Data Exercise, Week 4

This week's data exercise introduces some new tools. There are four objectives for this exercise:

- reinforce techniques and make sure you're comfortable with the skills from the first assignment;
- teach you how to make a new type of graph: area plots;
- teach you how to start with a basic plot and add elements to it;
- teach you how to plot multiple graphs in one: faceted plots.

We're going to use the same CER Energy Futures data that you used for the data assignment and that I used for the filtering demo. As usual, I'm going to use this document to introduce the concepts, and I'd like you to use the code I gave you to build your own RMarkdown document to make all of this happen.

> Use the template that I gave you for Assignment 1 if you want a clean RMD to start with, and then anything you see in a code chunk on this page, you should be able to use to populate your file and see if you can get it to run. Add one thing at a time, knit your file, and make sure you understand what each bit is doing. If not, re-read the notes here.

Always start with a code chunk for your packages. I also use `knitr::opts_chunk$set(message=F,warning=F)` to set the default for all code chunks in my document to suppress any R warnings or messages in my output. This will clean up your HTML a lot, but it also makes it harder for me to see things that might be causing you issues in your code.

```
#load packages
library(kableExtra)
library(readxl)
library(janitor)
library(tidyverse)
library(lubridate)
```

```
library(scales)
library(viridis)
#set default chunk options
knitr::opts_chunk$set(message=F,warning=F)
```

Downloading this week's data

Like the filtering tutorial, let's use the crude oil production dataset from the CER.

> A hint for this: you can speed up your code by including a chunk option to cache the data so it won't download every time, using something like this: `{r read data,cache=T}`. The catch is that, if there are changes in the data (for example, if you're using a weekly EIA data set), you'll have to clear your cache to get the new version. You can also use the trick I have below, which comes with the same caveat:

> This will check to see if you already have a file called `cer_crude.csv` in the folder in which this RMD file is saved, and if so it won't download a new one. But, as above, if the target file is updated, you'll have to delete your version or comment out the `if()` statement before R will download a new one.

Examining the data

A good habit to get into is to use a code chunk at this point (you can delete it later) to have a look at your data and make sure you have what you're expecting. You could also just run the chunk above and look at the data in your environment if you like, but I'm doing everything here in a self-contained document.

There are a couple of easy ways to do this.

First, you could just print a few rows of the data using `head()`:

```
#look at the first 5 observations from the data
cer_crude %>% head(5)
```

```
# A tibble: 5 × 7
     x1 scenario        unit                    region  variable     year value
  <dbl> <chr>           <chr>                   <chr>   <chr>       <dbl> <dbl>
1     1 Canada Net-zero Thousand Barrels per day Alberta (Upgraded …  2005  522.
2     2 Canada Net-zero Thousand Barrels per day Alberta (Upgraded …  2006  619.
3     3 Canada Net-zero Thousand Barrels per day Alberta (Upgraded …  2007  652.
4     4 Canada Net-zero Thousand Barrels per day Alberta (Upgraded …  2008  620.
5     5 Canada Net-zero Thousand Barrels per day Alberta (Upgraded …  2009  722.
```

You'll see a familiar structure with variables `scenario`, `region`, `variable`, `year`, `value`, and `unit`. `unit` and `region` are new, so you might want to check into that a bit more by looking at all the distinct values in those columns.

Here, you can use `select()` to grab a column, and then use the `distinct()` command to get all the unique elements in a variable. You could, alternatively, type `unique(cer_crude$unit)` in the console to get the same information.

First, for unit:

```
cer_crude %>% select(unit) %>% distinct()
```

```
# A tibble: 2 × 1
  unit
  <chr>
1 Thousand Barrels per day
2 Thousand Cubic Metres per day
```

The data has values expressed in different units, so that adds another choice for you to make when we graph or manipulate the data.

And now, region:

```
cer_crude %>% select(region) %>% distinct()
```

```
# A tibble: 9 × 1
  region
  <chr>
1 Alberta
2 Canada
3 British Columbia
```

```
4 Nova Scotia
5 Saskatchewan
6 Manitoba
7 Newfoundland and Labrador
8 Northwest Territories
9 Ontario
```

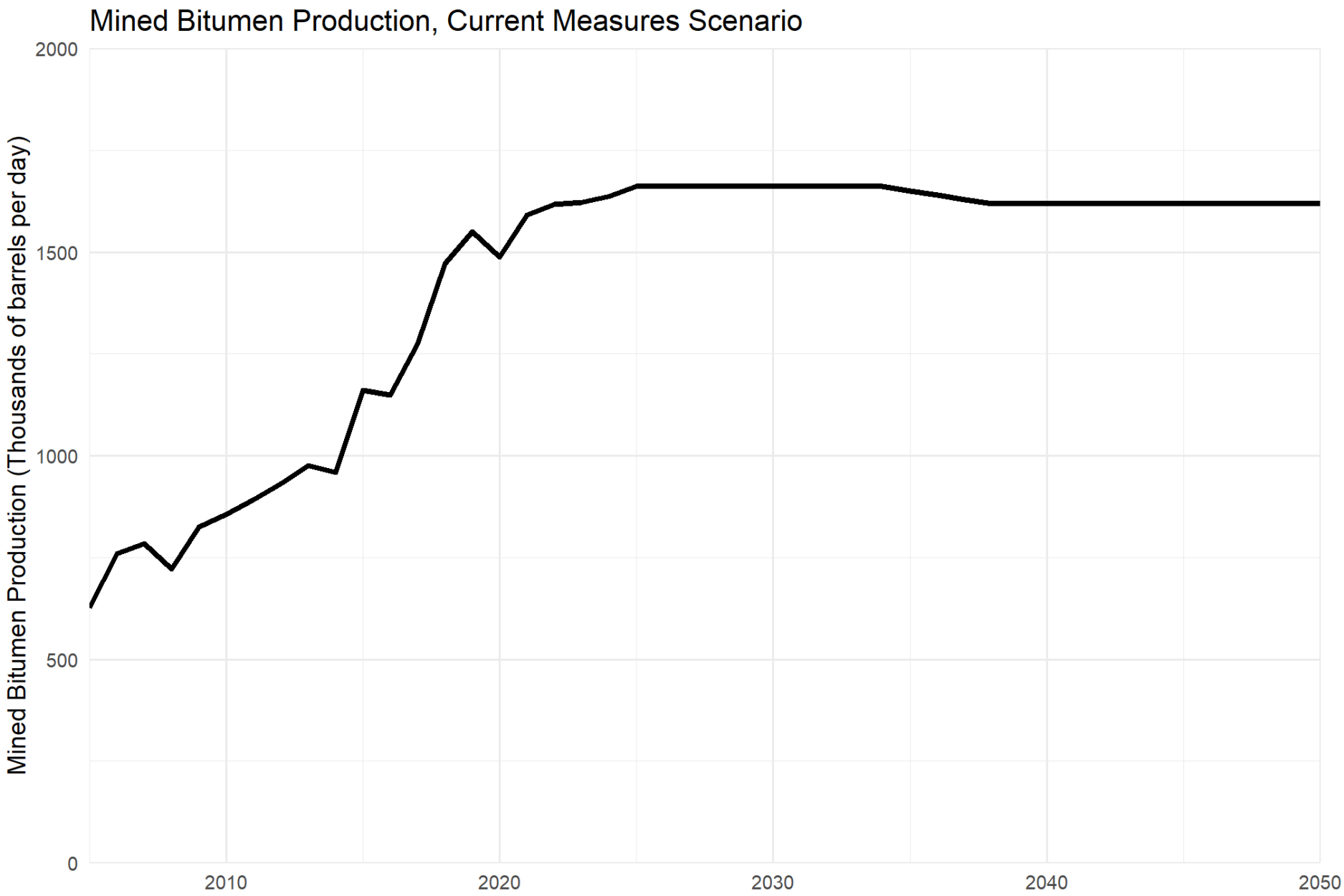
We have a Canadian total and then production by province and territories. Again, we'll want to keep that in mind when we filter.

