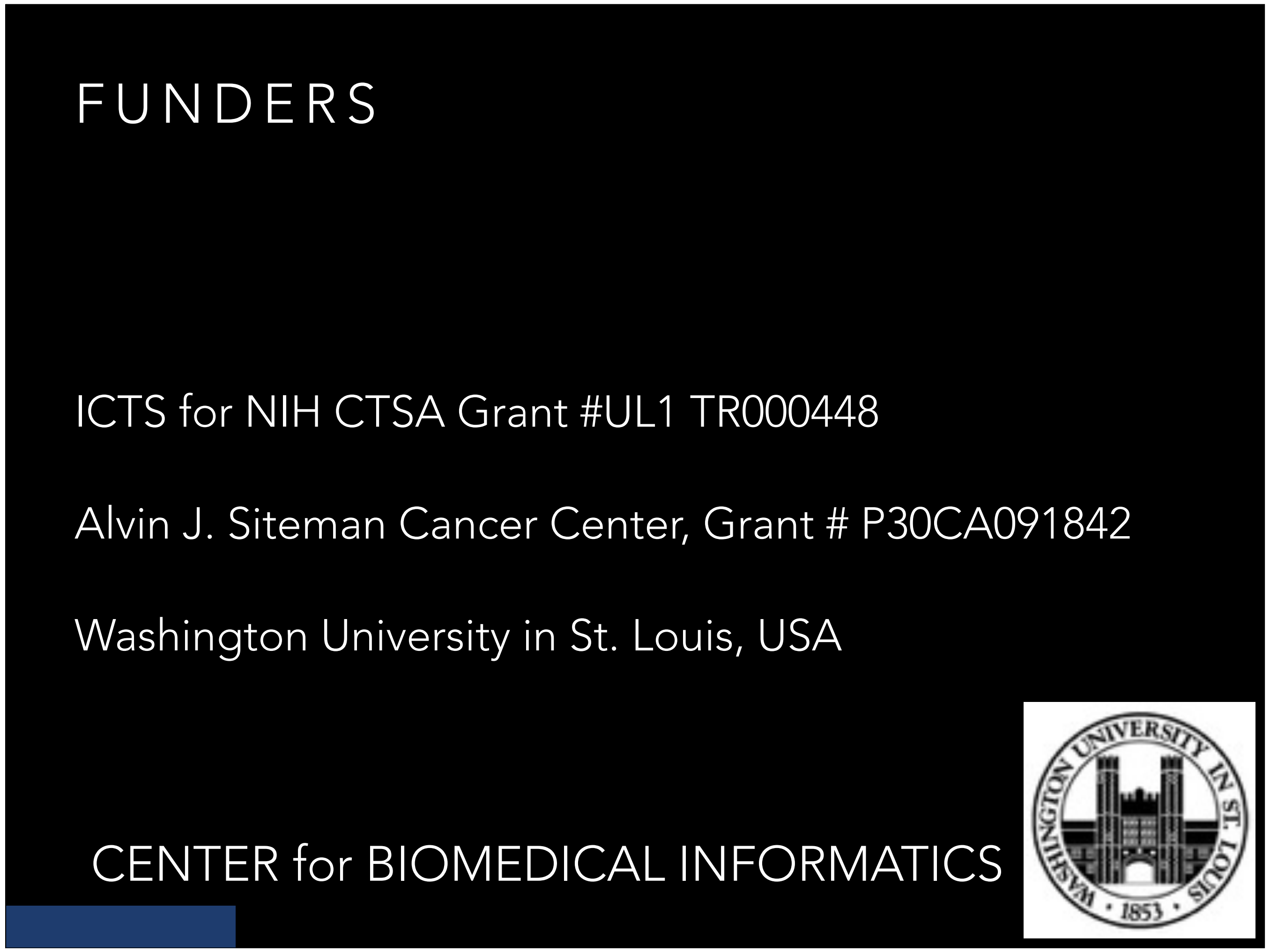
LESLIE MCINTOSH, PHD, MPH; CONNIE ZABAROVSKAYA, MBA

Visualizing Operational Informatics Data Using R
MEDINFO IMIA 2015 Tutorial



FUNDERS

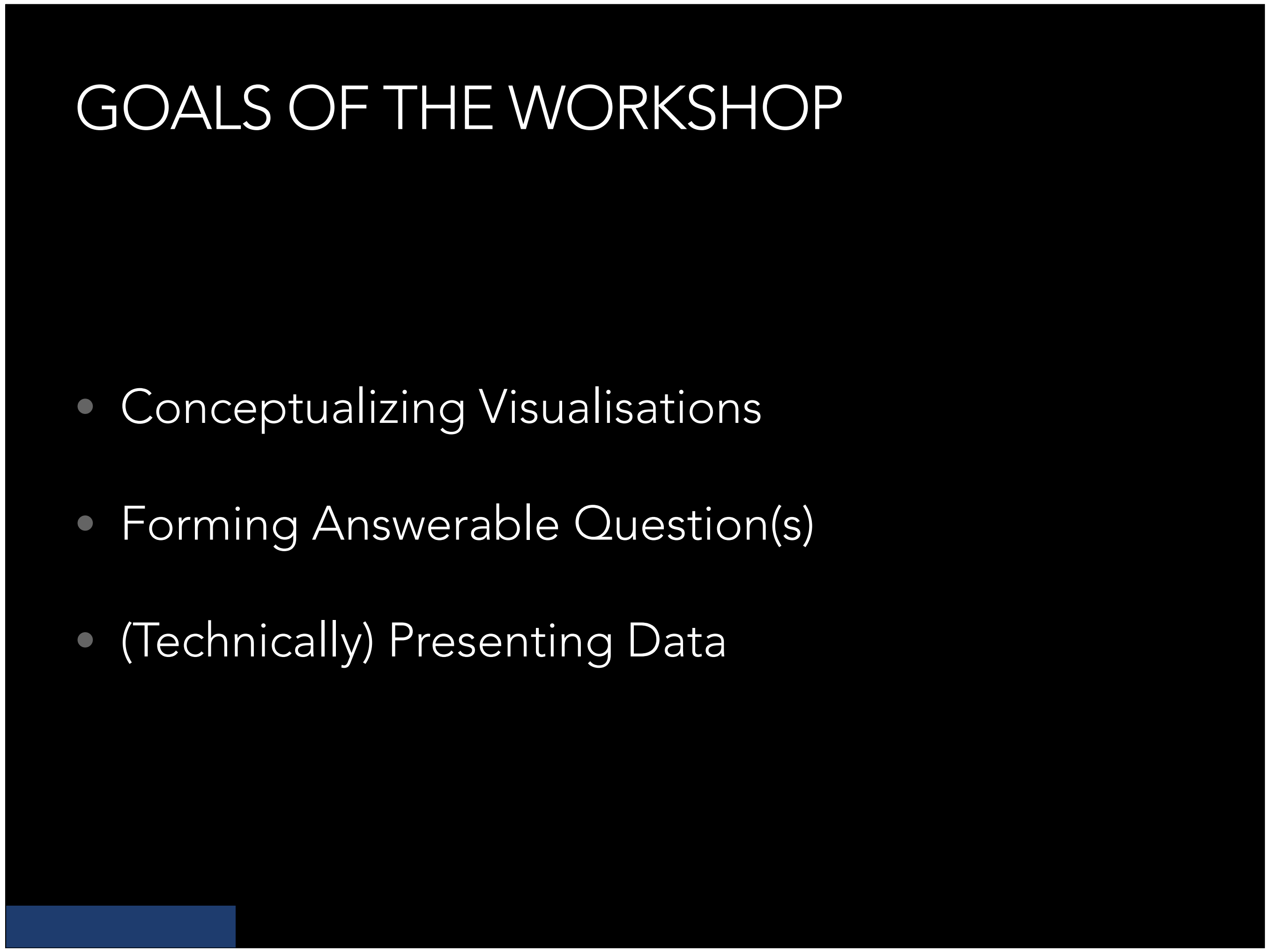
ICTS for NIH CTSA Grant #UL1 TR000448

Alvin J. Siteman Cancer Center, Grant # P30CA091842

Washington University in St. Louis, USA

CENTER for BIOMEDICAL INFORMATICS

We would like to thank our funding.



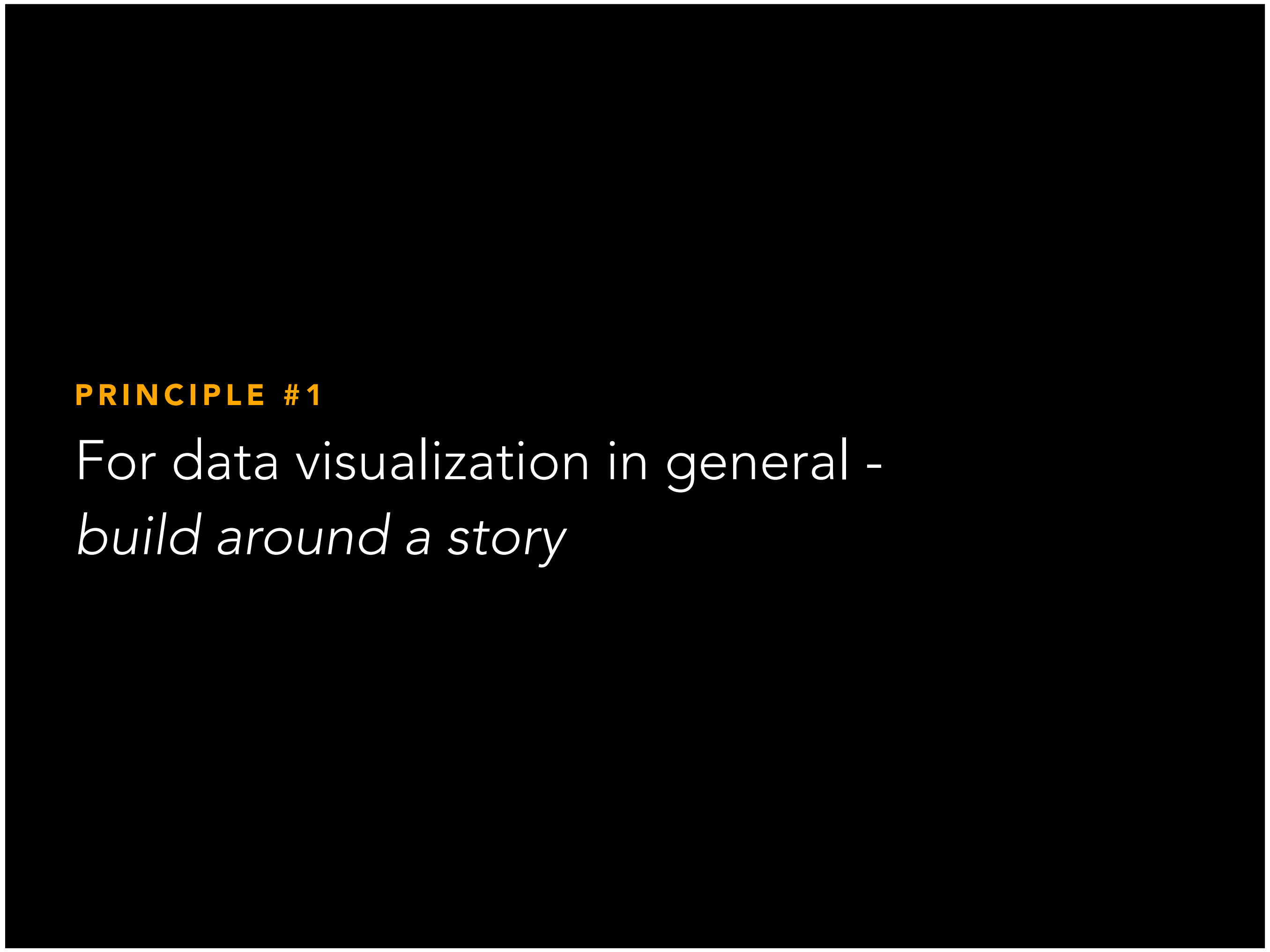
GOALS OF THE WORKSHOP

- Conceptualizing Visualisations
- Forming Answerable Question(s)
- (Technically) Presenting Data

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

PRINCIPLES OF EFFICIENT DATA VISUALIZATION

PART (1)

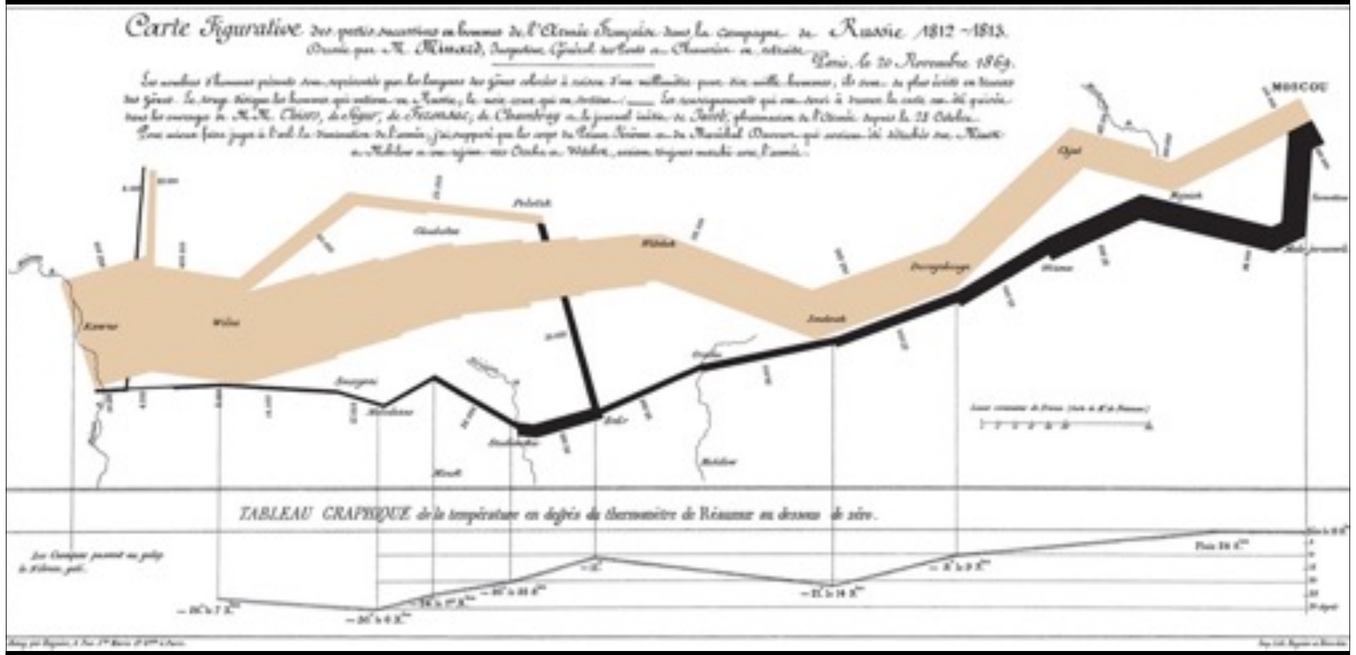


PRINCIPLE #1

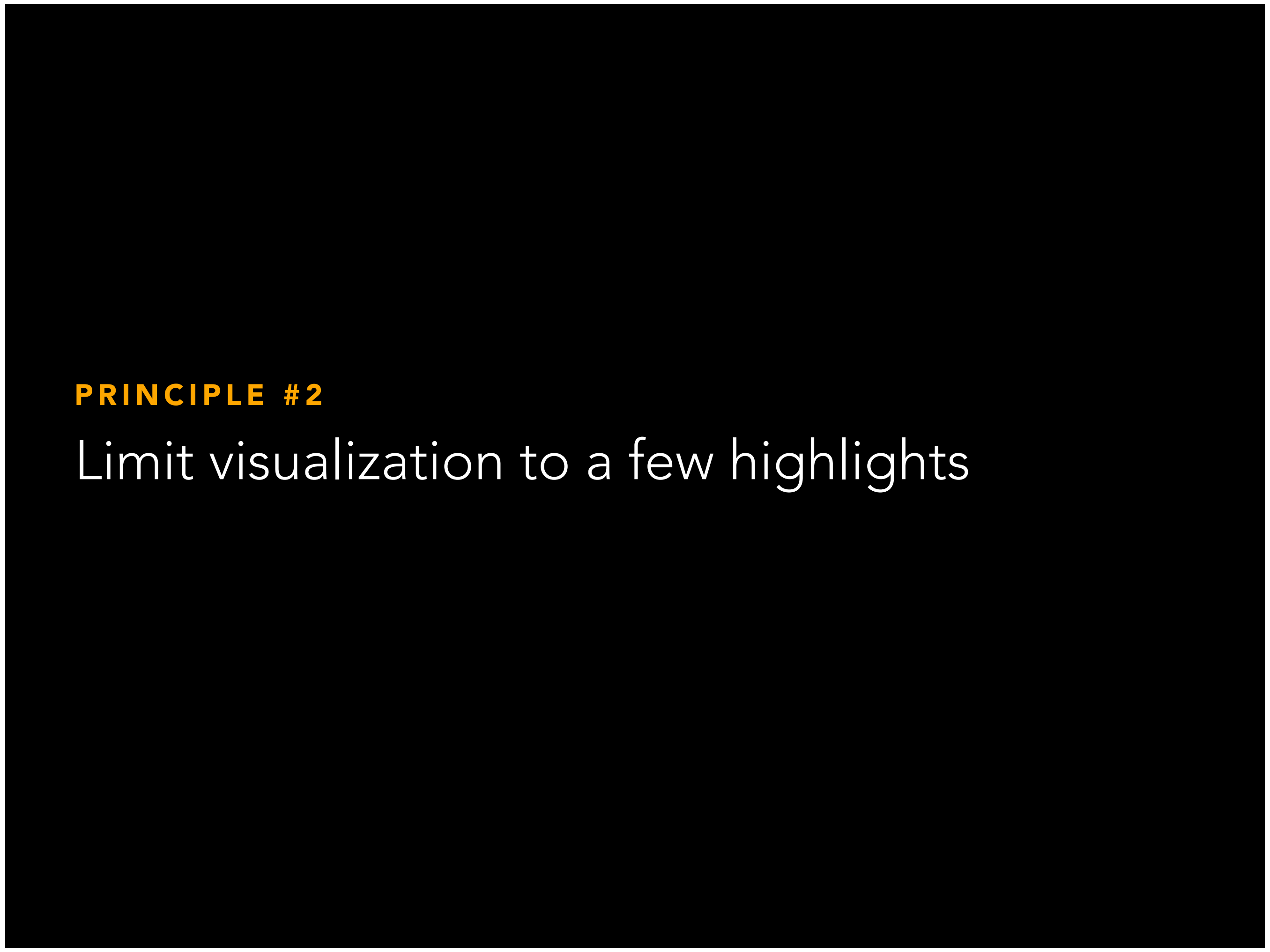
For data visualization in general -
build around a story

From Donna Wong

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



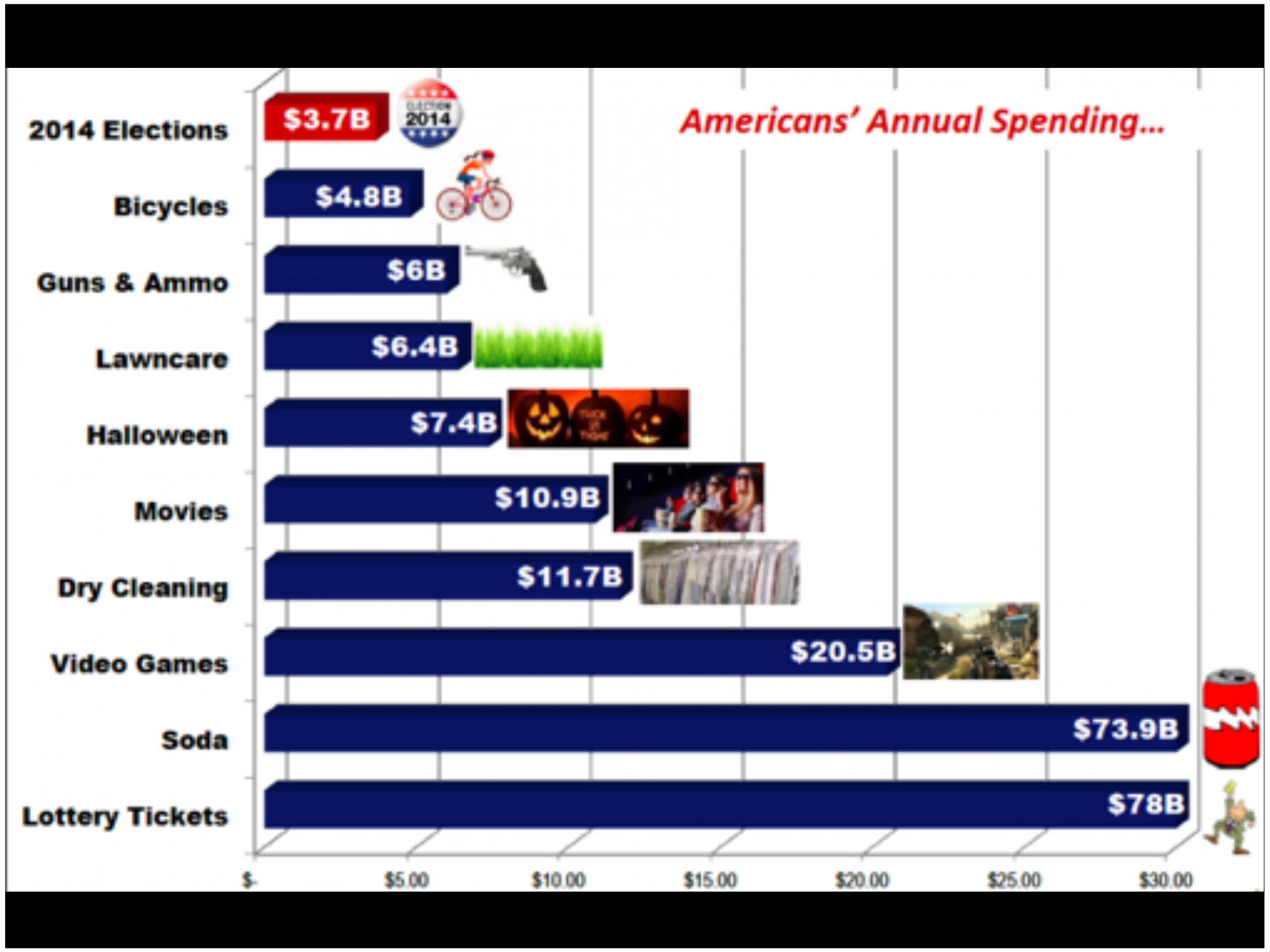
by Charles Joseph Minard



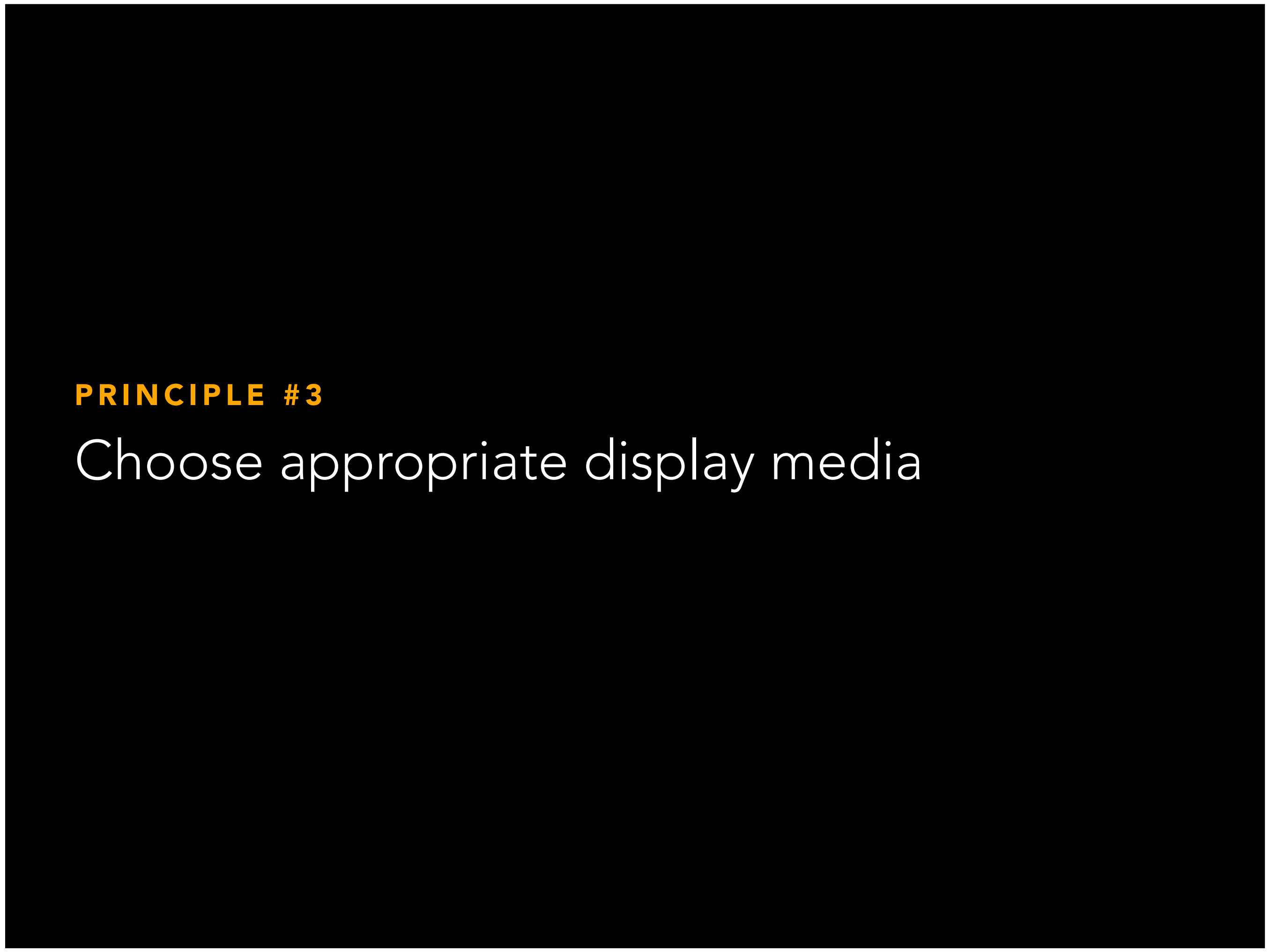
PRINCIPLE #2

Limit visualization to a few highlights

don't overload with color, shapes, text



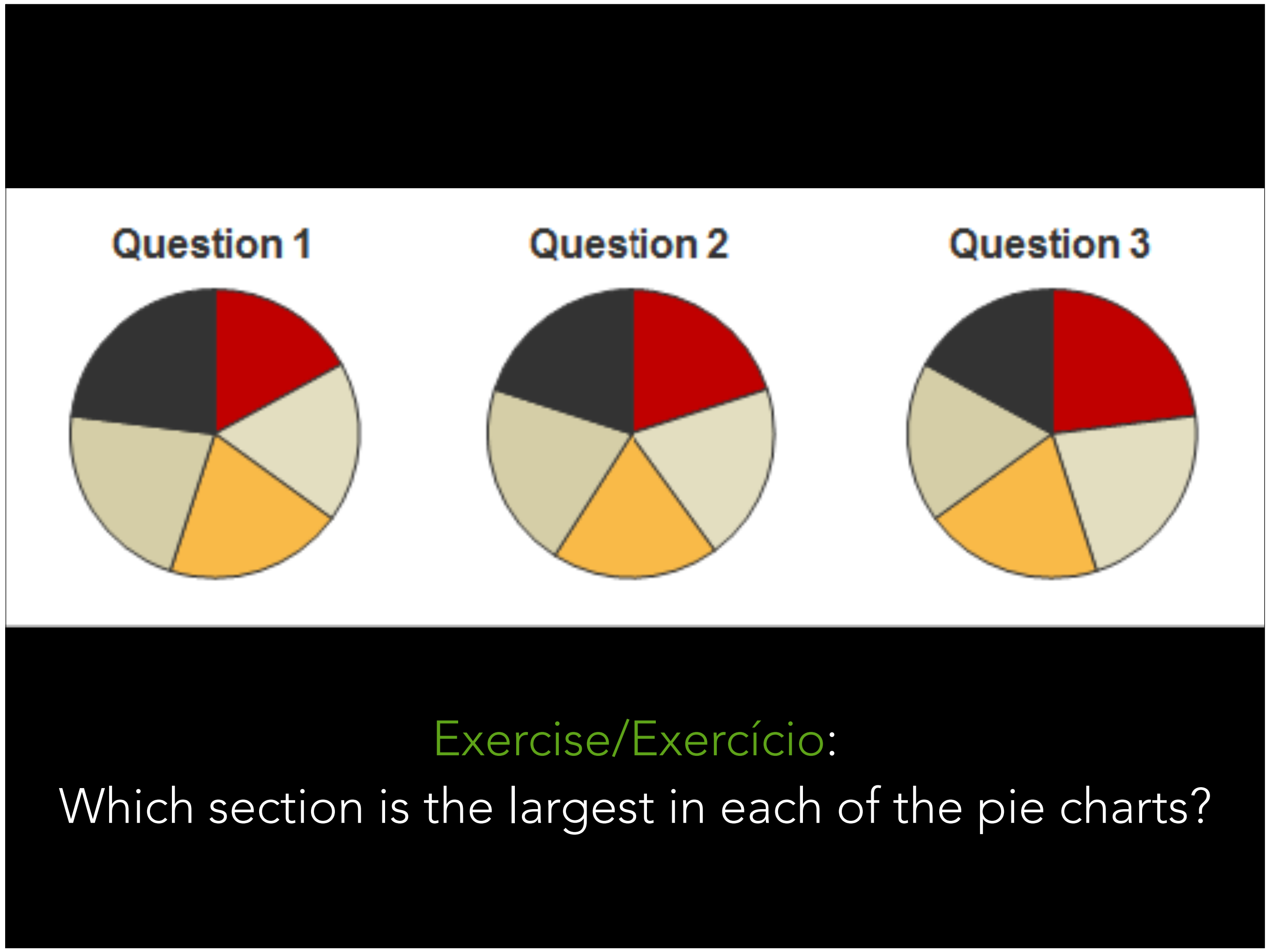
Example of a poor chart



PRINCIPLE #3

Choose appropriate display media

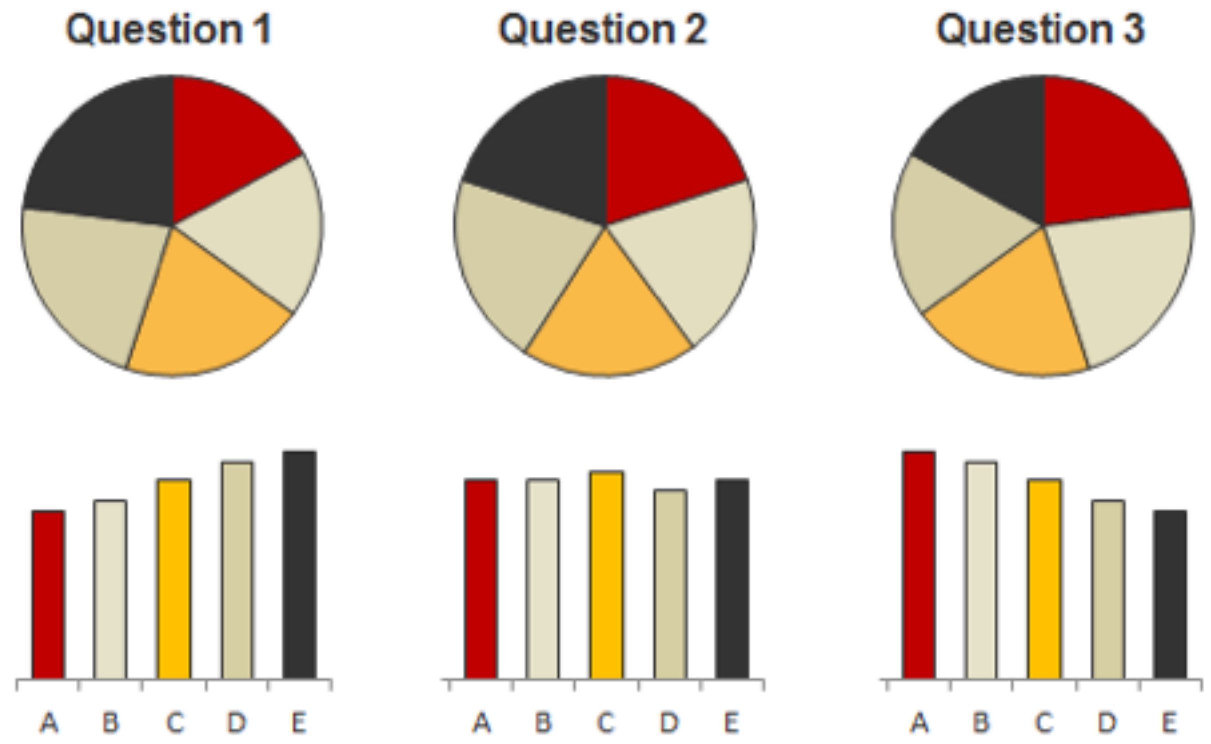
don't overload with color, shapes, text



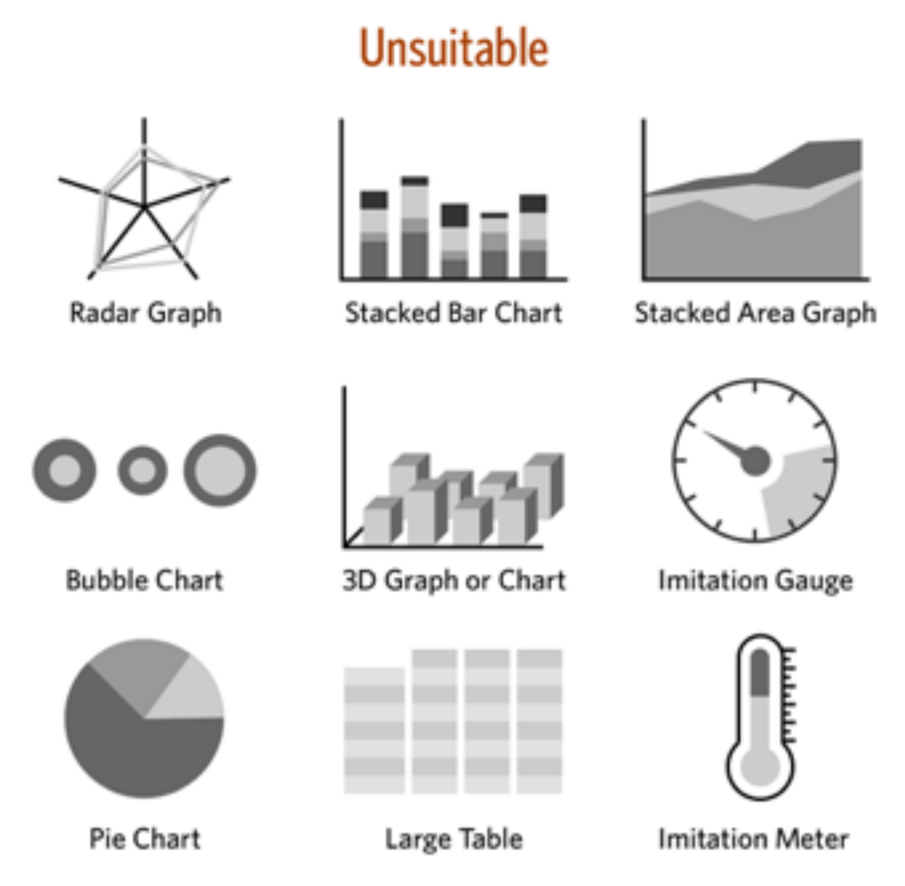
Exercise/Exercício:
Which section is the largest in each of the pie charts?

