Two Applications of the TSP for Data Analysis

Tools for Intelligent Data Analysis, GfKl 2007

Michael Hahsler and Kurt Hornik

Vienna University of Economics and Business Administration

Freiburg, March 9, 2007

Outline

1. The traveling salesperson problem
2. TSP package for R
3. Applications for data analysis

The Traveling Salesperson Problem

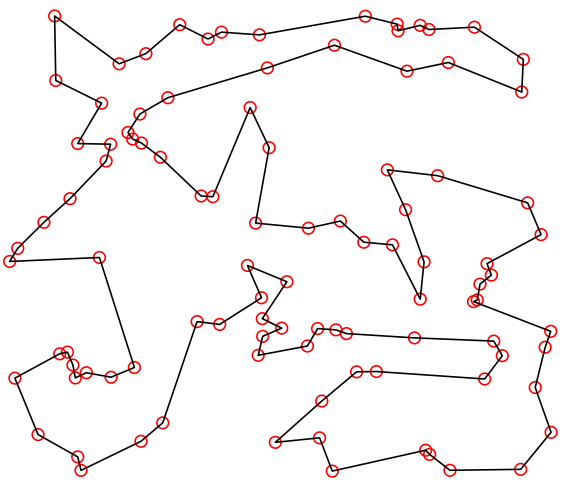
Introduction

The traveling salesperson problem (TSP; Lawler, Lenstra, Rinnooy Kan, and Shmoys, 1985; Gutin and Punnen, 2002) is a well known and important combinatorial optimization problem.

The goal is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city.

Applications (Lenstra and Kan, 1975; Punnen, 2002):

- vehicle routing
- computer wiring
- machine sequencing and job scheduling
- data analysis: reordering and clustering



Definition

The distances between n cities are stored in a distance matrix

$$\mathbf{D} = [d_{ij}],\ i, j = 1 \ldots n,\ d_{ii} = 0$$

A ***tour*** is a cyclic permutation π of $\{1, 2, \ldots, n\}$ where $\pi(i)$ represents the city that follows city i on the tour. The traveling salesperson problem is then the optimization problem to find a permutation π^* that minimizes the ***length of the tour*** denoted by

$$\sum_{i=1}^{n} d_{i\pi(i)}.$$

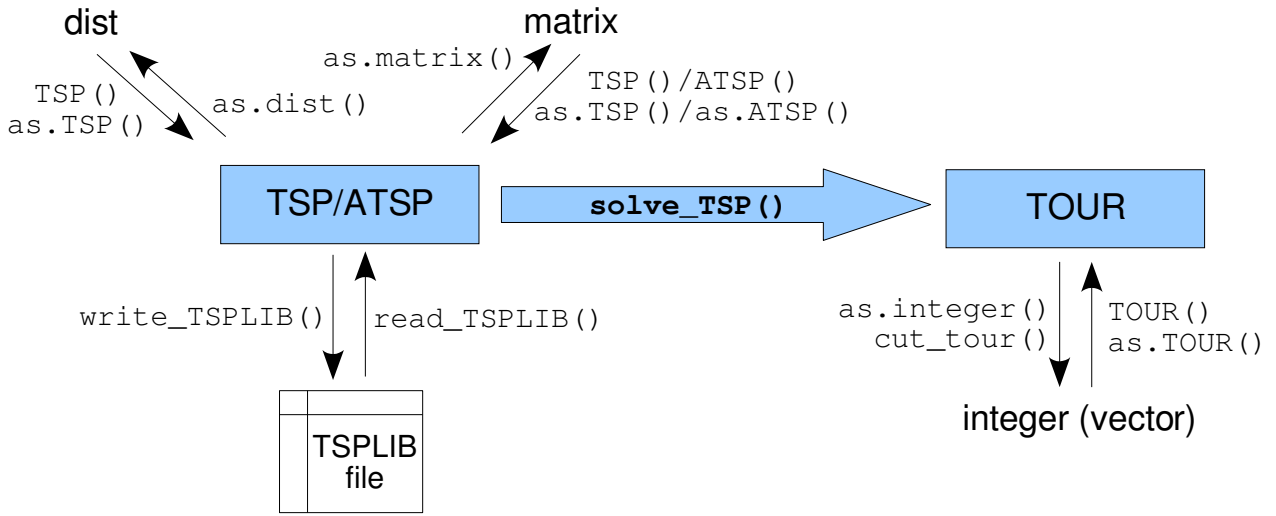
Finding the optimal permutation vector π^* in the set Π of all $(n-1)!$ possible cyclic permutations is NP-complete (Johnson and Papadimitriou, 1985).

