Integraciones del paquete de análisis cuantitativo R en tidyquant

Matt Dancho

2023-03-31

> Funciones que aprovechan la funcionalidad de análisis cuantitativo de `xts`, `zoo`, `quantmod`, `TTR`y`PerformanceAnalytics`

Descripción general

Existe una amplia gama de funciones de análisis cuantitativo útiles que funcionan con objetos de series temporales. El problema es que muchas de estas *maravillosas* funciones no funcionan con marcos de datos ni con el `tidyverse`flujo de trabajo. ¡Eso es hasta ahora! El `tidyquant`paquete integra las funciones más útiles de los paquetes `xts`, `zoo`, `quantmod`, `TTR`y `PerformanceAnalytics`. Esta viñeta se centra en las siguientes *funciones principales* para demostrar cómo funciona la integración con los paquetes financieros cuantitativos:

- Transmutar `tq_transmute()`: Devuelve un nuevo marco de datos ordenado, normalmente con una periodicidad diferente a la de la entrada.
- Mutar `tq_mutate()`: agrega columnas al marco de datos ordenado existente.

Consulte Análisis de rendimiento con tidyquant para obtener una discusión completa sobre el análisis de rendimiento y la atribución de cartera con `tidyquant`.

Requisitos previos

Cargue el `tidyquant`paquete para comenzar.

```
# Loads tidyquant, lubridate, xts, quantmod, TTR
library(tidyverse)
library(tidyquant)
```

1.0 Compatibilidad de funciones

`tq_transmute_fun_options()`devuelve una lista de las **funciones de mutación compatibles** con cada paquete. Discutiremos brevemente estas opciones por paquete.

```
tq_transmute_fun_options() %>% str()
```

```
## List of 5
##  $ zoo                : chr [1:14] "rollapply" "rollapplyr" "rollmax" "rollmax.default" ...
##  $ xts                : chr [1:27] "apply.daily" "apply.monthly" "apply.quarterly"
"apply.weekly" ...
##  $ quantmod           : chr [1:25] "ClCl" "Delt" "HiCl" "Lag" ...
##  $ TTR                : chr [1:64] "ADX" "ALMA" "ATR" "BBands" ...
##  $ PerformanceAnalytics: chr [1:7] "Return.Geltner" "Return.annualized"
"Return.annualized.excess" "Return.clean" ...
```

Funcionalidad del zoológico

```
# Get zoo functions that work with tq_transmute and tq_mutate
tq_transmute_fun_options()$zoo
```

```
##  [1] "rollapply"          "rollapplyr"         "rollmax"
##  [4] "rollmax.default"    "rollmaxr"           "rollmean"
##  [7] "rollmean.default"   "rollmeanr"          "rollmedian"
## [10] "rollmedian.default" "rollmedianr"        "rollsum"
## [13] "rollsum.default"    "rollsumr"
```

Las `zoo`funciones que son compatibles se enumeran arriba. En términos generales, estos son los:

- Funciones de aplicación de rollo:
  - Una función genérica para aplicar una función a los márgenes móviles.
  - Forma: `rollapply(data, width, FUN, ..., by = 1, by.column = TRUE, fill = if (na.pad) NA, na.pad = FALSE, partial = FALSE, align = c("center", "left", "right"), coredata = TRUE)`.
  - Las opciones incluyen `rollmax`, `rollmean`, `rollmedian`, `rollsum`, etc.

Funcionalidad xts

```
# Get xts functions that work with tq_transmute and tq_mutate
tq_transmute_fun_options()$xts
```

```
##  [1] "apply.daily"     "apply.monthly"   "apply.quarterly" "apply.weekly"
##  [5] "apply.yearly"    "diff.xts"        "lag.xts"         "period.apply"
```

```
##  [9] "period.max"      "period.min"      "period.prod"     "period.sum"
## [13] "periodicity"     "to.daily"        "to.hourly"       "to.minutes"
## [17] "to.minutes10"    "to.minutes15"    "to.minutes3"     "to.minutes30"
## [21] "to.minutes5"     "to.monthly"      "to.period"       "to.quarterly"
## [25] "to.weekly"       "to.yearly"       "to_period"
```

Las `xts`funciones que son compatibles se enumeran arriba. En términos generales, estos son los:

- Funciones de aplicación de período:
  - Aplicar una función a un segmento de tiempo (por ejemplo `max`, `min`, `mean`, etc.).
  - Forma: `apply.daily(x, FUN, ...)`.
  - Las opciones incluyen `apply.daily`, `weekly`, `monthly`, `quarterly`, `yearly`.
- Funciones hasta el período:
  - Convierta una serie temporal en una serie temporal de menor periodicidad (por ejemplo, convierta una periodicidad diaria en mensual).
  - Forma: `to.period(x, period = 'months', k = 1, indexAt, name = NULL, OHLC = TRUE, ...)`.
  - Las opciones incluyen `to.minutes`, `hourly`, `daily`, `weekly`, `monthly`, `quarterly`, `yearly`.
  - **Nota 1 (Importante)** : La estructura de devolución es diferente para `to.period`los formularios y `to.monthly` (`to.weekly`,, `to.quarterly`etc.). `to.period`devuelve una fecha, mientras que `to.months` devuelve un carácter MON AAAA. Es mejor utilizarlo `to.period`si desea trabajar con series temporales a través de `lubridate`.

Funcionalidad cuántica

```
# Get quantmod functions that work with tq_transmute and tq_mutate
tq_transmute_fun_options()$quantmod
```

```
##  [1] "ClCl"            "Delt"            "HiCl"            "Lag"
##  [5] "LoCl"            "LoHi"            "Next"            "OpCl"
##  [9] "OpHi"            "OpLo"            "OpOp"            "allReturns"
## [13] "annualReturn"    "dailyReturn"     "monthlyReturn"   "periodReturn"
## [17] "quarterlyReturn" "seriesAccel"     "seriesDecel"     "seriesDecr"
## [21] "seriesHi"        "seriesIncr"      "seriesLo"        "weeklyReturn"
## [25] "yearlyReturn"
```

Las `quantmod`funciones que son compatibles se enumeran arriba. En términos generales, estos son los:

- Funciones de cambio porcentual (Delt) y retraso
  - Delt: `Delt(x1, x2 = NULL, k = 0, type = c("arithmetic", "log"))`
    - Variaciones de Delt: ClCl, HiCl, LoCl, LoHi, OpCl, OpHi, OpLo, OpOp
    - Forma:`OpCl(OHLC)`
  - Lag: `Lag(x, k = 1)`/ Siguiente: `Next(x, k = 1)` (También puede usar `dplyr::lag`y `dplyr::lead`)
- Funciones de devolución del período:
  - Obtenga los rendimientos aritméticos o logarítmicos para diversas periodicidades, que incluyen diaria, semanal, mensual, trimestral y anual.
  - Forma: `periodReturn(x, period = 'monthly', subset = NULL, type = 'arithmetic', leading = TRUE, ...)`
- Funciones de la serie:
  - Valores de retorno que describen la serie. Las opciones incluyen describir los aumentos/disminuciones, la aceleración/desaceleración y alto/bajo.
  - Formas: `seriesHi(x)`, `seriesIncr(x, thresh = 0, diff. = 1L)`, `seriesAccel(x)`

Funcionalidad TTR

```
# Get TTR functions that work with tq_transmute and tq_mutate
tq_transmute_fun_options()$TTR
```

```
##  [1] "ADX"                "ALMA"               "ATR"
##  [4] "BBands"             "CCI"                "CLV"
##  [7] "CMF"                "CMO"                "CTI"
## [10] "DEMA"               "DPO"                "DVI"
## [13] "DonchianChannel"    "EMA"                "EMV"
## [16] "EVWMA"              "GMMA"               "HMA"
## [19] "KST"                "MACD"               "MFI"
## [22] "OBV"                "PBands"             "ROC"
## [25] "RSI"                "SAR"                "SMA"
## [28] "SMI"                "SNR"                "TDI"
## [31] "TRIX"               "VHF"                "VMA"
## [34] "VWAP"               "VWMA"               "WMA"
## [37] "WPR"                "ZLEMA"              "ZigZag"
## [40] "adjRatios"          "aroon"              "chaikinAD"
## [43] "chaikinVolatility"  "growth"             "keltnerChannels"
## [46] "lags"               "momentum"           "rollSFM"
## [49] "runCor"             "runCov"             "runMAD"
## [52] "runMax"             "runMean"            "runMedian"
## [55] "runMin"             "runPercentRank"     "runSD"
## [58] "runSum"             "runVar"             "stoch"
## [61] "ultimateOscillator" "volatility"         "wilderSum"
## [64] "williamsAD"
```