Example of a poor chart

Bar Charts are Easier to Interpret than Pie Charts

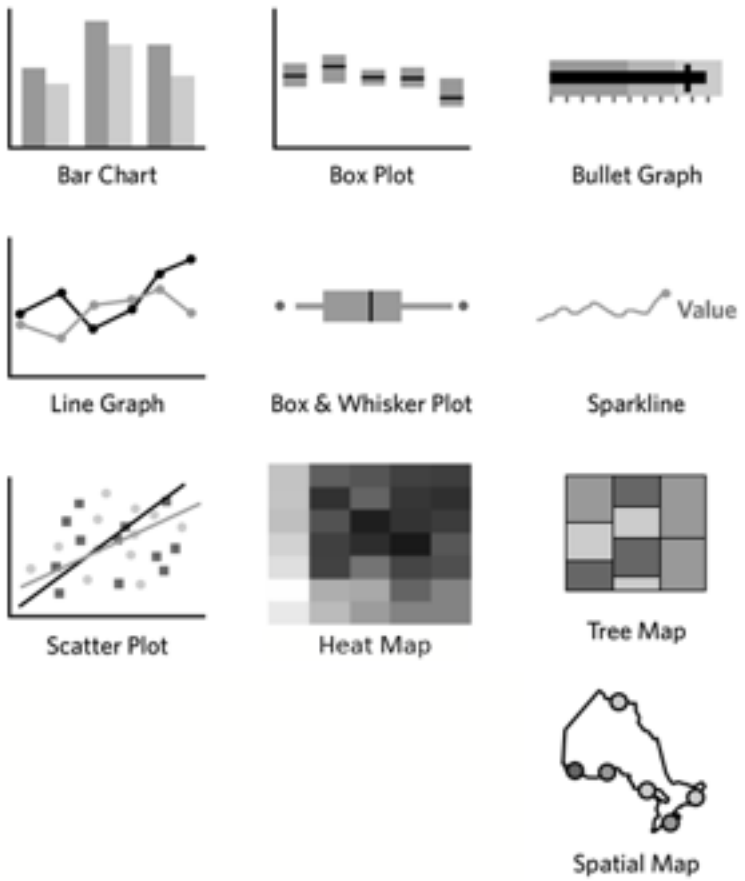


CHOOSING CHARTS



by Stephen Few

Suitable



Bar Chart

Box Plot

Bullet Graph

Line Graph

Box & Whisker Plot

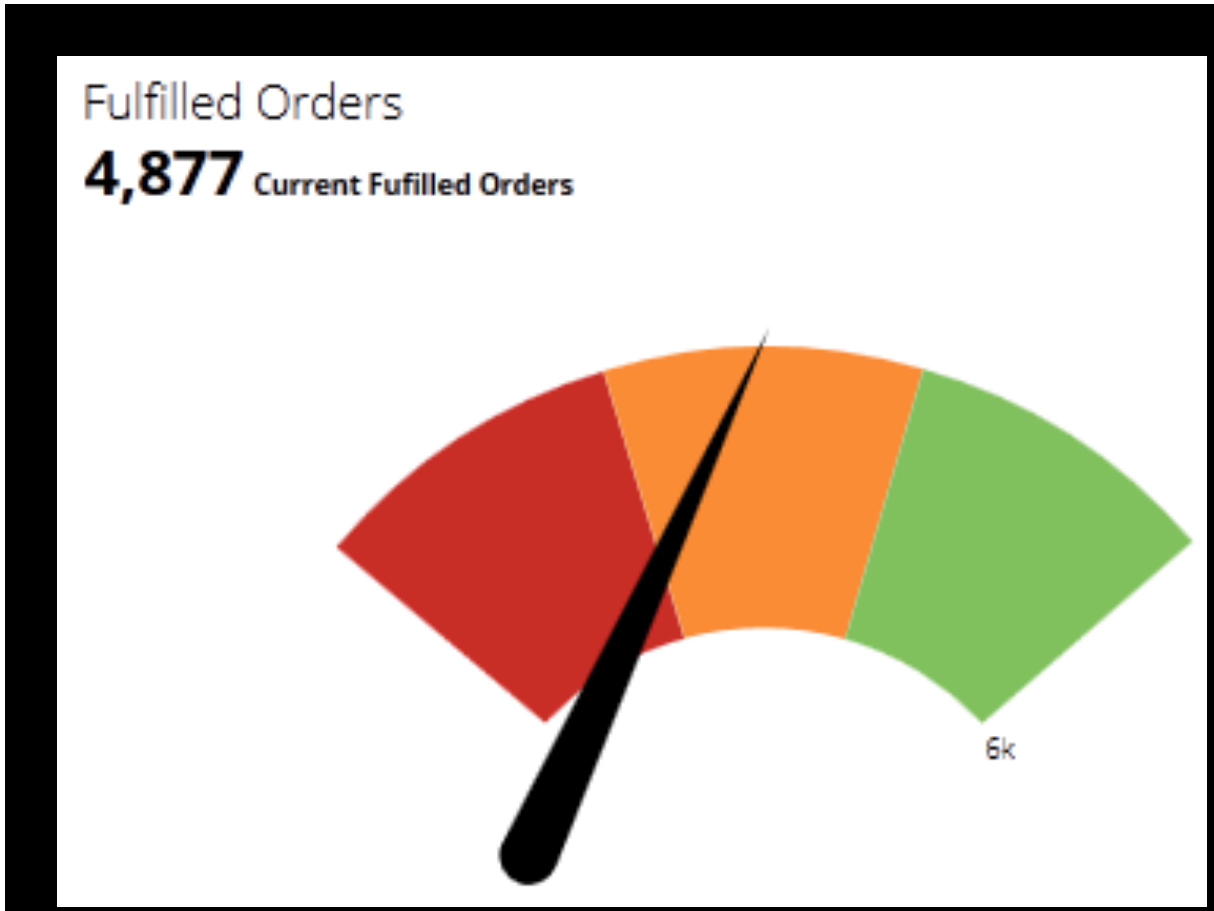
Sparkline

Scatter Plot

Heat Map

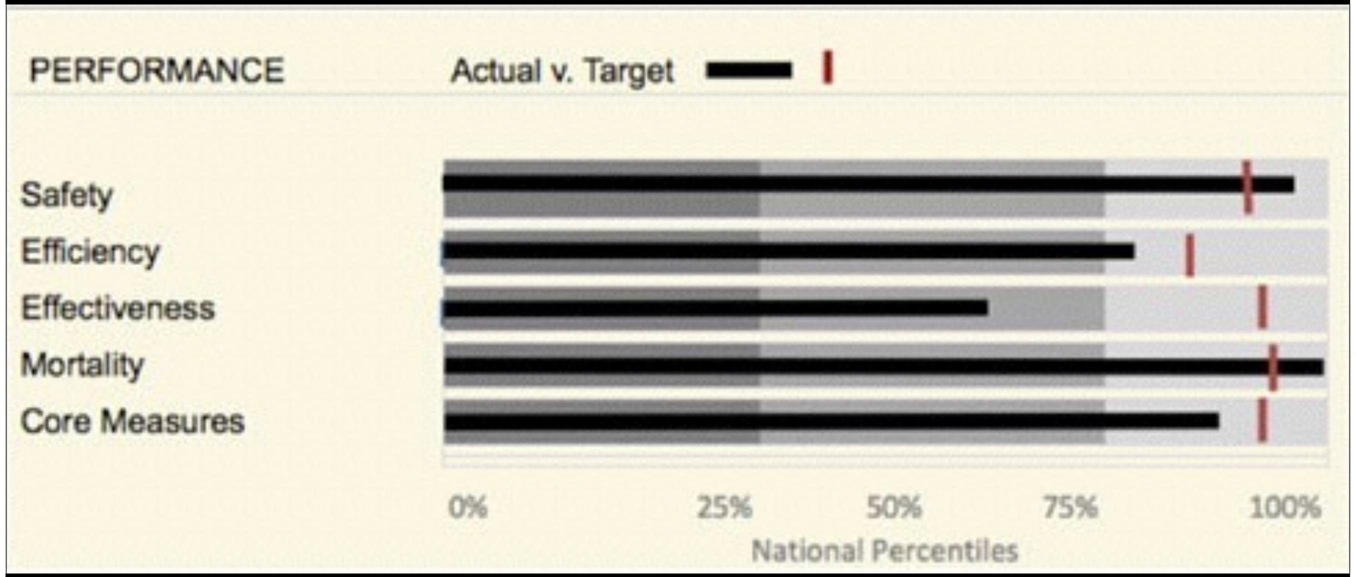
Tree Map

Spatial Map

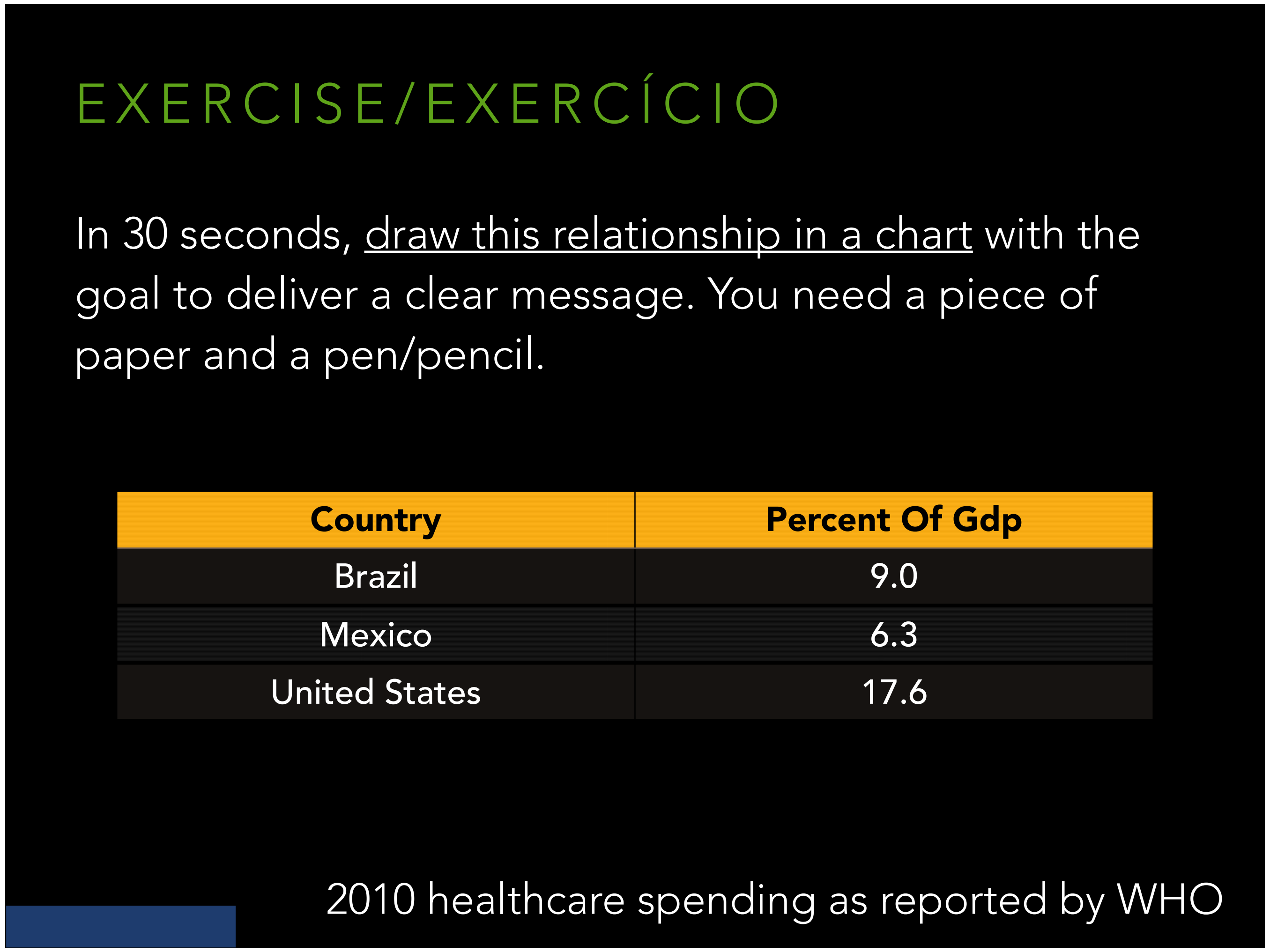


Domo http://www.domo.com/roles/operations

Gauges are distracting, hard to interpret, can easily be done wrong and take up too much space.



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



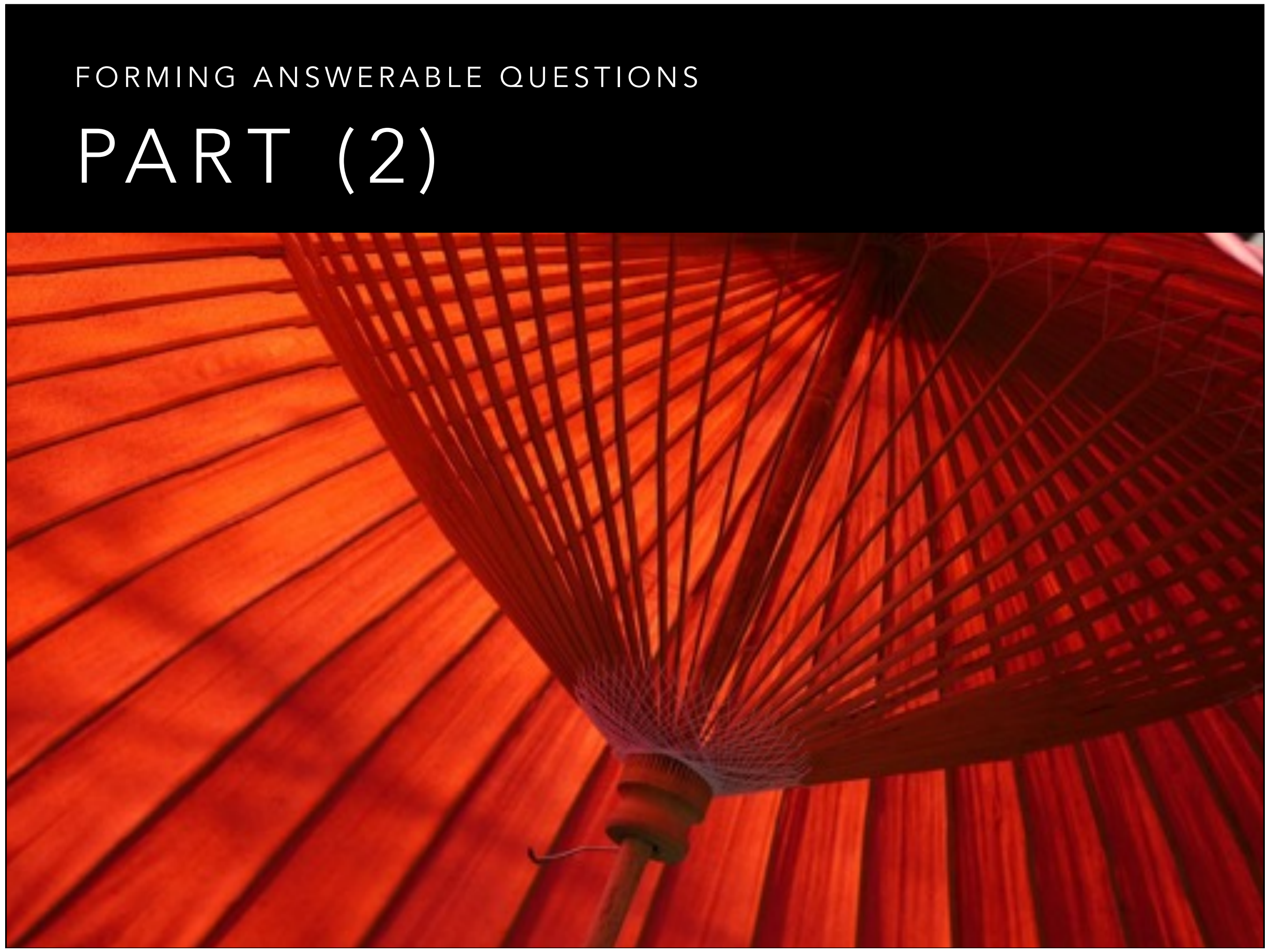
EXERCISE/EXERCÍCIO

In 30 seconds, draw this relationship in a chart with the goal to deliver a clear message. You need a piece of paper and a pen/pencil.

Country	Percent Of Gdp
Brazil	9.0
Mexico	6.3
United States	17.6

2010 healthcare spending as reported by WHO

The right way to do it is to draw a clean-looking bar chart. It doesn’t need any embellishments or extra features, just like it doesn’t need it in the software tools we use.

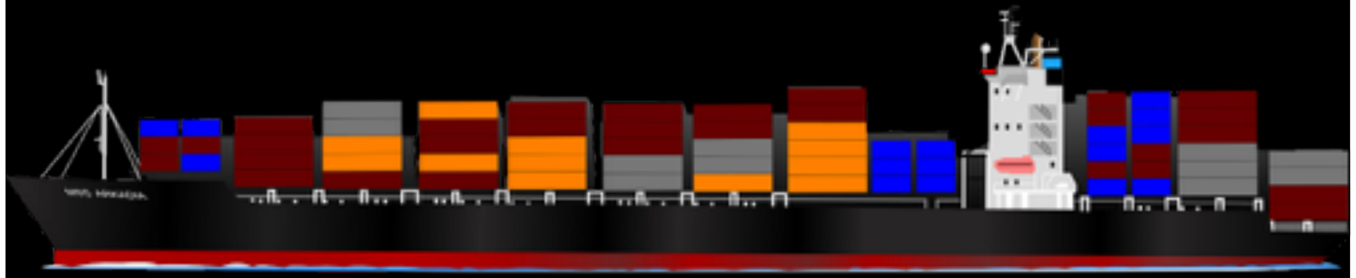


FORMING ANSWERABLE QUESTIONS

PART (2)

Most important parts
Think about what to ask
Think about how to present

KEY PERFORMANCE INDICATORS (KPI)



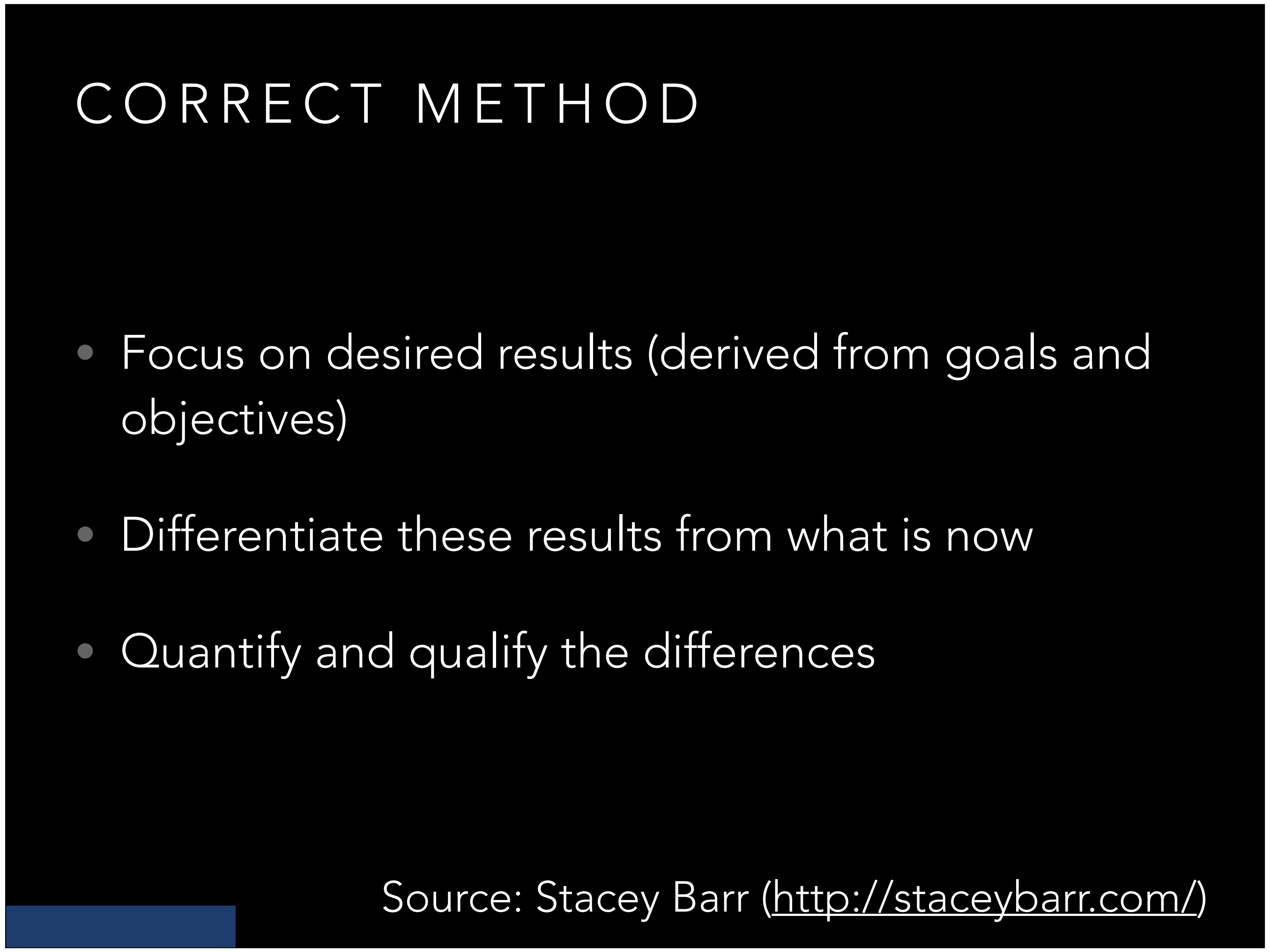
COMMON METHODS IN KPI DEVELOPMENT

Brainstorming

Adoption from other organizations or benchmarking

Using existing measures

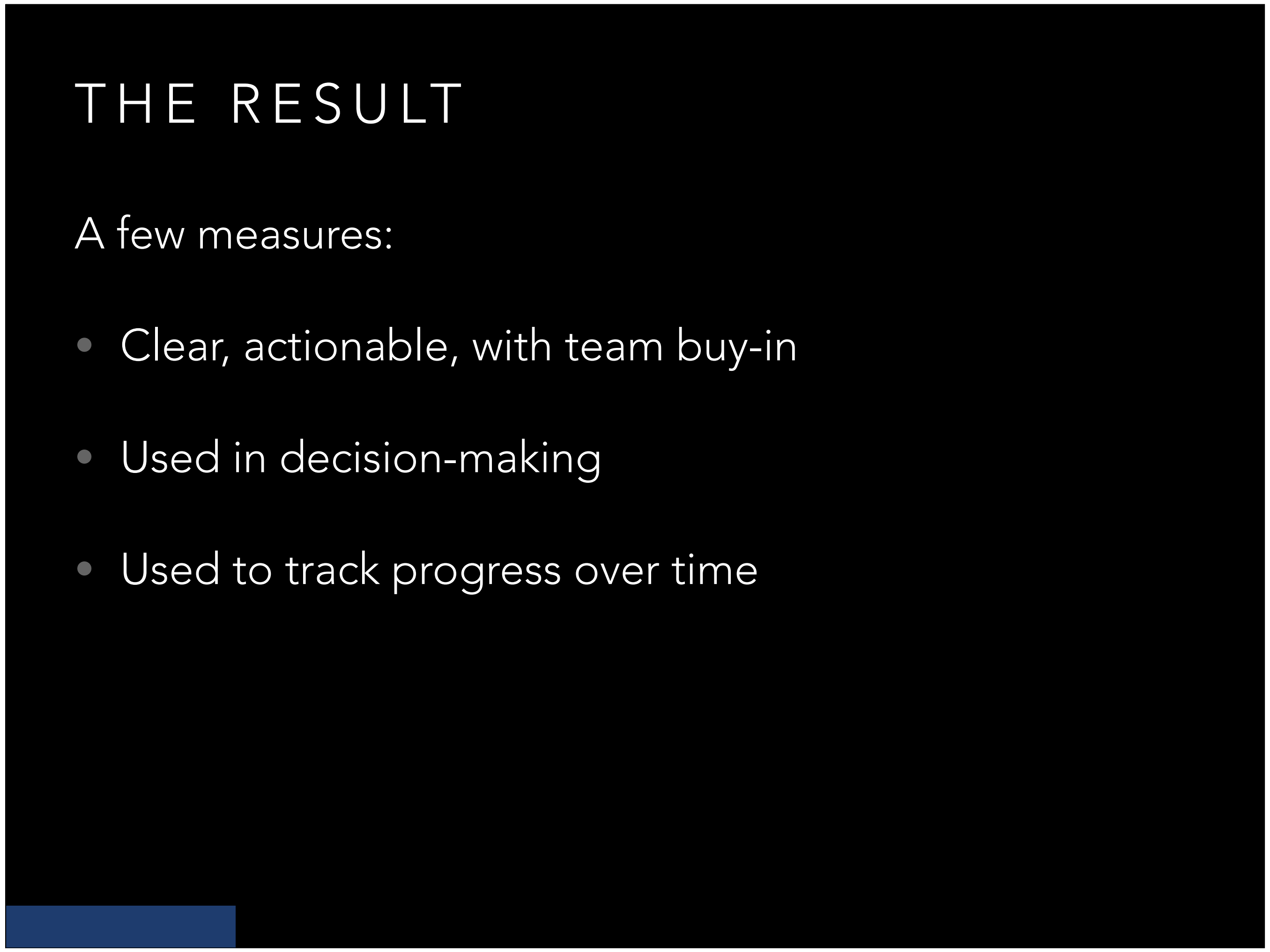
Measures enforced by stakeholders or executives



CORRECT METHOD

- Focus on desired results (derived from goals and objectives)
- Differentiate these results from what is now
- Quantify and qualify the differences

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



THE RESULT

A few measures:

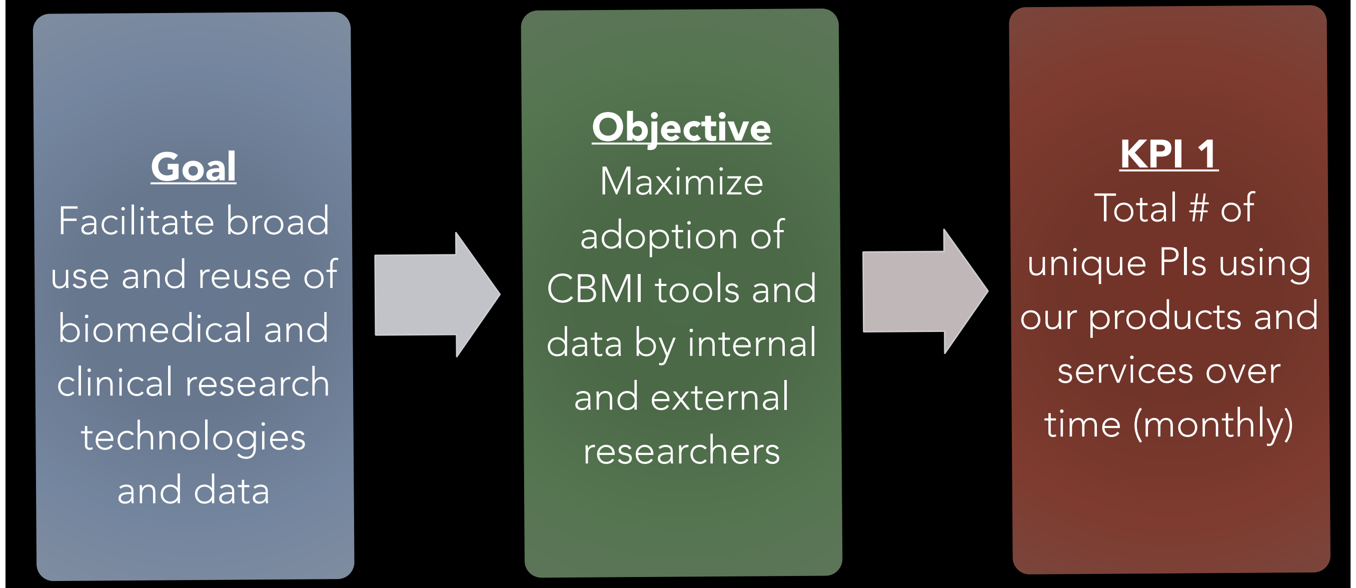
- Clear, actionable, with team buy-in
- Used in decision-making
- Used to track progress over time

towards the strategic result

COMMON PITFALLS OF KPIS

- Data inaccessible
- No consistency or clarity in measurement

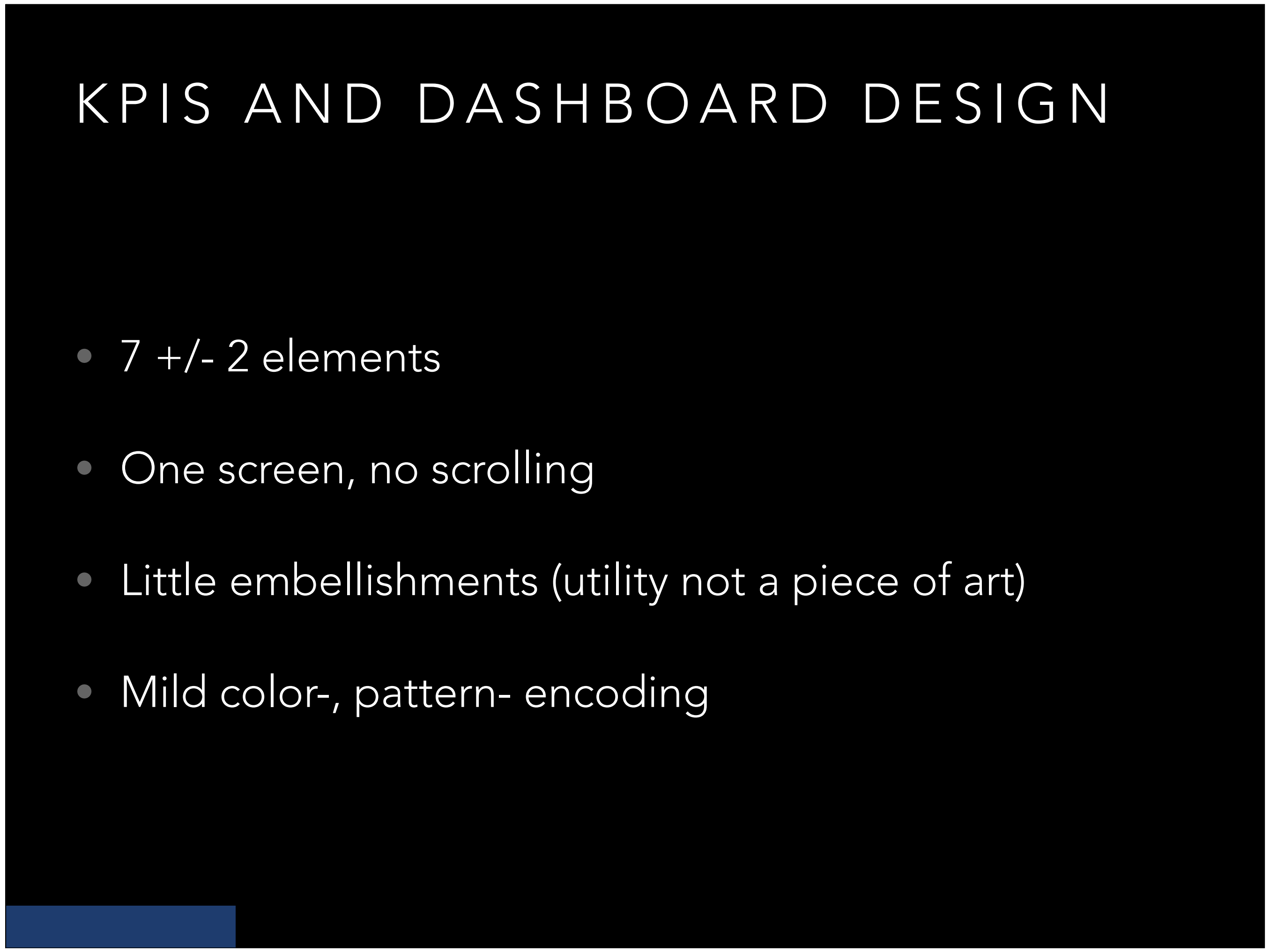
WASHU-CBMI EXAMPLE



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 over time (monthly)



