It simultaneously:

1. Threw a bit of *“shade”* on the ToS for Axios (well done Bob)
2. Showed how to use EtherCalc as a data entry tool
3. And, most importantly to me, showed how to make great use of a slopegraph

I wanted to look at Bob’s post in detail after very quickly agreeing with his  
premise that it was a much better choice than a “dumbbell chart”. So this post  
is about what I learned and the adjustments I made to my own function.

**This post assumes that you’ve read the earlier posts.**

Let’s quickly recreate the dataset Bob created in EtherCalc keeping it simple  
using the str function and choosing to make it a dataframe not a tibble.

library(tidyverse)

library(CGPfunctions)

thedata <- structure(list(

topic = structure(c(6L, 3L, 5L, 4L, 11L, 13L,

2L, 8L, 9L, 12L, 7L, 1L, 14L, 10L),

.Label = c("Arts & entertainment",

"Business", "Climate change", "Economics", "Education", "Health care",

"Immigration", "National Security", "Politics", "Religion", "Science",

"Sports", "Technology", "U.S. foreign policy"),

class = "factor"),

actually\_read = c(7L, 5L, 11L, 6L, 10L, 14L, 13L, 1L, 2L, 3L, 4L, 8L, 9L, 12L),

say\_want\_covered = c(1L, 2L, 3L, 4L, 7L, 8L, 11L, 5L, 10L, 14L, 6L, 13L, 9L, 12L)),

class = "data.frame", row.names = c(NA, -14L))

thedata

## topic actually\_read say\_want\_covered

## 1 Health care 7 1

## 2 Climate change 5 2

## 3 Education 11 3

## 4 Economics 6 4

## 5 Science 10 7

## 6 Technology 14 8

## 7 Business 13 11

## 8 National Security 1 5

## 9 Politics 2 10

## 10 Sports 3 14

## 11 Immigration 4 6

## 12 Arts & entertainment 8 13

## 13 U.S. foreign policy 9 9

## 14 Religion 12 12

**Making slopegraphs easy**

When you look at Bob’s post there’s actually a lot of code in there to make a  
very nice graphic. Being extraordinarily lazy I wrote my function to get a  
slopegraph with the least amount of work possible. The first step, which is  
unavoidable if you want to make use of newggslopegraph, though is to reshape  
the data into a “longer” format. We’ll use reshape2::melt and keep the topic  
column but collapse the other two columns into a factor called Saydo and put the  
actual “rank” into a column called Rank. Since “actually\_read” and  
“say\_want\_covered” are now factor levels instead of column names we can use  
forcats::fct\_recode to make them much nicer built in labels when we make our  
plot. Voila a new dataframe called temp.

temp <- reshape2::melt(data = thedata,

id = "topic",

[variable.name](http://variable.name) = "Saydo",

[value.name](http://value.name) = "Rank")

temp$Saydo <- forcats::fct\_recode(temp$Saydo,

"Actually read" = "actually\_read",

"Say they want" = "say\_want\_covered")

temp

## topic Saydo Rank

## 1 Health care Actually read 7

## 2 Climate change Actually read 5

## 3 Education Actually read 11

## 4 Economics Actually read 6

## 5 Science Actually read 10

## 6 Technology Actually read 14

## 7 Business Actually read 13

## 8 National Security Actually read 1

## 9 Politics Actually read 2

## 10 Sports Actually read 3

## 11 Immigration Actually read 4

## 12 Arts & entertainment Actually read 8

## 13 U.S. foreign policy Actually read 9

## 14 Religion Actually read 12

## 15 Health care Say they want 1

## 16 Climate change Say they want 2

## 17 Education Say they want 3

## 18 Economics Say they want 4

## 19 Science Say they want 7

## 20 Technology Say they want 8

## 21 Business Say they want 11

## 22 National Security Say they want 5

## 23 Politics Say they want 10

## 24 Sports Say they want 14

## 25 Immigration Say they want 6

## 26 Arts & entertainment Say they want 13

## 27 U.S. foreign policy Say they want 9

## 28 Religion Say they want 12

Once we get the data in the right shape I tried to make newggslopegraph as  
simple and intuitive as possible. I love working with ggplot but I will admit  
it can get quite complex. So to create the default plot all we need to do is:

newggslopegraph(dataframe = temp,

Times = Saydo,

Measurement = Rank,

Grouping = topic)

##

## Converting 'Saydo' to an ordered factor

That was pretty painless wasn’t it? But clearly there’s a lot of room for  
tweaking! Let’s make it better!

**Tweaking**

Whole books can and have been written just on the issue of graphic design so I’m  
not going to try and summarize it all in one little blog post. I will however,  
for the impatient reader, immediately take care of a few key things:

1. **Titles, subtitles and captions are important!** Don’t ignore them or give  
   them short change. You’ll notice that since we didn’t initially specify them,  
   placeholders appear. That’s to be shameless about making you think about them  
   even if you eventually decide to turn them “off”
2. The default is that every line is it’s own color. That’s seldom a good choice  
   for telling a story unless the number of topics (a.k.a. Groups) is very  
   small. For now let’s make them all “black” and come back to this in a bit.
3. By default Measurement is treated as a real number so the highest values  
   are on the top of the graph. Makes more sense here to reverse the scale and  
   put the highest ranked “1” at he top. ReverseYAxis = TRUE. If we needed or  
   wanted to ReverseXAxis = TRUE might be useful.

Our second attempt looks like this:

newggslopegraph(dataframe = temp,

Times = Saydo,

Measurement = Rank,

Grouping = topic,

ReverseYAxis = TRUE,

Title = "14 Topics Ranked by What Americans Read vs Want Covered",

SubTitle = "'Read' rank from Parse.ly May 2019 data.\n'Want covered' rank from Axios/SurveyMonkey poll conducted May 17-20, 2019",

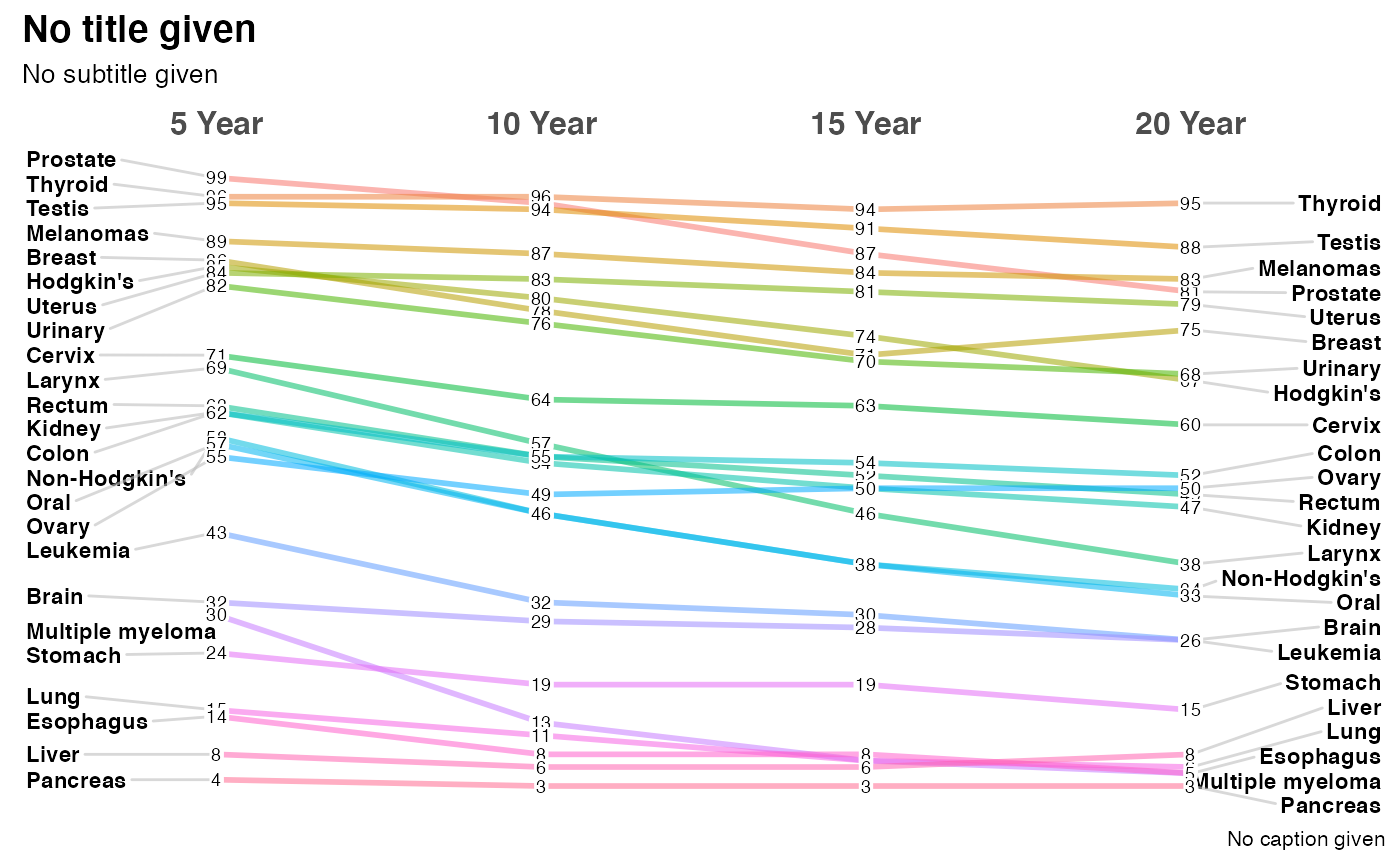
Caption = "Source: Axios \nMakeover ",

LineColor = "black"

