77e- 3
 3 Adelaide SA    Business arima  constant   30.2      1.39       21.8  2.18e-32
 4 Adelaide SA    Holiday  arima  sar1        0.283    0.121       2.35 2.17e- 2
 5 Adelaide SA    Holiday  arima  sar2        0.327    0.127       2.58 1.19e- 2
 6 Adelaide SA    Holiday  arima  constant   60.3      2.33       25.9  5.59e-37
 7 Adelaide SA    Other    arima  ma1        -0.877    0.0670    -13.1  3.73e-20
 8 Adelaide SA    Other    arima  constant    0.422    0.215       1.96 5.39e- 2
 9 Adelaide SA    Visiting arima  sar1        0.947    0.0666     14.2  4.89e-22
10 Adelaide SA    Visiting arima  sma1       -0.796    0.136      -5.84 1.61e- 7
# … with 576 more rows
```

Por último se puede observar el valor ajustado y el residual por observación y modelo con la función *augment*

```
fit %>%
  augment()
```

```
# A tsibble: 62,016 x 9 [1Q]
# Key:       Region, State, Purpose, .model [912]
   Region                  State Purpose  .model Quarter Trips   .fitted   .resid   .innov
   <chr>                   <chr> <chr>    <chr>    <qtr> <dbl>     <dbl>    <dbl>    <dbl>
 1 Geelong and the Bellarine VIC   Other    arima  2003 Q4 1.19  -9.36e+ 0 1.05e+ 1 1.05e+ 1
 2 Geelong and the Bellarine VIC   Other    arima  2004 Q4 1.11  -7.15e+ 0 8.26e+ 0 8.26e+ 0
 3 Fraser Coast              QLD   Other    arima  2002 Q4 0     -6.42e- 1 6.42e- 1 6.42e- 1
 4 Upper Yarra               VIC   Business arima  2005 Q1 3.64  -6.24e- 1 4.26e+ 0 4.26e+ 0
 5 Fraser Coast              QLD   Other    arima  2005 Q3 0     -2.07e- 1 2.07e- 1 2.07e- 1
 6 Wimmera                   VIC   Other    arima  2006 Q2 0     -2.81e-18 2.81e-18 2.81e-18
 7 Wimmera                   VIC   Other    arima  2009 Q2 0     -3.98e-20 3.98e-20 3.98e-20
 8 Wimmera                   VIC   Other    arima  2009 Q3 0     -3.48e-20 3.48e-20 3.48e-20
 9 Wimmera                   VIC   Other    arima  2004 Q4 0     -8.74e-52 8.74e-52 8.74e-52
10 Adelaide Hills            SA    Business snaive 1999 Q1 0.670  0.       6.70e- 1 6.70e- 1
# … with 62,006 more rows
```

De la misma forma, se puede realizar un análisis análogo para los otros modelos. Por ejemplo para ETS se puede proporcionar la especificación del modelo, sus parámetros de smoothing y el AIC y BIC correspondiente a cada modelo entrenado:

```
fit %>%
  filter(Region == "Snowy Mountains", Purpose == "Holiday") %>%
  select(ets) %>%
  report()
```

```
Series: Trips
Model: ETS(M,N,A)
  Smoothing parameters:
    alpha = 0.1571013
    gamma = 0.0001000991

  Initial states:
       l        s1       s2        s3        s4
 141.6782 -60.95904 130.8567 -42.23776 -27.65986

  sigma^2:  0.0388

     AIC     AICc      BIC
852.0452 853.6008 868.7194
```

Por último para el Snaive también es posible exportar la especificación del modelo:

```
fit %>%
  filter(Region == "Snowy Mountains", Purpose == "Holiday") %>%
  select(snaive) %>%
  report()
```

