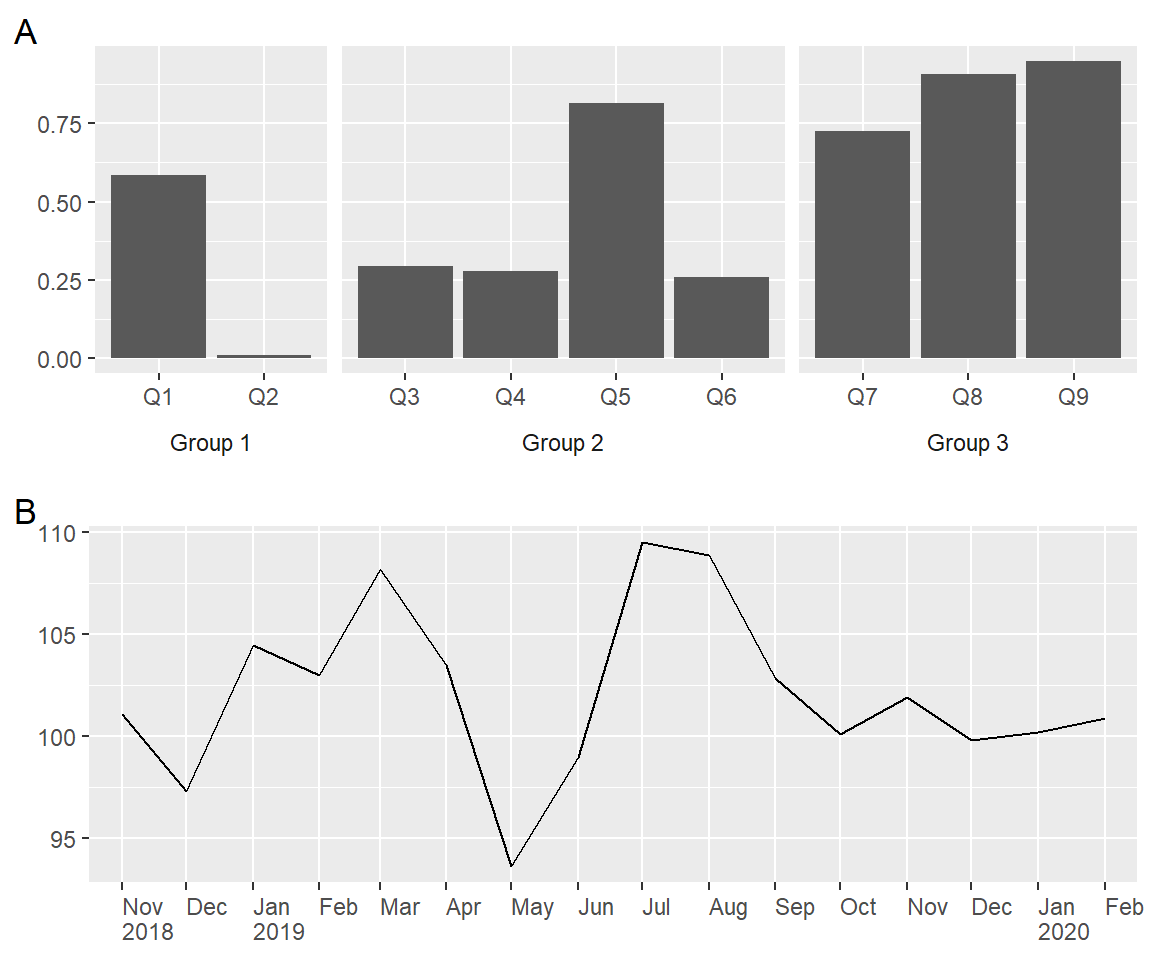
**Multi-level labels with ggplot2**

Categories: [data-viz](https://deploy-preview-15--dmitrijskass.netlify.com/categories/data-viz)

2019-06-30

* [Bar chart](https://deploy-preview-15--dmitrijskass.netlify.com/2019/06/30/multi-level-labels-with-ggplot2/#bar-chart)
  + [Possible solutions](https://deploy-preview-15--dmitrijskass.netlify.com/2019/06/30/multi-level-labels-with-ggplot2/#possible-solutions)
  + [Example with faceting](https://deploy-preview-15--dmitrijskass.netlify.com/2019/06/30/multi-level-labels-with-ggplot2/#example-with-faceting)
* [Line chart with dates](https://deploy-preview-15--dmitrijskass.netlify.com/2019/06/30/multi-level-labels-with-ggplot2/#line-chart-with-dates)
  + [Possible solutions](https://deploy-preview-15--dmitrijskass.netlify.com/2019/06/30/multi-level-labels-with-ggplot2/#possible-solutions-1)
  + [Example with text wrap](https://deploy-preview-15--dmitrijskass.netlify.com/2019/06/30/multi-level-labels-with-ggplot2/#example-with-text-wrap)

