The future of time series and financial analysis in the tidyverse

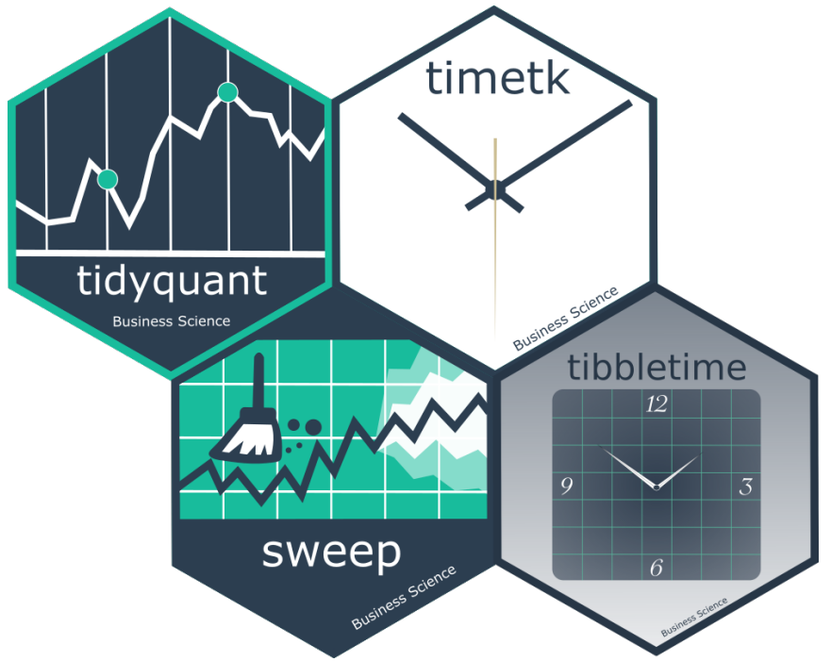
Davis Vaughan
@dvaughan32

Manager, Software
Business Science

Business Science
Applying Data Science to Business and Financial Analysis

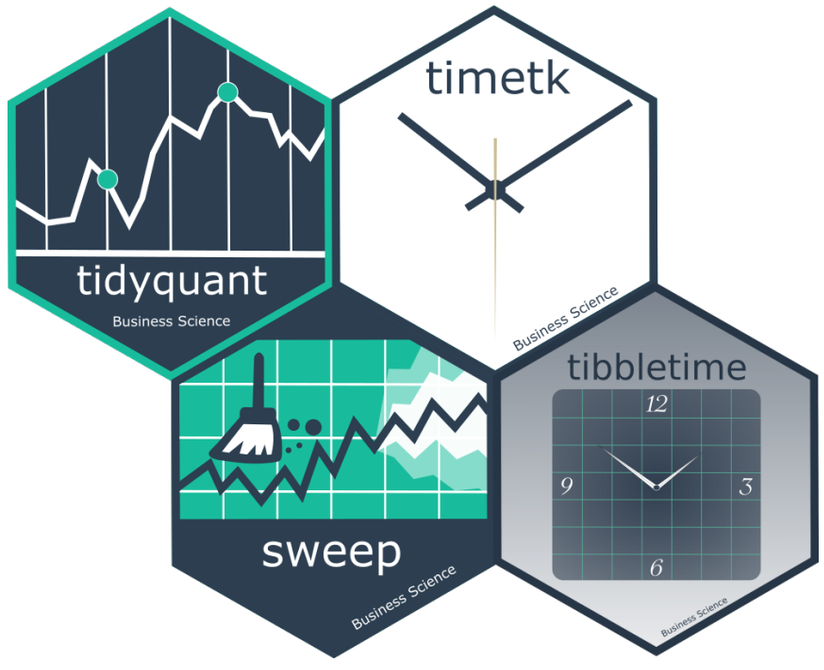
Who am I?

- Davis Vaughan
  - Manager, Software @ Business Science
  - Quantitative finance
  - **Obsessed with making your life easier**



Who am I?

- Davis Vaughan
  - Manager, Software @ Business Science
  - Quantitative finance
  - **Obsessed with making your life easier**
- Business Science
  - Consulting firm
  - Open source software
  - **Business Science University - March 2018!**



https://www.redbubble.com/people/geeknirvana/works/13977077-todays-agenda-fun?p=tote-bag

The current state of the world

https://www.redbubble.com/people/geeknirvana/works/13977077-todays-agenda-fun?p=tote-bag

https://www.redbubble.com/people/geeknirvana/works/13977077-todays-agenda-fun?p=tote-bag

The current state of the world

"It's tibbletime"

+

Extensions

https://www.redbubble.com/people/geeknirvana/works/13977077-todays-agenda-fun?p=tote-bag

The current state of the world

"It's tibbletime"
+
Extensions

The Business Analyst
Workflow

The current state of the world

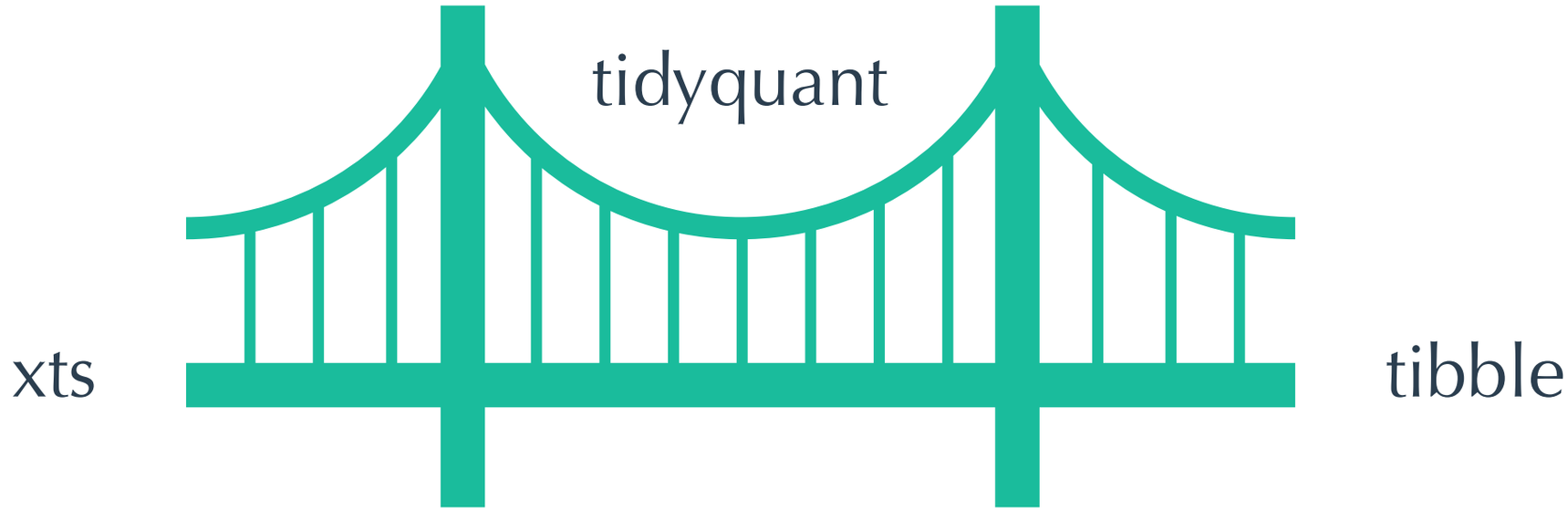
xts	tibble
Native time-index support	Powerful generalized data manipulation
Specialized (& fast) time-based manipulation	Grouped analysis
Homogeneous data (built on matrices)	Readability > Performance
Packages for financial analysis (quantmod, PerformanceAnalytics, …)	Heterogeneous data + list-column support

