Scanpath – An R Package for Analyzing Scanpaths

Titus von der Malsburg

January 30, 2018

Contents

An R package for analyzing scanpaths in eye movement data. The package includes a simple toy dataset and example code. Consult von der Malsburg & Vasishth (2011) for the details of this analysis method. The manual of the package can be found here and a PDF-version of this page here.

1 News

[2018-01-30 Tue] Version 1.06 doesn't logarithmize fixation durations when calculating scanpath similarities. (In previous versions, when `normalize="durations"` was used, the normalization was done using non-log-transformed durations, which could in some rare cases break the triangle inequality.)

2 Install

On Linux and macOS, this should work out of the box. I have no idea what needs to be done to make this work on Windows. It may work, but I haven't tried it. Feedback welcome.

3 Usage example

You can find all code shown below in the file `README.R`. You can open that file in RStudio and play with the code as you read through this mini-tutorial.

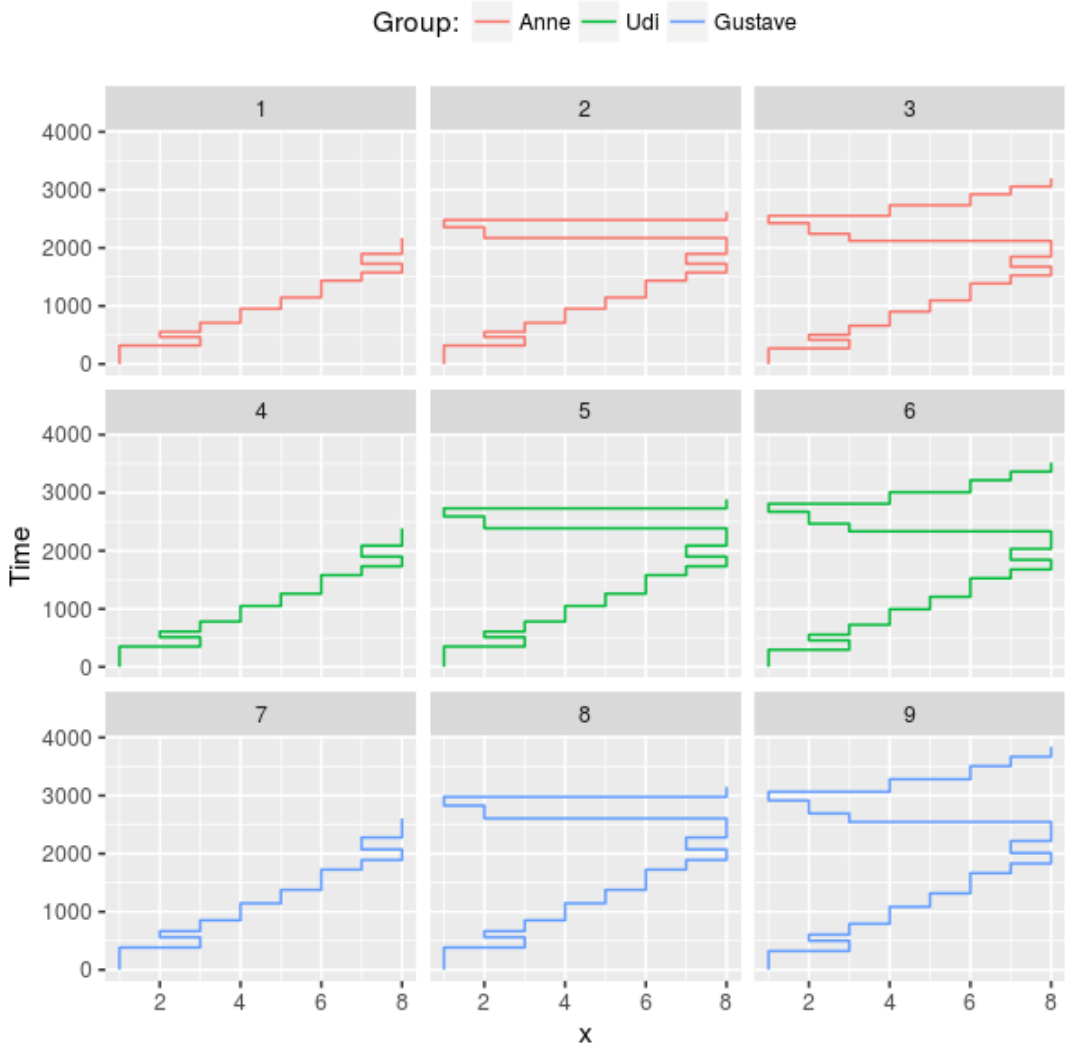
Let's have a look at the toy data that is included in the package:

```
library(scanpath)
data(eyemovements)
head(eyemovements)
```

