



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
WORK INTEGRATED LEARNING PROGRAMMES**

COURSE HANDOUT

Part A: Content Design

Course Title	Data Visualization and Interpretation
Course No(s)	ZG555
Credit Units	5
Course Author	Febin.A.Vahab
Version No	1.0
Date	

Course Description

The course provides an insight on the best practices used in Data Visualization and also illustrates the best tools used to achieve the same

Course Objectives

No	Description
CO1	To introduce key techniques and theory used in visualization, including data models, graphical perception and techniques for visual encoding and interaction.
CO2	Solving various visualization problem using tools like Tableau, Python (Matplotlib)
CO3	Best Practices of Dashboard Design, Designing dashboards meeting the design principles for various requirements

Text Book(s)	
T1	Storytelling with Data, A data visualization guide for business professionals, by Cole Nussbaumer Knaflic; Wiley
T2	Data Visualisation : A Successful Design Process By Andy Kirk
T3	Visualize This: The Flowing Data Guide to Design, Visualization & Statistics, by Nathan Yau, Wiley
T4	Information Dashboard Design: Displaying data for at-a-glance monitoring, Stephen Few, second edition
T5	Tableau Your Data: Fast and Easy Visual Analysis with Tableau Software, by Daniel G Murray
T6	Matplotlib for Python Developers: Effective techniques for data visualization with Python, by Aldrin Yim, Claire Chung and Allen Yu
T7	Hands on Data Visualization with Bokeh: Interactive web plotting for Python using Bokeh, by Kevin Jolly
R1	Mastering Tableau, by David Baldwin



Learning Outcomes:

No	Learning Outcomes
LO1	Concepts and best practices of Data Visualization
LO2	Best practices of Information Dashboard Design
LO3	Data Visualization using Tableau
LO4	Data Visualization using Python (Matplotlib)



Part B: Content Development Plan

Academic Term	First Semester 2019-20
Course Title	Data Visualization and Interpretation
Course No	DSECL ZG555
Credit	5
Content Developer	Febin.A.Vahab

Glossary of Terms

Module	M	Module is a standalone quantum of designed content. A typical course is delivered using a string of modules. M2 means module 2.
Contact Session	CS	Contact Session (CS) stands for a 2 hour long live session with students conducted either in a physical classroom or enabled through technology. In this model of instruction, instructor led sessions will be for 16 CS.
Recorded Lecture	RL	RL stands for Recorded Lecture or Recorded Lesson. It is presented to the student through an online portal. A given RL unfolds as a sequences of video segments interleaved with exercises.
Lab Exercises	LE	Lab exercises associated with various modules
Self-Study	SS	Specific content assigned for self study
Homework	HW	Specific problems/design/lab exercises assigned as homework

Modular Structure

Module Summary

No.	Title of the Module
M1	Data Visualizations and Practices
M2	Effective Dashboard Design
M3	Data Visualization with Tableau
M4	Data Visualization with Python – 1 (Matplotlib)
M5	Data Visualization with Python – 2 (Bokeh, Seaborn)



Detailed Structure

M1: Data Visualizations and Practices Contact Session 1-3

Type	Description/Plan	Reference Text Book/Chapters
CS1	<ul style="list-style-type: none">• Introduction• Exploiting the Digital age• Visualisation as a Discovery tool• Visualisation skills for the masses• The Visualisation methodology• Visualisation design objectives• Exploratory vs. explanatory analysis• Understanding the context for data presentations• 3 minute story• Effective Visuals<ul style="list-style-type: none">◦ Textuals◦ Tabulars◦ Graphicals	T1 Ch 1 and 2, T2 Ch1
	<ul style="list-style-type: none">• Gestalt principles of visual perception• Visual Ordering• Decluttering	T1 Ch 3
CS2	<ul style="list-style-type: none">• Preattentive attributes in text and graphs<ul style="list-style-type: none">◦ Size◦ Color◦ Position• Data Design concepts<ul style="list-style-type: none">◦ Affordances◦ Accessibility◦ Aesthetics	T1 Ch 4
	<ul style="list-style-type: none">• Storytelling• Visualization Design Lessons	T1 Ch 5
CS3	Taxonomy of Data Visualisation Methods <ul style="list-style-type: none">• Comparing Categories of Plots• Dot Plot• Bar Chart• Floating Bar• Histogram• Radial Chart• Glyph Chart	T2 Ch 5
	<ul style="list-style-type: none">• Case Studies<ul style="list-style-type: none">◦ Visualizing Pattern Over time◦ Visualizing Proportions◦ Visualizing Relationships	T3 , Ch 4, 5, 6
<u>SELF STUDY</u>		
<ul style="list-style-type: none">• Data-Driven Documents (D3.js charts)<ul style="list-style-type: none">◦ Exploring visual gallery		



- Simple charts creation
- <https://d3js.org/>
- Explore more D3 charts examples
- Explore Google charts library
- <https://developers.google.com/chart/>
- Good Enough to Great: A Quick Guide for Better Data Visualizations
- <https://www.tableau.com/learn/whitepapers/good-enough-great-quick-guide-better-data-visualizations>

M2: Data Visualization with Tableau

Contact Session 4-7

Type	Description/Plan	Reference
CS4	<ul style="list-style-type: none">• Exploring Tableau<ul style="list-style-type: none">○ User Interface○ Tableau Prep○ Data Connection○ Data Preparation	T5 Ch 1, 2, 3 https://www.tableau.com/learn/training
CS5	<ul style="list-style-type: none">• Visual Analytics<ul style="list-style-type: none">○ Data Analysis○ Visuals	https://www.tableau.com/learn/training T5 Ch 3, 4
CS6	<ul style="list-style-type: none">• Maps	T3 Ch 6
	<ul style="list-style-type: none">• Dashboard and Stories	https://www.tableau.com/learn/training T5 Ch 8
CS7	<ul style="list-style-type: none">• Beyond the Basic Chart Types<ul style="list-style-type: none">○ Bullet graphs○ Pareto charts○ Custom background images	R1 Ch 7
	<ul style="list-style-type: none">• Visualization Best Practices and Dashboard Design	R1 Ch 10

SELF STUDY

- Explore the different types of visuals that can be plotted with Tableau interface

M3: Effective Dashboard Design

Contact Session 8-10

Type	Description/Plan	Reference
CS8	<ul style="list-style-type: none">• Dashboard• Dashboard categorization and typical data• Characteristics of a Well-Designed Dashboard• Key Goals in the Visual Design Process	T4 Ch 2 and 5
	<ul style="list-style-type: none">• Common Mistakes in Dashboard Design	T4 Ch 3
CS9	<ul style="list-style-type: none">• Power of Visual Perception<ul style="list-style-type: none">○ Visually Encoding Data for Rapid Perception○ Applying the Principles of Visual Perception to	T4 Ch 4



	Dashboard Design	
	<ul style="list-style-type: none"> Effective Dashboard Display Media Dashboards design for Usability 	T4 Ch 6 T4 Ch 7
CS10	<ul style="list-style-type: none"> Case Studies <ul style="list-style-type: none"> Sample Sales Dashboard Sample CIO Dashboard Sample Telesales Dashboard Sample Marketing Analysis Dashboard Bringing it all together with Dashboards <ul style="list-style-type: none"> How Dashboard Facilitates Analysis and Understanding How Tableau Improves the Dashboard-building process The right way to build a Dashboard Best Practices for Dashboard building 	T4 Ch 8
		T5 Ch8

SELF STUDY

- Explore any 2 dashboard design tools
<https://dzone.com/articles/20-free-and-open-source-data-visualization-tools>
- Build Your Competitive Edge: 12 Powerful Retail Dashboards
<https://www.tableau.com/learn/whitepapers/powerful-retail-dashboards>
- 10 Best Practices for Building Effective Dashboards
<https://www.tableau.com/learn/whitepapers/10-best-practices-building-effective-dashboards>

M4: Data Visualization with Python – 1 (Matplotlib)

Contact Session 11-14

Type	Description/Plan	Reference
CS11	<ul style="list-style-type: none"> Merits of Matplotlib The Lifecycle of a Plot Pyplot Matplotlib visuals basics 	https://matplotlib.org/tutorials/index.html T6 Ch 1 and Ch2
CS12	<ul style="list-style-type: none"> Plot styles types Visual Decorations 	http://www.labri.fr/perso/nrougier/teaching/matplotlib/ T6 Ch 3
CS13	<ul style="list-style-type: none"> Advanced Matplotlib 	T6 Ch4
CS14	Matplotlib in the real world <ul style="list-style-type: none"> Plotting data from a database Plotting data from a CSV file Plotting extrapolated data using curve fitting Plotting geographical data 	T6 Ch9

SELF STUDY

- Analysis of time series data using matplotlib



- Plotting Univariate Distributions
- Plotting Bivariate Distributions

M5: Data Visualization with Python – 2 (Seaborn and Bokeh)

Contact Session 15-16

Type	Description/Plan	Reference
CS15	<ul style="list-style-type: none">• Seaborn package<ul style="list-style-type: none">○ Seaborn vs Matplotlib○ Data Loading○ Seaborn Basic Plots	https://seaborn.pydata.org/ https://www.datacamp.com/community/tutorials/seaborn-python-tutorial
	<ul style="list-style-type: none">• Statistical plots with Seaborn	https://www.datacamp.com/courses/introduction-to-data-visualization-with-python
CS16	<ul style="list-style-type: none">• Plotting using Glyphs• Plotting with different Data Structures	T7 Ch 1 T7 Ch 2
	<ul style="list-style-type: none">• Using Annotations, Widgets, and Visual Attributes for Visual Enhancement• Building and Hosting Applications Using the Bokeh Server	T7 Ch 4 T7 Ch 5
SELF STUDY		
<ul style="list-style-type: none">• Try out all the statistical plots mentioned in datacamp's tutorial• https://www.datacamp.com/courses/introduction-to-data-visualization-with-python• Try out the Bokeh tutorial• https://www.analyticsvidhya.com/blog/2015/08/interactive-data-visualization-library-python-bokeh/		

Evaluation Scheme:

Legend: EC = Evaluation Component; AN = After Noon Session; FN = Fore Noon Session

No	Name	Type	Duration	Weight	Day, Date, Session, Time
EC-1	Quiz	Online	-	5%	Dec
EC-1	Assignment	Online	-	25%	Dec,Feb/March
EC-2	Mid-Semester Test	Closed Book	1.5 hours	30%	Dec
EC-3	Comprehensive Exam	Open Book	2.5 hours	40%	March

Note: Assignment can be replaced by QUIZ also.

Syllabus for Mid-Semester Test (Closed Book): Topics in Session Nos. 1 to 7

Syllabus for Comprehensive Exam (Open Book): All topics (Session Nos. 1 to 16)



Important links and information:

CANVAS(LMS)

Students are expected to visit the CANVAS course page on a regular basis and stay up to date with the latest announcements and deadlines.

Contact sessions: Students should attend the online lectures as per the schedule provided on CANVAS.

Evaluation Guidelines:

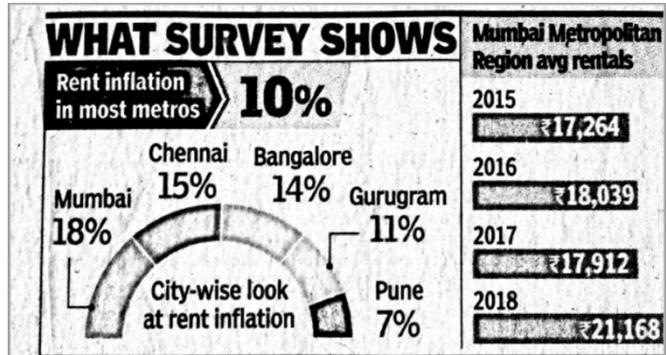
1. EC1 consists of two assignments(Quiz/Assignment). Announcements will be made on the portal, in a timely manner.
2. For Closed Book tests: No books or reference material of any kind will be permitted.
3. For Open Book exams: Use of books and any printed / written reference material (filed or bound) is permitted. However, loose sheets of paper will not be allowed. Use of calculators is permitted in all exams. Laptops/Mobiles of any kind are not allowed. Exchange of any material is not allowed.
4. If a student is unable to appear for the Regular Test/Exam due to genuine exigencies, the student should follow the procedure to apply for the Make-Up Test/Exam which will be made available in CANVAS. The Make-Up Test/Exam will be conducted only at selected exam centres on the dates to be announced later.

It shall be the responsibility of the individual student to be regular in maintaining the self study schedule as given in the course handout, attend the online lectures, and take all the prescribed evaluation components such as Assignment/Quiz, Mid-Semester Test and Comprehensive Exam according to the evaluation scheme provided in the handout.

Sample Questions



The India Rent Report 2018 (Residential) reveals that The Mumbai Metropolitan Region (MMR) which includes Thane and Navi Mumbai, experienced a maximum rent inflation of 18% in 2018, the highest among five Indian cities. Its followed by Chennai, Bangalore, Bangalore and Gurugram. The average rent in MMR jumped to Rs 21,168 from last year's Rs 17912 in 2017 followed by Chennai and Bangalore. The visuals showcased below are trying to convey two rent related stories. Makeover these two visualization with lessons learnt in the course.



BITS Pilani

1

Sample Questions



Identify areas of improvement and design a new dashboard



BITS Pilani

2

Sample Questions



Suggest a CIO dashboard where the following data will be reported:

- System availability (uptime)
- Expenses
- Customer satisfaction
- Severe problem count
- CPU usage relative to capacity
- Storage usage relative to capacity
- Network traffic
- Application response time
- Major project milestones
- Top projects in the queue
- Other critical events

BITS Pilani

3

Sample Questions



BITS Pilani

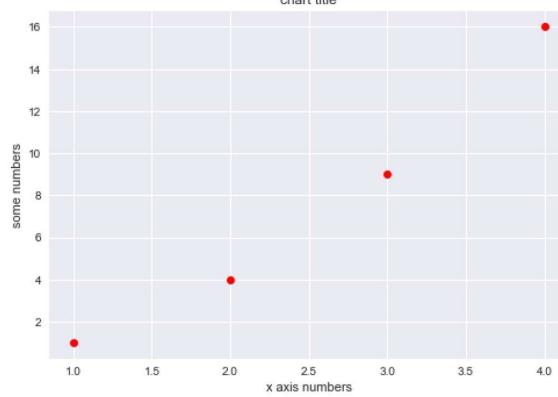
4



Sample Questions

Complete the following Matplotlib code snippet so that it generates the shown outcome.

1. import _____ as plt
2. x = [1, 2, 3, 4]
3. y = [1, 4, 9, 16]
4. plt._____ (_____, _____, "ro")
5. plt._____ ("chart title")
6. plt._____ ('some numbers')
7. plt.xlabel('x axis numbers')
8. plt._____()



BITS Pilani



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
WORK INTEGRATED LEARNING PROGRAMMES**

COURSE HANDOUT

Part A: Content Design

Course Title	Data Visualization and Interpretation
Course No(s)	ZG555
Credit Units	5
Course Author	Febin.A.Vahab
Version No	1.0
Date	

Course Description

The course provides an insight on the best practices used in Data Visualization and also illustrates the best tools used to achieve the same

Course Objectives

No	Description
CO1	To introduce key techniques and theory used in visualization, including data models, graphical perception and techniques for visual encoding and interaction.
CO2	Solving various visualization problem using tools like Tableau, Python (Matplotlib)
CO3	Best Practices of Dashboard Design, Designing dashboards meeting the design principles for various requirements

Text Book(s)	
T1	Storytelling with Data, A data visualization guide for business professionals, by Cole Nussbaumer Knaflic; Wiley
T2	Data Visualisation : A Successful Design Process By Andy Kirk
T3	Visualize This: The Flowing Data Guide to Design, Visualization & Statistics, by Nathan Yau, Wiley
T4	Information Dashboard Design: Displaying data for at-a-glance monitoring, Stephen Few, second edition
T5	Tableau Your Data: Fast and Easy Visual Analysis with Tableau Software, by Daniel G Murray
T6	Matplotlib for Python Developers: Effective techniques for data visualization with Python, by Aldrin Yim, Claire Chung and Allen Yu
T7	Hands on Data Visualization with Bokeh: Interactive web plotting for Python using Bokeh, by Kevin Jolly
R1	Mastering Tableau, by David Baldwin



Learning Outcomes:

No	Learning Outcomes
LO1	Concepts and best practices of Data Visualization
LO2	Best practices of Information Dashboard Design
LO3	Data Visualization using Tableau
LO4	Data Visualization using Python (Matplotlib)



Part B: Content Development Plan

Academic Term	First Semester 2019-20
Course Title	Data Visualization and Interpretation
Course No	DSECL ZG555
Credit	5
Content Developer	Febin.A.Vahab

Glossary of Terms

Module	M	Module is a standalone quantum of designed content. A typical course is delivered using a string of modules. M2 means module 2.
Contact Session	CS	Contact Session (CS) stands for a 2 hour long live session with students conducted either in a physical classroom or enabled through technology. In this model of instruction, instructor led sessions will be for 16 CS.
Recorded Lecture	RL	RL stands for Recorded Lecture or Recorded Lesson. It is presented to the student through an online portal. A given RL unfolds as a sequences of video segments interleaved with exercises.
Lab Exercises	LE	Lab exercises associated with various modules
Self-Study	SS	Specific content assigned for self study
Homework	HW	Specific problems/design/lab exercises assigned as homework

Modular Structure

Module Summary

No.	Title of the Module
M1	Data Visualizations and Practices
M2	Effective Dashboard Design
M3	Data Visualization with Tableau
M4	Data Visualization with Python – 1 (Matplotlib)
M5	Data Visualization with Python – 2 (Bokeh, Seaborn)



Detailed Structure

M1: Data Visualizations and Practices Contact Session 1-3

Type	Description/Plan	Reference Text Book/Chapters
CS1	<ul style="list-style-type: none">• Introduction• Exploiting the Digital age• Visualisation as a Discovery tool• Visualisation skills for the masses• The Visualisation methodology• Visualisation design objectives• Exploratory vs. explanatory analysis• Understanding the context for data presentations• 3 minute story• Effective Visuals<ul style="list-style-type: none">◦ Textuals◦ Tabulars◦ Graphicals	T1 Ch 1 and 2, T2 Ch1
	<ul style="list-style-type: none">• Gestalt principles of visual perception• Visual Ordering• Decluttering	T1 Ch 3
CS2	<ul style="list-style-type: none">• Preattentive attributes in text and graphs<ul style="list-style-type: none">◦ Size◦ Color◦ Position• Data Design concepts<ul style="list-style-type: none">◦ Affordances◦ Accessibility◦ Aesthetics	T1 Ch 4
	<ul style="list-style-type: none">• Storytelling• Visualization Design Lessons	T1 Ch 5
CS3	Taxonomy of Data Visualisation Methods <ul style="list-style-type: none">• Comparing Categories of Plots• Dot Plot• Bar Chart• Floating Bar• Histogram• Radial Chart• Glyph Chart	T2 Ch 5
	<ul style="list-style-type: none">• Case Studies<ul style="list-style-type: none">◦ Visualizing Pattern Over time◦ Visualizing Proportions◦ Visualizing Relationships	T3 , Ch 4, 5, 6
<u>SELF STUDY</u>		
<ul style="list-style-type: none">• Data-Driven Documents (D3.js charts)<ul style="list-style-type: none">◦ Exploring visual gallery		



- Simple charts creation
- <https://d3js.org/>
- Explore more D3 charts examples
- Explore Google charts library
- <https://developers.google.com/chart/>
- Good Enough to Great: A Quick Guide for Better Data Visualizations
- <https://www.tableau.com/learn/whitepapers/good-enough-great-quick-guide-better-data-visualizations>

M2: Data Visualization with Tableau

Contact Session 4-7

Type	Description/Plan	Reference
CS4	<ul style="list-style-type: none">• Exploring Tableau<ul style="list-style-type: none">○ User Interface○ Tableau Prep○ Data Connection○ Data Preparation	T5 Ch 1, 2, 3 https://www.tableau.com/learn/training
CS5	<ul style="list-style-type: none">• Visual Analytics<ul style="list-style-type: none">○ Data Analysis○ Visuals	https://www.tableau.com/learn/training T5 Ch 3, 4
CS6	<ul style="list-style-type: none">• Maps	T3 Ch 6
	<ul style="list-style-type: none">• Dashboard and Stories	https://www.tableau.com/learn/training T5 Ch 8
CS7	<ul style="list-style-type: none">• Beyond the Basic Chart Types<ul style="list-style-type: none">○ Bullet graphs○ Pareto charts○ Custom background images	R1 Ch 7
	<ul style="list-style-type: none">• Visualization Best Practices and Dashboard Design	R1 Ch 10

SELF STUDY

- Explore the different types of visuals that can be plotted with Tableau interface

M3: Effective Dashboard Design

Contact Session 8-10

Type	Description/Plan	Reference
CS8	<ul style="list-style-type: none">• Dashboard• Dashboard categorization and typical data• Characteristics of a Well-Designed Dashboard• Key Goals in the Visual Design Process	T4 Ch 2 and 5
	<ul style="list-style-type: none">• Common Mistakes in Dashboard Design	T4 Ch 3
CS9	<ul style="list-style-type: none">• Power of Visual Perception<ul style="list-style-type: none">○ Visually Encoding Data for Rapid Perception○ Applying the Principles of Visual Perception to	T4 Ch 4



	Dashboard Design	
	<ul style="list-style-type: none"> Effective Dashboard Display Media Dashboards design for Usability 	T4 Ch 6 T4 Ch 7
CS10	<ul style="list-style-type: none"> Case Studies <ul style="list-style-type: none"> Sample Sales Dashboard Sample CIO Dashboard Sample Telesales Dashboard Sample Marketing Analysis Dashboard Bringing it all together with Dashboards <ul style="list-style-type: none"> How Dashboard Facilitates Analysis and Understanding How Tableau Improves the Dashboard-building process The right way to build a Dashboard Best Practices for Dashboard building 	T4 Ch 8
		T5 Ch8

SELF STUDY

- Explore any 2 dashboard design tools
<https://dzone.com/articles/20-free-and-open-source-data-visualization-tools>
- Build Your Competitive Edge: 12 Powerful Retail Dashboards
<https://www.tableau.com/learn/whitepapers/powerful-retail-dashboards>
- 10 Best Practices for Building Effective Dashboards
<https://www.tableau.com/learn/whitepapers/10-best-practices-building-effective-dashboards>

M4: Data Visualization with Python – 1 (Matplotlib)

Contact Session 11-14

Type	Description/Plan	Reference
CS11	<ul style="list-style-type: none"> Merits of Matplotlib The Lifecycle of a Plot Pyplot Matplotlib visuals basics 	https://matplotlib.org/tutorials/index.html T6 Ch 1 and Ch2
CS12	<ul style="list-style-type: none"> Plot styles types Visual Decorations 	http://www.labri.fr/perso/nrougier/teaching/matplotlib/ T6 Ch 3
CS13	<ul style="list-style-type: none"> Advanced Matplotlib 	T6 Ch4
CS14	Matplotlib in the real world <ul style="list-style-type: none"> Plotting data from a database Plotting data from a CSV file Plotting extrapolated data using curve fitting Plotting geographical data 	T6 Ch9

SELF STUDY

- Analysis of time series data using matplotlib



- Plotting Univariate Distributions
- Plotting Bivariate Distributions

M5: Data Visualization with Python – 2 (Seaborn and Bokeh)

Contact Session 15-16

Type	Description/Plan	Reference
CS15	<ul style="list-style-type: none">• Seaborn package<ul style="list-style-type: none">○ Seaborn vs Matplotlib○ Data Loading○ Seaborn Basic Plots	https://seaborn.pydata.org/ https://www.datacamp.com/community/tutorials/seaborn-python-tutorial
	<ul style="list-style-type: none">• Statistical plots with Seaborn	https://www.datacamp.com/courses/introduction-to-data-visualization-with-python
CS16	<ul style="list-style-type: none">• Plotting using Glyphs• Plotting with different Data Structures	T7 Ch 1 T7 Ch 2
	<ul style="list-style-type: none">• Using Annotations, Widgets, and Visual Attributes for Visual Enhancement• Building and Hosting Applications Using the Bokeh Server	T7 Ch 4 T7 Ch 5
SELF STUDY		
<ul style="list-style-type: none">• Try out all the statistical plots mentioned in datacamp's tutorial• https://www.datacamp.com/courses/introduction-to-data-visualization-with-python• Try out the Bokeh tutorial• https://www.analyticsvidhya.com/blog/2015/08/interactive-data-visualization-library-python-bokeh/		

Evaluation Scheme:

Legend: EC = Evaluation Component; AN = After Noon Session; FN = Fore Noon Session

No	Name	Type	Duration	Weight	Day, Date, Session, Time
EC-1	Quiz	Online	-	5%	Dec
EC-1	Assignment	Online	-	25%	Dec,Feb/March
EC-2	Mid-Semester Test	Closed Book	1.5 hours	30%	Dec
EC-3	Comprehensive Exam	Open Book	2.5 hours	40%	March

Note: Assignment can be replaced by QUIZ also.

Syllabus for Mid-Semester Test (Closed Book): Topics in Session Nos. 1 to 7

Syllabus for Comprehensive Exam (Open Book): All topics (Session Nos. 1 to 16)



Important links and information:

CANVAS(LMS)

Students are expected to visit the CANVAS course page on a regular basis and stay up to date with the latest announcements and deadlines.

Contact sessions: Students should attend the online lectures as per the schedule provided on CANVAS.

Evaluation Guidelines:

1. EC1 consists of two assignments(Quiz/Assignment). Announcements will be made on the portal, in a timely manner.
2. For Closed Book tests: No books or reference material of any kind will be permitted.
3. For Open Book exams: Use of books and any printed / written reference material (filed or bound) is permitted. However, loose sheets of paper will not be allowed. Use of calculators is permitted in all exams. Laptops/Mobiles of any kind are not allowed. Exchange of any material is not allowed.
4. If a student is unable to appear for the Regular Test/Exam due to genuine exigencies, the student should follow the procedure to apply for the Make-Up Test/Exam which will be made available in CANVAS. The Make-Up Test/Exam will be conducted only at selected exam centres on the dates to be announced later.

It shall be the responsibility of the individual student to be regular in maintaining the self study schedule as given in the course handout, attend the online lectures, and take all the prescribed evaluation components such as Assignment/Quiz, Mid-Semester Test and Comprehensive Exam according to the evaluation scheme provided in the handout.

CONTACT SESSION 1 -PLAN



Contact Sessions(#)	List of Topic Title	Text/Ref Book/external resource
1	<ul style="list-style-type: none"> • Introduction • Exploiting the Digital age • Visualisation as a Discovery tool • Visualisation skills for the masses • The Visualisation methodology • Visualisation design objectives • Exploratory vs. explanatory analysis • Understanding the context for data presentations • 3 minute story • Effective Visuals • Gestalt principles of visual perception • Visual Ordering • Decluttering 	T1 Ch 1 and 2, T2 Ch1

1

BITS Pilani

Bad graphs are everywhere



- We learn a lot about language and math.
- We learn how to put words together into sentences and into stories.
- We learn to make sense of numbers.

2

BITS Pilani



Bad graphs are everywhere

- Often end up relying on our tools to understand best practices
- Tool defaults and general practices tend to leave our data and the stories we want to tell with that data sorely lacking.
- There is a story in your data. But your tools don't know what that story is.
- That's where it takes you to bring that story visually and contextually to life.

3

BITS Pilani

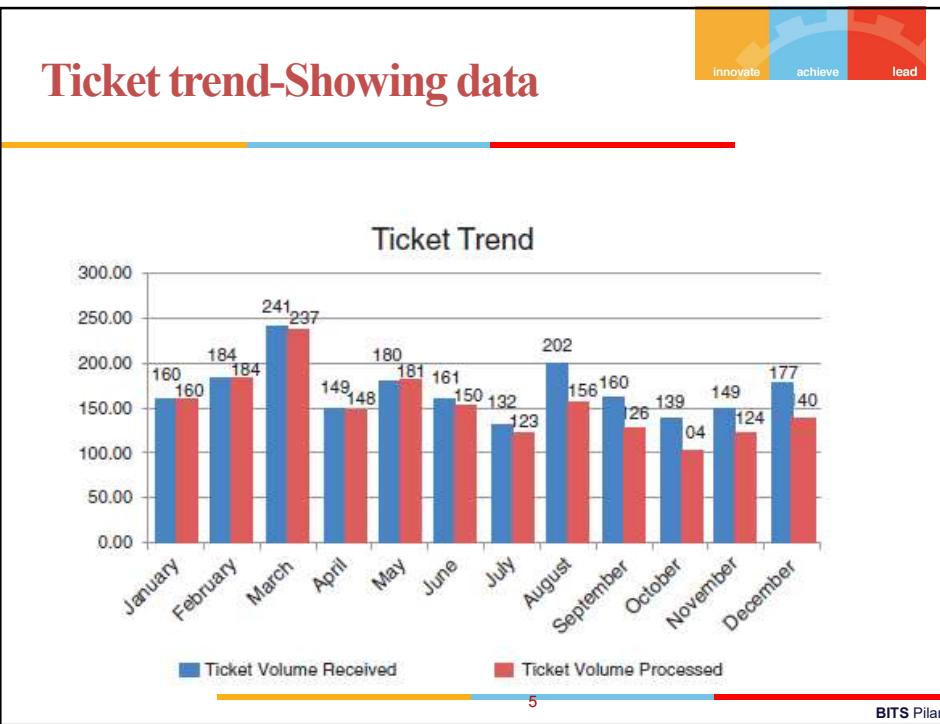
Bad graphs are everywhere



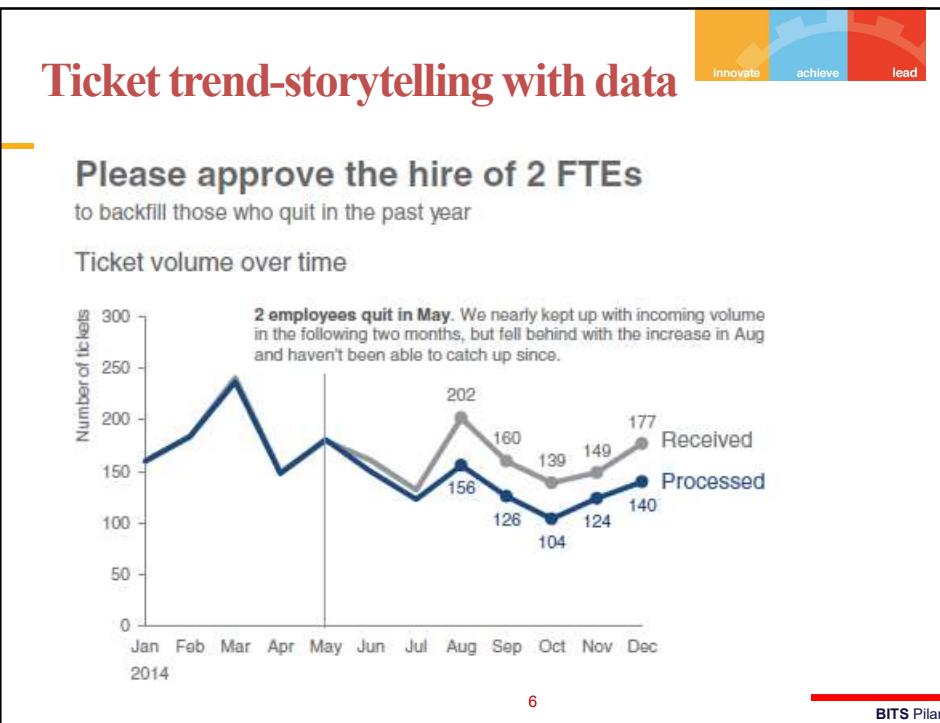
- The following are a few examples before-and-afters to give you a visual sense of what is story telling.(will learn in detail in later sessions)

4

BITS Pilani



5



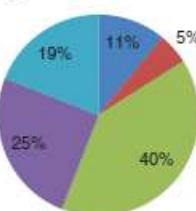
6

Survey Results-Showing data

innovate achieve lead

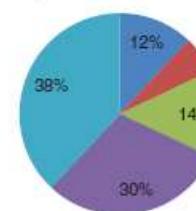
Survey Results

PRE: How do you feel about doing science?



Feeling	Percentage
Bored	5%
Not great	11%
OK	40%
Kind of interested	25%
Excited	19%

POST: How do you feel about doing science?



Feeling	Percentage
Bored	12%
Not great	6%
OK	14%
Kind of interested	30%
Excited	38%

BITS Pilani

7

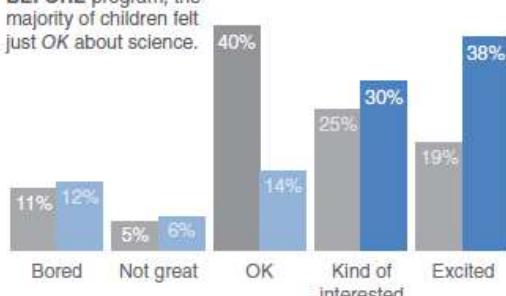
Survey Results-Story Telling with data

innovate achieve lead

Pilot program was a success

How do you feel about science?

BEFORE program, the majority of children felt just **OK** about science.



Feeling	Percentage
Bored	11%
Not great	5%
OK	40%
Kind of interested	25%
Excited	19%

AFTER program, more children were *Kind of interested* & *Excited* about science.

Based on survey of 100 students conducted before and after pilot program (100% response rate on both surveys).

8



Story telling with Data

- Being able to tell stories with data is a skill that's becoming ever more important in our world of increasing data and desire for data driven decision making.

9

BITS Pilani

9



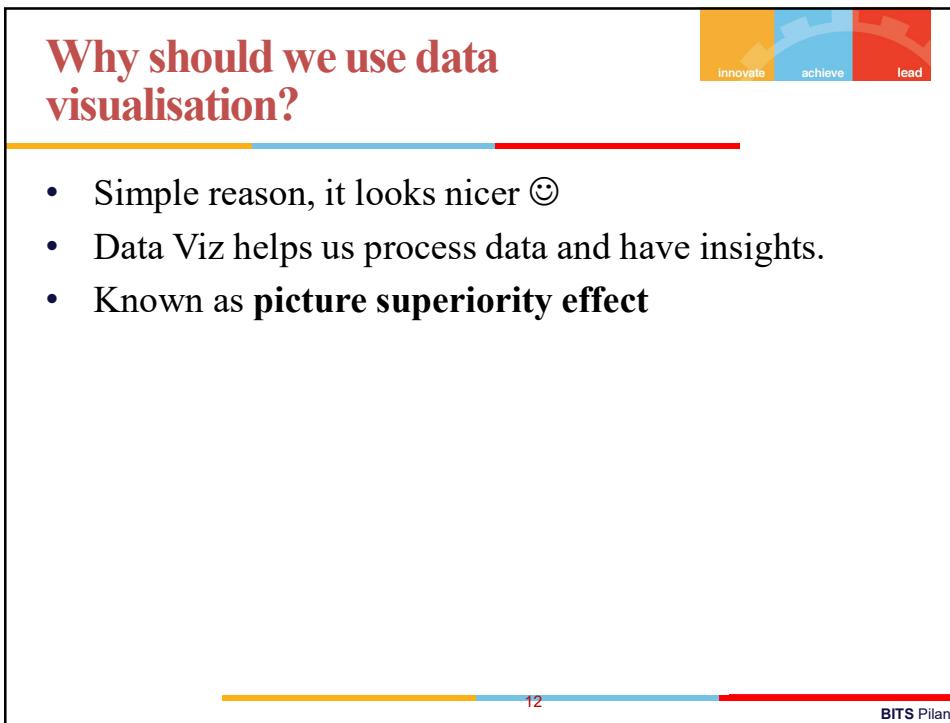
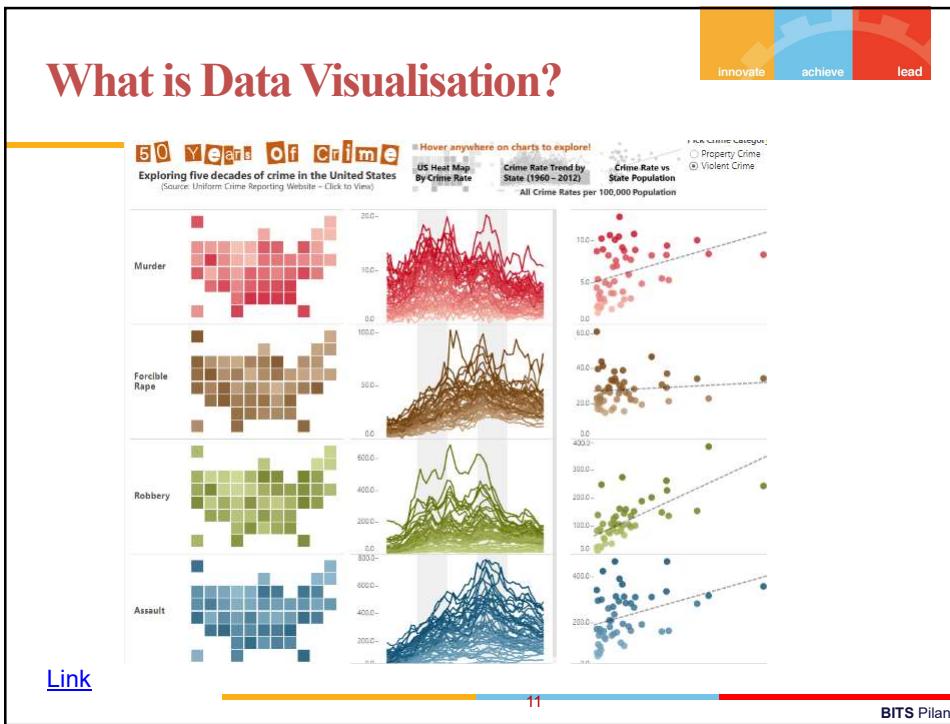
What is Data Visualisation?

- Simply put,
- **Data Viz is using pictures to represent data.**
- It has been there for many years but the recent technological advancements have made it easy to summarize large volumes of data very quickly.

10

BITS Pilani

10



Why should we use data visualisation?



	Product A	Product B	Product C	Product D	Product E	Product F	Product G	Product H
Northeast	\$ 15,749.00	\$ 40,195.00	\$ 15,472.00	\$ 63,029.00	\$ 8,509.00	\$ 42,987.00	\$ 27,778.00	\$ 12,510.00
Southeast	\$ 48,044.00	\$ 20,741.00	\$ 40,643.00	\$ 15,687.00	\$ 12,342.00	\$ 23,297.00	\$ 10,401.00	\$ 10,522.00
Central	\$ 5,240.00	\$ 45,296.00	\$ 16,114.00	\$ 63,359.00	\$ 58,198.00	\$ 24,191.00	\$ 46,826.00	\$ 50,278.00
Northwest	\$ 12,860.00	\$ 11,548.00	\$ 43,134.00	\$ 19,331.00	\$ 60,563.00	\$ 51,475.00	\$ 28,954.00	\$ 14,405.00
Southwest	\$ 37,087.00	\$ 61,506.00	\$ 54,084.00	\$ 14,715.00	\$ 17,811.00	\$ 32,814.00	\$ 47,853.00	\$ 44,639.00

13

BITS Pilani

13

Why should we use data visualisation?



	Product A	Product B	Product C	Product D	Product E	Product F	Product G	Product H
Northeast	\$ 15,749.00	\$ 40,195.00	\$ 15,472.00	\$ 63,029.00	\$ 8,509.00	\$ 42,987.00	\$ 27,778.00	\$ 12,510.00
Southeast	\$ 48,044.00	\$ 20,741.00	\$ 40,643.00	\$ 15,687.00	\$ 12,342.00	\$ 23,297.00	\$ 10,401.00	\$ 10,522.00
Central	\$ 5,240.00	\$ 45,296.00	\$ 16,114.00	\$ 63,359.00	\$ 58,198.00	\$ 24,191.00	\$ 46,826.00	\$ 50,278.00
Northwest	\$ 12,860.00	\$ 11,548.00	\$ 43,134.00	\$ 19,331.00	\$ 60,563.00	\$ 51,475.00	\$ 28,954.00	\$ 14,405.00
Southwest	\$ 37,087.00	\$ 61,506.00	\$ 54,084.00	\$ 14,715.00	\$ 17,811.00	\$ 32,814.00	\$ 47,853.00	\$ 44,639.00

14

BITS Pilani

14



Why should we use data visualisation?

Big data

- Wind map
<http://hint.fm/wind/>
- Real time credit card monitoring

15

BITS Pilani

15



Data Visualization



What?

Graphical / Visual representation of data

Why?

- Way to identify patterns, trends and outliers in data
- Helps in making data-driven decisions

How?

- *That is what we will learn in this course!*

16

BITS Pilani

16



Purpose

"The ability to take data—to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it—that's going to be a hugely important skill in the next decades" –

Hal Varian, Google's chief economist

"The purpose of visualization is insight, not just picture. " –

Ben Shneiderman, Data visualization pioneer

17

BITS Pilani

Exploiting the digital age



- Data is now rightly seen as an invaluable asset.
- "Data is the new oil"

18

BITS Pilani

18

Visualization as a discovery tool




"The greatest value of a picture is when it forces us to notice what we never expected to see" - John W Tukey

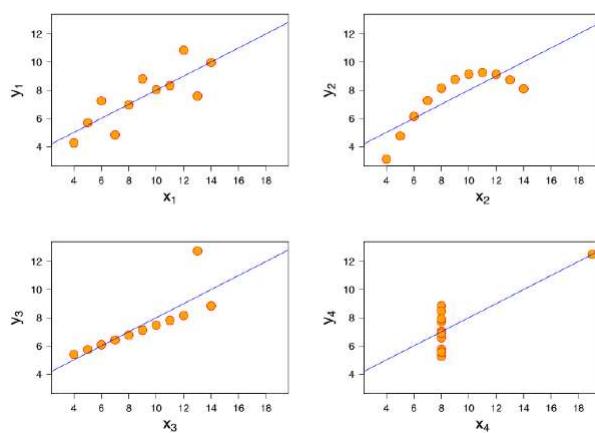
x1	y1	x2	y2	x3	y3	x4	y4
10	8.04	10	9.14	10	7.46	8	6.58
8	6.95	8	8.14	8	6.77	8	5.76
13	7.58	13	8.74	13	12.74	8	7.71
9	8.81	9	8.77	9	7.11	8	8.84
11	8.33	11	9.26	11	7.81	8	8.47
14	9.96	14	8.1	14	8.84	8	7.04
6	7.24	6	6.13	6	6.08	8	5.25
4	4.26	4	3.1	4	5.39	19	12.5
12	10.84	12	9.13	12	8.15	8	5.56
7	4.82	7	7.26	7	6.42	8	7.91
5	5.68	5	4.74	5	5.73	8	6.89

19

BITS Pilani

19

Visualization as a discovery tool

- the general tendency about a trend line in **X1, Y1**
- the curvature pattern of **X2, Y2**
- the strong linear pattern with single outlier in **X3, Y3**
- the similarly strong linear pattern with an outlier for **X4, Y4**

20

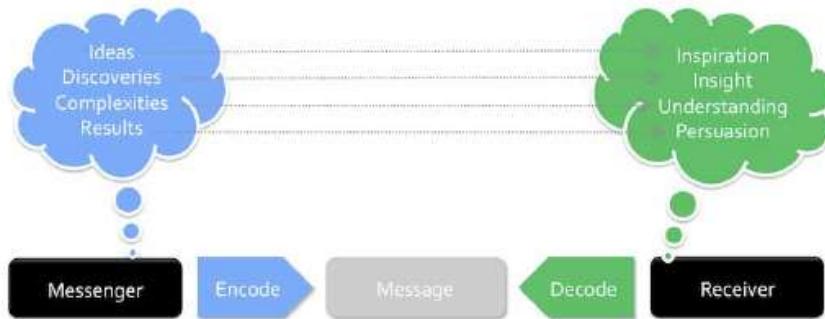
BITS Pilani

20

Definition



“ Data visualization is the use of visual representations to explore, make sense of, and communicate data.” - Data visualization expert, Stephen Few



21

BITS Pilani

21

Definition



- The representation and presentation of data that exploits our visual perception abilities in order to amplify cognition.

22

BITS Pilani

22

Human Brain and Data Visualisation



“The purpose of visualization is insight, not just picture.” – Ben Shneiderman, Data visualization pioneer

- This means the most important transformation that you can make as a designer is not turning data into charts
- But turning data into something clear and meaningful in the minds of your audiences

23

BITS Pilani

23

Human Brain and Data Visualisation



“The purpose of visualization is insight, not just picture.” – Ben Shneiderman, Data visualization pioneer

Visual Processing

- 30-60% of Human Brains capacity is used for visual processing
- By comparison, touch is about 8% for example and hearing is only about 2%.
- Processes 10 million bits per second
- Fastest mode of processing
- Our brains are wired for **fast** visual processing

24

BITS Pilani

24

Human Brain and Data Visualisation



For our ancestors this need for speed could tip the balance between life and death.

25

BITS Pilani

Human Brain and Data Visualisation



- The impulse to detect, match, and make sense of patterns is so strong that many of us do it not only for practical purposes, but also as a form of pure play in many games.

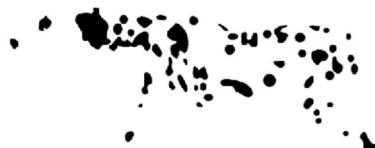
26

BITS Pilani

26

13

WHAT DO YOU SEE?



27

BITS Pilani

Human Brain and Data Visualisation



- A core part of data visualization design involves repurposing our innate visual abilities to adapt to a world that's flooding with data.
- And the more you become aware of people's visual processing strengths and limitations, the better your designs will become

28

BITS Pilani

28

Human Brain and Data Visualisation



Child vaccination rates

Measles, % of children, 1980 – 2013

Source: OECD Health Statistics : OECD Health Data: Health care utilisation

Show: Chart

Table

full

share

download

add to pinboard

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1'
France	15.0	26.0	35.0	39.0	45.0	51.0	58.0	71.0	74.0	76.0	78.0	80.0	83.0	84.0	83.0	82.0	81.0	81.0	81.0	
Turkey	27.0	52.0	64.0	63.0	62.0	61.0	64.0	67.0	82.0	71.0	78.0	73.0	78.0	72.0	76.0	65.0	72.0	79.0	81.0	8
United States	86.0	97.0	97.0	98.0	98.0	97.0	97.0	82.0	98.0	94.0	90.0	87.0	83.0	84.0	89.0	88.0	91.0	91.0	92.0	5

Information on data for Israel: <http://oe.cd/israel-disclaimer>

Not available; Break in series; Confidential data; Estimated value; Forecast value; Not applicable



29

BITS Pilani

29

Human Brain and Data Visualisation



- In some instances, data sets can consist of thousands or even millions of rows and columns
- So the direct use of only tables becomes impossible

30

BITS Pilani

30

Visualization skills for the masses



"The skills required for most effectively displaying information are not intuitive and rely largely on principles that must be learned." – Stephen Few from his book Show Me the Numbers

31

BITS Pilani

31

Visualization skills for the masses



- You have a certain design style based on personal taste
- You just play around until something emerges that you instinctively like the look of
- You trust software defaults and don't go beyond that in terms of modifying the design
- You have limited software capabilities, so you don't know how to modify a design
- You just do as the boss tells you—"can you do me some fancy charts?"

32

BITS Pilani

32

Visualization skills for the masses



Citizen Data Scientists

- Required skill set in data driven world
- Modern world needs both technical analysis and creative storytelling
- Data visualization sits exactly in middle of both
- Increasingly valuable asset

33

BITS Pilani

Visualization methodology



- A more organized and sequenced way
- To reduce inefficiency and wasted resource
- Require the assistance of a variety of applications and programs (the focus of this methodology is intended to be technology-neutral, placing an emphasis on the conceiving, reasoning, and decision-making)
- Reasoned decision-making
- Rarely, if ever, a single right answer or single best solution

34

BITS Pilani

34

Visualization Design Objectives



- 1. Strive for form and function (*style over substance*)**
- 2. Justifying the selection of everything we do**
- 3. Creating accessibility through intuitive design**
- 4. Never deceive the receiver**

35

BITS Pilani

Visualization Design Objectives



- 1. Strive for form and function (*style over substance*)**
 - Wind map
 - <http://hint.fm/wind/>

36

BITS Pilani

36

Visualization Design Objectives



1. Strive for form and function (*style over substance*)

Cognitive vs Perceptual Design Distinction

- Help people make clear, accurate interpretations
- Gain useful insights based on what they see.

37

BITS Pilani

Visualization Design Objectives



1. Strive for form and function (*style over substance*)

Cognitive vs Perceptual Design Distinction

- Nobel prize winning psychologist Daniel Kahneman suggested that **there are two fundamental systems that drive how we think and make judgments**.
- As a data visualization designer, your job is to leverage both of them

38

BITS Pilani

38

Visualization Design Objectives



1. Strive for form and function (*style over substance*)

Cognitive vs Perceptual Design Distinction

- The **first system** involves more automatic and immediate perception, such as noticing an unusual pattern of movement in the bushes. This is where visual encoding and things like Gestalt Principles come into play.
- The **second system** is about slower and more deliberate cognition. That is thinking about the meaning of certain sensory cues.

39

BITS Pilani

39

Visualization Design Objectives



1. Strive for form and function (*style over substance*)

Cognitive vs Perceptual Design Distinction

- The **first system** involves more automatic and immediate perception, such as noticing an unusual pattern of movement in the bushes. This is where visual encoding and things like Gestalt Principles come into play.
- The **second system** is about slower and more deliberate cognition. That is thinking about the meaning of certain sensory cues.

40

BITS Pilani

40

Visualization Design Objectives



1. Strive for form and function (*style over substance*)

Cognitive vs Perceptual Design Distinction

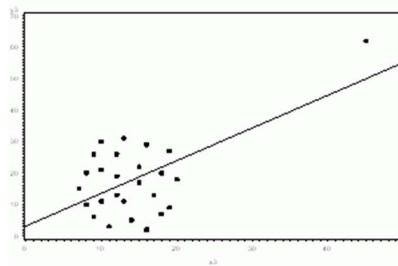
- The **first system** involves more automatic and immediate perception, such as noticing an unusual pattern of movement in the bushes. This is where visual encoding and things like Gestalt Principles come into play.
- The **second system** is about slower and more deliberate cognition. That is thinking about the meaning of certain sensory cues.

41

BITS Pilani

41

Visualization Design Objectives



42

BITS Pilani

42



Visualization Design Objectives

- The brain relies on some shortcuts and assumptions to help make our perception rapid, and that's a key to good data visualizations.
- But some of the same mental shortcuts can also create false interpretations and provide challenges and present challenges for visualization designs

43

BITS Pilani

43



Visualization Design Objectives



44

BITS Pilani

44

Visualization Design Objectives



- 1. Strive for form and function (*style over substance*)**
- 2. Justifying the selection of everything we do**
- 3. Creating accessibility through intuitive design**
- 4. Never deceive the receiver**

45

BITS Pilani

Visualization Design Objectives



- 2. Justifying the selection of everything we do**

"We're so busy thinking about if we can do things, we forget to consider whether we should."

- Everything you do must be thoroughly planned, understood, and reasoned

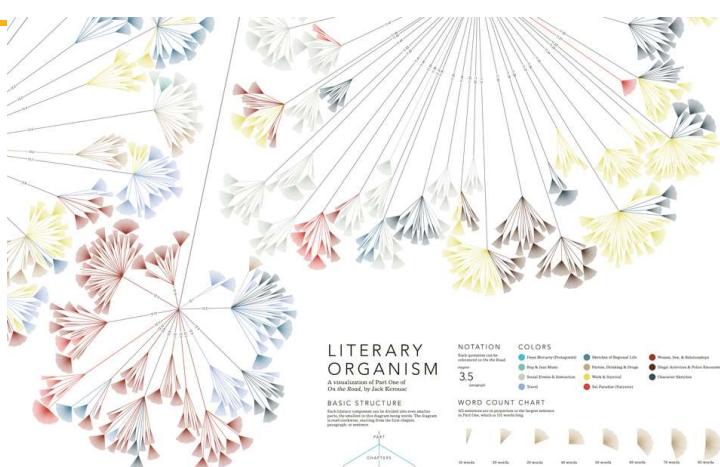
46

BITS Pilani

46


 The logo for BITS Pilani, featuring three colored squares (orange, blue, red) with the words "innovate", "achieve", and "lead" respectively.

Visualization Design Objectives



LITERARY ORGANISM

A visualization of the *Basic Structure of a Book* by William Korten

BASIC STRUCTURE

Book structure can be divided into two main levels. Each chapter is composed of one or more paragraphs. Each paragraph is composed of one or more sentences. Each sentence is composed of one or more words.

NOTATION

- Red pentagon = Part of Speech
- Blue hexagon = Word
- Yellow triangle = Chapter
- Green triangle = Paragraph
- Blue circle = Sentence
- Blue square = Thread
- Blue diamond = Association
- Blue star = Threading
- Blue circle with a dot = Regional Life
- Blue circle with a dot = Poetry, Prose, Drama & Tragedy
- Blue circle with a dot = Fiction & Non-fiction
- Blue circle with a dot = Legal & Political Discourse
- Blue circle with a dot = Gender Studies
- Blue circle with a dot = The Paradise (Character)

WORD COUNT CHART

All numbers are in proportion to our largest sentence (10 words).

Words	10 words	20 words	30 words	40 words	50 words	60 words	70 words	80 words	90 words	100 words
Chapter	1	2	3	4	5	6	7	8	9	10
Paragraph	10	20	30	40	50	60	70	80	90	100
Sentence	100	200	300	400	500	600	700	800	900	1000

<http://www.stefanieposavec.com/writing-without-words>

47

BITS Pilan

47



innovate **achieve** **lead**

Visualization Design Objectives

2. Justifying the selection of everything we do

- We should consider the idea of deliberate design, which means that the inclusion, exclusion, and execution of every single mark, characteristic, and design feature is done for a reason
- It is also important to make sure that any visual property that is included, but does not represent data, such as shading, labels, colors, and axes among other properties, should only be included to aid the process of visual perception, not hinder it.

48

BITS Pilan

48

Visualization Design Objectives



- 1. Strive for form and function (*style over substance*)**
- 2. Justifying the selection of everything we do**
- 3. Creating accessibility through intuitive design**
- 4. Never deceive the receiver**

49

BITS Pilani

Visualization Design Objectives



3. Creating accessibility through intuitive design

"Overload, clutter, and confusion are not attributes of information, they are failures of design."

- It is important to create a distinction between accessibility and immediacy. The speed with which you are able to read or interpret a visualization should be determined by the complexity of the subject and the purpose of the project, not by the ineffectiveness of design

50

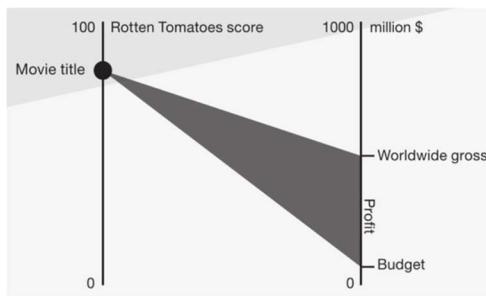
BITS Pilani

50

Visualization Design Objectives



3. Creating accessibility through intuitive design



51

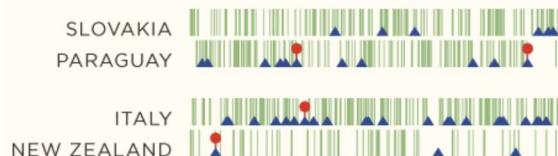
BITS Pilani

51

Visualization Design Objectives



3. Creating accessibility through intuitive design



52

BITS Pilani

52

Visualization Design Objectives

1. Strive for form and function (*style over substance*)
2. Justifying the selection of everything we do
3. Creating accessibility through intuitive design
4. Never deceive the receiver

53

BITS Pilani

53

Visualization Design Objectives

4. Never deceive the receiver

54

BITS Pilani

54

Visualization Design Objectives



4. Never deceive the receiver



Avoid any significant distortion of the truth.

55

BITS Pilani

55

Usage of Data Visualization



Exploratory Analysis

- Explore data to become familiar with data
- Dig through the data
- Find trends and relationships w.r.t. specific goals
- Helps determine analyses to apply to data

56

BITS Pilani

56

Usage of Data Visualization



Explanatory Analysis

- Explain outcomes or results of analysis
- Tell a story with data

57

BITS Pilani

Exploratory and Explanatory Analysis



- Exploratory: Try different variables and look for patterns.
- Explanatory: Explain the patterns discovered.

58

BITS Pilani

58

Exploratory and Explanatory Analysis-Qn



- With your data you need to show which district in your region is not meeting its sales goals. Is this an example of exploratory or explanatory analysis?
- Explanatory Analysis
- As this is a specific question you need to answer and you already recognize the patterns and need to explain it.

59

BITS Pilani

59

CONTACT SESSION 2 -PLAN



Contact Sessions(#)	List of Topic Title	Text Book
CS2	<ul style="list-style-type: none"> Understanding the context for data presentations 3 minute story Effective Visuals Gestalt principles of visual perception Visual Ordering Decluttering 	T1 Ch 1,2,3

60

BITS Pilani

60



Visualization Context

“Success in data visualization does not start with data visualization itself.”

-- Cole Knaflic

...understanding context sets solid foundation for data visualization creation

61

BITS Pilani

61



Visualization Context

Context includes :

❖ Who

To whom you are communicating?

❖ What

What do you want your audience to know or to do?

❖ How

How can you use data to help make your point?

62

BITS Pilani

62

Identifying Audience



Who?

- Who is audience? To whom you are communicating?
- How they perceive you?
- Help to find common ground that helps to convey your message

63

BITS Pilani

63

Identifying Audience



Who?

- ❖ Your audience
 - Knowing them place you in better position for communication
 - Be specific while identifying audience
 - Different content for different set of audience

64

BITS Pilani

64

Identifying Audience



Who?

❖ You

- How your audience perceive you?
- First time interaction or established relationship?
- Know you as expert or need to set credibility?

65

BITS Pilani

65

Know or Act



What?

❖ What do you want your audience to know or to do?

- Action
- Mechanism
- Tone

66

BITS Pilani

66



Know or Act

What?

❖ Action

- Make sure audience care about what you say
- You are subject matter expert – unique position to interpret the data and help lead people to understanding and take action
- If no action recommendation possible / feasible, then encourage discussion towards one

67

BITS Pilani

67



Know or Act

What?



❖ Mechanism

- How will you communicate to your audience?
- Determines level of control and level of detail
 - ✓ Live presentation
 - ✓ Written document
 - ✓ Slideument

68

BITS Pilani

68

Know or Act (cont...)

What? Mechanism

LIVE PRESENTATION WRITTEN DOC or EMAIL

amount of CONTROL you have

level of DETAIL needed

The "SLIDEUMENT"

69

BITs Pilani

69

Know or Act (cont...)

What?

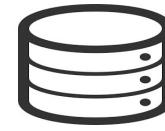
- ❖ Tone
 - ❑ What tone do you want your communication to set?
 - ✓ Celebrating success?
 - ✓ Lighting a fire to drive action?
 - ✓ Is topic light-hearted or serious?

70

BITs Pilani

70

Data



How?

- ❖ How can you use data to help make your point?
- ❖ What data is available?
 - Supporting evidence of the story

71

BITS Pilani

Context by Example



You are the advertising media analyst who is made responsible for providing recommendation on media spend of product

- Who : The marketing team that allocates funding for media advertisement for a product
- What : The current advertising campaign went well on TV but find very limited success in print media
- How : Illustrate success with data available through analysis of spends and product revenues(Can use live presentation)

72

BITS Pilani

72



3-minute Story

- Able to boil the “so-what” down to a paragraph and, ultimately, to a single, concise statement.
- If you had only three minutes to tell your audience what they need to know, what would you say?

73

BITS Pilani

73



Case Study

Imagine you are a fourth grade science teacher. You just wrapped up an experimental pilot summer learning program on science that was aimed at giving kids exposure to the unpopular subject. You surveyed the children at the onset and end of the program to understand whether and how perceptions toward science changed. You believe the data shows a great success story. You would like to continue to offer the summer learning program on science going forward.

Identify the Visualisation Context.(WHO,WHAT and HOW)

74

BITS Pilani

74

WHO

We want to communicate to is **THE BUDGET COMMITTEE**, which controls the funding we need, to continue the program.

75

BITS Pilani

WHAT

Demonstrate the success of the program and ask for a specific funding amount to continue to offer it

76

BITS Pilani

76



HOW

Use the data collected via survey at the onset and end of the program to illustrate the increase in positive perceptions of science before and after the pilot summer learning program.

77

BITS Pilani

77



Summary

Who: The budget committee that can approve funding for continuation of the summer learning program.

What: The summer learning program on science was a success; please approve budget of \$X to continue.

How: Illustrate success with data collected through the survey conducted before and after the pilot program.

78

BITS Pilani

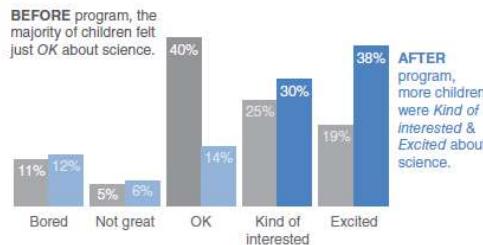
78

Summary



Pilot program was a success

How do you feel about science?



Based on survey of 100 students conducted before and after pilot program (100% response rate on both surveys).

79
BITS Pilani

79

3 -minute Story



3-minute story: A group of us in the science department were brainstorming about how to resolve an ongoing issue we have with incoming fourth-graders. It seems that when kids get to their first science class, they come in with this attitude that it's going to be difficult and they aren't going to like it. It takes a good amount of time at the beginning of the school year to get beyond that. So we thought, what if we try to give kids exposure to science sooner? Can we influence their perception? We piloted a learning program last summer aimed at doing just that. We invited elementary school students and ended up with a large group of second- and third-graders. Our goal was to give them earlier exposure to science in hopes of forming positive perception. To test whether we were successful, we surveyed the students before and after the program. We found that, going into the program, the biggest segment of students, 40%, felt just "OK" about science, whereas after the program, most of these shifted into positive perceptions, with nearly 70% of total students expressing some level of interest toward science. We feel that this demonstrates the success of the program and that we should not only continue to offer it, but also to expand our reach with it going forward.

BITS Pilani

80

BIG IDEA



The pilot summer learning program was successful at improving students' perceptions of science and, because of this success, we recommend continuing to offer it going forward; please approve our budget for this program.

BITS Pilani

81

Effective and Ineffective visuals



- The types of visualizations that are available
- **Textuals**
 - Just a simple text visualization
 - High Level Information
 - Conveys exactly what you need
 - KPI or key performance indicator

82

BITS Pilani

82



Textuals

Page Views:

10,267

83

BITS Pilani

83



Tables

- The types of visualizations that are available
- **Tables**
 - Conveys a lot of information
 - Conveys comparison across categories
 - Too big=less effective
 - A 2 X 2 table has a lot of power.