Aquí hay una breve descripción de las funciones más populares de `TTR`:

- Índice de movimiento direccional de Welles Wilder:
  - `ADX(HLC, n = 14, maType, ...)`
- Bandas de Bollinger:
  - `BBands(HLC, n = 20, maType, sd = 2, ...)`: Bandas de Bollinger
- Tasa de cambio/impulso:
  - `ROC(x, n = 1, type = c("continuous", "discrete"), na.pad = TRUE)`: Tasa de cambio
  - `momentum(x, n = 1, na.pad = TRUE)`: Impulso
- Medias móviles (maType):
  - `SMA(x, n = 10, ...)`: Media móvil simple
  - `EMA(x, n = 10, wilder = FALSE, ratio = NULL, ...)`: Media móvil exponencial
  - `DEMA(x, n = 10, v = 1, wilder = FALSE, ratio = NULL)`: Media móvil exponencial doble
  - `WMA(x, n = 10, wts = 1:n, ...)`: Media móvil ponderada
  - `EVWMA(price, volume, n = 10, ...)`: Media móvil elástica ponderada por volumen
  - `ZLEMA(x, n = 10, ratio = NULL, ...)`: Media móvil exponencial de retardo cero
  - `VWAP(price, volume, n = 10, ...)`: Precio promedio móvil ponderado por volumen
  - `VMA(x, w, ratio = 1, ...)`: Media móvil de longitud variable
  - `HMA(x, n = 20, ...)`: Media móvil del casco
  - `ALMA(x, n = 9, offset = 0.85, sigma = 6, ...)`: Arnaud Legoux Media móvil
- Oscilador MACD:
  - `MACD(x, nFast = 12, nSlow = 26, nSig = 9, maType, percent = TRUE, ...)`
- Índice de Fuerza Relativa:
  - `RSI(price, n = 14, maType, ...)`
- correrDiversión:
  - `runSum(x, n = 10, cumulative = FALSE)`: devuelve sumas durante una ventana móvil de n períodos.

    - `runMin(x, n = 10, cumulative = FALSE)`: devuelve mínimos durante una ventana móvil de n períodos.
    - `runMax(x, n = 10, cumulative = FALSE)`: devuelve máximos en una ventana móvil de n períodos.
    - `runMean(x, n = 10, cumulative = FALSE)`: devuelve medios sobre una ventana móvil de n períodos.
    - `runMedian(x, n = 10, non.unique = "mean", cumulative = FALSE)`: devuelve medianas durante una ventana móvil de n períodos.
    - `runCov(x, y, n = 10, use = "all.obs", sample = TRUE, cumulative = FALSE)`: devuelve covarianzas sobre una ventana móvil de n períodos.
    - `runCor(x, y, n = 10, use = "all.obs", sample = TRUE, cumulative = FALSE)`: devuelve correlaciones durante una ventana móvil de n períodos.
    - `runVar(x, y = NULL, n = 10, sample = TRUE, cumulative = FALSE)`: devuelve variaciones durante una ventana móvil de n períodos.
    - `runSD(x, n = 10, sample = TRUE, cumulative = FALSE)`: devuelve desviaciones estándar durante una ventana móvil de n períodos.
    - `runMAD(x, n = 10, center = NULL, stat = "median", constant = 1.4826, non.unique = "mean", cumulative = FALSE)`: devuelve las desviaciones absolutas mediana/media durante una ventana móvil de n períodos.
    - `wilderSum(x, n = 10)`: devuelve una suma ponderada al estilo de Welles Wilder sobre una ventana móvil de n períodos.
- Oscilador estocástico / Índice de momento estocástico:
    - `stoch(HLC, nFastK = 14, nFastD = 3, nSlowD = 3, maType, bounded = TRUE, smooth = 1, ...)`: Oscilador estocástico
    - `SMI(HLC, n = 13, nFast = 2, nSlow = 25, nSig = 9, maType, bounded = TRUE, ...)`: Índice de impulso estocástico

Funcionalidad de análisis de rendimiento

```
# Get PerformanceAnalytics functions that work with tq_transmute and tq_mutate
tq_transmute_fun_options()$PerformanceAnalytics
```

```
## [1] "Return.Geltner"           "Return.annualized"
## [3] "Return.annualized.excess" "Return.clean"
## [5] "Return.cumulative"        "Return.excess"
## [7] "zerofill"
```

Todas las `PerformanceAnalytics`funciones de mutación se ocupan de devoluciones:

- `Return.annualized`y `Return.annualized.excess`: toma los rendimientos del período y los consolida en rendimientos anualizados.
- `Return.clean`: elimina los valores atípicos de las devoluciones
- `Return.excess`: Elimina la tasa libre de riesgo de los rendimientos para generar rendimientos superiores a la tasa libre de riesgo
- `zerofill`: Se utiliza para reemplazar `NA`valores con ceros.

2.0 Poder cuantitativo en acción

Revisaremos algunos ejemplos, pero primero obtengamos algunos datos. `FANG`Se utilizará el conjunto de datos que consta de los precios de las acciones de FB, AMZN, NFLX y GOOG desde principios de 2013 hasta finales de 2016.

```
data("FANG")

FANG
```

```
## # A tibble: 4,032 x 8
##    symbol date        open  high   low close    volume adjusted
##    <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
##  1 FB     2013-01-02  27.4  28.2  27.4  28    69846400     28
##  2 FB     2013-01-03  27.9  28.5  27.6  27.8  63140600     27.8
##  3 FB     2013-01-04  28.0  28.9  27.8  28.8  72715400     28.8
##  4 FB     2013-01-07  28.7  29.8  28.6  29.4  83781800     29.4
##  5 FB     2013-01-08  29.5  29.6  28.9  29.1  45871300     29.1
##  6 FB     2013-01-09  29.7  30.6  29.5  30.6 104787700     30.6
##  7 FB     2013-01-10  30.6  31.5  30.3  31.3  95316400     31.3
##  8 FB     2013-01-11  31.3  32.0  31.1  31.7  89598000     31.7
##  9 FB     2013-01-14  32.1  32.2  30.6  31.0  98892800     31.0
## 10 FB     2013-01-15  30.6  31.7  29.9  30.1 173242600     30.1
## # ... with 4,022 more rows
```

Ejemplo 1: utilizar quantmod periodReturn para convertir precios en rentabilidad

La `quantmod::periodReturn()`función genera retornos por periodicidad. Revisaremos un par de casos de uso.

Ejemplo 1A: Obtener y registrar las rentabilidades anuales

Queremos utilizar la columna de precios de cierre ajustados (ajustada para divisiones de acciones, lo que puede hacer que parezca que una acción tiene un mal desempeño si se incluye una división). Establecimos `select = adjusted`. Investigamos la `periodReturn`función y descubrimos que acepta `type = "arithmetic"`y `period = "yearly"`, que devuelve los rendimientos anuales.

