Blocking: An R Package for Blocking of Records for Record Linkage and Deduplication

by Maciej Beręsewicz and Adam Struzik

Abstract An abstract of less than 250 words.

1 Introduction

Interactive data graphics provides plots that allow users to interact them. One of the most basic types of interaction is through tooltips, where users are provided additional information about elements in the plot by moving the cursor over the plot.

This paper will first review some R packages on interactive graphics and their tooltip implementations. A new package **ToOoOlTiPs** that provides customized tooltips for plot, is introduced. Some example plots will then be given to showcase how these tooltips help users to better read the graphics.

2 Background

Some packages on interactive graphics include **plotly** (Sievert, 2020) that interfaces with Javascript for web-based interactive graphics, **crosstalk** (Cheng and Sievert, 2021) that specializes cross-linking elements across individual graphics. The recent R Journal paper **tsibbletalk** (Wang and Cook, 2021) provides a good example of including interactive graphics into an article for the journal. It has both a set of linked plots, and also an animated gif example, illustrating linking between time series plots and feature summaries.

3 Blocking of records using `blocking` function

4 Integration with existing packages

5 Case study

5.1 Record linkage example

Let us first load the required packages.

```
library(blocking)
library(data.table)
```

We will demonstrate the use of `blocking` function for record linkage with the `foreigners` dataset included in the package. This fictional representation of the foreign population in Poland was generated based on publicly available information, preserving the distributions from administrative registers. It contains 110,000 rows with 100,000 entities. Each row represents one record, with the following columns:

- `fname` – first name,
- `sname` – second name,
- `surname` – surname,
- `date` – date of birth,
- `region` – region (county),

- `country` – country,
- `true_id` – person ID.

```
data(foreigners)
head(foreigners)
```

```
#>     fname  sname    surname       date region country true_id
#>    <char> <char>     <char>     <char> <char>  <char>   <num>
#> 1:   emin            imanov 1998/02/05            031       0
#> 2: nurlan        suleymanli 2000/08/01            031       1
#> 3:   amio        maharrsmov 1939/03/08            031       2
#> 4:   amik        maharramof 1939/03/08            031       2
#> 5:   amil        maharramov 1993/03/08            031       2
#> 6:  gadir        jahangirov 1991/08/29            031       3
```

We split the dataset into two separate files: one containing the first appearance of each entity in the `foreigners` dataset, and the other containing its subsequent appearances.

```
foreigners_1 <- foreigners[!duplicated(foreigners$true_id), ]
foreigners_2 <- foreigners[duplicated(foreigners$true_id), ]
```

Now in both datasets we remove slashes from the `date` column and create a new string column that concatenates the information from all columns (excluding `true_id`) in each row.

```
foreigners_1[, date := gsub("/", "", date)]
foreigners_1[, txt := paste0(fname, sname, surname, date, region, country)]
foreigners_2[, date := gsub("/", "", date)]
foreigners_2[, txt := paste0(fname, sname, surname, date, region, country)]
head(foreigners_1)
```

```
#>     fname  sname    surname     date region country true_id
#>    <char> <char>     <char>   <char> <char>  <char>   <num>
#> 1:   emin            imanov 19980205            031       0
#> 2: nurlan        suleymanli 20000801            031       1
#> 3:   amio        maharrsmov 19390308            031       2
#> 4:  gadir        jahangirov 19910829            031       3
#> 5:   zaur         bayramova 19961006  01261     031       4
#> 6:   asif          mammadov 19970726            031       5
#>                            txt
#>                         <char>
#> 1:         eminimanov19980205031
#> 2:   nurlansuleymanli20000801031
#> 3:     amiomaharrsmov19390308031
#> 4:    gadirjahangirov19910829031
#> 5: zaurbayramova1996100601261031
#> 6:       asifmammadov19970726031
```

We use the newly created columns in the `blocking` function, which relies on the default **rnndescent** (Nearest Neighbor Descent) algorithm based on cosine distance. Additionally, we set `verbose = 1` to monitor progress.

```
result_reclin <- blocking(x = foreigners_1$txt, y = foreigners_2$txt, verbose = 1)
```

```
#> ===== creating tokens =====
#> ===== starting search (nnd, x, y: 100000, 10000, t: 1232) =====
#> ===== creating graph =====
```

Now we examine the results of record linkage.

