Chapter 11: Parallel Axis Plots

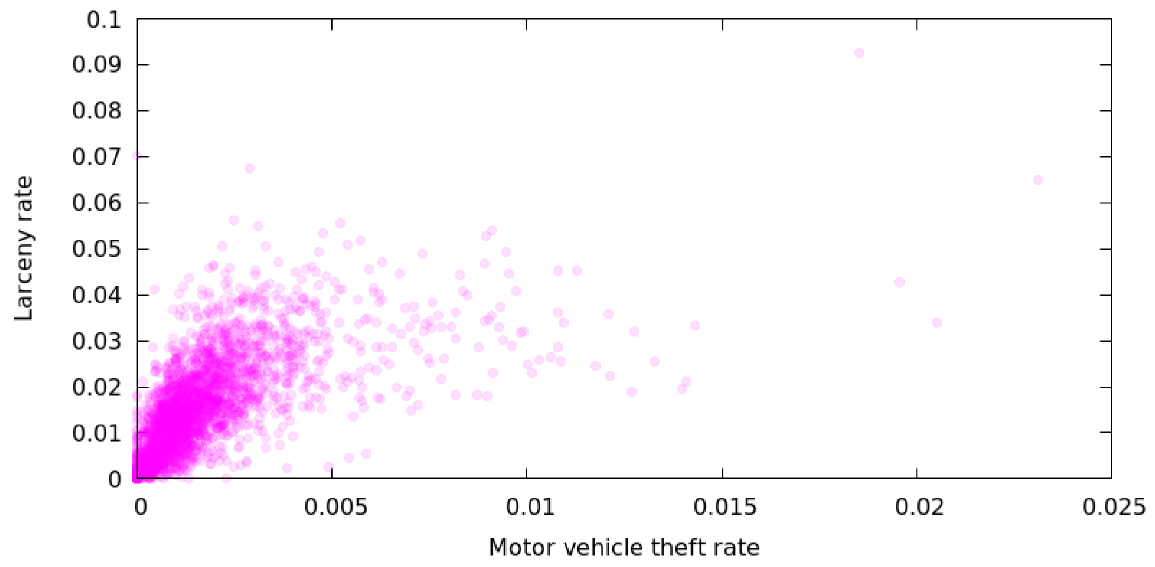
Gnuplot version 5 gained the ability to create *parallel axis plots*. These are sufficiently dissimilar from gnuplot's other plot types to merit their own chapter. Sometimes called "parallel coordinate plots," these plots are a way to visualize data in multiple dimensions. They are rather unusual, and you may never have seen one before. We'll approach them by way of some U.S. census data, courtesy of https://www.census.gov/support/USACdataDownloads.html. We've assembled some data, from different census files, into one data file called "census.dat". If you want to play along, and you should, you can find this file in the usual place—the download area where you can retrieve copies of this book and all the other supporting material. The file is in tab-separated-value, or TSV, format; we'll have to let gnuplot know about that, so it can read the data into the proper columns.

The file has 3,199 rows: one for each county in the United States, plus rows for state totals, the U.S. total, and the first row for column names. We've "commented-out" the state total rows and the U.S. total row. The first column gives the name of the county, but we won't be including those names in the plots. The remaining columns give data for the year 2005, per county; they are, in order, total larcenies, total murders, total motor vehicle thefts, total robberies, percentage of people under the age of 18 who do not have health insurance, and the estimated population as of July of that year.

When exploring a set of data like this, one can start by looking for trends and associations between quantities, always remembering that, as the saying goes, "correlation does not imply causation." Let's pick two columns and make a scatterplot. We'll plot motor vehicle thefts *versus* larcenies, dividing each quantity by the county population, in order to plot rates rather than raw counts. We'll make the data points transparent, by using a color specification with an alpha channel, to create a higher visual density in regions where many data points overlap. The simple script that does all this is

```
unset key
set datafile sep tab
set xlab  "Motor vehicle theft rate"
set ylab "Larceny rate"
plot "census.dat" u ($4/$7):($2/$7) pt 7 lc "#e0ff00ff"
```

There are no new commands here. The result of the script is the figure below. It may not be surprising that two types of property crime are pretty highly correlated.



