# VISUALISING OPERATIONAL INFORMATICS DATA USING R

DI4R 2016 KRAKÓW

LESLIE
MCINTOSH, PHD

@mcintold

CONNIE
ZABAROUSKAYA, MBA

@connie_zest

## FUNDERS

- ICTS for NIH CTSA Grant #UL1 TR000448
- Alvin J. Siteman Cancer Center, Grant # P30CA091842
- Washington University in St. Louis, USA

## GOALS OF THE WORKSHOP

Conceptualizing Visualisations

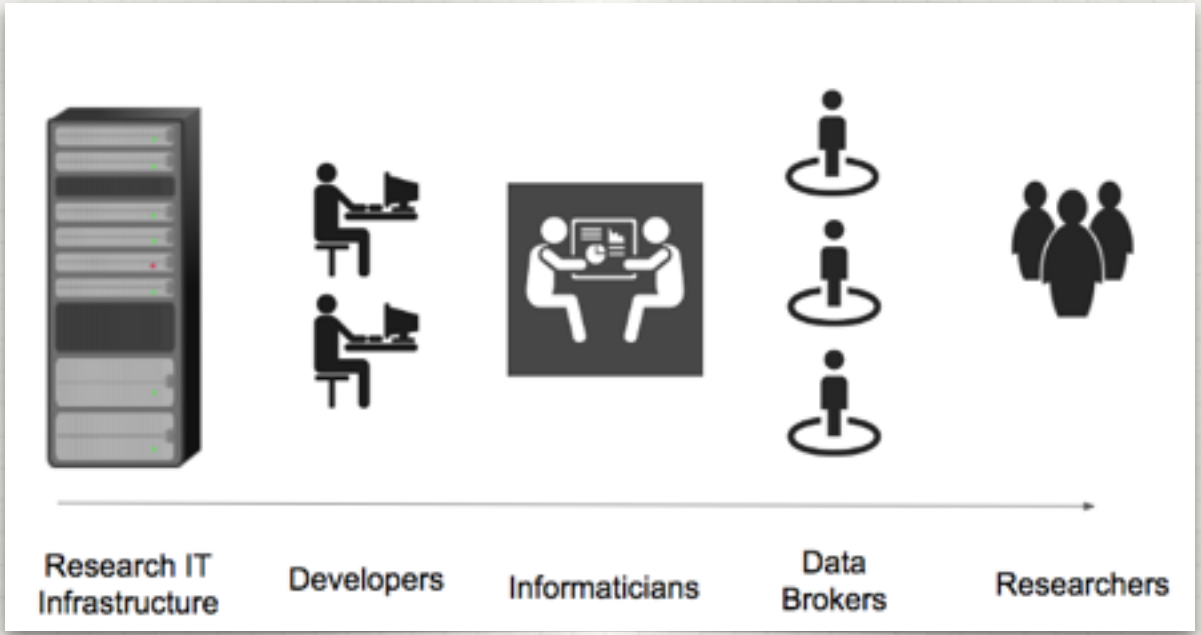
Forming Answerable Question(s)

(Technically) Presenting Data

https://github.com/CBMIWU/DI4R_TutorialVisData

Discuss the main principles of efficient data visualization
Identify and analyze Key Performance Indicators with the goal of providing actionable insights
Explain the meaning and purpose of Grammar of Graphics
Build charts using rCharts package for R
Develop Shiny apps and dashboards, and implement rCharts inside these apps

## CENTER FOR BIOMEDICAL SCIENCES (CBMI)



# PART (I)

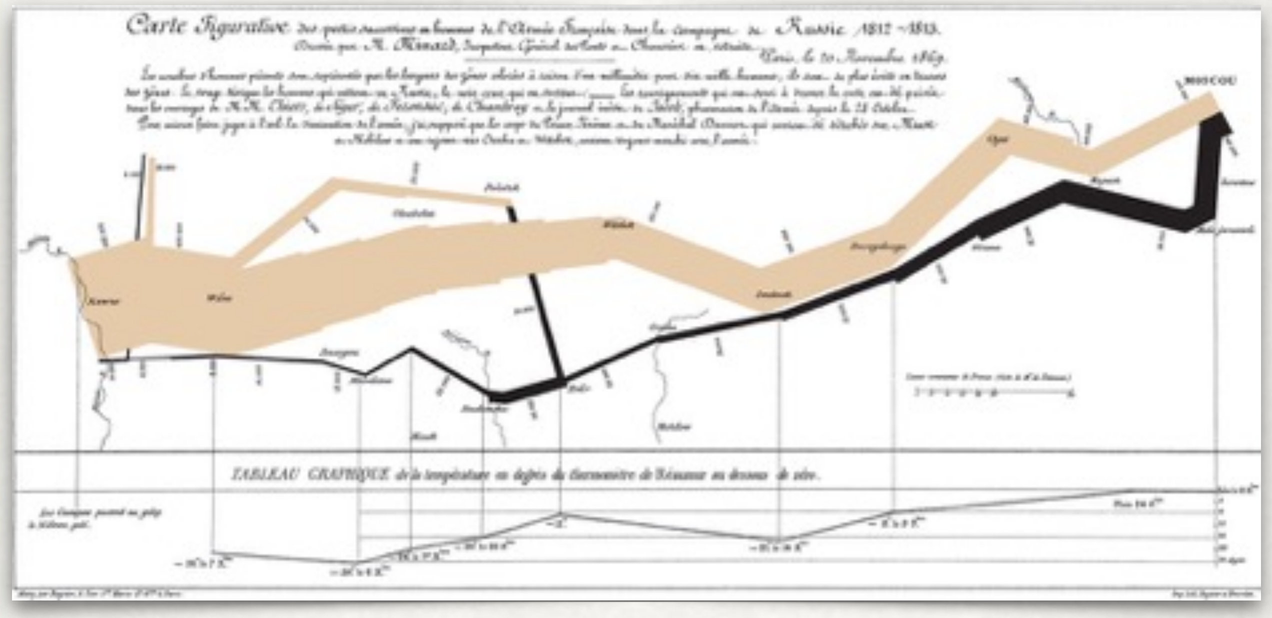
## PRINCIPLES OF EFFICIENT DATA VISUALIZATION

## PRINCIPLE 1

FOR DATA VISUALIZATION
IN GENERAL,
BUILD AROUND A STORY

quote Dona Wong)

## PRINCIPLE 1



by Charles Joseph Minard

Graphs and charts are not meant for decoration, they must deliver a clear message.

One of the first and most important principles is to build your visualization around a story. You have heard it a million times already, and yet, some still create charts because they have to make the report look pretty, or because that's the data they have available. According to Jer Thorpe effective data visualization should be both visually pleasing and educational (check quote).

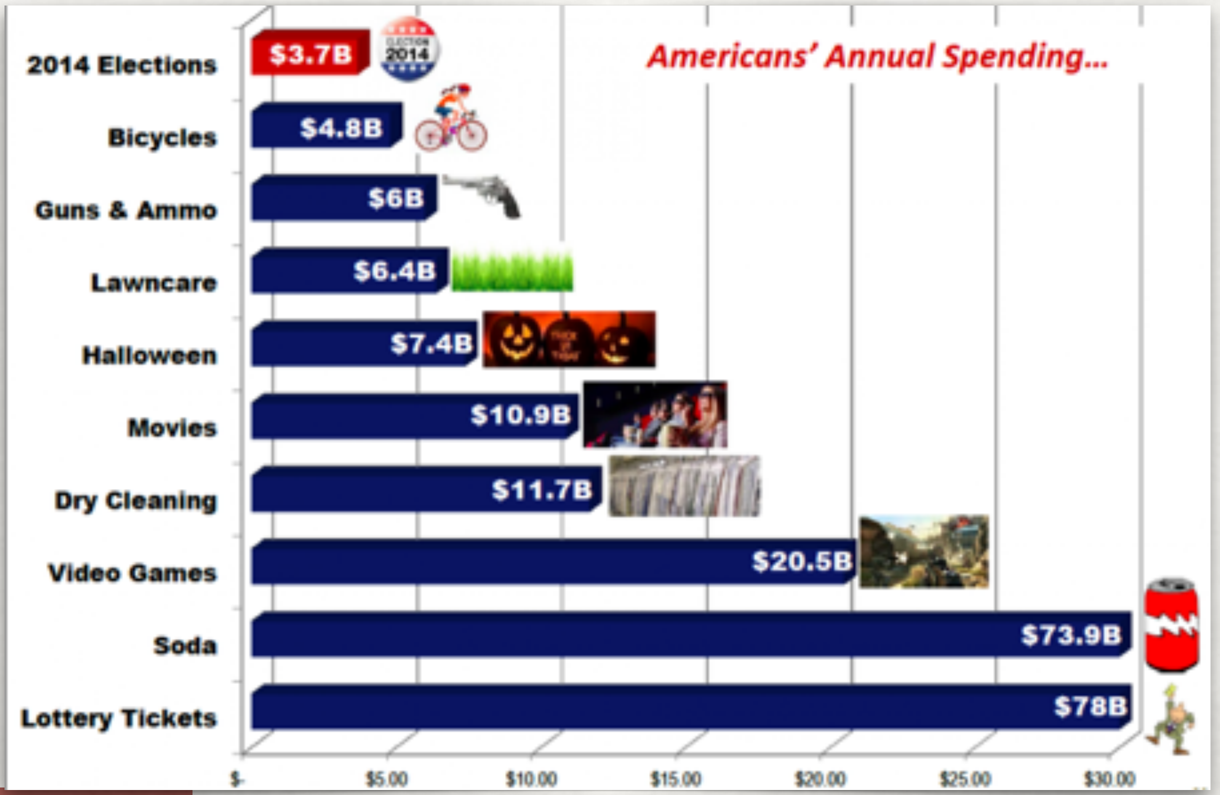
## PRINCIPLE 2

**LIMIT VISUALISATIONS TO A FEW HIGHLIGHTS**

DON'T OVERLOAD WITH COLOR, SHAPES, TEXT

quote Dona Wong)

## PRINCIPLE 2



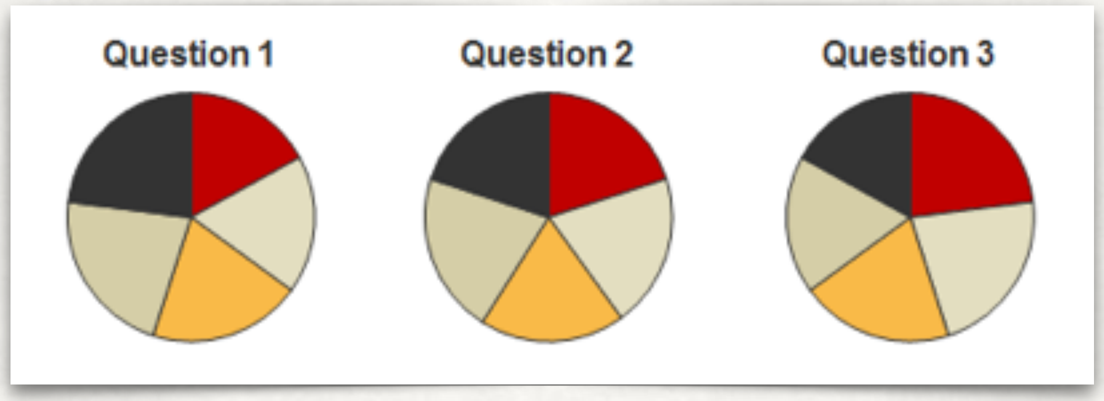
Example of poor chart
Charts that tell a story and deliver a clear message do not normally need any embellishment, such as abundance of colors, shades, even grid lines, or creative fonts. It's best to use as little text as possible, and as little colors as possible, so that they do not distract the audience from the message in the charts.

PRINCIPLE 3

## CHOOSE APPROPRIATE DISPLAY MEDIA

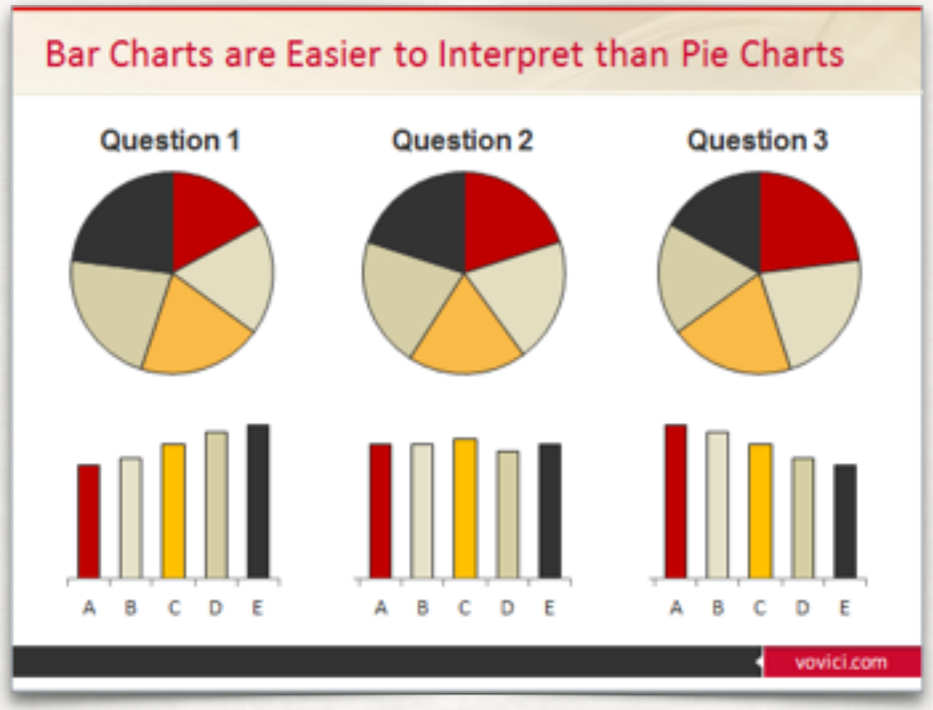
quote Dona Wong)

## WHICH SECTION IN THE PIE CHARTS IS THE LARGEST IN EACH OF THE PIE CHARTS?



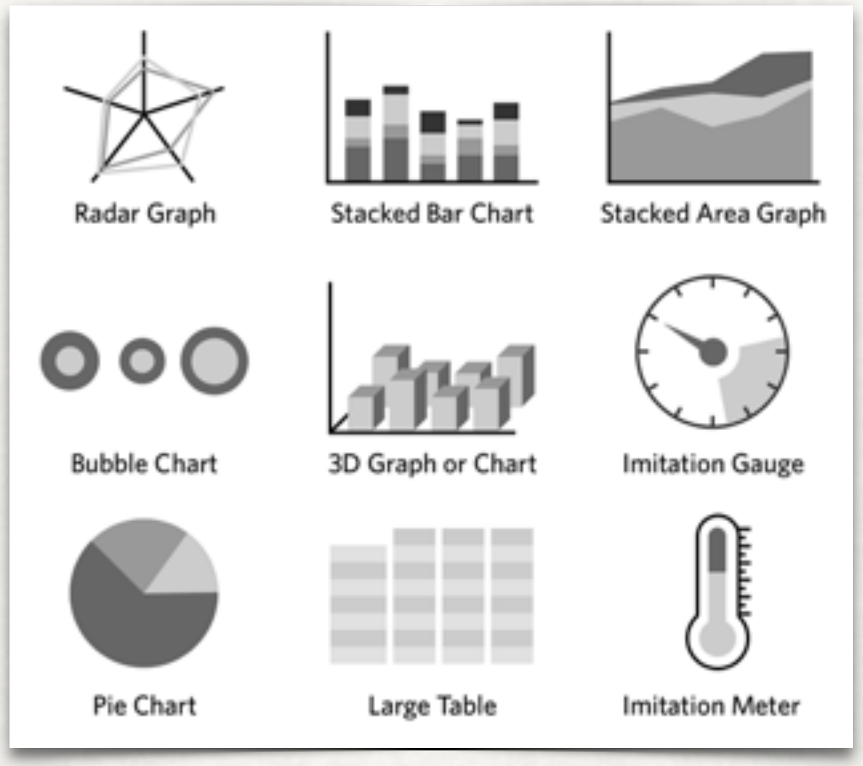
The next principle is choosing the appropriate media representation for the data you have. It has to be optimal to deliver the message in mind, and at the same time be optimal from the point of view of data. It is common for instance to use pie-charts for parts of the whole, however, this type of chart has been known recently to be quite abusive to the audience, because they have to do so much extra work to discern any insight from it. Let's take this example. Can you tell me what section is greater in all these 3 charts? It is pretty difficult because all the sections are pretty close, so let's look at the data in a different format.

## PRINCIPLE 3



Now it's obvious which sections are the greatest. For these situations it is in fact recommended to use horizontal bar charts with ascending or descending order depending on the context.

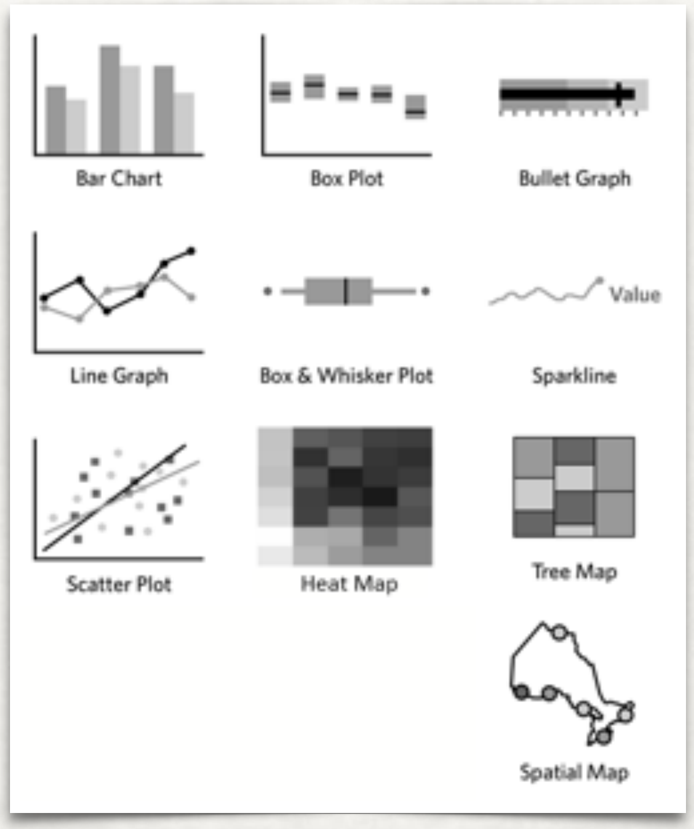
## **UNSUITABLE** CHOICE OF CHARTS



Stephen Few

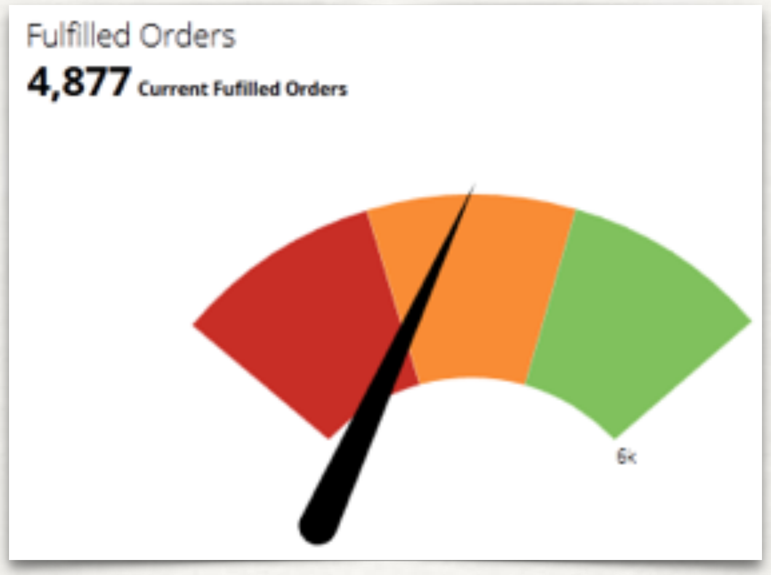
Speaking of charts, there is a guru in the field. Stephen Few founded Perceptual Edge in 2003. With 30 years of experience as an innovator, consultant, and educator in the fields of business intelligence and information design, Stephen is now a leading expert in data visualization for sensemaking and communication. And here is the list of unsuitable charts according to his theory, which is quite easy to agree with. To give you a few examples, as we have seen already. He recommends to substitute pie-charts with vertical or horizontal bar graphs, as well as substitute gauges with bullet graphs, while the latter is in fact his own invention, that's why they are not easy to find in Microsoft Word tools or some other popular visualization software tools. 3D charts are in fact a very bad practice, because they distort the perception of data completely, and you should steer clear of them, unless you design spatial maps or perform statistical analyses with multiple dimensions.

## **SUITABLE** CHOICE OF CHARTS



As you can see clean uncluttered design makes more sense (even in black and white), than some fully colored charts with many unnecessary details.

