Reviewer 1

The authors present their new version of the popular Percolator tool. Their main focus is on making Percolator compatible with very large spectrum data sets (millions of spectra), for which I believe there is indeed an increasing demand. The authors claim that using 100k spectra instead of the full data set is sufficient for Percolator to compute accurate PSM scores. In addition the authors investigate different protein inference algorithms that can can work efficiently on very large spectrum data sets.

I have some concerns that need to clarified for the reader and the reviewer.

When introducing the method of Savitski the authors mention that it was shown to work "on large-scale data sets". The data sets used in the manuscript are also large scale except one. However on the latter the method seems to work well. Does this mean Percolator 3 protein inference is only suitable for large very data sets, or do the results on the smaller data set show that it can be used for any data set?

We have added a paragraph concerning this issue in the Discussion section:

"We have not extensively tested our new protein inference functionality on smaller data sets. However, based on the results for the hm_yeast set and the small random subsets of the Kim set, we believe that the estimated FDRs will remain accurate and that it is unlikely that any of the other evaluated protein inference methods will identify many more proteins."

To create the entrapment set 9 random shuffles of the target are added, why 9? Would this choice not influence the conclusions? Does this not influence the entrapment FDR?

On page 6: the authors mention "inadvertently inferred absent sample proteins". I don't understand what this means.

Both reviewers asked for clarifications regarding the entrapment database. We therefore revised the manuscript to address the reviewers' concerns, as follows:

"The goal of this approach is to provide a ground truth regarding the correctness of peptide and protein inferences made by the algorithm under investigation. The sample database contains the proteins of interest, while the entrapment database consists of protein sequences in which the peptide sequences of the sample database are shuffled. We search against the concatenated database of the sample and entrapment database, and subsequently assume that

any match to the sample database is a true positive and any match to the entrapment database is a false positive.

The assumption that any match to the sample database is a true positive is not necessarily true, because the peptide or protein could have been inferred through an incorrect PSM. The purpose of the entrapment database is to attract the majority of these incorrect target PSMs, thereby ensuring that the majority of PSMs, peptides and proteins matching to the sample database are correct. The larger the entrapment database, the higher the probability that an assumed true positive, *i.e.* a match to the sample database, is actually true. For example, using an entrapment database 9 times the size of the sample database, means that we will underestimate the true amount of false positives in the entrapment FDR by ~11% on the PSM level. However, under the assumption that many of the proteins in the sample database are in fact present, this underestimation is far lower on the protein level, because correct proteins in the sample database will conceal incorrect PSMs matching to it."

> *The down-sampling procedure is not clearly explained.*

We have attempted to clarify this procedure by assembling all relevant parts in the Methods section:

"To implement subset training, we used a downsampling strategy in which we randomly sample a subset of the PSMs. Subsequently, we applied Percolator's normal training algorithm to this subset, resulting in three SVM classifiers in the same fashion as mentioned above. For each PSM, including those not selected for training, we then calculated its score as the average of the scores from each of the three SVMs. This strategy involves some overlap between training and test sets, because PSMs selected as part of the training subset will be evaluated by two SVM classifiers which were trained on this particular PSM, and will only avoid problems of overfitting when the strategy is carried out on large data sets. This strategy was adopted for simplicity of implementation. In a future version of the code, we will implement a scheme in which we downsample each cross-validation bin individually.

Preliminary results showed that including target and decoy PSMs belonging to the same spectrum together during the selection of the random subset gave more stable performance than sampling without taking this distinction into account. Therefore, this strategy was applied in the random sampling process."

> *On page 7 the authors mention "We trained using these subsets and applied the trained classifier to the full data set". Is there no more cross-validation to prevent overfitting? I believe this should be better explained as this is an important part in the paper.*

There is cross-validation in the sense that each SVM is trained and tested on separate data. However, during the final step which combines the scores for all classifiers, some data that was

used as training data for two out of three of the SVMs is at risk of being overfit by these two SVMs. As we mention in the discussion, this is something we plan to eliminate in a later release.

Typo: page 5: "and computes trains three separate"

We have changed this to "and trains three separate..."

Reviewer 2

In this article The et al. describe new developments to the Percolator software. This software is a post-processing machine learning method that implements cross-validating support vector machines for the rescoring of peptide-spectrum matches from sequence database searches of proteomic mass spectrometry data. Percolator provides robust statistical values estimating PSM- and peptide-level identification error rates in the form of q-values and posterior error probabilities. It is widely integrated into many proteomics analysis pipelines and has been adapted for use with many of the most widely used proteomics identification applications. In this most recent major release of the software, the algorithms have been further refined to improve performance on the large proteomics datasets that are becoming more common. Although much faster than the initial database search, Percolator can take a large amount of resources and runtime when processing these large datasets; in this latest version of the software the runtime and required computational resources have been significantly reduced. This has been achieved by training the SVMs using a randomly selected subset of the identifications and then taking the average score across multiple cross-validated SVMs. This application would be of great use to many researchers in the field of proteomics analysis in general. Further Percolator development is of importance to the proteomics community at large and the ability to easily and quickly handle large datasets is becoming essential. Protein inference and statistics have been an issue in many proteomics studies and the methods integrated into this Percolator release should help address this.