- We have created 6,469 blocks.
- The blocking process utilized 1,232 columns (2 character shingles).
- We have 3,916 blocks of 2 elements, 1,604 blocks of 3 elements,..., 2 blocks of 7 elements.

```
result_reclin
```

```
#> ========================================================
#> Blocking based on the nnd method.
#> Number of blocks: 6469.
#> Number of columns used for blocking: 1232.
#> Reduction ratio: 0.9999.
#> ========================================================
#> Distribution of the size of the blocks:
#>    2    3    4    5    6    7
#> 3916 1604  926   19    2    2
```

Structure of the object is as follows:

- `result` – a `data.table` with identifiers and block IDs,
- `method` – name of the ANN algorithm used,
- `deduplication` – whether deduplication was applied,
- `representation` – whether shingles or vectors were used,
- `metrics` – metrics for quality assessment (here `NULL`),
- `confusion` – confusion matrix (here `NULL`),
- `colnames` – column names used for the comparison,
- `graph` – an **igraph** object, mainly for visualization (here `NULL`).

```
str(result_reclin, 1)
```

```
#> List of 8
#>  $ result     :Classes 'data.table' and 'data.frame': 10000 obs. of  4 variables:
#>   ..- attr(*, ".internal.selfref")=<externalptr>
#>  $ method        : chr "nnd"
#>  $ deduplication : logi FALSE
#>  $ representation: chr "shingles"
#>  $ metrics       : NULL
#>  $ confusion     : NULL
#>  $ colnames      : chr [1:1232] "0a" "0b" "0c" "0m" ...
#>  $ graph         : NULL
#>  - attr(*, "class")= chr "blocking"
```

The resulting `data.table` has four columns:

- `x` – reference dataset (i.e. `foreigners_1`) – this may not contain all units of `foreigners_1`,
- `y` – query (each row of `foreigners_2`) – this may not contain all units of `foreigners_2`,
- `block` – block ID,
- `dist` – distance between objects.

```
head(result_reclin$result)
```

```
#>        x     y block      dist
#>    <int> <int> <num>     <num>
#> 1:     3     1     1 0.2216882
```

```
#> 2:     3     2     1 0.2122737
#> 3:    21     3     2 0.1172652
#> 4:    57     4     3 0.1863238
#> 5:    57     5     3 0.1379310
#> 6:    61     6     4 0.2307692
```

Let's examine the first pair. Obviously, there are typos in the `fname` and `surname`. Nevertheless, the pair appears to be a match.

```
cbind(t(foreigners_1[3, 1:6]), t(foreigners_2[1, 1:6]))
```

```
#>         [,1]         [,2]
#> fname   "amio"       "amik"
#> sname   ""           ""
#> surname "maharrsmov" "maharramof"
#> date    "19390308"   "19390308"
#> region  ""           ""
#> country "031"        "031"
```

Now we use the `true_id` values to evaluate our approach.

```
matches <- merge(x = foreigners_1[, .(x = 1:.N, true_id)],
                 y = foreigners_2[, .(y = 1:.N, true_id)],
                 by = "true_id")
matches[, block := rleid(x)]
head(matches)
```

```
#> Key: <true_id>
#>    true_id     x     y block
#>      <num> <int> <int> <int>
#> 1:       2     3     1     1
#> 2:       2     3     2     1
#> 3:      20    21     3     2
#> 4:      56    57     4     3
#> 5:      56    57     5     3
#> 6:      60    61     6     4
```

We have 10,000 matched pairs. We use the `true_blocks` parameter in the `blocking` function to specify the true block assignments. We obtain the quality metrics for the assessment of record linkage.

```
result_2_reclin <- blocking(x = foreigners_1$txt, y = foreigners_2$txt, verbose = 1,
                            true_blocks = matches[, .(x, y, block)])
```

```
#> ===== creating tokens =====
#> ===== starting search (nnd, x, y: 100000, 10000, t: 1232) =====
#> ===== creating graph =====
```

```
result_2_reclin
```

```
#> ==========================================================
#> Blocking based on the nnd method.
#> Number of blocks: 6469.
#> Number of columns used for blocking: 1232.
#> Reduction ratio: 0.9999.
#> ==========================================================
```

