Graphics for LaTeX users

Agostino De Marco
Università degli Studi di Napoli Federico II
Dipartimento di Ingegneria Industriale
Gruppo Utilizzatori Italiani di TeX

GUIT meeting 2019
Politecnico di Torino
26 October 2019
Outline

General guidelines on illustration design

Drawing with LaTeX-aware software
  Using Inkscape + TexText extension

Drawing with natively available LaTeX environments/packages
  The standard environment picture
  The package pstricks (PostScript)
  The package tikz

Data plots with package pgfplots
General guidelines on illustration design

Drawing with \LaTeX-aware software
   Using Inkscape + TexText extension

Drawing with natively available \LaTeX environments/packages
   The standard environment picture
   The package pstricks (PostScript)
   The package tikz

Data plots with package pgfplots
The term *illustration* refers to all kind of pictorial graphics — photographs, drawings, diagrams, and schematics.
The term *illustration* refers to all kinds of pictorial graphics — photographs, drawings, diagrams, and schematics.

**The Three Regression Types**

Generalized Linear Models (GLM) extend the ordinary linear regression and allow the response variable $y$ to have an error distribution other than the normal distribution.

GLMs are:

- a) Easy to understand
- b) Simple to fit and interpret in any statistical package
- c) Sufficient in a lot of practical applications

### Linear Regression

- Econometric modelling
- Marketing Mix Model
- Customer Lifetime Value

#### Linear Regression

$y = \alpha_0 + \sum_{i=1}^{N} \alpha_i x_i$

- Continuous $\rightarrow$ Continuous
- 1 unit increase in $x$ increases $y$ by $\alpha$

### Logistic Regression

- Customer Choice Model
- Click-through Rate
- Conversion Rate
- Credit Scoring

#### Logistic Regression

$y = \frac{1}{1 + e^{-x}}$

- Continuous $\rightarrow$ True/False
- 1 unit increase in $x$ increases log odds by $\alpha$

### Poisson Regression

- Number of orders in lifetime
- Number of visits per user

#### Poisson Regression

$y \sim \text{Poisson}(\lambda)$

- Continuous $\rightarrow 0, 1, 2, ...$
- $\ln \lambda = \alpha_0 + \sum_{i=1}^{N} \alpha_i x_i$
- 1 unit increase in $x$ multiplies $y$ by $e^{\alpha}$

MarketingDistillery.com is a group of practitioners in the area of e-commerce marketing.

Our fields of expertise include:
- marketing strategy and optimization, customer tracking and on-site analytics, predictive analytics, econometrics, data warehousing, and big data systems, marketing channel insights in Paid Search, Social, SEO, CRM and brand.

(cc-by) Kamil Bartocha, MarketingDistillery.com
Illustrations

(a) An example of technical illustration showing the Reflex principle.

(b) A newspaper illustration. This example shows a particular kind of artwork known as ‘infographics.’

Figure 1: Examples of on-the-job technical illustrations.
It is important in typography to maintain a consistency between text and graphics.
Figure 2: A technical book in the hands of a reader. The right-hand page contains a full-height annotated illustration.
Illustrations – Benefits

Benefits coming from a careful use of visual material in technical documents:
Benefits coming from a careful use of *visual material in technical documents*:

- *Readers look for and want graphics.*
Benefits coming from a careful use of visual material in technical documents:

- Readers look for and want graphics.
- Graphics enhance a communication’s visual appeal, thereby increasing the readers’ concentration on its message.
Benefits coming from a careful use of visual material in technical documents:

- Readers look for and want graphics.
- Graphics enhance a communication’s visual appeal, thereby increasing the readers’ concentration on its message.
- Well-crafted graphics really can say more than many lines of text, much more efficiently than prose.
Benefits coming from a careful use of visual material in technical documents:

- **Readers look for and want graphics.**
- Graphics enhance a communication’s visual appeal, thereby increasing the readers’ concentration on its message.
- **Well-crafted graphics really can say more than many lines of text**, much more efficiently than prose.
- Graphics enable writers to convey information to readers who do not share a common language with the writers—or with each other.
Benefits coming from a careful use of visual material in technical documents:

- **Readers look for and want graphics.**
- Graphics enhance a communication’s visual appeal, thereby increasing the readers’ concentration on its message.
- **Well-crafted graphics really can say more than many lines of text,** much more efficiently than prose.
- Graphics enable writers to **convey information to readers who do not share a common language** with the writers—or with each other.
- **Graphics communicate information so effectively that they sometimes convey the entire message** (see Figure 1a, Reflex camera).
*Keep in mind* that, at some point, readers’ attention will be going back and forth between text and figures, necessarily.
Illustrations – Design guidelines

*Keep in mind* that, at some point, readers’ attention will be going back and forth between text and figures, necessarily.

