Introduction_to_settleR

Introduction

Background and Motivation

I had a need to convey combinations of sets and found the the usual method to do this, venn-diagrams or Euler plots too limiting. My research invovles identifying differentially expressed genes which involved looking large numbers of datasets which could be grouped any number of ways. I found UpSet plots to offer a succint way to visualize these comparisons which is relatively intuitive to my audience. There is already an UpSet package, however, I found it was not amenable to customization. In my experience, it is very difficult to write re-usable, modular code creating plots for anything besides one-off functions that will be used once, or at most a single context.

settleR: A simple and customizable package for creating UpSet plots

I created the settleR package to to address the percieved challenges primarily for my own needs, but I hope that others may find it useful too. The goal of settleR is to:

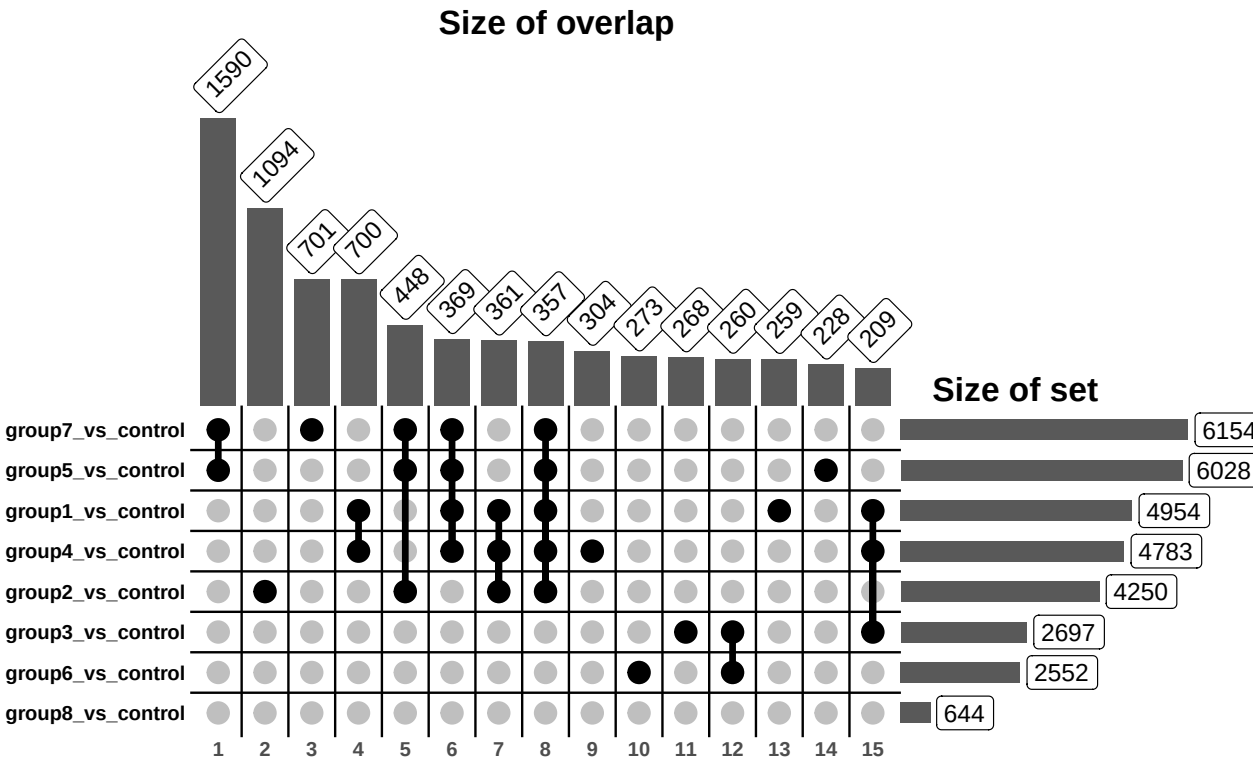
1. Create a simple/modular framework for UpSet-like figures
2. Make creating figures as painless as possible
3. Allow for arbitrary customization

```
library(settleR)
#> Loading required package: dplyr
#>
#> 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
#> Loading required package: ggplot2
#> Loading required package: tibble
library(grid)
```