84

BITS Pilani

84

Tables



	Right Handed	Left Handed
Male	67	8
Female	65	14

85

BITS Pilani

85

Tables



Heavy borders

Group	Metric A	Metric B	Metric C
Group 1	S.X.	Y%	Z.ZZZ
Group 2	S.X.	Y%	Z.ZZZ
Group 3	S.X.	Y%	Z.ZZZ
Group 4	S.X.	Y%	Z.ZZZ
Group 5	S.X.	Y%	Z.ZZZ

Light borders

Group	Metric A	Metric B	Metric C
Group 1	S.X.	Y%	Z.ZZZ
Group 2	S.X.	Y%	Z.ZZZ
Group 3	S.X.	Y%	Z.ZZZ
Group 4	S.X.	Y%	Z.ZZZ
Group 5	S.X.	Y%	Z.ZZZ

Minimal borders

Group	Metric A	Metric B	Metric C
Group 1	S.X.	Y%	Z.ZZZ
Group 2	S.X.	Y%	Z.ZZZ
Group 3	S.X.	Y%	Z.ZZZ
Group 4	S.X.	Y%	Z.ZZZ
Group 5	S.X.	Y%	Z.ZZZ

86

BITS Pilani

86

Heat Maps



- The types of visualizations that are available
- Tables-Heat Maps**
 - Colorful version of a table,
 - It takes information that you have in a table and then colors it based on a certain set of parameters.
 - Use color to visually emphasize information

87

BITS Pilani

87

Heat Maps



CUSTOMER CONTACTS PER EMPLOYEE PER WEEKDAY

	Charles	Toby	Amanda	Troy	Sam	Tammy	Matt	Jason	Amy	Alice	
Friday	67	48	52	16	115	120	96	30	84	91	
Thursday	24	117	64	19	117	6	98	32	64	48	
Wednesday	8	78	123	114	8	12	88	82	123	31	
Tuesday	19	58	15	132	5	32	44	1	97	114	
Monday	10	92	35	72	38	88	13	31	85	47	

88

BITS Pilani

88

Effective and Ineffective visuals

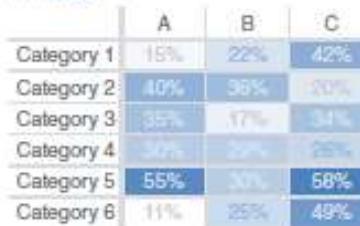


Table

	A	B	C
Category 1	15%	22%	42%
Category 2	40%	36%	20%
Category 3	35%	17%	34%
Category 4	30%	29%	26%
Category 5	55%	30%	58%
Category 6	11%	25%	49%

Heatmap

LOW HIGH



89

BITS Pilani

89

GRAPHS



- Graphs interact with our visual system
- Faster at processing information

90

BITS Pilani

90

Gestalt Principles of Visual Perception

Gestaltism



- Identifying which elements in our visuals are **signal**
- Which might be **noise**
- **How individuals perceive order in the world around them**

91

BITS Pilani

Gestalt Principles of Visual Perception



- The human brain loves simplicity and it tends to process simple patterns — patterns that are regular, even, and orderly — faster than patterns that are more complex

92

BITS Pilani

92

Gestalt Principles of Visual Perception



Principles

- Proximity
- Similarity
- Enclosure
- Closure
- Continuity
- Connection.
- Focal point
- Figure and ground

93

BITS Pilani

93

Gestalt Principles of Visual Perception



- Gestalt Law of Proximity



94

BITS Pilani

94

Gestalt Principles of Visual Perception



Gestalt Law of Proximity (example)



95

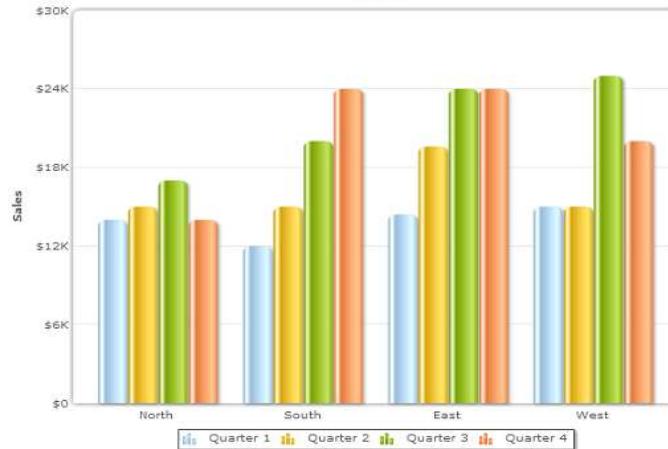
BITS Pilani

95

Gestalt Principles of Visual Perception



Quarterly Regionwise Sales (in USD)
(2012)



96

BITS Pilani

96

Gestalt Principles of Visual Perception



- Gestalt Law of Similarity



97

BITS Pilani

97

Gestalt Principles of Visual Perception



- Gestalt Law of Similarity



98

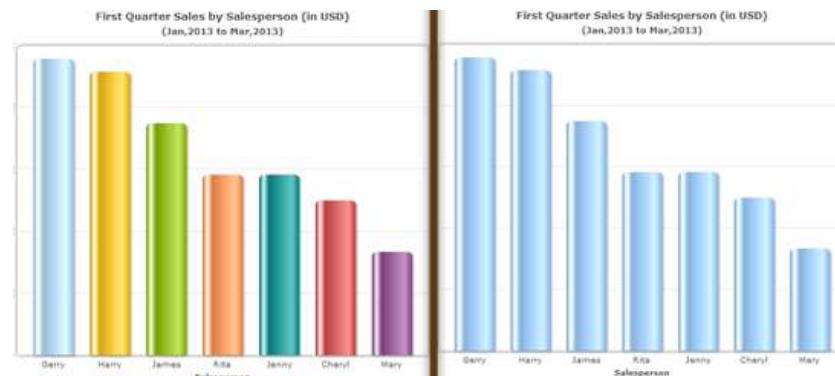
BITS Pilani

98

Gestalt Principles of Visual Perception



- Gestalt Law of Similarity (example)



99

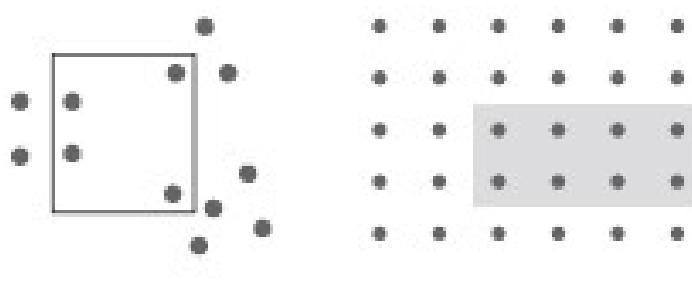
BITS Pilani

99

Gestalt Principles of Visual Perception



- Gestalt Law of Enclosure

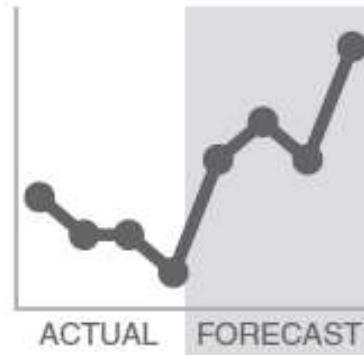


100

BITS Pilani

100

Gestalt Principles of Visual Perception



101

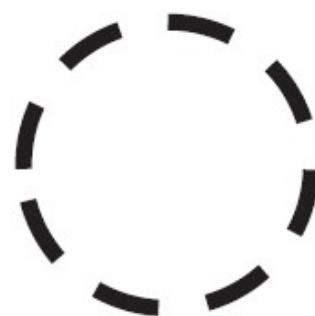
BITS Pilani

101

Gestalt Principles of Visual Perception



- Gestalt Law of Closure



102

BITS Pilani

102

Gestalt Principles of Visual Perception



- Gestalt Law of Closure



103

BITS Pilani

103

Gestalt Principles of Visual Perception



- Gestalt Law of Continuity



104

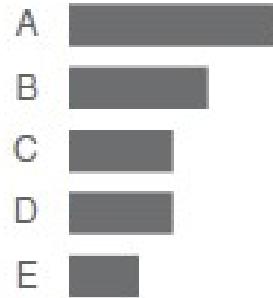
BITS Pilani

104

Gestalt Principles of Visual Perception



- Gestalt Law of Continuity



105

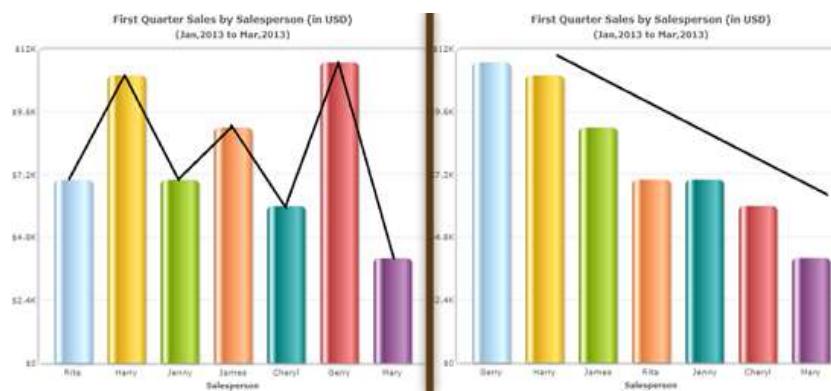
BITS Pilani

105

Gestalt Principles of Visual Perception



Gestalt Law of Continuity (example)



106

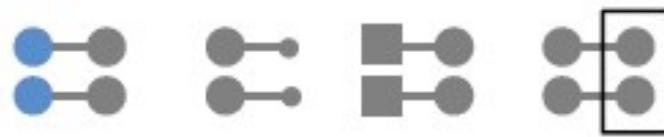
BITS Pilani

106

Gestalt Principles of Visual Perception



- Gestalt Law of Connection



107

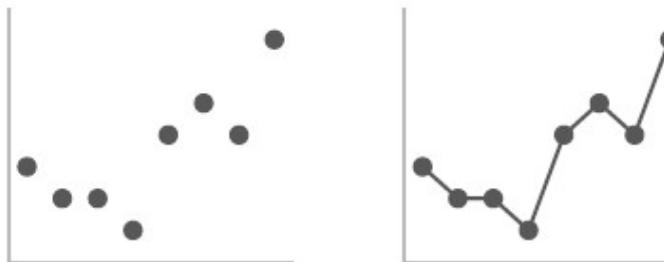
BITS Pilani

107

Gestalt Principles of Visual Perception



- Gestalt Law of Connection



108

BITS Pilani

108

Gestalt Principles of Visual Perception



- Gestalt Law of Focal Point



109

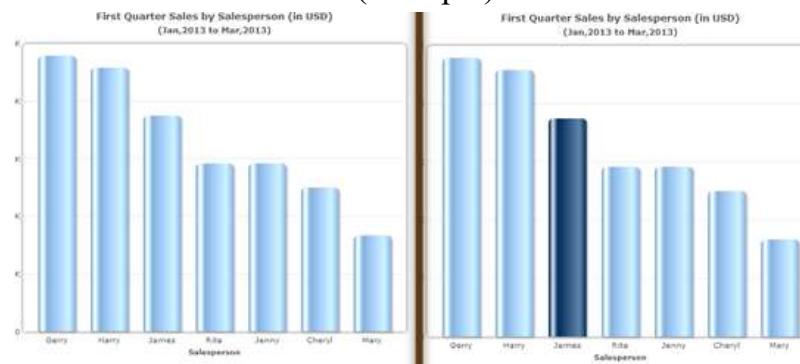
BITS Pilani

109

Gestalt Principles of Visual Perception



Gestalt Law of Focal Point (example)



110

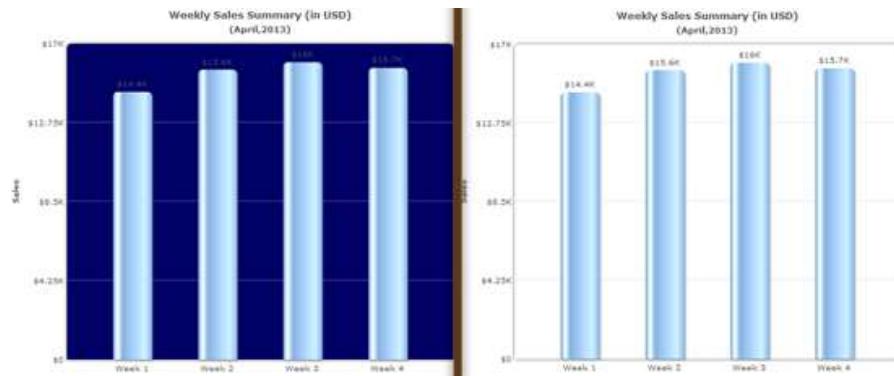
BITS Pilani

110

Gestalt Principles of Visual Perception



- Gestalt Law of Figure/Ground

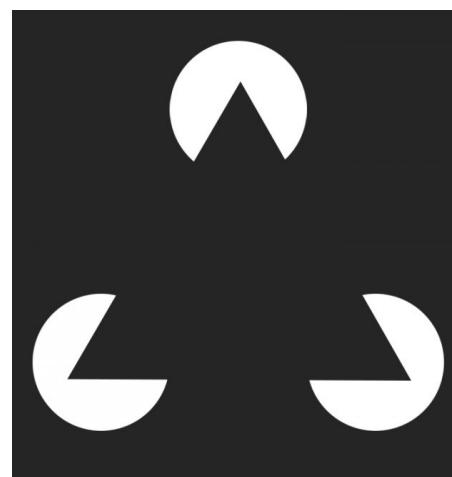


111

BITS Pilani

111

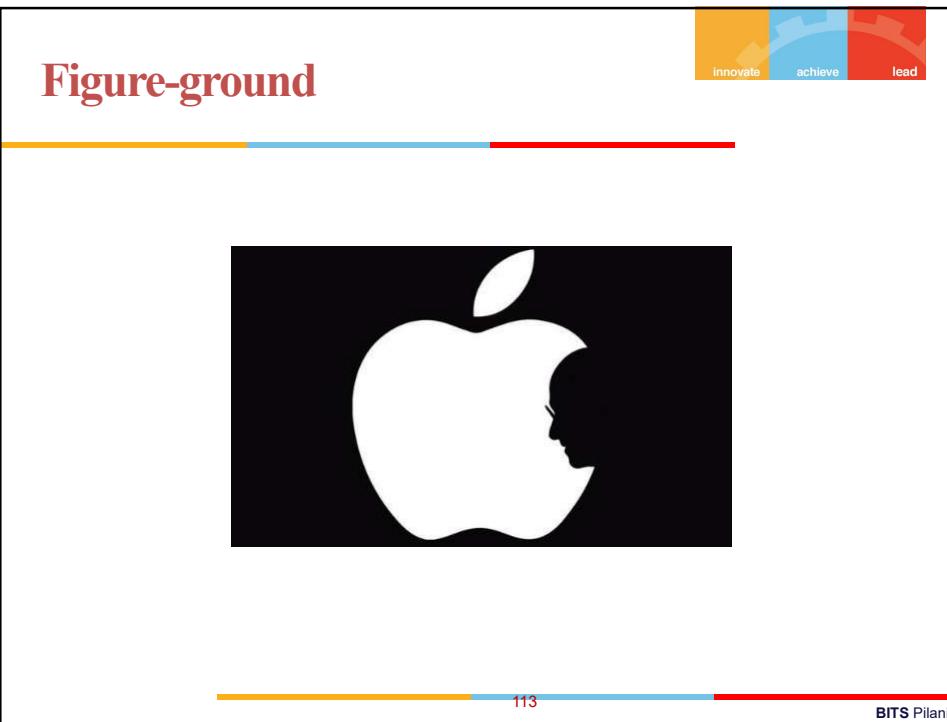
WHAT DO YOU SEE?



112

BITS Pilani

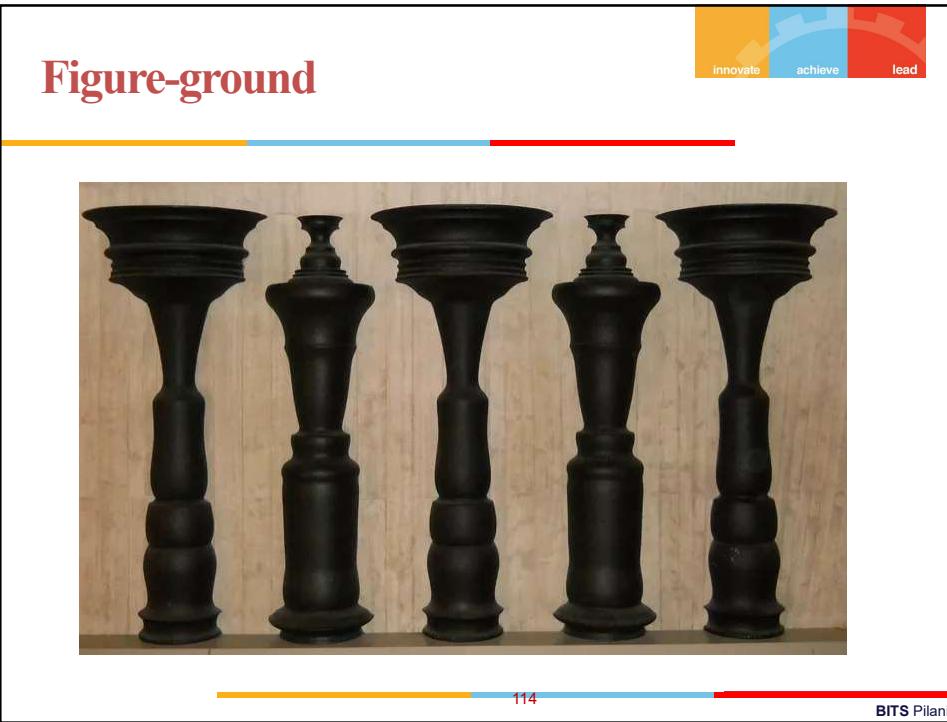
112



113



BITS Pilani



114



BITS Pilani

114



Figure AND Ground

- Blurriness
- Size
- Separation

115

BITS Pilani

115



Question

Which of the following Gestalt principles that we've discussed thus far deal with the contrast of information?

- Proximity
- Closure
- Similarity
- Enclosure

116

BITS Pilani

116



Question

Which of the following Gestalt principles that we've discussed thus far deal with the contrast of information?

- Proximity
- Closure
- Similarity
- **Enclosure**

- Enclosure is about using contrast in order to differentiate between information

117

BITS Pilani

117



Cognitive Load

- **It is the amount of mental effort that we use to get the information that we need.**
- signal-to-noise ratio

118

BITS Pilani

118



Cognitive Load

- There are three types of cognitive load
 - Intrinsic,
 - Extraneous, and
 - Germane.
- Different tasks need different amount of thoughts and attention

119

BITS Pilani

119



Cognitive Load

- **Intrinsic Cognitive Load**
- The amount of memory that we need to understand something.
- **Extraneous cognitive load**
- Refers to the amount of **extra** brain power required to analyze and process information.
- **Germane cognitive load**
- A way for the brain to look for patterns to develop context.

120

BITS Pilani

120

Question



- Which type of the cognitive loads just discussed pertains to the baseline level amount of brain power needed to understand an idea?
- Germane Cognitive Load
- Intrinsic Cognitive Load
- Extraneous Cognitive Load

121

BITS Pilani

121

Question



- Which type of the cognitive loads just discussed pertains to the baseline level amount of brain power needed to understand an idea?
- Germane Cognitive Load
- **Intrinsic Cognitive Load**
- Extraneous Cognitive Load

122

BITS Pilani

122

Clutter



- Presence of unnecessary elements on the screen/page that
 - Occupies space
 - Reduces understanding
 - Increases complexity
 - *KNOW IT WHEN YOU SEE IT*

123

BITS Pilani

123

Declutter



- How to eliminate clutter?
 - Gestalt Principles of Visual Perception
 - Identify signal (what needs to be communicated)
 - Identify noise (what needs to be eliminated)

124

BITS Pilani

124

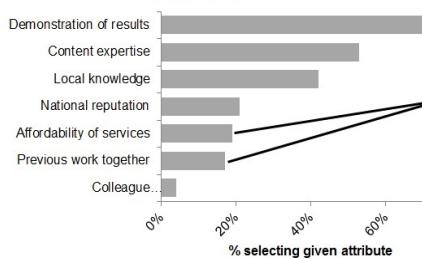


Declutter-Visual Ordering

Lack of Visual Order (Example)

Demonstrating effectiveness is most important consideration when selecting a provider

In general, what attributes are the most important to you in selecting a service provider?
(Choose up to 3)



Survey shows that demonstration of results is the single most important dimension when choosing a service provider.

Affordability and experience working together previously, which were hypothesized to be very important in the decision making process, were both cited less frequently as important attributes.

Data source: xyz; includes N number of survey respondents. Note that respondents were able to choose up to 3 options.

125

BITS Pilani

125

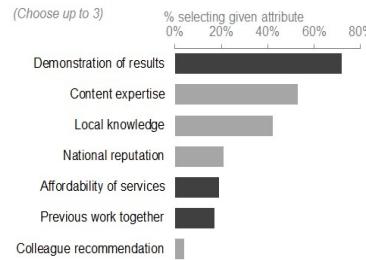


Declutter-Visual Ordering

Lack of Visual Order (Example Improved)

Demonstrating effectiveness is most important consideration when selecting a provider

In general, what attributes are the most important to you in selecting a service provider?
(Choose up to 3)



Survey shows that demonstration of results is the single most important dimension when choosing a service provider.

Affordability and experience working together previously, which were hypothesized to be very important in the decision making process, were both cited less frequently as important attributes.

Data source: xyz; includes N number of survey respondents.
Note that respondents were able to choose up to 3 options.

126

BITS Pilani

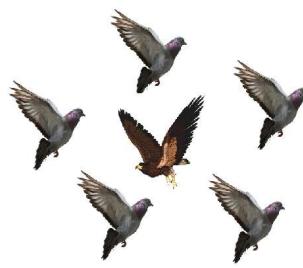
126



Visual Ordering

Contrast

"It's easy to spot a hawk in a sky full of pigeons, but as the variety of birds increases, that hawk becomes harder and harder to pick out." - Colin Ware, Information Visualization : Perception for Design



127

BITS Pilani

127



Visual Ordering-Contrast



128

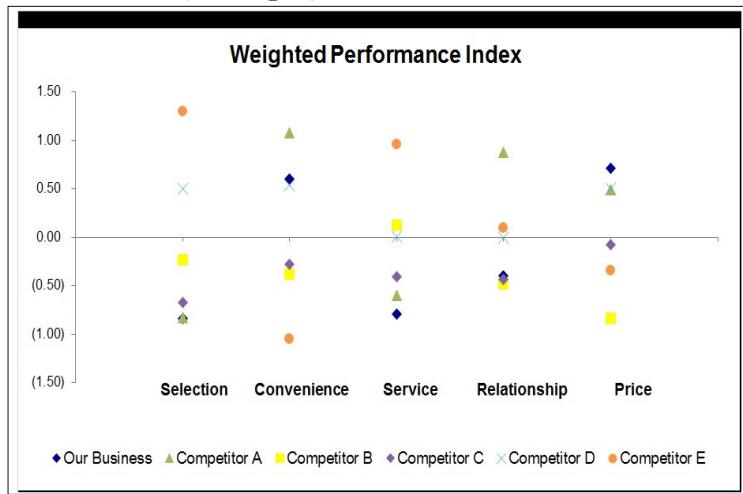
BITS Pilani

128



Visual Ordering

Lack of Contrast (Example)



BITS Pilani

129



Visual Ordering

- Lack of Contrast (Example)



- What's going on?
 - Lot of elements
 - Each one trying to grab attention
 - Difficult to focus attention

130

BITS Pilani

130

Visual Ordering (cont...)



Lack of Contrast (Example Improved)

Performance overview

- Our business
- Competitor A
- Competitor B
- Competitor C
- Competitor D
- Competitor E



131

BITS Pilani

131

Case Study-Strategic use of contrast Study



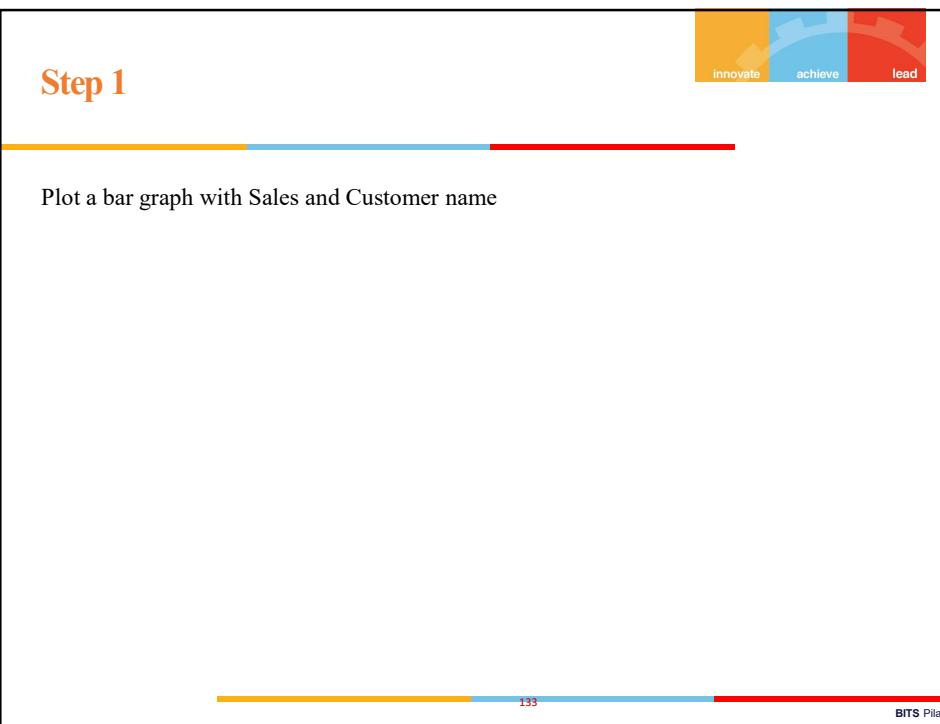
The question that we're going to answer today is whether or not “profit and sales go hand in hand.”

The question was asked by the CFO of the company. She doesn't have time to try to understand complex visualizations. She understands data very well, but is now in meetings all day, and really isn't in a position to do a deep dive on the data. What she needs is the ability to quickly get at what the issue is, if there is an issue with profit and sales from a list of people

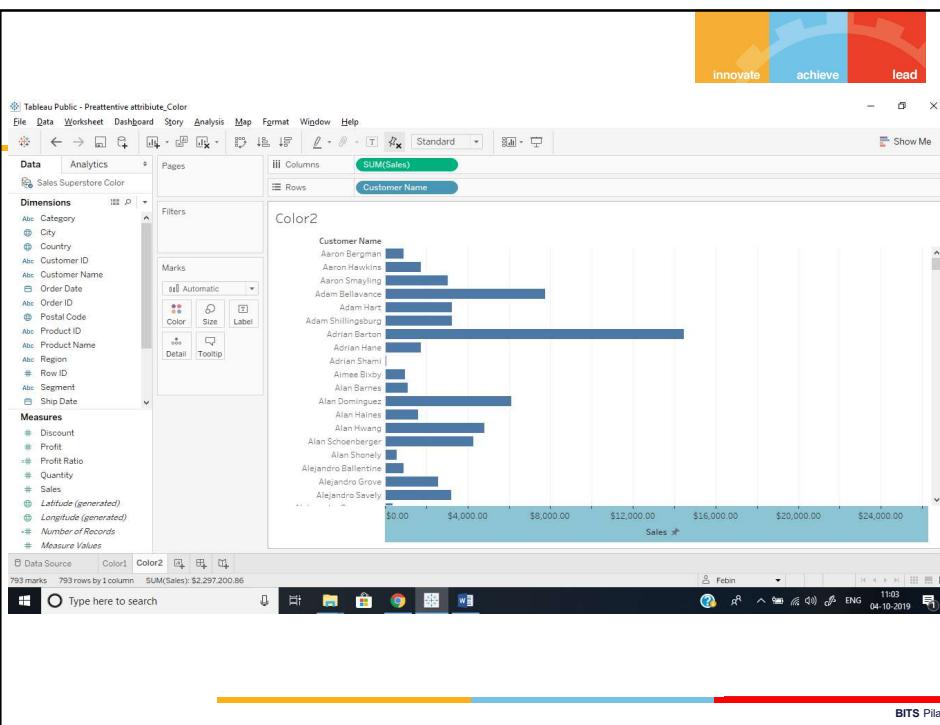
132

BITS Pilani

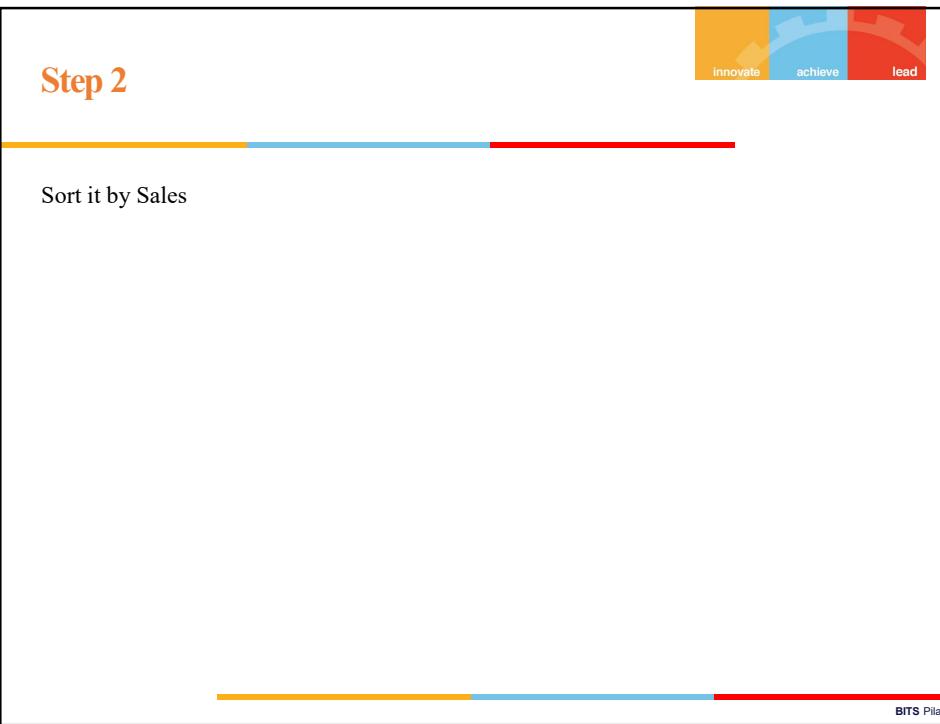
132



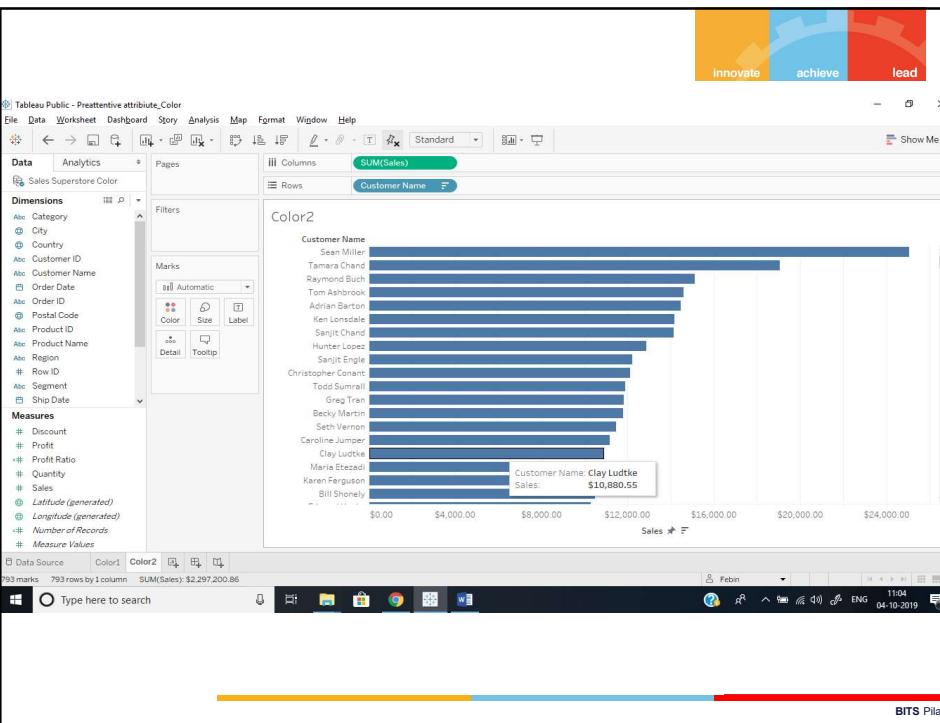
133



134



135



136

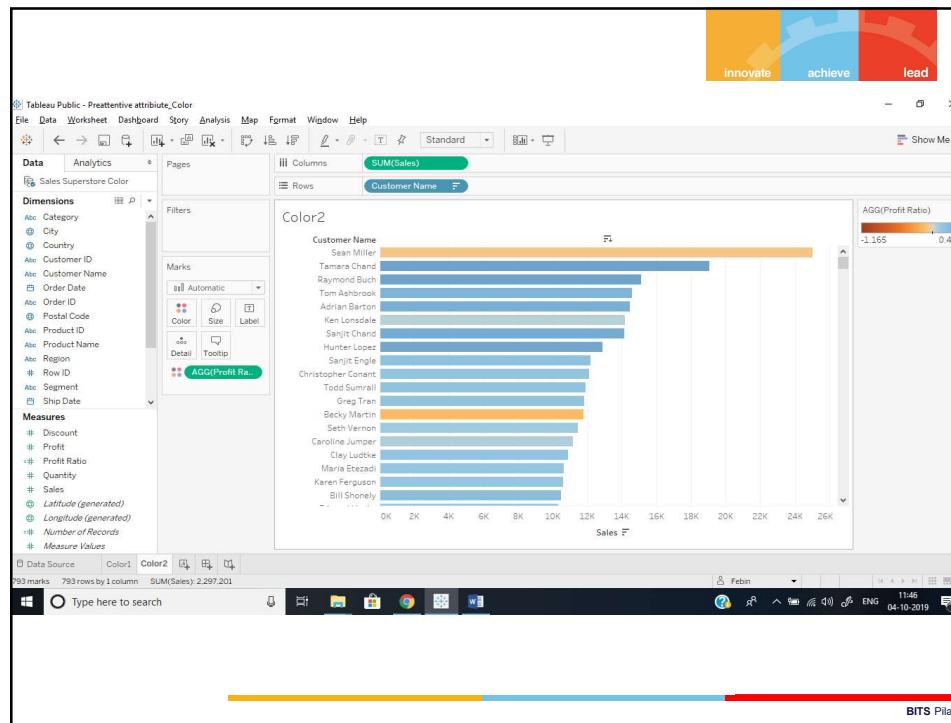
Step 3



Add Profit Ratio to the graph with Color Mark

BITS Pilani

137



138

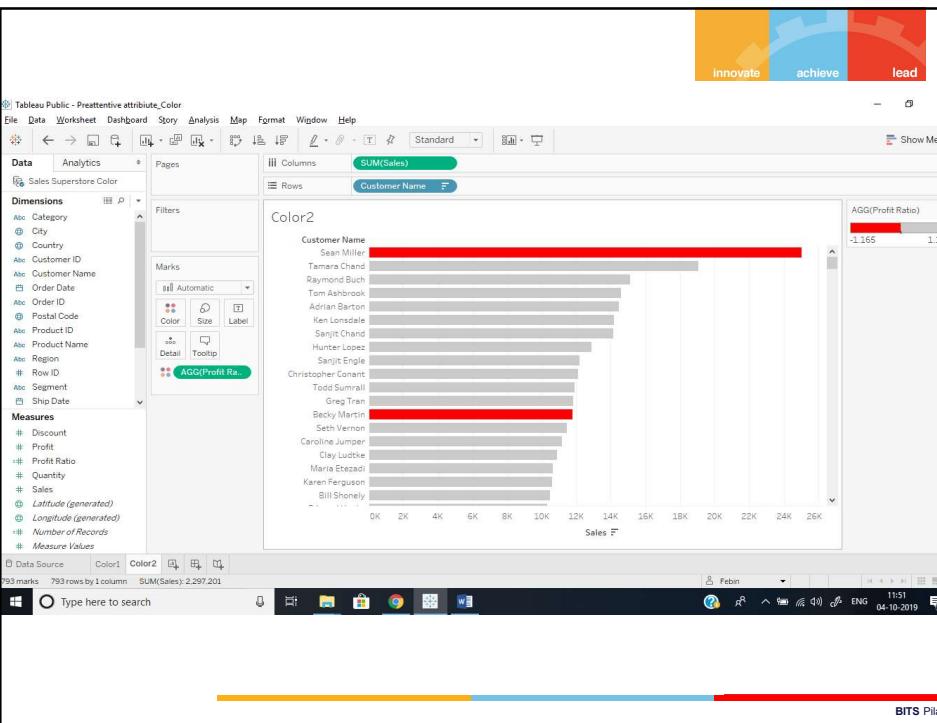
Step 4

Since It needs to be clear to the CFO whose profits are less than zero. In other words, who is not making money? ,we need only 2 Colors.

The color is red if the profit was less than zero, gray, a very light gray, if they made money.

BITS Pilani

139

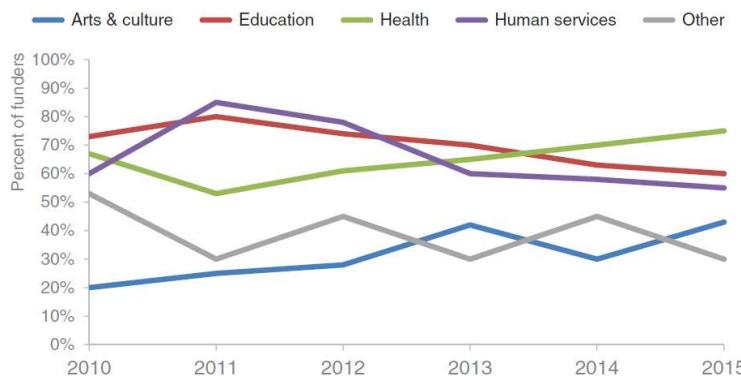


140

Case Study -Clutter



Types of non-profits supported by the corporate workforce



Data is self-reported by funders; percents sum to greater than 100 because respondents can make multiple selections.

- **To Do:** You are a Data Science Consultant. Convey to your CSR leadership team where they should plan to focus the organization's CSR activity in the future

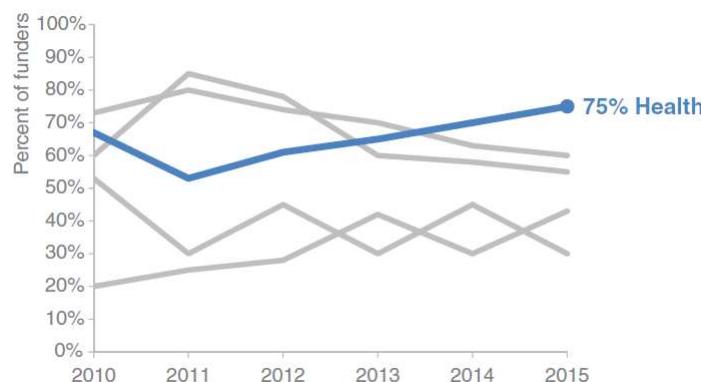
BITS Pilani

141

Case Study – probable solutions

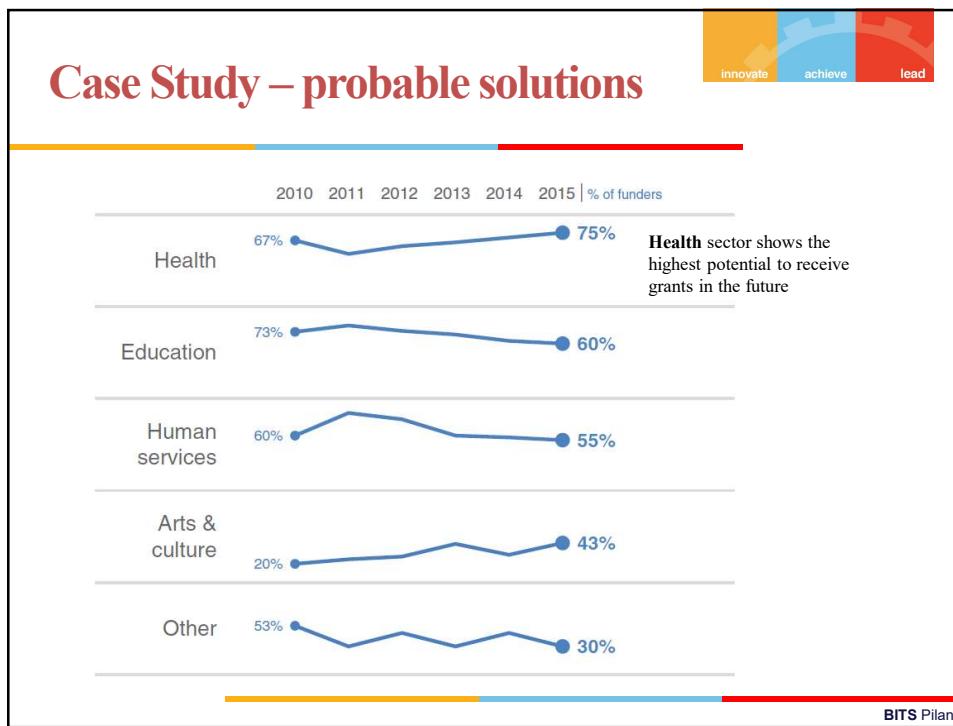


Health sector shows the highest potential to receive grants in the future

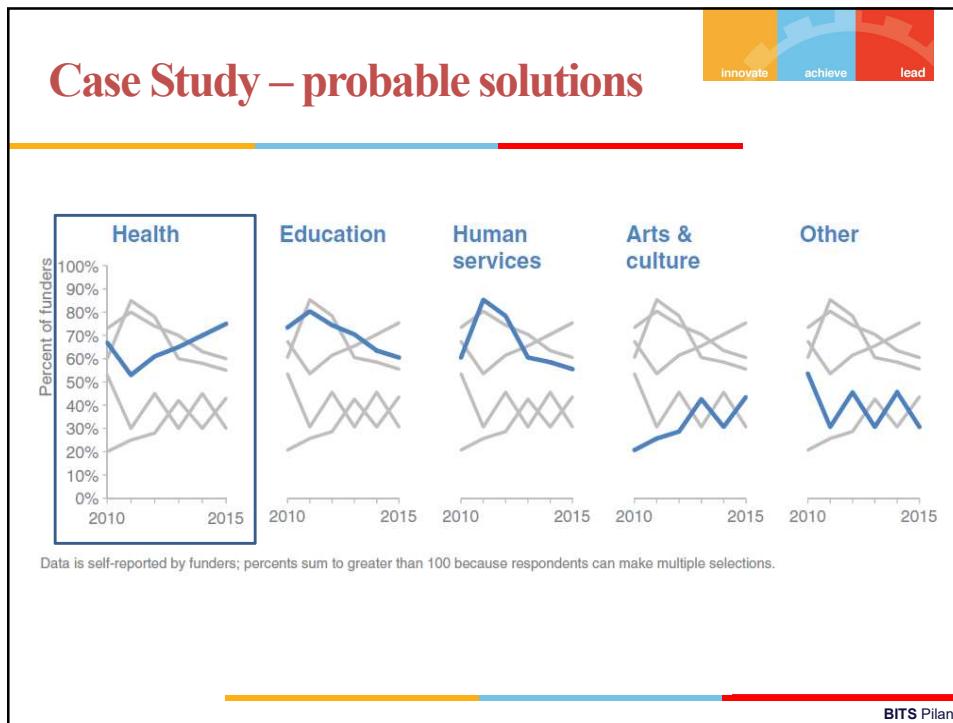


BITS Pilani

142



143



144



Case Study - Clutter

The visualization that we're going to de-clutter is from the United States Department of Agriculture or USDA.

Because the information that the USDA has is generally public, we happen to have the underlying data set for this visualization

www.ers.usda.gov/mediaImport/1706918/msagroceryinflation.xls

In the next slide is the image from the US Department of Agriculture

145

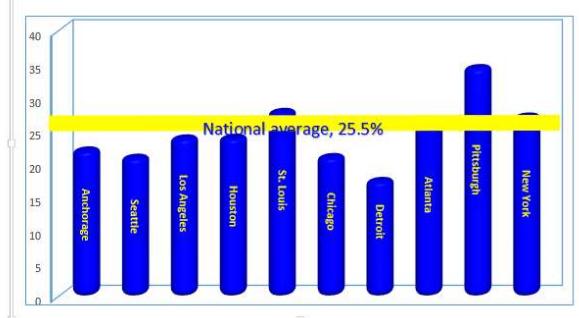
BITS Pilani

145



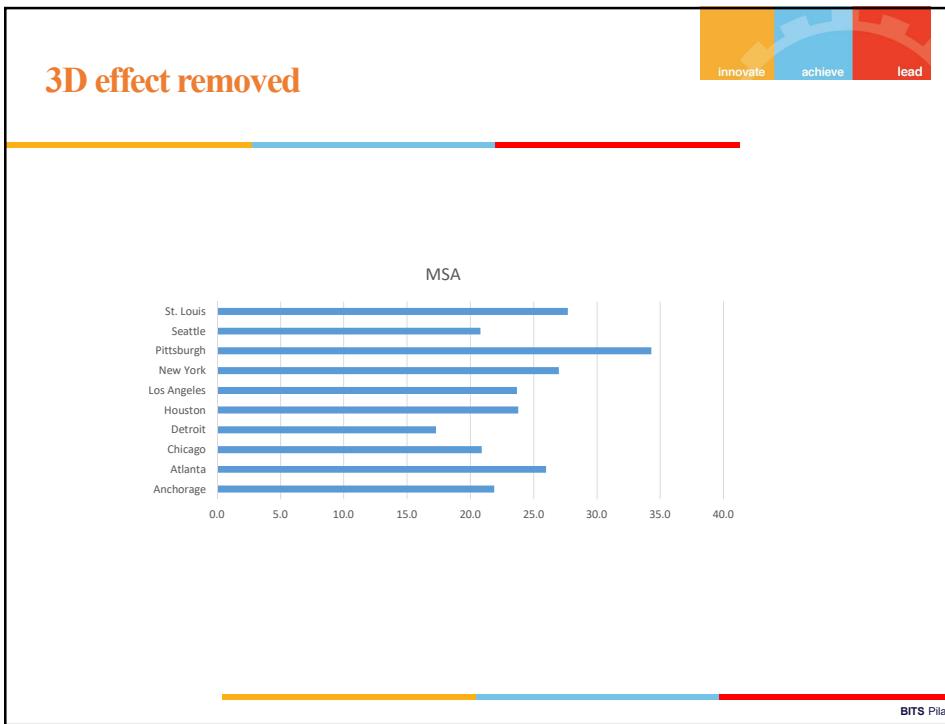
Case study

Retail food price inflation by MSA



BITS Pilani

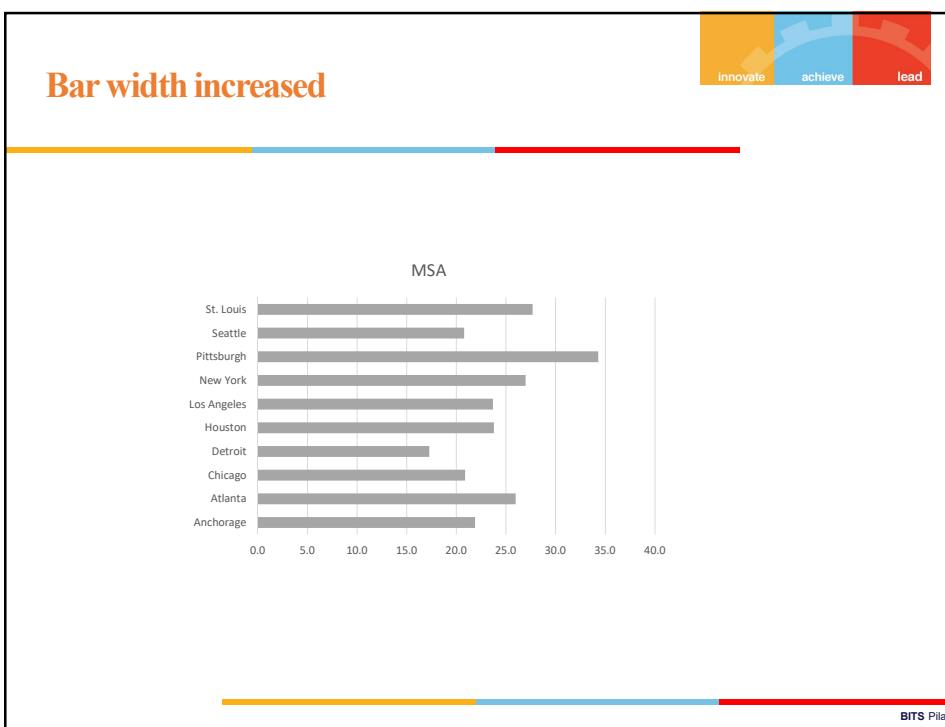
146



147



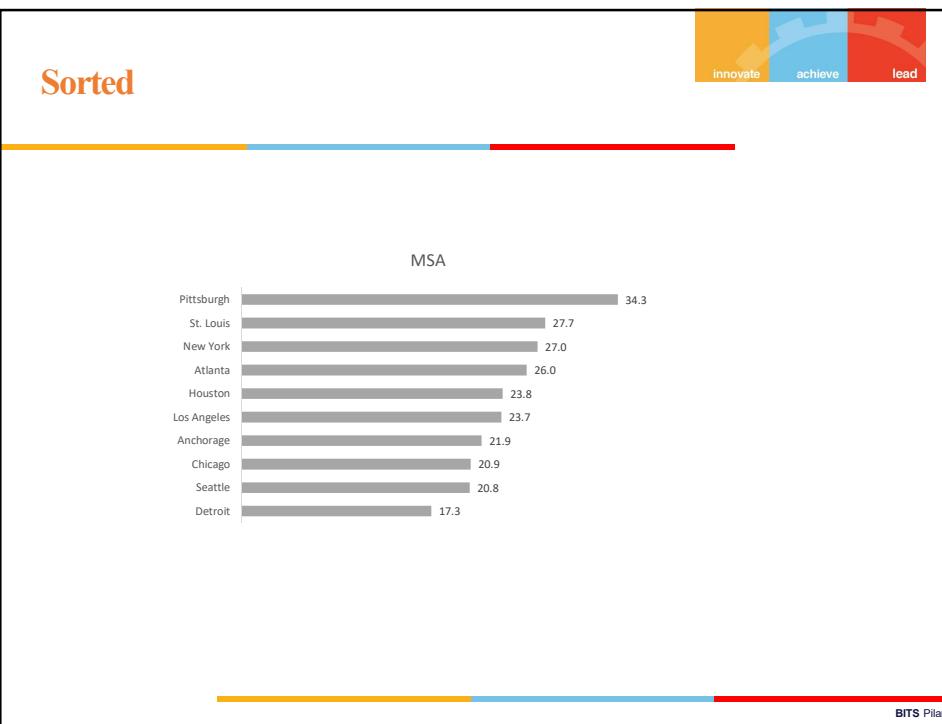
148



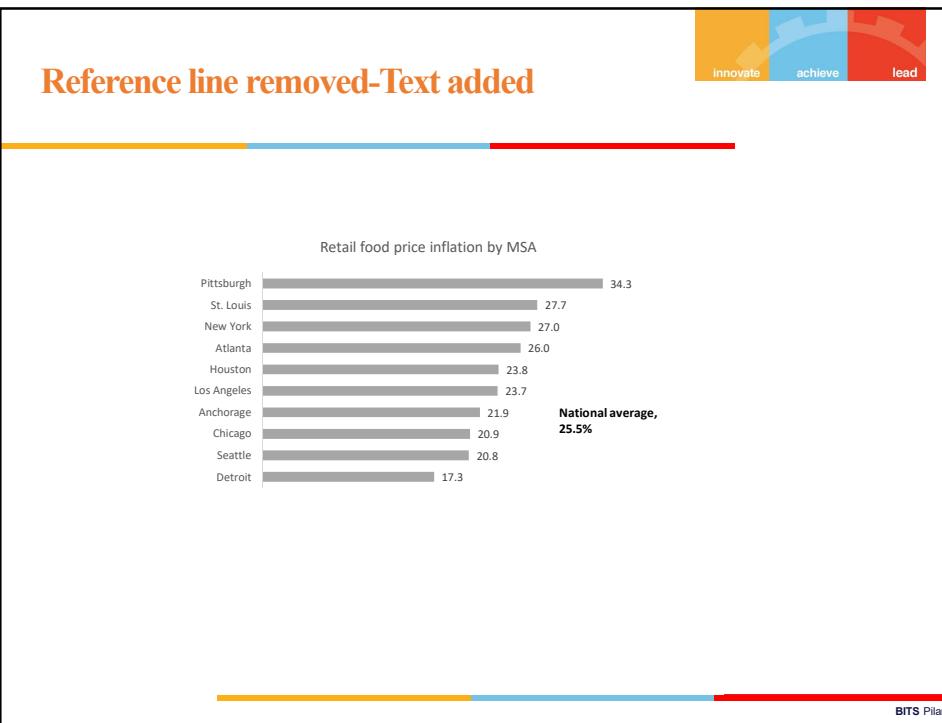
149



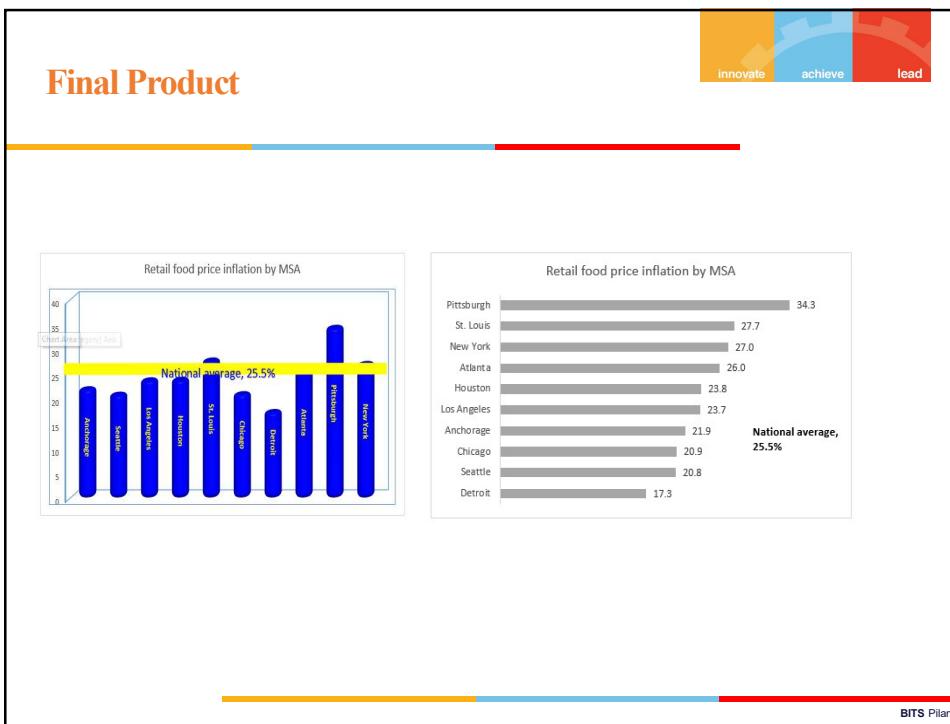
150



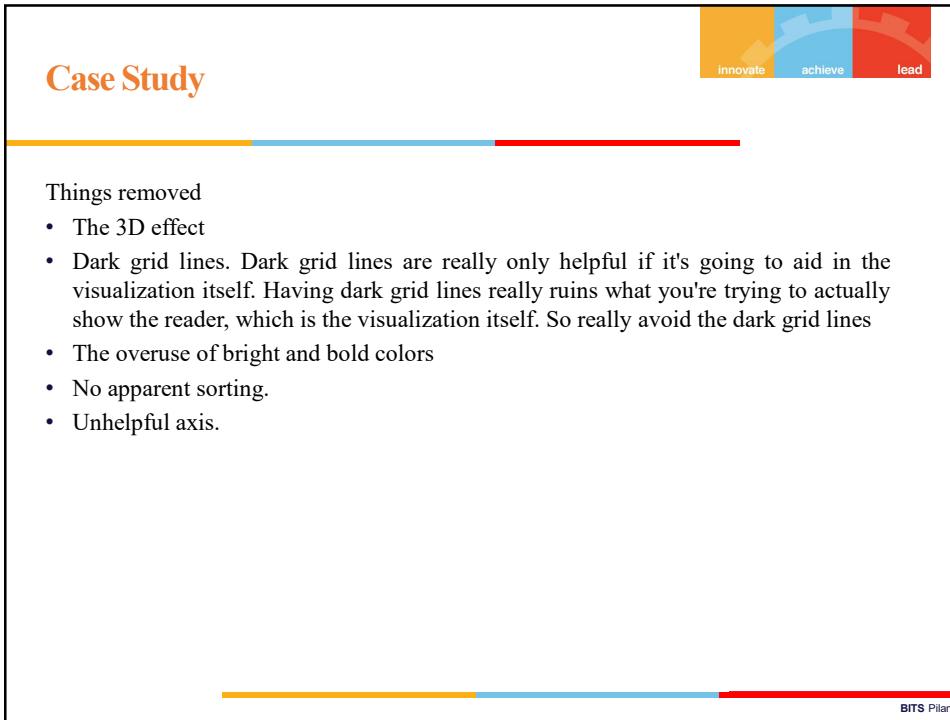
151



152



153



154

CASE STUDY 3 - Scenario



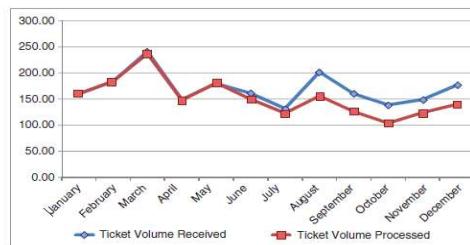
Imagine that you manage an information technology (IT) team. Your team receives tickets, or technical issues, from employees. In the past year, you've had a couple of people leave and decided at the time not to replace them. You have heard a rumbling of complaints from the remaining employees about having to "pick up the slack." You've just been asked about your hiring needs for the coming year and are wondering if you should hire a couple more people. First, you want to understand what impact the departure of individuals over the past year has had on your team's overall productivity. You plot the monthly trend of incoming tickets and those processed over the past calendar year. You see that there is some evidence your team's productivity is suffering from being short-staffed and now want to turn the quick-and-dirty visual you created into the basis for your hiring request.

155

BITS Pilani

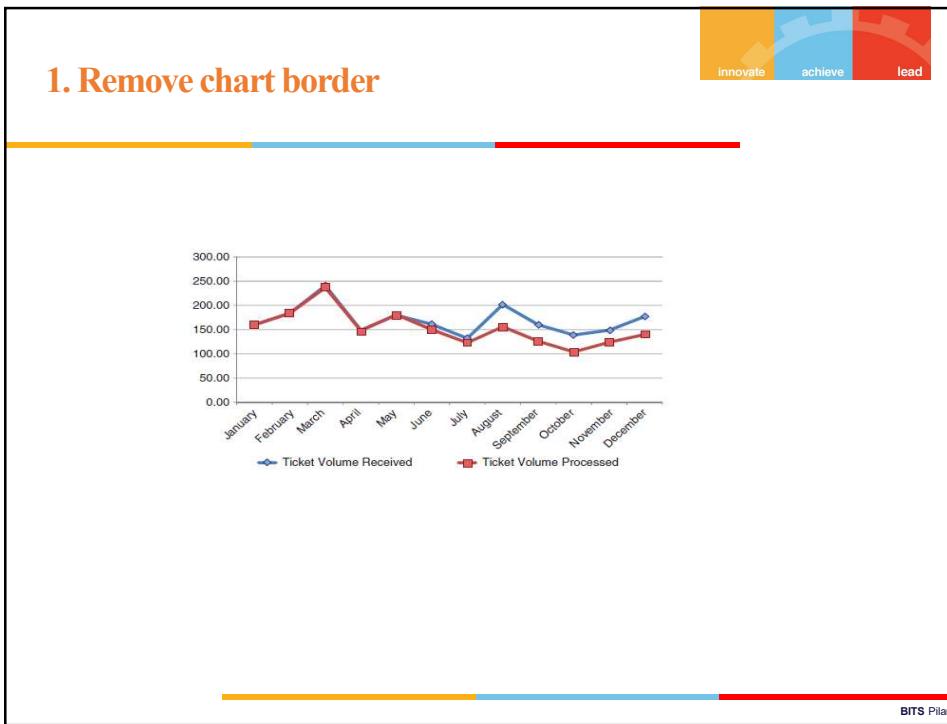
155

Case study

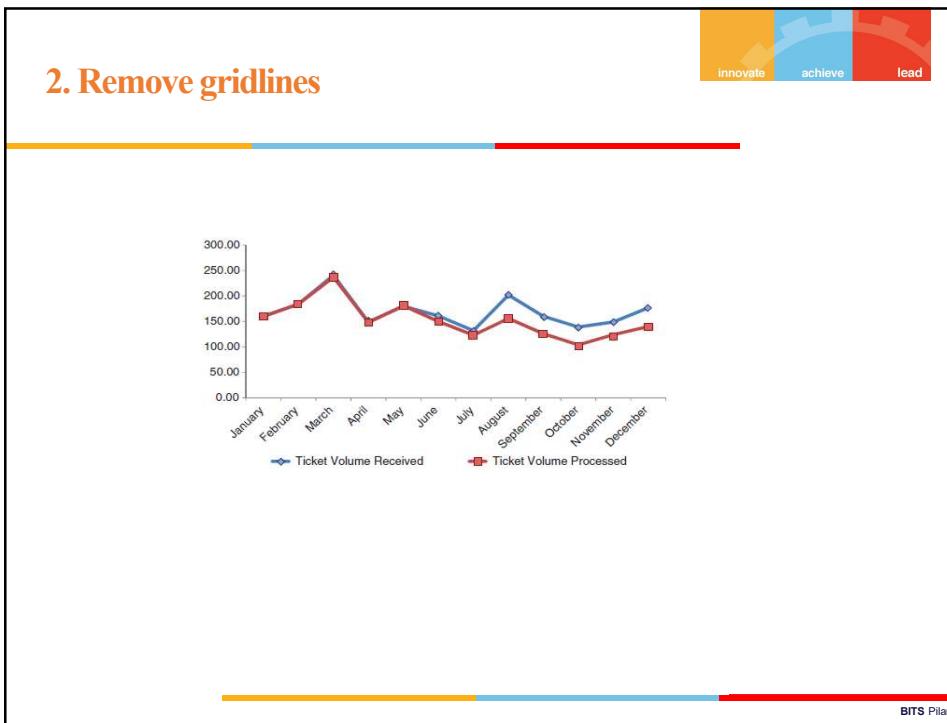


BITS Pilani

156

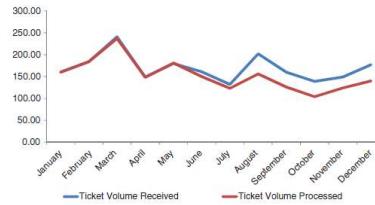


157



158

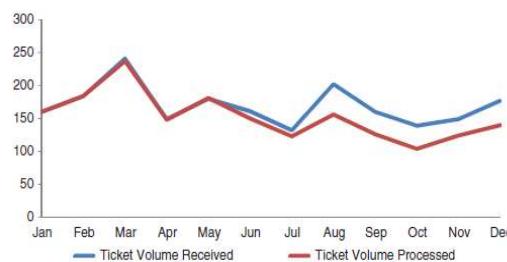
3. Remove data markers



BITS Pilani

159

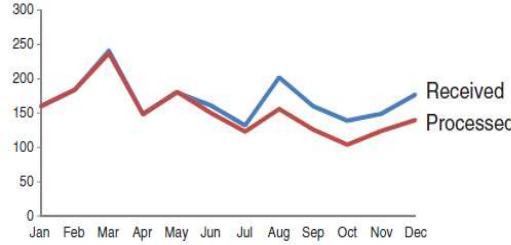
4. Clean up axis labels



BITS Pilani

160

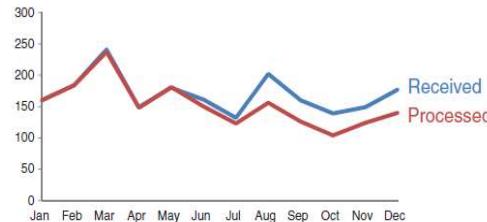
5. Label data directly



BITS Pilani

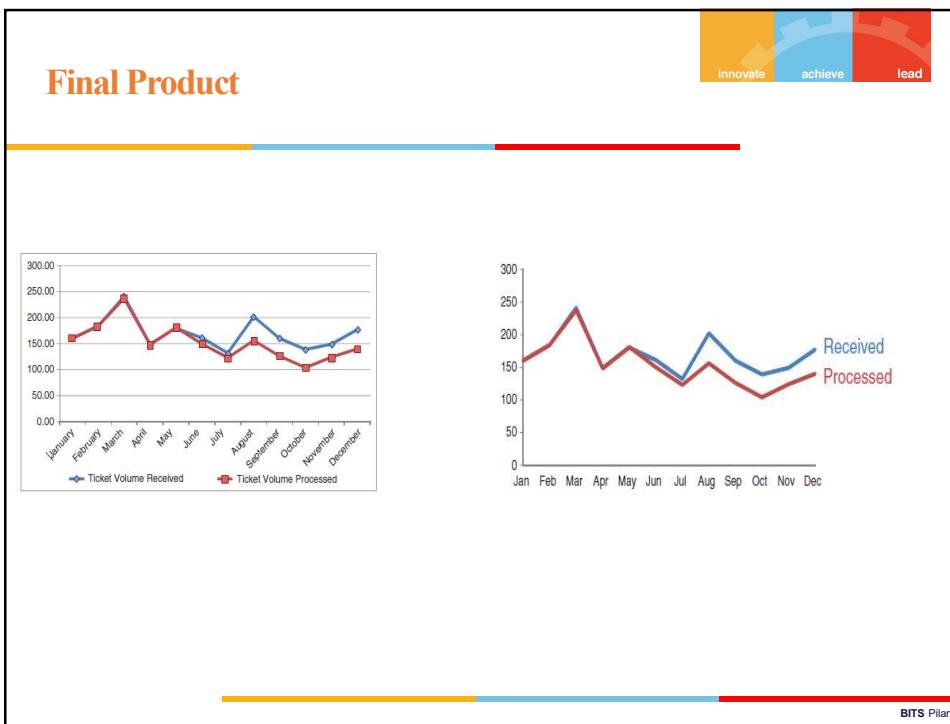
161

6. Leverage consistent color



BITS Pilani

162



163

The slide features a title "CONTACT SESSION 3 -PLAN" in red at the top left. At the top right is a decorative bar divided into three colored segments: yellow (innovate), light blue (achieve), and red (lead). Below the title is a table with three columns: Contact Sessions(#), List of Topic Title, and Text Book.

