AcroT_EX.Net

The **annot_pro** Package

Text, Stamp, File Attachment, and Text Box Annotations

D. P. Story

Table of Contents

1. Introduction

This package is used to create text, stamp, and file attachment annotations using **Adobe Distiller**, these annotations can be viewed in Adobe Reader. For users of pdf(la)tex, use the pdfcomment package by Josef Kleber.[1]

The package primarily in support of my AcroTeX PDF Blog. I plan to use sticky notes, file attachments, and custom stamps to make side-comments, and to provide source files.

2. Requirements

The requirements for your LaTeX system, and well as any other software, is highlighted in this section.

2.1. LaTeX Package Requirements

The following packages, in addition to the standard LaTeX distribution, are required:

1. The xkeyval package is used to set up the key-value pairs of the \annotpro command. Get a recent version.

2. The xcolor package is strongly recommended.

3. The hyperref package, a recent version.

4. If you want to create custom stamps (on the fly) using the techniques developed for that purpose, the graphicxsp Package is required.[2]

The annot_pro package is part of the AeB Pro family, which means **Adobe Distiller** is required. The components of AeB and AeB Pro are not required.[3,4]

---

[1] Available at ctan.org/pkg/pdfcomment
[2] Available at ctan.org/pkg/graphicxsp
[3] AeB: ctan.org/pkg/acrotex
[4] AeB Pro: ctan.org/pkg/aeb-pro

2.2. PDF Creator Requirements

The `annot_pro` package requires **Acrobat Distiller 5.0** (or later) as the PDF creator. The document author typically uses dvips (or dvipsone) to produce a PostScript file, which is then distilled to obtain a PDF.

If you wish to use (dynamically) created stamps that have opacity less than 1, you need to distill using **Standard_transparancy.joboptions** with distiller, in this case, **Acrobat Distiller 6.0** (or later) is required; otherwise, this distiller job options file is not needed.

The **Standard_transparancy.joboptions** file is supplied with the `graphicxsp` package; the documentation of the `graphicxsp` package includes installation instructions.

3. Installation

The installation is simple enough. Unzip `annot_pro.zip` in a folder that is on your LaTeX search path. Refresh your filename database, if appropriate.

Important: When creating a file attachment annotation, you must specify a path to the file to be attached, and distiller must embed this file. In recent versions of Acrobat, security restrictions have been put in place to prevent **Distiller** from reading files (the PostScript **file** operator does not work). Fortunately, Distiller has a switch that turns off this particular restriction. To successfully use this package, therefore, you need to run Distiller by using the `-F` command line switch.

Those using `Windows OS` can create a shortcut on the desktop, for example, that starts `Distiller` with the `-F` switch. The `Target` of the shortcut might read

```
"C:\Program Files (x86)\Adobe\
    Acrobat 9.0\Acrobat\acrodist.exe" -F
```

where we have wrapped the path to the next line to fit within the margins. Once `Distiller` is started with `-F`, the switch remains in effect until `Distiller` is closed.

If this package is used to create file attachment annotations without the `-F` switch, you typically get the following error message in the Distiller log file

```
%%[ Error: undefinedfilename; OffendingCommand: file ]%%
```

This tells you that either you have not started Distiller with the -F command line switch, or Distiller can't find one of the files that the **file** operator was trying to read.

Mac OS Users. The above comments on the -F command line switch is for Windows users, Mac OS users must choose the AllowPSFileOps user preference, this is located in the plist, possibly located at

/Users/[User]/Library/Preferences/com.adobe.distiller9.plist

You can also use Spotlight, the search utility on Mac, to search for the string com.adobe.distiller; the result might be

com.adobe.distiller9.plist.[5]