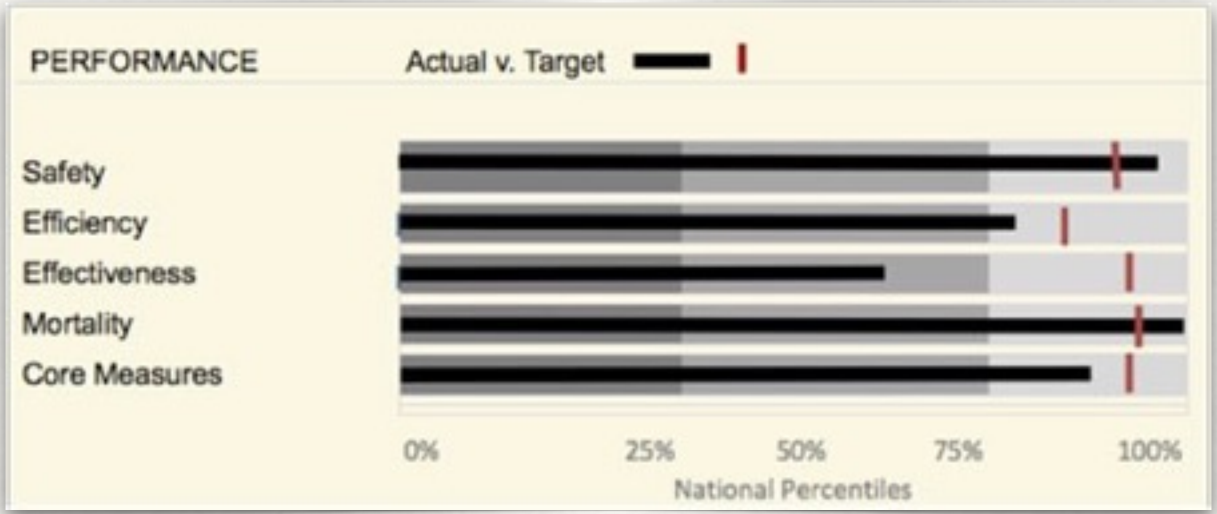
## A NOTE ABOUT GAUGES AND BULLET GRAPHS



Gauge example from Domo http://www.domo.com/roles/operations (no longer live)

Gauges are distracting, hard to interpret, can easily be done wrong and take up too much space.
We wanted to iterate one more time that even though gauges have been extensively used on dashboards ever since the dashboards first appeared, they are a very ineffective and space consuming tool. Especially, so when they are designed improperly. We took this example from the website of Domo, a dashboarding software provider. As you can see it is highly useless, because even the arrow is not placed right and moves on the bottom side and not on the top, and little sense can be made from it, unless you look at the number displayed in the top left corner. Now look at how simple and clear the bullet graphs are and how little space they take up.

## A NOTE ABOUT GAUGES AND BULLET GRAPHS



On the other hand, bullet charts deliver the message of many dimensions of data quicker and take up less space.

## PART (II)

FORMING ANSWERABLE QUESTIONS

## WHY USE KPIS?

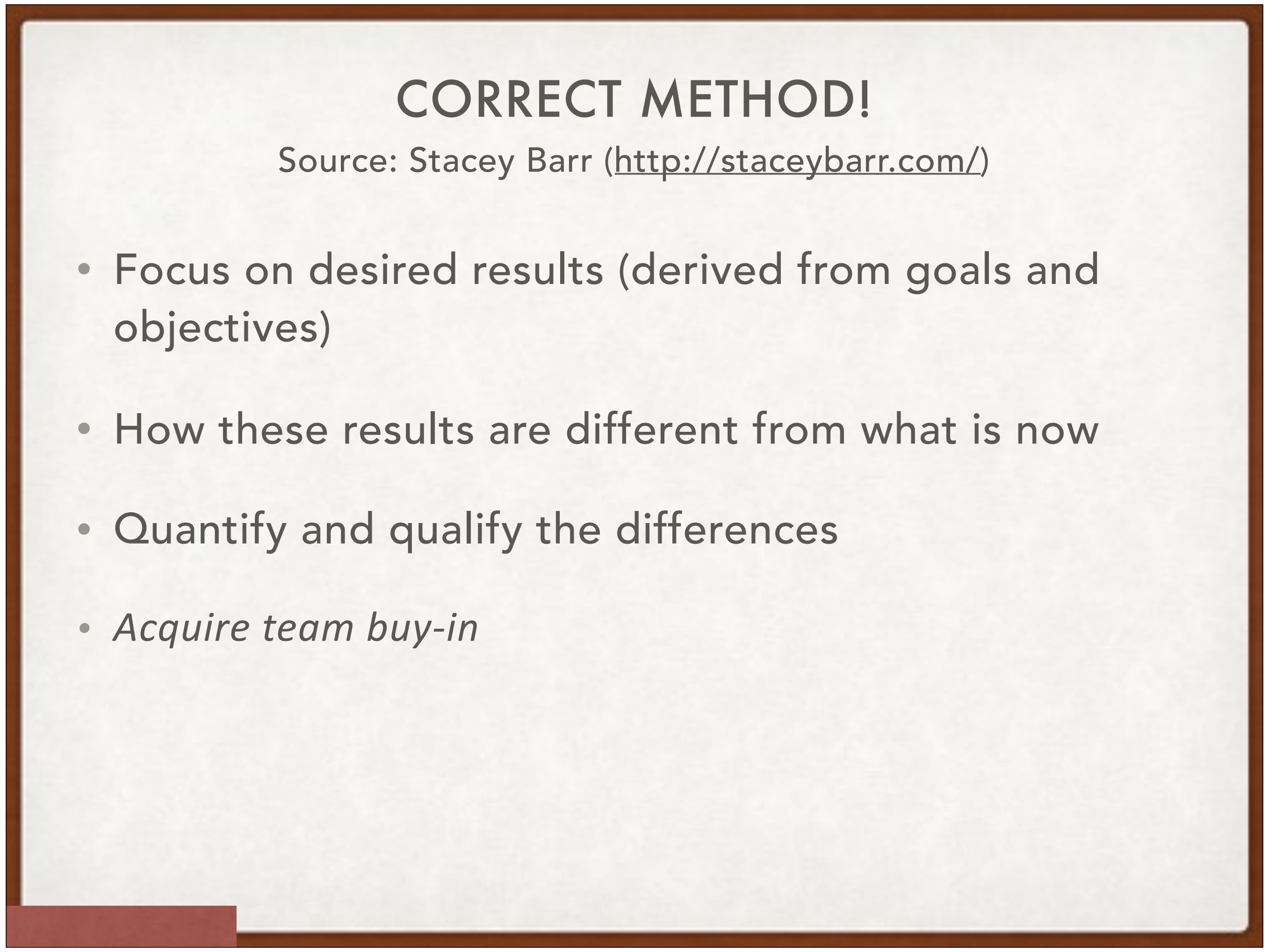
- Key Performance Indicators (KPI)
- Replace long and meaningless reports with actionable metrics
- Improve team performance through transparent insight and clear relatable objectives

Value of Using KPIs?
Resource: David Parmenter, (book title Key Performance Indicators (KPI) : Developing, Implementing, and Using Winning KPIs )
Each measure should have a reason to exist, a linkage to a certain critical success factor, and what gets reported should always get actioned.
Replace long and meaningless reports with actionable metrics
Improve team performance through transparent insight and clear relatable objectives
Quoting: Parmenter, David. Key Performance Indicators (KPI) : Developing, Implementing, and Using Winning KPIs (2). Hoboken, US: Wiley, 2010. ProQuest ebrary. Web. 17 August 2016. Copyright © 2010. Wiley. All rights reserved.

## COMMON METHODS IN KPI DEVELOPMENT

- Brainstorming
- Adoption from other organizations or benchmarking
- Using existing measures
- Measures enforced by stakeholders or executives

Let's now turn our attention to the theory and methods of identifying and analyzing Key Performance Indicators. Here are some common methods of developing KPIs (show the slide). Let's see a show of hands of who uses the brainstorming technique? Now the benchmarking or adoption of other institutions KPIs? What about using the existing measures collected? What about the often reported or stakeholder measures?
According to Stacey Bar, an expert in Performance Measurement, all of these techniques should not be used for developing KPIs.



## CORRECT METHOD!

Source: Stacey Barr (http://staceybarr.com/)

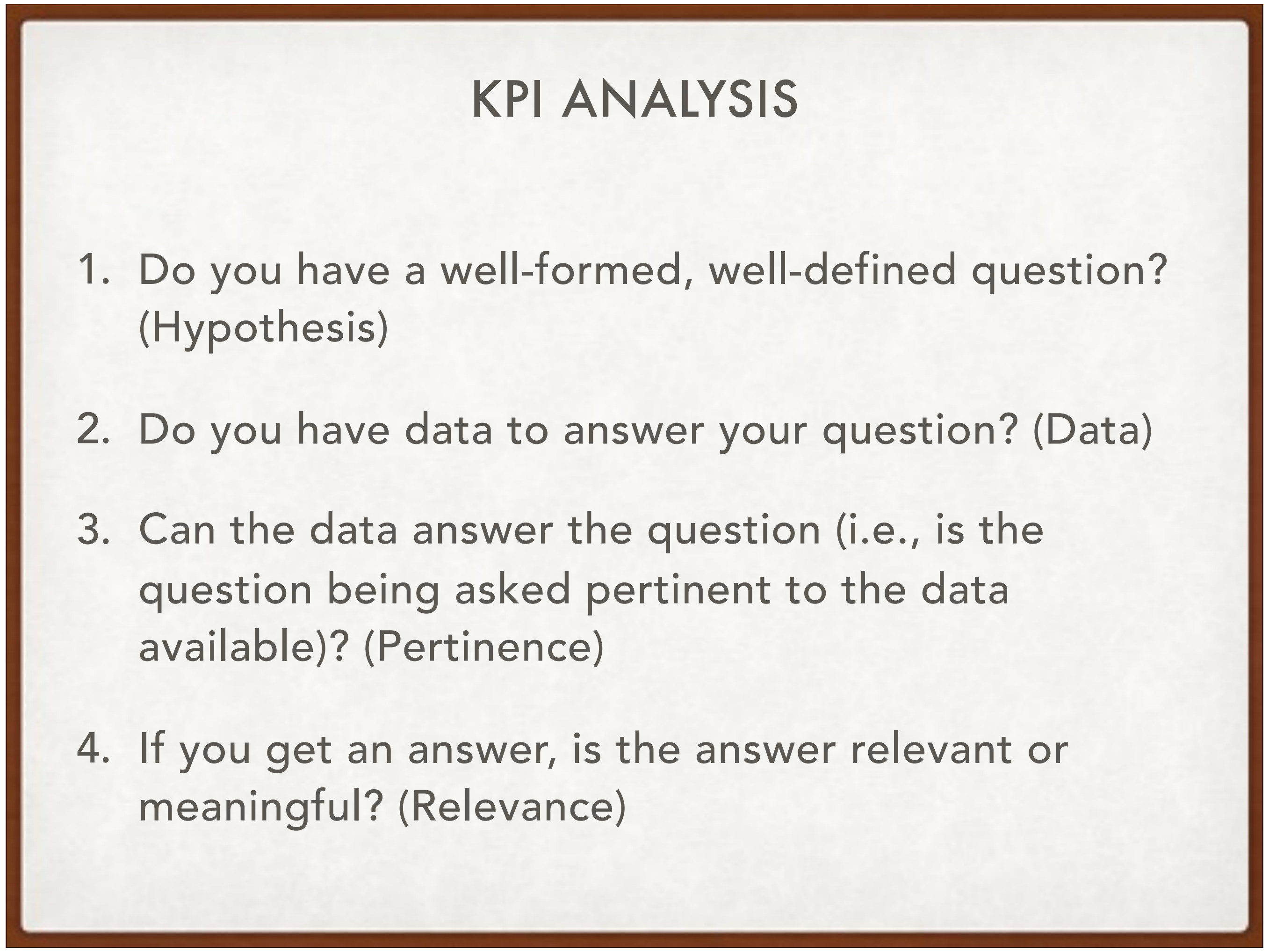
- Focus on desired results (derived from goals and objectives)
- How these results are different from what is now
- Quantify and qualify the differences
- *Acquire team buy-in*

In her theory, the correct way to choose KPIs is an iterative process with all levels of involved stakeholders. You work backwards from the desired results, which you derive from the goals and objectives of the organization. You quantify and qualify the differences between those results and what happens now, and then convert those metrics into charts. The result is a few measures that are clear, actionable and engaging to all who participated in the dialogue of creating the measures. They are actively used in decision making, have owners and measure progress over time to the strategic result. This approach also prevents information overload.

The last point (acquire team buy-in) is from Parmenter, David. Key Performance Indicators (KPI) : Developing, Implementing, and Using Winning KPIs (2). Hoboken, US: Wiley, 2010. ProQuest ebrary. Web. 17 August 2016. Copyright © 2010. Wiley. All rights reserved.

## THE RESULT

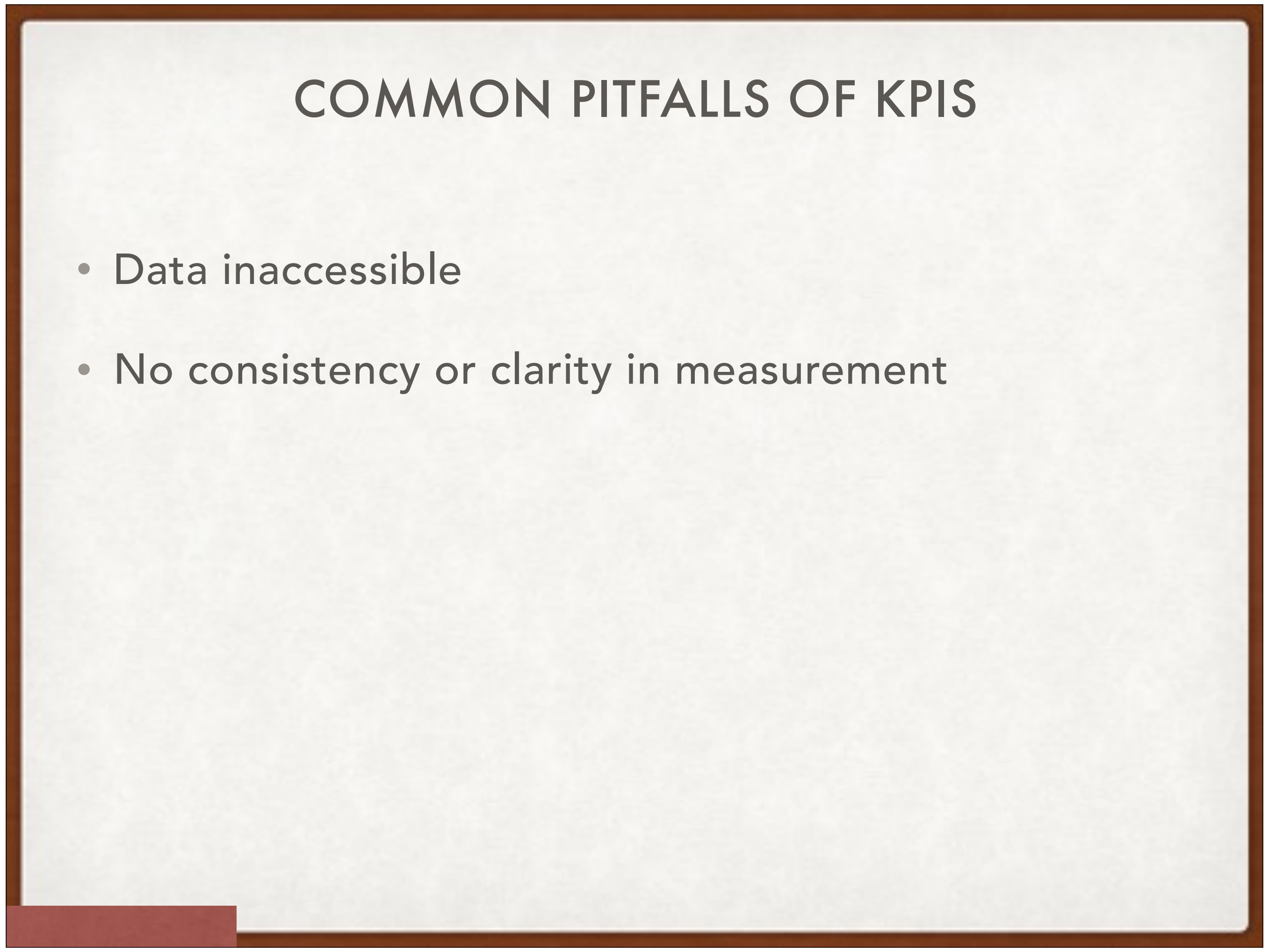
- A few clear, actionable measures with team buy-in
- Measures are used in decision making
- Measures track progress over time towards the strategic result



## KPI ANALYSIS

1. Do you have a well-formed, well-defined question? (Hypothesis)
2. Do you have data to answer your question? (Data)
3. Can the data answer the question (i.e., is the question being asked pertinent to the data available)? (Pertinence)
4. If you get an answer, is the answer relevant or meaningful? (Relevance)

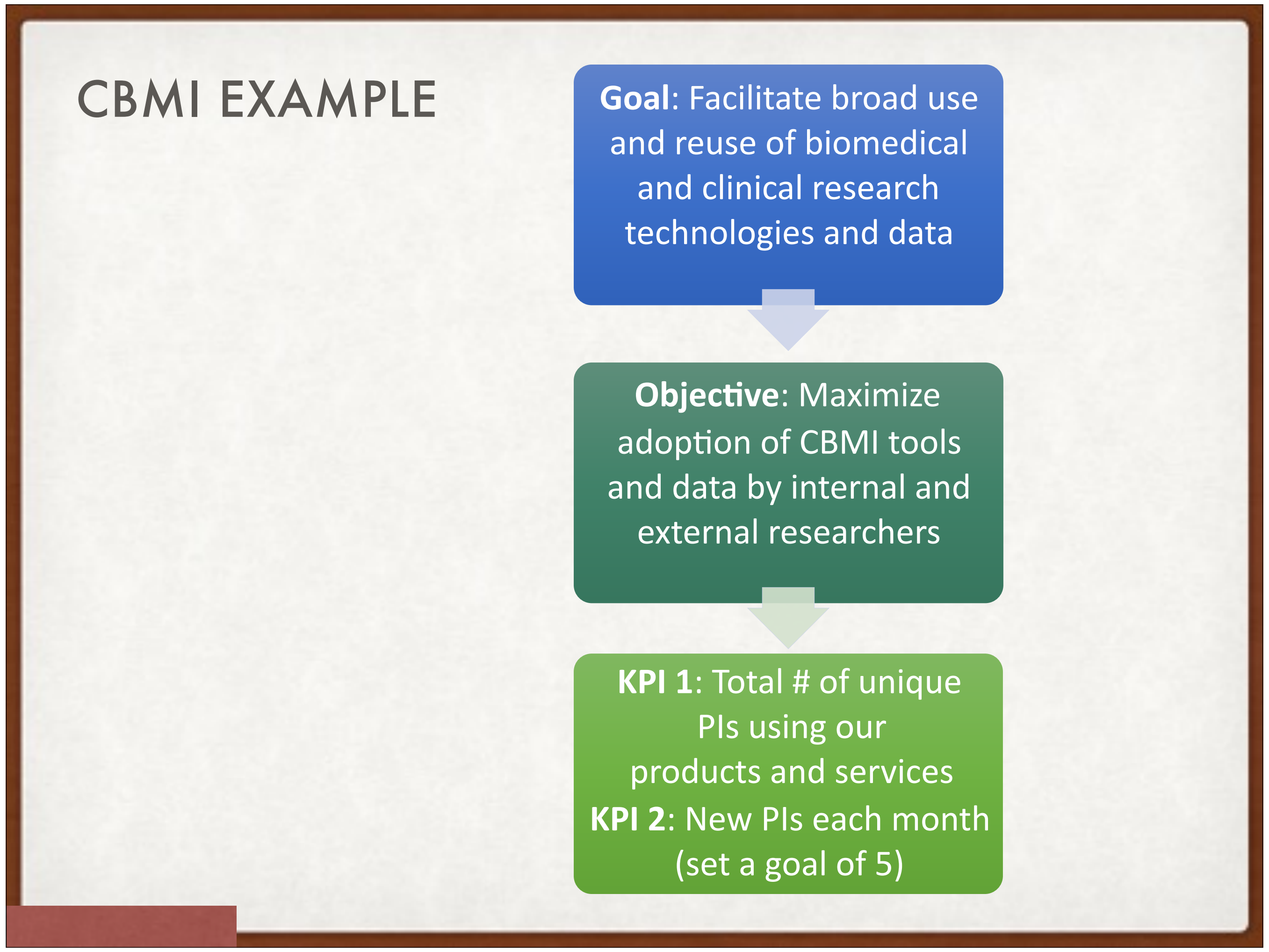
You can try to apply the research method to the KPIs you developed, following this approach:
For any research study one needs to answer multiple questions:
1. Do you have a well-formed, well-defined question? (Hypothesis)
2. Do you have data to answer your question? (Data) Is all the information (data, metadata, documentation) available to reuse the data for a study? (Replicability/Reproducibility) Are the data of sound quality to want to reuse for research purposes? (Quality)
3. Can the data answer the research hypothesis (i.e., is the question being asked pertinent to the data available)? (Pertinence) Or Can you collect data to answer your question? (Data)
4. If you get an answer, is the answer relevant or meaningful? (Relevance)  Cite the RDA paper this came from



## COMMON PITFALLS OF KPIS

- Data inaccessible
- No consistency or clarity in measurement

It may appear to you that after you have developed your list of KPIs, you just realized that not all of them have data readily available or easily accessible, which would make these KPIs hard to manage or not easily automated. It shouldn't cause you to abandon those KPIs: start monitoring those that have data available and look for ways to collect data on those that don't. It may take time, but that happens quite a lot. We ran into that situation at CBMI, where we really want to capture the number of publications and grants received by the researchers we work with as a measure of successful outcomes of CBMI, but we haven't been able to identify a fully reliable source of such information, but we are working on reviving our VIVO instance, so we could start capturing that information and mine the existing sources in an effective way. Another pitfall of KPIs is the assumptions about how to measure a particular KPI, how it's calculated, as different people may have different ideas. It may be useful to come up with a standardized Corporate Performance Measure Dictionary, where it's documented how data for particular measures are collected and analyzed.



