Introduction to Visualizations with R
for Exploratory Data Analysis

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

March 3, 2016

Tutorial Outline

Preliminaries
Introduction to R
Visualizations
Working with R

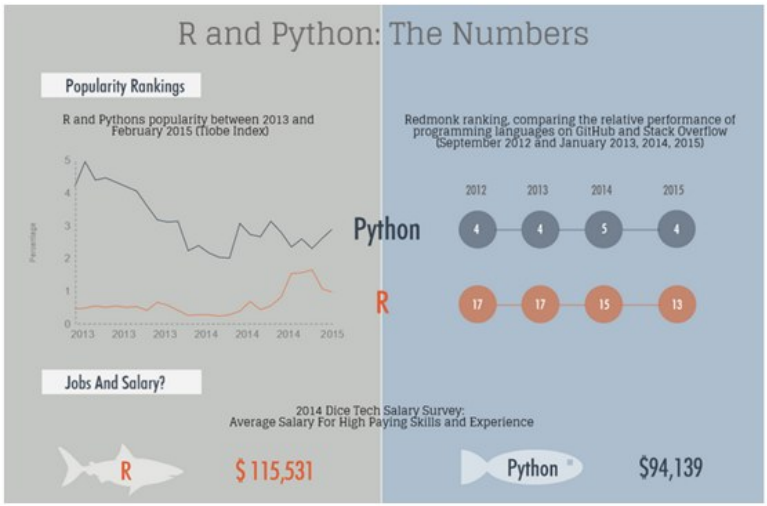
Part I

Preliminaries

Kukuyeva Consulting

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]

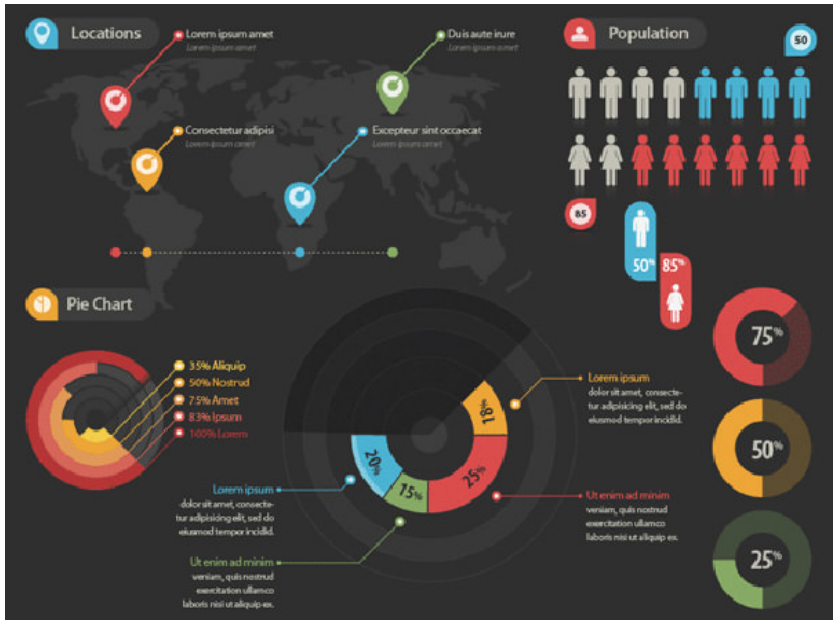
Why R?

- 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).

Approach 1: Infographic

Inforgraphic

- Manually created
- Customized to a data set
- Art



[10]

Approach 2: Data Visualization I

Data Visualization

- Automated via software
- Reproducible for a different data set
- (Less? of an) art

There are two types of data visualization:

1. Exploratory

- Capture uncertainty in the data
- Explore potential relationships/trends/missing patterns (or absence thereof)

Approach 2: Data Visualization II

2. Explanatory

- Convey key information

[8]

Tips for visualizations

- Know your audience
- *15 second rule:* if your audience won't be able to understand the graphic in 15 seconds, simplify
- Layout [9] and font choices
- Color choices
  - Available colors in R:
    - `http://research.stowers-institute.org/efg/R/Color/Chart/`
  - Color blindness simulation [11]:
    - `http://vischeck.com`
    - `http://rsb.info.nih.gov/ij/`
- Add text/labels to figure

Kukuyeva Consulting

Part II

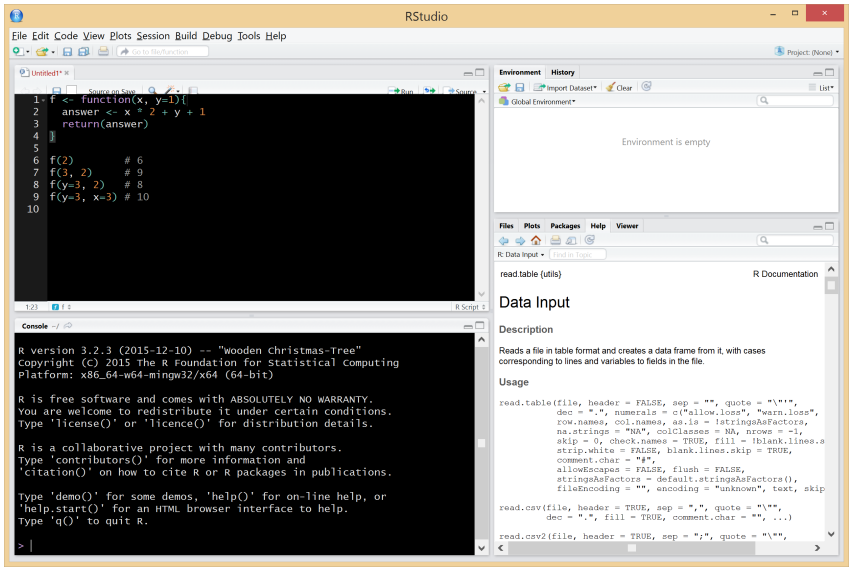
Introduction to R

Kukuyeva Consulting

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).



Kukuyeva Consulting

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."

Kukuyeva Consulting

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.)

Functions in R

Function by example

```
1   f <- function(x, y=1){
2           answer <- x * 2 + y + 1
3           return(answer)
4   }
5
6   f(2)          # 6
7   f(3, 2)       # 9
8   f(y=3, 2)     # 8
9   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

```
1  x = 3
2  y = 5
3  z = c(x, y)
4  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
```

Kukuyeva Consulting

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

Kukuyeva Consulting

Reading Data from File II

```
1  filepath = "http://www.ats.ucla.edu/stat/data/test
       _missing_comma.txt"
2  ### Other valid paths:
3  # filepath = "C:/Documents/test_missing_comma.txt"
4  # filepath = "./test_missing_comma.txt"
5
6  df <- read.table(
7            file = filepath,
8            header = TRUE,
9            sep = ","
10           )
```

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.)

```
1  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()`:

```
1 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()`:

```
1 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()`:

```
1  names(df)
```

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

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

```
1  dim(df)
```

```
[1] 8 6
```

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

```
1  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()`:

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

```
[1] 0 1
```

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

```
1 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:

```
1  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()`:

```
1 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
```

Kukuyeva Consulting

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()`:

```
1 install.packages("lattice")
2 install.packages("ggplot2")
3 # OR
4 install.packages( c("lattice", "ggplot2") )
```

Kukuyeva Consulting

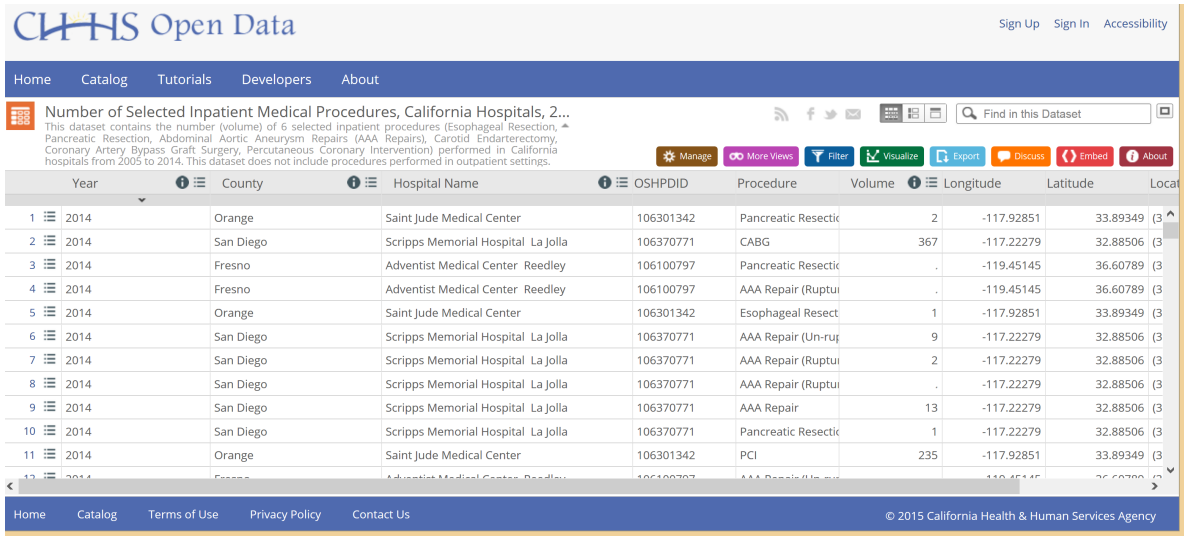
Part III

Visualizations

Data Set for Tutorial I

Data set for this tutorial comes from State of California. Please export it to CSV from:

`https://chhs.data.ca.gov/Healthcare/Number-of-Selected-Inpatient-Medical-Procedures-Ca/mdt8-gwyw`



Kukuyeva Consulting

Data Set for Tutorial II

To read it into R, type:

```
df <- read.table(
  "Number_of_Selected_Inpatient_Medical_Procedures__California_Hospitals__2005-2014.csv",
    sep = ",",
    header=TRUE,
    stringsAsFactors = FALSE
    )
```

Check that the file was read-in correctly:

```
dim(df) # 45,438 rows and 9 columns
```

Rename the first variable to make it easier to work with:

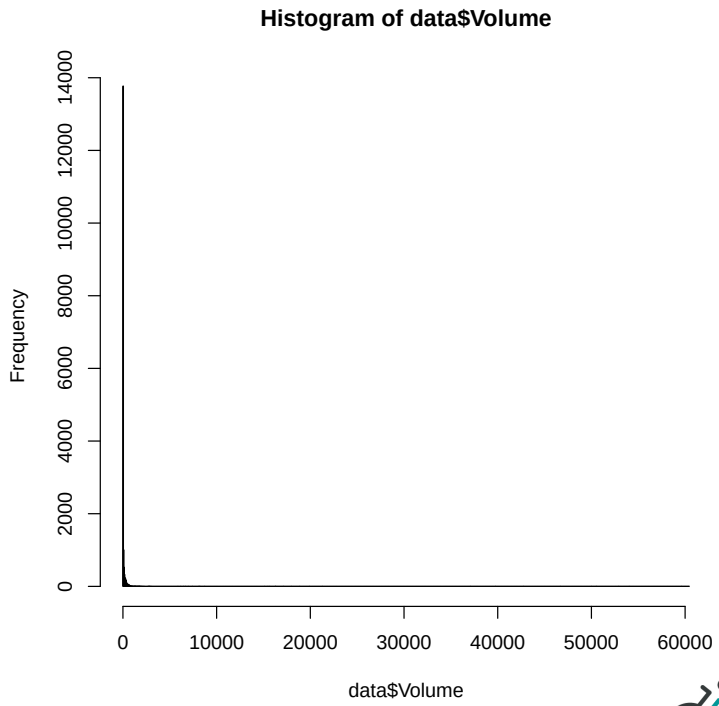
```
names(df)[1] = 'Year'
```

Kukuyeva Consulting

Histogram I

To see the counts of values that a variable has (i.e. its distribution), use `hist()`:

```
1  hist(
2     df$Volume,
3     breaks=1000
4     )
5  # Is anything 'off'?
```



Histogram II

Let's examine the potential outlier:

```
1  ### Step 0: Load required packages
2  library(dplyr)
3
4  ### Step 1: Examine entry when volume is
       largest in data set
5  df %>%
6     filter( Volume == max(Volume, na.rm=TRUE) )
```

```
       Year    County Hospital.Name OSHPDID Procedure Volume Longitude Latitude Location
41783  2005 STATEWIDE     STATEWIDE      NA       PCI  60418        NA       NA
```

Entry seems to be an aggregate number, not county/year level.

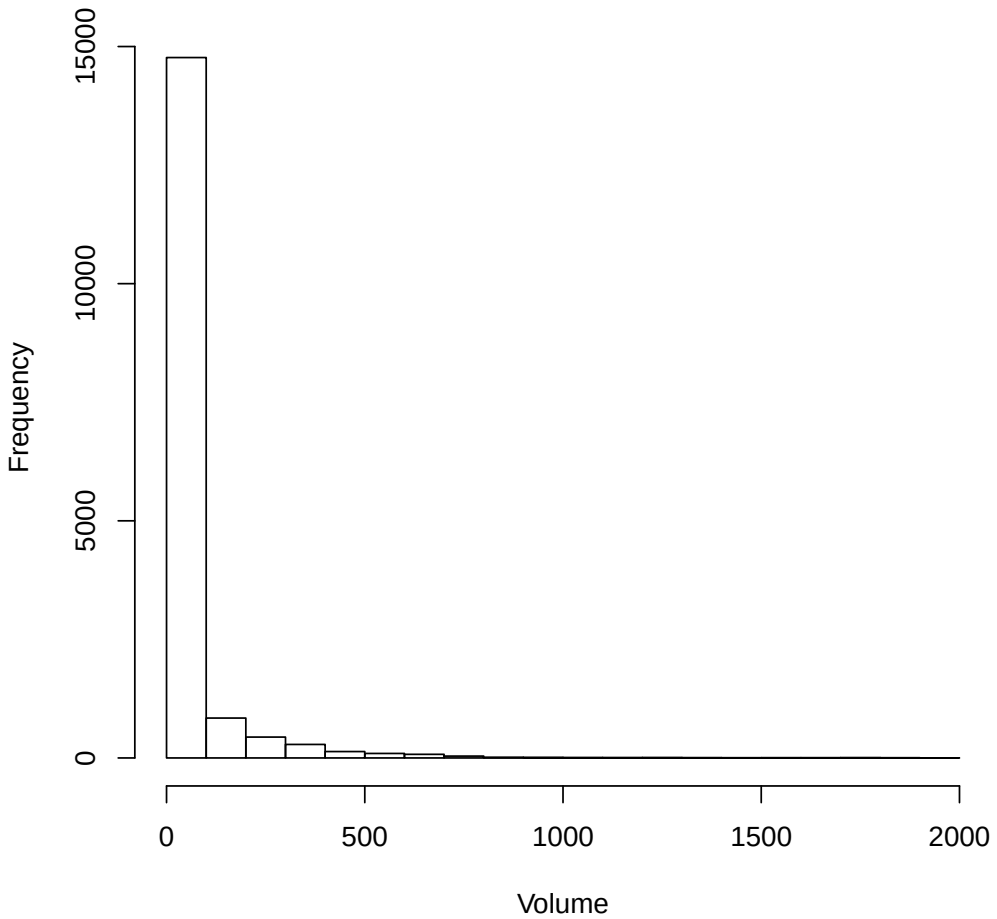
Histogram III

To remove any extraneous data points:

```
1  ### Step 0: Load required packages
2  library(dplyr)
3
4  ### Step 1: Remove any 'statewide' counts
5  data_clean <- df %>%
6     filter( County != "STATEWIDE" )
7
8  ### Step 2: Visualize
9  hist(data_clean$Volume,
10    xlab="Volume",
11    main=""
12    )
13 # Is anything 'off' now? (Exercise.)
```

Kukuyeva Consulting

Histogram IV



Side-by-side Histogram I

To compare distributions of two variables side-by-side, use `beanplot()`:

```
### Step 0: Load required packages
library(beanplot)

### Step 1: Subset data set to LA and SF
data_LA_SF <- data_clean %>%
  filter(
    (County == "Los Angeles") |
    (County == "San Francisco")
  )
```

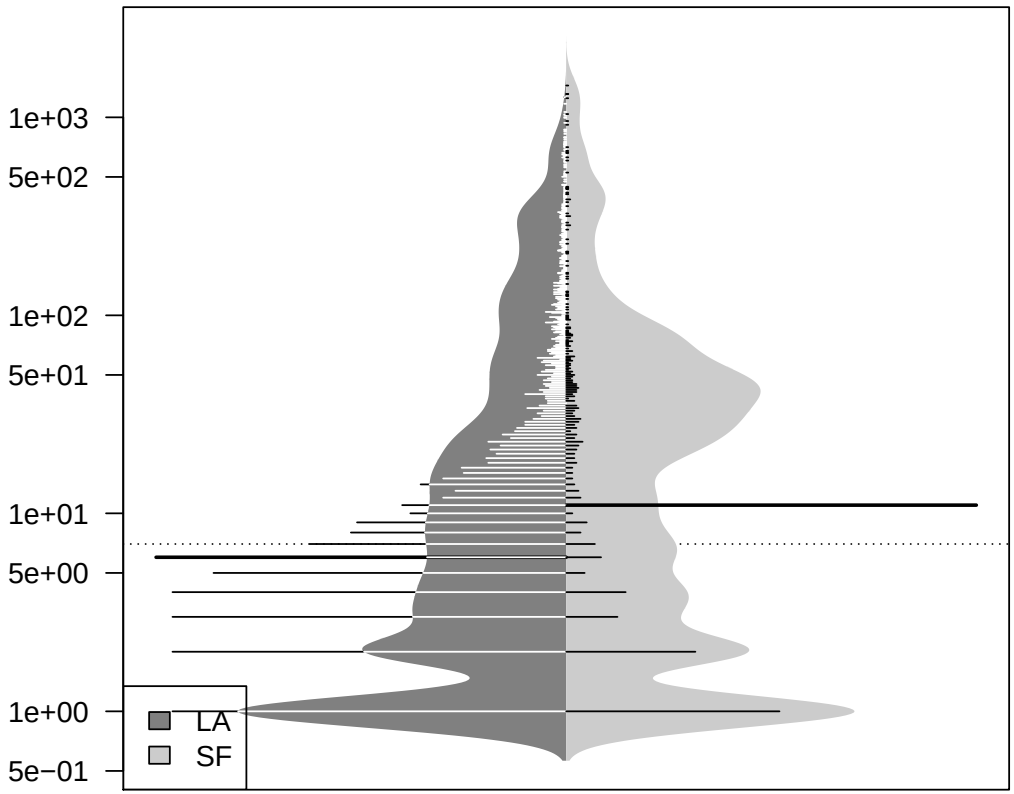
Side-by-side Histogram II

