ftc-notebook — Formating for FIRST Tech Challenge (FTC) Notebooks*

FTC 9773, Robocracy†

Released Feb 16th, 2019

Abstract

The `ftc-notebook` package will greatly simplify filling entries for your FTC engineering or outreach notebook. We build on top of LaTeX, a robust system that can easily accommodates 100+ pages of documents, figures, and tables while providing support for cross-references. We developed this package to support most frequently used constructs encountered in an FTC notebook: meetings, tasks, decisions with pros and cons, tables, figures with explanations, team stories and bios, and more. We developed this package during the 2018-2019 season and are using it for our engineering notebook. Team Robocracy is sharing this style in the spirit of coopertition.

Contents

*This file describes version 1.1, last revised on 2019/02/16.

†E-mail: ftcrobocracy@gmail.com

List of Figures

List of Tables

1 Overview

The LaTeX package `ftc-notebook` provides help to format a FIRST Tech Challenge (FTC) engineering or outreach notebook. Using this style, you will be able to seamlessly produce a high quality notebook. Its main features are as follows.

- Esthetically pleasing cover pages for the notebook and monthly updates.
- Easy to use format to enter a team story and a bio for each of the team members.
- Quick references using list of tasks, figures, and tables.
- Meeting entries separated into lists of tasks.
- Each task is visually labeled as one of several kind of activities, such as Strategy, Design, Build, Software,... Activity kind can be customized to reflect a team's particular focus.
- Support for supporting your decisions in clear tables that list the pros and cons of each of your decisions.
- Support for illustrating your robot using pictures with callouts. A callout is text in a box with an arrow pointing toward an interesting feature on your picture.
- Support for pictures with textual explanation, and groups of picture within a single figure.

We developed this style during the 2018-2019 FTC season and we used it successfully during our competitive season. Compared to other online documents, it is much more robust for large documents. By designing a common style for all frequent patterns, the document also has a much cleaner look. LaTeX is also outstanding at supporting references. Try combining it with an online service like Overleaf, and your team will be generating quality notebooks in no time by actively collaborating online.

We developed this package to require little knowledge of LaTeX. We have tried to hide of the implementation details as much as possible. We explain LaTeX concepts as we encountered them in the document, so we recommend that LaTeX novices read the document once from front to back. Experienced users may jump directly to figures and sections explaining specific environment and commands.

The overall structure of an FTC notebook should be as shown in Figure 1 below.

```
\documentclass[11pt]{article}
\usepackage[Num=FTC~9773, Name=Robocracy]{ftc-notebook}
\begin{document}
  % 1: cover page and lists
  \CoverPage{2018-19}{robocracy18.jpg}
  \ListOfTasks
  \ListOfFigures
  \ListOfTables

  % 2: start of the actual notebook with optional team story and bios
  \StartNotebook
  \input{src/story.tex}
  \input{src/bio.tex}

  % 3: meeting entries with optional month delimiters
  \Month{August}{aug18.jpg}
  \input{src/aug19.tex}
  \input{src/aug21.tex}
  % repeat for successive months until the end of your successful season
\end{document}
```

Figure 1: Template for notebook.

A document consists of three distinct parts. First, we generate a cover page, followed by lists of tasks, figures, and tables. Pages use alphabetical numbering, as customary for initial front matter. As shown in Figure 1, a Latex document starts with a `\documentclass` command, followed by a list of packages used, and then a `\begin{document}` command. In LaTeX, comments use "%."

Second, we indicate the beginning of the actual notebook using the `\StartNotebook` command. Pages are then numbered with numerical page numbers starting at 1. A team story and team bio can be entered here, and have specific LaTeX commands detailed in Sections 4 and 5. For users unfamiliar with LaTeX, `\input` commands are used to include separate files whose file names are passed as arguments. The included files are processed as if they were directly listed in the original file. We will use this feature extensively to manage large documents such as an engineering notebook.

