Código del Modelo

Analítica

2020-03-25

## R Markdown

Este documento fue generado a través de R Markdown un paquete de formatos que nos permite compartir información de sintaxis, utilizando HTML, PDF y en este caso documentos de Microsoft Word, el propósito de este capítulo, es compartir nuestra investigación sobre el código del modelo y cuáles son las principales funciones que hemos utilizado y su funcionamiento, junto con ejemplos prácticos para la interpretación del usuario. Si desea saber más sobre el paquete en cuestón visite <http://rmarkdown.rstudio.com>.

En caso de que este observando este código directamente desde la carpeta de contenidos de la tesis, y no en el documento final, tenga en cuenta que al presionar el boton **Knit** en el borde superior del IDE Rstudio podrá generar de manera automatica el documento en Word.

El código del modelo, será presentado en cuadros de outputs, que exhibiran el resultado final, así como comentarios y referencias sobre la fuente de los datos y su significado e importancia para el modelo.

Tener en cuenta que el modelo, este documento y su respectivo instrumento ó aplicativo, utilizaran los siguientes paquetes, de los cuáles podrá acceder a su documentación oficial con los links disponibles en la última página de este documento:

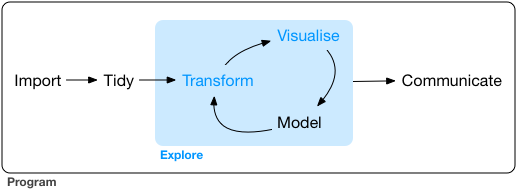
library(tidyverse) # Integración de Recursos para Data Science.   
library(lubridate) # Configuración de strings de fechas.   
library(kableExtra) # Creación de Tablas tipo HTML.  
library(knitr) # Generación de Informes dínamicos.  
library(tidyquant) # Integración para analísis Financiero.  
library(xml2) # Conversión de fragmentos a HTML  
library(timetk) # Herramienta para trabajar Series de Tiempo.  
library(corrplot) # Librería para dibujar matrices de correlación.  
library(alphavantager) # API para importar datos de Mercado.  
library(plotly) # Herramienta de Visualización interactiva.  
library(prophet) # Analísis de series temporales.

Como puede apreciarse en la documentación, cada paquete contiene pluralidad de funciones, por lo cual la lectura de la documentación es clave para aquellos que deseen un mejor entendimiento de la dinámica expuesta.

## Logíca General

Para explorar el código seguiremos la logica del modelo general, para analítica, expuesto previamente en el capítulo sobre el paradigma del Data Science y su importancia para los procesos de toma de decisiones.

Este modelo general ha sido extraido del libro R for Data Science, un esfuerzo colaborativo, disponible online :  
acceda al libro a través de este link: <https://r4ds.had.co.nz/explore-intro.html>.



A caption

Es importante tener en cuenta que el conjunto de heramientas, armoniza en su filosofia Tidy, que esencialmente significa, que los conjuntos de datos, tienen una estructura basica, donde cada columna representa una variable y cada fila una observación, no todos los outputs de las funciones utilizadas, tiene la estructura deseada, pero gracias a las herramientas seleccionadas, su manipulación puede ser automatizada, con el fin de que cumplan con la arquitectura base. para explorar más sobre esta “filosofia” del Data Science invitamos a la siguiente lectura del Journal of Statistical Software: <“<https://vita.had.co.nz/papers/tidy-data.pdf>”>

## Importación de Datos:

Inicialmente se importaran los datos, estos serán importados directamente de la Reserva Fededral de St. Louis, los datos pueden explorarse en la siguiente página web: <https://fred.stlouisfed.org/> donde cada activo o indicador economico, tiene un ticket que puede se encontrado buscando el nombre general. Sín embargo para automatizar la descarga necesitamos conocer el nombre tecnico e introducirlo como argumento de la función de importación que utiliaremos, funcion **(“tq\_get”)** dicha funcion posee tres argumentos principales, la variable **(x)** que se refiere al activo ó indicador a analizar, para que sea interpretable por la función, usar el ticket respectivo, el argumento “get” hace referencia al api o lugar desde el cual, se pueden importar los datos, en este caso seleccionamos la opcion **“economic.data”** puesto que nos conecta con el API de la FRED y finalmente un rango de fechas donde: **“from”** es el momento anterior y **“to”** el posterior. Se ha seleccionado este paquete por su flexibilidad y por la calidad, prestigio e importancia de la institucion que los comparte

### Ejercicio de Importación:

Importar la medición sobre los niveles de incertidumbre…

EPU <- tq\_get("USEPUINDXD", get = "economic.data", from = "2017-01-01")  
  
EPU <- EPU %>% arrange(desc(date))  
  
EPU

## # A tibble: 1,179 x 3  
## symbol date price  
## <chr> <date> <dbl>  
## 1 USEPUINDXD 2020-03-24 690.  
## 2 USEPUINDXD 2020-03-23 621.  
## 3 USEPUINDXD 2020-03-22 666.  
## 4 USEPUINDXD 2020-03-21 555.  
## 5 USEPUINDXD 2020-03-20 632.  
## 6 USEPUINDXD 2020-03-19 527.  
## 7 USEPUINDXD 2020-03-18 465.  
## 8 USEPUINDXD 2020-03-17 383.  
## 9 USEPUINDXD 2020-03-16 329.  
## 10 USEPUINDXD 2020-03-15 475.  
## # ... with 1,169 more rows

# Aplicación sencilla de la función.

El ejemplo presentado, arroja como output un tibble o data frame, que contiene el símbolo importado, la fecha y su precio en el momento observado, se debe tener en cuenta que en este caso especifico, no se trata de un precio sino de un indicador de incertidumbre económica, El indicador en cuestión, es conocido como **“EPU”** cuyas siglas corresponden a (ECONOMIC POLICY UNCERTAINTY), este indicador fue creado por el **NBER** (NATIONAL BUREAU OF ECONOMIC RESEARCH), el indicador fue construido para medir los niveles de incertidumbre de USA a través de proxies que identifican y calculan la frecuencia con la que se hace mención a la incertidumbre respecto a las politicas públicas económicas de impacto global, en los principales periodicos del país utilizando datos de primera calidad, para explorar más sobre este indicador específico, los remitimos a la documentación oficial de la institución creadora, y a la serie temporal disponible en **FRED.**

**NBER**, Measuring Economic Policy Uncertainty : <http://www.policyuncertainty.com/media/BakerBloomDavis.pdf>

**EPU** , FRED time series for Economic Policy Uncertainty: <https://fred.stlouisfed.org/series/USEPUINDXD/>

Se debe tener en cuenta que la variable “EPU” gracias al asignador del código, ya contiene los datos importados, es decir que para realizar cualquier tipo de modificación ó transformación sobre los datos importados, basta con llamar la variable u objeto R y anexar los cambios ó transformaciones deseadas, es por esta razón, en el ejemplo planteado, después de asignar la variable “EPU” se anexo un ordenamiento que nos permite observar los datos más recientes en el tiempo, donde el operador **(“%>%”)** traduce “AND THEN” en este caso seguido de **“arrange(desc(date))”**, sintaxis que logra ordenar la data de forma tal que las fechas se ordenen de mayor a menor para lograr observar los puntos más recientes en el tiempo.

## Visualización:

Para visualizar utilizaremos el paquete plotly, este paquete permite al usuario interactuar con el objeto representado, siempre y cuando este en un ambiente que permita dicha interaccion de lo contrario, solo cumplirá el proposito visual, para este ejemplo, visualizaremos la variable importada, **EPU** utilizando el paquete **ggplot2**, esta herramienta nos permite explicar a la funcion con facilidad, cuales con las variables a tener en cuenta para graficar en el plano cartesiano, en nuestro caso, el eje horizontal sera el tiempo y el verticar, el nivel del indicador.

Ademas de graficar la evolucion en el tiempo de la variable **EPU**, se trazará una regresion lineal simple **(orange)**.

### **EPU**

(Economic Policy Uncertainty, codigo para graficar ) :

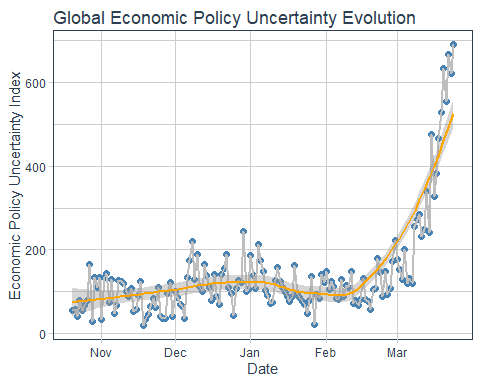
# A ggplot of the Global Economic Policy Uncertainty :   
uncertainty\_index <-   
# ggplotly(  
 EPU %>%  
 filter(date >= "2019-10-20") %>%   
 ggplot(aes(x = date, y = price))+  
 theme\_tq()+  
 geom\_point(color = "steelblue", size = 2)+  
 geom\_line(color = "gray", size = 1)+  
 geom\_smooth(color = "orange", method = loess)+  
 labs(x = "Date", y = "Economic Policy Uncertainty Index",  
 title = "Global Economic Policy Uncertainty Evolution")   
# )  
  
# Se puede apreciar de forma muy intuitiva, a que corresponde cada variable y color de la funcion ggplot,  
# ggplot no necesita ser importado puesto que es contenido por tidyverse.

### **EPU**

(Economic Policy Uncertainty, Representación Visual) :

uncertainty\_index

## `geom\_smooth()` using formula 'y ~ x'



# Se observa un incremento considerable en los niveles de incertidumbre, producto de la coyuntuta actual, que a la fecha de ejecucion, de este reporte dinamico, es el COVID-19  
  
Sys.Date()

## [1] "2020-03-25"

## Correlación

Hasta ahora, un codigo tan simple ya es capaz de añadir valor a nuestro analísis, las razones por las que añade valor son varias, pero conviene destacar la calidad de la fuente primaria, la automatización del proceso y la calidad de la grafica, es de tener en cuenta que este documento en Word fue credo directamente desde Rstudio con relativamente poco esfuerzo, Word es un ambiente no reactivo, es decir que no cambia de manera dinamica, sin embargo el reporte cada vez que se ejecuta en el **IDE** de **R** intera en cada uno de sus pasos, y nos brinda informacion nueva de alta calidad y valor, ahora bien, solo hemos importado datos de un indicador específico y hemos graficado el comportamiento de dicho indicador **(EPU)**, este indicador es poderoso, pero las variables aisladas son poco dicientes, de manera de ejemplo, observemos como se relaciona esta variable con otras variables de mercado que son importantes para percibir los riesgos de mercado en los escenarios Mark to Market. Ya esta claro el proceso de importacion de datos, es necesario escoger otras variables de mercado para ver como se relacionan entre si, para ello se han seleccionado, el **DOW JONES, SP500, VIX, WTI, Treasury\_10y, CPI y Unemployment Rate** El **Dow Jones** y **SP500** represtaran el comportamiento de la renta variable, al ser dos de los indices mas importantes de la economia, El **VIX** mide los niveles de volatilidad conocido por sus siglas como **(Exchange Market Volatility Inde)** indice calculado por **Chicago Board Options**, el **Treasury\_10y** se refiere a los bonos del Tesoro de Estados Unidos a 10 años, es decir un horizonte largoplacista, escogido para este ejemplo simplificado como indicador de la renta fija. Los ultimos dos seleccionados **CPI** y **Unemployment Rate** son seleccionados como indicadores economicos de relevancia, el **Consume Price Index** es un indicador que da pistas sobre el comportamiento inflacionario, una economia que crece de manera sana, tiene niveles de inflacion controlados, al rededor del 2% porque con esto se presume que hay una demanda estable de bienes y servicios, gran parte de la demanda esta justificada por los niveles de ingreso de los ciudadanos, que a grosso modo es muy influido en promedio, por los niveles de empleo **UNRATE**, es decir que a niveles altos de empleo, lo normal es que se estimule el consumo, lo que a su vez estimula la economia.

Con niveles de incertidumbre tan altos vale la pena observar estas correlaciónes junto al **USD** que mediremos utilizando el indice **DXY CUR** que es un indicador del valor relativo del dolar americano respecto a otras fiat de gran importancia en la economia global.

Para cumplir dicho objetivo el primer paso es la importación de los datos de interés que a partir del ejemplo anterior, es generalizable:

### Importación de Datos de Mercado:

market\_data <- tq\_get(c("DJIA", "SP500", "VIXCLS", "DTWEXAFEGS",  
 "UNRATE", "CPIAUCSL", "DGS10"),  
 get = "economic.data", from = "2010-01-01")  
  
market\_data

## # A tibble: 13,464 x 3  
## symbol date price  
## <chr> <date> <dbl>  
## 1 DJIA 2010-03-26 10850.  
## 2 DJIA 2010-03-29 10896.  
## 3 DJIA 2010-03-30 10907.  
## 4 DJIA 2010-03-31 10857.  
## 5 DJIA 2010-04-01 10927.  
## 6 DJIA 2010-04-02 NA   
## 7 DJIA 2010-04-05 10974.  
## 8 DJIA 2010-04-06 10970.  
## 9 DJIA 2010-04-07 10898.  
## 10 DJIA 2010-04-08 10927.  
## # ... with 13,454 more rows

# Datos de las variables seleccionadas desde el año 2010, "post Sub-prime Crisis" en el 2008  
# Los datos están representados de forma Tidy.

### Transformacion:

Con el fin de calcular la correlación de **Pearson**, necesitamos transformar los datos, es decir transponer por variable, para crear una matriz que nos permita crear la matriz de correlación, para ello operaremos sobre el objeto que ya contiene la información, los datos no se generan con la misma frecuencia, es decir los indicadores economicos se publican cada semana o pueden tener una temporalidad incluso mensual, sin embargo los precios de los activos, si que cotizan en el dia a dia, por lo que habrá momentos en el tiempo en los que no existan todos los datos, para solucionar el problema, asumiremos que los datos de baja frecuencia de publicación, permanencen constantes hasta que son actualizados, ademas la matriz no debe contemplar fechas, puesto que no es un valor al que vayamos a aplicarle la correlación, ya es sabido por el primer paso la franja temporal analizada.

market\_matrix <- market\_data %>%  
 pivot\_wider(names\_from = symbol, values\_from = price) %>%   
 na.locf() %>%   
 select(-date)   
  
market\_matrix

## # A tibble: 2,700 x 7  
## DJIA SP500 VIXCLS DTWEXAFEGS UNRATE CPIAUCSL DGS10  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 10927. 1178. 17.5 89.7 9.9 217. 3.89  
## 2 10927. 1178. 17.5 90.2 9.9 217. 3.96  
## 3 10974. 1187. 17.0 89.9 9.9 217. 4.01  
## 4 10970. 1189. 16.2 90.1 9.9 217. 3.98  
## 5 10898. 1182. 16.6 90.1 9.9 217. 3.89  
## 6 10927. 1186. 16.5 90.0 9.9 217. 3.91  
## 7 10997. 1194. 16.1 89.8 9.9 217. 3.9   
## 8 11006. 1196. 15.6 89.4 9.9 217. 3.87  
## 9 11019. 1197. 16.2 89.3 9.9 217. 3.84  
## 10 11123. 1211. 15.6 88.9 9.9 217. 3.88  
## # ... with 2,690 more rows

# Con la matriz de precios depurada, el siguiente paso es correlar.

### Matriz

Para correlaciónar la matriz de precios historicos, solo hemos de aplicar la funcion cor(), se debe tener en cuenta que el coeficiente de correlación solo indica la direccion y fuerza de la correlación, pero no es indicador alguno de causalidad. la matriz de correlación calculada, ofrece mucho valor, sin embargo no es de interpretación intuitiva.

Para solucionar este problema, y que el esfuerzo por interpretar el resultado sea menor, inmediatamente se explicara como crear una visualización y clusterizado eficiente y de facil ejecución.

corr\_matrix <- market\_matrix %>% cor()  
  
  
corr\_matrix %>% print()

## DJIA SP500 VIXCLS DTWEXAFEGS UNRATE CPIAUCSL  
## DJIA 1.00000000 0.99318960 -0.31429240 0.7529022 -0.8604605 0.9014971  
## SP500 0.99318960 1.00000000 -0.32409536 0.7835468 -0.8747203 0.8926994  
## VIXCLS -0.31429240 -0.32409536 1.00000000 -0.2117717 0.3296169 -0.2699850  
## DTWEXAFEGS 0.75290221 0.78354675 -0.21177170 1.0000000 -0.8879635 0.7836469  
## UNRATE -0.86046053 -0.87472026 0.32961692 -0.8879635 1.0000000 -0.9547005  
## CPIAUCSL 0.90149712 0.89269938 -0.26998495 0.7836469 -0.9547005 1.0000000  
## DGS10 -0.04892799 -0.06509207 -0.03328498 -0.2964059 0.2952094 -0.2954485  
## DGS10  
## DJIA -0.04892799  
## SP500 -0.06509207  
## VIXCLS -0.03328498  
## DTWEXAFEGS -0.29640589  
## UNRATE 0.29520942  
## CPIAUCSL -0.29544854  
## DGS10 1.00000000

#Como resultado podemos observar el coeficiente de correlación de los datos historicos de las variables de interes.

Ademas de que la matriz de correlación tiene las limitaciones previamente expresadas, tambien se debe tener en cuenta que el coeficiente presentado, no cambia con mucha facilidad, es decir esta calculado con base a información historica, pero la correlación puede ser un fenomeno más dinámico, es decir si bién el coefiente de correlación siempre se encuenta en el rango **[-1 : 1]** es posible que cambie en el corto y mediano plazo, es decir que la correlación oscile en el paso del tiempo.

El coeficiente de correlación, se interpreta como:

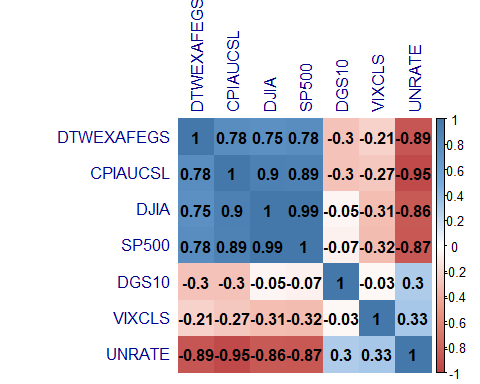
1 = Perfecta Correlación Positiva.  
 -1 = Perfecta Correlación Negativa.

