Untitled

Alexis Solis

9/25/2020

1. Introduction & Importing Data

We'll work with intraday data for the *S&P/BMV IPC Equity Index*. The data consists of `n = 2,133,890` observations and `k = 23` variables. The time-series is composed of prices and trades per minute, spanning from the beginning of 1996 through the first half of 2018.

```
# Read the data
IPC <- arrow::read_parquet(file = here("01-Data", "parquet", "raw_MEXICO_IPC.parquet"))
```

First thing we do is take a look at the columns and data types that we have:

```
## Rows: 2,133,890
## Columns: 23
## $ `#RIC`                  <chr> ".MXX", ".MXX", ".MXX", ".MXX", ".MXX", ".M...
## $ `Date[L]`               <dbl> 19960102, 19960102, 19960102, 19960102, 199...
## $ `Time[L]`               <time> 08:36:00, 08:38:00, 08:39:00, 08:40:00, 08...
## $ Type                    <chr> "Intraday 1Min", "Intraday 1Min", "Intraday...
## $ Open                    <dbl> 2777.47, 2777.47, 2777.47, 2777.47, 2777.47...
## $ High                    <dbl> 2777.47, 2777.47, 2777.47, 2777.47, 2777.47...
## $ Low                     <dbl> 2777.47, 2777.47, 2777.47, 2777.47, 2777.14...
## $ Last                    <dbl> 2777.47, 2777.47, 2777.47, 2777.47, 2777.14...
## $ Volume                  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ `Ave. Price`            <dbl> 2777.470, 2777.470, 2777.470, 2777.470, 277...
## $ VWAP                    <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `No. Trades`            <dbl> 1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3...
## $ `Correction Qualifiers` <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `Open Bid`              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `High Bid`              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `Low Bid`               <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `Close Bid`             <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `No. Bids`              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
## $ `Open Ask`              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `High Ask`              <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `Low Ask`               <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `Close Ask`             <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,...
## $ `No. Asks`              <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
```

We can count how many `NA`s are present in our data. We do this per column:

```
## Rows: 1
## Columns: 23
## $ ticker                <int> 0
## $ raw_date              <int> 0
## $ raw_time              <int> 0
## $ type                  <int> 0
## $ open                  <int> 0
## $ high                  <int> 0
## $ low                   <int> 0
## $ last                  <int> 0
## $ volume                <int> 0
## $ average_price         <int> 0
## $ vwap                  <int> 2133890
## $ no_trades             <int> 0
## $ correction_qualifiers <int> 2133890
## $ open_bid              <int> 2133890
## $ high_bid              <int> 2133890
## $ low_bid               <int> 2133890
## $ close_bid             <int> 2133890
## $ no_bids               <int> 0
## $ open_ask              <int> 2133890
## $ high_ask              <int> 2133890
## $ low_ask               <int> 2133890
## $ close_ask             <int> 2133890
## $ no_ask                <int> 0
```

We see that there are 10 columns (variables) that have all values as `NA`. We assign these variables to the `columns_to_remove` object and remove them from the data.

```
##  [1] "vwap"                 "correction_qualifiers" "open_bid"
##  [4] "high_bid"             "low_bid"               "close_bid"
##  [7] "open_ask"             "high_ask"              "low_ask"
## [10] "close_ask"
```

