What is Shiny ?

- R framework for interactive webapps.
- Only R, No CSS / Javascript / HTML
- Examples
  - Shiny Kmeans
  - Simple Box plot
  - Interactive Map display

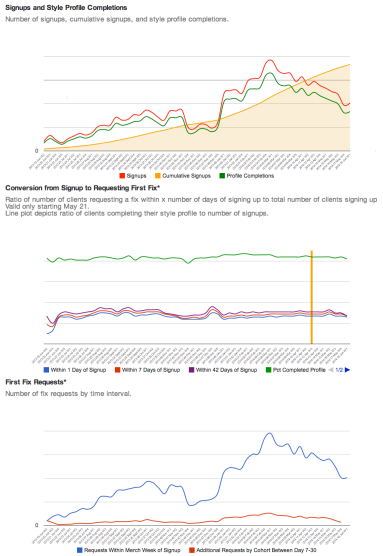
Shiny @ Stitch Fix

- Stitch Fix is an algorithm approach to e-commerce
- Shiny Uses: A/B test Reporting, Monitoring KPMs, External Dashboards, Data Analysis Presentations
- Why Shiny ?
  - Shiny makes Data Scientist autonomous.
  - R
  - Easy to get started.

What is ETD ?

Design Pattern Inspired by ETL

ETD = Extract Transform ~~Load~~ Display



shinyServer(function(input, output) {

E

```
my_fp_mgmt <- file.path(SHINY_HOME, 'apps',
                                    'mgmt',
                                    'mgmt.csv')
rs <- dbSendQuery(con, sql_mgmt)
```

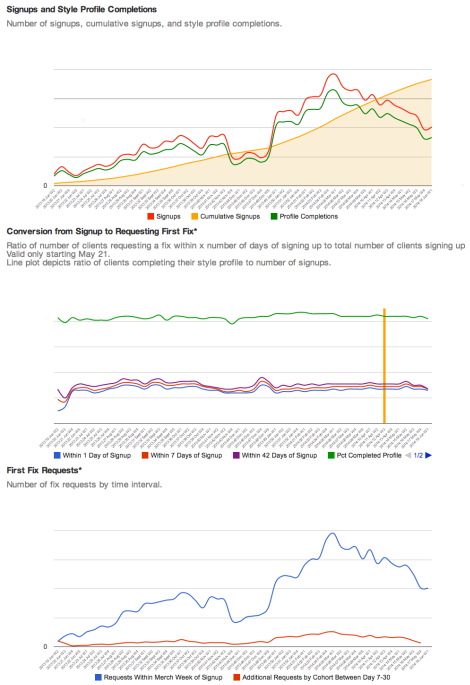
T

```
rawdat$signups_cum <- cumsum(rawdat$signups)
```