Cuando la correlación es **1**, las variables analizadas se comportan la mayor parte del tiempo, en el mismo sentido, dirección y fuerza cuando el valor es **-1**, las variables se comportan en sentido y dirección contraria, con la misma intensidad o fuerza.

### Visualización de Matriz de Correlación:

Se puede apreciar, que los coeficientes de correlación, han sido ordenados, siguiendo un patrón de agrupamiento jerárquico, esto con el fin de mejorar la interpretabilidad.

col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))  
  
  
matrix\_corrplot <-   
 corrplot(corr\_matrix,  
 method = "color",  
 col = col(200),  
 addCoef.col = "black",  
 tl.col = "darkblue",  
 order = "hclust")



Entonces bien, conocer en que sentido, dirección y con que intensidad se relacionan las variables de mercado es bastante interesante, pero que tal seria observar el comportamiento de esta relacion en el transcurso del tiempo ? para se puede aplicar la misma funcion de correlacion en distintas ventanas de tiempo, esta tecnica se conoce como **“Rolling Window”**

Claro esta que al ser dinamica seria muy laborioso visualizarla como una matriz, visualicemos una linea de evolución en el tiempo, despues de calcular esta correlacion dinamica de las mismas variables en cuestion pero respecto el **DXY CUR** como referencia, es decir si bien ya se tiene respuesta a la pregunta:

¿Como se relacionan la variables entre sí ?

A continuación buscaremos respuesta a la pregunta:

¿Como evoluciona la relacion de las variables en el tiempo?

Se debe tener en cuenta que esta evolucion en el tiempo, sera respetando el rango permitido de correlación, es decir el máximo valor posible en el tiempo sera **+1** y el mínimo posible **-1**

### Correlación Dinámica:

A diferencia de la matrix anterior, en ese ejercicio, se correlaciona las variaciones del precio no el precio como tal, es decir el retorno del activo o variación porcentual, el cociente entre el valor en el momento 1 y anterior.

# Retorno o variacion de las variables seleccionadas.  
  
market\_daily\_returns <- market\_data %>%   
 group\_by(symbol) %>%   
 tq\_transmute(select = price,  
 mutate\_fun = periodReturn,  
 period = "daily")  
  
market\_daily\_returns

## # A tibble: 12,946 x 3  
## # Groups: symbol [7]  
## symbol date daily.returns  
## <chr> <date> <dbl>  
## 1 DJIA 2010-03-26 0   
## 2 DJIA 2010-03-29 0.00419   
## 3 DJIA 2010-03-30 0.00106   
## 4 DJIA 2010-03-31 -0.00466   
## 5 DJIA 2010-04-01 0.00649   
## 6 DJIA 2010-04-05 0.00425   
## 7 DJIA 2010-04-06 -0.000324  
## 8 DJIA 2010-04-07 -0.00661   
## 9 DJIA 2010-04-08 0.00271   
## 10 DJIA 2010-04-09 0.00643   
## # ... with 12,936 more rows

# Retornos del dolar, DXY CUR, valor base "baseline"  
baseline\_daily <- "DTWEXAFEGS" %>%  
 tq\_get(get = "economic.data",  
 from = "2010-01-01",  
 to = Sys.Date()) %>%  
 tq\_transmute(select = price,  
 mutate\_fun = periodReturn,  
 period = "daily")  
  
baseline\_daily

## # A tibble: 2,539 x 2  
## date daily.returns  
## <date> <dbl>  
## 1 2010-01-04 NA   
## 2 2010-01-05 -0.000654   
## 3 2010-01-06 0.0000706  
## 4 2010-01-07 0.00509   
## 5 2010-01-08 -0.00309   
## 6 2010-01-11 -0.00848   
## 7 2010-01-12 0.000511   
## 8 2010-01-13 -0.000678   
## 9 2010-01-14 -0.00244   
## 10 2010-01-15 0.00401   
## # ... with 2,529 more rows

A continuación, una sola matriz que contenga ambos conjuntos de datos, significa que la correlación será de cada una de las variables respecto al Baseline que en este caso es: **DXY CUR**, para observar la correlación de dichas variables respecto al dollar en el transcurso del tiempo.

returns\_joined <- left\_join(market\_daily\_returns,  
 baseline\_daily,  
 by = "date")%>%   
 na.omit()  
  
returns\_joined

## # A tibble: 12,705 x 4  
## # Groups: symbol [7]  
## symbol date daily.returns.x daily.returns.y  
## <chr> <date> <dbl> <dbl>  
## 1 DJIA 2010-03-26 0 0.00125   
## 2 DJIA 2010-03-29 0.00419 -0.00500   
## 3 DJIA 2010-03-30 0.00106 0.000739  
## 4 DJIA 2010-03-31 -0.00466 -0.00462   
## 5 DJIA 2010-04-01 0.00649 -0.00377   
## 6 DJIA 2010-04-05 0.00425 -0.00369   
## 7 DJIA 2010-04-06 -0.000324 0.00207   
## 8 DJIA 2010-04-07 -0.00661 0.000712  
## 9 DJIA 2010-04-08 0.00271 -0.000965  
## 10 DJIA 2010-04-09 0.00643 -0.00262   
## # ... with 12,695 more rows

# Matriz unificada para correlar

El siguiente paso es aplicar la formula de correlación a las variables ya ordenadas en el objeto anterior, para ello utilizaremos la funcion tq\_transmute de tidyquant y la aplicaremos a ambas variables o columnas, la base : **DXY CUR** y el conjunto de variables.

#Calculate the correlation.   
  
rolling\_corr <- returns\_joined %>%  
 tq\_transmute\_xy(x = daily.returns.x,  
 y = daily.returns.y,  
 mutate\_fun = runCor,  
 n = 6,  
 col\_rename = "rolling.corr")  
  
rolling\_corr %>%   
 print(n = 25)

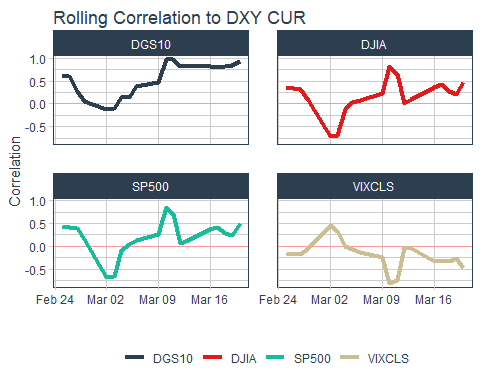
## # A tibble: 12,705 x 3  
## # Groups: symbol [7]  
## symbol date rolling.corr  
## <chr> <dttm> <dbl>  
## 1 DJIA 2010-03-26 00:00:00 NA   
## 2 DJIA 2010-03-29 00:00:00 NA   
## 3 DJIA 2010-03-30 00:00:00 NA   
## 4 DJIA 2010-03-31 00:00:00 NA   
## 5 DJIA 2010-04-01 00:00:00 NA   
## 6 DJIA 2010-04-05 00:00:00 -0.187  
## 7 DJIA 2010-04-06 00:00:00 -0.220  
## 8 DJIA 2010-04-07 00:00:00 -0.313  
## 9 DJIA 2010-04-08 00:00:00 -0.333  
## 10 DJIA 2010-04-09 00:00:00 -0.784  
## 11 DJIA 2010-04-12 00:00:00 -0.561  
## 12 DJIA 2010-04-13 00:00:00 -0.490  
## 13 DJIA 2010-04-14 00:00:00 -0.704  
## 14 DJIA 2010-04-15 00:00:00 -0.416  
## 15 DJIA 2010-04-16 00:00:00 -0.648  
## 16 DJIA 2010-04-19 00:00:00 -0.429  
## 17 DJIA 2010-04-20 00:00:00 -0.456  
## 18 DJIA 2010-04-21 00:00:00 -0.467  
## 19 DJIA 2010-04-22 00:00:00 -0.226  
## 20 DJIA 2010-04-23 00:00:00 -0.336  
## 21 DJIA 2010-04-26 00:00:00 0.214  
## 22 DJIA 2010-04-27 00:00:00 -0.737  
## 23 DJIA 2010-04-28 00:00:00 -0.755  
## 24 DJIA 2010-04-29 00:00:00 -0.790  
## 25 DJIA 2010-04-30 00:00:00 -0.656  
## # ... with 1.268e+04 more rows

#Observese que los valores "NA" corresponden a la ventana de tiempo contemplada\*

### Visualización de Correlación Dinámica:

A continuación, despues de haber calculado la correlación, el ultimo paso para este analísis, es la visualización, para ello utilizaremos lineas y crearemos una division para cada uno de los objetos a visualizar.

rolling\_corr\_plot <-   
# ggplotly(   
 rolling\_corr %>%  
 filter(symbol != "DTWEXAFEGS",  
 date>Sys.Date()-30) %>%   
 ggplot(aes(x = date, y = rolling.corr, color = symbol)) +  
 geom\_hline(yintercept = 0, color = palette\_light()[[8]]) +  
 geom\_line(size = 1.5) +  
 labs(title = "Rolling Correlation to DXY CUR",  
 x = "", y = "Correlation", color = "") +  
 facet\_wrap(~ symbol, ncol = 2) +  
 theme\_tq() +  
 scale\_color\_tq()  
#)  
  
rolling\_corr\_plot



# Comportamiento de la correlación para los ultimas 30 dias.

Despues de haber analizado el comportamiento de la correlación y gracias a la importación de los datos sobre niveles de incertidumbre, podemos tener una clara intuición de que los niveles de riesgo deben ser altos en este momento , puesto que la inceridumbre esta explorando nuevos maximos, ahora bien que hay del mito que ente más riesgo mayor el beneficio o rentabilidad?

importemos datos del **Dow Jones** y **Standard & Poors 500**, un indice importante conformado por las empresas mas representativas, y un portafolio bastante equilibrado y diversificado, despues comparemos el comportamiento del riesgo vs el retorno en el paso del tiempo para cada elemento, **Dow Jones & SP500**.

# Ratio de Riesgo vd Rentabilidad:

# DOW JONES  
dow\_jones <-  
 tq\_index("DOW") %>%  
 tq\_get() %>%  
 group\_by(symbol) %>%  
 tq\_transmute(adjusted,  
 mutate\_fun = periodReturn,  
 period = "daily",  
 type = "log",  
 col\_rename = "dlr") %>%  
 summarize(MDLR = mean(dlr),  
 SDDLR = sd(dlr))

## Getting holdings for DOW

dow\_jones

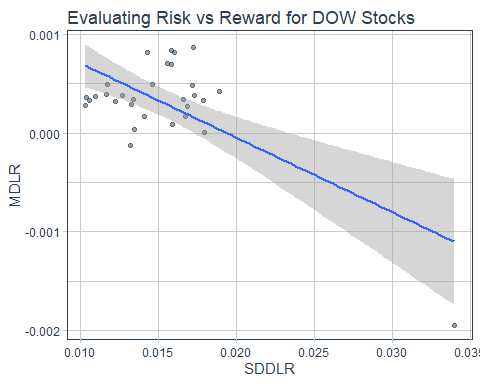
## # A tibble: 30 x 3  
## symbol MDLR SDDLR  
## <chr> <dbl> <dbl>  
## 1 AAPL 0.000867 0.0173  
## 2 AXP 0.000339 0.0166  
## 3 BA 0.000419 0.0189  
## 4 CAT 0.000330 0.0179  
## 5 CSCO 0.000273 0.0168  
## 6 CVX 0.0000825 0.0159  
## 7 DIS 0.000488 0.0146  
## 8 DOW -0.00195 0.0340  
## 9 GS 0.00000987 0.0179  
## 10 HD 0.000819 0.0143  
## # ... with 20 more rows

# SP500  
# sp500 <-  
# tq\_index("SP500") %>%  
# tq\_get() %>%  
# group\_by(symbol) %>%  
# tq\_transmute(adjusted,  
# mutate\_fun = periodReturn,  
# period = "daily",  
# type = "log",  
# col\_rename = "dlr") %>%  
# summarize(MDLR = mean(dlr),  
# SDDLR = sd(dlr))  
#   
# sp500

## Visualizacion de Ratio:

# Evaluating Risk vs Reward for Dow Jones & SP500 Stocks:  
  
dow\_jones %>%  
 ggplot(aes(x = SDDLR, y = MDLR)) +  
 geom\_point(color = palette\_light()[[1]], alpha = 0.5) +  
 geom\_smooth(method = "lm") +  
 labs(title = "Evaluating Risk vs Reward for DOW Stocks") +  
 theme\_tq()

## `geom\_smooth()` using formula 'y ~ x'



# sp500 %>%  
# ggplot(aes(x = SDDLR, y = MDLR)) +  
# geom\_point(color = palette\_light()[[1]], alpha = 0.5) +  
# geom\_smooth(method = "lm") +  
# labs(title = "Evaluating Risk vs Reward for S&P500 Stocks") +  
# theme\_tq()

Podemos deducir facilmente, que los mayores niveles de riesgo no necesariamente se relacionan con mayor rentabilidad esperada, mucho menos en escenarios como el actual donde la incertidumbre es tan alta.

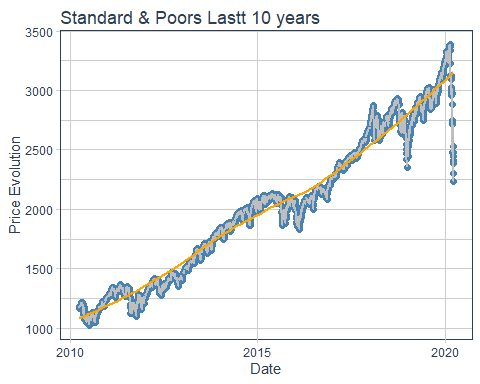
Observemos a continuación, el comportamiento historico del **SP500**, importando los datos y creando un grafico sencillo con regresión lineal.

sp\_500 <- tq\_get("SP500", get = "economic.data", from = "2010-01-01" )  
  
sp\_500 %>% arrange(desc(date)) %>%   
 print(n=25)

## # A tibble: 2,609 x 3  
## symbol date price  
## <chr> <date> <dbl>  
## 1 SP500 2020-03-25 2476.  
## 2 SP500 2020-03-24 2447.  
## 3 SP500 2020-03-23 2237.  
## 4 SP500 2020-03-20 2305.  
## 5 SP500 2020-03-19 2409.  
## 6 SP500 2020-03-18 2398.  
## 7 SP500 2020-03-17 2529.  
## 8 SP500 2020-03-16 2386.  
## 9 SP500 2020-03-13 2711.  
## 10 SP500 2020-03-12 2481.  
## 11 SP500 2020-03-11 2741.  
## 12 SP500 2020-03-10 2882.  
## 13 SP500 2020-03-09 2747.  
## 14 SP500 2020-03-06 2972.  
## 15 SP500 2020-03-05 3024.  
## 16 SP500 2020-03-04 3130.  
## 17 SP500 2020-03-03 3003.  
## 18 SP500 2020-03-02 3090.  
## 19 SP500 2020-02-28 2954.  
## 20 SP500 2020-02-27 2979.  
## 21 SP500 2020-02-26 3116.  
## 22 SP500 2020-02-25 3128.  
## 23 SP500 2020-02-24 3226.  
## 24 SP500 2020-02-21 3338.  
## 25 SP500 2020-02-20 3373.  
## # ... with 2,584 more rows

# tibble del comportamiento historico del SP500   
  
sp\_ggplot <-   
 sp\_500%>%  
 ggplot(aes(x = date, y = price))+  
 theme\_tq()+  
 geom\_point(color = "steelblue", size = 2)+  
 geom\_line(color = "gray", size = 1)+  
 geom\_smooth(color = "orange", method = loess)+  
 labs(x = "Date", y = "Price Evolution",  
 title = paste0("Standard & Poors Lastt 10 years"))  
  
sp\_ggplot

## `geom\_smooth()` using formula 'y ~ x'

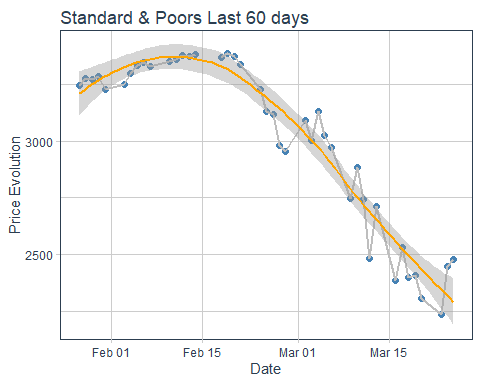


# Evolución del precio historico

En casos como estos en los que poseemos tanta información, resulta prudente y necesario, cortar la serie temporal, es decir observar un periodo de tiempo que este mas acotado, para lograrlo podemos filtrar en el tiempo, para dotarlo de dinamismo, observaremos solo los ultimos 60 dias de manera automatica utilizando la fecha del sistema:

sp500\_ts <-   
 sp\_500 %>%   
 filter(date >= Sys.Date()-60) %>%   
 ggplot(aes(x = date, y = price))+  
 theme\_tq()+  
 geom\_point(color = "steelblue", size = 2)+  
 geom\_line(color = "gray", size = 1)+  
 geom\_smooth(color = "orange", method = loess)+  
 labs(x = "Date", y = "Price Evolution",  
 title = paste0("Standard & Poors Last 60 days"))  
  
sp500\_ts

## `geom\_smooth()` using formula 'y ~ x'



# Comportamiento de los ultimos 60 dias hasta la fecha de ejecución   
  
today()

## [1] "2020-03-25"

# Podemos observar que a pesar de la tendencia alcista que subyacente al SP500, estos ultimos 2 meses se ha estado precipitando agresivamente.

Una vez el escenario macroeconomico mejore, la compra del SP500 no entrañara tanto riesgo, la pregunta esta en el timing, es decir cuando se darán las circunstancias para que el comportamiento mejore, lo aconsejable en el momwnto de ejecución de este reporte es la mesura y causión, sin embargo como proposito de seguir elaborando el ejercicio, haremos un pronóstico del comportamiento de los datos utilizando un recurso creado por el equipo de Data Science de Facebook que se llama Prophet, este paquete se especializa en el analísis de series temporales. Las series temporales son un conjunto de herramientas de gran ayuda en el analísis cuantitativo de este tipo de comportamientos, sin embargo su complejidad es alta y deben ser interpretadas como variables cuantitativas no como pronósticos exactos, porque no existen las predicciones perfectas, menos aun cuando los niveles de incertidumbre están tan elevados.