Third, we have the actual content of the notebook. We structure entries by meeting and suggest that each meeting uses a distinct input file for its text and a corresponding subdirectory for its supporting material, such as pictures. A meeting entry typically consists of a list of tasks. Optionally, a new month can be started with a cover page that includes a picture that highlights the accomplishment of the team for that month.

We strongly recommend that you use the following file structure for your notebook.

```
Directory structure:
  notebook.tex:     Your main latex file.
  ftc-notebook.sty: This package files, unless the package was
                    installed in your LaTeX install directory.
  newmeeting.sh:    A bash script that allows you to create a new
                    meeting file that is pre-filled. The script can
                    be customized for your team.
  src:              A directory where all the meeting info will go.
  |
  --> image:        A subdirectory where all the global pictures will
  |                 go. We recommend to place there the team logo,
  |                 team picture, and monthly pictures.
  |                 Pictures are searched there by default.
  --> aug19.tex     A file that includes all the text for your
  |                 (hypothetical) August 19th meeting.
  --> aug19:        A subdirectory where with all the images
                    needed for your aug19.tex LaTeX file.
```

Figure 2: Directory structure.

We recommend to use a pair of "date.tex" LaTeX file and "date" subdirectory for each meeting. This structure minimizes the risk of name conflicts for pictures and other attachments during the FTC season. Generally, directories logically organized by dates also facilitate searching for specific information.

2 Package Description and Customization

The package parameters are used to customize the notebook with the team name, team number, and the filename for a jpeg logo file. These are required parameters, as this style will not work without them. In particular, the logo file is expected to be either in the root directory or in the src/image subdirectory. The second line of Figure 1 illustrates the mandatory parameters that we used for our team. Use a "~" character instead of a space when defining your parameters.

In addition, each task in an FTC season is categorized by 6 distinct activity kinds. The default activity kinds are "Strategy," "Design," "Build," "Physics and Math," "Software," and "Team". The default values can be changed easily using the Kind parameters listed in Table 1.

option name	description	default
`Num`	team number	FTC~000
`Name`	team name	Outstanding~Team
`Notebook`	notebook type	Engineering~Notebook
`Green`	green color scheme	(default)
`Orange`	orange color scheme	
`Blue`	blue color scheme	
`Grid`	enables grid overlay over pictures	
`KindOne`	first task kind	Strategy
`KindTwo`	second task kind	Design
`KindThree`	third task kind	Build
`KindFour`	fourth task kind	Physics and Math
`KindFive`	fifth task kind	Software
`KindSix`	sixth task kind	Team

Table 1: Parameters for package, using "~" wherever a space is needed.

Additionally, default colors can be changed using one of the Green, Orange, or Blue parameters. If more custom colors are desired, the `\definecolor` commands as shown below can be used to provide a specific RGB triplet of values for each of the five colors used by the package.

```
\definecolor{TitleColor}{rgb}{0.65, 0.73, 0.29}
\definecolor{MainTableHeaderColor}{rgb}{0.84, 0.96, 0.29}
\definecolor{MainTableCellColor}{rgb}{0.70, 0.82, 0.32}
\definecolor{NormalTableHeaderColor}{rgb}{0.84, 0.96, 0.29}
\definecolor{NormalTableCellColor}{rgb}{0.94, 0.99, 0.78}
\definecolor{NormalTableCellWhite}{rgb}{1.0, 1.0, 1.0}
```

Figure 3: Example of how to change the default colors of the titles.

These commands must be placed after the `\usepackage` command and before the `\begin{document}` command.

3 Entering Text for a Meeting

In this section, we describe the environments and commands used to generate a report summarizing the activities of a meeting. A meeting can report a day's worth of work, or a weekend, or even a week. Our preference is to use one meeting report per weekend.

The structure of a meeting is as follows. It starts with a high level description of the meeting, including date, title, and a list of members that participated to the meeting. The high level description also includes a list of task accomplished. The

goal of this high level description is to allow a team to quickly locate prior tasks that were accomplished in a meeting. No-one will have to read the whole file to determine if a new drivetrain was attempted on that weekend, or not. It should be right at the beginning of the meeting file.