The paper describes implementation and validation of a straightforward subsampling approach that helps to reduce the runtime and memory consumption of Percolator on large datasets, without making significant sacrifices in accuracy. This is certainly a useful, if incremental improvement to the software. The authors have clearly assessed the minimum sampling size required to achieve accurate and stable PSM scores, and have demonstrated that these subsets have negligible effect on the final scoring and statistical significance of PSMs.

More importantly, the paper compares different methods for protein inference and protein-level FDR estimation that are applicable to large datasets. The performance of these methods is evaluated convincingly based on different datasets. It is stated that the best method was implemented in Percolator, which would make a welcome addition to the software; however, the advertised version 3.0 of Percolator was not available for download at the website mentioned in the abstract, so we could not verify its usability. Independent of that, the presented analysis of different protein inference methods would

likely be of interest to a wider audience within the (computational) proteomics community.

We have now released Percolator 3.0. Source code and binaries are available on https://github.com/percolator/percolator.

Beyond minor questions and comments, we find only one regrettable omission in the paper, which is a comparison of their selected protein inference strategy to a state-of-the-art protein inference engine like Fido. Since Fido is already integrated into Percolator, this would be an obvious comparison to make. Although, as the authors state, Fido may not be suitable for large-scale datasets, it would be important to know what the performance difference is between their simpler (best-scoring peptide) approach and the sophisticated (Bayesian inference) method in Fido. A comparison between the two approaches on a dataset that is as large as possible, while still being usable in Fido, would be very valuable.

We have added a comparison with Fido, which shows that Fido produces inaccurate FDR estimates on our test data and should therefore be used with caution. Furthermore, Fido identified significantly fewer protein groups on one of the small test data sets, though this is by no means conclusive evidence that this result will hold for other small sets as well.

More detailed comments (the ones in bold go beyond proof-reading):

All proof-reading comments have been resolved; answers have been provided for the comments in bold.

Why do the numbers in the left margin not actually line up with the lines in the text?!

The numbers in the left margin are inserted automatically by the JPR submission system.

1. p. 3, l. 6: “X! Tandem” (add space, official spelling)

2. p. 3, l. 20: “proteomics experiments” (add “s”, consistent with earlier use)

3. p. 3, l. 29: “a relatively low number” (not “an”)

4. p. 3, l. 32/33: “use the resulting score vectors” (not “used”)

5. p. 3, l. 34/35: “shorter analysis times” (not “faster”)

6. p. 3, l. 48/49: “pros and cons. Savitski” (missing “.”)

7. p. 3, l. 52: “lower-scoring peptides” (add “-“)

8. p. 3, l. 53/54: “best-scoring PSM for each protein” (not “PSMs”)

9. p. 3, l. 55/56: “Perhaps the simplest way” – isn’t using the single best peptide even simpler?

We changed the text to say: "A simple way ..."

10. p. 3, l. 55/56: “is the widely used two-peptide rule” (word order)

11. p. 3, l. 59: “spurious PSM yields” (not “PSMs”)

12. p. 4, l. 14: “penalizing one-hit wonders on the basis of its many accompanying incorrect peptide inferences” – there is only one PSM for a one-hit wonder, so where do “many incorrect peptide inferences” come from?

In our consideration, the "one-hit" refers to a single high scoring PSM for a protein. Typically, several low scoring PSMs are also present for a protein, which are usually ignored after applying a PSM or peptide-level threshold. The application of Fisher's method does not apply this threshold and can therefore make use of this information as evidence speaking against the presence of the protein.

13. p. 4, l. 14: “on the basis of their” (not “its”)

14. p. 4, l. 29/30: “comprises 2212 runs” (present, still does)

15. p. 4, l. 31/32: “~25 million spectra [15]” (no space after “~”, or use “approx.”; remove “from”)

16. p. 4, l. 31-33: “on LTQ Orbitrap Velos and Elite instruments (Thermo Scientific) equipped with Easy-nLC II nanoflow LC systems (Waters)” (IIRC there were multiple instruments; don’t capitalize “System(s)”)

17. p. 4, l. 38/39: “consists of 561 runs” (present, still does)

18. p. 4, l. 40: “~9 million” (no space after “~”, or use “approx.”)

19. p. 4, l. 43/44: “UPLC system” (don’t capitalize “System”)

20. p. 4, l. 50/51: “the “hm_yeast” set” (add quotation marks – also later!)

21. p. 4, l. 52/53: "MS1 and MS2 files" – what is meant there (separate mzML files for MS1/MS2)?

We clarified this to read: "two separate files in the MS1 and MS2 formats respectively" (http://www.ncbi.nlm.nih.gov/pubmed/15317041).