If you want to do more looking into the data, you can always read it into your R environment and view it, or you can look at the spreadsheet you downloaded.

Two Introductory Graphs

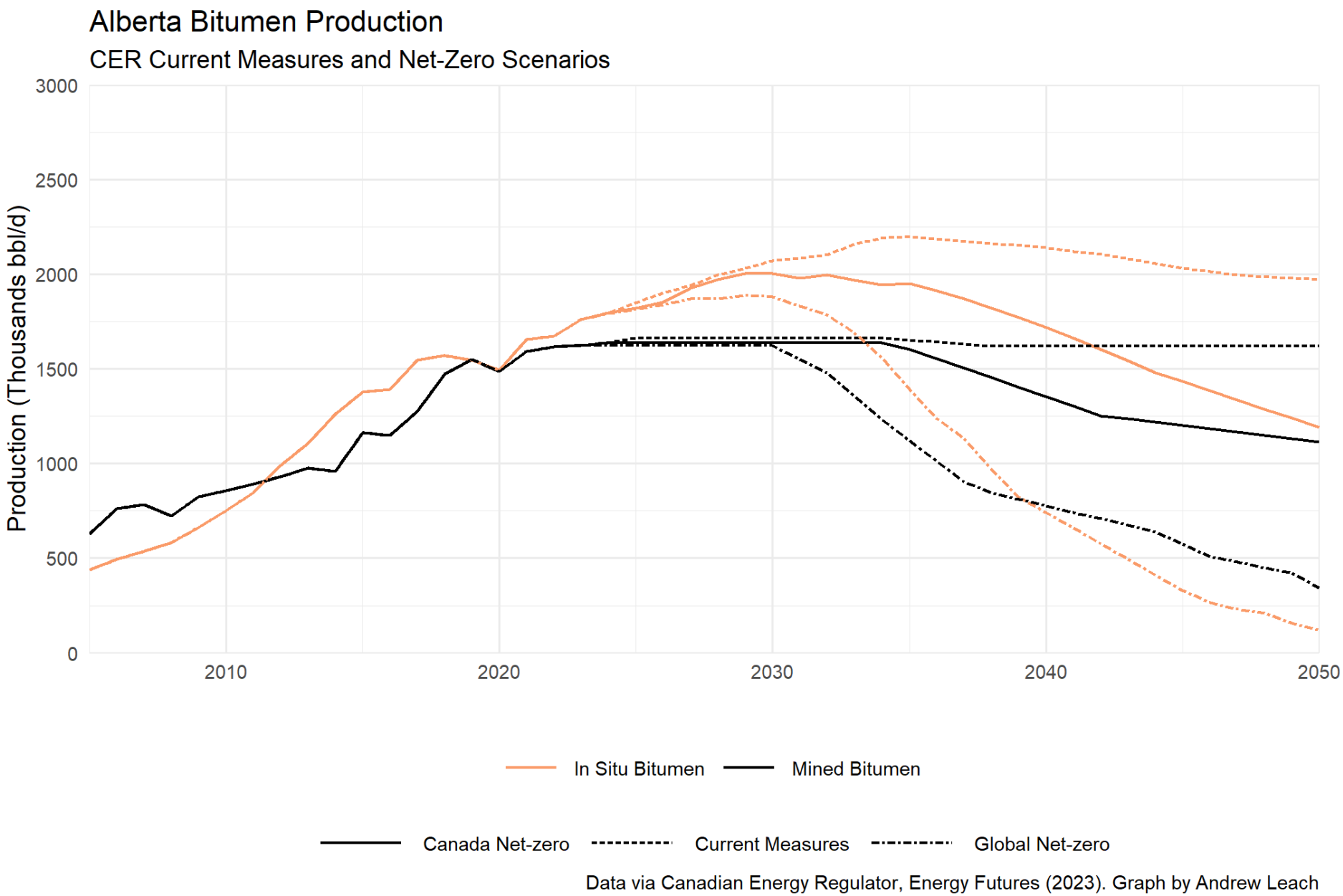
So that you're clear on what we have in front of us, let's use some of the techniques from the last data assignment to visualize the data. In particular, I want to look at Canadian production of mined bitumen using a line graph:

```
#notice, I am not creating a new object, just working from my original data, cer_crude, in memory from the earlier chunk
cer_crude %>%
  #stacked filters, in this case all in one filter command
filter(region=="Canada",unit=="Thousand Barrels per day",variable=="Mined Bitumen",scenario=="Current Measures")%>%
  ggplot()+
  geom_line(aes(year,value,group=variable,color=variable),color="black",linewidth=1.2)+
  scale_x_continuous(breaks=pretty_breaks(), expand=c(0,0))+
  scale_y_continuous(breaks=pretty_breaks(), expand=c(0,0))+
  expand_limits(y=c(0,2000))+
  scale_color_viridis(NULL,option = "B",direction = 1,discrete = T)+
  labs(
      title = "Mined Bitumen Production, Current Measures Scenario",
      y="Mined Bitumen Production (Thousands of barrels per day)",x=NULL)+
    theme_minimal()+
  theme(plot.margin = unit(c(1,1,0.2,1), "cm"))
```


Mined Bitumen Production, Current Measures Scenario
Mined Bitumen Production (Thousands of barrels per day)
2000
1500
1000
500
0
2010
2020
2030
2040
2050

Adding in an extra data series and eliminating a filtered variable (combined bitumen production, for all three scenarios) but with our attention restricted to Alberta, we can create a graph of the impact of different scenarios on bitumen production (a version of the challenge graph from data assignment 1):

