Create amazing PowerPoint slides using R - The basics

Part 1 (3)

Let's face it, PowerPoint isn't going anywhere. Even if you use R (or Python) for data analysis, PowerPoint is how you distribute and communicate results, and learning how to create those decks as part of your R workflow can pay off. Beyond efficiency, and repeatability, programmatic access enables you to do things that just aren't possible with point and click.

(Here's a short video of what we're aiming for, how it was created and the R code.)

In this talk, S Anand uses Python and *pywin32* to create some jaw dropping effects in PowerPoint, scraping data from IMDB and creating a PowerPoint slide using the data. *RDCOMClient* by Duncan Temple Lang allows you to do the same thing using R. It provides the ability to "access and control applications such as Excel, Word, PowerPoint, Web browsers etc."

We'll try to recreate some elements of S Anand's talk, and add a few things, with a focus on interaction and animation. We'll go through the basics of accessing methods and properties of PowerPoint VBA objects, scrape data on Clint Eastwood's movies from IMDB, and use it to create a slide. The end result should be a fun PowerPoint slide with Clint Eastwood's filmography.

Since the goal is to learn and explore a bit, there will be lots of code and animation that's not strictly necessary.

(Clearly PowerPoint has a powerful animation engine, and the object model allows you to programmatically manipulate almost everything. The Microsoft documentation, however, seems to be organized as a mystery.)

In this first part, we look at the basics of manipulating PowerPoint from R.

Let's get started.

Setup

I'm using:

- Windows 7 (64bit)
- Office 2013 (64bit)
- R version 3.1.3
- RStudio verison 0.98

For now we just need this one package.

1. RDOCMClient by Duncan Temple Lang. COM client that allows us to manipulate PowerPoint using R.

Start RStudio and Load the packages:

```
install.packages("RDCOMClient")
library(RDCOMClient)
```

RDCOMClient Basics

Some RDCOMClient basics we'll need to know:

- To start a new instance of an application: `COMCreate("xxx.Application")`
- Executing methods takes the form: `comObj$methodName(arg1,arg2,arg3,...)`
- Setting properties takes the form: `myObj[["Property"]] = TRUE`

With these basic capabilities, we can access and manipulate all the objects exposed in the PowerPoint API.

Let's create a new PowerPoint presentation from R

```
# Start up PowerPoint
pp <- COMCreate("Powerpoint.Application")

# Make the application visible
pp[["Visible"]] = 1

# Add a new presentation
presentation <- pp[["Presentations"]]$Add()

# The presentation is empty.  Add a slide to it.
# We'll get to the enumerated constants defined by Microsoft.
# For now, let's use the value (12) that results in a blank layo
# slide1 <- presentation[["Slides"]]$Add(1,ms$ppLayoutBlank)
slide1 <- presentation[["Slides"]]$Add(1,12)
```

The critical part is, of course, knowing what methods and properties are available. This is where the PowerPoint 2013 Developer Reference is handy, but not terribly user-friendly. (You can also get this information from the Object Browser in the VBA editor.)

To understand how to go from VBA to R, let's recreate one of the basic examples in the documentation, Applying Animations to Shapes in Office 2010, which works with Office 2013 as well.

The VBA code:

```
Sub TestPickupAnimation()
    With ActivePresentation.Slides(1)
        Dim shp1, shp2, shp3 As Shape
        ' Create the initial shape and apply the animations.
        Set shp1 = .Shapes.AddShape(msoShape12pointStar, 20, 20,

        .TimeLine.MainSequence.AddEffect shp1, msoAnimEffectFade
        .TimeLine.MainSequence.AddEffect shp1, msoAnimEffectPath
        .TimeLine.MainSequence.AddEffect shp1, msoAnimEffectSpir

        ' Now create a second shape, and apply the same animatic
        shp1.PickupAnimation

        Set shp2 = .Shapes.AddShape(msoShapeHexagon, 100, 20, 10
        shp2.ApplyAnimation

        ' And one more:
        Set shp3 = .Shapes.AddShape(msoShapeCloud, 180, 20, 100,
        shp3.ApplyAnimation

    End With
End Sub
```

The code above has constants that Microsoft has defined for each element of the presentation, such as shape, animation, trigger, etc. It wasn't easy to find a consolidated list of all the enumerated constants. (I could not find the site I got the constants from, so apologies for the lack of attribution.) I created a file and just read them into one variable, `ms`. (Download it to your working directory if you want to execute the code.)

In the `AddEffect` method above, you'll notice an "empty" argument. That tripped me up for a while, since it doesn't work with R. The AddEffect method shows the arguments the method expect `(expression.AddEffect(Shape, effectId, Level, trigger, Index))`. Using an explicitly named argument (trigger) works.

