**Reproducible Reports**

Remember to read the book chapter on Slidify and RStudio Presenter.

We don’t cover making reproducible reports, as Roger Peng has an entire book devoted to the subject as part of the specialization. For the same reason we want reproducible reports, we also want reproducible presentations. In fact, in many ways, reproducible presentations should be more common as presentations are the default communication method for many projects in progress. Reports, often come later. Fortunately, the same framework accomodates both in R. Specifically, R markdown (knitr) documents. To make a presentation in R markdown, one writes an RPres file in a simplified markup language (called markdown) that is trivial to learn. One embeds the R code and data analysis in this document, and then the presentation is compiled using the R command prompt. The end result is a visually pleasing html5 presentation. Since the presentation is html5, it just runs in a browser, so no more worrying about different PowerPoint versions! The act of embedding the code in the document helps one organize their thoughts around the presentation. Moreover, it makes all of the numbers and figures perfectly reproducible. Finally, it’s incredibly useful for yourself when you come back to the presentation after a while and want to figure out what you did. There are two main platforms for reproducible presentations in R, slidify and RStudio’s presenter. We’ll cover both.

# 01 : slidify

**What is Slidify?**

* Slidify was created by [Ramnath Vaidyanathan](https://github.com/ramnathv) in order to streamline the process of creating and publishing R driven presentations.
* Slidify is an amalgamation of other technologies including knitr, Markdown, and several javascript libaries for HTML5 presentations.
* Slidify is infinitely extendable and customizable, yet it is easy to use!
* Slidify allows embedded code chunks and mathematical formulas which keeps your presentation reproducable.
* Slidify presentations are just HTML files, so you can view them with any web browser and share them easily on Github, Dropbox, or your own website.

**Getting Slidify**

* Fire up RStudio!
* First, make sure you have devtools installed and loaded

install.packages("devtools")

library(devtools)

* Second, install Slidify

install\_github('slidify', 'ramnathv')

install\_github('slidifyLibraries', 'ramnathv')

* Third, load Slidify

library(slidify)

**Getting started with Slidify**

* Set the working directory to where you want to create your Slidify project

setwd("~/sample/project/")

* Create your project and give your project a name (My project is named "first\_deck")

author("first\_deck")

**Getting started with Slidify**

* author("first\_deck") causes the following to happen:

---

title :

subtitle :

author :

job :

framework : io2012 # {io2012, html5slides, shower, dzslides, ...}

highlighter : highlight.js # {highlight.js, prettify, highlight}

hitheme : tomorrow #

widgets : [] # {mathjax, quiz, bootstrap}

mode : selfcontained # {standalone, draft}

knit : slidify::knit2slides

---

## Read-And-Delete

1. Edit YAML front matter

2. Write using R Markdown

3. Use an empty line followed by three dashes to separate slides!

--- .class #id

## Slide 2

1. A directory with the name of your project is created inside of your current directory.
2. Inside of this directory an assets directory and a file called "index.Rmd" is created.
3. The assets directory is populated with the following empty folders: css, img, js, and layouts.
4. The newly created index.Rmd R Markdown file will open up in RStudio.

* Any custom css, images, or javascript you want to use should respecively be put into the newly created css, img, and js folders.

**Getting to know index.Rmd : YAML**

* index.Rmd is the R Markdown document which you will use to compose the conent of your presentation.
* The first part of an index.Rmd file is a bit of YAML code which will look like this:

---

title :

subtitle :

author :

job :

framework : io2012 # {io2012, html5slides, shower, dzslides, ...}

highlighter : highlight.js # {highlight.js, prettify, highlight}

hitheme : tomorrow #

widgets : [] # {mathjax, quiz, bootstrap}

mode : selfcontained # {standalone, draft}

---

**Getting to know index.Rmd : YAML**

* You can edit your YAML to include the title, subtitle, author, and job of the author, including what slide framework you wish to use, which code highlighter you wish to use, and any widgets you want to include.
* Other fields you can include in your YAML: a logo to appear in your title slide under logo, the path to your assets folder and the paths to any other folders you may be using under url, and the specific theme for your code highlighter of choice under hitheme.

logo : my\_logo.png

url :

assets: ../assets

highlighter : highlight.js # {highlight.js, prettify, highlight}

hitheme : zenburn #

* Remember that ../ signifies the parent directory.