Creating a basic plot

```
# Loads an example setlist
gene_setlist <-
  system.file('extdata','ex_gene_setlist.rds', package = 'settleR') %>%
```

```
  readRDS(.)

# Creating the SettleR object
setobj <- SettleR(gene_setlist)
#> Warning: Vectorized input to `element_text()` is not officially supported.
#> Results may be unexpected or may change in future versions of ggplot2.

# Plotting to the device
settleR_plot(setobj)
```



Recommend way of saving a plot

You're free to save the plot anyway you like, which is likely different depending on your workflow. I personally end up spending a great deal of time perfecting the dimensions of my plots. The settleR plots are no different. However, I've managed to automate this through trial & error for these plots. Once the plot is to your liking, I recommend using the `settleR_save` to save the plot.

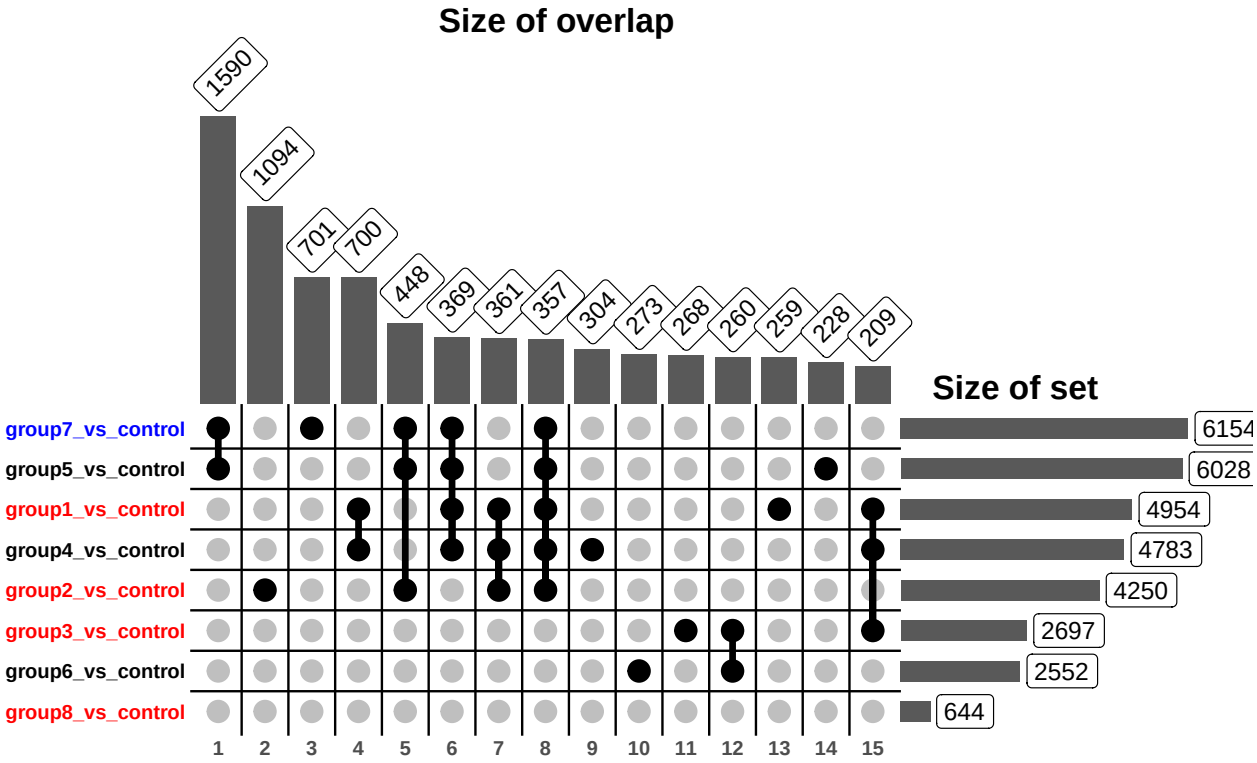
```
# Recommended way to save the plot
settleR_save(setobj, 'example_settleR_plot.pdf')
```

