The second case happened to me this week. been aware of slopegraphs  
and bumpcharts for quite some time, and certainly am aware.

The article does a good job of looking at slopegraphs in both R (via  
plotrix) and Stata, even providing the code to do the work. My  
challenge was that even though I’m adequate at plotting in base R, I  
much prefer using ggplot2 wherever and whenever possible. My memory  
was that I had seen another article on the related topic of a  
bumpchart on R-bloggers in the not too distant past.

Code Chunks – Bump Chart

## Required packages

Importing the ggplot2 package is mandatory. dplyr on the other hand can be replaced by other data manipulation libraries.

library(dplyr) *# data manipulation*

library(ggplot2) *# visualisation*

## Data foundation

Before we can start building the Bump Chart we first need to get the data in the correct shape. Let’s assume that we start with the number of cumulated medals after each competition day per country.

## date country gold silver bronze

## 1 2018-02-10 GER 2 0 0

## 2 2018-02-10 NED 1 2 1

## 3 2018-02-10 KOR 1 0 0

## 4 2018-02-10 SWE 1 0 0

## 5 2018-02-10 NOR 0 3 1

## date country gold silver bronze

## 451 2018-02-25 NOR 14 14 11

## 452 2018-02-25 GER 14 10 7

## 453 2018-02-25 CAN 11 8 10

## 454 2018-02-25 USA 9 8 6

## 455 2018-02-25 NED 8 6 6

The data shown here, was extracted from the official homepage with the help of the rvest package. To ensure reproducibility of the plots I attached the data at the end of the blog post.

We first have to rank the countries based on their accumulated number of gold, silver and bronze medals for every day of the Olympic Games. In case of a tie the countries are ranked in alphabetical order.

df.rankings **<-** df **%>%**

group\_by(date) **%>%**

arrange(date, desc(gold), desc(silver), desc(bronze), country) **%>%**

mutate(ranking **=** row\_number(),

day **=** **as.numeric**(as.Date(date)) **-** 17571) **%>%**

as.data.frame()

## date country gold silver bronze ranking day

## 1 2018-02-10 GER 2 0 0 1 1

## 2 2018-02-10 NED 1 2 1 2 1

## 3 2018-02-10 KOR 1 0 0 3 1

## 4 2018-02-10 SWE 1 0 0 4 1

## 5 2018-02-10 NOR 0 3 1 5 1

## date country gold silver bronze ranking day

## 451 2018-02-25 NOR 14 14 11 1 16

## 452 2018-02-25 GER 14 10 7 2 16

## 453 2018-02-25 CAN 11 8 10 3 16

## 454 2018-02-25 USA 9 8 6 4 16

## 455 2018-02-25 NED 8 6 6 5 16

## Basic Version

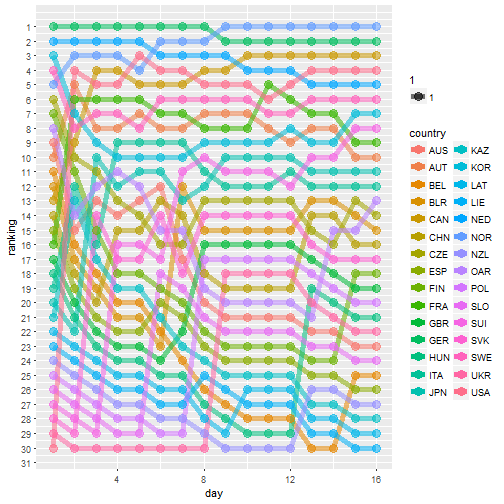
A simple version of the bump chart is created by using the syntax for a line chart and reversing the y axis.

ggplot(data **=** df.rankings, aes(x **=** day, y **=** ranking, group **=** country)) **+**

geom\_line(aes(color **=** country, alpha **=** 1), size **=** 2) **+**

geom\_point(aes(color **=** country, alpha **=** 1), size **=** 4) **+**

scale\_y\_reverse(breaks **=** 1**:**nrow(df.rankings))



Now we can see, that the data is in the correct format but the plot looks really ugly and shouldn’t be used in its current shape.

## Style it

With the help some simple theme changes and adding the country information next to the lines we can clean up the mess.

my\_theme **<-** **function**() {

*# Colors*

color.background **=** "white"

color.text **=** "#22211d"

*# Begin construction of chart*

theme\_bw(base\_size**=**15) **+**

*# Format background colors*

theme(panel.background **=** element\_rect(fill**=**color.background, color**=**color.background)) **+**

theme(plot.background **=** element\_rect(fill**=**color.background, color**=**color.background)) **+**

theme(panel.border **=** element\_rect(color**=**color.background)) **+**

theme(strip.background **=** element\_rect(fill**=**color.background, color**=**color.background)) **+**

*# Format the grid*

theme(panel.grid.major.y **=** element\_blank()) **+**

theme(panel.grid.minor.y **=** element\_blank()) **+**

theme(axis.ticks **=** element\_blank()) **+**

*# Format the legend*

theme(legend.position **=** "none") **+**

*# Format title and axis labels*

theme(plot.title **=** element\_text(color**=**color.text, size**=**20, face **=** "bold")) **+**

theme(axis.title.x **=** element\_text(size**=**14, color**=**"black", face **=** "bold")) **+**

theme(axis.title.y **=** element\_text(size**=**14, color**=**"black", face **=** "bold", vjust**=**1.25)) **+**

theme(axis.text.x **=** element\_text(size**=**10, vjust**=**0.5, hjust**=**0.5, color **=** color.text)) **+**

theme(axis.text.y **=** element\_text(size**=**10, color **=** color.text)) **+**

theme(strip.text **=** element\_text(face **=** "bold")) **+**

*# Plot margins*

theme(plot.margin **=** unit(**c**(0.35, 0.2, 0.3, 0.35), "cm"))

}

For the sake of clarity we will only focus on the top 10 show.top.n <- 10 observations. Be careful to limit the y axis in the plot and not filter the data beforehand, otherwise the plot will be incorrect.

show.top.n **<-** 10

ggplot(data **=** df.rankings, aes(x **=** day, y **=** ranking, group **=** country)) **+**

geom\_line(aes(color **=** country, alpha **=** 1), size **=** 2) **+**

geom\_point(aes(color **=** country, alpha **=** 1), size **=** 4) **+**

geom\_point(color **=** "#FFFFFF", size **=** 1) **+**

scale\_y\_reverse(breaks **=** 1**:**show.top.n) **+**

scale\_x\_continuous(breaks **=** 1**:**16, minor\_breaks **=** 1**:**16, expand **=** **c**(.05, .05)) **+**

geom\_text(data **=** df.rankings **%>%** filter(day **==** "1"),

aes(label **=** country, x **=** 0.5) , hjust **=** .85, fontface **=** "bold", color **=** "#888888", size **=** 4) **+**

geom\_text(data **=** df.rankings **%>%** filter(day **==** "16"),

aes(label **=** country, x **=** 16.5) , hjust **=** 0.15, fontface **=** "bold", color **=** "#888888", size **=** 4) **+**

coord\_cartesian(ylim **=** **c**(1,show.top.n)) **+**

theme(legend.position **=** "none") **+**

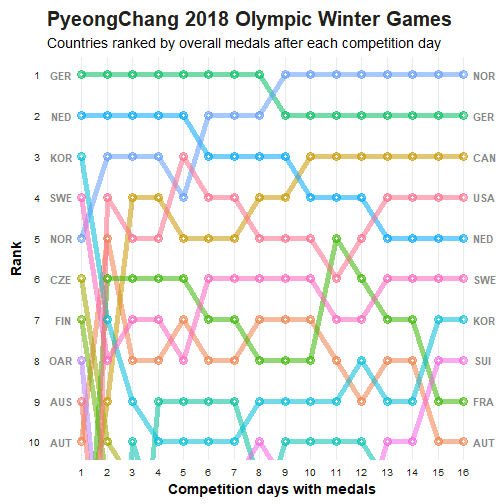
labs(x **=** "Competition days with medals",

y **=** "Rank",

title **=** "PyeongChang 2018 Olympic Winter Games",

subtitle **=** "Countries ranked by overall medals after each competition day") **+**

my\_theme()



This already looks way better and we can improve it even further.