)

# the minimum command to generate a plot

newggslopegraph(newcancer, Year, Survival, Type)



# adding a title which is always recommended

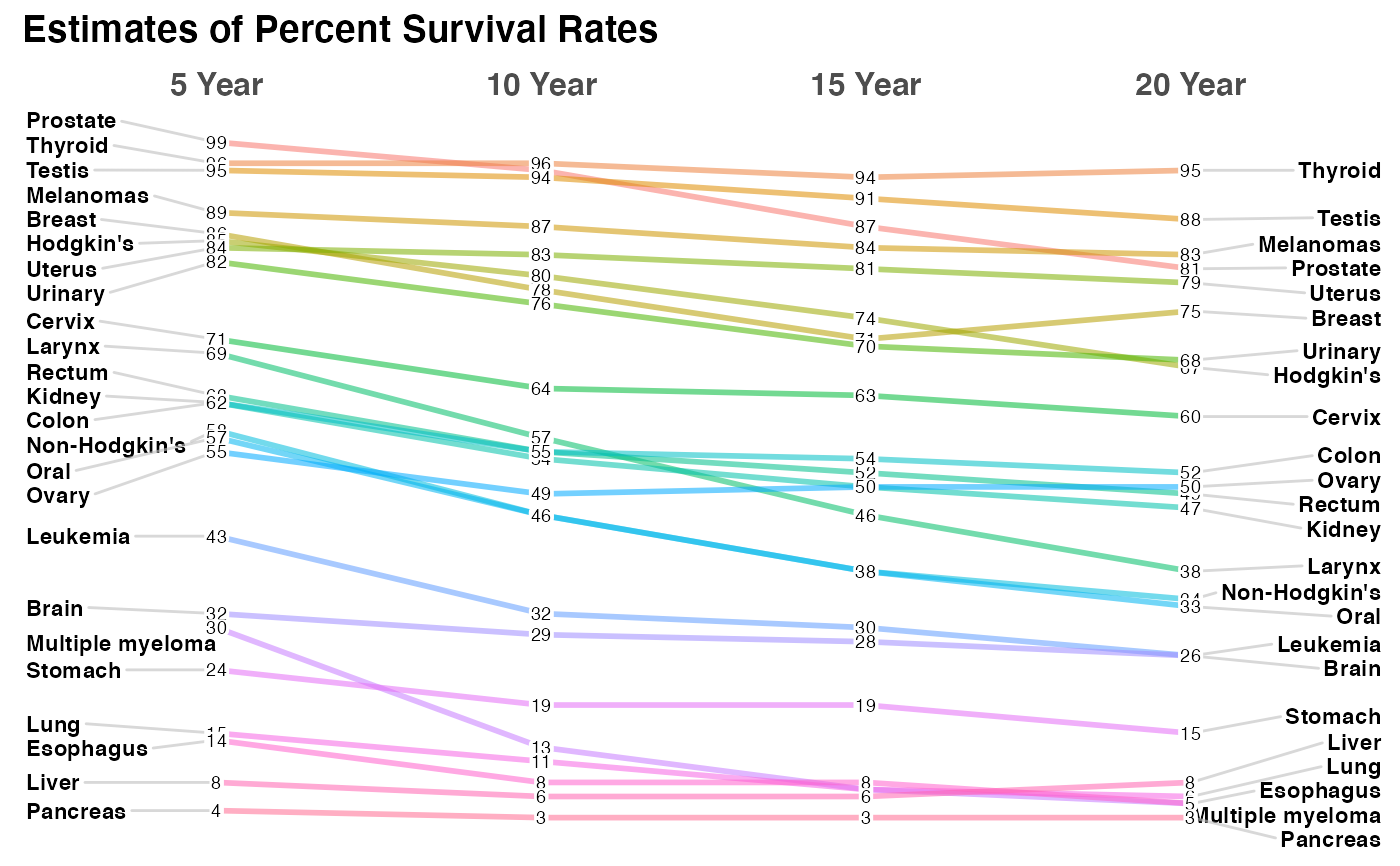
newggslopegraph(newcancer, Year, Survival, Type,

Title = "Estimates of Percent Survival Rates",

SubTitle = NULL,

Caption = NULL

)



# simple formatting changes

newggslopegraph(newcancer, Year, Survival, Type,

Title = "Estimates of Percent Survival Rates",

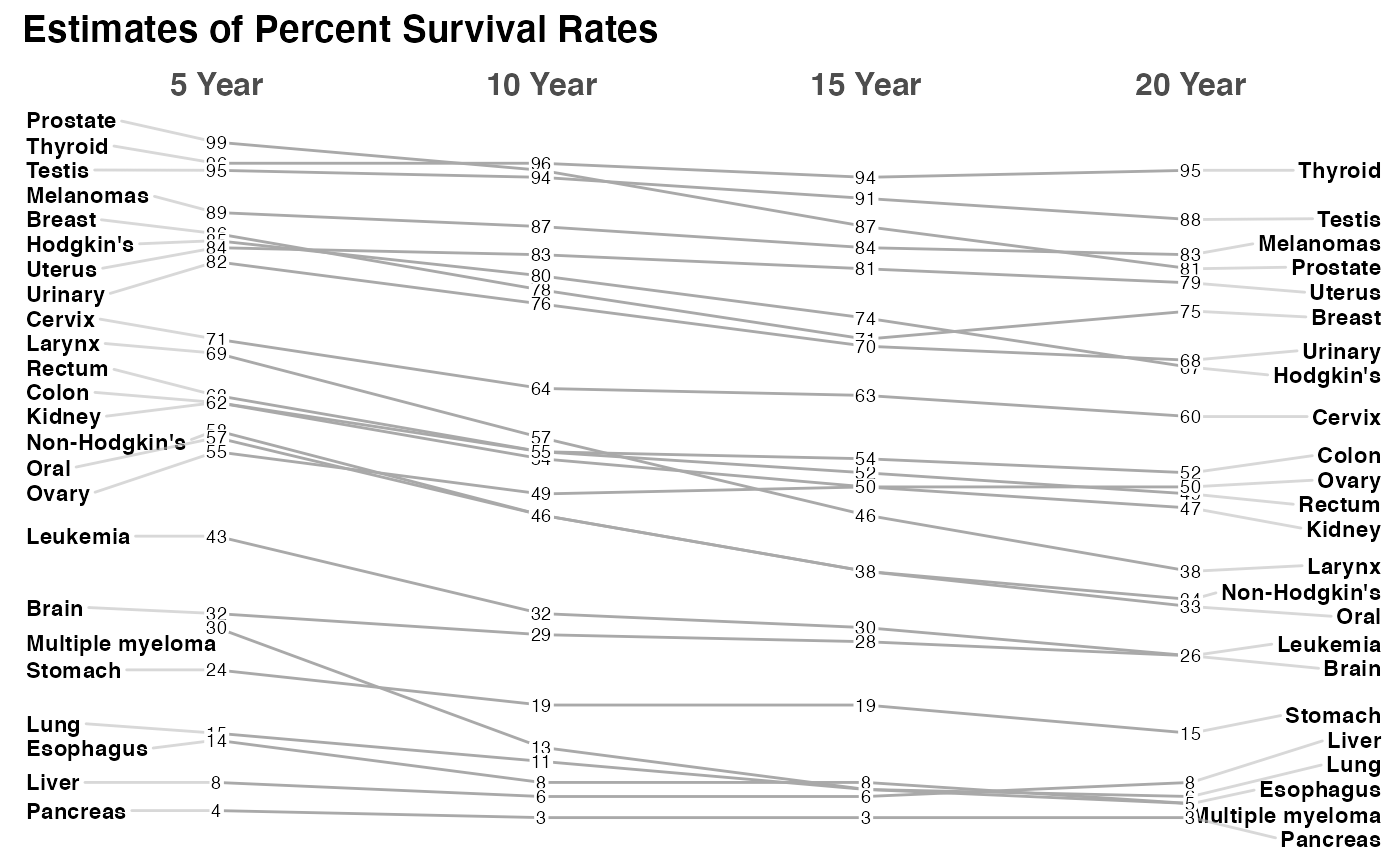
LineColor = "darkgray",

LineThickness = .5,

SubTitle = NULL,

Caption = NULL

)



# complex formatting with recycling and wider labels see vignette for more examples

newggslopegraph(newcancer, Year, Survival, Type,

Title = "Estimates of Percent Survival Rates",

SubTitle = "Based on: Edward Tufte, Beautiful Evidence, 174, 176.",

Caption = "https://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg\_id=0003nk",