*Make the effort* of having the readers feel at ease during the process.
Illustrations – Design guidelines

*Keep in mind* that, at some point, readers’ attention will be going back and forth between text and figures, necessarily.

*Make the effort* of having the readers feel at ease during the process.

Design graphics with a special *focus on usability*. 
Keep in mind that, at some point, readers’ attention will be going back and forth between text and figures, necessarily.

Make the effort of having the readers feel at ease during the process.

Design graphics with a special focus on usability.

Graphics should have the same good qualities of author’s prose, easy for readers to understand and use.
Figure 3: Aerospace engineering textbook.
http://wpage.unina.it/agodemar/DSV-DQV/DSV-DQV_Quaderno_17.pdf
Design to **support any possible readers’ tasks**. Imagine your readers in the act of using your graphic material.
Illustrations – Usability rules

Design to *support any possible readers’ tasks*. Imagine your readers in the act of using your graphic material.

**Consider carefully your readers’ knowledge and expectations.** Specialized graphics as opposed to simplified visuals (‘infographics’).
Design to **support any possible readers’ tasks**. Imagine your readers in the act of using your graphic material.

**Consider carefully your readers’ knowledge and expectations.** Specialized graphics as opposed to simplified visuals (‘infographics’).

**Seek for simplicity.** Especially for graphics that will be read on a computer screen or from a projected image. To keep it simple:

- Include only a manageable amount of material.
- Eliminate unnecessary details.
Design to support any possible readers’ tasks. Imagine your readers in the act of using your graphic material.

Consider carefully your readers’ knowledge and expectations. Specialized graphics as opposed to simplified visuals (‘infographics’).

Seek for simplicity. Especially for graphics that will be read on a computer screen or from a projected image. To keep it simple:

- Include only a manageable amount of material.
- Eliminate unnecessary details.

Seek for the effectiveness of textual labels. Important content should always be labelled clearly.
Illustrations – Usability rules

Design to **support any possible readers’ tasks**. Imagine your readers in the act of using your graphic material.

**Consider carefully your readers’ knowledge and expectations.** Specialized graphics as opposed to simplified visuals (‘infographics’).

**Seek for simplicity.** Especially for graphics that will be read on a computer screen or from a projected image. To keep it simple:

- Include only a manageable amount of material.
- Eliminate unnecessary details.

**Seek for the effectiveness of textual labels.** Important content should always be labelled clearly.

**Choose effective informative titles** (figure and table captions). Possibly, make them brief and informative at the same time.
Outline

General guidelines on illustration design

**Drawing with \LaTeX-aware software**

*Using Inkscape + TexText extension*

**Drawing with natively available \LaTeX environments/packages**

*The standard environment picture*

*The package pstricks (PostScript)*

*The package tikz*

**Data plots with package pgfplots**
Engineering illustrations – Example

Figure 4: Aerospace engineering textbook.

http://wpage.unina.it/agodemar/DSV-DQV/DSV-DQV_Quaderno_1.pdf
http://www.inkscape.org

**Inkscape** is an open source and well-supported vector graphics/SVG editor available for all major operating systems.

*Provides effective LaTeX-related capabilities*, e.g., the TexText Python-based plugin extension.

[https://textext.github.io/textext](https://textext.github.io/textext)

TexText provides the possibility to *add and re-edit (multi-line) LaTeX/XeLaTeX/LuaLaTeX generated SVG elements to a drawing.*
Figure 5: A screenshot of Inkscape with TexText extension in use.
Inkscape

(a) Selecting TexText from Inkscape Extensions menu.

(b) The TexText dialog window.