## Highlight

By highlighting only a few of the lines we can emphasise these countries in the plot. Therefore, we need to create a new feature for the colour parameter where all non-highlighted observations are labeled as ‘zzz’.

df.rankings **<-** df.rankings **%>%**

mutate(flag **=** ifelse(country **%in%** **c**("NOR","GER","CAN","USA","NED"), **TRUE**, **FALSE**),

country\_col **=** if\_else(flag **==** **TRUE**, country, "zzz"))

The bump chart code needs only minor changes. Both the color parameter and the colour palette scale\_color\_manual() are updated accordingly.

show.top.n **<-** 10

ggplot(data **=** df.rankings, aes(x **=** day, y **=** ranking, group **=** country)) **+**

geom\_line(aes(color **=** country\_col, alpha **=** 1), size **=** 2) **+**

geom\_point(color **=** "#FFFFFF", size **=** 4) **+**

geom\_point(aes(color **=** country\_col, alpha **=** 1), size **=** 4) **+**

geom\_point(color **=** "#FFFFFF", size **=** 1) **+**

scale\_y\_reverse(breaks **=** 1**:**show.top.n) **+**

scale\_x\_continuous(breaks **=** 1**:**16, minor\_breaks **=** 1**:**16, expand **=** **c**(.05, .05)) **+**

geom\_text(data **=** df.rankings **%>%** filter(day **==** "1"),

aes(label **=** country, x **=** 0.5) , hjust **=** .85, fontface **=** "bold", color **=** "#888888", size **=** 4) **+**

geom\_text(data **=** df.rankings **%>%** filter(day **==** "16"),

aes(label **=** country, x **=** 16.5) , hjust **=** 0.15, fontface **=** "bold", color **=** "#888888", size **=** 4) **+**

coord\_cartesian(ylim **=** **c**(1,show.top.n)) **+**

theme(legend.position **=** "none") **+**

labs(x **=** "Competition days with medals",

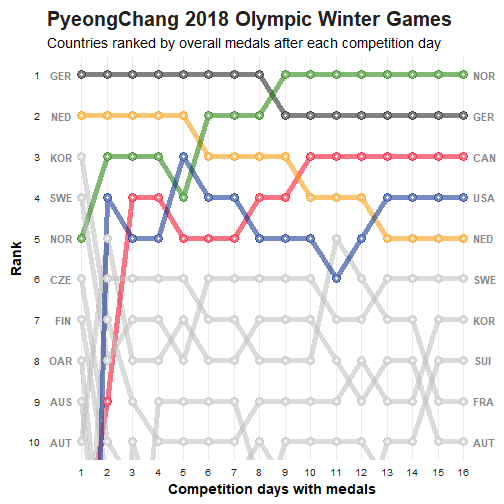
y **=** "Rank",

title **=** "PyeongChang 2018 Olympic Winter Games",

subtitle **=** "Countries ranked by overall medals after each competition day") **+**

my\_theme() **+**

scale\_color\_manual(values **=** **c**("#F70020","#191A1A","#FB9701","#1A7D00","#072C8F","grey"))



Thanks to these small adjustments it is easy to track the winning countries of the Olympic Games in PyeongChang.

My post today will draw a little bit from all their work and hopefully  
provide some useful samples for others to draw on if they share some of  
my quirks about data layout and a preference for ggplot2 versus base  
plot. I’m going to focus almost exclusively on slopegraphs, although  
much of the work could be extended to bumpcharts as well.

**Setup and library loading**

We’re going to make occasional use of dplyr to manipulate the data,  
extensive use of ggplot2 to do the plotting and ggrepel to solve one  
specific labeling problem. We’ll load them and I am suppressing the  
message from dplyr about namespace overrides.

require(dplyr)

require(ggplot2)

require(ggrepel)

require(kableExtra)

**Politics in Ontario**

For the reader’s convenience I’ve made the data available via a structure  
command. We have data about two different polling dates, for 5 political  
parties, and the measured variable is percent of people supporting  
expressed as x.x (i.e. already multiplied by  
100).

data <- structure(list( Date = structure(c(1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L, 2L),

.Label = c("11-May-18", "18-May-18"),

class = "factor"),

Party = structure(c(5L, 3L, 2L, 1L, 4L, 5L, 3L, 2L, 1L, 4L),

.Label = c("Green", "Liberal", "NDP", "Others", "PC"),

class = "factor"),

Pct = c(42.3, 28.4, 22.1, 5.4, 1.8, 41.9, 29.3, 22.3, 5, 1.4)),

class = "data.frame",

row.names = c(NA, -10L))

str(data)

## 'data.frame': 10 obs. of 3 variables:

## $ Date : Factor w/ 2 levels "11-May-18","18-May-18": 1 1 1 1 1 2 2 2 2 2

## $ Party: Factor w/ 5 levels "Green","Liberal",..: 5 3 2 1 4 5 3 2 1 4

## $ Pct : num 42.3 28.4 22.1 5.4 1.8 41.9 29.3 22.3 5 1.4

head(data)

## Date Party Pct

## 1 11-May-18 PC 42.3

## 2 11-May-18 NDP 28.4

## 3 11-May-18 Liberal 22.1

## 4 11-May-18 Green 5.4

## 5 11-May-18 Others 1.8

## 6 18-May-18 PC 41.9

Let’s just take the data as we have it and feed it to ggplot in a nice  
simple fashion and see what we get with very little effort.

ggplot(data = data, aes(x = Date, y = Pct, group = Party)) +

geom\_line(aes(color = Party, alpha = 1), size = 2) +

geom\_point(aes(color = Party, alpha = 1), size = 4) +

# Labelling as desired

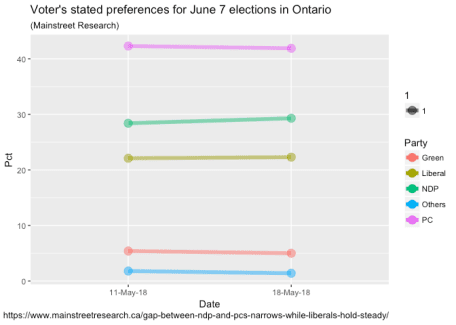
labs(

title = "Voter's stated preferences for June 7 elections in Ontario",

subtitle = "(Mainstreet Research)",

caption = "https://www.mainstreetresearch.ca/gap-between-ndp-and-pcs-narrows-while-liberals-hold-steady/"

)



The nice thing about ggplot is once you get used to the syntax it  
becomes very “readable”. We’ve identified our dataset, the x & y  
variables and our grouping variable. Lines too big? An adjustment to  
size = 2 does it. Don’t like colors? Pull the color = Party clause.

So we’re already pretty close to what we need. Things are scaled  
properly and the basic labeling of titles etc. is accomplished. Our  
biggest “problem” is that ggplot has been a little too helpful and  
adding some things we’d like to remove to give it a more *“Tuftesque”*  
look. So what we’ll do in the next few steps is add lines of code – but  
they are mainly designed to remove unwanted elements. This is in  
contrast to a base plot where we have to write the code to add elements.

So lets:

* Move the x axis labels to the top with scale\_x\_discrete(position =  
  "top")
* Change to a nice clean black and white theme theme\_bw()
* Not display any legend(s) theme(legend.position = "none")
* Remove the default border from our plot theme(panel.border =  
  element\_blank())

ggplot(data = data, aes(x = Date, y = Pct, group = Party)) +

geom\_line(aes(color = Party, alpha = 1), size = 2) +

geom\_point(aes(color = Party, alpha = 1), size = 4) +

# move the x axis labels up top

scale\_x\_discrete(position = "top") +

theme\_bw() +

# Format tweaks

# Remove the legend

theme(legend.position = "none") +

# Remove the panel border

theme(panel.border = element\_blank()) +

# Labelling as desired

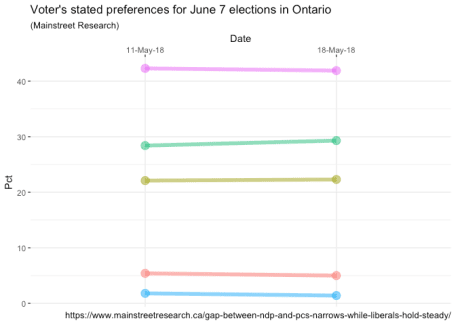
labs(

title = "Voter's stated preferences for June 7 elections in Ontario",

subtitle = "(Mainstreet Research)",

caption = "https://www.mainstreetresearch.ca/gap-between-ndp-and-pcs-narrows-while-liberals-hold-steady/"

)



Nice progress! Continuing to remove things that can be considered  
“clutter” we add some additional lines that all end in  
element\_blank() and are invoked to remove default plot items such as  
the plot grid, the y axcis text, etc..

