Introduction to Visualizations with R

for Exploratory Data Analysis

Irina Kukuyeva, Ph.D.
www.KukuyevaConsulting.com

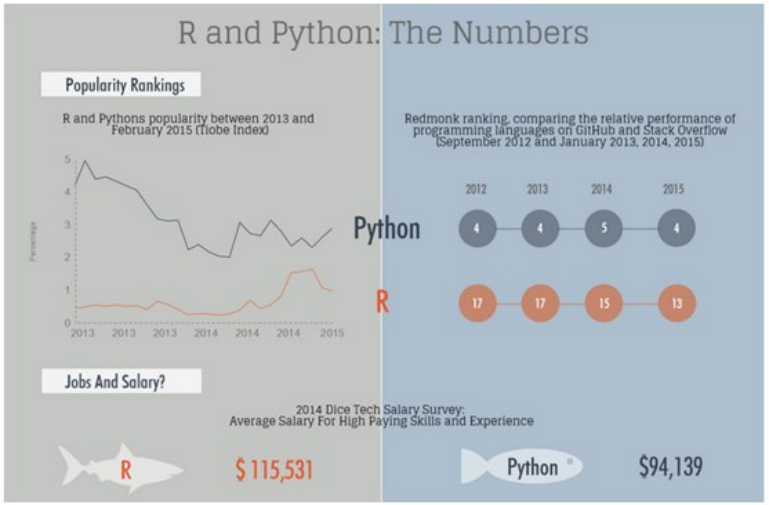
March 3, 2016

Tutorial Outline

Working with R

Why Tutorial?

- Derive and convey key insights from data
- Learn and use open source software
- Explore first/another programming language
- Gain ability to create insightful visualizations
- Be more marketable



[6]

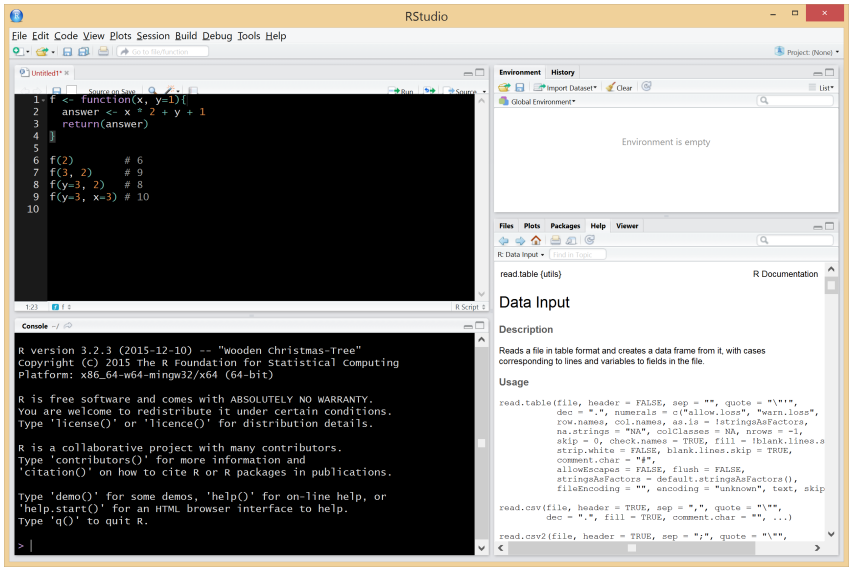
Why R?

- Absolutely free!
- Used in industry and academia.
- Has a great community:
  - StackOverflow
  - Blogs
  - Meetup groups
  - MOOCs
  - many, many others
- Has over 7900 packages available for use (for free!).
- Transparent code (e.g. easier to check for bugs).

Installing R and RStudio

R: Go to `http://cran.r-project.org/`, select your operating system and download the latest version: 3.2.3 (Release 2015/12/10).

RStudio: Go to `https://www.rstudio.com/products/rstudio/download/`, select your operating system and download the installer (if available).



Overview of R per John Chambers [1]:

"To understand computations in R, two slogans are helpful:

- Everything that exists is an object.
- Everything that happens is a function call."