D

```
gvComboChart(rawdat,
    ……
    )
```

Shiny (Black Box) + Data Scientists



shinyServer(function(input, output) {

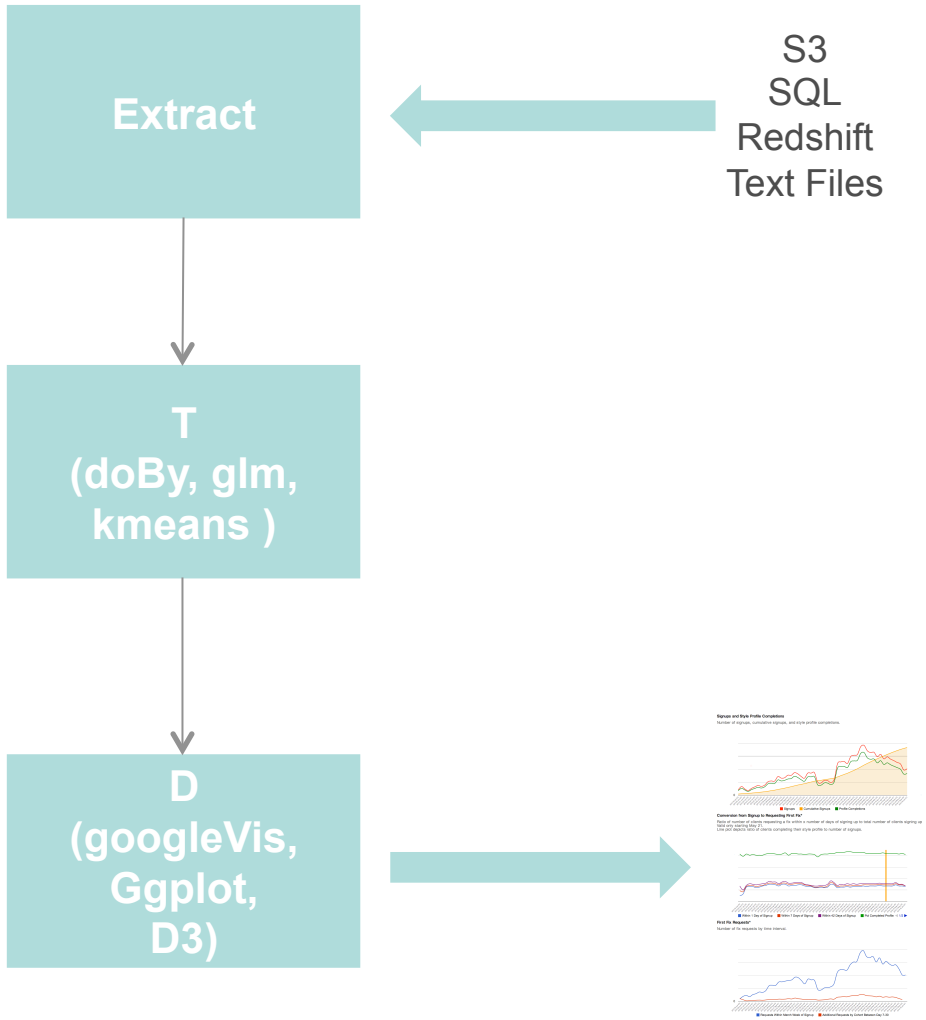
```
my_fp_mgmt <- file.path(SHINY_HOME, 'apps', 'mgmt', 'mgmt.csv')
my_fp_style <- file.path(SHINY_HOME, 'apps', 'mgmt', 'style.csv')
my_fp_time <- file.path(SHINY_HOME, 'apps', 'mgmt', 'last_run.csv')

mgmtData <- reactive({

if(input$refresh == 0){
if((file.exists(my_fp_mgmt) & file.exists(my_fp_time))){
mytime <- read.csv(my_fp_time)
# If it has been more than one day since it's been run, then run the query
again
if(as.numeric(as.Date(format(Sys.time(), tz="America/
Los_Angeles",usetz=TRUE)) - as.Date(mytime$x)) >= 0){ #as.Date("2013-09-05"))
#input$refresh
mgmt.data <- read.csv(my_fp_mgmt, header=TRUE)
return(mgmt.data[,-1])
} else {
# Otherwise read the existing csv and return the data
mgmt.data <- read.csv(my_fp_mgmt, header=TRUE)
return(mgmt.data[,-1])
}
} else {
input$refresh
}
}

isolate({
rs <- dbSendQuery(con, sql_mgmt)

rawdat <- fetch(rs,n=-1)
rawdat$signups_cum <- cumsum(rawdat$signups)
rawdat <- tail(rawdat, 52)

write.csv(format(Sys.time(), tz="America/Los_Angeles",usetz=TRUE),
my_fp_time)
write.csv(rawdat, my_fp_mgmt)
```