After the high level info, we will have the text, figures, and tables associated with each of the tasks.

3.1 Meeting Description

`Meeting` Let us now look at the format of a meeting entry, for example for an hypothetical August 19 meeting written in a "src/aug18.tex" LaTeX file.

```
\begin{Meeting}[<kind>]%
  {<title>} {<date>} {<duration>} {<members>}%
  {% list of tasks
    \TaskInfo{<task title>}{<task label>}{<task reflection>}%
  }
  % meeting info
\end{meeting}
```

Figure 4: Meeting entry (first and last command in a given meeting file).

The overall structure is a `Meeting` environment. In LaTeX, environments are delineated with a `\begin{env}` command and a matching `\end{env}` command at the end of the environment. The `\begin{Meeting}` command initialize the `Meeting` environment with one optional argument and five mandatory arguments. This command should be the first LaTeX command in the "src/aug18.tex" file.

In LaTeX, mandatory arguments are given in curly braces, and optional arguments are passed in square brackets. For commands with long parameters, it is tempting to separate the arguments in many lines. When doing so, especially with optional arguments, we recommend to start a comment at the end of each of the parameter lines (i.e. add a "%" character). This will let LaTeX know that the next lines may contain an additional argument.

The first argument `[`⟨*kind*⟩`]` is optional and describes the meeting kind. A typical meeting kind may be "Meeting," "Preseason," "Competition," "Outreach," or something of this sort. "Meeting" is used by default if the optional argument is omitted. The second argument `{`⟨*title*⟩`}` is mandatory and indicates the meeting's title. The third argument `{`⟨*date*⟩`}` is mandatory and provides the meeting's date or date range. The fourth argument `{`⟨*duration*⟩`}` is mandatory and describes the duration of the meeting. The fifth argument `{`⟨*members*⟩`}` is mandatory and lists the name of the team members present at the meeting. The sixth and last argument is mandatory and provides a list of tasks performed at the meeting. This list provides one `\TaskInfo` command per task. This command is detailed in Section 3.2.

After the `\begin{Meeting}` command, we have the actual content of the meeting. We provide many commands to properly format most typical entries, such as sections,

subsections, figures, tables, list of decisions... These commands will be detailed in subsequent sections.

The file is terminated by the `\end{Meeting}` command to indicate that all of the info about this meeting has be given. It will print a box where team members can sign the entry as requested by FTC best practices.

`\Signee` When giving the team member names in the mandatory `{`⟨*members*⟩`}` argument for the `Meeting` environment, you can indicate the members that should sign the entry using the `\Signee` command with the `{`⟨*name*⟩`}` argument. For example, if Ann, Ben, and Carol attended the meeting and you would like Ben and Carol to sign the entry, the `{`⟨*members*⟩`}` should be given as follows: "Ann, `\Signee`{Ben}, `\Signee`{Carol}".

3.2 Task Description

`\TaskInfo` A meeting is structured in several tasks. Each task is first described within the `\begin{Meeting}` command's sixth argument using the `\TaskInfo` command. This command has three mandatory arguments, as shown in Figure 4.

The first argument `{`⟨*task title*⟩`}` indicates the title of the task. The second argument `{`⟨*task label*⟩`}` provides a unique label for this task. The third argument `{`⟨*task reflection*⟩`}` is used to provide a short reflection on the status of this task. This can typically be something like "we discovered new problems," "we realized that this approach works much better than previous solutions," or any other high-level observation that you want to share with the readers.

If you accomplished three tasks during a given meeting, then three `\TaskInfo` entries must be provided within the list of tasks.

