Hands-on DH methods tutorial - j.mp/dhh15ho

1. Exploration of Structured Data

1.1 Data cleanup in OpenRefine

1. Install OpenRefine
2. Run OpenRefine and go to http://localhost:3333/
3. Copy and paste the content of https://raw.githubusercontent.com/dhh15/fnewspapers/f3d14a42596d41896e57e68ac1e186ec638b65cb/data/timeseries2_others.csv as a new project into OpenRefine (create project from clipboard). Remove the first line and add a heading line “year,word,frequency”)
4. Transform frequencies into numbers using Edit cells -> Common transforms -> To number
5. Create a facet for “frequency” using Facet -> Numeric facet
6. Search for overpowering frequencies
7. Create a facet for “word” using Facet -> Text facet
8. Limit to words with overpowering frequencies (“työ”)
9. Under “All”, remove all matching rows
10. Under “word”, Edit cells -> Cluster and edit. Try different options for clustering, merge cells that should be the same
11. Export the data as CSV

1.2 Data visualization in RAW

1. Copy and paste the exported CSV data (or https://raw.githubusercontent.com/jiemakel/dhintro/dad05a321dee7ab37ec4269555de428d6baa358c/socialist-frequent-words-filtered-timeseries.csv) into RAW.
2. Select a suitable visualization (Bump Chart or Streamgraph)
3. Drag relevant fields into their positions, tune visualization (you can export the visualization as an SVG file)

1.3 Data extension (using geocoding as an example) in OpenRefine

1. Copy and paste https://raw.githubusercontent.com/humanitiesplusdesign/palladio-app/master/sample%20data/Letters.txt as a new project in OpenRefine
2. Sort by one of the place fields
3. Select “Reorder rows permanently” from the sort menu that just appeared
4. Select “Blank down” from the edit cells menu in the place column so each place is mentioned only once
5. Select “Place -> Facet -> Customized facets -> Facet by blank”. Filter by blank, and select “All -> Edit rows -> Remove all matching rows” to remove extraneous rows
6. Follow the geocoding recipe at https://github.com/OpenRefine/OpenRefine/wiki/Geocoding to add coordinates based on the place column
7. Export the place and coordinate fields as CSV using “Export -> Custom tabular exporter”

1.4 Data visualization in Palladio

1. Copy and paste https://raw.githubusercontent.com/humanitiesplusdesign/palladio-app/master/sample%20data/Letters.txt into Palladio
2. Click on a place field, choose “Add new table” and copy and paste your geocoded place table (or https://raw.githubusercontent.com/humanitiesplusdesign/palladio-app/master/sample%20data/Cities.tx)t as an extension to use the map view

3. Click on “author/recipient”, choose extend and copy and paste https://raw.githubusercontent.com/humanitiesplusdesign/palladio-app/master/sample%20data/People.txt as an extension table to enable facet filtering by person metadata. Do the same for the remaining attribute
4. Fiddle around with the various views (you can often export a visualization as an SVG file, or you can save the whole project to allow someone else to load your state)

2. Topic Modeling of raw texts using LDA

LDA is a topic modeling methodology, where we assume that in a set of documents, there are N underlying topics that are being written about, which LDA tries to find out. What comes out are statistical spreads, which give a distribution of 1) how much a topic is present in each of the documents and 2) how probable are each of the words to occur in each topic.

1. Install RStudio
2. Install any missing packages if you are on a fresh install:
    a. install.packages(c("topicmodels","tm","LDAvis"))
3. Paste in this script and point it to your corpus:

```
# Required packages
library(topicmodels)
library(tm)
library(LDAvis)

# Load corpus and preprocess
corpus <- VCorpus(DirSource("/srv/data/varieng/ceec-subcorpora/scot-1700-1719/"))
corpus <- tm_map(corpus,content_transformer(tolower))
corpus <- tm_map(corpus,removeNumbers)
corpus <- tm_map(corpus,removePunctuation)
corpus <- tm_map(corpus,removeWords,stopwords("SMART"))
corpus <- tm_map(corpus,stripWhitespace)
doc_term <- DocumentTermMatrix(corpus)

# Do LDA
set.seed(43523) # for reproducibility
numtopics <- 20
lda <- LDA(doc_term, numtopics)

# Write out LDA results
write.csv(posterior(lda)$topics,"lda-documents-topics.csv")
write.csv(topics(lda),"document-main-topic.csv")
write.csv(posterior(lda)$terms,"lda-terms-topics.csv")
write.csv(terms(lda,50),"topic-50-main-terms.csv")

json <- createJSON(phi = exp(lda@beta), theta = lda@gamma,
                   doc.length = rowSums(as.matrix(doc_term)), vocab = lda@terms,
                   term.frequency = colSums(as.matrix(doc_term)))
serVis(json)
```

4. Explore in the user interface. Of particular interest is the relevance metric adjustment slider (λ), which allows you to weigh how much topic specificity (how much a term appears also in other topics) is weighted against pure topic "relatedness". Identification of what the different topics actually mean will probably necessitate finding a suitable sweet spot using this tool.
5. Future versions will most probably support more functionality out of the box, so you don't have to e.g. look at the topic-document -matrix from separate csv files: http://glimmer.rstudio.com/cpsievert/xkcd/ (I think you can actually now take the json produced by the above script as well as the document-main-topic.csv and upload them to this version for visualization)