Wouldn’t it be nice to have a **tibble**

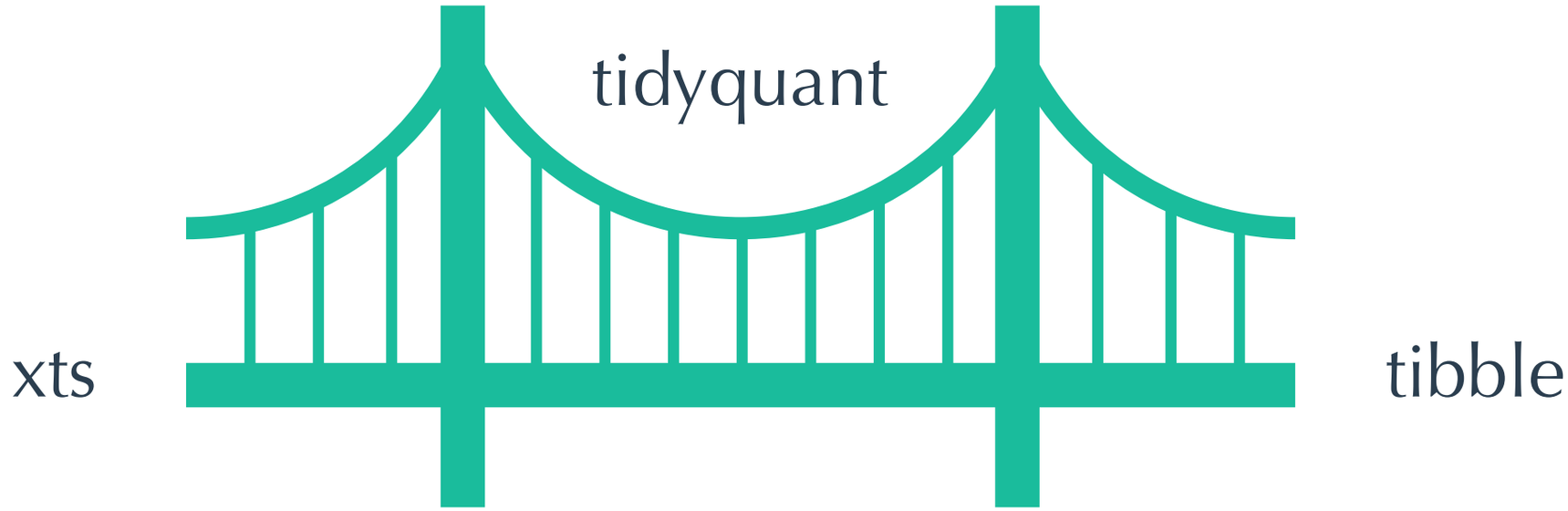
with **time-index support**,

fully leveraging the tools of the tidyverse?

A first attempt - tidyquant



A first attempt - tidyquant

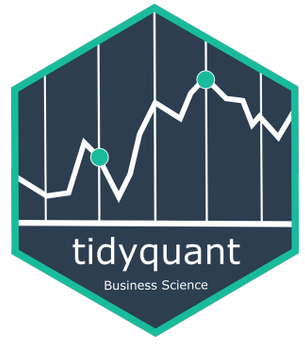


Convert to xts

Apply time-based function

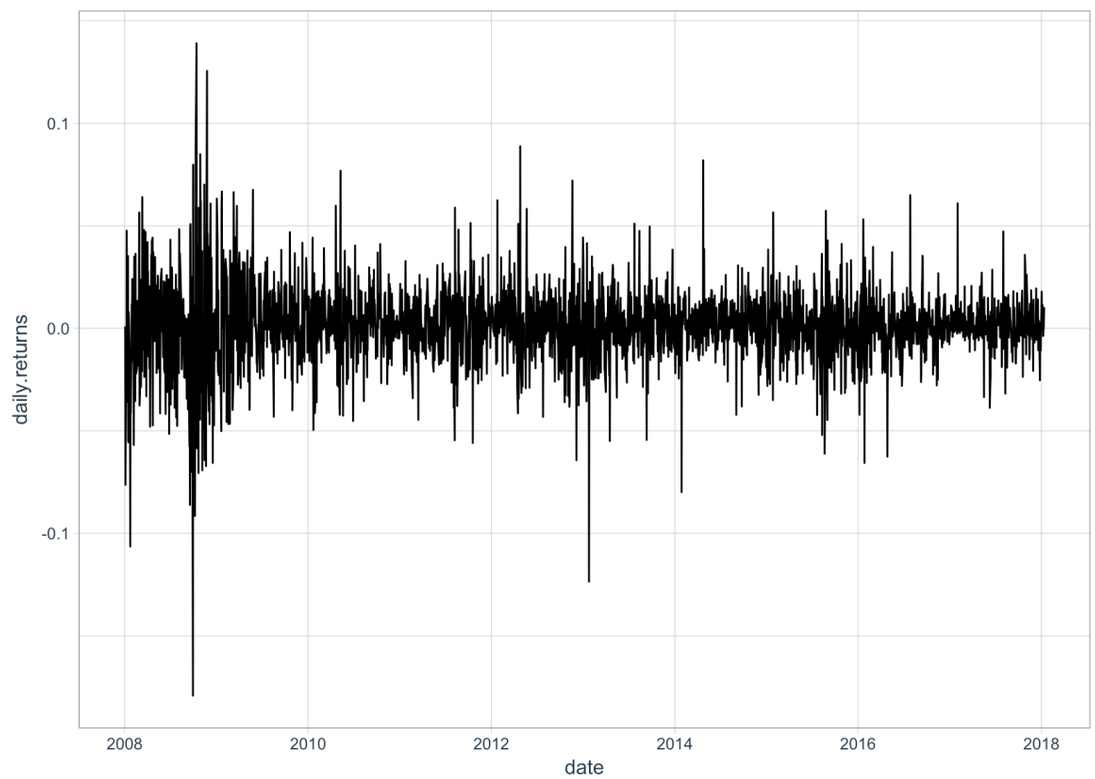
Convert to tibble

A first attempt - tidyquant




```
tq_get("AAPL") %>%
  tq_mutate(select = adjusted, mutate_fun = dailyReturn) %>%
  ggplot(aes(x = date, y = daily.returns)) +
  geom_line() +
  theme_tq()
```

- Quickly pull financial data as a tibble
- Apply any xts, quantmod, TTR, and PerformanceAnalytics function
- Pipe the result straight into other tidyverse packages



Mara Averick
@dataandme

Following

Replying to @mdancho84

What an overly gratifying activity. Like, pulling that stock data and plotting it was a minute-long affair (with thinking and getting distracted)! 💪 package *and* tutorial!

4:18 PM - 4 Jan 2018

Lots of functionality for free


```
> tq_mutate_fun_options()
```

$zoo

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

$xts

```
 [1] "apply.daily"      "apply.monthly"  "apply.quarterly"  "apply.weekly"   "apply.yearly"   "diff.xts"        "lag.xts"
 [8] "period.apply"     "period.max"     "period.min"       "period.prod"    "period.sum"     "periodicity"     "to_period"
[15] "to.daily"         "to.hourly"      "to.minutes"       "to.minutes10"   "to.minutes15"   "to.minutes3"     "to.minutes30"
[22] "to.minutes5"      "to.monthly"     "to.period"        "to.quarterly"   "to.weekly"      "to.yearly"
```

$quantmod

```
 [1] "allReturns"       "annualReturn"   "ClCl"             "dailyReturn"  "Delt"           "HiCl"          "Lag"
 [8] "LoCl"             "LoHi"           "monthlyReturn"    "Next"         "OpCl"           "OpHi"          "OpLo"
[15] "OpOp"             "periodReturn"   "quarterlyReturn"  "seriesAccel"  "seriesDecel"    "seriesDecr"    "seriesHi"
[22] "seriesIncr"       "seriesLo"       "weeklyReturn"     "yearlyReturn"
```

$TTR

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

$PerformanceAnalytics

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

What are we missing?

Conversion is **slow**

Limited in functionality

Indirectly using both the tidyverse and xts

No support for a **time-based index**

Wouldn’t it be nice to have a **tibble**

with **time-index support**,

fully leveraging the tools of the tidyverse?

MADEUPMONKEYSHIT

https://giphy.com/gifs/agentm-agent-m-Xdcj75alQutVe/links

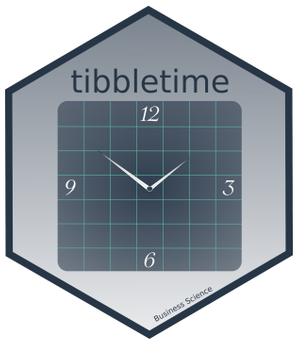
It's tibbletime.

MADEUPMONKEYSHIT

https://giphy.com/gifs/agentm-agent-m-Xdcj75alQutVe/links

It's tibbletime.

What?

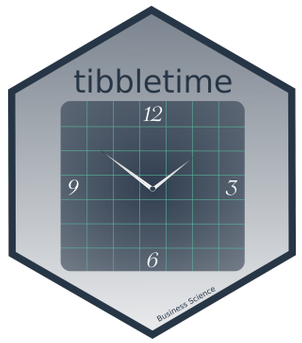


tibbletime is an **extension** of the tidyverse

that allows for the creation of time-aware **tibbles**

through the setting of a **time-index** column.

What?



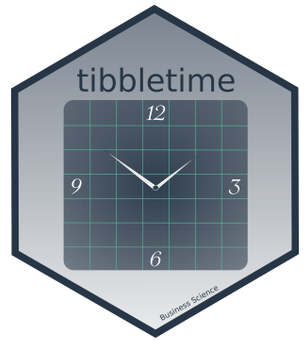
Leveraging existing tools

tibbletime is an **extension** of the tidyverse

that allows for the creation of time-aware **tibbles**

through the setting of a **time-index** column.

