tidyverse

immediate

April 30, 2018

Contents

Interaction with the tidyverse and ggplot2

The tidyverse, ggplot2, and destiny are a great fit!

```
In [2]: suppressPackageStartupMessages({
            library(destiny)
            library(tidyverse)
            library(forcats)  # not in the default tidyverse loadout
        })
```

ggplot has a peculiar method to set default scales: You just have to define certain variables.

```
In [3]: scale_colour_continuous <- scale_color_viridis_c
```

When working mainly with dimension reductions, I suggest to hide the (useless) ticks:

```
In [4]: theme_set(theme_gray() + theme(
            axis.ticks = element_blank(),
            axis.text  = element_blank()))
```

Let's load our dataset

```
In [5]: data(guo_norm)
```

Of course you could use tidyr::gather() to tidy or transform the data now, but the data is already in the right form for destiny, and R for Data Science is a better resource for it than this vignette. The long form of a single cell `ExpressionSet` would look like:

```
In [6]: guo_norm %>%
            as('data.frame') %>%
            gather(Gene, Expression, one_of(featureNames(guo_norm)))
```

Cell	num_cells	Gene	Expression
2C 1.1	2	Actb	-0.575
2C 1.2	2	Actb	-0.435
2C 2.1	2	Actb	0.460
2C 2.2	2	Actb	0.610
2C 3.1	2	Actb	1.970
⋮	⋮	⋮	⋮
64C 7.10	64	Tspan8	3.220
64C 7.11	64	Tspan8	3.415
64C 7.12	64	Tspan8	4.540
64C 7.13	64	Tspan8	5.315
64C 7.14	64	Tspan8	2.865

But destiny doesn't use long form data as input, since all single cell data has always a more compact structure of genes×cells, with a certain number of per-sample covariates (The structure of `ExpressionSet`).

```
In [7]: dm <- DiffusionMap(guo_norm)
```

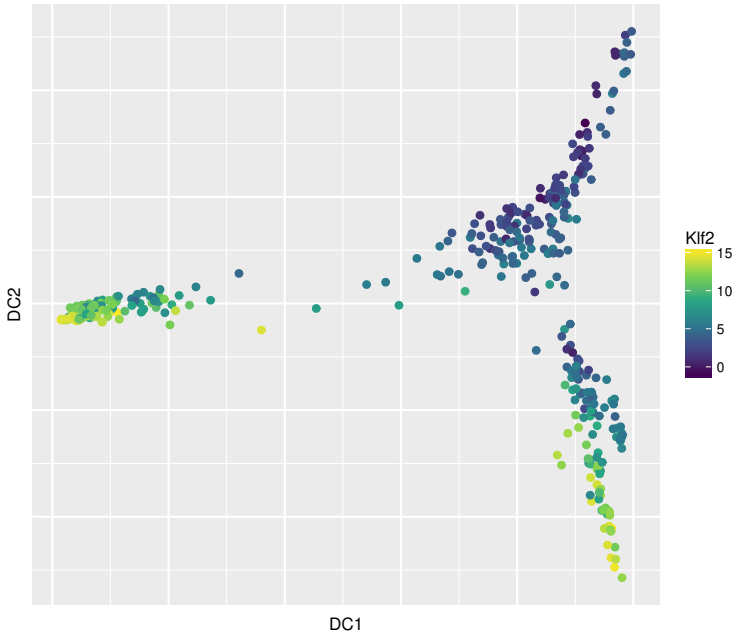
`names(dm)` shows what names can be used in `dm$<name>`, `as.data.frame(dm)$<name>`, or `ggplot(dm, aes(<name>))`:

```
In [8]: names(dm)  # namely: Diffusion Components, Genes, and Covariates
```