We name the *clean* dataset as `IPC_ip` (IPC intraday prices) and again see the column names and each data type.

```
## Rows: 2,133,890
## Columns: 13
## $ ticker        <chr> ".MXX", ".MXX", ".MXX", ".MXX", ".MXX", ".MXX", ".MXX...
## $ raw_date      <dbl> 19960102, 19960102, 19960102, 19960102, 19960102, 199...
## $ raw_time      <time> 08:36:00, 08:38:00, 08:39:00, 08:40:00, 08:41:00, 08...
## $ type          <chr> "Intraday 1Min", "Intraday 1Min", "Intraday 1Min", "I...
## $ open          <dbl> 2777.47, 2777.47, 2777.47, 2777.47, 2777.47, 2777.14,...
## $ high          <dbl> 2777.47, 2777.47, 2777.47, 2777.47, 2777.47, 2777.14,...
## $ low           <dbl> 2777.47, 2777.47, 2777.47, 2777.47, 2777.14, 2777.14,...
## $ last          <dbl> 2777.47, 2777.47, 2777.47, 2777.47, 2777.14, 2777.14,...
## $ volume        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ average_price <dbl> 2777.470, 2777.470, 2777.470, 2777.470, 2777.360, 277...
## $ no_trades     <dbl> 1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 3, 3, 3, 3,...
## $ no_bids       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ no_ask        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

2. Feature Engineering & Data Wrangling

We then carry on with the analysis by creating some new variables (a.k.a. *Feature Engineering*) and manipulating the data.

First, we create a `tidy_date` variable where we store the date according to the *ISO 8601* standard that states that dates should be expressed in the `YYYY-MM-DD` format. In consequence, the `raw_date` column is dropped and we keep the newly created `tidy_date` variable instead.

Also, we drop the `ticker`, `type`, `open`, `high`, and `low` columns because we think they are of no use for the analysis.

We store the modified data into the `IPC_tbl` (IPC tibble) object.

Time-Series Data Print

Next, we create some time-related variables, such as:

`tidy_year`: a `dbl` that stores the year (from 1996 - 2018).

`tidy_month`: a `dbl` that stores the month as a number (from 1 through 12).

`tidy_mday`: a `dbl` that stores the day number within each month (from 1 through 31).

`tidy_wday`: a categorical variable (`fctr`) that includes: `Mon Tue Wed Thu Fri`.

`tidy_hour`: a `dbl` that stores the hour (we have data from 5 through 20 hours).

`tidy_minute`: a `dbl` that stores the minute of the trade (from 0 through 59).

`tidy_time`: an `hms` (hour-minute-second) object that stores the time of the trade.

Table 1: Data summary

Name	Piped data
Number of rows	2133890
Number of columns	15
Column type frequency:	
Date	1
difftime	1
factor	1
numeric	12
Group variables	None

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
tidy_date	0	1	1996-01-02	2018-06-05	2007-07-25	5613

Variable type: difftime

skim_variable	n_missing	complete_rate	min	max	median	n_unique
tidy_time	0	1	19920 secs	73320 secs	42300 secs	647

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
tidy_wday	0	1	TRUE	5	Wed: 435537, Tue: 434397, Thu: 426174, Fri: 425693

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
trade_id	0	1	1066945.50	616001.13	1.00	533473.25	1066945.50	1600417.75	2.133890e+06	
last	0	1	23945.92	16340.42	2731.21	16502.76	24198.56	40435.79	7.813872e+04	
volume	0	1	5534110.54	5476930.82	0.00	1169795.00	4261727.00	8370513.25	1.259968e+09	
average_price	0	1	23945.91	16340.43	2731.36	16502.69	24199.01	40435.50	7.813872e+04	
no_trades	0	1	34.98	39.95	1.00	8.00	21.00	53.00	3.693000e+03	
no_bids	0	1	0.00	0.00	0.00	0.00	0.00	0.00	0.000000e+00	
no_ask	0	1	0.00	0.00	0.00	0.00	0.00	0.00	0.000000e+00	
tidy_year	0	1	2007.00	6.39	1996.00	2002.00	2007.00	2013.00	2.018000e+03	
tidy_month	0	1	6.43	3.43	1.00	3.00	6.00	9.00	1.200000e+01	
tidy_mday	0	1	15.84	8.75	1.00	8.00	16.00	23.00	3.100000e+01	
tidy_hour	0	1	11.24	1.92	5.00	10.00	11.00	13.00	2.000000e+01	
tidy_minute	0	1	30.52	17.33	0.00	16.00	31.00	46.00	5.900000e+01	

2.1 Computing Intraday Log-Returns

Next, we compute the intraday returns and assign them to the `log_ret` variable. We also convert our data into a time-friendly type of object called `tibble time` (we do this via the `as_tbl_time()` function).