Clicking on this find, Spotlight opens com.adobe.distiller9.plist in the plist editor, see Figure 1. If necessary, click on the arrow next to the Root to expand the choices, then click the up and down arrows at the far right in the AllowPSFileOps row to select Yes as the value.
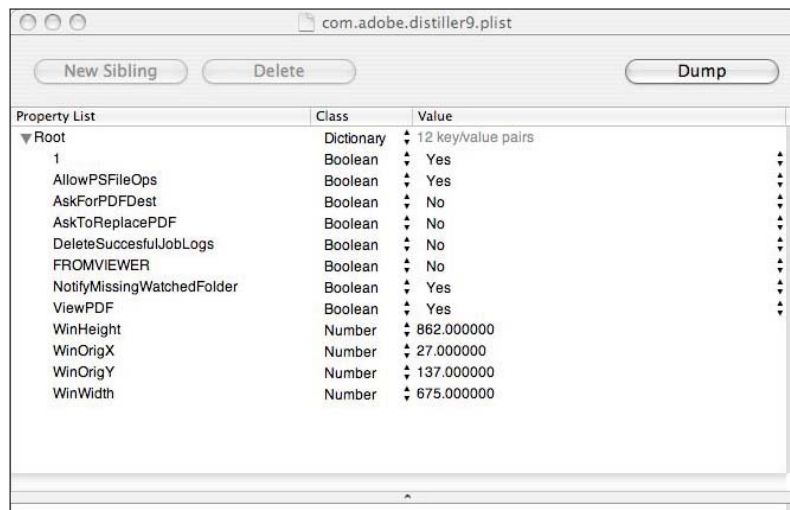

Figure 1: com.adobe.distiller9.plist

---

[5]In the case of Adobe Distiller, version 9.0

4. The \annotpro Command

The main command of this package is \annotpro; the command is controlled by its optional parameters. The same command can create a text annotation (sticky note), a stamp annotation, a file attachment or a Free Text (Text Box) annotation. The syntax of this command is

$$\boxed{\texttt{\textbackslash annotpro[}\langle \textit{KV-pairs}\rangle\texttt{]\{}\langle \textit{text}\rangle\texttt{\}}} \tag{1}$$

Parameter Description: The optional argument is one or more key-value pairs that describe the annotation; the required argument, $\langle \textit{text}\rangle$, is the "contents" of the annotation; for the text and stamp annotation, $\langle \textit{text}\rangle$ becomes the contents of the popup annotation, for a file attachment annotation, which has not associated pop-up, it is the description of the file attachment that appears in the file attachment panel of Adobe Reader. In the latter case, the value of $\langle \textit{text}\rangle$ should be rather short. The key-value pairs ($\langle \textit{KV-pairs}\rangle$) are described over the next few sections.

Sample files. The sample files annots.tex and textbox.tex illustrate the features of the annotpro package.

4.1. Key-values common to all annotations

The following are key-value pairs common to all annotations.

- type=⟨text|stamp|fileattachment|textbox⟩ The type key determines the type of annotation to be produced; permissible values are text, stamp, and fileattachment. This key is optional, if not present, type=text is assumed.

- name=⟨*name*⟩ The name of the icon to use for the declared type. Permissible values are dependent on the type, and are discussed in subsequent sections.

- title=⟨*text*⟩ Text to be displayed in the title bar of the annotation's pop-up window when open and active. By convention, this entry identifies the user who made the annotation, though any (short) text may be used. You can use \setAnnotOptions to globally set the title of each annotation, perhaps using your name.

      \setAnnotOptions{title={D. P. Story}}

- subject=⟨*text*⟩ Text representing a short description of the subject of the annotation. You can use \setAnnotOptions to globally set the subject of each annotation,

  ```
  \setAnnotOptions{title={D. P. Story},
      subject={AcroTeX Communiqu\'e}}
  ```

- color: The color of the title bar of the pop-up window of the annotation.

- readonly=⟨true|false⟩ Set the annotation to readonly. The user can click on the annot to see the popup, but the user, if using Acrobat, cannot move the annotation around on the page. The popup window can still be move by the user. (This property makes no difference to the user of Adobe Reader.)

