

Advanced Dogfood Eating

Interactive graphics from R with Shiny and GoogleVis

Josh Laurito

To Dos

- Announcements
- Intro to DataKind from Peter Darche
- Cover last module's homework
- Moving to interactive graphics
- googleVis
- shiny
- Next module's reading
- Next module's assignment

Announcemments

- Blackboard is not providing you with enough detail about your assignments
- We will be changing how feedback is shared
- We also will be having an NYC meetup

Intro to DataKind from Peter Darche

• www.datakind.org



Last module's homework

- Thank you all for being great about installing software early
- How did you combine files? I used `csvkit` <http://csvkit.readthedocs.org/>
- load what we need

```
library("ggplot2")  
library("plyr")  
library("bigvis")
```

```
## Loading required package: Rcpp  
##  
## Attaching package: 'bigvis'  
##  
## The following object is masked from 'package:ggplot2':  
##  
##      movies  
##  
## The following object is masked from 'package:stats':  
##  
##      smooth
```

```
# library('bigmemory')  
pData <- read.csv("/Users/JL/Box Sync/CUNY/datasets/all_PLUTO_data.csv")
```

Last week's homework - data cleaning

```
builtFar <- pData$BuiltFAR[pData$YearBuilt > 1850 & pData$NumFloors != 0]

numFloor <- pData$NumFloors[pData$YearBuilt > 1850 & pData$NumFloors != 0]

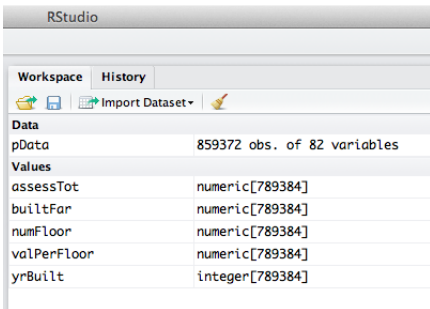
yrBuilt <- pData$YearBuilt[pData$YearBuilt > 1850 & pData$NumFloors != 0]

assessTot <- pData$AssessTot[pData$YearBuilt > 1850 & pData$NumFloors != 0]

valPerFloor <- assessTot/numFloor
```

Last week's homework - lots of data

- You probably noticed that there was a lot of data
- Slows down analysis: viz becomes more important and harder to do



The screenshot shows the RStudio interface. At the top, the title bar says 'RStudio'. Below it, there are tabs for 'Workspace' and 'History'. Under the 'Workspace' tab, there are icons for a file, a folder, and a document, followed by a dropdown menu labeled 'Import Dataset'. Below this, the 'Data' pane shows a table with two columns: 'Data' and 'Values'. The 'Data' column lists several variables: 'pData', 'assessTot', 'builtFar', 'numFloor', 'valPerFloor', and 'yrBuilt'. The 'Values' column shows the data type and the number of observations for each variable: '859372 obs. of 82 variables' for 'pData', and 'numeric[789384]' for 'assessTot', 'builtFar', 'numFloor', and 'valPerFloor'. 'yrBuilt' is listed as 'integer[789384]'.

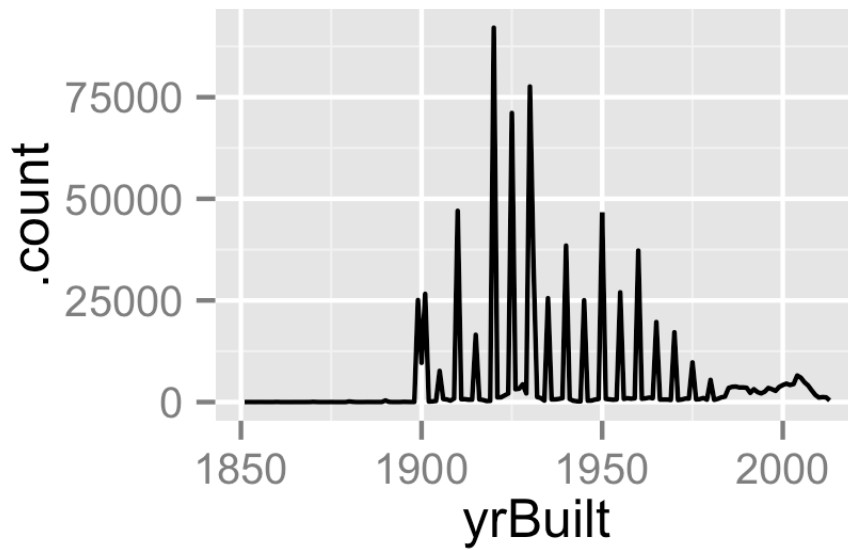
| Data | Values |
|-------------|-----------------------------|
| pData | 859372 obs. of 82 variables |
| assessTot | numeric[789384] |
| builtFar | numeric[789384] |
| numFloor | numeric[789384] |
| valPerFloor | numeric[789384] |
| yrBuilt | integer[789384] |