subject	trial	word	x	y	duration
Anne	1	1	46.145	384.075	319
Anne	1	3	131.4	387.77	147
Anne	1	2	106.22	385.94	88
Anne	1	3	165.26	386.75	156
Anne	1	4	185.615	385.76	244
Anne	1	5	264.11	387.78	193

To get a sense of what is going on in this data set, we create a series of plots. For this purpose, we use the function `plot_scanpaths` from the *scanpath* package. In the first plot below, each panel shows the data from one trial. There are three participants which are coded by color. The data is from a sentence reading task. The x-axis shows words and the y-axis time within trial in milliseconds.

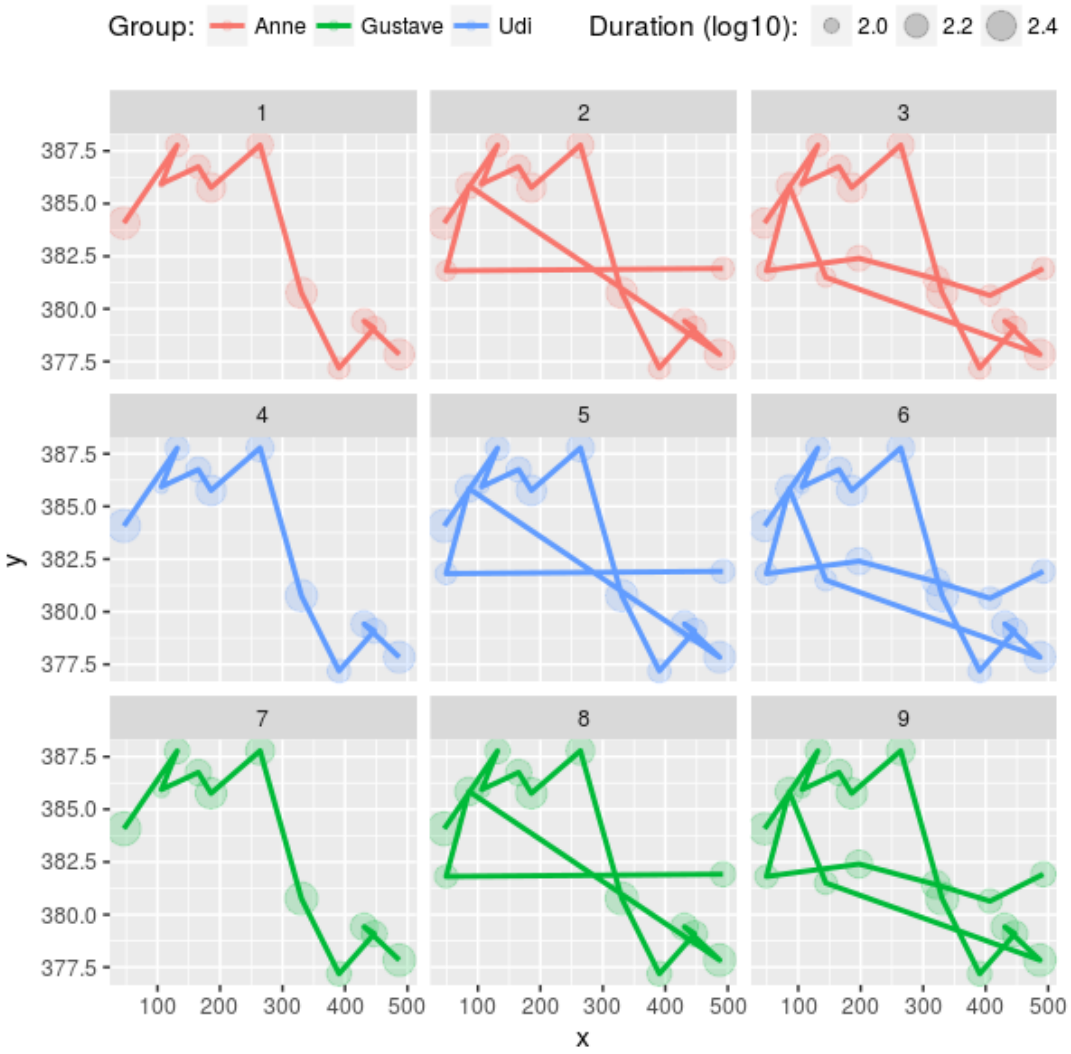
```
plot_scanpaths(duration ~ word | trial, eyemovements, subject)
```



