

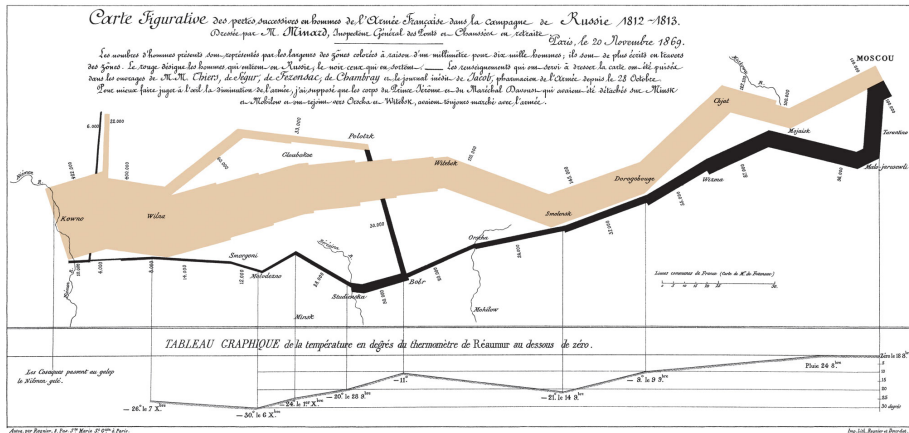
Sankey Plots for Visualising Bilateral Migration

Guy J. Abel

Background

- An alternative approach to visualize bilateral migration are Sankey or alluvial plots.
- Sankey plots feature arrows with width proportional to the flow quantity.
- Named after Irish Captain Sankey, who used to show the energy efficiency of a steam engine in 1898.
- Minard's plot of Napoleon's Russian Campaign of 1812 was made in 1869 - before Sankey
- Alluvial plots are a form of Sankey plot
 - Contain blocks at nodes (e.g. origin and destination of migration flows)
 - No space between blocks, implying a meaningful axis, unlike Sankey plots that do have spaces

Men in Napoleon's 1812 Russian Campaign



Sankey plot of migration in Nature by Butler (2017)

REFUGEES IN FOCUS

THE BIGGEST CONCENTRATIONS
OF DISPLACED PEOPLE LIE FAR
FROM THE SPOTLIGHT.

BY DECLAN BUTLER
DESIGN BY JASKEER SETHI/TORAK

Growing concerns over an 'invasion' of refugees and migrants helped to elect Donald Trump and sway Brexit voters. Yet the data suggest that the situation is very different from how it is often portrayed.

Researchers warn that misleading reports about the magnitude of flows into Europe and the United States are creating unjustified fears about refugees. That is undermining efforts to manage the massive humanitarian problems faced by those fleeing Syria and other hotspots.

The alleged increase in migration and forced displacement tells us more about the moral panic on migration than the reality," says Nando Sigona, a social scientist at the University of Birmingham, UK.

The number of refugees and migrants entering the European Union is low compared with the bloc's population. Nations in Africa and Asia are absorbing many more.

"The number of refugees in Europe is a classic example of perception versus reality," says geographer Nélida Sander at the University of Groningen in the Netherlands.

Experts also question assessments of the global situation. The UN refugee agency (UNHCR) declared in 2015 that the world was "witnessing the highest levels of displacement on record". Around 40 million people were "internally displaced" within their home countries. But researchers say that such estimates are often unreliable.

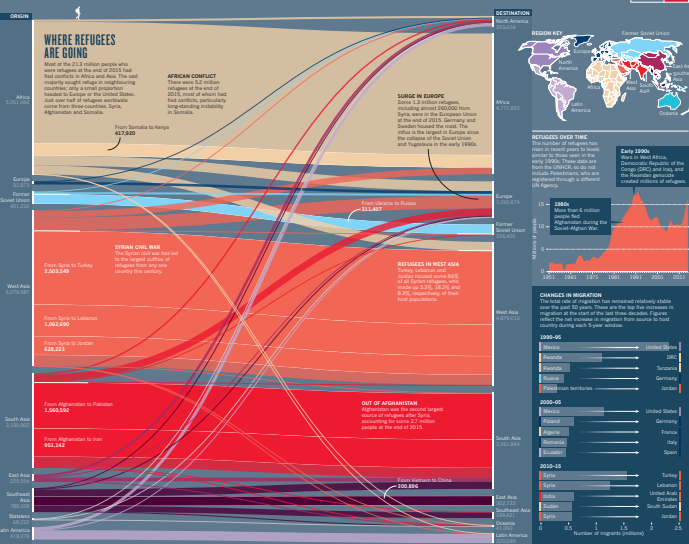
Refugee numbers are easier to track. The UNHCR estimates that there were 21.5 million refugees in 2015; that is only slightly higher than the 1992 figure of 20.6 million, when the global population was just two-thirds of today's.

Researchers also warn about misinterpreting estimates of international migrants — those who move for economic or other reasons. These numbers can be problematic because the most widely cited UN figures are cumulative. Guy Abel, a statistician at the Vienna Institute of Demography, has studied the dynamic flow and found that the number of people migrating has remained stable over the past 50 years. His latest estimates indicate that migration rate, as a share of global population, has dropped to its lowest point in 50 years.

Refugee numbers are easier to track. The UNHCR estimates that there were 21.5 million refugees in 2015; that is only slightly higher than the 1992 figure of 20.6 million, when the global population was just two-thirds of today's.

