The goal here is to provide a very short and practical tutorial to get things done when it comes to drawing syntactic trees. I try to summarize the workings of one online generator and two LaTeX packages, so this is by no means intended as an exhaustive list of resources.

1 An online tree-generator: RSyntaxTree

- You can access it through this link.
- As the title says, this is an online generator. One of its advantages is that you don't have to install anything on your computer and you can use it from any computer. It is also free.
- A further advantage is that it creates tree diagrams from linear representations using brackets. These representations are often used in syntax papers and books, so this is something you will know anyway. If not, don't worry! Bracket representations are pretty straightforward.
- The generator also provides resources to format the tree (e.g., color vs. black and white trees, different fonts and sizes, output formats etc).

1.1 How to use it

- Here is a tree diagram for the DP *the cake*:\(^1\)

```
   DP
   /\  
  D  NP
  /   |
the  N
    |
    cake
```

- How to draw this DP:
  - Brackets represent tree nodes and branching.
  - The label is inserted after the left bracket, e.g., [DP

---

\(^1\)This is the very first tutorial I am writing. If you have any suggestions, complaints or comments, I will be very happy to hear from you! Also, should you have any question, feel free to ask.

\(^1\)Regarding the trees here, maybe you will be familiar with other labels, conventions and presuppositions. The point here is to see how a tree diagram can be drawn with RSyntaxTree.
Suppose you want DP to be a binary branching node. Then, insert a pair of brackets for each of the intended daughters: \([DP \ [\ [\ ]]\]

- More generally: if you want a given node to be an \(X\)-ary branching node, then insert \(X\) pairs of brackets inside this node.
- You can try exercise 1 below.

Now, \([DP \ [\ [\ ]]\) won't give us anything. We have to add labels to the brackets: \([DP \ [D \ ] \ [NP]]\)

Suppose we want NP to have a single child, as we see in our target representation. Just apply the general rule, i.e., insert a single pair of brackets: \([DP \ [D \ ] \ [NP \ [N \ ]]]\)

The last step is to add the lexical entries as the terminal nodes. Add a single space after the label, insert the lexical entry and then hit the space bar again. Once we add the and cake, this is the final result: \([DP \ [D \ the \ ] \ [NP \ [N \ cake \ ]]]\)

- **NB**: This is the order that I thought would be the easiest way to present what there must be in the bracket representation to generate a tree. Once you get used to the generator, you will have your own way of drawing trees.

- We can also use triangles to summarize chunks of the tree whenever we don't want to show its internal structure.

  - This is good to know, but not good practice for your homework!
  - Suppose we want to draw the DP *the conclusion that syntax is fabulous*.

```
  DP
   \-- D
      the
   \-- NP
      N conclusion
       \-- CP
          that syntax is fabulous
```

- *The conclusion* with a placeholder for its complement looks like this:
  \([DP \ [D \ the \ ] \ [NP \ [N \ conclusion \ ] \ [CP \ ]]]\)

Just write the longer phrase inside the CP bracket. The longer phrase will be automatically converted into a triangle.

\([DP \ [D \ the \ ] \ [NP \ [N \ conclusion \ ] \ [CP \ that \ syntax \ is \ fabulous\ ]]]\)

**Exercises**

1. Draw a ternary-branching mini-tree with A as the mother and B, C and D as the daughters.

2. Work out the syntax to generate the following tree:
2 Drawing trees with \textit{\LaTeX}

Before getting to trees:

- \textit{This} is a short and very useful tutorial for all things linguistics if you use \LaTeX. It summarizes in a very direct way some of the most used packages that linguists use.

- Packages that may be particularly useful for linguists are those used to number examples. Linguex is the easiest option. Gb4e is a little more complex, but is also more refined.

2.1 \texttt{Qtree}

- \texttt{Qtree} is possibly the most straightforward package to draw syntactic trees with \LaTeX.

- It also generates trees from bracket representations.