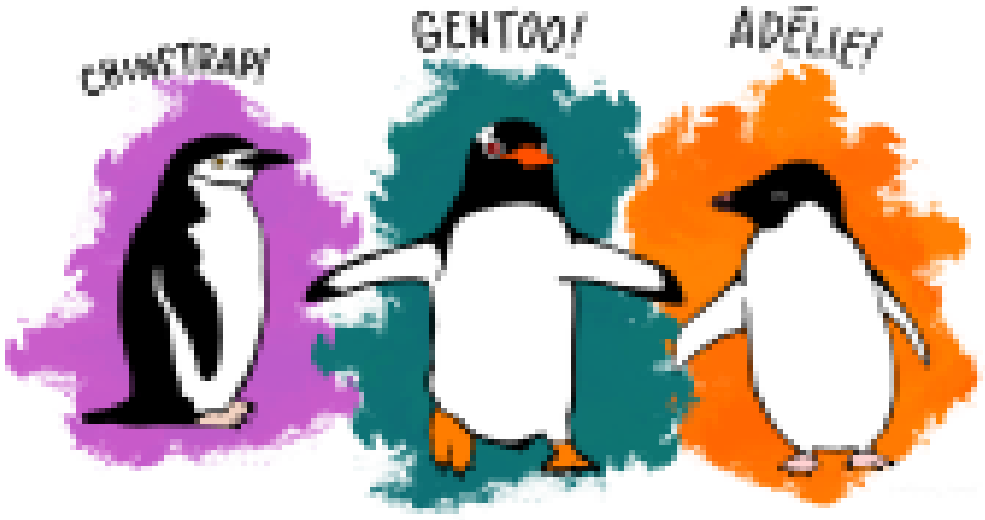

Figure 1: Artwork by allison_horst

Table 1: A basic table

species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Adelie	Torgersen	39.1	18.7	181	3750	male	2007
Adelie	Torgersen	39.5	17.4	186	3800	female	2007
Adelie	Torgersen	40.3	18.0	195	3250	female	2007
Adelie	Torgersen	NA	NA	NA	NA	NA	2007
Adelie	Torgersen	36.7	19.3	193	3450	female	2007
Adelie	Torgersen	39.3	20.6	190	3650	male	2007

```
#> Distribution of the size of the blocks:
#>    2    3    4    5    6    7
#> 3916 1604  926   19    2    2
#> ========================================================
#> Evaluation metrics (standard):
#>      recall   precision         fpr         fnr    accuracy specificity
#>     96.7782     78.7000      0.0038      3.2218     99.9957     99.9962
#>    f1_score
#>     86.8079
```

6 Customizing tooltip design with ToOoOlTiPs

ToOoOlTiPs is a packages for customizing tooltips in interactive graphics, it features these possibilities.

7 A gallery of tooltips examples

The **palmerpenguins** data (Horst et al., 2020) features three penguin species which has a lovely illustration by Alison Horst in Figure 1.

Table 1 prints at the first few rows of the `penguins` data:

Figure 2 shows an plot of the penguins data, made using the **ggplot2** package.

```
penguins %>%
  ggplot(aes(x = bill_depth_mm, y = bill_length_mm,
             color = species)) +
  geom_point()
```