**Getting to know index.Rmd : YAML**

* The YAML for the presentation you're currently viewing looks like this:

---

title : Slidify

subtitle : Data meets presentation

author : Jeffrey Leek, Assistant Professor of Biostatistics

job : Johns Hopkins Bloomberg School of Public Health

logo : bloomberg\_shield.png

framework : io2012 # {io2012, html5slides, shower, dzslides, ...}

highlighter : highlight.js # {highlight.js, prettify, highlight}

hitheme : tomorrow #

url:

lib: ../../libraries

assets: ../../assets

widgets : [mathjax] # {mathjax, quiz, bootstrap}

mode : selfcontained # {standalone, draft}

---

**Getting to know index.Rmd : Making Slides**

Your first two slides are made for you under the YAML:

**## Read-And-Delete**

1. Edit YAML front matter

2. Write using R Markdown

3. Use an empty line followed by three dashes to separate slides!

- - - .class #id

**## Slide 2**

* Whatever you put after ## will be the title of the slide.
* --- marks the end of the slide.
* .class #id are CSS attributes you can use to customize the slide.
* Whatever you put between ## and --- is up to you! As long as it is valid R Markdown or HTML.

**Getting to know index.Rmd : Making Slides**

* To compile your presentation make sure the working directory contains your index.Rmd file and enter the following command:

slidify("index.Rmd")

* An HTML flie should appear in your current directory, open it with your favorite web browser and enjoy your Slidify deck!

browseURL("index.html")

**Publishing to Github**

* First, log in to GitHub and [create](https://help.github.com/articles/creating-a-new-repository) a new empty repository.
* Use the following command, but replace user with your username and repo with the name of your new repository (both arguments are strings).

publish\_github(user, repo)

**HTML5 Deck Frameworks**

The following frameworks are compatible with Slidify for making your presentations:

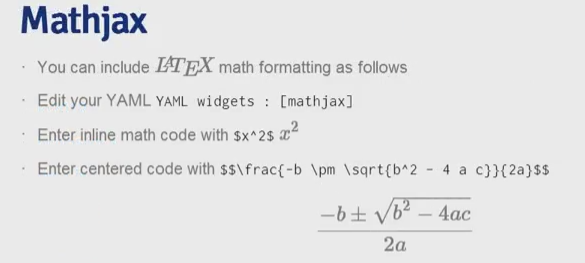
* [io2012](https://code.google.com/p/io-2012-slides/)
* [html5slides](https://code.google.com/p/html5slides/)
* [deck.js](http://imakewebthings.com/deck.js/)
* [dzslides](http://paulrouget.com/dzslides/)
* [landslide](https://github.com/adamzap/landslide)
* [Slidy](http://www.w3.org/Talks/Tools/Slidy2/Overview.html)

**Mathjax**

* You can include $\LaTeX$ math formatting as follows
* Edit your YAML

widgets : [mathjax]

* Enter inline math code with $x^2$ $x^2$
* Enter centered code with $$\frac{-b \pm \sqrt{b^2 - 4 a c}}{2a}$$ $$\frac{-b \pm \sqrt{b^2 - 4 a c}}{2a}$$



**HTML**

* Just include html in the Rmd file and it will get kept as html when it's slidified
* Especially useful for stuff like images or tables where you need finer control of the html options
* Also, remember you can edit the final html slide
  + This isn't the best solution (since why do mostly slidify, a reproducible format if you're going to break that reproducibility at the last step?)
  + But, sometimes useful in a pinch (like if you're frantically preparing course slides at the last minute)
* Similarly, you can incorporate JS, or anything else you can do in a web page

**Adding interactive elements to slidify**

* You can add interactive elements to slidify
  + Quiz questions
  + interactive Rcharts plots
  + Shiny apps
* Of course, you could do this directly with html/js
* More easily, the dev version of slidify has this built in
* See <http://slidify.github.io/dcmeetup/demos/interactive/>
  + The following example was taken from there