ggplot(data = data, aes(x = Date, y = Pct, group = Party)) +

geom\_line(aes(color = Party, alpha = 1), size = 2) +

geom\_point(aes(color = Party, alpha = 1), size = 4) +

# move the x axis labels up top

scale\_x\_discrete(position = "top") +

theme\_bw() +

# Format tweaks

# Remove the legend

theme(legend.position = "none") +

# Remove the panel border

theme(panel.border = element\_blank()) +

# Remove just about everything from the y axis

theme(axis.title.y = element\_blank()) +

theme(axis.text.y = element\_blank()) +

theme(panel.grid.major.y = element\_blank()) +

theme(panel.grid.minor.y = element\_blank()) +

# Remove a few things from the x axis and increase font size

theme(axis.title.x = element\_blank()) +

theme(panel.grid.major.x = element\_blank()) +

theme(axis.text.x.top = element\_text(size=12)) +

# Remove x & y tick marks

theme(axis.ticks = element\_blank()) +

# Labelling as desired

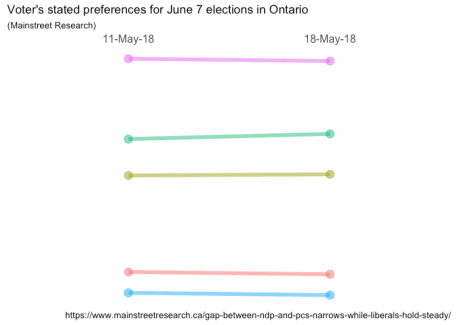
labs(

title = "Voter's stated preferences for June 7 elections in Ontario",

subtitle = "(Mainstreet Research)",

caption = "https://www.mainstreetresearch.ca/gap-between-ndp-and-pcs-narrows-while-liberals-hold-steady/"

)



Very nice! We’re almost there! The “almost” is because now that we  
have removed both the legend and all scales and tick marks we no longer  
know who is who, and what the numbers are! Plus, I’m a little unhappy  
with the way the titles are formatted, so we’ll play with that. Later,  
I’ll get fancy but for now let’s just add some simple text labels on  
the left and right to show the party name and their percentage. The code  
geom\_text(aes(label = Party)) will place the party name right on top  
of the points that anchor either end of the line. If we make that  
geom\_text(aes(label = paste0(Party, " - ", Pct, "%"))) then we’ll get  
labels that have both the party and the percent all neatly formatted,  
but still right on top of the points that anchor the ends of the line.  
hjust controls horizontal justification so if we change it to  
geom\_text(aes(label = paste0(Party, " - ", Pct, "%")), hjust = 1.35)  
both sets of labels will slide to the left which is exactly what we want  
for the May 11 labels but not the May 18 labels. If we feed hjust a  
negative number they’ll go the other way. So what we’ll do is filter the  
data using the filter function from dplyr and place the left hand  
labels differently than the right hand labels. While we’re at it we’ll  
make it bold face font and a little larger…

ggplot(data = data, aes(x = Date, y = Pct, group = Party)) +

geom\_line(aes(color = Party, alpha = 1), size = 2) +

geom\_point(aes(color = Party, alpha = 1), size = 4) +

geom\_text(data = data %>% filter(Date == "11-May-18"),

aes(label = paste0(Party, " - ", Pct, "%")) ,

hjust = 1.35,

fontface = "bold",

size = 4) +

geom\_text(data = data %>% filter(Date == "18-May-18"),

aes(label = paste0(Party, " - ", Pct, "%")) ,

hjust = -.35,

fontface = "bold",

size = 4) +

# move the x axis labels up top

scale\_x\_discrete(position = "top") +

theme\_bw() +

# Format tweaks

# Remove the legend

theme(legend.position = "none") +

# Remove the panel border

theme(panel.border = element\_blank()) +

# Remove just about everything from the y axis

theme(axis.title.y = element\_blank()) +

theme(axis.text.y = element\_blank()) +

theme(panel.grid.major.y = element\_blank()) +

theme(panel.grid.minor.y = element\_blank()) +

# Remove a few things from the x axis and increase font size

theme(axis.title.x = element\_blank()) +

theme(panel.grid.major.x = element\_blank()) +

theme(axis.text.x.top = element\_text(size=12)) +

# Remove x & y tick marks

theme(axis.ticks = element\_blank()) +

# Format title & subtitle

theme(plot.title = element\_text(size=14, face = "bold", hjust = 0.5)) +

theme(plot.subtitle = element\_text(hjust = 0.5)) +

# Labelling as desired

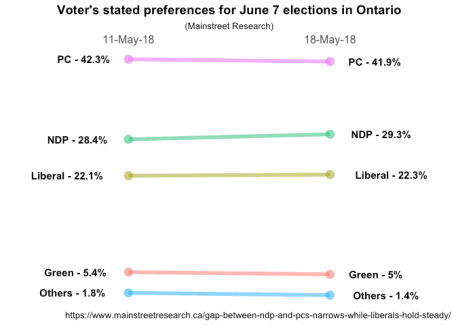
labs(

title = "Voter's stated preferences for June 7 elections in Ontario",

subtitle = "(Mainstreet Research)",

caption = "https://www.mainstreetresearch.ca/gap-between-ndp-and-pcs-narrows-while-liberals-hold-steady/"

)



Eureka! Not perfect yet but definitely looking good.

**Adding complexity**

I’m feeling pretty good about the solution so far but there are three  
things I’d like to make better.

1. How well will this solution work when we have more than two time  
   periods? Need to make sure it generalizes to a more complex case.
2. Our very neat little labels will overlap each other.  
   In his post I believe he mentions that he manually moved them in  
   some cases. Let’s try and fix that.
3. Oh my, that’s a lot of code to keep cutting and pasting, can we  
   simplify?

To test #1 and #2 I have “invented”” a new dataset called moredata.  
**It is fictional** it’s labelled May 25th but today is actually May  
24th. But I created it to add a third polling date and to make sure that  
we had a chance to test what happens when we have two identical  
datapoints on the same day. Notice that on May 25th the polling numbers  
for the Liberals and the NDP are identical at  
26.8%.

moredata <- structure(list(Date = structure(c(1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L, 2L, 3L, 3L, 3L, 3L, 3L),

.Label = c("11-May-18", "18-May-18", "25-May-18"),

class = "factor"),

Party = structure(c(5L, 3L, 2L, 1L, 4L, 5L, 3L, 2L, 1L, 4L, 5L, 3L, 2L, 1L, 4L),

.Label = c("Green", "Liberal", "NDP", "Others", "PC"),

class = "factor"),

Pct = c(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(NA, -15L))

tail(moredata)

## Date Party Pct

## 10 18-May-18 Others 1.4

## 11 25-May-18 PC 41.9

## 12 25-May-18 NDP 26.8

## 13 25-May-18 Liberal 26.8

## 14 25-May-18 Green 5.0

## 15 25-May-18 Others 1.4

You’ll notice at the beginning of this post I loaded the ggrepel  
library. ggrepel works with ggplot2 to *repel* things that overlap,  
in this case our geom\_text labels. The invocation is geom\_text\_repel  
and it is very similar to geom\_text but allows us to deconflict the  
overlaps. We’ll use hjust = "left" and hjust = "right" to control  
justifying the labels. We’ll use a fixed nudge left and right nudge\_x =  
-.45 and nudge\_x = .5 to move the labels left and right off the  
plotted data points and we will explicitly tell geom\_text\_repel to  
only move the labels vertically to avoid overlap with direction = "y".  
Everything else remains the same.