Más información sobre esta herramienta se encuentra disponible en este link <https://facebook.github.io/prophet/>

## Prophet



A caption

El primer paso es segmentar la serie temporal como lo hicimos antes, se debe tener en cuenta que cualquier modelo o por lo menos la mayoria de estos, tienen un margen de error inferior en momentos no muy alejados del momento inical, puesto que a mayor horizonte temporal masyor incertidumbre respecto al desarrollo del escenario. Se utilizara la serie anterior correspondiente al SP500 pero con datos del ultimpo año, el paquete es muy sencillo, solo utiliza dos variables, **ds** referente al tiempo y **yhat** equivalente al valor a proyectar, por lo tanto, tomamos el objeto anterior y lo transformamos para poder utilizar las funciones de analísis.

# fitting the model by price  
sp <-   
 sp\_500 %>%   
 filter(date >= Sys.Date()-600) %>%   
 select(date, price) %>%   
 rename(ds = date, y = price) %>%   
 na.locf() %>%   
 print(n = 30)

## # A tibble: 429 x 2  
## ds y  
## <date> <dbl>  
## 1 2018-08-03 2840.  
## 2 2018-08-06 2850.  
## 3 2018-08-07 2858.  
## 4 2018-08-08 2858.  
## 5 2018-08-09 2854.  
## 6 2018-08-10 2833.  
## 7 2018-08-13 2822.  
## 8 2018-08-14 2840.  
## 9 2018-08-15 2818.  
## 10 2018-08-16 2841.  
## 11 2018-08-17 2850.  
## 12 2018-08-20 2857.  
## 13 2018-08-21 2863.  
## 14 2018-08-22 2862.  
## 15 2018-08-23 2857.  
## 16 2018-08-24 2875.  
## 17 2018-08-27 2897.  
## 18 2018-08-28 2898.  
## 19 2018-08-29 2914.  
## 20 2018-08-30 2901.  
## 21 2018-08-31 2902.  
## 22 2018-09-03 2902.  
## 23 2018-09-04 2897.  
## 24 2018-09-05 2889.  
## 25 2018-09-06 2878.  
## 26 2018-09-07 2872.  
## 27 2018-09-10 2877.  
## 28 2018-09-11 2888.  
## 29 2018-09-12 2889.  
## 30 2018-09-13 2904.  
## # ... with 399 more rows

sp

## # A tibble: 429 x 2  
## ds y  
## <date> <dbl>  
## 1 2018-08-03 2840.  
## 2 2018-08-06 2850.  
## 3 2018-08-07 2858.  
## 4 2018-08-08 2858.  
## 5 2018-08-09 2854.  
## 6 2018-08-10 2833.  
## 7 2018-08-13 2822.  
## 8 2018-08-14 2840.  
## 9 2018-08-15 2818.  
## 10 2018-08-16 2841.  
## # ... with 419 more rows

m <- prophet(sp)

## Disabling yearly seasonality. Run prophet with yearly.seasonality=TRUE to override this.

## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.

m

## $growth  
## [1] "linear"  
##   
## $changepoints  
## [1] "2018-08-23 GMT" "2018-09-11 GMT" "2018-10-01 GMT" "2018-10-19 GMT"  
## [5] "2018-11-07 GMT" "2018-11-27 GMT" "2018-12-17 GMT" "2019-01-03 GMT"  
## [9] "2019-01-23 GMT" "2019-02-12 GMT" "2019-03-01 GMT" "2019-03-21 GMT"  
## [13] "2019-04-10 GMT" "2019-04-30 GMT" "2019-05-17 GMT" "2019-06-06 GMT"  
## [17] "2019-06-26 GMT" "2019-07-15 GMT" "2019-08-02 GMT" "2019-08-22 GMT"  
## [21] "2019-09-10 GMT" "2019-09-30 GMT" "2019-10-18 GMT" "2019-11-06 GMT"  
## [25] "2019-11-26 GMT"  
##   
## $n.changepoints  
## [1] 25  
##   
## $changepoint.range  
## [1] 0.8  
##   
## $yearly.seasonality  
## [1] "auto"  
##   
## $weekly.seasonality  
## [1] "auto"  
##   
## $daily.seasonality  
## [1] "auto"  
##   
## $holidays  
## NULL  
##   
## $seasonality.mode  
## [1] "additive"  
##   
## $seasonality.prior.scale  
## [1] 10  
##   
## $changepoint.prior.scale  
## [1] 0.05  
##   
## $holidays.prior.scale  
## [1] 10  
##   
## $mcmc.samples  
## [1] 0  
##   
## $interval.width  
## [1] 0.8  
##   
## $uncertainty.samples  
## [1] 1000  
##   
## $specified.changepoints  
## [1] FALSE  
##   
## $start  
## [1] "2018-08-03 GMT"  
##   
## $y.scale  
## [1] 3386.15  
##   
## $logistic.floor  
## [1] FALSE  
##   
## $t.scale  
## [1] 51840000  
##   
## $changepoints.t  
## [1] 0.03333333 0.06500000 0.09833333 0.12833333 0.16000000 0.19333333  
## [7] 0.22666667 0.25500000 0.28833333 0.32166667 0.35000000 0.38333333  
## [13] 0.41666667 0.45000000 0.47833333 0.51166667 0.54500000 0.57666667  
## [19] 0.60666667 0.64000000 0.67166667 0.70500000 0.73500000 0.76666667  
## [25] 0.80000000  
##   
## $seasonalities  
## $seasonalities$weekly  
## $seasonalities$weekly$period  
## [1] 7  
##   
## $seasonalities$weekly$fourier.order  
## [1] 3  
##   
## $seasonalities$weekly$prior.scale  
## [1] 10  
##   
## $seasonalities$weekly$mode  
## [1] "additive"  
##   
## $seasonalities$weekly$condition.name  
## NULL  
##   
##   
##   
## $extra\_regressors  
## list()  
##   
## $country\_holidays  
## NULL  
##   
## $stan.fit  
## $stan.fit$par  
## $stan.fit$par$k  
## [1] -0.3594857  
##   
## $stan.fit$par$m  
## [1] 0.8588717  
##   
## $stan.fit$par$delta  
## [1] 3.044366e-08 -9.437413e-09 -9.379232e-09 -1.059849e-08 -2.099877e-10  
## [6] 7.961587e-04 3.286843e-01 3.444049e-01 5.583713e-07 1.129532e-08  
## [11] -1.358000e-07 -1.606875e-03 -7.985359e-02 -1.404382e-02 -5.167481e-07  
## [16] -7.974438e-08 -5.066147e-08 1.709213e-08 3.844016e-07 4.574996e-02  
## [21] 6.799682e-02 2.112696e-02 5.573774e-06 -6.305949e-09 -6.021997e-01  
##   
## $stan.fit$par$sigma\_obs  
## [1] 0.03965979  
##   
## $stan.fit$par$beta  
## [1] -0.0055870220 0.0057930167 0.0050464626 0.0012208872 -0.0002906588  
## [6] -0.0015517122  
##   
##   
## $stan.fit$value  
## [1] 1140.573  
##   
## $stan.fit$return\_code  
## [1] 0  
##   
## $stan.fit$theta\_tilde  
## k m delta[1] delta[2] delta[3]  
## [1,] -0.3594857 0.8588717 3.044366e-08 -9.437413e-09 -9.379232e-09  
## delta[4] delta[5] delta[6] delta[7] delta[8] delta[9]  
## [1,] -1.059849e-08 -2.099877e-10 0.0007961587 0.3286843 0.3444049 5.583713e-07  
## delta[10] delta[11] delta[12] delta[13] delta[14] delta[15]  
## [1,] 1.129532e-08 -1.358e-07 -0.001606875 -0.07985359 -0.01404382 -5.167481e-07  
## delta[16] delta[17] delta[18] delta[19] delta[20]  
## [1,] -7.974438e-08 -5.066147e-08 1.709213e-08 3.844016e-07 0.04574996  
## delta[21] delta[22] delta[23] delta[24] delta[25] sigma\_obs  
## [1,] 0.06799682 0.02112696 5.573774e-06 -6.305949e-09 -0.6021997 0.03965979  
## beta[1] beta[2] beta[3] beta[4] beta[5]  
## [1,] -0.005587022 0.005793017 0.005046463 0.001220887 -0.0002906588  
## beta[6]  
## [1,] -0.001551712  
##   
##   
## $params  
## $params$k  
## [1] -0.3594857  
##   
## $params$m  
## [1] 0.8588717  
##   
## $params$delta  
## [,1] [,2] [,3] [,4] [,5]  
## [1,] 3.044366e-08 -9.437413e-09 -9.379232e-09 -1.059849e-08 -2.099877e-10  
## [,6] [,7] [,8] [,9] [,10] [,11]  
## [1,] 0.0007961587 0.3286843 0.3444049 5.583713e-07 1.129532e-08 -1.358e-07  
## [,12] [,13] [,14] [,15] [,16]  
## [1,] -0.001606875 -0.07985359 -0.01404382 -5.167481e-07 -7.974438e-08  
## [,17] [,18] [,19] [,20] [,21] [,22]  
## [1,] -5.066147e-08 1.709213e-08 3.844016e-07 0.04574996 0.06799682 0.02112696  
## [,23] [,24] [,25]  
## [1,] 5.573774e-06 -6.305949e-09 -0.6021997  
##   
## $params$sigma\_obs  
## [1] 0.03965979  
##   
## $params$beta  
## [,1] [,2] [,3] [,4] [,5]  
## [1,] -0.005587022 0.005793017 0.005046463 0.001220887 -0.0002906588  
## [,6]  
## [1,] -0.001551712  
##   
##   
## $history  
## # A tibble: 429 x 5  
## ds y floor t y\_scaled  
## <dttm> <dbl> <dbl> <dbl> <dbl>  
## 1 2018-08-03 00:00:00 2840. 0 0 0.839  
## 2 2018-08-06 00:00:00 2850. 0 0.005 0.842  
## 3 2018-08-07 00:00:00 2858. 0 0.00667 0.844  
## 4 2018-08-08 00:00:00 2858. 0 0.00833 0.844  
## 5 2018-08-09 00:00:00 2854. 0 0.01 0.843  
## 6 2018-08-10 00:00:00 2833. 0 0.0117 0.837  
## 7 2018-08-13 00:00:00 2822. 0 0.0167 0.833  
## 8 2018-08-14 00:00:00 2840. 0 0.0183 0.839  
## 9 2018-08-15 00:00:00 2818. 0 0.02 0.832  
## 10 2018-08-16 00:00:00 2841. 0 0.0217 0.839  
## # ... with 419 more rows  
##   
## $history.dates  
## [1] "2018-08-03 GMT" "2018-08-06 GMT" "2018-08-07 GMT" "2018-08-08 GMT"  
## [5] "2018-08-09 GMT" "2018-08-10 GMT" "2018-08-13 GMT" "2018-08-14 GMT"  
## [9] "2018-08-15 GMT" "2018-08-16 GMT" "2018-08-17 GMT" "2018-08-20 GMT"  
## [13] "2018-08-21 GMT" "2018-08-22 GMT" "2018-08-23 GMT" "2018-08-24 GMT"  
## [17] "2018-08-27 GMT" "2018-08-28 GMT" "2018-08-29 GMT" "2018-08-30 GMT"  
## [21] "2018-08-31 GMT" "2018-09-03 GMT" "2018-09-04 GMT" "2018-09-05 GMT"  
## [25] "2018-09-06 GMT" "2018-09-07 GMT" "2018-09-10 GMT" "2018-09-11 GMT"  
## [29] "2018-09-12 GMT" "2018-09-13 GMT" "2018-09-14 GMT" "2018-09-17 GMT"  
## [33] "2018-09-18 GMT" "2018-09-19 GMT" "2018-09-20 GMT" "2018-09-21 GMT"  
## [37] "2018-09-24 GMT" "2018-09-25 GMT" "2018-09-26 GMT" "2018-09-27 GMT"  
## [41] "2018-09-28 GMT" "2018-10-01 GMT" "2018-10-02 GMT" "2018-10-03 GMT"  
## [45] "2018-10-04 GMT" "2018-10-05 GMT" "2018-10-08 GMT" "2018-10-09 GMT"  
## [49] "2018-10-10 GMT" "2018-10-11 GMT" "2018-10-12 GMT" "2018-10-15 GMT"  
## [53] "2018-10-16 GMT" "2018-10-17 GMT" "2018-10-18 GMT" "2018-10-19 GMT"  
## [57] "2018-10-22 GMT" "2018-10-23 GMT" "2018-10-24 GMT" "2018-10-25 GMT"  
## [61] "2018-10-26 GMT" "2018-10-29 GMT" "2018-10-30 GMT" "2018-10-31 GMT"  
## [65] "2018-11-01 GMT" "2018-11-02 GMT" "2018-11-05 GMT" "2018-11-06 GMT"  
## [69] "2018-11-07 GMT" "2018-11-08 GMT" "2018-11-09 GMT" "2018-11-12 GMT"  
## [73] "2018-11-13 GMT" "2018-11-14 GMT" "2018-11-15 GMT" "2018-11-16 GMT"  
## [77] "2018-11-19 GMT" "2018-11-20 GMT" "2018-11-21 GMT" "2018-11-22 GMT"  
## [81] "2018-11-23 GMT" "2018-11-26 GMT" "2018-11-27 GMT" "2018-11-28 GMT"  
## [85] "2018-11-29 GMT" "2018-11-30 GMT" "2018-12-03 GMT" "2018-12-04 GMT"  
## [89] "2018-12-05 GMT" "2018-12-06 GMT" "2018-12-07 GMT" "2018-12-10 GMT"  
## [93] "2018-12-11 GMT" "2018-12-12 GMT" "2018-12-13 GMT" "2018-12-14 GMT"  
## [97] "2018-12-17 GMT" "2018-12-18 GMT" "2018-12-19 GMT" "2018-12-20 GMT"  
## [101] "2018-12-21 GMT" "2018-12-24 GMT" "2018-12-25 GMT" "2018-12-26 GMT"  
## [105] "2018-12-27 GMT" "2018-12-28 GMT" "2018-12-31 GMT" "2019-01-01 GMT"  
## [109] "2019-01-02 GMT" "2019-01-03 GMT" "2019-01-04 GMT" "2019-01-07 GMT"  
## [113] "2019-01-08 GMT" "2019-01-09 GMT" "2019-01-10 GMT" "2019-01-11 GMT"  
## [117] "2019-01-14 GMT" "2019-01-15 GMT" "2019-01-16 GMT" "2019-01-17 GMT"  
## [121] "2019-01-18 GMT" "2019-01-21 GMT" "2019-01-22 GMT" "2019-01-23 GMT"  
## [125] "2019-01-24 GMT" "2019-01-25 GMT" "2019-01-28 GMT" "2019-01-29 GMT"  
## [129] "2019-01-30 GMT" "2019-01-31 GMT" "2019-02-01 GMT" "2019-02-04 GMT"  
## [133] "2019-02-05 GMT" "2019-02-06 GMT" "2019-02-07 GMT" "2019-02-08 GMT"  
## [137] "2019-02-11 GMT" "2019-02-12 GMT" "2019-02-13 GMT" "2019-02-14 GMT"  
## [141] "2019-02-15 GMT" "2019-02-18 GMT" "2019-02-19 GMT" "2019-02-20 GMT"  
## [145] "2019-02-21 GMT" "2019-02-22 GMT" "2019-02-25 GMT" "2019-02-26 GMT"  
## [149] "2019-02-27 GMT" "2019-02-28 GMT" "2019-03-01 GMT" "2019-03-04 GMT"  
## [153] "2019-03-05 GMT" "2019-03-06 GMT" "2019-03-07 GMT" "2019-03-08 GMT"  
## [157] "2019-03-11 GMT" "2019-03-12 GMT" "2019-03-13 GMT" "2019-03-14 GMT"  
## [161] "2019-03-15 GMT" "2019-03-18 GMT" "2019-03-19 GMT" "2019-03-20 GMT"  
## [165] "2019-03-21 GMT" "2019-03-22 GMT" "2019-03-25 GMT" "2019-03-26 GMT"  
## [169] "2019-03-27 GMT" "2019-03-28 GMT" "2019-03-29 GMT" "2019-04-01 GMT"  
## [173] "2019-04-02 GMT" "2019-04-03 GMT" "2019-04-04 GMT" "2019-04-05 GMT"  
## [177] "2019-04-08 GMT" "2019-04-09 GMT" "2019-04-10 GMT" "2019-04-11 GMT"  
## [181] "2019-04-12 GMT" "2019-04-15 GMT" "2019-04-16 GMT" "2019-04-17 GMT"  
## [185] "2019-04-18 GMT" "2019-04-19 GMT" "2019-04-22 GMT" "2019-04-23 GMT"  
## [189] "2019-04-24 GMT" "2019-04-25 GMT" "2019-04-26 GMT" "2019-04-29 GMT"  
## [193] "2019-04-30 GMT" "2019-05-01 GMT" "2019-05-02 GMT" "2019-05-03 GMT"  
## [197] "2019-05-06 GMT" "2019-05-07 GMT" "2019-05-08 GMT" "2019-05-09 GMT"  
## [201] "2019-05-10 GMT" "2019-05-13 GMT" "2019-05-14 GMT" "2019-05-15 GMT"  
## [205] "2019-05-16 GMT" "2019-05-17 GMT" "2019-05-20 GMT" "2019-05-21 GMT"  
## [209] "2019-05-22 GMT" "2019-05-23 GMT" "2019-05-24 GMT" "2019-05-27 GMT"  
## [213] "2019-05-28 GMT" "2019-05-29 GMT" "2019-05-30 GMT" "2019-05-31 GMT"  
## [217] "2019-06-03 GMT" "2019-06-04 GMT" "2019-06-05 GMT" "2019-06-06 GMT"  
## [221] "2019-06-07 GMT" "2019-06-10 GMT" "2019-06-11 GMT" "2019-06-12 GMT"  
## [225] "2019-06-13 GMT" "2019-06-14 GMT" "2019-06-17 GMT" "2019-06-18 GMT"  
## [229] "2019-06-19 GMT" "2019-06-20 GMT" "2019-06-21 GMT" "2019-06-24 GMT"  
## [233] "2019-06-25 GMT" "2019-06-26 GMT" "2019-06-27 GMT" "2019-06-28 GMT"  
## [237] "2019-07-01 GMT" "2019-07-02 GMT" "2019-07-03 GMT" "2019-07-04 GMT"  
## [241] "2019-07-05 GMT" "2019-07-08 GMT" "2019-07-09 GMT" "2019-07-10 GMT"  
## [245] "2019-07-11 GMT" "2019-07-12 GMT" "2019-07-15 GMT" "2019-07-16 GMT"  
## [249] "2019-07-17 GMT" "2019-07-18 GMT" "2019-07-19 GMT" "2019-07-22 GMT"  
## [253] "2019-07-23 GMT" "2019-07-24 GMT" "2019-07-25 GMT" "2019-07-26 GMT"  
## [257] "2019-07-29 GMT" "2019-07-30 GMT" "2019-07-31 GMT" "2019-08-01 GMT"  
## [261] "2019-08-02 GMT" "2019-08-05 GMT" "2019-08-06 GMT" "2019-08-07 GMT"  
## [265] "2019-08-08 GMT" "2019-08-09 GMT" "2019-08-12 GMT" "2019-08-13 GMT"  
## [269] "2019-08-14 GMT" "2019-08-15 GMT" "2019-08-16 GMT" "2019-08-19 GMT"  
## [273] "2019-08-20 GMT" "2019-08-21 GMT" "2019-08-22 GMT" "2019-08-23 GMT"  
## [277] "2019-08-26 GMT" "2019-08-27 GMT" "2019-08-28 GMT" "2019-08-29 GMT"  
## [281] "2019-08-30 GMT" "2019-09-02 GMT" "2019-09-03 GMT" "2019-09-04 GMT"  
## [285] "2019-09-05 GMT" "2019-09-06 GMT" "2019-09-09 GMT" "2019-09-10 GMT"  
## [289] "2019-09-11 GMT" "2019-09-12 GMT" "2019-09-13 GMT" "2019-09-16 GMT"  
## [293] "2019-09-17 GMT" "2019-09-18 GMT" "2019-09-19 GMT" "2019-09-20 GMT"  
## [297] "2019-09-23 GMT" "2019-09-24 GMT" "2019-09-25 GMT" "2019-09-26 GMT"  
## [301] "2019-09-27 GMT" "2019-09-30 GMT" "2019-10-01 GMT" "2019-10-02 GMT"  
## [305] "2019-10-03 GMT" "2019-10-04 GMT" "2019-10-07 GMT" "2019-10-08 GMT"  
## [309] "2019-10-09 GMT" "2019-10-10 GMT" "2019-10-11 GMT" "2019-10-14 GMT"  
## [313] "2019-10-15 GMT" "2019-10-16 GMT" "2019-10-17 GMT" "2019-10-18 GMT"  
## [317] "2019-10-21 GMT" "2019-10-22 GMT" "2019-10-23 GMT" "2019-10-24 GMT"  
## [321] "2019-10-25 GMT" "2019-10-28 GMT" "2019-10-29 GMT" "2019-10-30 GMT"  
## [325] "2019-10-31 GMT" "2019-11-01 GMT" "2019-11-04 GMT" "2019-11-05 GMT"  
## [329] "2019-11-06 GMT" "2019-11-07 GMT" "2019-11-08 GMT" "2019-11-11 GMT"  
## [333] "2019-11-12 GMT" "2019-11-13 GMT" "2019-11-14 GMT" "2019-11-15 GMT"  
## [337] "2019-11-18 GMT" "2019-11-19 GMT" "2019-11-20 GMT" "2019-11-21 GMT"  
## [341] "2019-11-22 GMT" "2019-11-25 GMT" "2019-11-26 GMT" "2019-11-27 GMT"  
## [345] "2019-11-28 GMT" "2019-11-29 GMT" "2019-12-02 GMT" "2019-12-03 GMT"  
## [349] "2019-12-04 GMT" "2019-12-05 GMT" "2019-12-06 GMT" "2019-12-09 GMT"  
## [353] "2019-12-10 GMT" "2019-12-11 GMT" "2019-12-12 GMT" "2019-12-13 GMT"  
## [357] "2019-12-16 GMT" "2019-12-17 GMT" "2019-12-18 GMT" "2019-12-19 GMT"  
## [361] "2019-12-20 GMT" "2019-12-23 GMT" "2019-12-24 GMT" "2019-12-25 GMT"  
## [365] "2019-12-26 GMT" "2019-12-27 GMT" "2019-12-30 GMT" "2019-12-31 GMT"  
## [369] "2020-01-01 GMT" "2020-01-02 GMT" "2020-01-03 GMT" "2020-01-06 GMT"  
## [373] "2020-01-07 GMT" "2020-01-08 GMT" "2020-01-09 GMT" "2020-01-10 GMT"  
## [377] "2020-01-13 GMT" "2020-01-14 GMT" "2020-01-15 GMT" "2020-01-16 GMT"  
## [381] "2020-01-17 GMT" "2020-01-20 GMT" "2020-01-21 GMT" "2020-01-22 GMT"  
## [385] "2020-01-23 GMT" "2020-01-24 GMT" "2020-01-27 GMT" "2020-01-28 GMT"  
## [389] "2020-01-29 GMT" "2020-01-30 GMT" "2020-01-31 GMT" "2020-02-03 GMT"  
## [393] "2020-02-04 GMT" "2020-02-05 GMT" "2020-02-06 GMT" "2020-02-07 GMT"  
## [397] "2020-02-10 GMT" "2020-02-11 GMT" "2020-02-12 GMT" "2020-02-13 GMT"  
## [401] "2020-02-14 GMT" "2020-02-17 GMT" "2020-02-18 GMT" "2020-02-19 GMT"  
## [405] "2020-02-20 GMT" "2020-02-21 GMT" "2020-02-24 GMT" "2020-02-25 GMT"  
## [409] "2020-02-26 GMT" "2020-02-27 GMT" "2020-02-28 GMT" "2020-03-02 GMT"  
## [413] "2020-03-03 GMT" "2020-03-04 GMT" "2020-03-05 GMT" "2020-03-06 GMT"  
## [417] "2020-03-09 GMT" "2020-03-10 GMT" "2020-03-11 GMT" "2020-03-12 GMT"  
## [421] "2020-03-13 GMT" "2020-03-16 GMT" "2020-03-17 GMT" "2020-03-18 GMT"  
## [425] "2020-03-19 GMT" "2020-03-20 GMT" "2020-03-23 GMT" "2020-03-24 GMT"  
## [429] "2020-03-25 GMT"  
##   
## $train.holiday.names  
## NULL  
##   
## $train.component.cols  
## additive\_terms weekly multiplicative\_terms  
## 1 1 1 0  
## 2 1 1 0  
## 3 1 1 0  
## 4 1 1 0  
## 5 1 1 0  
## 6 1 1 0  
##   
## $component.modes  
## $component.modes$additive  
## [1] "weekly" "additive\_terms"   
## [3] "extra\_regressors\_additive" "holidays"   
##   
## $component.modes$multiplicative  
## [1] "multiplicative\_terms" "extra\_regressors\_multiplicative"  
##   
##   
## $fit.kwargs  
## list()  
##   
## attr(,"class")  
## [1] "prophet" "list"

