svdvis: Visualizing SVD, PCA, and related methods

Neo Christopher Chung
nchchung@gmail.com

2015-12-02

This package provides several visualization functions for singular value decomposition (SVD), principal component analysis (PCA), factor analysis (FA), logistic factor analysis (LFA), and other related methods.

Simulated data

To use in this vignette, we create a simulated dataset, with `m=500` variables (rows) and `n=20` samples (columns). Particularly, it contains a latent variable that resembles a case-control study. After applying SVD to the datasets, we also name the rows and the columns of the right singular vectors `svd.obj$v` for labels in visualization.

```r
set.seed(1234)
library(svdvis)
B = c(runif(100, min=0, max=1), rep(0,400))
L = c(rep(1, 10), rep(-1, 10))
L = L / sd(L)
E = matrix(rnorm(500*20), nrow=500)
Y = B %*% t(L) + E

svd.obj = svd(Y)
colnames(svd.obj$v) = paste0("V",1:20)
rownames(svd.obj$v) = paste0("Sample",1:20)
```

In this setup, a few right singular vectors contained in `svd.obj$v` may capture systematic variation in the observed data `Y`. Since the right singular vectors are ordered according to the singular values in a descending order, the top (or first) `r` right singular vectors refers to `svd.obj$v[,1:r]`. Note that principal components (PCs) can be obtained by multiplying singular values `svd.obj$d` and right singular vectors `svd.obj$v`. All examples in this vignette and all functions in `svdvis` can utilize `weights="sv"` to quickly visualize PCs.