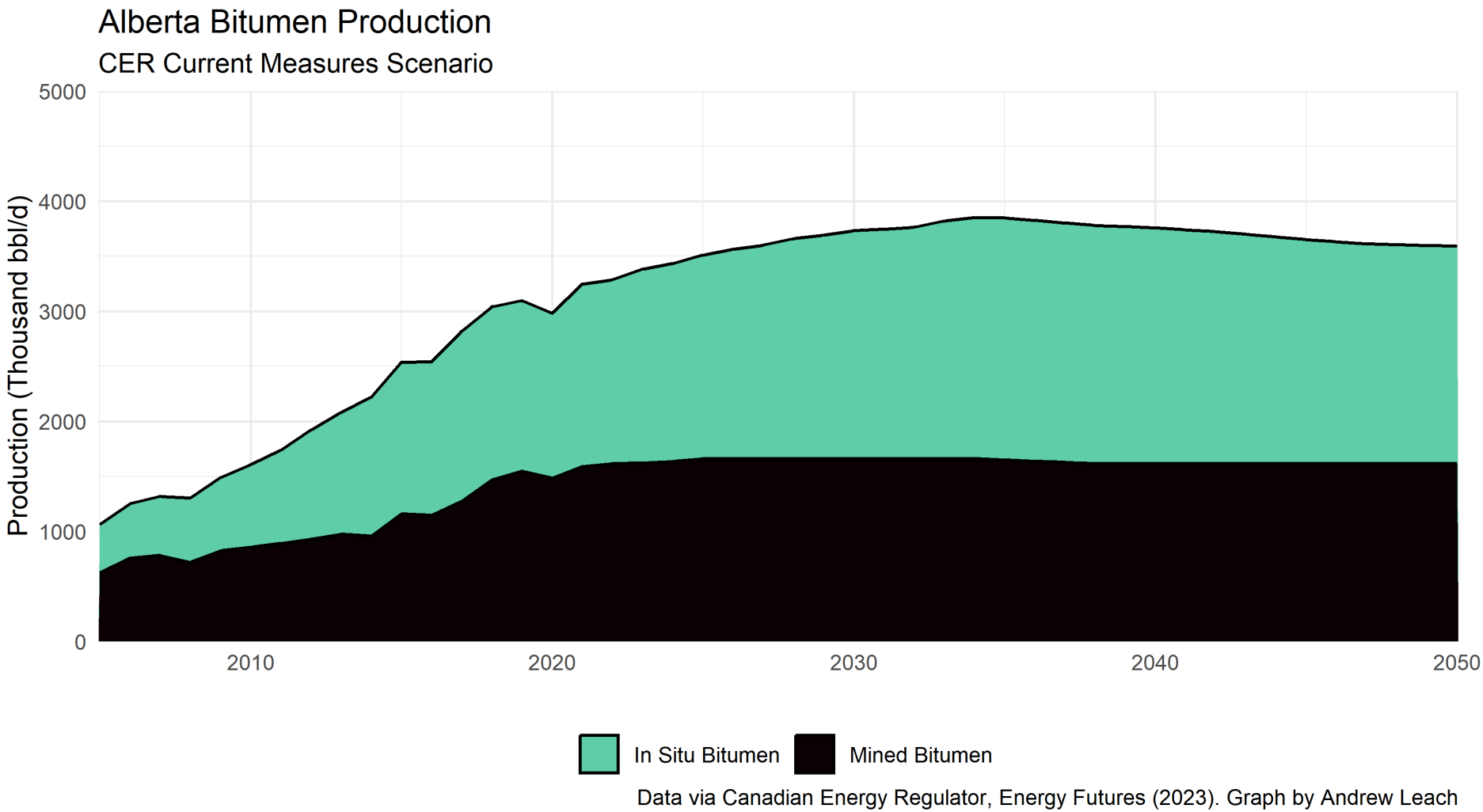
```
bitumen_set<-c("In Situ Bitumen","Mined Bitumen")
cer_crude %>%
  filter(variable %in% bitumen_set)%>%
  filter(unit=="Thousand Barrels per day")%>%
  filter(region %in% c("Alberta"))%>%
  ggplot()+ #make a graph
  geom_line(aes(year,value,group=interaction(variable,scenario),color=variable,linetype=scenario),linewidth=.65)+
  scale_x_continuous(breaks=pretty_breaks(), expand=c(0,0))+
  scale_y_continuous(breaks=pretty_breaks(), expand=c(0,0))+
  theme_minimal()+
  expand_limits(y=c(0,3000))+
  scale_linetype_manual("",values=c("solid","21","3111"))+
  scale_color_viridis("",option = "A",discrete=T,begin = 0,end = .8, direction=-1)+
  #stack the legends at the bottom of the graph
  theme(legend.position = "bottom", legend.box = "vertical",
        legend.margin=margin(t = 0,b=0, unit='cm'))+
  #plot margins
  #t, r, b, l (To remember order, think trouble)
  theme(plot.margin = unit(c(1,1,0.2,1), "cm"))+
  #change what the legends look like
  guides(color = guide_legend(keywidth = unit(1,"cm"),keyheight = unit(1,"cm"),nrow = 1),
    linetype = guide_legend(keywidth = unit(1.6,"cm"),nrow = 1))+
  labs(y="Production (Thousands bbl/d)",x="",
         title="Alberta Bitumen Production",
         subtitle="CER Current Measures and Net-Zero Scenarios",
         caption = "Data via Canadian Energy Regulator, Energy Futures (2023). Graph by Andrew Leach"
         )+
  NULL
```


Alberta Bitumen Production
CER Current Measures and Net-Zero Scenarios
Production (Thousands bbl/d)
3000
2500
2000
1500
1000
500
0
2010
2020
2030
2040
2050
In Situ Bitumen
Mined Bitumen
Canada Net-zero
Current Measures
Global Net-zero
Data via Canadian Energy Regulator, Energy Futures (2023). Graph by Andrew Leach

Area plots

For these data, there are better ways to show the information you want: **using areas and facets**.