What?

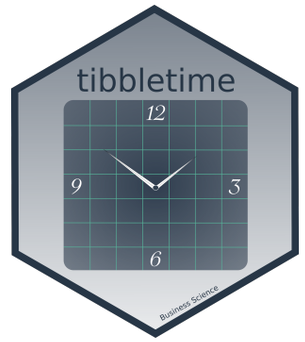


Leveraging existing tools

tibbletime is an **extension** of the tidyverse that allows for the creation of time-aware **tibbles** through the setting of a **time-index** column.

Underlying data type is the same

What?



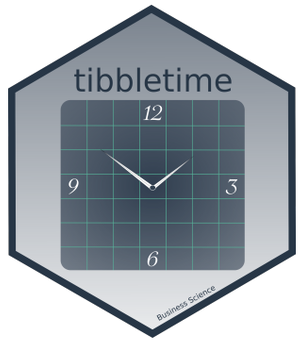
Leveraging existing tools

tibbletime is an **extension** of the tidyverse that allows for the creation of time-aware **tibbles** through the setting of a **time-index** column.

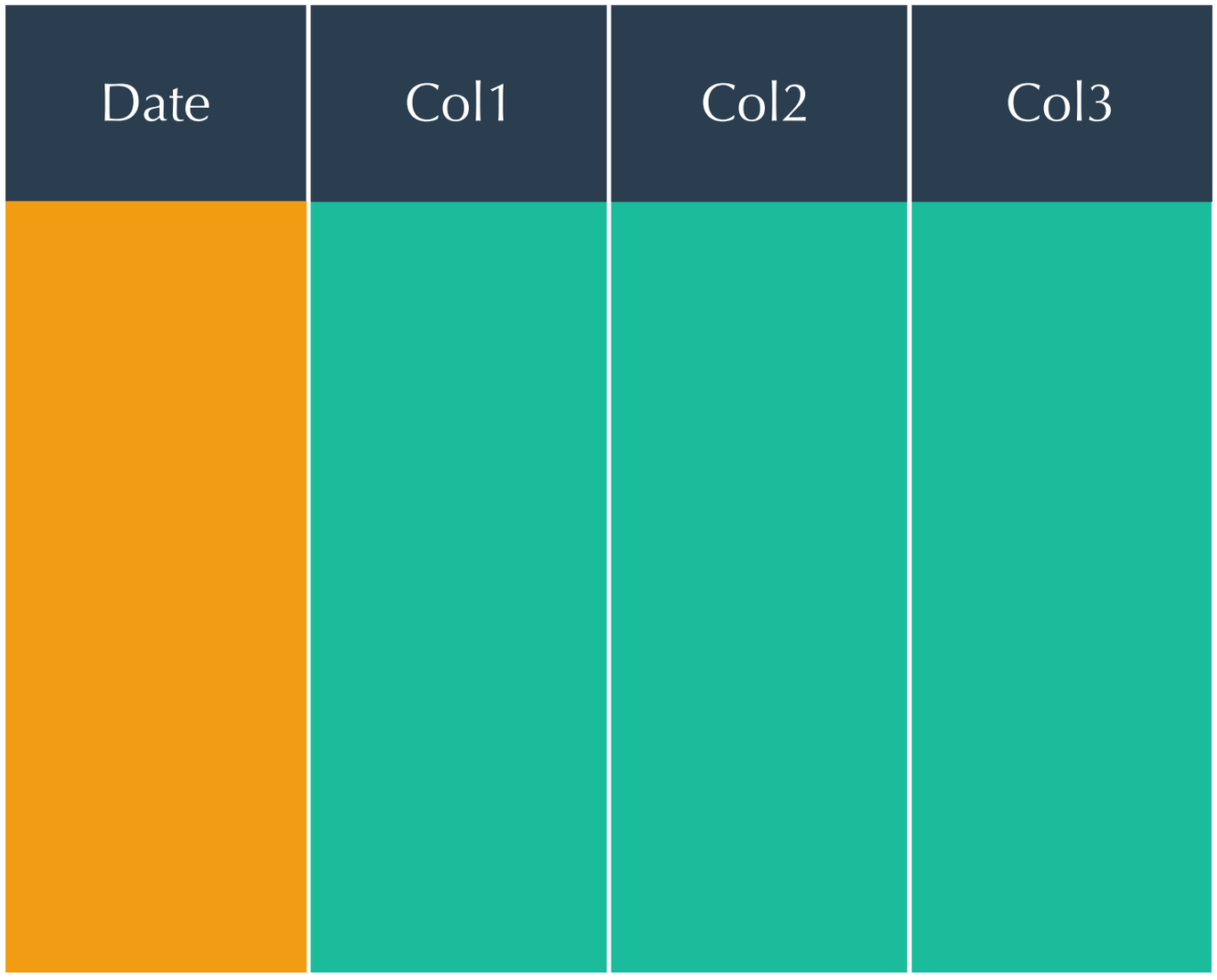
Utilizes extra knowledge

Underlying data type is the same

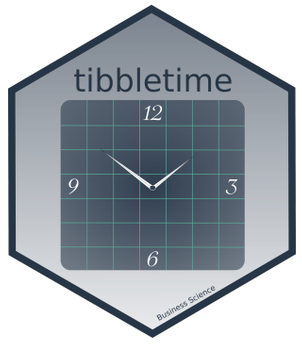
Creation




```
tbl_time(df, index = Date)
```



Why?



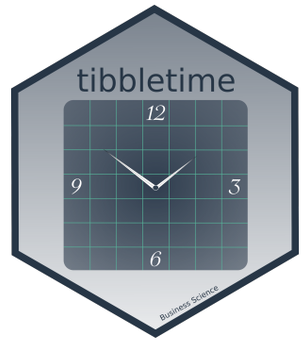
1. Perform time-based manipulations on tibbles
2. Work more naturally with time series in the tidyverse

Perform time-based manipulations on tibbles

San Diego Airbnb bookings

```
> airbnb
# A time tibble: 9,111 x 4
# Index: last_modified
   last_modified       price  latitude  longitude
   <dttm>              <dbl>     <dbl>      <dbl>
 1 2017-07-11 15:05:35  30.0      32.7      -117.1
 2 2017-07-11 15:05:36  25.0      32.8      -117.1
 3 2017-07-11 15:05:36  32.0      32.7      -117.0
 4 2017-07-11 15:05:36  32.0      32.8      -117.1
 5 2017-07-11 15:05:36  35.0      32.7      -117.1
 6 2017-07-11 15:05:36  25.0      32.7      -117.1
 7 2017-07-11 15:05:36  34.0      32.9      -117.1
 8 2017-07-11 15:05:36  33.0      32.7      -117.2
 9 2017-07-11 15:05:36  35.0      32.8      -117.2
10 2017-07-11 15:05:36  29.0      32.8      -117.2

# ... with 9,101 more rows
```

Slicing up your time series

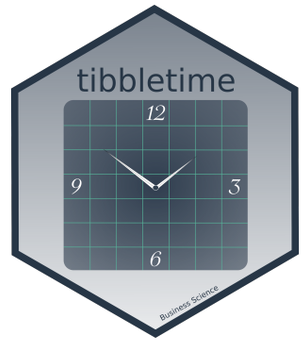


The 2nd hour of 2017-07-12

```
airbnb %>%
  filter(
    last_modified ⩾ as.POSIXct("2017-07-12 02:00:00", tz = "UTC"),
    last_modified ⩽ as.POSIXct("2017-07-12 02:59:59", tz = “UTC”)
)
```

```
# A time tibble: 67 x 4
# Index: last_modified
   last_modified       price latitude   longitude
   <dttm>              <dbl>    <dbl>       <dbl>
 1 2017-07-12 02:06:01   500    32.71      -117.2
 2 2017-07-12 02:13:36   667    32.79      -117.2
 3 2017-07-12 02:14:37   575    32.95      -117.2
 4 2017-07-12 02:15:02   678    32.95      -117.3
 5 2017-07-12 02:16:05   575    32.80      -117.3
 6 2017-07-12 02:18:44   800    32.79      -117.2
 7 2017-07-12 02:18:47   724    32.83      -117.3
 8 2017-07-12 02:18:47   825    32.73      -117.2
 9 2017-07-12 02:18:47   900    32.84      -117.2
10 2017-07-12 02:18:47   989    32.78      -117.2
# ... with 57 more rows
```