KPIS AND DASHBOARD DESIGN

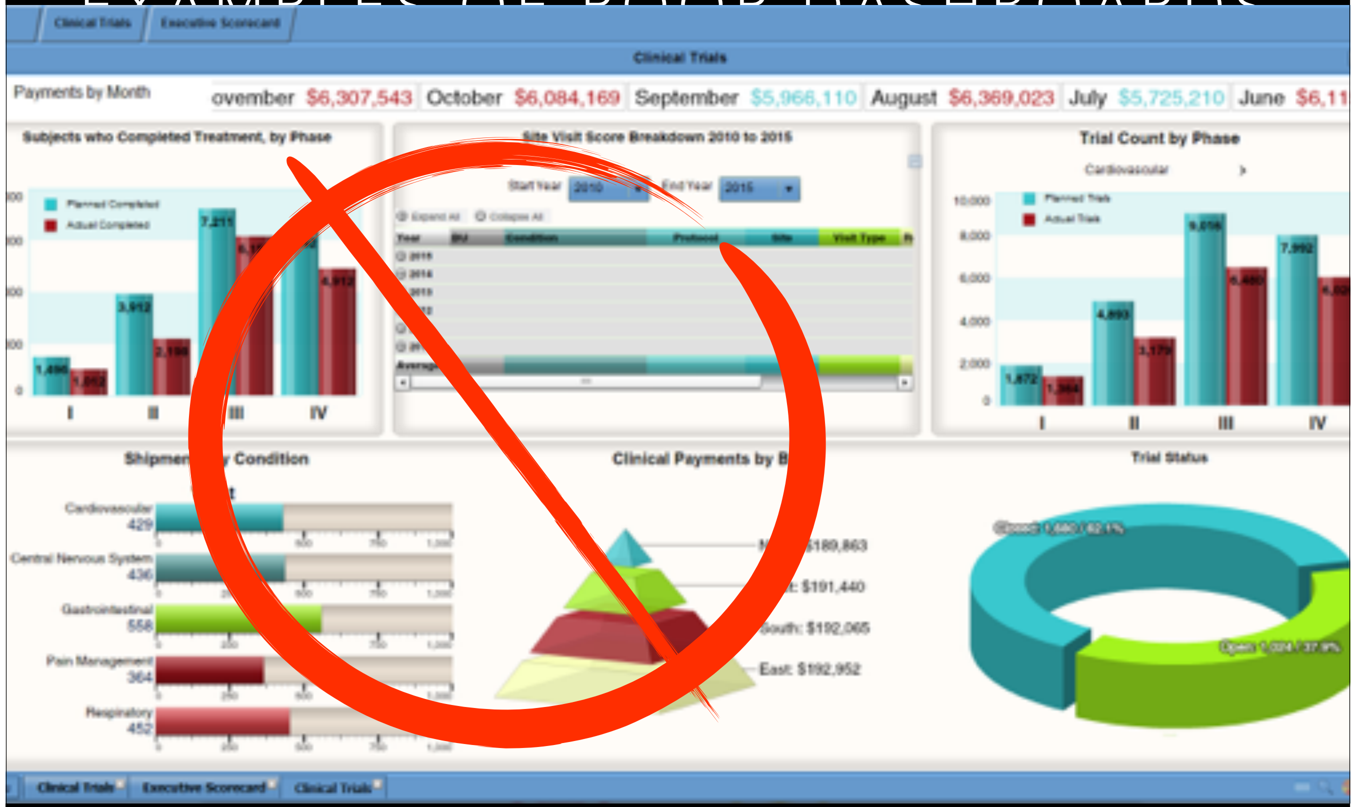
- 7 +/- 2 elements
- One screen, no scrolling
- Little embellishments (utility not a piece of art)
- Mild color-, pattern- encoding

Main principles of dashboard design by Stephen Few

EXAMPLES OF DASHBOARDS

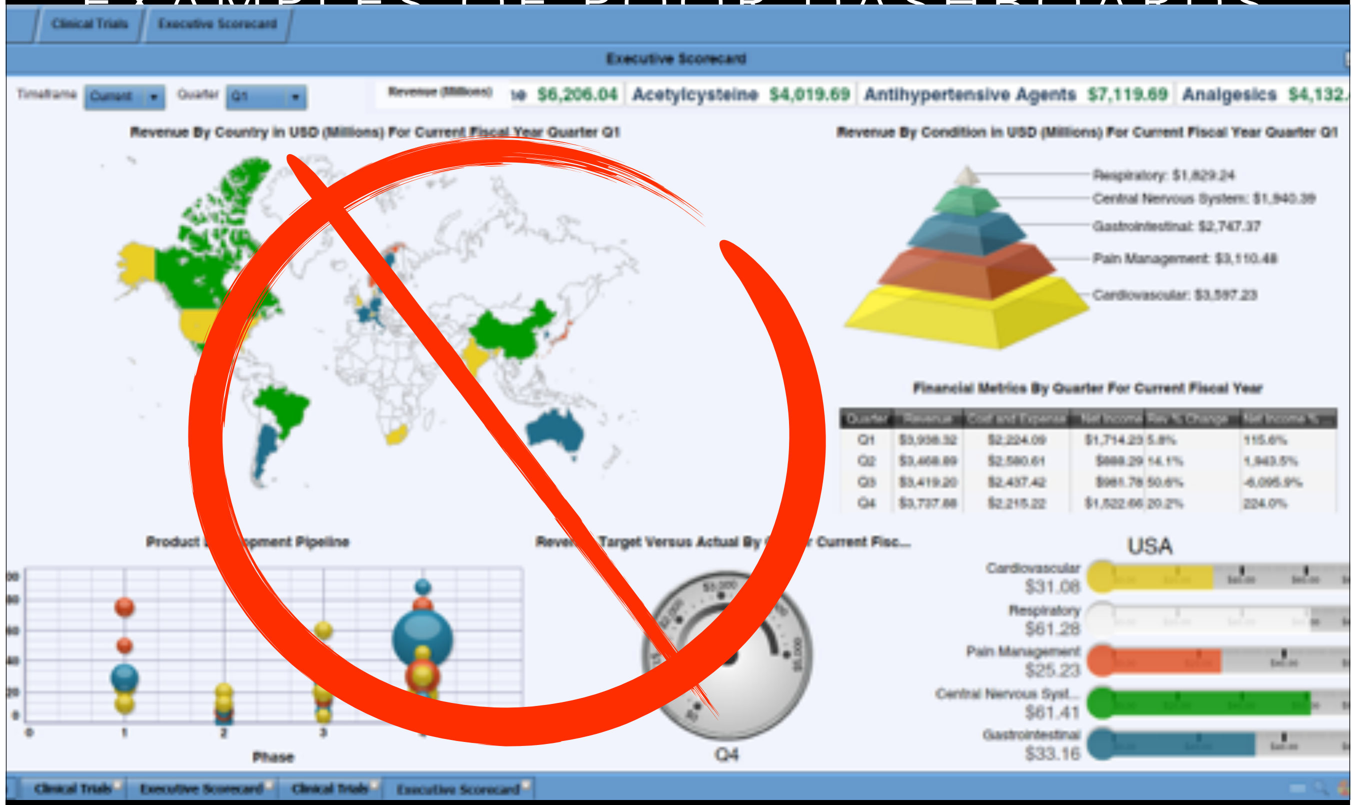
1 and 2 are from iDashboards Pharmaceuticals demo

EXAMPLES OF POOR DASHBOARDS



from iDashboards Pharmaceuticals

EXAMPLES OF POOR DASHBOARDS



Executive Scorecard

Timeframe: Current | Quarter: Q1

Revenue (Millions) …e $6,206.04 | Acetylcysteine $4,019.69 | Antihypertensive Agents $7,119.69 | Analgesics $4,132.…

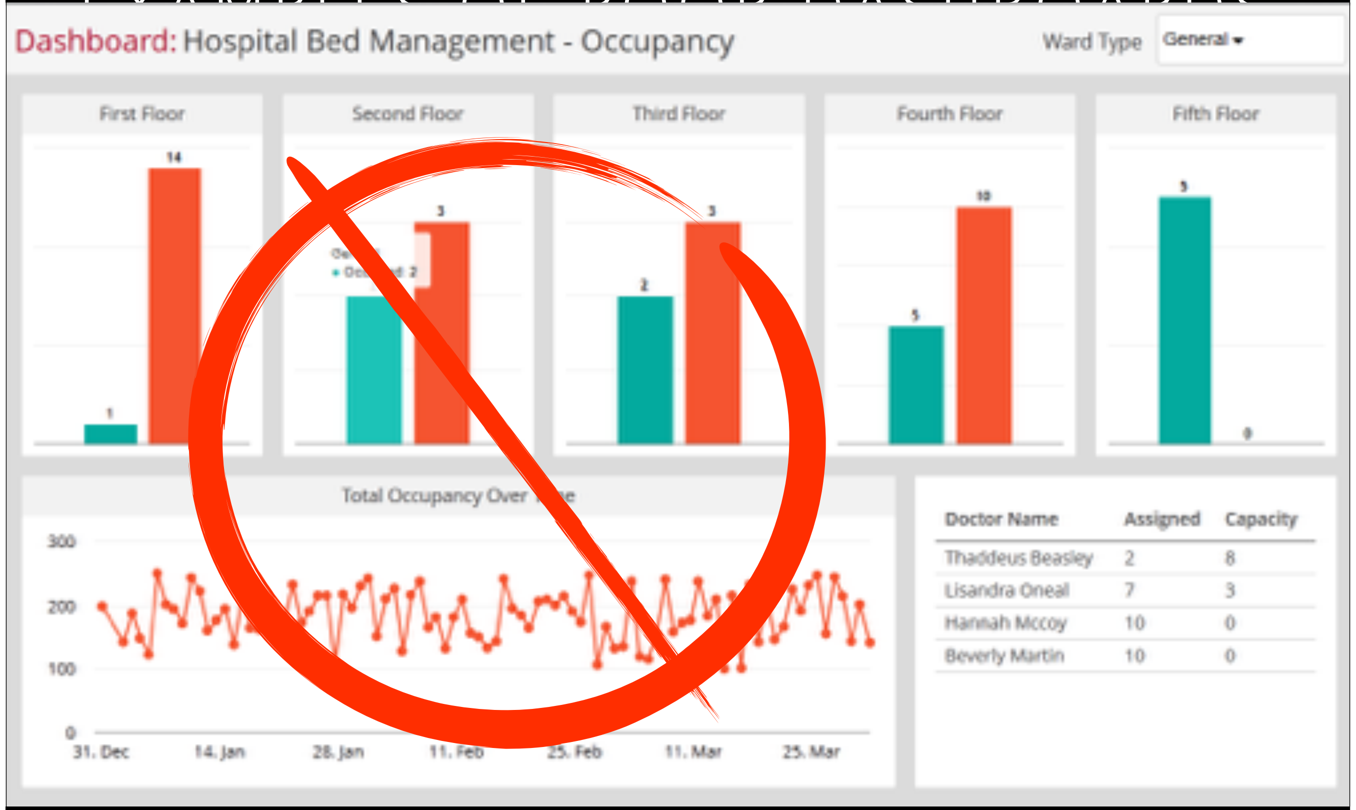
Financial Metrics By Quarter For Current Fiscal Year

Quarter	Revenue	Cost and Expense	Net Income	Rev % Change	Net Income % …
Q1	$3,938.32	$2,224.09	$1,714.23	5.8%	115.6%
Q2	$3,468.89	$2,580.61	$888.29	14.1%	1,943.5%
Q3	$3,419.20	$2,437.42	$981.78	50.6%	-6,095.9%
Q4	$3,737.88	$2,215.22	$1,522.66	20.2%	224.0%

Clinical Trials
Executive Scorecard
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
Product … Pipeline
Phase
Reve… Target Versus Actual By … Current Fisc...
Q4
USA
Cardiovascular
$31.08
Respiratory
$61.28
Pain Management
$25.23
Central Nervous Syst...
$61.41
Gastrointestinal
$33.16
Clinical Trials
Executive Scorecard
Clinical Trials
Executive Scorecard

from iDashboards Pharmaceuticals

EXAMPLES OF POOR DASHBOARDS

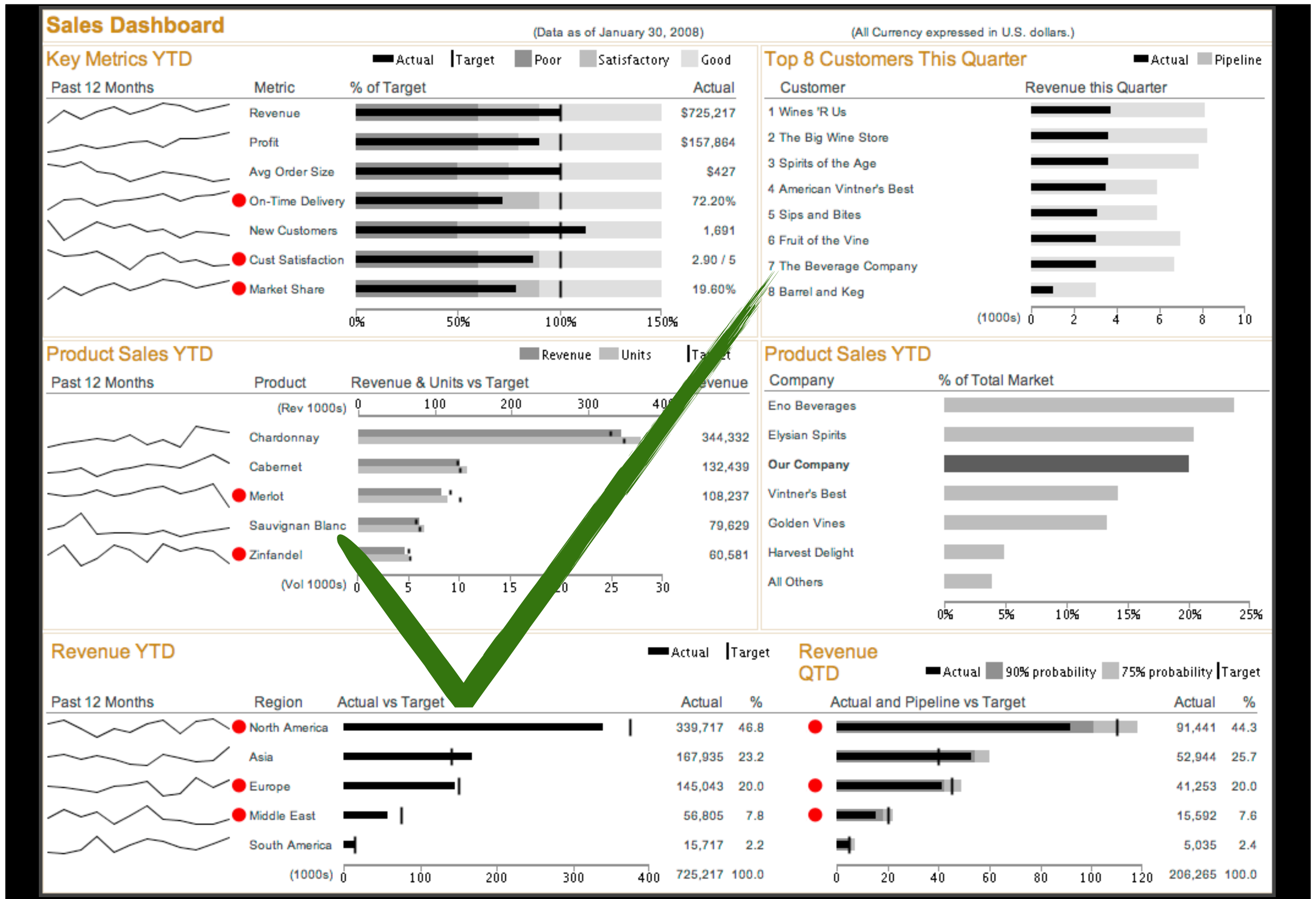


Dashboard: Hospital Bed Management - Occupancy

Ward Type General ▾

Doctor Name	Assigned	Capacity
Thaddeus Beasley	2	8
Lisandra Oneal	7	3
Hannah Mccoy	10	0
Beverly Martin	10	0

from Clicdata demo



Sales Dashboard (Data as of January 30, 2008) (All Currency expressed in U.S. dollars.)

Key Metrics YTD — Actual | Target | Poor | Satisfactory | Good

Past 12 Months	Metric	% of Target	Actual
	Revenue		$725,217
	Profit		$157,864
	Avg Order Size		$427
	● On-Time Delivery		72.20%
	New Customers		1,691
	● Cust Satisfaction		2.90 / 5
	● Market Share		19.60%

Top 8 Customers This Quarter — Actual | Pipeline

Customer	Revenue this Quarter
1 Wines 'R Us	
2 The Big Wine Store	
3 Spirits of the Age	
4 American Vintner's Best	
5 Sips and Bites	
6 Fruit of the Vine	
7 The Beverage Company	
8 Barrel and Keg	

Product Sales YTD — Revenue | Units | Target

Past 12 Months	Product	Revenue & Units vs Target	Revenue
	Chardonnay		344,332
	Cabernet		132,439
	● Merlot		108,237
	Sauvignan Blanc		79,629
	● Zinfandel		60,581

Product Sales YTD

Company	% of Total Market
Eno Beverages	
Elysian Spirits	
Our Company	
Vintner's Best	
Golden Vines	
Harvest Delight	
All Others	

Revenue YTD — Actual | Target

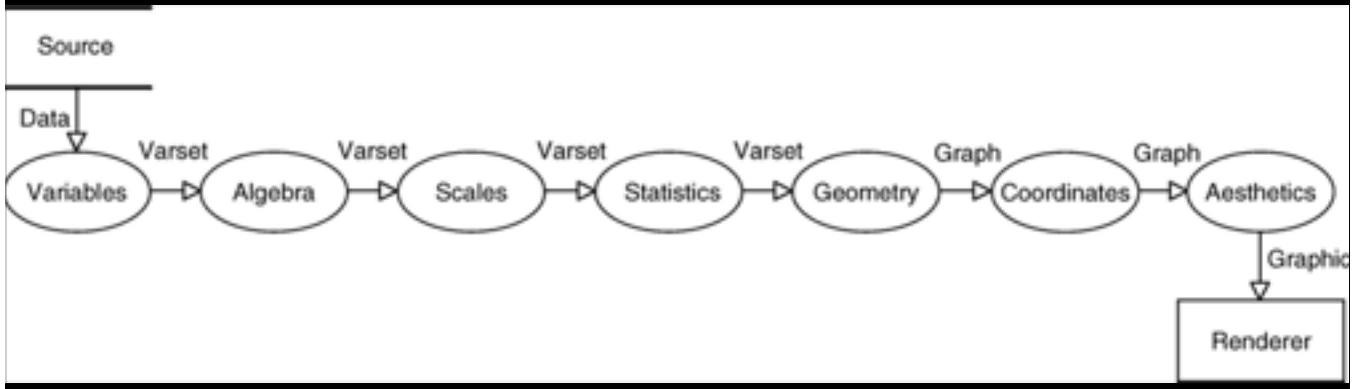
Past 12 Months	Region	Actual vs Target	Actual	%
	● North America		339,717	46.8
	Asia		167,935	23.2
	● Europe		145,043	20.0
	● Middle East		56,805	7.8
	South America		15,717	2.2
	(1000s)	0 100 200 300 400	725,217	100.0

Revenue QTD — Actual | 90% probability | 75% probability | Target

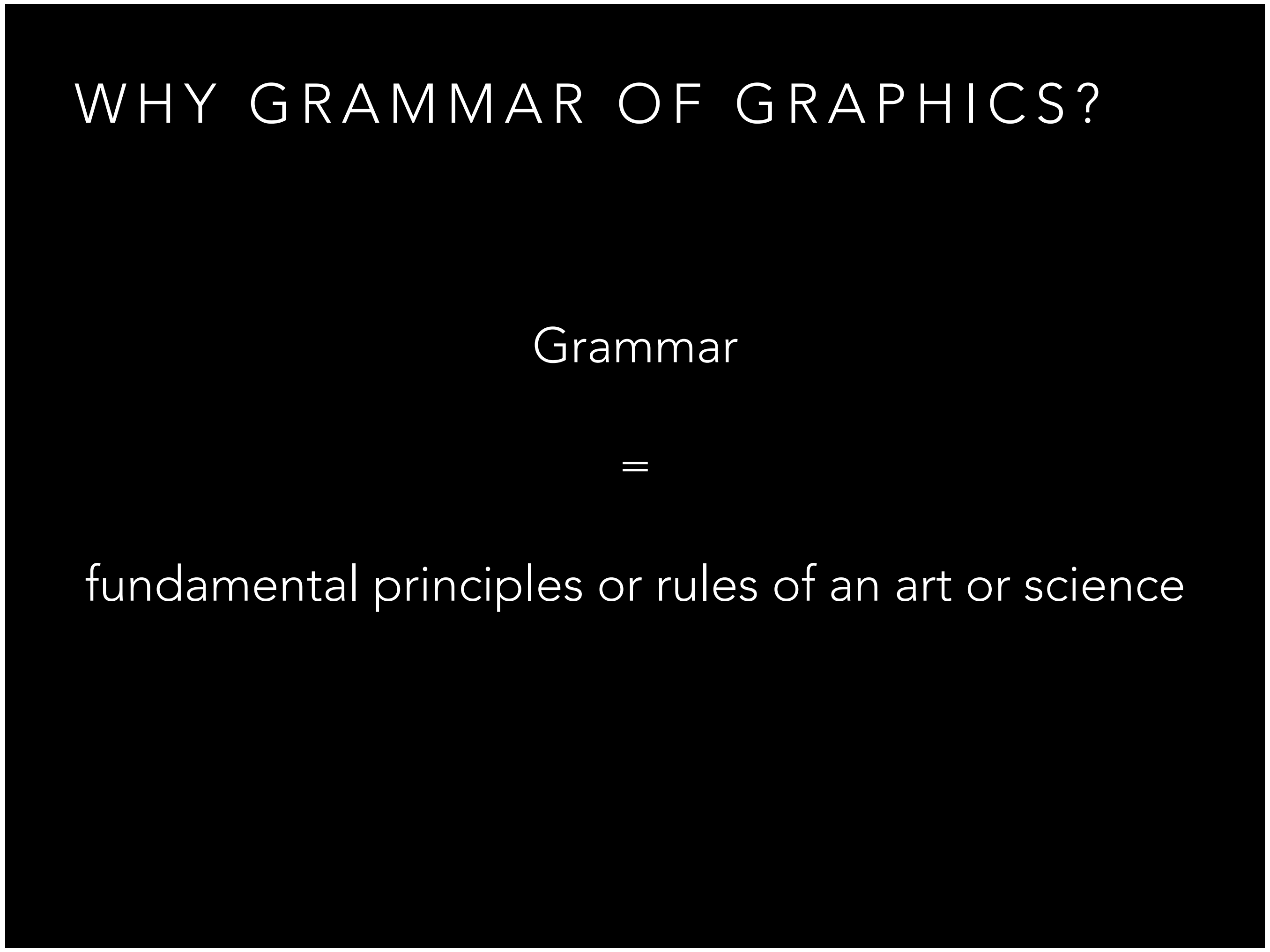
Actual and Pipeline vs Target	Actual	%
●	91,441	44.3
	52,944	25.7
●	41,253	20.0
●	15,592	7.6
	5,035	2.4
0 20 40 60 80 100 120	206,265	100.0

by Stephen Few

BRIEF INTRO TO GRAMMAR OF GRAPHICS



from Wilkenson



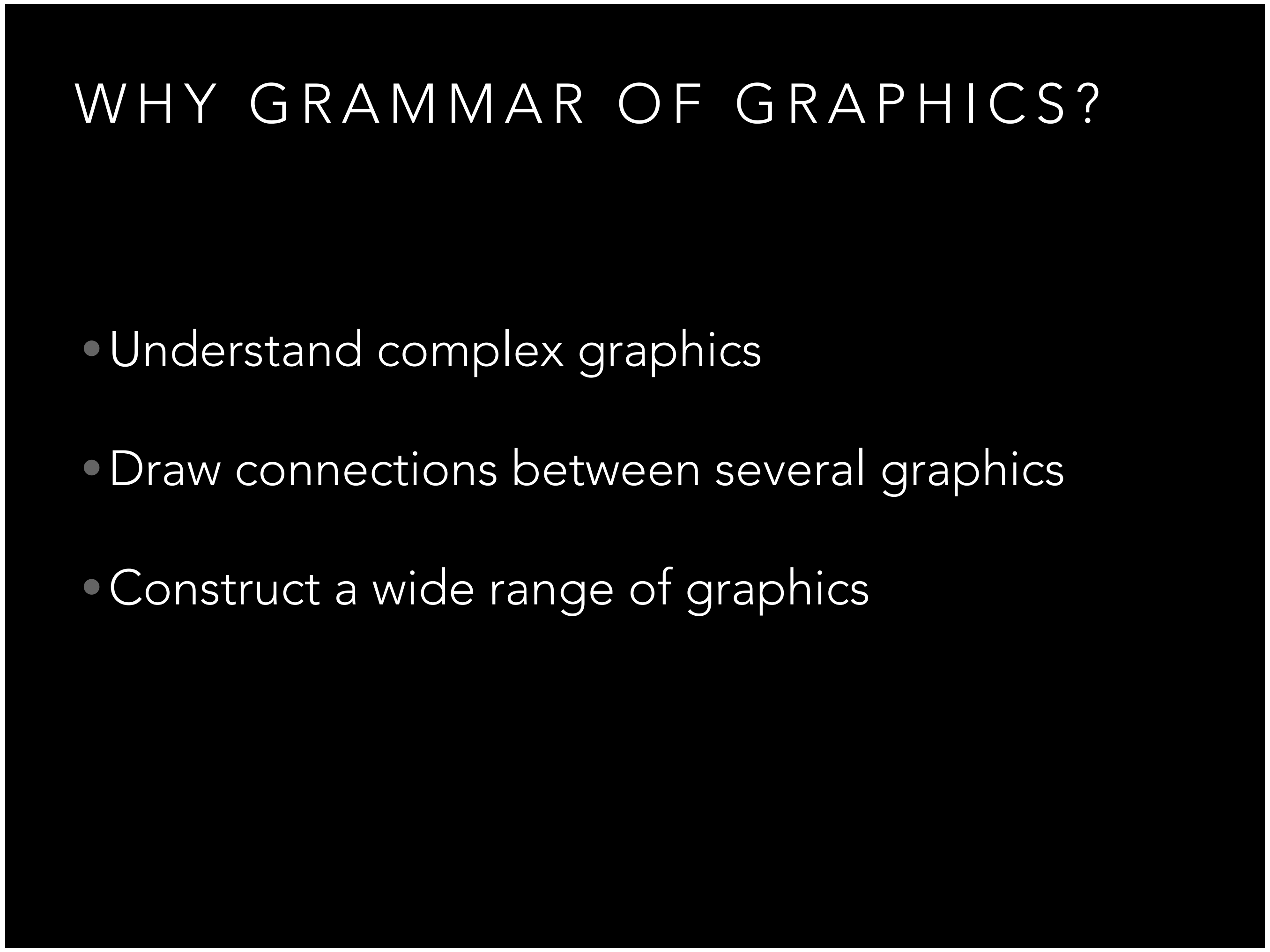
WHY GRAMMAR OF GRAPHICS?

Grammar

=

fundamental principles or rules of an art or science

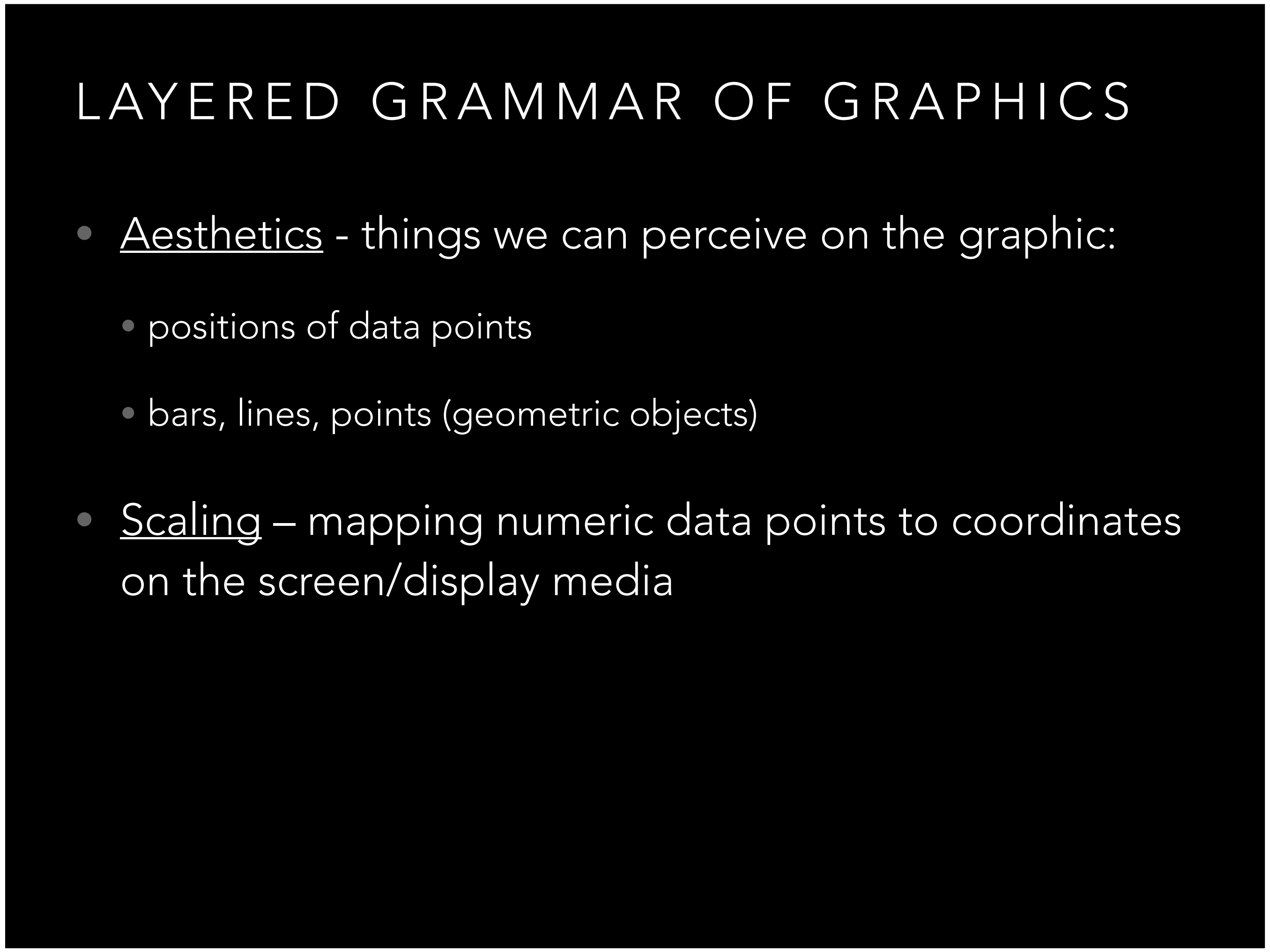
One way to answer questions about statistical graphics is to develop a grammar: “the fundamental principles or rules of an art or science” (OED Online 1989).



WHY GRAMMAR OF GRAPHICS?