settleR plot customizations

Adding color to the groups

You can create a `SettleR` plot with colour coded names on the left side of the grid plot. To do this, pass a named vector of colours which match the names of the groups. This can be done by either passing the `colMap` parameter while the object is initilized, or after.

```
col_map <-
  system.file('extdata','ex_col_map.rds', package = 'settleR') %>%
  readRDS(.)


# Creating the SettleR object w/ colors
setobj <- SettleR(gene_setlist, colMap = col_map)
#> Warning: Vectorized input to `element_text()` is not officially supported.
#> Results may be unexpected or may change in future versions of ggplot2.

# Changning the colors on a created object
colMap(setobj) <- col_map
#> Warning: Vectorized input to `element_text()` is not officially supported.
#> Results may be unexpected or may change in future versions of ggplot2.

# Plotting to the device
settleR_plot(setobj)
```

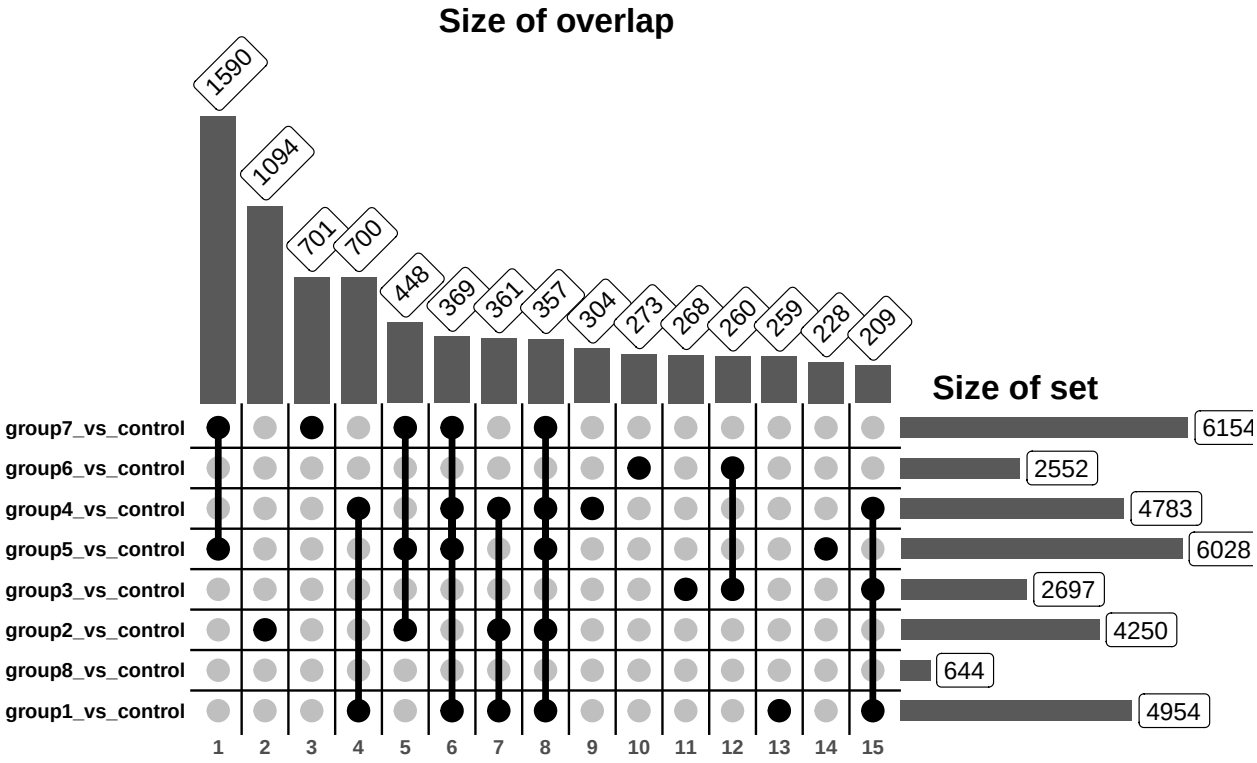


Specifying the order of the groups

This is an example of how you can control the order of the groups in the plot.

```
set_levels <-
  system.file('extdata','ex_set_levels.rds', package = 'settleR') %>%
  readRDS(.)

# Creating the SettleR object ordered groups
setobj <- SettleR(gene_setlist, setLevels = set_levels)
```