Now is a good time to mention how labels and references are used in LaTeX. Wherever a reference is needed, we must give a unique string that will describe the location in the document associated with the label, typically a page, task, figure, or table number. We recommend to structure the label with the kind (`task`, `fig`, `tab`), followed by the file name, followed by a string that make sense to the particular label. Later in the text, you can explicitly reference the given label, for example with "Task~`\ref{task:aug19:challenge}`," "Figure~`\ref{fig:aug19:lift}`," or "Table~`\ref{tab:aug19:decision}`." For users not familiar with LaTeX, a "~" character is used to represent a space where the text before and after the tilde cannot be separated by an end-of-line.

`\TaskRef` `\TableRef` `\FigureRef` We also provide custom commands to generate the task/table/figure number as well as the page number of where to find the task/table/figure: `\TaskRef{<label>}`, `\TableRef{<label>}`, or `\FigureRef{<label>}`. These commands are convenient to refer to task/table/figures far in the past.

`\Task` Between the `\begin{Meeting}` and `\end{Meeting}` constructs, we must provide detailed info for each task. Below is the command used to do to start the actual description of a given task.

```
\Task [<prior task>] {<first kind>} [<second kind>]
```

Figure 5: Start of a task description.

This command will start a new task section. One such command is needed for each of the `\TaskInfo` listed in the `\begin{Meeting}` command. This command will reuse the title listed by the `\TaskInfo` command in the same order. We decided on this structure so that if a team member were to change the title of the task in the `\TaskInfo` arguments, this change would automatically be reflected at the start of the task's description here.

The first optional argument `[`⟨*prior task*⟩`]` let us indicates if this task is a continuation of one or more prior tasks. We strongly recommend to fill in this argument when appropriate. For example, if a given task is the continuation of a task with label `task:aug06:challenge`, the argument should be set to `\TaskRef{task:aug06:challenge}`.

The second mandatory argument `{`⟨*first kind*⟩`}` indicates which kind of tasks this is. Recall that we are able to change the default kinds in Section 2 in Table 1. If the type corresponds to the first kind (e.g. "Strategy" by default), then we expect the number "1" here. Acceptable values are numbers between 1 and 6, inclusively. Sometimes, it may be hard to classify a task with only one type: the optional third argument `[`⟨*second kind*⟩`]` let us input a second kind. The order between the first and second numbers is not important.

3.3 Sections and Lists

`\Section` `\Section*` `\Subsection` `\Subsection*`

Now that we have a task section, we can can start filling the description for that task. Most likely, we will want to have sections and subsections to group related material together. `\Section{<title>}` and `\Subsection{<title>}` respectively starts a numbered section and subsection with the given title. The starred versions are variants that omit the section numbering.

`\MeetingSummary`

The `\MeetingSummary` simply emits a unnumbered section title with a "Meeting Summary" title.

`EnumerateWithTitle` `ItermizeWithTitle`

A frequent pattern is to have a section title followed by a list of items. In LaTeX, numbered lists are referred as `enumerate` environment and unnumbered or bulleted lists are referred as `itemize` environment. Regardless of the type of list, elements of the list always start with an `\item` command.

While traditional LaTeX list environments can be used, we define here two custom list environments that provide for an unnumbered section title followed by a list of numbered or unnumbered list element. The figure below illustrates a possible use of such constructs.

```
\begin{EnumerateWithTitle}{A numbered list title}
  \item one enumeration
  \item another enumeration, add more as needed
\end{EnumerateWithTitle}

\begin{ItemizeWithTitle}{A bulleted list title}
  \item one bullet, add more as needed
\end{ItemizeWithTitle}
```

Figure 6: Titled sections with numbered and unnumbered lists.

In general, you can always use the LaTeX default list environments (to be used with `\begin{<environment>}` and `\end{<environment>}`, namely `enumerate`, `itemize`, or `compactitem` for, respectively, a number list of items, a list of bulleted items, or a compact list of bulleted items.

