Topic06.Data Visualization

資料科學 Data Science
張家銘 Jia-Ming Chang
政治大學資訊科學系

Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.
- The web site of the book
- The credit of individual is indicated in the bottom part of the slide.
  - ie.,

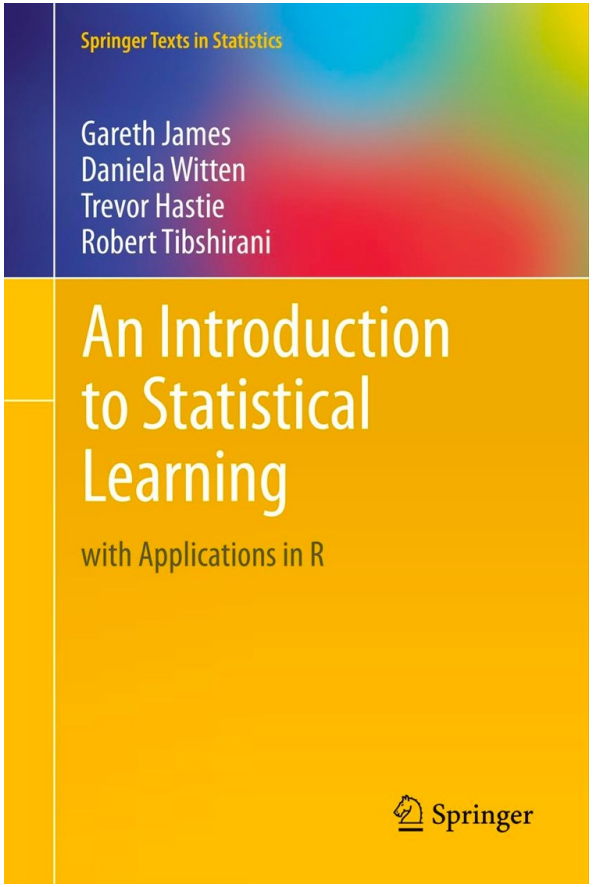


Figure 3.18, *An Introduction to Statistical Learning with Applications in R,* by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani

Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "Practical Data Science with R (Manning, 2019)"
- The web site of the book
- The credit of individual is indicated in the bottom part of the slide.
  - ie.,

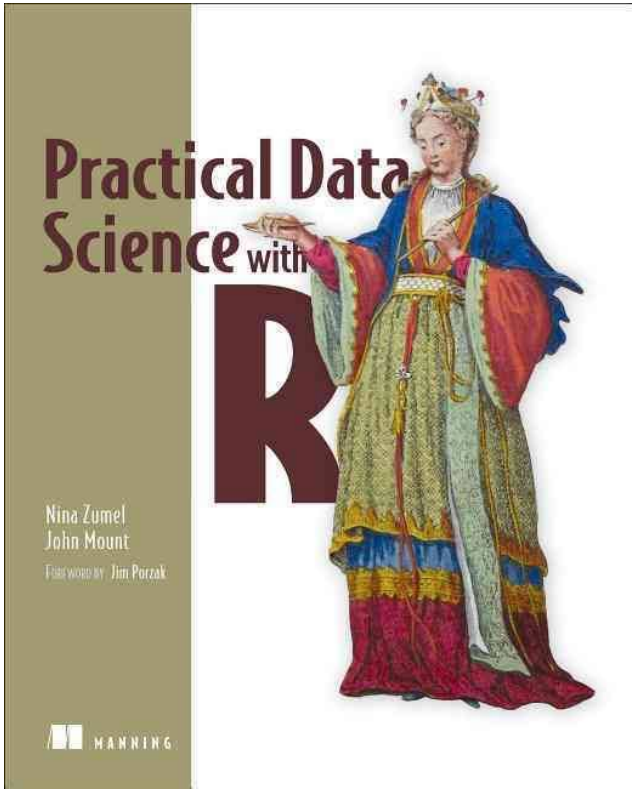
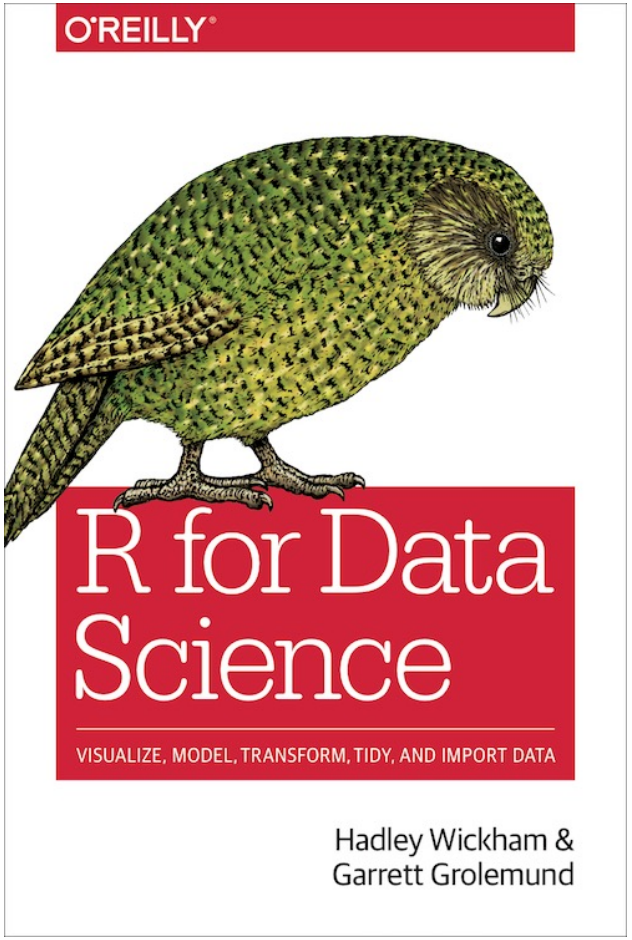


Figure 7.6, *Practical Data Science with R* by Nina Zumel and John Mount

Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "R for Data Science" under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 License.
- The web site of the book
- The credit of individual is indicated in the bottom part.
  - ie.,

R for Data Science by Garrett Grolemund, Hadley Wickham



Data Science this week

- **Elon Musk's OpenAI Unveils a Simpler Way for Machines to Learn**
- The group says it has a more practical way to get software to learn tasks, such as steering robots, that require multiple actions.
- by Tom Simonite March 27, 2017
  - https://goo.gl/LhEX3X
  - https://openai.com/

Recap for the last week

Typical problems revealed by data summaries

- Missing values
- Invalid values and outliers
- Data range
- Units

```
     custid        sex      is.employed         income                       marital.stat health.ins                          housing.type recent.move
 Min.   :   2068   F:440   Mode :logical   Min.   : -8700   Divorced/Separated:155   Mode :logical   Homeowner free and clear    :157   Mode :logical
 1st Qu.: 345667   M:560   FALSE:73        1st Qu.: 14600   Married           :516   FALSE:159       Homeowner with mortgage/loan:412   FALSE:820
 Median : 693403           TRUE :599       Median : 35000   Never Married     :233   TRUE :841       Occupied with no rent       : 11   TRUE :124
 Mean   : 698500           NA's :328       Mean   : 53505   Widowed           : 96   NA's :0         Rented                      :364   NA's :56
 3rd Qu.:1044606                           3rd Qu.: 67000                                            NA's                        : 56
 Max.   :1414286                           Max.   :615000

  num.vehicles        age              state.of.res
 Min.   : 0.000   Min.   :  0.0   California  :100
 1st Qu.: 1.000   1st Qu.: 38.0   New York    : 71
 Median : 2.000   Median : 50.0   Pennsylvania: 70
 Mean   : 1.916   Mean   : 51.7   Texas       : 56
 3rd Qu.: 2.000   3rd Qu.: 64.0   Michigan    : 52
 Max.   : 6.000   Max.   :146.7   Ohio        : 51
 NA's   :56.000                   (Other)     :600
```

Some information is easier to read from a graph, and some from a summary.

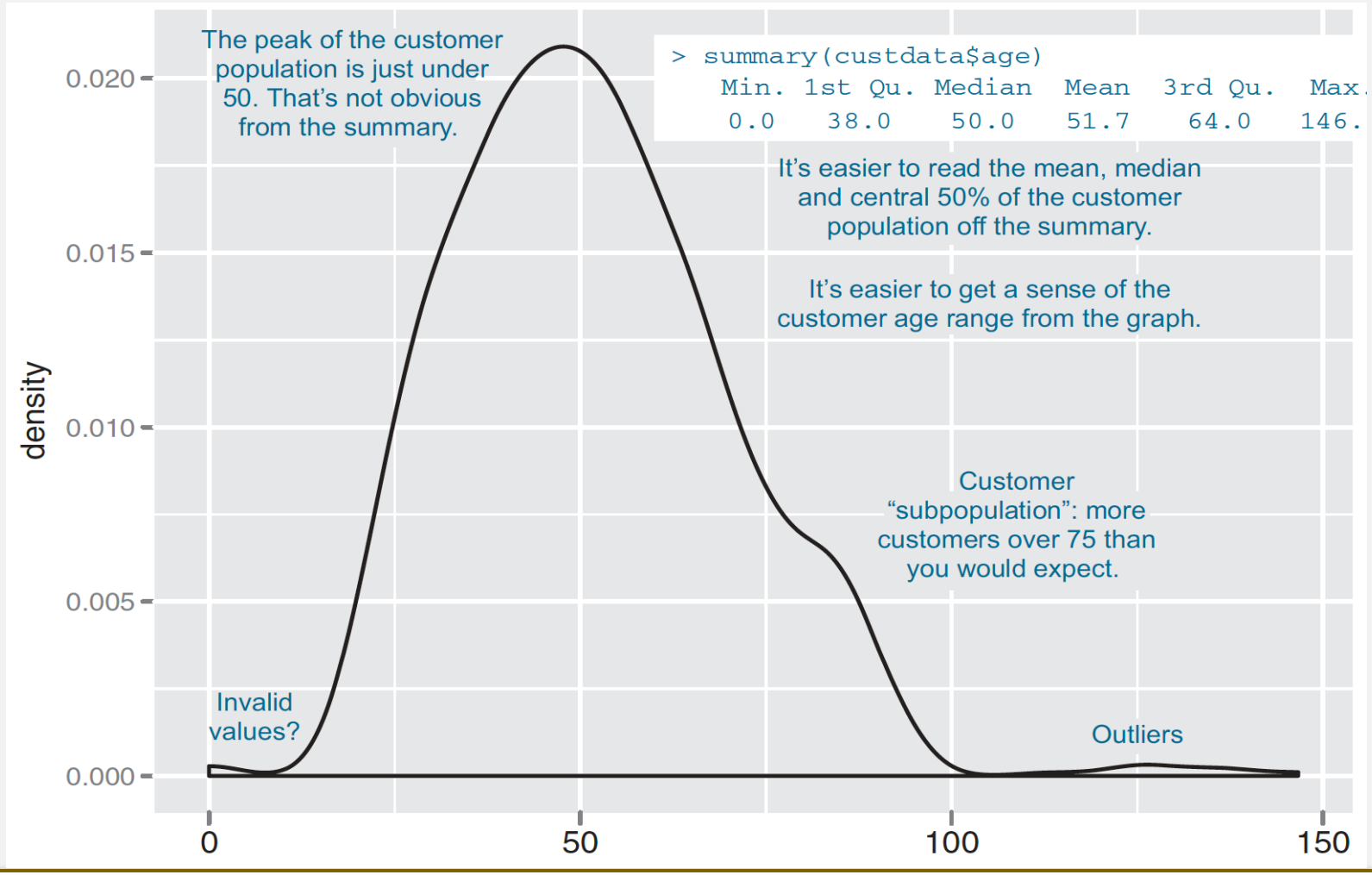
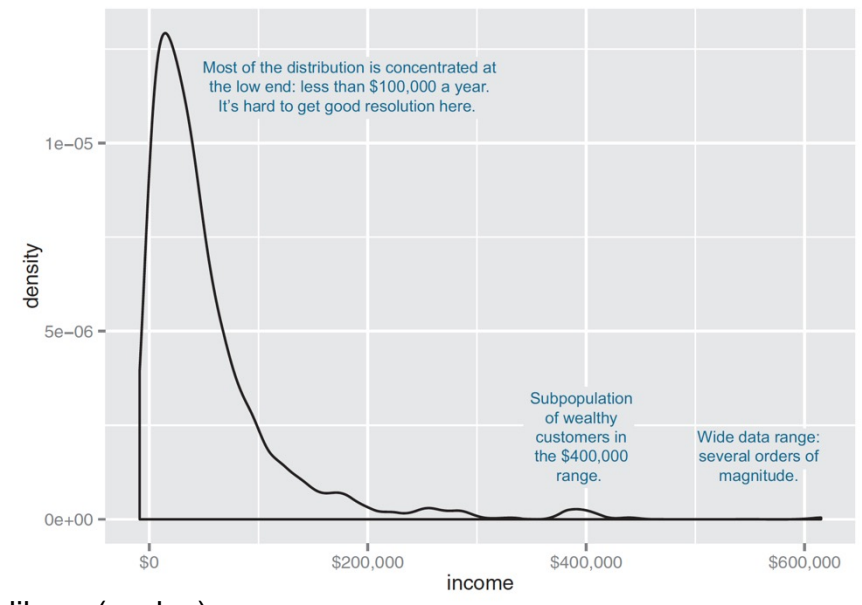


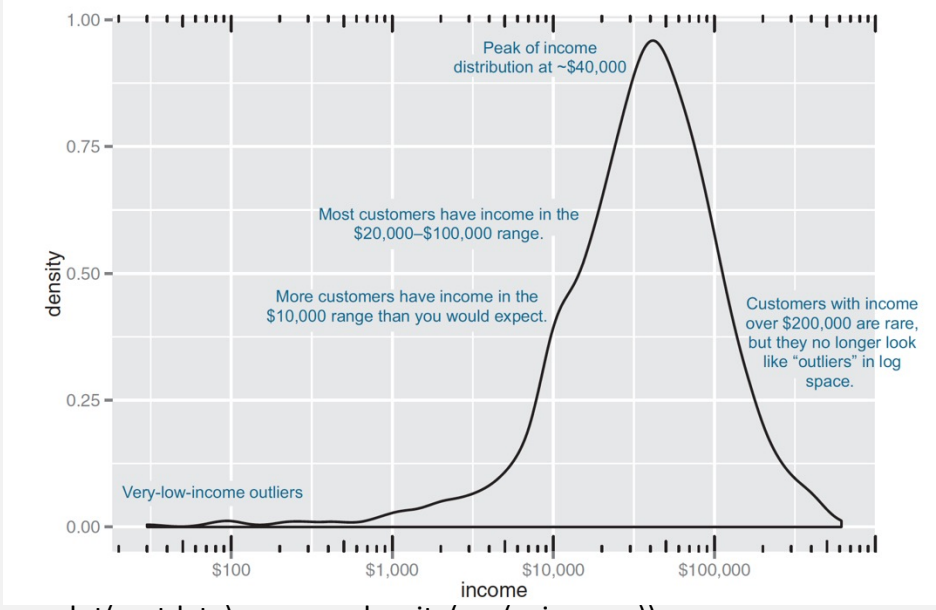
Figure 3.1, *Practical Data Science with R* by Nina Zumel, John Mount

DENSITY PLOTS

- When the data range
  - very wide
  - the mass of the distribution is heavily concentrated to one side (difficult to see the details of its shape)
  - non-negative




```
library(scales)
ggplot(custdata) + geom_density(aes(x=income)) +
scale_x_continuous(labels=dollar)
```

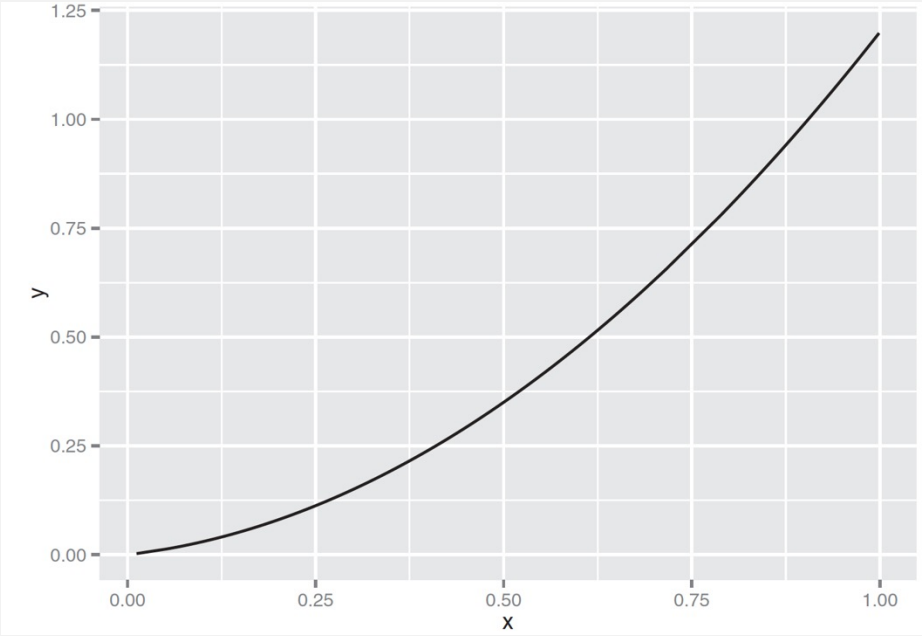



```
ggplot(custdata) + geom_density(aes(x=income)) +
  scale_x_log10(breaks=c(100,1000,10000,100000), labels=dollar) +
annotation_logticks(sides="bt")
```

Line points vs. smoothing curve

relation.R

- x <- runif(100)
- y <- x^2 + 0.2*x
- ggplot(data.frame(x=x,y=y), aes(x=x,y=y)) + geom_line()
- ggplot(custdata2, aes(x=age, y=income)) + geom_point() + geom_smooth() + ylim(0, 200000)