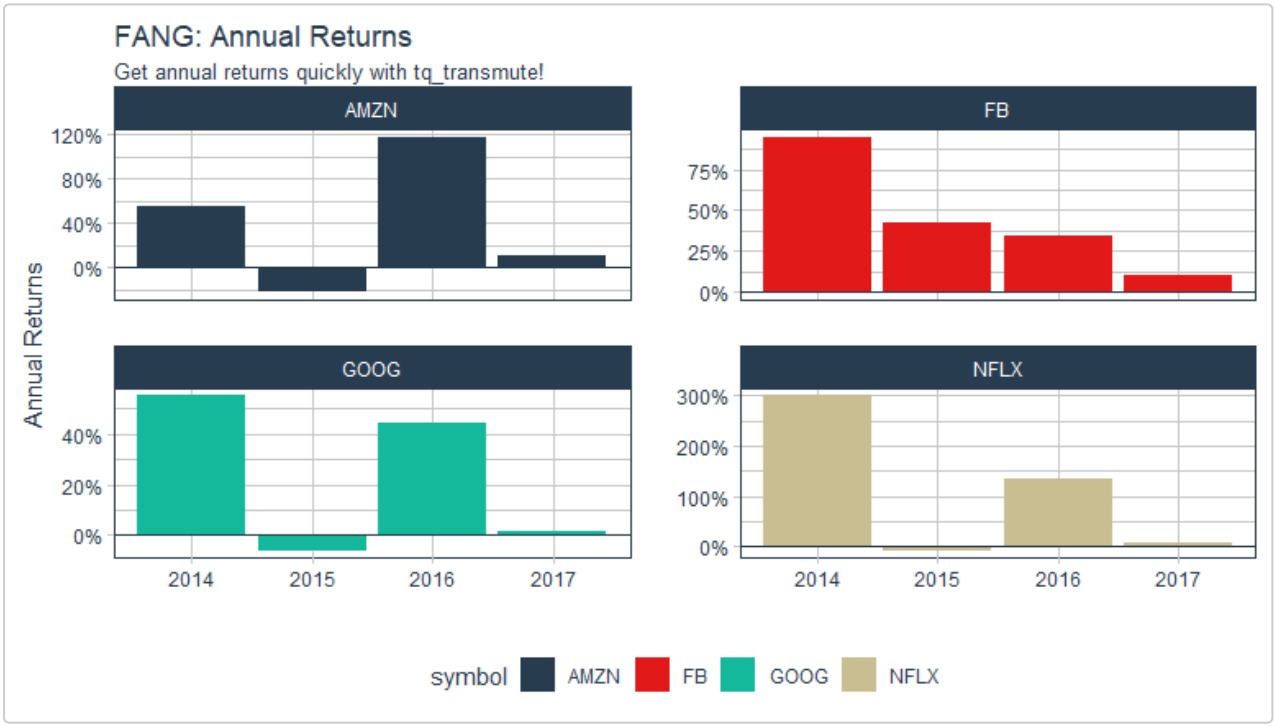
```
FANG_annual_returns <- FANG %>%
    group_by(symbol) %>%
    tq_transmute(select     = adjusted,
                 mutate_fun = periodReturn,
                 period     = "yearly",
                 type       = "arithmetic")
FANG_annual_returns
```

```
## # A tibble: 16 x 3
## # Groups:   symbol [4]
##    symbol date       yearly.returns
##    <chr>  <date>              <dbl>
##  1 FB     2013-12-31         0.952
##  2 FB     2014-12-31         0.428
##  3 FB     2015-12-31         0.341
##  4 FB     2016-12-30         0.0993
##  5 AMZN   2013-12-31         0.550
##  6 AMZN   2014-12-31        -0.222
```

```
##  7 AMZN   2015-12-31         1.18
##  8 AMZN   2016-12-30         0.109
##  9 NFLX   2013-12-31         3.00
## 10 NFLX   2014-12-31        -0.0721
## 11 NFLX   2015-12-31         1.34
## 12 NFLX   2016-12-30         0.0824
## 13 GOOG   2013-12-31         0.550
## 14 GOOG   2014-12-31        -0.0597
## 15 GOOG   2015-12-31         0.442
## 16 GOOG   2016-12-30         0.0171
```

Graficar los rendimientos anuales es sólo un uso rápido del `ggplot2`paquete.

```
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()
```

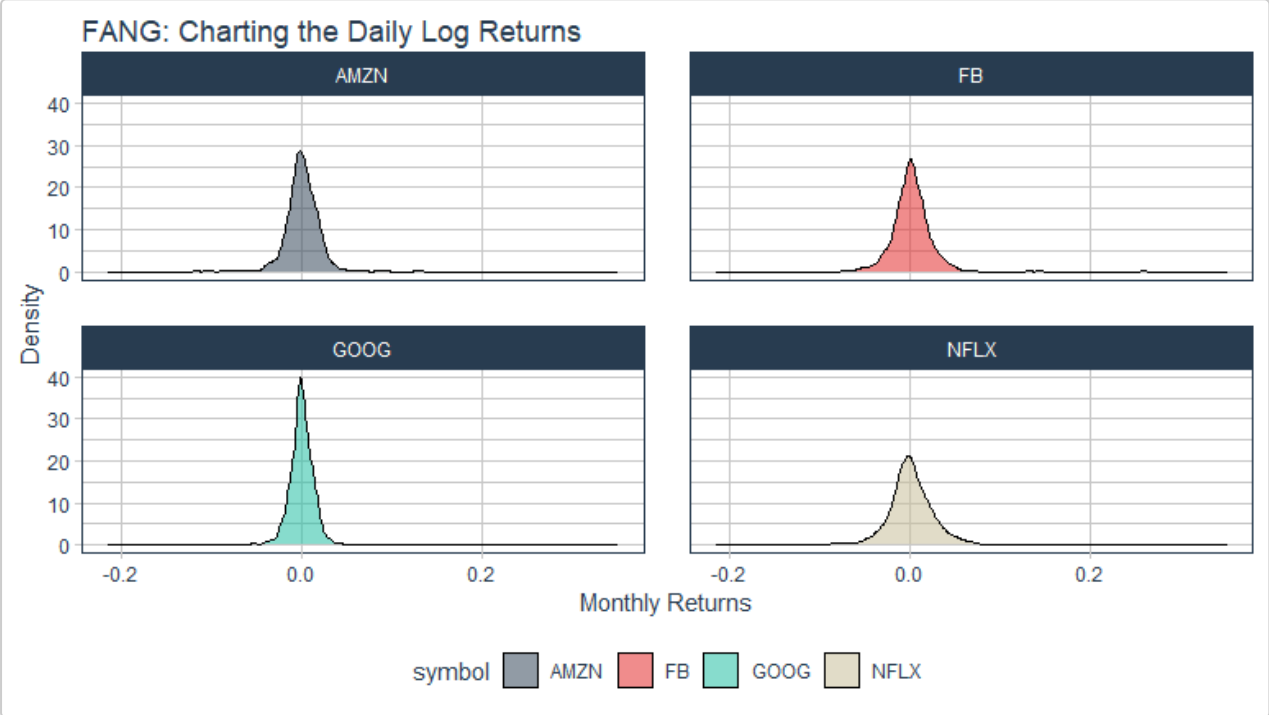


Ejemplo 1B: Obtener devoluciones de registros diarios

Los retornos de registros diarios siguen un enfoque similar. Normalmente uso una función de transmutación, `tq_transmute`porque la `periodReturn`función acepta diferentes opciones de periodicidad, y cualquier cosa que no sea diaria hará estallar una mutación. Pero, en nuestra situación, la periodicidad de los rendimientos del período es la misma que la periodicidad de los precios de las acciones (ambas diarias), por lo que podemos usar cualquiera de las dos. Queremos utilizar la columna de precios de cierre ajustados (ajustada para divisiones de acciones, lo que puede hacer que parezca que una acción tiene un mal desempeño si se incluye una división), por lo que configuramos `select = adjusted`. Investigamos la

`periodReturn`función y descubrimos que acepta `type = "log"`y `period = "daily"`, que devuelve los registros diarios.

```
FANG_daily_log_returns <- FANG %>%
    group_by(symbol) %>%
    tq_transmute(select     = adjusted,
                 mutate_fun = periodReturn,
                 period     = "daily",
                 type       = "log",
                 col_rename = "monthly.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)
```



Ejemplo 2: utilice xts to.period para cambiar la periodicidad de diaria a mensual

La `xts::to.period`función se utiliza para la agregación de periodicidad (conversión de una periodicidad de nivel inferior a un nivel superior, como minutos a horas o meses a años). Debido a que buscamos una estructura de retorno que esté en una escala de tiempo diferente a la de entrada (diaria versus semanal), necesitamos usar una función de transmutación. Seleccionamos `tq_transmute()`y pasamos las columnas apertura, alto, bajo, cierre y volumen vía `select = open:volume`. Al mirar la documentación de `to.period`, vemos que acepta un `period`argumento que podemos establecer en `"weeks"`. El resultado son los datos OHLCV devueltos con las fechas cambiadas a un día por semana.

