Interactive Data Visualization in R

Lingge Li

3/20/2016

Packages

ggplot

- general data visualization package

ggmap

- built on top of ggplot for maps

shiny

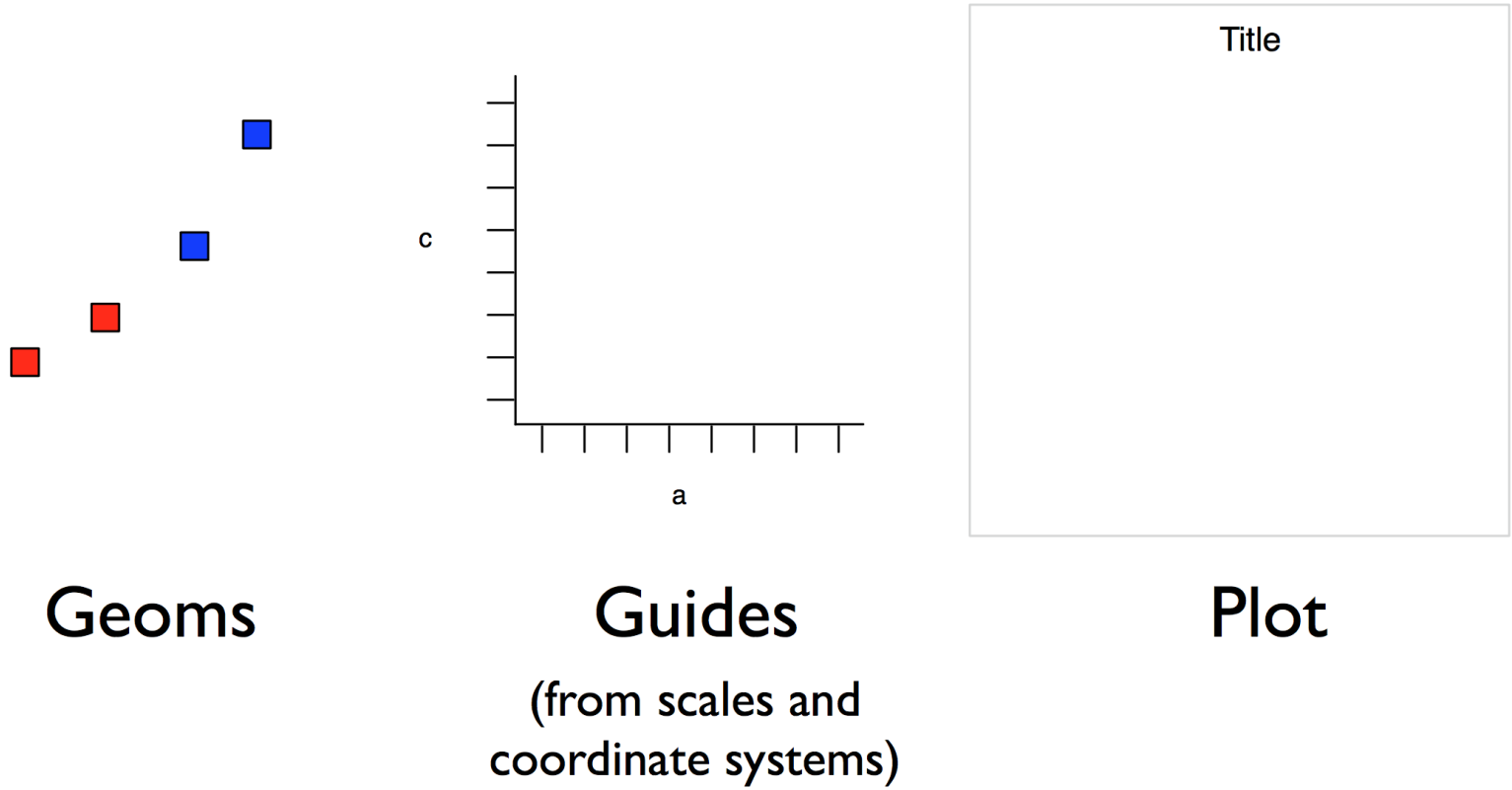
- R based web framework

Motivating example

`http://www.calands.org/map`

Layered grammar of graphics

- ggplot2 follows a specific grammar of graphics



Example taken from Hadley Wickham's book

`http://vita.had.co.nz/papers/layered-grammar.pdf`

How to make a plot

- Geometric objects (geom)
- Aesthetic mapping (aes)
- Statistical transformation (stat)
- Scales and coordinate system

Geoms

- Wide range of geometric objects from points to complex shapes
- geom_point, geom_line, geom_polygon...
- Multiple geometric objects on the same plot with +

Aesthetics

- Coordinate positions (always needed)
- Colour, fill, shape, size. . .