Researchers also warn about misinterpreting estimates of international migrants — those who move for economic or other reasons. These numbers can be problematic because the most widely cited UN figures are cumulative. Guy Abel, a statistician at the Vienna Institute of Demography, has studied the dynamic flow and found that the number of people migrating has remained stable over the past 50 years. His latest estimates indicate that migration rate, as a share of global population, has dropped to its lowest point in 50 years.

HUMAN MIGRATION
A Nature special issue
nature.com/migration



Sankey plots in R

- As the number of regions or countries increases the plot become more cumbersome
 - Labels for the smaller areas get too small and the plotting area becomes a very long rectangle making it awkward to fit on paper or view on the screen.
 - In such cases I prefer chord diagrams
- There are a few packages in R that have functions for Sankey plots, such as *sankey*, *PantaRhei*, *networkD3*, *sankeywheel*, *plotly*, *ggsankey*.
 - Also *ggalluvial* which produces an alluvial plot, but without any spaces between each sectors.
- I am going to use *ggforce* which I think is the most flexible
 - At the cost of a new layout for the data set
 - Good labels need a some work - as in *circlize* - because Sankey plots tend to have many set axis
 - Migration data tend to have only two set axis (origin and destinations)

Sankey plots in R

- For Sankey plots with *ggforce* the `gather_set_data()` function formats the data so that every migration corridor has two rows for the size of the migration at the origin and destination
- Can then use standard `ggplot()` function to set up the plot format. The mapping argument includes
 - `id` the id of the ribbons
 - `value` the size of the ribbons
 - `split` categories for splitting of the ribbons
- Add on layers for the ribbons themselves using `geom_parallel_sets()`
- Add blocks at the end of the ribbons to allow for clear identification of origin and destinations using `geom_parallel_sets_axes()`
- Add labels at the start and end of the ribbons using `geom_parallel_sets_axes()`

UN international migrant stock data 2020

- United Nations Department of Economic and Social Affairs Population Division (2020) stock data as before

```
> library(tidyverse)
> un <- read_csv(file = "../data/un_desa_ims_tidy.csv")
> un
# A tibble: 259,357 x 6
   year      stock por_name por_code pob_name      pob_code
  <dbl>    <dbl> <chr>    <dbl> <chr>    <dbl>
1  1990  152986157 WORLD      900 WORLD      900
2  1995  161289976 WORLD      900 WORLD      900
3  2000  173230585 WORLD      900 WORLD      900
4  2005  191446828 WORLD      900 WORLD      900
5  2010  220983187 WORLD      900 WORLD      900
6  2015  247958644 WORLD      900 WORLD      900
7  2020  280598105 WORLD      900 WORLD      900
8  1990   15334807 WORLD      900 Sub-Saharan Africa  947
9  1995   16488973 WORLD      900 Sub-Saharan Africa  947
10 2000   15638014 WORLD      900 Sub-Saharan Africa  947
# ... with 259,347 more rows
```

UN international migrant stock data 2020

- Plot between World Bank income groups

```
> # codes for income groups
> cc <- c(1503:1500, 2003)
> d <- un %>%
+   filter(por_code %in% cc,
+         pob_code %in% cc,
+         year == 2020) %>%
+   rename(orig = pob_name,
+         dest = por_name) %>%
+   mutate(stock = stock/1e6)
> d
# A tibble: 16 x 6
  year stock dest por_code orig pob_code
<dbl> <dbl> <chr> <dbl> <chr> <dbl>
1 2020 45.8 High-income countries 1503 High-income cou~ 1503
2 2020 59.9 High-income countries 1503 Upper-middle-in~ 1502
3 2020 58.0 High-income countries 1503 Lower-middle-in~ 1501
4 2020 10.5 High-income countries 1503 Low-income coun~ 1500
5 2020 5.66 Upper-middle-income countries 1502 High-income cou~ 1503
6 2020 20.6 Upper-middle-income countries 1502 Upper-middle-in~ 1502
7 2020 18.3 Upper-middle-income countries 1502 Lower-middle-in~ 1501
8 2020 10.8 Upper-middle-income countries 1502 Low-income coun~ 1500
9 2020 0.961 Lower-middle-income countries 1501 High-income cou~ 1503
10 2020 6.45 Lower-middle-income countries 1501 Upper-middle-in~ 1502
```


Data format

- Format data for Sankey plot using `gather_set_data()` function in *ggforce*

```
> library(ggforce)
>
> s <- d %>%
+   select(orig, dest, stock) %>%
+   gather_set_data(x = 1:2)
> s
```

A tibble: 32 x 6

	orig <chr>	dest <chr>	stock <dbl>	id x <int>	y <chr>
1	High-income countries	High-income c~	45.8	1	orig High-income ~
2	Upper-middle-income countries	High-income c~	59.9	2	orig Upper-middle~
3	Lower-middle-income countries	High-income c~	58.0	3	orig Lower-middle~
4	Low-income countries	High-income c~	10.5	4	orig Low-income c~
5	High-income countries	Upper-middle~	5.66	5	orig High-income ~
6	Upper-middle-income countries	Upper-middle~	20.6	6	orig Upper-middle~
7	Lower-middle-income countries	Upper-middle~	18.3	7	orig Lower-middle~
8	Low-income countries	Upper-middle~	10.8	8	orig Low-income c~
9	High-income countries	Lower-middle~	0.961	9	orig High-income ~
10	Upper-middle-income countries	Lower-middle~	6.45	10	orig Upper-middle~