**Figure 6:** Using TexText extension plugin in Inkscape.
\documentclass{article}
% documentclass{article} 
% \input{default_packages}
% \pagestyle{empty}
% \begin{document}
% \usepackage{amsmath,amsthm,amssymb, amsfonts}
% \usepackage{color}
% % default: 
% % \begin{document}
% \pagestyle{empty}
% \begin{document}
% \usepackage{amsmath,amsthm,amssymb, amsfonts}
% \usepackage{color}
% \begin{document}
% \pagestyle{empty}
% \begin{document}

\begin{figure}[h]
    \centering
    \includegraphics[width=\textwidth]{example-image-a}
    \caption{Example figure with captions.}
    \label{fig:example}
\end{figure}

A. De Marco | Graphics for \LaTeX\ users | \LaTeX\ meeting 2019
**Figure 7:** SVG element resulting from user’s input compilation (see Figure 6b).

*The final SVG object is re-editable via the TexText dialog!*
Outline

General guidelines on illustration design

Drawing with \LaTeX-aware software
Using Inkscape + TexText extension

Drawing with natively available \LaTeX environments/packages
The standard environment picture
The package pstricks (PostScript)
The package tikz

Data plots with package pgfplots
Making drawings with code

A completely different paradigm.

No pseudo-synchronous visual tools.

Similar to the asynchronous typesetting workflow.
The native environment **picture**

\begin{quote}
\% in preamble
\usepackage{pict2e}
\%
\begin{picture}(120,80)
  \put(30,30){\circle*{3}}
  \put(30,33){\makebox(0,0)[br]{$A$}}
  \put(90,43){\circle*{3}}
  \put(88,47){\makebox(0,0)[bl]{$B$}}
  \linethickness{1.2pt}
  \Line(30,30)(90,43)
  \put(10,10){\vector(1,0){100}}
  \put(110,14){\makebox(0,0)[b]{$x$}}
  \put(10,10){\vector(0,1){60}}
  \put(14,70){\makebox(0,0)[l]{$y$}}
\end{picture}
\end{quote}

**Figure 8:** A drawing made with the standard picture environment enhanced by the pict2e package.
Drawing with \textit{pstricks} \hfill Figure 9: Placing whatever, wherever in a \textit{pspicture} environment.
Then we have \texttt{pgf} and \texttt{tikz} by Till Tantau ...

https://www.ctan.org/pkg/pgf


https://pgf-tikz.github.io/ (manual on the web)
The name **pgf** means *Portable Graphics Format*.

It is a package for creating **inline graphics**: defines a number of \TeX\ commands that can draw graphics within the typesetting process.

*Graphics objects are put into boxes* and treated as normal items to be taken care of by the \LaTeX\ output routine.

The package pgf exposes a **frontend layer**, i.e. a set of commands or a special syntax that makes using the functionalities implemented by basic layer easier.

This frontend is what is called **TikZ**, the \LaTeX\ package tikz that incorporates pgf.

The name **TikZ** is an acronym of *Tikz Ist Kein Zeichenprogramm* (German for ‘tikz is not a drawing program’).
In preamble: \texttt{usepackage\{tikz\}}

The package provides the command \texttt{\tikz} as in the following examples.

\texttt{\tikz \draw (0pt,0pt) -- (20pt,6pt);}

yields the line \(\overline{\quad}\), or

\texttt{\tikz \fill[color=orange] (1ex,1ex) circle(1ex);}

yields the orange circle \(\bullet\).

The argument passed to \texttt{\tikz} is a semicolon-terminated string.
The tikzpicture environment

More elaborate drawings are embedded into the environment `tikzpicture`:

```
\begin{tikzpicture}
  \draw (0,0) -- (1,0) -- (1,1) -- cycle;
\end{tikzpicture}
```

```
\begin{tikzpicture}
  \draw (0,0) rectangle (2,1);
  \draw (0,0) -- (2,1);
  \draw (0,1) -- (2,0);
\end{tikzpicture}
```
The tikzpicture environment

A tikzpicture can be used *inline* with the running text of a paragraph, like any other box object:

The following draws a
$0.4 \times 0.2$ crossed rectangle:
\begin{tikzpicture}
\draw (0.0,0.0) rectangle (0.4,0.2);
\draw (0.0,0.0) -- (0.4,0.2);
\draw (0.0,0.2) -- (0.4,0.0);
\end{tikzpicture}.