# fitting the model by returns  
sp\_ret <- "SP500" %>%  
 tq\_get(get = "economic.data",  
 from = Sys.Date()-600,  
 to = Sys.Date()) %>%  
 tq\_transmute(select = price,  
 mutate\_fun = periodReturn,  
 period = "daily") %>%  
 rename(ds = date, y = daily.returns) %>%  
 na.locf()

## Warning in to\_period(xx, period = on.opts[[period]], ...): missing values  
## removed from data

sp\_ret

## # A tibble: 413 x 2  
## ds y  
## <date> <dbl>  
## 1 2018-08-03 0   
## 2 2018-08-06 0.00354   
## 3 2018-08-07 0.00282   
## 4 2018-08-08 -0.000262  
## 5 2018-08-09 -0.00144   
## 6 2018-08-10 -0.00711   
## 7 2018-08-13 -0.00401   
## 8 2018-08-14 0.00639   
## 9 2018-08-15 -0.00760   
## 10 2018-08-16 0.00792   
## # ... with 403 more rows

n <- prophet(sp\_ret)

## Disabling yearly seasonality. Run prophet with yearly.seasonality=TRUE to override this.  
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.

Una vez despues de ajustar el modelo, podemos conocer los outputs del mismo,y por ejemplo, crear una tibble o tabla con los valores proyectados en el futuro, en este caso estamos proyectando

# Predict --- Prices  
  
future <- make\_future\_dataframe(m, periods = 15)  
future %>% tail(15)

## ds  
## 430 2020-03-26  
## 431 2020-03-27  
## 432 2020-03-28  
## 433 2020-03-29  
## 434 2020-03-30  
## 435 2020-03-31  
## 436 2020-04-01  
## 437 2020-04-02  
## 438 2020-04-03  
## 439 2020-04-04  
## 440 2020-04-05  
## 441 2020-04-06  
## 442 2020-04-07  
## 443 2020-04-08  
## 444 2020-04-09

forecast <- predict(m, future)  
forecast[c('ds', 'yhat', 'yhat\_lower', 'yhat\_upper')] %>%   
 tail(20)

## ds yhat yhat\_lower yhat\_upper  
## 425 2020-03-19 3007.011 2829.224 3172.889  
## 426 2020-03-20 3004.599 2838.201 3176.197  
## 427 2020-03-23 2991.508 2820.291 3164.654  
## 428 2020-03-24 2995.228 2820.894 3177.049  
## 429 2020-03-25 2994.706 2815.424 3180.520  
## 430 2020-03-26 2997.196 2825.208 3160.466  
## 431 2020-03-27 2994.785 2826.594 3177.143  
## 432 2020-03-28 2939.442 2767.541 3111.999  
## 433 2020-03-29 2938.040 2770.598 3109.385  
## 434 2020-03-30 2981.694 2794.552 3143.390  
## 435 2020-03-31 2985.414 2817.556 3156.749  
## 436 2020-04-01 2984.892 2808.749 3156.721  
## 437 2020-04-02 2987.382 2809.391 3151.170  
## 438 2020-04-03 2984.971 2803.712 3145.406  
## 439 2020-04-04 2929.628 2748.665 3099.615  
## 440 2020-04-05 2928.226 2756.500 3095.586  
## 441 2020-04-06 2971.880 2806.422 3146.998  
## 442 2020-04-07 2975.600 2793.657 3139.677  
## 443 2020-04-08 2975.077 2802.111 3149.532  
## 444 2020-04-09 2977.568 2795.303 3153.586

# Predict --- Returns  
  
future\_n <- make\_future\_dataframe(n, periods = 15)  
future\_n %>% tail(15)

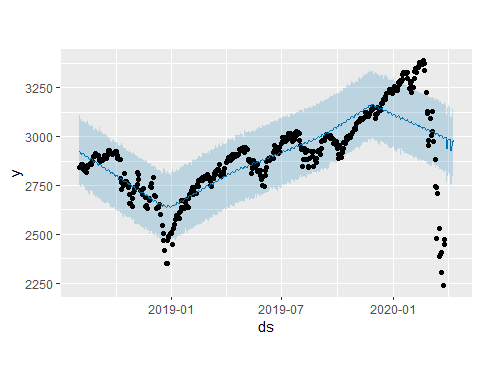
## ds  
## 414 2020-03-26  
## 415 2020-03-27  
## 416 2020-03-28  
## 417 2020-03-29  
## 418 2020-03-30  
## 419 2020-03-31  
## 420 2020-04-01  
## 421 2020-04-02  
## 422 2020-04-03  
## 423 2020-04-04  
## 424 2020-04-05  
## 425 2020-04-06  
## 426 2020-04-07  
## 427 2020-04-08  
## 428 2020-04-09

forecast\_n <- predict(n, future)  
forecast\_n[c('ds', 'yhat', 'yhat\_lower', 'yhat\_upper')] %>%  
 tail(20)

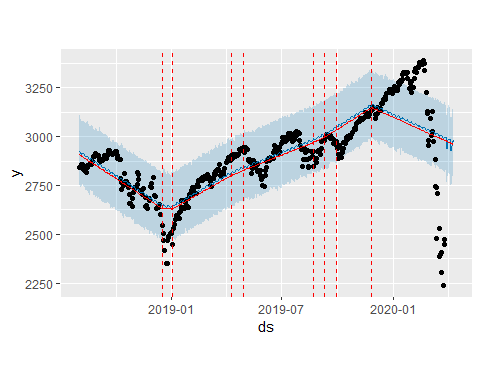
## ds yhat yhat\_lower yhat\_upper  
## 425 2020-03-19 -0.0018512893 -0.02190296 0.01805703  
## 426 2020-03-20 -0.0008613120 -0.02115326 0.01990009  
## 427 2020-03-23 -0.0044381813 -0.02343745 0.01505527  
## 428 2020-03-24 0.0013175475 -0.01953887 0.02153025  
## 429 2020-03-25 -0.0007120644 -0.01959008 0.01932898  
## 430 2020-03-26 -0.0018808381 -0.02096378 0.01858215  
## 431 2020-03-27 -0.0008908609 -0.02148616 0.02112343  
## 432 2020-03-28 -0.0016585321 -0.02104215 0.01748045  
## 433 2020-03-29 -0.0016627534 -0.02220822 0.01804931  
## 434 2020-03-30 -0.0044677302 -0.02565703 0.01348214  
## 435 2020-03-31 0.0012879987 -0.01857124 0.02170128  
## 436 2020-04-01 -0.0007416133 -0.02038646 0.01948576  
## 437 2020-04-02 -0.0019103870 -0.02133334 0.01778430  
## 438 2020-04-03 -0.0009204097 -0.02128806 0.01809702  
## 439 2020-04-04 -0.0016880810 -0.02147298 0.01910958  
## 440 2020-04-05 -0.0016923022 -0.02218071 0.01807040  
## 441 2020-04-06 -0.0044972790 -0.02541577 0.01633097  
## 442 2020-04-07 0.0012584498 -0.01931497 0.02155420  
## 443 2020-04-08 -0.0007711621 -0.02101502 0.01955089  
## 444 2020-04-09 -0.0019399359 -0.02206828 0.01801078

Se puede observar que ya se han obtenido los valores proyectados en el futuro, con un intervalo de posible error que aumenta a lo largo del tiempo, en la tibble, se pueden observar sistintas variables, **ds** que es el tiempo, **yhat** la variable proyectada en el tiempo, **yhat\_lower** de la proyeccion, valor minimo probable y finalmente **yhat\_upper** valor maximo probable. La tabla con los valores proyectados es muy util para cualquier analísis que se desea hacer con mayor nivel de profundidad o detalle, por ejemplo para crear una alerta que notifique una proyeccion muy baja o alta. Puede ser herramientas que combinadas con conocimiento del negocio y entendimiento de la dinamica actual, contribuya a que los procesos de toma de decisiones sea más sofisticado. A continuacion observaremos la visualización de los datos calculados, donde los valores proyectados, seran representados de color azul.

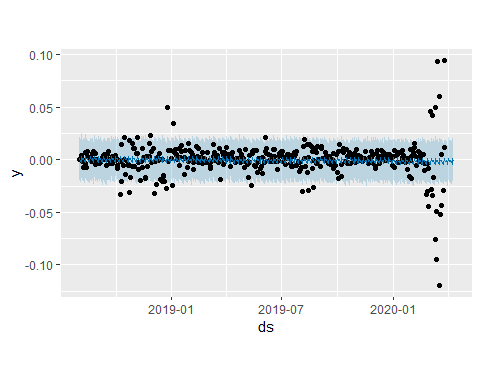
plot(m, forecast)



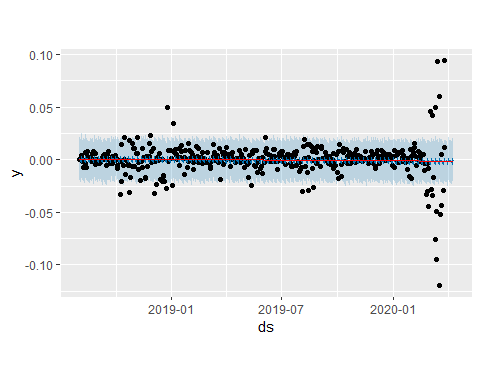
# identificamos con lineas rojas los cambios de dirección en el precio  
plot(m, forecast) + add\_changepoints\_to\_plot(m)



#Proyectar retornos en lugar de precios:  
plot(n, forecast\_n)



# identificamos con lineas rojas los cambios de dirección en el retorno  
plot(n, forecast\_n) + add\_changepoints\_to\_plot(n)



Debe tenerse en cuenta que los niveles de volatitlidad en los ultimos dias, hacen que la serie temporal tenga un comportamient muy erratico, lo que dificulta el proceso de analisis y proyección, para intentar atenuar el efecto, podriamos buscar mas puntos de datos en menos horizonte temporal, para ello la importacion de datos debe tener una mayor frecuencia, es decir, la importación tradicional es de cada dia al cierre, se intentara importar datos intraday, es decir, hora a hora, o a lo mejor cada diez minutos.

Para importar datos con una frecuencia tan alta, fue necesario investigar de donde descargar los datos, de una fuente que sea confiable, segura y de calidad, lo suficientemente rapida para proveer el servicio, este tipo se servicios son pagos, sin embargo se puede acceder a funciones de prueba, a continuación exploramos **AlphaVantage**, <“<https://www.alphavantage.co/>”> un API que permite infomación historica y en tiempo real, de series de tiempo del mercado.

Inicialmente se importaran los datos correspondientes al **SP500**, datos intraday con frecuencia de 1 minuto. para poder acceder a dicha informacion se necesita una llave de acceso al API esta llave es personal, y se puede generar en el link compartido previamente.