... with 22 more rows

Data format

```
> tail(s)
```

```
# A tibble: 6 x 6
```

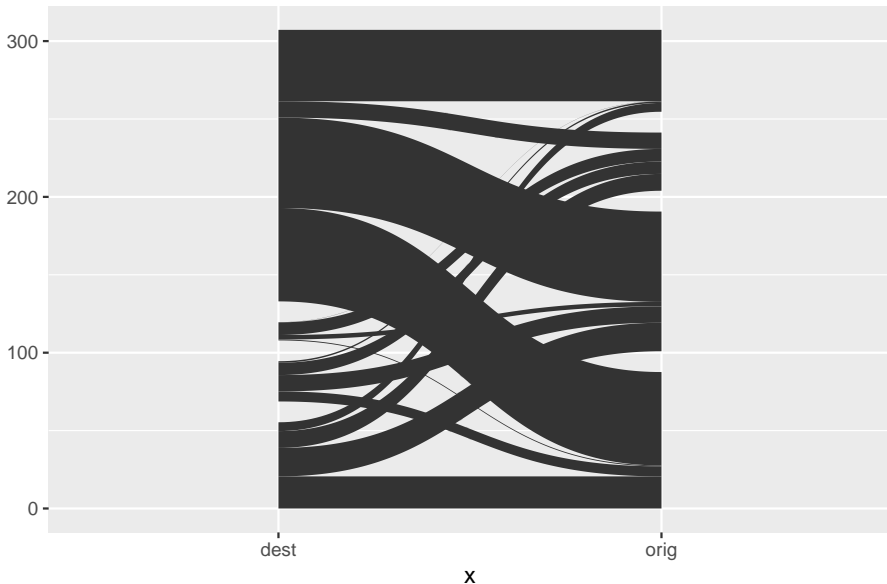
	orig <chr>	dest <chr>	stock <dbl>	id x <int>	y <chr>
1	Lower-middle-income countries	Lower-middle-	10.5	11	dest Lower-middle-
2	Low-income countries	Lower-middle-	7.93	12	dest Lower-middle-
3	High-income countries	Low-income co~	0.102	13	dest Low-income co~
4	Upper-middle-income countries	Low-income co~	0.579	14	dest Low-income co~
5	Lower-middle-income countries	Low-income co~	2.90	15	dest Low-income co~
6	Low-income countries	Low-income co~	8.12	16	dest Low-income co~

Default Plot

- Pass the different columns to `ggplot()` mappings
- The `geom_parallel_sets()` plots the ribbons

```
> ggplot(data = s,  
+       mapping = aes(x = x, id = id, value = stock, split = y)) +  
+       geom_parallel_sets()
```

Default Plot

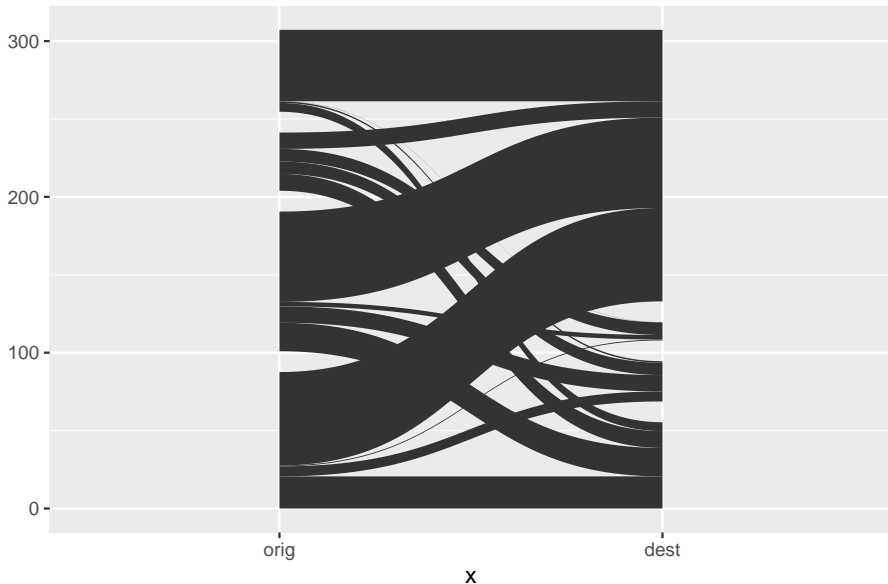


Default Plot

- By default the x-axis goes in alphabetical order
 - Use factors to set ordering of categorical variable

```
> levels(s$x)
NULL
> s <- mutate(s, x = fct_rev(x))
> levels(s$x)
[1] "orig" "dest"
>
> ggplot(data = s,
+       mapping = aes(x = x, id = id, value = stock, split = y)) +
+   geom_parallel_sets()
```