Objects in R I

Creating objects in R

Objects can be created in different ways, via:

- equal sign: `x = 3`
- left arrow: `y <- 2 * x + 1`
- right arrow: `10 -> z`

Objects in R II

Examples of Objects

- Variables, which can be either discrete or continuous:

Discrete: e.g. number of people at the tutorial
Continuous: e.g. how long it took to get to the tutorial

- Data frames, which contain data sets with at least one variable:

EX 1: e.g. data set of commonly performed procedures at CA hospitals (variables include hospital name, procedure type, number of patients who had that procedure, etc.)
EX 2: e.g. data set of popular movies (variables include movie name, genre, lead actor/actress, etc.)

Objects in R III

Exercise:

Create a variable called y with a value of -10.

Functions in R

Function by example

```
f <- function(x, y=1){
        answer <- x * 2 + y + 1
        return(answer)
}

f(2)          # 6
f(3, 2)       # 9
f(y=3, 2)     # 8
f(y=3, x=3)   # 10
```

Components of a function:

- Assign function to a variable
- Add the 'function' keyword
- Specify arguments (if any) that function needs to compute result
- Specify any argument defaults

Making Comparisons

Comparisons return a TRUE or FALSE depending on the condition evaluated:

```
# Suppose x and y are:
x = 3
y = 5
```

```
What should the following comparisons return?
x == 3   # is x 3?
x != y   # are x and y different?
x >= y   # is x greater than or equal to y?
(x==3) & (y==4) # is it true that x is 3 and y is 4?
(x==3) | (y==4) # is it true that x is 3 or y is 4?
```

Notation/Conventions I

%>%: 'pipe operator' which sends objects on the left-hand side to be processed by functions on the right-hand side [15]

- x %>% f is equivalent to f(x)

c(): 'concatenation' which combines objects together

```
x = 3
y = 5
z = c(x, y)
z
```

```
[1] 3 5
```

Notation/Conventions II

Brackets: 'subsetting' which says what observation we are (not) interested in

```
# Example 1:
z = c(3, 5)
z[2]
```

```
[1] 5
```

```
# Example 2:
z = c(3, 5)
z[ z > 3 ]
```

```
[1] 5
```

Reading Data from File I

One approach is via `read.table()`. In the arguments of the function:

- `file:` specifies the (relative) location, file name and file extension
- `header:` if TRUE, tells R to include variables names when importing
- `sep:` tells R how the entires in the data set are separated
  - `sep=",":` when entries are separated by COMMAS
  - `sep="\t":` when entries are separated by TAB
  - `sep=" ":` when entries are separated by SPACE

Reading Data from File II

```
filepath = "http://www.ats.ucla.edu/stat/data/test_missing_comma.txt"
### Other valid paths:
# filepath = "C:/Documents/test_missing_comma.txt"
# filepath = "./test_missing_comma.txt"

df <- read.table(
        file = filepath,
        header = TRUE,
        sep = ","
        )
```

Working with Data Frames I

To check that a data set has been read-in correctly:

- View the complete data set by typing the variable name. (This is not recommended for large data sets.)

```
df
```

```
    prgtype gender  id ses schtyp level
1   general      0  70   4      1     1
2    vocati      1 121   4     NA     1
3   general      0  86  NA     NA     1
4    vocati      0 141   4      3     1
5  academic      0 172   4      2     1
6  academic      0 113   4      2     1
7   general      0  50   3      2     1
8  academic      0  11   1      2     1
```

Working with Data Frames II

- View the first 5 lines of the dataset via `head()`:

```
head(df, 5)
```

```
    prgtype gender  id ses schtyp level
1   general      0  70   4      1     1
2    vocati      1 121   4     NA     1
3   general      0  86  NA     NA     1
4    vocati      0 141   4      3     1
5  academic      0 172   4      2     1
```

Working with Data Frames III

- View the last 7 lines of the dataset via `tail()`:

```
tail(df, 7)
```

```
    prgtype gender  id ses schtyp level
2    vocati      1 121   4     NA     1
3   general      0  86  NA     NA     1
4    vocati      0 141   4      3     1
5  academic      0 172   4      2     1
6  academic      0 113   4      2     1
7   general      0  50   3      2     1
8  academic      0  11   1      2     1
```