ggplot(data = moredata, aes(x = Date, y = Pct, group = Party)) +

geom\_line(aes(color = Party, alpha = 1), size = 2) +

geom\_point(aes(color = Party, alpha = 1), size = 4) +

geom\_text\_repel(data = moredata %>% filter(Date == "11-May-18"),

aes(label = paste0(Party, " - ", Pct, "%")) ,

hjust = "left",

fontface = "bold",

size = 4,

nudge\_x = -.45,

direction = "y") +

geom\_text\_repel(data = moredata %>% filter(Date == "25-May-18"),

aes(label = paste0(Party, " - ", Pct, "%")) ,

hjust = "right",

fontface = "bold",

size = 4,

nudge\_x = .5,

direction = "y") +

# move the x axis labels up top

scale\_x\_discrete(position = "top") +

theme\_bw() +

# Format tweaks

# Remove the legend

theme(legend.position = "none") +

# Remove the panel border

theme(panel.border = element\_blank()) +

# Remove just about everything from the y axis

theme(axis.title.y = element\_blank()) +

theme(axis.text.y = element\_blank()) +

theme(panel.grid.major.y = element\_blank()) +

theme(panel.grid.minor.y = element\_blank()) +

# Remove a few things from the x axis and increase font size

theme(axis.title.x = element\_blank()) +

theme(panel.grid.major.x = element\_blank()) +

theme(axis.text.x.top = element\_text(size=12)) +

# Remove x & y tick marks

theme(axis.ticks = element\_blank()) +

# Format title & subtitle

theme(plot.title = element\_text(size=14, face = "bold", hjust = 0.5)) +

theme(plot.subtitle = element\_text(hjust = 0.5)) +

# Labelling as desired

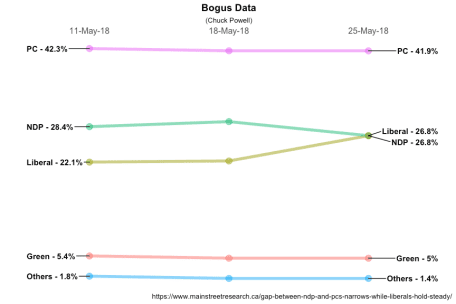
labs(

title = "Bogus Data",

subtitle = "(Chuck Powell)",

caption = "https://www.mainstreetresearch.ca/gap-between-ndp-and-pcs-narrows-while-liberals-hold-steady/"

)



Very nice! We have confirmed that our solution works for more than two  
dates without any additional changes and we have found a solution to the  
label overlap issue. In a little while we’ll talk about labeling the  
data points in the center (if we want to).

Before we move on let’s make our life a little simpler. While the output  
plot is good it’s a lot of code to produce one graph. Let’s see if we  
can simplify…

Since ggplot2 objects are just regular R objects, you can put them in a  
list. This means you can apply all of R’s great functional programming  
tools. For example, if you wanted to add different geoms to the same  
base plot, you could put them in a list and use lapply().

But for now let’s at least take all the invariant lines of code and put  
them in a list. Then when we go to plot we can just invoke the list and  
remain confident we get the right formatting. For now let’s name this  
list something quaint and obvious like MySpecial.

MySpecial <- list(

# move the x axis labels up top

scale\_x\_discrete(position = "top"),

theme\_bw(),

# Format tweaks

# Remove the legend

theme(legend.position = "none"),

# Remove the panel border

theme(panel.border = element\_blank()),

# Remove just about everything from the y axis

theme(axis.title.y = element\_blank()),

theme(axis.text.y = element\_blank()),

theme(panel.grid.major.y = element\_blank()),

theme(panel.grid.minor.y = element\_blank()),

# Remove a few things from the x axis and increase font size

theme(axis.title.x = element\_blank()),

theme(panel.grid.major.x = element\_blank()),

theme(axis.text.x.top = element\_text(size=12)),

# Remove x & y tick marks

theme(axis.ticks = element\_blank()),

# Format title & subtitle

theme(plot.title = element\_text(size=14, face = "bold", hjust = 0.5)),

theme(plot.subtitle = element\_text(hjust = 0.5))

)

summary(MySpecial)

## Length Class Mode

## [1,] 17 ScaleDiscretePosition environment

## [2,] 57 theme list

## [3,] 1 theme list

## [4,] 1 theme list

## [5,] 1 theme list

## [6,] 1 theme list

## [7,] 1 theme list

## [8,] 1 theme list

## [9,] 1 theme list

## [10,] 1 theme list

## [11,] 1 theme list

## [12,] 1 theme list

## [13,] 1 theme list

## [14,] 1 theme list

MySpecial is actually an incredibly complex structure so I used the  
summary function. What’s important to us is that in the future all we  
need to do is include it in the ggplot command and magic happens.  
Perhaps another day I’ll make it a proper function but for now I can  
change little things like line size or titles and labels without  
worrying about the rest. So here it is with some little things changed.

ggplot(data = moredata, aes(x = Date, y = Pct, group = Party)) +

geom\_line(aes(color = Party, alpha = 1), size = 1) +

geom\_point(aes(color = Party, alpha = 1), size = 3) +

geom\_text\_repel(data = moredata %>% filter(Date == "11-May-18"),

aes(label = paste0(Party, " : ", Pct, "%")) ,

hjust = "left",

fontface = "bold",

size = 4,

nudge\_x = -.45,

direction = "y") +

geom\_text\_repel(data = moredata %>% filter(Date == "25-May-18"),

aes(label = paste0(Party, " : ", Pct, "%")) ,

hjust = "right",

fontface = "bold",

size = 4,

nudge\_x = .5,

direction = "y") +

MySpecial +

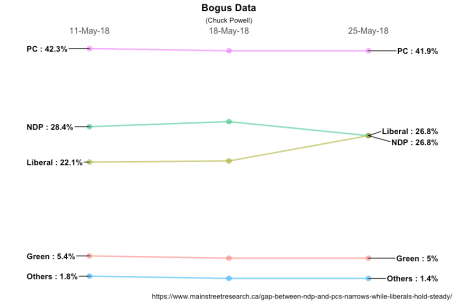
labs(

title = "Bogus Data",

subtitle = "(Chuck Powell)",

caption = "https://www.mainstreetresearch.ca/gap-between-ndp-and-pcs-narrows-while-liberals-hold-steady/"

)



**Even more complex**

**N.B. that the original Tufte is not accurate on the vertical scale.  
Look at Prostate and Thyroid for example since visually I would argue  
they should cross to reflect the data**.

Let’s grab the data as laid out by  
Tufte.

cancer <- structure(list(Year.5 = c(99, 96, 95, 89, 86, 85, 84, 82, 71, 69, 63, 62, 62, 58, 57, 55, 43, 32, 30, 24, 15, 14, 8, 4),

Year.10 = c(95, 96, 94, 87, 78, 80, 83, 76, 64, 57, 55, 54, 55, 46, 46, 49, 32, 29, 13, 19, 11, 8, 6, 3),

Year.15 = c(87, 94, 91, 84, 71, 74, 81, 70, 63, 46, 52, 50, 54, 38, 38, 50, 30, 28, 7, 19, 7, 8, 6, 3),

Year.20 = c(81, 95, 88, 83, 75, 67, 79, 68, 60, 38, 49, 47, 52, 34, 33, 50, 26, 26, 5, 15, 6, 5, 8, 3)),

class = "data.frame",

row.names = c("Prostate", "Thyroid", "Testis", "Melanomas", "Breast", "Hodgkin's", "Uterus", "Urinary", "Cervix", "Larynx", "Rectum", "Kidney", "Colon", "Non-Hodgkin's", "Oral", "Ovary", "Leukemia", "Brain", "Multiple myeloma", "Stomach", "Lung", "Esophagus", "Liver", "Pancreas"))

str(cancer)

## 'data.frame': 24 obs. of 4 variables:

## $ Year.5 : num 99 96 95 89 86 85 84 82 71 69 ...

## $ Year.10: num 95 96 94 87 78 80 83 76 64 57 ...

## $ Year.15: num 87 94 91 84 71 74 81 70 63 46 ...

## $ Year.20: num 81 95 88 83 75 67 79 68 60 38 ...

kable(head(cancer,10)) %>%

kable\_styling(bootstrap\_options = c("striped", "hover", "condensed", "responsive"))

|  | **Year.5** | **Year.10** | **Year.15** | **Year.20** |
| --- | --- | --- | --- | --- |
| Prostate | 99 | 95 | 87 | 81 |
| Thyroid | 96 | 96 | 94 | 95 |
| Testis | 95 | 94 | 91 | 88 |
| Melanomas | 89 | 87 | 84 | 83 |
| Breast | 86 | 78 | 71 | 75 |
| Hodgkin’s | 85 | 80 | 74 | 67 |
| Uterus | 84 | 83 | 81 | 79 |
| Urinary | 82 | 76 | 70 | 68 |
| Cervix | 71 | 64 | 63 | 60 |
| Larynx | 69 | 57 | 46 | 38 |

There, we have it in a neat data frame but not organized as we need it.  
Not unusual, and an opportunity to use some other tools from broom and  
reshape2. Let’s do the following:

1. Let’s transpose the data with t
2. Let’s use broom::fix\_data\_frame to get valid column names and  
   convert rownames to a proper column all in one function. Right now  
   the types of cancer are nothing but rownames.
3. Use reshape2::melt to take our transposed dataframe and convert it  
   to long format so we can send it off to ggplot. Along the way  
   we’ll rename the resulting dataframe newcancer with columns  
   named Year, Type and Survival.