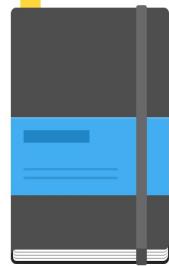
Contact Sessions(#)	List of Topic Title	Text Book
CS3	<ul style="list-style-type: none"> • Decluttering • Pre-attentive attributes in text and graphs • Visualisation Design concepts 	T1 Ch 3 T1 Ch 4 T1 Ch 5

164

BITS Pilani

164

Design Concepts



Traditional Design Concepts

- Affordances
- Accessibility
- Aesthetics
- Acceptance

165

BITS Pilani

165

Form Follows Function



Function

- Think about what our audience be able to do with data

Form

- Create a visualization

166

BITS Pilani

166

Affordances



“A situation where an object’s sensory characteristics intuitively imply its functionality and use”

-- CrowdCube

167

BITS Pilani

Affordances



Affordances

- ❑ Inherent usability aspect in the product
- ❑ How to interact with or operate the product?
- ❑ With sufficient presence of affordance, good design fades into background
- ❑ Example, teapot has a handle, door has a handle

168

BITS Pilani

168




Affordances

Lessons

- Highlight the important stuff
- Eliminate distractions
- Create clear hierarchy of information

169

BITS Pilani



Affordances

Highlighting

- Focus on a fraction of overall visual (10% highlighting)
- Guidelines
 - Bold, italics, underlining :
 - ✓ Use for titles, labels, captions and short word sequences
 - ✓ Bolding preferred as it adds little noise
 - ✓ Use underlining sparingly as it add lot of noise
 - Case, typeface :
 - ✓ for short word sequences use uppercase
 - ✓ Can be used for titles, labels, keywords
 - ✓ Avoid different fonts as it disrupts aesthetics

170

BITS Pilani

Affordances



Highlighting

- ❑ Guidelines
 - ❑ Color :
 - ✓ Effective highlighting technique
 - ✓ Use sparingly
 - ❑ Size :
 - ✓ Way to attract attention
 - ✓ Signals relative importance
 - ❑ Blinking / Flashing:
 - ✓ Avoid or
 - ✓ Use only to indicate highly critical information that requires immediate response

171 BITS Pilani

171

Affordances

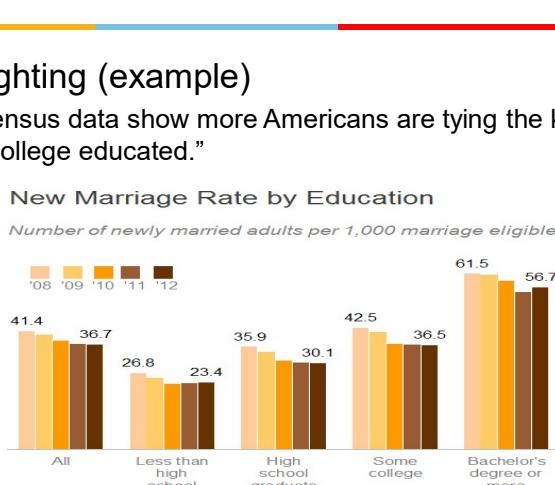


Highlighting (example)

“New census data show more Americans are tying the knot, but mostly it’s the college educated.”

New Marriage Rate by Education

Number of newly married adults per 1,000 marriage eligible adults



Education Level	'08	'09	'10	'11	'12
All	41.4	36.7	36.7	36.7	36.7
Less than high school	26.8	23.4	23.4	23.4	23.4
High school graduate	35.9	30.1	30.1	30.1	30.1
Some college	42.5	36.5	36.5	36.5	36.5
Bachelor's degree or more	61.5	56.7	56.7	56.7	56.7

Note: Marriage eligible includes the newly married plus those widowed, divorced or never married at interview.
 Source: US Census
 Adapted from PEW RESEARCH CENTER

172 BITS Pilani

172

Affordances

innovate achieve lead

Highlighting (example)
Changing the use of color completely redirect the focus

New Marriage Rate by Education
Number of newly married adults per 1,000 marriage eligible adults

Education Level	Marriage Rate (2008-2011)
All	41.4
Less than high school	26.8
High school graduate	35.9
Some college	42.5
Bachelor's degree or more	61.5
	36.7
	23.4
	30.1
	36.5
	56.7

Note: Marriage eligible includes the newly married plus those widowed, divorced or never married at interview.
Source: US Census
Adapted from PEW RESEARCH CENTER

173

BITs Pilani

Affordances

innovate achieve lead

Eliminate Distractions

“You know you have achieved perfection, not when have nothing to add, but when you have nothing to take away.”
--Antoine de Saint-Exupery,
Airman’s Odyssey

... what to cut or de-emphasize can be more important than what to include

174

BITs Pilani

Affordances



Eliminate Distractions

- Focus on clutter and context
- Considerations
 - ✓ Not all data are equally important
 - ✓ When detail is not needed, summarize
 - ✓ Ask: would eliminating this change anything?
 - ✓ Push necessary, but non-message conveying items to background

175

BITS Pilani

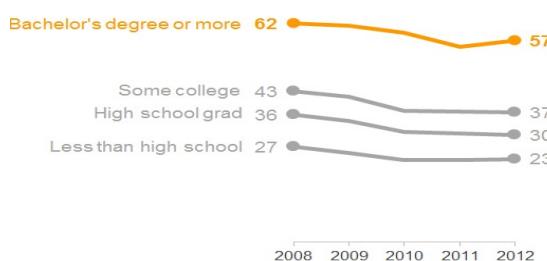
175

Affordances



Eliminate Distractions (example)

New marriage rate by education
Number of newly married adults per 1,000 marriage eligible adults



2008 2009 2010 2011 2012

Note: Marriage eligible includes the newly married plus those widowed, divorced or never married at interview.

Source: US Census
Adapted from PEW RESEARCH CENTER

176

BITS Pilani

176

Affordances

Eliminate Distractions (example)

- What's improved?
 - ✓ Change from bar to line
 - ✓ Reduced bars from 25 to 4 lines
 - ✓ Same x –axis instead of legends
 - ✓ “All” category removed
 - ✓ Decimal points rounded to whole digits
 - ✓ Italics in font changed in subtitle
 - ✓ Highlighting of “Bachelors degree or more” category

177

BITS Pilani

177

Affordances

Visual Hierarchy of information

- Preattentive attributes can be used for focusing
- Pull some item foreground, push some items to background indicating the order in which to process the information

178

BITS Pilani

178

Affordances

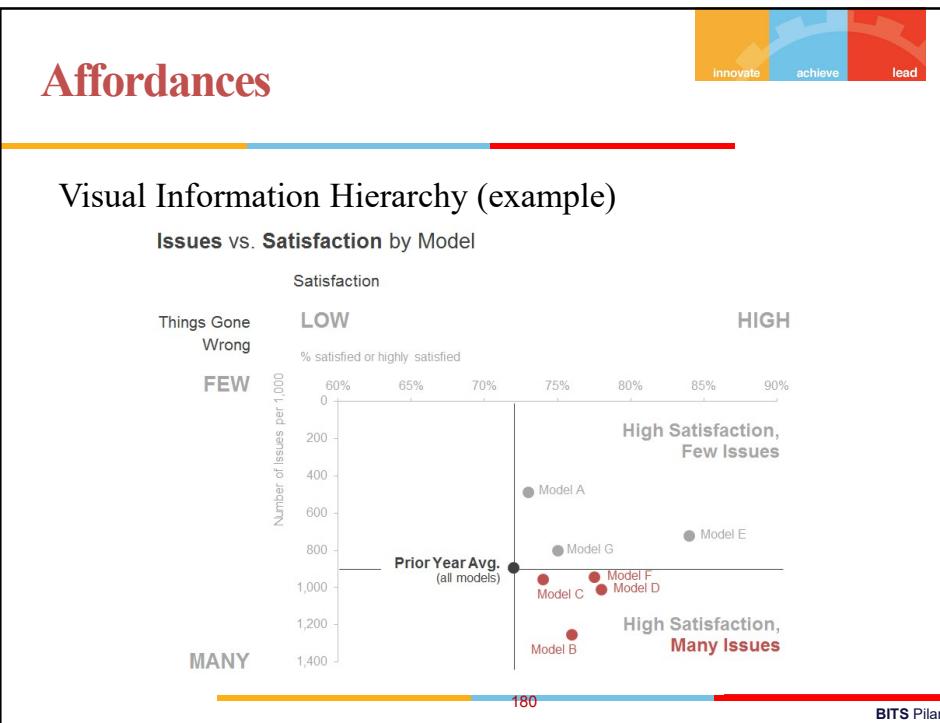
Visual Information Hierarchy (example)

Car manufacturer

- ❑ Dimensions used for determining success of make and model
 - ✓ Customer satisfaction
 - ✓ Frequency of car issues

179

BITs Pilani



Affordances



Visual Information Hierarchy (example)

Hierarchy components

- ✓ Graph title: Bolding of words used
- ✓ y-axis label: Issues scale visible
- ✓ x-axis label: Satisfaction scale visible
- ✓ Dark point in visual: Last years average
- ✓ Red points in last quadrant: highlighting used

181

BITS Pilani

181

Accessibility



Accessibility

Designs should be usable by people of diverse abilities

- ✓ Example, mobile phone screens

Work to make your data visualizations similarly straightforward and easy to use

182

BITS Pilani

182

Accessibility



Accessibility strategies

- Don't overcomplicate
- Use text to label, introduce, explain, reinforce, highlight, recommend, and tell a story

183

BITS Pilani

183

Accessibility



Don't overcomplicate

"If it's hard to read, it's hard to do it."

"The more complicated the visual looks, the more time audience perceives it will take to understand and less likely they are to spend time to understand it."

-- Cole Knaflic

184

BITS Pilani

184



Accessibility

Don't overcomplicate

Guidelines

- ✓ Make it legible
 - ❖ use consistent, easy to read font
- ✓ Keep it clean
 - ❖ use visual affordance, remove clutter
- ✓ Use straightforward language
 - ❖ over complex one
- ✓ Remove unnecessary complexity

185

BITS Pilani

185



Accessibility

Text is your Friend

Accessible visual

Use text to label, introduce, explain, reinforce, highlight, recommend, and tell a story

Must have text for chart titles, axis titles

Use text for conclusion or suggestion or recommended action

186

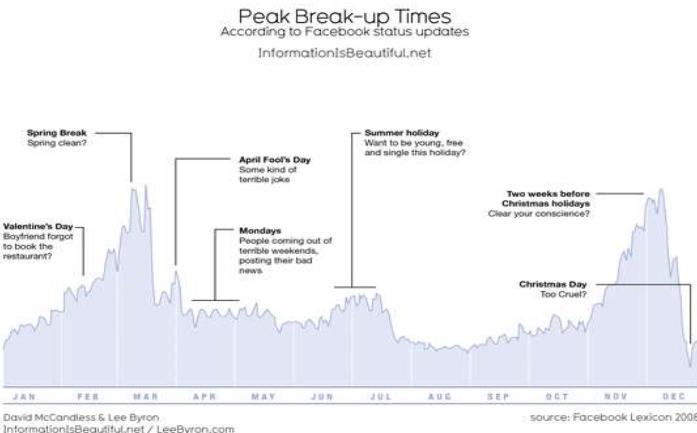
BITS Pilani

186

Accessibility



Text is your friend (example: annotations)



187

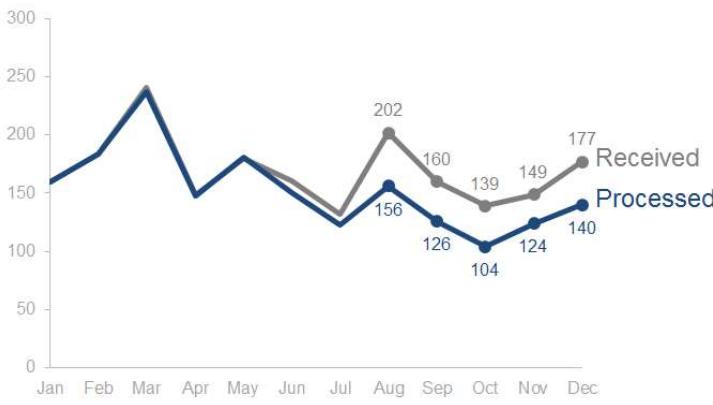
BITS Pilani

187

Accessibility



Text is your friend (example)



188

BITS Pilani

188

Accessibility

The chart displays two data series: 'Received' (blue line with circles) and 'Processed' (grey line with circles). The y-axis represents the 'Number of tickets' from 0 to 300. The x-axis shows the months from Jan to Dec 2014. A vertical grey line is drawn at May.

Month	Received	Processed
Jan	160	160
Feb	180	180
Mar	230	230
Apr	150	150
May	180	180
Jun	160	160
Jul	120	120
Aug	202	156
Sep	160	126
Oct	139	104
Nov	149	124
Dec	177	140

Data source: XYZ Dashboard, as of 12/31/2014

189

Accessibility

Text is your friend (example)

Please approve the hire of 2 FTE
to backfill those who quit in the past year

The chart displays two data series: 'Received' (blue line with circles) and 'Processed' (grey line with circles). The y-axis represents the 'Number of tickets' from 0 to 300. The x-axis shows the months from Jan to Dec 2014. A vertical grey line is drawn at May.

Month	Received	Processed
Jan	160	160
Feb	180	180
Mar	230	230
Apr	150	150
May	180	180
Jun	160	160
Jul	120	120
Aug	202	156
Sep	160	126
Oct	139	104
Nov	149	124
Dec	177	140

2 employees quit in May. We nearly kept up with incoming volume in the following two months, but fell behind with the increase in Aug and haven't been able to

Data source: XYZ Dashboard, as of 12/31/2014 | A detailed analysis on tickets processed per person and time to resolve issues was undertaken to inform this request and can be provided if needed.

190

Aesthetics



“People perceive more aesthetic designs as easier to use than less aesthetics designs – whether they actually are or not.”

- Cole, Storytelling with data

.... Increases our chance of success for getting our message across

191

BITS Pilani

Aesthetics



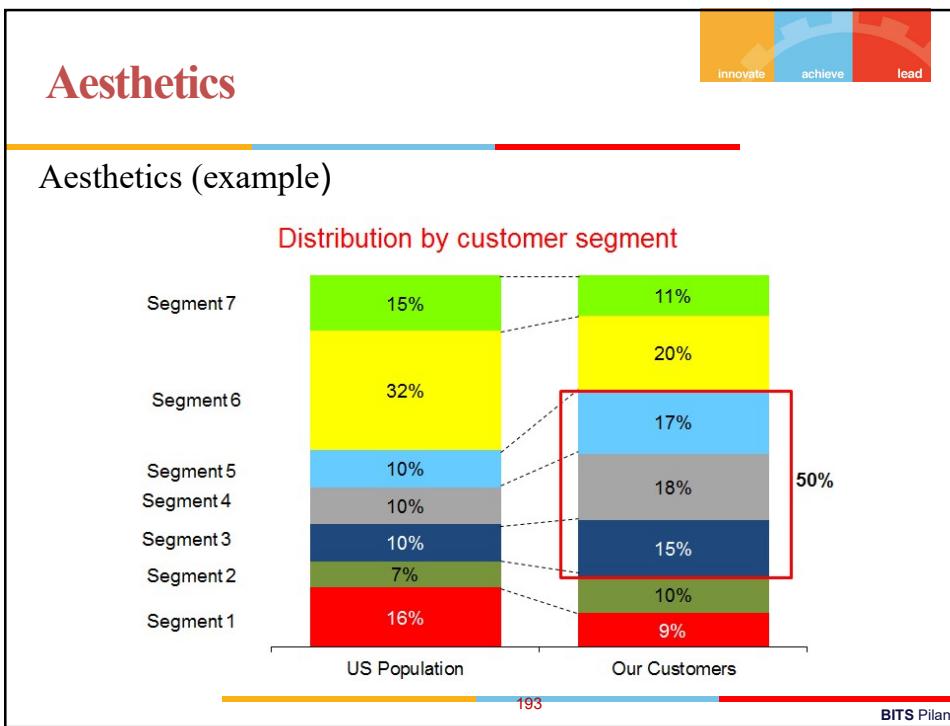
❑ Considerations

- Be smart with color
 - ✓ Use it sparingly and strategically
- Pay attention to alignment
 - ✓ Create sense of unity and cohesion
- Leverage white space
 - ✓ Preserve margins
 - ✓ Don't stretch graphics or add things

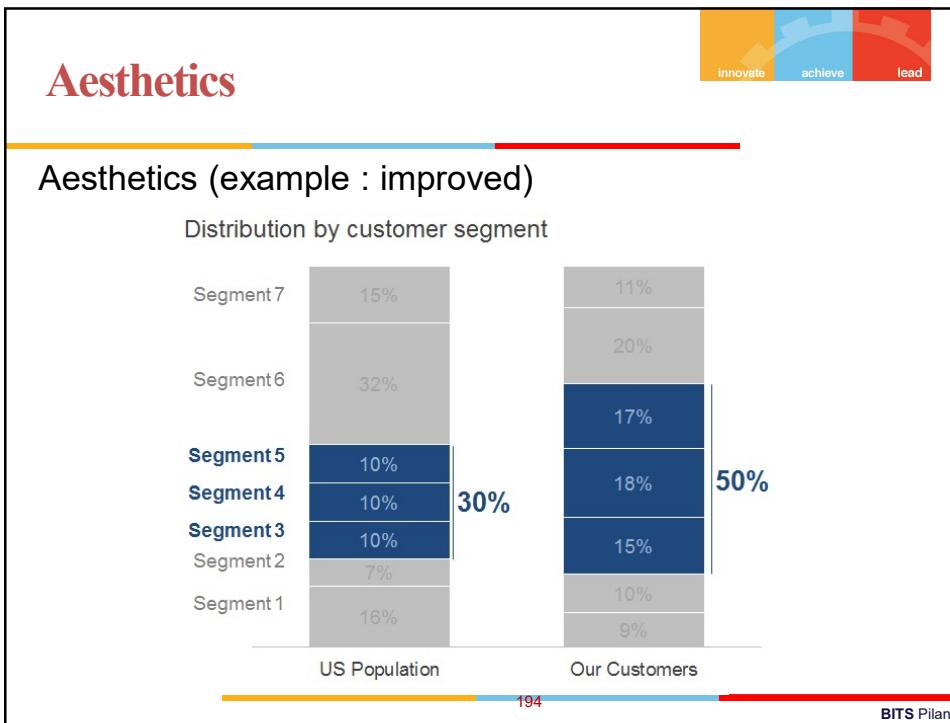
192

BITS Pilani

192



193

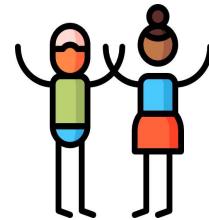


194



Acceptance

“For a design to be effective, it must be accepted by its intended audience.”



195

BITS Pilani

195



Acceptance



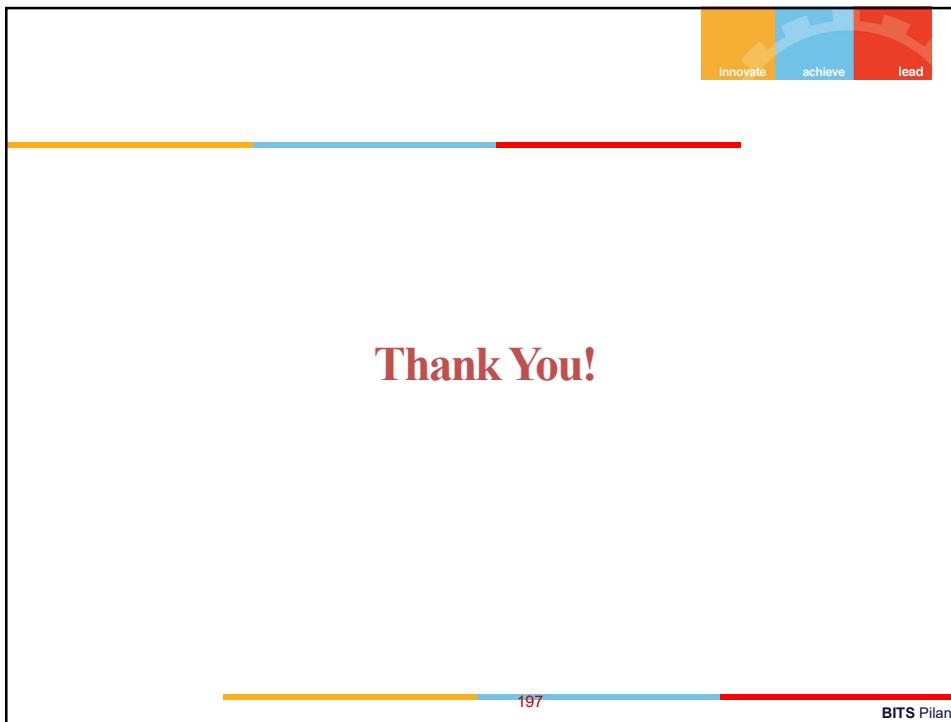
Strategies for gaining acceptance

- Articulate the benefits of the new or different approach
- Show the side-by-side
- Provide multiple options and seek input
- Get a vocal member of your audience on board

196

BITS Pilani

196



197

CONTACT SESSION 3 -PLAN		
Contact Sessions(#)	List of Topic Title	Text Book
CS3	<ul style="list-style-type: none"> • Decluttering • Pre-attentive attributes in text and graphs • Data Design concepts 	T1 Ch 3 T1 Ch 4 T1 Ch 5

198

BITS Pilani

198



Cognitive Load

- It is the amount of mental effort that we use to get the information that we need.
- signal-to-noise ratio

199

BITS Pilani

199



Cognitive Load

- There are three types of cognitive load
 - Intrinsic,
 - Extraneous, and
 - Germane.
- Different tasks need different amount of thoughts and attention

200

BITS Pilani

200



Cognitive Load

- **Intrinsic Cognitive Load**
- The amount of memory that we need to understand something.
- **Extraneous cognitive load**
- Refers to the amount of **extra** brain power required to analyze and process information.
- **Germane cognitive load**
- A way for the brain to look for patterns to develop context.

201

BITS Pilani

201



Question

- Which type of the cognitive loads just discussed pertains to the baseline level amount of brain power needed to understand an idea?
- Germane Cognitive Load
- Intrinsic Cognitive Load
- Extraneous Cognitive Load

202

BITS Pilani

202

Question



- Which type of the cognitive loads just discussed pertains to the baseline level amount of brain power needed to understand an idea?
- Germane Cognitive Load
- **Intrinsic Cognitive Load**
- Extraneous Cognitive Load

203

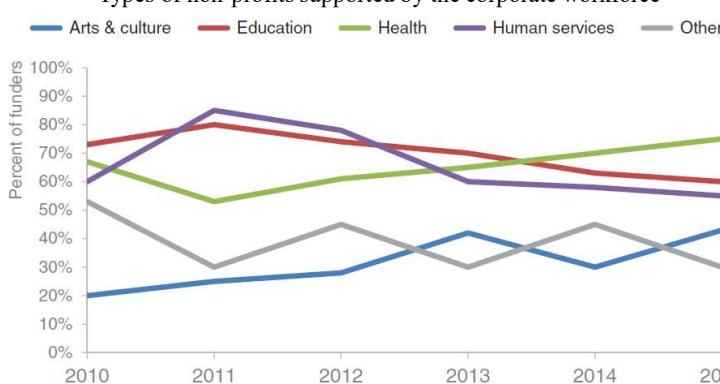
BITS Pilani

203

Case Study -Clutter



Types of non-profits supported by the corporate workforce



Data is self-reported by funders; percents sum to greater than 100 because respondents can make multiple selections.

- **To Do:** You are a Data Science Consultant. Convey to your CSR leadership team where they should plan to focus the organization's CSR activity in the future

204

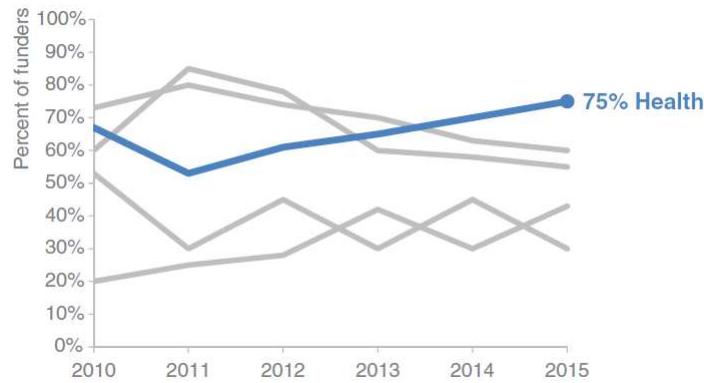
BITS Pilani

204

Case Study – probable solutions



Health sector shows the highest potential to receive grants in the future



205

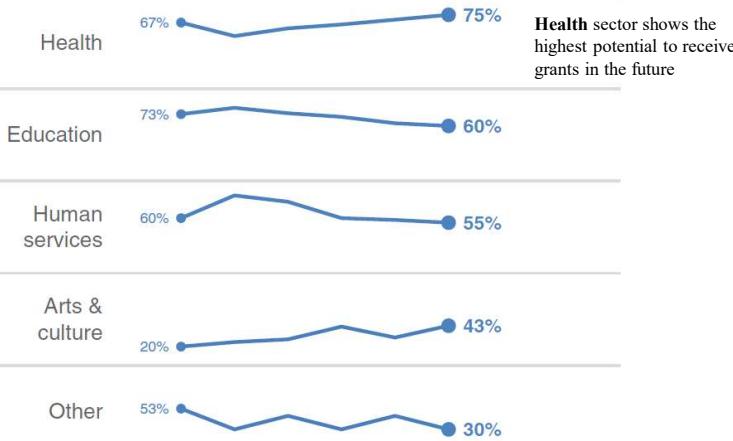
BITS Pilani

205

Case Study – probable solutions



2010 2011 2012 2013 2014 2015 | % of funders

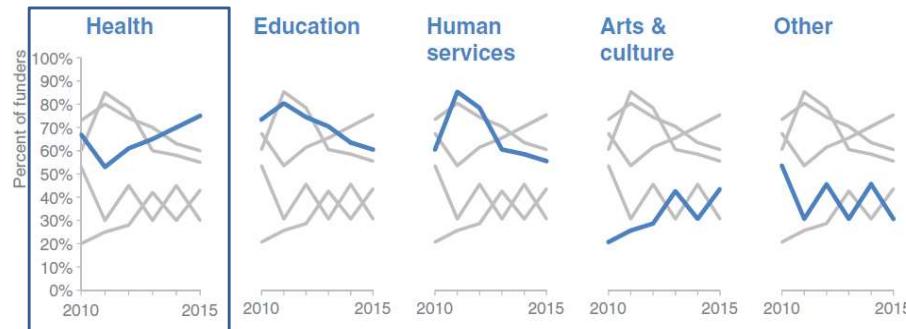


206

BITS Pilani

206

Case Study – probable solutions



Data is self-reported by funders; percents sum to greater than 100 because respondents can make multiple selections.

207

BITS Pilani

Case study 2-Clutter



208

BITS Pilani

208



Human Memory

1. How human sees?
2. How you can use that to your advantage when creating visuals?

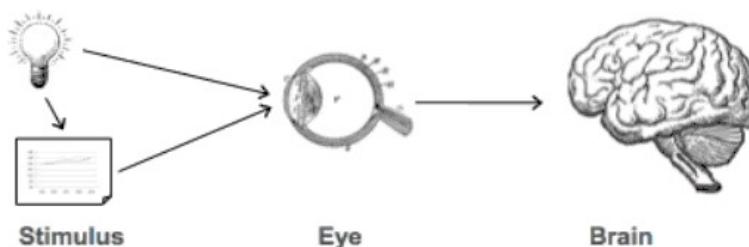
209

BITS Pilani

209



How Human sees?



210

BITS Pilani

210

See with Brain

Types of Human Memory

- Iconic
- Short term
- Long term

211

BITS Pilani

211

See with Brain

Stimuli/input → **Rehearsal**

```

graph LR
    Stimuli[Stimuli/input] --> STM[Short-term memory  
15 - 30 secs]
    STM --> LTM[Long-term memory  
1sec - Lifetime]
    STM --> SM[Sensory memory  
1-3 secs]
    LTM --> Forgetting[Forgetting  
Caused by biological factors, or antecedent processes]
    SM --> Forgetting
    
```

Forgetting
Caused by biological factors, or antecedent processes

Adapted from: <https://www.interaction-design.org/literature/article/the-properties-of-human-memory-and-their-importance-for-information-visualization>

212

BITS Pilani

212

Types of Human Memory



Iconic

- Stored for tiny time periods
- Originate from our sensory organs such as our eyes
- Typically retained for less than 500 milliseconds
- Raw information that the brain receives from the eye
- Store and process sensory memories automatically – that is without any conscious effort to do so

213

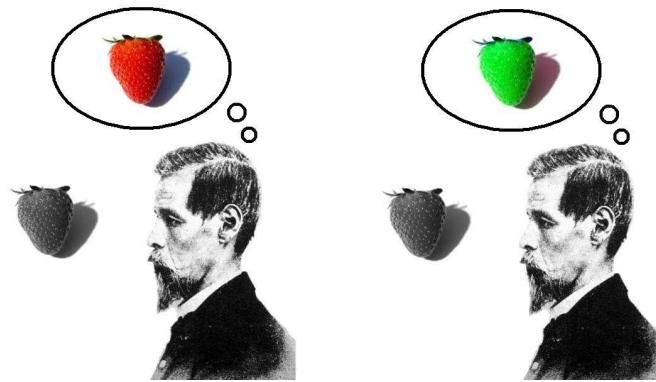
BITS Pilani

213

Types of Human Memory



Iconic Memory (example)



214

BITS Pilani

214



Types of Human Memory

Short-term memory

- Limited capacity
- Used to process sensory memories which are of interest to us – for whatever reason
- The sensory memory is transferred to the short-term memory where it may be processed for up to a minute

215

BITS Pilani

215



Types of Human Memory

Long-term memory

- Built up over a lifetime
- Important for pattern recognition and cognitive processing
- Visuals persisted here can have longer lasting impact

216

BITS Pilani

216



How to use human sight ?

Preattentive Attributes

- Size
- Color
- Position

Help to direct audience's attention to where you want them to focus in the visuals

217

BITS Pilani

Preattentive Signals

Example

756395068473
658663037576
860372658602
846589107830

218

BITS Pilani

218



Preattentive Signals

Example

756395068473
658663037576
860372658602
846589107830

219

BITS Pilani



Preattentive Signals

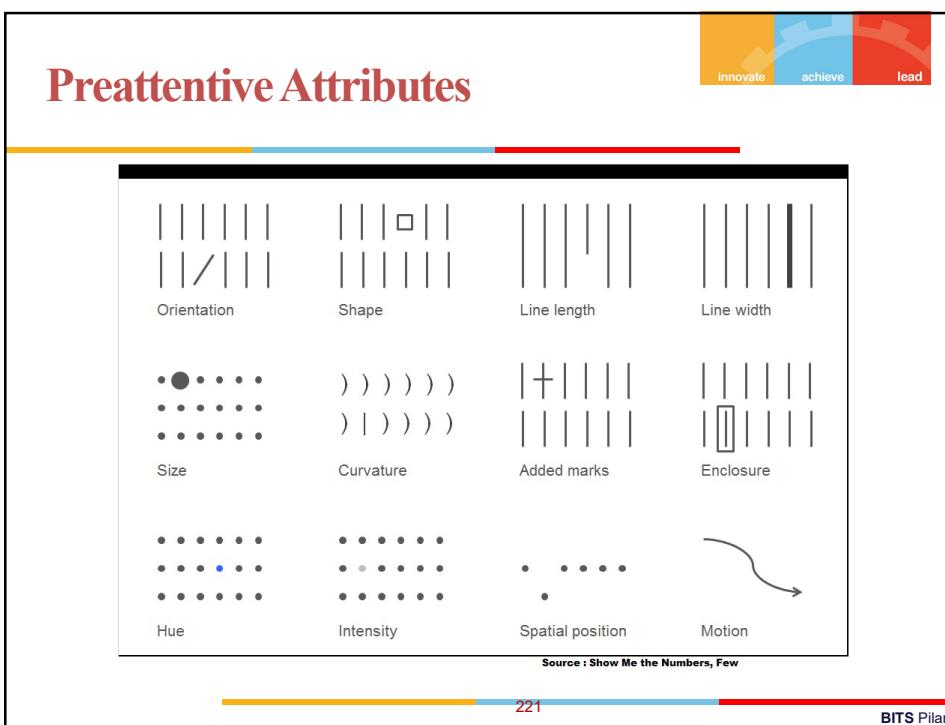
• ● • • •
• • • • •
• • • • •

“Enable our audience to see what we want them to see before they even know they’re seeing it!”

- Cole Knaflic

220

BITS Pilani



221

Preattentive Attributes

innovate achieve lead

Importance

- Drawing audience's attention quickly to where you want them to look
- Creating a visual hierarchy of information

222

BITS Pilani

222



Preattentive Attributes In text

Example

No preattentive attributes

What are we doing well? Great Products. These products are clearly the best in their class. Replacement parts are shipped when needed. You sent me gaskets without me having to ask. Problems are resolved promptly. Bev in the billing office was quick to resolve a billing issue I had. General customer service exceeds expectations. The account manager even called to check in after normal business hours.

You have a great company – keep up the good work!

223

BITS Pilani

223



Preattentive Attributes In text

Example (use of color)



Color

What are we doing well? Great Products. **These products are clearly the best in their class.** Replacement parts are shipped when needed. You sent me gaskets without me having to ask. Problems are resolved promptly. Bev in the billing office was quick to resolve a billing issue I had. General customer service exceeds expectations. The account manager even called to check in after normal business hours.

You have a great company – keep up the good work!

224

BITS Pilani

224

Preattentive Attributes In text

T T

Example (use of Size)

Size

What are we doing well? Great Products. These products are the best in their class. Replacement parts are shipped when needed. You sent gaskets without me having to ask. Problems are resolved promptly. Bev in the billing office was quick to resolve a billing issue I had. General customer service exceeds expectations. The account manager even called to check in after normal business hours. You have a great company – keep up the good work!

225

BITS Pilani

Preattentive Attributes In text

 **Example (use of enclosure)**

Outline (enclosure)

What are we doing well? Great Products. These products are clearly the best in their class. Replacement parts are shipped when needed. You sent me gaskets without me having to ask. Problems are resolved promptly. Bev in the billing office was quick to resolve a billing issue I had. General customer service exceeds expectations. The account manager even called to check in after normal business hours.

You have a great company – keep up the good work!

226

BITS Pilani



Preattentive Attributes In text

Example (information hierarchy)

What are we doing well?

Themes & example comments

- **Great products:** "These products are clearly the best in class."
- **Replacement parts are shipped when needed:**
"You sent me gaskets without me having to ask, and I really needed them, too!"
- **Problems are resolved promptly:** "Bev in the billing office was quick to resolve a billing issue I had."
- **General customer service exceeds expectations:**
"The account manager even called after normal business

227

BITS Pilani

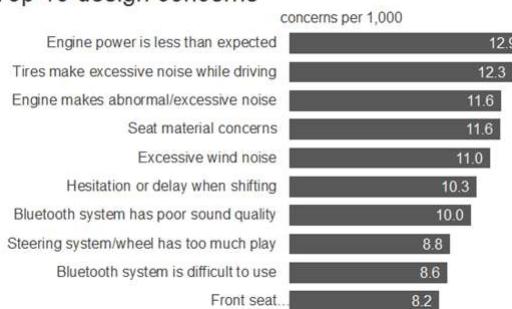
227



Preattentive Attributes In graphs

Example

Top 10 design concerns



228

BITS Pilani

228

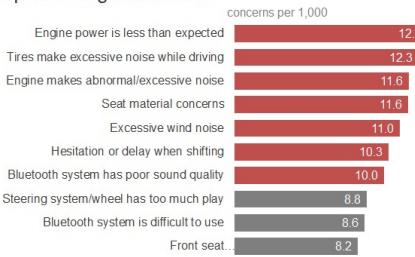
Preattentive Attributes In graphs



Example (use of color and text attributes)

7 of the top 10 design concerns have 10 or more concerns per 1,000.
Discussion: is this an acceptable default rate?

Top 10 design concerns



229

BITS Pilani

229

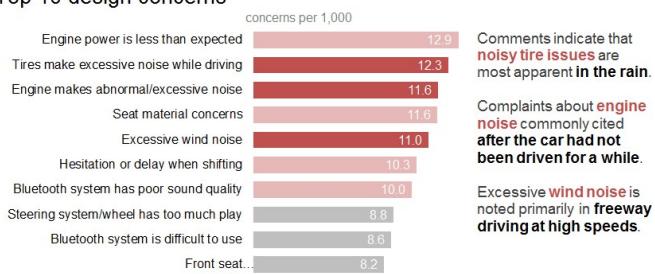
Preattentive Attributes In graphs



Example (Modified focus and text)

Of the top design concerns, three are noise-related.

Top 10 design concerns



230

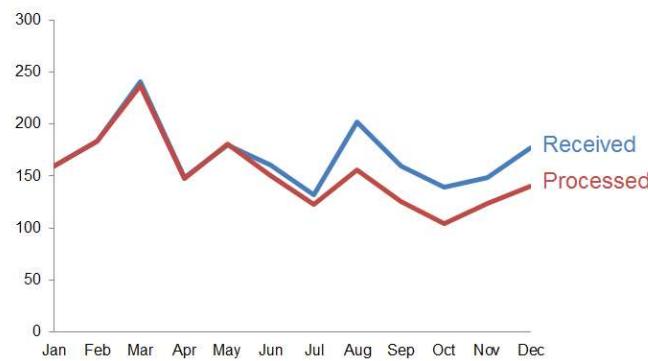
BITS Pilani

230

Preattentive Attributes In graphs



Another example



231

BITS Pilani

Preattentive Attributes In graphs



Another example (everything in background)



232

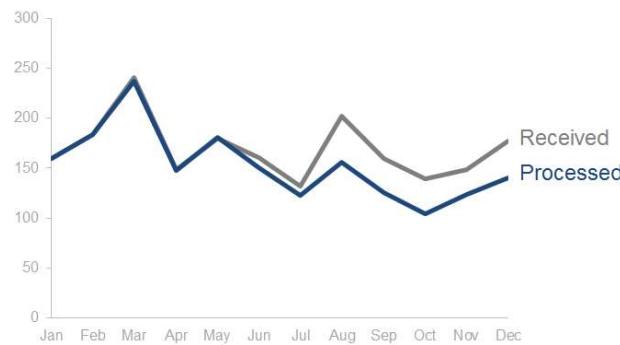
BITS Pilani

232

Preattentive Attributes In graphs



Another example (use color)



233

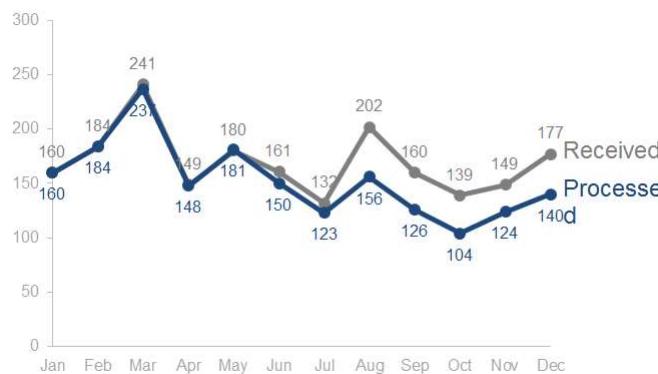
BITS Pilani

233

Preattentive Attributes In graphs



Another example (data labels and clutter)



234

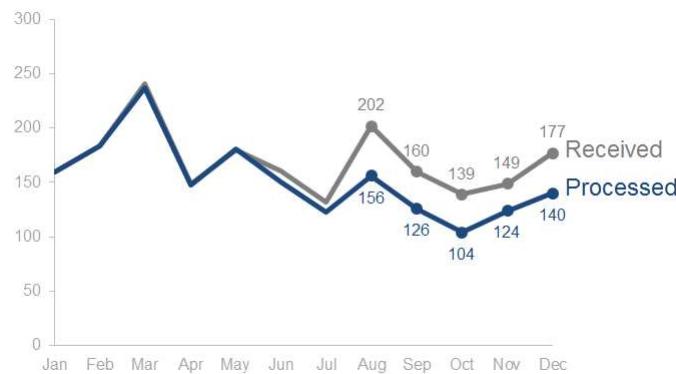
BITS Pilani

234

Preattentive Attributes In graphs



Another example (use labels sparingly)



BITS Pilani

235

Size



Preattentive Attribute – Size

- Size matters!
- Relative Size --→ Relative Importance!
- Similar things -→ Similar Size!

BITS Pilani

236

Color



Preattentive Attribute – Color

- Use sparingly
- Use Consistently
- Design with colorblind in mind
- Be thoughtful of the tone that color conveys
- Use Brand colors

237

BITS Pilani

237

Use color sparingly



Country Level Sales Rank Top 5 Drugs

Rainbow distribution in color indicates sales rank in given country from #1 (red) to #10 or higher (dark purple)

Country	A	B	C	D	E
AUS	1	2	3	6	7
BRA	1	3	4	5	6
CAN	2	3	6	7	8
CHI	1	2	3	4	7
FRA	3	2	4	8	9
GER	3	1	6	5	4
IND	4	1	3	7	5
ITA	2	4	5	9	8
MEX	1	5	4	6	3
RUS	4	3	7	9	10
SPA	2	3	4	5	11
TUR	7	2	3	4	5
UK	1	2	3	6	7
US	1	2	4	3	5

Top 5 drugs: country-level sales rank

COUNTRY DRUG	RANK				
	A	B	C	D	E
Australia	1	2	3	6	7
Brazil	1	3	4	5	6
Canada	2	3	6	12	8
China	1	2	8	4	7
France	3	2	4	8	10
Germany	5	1	6	5	4
India	4	1	8	10	5
Italy	2	1	10	9	8
Mexico	1	5	4	6	11
Russia	3	7	9	8	12
Spain	2	3	4	5	11
Turkey	7	2	9	4	8
United Kingdom	1	2	3	6	7
United States	1	2	4	3	5

238

BITS Pilani

238

Brand color

Leverage brand color Draw attention with black Use complementary color

Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7
7	7	7	4	4	4	7
Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7
Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7

ClientLogo *ClientLogo* *ClientLogo*

239

BITS Pilani

Position

Pre-attentive Attribute – Position

- Use top left corner wisely for important stuff
- Be mindful of how you position elements on page
- Aim to do so in way that will feel natural for audience to consume information

240

BITS Pilani

CONTACT SESSION 3 -PLAN		
Contact Sessions(#)	List of Topic Title	Text Book
CS3	<ul style="list-style-type: none"> • Decluttering • Pre-attentive attributes in text and graphs • Data Design concepts • Visualisation Types 	T1 Ch 3 T1 Ch 4 T1 Ch 5 T2 Ch 5

— 241 —

BITS Pilani

241

Data Visualisation Methods		
Textuals		
Text Based Visualizations		
<input type="checkbox"/> Simple Text	<input type="checkbox"/> Tables	<input type="checkbox"/> Heatmap

— 242 —

BITS Pilani

242




Graphs

 Graphs

- Interacts with visual system
- Well designed graph get information across more quickly than well designed table

243

BITSPilani




Graphicals

 Graphs

- Points
- Lines
- Bars
- Area

244

BITSPilani

244

Data Visualisation Methods- Comparing Categories



- Dot Plot
- Bar Chart
- Floating Bar
- Histogram
- Slope Graph
- Radial Chart
- Glyph Chart

245

BITS Pilani

Dot Plots



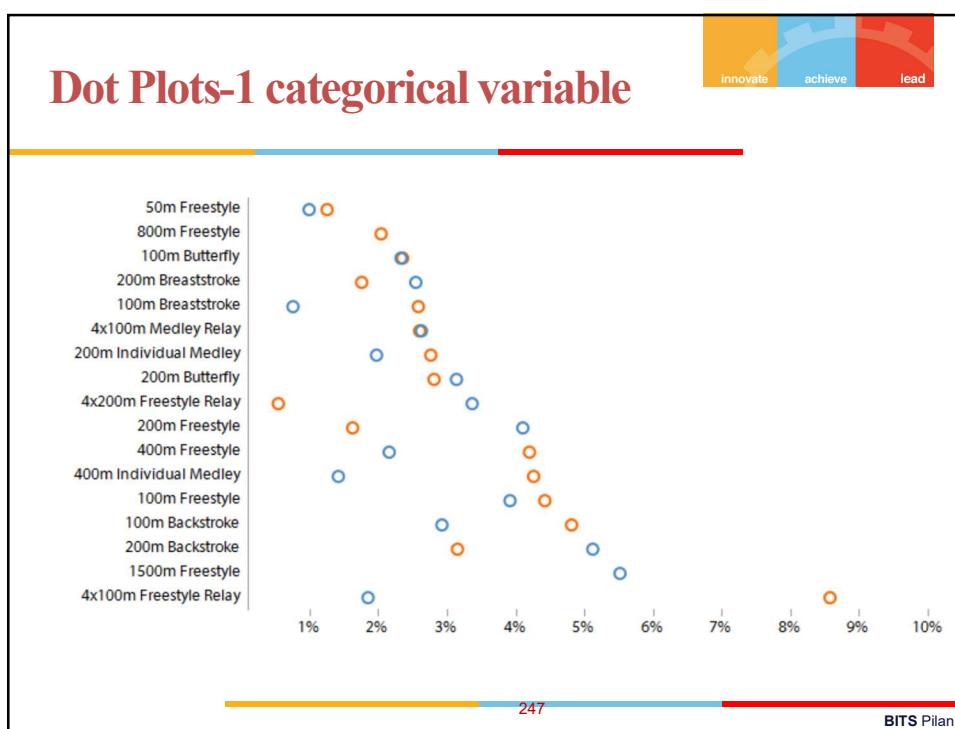
Data variables: 2 x categorical, 1 x quantitative.

Visual variables: Position, color-hue, symbol.

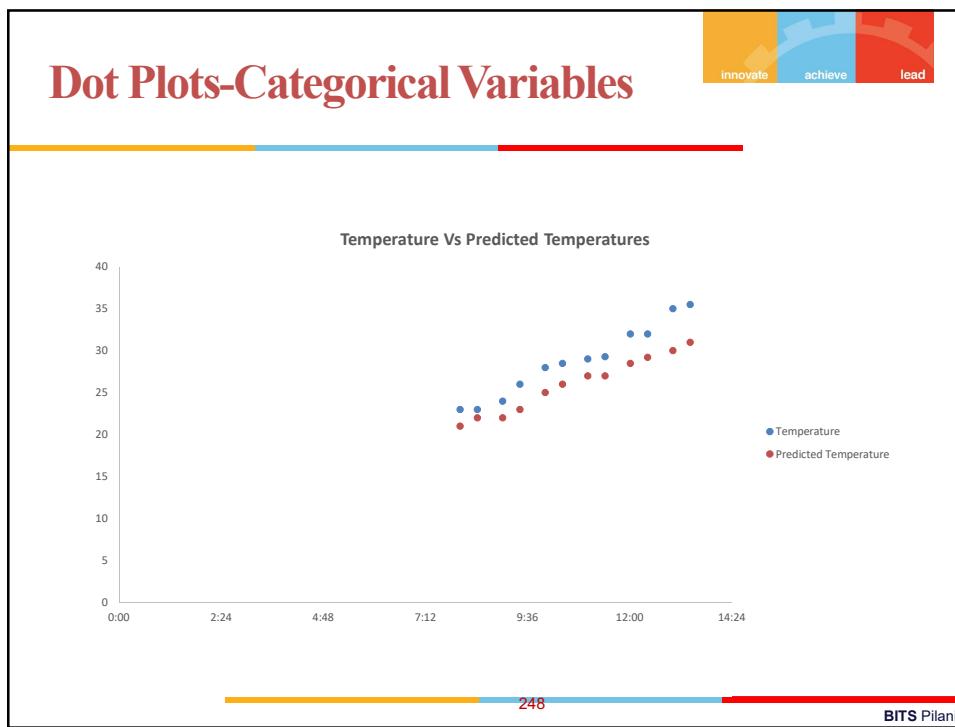
246

BITS Pilani

246



247



248

Bar Charts



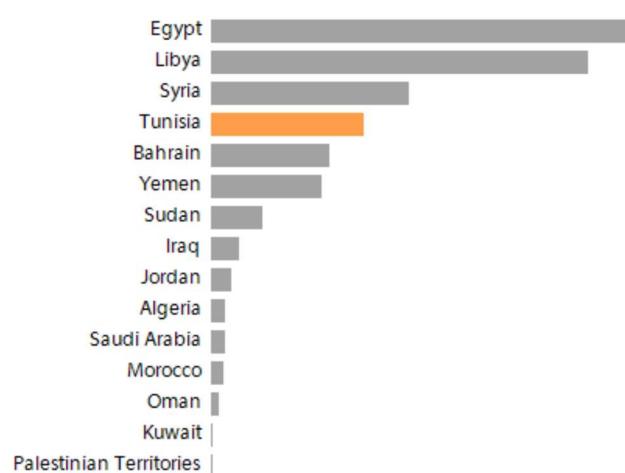
Data variables: 1 x categorical, 1 x quantitative-ratio.

Visual variables: Length/height, color-hue.

249

BITS Pilani

Bar Charts



5% 10% 15% 20% 25%

250

BITS Pilani

250

Floating Bars/Gantt Charts



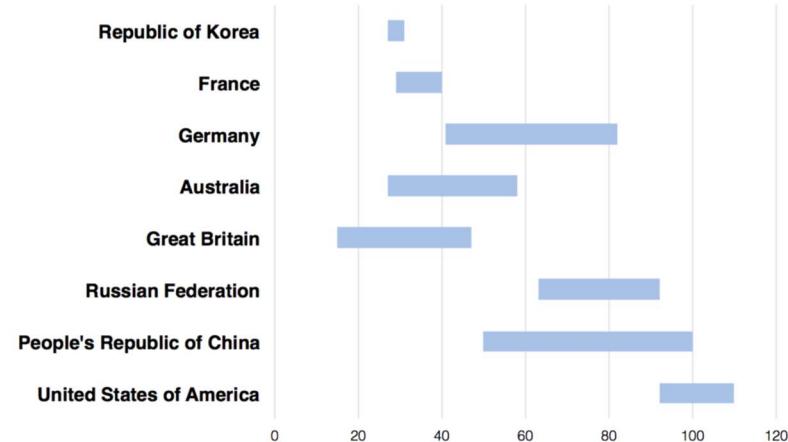
Data variables: 1 x categorical-nominal, 2 x quantitative.

Visual variables: Position, length.

251

251

Data Visualisation Methods - Floating Bars



252

252

Histogram



Data variables: 1 x quantitative-interval, 1 x quantitative-ratio.

Visual variables: Height, width.

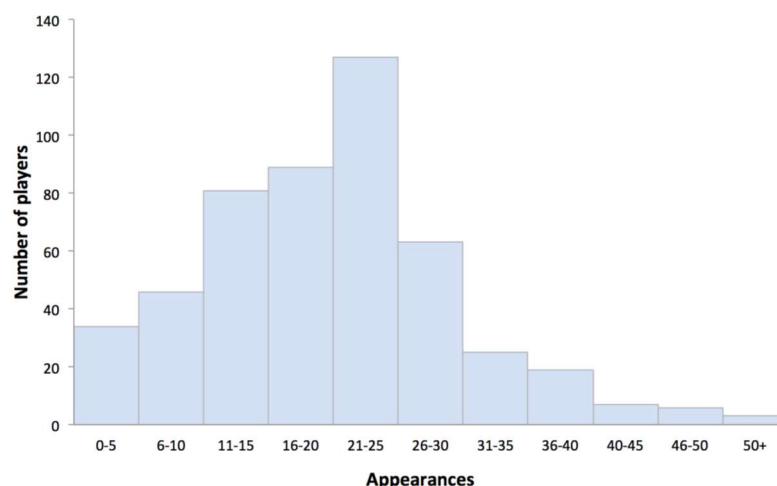
- show distribution through the frequency of quantitative values (y axis) against defined intervals of quantitative values(x axis).

253

BITS Pilani

253

Data Visualisation Methods Histogram



254

BITS Pilani

254

Slope Graph



Data variables: 1 x categorical, 2 x quantitative.

Visual variables: Position, connection, color-hue.

255

BITS Pilani

Slope Graph



2010/2011

2011/2012

Manchester United, 80

Chelsea, 71
Manchester City, 71
Arsenal, 68

Tottenham, 62

Liverpool, 58

Everton, 54

Newcastle, 46

Manchester City, 89
Manchester United, 89

Arsenal, 70

Tottenham, 69

Newcastle, 65

Chelsea, 64

Everton, 56

Liverpool, 52

256

BITS Pilani

256

Slope Graph



BITS Pilani

257

Glyph Chart



Data variables: Multiple x categorical, multiple x quantitative.

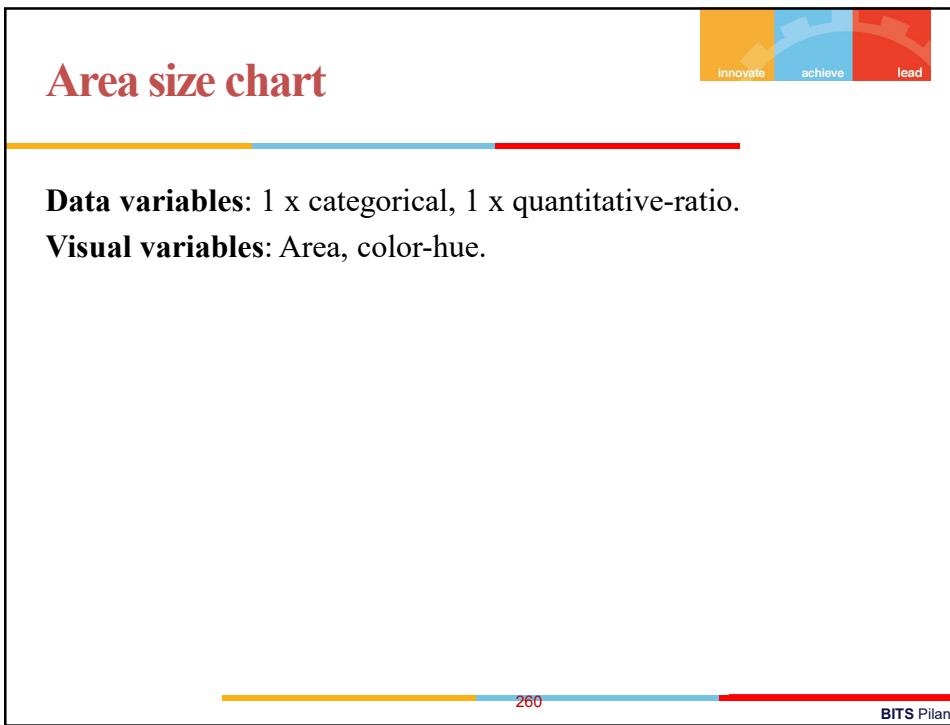
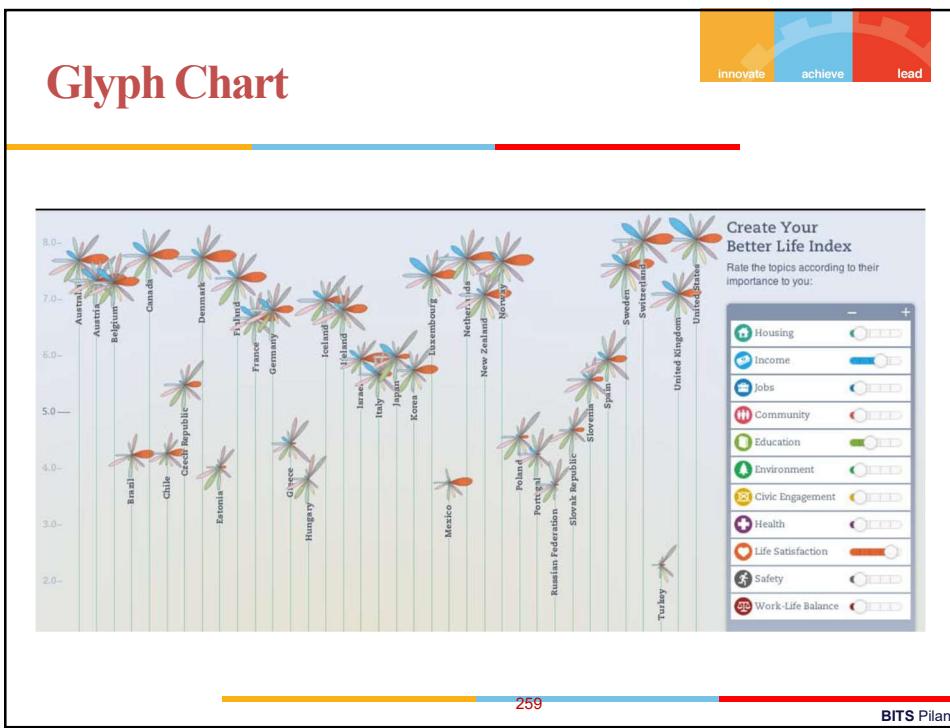
Visual variables: Shape, size, position, color-hue.

Description: A glyph chart is based on a shape (in the following example, a flower) being the main artifact of representation.

258

BITS Pilani

258



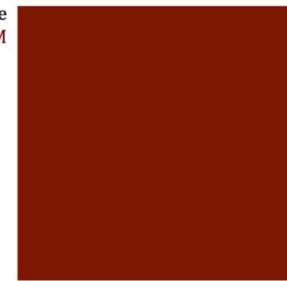
Area size chart



A Tale of Two Leagues: Comparing Transfer Spend (Summer 2012)

English Premier League
£554M

Scottish Premier League
£22M



261

BITS Pilani

261

Trellis chart



Data variables: Multiple x categorical, multiple x quantitative.

Visual variables: Position, any visual variable.

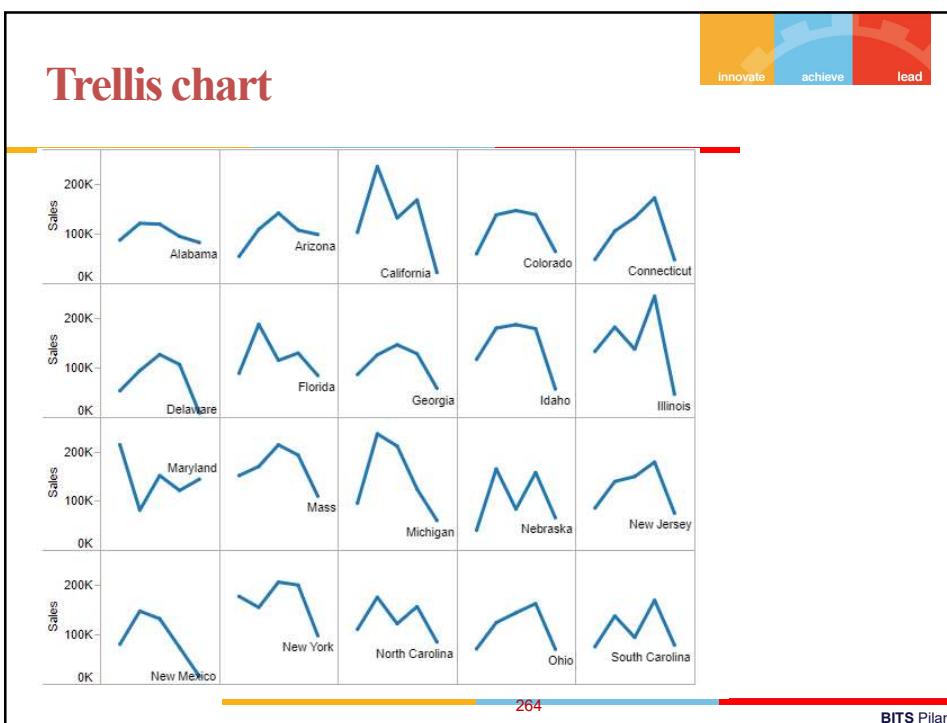
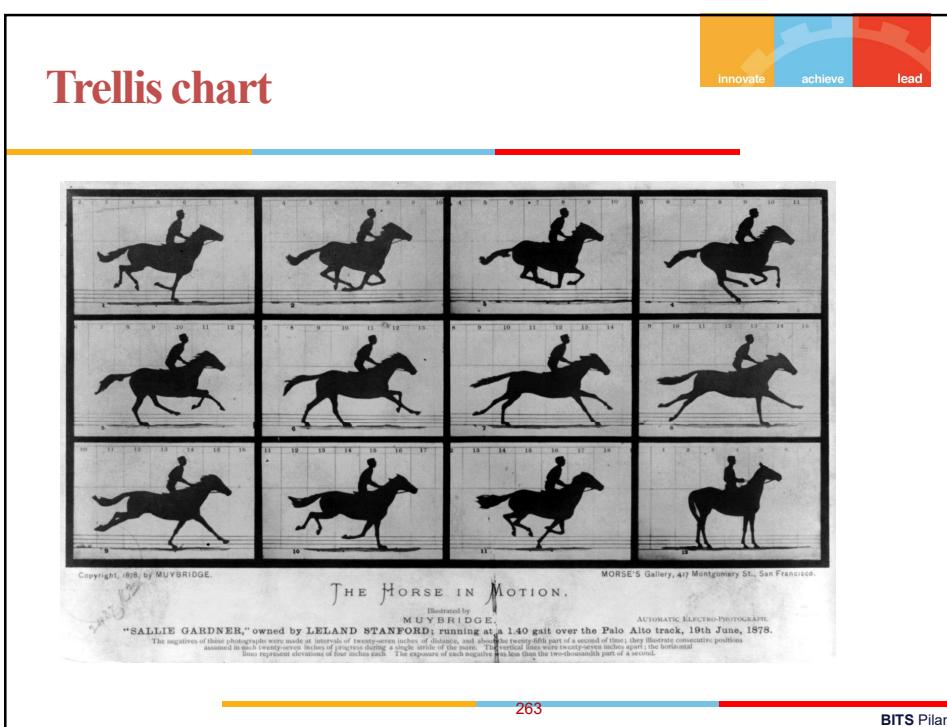
Description:

Small multiples are not really a separate chart type but an arrangement approach that facilitates efficient and effective comparisons to be made across a multipanel display of small chart elements.