ETD to the rescue

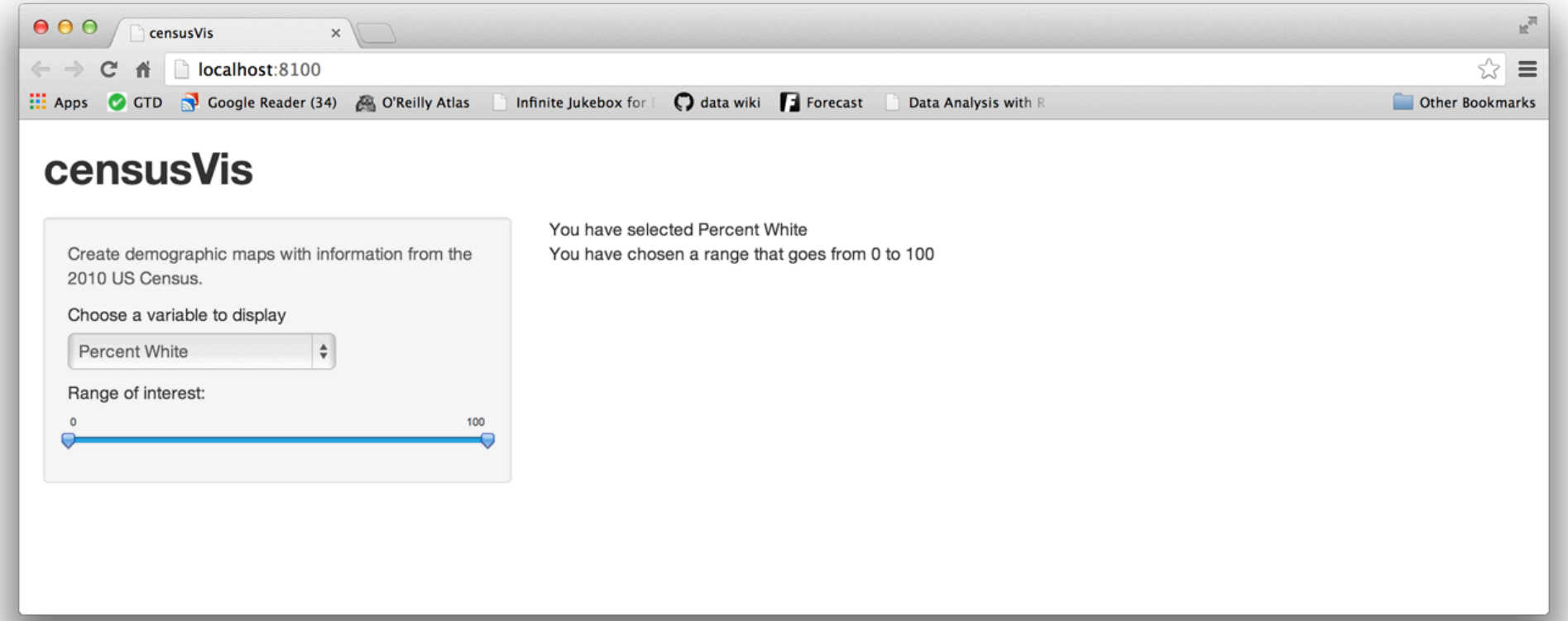
```
shinyServer(function(input, output) {

E
  my_fp_mgmt <- file.path(SHINY_HOME, 'apps', 'mgmt', 'mgmt.csv')
  my_fp_style <- file.path(SHINY_HOME, 'apps', 'mgmt', 'style.csv')
  my_fp_time <- file.path(SHINY_HOME, 'apps', 'mgmt', 'last_run.csv')

  mgmtData <- reactive({

  if(input$refresh == 0){
  if((file.exists(my_fp_mgmt) & file.exists(my_fp_time))){
  mytime <- read.csv(my_fp_time)
  # If it has been more than one day since it's been run, then run the query again
  if(as.numeric(as.Date(format(Sys.time(), tz="America/Los_Angeles",usetz=TRUE))
  - as.Date(mytime$x)) >= 0){ #as.Date("2013-09-05"))
  #input$refresh
T
  mgmt.data <- read.csv(my_fp_mgmt, header=TRUE)
  return(mgmt.data[,-1])
  } else {
  # Otherwise read the existing csv and return the data
  mgmt.data <- read.csv(my_fp_mgmt, header=TRUE)
  return(mgmt.data[,-1])
  }
  } else {
  input$refresh
  }
  }

D
  isolate({
  rs <- dbSendQuery(con, sql_mgmt)

  rawdat <- fetch(rs,n=-1)
  rawdat$signups_cum <- cumsum(rawdat$signups)
  rawdat <- tail(rawdat, 52)

  write.csv(format(Sys.time(), tz="America/Los_Angeles",usetz=TRUE), my_fp_time)
  write.csv(rawdat, my_fp_mgmt)

}
```

```
Graph1 = {
   'extract'=foo,
   'transform'=function(dataframe) {
// do transformation
    },
   'display'=function(dataframe) {
// do display
    },
Graph2 = { …. }
Graph3 = { …. }

shinyServer(function(input, output) {
   renderWidget(Graph1, input);
   renderWidget(Graph2, input);
   renderWidget(Graph3, input);
})
```

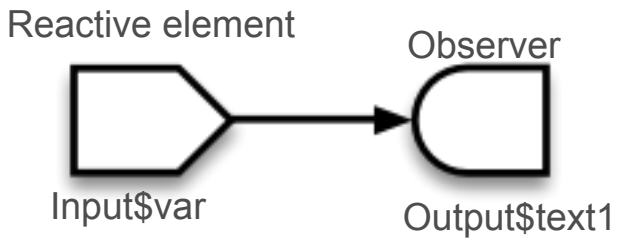
Eliminate repetition



Simple Shiny webapp



Shiny Internals : Shiny reactivity in a nutshell



Reactive Context

```
# server.R

shinyServer(
  function(input, output) {

    output$text1 <- renderText({
      paste("You have selected", input$var)
    })

  }
)
```

Key to the ETD design pattern = Separation, separation, separation

Non-reactive

```
Widget = list(
    'extract'= SQL / S3 / Memcache / Redshift
    'transform'=function(dat, input){
     },
    'display'=function(dat, input) {
})

render_widget <- function(widget, input) {
}
```


Library Functions

```
shinyServer(function(input, output)  {
   output$widget1 <- renderGvis({
      render_widget(occupWidget, 1)
   })
}
```

Reactive

Application of ETD design pattern @ Stitch Fix

- Template for modular and readable code.
- Reusable and standardized components (E,T, Ds)
- Time taken to build dashboards reduced drastically.
- Less code = Happy Data Scientists

Concluding

- Future : Could evolve into dashboard building framework on top of Shiny
- Lets all go write more awesome Shiny apps !!
- We are hiring.