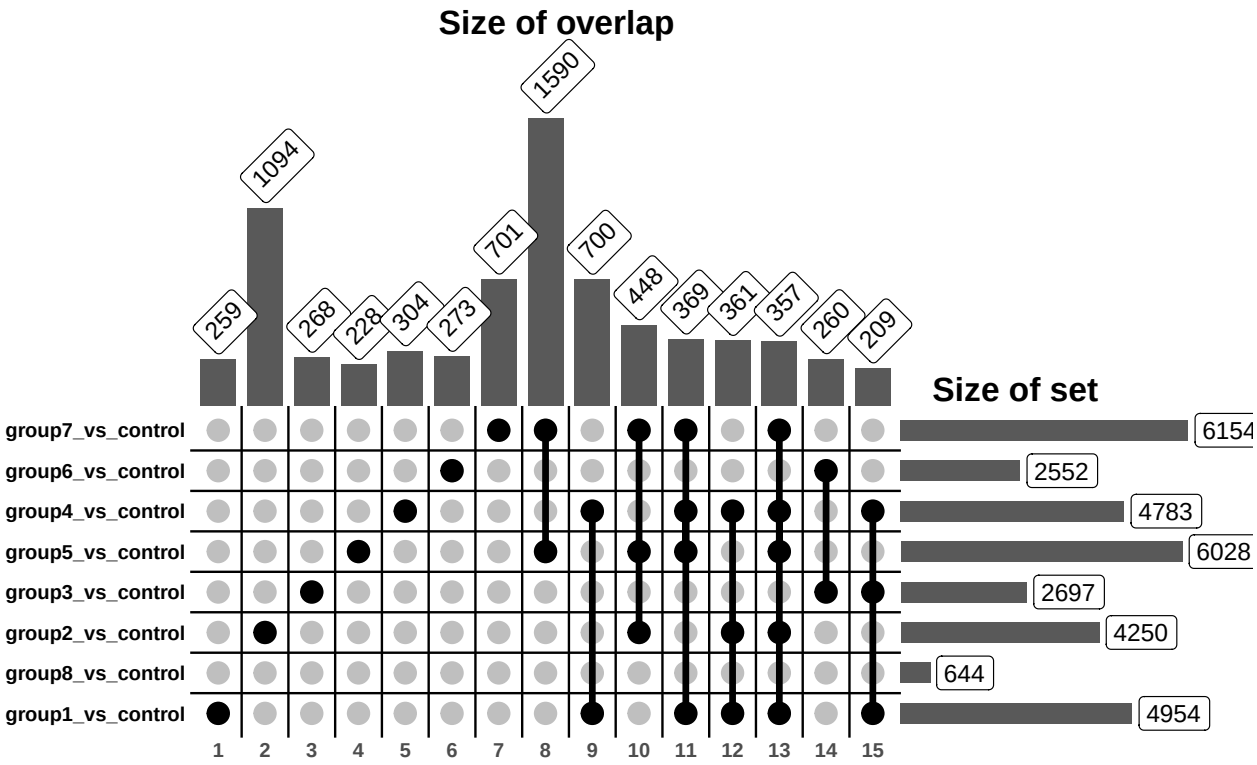
```
#> Warning: Vectorized input to `element_text()` is not officially supported.
#> Results may be unexpected or may change in future versions of ggplot2.

# Changning group order on a created object
setLevels(setobj) <- set_levels
#> Warning: Vectorized input to `element_text()` is not officially supported.
#> Results may be unexpected or may change in future versions of ggplot2.


# Plotting to the device
settleR_plot(setobj)
```



Place the singleton intersects first

It can be helpful to see which items are exclusive to a given group. I've dubbed these singletons, and can be placed in the front of the plot.

```
setobj <- SettleR(gene_setlist, setLevels = set_levels)
#> Warning: Vectorized input to `element_text()` is not officially supported.
#> Results may be unexpected or may change in future versions of ggplot2.
setobj <- reorder_by_singletons(setobj)
#> Warning: Vectorized input to `element_text()` is not officially supported.
#> Results may be unexpected or may change in future versions of ggplot2.
settleR_plot(setobj)
```