A Continuación se comparte una vista de la data descargada en el momento de ejecusión para posteriormente dibujar los datos en el tiempo. Es de esperarse, que el comportamiento se erratico, por el escenario o conyuntura actual, y la alta frecuencia. Conoce el servicio premium en <“<https://www.alphavantage.co/premium/>”>

#key  
av\_api\_key("K40OVQLGD2QIE4TO")  
  
#SP500  
sp\_data <- c("SPX") %>%  
 tq\_get(get = "alphavantager", av\_fun = "TIME\_SERIES\_INTRADAY", interval = "1min")  
  
sp\_data %>% print(n = 35)

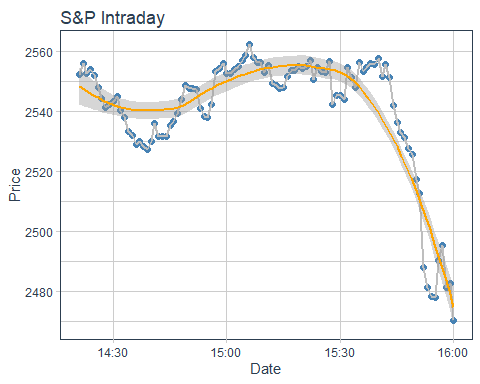
## # A tibble: 100 x 7  
## symbol timestamp open high low close volume  
## <chr> <dttm> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 SPX 2020-03-25 14:21:00 2552. 2556. 2551. 2556. 7846916  
## 2 SPX 2020-03-25 14:22:00 2556. 2556. 2550. 2553. 6621613  
## 3 SPX 2020-03-25 14:23:00 2553. 2554. 2552. 2554. 5831499  
## 4 SPX 2020-03-25 14:24:00 2554. 2555. 2551. 2551. 5691185  
## 5 SPX 2020-03-25 14:25:00 2552. 2552. 2547. 2548. 5965811  
## 6 SPX 2020-03-25 14:26:00 2548. 2549. 2544. 2544. 6519061  
## 7 SPX 2020-03-25 14:27:00 2544. 2548. 2541. 2541. 6692576  
## 8 SPX 2020-03-25 14:28:00 2541. 2542. 2537. 2542. 7257700  
## 9 SPX 2020-03-25 14:29:00 2542. 2546. 2542. 2544. 6208173  
## 10 SPX 2020-03-25 14:30:00 2543. 2548. 2543. 2545. 6206531  
## 11 SPX 2020-03-25 14:31:00 2545 2546. 2540. 2540. 7103475  
## 12 SPX 2020-03-25 14:32:00 2540. 2541. 2536. 2538. 6787856  
## 13 SPX 2020-03-25 14:33:00 2538. 2540. 2533. 2533. 6897160  
## 14 SPX 2020-03-25 14:34:00 2533. 2537. 2532. 2532. 5999002  
## 15 SPX 2020-03-25 14:35:00 2532. 2532. 2527. 2528. 7350341  
## 16 SPX 2020-03-25 14:36:00 2529. 2532. 2529. 2530. 7265818  
## 17 SPX 2020-03-25 14:37:00 2530. 2534. 2528. 2528. 6471201  
## 18 SPX 2020-03-25 14:38:00 2528. 2529. 2524. 2527. 7797507  
## 19 SPX 2020-03-25 14:39:00 2527. 2532. 2525. 2530. 6883294  
## 20 SPX 2020-03-25 14:40:00 2530. 2537. 2528. 2536. 7098246  
## 21 SPX 2020-03-25 14:41:00 2536. 2537. 2533. 2533. 5687062  
## 22 SPX 2020-03-25 14:42:00 2532. 2532. 2530. 2531. 8084849  
## 23 SPX 2020-03-25 14:43:00 2532. 2535. 2531. 2533. 5518469  
## 24 SPX 2020-03-25 14:44:00 2531. 2536. 2531. 2535. 6131071  
## 25 SPX 2020-03-25 14:45:00 2535. 2538. 2534. 2536. 6006547  
## 26 SPX 2020-03-25 14:46:00 2537. 2539. 2537. 2538. 6391751  
## 27 SPX 2020-03-25 14:47:00 2539. 2545. 2539. 2544. 6902766  
## 28 SPX 2020-03-25 14:48:00 2544. 2549. 2543. 2549. 7647755  
## 29 SPX 2020-03-25 14:49:00 2548. 2549. 2546. 2547. 6199553  
## 30 SPX 2020-03-25 14:50:00 2548. 2550. 2546. 2548. 7111577  
## 31 SPX 2020-03-25 14:51:00 2547. 2550. 2547. 2547. 6341323  
## 32 SPX 2020-03-25 14:52:00 2547. 2548. 2541. 2541. 7071157  
## 33 SPX 2020-03-25 14:53:00 2541. 2542. 2536. 2538. 7625376  
## 34 SPX 2020-03-25 14:54:00 2538. 2540. 2537. 2538. 5569043  
## 35 SPX 2020-03-25 14:55:00 2538. 2542. 2537. 2542. 5988835  
## # ... with 65 more rows

### Visualización con datos Intraday.

graficación de los datos intradía con regresión lineal de ggplot.

sp\_data %>%   
 ggplot(aes(x = timestamp, y = open))+  
 theme\_tq()+  
 geom\_point(color = "steelblue", size = 2)+  
 geom\_line(color = "gray", size = 1)+  
 geom\_smooth(color = "orange", method = loess)+  
 labs(x = "Date", y = "Price",  
 title = "S&P Intraday")

## `geom\_smooth()` using formula 'y ~ x'



Esta serie temporal del S&P500 es intradia y por la representación grafica y la corta ventana temporal y alta frecuencia, a simple vista no parece ser estacional ni tiene una tendencia clara.

A continuacion aplicaremos el modelo anteriormente aplicado de la libreria Prophet, para ello se deben transformar los datos de forma tal que puedan ser interpretables por la librería.

Para que la serie tenga un comportamiento mas suavizado, se creara una columna llamada **adj** resultado del promedio del **open, close, high & low** en cada instante de tiempo.

sp\_data\_ts <- sp\_data %>%   
 group\_by(timestamp) %>%   
 mutate(adj = mean(c(open,high,close,low))) %>%   
 select("timestamp", "adj") %>%   
 rename(ds = timestamp, y = adj) %>%  
 na.locf()   
  
sp\_data\_ts

## # A tibble: 100 x 2  
## # Groups: ds [100]  
## ds y  
## <dttm> <dbl>  
## 1 2020-03-25 14:21:00 2554.  
## 2 2020-03-25 14:22:00 2554.  
## 3 2020-03-25 14:23:00 2553.  
## 4 2020-03-25 14:24:00 2553.  
## 5 2020-03-25 14:25:00 2550.  
## 6 2020-03-25 14:26:00 2546.  
## 7 2020-03-25 14:27:00 2544.  
## 8 2020-03-25 14:28:00 2540.  
## 9 2020-03-25 14:29:00 2544.  
## 10 2020-03-25 14:30:00 2545.  
## # ... with 90 more rows

m <- prophet(sp\_data\_ts, daily.seasonality = "auto",  
 fit = TRUE, seasonality.mode = "multiplicative",  
 uncertainty.samples = 100)

## Disabling yearly seasonality. Run prophet with yearly.seasonality=TRUE to override this.

## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.

## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.

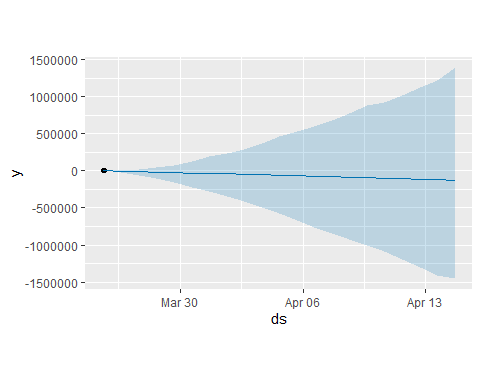
# Predict --- Prices  
  
future <- make\_future\_dataframe(m, periods = 20)  
future %>% tail(15)

## ds  
## 106 2020-03-31 16:00:00  
## 107 2020-04-01 16:00:00  
## 108 2020-04-02 16:00:00  
## 109 2020-04-03 16:00:00  
## 110 2020-04-04 16:00:00  
## 111 2020-04-05 16:00:00  
## 112 2020-04-06 16:00:00  
## 113 2020-04-07 16:00:00  
## 114 2020-04-08 16:00:00  
## 115 2020-04-09 16:00:00  
## 116 2020-04-10 16:00:00  
## 117 2020-04-11 16:00:00  
## 118 2020-04-12 16:00:00  
## 119 2020-04-13 16:00:00  
## 120 2020-04-14 16:00:00

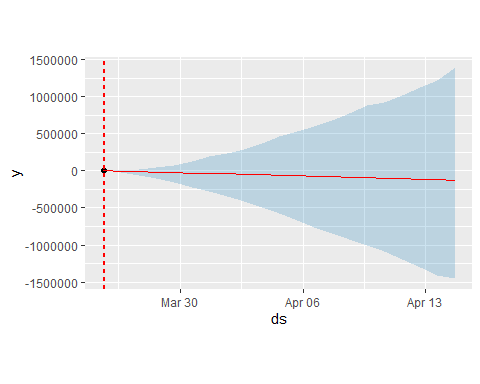
forecast <- predict(m, future)  
forecast[c('ds', 'yhat', 'yhat\_lower', 'yhat\_upper')] %>%   
 tail(20)

## ds yhat yhat\_lower yhat\_upper  
## 101 2020-03-26 16:00:00 -3994.62 -20607.73 10456.62  
## 102 2020-03-27 16:00:00 -10455.63 -56990.23 21448.65  
## 103 2020-03-28 16:00:00 -16916.63 -102317.77 40303.75  
## 104 2020-03-29 16:00:00 -23377.64 -161551.03 74101.40  
## 105 2020-03-30 16:00:00 -29838.64 -224925.35 128842.18  
## 106 2020-03-31 16:00:00 -36299.65 -284791.77 188427.89  
## 107 2020-04-01 16:00:00 -42760.66 -344622.04 233158.47  
## 108 2020-04-02 16:00:00 -49221.66 -411315.56 290873.83  
## 109 2020-04-03 16:00:00 -55682.67 -488772.48 364707.17  
## 110 2020-04-04 16:00:00 -62143.68 -577997.71 461617.37  
## 111 2020-04-05 16:00:00 -68604.68 -670809.05 532613.82  
## 112 2020-04-06 16:00:00 -75065.69 -762149.19 601473.75  
## 113 2020-04-07 16:00:00 -81526.69 -838890.50 675578.74  
## 114 2020-04-08 16:00:00 -87987.70 -926808.32 774124.54  
## 115 2020-04-09 16:00:00 -94448.71 -1006105.49 877857.43  
## 116 2020-04-10 16:00:00 -100909.71 -1093709.13 926122.33  
## 117 2020-04-11 16:00:00 -107370.72 -1195145.54 1008695.56  
## 118 2020-04-12 16:00:00 -113831.73 -1298615.71 1122944.42  
## 119 2020-04-13 16:00:00 -120292.73 -1406088.03 1218438.33  
## 120 2020-04-14 16:00:00 -126753.74 -1457612.57 1387606.04

plot(m, forecast)



# identificamos con lineas rojas los cambios de dirección en el precio  
plot(m, forecast) + add\_changepoints\_to\_plot(m)



Al leer el codigo detalladamente, se puede observar que el objeto que contiene la información, denominado **sp\_data\_ts**, es aquel que contiene los datos intradía, descargados con el API de AlphaVantager, ahora bien los datos, tienen una frecuencia de un minuto, es decir que cada fila es una observación de la variable minuto a minuto, y la columna, no incorpora más que un dia de información, esta frecuencia tan alta de importación de datos, genera un conflicto con las herramientas del packete **Prophet** y también con las herramientas de visualización, para slocuionarlo, se sugiere una transformacion de los datos en los que tomando la matriz original **sp\_data\_ts**, creamos una columna temporal que no es más que el conteo de las filas de la serie, esta es creada con el fin de modificar la variable temporal **ds** y añadir un dia a cada observación para que el modelo pueda funcionar y desplegar su output de manera exitosa.

sp\_data\_ts <- sp\_data\_ts %>%   
 ungroup() %>%   
 mutate(ds = as.Date(ds),  
 day = row\_number(),  
 ds = ds+day) %>%   
 select(-day)  
  
sp\_data\_ts

## # A tibble: 100 x 2  
## ds y  
## <date> <dbl>  
## 1 2020-03-26 2554.  
## 2 2020-03-27 2554.  
## 3 2020-03-28 2553.  
## 4 2020-03-29 2553.  
## 5 2020-03-30 2550.  
## 6 2020-03-31 2546.  
## 7 2020-04-01 2544.  
## 8 2020-04-02 2540.  
## 9 2020-04-03 2544.  
## 10 2020-04-04 2545.  
## # ... with 90 more rows

m <- prophet(sp\_data\_ts, daily.seasonality = "auto",  
 fit = TRUE, seasonality.mode = "multiplicative",  
 uncertainty.samples = 100)

## Disabling yearly seasonality. Run prophet with yearly.seasonality=TRUE to override this.

## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.

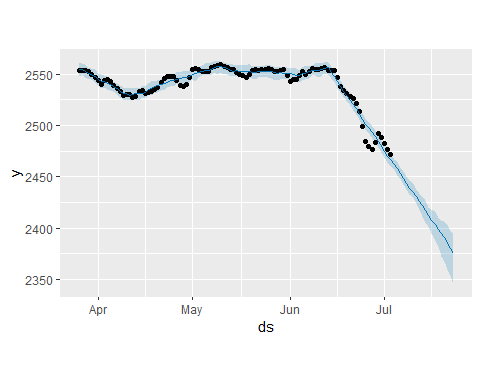
# Predict --- Prices  
  
future <- make\_future\_dataframe(m, periods = 20)  
future %>% tail(15)

## ds  
## 106 2020-07-09  
## 107 2020-07-10  
## 108 2020-07-11  
## 109 2020-07-12  
## 110 2020-07-13  
## 111 2020-07-14  
## 112 2020-07-15  
## 113 2020-07-16  
## 114 2020-07-17  
## 115 2020-07-18  
## 116 2020-07-19  
## 117 2020-07-20  
## 118 2020-07-21  
## 119 2020-07-22  
## 120 2020-07-23

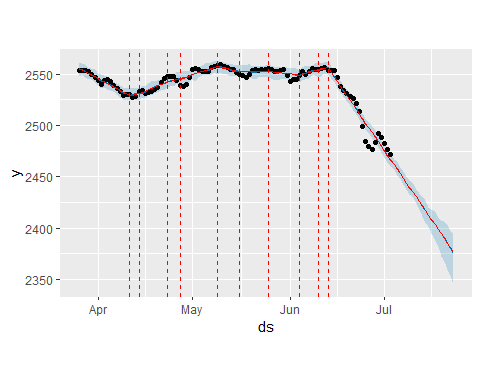
forecast <- predict(m, future)  
forecast[c('ds', 'yhat', 'yhat\_lower', 'yhat\_upper')] %>%   
 tail(20)

## ds yhat yhat\_lower yhat\_upper  
## 101 2020-07-04 2461.886 2457.394 2466.755  
## 102 2020-07-05 2457.887 2451.806 2464.677  
## 103 2020-07-06 2453.378 2448.436 2458.664  
## 104 2020-07-07 2448.689 2442.459 2454.995  
## 105 2020-07-08 2443.317 2437.212 2450.549  
## 106 2020-07-09 2438.556 2432.921 2446.239  
## 107 2020-07-10 2435.154 2430.110 2443.426  
## 108 2020-07-11 2430.434 2424.026 2437.704  
## 109 2020-07-12 2426.428 2418.059 2433.848  
## 110 2020-07-13 2421.920 2411.114 2431.282  
## 111 2020-07-14 2417.233 2405.603 2427.118  
## 112 2020-07-15 2411.873 2401.226 2420.127  
## 113 2020-07-16 2407.115 2395.278 2417.874  
## 114 2020-07-17 2403.698 2388.448 2416.332  
## 115 2020-07-18 2398.981 2382.511 2410.731  
## 116 2020-07-19 2394.969 2372.151 2406.046  
## 117 2020-07-20 2390.461 2367.849 2406.624  
## 118 2020-07-21 2385.776 2359.883 2403.107  
## 119 2020-07-22 2380.428 2353.919 2397.237  
## 120 2020-07-23 2375.673 2343.799 2394.650

plot(m, forecast)



# identificamos con lineas rojas los cambios de dirección en el precio  
plot(m, forecast) + add\_changepoints\_to\_plot(m)



Finalmente al realizar las observaciones e implementaciones de modelo, este sugiere que según las ultimas observaciones, la tendencia al alza persistira, un ejercicio interesante sera observar el desarrollo de los mercados a ver cuando se recuperan de la crisis actual, en el momento que se perciba acercamiento de la recuperación y la economia empiece a responder a los estimulos economicos, es muy probable que se recupere el equity.

El modelo y aplicativo o instrumento de la tesis, permite al usuario hacer este tipo de analísis con más variables, y un mayor nivel de interactividad entre estas.

Este reporte dinámico, no constituye el instrumento de la investigación si no su socialización, el codigo para ejecutar y reproducir este documento con valores actuales al momento de ejecución, esta disponible en el siguiente repositorio, <“<https://github.com/Aggarch/MQF-UAH>”>, lugar en el que también se encuentra disponible el código que compartiremos a continuación:

Estte repositorio permite hacer el control de versiones del instrumento para que pueda ser sometido a la mejora continua.

El codigo que se desplegara a continuación sigue la misma logica general, de los ejercicios desarrollados en este R Markdown.

El lector los podrá ejecutar sin dificultad en su consola R y buscar sus propios **insights** de negocio, tener en cuenta que cualquiera de los pasos y lineas de codigo a continuación, cumple uno o más de los siguientes objetivos;

**Importar, Transformar, Modelar y/o Comunicar**

Y que se trata de una dinamica Iterativa.

Codigo perfectamente reproducible si se cuentan con conocimientos muy basicos de la herramienta.