**Rmd syntax**

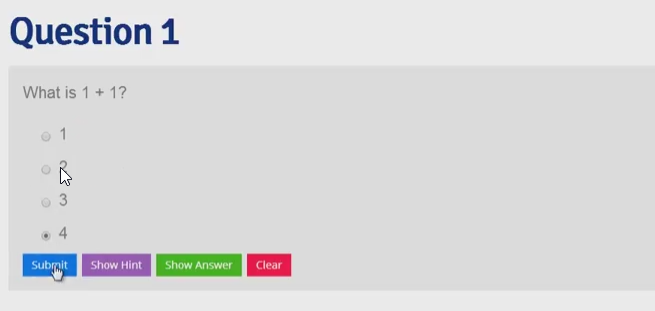
## Question 1

What is 1 + 1?

1. 1
2. \_2\_
3. 3
4. 4

\*\*\* .hint This is a hint

\*\*\* .explanation This is an explanation



--- &radio

**Question 1**

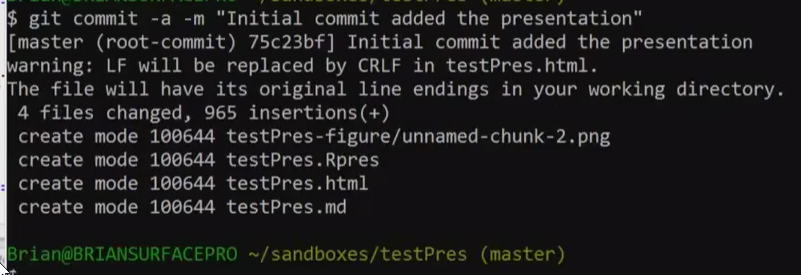
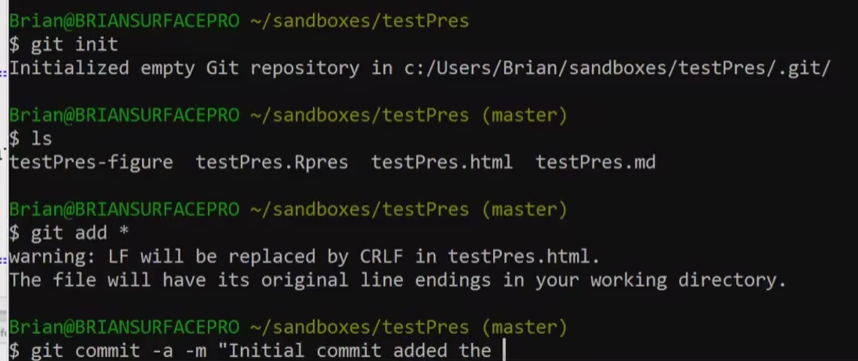
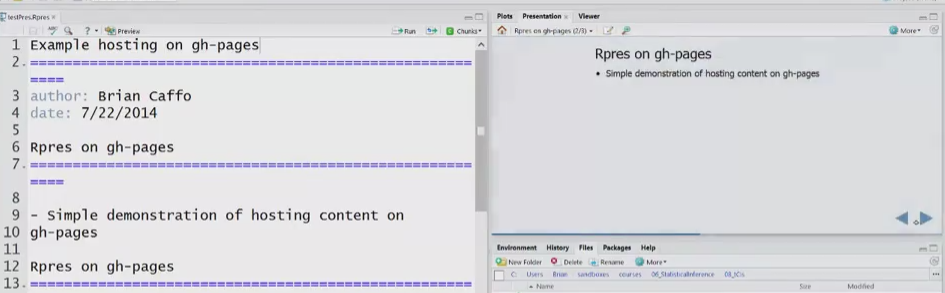
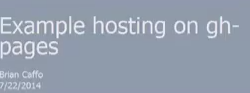
What is 1 + 1?