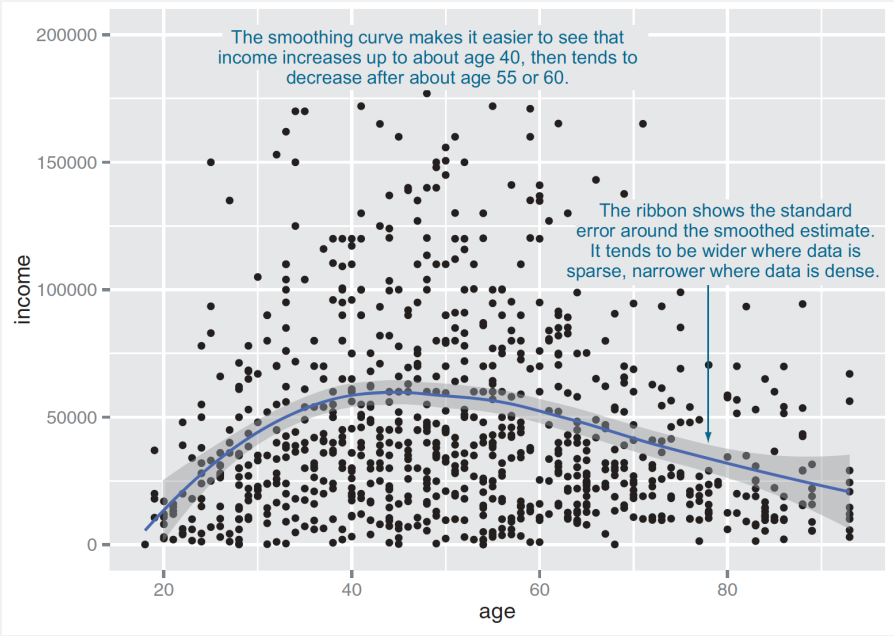


Figure3.9 & 3.12, *Practical Data Science with R* by Nina Zumel, John Mount

A nearly lognormal distribution and its log

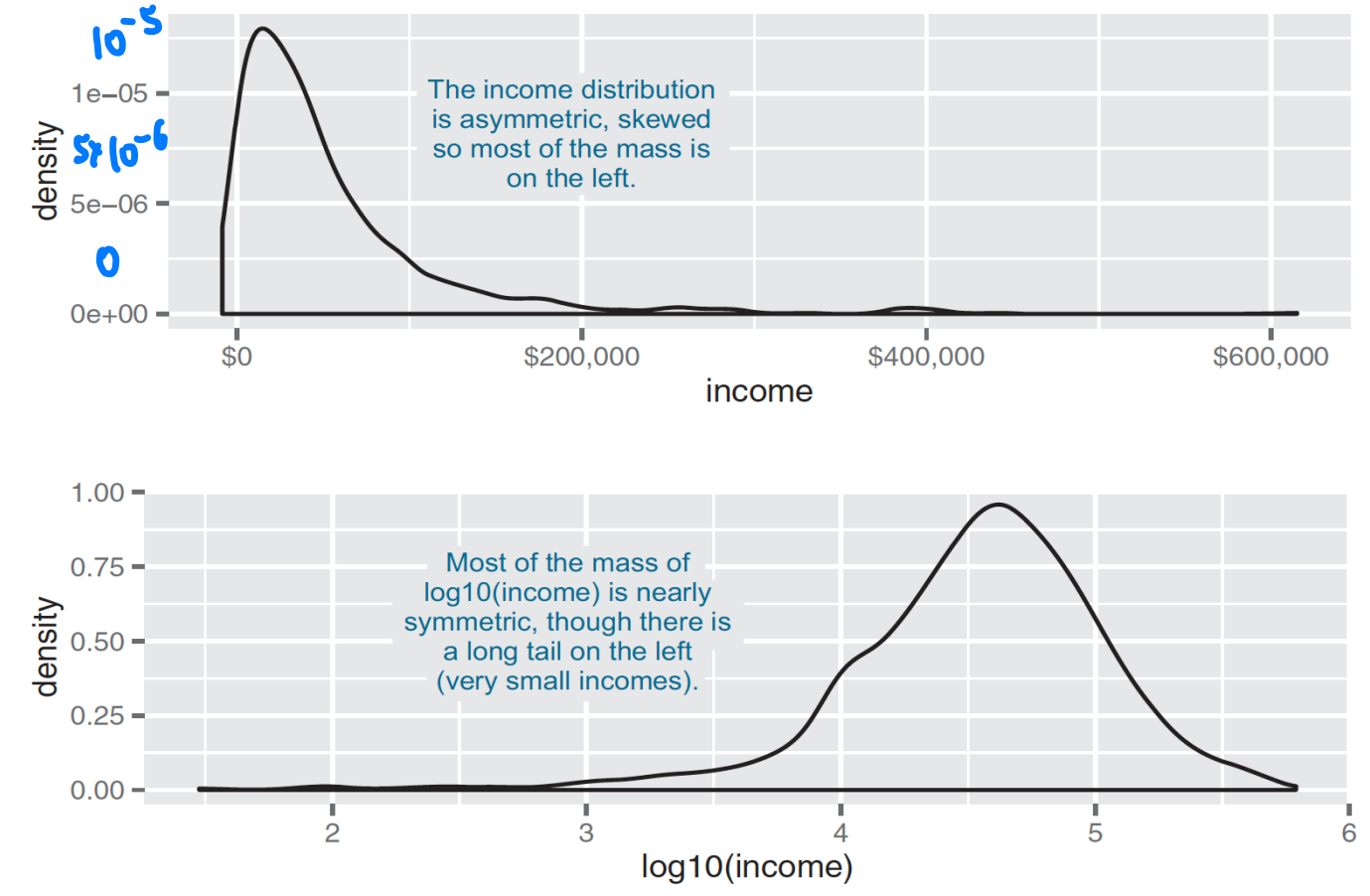
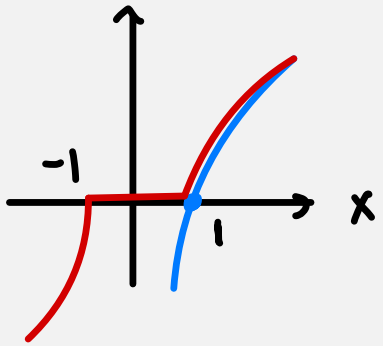


Figure 4.4, *Practical Data Science with R* by Nina Zumel, John Mount

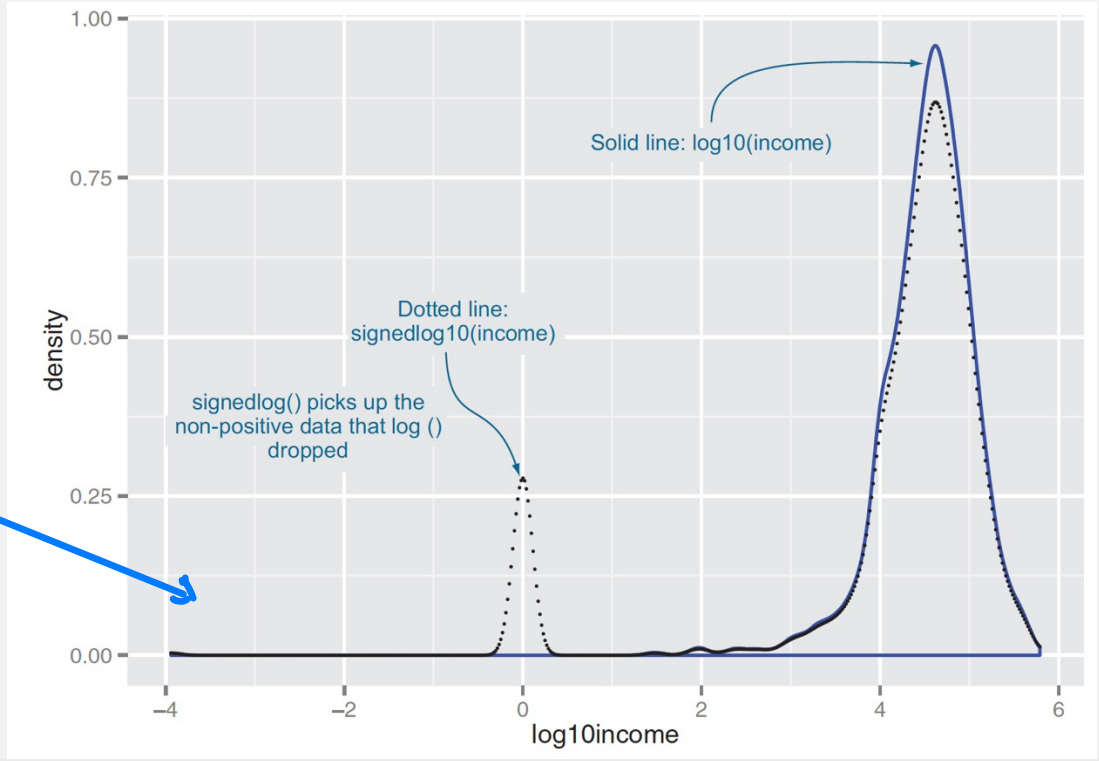
$y = |x| \le 1 \; ? \; 0 : \text{sign}(x) \times \log_{10}|x|$

$y = \log_{10} x$



If the data is negative?

```
signedlog10 <- function(x) {
        ifelse(abs(x) <= 1, 0, sign(x)*log10(abs(x)))
}
```



no negative data in the income dataset

Sampling for modeling and validation

runif (n, min = 0, max = 1)

- train/calibration, test
- Splitting into test and training using a random group mark

# of records in the data frame

```
custdata$gp <- runif(dim(custdata)[1])
testSet <- subset(custdata, custdata$gp
<= 0.1)
trainingSet <- subset(custdata,
custdata$gp > 0.1)
```

Record grouping

households

household_id gp

```
hh <- unique(hhdata$household_id)
households <- data.frame(household_id = hh, gp =
runif(length(hh)))
hhdata <- merge(hhdata, households, by="household_id")
```

hh1 ~ hh5

length(hh) → 5

hhdata

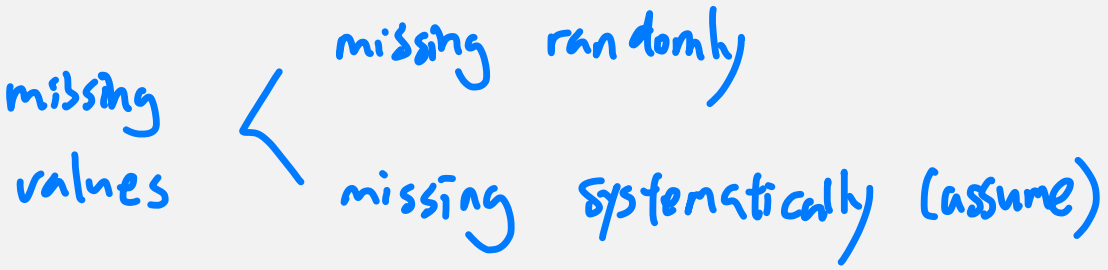
	household_id	cust_id	income
Household 1	hh1	cust1	30200
Household 2	hh2	cust1	24800
	hh2	cust2	134800
Household 3	hh3	cust1	299000
	hh3	cust2	65000
	hh3	cust3	95000
Household 4	hh4	cust1	38800
	hh4	cust2	0
Household 5	hh5	cust1	100300
	hh5	cust2	27000

hhdata after merge

	household_id	cust_id	income	gp
Household 1	hh1	cust1	30200	0.8625189
Household 2	hh2	cust1	24800	0.8880607
	hh2	cust2	134800	0.8880607
Household 3	hh3	cust1	299000	0.9130094
	hh3	cust2	65000	0.9130094
	hh3	cust3	95000	0.9130094
Household 4	hh4	cust1	38800	0.5244124
	hh4	cust2	0	0.5244124
Household 5	hh5	cust1	100300	0.5388283
	hh5	cust2	27000	0.5388283

Note that each member of a household has the same group number

Summary



- What you do with missing values depends on how many there are, and whether they're missing randomly or systematically.
- When in doubt, assume that missing values are missing systematically.
- Appropriate data transformations can make the data easier to understand and easier to model.
- Normalization and rescaling are important when relative changes are more important than absolute ones.
- Data provenance records help reduce errors as you iterate over data collection, data treatment, and modeling.

Data visualization

Data Relationships

NOMINAL COMPARISON
This is a simple comparison of the quantitative values of subcategories. Example: Number of visitors to various websites.

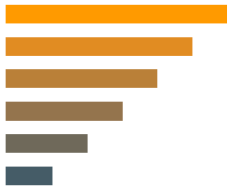
DEVIATION
This examines how data points relate to each other, particularly how far any given data point differs from the mean. Example: Amusement park tickets sold on a rainy day vs. a regular day.

TIME-SERIES
This tracks changes in values of a consistent metric over time. Example: Monthly sales.

DISTRIBUTION
This shows data distribution, often around a central value. Example: Heights of players on a basketball team.

CORRELATION
This is data with two or more variables that may demonstrate a positive or negative correlation to each other. Example: Salaries according to education level.

PART-TO-WHOLE RELATIONSHIPS
This shows a subset of data compared to the larger whole. Example: Percentage of customers purchasing specific products.



RANKING
This shows how two or more values compare to each other in relative magnitude. Example: Historic weather patterns, ranked from the hottest months to the coldest.

Now that you've got a handle on the most common data types and relationships you'll most likely have to work with, let's dive into the different ways you can visualize that data to get your point across.

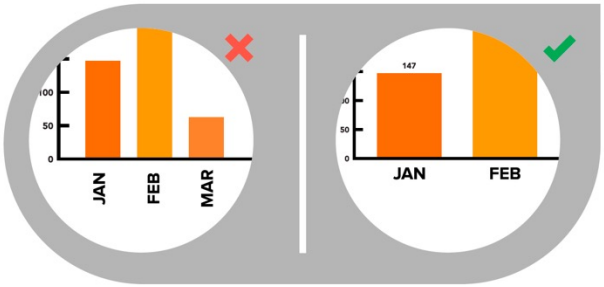
Credit by *Data Visualization 101: How To Design Charts and Graphs, by HubSpot and Visage*

Guide to chart types

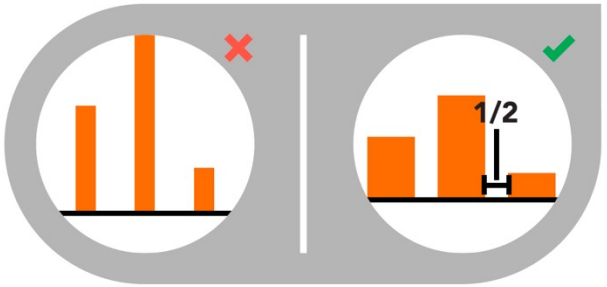
Credit by https://visage.co/data-visualization-101-pie-charts/

BAR CHART

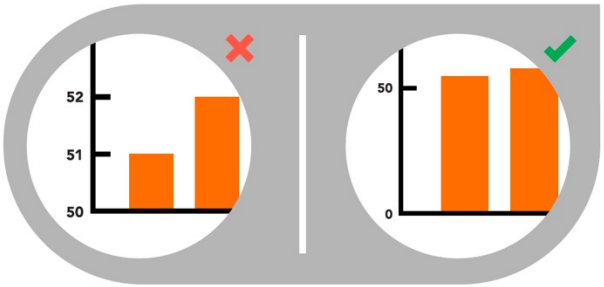
DESIGN BEST PRACTICES



USE HORIZONTAL LABELS
Avoid steep diagonal or vertical type, as it can be difficult to read.

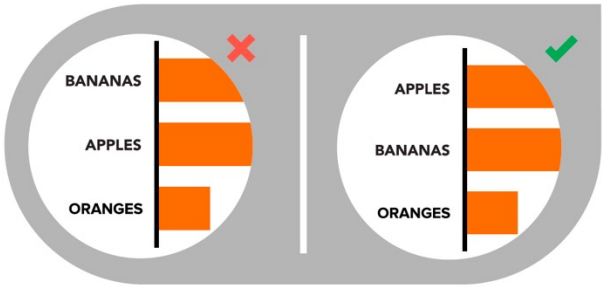


SPACE BARS APPROPRIATELY
Space between bars should be ½ bar width.



START THE Y-AXIS VALUE AT 0
Starting at a value above zero truncates the bars and doesn't accurately reflect the full value.

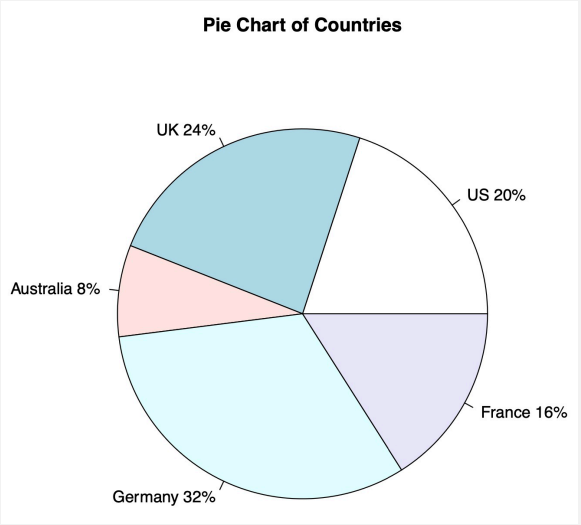
USE CONSISTENT COLORS
Use one color for bar charts. You may use an accent color to highlight a significant data point.



ORDER DATA APPROPRIATELY
Order categories alphabetically, sequentially, or by value.