```
Series: Trips
Model: SNAIVE

sigma^2: 672.9192
```

Como se observa, gran parte de la paquetería está basada en *fable*, por lo que es relevante compararlo con uno de los paquetes más usados en R: *forecast*

- *fable* ocupa la estructura de tsibble y *forecast* la estructura ts
- *fable* puede manejar múltiples series de tiempo a la vez, mientras que *forecast* solo una.
- *fable* permite entrenar varios modelos a la vez, mientras que *forecast* solo uno
- con *fable* se puede manejar ensemble forecast facilmente, mientras con *forecast* no es posible hacerlo

Ambos paquetes comparten un mismo autor (R. Hyndman), por lo que según otro de los autores del tidyverts, Mitchell O'Hara-Wild: "*fable* es considerada la siguiente iteración del paquete *forecast*, el cual provee las herramientas y extensión necesitada para los retos actuales y futuros de las series de tiempo."

Capítulo 3

Presentación de la solución

En este capítulo introducimos la solución utilizada para resolver el problema y detallamos su aplicación a la base de datos.

3.1. Modelo utilizado

Para resolver el problema de conciliación de pronósticos de series de tiempo jerárquicas se utilizó el paquete de *fable.tools* el cual a través de optimización busca encontrar la solución óptima al problema de reconciliación. Adicional, ejemplificamos como usar un modelo más robusto para series de tiempo que se adecua con el framework de *fable*. Este método se encuentra implementada en una librería llamada *Prophet*. Dicha libreria fue diseñada por Facebook para realizar pronósticos de series de tiempo a gran escala y es de acceso público mediante Python o R.

Prophet permite realizar pronósticos de series de tiempo basado en un modelo aditivo, donde se puede ajustar tendencias lineales o no lineales con múltiples estacionalidades (anual, semanal, diaria). Adicionalmente, al tratarse de un método aditivo, es posible añadir efectos de vacaciones u otras covariables. Adicionalmente tiene la posibilidad de detectar cambios en tendencia a través de puntos de cambio [**?**].

La implementación de *Prophet* está basada en la siguiente fórmula, donde g(t) está asociado a la componente de tendencia, s(t) a la componente de estacionalidad y por ultimo, h(t) al efecto de los dias o periodos feriados, **??** (más un factor de error epsilon).

$$y_t = g(t) + s(t) + h(t) + \epsilon \tag{3.1}$$

El modelo fue comparado contra otras técnicas más populares como el suavizamiento exponencial (ETS) y el autorregresivo integrado de promedio móvil (ARIMA) dando buenos pronósticos inclusive ante cambios bruscos de inclinación en las series de tiempo.

Prophet implementa 2 modelos basados en tendencia: el modelo de crecimiento saturado y el modelo lineal por partes. En nuestro caso, utilizamos *Prophet* como regresión lineal para pronosticar el número de noches apartadas por visitantes domésticos a las diferentes regiones y estados de la base de datos de turismo descrita.

3.2. Implementación

Utilizando las mejores prácticas descritas en [**?**] para el desarrollo de pronósticos en series de tiempo, así como los paquetes mencionados previamente, trabajamos el proyecto de la siguiente manera. Como primer paso, agregamos todas las librerias requeridas para el procesamiento de datos. Desde tidyverse, hasta las correspondientes de tidyverts.

```
# Importar librerias requeridas ------
library(tidyverse)
library(tsibble)
library(fable)
library(tsibbledata)
library(feasts)
library(fable.prophet)
library(lubridate)
```

3.2.1. Generación de TS por agrupación jerárquica

Como segundo paso utilizamos la función *aggregate_key()* cuya finalidad es producir una serie de tiempo agregada según niveles de la forma padre/hijo. De esta manera obtenemos un estructura compatible con el enfoque jerárquico para conciliación de series de tiempo. En nuestro problema seleccionamos las variables de agregación **State** y **Region** generando grupos sobre la variable **Purpose**.