1. 1
2. *2*
3. 3
4. 4

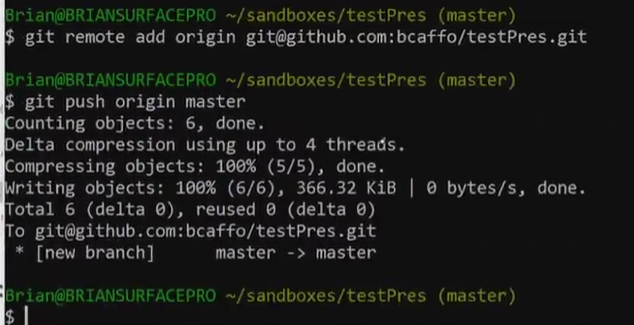
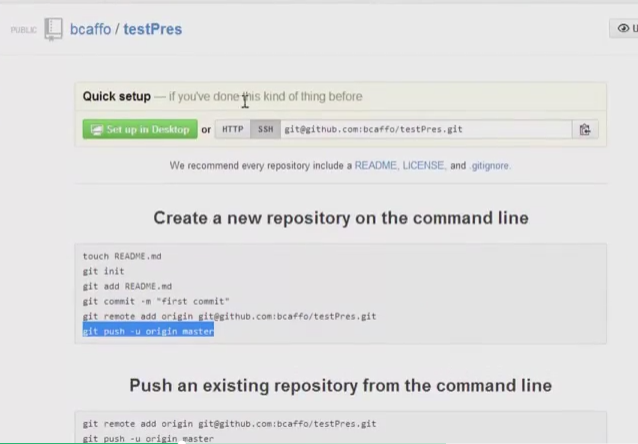
\*\*\* .hint This is a hint

\*\*\* .explanation This is an explanation

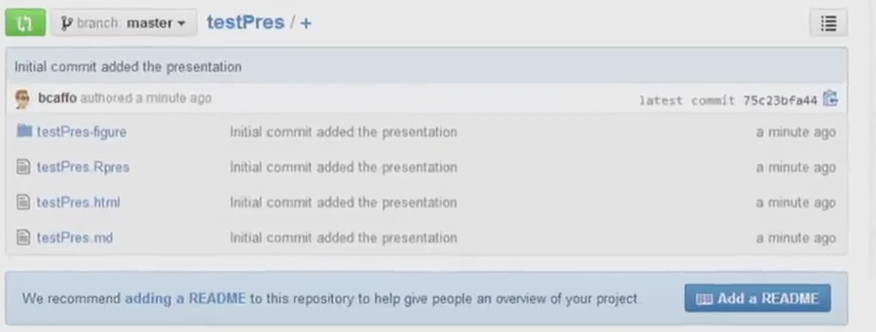
**introduction to gh-pages**



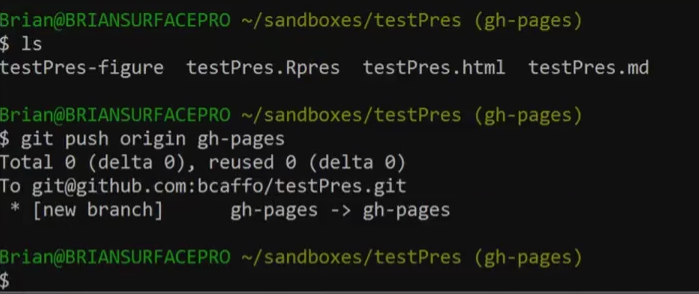
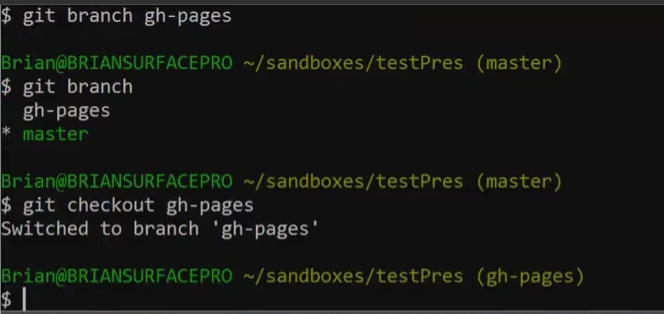
Push to github repository