262

BITS Pilani

262



264

Word cloud

Data variables: 1 x categorical, 1 x quantitative-ratio.

Visual variables: Size.

265

BITS Pilani

265

Word cloud

design
data
visualization
image

266

BITS Pilani

266

Assessing hierarchies and part-to-whole Relationships- Pie chart



Data variables: 1 x categorical, 1 x quantitative-ratio.

Visual variables: Angle, area, color-hue.



BITS Pilani

267

Stacked bar chart

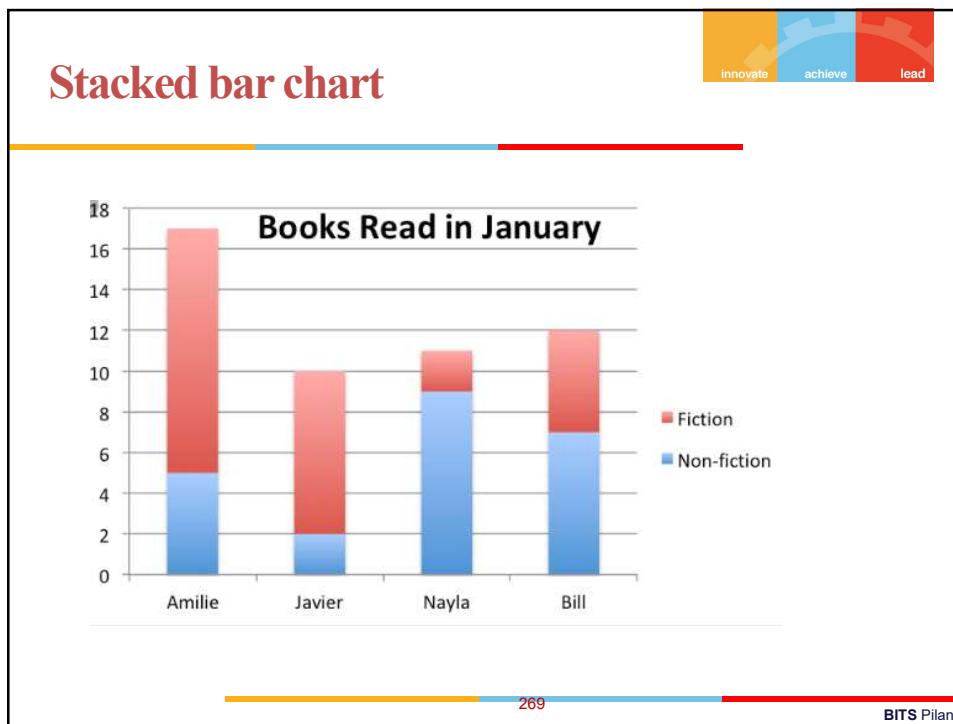


Data variables: 2 x categorical, 1 x quantitative-ratio.

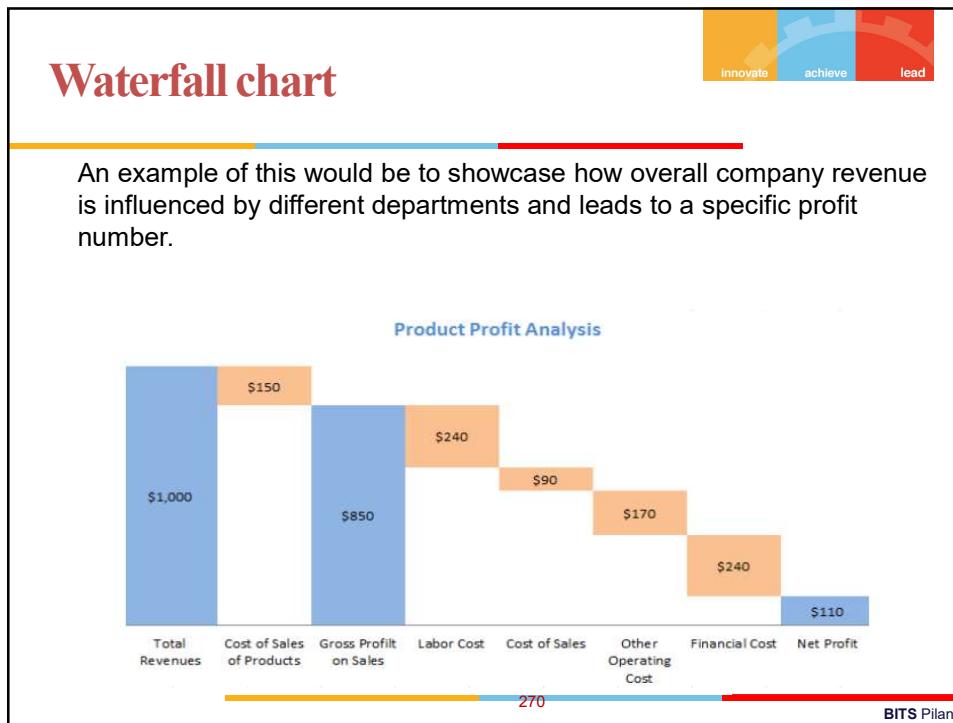
Visual variables: Length, color-hue, position, color-saturation/lightness.

BITS Pilani

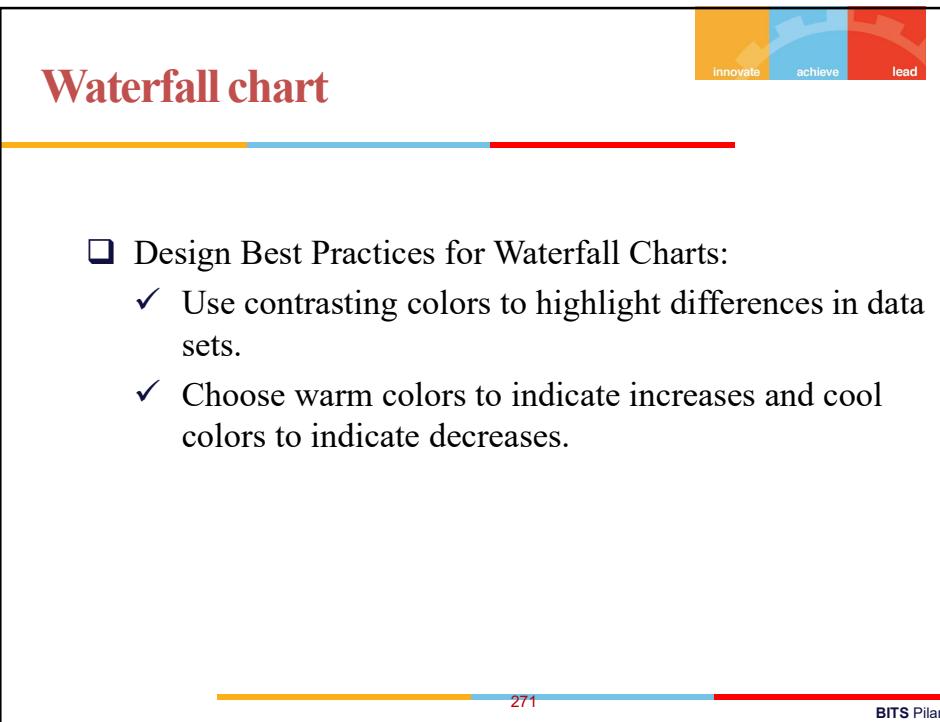
268



269



270



Waterfall chart

inovate achieve lead

- ❑ Design Best Practices for Waterfall Charts:
- ✓ Use contrasting colors to highlight differences in data sets.
- ✓ Choose warm colors to indicate increases and cool colors to indicate decreases.

271

BITs Pilani



Tree hierarchy

inovate achieve lead

LITERARY ORGANISM
A visualization of Part One of *On the Road*, by Jack Kerouac

BASIC STRUCTURE
Each literary component can be divided into even smaller parts, the smallest in this diagram being words. The diagram is organized into four levels: chapters, paragraphs, sentences, and words.

NOTATION
Each quotation can be positioned at the first chapter, 3.5 paragraph.

COLORS

- Dean Moriarty (Protagonist)
- Hop & Jean鼠
- Social Events & Interaction
- Travel
- Merchants of Regional Life
- Parties, Dressing & Drags
- Work & Survival
- Red Parallel (Satire)
- Women, Sex, & Relationships
- Digital Activities & Police Encounters
- Character Backdrops

WORD COUNT CHART
All sentences are in proportion to the longest sentence in Part One, which is 131 words long.

80 words 90 words 100 words 110 words 120 words 130 words 140 words 150 words 160 words

<http://www.stefanieposavec.com/writing-without-words>

272

BITs Pilani

Showing changes over time-Line Chart



Data variables: 1 x quantitative-interval, 1 x quantitative-ratio, 1 x categorical.

Visual variables: Position, slope, color-hue.

273

BITS Pilani

Showing changes over time-Line Chart



The Contrasting Fortunes of German and Chinese Olympic Success

Percentage of total medals won across past five Olympics (eight countries selected based on ranking at 2008)

Germany
10.1%

China
6.6%

10.5%

4.3%

1992 1996 2000 2004 2008

274

BITS Pilani

274

Area Charts



Data variables: 1 x quantitative-interval, 1 x categorical, 1 x quantitative-ratio.

Visual variables: Height, slope, area, color-hue.

275

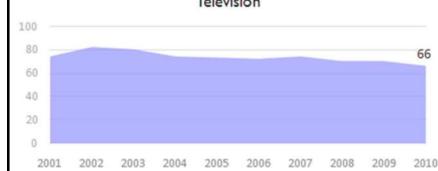
BITS Pilani

275

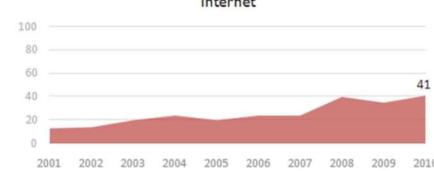
Area chart



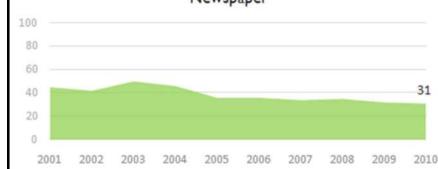
Television



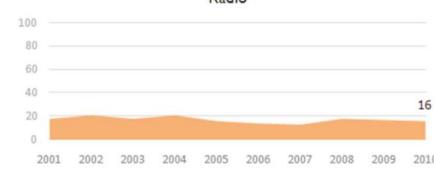
Internet



Newspaper



Radio



276

BITS Pilani

276

Area chart



Design Best Practices for Area Charts:

- Use transparent colors so information isn't obscured in the background.
- Don't display more than four categories to avoid clutter.
- Organize highly variable data at the top of the chart to make it easy to read.

277

BITS Pilani

Stacked area chart



Data variables: 1 x quantitative-interval, 1 x categorical, 1 x quantitative-ratio.

Visual variables: Height, area, color-hue.

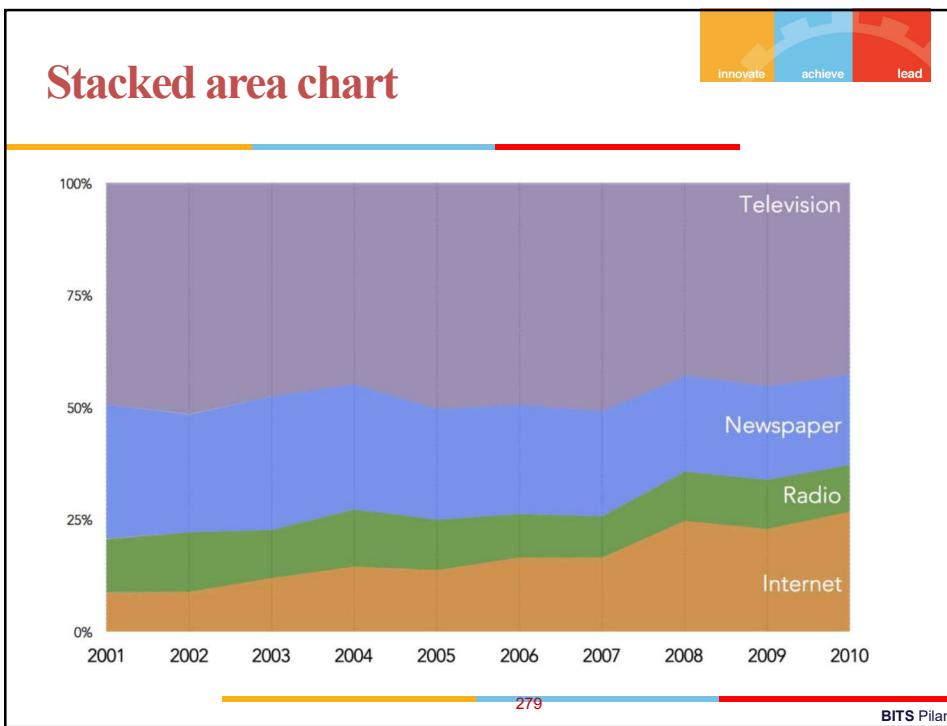
Stacked Area Graphs Are Not Your Friend

<https://everydayanalytics.ca/2014/08/stacked-area-graphs-are-not-your-friend.html>

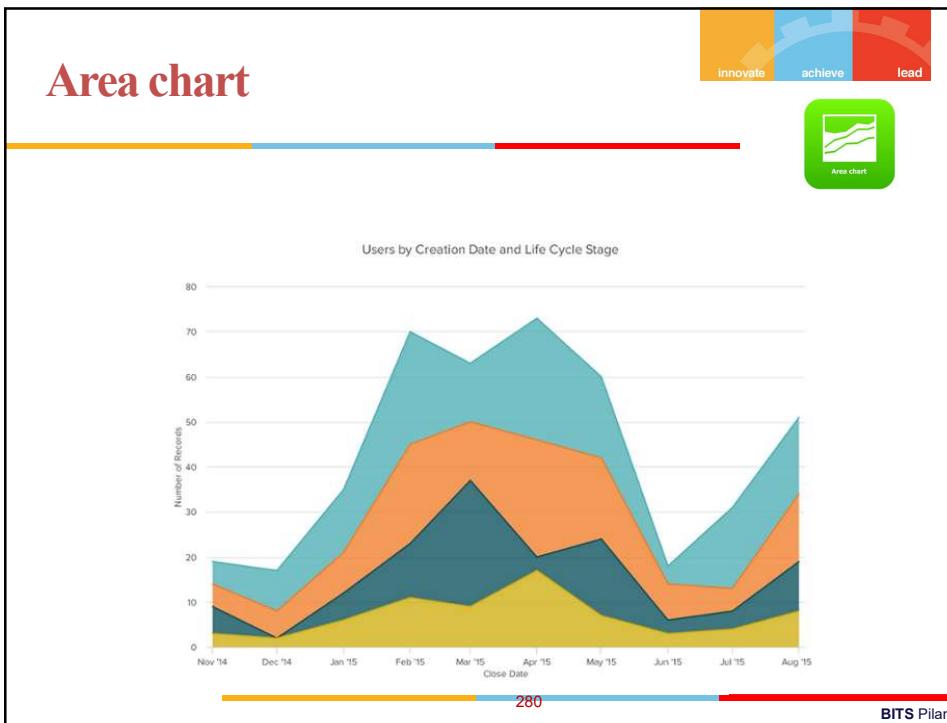
278

BITS Pilani

278



279



280

Candlestick chart



Data variables: 1 x quantitative-interval, 4 x quantitative-ratio.

Visual variables: Position, height, color-hue.

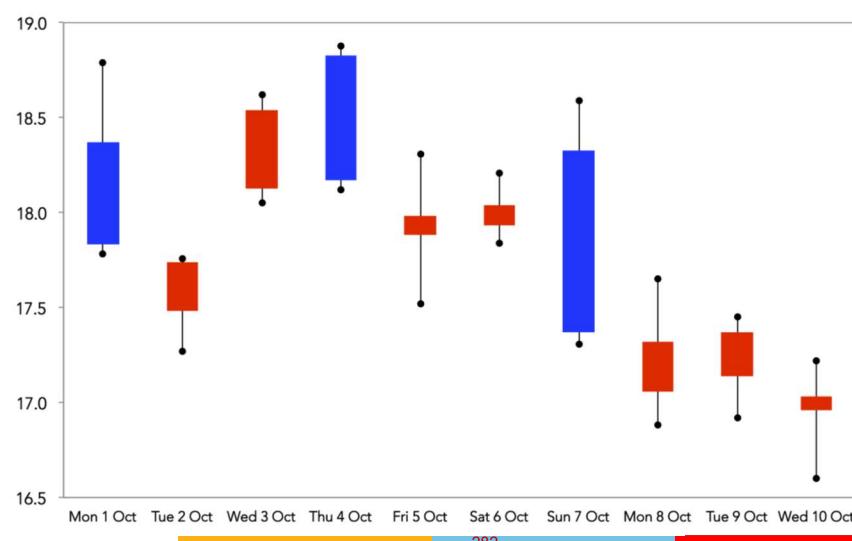
Description: The candlestick chart is commonly used in financial contexts to reveal the key statistics about a stock market for a given timeframe (often daily).

281

BITS Pilani

281

Candlestick chart



282

BITS Pilani

282



Flow map

Data variables: Multiple x quantitative-interval, 1 x categorical, 1 x quantitative-ratio.

Visual variables: Position, height/width, color-hue.

Description: A flow map portrays the flow of a quantitative value as it is transformed over time and/or space

283

BITS Pilani

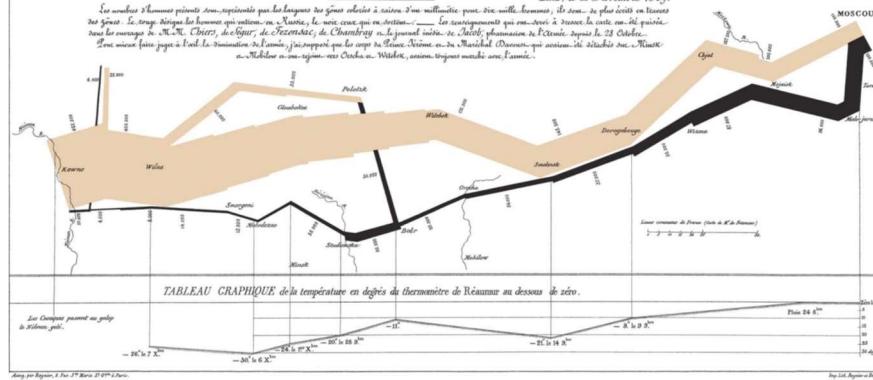
283



Flow map

Carte Figurative des pertes subies en hommes de l'Armée Française dans la campagne de Russie. 1812 - 1813.

Créée par M. Minard, Ingénieur Général des Ponts et Chaussées en Sénat. Paris, le 20 Novembre 1869.
Les nombres d'hommes perdus, représentés par les largeurs des flèches colorées à raison d'une millième pour être nulle, montrent, de tout, de plus précisément que toutes les pertes. Le temps dépend des batailles qui ont lieu, en Russie, le mois, qui se déroule, les distances qui sont parcourues à diverses époques du périple, dans les campagnes de Moscou, de Crimée, de l'Empereur, de Chouïev, de l'Amiral, de l'Amiral, qui suivent les 22 Octobre. Les routes font jusqu'à l'est, la frontière de l'Asie, j'ai supposé que les corps de l'Armée Napoléon en la Marche d'Orient sur l'Asie, en Malaisie ou au contraire, aux Indes ou l'Asie, étaient toujours marchés avec l'assaut.



284

BITS Pilani

284

Plotting connections and relationships-Scatter plot



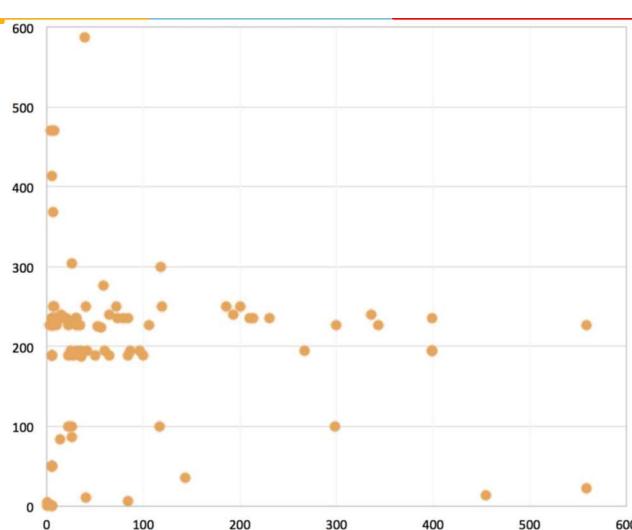
Data variables: 2 x quantitative.

Visual variables: Position, color-hue.

285

BITS Pilani

Scatter plot



286

BITS Pilani

286



Scatter Plots

- **Best Practices for designing Scatter Plots:**
- Include more variables, such as different sizes, to incorporate more data.
- Start y-axis at 0 to represent data accurately.
- If you use trend lines, only use a maximum of two to make your plot easy to understand.

287

BITS Pilani

287



Bubble plot

Data variables: 2 x quantitative, 2 x categorical.

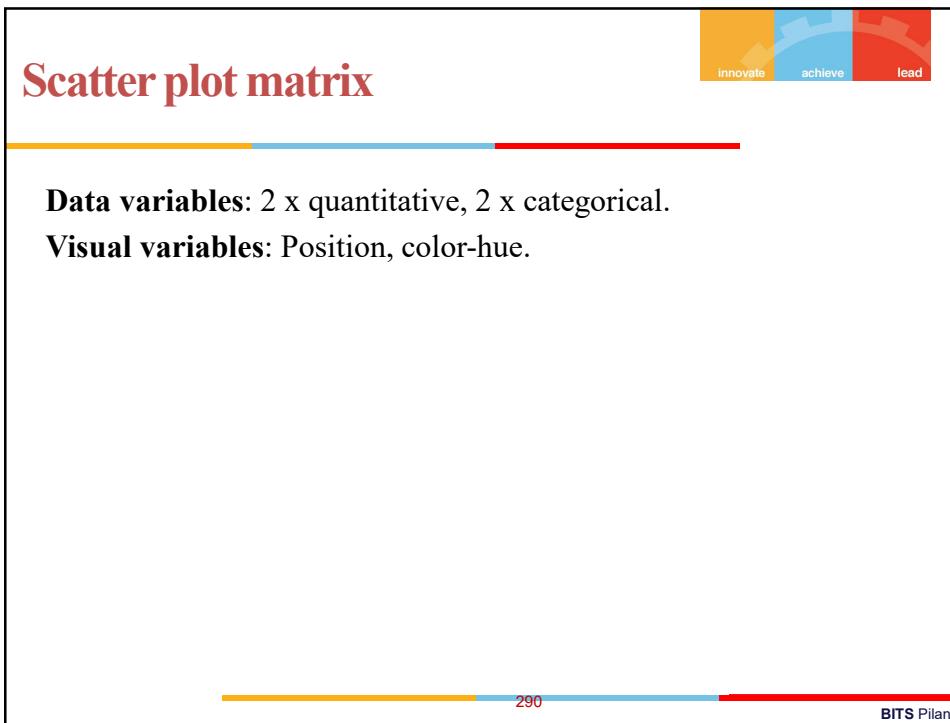
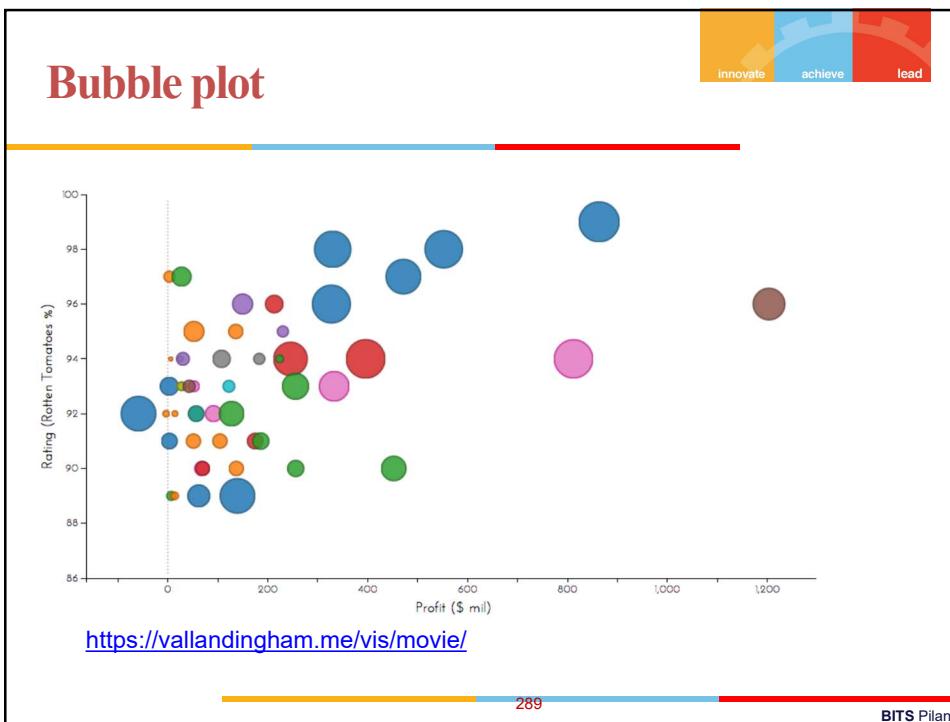
Visual variables: Position, area, color-hue.

Description: A bubble plot extends the potential of a scatter plot through multiple encoding of the data mark

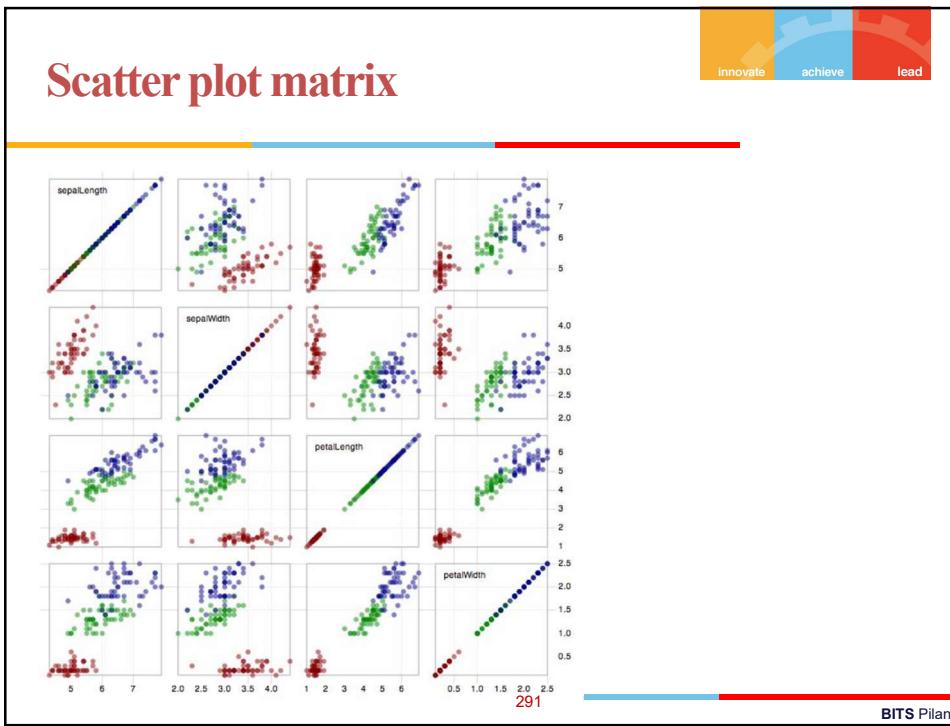
288

BITS Pilani

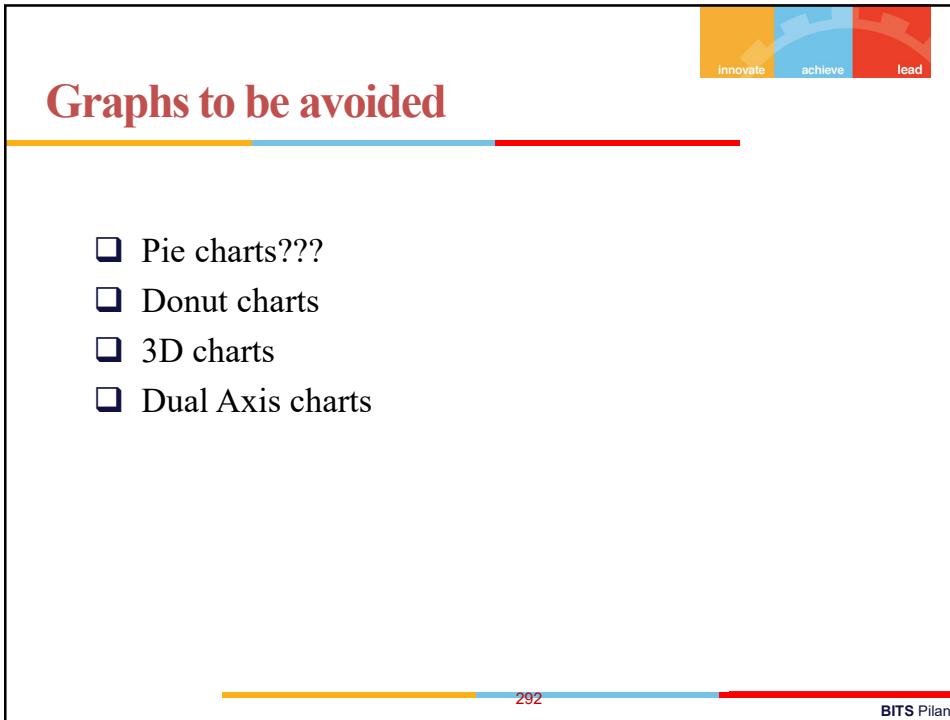
288



290



291



292



Graphs to be avoided

Pie and Donut charts

- Most widely used (and misused)
- used to visualize a part to whole relationship or a composition
- typically represents numbers in percentages,

- The human mind thinks linearly but, when it comes to angles and areas, most of us can't judge them well.

293

BITS Pilani

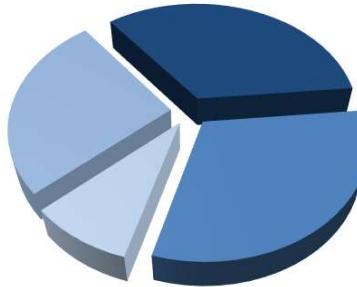
293



Graphs to be avoided

Pie chart Example

Supplier Market Share



- Supplier A
- Supplier B
- Supplier C
- Supplier D

294

BITS Pilani

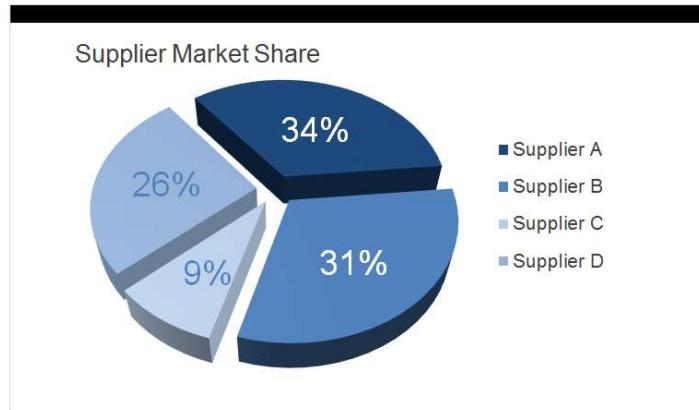
294



Graphs to be avoided



Pie chart Example (with numbers)



BITS Pilani

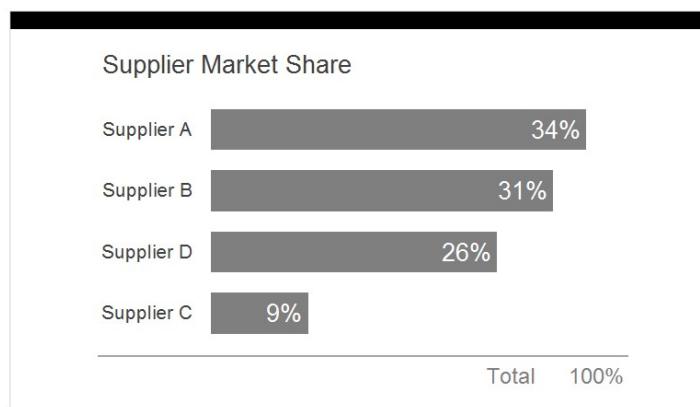
295



Graphs to be avoided



Alternative to Pie chart



BITS Pilani

296



Graphs to be avoided



Pie charts (Dos and Don'ts)

- Make sure that the total sum of all segments equals 100 percent.
- Use pie charts only if you have less than six categories, unless there's a clear winner you want to focus on.
- Ideally, there should be only two categories, like men and women visiting your website, or only one category, like a market share of your company, compared to the whole market.
- Don't use a pie chart if the category values are almost identical or completely different. You could add labels, but that's a patch, not an improvement.
- Don't use 3D or blow apart effects — they reduce comprehension and show incorrect proportions.

297

BITS Pilani

297



Graphs to be avoided



3D charts

- Never use 3D to plot one or two dimensions
- Only use when really third dimension is needed
- Makes difficult to interpret number
- Makes comparisons impossible

298

BITS Pilani

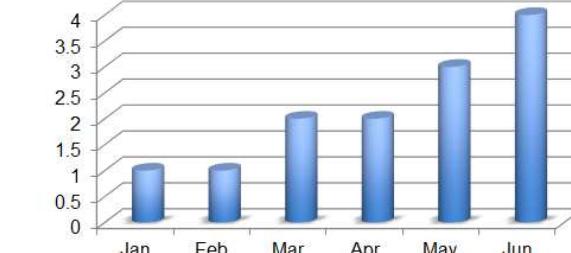
298



Graphs to be avoided

3D Chart Example

Number of issues



Month	Number of issues
Jan	1.2
Feb	1.2
Mar	2.2
Apr	2.2
May	3.2
Jun	4.2

BITS Pilani

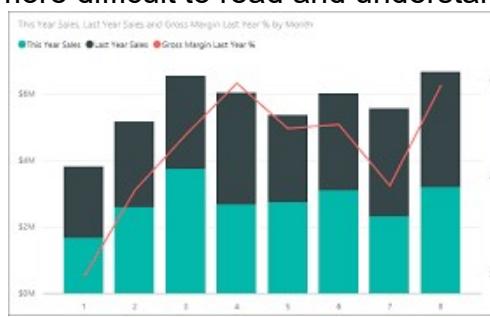
299



Graphs to be avoided

Dual Axis charts

- Used when to show relationships and compare variables on vastly different scales
- Plot data using two y-axes and one shared x-axis
- Much more difficult to read and understand



Month	This Year Sales (\$M)	Last Year Sales (\$M)	Gross Margin Last Year (%)
1	10	10	40%
2	12	12	42%
3	15	15	45%
4	14	14	43%
5	13	13	41%
6	14	14	42%
7	12	12	39%
8	16	16	45%

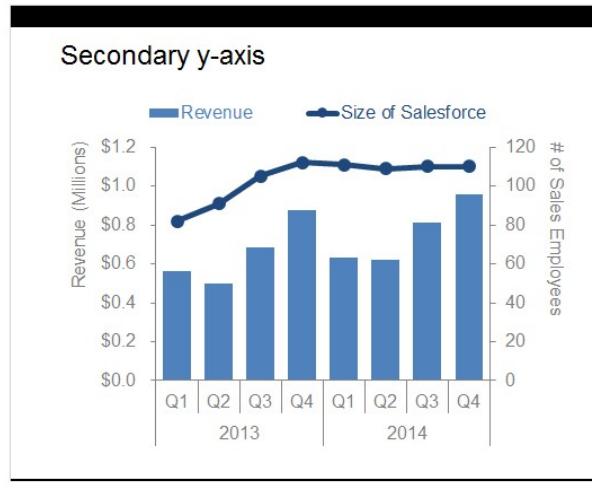
BITS Pilani

300



Graphs to be avoided

Dual Axis chart Example



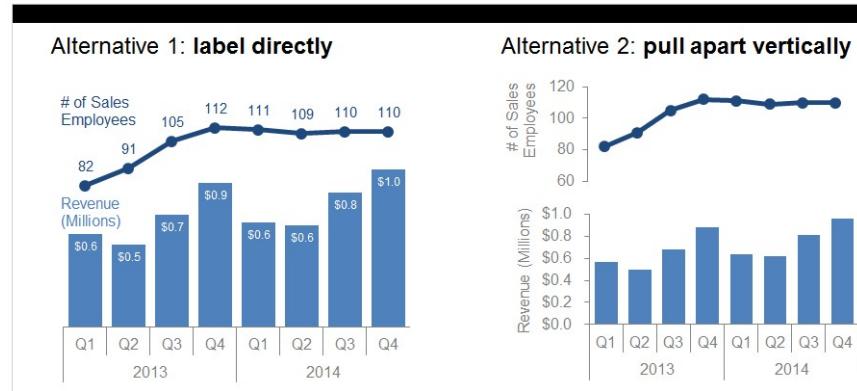
BITS Pilani

301



Graphs to be avoided

Dual Axis chart Alternative



BITS Pilani

302

Recap



❑ Visualization Types

❑ Text Based

- Simple text
- Table
- Heatmap

❑ Graphs

- Points
- Lines
- Bars
- Areas

❑ To be avoided

- Pie and Donut
- 3D
- Dual Axis

303

BITS Pilani

303

Cases

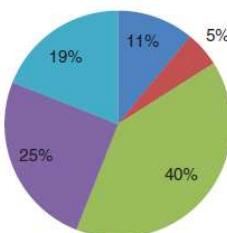


Case 1: Alternative to Pie Charts

Survey results: summer learning program on science

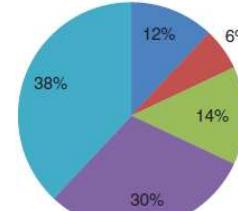
PRE: How do you feel about doing science?

■ Bored ■ Not great ■ OK ■ Kind of interested ■ Excited



POST: How do you feel about doing science?

■ Bored ■ Not great ■ OK ■ Kind of interested ■ Excited



304

BITS Pilani

304

Case 1



Alternative 1: Use Texts to show the numbers

Pilot program was a success

After the pilot program,

68%

of kids expressed interest towards science,
compared to 44% going into the program.

Based on survey of 100 students conducted before and after pilot program (100% response rate on both surveys).

305
BITS Pilani

305

Case 1

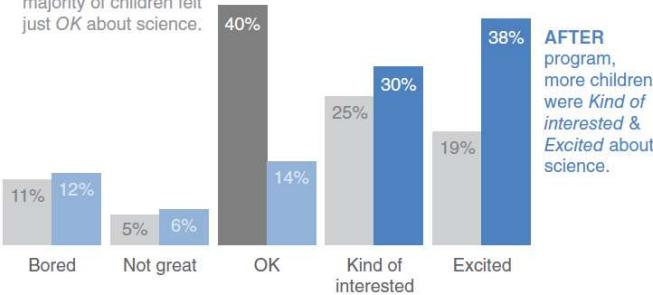


Alternative 2: Simple bar graph

Pilot program was a success

How do you feel about science?

BEFORE program, the majority of children felt just *OK* about science.



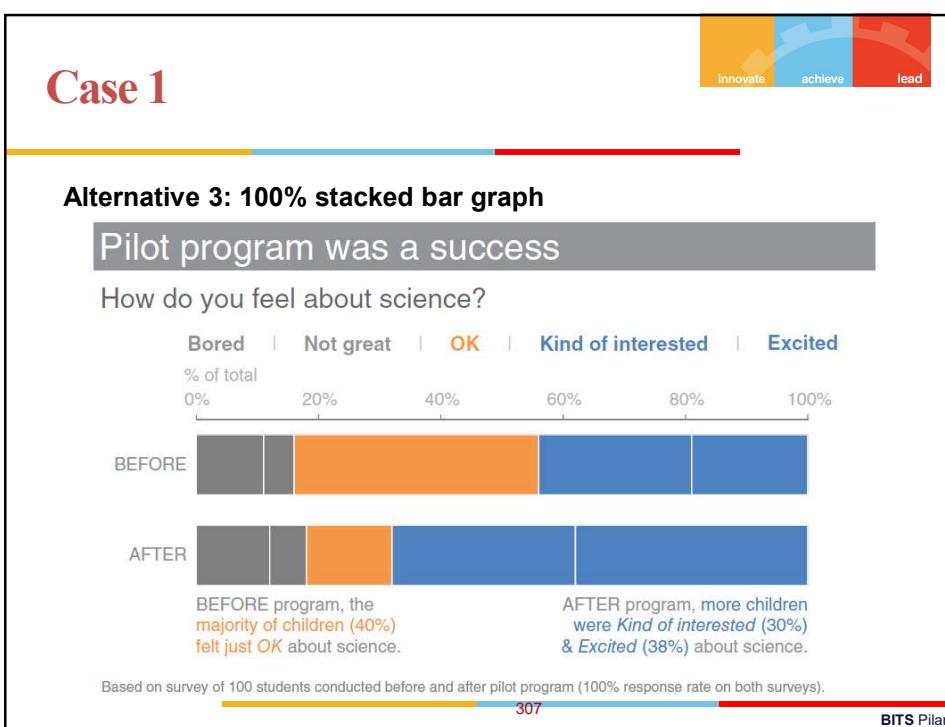
Feeling	Percentage (Before)	Percentage (After)
Bored	11%	12%
Not great	5%	6%
OK	40%	14%
Kind of interested	25%	30%
Excited	19%	38%

AFTER program, more children were *Kind of interested* & *Excited* about science.

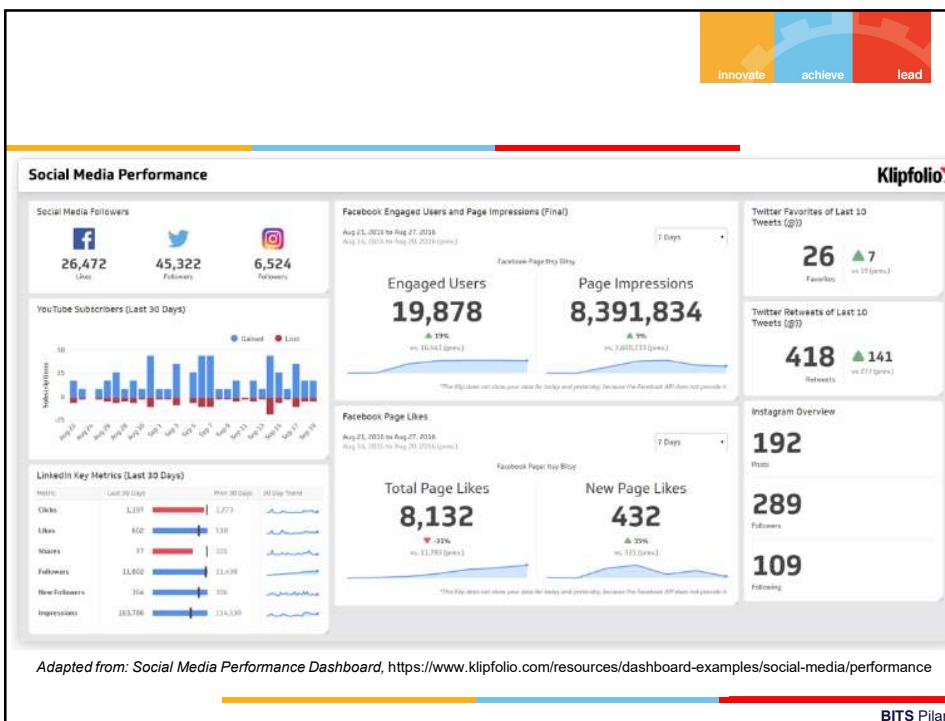
Based on survey of 100 students conducted before and after pilot program (100% response rate on both surveys).

306
BITS Pilani

306



307



308



Agenda

- ❑ Information Dashboard
 - ✓ Definition
 - ✓ Three 3's
 - ✓ Characteristics
- ❑ Common Dashboard Design Mistakes
 - ✓ Design issues
 - ✓ Display issues
 - ✓ Data issues

BITS Pilani

309



Information Dashboard

Definition

A performance dashboard is a layered information delivery system that parcels out information, insights and alerts to users on demand so they can measure, monitor and manage business performance more effectively.

-- Wayne Eckerson



BITS Pilani

310

Information Dashboard

Definition

A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance.¹

BITS Pilani

311

Information Dashboard

Dashboard

- Organization's magnifying glass
- Translates organization's strategy into objectives, measures, metrics
- Provides timely information and insights to enable business users to improve decisions, optimize processes, plans

BITS Pilani

312

Information Dashboard

Lets business people

- Monitor
 - ✓ Critical business processes and activities
- Analyze
 - ✓ The root cause of problems
- Manage
 - ✓ People and processes to improve decisions, optimize processes

BITS Pilani

313

Information Dashboard

Benefits

- Communicate strategy
- Refine strategy
- Increase visibility
- Increase coordination
- Consistent view of business
- Reduce costs and redundancy
- Deliver actionable information

BITS Pilani

314

Three 3's




Three 3's

- Three Applications
- Three Layers
- Three Types

BITS Pilani

315

Three 3's



Three Applications

- Monitoring
 - ✓ Monitor performance against metrics defined
 - ✓ Usually used at operational levels
- Analysis
 - ✓ Allows exploration of data across many dimensions
 - ✓ Helps to identify the root cause of exceptional conditions
- Management
 - ✓ Foster collaboration and decision making
 - ✓ Support executive meetings

BITS Pilani

316

Three 3's



Three Applications

	Monitoring	Analysis	Management
Purpose	Provide information at one sight	Observe exceptions and drill to detail	Decide and refine business strategy
Consists of	Dashboard Scorecard BI Portals Alerts	OLAP Reporting Ad hoc queries	KPIs

BITS Pilani

317

Three 3's

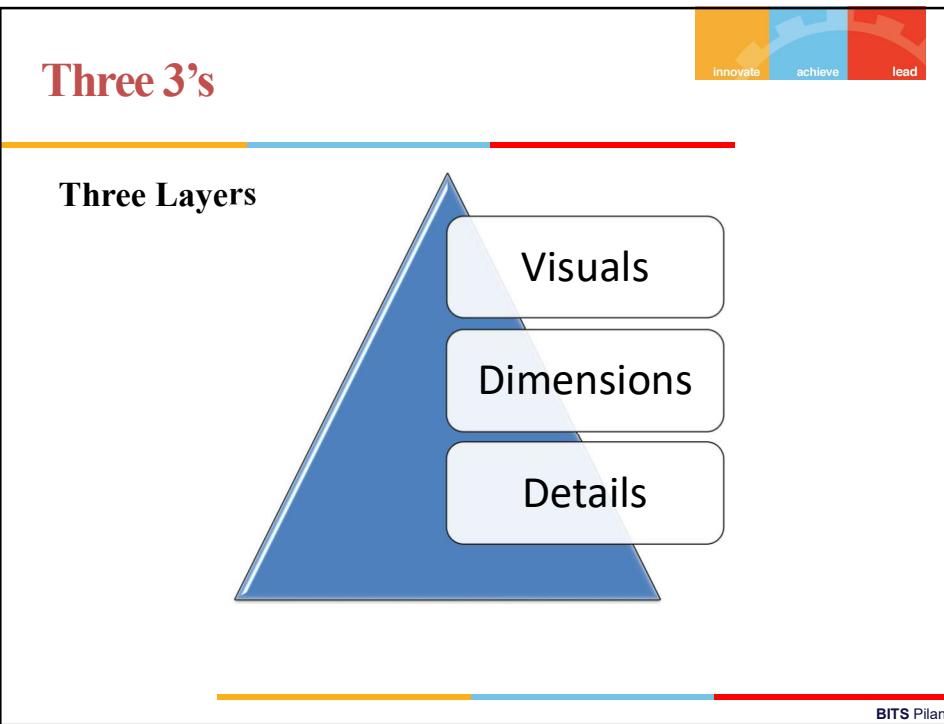


Three Layers

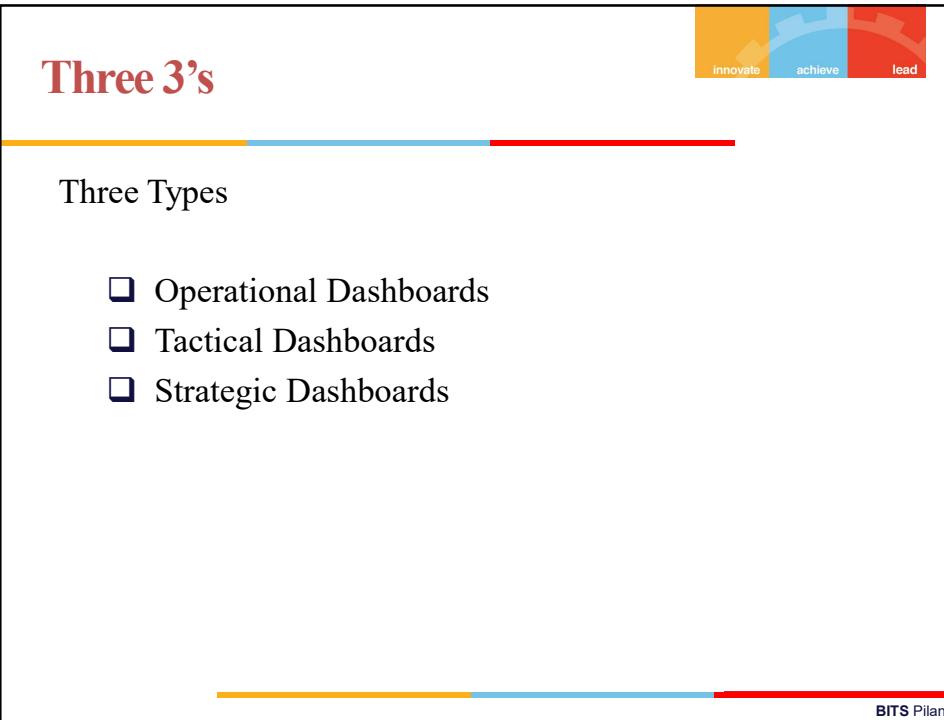
- Summarised / KPI data
- Dimensional data
- Detailed data

BITS Pilani

318



319



320



Three 3's

Operational Dashboard

- Used by front line managers to manage processes
- Uses detailed data which is refreshed continuously
- Monitoring application

BITS Pilani

321



Three 3's

Tactical Dashboard

- More emphasis on analysis
- Uses multidimensional view of data
- Monitor and manage departmental processes and projects

BITS Pilani

322

Three 3's



Strategic Dashboard

- Implemented as scorecards
- Monitors execution of strategic objectives
- Used for review performance by high level executives

BITS Pilani

323

Three 3's



Types comparison

	Operational	Tactical	Strategic
Objective	Processes Monitoring	Process Improvement	Strategy refinement
Level	Process	Department	Organization
Users	Front line / LOB managers	Analyst / Managers	Executives
Focus area	Act immediately	Root cause analysis	Future Perspective
Data refresh	Daily	Week or higher	Month or higher
Data	Detailed	Dimensional	Summarized

BITS Pilani

324



Dashboard Characteristics

Visual Displays

- Displays information needed to achieve objective
- Fits on single computer screen
- Displays information in web browser
- Monitors information at a glance
- Small, clear, intuitive display mechanisms
- Customized

BITS Pilani

325



Dashboard Design

Challenge

How to fit great amount of information into small amount of space?

--- still keeping the easily usable and understandable display

BITS Pilani

326

Dashboard Design Mistakes

Design issues

1. Exceeding the boundaries of single screen
2. Meaningless Variety
3. Cluttering the display with useless decoration
4. Unattractive visual display

BITS Pilani

327

Design issues

Exceeding the boundaries of single screen - segments

Executive Dashboard

Past Due Loans as a % of Total Loans
June 30, 2003

Channel	Loan Type	Balance
ALL	ALL	ALL
Less than 30 days		1.26%
30-60 Days		1.47%
More than 60 Days		1.26%
		3.98%

Figure 3-1. This dashboard fragments the data in a way that undermines the viewer's ability to see meaningful relationships.

BITS Pilani

328

Design issues

Exceeding the boundaries of single screen - scrolling

The dashboard includes several charts and a matrix diagram:

- Top Products by Profit:** Two donut charts for 'Promoted Products - Profit' and 'Top 10 Products - Profit' comparing 01/01/2000 and 01/01/2001.
- PC Sales:** A donut chart for 'Units Sold - PCs' comparing 01/01/2000 and 01/01/2001.
- Product Tracker:** A summary section with a pie chart for 'No Transactions' (5,360, 8,665, 59.8%), a profit comparison for 'Last Year' (3,742,457) and 'This Year' (3,278,919, -12.4%), and a value comparison for 'Value Invoiced' (7,904, 10,748, 63.3%).
- Top 10 Products by Units Sold - Current Full Year:** A table listing products with their units sold, units different, % change, and contribution.
- Product Matrix:** A 4x4 matrix diagram with arrows indicating flow between categories based on growth and volume.

Figure 3-3. This dashboard demonstrates the effectiveness that is sacrificed when scrolling is required to see all the information.

BITS Pilani

329

Design issues

Meaningless Variety

The dashboard displays the following components:

- Revenue by Region:** A title bar with 'Cognos Visualizer Web Edition 1.6' and a toolbar.
- Revenue over Margin:** Three circular icons representing Barbados (red), Canada (green), and USA (yellow).
- Revenue vs Average Margin:** A line chart comparing revenue against average margin for Barbados, Canada, and USA.
- Revenue by Country:** A bar chart showing revenue for Barbados, Canada, and USA.
- Order Method:** A bar chart showing sales by order method: TeleSales, e-Commerce, Mail Order, and Direct Sales.
- Table:** A summary table with columns: Total Sales Net Amount, Total Sales Profit Margin, Total Number of Sales, and Average Sales Net Amount.

Figure 3-18. This dashboard exhibits an unnecessary variety of display media.

BITS Pilani

330

Design issues



Cluttering the display with useless decoration

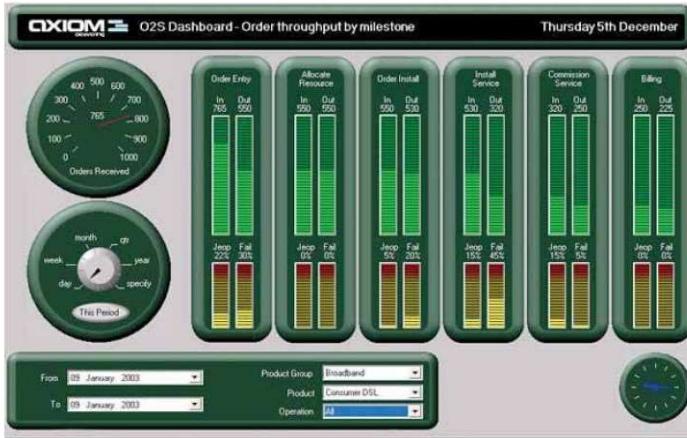


Figure 3-28. This dashboard is trying to look like something that it is not, resulting in useless and distracting decoration.

BITS Pilani

331

Design issues



Cluttering the display with useless decoration

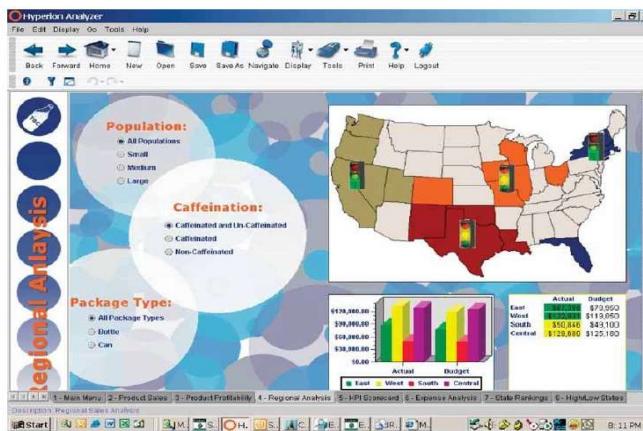


Figure 3-30. This dashboard is a vivid example of distracting ornamentation.

BITS Pilani

332

Design issues

Unattractive visual display

Figure 3-32. This is an example of a rather unattractive dashboard.

BITS Pilani

333

Dashboard Design Mistakes

Display issues

- 5. Inappropriate display media
- 6. Poorly designed display media
- 7. Misusing or overusing colors

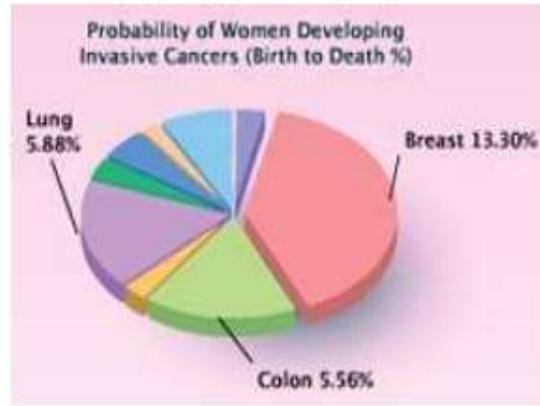
BITS Pilani

334



Display issues

Inappropriate display media – Problems with Pie chart



BITS Pilani

335



Display issues

Inappropriate display media

- Uses the 2D area of circles to encode their values, which needlessly obscures the data



BITS Pilani

336

Display issues



Inappropriate display media

- Uselessly encodes quantitative values on a map of the US



BITS Pilani

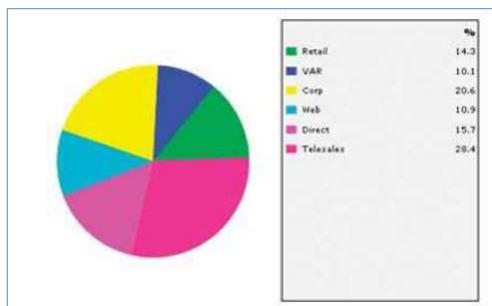
337

Display issues



Poorly designed display media

- This pie chart illustrates several design problems



BITS Pilani

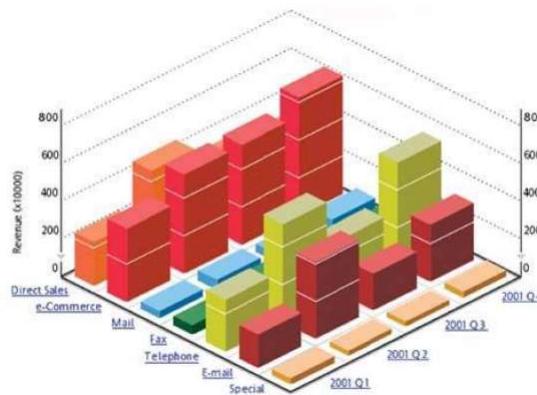
338

Display issues



Poorly designed display media

- Difficult to read for the 3D design



BITS Pilani

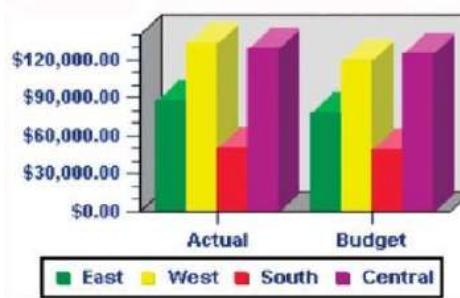
339

Display issues



Poorly designed display media

- Many examples of poor design features



BITS Pilani

340

Dashboard Design Mistakes



Data issues

8. Inadequate context for the data
9. Using excessive detail or precision
10. Using deficient measure
11. Incorrect data encoding
12. Poor data arrangement
13. Ineffective data highlighting

BITS Pilani

341

Data issues



Inadequate context for the data

- These dashboard gauges fail to provide adequate context to make the measures meaningful



BITS Pilani

342

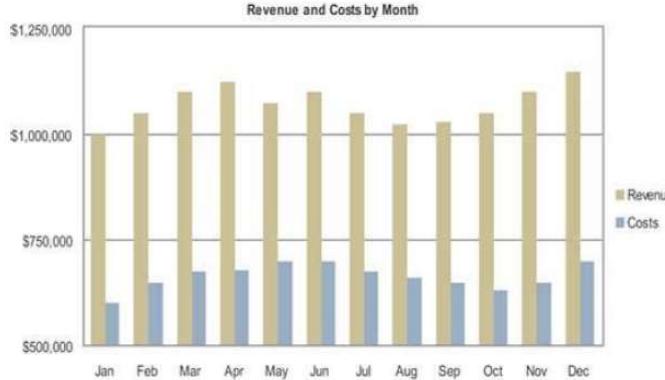
The screenshot shows the CELEQUEST Quality Yield Analysis dashboard. At the top, there's a navigation tree with 'Dashboards' selected, and under it, 'All Dashboards' and 'Bookmarked Dashboards'. Below this is a section titled 'Active Alert Messages' with a table showing several alerts, each with a subject, importance level (e.g., Normal, High), and activation date. Below the alert table are three charts: 'Board Yield Barchart', 'Board Yield Change Barchart', and 'Tests Breakdown Pie'. The 'Board Yield Barchart' shows yield percentages for different categories. The 'Board Yield Change Barchart' shows yield change percentages. The 'Tests Breakdown Pie' chart shows the distribution of test types. At the bottom, there's a 'Board Yield Table Summary' table with columns like PRODUCT_DESC, YIELD_2004, YIELD_2005, etc. The dashboard has a header with 'innovate', 'achieve', and 'lead' sections, and a footer with 'BITS Pilani'.

343

The screenshot shows a line graph titled 'Revenue Actual to Budget Variance'. The y-axis represents revenue in dollars from \$0 to \$120,000. The x-axis represents weeks from 1 to 20. Two lines are plotted: 'Budget' (blue line) and 'Actual' (orange line). The actual revenue generally follows the budget line but shows significant fluctuations, particularly a peak around week 15 and a dip around week 17. The graph has a header with 'innovate', 'achieve', and 'lead' sections, and a footer with 'BITS Pilani'.

344

Data issues



The chart displays monthly revenue and costs. Revenue is represented by brown bars and costs by blue bars. The y-axis ranges from \$500,000 to \$1,250,000. The x-axis shows months from January to December. Revenue values are approximately: Jan (\$1,050,000), Feb (\$1,100,000), Mar (\$1,150,000), Apr (\$1,200,000), May (\$1,100,000), Jun (\$1,150,000), Jul (\$1,050,000), Aug (\$1,000,000), Sep (\$1,050,000), Oct (\$1,100,000), Nov (\$1,150,000), Dec (\$1,200,000). Cost values are approximately: Jan (\$600,000), Feb (\$650,000), Mar (\$680,000), Apr (\$680,000), May (\$700,000), Jun (\$720,000), Jul (\$680,000), Aug (\$680,000), Sep (\$680,000), Oct (\$650,000), Nov (\$680,000), Dec (\$700,000).

Incorrect data encoding

Revenue and Costs by Month

Revenue

Costs

BITs Pilani

345

Data issues



The dashboard features several charts and an alert panel. The charts include a donut chart for Order Sizes, a pie chart for Sales, a line graph for Order Size Trends, a gauge chart for Count, and a line graph for Profit Trends. The alert panel lists various notifications such as 'Alerts have exceeded 40s', 'Levers have exceeded 100s', and 'Revenue pipe for quarter 25 before target'. A separate chart titled 'Computers Returns Across Models' shows the percentage of returns for three desktop models across different categories.

