Passo 22: Adicionar ambiente para múltiplas figuras com `subfigure`

- Primeiro, carregaremos um novo pacote: `subcaption`.

```latex
\usepackage{subcaption} % subfigures
```

- Em seguida, vamos fazer o upload das figuras `relu.png` e `sigmoid.png` para o diretório `figs`.

- Então, vamos construir um bloco para as duas figuras:

```latex
\begin{figure}
    \centering
    \begin{subfigure}[b]{0.3\textwidth}
        \centering
        \includegraphics[width=\textwidth]{figs/relu.png}
        \caption{Relu.}
    \end{subfigure}
    \hfill
    \begin{subfigure}[b]{0.3\textwidth}
        \centering
        \includegraphics[width=\textwidth]{figs/sigmoid.png}
        \caption{Sig. { \color{red} --} $a=0.5$
                     { \color{green} --} $a=1$
                     { \color{blue} --} $a=1.5$.
                }
    \end{subfigure}
       \caption{Exemplos de funções de ativação.}
       \label{fig:plots}
\end{figure}
```

- Analisemos os elementos de `subfigure`
  - `\centering`
    - O primeiro `\centering` centraliza o ambiente inteiro
    - O segundo `\centering` centraliza a primeira figura
    - O terceiro `\centering` centraliza a segunda figura
  - `\textwidth`
    - Este comando retorna a largura do texto.
    - `x\textwidth` significa `x` vezes a largura do texto. Em nosso exemplo, usamos 30%, vistoq que `\textwidth` é uma largura medida da esquerda à direita.
  - `\caption`
    - Podem ser até três: um para o bloco e um para cada figura.
  - `\hfill`
    - É um preenchimento horizontal da linha com espaços. Poderíamos usar `\\` para quebra a linha também.
    - `\color`: usado para colorir texto com cores padrão. Neste exemplo, estamos simulando linhas coloridas com

`--`. nas cores vermelho, verde e azul em correspondência às sigmoides do gráfico.

- As cores são predefinidas pelo pacote `xcolor`. Vide **Passo 24**.

Passo 23: Adicionar ambiente para código com o pacote `listings`

- Criamos um ambiente de código com `\lstinputlisting` para "puxar" um código escrito em Python contido no diretório `py`.

```latex
\lstinputlisting[language=Python,
                 style=mycode,
                 caption=Código Python.]
                 {./py/func-ativ.py}
```

Passo 24: Adicionar capacidades no documento por inclusão com `input`

- O bloco de código só funcionará se carregarmos um arquivo de estilo definido por nós no preâmbulo: `style.tex`.

```latex
\input{./misc/style.tex} % inserindo arquivo externo de estilo
```

- Podemos estilizar códigos em várias outras linguagens também.

Passo 25: Discutir o conteúdo do arquivo `misc/style.tex`

- Podemos desmembrar um arquivo `.tex` em várias partes e criar uma estrutura modular a ser remontada com `\input`.
- Por exemplo, em uma tese, poderíamos incluir em um arquivo principal `main.tex`, a seguinte estrutura capitular:

```latex
\input{cap1.tex}
\input{cap2.tex}
\input{cap3.tex}
...
```

- O desmembramento permite que trabalhemos em cada capítulo de maneira separada e compilemos um de cada vez. O código abaixo só compilaria o primeiro capítulo, por exemplo:

```latex
\input{cap1.tex}
%\input{cap2.tex}
%\input{cap3.tex}
...
```

Passo 26: Discutir o conteúdo do arquivo `misc/rnaas.tex`

- Produzindo uma caixa de texto com estilo "papel rasgado" (`tornpaper`) com o pacote `tikz`.
- Comentar sobre a definição de funções com `\def`.

Passo 27: Discutir espaçamento de texto e tamanhos de fonte padronizados

- Espaçamento de linhas na horizontal e vertical

```latex
\smallskip
\medskip
\largeskip
\hspace{1cm}
\vspace{1cm}
\vspace*{1cm} (recuo negativo)
\quad
\qquad
```

- Tamanhos de fonte

```latex
\Huge
\huge
\LARGE
\Large
\large
\normalsize
\small
\footnotesize
\scriptsize
\tiny
```