3.4 Tables

When entering data for a task, one pattern that we often use is a table of pros and cons arguments. We need a table with three columns: one describing the option considered, a second column describing the pros of this option, and a third column describing the drawbacks of this option. Below is an example of such a decision tables.

```
\begin{DecisionTable}
  {Decisions about lift}
  {tab:aug06:lift:decision}
   %
    \TableEntryTextItemItem{
      drawer slides
    } {% pros items
      \item works well
    } {% minus item
      \item heavy
    }
    \\ \hline
   %
    \TableEntryTextItemItem{
      rev slides
    } {% pros items
      \item light
    } {% minus item
      \item difficult under heavily load
    }
    % omit "\\ \hline" for last row
\end{DecisionTable}
```

Figure 7: Example of decision table with pros and cons.

Before going in the details of the environments and commands, Figure 7 illustrates a comparison for a lift, comparing drawer slides to REV slides. Each row is given by a `\TableEntry` type of commands, two in this case, separated by `\\ \hline` commands. This first type of commands provides the data for a given row, and that latter type of commands forces LaTeX to create a new line and separate the entries by a horizontal line inside of the table.

`DecisionTable` A decision tables always consists of a table environment with three columns. Its arguments are as follows.

```
\begin{DecisionTable} [<first col name>]%
  [<second col name>] [<third col name>]%
  {<caption>} {<label>}

\end{DecisionTable}
```

Figure 8: Decision table.

The first three optional arguments, `[`⟨*first col name*⟩`]`, `[`⟨*second col name*⟩`]`, and `[`⟨*third col name*⟩`]`, are used to change the default column names, respectively "Op-

tion," "Pro," and "Cons." The fourth mandatory argument {⟨*caption*⟩} is the title of the table. This caption will also be listed in the list of tables generated by the `\ListOfTables` command. The fifth mandatory argument {⟨*label*⟩} is a label to be used when referencing the table. We strongly encourage each table to be referenced at least once in the text.

`DescriptionTable`
`DescriptionTable*`

We often need tables with two columns. For this purpose, we propose the following `DescriptionTable` environments.

```
\begin{DescriptionTable} {<first col name>}%
  {<second col name>} {<caption>} {<label>}

\end{DescriptionTable}
```

Figure 9: Decision table.

The first two mandatory arguments {⟨*first col name*⟩} and {⟨*second col name*⟩} indicate the titles of each column. The next tow mandatory arguments {⟨*caption*⟩} and {⟨*label*⟩} provides, respectively, the caption and a label to refer to the table.

The two environments are very similar. The non-starred environment has a narrow column followed by a wider column, ideal for title followed by a longer text in the second column. The starred environment has two columns of equal sizes.

`\TableEntryTextTextText`
`\TableEntryTextItemItem`

The rows of tables with three columns are entered by the `\TableEntryTextTextText` and `\TableEntryTextItemItem` commands. Each take three mandatory arguments to input the data for each of the three columns in a given row. The first command takes three text entries, and the second command takes one text entry followed by two bulleted lists. As shown in Figure 7, the bulleted lists are entered as lines each starting by the `\item` command. In tables, LaTeX does not like empty lines. So do not include empty lines, and if you need a line break, you can always enter the `\\` new line command explicitly.

`\TableEntryTextText`
`\TableEntryTextItem`
`\TableEntryItemItem`

The rows of tables with two columns are similarly entered with the `\TableEntryTextText`, `\TableEntryTextItem`, and `\TableEntryItemItem` commands. Each takes two mandatory arguments, one for each column. Text or items are expected in the arguments depending on the name of the command used.

`RawDecisionTable`
`RawDescriptionTable`

Sometimes, you may need to get a decision or description table with an arbitrary number of columns of varying different sizes. The `RawDecisionTable` and `RawDescriptionTable` environments allow you to create such tables. They take four mandatory arguments: a column format given by {⟨*format*⟩}, an "&" separated list of column titles given by {⟨*titles*⟩}, a caption text given by {⟨*caption*⟩}, and a table label given by {⟨*label*⟩}. Please refer to the LaTeX `tabluar` environment for hints on how to express table column format.

```
\PictFigure [<location>] {<jpeg file>} [<horizontal fraction>]%
  {<caption>} {<label>} [<callout>]
```

Figure 10: Figure with one picture.

