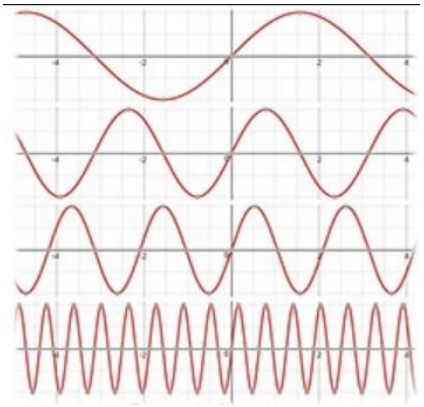
Gekko frequency blueprint



Thomas Thomsen, 24/6 2020

In this blueprint, some issues regarding frequencies in Gekko are discussed.

Current frequencies

The current frequencies annual (a), quarterly (q) and monthly (m) are relatively easy. These have the following characteristics:

- **Nesting**. The current frequencies fit cleanly inside each other.
- **Fullness**. Missing observations are not expected inside a sequence of observations. For instance, the months 2020m1, 2020m2 and 2020m3 fit inside the quarter 2020q1, and if a timeseries has data for 2020m1 and 2020m3, it would not be expected that 2020m2 could be missing (this would be regarded as a data error/omission).
- **Duration**. the current frequencies each have similar duration. The length of the years vary with at most one day, whereas the quarters and months are pretty stable, too, apart from February being a bit shorter.

In the following, weekly (w) and daily (d) frequencies are considered.

Weekly frequency

The definition of a week is a little bit technical, and we will only consider the week definition of the ISO 8601 standard (official standard in many countries including Danmark, https://en.wikipedia.org/wiki/ISO_8601). This standard entails that a week always has 7 days (no weeks are cut into two parts). If a week starts in December and ends in January, the week is assigned to the year wherein the majority of days reside (same as looking at where the Thursday of the week resides).

- **Nesting**. Following the ISO 8601 standard, a week is always assigned to a particular year, and will have a number up to 52 or 53. The week number can be easily obtained by counting the number of Thursdays since January 1. Regarding nesting, there is the following problem. The first week of a year may have up to three days that actually belong to the calendar year that is ending, and the last week of a year may have up to three days that actually belong to the calendar year that is starting. Therefore, there is no clean aggregation of weeks into calendar year. Similarly, there

is no clean nesting of weeks inside quarters or months, whereas days obviously nest cleanly inside weeks.

- **Fullness**. Missing data for a whole week is not to be expected, since ISO 8601 does not cut weeks into two parts to make them fit cleanly inside years. Any week of the year usually has at least one formal business day.
- **Duration**. An ISO 8601 week always has the same duration (7 days). Weekly data may, of course, represent only business days (5 days), but the week numbering scheme is still the same. Assigning "active" weekdays of the week is also relevant regarding daily frequency, and for weekly frequency, knowledge of implicit "active" weekdays can be used when converting to other frequencies.
- **Note**. Following ISO 8601, the first day of a week is always a Monday. In most countries, the working week is from Monday to Friday, but many countries observe a Sunday to Thursday working week (for such countries, the ISO 8601 standard does not fit as well).

Daily frequency

Days have the following characteristics:

- **Nesting**. A day always belongs to a particular year, quarter or month. The relationship regarding weeks is a bit more tricky around New Year. As mentioned above, up to three days at the end of a calendar year may belong to the first week of the following year. And up to three days at the start of a calendar year may belong to the last week of the preceding year (week 52 or 53 of that year).
- **Fullness**. Missing data for a daily observation is often to be expected. Among the weekdays, "active" days can be defined, for instance Monday to Friday. Holidays etc. are more difficult, since official holidays may vary irregularly between years and countries, and some holidays may even be one-time events. But again, whereas a missing year, quarter, month or week in the middle of other years/quarters/months/weeks would raise a red flag (data error/omission), a missing day in the middle of a sequence of other days would often not be considered an error, but rather a cause for further investigation (is it a weekend day, a holiday, or does it rather just signify that no activity took place that day?). Conversions to and from other frequencies should probably have some lenience, so that the odd missing day here and there does not crash the calculations. More on this in the recommendations.
- **Duration**. A day always has the same duration.

Recommendations

Calendar details are easy to get caught up into, including the date representations (string formatting) of different frequencies. The following proposal tries to implement a "reasonably correct" approach, without spending too much effort on details that are unlikely to be of practical relevance.

In general, the approach is to handle "holes" in high frequencies like weeks and days via filtering. So the weeks or days are defined in their broadest sense: up to 53 weeks or 366 days per year, and then we apply filtering to suppress some of these if they are not present/active. Regarding weeks, the issue (filter) only concerns whether a particular year has 52 or 53 weeks.