- Understand complex graphics
- Draw connections between several graphics
- Construct a wide range of graphics

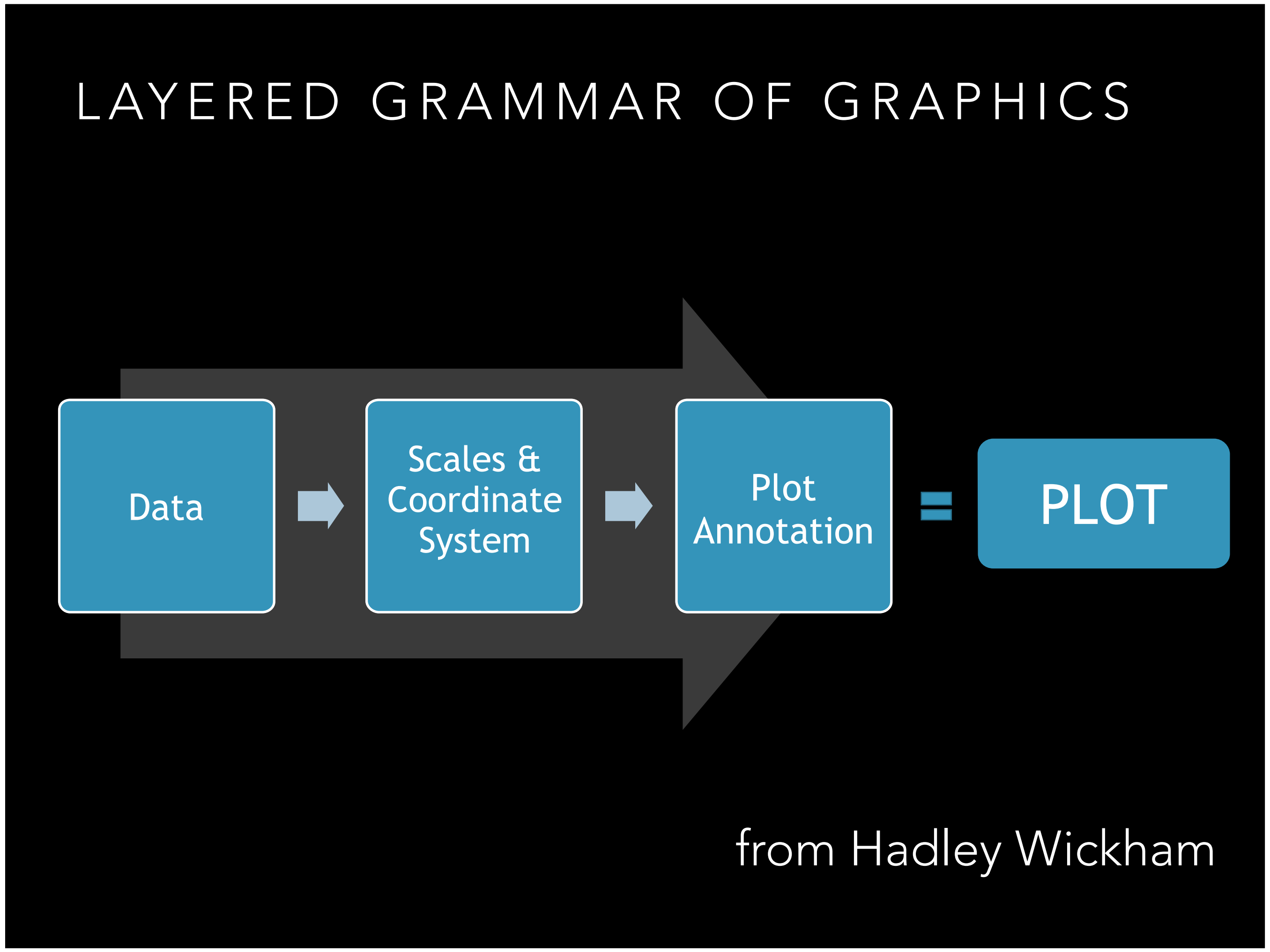
A good grammar will allow us to gain insight into the composition of complicated graphics, and reveal unexpected connections between seemingly different graphics. (Cox 1979)



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.

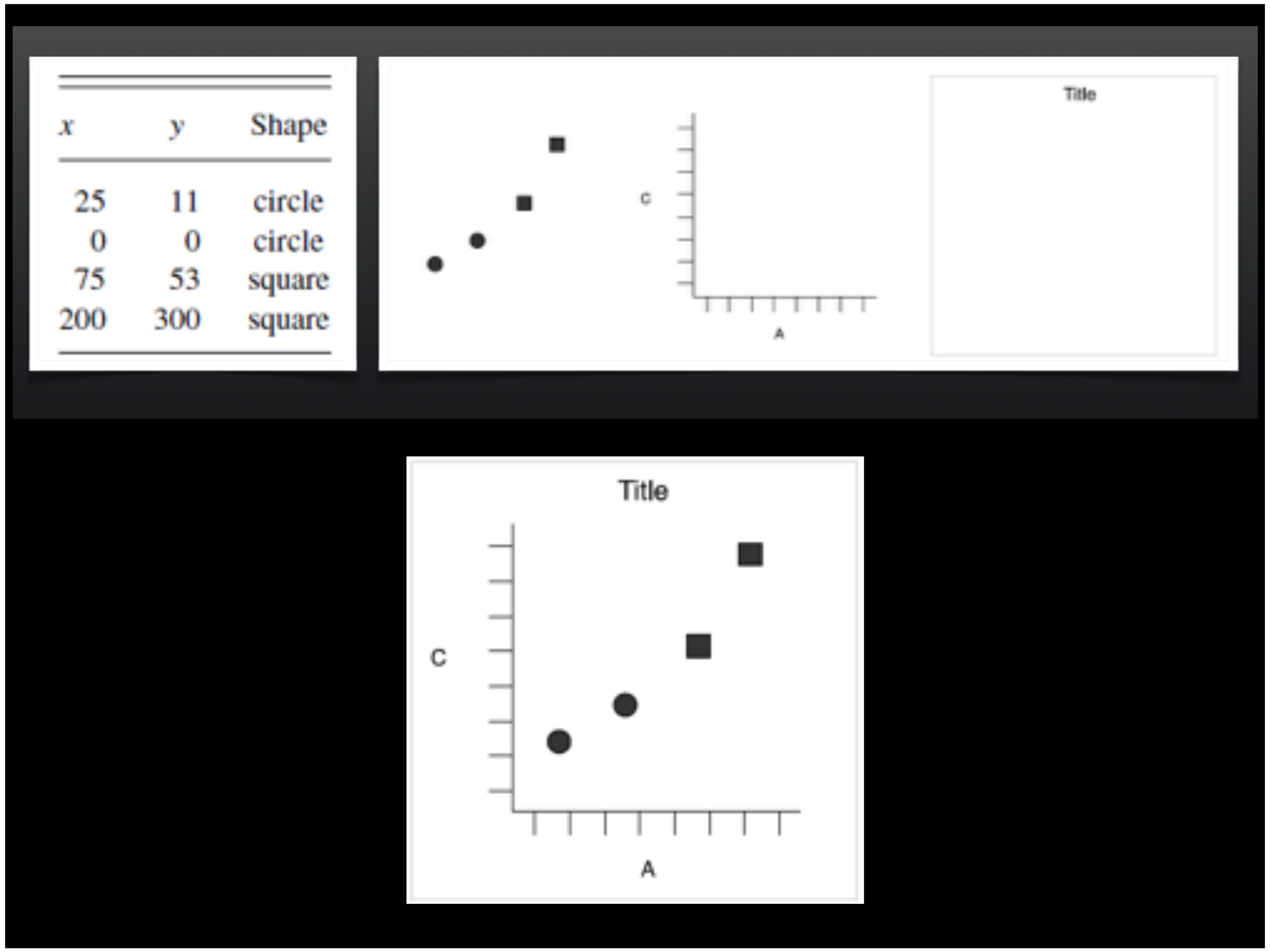


LAYERED GRAMMAR OF GRAPHICS

Data → Scales & Coordinate System → Plot Annotation = PLOT

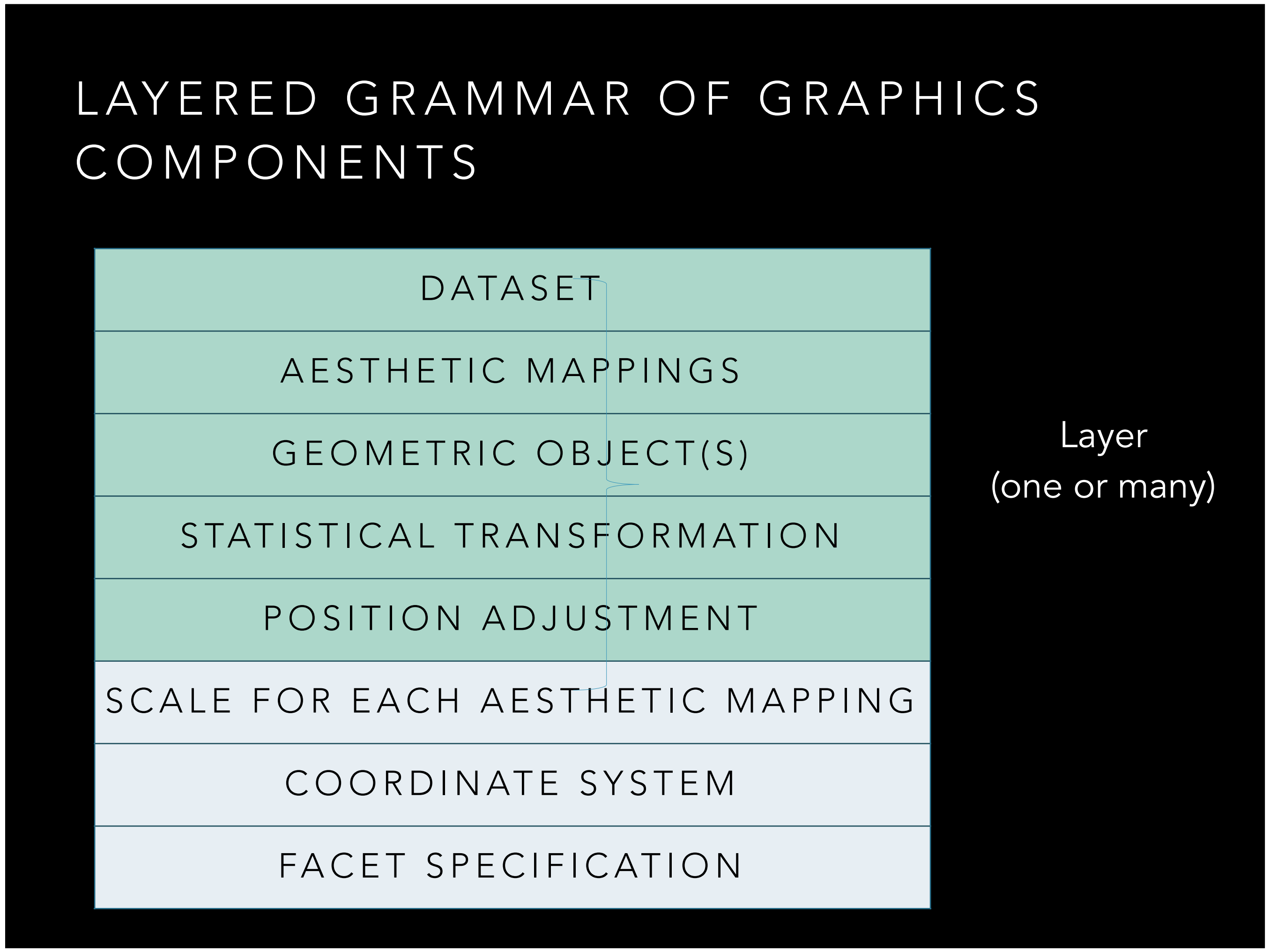
from Hadley Wickham

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.



x	y	Shape
25	11	circle
0	0	circle
75	53	square
200	300	square

Fore more complex plots
Faceting splits the original dataset into a dataset for each subset, so the data that underlie Figure 3 look like Table 4.
Scale transformation occurs before statistical transformation so that statistics are computed on the scale-transformed data. This ensures that a plot of log*(x)* versus log*(y)* on linear scales looks the same as *x* versus *y* on log scales. See Section 6.3 for more details. Transformation is only necessary for nonlinear scales, because all statistics are location-scale invariant.



LAYERED GRAMMAR OF GRAPHICS COMPONENTS

DATASET

AESTHETIC MAPPINGS

GEOMETRIC OBJECT(S)

STATISTICAL TRANSFORMATION

POSITION ADJUSTMENT

SCALE FOR EACH AESTHETIC MAPPING

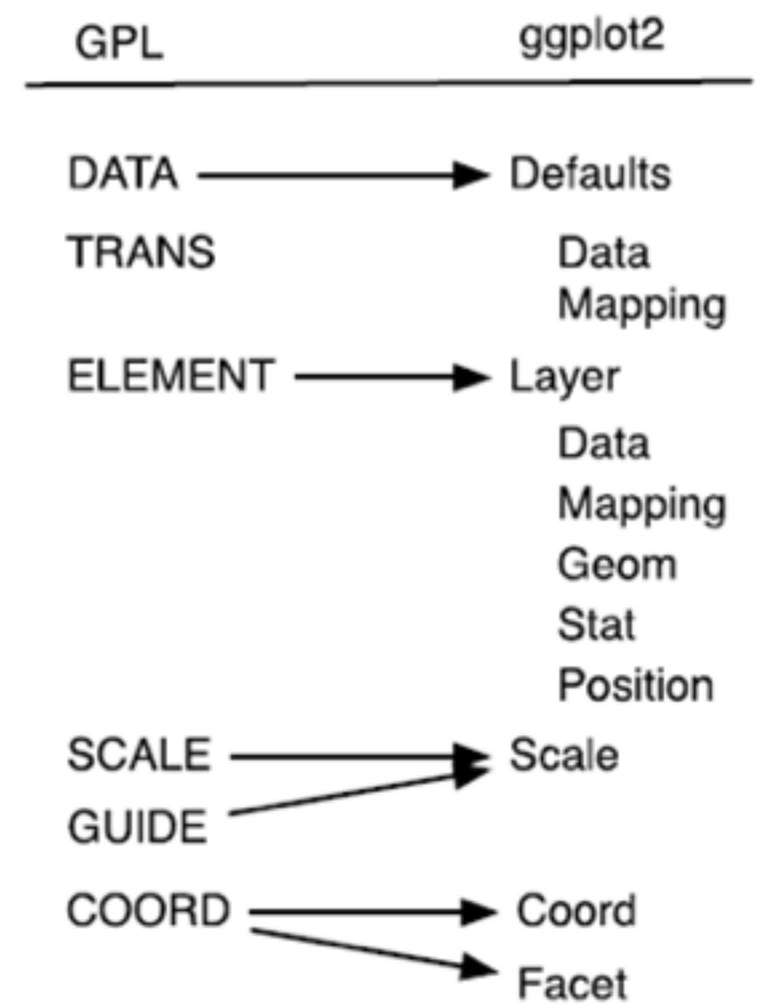
COORDINATE SYSTEM

FACET SPECIFICATION

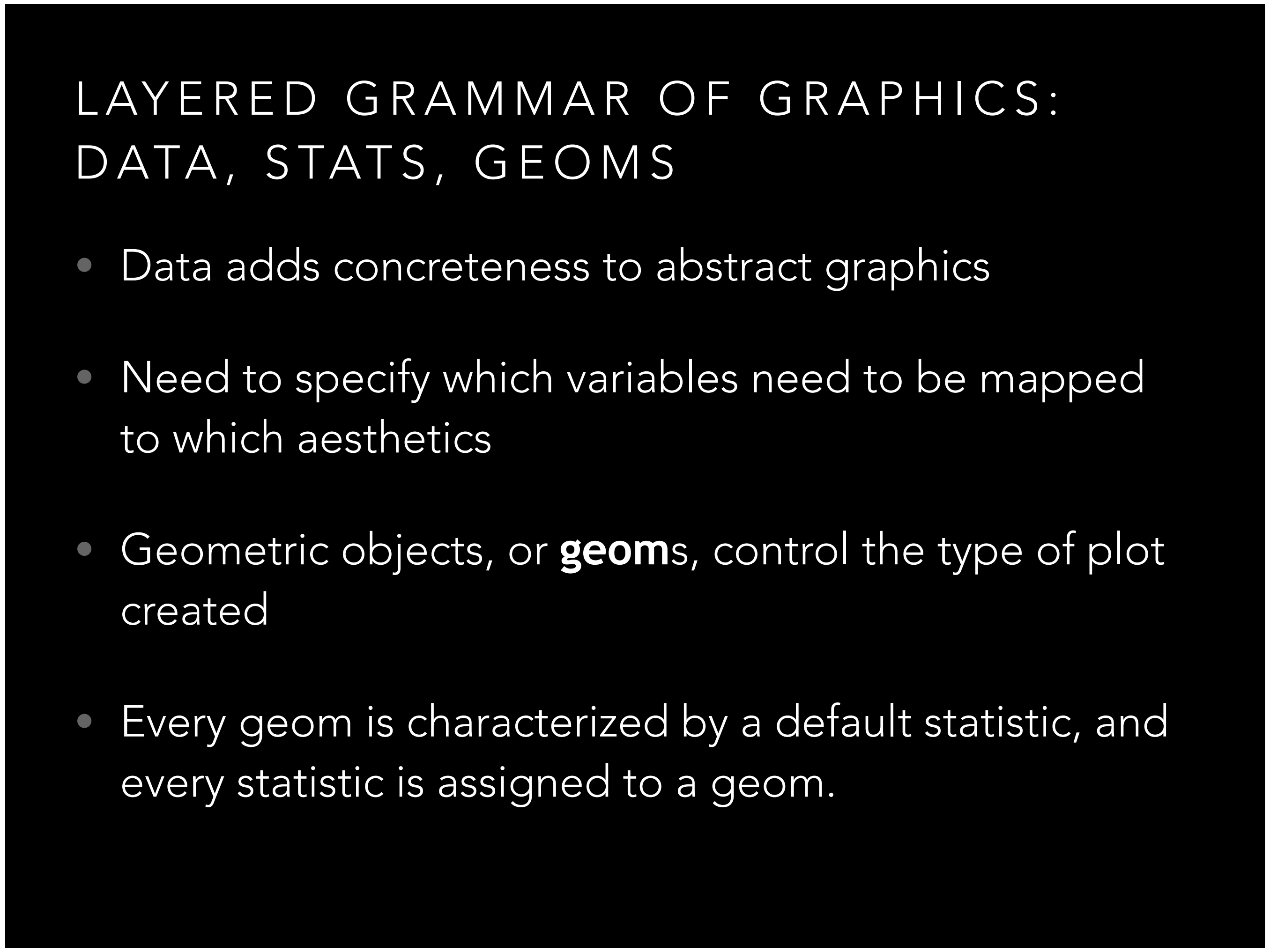
Layer
(one or many)

The layer component is particularly important as it determines the physical representation of the data
In practice, many plots have (at least) three layers: the data, context for the data, and a statistical summary of the data.
Layers are responsible for creating the objects that we perceive on the plot. A layer is
composed of four parts:
• data and aesthetic mapping,
a statistical transformation (stat),
• a geometric object (geom), and
• a position adjustment.

COMPARING COMPONENTS



Mapping between components of Wilkinson’s grammar (left) and the layered grammar (right). TRANS has no correspondence in ggplot2: its role is played by built-in R features.

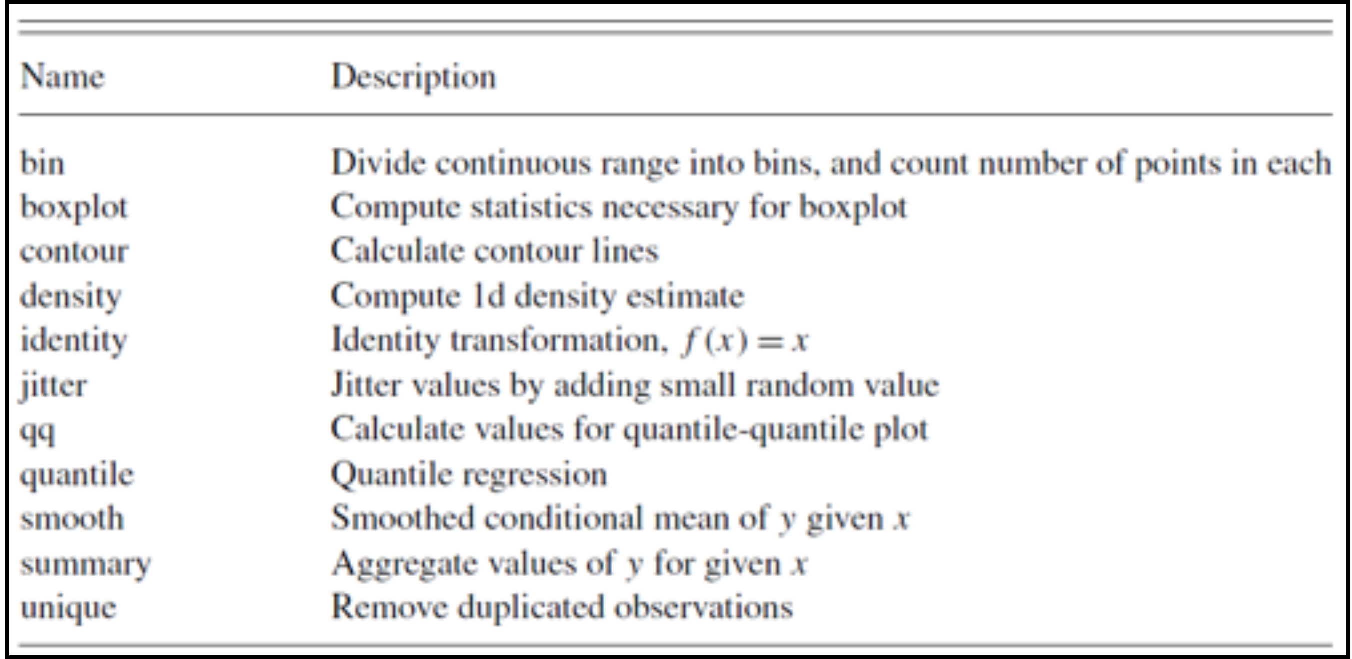


LAYERED GRAMMAR OF GRAPHICS: DATA, STATS, GEOMS

- Data adds concreteness to abstract graphics
- Need to specify which variables need to be mapped to which aesthetics
- Geometric objects, or **geom**s, control the type of plot created
- Every geom is characterized by a default statistic, and every statistic is assigned to a geom.

Data are obviously a critical part of the plot, but it is important to remember that they are independent from the other components: we can construct a graphic that can be applied to multiple datasets.
Data are what turns an abstract graphic into a concrete graphic.
Along with the data, we need a specification of which variables are mapped to which aesthetics.
To make sense in a graphical context a stat must be location-scale invariant: $f(x+a) = f(x)+a$ and $f(b \cdot x) = b \cdot f(x)$.
This ensures that the transformation is invariant under translation and scaling, common operations on a graphic
Geometric objects, or **geom**s for short, control the type of plot that you create. For example, using a point geom will create a scatterplot, whereas using a line geom will create a line plot. Every geom has a default statistic, and every statistic a default geom.

STATISTICAL TRANSFORMATIONS IN LAYERED GRAMMAR OF GRAPHICS



Name	Description
bin	Divide continuous range into bins, and count number of points in each
boxplot	Compute statistics necessary for boxplot
contour	Calculate contour lines
density	Compute 1d density estimate
identity	Identity transformation, $f(x) = x$
jitter	Jitter values by adding small random value
qq	Calculate values for quantile-quantile plot
quantile	Quantile regression
smooth	Smoothed conditional mean of y given x
summary	Aggregate values of y for given x
unique	Remove duplicated observations

Some statistical transformations provided by ggplot2. The user is able to supplement this list in a straightforward manner.

many statistical operations have not been derived for non-Cartesian coordinates
and so we use Cartesian coordinates for calculation, which, while not strictly correct, will
normally be a fairly close approximation.

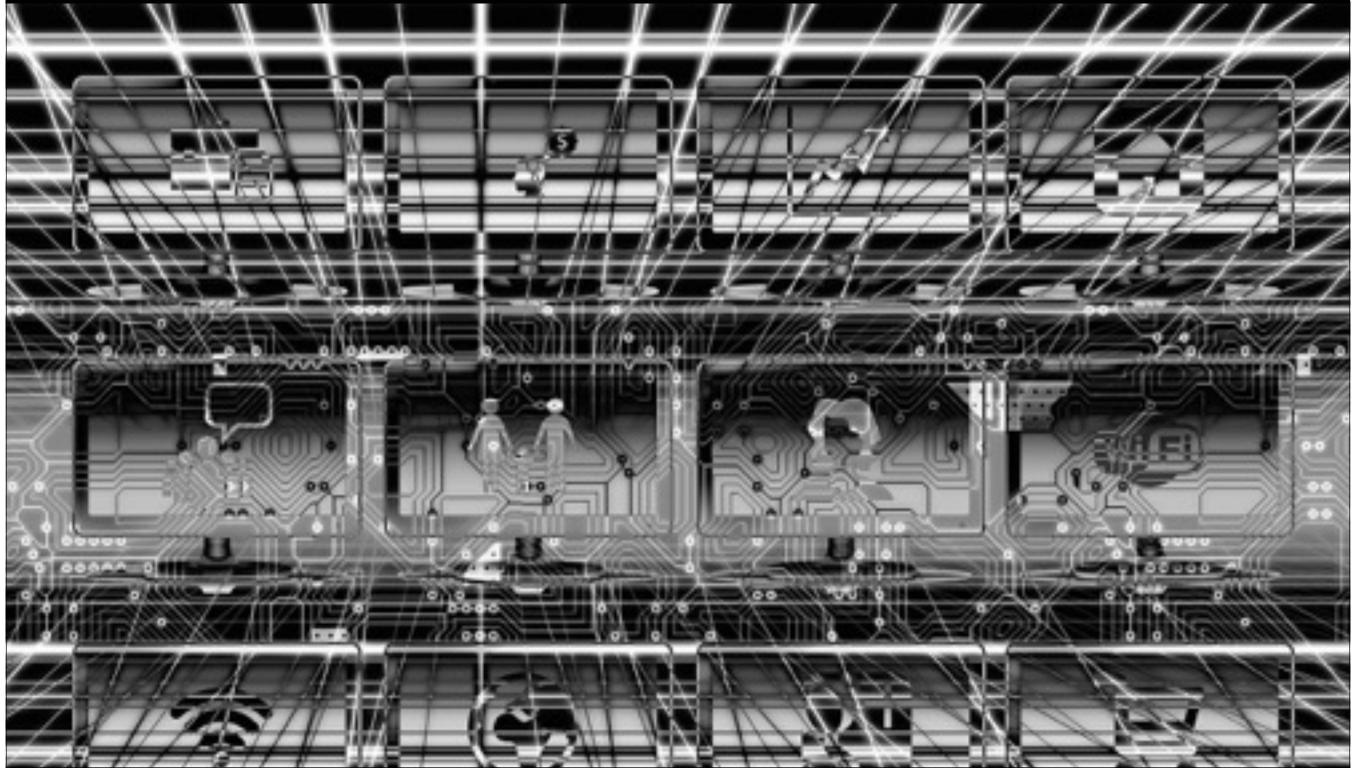
LAYERED GRAMMAR OF GRAPHICS: POSITION, SCALES, COORD, FACETS

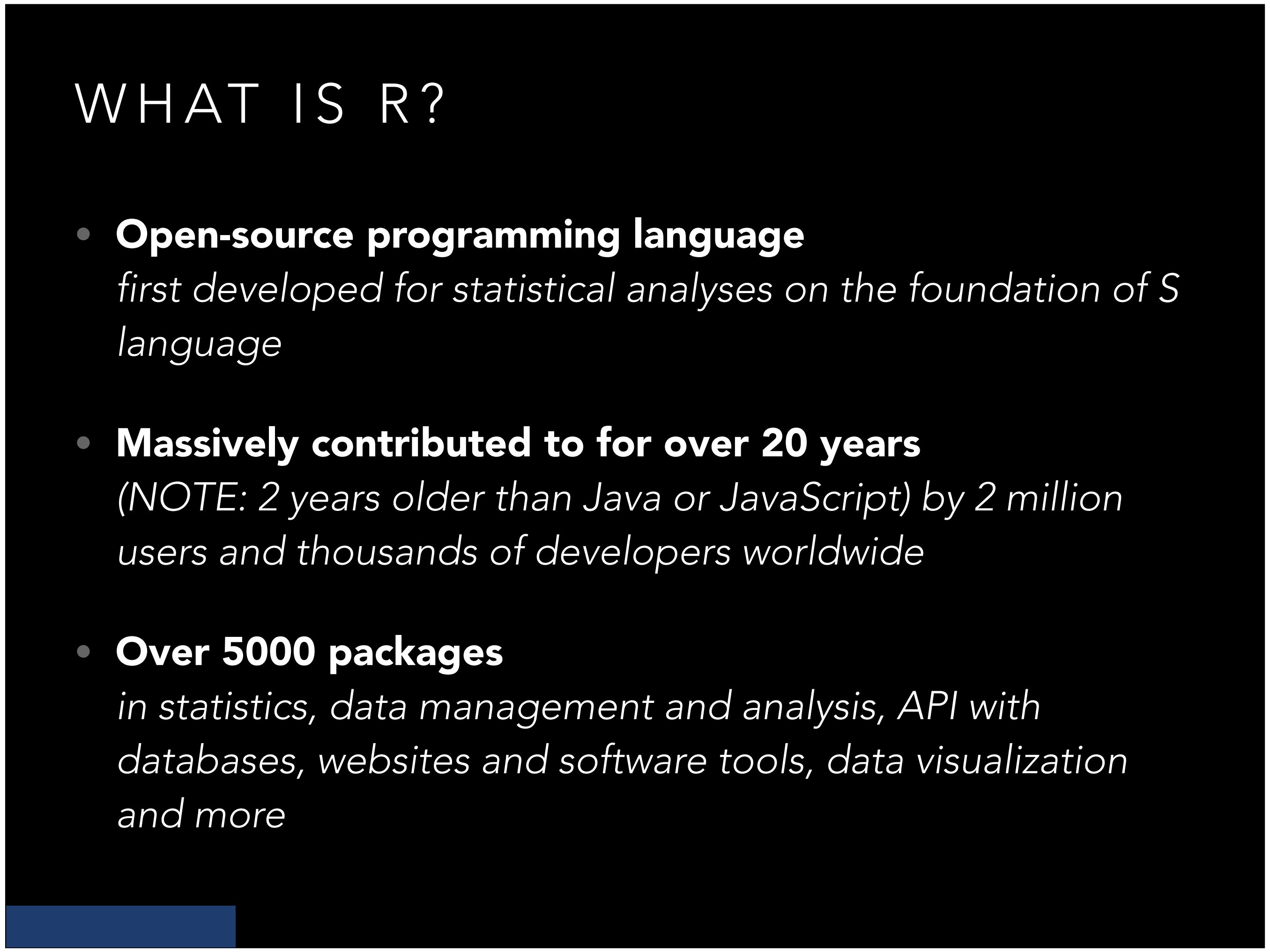
- Position adjustment serves purpose of visibility
- A **scale** controls the mapping from data to aesthetic attributes, which necessitates using a scale for each aesthetic property inside a layer
- **Coord** (coordinate system) positions the objects onto the plane of the plot
- Faceting split up the data into several planes based on a specified variable, while arranging them in a specified manner

Sometimes we need to tweak the position of the geometric elements on the plot, when
otherwise they would obscure each other. A **scale** controls the mapping from data to aesthetic attributes, and so we need one scale for each aesthetic property used in a layer.
Scales typically map from a single variable to a single aesthetic, but there are exceptions.
For example, we can map one variable to hue and another to saturation, to create a single
aesthetic, color.
A coordinate system, **coord** for short, maps the position of objects onto the plane of
the plot. The faceting specification describes which
variables should be used to split up the data, and how they should be arranged.