Last week's homework - building 'cut-off date'

- Poorly specified problem
- General concept: what does it mean to be an 'old building'
- We should start by checking when buildings were built

```
summary(yrBuilt)
yr <- condense(bin(yrBuilt, 1))
autoplot(yr)
ggsave('assets/img/yrBuilt.png', height=2, width = 3)
```


Last week's homework - building 'cut-off date'

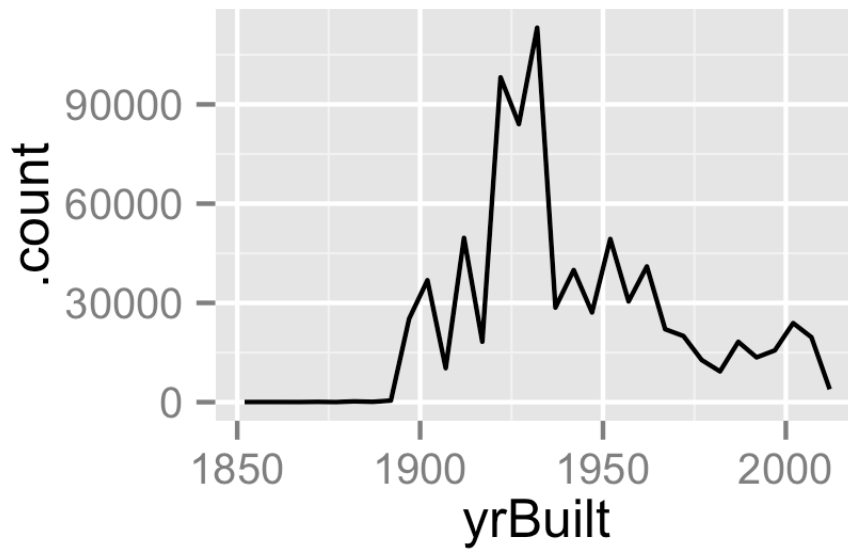


Last week's homework - building 'cut-off date'

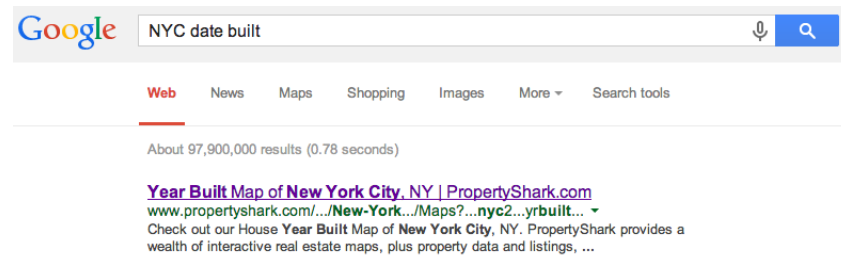
- Oh no! Our data stinks!
- Clearly estimated when year \leq 1980
- Can we proceed? Depends on context of questions

```
yr <- condense(bin(yrBuilt, 5))  
autoplot(yr)  
ggsave('assets/img/yrBuilt5.png', height=2, width = 3)
```

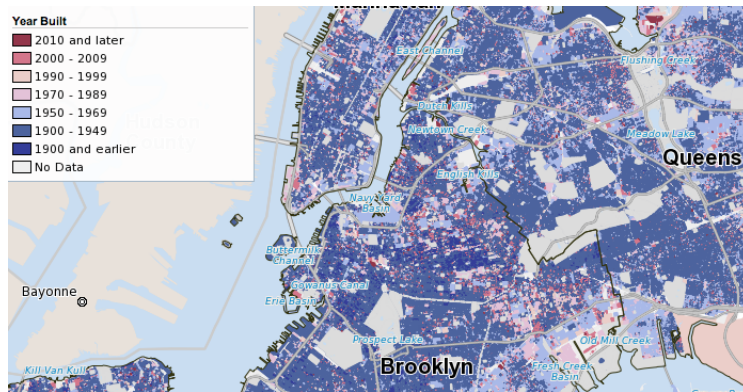
Last week's homework - building 'cut-off date'