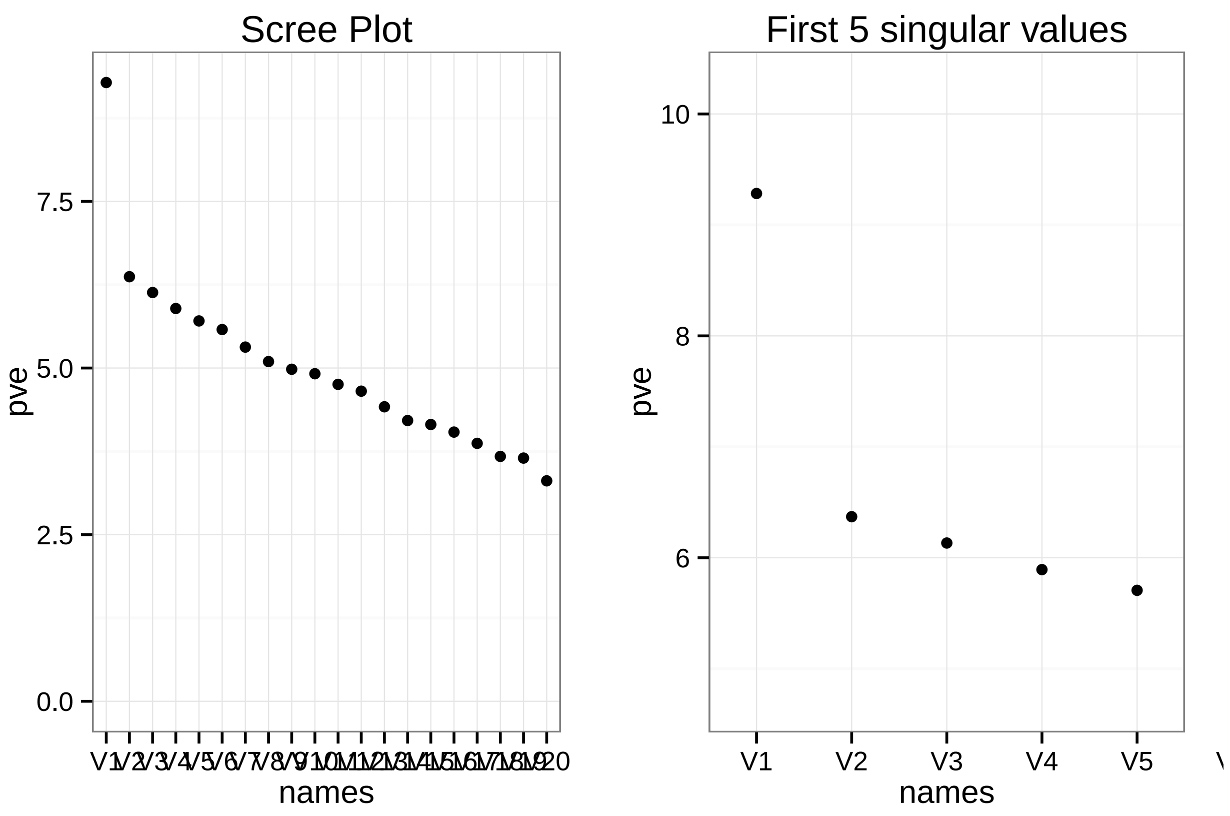
Scree plot

A scree plot visualizes percentages of variance explained by singular vectors in a descending order. `svd.scree` is simply a wrapper function using `ggplot2`. In high-dimensional datasets, the number of points in a scree plot may be too large. It may be good to look at a subset of singular values. You can specify `subr` in `svd.scree` function, which "zooms in" to the top `subr` singular values.

```r
svd.scree(svd.obj, subr=5,
          axis.title.x="Full scree plot", axis.title.y="% Var Explained")
```

```
## [1] "Your input data is treated as a SVD output, with u, d, v corresponding to left singular vector,
## [1] "Scree Plot"
```

```
## Warning: Removed 9 rows containing missing values (geom_point).
```



```
## TableGrob (1 x 2) "arrange": 2 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
```

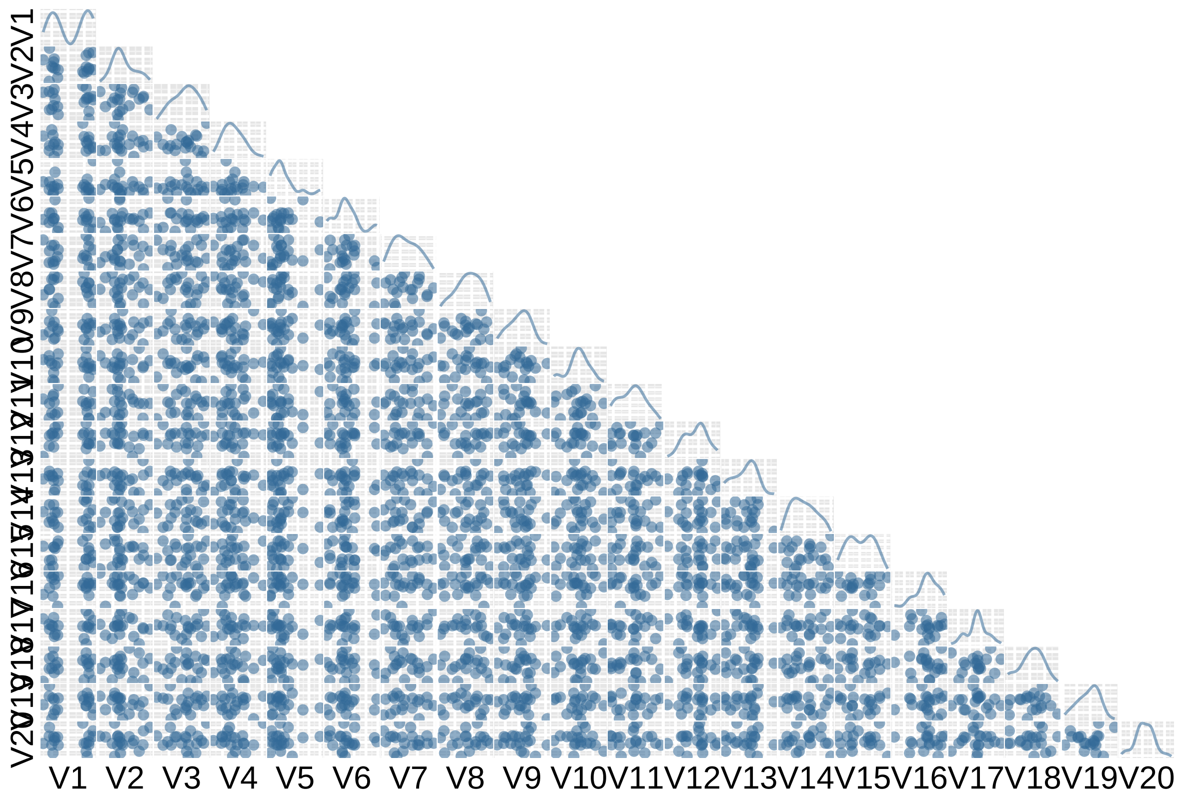
Note that if `subr` is not specified, one full-sized scree plot is returned.

