Intermediate LaTeX and BibTeX

MATTHEW DENNY SUNDAY 21ST SEPTEMBER, 2014

Please note that this tutorial assumes that you are using TeXMaker as your LaTeX editor. It is available for download here: http://www.xm1math.net/texmaker/.

1 Hyper-Linking

Before we get started, there are several packages you will want to call in your document header in order to be able to use hyper-linking in your document.

```
% Color package and define colors
\usepackage{color} %the base color package
\usepackage[usenames,dvipsnames]{xcolor} %more pre-named colors
\definecolor{darkblue}{rgb}{0,0,1} %you can define any color this way
% Hyper Reference Formatting
\usepackage{hyperref} %[pagebackref] option links back from the bibliography
\hypersetup{colorlinks,breaklinks,linkcolor=darkblue,urlcolor=darkblue,
           anchorcolor=darkblue,citecolor=black}
```

Hyperlinking in LaTeX is one of the coolest features because it holds the possibility of improving the readability and utility of electronic versions of the PDF's you create for the reader and whats more, LaTeX will do this automatically for you! There are three main uses for hyperlinking in your documents the first, and most obvious is linking to web resources like the LaTeX Wikibook page on hyper-references.

```
web resources like the \href{http://en.wikibooks.org/wiki/LaTeX/Hyperlinks}
{\LaTeX\ Wikibook page on hyper-references.}
```

The second use for hyperlinks is to reference pictures, tables and sections in you document so that if you are talking about them in some other part of the document your reader can simply follow a link to the resource to see what you are talking about To find out more check out the LaTeX Wikibook page on cross-referencing. This can be done using the label and ref commands as follows:

```
This is a cross-reference to section \ref{sec:advanced graphics} which
covers advanced graphics.

%.... skip down to the beginning of the advanced graphics section

\section{Advanced Graphics}
\label{sec:advanced graphics}
```

This is a cross-reference to section 4 which covers advanced graphics. Commands are also available to reference figures and charts and you can find out more by checking out the website above. The final and most interesting (to me, anyway) set of references are to citations. One of the cool features of the hyperref package is that it will automatically link citations to their bibliography entry when you use the cite command.

```
This is a citation to \citet{newman2003structure} which is a neat article about
networks.
```

This is a citation to Newman (2003) which is a neat article about networks. We will cover this issue in greater depth in section 3. Note that it usually takes two passes through the LaTeX rendering engine to correctly display any sort of cross or citation reference so if you see a ?? instead of a number, just click the Quick-Build button again (assuming you are using TeXMaker). There is a lot you can do with references so I encourage you to dig in and give it a try. The internet is burting with examples and forum posts abotu how to do this so if you run into any trouble jsut type something like "latex reference to figure" into Google and you should get a lot of helpful forum posts as results.

2 Getting Output from R and Stata to LaTeX

Getting output out of Stata into LaTeX is pretty easy and is also covered in the Intro to LaTeX tutorial. However we will review it here. You want to begin by installing the package to output LaTeX code in Stata. Open up Stata and type in the following command:

```
findit outtex /* Brings you to a page where you can download the package */
```

This will take you to a link where you can download the package. Once it is installed (which should only take a few seconds and be done automatically by Stata), you can run your analysis. If you want to output the results of a regression, for example, you should execute the command to run the regression and then enter the following command into the Stata console before doing anything else:

```
outtex, detail level below title(Dep = 'Your dependent variable name')
/* creates the output */
```

This will print a bunch of LaTeX code to the console which you can then simply copy and paste into the appropriate place in your document.

Getting output from R into LaTeX is similarly as easy simply you need to download the stargazer package, load it and then use it in the same way you would use the outtex package in Stata. Begin by downloading the package (make sure your version of R is up to date and your mirror is set, otherwise use the package manager drop down menu in base R):

```
install.packages("stargazer")
```

You will then need to load the package during the R session in which you would like to use it as you would any other package:

```
library("stargazer")
```

Run your analysis and save your regressions to objects. The cool thing about stargazer is you can put a bunch of regression objects in the same table automatically jsut by giving them as ordered arguments to the stargazer function. This will output a bunch of LaTeX code to the console which you can just copy and paste into your document.

```
model1 <- lm(deviations ~ AvgPreviousContribs + standard_deviation +
 high_outliers + low_outliers, data= treatment1)

model2 <- lm(deviation_difference ~ AvgPreviousContribs + empirical_diff
 + empirical_alloc_diff + standard_deviation + high_outliers + low_outliers,
  data= treatment2)

#output TeX code
stargazer(model1,model2)
```

3 BibTeX

Before we dive in to the meat of this section, you will want to have the following lines as part of your document header. They will give you access to a whole bunch of cool citation styles and automatic formatting.

```
\usepackage[round]{natbib} % gives you access to a lot of different citation styles
\usepackage{cite}
% the following command will make your citations default to author (year)
%\let\cite\citet
```

BibTeX is a format for saving references in a plain text file that lets you reuse them. You create a file called **library.bib** (by convention) and then copy in citations formatted in the BibTeX format and the LaTeX engine will be able to automatically recognize them. Some example references that I took from my own **library.bib** file are shown below:

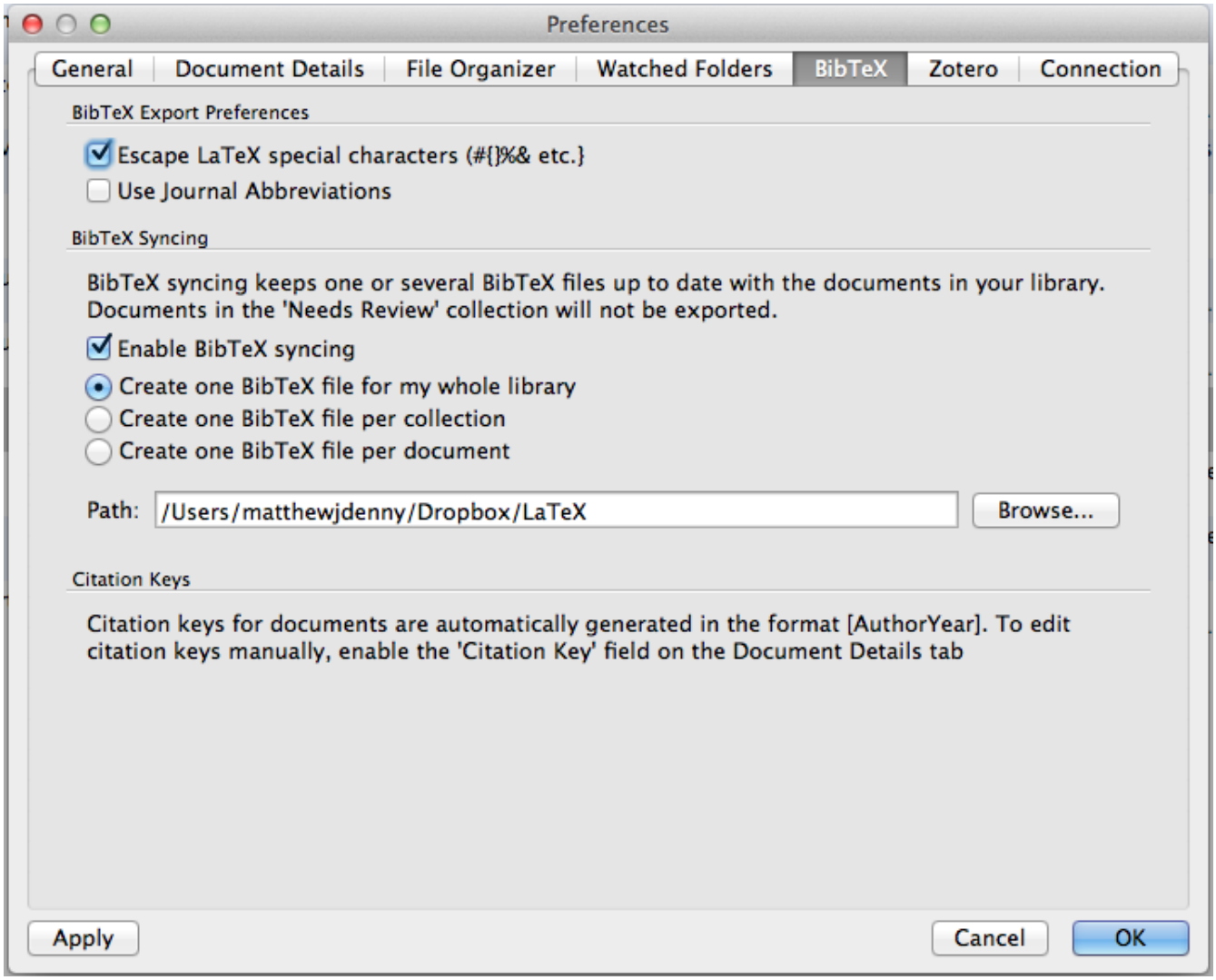
```
@unpublished{Lubin2013,
archivePrefix = {arXiv},
arxivId = {1308.3174},
author = {Lubin, Ben and Shore, Jesse and Ishakian, Vatche},
eprint = {1308.3174},
institution = {University of Pennsylvania Wharton School},
pages = {1--21},
title = {{Communication Network Design: Balancing Modularity and Mixing via
Extremal Graph Spectra}},
url = {http://mackinstitute.wharton.upenn.edu/wp-content/uploads/2013/07/Shore-Jesse
\_Communication-network-design.pdf},
year = {2013}
}
@article{Mutz2004,
author = {Mutz, Diana C.},
doi = {10.1017/S0003055402004264},
journal = {American Political Science Review},
month = mar,
number = {01},
title = {{Cross-cutting Social Networks: Testing Democratic Theory in Practice}},
url = {http://www.journals.cambridge.org/abstract\_S0003055402004264},
volume = {96},
year = {2004}
}
@book{Davis1993,
address = {Princeton, NJ},
author = {Davis, Douglas D and Holt, Charles A},
publisher = {Princeton University Press},
title = {{Experimental Economics}},
year = {1993}
}
```

You can store a bunch of references in no particular order and the LaTeX engine will automatically look things up when you make a citation and automatically generate a bibliography for you.