We could continue in this fashion, plotting various combinations of pairs of quantities; essentially looking at 2D slices of the data, which can be thought of as 3,000 points embedded within a 6D space: one dimension for each statistical quantity, not including the county name. We could even encode additional quantities, as we've done in some previous chapters, using dot size or color, perhaps combined with a 3D perspective plot, to look at more than two dimensions at once. But this does not treat every dimension the same, encoding some into position, others into color, etc.

The parallel axis plot is one approach to visualizing multidimensional data. The principles of its construction are simple, but interpretation can take practice. For each dimension, the plot has a vertical axis; these parallel axes are spaced equally, and may or may not feature tics or labels. For each data point, a line is drawn connecting its values along each of the axes. Points that are close together in the multidimensional space will thus lead to lines that are close together, allowing you to see clusters or associations in the data, or so it is hoped. These visualizations are probably most effective when the number of data points is much larger than the number of dimensions, as in our examples in this chapter. The maximum number of parallel axes that you can use in your plots is fixed in gnuplot at compile time. You can view this number by typing `show version long` at the interactive prompt and looking for the MAX_PARALLEL_AXES parameter; in my version this value = 7.

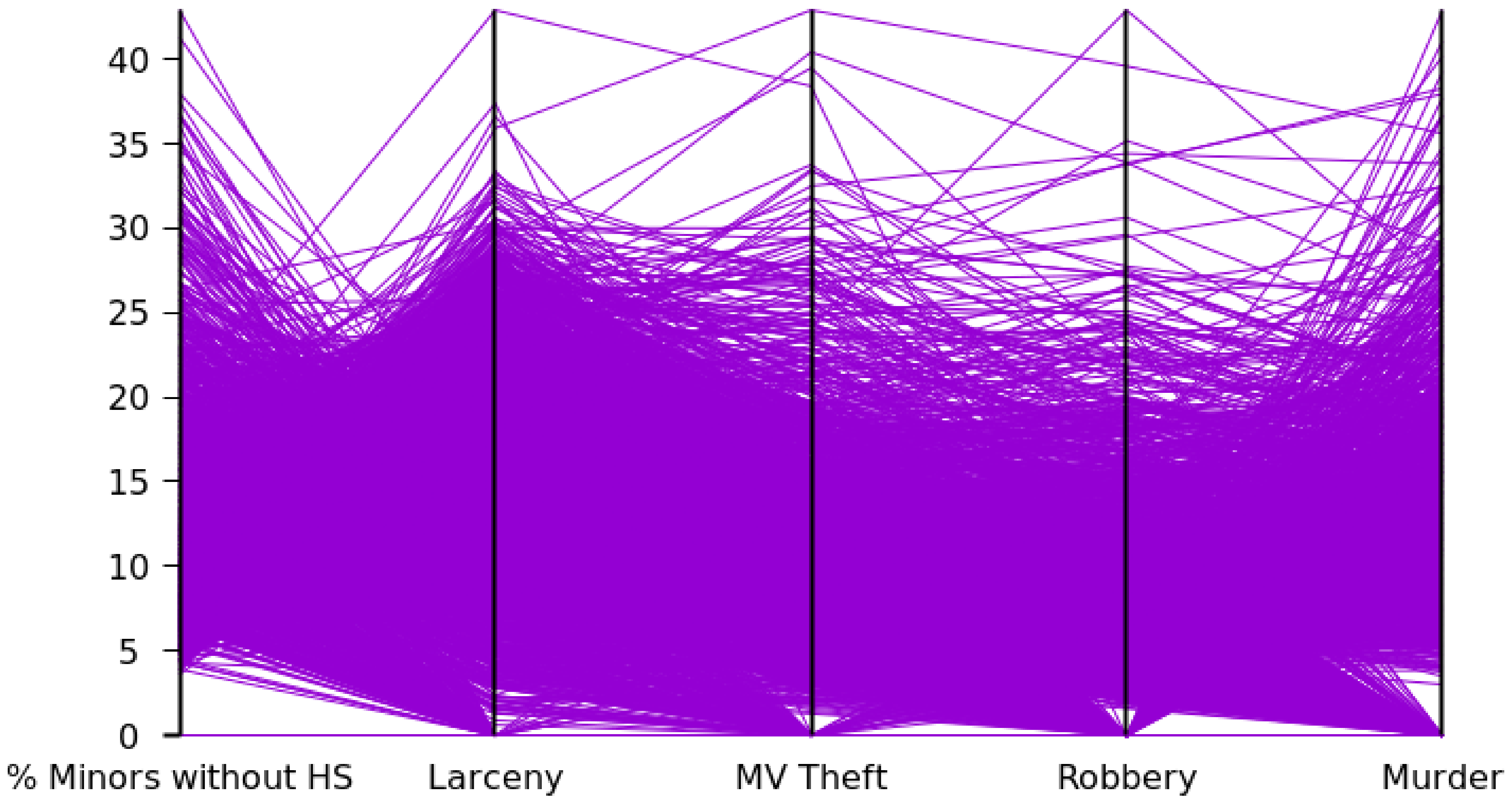
To make a parallel axis plot, use the `with parallel` clause within the plot command. It's usually best to disable the border and the default ytics. By default, the parallel axes will be unadorned, but you can put tics on them if you like. The command for that is the first highlighted command here. In previous examples we've used gnuplot's default whitespace separator for data columns, but for this one our data file uses tabs; we need to tell gnuplot about this, which is done in the second highlighted command.