- opacity=⟨*dec*⟩: The opacity value ($0 \leq \langle dec \rangle \leq 1$) to be used in painting the (icon of the) annotation, but does apply to the pop-up window. The default is 1.0.

  Adobe Distiller handles the opacity for us in all cases except when we create a (dynamic) stamp. If an opacity value less than 1 is desired, special techniques are needed, and the file needs to be distilled using the **Standard_tranparency** job options.

The following key-values are LaTeX based concerning placement of the annotation in the margin.

- margin: Use this key (it has no value), to declare that you want the annotation to appear in the margin. The \marginpar command from core LaTeX is used, the placement of the annotation follows the rules set down by LaTeX. You can reverse the placement of the annotation by using the LaTeX command \reversemarginpar (annots placed in the right margin, and now placed in the left); you can return to the default by using \normalmarginpar.

  Given that you have use the margin key, there is an associated key that can be used, as well as a command.

  - margintext: The value of this key is text that will be typeset just below the annotation icon.

– \marginpartextformat: A LaTeX command for formatting the text in the margin, the default definition is

\margintextformat{\bfseries\tiny\color{blue}}

For an annotation placed in the margin with margin text, you might want to use the readonly key, this prevents the user—even one using Acrobat—from moving the annotation away from its caption.

The following key-value is for your convenience.

- presets: A key to allow the introduction of pre-defined options, for example, you might like all your comment annotations to be red, so you can define

\def\myComments{type=text,name=Comment,color=red}

then say

\annotpro[presets=\myComment]{Way to go!}

Additional options may be included,

\annotpro[presets=\myComment,margin]{Way to go!}

for example.

4.2. Key-values for text annotations

The position of the annotation is determined by its bounding rectangle; for the text annotation, an icon is placed so that its upper left corner is at the upper left corner of the bounding rectangle. The icons themselves have certain dimensions that have been recorded within the annot_pro package, so you need not worry about leaving space for them.

An important fact about the icons used by text annotation is that they *do not zoom in or out* as the page magnification is changed; furthermore, the graphics commands \scalebox and \resizebox do not rescale the icons as expected.

The following are options specific to the text annotation. Recall, the text annotation is of type=text.

- `name`: The name of the icon to use when displaying this annotation in closed form (no pop-up window visible). Possible values—as specified in the *PDF Reference*—are `Comment`, `Key`, `Note`, `Help`, `NewParagraph`, `Paragraph`, and `Insert`. Additional icons that are available in recent versions of Adobe Reader are

    > `Check`, `Circle`, `Cross`, `Star`, `RightArrow`, `RightPointer`, `UpArrow`, `UpLeftArrow`

    If you are using comments in your document, and your audience have older versions of Adobe Reader, it is best to use only the seven listed in the *PDF Reference*.

- `open`: A Boolean value that determines whether the pop-up window is open or not. When `true` the pop-up is open. The package default is `false`. You can use `\setAnnotOptions` to set this option globally.

- `nohspace`, `novspace`, `nospace`: The presence of these commands zeroes out the dimension(s) of the bounding rectangle. Specifying `nohspace` as an option causes the icon to take up no horizontal space as the page is latexed. Here ▤ an example of a sticky note with the `nohspace` option. Without any of these three keys, the text annotation ⚷ fits exactly into the allotted space at 100% magnification, try it.

    When the icon takes up no TeX space, it may cover content on the page, as it does above. Acrobat users can move the icon around, but AR users cannot move the icon. The pop-up window is movable and scalable, but the icon cannot be moved. Therefore, you must be careful about placement.

    Additional positioning of the icons can be made using standard LaTeX commands such as `\raisebox` and `\smash`. For example, the blue icon above was created by

    ```
    For example,\smash{\raisebox{1in}\raisebox{1in}{%
        \annotpro[nospace,color=blue,
            opacity=.25]{...clever message...}}}
    ```

    Note that I've set the `opacity` to .25.