Recently I needed to create **multi-level labels with ggplot2** and had no idea how to do it. Multi-level labels imply some sort of hierarchical structure in data. For example, survey questions may be grouped by topics and dates on the timeline may be grouped by years. A 15-minute search on Google provided me with various solutions described on Stack Overflow that worked with varying success for different types of charts. An important aspect is whether data points between groups should be connected. The bar chart (plot A below) is an example where data points from different groups should not be connected and a line chart (plot B below) is an example where data points should be connected both within and between groups:



Below, I discuss possible solutions to multi-level labels for these two charts. The line chart will have dates on the x axis as this is probably the most popular case. I deliberately keep charts simple with no customization of colours, etc.

**Bar chart**

First, let’s simulate the data for the bar chart. Let it be a result of a hypothetical survey with 9 questions labelled from Q1 to Q9 and combined into 3 groups. These will substitute the x axis labels. The y axis values are drawn from a uniform distribution and represent a proportion of respondents that correctly answered corresponding questions.

*# For `tibble()`, `if\_else()` and pipe `%>%`.*

**…**

*# Reproducibility.*

…

*# Counts of elements in each of the three groups.*

…

*# Simulate data.*

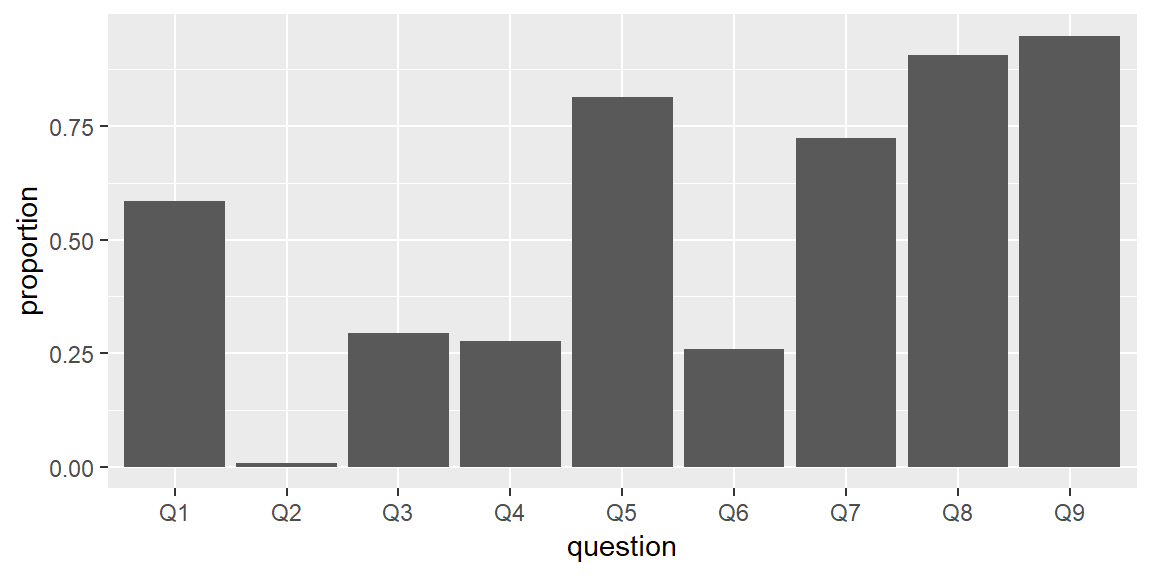
…

*# Print data.*

…

Here is a simple bar chart:

**…**



Why would you want to add a grouping variable? To make the chart more intuitive for the reader by clearly showing that questions belong to different topics.