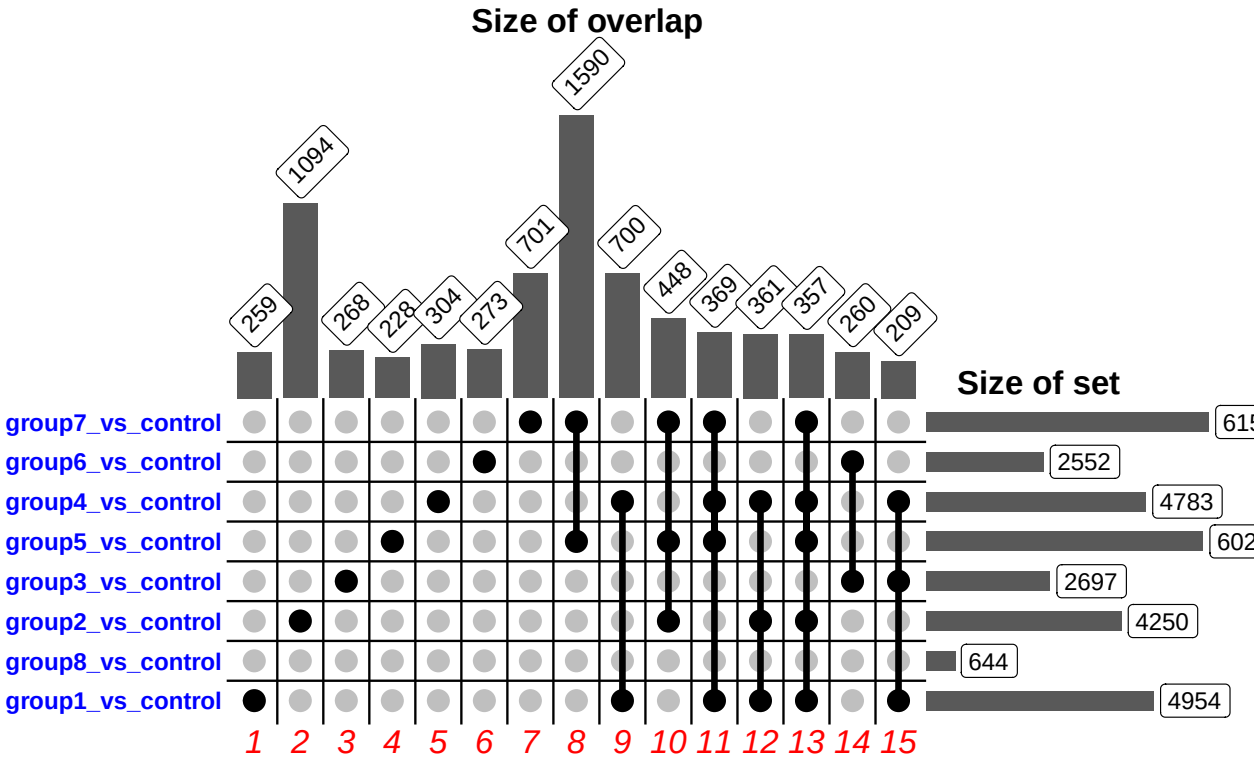


Advanced customizations

Modifications of the underlying ggplots

This is where `settleR` really shines. As described in the intro, I made this package with modularity in mind. Below is an example of how you can modify one of the underlying `ggplots`, then put this back into the `SettleR` object and then make a new `settleRPlot`. A simple example for illustrative purposes is shown below.