Paired scatterplots

Scatter plots are often utilized to look at the top 2 right singular vectors. `svd.scatter` produces a matrix of scatterplots of all pairs among `r` right singular vectors.

```
svd.scatter(svd.obj)
```

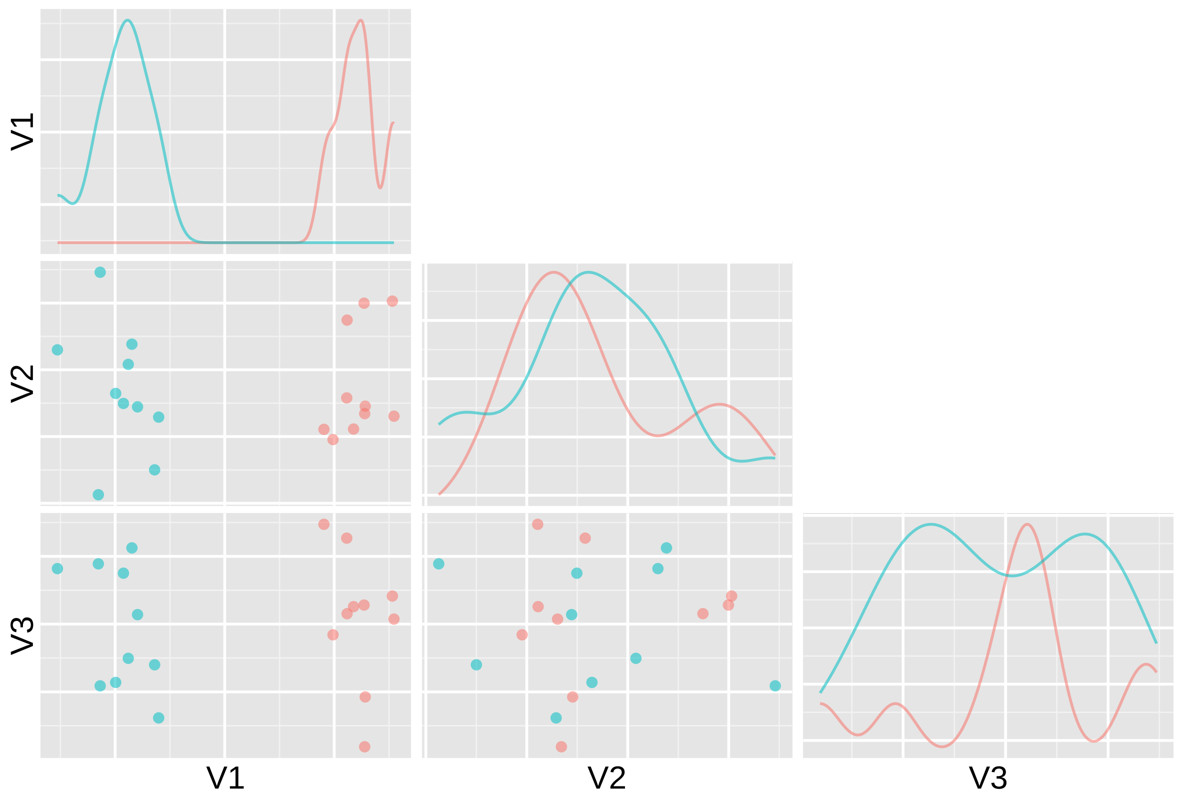
```
## [1] "Your input data is treated as a SVD output, with u, d, v corresponding to left singular vector,
## [1] "Multiple Scatter Plots"
## [1] "It may not be good to visualize too many singular vectors or principal components at one."
```