# stepping through for demonstration purposes

t(cancer) # returns a matrix

## Prostate Thyroid Testis Melanomas Breast Hodgkin's Uterus Urinary

## Year.5 99 96 95 89 86 85 84 82

## Year.10 95 96 94 87 78 80 83 76

## Year.15 87 94 91 84 71 74 81 70

## Year.20 81 95 88 83 75 67 79 68

## Cervix Larynx Rectum Kidney Colon Non-Hodgkin's Oral Ovary

## Year.5 71 69 63 62 62 58 57 55

## Year.10 64 57 55 54 55 46 46 49

## Year.15 63 46 52 50 54 38 38 50

## Year.20 60 38 49 47 52 34 33 50

## Leukemia Brain Multiple myeloma Stomach Lung Esophagus Liver

## Year.5 43 32 30 24 15 14 8

## Year.10 32 29 13 19 11 8 6

## Year.15 30 28 7 19 7 8 6

## Year.20 26 26 5 15 6 5 8

## Pancreas

## Year.5 4

## Year.10 3

## Year.15 3

## Year.20 3

broom::fix\_data\_frame(

t(cancer),

newcol = "Year") # make it a dataframe with Year as a proper column

## Year Prostate Thyroid Testis Melanomas Breast Hodgkin.s Uterus

## 1 Year.5 99 96 95 89 86 85 84

## 2 Year.10 95 96 94 87 78 80 83

## 3 Year.15 87 94 91 84 71 74 81

## 4 Year.20 81 95 88 83 75 67 79

## Urinary Cervix Larynx Rectum Kidney Colon Non.Hodgkin.s Oral Ovary

## 1 82 71 69 63 62 62 58 57 55

## 2 76 64 57 55 54 55 46 46 49

## 3 70 63 46 52 50 54 38 38 50

## 4 68 60 38 49 47 52 34 33 50

## Leukemia Brain Multiple.myeloma Stomach Lung Esophagus Liver Pancreas

## 1 43 32 30 24 15 14 8 4

## 2 32 29 13 19 11 8 6 3

## 3 30 28 7 19 7 8 6 3

## 4 26 26 5 15 6 5 8 3

reshape2::melt(

broom::fix\_data\_frame(

t(cancer),

newcol = "Year"),

id="Year", variable.name="Type", value.name = "Survival") # melt it to long form

## Year Type Survival

## 1 Year.5 Prostate 99

## 2 Year.10 Prostate 95

## 3 Year.15 Prostate 87

## 4 Year.20 Prostate 81

## 5 Year.5 Thyroid 96

## 6 Year.10 Thyroid 96

## 7 Year.15 Thyroid 94

## 8 Year.20 Thyroid 95

## 9 Year.5 Testis 95

## 10 Year.10 Testis 94

## 11 Year.15 Testis 91

## 12 Year.20 Testis 88

## 13 Year.5 Melanomas 89

## 14 Year.10 Melanomas 87

## 15 Year.15 Melanomas 84

## 16 Year.20 Melanomas 83

## 17 Year.5 Breast 86

## 18 Year.10 Breast 78

## 19 Year.15 Breast 71

## 20 Year.20 Breast 75

## 21 Year.5 Hodgkin.s 85

## 22 Year.10 Hodgkin.s 80

## 23 Year.15 Hodgkin.s 74

## 24 Year.20 Hodgkin.s 67

## 25 Year.5 Uterus 84

## 26 Year.10 Uterus 83

## 27 Year.15 Uterus 81

## 28 Year.20 Uterus 79

## 29 Year.5 Urinary 82

## 30 Year.10 Urinary 76

## 31 Year.15 Urinary 70

## 32 Year.20 Urinary 68

## 33 Year.5 Cervix 71

## 34 Year.10 Cervix 64

## 35 Year.15 Cervix 63

## 36 Year.20 Cervix 60

## 37 Year.5 Larynx 69

## 38 Year.10 Larynx 57

## 39 Year.15 Larynx 46

## 40 Year.20 Larynx 38

## 41 Year.5 Rectum 63

## 42 Year.10 Rectum 55

## 43 Year.15 Rectum 52

## 44 Year.20 Rectum 49

## 45 Year.5 Kidney 62

## 46 Year.10 Kidney 54

## 47 Year.15 Kidney 50

## 48 Year.20 Kidney 47

## 49 Year.5 Colon 62

## 50 Year.10 Colon 55

## 51 Year.15 Colon 54

## 52 Year.20 Colon 52

## 53 Year.5 Non.Hodgkin.s 58

## 54 Year.10 Non.Hodgkin.s 46

## 55 Year.15 Non.Hodgkin.s 38

## 56 Year.20 Non.Hodgkin.s 34

## 57 Year.5 Oral 57

## 58 Year.10 Oral 46

## 59 Year.15 Oral 38

## 60 Year.20 Oral 33

## 61 Year.5 Ovary 55

## 62 Year.10 Ovary 49

## 63 Year.15 Ovary 50

## 64 Year.20 Ovary 50

## 65 Year.5 Leukemia 43

## 66 Year.10 Leukemia 32

## 67 Year.15 Leukemia 30

## 68 Year.20 Leukemia 26

## 69 Year.5 Brain 32

## 70 Year.10 Brain 29

## 71 Year.15 Brain 28

## 72 Year.20 Brain 26

## 73 Year.5 Multiple.myeloma 30

## 74 Year.10 Multiple.myeloma 13

## 75 Year.15 Multiple.myeloma 7

## 76 Year.20 Multiple.myeloma 5

## 77 Year.5 Stomach 24

## 78 Year.10 Stomach 19

## 79 Year.15 Stomach 19

## 80 Year.20 Stomach 15

## 81 Year.5 Lung 15

## 82 Year.10 Lung 11

## 83 Year.15 Lung 7

## 84 Year.20 Lung 6

## 85 Year.5 Esophagus 14

## 86 Year.10 Esophagus 8

## 87 Year.15 Esophagus 8

## 88 Year.20 Esophagus 5

## 89 Year.5 Liver 8

## 90 Year.10 Liver 6

## 91 Year.15 Liver 6

## 92 Year.20 Liver 8

## 93 Year.5 Pancreas 4

## 94 Year.10 Pancreas 3

## 95 Year.15 Pancreas 3

## 96 Year.20 Pancreas 3

# all those steps in one long line saved to a new dataframe

newcancer <- reshape2::melt(broom::fix\_data\_frame(t(cancer), newcol = "Year"), id="Year", variable.name="Type", value.name = "Survival")

Now we have whipped the data into the shape we need it. 96 rows with the  
three columns we want to plot, Year, Type, and Survival. If you  
look at the data though, you’ll notice two small faults. First, Year  
is not a factor. The plot will work but have an annoying limitation.  
Since “Year.5” is a character string it will be ordered after all the  
other years. We could fix that on the fly within our ggplot call but I  
find it cleaner and more understandable if I take care of that first.  
I’ll use the factor function from base R to accomplish that and  
while I’m at it make the values nicer looking. Second in three cases R  
changed cancer type names because they couldn’t be column names in a  
dataframe. I’ll use forcats::fct\_recode to make them look better.