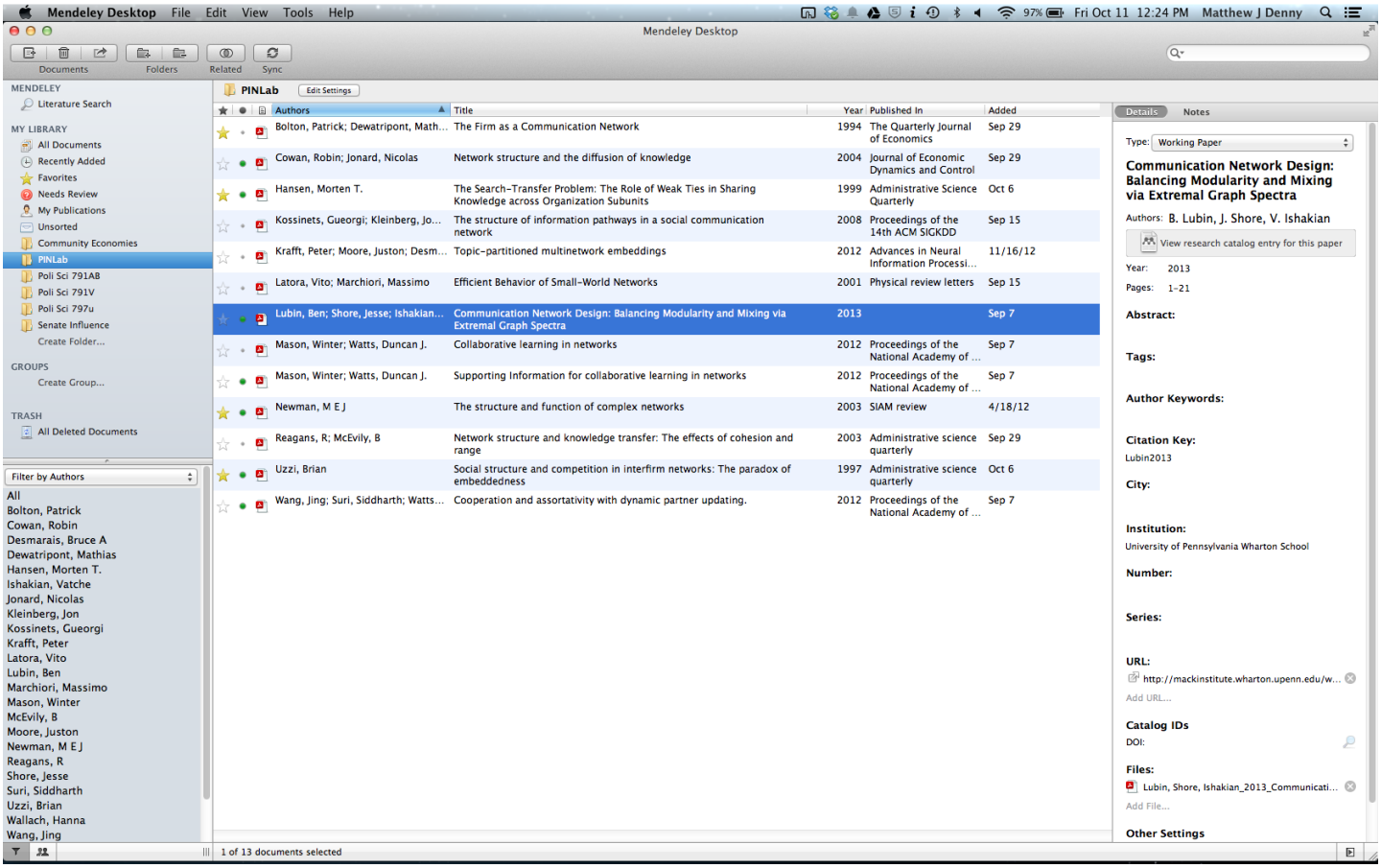
For the purposes of this tutorial, I will be using the Mendeley reference management software. It is

free, open source, works across all platforms and is available here: http://www.mendeley.com/. Pretty much any reference managment software will work including zotero and refworks and the LaTeX distribution for OS X even comes with its own bibliography management software (BibDesk). THe only thing you need to make sure of is that your software of choice supports "BibTeX integration" which is a fancy way of saying that when you add a reference to your library it will automatically write the information to a designated bibTeX library file so you don't have to. In practice, if you use Mendeley you will pretty much never have to touch your **library.bib** file. The point of using a reference manager is to make things easy!

If you are using mendeley you will want to go the the options/preferences menu depending on what system you are using. and click on the **BibTeX** tab. You will want to select the check boxes as in the picture below and specify the Path as the folder where you keep all of your LaTeX documents:



Now when you add files to your library in Mendeley they will automatically be added to your **library.bib** file that Mendeley automatically created for you. To add a citation to the document you are working on, just click on the entry in Mendeley to select it and then enter **Command + K** on a Mac or **Control + K** on a windows machine to automatically copy the citation key. This is illustrated in the screen shot below:



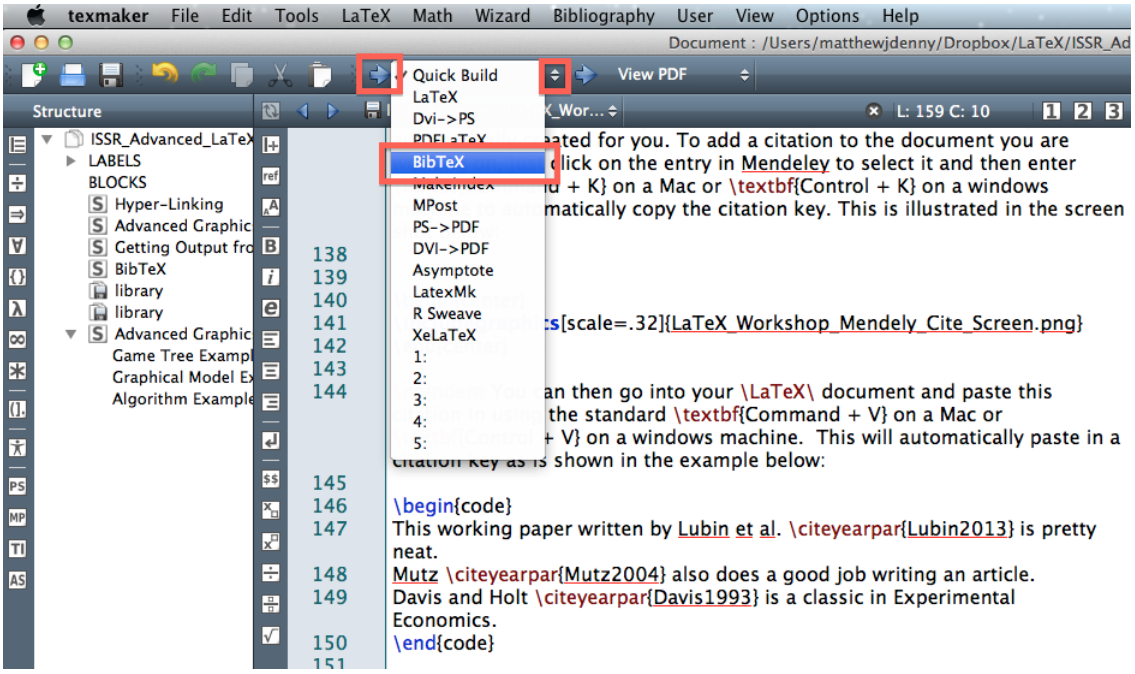
You can then go into your LaTeX document and paste this citation in using the standard **Command + V** on a Mac or **Control + V** on a windows machine. This will automatically paste in a citation key as is shown in the example below:

```
This working paper written by \citet{Lubin2013} is pretty neat.
 \citet{Mutz2004} also does a good job writing an article.
\citet{Davis1993} is a classic in Experimental Economics.
```