Data + mapping

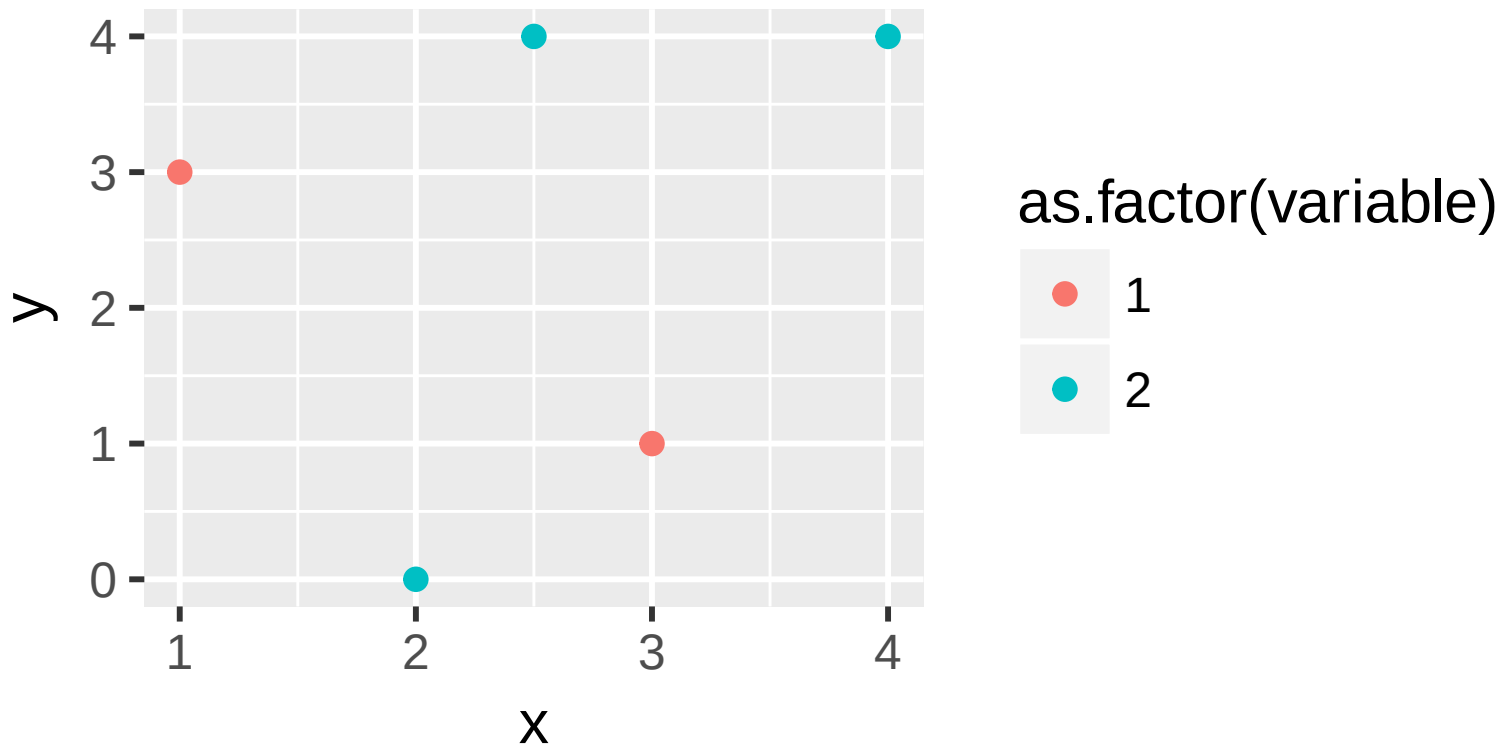
- aes() maps a dataframe to geom
- Each geom can have its own mapping

```
geom_point(data, aes(x, y))
```

Points

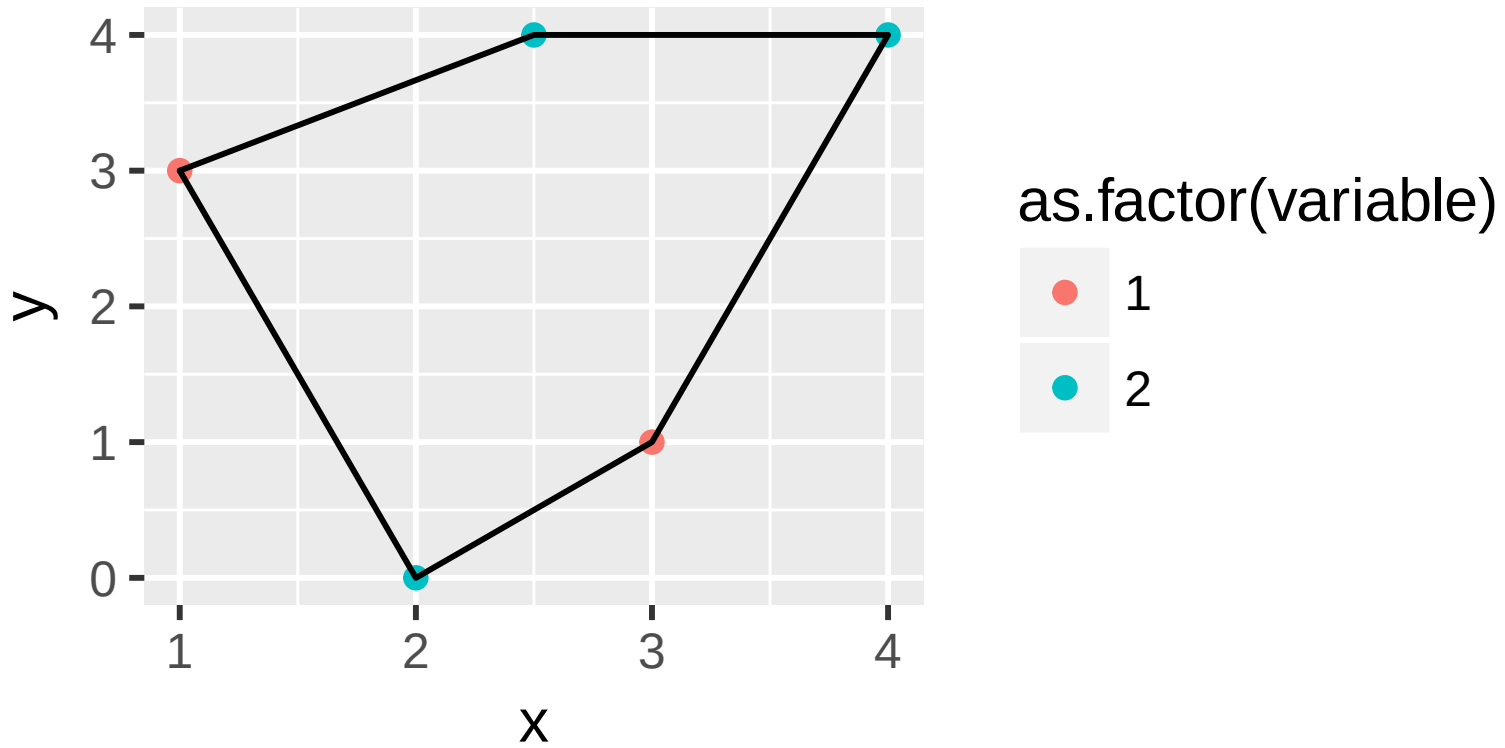
```
## Warning: package 'ggplot2' was built under R version 3.2
```

```
ggplot(data=example) +
  geom_point(aes(x=x, y=y, colour=as.factor(variable)))
```