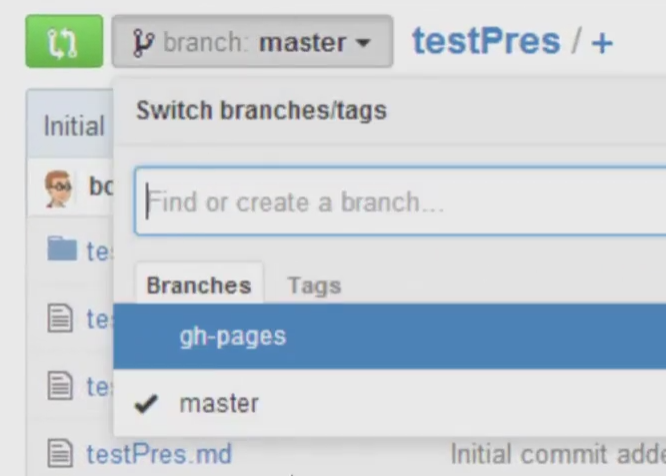


Files is oush to github testPres repository

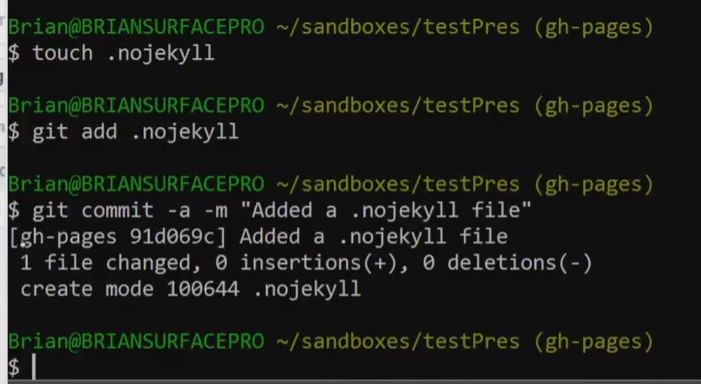


Create a github pages ot host the html pages

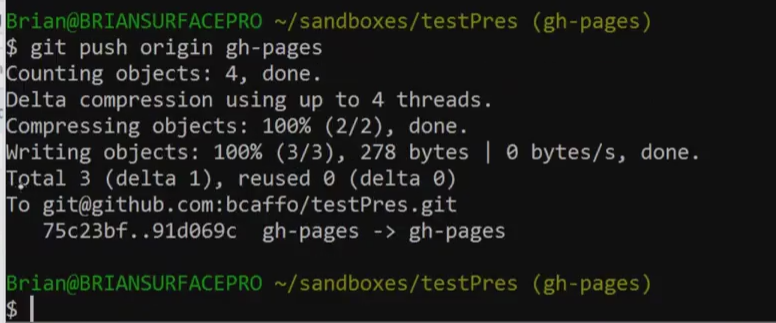


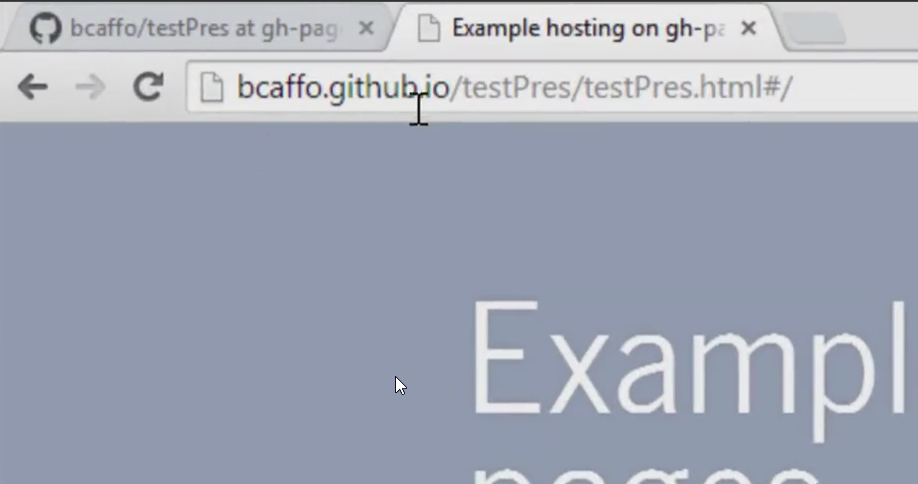


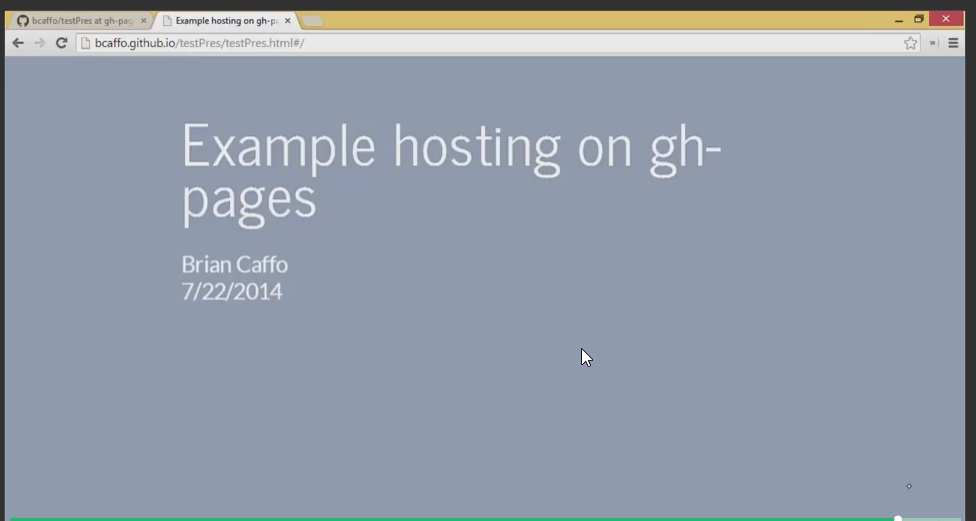
Another thing you need on your GitHub page is a .nojekyll file. The .nojekyll file merely tells Git when it's hosting to not do some of the fancy HTML stuff that GitHub pages allows for, and just do straight HTML. On the bash command line you can just do touch .nojekyll.



Push the changes to github repository







# 02: RStudio Presenter

Not to be outdone, RStudio has its own presentation software. It works much like slidify, but with small syntax differences and more integration into RStudio. If you are familiar with slidify, you will also be familiar with this tool. First, code is authored in a generalized markdown format that allows for code chunks. Secondly, the output is an html5 presentation. The file extension for the presenter file is .Rpres, which gets converted to an .md (markdown) file and then to an html file if desired. What’s nice is that there’s a preview tool in RStudio and GUIs for publishing to Rpubs or viewing/creating an html file.

**RStudio Presentation**

* RStudio created a presentation authoring tool within their development environment.
* If you are familiar with slidify, you will also be familiar with this tool
  + Code is authored in a generalized markdown format that allows for code chunks