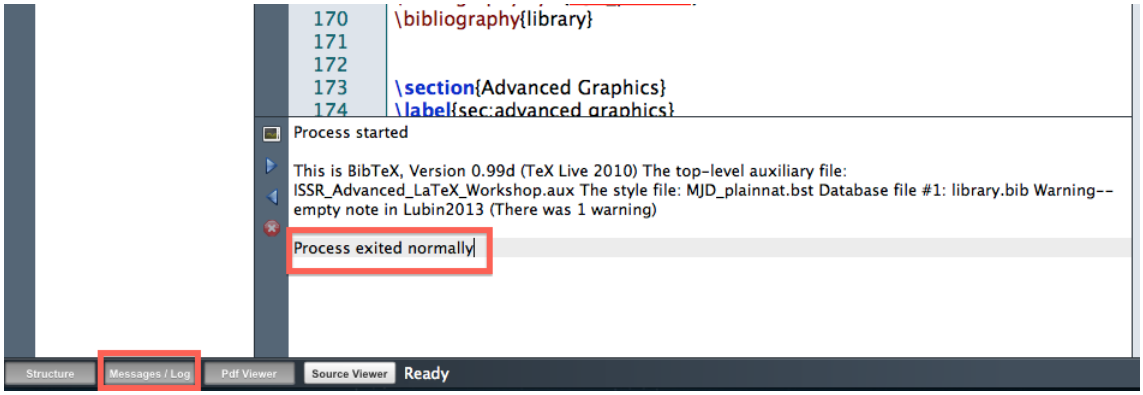
This working paper written by Lubin et al. (2013) is pretty neat. Mutz (2004) also does a good job writing an article. Davis and Holt (1993) is a classic in Experimental Economics. To display the bibliography you will need to put the following two lines of code in your document where you want the References section to display:

```
\bibliographystyle{plainnat}
\bibliography{library}
```

Once you have all of the above in place, render your document using the **Quick Build** button. You will notice that nothing has changed. This is an important preliminary as you will now need to select BibTeX from the drop-down menu . Once you have don this you should click on the arrow next to the drop down menu where you selected **BibTeX** just as you would with **Quick Build**.



You will want to do this **twice** because **BibTeX** will need to create and order the bibliography to display and this requires two passes through the rendering process. You ought to see a message similar to the one displayed below in your **message/log** console if everything went well.



Now if you switch back to the **Quick Build** button and click on that twice you should see a references section appear:

References

Davis, Douglas D and Charles A Holt. *Experimental Economics*. Princeton University Press, Princeton, NJ, 1993. 5

Lubin, Ben, Jesse Shore, and Vatche Ishakian. Communication Network Design: Balancing Modularity and Mixing via Extremal Graph Spectra. 2013. http://mackinstitute.wharton.upenn.edu/wp-content/uploads/2013/07/Shore-Jesse_Communication-network-design.pdf. 5

Mutz, Diana C. Cross-cutting Social Networks: Testing Democratic Theory in Practice. *American Political Science Review*, 96(01), March 2004. http://www.journals.cambridge.org/abstract_S0003055402004264. 5

Newman, M E J. The structure and function of complex networks. *SIAM review*, pages 167–256, 2003. 2

This references section can be formatted in a bunch of different ways just by changing the argument of the bibliographystyle command. There are a bunch of different styles which can be found here: http://www.cs.stir.ac.uk/ kjt/software/latex/showbst.html. Just try substituting in a different parameter as follows:

```
\bibliographystyle{nature} %could be anything from the list above.
%or it could be:
\bibliographystyle{Denny} %my customized bibliography style
```

You will have to follow the steps above of running the **BibTeX** command twice and then the **QuickBuild** button twice every time you want to update your bibliography but it is okay to add several cites before doing this. If you render your document you will just see (?) where the new cite keys are. You will definitely want to check out the Wikibooks page on bibliography management for more information on formatting your bibliography and citations. Here is a link to the site :
http://en.wikibooks.org/wiki/LaTeX/Bibliography_Management

4 Advanced Graphics

You will need the following package to make sure that your figures postion themselves correctly:

```
\usepackage{float}
\usepackage{graphicx} %always need to include for figures
```

You may also want to stick your figures in a separate sub-directory if you are going to be working with lots of them. To do this, you will need to add the following to your document preamble:

```
\graphicspath{{Figures/}{../Figures/}}
```

This section will cover some general tips on including graphics in your documents and then give some example code and output for making game trees, graphical models and algorithms. Before we get into specific examples it is a good idea to go over the use of the figure environment, as it is a good way to add captions and links to your included figures. Lets begin with an example:

```
\begin{center} %center the figure as a whole

\begin{figure}[H] %[H] means put the figure right HERE, no exceptions and no moving
%Things around even if this mean leaving big white spaces in the text. Alternatively use
%[h] which is a bit less harsh and will sometimes move the picture around if it means
%leaving much less white space.

\centering % centers the picture in the figure (kind of silly)
\includegraphics[scale=0.6]{791P_Term_Paper_Optimal_Influence_plots.pdf}
\caption{\label{fig:measures} % the caption command prints a caption with the figure
% the \label{fig:measures} labels the figure so it can be referenced to using the
%\ref{fig:measures} command
This figure shows four different measures of influence,
plotted by DW-NOMINATE score. }
\end{figure}
\end{center}
```

I used this same code to create figure 1. To see how I linked to the figure lets look at the code that created this sentence.

```
\noindent I used this same code to create figure \ref{fig:measures}. To see how I
linked to the figure lets look at the code that created this sentence.
```