Poor data arrangement

- This dashboard exemplifies poorly managed data

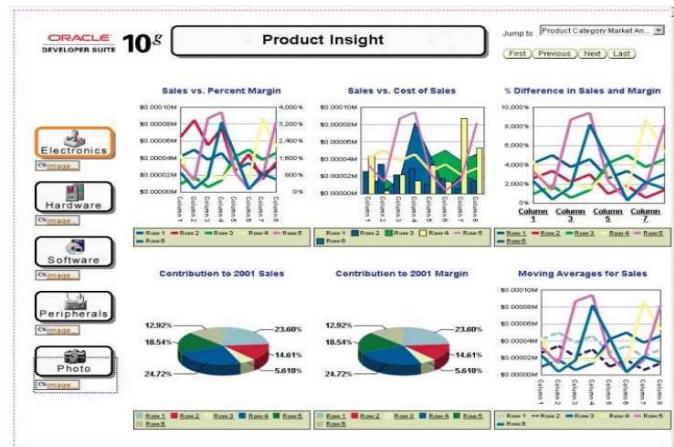
BITs Pilani

346



Data issues

Ineffective data highlighting



BITS Pilani

347



Recap

Information Dashboard

- ✓ Definition
- ✓ Three 3's
 - ❖ Three Applications
 - ❖ Three Layers
 - ❖ Three types
- ✓ Characteristics

BITS Pilani

348

Recap

Common Dashboard Design Mistakes

- ✓ Design issues
 - ❖ Exceeding the boundaries of single screen
 - ❖ Meaningless Variety
 - ❖ Cluttering the display with useless decoration
 - ❖ Unattractive visual display
- ✓ Display issues
 - ❖ Inappropriate display media
 - ❖ Poorly designed display media
 - ❖ Misuse or overuse of colors
- ✓ Data issues
 - ❖ Inadequate context for the data
 - ❖ Using excessive detail or precision
 - ❖ Using deficient measure
 - ❖ Incorrect data encoding
 - ❖ Poor data arrangement
 - ❖ Ineffective data highlighting



BITS Pilani

349

References

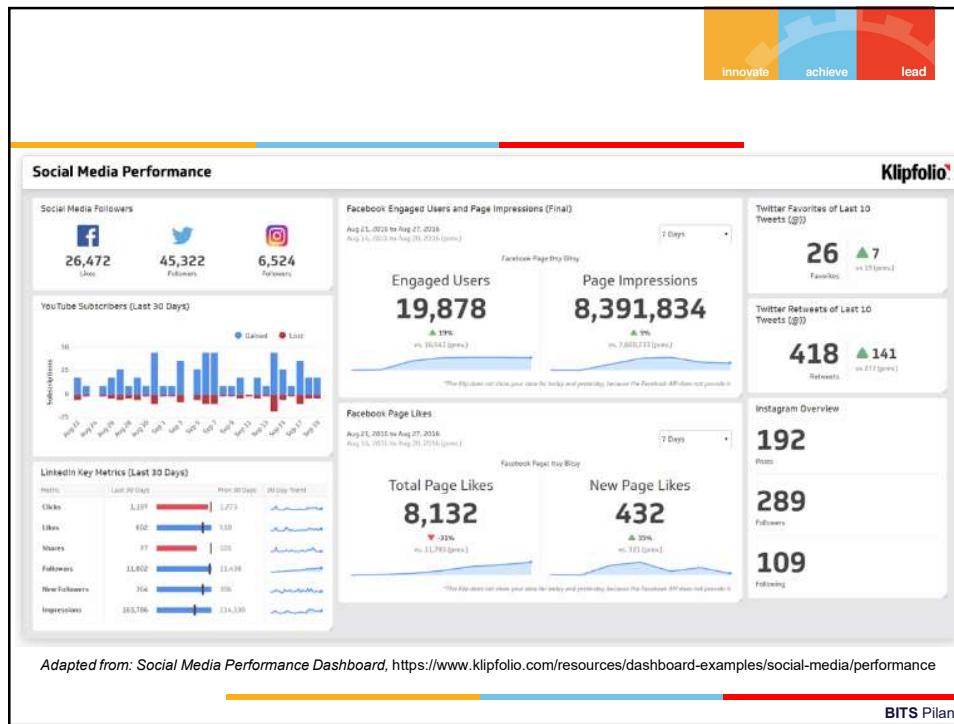
Information Dashboard Design - Stephen Few

- Chapter 1 : Clarifying the vision
- Chapter 2: Variations in Dashboard Uses and Data
- Chapter 3 : Thirteen Common Mistakes in Dashboard Design



BITS Pilani

350



351

Contact Session 7-Agenda

inovate achieve lead

- ❑ Information Dashboard
 - ✓ Definition
 - ✓ Three 3's
 - ✓ Characteristics
- ❑ Common Dashboard Design Mistakes
 - ✓ Design issues
 - ✓ Display issues
 - ✓ Data issues

BITS Pilani

352

Information Dashboard

Definition

A performance dashboard is a layered information delivery system that parcels out information, insights and alerts to users on demand so they can measure, monitor and manage business performance more effectively.

-- Wayne Eckerson

BITS Pilani

353

Information Dashboard

Definition

A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance.¹

BITS Pilani

354

Dashboard Characteristics



Visual Displays

- Displays information needed to achieve objective
- Fits on single computer screen
- Displays information in web browser
- Monitors information at a glance
- Small, clear, intuitive display mechanisms
- Customized

BITS Pilani

355

Information Dashboard



Dashboard

- Organization's magnifying glass
- Translates organization's strategy into objectives, measures, metrics
- Provides timely information and insights to enable business users to improve decisions, optimize processes, plans

BITS Pilani

356

Information Dashboard

Lets business people

- Monitor
 - ✓ Critical business processes and activities
- Analyze
 - ✓ The root cause of problems
- Manage
 - ✓ People and processes to improve decisions, optimize processes

BITs Pilani

357

Information Dashboard

Benefits

- Communicate strategy
- Refine strategy
- Increase visibility
- Increase coordination
- Consistent view of business
- Reduce costs and redundancy
- Deliver actionable information

BITs Pilani

358

Three 3's




Three 3's

- Three Applications
- Three Layers
- Three Types

BITS Pilani

359

Three 3's



Three Applications

- Monitoring
 - ✓ Monitor performance against metrics defined
 - ✓ Usually used at operational levels
- Analysis
 - ✓ Allows exploration of data across many dimensions
 - ✓ Helps to identify the root cause of exceptional conditions
- Management
 - ✓ Foster collaboration and decision making
 - ✓ Support executive meetings

BITS Pilani

360

Three 3's



Three Applications

	Monitoring	Analysis	Management
Purpose	Provide information at one sight	Observe exceptions and drill to detail	Decide and refine business strategy
Consists of	Dashboard Scorecard BI Portals Alerts	OLAP Reporting Ad hoc queries	KPIs

BITS Pilani

361

Three 3's

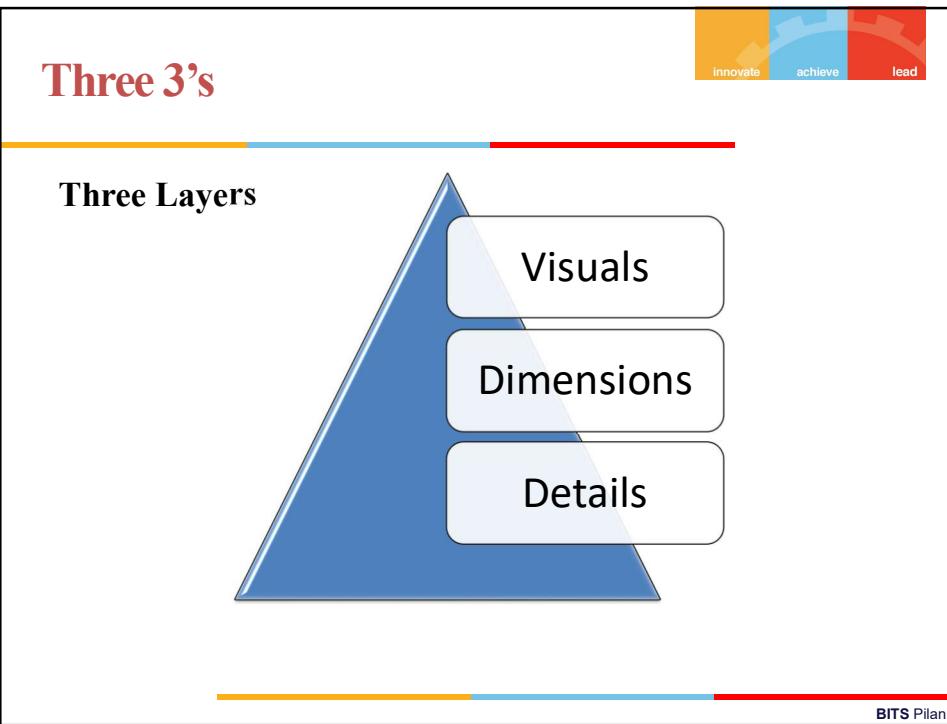


Three informational Layers

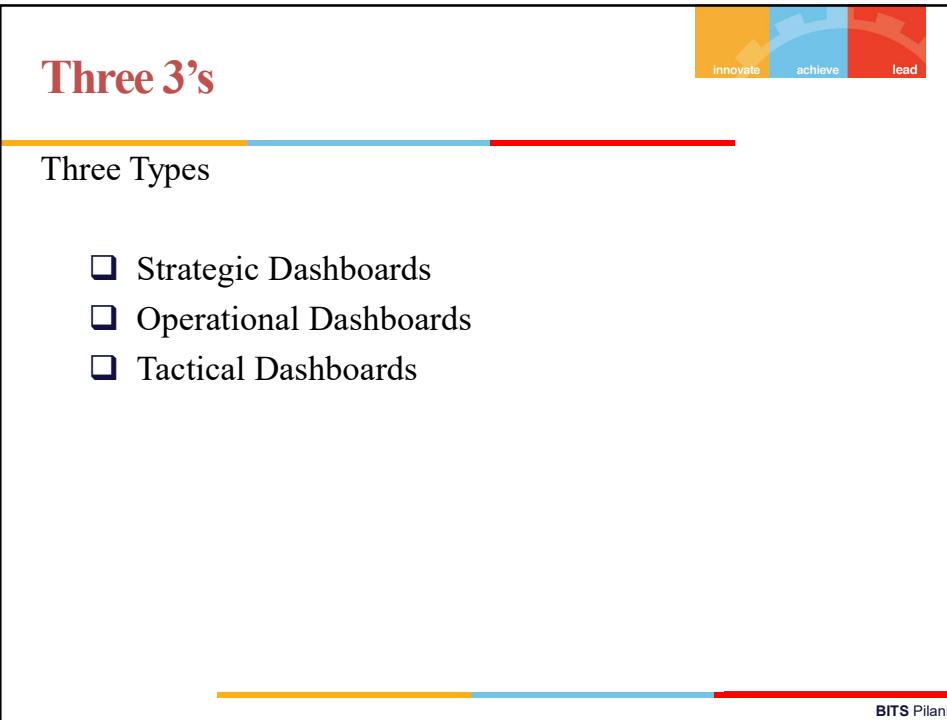
- Summarised / KPI data
- Dimensional data
- Detailed data

BITS Pilani

362



363



364

Three 3's



Strategic Dashboard

- Implemented as scorecards
- Monitors execution of strategic objectives
- Used for review performance by high level executives

BITS Pilani

365

Three 3's



Tactical Dashboard

- More emphasis on analysis
- Uses multidimensional view of data
- Monitor and manage departmental processes and projects

BITS Pilani

366



Three 3's

Operational Dashboard

- Used by front line managers to manage processes
- Uses detailed data which is refreshed continuously
- Monitoring application

BITS Pilani

367



Three 3's

Types comparison

	Operational	Tactical	Strategic
Objective	Processes Monitoring	Process Improvement	Strategy refinement
Level	Process	Department	Organization
Users	Front line / LOB managers	Analyst / Managers	Executives
Focus area	Act immediately	Root cause analysis	Future Perspective
Data refresh	Daily	Week or higher	Month or higher
Data	Detailed	Dimensional	Summarized

BITS Pilani

368

Dashboard Design



Challenge

How to fit great amount of information into small amount of space?

--- still keeping the easily usable and understandable display

BITS Pilani

369

Dashboard Design Mistakes



Design issues

1. Exceeding the boundaries of single screen
2. Meaningless Variety
3. Cluttering the display with useless decoration
4. Unattractive visual display

BITS Pilani

370

Dashboard Design Mistakes



Display issues

5. Inappropriate display media
6. Poorly designed display media
7. Misusing or overusing colors

BITS Pilani

371

Dashboard Design Mistakes



Data issues

8. Inadequate context for the data
9. Using excessive detail or precision
10. Using deficient measure
11. Incorrect data encoding
12. Poor data arrangement
13. Ineffective data highlighting

BITS Pilani

372

Design issues

innovate achieve lead

Exceeding the boundaries of single screen - segments

Figure 3-1. This dashboard fragments the data in a way that undermines the viewer's ability to see meaningful relationships.

BITS Pilani

373

Design issues

innovate achieve lead

Exceeding the boundaries of single screen - scrolling

Figure 3-3. This dashboard demonstrates the effectiveness that is sacrificed when scrolling is required to see all the information.

BITS Pilani

374

Data issues

Inadequate context for the data

- These dashboard gauges fail to provide adequate context to make the measures meaningful

October Units YTD Units Returns Rate

BITs Pilani

375

Data issues

Using excessive detail or precision

The screenshot shows the CELEQUEST software interface with the following details:

- Navigation Tree:** Dashboards > All Dashboards > Quality Test Analysis.
- Active Alert Messages:**
 - 8/16/2003 Yield Drop in TSS on 60-000 - Normal [03/15/2004 17:16:00]
 - 8/16/2003 Yield Drop on 60-000160 - High [03/15/2004 17:16:00]
 - 8/16/2003 Yield Drop in ESS on 60-000200 - Normal [03/15/2004 17:10:01]
 - 8/15/2003 Critical Component Failure 60-000 - High [03/15/2004 17:10:00]
 - 8/15/2003 Impacted Boards for 11-000000 - High [03/15/2004 17:09:59]
 - 8/15/2003 Yield Drop in TSS on 60-000 - Normal [03/15/2004 17:09:46]
- Board Yield Barchart:** Shows yield percentages for various board types: 110, 95, 64, 45, 22, 4, 2, 0, -4, -17%, -41%, -6%, -4. The legend includes Yield_11y, Yield_20yrs, Yield_20yrs, and Yield_24yrs.
- Board Yield Change Barchart:** Shows yield change percentages for various board types: 3, -1.7%, -40.0%, -10.0%, -6.0%, -0.8%, -0.6%, -0.2%. The legend includes Yield_Change_ID_1, Yield_Change_ID_2, Yield_Change_ID_3, and Yield_Change_ID_4.
- Tests Breakdown Pie:** A pie chart showing the distribution of test results across four categories: 40-000000x (0%), 60-0000007 (17%), 60-0000201 (49%), and 60-000161 (33%).
- Board Yield Table Summary:** A table showing board yield data for various components:

Product_Mfg	Product_Spec	Yield	Yield_L	Yield_H	Yield_D	Yield_C	Yield_P
60-00000000	PCB-00000000-PN00	100.0000	100.0000	100.0000	0.0000	0.0000	0.0000
60-00000072-01	ASSY-16 PORT CARD,SL,SW2000	99.4000	99.4000	99.4000	0.4505	0.4505	0.4505
60-00002424-06	ASSY-C PLUG LENGTH	100.00000000	100.0000	99.1549	0.8451	0.8451	0.8451
60-0001663-03	ASSY, INNER BOX W/M, SW360	100.00000000	100.0000	99.1111	0.8889	0.8889	0.8889

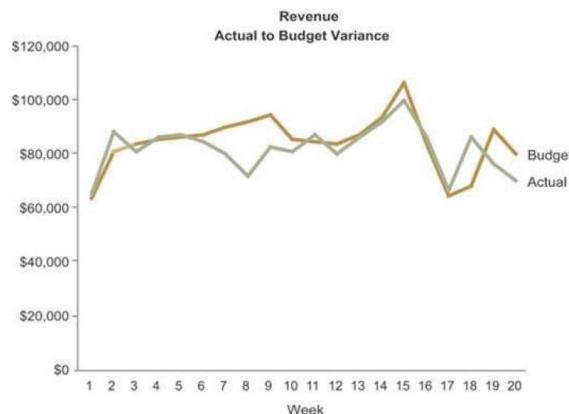
BITs Pilani

376

Data issues



Using deficient measure



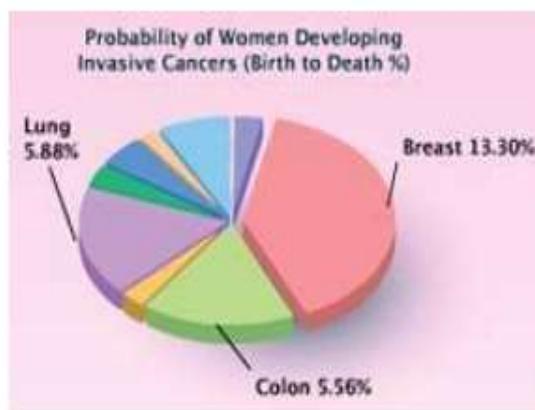
BITS Pilani

377

Display issues



Inappropriate display media – Problems with Pie chart



BITS Pilani

378

189

Display issues

Inappropriate display media
- Uselessly encodes quantitative values on a map of the US

BITS Pilani

379

Design issues

Meaningless Variety

	Total Sales Net Amount	Total Sales Profit Margin	Total Number of Sales	Average Sales Net Amount
1 Canada	\$340,924,098	\$18,480,765	10,760	\$31,300
2 USA	\$413,95K	\$195,275	206	\$1,400K
3 Barbados	\$113,27K	\$11,485	31	\$4,99K

Figure 3-18. This dashboard exhibits an unnecessary variety of display media.

BITS Pilani

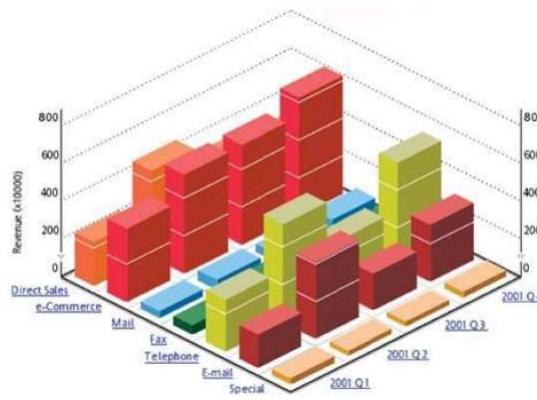
380



Display issues

Poorly designed display media

- Difficult to read for the 3D design



BITS Pilani

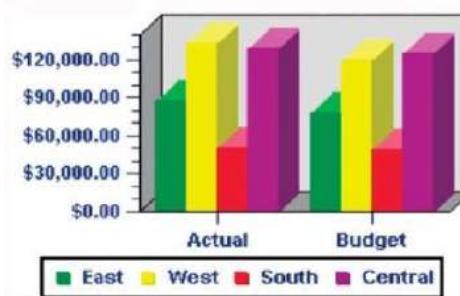
381



Display issues

Poorly designed display media

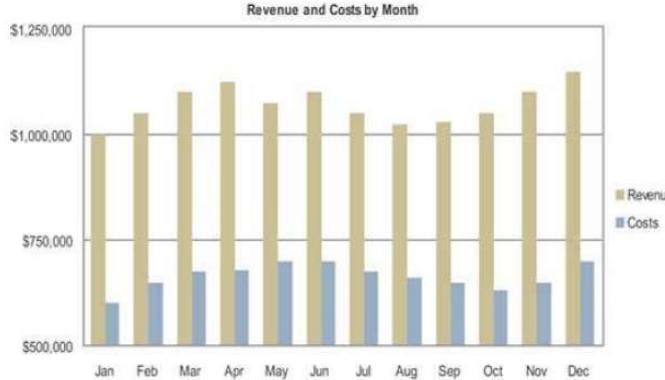
- Many examples of poor design features



BITS Pilani

382

Data issues



The chart displays monthly revenue and costs. Revenue is represented by brown bars and costs by blue bars. The y-axis ranges from \$500,000 to \$1,250,000. The x-axis shows months from January to December. Revenue values are approximately: Jan (\$1,050,000), Feb (\$1,100,000), Mar (\$1,150,000), Apr (\$1,200,000), May (\$1,100,000), Jun (\$1,150,000), Jul (\$1,050,000), Aug (\$1,000,000), Sep (\$1,050,000), Oct (\$1,100,000), Nov (\$1,150,000), Dec (\$1,200,000). Cost values are approximately: Jan (\$600,000), Feb (\$650,000), Mar (\$680,000), Apr (\$680,000), May (\$700,000), Jun (\$700,000), Jul (\$680,000), Aug (\$680,000), Sep (\$680,000), Oct (\$650,000), Nov (\$680,000), Dec (\$700,000).

Incorrect data encoding

Revenue and Costs by Month

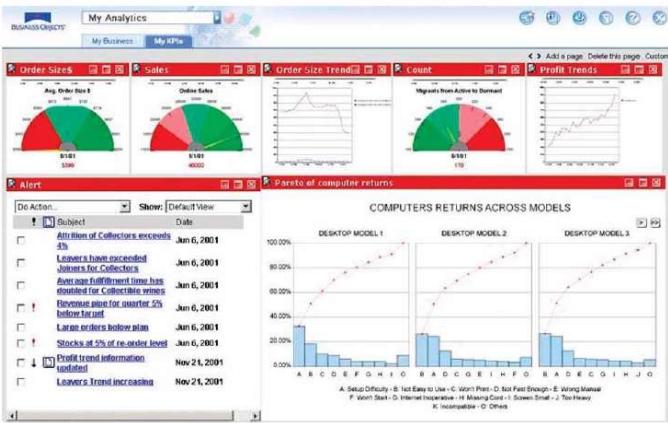
Revenue Costs

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

BITs Pilani

383

Data issues



The dashboard features several data visualizations and an alert list.

- Order Sizes:** A donut chart showing order sizes: 80% S, 10% M, 10% L.
- Sales:** A donut chart showing sales: 80% S, 10% M, 10% L.
- Order Size Trends:** A line graph showing order size trends over time.
- Count:** A donut chart showing counts: 80% S, 10% M, 10% L.
- Profit Trends:** A line graph showing profit trends over time.
- Alert:** A list of alerts:
 - Attention of Collectors exceeds 4%
 - Leavers have exceeded 100% for Collectors
 - Revenue has exceeded 100% for Collectable wines
 - Revenue pipe for quarter 25 before target
 - Larson orders below all
 - Stocks at 5% of re-order level
 - Profit trend information updated
 - Leavers Trend increasing
- Pareto of computer returns:** A chart titled "COMPUTERS RETURNS ACROSS MODELS" showing returns across three desktop models (Model 1, Model 2, Model 3) for various reasons (A-J).

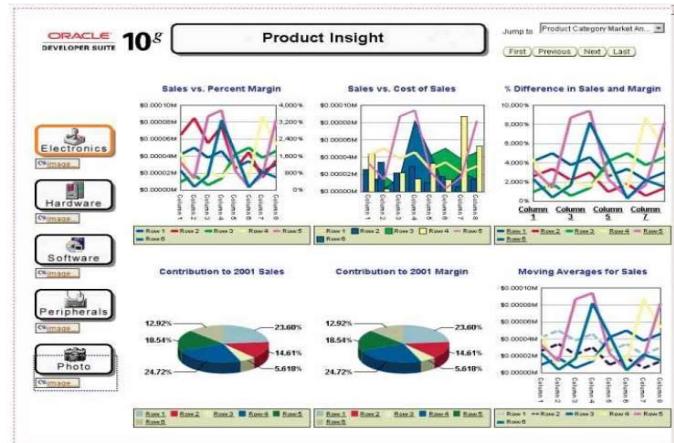
BITs Pilani

384



Data issues

Ineffective data highlighting



BITS Pilani

385



Design issues

Cluttering the display with useless decoration

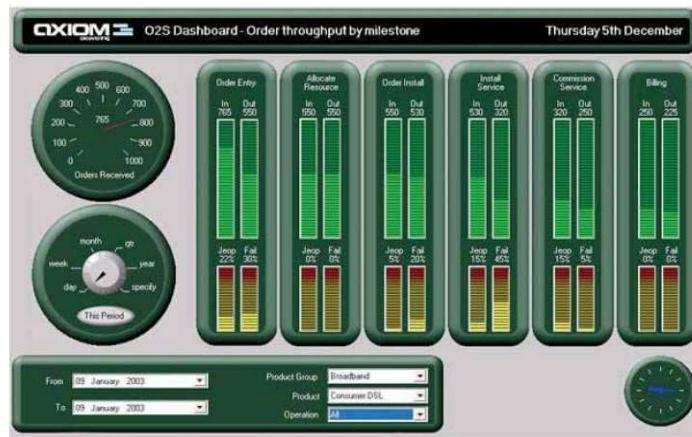


Figure 3-28. This dashboard is trying to look like something that it is not, resulting in useless and distracting decoration.

BITS Pilani

386

Design issues

Cluttering the display with useless decoration

The dashboard is titled 'Hyperion Analyzer' and includes several sections: 'Population' (with bubble charts), 'Caffeination' (a map of the US with colored states), 'Package Type' (radio buttons for bottle and can), and a bar chart comparing 'Actual' and 'Budget' values across East, West, South, and Central regions. The interface is heavily decorated with blue and white circles on the left, a toolbar at the top, and a status bar at the bottom.

Figure 3-30. This dashboard is a vivid example of distracting ornamentation.

BITS Pilani

387

Design issues

Unattractive visual display

The dashboard is titled 'Manufacturing' and contains several charts: a scatter plot of 'Units Produced (x1000)' vs 'Units Required (x1000)', a bar chart for 'Uptime', a histogram for 'LMS IR', a donut chart for 'Material Cost (L1000)', a bar chart for 'Projects', and a bar chart for 'Product Line'. The overall aesthetic is dark and lacks visual appeal.

Figure 3-32. This is an example of a rather unattractive dashboard.

BITS Pilani

388

Recap



Information Dashboard

- ✓ Definition
- ✓ Three 3's
 - ❖ Three Applications
 - ❖ Three Layers
 - ❖ Three types
- ✓ Characteristics

BITS Pilani

389

Recap



Common Dashboard Design Mistakes

- ✓ Design issues
 - ❖ Exceeding the boundaries of single screen
 - ❖ Meaningless Variety
 - ❖ Cluttering the display with useless decoration
 - ❖ Unattractive visual display
- ✓ Display issues
 - ❖ Inappropriate display media
 - ❖ Poorly designed display media
 - ❖ Misuse or overuse of colors
- ✓ Data issues
 - ❖ Inadequate context for the data
 - ❖ Using excessive detail or precision
 - ❖ Using deficient measure
 - ❖ Incorrect data encoding
 - ❖ Poor data arrangement
 - ❖ Ineffective data highlighting

BITS Pilani

390



References

Information Dashboard Design - Stephen Few

- Chapter 1 : Clarifying the vision
- Chapter 2: Variations in Dashboard Uses and Data
- Chapter 3 : Thirteen Common Mistakes in Dashboard Design

BITS Pilani

391



Anscombe's Quartet

- Anscombe stated that you can't just use summary statistics to understand the data, you have to visualize it
- It's not to say that summary statistics aren't important. They are absolutely essential, but you must also visualize it

BITS Pilani

392

Anscombe's Quartet



This is the data that Francis Anscombe used.

x1	y1	x2	y2	x3	y3	x4	y4
10	8.04	10	9.14	10	7.46	8	6.58
8	6.95	8	8.14	8	6.77	8	5.76
13	7.58	13	8.74	13	12.74	8	7.71
9	8.81	9	8.77	9	7.11	8	8.84
11	8.33	11	9.26	11	7.81	8	8.47
14	9.96	14	8.10	14	8.84	8	7.04
6	7.24	6	6.13	6	6.08	8	5.25
4	4.26	4	3.10	4	5.39	8	5.56
12	10.84	12	9.13	12	8.15	19	12.50
7	4.82	7	7.26	7	6.42	8	7.91
5	5.68	5	4.74	5	5.73	8	6.89
x1	y1	x2	y2	x3	y3	x4	y4

BITS Pilani

393

Anscombe's Quartet



	x1	y1	x2	y2	x3	y3	x4	y4
10	8.04	10	9.14	10	7.46	8	6.58	
8	6.95	8	8.14	8	6.77	8	5.76	
13	7.58	13	8.74	13	12.74	8	7.71	
9	8.81	9	8.77	9	7.11	8	8.84	
11	8.33	11	9.26	11	7.81	8	8.47	
14	9.96	14	8.10	14	8.84	8	7.04	
6	7.24	6	6.13	6	6.08	8	5.25	
4	4.26	4	3.10	4	5.39	8	5.56	
12	10.84	12	9.13	12	8.15	19	12.50	
7	4.82	7	7.26	7	6.42	8	7.91	
5	5.68	5	4.74	5	5.73	8	6.89	
x1	y1	x2	y2	x3	y3	x4	y4	
Mean of x	9	9		9		9		
Variance of x	11		11		11		11	
Mean of y		7.5		7.5		7.5		
Variance of y		4.122		4.122		4.122		
Correlation between								
x & y	0.816	0.816		0.816		0.816		
	y1 = 3 + 0.5x1	y2 = 3 + 0.5x2		y3 = 3 + 0.5x3		y4 = 3 + 0.5x4		

BITS Pilani

394



Anscombe's Quartet

- So one would naturally assume that the sets would look roughly the same when shown visually.
- However, once plotted, we realise this is far from the truth.

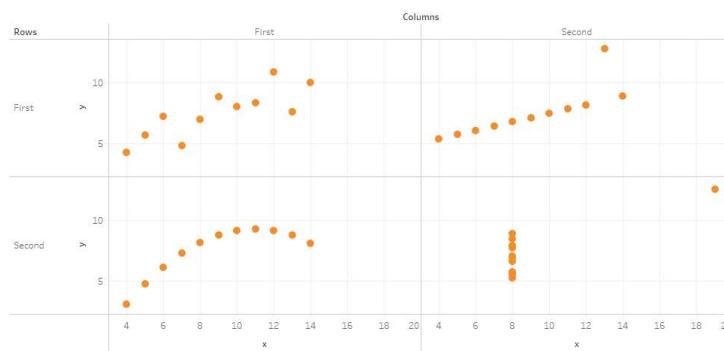
BITS Pilani

395



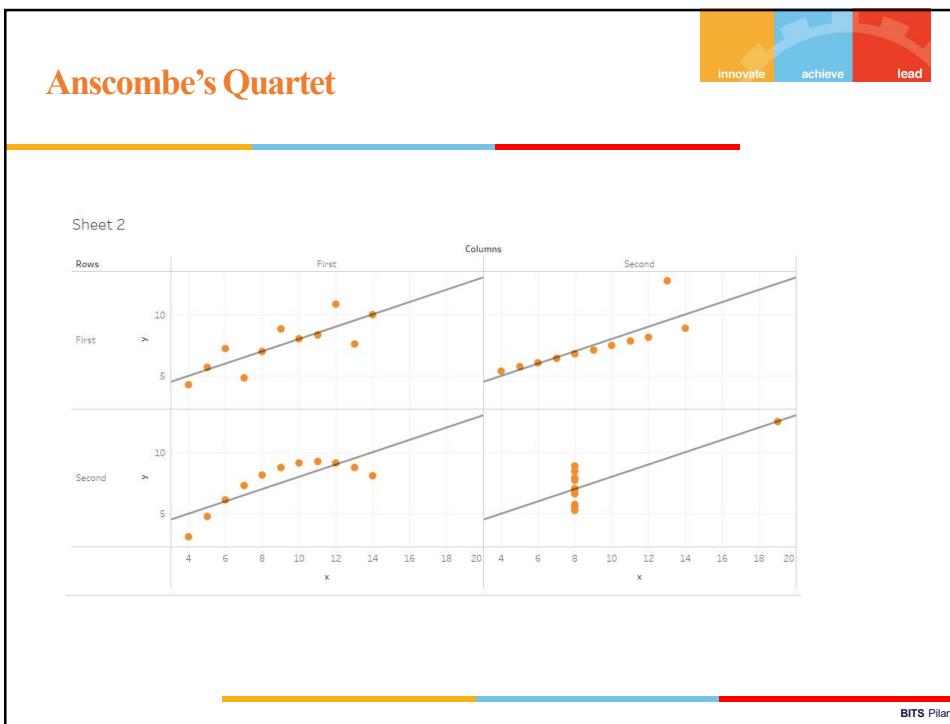
Anscombe's Quartet

Sheet 2

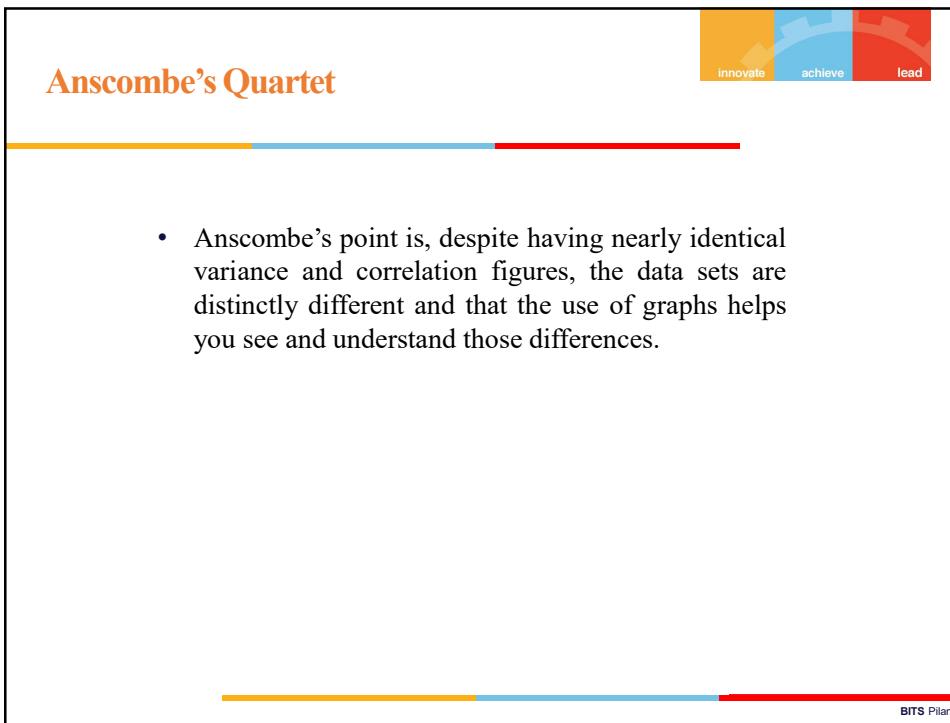


BITS Pilani

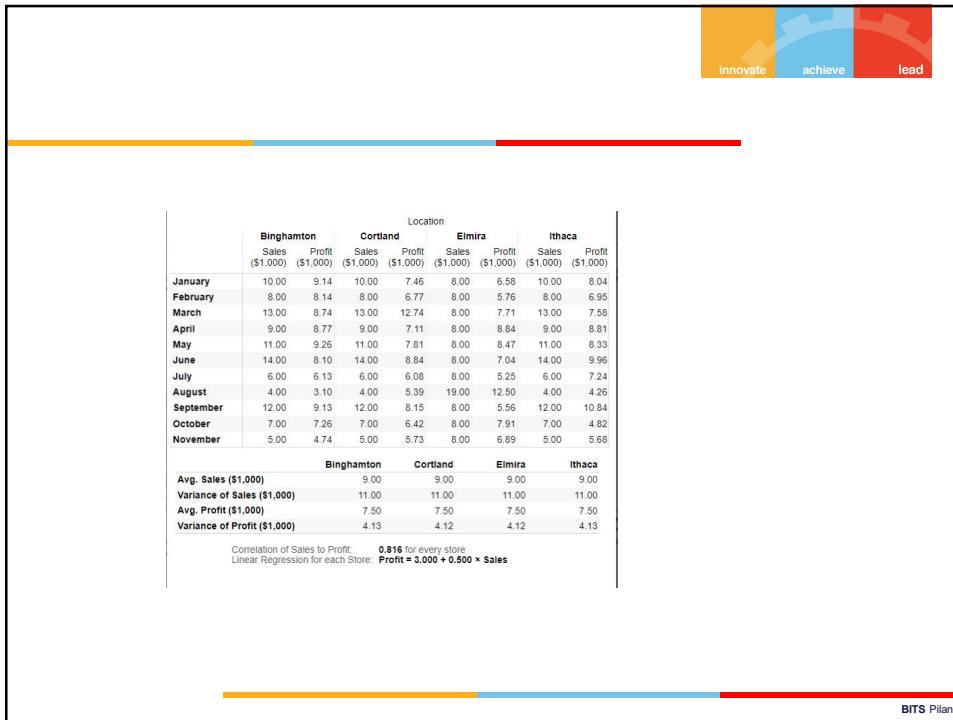
396



397



398



399



**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
WORK INTEGRATED LEARNING PROGRAMMES**

COURSE HANDOUT

Part A: Content Design

Course Title	Data Visualization and Interpretation
Course No(s)	ZG555
Credit Units	5
Course Author	Febin.A.Vahab
Version No	1.0
Date	

Course Description

The course provides an insight on the best practices used in Data Visualization and also illustrates the best tools used to achieve the same

Course Objectives

No	Description
CO1	To introduce key techniques and theory used in visualization, including data models, graphical perception and techniques for visual encoding and interaction.
CO2	Solving various visualization problem using tools like Tableau, Python (Matplotlib)
CO3	Best Practices of Dashboard Design, Designing dashboards meeting the design principles for various requirements

Text Book(s)	
T1	Storytelling with Data, A data visualization guide for business professionals, by Cole Nussbaumer Knaflic; Wiley
T2	Data Visualisation : A Successful Design Process By Andy Kirk
T3	Visualize This: The Flowing Data Guide to Design, Visualization & Statistics, by Nathan Yau, Wiley
T4	Information Dashboard Design: Displaying data for at-a-glance monitoring, Stephen Few, second edition
T5	Tableau Your Data: Fast and Easy Visual Analysis with Tableau Software, by Daniel G Murray
T6	Matplotlib for Python Developers: Effective techniques for data visualization with Python, by Aldrin Yim, Claire Chung and Allen Yu
T7	Hands on Data Visualization with Bokeh: Interactive web plotting for Python using Bokeh, by Kevin Jolly
R1	Mastering Tableau, by David Baldwin



Learning Outcomes:

No	Learning Outcomes
LO1	Concepts and best practices of Data Visualization
LO2	Best practices of Information Dashboard Design
LO3	Data Visualization using Tableau
LO4	Data Visualization using Python (Matplotlib)



Part B: Content Development Plan

Academic Term	First Semester 2019-20
Course Title	Data Visualization and Interpretation
Course No	DSECL ZG555
Credit	5
Content Developer	Febin.A.Vahab

Glossary of Terms

Module	M	Module is a standalone quantum of designed content. A typical course is delivered using a string of modules. M2 means module 2.
Contact Session	CS	Contact Session (CS) stands for a 2 hour long live session with students conducted either in a physical classroom or enabled through technology. In this model of instruction, instructor led sessions will be for 16 CS.
Recorded Lecture	RL	RL stands for Recorded Lecture or Recorded Lesson. It is presented to the student through an online portal. A given RL unfolds as a sequences of video segments interleaved with exercises.
Lab Exercises	LE	Lab exercises associated with various modules
Self-Study	SS	Specific content assigned for self study
Homework	HW	Specific problems/design/lab exercises assigned as homework

Modular Structure

Module Summary

No.	Title of the Module
M1	Data Visualizations and Practices
M2	Effective Dashboard Design
M3	Data Visualization with Tableau
M4	Data Visualization with Python – 1 (Matplotlib)
M5	Data Visualization with Python – 2 (Bokeh, Seaborn)



Detailed Structure

M1: Data Visualizations and Practices Contact Session 1-3

Type	Description/Plan	Reference Text Book/Chapters
CS1	<ul style="list-style-type: none">• Introduction• Exploiting the Digital age• Visualisation as a Discovery tool• Visualisation skills for the masses• The Visualisation methodology• Visualisation design objectives• Exploratory vs. explanatory analysis• Understanding the context for data presentations• 3 minute story• Effective Visuals<ul style="list-style-type: none">◦ Textuals◦ Tabulars◦ Graphicals	T1 Ch 1 and 2, T2 Ch1
	<ul style="list-style-type: none">• Gestalt principles of visual perception• Visual Ordering• Decluttering	T1 Ch 3
CS2	<ul style="list-style-type: none">• Preattentive attributes in text and graphs<ul style="list-style-type: none">◦ Size◦ Color◦ Position• Data Design concepts<ul style="list-style-type: none">◦ Affordances◦ Accessibility◦ Aesthetics	T1 Ch 4
	<ul style="list-style-type: none">• Storytelling• Visualization Design Lessons	T1 Ch 5
CS3	Taxonomy of Data Visualisation Methods <ul style="list-style-type: none">• Comparing Categories of Plots• Dot Plot• Bar Chart• Floating Bar• Histogram• Radial Chart• Glyph Chart	T2 Ch 5
	<ul style="list-style-type: none">• Case Studies<ul style="list-style-type: none">◦ Visualizing Pattern Over time◦ Visualizing Proportions◦ Visualizing Relationships	T3 , Ch 4, 5, 6
<u>SELF STUDY</u>		
<ul style="list-style-type: none">• Data-Driven Documents (D3.js charts)<ul style="list-style-type: none">◦ Exploring visual gallery		



- Simple charts creation
- <https://d3js.org/>
- Explore more D3 charts examples
- Explore Google charts library
- <https://developers.google.com/chart/>
- Good Enough to Great: A Quick Guide for Better Data Visualizations
- <https://www.tableau.com/learn/whitepapers/good-enough-great-quick-guide-better-data-visualizations>

M2: Data Visualization with Tableau

Contact Session 4-7

Type	Description/Plan	Reference
CS4	<ul style="list-style-type: none">• Exploring Tableau<ul style="list-style-type: none">○ User Interface○ Tableau Prep○ Data Connection○ Data Preparation	T5 Ch 1, 2, 3 https://www.tableau.com/learn/training
CS5	<ul style="list-style-type: none">• Visual Analytics<ul style="list-style-type: none">○ Data Analysis○ Visuals	https://www.tableau.com/learn/training T5 Ch 3, 4
CS6	<ul style="list-style-type: none">• Maps	T3 Ch 6
	<ul style="list-style-type: none">• Dashboard and Stories	https://www.tableau.com/learn/training T5 Ch 8
CS7	<ul style="list-style-type: none">• Beyond the Basic Chart Types<ul style="list-style-type: none">○ Bullet graphs○ Pareto charts○ Custom background images	R1 Ch 7
	<ul style="list-style-type: none">• Visualization Best Practices and Dashboard Design	R1 Ch 10

SELF STUDY

- Explore the different types of visuals that can be plotted with Tableau interface

M3: Effective Dashboard Design

Contact Session 8-10

Type	Description/Plan	Reference
CS8	<ul style="list-style-type: none">• Dashboard• Dashboard categorization and typical data• Characteristics of a Well-Designed Dashboard• Key Goals in the Visual Design Process	T4 Ch 2 and 5
	<ul style="list-style-type: none">• Common Mistakes in Dashboard Design	T4 Ch 3
CS9	<ul style="list-style-type: none">• Power of Visual Perception<ul style="list-style-type: none">○ Visually Encoding Data for Rapid Perception○ Applying the Principles of Visual Perception to	T4 Ch 4



	Dashboard Design	
	<ul style="list-style-type: none"> Effective Dashboard Display Media Dashboards design for Usability 	T4 Ch 6 T4 Ch 7
CS10	<ul style="list-style-type: none"> Case Studies <ul style="list-style-type: none"> Sample Sales Dashboard Sample CIO Dashboard Sample Telesales Dashboard Sample Marketing Analysis Dashboard Bringing it all together with Dashboards <ul style="list-style-type: none"> How Dashboard Facilitates Analysis and Understanding How Tableau Improves the Dashboard-building process The right way to build a Dashboard Best Practices for Dashboard building 	T4 Ch 8
		T5 Ch8

SELF STUDY

- Explore any 2 dashboard design tools
<https://dzone.com/articles/20-free-and-open-source-data-visualization-tools>
- Build Your Competitive Edge: 12 Powerful Retail Dashboards
<https://www.tableau.com/learn/whitepapers/powerful-retail-dashboards>
- 10 Best Practices for Building Effective Dashboards
<https://www.tableau.com/learn/whitepapers/10-best-practices-building-effective-dashboards>

M4: Data Visualization with Python – 1 (Matplotlib)

Contact Session 11-14

Type	Description/Plan	Reference
CS11	<ul style="list-style-type: none"> Merits of Matplotlib The Lifecycle of a Plot Pyplot Matplotlib visuals basics 	https://matplotlib.org/tutorials/index.html T6 Ch 1 and Ch2
CS12	<ul style="list-style-type: none"> Plot styles types Visual Decorations 	http://www.labri.fr/perso/nrougier/teaching/matplotlib/ T6 Ch 3
CS13	<ul style="list-style-type: none"> Advanced Matplotlib 	T6 Ch4
CS14	Matplotlib in the real world <ul style="list-style-type: none"> Plotting data from a database Plotting data from a CSV file Plotting extrapolated data using curve fitting Plotting geographical data 	T6 Ch9

SELF STUDY

- Analysis of time series data using matplotlib



- Plotting Univariate Distributions
- Plotting Bivariate Distributions

M5: Data Visualization with Python – 2 (Seaborn and Bokeh)

Contact Session 15-16

Type	Description/Plan	Reference
CS15	<ul style="list-style-type: none">• Seaborn package<ul style="list-style-type: none">○ Seaborn vs Matplotlib○ Data Loading○ Seaborn Basic Plots	https://seaborn.pydata.org/ https://www.datacamp.com/community/tutorials/seaborn-python-tutorial
	<ul style="list-style-type: none">• Statistical plots with Seaborn	https://www.datacamp.com/courses/introduction-to-data-visualization-with-python
CS16	<ul style="list-style-type: none">• Plotting using Glyphs• Plotting with different Data Structures	T7 Ch 1 T7 Ch 2
	<ul style="list-style-type: none">• Using Annotations, Widgets, and Visual Attributes for Visual Enhancement• Building and Hosting Applications Using the Bokeh Server	T7 Ch 4 T7 Ch 5
SELF STUDY		
<ul style="list-style-type: none">• Try out all the statistical plots mentioned in datacamp's tutorial• https://www.datacamp.com/courses/introduction-to-data-visualization-with-python• Try out the Bokeh tutorial• https://www.analyticsvidhya.com/blog/2015/08/interactive-data-visualization-library-python-bokeh/		

Evaluation Scheme:

Legend: EC = Evaluation Component; AN = After Noon Session; FN = Fore Noon Session

No	Name	Type	Duration	Weight	Day, Date, Session, Time
EC-1	Quiz	Online	-	5%	Dec
EC-1	Assignment	Online	-	25%	Dec,Feb/March
EC-2	Mid-Semester Test	Closed Book	1.5 hours	30%	Dec
EC-3	Comprehensive Exam	Open Book	2.5 hours	40%	March

Note: Assignment can be replaced by QUIZ also.

Syllabus for Mid-Semester Test (Closed Book): Topics in Session Nos. 1 to 7

Syllabus for Comprehensive Exam (Open Book): All topics (Session Nos. 1 to 16)



Important links and information:

CANVAS(LMS)

Students are expected to visit the CANVAS course page on a regular basis and stay up to date with the latest announcements and deadlines.

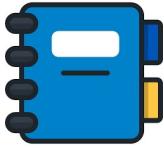
Contact sessions: Students should attend the online lectures as per the schedule provided on CANVAS.

Evaluation Guidelines:

1. EC1 consists of two assignments(Quiz/Assignment). Announcements will be made on the portal, in a timely manner.
2. For Closed Book tests: No books or reference material of any kind will be permitted.
3. For Open Book exams: Use of books and any printed / written reference material (filed or bound) is permitted. However, loose sheets of paper will not be allowed. Use of calculators is permitted in all exams. Laptops/Mobiles of any kind are not allowed. Exchange of any material is not allowed.
4. If a student is unable to appear for the Regular Test/Exam due to genuine exigencies, the student should follow the procedure to apply for the Make-Up Test/Exam which will be made available in CANVAS. The Make-Up Test/Exam will be conducted only at selected exam centres on the dates to be announced later.

It shall be the responsibility of the individual student to be regular in maintaining the self study schedule as given in the course handout, attend the online lectures, and take all the prescribed evaluation components such as Assignment/Quiz, Mid-Semester Test and Comprehensive Exam according to the evaluation scheme provided in the handout.

Agenda



- Visual Design Process
- Dashboard Characteristics
- Key goals in visual design process

1

BITS Pilani

Dashboard

A dashboard is visual display of the most important information needed to achieve one or more objectives, consolidated and arranged on a single screen, so that information can be seen at a glance.

--- Stephen Few

2

BITS Pilani

Dashboard (cont...)

Dashboard Challenge

- To display all required information on a single screen
 - ❖ Clearly and without distraction
 - ❖ In a manner that can be quickly examined and understood

3

BITS Pilani

Dashboard Characteristics

Characteristics of well designed dashboard

- Very well organized
- Summarized
- Customised
- Uses appropriate visual medium

4

BITS Pilani

Dashboard Characteristics (cont...)



Fundamental Principles for Dashboard display media

- Best suited for display of information
- Should be able to convey the same message when restricted to small area

5

BITS Pilani

Visual Design Process



Data-ink ratio

A large share of ink on a graphic should present data-information, the ink changing as the data change. Data-ink is non erasable core of a graphic, the non-redundant ink arranged in response to variation in the numbers represented.

-- Edward Tufte

6

BITS Pilani

5

6

Visual Design Process(cont...)



Data-ink ratio

- = data-ink / total ink used to print the graphic
- = proportion of a graphic's ink devoted to the non-redundant display of information
- = 1 – proportion of a graphic that can be erased without loss of data-information

7

BITS Pilani

Visual Design Process(cont...)



Data-ink ratio

Maximize the data-ink ratio, within reason. Every bit on a graphic requires a reason. And nearly always that reason should be that the ink presents new information.

- Edward Tufte

8

BITS Pilani

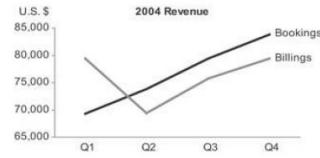
7

8

Visual Design Process(cont...)

Data-ink example

2005 YTD (U.S. \$)		
Region	Units	Bookings
Americas	3,888	229,392
Europe	2,838	167,442
Asia	1,788	105,492
Other	509	30,031
Total	\$9,023	\$532,357
		100%



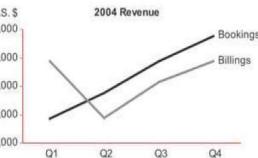
BITS Pilani

9

Visual Design Process(cont...)

Data-ink example

2005 YTD (U.S. \$)		
Region	Units	Bookings
Americas	3,888	229,392
Europe	2,838	167,442
Asia	1,788	105,492
Other	509	30,031
Total	\$9,023	\$532,357
		100%



Note: Here, the non-data ink is highlighted in red

BITS Pilani

10

Visual Design Process(cont...)

Poor Dashboard Design



11

11

Visual Design Process(cont...)

Poor Dashboard Design

- Non-data pixels
 - ❖ Third dimension in pie and bars
 - ❖ Grid lines in bar graph
 - ❖ Background decoration
 - ❖ Background color variation in graphs

BITS Pilani

12

Visual Design Process(cont...)



Fundamental goals in Dashboard Design

- Reduce the non-data pixels
- Enhance the data pixels

13

BITS Pilani

Visual Design Process(cont...)



Reduce the non-data pixels

- Eliminate all unnecessary non-data pixels
- De-emphasize and regularise the non-data pixels that remain

14

BITS Pilani

14

Visual Design Process(cont...)



Eliminate all unnecessary non-data pixels:

- Unnecessary decoration



* You should eliminate graphics that provide nothing but decoration

15

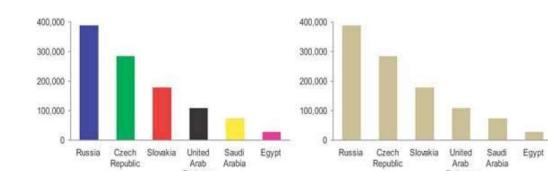
BITS Pilani

Visual Design Process(cont...)



Eliminate all unnecessary non-data pixels

- Unnecessary color variation



16

BITS Pilani

15

16

Visual Design Process(cont...)

innovate achieve lead

Eliminate all unnecessary non-data pixels

Enclosure and borders

of New Customers
Revenue (%)
TOP-10 Revenue (%)
Order Value (%)

more trends
more trends
more trends
more trends

17 BITS Pilani

17

Visual Design Process(cont...)

innovate achieve lead

Eliminate all unnecessary non-data pixels

- Gridlines

* Grid lines in graphs are rarely useful. They are one of the most prevalent forms of distracting non-data pixels found in dashboards

18 BITS Pilani

18

Visual Design Process(cont...)

innovate achieve lead

Eliminate all unnecessary non-data pixels

- Gridlines

Salesperson	Jan	Feb	Mar	Salesperson	Jan	Feb	Mar
Robert Jones	2,834	4,838	6,131	Robert Jones	2,834	4,838	6,131
Mandy Rodriguez	5,890	6,482	8,002	Mandy Rodriguez	5,890	6,482	8,002
Terri Moore	7,398	9,374	11,748	Terri Moore	7,398	9,374	11,748
John Donnelly	9,375	12,387	13,024	John Donnelly	9,375	12,387	13,024
Jennifer Taylor	10,393	12,383	14,197	Jennifer Taylor	10,393	12,383	14,197
Total	\$35,890	\$45,464	\$53,102	Total	\$35,890	\$45,464	\$53,102

19 BITS Pilani

19

Visual Design Process(cont...)

innovate achieve lead

Eliminate all unnecessary non-data pixels

- Fill colors

Sell		Alerts ▾	Result ▾	Alert Spec ▾	Last Update ▾
Metric					
QTD Sales (\$MM)		2	● \$ 153.0	\$ 166.0	1/21/02
QTD Average Daily Order Rate (ADOR) (\$MM)		1	● \$ 16.1	\$ 11.9	1/21/02
Previous Day's Orders (\$MM)		0	● \$ 26.2	\$ 11.9	1/21/02
QTD % e-Orders		3	● 53.0%	59.0%	1/21/02
Current Qtr Price vs Target (\$/lb)		6	● \$ 1.27	\$ 1.20	1/21/02

* Fill colors should be used to delineate rows in a table when this is necessary to help viewers' eyes track across the rows

20 BITS Pilani

20

Visual Design Process(cont...)

Eliminate all unnecessary non-data pixels

- 3D

*3D should always be avoided when the added dimension of depth doesn't represent actual data

21 BITS Pilani

21

Visual Design Process(cont...)

De-emphasize and regularise the non-data pixels that remain

- Axis lines

22 BITS Pilani

22

Visual Design Process(cont...)

De-emphasize and regularise the non-data pixels that remain

- Lines, borders or fill colors

*Lines can be used effectively to delineate adjacent sections of the display from one another, but the weight of these lines can be kept to a minimum

23 BITS Pilani

23

Visual Design Process(cont...)

De-emphasize and regularise the non-data pixels that remain

- Grid lines (if necessary)

24 BITS Pilani

Visual Design Process(cont...)

innovate achieve lead

De-emphasize and regularise the non-data pixels that remain

- Grid lines and fill colors

Product	Jan	Feb	Mar	Q1 Total	Apr	May	Jun	Q2 Total	YTD Total
Product A	93,993	84,773	88,833	267,599	95,838	93,874	83,994	273,706	541,305
Product B	87,413	78,839	82,615	248,867	89,129	87,303	78,114	254,547	503,414
Product C	90,036	81,204	85,093	256,333	91,803	89,922	80,458	262,183	510,516
Product D	92,737	83,640	87,646	264,023	94,557	92,620	82,872	270,048	534,072
Product E	83,733	75,520	79,137	238,390	85,377	83,627	74,826	243,830	482,220
Total	447,913	403,976	423,323	1,275,212	456,705	447,346	400,264	1,304,314	2,579,526

Product	Jan	Feb	Mar	Q1 Total	Apr	May	Jun	Q2 Total	YTD Total
Product A	93,993	84,773	88,833	267,599	95,838	93,874	83,994	273,706	541,305
Product B	87,413	78,839	82,615	248,867	89,129	87,303	78,114	254,547	503,414
Product C	90,036	81,204	85,093	256,333	91,803	89,922	80,458	262,183	510,516
Product D	92,737	83,640	87,646	264,023	94,557	92,620	82,872	270,048	534,072
Product E	83,733	75,520	79,137	238,390	85,377	83,627	74,826	243,830	482,220
Total	447,913	403,976	423,323	1,275,212	456,705	447,346	400,264	1,304,314	2,579,526

*Grid lines and fill colors can be used in tables to clearly distinguish some columns from others, but this should be done in the muted manner seen below rather than the heavy handed manner seen above

25

BITS Pilani

25

Visual Design Process(cont...)

innovate achieve lead

De-emphasize and regularise the non-data pixels that remain

- Buttons and user controls



26

BITS Pilani

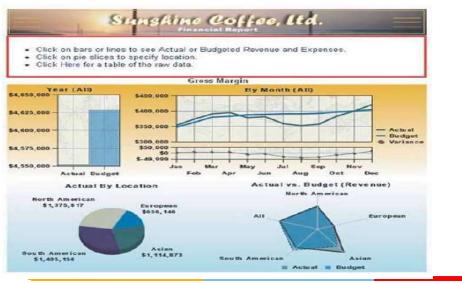
26

Visual Design Process(cont...)

innovate achieve lead

De-emphasize and regularise the non-data pixels that remain

- Instructions / text



27

BITS Pilani

27

Visual Design Process(cont...)

innovate achieve lead

Enhance data pixels

- Eliminate all unnecessary data pixels
- Highlight the most important data pixels that remain

28

BITS Pilani

28

Visual Design Process(cont...)



Eliminate all unnecessary data pixels

- Remove less relevant data
- Used condensed, summarised data

29

BITS Pilani

Visual Design Process(cont...)



Eliminate all unnecessary data pixels

- Use multi-foci displays



*These three time-series graphs displaying public transportation rider statistics contain three levels of detail: daily for the current month, monthly for the current year, and yearly for the last 10 years.

30

BITS Pilani

29

30

Visual Design Process(cont...)



Categories of the information

- Information that is always important
 - Key business measures
- Information that is only important at the moment
 - A measure that has fallen far behind its target

Both requires different ways of highlighting

31

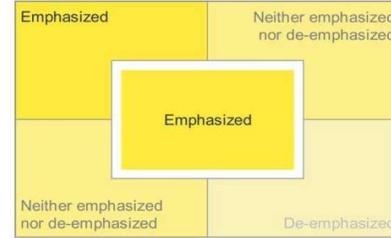
BITS Pilani

Visual Design Process(cont...)



Highlight the most important data pixels that remain

- Most important information should be placed at top-left and center.



*Different degrees of visual emphasis are associated with different regions of a dashboard

32

BITS Pilani

31

32

Visual Design Process(cont...)

Highlight the most important data pixels that remain

- Waste of critical retail estate



*The most valuable real estate on this dashboard is dedicated to a company logo and meaningless decoration

33

BITS Pilani

Visual Design Process(cont...)

Approaches to static and dynamic highlighting of data

- ❑ Use Visual attributes that are greater than the norm
 - ❖ brighter or darker colors
- ❑ Use Visual attributes that simply contrast with the norm
 - ❖ Different color than the ones used in visual

34

BITS Pilani

33

34

Visual Design Process(cont...)

Use Visual attributes that are greater than the norm

Visual attribute	Useful expressions	Illustrations
Color intensity	A darker or more fully saturated version of any hue is naturally perceived as greater than a lighter or less-saturated version.	
Size	Bigger things clearly stand out as more important than smaller things.	
Line width	Thicker lines stand out as more important than thinner lines.	

35

BITS Pilani

35

Visual Design Process(cont...)

Use Visual attributes that simply contrast with the norm

Visual attribute	Useful expressions	Illustrations
Hue	Any hue that is distinct from the norm will stand out. ¹	
Orientation	Anything oriented differently than the norm will stand out.	
Enclosure	Anything enclosed by borders or surrounded by a fill color will stand out if different from the norm.	
Added marks	Anything with something distinctly added to it or adjacent to it will stand out.	

36

BITS Pilani

36

Visual Design Process(cont...)

Example : Use of added marks

Metric	Actual	Variance
Revenue	\$913,394	+\$136,806
Profit	\$193,865	-\$73,055
Avg Order Size	\$5,766	-\$297
On Time Delivery	104%	+4%
New Customers	247	-62
Cust Satisfaction	4.73 / 5	+0.23

*Simple symbols can be used along with varying color intensities to dynamically highlight data

37

Agenda



- ❑ Dashboard Design for Usability

38

Dashboard Design for Usability

Usability aspects



- ❑ Information organization to support its meaning and use
- ❑ Consistency for quick and accurate interpretation
- ❑ Aesthetically pleasing viewing experience
- ❑ Design for a use as a launch pad
- ❑ Usability testing

39

Dashboard Design for Usability (cont...)

Information organization to support its meaning & use

Key points

- ❑ Grouping according to business functions, entities, roles
- ❑ Co-locating of objects belonging to same group
- ❑ Group delineation by least visible means
- ❑ Enhance meaningful comparison
- ❑ Discourage meaningless comparison

40

Dashboard Design for Usability (cont...)



Grouping according to business functions, entities, roles

- Information can be organized by
 - ❖ Business functions – like ordering, shipping, budgeting etc
 - ❖ Entities – like various departments , projects
 - ❖ Uses of data – like comparing sales metrics, cost metrics

- Learn
 - ❖ How information will be used
 - ❖ How the pieces ought to be arranged to best serve

41

BITS Pilani

Dashboard Design for Usability (cont...)



Co-locating of objects belonging to same group

- Use principle of similarity
- Place similar / related items closer to each other
- Still delineate them in simple manner

42

BITS Pilani

41

42

Dashboard Design for Usability (cont...)



Group delineation by least visible means

- Use white space effectively
- Otherwise, if space is issue, use subtle borders

43

BITS Pilani

Dashboard Design for Usability (cont...)



Group delineation by least visible means

Example – use white space

Product	Units Sold	Actual Revenue	Region	Units Sold	Actual Revenue
Shirts	938	187,600	North	2,263	133,066
Blouses	1,093	114,765	South	1,920	112,905
Pants	3,882	62,112	East	1,303	76,614
Skirts	873	36,666	West	754	44,355
Dresses	72	2,088	Canada	618	36,291
Total	6,858	\$403,231	Total	6,858	\$403,231

Channel	Units Sold	Actual Revenue	Warehouse	Units Sold	Actual Revenue
Direct	2,057	120,969	Virginia	2,537	149,195
Distributor	1,921	119,903	California	1,920	112,905
Reseller	1,783	104,840	Texas	1,372	80,646
OEM	1,097	64,519	Calgary	1,029	60,485
Total	6,858	\$403,231	Total	6,858	\$403,231

44

BITS Pilani

43

44

Dashboard Design for Usability (cont...)



Group delineation by least visible means

Example – use borders

Product	Units Sold		Actual Revenue		Region	Units Sold		Actual Revenue	
	Units	Sold	Revenue	Region		Units	Sold	Revenue	
Shirts	938	187,600	North	2,263	133,066				
Blouses	1,093	114,765	South	1,920	112,905				
Pants	3,882	62,112	East	1,303	76,614				
Skirts	873	36,666	West	754	44,355				
Dresses	72	2,088	Canada	618	36,291				
Total	6,858	\$403,231	Total	6,858	\$403,231				

Channel	Units Sold		Actual Revenue		Warehouse	Units Sold		Actual Revenue	
	Units	Sold	Revenue	Warehouse		Units	Sold	Revenue	
Direct	2,057	120,969	Virginia	2,537	149,195				
Distributor	1,921	119,903	California	1,920	112,905				
Reseller	1,783	104,840	Texas	1,372	80,646				
OEM	1,097	64,519	Calgary	1,029	60,485				
Total	6,858	\$403,231	Total	6,858	\$403,231				

45

BITS Pilani

Dashboard Design for Usability (cont...)



Enhance meaningful comparison

By

- ❖ Combining items in single table or graph
- ❖ Placing items close to each other
- ❖ Using different colors for different groups
- ❖ Use ratios, percentages i.e. actual values

46

BITS Pilani

45

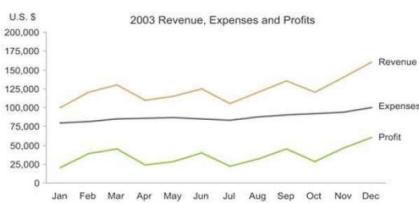
46

Dashboard Design for Usability (cont...)