```
# Agregar niveles jerarquicos a la base de turismo ------
tourism_aggregated <-
  tourism %>%
  aggregate_key((State / Region) * Purpose, Trips = sum(Trips))
```

```
> tourism_aggregated
# A tsibble: 34,000 x 5 [1Q]
# Key:       State, Purpose, Region [425]
   Quarter State        Purpose      Region        Trips
     <qtr> <chr>        <chr>        <chr>         <dbl>
 1 1998 Q1 <aggregated> <aggregated> <aggregated> 23182.
 2 1998 Q2 <aggregated> <aggregated> <aggregated> 20323.
 3 1998 Q3 <aggregated> <aggregated> <aggregated> 19827.
 4 1998 Q4 <aggregated> <aggregated> <aggregated> 20830.
 5 1999 Q1 <aggregated> <aggregated> <aggregated> 22087.
 6 1999 Q2 <aggregated> <aggregated> <aggregated> 21458.
 7 1999 Q3 <aggregated> <aggregated> <aggregated> 19914.
 8 1999 Q4 <aggregated> <aggregated> <aggregated> 20028.
 9 2000 Q1 <aggregated> <aggregated> <aggregated> 22339.
10 2000 Q2 <aggregated> <aggregated> <aggregated> 19941.
# … with 33,990 more rows
```

Para entender mejor qué realiza esta función, se va a explicar a detalle la salida mostrada. En total se muestran 34,000 registros los cuales podemos detallar en la siguiente tabla:

Estado	Propósito	Región	Combin	Registros
Aggregated	Aggregated	agreggated	1	80
Estado	Aggregated	agreggated	8	640
Aggregated	Propósito	agreggated	4	320
Estado	Aggregated	Región	76	6,080
Estado	Propósito	Aggregated	32	2,560
Estado	Propósito	Región	304	24,320
Total			425	34,000

Cuadro 3.1: Número total de combinaciones considerando la jerarquía

Algo importante de mencionar es que el total de combinaciones de no haber considerado las jerarquías, hubiera incrementado en gran número. Esto debido a que el introducir una jerarquía se vuelve una restricción al número de combinaciones. Simplemente, si se consideraran las combinaciones de estado (8), propósito (4) y región (76) hubieran resultado en 2,432 combinaciones, que multiplicadas por los 80 registros, se hubiera tenido más de 190,000 registros.

3.2.2. Train - Test split

Después, separamos los datos en una muestra de entrenamiento y otra de prueba mediante un filtro con la función *yearquarter()* con un punto de corte de 2015 Q1 para poder calcular el accuracy de los modelos. Basados en esto, ajustamos nuestros datos de entrenamiento con modelos clásicos de predicción de series de tiempo como el ETS y ARIMA implementados por las funciones *ETS()* y *ARIMA()* respectivamente. Estos modelos nos servirán de benchmark para evaluar el desempeño del nuestro. Adicionalmente, ajustamos nuestro modelo lineal indicando como variable responsiva el número de noches y como variable predictora el componente estacionario anual. También calculamos un pronóstico combinado al promediar los 3 modelos descritos (*ETS, ARIMA, Prophet*).

```
# Filtrar los datos para poder calcular el accuracy y realizar el fit de los modelos definidos
tourism_fit <-
  tourism_aggregated %>%
  filter(Quarter < yearquarter("2015 Q1")) %>%
  model(
    ets_n = ETS(Trips ~ trend("N")), # benchmark
    arima = ARIMA(Trips), # benchmark
    prophet = fable.prophet::prophet(Trips ~ season("year"))
  ) %>%
  mutate(combn = (ets_n + arima + prophet)/3)
```

NOTA: es importante considerar que para un adecuado calculo del error no solo hay que considerar una partición de train-test, ya que puede suceder el caso donde el nivel de error depende de la seasonality de la serie de tiempo. En este texto por simplicidad solamente se experimentó con conjuntos de train-test

3.2.3. Reconciliación de Forecast Jerárquicos

Dado que nuestro problema se origina con las series de tiempo jerárquicas, debemos ahora realizar el paso de conciliación para obtener pronósticos coherentes. Esto se logra con la función *reconcile()*, la cual espera el método elegido para conciliar. Nosotros trabajamos con OLS que es un método de combinación óptima el cual calcula la suma ponderada de las predicciones de los nodos del modelo jerárquico. (De acuerdo a Hyndman, el uso de estos métodos depende de la forma de la matriz Σ_h, ya que es dificil de estimar con $h > 1$. Las soluciones que propone son:)

- **OLS:** Ignorar Σ_h
- **WLS:** Asumir que $\Sigma_h = k_h\Sigma_1$ es diagonal

El alcance de este proyecto considera la alternativa más sencilla, es decir ignorar Σ_h, por lo que utilizamos OLS:

```
# Reconciliacion de forecast
tourism_fc_reconciled <-
  tourism_fit %>%
  reconcile(coherent = min_trace(combn, method = "ols")) %>%
  forecast(h = "3 years")
```

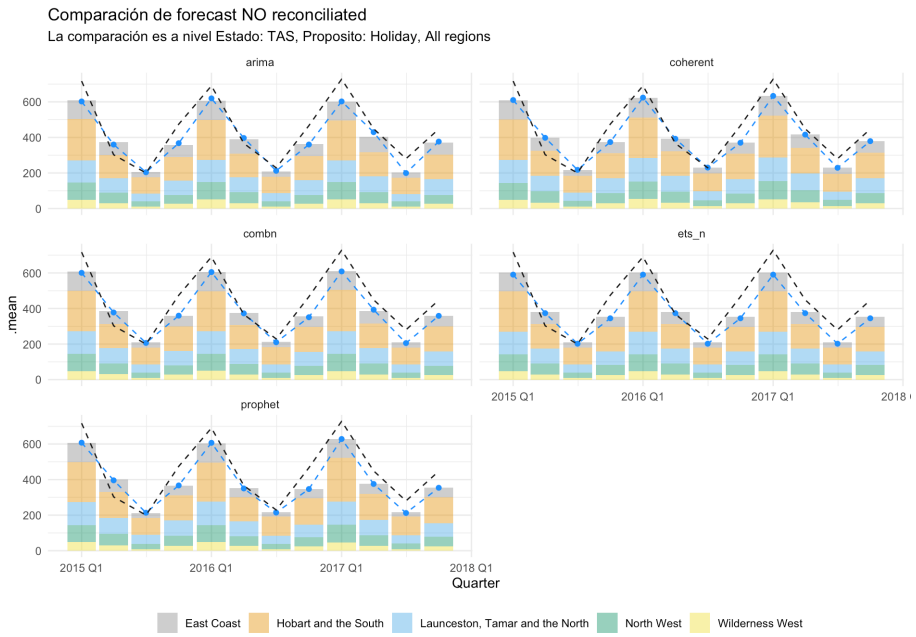
```
# A fable: 25,500 x 7 [1Q]
# Key:     State, Purpose, Region, .model [2,125]
   State Purpose  Region   .model Quarter        Trips .mean
   <chr> <chr>    <chr>    <chr>    <qtr>       <dist> <dbl>
 1 ACT   Business Canberra ets_n  2015 Q1  N(113, 620)  113.
 2 ACT   Business Canberra ets_n  2015 Q2 N(158, 1221)  158.
 3 ACT   Business Canberra ets_n  2015 Q3 N(151, 1112)  151.
 4 ACT   Business Canberra ets_n  2015 Q4 N(148, 1072)  148.
 5 ACT   Business Canberra ets_n  2016 Q1  N(113, 620)  113.
 6 ACT   Business Canberra ets_n  2016 Q2 N(158, 1221)  158.
 7 ACT   Business Canberra ets_n  2016 Q3 N(151, 1112)  151.
 8 ACT   Business Canberra ets_n  2016 Q4 N(148, 1072)  148.
 9 ACT   Business Canberra ets_n  2017 Q1  N(113, 620)  113.
10 ACT   Business Canberra ets_n  2017 Q2 N(158, 1221)  158.
# … with 25,490 more rows
```

En la salida anterior, se puede observar que hay 25,500 registros. Como la predicción fue por 3 años, esto implica que cada combinación aparecerá 12 veces (es trimestral). Además, aparecerá 5 veces debido a cada repeticion por modelo (ets, arima, prophet, el promedio de forecast y una extra que es el reconciled). Es decir multiplicamos por 60 cada combinacion. De esta manera podemos presentar la información previa como la tabla **??**:

Estado	Propósito	Región	Combin	Registros
Aggregated	Aggregated	agreggated	1	60
Estado	Aggregated	agreggated	8	480
Aggregated	Propósito	agreggated	4	240
Estado	Aggregated	Región	76	4,560
Estado	Propósito	Aggregated	32	1,920
Estado	Propósito	Región	304	18,240
Total			425	25,500

Cuadro 3.2: Número total de predicciones considerando la jerarquía

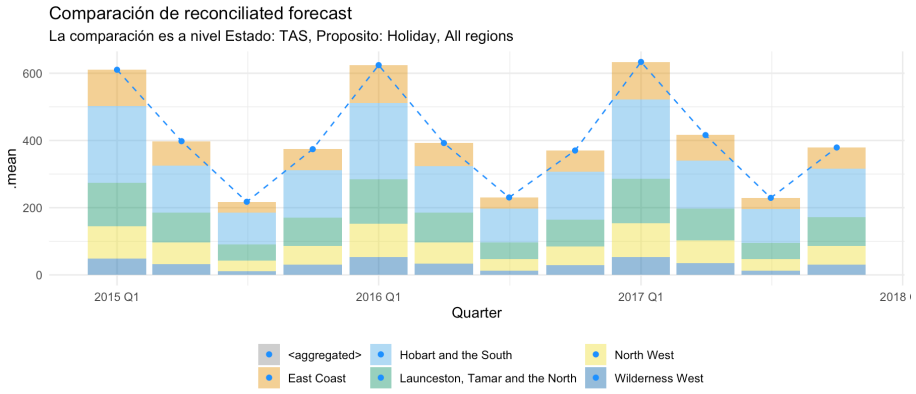
En la siguiente gráfica, presentamos los pronostcos obtenidos. Por simplicidad, solamente tomamos un Estado y un Propósito de viaje: "TAS" (Tasmania), "Holiday". En la gráfica se muestra en una linea negra el dato real, en linea azul el dato agregado y en barras el detalle del forecast para cada región. En particular en este estado y propósito se ve una clara estacionalidad en los trimestres. En cada Facet aparece un método distinto, además del coherente:



La siguiente tabla muestra el detalle para algunas observaciones de la serie anterior. Se puede observar como no existe una coherencia entre el dato agregado y el desagregado.

	State	Purpose	Quarter	Desagregado	Agregado
	<chr>	<chr>	<qtr>	<dbl>	<dbl>
1	TAS	Holiday	2015 Q1	591.	603.
2	TAS	Holiday	2015 Q2	374.	380.
3	TAS	Holiday	2015 Q3	202.	211.
4	TAS	Holiday	2015 Q4	345.	354.
5	TAS	Holiday	2016 Q1	591.	603.
6	TAS	Holiday	2016 Q2	374.	380.
7	TAS	Holiday	2016 Q3	202.	211.

Por otra parte, cuando lo comparamos con el modelo coherent (en este caso aplicamos la función reconcile al modelo combinado) se observa que los datos son coherentes entre los niveles de agregación.