Default Plot

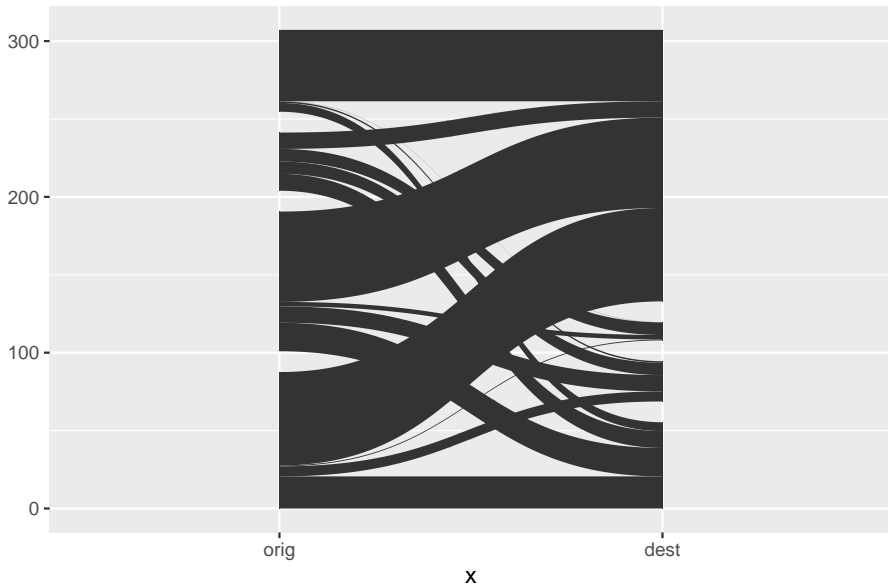


Set Axes

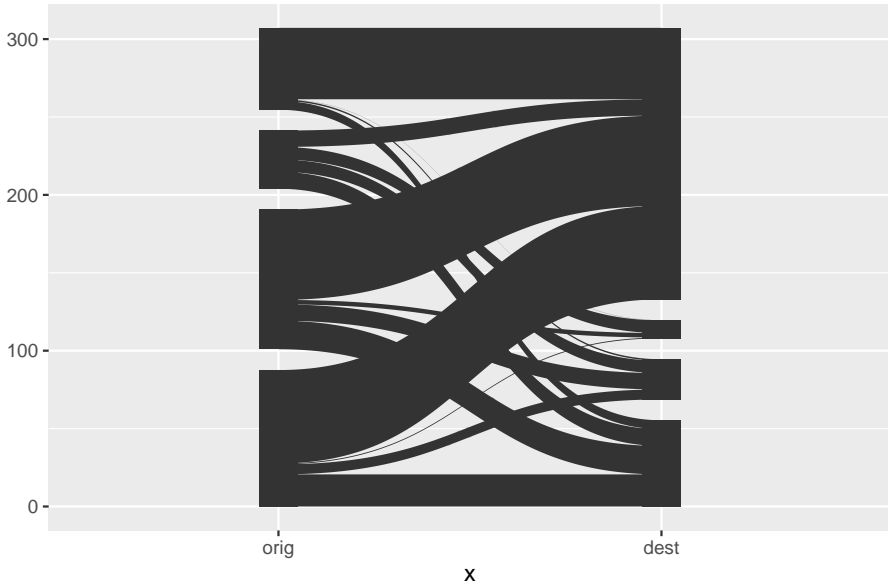
- The `geom_parallel_sets_axes()` function adds blocks besides the start and end of the ribbons
 - Set the width (as a proportion) using `axis.width`

```
> # default axis.width
> ggplot(data = s,
+       mapping = aes(x = x, id = id, value = stock, split = y)) +
+   geom_parallel_sets() +
+   geom_parallel_sets_axes()
>
> # wider axis.width
> ggplot(data = s,
+       mapping = aes(x = x, id = id, value = stock, split = y)) +
+   geom_parallel_sets() +
+   geom_parallel_sets_axes(axis.width = 0.1)
```

Set Axes



Set Axes



Colour

- Use mapping in `geom_parallel_sets()` to set the colours
 - Fill the colours following the origin regions, as was the case in the chord diagrams
- The `geom_parallel_sets_axes()` cannot take a fill colour from the data frame

```
> # geom_parallel_sets_axes cannot take fill colours from data
```

```
> ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y, fill = orig)) +
```

```
+   geom_parallel_sets() +
```

```
+   geom_parallel_sets_axes()
```

```
Warning: Computation failed in `stat_parallel_sets_axes()`:
```

```
Axis aesthetics must be constant in each split
```

```
>
```

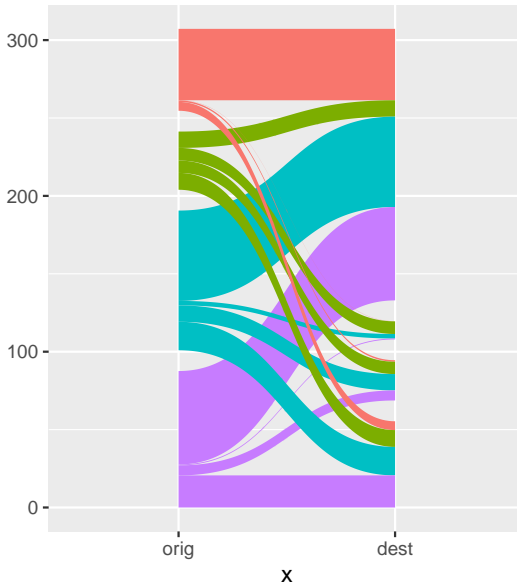
```
> # set fill colour for parallel_sets only
```

```
> ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
```

```
+   geom_parallel_sets(mapping = aes(fill = orig)) +
```

```
+   geom_parallel_sets_axes()
```