4.3. Key-values for stamp annotations

A stamp annotation is similar to a text annotation in the sense that it has a pop-up window in which the contents of the message is displayed; however, unlike the text annotation icons, the stamp appearance by be re-scaled using \scalebox and \resizebox of the graphics package; however, the keys scale, widthTo, and heightTo, as described below, are the recommended methods of re-scaling a stamp.

The following are the key-values associated with this annotation type.

- name: The stamps listed in the *PDF Reference* are Approved, AsIs, Confidential, Departmental, Draft, Experimental, Expired, Final, ForComment, ForPublicRelease, NotApproved, NotForPublicRelease, Sold, and TopSecret. If name is not specified, Draft is the default.

  There are other stamps, not listed in the *PDF Reference*, but available in more recent versions of Acrobat. The file stamps.pdf lists all the stamps that I have access to. The names of these other stamps are recognized by the name key.

  The dimensions of the stamps listed above are all the same, they are 150bp width and 40bp high.

  Additional stamps are shipped by Acrobat, a listing of these appears in the file stamps.pdf (stamps.tex).

- width, height: If the value of name key is something other than one of the stamps listed above (one of the stamps listed in the file stamps.pdf), the width and height are not known to annot_pro. In this case, use the height and width keys to set the dimensions of the bounding box. Adobe Distiller will resize the stamp to the stamp is the largest one that can fit in the bounding box, the stamp will be centered vertically and horizontally within the bounding box.

  Here are a couple of examples, the bounding box is should as a black \fbox.

  - This one fits more or less exactly. I determined the dimensions of this stamp through some controls of

the user interface. Contrast this this stamp APPROVED obtained

by setting `width=50bp` and `height=50bp`. Notice the "best fit," and that the bounding box takes up space. We can use `\smash` to smash its vertical height, let's see how that looks, APPROVED

The code for producing these stamps are given in the margins.

You can resize these stamps using `\scalebox` and `\resizebox`, like so. APPROVED   The code for producing this stamp are given here ▤ .

– To create a stamp that ~~make a PDF~~ ~~note, at least, use~~
  `\smash` and `\make`~~box[0pt][l]{text}~~, ~~here is one of the stan~~
  dard stamps as ~~listed in the PDF Reference. The code for this~~
  stamp is given in this note ▤ .

APPROVED

Good Fit

Bad Fit

Try changing the magnification of the page, you'll see that stamps are re-scaled as you zoom in or out, while the text annotations are not. I don't like text annotations for this reason.

- `rotate`: Stamps can be rotated, use this key to enter an angle of rotation, for example, `rotate=30` rotates the stamp 30° counter clockwise.

☞ Do not use the `\rotatebox` command of `graphics` package, this command *does not rotate* the stamp.

- `scale`: Use this key to re-scale the stamp; the value of this key is a number between 0 and 1. For example, `scale=.5` reduces both width and height in half.

- `widthTo`: This key resizes the stamp so that the width is the value of this key; for example, `widthTo=2in` re-scales the stamp to have a 2in width.

- `heightTo`: This key resizes the stamp so that the height is the value of this key; for example, `heightTo=2in` re-scales the stamp to have a 2in height.

Only one of the keys scale, widthTo, and heightTo are recognized for any stamp annotation. If all three are specified, only scale is used. If widthTo and heightTo are both specified, then widthTo is used.

☞ The use of these keys is the recommend way of re-scaling a stamp. These methods are compatible with the rotate key.

- customStamp: You can create a custom stamp from any image you wish to use, here is [image] one such example. The code for the above stamp is given here [image].

- ap: When the customStamp key is used, you have to supply an indirect reference to the appearance of the stamp. This reference is made through the ap key. The example above demonstrates the use of the ap key.