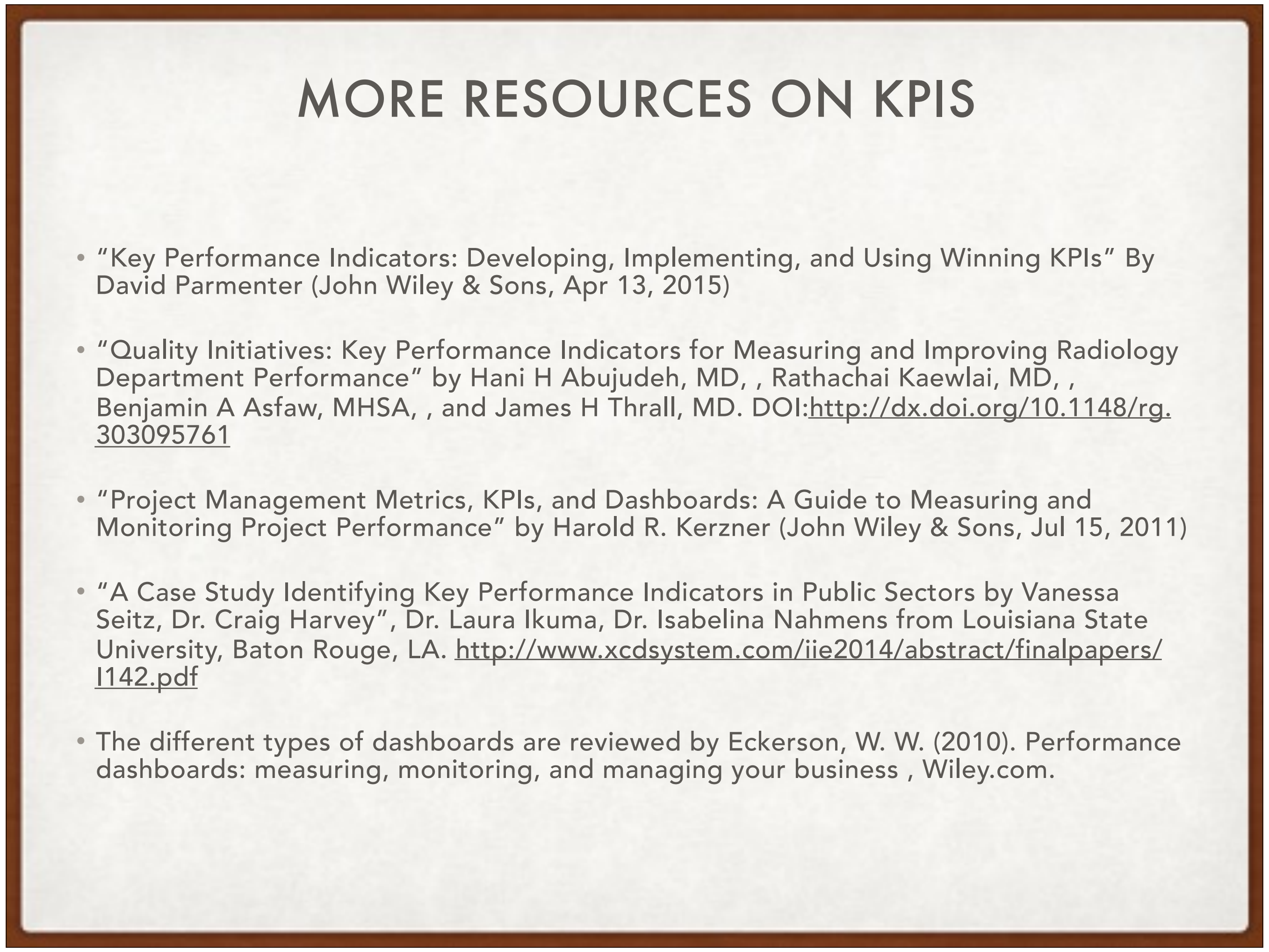
## CBMI EXAMPLE

**Goal**: Facilitate broad use and reuse of biomedical and clinical research technologies and data

**Objective**: Maximize adoption of CBMI tools and data by internal and external researchers

**KPI 1**: Total # of unique PIs using our products and services
**KPI 2**: New PIs each month (set a goal of 5)

Here's an example of the KPI development process we went through with CBMI. One of our goals is aligned with one of CTSA's goals: "Facilitate broad use and reuse of biomedical and clinical research digital assets (tools, technologies, datasets, and models) by making them discoverable, accessible, and citable to test new approaches that foster innovation in the real world". Based on that goal we developed an objective: to maximize adoption of CBMI tools and data by internal and external researchers, which would help achieve the goal above. And we came to the conclusion that an effective way to track progress on that objective is to track monthly the total cumulative number of unique PIs that CBMI has worked with – they either used our tools or services. Ultimately we would like to reach 90% of the Medical School internally, and externally, at least 30% of ICTS members.



## MORE RESOURCES ON KPIS

- “Key Performance Indicators: Developing, Implementing, and Using Winning KPIs” By David Parmenter (John Wiley & Sons, Apr 13, 2015)
- “Quality Initiatives: Key Performance Indicators for Measuring and Improving Radiology Department Performance” by Hani H Abujudeh, MD, , Rathachai Kaewlai, MD, , Benjamin A Asfaw, MHSA, , and James H Thrall, MD. DOI:http://dx.doi.org/10.1148/rg.303095761
- “Project Management Metrics, KPIs, and Dashboards: A Guide to Measuring and Monitoring Project Performance” by Harold R. Kerzner (John Wiley & Sons, Jul 15, 2011)
- “A Case Study Identifying Key Performance Indicators in Public Sectors by Vanessa Seitz, Dr. Craig Harvey”, Dr. Laura Ikuma, Dr. Isabelina Nahmens from Louisiana State University, Baton Rouge, LA. http://www.xcdsystem.com/iie2014/abstract/finalpapers/I142.pdf
- The different types of dashboards are reviewed by Eckerson, W. W. (2010). Performance dashboards: measuring, monitoring, and managing your business , Wiley.com.

1. David talks about 6 stages of migrating to performance improvement using KPIs, including getting stakeholder buy-in and identifying critical success factors. David argues that at the foundation of using KPIs correctly, among other things, lies the concept of measuring and reporting only what matters.
2. The article describes the experience of a large academic radiology department in formulating and implementing a list of radiology-specific KPIs aligned with the institutional vision, strategies, and goals. The departmental leadership identified 67 radiology-specific KPIs and 125 measurement parameters.
3. 
4. Authors present a step-by-step walkthrough of how KPIs were identified for Louisiana’s Department of Health and Hospitals and then sorted into several categories, to be displayed on corresponding dashboards.

8.

## KPIS AND DASHBOARD DESIGN

### MAIN PRINCIPLES OF DASHBOARD DESIGN BY STEPHEN FEW

- 7 +/- 2 elements
- One screen, no scrolling
- A dashboard is a utility not a piece of art -> little embellishments
- Mild color-, pattern- encoding

## DASHBOARD TYPES

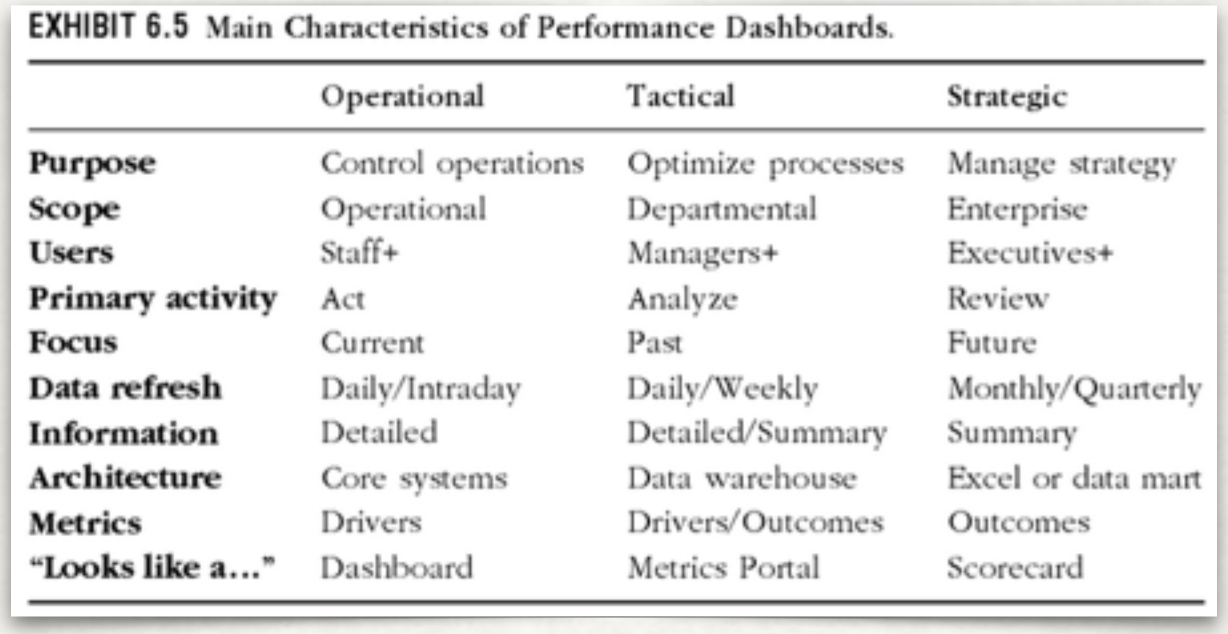


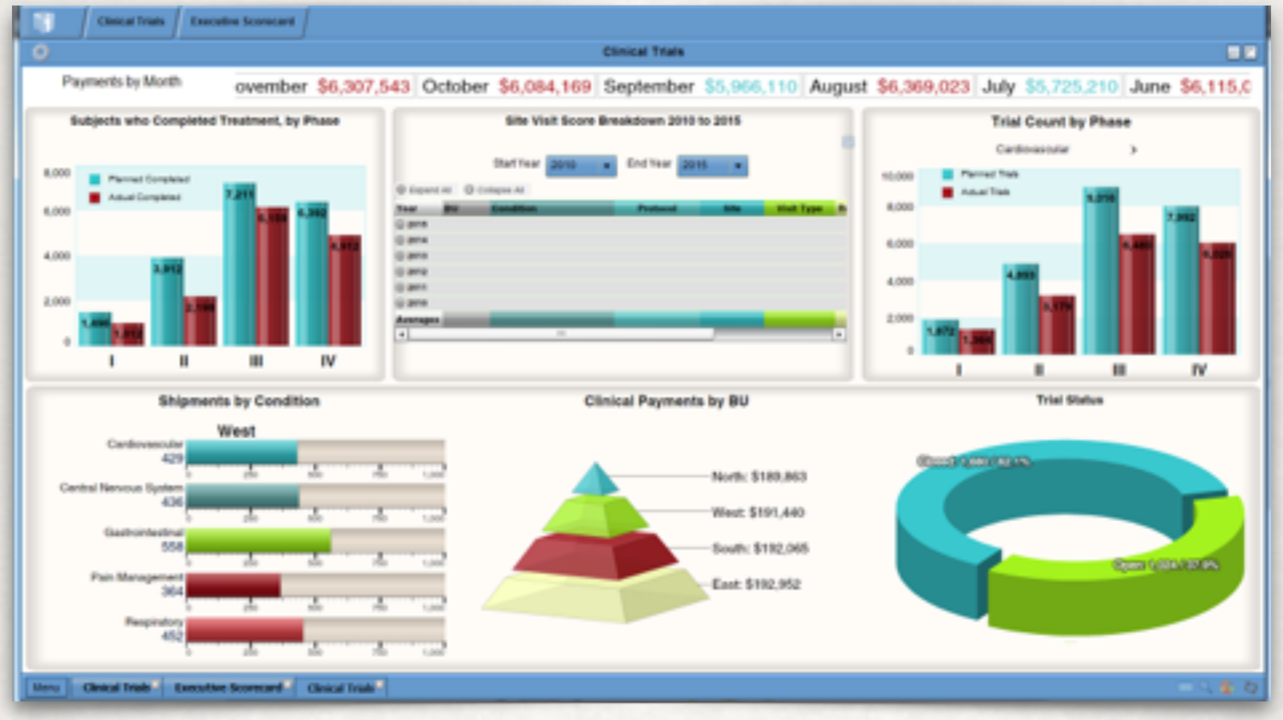
**EXHIBIT 6.5** Main Characteristics of Performance Dashboards.

| | Operational | Tactical | Strategic |
|---|---|---|---|
| **Purpose** | Control operations | Optimize processes | Manage strategy |
| **Scope** | Operational | Departmental | Enterprise |
| **Users** | Staff+ | Managers+ | Executives+ |
| **Primary activity** | Act | Analyze | Review |
| **Focus** | Current | Past | Future |
| **Data refresh** | Daily/Intraday | Daily/Weekly | Monthly/Quarterly |
| **Information** | Detailed | Detailed/Summary | Summary |
| **Architecture** | Core systems | Data warehouse | Excel or data mart |
| **Metrics** | Drivers | Drivers/Outcomes | Outcomes |
| **"Looks like a…"** | Dashboard | Metrics Portal | Scorecard |

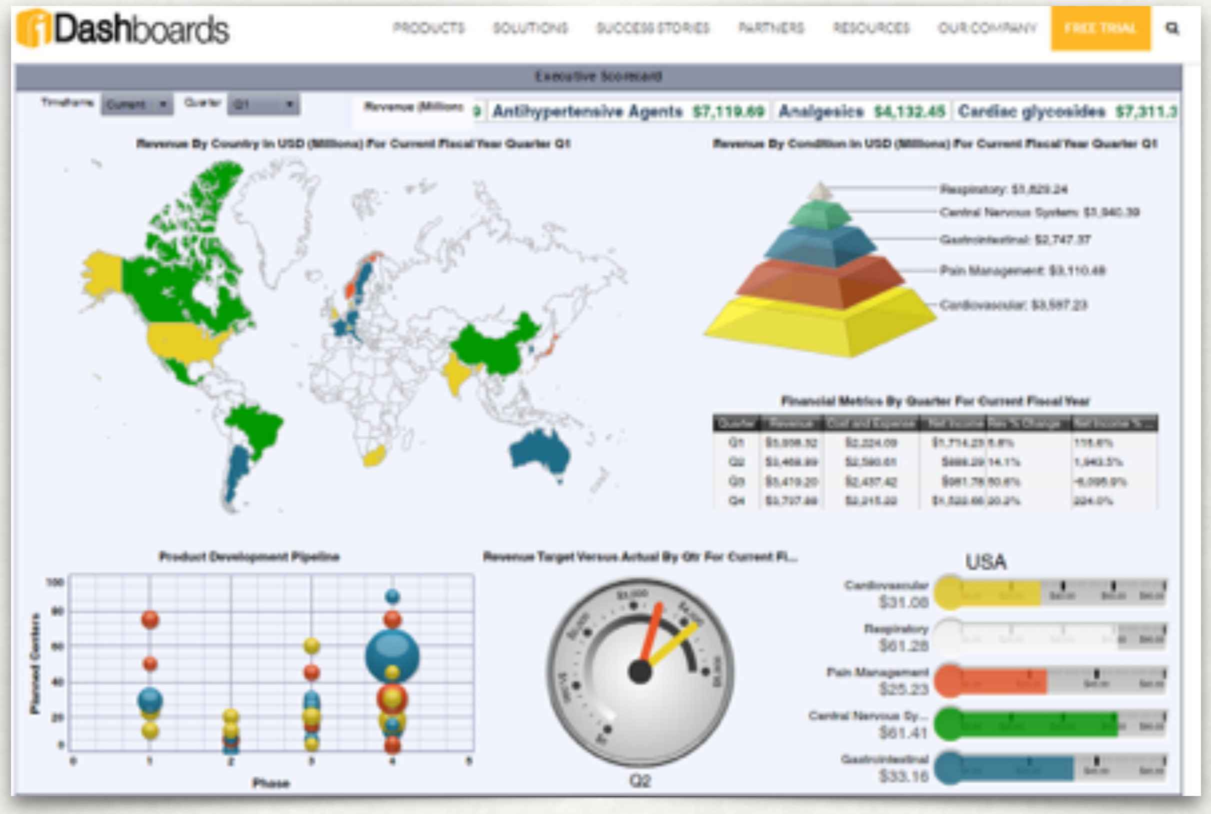
"Performance Dashboards : Measuring, Monitoring, and Managing Your Business" by Eckerson, Wayne W..

The different types of dashboards are reviewed by Eckerson, W. W. (2010).
Different types of dashboards can dictate specific design requirements
Performance
dashboards: measuring, monitoring, and managing your business , Wiley.com.
Eckerson, Wayne W.. Performance Dashboards : Measuring, Monitoring, and Managing Your Business (2). Hoboken, US: Wiley, 2010. ProQuest ebrary. Web. 17 August 2016.