  + The output is an html5 presentation
  + The file index for the presenter file is .Rpres, which gets converted to an .md file and then to an html file if desired
  + There's a preview tool in RStudio and GUIs for publishing to Rpubs or viewing/creating an html file

**Authoring content**

* This is a fairly complete guide
  + <http://www.rstudio.com/ide/docs/presentations/overview>
* Quick start is
  + file then New File then R Presentation
  + (alt-f then f then p if you want key strokes)
  + Use basically the same R markdown format for authoring as slidify/knitr
    - Single quotes for inline code
    - Tripple qutoes for block code
    - Same options for code evaluation, caching, hiding etcetera

**Compiling and tools**

* R Studio auto formats and runs the code when you save the document
* Mathjax JS library is loaded by default so that $x^2$ yields 
* Slide navigation button on the preview; clicking on the notepad icon takes you to that slide in the deck
* Clicking on more yields options for
  + Clearning the knitr cache
  + Viewing in a browser (creates a temporay html file in AppData/local/temp for me)
  + Create a html file to save where you want)
* A refresh button
* A zoom button that brings up a full window

**Visuals**

transition: linear

* R Studio has made it easy to get some cool html5 effects, like cube transitions with simple options in YAML-like code after the first slide such as transition: rotate
* You can specify it in a slide-by-slide basis