A nice area plot for Alberta bitumen production in the Current Measures scenario would look something like this:

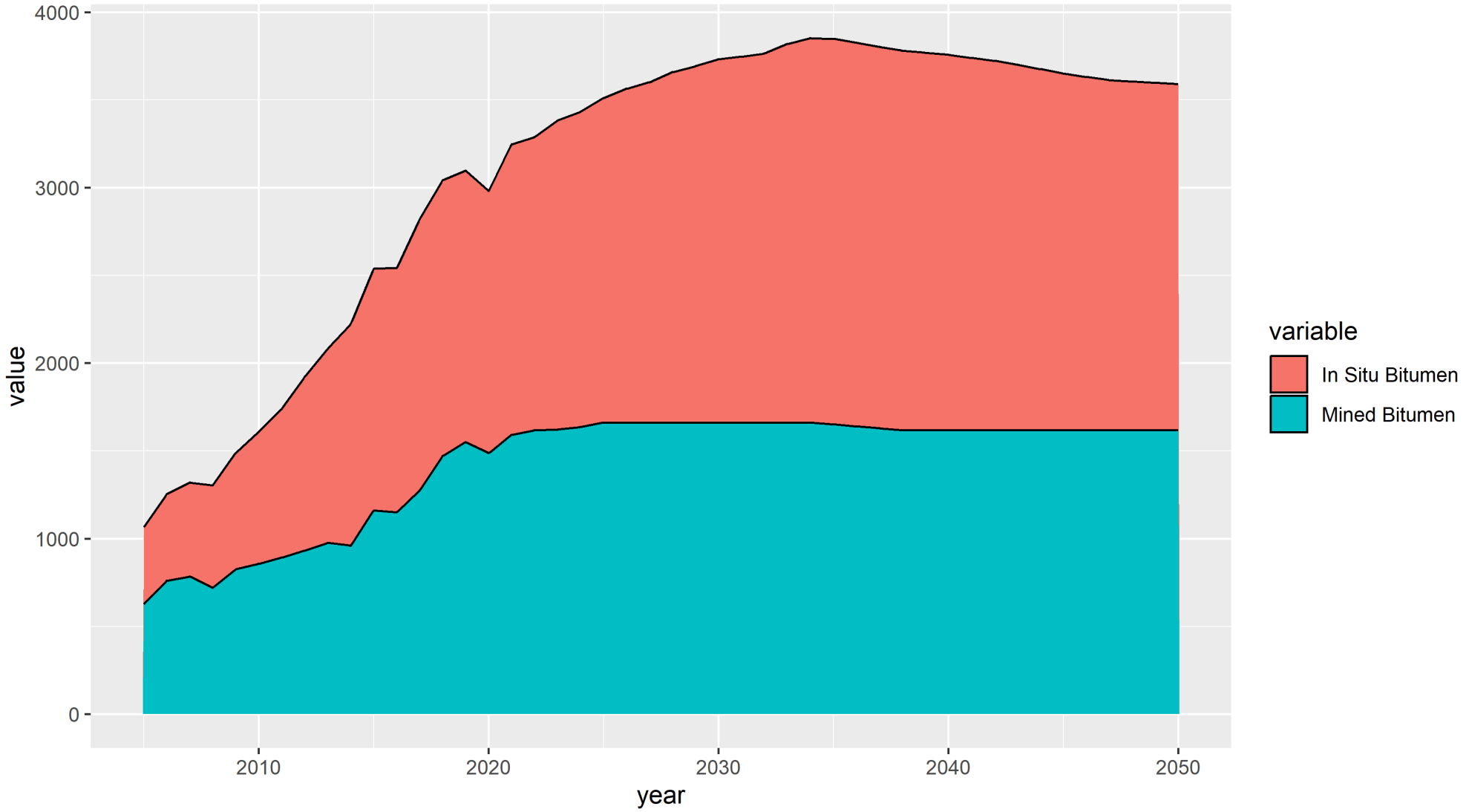


Let's work on how to build it.

The basic elements that we're learning in this are area plots (`geom_area()`) and fill scales (e.g. `scale_fill_viridis()`).

Let's start with a basic graph with no bells and whistles that will look a lot like you're line graphs of the past:

```
bitumen_set<-c("In Situ Bitumen","Mined Bitumen")
#a little different approach here: creating a graph and storing it in memory
area_plot_1<-cer_crude %>%
  filter(variable %in% bitumen_set)%>%
  filter(unit=="Thousand Barrels per day")%>%
  filter(region %in% c("Alberta"))%>%
  filter(scenario == "Current Measures")%>%
  ggplot()+ #make a graph
  geom_area(aes(year,value,group=variable,fill=variable),position = "stack",colour="black",linewidth=0.5)+
  NULL
#and printing it
area_plot_1
```



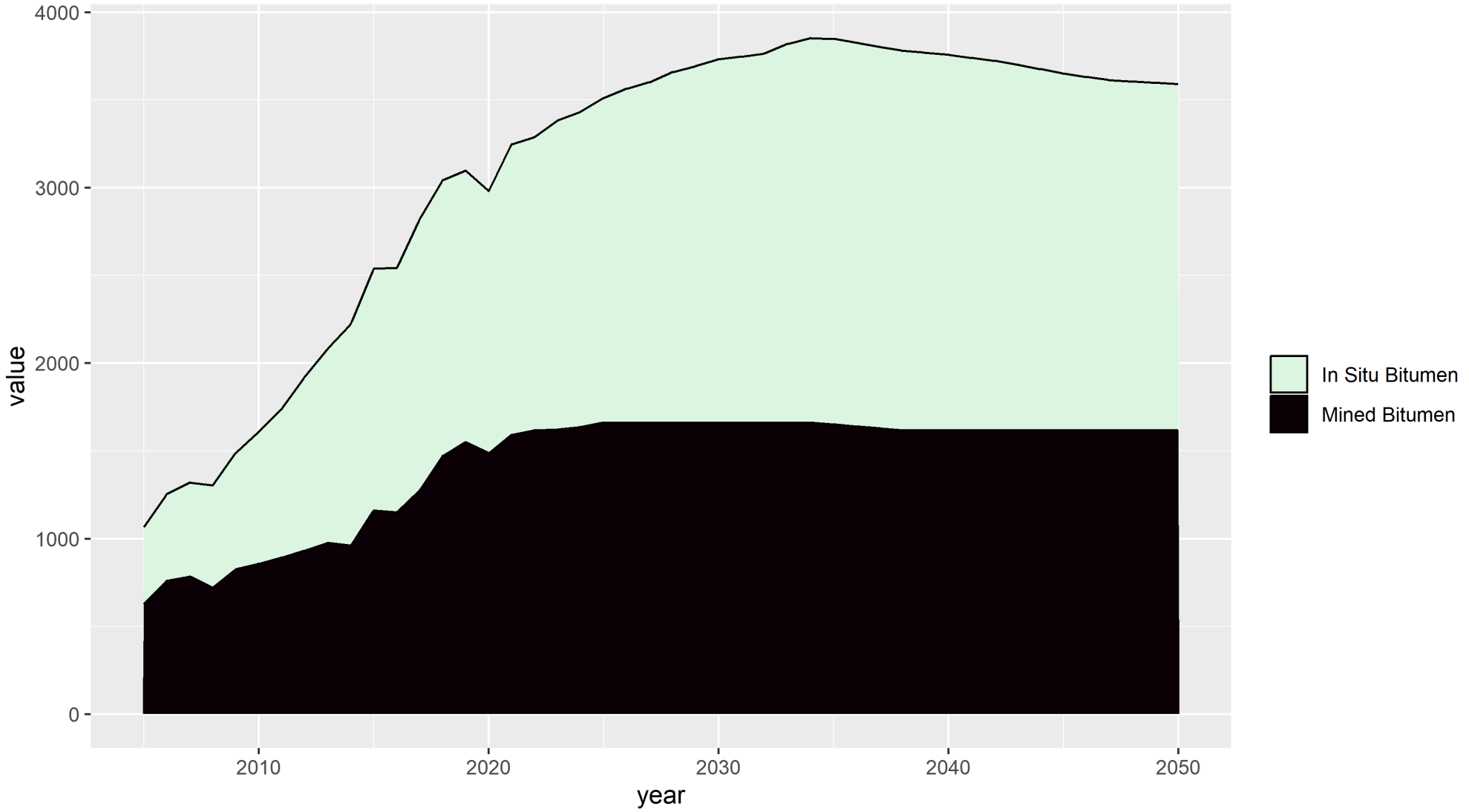
There's one new command in here: `geom_area(aes(year,value,group=variable,fill=variable),position = "stack",colour="black",linewidth=0.65)`, one new aesthetic (`fill`), and one new option (`position = "stack"`). The command as I've written it, I'm telling R to make some area plots (one for each group `variable`), to assign fill colours on the basis of `variable`, and to stack the areas plots. I haven't added any scales or other elements to the plot yet.