We can see that the participants differ in their reading speed. Also we see that each participant read the sentence more or less straight from left to right (trials: 1, 4, 7), or with a short regressions from the end of the sentence to its beginning (trials: 2, 5, 8), or with a long regression from the end of the sentence (trials: 3, 6, 9).

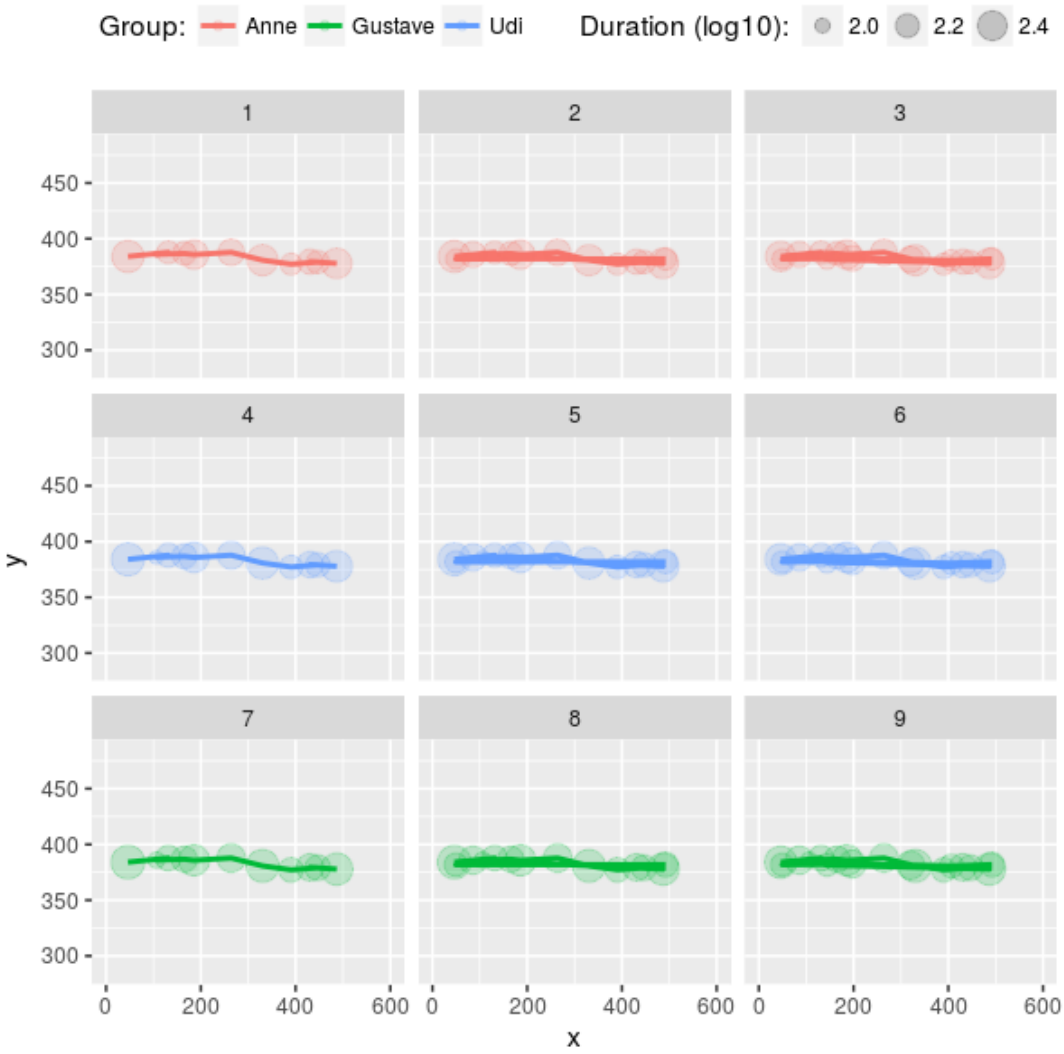
In the next plot, we use the fixations' x- and y-coordinates. Each circle is a fixation and the size of the circle represents the duration of the corresponding fixation.

```
plot_scanpaths(duration ~ x + y | trial, eyemovements, subject)
```