The above plot crams in too many pairs. We can specify `r` to visualize only the top `r` right singular vectors. In this example, additional arguments such as `group` and `alpha` are included:

```
svd.scatter(svd.obj, r=3, alpha=.5,
            group=c(rep("Group 1", 10), rep("Group 2", 10)))
```

```
## [1] "Your input data is treated as a SVD output, with u, d, v corresponding to left singular vector,
## [1] "Multiple Scatter Plots"
```

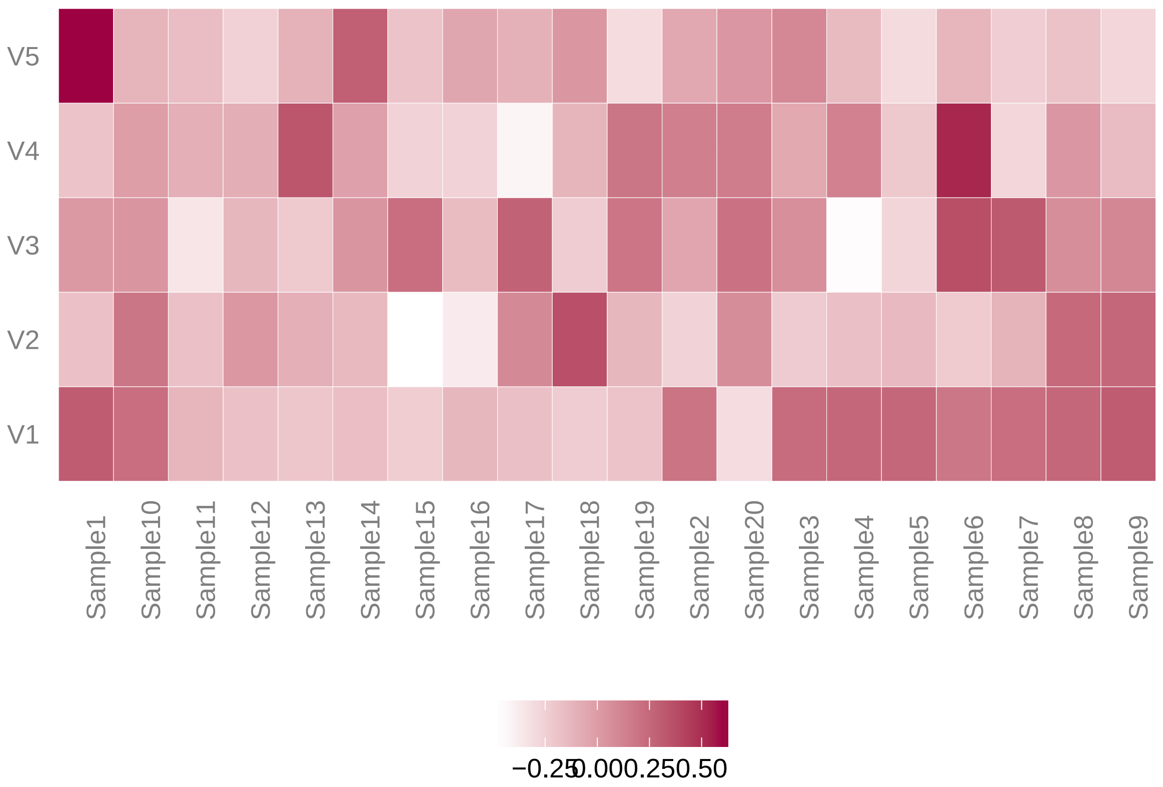


Heat map

Let's create a heat map of the top `r=5` right singular vectors:

```
svd.heatmap(svd.obj, r=5)
```

```
## [1] "Your input data is treated as a SVD output, with u, d, v corresponding to left singular vector,
## [1] "SVD Heatmap"
```