Ribbon colour - failed axis colour



orig



High-income countries



Low-income countries

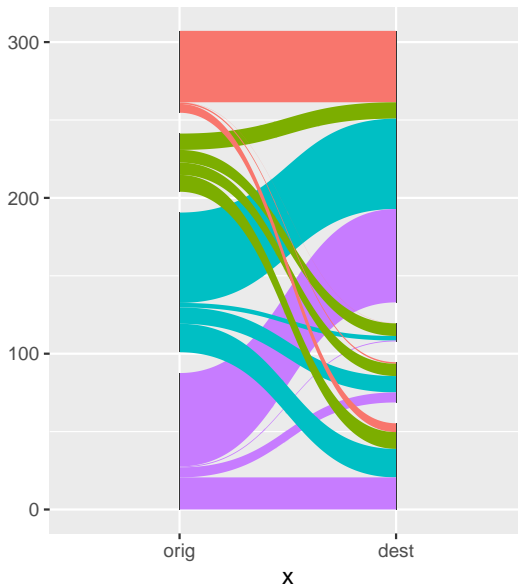


Lower-middle-income countries



Upper-middle-income countries

Ribbon colour

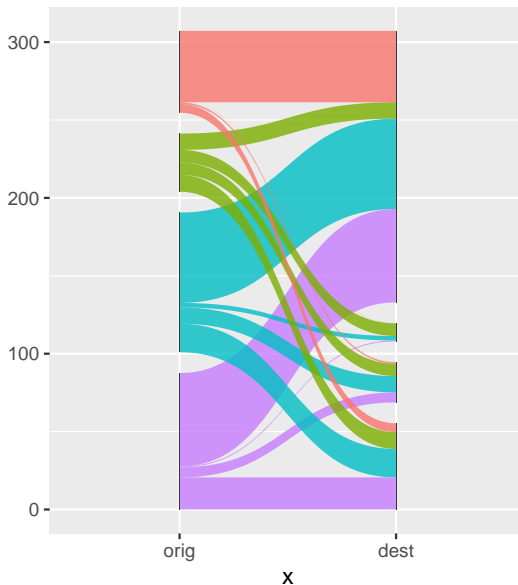


Ribbon transparency

- Add some transparency in the ribbons using the `alpha` argument in `geom_parallel_sets()`

```
> # transparency of 0.8
> ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
+   geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8) +
+   geom_parallel_sets_axes()
>
> # transparency of 0.2
> ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
+   geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.2) +
+   geom_parallel_sets_axes()
```

Ribbon colour



orig



High-income countries



Low-income countries

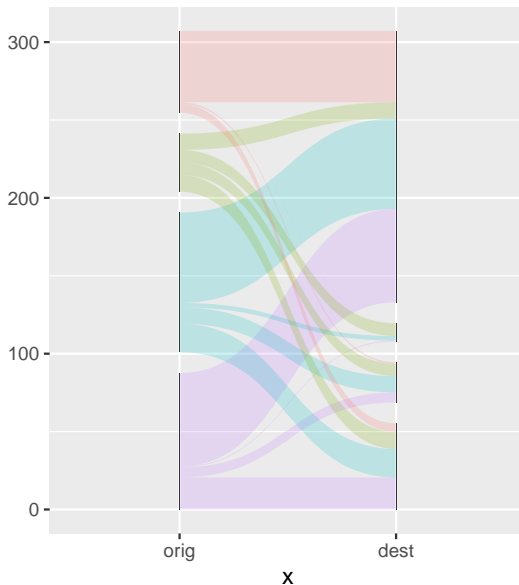


Lower-middle-income countries



Upper-middle-income countries

Ribbon colour

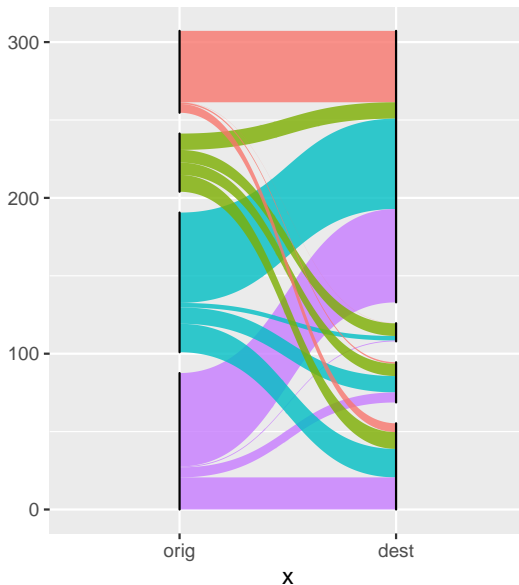


Axis colour

- To see the set axis colours we can draw an outline using the colour argument.
- Also set `fill = "transparent"` in order to view the underlying ribbons

```
> # geom_parallel_sets_axes is an axis, can provide outline colour only
> ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
+   geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8) +
+   geom_parallel_sets_axes(colour = "black")
>
> # geom_parallel_sets_axes is an axis, can provide outline colour only
> ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +
+   geom_parallel_sets(mapping = aes(fill = orig)) +
+   geom_parallel_sets_axes(fill = "transparent", colour = "black",
+                             axis.width = 0.1)
```