Slicing up your time series

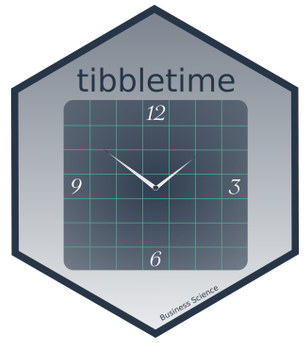


The 2nd hour of 2017-07-12

```
airbnb %>%
  filter(
    last_modified ⩾ as.POSIXct("2017-07-12 02:00:00", tz = "UTC"),
    last_modified ⩽ as.POSIXct("2017-07-12 02:59:59", tz = “UTC”)
)
```

```
airbnb %>%
  filter_time("2017-07-12 02:00:00" ~ "2017-07-12 02:59:59")
```

Slicing up your time series



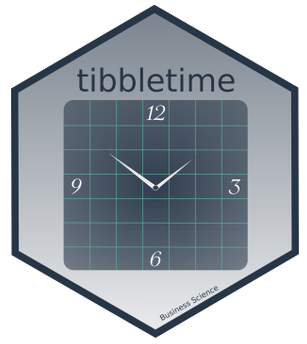
The 2nd hour of 2017-07-12

```
airbnb %>%
  filter(
    last_modified ⩾ as.POSIXct("2017-07-12 02:00:00", tz = "UTC"),
    last_modified ⩽ as.POSIXct("2017-07-12 02:59:59", tz = “UTC”)
)
```

```
airbnb %>%
  filter_time("2017-07-12 02:00:00" ~ "2017-07-12 02:59:59")
```

```
airbnb %>%
  filter_time(~"2017-07-12 02")
```

Slicing up your time series



The 2nd hour of 2017-07-12

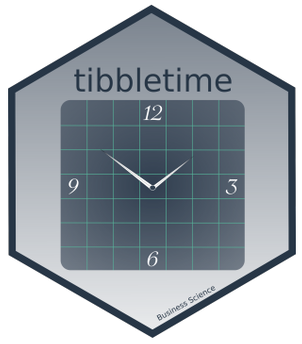
```
airbnb %>%
  filter(
    last_modified ⩾ as.POSIXct("2017-07-12 02:00:00", tz = "UTC"),
    last_modified ⩽ as.POSIXct("2017-07-12 02:59:59", tz = "UTC")
)
```

```
airbnb %>%
  filter_time("2017-07-12 02:00:00" ~ "2017-07-12 02:59:59")
```

```
airbnb %>%
  filter_time(~"2017-07-12 02")
```

```
airbnb[~"2017-07-12 02"]
```

The time formula




```
airbnb %>%
  filter_time(~"2017")
```

```
airbnb %>%
  filter_time("2017-11" ~ "2017-12")
```

```
airbnb %>%
  filter_time("start" ~ "2017-12")
```

Work more naturally with time series in the tidyverse

A new way to group

```
collapse_by(airbnb, period = "1 day")
```

```
# A time tibble: 9,111 x 4
# Index: last_modified
   last_modified          price latitude longitude
   <dttm>                 <dbl>    <dbl>     <dbl>
 1 2017-07-11 22:58:12  30.000   32.746    -117.1
 2 2017-07-11 22:58:12  25.000   32.759    -117.1
 3 2017-07-11 22:58:12  32.000   32.689    -117.0
 4 2017-07-11 22:58:12  32.000   32.754    -117.1
...
```

A new way to group

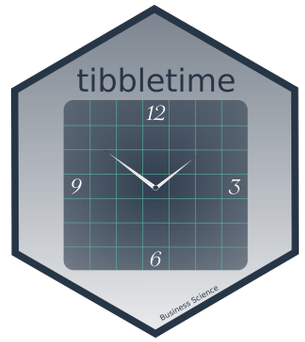
```
collapse_by(airbnb, period = "1 day")
```

```
# A time tibble: 9,111 x 4
# Index: last_modified
   last_modified          price latitude longitude
   <dttm>                 <dbl>    <dbl>     <dbl>
 1 2017-07-11 22:58:12  30.000   32.746    -117.1
 2 2017-07-11 22:58:12  25.000   32.759    -117.1
 3 2017-07-11 22:58:12  32.000   32.689    -117.0
 4 2017-07-11 22:58:12  32.000   32.754    -117.1
 ...
```

```
collapse_by(airbnb, period = "1 day") %>% tail()
```

```
# A time tibble: 6 x 4
# Index: last_modified
  last_modified          price latitude longitude
  <dttm>                 <dbl>    <dbl>     <dbl>
1 2017-07-12 05:20:42  73.000   32.794    -117.3
2 2017-07-12 05:20:42  68.000   32.724    -117.2
3 2017-07-12 05:20:42  90.000   32.797    -117.3
4 2017-07-12 05:20:42  90.000   32.798    -117.3
 ...
```

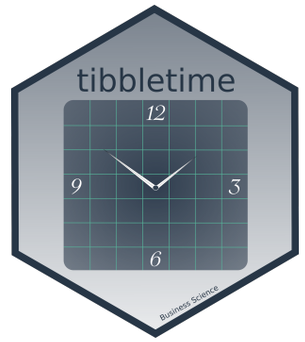
A new way to group




```
airbnb %>%
  collapse_by(period = "2 hour") %>%
  group_by(last_modified) %>%
  summarise(median_price = median(price))
```

```
# A time tibble: 8 x 2
# Index: last_modified

     last_modified                    median_price
     <dttm>                           <dbl>
1    2017-07-11 15:59:42 [14-16)      55.0
2    2017-07-11 17:59:54 [16-18)      100
3    2017-07-11 19:59:57 [18-20)      199
4    2017-07-11 21:48:16 [20-22)      450
5    2017-07-11 22:58:12 [22-00)      152
6    2017-07-12 00:59:43 [00-02)      285
7    2017-07-12 03:59:26 [02-04)      882
8    2017-07-12 05:20:42 [04-06)      40.0
```

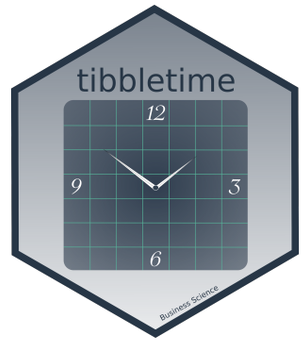
A new way to group




```
airbnb %>%
  collapse_by(period = "2 hour", clean = TRUE) %>%
  group_by(last_modified) %>%
  summarise(median_price = median(price))
```

```
# A time tibble: 8 x 2
# Index: last_modified

     last_modified                    median_price
     <dttm>                           <dbl>
1    2017-07-11 16:00:00 [14-16)      55.0
2    2017-07-11 18:00:00 [16-18)      100
3    2017-07-11 20:00:00 [18-20)      199
4    2017-07-11 22:00:00 [20-22)      450
5    2017-07-12 00:00:00 [22-00)      152
6    2017-07-12 02:00:00 [00-02)      285
7    2017-07-12 04:00:00 [02-04)      882
8    2017-07-12 06:00:00 [04-06)      40.0
```