Since this data tends to be crowded at low values with a scattering of high values, it's helpful to scale the data to spread it out more uniformly. This is often done with log scaling, but here we defined a scaling function, `s(x)`, that takes the square root of the data.

Nevertheless, because of the large number of datapoints, the plot is a nearly uniform mass, and doesn't convey much insight. We'll try to handle this problem in the following examples.

```
unset key
unset border; unset ytics
set paxis 1 tics 5
s(x) = x**.5
set datafile sep tab
set xtics ("%% Minors without HS" 1, "Larceny" 2,\
   "MV Theft" 3,"Robbery" 4, "Murder" 5)
plot "census.dat" u 6:(s($2/$7)):(s($4/$7)):(s($5/$7)):(s($3/$7)) w parallel
```

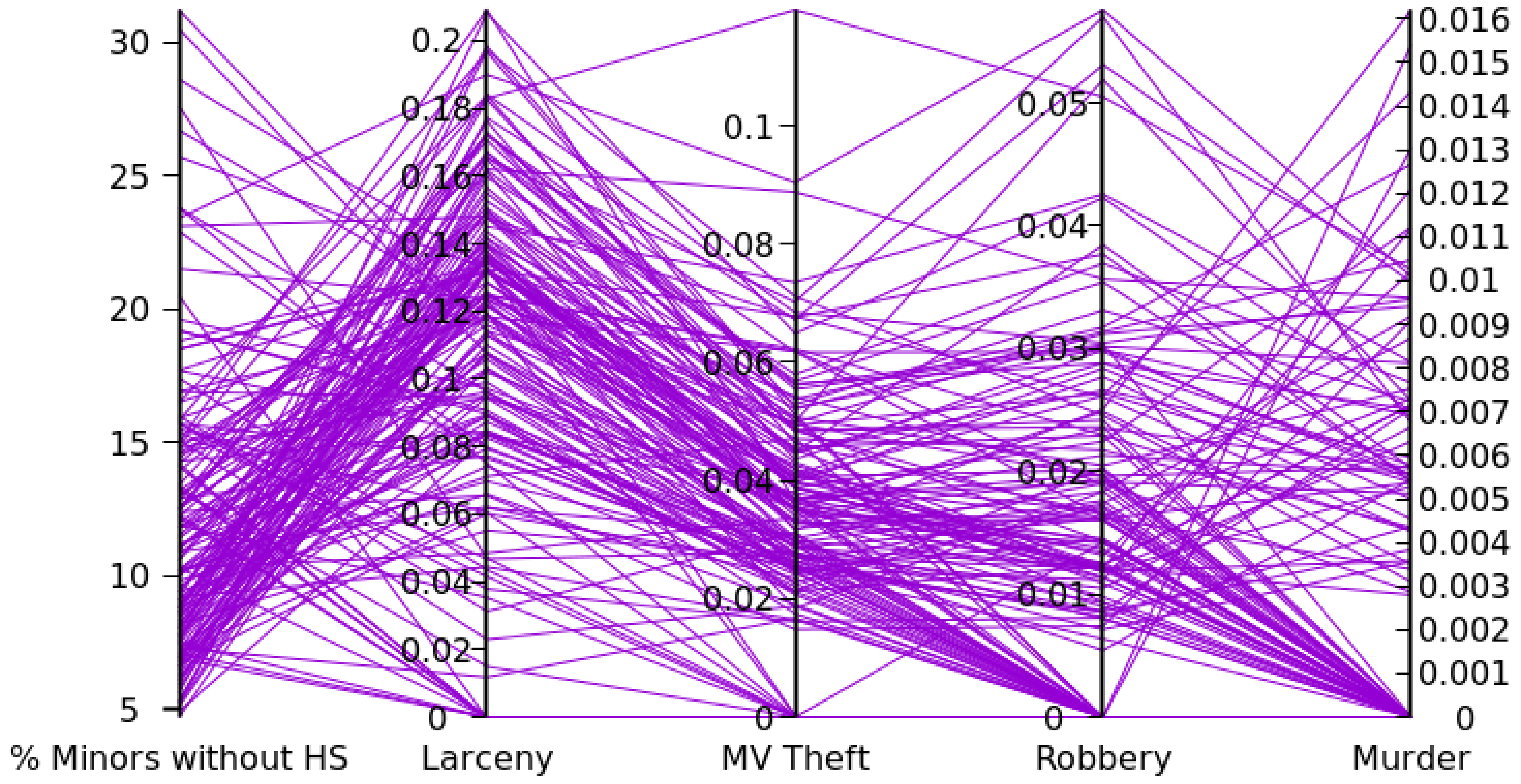
Open script


40
35
30
25
20
15
10
5
0
% Minors without HS
Larceny
MV Theft
Robbery
Murder

One way to deal with too much data is simply to skip some data points. In this example we make the same plot, but use the `every` command to only plot every 20th point. In addition, we've added tics to the remaining axes. You can use the same tic commands, including the label offset used in this script, as in `set ytics`, etc. This random subsample of the data does reveal more structure than the full plot above, but it would be preferable to find a way to see patterns without leaving out points, if possible. We'll return to this shortly.

```
unset key
unset border; unset ytics
set paxis 1 tics 5
set paxis 2 tics .02
set paxis 3 tics .02
set paxis 4 tics .01
set paxis 5 tics .001 offset 5
s(x) = x**.5
set datafile sep tab
set xtics ("%% Minors without HS" 1, "Larceny" 2,\
   "MV Theft" 3,"Robbery" 4, "Murder" 5)
plot "census.dat" u 6:(s($2/$7)):(s($4/$7)):(s($5/$7)):(s($3/$7)) every 20\
   w parallel
```