① 'DC1' ② 'DC2' ③ 'DC3' ④ 'DC4' ⑤ 'DC5' ⑥ 'DC6' ⑦ 'DC7' ⑧ 'DC8' ⑨ 'DC9' ⑩ 'DC10' ⑪ 'DC11' ⑫ 'DC12' ⑬ 'DC13' ⑭ 'DC14' ⑮ 'DC15' ⑯ 'DC16' ⑰ 'DC17' ⑱ 'DC18' ⑲ 'DC19' ⑳ 'DC20' ㉑ 'Actb' ㉒ 'Ahcy' ㉓ 'Aqp3' ㉔ 'Atp12a' ㉕ 'Bmp4' ㉖ 'Cdx2' ㉗ 'Creb312' ㉘ 'Cebpa' ㉙ 'Dab2' ㉚ 'Dppa1' ㉛ 'Eomes' ㉜ 'Esrrb' ㉝ 'Fgf4' ㉞ 'Fgfr2' ㉟ 'Fn1' ㊱ 'Gapdh' ㊲ 'Gata3' ㊳ 'Gata4' ㊴ 'Gata6' ㊵ 'Grhl1' ㊶ 'Grhl2' ㊷ 'Hand1' ㊸ 'Hnf4a' ㊹ 'Id2' ㊺ 'Klf2' ㊻ 'Klf4' ㊼ 'Klf5' ㊽ 'Krt8' ㊾ 'Lcp1' ㊿ 'Mbnl3' (51) 'Msc' (52) 'Msx2' (53) 'Nanog' (54) 'Pdgfa' (55) 'Pdgfra' (56) 'Pecam1' (57) 'Pou5f1' (58) 'Runx1' (59) 'Sox2' (60) 'Sall4' (61) 'Sox17' (62) 'Snail' (63) 'Sox13' (64) 'Tcfap2a' (65) 'Tcfap2c' (66) 'Tcf23' (67) 'Utf1' (68) 'Tspan8' (69) 'Cell' (70) 'num_cells'

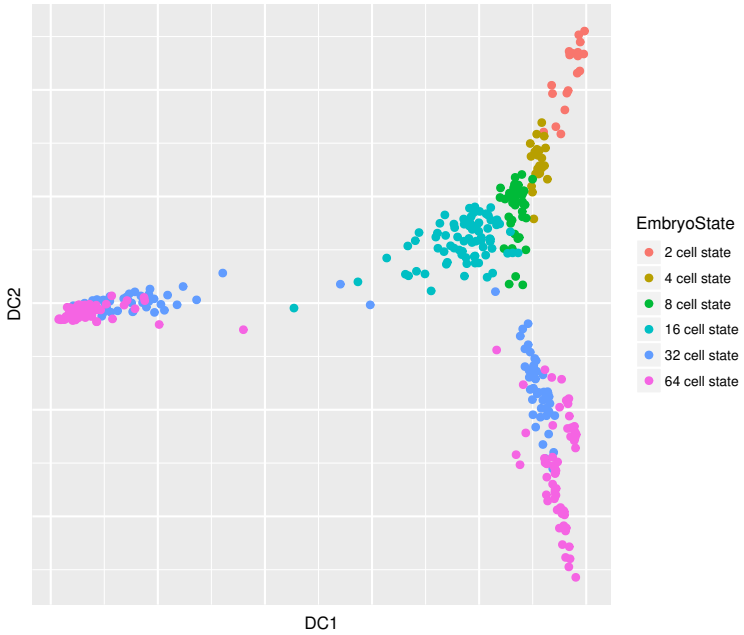
Due to the `fortify` method (which here just means `as.data.frame`) being defined on `DiffusionMap` objects, `ggplot` directly accepts `DiffusionMap` objects:

```
In [9]: ggplot(dm, aes(DC1, DC2, colour = Klf2)) +
            geom_point()
```