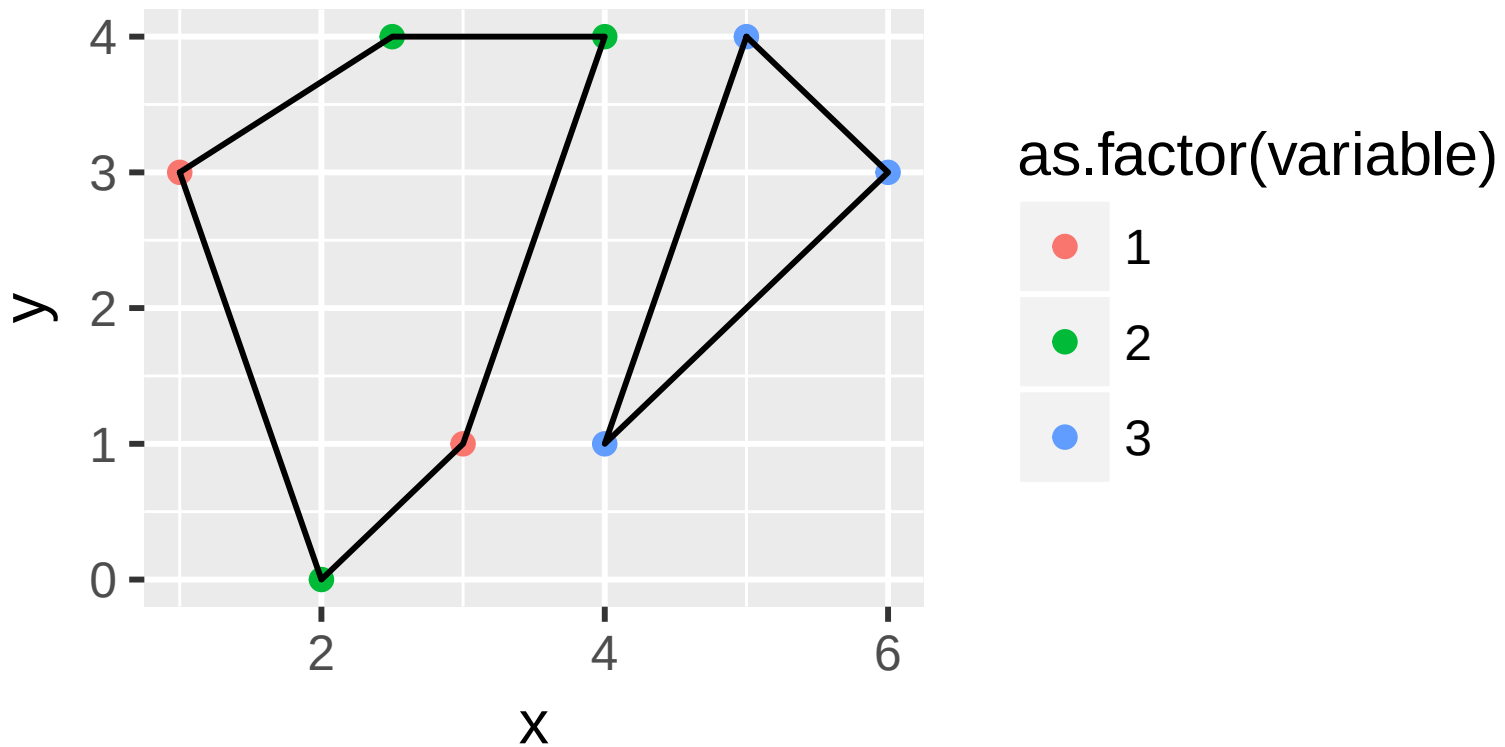
Shape

```
ggplot(data=example) +
  geom_point(aes(x=x, y=y, colour=as.factor(variable))) +
  geom_polygon(aes(x=x, y=y), colour='black', fill=NA)
```