2.1.1 How to use it

- Let's start with our cake example again:

\[
\begin{array}{c}
\text{DP} \\
\text{D} \quad \text{NP} \\
\text{the} \quad \text{N} \\
\text{cake}
\end{array}
\]

- This is what you need in your preamble: \texttt{\usepackage{qtree}}

- Here is the syntax: \texttt{\Tree [.DP [.D the ] [.NP [.N cake ] ] ] ]}
  
  - If you’re following this tutorial linearly: notice how the notation is similar to that used in RSyntaxTree. But also pay attention to the crucial differences below.
  
  - Notice the use of dots before the labels. Things go wrong if you forget them.
  
  - Notice also the space between either a label or a terminal node and the left bracket. If you forget it, it could even be the case that no file will be generated.
○ **Tip:** it is worth compiling your file several times while you draw the tree. This way, mistakes are easier to spot.

○ **Tip:** a very common problem is to forget to close brackets. A way to prevent this from happening is to type both the left and the right brackets before adding stuff between them.

- It is also possible to draw triangles using `qtree`.

```
\Tree [.DP [.D the ] [.NP [.N conclusion ] \proof{that syntax is fabulous}.CP ] ]
```

○ Here is how:

- What we need is the command `\proof{}`. The sequence that you want to go under the triangle is the argument of `\proof{}`.
- Don’t forget to add the `\label{}` (always preceded by a dot!) after `\proof{}`. Otherwise, this may also cause problems when compiling the file.

**Exercise**

Same exercise from before: work out the syntax to generate the following tree:

```
```

○ **Tip:** intermediate and zero levels are generated with `\textsubscript{1}` and `\textsubscript{0}`, respectively (add them right after the label, e.g., `V` and `T`).
2.2 Tikz

- Tikz is a very powerful package to draw things in general with LaTeX.
- But it is also fairly complex. The good news is that there is a version that is compatible with qtree's syntax, which is considerably simpler.
- This is a useful and easy to understand tutorial.

2.2.1 How to use it

- Start by adding this package to use tikz with the qtree syntax: \usepackage{tikz-qtree}.
- To use tikz, you'll need a tikzpicture environment:
  \begin{tikzpicture}
  ...
  \end{tikzpicture}
- Now just copy the code that we used with qtree above:
  \begin{tikzpicture}
  \Tree [.DP [.D the ] [.NP [.N cake ] ] ]
  \end{tikzpicture}
  This is the output:
  \begin{center}
  \begin{tikzpicture}
  \Tree [.DP [.D the ] [.NP [.N cake ] ] ]
  \end{tikzpicture}
  \end{center}

2.2.2 Drawing arrows

- You may be wondering why one would use tikz and not just qtree if it is more complicated.
  \begin{itemize}
  \item One reason is that it is much simpler to draw arrows (representing, e.g., movement) with tikz.
  \item It is also possible to do it using qtree (you can check this here).
  \item But I think tikz allows for a more fine-grained control of the arrow (and of the overall tree).
  \end{itemize}
- Here, I’ll give just one example of how to draw an arrow with tikz, so that you can get things done if you want to do it too. Check the tutorial mentioned above for a more complete guide.
• Here is how you can draw this tree:

  o This is the syntax without the arrow:

    \begin{tikzpicture}
    \end{tikzpicture}

  o First, identify the elements that you want to connect with the arrow, namely, $Y$ and $X$. This is done with the following syntax:

    \node(a){b};

    - $a$ is the name you are going to refer the relevant node with and $b$, the actual label that will show up in the output tree.
    - Suppose you already have $X$ in your tree and want to name it $x$. This will be our resulting syntax for node $X$ in our tree:

    \node(x){X};

    - **Tip**: make sure you don't forget the ; at the end!

  o If we do the same with $Y$, here is the final result (with the complete tree):

    \begin{tikzpicture}
    \end{tikzpicture}

  o Now, just add the command to actually draw something:

    \draw[<->] (x) to [bend right=90] (y);

    - The optional argument [<-] is specifying where the tip of the arrow should go.
    - [bend right=90] just adds some curve to the arrow. Erase it and you will get a straight line.

    Play with the angle to get the arrow to look the way you want.

    - **Tip**: again, don't forget ; at the end.

  o This is the final code:

    \begin{tikzpicture}
    \draw[<->] (x) to [bend right=90] (y);
    \end{tikzpicture}