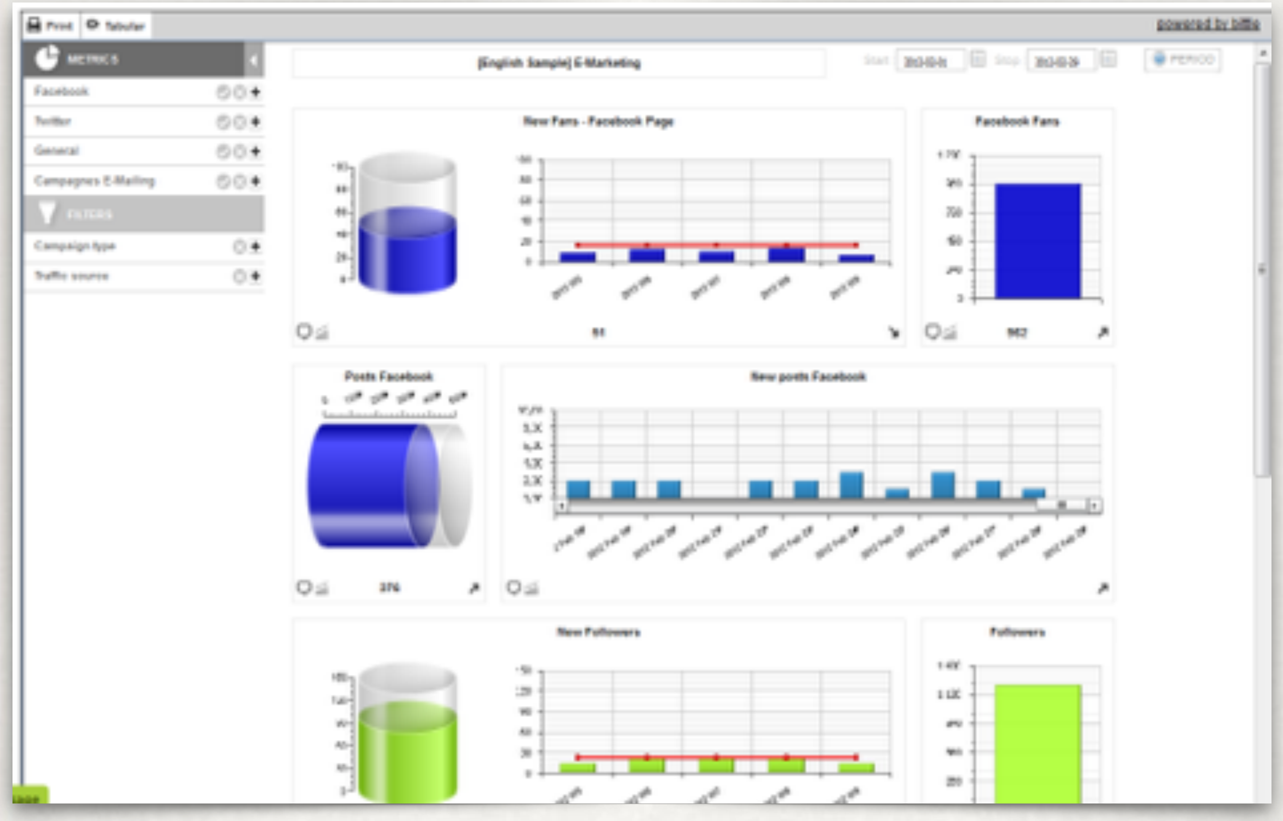
## EXAMPLES OF POOR DASHBOARDS

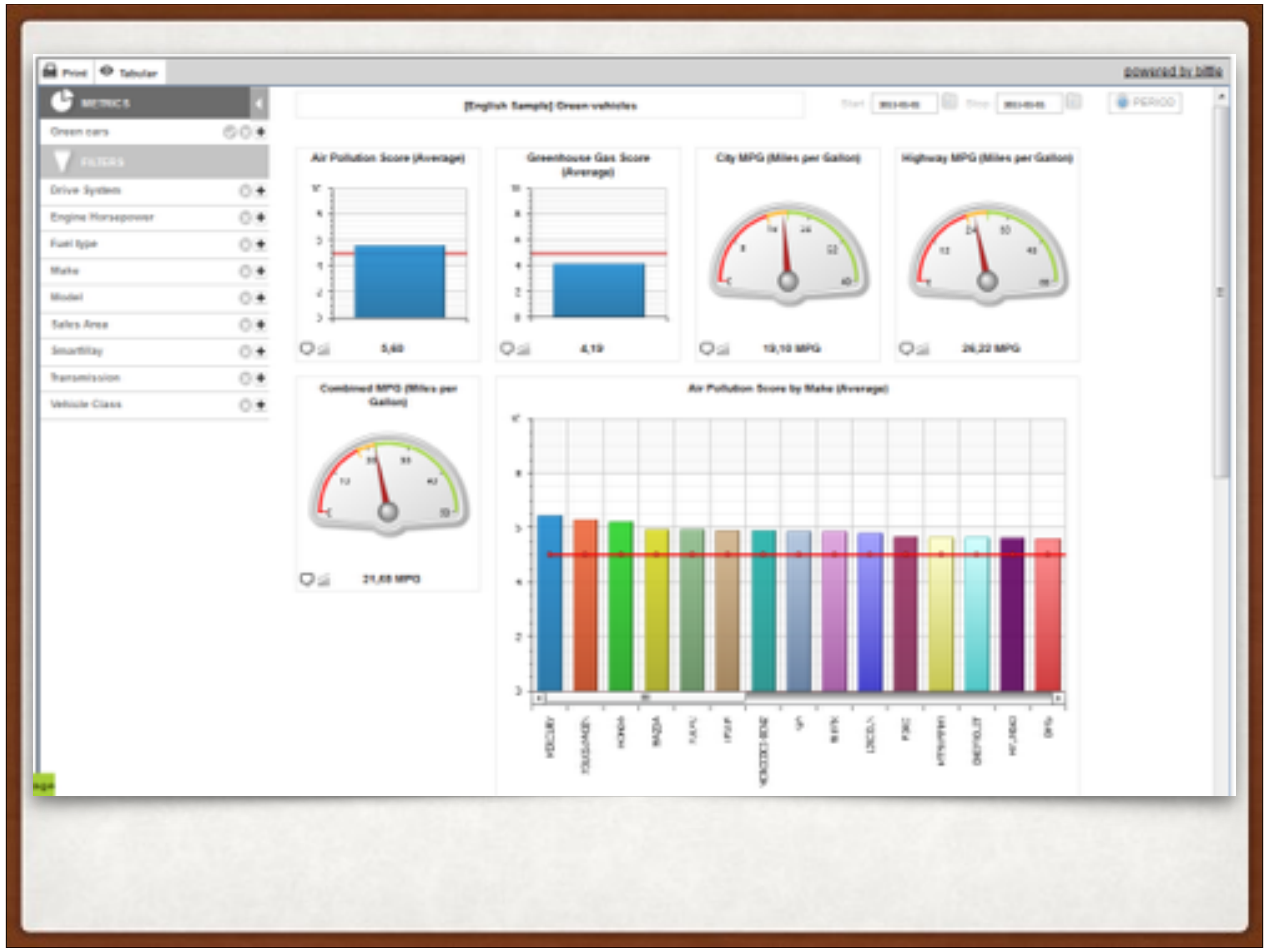


http://gallery.idashboards.com/preview/?guestuser=webpharm&dashID=89


iDashboards
PRODUCTS
SOLUTIONS
SUCCESS STORIES
PARTNERS
RESOURCES
OUR COMPANY
FREE TRIAL
Executive Scorecard
Antihypertensive Agents $7,119.69
Analgesics $4,132.45
Cardiac glycosides $7,311.3
Revenue By Country In USD (Millions) For Current Fiscal Year Quarter Q1
Revenue By Condition In USD (Millions) For Current Fiscal Year Quarter Q1
Respiratory: $1,829.24
Central Nervous System: $1,940.39
Gastrointestinal: $2,747.37
Pain Management: $3,110.48
Cardiovascular: $3,597.23
Financial Metrics By Quarter For Current Fiscal Year
Product Development Pipeline
Planned Centers
Phase
Revenue Target Versus Actual By Qtr For Current Fi...
Q2
USA
Cardiovascular $31.08
Respiratory $61.28
Pain Management $25.23
Central Nervous Sy... $61.41
Gastrointestinal $33.16

http://www.idashboards.com/live-dashboard-examples/pharmaceutical-biotech-scorecard-dashboard/

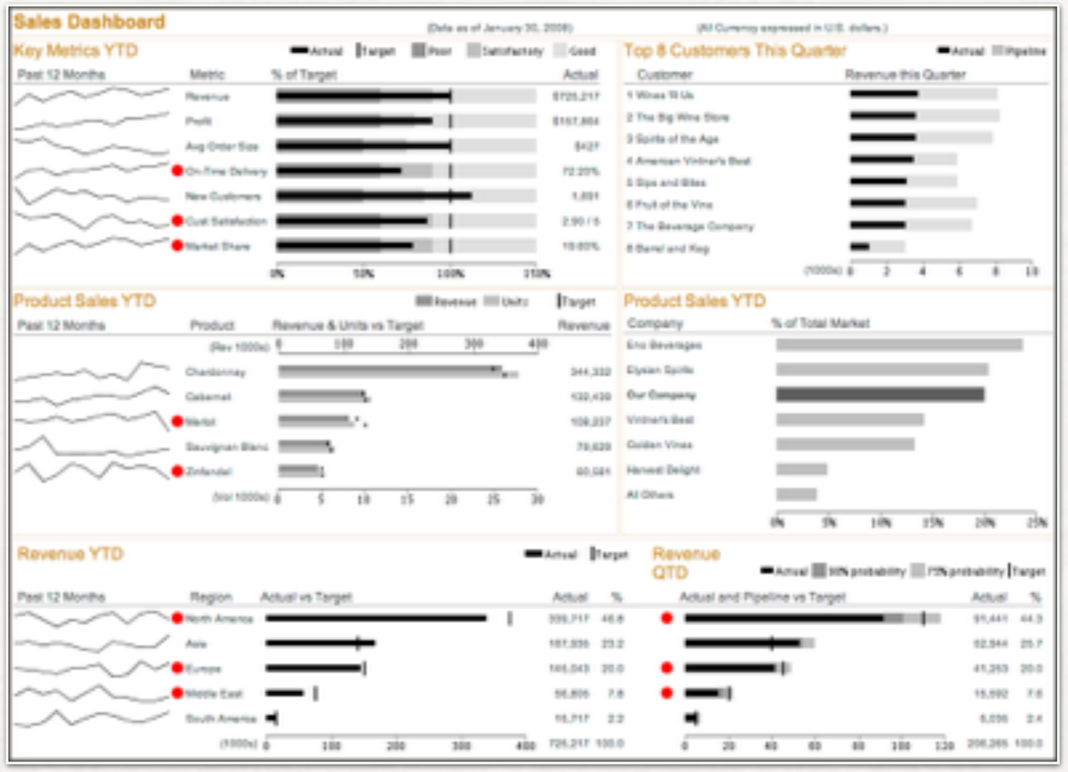

Print
Tabular
powered by bittle
METRICS
Facebook
Twitter
General
Campagnes E-Mailing
FILTERS
Campaign type
Traffic source
[English Sample] E-Marketing
PERIOD
New Fans - Facebook Page
61
Facebook Fans
962
Posts Facebook
376
New posts Facebook
New Followers
Followers


Print
Tabular
powered by bime
METRICS
Green cars
FILTERS
Drive System
Engine Horsepower
Fuel type
Make
Model
Sales Area
SmartWay
Transmission
Vehicle Class
[English Sample] Green vehicles
PERIOD
Air Pollution Score (Average)
Greenhouse Gas Score (Average)
City MPG (Miles per Gallon)
Highway MPG (Miles per Gallon)
5,60
4,19
19,10 MPG
26,22 MPG
Combined MPG (Miles per Gallon)
21,68 MPG
Air Pollution Score by Make (Average)

Observation from trying to find bad dashboard examples.

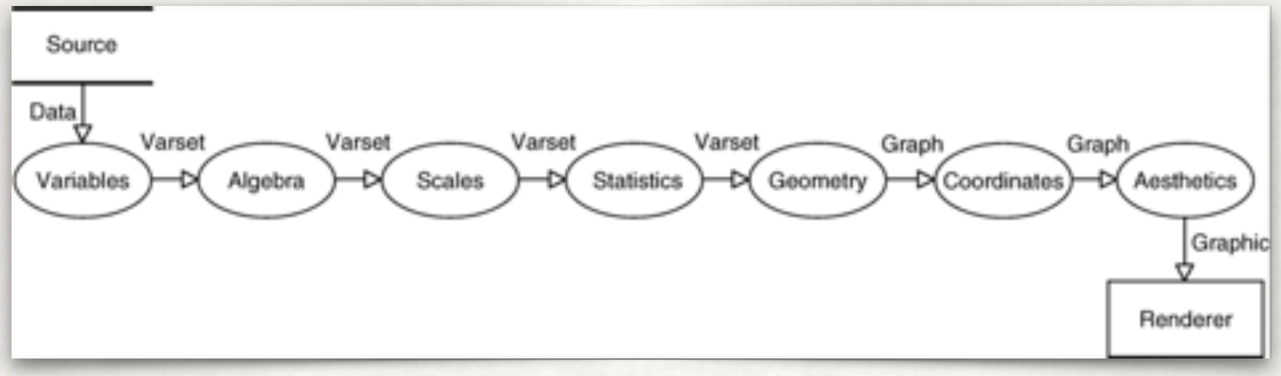
Over the past several years  as we've been looking for bad dashboard examples, we've noticed that dashboard companies have drastically improved, they use slick and minimalistic design, following most of best practices. Specifically, Domo and ClicData have removed their bad dashboard samples, and Geckoboard has switched focus to simple colors, numbers and lines. So good trends now prevail.

## EXAMPLE OF AN EFFECTIVE DASHBOARD



by Stephen Few

## BRIEF INTRO TO GRAMMAR OF GRAPHICS



“The Grammar of Graphics” by Wilkinson, Anand, and Grossman (2005).

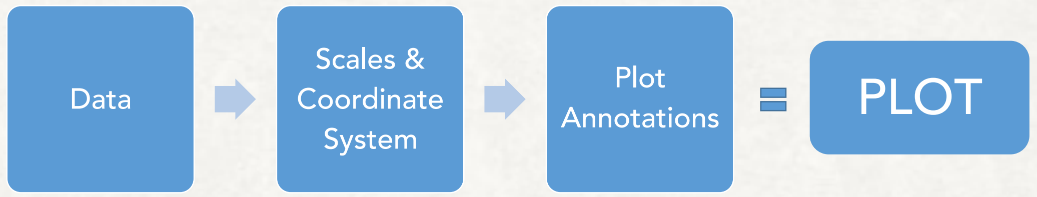
## WHY GRAMMAR OF GRAPHICS?

- Grammar
  *fundamental principles or rules of an art or science*
- Grammar of Graphics helps
  - understand complex graphics
  - draw connections between several graphics
  - construct a wide range of graphics

One way to answer questions about statistical graphics is to develop a grammar: “the fundamental principles or rules of an art or science” (OED Online 1989). A good grammar will allow us to gain insight into the composition of complicated graphics, ad reveal unexpected connections between seemingly different graphics. (Cox 1979)

## LAYERED GRAMMAR OF GRAPHICS

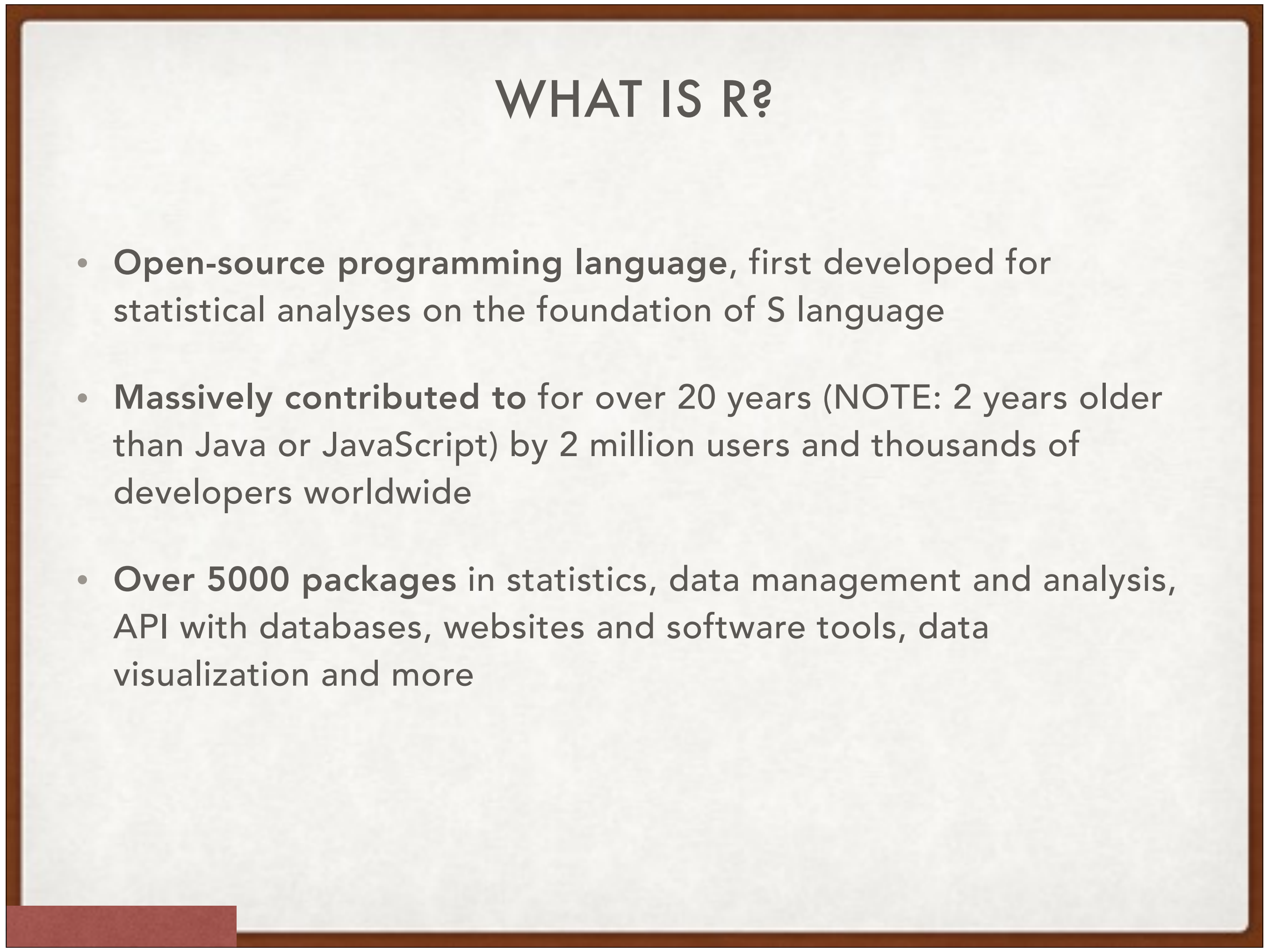
- Aesthetics - things we can perceive on the graphic:
  - positions of data points
  - bars, lines, points (geometric objects)
- Scaling – mapping numeric data points to coordinates on the screen/display media



*x*-position, *y*-position, and shape are examples of aesthetics, things that we can perceive on the graphic.
Bars, lines, and points are all examples of geometric objects.
The results of these scalings are shown in Table 3. These transformations
are the responsibility of *scales*,
To create a complete plot we need to combine graphical objects from
three sources: the *data*, represented by the point geom; the *scales and coordinate system*,
which generates axes and legends so that we can read values from the graph; and the
*plot annotations*, such as the background and plot title.

## PART (III)

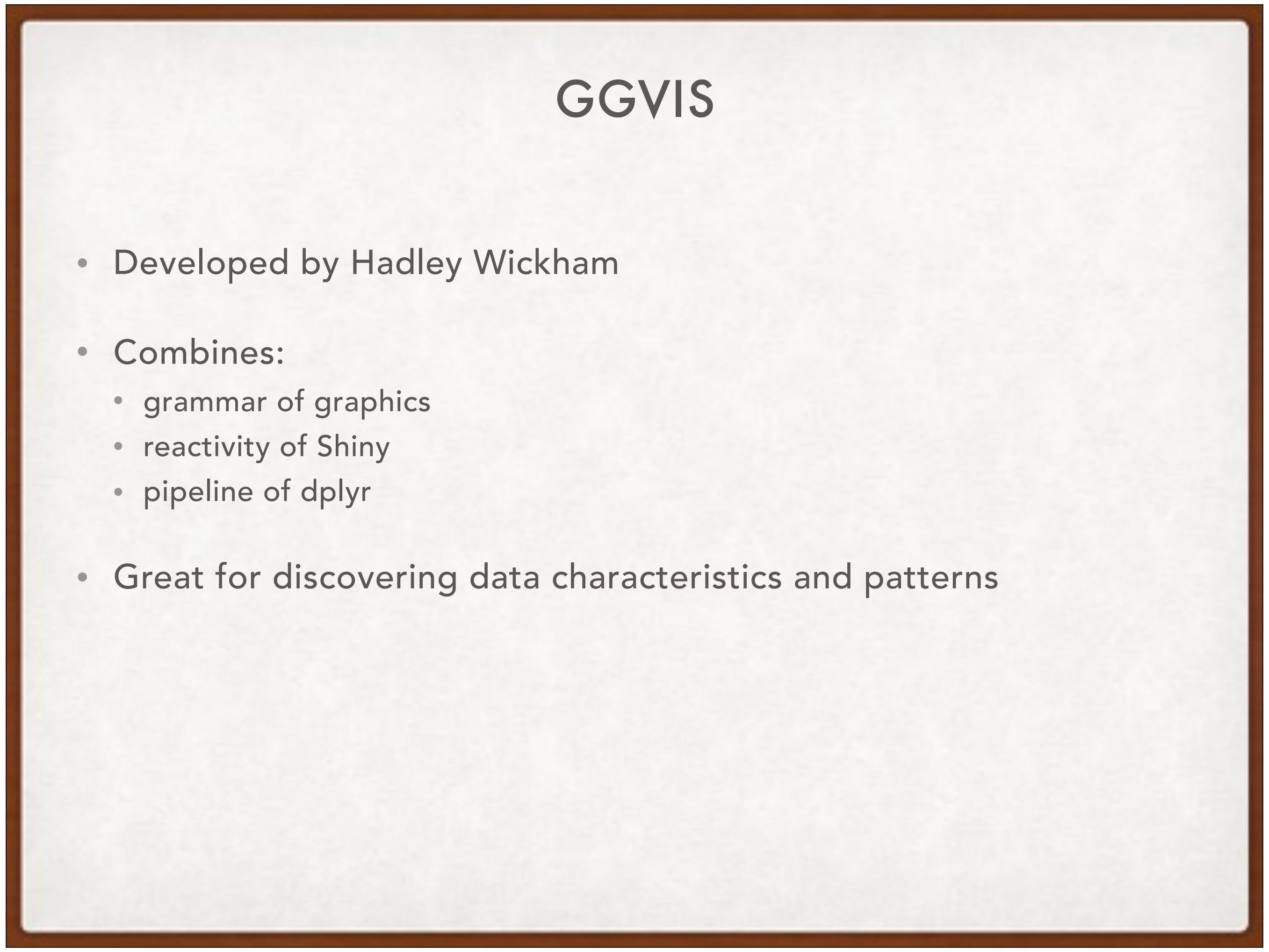
(TECHNICALLY ) PRESENTING DATA



## WHAT IS R?

- **Open-source programming language**, first developed for statistical analyses on the foundation of S language
- **Massively contributed to** for over 20 years (NOTE: 2 years older than Java or JavaScript) by 2 million users and thousands of developers worldwide
- **Over 5000 packages** in statistics, data management and analysis, API with databases, websites and software tools, data visualization and more

Although some organizations may not use it on production, since it's open-source, it is a great tool to be used internally. There are however, fully-supported versions of R offered by Oracle, Revolution Analytics, RStudio and some other providers, which will guarantee a reliable performance worthy of Production level.
Some of you may point out limitations of R, such as by default it's single-thread and loads all data in memory. However, this limitation has been overcome by developers of packages that allow R to support parallelization of processes which helps when you work with large amounts of data (over 1.5 mln rows).