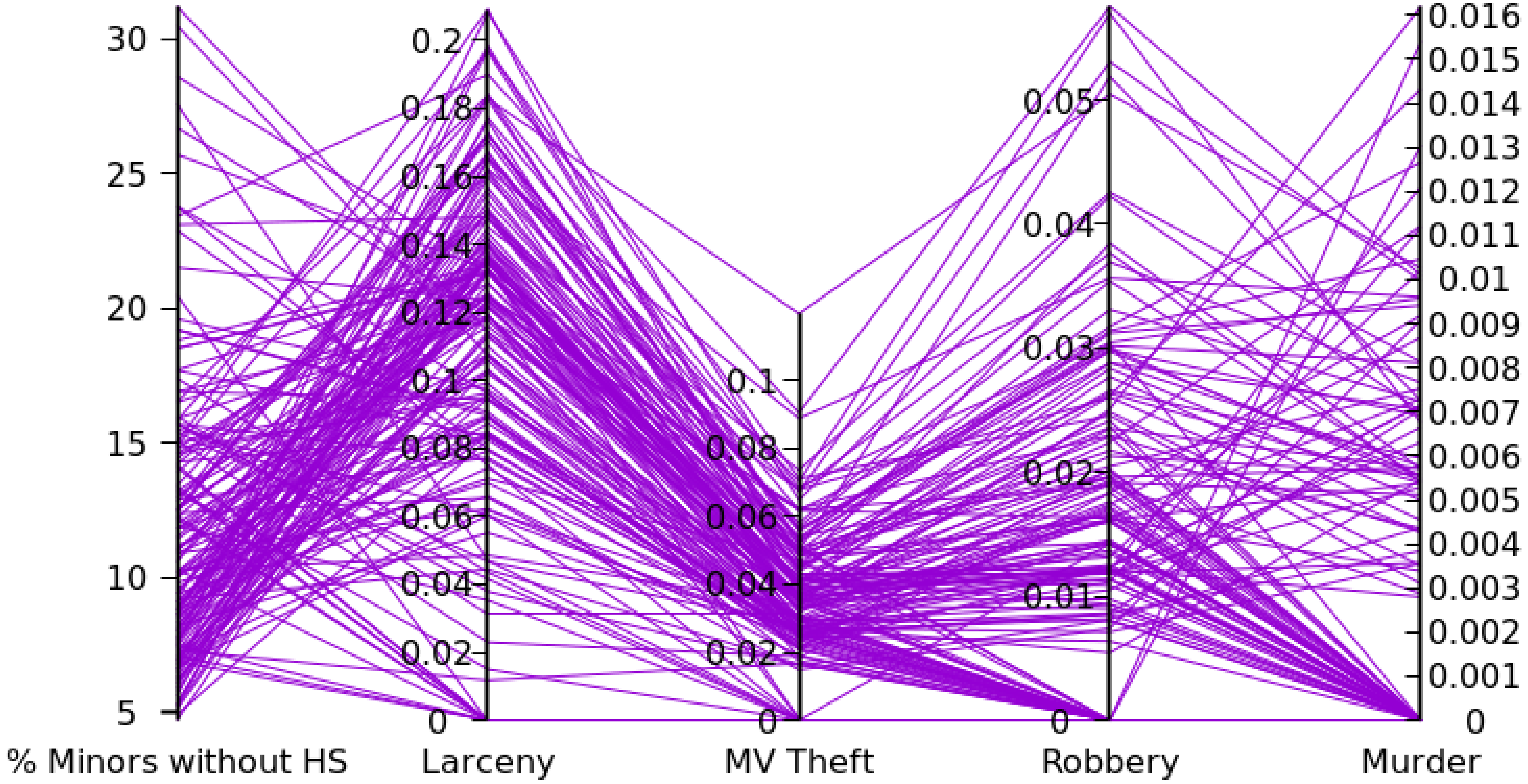
Open script


30
25
20
15
10
5
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0.1
0.08
0.06
0.04
0.02
0
0.05
0.04
0.03
0.02
0.01
0
0.016
0.015
0.014
0.013
0.012
0.011
0.01
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
% Minors without HS
Larceny
MV Theft
Robbery
Murder

Before we return to the issue of revealing structure in the data, it is sometimes useful to force some of the axes to have their tics' values aligned. To do this, give the axes in question the same range and tic specification. We've repeated the previous plot here, forcing the Larceny and MV Theft axes to be aligned. Note that gnuplot will not extend a paxis beyond the data, no matter how you set the range and tics, unlike a normal x- or y-axis. This is why the third axis in this example plot is shorter than the others.

```
unset key
unset border; unset ytics
set paxis 1 tics 5
set paxis 2 tics .02
set paxis 3 tics .02
set paxis 4 tics .01
set paxis 5 tics .001 offset 5
set paxis 2 range [0: 0.21]
set paxis 3 range [0: 0.21]
s(x) = x**.5
set datafile sep tab
set xtics ("%% Minors without HS" 1, "Larceny" 2,\
    "MV Theft" 3,"Robbery" 4, "Murder" 5) nomirror
plot "census.dat" u 6:(s($2/$7)):(s($4/$7)):(s($5/$7)):(s($3/$7)) every 20\
    w parallel
```