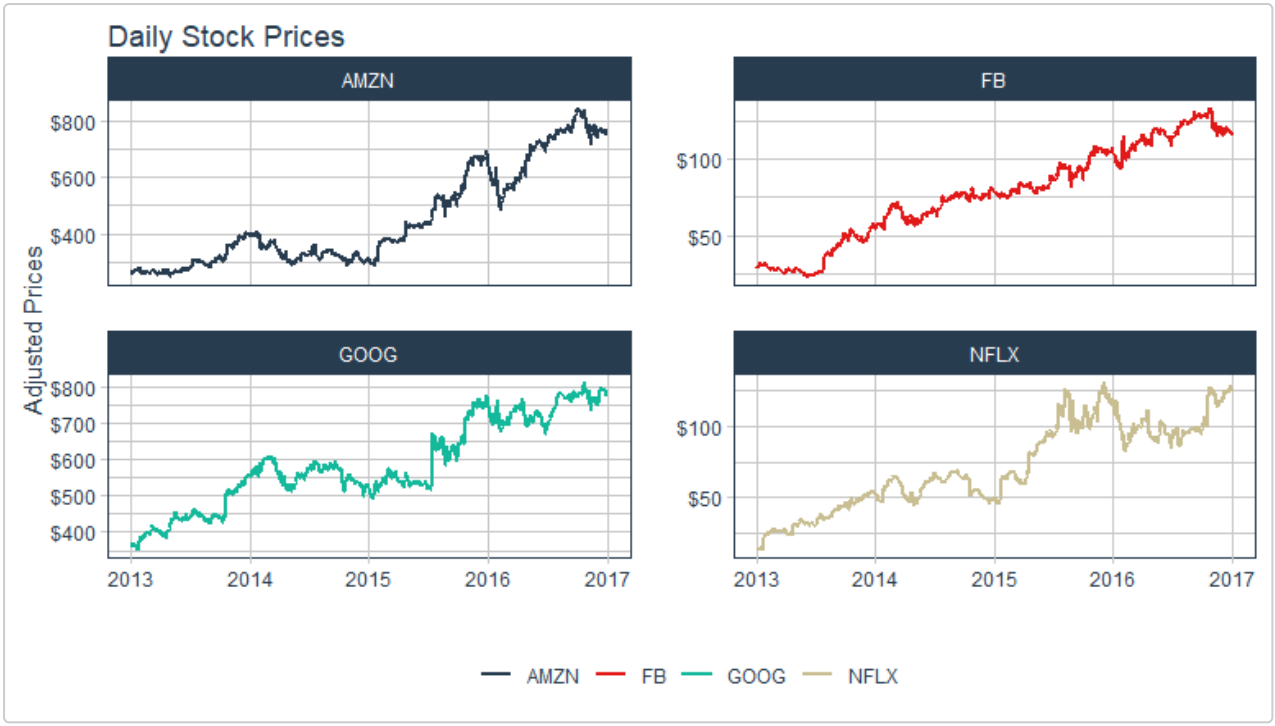
```
FANG %>%
    group_by(symbol) %>%
    tq_transmute(select     = open:volume,
                 mutate_fun = to.period,
                 period     = "months")
```

```
## # A tibble: 192 x 7
## # Groups:   symbol [4]
##    symbol date        open  high   low close    volume
##    <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>
##  1 FB     2013-01-31  29.2  31.5  28.7  31.0 190744900
##  2 FB     2013-02-28  26.8  27.3  26.3  27.2  83027800
##  3 FB     2013-03-28  26.1  26.2  25.5  25.6  28585700
##  4 FB     2013-04-30  27.1  27.8  27.0  27.8  36245700
##  5 FB     2013-05-31  24.6  25.0  24.3  24.4  35925000
##  6 FB     2013-06-28  24.7  25.0  24.4  24.9  96778900
##  7 FB     2013-07-31  38.0  38.3  36.3  36.8 154828700
##  8 FB     2013-08-30  42.0  42.3  41.1  41.3  67735100
##  9 FB     2013-09-30  50.1  51.6  49.8  50.2 100095000
## 10 FB     2013-10-31  47.2  52    46.5  50.2 248809000
## # ... with 182 more rows
```

Un caso de uso común es reducir la cantidad de puntos para suavizar los gráficos de series temporales. Veamos la diferencia entre gráficos diarios y mensuales.

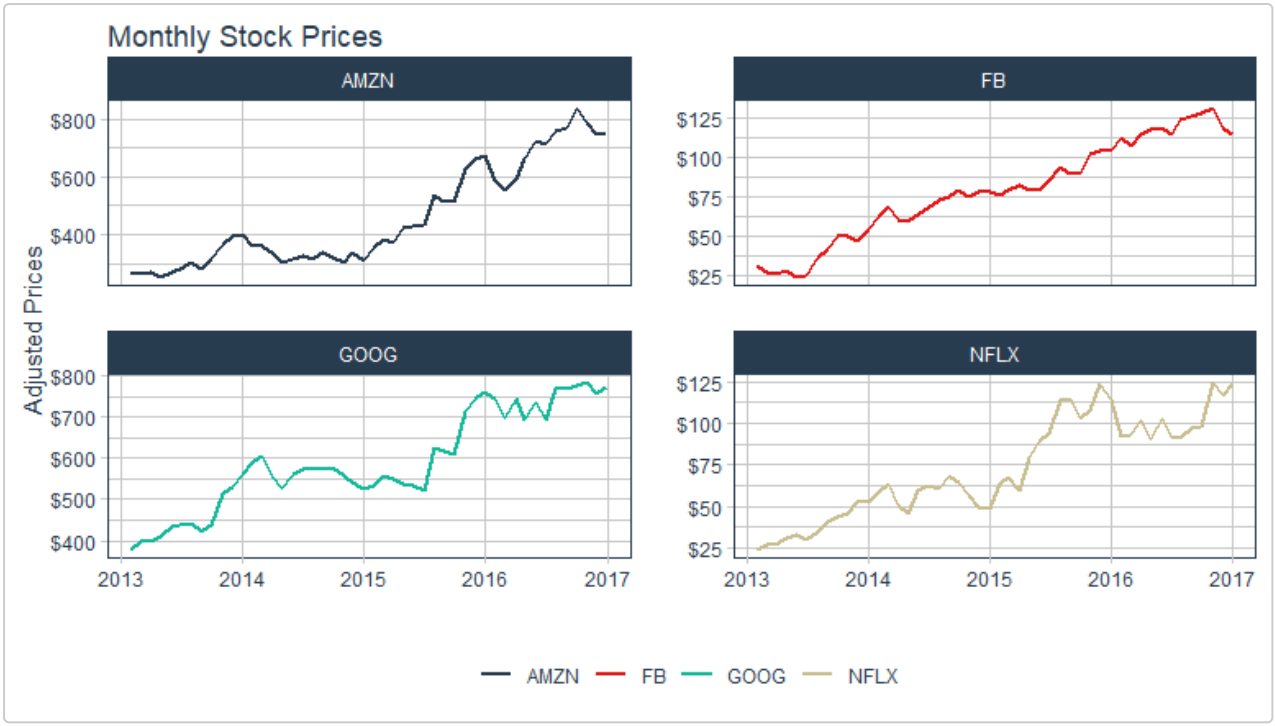
Sin agregación de periodicidad

```
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()
```



Con agregación de periodicidad mensual

```
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()
```



Ejemplo 3: utilice TTR runCor para visualizar correlaciones continuas de rendimientos

Las correlaciones de rendimiento son una forma común de analizar en qué medida un activo o cartera imita un índice o fondo de referencia. Necesitaremos un conjunto de rendimientos tanto para las acciones como para el valor base. La acción será el `FANG`conjunto de datos y la línea de base será el sector tecnológico Spdr XLK. Tenemos los precios de las acciones "FANG", por lo que utilizamos `tq_get`para recuperar los precios "XLK". Los rendimientos se pueden calcular a partir de los precios "ajustados" utilizando el proceso del Ejemplo 1.

```
# 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 = "2013-01-01",
           to   = "2016-12-31") %>%
    tq_transmute(select     = adjusted,
                 mutate_fun = periodReturn,
                 period     = "monthly")
```