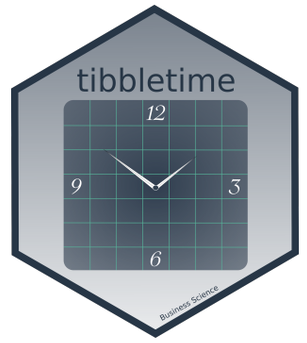
A new way to group



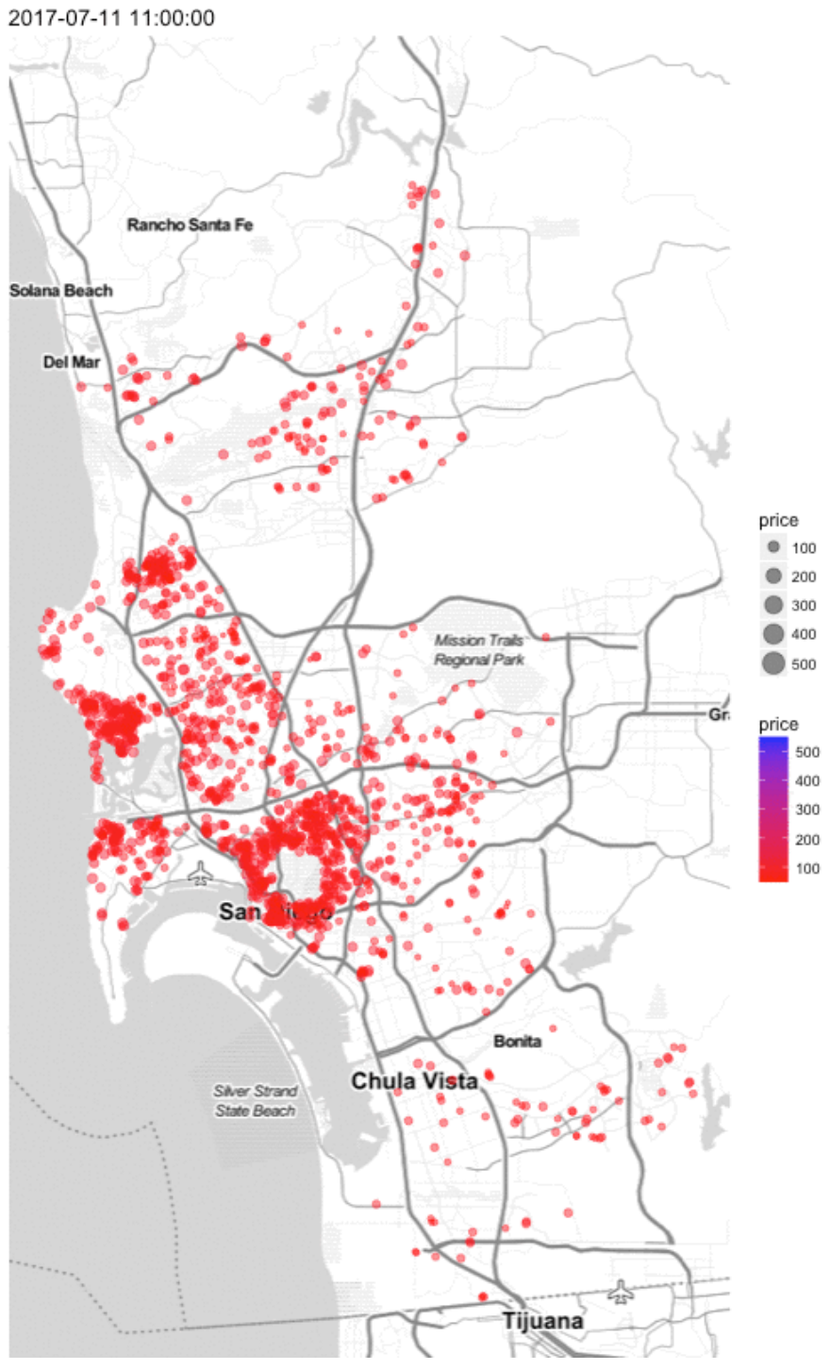

```
airbnb %>%
  collapse_by(period = "2 hour", clean = TRUE, side = "start") %>%
  group_by(last_modified) %>%
  summarise(median_price = median(price))
```

```
# A time tibble: 8 x 2
# Index: last_modified

     last_modified                median_price
     <dttm>                             <dbl>
1    2017-07-11 14:00:00 [14-16)         55.0
2    2017-07-11 16:00:00 [16-18)         100
3    2017-07-11 18:00:00 [18-20)         199
4    2017-07-11 20:00:00 [20-22)         450
5    2017-07-11 22:00:00 [22-00)         152
6    2017-07-12 00:00:00 [00-02)         285
7    2017-07-12 02:00:00 [02-04)         882
8    2017-07-12 04:00:00 [04-06)         40.0
```

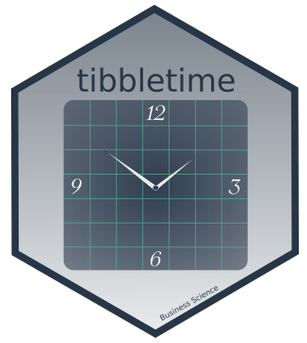
General time-based grouping



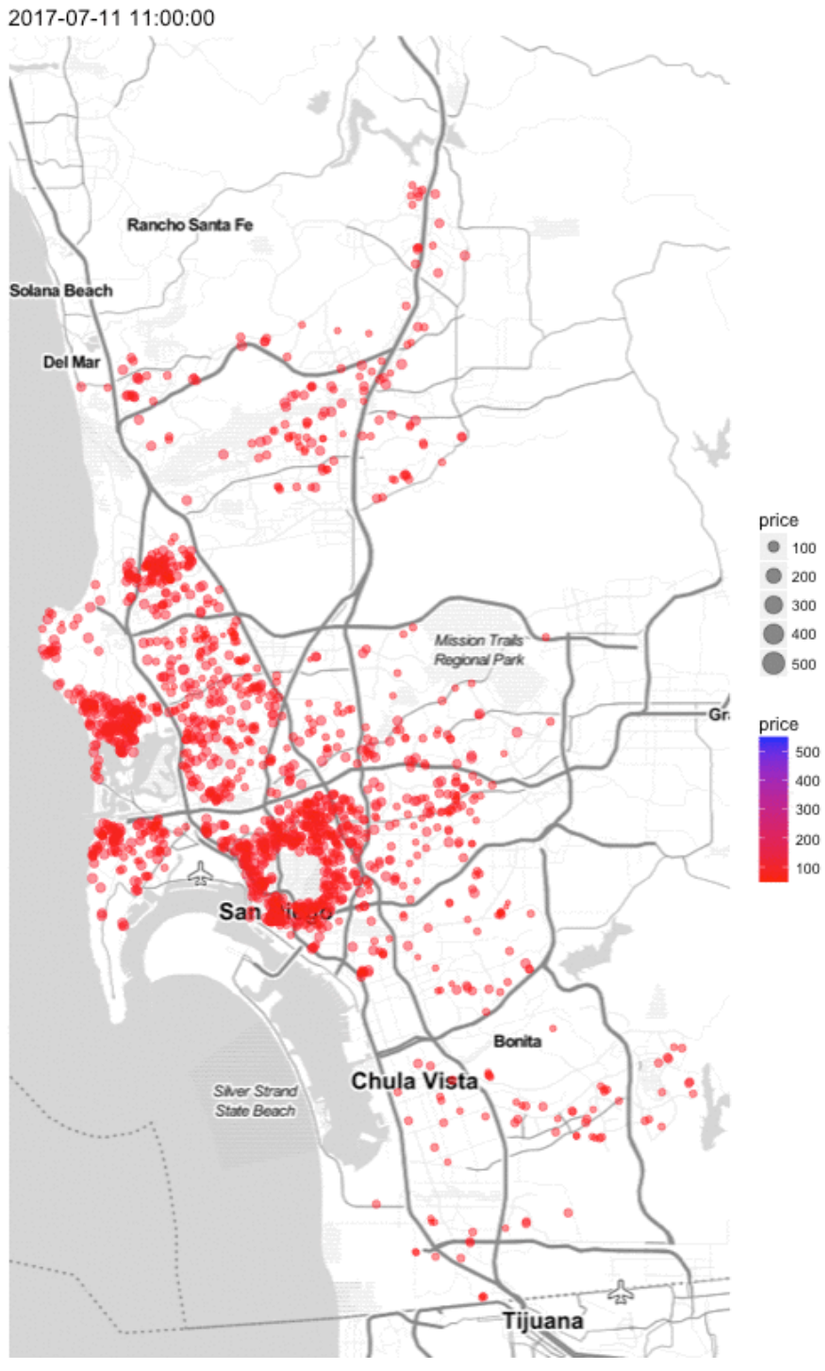

```
library(ggmap)
library(gganimate)

airbnb_plot ← airbnb %>%

  # Collapse and clean
  collapse_by(period = "hour", clean = TRUE, side = "start") %>%

  # Throw out a few outliers
  filter(
    between(price, quantile(price, .05), quantile(price, .95))
  ) %>%

  # Map and animate
  qmplot(longitude, latitude, data = ., geom = "blank") +
  geom_point(
    aes(color = price, size = price, frame = last_modified),
    alpha = .5) +
  scale_color_continuous(low = "red", high = "blue")

gganimate(airbnb_plot)
```



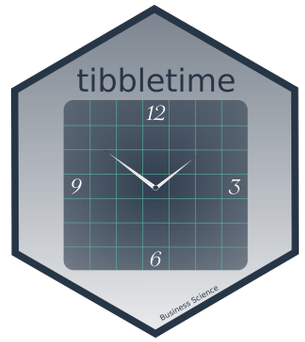
General time-based grouping




```
library(ggmap)
library(gganimate)

airbnb_plot ← airbnb %>%

  # Collapse and clean
  collapse_by(period = "hour", clean = TRUE, side = "start") %>%

  # Throw out a few outliers
  filter(
    between(price, quantile(price, .05), quantile(price, .95))
  ) %>%

  # Map and animate
  qmplot(longitude, latitude, data = ., geom = "blank") +
  geom_point(
    aes(color = price, size = price, frame = last_modified),
    alpha = .5) +
  scale_color_continuous(low = "red", high = "blue")

gganimate(airbnb_plot)
```