### Documentación

tidyverse: <“<https://cran.r-project.org/web/packages/tidyverse/index.html>> lubridate: <”<https://cran.r-project.org/web/packages/lubridate/vignettes/lubridate.html>“> knitr: <”<https://cran.r-project.org/web/packages/knitr/index.html>“> tidyquant <”<https://cran.r-project.org/web/packages/tidyquant/index.html>“> xml2 <”<https://cran.r-project.org/web/packages/xml2/index.html>“> corrplot <”<https://cran.r-project.org/web/packages/corrplot/corrplot.pdf>“> alphavantager <”<https://cran.r-project.org/web/packages/alphavantager/alphavantager.pdf>“> plotly <”<https://cran.r-project.org/web/packages/plotly/index.html>“> prophet <”<https://cran.r-project.org/web/packages/prophet/index.html>">

#### QuantApp MBA Andrés García & Sagith Amín ####  
  
# ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::: ----  
  
# Infrastructure : tidyquant {timetk, xts, zoo, Qunatmod, TTR, PerformanceAnalytics}  
# Data Sources ::: FRED, Bloomberg, Quandl, Morningstar, Yahoo Finance  
# Data Since ::: 2010-01-01 {After Sub-prime cicle, to reduce RV}  
  
  
# Vignettes :::  
  
# browseVignetteas(package = "tidyquant")  
# browseVignettes(package = "timetk")  
  
  
# Thesis: Data treatment philosophy:  
# 1.) Tidy - Data: ####  
# makes reference to the macroeconomic data and treasury rates in a tidy format:   
# Data tidyng phiosophy: Each Column represents a variable, each row show the values.  
browseURL("https://vita.had.co.nz/papers/tidy-data.pdf")  
  
# Displpay some tidyquant options :::  
  
tq\_mutate\_fun\_options()  
tq\_performance\_fun\_options()  
# ... ----  
  
# 2.) MACROECONOMICS & PRIMES ####  
# Macroeconomics:   
  
# Code may be optimize, however I prefer to indicate it step by step  
  
# One by Variable:  
  
# GDP per capita is gross domestic product divided by population.  
# GDP is the sum of gross value added by all resident  
# producers in the economy plus any product taxes and minus any  
# subsidies not included in the value of the products. It is calculated  
# without making deductions for depreciation of fabricated assets or for  
# depletion and degradation of natural resources.  
# World Bank national accounts data, and OECD National Accounts data files.  
  
# # It's easier to make only one call to the data using the tq\_get(), however for clarification   
# # purposes, we prefer to create one object by each variable. :   
  
# Like :   
# Market <- tq\_get(c("DTWEXM", "DEXJPUS", "DEXCAUS", "DEXUSAL",  
# "DEXUSUK", "DEXUSEU", "DGS10", "FEDFUNDS", "DCOILWTICO",   
# "GOLDAMGBD228NLBM","DJIA","SP500"),  
# get = "economic.data", from = "2006-01-01") %>%   
# filter(year(date) >= "2010-01-01") %>%   
# pivot\_wider(names\_from = symbol, values\_from = price) %>%   
# na.locf() %>%   
# rename(DXY\_CUR = DTWEXM)   
  
  
  
# GDP stands for "Gross Domestic Product" and represents the total monetary value of all final goods and services  
# produced (and sold on the market) tthis index make reference to the Gobal market.  
Global\_GDP <- tq\_get("NYGDPPCAPKDWLD", get = "economic.data", from = "2017-01-01")  
  
# Economic Policy Uncertainty Index: EPU  
#browseURL("http://www.policyuncertainty.com/media/BakerBloomDavis.pdf")  
Economic\_Policy\_Risk <- tq\_get("USEPUINDXD", get = "economic.data", from = "2017-01-01")  
  
# Global Economic Policy Uncertainty:   
#browseURL("https://www.nber.org/papers/w22740")  
Global\_Economic\_Uncertainty <- tq\_get("GEPUPPP", get = "economic.data", from = "2010-01-01")  
  
# Trade Policy uncertainty Index  
#browseURL("https://www.policyuncertainty.com/china\_monthly.html")  
Trade\_Policy\_Risk <- tq\_get("CHNMAINLANDTPU", get = "economic.data", from = "2010-01-01")  
  
# Monthly Supplpy of Houses:   
Houses\_Month\_supply <- tq\_get("MSACSR", get = "economic.data", from = "2010-01-01")   
  
# Consumer price Index - Inflation measure  
CPI <- tq\_get("CPIAUCSL", get = "economic.data", from = "2010-01-01", to = Sys.Date())  
  
# Real Gross Domestic Prduct for US  
Real\_GDP <- tq\_get("A191RL1Q225SBEA", get = "economic.data", from = "2010-01-01")  
  
# Industrial Production Index  
IPI <- tq\_get("INDPRO", get = "economic.data", from = "2010-01-01")   
  
# Non-Farm Payrrols / Employment Measure  
Non\_Farm\_Payrolls <- tq\_get("PAYEMS", get = "economic.data", from = "2010-01-01")  
  
# Unemployment Rate  
Unemployment\_Rate <- tq\_get("UNRATE", get = "economic.data", from = "2010-01-01")  
   
# US Treasury 10 Years  
Treasury\_10y <- tq\_get("DGS10", get = "economic.data", from = "2010-01-01")  
  
# US Treasury 2 Years   
Treasury\_2y <- tq\_get("GS2", get = "economic.data", from = "2010-01-01")  
  
# Reference interest Rates / setted by Central Bank>   
FED\_Interest\_Rate <- tq\_get("FEDFUNDS", get = "economic.data", from = "2010-01-01")  
  
  
# A ggplot of the Global Economic Policy Uncertainty :   
Economic\_policy\_index <-   
 ggplotly(  
 Economic\_Policy\_Risk %>%  
 filter(date >= "2019-10-20") %>%   
 ggplot(aes(x = date, y = price))+  
 theme\_tq()+  
 geom\_point(color = "steelblue", size = 2)+  
 geom\_line(color = "gray", size = 1)+  
 geom\_smooth(color = "orange", method = loess)+  
 labs(x = "Date", y = "Economic Policy Uncertainty Index",  
 title = "Global Economic Policy Uncertainty Evolution")  
 )  
  
# 2.1) Tidy tibble of Macro & Primes ####  
Macro\_primes <-   
 bind\_rows("Economic\_Policy\_Risk" = Economic\_Policy\_Risk,  
 "Global\_Uncertainty" = Global\_Economic\_Uncertainty,  
 "Trade\_Policy\_Risk" = Trade\_Policy\_Risk,  
 "CPI" = CPI,  
 "Houses\_Month\_supply" = Houses\_Month\_supply,  
 "IPI" = IPI,   
 "Non\_Farm\_Payrolls" = Non\_Farm\_Payrolls,  
 "Unemployment\_Rate" = Unemployment\_Rate,  
 "Treasury\_10y" = Treasury\_10y,   
 "Treasury\_2y" = Treasury\_2y, .id = "commodity\_name")  
Macro\_primes  
  
# 2.2) Correlation Macro & Primes ####  
# This correlation can be calculated after the data spread and matrix building   
Macros\_Primes\_corr <-   
 Macro\_primes %>%   
 pivot\_wider(names\_from = commodity\_name,values\_from = price) %>%   
 na.locf() %>%   
 select(-date) %>%   
 cor()   
Macros\_Primes\_corr  
  
# Correlation matrix ploted   
Macros\_Primes\_Corrplot <-   
 corrplot(Macros\_Primes\_corr, method = "shade" ,  
 addCoef.col = "gray",   
 tl.col = "darkblue",  
 order = "hclust",)  
Macros\_Primes\_Corrplot  
  
# Corrplot looks good at 80$ zoom  
  
# ... ----  
  
# 3.) INDEXES ####  
  
# 3.1) Composition of Indexes ####  
SP500 <- tq\_index("SP500")  
DJI <- tq\_index("DOW")  
  
  
# 3.2) Sectors Pareto ####  
SP500\_sectors <-   
 SP500 %>%   
 group\_by(sector) %>%  
 summarise(weight2 = sum(weight)) %>%  
 arrange(desc(weight2)) %>%   
 mutate(Pareto = cumsum(weight2))  
  
DJI\_sectors <-   
 DJI %>%  
 mutate(pareto = cumsum(weight)) %>%  
 group\_by(sector) %>%  
 summarise(weight2 = sum(weight)) %>%  
 arrange(desc(weight2)) %>%  
 mutate(cumulat = cumsum(weight2))  
  
# NASDAQ Inside ::.  
# NDAQ <- tq\_exchange("NASDAQ")  
#   
# NDAQ\_sectors <-   
# NSDAQ%>%  
# group\_by(sector) %>%  
# summarise(n = n() , Mcap2 = cumsum(market.cap)/sum(market.cap)) %>%  
# arrange(desc(Mcap2))  
  
# 3.3) Historic of Indexes ####  
SP500 <- tq\_get("SP500", get = "economic.data", from = "2010-01-01" )  
DOW <- tq\_get("DJIA", get = "economic.data", from = "2010-01-01" )  
NASDAQ <- tq\_get("NASDAQCOM", get = "economic.data", from = "2010-01-01")  
NIKKEI <- tq\_get("NIKKEI225", get = "economic.data", from = "2010-01-01" )   
VIX <- tq\_get("VIXCLS", get = "economic.data", from = "2010-01-01" )  
DXY\_CUR <- tq\_get("DTWEXM", get = "economic.data",from = "2010-01-01")   
  
  
# 3.4) Tidy tibble of Indexes ####  
Indexes <- bind\_rows("DOW" = DOW, "SP500" = SP500,  
 "NDAQ" = NASDAQ, "NIKKEI" = NIKKEI,  
 "VIX" = VIX, "DXY" = DXY\_CUR, .id = "Indexes")  
  
Indexes <- bind\_rows("DOW" = DOW, "SP500" = SP500,  
 "NDAQ" = NASDAQ, "NIKKEI" = NIKKEI,  
 "VIX" = VIX, "Economic\_Policy\_Risk" = Economic\_Policy\_Risk,  
 .id = "Indexes")  
  
Indexes <- bind\_rows("DOW" = DOW, "SP500" = SP500,  
 "NDAQ"= NASDAQ, "NIKKEI" = NIKKEI,  
 "VIX" = VIX, "EPU" = Economic\_Policy\_Risk,  
 "DXY" = DXY\_CUR,  
 .id = "Indexes")  
  
Indexes  
  
# 3.5) Correlation of Indexes ####  
# We can observe that the correlation between the positive returns of stocks against volatility its negative   
# It drives us to dive into the relationship between Risk & Return   
  
Indexes %>%   
 pivot\_wider(names\_from = Indexes, values\_from = price) %>%   
 select(-date,-symbol) %>%   
 na.omit() %>%   
 cor() -> index\_cor  
  
corrplot(index\_cor)  
  
Index\_Corrplot <-   
 corrplot(index\_cor, method = "square",  
 addCoef.col = "gray",   
 tl.col = "darkblue",  
 order = "hclust")  
  
# Whats the Mean correlation of this heatmap,  
# It may be interesting to plot it as the intercept of a rolling corr.   
DXY\_mean\_cor <- mean(index\_cor [,7:7])  
  
# 3.6) Rolling Correlation ####  
  
# 1) Calculate returns: (daily)  
Index\_daily\_returns <- Indexes %>%   
 group\_by(Indexes) %>%   
 tq\_transmute(select = price,  
 mutate\_fun = periodReturn,  
 period = "daily")%>%   
 filter(date >= "2019-09-15")  
  
# 2) Baseline  
baseline\_returns\_daily <- "DTWEXM" %>%  
 tq\_get(get = "economic.data",  
 from = "2019-09-15",  
 to = Sys.Date()) %>%  
 tq\_transmute(select = price,  
 mutate\_fun = periodReturn,  
 period = "daily")  
  
returns\_joined <- left\_join(Index\_daily\_returns,  
 baseline\_returns\_daily,  
 by = "date") %>% na.omit  
  
# 3) Calculate the correlation.   
Index\_rolling\_corr <- returns\_joined %>%  
 tq\_transmute\_xy(x = daily.returns.x,  
 y = daily.returns.y,  
 mutate\_fun = runCor,  
 n = 6,  
 col\_rename = "rolling.corr")  
  
ggplotly(   
 Index\_rolling\_corr %>%  
 filter(Indexes != "DXY") %>%   
 ggplot(aes(x = date, y = rolling.corr, color = Indexes)) +  
 geom\_hline(yintercept = c(DXY\_mean\_cor, 0), color = palette\_light()[[8]]) +  
 geom\_line(size = 1.5) +  
 labs(title = "Rolling Correlation to DXY",  
 x = "", y = "Correlation", color = "") +  
 facet\_wrap(~ Indexes, ncol = 2) +  
 theme\_tq() +  
 scale\_color\_tq()  
)  
  
# ... ----  
  
# 4.) STOCKS ####  
AAPL <- tq\_get("AAPL", get = "stock.prices", from = "2010-01-01")  
GOOG <- tq\_get("GOOG", get = "stock.prices", from = "2010-01-01")  
MSFT <- tq\_get("MSFT", get = "stock.prices", from = "2010-01-01")  
AMZN <- tq\_get("AMZN", get = "stock.prices", from = "2010-01-01")  
FB <- tq\_get("FB", get = "stock.prices", from = "2010-01-01")  
TWTR <- tq\_get("TWTR", get = "stock.prices", from = "2010-01-01")  
  
# 4.1) Tidy tibble of Stocks ####  
Stocks <- bind\_rows("AAPL" = AAPL, "GOOG" = GOOG, "MSFT" = MSFT,  
 "AMZN" = AMZN, "FB" = FB, "TWTR" = TWTR, .id = "id")   
  
# 4.2) Monthly price ####  
Stocks %>%   
 group\_by(id) %>%   
 tq\_transmute(select = close,   
 mutate\_fun = to.monthly,  
 indexAt = "lastof")  
  
# 4.3) Rollapply: ####  
# Calculate returns   
Returns <-   
 Stocks %>%   
 group\_by(id) %>%   
 tq\_transmute(adjusted,   
 periodReturn, period = "weekly",   
 col\_rename = "Returns")  
  
apple\_returns <-   
 AAPL %>%   
 tq\_transmute(adjusted,  
 periodReturn, period = "weekly",  
 col\_rename = "APPLE.returns")  
  
fb\_returns <-   
 FB %>%   
 tq\_transmute(adjusted,  
 periodReturn, period = "weekly",  
 col\_rename = "FB.returns")  
  
# To blend returns or combine them calculate indicidually a join them.  
blend\_returns <- left\_join(fb\_returns, apple\_returns, by = "date")  
  
# Apply custom function  
regr\_fun <- function(data) {  
 coef(lm(FB.returns ~ APPLE.returns, data = timetk::tk\_tbl(data, silent = TRUE)))  
}  
  
# ... ----   
  
# 5.) Rolling regression with rollapply ####  
# Apply the custom function.   
  
blend\_returns %>%  
 tq\_mutate(mutate\_fun = rollapply,  
 width = 12,  
 FUN = regr\_fun,  
 by.column = FALSE,  
 col\_rename = c("coef.0", "coef.1"))  
  
# ... ----   
  
# 6.) COMMODITIES ####  
  
# Set category id for Oil and Gold.  
WTI <- tq\_get("WTISPLC", get = "economic.data", from = "2010-01-01" )  
Brent <- tq\_get("POILBREUSDM", get = "economic.data", from = "2010-01-01")  
Gold <- tq\_get("GOLDAMGBD228NLBM", get = "economic.data", from = "2010-01-01")  
Aluminum <- tq\_get("PALUMUSDM", get = "economic.data", from = "2010-01-01")  
#Copper <- tq\_get("PCOPPUSDM", get = "economic.data", from = "2010-01-01")  
Corn <- tq\_get("PMAIZMTUSDM", get = "economic.data", from = "2010-01-01")   
Soy <- tq\_get("PSOYBUSDQ", get = "economic.data", from = "2010-01-01")  
  
# 6.1) Tidy tibble of Commodities ####  
Commodities <-   
 bind\_rows("WTI" = WTI, "Brent" = Brent,  
 "Gold" = Gold, "Aluminum" = Aluminum,   
 "Corn" = Corn, "Soy" = Soy,  
 .id = "commodity\_name")  
Commodities  
  
  
# 6.2) Commodities Correlation ####  
Commodities\_cor <-   
 Commodities %>%   
 pivot\_wider(names\_from = commodity\_name,values\_from = price) %>%   
 na.locf() %>%   
 select(-date) %>%   
 cor()   
Commodities\_cor  
  
#Quaterly  
Commodities\_Corrplot <-   
 corrplot(Commodities\_cor, method = "square",  
 addCoef.col = "gray",   
 tl.col = "darkblue",  
 order = "hclust")  
Commodities\_Corrplot  
  
# ... ----  
  
# 7.) CURRENCIES ####  
# Pricipal currencies in the world:  
# browseURL("https://www.investopedia.com/trading/most-tradable-currencies/")  
# Dollar Basket:  
# browseURL("https://www.bloomberg.com/quote/DXY:CUR")  
  
#Dollar Basket:  
# DXY\_CUR <- tq\_get("DTWEXM", get = "economic.data",from = "2010-01-01")   
  
  
DXY\_CUR <- tq\_get("DTWEXAFEGS", get = "economic.data",from = "2010-01-01")   
EUR\_USD <- tq\_get("DEXUSEU", get = "economic.data", from = "2010-01-01")  
GBP\_USD <- tq\_get("DEXUSUK", get = "economic.data", from = "2010-01-01")  
USD\_JPY <- tq\_get("DEXJPUS", get = "economic.data", from = "2010-01-01")  
USD\_CAD <- tq\_get("DEXCAUS", get = "economic.data", from = "2010-01-01")  
AUD\_USD <- tq\_get("DEXUSAL", get = "economic.data", from = "2010-01-01")  
USD\_CHF <- tq\_get("DEXSZUS", get = "economic.data", from = "2010-01-01")  
USD\_SEK <- tq\_get("DEXSDUS", get = "economic.data", from = "2010-01-01")  
  
# Primes are upside in Macroeconomics   
  
FX <- bind\_rows("DXY:CUR" = DXY\_CUR,  
 "EUR/USD" = EUR\_USD, "GBP/USD" = GBP\_USD,  
 "USD/JPY" = USD\_JPY, "USD/CAD" = USD\_CAD,  
 "AUD/USD" = AUD\_USD, "USD/CHF" = USD\_CHF,  
 "USD/SEK" = USD\_SEK, .id = "currency")  
  
FX  
  
# 7.1) Tidy tibble of Fx + primes ####  
FX\_primes <- bind\_rows("DXY:CUR" = DXY\_CUR,   
 "EUR/USD" = EUR\_USD, "GBP/USD" = GBP\_USD,  
 "USD/JPY" = USD\_JPY, "USD/CAD" = USD\_CAD,  
 "AUD/USD" = AUD\_USD, "USD/CHF" = USD\_CHF,  
 "ZAR/USD"=ZAR\_USD, "FED\_Interest\_Rate" = FED\_Interest\_Rate,   
 "Treasury\_10y" = Treasury\_10y, "Treasury\_2y" = Treasury\_10y,  
 .id = "currency")   
FX\_primes  
  
# 7.2) Correlate the F Exchange ####  
FX\_primes\_cor <-   
 FX\_primes %>%   
 pivot\_wider(names\_from = currency, values\_from = price) %>%   
 select(-date) %>%   
 na.omit() %>%   
 cor()   
  
  
FX\_Primes\_Corrplot <-   
 corrplot(FX\_primes\_cor, method = "square",  
 addCoef.col = "gray",   
 tl.col = "darkblue",  
 order = "hclust")  
  
# ... ----  
  
# 8) DATA LAKE TABLE xlsx ####  
# Data lost cause bind\_rows step,   
# Full Data coverage, do data frames really affect the result ?   
  