(TECHNICALLY) PRESENTING DATA

PART (3)





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

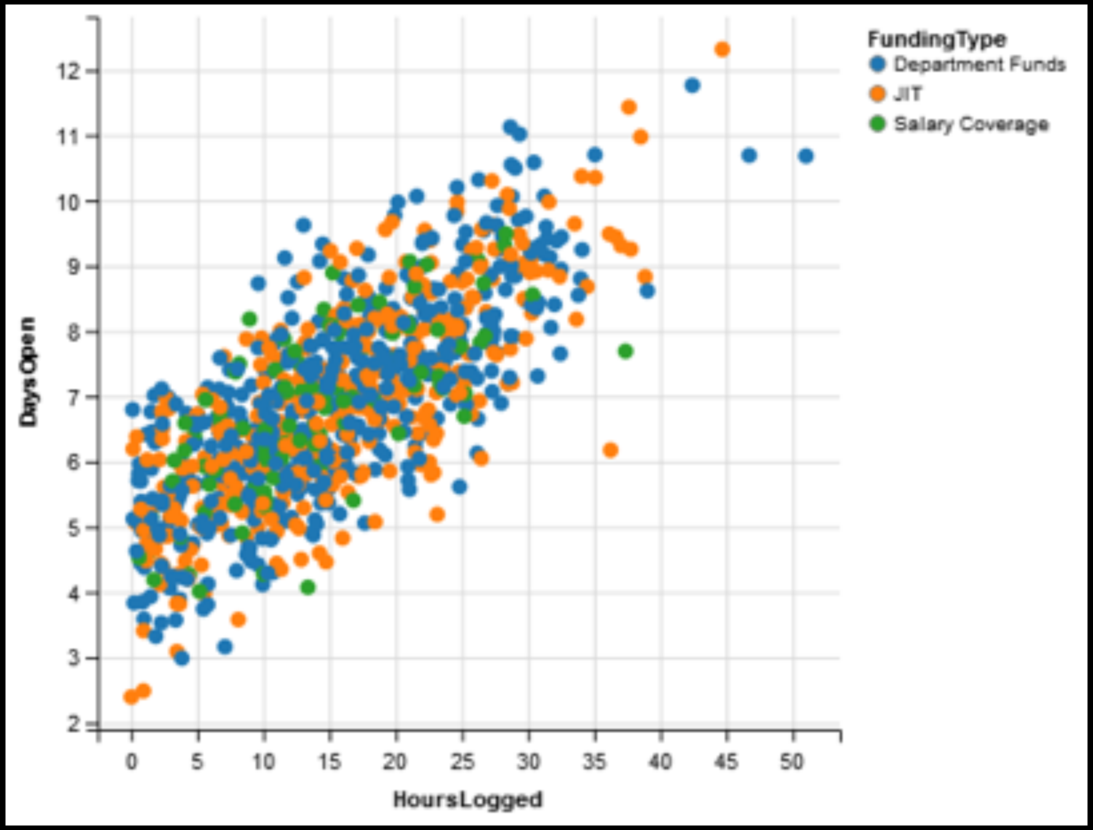
GGVIS

Combines

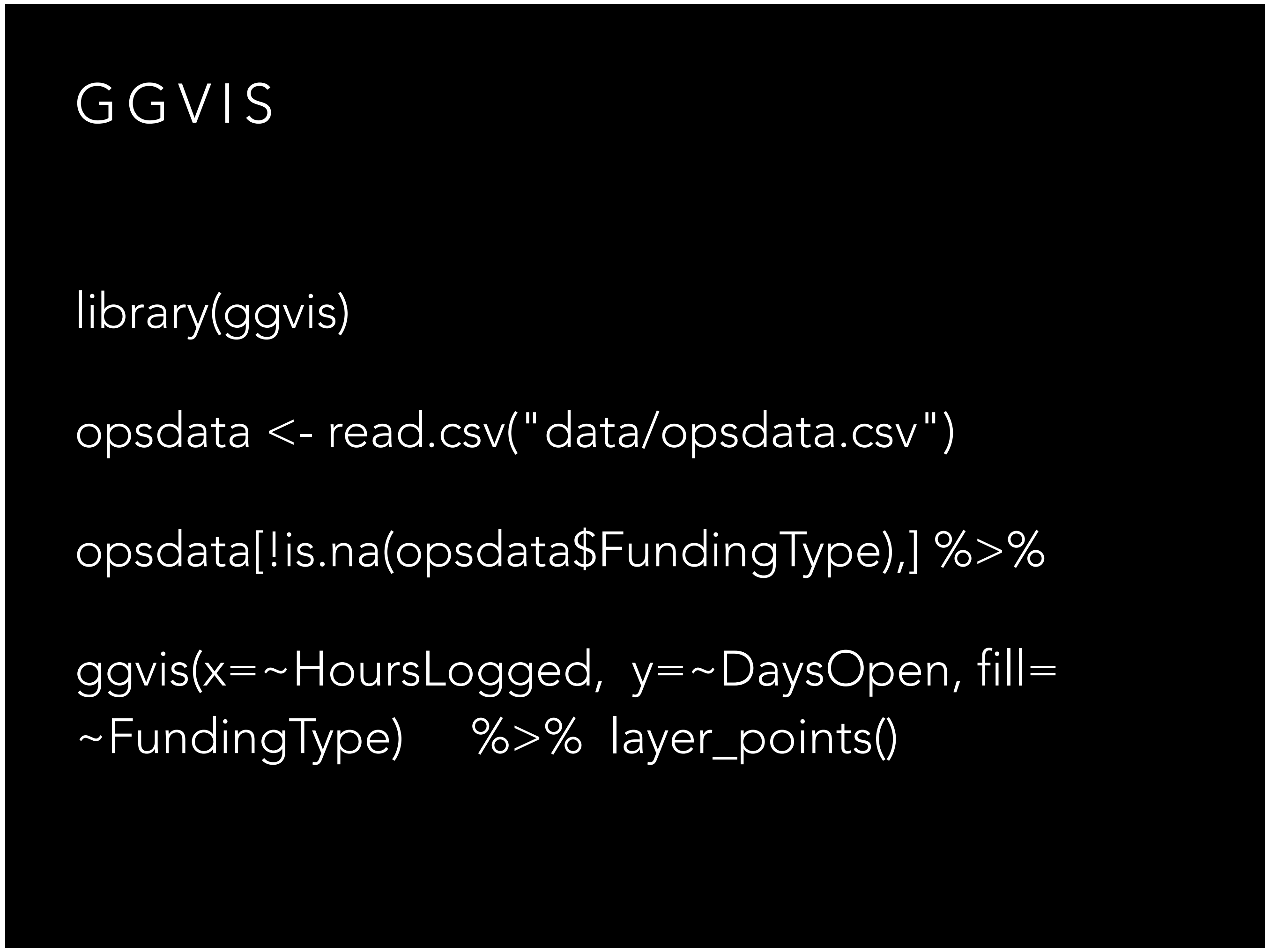
- grammar of graphics
- reactivity of Shiny
- pipeline of dplyr

Great for discovering data characteristics and patterns

GGVIS



FundingType
● Department Funds
● JIT
● Salary Coverage
DaysOpen
2
3
4
5
6
7
8
9
10
11
12
HoursLogged
0
5
10
15
20
25
30
35
40
45
50



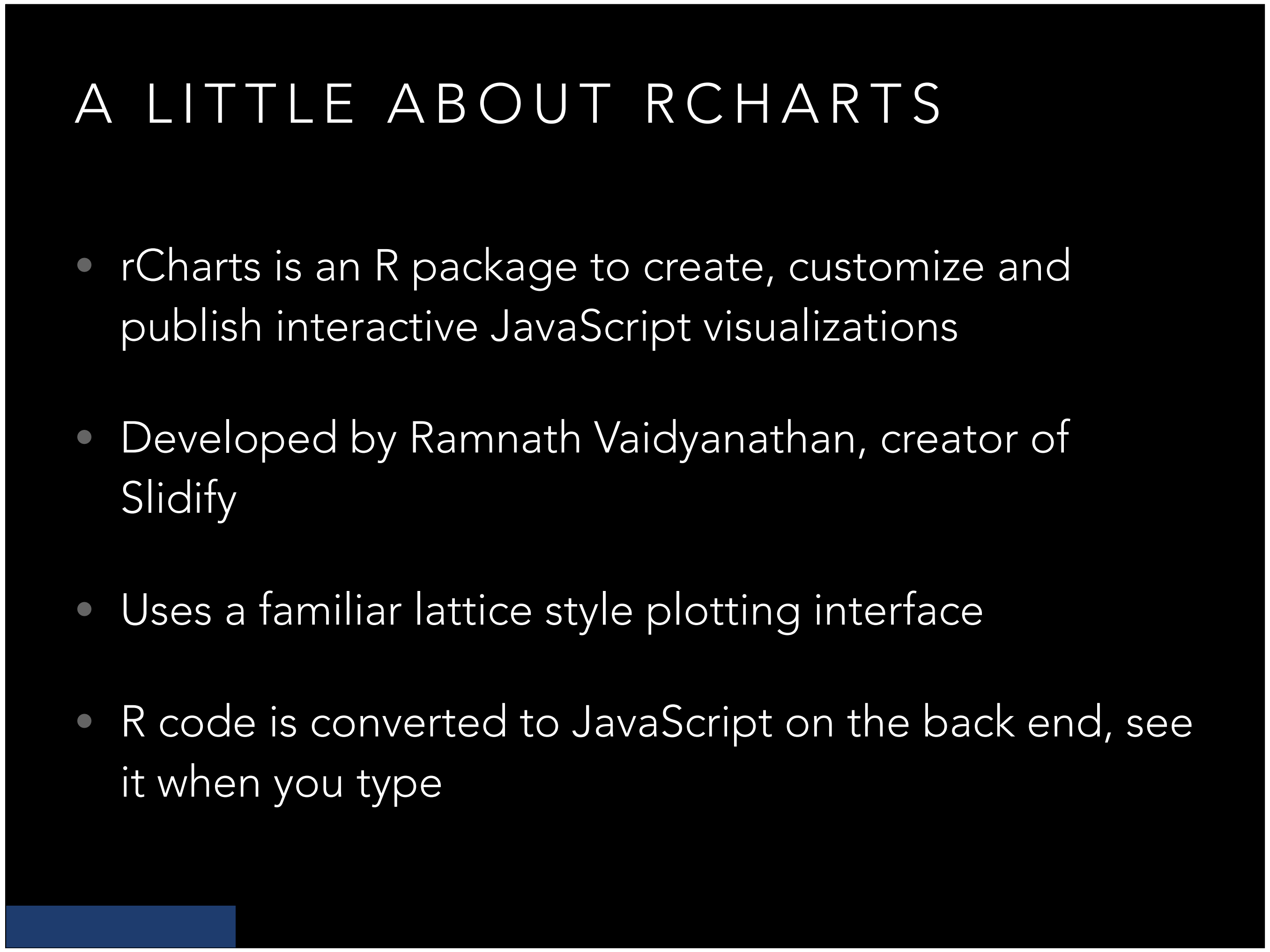
GGVIS

```
library(ggvis)

opsdata <- read.csv("data/opsdata.csv")

opsdata[!is.na(opsdata$FundingType),] %>%

ggvis(x=~HoursLogged,  y=~DaysOpen, fill=
~FundingType)     %>%  layer_points()
```

GGVIS: REACTIVITY

```
# using filters and widgets:
opsdata[!is.na(opsdata$FundingType),] %>%
ggvis(
    x = input_select(c('HoursLogged', 'DaysOpen'),map =
as.name, label = "Choice Variable"),
    y = ~DaysOpen,
    fill = input_select(c("FundingType", "Application"),
map = as.name, label = "Fill Variable")
  ) %>%
  layer_points()
```

RCHARTS



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

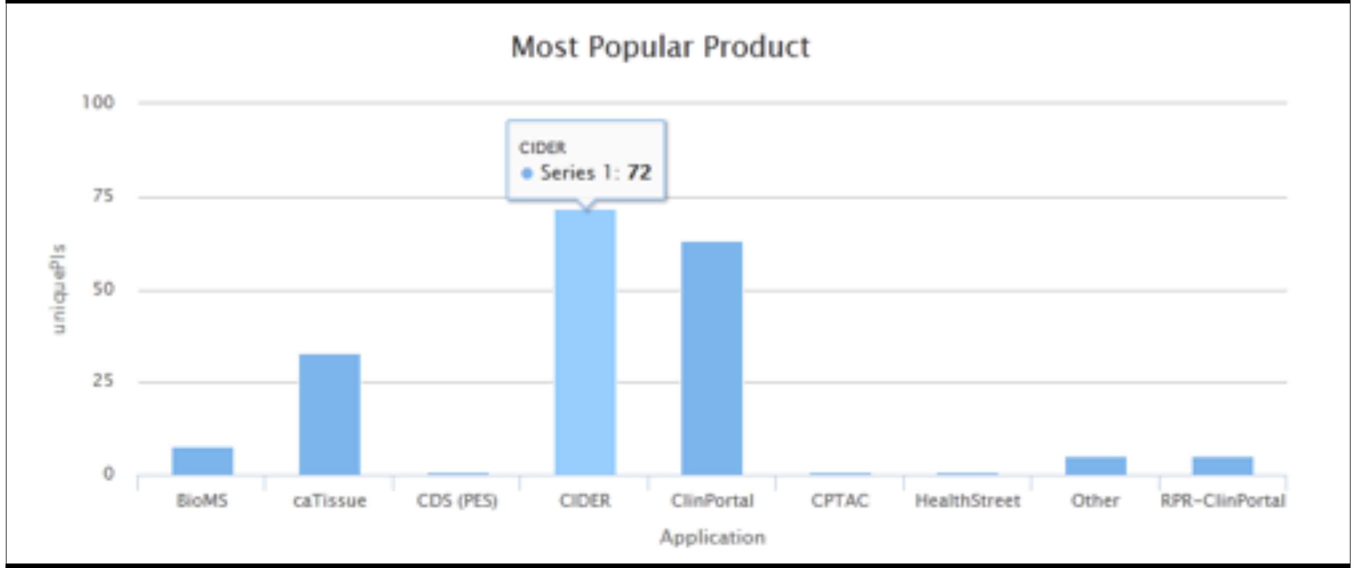
Unlike many fundamental visualization packages in R (base, ggplot2, ggvis), rCharts gives you a much higher level of precision through its interactivity – you hover over a point of data and it tells you whatever you want it to tell you (for the most part).

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)

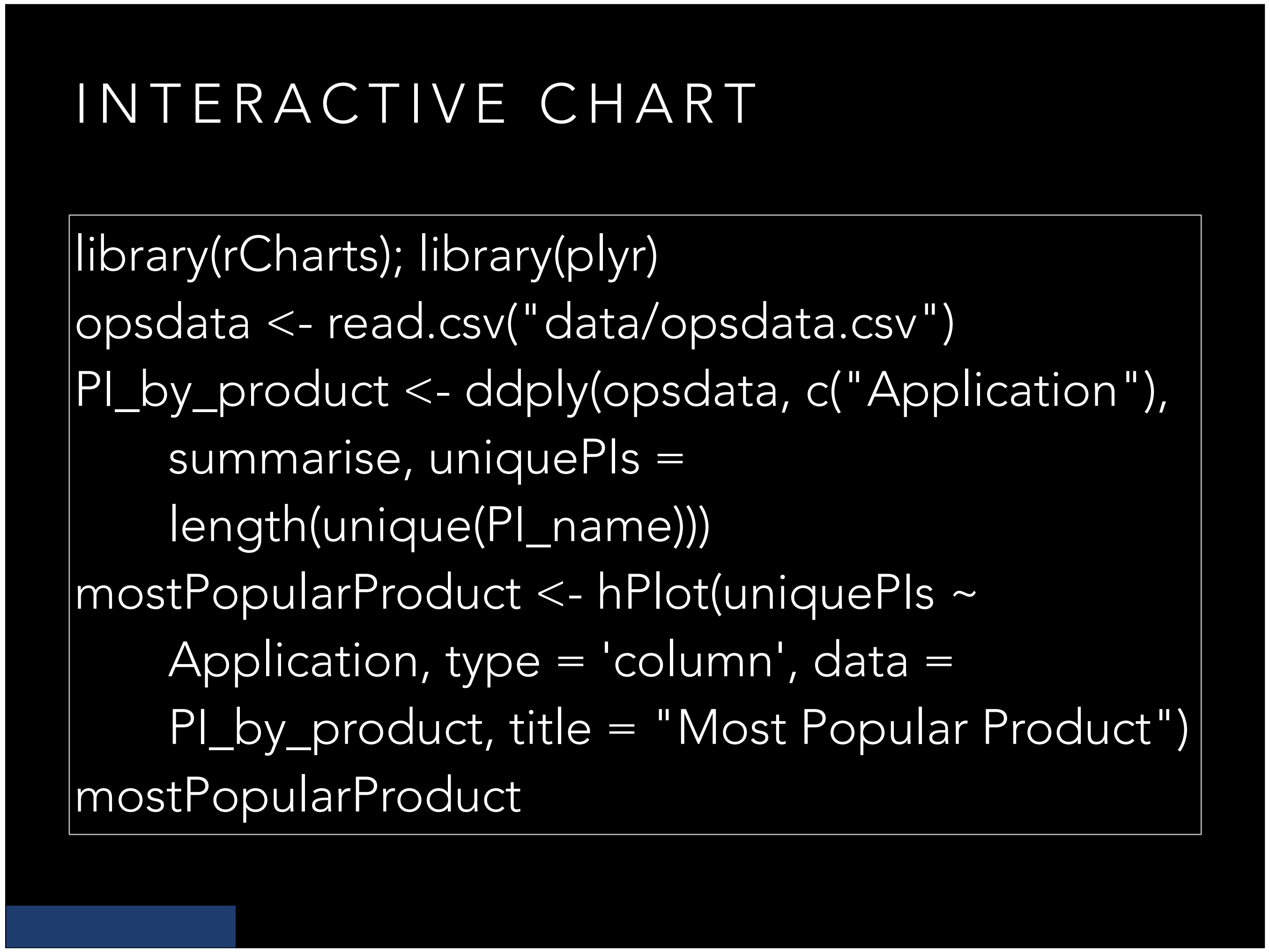
```
require(devtools)
install_github('rCharts', 'ramnathv')
```

INTERACTIVE CHART



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).



INTERACTIVE CHART

```
library(rCharts); library(plyr)
opsdata <- read.csv("data/opsdata.csv")
PI_by_product <- ddply(opsdata, c("Application"),
      summarise, uniquePIs =
      length(unique(PI_name)))
mostPopularProduct <- hPlot(uniquePIs ~
      Application, type = 'column', data =
      PI_by_product, title = "Most Popular 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).

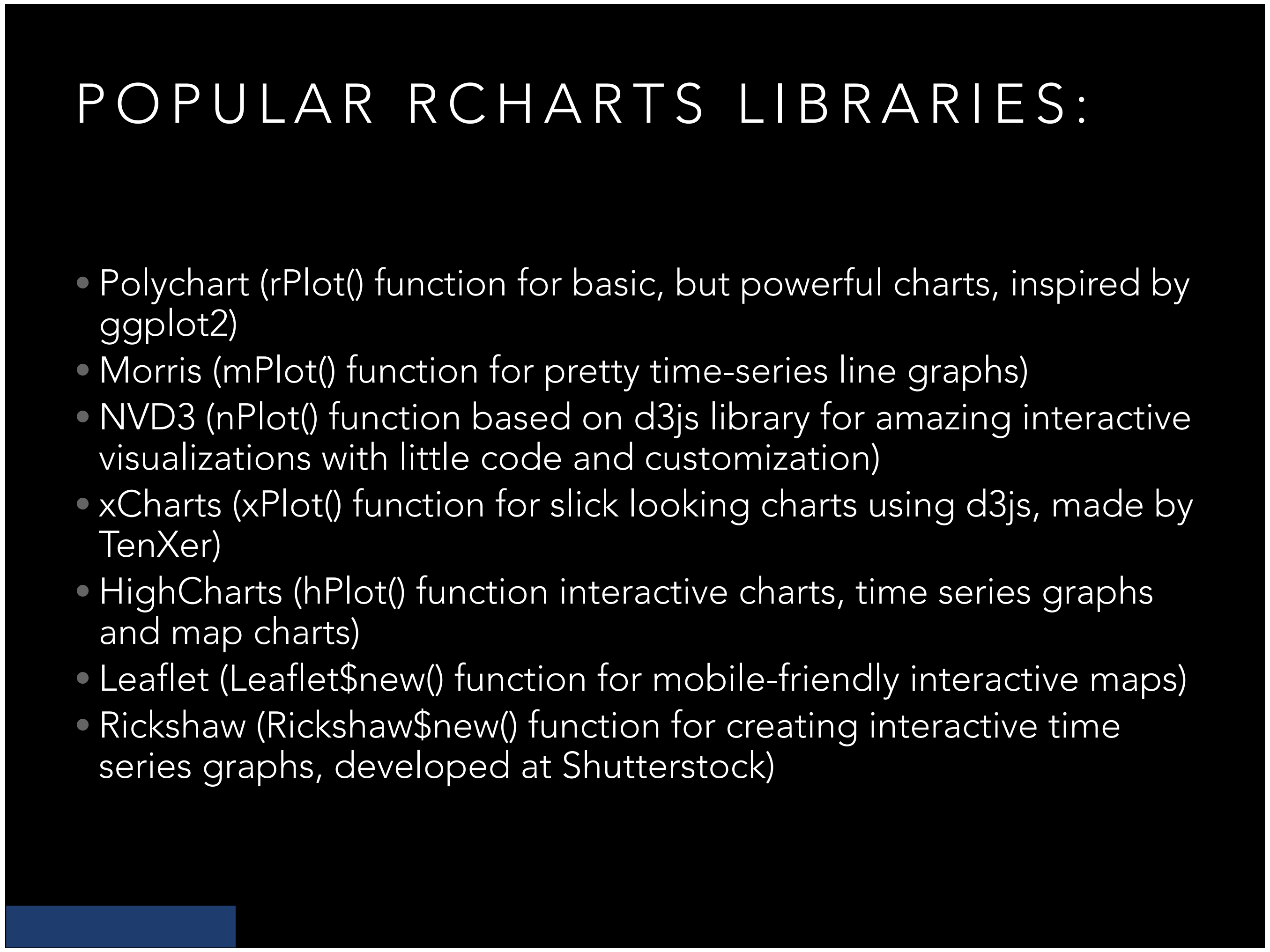
DATA FOR THE INTERACTIVE CHART

Preview of part of opsdata

ID	CREATED	STATUS	TITLE	PI_NA	FUNDI	APPLIC
1	7/24/2014 11:32	CLOSED	PROJECT 44	ELVIS DANIELS	JIT	BIOMS
2	8/15/2014 9:52	CLOSED	PROJECT 330	SCOTT LOTT	JIT	BIOMS
3	9/4/2014 12:30	CLOSED	PROJECT 470	CASSIDY GRIFFITH	JIT	BIOMS
4	3/12/2014 10:51	CLOSED	PROJECT 477	MOLLIE CHANG	JIT	BIOMS

Preview of PI_by_product

APPLICATION	UNIQUEPIS
BIOMS	8
CATISSUE	33
CDS (PES)	1
CIDER	72

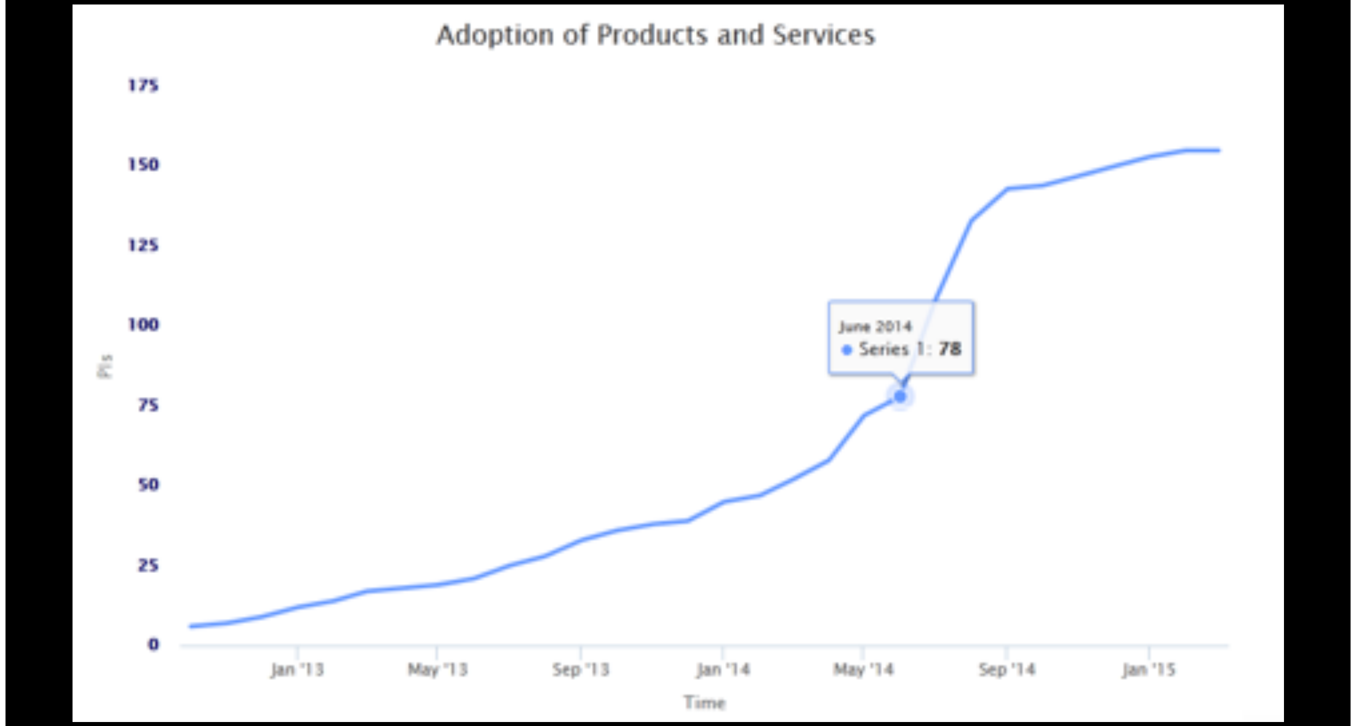


POPULAR RCHARTS LIBRARIES:

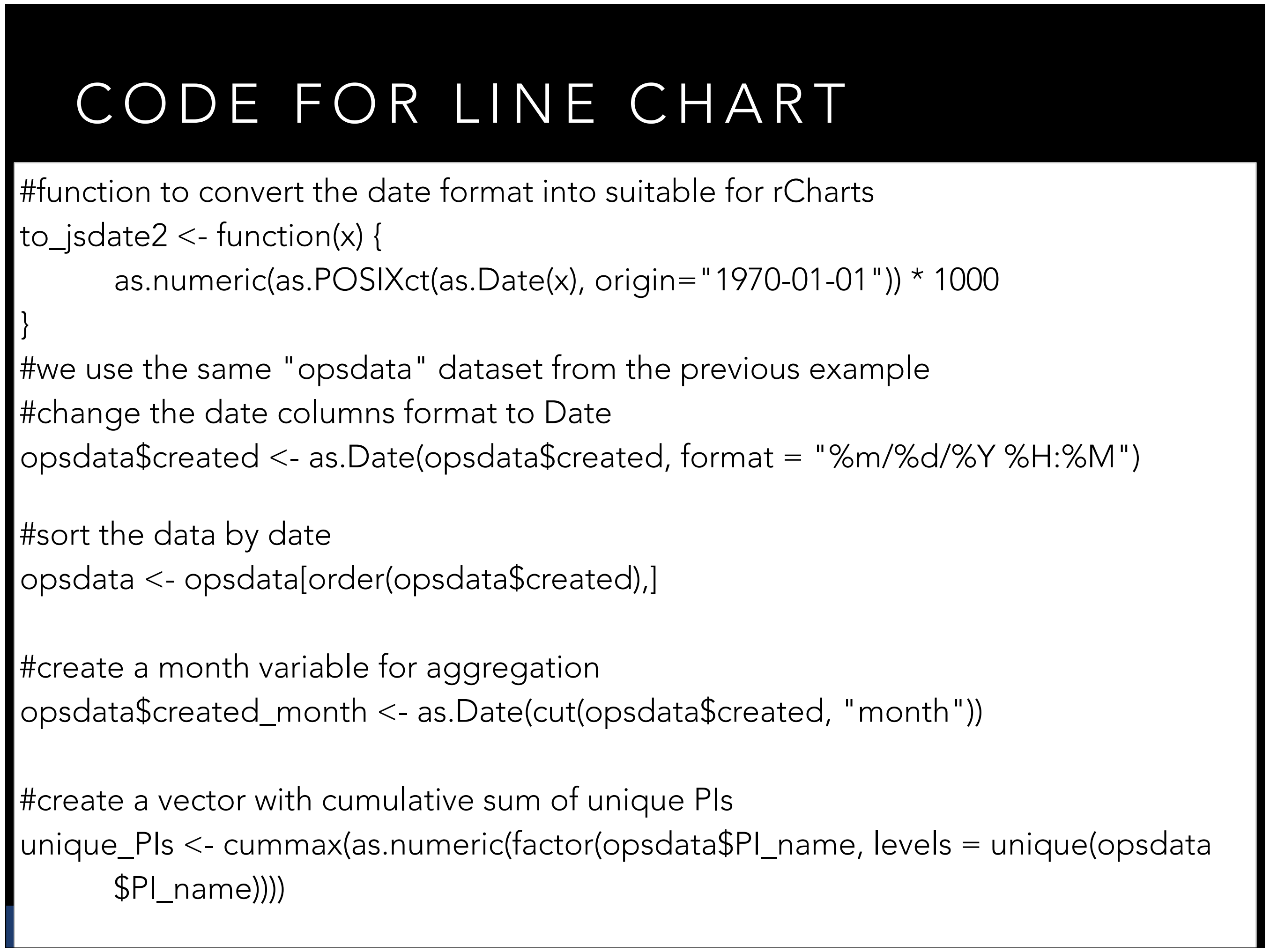
- Polychart (rPlot() function for basic, but powerful charts, inspired by ggplot2)
- Morris (mPlot() function for pretty time-series line graphs)
- NVD3 (nPlot() function based on d3js library for amazing interactive visualizations with little code and customization)
- xCharts (xPlot() function for slick 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



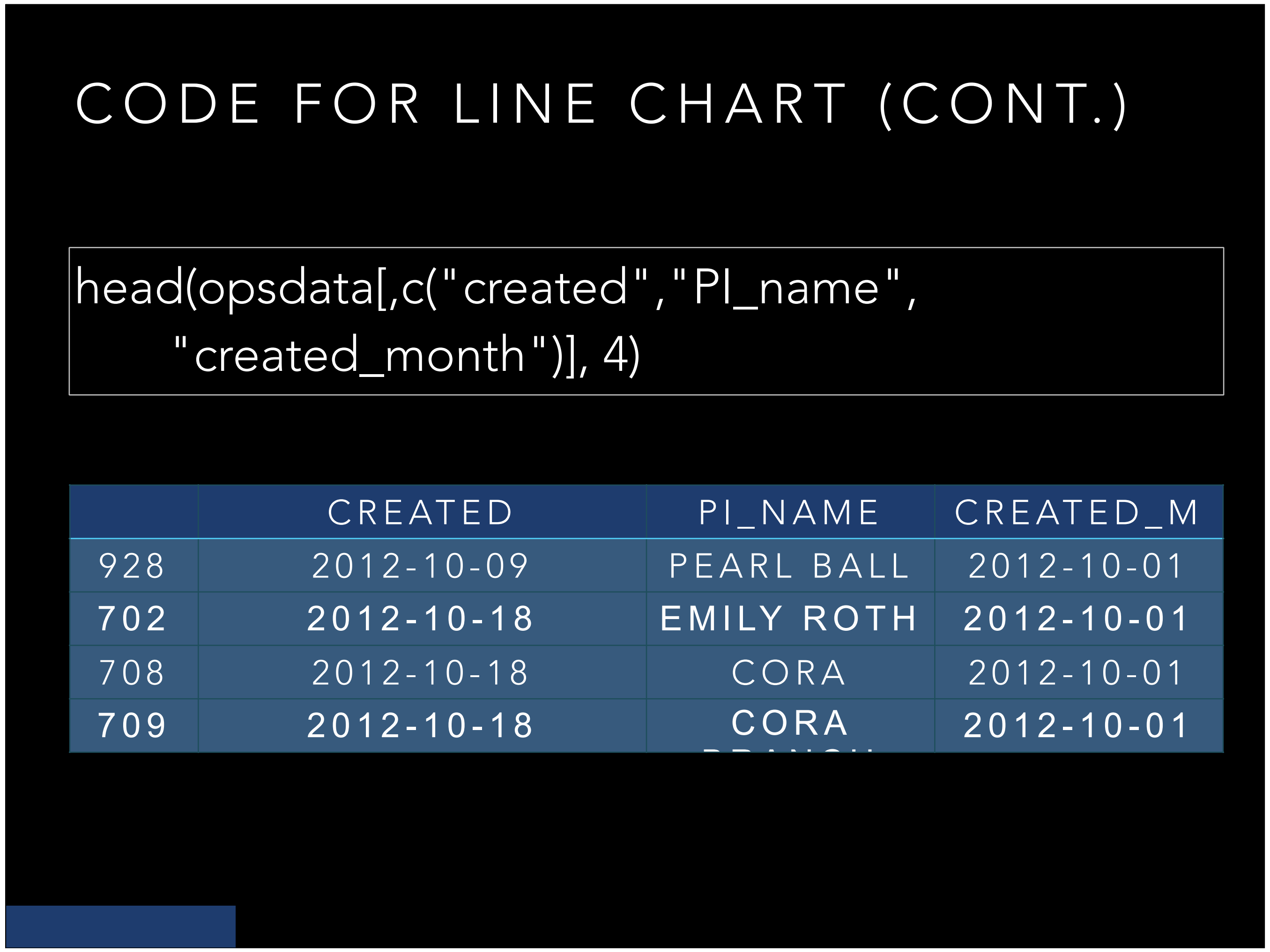
This example is a little more complex than the first one, because it involves a line chart and a lot of customization, which we will walk you through. First is a time series chart that shows cumulative growth of PIs that have worked CBMI's products and services over the last several years.



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))))
```

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.