Solving a TSP

1. **Exact solution**
   - ***Dynamic programming*** (Held and Karp, 1962) for small instances (< 100 cities)
   - ***Branch-and-cut*** (Padberg and Rinaldi, 1990) for increasingly larger instances (up to $24,978$ cities solved by Concorde)
2. **Heuristics**
   - ***Construction:*** nearest neighbor, insertion algorithms (Rosenkrantz, Stearns, and Philip M. Lewis, 1977),...
   - ***Local improvement:*** k-Opt, Lin-Kernighan (Lin and Kernighan, 1973),...

The TSP package for R

Package infrastructure



The **TSP** package contains the basic infrastructure to conveniently handle the needed data structures.

The package is available at `http://CRAN.R-project.org`

Solvers

All TSP solvers in package **TSP** use the simple common interface:

```
solve_TSP(x, method, control)
```

Currently implemented methods:

Algorithm	Method argument	Applicable to
Construction heuristics		
Nearest neighbor algorithm	`"nn"`	TSP/ATSP
Repetitive nearest neighbor algorithm	`"repetitive_nn"`	TSP/ATSP
Nearest insertion	`"nearest_insertion"`	TSP/ATSP
Farthest insertion	`"farthest_insertion"`	TSP/ATSP
Cheapest insertion	`"cheapest_insertion"`	TSP/ATSP
Arbitrary insertion	`"arbitrary_insertion"`	TSP/ATSP
Improvement heuristics		
2-Opt improvement heuristic	`"2-opt"`	TSP/ATSP
Chained Lin-Kernighan	`"linkern"`	TSP
Exact Solver		
Concorde TSP solver	`"concorde"`	TSP