LineColor = [c](https://rdrr.io/r/base/c.html)("black", "red", "grey"),

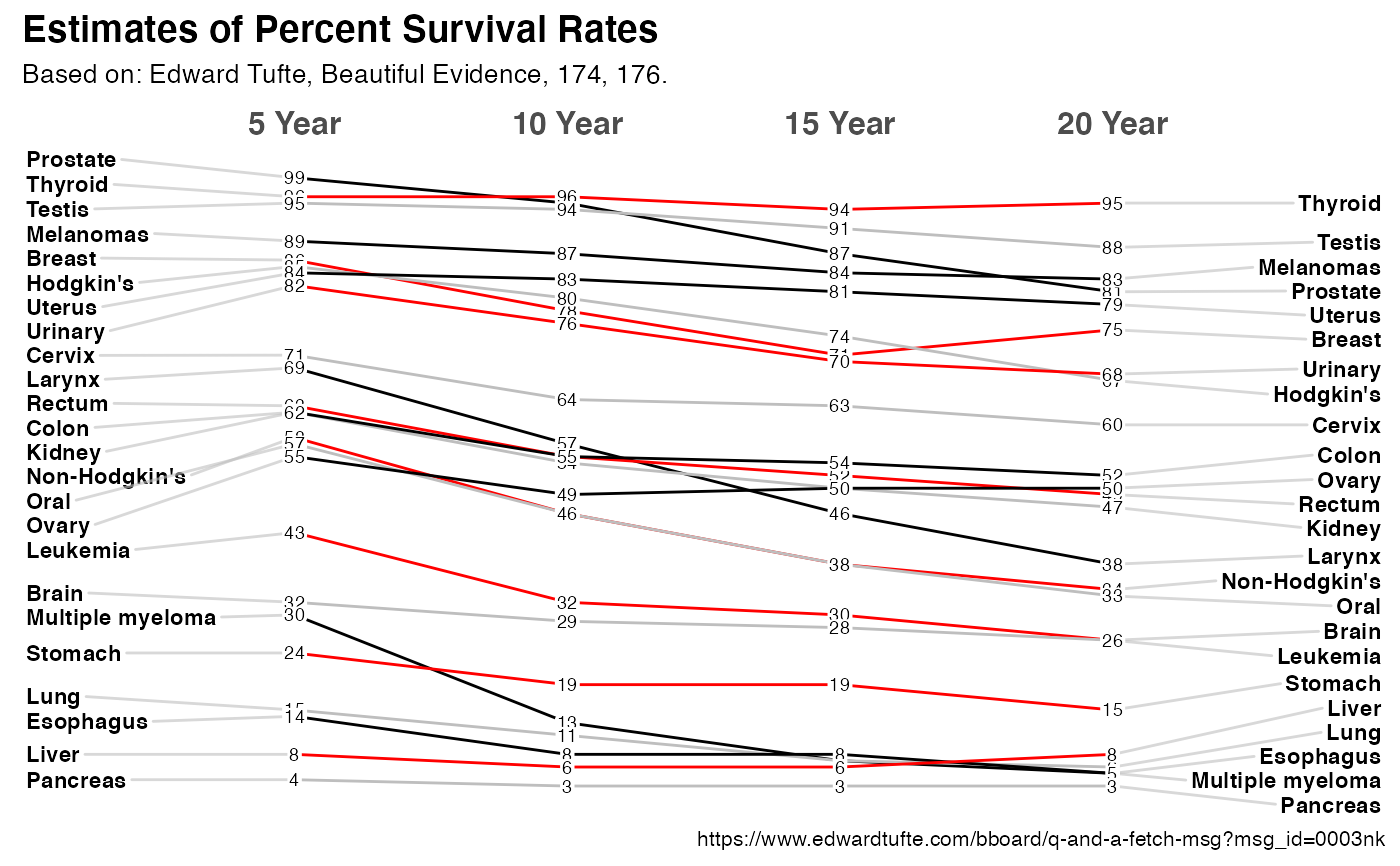
LineThickness = .5,

WiderLabels = TRUE

)

#>

#> **You gave me 3 colors I'm recycling colors because you have 24 Types**



# not a great example but demonstrating functionality

newgdp$rGDP <- [round](https://rdrr.io/r/base/Round.html)(newgdp$GDP)

newggslopegraph(newgdp,

Year,

rGDP,

Country,

LineColor = [c](https://rdrr.io/r/base/c.html)([rep](https://rdrr.io/r/base/rep.html)("grey", 3), "red", [rep](https://rdrr.io/r/base/rep.html)("grey", 11)),

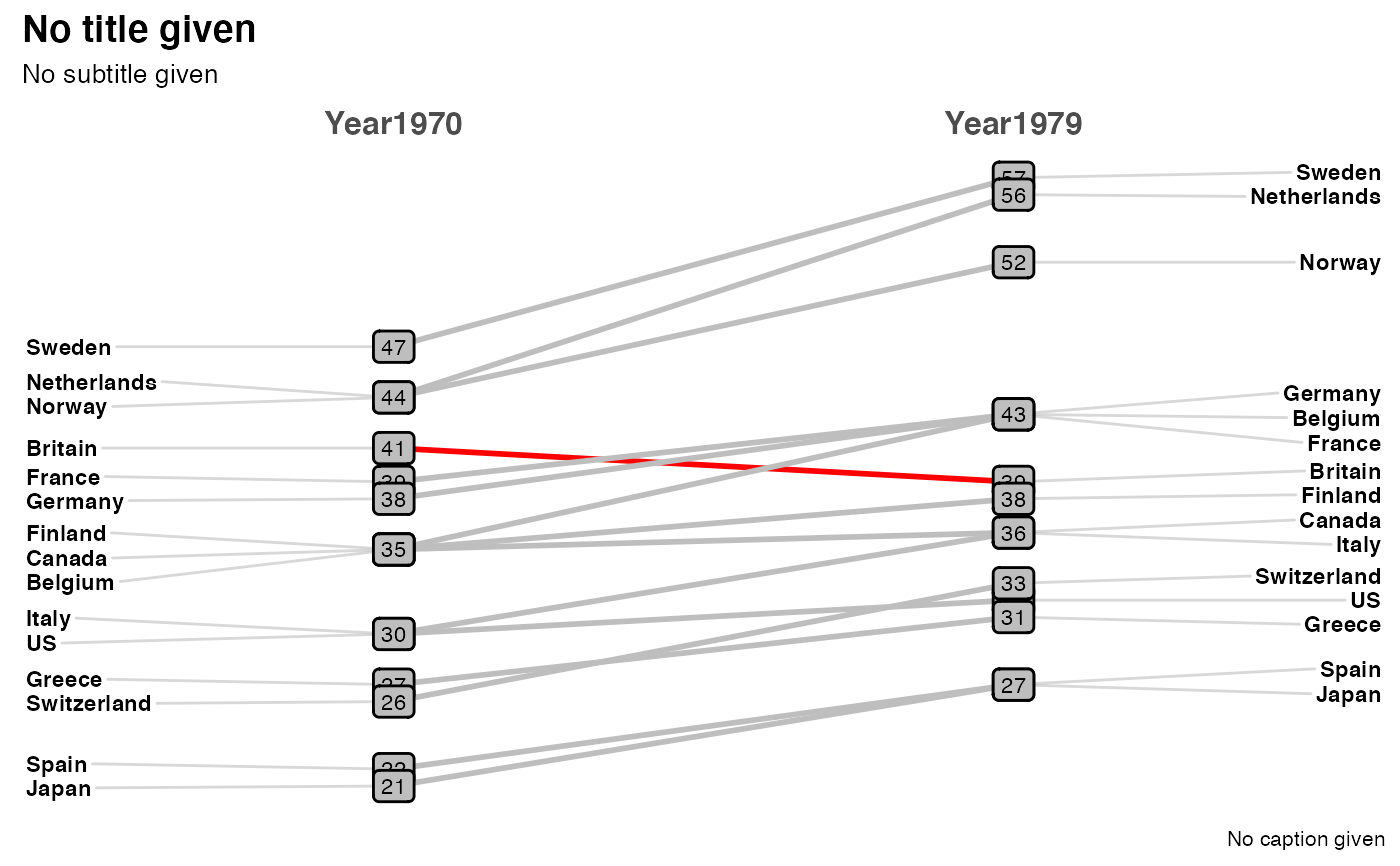
DataTextSize = 3,

DataLabelFillColor = "gray",

DataLabelPadding = .2,

DataLabelLineSize = .5

)



Alright, that’s looking a little bit better for basic layout. But it doesn’t yet  
tell the reader a story and focus their attention on the message we want to  
convey. To be honest I’m not a huge fan of adding a lot of annotations to a plot  
so let’s first try to catch the readers attention by using color selectively.

**Emphasizing the “slope” in slopegraph**

As the name implies slopegraphs get the reader to attend to relative differences  
in slope, right now our choice of “black” as the only color is marginally better  
than our original multicolor mess but still falls far short of conveying a  
message. The LineColor parameter is quite flexible. The default is suitable  
for a small number of topics, a single color can be the right choice on occasion,  
but we can also pass it a character vector of colors that is as customized as we  
like. For example LineColor = c("black", "red") would recycle the colors red  
and black to create an alternating pattern. We could even build a named list  
that associates a color to each of the topic areas if we desired.

But right now, that is too much effort and I’d like to  
handle this by algorithm not by manually entry.