22. p. 4, l. 52/53: "ProteoWizard" (capitalize "W", official spelling)

23. p. 4, l. 54: "at their default values" (not "to")

24. p. 5, l. 21: "For the decoy proteins, …" – rewrite this sentence (beginning sounds clumsy; don't mix active and passive voice in one sentence)

25. p. 5, l. 59 – p. 6, l. 36: I can't assess the validity of this part.

Both reviewers asked for clarifications regarding the entrapment database, we therefore rewrote the paragraph addressing this point:

"The goal of this approach is to provide a ground truth regarding the correctness of peptide and protein inferences made by the algorithm under investigation. The sample database contains the proteins of interest, while the entrapment database consists of protein sequences in which the peptide sequences of the sample database are shuffled. We search against the concatenated database of the sample and entrapment database, and subsequently assume that any match to the sample database is a true positive and any match to the entrapment database is a false positive.

The assumption that any match to the sample database is a true positive is not necessarily true, as the peptide or protein could have been inferred through an incorrect PSM. The purpose of the entrapment database is to attract the majority of these incorrect target PSMs, thereby ensuring that the majority of PSMs, peptides and proteins matching to the sample database are correct. The larger the entrapment database, the higher the probability that a true positive is actually true. For example, using an entrapment database 9 times the size of the sample database, means that, on PSM level, we will underestimate the amount of false positives in the entrapment FDR by at most 11%. However, assuming that many of the proteins in the sample database are in fact present, these false positives will not be present at protein-level, due to PSM to peptide reduction and peptide to protein reduction."

26. p. 7, l. 49/50: 30 GB is still a lot of memory for a "commodity" computer. Why did the memory consumption only drop by a third, when less than 1% of the data was used?

The majority of the memory consumption comes from the PSM and protein identifiers, which are stored in memory. The only way to circumvent this is to ignore this information if it's not needed

or to temporarily store this information on disk. We agree that 30 GB is quite a lot for a commodity computer, but at the same time believe that most labs doing large-scale analyses will have a machine with this amount of RAM.

27. p. 8, l. 27/28: "~3% and ~1%" (no spaces after "~")

28. p. 9, l. 60/61 – p. 11, l. 27/28: Does the processing time include the in-silico digestion and grouping of the database proteins? If not, how long does that take?

We have added several clarifications on this topic in the revised manuscript:

"By using a subset of 500 000 PSMs to train the SVM, Percolator's runtime for producing peptide-level results was reduced from almost a full day to less than 10 minutes."

"The in-silico protein digestion and subsequent grouping required only 15 seconds for the Swiss-Prot database and 4 minutes for the Swiss-Prot + TrEMBL database."

"For our largest evaluated data set, the 73 million PSMs of the Kim data set, the combined runtime for Percolator 3.0, *i.e.* from PSMs to protein-level FDRs, was 10 and 15 minutes for the Swiss-Prot and Swiss-Prot + TrEMBL database respectively."

29. p. 10, l. 47: "fewer protein groups" (not "less")

30. p. 11, l. 36: "3-5% margin" (formatting)

31. p. 11, l. 36: "second-best method" (add "-")

32. p. 11, l. 43: "resource-saving" (add "-")

33. p. 12, l. 2/3-4/5: "human proteome-scale study" (add "-")

34. p. 12, l. 6: "The downsampling approach for …" (add context to remind the reader)

35. p. 12, l. 13/14: "of positive training examples" – what about negative examples?

Using separate target-decoy searches will typically give many more negative than positive training examples, because all decoy PSMs are negative examples, but only a small subset of the target PSMs will be positive examples. For a concatenated search the number of negative examples might indeed be a relevant number to keep track of.

36. p. 12, l. 25/26: "lower-scoring peptide inferences" (add "-")

37. p. 12, l. 50-54: "Consequently, ambiguity about which specific proteins are present is much rarer." – At the same time, there are also many more cases of shared peptides, which have to be removed because they don't map uniquely to a protein group. This means that the number of protein groups that can be inferred may be lower. (Or not?)

Given a fixed list of PSMs and a protein-level threshold, the number of inferred protein groups will indeed typically be lower by ignoring shared peptides. However, in our approach, the average number of proteins in a protein group will be much lower (typically only 1 or 2) and accurate FDR estimates can be achieved by simple decoy models.

We added the latter specification to the manuscript:

"For the Swiss-Prot database, virtually all protein groups contained only 1 protein and for the Swiss-Prot + TrEMBL database they typically contained just 1 or 2 proteins."

38. Page 5, Line 10 Typo: "We used Tide's default fragment tolerance, and the other search parameters were kep the same as in [27]" – missing t in "kept"

39. Page 5, Line 18 Typo: "By default, Percolator's semi-supervised learning algorithm randomly splits the training set into three subsets and computes trains three separate SVM classifiers" – extra word "computes"

40. Figure 1 Legend typo: "The number significant PSMs and unique peptides does not drop significantly, even for subsets of 100 000 PSMs." – missing word "of"

41. Page 6, Line 10: "miscleavages and peptide lengths of [7, 50], specifically chosen" – "[7, 50] range has same formatting as references, consider representing range differently in text. The range on page 9 might need to be reformatted too, although as it's fractions does not look like a reference.