```
## # A time tibble: 2,133,890 x 16
## # Index: tidy_date
##    trade_id tidy_date   log_ret tidy_time  last volume average_price no_trades
##       <int> <date>        <dbl> <time>    <dbl>  <dbl>         <dbl>     <dbl>
##  1        1 1996-01-02 NA       08:36     2777.      0         2777.         1
##  2        2 1996-01-02  0.      08:38     2777.      0         2777.         2
##  3        3 1996-01-02  0.      08:39     2777.      0         2777.         3
##  4        4 1996-01-02  0.      08:40     2777.      0         2777.         3
##  5        5 1996-01-02 -1.19e-4 08:41     2777.      0         2777.         3
##  6        6 1996-01-02  0.      08:42     2777.      0         2777.         3
##  7        7 1996-01-02  0.      08:43     2777.      0         2777.         3
##  8        8 1996-01-02  0.      08:44     2777.      0         2777.         3
##  9        9 1996-01-02  0.      08:45     2777.      0         2777.         3
## 10       10 1996-01-02  0.      08:46     2777.      0         2777.         3
## # ... with 2,133,880 more rows, and 8 more variables: no_bids <dbl>,
## #   no_ask <dbl>, tidy_year <dbl>, tidy_month <dbl>, tidy_mday <int>,
## #   tidy_wday <ord>, tidy_hour <int>, tidy_minute <int>
```

2.2 Narrowing down the time window for trades

First, it's useful to see how many different trading-days we have in our data.

```
## [1] 5613
```

So we have `5,613` different trading- days.

It's worth noting that, for the `tidy_time` variable, we have data from `05:32:00` all the way through `20:22:00`.

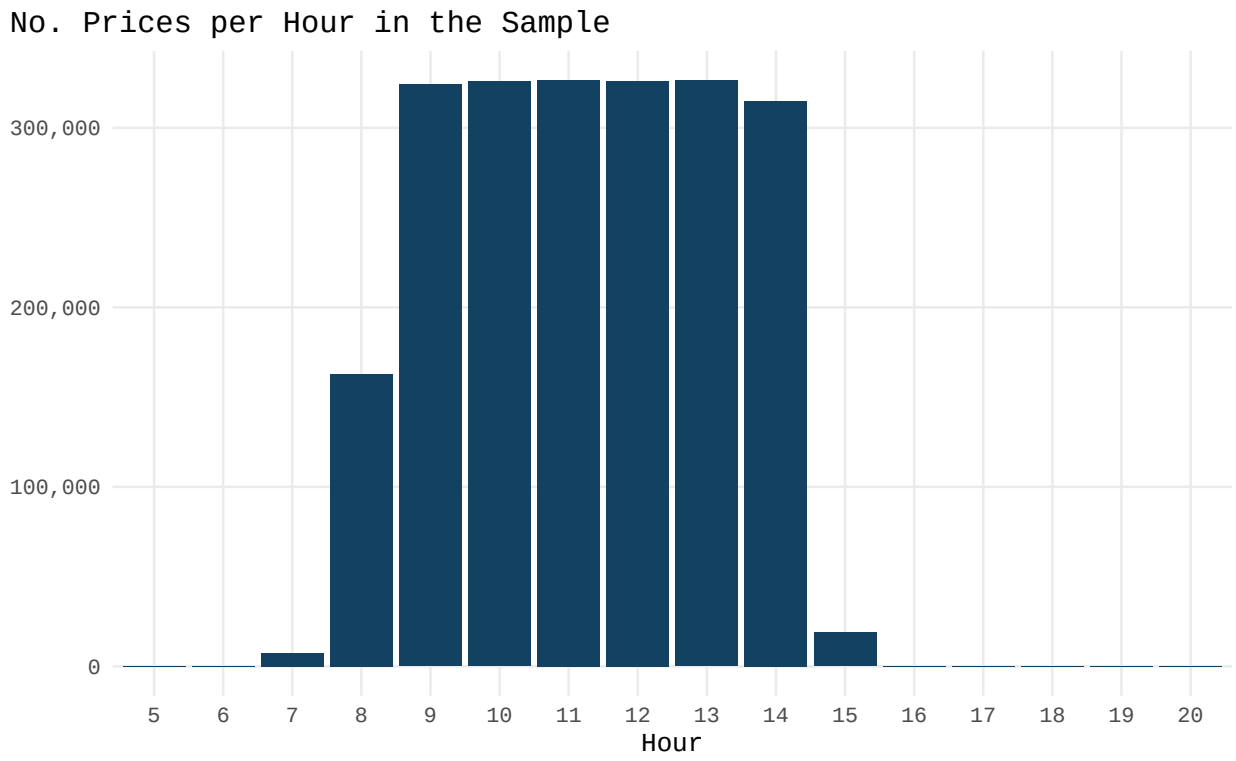
Let's see the earliest times in our data:

```
## # A tibble: 6 x 1
##   tidy_time
##   <time>
## 1 05:32
## 2 05:33
## 3 06:32
## 4 06:33
## 5 06:34
## 6 06:35
```

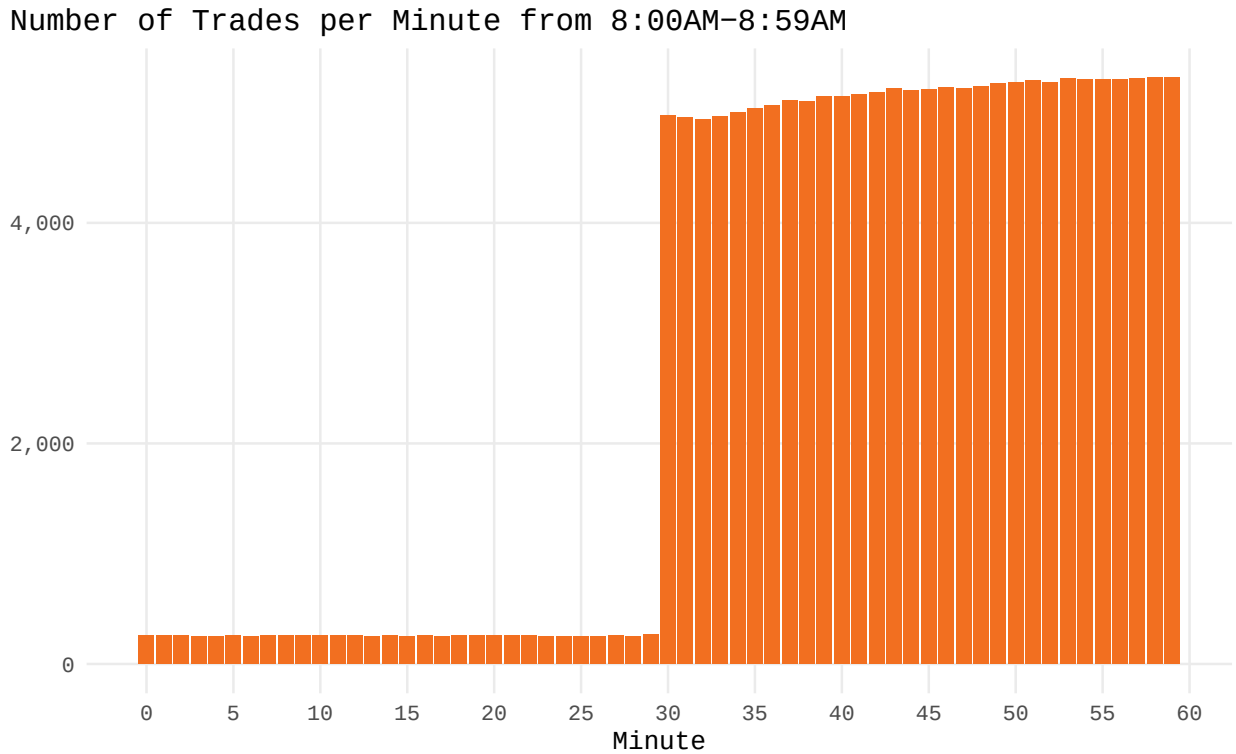
And the latest times in our data:

```
## # A tibble: 6 x 1
##   tidy_time
##   <time>
## 1 20:17
## 2 20:18
## 3 20:19
## 4 20:20
## 5 20:21
## 6 20:22
```

We can also visualize how many datapoints we have, per hour.

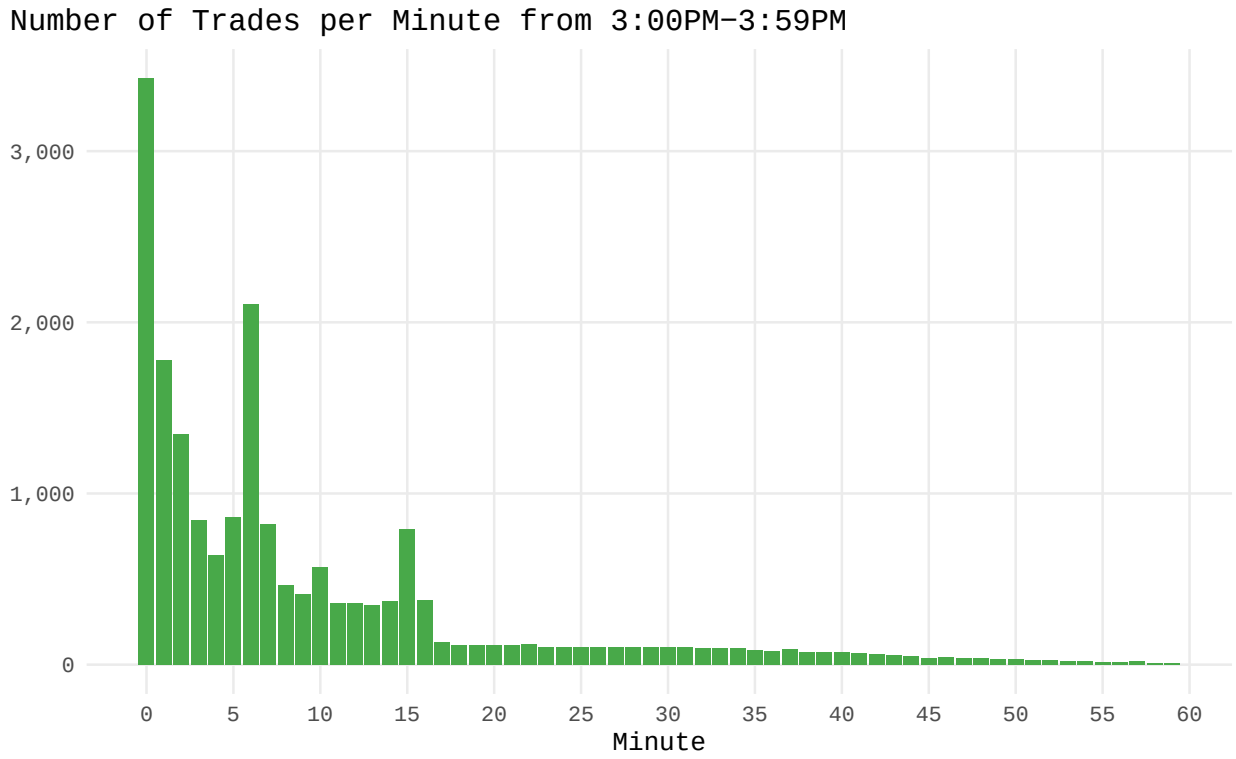


From the plot above, we see that most of the prices lie between 8:00AM - 3:00PM. It's interesting that the hours `8` and `15` have fewer datapoints. We can further inspect each one to see the hour-minute components of those trades.



As expected, the second half-hour is the most active (that is, the period from 8:30AM - 8:59AM).

Now, let's do the same for the `15` hour-mark:



Here, the time with most trades is 3:00PM exactly. That's what we need because we'll ignore the rest of the trades. Following Liu, Patton, Sheppard (2013), we'll discard any prices that were quoted outside normal business hours. Hence, we want trades that lie between `08:30` and `15:00`. We'll store these trades in a new object called `IPC_tfr_tbl` (IPC time-filtered returns tibble). Finally, we'll print the earliest and latest times in our data to check.