A continuación, una los rendimientos de los activos con los rendimientos de referencia por fecha.

```
returns_joined <- left_join(FANG_returns_monthly,
                            baseline_returns_monthly,
                            by = "date")
returns_joined
```

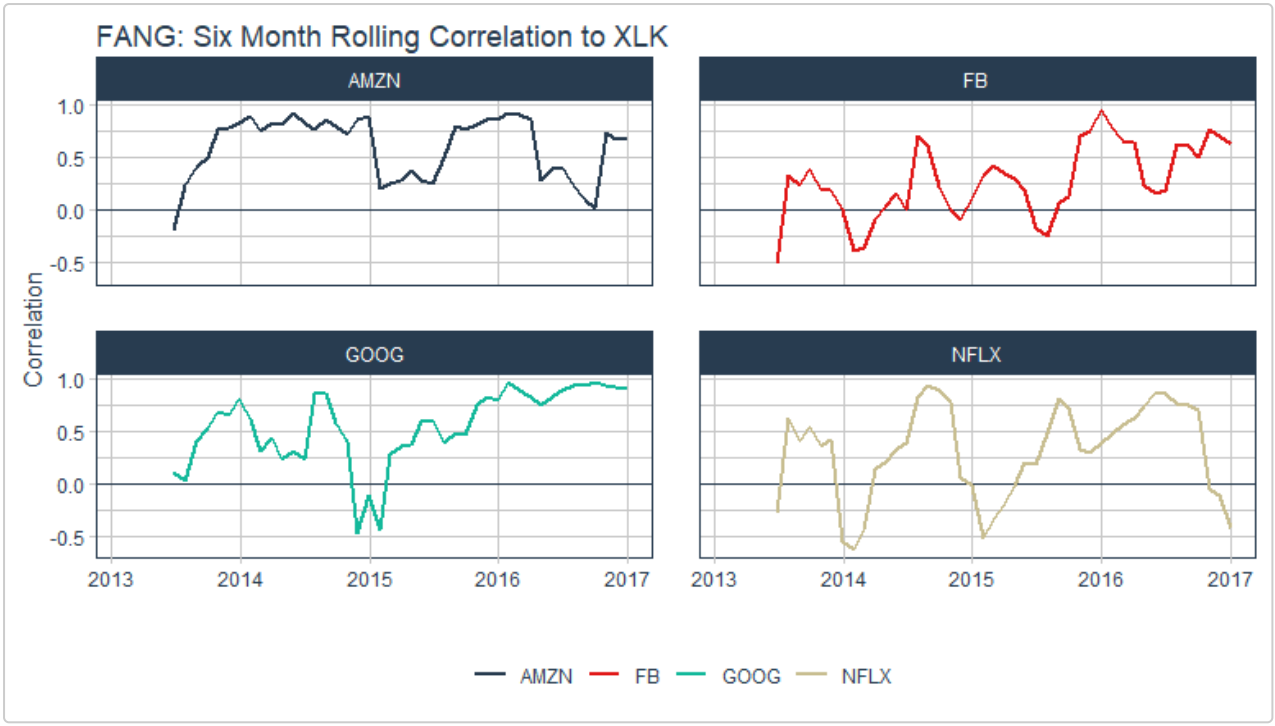
```
## # A tibble: 192 x 4
## # Groups:   symbol [4]
##    symbol date       monthly.returns.x monthly.returns.y
##    <chr>  <date>                 <dbl>             <dbl>
##  1 FB     2013-01-31           0.106            -0.0138
##  2 FB     2013-02-28          -0.120             0.00782
##  3 FB     2013-03-28          -0.0613            0.0258
##  4 FB     2013-04-30           0.0856            0.0175
##  5 FB     2013-05-31          -0.123             0.0279
##  6 FB     2013-06-28           0.0218           -0.0289
##  7 FB     2013-07-31           0.479             0.0373
##  8 FB     2013-08-30           0.122            -0.0104
##  9 FB     2013-09-30           0.217             0.0253
## 10 FB     2013-10-31          -0.000398          0.0502
## # ... with 182 more rows
```

La `TTR::runCor`función se puede utilizar para evaluar correlaciones rodantes utilizando el patrón xy. Al observar la documentación (`?runCor`), podemos ver que los argumentos incluyen `x`y `y`junto con algunos argumentos adicionales, incluido `n`el ancho de la correlación móvil. Debido a que la escala es mensual, optaremos `n = 6`por una correlación móvil de 6 meses. El `col_rename`argumento permite cambiar fácilmente el nombre de las columnas de salida.

```
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")
```

Y podemos trazar las correlaciones móviles para las acciones de FANG.

```
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) +
    labs(title = "FANG: Six Month Rolling Correlation to XLK",
         x = "", y = "Correlation", color = "") +
    facet_wrap(~ symbol, ncol = 2) +
    theme_tq() +
    scale_color_tq()
```



Ejemplo 4: Utilice TTR MACD para visualizar la divergencia de convergencia de la media móvil

Al revisar las opciones disponibles en el `TTR`paquete, vemos que `MACD`nos dará la Media Móvil de Convergencia y Divergencia (MACD). Al investigar la documentación, la devolución tiene la misma periodicidad que la entrada y las funciones funcionan con funciones OHLC, por lo que podemos usar `tq_mutate()`. MACD requiere un precio, por lo que seleccionamos `close`.

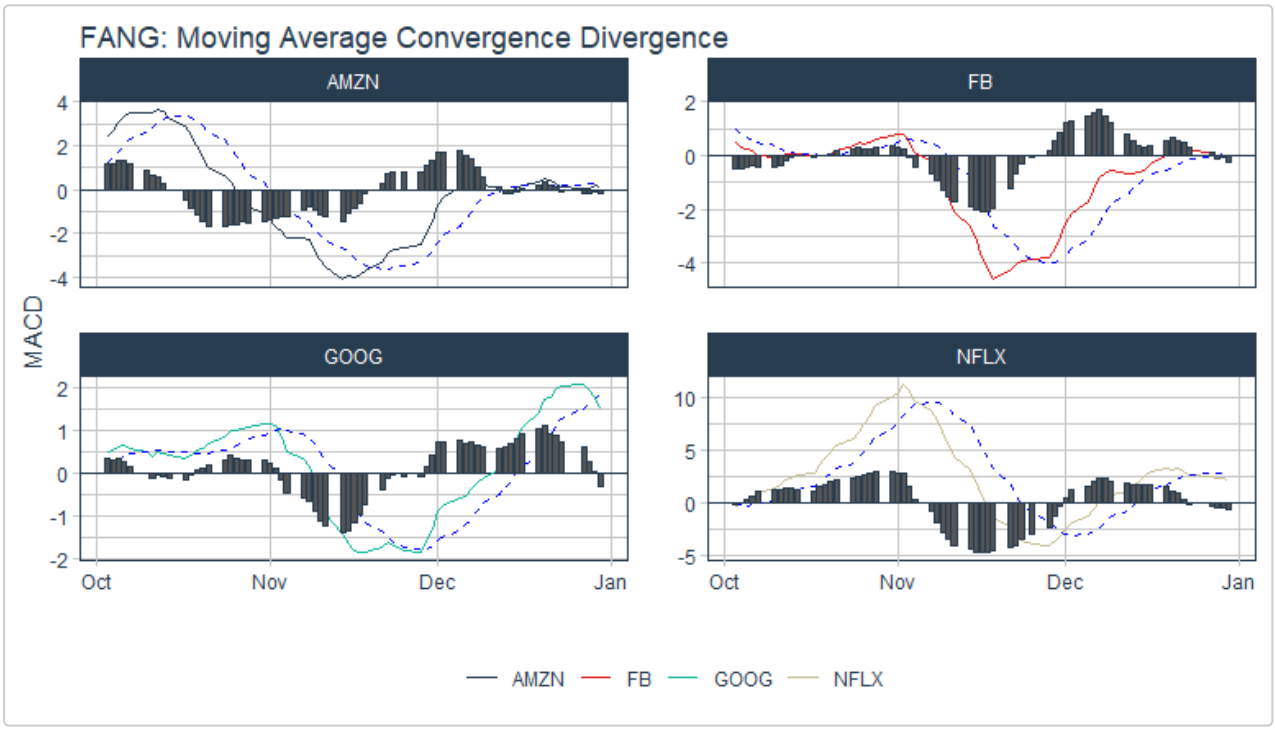
```
FANG_macd <- FANG %>%
    group_by(symbol) %>%
    tq_mutate(select     = close,
              mutate_fun = MACD,
              nFast      = 12,
              nSlow      = 26,
              nSig       = 9,
              maType     = SMA) %>%
    mutate(diff = macd - signal) %>%
    select(-(open:volume))
FANG_macd
```

```
## # A tibble: 4,032 x 6
## # Groups:   symbol [4]
##    symbol date       adjusted  macd signal  diff
##    <chr>  <date>        <dbl> <dbl>  <dbl> <dbl>
##  1 FB     2013-01-02     28      NA     NA    NA
##  2 FB     2013-01-03     27.8    NA     NA    NA
##  3 FB     2013-01-04     28.8    NA     NA    NA
##  4 FB     2013-01-07     29.4    NA     NA    NA
##  5 FB     2013-01-08     29.1    NA     NA    NA
##  6 FB     2013-01-09     30.6    NA     NA    NA
##  7 FB     2013-01-10     31.3    NA     NA    NA
##  8 FB     2013-01-11     31.7    NA     NA    NA
```

```
##  9 FB     2013-01-14     31.0    NA     NA     NA
## 10 FB     2013-01-15     30.1    NA     NA     NA
## # ... with 4,022 more rows
```