Extra functionality

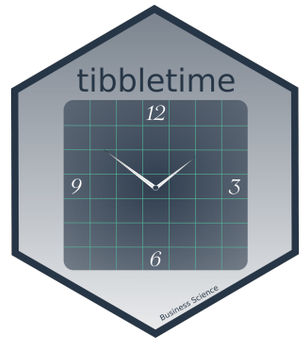


Control the start date for the grouping

```
airbnb %>%
  collapse_by(period = "2 hour", clean = TRUE, side = "start") %>%
  group_by(last_modified) %>%
  summarise(median_price = median(price))
```

```
# A time tibble: 8 x 2
# Index: last_modified

     last_modified                median_price
     <dttm>                       <dbl>
1    2017-07-11 14:00:00 [14-16)  55.0
2    2017-07-11 16:00:00 [16-18)  100
3    2017-07-11 18:00:00 [18-20)  199
 ...
```

Extra functionality

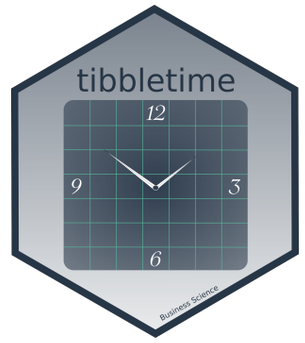


Control the start date for the grouping

```
airbnb %>%
  collapse_by(period = "2 hour", clean = TRUE, side = "start",
              start_date = "2017-07-11 15:00:00”) %>%
  group_by(last_modified) %>%
  summarise(median_price = median(price))
```

```
# A time tibble: 8 x 2
# Index: last_modified

     last_modified                median_price
     <dttm>                             <dbl>
1    2017-07-11 15:00:00 [15-17)         65.0
2    2017-07-11 17:00:00 [17-19)         123
3    2017-07-11 19:00:00 [19-21)         275
 ...
```

Extra functionality

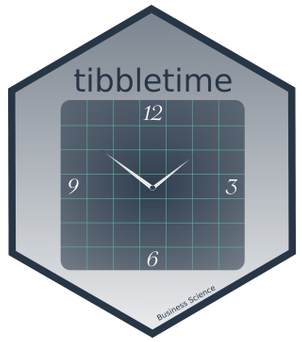


Flexible periods

```
boundaries ← as.POSIXct(c("2017-07-11 00:00:00", "2017-07-11 20:00:00”),
                        tz = “UTC")

airbnb %>%
  collapse_by(period = boundaries, clean = TRUE, side = “start”) %>%
  group_by(last_modified) %>%
  summarise(median_price = median(price))
```

```
# A time tibble: 2 x 2
# Index: last_modified

     last_modified                median_price
     <dttm>                       <dbl>
1    2017-07-11 00:00:00          115
2    2017-07-11 20:00:00          420
```

Extra functionality



Multi-class support

Date

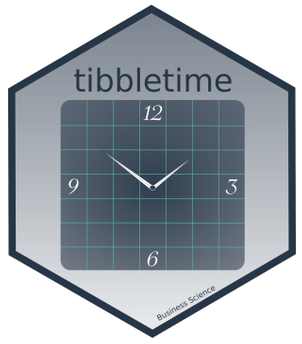
Posixct

yearmon

yearqtr

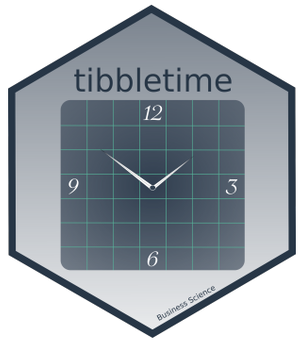
hms

Let's get things rolling



`rollify()`

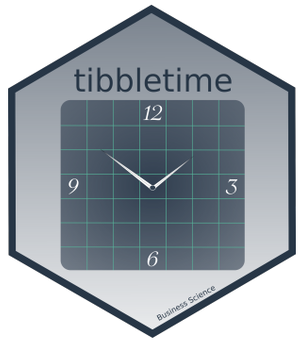
Let's get things rolling



`rollify()`

Adverb - A word used to modify a verb.

Let's get things rolling

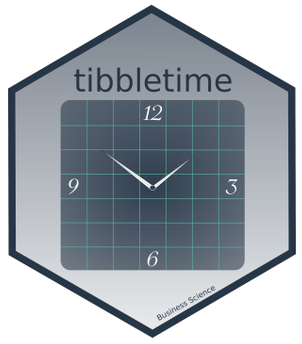


`rollify()`

Adverb - A word used to modify a verb.

function

Let's get things rolling



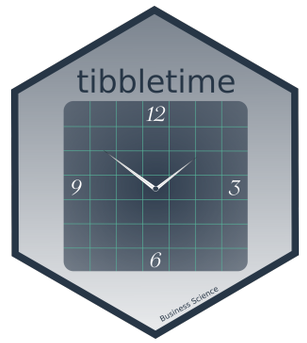
`rollify()`

Adverb - A word used to modify a verb.

function

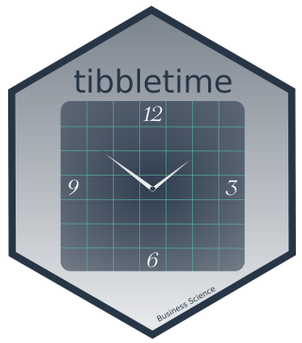
rollify() takes a function and turns it
into a rolling version of itself

Let's get things rolling




```
> FB
# A tibble: 1,008 x 3
   date       adjusted    volume
   <date>        <dbl>     <dbl>
 1 2013-01-02     28.0  69846400
 2 2013-01-03     27.8  63140600
 3 2013-01-04     28.8  72715400
 4 2013-01-07     29.4  83781800
 5 2013-01-08     29.1  45871300
 6 2013-01-09     30.6 104787700
 7 2013-01-10     31.3  95316400
 8 2013-01-11     31.7  89598000
 9 2013-01-14     31.0  98892800
10 2013-01-15     30.1 173242600
# ... with 998 more rows
```

Rolling averages




```
short_term_mean ← rollify(mean, window = 5)
long_term_mean  ← rollify(mean, window = 50)
```

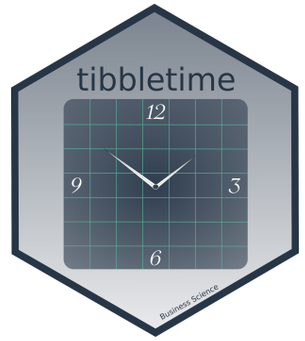
} mean() becomes a rolling mean

```
mutate(FB,
  short_mean = short_term_mean(adjusted),
  long_mean  = long_term_mean(adjusted)
)
```

```
# A tibble: 1,008 x 4
   date          adjusted    short_mean   long_mean
   <date>        <dbl>       <dbl>        <dbl>
 1 2013-01-02     28.0        NA           NA
 2 2013-01-03     27.8        NA           NA
 3 2013-01-04     28.8        NA           NA
 4 2013-01-07     29.4        NA           NA
 5 2013-01-08     29.1        28.6         NA
 6 2013-01-09     30.6        29.1         NA
 7 2013-01-10     31.3        29.8         NA
 8 2013-01-11     31.7        30.4         NA
 9 2013-01-14     31.0        30.7         NA
10 2013-01-15     30.1        30.9         NA
```

Rolling averages

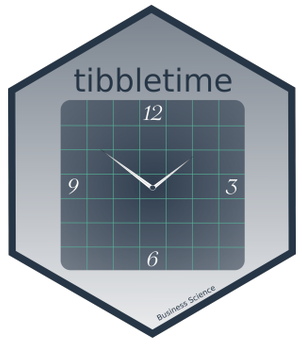
Rolling linear models




```
lm_roll ← rollify(
  .f      = ~ lm(.y ~ .x),
  window = 5, unlist = FALSE)

FB_model ← FB %>%
  mutate(
    lag_volume = lag(volume),
  model = lm_roll(lag_volume, adjusted)
)
```

```
# A tibble: 1,008 x 5
   date       adjusted    volume lag_volume  model
   <date>        <dbl>     <dbl>      <dbl>  <list>
 1 2013-01-02     28.0  69846400         NA  <lgl [1]>
 2 2013-01-03     27.8  63140600   69846400  <lgl [1]>
 3 2013-01-04     28.8  72715400   63140600  <lgl [1]>
 4 2013-01-07     29.4  83781800   72715400  <lgl [1]>
 5 2013-01-08     29.1  45871300   83781800  <S3: lm>
 6 2013-01-09     30.6 104787700   45871300  <S3: lm>
 7 2013-01-10     31.3  95316400  104787700  <S3: lm>
 8 2013-01-11     31.7  89598000   95316400  <S3: lm>
 9 2013-01-14     31.0  98892800   89598000  <S3: lm>
10 2013-01-15     30.1 173242600   98892800  <S3: lm>
# ... with 998 more rows
```