Credit by https://visage.co/data-visualization-101-pie-charts/

Pie Chart



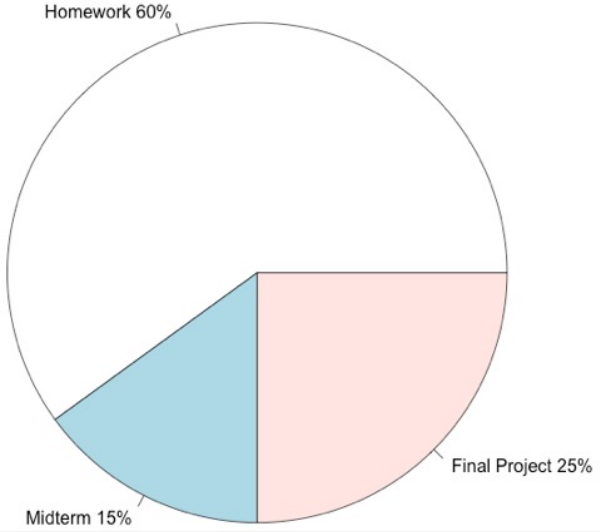
- piechart.R

```
slices <- c(10, 12, 4, 16, 8)
lbls <- c("US", "UK", "Australia", "Germany", "France")
pct <- round(slices/sum(slices)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(slices,labels = lbls, main="Pie Chart of Countries")
```

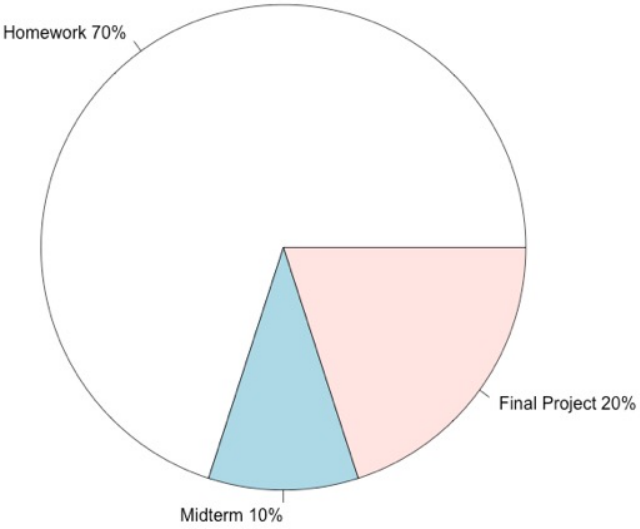
labels
percents: 20 24 8 32 16

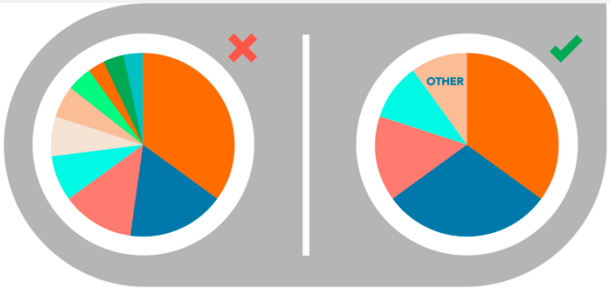
Data visualization

- Grading1
  - Homework 60%
  - Midterm 15%
  - Final Project 25%



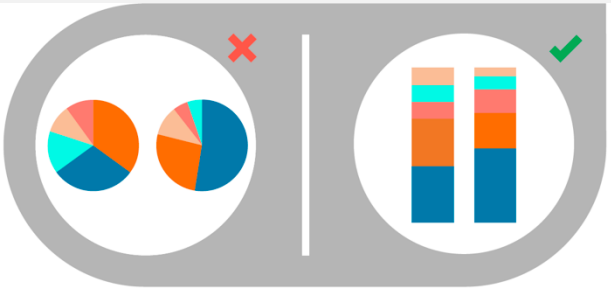
- Grading2
  - Homework 70%
  - Midterm 10%
  - Final project 20%





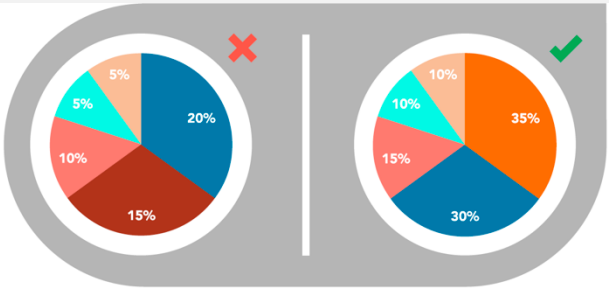
VISUALIZE NO MORE THAN 5 CATEGORIES PER CHART

It is difficult to differentiate between small values; depicting too many slices decreases the impact of the visualization. If needed, you can group smaller values into an “other” or “miscellaneous” category, but make sure it does not hide interesting or significant information.



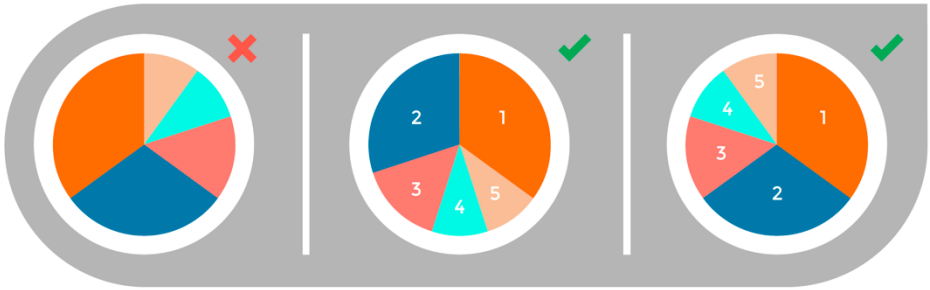
DON’T USE MULTIPLE PIE CHARTS FOR COMPARISON

Slice sizes are very difficult to compare side-by-side. Use a stacked bar chart instead.



MAKE SURE ALL DATA ADDS UP TO 100%

Verify that values total 100% and that pie slices are sized proportionate to their corresponding value.



PIE CHART

DESIGN BEST PRACTICES

ORDER SLICES CORRECTLY

There are two ways to order sections, both of which are meant to aid comprehension:

OPTION 1

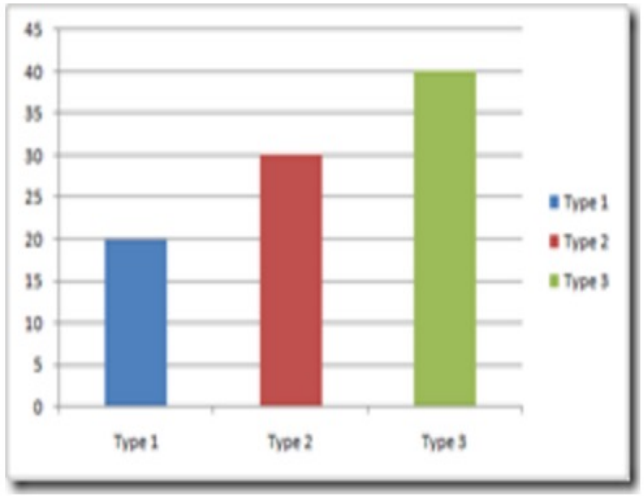
Place the largest section at 12 o’clock, going clockwise. Place the second largest section at 12 o’clock, going counterclockwise. The remaining sections can be placed below, continuing

OPTION 2

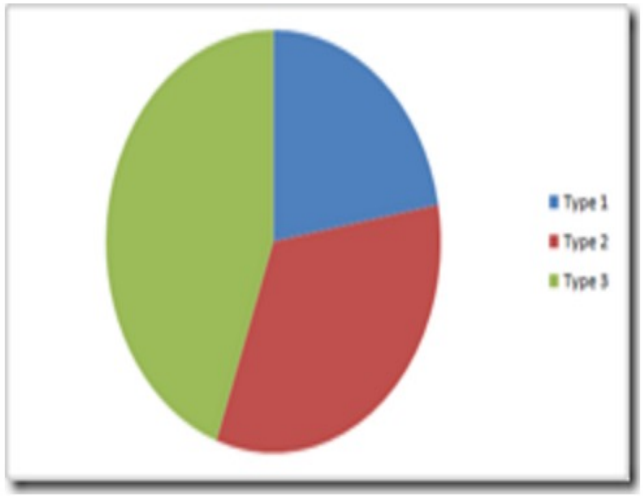
Start the largest section at 12 o’clock, going clockwise. Place remaining sections in descending order, going clockwise.

Pie Chart vs Bar Chart

- Visualize no more than 5 categories per chart.



Simplest Visual Comparison

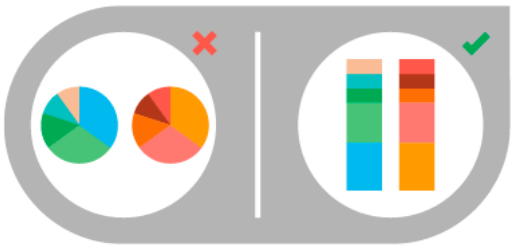


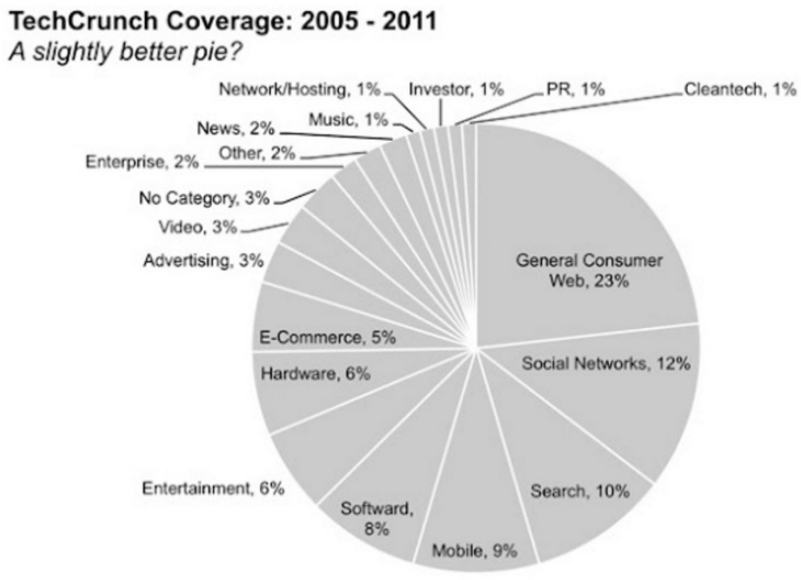
Same Numbers, Harder to Compare

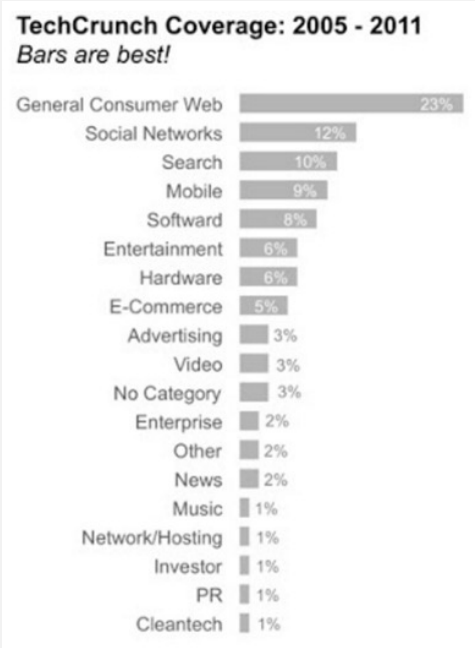
Credit by http://www.powerpivotpro.com/2009/12/the-death-of-pie-charts/

Pie Chart vs Bar Chart

- Visualize no more than 5 categories per chart.



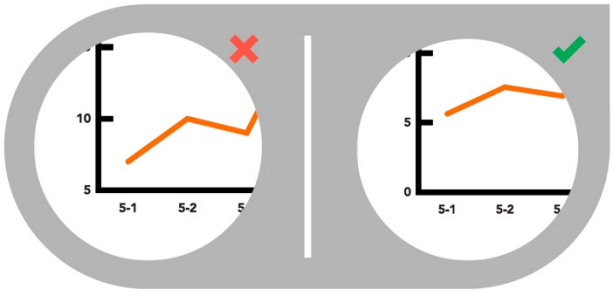




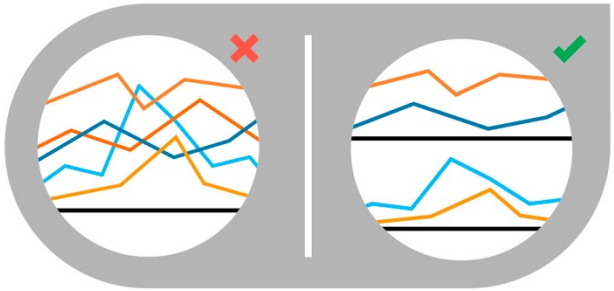
http://www.storytellingwithdata.com/blog/2011/07/death-to-pie-charts

LINE CHART

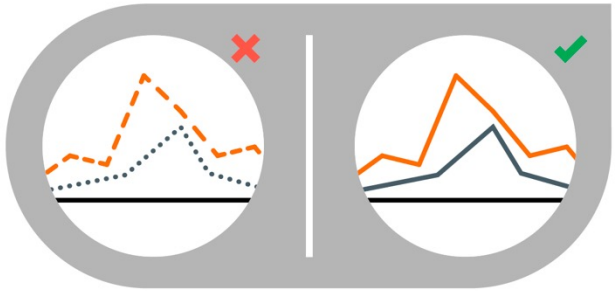
DESIGN BEST PRACTICES



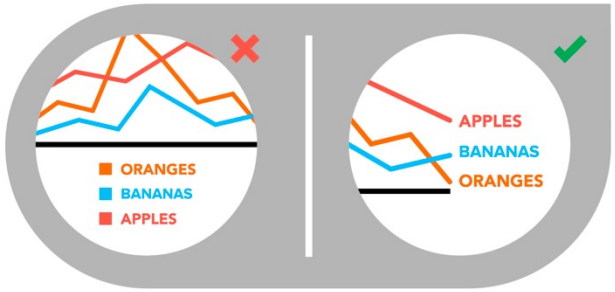
INCLUDE A ZERO BASELINE IF POSSIBLE
Although a line chart does not have to start at a zero baseline, it should be included if possible. If relatively small fluctuations in data are meaningful (e.g., in stock market data), you may truncate the scale to showcase these variances.



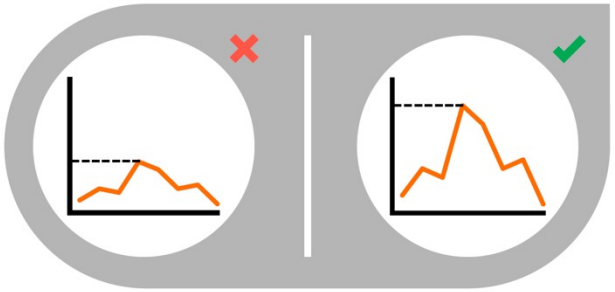
DON'T PLOT MORE THAN 4 LINES
If you need to display more, break them out into separate charts for better comparison.



USE SOLID LINES ONLY
Dashed and dotted lines can be distracting.



LABEL THE LINES DIRECTLY
This lets readers quickly identify lines and corresponding labels instead of referencing a legend.



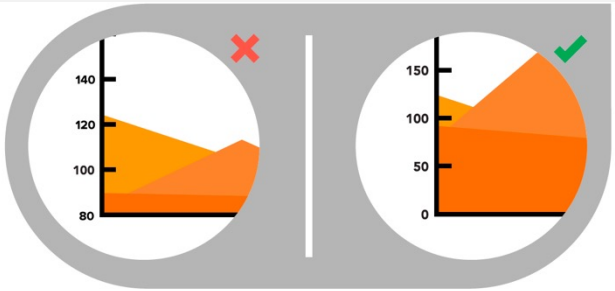
USE THE RIGHT HEIGHT
Plot all data points so that the line chart takes up approximately two-thirds of the y-axis' total scale.

Credit by *Data Visualization 101: How To Design Charts and Graphs, by HubSpot and Visage*

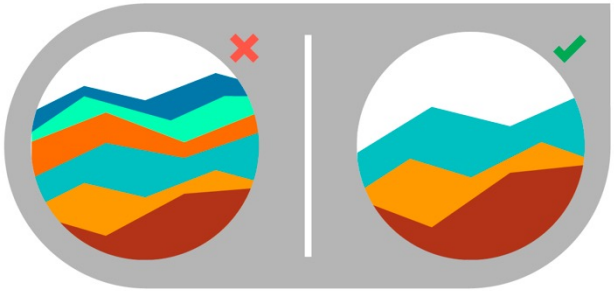
AREA CHART

DESIGN BEST PRACTICES

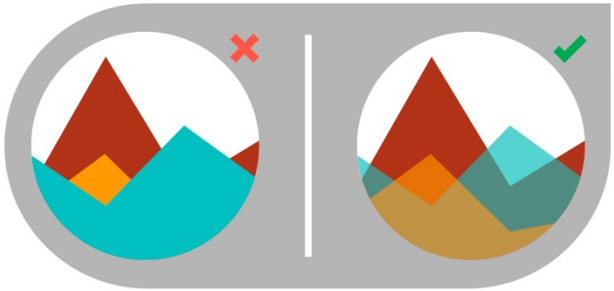
MAKE IT EASY TO READ
In stacked area charts, arrange data to position categories with highly variable data on the top of the chart and low variability on the bottom.



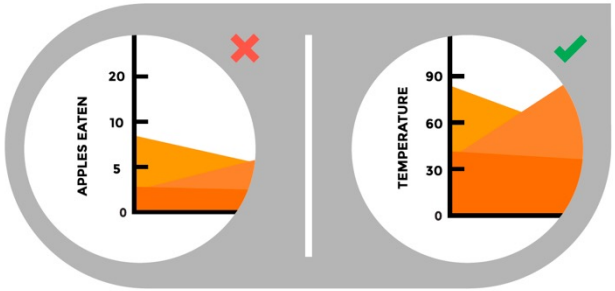
START Y-AXIS VALUE AT 0
Starting the axis above zero truncates the visualization of values.



DON'T DISPLAY MORE THAN 4 DATA CATEGORIES
Too many will result in a cluttered visual that is difficult to decipher.



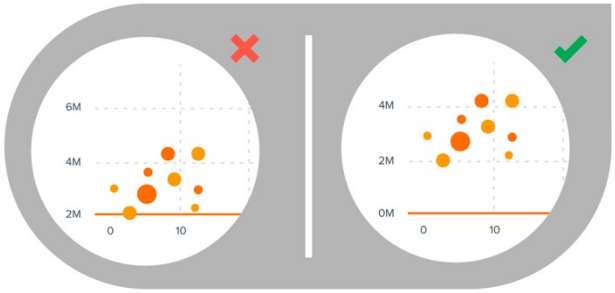
USE TRANSPARENT COLORS
In standard area charts, ensure data isn't obscured in the background by ordering thoughtfully and using transparency.



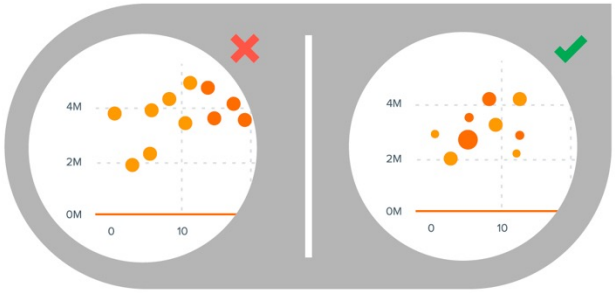
DON'T USE AREA CHARTS TO DISPLAY DISCRETE DATA
The connected lines imply intermediate values, which only exist with continuous data.

SCATTER PLOT

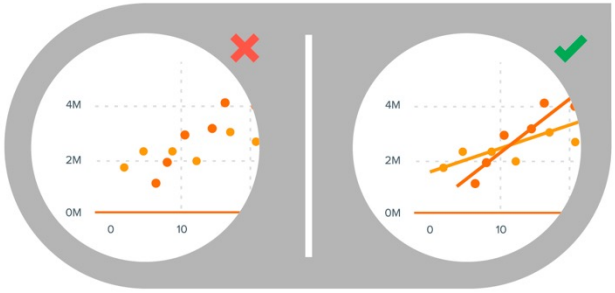
DESIGN BEST PRACTICES



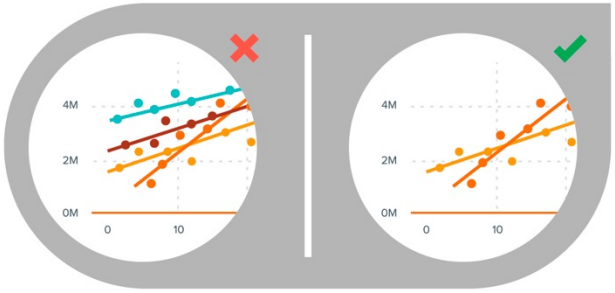
START Y-AXIS VALUE AT 0
Starting the axis above zero truncates the visualization of values.