Important: When using the stamps of Acrobat always perform a SaveAs on the file when you have finished building the file. This imports the appearances of the stamps into the document and saves them.

▶ The creation of a custom stamp requires detailed list of steps, at some point I'll write a white paper on the subject.[6]

Here's an example of the keys rotate and widthTo:

4.4. Key-values file attachment annotations

The file attachment annotation has no pop-up window, the value of the required parameter is used as a description of the attached file, and appears in the file attachment window.

---

[6]Use the mkstmp package (ctan.org/pkg/mkstmp), details of how to create custom stamps are included in the documentation.

The key-value pairs special to this type of annotation are as follows.

- `name`: The name of the icon to use, permitted values are `Graph`, `Paperclip`, `PushPin`, and `Tag`. If the value of name is not specified, the default is `PushPin`. The annot_pro package knows the dimensions of each of these icons, so you need not worry about them. They can be re-scale using standard commands from the graphics package, though, there may be little need of doing so.

- `file`: The value of the file key is the *absolute path* to the file to be attached. I've devised a helper command `\defineAPath` that can be used to define the path to your file. For example, we can define a path to wherever the files are, like so

```
\defineAPath{\graphicsPath}
  {C:/Users/Public/Documents/My TeX Files/%
      tex/latex/aeb/aebpro/annot_pro/examples/graphics}
```

The command takes two arguments, the name of the command to be defined, and the path.

We can create a file attachment 📎 like so, ho, ho. Here is the code for this file attachment ⭐. Clicking the file attachment icon will open the file, in recent versions of Acrobat, the file is listed in the file attachments panel. Open it using the user interface, and you'll see the file listed, as well as a description, as passed to the annot as the second argument of `\annotpro`.

The file attachment icons can be resized using any of the graphics commands, `\scalebox` or `\resizebox`.

4.5. The Text Box (Free Text) annotation

A `Text Box` annotation is a rectangular region in which the user can enter rich text. The annotation may be created and 'pre-populated' with rich text from a LaTeX source through the annot_pro package.

The `Text Box` annotation, as implemented by this package, requires the `richtex` package, dated 2016/09/30 or later. A one simple method for introducing `richtext` is through the `useTextBox` option of annot_pro. Placing the following line in the preamble:

```
\usepackage[useTextBox]{annot_pro}
```

declares you are going to use Text Box annotations. The option does nothing more than to execute \usepackage{richtext}[2016/09/30]. The option is more of a convenience.

To create a Text Box (originally named Free Text) annotation use the \annotpro command.

$$\annotpro[\textbf{type=textbox},\langle KV\text{--}pairs\rangle]$$
$$\{\texttt{richtext}=\langle name\rangle,\texttt{defstyle}=\langle name\rangle\} \tag{2}$$

Notice the second argument is not ⟨*text*⟩ as it is with the other annotations, but consists of key-value pairs; recognized keys are richtext and defstyle.

The sample file textbox.tex illustrates the features of this section.

• Creating an empty Text Box

A common application would be to create an empty Text Box for the document consumer to type in comments. Below is an example of a Free Text annotation, called Text Box by the user-interface of Acrobat/Adobe Reader.
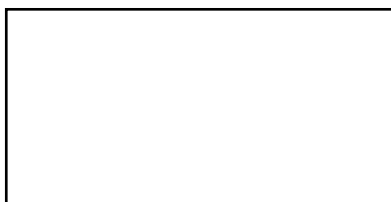
Press Ctrl+E (Windows) or Cmd+E (Mac OS) to obtain the Properties toolbar, now click on the text box to obtain the Text Box Properties toolbar. Double clicking on the text box brings up the Text Box Text Properties toolbar.

The verbatim listing of the above Text Box is,

```
\annotpro[type=textbox,
    title=dpstory,subject=Empty Text Box]{}
```

The required second argument is empty, which leads to an empty Text Box.