Axis colour



orig



High-income countries



Low-income countries

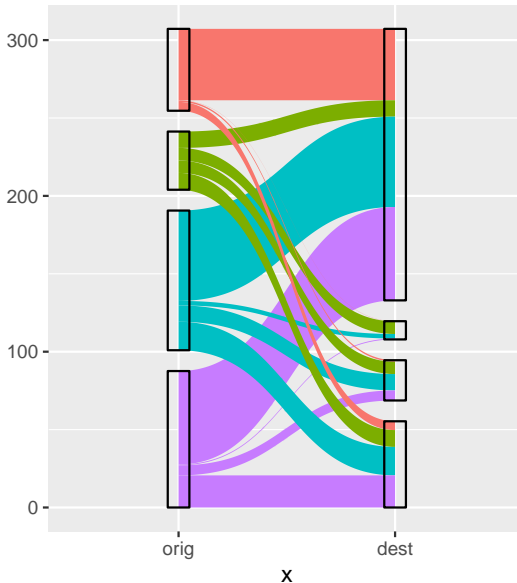


Lower-middle-income countries



Upper-middle-income countries

Axis colour



orig



High-income countries



Low-income countries



Lower-middle-income countries



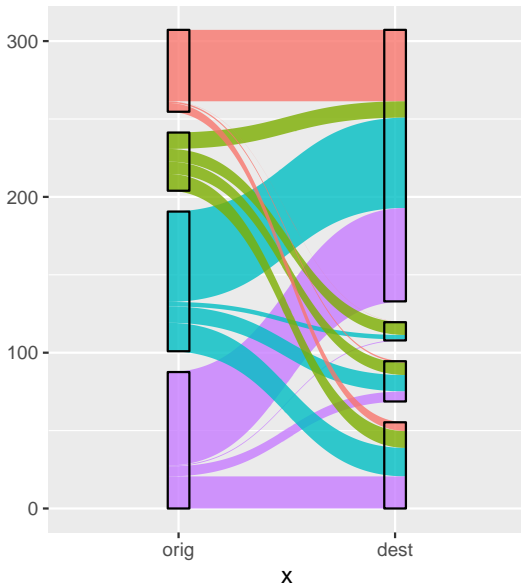
Upper-middle-income countries

Axis colour

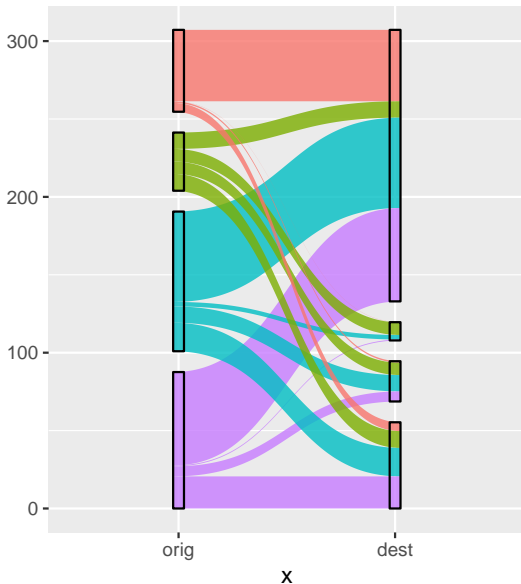
- Tweak the width in `geom_parallel_sets()` so that it fills into the axis box
 - Need to set `fill = "transparent"`

```
> ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +  
+   geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8, axis.width = -0.1)  
+   geom_parallel_sets_axes(fill = "transparent", colour = "black",  
+                           axis.width = 0.1)  
>  
> # narrower set axes  
> ggplot(data = s, mapping = aes(x = x, id = id, value = stock, split = y)) +  
+   geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8, axis.width = -0.05)  
+   geom_parallel_sets_axes(fill = "transparent", colour = "black",  
+                           axis.width = 0.05)
```