**Possible solutions**

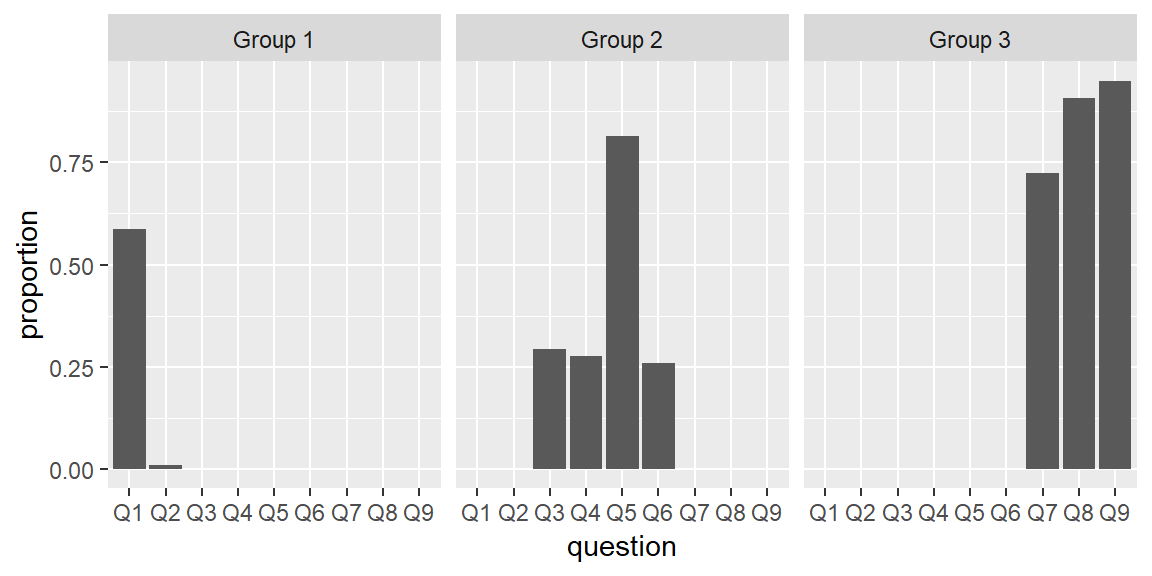
The most popular solutions for bar charts employ faceting with either facet\_grid() (for example, [here](https://stackoverflow.com/questions/48552671/ggplot2-show-category-and-sub-category-for-x-axis-labels/48555862#48555862)) or facet\_wrap() (for example, [here](https://stackoverflow.com/questions/44247239/grouping-on-the-x-axis-in-ggplot2/44249048#44249048)). The latter does not have an argument space that lets the width of facets vary, which in turn forces all bars to have the same width. Therefore, if a number of categories differ among groups then facet\_grid() should be preferred. I really liked this approach and demonstrate it below in steps.

**Example with faceting**

The faceting approach starts with a simple bar chart that is turned into a [small multiples](https://en.wikipedia.org/wiki/Small_multiple) bar chart with a

facet\_grid():

…

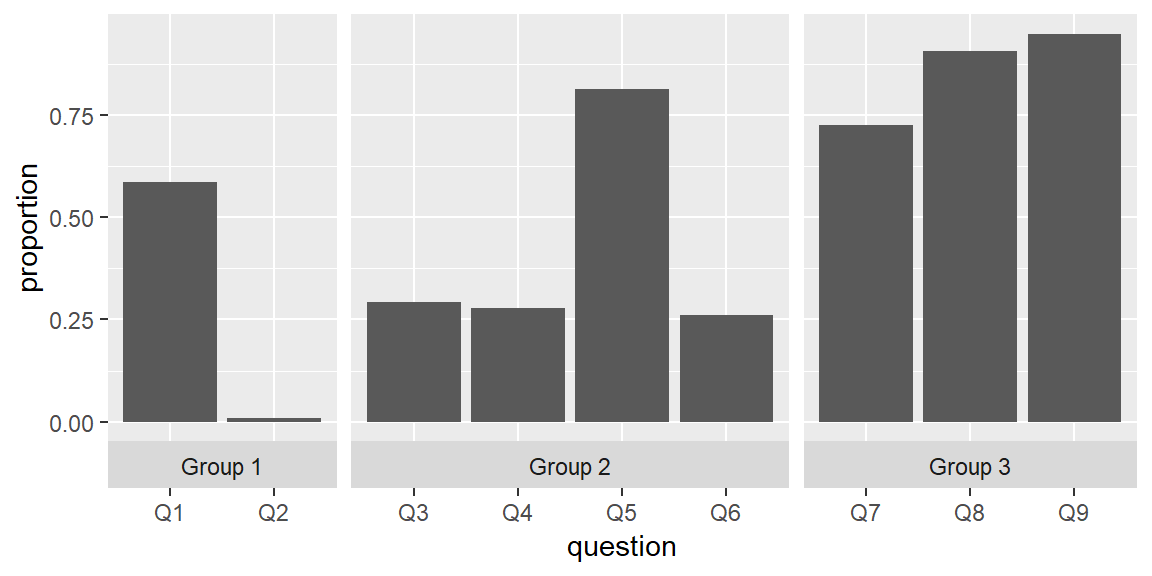


Let’s rewrite the code and add three additional arguments for the faceting function:

… *# Let the x axis vary across facets.*

… *# Let the width of facets vary and force all bars to have the same width.*

… *# Move the facet labels to the bottom.*

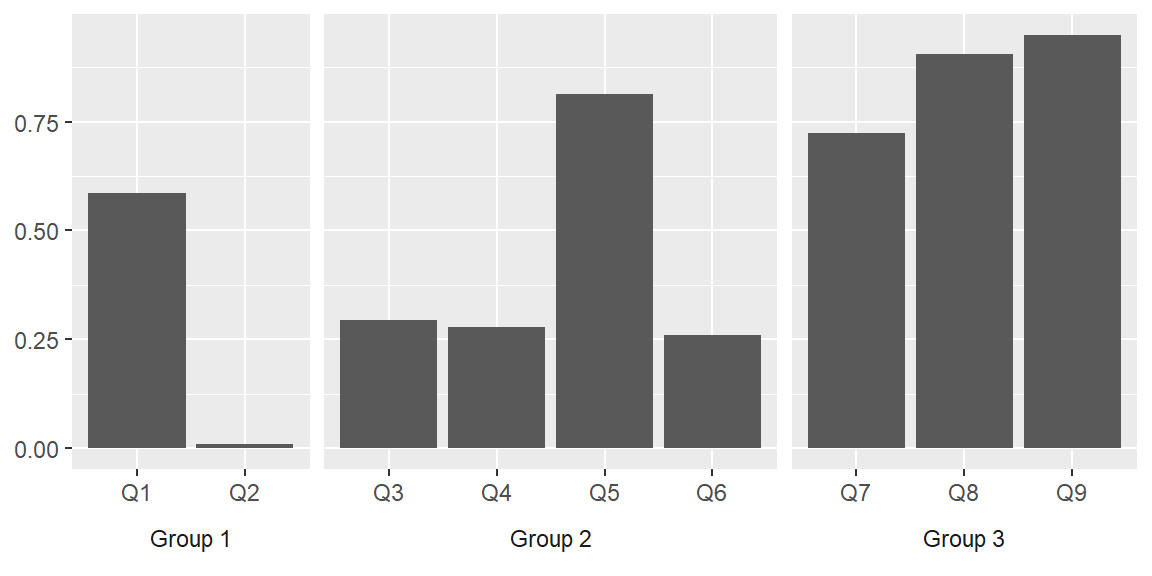


The final step is to customize the non-data components of the chart with the theme() function:

… *# Place facet labels outside x axis labels.*

… *# Make facet label background white.*

… *# Remove x and y axis titles.*



Done.

**Line chart with dates**

Again, we start with simulating data for the line chart. Let it be the sales volume of some healthy product over 16 months from November 2018 to February 2020. I deliberately chose this period as it partly covers three years.