As a start point let’s assume we’d like to get the reader to focus on  
understanding which topics increase in rank, decrease in rank or stay the same.  
We’ll color increase as black, decreases as red and things that remain level as  
light gray. We can accomplish that through a series of pipes and dplyr  
verbs.

colorvect <- temp %>% group\_by(topic) %>%

summarise(difference = diff(Rank)) %>%

mutate(whatcolor = case\_when(

difference == 0 ~ "light gray",

difference > 0 ~ "red",

difference < 0 ~ "black"

)) %>%

select(topic, whatcolor) %>%

tibble::deframe()

colorvect

## Arts & entertainment Business Climate change

## "red" "black" "black"

## Economics Education Health care

## "black" "black" "black"

## Immigration National Security Politics

## "red" "red" "red"

## Religion Science Sports

## "light gray" "black" "red"

## Technology U.S. foreign policy

## "black" "light gray"

Each topic now has a color assigned, and it’s trivial to pass our color vector  
to newggslopegraph. While we’re at it we can showcase some of the other  
formatting options, like changing font sizes for the labels. DataLabelPadding  
is important if you are likely to have datapoints close together

but in this case we can be more generous since  
ranks won’t overlap.

newggslopegraph(dataframe = temp,

Times = Saydo,

Measurement = Rank,

Grouping = topic,

ReverseYAxis = TRUE,

DataTextSize = 3.5,

YTextSize = 4,

XTextSize = 16,

DataLabelPadding = .2,

Title = "Topic Rankings Compared Between\nWhat Americans Actually Read vs Want Covered",

SubTitle = "'Actually Read' rank from Parse.ly May 2019 data.\n'Want covered' rank from Axios/SurveyMonkey poll conducted May 17-20, 2019",

Caption = "Source: Axios \nMakeover by @hrbrmstr",

LineColor = colorvect

)

Very nice looking, but I think it is still too crowded with colors. Let’s adjust  
our coloring to highlight only the larger rank differences. It’s a matter of  
personal taste but easy to adjust our little script and test, rinse and repeat  
until we’re happy. Let’s adjust so that changes of greater than 4 or less than 4  
are highlighted and the rest are gray.

colorvect <- temp %>% group\_by(topic) %>%

summarise(difference = diff(Rank)) %>%

mutate(whatcolor = case\_when(

difference >= 4 ~ "red",

difference <= -4 ~ "black",

TRUE ~ "light gray"

)) %>%

select(topic, whatcolor) %>%

tibble::deframe()

Then we can run the same lines into newggslopegraph.

newggslopegraph(dataframe = temp,

Times = Saydo,

Measurement = Rank,

Grouping = topic,

ReverseYAxis = TRUE,

DataTextSize = 3.5,

YTextSize = 4,

XTextSize = 16,

DataLabelPadding = .2,

Title = "Topic Rankings Compared Between\nWhat Americans Actually Read vs Want Covered",

SubTitle = "'Actually Read' rank from Parse.ly May 2019 data.\n'Want covered' rank from Axios/SurveyMonkey poll conducted May 17-20, 2019",

Caption = "Source: Axios \nMakeover by @hrbrmstr",

LineColor = colorvect

)

Personally, I think that even 7 topics may be too much, but hopefully you’re  
getting the point that while we’re not losing any information, we’re making it  
easier for the reader to focus on the big changes in the data. It’s easy to  
discern the pattern whether it’s answering a simple question, such as what is the  
number one thing they say they want to read about (Health care), or a more  
complex question such as which topic has the biggest disparity (Sports).

**Use titles, subtitles and captions well**

One thing we can do to make our message clearer is make better use of the title  
and subtitle areas. It seems simple but is too often forgotten. While we’re at  
it I’ll highlight a couple of new capabilities I added to the function:

1. The ability to choose from a select number of ggplot themes. In this case Bob  
   Rudis ipsum\_rc theme.
2. Control the justification of the titles and subtitles and caption.

But the most important change here IMHO is simply choosing words for the title  
and subtitle that convey what we want to look for in the plot or think about.

newggslopegraph(dataframe = temp,

Times = Saydo,

Measurement = Rank,

Grouping = topic,

ReverseYAxis = TRUE,

DataTextSize = 3.5,

YTextSize = 3.2,

XTextSize = 14,

DataLabelPadding = .2,

Title = "Americans Don't Actually Read the News They Say They Want",

SubTitle = "Many sharp differences in rankings in both directions. Hypocrisy, laziness or gratification?",

Caption = "Source: Rud.is \nMakeover by @hrbrmstr",

LineColor = colorvect,

ThemeChoice = "ipsum",

TitleTextSize = 18,

SubTitleTextSize = 12,

SubTitleJustify = "right")

The same plot in Wall Street Journal style (wsj)

newggslopegraph(dataframe = temp,

Times = Saydo,

Measurement = Rank,

Grouping = topic,

ReverseYAxis = TRUE,

ReverseXAxis = TRUE,

DataTextSize = 3.5,

YTextSize = 4,

XTextSize = 13,

DataLabelPadding = .2,

Title = "Americans Don't Actually Read the News They Say They Want",

SubTitle = "Many sharp differences in rankings in both directions.\nHypocrisy or laziness or gratification?",

Caption = "Source: Rud.is \nMakeover by @hrbrmstr",

LineColor = colorvect,

ThemeChoice = "wsj",

TitleTextSize = 15,

CaptionTextSize = 6,

SubTitleTextSize = 11,

SubTitleJustify = "right"

)

I’m not actually sure I like it better at all but simply demonstrating capability

**A final example**

In Bob’s blog post he demonstrated how to add lines and arrows and text to add  
annotation to his plot. I’m of the mindset that less is more and too much  
annotation can be a distraction not an aid to telling our story. As you can see  
from his code it is also relatively complex using native ggplot::geom\_\*’s to  
place things exactly right.

I actually find cowplot easier to use for simple annotation. In the example  
below I’ll shift to the gdocs theme. Save the plot and then use cowplot to add  
one important factoid!

p <- newggslopegraph(dataframe = temp,

Times = Saydo,

Measurement = Rank,

Grouping = topic,

ReverseYAxis = TRUE,

DataTextSize = 3.5,

YTextSize = 4,

XTextSize = 14,

DataLabelPadding = .2,

Title = "Americans Don't Actually Read the News They Say They Want",

SubTitle = "Many sharp differences in rankings in both directions. Hypocrisy, laziness or gratification?",

Caption = "Source: Rud.is \nMakeover by @hrbrmstr",

LineColor = colorvect,

ThemeChoice = "gdocs",

TitleTextSize = 16,

TitleJustify = "center",

SubTitleTextSize = 12,

SubTitleJustify = "center"

)

cowplot::ggdraw(p) +

cowplot::draw\_label(label = "Reading about sports shows the\nlargest difference in ranking --\n11 places!",

colour = "dark blue",

size = 10,

y = .10,

x = .53,

fontface = "italic")

Installation and setup

Long term I’ll try and ensure the version on CRAN is well maintained but for now you’re better served by grabbing the current version from GITHUB.

knitr::[opts\_chunk](https://rdrr.io/pkg/knitr/man/opts_chunk.html)$set(

collapse = TRUE,

comment = "#>"

)

[library](https://rdrr.io/r/base/library.html)([CGPfunctions](https://github.com/ibecav/CGPfunctions))

[library](https://rdrr.io/r/base/library.html)([tidyr](https://tidyr.tidyverse.org/))

[library](https://rdrr.io/r/base/library.html)([dplyr](https://dplyr.tidyverse.org/))

##

## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':