Working with Data Frames IV

- Check the variable names via `names()`:

```
names(df)
```

```
[1] "prgtype" "gender"  "id"      "ses"     "schtyp"  "level"
```

- Check the size of the data set via `dim()`:

```
dim(df)
```

```
[1] 8 6
```

- See the first 6 entries for variable 'gender' via `head()`:

```
head(df$gender)
```

```
[1] 0 1 0 0 0 0
```

Working with Data Frames V

- Examine how many unique levels a variable has via `unique()`:

```
unique(df$gender)
```

```
[1] 0 1
```

- Examine the counts of levels that a variable has via `table()`:

```
table(df$gender, useNA='always')
```

```
0    1 <NA>
7    1    0
```

Working with Data Frames VI

- View entries for a particular variable:

```
## Returns variable as a row/vector:
df$gender # OR
df[, 'gender']
## Returns variable as a column:
df[ 'gender' ]
```

```
[1] 0 1 0 0 0 0 # OR

  gender
1      0
2      1
3      0
4      0
5      0
6      0
7      0
8      0
```

Working with Data Frames VII

- View entries for a few variables:

```
head( df[, c('gender', 'ses') ], 3)
```

```
  gender ses
1      0   4
2      1   4
3      0  NA
```

Working with Data Frames VIII

- Verify ranges and check for missing data via `summary()`:

```
summary(df)
```

```
    prgtype      gender            id              ses            schtyp      level
  vocati:2   Min.   :0.000   Min.   : 11.0   Min.   :1.000   Min.   :1   Min.   :1
 general:3   1st Qu.:0.000   1st Qu.: 65.0   1st Qu.:3.500   1st Qu.:2   1st Qu.:1
academic:3   Median :0.000   Median : 99.5   Median :4.000   Median :2   Median :1
             Mean   :0.125   Mean   : 95.5   Mean   :3.429   Mean   :2   Mean   :1
             3rd Qu.:0.000   3rd Qu.:126.0   3rd Qu.:4.000   3rd Qu.:2   3rd Qu.:1
             Max.   :1.000   Max.   :172.0   Max.   :4.000   Max.   :3   Max.   :1
                                             NA's   :1       NA's   :2
```

Adding Functionality to Base R

- Base R is what you download off CRAN
  `www.cran.r-project.org`
- Available packages are listed here:
  `https://cran.r-project.org/web/packages/`
- Install an R package(s) via `install.packages()`:

```
install.packages("lattice")
install.packages("ggplot2")
# OR
install.packages( c("lattice", "ggplot2") )
```

Part I

Working with R

Error: syntax error

Possible causes:

- Misspelling the object's name
- Including a "+" when copying code from console/website, etc.
- Having an extra parenthesis at the end of a function
- Having an extra bracket when subsetting

Trailing +

Possible causes:

- Not closing a function call with a parenthesis
- Not closing brackets when subsetting
- Not closing a function you wrote with a squiggly brace

Error in ... : requires numeric matrix/vector arguments

Possible causes:

1. Objects are data frames, not matrices
2. Elements of the vectors are characters

Possible solutions:

1. Coerce (a copy of) the data set to be a matrix, with the `as.matrix()` command
2. Coerce (a copy of) the vector to have numeric entries, with the `as.numeric()` command

Error: ... object not found

Possible causes:

1. Misspelling the object's name
2. Package containing the object has not been loaded

Silent Errors

Most common silent errors:

1. (Inadvertently) Creating a data set with no rows or columns.
2. (Inadvertently) Recycling (and padding) of entries in a variable with a smaller number of observations than the one it is compared to.

Possible solutions:

1. Always check the dimensionality of the data set after subsetting.
2. Always check the lengths of variables ahead of comparison, especially if subsetting just took place.

For more caveats and solutions, read the "R Inferno":
`http://www.burns-stat.com/pages/Tutor/R_inferno.pdf`

Where to go from here?