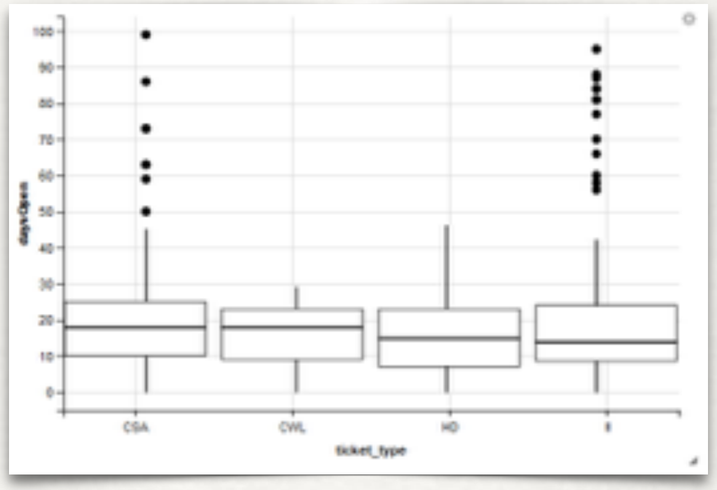


## GGVIS

- Developed by Hadley Wickham
- Combines:
  - grammar of graphics
  - reactivity of Shiny
  - pipeline of dplyr
- Great for discovering data characteristics and patterns

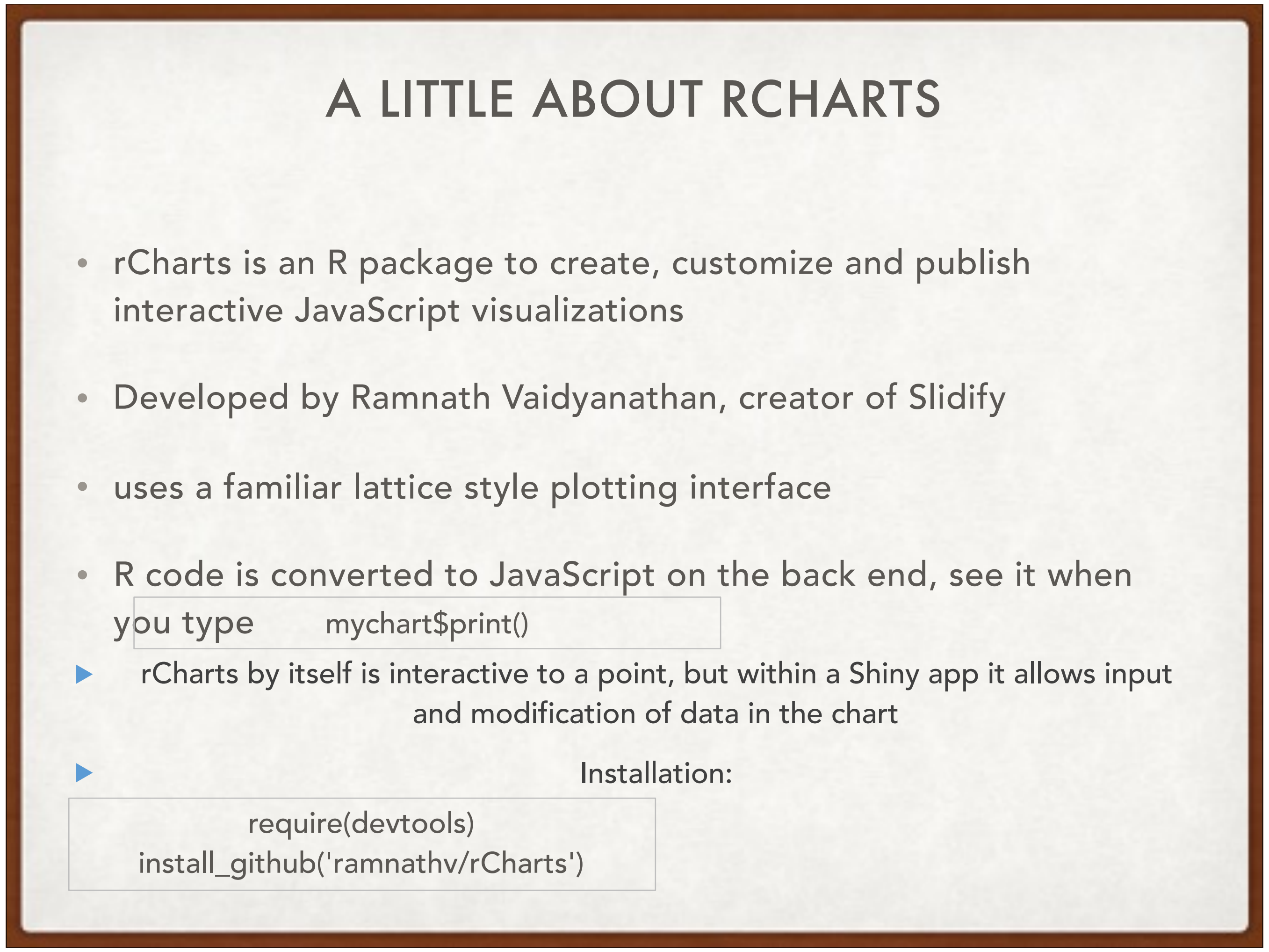
ggvis package allows for interactive graphics in R, which makes exploration of data much easier. ggvis combines concepts of grammar of graphics in ggplot2, uses reactive functions and filters of shiny and easy-to-read data transformation pipeline of dplyr.

## GGVIS (CONT.)



```
# read in JIRA data
opsdata <- read.csv("data/opsdata.csv")

# Example of a static ggvis chart
opsdata[opsdata$daysOpen < 100,] %>%
  ggvis(~ticket_type, ~daysOpen) %>%
  layer_boxplots()
```

Here you see how to construct a visualization piping pieces together. First pass the dataset to ggvis() function, specifying the x and y variables of a boxplot. Then the result is plotted using layer_boxplots() function, instead of which a number of other functions could have been used (layer_bars(), layer_lines(), layer_rects(), etc.)
However, this is a static example. Elements of Shiny can be used with ggvis, which is helpful in discovering data, for example you can add filters - widgets (a date range, a drop down, text or numeric box):
ggvis is great to use to discover patterns and relationships in data, before the KPI dashboard is built. However, due to the lack of interactivity of individual data points, this package may not be an ideal choice for the dashboard itself. It’s much better to use a JavaScript based library, such as one of the rCharts libraries, or googleVis.

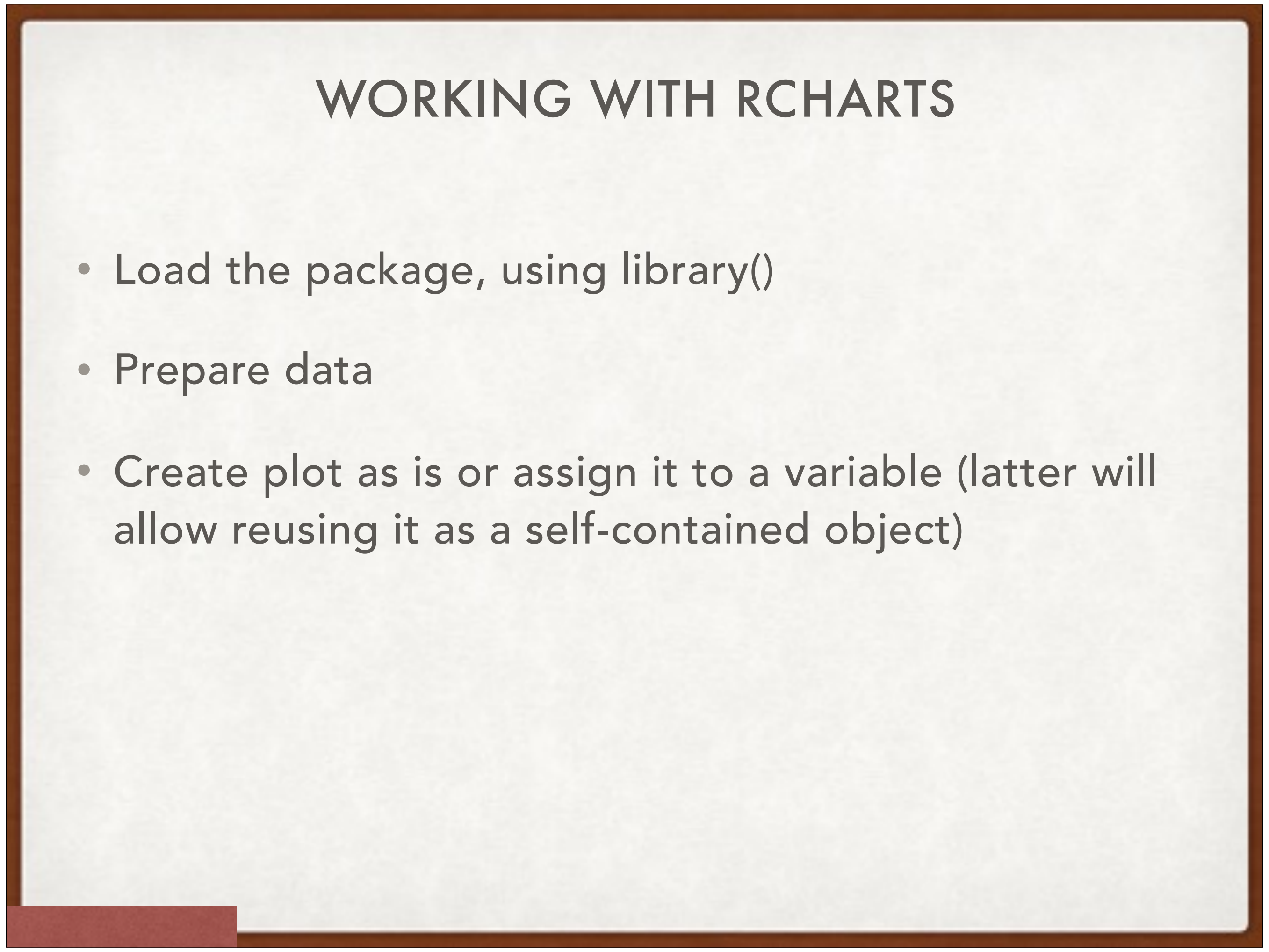


## A LITTLE ABOUT RCHARTS

- rCharts is an R package to create, customize and publish interactive JavaScript visualizations
- Developed by Ramnath Vaidyanathan, creator of Slidify
- uses a familiar lattice style plotting interface
- R code is converted to JavaScript on the back end, see it when you type `mychart$print()`
- rCharts by itself is interactive to a point, but within a Shiny app it allows input and modification of data in the chart
- Installation:

```
require(devtools)
install_github('ramnathv/rCharts')
```

Unlike many fundamental visualization packages in R (base, ggplot2, ggvis), rCharts gives you a much higher level of precision through its interactivity – you hover over a point of data and it tells you whatever you want it to tell you (for the most part).

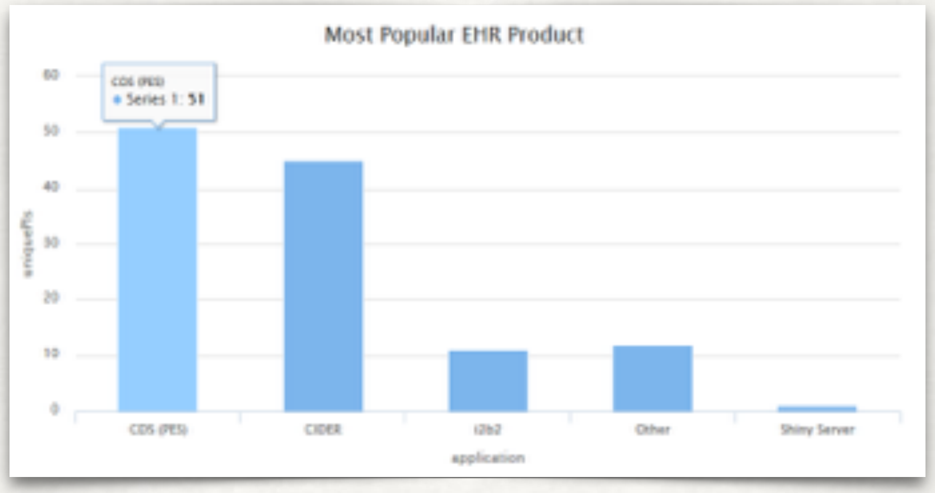


## WORKING WITH RCHARTS

- Load the package, using library()
- Prepare data
- Create plot as is or assign it to a variable (latter will allow reusing it as a self-contained object)

Working with rCharts is like working with any other package in R, you first need to load it, prepare the data to be in the suitable format (rows, columns, calculations), and then run a function from the package. The result of that function will be displayed in the viewer, or you can assign the result to a variable, so it can be reused multiple times.

## INTERACTIVE CHART



```
library(rCharts); library(plyr)
opsdata <- read.csv("data/opsdata.csv")
PI_by_product <- ddply(opsdata[opsdata$ticket_type == 'II',], c("application"), summarise,
uniquePIs = length(unique(pi_name)))
mostPopularProduct <- hPlot(uniquePIs ~ application, type = 'column', data = PI_by_product,
                        title = "Most Popular EHR Product")
mostPopularProduct
```

Note about the data: the data used in the following examples are a sample of real dataset from operational database of CBMI that have been altered significantly for this demonstration; only the names of Products are real.
Let's look at the code now. If you look at the data for this chart (peek at next slide), you will see that the original dataset is not very suitable for the purposes of this chart, because the same PI can be mentioned several times per one application, and we want only unique cases of them using that application. So we use ddply function to manipulate the data first and get to those unique cases. The results can be seen in the second table (peek at next slide). In the function we specify what to plot – the x and y axes, which you can do in two ways – by explicitly saying "x=.." and "y=…" or you can use tilde sign and reverse the order. As here your numeric value is in first position and the categories are in the second position. You also need to specify the type of chart you wish to display – here it's "column" and then what the dataset is called, as well as the title you prefer. Notice that the R objects such as dataset name and column/variable names are not in quotation marks, while your values for properties are. After that to display your chart in the viewer window or browser you just call the object (this will be different for when you use them in a dashboard).

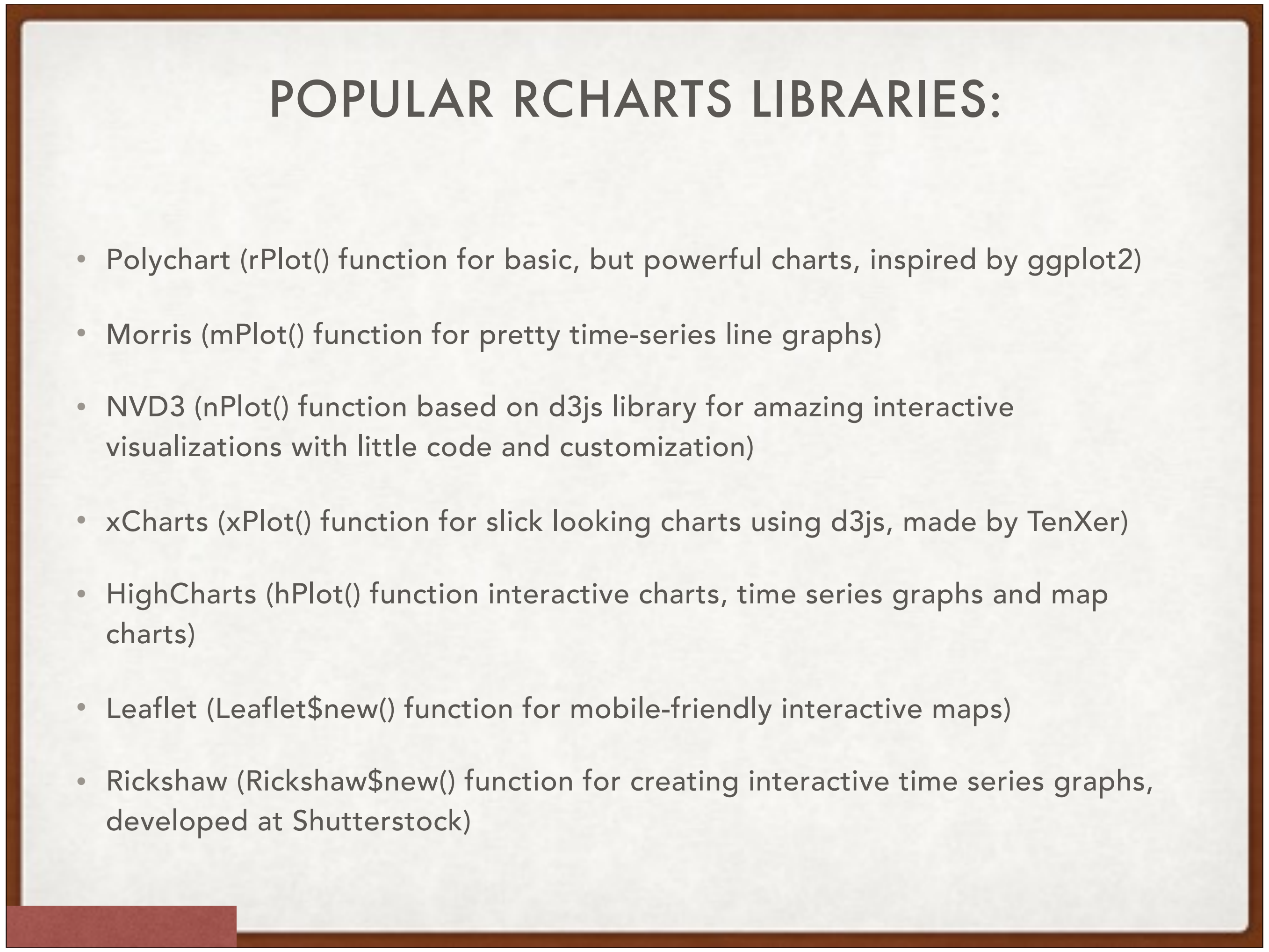
## DATA FOR THE INTERACTIVE CHART ABOVE

Preview of part of opsdata

| id | created | status | ticket_type | title | PI_name | FundingType | Application |
|---|---|---|---|---|---|---|---|
| 1 | 1/2/2015 | Closed | HD | Ticket 1 | Elvis Daniels | Department Funds | CPTAC |
| 2 | 1/2/2015 | Closed | HD | Ticket 2 | Scott Lott | Not billable | RPR-ClinPortal |
| 3 | 1/5/2015 | Closed | HD | Ticket 3 | Cassidy Griffith | Department Funds | BioMS |
| 4 | 1/5/2015 | Closed | HD | Ticket 4 | Mollie Chang | Department Funds | CIDER |

Preview of PI_by_product

| application | uniquePIs |
|---|---|
| CDS (PES) | 51 |
| CIDER | 45 |
| i2b2 | 11 |
| Other | 12 |

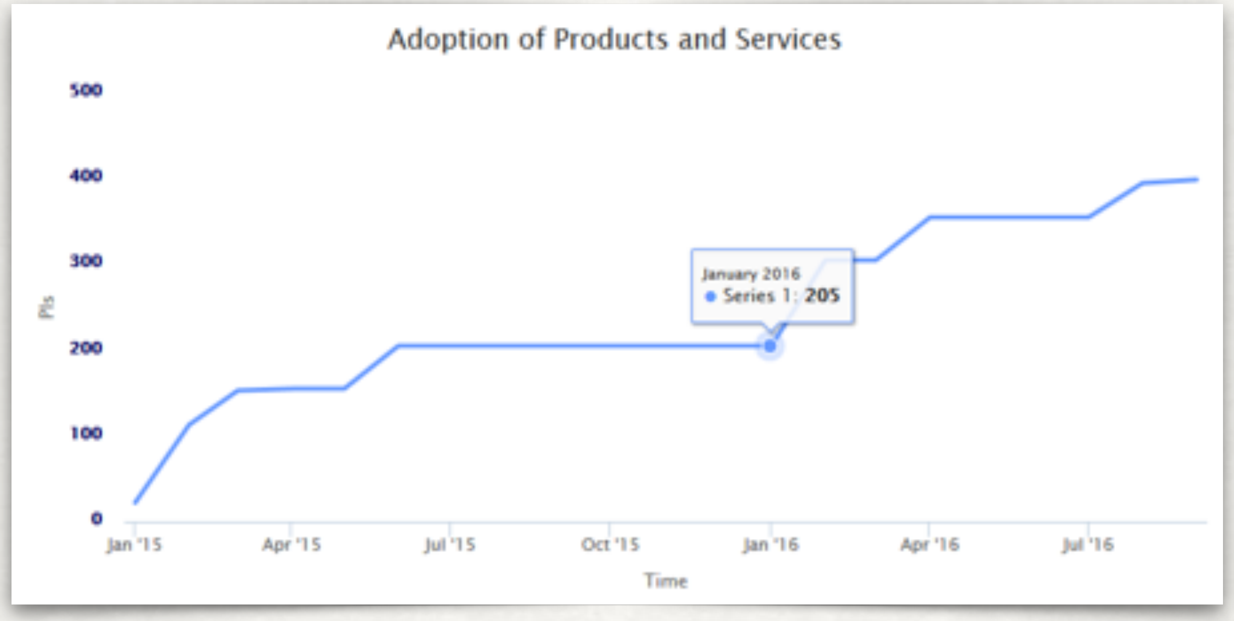


## POPULAR RCHARTS LIBRARIES:

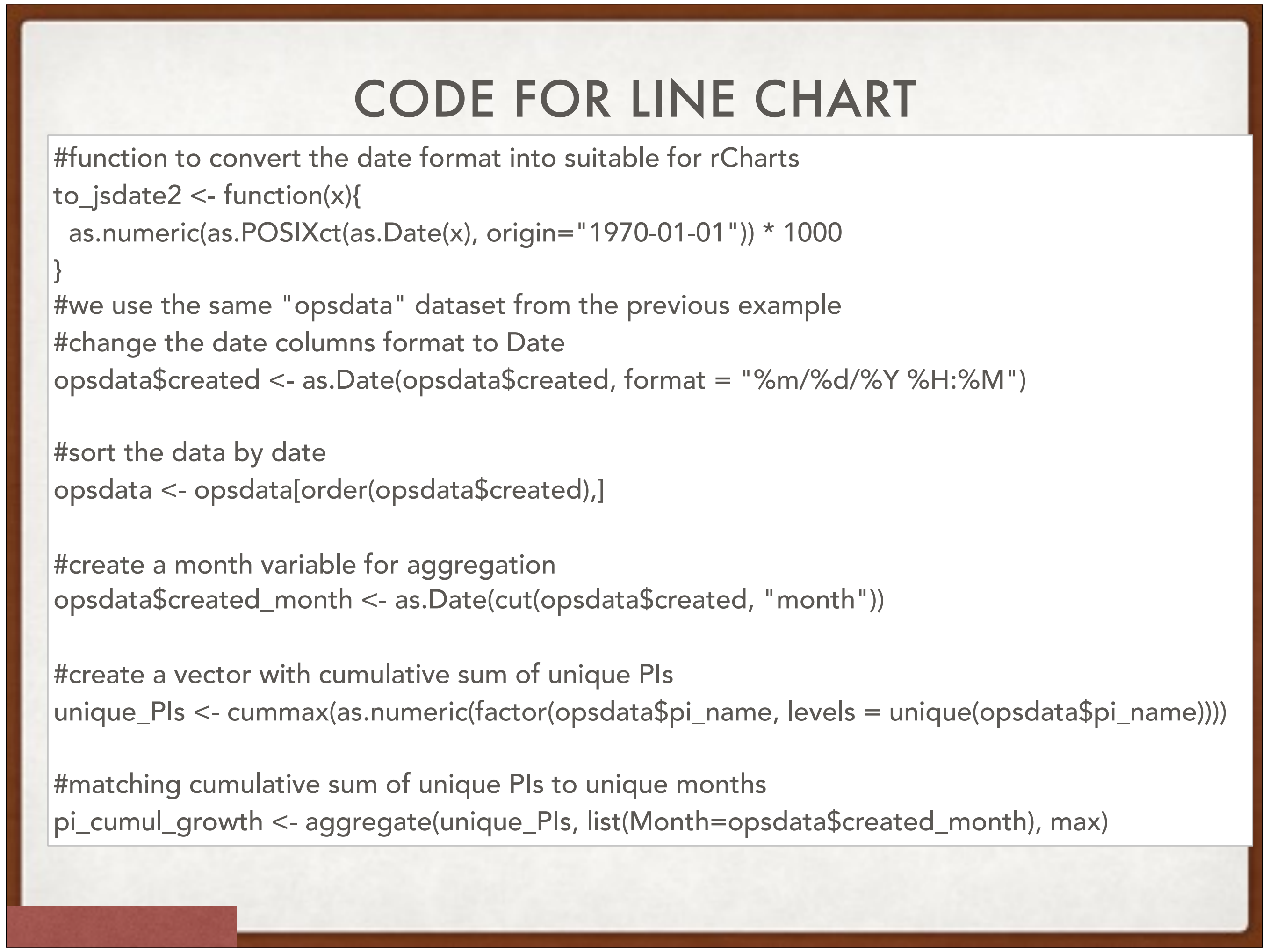
- Polychart (rPlot() function for basic, but powerful charts, inspired by ggplot2)
- Morris (mPlot() function for pretty time-series line graphs)
- NVD3 (nPlot() function based on d3js library for amazing interactive visualizations with little code and customization)
- xCharts (xPlot() function for slick looking charts using d3js, made by TenXer)
- HighCharts (hPlot() function interactive charts, time series graphs and map charts)
- Leaflet (Leaflet$new() function for mobile-friendly interactive maps)
- Rickshaw (Rickshaw$new() function for creating interactive time series graphs, developed at Shutterstock)

On this slide you can see the popular rCharts libraries. The chart we just saw is built using the HighCharts library, which is perhaps the most customizable out of them all.

