1. I ran all possible combinations of blocking characteristics for standard attribute-based blocking and Soundex blocking plus a single run of SLK blocking (as this uses set attributes). I assessed the evaluation metrics. Of all the scenarios I ran, a number of combinations yielded the highest pair quality while also showing relatively high pair completeness and reduction ratios. These were combinations that used Soundex blocking of middle name, last name, street address and suburb along with other variables. I decided however that just the combination of last name, street address and suburb were ideal as it was parsimonious (included the fewest variables of any combination so close to the top, good for generalisability) and had very high pair completeness along with high quality and a moderate to high reduction ratio. I decided to opt for just these in the interests of parsimony and because I thought the very marginal gains from other variables was just statistical noise (I wanted to prevent overfitting in case I had to join more data where I lacked ground-truth data).  
     
   The worst performing attributes tended to be combinations of few variables where many cases would share the same values and where values would be expected to change. For example there were very poor blocking results across postcode as many people live in a given postcode and people often move into new postcodes over their lives.

b) TODO: SIMULATE  
There is absolutely a trade-off here because creating many blocks can reduce the number of cases considered in linkage but in doing this we can also rule out some correct links. The correct trade-off depends on the computing power (and memory size) we have at our disposal. Theoretically, we should get the best results from no blocking at all, simply comparing all cases in our dataset with some kind of complex algorithm that can correctly identify approximate matches. However, this is not practical (my computer ran out of memory before it could properly join the data with no blocking) and so blocking can be seen as a computationally cheap first-pass at linkage.

c) This trade-off absolutely also depends on the data and the situation. If we have reason to doubt the quality of our best blocking variables, particularly if these quality issues cannot be resolved by encoding techniques, we should rely less on blocking and use more computationally intense linkage. For example, if we are concerned about typos from data given over the phone, we might just block on gender and state which will suffer less from this problem, although they will yield huge blocks.

2)

a) Different sets of functions had different outcomes. Most combinations tied for worst with an F-measure of 0.65 while the highest F-measure was 0.75. I selected my comparison functions by iterating through all possible combinations of functions and then selecting the functions that had the highest metrics for linkage (first sorting by F-measure, precision, recall, accuracy, then speed). I used the blocking I had already found and I used a similarity threshold classification with a threshold of 0.5. Ideally I would have iterated through all the classification functions and parametres as well as classification and comparison might affect each other’s ideal settings, but this was too computationally intense. In the end, a number of combinations had equal best precision and recall, they tended to use Jaro-Winkler distance functions for middle and last name. My choice was

First name: bag\_dist\_simp\_comp

Middle name: jaro\_winkler\_comp

Last name: jaro\_winkler\_comp

Address: bag\_dist\_simp\_comp

Suburb: bag\_dist\_simp\_comp

State: bag\_dist\_simp\_comp

b) The choice of classification function also had an effect on evaluation metrics. I ended up iterating through all the functions and function parametres but the best (at the expense of being much slower than other functions) was using a decision-tree while the worst was exact-match. The decision-tree actually gave slightly lower recall than similarity thresholds but had higher precision and a high F-measure (thought this result may vary depending on the random\_state seed used).

d) The huge number of negative comparisons means that accuracy is not a useful metric for our purposes as even the worst configuration of the linakge function gave an accuracy of 0.999…. I did not look over the numbers in the confusion matrix or even really use precision or recall as a basis for decision-making if I were particularly concerned about false matches or false non-matches in my data, I may have had a preference as to which of these I would have prioritised and looked less at the F-measure. However, for simplicity I did not do this.

e) It is very sensitive to the choice of an approximate match classification function (and to a lesser extent comparison function) over an approximate match function. This makes sense given the number of minor typos in the data. Generally it benefitted from lower similarity (and minimum similarity) thresholds over higher ones for a similar reason. As mentioned above, the choice of a Jaro-Winkler function for middle name and last name showed particular benefits in comparison.

f) I ran a lot of simulations because I had the time to try and get as close to ideal as possible. I have a lot of evaluation results for unused algorithms.

3)

a) The best linkage I could get used the following parametres:

* Soundex encoding for blocking
* Blocking of last name, street address and suburb
* For linkage:
  + First name: bag\_dist\_simp\_comp
  + Middle name: jaro\_winkler\_comp
  + Last name: jaro\_winkler\_comp
  + Address: bag\_dist\_simp\_comp
  + Suburb: bag\_dist\_simp\_comp
  + State: bag\_dist\_simp\_comp
* Decision-tree for classification

This makes sense to me because Soundex encoding is a relatively computationally cheap was to get rid of some of the mistyping, particularly around names. Last name, street address and suburb are decent ways to quickly reduce down data as these tend to be relatively evenly distributed and while they are not necessarily the most immutable characteristics, they strike a fairly good balance between immutability and small group size compared to other characteristics (e.g. gender which would give huge groups or first name where different variations might be given to different sources). The comparison functions make sense because it seems we are very reliant on middle and last name for our joins, these use the relatively computationally intensive Jaro-Winkler distance to get a robust approximate match while the other characteristics where differences are simpler or less important (e.g. street names without numbers, states where we do not have typos) can use a simpler function, though one that still takes into account some variation between matches. The two trade-offs on a decision-tree would be the need for training data and computing resources. As it is, we have a dataset that is not that large and is well blocked. We also have training data. It makes sense then that supervised learning would perform best under these conditions.

b) The evaluation metrics were as follows:

|  |  |
| --- | --- |
| rr | 0.99998324 |
| pc | 0.6423 |
| pq | 0.95779899 |
| blocking\_time | 0.41485095 |
| accuracy | 0.99999046 |
| precision | 0.96467759 |
| recall | 0.6418 |
| fmeasure | 0.77079205 |
| linkage\_time | 4.20734096 |

These evaluation metrics show the linkage performing well compared to the examples in class. The one exception is recall which is quite low.

4)

a) It seems to me that this dataset is roughly as dirty as the ‘little dirty’ datasets from the labs.

b) Because we have ground-truth data and no duplicates, I can look at how many are exact matches. Then I can also look at the average edit distances between matches to see if how badly matches diverge. I can look at completeness, how many missing variables there are between the two input sets. Then I can also look at whether the dates are formatted as dates, whether the emails are correctly formatted email addresses, whether postcodes are formatted correctly etc.