The following draws a $0.4 \times 0.2$ crossed rectangle: □.
Inside a `tikzpicture` environment *everything is drawn by starting a path and by extending the path*. Paths are constructed using the \texttt{\path} command.

\begin{tikzpicture}
\path[draw] (0,0) -- (1,1);
\path[draw] (1,0) -- (2,0);
\end{tikzpicture}
Path extensions operations

Inside a `tikzpicture` environment *everything is drawn by starting a path and by extending the path*. Paths are constructed using the `\path` command.

```
\begin{tikzpicture}
\path[draw] (0,0) -- (1,1);
\path[draw] (1,0) -- (2,0);
\end{tikzpicture}
```
Inside a \texttt{tikzpicture} environment everything \textit{is drawn by starting a path and by extending the path}. Paths are constructed using the \texttt{path} command.

\begin{tikzpicture}
  \path[draw] (0,0) -- (1,1);
  \path[draw] (1,0) -- (2,0);
\end{tikzpicture}
Inside a `tikzpicture` environment, \textit{everything is drawn by starting a path and by extending the path.} Paths are constructed using the \texttt{path} command.

\begin{tikzpicture}
\path[draw] (0,0) -- (1,1);
\path[draw] (1,0) -- (2,0);
\end{tikzpicture}
Inside a `tikzpicture` environment *everything is drawn by starting a path and by extending the path*. Paths are constructed using the `\path` command.

\begin{tikzpicture}
  \path[draw] (0,0) -- (1,1);
  \path[draw] (1,0) -- (2,0);
\end{tikzpicture}
Line-to and move-to operations

Command `\draw` stands for `\path[draw]`:

\begin{verbatim}
\path[draw] (0,0) -- (1,1);
\path[draw] (1,0) -- (2,0);
\end{verbatim}

\begin{verbatim}
\draw (0,0) -- (1,1);
\draw (1,0) -- (2,0);
\end{verbatim}

Multiple paths can be traced with one single command:

\begin{verbatim}
\draw (0,0) -- (1,1)
  % move-to operation
  (1,0) -- (2,0);
\end{verbatim}
The *grid* and *cycle* operations

A *grid* is a path extension operation between two coordinates, much like a line (--):

\[
\begin{array}{l}
\text{\% fine, thin grid} \\
% \text{draw} [\text{line width}=0.1\text{pt}, \text{gray}!30, \text{step}=5\text{mm}] \\
(0,0) \text{ grid } (3,2); \\
\text{\% coarse, thicker grid} \\
% \text{draw} [\text{help lines}] \\
(0,0) \text{ grid } (3,2); \\
\text{\% a thick, closed path} \\
% \text{draw} [\text{thick}] (1,1) -- (2,2) -- (2,1) \\
\text{-- cycle;} \\
\end{array}
\]

The *cycle* operation closes a path connecting the current point with the initial point on the path.
More line-to and move-to operations

:\draw[help lines] (0,0) grid (3,2);
\draw[thick] (0,0) -- (1,1)
  \% then move-to
  (2,1) -- (3,2);

\draw[help lines] (0,0) grid (3,2);
\draw[thick] (0,0) -- (1,1) --
  (2,0) -- (3,2);

\draw[help lines] (0,0) grid (3,2);
\draw[thick] (0,0) -- (1,1)
  \% then move-to
  (2,0) -- (3,0) --
  (3,1) -- cycle;
More line-to operations

\texttt{\textbf{\textbackslash{draw}}[thick]} (0.0,0.0) \textendash{} (2.0,0.5) \\
\texttt{\textbackslash{draw}}[thick] (1.0,1.0) \textendash{} (3.0,0.0);
The \textit{curve-to} operation