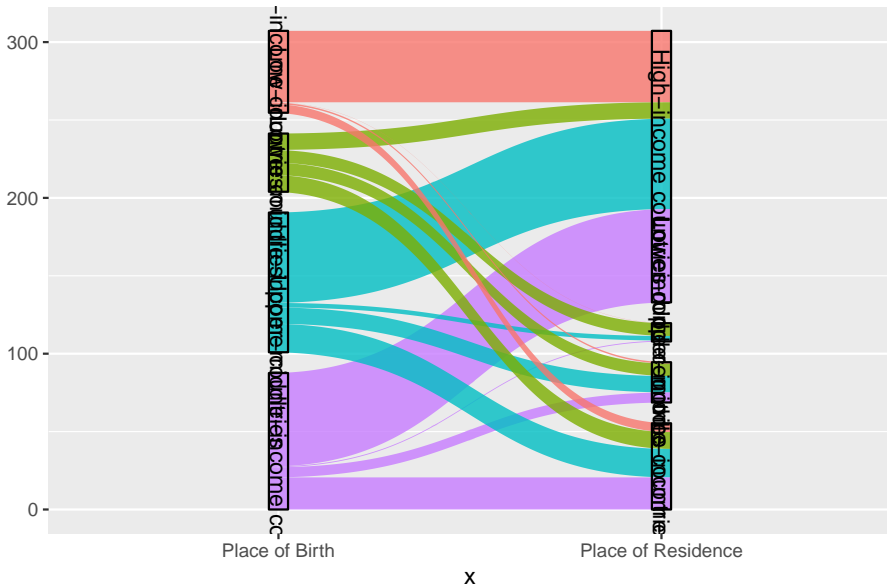
Axis colour



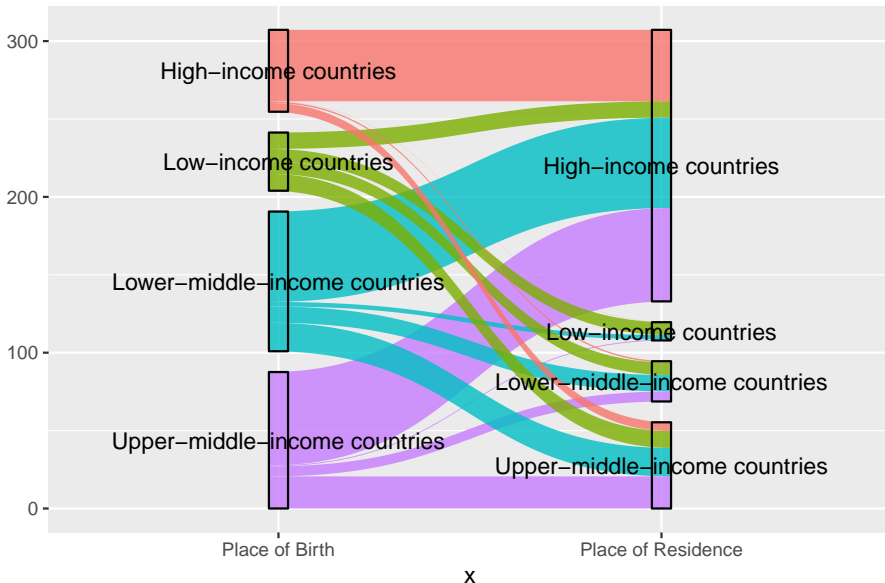
Axis colour



Default labels



Labels with angle = 0



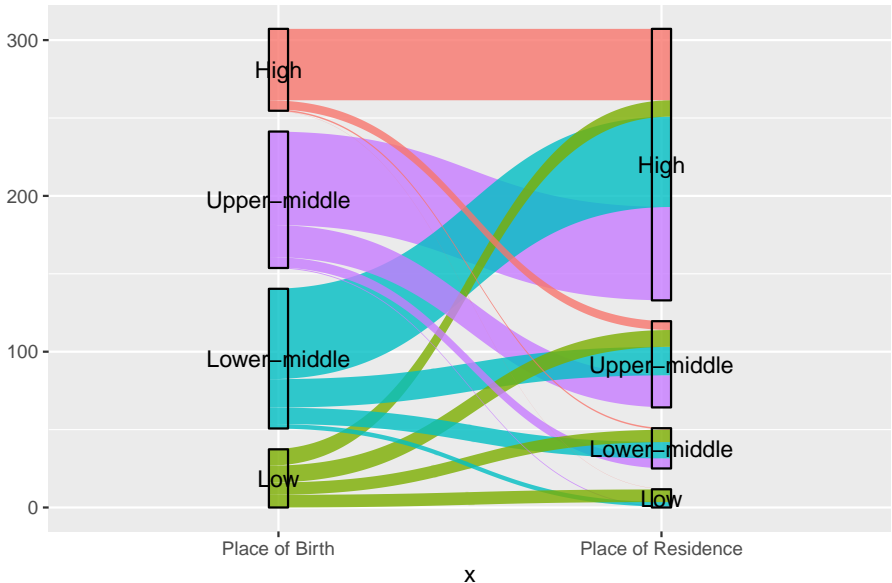
Labels

- Change order of origin and destinations by modifying the levels of the factors
 - Set levels to order they appear in the y column using `fct_inorder()` in the *forcats* package
 - Remove unnecessary parts in the label

```
> levels(s$y)
NULL
> s <- s %>%
+   mutate(y = str_remove(string = y, pattern = "-income countries"),
+          y = fct_inorder(y))
> levels(s$y)
[1] "High"          "Upper-middle" "Lower-middle" "Low"
> s
# A tibble: 32 x 6
```

	orig	dest	stock	id	x	y
	<chr>	<chr>	<dbl>	<int>	<fct>	<fct>
1	High-income countries	High-income countr~	45.8	1	orig	High
2	Upper-middle-income countries	High-income countr~	59.9	2	orig	Upper-m~
3	Lower-middle-income countries	High-income countr~	58.0	3	orig	Lower-m~
4	Low-income countries	High-income countr~	10.5	4	orig	Low
5	High-income countries	Upper-middle-incom~	5.66	5	orig	High
6	Upper-middle-income countries	Upper-middle-incom~	20.6	6	orig	Upper-m~
7	Lower-middle-income countries	Upper-middle-incom~	18.3	7	orig	Lower-m~
8	Low-income countries	Upper-middle-incom~	10.8	8	orig	Low
9	High-income countries	Lower-middle-incor~	0.061	9	orig	High

New, shorter labels



Labels

- Set up a label data frame to adjust position and alignment

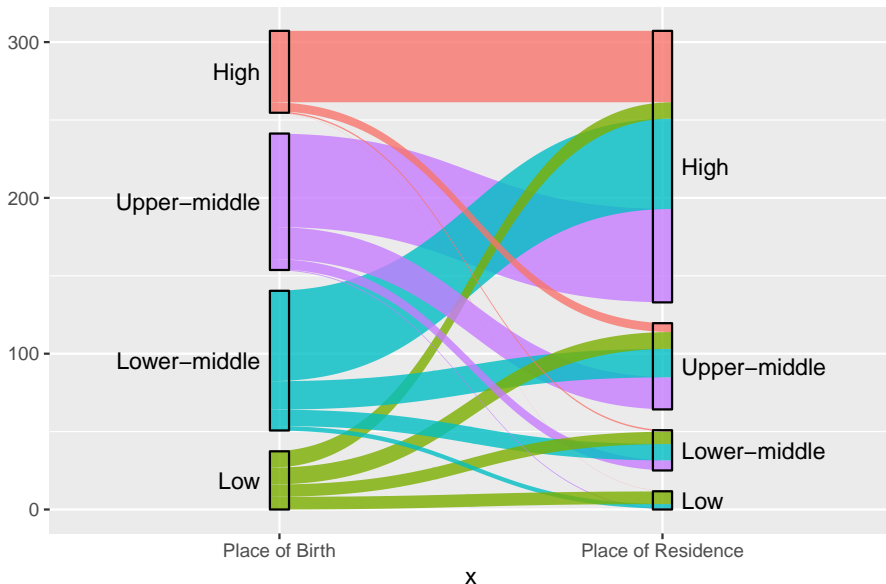
```
> p <- s %>%
+   distinct(x, y) %>%
+   mutate(h = as.numeric(x == "orig"),
+          n = h * -0.1 + 0.05)
```

```
> p
```

```
# A tibble: 8 x 4
```