3.5 Figures

Figures are important in notebooks, and the package proposes several commands to help us with them. First, you need to know that LaTeX primarily likes JPEG format, so pictures will need to be converted to JPEGs. By convention, the pictures are expected in the "src/aug19" subdirectory when processing the "src/aug19.tex" meeting entry. It's a convention, not mandatory, but useful to follow as long term there are lots of pictures, so grouping them by meeting is convenient.

`\PictFigure` [H]

The first, simplest command will generate a figure with a caption for a single JPEG file. It expects the following arguments.

The second mandatory argument `{`⟨*jpeg file*⟩`}` provides the path to the JPEG file as well as the file name, including the ".jpg" extension. It is best to use the full path from the root directory, for example "src/aug19/game.jpg". The third optional argument `[`⟨*horizontal fraction*⟩`]` provides an optional fraction (between 0 and 1) that indicates the fraction of the horizontal page that should be utilized for the picture. Default is 0.9 or 90% of the horizontal space. The fourth argument `{`⟨*caption*⟩`}` is the caption text, and the fifth argument `{`⟨*label*⟩`}` is a label string used to create a unique reference to this figure. The caption of each figure will also be listed in the list of figures generated by the `\ListOfFigure` command.

Figures are floating object, they move where LaTeX estimate they fit best. You can have some influence over the placement of the figures using the first optional argument `[`⟨*location*⟩`]`. Possible values are combinations of the entries shown in the table below.

Key	**Description**
h for here	try to put the figure close to where it is in the text
t for top	try to put it at the top of a page
b for bottom	try to put the figure at the bottom of a page
p for page	try to put it on a separate page (not with other text)
H for HERE	try harder to put it here
!	try to force your choice.
htb	You can put several choices at once, e.g. a frequent one is [htb] for here, or top, or bottom.

Table 2: Values for floating object location in LaTex.

The last optional argument `[`⟨*callout*⟩`]` is used to add one or more callouts. Callouts are used to overlay a text box on a figure, along with an arrow pointing to

the location of the interesting feature is. So for example, the `\Callout{-6,5}{Look Here}{-1,1}` command place a text box with text "Look Here" at the coordinate (X, Y) = (-6, 5) with an arrow starting at the box and pointing to the coordinate (X, Y) = (-1, 1). You may have arbitrary many callouts inside the `[`⟨*callout*⟩`]` optional argument. Recall that if you write the callout argument on a new line, we recommend to start a comment at the end of the prior argument lines (i.e. add a "%" character). This will let LaTeX know that the next line contains the optional argument.

Because callouts require us to know the precise coordinates at which to locate the text box and the arrow, we have added a parameter to the `ftc-notebook` package, namely `grid`, as defined in Table 1. When setting this package parameter, every figure will include a grid that will help us determine the proper coordinates for the callouts. Simply remove the parameter when done.

`\ExplainedPictFigure`

Another frequent pattern is to have a picture one side of the page, with a textual explanation next to it. For this pattern, we introduce the `\ExplainedPictFigure` command which add one parameter to the previous `\PictFigure` command to provide for a textual explanation.

```
\ExplainedPictFigure [<location>] {<jpeg files>}%
  [<horizontal fraction>] {<caption>} {<label>}
  {<explanation>} [<callout>]
```

Figure 11: An explained picture.

Because the picture and the textual explanation share the width of the page, we suggest to use a smaller fraction of the horizontal space for the picture. In Figure 11 we may use a `{`⟨*horizontal value*⟩`}` of 0.5 for 50% of the space for the picture, thus leaving the remaining 50% for the text. In the mandatory argument `{`⟨*explanation*⟩`}` we can enter a text such as the one below.

```
{% explanation (no empty line allowed);
  Interesting features:
  \begin{compactitem}
    \item compact
  \end{compactitem}
}
```

Figure 12: Example of an explanation text.

In this example, we have a phrase followed by a compact list of bullets.