##

## filter, lag

## The following objects are masked from 'package:base':

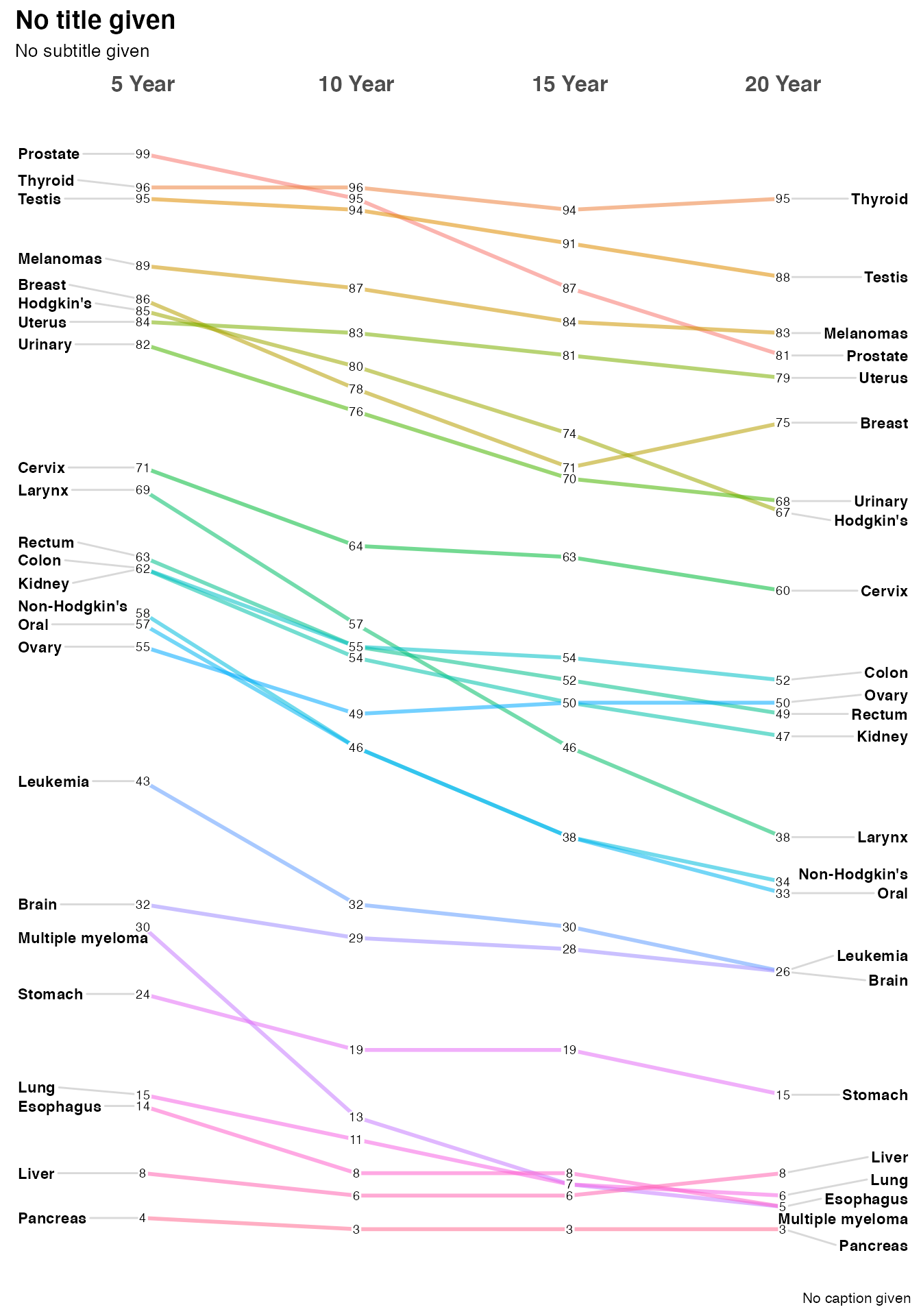
##

## intersect, setdiff, setequal, union

Simple examples

If you’re unfamiliar with slopegraphs or just want to see what the display is all about the dataset I’ve provided can get you started in one line

[newggslopegraph](https://ibecav.github.io/CGPfunctions/reference/newggslopegraph.html)(newcancer,Year,Survival,Type)



Optionally you can provide important label information through Title, Subtitle, and Caption arguments. You can suppress them all together by setting them = NULL but since I think they are very important the default is to gently remind you, that you have not provided any information. Let’s provide a title and sub-title but skip the caption.

[newggslopegraph](https://ibecav.github.io/CGPfunctions/reference/newggslopegraph.html)(dataframe = newcancer,

Times = Year,

Measurement = Survival,

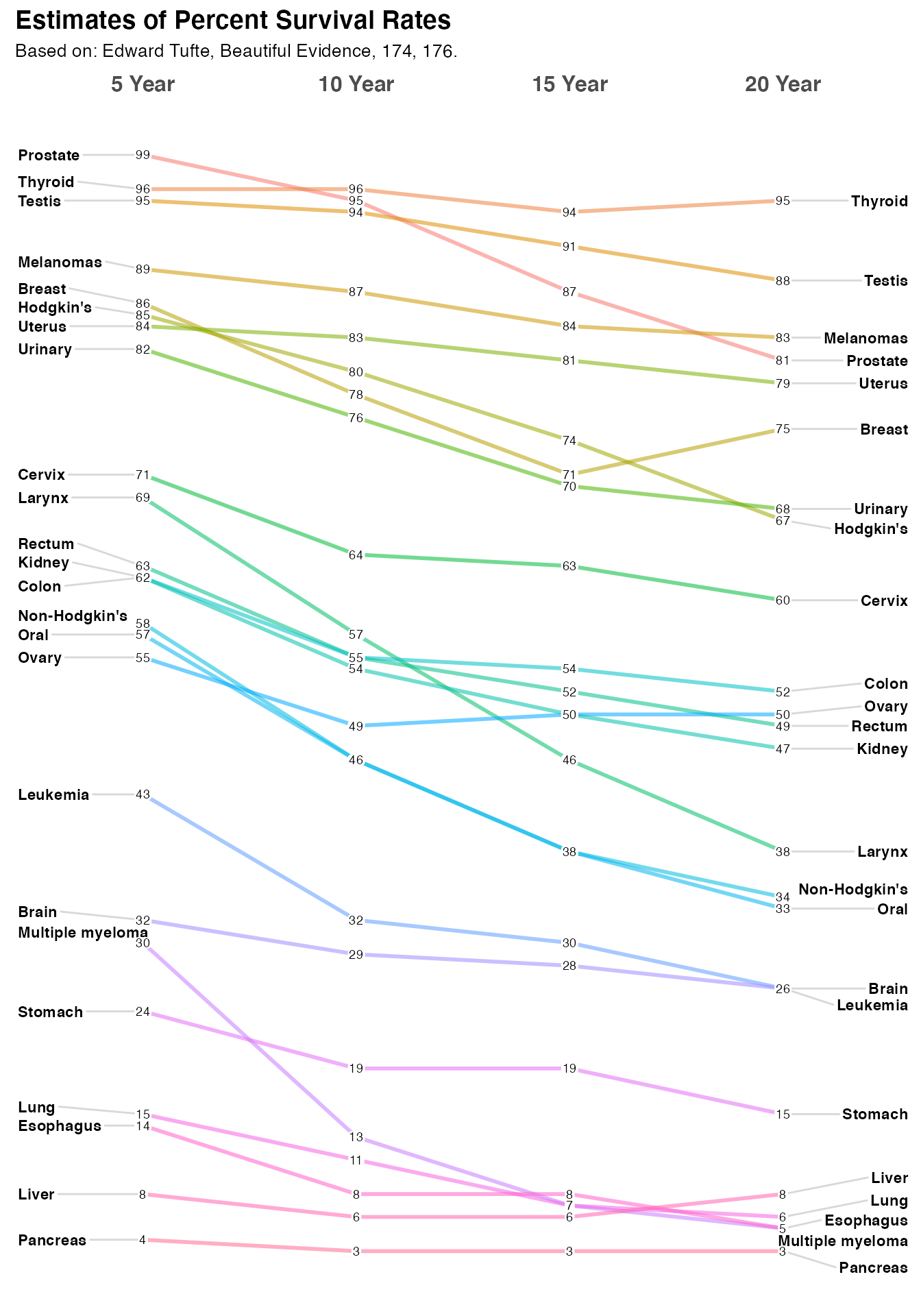
Grouping = Type,

Title = "Estimates of Percent Survival Rates",

SubTitle = "Based on: Edward Tufte, Beautiful Evidence, 174, 176.",

Caption = NULL

)



How it all works

It’s all well and good to get the little demo to work, but it might be useful for you to understand how to extend it out to data you’re interested in.

You’ll need a dataframe with at least three columns. The function will do some basic error checking and complain if you don’t hit the essentials.

1. Times is the column in the dataframe that corresponds to the x axis of the plot and is normally a set of moments in time expressed as either characters, factors or ordered factors (in our case newcancer$Year. If it is truly time series data (especially with a lot of dates you’re much better off using an R function purpose built for that). In newcancer it’s an ordered factor, mainly because if we fed the information in as character the sort order would be Year 10, Year 15, Year 20, Year 5 which is very suboptimal. A command like newcancer$Year <- factor(newcancer$Year,levels = c("Year.5", "Year.10", "Year.15", "Year.20"), labels = c("5 Year","10 Year","15 Year","20 Year"), ordered = TRUE) would be the way to force things they way you want them.
2. Measurement is the column that has the actual numbers you want to display along the y axis. Frequently that’s a percentage but it could just as easily be any number. Watch out for scaling issues here you’ll want to ensure that its not disparate. In our case newcancer$Survival is the percentage of patients surviving at that point in time, so the maximum scale is 0 to 100.
3. Grouping is what controls how many individual lines are portrayed. Every attempt is made to color them and label them in ways that lead to clarity but eventually you can have too many. In our example case the column is newcancer$Type for the type of cancer or location.

Another quick example

In this case we’re going to plot the percent of the vote captured by some Canadian political parties.