% the supporting grid
\draw[help lines] (-2,-4) grid (2,4);
% define labels (nodes)
\path (-2, 0) coordinate (c1)
    (-1, 3) coordinate (c2)
    (0, -3) coordinate (c3)
    (2, -1) coordinate (c4);
% segments connecting nodes
\draw[dashed] (c1) -- (c2) -- (c3) -- (c4);
% control points
\draw (c1) circle (2pt)
    (c2) circle (2pt)
    (c3) circle (2pt)
    (c4) circle (2pt);
% the Bézier curve
\draw[thick] (c1) .. controls (c2) and (c3) .. (c4);
% text labels
\path
    (c1) node[anchor=west] {\texttt{c1}}
    (c2) node[anchor=west] {\texttt{c2}}
    (c3) node[anchor=east] {\texttt{c3}}
    (c4) node[anchor=east] {\texttt{c4}};
The *rectangle* and *circle* operations

\begin{verbatim}
\draw[thick] (0,0) rectangle (1,1) rectangle (3,2);
\draw[thick] (0,0) rectangle (1,1);
\draw[ultra thick,red] (1,1) rectangle (3,2);
\draw[thick] (0,0) circle (2pt) rectangle (3,1) circle (4pt);
\end{verbatim}
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (3,4);
  \draw[thick] (0,0) circle (2pt)
    -- (1,1) rectangle (2,3)
    -- (3,4)
    -- (2,4) circle (2pt);
\end{tikzpicture}
The *ellipse* operation

\begin{verbatim}
\draw[help lines] (0,0) grid (10,4);
\draw (2,2) ellipse (1cm and 1cm)
(3,2) ellipse (3cm and 2cm);
\draw[red] (8,2) ellipse (1cm and 1cm);
\end{verbatim}
The *node* operation

You can add text, math, and other material to paths with the *node path extension operation*.

The node operation

- places a given textual content at the current position;
- the current position becomes a node in the path;
- a label (variable name) can be associated to the node;
- named nodes can be used in further drawing operations;

Each node added to a path has an *outer shape*. The outer shape is only drawn if *draw* is part of the options. *The default node shape is a rectangle.*
The node operation

% in preamble: \usetikzlibrary{calc,positioning}
\path[draw] (0,1) node (p1) [\texttt{draw=none}] \{$P_1$};
\path[\texttt{draw,fill}] (1,0) \texttt{circle} (2pt) node (p2) [\texttt{anchor=north east}] \{$P_2$};
\path (2,1) \texttt{coordinate} (p3);
\path[\texttt{draw}] (p3) \texttt{circle} (10pt) node[\texttt{draw=none}] \{$P_3$};
\path (3,0) \texttt{coordinate} (p4);
\path[\texttt{draw=none}] (p5) \texttt{circle} (8pt) \{\texttt{box}\};
\node [\texttt{right=2.0cm of p5, anchor=center, inner sep=0pt, shape=circle, draw=red}] (p6) \{\texttt{box}\};
\node[\texttt{below right=1.0cm and 2.0cm of p6.center, anchor=center, inner sep=2pt, shape=circle, draw=red}] (p7) \{\texttt{box}\};
\node [\texttt{above right=1.0cm and 2.0cm of p7.center, anchor=center, inner sep=2pt, draw=red, fill=red, fill=white}] (p7) \{$\textbf{box}$};
Placing textual labels

\begin{tikzpicture}
\draw (0,0)
node (hello) [
  scale=2.0,
  inner sep=0pt,outer sep=0pt,
  draw=red]
{\fbox{\textbf{Hello \textsc{GuIT}}}};
\draw (hello.north east) circle (2pt) node[anchor=south west] {north east};
\draw (hello.north) circle (2pt) node[anchor=south] {north};
\draw (hello.north west) circle (2pt) node[anchor=south east] {north west};
\draw (hello.west) circle (2pt) node[anchor=east] {west};
\draw (hello.south west) circle (2pt) node[anchor=north east] {south west};
\draw (hello.south) circle (2pt) node[anchor=north] {south};
\draw (hello.south east) circle (2pt) node[anchor=north west] {south east};
\draw (hello.east) circle (2pt) node[anchor=west] {east};
\end{tikzpicture}
The arc operation

The arc operation adds an arc to the path.

- The arc starts at the current point, $P_1$. The user supplies two angles, $\alpha$ and $\beta$, and a radius $r$.

- The centre of the circle, $C$, is determined by the equation

\[
P_1 = C + (r \cos \alpha, r \sin \alpha)
\]

The end point of the arc is given by $P_2 = C + (r \cos \beta, r \sin \beta)$.