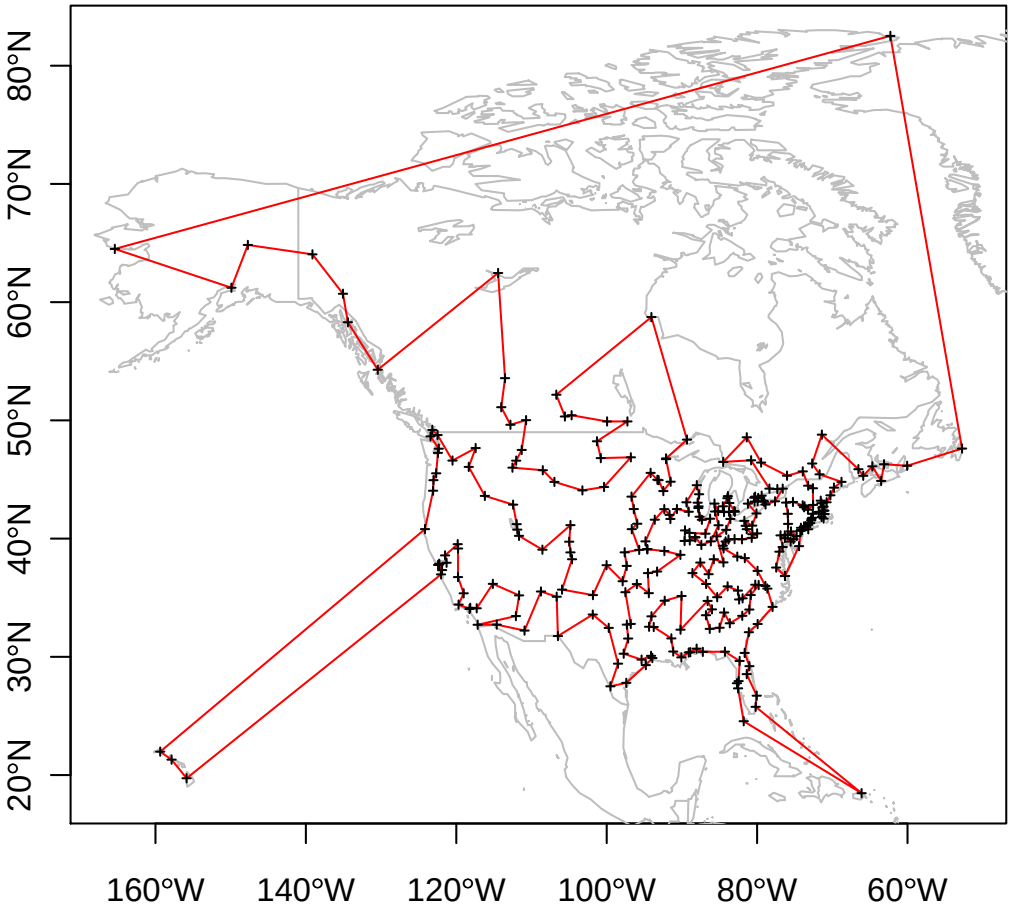
Usage

```
> library("TSP")
> data("USCA312")
> tsp <- TSP(USCA312)
> tsp

object of class 'TSP'
312 cities (distance 'euclidean')

> tour <- solve_TSP(tsp, method = "2-opt")
> tour

object of class 'TOUR'
result of method '2-opt' for 312 cities
tour length: 41381
```


80°N
70°N
60°N
50°N
40°N
30°N
20°N
160°W
140°W
120°W
100°W
80°W
60°W

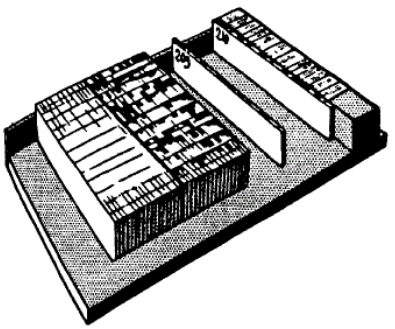
Applications

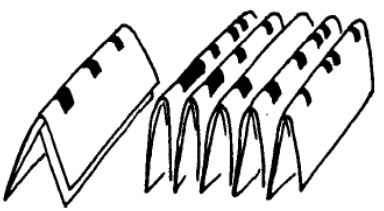
Matrix visualization

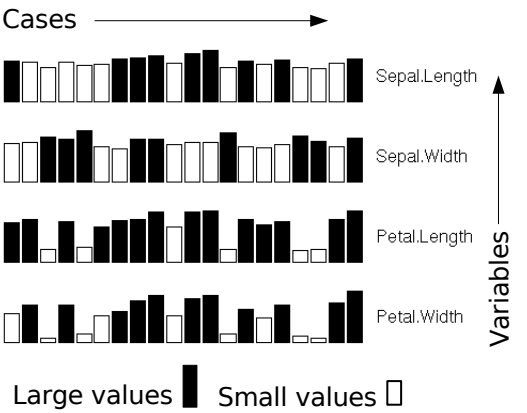
Bertin (1999) introduced permutation matrices to analyze multivariate data with medium to low sample size.

Idea: make a data matrix more understandable by simultaneously rearranging rows (variables) and columns (cases) + grouping

Mechanical tools (Bertin, 1999) and a modern display option:







Matrix visualization (cont.)

Aim: Obtain a more homogeneous structure.