• Creating a non-empty Text Box

A much more interesting exercise is to pre-populate the Text Box with rich text for the document consumer to read and/or to respond.

¶ **Steps to create rich text content.** We briefly outline the techniques to create rich text for a Text Box annotation.

- Use the textboxpara environment and the \rtpara command to declare your rich text paragraph.

$$
\boxed{\begin{array}{l} \text{\textbackslash begin\{textboxpara\}} \\ \text{\textbackslash rtpara[}\langle KV\text{-}pairs\rangle\text{]\{}\langle name_{para}\rangle\text{\}\{}\langle rich\text{-}content\rangle\text{\}} \\ \text{...} \\ \text{\textbackslash end\{textboxpara\}} \end{array}} \tag{3}
$$

  Details of the \rtpara command are found in the documentation manual of the richtext package. The textboxpara environment is needed for certain redefinitions of internals because of the different way rich text is supported and implemented in the Text Box annotation verses the rich text field.

- Use the \setRVVContent command on your \rtpara-declared rich text and give it a name $\langle name_{rvvc}\rangle$.

$$
\boxed{\begin{array}{l} \text{\textbackslash setRVVContent\{}\langle name_{rvvc}\rangle\text{\}\{}\langle name_{para}\rangle\text{\}} \\ \text{...} \end{array}} \tag{4}
$$

  This command expands the paragraph named $\langle name_{para}\rangle$ and develops the rich text version and the plain text version. It is $\langle name_{rvvc}\rangle$ that is used as a value for the richtext key above and illustrated below. To reduce the number of names, you can use the same name for $\langle name_{rvvc}\rangle$ as for $\langle name_{para}\rangle$ (\setRVVContent{para1}{para1}).

  The rich text *form field* supports multiple paragraphs, and richer formatting options than the Text Box annotation. Of importance is that the Text Box annotation only permits a *single paragraph*.

- (Optional) Declare a (named) default style using \setDefaultStyle:

$$
\boxed{\begin{array}{l} \text{\textbackslash setDefaultStyle\{}\langle name_{ds}\rangle\text{\}\{}\langle KV\text{-}pairs\rangle\text{\}} \\ \text{...} \end{array}} \tag{5}
$$

  This default style is assignment a name that is used as the value of the defstyle key mentioned earlier, and illustrated below. If a value for defstyle is not provided, a standard default style is used.

Once the \rtpara-declared paragraphs have been made and their names have passed through \setRVVContent, you are ready to create a Text Box annotation.

¶ Key-values for second argument. The required second argument, refer to display (2), has two keys, both of which are optional. You are encouraged to read the documentation for the richtext package for greater understanding of the descriptions and examples found below.

richtext=⟨$name_{rvvc}$⟩ The richtext key is the way the rich text is passed to the Text Box. The ⟨$name_{rvvc}$⟩ is declared by the \setRVVContent command. Use the command \rtpara within the textboxpara environment to define the actual rich text paragraph. For example,

```
\begin{textboxpara}
\rtpara{para1}{\span{color=FF0000}{Hello world},
    this is \bf{rich text}}
\end{textboxpara}
\setRVVContent{myContent}{para1}
\annotpro[type=textbox,title=dpstory,
    subject=Text Box]{richtext=myContent}
```

The above code produces the following Text Box:

> Hello world, this is **rich text**!
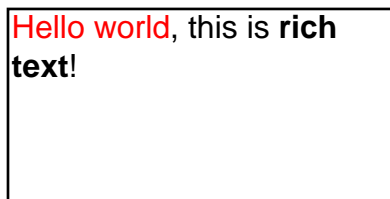
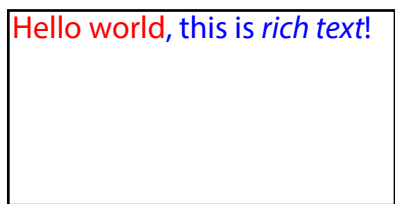defstyle=⟨$name_{ds}$⟩ Through the defstyle you can define set the default style (refer to the documentation of the richtext package. If this key does not appear, then a predefined default style is provided.

```
\setDefaultStyle{myDS}{font={'Myriad Pro',sans-serif},
    color=0000FF}
\begin{textboxpara}
\rtpara{para1}{\span{color=FF0000}{Hello world},
    this is \it{rich text}}
\setRVVContent{para1}{para1}
\end{textboxpara}
```

```
\annotpro[type=textbox,title=dpstory,
    subject=Text Box]{richtext=para1,defstyle=myDS}
```