INCLUDE MORE VARIABLES
Use size and dot color to encode additional data variables.

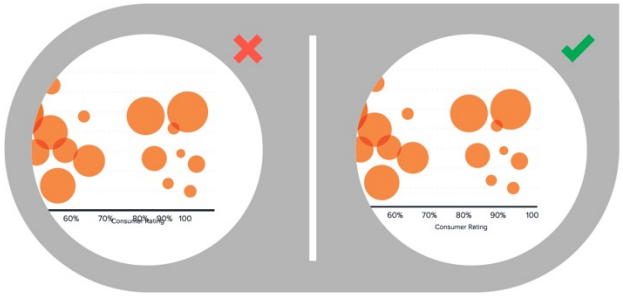


USE TREND LINES
These help draw correlation between the variables to show trends.

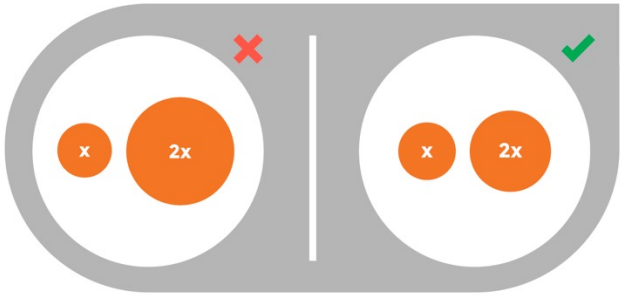


DON'T COMPARE MORE THAN 2 TREND LINES
Too many lines make data difficult to interpret.

Credit by *Data Visualization 101: How To Design Charts and Graphs, by HubSpot and Visage*



MAKE SURE LABELS ARE VISIBLE
All labels should be unobstructed and easily identified with the corresponding bubble.



SIZE BUBBLES APPROPRIATELY
Bubbles should be scaled according to area, not diameter.

DON'T USE ODD SHAPES
Avoid adding too much detail or using shapes that are not entirely circular; this can lead to inaccuracies.

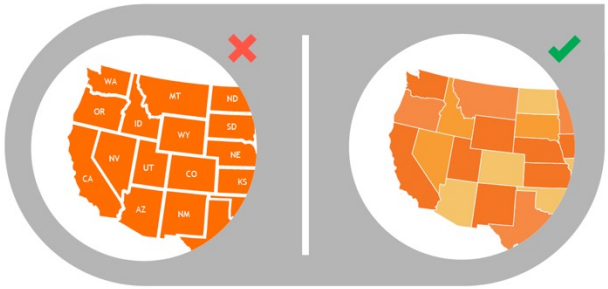
BUBBLE CHART

DESIGN BEST PRACTICES

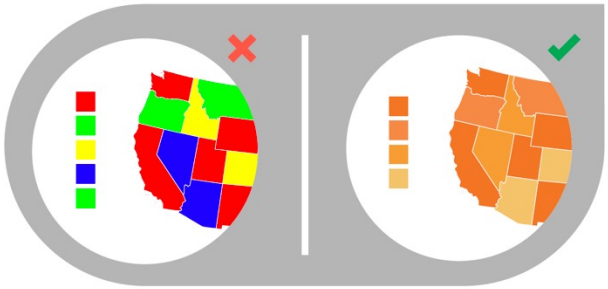
Credit by *Data Visualization 101: How To Design Charts and Graphs, by HubSpot and Visage*

HEAT MAP

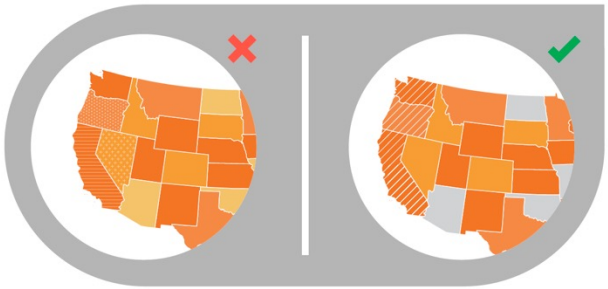
DESIGN BEST PRACTICES



USE A SIMPLE MAP OUTLINE
These lines are meant to frame the data, not distract.



SELECT COLORS APPROPRIATELY
Some colors stand out more than others, giving unnecessary weight to that data. Instead, use a single color with varying shade or a spectrum between two analogous colors to show intensity. Also remember to intuitively code color intensity according to values.



USE PATTERNS SPARINGLY
A pattern overlay that indicates a second variable is acceptable, but using multiple is overwhelming and distracting.

CHOOSE APPROPRIATE DATA RANGES
Select 3-5 numerical ranges that enable fairly even distribution of data between them. Use +/- signs to extend high and low ranges.

10 DATA DESIGN DOS AND DON'TS

Designing your data doesn't have to be overwhelming. With a basic understanding of how different data sets should be visualized, along with a few fundamental design tips and best practices, you can create more accurate, more effective data visualizations. Follow these 10 tips to ensure your design does your data justice.

1 | DO USE ONE COLOR TO REPRESENT EACH CATEGORY.

2 | DO ORDER DATA SETS USING LOGICAL HEIRARCHY.

3 | DO USE CALLOUTS TO HIGHLIGHT IMPORTANT OR INTERESTING INFORMATION.

4 | DO VISUALIZE DATA IN A WAY THAT IS EASY FOR READERS TO COMPARE VALUES.

5 | DO USE ICONS TO ENHANCE COMPREHENSION AND REDUCE UNNECESSARY LABELING.

6 | DON'T USE HIGH CONTRAST COLOR COMBINATIONS SUCH AS RED/GREEN OR BLUE/YELLOW.

7 | DON'T USE 3D CHARTS. THEY CAN SKEW PERCEPTION OF THE VISUALIZATION.

8 | DON'T ADD CHART JUNK. UNNECESSARY ILLUSTRATIONS, DROP SHADOWS, OR ORNAMENTATIONS DISTRACT FROM THE DATA.

9 | DON'T USE MORE THAN 6 COLORS IN A SINGLE LAYOUT.

10 | DON'T USE DISTRACTING FONTS OR ELEMENTS (SUCH AS BOLD, ITALIC, OR UNDERLINED TEXT).

Credit by *Data Visualization 101: How To Design Charts and Graphs, by HubSpot and Visage*

Graphical Capabilities

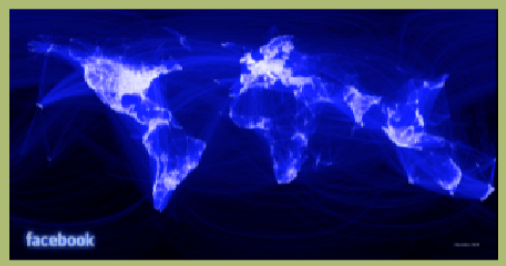
A picture says more than a thousand words

Visualized data can be understood more efficiently and effectively than the raw numbers alone.

R + visualization
= perfect match

ggplot2	To make pretty graphs, including the opportunity to use grammar of graphics to create layered, customizable plots
lattice	To easily display multivariate relationships
rCharts	To create, customize and publish interactive javascript visualizations from R
googleVis	To use Google Chart tools to visualize data in R
ggvis	To implement interactive grammar of graphics, while rendering in a web browser

e.g.: Visualizing Facebook friends with R



Credit by https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis

ggplot2

ggplot2

- http://ggplot2.org/
- one of the best static visualization packages in *R*
- a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

ggplot2 call

- ggplot.R
- Which one will you use?

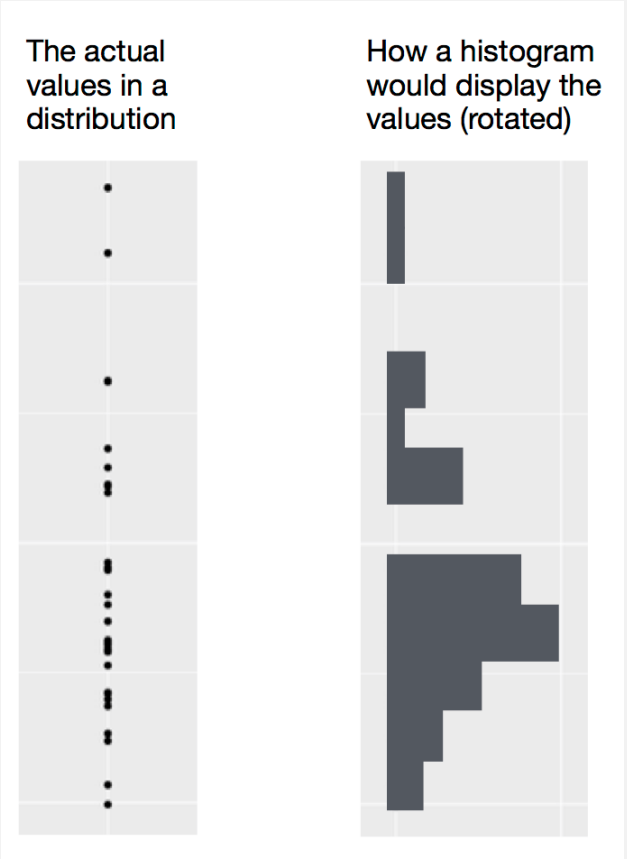
named arguments

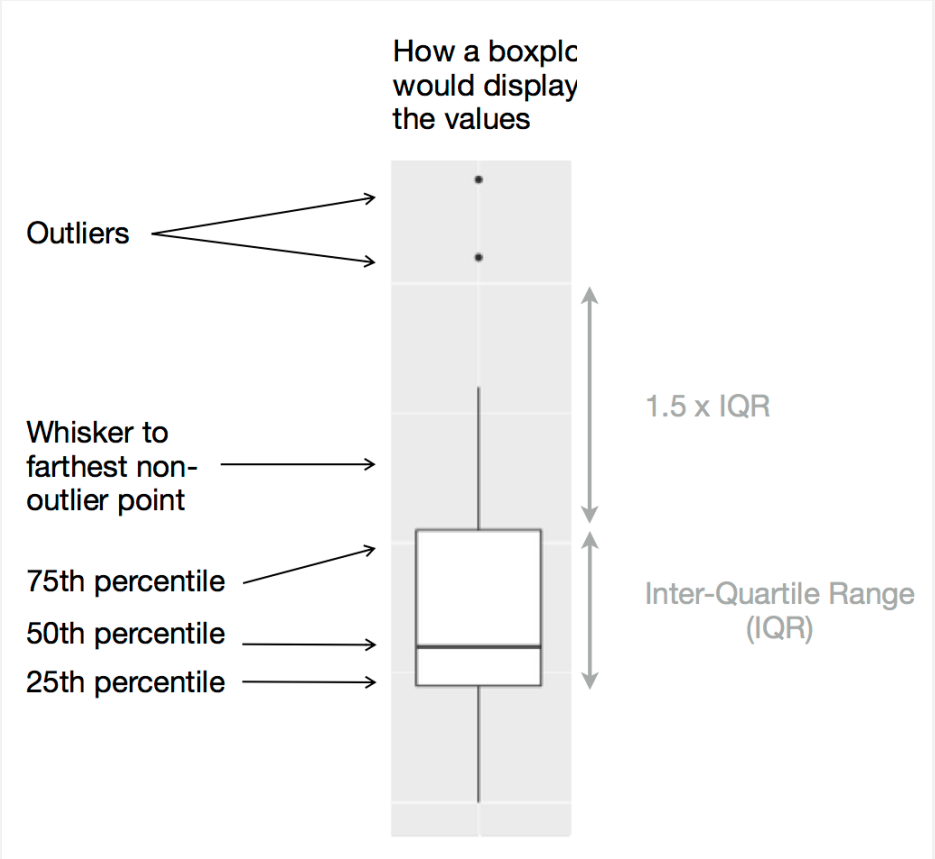
```
ggplot(data = faithful, mapping = aes(x =
eruptions)) + geom_freqpoly(binwidth = 0.25)

ggplot(faithful, aes(eruptions)) +
geom_freqpoly(binwidth = 0.25)
```

Box Plot

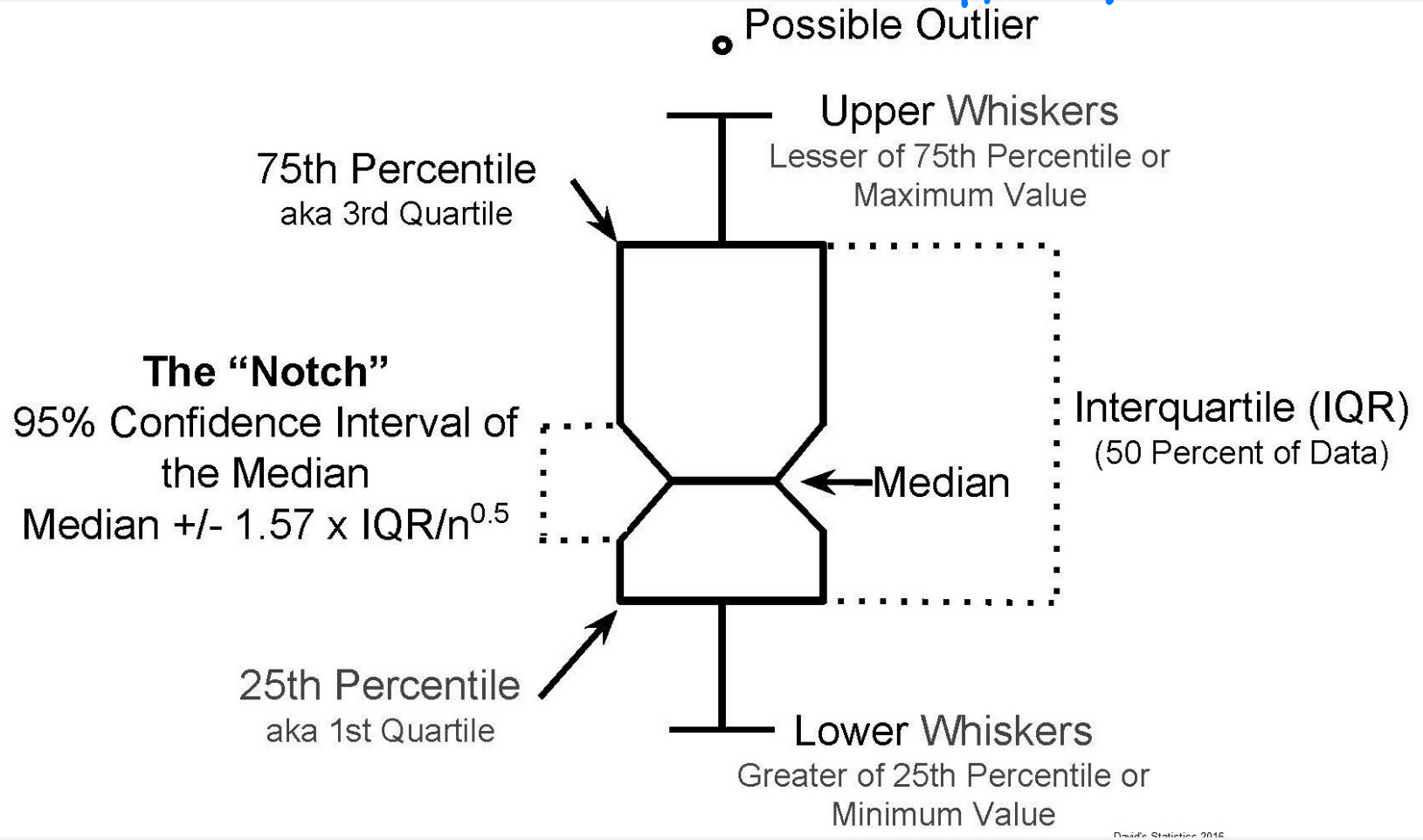
Boxplot


The actual values in a distribution
How a histogram would display the values (rotated)


How a boxplc would display the values
Outliers
Whisker to farthest non-outlier point
75th percentile
50th percentile
25th percentile
1.5 x IQR
Inter-Quartile Range (IQR)

Notch

Q1: first quartile / lower quartile
Q2: second quartile / median
Q3: third quartile / upper quartile



boxplot

- boxplot.R

```
ggplot(data = mpg, mapping = aes(x = class, y =
hwy)) +  geom_boxplot()
```

  - What is problem? no ordering

```
ggplot(data = mpg) + geom_boxplot(mapping = aes(x
= reorder(class, hwy, FUN = median), y = hwy))
```

  - What is problem? x labels overlap

```
ggplot(data = mpg) + geom_boxplot(mapping = aes(x =
reorder(class, hwy, FUN = median), y = hwy)) +
coord_flip()
```

Two variables?

Visually checking relationships between two variables

- Is there a relationship
  - between the *two inputs* age and income in my data?
  - between the *input* marital status and the *output* health
- What kind of relationship, and how strong?