*# Number of months.*

…

*# Reproducibility.*

…

*# Simulate data.*

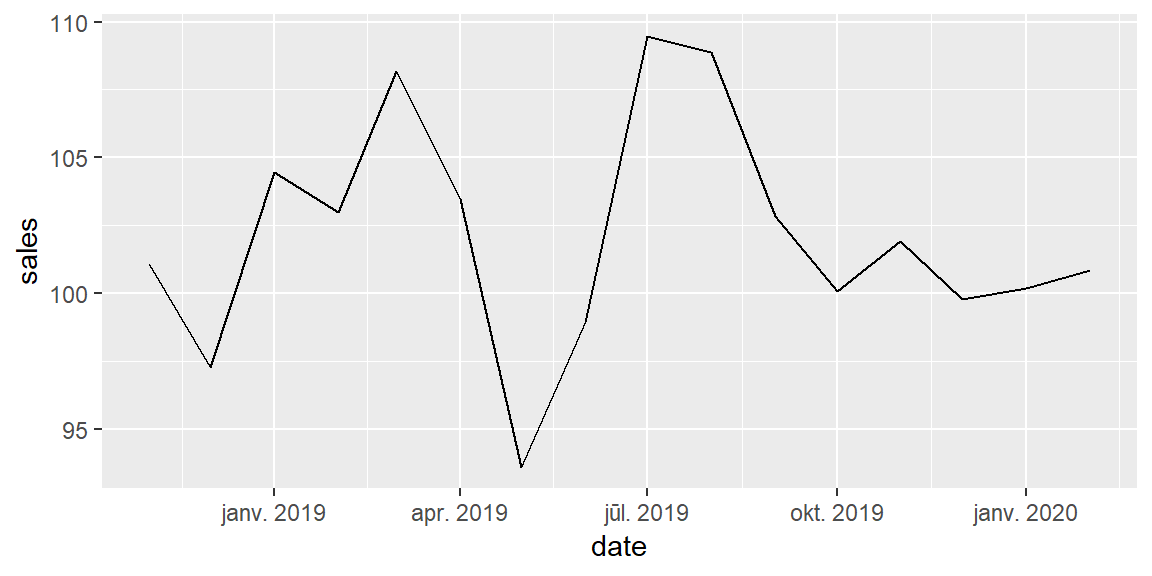
…

*# Print data.*

…

The first step is to create a simple line chart:

…



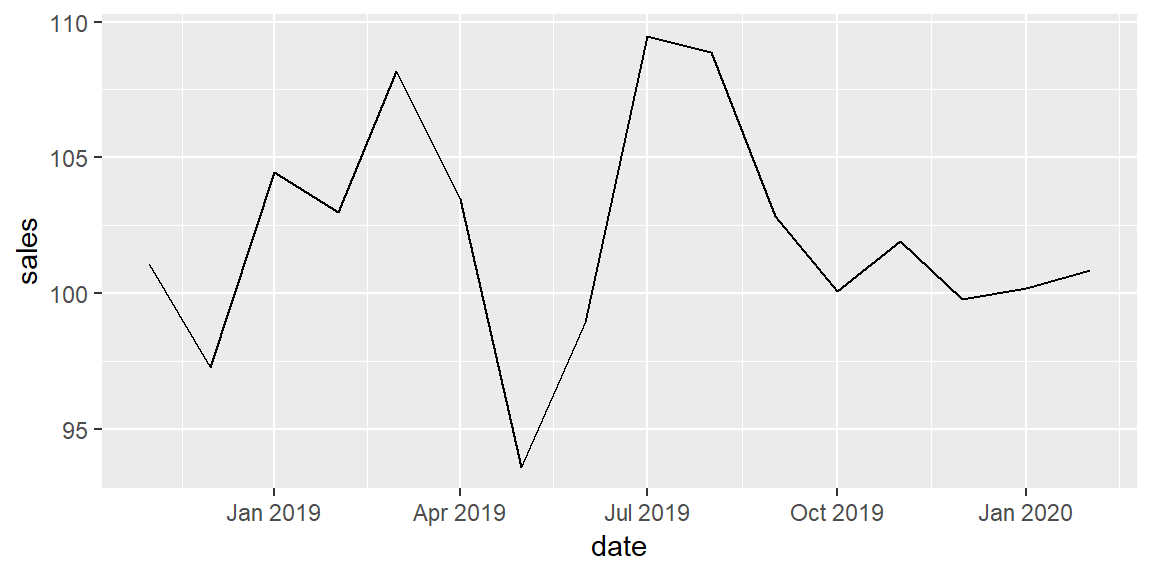
Your x axis labels may look differently depending on regional settings. My default region is Latvia. Locale can be changed with Sys.setlocale():