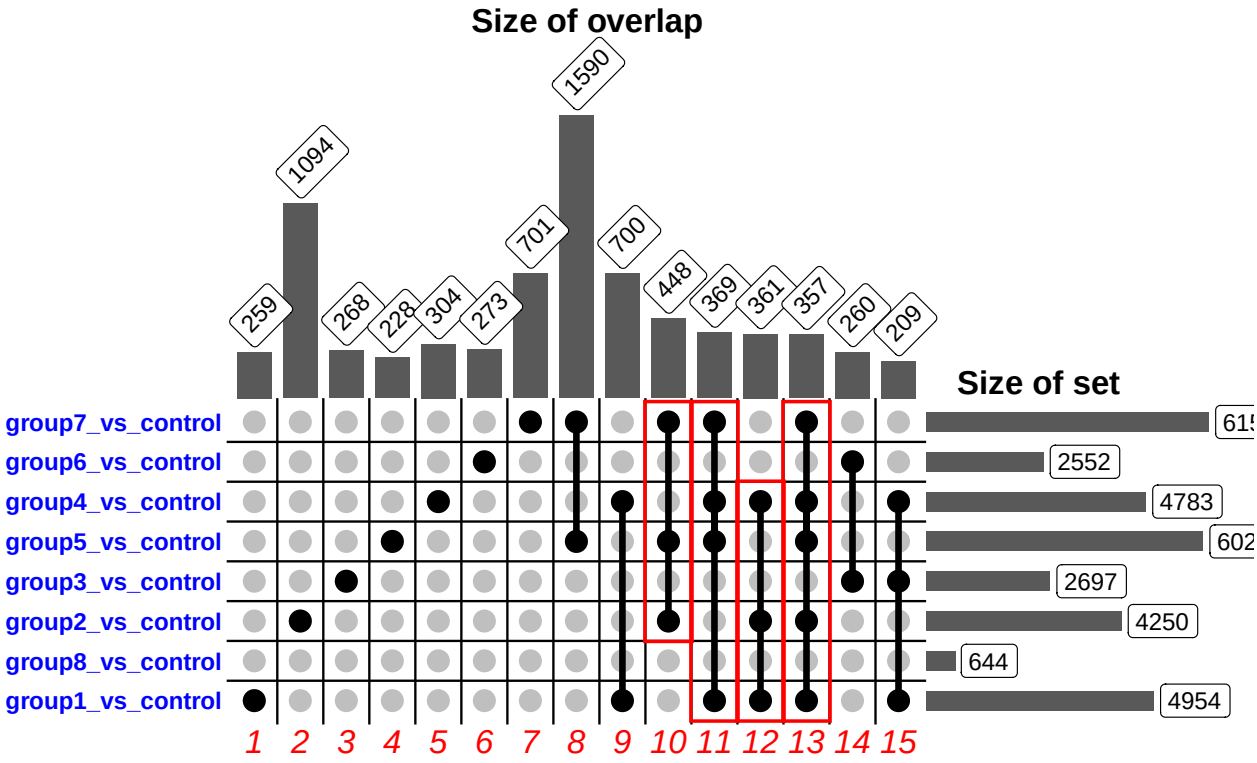
```
# Getting the grid plot
tmp_plt <- gridPlot(setobj)
# Making modifications to the text using "theme" from ggplot2
tmp_plt <- tmp_plt +
  theme(axis.text.y = element_text(colour = 'blue', face="bold", size = rel(.875)),
        axis.text.x = element_text(colour='red', face='italic', size = rel(1.25)))
# Updating the SettleR object with the modified plot
gridPlot(setobj) <- tmp_plt
settleR_plot(setobj)
```



Adding boxes around specific groups

Here's a more complicated example. I had a need of highlighting specific intersects on the plot. The solution I cam up with is to draw a red box around the intersect(s) of interest. The function to include the dimensions is included in `settleR`.

```
# Extracting the grid plot and intersect levels
tmp_plt <-  gridPlot(setobj)
intersect_lvls <- intersectLevels(setobj)

# Getting the dimensions for the box
bound_boxes <- intersect_lvls[c(10:13)] %>%
  box_intercepts_dims(tmp_plt, .)

# Using geom_rect to draw red boxes
tmp_plt <- tmp_plt +
  geom_rect(data=bound_boxes,aes(xmin=xmin, xmax=xmax, ymin=ymin,ymax=ymax),
            inherit.aes = FALSE,
            color='red',
            fill=NA,
            size=.75
  )
# Replacing the modified gridPlot
gridPlot(setobj) <- tmp_plt
settleR_plot(setobj)
```


Size of overlap
259
1094
268
228
304
273
701
1590
700
448
369
361
357
260
209
Size of set
group7_vs_control
group6_vs_control
group4_vs_control
group5_vs_control
group3_vs_control
group2_vs_control
group8_vs_control
group1_vs_control
2552
4783
2697
4250
644
4954
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15