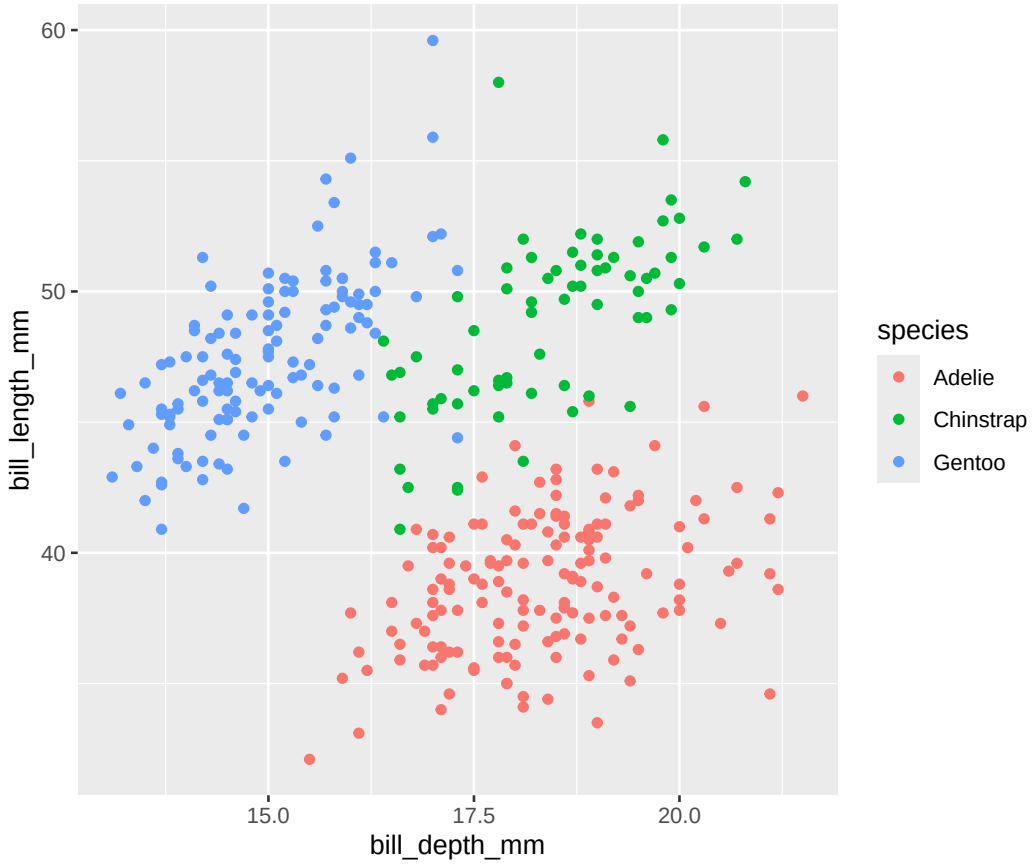

Figure 2: A basic non-interactive plot made with the ggplot2 package on palmer penguin data. Three species of penguins are plotted with bill depth on the x-axis and bill length on the y-axis. Visit the online article to access the interactive version made with the plotly package.

8 Summary

We have displayed various tooltips that are available in the package **ToOoOlTiPs**.

9 Acknowledgements

Work on this package is supported by the National Science Centre, OPUS 20 grant no. 2020/39/B/HS4/00941

References

J. Cheng and C. Sievert. *crosstalk: Inter-Widget Interactivity for HTML Widgets*, 2021. URL https://CRAN.R-project.org/package=crosstalk. R package version 1.1.1. [p1]

A. M. Horst, A. P. Hill, and K. B. Gorman. *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*, 2020. URL https://allisonhorst.github.io/palmerpenguins/. R package version 0.1.0. [p5]

C. Sievert. *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC, 2020. ISBN 9781138331457. URL https://plotly-r.com. [p1]

E. Wang and D. Cook. Conversations in time: interactive visualisation to explore structured temporal data. *The R Journal*, 2021. doi: 10.32614/RJ-2021-050. URL https://journal.r-project.org/archive/2021/RJ-2021-050/index.html. [p1]

Maciej Beręsewicz
University of Economics and BusinessStatisical Office in Poznań
Department of Statistics, Poznań, Poland
Centre for the Methodology of Population Studies
https://maciejberesewicz.com
ORCiD: 0000-0002-8281-4301
maciej.beresewicz@poznan.pl

Adam Struzik
Adam Mickiewicz UniversityStatisical Office in Poznań
Department of Mathematics, Poznań, Poland
Centre for Urban Statistics
adastr5@st.amu.edu.pl