	x	y	h	n
	<fct>	<fct>	<dbl>	<dbl>
1	orig	High	1	-0.05
2	orig	Upper-middle	1	-0.05
3	orig	Lower-middle	1	-0.05
4	orig	Low	1	-0.05
5	dest	High	0	0.05
6	dest	Upper-middle	0	0.05
7	dest	Lower-middle	0	0.05
8	dest	Low	0	0.05

Looking good

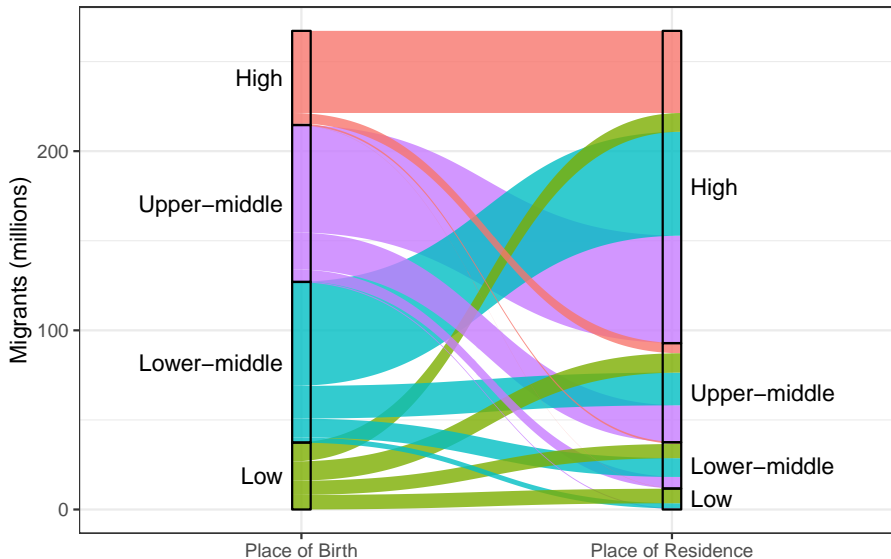


Spacing

- We convert the Sankey plot to an alluvial plot by reducing the space separating the parallel sets to zero via the `sep` argument
 - Need to set `sep` in all the geom functions for alignment.
 - Default is `sep = 0.05` (5%)
 - Might need to reduce when have many regions
- In alluvial plots the y-axis are more meaningful
 - Add y-axis labels via `labs()` function
- Set background to white using `theme_bw()` function

```
> ggplot(data = s,
+       mapping = aes(x = x, id = id, value = stock, split = y)) +
+   geom_parallel_sets(mapping = aes(fill = orig), alpha = 0.8,
+   axis.width = -0.05, sep = 0) +
+   geom_parallel_sets_axes(fill = "transparent", colour = "black",
+   axis.width = 0.05, sep = 0) +
+   guides(fill = "none") +
+   geom_parallel_sets_labels(angle = 0, hjust = p$h,
+   position = position_nudge(x = p$n, ), sep = 0) +
+   scale_x_discrete(labels = c(orig = "Place of Birth",
+   dest = "Place of Residence")) +
+   labs(y = "Migrants (millions)", x = "") +
+   theme_bw()
```

Alluvial plot



Exercise (ex9.R)

```
# 0. a) Load the KOSTAT2021.Rproj file.
#      Run the getwd() below. It should print the directory where the
#      KOSTAT2021.Rproj file is located.
getwd()
#      b) Load the packages used in this exercise
library(tidyverse)
library(ggforce)
##
##
##
##
# 1. Run the code below to read in the migrant stock data from Gabon taken
#      from Table 21-6 in Shryock & Siegel (1979)
ga <- read_csv("./data/gabon_1961_tidy.csv")
ga
# 2. Run the code below to remove the totals groups and migrants from abroad
d <- ga %>%
  rename(orig = place_of_birth,
         dest = place_of_enumeration) %>%
  filter(sex == "total",
         !orig %in% c("Grand total", "Abroad", "Total Gabon"),
         dest != "Total") %>%
  select(-sex)
d
# 3. Create a data frame s1 using the gather set data() function to organise the
```

References

Butler, Declan. 2017. "What the numbers say about refugees." *Nature* 543 (7643): 22–23.
<https://doi.org/10.1038/543022a>.

United Nations Department of Economic and Social Affairs Population Division. 2020. "International Migrant Stock 2020 (United Nations database, POP/DB/MIG/Stock/Rev.2020)." New York, New York, USA: United Nations Department of Economic; Social Affairs/Population Division.
<https://doi.org/10.18356/b4899381-en>.