Notice that we've used the name 'para1' for both \setRVVContent and \rtpara. The above code produces the following Text Box:

Hello world, this is *rich text*!

Press Ctrl+E (Windows) or Cmd+E (Mac OS) to obtain the Properties toolbar, now double click on the text box to obtain the Text Box Text Properties toolbar to verify the font used is indeed Myriad Pro.

If you are at all interested in generating the Text Box annotation using rich text strings, you are encouraged to read the documentation on richtext, there you will learn about all the key-values available to format the text and the paragraph.

The richtext package was written for rich text form fields, but applies to rich text annotations as well; however, it should be noted that there are differences between forms and text box annotations in how they handle rich text. One of the major differences is that rich text annotations (Text Box) *do not support* multiple paragraphs as form fields do; as a result, features listed in the Paragraph and Link tabs of the Form Field Text Properties dialog box are not available for the Text Box.

¶ **Keys & commands inherited from the richtext package.** The following keys are supported by the Text Box annotation:

$$\text{font, size, } \sout{\text{raise}}, \text{ ulstyle, color, } \sout{\text{url}}, \qquad (6)$$
$$\text{style, } \sout{\text{raw}}, \text{ halign}$$

The following commands are supported:

$$\text{\span, \br, \it, \bf, \sup and \sub} \qquad (7)$$

The ones having an overstrike are supported in a rich text *form field*, but not within an Text Box. Refer to the documentation of the richtext for details of these keys and commands. In this document, we illustrate by example.

¶ **Key-values of \rtpara.** The keys of display (6) – excluding the overstrike ones – may be used in the ⟨*KV–pairs*⟩ argument of \rtpara of display (3).

¶ Permissible commands within ⟨*rich-content*⟩. The ⟨*rich-content*⟩ argument of display (3) (normally) consists of Latin 1 characters, plus any markup in the form of the commands listed in display (7).

- \span has two arguments, more on this command in the paragraph below titled 'Some comments on the \span command' on page 18.

- \br is a line break, it has no argument.

- \it is italic font; it has one argument, the text to be placed in italics: \it{this is italic}.

- \bf is bold font; it has one argument, the text to be placed in bold: \bf{this is bold}.

  \it and \bf may be nested: \it{\bf{bold and italic}}.

- \sup and \sub are superscript and subscript, respectively; they each have one argument, the text to be raised or lowered. For example, We can \sup{raise} or we can \sub{lower} text.

The above markups, with the exception of \span are illustrated below.

This is italic, whereas **this is bold**, but wait, we can do ***bold and italic***.

Moving on, we can ^raise^ or we can ~lower~ text.

The verbatim listing for this example is in the sticky note in the margin.

¶ Some comments on the \span command. the \span command, defined only locally within the ⟨*rich-text*⟩ argument \rtspan is a general purpose command to format text. It has two argument:

> \span{⟨*KV-pairs*⟩}{⟨*text*⟩}.

The ⟨*KV-pairs*⟩ argument can be the keys of display (6) (again excluding the overstrike keys). The ⟨*text*⟩ argument is the text to be made rich; experience shows that \it, \bf, \sub and \sup may be used within ⟨*text*⟩. Italic and bold may be accomplished using the style key, probably preferred over using \it and \bf within ⟨*text*⟩.