Last week's homework - double check suspicious data



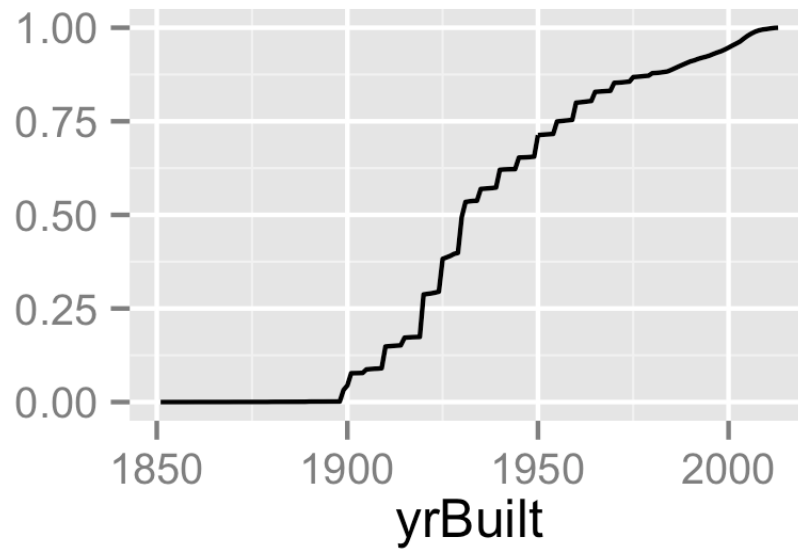
Last week's homework - double check suspicious data



Last week's homework - finding cut-offs

```
yr2 <- condense(bin(yrBuilt, 1))
total <- sum(yr2$.count)
yr2$perc_built <- cumsum(yr2$.count)/total
ggplot(yr2, aes(x= yrBuilt, y=perc_built)) + geom_line() + ylab('')
ggsave('assets/img/cumcurve.png', height=2, width = 3, dpi = 100)
```

Last week's homework - finding cut-offs

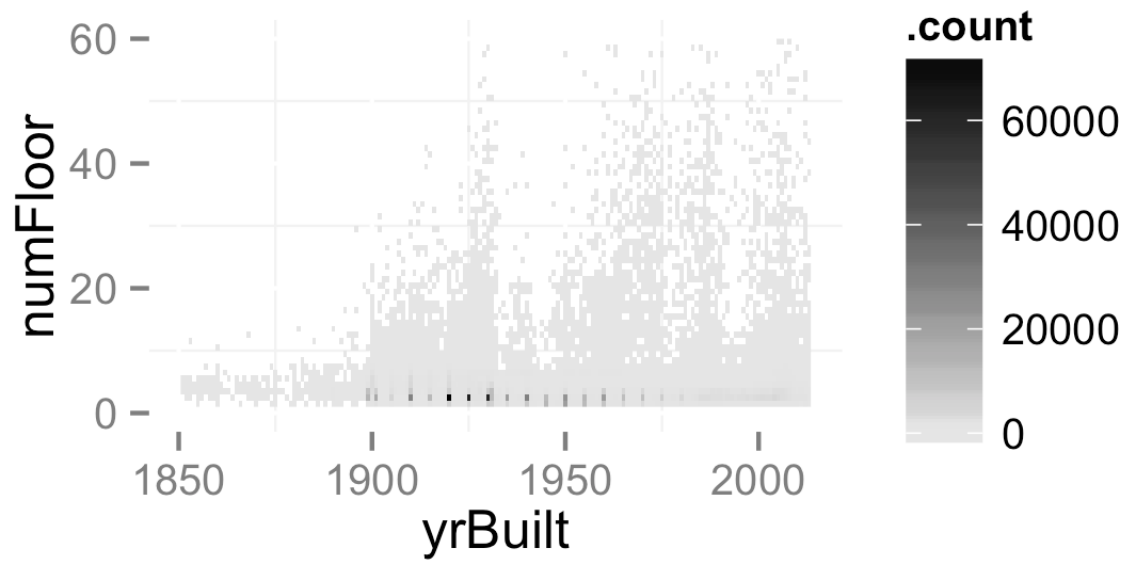


Last week's homework - cut-offs by group

- Similar question to previous, only with groups

```
fldrVsYr <- condense(bin(yrBuilt, 1), bin(numFloor, 1))
p <- autoplot(fldrVsYr) + theme(panel.background=element_rect(fill='white')) + ylim(0,60)
p + scale_fill_gradient(limits= c(1,100000),
                        low='grey',
                        high='blue',
                        trans="log",
                        breaks=myBreaks)
ggsave('assets/img/yr_v_fldr.png', height=2, width = 4, dpi = 300)
```