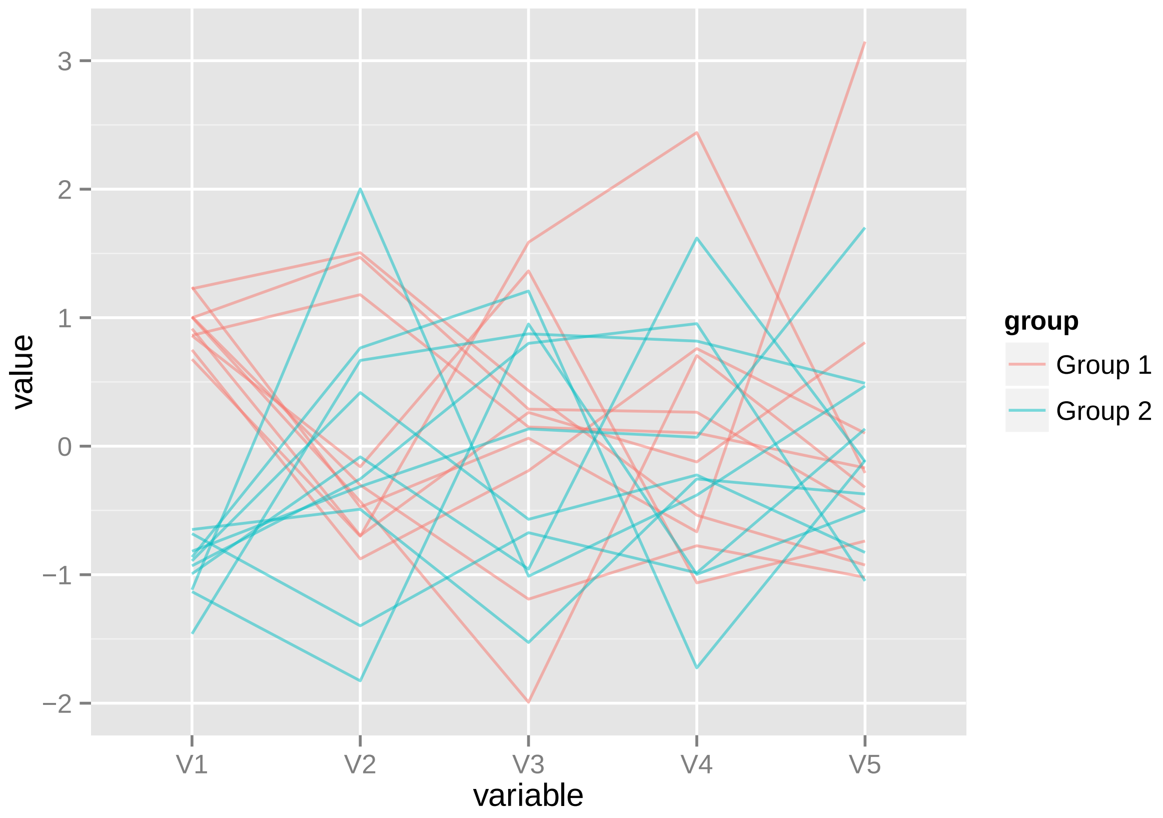


Parallel coordinates plot

A parallel coordinates plot shows `r` dimensions in `r` parallel lines, which are equally spaced. All data points are rescaled to (0,1) and the top `r` singular vectors are visualized from left to right. Different groups are colored accordingly:

```
svd.parallel(svd.obj, r=5, alpha=.5,
             group=c(rep("Group 1", 10), rep("Group 2", 10)))
```

```
## [1] "Your input data is treated as a SVD output, with u, d, v corresponding to left singular vector,
## [1] "Parallel Coordinates Plot"
```