```
### Step 3: Visualize
op <- par(las=2)    # orient y-axis labels
beanplot(
   Volume ~ as.factor(County),
   data = data_LA_SF,
   xlab = "",
   log = "y",
   side = "both",
   col =list( c(grey(0.5), "white"), grey(0.8) ),
   border = NA,
   overalline = "median",
   ll = 0.005,
   show.names=FALSE
   )
```

Side-by-side Histogram III

```
1 legend(
2    x = "bottomleft",
3    fill=c( grey(0.5), grey(0.8) ),
4    legend=c( "LA", "SF" )
5    )
6 par(op)
```

Side-by-side Histogram IV

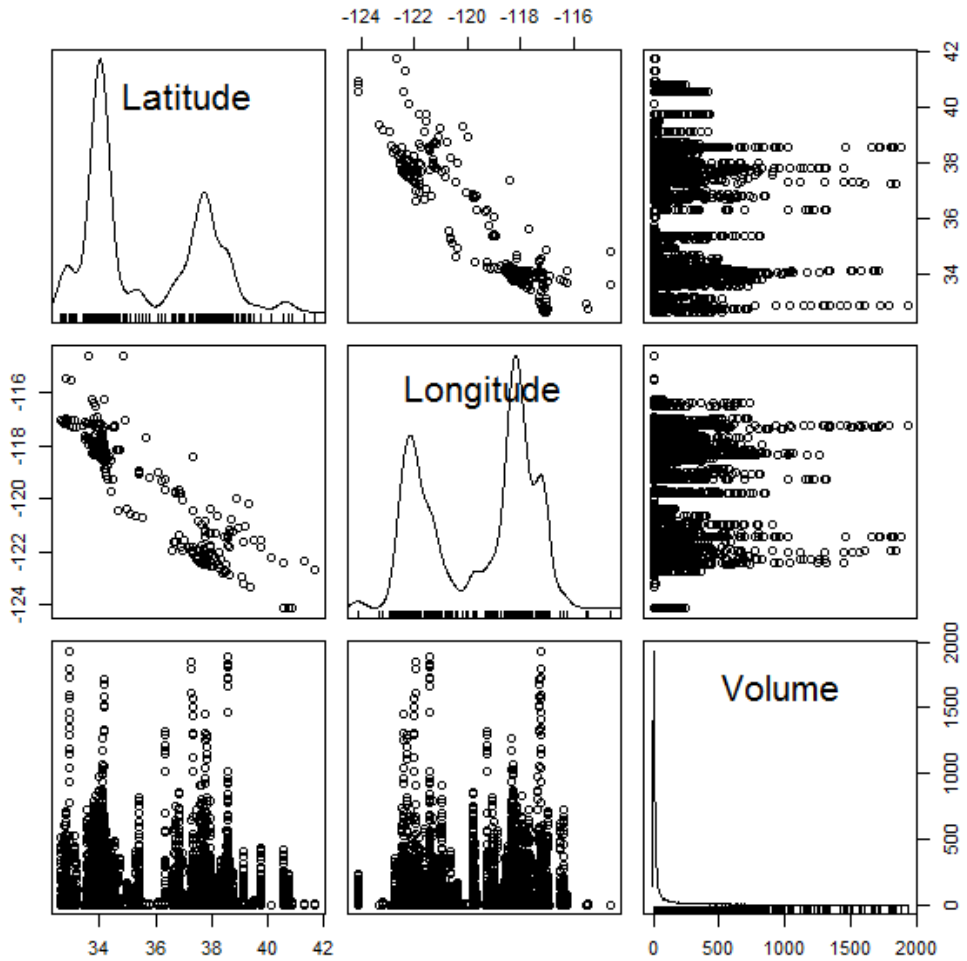


Scatterplot (v0.1) I

To compare a few variables in the data set at the same time, use `scatterplotMatrix()`:

```
# Load package 'car' to use scatterplotMatrix():
library(car)
scatterplotMatrix(
   x = df[, c("Latitude", "Longitude", "Volume")],
   smoother = FALSE,
   reg.line = FALSE
   )
```

Scatterplot (v0.1) II

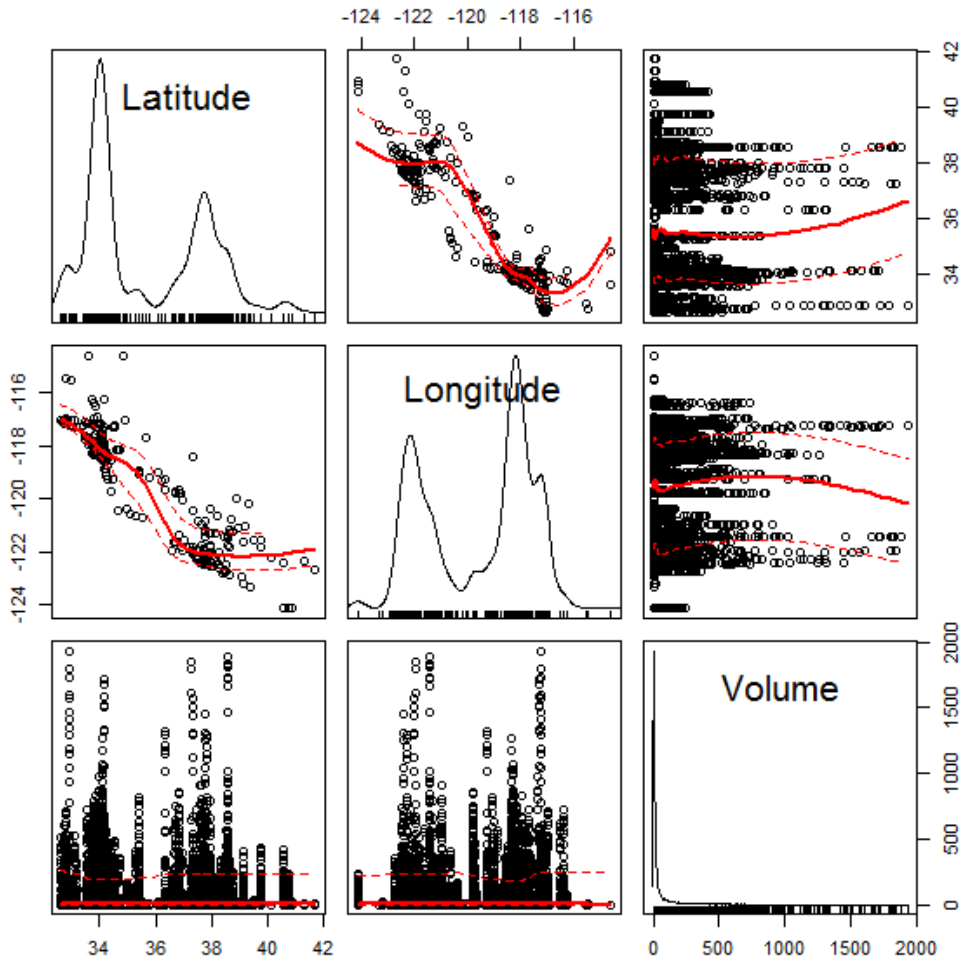


Scatterplot (v0.2) I

To compare a few variables in the data set and their trends at the same time, modify arguments of `scatterplotMatrix()`:

```
?scatterplotMatrix # for documentation
library(car)
scatterplotMatrix(
   x = df[, c("Latitude", "Longitude", "Volume")],
   reg.line = FALSE
   )
```

Scatterplot (v0.2) II



Scatterplot (v0.3) I

To improve the color scheme, modify arguments of `scatterplotMatrix()`.