# FX <-   
# DXY\_CUR %>%   
# left\_join(EUR\_USD, by = c("date" = "date")) %>%   
# left\_join(GBP\_USD, by = c("date" = "date")) %>%   
# left\_join(USD\_JPY, by = c("date" = "date")) %>%   
# left\_join(USD\_CAD, by = c("date" = "date")) %>%   
# left\_join(AUD\_USD, by = c("date" = "date")) %>%   
# left\_join(USD\_CHF, by = c("date" = "date")) %>%   
# left\_join(ZAR\_USD, by = c("date" = "date")) %>%   
#   
# rename(DXY\_CUR = price.x, EUR\_USD = price.y, GBP\_USD = price.x.x, USD\_JPY = price.y.y,  
# USD\_CAD = price.x.x.x, AUD\_USD = price.y.y.y, USD\_CHF = price.x.x.x.x, ZAR\_USD = price.y.y.y.y ) %>%   
# na.locf()  
  
  
  
# Macroeconomics   
MACROS <-   
 Macro\_primes %>%   
 pivot\_wider(names\_from = commodity\_name,values\_from = price) %>%   
 na.locf()  
MACROS  
  
# Indexes   
INDEX <-   
 Indexes %>%   
 pivot\_wider(names\_from = Indexes, values\_from = price) %>%   
 na.locf()  
INDEX  
  
# Commodities   
COMMODITIES <-   
 Commodities %>%   
 pivot\_wider(names\_from = commodity\_name,values\_from = price) %>%   
 na.locf()  
COMMODITIES  
  
# FX exchanges   
FX <-   
 FX %>%   
 pivot\_wider(names\_from = currency, values\_from = price) %>%   
 na.locf()  
FX  
  
# FX exchanges and Primes in one tibble   
FX\_PRIMES <-   
 FX\_primes %>%   
 pivot\_wider(names\_from = currency, values\_from = price) %>%   
 na.locf()  
FX\_PRIMES  
  
# Global jointtt  
GLOBAL\_MARKET\_RISK <-   
 MACROS %>%   
 left\_join(INDEX, by = c("date" = "date")) %>%   
 left\_join(COMMODITIES, by = c("date" = "date")) %>%   
 left\_join(FX, by = c("date" = "date")) %>%   
 na.locf()  
GLOBAL\_MARKET\_RISK  
  
#install.packages("openxlsx")  
  
# Lets save our datasets in a easy to read Excel tables, verify the Working Directory.   
  
getwd()  
setwd("C:/Users/ANALITICA/Documents/Thesis DataFrames")  
  
library(openxlsx)  
  
wb <- createWorkbook()  
addWorksheet(wb = wb, sheetName = "FX", gridLines = TRUE)  
writeDataTable(wb = wb, sheet = 1, x = FX)  
  
openxlsx::write.xlsx(MACROS, "1.) MACROECONOMICS.xlsx", asTable = TRUE, sheetName = "MACROECONOMICS")  
openxlsx::write.xlsx(INDEX, "2.) INDEX.xlsx", asTable = TRUE, sheetName = "Index")  
openxlsx::write.xlsx(COMMODITIES, "3.) COMMODITIES.xlsx", asTable = TRUE, sheetName = "Commodities")  
openxlsx::write.xlsx(FX, "4.) FX.xlsx", asTable = TRUE, sheetName = " Currencies")  
openxlsx::write.xlsx(FX\_PRIMES, "5.) FX\_PRIMES.xlsx", asTable = TRUE, sheetName = "FX and Primes")  
openxlsx::write.xlsx(GLOBAL\_MARKET\_RISK, "6.) GLOBAL\_MARKET\_RISK.xlsx", asTable = TRUE, sheetName = "Global\_Market\_Risk")   
# Nuestro pool de datos es de 3578 registros sobre 31 variables.   
  
  
# To succesfully combine the 3 assets cathegories into one tidy data tibble,   
# the id label must be the same,   
# then we can spread the data and correlate.   
  
# GLOBAL\_MARKET\_RISK   
# Already Tidy   
GLOBAL\_MARKET\_RISK   
  
Global\_cor <-   
 GLOBAL\_MARKET\_RISK %>%   
 select(-date) %>%   
 na.locf() %>%   
 cor()   
  
  
# Correlation of GLOBAL\_MARKET   
Global\_market\_Corrplot <-   
 corrplot(Global\_cor, method = "square",  
 tl.col = "darkblue",  
 order = "hclust")  
Global\_market\_Corrplot  
  
# ... ----  
  
# 9) PORTFOLIO ####  
  
#### time Series ####  
  
library(prophet)  
library(tidyverse)  
library(lubridate)  
  
setwd("C:/Users/ANALITICA/Desktop")  
  
DXY\_CUR <- tq\_get("DTWEXM", get = "economic.data",from = "2017-01-01") %>%   
 select(date, price) %>%   
 rename(ds = date, y = price)  
  
m <- prophet(DXY\_CUR)  
  
future <- make\_future\_dataframe(m, periods = 95)  
tail(future)  
  
forecast <- predict(m, future)  
tail(forecast[c('ds', 'yhat', 'yhat\_lower', 'yhat\_upper')])  
  
  
# R  
ggplotly(   
 plot(m, forecast)  
)  
  
# plotting with dygraphs  
library(dygraphs)   
dyplot.prophet(m, forecast)  
  
  
# Series Components  
prophet\_plot\_components(m, forecast)  
  
# Add checkpoints   
plot(m, forecast) + add\_changepoints\_to\_plot(m)  
  
# R  
m <- prophet(DXY\_CUR, changepoint.prior.scale = 0.5)  
forecast <- predict(m, future)  
plot(m, forecast)  
  
  
#### Sources ::::::::::::::::::::::::::::::::::::::::::::::::::::: ####  
  
#### Simple Portfolio ####   
  
stock\_returns\_monthly <- c("AAPL", "GOOG", "NFLX") %>%  
 tq\_get(get = "stock.prices",  
 from = "2010-01-01",  
 to = "2015-12-31") %>%  
 group\_by(symbol) %>%  
 tq\_transmute(select = adjusted,   
 mutate\_fun = periodReturn,   
 period = "monthly",   
 col\_rename = "Ra")  
stock\_returns\_monthly  
  
baseline\_returns\_monthly <- "XLK" %>%  
 tq\_get(get = "stock.prices",  
 from = "2010-01-01",  
 to = "2015-12-31") %>%  
 tq\_transmute(select = adjusted,   
 mutate\_fun = periodReturn,   
 period = "monthly",   
 col\_rename = "Rb")  
baseline\_returns\_monthly  
  
wts <- c(0.5, 0.0, 0.5)  
portfolio\_returns\_monthly <- stock\_returns\_monthly %>%  
 tq\_portfolio(assets\_col = symbol,   
 returns\_col = Ra,   
 weights = wts,   
 col\_rename = "Ra")  
portfolio\_returns\_monthly  
  
wts\_map <- tibble(  
 symbols = c("AAPL", "NFLX"),  
 weights = c(0.5, 0.5)  
)  
wts\_map  
  
stock\_returns\_monthly %>%  
 tq\_portfolio(assets\_col = symbol,   
 returns\_col = Ra,   
 weights = wts\_map,   
 col\_rename = "Ra\_using\_wts\_map")  
  
RaRb\_single\_portfolio <- left\_join(portfolio\_returns\_monthly,   
 baseline\_returns\_monthly,  
 by = "date")  
RaRb\_single\_portfolio  
  
portfolio\_returns\_monthly %>%  
 ggplot(aes(x = date, y = Ra)) +  
 geom\_bar(stat = "identity", fill = palette\_light()[[1]]) +  
 labs(title = "Portfolio Returns",  
 subtitle = "50% AAPL, 0% GOOG, and 50% NFLX",  
 caption = "Shows an above-zero trend meaning positive returns",  
 x = "", y = "Monthly Returns") +  
 geom\_smooth(method = "lm") +  
 theme\_tq() +  
 scale\_color\_tq() +  
 scale\_y\_continuous(labels = scales::percent)  
  
wts <- c(0.5, 0, 0.5)  
portfolio\_growth\_monthly <- stock\_returns\_monthly %>%  
 tq\_portfolio(assets\_col = symbol,   
 returns\_col = Ra,   
 weights = wts,   
 col\_rename = "investment.growth",  
 wealth.index = TRUE) %>%  
 mutate(investment.growth = investment.growth \* 10000)  
  
portfolio\_growth\_monthly %>%  
 ggplot(aes(x = date, y = investment.growth)) +  
 geom\_line(size = 2, color = palette\_light()[[1]]) +  
 labs(title = "Portfolio Growth",  
 subtitle = "50% AAPL, 0% GOOG, and 50% NFLX",  
 caption = "Now we can really visualize performance!",  
 x = "", y = "Portfolio Value") +  
 geom\_smooth(method = "loess") +  
 theme\_tq() +  
 scale\_color\_tq() +  
 scale\_y\_continuous(labels = scales::dollar)  
  
  
monthly\_returns\_stocks <- FANG %>%  
 group\_by(symbol) %>%  
 tq\_transmute(adjusted, periodReturn, period = "monthly")  
  
weights <- c(0.50, 0.25, 0.25, 0)  
tq\_portfolio(data = monthly\_returns\_stocks,  
 assets\_col = symbol,  
 returns\_col = monthly.returns,  
 weights = weights,  
 col\_rename = NULL,  
 wealth.index = FALSE)  
  
  
  
#### Multi Portfolio ####  
  
stock\_returns\_monthly\_multi <- stock\_returns\_monthly %>%  
 tq\_repeat\_df(n = 3)  
stock\_returns\_monthly\_multi  
  
  
weights <- c(  
 0.50, 0.25, 0.25,  
 0.25, 0.50, 0.25,  
 0.25, 0.25, 0.50  
)  
stocks <- c("AAPL", "GOOG", "NFLX")  
weights\_table <- tibble(stocks) %>%  
 tq\_repeat\_df(n = 3) %>%  
 bind\_cols(tibble(weights)) %>%  
 group\_by(portfolio)  
weights\_table  
  
  
portfolio\_returns\_monthly\_multi <- stock\_returns\_monthly\_multi %>%  
 tq\_portfolio(assets\_col = symbol,   
 returns\_col = Ra,   
 weights = weights\_table,   
 col\_rename = "Ra")  
portfolio\_returns\_monthly\_multi  
  
RaRb\_multiple\_portfolio <- left\_join(portfolio\_returns\_monthly\_multi,   
 baseline\_returns\_monthly,  
 by = "date")  
RaRb\_multiple\_portfolio  
  
  
portfolio\_growth\_monthly\_multi <- stock\_returns\_monthly\_multi %>%  
 tq\_portfolio(assets\_col = symbol,   
 returns\_col = Ra,   
 weights = weights\_table,   
 col\_rename = "investment.growth",  
 wealth.index = TRUE) %>%  
 mutate(investment.growth = investment.growth \* 10000)  
  
  
portfolio\_growth\_monthly\_multi %>%  
 ggplot(aes(x = date, y = investment.growth, color = factor(portfolio))) +  
 geom\_line(size = 2) +  
 labs(title = "Portfolio Growth",  
 subtitle = "Comparing Multiple Portfolios",  
 caption = "Portfolio 3 is a Standout!",  
 x = "", y = "Portfolio Value",  
 color = "Portfolio") +  
 geom\_smooth(method = "loess") +  
 theme\_tq() +  
 scale\_color\_tq() +  
 scale\_y\_continuous(labels = scales::dollar)  
  
  
# Analize the SP500 Index Portfoloio Returns.   
  
SP500\_returns <- SP500 %>%  
 tq\_transmute(select = price,  
 mutate\_fun = periodReturn,  
 period = "monthly",  
 col\_rename = "Ra")  
  
  
# All Weather : ####  
  
#1.0 IMPORT DATA ----  
symbols <- c("VTI", "TLT", "IEF", "GLD", "DBC")  
end <- "2019-06-30" %>% ymd()  
start <- end - years(30) + days(1)  
  
raw\_data <- symbols %>%   
 tq\_get(get = "stock.prices",  
 from = start,  
 to = end)  
  
raw\_data %>%   
 group\_by(symbol) %>%   
 summarise(min\_date = min(date),   
 max\_date = max(date))  
  
# 2.0 TRANSFORM TO RETURNS ----  
# normal returns  
returns\_reg\_tbl <- raw\_data %>%   
 select(symbol, date, adjusted) %>%   
 group\_by(symbol) %>%   
 tq\_transmute(select = adjusted,  
 mutate\_fun = periodReturn,  
 period = "monthly") %>%   
 ungroup() %>%   
   
 #rollback to first day of the month - ETF Issue ----  
mutate(date = lubridate::rollback(date, roll\_to\_first = TRUE))  
  
raw\_data %>%   
 select(symbol, date, adjusted) %>%   
 group\_by(symbol) %>%   
 tq\_transmute(select = adjusted,  
 mutate\_fun = periodReturn,  
 period = "monthly") %>%   
 ungroup() %>%   
 spread(symbol, monthly.returns) %>%   
 na.omit()  
  
returns\_reg\_date\_tbl <- returns\_reg\_tbl %>%   
 group\_by(symbol) %>%   
 filter(date >= "2006-02-01")  
  
wts\_tbl <- c(0.3, 0.4, 0.15, 0.075, 0.075)  
  
returns\_port\_tbl <- returns\_reg\_date\_tbl %>%   
 tq\_portfolio(assets\_col = symbol,  
 returns\_col = monthly.returns,  
 weights = wts\_tbl,  
 rebalance\_on = "years") %>%   
 add\_column(symbol = "Portfolio", .before = 1) %>%   
 rename(monthly.returns = portfolio.returns)  
end\_port\_date <- last(returns\_port\_tbl$date)  
  
returns\_port\_tbl  
  
# Weights Matrix ####  
w\_2 <- c(0.3, 0.4, 0.15, 0.075, 0.075,  
 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)  
  
weights\_tbl <- tibble(symbols) %>%   
 tq\_repeat\_df(n = 6) %>%   
 bind\_cols(tibble(w\_2)) %>%   
 group\_by(portfolio)  
  
allocation <- weights\_tbl %>%   
 ungroup() %>%   
 mutate(w\_2 = paste0(w\_2\*100, "%")) %>%   
 pivot\_wider(names\_from = symbols, values\_from = w\_2) %>%   
 mutate(portfolio = case\_when(portfolio == 1 ~ "All Seasons Portfolio",  
 portfolio == 2 ~ "VTI",  
 portfolio == 3 ~ "TLT",  
 portfolio == 4 ~ "IEF",  
 portfolio == 5 ~ "GLD",  
 portfolio == 6 ~ "DBC"))   
  
  
##### Multi Returns & Potfolio   
  
returns\_multi\_reg\_date\_tbl <- returns\_reg\_date\_tbl %>%   
 ungroup() %>%   
 tq\_repeat\_df(n = 6)  
  
port\_returns\_investment\_tbl <- returns\_multi\_reg\_date\_tbl %>%   
 tq\_portfolio(assets\_col = symbol,  
 returns\_col = monthly.returns,  
 weights = weights\_tbl,  
 wealth.index = TRUE) %>%   
 mutate(investment.growth = portfolio.wealthindex \* 10000)  
  
end\_port\_returns\_investment\_tbl <- last(port\_returns\_investment\_tbl$date)  
  
# ready to graph.   
  
  
# Tech small portfolio: ####   
  
Ra <- c("AAPL", "GOOG", "NFLX") %>%  
 tq\_get(get = "stock.prices",  
 from = "2010-01-01",  
 to = "2015-12-31") %>%  
 group\_by(symbol) %>%  
 tq\_transmute(select = adjusted,   
 mutate\_fun = periodReturn,   
 period = "monthly",   
 col\_rename = "Ra")  
Ra  
  
# Lets Compare the returns above, wth the XLK SPDR   
  
Rb <- "XLK" %>%  
 tq\_get(get = "stock.prices",  
 from = "2010-01-01",  
 to = "2015-12-31") %>%  
 tq\_transmute(select = adjusted,   
 mutate\_fun = periodReturn,   
 period = "monthly",   
 col\_rename = "Rb")  
Rb  
  
# Combine the wth a left join.   
  
RaRb <- left\_join(Ra, Rb, by = c("date" = "date"))  
RaRb  
  
# Finally, we can retrieve the performance metrics  
# Use tq\_performance\_fun\_options() to see the list of compatible performance functions.  
  