Y podemos visualizar los datos así.

```
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()
```



Ejemplo 5: utilice xts apply. Quarterly para obtener el precio máximo y mínimo para cada trimestre

La `xts::apply.quarterly()`función que forma parte del grupo de aplicación de períodos se puede utilizar para aplicar funciones por segmentos de tiempo trimestrales. Debido a que buscamos una estructura de retorno que esté en una escala de tiempo diferente a la de entrada (trimestral versus diaria), necesitamos usar una función de transmutación. Seleccionamos `tq_transmute`y pasamos el precio de cierre usando `select`y enviamos este subconjunto de datos a la `apply.quarterly`función a través del `mutate_fun`argumento. Al mirar la documentación de `apply.quarterly`, vemos que podemos pasar una función al argumento `FUN`. Queremos los valores máximos, así que los configuramos `FUN = max`. El resultado son los trimestres devueltos como una fecha y el precio máximo de cierre durante el trimestre devuelto como un doble.

```
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
```

```
## # A tibble: 64 x 3
## # Groups:   symbol [4]
##    symbol max.close year.qtr
##    <chr>      <dbl> <chr>
##  1 FB          32.5 2013-Q1
##  2 FB          29.0 2013-Q2
##  3 FB          51.2 2013-Q3
##  4 FB          58.0 2013-Q4
##  5 FB          72.0 2014-Q1
##  6 FB          67.6 2014-Q2
##  7 FB          79.0 2014-Q3
##  8 FB          81.4 2014-Q4
##  9 FB          85.3 2015-Q1
## 10 FB          88.9 2015-Q2
## # ... with 54 more rows
```

El mínimo de cada trimestre se puede recuperar de la misma manera. Los marcos de datos se pueden unir `left_join`para obtener el máximo y el mínimo por trimestre.

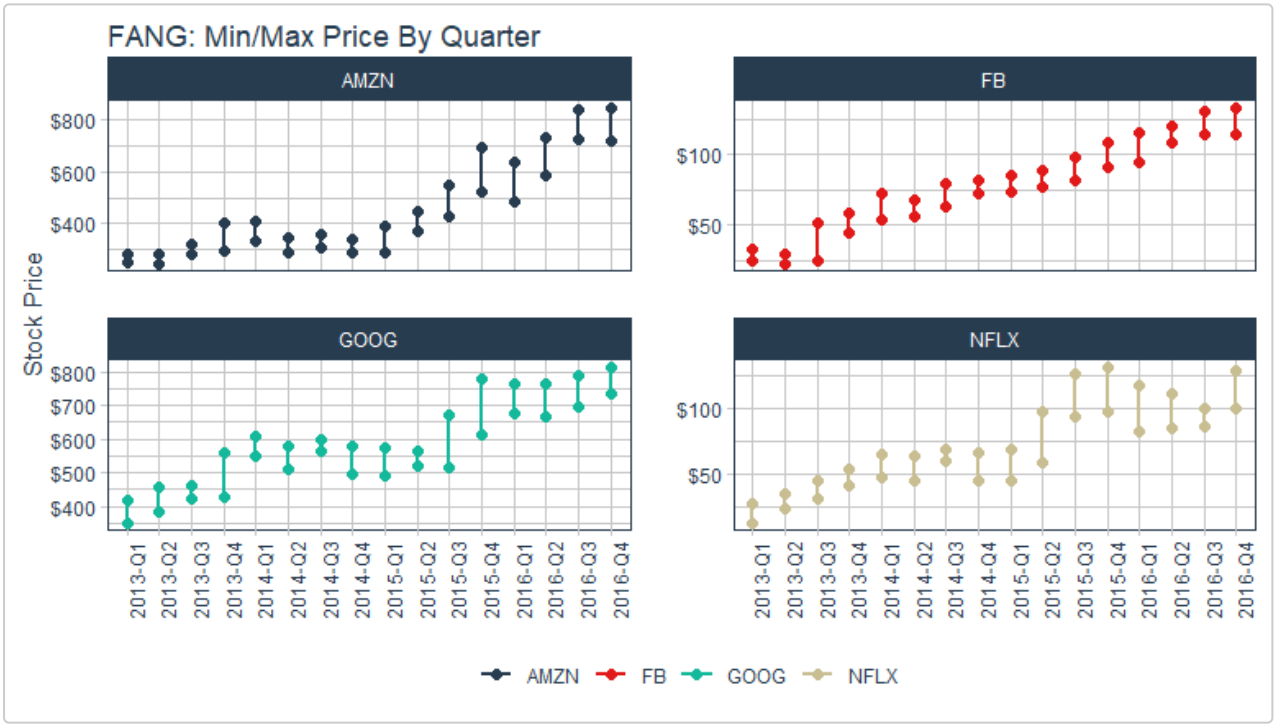
```
FANG_min_by_qtr <- FANG %>%
    group_by(symbol) %>%
    tq_transmute(select     = adjusted,
                 mutate_fun = apply.quarterly,
                 FUN        = min,
                 col_rename = "min.close") %>%
    mutate(year.qtr = paste0(year(date), "-Q", quarter(date))) %>%
    select(-date)

FANG_by_qtr <- left_join(FANG_max_by_qtr, FANG_min_by_qtr,
                         by = c("symbol"   = "symbol",
                                "year.qtr" = "year.qtr"))
FANG_by_qtr
```

```
## # A tibble: 64 x 4
## # Groups:   symbol [4]
##    symbol max.close year.qtr min.close
##    <chr>      <dbl> <chr>        <dbl>
##  1 FB          32.5 2013-Q1       25.1
##  2 FB          29.0 2013-Q2       22.9
##  3 FB          51.2 2013-Q3       24.4
##  4 FB          58.0 2013-Q4       44.8
##  5 FB          72.0 2014-Q1       53.5
##  6 FB          67.6 2014-Q2       56.1
```

```
##  7 FB          79.0 2014-Q3       62.8
##  8 FB          81.4 2014-Q4       72.6
##  9 FB          85.3 2015-Q1       74.1
## 10 FB          88.9 2015-Q2       77.5
## # ... with 54 more rows
```

Y podemos visualizar los datos así.

```
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())
```