## LINE CHART WITH CUSTOMIZATION



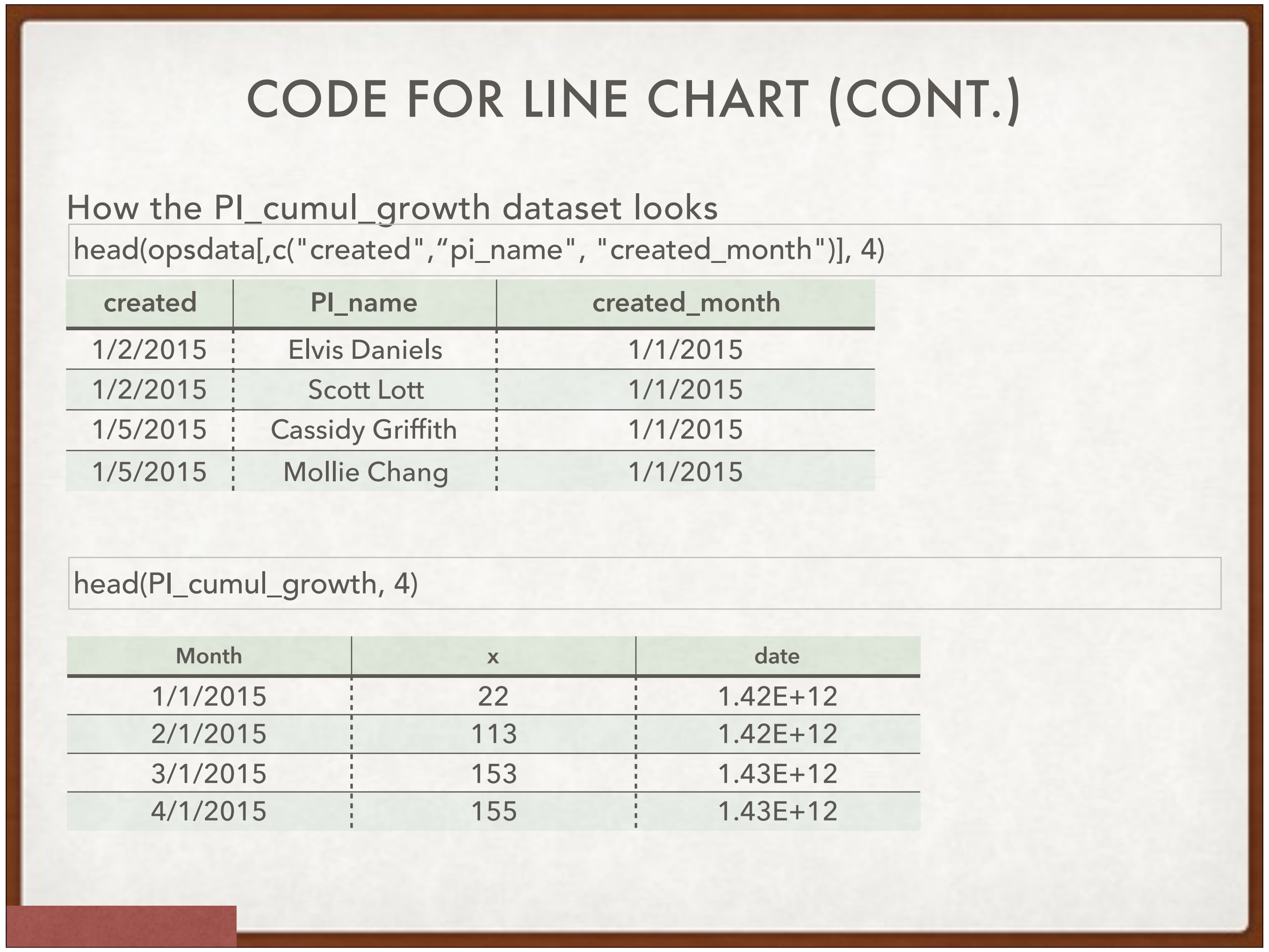
This example is a little more complex than the first one, because it involves a line chart and a lot of customization, which we will walk you through. First is a time series chart that shows cumulative growth of PIs that have worked CBMI's products and services over the last several years.



## CODE FOR LINE CHART

```
#function to convert the date format into suitable for rCharts
to_jsdate2 <- function(x){
  as.numeric(as.POSIXct(as.Date(x), origin="1970-01-01")) * 1000
}
#we use the same "opsdata" dataset from the previous example
#change the date columns format to Date
opsdata$created <- as.Date(opsdata$created, format = "%m/%d/%Y %H:%M")

#sort the data by date
opsdata <- opsdata[order(opsdata$created),]

#create a month variable for aggregation
opsdata$created_month <- as.Date(cut(opsdata$created, "month"))

#create a vector with cumulative sum of unique PIs
unique_PIs <- cummax(as.numeric(factor(opsdata$pi_name, levels = unique(opsdata$pi_name))))

#matching cumulative sum of unique PIs to unique months
pi_cumul_growth <- aggregate(unique_PIs, list(Month=opsdata$created_month), max)
```

You can look through this code as it's posted on github link. But the main take-home points are: 1) you need to adjust the date to be in suitable format for rCharts (because it's JavaScript based) and 2) you need to end up with 2 columns of data – one with your numeric data and second with the dates.



## CODE FOR LINE CHART (CONT.)

How the PI_cumul_growth dataset looks

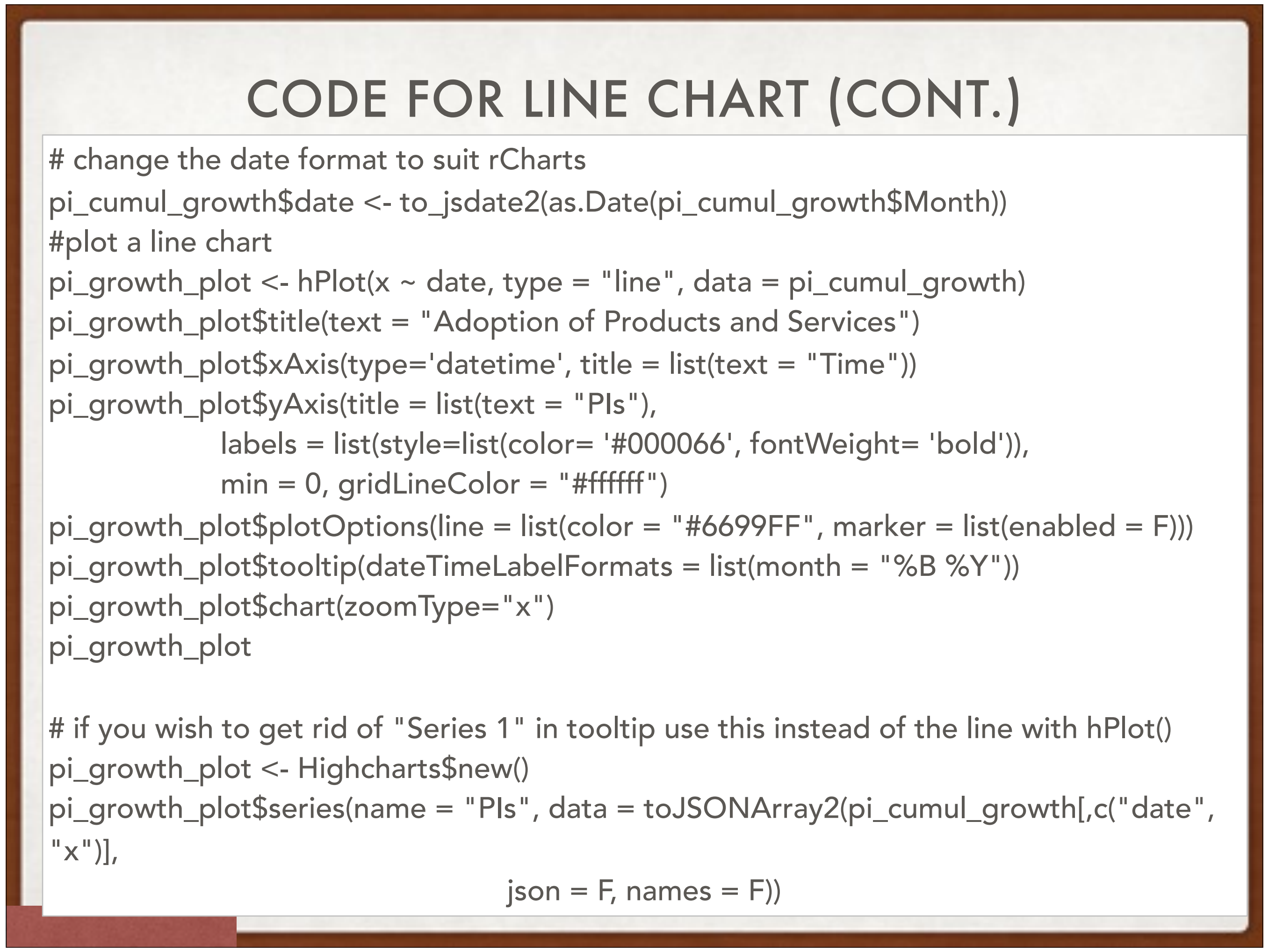
```
head(opsdata[,c("created",“pi_name", "created_month")], 4)
```

| created | PI_name | created_month |
|---|---|---|
| 1/2/2015 | Elvis Daniels | 1/1/2015 |
| 1/2/2015 | Scott Lott | 1/1/2015 |
| 1/5/2015 | Cassidy Griffith | 1/1/2015 |
| 1/5/2015 | Mollie Chang | 1/1/2015 |

```
head(PI_cumul_growth, 4)
```

| Month | x | date |
|---|---|---|
| 1/1/2015 | 22 | 1.42E+12 |
| 2/1/2015 | 113 | 1.42E+12 |
| 3/1/2015 | 153 | 1.43E+12 |
| 4/1/2015 | 155 | 1.43E+12 |

This is how the data look like now, and the two variables we will use are Month and X (the cumulative total number of unique PIs in that month).



## CODE FOR LINE CHART (CONT.)

```
# change the date format to suit rCharts
pi_cumul_growth$date <- to_jsdate2(as.Date(pi_cumul_growth$Month))
#plot a line chart
pi_growth_plot <- hPlot(x ~ date, type = "line", data = pi_cumul_growth)
pi_growth_plot$title(text = "Adoption of Products and Services")
pi_growth_plot$xAxis(type='datetime', title = list(text = "Time"))
pi_growth_plot$yAxis(title = list(text = "PIs"),
                labels = list(style=list(color= '#000066', fontWeight= 'bold')),
                min = 0, gridLineColor = "#ffffff")
pi_growth_plot$plotOptions(line = list(color = "#6699FF", marker = list(enabled = F)))
pi_growth_plot$tooltip(dateTimeLabelFormats = list(month = "%B %Y"))
pi_growth_plot$chart(zoomType="x")
pi_growth_plot

# if you wish to get rid of "Series 1" in tooltip use this instead of the line with hPlot()
pi_growth_plot <- Highcharts$new()
pi_growth_plot$series(name = "PIs", data = toJSONArray2(pi_cumul_growth[,c("date",
"x")],
                                        json = F, names = F))
```

This slide shows the code actually build the chart. Notice how every line, except the top one starts with the same variable name "PI_growth_plot". That's because after you run the first line "PI_growth_plot <- Highcharts$new()", which creates a new chart, you add more to it – data in the next line, title in the third line, etc. It may seem tedious at first, but that's how you ensure you get the fully customized look. However, if you have a format that you will be re-using often, you won't have to type it from scratch every time, you just tweak something or save it in a function for future use, which if you're interested learning, let us know and we'll add a note on the GitHub page of how to do that.
As you can see each line touches on a particular property of the chart – title, xAxis, yAxis, plotOptions, tooltip, etc. One of these is critical to the time series chart – specifically the line with xAxis –because there you specify that it's a datetime type axis, so it sorts the data appropriately. As you can see we can use a separate line to add title to the chart ($title), change default xAxis title ($xAxis), and yAxis. In yAxis we are demonstrating that you can change how labels look like – make them bigger or smaller, and a particular color. PlotOptions ($plotOptions) is used to specify that it's a line, with a certain color, and disabled markers). We can change how the tooltip looks like – here we only want to see the month and the year. There is also a neat feature in rCharts that allows you to zoom in on an year – basically draw a line with your mouse and the chart zooms in.
There is a trick to change completely how the tooltip looks like, using "formatter" property  (more on that can be found in our references links), and if you wish to get rid of the "Series 1" text in it – replace the single line with hPlot() function with two lines on the bottom of this slide.

## HANDS-ON RCHARTS CODING

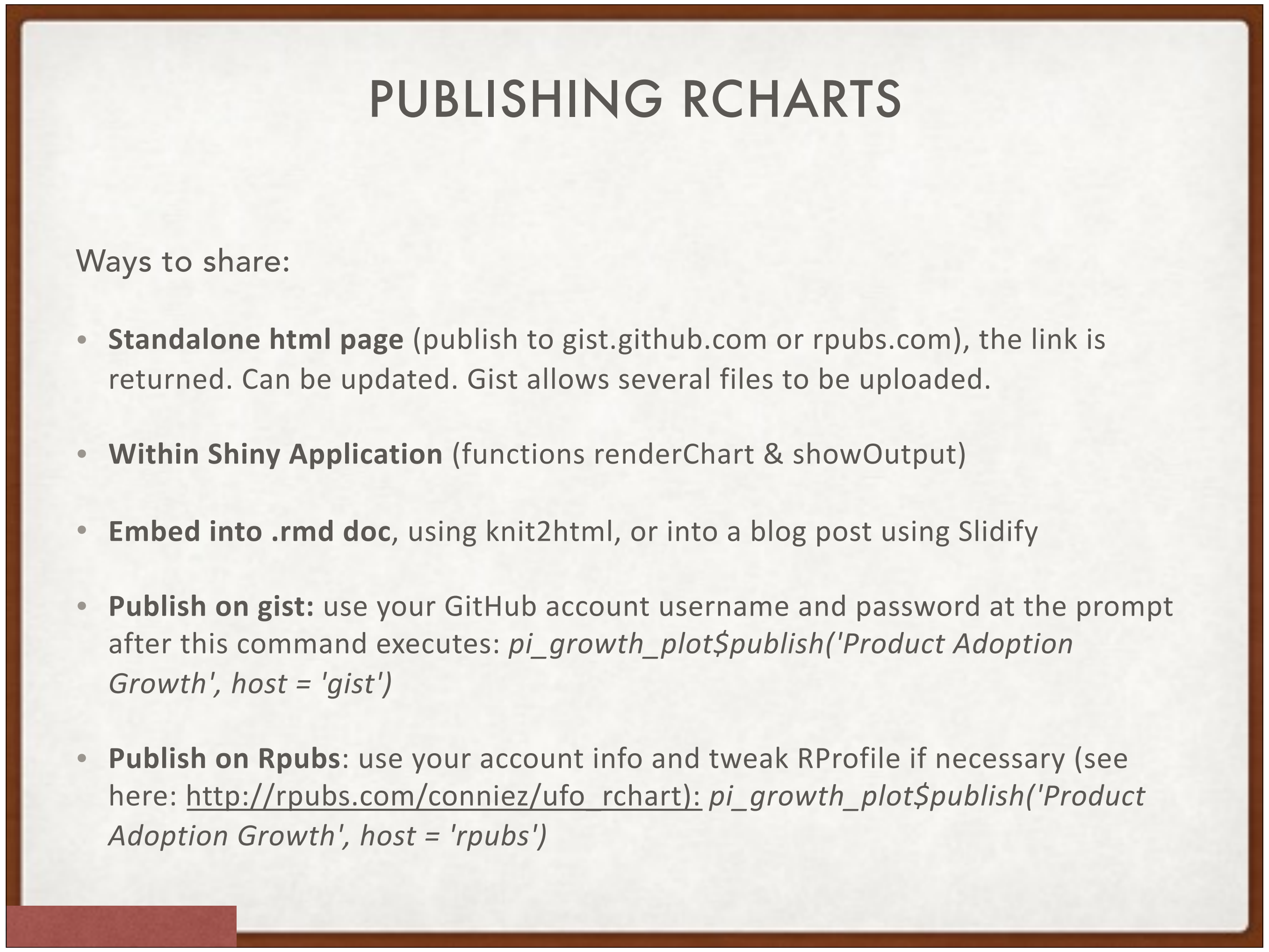
## BULLET GRAPHS WITH R

- Not in rCharts yet. Find the working source code and info here: https://github.com/sipemu/BulletGraph
- Developed by Simon Muller, who is also working on: https://github.com/sipemu/d3Dashboard

There’s two ways you can build them now – using ggplot2 or using an open-source file developed by the talented Simon Müller. You can see the links to his code on this slide. And this is an example of how you would link to his source file, prepare your data for graphing and call the drawing function.

## BULLET GRAPHS WITH R

```
source("helpers/BulletGraphSipemu.txt")
library(grid)

#create a dataframe with all the parameters - for two bullet graphs
bulletLoggedHours <- data.frame(measure = c("Logged This Month", "Logged Last Month"),
                   units = c("Hours", "Hours"),
                   low = c(100, 100),
                   mean = c(600, 1000),
                   high = c(1825,1825),
                   target = c(1522, 1522),
                   value = c(800,1600))
#run the horizontal bullet graph function
gridBulletGraphH(bulletLoggedHours, nticks=c(10, 10),
          format=c("s","s"), bcol=c("#6699CC", "#85ADD6", "#C2D6EB"), font=11,
          scfont=9, ptitle="Logged Hours (Monthly Target = 1522)")
```



## PUBLISHING RCHARTS

Ways to share:

- **Standalone html page** (publish to gist.github.com or rpubs.com), the link is returned. Can be updated. Gist allows several files to be uploaded.
- **Within Shiny Application** (functions renderChart & showOutput)
- **Embed into .rmd doc**, using knit2html, or into a blog post using Slidify
- **Publish on gist:** use your GitHub account username and password at the prompt after this command executes: *pi_growth_plot$publish('Product Adoption Growth', host = 'gist')*
- **Publish on Rpubs**: use your account info and tweak RProfile if necessary (see here: http://rpubs.com/conniez/ufo_rchart): *pi_growth_plot$publish('Product Adoption Growth', host = 'rpubs')*

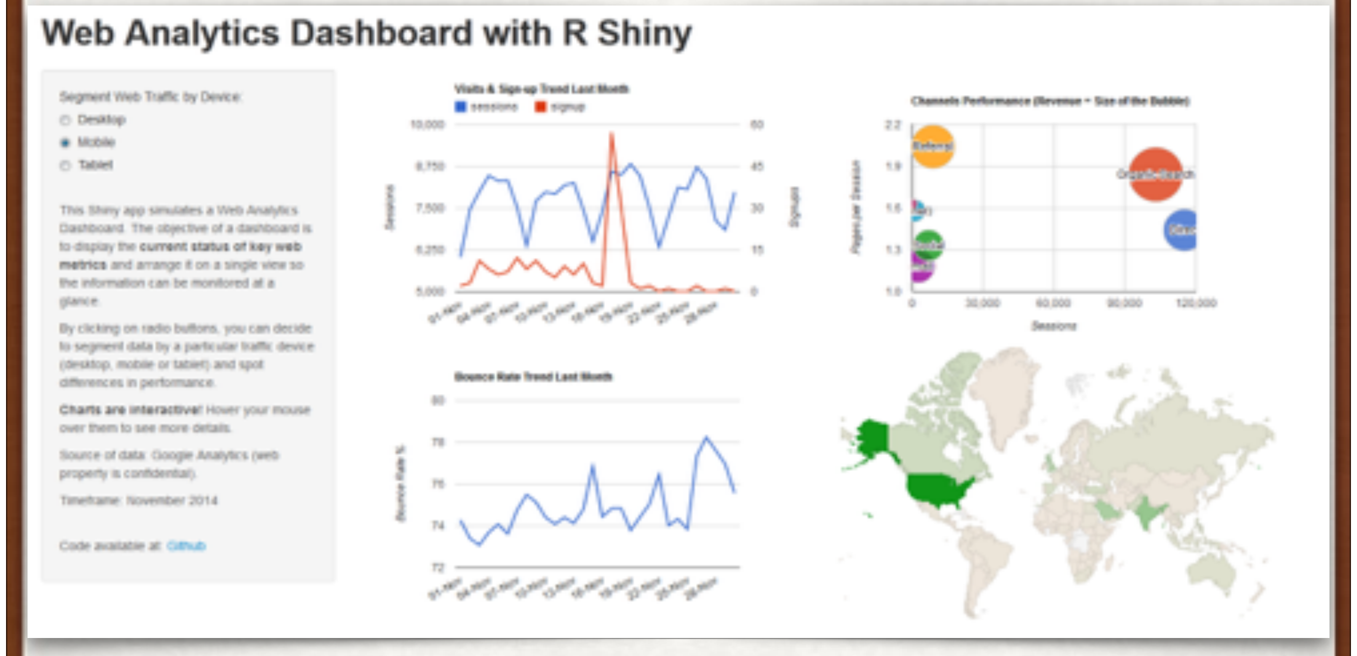
Publishing rCharts is possible in many ways, but that's something we are not going to cover in this short tutorial. Please feel free to peruse this slide afterwards and make use of the links in References.