In the `geom_area()` command, I have included an option for a fixed `colour="black"` and another one for `linewidth=0.5`. In an area plot, R still generates outside lines, but by default they have no colour or size. I tend to prefer a narrow, black line to split areas, but you may not want that. If you don't, you can take that out and just run `geom_area(aes(year,value,group=variable,fill=variable),position = "stack")`.

I've also put in a bit of a new approach here for you: I've created a basic plot in memory, called `area_plot_1` and then, at the end of the chunk, I've printed the plot by calling its name. You'll see in a second why I am doing it this way - and it's one of the more powerful features of plotting in R.

I'm going to add elements **directly to the plot**. Let's start with a scale.

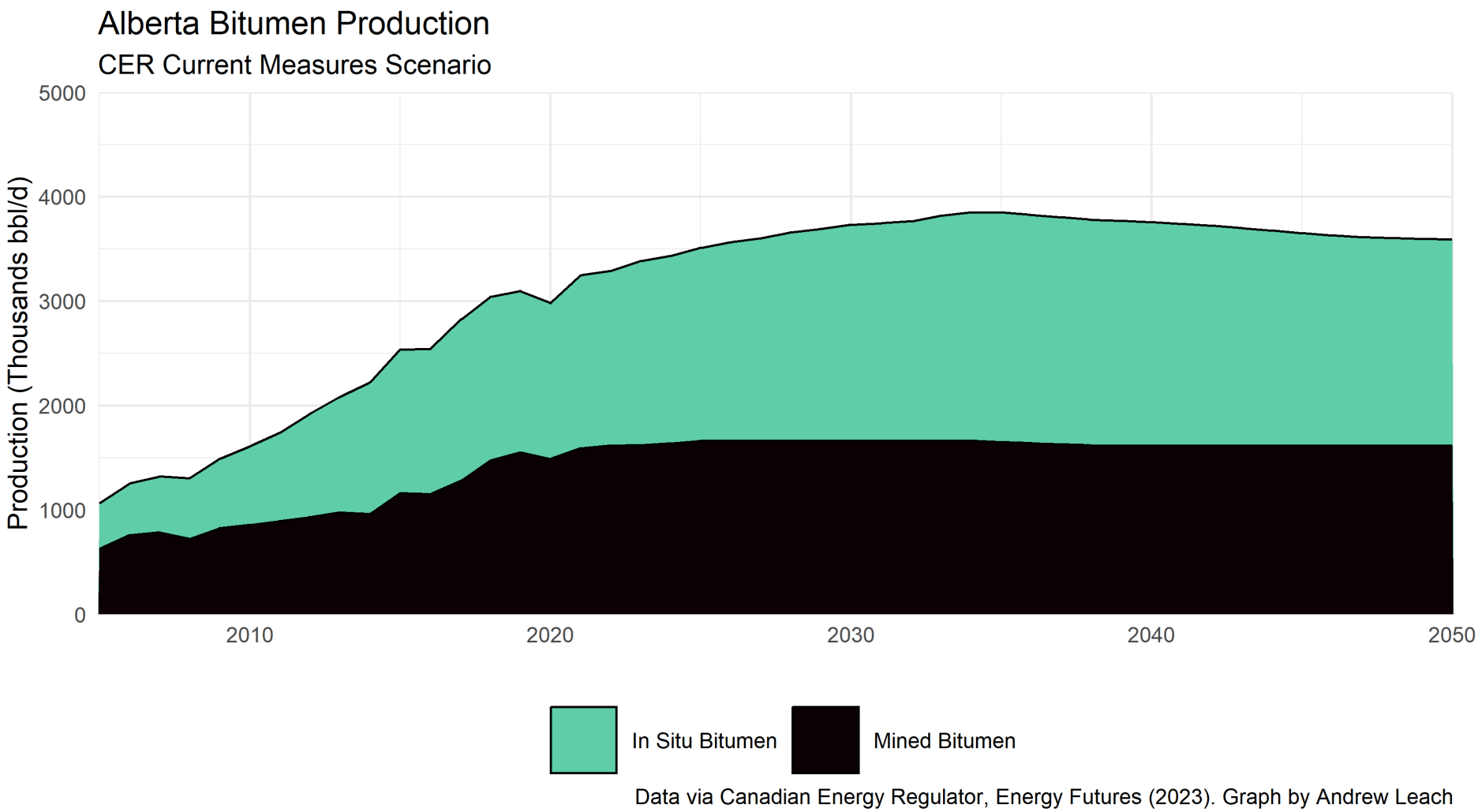
```
area_plot_1+
  scale_fill_viridis("",option="mako",discrete = T,direction=-1)
```



When you add elements to a stored plot object, it's as if you go back to your original graph code and add another line to it. It's very useful. I can add all the elements of my plot from above here now:

```
#note that I am not over-writing the plot - I am, as we did with data, pushing it through a 'machine' of new commands
area_plot_1+
  #add a fill scale
```

```
  scale_fill_viridis("",option = "mako",discrete=T,begin = 0,end = .8, direction=-1)+
  #format X and Y axes
  scale_x_continuous(breaks=pretty_breaks(), expand=c(0,0))+
  scale_y_continuous(breaks=pretty_breaks(), expand=c(0,0))+
  expand_limits(y=c(0,5000))+
  scale_linetype_manual("",values=c("solid","21"))+
  #theme
  theme_minimal()+
  #stack the legends at the bottom of the graph
  theme(legend.position = "bottom", legend.box = "vertical",
        legend.margin=margin(t = 0,b=0, unit='cm'))+
  #plot margins
  #t, r, b, l (To remember order, think trouble)
  theme(plot.margin = unit(c(1,1,0.2,1), "cm"))+
  #change what the legends look like
  guides(fill = guide_legend(keywidth = unit(1,"cm"),keyheight = unit(1,"cm"),nrow = 1))+
  labs(y="Production (Thousands bbl/d)",x="",
         title="Alberta Bitumen Production",
         subtitle="CER Current Measures Scenario",
         caption = "Data via Canadian Energy Regulator, Energy Futures (2023). Graph by Andrew Leach"
         )+
  NULL
```