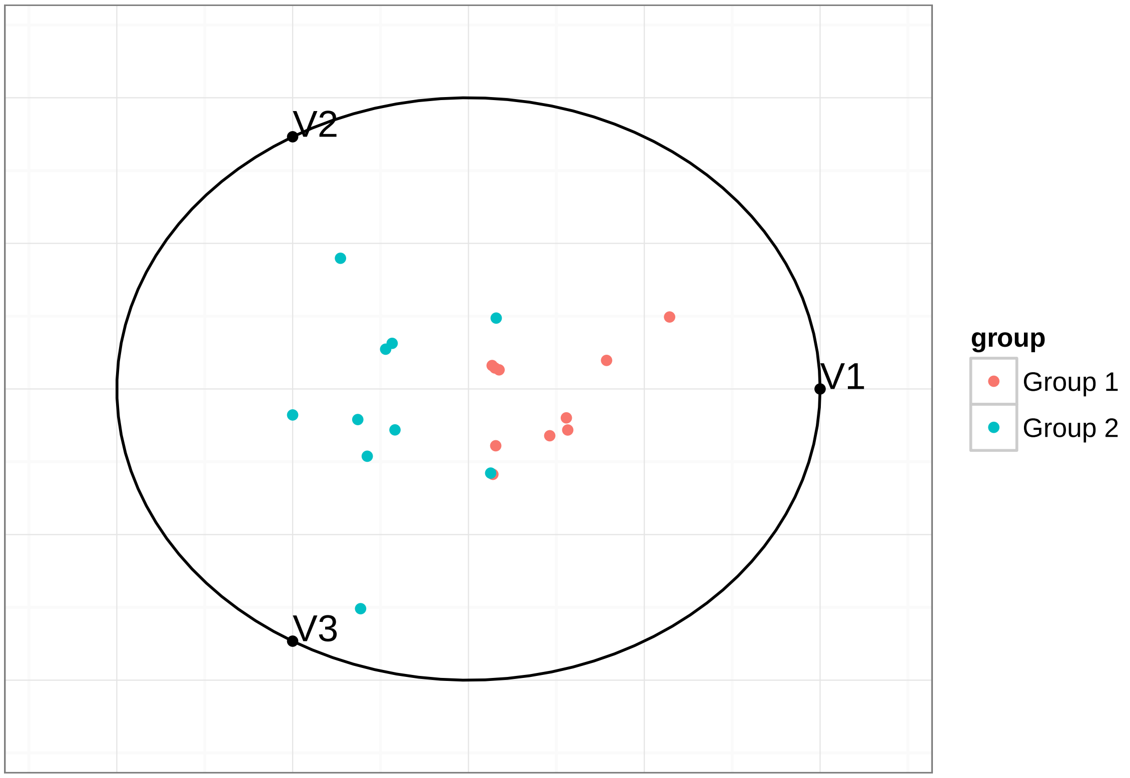


Radial coordinates plot

A radial coordinates plot visualize `r` dimensions in a circle, around where `r` anchors are placed. Each of `n` vectors is mapped onto a circle, using its data as spring constants. Prior to mapping, each column is rescaled to have numeric values between 0 and 1.

```
svd.radial(svd.obj, r=3,
           group=c(rep("Group 1", 10), rep("Group 2", 10)))
```

```
## [1] "Radial Visualization Plots"
## [1] "Your input data is treated as a SVD output, with u, d, v corresponding to left singular vector,
```



Tips and remarks

All functions in `svdvis` use `ggplot2`. Therefore, the visual output can be saved and modified in a conventional manner. Feel free to experiment the source codes for more complex or interesting cases.

While this vignette focused on using the results of SVD, an optional argument `weights="sv"` can be used for visualizing PCs. Note that `weights="sv"` is simply calling `weights = svd.obj$d[1:r]`.

Outputs from other dimension reduction methods can be used. Provide the `r` vectors to `svd.obj` in any function. Note that the input must be a `n * r` matrix that contains `r` vectors as columns. An optional argument `group` can be used to differentially indicate `n` samples (points, lines, etc).

For example, logistic factor analysis captures population structure from a large and diverse set of genome sequences and is related to SVD and PCA. A R package `lfa` computes `r` logistic factors, as columns. You can easily make a parallel coordinates plot (and others) by `svd.parallel(svd.obj=lfa(genotypes, 10))`.