The function `plot_scanpaths` returns a *ggplot2* object. This means that we can style the plot before rendering it. For example, we can change the limits of the axes:

```
plot_scanpaths(duration ~ x + y | trial, eyemovements, subject) +
  xlim(0, 600) + ylim(284, 484)
```



Now, we calculate the pair-wise similarities of the nine scanpaths in the dataset using the *scasim* measure. A simplifying intuition is that the measure quantifies the time that was spent looking at different things or at the same things in different order. For a precise definition see von der Malsburg & Vasishth (2011).

```
d1 <- scasim(eyemovements, duration ~ x + y | trial, 512, 384, 60, 1/30,
             normalize=FALSE)
round(d1)
```

	1	2	3	4	5	6	7	8	9
1	0	454	1129	217	717	1395	434	979	1669
2	454	0	675	671	263	941	888	525	1215
3	1129	675	0	1346	938	320	1563	1200	640
4	217	671	1346	0	499	1242	217	762	1508
5	717	263	938	499	0	743	717	263	1009
6	1395	941	320	1242	743	0	1459	1005	320
7	434	888	1563	217	717	1459	0	545	1355
8	979	525	1200	762	263	1005	545	0	810
9	1669	1215	640	1508	1009	320	1355	810	0

Like the function `plot_scanpaths`, the function `scasim` takes a formula and a data frame as parameters. The formula specifies which columns in the data frame should be used for the calculations. To account for distortion due to visual perspective, the comparison of the scanpaths is carried out in visual field coordinates (latitude and longitude). In order to transform the pixel coordinates provided by the eye-tracker to visual field coordinates, the `scasim` function needs some extra information. The first is the position of the gaze when the participant looked straight ahead (512, 384, in the present case), the distance of the eyes from the screen (60 cm), and the size of one pixel in the unit that was used to specify the distance from the screen (1/30). Finally, we have to specify a normalization procedure. `normalize=FALSE` means that we don't want to normalize. See the documentation of the `scasim` function for details.

The time that was spent looking at different things of course depends on the duration of the two compared trials. (total duration of the two compared scanpaths constitutes an upper bound). This means that two long scanpaths may have a larger dissimilarity than two shorter scanpaths even if they look more similar. Depending on the research question, this may be undesirable. One way to get rid of the trivial influence of total duration is to normalize the dissimilarity scores. For example, we can divide them by the total duration of the two compared scanpaths:

```
d2 <- scasim(eyemovements, duration ~ x + y | trial, 512, 384, 60, 1/30,
             normalize="durations")
round(d2, 2)
```

	1	2	3	4	5	6	7	8	9
1	0	0.09	0.21	0.05	0.14	0.25	0.09	0.18	0.28
2	0.09	0	0.12	0.13	0.05	0.15	0.17	0.09	0.19
3	0.21	0.12	0	0.24	0.15	0.05	0.27	0.19	0.09
4	0.05	0.13	0.24	0	0.09	0.21	0.04	0.14	0.24
5	0.14	0.05	0.15	0.09	0	0.12	0.13	0.04	0.15
6	0.25	0.15	0.05	0.21	0.12	0	0.24	0.15	0.04
7	0.09	0.17	0.27	0.04	0.13	0.24	0	0.09	0.21
8	0.18	0.09	0.19	0.14	0.04	0.15	0.09	0	0.12
9	0.28	0.19	0.09	0.24	0.15	0.04	0.21	0.12	0

The number are smaller now and can be interpreted as the proportion of time that was spent looking at different things.

The numbers in the matrix above capture a lot of information about the scanpath variance in the data set. However, dissimilarity scores are somewhat tricky to analyze. One problem is that these values have strong statistical dependencies. When we change one scanpath, this affects *n* dissimilarity scores. This has to be kept in mind when doing inferential stats directly on the dissimilarity scores. While there are solutions for this, it is typically more convenient to produce a representation of scanpath variance that is free from this problem. One such representation is what we call the "map of scanpath space." On such a map, every point represents a scanpath and the distances on the map reflect the dissimilarities according to our scanpath measure, i.e. the dissimilarity scores in the matrix above.