```
## # A time tibble: 2,102,987 x 16
## # Index: tidy_date
##    trade_id tidy_date   log_ret tidy_time  last volume average_price no_trades
##       <int> <date>        <dbl> <time>    <dbl>  <dbl>         <dbl>     <dbl>
##  1        1 1996-01-02 NA       08:36     2777.      0         2777.         1
##  2        2 1996-01-02  0.      08:38     2777.      0         2777.         2
##  3        3 1996-01-02  0.      08:39     2777.      0         2777.         3
##  4        4 1996-01-02  0.      08:40     2777.      0         2777.         3
##  5        5 1996-01-02 -1.19e-4 08:41     2777.      0         2777.         3
##  6        6 1996-01-02  0.      08:42     2777.      0         2777.         3
##  7        7 1996-01-02  0.      08:43     2777.      0         2777.         3
##  8        8 1996-01-02  0.      08:44     2777.      0         2777.         3
##  9        9 1996-01-02  0.      08:45     2777.      0         2777.         3
## 10       10 1996-01-02  0.      08:46     2777.      0         2777.         3
## # ... with 2,102,977 more rows, and 8 more variables: no_bids <dbl>,
## #   no_ask <dbl>, tidy_year <dbl>, tidy_month <dbl>, tidy_mday <int>,
## #   tidy_wday <ord>, tidy_hour <int>, tidy_minute <int>
```

```
## # A tibble: 6 x 1
##   tidy_time
##   <time>
```

```
## 1 08:30
## 2 08:31
## 3 08:32
## 4 08:33
## 5 08:34
## 6 08:35
```

```
## # A tibble: 6 x 1
##   tidy_time
##   <time>
## 1 14:55
## 2 14:56
## 3 14:57
## 4 14:58
## 5 14:59
## 6 15:00
```

We have filtered out 30903 datapoints.

One last check we need to do regarding this point is the amount of different days that we just filtered. To start, let's see all unique dates before narrowing the time window:

```
## [1] 5613
```

Now let's check for all unique dates after narrowing the time window:

```
## [1] 5608
```

Thus, we see that we have lost 5 days of data. For purposes of this analysis, we deem that to be valid.

2.3 Setting Overnight Returns to Zero

Another aspect that we want to take into account is: the fact that the overnight return should not be taken into account. Therefore, we set the first log-return of each day to zero. Once again, we store these values in a new object called `IPC_oar_tbl` (which stands for IPC overnight adjusted returns tibble).

```
## # A time tibble: 2,102,987 x 17
## # Index: tidy_date
##    trade_id tidy_date   log_ret tidy_time  last volume average_price no_trades
##       <int> <date>        <dbl> <time>    <dbl>  <dbl>         <dbl>     <dbl>
##  1        1 1996-01-02 NA       08:36     2777.      0         2777.         1
##  2        2 1996-01-02  0.      08:38     2777.      0         2777.         2
##  3        3 1996-01-02  0.      08:39     2777.      0         2777.         3
##  4        4 1996-01-02  0.      08:40     2777.      0         2777.         3
##  5        5 1996-01-02 -1.19e-4 08:41     2777.      0         2777.         3
##  6        6 1996-01-02  0.      08:42     2777.      0         2777.         3
##  7        7 1996-01-02  0.      08:43     2777.      0         2777.         3
##  8        8 1996-01-02  0.      08:44     2777.      0         2777.         3
##  9        9 1996-01-02  0.      08:45     2777.      0         2777.         3
## 10       10 1996-01-02  0.      08:46     2777.      0         2777.         3
## # ... with 2,102,977 more rows, and 9 more variables: no_bids <dbl>,
## #   no_ask <dbl>, tidy_year <dbl>, tidy_month <dbl>, tidy_mday <int>,
## #   tidy_wday <ord>, tidy_hour <int>, tidy_minute <int>, day_change <dbl>
```

Our first log-return should be zero, but we have `NA`. Let's check that this is the only `NA`:

```
## # A time tibble: 1 x 2
## # Index: tidy_date
##   tidy_date  log_ret
##   <date>       <dbl>
## 1 1996-01-02      NA
```

It is indeed our only `NA`. We can set it to zero.

```
## # A time tibble: 2,102,987 x 17
## # Index: tidy_date
##    trade_id tidy_date   log_ret tidy_time  last volume average_price no_trades
##       <int> <date>         <dbl> <time>    <dbl>  <dbl>         <dbl>     <dbl>
##  1        1 1996-01-02  0.      08:36     2777.      0         2777.         1
##  2        2 1996-01-02  0.      08:38     2777.      0         2777.         2
##  3        3 1996-01-02  0.      08:39     2777.      0         2777.         3
##  4        4 1996-01-02  0.      08:40     2777.      0         2777.         3
##  5        5 1996-01-02 -1.19e-4 08:41     2777.      0         2777.         3
##  6        6 1996-01-02  0.      08:42     2777.      0         2777.         3
##  7        7 1996-01-02  0.      08:43     2777.      0         2777.         3
##  8        8 1996-01-02  0.      08:44     2777.      0         2777.         3
##  9        9 1996-01-02  0.      08:45     2777.      0         2777.         3
## 10       10 1996-01-02  0.      08:46     2777.      0         2777.         3
## # ... with 2,102,977 more rows, and 9 more variables: no_bids <dbl>,
## #   no_ask <dbl>, tidy_year <dbl>, tidy_month <dbl>, tidy_mday <int>,
## #   tidy_wday <ord>, tidy_hour <int>, tidy_minute <int>, day_change <dbl>
```

We can check that all overnight returns are zero. We've created a variable (`day_change`) that goes to `1` when we have a change in day.

```
## # A tibble: 1 x 1
##   log_ret
##     <dbl>
## 1       0
```

Moreover, the number of overnight returns that have been set to zero should be one less than the number of different dates we have for the `tidy_date` variable. Let's check if this holds true:

```
## # A time tibble: 5,607 x 3
## # Index: tidy_date
##    trade_id tidy_date  log_ret
##       <int> <date>       <dbl>
##  1      457 1996-01-03       0
##  2      890 1996-01-04       0
##  3     1339 1996-01-05       0
##  4     1766 1996-01-08       0
##  5     2194 1996-01-09       0
##  6     2636 1996-01-10       0
##  7     3064 1996-01-11       0
##  8     3501 1996-01-12       0
##  9     3939 1996-01-15       0
## 10     4354 1996-01-16       0
## # ... with 5,597 more rows
```

We have `5,607` datapoints where `day_change = 1`. This should be one less than the amount of different dates (one less because the first date doesn't have `day_change` set to one).