The data is loosely based on real data but is not actually accurate.

moredata$Date is the hypothetical polling date as a factor (in this case character would work equally well). moredata$Party is the various political parties and moredata$Pct is the percentage of the vote they are estimated to have.

moredata <- [structure](https://rdrr.io/r/base/structure.html)([list](https://rdrr.io/r/base/list.html)(Date = [structure](https://rdrr.io/r/base/structure.html)([c](https://rdrr.io/r/base/c.html)(1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L, 2L, 3L, 3L, 3L, 3L, 3L),

.Label = [c](https://rdrr.io/r/base/c.html)("11-May-18", "18-May-18", "25-May-18"),

class = "factor"),

Party = [structure](https://rdrr.io/r/base/structure.html)([c](https://rdrr.io/r/base/c.html)(5L, 3L, 2L, 1L, 4L, 5L, 3L, 2L, 1L, 4L, 5L, 3L, 2L, 1L, 4L),

.Label = [c](https://rdrr.io/r/base/c.html)("Green", "Liberal", "NDP", "Others", "PC"),

class = "factor"),

Pct = [c](https://rdrr.io/r/base/c.html)(42.3, 28.4, 22.1, 5.4, 1.8, 41.9, 29.3, 22.3, 5, 1.4, 41.9, 26.8, 26.8, 5, 1.4)),

class = "data.frame",

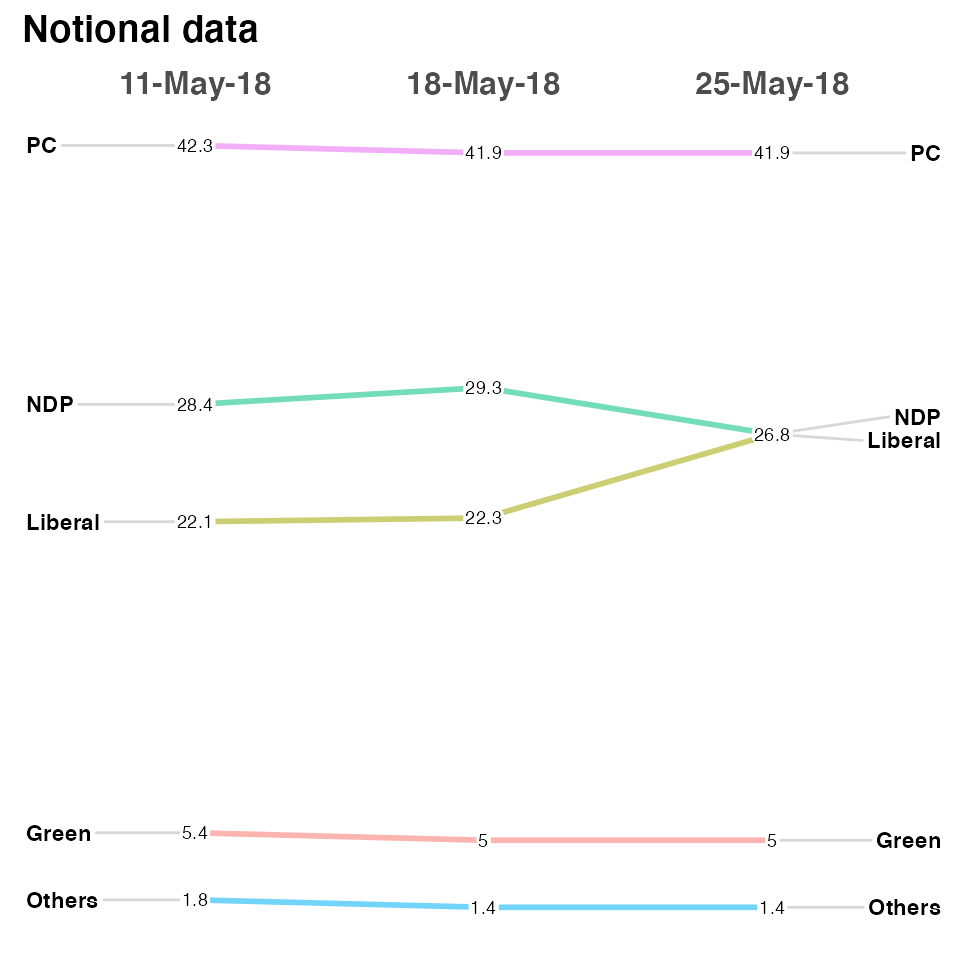
row.names = [c](https://rdrr.io/r/base/c.html)(NA, -15L))

#tail(moredata)

[newggslopegraph](https://ibecav.github.io/CGPfunctions/reference/newggslopegraph.html)(moredata,Date,Pct,Party, Title = "Notional data", SubTitle = NULL, Caption = NULL)

#>

#> Converting 'Date' to an ordered factor



There are a plethora of formatting options. See [?newggslopegraph](https://ibecav.github.io/CGPfunctions/reference/newggslopegraph.html) for all of them. Here’s a few.

[newggslopegraph](https://ibecav.github.io/CGPfunctions/reference/newggslopegraph.html)(moredata, Date, Pct, Party,

Title = "Notional data",

SubTitle = "none",

Caption = "imaginary",

LineColor = "gray",

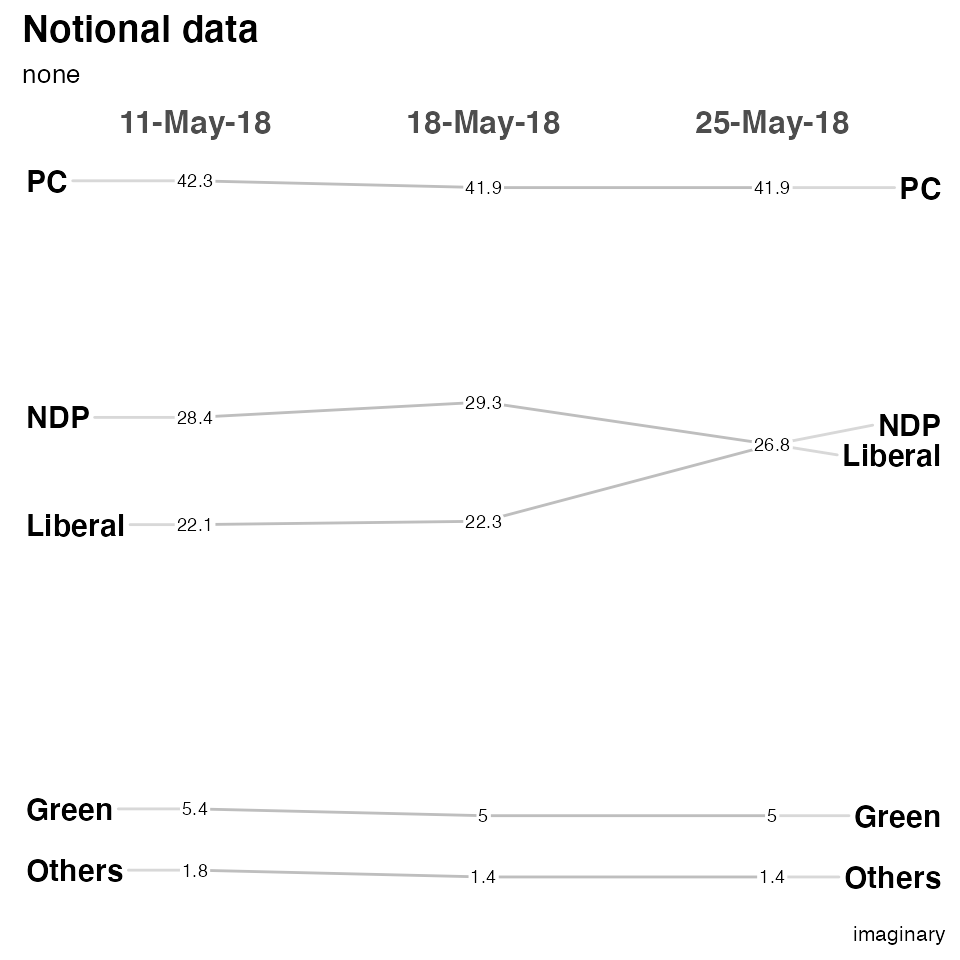
LineThickness = .5,

YTextSize = 4

)

#>

#> Converting 'Date' to an ordered factor



The most complex is LineColor where you can do the following if you want to highlight the difference between the Liberal and NDP parties while making the other three less prominent…

[newggslopegraph](https://ibecav.github.io/CGPfunctions/reference/newggslopegraph.html)(moredata, Date, Pct, Party,

Title = "Notional data",

SubTitle = "none",

Caption = "imaginary",

LineColor = [c](https://rdrr.io/r/base/c.html)("Green" = "gray", "Liberal" = "green", "NDP" = "red", "Others" = "gray", "PC" = "gray"),