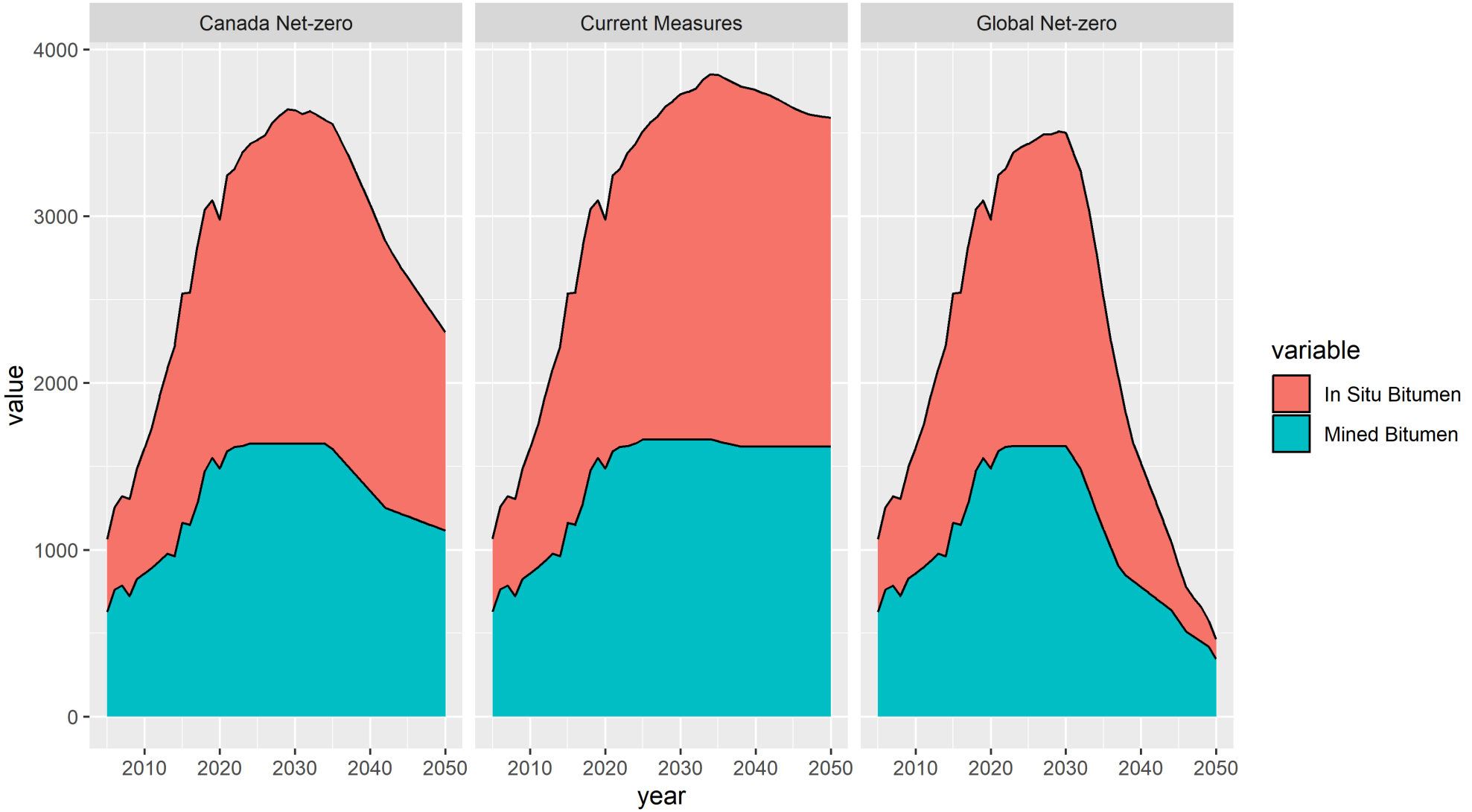
So, this lets you build a graph bit by bit. That's objective #3 for today.

Faceted plots

The last thing I want you to learn is a basic faceted plot. A faceted plot is a series of plots arranged in a table. They make it really easy to do things like compare across provinces or scenarios. For example, the last slide in the oil extraction deck is a faceted plot using these data.

Let's start again with a basic plot: a faceted version of the area plot from above. In this case, I am going to leave data from both scenarios in when I make my plot, but then I'm going to tell R to `facet_grid` by scenario. The `~` is important.

```
bitumen_set<-c("In Situ Bitumen","Mined Bitumen")
#a little different approach here: creating a graph and storing it in memory
faceted_plot_1<-cer_crude %>%
  filter(variable %in% bitumen_set)%>%
  filter(unit=="Thousand Barrels per day")%>%
  filter(region %in% c("Alberta"))%>%
  ggplot()+ #make a graph
  geom_area(aes(year,value,group=variable,fill=variable),position = "stack",colour="black",linewidth=0.5)+
  #one new line
  facet_wrap(~scenario)+
  NULL
#and printing it
faceted_plot_1
```