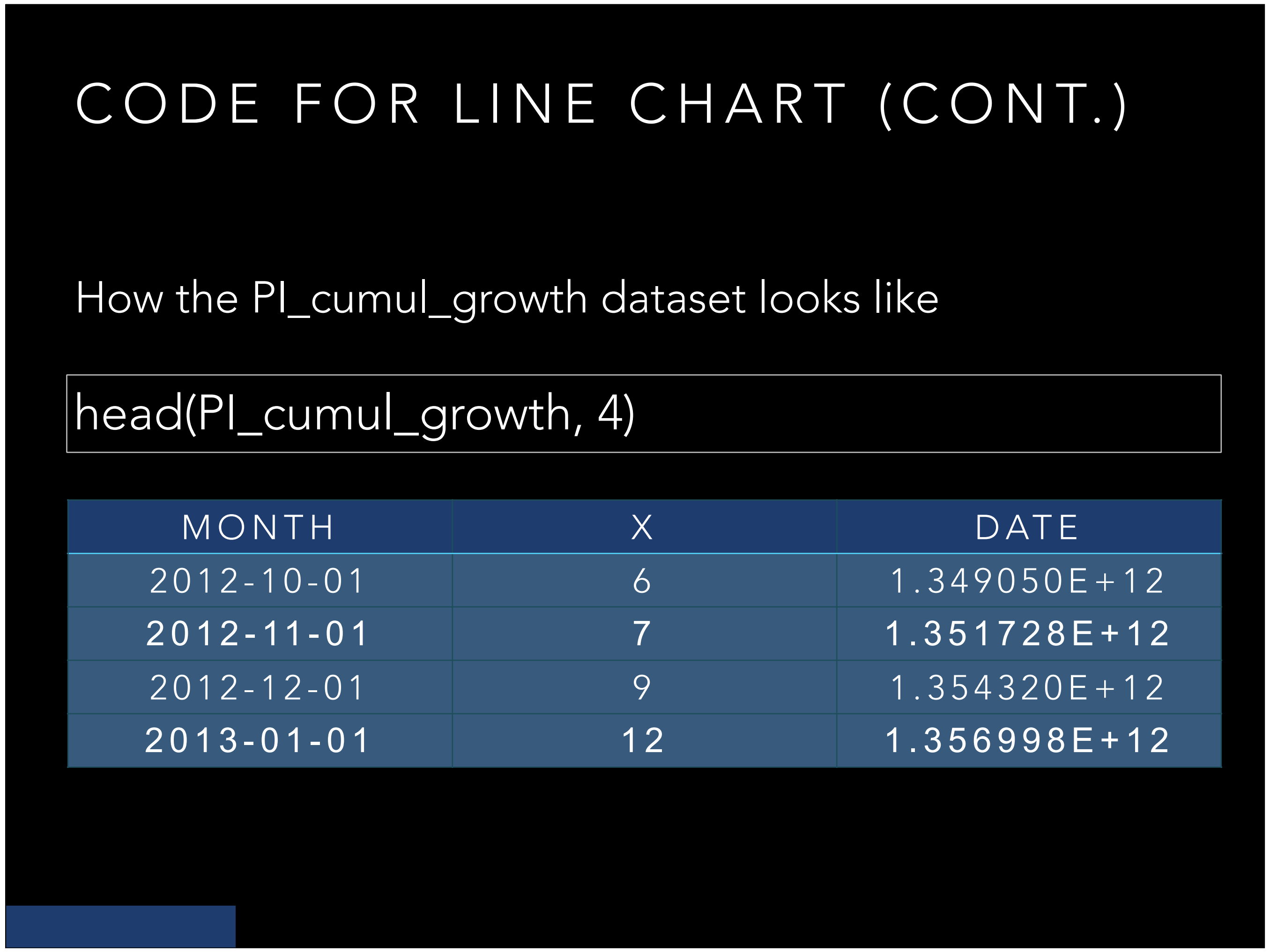


CODE FOR LINE CHART (CONT.)

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

	CREATED	PI_NAME	CREATED_M
928	2012-10-09	PEARL BALL	2012-10-01
702	2012-10-18	EMILY ROTH	2012-10-01
708	2012-10-18	CORA	2012-10-01
709	2012-10-18	CORA	2012-10-01

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).



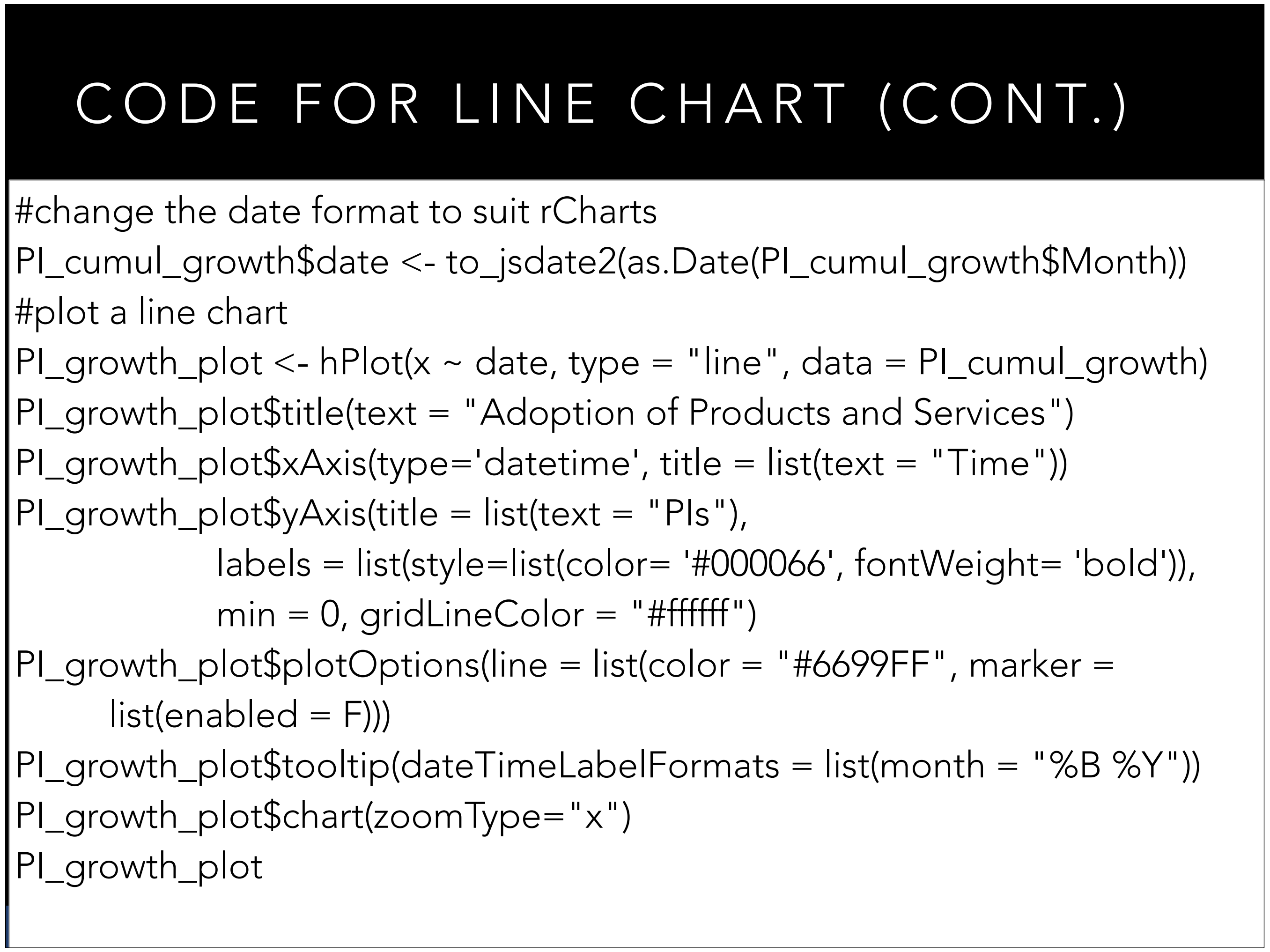
CODE FOR LINE CHART (CONT.)

How the PI_cumul_growth dataset looks like

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

MONTH	X	DATE
2012-10-01	6	1.349050E+12
2012-11-01	7	1.351728E+12
2012-12-01	9	1.354320E+12
2013-01-01	12	1.356998E+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.)

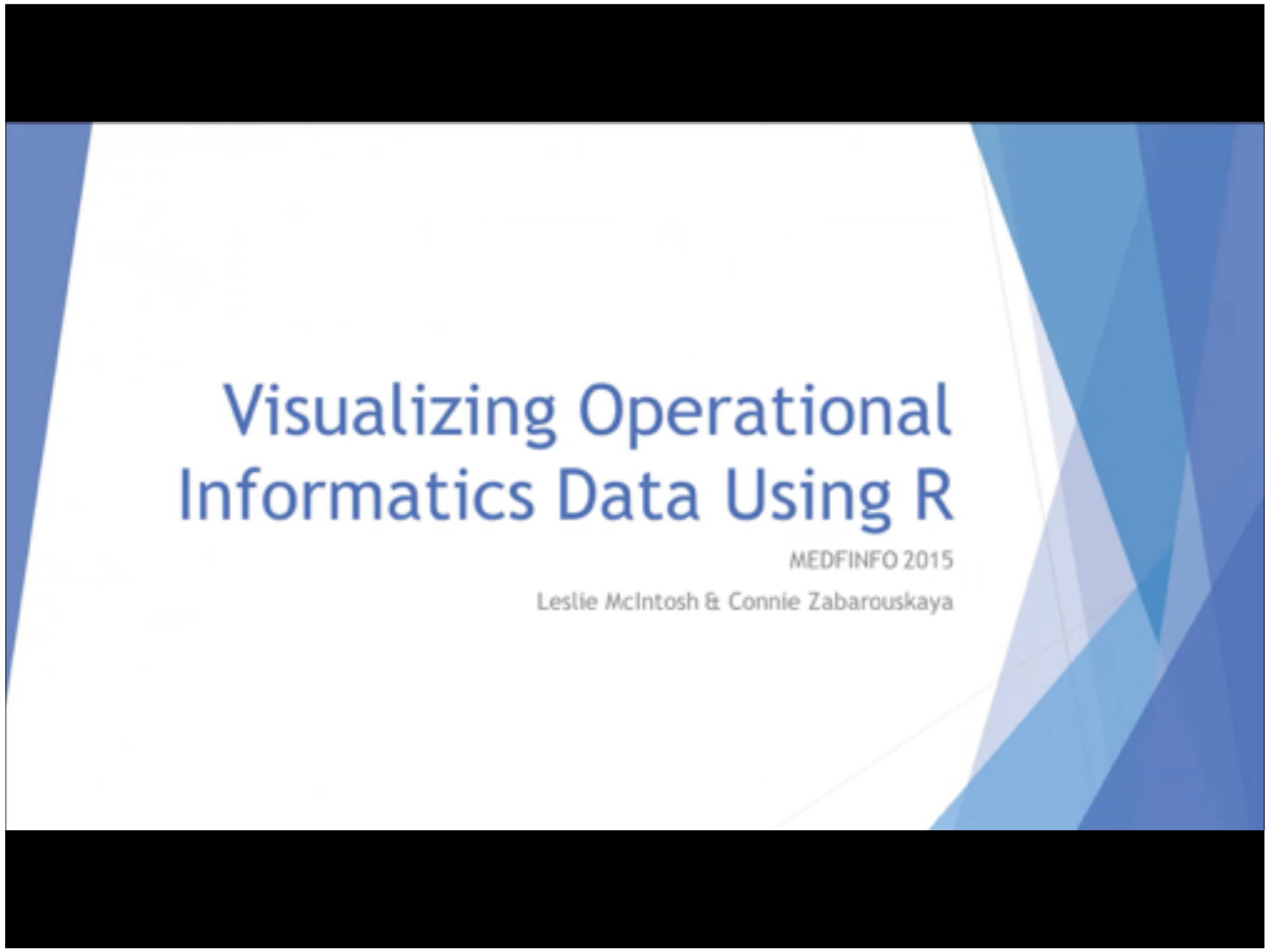
```
#change the date format to suit rCharts
PI_cumul_growth$date <- to_jsdate2(as.Date(PI_cumul_growth$Month))
#plot a line chart
PI_growth_plot <- hPlot(x ~ date, type = "line", data = PI_cumul_growth)
PI_growth_plot$title(text = "Adoption of Products and Services")
PI_growth_plot$xAxis(type='datetime', 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
```

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.

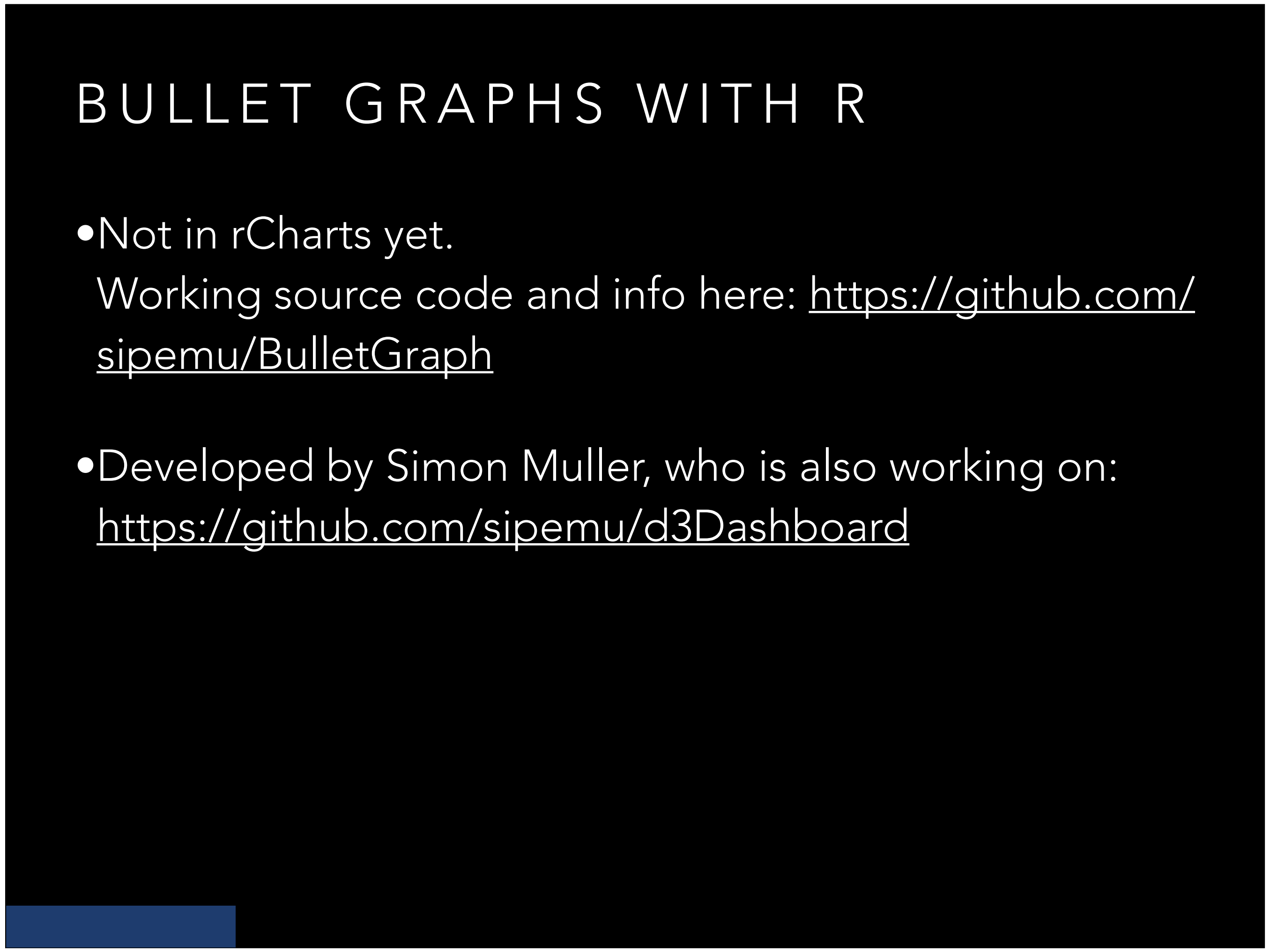
HANDS-ON RCHARTS CODING



Visualizing Operational Informatics Data Using R

MEDFINFO 2015

Leslie McIntosh & Connie Zabarouskaya



BULLET GRAPHS WITH R

- Not in rCharts yet.
  Working source code and info here: https://github.com/sipemu/BulletGraph

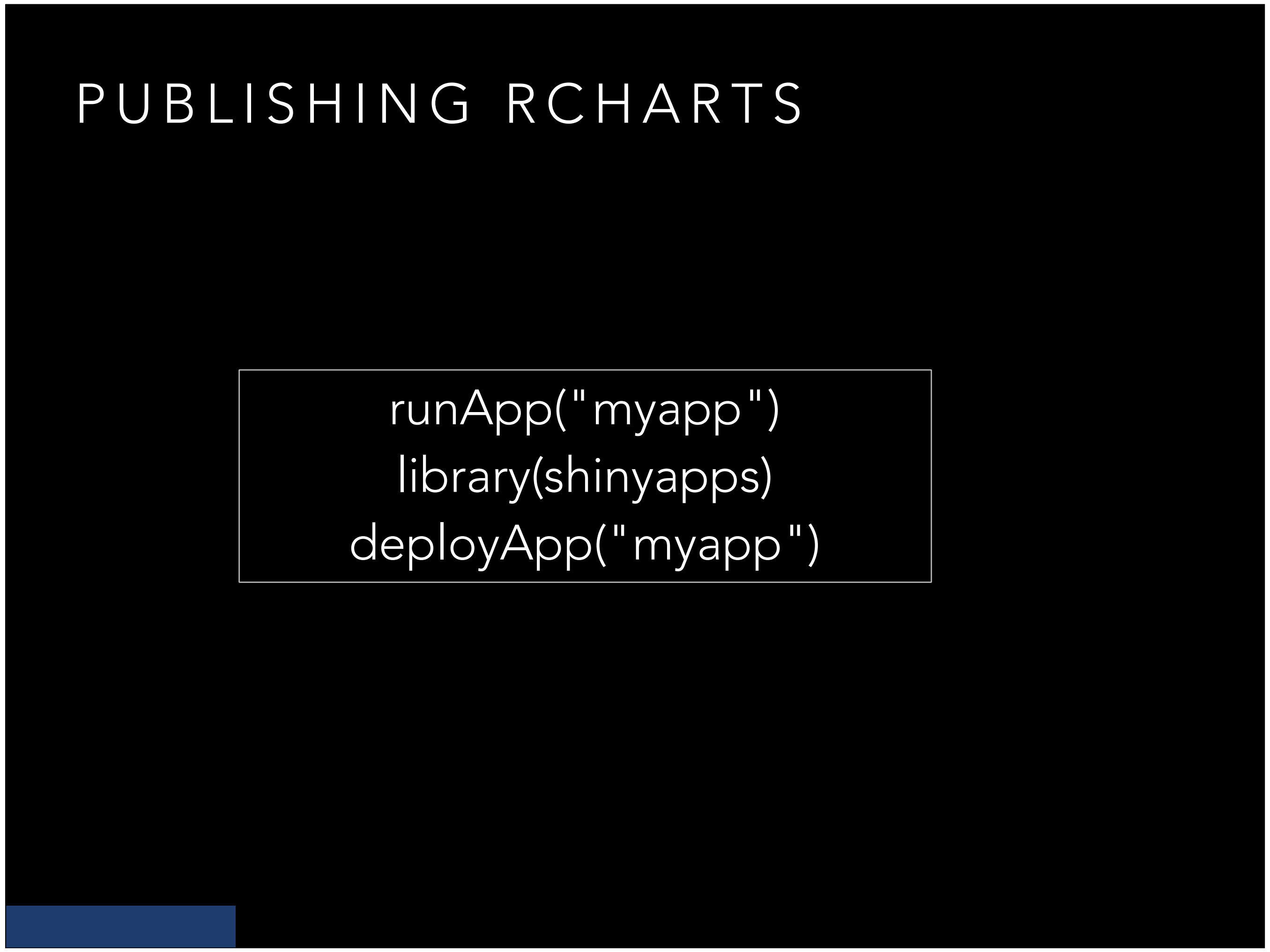
- Developed by Simon Muller, who is also working on: https://github.com/sipemu/d3Dashboard

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

```
#load the source code into memory to make use of the functions
source("helpers/BulletGraphSipemu.txt")

#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)")
```

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.



PUBLISHING RCHARTS

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

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.

GOOGLEVIS

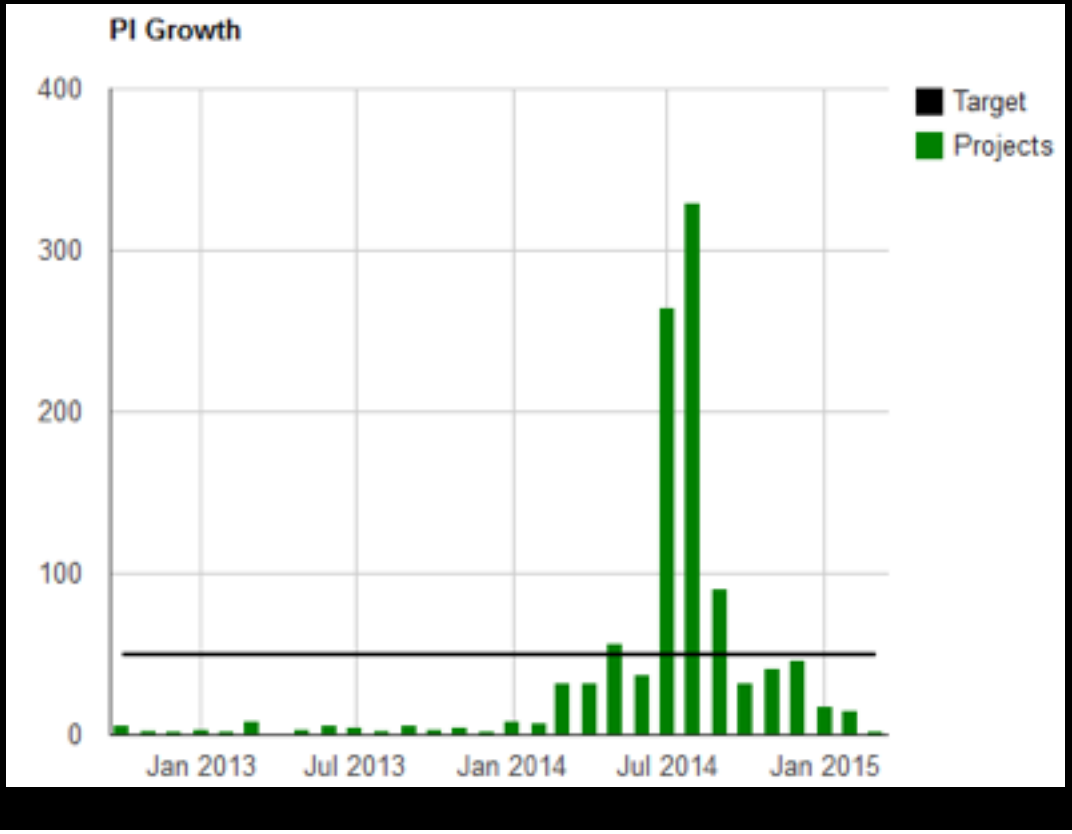
A WORD ABOUT GOOGLEVIS

- All charts require an Internet connection.
- The R function creates an HTML page
- The HTML page calls Google Charts
- The result is an interactive HTML graphic
- GPL-2 license

Some Common Types of Charts:

- Motion charts: gvisMotionChart
- Interactive maps: gvisGeoChart
- Interactive tables: gvisTable
- Line charts: gvisLineChart
- Bar charts: gvisColumnChart
- Tree maps: gvisTreeMap

GOOGLEVIS EXAMPLE



GOOGLEVIS EXAMPLE

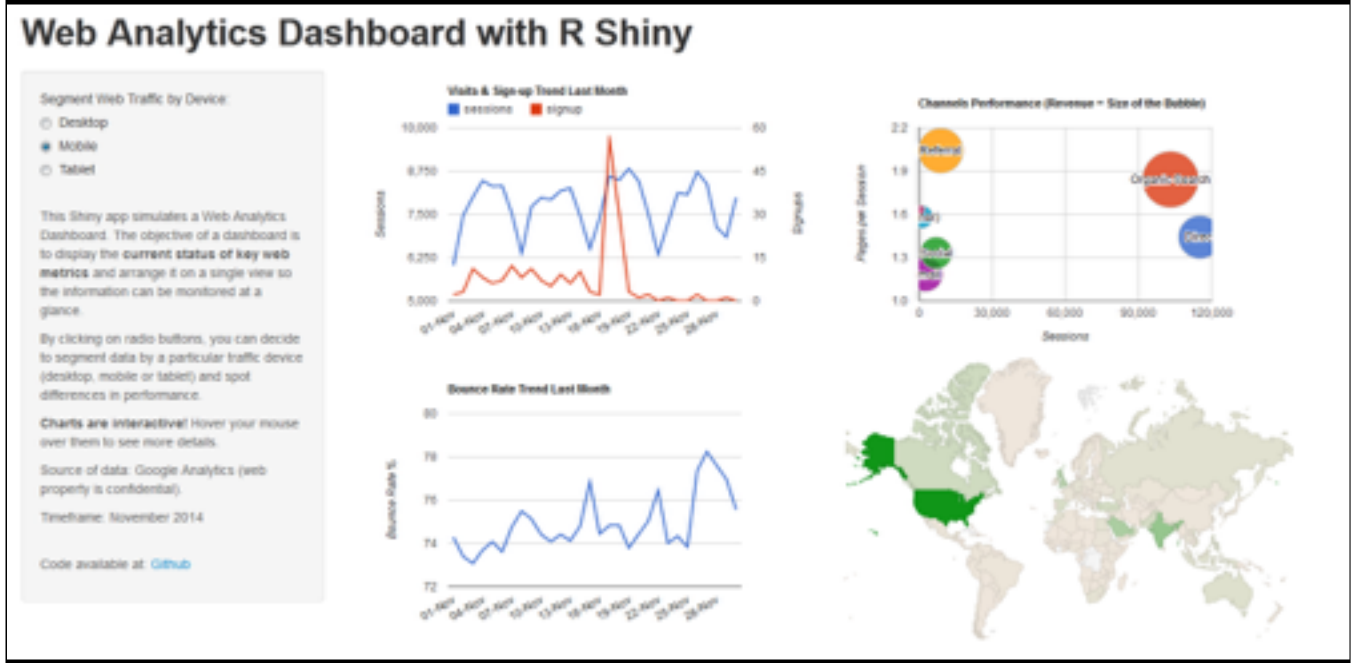
```
# Example of a googleVis combo chart
library(googleVis)
# aggregate number of projects created by month
projectsByMonth <- aggregate(opsdata$title, list(Month=opsdata$created_month), length)
names(projectsByMonth) <- c("Month", "Projects")
projectsByMonth$Target <- 50
comboChart <- gvisComboChart(projectsByMonth,
          xvar='Month',
            yvar=c('Target', 'Projects'),
            options=list(seriesType='bars',
                    width=600, height=500,
                    series="[{type:'line',
                        color:'black'},
                        {type:'bars',
                        color:'green'}]",
                    title='PI Growth'))
plot(comboChart)
```

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

DASHBOARD EXAMPLES



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.

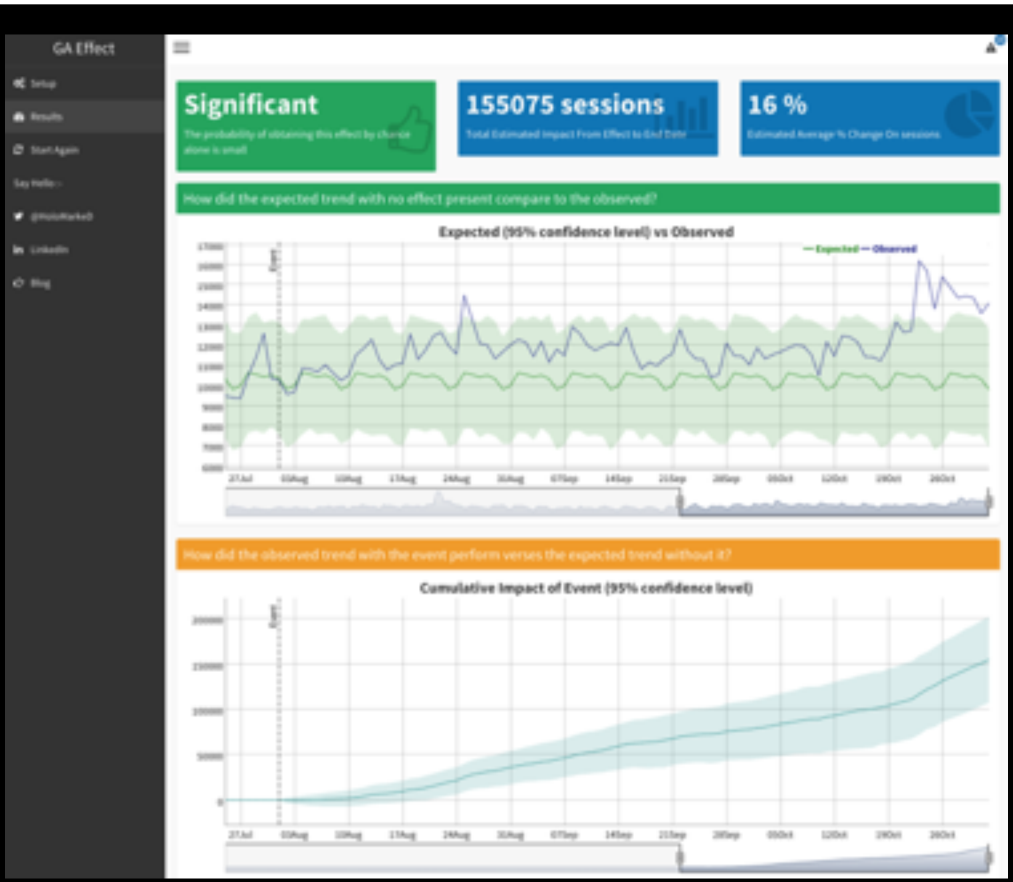
Charts are interactive! Hover your mouse over them to see more details.

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

Timeframe: November 2014

Code available at: Github

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



Significant

The probability of obtaining this effect by chance alone is small

155075 sessions

Total Estimated Impact From Effect to End Date

16 %

Estimated Average % Change On sessions

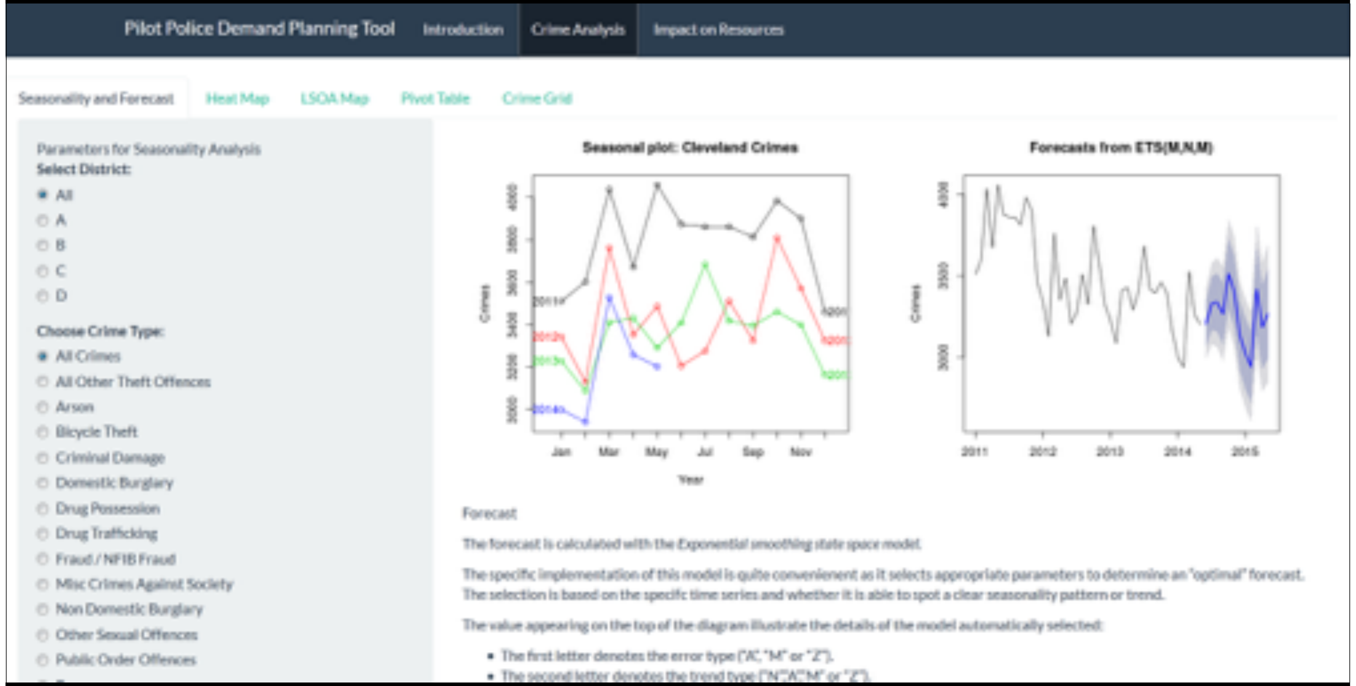
How did the expected trend with no effect present compare to the observed?