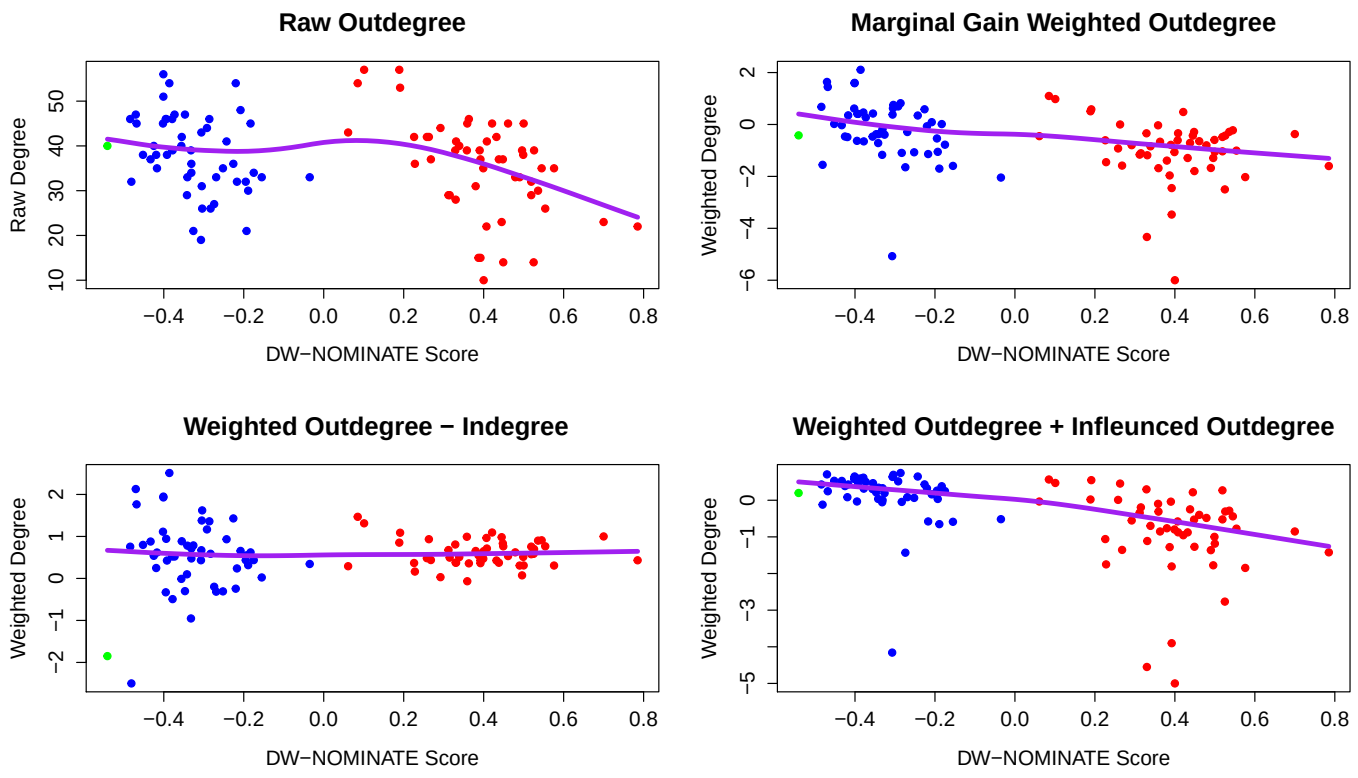


Figure 1: This figure shows four different measures of influence, plotted by DW-NOMINATE score.

The main point here is that we can use the **figure** environment to wrap any graphics you want to include in your document as a way to hyperlink to them and provide automatically formatted captions. Check out this page for more information: http://en.wikibooks.org/wiki/LaTeX/Floats,_Figures_and_Captions

4.1 Game Tree Examples

To create game trees in LaTeX, you will need to include the following packages in your document header:

```
\usepackage{tikz}
\usetikzlibrary{trees}
```

We will begin with a simple commented example and then I will include a kitchen sink example for those of you needing more complicated game trees. This environment is probably one of hte most difficult and frustrating in LaTeX. There is a lot of parentheses matching and it is relatively unclear where nodes are supposed to go. The only real way to learn how to make game trees is by trial and error, building off of the code provided here.