- Purity function: $\phi = \Phi(\mathbf{X})$
- Permutation of rows and columns: $\mathbf{X_p} = \pi(\mathbf{X})$

$\Rightarrow$ find π^* which maximizes $\Phi(\pi(\mathbf{X}))$ and display $\pi^*(\mathbf{X})$

A possible purity function Φ:
Given distances between rows $\mathbf{D_r}$ and columns $\mathbf{D_c}$, define purity as the sum of distances of adjacent rows/columns.

$\Rightarrow$ finding π^* means solving two (independent) TSPs

Matrix visualization (cont.)

We use the results of 8 referenda for 41 Irish communities.
The values are scaled using variable-wise ranking.

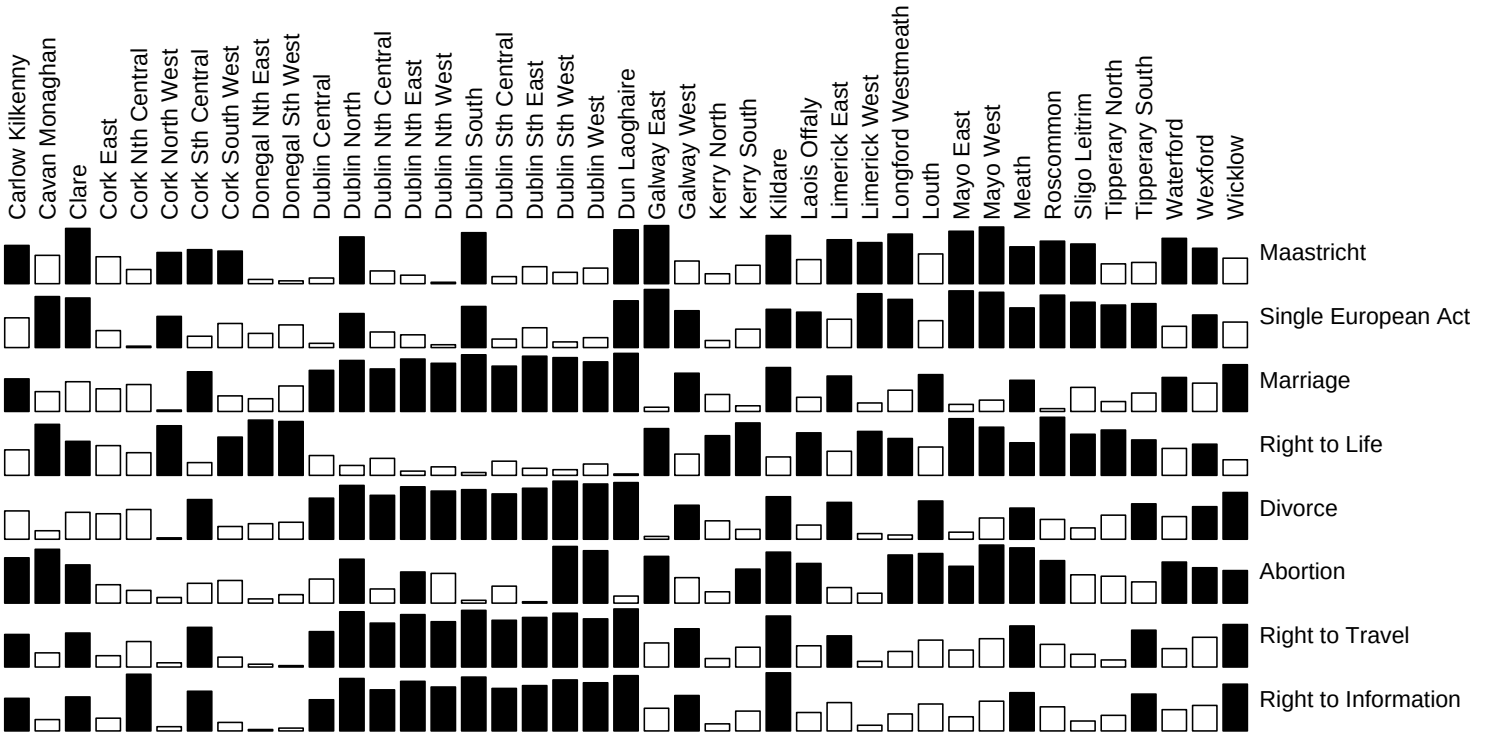
```
> load("Irish.rda")
> scale_by_rank <- function(x) apply(x, 2, rank)
> orig_matrix <- scale_by_rank(Irish[, -6])
```