```
library(car)
scatterplotMatrix(
  x = df[, c("Latitude", "Longitude", "Volume")],
  reg.line = FALSE,
  col=c(3,
    "orangered",
    rgb(176/256, 196/256, 222/256, alpha=0.5)
    ),
  pch=19,
  lwd=3
  )
```

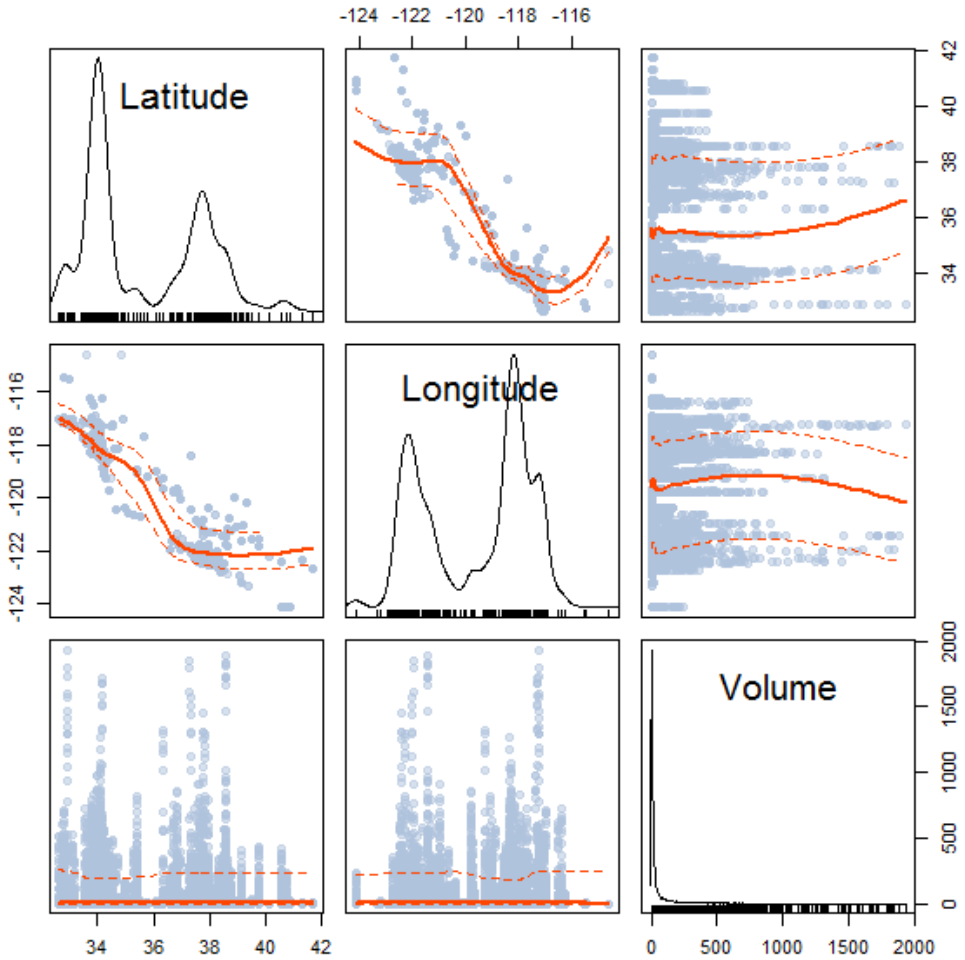
Scatterplot (v0.3) II

You can specify a color in R via:

- Number (e.g. 1 = black, 2 = red, etc.)
- Name (e.g. "black", "red", "dodgerblue")
- RGB (e.g. c(0,0,0) = black, c(1,0,0) = red, etc.)
- Please see `http://research.stowers-institute.org/efg/R/Color/Chart/` for color references.

Note the order of color arguments.

Scatterplot (v0.3) III



Exercise I

In the healthcare data set, after we removed statewide patient admissions, is the largest volume of patients now seen at a Californian hospital a potential outlier? How do you know?

Kukuyeva Consulting

Univariate Time Series Plots I

One way to plot one variable at a time (over time), is via `plot()`:

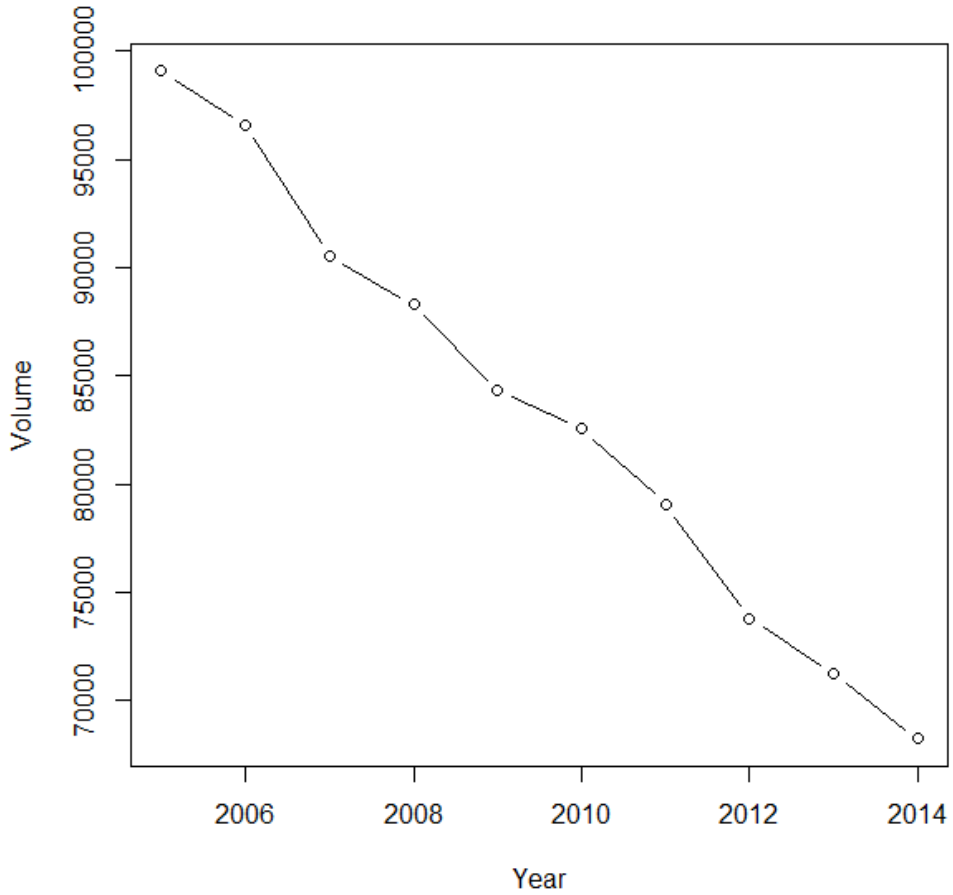
```
### Step 0: Load any required packages:
library(dplyr)

### Step 1: Get yearly counts across hospitals
    and procedures
df_summary <- data_clean %>%
  group_by(Year) %>%
  summarise(Total.for.Year = sum(Volume, na.
      rm=TRUE))
```

Univariate Time Series Plots II

```
### Step 3: Visualize
plot(
   x=df_summary$Year,
   y=df_summary$Total.for.Year,
   xlab="Year",
   ylab="Volume",
   type="b",
   main="Yearly Volume for 6 procedures in CA"
   )
```

What are some limitations of the data set (and hence plot)?

Univariate Time Series Plots III



Multivariate Time Series Plots: Approach 1 I

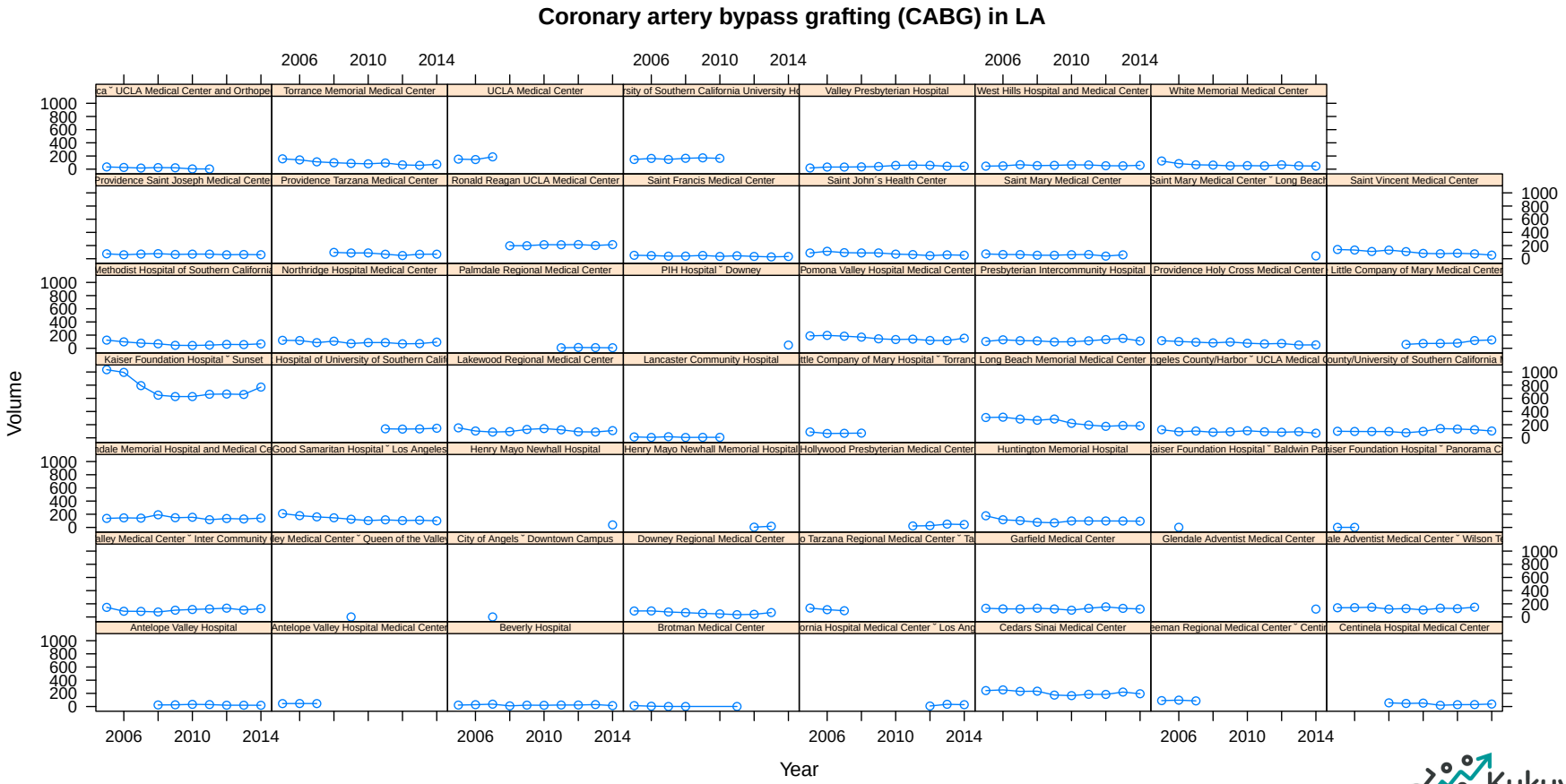
To plot more than one variable at a time, over time, you can ttfamily xyplot()[5]:

```
### Step 0: Load any required packages
library(dplyr)
library(lattice)

### Step 1: Get data for one procedure
df_LA_CABG <- df %>%
  filter(
    (County == "Los Angeles") &
    (Procedure == "CABG") &
    (Volume > 0)
    )
```

Multivariate Time Series Plots: Approach 1 II