Rolling linear models




```
FB_model %>%
  filter(!is.na(model)) %>%
  mutate(glanced = map(model, broom::glance)) %>%
  select(date, glanced) %>%
  unnest()
```

```
# A tibble: 1,004 x 12
   date       r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC deviance df.residual
   <date>         <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl> <dbl> <dbl>    <dbl>       <int>
 1 2013-01-08 0.113           -0.330  0.817  0.255      0.664     2  -3.48 13.0  11.1     1.34            2
 2 2013-01-09 0.326            0.102  0.972  1.45       0.314     2  -5.67 17.3  16.2     2.83            3
 3 2013-01-10 0.0895          -0.214  1.19   0.295      0.625     2  -6.68 19.4  18.2     4.23            3
 4 2013-01-11 0.130           -0.159  1.24   0.450      0.550     2  -6.91 19.8  18.7     4.65            3
 5 2013-01-14 0.106           -0.193  1.11   0.354      0.594     2  -6.36 18.7  17.5     3.72            3
 6 2013-01-15 0.0861          -0.219  0.691  0.282      0.632     2  -3.97 13.9  12.8     1.43            3
 7 2013-01-16 0.426            0.235  0.693  2.23       0.233     2  -3.98 14.0  12.8     1.44            3
 8 2013-01-17 0.180           -0.0932 0.808  0.659      0.476     2  -4.75 15.5  14.3     1.96            3
 9 2013-01-18 0.0000962       -0.333  0.569  0.000289   0.988     2  -3.00 12.0  10.8     0.972           3
10 2013-01-22 0.0845          -0.221  0.447  0.277      0.635     2  -1.79  9.58  8.41    0.599           3
# ... with 994 more rows
```

Wouldn’t it be nice to have a **tibble**

with **time-index support**,

fully leveraging the tools of the tidyverse?

Built on top of
tibbles

Wouldn't it be nice to have a **tibble**
with **time-index support**,

fully leveraging the tools of the tidyverse?

Built on top of tibbles

Wouldn't it be nice to have a **tibble** with **time-index support**, **fully leveraging** the tools of the tidyverse?

Learns about the index at creation

Built on top of tibbles

Wouldn't it be nice to have a **tibble** with **time-index support**, **fully leveraging** the tools of the tidyverse?

Learns about the index at creation

Usable with ANY tidyverse function

The vision for tibbletime

A platform for the development of time-based extensions of the tidyverse

The vision for tibbletime

A platform for the development of time-based extensions of the tidyverse

One such extension…

tidyfinance

One such extension…

tidyfinance ?

Facebook, Amazon, Netflix, Google

```
FANG_time ← FANG %>%
  group_by(symbol) %>%
  as_tbl_time(date)

slice(FANG_time, 1:2)
```

```
# A time tibble: 8 x 3
# Index: date
# Groups: symbol [4]
  symbol  date       adjusted
   <chr>  <date>        <dbl>
1 AMZN   2013-01-02     257
2 AMZN   2013-01-03     258
3 FB     2013-01-02     28.0
4 FB     2013-01-03     27.8
5 GOOG   2013-01-02     361
6 GOOG   2013-01-03     361
7 NFLX   2013-01-02     13.1
8 NFLX   2013-01-03     13.8
```

Calculate returns

```
FANG_time %>%
  calculate_return(adjusted, period = "daily")
```

```
# A time tibble: 4,032 x 4
# Index: date
# Groups: symbol [4]
   symbol date       adjusted adjusted_return
   <chr>  <date>        <dbl>           <dbl>
 1 FB     2013-01-02     28.0          0
 2 FB     2013-01-03     27.8         -0.00821
 3 FB     2013-01-04     28.8          0.0356
 4 FB     2013-01-07     29.4          0.0229
 5 FB     2013-01-08     29.1         -0.0122
 6 FB     2013-01-09     30.6          0.0526
 7 FB     2013-01-10     31.3          0.0232
 8 FB     2013-01-11     31.7          0.0134
 9 FB     2013-01-14     31.0         -0.0243
10 FB     2013-01-15     30.1         -0.0275
# ... with 4,022 more rows
```

Calculate returns

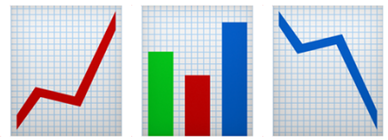
```
FANG_time %>%
  calculate_return(adjusted, period = "yearly")
```

```
# A time tibble: 20 x 4
# Index: date
# Groups: symbol [4]
   symbol date       adjusted adjusted_return
   <chr>  <date>        <dbl>           <dbl>
 1 FB     2013-01-02     28.0           0
 2 FB     2013-12-31     54.7           0.952
 3 FB     2014-12-31     78.0           0.428
 4 FB     2015-12-31     105            0.341
 5 FB     2016-12-30     115            0.0993
 6 AMZN   2013-01-02     257            0
 7 AMZN   2013-12-31     399            0.550
 8 AMZN   2014-12-31     310           -0.222
 9 AMZN   2015-12-31     676            1.18
10 AMZN   2016-12-30     750            0.109
# ... with 10 more rows
```

Calculate returns

```
FANG_return ← FANG_time %>%
  calculate_return(adjusted, period = "daily") %>%
  mutate(drawdown = drawdown(adjusted_return),
         cum_ret  = cumulative_return(adjusted_return))
```

```
# A time tibble: 4,032 x 6
# Index: date
# Groups: symbol [4]
   symbol date       adjusted adjusted_return  drawdown   cum_ret
   <chr>  <date>        <dbl>           <dbl>     <dbl>     <dbl>
 1 FB     2013-01-02     28.0          0          0         0
 2 FB     2013-01-03     27.8         -0.00821  -0.00821  -0.00821
 3 FB     2013-01-04     28.8          0.0356    0         0.0271
 4 FB     2013-01-07     29.4          0.0229    0         0.0507
 5 FB     2013-01-08     29.1         -0.0122   -0.0122    0.0379
 6 FB     2013-01-09     30.6          0.0526    0         0.0925
 7 FB     2013-01-10     31.3          0.0232    0         0.118
 8 FB     2013-01-11     31.7          0.0134    0         0.133
 9 FB     2013-01-14     31.0         -0.0243   -0.0243    0.105
10 FB     2013-01-15     30.1         -0.0275   -0.0511    0.0750
# ... with 4,022 more rows
```

tidyfinance + tibbletime = 

```
FANG_return_monthly ← FANG_return %>%
  collapse_by("month") %>%
  group_by(symbol, date) %>%
  summarise(monthly_return = total_return(adjusted_return))
```

```
# A time tibble: 192 x 3
# Index: date
# Groups: symbol [?]
   symbol date       monthly_return
   <chr>  <date>              <dbl>
 1 AMZN   2013-01-31         0.0318
 2 AMZN   2013-02-28        -0.00463
 3 AMZN   2013-03-28         0.00840
 4 AMZN   2013-04-30        -0.0476
 5 AMZN   2013-05-31         0.0606
 6 AMZN   2013-06-28         0.0315
 7 AMZN   2013-07-31         0.0847
 8 AMZN   2013-08-30        -0.0672
 9 AMZN   2013-09-30         0.113
10 AMZN   2013-10-31         0.164
# ... with 182 more rows
```

Performance summary

Cumulative returns

```
plot_cum_ret ← FANG_return %>%
  ggplot(aes(x = date, y = cum_ret, color = symbol)) +
  geom_line() +
  theme_tq() +
  theme(axis.title.x = element_blank(),
        axis.text.x  = element_blank(),
        axis.ticks.x = element_blank()) +
  labs(
   y     = "Cumulative Return",
   title = "Performance summary: Facebook,
            Amazon, Netflix, Google") +
  theme(legend.position="none") +
  scale_color_tq()
```

Monthly returns

```
plot_month_ret ← FANG_return_monthly %>%
  ggplot(aes(x = date, y = monthly_return, fill = symbol)) +
  geom_col(width = 15, position = position_dodge()) +
  theme_tq() +
  theme(axis.title.x = element_blank(),
        axis.text.x  = element_blank(),
        axis.ticks.x = element_blank()) +
  labs(y = "Monthly Return") +
  theme(legend.position="none") +
  scale_fill_tq()
```