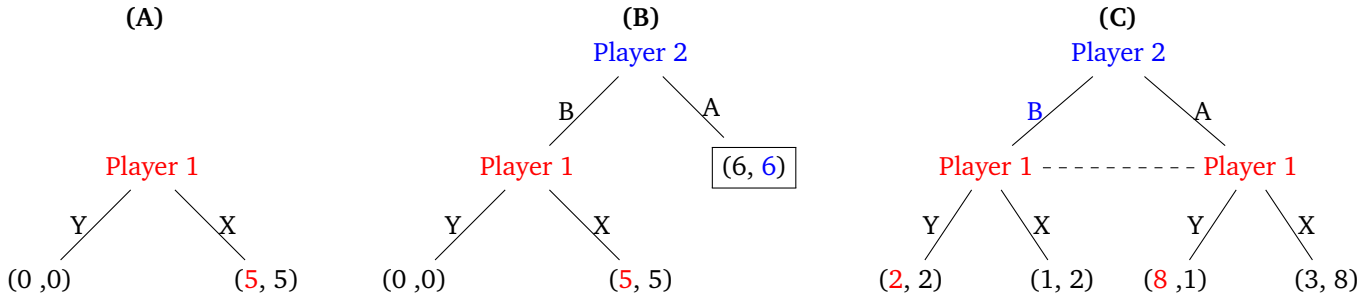
This is a simple example of three different game trees all in the same figure:

```
\begin{figure}[h]
\begin{center}
\textbf{(A)} \hspace{2.3in} \textbf{(B)} \hspace{2.2in} \textbf{(C)} \hspace{.5in}
\hphantom{} \\
% First, set the overall layout of the tree
% You might need to play with these sizes to ensure nothing overlaps.
% the level distance controlls the vertical distance between rows of nodes on
the game tree. The sibling distance controls the horizontal distance between
nodes on the same level. Level 1 is the top level and the descend from there.
\tikzstyle{level 1}=[level distance=1.5cm, sibling distance=3cm]
\tikzstyle{level 2}=[level distance=1.5cm, sibling distance=3cm]
\tikzstyle{level 3}=[level distance=1.5cm, sibling distance=1.5cm]
\tikzstyle{level 4}=[level distance=1.5cm, sibling distance=2cm]
\begin{tikzpicture}
%Start with the parent node, and slowly build out the tree
% with each "child" representing a new level of the diagram
% each "node" represents a labelled (or unlabeled if you
% want) node in the diagram.
\node{\color{red}Player 1}
% edge from parent
% node[left]{Left}
              child{
                              node{(0 ,0)}
                      edge from parent
                              node[left]{Y} % the {Y} is te node name used internally to
                              %draw things like information sets.
                              }
                              child{
                              node{({\color{red}5}, 5)} %put whatever you want in the
                              % node here
                              edge from parent
                              node[right]{X}
                              };
\end{tikzpicture}
\hspace{.1in} % space between trees
% First, set the overall layout of the tree
% You might need to play with these sizes to ensure nothing overlaps.
\tikzstyle{level 1}=[level distance=1.5cm, sibling distance=3cm]
\tikzstyle{level 2}=[level distance=1.5cm, sibling distance=3cm]
\tikzstyle{level 3}=[level distance=1.5cm, sibling distance=1.5cm]
\tikzstyle{level 4}=[level distance=1.5cm, sibling distance=2cm]
\begin{tikzpicture}
%Start with the parent node, and slowly build out the tree
% with each "child" representing a new level of the diagram
% each "node" represents a labelled (or unlabeled if you
% want) node in the diagram.
\node(b){\color{blue}Player 2}
           child{
              node{\color{red}Player 1}
% edge from parent
```

```
% node[left]{Left}
              child{
                            node{(0 ,0)}
                     edge from parent
                            node[left]{Y}
                            }
                            child{
                            node{({\color{red}5}, 5)}
                            edge from parent
                            node[right]{X}
                            }
                            edge from parent
                            node[left]{B}
           }
           child{
               node{\fbox{(6, {\color{blue}6})}}
               edge from parent
               node[right]{A}
           } ;
\end{tikzpicture}
\hspace{.1in}
\tikzstyle{level 1}=[level distance=1.5cm, sibling distance=3.5cm]
\tikzstyle{level 2}=[level distance=1.5cm, sibling distance=2cm]
\tikzstyle{level 3}=[level distance=1.5cm, sibling distance=1.5cm]
\tikzstyle{level 4}=[level distance=1.5cm, sibling distance=2cm]
\begin{tikzpicture}
%Start with the parent node, and slowly build out the tree
% with each "child" representing a new level of the diagram
% each "node" represents a labelled (or unlabeled if you
% want) node in the diagram.
\node(c){\color{blue}Player 2}
           child{
               node(f){\color{red}Player 1}
               child{
                            node{({\color{red}2}, 2)}
                     edge from parent
                            node[left]{Y}
                            }
                            child{
                            node{(1, 2)}
                            edge from parent
                            node[right]{X}
                            }
                             edge from parent
             node[left]{\color{blue}B}
           }
           child{
               node(e){\color{red}Player 1}
% edge from parent
% node[left]{Left}
```

```
                child{
                                node{({\color{red}8} ,1)}
                        edge from parent
                                node[left]{Y}
                                }
                                child{
                                node{(3, 8)}
                                edge from parent
                                node[right]{X}
                                }
                                edge from parent
                    node[right]{A}
            };
%Now I create the information set. Note that I utilize the names
% that I had previously assigned to nodes in my graph
\draw [dashed](f)--(e);
\end{tikzpicture}
\end{center}
\end{figure}
```

We can see what this looks like by checking out the game trees below:



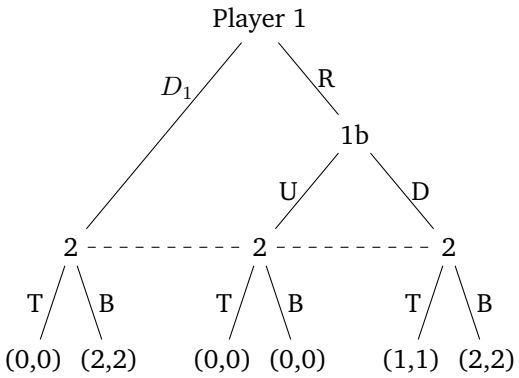
This is the code for game trees that I used to learn how to make them. I tinkered with the code below for several hours until I was able to get things to come out right. The website also provides some helpful information: http://www.drewdimmery.com/extensive-form-game-trees-in-tikz/. Check out the game tree this code creates in figure 2.

```
%example code for making baller game trees
%credit: http://www.drewdimmery.com/extensive-form-game-trees-in-tikz/
\begin{figure}
\begin{center}
% First, set the overall layout of the tree
% You might need to play with these sizes to ensure nothing overlaps.
\tikzstyle{level 1}=[level distance=1.5cm, sibling distance=2.5cm]
\tikzstyle{level 2}=[level distance=1.5cm, sibling distance=2.5cm]
\tikzstyle{level 3}=[level distance=1.5cm, sibling distance=1cm]
\tikzstyle{level 4}=[level distance=1.5cm, sibling distance=2cm]
\begin{tikzpicture}
%Start with the parent node, and slowly build out the tree
% with each "child" representing a new level of the diagram
% each "node" represents a labelled (or unlabeled if you
```

```
% want) node in the diagram.
\node {Player 1}
    child{
        child{
                        %Put the name of the node in parenthesis for
                        % reference later. The label shown in the diagram
                        % goes in the brackets. This label can use math mode.
            node(a){2}
            child{
                node{(0,0)}
                                %This allows us to attach a label to the
                                % edge between nodes. This label is just
                                % another node, so we can also name it and
                                % attach things to it.
                edge from parent
                node[left]{T}
            }
            child{
                node{(2,2)}
                edge from parent
                node[right]{B}
            }
                  %Invisible branch to make things align properly.
        } child{edge from parent[draw=none] }
    edge from parent
    node[left]{$D_1$}
    }
    child{
        node{1b}
        child{
            node(b){2}
            child{
                node{(0,0)}
                edge from parent
                node[left]{T}
            }
            child{
                node{(0,0)}
                edge from parent
                node[right]{B}
            }
        edge from parent
        node[left]{U}
        }
        child{
            node(c){2}
            child{
                node{(1,1)}
                edge from parent
                node[left]{T}
```

```
            }
            child{
                node{(2,2)}
                edge from parent
                node[right]{B}
            }
        edge from parent
        node[right]{D}
        }
    edge from parent
    node[right]{R}
    };
%Now I create the information set. Note that I utilize the names
% that I had previously assigned to nodes in my graph
\draw [dashed](a)--(b);
\draw [dashed](b)--(c);
\end{tikzpicture}
\end{center}
\end{figure}
```