Expected (95% confidence level) vs Observed

How did the observed trend with the event perform verses 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



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.*

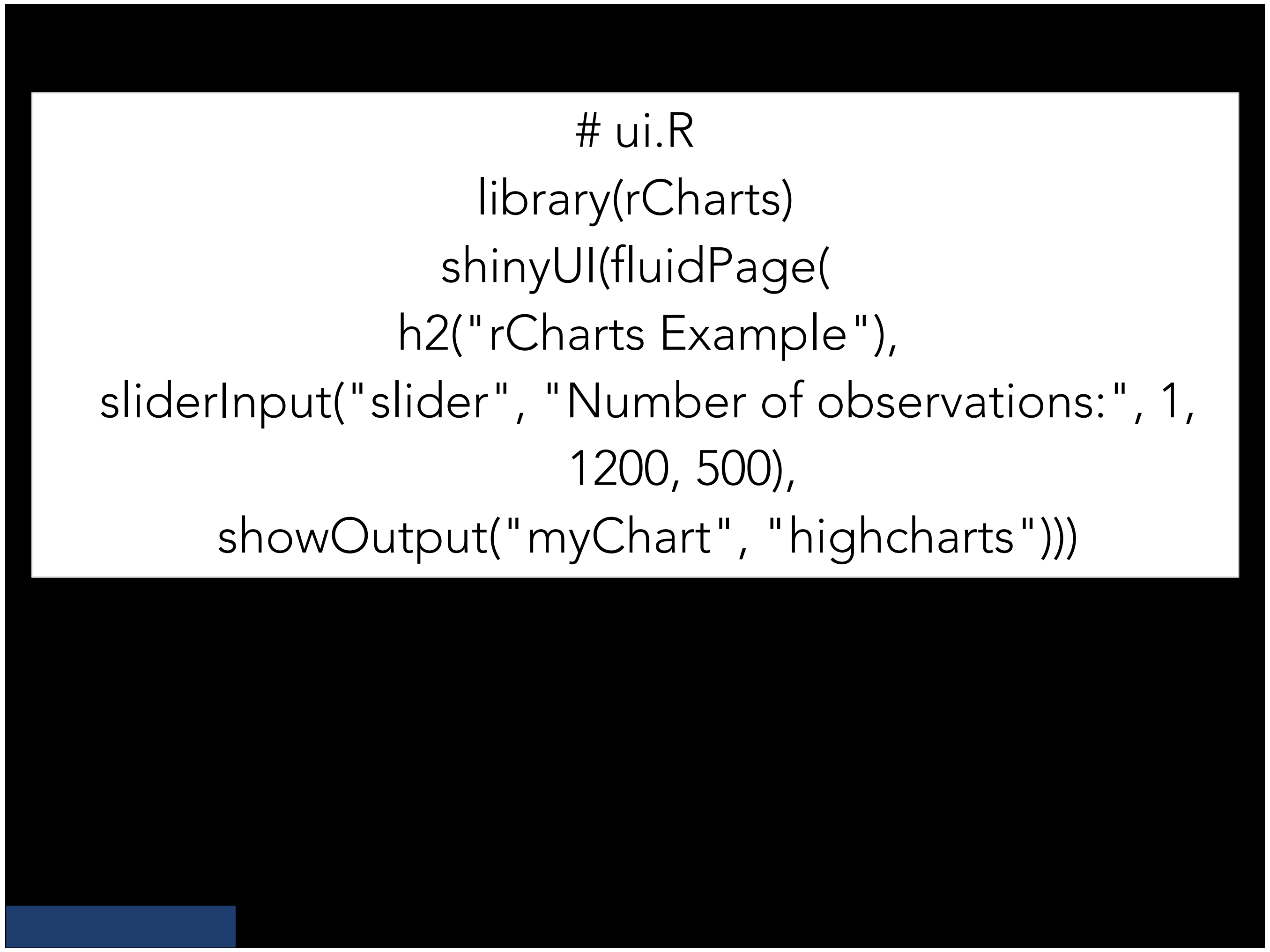
The specific implementation of this model is quite convenienent as it selects appropriate parameters to determine an "optimal" forecast. The selection is based on the specifc time series and whether it is able to spot a clear seasonality pattern or trend.

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/

SHINY



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

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

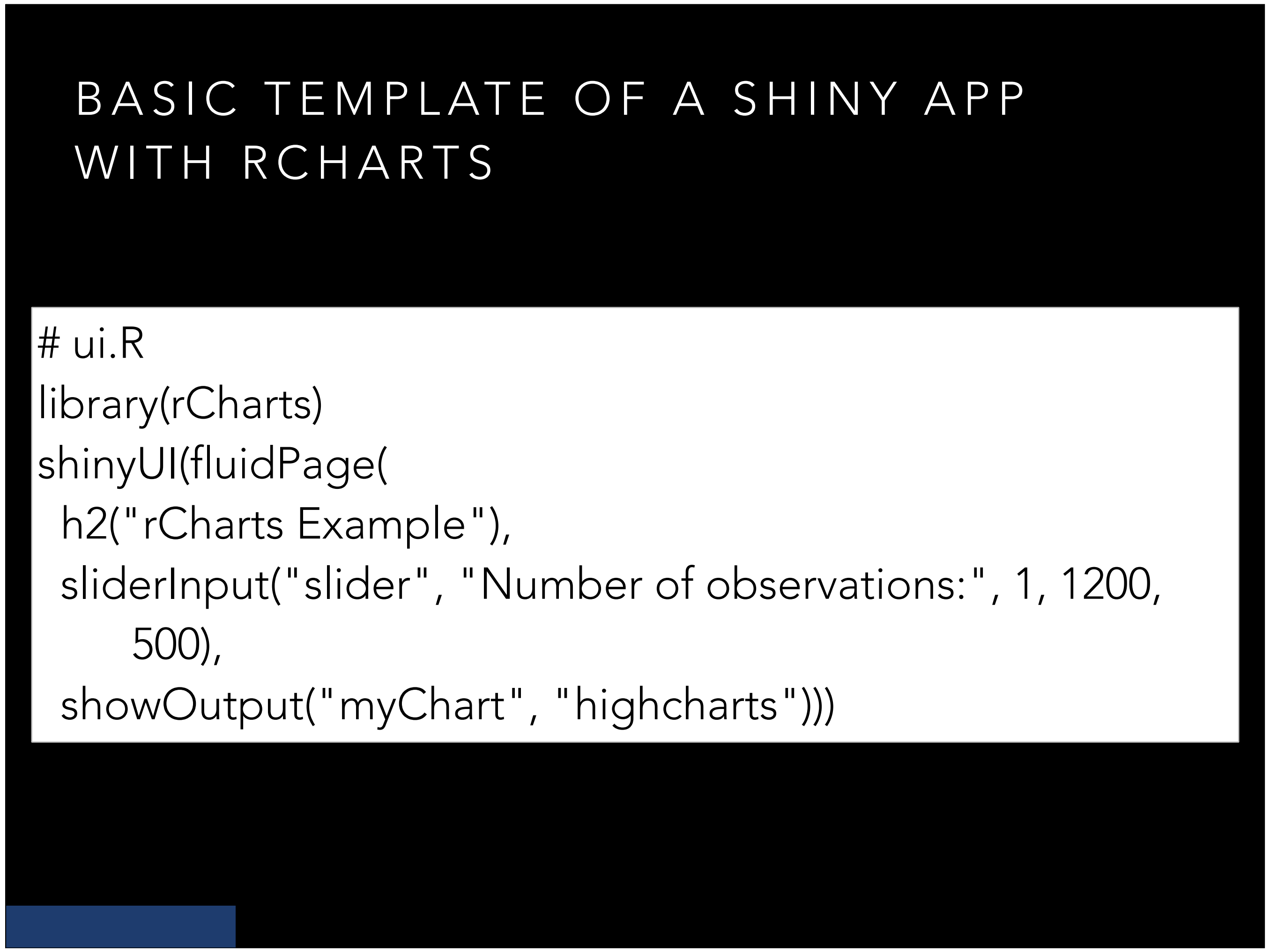
in ui.R you need a comma after each function call inside shinyUI() function

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.

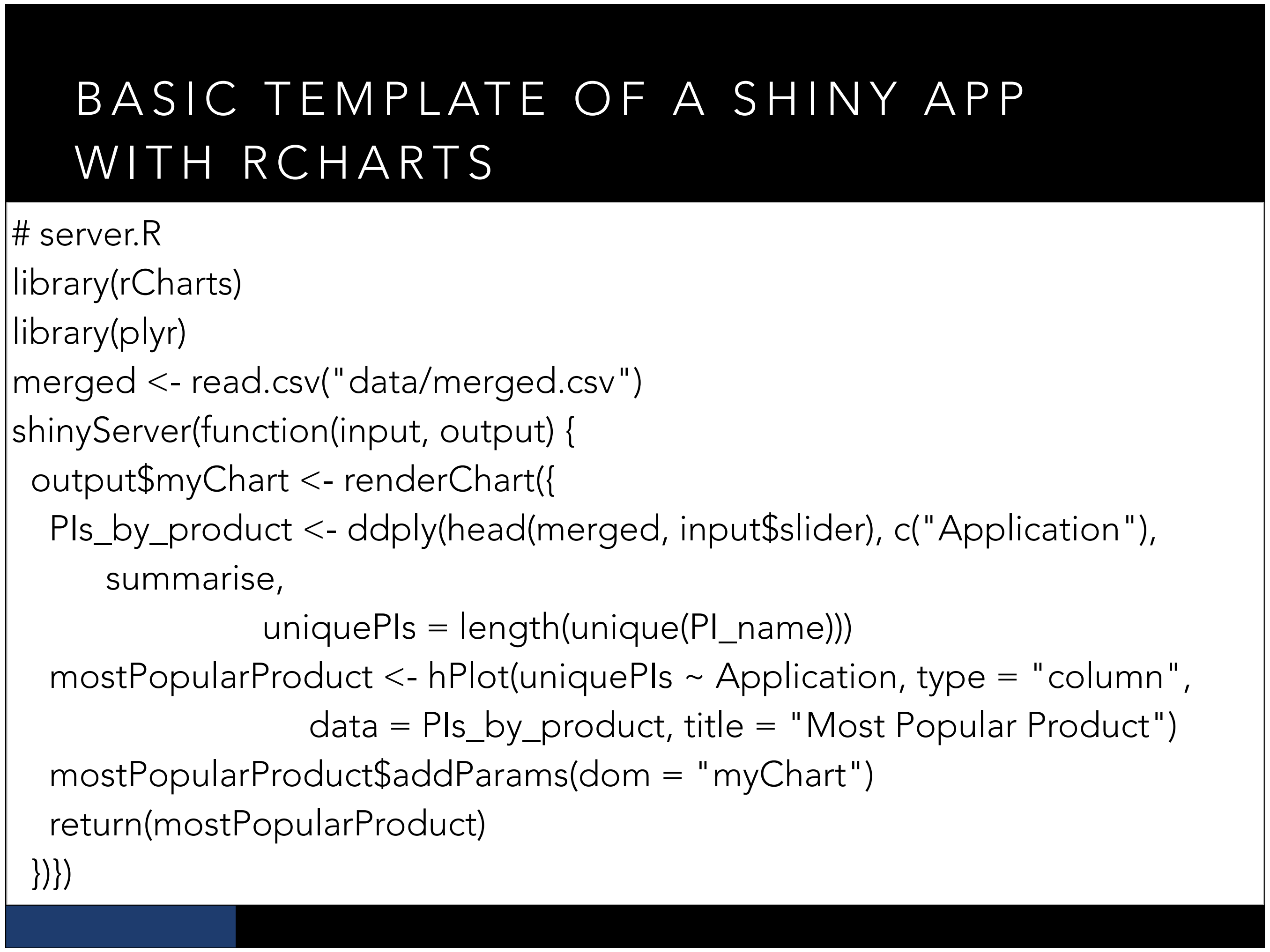
to build an rChart in a Shiny app you only need to wrap it in a renderChart() function and then add a few lines at the end to connect it to the domain name “myChart”, which will be referenced in the ui.R file inside showOutput() function. That’s how both of them are connected.



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")))
```

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.



BASIC TEMPLATE OF A SHINY APP WITH RCHARTS

```
# server.R
library(rCharts)
library(plyr)
merged <- read.csv("data/merged.csv")
shinyServer(function(input, output) {
  output$myChart <- renderChart({
    PIs_by_product <- ddply(head(merged, input$slider), c("Application"),
          summarise,
                        uniquePIs = length(unique(PI_name)))
    mostPopularProduct <- hPlot(uniquePIs ~ Application, type = "column",
                              data = PIs_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

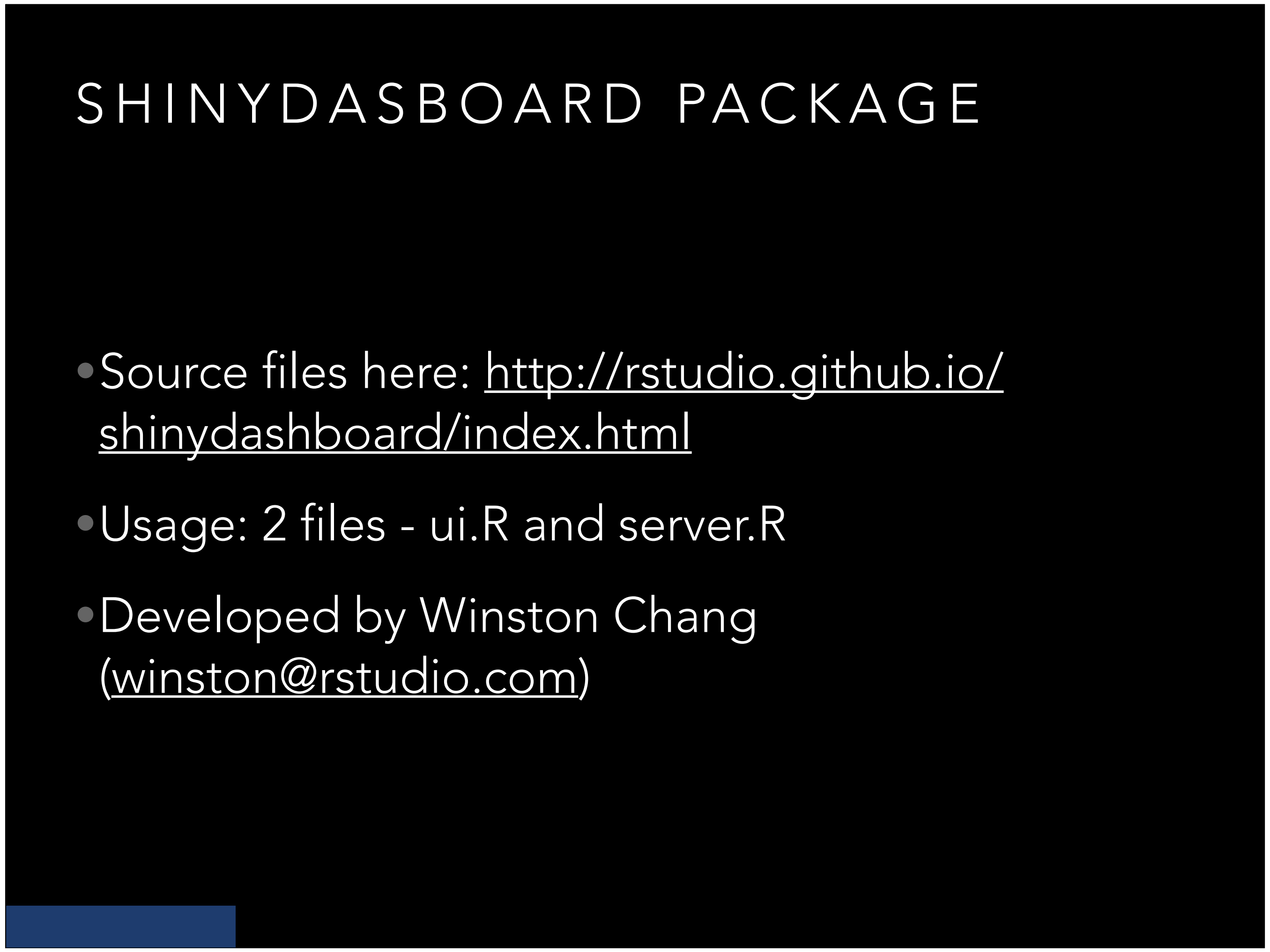
in ui.R you need a comma after each function call inside shinyUI() function

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.

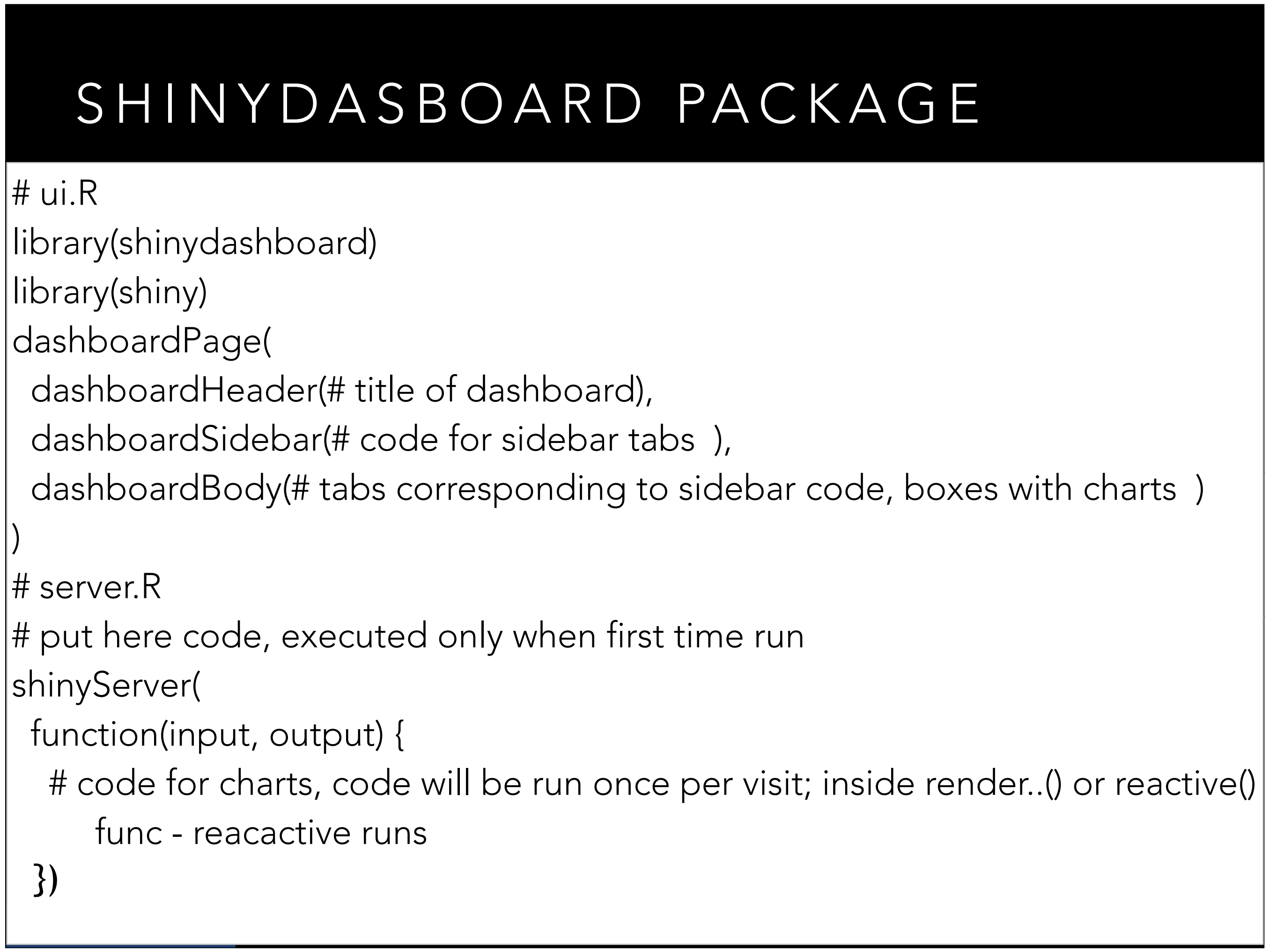
to build an rChart in a Shiny app you only need to wrap it in a renderChart() function and then add a few lines at the end to connect it to the domain name “myChart”, which will be referenced in the ui.R file inside showOutput() function. That’s how both of them are connected.



SHINYDASBOARD PACKAGE

- Source files here: http://rstudio.github.io/shinydashboard/index.html
- Usage: 2 files - ui.R and server.R
- Developed by Winston Chang (winston@rstudio.com)

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.

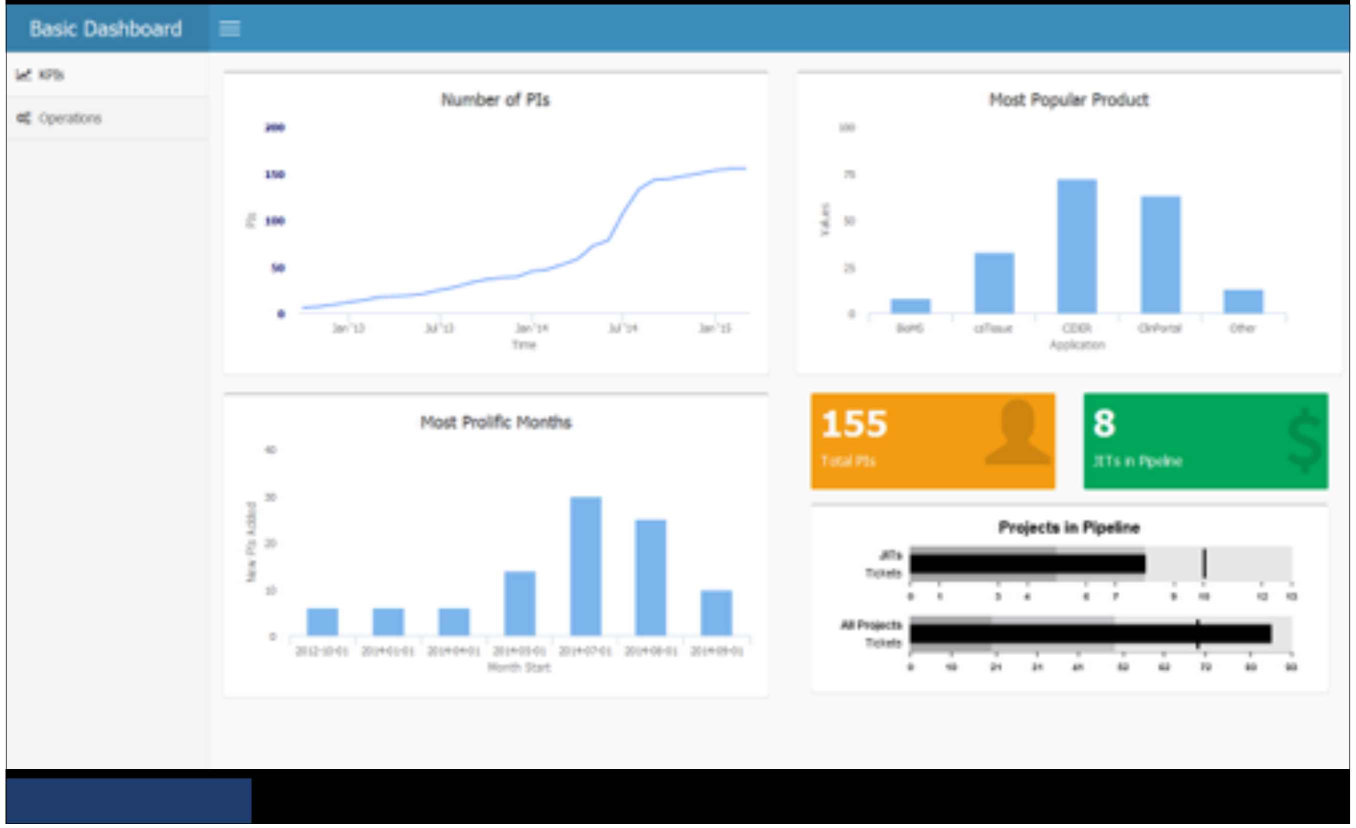


SHINYDASBOARD PACKAGE

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

DEMO DASHBOARD

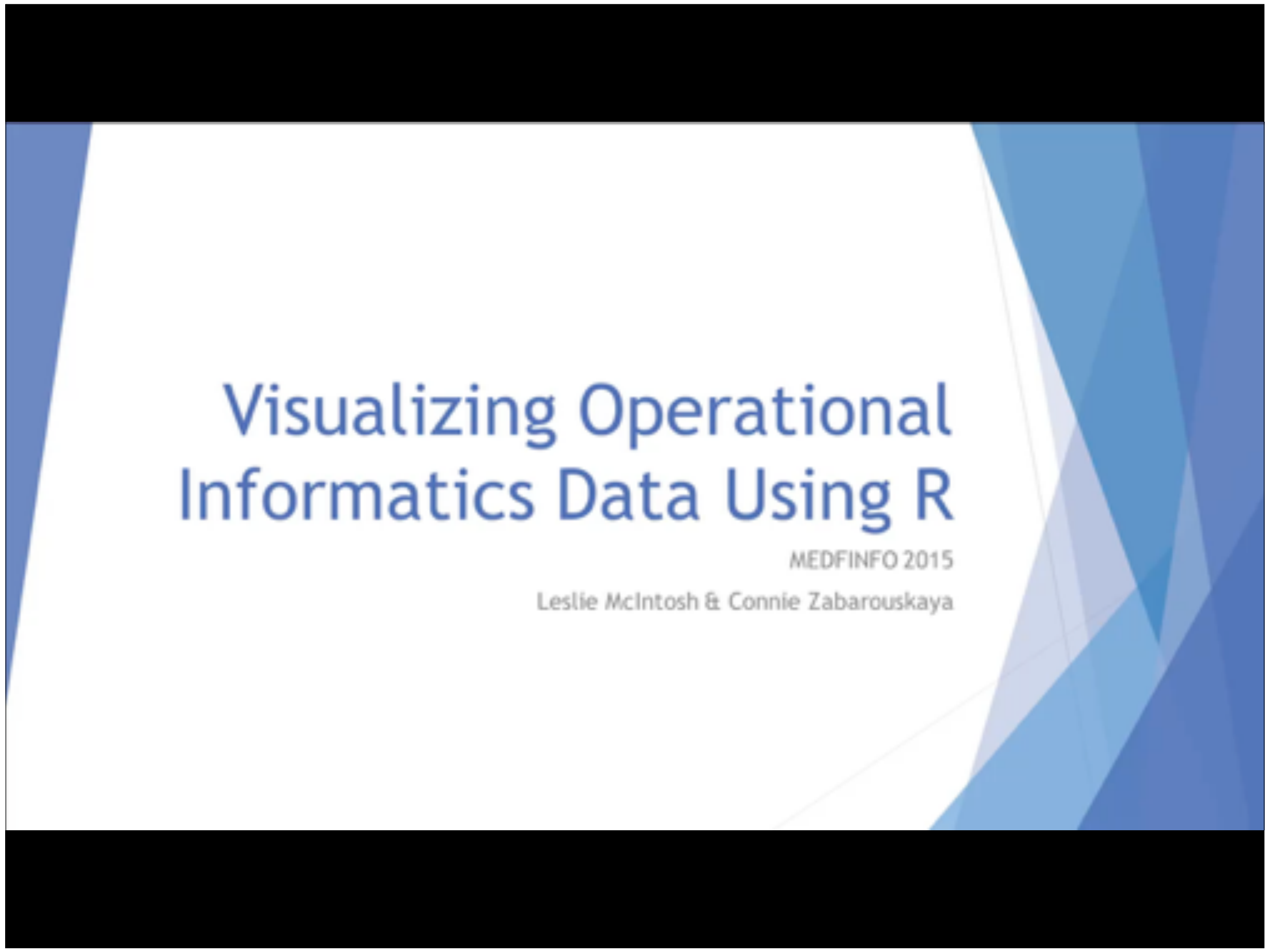


VERSATILE NATURE OF SHINY APPS

Use Shiny

- to automate reporting
- to discover patterns and information in data
- to interactively train people in skills: biostatistics, data analysis
- to support user authentication and different views of application based on user

HANDS-ON SHINY APPS: SHINYDASHBOARD PACKAGE



Visualizing Operational Informatics Data Using R

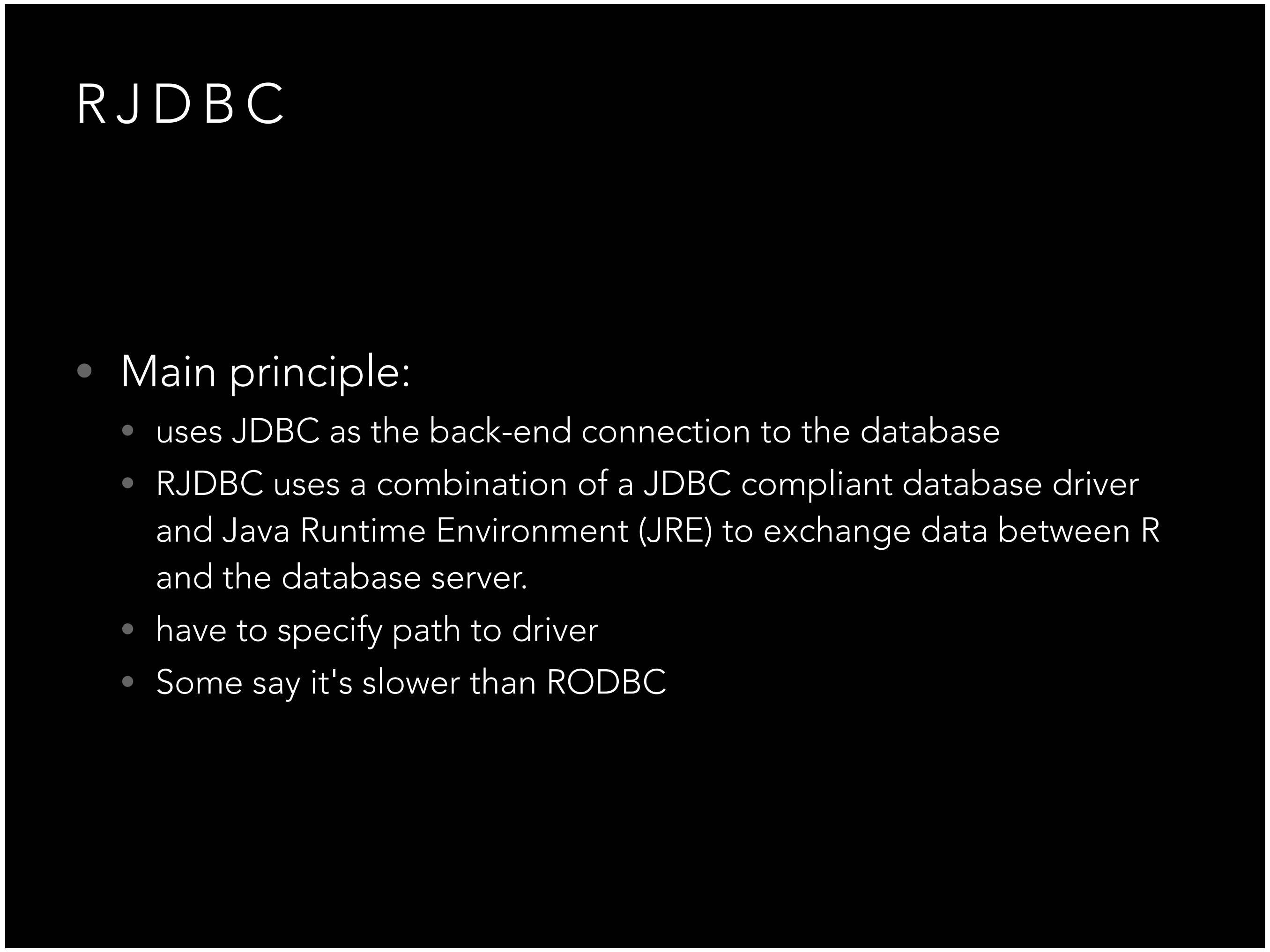
MEDFINFO 2015

Leslie McIntosh & Connie Zabarouskaya

CONNECTING TO DATABASES WITH R

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)



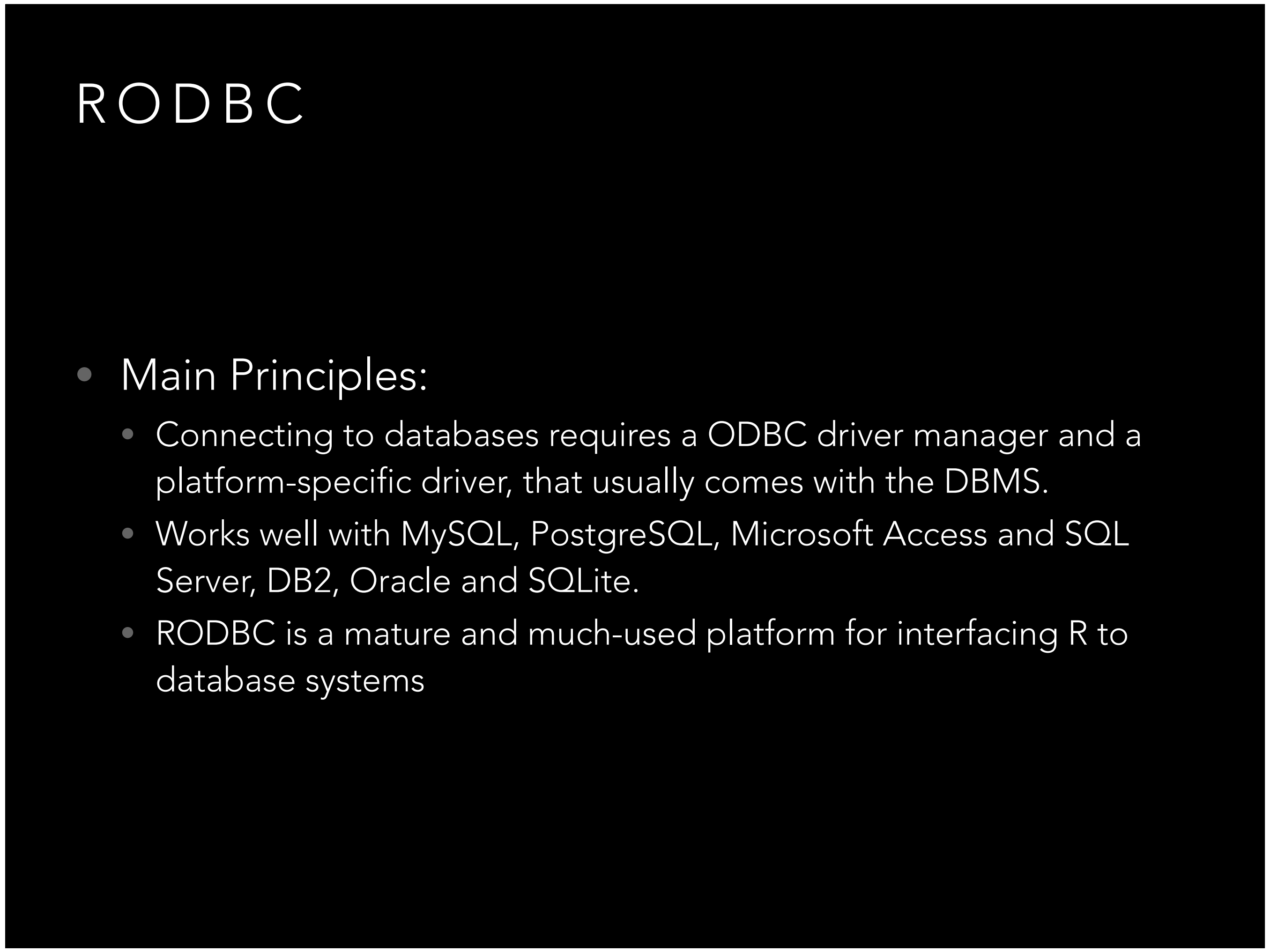
RJDBC

- Main principle:
  - uses JDBC as the back-end connection to the database
  - RJDBC uses a combination of a JDBC compliant database driver and Java Runtime Environment (JRE) to exchange data between R and the database server.
  - have to specify path to driver
  - Some say it's slower than RODBC