Polygons

```
example$group <- 1
triangle$group <- 2
both <- rbind(example, triangle)
ggplot(data=both) +
  geom_point(aes(x=x, y=y, colour=as.factor(variable))) +
  geom_polygon(aes(x=x, y=y, group=group), colour='black',
```



States dataframe

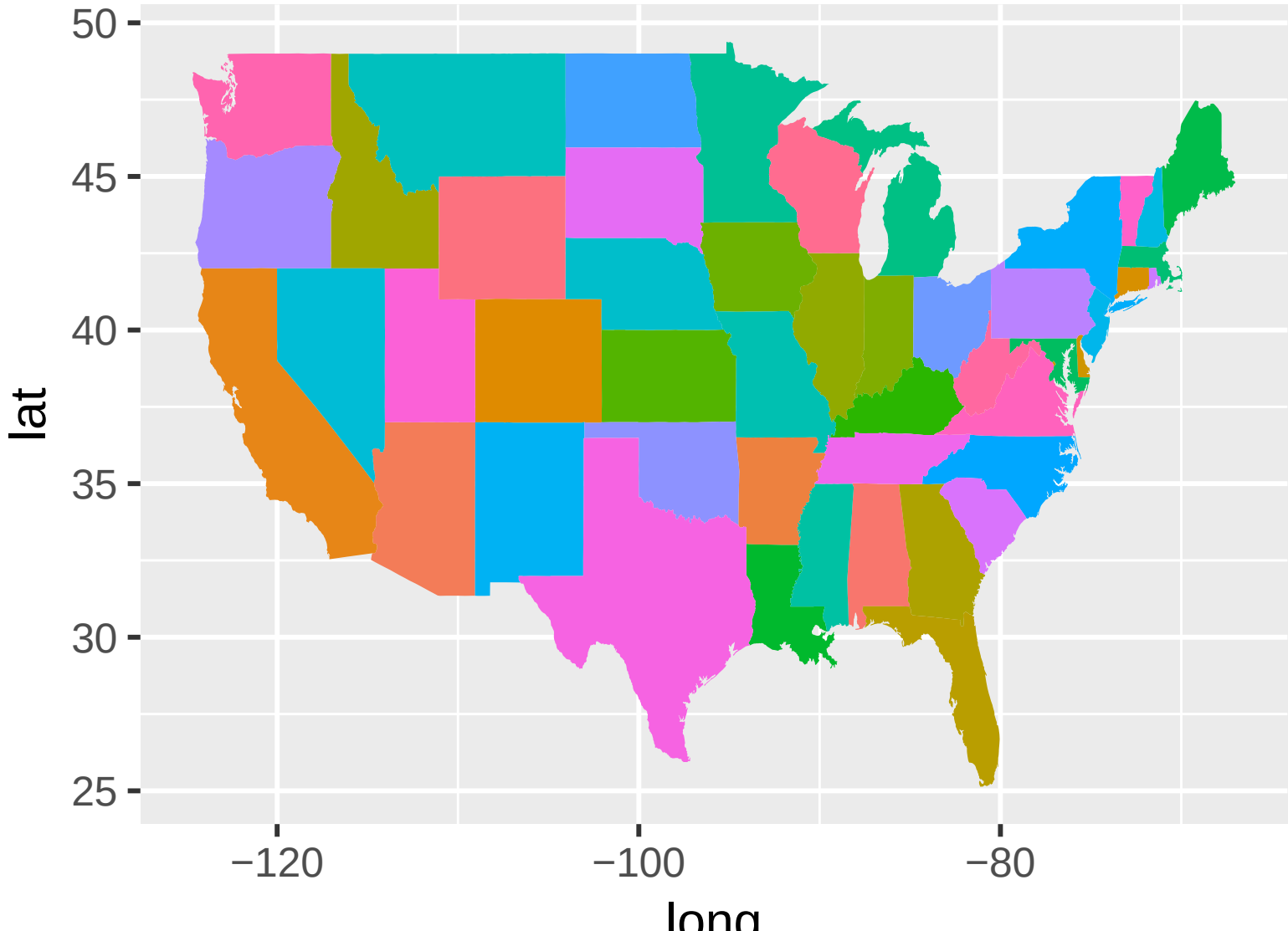
```
library(ggmap)
library(maps)

states <- map_data('state')
head(states)
```

```
##         long      lat group order  region subregion
## 1 -87.46201 30.38968     1     1 alabama      <NA>
## 2 -87.48493 30.37249     1     2 alabama      <NA>
## 3 -87.52503 30.37249     1     3 alabama      <NA>
## 4 -87.53076 30.33239     1     4 alabama      <NA>
## 5 -87.57087 30.32665     1     5 alabama      <NA>
## 6 -87.58806 30.32665     1     6 alabama      <NA>
```