Two kinds of "active" day definitions will be implemented:

- Weekday-based. Active weekdays are assigned in the form of a string of 0's and 1's. For instance, '111100' means that Monday to Friday are business days, whereas Saturday and Sunday are weekends. Therefore, other business days than Monday to Friday are easy to implement.
- Holiday calendar. Holidays can be loaded in the form of a list of dates. Gekko should issue a warning if a date outside this list span is queried. Inbuilt holiday calendars could be implemented as Gekko functions that return such lists (algorithms that get these dates approximately right for a given country exist).

Regarding weekly frequency: as mentioned above, a week belonging to a particular year may contain up to three days from a different year (around New Year). Weekly frequency can react to the two above-mentioned definitions of "active" days when converting between frequencies.

Why implement active days definitions at all? An alternative could be that active days are simply defined while loading data, for instance when loading rows of daily observations from an Excel sheet. In that case, the active days could be understood as the row dates shown in that spreadsheet, but there is a catch. What if some of the days are omitted by accident? Someone might have forgotten to update part of the dataset, so this should count as an is an error/omission, and not be interpreted as weekends/holidays.

Therefore, Gekko must implement some sense of which days are expected to contain data (business days, non-holidays), and which days can be safely ignored/omitted. When data is aggregated or converted to other frequencies, the notion of active days can be combined with some suitable options regarding missing values. Such options already exist regarding array-series (http://t-t.dk/gekko/docs/user-manual/index.html?appendix_missings.htm), and the options regarding weekly and daily observations could use the same ideas. For instance, if a particular day is designated as a holiday, should it still be shown when printing (and if so, with M for missing, N for non-existing, or with 0?), or should it be skipped? The

options could control this, and likewise what to do about missings when converting between frequencies (collapsing/interpolating).

So the main points are that (1) Gekko will implement ISO 8601 regarding the definition of weeks and their numbering, (2) business days inside the weeks can be stated, (3) a holiday calendar may be added in the form of an explicit ("manual") list of dates.

Details

- Besides 'd' frequency, there will be no separate 'b' frequency type to imply "business days". Instead, a business day pattern like '1111100' can be activated via options. (EViews and others have a separate 'b' frequency).
- Weeks will always start with Monday. The week number can always be found by counting the number of Thursdays since January 1.
- Date string formats are postponed for now. This is the question of how to represent, say, November 25, 2021? Is that "25/11 2021", or "11/25/2021", or "2021-11-25", or ... ?

Expressions and math

- Lags/leads in weeks are simple in the sense that omitted weeks are not expected, so x[2021w8-1] is always the same as x[2021w7]. The only issue in this calculation is to remember that some years have 53 weeks. Lags/leads in days are more complicated. Gekko will allow this to be optional, so that lags/leads can either be "raw" (no gaps), or defined via active days (business days + non-holidays). This means that for instance percentage growth from day to day will be easy to show in raw form (Monday showing difference relative to Sunday), or with business days (Monday showing difference relative to Friday). With a holiday calendar, Gekko can adjust for that, too. Still, it should be remembered that day to day percentage growth is probably not much used because of noise etc.
- Some functions must reflect business days/holidays options, for instance sumt() and avgt(). Also, date differences must reflect this, for instance 2021m12d31 - 2021m1d1 + 1 is counting the days of 2021, but this should only show the "active" days.
- In the same manner, functions like pack() and unpack() to convert series to and from vectors/matrices should observe business days/holidays options. This will prevent unwanted missing values to flow into the matrices.
- The TIMEFILTER command in Gekko has some overlap with all of this.

In general, when aggregating/collapsing/interpolating daily data, Gekko should have options that make it "forgiving" regarding missing days. Imagine that effort has been made to implement business days and a suitable holiday calendar for a daily dataset, but when collapsing the dataset into months, one of the months contains a missing value for some

daily date. If this is the result of a single day containing a missing value, this could perhaps be forgiven. But if it is due to 15 days missing from the month, it is probably because of a data error/omission. Forgiveness regarding daily frequency could be controlled via options, spanning from strict to completely loose, and commands like COLLAPSE, INTERPOLATE, and some of the in-built functions could have local options that control how missing data in daily observations are treated.

Technically, inside Gekko, timeseries are represented as 1-dimensional arrays of numbers. To keep matters as simple as possible, Gekko will internally implement a fixed 53 weeks per year, and 366 days per year, so that jumping a year ahead in the data is simply a question of adding 53 or 366 to some array index. Some of those weeks or days may be "inactive", so there will be a bit of data redundancy, especially if daily data is only present for business days, for instance Mondays to Fridays. Still, the redundant missing values in that case is a small price to pay for the logical clarity of the underlying Gekko data structures. The date corresponding to element *i* in such a Gekko array is simply a matter of counting days in a standard calendar, and does not depend upon settings like business days or a custom list of holidays. The only technicality in that respect is whether the year is a leap year or not.[1] Also, this makes it possible for other software packages to read Gekko protobuffer files (.gbk databanks) with relatively little headache and confusion regarding daily and weekly data.

[1] In general, years divisible by 4 are leap years. With this exception: years that are divisible by 100, but not by 400, do not contain a leap day.