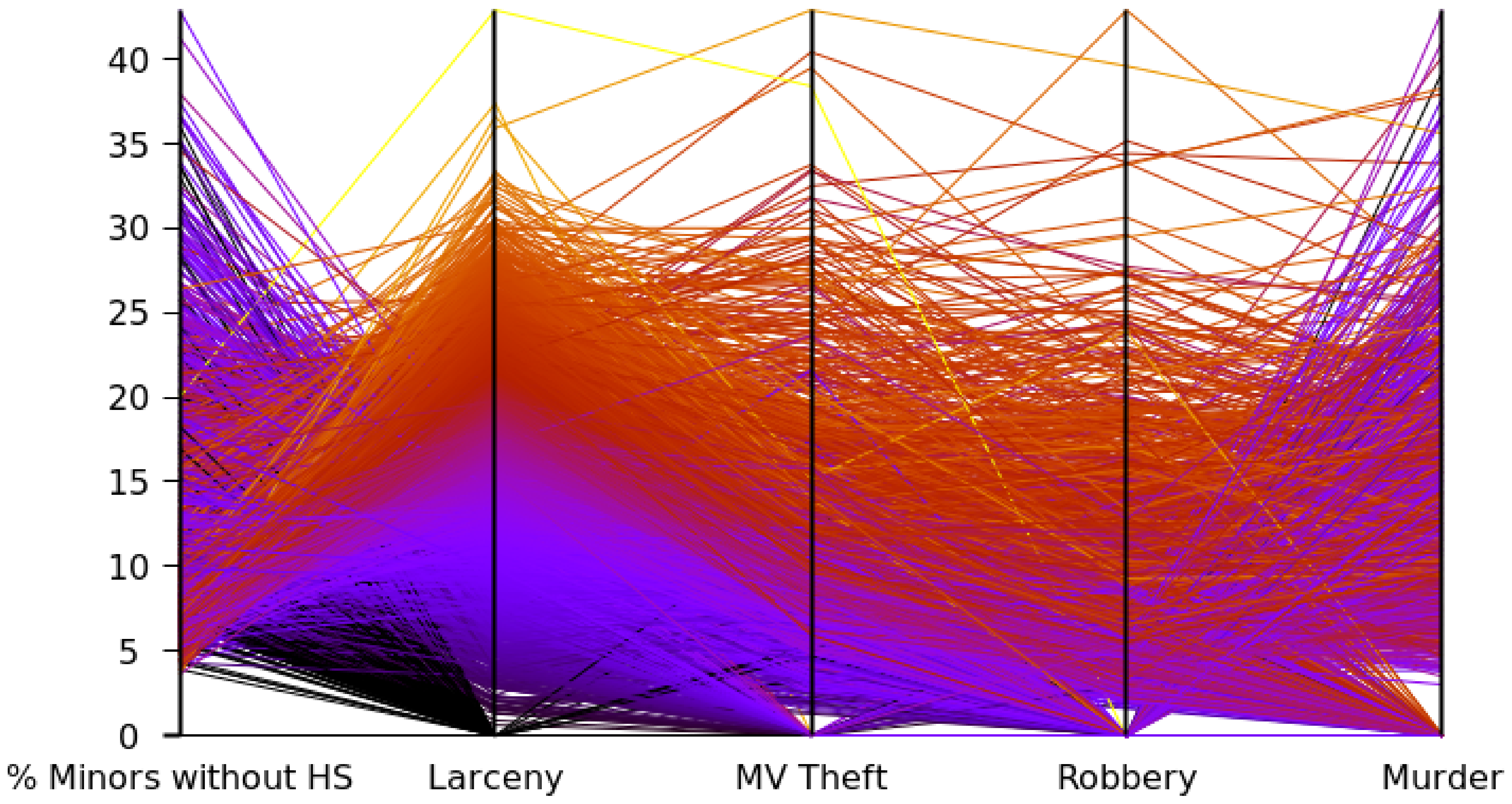
Open script


30
25
20
15
10
5
0.2
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0.1
0.08
0.06
0.04
0.02
0
0.05
0.04
0.03
0.02
0.01
0
0.016
0.015
0.014
0.013
0.012
0.011
0.01
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
% Minors without HS
Larceny
MV Theft
Robbery
Murder