# Here's the option "linear"

transition: linear

* Just put transition: linear right after the slide creation (three equal signs or more in a row)
* Tansition options
  + <http://www.rstudio.com/ide/docs/presentations/slide_transitions_and_navigation>

# **Hierarchical organization**

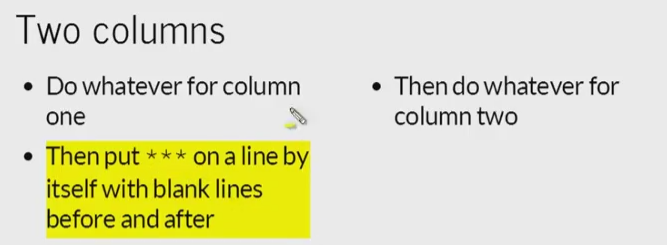
type: section

* If you want a hierarchical organization structure, just add a type: typename option after the slide
* This changes the default appearance
  + <http://www.rstudio.com/ide/docs/presentations/slide_transitions_and_navigation>
* This is of type section

# Here's a subsection

type: subsection

# **Two columns**

* Do whatever for column one
* Then put \*\*\* on a line by itself with blank lines before and after
* Then do whatever for column two

# **Changing the slide font**

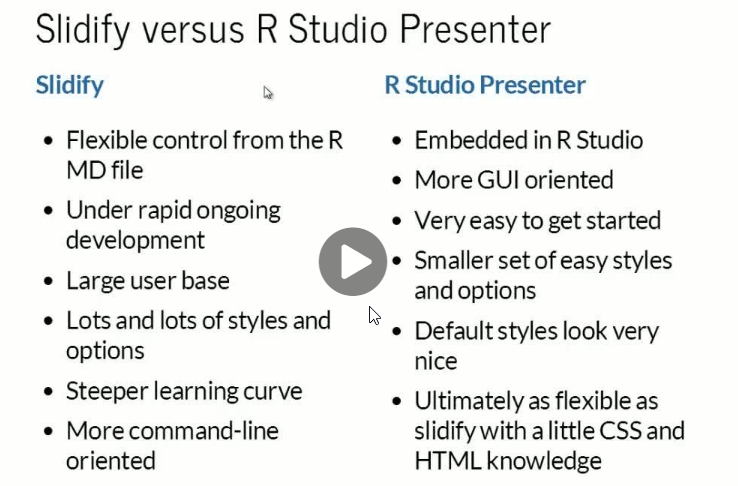
font-import: <http://fonts.googleapis.com/css?family=Risque> font-family: 'Risque'

* Add a font-family: fontname option after the slide
  + <http://www.rstudio.com/ide/docs/presentations/customizing_fonts_and_appearance>
* Specified in the same way as css font families
  + <http://www.w3schools.com/cssref/css_websafe_fonts.asp>
* Use font-import: url to import fonts
* Important caveats
  + Fonts must be present on the system that you're presenting on, or it will go to a fallback font
  + You have to be connected to the internet to use an imported font (so don't rely on this for offline presentations)
* This is the Risque
  + <http://fonts.googleapis.com/css?family=Risque>

**Really changing things**

* If you know html5 and CSS well, then you can basically change whatever you want
* A css file with the same names as your presentation will be autoimported
* You can use css: file.css to import a css file
* You have to create named classes and then use class: classname to get slide-specific style control from your css
  + (Or you can apply then within a <span>)
* Ultimately, you have an html file, that you can edit as you wish
  + This should be viewed as a last resort, as the whole point is to have reproducible presentations, but may be the easiest way to get the exact style control you want for a final product

**Slidify versus R Studio Presenter**



**Slidify**

* Flexible control from the R MD file
* Under rapid ongoing development
* Large user base
* Lots and lots of styles and options
* Steeper learning curve
* More command-line oriented

**R Studio Presenter**

* Embedded in R Studio
* More GUI oriented
* Very easy to get started
* Smaller set of easy styles and options
* Default styles look very nice
* Ultimately as flexible as slidify with a little CSS and HTML knowledge