> Welcome to my **poorrich text world**. We add a little color shall we try red or green?
>
> There are several styles of underlining basic underlining, double underlining, word underlining, and double word underlining. Cool.

The Code

The verbatim listing for this example is in the sticky note in the margin.

¶ **Key-values of \setDefaultStyle.** The keys of display (6) – excluding the overstrike ones – may be used in the ⟨*KV-pairs*⟩ argument of \setDefaultStyle of display (5); however, only the keys font, size, color, and halign are typically used. For example,

```
\setDefaultStyle{myDS}{font={'Myriad Pro',sans-serif},
    size=12.0,color=0000FF,halign=left}
```

The name 'myDS' may then be used as the value of defstyle key in the second argument of \annotpro, see display (2).

• **Key-values for text box annotations**

Section 4.1 lists keys that are common to all annotations; we list the ones that make sense for the Text Box annotation, and strikeout these that do not:

> type, ~~name~~, title, subject, ~~color~~, readonly, opacity, margin, presets

In addition to these keys, there are several keys particular to the text box annotation. We list these and describe them in detail.

- width=⟨*length*⟩: The width of the annotation, the default is 144bp (2in).

- height=⟨*length*⟩: The height of the annotation, the default is 72bp (1in).

- bgcolor=⟨*color*⟩ The color to be used as the background color of the text box annotation. If bgcolor has no value, transparent color is used. The default is white.

- bcolor=⟨*color*⟩ The color to be used as the boundary color of the annotation. The default is black.

- borderstyle=⟨*choice*⟩: This keys determines the style of border to be used. It is a choice key, choices are none, solid, dash1, dash2, dash3, dash4, dash5, dash6, cloud1, and cloud2. The default is solid.

- borderwidth=⟨*choice*⟩: The border width of the annotation, acceptable choices are .5, 1, 2, 3, 4, 6, 8, and 10. The default is 1.

```
\annotpro[type=textbox, width=\linewidth,
height=14bp*2, bgcolor=cornsilk, bcolor=blue]
```

```
\annotpro[type=textbox, width=\linewidth,
height=16bp*3, bgcolor, bcolor=red, borderstyle=dash2,
borderwidth=2]{richtext=para2}
```

```
\annotpro[type=textbox, width=\linewidth,
height=18bp*3, bcolor=red, borderstyle=cloud1]
{richtext=para3}
```

The second text box annotation above has transparent background color. Using your pointing device, move it around to verify the background is 'see through', compare with the other two by moving them around, not 'see through'.

- **Accented glyphs and the unicode characters**

There are several advantages that text box annotation have over rich text form fields: movability and unicode. An annotation can be moved around the page by the user quite easily, a form field typically cannot unless the user is using Acrobat. Also, when it comes to using unicode, text box annotations are far superior to rich text form fields. Unicode characters may be inserted using the convenience commands \uHex and \uDec, where the first take a hex code as its argument and the second takes a non-negative integer as its argument. Latin1 accented characters can be entered using octal notation for example, J\374rgen

Jürgen is a nice fellow, though I've never met him.

We've communicated, Jürgen and I, via email. Jürgen where are you?

The Code

But wait, we're not done. In theory, we can access any unicode character through the use of \uHex or \uDec. I'll randomly pick off some unicode characters.

Ɖ, Ⱦ, Σ, Ж, ψ, ⋉ ,☺ ,ق,   ,   ,   ,

The Code

Cool!

4.6. Setting Global Options with \setAnnotOptions

Global options are by using the \setAnnotOptions command. In the preamble of this document I placed

`\setAnnotOptions{subject={AcroTeX Communiqu\'e},title={D.P. Story}}`

That way, I didn't have to constantly type in my personal name for each example. These options can be overwritten by specifying options locally, if I say, \annotpro[author=Don Story]{Hi there!}, the author is now my alter ego, Don Story.

That's all for now, I simply must get back to my retirement. DS