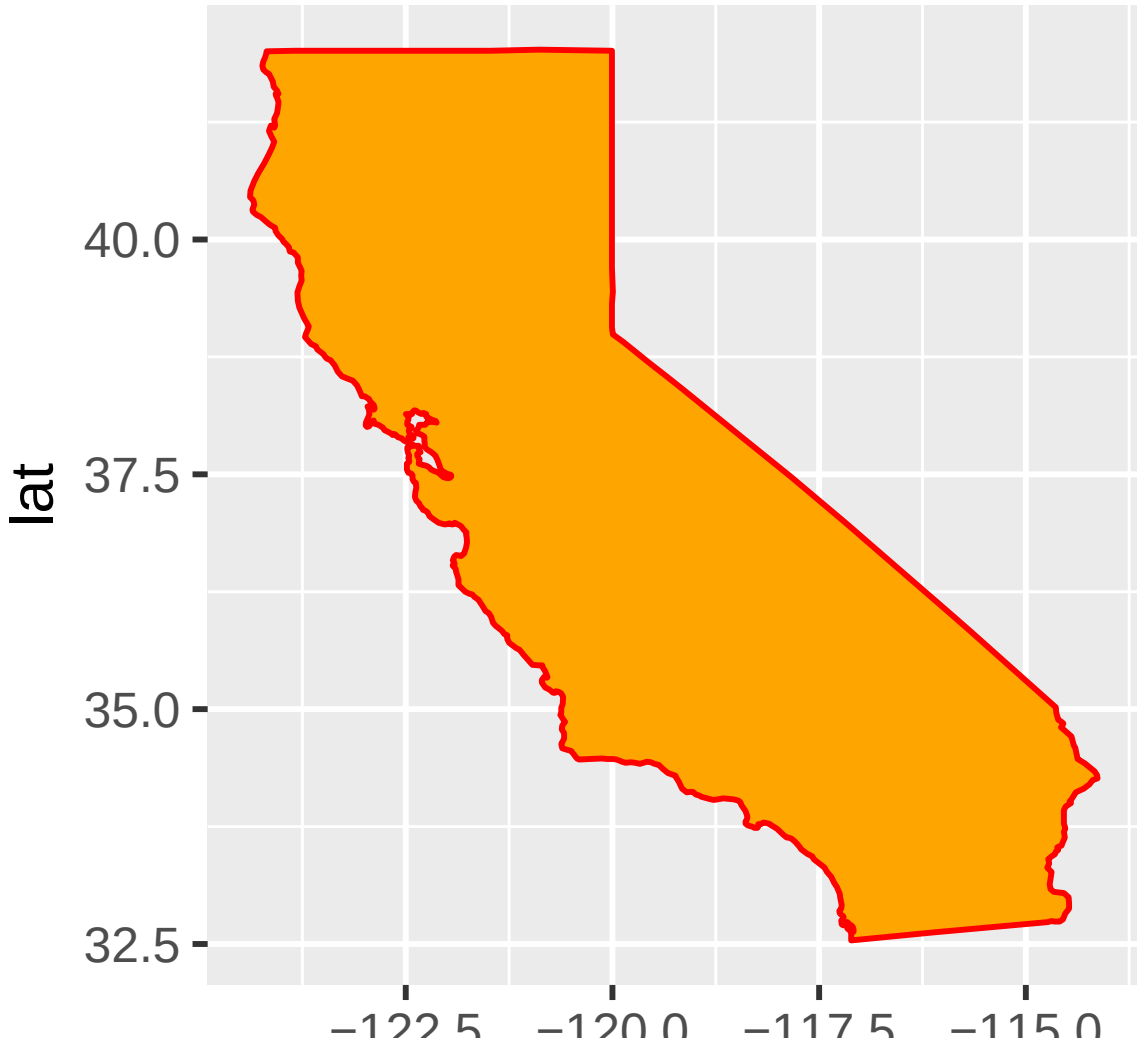
USA

```
ggplot(data=states) +
  geom_polygon(aes(x=long, y=lat, group=group, fill=region)
```