*# Change locale.*

…

*# Reprint the chart.*

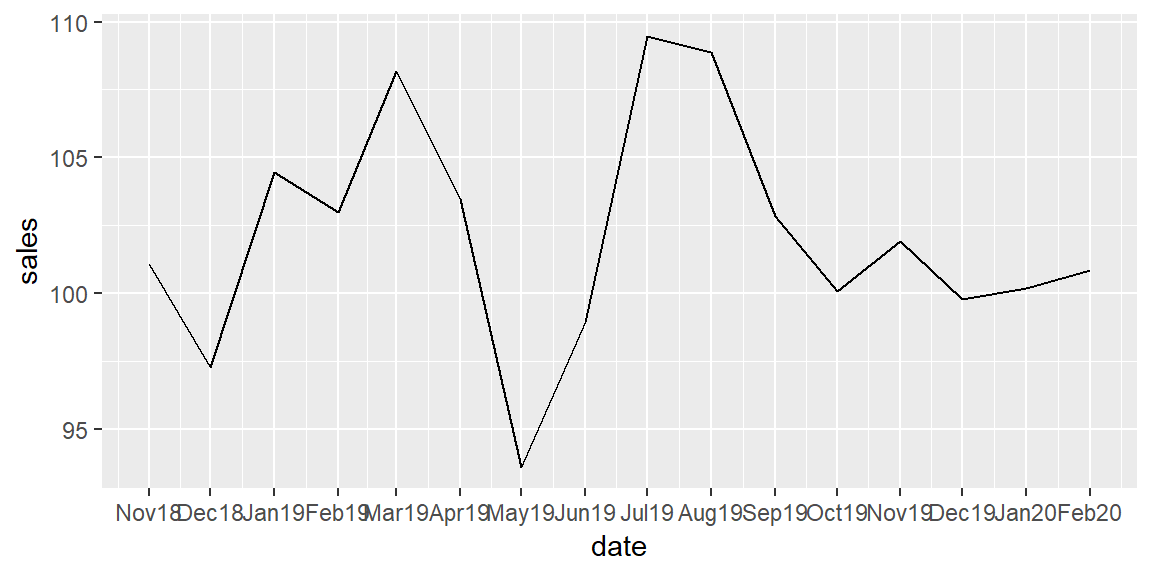
…



Why would you want to use multi-level labels for the x axis in this case? Because labelling each month creates a lot of clutter even with a short date format:

… *# Date labels for each month.*

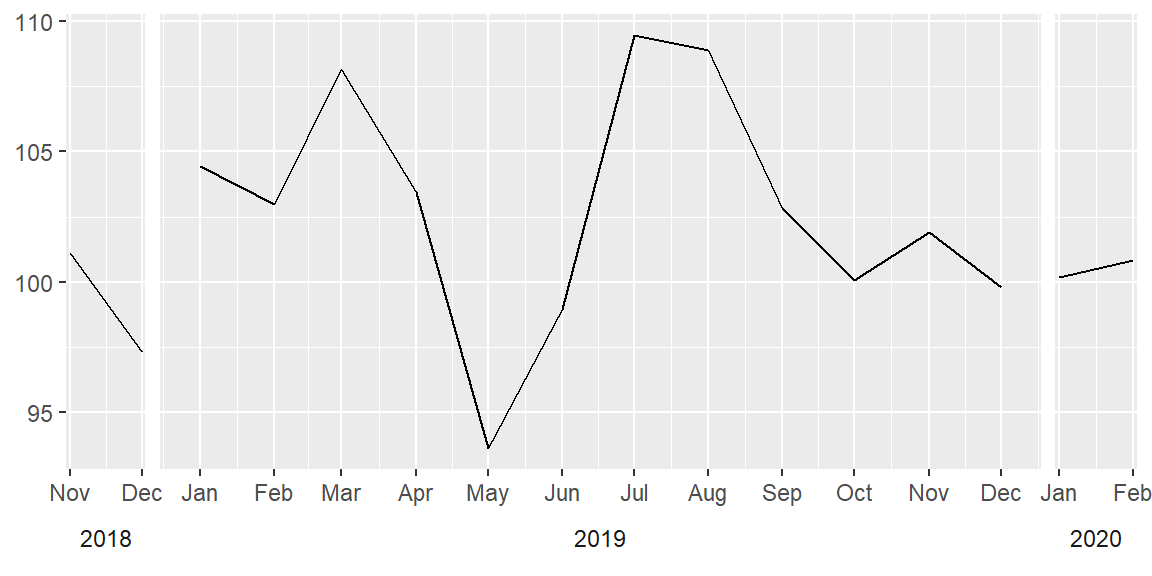
… *# Date format: abbreviated month and a 2-digit year.*