Find (a near) optimal permutation by solving two TSPs.
As distances we use the sum of rank differences.

```
> tour_cases <- solve_TSP(TSP(dist(orig_matrix, "minkowski",
+     p = 1)), method = "2-opt")
> tour_variables <- solve_TSP(TSP(dist(t(orig_matrix),
+     "minkowski", p = 1)), method = "2-opt")
> rearranged_matrix <- orig_matrix[tour_cases, tour_variables]
```

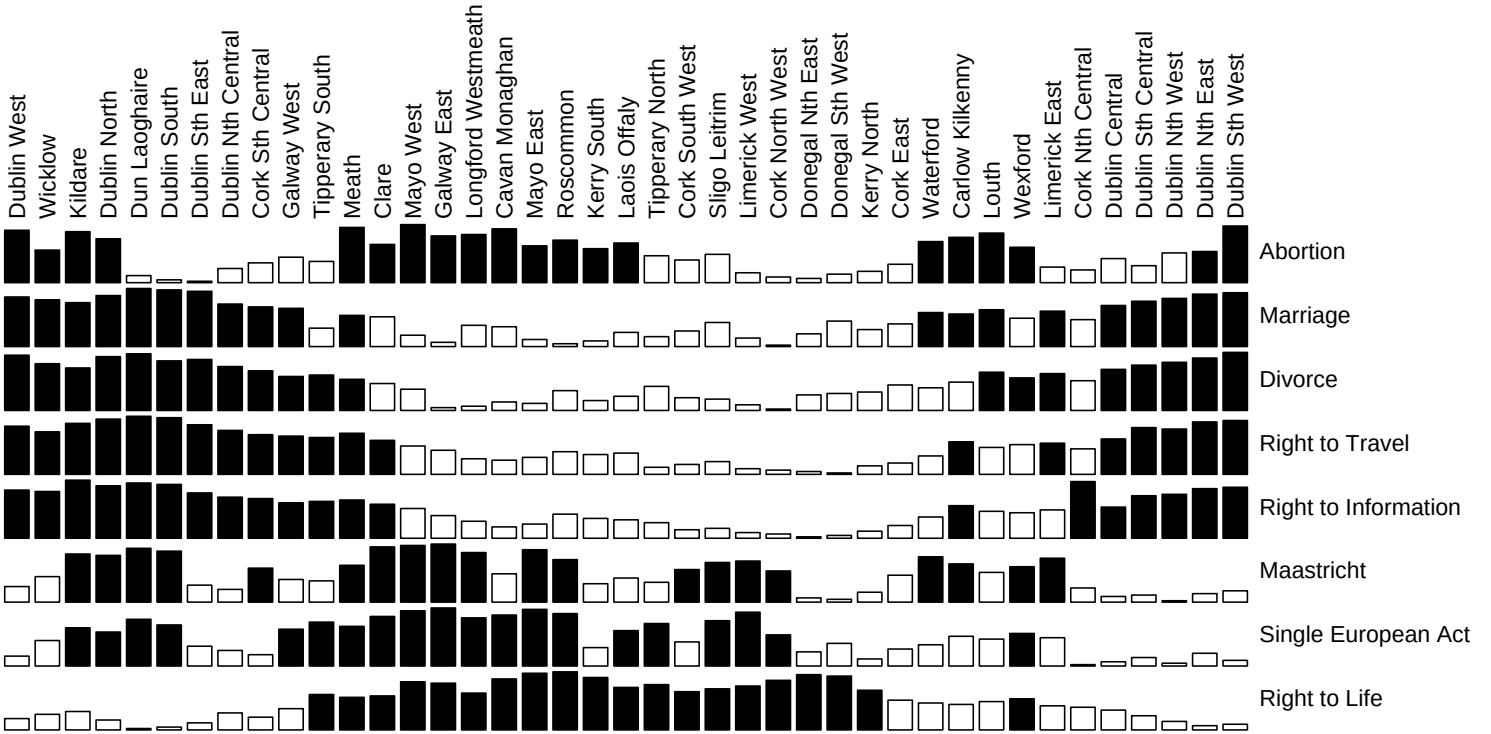
Matrix visualization (cont.)