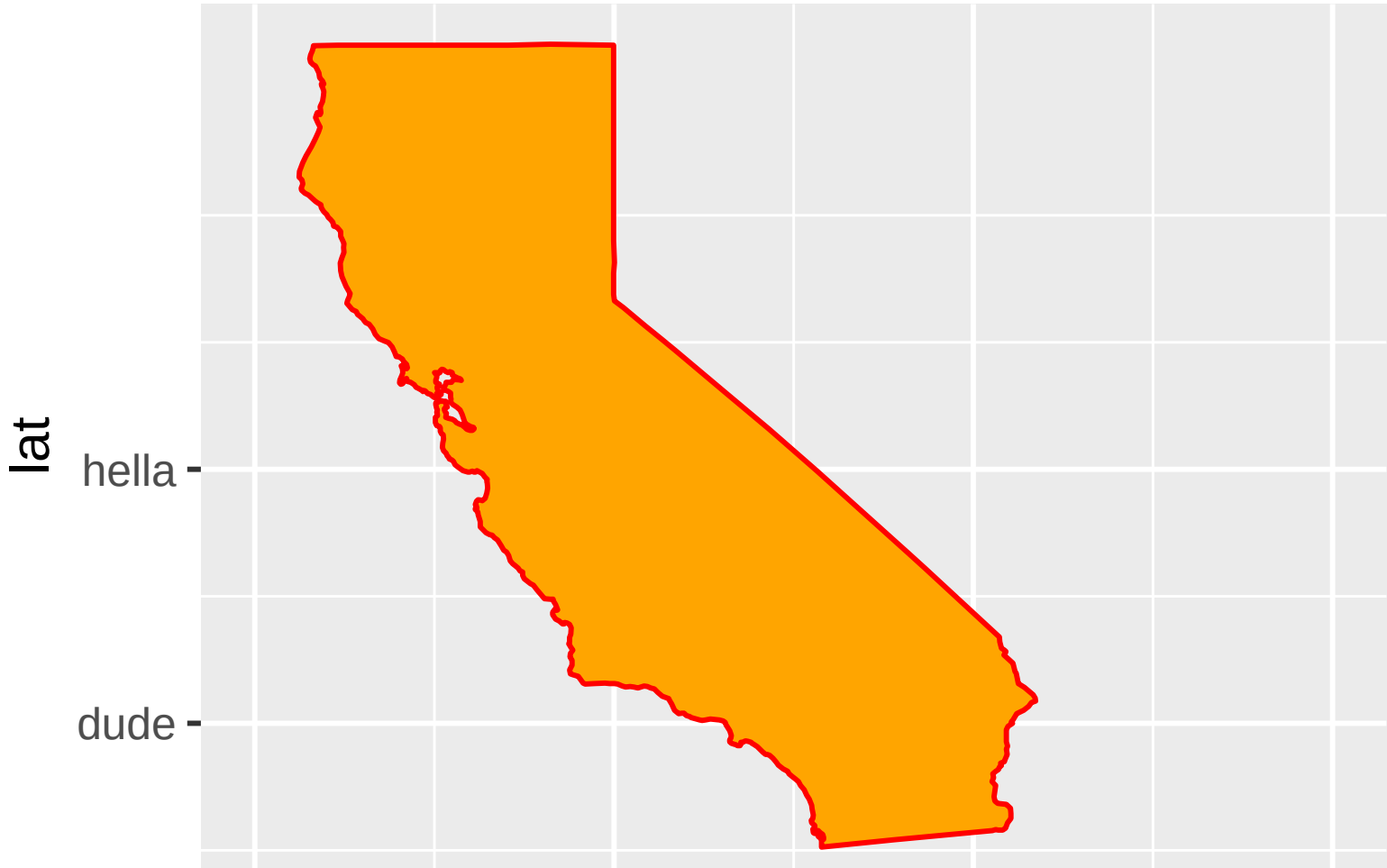
California

```
california <- states[states$region == 'california', ]
ggplot(data=california) +
  geom_polygon(aes(x=long, y=lat), fill='orange', color='re
```



Custom scale

```
ggplot(data=california) +
  geom_polygon(aes(x=long, y=lat), fill='orange', color='re
  scale_x_continuous(limits=c(-125, -110)) +
  scale_y_continuous(breaks=c(34, 37), labels=c('dude', 'he
```



Details

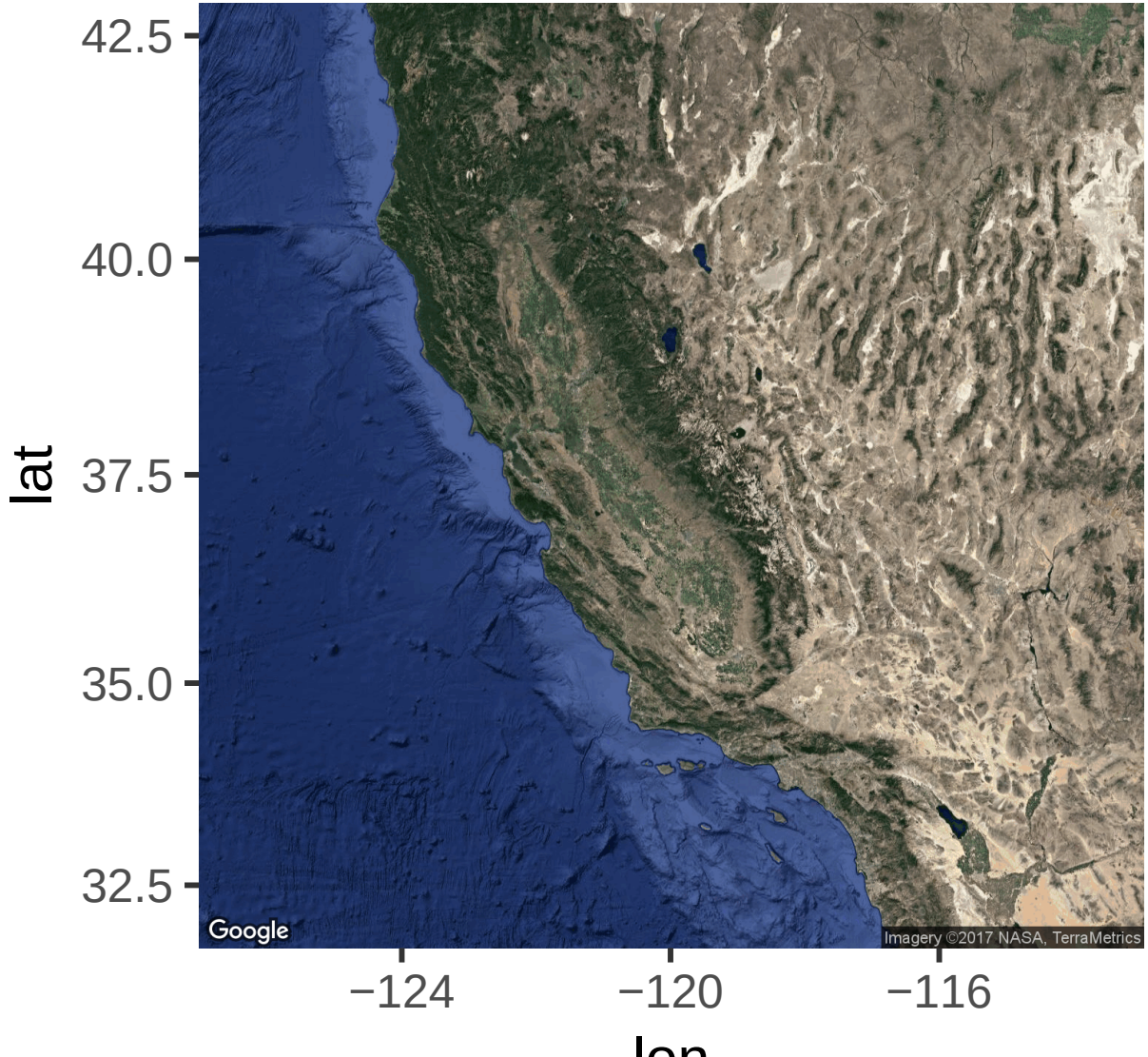
- Theme
- `http://docs.ggplot2.org/current/theme.html`