## DASHBOARDS IN R

## WHY BUILD A DASHBOARD IN R?

- Raw data is often messy (include cleaning and prepping processes in R)
- R was designed around statistics (prediction models, complex analyses)
- R allows connecting to most databases (except perhaps the newest)
- invalidateLater(), reactivePoll() functions in Shiny app
- No limit to graphical tools (any chart is possible)
- Control layout with shinydashboard or ShinyGridster
- Style sheets supported (unlimited style and UI customization)
- Absolute freedom of design, analyses and no restraints over layout and form
- Host locally, on Shiny Server (Pro) or shinyapps.io

## EXAMPLES OF DASHBOARDS AND SIMILAR APPS BUILT IN R


Visits & Sign-up Trend Last Month
sessions
signup
Sessions
Signups
Channels Performance (Revenue ~ Size of the Bubble)
Pages per Session
Sessions
Bounce Rate Trend Last Month
Bounce Rate %

Web Analytics Dashboard with R Shiny

Segment Web Traffic by Device:

Desktop [✓ Mobile] Tablet

This Shiny app simulates a Web Analytics Dashboard. The objective of a dashboard is to display the **current status of key web metrics** and arrange it on a single view so the information can be monitored at a glance.

By clicking on radio buttons, you can decide to segment data by a particular traffic device (desktop, mobile or tablet) and spot differences in performance.

**Charts are interactive!** Hover your mouse over them to see more details.

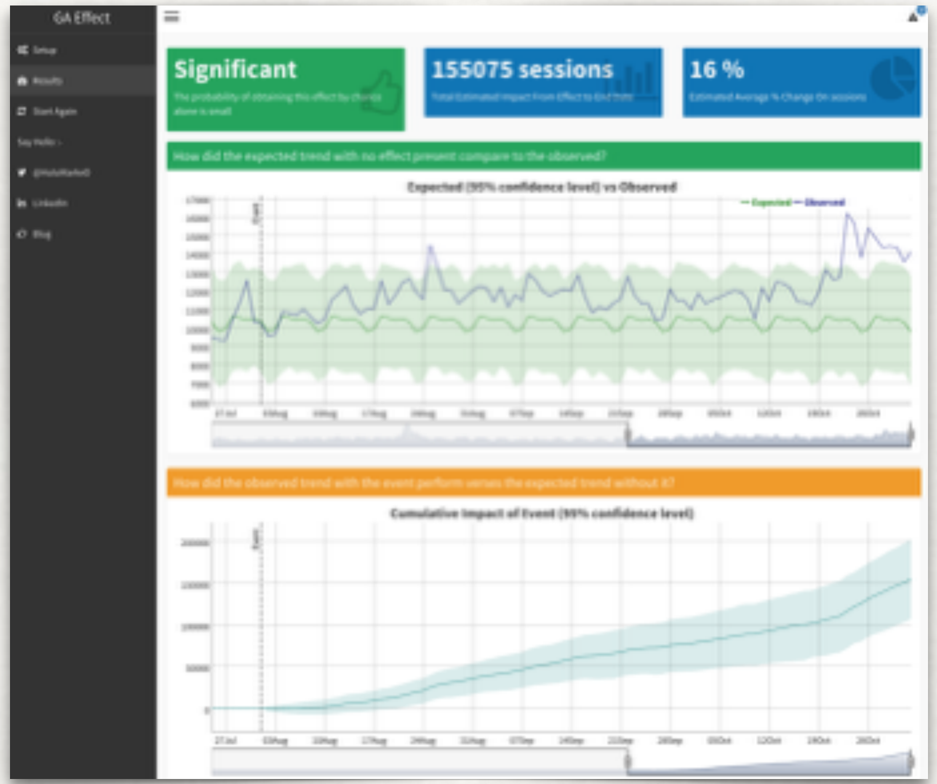
Source of data: Google Analytics (web property is confidential).

Timeframe: November 2014

Code available at: Github

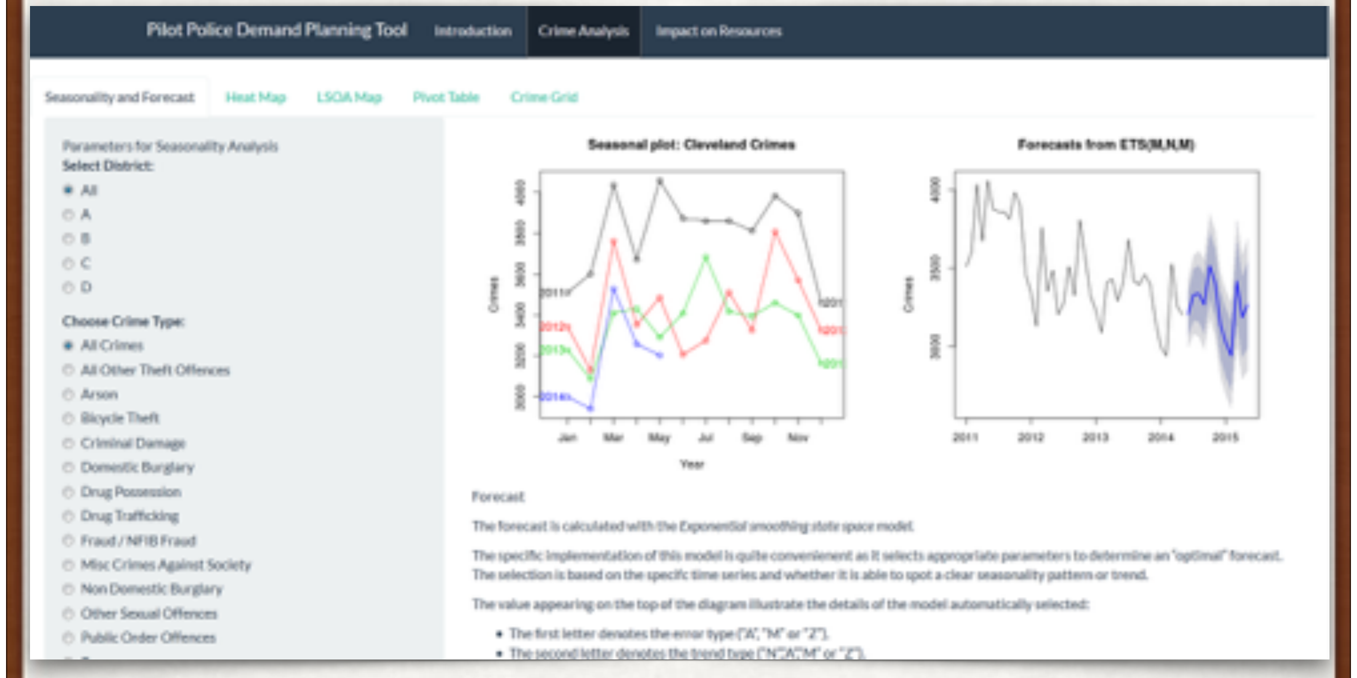
Source: https://mcpasin.shinyapps.io/WebAnalytics-Dashboard

## EXAMPLES OF DASHBOARDS AND SIMILAR APPS BUILT IN R


GA Effect
Significant
155075 sessions
16 %
How did the expected trend with no effect present compare to the observed?
Expected (95% confidence level) vs Observed
How did the observed trend with the event perform versus the expected trend without it?
Cumulative Impact of Event (95% confidence level)

Source: http://markedmondson.me/how-i-made-ga-effect-creating-an-online-statistics-dashboard-using-reais

## EXAMPLES OF DASHBOARDS AND SIMILAR APPS BUILT IN R



Parameters for Seasonality Analysis

**Select District:**

- [✓ All]
- A
- B
- C
- D

**Choose Crime Type:**

- [✓ All Crimes]
- All Other Theft Offences
- Arson
- Bicycle Theft
- Criminal Damage
- Domestic Burglary
- Drug Possession
- Drug Trafficking
- Fraud / NFIB Fraud
- Misc Crimes Against Society
- Non Domestic Burglary
- Other Sexual Offences
- Public Order Offences

Forecast

The forecast is calculated with the *Exponential smoothing state space model*.

The specific implementation of this model is quite convenient as it selects appropriate parameters to determine an "optimal" forecast. The selection is based on the specific time series and whether it is able to spot a clear seasonality pattern or trend.

The value appearing on the top of the diagram illustrate the details of the model automatically selected:

- The first letter denotes the error type ("A", "M" or "Z").
- The second letter denotes the trend type ("N","A","M" or "Z").

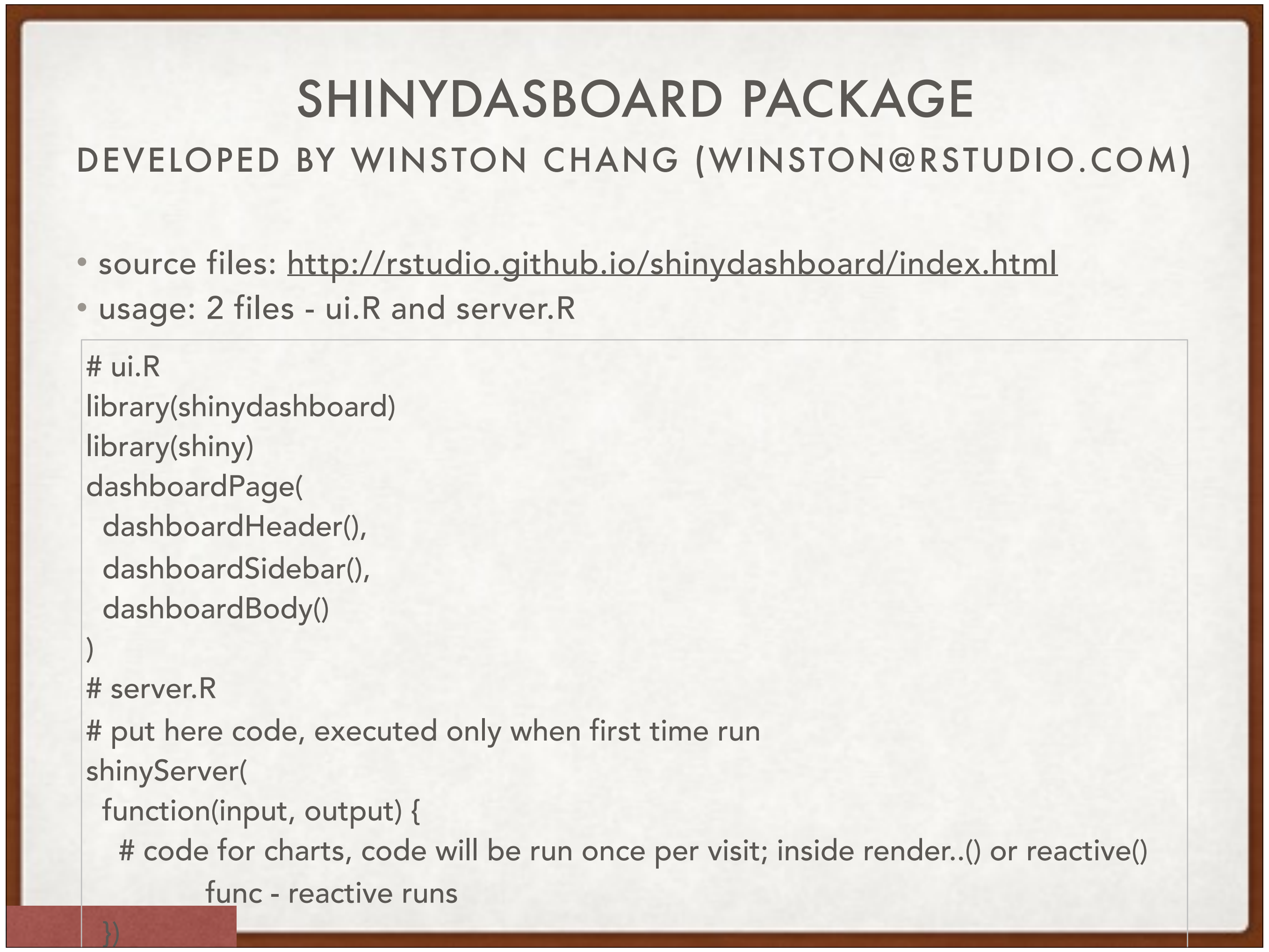
Source: https://smartinsightsfromdata.shinyapps.io/TSupplyDemand/

## BASIC TEMPLATE OF A SHINY APP WITH RCHARTS

```
# ui.R
library(rCharts)
shinyUI(fluidPage(
  h2("rCharts Example"),
  sliderInput("slider", "Number of observations:", 1, 1200, 500),
  showOutput("myChart", "highcharts")))
```

```
# server.R
library(rCharts)
library(plyr)
merged <- read.csv("data/opsdata.csv")
shinyServer(function(input, output) {
  output$myChart <- renderChart({
    pi_by_product <- ddply(head(merged, input$slider), c("application"), summarise,
                           uniquePIs = length(unique(pi_name)))
    mostPopularProduct <- hPlot(uniquePIs ~ application, type = "column",
                             data = pi_by_product, title = "Most Popular Product")
    mostPopularProduct$addParams(dom = "myChart")
    return(mostPopularProduct)
  })})
```

Before we look at dashboards built with R, we need to first get familiar with Shiny. Shiny is a web application framework for R, which takes R code and renders it as HTML, and JavaScript enabling us to build web applications in R, without knowing much HTML or JavaScript.
Here is a very basic Shiny application, that demos rCharts and widgets that allow you to change data input. Normally you would build a Shiny application with two files – one for User Interface, called ui.R and the other for data manipulations and building charts – called server.R.
There's a few things worth pointing out:
if you are working with non-base packages in R, such as rCharts for instance, you have to load them at the beginning of the app files (using library() function) at the beginning
in ui.R you need a comma after each function call inside shinyUI() function
sliderInput() will display a slider with the specified parameters which will cause manipulate the data used by the plot in server.R – here the number of observations used for the plot will be varying between 1 and 1200; in fact you can specify it to run from 1 to 1200, which will cause the plot to show progression of changes. Notice that we reference that widget inside our code of server.R using prefix input$, which is what ui.R passes into server.R.
in server.R you load the data before the shinyServer() function, if you wish for that to only execute once – at the first run of the application
whatever is inside shinyServer() function will run every time the application is accessed. And if you wish to connect to a database you can set it up that the data will be refreshed after a certain amount of seconds using special functions.
to build an rChart in a Shiny app you only need to wrap it in a renderChart() function and then add a few lines at the end to connect it to the domain name "myChart", which will be referenced in the ui.R file inside showOutput() function. That's how both of them are connected.



## SHINYDASBOARD PACKAGE

DEVELOPED BY WINSTON CHANG (WINSTON@RSTUDIO.COM)

- source files: http://rstudio.github.io/shinydashboard/index.html
- usage: 2 files - ui.R and server.R

```
# ui.R
library(shinydashboard)
library(shiny)
dashboardPage(
  dashboardHeader(),
  dashboardSidebar(),
  dashboardBody()
)
# server.R
# put here code, executed only when first time run
shinyServer(
  function(input, output) {
    # code for charts, code will be run once per visit; inside render..() or reactive()
              func - reactive runs
  })
```

Building a dashboard in R has become much easier with the very recent development of shinydashboard package by RStudio. You still need the two files shown above, but the contents of them will be slightly different. You no longer see shinUI() function in ui.R, it's been replaced by dashboardPage() function, where you specify the parameters for the dashboard., but the content of server.R can stay the same.

```
library(shinydashboard)
library(shiny)
dashboardPage(
  dashboardHeader(
    # title of dashboard
    ),
    dashboardSidebar(
      # code for sidebar tabs
      ),
      dashboardBody(
        # tabs corresponding to sidebar code, boxes with charts
        )
      )
```

## PUBLISHING SHINY APPS/DASHBOARDS

- Publish on shinyapps.io – free for limited use
- Publish on Shiny Server (public is free, Pro is paid)
- Some commands for working with Shiny and shinyapps.io

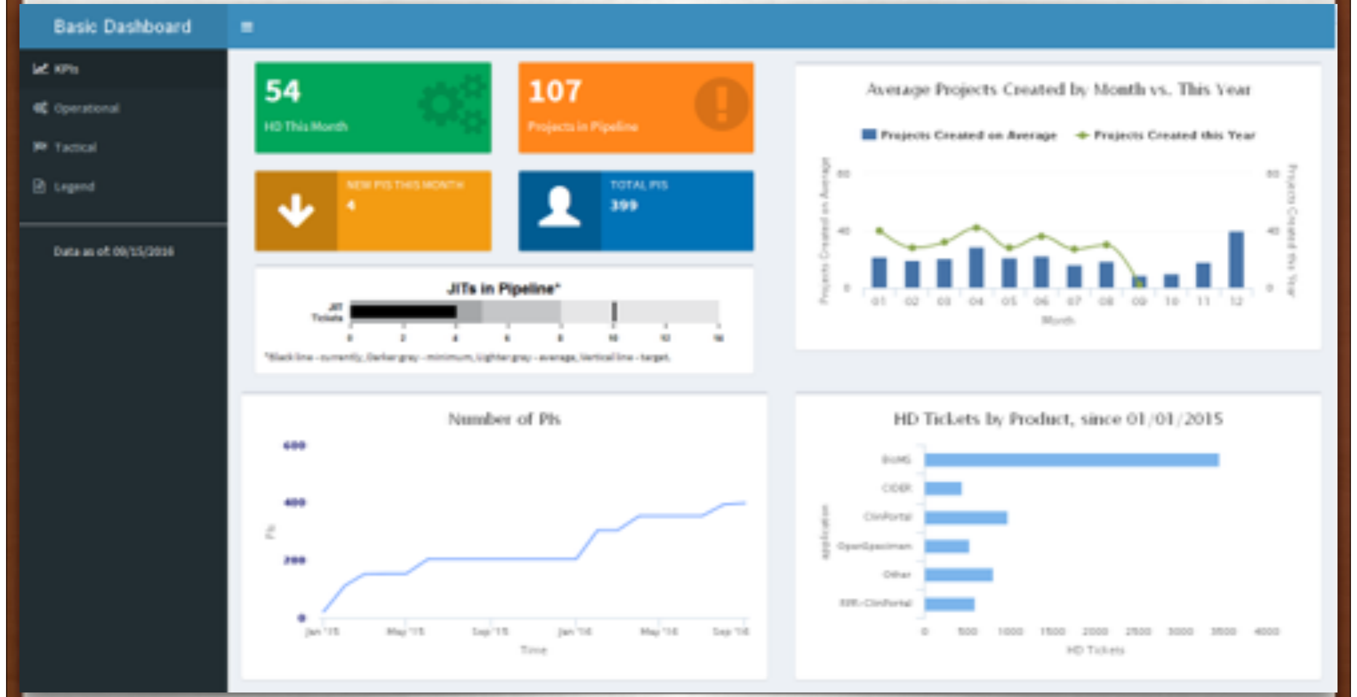
```
runApp("myapp")
library(shinyapps)
deployApp("myapp")
```

You can publish your dashboard on shinyapps.io for free, but there's a limit of 5 applications and 25 active hours per month. With a Standard subscription you are able to add authentication of users with passwords and a larger number of applications and active hours per month.

If cloud service is not for you, you can opt in to host a Shiny Server open-source edition or Shiny Server Pro. Shiny Server Pro allows for authentication, SSL options, and has no cap on how many applications or active hours you can get, however it limits the number of concurrent users to 20, while you can extend that number for an additional fee.

