Chris Hodapp
chodapp3@gatech.edu

2017-01-10, CSE6242 (Data & Visual Analytics)

Lesson 1 - R

- Why R:
  - Easy process for making new packages
  - Still quite fast with multidimensional arrays
  - Better visualization than Python
  - RStudio is awesome (nixos has it too)
- R: Stats, biostatistics, social sciences
- MATLAB: Engineering, applied math
- Python: Web dev, scripting
- To run R non-interactively:
  - From R, `source("foo.R")`
  - From shell, `R CMD BATCH foo.R`
    - or: `Rscript foo.R`
    - or: `#!/usr/bin/Rscript` at top, and `./foo.R`
- Other stuff:
  - `ls()` - List names in memory
  - `save.image(file="R_workspace")` - save R vars to file (all vars)
  - `save(new.var, legal.var.name, file="R_workspace")` - only certain vars
  - `load("R_workspace")` - load from file
  - Packages
    - `install.packages("ggplot2")` - download & install packages
    - `library(ggplot2)` - bring into scope
  - Shell: `system("ls -al")` for instance
- Types:
  - numeric (as.numeric)
  - integer (as.integer)
  - logical
  - string
  - factors - Sort of like enums, can be ordered or unordered. (See example)
  - vector - `c(...)` = concatenate
    - `length(...)` = show length
    - `vector(mode="logical", length=4)` - FALSE FALSE FALSE FALSE
      (false = default value)

[rep(32, times = 10) - Repeat
[seq(0, 1, by = 0.1) - Load in increments
[seq(0, 1, length.out = 11) - Same, but specific length
] Same result

Lesson 1 - R (cont.)

- Arrays: 'array' constructs (column-major)
  - x[2,3] - row 2, column 3; x[2,] - row 2, all columns
  - x[-1,] - All but 1st row, all columns
  - x[c(1,2), c(1,2)] - rows 1&2, cols. 1&2
  - x %*% y - matrix multiply (* is elementwise)
- 'outer', outer product
- rbind & cbind work with arrays

- Rcpp, RcppArmadillo, RcppEigen
  - dyn.load, .Call

Netflix interview

- He uses R & Python a lot, finds Python better for crunching through lots of data
- Other guy: Java & Scala, Spark on offline backend (and Hadoop, Hive, Pig)
- They prefer R & Python for prototyping, Java for production
  - R & Python are much more suited to line-by-line testing and quickly getting moving; Java works much better at larger-scale projects

2017-01-26, Data Visualization

Chapter 10, Data Viz

- ggplot2: Slower than built-in R plotting but much easier
- Visualization <
  - initially investigating datasets
  - confirming/refuting data models
  - elucidating mathematical/algorithmic concepts
- Main two ggplot2 functions:
  - qplot - quickly plot from dataframe with x, y, z
  - ggplot - more complex, works with layers that + adds.
- Labels are automatically made for axes & legends provided you have names in dataframe
- Simple scatter plot:

[ggplot(dataframe, aes(x=col1, y=col2)) + geom_points()

Chris Hodapp
chodapp3@gatech.edu

2017-01-26(6), CSE6242 (Data & Visual Analytics)

Lesson 2| Ch. 10 (Data Viz.), Cont.

- Histogram:
```
ggplot(FaithFul, aes(x=waiting)) +
    geom_histogram(binwidth = 1)
```
- They reference:
  "Probability: The Analysis of Data, Volume 1" (Guy Lebanon)
  Also: theanalysisofdata.com → hence TAOD
- For a "proper" density function (area = 1):
```
ggplot(faithful, aes(x=waiting, y = ..density..) + geom_histogram(binwidth =4)
```
- Smoothed histogram: $f_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_H(x - x^{(i)})$

  where $x^{(i)}$ is ith data point, $K_H: \mathbb{R} \to \mathbb{R}$ is kernel function (typ. $K_H(0)$ is maximum, and falls off as $|x - x^{(i)}|$ increases). Also, $\int K_H(x)\,dx = 1$ & $K_h(r) = h^{-1} K_1(r/h)$
  - K_1 is "base form" of kernel, denoted just K.
  - 4 popular kernels:

    Tricube: $K(r) = \left(1 - |r|^3\right)^3 \cdot 1_{\{|r|<1\}}$

    Triangular: $K(r) = \left(1 - |r|\right) \cdot 1_{\{|r|<1\}}$

    Uniform: $K(r) = 2^{-1} \cdot 1_{\{|r|<1\}}$

    Gaussian: $K(r) = \exp(-r^2/2) / \sqrt{2\pi}$

    where $K_h(r) = h^{-1} K(r/h)$
  - See p311 for examples
  - TAOD vol 1 chapter 2: f_h above actually is a formal estimator of the underlying distribution (this is why the $1/n$ is there and $\int K(x)dx = 1$)
- These can be much more useful & intuitive than non-smoothed histograms. Just as bin width is crucial there, choice of kernel function is also important here.
- ggplot2 gladly does smoothed histograms; see kernel & adjust keywords for gplot, geom_density
- Non-smoothed histograms are worse estimators of distribution but have a more direct relationship with data - which can be helpful too

- See stat_smooth with ggplot
- Pass 'plot' a dataframe, and it will plot an array of all pairs of variables
  (Or: ggpairs from package GGallly)
- Look at 'facets' keyword to ggplot
- Contour plots:
  - Add a 'z' variable to ggplot call, and + stat_contour()
- qq plots (quantile - quantile) - useful for trying to compare two distributions (more precisely, data from two distributions that may or may not be related).
- Smoothing: $y_s(x) = \sum_{i=1}^{n} \frac{K_h(x - x^{(i)})}{\sum_{i=1}^{n} K_h(x - x^{(i)})} y(i)$
  - "Locally constant regression"; it estimates x & y relationship without making parametric assumptions