ggmap

https://github.com/dkahle/ggmap

get_map

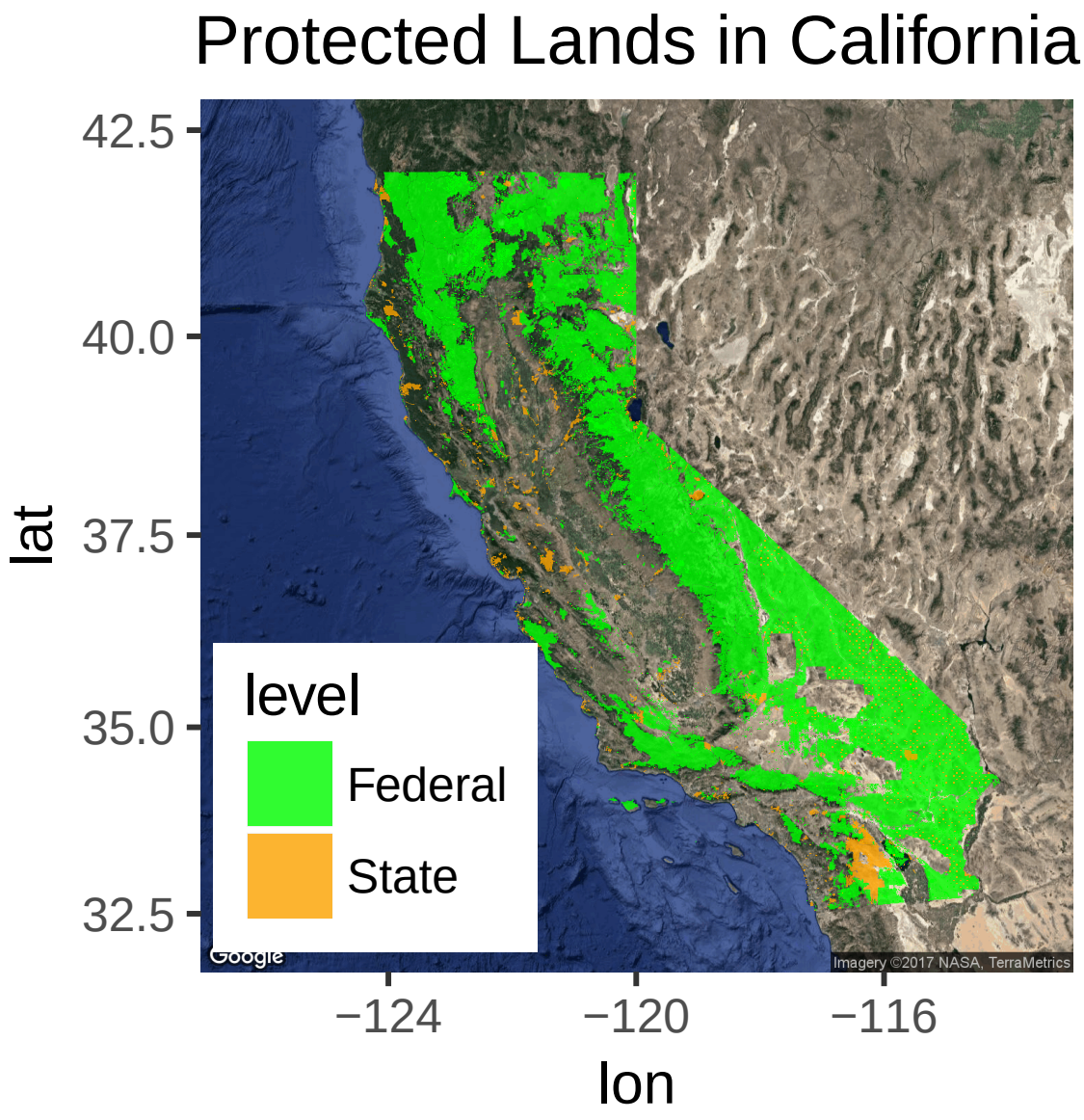
```
cali <- get_googlemap(center=c(lon=-120, lat=37.5), maptype
ggmap(cali)
```