Original matrix



Matrix visualization (cont.)

Rearranged matrix



Clustering

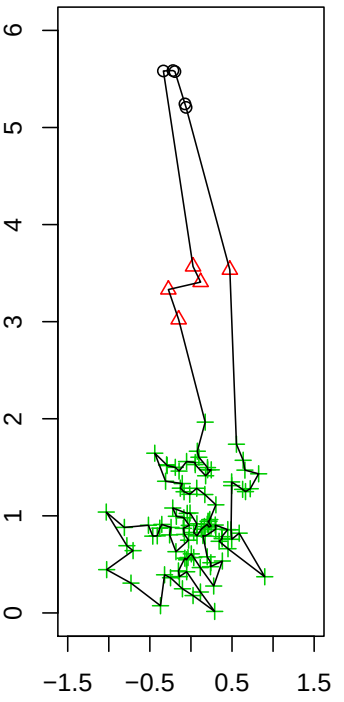
Solving a TSP to obtain a clustering was suggested several times in the literature (see, e.g., Lenstra, 1974; Alpert and Kahng, 1997; Johnson, Krishnan, Chhugani, Kumar, and Venkatasubramanian, 2004).

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmin}} \left(\sum_{i=1}^{n} d_{i\pi(i)} \right)$$

Where Π is the set of all cyclic permutations and π^* minimizes the sum of distances between neighboring objects.

The idea is that from one cluster to the next larger "jumps" are necessary and thus ***objects in the same cluster are visited in consecutive order.***

Clusters can later be separated.



Clustering (cont.)

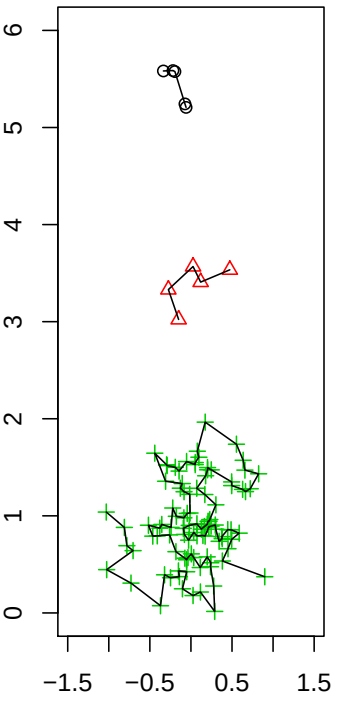
Climer and Zhang (2006) suggest for clustering with k clusters to add k ***dummy cities*** which have constant distance c to all other cities and are infinitely far from each other.

After solving the new TSP, the dummy cities separate the clusters.

The objective is modified to

$$\min\left(\sum_{i=1}^{k}\sum_{j=u_i}^{v_i-1} d_{j,j+1}\right),$$

where u_i and v_i are the first and the last object in cluster i, respectively.



Clustering (cont.)

The iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris.