One way to reveal structure in the data is to color datapoints by value. We've done this in other types of plots, using the `linecolor pal` clause in the `plot` or `splot` commands. In a parallel axis plot this will assign colors, taken from the active palette, to the lines according to the value in an extra column in the `using` clause. In this example we are coloring the data according to the value on the Larceny axis, using the default rainbow palette. Notice how this simple coloring of the data turns the undifferentiated mass of our first parallel axis plot above into a plot in which we can see some patterns, even though, as in the first graph, all of the data is plotted.

```
unset key
unset colorbox
unset border; unset ytics
set paxis 1 tics 5
s(x) = x**.5
set datafile sep tab
set xtics ("%% Minors without HS" 1, "Larceny" 2,\
    "MV Theft" 3,"Robbery" 4, "Murder" 5) nomirror
plot "census.dat" u 6:(s($2/$7)):(s($4/$7)):(s($5/$7)):(s($3/$7)):(s($2/$7))\
    w parallel lc pal
```

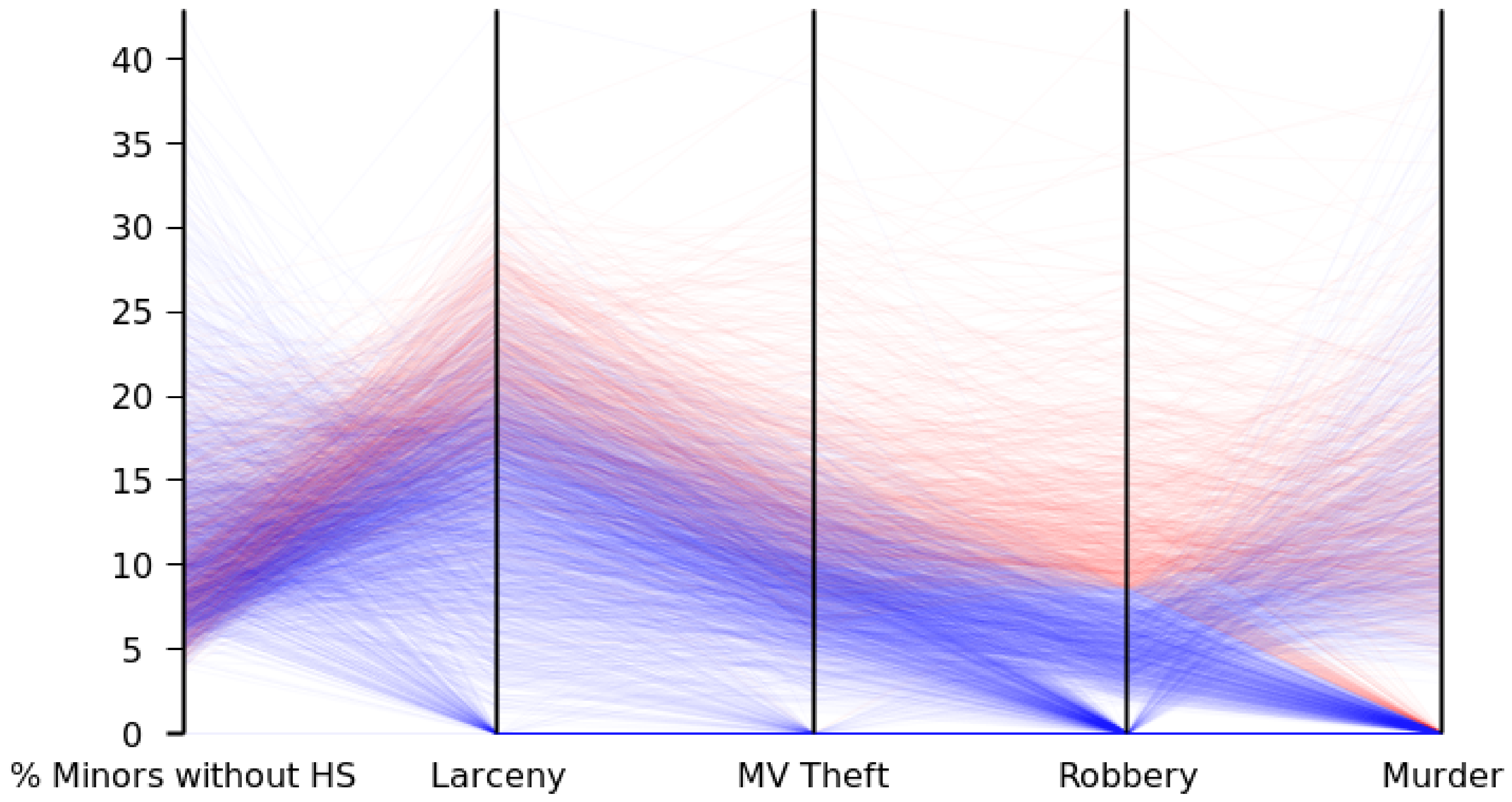
Open script


40
35
30
25
20
15
10
5
0
% Minors without HS
Larceny
MV Theft
Robbery
Murder

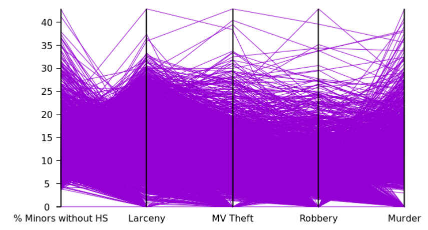
Another technique often used with parallel coordinate plots is to take the data falling within a particular, narrow range along a particular dimension, and give it a color or another visual attribute that contrasts with the data outside that range. In this way you can visualize the data within a narrow slab of dimension N-1 inside the N-dimensional space. Another useful visualization technique, particularly useful when plotting many data points, is to make