Shapefile

```
setwd('/Users/linggeli/Downloads/CAPD')
CAPD <- readOGR('.', 'CPAD_2016b1_SuperUnits')
CAPD <- spTransform(CAPD, CRS("+proj=longlat +ellps=WGS84 +
proj4string(CAPD) <- CRS("+proj=longlat +ellps=WGS84 +towgs
state <- CAPD[CAPD$MNG_AG_LEV == 'State', ]
state <- fortify(state)
state.low <- unique(data.frame(long=round(state$long, 2), l
```

Protected areas



What is Shiny

- A framework for building web applications
- Best for interactive data visualization
- Apps for exploratory analysis

`http://shiny.datascience.uci.edu/UCIDataScienceInitiative/ClimateActionShiny/`
`http://shiny.datascience.uci.edu/uciMetropolitanFutures/employment_centers/`

Server and UI

- ui.R has everything you see

input widgets, plots, tables...

- server.R does the work

```
shinyServer(function(input, output) {

})

shinyUI(fluidPage(

))
```

Input

- Input handled by specific widgets
- Each input has an id
- Access input value with input$id

`http://shiny.rstudio.com/gallery/widget-gallery.html`

Output

- Output rendered in server with output$id
- Then displayed in ui

```
library(shiny)
shinyServer(function(input, output) {
  output$histogram <- renderPlot({
    hist(faithful$eruptions)
  })
})

shinyUI(fluidPage(
  plotOutput(outputId='histogram')
))
```

Several types of output

- plotOutput (imageOutput)
- tableOutput (dataTableOutput)
- textOutput (html)
- verbatimTextOutput (console)
- htmlOutput (uiOutput)

Reactive environment

- Triggered when input changes
- Output changes accordingly
- Rendering functions reactive

Example

```
library(shiny)
library(ggplot2)
library(ggmap)

shinyServer(function(input, output) {

  load('/Users/linggeli/Downloads/CAPD/parks.Rda')

  output$distPlot <- renderPlot({
    cali <- get_googlemap(center=c(lon=-120, lat=37.5), map
    ggmap(cali) +
      geom_polygon(data=parks, aes(x=long, y=lat, group=gro
  })

})
```

```
shinyUI(fluidPage(
  plotOutput('distPlot', height='640px')
```

Slide to zoom

```
shinyUI(fluidPage(
  sliderInput('res', label='Zoom', min=6, max=9, value=6),
  plotOutput('distPlot', height='640px')
))
```

```
output$distPlot <- renderPlot({
    cali <- get_googlemap(center=c(lon=-120, lat=37.5), map
    ggmap(cali) +
      geom_polygon(data=parks, aes(x=long, y=lat, group=gro
  })
```

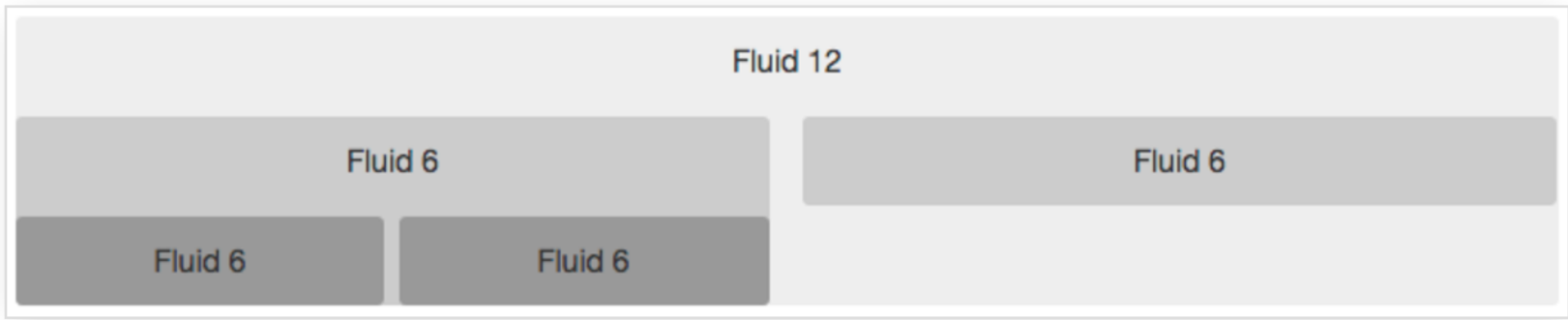
Double click to move

```
location <- reactiveValues(x=-120, y=37.5)

  observeEvent(input$plot_dblclick, {
    current <- input$plot_dblclick
    location$x <- current$x
    location$y <- current$y
  })
```

Layout

- Fluid grid layout (similar to bootstrap)
- 12 columns every row
- Tabset



Other packages

- Widgets for Javascript data visualization

`http://www.htmlwidgets.org/`

Resources

- Documentation

`http://docs.ggplot2.org/current/`

- Gallery with source code

`http://shiny.rstudio.com/gallery/`

- Cheatsheets

`https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf`
`http://shiny.rstudio.com/images/shiny-cheatsheet.pdf`