```
## [1] 5608
```

Thus, we see that this holds true. We have correctly set the overnight returns to zero.

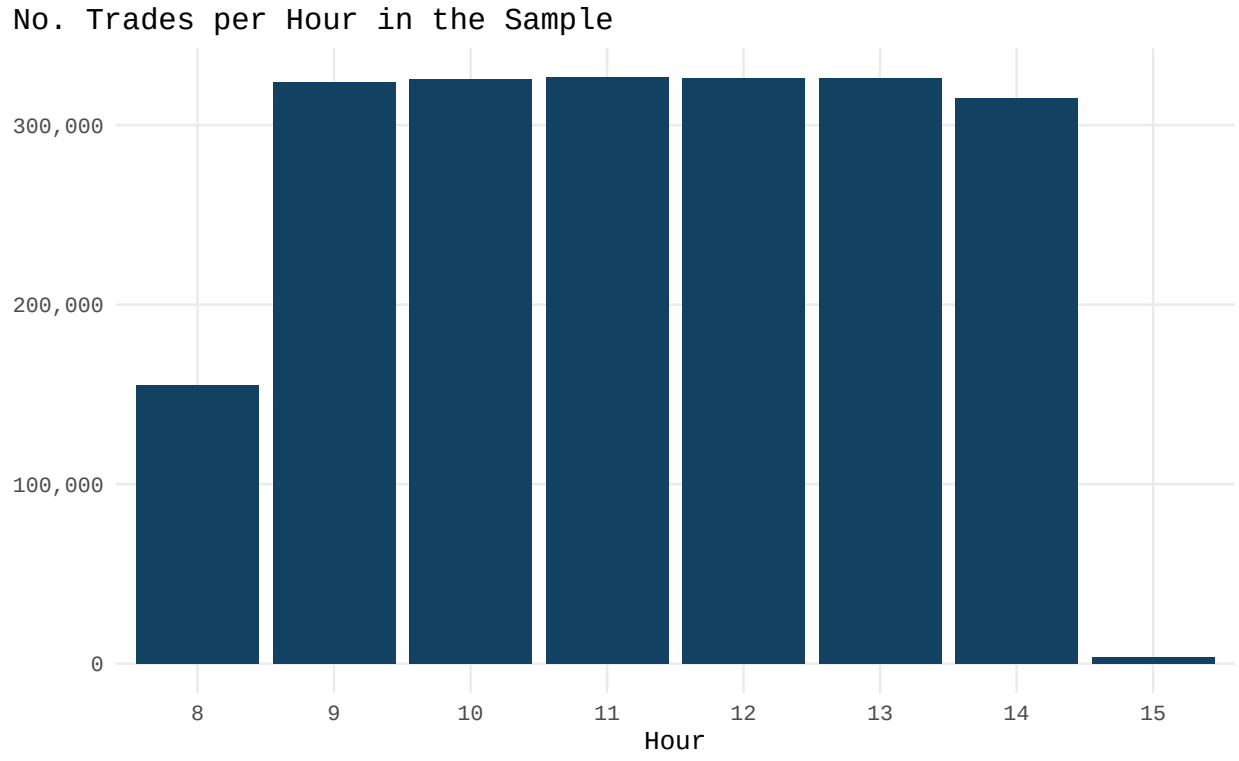
2.4 Removing Outliers

We want to filter out log-returns whose absolute value is greater than some threshold. We'll do this again later for 5-minute returns and 30-minute returns. We define the threshold such that we are filtering keeping $\sim 99\%$ of the original data. Log-returns that are outside of the defined threshold are immediately set to zero.

In this case we've set the variable `threshold` to the value (`0.0012`) such that we're filtering out $\sim 1\%$ of the data. We store the filtered data in the `IPC_fr` tibble (which stands for IPC filtered returns).

2.5 Padding the Time Series

Now, after all the data wrangling that we did, we still don't have a perfect time series, because it has some "holes" in it. We can check this by plotting, again, the number of log-returns we have per hour:



Most of the hours (9 - 14) have almost complete data. The hour number 8 has almost half the data points but that's natural since we only have half an hour there. However, almost all of the log-returns for the `15:00` mark seem to be missing. In normal business hours, we have 6 hours and a half of market trading. If we have one-minute trades, that means we should have $6.5 * 60 + 1 = 391$ trades/day. The `+1` in the equation comes from including the log-returns at exactly `15:00` hours.

```
## [1] "2020-09-11 08:30:00 UTC" "2020-09-11 08:31:00 UTC"
## [3] "2020-09-11 08:32:00 UTC" "2020-09-11 08:33:00 UTC"
## [5] "2020-09-11 08:34:00 UTC" "2020-09-11 08:35:00 UTC"
```

Therefore, we'll pad the time series with the mean log-return for each half-hour.