Let's add the shapes and animation from the Microsoft example to the presentation we created earlier:

```
source("mso.txt")
shp1 <- slide1[["Shapes"]]$AddShape(ms$msoShape12pointStar,20,20
slide1[["TimeLine"]][["MainSequence"]]$AddEffect(shp1,ms$msoAnim
                                                 trigger=
slide1[["TimeLine"]][["MainSequence"]]$AddEffect(shp1,ms$msoAnim
                                                 trigger=
slide1[["TimeLine"]][["MainSequence"]]$AddEffect(shp1,ms$msoAnim
                                                 trigger=

shp1$PickupAnimation()
shp2 <- slide1[["Shapes"]]$AddShape(ms$msoShapeHexagon,100,20,10
shp2$ApplyAnimation()

shp3 <- slide1[["Shapes"]]$AddShape(ms$msoShapeCloud,180,20,100,
shp3$ApplyAnimation()
```

This should create a presentation with the three shapes. If you go into animation mode, the shapes should appear, with the animations triggered.

The `shape` object comes with a long list of properties and methods. Let's see how to use some of them. We'll add text, and change colors.

Each `shape` has an associated `TextFrame property`, which returns an object containing all the text related elements of the shape.

```
# Add text to the shapes.  While this works, R files a complaint
shp1[["TextFrame"]][["TextRange"]][["Text"]] <- "Shp1"

# This way seems to function better
shp1_tr <- shp1[["TextFrame"]][["TextRange"]]
shp1_tr[["Text"]] <- "ONE"
```

Let's set some shape attributes.
The `Fill` property is used for the colors, and the `Line` property for the border.

```
shp1_color <- shp1[["Fill"]]
shp1_color[["ForeColor"]][["RGB"]] <- (0+170*256+170*256^2)
# That's how the RGB value is calculated: r +  g*256 + b*256*256

# Remove the line
shp1_line <- shp1[["Line"]]
shp1_line[["Visible"]] <- 0
```

Now, do it for the other shapes as well. We'll create a function to handle the color encoding:

```
# Create function for the rgb calculation
pp_rgb <- function(r,g,b) {
    return(r + g*256 + b*256^2)
}

shp2_tr <- shp2[["TextFrame"]][["TextRange"]]
shp2_tr[["Text"]] <- "TWO"
shp2_color <- shp2[["Fill"]]
shp2_color[["ForeColor"]][["RGB"]] <- pp_rgb(170,170,0)
shp2_line <- shp2[["Line"]]
shp2_line[["Visible"]] <- 0

shp3_tr <- shp3[["TextFrame"]][["TextRange"]]
shp3_tr[["Text"]] <- "THREE"
shp3_color <- shp3[["Fill"]]
shp3_color[["ForeColor"]][["RGB"]] <- pp_rgb(170,0,170)
shp3_line <- shp3[["Line"]]
shp3_line[["Visible"]] <- 0

# Finally, save the file in the working directory
presentation$SaveAs(paste0(getwd(),"/PowerPoint_R_Part_1.pptx"))
```

The code and the PowerPoint file created are available from Github.

Coming up:

In Part 2, we scrape some data using `rvest` and create a dataset of Clint Eastwood's movies.

Part 3 - Then some fun! We'll use the data to play around with more advanced animation and interaction in PowerPoint.

Written on May 10, 2015

"mso.txt" file.

• Reply • Share ›

Leon Katsnelson • 8 months ago

Is there a way to do this in PowerPoint for Mac?

• Reply • Share ›

Asif Salam Mod → Leon Katsnelson • 8 months ago

I'm not sure Leon. I think you'd need to figure out three things: 1 - if there's DCOM equivalent for the MAC, 2 - if the Mac PowerPoint object model is exposed in a similar way, and 3 - if the there's a package similar to RDCOMClient for the Mac version of R.

• Reply • Share ›

Christopher Harper • 8 months ago

Hi, I keep getting an error with the line "slide1 = presentation[["Slides"]]$Add(1,ms$ppLayoutBlank)"

Error in .COM(x, name, ...) : object 'ms' not found

• Reply • Share ›

Asif Salam Mod → Christopher Harper • 8 months ago

Added a link in the code to the mso.txt file that contains the enumerated constants. That file is sourced on line 25 in the code for Part 1. There's a link to the file in the tutorial as well.

1 • Reply • Share ›

Asif Salam Mod → Christopher Harper • 8 months ago

Sorry, that's an error in the code and the tutorial. The variable ms$ppLayoutBlank is used before the variable is defined. I have updated the code to explicitly use the value "12" which should result in blank slide.
Thanks for the feedback!

1 • Reply • Share ›

Comandante Mahendoy → Christopher Harper • 8 months ago

Yep, I'm getting the same error here.

• Reply • Share ›