Enhance meaningful comparison

Example – Multiple measures in one graph



47

BITS Pilani

Dashboard Design for Usability (cont...)



Enhance meaningful comparison

Example – Using table for all measures

Product	Units Sold	Actual Revenue	% of Total	Forecast Revenue	% of Fst
Product A	938	187,600	47%	175,000	107%
Product B	1,093	114,765	28%	130,000	88%
Product C	3,882	62,112	15%	50,000	124%
Product D	873	36,666	9%	40,000	92%
Product E	72	2,088	1%	50,000	4%
Total	6,858	\$403,231	100%	\$445,000	91%

48

BITS Pilani

47

48

Dashboard Design for Usability (cont...)



Discourage meaningless comparison

- By
 - ❖ Spatially separating items
 - ❖ Using different colors

49

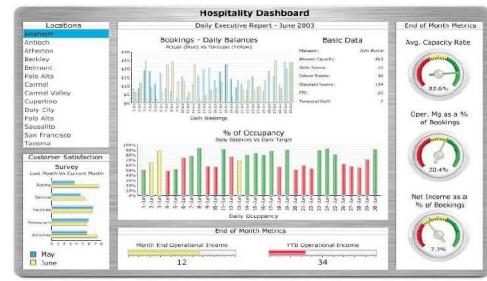
BITS Pilani

Dashboard Design for Usability (cont...)



Discourage meaningless comparison

Example – useless comparisons



*This dashboard inadvertently encourages meaningless comparison

50

BITS Pilani

Dashboard Design for Usability (cont...)



Consistency for quick and accurate interpretation

- Small difference triggers alarm
- Maintain consistency in
 - ❖ Visual appearances of display media
 - ❖ Choice of display media
- Use same mediums for same kind of comparisons
- Don't add variety just for sake of it

51

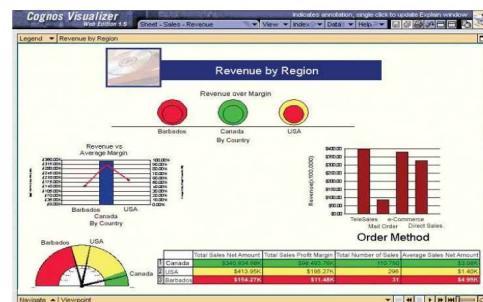
BITS Pilani

Dashboard Design for Usability (cont...)



Aesthetically pleasing viewing experience

- Example – ugly dashboard



52

BITS Pilani

51

52

Dashboard Design for Usability (cont...)



Aesthetically pleasing viewing experience

- Choose colors appropriately
 - ❖ Minimum bright colors to be used
 - ❖ Use less saturated colors
 - ❖ Use pale background colors
- Choose high resolution for clarity
 - ❖ Images with poor resolution are hard to read
- Choose right text
 - ❖ Use most legible font

53

BITS Pilani

Dashboard Design for Usability (cont...)



Design for use as a Launch Pad

- Design for interaction
 - ❖ Drill to details
 - ❖ Slicing and dicing of data
- Allow viewer to launch details section by clicking on data itself
- Use consistent launch actions

54

BITS Pilani

53

54

Dashboard Design for Usability (cont...)



Usability testing

- Difficult to get away with predetermined notions of user
- Present users with single prototype of most effective design
- Don't spoil users with the choices of design
- Present it to the actual users with real data for testing
- Take feedback and iterate over the design process again

55

BITS Pilani

Recap



Visual Design Process

- ✓ Characteristics of Dashboard
- ✓ Key goals of design process
 - ❖ Reduce the non-data pixels
 - ❖ Enhance the data pixels

56

56

BITS Pilani

55

Recap



i Dashboard Design for Usability

- ✓ Organization of information to support meaning
- ✓ Consistency for quick & accurate interpretation
- ✓ Aesthetically pleasing viewing experience
- ✓ Design for use as a launch pad
- ✓ Usability testing

57

BITS Pilani

References

Information Dashboard Design
Stephen Few

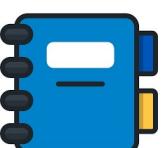
- Chapter 5 : Eloquence Through Simplicity
- Chapter 7 : Designing Dashboard for Usability



58

BITS Pilani

Agenda



- Dashboards by examples
 - Sales Dashboard
 - CIO dashboard
 - Telesales, and
 - Marketing Analysis dashboard
- Best Practices for Tableau Dashboard

BITS Pilani

Sample Sales Dashboard

Critical Metrics

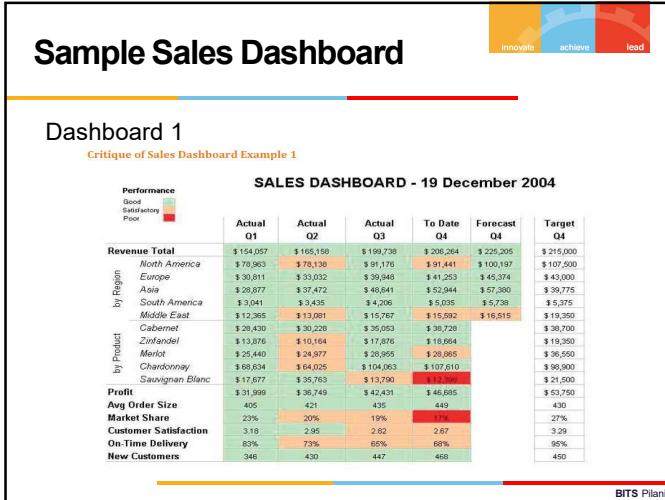
- Sales revenue
- Sales revenue in pipeline
- Profit
- Customer satisfaction rating
- Top 10 customers
- Market Share



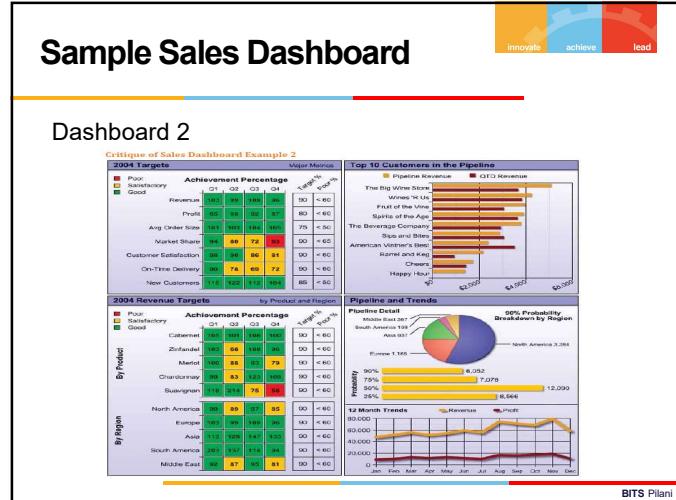
BITS Pilani

59

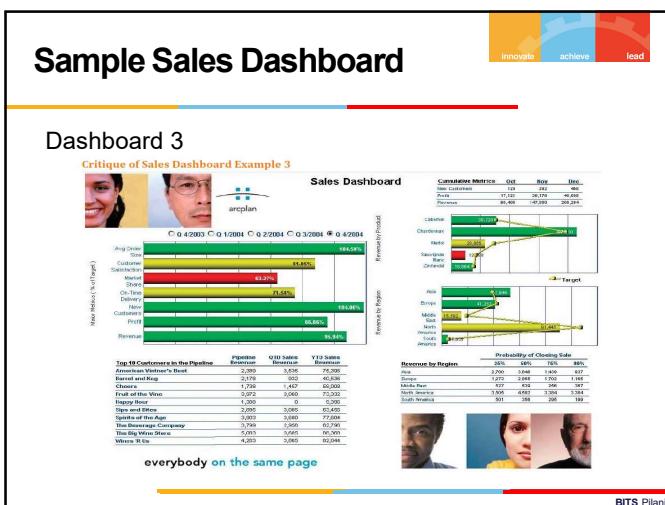
60



61



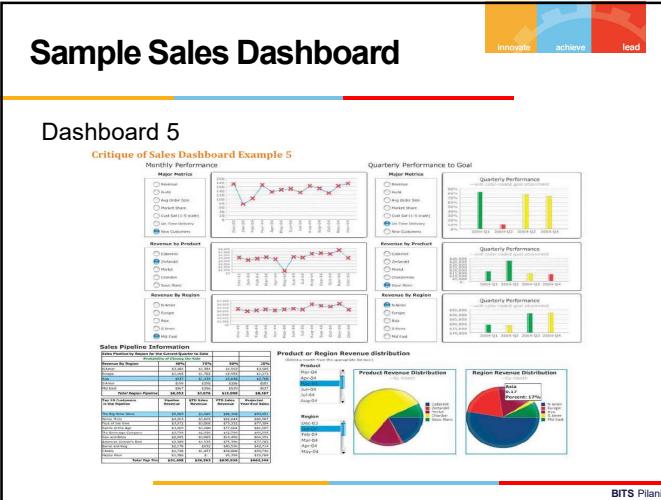
62



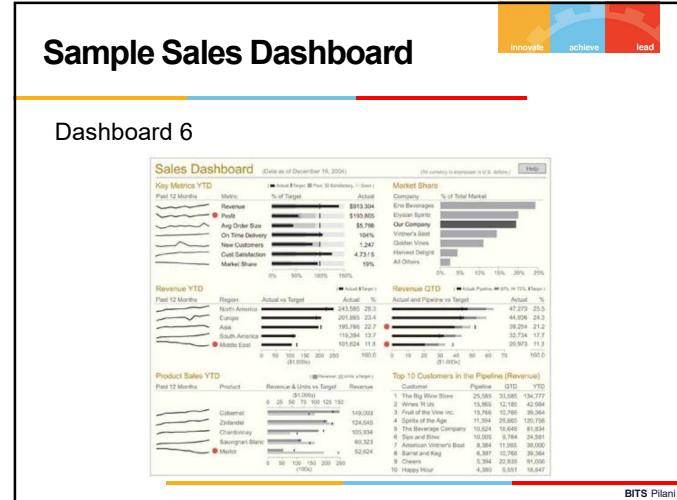
63



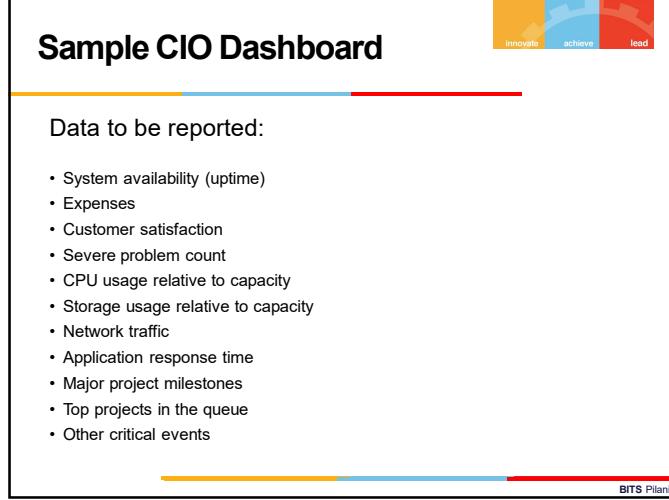
64



65



66



67



68

Sample Telesales Dashboard

Data to be reported:

- Call wait time
- Call duration
- Abandoned calls (that is, callers who got tired of waiting and hung up)
- Call volume
- Order volume
- Sales representative utilization (representatives online compared to the number available)

BITS Pilani

69

Sample Telesales Dashboard

BITS Pilani

70

Sample Marketing Dashboard

Data to be reported (based on website activities):

- Number of visitors (daily, monthly, and yearly)
- Number of orders
- Number of registered visitors
- Number of times individual products were viewed on the site
- Occasions when products that were displayed on the same page were rarely purchased together
- Occasions when products that were not displayed on the same page were purchased together
- Referrals from other web sites that have resulted in the most visits

BITS Pilani

71

Sample Marketing Dashboard

BITS Pilani

72

Best Practices for Dashboard building in Tableau

innovate achieve lead

- Size the dashboard to fit the in the worst-case available space.
- Employ 4-pane dashboard designs.
- Use Actions to filter instead of Quick Filters.
- Build cascading dashboard designs to improve load speed.
- Limit the use of color to one primary color scheme.
- Use small instructions near the work to make navigation obvious.
- Filter information presented in crosstabs to provide relevant details on-demand.
- Remove all non-data ink.
- Avoid One Size Fits All dashboards.

BITS Pilani

73

Best Practices for Dashboard building in Tableau

innovate achieve lead

Size the dashboard to fit the in the worst-case available space.

- Determining the pixel height and width of the worst-case dashboard consumption environment.
- Tableau provides defaults for the typical sizes you will need or allows you to define a custom size.

BITS Pilani

74

Best Practices for Dashboard building in Tableau

innovate achieve lead

Employ 4-pane dashboard designs.

The dashboard consists of four panels arranged in a 2x2 grid. The top-left panel shows a bar chart of Product Sales & Profitability across various categories. The top-right panel is a map titled 'State Sales by Product Category 2011' with bubbles representing sales volume and color-coded by category. The bottom-left panel is a line chart titled 'Sales Trend by Product Category 2011' showing quarterly sales trends from Q1 to Q4. The bottom-right panel is a scatter plot titled 'Profit vs Shipping Cost 2011' with points colored by category and sized by profit.

BITS Pilani

75

Best Practices for Dashboard building in Tableau

innovate achieve lead

Use small instructions near the work to make navigation obvious.

The dashboard features a tooltip example in the top-left corner, which highlights a specific data point on a bar chart with detailed information about the category, sales, profit, and profit percentage. To the right is a 'Read Me' section containing a 'Data Sources' box, a 'Format' box with a formula for profit ratio, and a note about validated data. A 'Phone' and 'Email' contact information is also provided.

BITS Pilani

76

Best Practices for Dashboard building in Tableau

innovate achieve lead

Filter information presented in crosstabs to provide relevant details on-demand

Better Use of Crosstab

Total Sales by Market

Market	Sales
North	271,364
East	193,876
South	193,876

Total Sales by State

State	Sales
California	271,364
Florida	193,876

Sales Trend

Metric: All, Region: All

Period	Coffee	Espresso	Latte	Tea	Grand Total
2010	210,028	221,346	287,214	67,773	698,951
2011	210,028	221,346	287,214	67,773	698,951

Detailed Metrics: Region: All

Metric	Value
Sales	271,364
COGS	161,056
Profit	110,308
Margin	37.72%
Profit Margin	37.72%
Budget Profit	110,308
Total Profit	110,308

Metric: East & West, Region: New York & California

Period	Coffee	Espresso	Latte	Tea	Grand Total
2010	42,345	53,329	67,719	12,254	177,748
2011	42,345	53,329	67,719	12,254	177,748

Metric: East & West, Region: New York & California

Metric	Value
Sales	42,345
COGS	29,652
Profit	12,253
Margin	25.48%
Profit Margin	25.48%
Budget Profit	12,253
Total Profit	12,253

Metric: East & West, Region: New York & California

Metric	Value
Sales	7,932
COGS	6,946
Profit	7,932
Margin	10.00%
Profit Margin	10.00%
Budget Profit	7,932
Total Profit	7,932

BITs Pilani

77

innovate achieve lead

References

Information Dashboard Design

Stephen Few

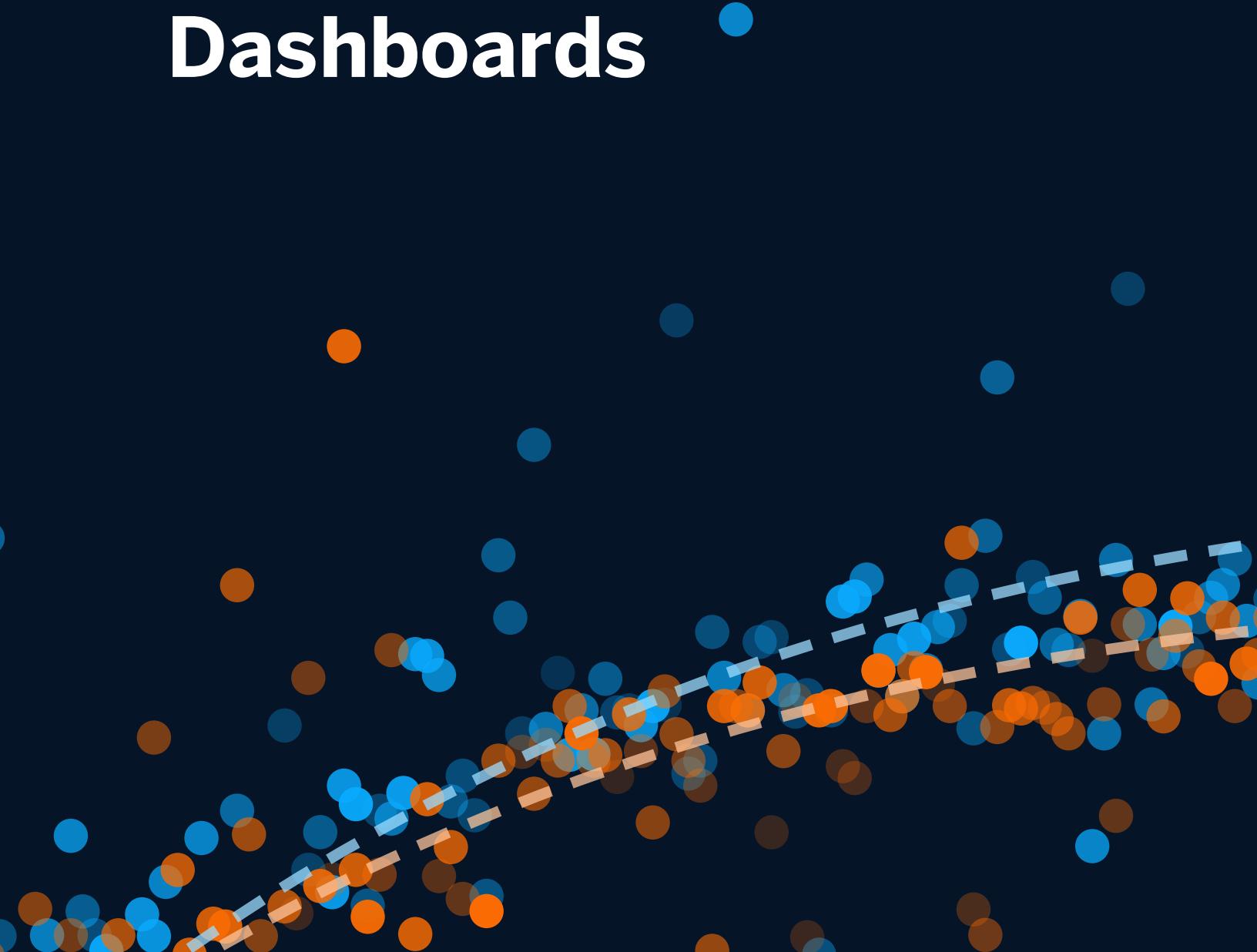
Chapter 8 : Putting It All Together

BITs Pilani

78



10 Best Practices for Building Effective Dashboards



A well-designed dashboard is a powerful launch point for data-driven conversations. Armed with the same collection of information, your business makes faster decisions based on a single source of truth.

A great dashboard's message and metrics are clear, its color enhances meaning, and it delivers the most relevant information to your audience. So how do you build dashboards for your organization that live up to this promise?

It really comes down to three things: thoughtful planning, informed design, and a critical eye for refining your dashboard.

Contents

Thoughtful Planning

1. Know your audience	3
2. Consider display size	4
3. Plan for fast load times	5

Informed Design

4. Leverage the sweet spot.....	6
5. Limit the number of views and colors	7
6. Add interactivity to encourage exploration.....	9
7. Format from largest to smallest	10

Refining Your Dashboard

8. Leverage tooltips, the story within your story	12
9. Eliminate clutter	14
10. Test your dashboard for usability.....	15

About Tableau	16
---------------------	----

Additional Resources.....	16
---------------------------	----

Thoughtful Planning

1. Know your audience

The best dashboards are built with their intended audience in mind. This doesn't happen by accident. Ask yourself, who am I designing this for? Is it a busy salesperson with 15 seconds to spare for key performance indicators, or is it a team reviewing quarterly dashboards over several hours?

It's also important to know your audience's level of expertise with the subject matter and data. For example, a beginner might need more action-oriented labelling for filters or parameters than an advanced user. If you don't know a lot about the audience, start by asking questions about their priorities and how they consume data to inform the best way to present the data. Remember that you can always create more dashboards. The best approach is to start simple.

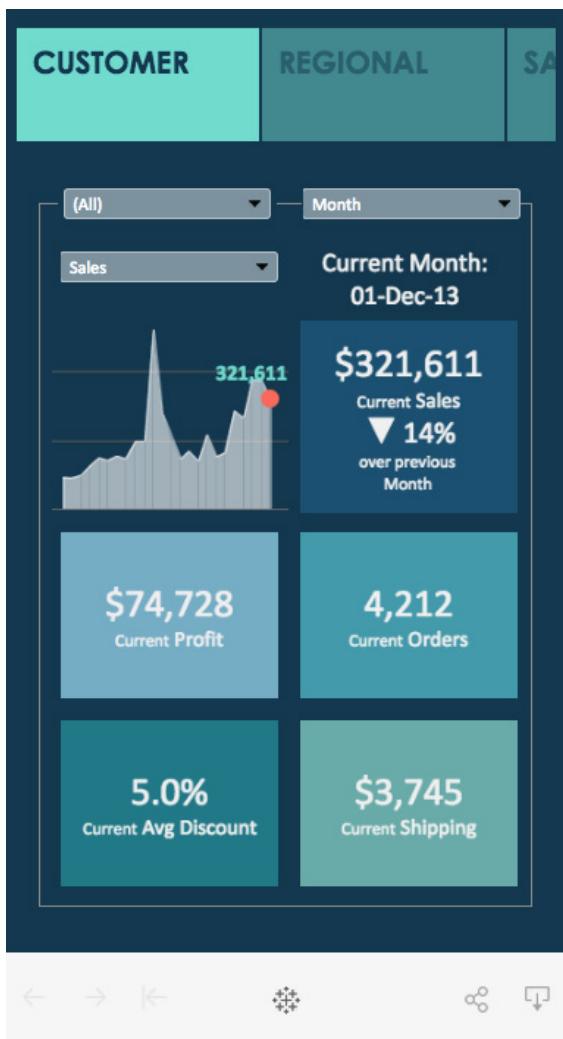


A financial team tasked with reviewing high-level information about international tourism income could easily use this dashboard.

2. Consider display size

If you build a dashboard solely for a desktop monitor, but your viewers primarily use their mobile phones to consume data, you're probably not going to have a very satisfied audience. Do some research up front to understand how your audience's habits might inform your dashboard design.

Surface the most important KPIs: Remember that your audience won't always be able to drill down on a small screen, so when you design for mobile phones or tablets, show only the most important metrics. In practice, this means creating dashboards with elements that are easy to click and have limited, intentional interactivity.



In this dashboard, there are no more than three interactions. This simplicity reduces confusion and helps with overall user experience on mobile.

Stack content vertically for phone screens: Most people use their phones in portrait mode. Unless you need to show a wide map view or timeline, prioritize optimizing your dashboard vertically for phones.

In Tableau, phone layouts are automatically generated whenever you create a new dashboard, arranging the dashboard's contents algorithmically in a phone-friendly manner. You can also choose the “Edit layout myself” option to manually add and arrange items to reflect changes to the Default dashboard. To see how your dashboards appear on different devices, review and add device layouts with [Device Preview](#).

3. Plan for fast load times

Even the most beautiful dashboard won't have an impact if it takes too long to load. Sometimes long load times are caused by your data, your dashboard, or a combination of the two.

Some of the most critical decisions you make as an author begin in the data preparation stage before you even create your first view. Wherever possible, especially on production views, perform calculations in the database to reduce overhead. Aggregate calculations are great for calculated fields in Tableau, but perform row-level calculations in the database when you can.

Determine if you need to limit the amount of data surfaced in your dashboard, either by [creating filters on a data source](#) or creating an extract. [Extracts](#) are typically much faster than a live data source, and are especially great for prototyping. Keep in mind that extracts are not always the long-term solution. When querying against constantly-refreshing data, a live connection often makes more sense when operationalizing the view.

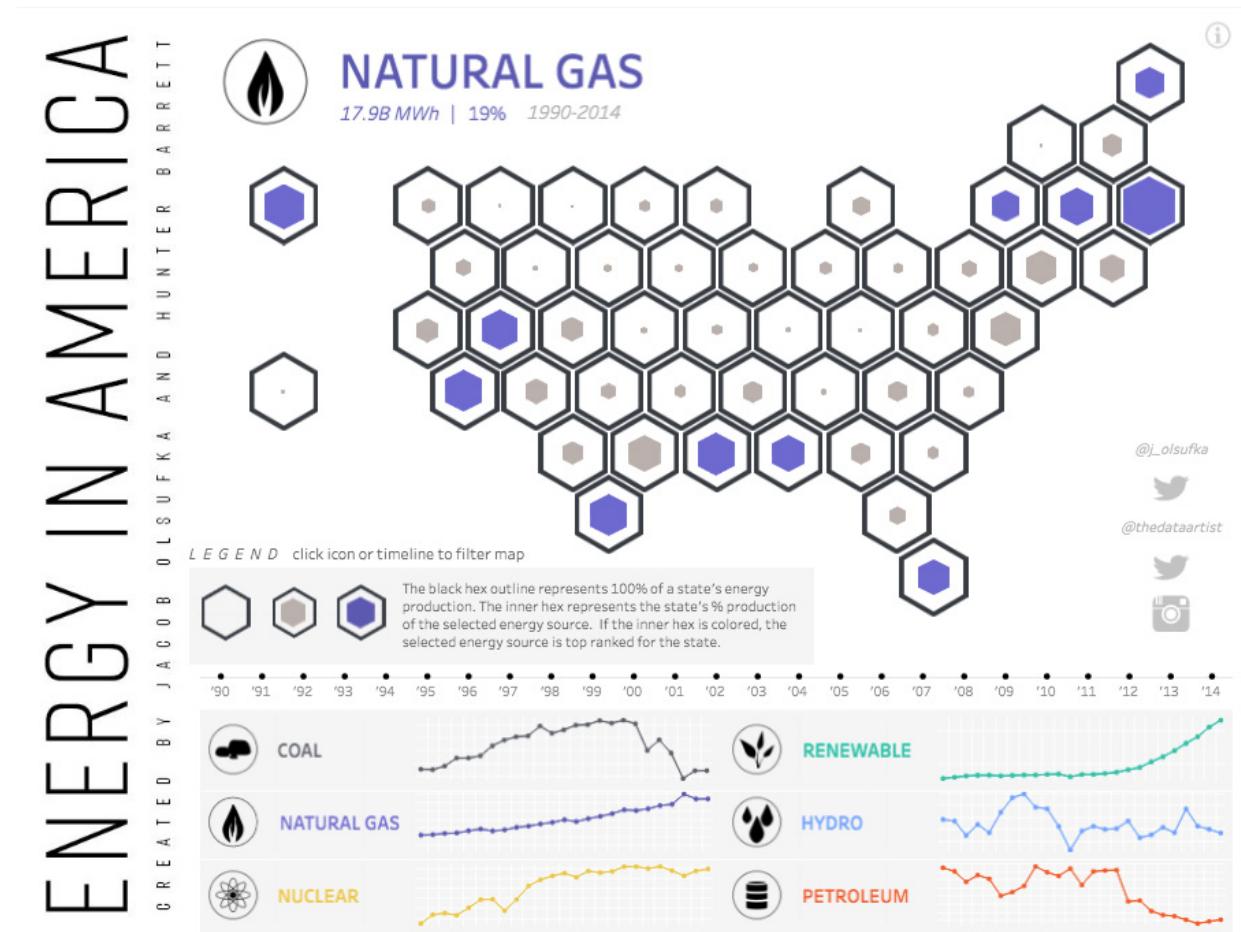
For more optimization tips, learn how to [Optimize Workbook Performance](#) in the Online Help. Knowing [Tableau's Order of Operations](#) may also help you shave time off your load times.

Informed Design

4. Leverage the sweetspot

Always consider how your audience will “read” your dashboard. Your dashboard should have a sensible “flow” and a logical layout of different pieces of information.

As you design your dashboard, consider the parts that form logical groups and use your design to group them together. Shading, lines, white space, and color are all useful ways to make the connections.



Jacob Olsufka grouped the hexes closely so we easily perceive the map of the United States. He also gave the legend and supporting text a common background, and grouped the social icons using proximity.

Most viewers scan web content starting at the top left of a web page. Once you know your dashboard’s main purpose, be sure to place your most important view so that it occupies or spans the upper-left corner of your dashboard. In the dashboard above, the author decided that the header and the map view hold the key messages.

5. Limit the number of views and colors

It's easy to get excited and want to cram your dashboard with every relevant view. But if you add too many, you'll sacrifice the big picture. In general, stick to two or three views. If you find that the scope needs to grow beyond that, create more dashboards or use a [story](#)—a sequence of visualizations that work together to guide the viewer through information.

Just like you can have too many views, you can also have too many colors. Color used correctly enhances analysis. Too many colors creates visual overload for your audience, slowing analysis and sometimes preventing it.

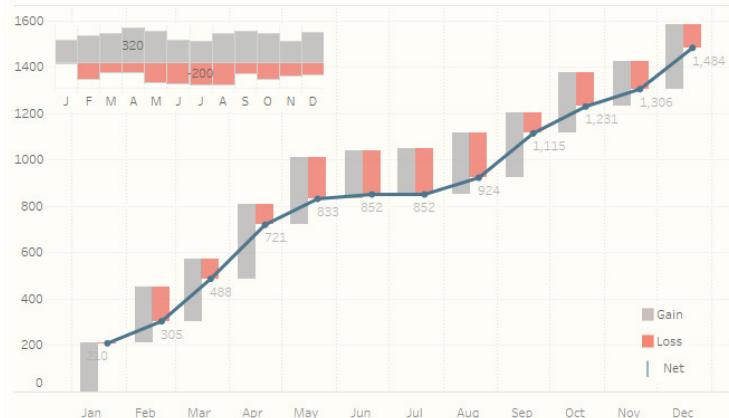
Here's a before-and-after for a dashboard about subscriber churn.

This “before” version uses harsh, more saturated colors and inconsistent shading, making it much harder for the viewer to see the relationship between the charts.



Subscriber Churn Analysis

Subscriber activity - All



Net subscriber activity by division



Details

		Gained	Lost	Net	Runr
West	January	80	0	80	80
	February	80	-15	65	65
	March	90	-30	60	60
	April	120	-25	95	95
	May	100	-50	50	50
	June	119	-77	42	42
	July	75	-45	30	30
	August	119	-77	42	42
	September	90	-30	60	60
	October	80	-15	65	65
	November	80	-20	60	60
	December	90	-30	60	60
Total		1,123	-414	709	
Central	January	60	0	60	60
	February	85	-45	40	40
	March	80	-27	53	53
	April	90	-17	73	73
	May	120	-33	87	87
	June	45	-80	-35	-35
	July	75	-45	30	30
	August	45	-80	-35	-35
	September	80	-27	53	53
	October	85	-45	40	40
	November	60	-35	25	25
	December	80	-27	53	53
Total		905	-461	444	
East	January	70	0	70	70
	February	80	-90	-10	-10
	March	100	-30	70	70
	April	110	-45	65	65
	May	70	-95	-25	-25
	June	45	-33	12	12
	July	50	-110	-60	-60
	August	99	-34	65	65
	September	112	-34	78	78
	October	99	-88	11	11
	November	55	-65	-10	-10
	December	110	-45	65	65
Total		1,000	-669	331	
Grand Total		3,028	-1,544	1,484	

This revised version of the same dashboard has a modern design with minimal colors, creating a gentle formatting.

Subscriber Churn, [The Big Book of Dashboards](#)

As addicting as it is to customize dashboards, avoid adding unnecessary objects that get in the way of your dashboard's capacity to quickly inform your audience.

6. Add interactivity to encourage exploration

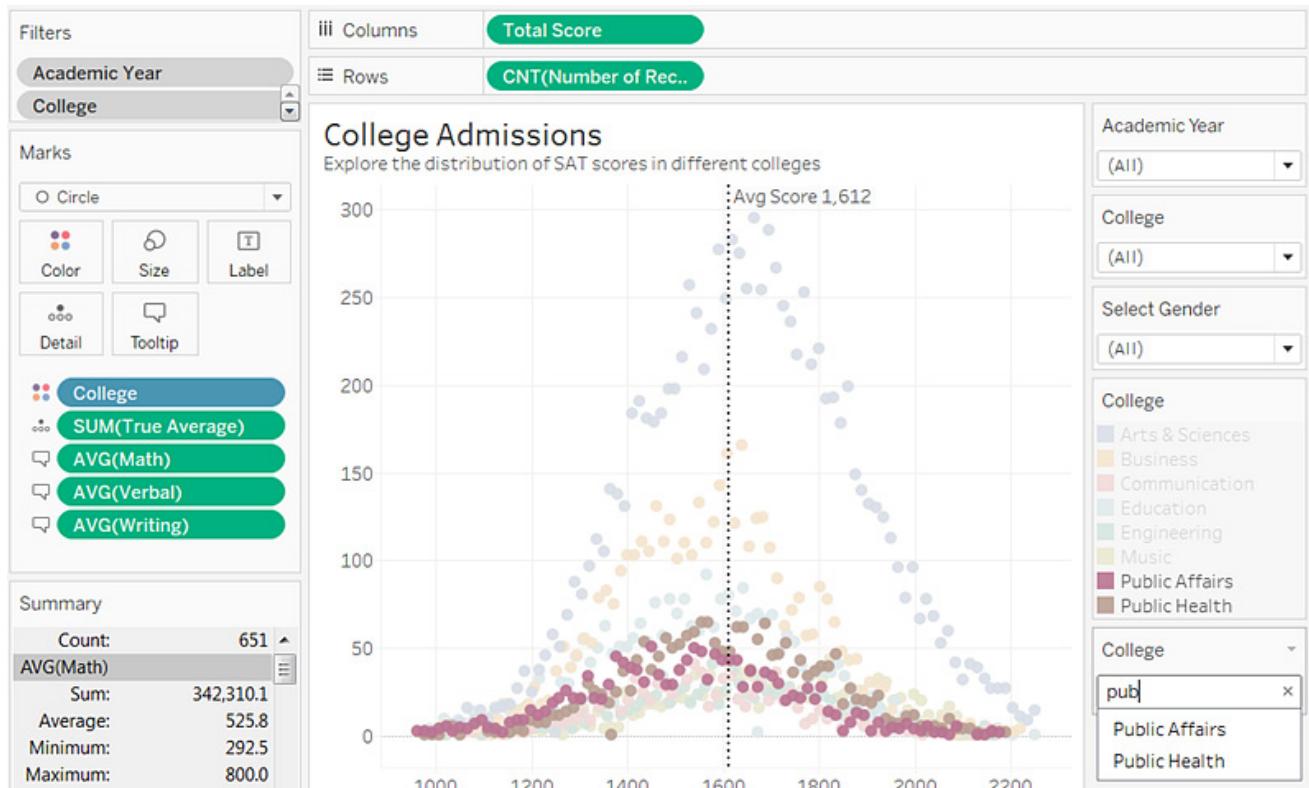
The power of dashboards lies in the author's ability to queue up specific views for side-by-side analysis. Filters super-charge that analysis and engage your audience. For example, you can have one view—your most important one—act as a filter on the other views in the dashboard. To do this, select Use as Filter from the view's shortcut menu.



This dashboard uses the area chart as a filter. When you click on the area chart, the bar chart below filters to only show the data classified as "Shipped Early," allowing the audience to dig into the data that is relevant to them.

You can also display filter cards for different types of data. For example, show filters as multi-select checkboxes, single select radio buttons, drop-down lists, etc. You can include a search box and edit the title of your filter to give your viewers clear instructions for interacting with the data.

Highlight actions are another powerful feature you can leverage, where a selection in one view highlights related data in the other views. For more advanced scenarios, you can use [set actions](#) or [parameter actions](#) to add deeper levels of interactivity.



This visualization uses [Highlight Actions](#) to increase interactivity. Searching “public” in the wildcard filter highlights the categories of colleges—in this case, public affairs and public health.

7. Format from largest to smallest

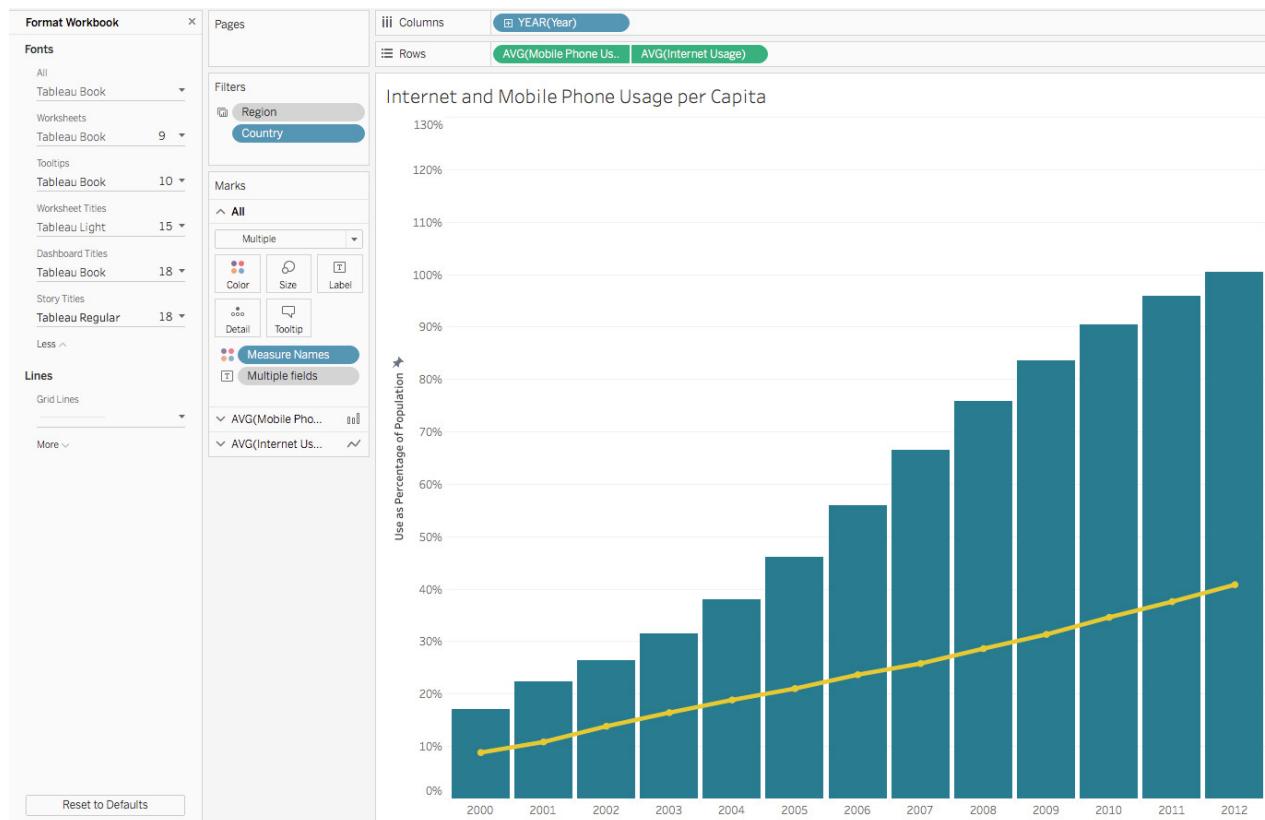
As you change the look and feel of your work, use a “largest to smallest” workflow. This will help you work quickly and keep you from accidentally overwriting your changes.

From a formatting perspective, the hierarchy of a dashboard looks like this:

1. Theme
2. Workbook
3. Worksheet

Start by confirming that you're using the right theme (Tableau's latest and greatest is always called Default). Choose one by going to Format > Workbook Theme.

The next step is to format at the **workbook level**. Here, you can change fonts, titles, and lines across your entire workbook.



Create consistency with formatting. Select Format > Workbook in Tableau to adjust the formatting for your entire workbook

Finally, move on to the **worksheet level**. For example, you might want to remove all the borders in a text table or add shading to every other column in a view. Save this step for last because when you make formatting changes at this level, they apply only to the view you're working on.

For tips on how to quickly give your dashboard a new look, including how to use your own custom fonts and colors, check out [Rebrand a Dashboard](#) in the Online Help.

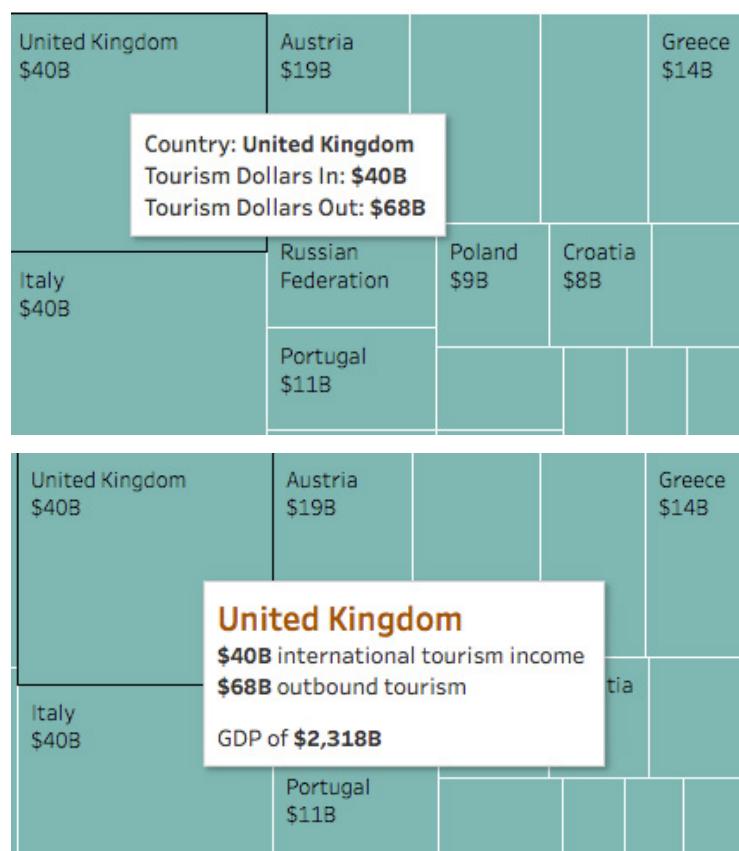
Refining your dashboard

8. Leverage tooltips, the story within your story

Once you're done with the main design work, take a look at your tooltips. Tooltips are a fantastic opportunity to reinforce the story you're trying to tell with your dashboard. They also add helpful context to your view. Tableau populates a view's tooltips automatically, but you can easily customize them by clicking Worksheet > Tooltip.

Just like you want to put your most important view in the upper left of your dashboard, you want the most important elements of your tooltip to be at the top.

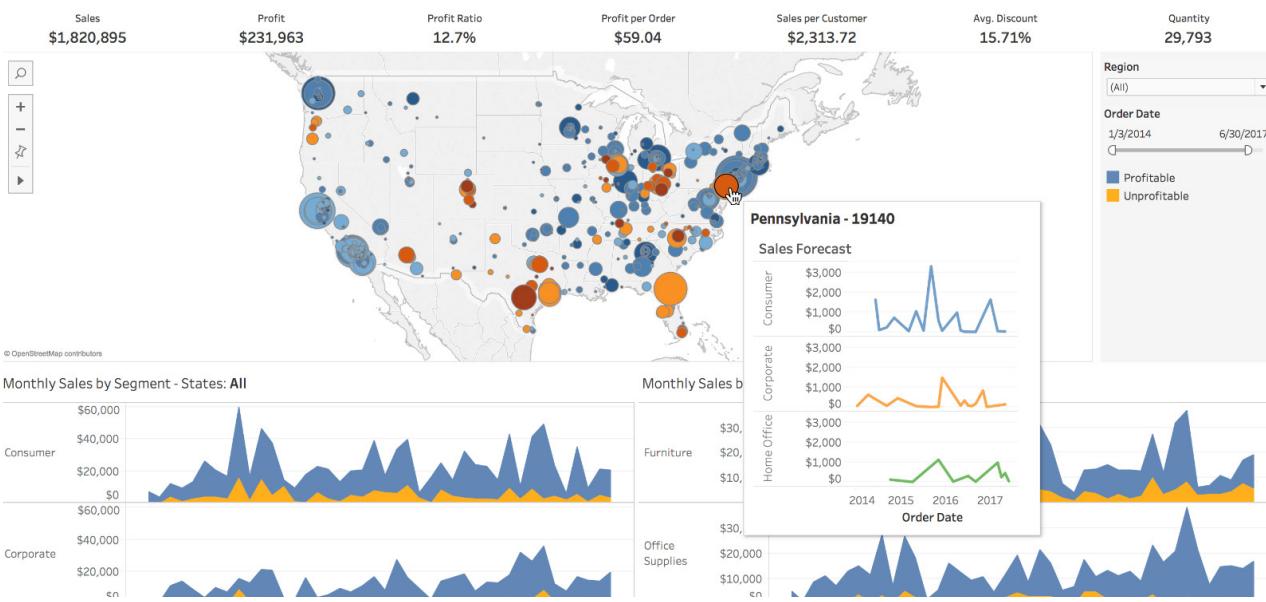
For example, let's say we're looking at a visualization showing international tourism by region and country.



At first glance, the tooltip on the left doesn't tell me what I need to know—what is the international tourism income for each country, related to its overall GDP?

When revised, this tooltip emphasizes the most important elements—the country, its inbound and outbound tourism dollars, and the GDP of the country.

You can also use the [Viz in Tooltip feature](#) to augment your dashboards and stories with relevant data without introducing more clutter. With Viz in Tooltip, place visualizations of your own design into tooltips, revealing them on hover or selection of individual marks. The data in the viz is automatically filtered to the mark you hover on or select, giving you and your users precise views of the pertinent data. As always, make sure that the visualization is enhancing, not distracting from the contents of your dashboard. When in doubt, keep the rest of the dashboard simple and inform the user that they will get more context in the tooltip.

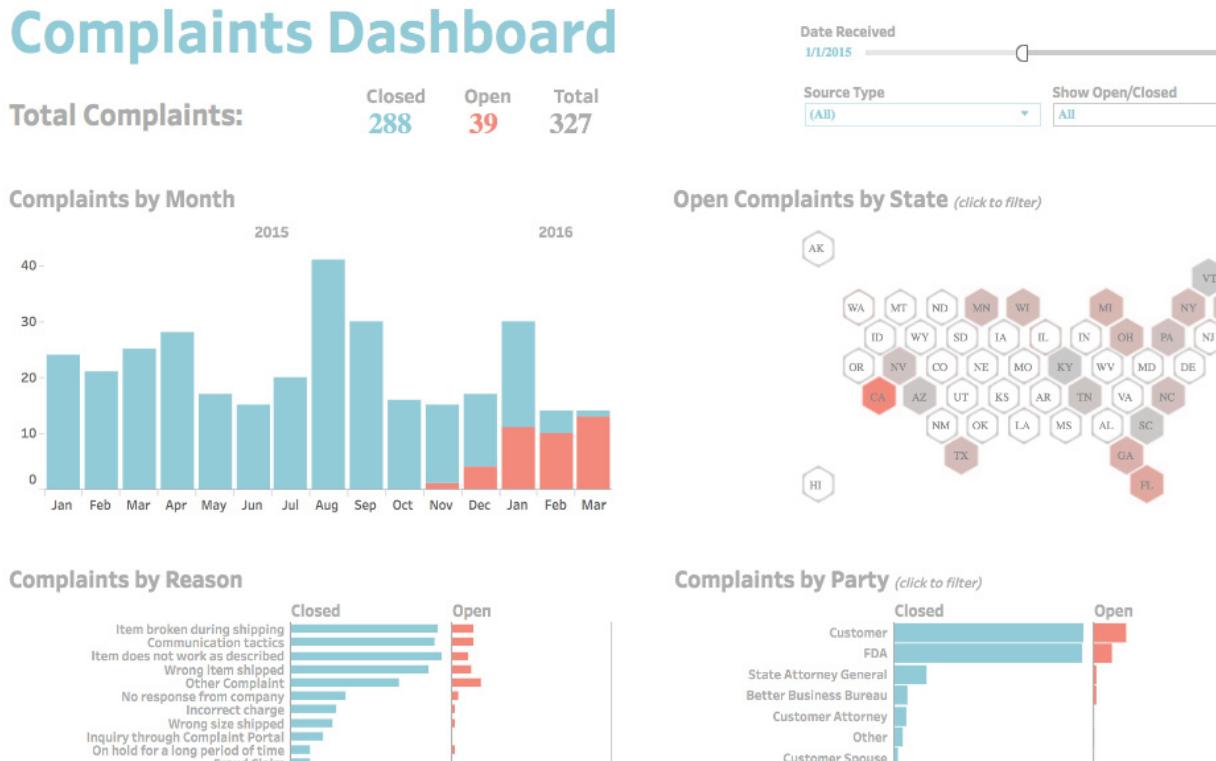


This dashboard leverages the Viz in Tooltip feature. When a user hovers over a mark on the map, the tooltip displays sales forecasts in each segment for that particular state. This adds context without crowding the dashboard.

9. Eliminate clutter

Take a step back and consider your dashboard from the perspective of someone who's never seen it. Every element should serve a purpose. If a title, legend, or axis label isn't necessary, consider getting rid of it.

If your dashboard needs more white space, consider a floating layout. If you go this route, give your dashboard a specific, fixed size so that the item that you floated stays put if the dashboard size changes.



This dashboard is a good example of simple, clean design. When you eliminate clutter and simplify your colors and layout, it is much easier to find hidden insights because you aren't trying to sift through all of the individual elements. Complaints Dashboard, [The Big Book of Dashboards](#).

Simplifying your dashboard design is often an iterative process, so keep going back to existing dashboards with fresh eyes. To start, look at the latest dashboard you created: does it have too much on it? Is there anything you can remove or rearrange to add clarity?

10. Test your dashboard for usability

An important element of dashboard design is user testing. After you build a prototype, ask your audience how they're using the dashboard and if it helps them answer their pressing questions. Have they created their own versions of the dashboard? Are they digging into certain views and ignoring others? Use this information to tweak the existing dashboard or develop new ones.

As with any successful project, good testing is key. Learning how your dashboards are received will help inform future designs and influence how data is leveraged within your organization.

About Tableau

Tableau helps people transform data into actionable insights that make an impact. Easily connect to data stored anywhere, in any format. Quickly perform ad-hoc analyses that reveal hidden opportunities. Drag and drop to create interactive dashboards with advanced visual analytics. Then share across your organization and empower teammates to explore their perspective on data. From global enterprises to early-stage startups and small businesses, people everywhere use Tableau's analytics platform to see and understand their data.

Explore other resources

[Product Demo](#)

[Training & Tutorials](#)

[Community & Support](#)

[Customer stories](#)

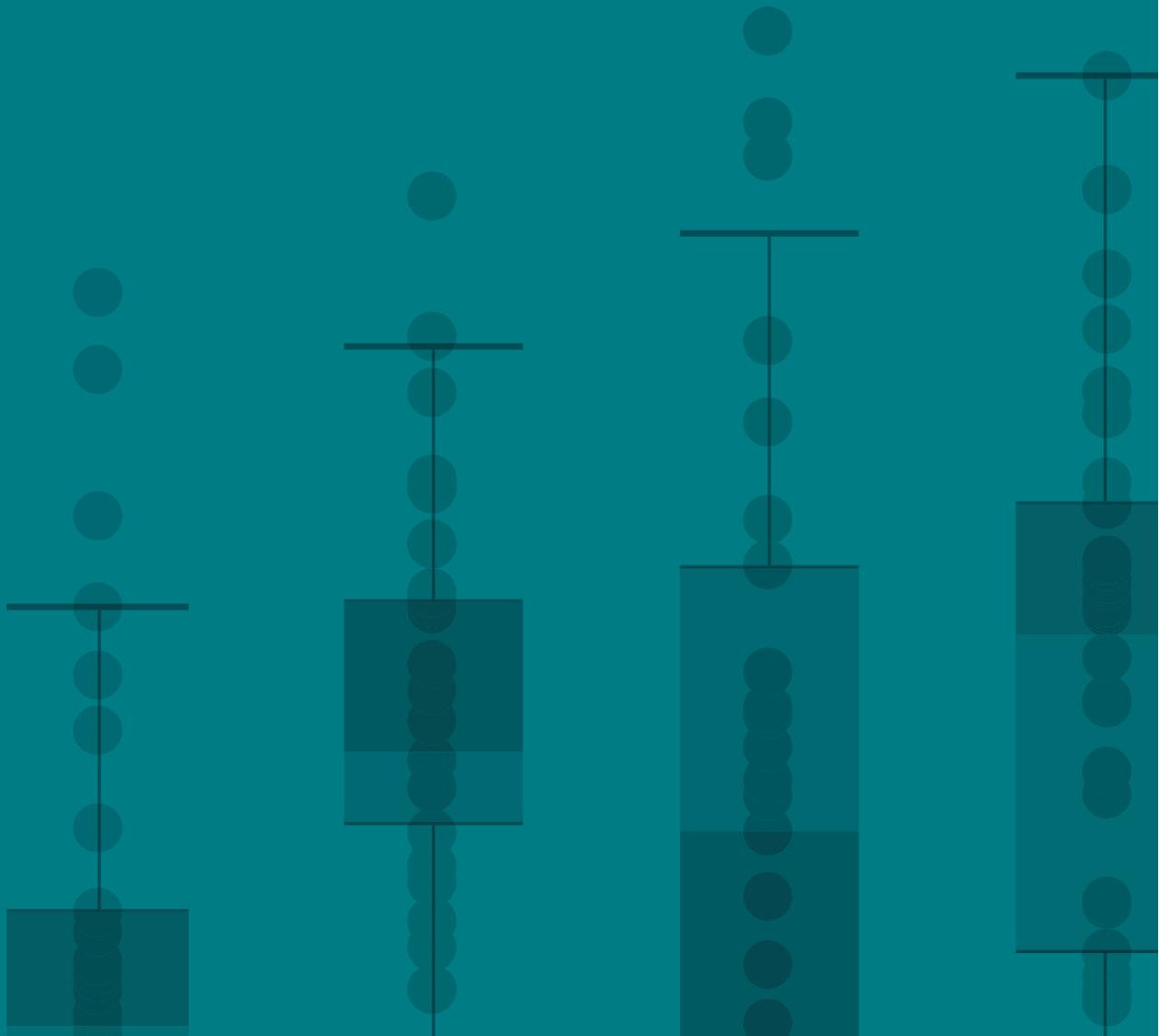
[Solutions](#)





Build Your Competitive Edge:

12 Powerful Retail Dashboards



A common ask from our customers is to see more real-world retail and consumer goods examples. Last year, we responded by asking some of our incredible partners from around the world to provide their best retail and consumer good visualizations—which became the basis of our Top 10 Retail Dashboards for Better Performance white paper.

This year, we've reached out once again to our partner community and our partners have responded with amazing, retail-focused visualizations centered around store operations, merchandising, and marketing. These areas have been a big focus for our customers over the last year, and these dashboards have played a critical role in helping our customers build a competitive edge.

We hope these new visualizations inspire you to help raise the bar of analytics within your organization. We also hope it helps to reinforce the amazing capabilities and industry knowledge our partners have.

Table of Contents

Section One: Store Operations

1.Store Level Product Availability (Atheon Analytics)	3
2.Waste Dashboard (Atheon Analytics).....	5
3.Retail Scorecard (Interworks)	7
4.Retail Executive Overview (Keyrus)	8
5.Emergency Weather Response Predicted Demand (North Highland).....	10
6.Retail Store Heatmapping (North Highland)	11
7.Regional Manager Dashboard (Automated Insights)	12
8.Store Manager Dashboard (Narrative Science)	13

Section Two: Marketing

1.Marketing Mix Models and ROI Engines (Keyrus)	15
2.Consumer Segmentation (Keyrus).....	16
3.Digital Content Optimization (North Highland).....	17
4. Voice of the Customer (VoiceBase)	18



1. Store Level Product Availability

Partner: **Atheon Analytics**

Find it on **Tableau Public**

What is Store Level Product Availability and how does data visualization help address it?

Having the right product in the right store at the right time is one of the fundamentals of retailing. Having the best price or the best marketing strategy in the world doesn't matter if the customer cannot buy what they want, where they want, and when they want to buy it.

Availability is important—and costly—for brands and retailers. If a product is out of stock, the customer may substitute with another product, which means the brand will record a lost sale and the customer may permanently switch to the competing product. Customers can be fickle. If a customer's favorite products are not available, they may choose to do their shopping elsewhere, and the retailer loses out on not just the unavailable product, but the entire shopping basket of related goods.

Carry too much product, and brands and retailers will face product spoilage and increase costly waste, or tie up shelf space with under-performing products. Too much product also increases needed working capital, and our friends in the finance department won't appreciate that added investment.

The problem is that retailers generally look at product availability as a percentage. While percentage is a useful measure, it doesn't capture the size and magnitude of the problem—that is, missing distribution points. Resources are wasted fixing low distribution/low sales products with low availability, rather than focusing on what will have the largest economic impact.



SKU A

- Allocated to 200 stores and is Out of Stock in 40 Stores
- Availability percentage of 80% $(200-40)/200$.

SKU B

- Allocated to 1000 stores and is Out of Stock in 100 Stores
- Availability percentage of 90% $(1000-100)/1000$.

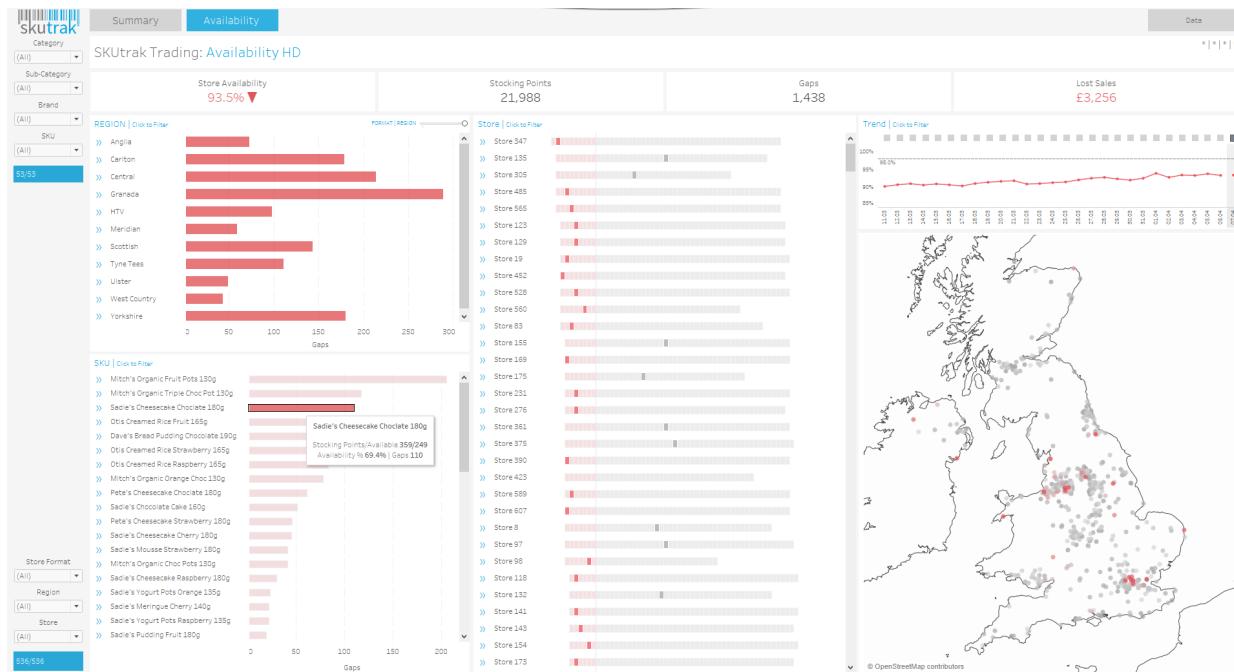
SKU B is missing in over 2.5x as many stores (over twice as many customers can't purchase the product), but from a percentage measure looks to have better availability than SKU A.

The other problem is not being able to see which stores have the problem (and is there a pattern to the out of stock, such as stores serviced from a specific depot or warehouse).

Key takeaway: better insight into store level availability helps retail managers and brands

By analyzing the data visually and adding interactivity, both the retailer and brand (FMCG Supplier) can quickly find the biggest problems by product, geography, format, and store in a few clicks. The worst products, geographies, formats, and stores needing immediate attention are easy to identify.

This visualization can be used every day to find and fix these issues. It's also useful for brand sales management to direct the field sales teams to stores with the biggest problems and maximize the return on investment. When this process is transformed via visual analytics, the customer, the brand, the retailer, and the financial departments all win.



Try it for yourself



2. Waste Manager

Partner: [Atheon Analytics](#)

Find it on [Tableau Public](#)

What is food waste and how does data visualization help identify it?

Waste on fresh products refers to products that are either thrown away or reduced in price due to expiring shelf life. Waste costs the retailer real money, and any savings on waste directly impacts profits. There is also corporate responsibility and sustainability pressure to reduce food waste.

Properly identifying wasteful products and the cause of waste is a big problem for retailers and quick serve restaurants (QSRs). Due to the difficulty of consuming large text files in spreadsheets, retailers are forced to look at waste in aggregation, and not at store level, preventing retailers or QSRs from taking appropriate action.

Often, products with high waste are only problems in specific stores. Correcting these stores—either by changing product forecasts, facings, removing them from the menu or delisting altogether—can have a huge impact on waste reduction, but can also cause customer churn.

Let's look at High End Steak, a hypothetical example. High End Steak only comes in a pack-size of 12, with shelf life of 7 days, and sells at a retail price of £8.

100 Stores (with customer segmentation: high traffic, weekly shop) sell at full retail price 24 units a week, and have a zero theoretical waste:

- Sales value £9,600
- Waste (at retail) £0

60 Stores (with customer segmentation: low affluence, low traffic, weekly shop) sell at full price 8 units a week, and another 4 at 40% reduction due to expiring product dates.

- Sales value £4,900
- Waste (at retail) £760

80 Stores (with customer segmentation: high affluence, high traffic, convenience food-to-go) sell at full price only 4 units a week, and another 3 at 40% reduction and bin 5 (you can't send in half a pack).

- Sales value £4,700
- Waste (at retail) £4,000 (RTC £800 + Bin £3,200)

The total sales revenue for this organization was £19,200. Waste at retail) £4,760, and waste as a percentage of sales was 25%.