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)

newcancer$Type <- forcats::fct\_recode(newcancer$Type,

"Hodgkin's" = "Hodgkin.s",

"Non-Hodgkin's" = "Non.Hodgkin.s",

"Multiple myeloma" = "Multiple.myeloma")

head(newcancer)

## Year Type Survival

## 1 5 Year Prostate 99

## 2 10 Year Prostate 95

## 3 15 Year Prostate 87

## 4 20 Year Prostate 81

## 5 5 Year Thyroid 96

## 6 10 Year Thyroid 96

Now that we have the data the way we want it we can make our slopegraph.  
Some of the necessary changes are obvious x = Year, y = Survival and  
group = Type for example. Since there are a lot of plotted lines I’ve  
reduced the weight or size of the individual lines. We no longer want to  
plot the big round points, we’re going to substitute in the actual  
numbers, so that line gets commented out. The left and right labels  
require no change and geom\_text\_repel will keep them from overlapping  
which is almost inevitable given the data. To put the actual survival  
numbers on the plot we’ll turn to geom\_label. It’s like geom\_text  
only it puts a label box around the text. We’ll choose a smallish size,  
minimize the amount of padding, and make the border of the box  
invisible. The end result is what we want. It overlays on top of the  
lines we’ve already plotted and the invisible padding gives us just  
enough room.

ggplot(data = newcancer, aes(x = Year, y = Survival, group = Type)) +

geom\_line(aes(color = Type, alpha = 1), size = 1) +

# geom\_point(aes(color = Type, alpha = .1), size = 4) +

geom\_text\_repel(data = newcancer %>% filter(Year == "5 Year"),

aes(label = Type) ,

hjust = "left",

fontface = "bold",

size = 3,

nudge\_x = -.45,

direction = "y") +

geom\_text\_repel(data = newcancer %>% filter(Year == "20 Year"),

aes(label = Type) ,

hjust = "right",

fontface = "bold",

size = 3,

nudge\_x = .5,

direction = "y") +

geom\_label(aes(label = Survival),

size = 2.5,

label.padding = unit(0.05, "lines"),

label.size = 0.0) +

MySpecial +

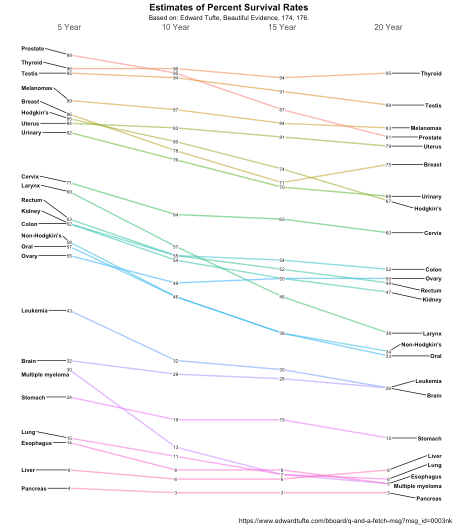
labs(

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"

)



Code Chunks of Slopegraph

Bump\_overlaps.R

|  |
| --- |
| # function for eliminating overlaps |
|  | bump\_overlaps <- function(yvec, decimals = 0L, h = strheight('M')){ |
|  |  |
|  | return(yvec) |
|  |  |
|  | y\_original <- yvec |
|  | yvec <- round(yvec, decimals) |
|  | ord <- order(yvec, decreasing = TRUE) |
|  | yvec <- yvec[ord] |
|  |  |
|  | upper <- yvec + (0.5 \* h) # upper limit of text |
|  | lower <- yvec - (0.5 \* h) # lower limit of text |
|  |  |
|  | is\_overlapped <- function(low, upp, height = h) { |
|  | ifelse(upp != (low + height), low - upp, 0) |
|  | } |
|  | ifna0 <- function(x) { |
|  | ifelse(is.na(x), 0, x) |
|  | } |
|  | bump\_up\_and\_down <- function(v, w, by = 1 \* h, off = 1) { |
|  | out <- v |
|  | w <- w < 0 |
|  | out[w] <- out[w] + by |
|  | nextrow <- c(rep(FALSE, off), head(w, length(w) - off)) |
|  | out[nextrow] <- out[nextrow] - by |
|  | return(out) |
|  | } |
|  |  |
|  | # overlap 3rd label below |
|  | overlap3 <- c(is\_overlapped(lower[1:(length(yvec)-3)], upper[4:(length(yvec))]), rep(0,3)) |
|  | yvec <- bump\_up\_and\_down(yvec, overlap3, by = 0.5 \* h, off = 3) |
|  |  |
|  | # overlap 2nd label below |
|  | upper <- yvec + (0.5 \* h) # upper limit of text |
|  | lower <- yvec - (0.5 \* h) # lower limit of text |
|  | overlap2 <- c(is\_overlapped(lower[1:(length(yvec)-2)], upper[3:(length(yvec))]), rep(0,2)) |
|  | yvec <- bump\_up\_and\_down(yvec, overlap2, by = 0.5 \* h, off = 2) |
|  |  |
|  | # overlap 1st label below |
|  | upper <- yvec + (0.5 \* h) # upper limit of text |
|  | lower <- yvec - (0.5 \* h) # lower limit of text |
|  | overlap1 <- c(is\_overlapped(lower[1:(length(yvec)-1)], upper[2:(length(yvec))]), rep(0,1)) |
|  | yvec <- bump\_up\_and\_down(yvec, overlap1, by = 0.5 \* h, off = 1) |
|  |  |
|  | out <- yvec[order(ord)] |
|  | #print(rbind(y\_original, out)) |
|  | return(out) |
|  | } |

Ggslopegraphs.R

To plot Graphs

|  |
| --- |
| ggslopegraph <- |
|  | function(data, |
|  | main = NULL, |
|  | xlab = "", |
|  | ylab = "", |
|  | xlabels = names(data), |
|  | xlim = c(-1L,ncol(data)+2L), |
|  | ylim = range(data, na.rm = TRUE), |
|  | labpos.left = 0.8, |
|  | labpos.right = ncol(data) + 0.2, |
|  | leftlabels = NULL, |
|  | rightlabels = NULL, |
|  | xbreaks = seq\_along(xlabels), |
|  | ybreaks = NULL, |
|  | yrev = ylim[1] > ylim[2], |
|  | decimals = 0L, |
|  | col.lines = "black", |
|  | col.lab = "black", |
|  | col.num = "black", |
|  | lwd = 0.5, |
|  | offset.x = NULL, |
|  | cex.lab = 3L, |
|  | cex.num = 3L, |
|  | na.span = FALSE) |
|  | { |
|  | # check decimal formatting |
|  | fmt <- paste0("%0.", decimals, "f") |
|  | # check data |
|  | if (ncol(data) < 2) { |
|  | stop("'data' must have at least two columns") |
|  | } |
|  | data[] <- lapply(data, round, decimals) |
|  | # segmentize |
|  | to\_draw <- segmentize(data, na.span = na.span) |
|  |  |
|  | # reshape for printing numeric value labels |
|  | long <- reshape(data, direction = "long", varying = names(data), v.names = "value", sep = "") |
|  |  |
|  | # expand formatting arguments |
|  | if (length(col.lines) == 1) { |
|  | col.lines <- rep(col.lines, nrow(data)) |
|  | } |
|  | if (length(lwd) == 1) { |
|  | lwd <- rep(lwd, nrow(data)) |
|  | } |
|  | if (length(col.num) == 1) { |
|  | col.num <- rep(col.num, nrow(data)) |
|  | } |
|  | if (length(col.lab) == 1) { |
|  | col.lab <- rep(col.lab, nrow(data)) |
|  | } |
|  | col.num <- col.num[long[["id"]]] |
|  | col.lines <- col.lines[to\_draw[["row"]]] |
|  | lwd <- lwd[to\_draw[["row"]]] |
|  |  |
|  | # draw |
|  | g <- ggplot() + |
|  | # x-axis labels |
|  | scale\_x\_continuous(name = xlab, breaks = xbreaks, |
|  | labels = xlabels, limits = xlim) + |
|  | # title |
|  | ggtitle(main) + |
|  | if (isTRUE(yrev)) { |
|  | scale\_y\_reverse(name = ylab, breaks = ybreaks, labels = NULL, limits = rev(ylim)) |
|  | } else { |
|  | scale\_y\_continuous(name = ylab, breaks = ybreaks, labels = NULL, limits = ylim) |
|  | } |
|  |  |
|  | if (is.null(offset.x)) { |
|  | offset.x <- (max(nchar(sprintf(fmt, long[["value"]]))) + 0.02)/2L |
|  | } |
|  |  |
|  | # segments |
|  | g <- g + geom\_segment(aes(x = x1 + offset.x, |
|  | y = ifelse(y1 == y2, y1, (y1+((y2-y1)\*offset.x))), |
|  | xend = x2 - offset.x, |
|  | yend = ifelse(y1 == y2, y2, (y2-((y2-y1)\*offset.x)))), |
|  | col = col.lines, size = lwd, |
|  | data = to\_draw, inherit.aes = FALSE) + guides(fill = FALSE) + |
|  | # numeric value labels |
|  | geom\_text(aes(x = time, y = bump\_overlaps(value), label = sprintf(fmt, value)), color = col.num, |
|  | data = long, inherit.aes = FALSE, |
|  | size = cex.num, hjust = 0.5) |
|  |  |
|  | if (is.null(leftlabels)) { |
|  | leftlabs <- data[!is.na(data[,1]), 1, drop = FALSE] |
|  | } else { |
|  | leftlabs <- leftlabels |
|  | } |
|  | if (is.null(rightlabels)) { |
|  | which\_right <- data[!is.na(data[,ncol(data)]), ncol(data), drop = FALSE] |
|  | } else { |
|  | which\_right <- rightlabels |
|  | } |
|  | # left-side row labels |
|  | if (!is.null(labpos.left)) { |
|  | g <- g + geom\_text(aes(x = labpos.left, y = bump\_overlaps(leftlabs[,1]), |
|  | label = rownames(leftlabs)), |
|  | color = col.lab[!is.na(data[,1])], |
|  | data = NULL, inherit.aes = FALSE, size = cex.lab, hjust = 1L) |
|  | } |
|  | # right-side row labels |
|  | if (!is.null(labpos.right)) { |
|  | g <- g + geom\_text(aes(x = labpos.right, y = bump\_overlaps(which\_right[,1]), |
|  | label = rownames(which\_right)), |
|  | color = col.lab[!is.na(data[,ncol(data)])], |
|  | data = NULL, inherit.aes = FALSE, size = cex.lab, hjust = 0L) |
|  | } |
|  | return(g + theme(legend.position="none") + guides(fill = FALSE)) |
|  | } |
|  |  |
|  | globalVariables(c("x1", "y1", "x2", "y2", "time", "value")) |