Figure 2: A complicated Game tree



4.2 Graphical Model Example

I will not be spending too much time on graphical models in this tutorial. To create a graphical model you will need the following lines in your document preamble:

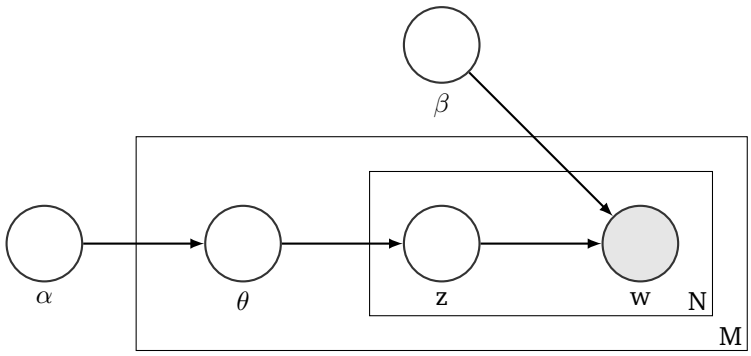
```
\usepackage{tikz}
\usetikzlibrary{fit,positioning}
```

The code to create graphical models is actually pretty straightforward compared to a lot of other things you could draw in LaTeX.

```
\begin{figure}[H]
\centering
\begin{tikzpicture}
\tikzstyle{main}=[circle, minimum size = 10mm, thick, draw =black!80, node distance =
 16mm]
%tells the lines what kind to be, don't mess with this
```

```
\tikzstyle{connect}=[-latex, thick]
%tells boxes to be black rectangles
\tikzstyle{box}=[rectangle, draw=black!100]
       %creates the nodes to go in the graphical model and gives them names and labels.
  \node[main, fill = white!100] (alpha) [label=below:$\alpha$] { };
  \node[main] (theta) [right=of alpha,label=below:$\theta$] { };
  \node[main] (z) [right=of theta,label=below:z] {};
  \node[main] (beta) [above=of z,label=below:$\beta$] { };
  \node[main, fill = black!10] (w) [right=of z,label=below:w] { };
  %creates the edges between nodes
  \path (alpha) edge [connect] (theta)
       (theta) edge [connect] (z)
       (z) edge [connect] (w)
       (beta) edge [connect] (w);
  %creates and positions the index for the plate
  \node[rectangle, inner sep=0mm, fit= (z) (w),label=below right:N, xshift=13mm] {};
  %creates the plate and tells it how much space to give the nodes
  \node[rectangle, inner sep=4.4mm,draw=black!100, fit= (z) (w)] {};
  \node[rectangle, inner sep=4.6mm, fit= (z) (w),label=below right:M, xshift=12.5mm] {};
  \node[rectangle, inner sep=9mm, draw=black!100, fit = (theta) (z) (w)] {};
\end{tikzpicture}
\end{figure}
```

This code gives us the following graphical model for Latent Dirichlet Allocation. Building off of this example code should be able to get you any graphical model you want. There are a number of good websites devoted to creating really nice graphical models if you are feeling adventurous. Check this site out: http://www.ruudwetzels.com/latexgm (warning, you can get into some pretty crazy stuff trying to draw really complicated graphical models).



4.3 Algorithm Example

The final example I am going to cover here is creating your own algorithms. This environment is probably the easiest to use overall and can be accessed by using the following package:

```
\usepackage[ruled,vlined]{algorithm2e}
%custom blue colored comments for algorithms
\newcommand{\com[1]}{ \textcolor{blue}{ \# {#1} }}
```

There are a number of specific commands defined for this environment which are described in the commented code below:

```
\begin{algorithm}[H]
\caption{\scshape{How to write algorithms}} % the \scshape command makes the letters
%be all small caps so they look realy nice.
 \SetAlgoLined %creates nice rules above and below the algorithm
 \KwData{this text} %put all of the data required by the algorithm in these bracketts
 \KwResult{how to write algorithm with \LaTeX2e } % what the algorithm will return.
 % I personally dont use this as I jsut have a return statement at the bottom.
 initialization\;
 \While{not at end of this document}{ %simple to create while loops
  read current\;
  \eIf{understand}{ % note that there can be no blank lines in an else/if statement
  %as the code will not compile
   go to next section\;
   current section becomes this one\;
   }{ % else part of the statement
   go back to the beginning of current section\;
  }
 }
 \KwRet{The data} %return the data
\end{algorithm}
```

Here is the output we get. This output is high enough quality to submit to a journal which is really nice.

Algorithm 1: How to write algorithms

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
  read current;
  if *understand* **then**
    go to next section;
    current section becomes this one;
  else
    go back to the beginning of current section;
  end
end
return *The data*