```
1  ### Step 2: Visualize
2  xyplot( Volume ~ Year | Hospital.Name,
3    data=df_LA_CABG,
4    par.strip.text=list(cex=0.5),
5    type="b",
6    main="LA Coronary artery bypass grafting (
         CABG)"
7    )
```

Multivariate Time Series Plots: Approach 1 III



Multivariate Time Series Plots: Approach 2 I

Another way to plot more than one variable at a time, is via ggplot():

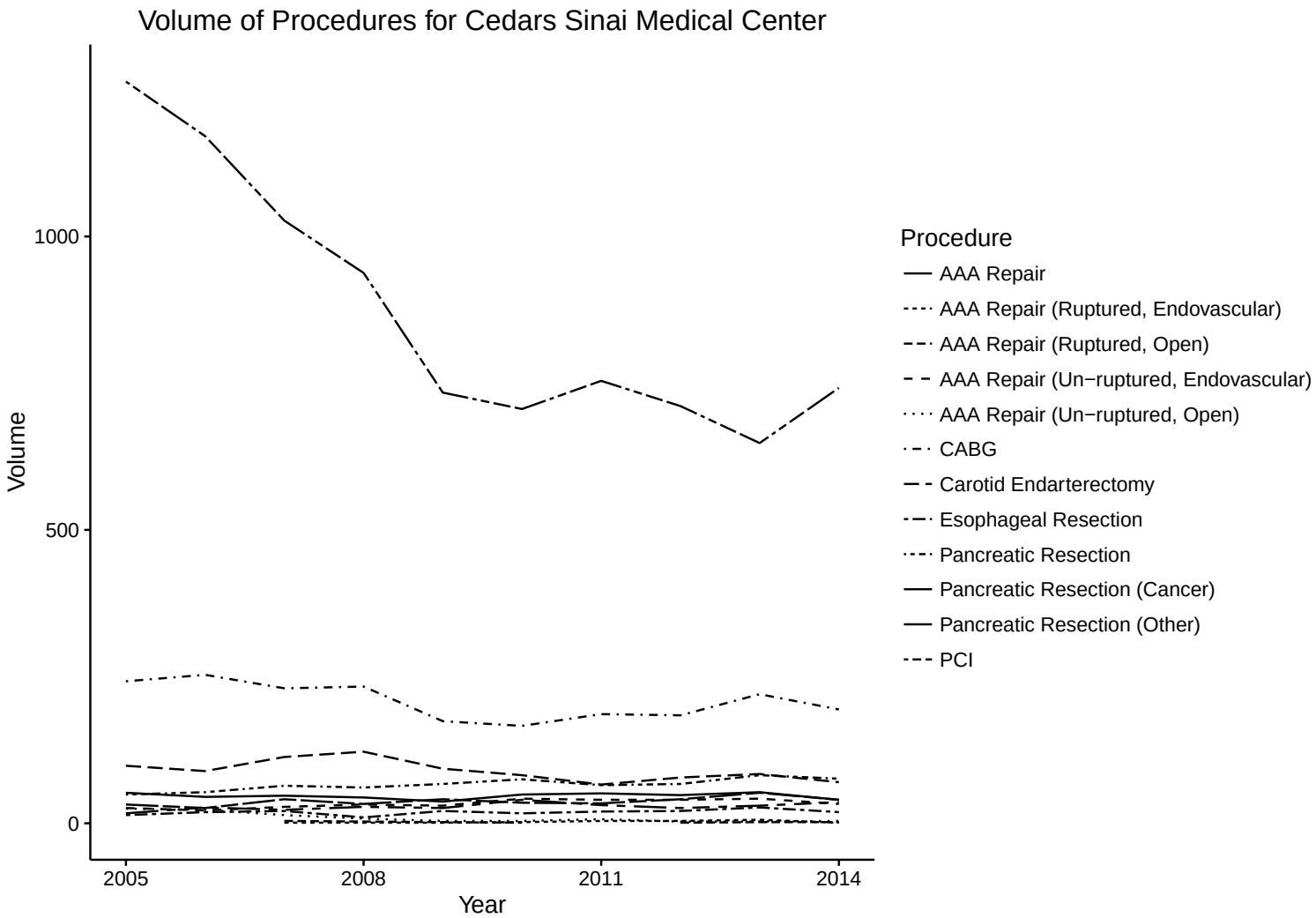
```
### Step 0: Load any required packages:
library(dplyr)
library(ggplot2)

### Step 1: Get data for one hospital:
df_Cedars <- data_clean %>%
  filter( Hospital.Name == 'Cedars Sinai
      Medical Center')
```

Multivariate Time Series Plots: Approach 2 II

```
### Step 2: Visualize
ggplot( data = df_Cedars ) +
  geom_line( aes(
    x = Year,
    y = Volume,
    linetype = Procedure) ) +
  scale_x_continuous( breaks=seq(
    from=2005,
    to=2014,
    by=3) ) +
  theme_classic() +
  ggtitle( "Volume of Procedures for Cedars
     Sinai Medical Center" )
```

Kukuyeva Consulting

Multivariate Time Series Plots: Approach 2 III



Exercise II

In the healthcare data set, recently what seems to be the most popular procedure in CA?

Hints:

- Should we aggregate the data? If so, to what level?
- What's one way to examine trends in the data?

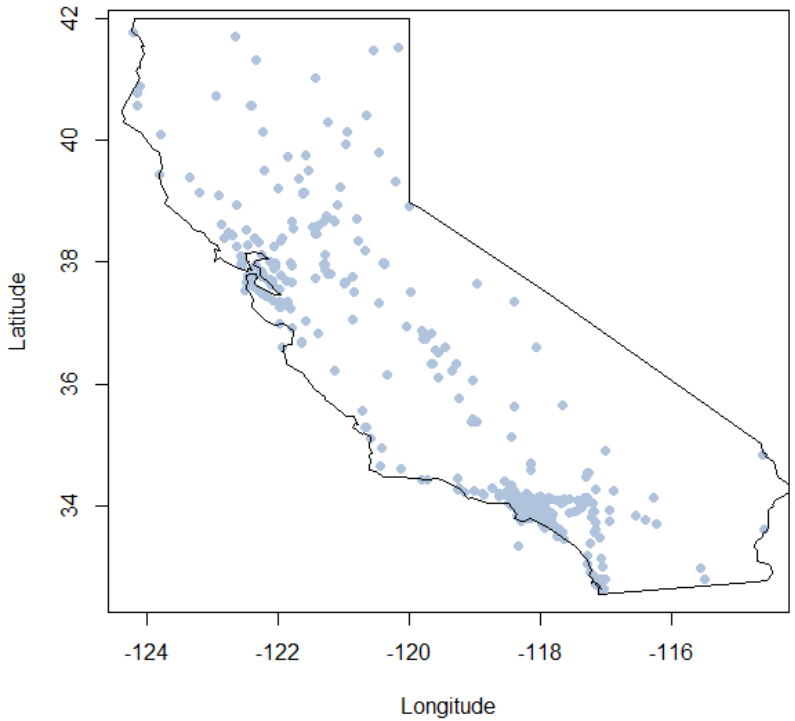
Kukuyeva Consulting

Geographic Maps (v0.1) I

To overlay a map to a plot containing latitude and longitude, load the package `maps`:

```
1  library(maps)
2  plot(
3    x = data_clean$Longitude,
4    y = data_clean$Latitude,
5    xlab = "Longitude",
6    ylab = "Latitude",
7    pch = 16,
8    col = rgb(176/256, 196/256, 222/256, alpha=0.5)
9  )
10 map("state", "california", add=TRUE)
```

Geographic Maps (v0.1) II



What are some limitations of the plot?

Geographic Maps (v0.2) I

To include information on volume of procedure, use cexargument of the `plot()` function:

```
1  ### Step 1: Aggregate data to be at hospital level:
2  df_hosp <- data_clean %>%
3    group_by( Latitude, Longitude ) %>%
4    summarise( Total.for.Hospital = sum(Volume, na.rm=
         TRUE) )
5
6  ### Step 2: Recode 'Volume' to have 2 categories
       only: high ( >100 cases ) and low ( <= 100 cases
       ):
7  df_hosp$volume_ind <- ifelse(
8   test = df_hosp$Total.for.Hospital > 100,
9   yes = 2,
10  no = 1 )
```

Geographic Maps (v0.2) II

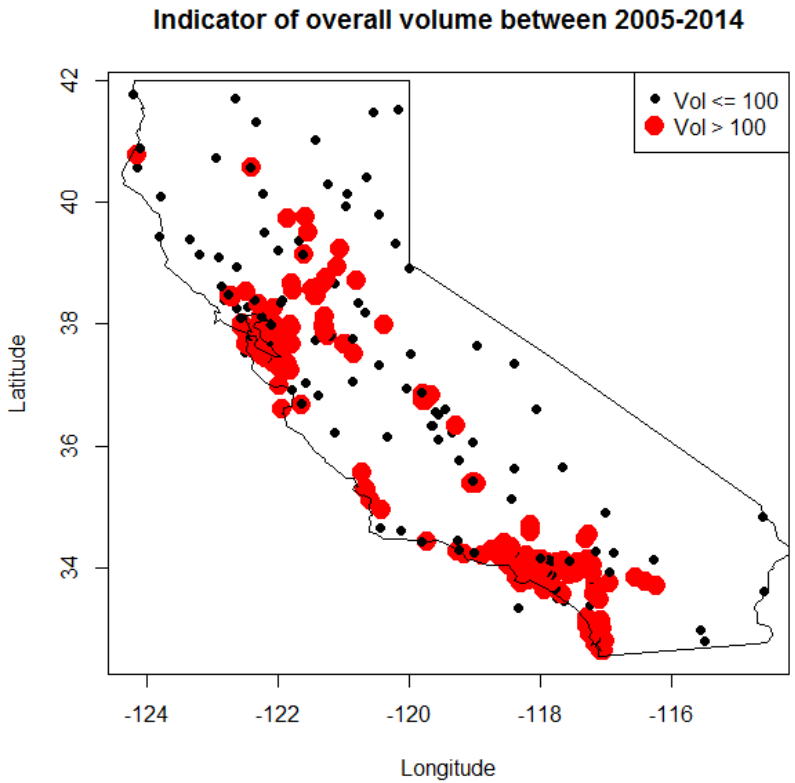
```
1  ### Step 3: Plot
2  plot(
3    x = df_hosp$Longitude,
4    y = df_hosp$Latitude,
5    pch = 19,
6    cex = df_hosp$volume_ind,
7    col = df_hosp$volume_ind,
8    xlab = "Longitude",
9    ylab = "Latitude",
10   main="Indicator of overall volume between
         2005-2014"
11 )
12 map("state", "california", add=TRUE)
```

Kukuyeva Consulting

Geographic Maps (v0.2) III