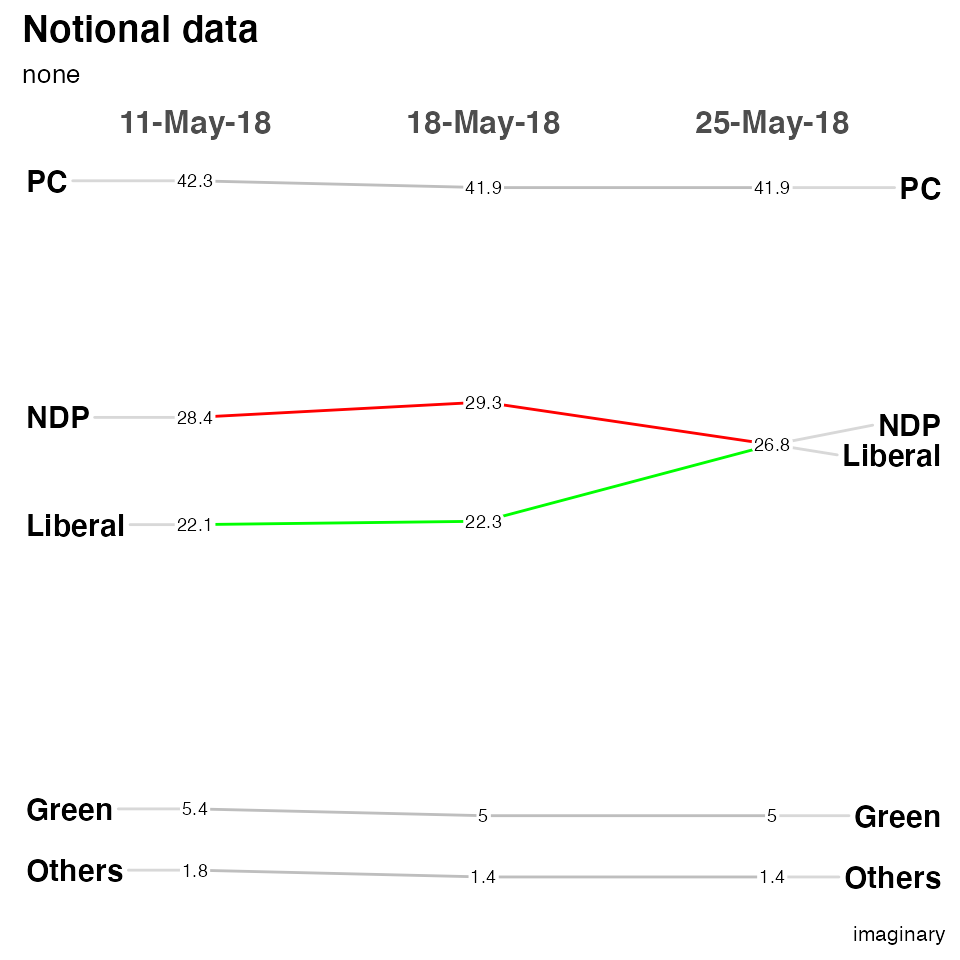
LineThickness = .5,

YTextSize = 4

)

#>

#> Converting 'Date' to an ordered factor



Slopegraph Best Practices

1. Scaling – this function plots to scale on an actual scale.
2. If the datapoints or labels are bunching up, expand the vertical size of the plot as necessary.
3. Names of the items on both the left-hand and right-hand axes are aligned, to make vertical scanning of the items’ names easier.
4. Many suggest using a thin, light gray line to connect the data. A too-heavy line is unnecessary and will make the chart harder to read.
5. When a chart features many slope intersections, judicious use of color can avoid what Ben Fry describes as the “pile of sticks” phenomenon (Visualizing Data, 121).
6. A table (with more statistical detail) might be a good complement to use alongside the slopegraph. As Tufte notes: “The data table and the slopegraph are colleagues in explanation not competitors. One display can serve some but not all functions.”

One last set of data

Also from Tufte, this is data about a select group of countries Gross Domestic Product (GDP). I’ll use it to show you a tricky way to highlight certain countries without making a named vector with LineColor = c(rep("gray",3), "red", rep("gray",3), "red", rep("gray",10)) the excess vector entries are silently dropped… The bottom line is that LineColor is simply a character vector that you can fill any way you choose.

[newggslopegraph](https://ibecav.github.io/CGPfunctions/reference/newggslopegraph.html)(newgdp,

Year,

GDP,

Country,

Title = "Gross GDP",

SubTitle = NULL,

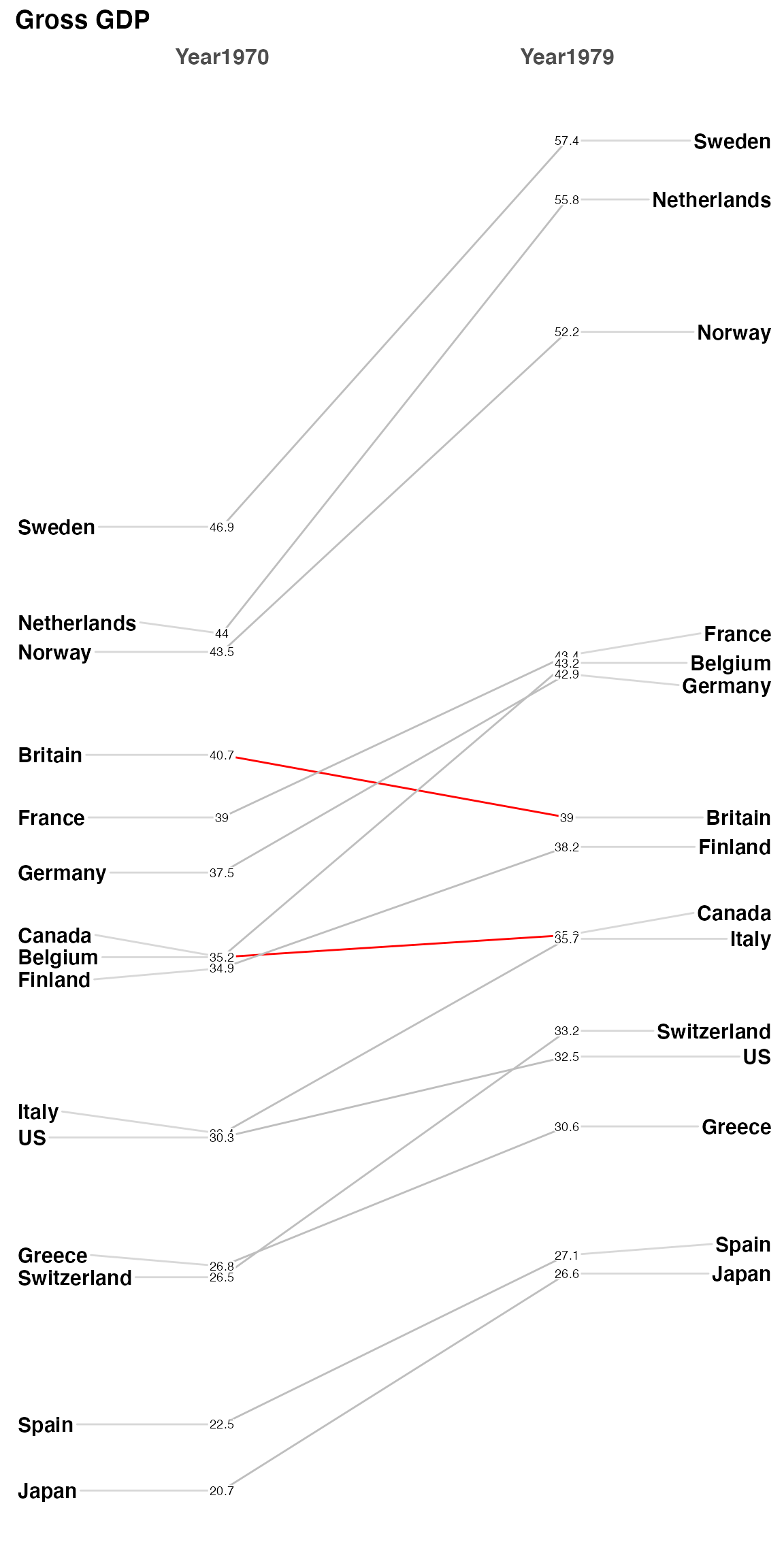
Caption = NULL,

LineThickness = .5,

YTextSize = 4,

LineColor = [c](https://rdrr.io/r/base/c.html)([rep](https://rdrr.io/r/base/rep.html)("gray",3), "red", [rep](https://rdrr.io/r/base/rep.html)("gray",3), "red", [rep](https://rdrr.io/r/base/rep.html)("gray",10))

)



Finally, let me take a moment about crowding and labeling. I’ve made every effort to try and deconflict the labels on the left and right axis (in this example the Country) and that should work automatically as you resize your plot dimensions. \*\* pro tip - if you use RStudio you can press the zoom icon and then use the rescaling of the window to see best choices \*\*.

But the numbers (GDP) are a different matter and there’s no easy way to ensure separation in a case like this data. There’s a decent total spread from 57.4 to 20.7 and some really close measurements like France, Belgium, and Germany on the right side. My suggestion is in a case like this one you create a new column in your dataframe with two significant places. So specifically it would be newgdp$rGDP <- signif(newgdp$GDP, 2). In my testing, at least, I’ve found this helps without creating inaccuracy and not causing you to try and “stretch” vertically to disambiguate the numbers. This time I’ll also use LineColor to highlight how Canada, Finland and Belgium fare from 1970 to 1979.