Scatter Plots

Two numerical variables

Two continuous variables

- Relation between carat & price
  - `ggplot(data = diamonds) + geom_point(mapping = aes(x = carat, y = price))`
    - What is problem? overplotting
  - `ggplot(data = diamonds) + geom_point(mapping = aes(x = carat, y = price), alpha = 1 / 100)`
    - What is problem? diamonds
  - `ggplot(data = ~~smaller~~) + geom_bin2d(mapping = aes(x = carat, y = price))`

adding transparency

visualizing density of points

Two continuous variables

- bin one continuous variable into a categorical variable

```
ggplot(data = smaller, mapping = aes(x = carat,
y = price)) + geom_boxplot(mapping = aes(group =
cut_width(carat, 0.1)))
```

diamonds

Line Plots

- relation.R

```
x <- runif(100)
y <- x^2 + 0.2*x
ggplot(data.frame(x=x,y=y), aes(x=x,y=y)) + geom_line()
```

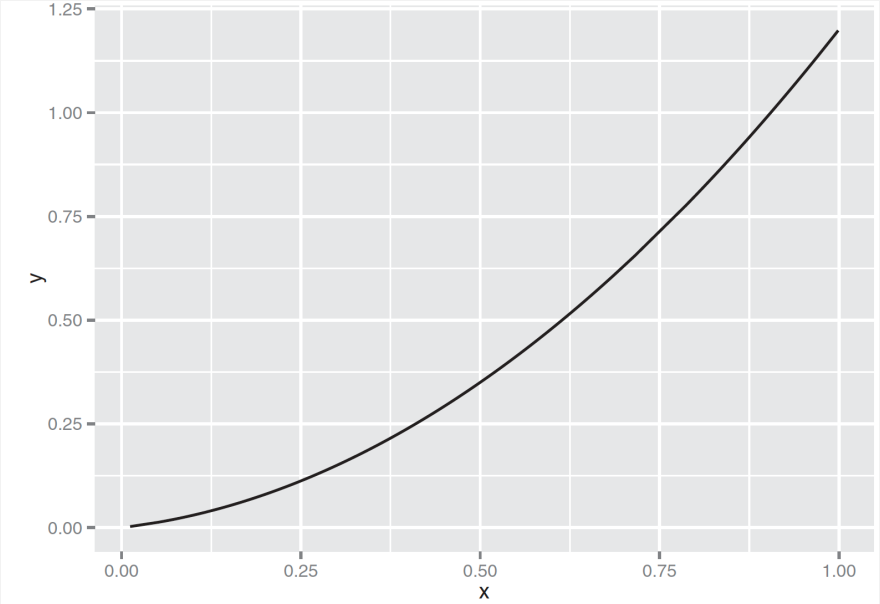


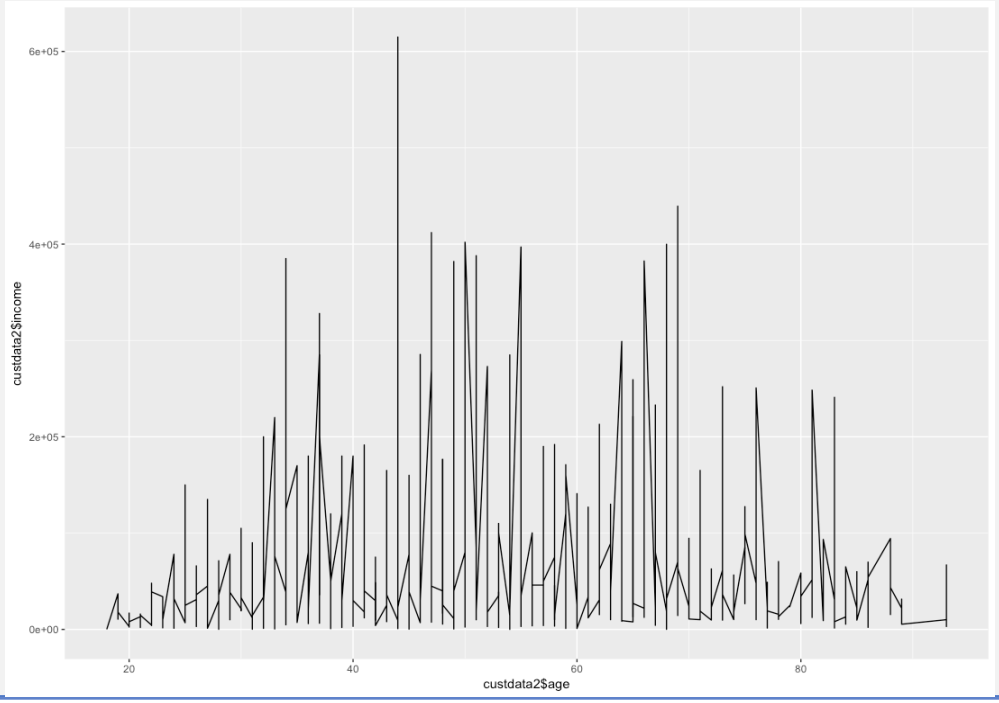
Figure3.9, *Practical Data Science with R* by Nina Zumel, John Mount

Line Plot does not work

- When the data is not so cleanly related, line plots aren't as useful!

```
custdata2 <- subset(custdata, (custdata$age > 0 & custdata$age < 100 &
custdata$income > 0))

ggplot(custdata2, aes(x=custdata2$age, y=custdata2$income)) +
geom_line()
```



Scatter Plots

```
ggplot(custdata2, aes(x=age, y=income)) +
geom_point() + ylim(0, 200000)
```

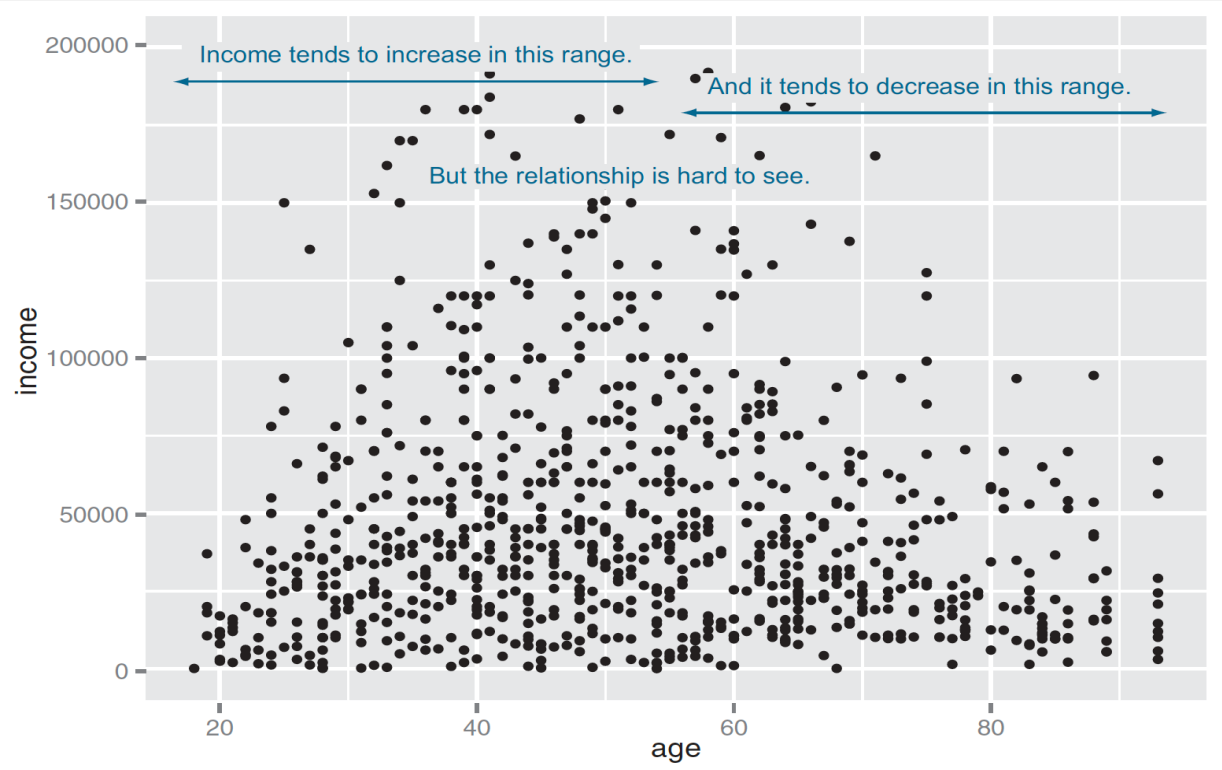


Figure3.10, *Practical Data Science with R* by Nina Zumel, John Mount

Linear fit

assumes linear relationship

```
ggplot(custdata2, aes(x=age, y=income)) + geom_point() +
stat_smooth(method="lm") + ylim(0, 200000)
cor(custdata2$age, custdata2$income)
```

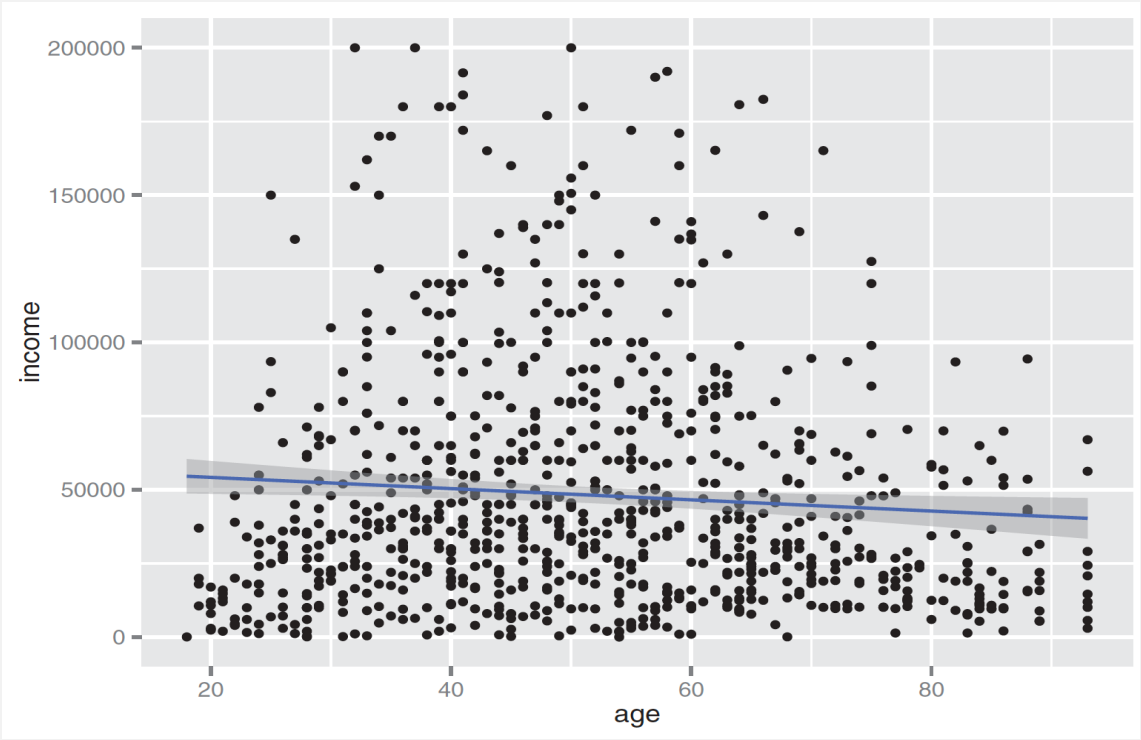


Figure3.11, *Practical Data Science with R* by Nina Zumel, John Mount

Smoothing curve

LOcally
Estimated
Scatterplot
Smoothing

- fit using the loess (or lowess) functions, which calculate smoothed local linear fits of the data.

```
ggplot(custdata2, aes(x=age, y=income)) + geom_point() +
geom_smooth() + ylim(0, 200000)
```

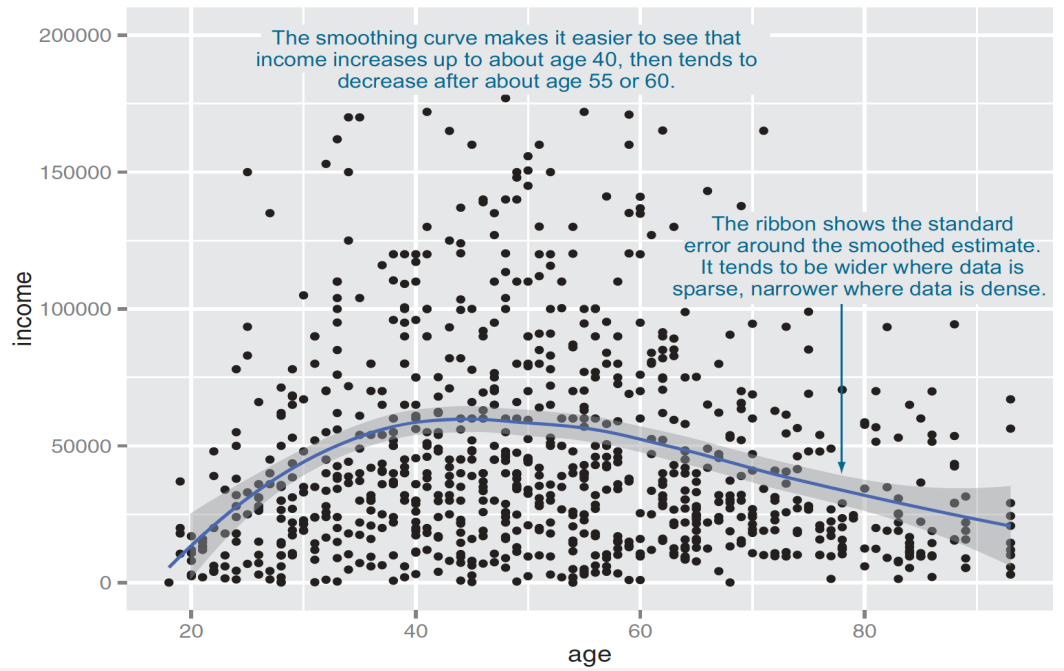


Figure3.11, *Practical Data Science with R* by Nina Zumel, John Mount

High-volume situations => HEXBIN PLOTS

```
library(hexbin)
ggplot(custdata2, aes(x=age, y=income)) +
geom_hex(binwidth=c(5, 10000)) +
geom_smooth(color="white", se=F) +  ylim(0,200000)
```

Parameters of `geom_smooth()`:

1. `color`: This parameter specifies the color of the smoothing line. In your plot, `color="white"` means the smoothing curve is white, making it stand out against the dark background of the hexbins.
2. `se`: The `se` parameter stands for "standard error," and it controls whether a confidence interval around the smooth line is displayed. Setting `se=F` (or `se=FALSE`) means that the confidence interval is not shown. This makes the plot cleaner, especially when the main interest is in the trend rather than the uncertainty around the trend estimates.

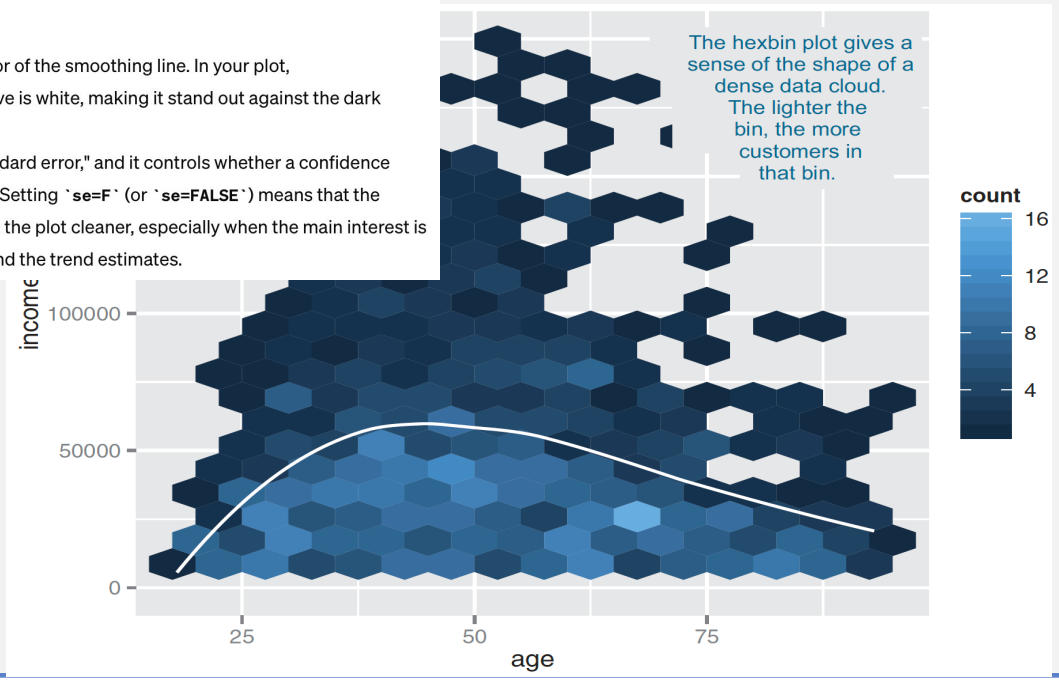


Figure3.14, *Practical Data Science with R* by Nina Zumel, John Mount

Smoothing curve for boolean variables

```
ggplot(custdata2, aes(x=age, y=as.numeric(health.ins))) +
geom_point(position=position_jitter(w=0.05, h=0.05)) +
geom_smooth()
```

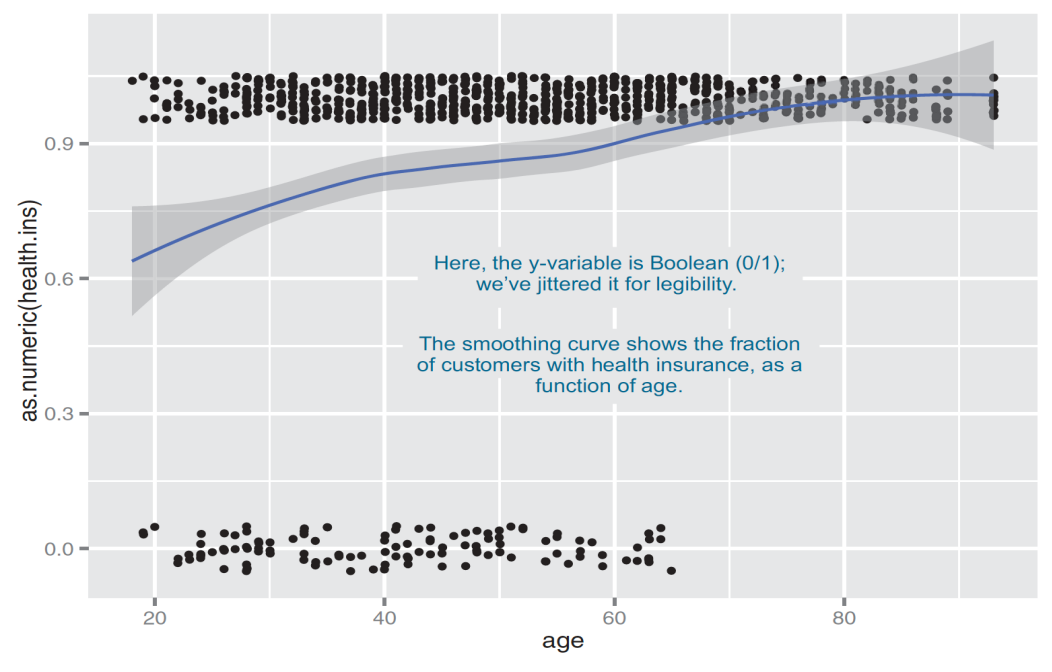


Figure3.13 ,*Practical Data Science with R* by Nina Zumel, John Mount

Plot Elements:

1. **Data and Aesthetics (`aes`)**:
   - `x=age`: Age is used as the x-axis.
   - `y=as.numeric(health.ins)`: The Boolean variable `health.ins` is converted to numeric, where `0` and `1` likely represent the absence or presence of health insurance, respectively.
2. **Jittering Points (`geom_point`)**:
   - `position=position_jitter(w=0.05, h=0.05)`: This adds random variation to the placement of the points to prevent overplotting where data points for `0` and `1` values might otherwise overlap completely. The `w` (width) and `h` (height) parameters control the amount of jitter in horizontal and vertical directions, respectively.
3. **Smoothing Curve (`geom_smooth`)**:
   - This function automatically adds a smoothing curve to the plot, which helps in understanding the trend or pattern in the data over age. The default method for `geom_smooth` is typically a LOESS (locally estimated scatterplot smoothing) curve unless the dataset is very large, in which case it defaults to using a generalized additive model (GAM).