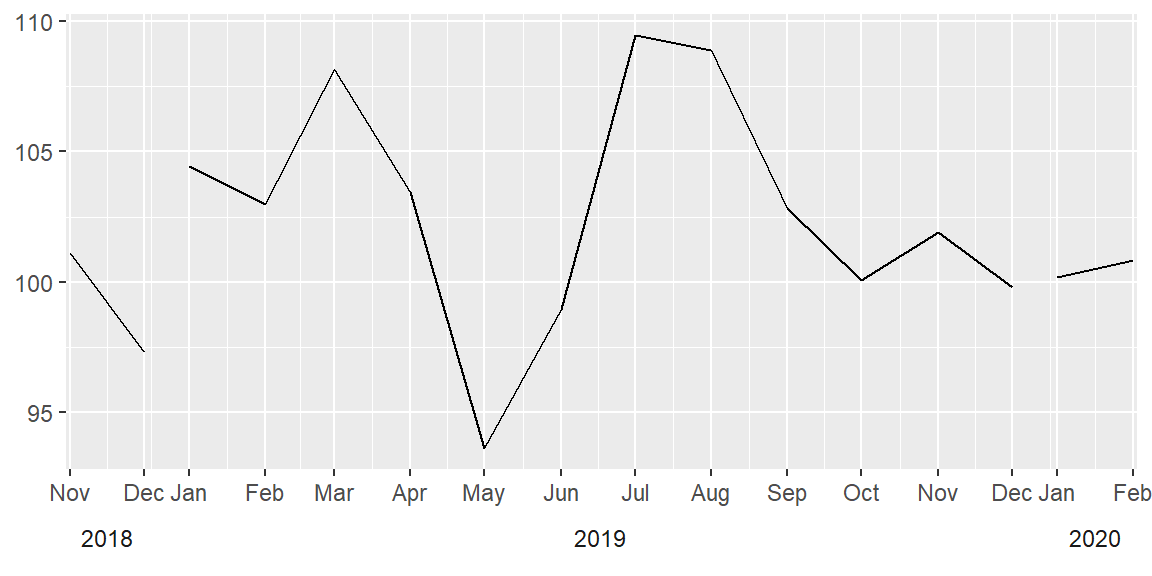
The faceting approach is not suitable in this scenario because we need to connect data points between facets:

*# Faceting approach does not suit this chart.*

…



Even if we remove the white space between the facets with theme(panel.spacing = unit(0, units = "cm"), the end data points of facets will still be disconnected. Notice the discontinuities between Decembers and Januaries:



**Possible solutions**

One solution offered [here](https://stackoverflow.com/questions/20571306/multi-row-x-axis-labels-in-ggplot-line-chart/20574024#20574024) is to use *annotations*. I don’t use this approach for two reasons:

1. coord\_cartesian(clip = "off") can cause unexpected results as it allows drawing of data points anywhere on the plot, including in the plot margins.
2. Fixed horizontal positions for text annotations in annotate(geom = "text", x = 2.5 + 4 \* (0:4)) do not depend on data and other ggplot2 customizations, such as axis limits. This solution may not work for a different data set.

Another solution with *grobs* [here](https://stackoverflow.com/questions/44616530/axis-labels-on-two-lines-with-nested-x-variables-year-below-months/44616739#44616739) appears overly complicated to me. Instead, I offer a different solution, which seems to me simple and generalizable, i.e. it does not depend on a particular data set.

**Example with text wrap**

My alternative solution is to modify x axis labels with a text formatting function (in this case - date formatting). Suppose we have a date vector. The function does three things:

1. Gets abbreviated names of months.
2. Gets the four-digit years.
3. Pastes the years behind months if
   * it is the first element in vector, or
   * the previous date in a vector has a different year component (it does not have to be January).

The separator is a new line symbol \n, which does the trick - it wraps the text and makes it *look* like a multi-line label. However, it is still a single vector of x axis labels, whose appearance may be manipulated with theme(axis.text.x = element\_text(...)).

Here is the function:

… *# Abbreviated name of the month.*

… *# Year as a 4-digit number.*

… *# Conditions for pasting.*

Try it on two date vectors:

*# Test the function.*

…

We are now ready. Let’s start from the beginning with a simple line chart. Then supply the above defined date formatting function to scale\_x\_date(labels = ...) among few other arguments. As the final touch, left-justify the labels with the theme().

