Group 4 workflow

1. We set up a GitHub **repository** for the project: <https://github.com/djbpitt/tufts_2018> ; with the following **folder structure**:

* (at the root):
  + index.html (the main page of our website, see below)
  + site.css: main css for the website (but not the only one)
  + README.md: markdown file that describes the repo
* (and these subfolders:)
  + composite: holds the html files for the pages of our digital edition of each plant (generated by XSLT, see below)
  + images: holds the images of the four plants from the French edition (edited by hand to create a transparent background with Gimp; format needs to be PNG, because jpeg does not allow transparent backgrounds)
  + meta: holds the html files for the metadata pages for each historical edition (generated by XSLT, see below)
  + old: files we don’t need anymore (similar folders can be found at other levels)
  + reading\_views: holds folders for each language, each of which holds for every plant
    - tt.xml files (converted from the original EXMARaLDA treetagger export) that have a numbered list of the sentences in the plant description (see below)
    - html files that were converted from these tt.xml files (see below)
  + sources: holds folders for each language, each of which contains data provided by the other teams (and Thomas), which had to be sometimes manually corrected:
    - .exb export files for every plant from EXMARaLDA
    - .tt (treetagger) xml files for every plant, converted from .exb via pepper by Thomas
    - metadata xml files for the book level
    - a pdf of the relevant pages of the historical edition
  + stemma: holds the GraphViz .dot file that produces the stemma, and the stemma svg itself (lightly edited by hand)
  + util: holds the xslt and python files, with some support files, for development (see below)
  + wireframes: contains a number of tests and sketches for the design of the website

1. We **set up a GitHubPages website** for the project (go to the properties of the project and set it to allow the setup of a GitHubPages site): <https://djbpitt.github.io/tufts_2018>

* When you create GitHubPages, a .yaml file is created; in GitHubPages you can use Jekyll templates (but we didn’t). For more on Jekyll, see: <https://jekyllrb.com> , <https://programminghistorian.org/en/lessons/building-static-sites-with-jekyll-github-pages>

1. **Homepage workflow**:

* We sketched out on paper what the website should look like (see pictures in the wireframes directory); thinking about how users would like to interact with the site, we decided to make separate pages for every plant, linked from the main page
* We put the index.html file, which we built by hand, in the Github repository’s root directory
* Three main sections on the main page:
  + Links to the plant pages: we took images from the French edition to hold the links to the plant pages
  + Stemma: We made a quick (but thoughtful) manual comparison of the texts of one plant to build a provisional stemma
  + Bibliography table with links to the pdf editions + metadata of every edition

1. **Metadata of the editions workflow**:

* converted the metadata files the groups produced to (well-formed) xml files, partly with a python script (in /tufts\_2018/util), partly by hand
* Annis link in the metadata: in Annis, we made a search for “sentence” in the relevant corpus, copied the link to the search results, and added it to the metadata file (xml base file)
* With an XSLT file (/tufts\_2018/util /meta-to-html.xsl), we transformed these xml metadata files into html pages linked from the index.html page, and residing in the meta subdirectory

1. **Digital edition workflow:**

* Thomas used pepper to convert the EXMARaLDA .exb files to treetagger (.tt) xml files
* We started from .tt files (in /tufts\_2018/sources (renaming them to be consistent, and fixing a number of xml errors by hand))
* Using the tt\_to\_html.xsl file, we transformed these to .tt.xml files that have numbered list of sentences; these were put into the language subdirectories of the reading\_views directory.   
  $ saxon –s:botrys.tt –o:botrys.tt.xml –xsl:../../util/tt\_to\_html.xsl

(we later automated this by writing a shell script, reading\_views.sh, in the util directory; it has to be run from the main directory)

* We hand-built an html table for every plant, that provides a comparison of the sentence numbers in every language (/tufts\_2018/util/mapping.html). Canonic line numbers were added to these tables to make comparison possible.
* Using the util/add\_sentence\_numbers.xsl (running it from the language subdirectories of the reading\_views directory), we added the canonic line numbers in the @class attribute of the ordered lists:

$ saxon –s:botrys.tt.xml –o: botrys.html –xsl:../../util/add\_sentence\_numbers.xsl

* The final plant html files were built with combine\_[plantname].xsl scripts (in the ideal world, we would have made just one xslt script that built all these pages), and put into the composite directory.   
  Run “combine\_artemisia\_herba-alba.xsl” (from the util directory): (BUT: adapt the file names + the number of files we are looping over + title+header)

$ saxon –it –xsl:combine\_artemisia\_herba-alba.xsl –o:../composite/ambrosia.html  
NB: -it = initial template: to use if there is not a direct input file for the xslt

* In order to have corresponding sentences light up when hovering over a sentence, we produced a javascript file (in /composite/scripts.js)
* We also added the metadata on each plant in each language, in a popup that appears when clicking the small circled ⓘ after each language title; and a link to a pdf of a scan of the relevant historical edition (which is located in the sources/[language] folder).

NB: saxon is an xslt engine (the same one used by oXygen)

Running saxon from Windows:

Download Saxon HE9 and save it in a directory (make sure that the path does not include spaces)

Instead of simply the command saxon, use:  
 java –jar C:/Programme/SaxonHE9-8-0-12J/saxon9he.jar

You can create an alias for saxon by typing:   
 doskey saxon= java –jar C:/Programme/SaxonHE9-8-0-12J/saxon9he.jar

Mac users can install saxon using homebrew:

$ brew install saxon