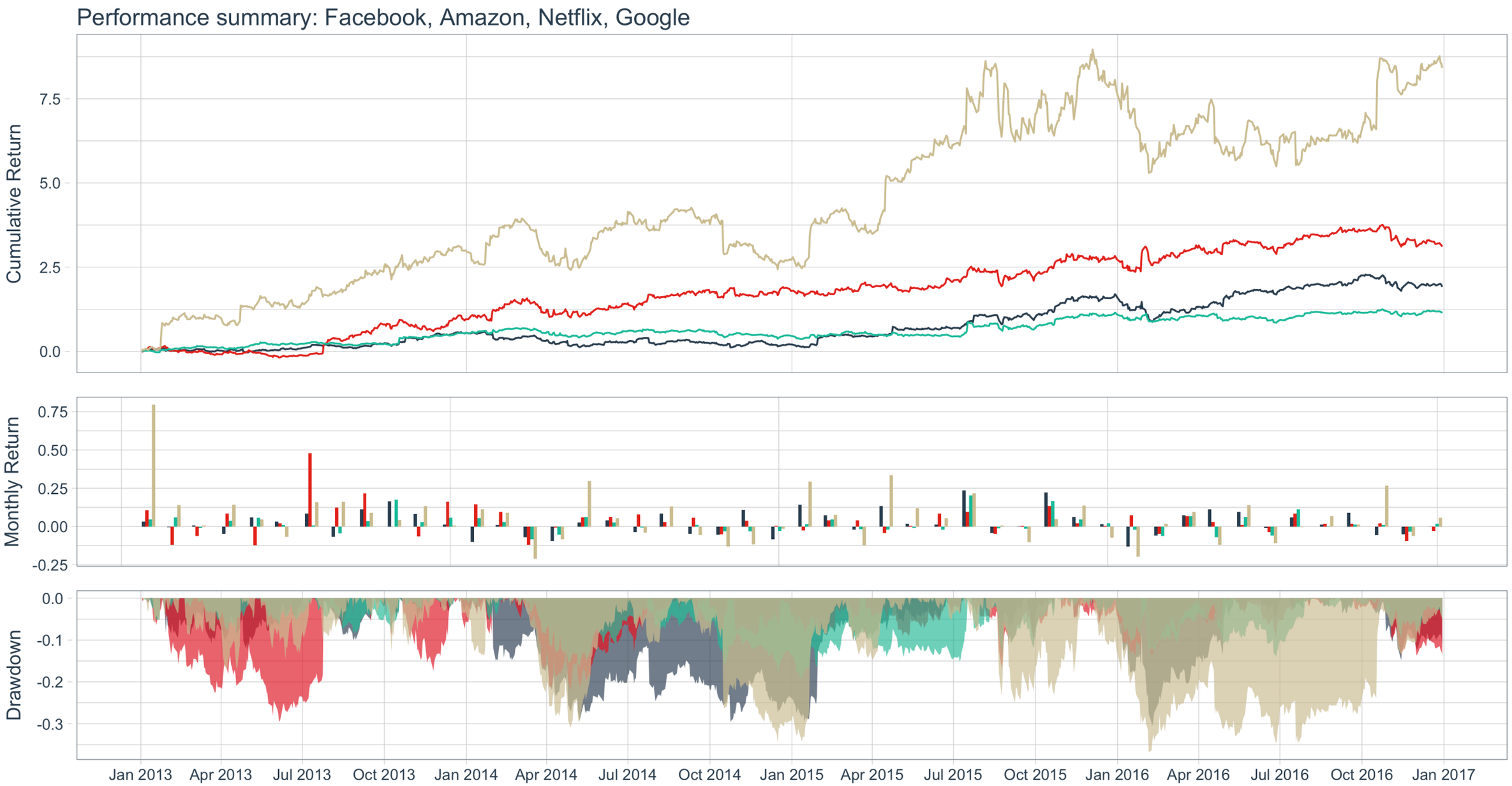
Drawdown

```
plot_drawdown ← FANG_return %>%
  ggplot(aes(x = date, y = drawdown, fill = symbol)) +
  geom_area(position = position_identity(), alpha = .7) +
  theme_tq() +
  scale_x_date(
    date_breaks = "3 months",
    date_labels = "%b %Y") +
  labs(x = "", y = "Drawdown") +
  scale_fill_tq()
```

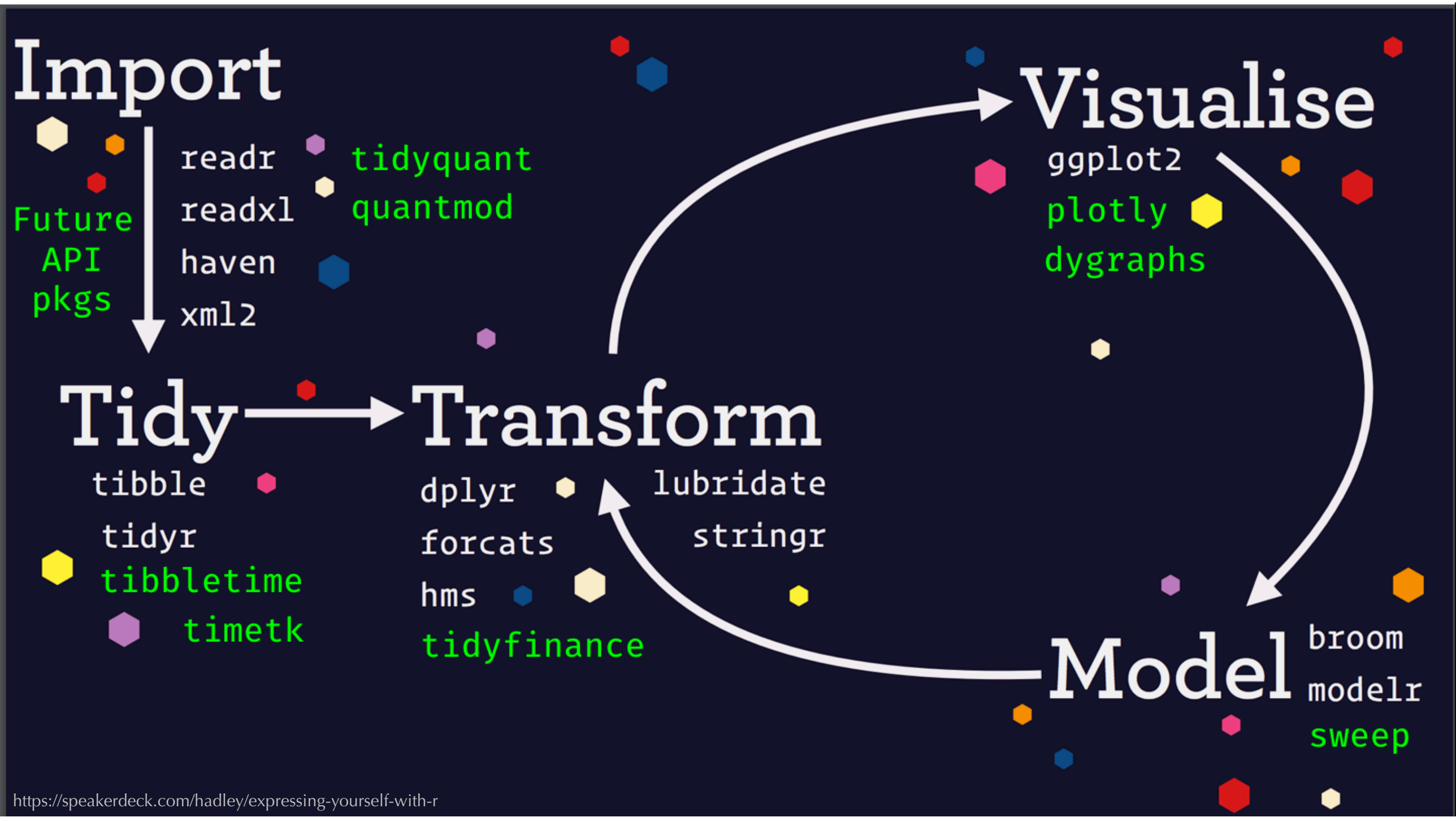
Patchwork combination

```
plot_cum_ret +
  plot_month_ret +
  plot_drawdown +
  plot_layout(ncol = 1, heights = c(2, 1, 1))
```

Performance summary



The Business Analyst Workflow


Import
readr
readxl
haven
xml2
tidyquant
quantmod
Future
API
pkgs
Visualise
ggplot2
plotly
dygraphs
Tidy
tibble
tidyr
tibbletime
timetk
Transform
dplyr
forcats
hms
tidyfinance
lubridate
stringr
Model
broom
modelr
sweep
https://speakerdeck.com/hadley/expressing-yourself-with-r

Conclusion

We now have

~~Wouldn't it be nice to have~~ a **tibble**

with **time-index support**,

fully leveraging the tools of the tidyverse!

What will you build with it?

Thank you!

Davis Vaughan

@dvaughan32

DavisVaughan

www.business-science.io