The method for calculating these maps is called multi-dimensional scaling and one simple version of the general idea is implemented in the function `cmdscale`.

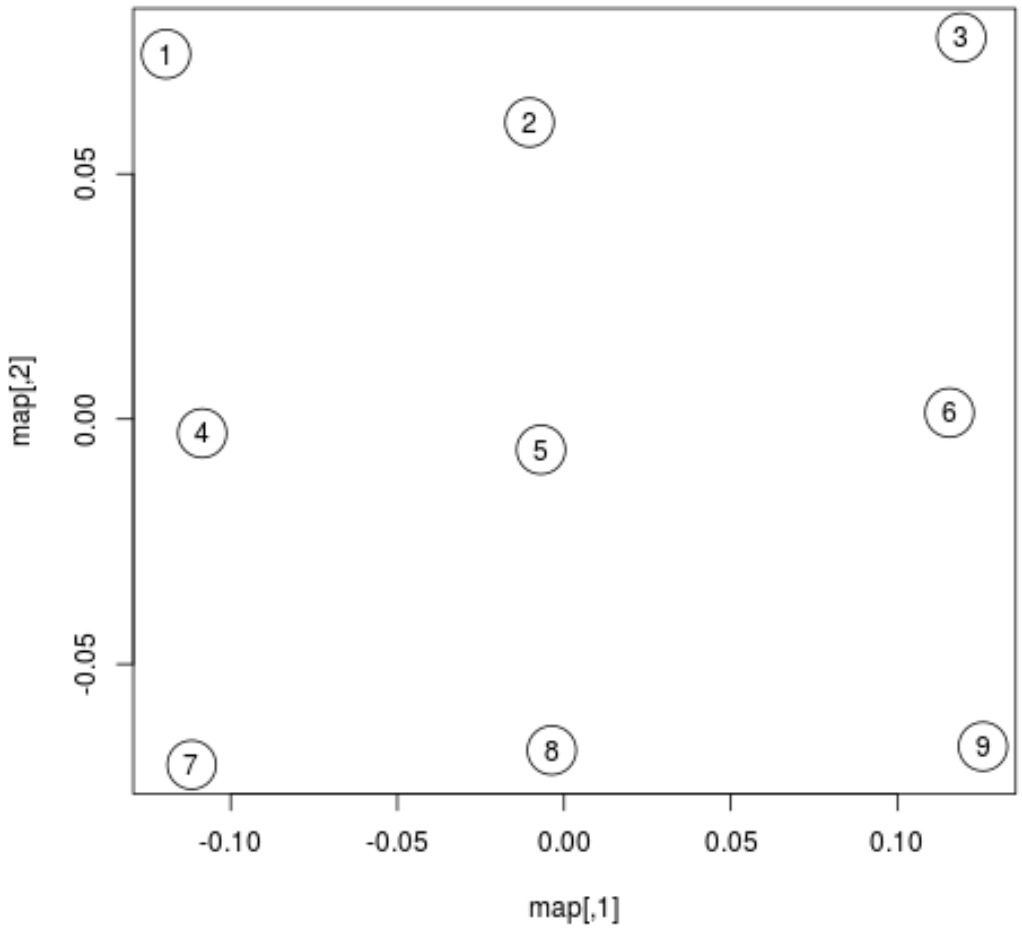
```
map <- cmdscale(d2)
round(map, 2)
```

	V1	V2
1	-0.12	-0.07
2	-0.01	-0.06
3	0.12	-0.08
4	-0.11	0
5	-0.01	0.01
6	0.12	0
7	-0.11	0.07
8	0	0.07
9	0.13	0.07

The table above contains two numbers for each scanpath in the data set. These numbers (`V1` and `V2`) determine a scanpath's location in the two-dimensional scanpath space created by `cmdscale`. How many dimensions we need is an empirical question.