- The arc is drawn in counterclockwise direction from the start point to the end point, which becomes the new current coordinate of the path.
The *arc* operation

\begin{verbatim}
\draw[dashed] (4,0) coordinate (p0) arc (0:180:2cm); % (\alpha, \beta, r)
\draw[fill=black] (2,0) coordinate (c) % \leftarrow C
circle (1pt) node[anchor=south east] {$C$};
\path (p0) arc (0:30:2cm) % (\alpha, \beta, r), no arc drawn
cordinate (p30); % \leftarrow P_1
\draw[fill=black] (p30) circle (1pt) node[anchor=south west] {$P_1$};
\draw[thick] (p30) arc (30:120:2cm) % (\alpha, \beta, r)
cordinate (p120) % \leftarrow P_2
circle (2pt) node[anchor=north west] {$P_2$};
\draw[->,thick] (c) -- node[anchor=south east] {$r$} (p30); % \leftarrow \vec{r}
\end{verbatim}
More \textit{arc} operations

\begin{verbatim}
\draw[dashed] (1,1) circle (1cm);
\draw (1,2) coordinate (a) circle (2pt)
    (2,1) coordinate (b) circle (3pt)
    (1,0) coordinate (c) circle (4pt);
\draw[->,thick] (a) arc (90:180:1cm);
\draw[->,thick] (b) arc (0:45:1cm);
\draw[->,thick] (c) arc (270:225:1cm);
\end{verbatim}

\begin{verbatim}
\draw[dashed] (1.5,0) circle (1.5cm and 1cm);
\draw[fill=black] (1.5,0) coordinate (c)
    circle (1pt);
\draw (3,0) coordinate (a) circle (2pt);
\draw (0,0) coordinate (b) circle (2pt);
\draw[->,thick] (a) arc (0:90:1.5cm and 1cm);
\draw[->,thick] (b) arc (180:340:1.5cm and 1cm);
\end{verbatim}
Drawing with \texttt{tikz}

What else?

- More actions on paths, e.g. line widths, dash patterns, coloring, filling, shading.
- Predefined styles of graphic elements and their customizations.
- Available coordinate systems and advanced coordinate calculations.

Please have a look at the article on \texttt{Ars\TeX nica} for more details on \texttt{tikz}:

\textit{De Marco, A. “Graphics for \LaTeX users”}

\textit{Ars\TeX nica 28 (October 2019), pp. 64–100.}

All \texttt{tikz} examples given in the article are viewable on Overleaf:

https://www.overleaf.com/read/mgskyfdpttzt
Outline

General guidelines on illustration design

Drawing with LaTeX-aware software
   Using Inkscape + TexText extension

Drawing with natively available LaTeX environments/packages
   The standard environment picture
   The package pstricks (PostScript)
   The package tikz

Data plots with package pgfplots
Plotting data with \texttt{pgfplots}

The package \texttt{pgfplots} is built on top of \texttt{pgf} and is designed to draw graphs in a variety of formats, with a consistent and professional look and feel.

The package also allows to import data stored in files in tabular format via the package \texttt{pgfplotstable}.

As is usual with the \texttt{pgf} family, their manuals are impressive.

\url{https://www.ctan.org/pkg/pgfplots}
\url{http://texdoc.net/texmf-dist/doc/latex/pgfplots/pgfplots.pdf}
\url{http://texdoc.net/texmf-dist/doc/latex/pgfplots/pgfplotstable.pdf}
The **axis** environment

The workhorse of the *pgfplots* package is an environment called `axis`, which may **define one or several plots** (2D and 3D).

Each plot is drawn with the command `\addplot`.

The *axis* environment is used inside a *tikzpicture* environment.