Ggslopegrapgs2.R

To plot advanced Graphs

|  |
| --- |
| ggslopegraph2 <- |
|  | function( |
|  | dataframe, |
|  | times, |
|  | measurement, |
|  | grouping, |
|  | title = "", |
|  | subtitle = "", |
|  | caption = "", |
|  | xtextsize = 12, |
|  | ytextsize = 3, |
|  | titletextsize = 14, |
|  | subtitletextsize = 10, |
|  | captiontextsize = 8, |
|  | linethickness = 1, |
|  | linecolor = "ByGroup", |
|  | datatextsize = 2.5, |
|  | datatextfamily = "sans", |
|  | datatextface = "plain", |
|  | labeltextfamily = "sans", |
|  | labeltextface = "bold" |
|  | ) { |
|  | # Since ggplot2 objects are just regular R objects, put them in a list |
|  | my\_special <- list( |
|  | # Format tweaks |
|  | scale\_x\_discrete(position = "top"), # move the x axis labels up top |
|  | theme\_bw(), |
|  | theme(legend.position = "none"), # Remove the legend |
|  | theme(panel.border = element\_blank()), # Remove the panel border |
|  | theme(axis.title.y = element\_blank()), # Remove just about everything from the y axis |
|  | theme(axis.text.y = element\_blank()), |
|  | theme(panel.grid.major.y = element\_blank()), |
|  | theme(panel.grid.minor.y = element\_blank()), |
|  | theme(axis.title.x = element\_blank()), # Remove a few things from the x axis |
|  | theme(panel.grid.major.x = element\_blank()), |
|  | theme(axis.text.x.top = element\_text(size = xtextsize)), # and increase font size |
|  | theme(axis.ticks = element\_blank()), # Remove x & y tick marks |
|  | theme(plot.title = element\_text(size = titletextsize, face = "bold")), # Format title |
|  | theme(plot.title = element\_text(hjust = 0.5)), # Center title & subtitle |
|  | theme(plot.subtitle = element\_text(hjust = 0.5, size = subtitletextsize)), |
|  | theme(plot.caption = element\_text(size = captiontextsize)) |
|  | ) |
|  | # for convenience store these |
|  | Ndataframe <- deparse(substitute(dataframe)) # name of dataframe |
|  | Ntimes <- deparse(substitute(times)) # name of times variable |
|  | Nmeasurement <- deparse(substitute(measurement)) # name of measurement variable |
|  | Ngrouping <- deparse(substitute(grouping)) # name of grouping variable |
|  | # error checking and setup |
|  | if (length(match.call()) <= 4) { |
|  | stop("Not enough arguments passed... requires a dataframe, plus at least three variables") |
|  | } |
|  | argList <- as.list(match.call()[-1]) |
|  | if (!exists(Ndataframe)) { |
|  | stop("The first object in your list '", Ndataframe ,"' does not exist. It should be a dataframe", call. = FALSE) |
|  | } |
|  | if (!is(dataframe, "data.frame")) { |
|  | stop(paste0("'", Ndataframe, "' does not appear to be a data frame"), call. = FALSE) |
|  | } |
|  | if (!Ntimes %in% names(dataframe)) { |
|  | stop(paste0("'", Ntimes, "' is not the name of a variable in '", Ndataframe, "'"), call. = FALSE) |
|  | } |
|  | if (!Nmeasurement %in% names(dataframe)) { |
|  | stop(paste0("'", Nmeasurement, "' is not the name of a variable in '", Ndataframe, "'"), call. = FALSE) |
|  | } |
|  | if (!deparse(substitute(grouping)) %in% names(dataframe)) { |
|  | stop(paste0("'", Ngrouping, "' is not the name of a variable in '", Ndataframe, "'"), call. = FALSE) |
|  | } |
|  | if (!class(dataframe[[Nmeasurement]]) %in% c("integer","numeric")) { |
|  | stop(paste0("Sorry I need the measured variable '", Nmeasurement, "' to be a number"), call. = FALSE) |
|  | } |
|  | if (!"ordered" %in% class(dataframe[[Ntimes]])) { # keep checking |
|  | if (!"character" %in% class(dataframe[[Ntimes]])) { # keep checking |
|  | if ("factor" %in% class(dataframe[[Ntimes]])) { # impose order |
|  | message(paste0("\nConverting '", Ntimes, "' to an ordered factor\n")) |
|  | dataframe[[Ntimes]] <- factor(dataframe[[Ntimes]], ordered = TRUE) |
|  | } else { |
|  | stop(paste0("Sorry I need the variable '", Ntimes, "' to be of class character, factor or ordered"), call. = FALSE) |
|  | } |
|  | } |
|  | } |
|  |  |
|  | times <- enquo(times) |
|  | measurement <- enquo(measurement) |
|  | grouping <- enquo(grouping) |
|  |  |
|  | if (length(linecolor) > 1) { |
|  | if (length(linecolor) < length(unique(dataframe[[Ngrouping]]))) { |
|  | message(paste0("\nYou gave me ", length(linecolor), " colors I'm recycling because you have ", length(unique(dataframe[[Ngrouping]])), " ", Ngrouping, "\n")) |
|  | linecolor <- rep(linecolor, length.out = length(unique(dataframe[[Ngrouping]]))) |
|  | } |
|  | line\_geom <- list(geom\_line(aes\_(color = grouping), size = linethickness), scale\_color\_manual(values = linecolor)) |
|  | } else { |
|  | if (linecolor == "ByGroup") { |
|  | line\_geom <- list(geom\_line(aes\_(color = grouping, alpha = 1), size = linethickness)) |
|  | } else { |
|  | line\_geom <- list(geom\_line(aes\_(), size = linethickness, color = linecolor)) |
|  | } |
|  | } |
|  |  |
|  | dataframe %>% |
|  | filter(!is.na(!! times), !is.na(!! measurement), !is.na(!! grouping)) %>% |
|  | ggplot(aes\_(group=grouping, y=measurement, x=times)) + |
|  | line\_geom + |
|  | geom\_text\_repel(data = dataframe %>% filter(!! times == min(!! times)), |
|  | aes\_(label = grouping) , |
|  | hjust = "left", |
|  | fontface = labeltextface, |
|  | family = labeltextfamily, |
|  | size = ytextsize, |
|  | nudge\_x = -.45, |
|  | direction = "y") + |
|  | geom\_text\_repel(data = dataframe %>% filter(!! times == max(!! times)), |
|  | aes\_(label = grouping), |
|  | hjust = "right", |
|  | fontface = labeltextface, |
|  | family = labeltextfamily, |
|  | size = ytextsize, |
|  | nudge\_x = .5, |
|  | direction = "y") + |
|  | geom\_label(aes\_(label = measurement), size = datatextsize, |
|  | label.padding = unit(0.05, "lines"), label.size = 0.0, |
|  | fontface = datatextface, family = datatextfamily) + |
|  | my\_special + |
|  | labs( |
|  | title = title, |
|  | subtitle = subtitle, |
|  | caption = caption |
|  | ) |
|  | } |