I will leave you with a more complicated example:

```
 \begin{algorithm}[H]
\caption{ {\scshape Calculate Connectedness Scores}}
 \SetAlgoLined
 \KwData{Number of Senators, Number of Bills, List of Bill Sponsors,
 List of Bill Cosponsors }
       Connectedness Scores $\leftarrow$ Vector(0, Number of Senators)\\
       Weight Matrix $\leftarrow$ Matrix(0, Number of Senators, Number of Senators)\\
       Distance Matrix $\leftarrow$ Matrix(0, Number of Senators, Number of Senators)\\
       Shortest Distance Matrix $\leftarrow$ Matrix(0, Number of Senators, Number
```

```
       of Senators)\\
      \com[Calculate weights]\\
      \For{ l= 0; l$<$ \emph{Number of Bills}; l++}{
      j $\leftarrow$ List of Bill Sponsors[l]\\
      Number of Bill Cosponsors $\leftarrow$ length(List of Bill Cosponsors[[l]])\\
      \For{k= 0; k$<$ \emph{Number of Bill Cosponsors}; k++}{
            Cosponsor $\leftarrow$ List of Bill Cosponsors[[l]][k]\\
            Weight Matrix[Cosponsor, Sponsor] += $\frac{1}{\text{Number of
            Bill Cosponsors} }$
      }
}
\com[Calculate Distances - They get smaller as weights get bigger]\\
\For{ i= 0; i$<$ \emph{Number of Senators}; i++}{
\For{ j= 0; j$<$ \emph{Number of Senators}; j++}{
      Distance Matrix[i,j] $\leftarrow$ $\frac{1}{\text{Weight Matrix[i,j]}}$
}
}
      \com[Calculate connectedness score for each Senator]\\
      \For{ j= 0; j$<$ \emph{Number of Senators}; j++}{
      \com[Now run Dijkstra's Algorithm]\\
      Sponsor Distances Vector $\leftarrow$ Distance Matrix[,j]\\
      Index Vector $\leftarrow$ c(1:Number of Senators)\\
      Closest Index $\leftarrow$ Index Vector[which(min(Sponsor Distances
      Vector[-j]))]\\
      Closest Value $\leftarrow$ Sponsor Distances Vector[which(min(Sponsor
      Distances Vector[-j]))]\\
      Shortest Distance Matrix[Closest Index,j] $\leftarrow$ Closest Value\\
      \bigskip
      \com[take out the current receiver]
      Sponsor Distances Vector $\leftarrow$ Sponsor Distances Vector[-j]\\
      Index Vector $\leftarrow$ Index Vector[-j]\\

      \For{ i= 0; i$<$ \emph{Number of Senators - 1}; i++}{
      \For{ k= 0; k$<$ \emph{length(Sponsor Distances Vector)}; k++}{
      Sponsor Distances Vector[k] $\leftarrow$ min(Closest Value, Distance
      Matrix[Index Vector[k],Closest Index], Distance Matrix[Index Vector[k],j]
            }
            Sponsor Distances Vector $\leftarrow$ Sponsor Distances Vector[-Closest
             Index]\\
            Index Vector $\leftarrow$ Index Vector[-Closest Index]\\
            \com[update to new closest index and value]
            Closest Index $\leftarrow$ Index Vector[which(min(Sponsor Distances
            Vector))]\\
            Closest Value $\leftarrow$ Sponsor Distances Vector[which(min(Sponsor
            Distances Vector))]\\
            Shortest Distance Matrix[Closest Index,j] $\leftarrow$ Closest Value\\
            }
      \bigskip
      Connectedness Scores[j] $\leftarrow$ $\frac{\text{Number of Senators - 1} }{
      \sum \text{Shortest Distance Matrix[,j] }}$
```

```
       }
\KwRet{\emph{Connectedness Scores}}
\end{algorithm}
```

Here is a link to the manual for the package, which has lots of helpful examples:
http://www.tug.org/texlive/Contents/live/texmf-dist/doc/latex/algorithm2e/algorithm2e.pdf

Algorithm 2: CALCULATE CONNECTEDNESS SCORES

Data: Number of Senators, Number of Bills, List of Bill Sponsors, List of Bill Cosponsors
Connectedness Scores ← Vector(0, Number of Senators)
Weight Matrix ← Matrix(0, Number of Senators, Number of Senators)
Distance Matrix ← Matrix(0, Number of Senators, Number of Senators)
Shortest Distance Matrix ← Matrix(0, Number of Senators, Number of Senators)
# Calculate weights
for *l= 0; l<* Number of Bills*; l++* **do**
  j ← List of Bill Sponsors[l]
  Number of Bill Cosponsors ← length(List of Bill Cosponsors[[l]])
  for *k= 0; k<* Number of Bill Cosponsors*; k++* **do**
    Cosponsor ← List of Bill Cosponsors[[l]][k]
    Weight Matrix[Cosponsor, Sponsor] $+= \frac{1}{\text{Number of Bill Cosponsors}}$
  end
end
# Calculate Distances - They get smaller as weights get bigger
for *i= 0; i<* Number of Senators*; i++* **do**
  for *j= 0; j<* Number of Senators*; j++* **do**
    Distance Matrix[i,j] ← $\frac{1}{\text{Weight Matrix[i,j]}}$
  end
end
# Calculate connectedness score for each Senator
for *j= 0; j<* Number of Senators*; j++* **do**
  # Now run Dijkstra's Algorithm
  Sponsor Distances Vector ← Distance Matrix[,j]
  Index Vector ← c(1:Number of Senators)
  Closest Index ← Index Vector[which(min(Sponsor Distances Vector[-j]))]
  Closest Value ← Sponsor Distances Vector[which(min(Sponsor Distances Vector[-j]))]
  Shortest Distance Matrix[Closest Index,j] ← Closest Value

  # take out the current receiver Sponsor Distances Vector ← Sponsor Distances Vector[-j]
  Index Vector ← Index Vector[-j]
  for *i= 0; i<* Number of Senators - 1*; i++* **do**
    for *k= 0; k<* length(Sponsor Distances Vector)*; k++* **do**
      Sponsor Distances Vector[k] ← min(Closest Value, Distance Matrix[Index Vector[k],Closest Index], Distance Matrix[Index Vector[k],j]
    end
    Sponsor Distances Vector ← Sponsor Distances Vector[-Closest Index]
    Index Vector ← Index Vector[-Closest Index]
    # update to new closest index and value Closest Index ← Index Vector[which(min(Sponsor Distances Vector))]
    Closest Value ← Sponsor Distances Vector[which(min(Sponsor Distances Vector))]
    Shortest Distance Matrix[Closest Index,j] ← Closest Value
  end

  Connectedness Scores[j] ← $\frac{\text{Number of Senators - 1}}{\sum \text{Shortest Distance Matrix[,j]}}$
end
return Connectedness Scores