Ejemplo 6: utilice zoo rollapply para visualizar una regresión continua

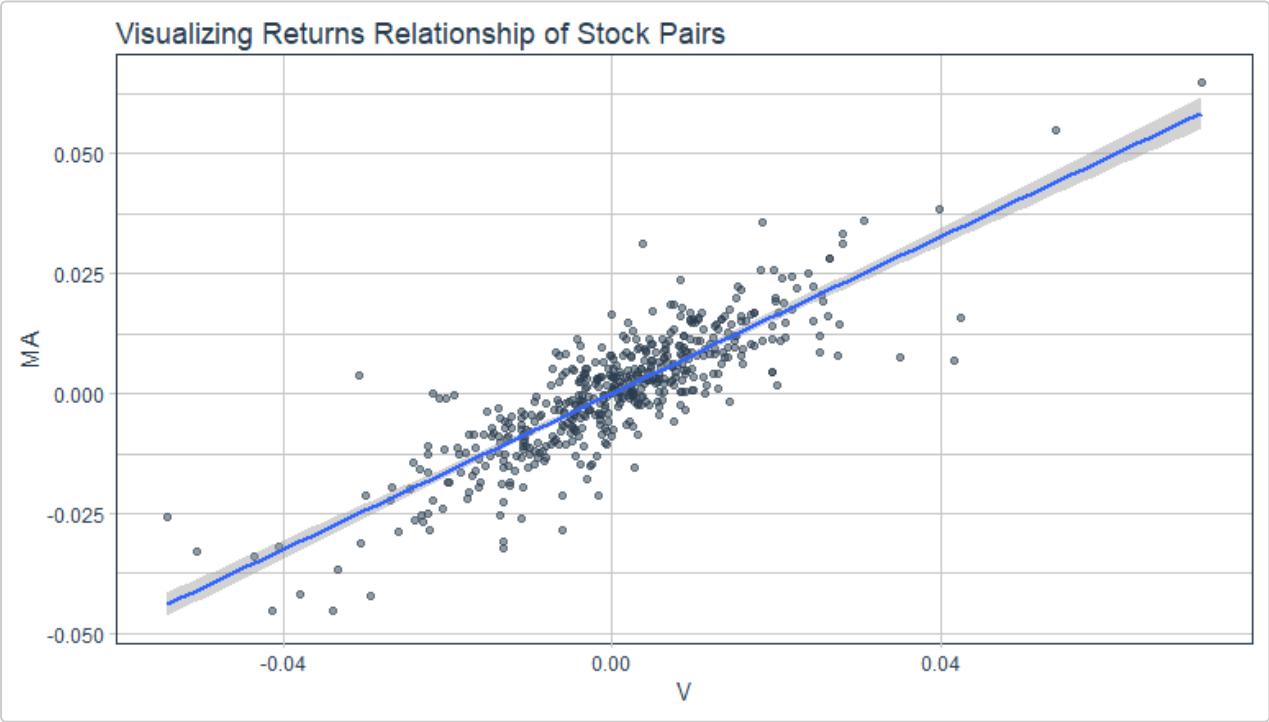
Una buena forma de analizar las relaciones a lo largo del tiempo es utilizar cálculos continuos que comparen dos activos. El comercio de pares es un mecanismo común para activos similares. Si bien no entraremos en un análisis del comercio de pares, analizaremos la relación entre dos activos similares como precursor de un comercio de pares. En este ejemplo analizaremos dos activos similares, Mastercard (MA) y Visa (V) para mostrar la relación mediante regresión.

Antes de analizar una regresión móvil, resulta útil ver la tendencia general de los rendimientos. Para hacer esto, utilizamos `tq_get()`para obtener los precios de las acciones de los activos y `tq_transmute()`transformar los precios diarios en rendimientos diarios. Recopilaremos los datos y los visualizaremos mediante un diagrama de dispersión.

```
# 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)
```

Podemos visualizar la relación entre los rendimientos de los pares de acciones así.

```
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()
```



Podemos obtener estadísticas sobre la relación a partir de la `lm` función. El modelo está altamente correlacionado con un valor p de cero esencial. La estimación del coeficiente para V (Coeficiente 1) es 0,8134, lo que indica una relación positiva, lo que significa que a medida que V aumenta, MA también tiende a aumentar.

```
    lm(MA ~ V, data = stock_pairs) %>%
        summary()
```

```
##
## Call:
## lm(formula = MA ~ V, data = stock_pairs)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.026957 -0.003966  0.000215  0.003965  0.028946
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0001130  0.0003097   0.365    0.715
## V           0.8133666  0.0226393  35.927   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.00695 on 502 degrees of freedom
## Multiple R-squared:   0.72,  Adjusted R-squared:  0.7194
## F-statistic:  1291 on 1 and 502 DF,  p-value: < 2.2e-16
```

Si bien esto caracteriza la relación general, falta el aspecto temporal. Afortunadamente, podemos usar la `rollapply`función del `zoo`paquete para trazar una regresión móvil, mostrando cómo el coeficiente del modelo varía de forma móvil a lo largo del tiempo. Calculamos regresiones móviles `tq_mutate()`en dos pasos adicionales:

1. Crear una función personalizada
2. Aplicar la función con `tq_mutate(mutate_fun = rollapply)`

Primero, cree una función de regresión personalizada. Un punto importante es que los "datos" se pasarán a la función de regresión como un `xts`objeto. La `timetk::tk_tbl`función se encarga de convertir a un marco de datos.

```
regr_fun <- function(data) {
    coef(lm(MA ~ V, data = timetk::tk_tbl(data, silent = TRUE)))
}
```

Ahora podemos usar `tq_mutate()`para aplicar la función de regresión personalizada sobre una ventana móvil usando `rollapply` el `zoo`paquete. Internamente, el `returns_combined`marco de datos se pasa automáticamente al `data`argumento de la `rollapply` función. Todo lo que necesita especificar es el `mutate_fun = rollapply`y cualquier argumento adicional necesario para aplicar la `rollapply`función. Especificaremos una ventana de 90 días mediante `width = 90`. El `FUN`argumento es nuestra función de regresión personalizada `regr_fun`. Es extremadamente importante especificar `by.column = FALSE`, lo que indica `rollapply`que se debe realizar el cálculo utilizando los datos como un todo en lugar de aplicar la función a cada columna de forma independiente. El `col_rename`argumento se utiliza para cambiar el nombre de las columnas agregadas.