Below is a plot showing the map of scanpaths:

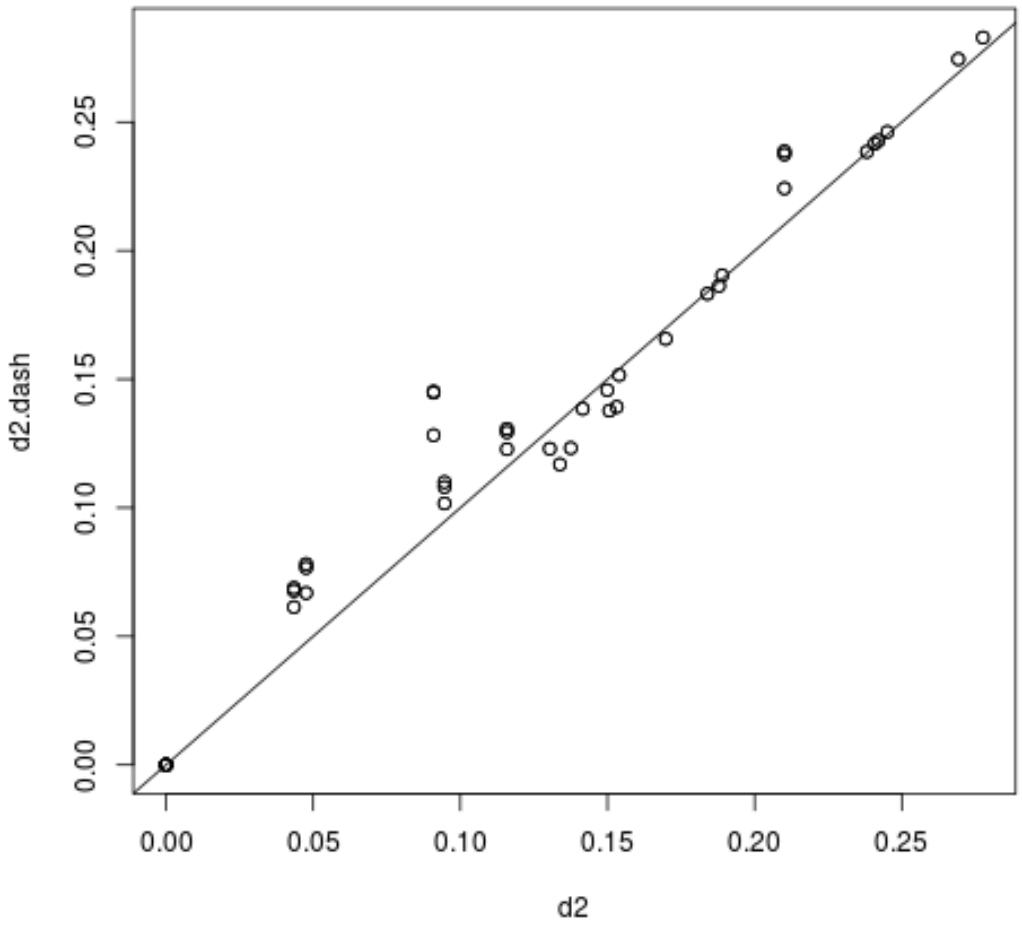
```
map <- map %*% matrix(c(1, 0, 0, -1), 2)  # flip y-axis
plot(map, cex=4)
text(map, labels=rownames(map))
```



Interestingly, the scanpaths are arranged in the same way as in the plot of the data at the top. Participants are arranged vertically and reading patterns are horizontally. This suggests that *scasim* not just recovered these two different kinds of information (reading speed and reading strategy) but also that it can distinguish between them.

To test how well this map represents the original dissimilarity scores, we can calculate the pair-wise differences on the map and compare them to the pair-wise *scasim* scores:

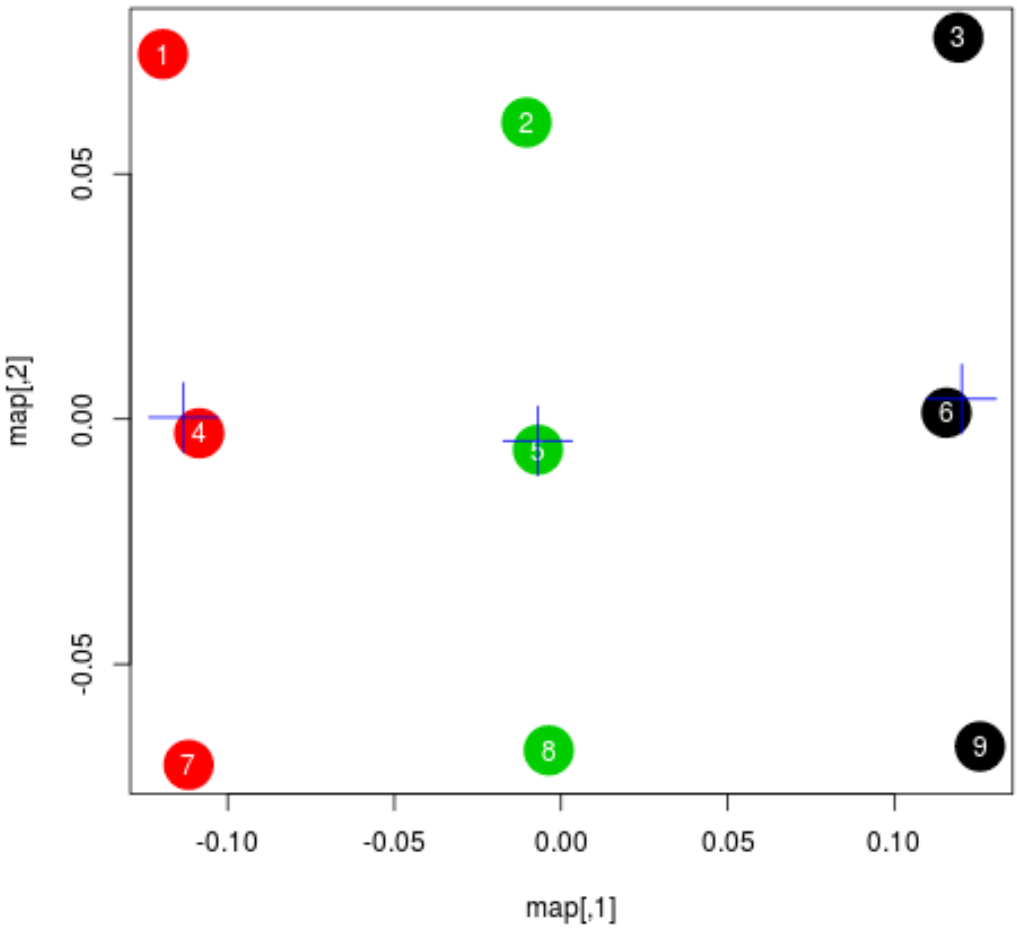
```
d2.dash <- as.matrix(dist(map))
plot(d2, d2.dash)
abline(0, 1)
```



This plot suggests that the map preserves the variance in dissimilarity scores really well. Given this very good fit of the map, it appears that two dimensions were sufficient to describe the scanpath variance that is captured by *scasim*. This is not surprising because the scanpaths in the toy data set were designed to vary with respect to two properties: 1.) The speed of the reader, and 2.) whether there was a regression back to the beginning of the sentence and how long it was.

The benefit of the map representation is that it has much weaker statistical dependencies and that it is much more suitable for all kinds of analyses. For example, we can choose among a large number of clustering algorithms to test whether there are groups of similar scanpaths in a data set. Below, we use the simple k-means algorithm to illustrate this:

```
set.seed(4)
clusters <- kmeans(map, 3, iter.max=100)
plot(map, cex=4, col=clusters$cluster, pch=19)
text(map, labels=rownames(map), col="white")
points(clusters$centers, col="blue", pch=3, cex=4)
```



In this plot, color indicates to which cluster a scanpath belongs and the crosses show the center of each cluster. We see that the clusters correspond to the different reading patterns and that participants are ordered according to their reading speed within the clusters.

Apart from cluster analyses there are many other ways to analyze scanpath variance. See the articles listed below for more details.

4 References

- von der Malsburg, T., & Vasishth, S. (2011). What is the scanpath signature of syntactic reanalysis? Journal of Memory and Language, 65(2), 109–127. `http://dx.doi.org/10.1016/j.jml.2011.02.004`
- von der Malsburg, T., Kliegl, R., & Vasishth, S. (2015). Determinants of scanpath regularity in reading. Cognitive Science, 39(7), 1675–1703. `http://dx.doi.org/10.1111/cogs.12208`

- von der Malsburg, T., & Vasishth, S. (2013). Scanpaths reveal syntactic underspecification and reanalysis strategies. Language and Cognitive Processes, 28(10), 1545–1578. http://dx.doi.org/10.1080/01690965.2012.728232
- von der Malsburg, T., Vasishth, S., & Kliegl, R. (2012). Scanpaths in reading are informative about sentence processing. In P. B. Michael Carl, & K. K. Choudhary, Proceedings of the First Workshop on Eye-tracking and Natural Language Processing (pp. 37–53). Mumbai, India: The COLING 2012 organizing committee.