Typically, one or more plots are created in *\LaTeX* as follows:

```latex
% in preamble
\usepackage{pgfplots}% loads tikz
...
\begin{tikzpicture}
  \begin{axis}[%graphic options]
    ...
  \end{axis}
\end{tikzpicture}
```

A. De Marco | Graphics for *\LaTeX* users | GIT meeting 2019
The axis environment

The simplest possible graph with pgfplots:

\begin{tikzpicture}
  \begin{axis}
    \end{axis}
  \end{tikzpicture}

An empty axis environment, with customized formatting options:

\begin{axis}[  
  xmin = -1, xmax = 1,  
  ymin = 0, ymax = 2,  
  grid = major,  
  xlabel = $x$, ylabel = $y$
]
\end{axis}
\begin{axis}[
    xmin = -1, xmax = 1,
    ymin = 0, ymax = 10,
    xtick = {-1,-0.5,...,1},
    ytick = {0,2,...,10},
    minor x tick num = 1,
    minor y tick num = 1,
    grid = major,
    xlabel = {$x$ (\si{meter})},
    ylabel = {\parbox{2cm}{\centering $\frac{\partial T}{\partial x}$ (\si{celsius/meter})}},
    title = {Temperature gradient},
    axis on top = true]
% a basic tikz drawing command
\fill[blue!40]
    (axis cs: -0.5, 0) -- (axis cs: 0.5, 0) --
    (axis cs: 0.5,10) -- (axis cs: -0.5,10) --
    cycle;
\end{axis}
The \addplot command

\begin{axis}[
  grid = major,
  xlabel = {$x$},
  ylabel = {$y$},
  y tick label style = {
    /pgf/number format/.cd,
    set thousands separator={}
  }
]
\addplot {-x^5 - 242};
\addlegendentry{model}
\addplot coordinates {
  (-4.77778, 2027.60977)
  (-3.55556, 347.84069)
  (-2.33333, 22.58953)
  (-1.11111, -493.50066)
  ( 0.11111, 46.66082)
  ( 1.33333, -205.56286)
  ( 2.55556, -341.40638)
  ( 3.77778, -1169.24780)
  ( 5.00000, -3269.56775)
};
\addlegendentry{estimate}
\end{axis}
The \texttt{\textbackslash addplot} command

Reading tabular data from file:

\begin{verbatim}
% in preamble
\usepackage{filecontents}
\begin{filecontents*}{data1.txt}
Level  Cost    Error
1    7   8.47178381e-02
2   31   3.04409349e-02
3  111   1.02214539e-02
4  351   3.30346265e-03
5 1023   1.03886535e-03
6 2815   3.19646457e-04
7 7423   9.65789766e-05
8 18943  2.87339125e-05
9 47103  8.43749881e-06
\end{filecontents*}

% ...
\begin{tikzpicture}
\begin{loglogaxis}[
xlabel={Cost},
ylabel={Error}]
\addplot[\color=red, \mark=x] coordinates {
(5, 8.3116034e-02)
(17, 2.54685628e-02)
(49, 7.40715288e-03)
% ...
(9217, 3.26101452e-06)
};
\addplot[\color=blue, \mark=*] \table[x=Cost, y=Error] {data1.txt};
\legend{Case 1, Case 2}
\end{loglogaxis}
\end{tikzpicture}
\end{verbatim}
Plotting data with \texttt{pgfplots}

\textbf{What else?}

\begin{itemize}
  \item Data column manipulation with \texttt{pgfplotstable}.
  \item Style customizations of graphic elements.
  \item Available coordinate systems and advanced coordinate calculations. 3D plots.
  \item Exporting \texttt{pgfplots} sources from other data plotting tools.
\end{itemize}

Please have a look at the article on \texttt{Ars\TeXnica} for more details on \texttt{pgfplots}:

\begin{itemize}
  \item \textit{De Marco, A., “Graphics for \LaTeX users”}.  \\
       \textit{Ars\TeXnica} 28 (October 2019), pp. 64–100.
  \item \textit{De Marco, A. and Giacomelli, R., “Creare grafici con pgfplots”}.  \\
       \textit{Ars\TeXnica} 13 (October 2011), pp. 9–35.
\end{itemize}
Conclusions

We have seen the most common scenarios encountered by `\LaTeX` users when they face the problem of producing quality graphics.

In cases of diagrams, pictures and more or less complicated illustrations the two approaches based on package `tikz` and on the `Inkscape` graphics vector software have been presented.

Examples of scientific plots with the package `pgfplots`.
Thank you …

Questions?

http://texample.net/tikz/examples/dartboard/