The species are:

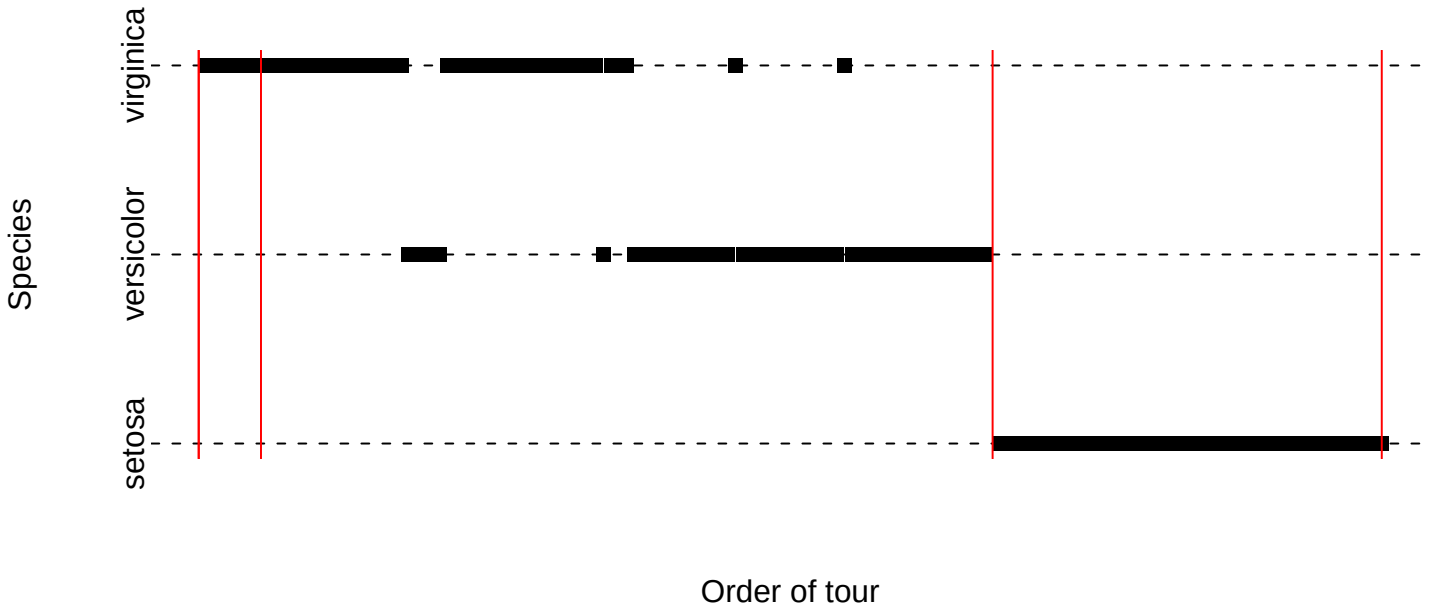
- Iris setosa
- Iris versicolor
- Iris virginica

Perform rearrangement clustering with 3 clusters:

```
> library("TSP")
> data("iris")
> tsp <- TSP(dist(iris[-5]), labels = iris[, "Species"])
> tsp_dummy <- insert_dummy(tsp, n = 3, label = "boundary")
> tour <- solve_TSP(tsp_dummy)
```