```
# A tibble: 12 x 5
```

	State	Purpose	Quarter	Desagregado	Agregado
	<chr>	<chr>	<qtr>	<dbl>	<dbl>
1	TAS	Holiday	2015 Q1	610.	610.
2	TAS	Holiday	2015 Q2	398.	398.
3	TAS	Holiday	2015 Q3	218.	218.
4	TAS	Holiday	2015 Q4	374.	374.
5	TAS	Holiday	2016 Q1	624.	624.
6	TAS	Holiday	2016 Q2	392.	392.
7	TAS	Holiday	2016 Q3	230.	230.

Algo interesante del método es que se puede realizar la conciliación de cualquier método que se proponga. En la tabla y gráficas mostradas solamente se realizó conciliación del método combinado (combn).

3.2.4. Cálculo de accuracy

Una vez realizada la conciliación se procede a la fase de cálculo del accuracy, que con el framework de *Tidyverts* también se realiza de manera sencilla. Así, podemos mostrar los modelos de acuerdo a las distintas métricas consideradas:

```
# Calculo del accuracy para cada modelo
tourism_fc_reconciled %>%
  accuracy(tourism_aggregated) %>%
  group_by(.model) %>%
  summarise_at(vars(ME:ACF1), median) %>%
  arrange(MASE)
```

$$\text{MASE} = \frac{1}{h}\sum_{t=T+1}^{T+h}\left|\frac{e_t}{\text{scale}}\right|, \qquad \text{scale} = \frac{1}{T-m}\sum_{t=m+1}^{T}|y_t - y_{t-m}|$$

Modelo	RMSE	MAE	MASE
ARIMA	16.9	13.9	0.906
ETS	15.9	13.1	0.853
Prophet	17.2	14.1	0.950
Combinado	16.4	13.6	0.896
Conciliado	15.0	12.4	0.872

Cuadro 3.3: Comparación de la evaluación de los modelos utilizados. La conciliación se hizo tomando como base el modelo combinado. Se presentan las métricas de RMSE, MAE, MASE (aunque se pueden utilizar otras)

3.3. Resultados

En la tabla **??** presentamos los resultados obtenidos donde apreciamos que nuestro modelo lineal con *Prophet* se desempeñó bastante similar a los modelos dedicados. Cabe destacar que la combinación de modelos obtuvo los mejores resultados en la mayoría de las métricas. Por ello, la conciliación de las predicciones se realizó tomando como base el modelo combinado.

Dado que estamos trabajando con mismas unidades en todas las series, podemos usar la columna de MAE para comparar. Observamos que el suavizamiento exponencial dio el mejor ajuste, en contraste, nuestro modelo con *Prophet* tiene un punto de diferencia. En el mismo sentido, nuestro modelo se equivoca en promedio en la misma magnitud que el de ARIMA.

Por lo tanto, podemos decir que la implementación con ML del pronóstico de datos de series de tiempo es comparable con los métodos clásicos de estadística para este problema. Asimismo, en cuanto a la conciliación de las predicciones de la jerarquía, podemos ver que el acercamiento combinado mejora el desempeño con respecto a sus versiones individuales.

Capítulo 4

Conclusiones

En suma, podemos concluir que la aplicación de técnicas de ML al pronóstico de series de tiempo es apenas comparable con las técnicas populares de estadística. No obstante, destacamos que paquetes como *Prophet* facilitan mucho la predicción en este tipo de datos, pues permiten al analista aplicar más de su conocimiento de dominio que del conocimiento técnico de series de tiempo.

Adicionalmente, destacar que la combinación de modelos fue superior en este problema con respecto a cualquiera de los modelos propuestos. Lo anterior nos recuerda que en ML no tenemos siempre un modelo superior para todos los problemas. Aprovechando el punto anterior, hemos logrado conciliar los pronósticos de la serie jerárquica mejorando el desempeño situación que no se consigue con las técnicas básicas como "arriba-abajo", "abajo-arriba" y "punto medio".

Como era de esperarse, y consultando la popular competencia M4, modelos de ML puros son aún inferiores a los benchmarks comunes como NAIVE, SNAIVE, ETS y ARIMA [**?**]. De ahí que el trabajo futuro comprenda utilizar las ideas de *Prophet* junto con otros métodos para explorar más el poder predictivo de combinación de modelos más allá del promedio de sus resultados individuales.