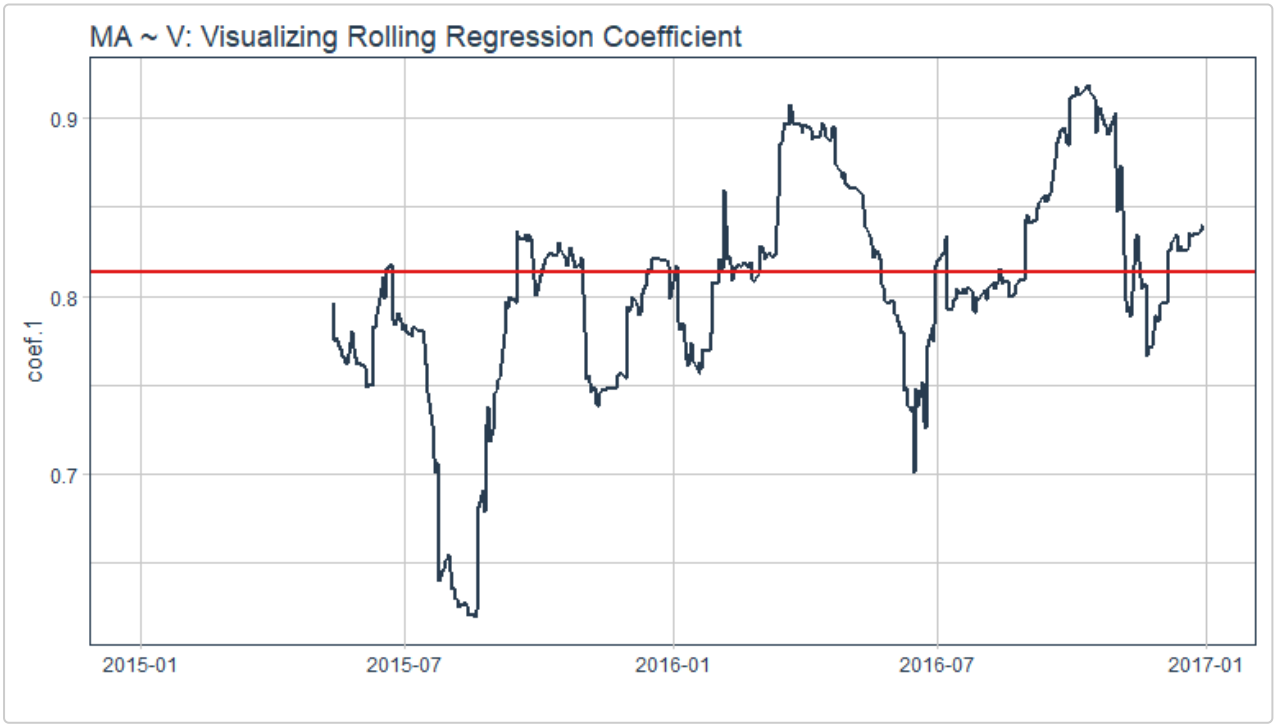
```
stock_pairs <- stock_pairs %>%
        tq_mutate(mutate_fun = rollapply,
                  width      = 90,
                  FUN        = regr_fun,
                  by.column  = FALSE,
```

```
                    col_rename = c("coef.0", "coef.1"))
stock_pairs
```

```
## # A tibble: 504 x 5
##    date             MA        V coef.0 coef.1
##    <date>        <dbl>    <dbl>  <dbl>  <dbl>
##  1 2015-01-02  0        0           NA     NA
##  2 2015-01-05 -0.0285  -0.0223      NA     NA
##  3 2015-01-06 -0.00216 -0.00646     NA     NA
##  4 2015-01-07  0.0154   0.0133      NA     NA
##  5 2015-01-08  0.0154   0.0133      NA     NA
##  6 2015-01-09 -0.0128  -0.0149      NA     NA
##  7 2015-01-12 -0.0129  -0.00196     NA     NA
##  8 2015-01-13  0.00228  0.00292     NA     NA
##  9 2015-01-14 -0.00108 -0.0202      NA     NA
## 10 2015-01-15 -0.0146  -0.00955     NA     NA
## # ... with 494 more rows
```

Finalmente, podemos visualizar el primer coeficiente así. Se agrega una línea horizontal utilizando el modelo de conjunto de datos completo. Esto nos da una idea de los momentos en los que la relación se desvía significativamente de la tendencia a largo plazo que se pueden explorar en busca de posibles oportunidades comerciales de pares.

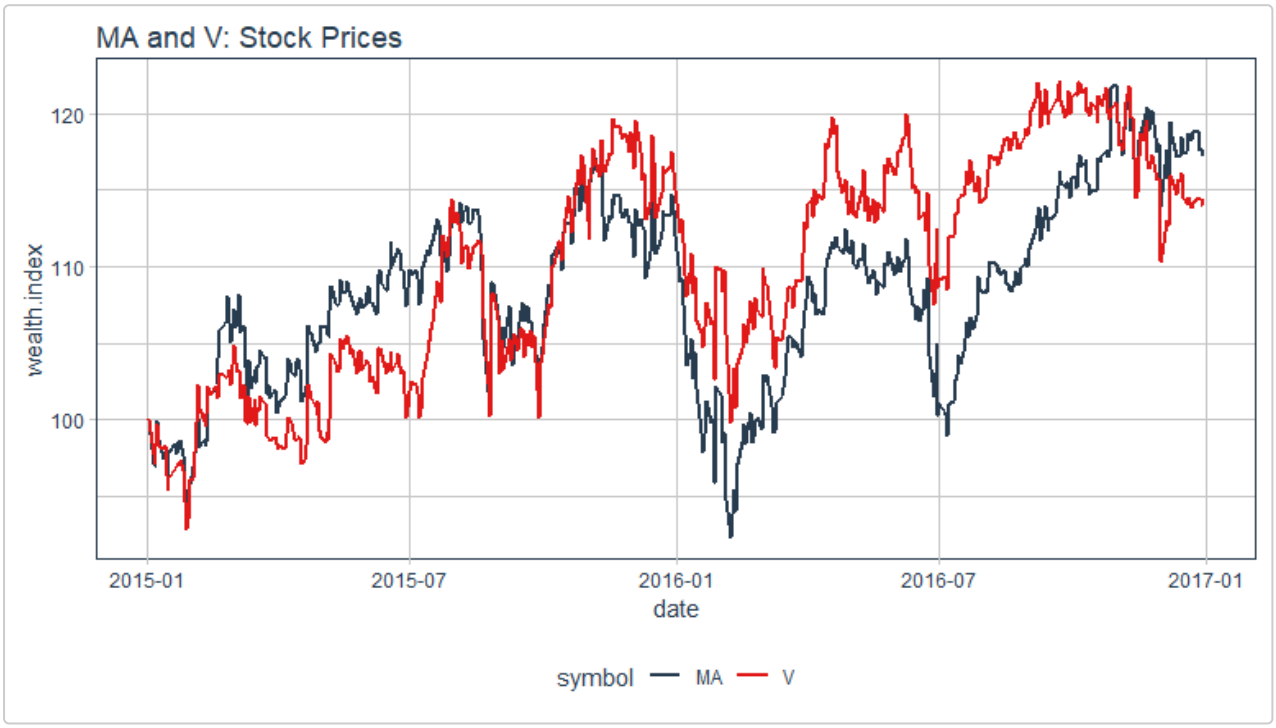
```
stock_pairs %>%
    ggplot(aes(x = date, y = coef.1)) +
    geom_line(size = 1, color = palette_light()[[1]]) +
    geom_hline(yintercept = 0.8134, size = 1, color = palette_light()[[2]]) +
    labs(title = "MA ~ V: Visualizing Rolling Regression Coefficient", x = "") +
    theme_tq()
```



Rentabilidad de las acciones durante este período de tiempo.

```
stock_prices %>%
    tq_transmute(adjusted,
```

```
                periodReturn,
                period = "daily",
                type = "log",
                col_rename = "returns") %>%
    mutate(wealth.index = 100 * cumprod(1 + returns)) %>%
    ggplot(aes(x = date, y = wealth.index, color = symbol)) +
    geom_line(size = 1) +
    labs(title = "MA and V: Stock Prices") +
    theme_tq() +
    scale_color_tq()
```



Ejemplo 7: utilice Return.clean y Return.excess para limpiar y calcular el exceso de rendimiento

En este ejemplo utilizamos varias de las `PerformanceAnalytics`funciones para limpiar y formatear declaraciones. El ejemplo utiliza tres aplicaciones progresivas de `tq_transmute`para aplicar varias funciones cuantitativas a los precios de las acciones agrupadas del `FANG`conjunto de datos. Primero, calculamos los rendimientos diarios usando `quantmod::periodReturn`. A continuación, utilizamos `Return.clean`para limpiar los valores atípicos de los datos devueltos. El `alpha`parámetro es el porcentaje de oultiers que se van a limpiar. Finalmente, los rendimientos excedentes se calculan utilizando una tasa libre de riesgo del 3% (dividida por 252 para 252 días hábiles en un año).

```
FANG %>%
    group_by(symbol) %>%
    tq_transmute(adjusted, periodReturn, period = "daily") %>%
    tq_transmute(daily.returns, Return.clean, alpha = 0.05) %>%
    tq_transmute(daily.returns, Return.excess, Rf = 0.03 / 252)
```

```
## # A tibble: 4,032 x 3
## # Groups:   symbol [4]
##    symbol date       `daily.returns > Rf`
##    <chr>  <date>                    <dbl>
```

```
##  1 FB     2013-01-02            -0.000119
##  2 FB     2013-01-03            -0.00833
##  3 FB     2013-01-04             0.0355
##  4 FB     2013-01-07             0.0228
##  5 FB     2013-01-08            -0.0124
##  6 FB     2013-01-09             0.0525
##  7 FB     2013-01-10             0.0231
##  8 FB     2013-01-11             0.0133
##  9 FB     2013-01-14            -0.0244
## 10 FB     2013-01-15            -0.0276
## # ... with 4,022 more rows
```