RaRb\_capm <- RaRb %>%   
 tq\_performance(Ra = Ra,  
 Rb = Rb,  
 performance\_fun = table.CAPM)  
RaRb\_capm  
  
# We can isolate attributes, such as alpha = measure of growth & beta, risk measure.   
  
RaRb\_capm %>%   
 select(Alpha, Beta)  
  
# 1A ####  
# Get Stock prices .::   
  
stock\_prices <-   
 c("AAPL", "GOOG", "NFLX") %>%   
 tq\_get(get = "stock.prices",  
 from = "2010-01-01",  
 to = today())  
stock\_prices  
  
# 2A ####  
# Mutate to Returrns :.:  
  
# monthly ...:::  
stock\_returns\_monthly <-   
 stock\_prices %>%   
 tq\_transmute(select = adjusted,   
 mutate\_fun = periodoReturn,  
 period = "monthly",  
 col\_rename = "Ra")  
  
# daily ...:::   
stock\_retunrs\_daily <-   
 stock\_prices %>%   
 tq\_transmute( select = adjusted,  
 mutate\_fun = periodReturn,   
 period = "daily",  
 col\_rename = "Ra")  
  
# 3A ####   
# Analyze Performance ::.  
  
stock\_retunrs\_monthly %>%   
 tq\_performance(Ra = Ra,   
 Rb= NULL,  
 performance\_fun = SharpeRatio)  
  
  
  
# EDA  
  
netflix <- tq\_get("NFLX", get = "stock.prices")  
  
# sp\_500 <- tq\_index("SP500") %>%  
# tq\_get(get = "stock.prices")  
  
sp\_500 <- tq\_index("SP500") %>%  
 tq\_get(get = "stock.prices") %>%  
 group\_by(symbol) %>%  
 filter(year(date) >= 2017)  
  
  
# Monthly Returns::  
sp\_500 %>%  
 group\_by(symbol) %>%  
 tq\_transmute(adjusted,mutate\_fun = monthlyReturn)  
  
sp\_500 <-  
 tq\_index("SP500") %>%  
 tq\_get() %>%  
 group\_by(symbol) %>%  
 tq\_transmute(adjusted,  
 mutate\_fun = periodReturn,  
 period = "daily",  
 type = "log",  
 col\_rename = "dlr") %>%  
 summarize(MDLR = mean(dlr),  
 SDDLR = sd(dlr))  
  
# Evaluating Risk vs Reward for S&P500 Stocks:  
sp\_500 %>%  
 ggplot(aes(x = SDDLR, y = MDLR)) +  
 geom\_point(color = palette\_light()[[1]], alpha = 0.5) +  
 geom\_smooth(method = "lm") +  
 labs(title = "Evaluating Risk vs Reward for S&P500 Stocks") +  
 theme\_tq()  
  
  
# Yearly Returns :::  
FANG\_annual\_returns <- FANG %>%  
 group\_by(symbol) %>%  
 tq\_transmute(select = adjusted,  
 mutate\_fun = periodReturn,  
 period = "yearly",  
 type = "arithmetic")  
  
FANG\_anual\_returns  
  
  
# FANG Annual Returns:  
  
FANG\_annual\_returns %>%  
 ggplot(aes(x = date, y = yearly.returns, fill = symbol)) +  
 geom\_col() +  
 geom\_hline(yintercept = 0, color = palette\_light()[[1]]) +  
 scale\_y\_continuous(labels = scales::percent) +  
 labs(title = "FANG: Annual Returns",  
 subtitle = "Get annual returns quickly with tq\_transmute!",  
 y = "Annual Returns", x = "") +  
 facet\_wrap(~ symbol, ncol = 2, scales = "free\_y") +  
 theme\_tq() +  
 scale\_fill\_tq()  
  
# FANG Daily Log returns:  
  
FANG\_daily\_log\_returns <- FANG %>%  
 group\_by(symbol) %>%  
 tq\_transmute(select = adjusted,  
 mutate\_fun = periodReturn,  
 period = "daily",  
 type = "log",  
 col\_rename = "monthly.returns")  
  
  
# Plotting log Returns:  
  
FANG\_daily\_log\_returns %>%  
 ggplot(aes(x = monthly.returns, fill = symbol)) +  
 geom\_density(alpha = 0.5) +  
 labs(title = "FANG: Charting the Daily Log Returns",  
 x = "Monthly Returns", y = "Density") +  
 theme\_tq() +  
 scale\_fill\_tq() +  
 facet\_wrap(~ symbol, ncol = 2)  
  
  
FANG %>%  
 group\_by(symbol) %>%  
 tq\_transmute(select = open:volume,  
 mutate\_fun = to.period,  
 period = "months")  
  
# Daily stock prices:  
  
FANG\_daily <- FANG %>%  
 group\_by(symbol)  
  
FANG\_daily %>%  
 ggplot(aes(x = date, y = adjusted, color = symbol)) +  
 geom\_line(size = 1) +  
 labs(title = "Daily Stock Prices",  
 x = "", y = "Adjusted Prices", color = "") +  
 facet\_wrap(~ symbol, ncol = 2, scales = "free\_y") +  
 scale\_y\_continuous(labels = scales::dollar) +  
 theme\_tq() +  
 scale\_color\_tq()  
  
# Monthly stock prices:  
  
FANG\_monthly <- FANG %>%  
 group\_by(symbol) %>%  
 tq\_transmute(select = adjusted,  
 mutate\_fun = to.period,  
 period = "months")  
  
FANG\_monthly %>%  
 ggplot(aes(x = date, y = adjusted, color = symbol)) +  
 geom\_line(size = 1) +  
 labs(title = "Monthly Stock Prices",  
 x = "", y = "Adjusted Prices", color = "") +  
 facet\_wrap(~ symbol, ncol = 2, scales = "free\_y") +  
 scale\_y\_continuous(labels = scales::dollar) +  
 theme\_tq() +  
 scale\_color\_tq()  
  
# Asset Returns  
  
---------------------------------------  
   
 FANG\_returns\_monthly <- FANG %>%  
 group\_by(symbol) %>%  
 tq\_transmute(select = adjusted,  
 mutate\_fun = periodReturn,  
 period = "monthly")  
  
# Baseline Returns  
baseline\_returns\_monthly <- "XLK" %>%  
 tq\_get(get = "stock.prices",  
 from = "2016-01-01",  
 to = Sys.Date()) %>%  
 tq\_transmute(select = adjusted,  
 mutate\_fun = periodReturn,  
 period = "monthly")  
  
returns\_joined <- left\_join(FANG\_returns\_monthly,  
 baseline\_returns\_monthly,  
 by = "date")  
returns\_joined  
  
  
  
# Rolling Correlations: !!!! ####   
  
FANG\_rolling\_corr <- returns\_joined %>%  
 tq\_transmute\_xy(x = monthly.returns.x,  
 y = monthly.returns.y,  
 mutate\_fun = runCor,  
 n = 6,  
 col\_rename = "rolling.corr.6")  
  
FANG\_rolling\_corr %>%  
 ggplot(aes(x = date, y = rolling.corr.6, color = symbol)) +  
 geom\_hline(yintercept = 0, color = palette\_light()[[1]]) +  
 geom\_line(size = 1.5) +  
 labs(title = "FANG: Six Month Rolling Correlation to XLK",  
 x = "", y = "Correlation", color = "") +  
 facet\_wrap(~ symbol, ncol = 2) +  
 theme\_tq() +  
 scale\_color\_tq()  
  
------------------------------------------------------  
   
 # Moving Average Convergence Divergence (MACD)  
   
 FANG\_macd <- FANG %>%  
 group\_by(symbol) %>%  
 tq\_mutate(select = close,  
 mutate\_fun = MACD,  
 nFast = 12,  
 nSlow = 26,  
 nSig = 9,  
 maType = SMA) %>%  
 mutate(diff = macd - signal) %>%  
 select(-(open:volume))  
FANG\_macd  
  
  
FANG\_macd %>%  
 filter(date >= as\_date("2016-10-01")) %>%  
 ggplot(aes(x = date)) +  
 geom\_hline(yintercept = 0, color = palette\_light()[[1]]) +  
 geom\_line(aes(y = macd, col = symbol)) +  
 geom\_line(aes(y = signal), color = "blue", linetype = 2) +  
 geom\_bar(aes(y = diff), stat = "identity", color = palette\_light()[[1]]) +  
 facet\_wrap(~ symbol, ncol = 2, scale = "free\_y") +  
 labs(title = "FANG: Moving Average Convergence Divergence",  
 y = "MACD", x = "", color = "") +  
 theme\_tq() +  
 scale\_color\_tq()  
  
# by quater  
  
FANG\_max\_by\_qtr <- FANG %>%  
 group\_by(symbol) %>%  
 tq\_transmute(select = adjusted,  
 mutate\_fun = apply.quarterly,  
 FUN = max,  
 col\_rename = "max.close") %>%  
 mutate(year.qtr = paste0(year(date), "-Q", quarter(date))) %>%  
 select(-date)  
FANG\_max\_by\_qtr  
  
# Visualize  
  
FANG\_by\_qtr %>%  
 ggplot(aes(x = year.qtr, color = symbol)) +  
 geom\_segment(aes(xend = year.qtr, y = min.close, yend = max.close),  
 size = 1) +  
 geom\_point(aes(y = max.close), size = 2) +  
 geom\_point(aes(y = min.close), size = 2) +  
 facet\_wrap(~ symbol, ncol = 2, scale = "free\_y") +  
 labs(title = "FANG: Min/Max Price By Quarter",  
 y = "Stock Price", color = "") +  
 theme\_tq() +  
 scale\_color\_tq() +  
 scale\_y\_continuous(labels = scales::dollar) +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1),  
 axis.title.x = element\_blank())  
  
# Get stock pairs  
stock\_prices <- c("MA", "V") %>%  
 tq\_get(get = "stock.prices",  
 from = "2015-01-01",  
 to = "2016-12-31") %>%  
 group\_by(symbol)  
  
stock\_pairs <- stock\_prices %>%  
 tq\_transmute(select = adjusted,  
 mutate\_fun = periodReturn,  
 period = "daily",  
 type = "log",  
 col\_rename = "returns") %>%  
 spread(key = symbol, value = returns)  
  
stock\_pairs %>%  
 ggplot(aes(x = V, y = MA)) +  
 geom\_point(color = palette\_light()[[1]], alpha = 0.5) +  
 geom\_smooth(method = "lm") +  
 labs(title = "Visualizing Returns Relationship of Stock Pairs") +  
 theme\_tq()  
  
  
# Stock Monthly Returns  
  
stock\_returns\_monthly <- c("AAPL", "GOOG", "NFLX") %>%  
 tq\_get(get = "stock.prices",  
 from = "2010-01-01",  
 to = "2015-12-31") %>%  
 group\_by(symbol) %>%  
 tq\_transmute(select = adjusted,  
 mutate\_fun = periodReturn,  
 period = "monthly",  
 col\_rename = "Ra")  
  
  
baseline\_returns\_monthly <- "XLK" %>%  
 tq\_get(get = "stock.prices",  
 from = "2010-01-01",  
 to = "2015-12-31") %>%  
 tq\_transmute(select = adjusted,  
 mutate\_fun = periodReturn,  
 period = "monthly",  
 col\_rename = "Rb")  
baseline\_returns\_monthly  
  
# weights Matrix  
  
wts <- c(.5, 0, .5)  
portfolio\_returns\_monthly <- stock\_returns\_monthly %>%  
 tq\_portfolio(assets\_col = symbol,  
 returns\_col = Ra,  
 weights = wts,  
 col\_rename = "Ra")  
portfolio\_returns\_monthly  
  
wts\_map <- tibble(  
 symbols = c("AAPL", "NFLX"),  
 weights = c(0.5, 0.5)  
)  
wts\_map  
  
  
# 10) Alternative Data Source ####   
#........####  
  
# Alpha Vantage Data: ####  
# To get Market data in Alpha:   
  
# install.packages("alphavantager")  
# browseURL("https://www.alphavantage.co/documentation/")  
# alphak = "K40OVQLGD2QIE4TO"  
  
av\_api\_key("K40OVQLGD2QIE4TO")  
  
# aplicable to currencies  
my\_intraday\_data <- c("FB", "MSFT") %>%  
 tq\_get(get = "alphavantager", av\_fun = "TIME\_SERIES\_INTRADAY", interval = "1min")  
  
eur\_usd <- c("EURUSD") %>%  
 tq\_get(get = "alphavantager", av\_fun = "TIME\_SERIES\_INTRADAY", interval = "1min")  
  
my\_intraday\_data <- c("FB", "MSFT") %>%  
 tq\_get(get = "alphavantager", av\_fun = "TIME\_SERIES\_DAILY\_ADJUSTED")  
  
  
  
#Intraday data for indexes:   
  
#SP500  
sp\_data <- c("SPX") %>%  
 tq\_get(get = "alphavantager", av\_fun = "TIME\_SERIES\_INTRADAY", interval = "1min")  
  
#DOW JONES  
dow\_data <- c("DJIA") %>%  
 tq\_get(get = "alphavantager", av\_fun = "TIME\_SERIES\_INTRADAY", interval = "1min")  
  
#NDAQ ??   
# my\_intraday\_data <- c("COMP") %>%  
# tq\_get(get = "alphavantager", av\_fun = "TIME\_SERIES\_INTRADAY", interval = "1min")  
  
  
EURUSD <- tq\_get(get = "alphavantager", av\_fun = "CURRENCY\_EXCHANGE\_RATE",  
 from\_currency=USD, to\_currency=EUR, interval = "30min",   
 outputsize = compact, apikey = "K40OVQLGD2QIE4TO")  
  
  
  
  
# SMA  
av\_get(symbol = "EURUSD", av\_fun = "SMA", interval = "5min", time\_period = 60, series\_type = "close") %>% print(n=100)  
  
  
  
# INTRADAY DATA   
# output size = "compact" # For daily data.  
# interval = "1min"  
  
# option available also for stocks   
  
setDefaults(getSymbols.av, api.key="K40OVQLGD2QIE4TO")  
  
getSymbols("EURUSD",  
 src="av",  
 api.key="K40OVQLGD2QIE4TO",  
 output.size="full",  
 periodicity="intraday")  
  
getSymbols("GBPUSD",  
 src="av",   
 api.key="K40OVQLGD2QIE4TO",  
 output.size="full",  
 periodicity="intraday")  
  
getSymbols("GBPUSD",  
 src = "av",  
 api.key = "K40OVQLGD2QIE4TO",   
 output.size = "full",  
 periodicity = "daily")  
  
EURUSD %>% tail(); GBPUSD %>% tail()  
  
browseURL( "https://www.rdocumentation.org/packages/quantmod/versions/0.4-15/topics/getSymbols.av")  
  
# More Data sources: ####  
# browseURL("https://towardsdatascience.com/free-financial-data-a122a3cd5531")  
# browseURL("https://www.openquants.com")  
  
  
# SEC ####  
install.packages("finreportr")  
library(finreportr)  
  
#General info :  
AAPL.Info<-CompanyInfo("AAPL")  
#Reports accesion:   
AAPL.reports<-AnnualReports("AAPL")  
  
# Essentially Company financials are organized into 3 segments:   
# Income statement, Balance sheet & Cash flow:   
AAPL.IS <- GetIncome("AAPL", 2017)  
  
#Metrics:  
unique(AAPL.IS$Metric)  
  
# Balance Sheet:   
AAPL.BS<-GetBalanceSheet("AAPL", 2017)   
  
# Cash Flow:   
AAPL.CF<-GetCashFlow("AAPL", 2018)  
  
# Quandl ####  
# browseURL("https://www.quandl.com/")  
  
install.packages("Quandl")  
library(Quandl)  
  
Quandl.api\_key("5qWrGohi2Gx16o4zG-o7")  
from.dat <- as.Date("01/01/2010", format="%d/%m/%Y")  
to.dat <- as.Date("01/01/2019", format="%d/%m/%Y")  
crude.oil.futures<-Quandl("CHRIS/CME\_CL1", start\_date = from.dat, end\_date = to.dat, type="xts")  
plot(crude.oil.futures$Last)  
  
  
  
  
gold.futures<-Quandl("CHRIS/CME\_GC6", start\_date = from.dat, end\_date = to.dat, type="xts")  
plot(gold.futures$Last)  
  
  
gold\_futures <-   
 Quandl("CHRIS/CME\_GC6", api\_key="5qWrGohi2Gx16o4zG-o7")  
  
fred.GDP <- Quandl("FRED/GDP", type = "xts")  
plot(fred.GDP)  
  
# Slice & Dice:   
GDP = Quandl("FRED/GDP", start\_date="2001-12-31", end\_date="2005-12-31", type = "xts")   
  
# sampling data:   
Quandl("FRED/GDP", collapse="annual")  
Quandl("FRED/GDP", transform="rdiff")  
  
  
mydata = Quandl.datatable("ZACKS/FC",  
 ticker=c("AAPL", "MSFT"),  
 per\_end\_date.gt="2015-01-01",  
 qopts.columns=c("m\_ticker", "per\_end\_date", "tot\_revnu"))  
  
  
# Tables wth GT   
  
  
# Define the start and end dates for the data range  
  
start\_date <- "2010-06-07"  
end\_date <- "2010-06-14"  
  
# Create a gt table based on preprocessed  
  
# `sp500` table data  
  
sp500 %>%  
 dplyr::filter(date >= start\_date & date <= end\_date) %>%  
 dplyr::select(-adj\_close) %>%  
 gt() %>%  
 tab\_header(  
 title = "S&P 500",  
 subtitle = glue::glue("{start\_date} to {end\_date}")  
 ) %>%  
   
 fmt\_date(  
 columns = vars(date),  
 date\_style = 3  
 ) %>%  
 fmt\_currency(  
 columns = vars(open, high, low, close),  
 currency = "USD"  
 ) %>%  
   
 fmt\_number(  
 columns = vars(volume),  
 suffixing = TRUE  
 )  
  
  
browseURL("https://rpubs.com/the5ac/all\_seasons\_portfolio\_part1")  
  
  
# 11) Financial Maths ####  
# .......####  
  
# 11.1) Modelling a cash flow ####  
#https://bookdown.org/wfoote01/faur/r-warm-ups-in-finance.html  
  
rates <- c(0.06, 0.07, 0.05, 0.09, 0.09,   
 0.08, 0.08, 0.08)  
t <- seq(1, 8)  
  
(pv.1 <- sum(1/(1 + rates)^t))  
  
### MQF - MBA Thesis. ----   
  
fth <- read.csv("fth.csv")   
rfth <- fth %>% filter(STATUS %in% c("Approved","Completed","Processing"))  
perfomance<-rfth %>% group\_by(TYPE) %>% summarise(n=n(),Amount=sum(AMOUNT))  
d <- perfomance %>% slice(1) %>% pull(Amount)  
w<- perfomance %>% slice(2) %>% pull(Amount)  
pl <- w-d  
trm <- 3250  
ppl<-pl\*trm