`GroupedFigures`
`PictSubfigure`
`ExplainedPictSubfigure`

Another frequent pattern is to add a group of pictures. To handle this case, we introduce here a `GroupedFigures` environment to encapsulate several subfigures. The commands are as below.

```
\begin{GroupedFigures} [<locations>] {<caption>} {<label>}

  \PictSubfigure {<jpeg file>} [<horizontal fraction>]%
    {<caption>} {<label>} [<callout>]

  \ExplainedPictFigure {<jpeg files>}%
    [<horizontal fraction>] {<caption>} {<label>}
    {<explanation>} [<callout>]

\end{GroupedFigures}
```

Figure 13: Example of grouped figures.

The `\begin{GroupedFigures}` has the usual optional argument `[`⟨*location*⟩`]` and mandatory arguments `{`⟨*caption*⟩`}` and `{`⟨*label*⟩`}`.

In turn, all of the subfigures have arguments similar to the figures command previously seen, expect that they omit the optional location argument as they are already grouped in a single floating figure.

`RawPict` There is one additional command that is convenient for pictures.

```
\RawPict {<jpeg file>} {<horizontal fraction>} {<callout>}
```

Figure 14: Raw picture.

This `\RawPict` command is used to insert a figure directly in a table without labels, captions, and other arguments. It just take the JPEG file name, its horizontal size, and callout which can be left empty by using `"{}"`.

This give you the essentials of the `ftc-notebook` package, many more basic LaTeX command are useful, here is a list: `enumerate`, `itemize`, `textbf`, handling native `tabular` (tables in LaTeX) or `longtable` (tables that can be split among consecutive pages).

4 Team Story

`TeamStory` We can add a "Team Story" page by using the following environment.

```
\begin{TeamStory}{<moto>}

  % team story

\end{TeamStory}
```

Figure 15: Team story.

This environment let us enter a team story page with a mandatory {⟨*moto*⟩} argument, for example the team goal of the team for the season.

5 Team Biography

`Bio` `BioEntry`

We can also add a list of biographies for each team member as follows.

```
\begin{Bio}

  \BioEntry {<name>} {<title>}%
    [<role title>] {<role description>}%
    [<outreach role title>] {<outreach role description>}%
    {<jpeg file>} [<horizontal fraction>]%
    {<full bio>}

\end{Bio}
```

Figure 16: Team biography table.

The `Bio` environment creates a table in which the bio of each team member can be given. The format for each team member comprises of a picture given by the {⟨*jpeg file*⟩} argument with the team member's name from the {⟨*name*⟩} argument and title from the {⟨*title*⟩} argument.

Below the picture, we will also generate two role entries, each with an optional role title and a mandatory role description. This is can be used to highlight the team role and outreach role of each team member. The full bio given by {⟨*full bio*⟩} is listed in a second column, next to the picture.

6 Miscellaneous Commands

`\CoverPage`

The `\CoverPage` command is used to generate a cover page for your notebook. It takes two mandatory arguments: {⟨*date*⟩} and {⟨*jpeg file*⟩}. The date is used to describe the season, e.g. "2018-2019". The second argument gives the picture to be used on the cover page. By default, the picture is expected in the "src/image" directory, but you can provide a different file path to the file, e.g. "src/story/teampict.jpg".

`\ListOfTasks` `\ListOfFigures` `\ListOfTables` These commands create the corresponding list of tasks, figures, and tables. We recommend that they are used before the `\StartNotebook` command as the page numbering prior to the start command uses alphabetical page numbering. This will allow the notebook to grow, with the corresponding lists of tasks, figures, and tables to also grow without changing the page numbering of your older notebook entry pages.

`\StartNotebook` Use this command to separate the front matter (cover pages and list of content) from your actual notebook entry.

`\Month` The `\month` command is used to create a cover page for a new month. It takes two mandatory arguments: `{`⟨*month*⟩`}` and `{`⟨*jpeg file*⟩`}`. The first argument indicates the month, and the second arguments gives a path to the picture you want to use for the month cover page.