… *# Customize x axis.*

… *# Date labels for each month.*

… *# No additional labels in-between `date\_breaks`.*

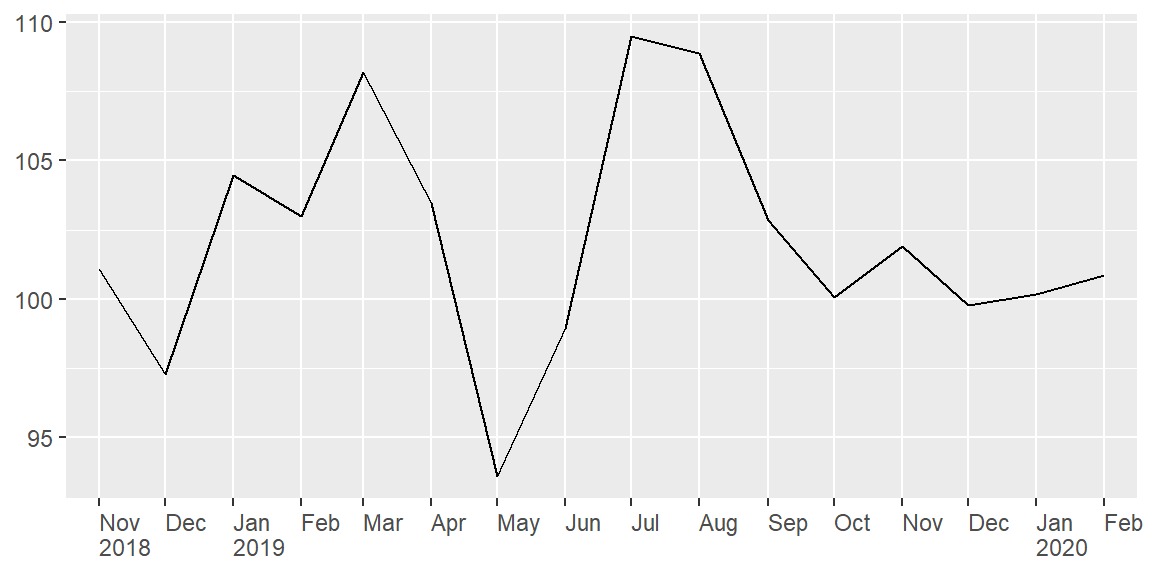
… *# Add 15 days to the x-axis on the left and on the right.*

… *# Supply a user-defined date formatting function.*

… *# Customize the non-data components.*

… *# Left-justify x-axis labels.*

… *# Remove x and y axis titles.*



Done.

If abbreviated month names still look cluttered, we may use just the first letter of each month’s name by altering the format\_dates() function and replacing

… *# Abbreviated name of the month.*

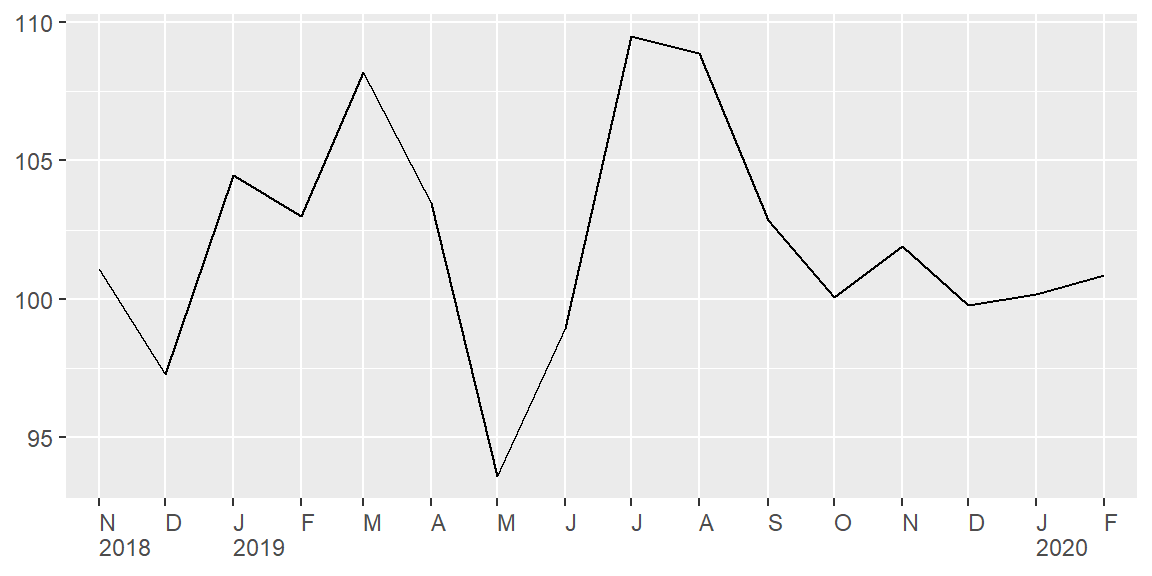
with

… *# Abbreviated name of the month.*

… *# May or may not be needed depending on your locale.*

… *# Extract just the first letter.*

Then the chart would look like this:



I would appreciate any comments or suggestions. Please leave them below, no login required if you check “I’d rather post as a guest”.