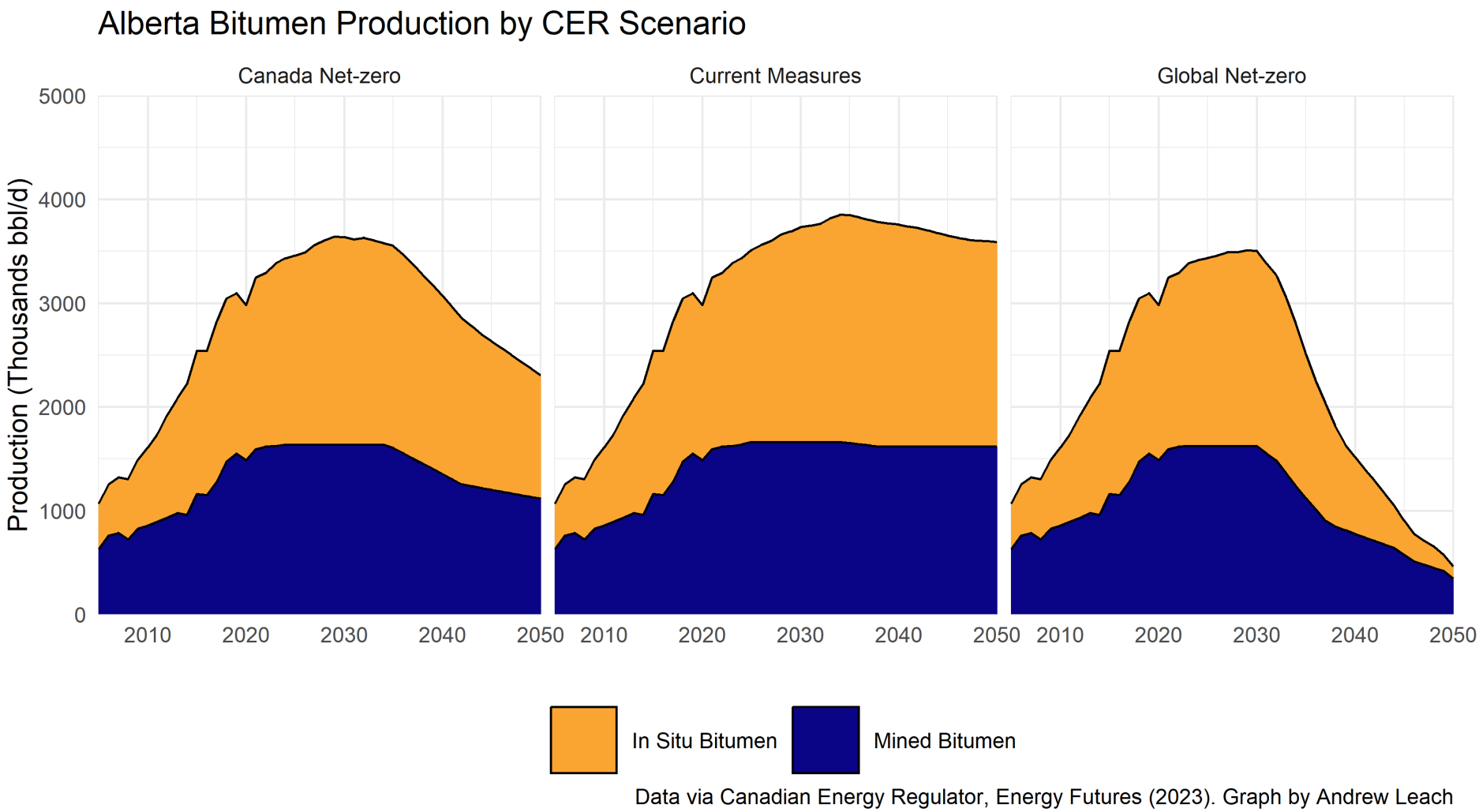


Well, that's pretty cool. And the only new command is the `facet_wrap(~scenario)` line, telling it to make a new plot for each scenario.

As we did before, we can apply the same formatting to the faceted plot just by adding it to the saved object:

```
faceted_plot_1+
  #add a fill scale
  scale_fill_viridis("",option = "C",discrete=T,begin = 0,end = .8, direction=-1)+
```

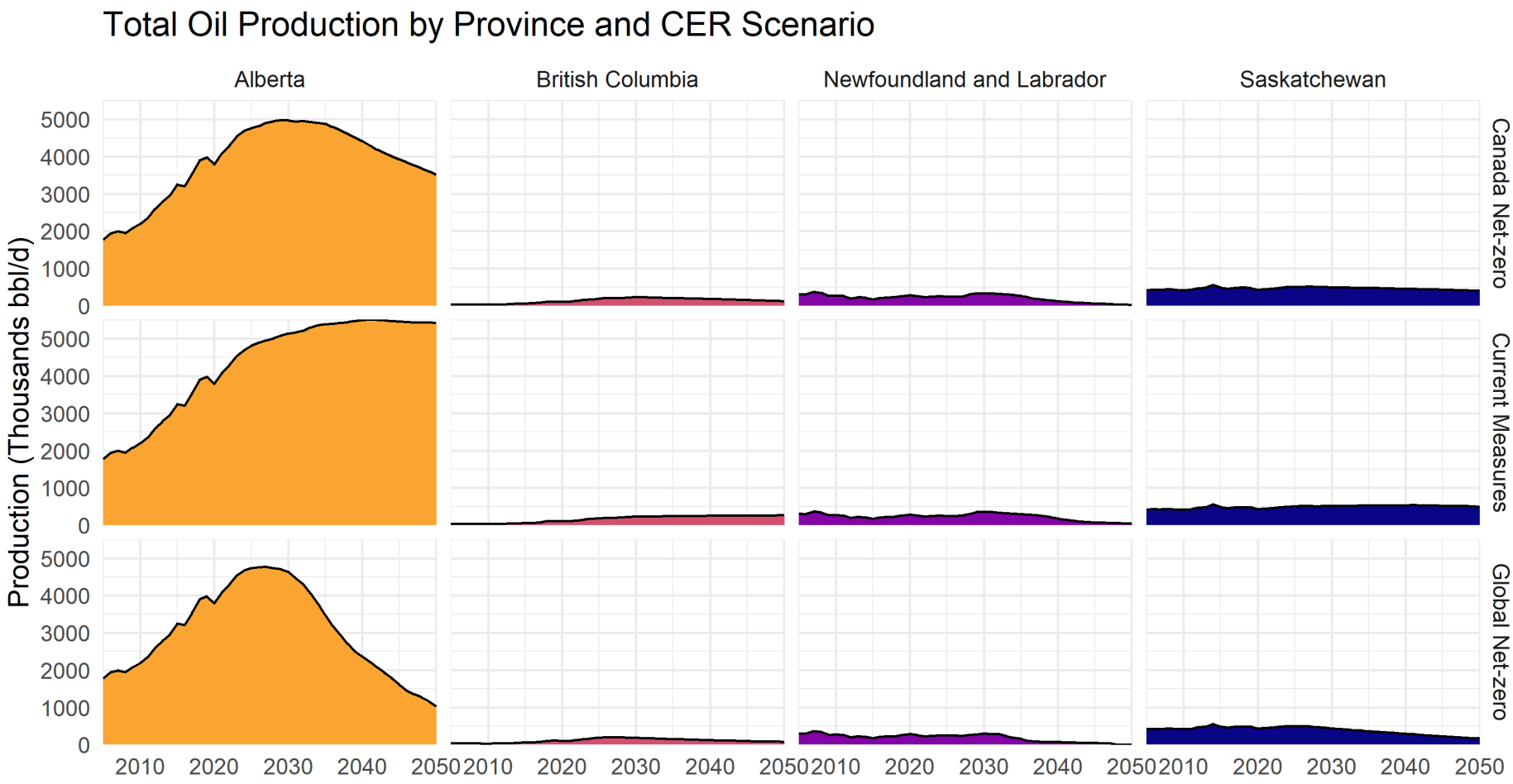
```
#format X and Y axes
scale_x_continuous(breaks=pretty_breaks(), expand=c(0,0))+
scale_y_continuous(breaks=pretty_breaks(), expand=c(0,0))+
expand_limits(y=c(0,5000))+
scale_linetype_manual("",values=c("solid","21"))+
#theme
theme_minimal()+
#stack the legends at the bottom of the graph
theme(legend.position = "bottom", legend.box = "vertical",
      legend.margin=margin(t = 0,b=0, unit='cm'))+
#plot margins
#t, r, b, l (To remember order, think trouble)
theme(plot.margin = unit(c(1,1,0.2,1), "cm"))+
#change what the legends look like
guides(fill = guide_legend(keywidth = unit(1,"cm"),keyheight = unit(1,"cm"),nrow = 1))+
labs(y="Production (Thousands bbl/d)",x="",
       title="Alberta Bitumen Production by CER Scenario",
       caption = "Data via Canadian Energy Regulator, Energy Futures (2023). Graph by Andrew Leach"
       )+
NULL
```