CBMI has elected to purchase the Shiny Server Pro license…

# DEMO DASHBOARD

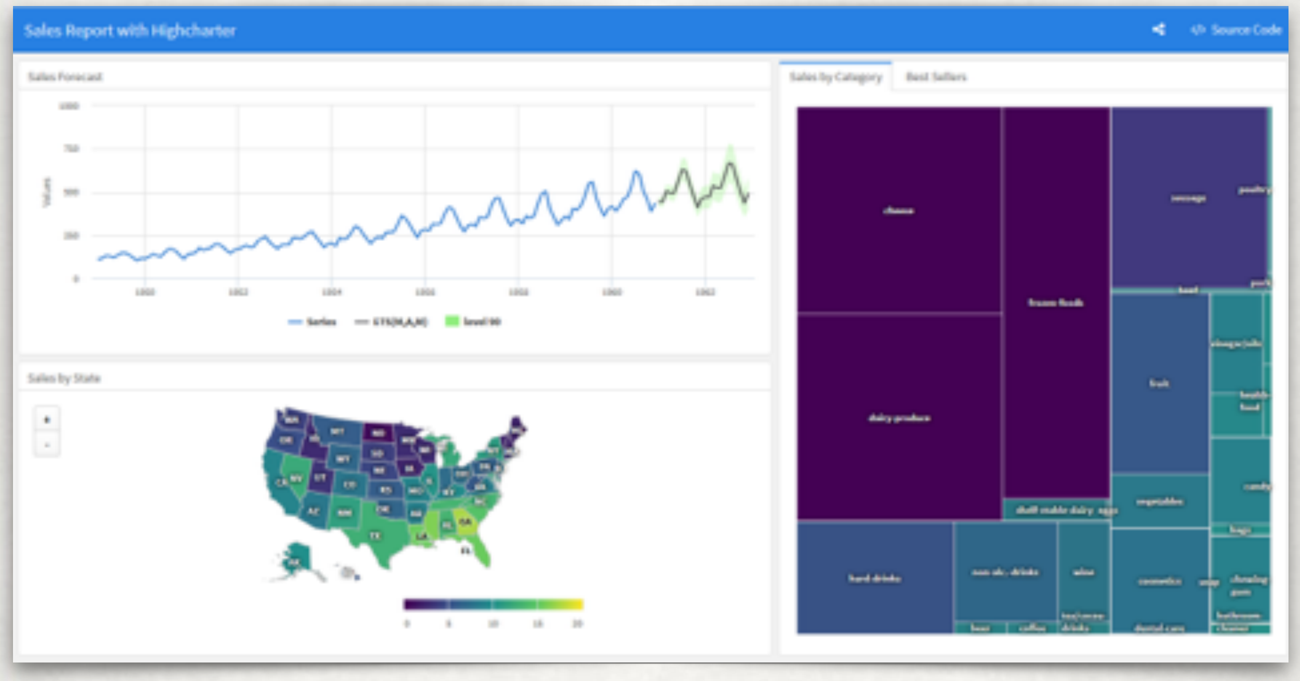


## HANDS-ON SHINY APPS: SHINYDASHBOARD PACKAGE

## VERSATILE NATURE OF SHINY APPS

- Automate reporting
- Discover patterns and information in data
- Interactively train people in skills: biostatistics, data analysis
- Support user authentication and different views of application based on user

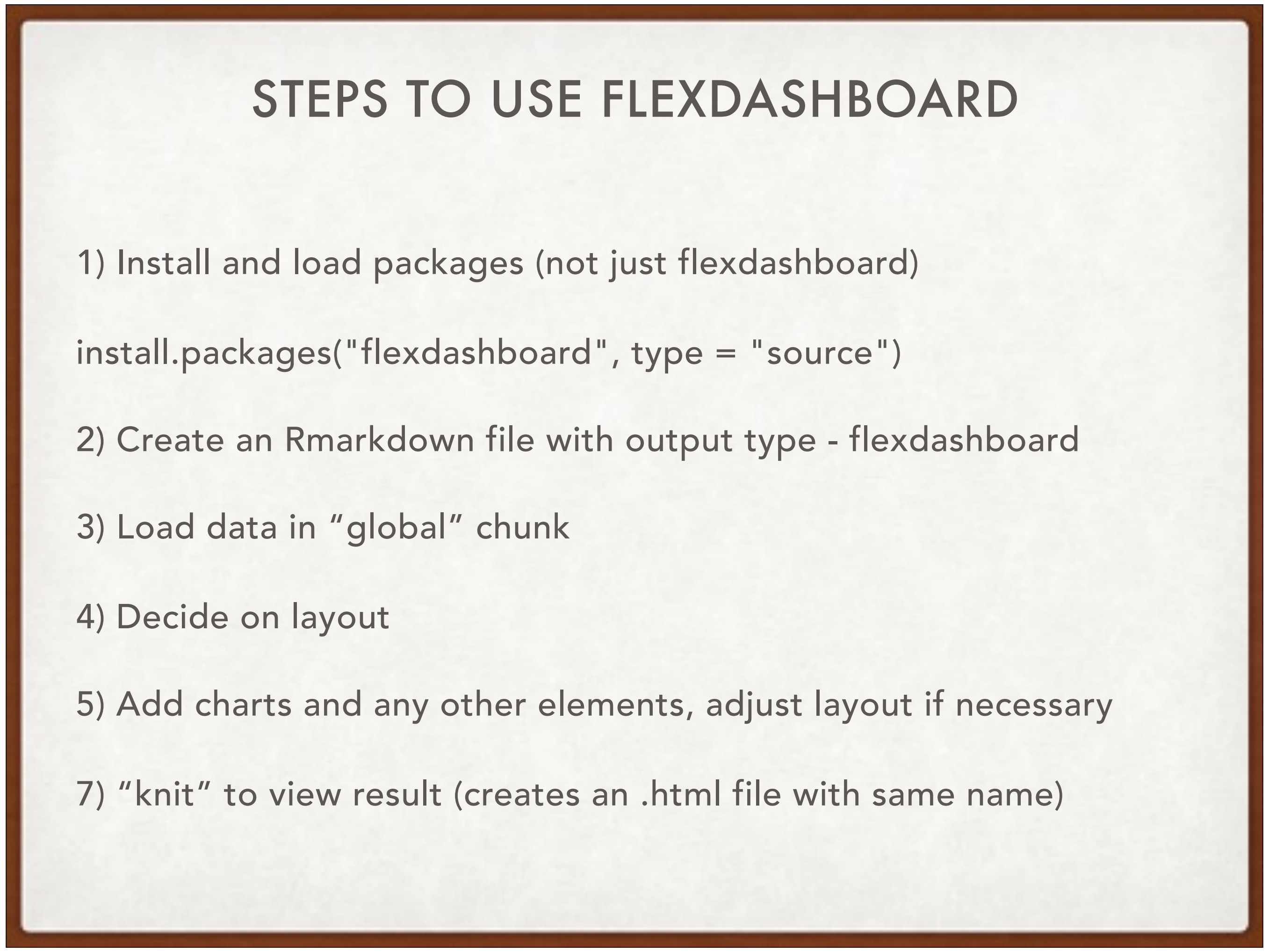
# FLEXDASHBOARD BRIEF OVERVIEW



https://beta.rstudioconnect.com/jjallaire/htmlwidgets-highcharter/

## WHAT IS FLEXDASHBOARD?

- R package to combine graphics into a dashboard-like view
- No strict format to follow (vs. shinydashboard)
- Write 1 file of Rmarkdown code to create a layout and add charts, widgets and other elements as necessary
- Dashboard is adapted for display on mobile devices
- Some pieces still in development
- Detailed explanation and tutorial: http://rmarkdown.rstudio.com/flexdashboard



## STEPS TO USE FLEXDASHBOARD

1) Install and load packages (not just flexdashboard)

```
install.packages("flexdashboard", type = "source")
```

2) Create an Rmarkdown file with output type - flexdashboard

3) Load data in "global" chunk

4) Decide on layout

5) Add charts and any other elements, adjust layout if necessary

7) "knit" to view result (creates an .html file with same name)

- have knitr and markdown packages installed, as well as any other graphical packages you plan to use for data visualization
- install and then load flexdashboard

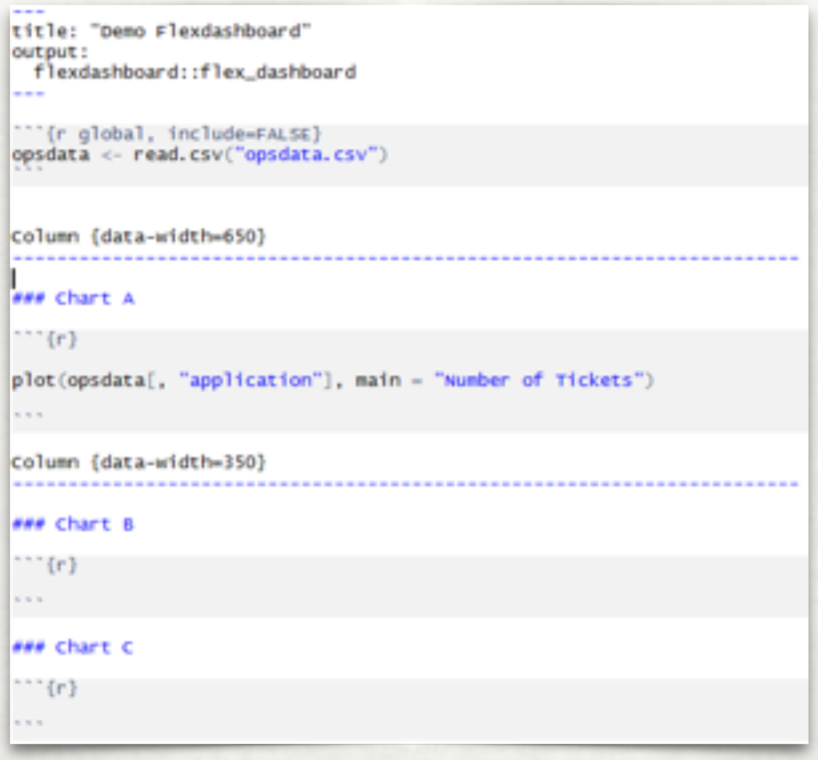
```
install.packages("flexdashboard", type = "source")
```

- create an Rmarkdown file with output type - flexdashboard, using RStudio: New file/R Markdown/From Template/Flex Dashboard, or by running:

```
rmarkdown::draft("dashboard2.Rmd", template = "flex_dashboard", package = "flexdashboard")
```

(ignore the error message in console)
- load data in "global" chunk so it's accessible to all charts
- decide on layout
- add charts
- add any other elements, adjust layout if necessary
- "knit" to view result and create an .html file with same name

## SAMPLE OF FLEXDASHBOARD CODE



~~~
---
title: "Demo Flexdashboard"
output:
  flexdashboard::flex_dashboard
---

```{r global, include=FALSE}
opsdata <- read.csv("opsdata.csv")
~~~

Column {data-width=650}
-----------------------------------------------------------------------

### Chart A

```{r}

plot(opsdata[, "application"], main = "Number of Tickets")

```

Column {data-width=350}
-----------------------------------------------------------------------

### Chart B

```{r}

```

### Chart C

```{r}

```
```


```
---
title: "Demo Flexdashboard"
output:
  flexdashboard::flex_dashboard
---

```{r global, include=FALSE}
opsdata <- read.csv("opsdata.csv")
```

Column {data-width=650}
-----------------------------------------------------------------------

### Chart A

```{r}

plot(opsdata[, "application"], main = "Number of Tickets")

```

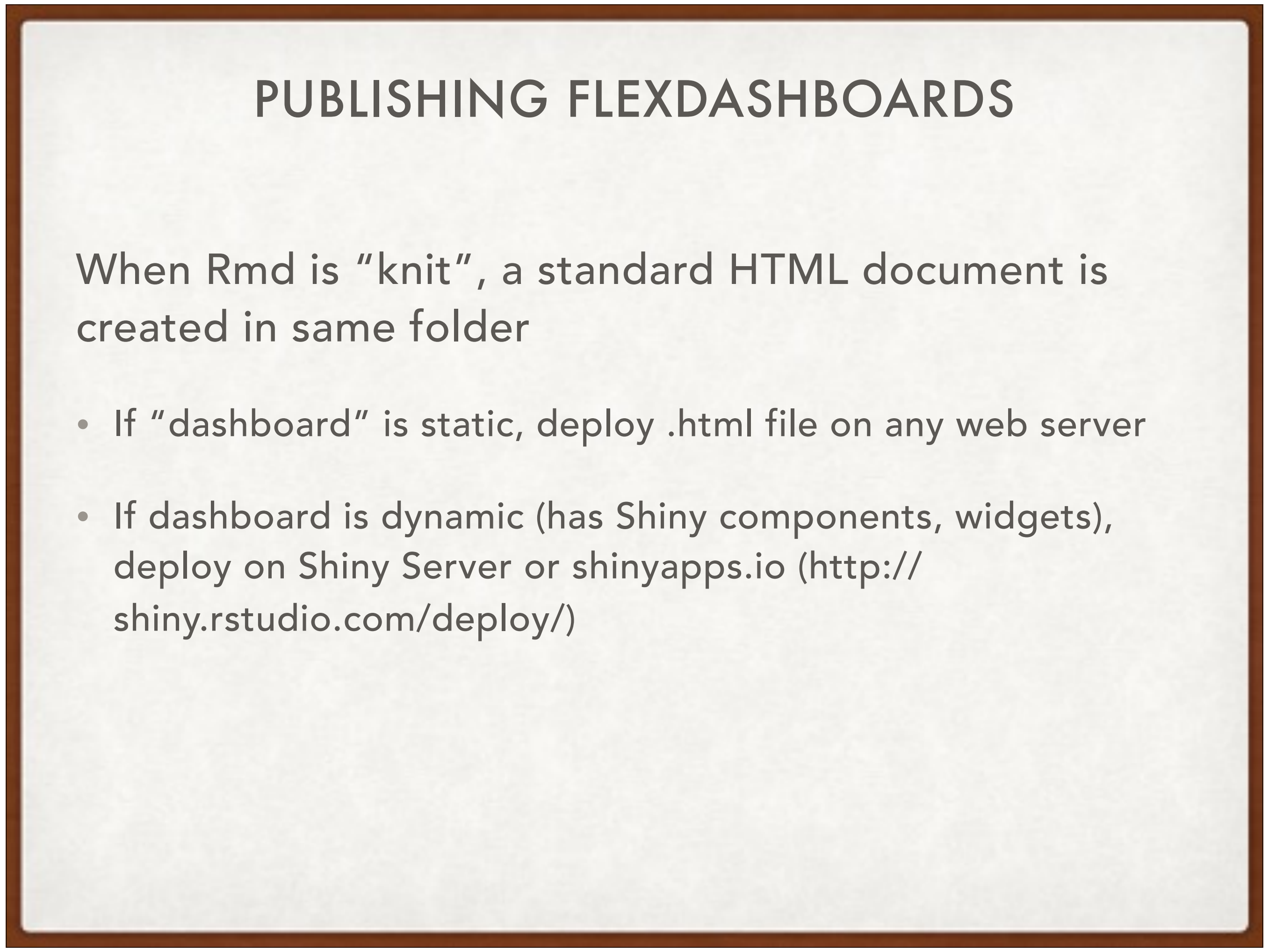
Column {data-width=350}
-----------------------------------------------------------------------

### Chart B

```{r}

```

### Chart C

```{r}

```
```
```



## PUBLISHING FLEXDASHBOARDS

When Rmd is “knit”, a standard HTML document is created in same folder

- If “dashboard” is static, deploy .html file on any web server
- If dashboard is dynamic (has Shiny components, widgets), deploy on Shiny Server or shinyapps.io (http://shiny.rstudio.com/deploy/)

By default dashboards are standard HTML documents that can be deployed on any web server or even attached to an email message. You can optionally add Shiny components for additional interactivity and then deploy on Shiny Server or shinyapps.io.

## CONNECTING TO DATABASES WITH R

There are several ways to connect to databases from R using packages:

- RJDBC
- RODBC
- Using specialized packages for specific databases, e.g. RMySQL, ROracle, RSQLite, etc.
- Using dplyr (works with MySQL, PostreSQL, SQLite)

## WHAT METHOD TO CHOOSE?

| RJDBC | RODBC | Specialized Package | dplyr |
|---|---|---|---|
| Very easy setup: Java + database JDBC driver | Quite complicated setup: <br>- Need ODBC Driver Manager (Windows ODBC Driver Manager, or iODBC, or unixODBC), and an ODBC driver. <br>- Some ODBC drivers are not free | Setup difficulty depends on the OS: <br>- Need database client installed | Setup difficulty is similar to Specialized package setup: <br>- Need database client installed <br>- Need specialized package for the database in question (e.g. RMySQL) |
| Seems to be somewhat slow | Quick and mature | Quick, since uses the native database client | dplyr in general is very quick, esp. because it's |
| Versatile, one package to work with many database types | Versatile, one package to work with many database types | One package works only with specific database: installation will need to happen for all necessary | Somewhat versatile, works with MySQL, PostreSQL, SQLite |

## CITATIONS & RESOURCES

## RESOURCES USED (ON KPIS, RCHARTS AND SHINY)

- Stacey Bar, Performance Measurement Process: http://staceybarr.com/
- Stephen Few. Information Dashboard Design: Displaying Data for At-a-Glance Monitoring: http://www.amazon.com/Information-Dashboard-Design-At-Glance/dp/1938377001/ref=pd_sim_b_3?ie=UTF8&refRID=1CZ30V5X6TCQGHEQ719J
- Dona Wong, Guide to Information Graphics: http://donawong.com/
- Charles Minard's map graphic: http://en.wikipedia.org/wiki/Charles_Joseph_Minard#/media/File:Minard.png
- Bad Chart Example (Annual Spending): http://tdworld.com/polls/featured-poll-how-will-election-results-impact-grid
- Bar Charts vs. Pie Charts Exercise: http://www.danielpradilla.info/blog/en/how-to-choose-the-right-chart/

## RESOURCES USED (ON KPIS, RCHARTS AND SHINY) (CONT'D)

- Choice of Charts: Suitable vs. Unsuitable According to Stephen Few, adapted from: http://www.stoyko.net/smithysmithy/archives/988
- Domo Charts Poor Gauge Example: http://www.domo.com/roles/operations
- Bullet Graph Example: http://www.healthdataviz.com/2012/04/13/a-magic-bullet-graph-for-weak-data-visuals/
- iDashboards demo (Poor Dashboards Example):http://www.idashboards.com/live-dashboard-examples/pharmaceutical-biotech-scorecard-dashboard/
- Another poor dashboard example: http://www.bittle-solutions.com/dashboard-e-marketing/
- Another poor dashboard example: http://www.bittle-solutions.com/dashboard-green-vehicles/
- Example of Stephen Few's Dashboard: http://www.startuplifeblog.com/tag/stephen-few/
- Getting Started with rCharts: http://ramnathv.github.io/rCharts/

## RESOURCES USED (ON KPIS, RCHARTS AND SHINY) (CONT'D)

- Grammar of Graphics Chart: "The Grammar of Graphics (Statistics and Computing)" by Leland Wilkinson
- Hadley Wickham (2010) A Layered Grammar of Graphics, Journal of Computational and Graphical Statistics, 19:1, 3-28, DOI: 10.1198/jcgs.2009.07098. http://dx.doi.org/10.1198/jcgs.2009.07098
- Parmenter, David. Key Performance Indicators (KPI) : Developing, Implementing, and Using Winning KPIs (2). Hoboken, US: Wiley, 2010. ProQuest ebrary. Web. 17 August 2016. Copyright © 2010. Wiley. All rights reserved.
- Eckerson, Wayne W.. Performance Dashboards : Measuring, Monitoring, and Managing Your Business (2). Hoboken, US: Wiley, 2010. ProQuest ebrary. Web. 17 August 2016. Copyright © 2010. Wiley. All rights reserved.
- Examples by creator: http://ramnathv.github.io/rChartsShiny/
- Example of shiny app with downloading data from internet: https://github.com/ramnathv/rChartsShiny/blob/gh-pages/rChartOECD/global.R
- Great examples of all types of charts in NVD3: http://ramnathv.github.io/posts/rcharts-nvd3/index.html

## RESOURCES USED (ON KPIS, RCHARTS AND SHINY) (CONT'D)

- Great examples with Highcharts: http://rpubs.com/kohske/12409 (and this one http://rstudio-pubs-static.s3.amazonaws.com/16699_4bc388ebe1454c84aaab3d22d17e3aaf.html)
- What chart to use when: http://timelyportfolio.github.io/rCharts_nvd3_systematic/cluster_weights.html
- Examples from Ramnath NVD3: https://github.com/ramnathv/rCharts/blob/master/inst/libraries/nvd3/examples.R
- How to embed into Rmarkdown: http://bl.ocks.org/ramnathv/raw/8084330/ (and this http://timelyportfolio.github.io/rCharts_share/showingoff.html)
- A very detailed explanation on how to use Highcharts API for rCharts: http://reinholdsson.github.io/rcharts-highcharts-api-docs/
- Flexdashboard Docs: http://rmarkdown.rstudio.com/flexdashboard/

## DATABASE CONNECTION RESOURCES

- http://simplyanalyticsblog.com/2014/02/20/rodbc-package-on-linux/
- http://www.unomaha.edu/mahbubulmajumder/data-science/fall-2014/lectures/20-database-mysql/20-database-mysql.html#/
- https://cran.r-project.org/web/packages/dplyr/vignettes/databases.html
- http://faculty.washington.edu/kenrice/sisg-adv/sisg14-adv-09.pdf

# THANKS

HTTPS://GITHUB.COM/CBMIWU/DI4R_TUTORIALVISDATA