Bar chart

Two categorical variables

Two categorical variables

- twoVariable.R
- Relation between cut & color
  - By count

```
ggplot(data = diamonds) + geom_count(mapping = aes(x =
cut, y = color))
```

  - By heatmap

```
diamonds %>% count(color, cut) %>% ggplot(mapping = aes(x
= color, y = cut)) + geom_tile(mapping = aes(fill = n))
```

1. `diamonds %>% count(color, cut)`:
   - `diamonds`: This is the dataset being used, which contains data on diamonds including attributes like color and cut.
   - `count(color, cut)`: This `dplyr` function counts the number of occurrences of each combination of `color` and `cut` in the diamonds dataset. It essentially summarizes the data, creating a new data frame where each row represents a unique combination of `color` and `cut`, and includes a new column `n` which contains the counts of each combination.
   - `%>%`: This operator takes the `diamonds` dataset and passes it as the first argument to the `count` function. The result is a new data frame with the count of each combination of `color` and `cut`.
2. `%>% ggplot(mapping = aes(x = color, y = cut))`:
   - The result from `count(color, cut)` (a data frame with columns `color`, `cut`, and `n`) is passed to the `ggplot` function.
   - `ggplot(mapping = aes(x = color, y = cut))`: This initializes a ggplot object, specifying how data should be mapped to aesthetic attributes of the plot. Here, `color` is mapped to the x-axis and `cut` to the y-axis.
3. `+ geom_tile(mapping = aes(fill = n))`:
   - After initializing the ggplot object, `geom_tile` is added to create a tile layer. The `fill` aesthetic is mapped to `n`, the count of diamonds for each `color` and `cut` combination, which colors each tile based on the number of diamonds in that category.

Stacked bar chart for two categorical variables

```
ggplot(custdata) + geom_bar(aes(x=marital.stat,
    fill=health.ins))
```

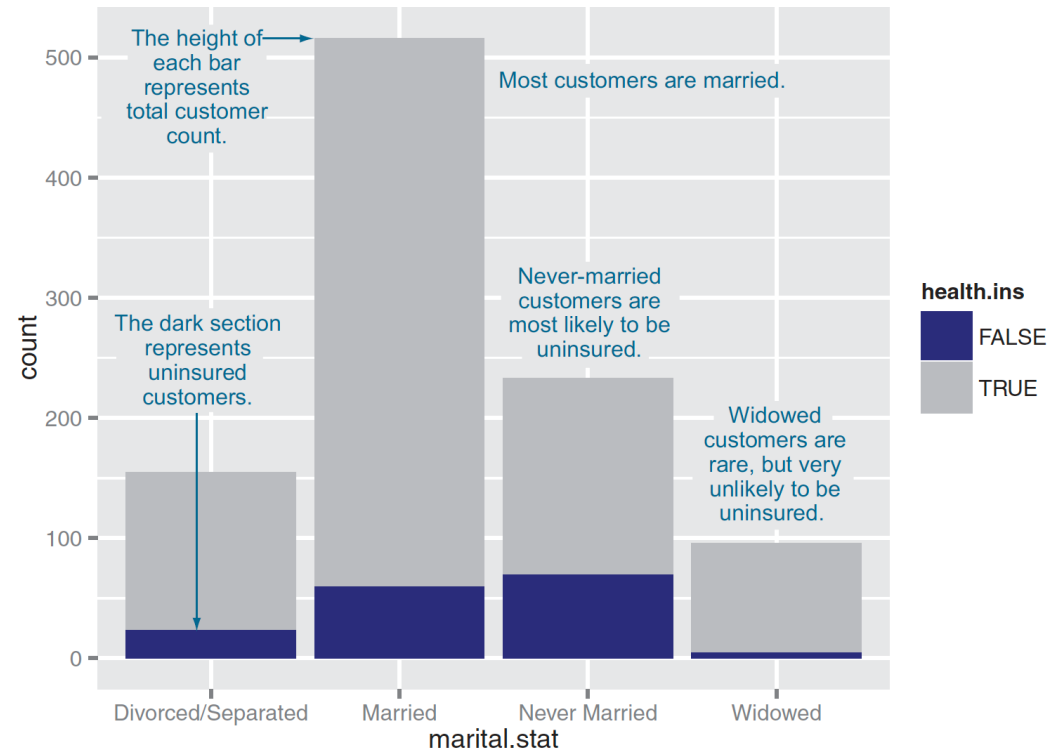


Figure3.15, *Practical Data Science with R* by Nina Zumel, John Mount

Side-by-side bar chart

```
ggplot(custdata) + geom_bar(aes(x=marital.stat,
    fill=health.ins),
    position="dodge")
```

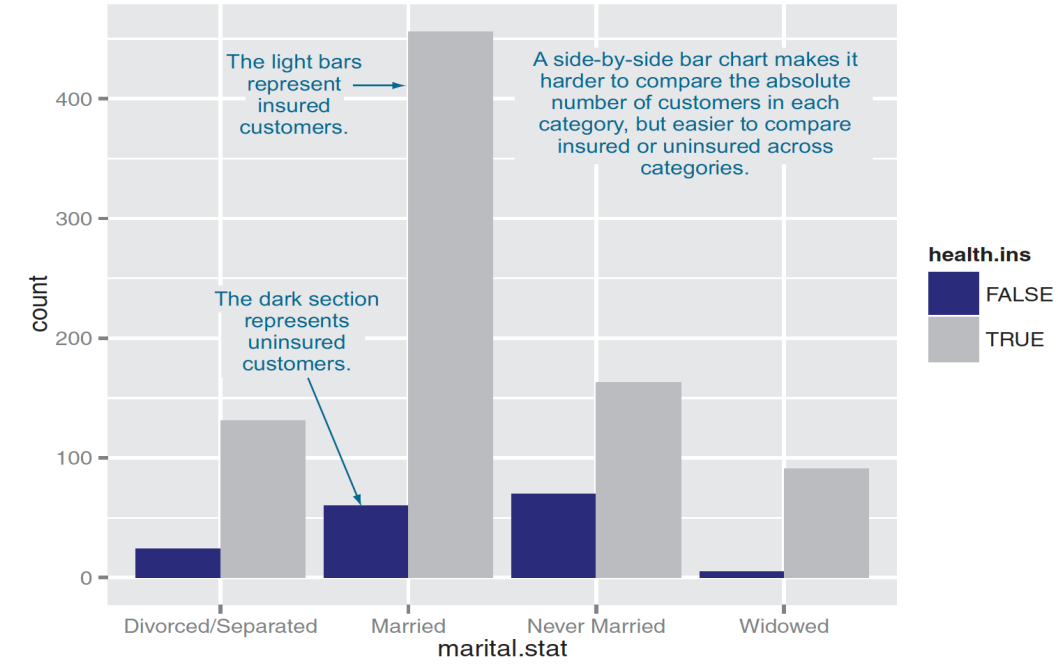
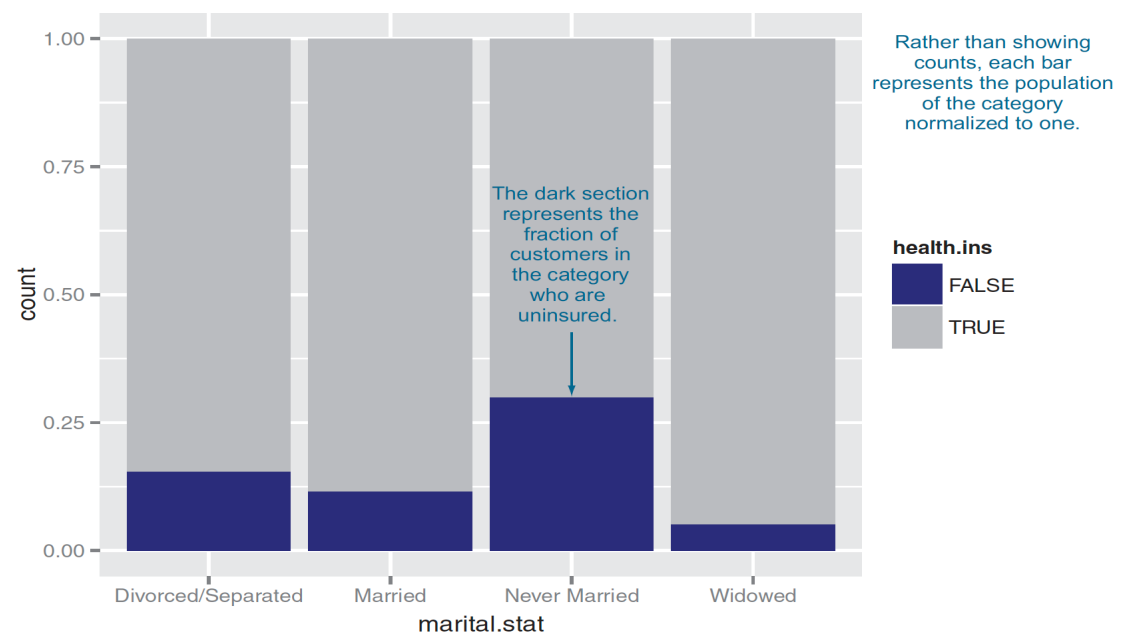


Figure3.16, *Practical Data Science with R* by Nina Zumel, John Mount

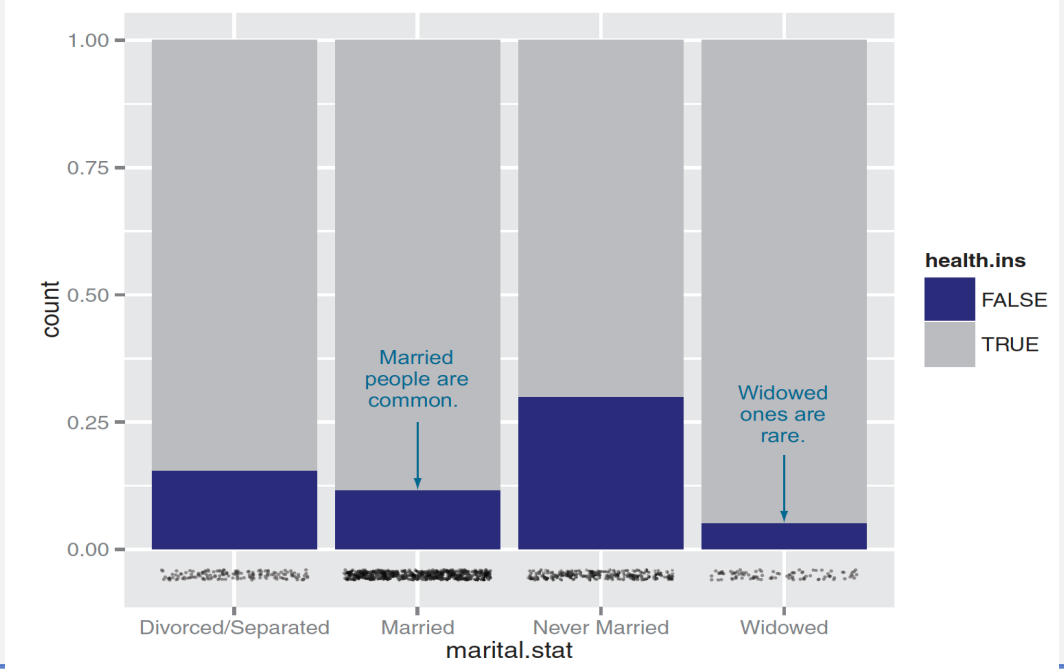
Filled bar chart

```
ggplot(custdata) + geom_bar(aes(x=marital.stat,
    fill=health.ins),
    position="fill")
```



Plotting data with a rug

```
ggplot(custdata, aes(x=marital.stat)) +
   geom_bar(aes(fill=health.ins), position="fill") +
   geom_point(aes(y=-0.05), size=0.75, alpha=0.3,
   position=position_jitter(h=0.01))
```



Bar Chart with a Rug Plot

Features:

- Combines `geom_bar()` with `geom_point()` to add a "rug" at the bottom.
- The bar portion uses the same aesthetic for marital status and insurance status but does not employ normalization (`position="fill"` is absent here).
- `geom_point()` adds individual data points near the base of each bar. These points are jittered horizontally to minimize overlap, using `position_jitter()`.

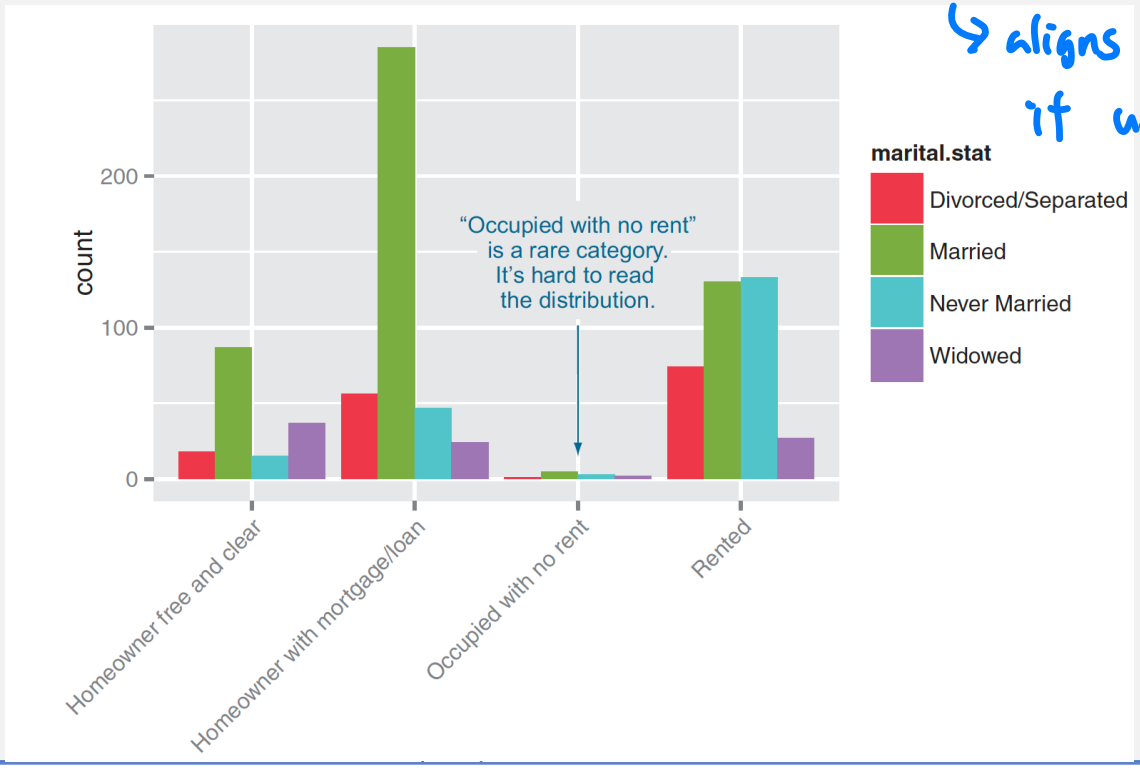
Purpose:

- The bars show the absolute counts of individuals in each category, giving an idea of the raw numbers.
- The rug (points at the base) adds an additional layer of granularity, illustrating the density of data points within each category.
- This visualization is more about showing the actual data distribution, providing a sense of both the counts and the distribution density.

Plotting a bar chart without facets

```
ggplot(custdata2)+
   geom_bar(aes(x=housing.type, fill=marital.stat ), position="dodge")+
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

↳ aligns the text to make it uniformly readable



Plotting a bar chart with facets

```
ggplot(custdata2) + geom_bar(aes(x=marital.stat),
position="dodge", fill="darkgray") +
facet_wrap(~housing.type, scales="free_y") +
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

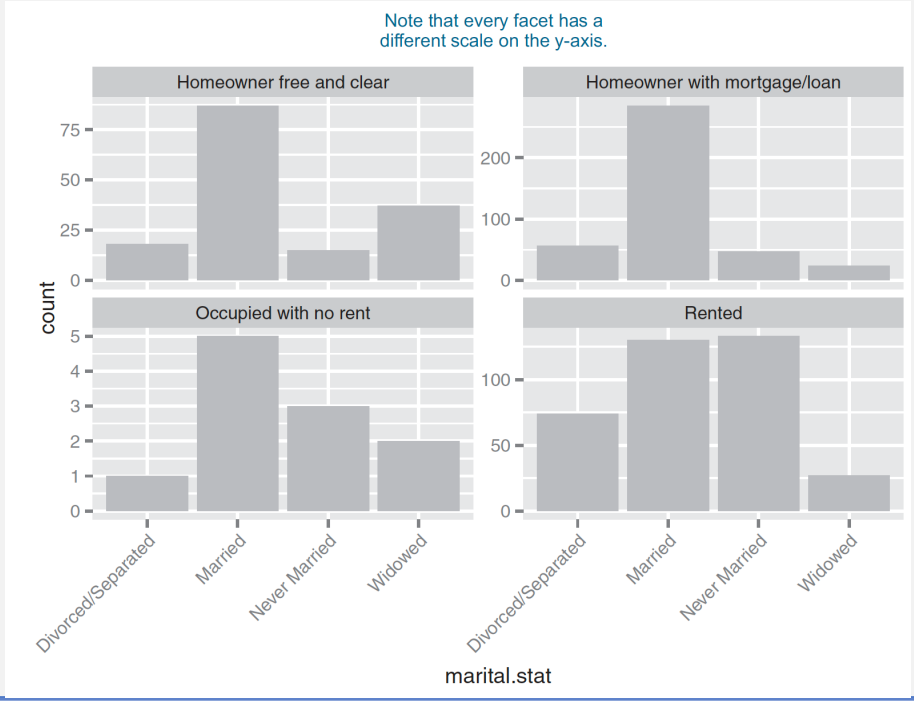


Figure3.20, *Practical Data Science with R* by Nina Zumel, John Mount

Syntax and Parameters:

- `facet_wrap(~housing.type, scales="free_y")`:
  - `~housing.type`: This tells `facet_wrap()` to create a new panel for each unique value in the `housing.type` variable. Each panel will contain a plot that is filtered to only show data from that specific category of housing.
  - `scales="free_y"`: This parameter allows each panel to have its own y-axis scale. It means the y-axis is adjusted to fit the data range of each panel independently, which can be useful when different subsets of data have vastly different ranges. It ensures that each plot is scaled in a way that best presents its data, improving readability and emphasizing differences or similarities among the categories.