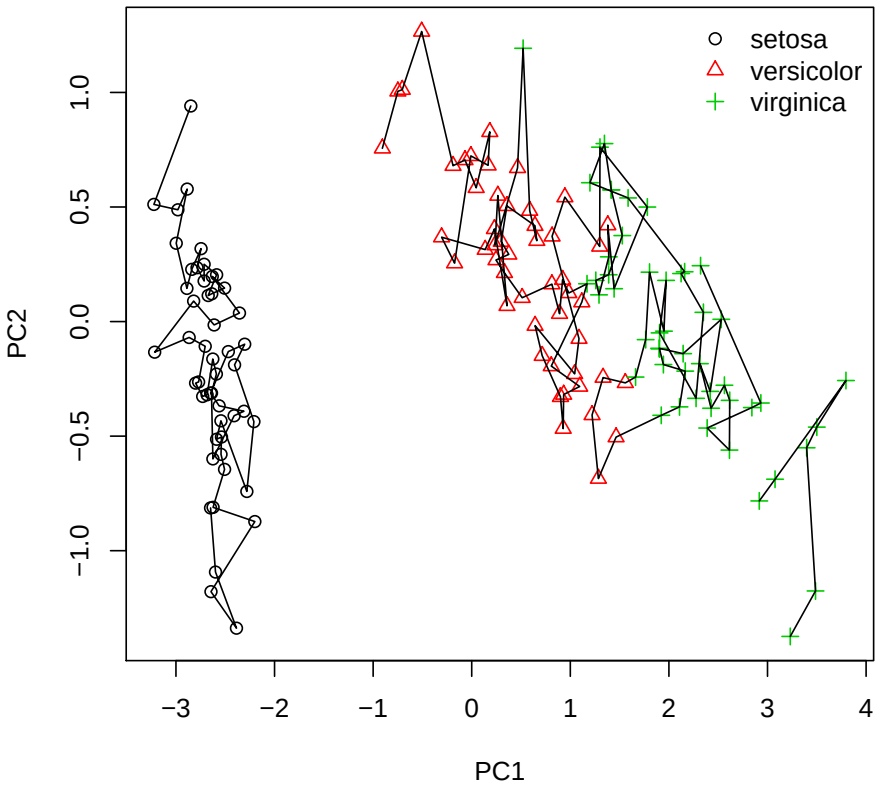
Clustering (cont.)

The result of clustering:



Clustering (cont.)

Data points are projected on the first 2 principal components.
Lines represent the 3 parts of the TSP tour.



Conclusion

- Solving TSPs is necessary for many applications.
- Modern, efficient algorithms (and heuristics) are available.
- The TSP package provides an easy-to-use infrastructure to develop applications.
- The TSP package enhances R's capabilities as a data analysis tool.

References

C. J. Alpert and A. B. Kahng. Splitting an ordering into a partititon to minimize diameter. ***Journal of Classification***, 14(1):51–74, 1997.

J. Bertin. ***Graphics and graphic information processing***. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999. ISBN 1-55860-533-9.

S. Climer and W. Zhang. Rearrangement clustering: Pitfalls, remedies, and applications. ***Journal of Machine Learning Research***, 7:919–943, June 2006.

G. Gutin and A. Punnen, editors. ***The Traveling Salesman Problem and Its Variations***, volume 12 of ***Combinatorial Optimization***. Kluwer, Dordrecht, 2002.

M. Held and R. Karp. A dynamic programming approach to sequencing problems. ***Journal of SIAM***, 10: 196–210, 1962.

D. Johnson and C. Papadimitriou. Computational complexity. In Lawler et al. (1985), chapter 3, pages 37–86.

D. Johnson, S. Krishnan, J. Chhugani, S. Kumar, and S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In ***Proceedings of the 30th VLDB Conference***, pages 13–23, 2004.

E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. ***The Traveling Salesman Problem***. Wiley, New York, 1985.

J. Lenstra and A. R. Kan. Some simple applications of the travelling salesman problem. ***Operational Research Quarterly***, 26(4):717–733, November 1975.

J. K. Lenstra. Clustering a data array and the traveling-salesman problem. ***Operations Research***, 22(2): 413–414, 1974.

S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. ***Operations Research***, 21(2):498–516, 1973.

M. Padberg and G. Rinaldi. Facet identification for the symmetric traveling salesman polytope. ***Mathematical Programming***, 47(2):219–257, 1990. ISSN 0025-5610.

A. Punnen. The traveling salesman problem: Applications, formulations and variations. In Gutin and Punnen (2002), chapter 1, pages 1–28.

D. J. Rosenkrantz, R. E. Stearns, and I. Philip M. Lewis. An analysis of several heuristics for the traveling salesman problem. ***SIAM Journal on Computing***, 6(3):563–581, 1977.