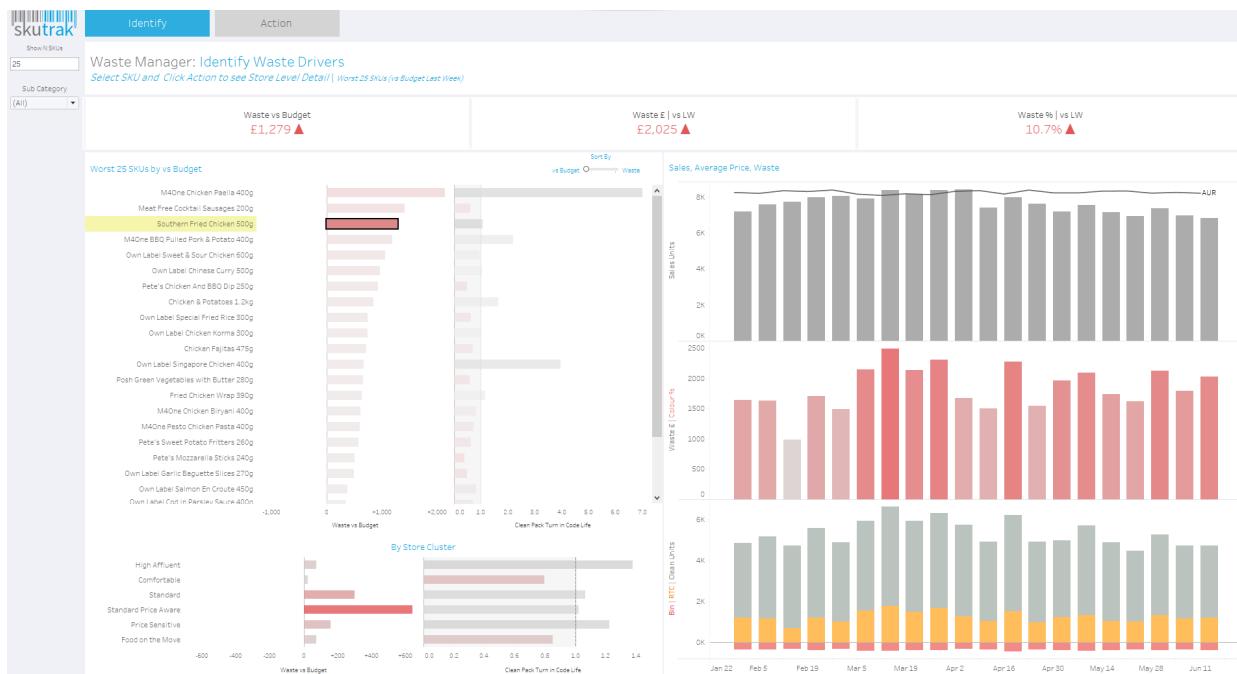


The organization has some choices to make:

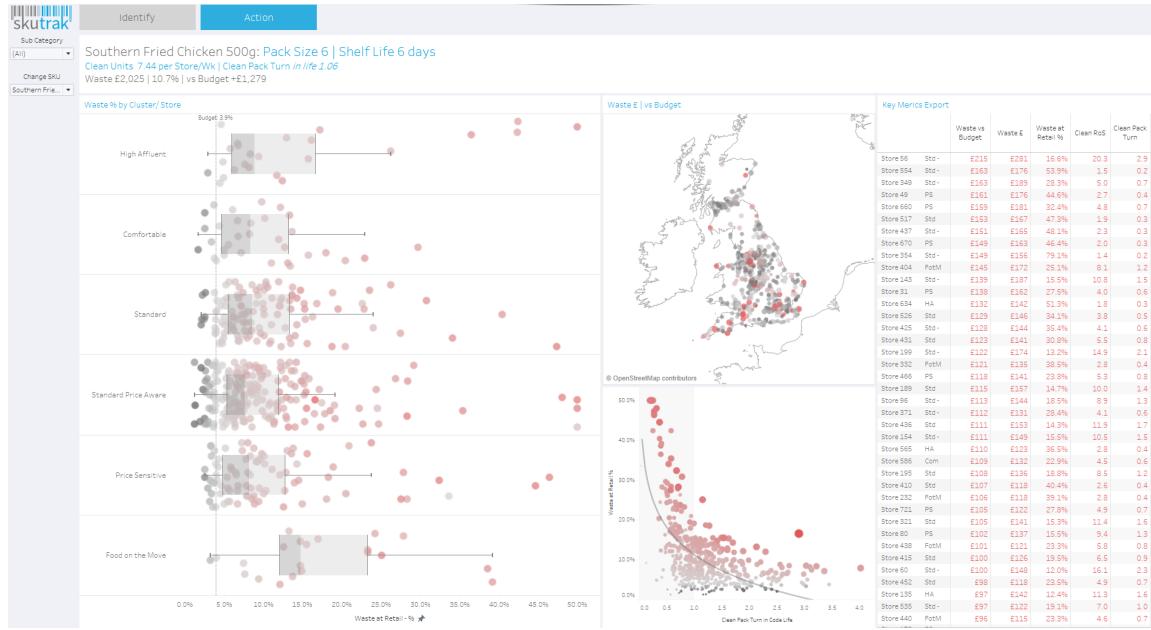
- Live with the waste to ensure customers will not be impacted as we hear that many customers love this product
- Remove the product from the product allocation—at the risk of annoying or losing customers (if we could see and make sense of the data at store level, we could remove it from the convenience stores selectively and maybe entice customers to choose ready-made meals, in the lowest traffic low'affluence stores)
- Optimize the impact of waste by reducing the pack size to 6, so most convenience stores could sell through their inventory before the expiration date.

Key takeaway: visualizing waste provides powerful, actionable information

Waste Manager dashboard allows the retailer to quickly identify the products that are selling. Then by deconstructing the waste value into its constituents (clean sales, RTC sales, and binned) the cause of the problem can be seen. It also looks at the main cause of waste—sales velocity in some stores is too low to sell a whole pack at full retail within the product's life. In addition to seeing enterprise level data, the dashboard also shows break down by store cluster. This lets the retailer or QSR quickly identify whether the problem is the same in all stores, or in a particular store type. This is also shown over time, so it can be seen if the waste is a consistent problem, only when promoted, or related to some other event.



The Waste Manager dashboard then allows the user to interact with the data and see the clusters and stores that are driving waste. As stores are selected, the key information is shown in the table, which can then be exported or shared with the vendor or the merchandising team.



Try it for yourself

3. Retail Scorecard

Partner: [Interworks](#)

Find it [here](#)

What is a retail scorecard and why is it important?

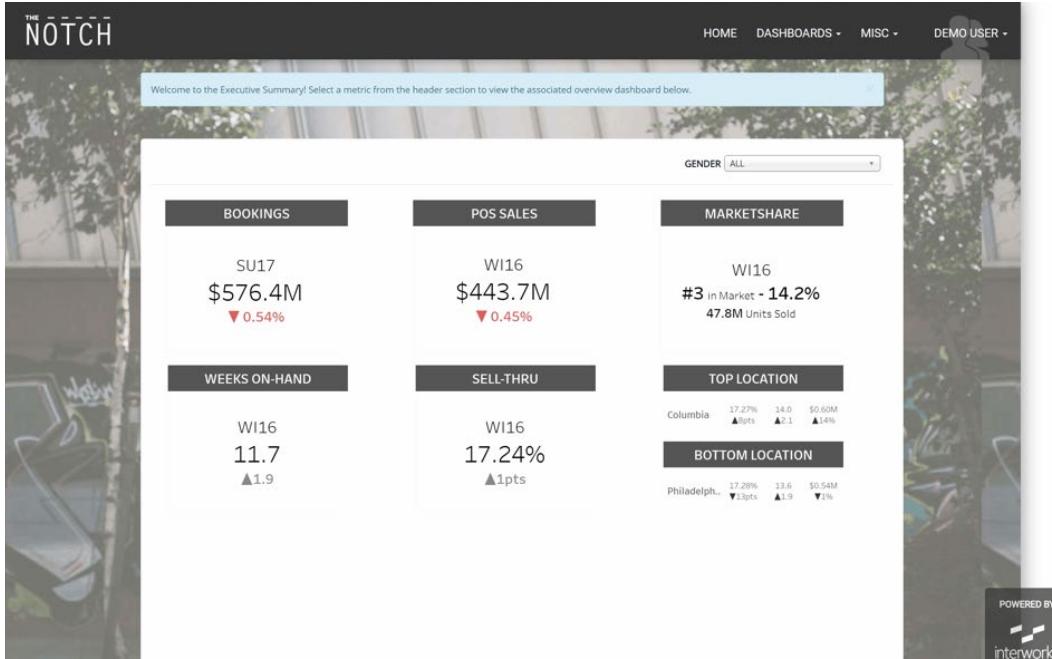
Retail data is typically siloed information that's difficult to access, analyze, or make actionable. With this retail scorecard key KPI screens are brought together in one place. Now, a general manager or VP can see their entire business across functions without having to navigate anywhere else. High level 'cards' contain snapshots of each KPI. Selecting a card accesses a drill-down dashboard specifically developed for that business function. A set of filters at the card level filters every card and every drill-down dashboard with one click.

Because time matters on the retail floor, JavaScript functions pre-load all the background dashboards to ensure each screen loads in less than one second. This is truly a next generation general manager or VP experience to view the entire business' health with detailed action items. This technique requires minimal expertise in Tableau Desktop and can be designed or maintained by any client team. Cards themselves are simple dashboards of their own.



Key takeaway: automated data analytics means more time to focus on the bottom line

This system is simple to build, applicable across all retail scenarios, built with speed-to-insight in mind and represents an achievement that most retailers strive to achieve.



4. Retail Executive Overview

Partner: **Keyrus**

Find it on [Tableau Public](#)

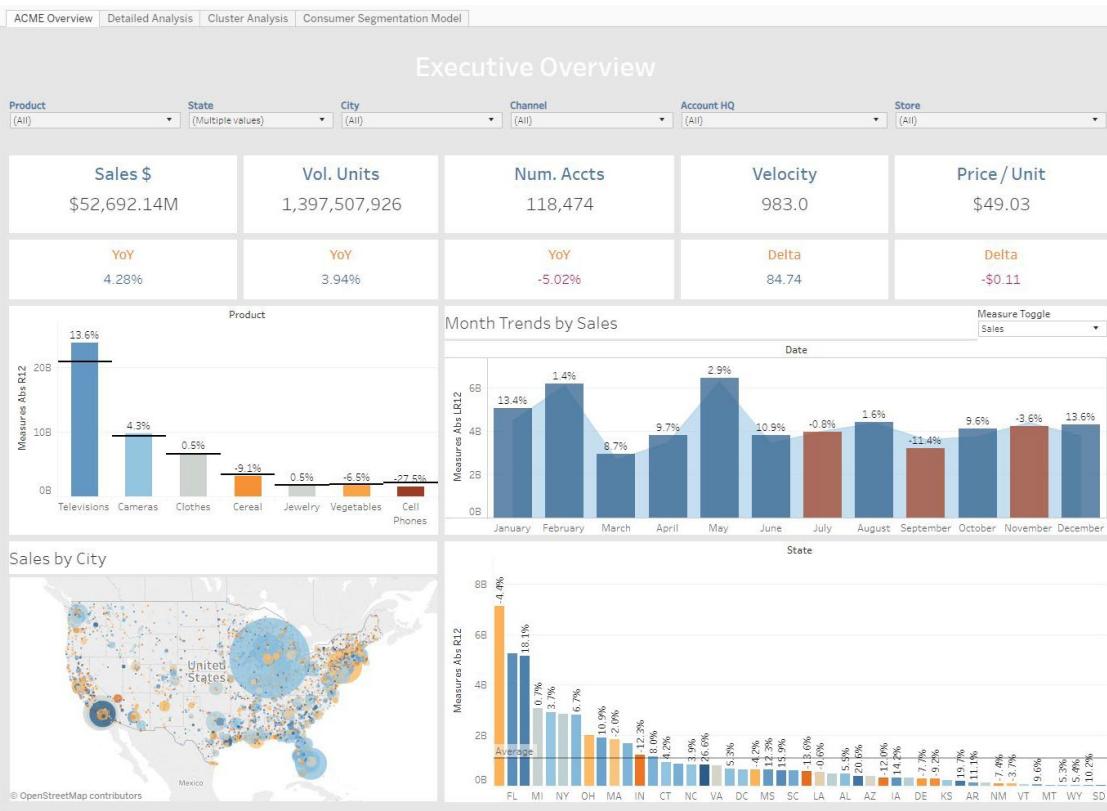
What does a Retail Executive Overview capture?

As we talk with our retail customers, many executives share excitement in being able to finally see a holistic view of their retail data. This dashboard shows high-level KPIs that convey the overall health of the organization.

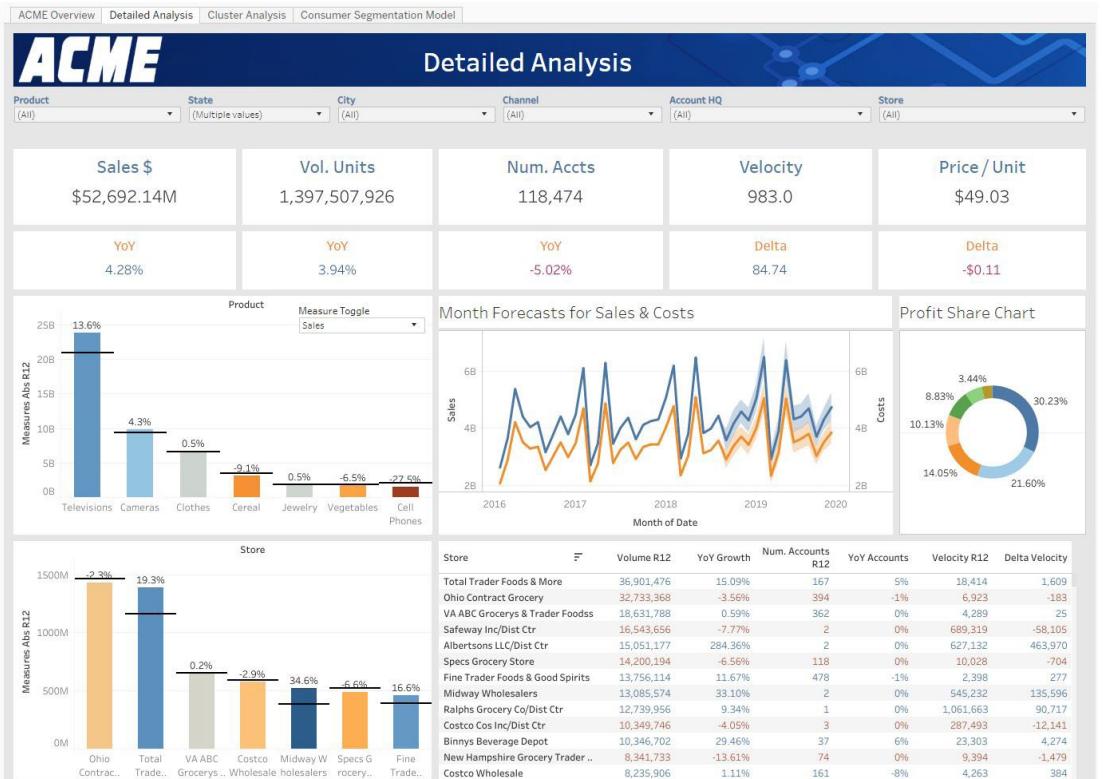
Key takeaway: a high-level overview provides the KPIs executives need, at a glance

Executives can quickly see trends, year-over-year performance by product, and geographic sales performance.





Executives are also empowered to ask additional questions of the data. They can use filters and built-in performance forecasting for any combination of product, state, city, or store.



Try it for yourself



5. Weather Response Predicted Demand

Partner: **North Highland**

Find it on [Tableau Public](#)

What is a Weather Response Predicted Demand dashboard and why is it important?

Not only do retailers have to deal with changing consumers, digitalization, and increased competition, but they also have to deal with extreme weather events.

In response to emergency weather events, North Highland's Predicted Demand dashboard enables mass retailers to predict extreme spikes in product demand, as well as the specific quantities, products, and locations that will be impacted. Insights from the dashboard enable retailers to funnel this new demand into their existing supply chain processes for quick execution in times of need.

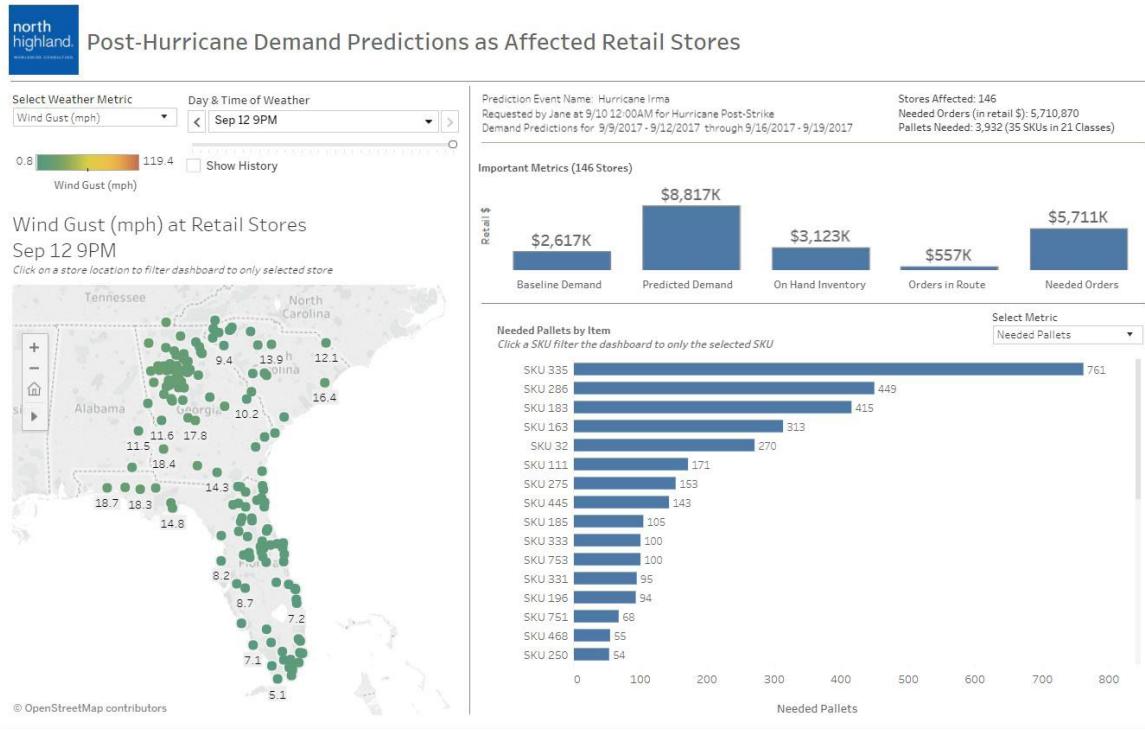
Powered by machine learning, North Highland's Tableau dashboard analyzes and distills insights from over 73 billion historical NOAA weather data points, regional demographics and psychographics, retail sales history, and inventory history to equip retailers with a succinct snapshot of anticipated demand during emergency weather events including floods, hurricanes, and tornadoes.

Key takeaway: On-demand weather data can be used in tandem with inventory data, to anticipate demand

The dashboard offers a clear summary of baseline demand, predicted demand, on-hand inventory, orders in route, and the remaining gap in needed orders—enabling the informed allocation of inventory and resources to ensure that supplies are delivered to the right place at the right time.

For large retailers with nationwide store footprints, the dashboard also enables users to drill in by region, product category, or timeframe to zero in on demand at the SKU, store, and day level—allowing decision makers to quickly mobilize on next steps and accelerate emergency response and recovery efforts, helping customers when they need it most.





Try it for yourself

6. Retail Store Heat Mapping

Partner: **North Highland**

Find it on [Tableau Public](#)

How can retail store heat mapping data improve sales?

Gone are the numbers only, grid-based sales flash reporting. Modern retailers are using visual analytics to gain greater insight within the physical space of brick and mortar stores.

Store heat-mapping analytics give retailers the ability to understand a wide variety of information about physical store layout and performance without having field teams physically travel to each store or consume valuable time from store staff.

Key takeaway: Heat maps allow retailers to optimize store layouts without the manual guesswork of the past

Leveraging industry-standard mapping technology, the solution reads store CAD maps and overlays analysis such as sales per square foot, product locator services, planogram compliance, seasonal reset location and performance, shelf/fixture profitability, and sales velocity.



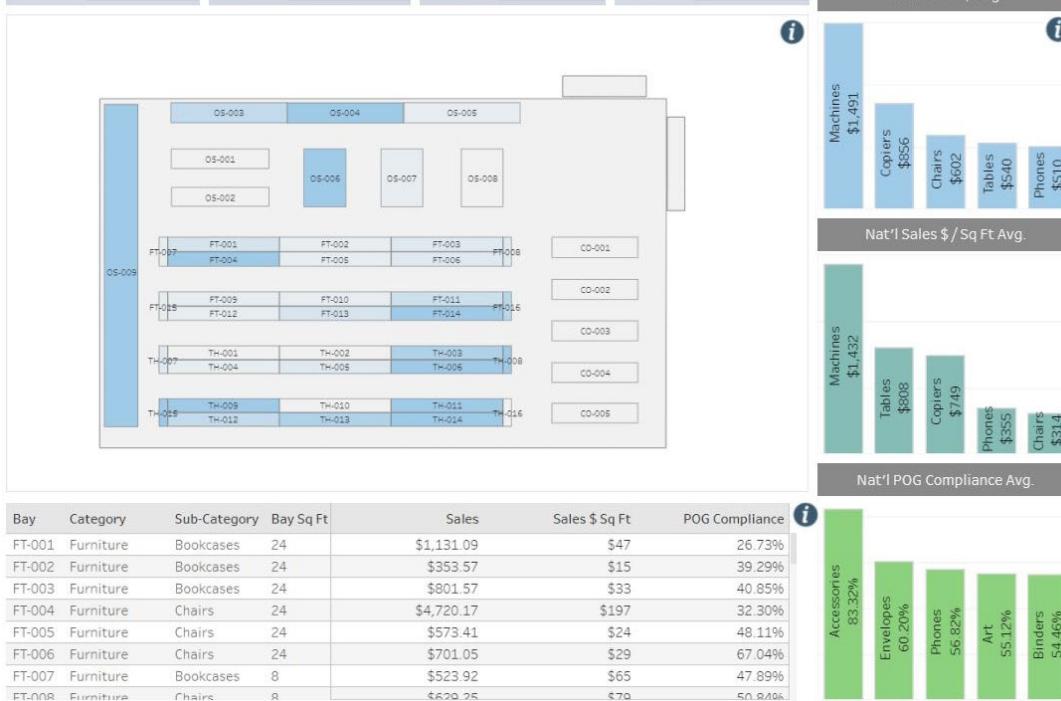


SUPERSTORE HEAT MAP

(Hover over the information buttons to learn more about each section)

Select the Time Period, State, Store # and Bay Metric to display on the Heat Map

Time Period: Active Season | State: New York | Store #: 10011 | Bay Metric: Sales \$



Try it for yourself

7. Regional Manager Dashboard

Partner: [Automated Insights](#)

Find it [here](#)

How does a Regional Manager dashboard help drive better decision-making?

Analytics have become one of the most powerful tools for retailers to derive operational insights from their point of sale databases. Using Tableau, analysts can build guided visual analytics that are quick and easy to digest and broadly distribute data across massive retail networks. Adding written analytics is a great way to explain the context behind visualizations and ensure the way information is consumed is consistent with the strategy and goals set forth by leadership.

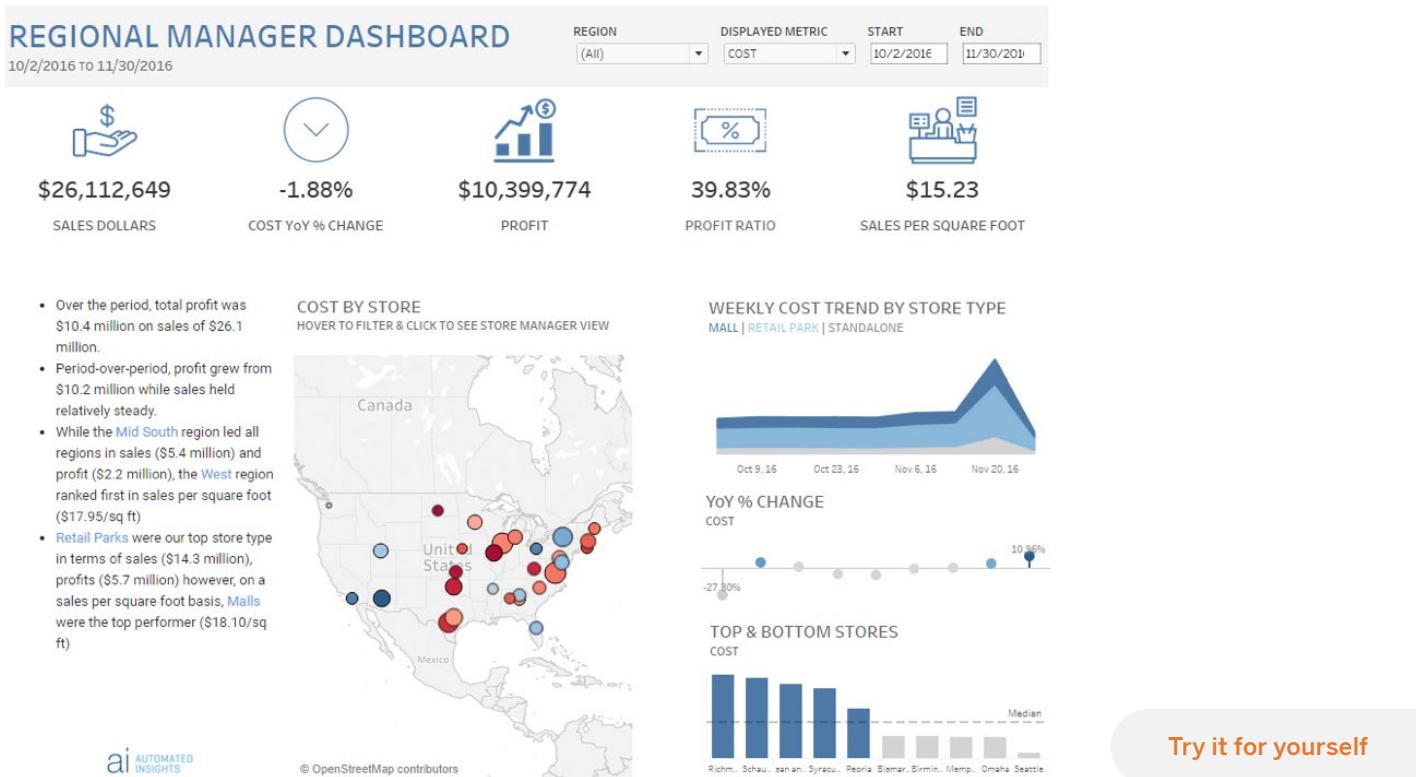
Combining visual analytics with a written explanation reduces the training required to interpret the information.

Whether it's sharing inventory analytics with your supply chain or distributing sales KPIs to brick-and-mortar locations, there's a lot to gain from creating a data-driven culture. This dashboard gives retailers the big picture of nationwide sales and merchandising by not only showing what's going on, but also describing the broader context of why it's happening or how it's relevant. The combination of both visual and written analytics ensures dashboard consumers are able to explore data in a meaningful way to identify trends, events, and KPIs efficiently without misinterpretation.



Key takeaway: by embedding automated written analysis, regional managers can receive role-based insights that provide complete context and drive action.

Regional managers can see their store performance in context of the greater enterprise and are able to drill into underperforming stores and focus on the exact departments and products that need attention.



8. Store Manager Dashboard

Partner: **Narrative Science**

Find it [here](#)

What is a Store Manager Dashboard and why is it important?

Store managers struggle with legacy BI reports that are tens of thousands of rows of data in Excel or non-interactive PDF reports. With store labor constantly being evaluated and reduced, operational efficiency has to be a focus. Getting from problem to analysis to action in today's fast-paced retail environment has to happen in seconds—not minutes.

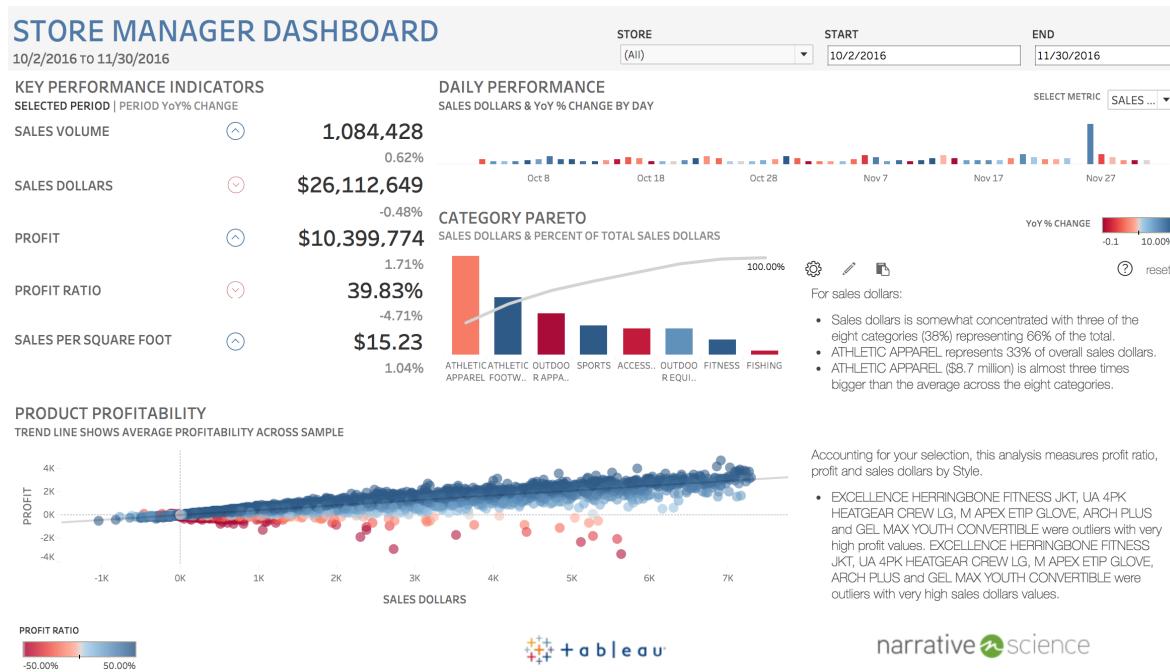
Narratives for Tableau instantly provides plain-English explanations of charts and graphs, making it possible for store managers to identify and communicate key insights faster. This natively-built Tableau dashboard extension provides a seamless experience, analyst-quality insights, and the ability to consume a narrative based on a store manager's interaction with the dashboard.



Key takeaway: store managers need actionable data—and they can obtain it quickly with this dashboard

In the example below, the Store Manager of fictional store Gravitas Sporting Goods located in Oklahoma City can go to their dashboard to see the primary driver of the decline in sales was due to weak performance in the Athletic Apparel category. Store managers see the visual analysis, read the narrative, and have confidence they have the right information to correct the issue.

From there, they can dig into the product performance within Athletic Apparel and come up with a plan for underperforming products.



Try it for yourself



1. Marketing Mix Models and ROI Engines

Partner: **Keyrus**

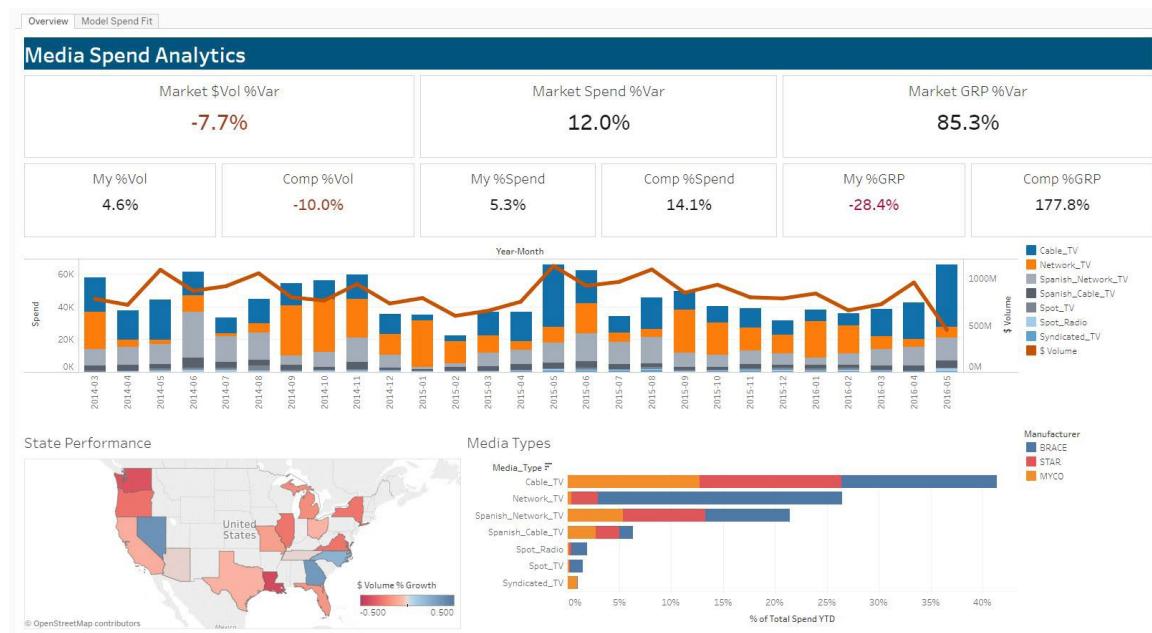
Find it on **Tableau Public**

What insights do Marketing Mix Models and ROI Engines provide?

Frequently, marketing performance measurement and sales activity are analyzed separately, because it's hard for many organizations to combine them. But measuring marketing spend and actual sales performance together can help identify marketing campaigns that have direct sales impact. Applying machine learning to the underlying combined datasets helps optimize the marketing mix and achieve maximum ROI.

Key takeaway: combining two distinct, and often siloed data sources, can help to identify marketing and sales effectiveness.

Retailers can use machine learning to maximize marketing mix to drive marketing ROI, ensuring marketing campaigns have a tangible impact on increasing revenues.



Try it for yourself



2. Marketing Consumer Segmentation

Partner: **Keyrus**

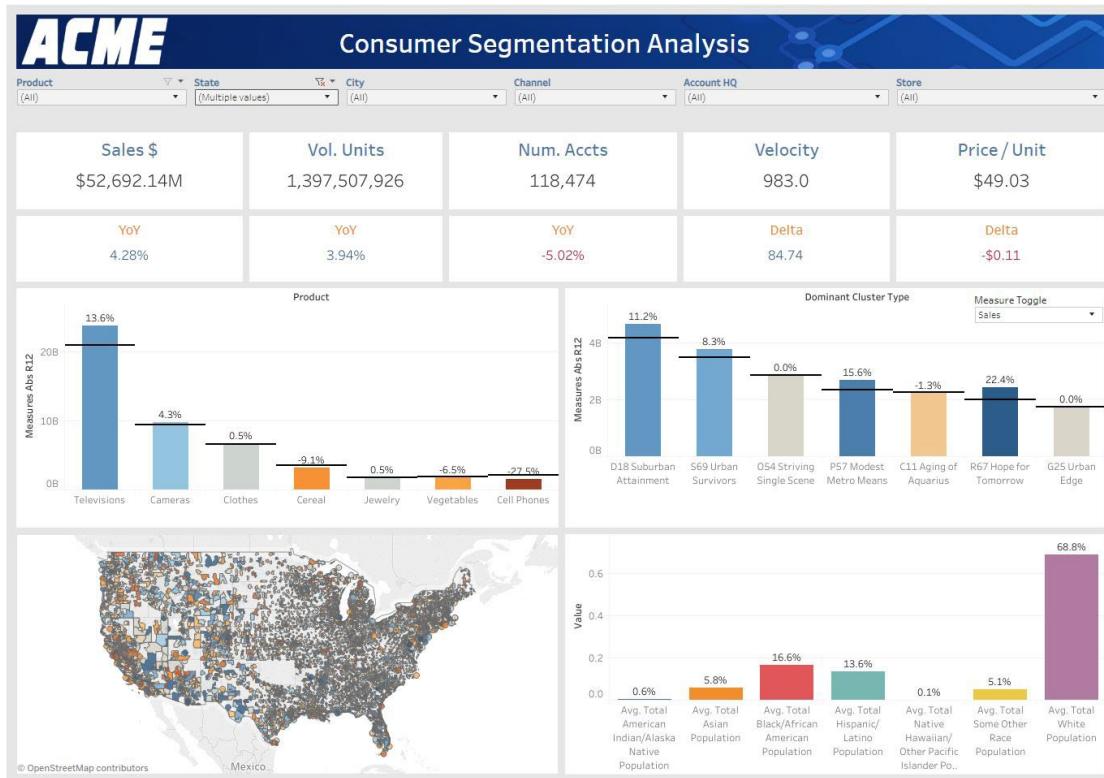
Find it on **Tableau Public**

What is Marketing Consumer Segmentation and why is it important?

With razor thin margins, every dollar of retail spending needs to be analyzed, scrutinized, justified, and optimized. Consumer brands need to know who their customer is, what they like, what they want, and where they are. With more pressure on marketing budgets, it is essential to optimize spending and know what consumer segments need to be prioritized.

Key takeaway: granular insight improves the quality of decision making

Marketing executives can measure KPIs by designated market area (DMA), sales impact, and dominant cluster type. With hyper focus and no guesswork, marketing can maximize every spend, expand key customer segments, or winning back key consumers whose loyalty may be wavering.



Try it for yourself



3. Digital Content Optimization

Partner: **North Highland**

Find it on **Tableau Public**

What is a Digital Content Optimization dashboard?

Consumers shop online in ever-growing numbers, so in order to keep their attention—and retain their loyalty—content has to be optimized for speed, engagement, and conversion. North Highland's Digital Content Optimization dashboard provides optimized, tactical recommendations that drive e-commerce performance, grounded in insights from the visual analysis of specific digital assets, including individual text, video, and image components.

This Tableau-powered dashboard determines the optimal combination of assets that results in the highest conversion of visitors to buyers, considering each digital asset on a specific product information page across all products in a product category. The dashboard leverages what we know about the assets already: image attributes such as type of image and count of images, text attributes such as number of words, bullets, paragraphs and mentions of specific topics, and video attributes such as length and count. It also supplements this information with what we can learn from cognitive recognition tools—such as objects and actions within images, the presence of people, language, and sentiment within videos—to fully understand what the customer is responding to. Once it's understood how these factors relate to sales, the solution recommends the optimal mix of digital assets to maximize conversion.

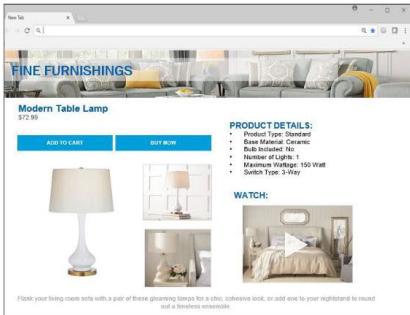
Key takeaway: understanding—and responding to—your online customers' habits and preferences is essential to driving revenue

The opportunity for digital optimization is large for retailers. One mass retailer realized 100x ROI from increased purchase conversion in the first year following use of North Highland's solution and dashboard. Another company realized over \$27M revenue opportunity for one of its brands, leading to actionable improvement recommendations that will be applied to other brands.

Ultimately, North Highland's dashboard solution helps leaders right-size their asset capabilities and marketing spend to help inform and define future state action plans against recommendations that align to content and brand standards.



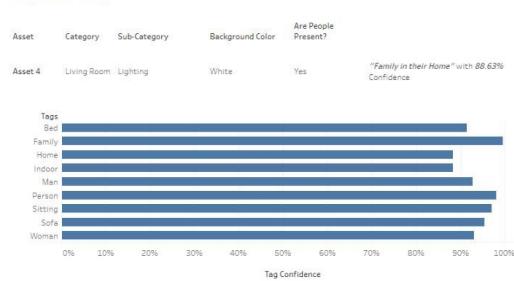
Product Web Page



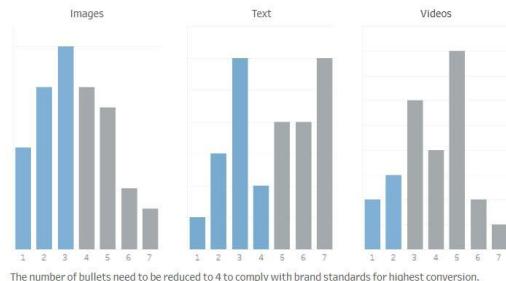
Digital Asset



Asset Analysis Metrics



Compliance and Key Takeaway



Try it for yourself

4. Voice of the Customer

Partner: **VoiceBase**

Find it on **Tableau Public**

What is Voice of the Customer and why is it important?

In today's digital marketplace, retailers must gain a better understanding of customers' needs, wants, and concerns by continually gathering, analyzing, and acting on customer feedback. And listening to the customer is more important than ever: according to a Walker study, by the year 2020 customer experience will overtake price and product as the key brand differentiator. Voice of the Customer (VoC) dashboards help retailers deliver enhanced customer experiences, engage employees and drive business change. The customer voice can be loudly heard in emails, support chats, voice calls and sales interactions, chatbots and messaging applications, tweets, social platforms, video, and surveys.

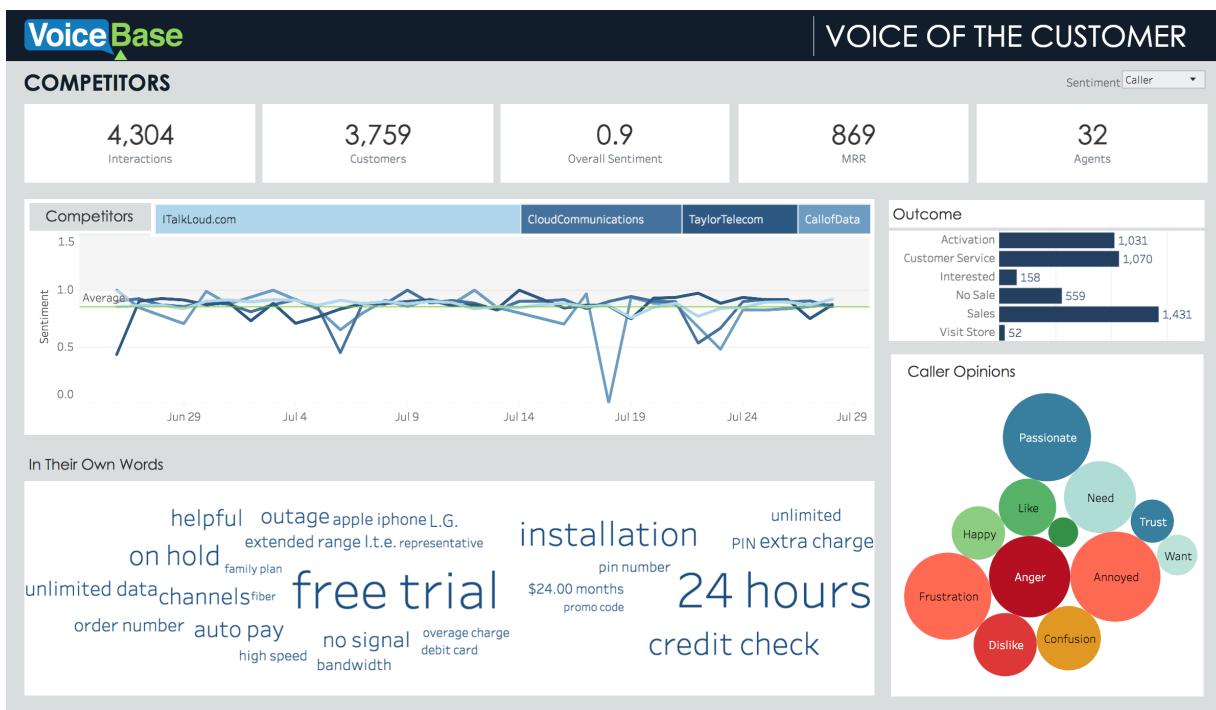
One primary channel for Voice of the Customer analysis is speech analytics, which transcribes and analyzes millions of customer calls to discover actionable insights, close more sales, and improve business performance. Speech analytics has historically been limited to the call center.

VoiceBase's open data architecture allows departments outside of the call center to harness rich insights available through voice calls and create true 'Voice of the Customer' dashboards with Tableau.



Key takeaway: VoC can help you understand your customer and how they view your products, services, and brand

The dashboard below shows the VoC Competitor analysis using VoiceBase transcribed calls to improve competitive positioning and enhance customer experience. You'll see KPIs of how customers describe competitive brands, including emotions, call volume trends and organic words used with sales and service representatives. This allows you to be able to quickly see how these trends affect the call outcome.



[Try it for yourself](#)



Summary

Today's retailers need an edge in order to thrive—not merely survive—in a turbulent landscape where consumer behavior is constantly changing, and customer-centric experiences are essential. To make more informed decisions, savvy retailers are increasingly embracing a data-driven approach.

Using data-driven dashboards, it's possible for retailers to see and understand their data in new ways, and build strategic and competitive advantage with granular, actionable insights. As you've seen in the dashboard examples in this paper, the width and breadth of what retailers can do with visual analytics is impressive: from assessing store level availability to optimizing store layouts, it's clear that having the right data at the right time is key to maximizing profitability and scaling for the future.

Connect with our Partners

Many thanks to our generous partners for sharing these dashboards with us. Please reach out to them to learn more about how you can implement Tableau dashboards like these and harness the power of your data.

Partner	Contact	Email	Phone
Atheon Analytics	Simon Runc	simon.runc@atheon.co.uk	+44-08444-145-501
Automated Insights	Peter Benson	peter@automatedinsights.com	919-824-7671
Interworks	Derrick Austin	derrick.austin@interworks.com	405-533-1039
Keyrus	Razvan Nistor	Razvan.nistor@keyrus.us	646-664-4872
Narrative Science	Shawn Parks	sparks@narrativescience.com	312-477-0590
North Highland	Dan Kopp	dan.kopp@northhighland.com	770-314-9418
VoiceBase	Emily Blazensky	emily@voicebase.com	408 702-7160

Resources

[7 Tips and Tricks from Dashboard Experts](#)

[Tableau Retail and Wholesale Analytics solutions page](#)

[Top 10 Retail Dashboards for Better Performance](#)





About Tableau

Over 80% of the top 100 retailers and over 7,000 retail and consumer goods companies around the world trust Tableau to help them understand their data and create actionable insights. On the Tableau platform, it's easy to explore your data, build dashboards, and perform ad hoc analyses in just a few clicks.

[Download a free trial](#) and experience the power of Tableau for yourself.

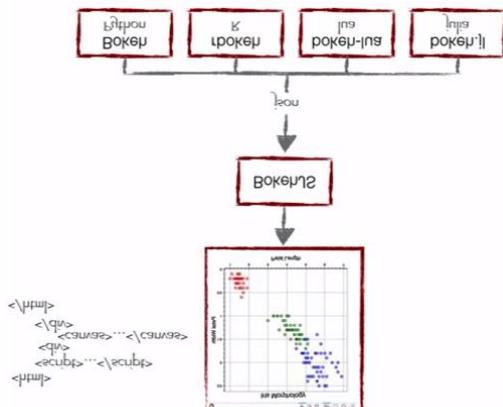
Interactive Data Visualization using Bokeh (in Python)

Introduction

Recently, I was going through a [video from SciPy 2015](#) conference, “[Building Python Data Apps with Blaze and Bokeh](#)”, recently held at Austin, Texas, USA. I couldn’t stop thinking about the power these two libraries provide to data scientists using Python across the globe. In this article, I will introduce you to the world of possibilities in data visualization using Bokeh and why I think this is a must learn/use library for every data scientist out there.

What is Bokeh?

Bokeh is a Python library for interactive visualization that targets web browsers for representation. This is the core difference between Bokeh and other visualization libraries. Look at the snapshot below, which explains the process flow of how Bokeh helps to present data to a web browser.



Source: Continuum Analytics

As you can see, Bokeh has multiple language bindings (Python, R, Lua and Julia). These bindings produce a JSON file, which works as an input for [BokehJS](#) (a Javascript library), which in turn presents data to the modern web browsers.

Bokeh can produce elegant and interactive visualization like D3.js with high-performance interactivity over very large or streaming datasets. Bokeh can help anyone who would like to quickly and easily create interactive plots, dashboards, and data applications.

What does Bokeh offer to a data scientist like me?

I started my data science journey as a BI professional and then worked my way through predictive modeling, data science and machine learning. I have primarily relied on tools like

QlikView & Tableau for data visualization and SAS & Python for predictive analytics & data science. I had near zero experience of using JavaScript.

So, for all my data products or ideas, I had to either outsource the work or had to pitch my ideas through wire-frames, both of which are not ideal for building quick prototypes. Now, with Bokeh, I can continue to work in Python ecosystem, but still create these prototypes quickly.

Benefits of Bokeh:

- Bokeh allows you to build complex statistical plots quickly and through simple commands
- Bokeh provides you output in various medium like html, notebook and server
- We can also embed Bokeh visualization to flask and django app
- Bokeh can transform visualization written in other libraries like matplotlib, seaborn, ggplot
- Bokeh has flexibility for applying interaction, layouts and different styling option to visualization

Challenges with Bokeh:

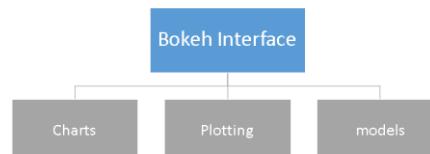
- Like with any upcoming open source library, Bokeh is undergoing a lot of development. So, the code you write today may not be entirely reusable in future.
- It has relatively less visualization options, when compared to D3.js. Hence, it is unlikely in near future that it will challenge D3.js for its crown.

Given the benefits and the challenges, it is currently ideal to rapidly develop prototypes. However, if you want to create something for production environment, D3.js might still be your best bet.

To install Bokeh, please follow the instruction given [here](#).

Visualization with Bokeh

Bokeh offers both powerful and flexible features which imparts simplicity and highly advanced customization. It provides multiple visualization interfaces to the user



as shown below:

- **Charts**: a *high-level* interface that is used to build complex statistical plots as quickly and in a simplistic manner.
- **Plotting**: an *intermediate-level* interface that is centered around composing visual glyphs.

- **Models**: a *low-level* interface that provides the maximum flexibility to application developers.

In this article, we will look at first two interfaces charts & plotting only. We will discuss models and other advance feature of this library in next post.

Charts

As mentioned above, it is a high level interface used to present information in standard visualization form. These forms include box plot, bar chart, area plot, heat map, donut chart and many others. You can generate these plots just by passing data frames, numpy arrays and dictionaries.

Let's look at the common methodology to create a chart:

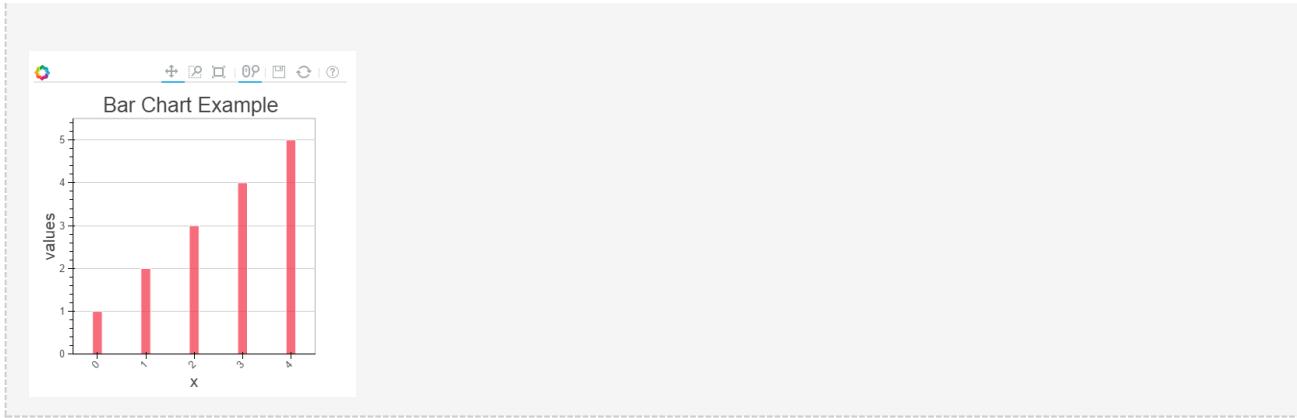
1. Import the library and functions/ methods
2. Prepare the data
3. Set the output mode (Notebook, Web Browser or Server)
4. Create chart with styling option (if required)
5. Visualize the chart

To understand these steps better, let me demonstrate these steps using example below:

Charts Example-1: Create a bar chart and visualize it on web browser using Bokeh

We will follow above listed steps to create a chart:

```
#Import library
from bokeh.charts import Bar, output_file, show #use output_notebook to visualize it
in notebook
# prepare data (dummy data)
data = {"y": [1, 2, 3, 4, 5]}
# Output to Line.HTML
output_file("lines.html", title="line plot example") #put output_notebook() for notebook
# create a new line chart with a title and axis labels
p = Bar(data, title="Line Chart Example", xlabel='x', ylabel='values', width=400, height=400)
# show the results
show(p)
```



In the chart above, you can see the tools at the top (zoom, resize, reset, wheel zoom) and these tools allows you to interact with chart. You can also look at the multiple chart options (legend, xlabel, ylabel, xgrid, width, height and many other) and various example of charts [here](#).

Chart Example-2: Compare the distribution of sepal length and petal length of IRIS data set using Box plot on notebook

To create this visualization, firstly, I'll import the iris data set using sklearn library. Then, follow the steps as discussed above to visualize chart in ipython notebook.

```
#IRIS Data Set
from sklearn.datasets import load_iris
import pandas as pd
iris = load_iris()
df=pd.DataFrame(iris.data)
df.columns=['petal_width','petal_length','sepal_width','sepal_length']
#Import library
from bokeh.charts import BoxPlot, output_notebook, show
data=df[['petal_length','sepal_length']]
# Output to Notebook
output_notebook()
# create a new line chat with a title and axis labels
p = BoxPlot(data, width=400, height=400)
# show the results
show(p)
```



Chart Example-3: Create a line plot to bokeh server

Prior to plotting visualization to Bokeh server, you need to run it.

If you are using a conda package, you can use run command **bokeh-server** from any directory using command. Else, **python ./bokeh-server** command should work in general. For more detail on this please refer this link "[Deploying Bokeh Server](#)".

There are multiple benefits of Plotting visualization on Bokeh server:

- Plots can be published to larger audience
- Visualize large data set interactively
- Streaming data to automatically updating plots
- Building dashboards and apps

To start plotting on Bokeh server, I have executed the command **bokeh-server** to initialize it followed by the commands used for visualization.

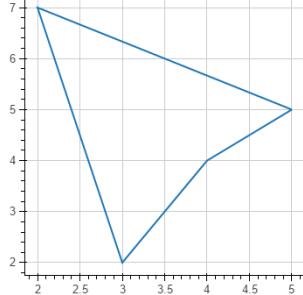
```
Command Prompt - bokeh-server
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Analytics\Udhyga>E:\>bokeh-server
Bokeh Server Configuration
=====
python version : 2.7.8
bokeh version : 0.9.0
listening : 127.0.0.1:5006
backend : memory
python options : debug:OFF, verbose:OFF, filter_logs:OFF, multi_user:OFF
js options : split_js:OFF, debug_js:OFF
```

```
from bokeh.plotting import figure, output_server, show
output_server("line")
p = figure(plot_width=400, plot_height=400)
# add a line renderer
p.line([5, 2, 3, 4, 5], [5, 7, 2, 4, 5], line_width=2)
show(p)
```

localhost:5006/bokeh/doc/3ce3cd91-1419-4f91-9f58-5e4b398be59b/455

link to this



Plotting

Plotting is an *intermediate-level* interface that is centered around composing visual glyphs. Here, you create a visualization by combining various [visual elements](#) (dot, circles, line, patch & many others) and [tools](#) (hover tool, zoom, Save, reset and others).

Bokeh plots created using the [bokeh.plotting](#) interface comes with a default set of tools and visual styles. For plotting, follow the below steps:

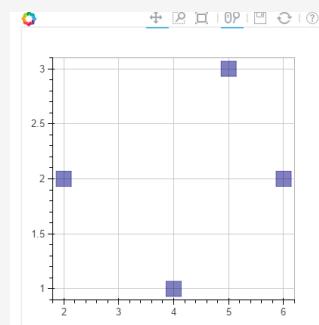
1. Import library, methods or functions
2. Select the output mode (notebook, web browser, server)
3. Activate a figure (similar like matplotlib)
4. Perform subsequent plotting operations, it will affect the generated figure.
5. Visualize it

To understand these steps better, let me demonstrate these steps using examples below:

Plot Example-1: Create a scatter square mark on XY frame of notebook

```
from bokeh.plotting import figure, output_notebook, show

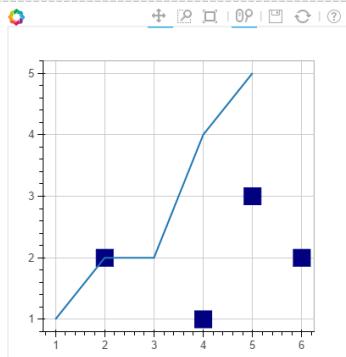
# output to notebook
output_notebook()
p = figure(plot_width=400, plot_height=400)
# add square with a size, color, and alpha
p.square([2, 5, 6, 4], [2, 3, 2, 1, 2], size=20, color="navy")
# show the results
show(p)
```



Similarly, you can create various other plots like line, wedges & arc, ovals, images, patches and many others, refer this [link](#) to see various example.

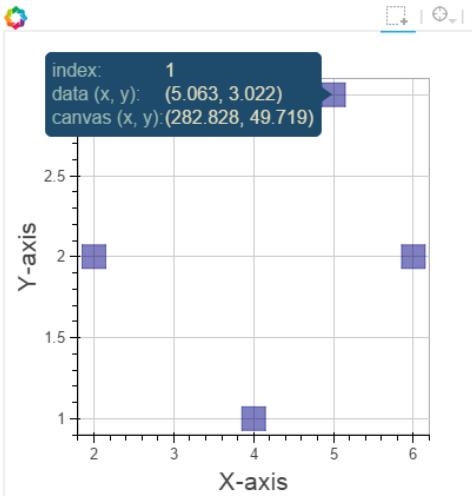
Plot Example-2: Combine two visual elements in a plot

```
from bokeh.plotting import figure, output_notebook, show
# output to notebook
output_notebook()
p = figure(plot_width=400, plot_height=400)
# add square with a size, color, and alpha
p.square([2, 5, 6, 4], [2, 3, 2, 1, 2], size=20, color="navy")
p.line([1, 2, 3, 4, 5], [1, 2, 2, 4, 5], line_width=2) #added a line plot to existing
figure
# show the results
show(p)
```



Plot Example-3: Add a hover tool and axis labels to above plot

```
from bokeh.plotting import figure, output_notebook, show
from bokeh.models import HoverTool, BoxSelectTool #For enabling tools
# output to notebook
output_notebook()
#Add tools
TOOLS = [BoxSelectTool(), HoverTool()]
p = figure(plot_width=400, plot_height=400, tools=TOOLS)
# add a square with a size, color, and alpha
p.square([2, 5, 6, 4], [2, 3, 2, 1, 2], size=20, color="navy", alpha=0.5)
#Visual Elements
p.xaxis.axis_label = "X-axis"
p.yaxis.axis_label = "Y-axis"
# show the results
show(p)
```



Plot Example-4: Plot map of India using latitude and longitude data for boundaries

Note: I have data for polygon of latitude and longitude for boundaries of India in a csv format. I will use that for plotting.

Here, we will go with patch plotting, let's look at the commands below:

```
#Import libraries
import pandas as pd
from bokeh.plotting import figure, show, output_notebook
#Import Latitude and lanogitude co-ordinates
India=pd.read_csv('E:/India.csv')
del India['ID']
India.index=['IN0','IN1','IN2','IN3','IN4','IN5']
#Convert string values to float as co-ordinates in dataframe are string
for j in range(0,len(India)):
    a = India['lats'][j]
    India['lats'][j] = [float(i) for i in a[1:len(a)-1].split(",")]
for j in range(0,len(India)):
    a = India['lons'][j]
    India['lons'][j] = [float(i) for i in a[1:len(a)-1].split(",")]
# Output option
output_notebook()
# Create your plot
p = figure(plot_height=400, plot_width=400, toolbar_location="right",x_axis_type=None
, y_axis_type=None)
p.patches(xs=India['lons'], ys=India['lats'], fill_color="white",line_color="black",
line_width=0.5)
#Visualize your chart
show(p)
```

Python For Data Science Cheat Sheet

Bokeh

Learn Bokeh [Interactively](#) at www.DataCamp.com, taught by Bryan Van de Ven, core contributor

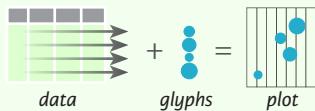


Plotting With Bokeh

The Python interactive visualization library **Bokeh** enables high-performance visual presentation of large datasets in modern web browsers.



Bokeh's mid-level general purpose `bokeh.plotting` interface is centered around two main components: data and glyphs.



The basic steps to creating plots with the `bokeh.plotting` interface are:

1. Prepare some data:
Python lists, NumPy arrays, Pandas DataFrames and other sequences of values
2. Create a new plot
3. Add renderers for your data, with visual customizations
4. Specify where to generate the output
5. Show or save the results

```
>>> from bokeh.plotting import figure
>>> from bokeh.io import output_file, show
>>> x = [1, 2, 3, 4, 5]           Step 1
>>> y = [6, 7, 2, 4, 5]
>>> p = figure(title="simple line example",      Step 2
              x_axis_label='x',
              y_axis_label='y')
>>> p.line(x, y, legend="Temp.", line_width=2)    Step 3
>>> output_file("lines.html")                   Step 4
>>> show(p)                                     Step 5
```

1 Data

Also see Lists, NumPy & Pandas

Under the hood, your data is converted to Column Data Sources. You can also do this manually:

```
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.DataFrame(np.array([[33.9, 4, 65, 'US'],
                               [32.4, 4, 66, 'Asia'],
                               [21.4, 4, 109, 'Europe']]),
                     columns=['mpg', 'cyl', 'hp', 'origin'],
                     index=['Toyota', 'Fiat', 'Volvo'])

>>> from bokeh.models import ColumnDataSource
>>> cds_df = ColumnDataSource(df)
```

2 Plotting

```
>>> from bokeh.plotting import figure
>>> p1 = figure(plot_width=300, tools='pan,box_zoom')
>>> p2 = figure(plot_width=300, plot_height=300,
               x_range=(0, 8), y_range=(0, 8))
>>> p3 = figure()
```

3 Renderers & Visual Customizations

Glyphs

Scatter Markers

```
>>> p1.circle(np.array([1,2,3]), np.array([3,2,1]),
             fill_color='white')
>>> p2.square(np.array([1.5,3.5,5.5]), [1,4,3],
             color='blue', size=1)
```

Line Glyphs

```
>>> p1.line([1,2,3,4], [3,4,5,6], line_width=2)
>>> p2.multi_line(pd.DataFrame([[1,2,3],[5,6,7]]),
                  pd.DataFrame([[3,4,5],[3,2,1]]),
                  color="blue")
```

Rows & Columns Layout

Rows

```
>>> from bokeh.layouts import row
```

```
>>> layout = row(p1,p2,p3)
```

Columns

```
>>> from bokeh.layouts import column
```

```
>>> layout = column(p1,p2,p3)
```

```
>>> layout = row(column(p1,p2), p3)
```

Grid Layout

```
>>> from bokeh.layouts import gridplot
>>> row1 = [p1,p2]
>>> row2 = [p3]
>>> layout = gridplot([[p1,p2], [p3]])
```

Tabbed Layout

```
>>> from bokeh.models.widgets import Panel, Tabs
>>> tab1 = Panel(child=p1, title="tab1")
>>> tab2 = Panel(child=p2, title="tab2")
>>> layout = Tabs(tabs=[tab1, tab2])
```

Legends

Legend Location

Inside Plot Area
`>>> p.legend.location = 'bottom_left'`

Outside Plot Area

```
>>> r1 = p2.asterisk(np.array([1,2,3]), np.array([3,2,1]))
>>> r2 = p2.line([1,2,3,4], [3,4,5,6])
>>> legend = Legend(items=[("One", [p1, r1]), ("Two", [r2])], location=(0, -30))
>>> p.add_layout(legend, 'right')
```

Linked Plots

Linked Axes

```
>>> p2.x_range = p1.x_range
>>> p2.y_range = p1.y_range
```

Linked Brushing

```
>>> p4 = figure(plot_width = 100, tools='box_select,lasso_select')
>>> p4.circle('mpg', 'cyl', source=cds_df)
>>> p5 = figure(plot_width = 200, tools='box_select,lasso_select')
>>> p5.circle('mpg', 'hp', source=cds_df)
>>> layout = row(p4,p5)
```

Also see Data

Legend Orientation

```
>>> p.legend.orientation = "horizontal"
>>> p.legend.orientation = "vertical"
```

Legend Background & Border

```
>>> p.legend.border_line_color = "navy"
>>> p.legend.background_fill_color = "white"
```

4 Output

Output to HTML File

```
>>> from bokeh.io import output_file, show
>>> output_file('my_bar_chart.html', mode='cdn')
```

Notebook Output

```
>>> from bokeh.io import output_notebook, show
>>> output_notebook()
```

Embedding

Standalone HTML

```
>>> from bokeh.embed import file_html
>>> html = file_html(p, CDN, "my_plot")
```

Components

```
>>> from bokeh.embed import components
>>> script, div = components(p)
```

Statistical Charts With Bokeh

Also see Data

Bokeh's high-level `bokeh.charts` interface is ideal for quickly creating statistical charts

Bar Chart

```
>>> from bokeh.charts import Bar
>>> p = Bar(df, stacked=True, palette=['red','blue'])
```

Box Plot

```
>>> from bokeh.charts import BoxPlot
>>> p = BoxPlot(df, values='vals', label='cyl',
                legend='bottom_right')
```

Histogram

```
>>> from bokeh.charts import Histogram
>>> p = Histogram(df, title='Histogram')
```

Scatter Plot

```
>>> from bokeh.charts import Scatter
>>> p = Scatter(df, x='mpg', y ='hp', marker='square',
                xlabel='Miles Per Gallon',
                ylabel='Horsepower')
```

5 Show or Save Your Plots

```
>>> show(p1)
>>> show(layout)
```

```
>>> save(p1)
>>> save(layout)
```

Customized Glyphs

Selection and Non-Selection Glyphs

```
>>> p = figure(tools='box_select')
>>> p.circle('mpg', 'cyl', source=cds_df,
            selection_color='red',
            nonselection_alpha=0.1)
```

Hover Glyphs

```
>>> hover = HoverTool(tooltips=None, mode='vline')
>>> p3.add_tools(hover)
```

Colormapping

```
>>> color_mapper = CategoricalColorMapper(
            factors=['US', 'Asia', 'Europe'],
            palette=['blue', 'red', 'green'])
>>> p3.circle('mpg', 'cyl', source=cds_df,
            color=dict(field='origin',
            transform=color_mapper),
            legend='Origin'))
```

Also see Data



Python For Data Science Cheat Sheet

Matplotlib

Learn Python Interactively at www.DataCamp.com



Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.