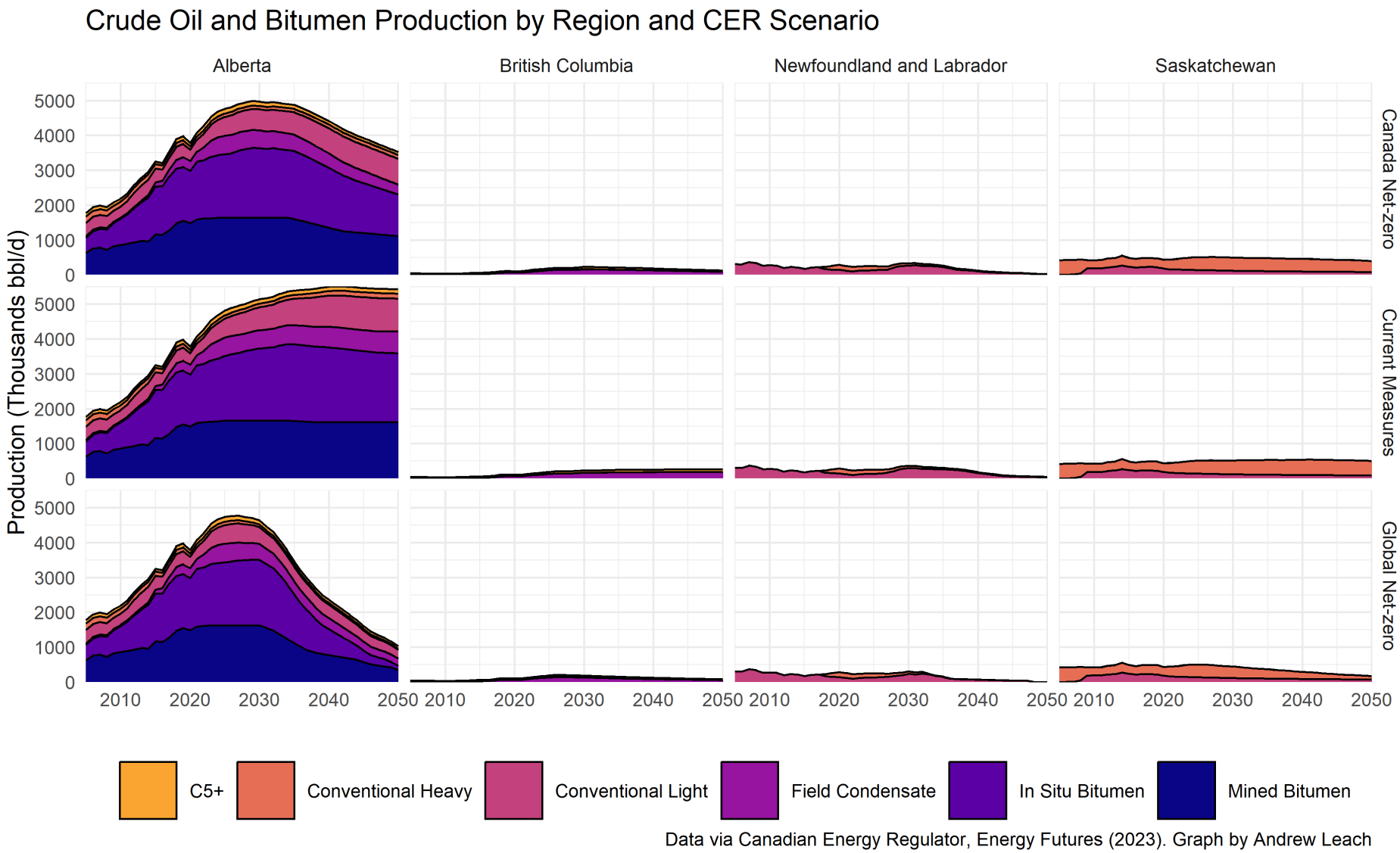
There is another option, `facet_grid()`, which is more powerful but also more complex to use. `facet_grid()` allows you to compare across two levels. For example, I could use `facet_grid(rows = vars(scenario),cols = vars(region))` to compare across regions and scenarios:

```
oil_regions<-c("British Columbia","Alberta","Saskatchewan", "Newfoundland and Labrador")
```

```
cer_crude %>%
  filter(variable == "Total")%>%
  filter(unit=="Thousand Barrels per day")%>%
  filter(region %in% oil_regions)%>%
  ggplot()+ #make a graph
  geom_area(aes(year,value,group=variable,fill=region),position = "stack",colour="black",linewidth=0.5)+
  #one new line
  facet_grid(rows = vars(scenario),cols = vars(region))+
 #fill options
  scale_fill_viridis("",option = "C",discrete=T,begin = 0,end = .8, direction=-1)+
  #format X and Y axes
  scale_x_continuous(breaks=pretty_breaks(), expand=c(0,0))+
  scale_y_continuous(breaks=pretty_breaks(), expand=c(0,0))+
  expand_limits(y=c(0,5000))+
  scale_linetype_manual("",values=c("solid","21"))+
  #theme
  theme_minimal()+
  #stack the legends at the bottom of the graph
  theme(legend.position = "none")+
  #t, r, b, l (To remember order, think trouble)
  theme(plot.margin = unit(c(1,1,0.2,1), "cm"))+
  #change what the legends look like
  labs(y="Production (Thousands bbl/d)",x="",
         title="Total Oil Production by Province and CER Scenario",
         caption = "Data via Canadian Energy Regulator, Energy Futures (2023). Graph by Andrew Leach"
         )+
  NULL
```



For this week's exercise, I'd like to see you create an RMD that replicates all of these plots. And then, if you want to test your skills, do a stacked area plot by region and scenario using the code from the last graph, but do it by individual crude oil type. Something like this:



Hint: look at the `geom_area()` and ask how colours are assigned to filled areas.

Next week, we'll learn about factors and how to deal with getting these crude streams in logical order and the regions in geographic order.

Made with R and Quarto
View the source at GitHub