Segmentize.R – For Segmentation

|  |
| --- |
| segmentize <- function(data, na.span = FALSE, na.omit = TRUE) { |
|  |  |
|  | # `pairsmat`: matrix of pairs of adjacent columns in each row |
|  | if (ncol(data) == 2) { |
|  | pairsmat <- matrix(1:2, nrow = 1L) |
|  | } else { |
|  | pairsmat <- embed(seq\_len(ncol(data)), 2)[,2:1] |
|  | } |
|  | # output |
|  | out <- matrix(NA\_real\_, nrow = nrow(data) \* nrow(pairsmat), ncol = 5L) |
|  | k <- 1L |
|  | for (i in seq\_len(nrow(data))) { |
|  | for (j in seq\_len(nrow(pairsmat))) { |
|  | out[k, 1:3] <- c(i, pairsmat[j,1], pairsmat[j,2]) |
|  | if (is.na(data[i, pairsmat[j,1]])) { |
|  | # left value is missing (or both left and right are missing) |
|  | # values are already missing in `out`, so do nothing |
|  | } else if (is.na(data[i, pairsmat[j,2]])) { |
|  | # right value is missing, but left value is present |
|  | if (isTRUE(na.span)) { |
|  | # if segments should span missing values |
|  | # specify the leftmost non-missing value to the right |
|  | #browser() |
|  | # get values for this row/observation |
|  | rowvals <- unlist(data[i, (pairsmat[j,2]):ncol(data), drop = TRUE]) |
|  | if (length(rowvals) > 1) { |
|  | # if right isn't the last value in the row, find the first non-missing |
|  | nexty <- which(!is.na(rowvals)) |
|  | if (length(nexty)) { |
|  | out[k, 3:5] <- c((pairsmat[j,2] + (nexty[1L] - 1)), # x2 |
|  | data[i, pairsmat[j,1]], # y1 |
|  | rowvals[nexty[1L]]) # y2 |
|  | } |
|  | rm(nexty) |
|  | } |
|  | rm(rowvals) |
|  | } |
|  | # otherwise do nothing |
|  | } else { |
|  | # both left and right values are present |
|  | out[k, 4:5] <- c(data[i, pairsmat[j,1]], data[i, pairsmat[j,2]]) |
|  | } |
|  | k <- k + 1L |
|  | } |
|  | } |
|  | # return, optionally dropping missing values |
|  | if (isTRUE(na.omit)) { |
|  | out <- na.omit(out) |
|  | } |
|  | return(setNames(as.data.frame(out), c("row", "x1", "x2", "y1", "y2"))) |
|  | } |

SLopegraph.R

|  |
| --- |
| slopegraph <- function( |
|  | data, |
|  | main = NULL, |
|  | xlab = '', |
|  | ylab = '', |
|  | xlabels = names(data), |
|  | xlim = c(.5,ncol(data)+.5), |
|  | ylim = c(min(data,na.rm=TRUE)-diff(range(data,na.rm=TRUE))/100,max(data,na.rm=TRUE)+diff(range(data,na.rm=TRUE))/100), |
|  | bty = 'n', |
|  | xaxt = 'n', |
|  | yaxt = 'n', |
|  | panel.first = NULL, |
|  | panel.last = NULL, |
|  | labpos.left = 2, |
|  | labpos.right = 4, |
|  | leftlabels = NULL, |
|  | rightlabels = NULL, |
|  | decimals = 0L, |
|  | col.lines = par('fg'), |
|  | col.lab = col.lines, |
|  | col.num = col.lines, |
|  | col.xaxt = par('fg'), |
|  | offset.x = NULL, |
|  | offset.lab = .1, |
|  | cex.lab = 1, |
|  | cex.num = 1, |
|  | family = "serif", |
|  | font.lab = 1, |
|  | font.num = 1, |
|  | lty = par("lty"), |
|  | lwd = par("lwd"), |
|  | mai = NULL, |
|  | na.span = FALSE, |
|  | ...) |
|  | { |
|  | # check decimal formatting |
|  | fmt <- paste0("%0.", decimals, "f") |
|  | # check data |
|  | if (ncol(data) < 2) { |
|  | stop('`data` must have at least two columns') |
|  | } |
|  | data[] <- lapply(data, round, decimals) |
|  | # segmentize |
|  | to\_draw <- segmentize(as.matrix(data)) |
|  | # reshape for printing numeric value labels |
|  | long <- reshape(data, direction = "long", varying = names(data), v.names = "value", sep = "") |
|  |  |
|  | # draw margins |
|  | if (is.null(mai)) { |
|  | op <- par(mai=c(1, 0, if(is.null(main)) 0 else 1, 0)) |
|  | on.exit(par(op)) |
|  | } else { |
|  | op <- par(mai=mai) |
|  | on.exit(par(op)) |
|  | } |
|  | plot(NA, xlim=xlim, ylim=ylim, main=main, family=family, |
|  | bty=bty, yaxt=yaxt, xaxt=xaxt, xlab=xlab, ylab=ylab, ...) |
|  | # optional expression |
|  | if (!is.null(panel.first)) { |
|  | eval(panel.first) |
|  | } |
|  |  |
|  | # expand formatting arguments |
|  | if (length(col.lab) == 1L) { |
|  | col.lab <- rep(col.lab, length.out = nrow(data)) |
|  | } |
|  | if (length(col.num) == 1L) { |
|  | col.num <- rep(col.num, length.out = nrow(data)) |
|  | } |
|  | if (length(col.lines) == 1L) { |
|  | col.lines <- rep(col.lines, length.out = nrow(data)) |
|  | } |
|  | if (length(lwd) == 1) { |
|  | lwd <- rep(lwd, nrow(data)) |
|  | } |
|  | if (length(lty) == 1) { |
|  | lty <- rep(lty, nrow(data)) |
|  | } |
|  | col.num <- col.num[long[["id"]]] |
|  |  |
|  | # x-axis |
|  | axis(1, 1:ncol(data), labels = xlabels, col = col.xaxt, col.ticks = col.xaxt, family = family) |
|  |  |
|  | # left-side labels |
|  | if (!is.null(labpos.left)) { |
|  | if (is.null(leftlabels)) { |
|  | leftlabs <- data[!is.na(data[,1]),1, drop = FALSE] |
|  | } else { |
|  | leftlabs <- leftlabels |
|  | } |
|  | text(1-offset.lab, bump\_overlaps(leftlabs[,1], decimals = decimals), |
|  | col=col.lab[which(!is.na(data[,1]))], rownames(leftlabs), pos=labpos.left, |
|  | cex=cex.lab, font=font.lab, family=family) |
|  | } |
|  | # right-side labels |
|  | if (!is.null(labpos.right)) { |
|  | if (is.null(rightlabels)) { |
|  | rightlabs <- data[!is.na(data[,ncol(data)]), ncol(data), drop = FALSE] |
|  | } else { |
|  | rightlabs <- rightlabels |
|  | } |
|  | text(ncol(data)+offset.lab, bump\_overlaps(rightlabs[,1], decimals = decimals), |
|  | col=col.lab[which(!is.na(data[,ncol(data)]))], rownames(rightlabs), pos=labpos.right, |
|  | cex=cex.lab, font=font.lab, family=family) |
|  | } |
|  |  |
|  | if (is.null(offset.x)) { |
|  | offset.x <- (max(strwidth(sprintf(fmt, long[["value"]]))) + 0.02)/2L |
|  | } |
|  | to\_draw2 <- to\_draw[!duplicated(to\_draw),] |
|  | apply(to\_draw2, 1, function(rowdata){ |
|  | i <- rowdata[1] |
|  | x1 <- rowdata[2] |
|  | x2 <- rowdata[3] |
|  | y1 <- rowdata[4] |
|  | y2 <- rowdata[5] |
|  | # draw lines |
|  | ysloped <- (y2-y1)\*offset.x |
|  | segments(x1+offset.x, if(y1==y2) y1 else (y1+ysloped), |
|  | x2-offset.x, if(y1==y2) y2 else (y2-ysloped), |
|  | col = col.lines[i], |
|  | lty = lty[i], |
|  | lwd = lwd[i]) |
|  | }) |
|  | # numeric value labels |
|  | if (!is.null(col.num)) { |
|  | text(long[["time"]], bump\_overlaps(long[["value"]]), sprintf(fmt, long[["value"]]), |
|  | col = col.num, cex = cex.num, font = font.num, family = family) |
|  | } |
|  | # optional expression |
|  | if (!is.null(panel.last)) { |
|  | eval(panel.last) |
|  | } |
|  | # return `to\_draw` invisibly |
|  | invisible(to\_draw) |
|  | } |