Benefits of Using `facet_wrap()`:

1. **Comparative Analysis**: Faceting makes it easier to compare the distributions of a variable across different subgroups. In your chart, viewers can quickly compare the distribution of marital statuses across different housing types.
2. **Clarity and Organization**: By separating the data into distinct panels, `facet_wrap()` avoids cluttering and makes the plots clearer and more digestible. Each facet acts as a standalone plot, tied together by a common theme.
3. **Customizable Layouts**: `facet_wrap()` can wrap a 1D ribbon of panels into a 2D layout, and it automatically determines the grid layout that best fits the available space. The function also has parameters to control the number of rows or columns.

Visualizations for two variables

Graph type	Uses
Line plot	Shows the relationship between two continuous variables. Best when that relationship is functional, or nearly so.
Scatter plot	Shows the relationship between two continuous variables. Best when the relationship is too loose or cloud-like to be easily seen on a line plot.
Smoothing curve	Shows underlying "average" relationship, or trend, between two continuous variables. Can also be used to show the relationship between a continuous and a binary or Boolean variable: the fraction of *true* values of the discrete variable as a function of the continuous variable.
Hexbin plot	Shows the relationship between two continuous variables when the data is very dense.
Stacked bar chart	Shows the relationship between two categorical variables (`var1` and `var2`). Highlights the frequencies of each value of `var1`.
Side-by-side bar chart	Shows the relationship between two categorical variables (`var1` and `var2`). Good for comparing the frequencies of each value of `var2` across the values of `var1`. Works best when `var2` is binary.
Filled bar chart	Shows the relationship between two categorical variables (`var1` and `var2`). Good for comparing the relative frequencies of each value of `var2` within each value of `var1`. Works best when `var2` is binary.
Bar chart with faceting	Shows the relationship between two categorical variables (`var1` and `var2`). Best for comparing the relative frequencies of each value of `var2` within each value of `var1` when `var2` takes on more than two values.

Interactive visualization

ggvis

- http://ggvis.rstudio.com/
- interactive plots from the makers of ggplot2
- a data visualization package for *R* which lets you:
  - Declaratively describe data graphics with a syntax similar in spirit to ggplot2.
  - Create rich interactive graphics that you can play with locally in Rstudio or in your browser.
  - Leverage shiny's infrastructure to publish interactive graphics usable from any browser (either within your company or to the world).

ggvis

- The goal is to combine
  - the best of R (e.g. every modelling function you can imagine)
  - the best of the web (everyone has a web browser).
- Data manipulation and transformation are done in *R*, and the graphics are rendered in a web browser, using Vega.
- For RStudio users, ggvis graphics display in a viewer panel, which is possible because RStudio is a web browser.

Install necessary packages & Load packages

- ggvis.R

```
## Install necessary packages
install.packages("devtools")
library("devtools")
install.packages("ggvis")
# Load packages
library("ggvis")
library(ggplot2)
```

Prepare data

```
# Define image sizes
img.width <- 450
img.height <- 300
# Use mtcars data data(mtcars)
mtcars$cyl <- factor(mtcars$cyl)
mtcars$am <- factor(mtcars$am)
# Compute mean mpg per cyl and am
mtcars.mean <- mtcars %>% group_by(cyl, am) %>%
summarise(mpg_mean=mean(mpg)) %>% select(cyl, am, mpg_mean) %>%
ungroup()
```

Histograms

- ggplot

```
hist.ggplot <- ggplot(mtcars, aes(x=mpg)) +
geom_histogram(binwidth=1)
hist.ggplot
```

- ggvis

```
hist.ggvis <- mtcars %>% ggvis(x = ~mpg) %>%
layer_histograms(width=1) %>%
set_options(width = img.width, height =
img.height)
hist.ggvis
```

Why %>%?

- ggvis uses the pipe operator %>% (familiar for dplyr users), replacing the + in ggplot2
- *# convert engine displacment to litres*

```
mtcars %>%
ggvis(x = ~mpg, y = ~disp) %>%
mutate(disp = disp / 61.0237) %>%
layer_points()

mtcars %>%
ggvis(x = ~mpg, y = ~disp) %>%
layer_points()
```

Scatter plots

- ggplot

```
scatter.ggplot <- ggplot(mtcars, aes(x=wt, y=mpg)) + geom_point()
scatter.ggplot
```

- ggvis

```
scatter.ggvis <- mtcars %>% ggvis(x = ~wt, y = ~mpg) %>% layer_points() %>% set_options(width = img.width, height = img.height)
scatter.ggvis
mtcars %>% ggvis(~wt, ~mpg) %>% layer_points() %>% layer_model_predictions(model = "lm", se = TRUE)
```

ggvis

```
p <- ggvis(mtcars, x = ~wt, y = ~mpg)
layer_points(p)
=>
layer_points(ggvis(mtcars, x = ~wt,
y = ~mpg))
=>
mtcars %>% ggvis(x = ~wt, y = ~mpg)
%>% layer_points()
```

Scatter plots by group

- ggplot

```
scatter.ggplot <- ggplot(mtcars,
aes(x=wt, y=mpg, colour=cyl)) +
geom_point()
scatter.ggplot
```

Scatter plots by group

- ggvis

```
scatter.ggvis <- mtcars %>% ggvis(x = ~wt,
y = ~mpg, fill = ~cyl) %>% layer_points()
%>% set_options(width = img.width, height =
img.height)
scatter.ggvis
mtcars %>%
  ggvis(~wt, ~mpg, fill = ~factor(cyl)) %>%
  layer_points() %>%
  group_by(cyl) %>%
  layer_model_predictions(model = "lm", se
= TRUE)
```

Line plots

- ggplot

```
line.ggplot <- ggplot(mtcars.mean,
aes(x=cyl, y=mpg_mean, colour=am)) +
geom_line(aes(group=am))
line.ggplot
```

- ggvis

```
line.ggvis <- mtcars.mean %>% ggvis(x =
~cyl, y = ~mpg_mean, stroke = ~am) %>%
layer_lines() %>% set_options(width =
img.width, height = img.height)
line.ggvis
```

Layers

- Simple, which include primitives like points, lines and rectangles.

```
mtcars %>% ggvis(~wt, ~mpg) %>% layer_points()
```

- Compound, which combine data transformations with one or more simple layers.

```
mtcars %>% ggvis(~mpg) %>% layer_histograms()
```

- Multiple layers

```
mtcars %>% ggvis(~wt, ~mpg) %>% layer_points()
%>% layer_smooths()
```

Interaction: input_*

- `input_checkbox()`: a check-box
- `input_checkboxgroup()`: a group of check boxes
- `input_numeric()`: a spin box
- `input_radiobuttons()`: pick one from a set options
- `input_select()`: create a drop-down text box
- `input_text()`: arbitrary text input

density plot

- `plot(density(mtcars$wt))`
  - The (S3) generic function density computes kernel density estimates. Its default method does so with the given *kernel* and *bandwidth* for univariate observations.
    - Kernel
    - bandwidth

- **Kernel**: The kernel is a function used in the smoothing part of the KDE. It defines the shape of the curve used to generate the density estimation. Common kernels include Gaussian, Epanechnikov, and uniform.
- **Bandwidth**: The bandwidth is a crucial parameter in KDE that affects how smooth the resulting density curve will be. A larger bandwidth leads to a smoother curve, while a smaller bandwidth can capture more detail but may lead to overfitting.

Density plot with `input_slider` + check box

```
mtcars %>% ggvis(x = ~wt) %>%

layer_densities( adjust = input_slider(.1, 2,
value = 1, step = .1, label = "Bandwidth
adjustment"),

kernel = input_select( c("Gaussian" =
"gaussian", "Epanechnikov" = "epanechnikov",
"Rectangular" = "rectangular", "Triangular" =
"triangular", "Biweight" = "biweight",
"Cosine" = "cosine", "Optcosine" =
"optcosine"), label = "Kernel") )
```

Interaction

```
mtcars %>% ggvis(~mpg, ~disp, size = ~vs) %>%
layer_points()

mtcars %>% ggvis(~wt, ~mpg, size := 300, opacity := 0.4)
%>% layer_points()

=>

mtcars %>% ggvis(~wt, ~mpg, size := input_slider(10,
100), opacity := input_slider(0, 1) ) %>% layer_points()

=>

slider <- input_slider(10, 1000)

mtcars %>% ggvis(~wt, ~mpg) %>% layer_points(fill :=
"red", opacity := 0.5, size := slider)
```

Shiny

- an *R* package that makes it easy to build interactive web applications (apps) straight from R.
- http://shiny.rstudio.com/

Credit by http://ggvis.rstudio.com/ggvis-basics.html

Shiny

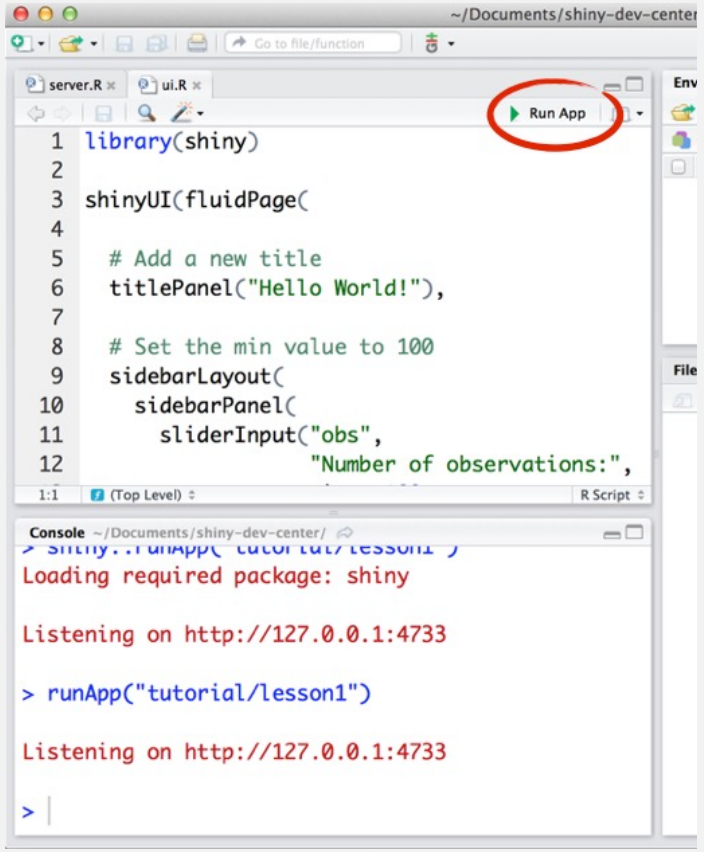
- helloShiny.R

```
install.packages("shiny")
library(shiny)
runExample("01_hello")
```

- Shiny apps have two components:
  - a user-interface script
  - a server script

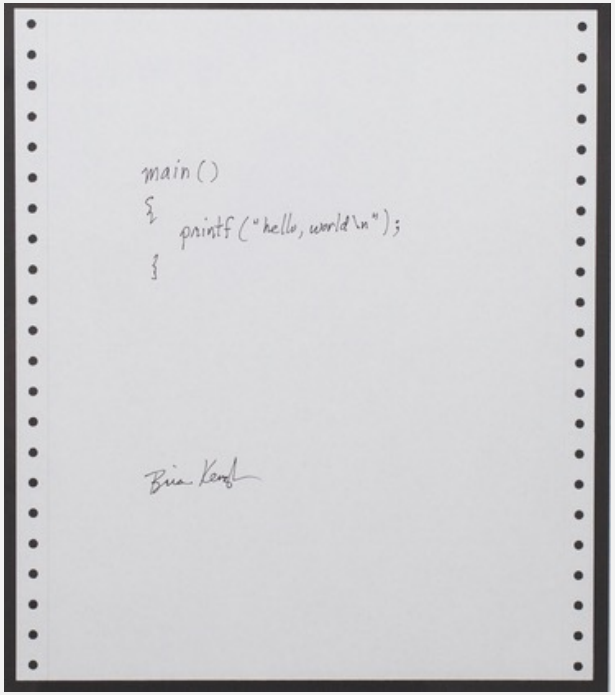
Running an App

- runApp1.R
- Hello World
  - Change the title from “Hello Shiny!” to “Hello World!”.
  - Set the minimum value of the slider bar to 5.
  - Change the histogram color from "darkgray" to "skyblue".
- Relaunching Apps



hello, world!

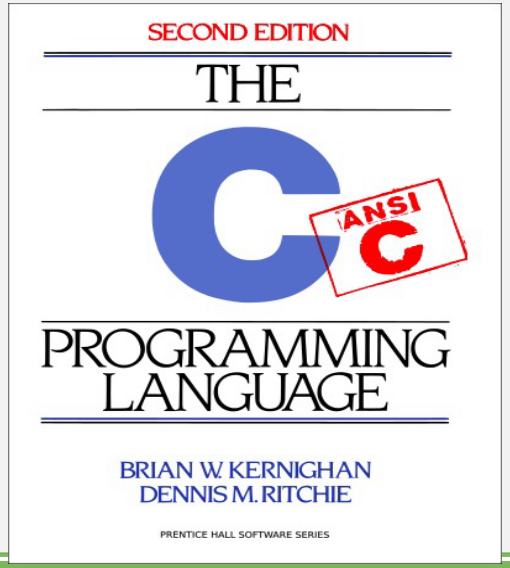
While small test programs have existed since the development of programmable computers, the tradition of using the phrase "Hello, world!" as a test message was influenced by an example program in the seminal book *The C Programming Language*. The example program from that book prints "hello, world" (without capital letters or exclamation mark), and was inherited from a 1974 Bell Laboratories internal memorandum by Brian Kernighan, *Programming in C.A Tutorial.*



Credit by https://en.wikipedia.org/wiki/%22Hello,_World!%22_program

The C Programming Language

- sometimes termed ***K&R***
- computer programming book written by Brian Kernighan and Dennis Ritchie, the latter of whom originally designed and implemented the language.
- The book was central to the development and popularization of the C programming language and is still widely read and used today.

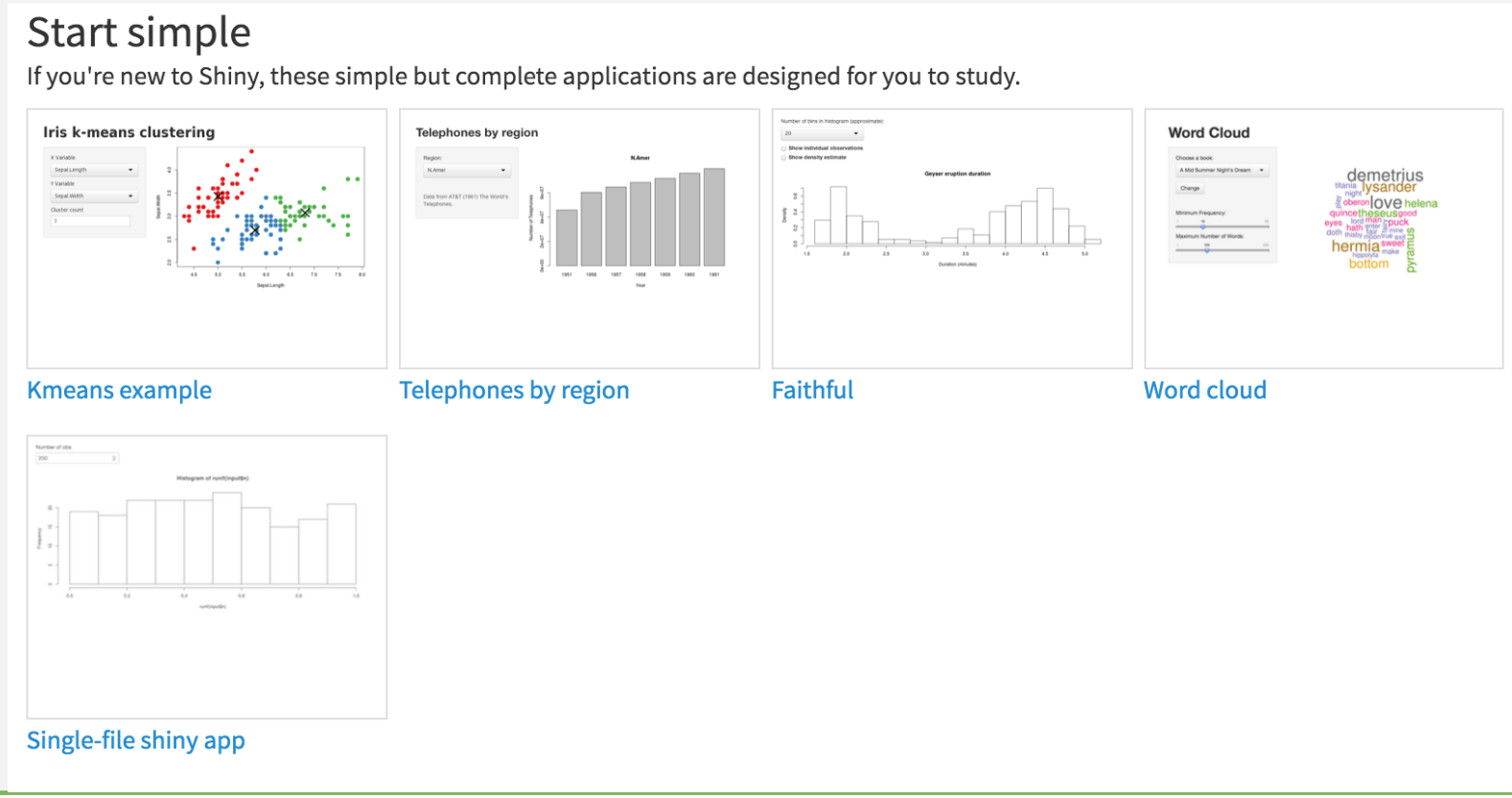


Shiny examples

```
system.file("examples", package="shiny") # Find the full names of files in packages etc.
runExample("01_hello") # a histogram
runExample("02_text") # tables and data frames
runExample("03_reactivity") # a reactive expression
runExample("04_mpg") # global variables
runExample("05_sliders") # slider bars
runExample("06_tabsets") # tabbed panels
runExample("07_widgets") # help text and submit buttons
runExample("08_html") # Shiny app built from HTML
runExample("09_upload") # file upload wizard
runExample("10_download") # file download wizard
runExample("11_timer") # an automated timer
```