Part 1: Explore other visualizations

Other exploratory visualizations in R that we didn't get to cover:

- Sankey graphs
- (Interactive) dashboards via R package 'shiny'
- Creating animations with R
- ...

Where to go from here?

Part 2: Familiarize yourself with R (as it relates to EDA)

Other data-related topics that we didn't get to cover:

- Variety of ways to aggregate data
- Getting data via an API (e.g. meetup, yelp, etc.)
- Developing reproducible reports (via 'knitr' package)
- ...

Where to go from here?

Part 3: Familiarize yourself with HTML/JS/d3/···

Other visualization topics that we didn't get to cover:

- Ability to customize design of interactive graphics (e.g. `http://datascience.la/interactive-visualizations-from-r-using-rcharts/` and `http://www.slideshare.net/f0008/javascriptbased-visualization-in-r`)
- Ability to customize design of interactive dashboards (e.g. `http://shiny.rstudio.com/articles/`)
- Ability to customize design of reproducible reports (e.g. `http://shiny.rstudio.com/gallery/download-knitr-reports.html`)
- ...

Where to go from here?

Part 4: Familiarize yourself with data analysis models

We did not cover model building as a way to explain relationships in the data, such as:

- Different types of regression models for modeling numerical data
- Different types of decision trees for modeling numerical data
- Different types of models for analyzing text/image/video/audio data
- ...

Please see the 'Online Resources for R' section (below) for more information.

Where to go from here?

Part 5: Familiarize yourself with 'best practices'

We did not explicitly cover any best practices such as:

- Commenting code
- Clear variable names
- Version control
- ...

Please see the following for more information:

- Google's R style guide: `https://google.github.io/styleguide/Rguide.xml`
- Joel's Test for writing better code: `http://www.joelonsoftware.com/articles/fog0000000043.html`
- Iliinsky and Steele's book on *Designing Data Visualizations*
- ...

Where to go from here?

Part 6: Connect with Other Data Scientists

- meetup.com (e.g. R Users' Group, Data Viz LA, etc.)
- LinkedIn
- Conferences: userR 2016, Big Data LA
- ...

R Help: Approach 1

For help with any function in R, add a question mark before the function name to see the documentation (which includes explanation of the function's arguments/inputs, function outputs and example use cases.

```
?plot
```

plot {graphics} R Documentation

Generic X-Y Plotting

Description

Generic function for plotting of R objects. For more details about the graphical parameter arguments, see `par`.

For simple scatter plots, `plot.default` will be used. However, there are `plot` methods for many R objects, including `function`s, `data.frame`s, `density` objects, etc. Use `methods(plot)` and the documentation for these.

Usage

```
plot(x, y, ...)
```

Arguments

`x`
the coordinates of points in the plot. Alternatively, a single plotting structure, function or *any R object with a `plot` method* can be provided.

`y`
the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.

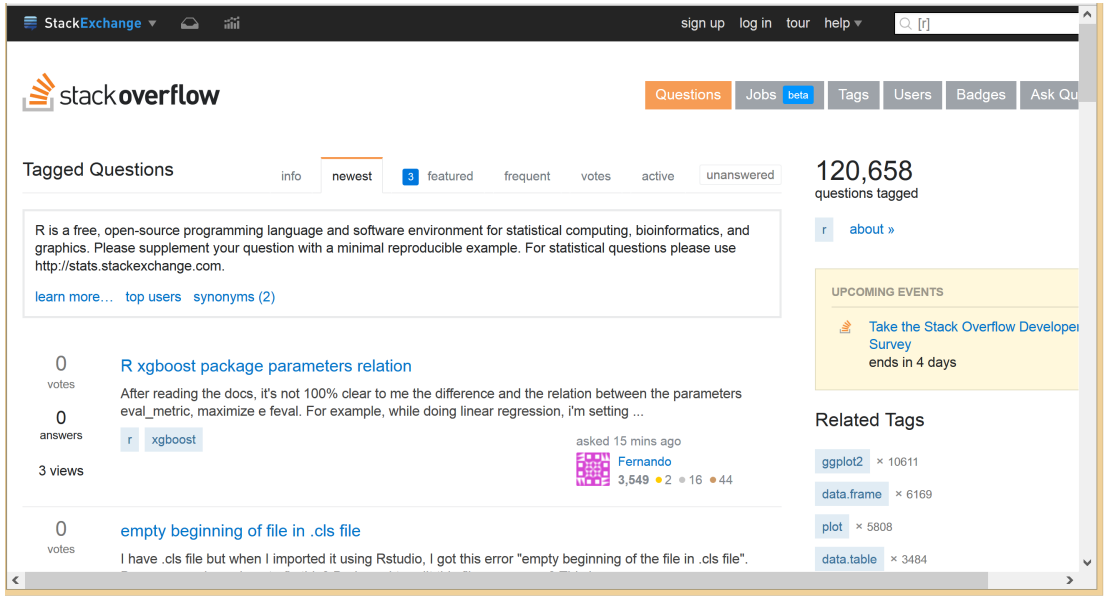
`...`
Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type`
what type of plot should be drawn. Possible types are