```
### Step 4: Add legend
legend("topright",
 pch = 19,
 pt.cex = 1:2,
 col = 1:2,
 c("Vol <= 100", "Vol > 100")
)
```

Geographic Maps (v0.2) IV



What are some limitations of the plot? (Exercise.)

Kukuyeva Consulting

Geographic Maps (v0.3) I

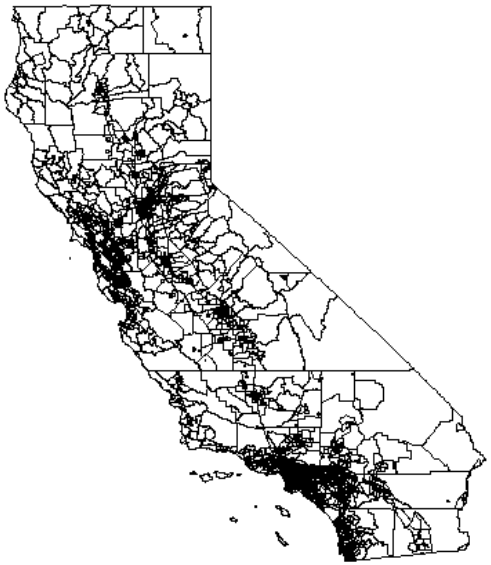
Another way to visualize geographical information is via shapefiles. Please download shapefiles for California for 2014 from `https://www.census.gov/geo/maps-data/data/cbf/cbf_tracts.html` and unzip the folder.

```
### Step 1: Load in required packages
library(sp)
library(rgdal)
```

Geographic Maps (v0.3) II

```
### Step 2: Read in shapefile and add latitude
    and longitude coordinates to it:
tracts = spTransform(
        readOGR(
                file.path("cb_2014_06_tract
                    _500k"),
                layer = "cb_2014_06_tract_500k
                    "
                ),
        CRS("+proj=longlat +datum=WGS84")
        )

### Step 3: Visualize
plot(tracts)
```

Geographic Maps (v0.3) III



Plot of the shapefile tracts.

Geographic Maps (v0.3) IV

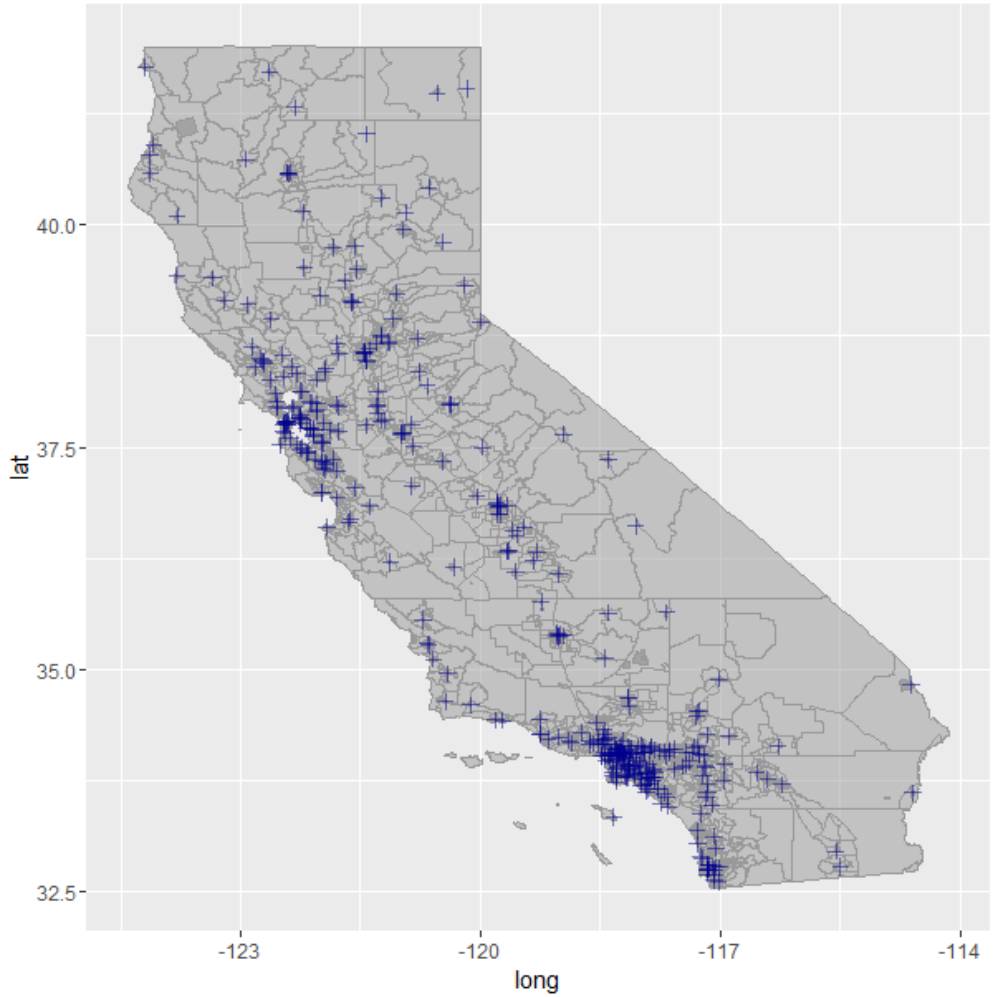
Now let's add the locations of hospitals to the map [12]:

```
### Step 1: Load required packages
library("rgdal")
library("maptools")
library("ggplot2")
library("plyr")

### Step 2: Preprocess data ahead of plotting
tracts@data$id = rownames(tracts@data)
tracts.points = fortify(tracts, region="id")
tracts.df = join(
 tracts.points,
 tracts@data,
 by="id"
 )
```

Geographic Maps (v0.3) V

```
### Step 3: Visualize
ggplot() +
  geom_polygon(data = tracts.df,
               aes(x = long, y = lat, group =
                   group),
               fill = grey(0.6),
               color = grey(0.6),
               alpha = 0.5
               ) +
  geom_point(data = df_hosp,
             aes(x = Longitude, y = Latitude),
             color = "blue4",
             alpha = 0.5,
             shape = 3,
             size = 2 )
```

Geographic Maps (v0.3) VI

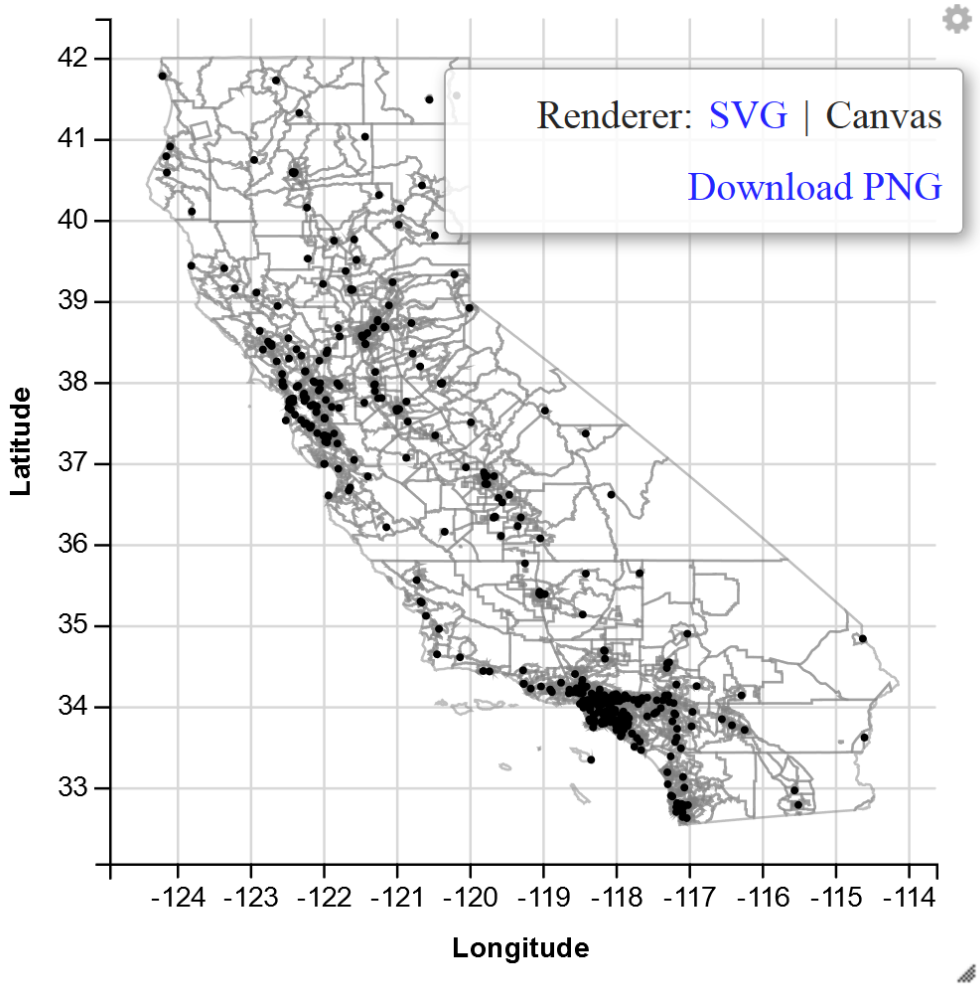


Geographical Plots via 'ggvis' I

One way to make high resolution and interactive images is via 'ggvis' [13]:

```
# Please note, it may take some time to load as the
    shapefile is high resolution:
library(ggvis)
tracts.points %>%
  ggvis(~long, ~lat) %>%
  group_by(group, id) %>%
  layer_paths(strokeOpacity:=0.5, stroke:=grey(0.5))
      %>%
  layer_points(data=df_hosp, x=~Longitude, y=
      ~Latitude, size:=8) %>%
  hide_legend("fill") %>%
  set_options(width=400, height=400, keep_aspect=
      TRUE)
```

Geographical Plots via 'ggvis' II

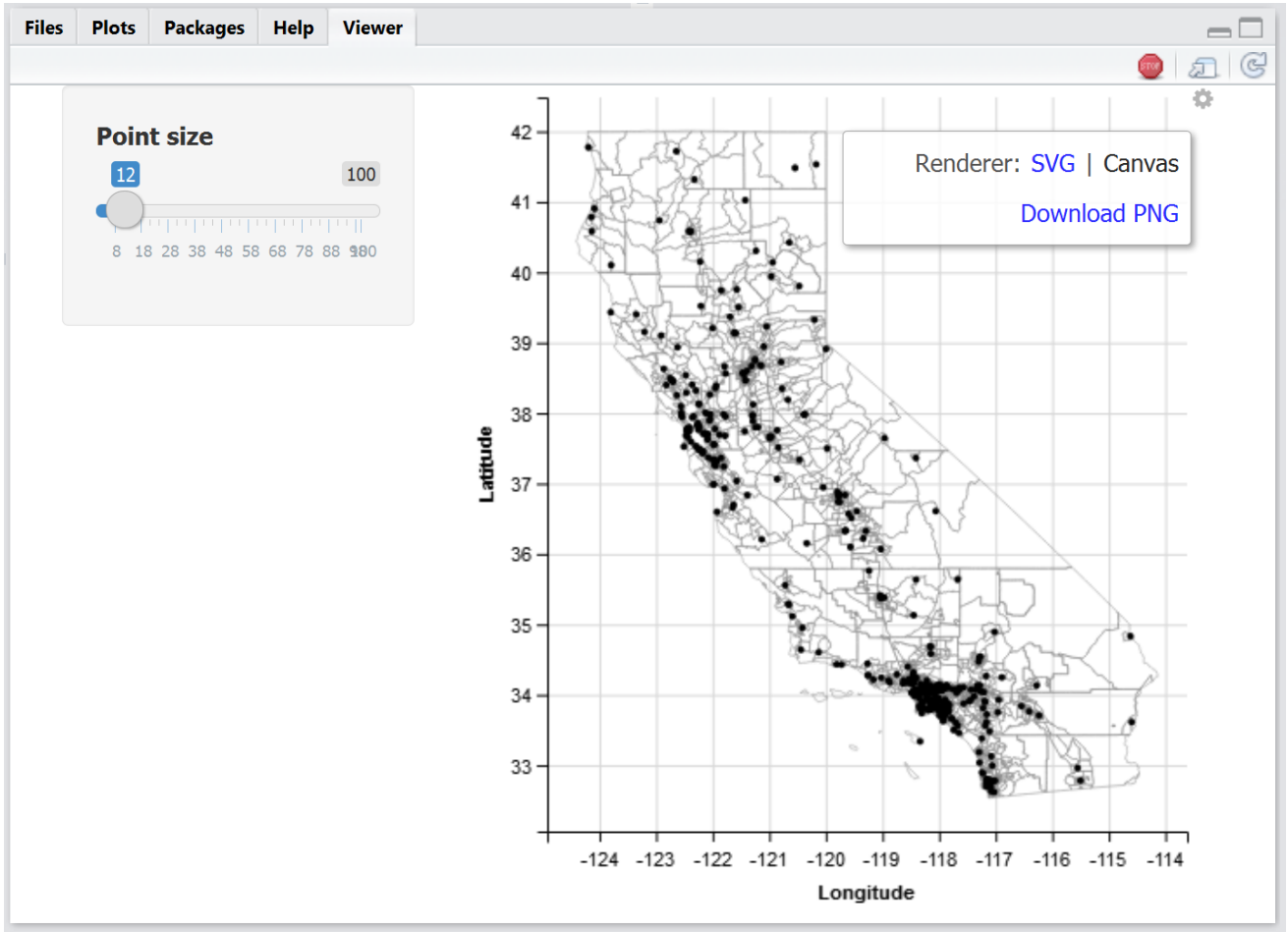


Geographical Plots via 'ggvis' III

Now we can add (basic) interactivity by resizing the hospital locations:

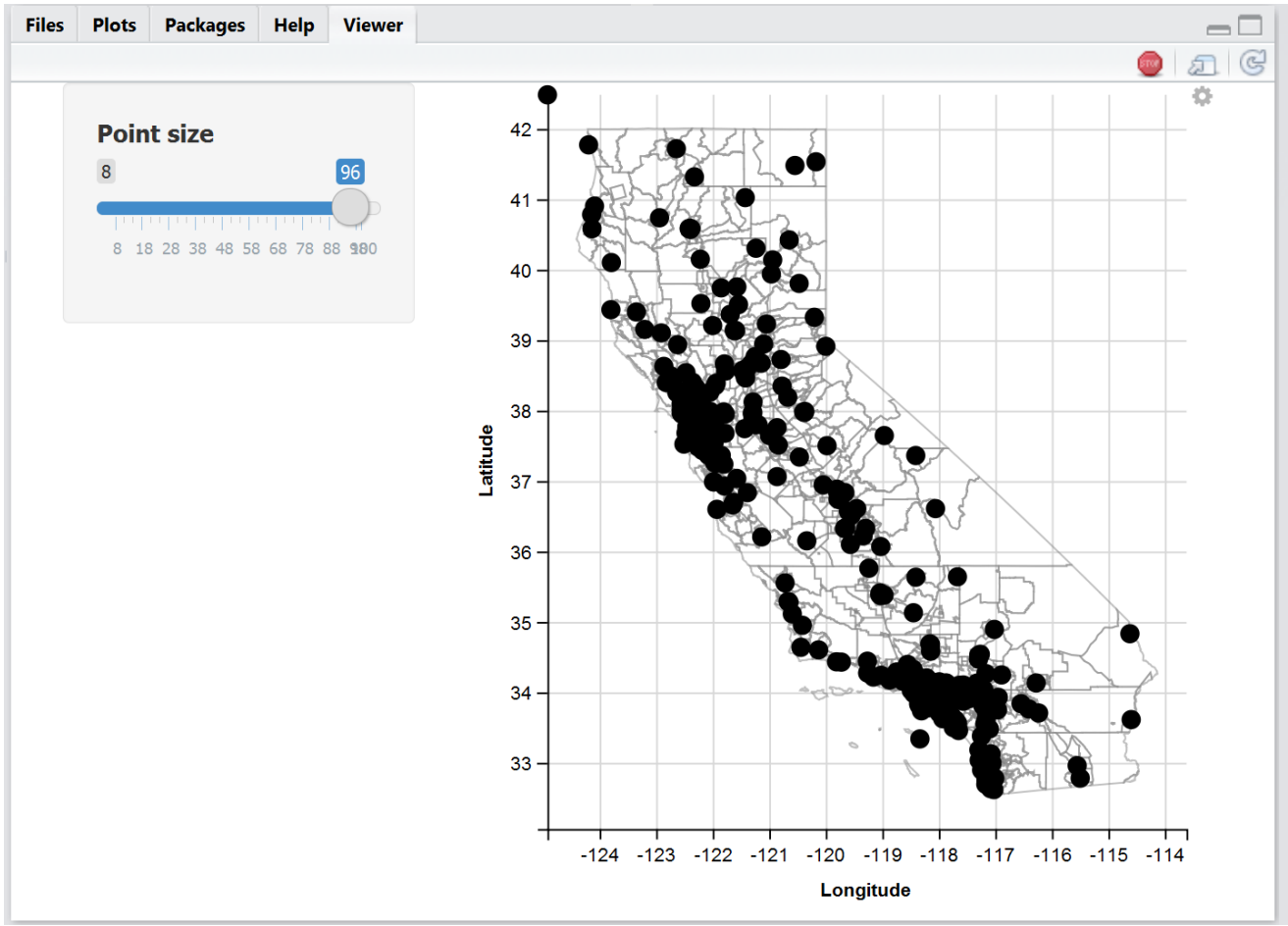
```
library(ggvis)
tracts.points %>%
  ggvis(~long, ~lat) %>%
  group_by(group, id) %>%
  layer_paths(
    strokeOpacity:=0.5,
    stroke:=grey(0.5)) %>%
  layer_points(
    data=df_hosp,
    x=~Longitude,
    y=~Latitude,
    size:=input_slider(8, 100, value = 12, label='Point size')) %>%
  hide_legend("fill") %>%
  set_options(width=400,
    height=400,
    keep_aspect=TRUE)
```

Geographical Plots via 'ggvis' IV



Kukuyeva Consulting

Geographical Plots via 'ggvis' V



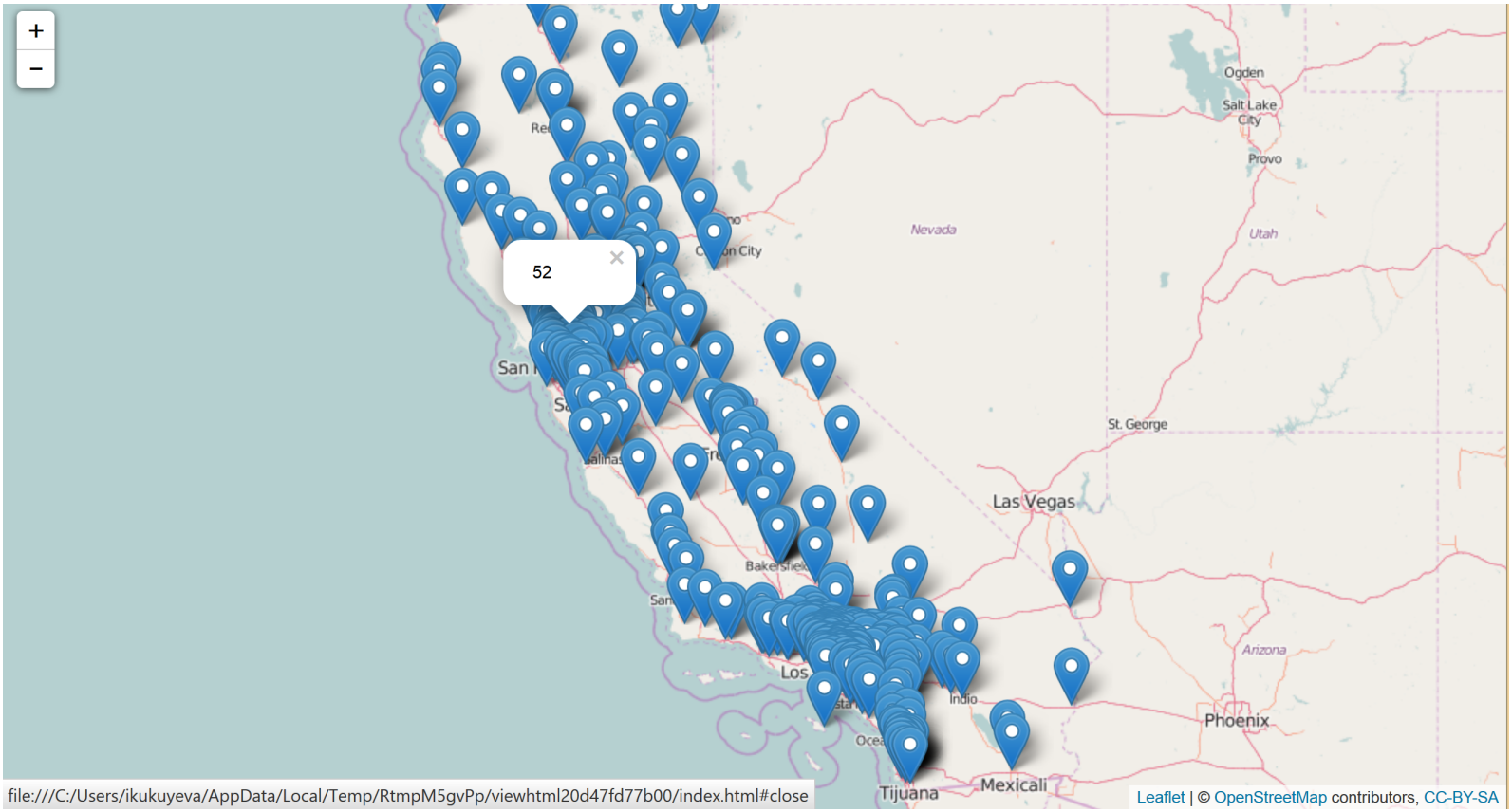
Kukuyeva Consulting

Geographical Plots via 'leaflet' I

Another way to make interactive images is via 'leaflet':