Note that the life time of a connection, result set, driver etc. is determined by the lifetime of the corresponding R object. Once the R handle goes out of scope (or if removed explicitly by rm) and is garbage-collected in R, the corresponding connection or result set is closed and released. This is important for databases that have limited resources (like Oracle) - you may need to add gc() by hand to force garbage collection if there could be many open objects. The only exception are drivers which stay registered in the JDBC even after the corresponding R object is released as there is currently no way to unload a JDBC driver (in RJDBC).
From <https://www.rforge.net/RJDBC/>
Type Handling
Type-handling is a rather complex issue, especially with JDBC as different databases support different data types. RJDBC attempts to simplify this issue by internally converting all data types to either character or numeric values. When retrieving results, all known numeric types are converted to R's numeric representation and all other types are treated as characters. When assigning parameters in parametrized queries, numeric, integer and character are the types used. Convenience methods like dbReadTable and dbWriteTable can only use the most basic SQL types, because they don't know what DBMS will be used. Therefore dbWriteTable uses only INTEGER, DOUBLE PRECISION or VARCHAR(255) to create the table. For all other types you'll have to use DBML statements directly.
From <https://www.rforge.net/RJDBC/>

RJDBC GENERAL STEPS

- General Steps:
  - Install Java
  - Download the JDBC driver for the specific database, record the path to that driver
  - Install RJDBC package
  - In R, create the connection driver object, specifying the type of driver and the path to driver
  - Connect to the database, specifying the driver object, host info, database and login credentials in the function dbConnect()
  - Work with your database



RODBC

- Main Principles:
  - Connecting to databases requires a ODBC driver manager and a platform-specific driver, that usually comes with the DBMS.
  - Works well with MySQL, PostgreSQL, Microsoft Access and SQL Server, DB2, Oracle and SQLite.
  - RODBC is a mature and much-used platform for interfacing R to database systems

RODBC is a mature and much-used platform for interfacing R to database systems
From <https://cran.r-project.org/web/packages/RODBC/vignettes/RODBC.pdf>
An ODBC environment consists of an ODBC Driver Manager and an ODBC compliant driver for the database server you would like to use. On Windows, the ODBC Driver Manager is built into the platform, but on Linux or other platforms an ODBC Driver Manager should be installed.
Connecting to a database from the RODBC driver involves identifying the location of the server, the name of the database, and supporting credentials (for example, user name and password). The name of the database is usually defined as an ODBC DSN. A DSN is a detailed reference to a database that is either local or remote from the client computer. You can consider a DSN as an alias to the database—it does not need to match the actual name of the database defined on the server.
From <http://brazenly.blogspot.in/2014/05/r-how-to-connect-r-to-database-oracle.html>

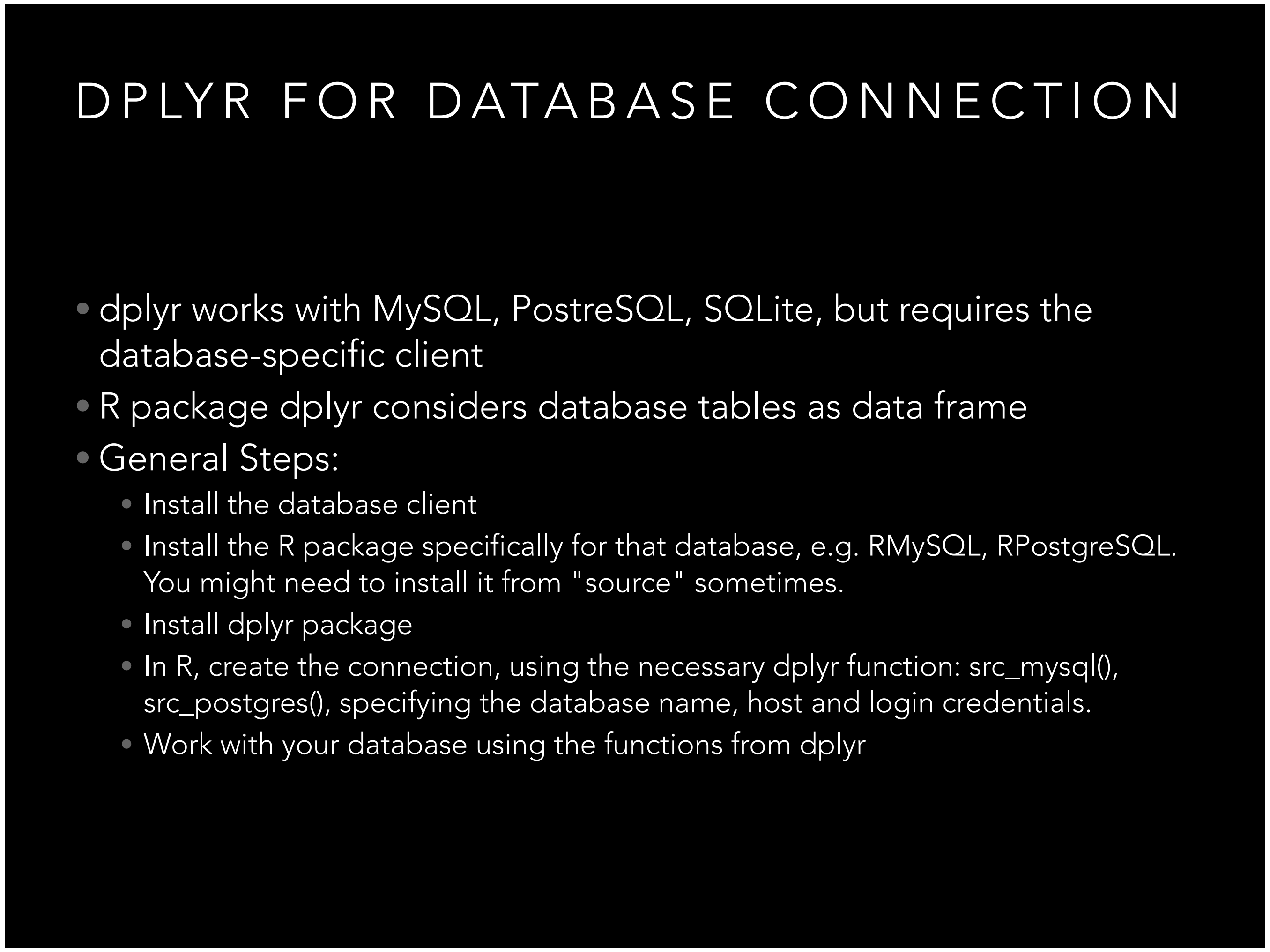
ODBC stands for Open Database Connectivity
provides API to access DBMS
RODBC does the same for R
From <http://www.unomaha.edu/mahbubulmajumder/data-science/fall-2014/lectures/20-database-mysql/20-database-mysql.html#/17>

RODBC GENERAL STEPS

- General Steps:
  - Install ODBC driver manager: for Windows it's already installed, for Mac/Linux/Unix need to install iODBC or unixODBC
  - Install the ODBC driver for the database in question on Windows. unixODBC seems to include some drivers in their default installation (consult http://www.unixodbc.org/ click on Drivers).
  - Configure the DSN (Data Source Name) in the GUI for Windows, or using config files on Mac and *nix machines
  - Install RODBC package. Careful on Mac and *nix machines, for unixODBC need to install RODBC from "source".
  - In R, create the connection, specifying the DSN and login credentials inside odbcConnect() function
  - Work with your database

SPECIALIZED PACKAGES

- Specialized packages for specific databases, e.g. RMySQL, ROracle, RSQLite, etc. may perform better than RJDBC or RODBC
- General Steps:
  - Install the database client
  - Install the R package specifically for that database, e.g. RMySQL. You might need to install it from "source" sometimes.
  - In R, create the connection, specifying the driver as general name for you database, e.g. MySQL(), database and login credentials inside dbConnect() function.
  - Work with your database



DPLYR FOR DATABASE CONNECTION

- dplyr works with MySQL, PostreSQL, SQLite, but requires the database-specific client
- R package dplyr considers database tables as data frame
- General Steps:
  - Install the database client
  - Install the R package specifically for that database, e.g. RMySQL, RPostgreSQL. You might need to install it from "source" sometimes.
  - Install dplyr package
  - In R, create the connection, using the necessary dplyr function: src_mysql(), src_postgres(), specifying the database name, host and login credentials.
  - Work with your database using the functions from dplyr

R package dplyr considers database tables as data frame
From <http://www.unomaha.edu/mahbubulmajumder/data-science/fall-2014/lectures/20-database-mysql/20-database-mysql.html#/11>
In terms of functionality, MySQL lies somewhere between SQLite and PostgreSQL. It provides a wider range of built-in functions, but it does not support window functions (so you can’t do grouped mutates and filters).
From <https://cran.r-project.org/web/packages/dplyr/vignettes/databases.html>
Picking a database
If you don’t already have a database, here’s some advice from my experiences setting up and running all of them. SQLite is by far the easiest to get started with, but the lack of window functions makes it limited for data analysis. PostgreSQL is not too much harder to use and has a wide range of built in functions. Don’t bother with MySQL/MariaDB: it’s a pain to set up and the documentation is subpar. Google bigquery might be a good fit if you have very large data, or you’re willing to pay (a small amount of) money for someone else to look after your database.
From <https://cran.r-project.org/web/packages/dplyr/vignettes/databases.html>

WHAT METHOD TO CHOOSE?

RJDBC	RODBC	SPECIALIZED	DPLYR
Very Easy Setup: Java + Database Jdbc Driver	Quite Complicated Setup: - Need Odbc Driver Manager (Windows Odbc Driver Manager, Or Iodbc, Or Unixodbc), And An Odbc Driver. - Some Odbc Drivers Are Not Free	Setup Difficulty Depends On The Os: - Need Database Client Installed	Setup Difficulty Is Similar To Specialized Package Setup: - Need Database Client Installed - 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 "Lazy"
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 Databases	Somewhat Versatile, Works With Mysql, Postresql, Sqlite

R & BIG DATA

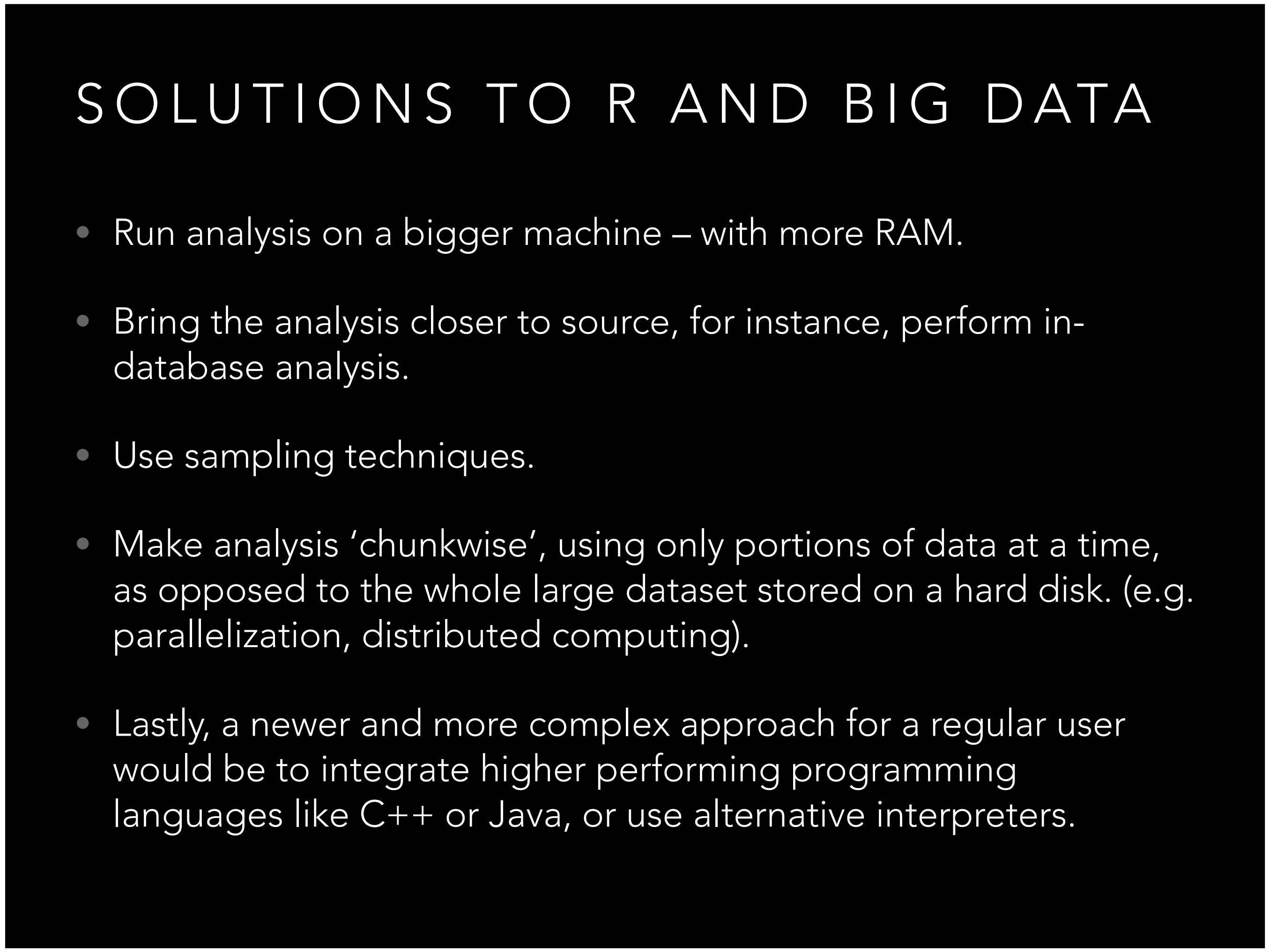
R AND BIG DATA

- What is Big Data anyway?
- Base R is good for up to 1M records
- Handling data beyond 1M records is possible but needs additional effort

Big Data are, “data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it”. When they talk about Big Data they also refer to the three or sometimes four V’s – volume, velocity, variety, and the fourth - veracity, the latter being common in healthcare field.
In practical terms however, Big Data can be understood as data that can’t be analyzed in memory, whether the RAM memory is 4GB on a standard desktop or larger on a server.

Jan Wijffels proposed in his talk at the useR!-Conference 2013 a trisection of data according to its size. As a rule of thumb: Data sets that contain up to one million records can easily be processed with standard R. Data sets with about one million to one billion records can also be processed in R, but need some additional effort. Data sets that contain more than one billion records need to be analyzed by map reduce algorithms. These algorithms can be designed in R and processed with connectors to Hadoop and the like.

Through our experience working in the Medical School, we have noticed that while the Big Data movement has been underway for many years, the healthcare industry lags behind in adopting and implementing innovative applications and algorithms that can turn Big Data into actionable knowledge. This slower adoption may be because healthcare data comprise all four V’s as well as the concerns about privacy. Regardless, the field has reached the critical point where there is a need and opportunity to facilitate moving health data to information more quickly to improve health outcomes. The same refers to operations data in Bioinformatics field.

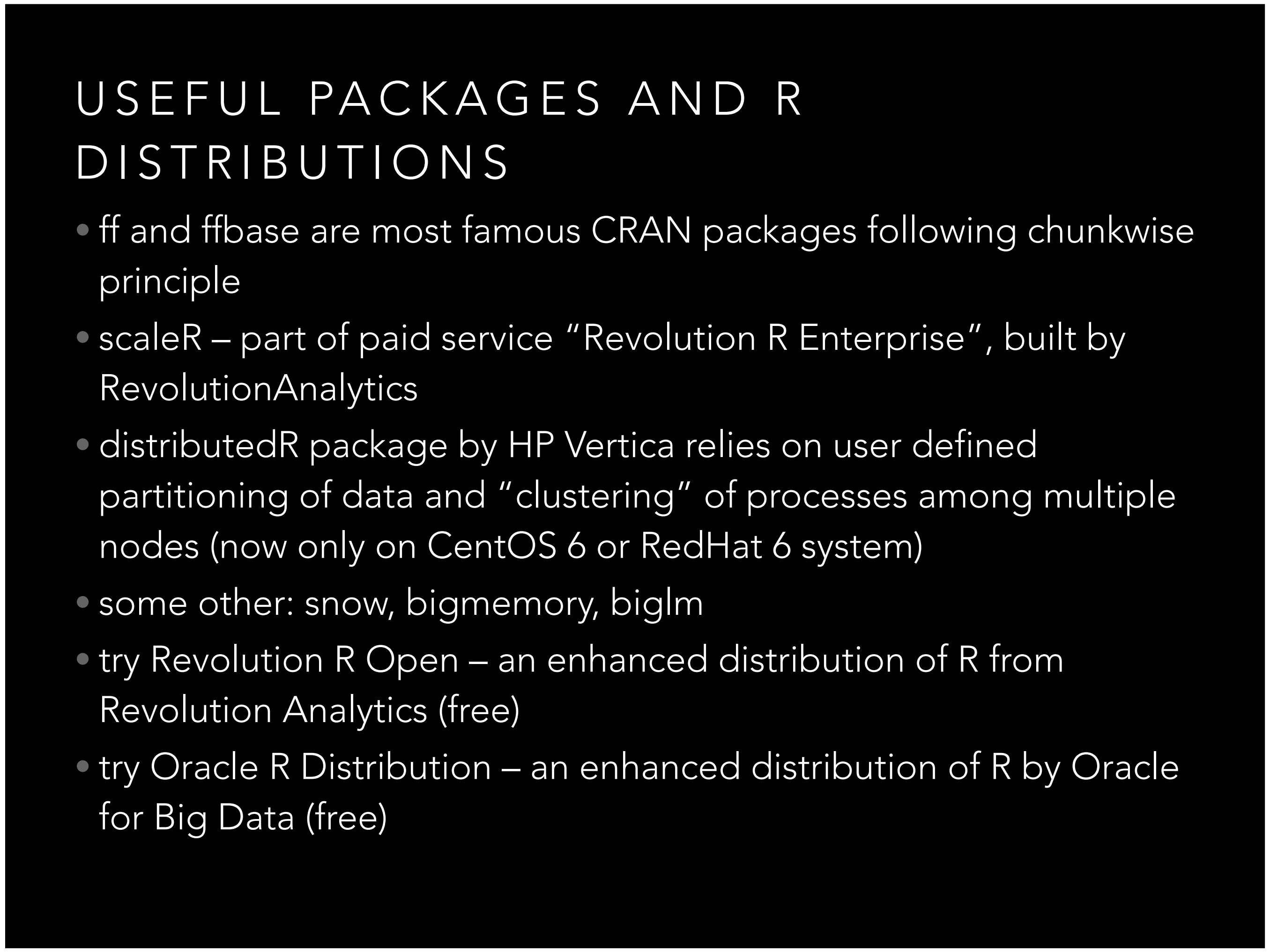


SOLUTIONS TO R AND BIG DATA

- Run analysis on a bigger machine – with more RAM.
- Bring the analysis closer to source, for instance, perform in-database analysis.
- Use sampling techniques.
- Make analysis 'chunkwise', using only portions of data at a time, as opposed to the whole large dataset stored on a hard disk. (e.g. parallelization, distributed computing).
- Lastly, a newer and more complex approach for a regular user would be to integrate higher performing programming languages like C++ or Java, or use alternative interpreters.

1. Run analysis on a bigger machine – with more RAM. For instance on a common 4U Intel server, which can hold up to 2TB of RAM, R would be much more light and fast, however, hogging an entire 2TB server for one personal R instance might be a bit wasteful. So people run large cloud instances for as long as they need them, run VMs on their server hardware, or run the likes of RStudio Server on their server hardware.
2. Bring the analysis closer to source, for instance, perform in-database analysis, as opposed to streaming data into R and then analyzing in memory. Some packages that enable this method are RHadoop, SparkR, etc.
3. Use sampling techniques. According to Hadley Wickham's useR! Talk (user Conf 2013), sample based model building is acceptable, at least if the size of data is greater than one billion records.
4. Make analysis 'chunkwise', using only portions of data at a time, as opposed to the whole large dataset stored on a hard disk. As a side effect, splitting the dataset into chunks essentially means parallelization of analysis in the spirit of MapReduce, Spark and Hadoop, especially if the algorithms allow parallel analysis of the chunks in principle. However, you have to keep in mind that you need to use functions and commands that are explicitly designed to deal with hard disc specific datatypes, more on this later. Another method, similar to chunkwise is to use distributed computing, for instance to install package distributedR, developed by HP's Vertica team In fact, on a local machine or perhaps even install Revolution R Open as an enhanced distribution of R from Revolution Analytics, which speeds up all R functions considerably.
5. Lastly, a newer and more complex approach for a regular user would be to integrate higher performing programming languages like C++ or Java, or use alternative interpreters, example of this implementation include pqR and Renjin, and even OracleR, all open-source products. pqR (Pretty Quick R) has been known to provide better memory management and support for automatic multithreading.

If we come back to the issue of velocity of Big Data, you may experience different scenarios. Your Big Data may be static, as in one large file of several million rows of data, for example, if you downloaded the large dataset from a source one time, or if you get regular daily database exports, or it can be dynamic, where you have to accept constant feeds of data and analyze on the fly. In the former scenario, chunkwise parallel methods would be good ways to handle analyses, for instance with ff and ffbase packages, while in the latter case, you may need to bring analysis to the source, the database or more likely to the cloud.



USEFUL PACKAGES AND R DISTRIBUTIONS

- ff and ffbase are most famous CRAN packages following chunkwise principle
- scaleR – part of paid service “Revolution R Enterprise”, built by RevolutionAnalytics
- distributedR package by HP Vertica relies on user defined partitioning of data and “clustering” of processes among multiple nodes (now only on CentOS 6 or RedHat 6 system)
- some other: snow, bigmemory, biglm
- try Revolution R Open – an enhanced distribution of R from Revolution Analytics (free)
- try Oracle R Distribution – an enhanced distribution of R by Oracle for Big Data (free)

Let’s talk about chunkwise methods of working with Big Data, using R. “ff” and “ffbase” are probably the most famous CRAN packages following this principle. Revolution R Enterprise, as a commercial product, uses this strategy with their popular “scaleR” package as well, but it’s part of a paid service – Revolution R Enterprise. Compared to ff and ffbase, Revolution scaleR offers a wider range and faster growth of analytic functions. For instance, the Random Forest algorithm has recently been added to the scaleR function set, which is not yet available in ffbase. However, the distributedR package includes random forest algorithm, along with k-means clustering and logistic regression.

Revolution R Open info: https://mran.revolutionanalytics.com/rro

Oracle R Distribution: http://www.oracle.com/technetwork/database/database-technologies/r/r-distribution/overview/index.html

R & BIG DATA

DEMONSTRATION

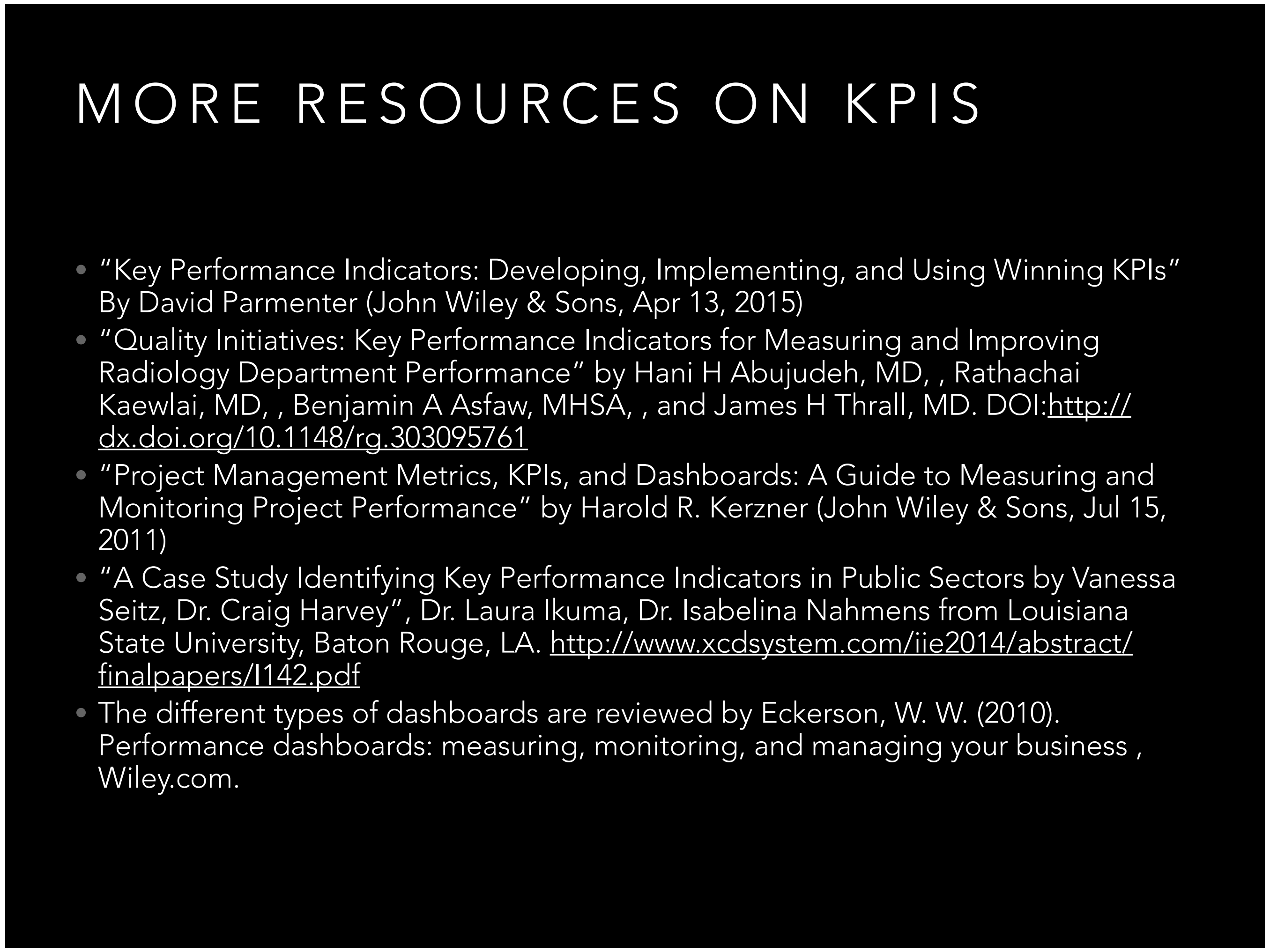
RESOURCES

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/
- 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://gallery.idashboards.com/preview/?guestuser=webpharm&dashID=89
- ClicData Poor Dashboard Examle: http://www.clicdata.com/example-health/
- 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)

- Grammar of Graphics Chart: “The Grammar of Graphics (Statistics and Computing)” by Leland Wilkinson
- What happens behind the scenes: http://rcharts.io/howitworks/
- Presentation on how to share: http://rcharts.io/NYC_May_2014/slides/02_share/#5
- 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
- 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/



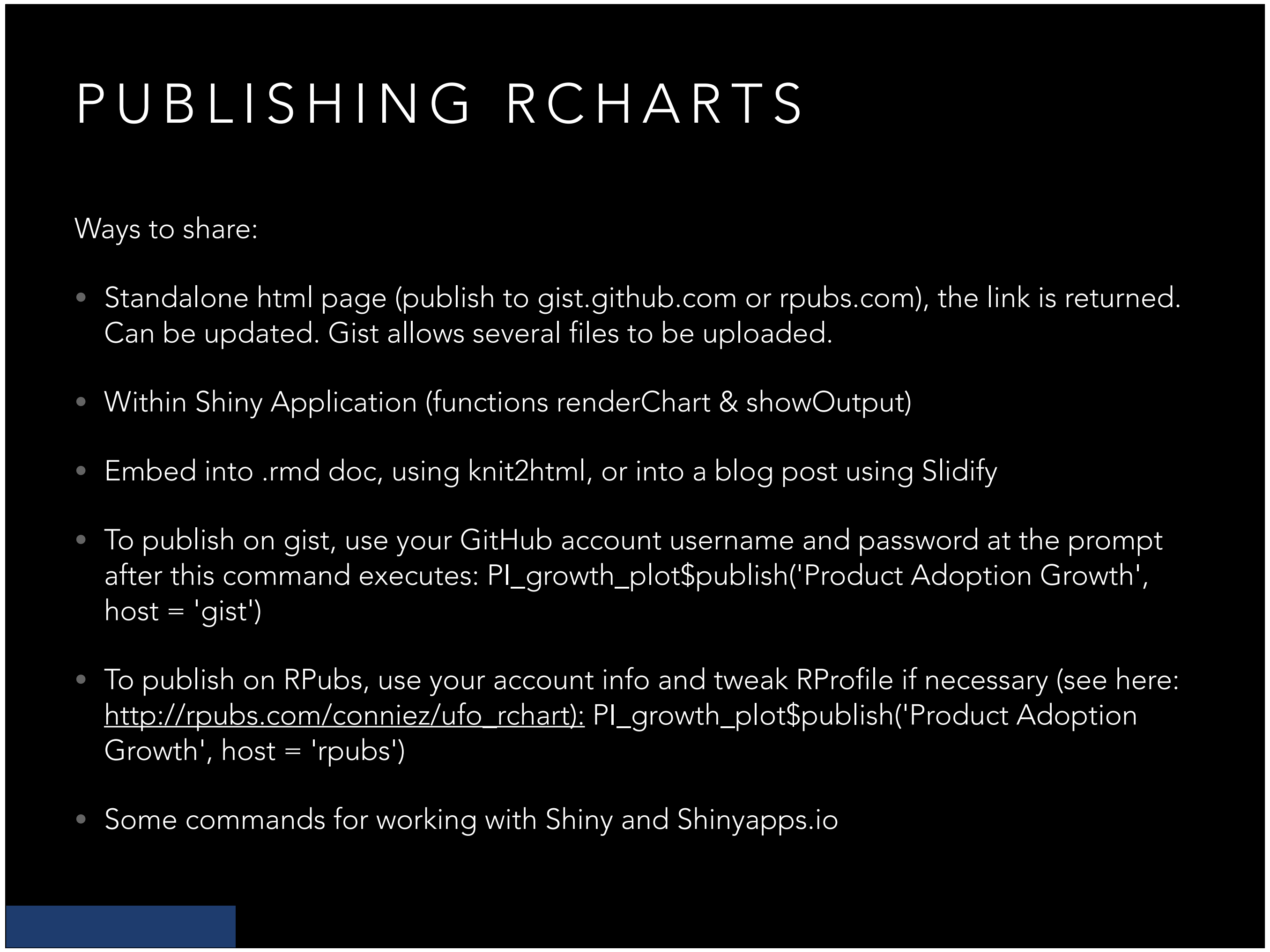
MORE RESOURCES ON KPIS

- "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.

RESOURCES ON GOOGLEVIS

- Find more comparisons and reviews here:
- http://ouzor.github.io/blog/2014/11/21/interactive-visualizations.html
- https://rpubs.com/miguelpatricio/r_charts_googleVis
- More examples and instructions:
- https://cran.r-project.org/web/packages/googleVis/vignettes/googleVis_examples.html
- https://cran.r-project.org/web/packages/googleVis/vignettes/googleVis.pdf



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
- To publish on gist, use your GitHub account username and password at the prompt after this command executes: PI_growth_plot$publish('Product Adoption Growth', host = 'gist')
- To publish on RPubs, use your account info and tweak RProfile if necessary (see here: http://rpubs.com/conniez/ufo_rchart): PI_growth_plot$publish('Product Adoption Growth', host = 'rpubs')
- Some commands for working with Shiny and Shinyapps.io

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.

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

BIG DATA RESOURCES

- M. Zeleny, Management support systems: towards integrated knowledge management, Human Systems Management 7(1) (1987) 59–70.
- http://www.infoworld.com/article/2880360/big-data/learn-to-crunch-big-data-with-r.html
- http://www.r-bloggers.com/five-ways-to-handle-big-data-in-r/
- http://www.vertica.com/hp-vertica-documentation/hp-vertica-distributed-r-product-documentation/
- http://www8.hp.com/us/en/software-solutions/asset/software-asset-viewer.html?asset=1982345&module=1966839&docname=4AA5-8375ENW&page=1966052
- Revolution R Open info: http://mran.revolutionanalytics.com/rro/
- Oracle R Distribution: http://www.oracle.com/technetwork/database/database-technologies/r/r-distribution/overview/index.html

SLIDES AND DEMO-DASHBOARD

https://github.com/CBMIWU/MEDINFOTutorial_VisualizingOpsData

OBRIGADO