make the lines partially transparent, allowing you to see behind them. In this example we use both techniques. First we define two linetypes with rather high transparency values of 0xFA; linetype 8 will be red and linetype 9 will be blue (to review the details of the various ways to define colors in gnuplot, type `help colorspec` at the interactive prompt). The clause `linecolor var` in a plot command colors whatever is being plotting according to an extra column in the `using` clause; but, instead of taking the colors from the palette, it uses the value in the extra column to select a `linetype`. Our extra column checks the value of the Robbery dimension (normalized by county population); if it is less than .0004, it evaluates to 9 (blue lines); otherwise, to 8 (red lines), using gnuplot's ternary syntax. Whether this plot, or the previous ones in this chapter, actually provide any insight into the data behind them, I'll leave to the judgement of the reader.

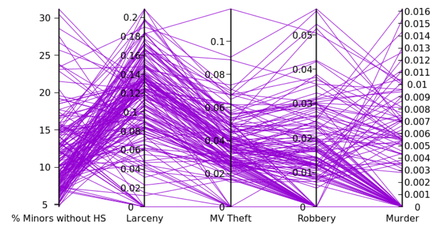
Open script

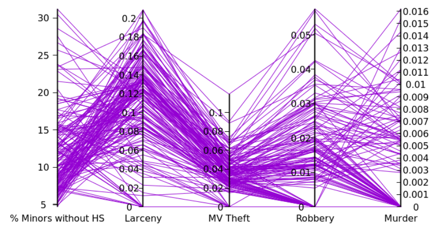
```
unset key
unset border; unset ytics
set paxis 1 tics 10
set lt 8 lc "#faff0000"
set lt 9 lc "#fa0000ff"
set paxis 1 tics 5
s(x) = x**.5
set datafile sep tab
set xtics ("%% Minors without HS" 1, "Larceny" 2,\
    "MV Theft" 3,"Robbery" 4, "Murder" 5)
plot "census.dat" u 6:(s($2/$7)):(s($4/$7)):(s($5/$7)):(s($3/$7)):\
    ($5/$7 < 0.0004 ? 9 : 8) w parallel lc var
```

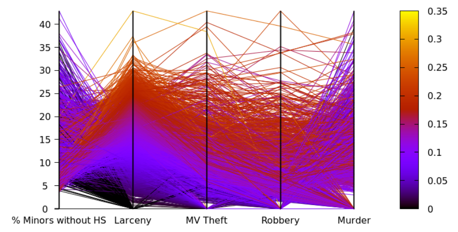

0
5
10
15
20
25
30
35
40
% Minors without HS
Larceny
MV Theft
Robbery
Murder

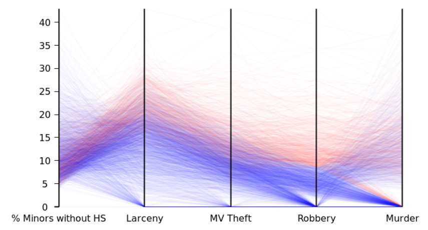
Index of Plots











Index