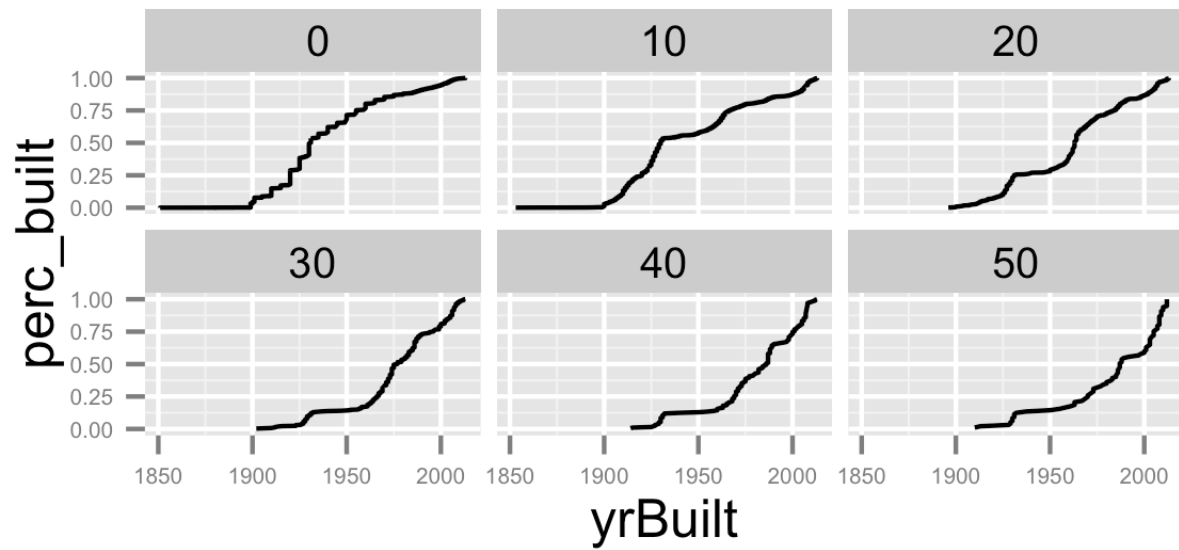

Last week's homework - cut-offs by group



Last week's homework - cut-offs by group

```
flrVsYr$stories <- 10*trunc(flrVsYr$numFloor/10)
flrVsYr$stories <- sapply(flrVsYr$stories, min, 50)
flrVsYr$num_built <- ave(flrVsYr$.count, flrVsYr$stories, FUN=cumsum)
flrVsYr$perc_built <- flrVsYr$num_built / ave(flrVsYr$.count, flrVsYr$stories, FUN=sum)
ggplot(flrVsYr, aes(x=yrBuilt, y=perc_built, group=stories)) + geom_line() +
facet_wrap(~ stories) + theme(axis.text=element_text(size=5))
ggsave('assets/img/stories.png', height=2, width = 4, dpi = 300)
```

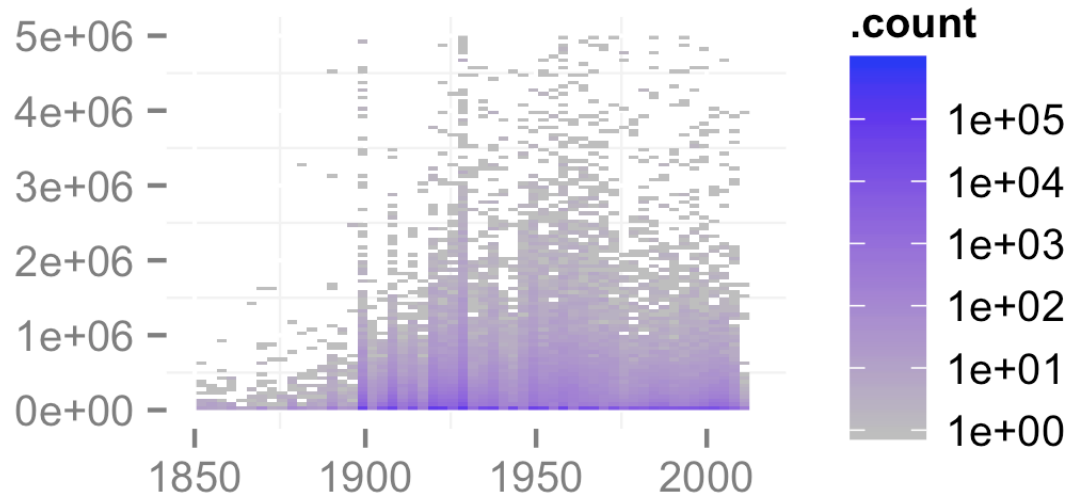
Last week's homework - cut-offs by group