```
1 library(leaflet)
2 leaflet(data = df_hosp) %>%
3   addTiles() %>%
4   addMarkers(
5     ~Longitude,
6     ~Latitude,
7     popup = ~as.character(Total.for.Hospital)
8     )
```

Geographical Plots via 'leaflet' II



Exercise III

In the healthcare data set, what's one way to showcase higher volume procedures in the plot?

Part IV

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
- ...

Kukuyeva Consulting

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)
- ...

Kukuyeva Consulting

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.

Kukuyeva Consulting

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?

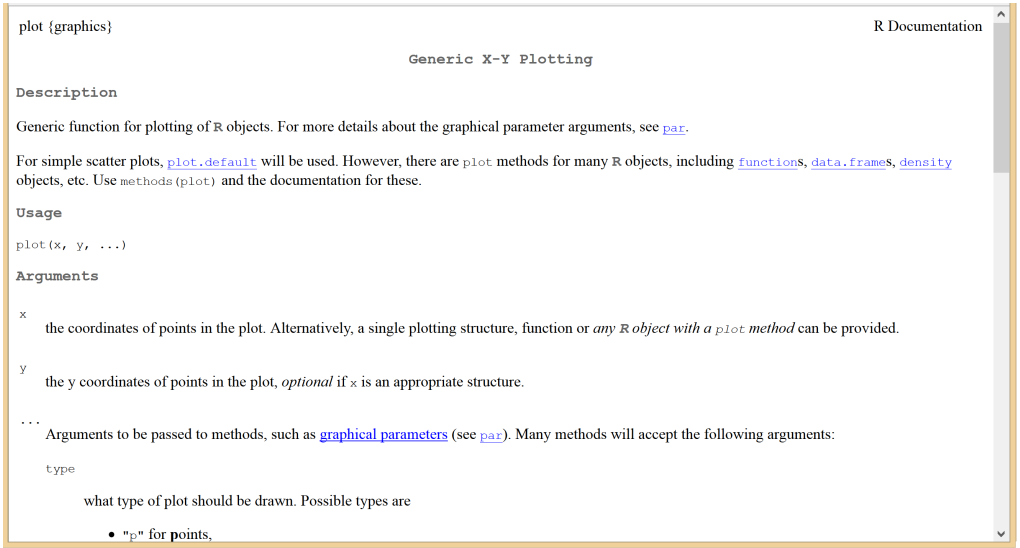
Part 6: Connect with Other Data Scientists

- 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.

```
1  ?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 functions, data.frames, 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.

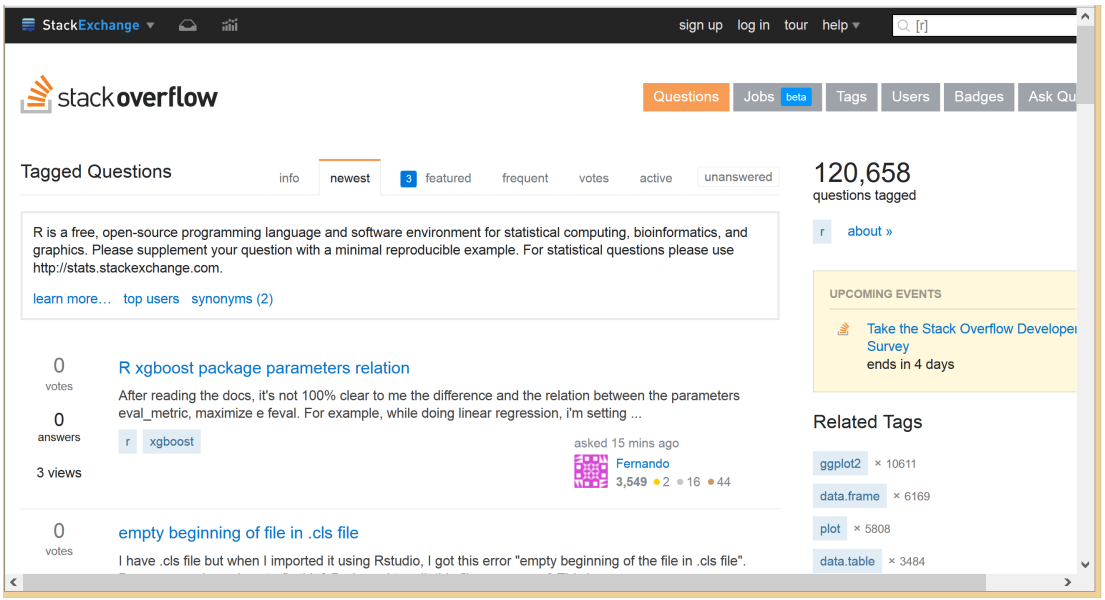
...
Arguments to be passed to methods, such as graphical parameters (see par). Many methods will accept the following arguments:

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`



Kukuyeva Consulting

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

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

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

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

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

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

Kukuyeva Consulting

Online Resources for R II

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/*

Kukuyeva Consulting

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/

Kukuyeva Consulting

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?