- "`p`" for points,

R Help: Approaches 2 and 3

- For help with any function in R, search answers on StackOverflow (SO).
- For help with any function in R, when all else fails, ask a question on StackOverflow. Don't forget to follow the SO tips: `http://stackoverflow.com/help/how-to-ask`



Online Resources for R I

Download R: *http://cran.stat.ucla.edu/*

Download RStudio: *https://www.rstudio.com/*

R Reference Card:
http://cran.r-project.org/doc/contrib/Short-refcard.pdf

More R tutorials:

Courses: Code School, Coursera, DataCamp, DataRobot, edX, RStudio, swirl

IK: *http://www.KukuyevaConsulting.com/tutorials*

UCLA IDRE: *http://www.ats.ucla.edu/stat/r/*

UCLA SCC: *http://scc.stat.ucla.edu/mini-courses/*

Online Resources for R II

R Graphics Gallery: *http://research.stowers-institute.org/efg/R/*

R Graph Gallery: *http://addictedtor.free.fr/graphiques/*

Book: *http://book.flowingdata.com/*

Book: 'Interactive Data Visualization' (O'Reilly)

Bokeh: *http://bokeh.pydata.org/en/latest/*

Stacked graph: *http://menugget.blogspot.com/2013/12/data-mountains-and-streams-stacked-area.html*

Blog: *http://spatial.ly/*

Blogs: *http://www.r-bloggers.com/*

JSS: *https://www.jstatsoft.org/index*

Stackoverflow: *http://stackoverflow.com/tags/r/info*

1. http://adv-r.had.co.nz/
2. http://www.sixhat.net/how-to-plot-multpile-data-series-with-ggplot.html
3. http://stackoverflow.com/questions/17584248/exact-axis-ticks-and-labels-in-r-lattice-xyplot
4. https://rstudio-pubs-static.s3.amazonaws.com/3364_d1a578f521174152b46b19d0c83cbe7e.html
5. www.jstatsoft.org/v25/c01/paper
6. http://www.kdnuggets.com/2015/05/r-vs-python-data-science.html
7. http://flowingdata.com/2014/02/05/where-people-run/

8. https://www.facebook.com/notes/facebook-engineering/visualizing-friendships/469716398919
9. Iliinsky, N. and Steele, J. (2011) *Designing Data Visualizations* Sebastopol, CA: O'Reilly.
10. http://www.pixel-push.com/2013/09/24/ultimate-infographic-resource-kits-for-designers/
11. Heiberger, R. M. (2016, February). Design of Not-Simple Graphs. Talk presented at the meeting of the American Statistical Association Conference on Statistical Practice, San Diego, CA.

12. https://github.com/hadley/ggplot2/wiki/plotting-polygon-shapefiles
13. http://www.r-bloggers.com/making-static-interactive-maps-with-ggvis-using-ggvis-
14. http://ggvis.rstudio.com/interactivity.html
15. https://github.com/smbache/magrittr

Thank you.
Any questions?