Last week's homework - cut-offs by group

```
assessVsYr <- condense(bin(yrBuilt[assessTot < 5000000],3), bin(valPerFloor[assessTot < 5000000], 5000))
p <- autoplot(assessVsYr) + theme(panel.background=element_rect(fill='white')) + xlim(1850,2015) + xlab('')
p + scale_fill_gradient(limits= c(1,1000000),
                        low='grey',
                        high='blue',
                        trans="log",
                        breaks=myBreaks)
ggsave('assets/img/assess.png', height=2, width = 4, dpi = 300)
```

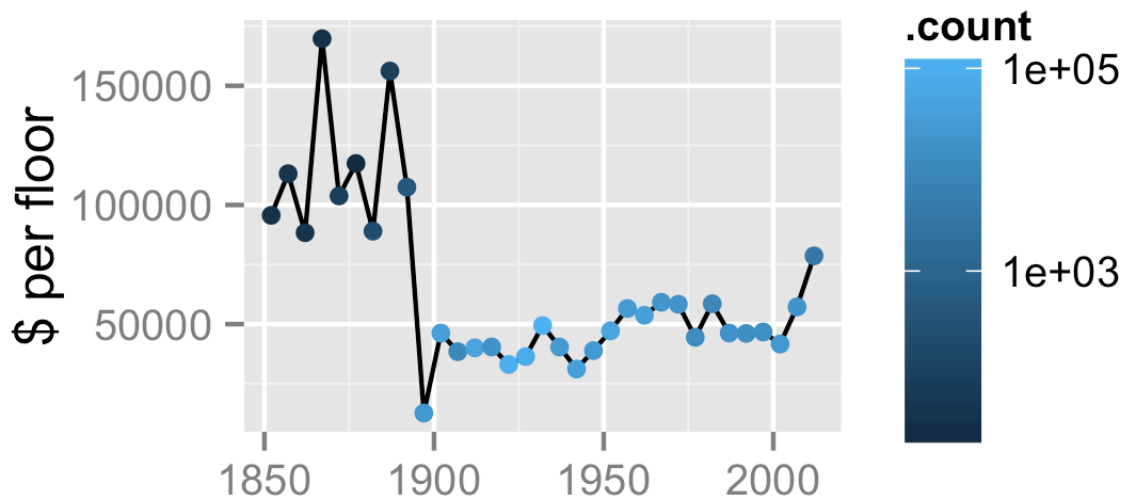
Last week's homework - cut-offs by group



Last week's homework - wartime

```
flrVal <- condense(bin(yrBuilt[assessTot < 5000000],5), z =valPerFloor[assessTot < 5000000] )  
autoplot(flrVal) +xlab('') + ylab('$ per floor')  
ggsave('assets/img/flrVal.png', height=2, width = 4, dpi = 300)
```

Last week's homework - cut-offs by group



Moving to interactive graphics

- Painful to create multiple different charts for investigation
- We are moving away from statistics here and towards design
- `ggplot2` is doing this too- moving to `ggvis`

Moving to interactive graphics - terms

- 'Server-side': what happens on the server (back-end, database)
- 'Client-side': what happens on the user's computer (Browser, JS)
- Scalable: how a site handles many visits at once

googleVis

- Often known as 'Hans Robling style charts'
- Interface to Google Chart API
- Compiles to HTML/Javascript
- Great demo by running `demo('googleVis')`

googleVis - advantages

- Fast and Easy
- Output is interactive and in-browser
- Very simple to combine charts

googleVis - drawbacks

- Warning: not all google charts are secure
- Need web access to run
- Risky: Google has history of deprecating projects

shiny

- Sponsored by RStudio: similar to googleVis but has a server-side component
- Can be integrated with googleVis
- Open Source

Next module's reading

- All on Blackboard

Next module's assignment

- CDC Wonder Data <http://wonder.cdc.gov/controller/datarequest/D76>