Then to demonstrate how flexible LineColor really is I’ll use some tidyverse tools to build a named list of countries and colors. The country’s line color will be determined by whether the difference between 1979 is positive, near neutral or negative.

newgdp$rGDP <- [signif](https://rdrr.io/r/base/Round.html)(newgdp$GDP, 2)

[newggslopegraph](https://ibecav.github.io/CGPfunctions/reference/newggslopegraph.html)(newgdp,

Year,

rGDP,

Country,

Title = "Gross GDP",

SubTitle = NULL,

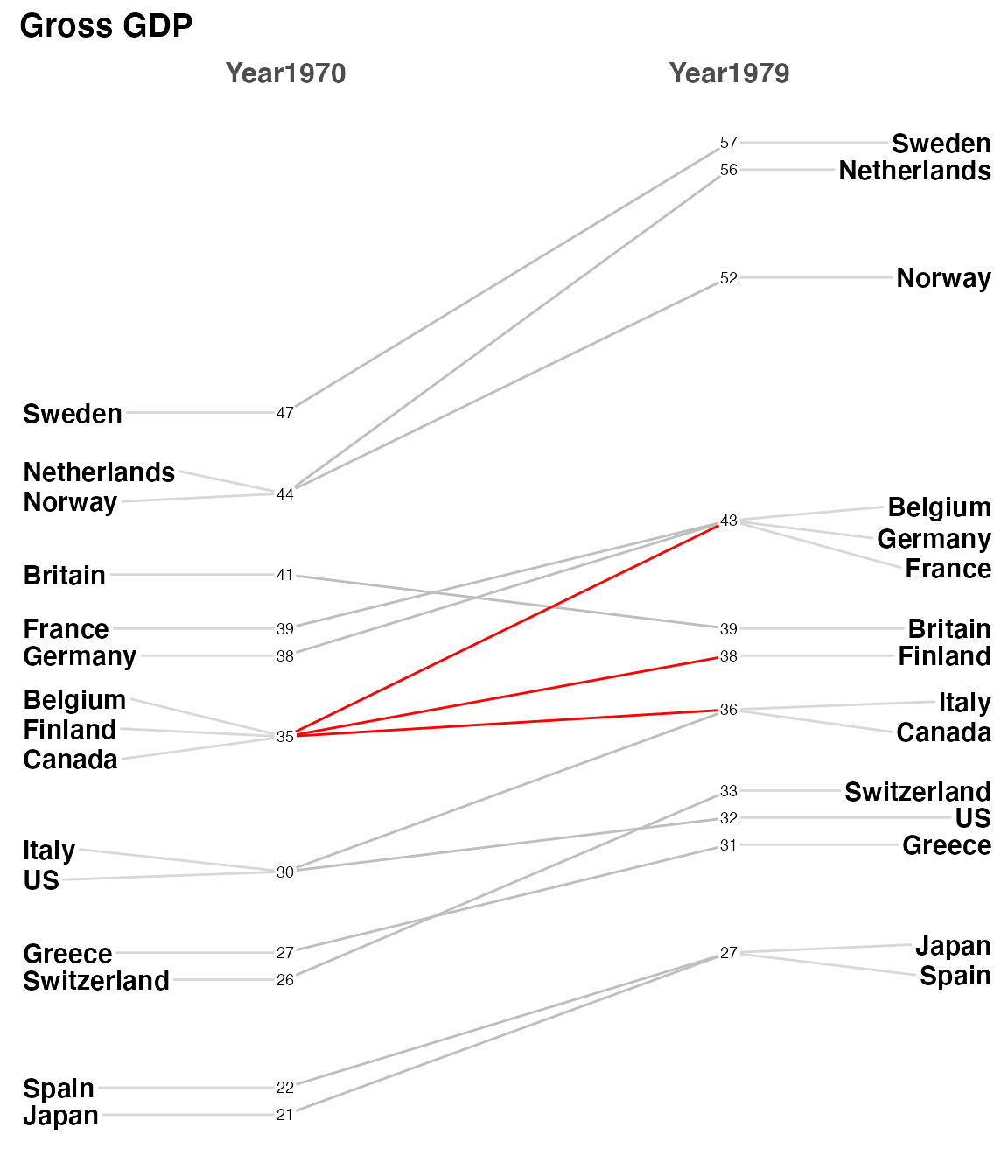
Caption = NULL,

LineThickness = .5,

YTextSize = 4,

LineColor = [c](https://rdrr.io/r/base/c.html)([rep](https://rdrr.io/r/base/rep.html)("gray",6), [rep](https://rdrr.io/r/base/rep.html)("red",2), "red", [rep](https://rdrr.io/r/base/rep.html)("gray",10))

)



custom\_colors <- tidyr::[pivot\_wider](https://tidyr.tidyverse.org/reference/pivot_wider.html)(newgdp,

id\_cols = Country,

names\_from = Year,

values\_from = GDP) %>%

[mutate](https://dplyr.tidyverse.org/reference/mutate.html)(difference = Year1979 - Year1970) %>%

[mutate](https://dplyr.tidyverse.org/reference/mutate.html)(trend = [case\_when](https://dplyr.tidyverse.org/reference/case_when.html)(

difference >= 2 ~ "green",

difference <= -1 ~ "red",

TRUE ~ "gray"

)

) %>%

[select](https://dplyr.tidyverse.org/reference/select.html)(Country, trend) %>%

tibble::[deframe](https://tibble.tidyverse.org/reference/enframe.html)()

custom\_colors

#> Sweden Netherlands Norway Britain France Germany

#> "green" "green" "green" "red" "green" "green"

#> Belgium Canada Finland Italy US Greece

#> "green" "gray" "green" "green" "green" "green"

#> Switzerland Spain Japan

#> "green" "green" "green"

[newggslopegraph](https://ibecav.github.io/CGPfunctions/reference/newggslopegraph.html)(newgdp,

Year,

rGDP,

Country,

Title = "Gross GDP",

SubTitle = NULL,

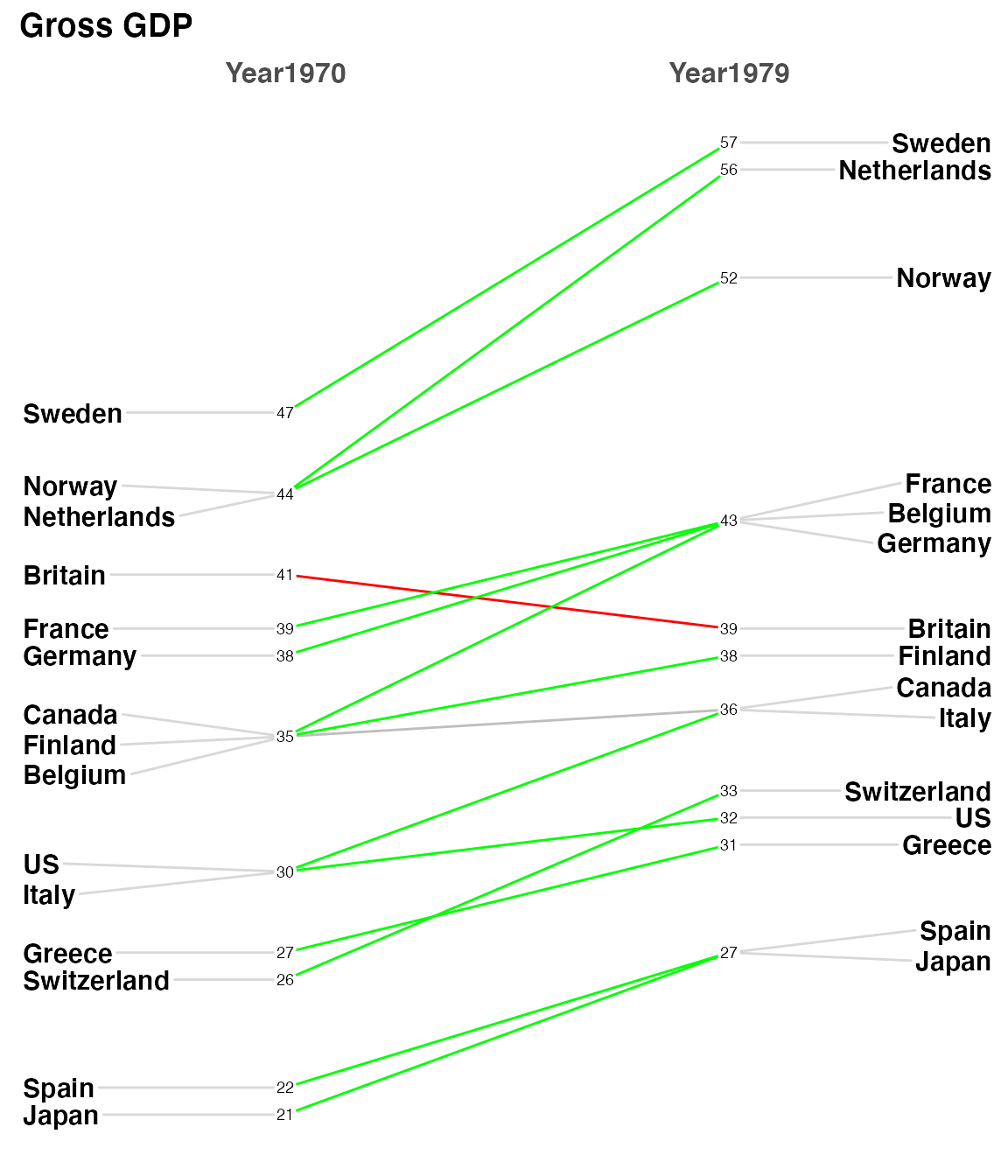
Caption = NULL,

LineThickness = .5,

YTextSize = 4,

LineColor = custom\_colors

)



**Done**

I really enjoyed Bob’s original post and was happy to be inspired to make a few  
changes to newggslopegraph to hopefully make it more useful . I am always open  
to comments, corrections and suggestions. Feel free to leave a comment in disqus  
or send me an email.