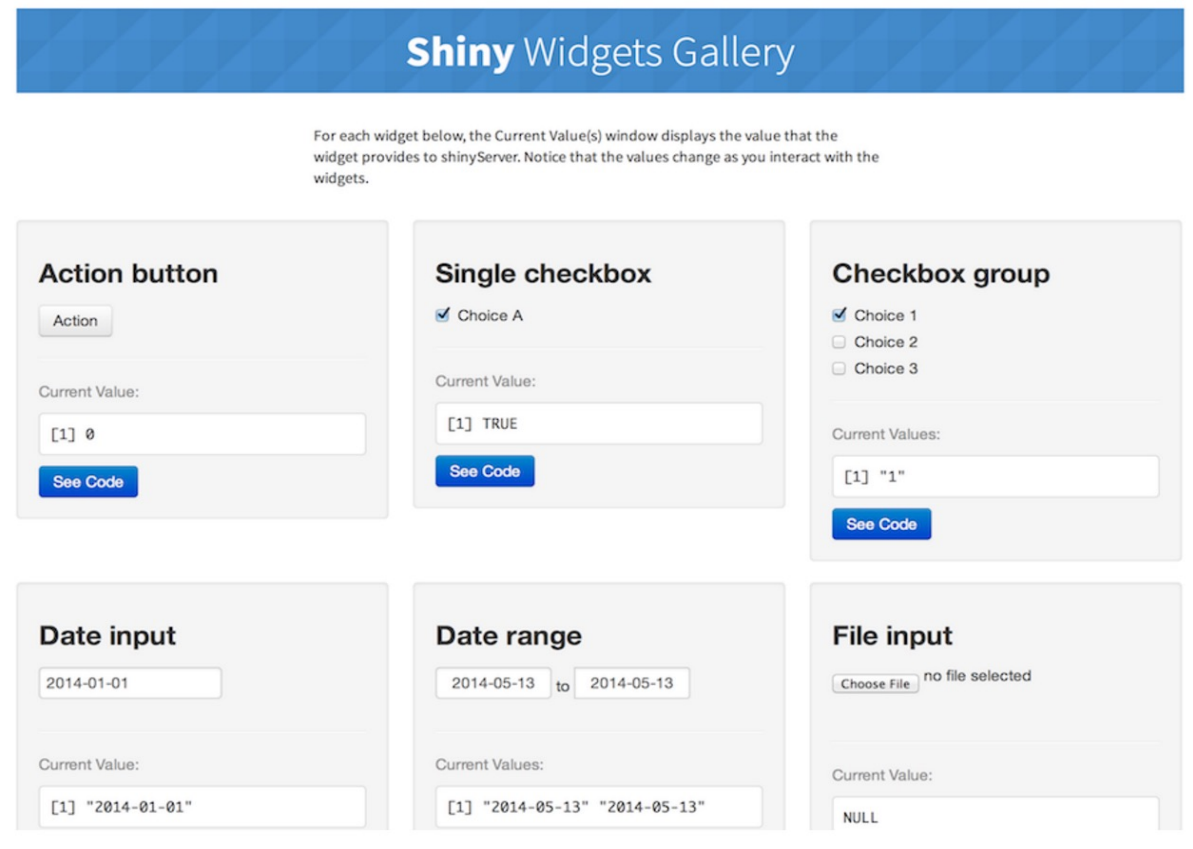
Shiny Gallery

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



Shiny Widgets Gallery

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



Share your apps

- Share your Shiny app as one file with two function
  - server
  - ui
- Share your Shiny app as a web page.

Share your Shiny app as two files

- runUrl

```
runUrl('url')
runUrl('https://github.com/rstudio/shiny_example/archive/master.tar.gz')
```

- runGitHub

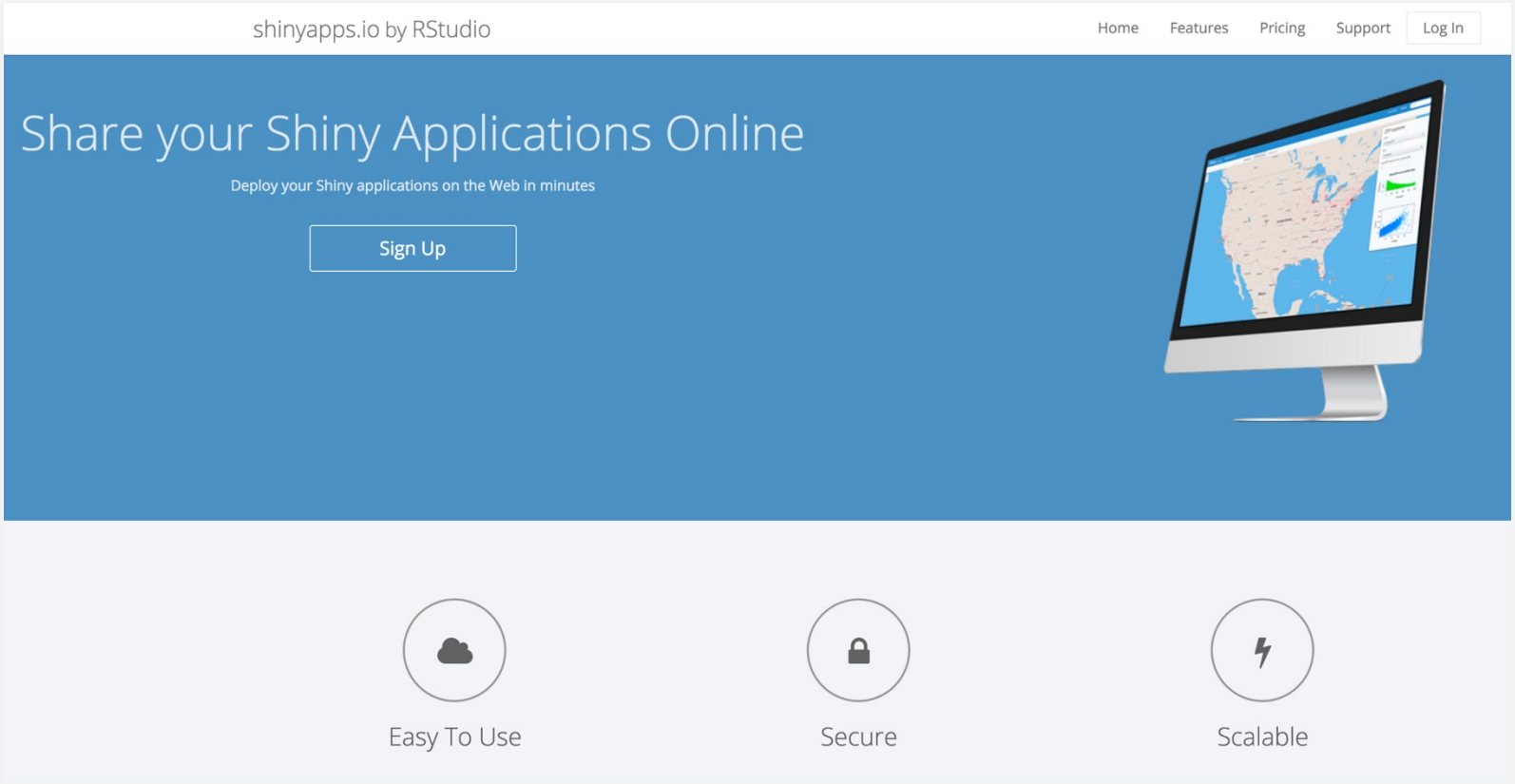
```
runGitHub( "<your repository name>", "<your user name>")
runGitHub("shiny_example", "rstudio")
```

- runGist

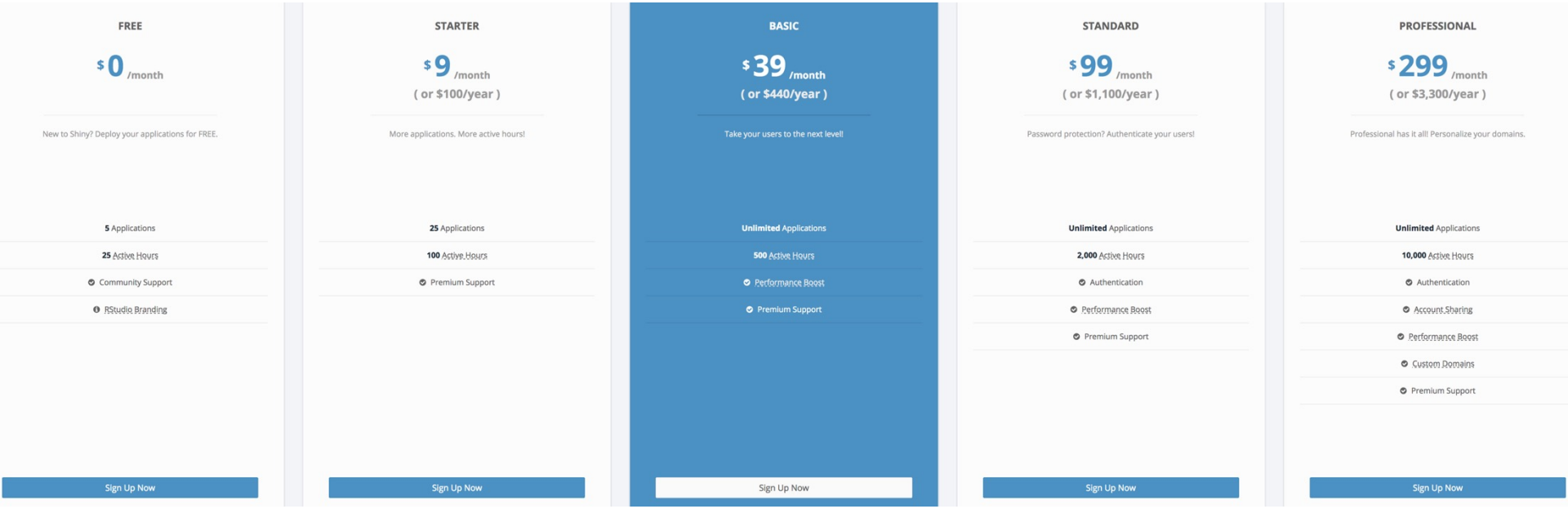
```
runGist("3239667")
```

Share as a web page

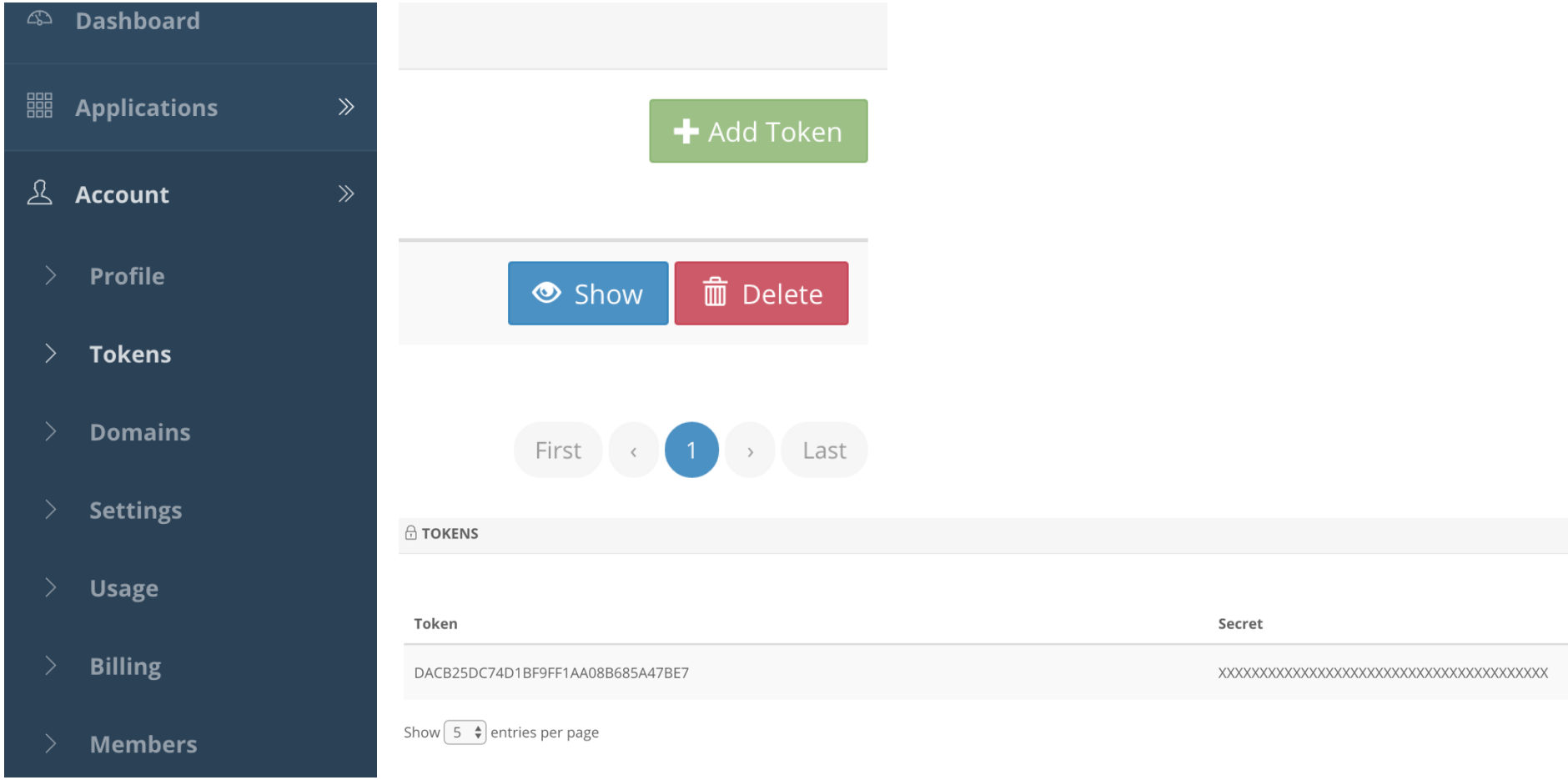
- Shinyapps.io



Share as a web page



Registration



Share by Shinyapps.io

- **STEP 1** – INSTALL RSCONNECT

```
install.packages('rsconnect')
```

- **STEP 2** – AUTHORIZE ACCOUNT

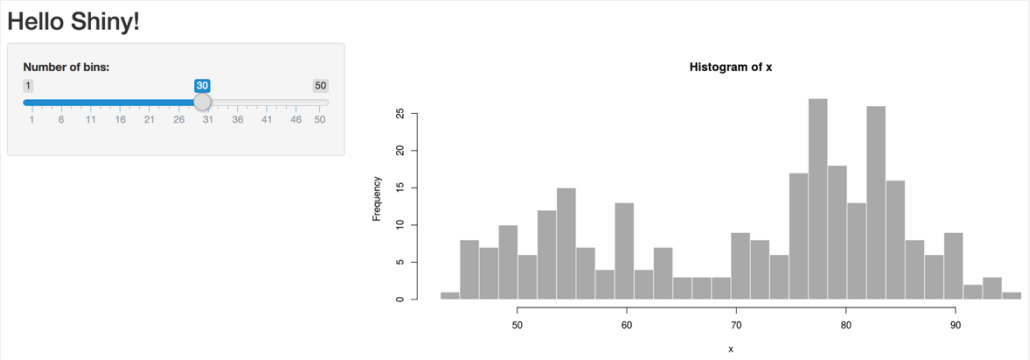
```
rsconnect::setAccountInfo(name='jmchang',
token='***', secret='***')
```

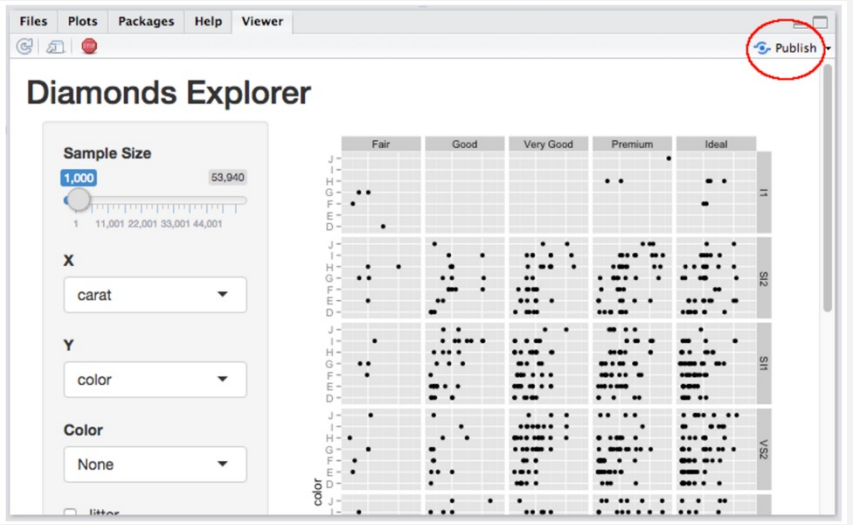
- **STEP 3** – DEPLOY

```
library(rsconnect)
rsconnect::deployApp('path/to/your/app')
```

Share by Shinyapps.io

- https://changlab.shinyapps.io/app1/





ggvis + shiny

- https://changlab.shinyapps.io/ggvisExample/
- server.r

```
library(shiny)
library(ggvis)
shinyServer(function(input, output, session) {
  mtc <- reactive({ mtcars[1:input$n, ] })
  mtc %>%
    ggvis(~wt, ~mpg) %>%
    layer_points() %>%
    bind_shiny("plot", "plot_ui")
  output$mtc_table <- renderTable({
    mtc()[, c("wt", "mpg")]
  })
})
```

ggvis + shiny

- ui.r

```
library(ggvis)
shinyUI(pageWithSidebar(
  div(),
  sidebarPanel(
    sliderInput("n", "Number of points", min = 1, max =
nrow(mtcars),
                value = 10, step = 1),
    uiOutput("plot_ui")
  ),
  mainPanel(
    ggvisOutput("plot"),
    tableOutput("mtc_table")
  )
))
```

Quick-R

- http://www.statmethods.net/

Reference

- Visualizations 101 by Logi ANALYTICS
  - https://www.logianalytics.com/wp-content/uploads/2015/12/Visualizations-101.pdf

Reference

- ggplot2 book: https://amzn.com/331924275X
- *R Graphics Cookbook* by Winston Chang
  - Online http://www.cookbook-r.com/Graphs/
- *Graphical Data Analysis with R*, by Antony Unwin

2017 台灣資料科學年會

- Big Data Visualization 馬匡六, University of California at Davis Professor
- 重新認識資料視覺化 王昱舜, 交通大學資訊工程系副教授

AITC
教育部人工智慧技術及應用人才培育計畫
Artificial Intelligence Talent Cultivation Program