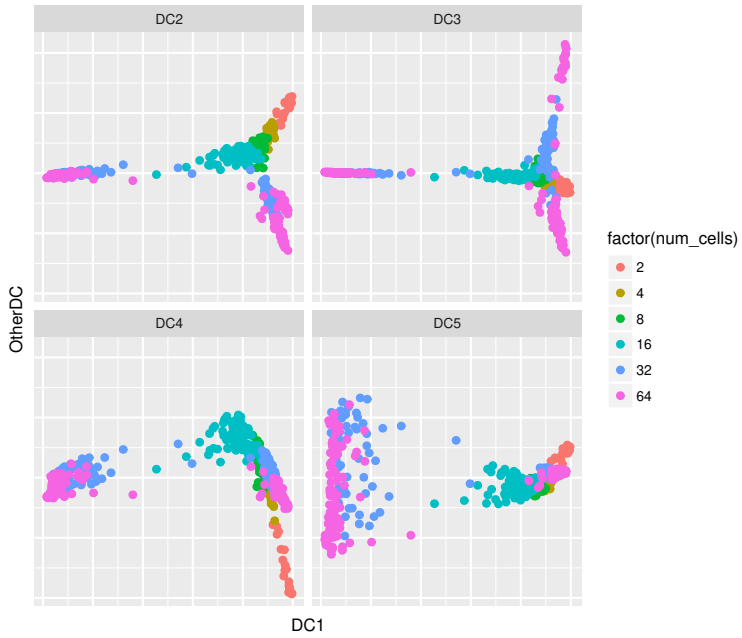
When you want to use a Diffusion Map in a dplyr pipeline, you need to call `fortify`/`as.data.frame` directly:

```
In [10]: fortify(dm) %>%
             mutate(
                 EmbryoState = factor(num_cells) %>%
                     lvls_revalue(paste(levels(.), 'cell state'))
             ) %>% ggplot(aes(DC1, DC2, colour = EmbryoState)) +
                 geom_point()
```



The Diffusion Components of a converted Diffusion Map, similar to the genes in the input `ExpressionSet`, are individual variables instead of two columns in a long-form data frame, but sometimes it can be useful to "tidy" them:

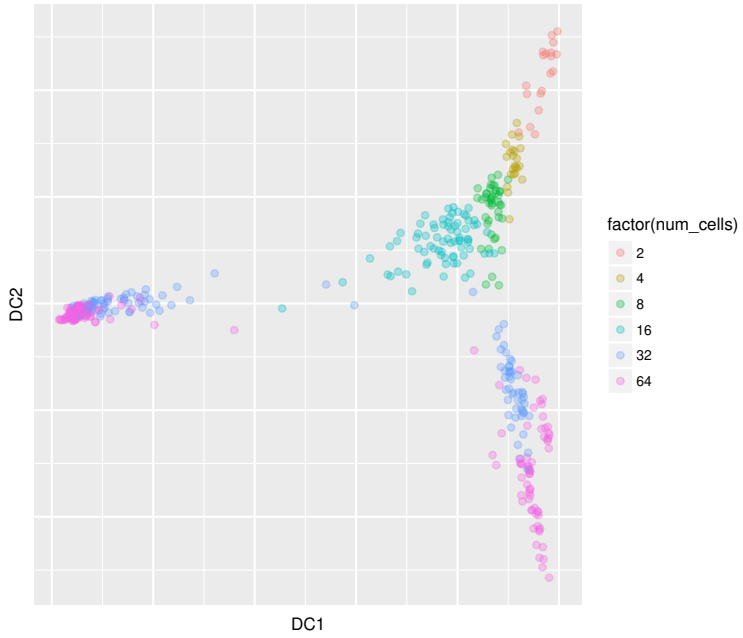
```
In [11]: fortify(dm) %>%
             gather(DC, OtherDC, num_range('DC', 2:5)) %>%
             ggplot(aes(DC1, OtherDC, colour = factor(num_cells))) +
                 geom_point() +
                 facet_wrap(~ DC)
```



Another tip: To reduce overplotting, use `sample_frac(., 1.0, replace = FALSE)` (the default) in a pipeline.

Adding a constant `alpha` improves this even more, and also helps you see density:

```
In [12]: fortify(dm) %>%
             sample_frac() %>%
             ggplot(aes(DC1, DC2, colour = factor(num_cells))) +
                 geom_point(alpha = .3)
```


DC2
DC1
factor(num_cells)
2
4
8
16
32
64