1 Prepare The Data

Also see [Lists & NumPy](#)

1D Data

```
>>> import numpy as np  
>>> x = np.linspace(0, 10, 100)  
>>> y = np.cos(x)  
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))  
>>> data2 = 3 * np.random.random((10, 10))  
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]  
>>> U = -1 - X**2 + Y  
>>> V = 1 + X - Y**2  
>>> from matplotlib.cbook import get_sample_data  
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

2 Create Plot

```
>>> import matplotlib.pyplot as plt
```

Figure

```
>>> fig = plt.figure()  
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

Axes

All plotting is done with respect to an Axes. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()  
>>> ax1 = fig.add_subplot(221) # row-col-num  
>>> ax3 = fig.add_subplot(212)  
>>> fig3, axes = plt.subplots(nrows=2, ncols=2)  
>>> fig4, axes2 = plt.subplots(ncols=3)
```

3 Plotting Routines

1D Data

```
>>> lines = ax.plot(x,y)  
>>> ax.scatter(x,y)  
>>> axes[0,0].bar([1,2,3],[3,4,5])  
>>> axes[1,0].barh([0.5,1,2.5],[0,1,2])  
>>> axes[1,1].axhline(0.45)  
>>> axes[0,1].axvline(0.65)  
>>> ax.fill(x,y,color='blue')  
>>> ax.fill_between(x,y,color='yellow')
```

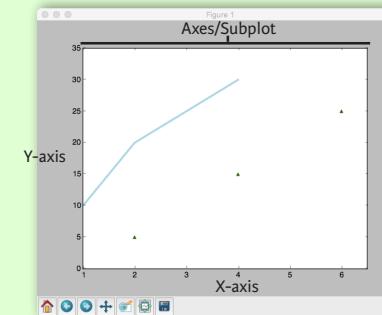
2D Data or Images

```
>>> fig, ax = plt.subplots()  
>>> im = ax.imshow(img,  
                  cmap='gist_earth',  
                  interpolation='nearest',  
                  vmin=-2,  
                  vmax=2)
```

Colormapped or RGB arrays

Plot Anatomy & Workflow

Plot Anatomy



Figure

Workflow

The basic steps to creating plots with matplotlib are:

- 1 Prepare data
- 2 Create plot
- 3 Plot
- 4 Customize plot
- 5 Save plot
- 6 Show plot

```
>>> import matplotlib.pyplot as plt  
>>> x = [1,2,3,4]  
>>> y = [10,20,25,30] Step 1  
>>> fig = plt.figure() Step 2  
>>> ax = fig.add_subplot(111) Step 3  
>>> ax.plot(x, y, color='lightblue', linewidth=3) Step 3.4  
>>> ax.scatter([2,4,6],  
             [5,15,25],  
             color='darkgreen',  
             marker='^')  
>>> ax.set_xlim(1, 6.5)  
>>> plt.savefig('foo.png')  
>>> plt.show() Step 6
```

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x, x**2, x, x**3)  
>>> ax.plot(x, y, alpha = 0.4)  
>>> ax.plot(x, y, c='k')  
>>> fig.colorbar(im, orientation='horizontal')  
>>> im = ax.imshow(img,  
                  cmap='seismic')
```

Markers

```
>>> fig, ax = plt.subplots()  
>>> ax.scatter(x,y,marker=".")  
>>> ax.plot(x,y,marker="o")
```

Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)  
>>> plt.plot(x,y,ls='solid')  
>>> plt.plot(x,y,ls='--')  
>>> plt.plot(x,y,'-.',x**2,y**2,'-.')  
>>> plt.setp(lines,color='r',linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(1,-2.1,  
           'Example Graph',  
           style='italic')  
>>> ax.annotate("Sine",  
               xy=(8, 0),  
               xycoords='data',  
               xytext=(10.5, 0),  
               textcoords='data',  
               arrowprops=dict(arrowstyle="->",  
                               connectionstyle="arc3"),)
```

Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)  
>>> axes[1,1].quiver(y,z)  
>>> axes[0,1].streamplot(X,Y,U,V)
```

Add an arrow to the axes
Plot a 2D field of arrows
Plot 2D vector fields

Data Distributions

```
>>> ax1.hist(y)  
>>> ax3.boxplot(y)  
>>> ax3.violinplot(z)
```

Plot a histogram
Make a box and whisker plot
Make a violin plot

Mathtext

```
>>> plt.title(r'$\sigma_i=15$', fontsize=20)
```

Limits, Legends & Layouts

```
>>> ax.margins(x=0.0,y=0.1)  
>>> ax.axis('equal')  
>>> ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])  
>>> ax.set_xlim(0,10.5)
```

Legends

```
>>> ax.set(title='An Example Axes',  
           ylabel='Y-Axis',  
           xlabel='X-Axis')  
>>> ax.legend(loc='best')
```

Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),  
                  ticklabels=[3,100,-12,"foo"])  
>>> ax.tick_params(axis='y',  
                           direction='inout',  
                           length=10)
```

Subplot Spacing

```
>>> fig3.subplots_adjust(wspace=0.5,  
                           hspace=0.3,  
                           left=0.125,  
                           right=0.9,  
                           top=0.9,  
                           bottom=0.1)
```

```
>>> fig.tight_layout()
```

Axis Spines

```
>>> ax1.spines['top'].set_visible(False)  
>>> ax1.spines['bottom'].set_position(('outward',10))
```

Add padding to a plot
Set the aspect ratio of the plot to 1
Set limits for x-and y-axis
Set limits for x-axis

Set a title and x-and y-axis labels

No overlapping plot elements

Manually set x-ticks

Make y-ticks longer and go in and out

Adjust the spacing between subplots

Fit subplot(s) in to the figure area

Make the top axis line for a plot invisible

Move the bottom axis line outward

5 Save Plot

Save figures

```
>>> plt.savefig('foo.png')
```

Save transparent figures

```
>>> plt.savefig('foo.png', transparent=True)
```

6 Show Plot

```
>>> plt.show()
```

Close & Clear

```
>>> plt.cla()  
>>> plt.clf()  
>>> plt.close()
```

Clear an axis
Clear the entire figure
Close a window



Python For Data Science Cheat Sheet

Seaborn

Learn Data Science interactively at www.DataCamp.com



Statistical Data Visualization With Seaborn

The Python visualization library **Seaborn** is based on `matplotlib` and provides a high-level interface for drawing attractive statistical graphics.

Make use of the following aliases to import the libraries:

```
>>> import matplotlib.pyplot as plt  
>>> import seaborn as sns
```

The basic steps to creating plots with Seaborn are:

1. Prepare some data
2. Control figure aesthetics
3. Plot with Seaborn
4. Further customize your plot

```
>>> import matplotlib.pyplot as plt  
>>> import seaborn as sns  
>>> tips = sns.load_dataset("tips")  
>>> sns.set_style("whitegrid")  
Step 1  
>>> g = sns.lmplot(x="tip",  
y="total_bill",  
data=tips,  
aspect=2)  
Step 2  
>>> g.set_axis_labels("Tip", "Total bill(USD)")  
set(xlim=(0,10), ylim=(0,100))  
Step 3  
>>> plt.title("title")  
Step 4  
>>> plt.show(g)  
Step 5
```

1) Data

Also see [Lists, NumPy & Pandas](#)

```
>>> import pandas as pd  
>>> import numpy as np  
>>> uniform_data = np.random.rand(10, 12)  
>>> data = pd.DataFrame({'x':np.arange(1,101),  
'y':np.random.normal(0,4,100)})
```

Seaborn also offers built-in data sets:

```
>>> titanic = sns.load_dataset("titanic")  
>>> iris = sns.load_dataset("iris")
```

2) Figure Aesthetics

Seaborn styles

```
>>> sns.set()  
>>> sns.set_style("whitegrid")  
>>> sns.set_style("ticks",  
{"xtick.major.size":8,  
"ytick.major.size":8})  
>>> sns.axes_style("whitegrid")
```

(Re)set the seaborn default
Set the matplotlib parameters
Set the matplotlib parameters
Return a dict of params or use with
with to temporarily set the style

Context Functions

```
>>> sns.set_context("talk")  
>>> sns.set_context("notebook",  
font_scale=1.5,  
rc={"lines.linewidth":2.5})
```

Color Palette

```
>>> sns.set_palette("husl",3)  
>>> sns.color_palette("husl")  
>>> flatui = ["#9b59b6","#3498db","#95a5e6","#e74c3c","#34495e","#2ecc71"]  
>>> sns.set_palette(flatui)
```

3) Plotting With Seaborn

Axis Grids

```
>>> g = sns.FacetGrid(titanic,  
col="survived",  
row="sex")  
>>> g.map(plt.hist,"age")  
>>> sns.factorplot(x="pclass",  
y="survived",  
hue="sex",  
data=titanic)  
>>> sns.lmplot(x="sepal_width",  
y="sepal_length",  
hue="species",  
data=iris)
```

Subplot grid for plotting conditional relationships

Draw a categorical plot onto a Facetgrid

Plot data and regression model fits across a FacetGrid

```
>>> h = sns.PairGrid(iris)  
>>> h = h.map(plt.scatter)  
>>> sns.pairplot(iris)  
>>> i = sns.JointGrid(x="x",  
y="y",  
data=data)  
>>> i = i.plot(sns.regplot,  
sns.distplot)  
>>> sns.jointplot("sepal_length",  
"sepal_width",  
data=iris,  
kind='kde')
```

Subplot grid for plotting pairwise relationships
Plot pairwise bivariate distributions
Grid for bivariate plot with marginal univariate plots

Plot bivariate distribution

Categorical Plots

Scatterplot

```
>>> sns.stripplot(x="species",  
y="petal_length",  
data=iris)  
>>> sns.swarmplot(x="species",  
y="petal_length",  
data=iris)
```

Bar Chart

```
>>> sns.barplot(x="sex",  
y="survived",  
hue="class",  
data=titanic)
```

Count Plot

```
>>> sns.countplot(x="deck",  
data=titanic,  
palette="Greens_d")
```

Point Plot

```
>>> sns.pointplot(x="class",  
y="survived",  
hue="sex",  
data=titanic,  
palette={"male":"g",  
"female":"m"},  
markers=["^","o"],  
linestyles=["-","--"])
```

Boxplot

```
>>> sns.boxplot(x="alive",  
y="age",  
hue="adult_male",  
data=titanic)
```

Violinplot

```
>>> sns.violinplot(x="age",  
y="sex",  
hue="survived",  
data=titanic)
```

Scatterplot with one categorical variable

Categorical scatterplot with non-overlapping points

Show point estimates and confidence intervals with scatterplot glyphs

Show count of observations

Show point estimates and confidence intervals as rectangular bars

Boxplot

Boxplot with wide-form data

Violin plot

Regression Plots

```
>>> sns.regplot(x="sepal_width",  
y="sepal_length",  
data=iris,  
ax=ax)
```

Plot data and a linear regression model fit

Distribution Plots

```
>>> plot = sns.distplot(data.y,  
kde=False,  
color="b")
```

Plot univariate distribution

Matrix Plots

```
>>> sns.heatmap(uniform_data,vmin=0,vmax=1)
```

Heatmap

4) Further Customizations

Also see [Matplotlib](#)

Axisgrid Objects

```
>>> g.despine(left=True)  
>>> g.set_ylabels("Survived")  
>>> g.set_xticklabels(rotation=45)  
>>> g.set_axis_labels("Survived",  
"Sex")  
>>> h.set(xlim=(0,5),  
ylim=(0,5),  
xticks=[0,2.5,5],  
yticks=[0,2.5,5])
```

Remove left spine
Set the labels of the y-axis
Set the tick labels for x
Set the axis labels

Set the limit and ticks of the x-and y-axis

Plot

```
>>> plt.title("A Title")  
>>> plt.ylabel("Survived")  
>>> plt.xlabel("Sex")  
>>> plt.ylim(0,100)  
>>> plt.xlim(0,10)  
>>> plt.setp(ax,yticks=[0,5])  
>>> plt.tight_layout()
```

Add plot title
Adjust the label of the y-axis
Adjust the label of the x-axis
Adjust the limits of the y-axis
Adjust the limits of the x-axis
Adjust a plot property
Adjust subplot params

5) Show or Save Plot

Also see [Matplotlib](#)

```
>>> plt.show()  
>>> plt.savefig("foo.png")  
>>> plt.savefig("foo.png",  
transparent=True)
```

Show the plot
Save the plot as a figure
Save transparent figure

Close & Clear

```
>>> plt.cla()  
>>> plt.clf()  
>>> plt.close()
```

Clear an axis
Clear an entire figure
Close a window



Python Seaborn Tutorial For Beginners

This Seaborn tutorial introduces you to the basics of statistical data visualization

Seaborn: Python's Statistical Data Visualization Library

One of the best but also more challenging ways to get your insights across is to visualize them: that way, you can more easily identify patterns, grasp difficult concepts or draw the attention to key elements. When you're using Python for data science, you'll most probably will have already used [Matplotlib](#), a 2D plotting library that allows you to create publication-quality figures. Another complimentary package that is based on this data visualization library is [Seaborn](#), which provides a high-level interface to draw statistical graphics.

Today's post will cover some of the most frequently asked questions users had while they started out working with the Seaborn library. How many of the following questions can you answer correctly?

1. [Seaborn vs Matplotlib?](#)
2. [How To Load Data To Construct Seaborn Plots](#)
 - o Loading A Built-in Data Set
 - o Loading Your Pandas DataFrame
3. [How To Show Seaborn Plots](#)
4. [How To Use Seaborn With Matplotlib Defaults](#)
5. [How To Use Seaborn's Colors As A colormap in Matplotlib?](#)
6. [How To Scale Seaborn Plots For Other Context](#)
7. [How To Temporarily Set The Plot Style](#)
8. [How To Set The Figure Size in Seaborn](#)
9. [How To Rotate Label Text](#)
10. [How To Set `xlim` or `ylim` in Seaborn](#)
11. [How To Set Log Scale](#)

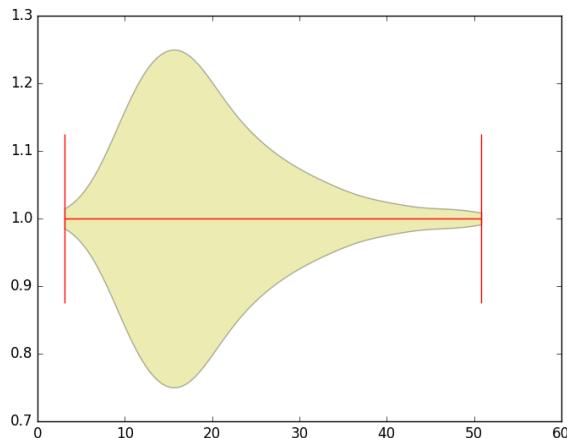
12. How To Add A Title

Seaborn vs Matplotlib

As you have just read, Seaborn is complimentary to Matplotlib and it specifically targets statistical data visualization. But it goes even further than that: Seaborn extends Matplotlib and that's why it can address the two biggest frustrations of working with Matplotlib. Or, as Michael Waskom says in the "[introduction to Seaborn](#)": "If matplotlib "tries to make easy things easy and hard things possible", seaborn tries to make a well-defined set of hard things easy too."

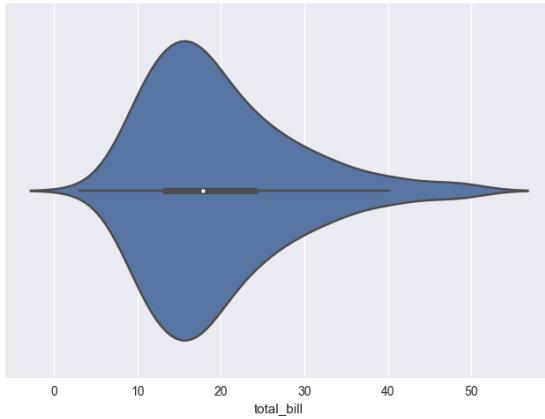
One of these hard things or frustrations had to do with the default Matplotlib parameters. Seaborn works with different parameters, which undoubtedly speaks to those users that don't use the default looks of the Matplotlib plots.

```
# Import the necessary libraries
import matplotlib.pyplot as plt
import pandas as pd
# Initialize Figure and Axes object
fig, ax = plt.subplots()
# Load in data
tips = pd.read_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/master/tips.csv")
# Create violinplot
ax.violinplot(tips["total_bill"], vert=False)
# Show the plot
plt.show()
```



```
# Import the necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
# Load the data
```

```
tips = sns.load_dataset("tips")
# Create violinplot
sns.violinplot(x = "total_bill", data=tips)
# Show the plot
plt.show()
```



The Matplotlib defaults that usually don't speak to users are the colors, the tick marks on the upper and right axes, the style,...

The examples above also makes another frustration of users more apparent: the fact that working with DataFrames doesn't go quite as smoothly with Matplotlib, which can be annoying if you're doing exploratory analysis with Pandas. And that's exactly what Seaborn addresses: the plotting functions operate on DataFrames and arrays that contain a whole dataset.

As Seaborn complements and extends Matplotlib, the learning curve is quite gradual: if you know Matplotlib, you'll already have most of Seaborn down.

How To Load Data To Construct Seaborn Plots

When you're working with Seaborn, you can either use one of the built-in data sets that the library itself has to offer or you can load a Pandas DataFrame. In this section, you'll see how to do both.

Loading A Built-in Seaborn Data Set

To start working with a built-in Seaborn data set, you can make use of the `load_dataset()` function. To get an overview or inspect all data sets that this function opens up to you, go [here](#). Check out the following example to see how the `load_dataset()` function works:

```
# Import necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt
# Load iris data
iris = sns.load_dataset("iris")
# Construct iris plot
sns.swarmplot(x="species", y="petal_length", data=iris)
# Show plot
plt.show()
```



As an anecdote, it might be interesting for you to know that the import convention `sns` comes from the fictional character Samuel Norman “Sam” Seaborn on the television serial drama *The West Wing*. It’s an inside joke by the core developer of Seaborn, namely, Michael Waskom.

Loading Your Pandas DataFrame Getting Your Data

Of course, most of the fun in visualizing data lies in the fact that you would be working with your own data and not the built-in data sets of the Seaborn library. Seaborn works best with Pandas DataFrames and arrays that contain a whole data set.

Remember that DataFrames are a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a

DataFrame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc. Specifically for Python, DataFrames come with the Pandas library, and they are defined as a two-dimensional labeled data structures with columns of potentially different types.

The reason why Seaborn is so great with DataFrames is, for example, because labels from DataFrames are automatically propagated to plots or other data structures, as you saw in the first example of this tutorial, where you plotted a violinplot with Seaborn. There, you saw that the x-axis had a legend `total_bill`, while this was not the case with the Matplotlib plot. This already takes a lot of work away from you.

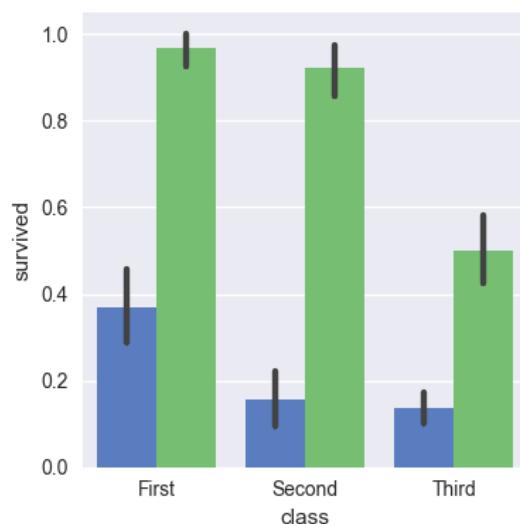
But that doesn't mean that all the work is done -quite the opposite. In many cases, you'll need to still manipulate your Pandas DataFrame so that the plot will render correctly. If you want to know more, check out DataCamp's [Pandas Tutorial on DataFrames in Python](#) or the [Pandas Foundations](#) course.

How To Show Seaborn Plots

Matplotlib still underlies Seaborn, which means that the anatomy of the plot is still the same and that you'll need to use `plt.show()` to make the image appear to you. You might have already seen this from the previous example in this tutorial. In any case, here's another example where the `show()` function is used to show the plot:

```
# Import necessarily libraries
import matplotlib.pyplot as plt
import seaborn as sns
# Load data
titanic = sns.load_dataset("titanic")
# Set up a factorplot
g = sns.factorplot("class", "survived", "sex", data=titanic, kind="bar", palette="muted", legend=False)

# Show plot
plt.show()
```



Note that in the code chunk above you work with a built-in Seaborn data set and you create a factorplot with it. A factorplot is a categorical plot, which in this case is a bar plot. That's because you have set the `kind` argument to `"bar"`. Also, you set which colors should be displayed with the `palette` argument and that you set the `legend` to `False`.

How To Use Seaborn With Matplotlib Defaults

As you read in the introduction, the Matplotlib defaults are something that users might not find as pleasing than the Seaborn defaults. However, there are also many questions in the opposite direction, namely, those use Seaborn and that want to plot with Matplotlib defaults.

Before, you could solve this question by importing the `apionly` module from the Seaborn package. This is now deprecated (since July 2017). The default style is no longer applied when Seaborn is imported, so you'll need to explicitly call `set()` or one or more of `set_style()`, `set_context()`, and `set_palette()` to get either Seaborn or Matplotlib defaults for plotting.

```
# Import Matplotlib
import matplotlib.pyplot as plt
# Check the available styles
plt.style.available
# Use Matplotlib defaults
plt.style.use("classic")
```

How To Use Seaborn's Colors As A colormap in Matplotlib?

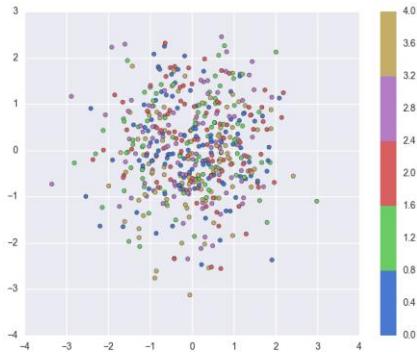
Besides using Seaborn with Matplotlib defaults, there's also questions on how to bring in Seaborn colors into Matplotlib plots. You can make use of `color_palette()` to define a color map that you want to be using and the number of colors with the argument `n_colors`. In this case, the example will assume that there are 5 labels assigned to the data points that are defined in `data1` and `data2`, so that's why you pass `5` to this argument and you also make a list with length equal to `N` where 5 integers vary in the variable `colors`.

```
# Import the necessary libraries
import seaborn as sns
```

```

import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import ListedColormap
# Define a variable N
N = 500
# Construct the colormap
current_palette = sns.color_palette("muted", n_colors=5)
cmap = ListedColormap(sns.color_palette(current_palette).as_hex())
# Initialize the data
data1 = np.random.randn(N)
data2 = np.random.randn(N)
# Assume that there are 5 possible labels
colors = np.random.randint(0,5,N)
# Create a scatter plot
plt.scatter(data1, data2, c=colors, cmap=cmap)
# Add a color bar
plt.colorbar()
# Show the plot
plt.show()

```



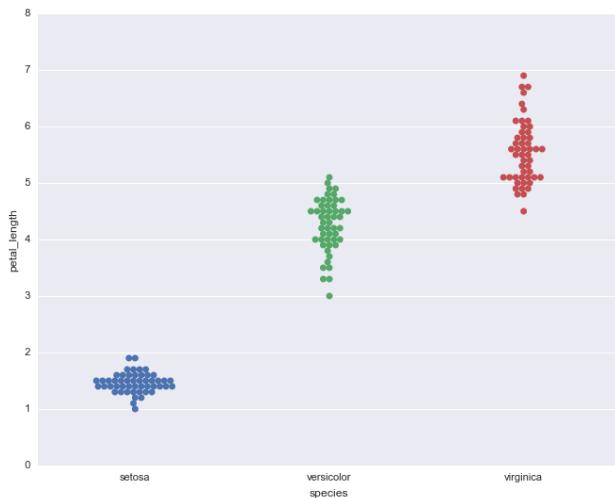
How To Scale Seaborn Plots For Other Contexts

If you need your plots for talks, posters, on paper or in notebooks, you might want to have larger or smaller plots. Seaborn has got you covered on this. You can make use of `set_context()` to control the plot elements:

```

# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
# Reset default params
sns.set()
# Set context to "paper"
sns.set_context("paper")
# Load iris data
iris = sns.load_dataset("iris")
# Construct iris plot
sns.swarmplot(x="species", y="petal_length", data=iris)
# Show plot
plt.show()

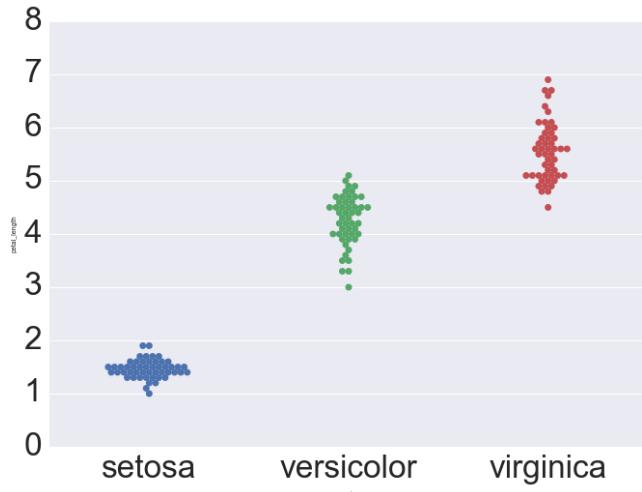
```



The four predefined contexts are "paper", "notebook", "talk" and "poster". **Tip:** try changing the context in the DataCamp Light chunk above to another context to study the effect of the contexts on the plot.

You can also pass more arguments to `set_context()` to scale more plot elements, such as `font_scale` or more parameter mappings that can override the values that are preset in the Seaborn context dictionaries. In the following code chunk, you overwrite the values that are set for the parameters `font.size` and `axes.labelsize`:

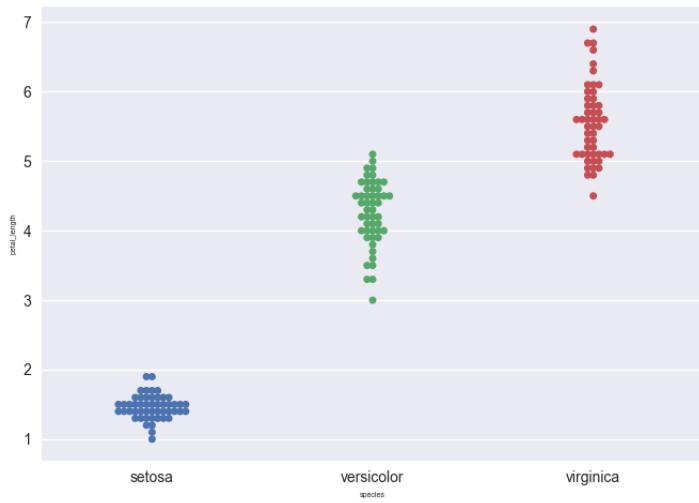
```
# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
# Set context to ``paper``
sns.set_context("paper", font_scale=3, rc={"font.size":8,"axes.labelsize":5})
# Load iris data
iris = sns.load_dataset("iris")
# Construct iris plot
sns.swarmplot(x="species", y="petal_length", data=iris)
# Show plot
plt.show()
```



Note that in the first code chunk, you have first done a reset to get the default Seaborn parameters back. You did this by calling `set()`. This is extremely handy if you have experimented with setting other parameters before, such as the plot style.

Additionally, it's good to keep in mind that you can use the higher-level `set()` function instead of `set_context()` to adjust other plot elements:

```
# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
# Reset default params
sns.set(rc={"font.size":8,"axes.labelsize":5})
# Load iris data
iris = sns.load_dataset("iris")
# Construct iris plot
sns.swarmplot(x="species", y="petal_length", data=iris)
# Show plot
plt.show()
```



One of the hardest things about data visualizations is customizing the graphs further until they meet your expectations and this stays the same when you’re working with Seaborn. That’s why it’s good to keep in mind the anatomy of the Matplotlib plot and also what this means for the Seaborn library.

As for Seaborn, you have two types of functions: axes-level functions and figure-level functions. The ones that operate on the Axes level are, for example, `regplot()`, `boxplot()`, `kdeplot()`, ..., while the functions that operate on the Figure level are `lmplot()`, `factorplot()`, `jointplot()` and a couple others.

This means that the first group is identified by taking an explicit `ax` argument and returning an `Axes` object, while the second group of functions create plots that potentially include `Axes` which are always organized in a “meaningful” way. The Figure-level functions will therefore need to have total control over the figure so you won’t be able to plot an `lmplot` onto one that already exists. When you call the Figure-level functions, you always initialize a figure and set it up for the specific plot it’s drawing.

You can easily see this when you make a boxplot and an `lmplot`, for example:

```
>>> sns.boxplot(x="total_bill", data=tips)
```

```
<matplotlib.axes._subplots.AxesSubplot object at 0x117e8da20>

>>> sns.lmplot('x', 'y', data, size=7, truncate=True, scatter_kws={"s": 100})
<seaborn.axisgrid.FacetGrid object at 0x11fa03438>
```

However, you see that, once you've called `lmplot()`, it returns an object of the type `FacetGrid`. This object has some methods for operating on the resulting plot that know a bit about the structure of the plot. It also exposes the underlying figure and array of axes at the `FacetGrid.fig` and `FacetGrid.axes` arguments.

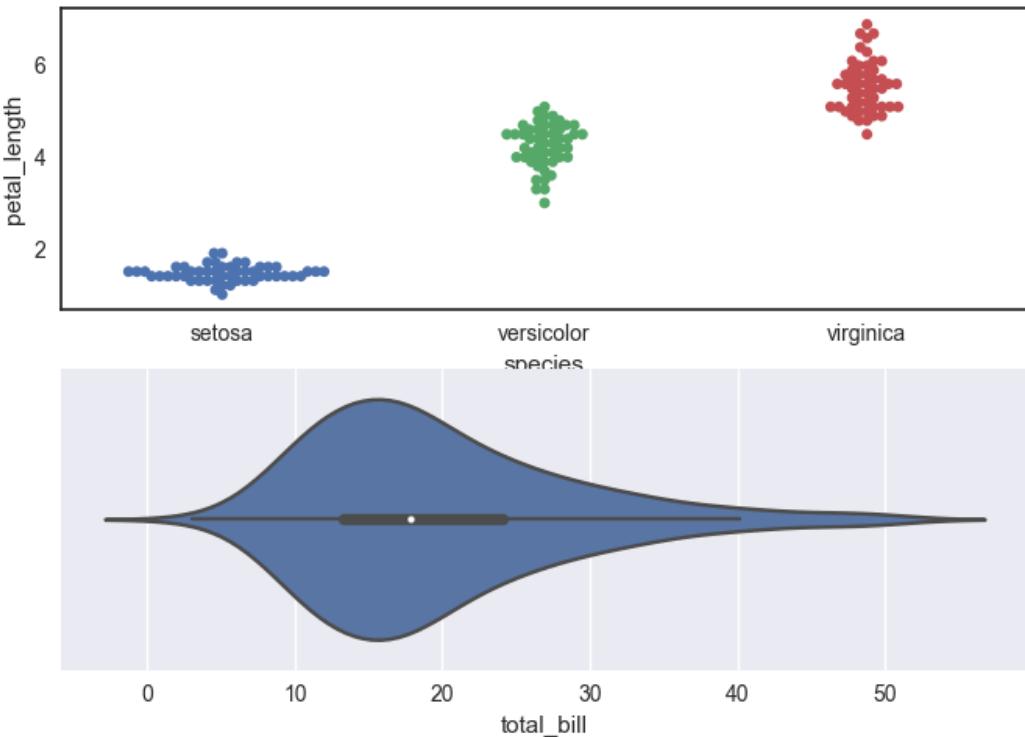
When you're customizing your plots, this means that you will prefer to make customizations to your regression plot that you constructed with `regplot()` on `Axes` level, while you will make customizations for `lmplot()` on Figure level.

Let's see how this works in practice by covering some of the following, most frequently asked questions:

How To Temporarily Set The Plot Style

You can use `axes_style()` in a `with` statement to temporarily set the plot style. This, in addition to the use of `plt.subplot()`, will allow you to make figures that have differently-styled axes, like in the example below:

```
# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
# Load data
iris = sns.load_dataset("iris")
tips = sns.load_dataset("tips")
# Set axes style to white for first subplot
with sns.axes_style("white"):
    plt.subplot(211)
    sns.swarmplot(x="species", y="petal_length", data=iris)
# Initialize second subplot
plt.subplot(212)
# Plot violinplot
sns.violinplot(x = "total_bill", data=tips)
# Show the plot
plt.show()
```



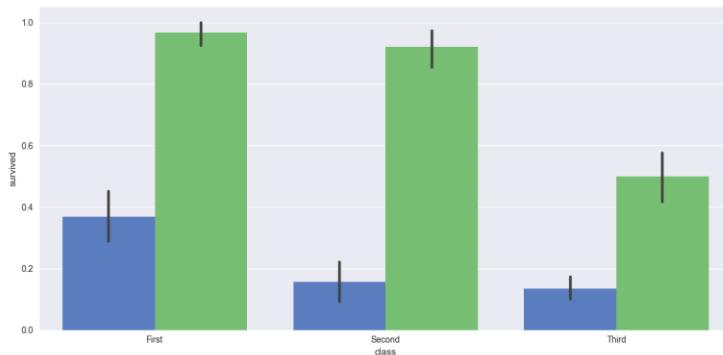
How To Set The Figure Size in Seaborn

For axes level functions, you can make use of the `plt.subplots()` function to which you pass the `figsize` argument.

```
# Import necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt
# Initialize Figure and Axes object
fig, ax = plt.subplots(figsize=(10,4))
# Load in the data
iris = sns.load_dataset("iris")
# Create swarmplot
sns.swarmplot(x="species", y="petal_length", data=iris, ax=ax)
# Show plot
plt.show()
```



```
# Import the libraries
import matplotlib.pyplot as plt
import seaborn as sns
# Load data
titanic = sns.load_dataset("titanic")
# Set up a factorplot
g = sns.factorplot("class", "survived", "sex", data=titanic, kind="bar", size=6, aspect=2, palette="muted",
legend=False)
# Show plot
plt.show()
```



How To Rotate Label Text in Seaborn

To rotate the label text in a Seaborn plot, you will need to work on the Figure level. Note that in the code chunk below, you make use of one of the FacetGrid methods, namely,

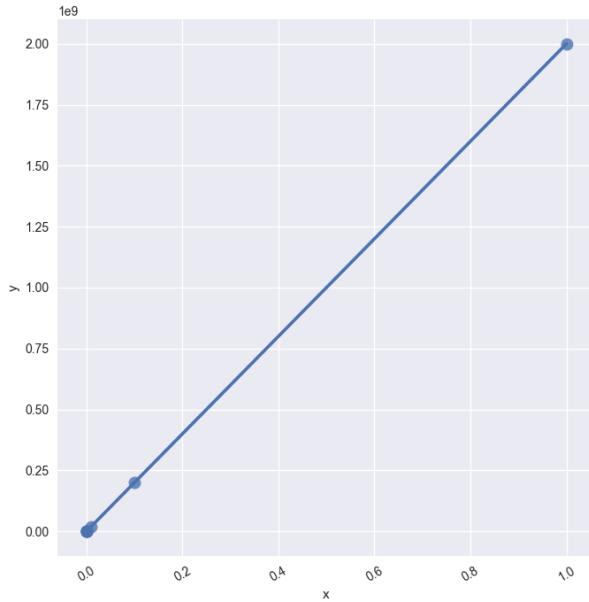
`set_xticklabels`, to rotate the text label:

```
# Import the necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
# Initialize the data
```

```

x = 10 ** np.arange(1, 10)
y = x * 2
data = pd.DataFrame(data={'x': x, 'y': y})
# Create an lmplot
grid = sns.lmplot('x', 'y', data, size=7, truncate=True, scatter_kws={"s": 100})
# Rotate the labels on x-axis
grid.set_xticklabels(rotation=30)
# Show the plot
plt.show()

```



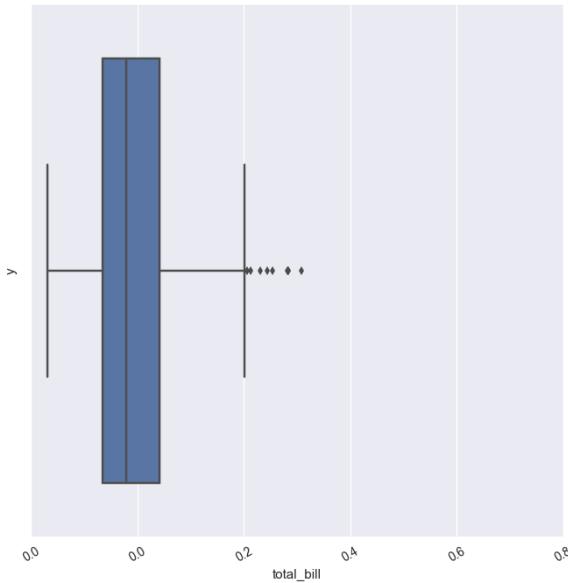
How To Set `xlim` or `ylim` in Seaborn

For a boxplot, which works at the Axes level, you'll need to make sure to assign your boxplot to a variable `ax`, which will be a `matplotlib.axes._subplots.AxesSubplot` object, as you saw above. With the object at Axes level, you can make use of the `set()` function to set `xlim`, `ylim`,... Just like in the following example:

```

# Import necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt
# Load the data
tips = sns.load_dataset("tips")
# Create the boxplot
ax = sns.boxplot(x="total_bill", data=tips)
# Set the `xlim`
ax.set(xlim=(0, 100))
# Show the plot
plt.show()

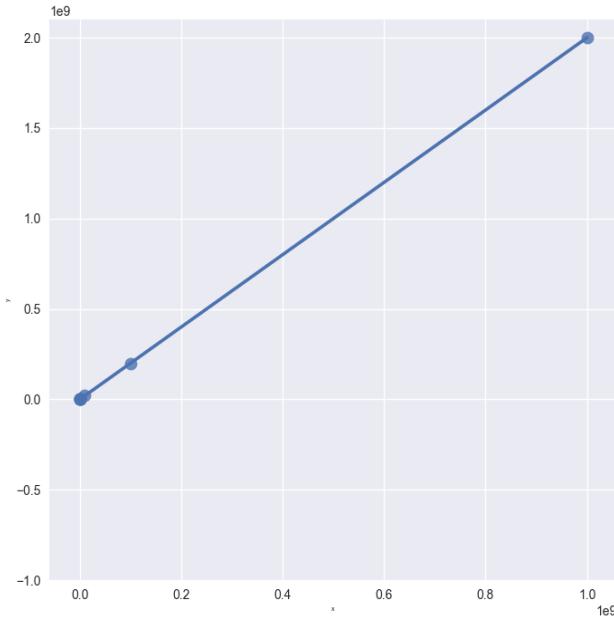
```



Note that alternatively, you could have also used `ax.set_xlim(10,100)` to limit the x-axis.

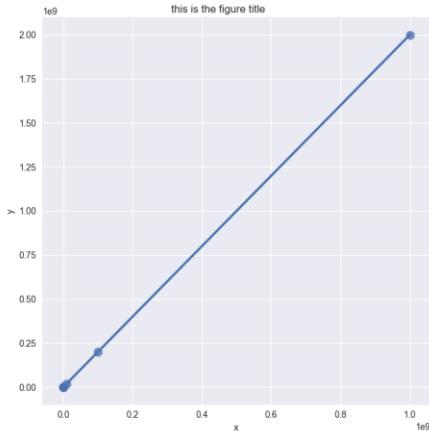
Now, for functions at Figure-level, you can access the `Axes` object with the help of the `axes` argument. Let's see how you can use the `ax` argument to your advantage to set the `xlim` and `ylim` properties:

```
# Import the necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
# Initialize the data
x = 10 ** np.arange(1, 10)
y = x * 2
data = pd.DataFrame(data={'x': x, 'y': y})
# Create lmplot
lm = sns.lmplot('x', 'y', data, size=7, truncate=True, scatter_kws={"s": 100})
# Get hold of the `Axes` objects
axes = lm.ax
# Tweak the `Axes` properties
axes.set_xlim(-10000000000.)
# Show the plot
plt.show()
```



Likewise, `FacetGrid` exposes the underlying `Figure` with the help of the `fig` argument.

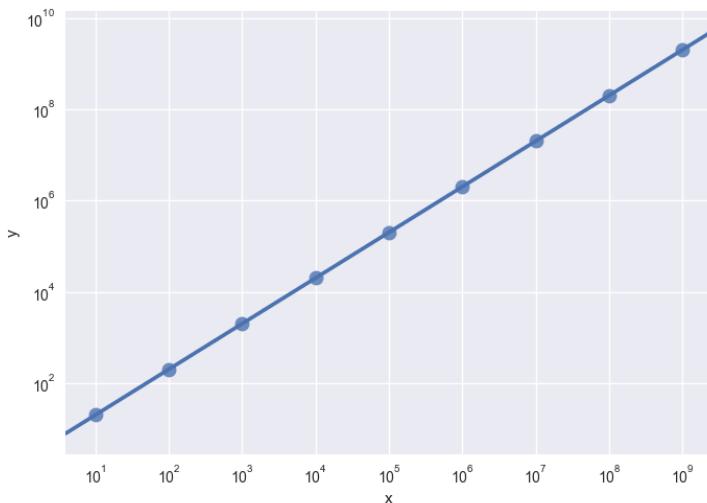
```
# Import the necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
# Initialize the data
x = 10 ** np.arange(1, 10)
y = x * 2
data = pd.DataFrame(data={'x': x, 'y': y})
# Create lmplot
lm = sns.lmplot('x', 'y', data, size=7, truncate=True, scatter_kws={"s": 100})
# Access the Figure
fig = lm.fig
# Add a title to the Figure
fig.suptitle('this is the figure title', fontsize=12)
# Show the plot
plt.show()
```



How To Set Log Scale

You can modify the scale of your axes to better show trends. That's why it might be useful in some cases to use the logarithmic scale on one or both axes. For a simple regression with `regplot()`, you can set the scale with the help of the `Axes` object.

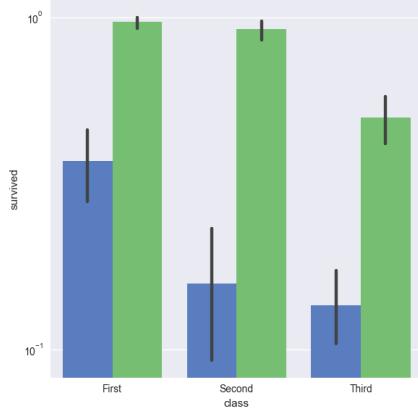
```
# Import the necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
# Create the data
x = 10 ** np.arange(1, 10)
y = x * 2
data = pd.DataFrame(data={'x': x, 'y': y})
# Initialize figure and ax
fig, ax = plt.subplots()
# Set the scale of the x-and y-axes
ax.set(xscale="log",yscale="log")
# Create a regplot
sns.regplot("x", "y", data, ax=ax, scatter_kws={"s": 100})
# Show plot
plt.show()
```



When you're working with Figure level functions, you can set the `xscale` and `yscale` properties with the help of the `set()` method of the `FacetGrid` object:

```
# Import the libraries
import matplotlib.pyplot as plt
import seaborn as sns
# Load data
titanic = sns.load_dataset("titanic")
```

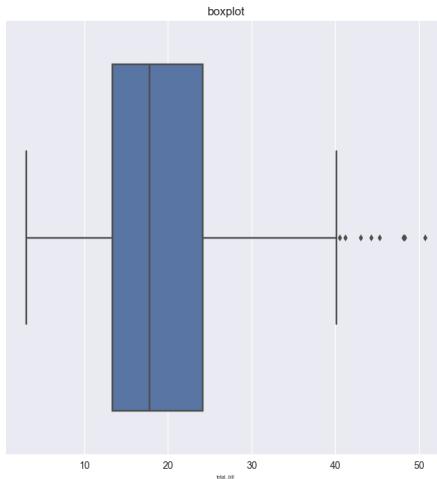
```
# Set up a factorplot
g = sns.factorplot("class", "survived", "sex", data=titanic, kind="bar", size=6, palette="muted", legend=False)
# Set the `yscale`
g.set(yscale="log")
# Show plot
plt.show()
```



How To Add A Title

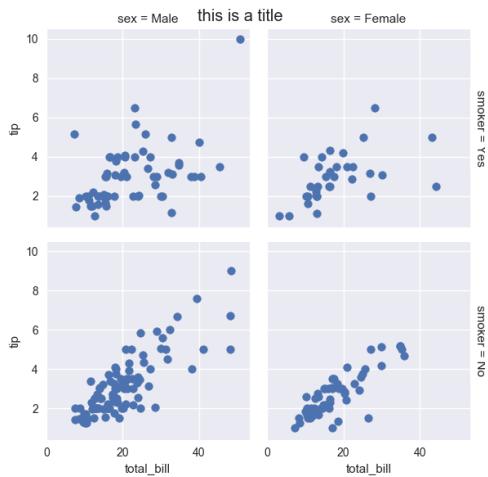
To add titles to your Seaborn plots, you basically follow the same procedure as you have done in the previous sections. For Axes-level functions, you'll adjust the title on the `Axes` level itself with the help of `set_title()`. Just pass in the title that you want to see appear:

```
# Import the libraries
import matplotlib.pyplot as plt
import seaborn as sns
tips = sns.load_dataset("tips")
# Create the boxplot
ax = sns.boxplot(x="total_bill", data=tips)
# Set title
ax.set_title("boxplot")
# Show the plot
plt.show()
```



For Figure-level functions, you can go via `fig`, just like in the factorplot that you have made in one of the previous sections, or you can also work via the Axes:

```
# Import the necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
# Load the data
tips = sns.load_dataset("tips")
# Create scatter plots
g = sns.FacetGrid(tips, col="sex", row="smoker", margin_titles=True)
g.map(sns.plt.scatter, "total_bill", "tip")
# Add a title to the figure
g.fig.suptitle("this is a title")
# Show the plot
plt.show()
```



Basic Plotting: Introduction to matplotlib

In this section, we will:

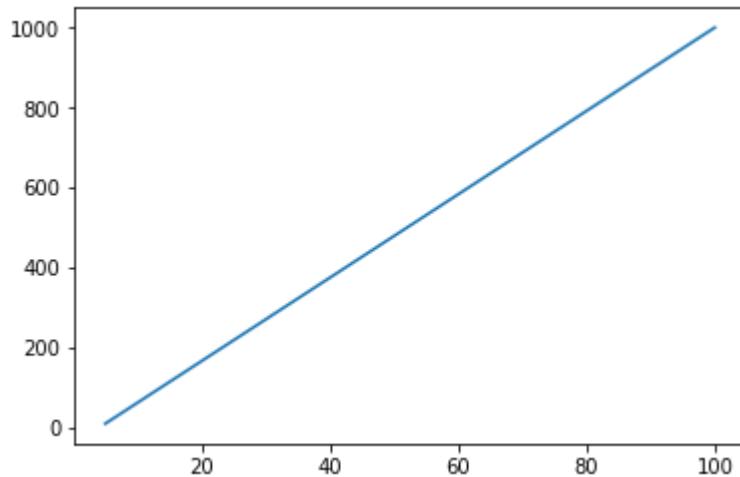
- Create basic plots using `matplotlib.pyplot`
- Put axis labels and titles
- Create multiple plots (subplots) in the same figure
- Change the scales of x and y axes
- Create common types of plots: Histograms, boxplots, scatter plots and bar charts
- Working with images

Basic Plotting, Axes Labels and Titles

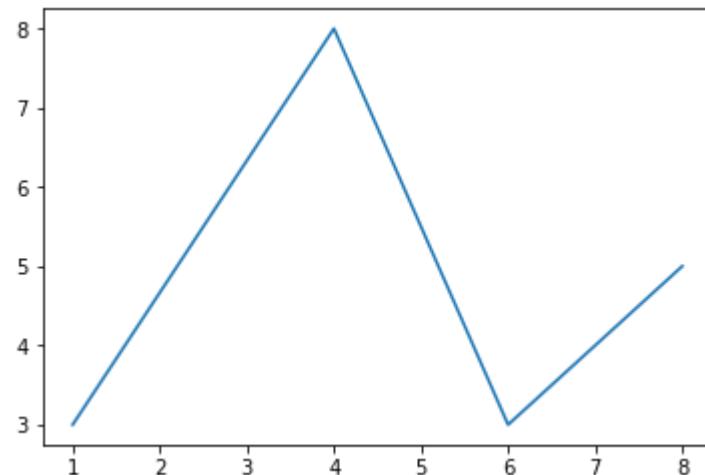
```
In [4]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Plotting two 1-D numpy arrays
x = np.linspace(5, 100, 100)
y = np.linspace(10, 1000, 100)

plt.plot(x, y)
plt.show()
```



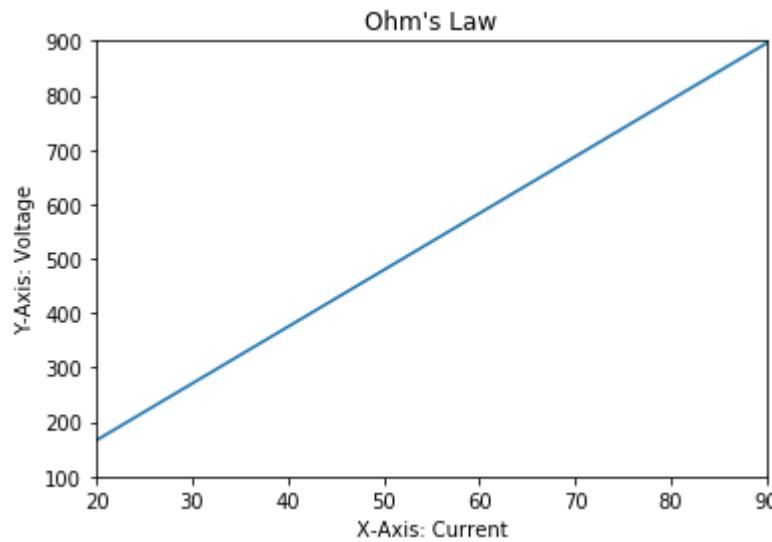
```
In [5]: # let us plot our first basic chart  
plt.plot([1, 4, 6, 8], [3, 8, 3, 5])  
plt.show()
```



Let's see how to put labels and the x and y axes and the chart title.

Also, you can specify the limits of x and y labels as a range using `xlim([xmin, xmax])` and `ylim([ymin, ymax])`.

```
In [6]: # Axis Labels and title  
plt.plot(x, y)  
  
# x and y labels, and title  
plt.xlabel("X-Axis: Current")  
plt.ylabel("Y-Axis: Voltage")  
plt.title("Ohm's Law")  
  
# Define the range of labels of the axis  
# Arguments: plt.axis(xmin, xmax, ymin, ymax)  
plt.xlim([20, 90])  
plt.ylim([100, 900])  
plt.show()
```



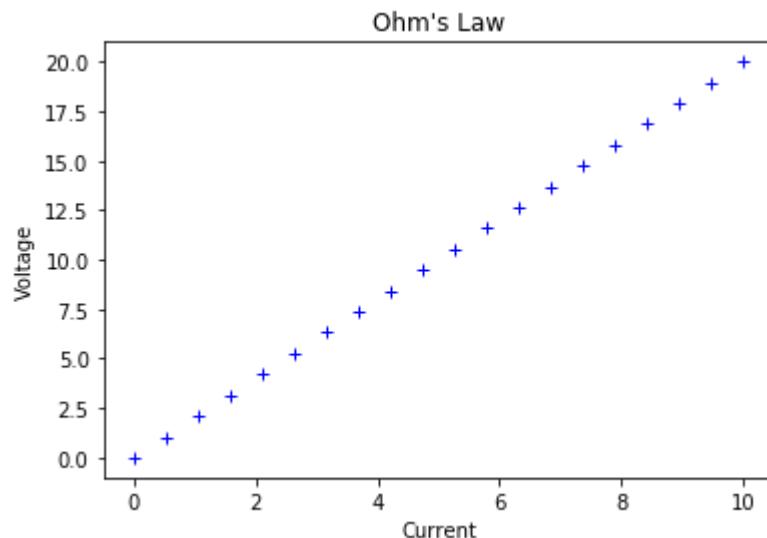
```
In [7]: # Change the colors and line type
```

```
# initialising x and y arrays
x = np.linspace(0, 10, 20)
y = x**2

# color blue, line type '+'
plt.plot(x, y, 'b+')

# put x and y Labels, and the title
plt.xlabel("Current")
plt.ylabel("Voltage")
plt.title("Ohm's Law")

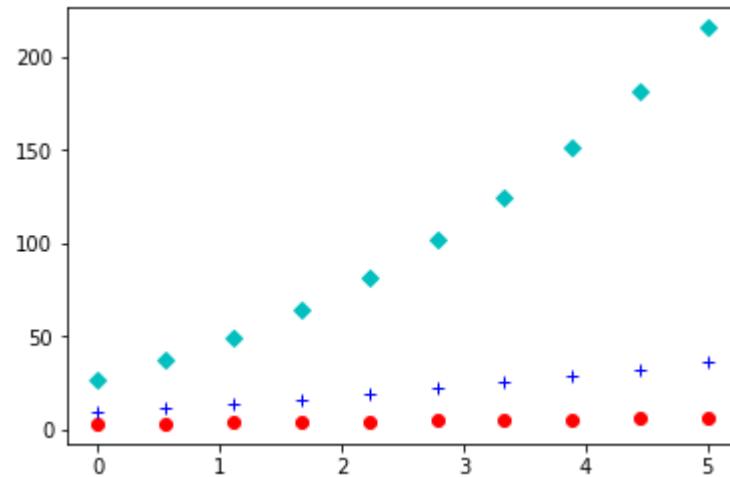
plt.show()
```



In [8]: # Plotting multiple lines on the same plot

```
x = np.linspace(0, 5, 10)
y = np.linspace(3, 6, 10)

# plot three curves: y, y**2 and y**3 with different line types
plt.plot(x, y, 'ro', x, y**2, 'b+', x, y**3, 'cD')
plt.show()
```



In [9]: # Plotting a complete chart

```
x = np.arange(1, 5)

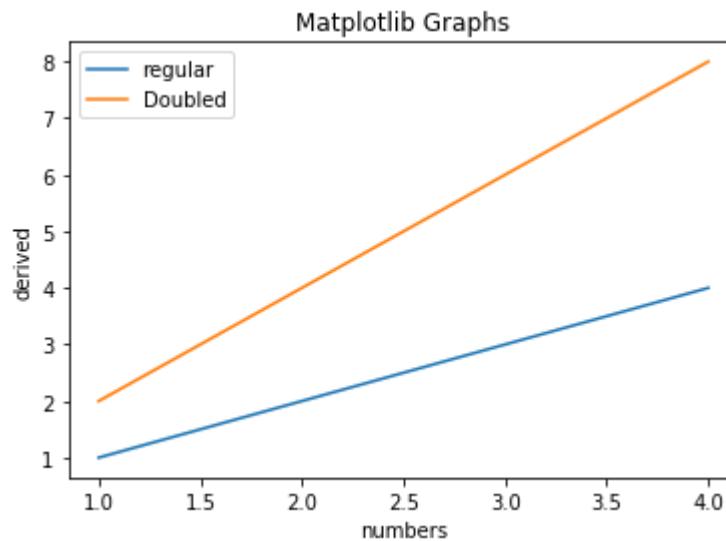
plt.title("Matplotlib Graphs")

plt.plot(x, x*1, label = "regular")
plt.plot(x, x*2, label = "Doubled")

plt.grid(False)
plt.xlabel('numbers')
plt.ylabel('derived')

plt.legend()

plt.show()
```



Decoration with Plot styles and Types

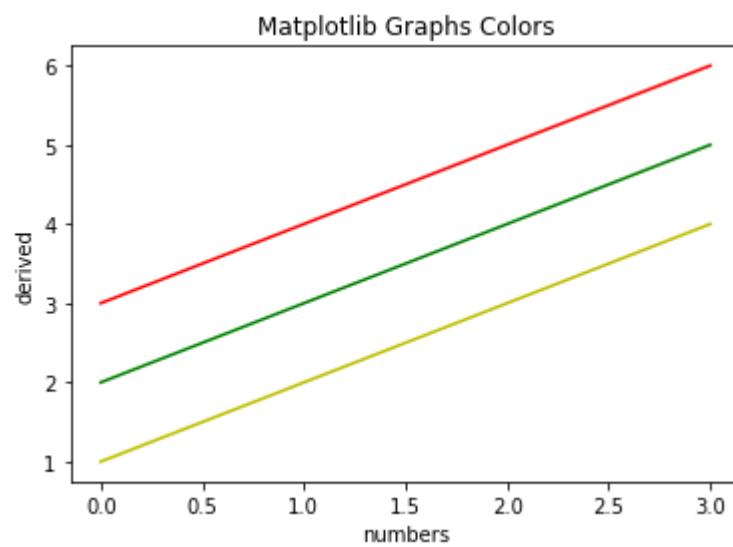
```
In [10]: x = np.arange(1, 5)

plt.title("Matplotlib Graphs Colors")

plt.plot(x, 'y')
plt.plot(x + 1, 'g')
plt.plot(x + 2, 'r')

plt.grid(False)
plt.xlabel('numbers')
plt.ylabel('derived')

plt.show()
```



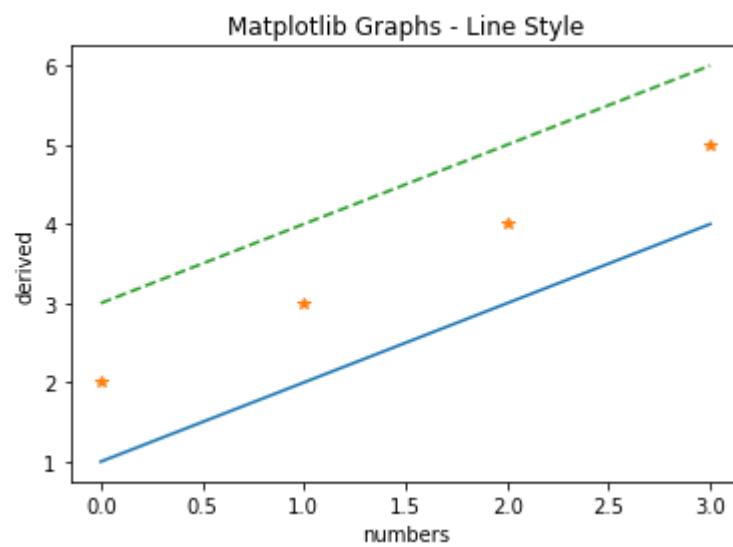
```
In [11]: x = np.arange(1, 5)

plt.title("Matplotlib Graphs - Line Style")

plt.plot(x, '-')
plt.plot(x + 1, '*')
plt.plot(x + 2, '--')

plt.grid(False)
plt.xlabel('numbers')
plt.ylabel('derived')

plt.show()
```



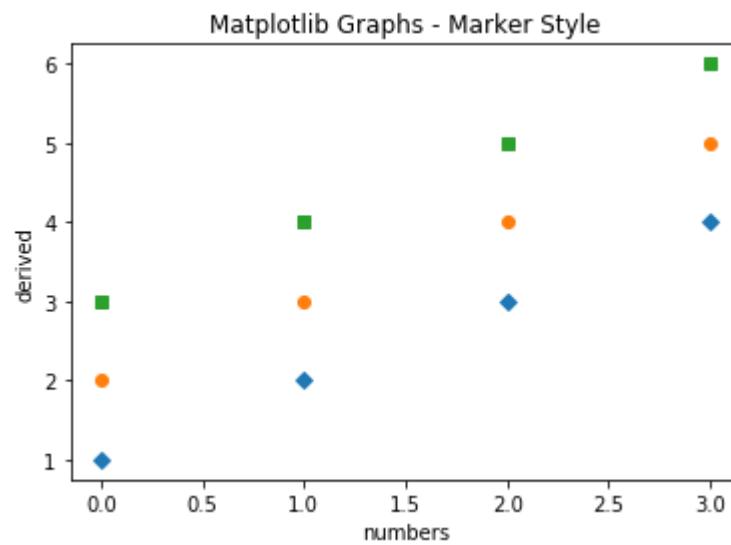
```
In [12]: x = np.arange(1, 5)

plt.title("Matplotlib Graphs - Marker Style")

plt.plot(x, 'D')
plt.plot(x + 1, 'o')
plt.plot(x + 2, 's')

plt.grid(False)
plt.xlabel('numbers')
plt.ylabel('derived')

plt.show()
```



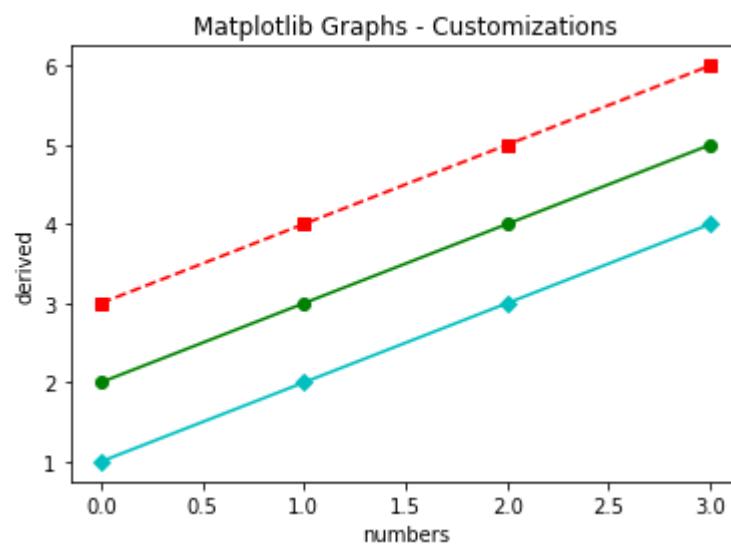
```
In [13]: x = np.arange(1, 5)

plt.title("Matplotlib Graphs - Customizations")

plt.plot(x, 'cD-')
plt.plot(x + 1, 'go-')
plt.plot(x + 2, 'rs--')

plt.grid(False)
plt.xlabel('numbers')
plt.ylabel('derived')

plt.show()
```



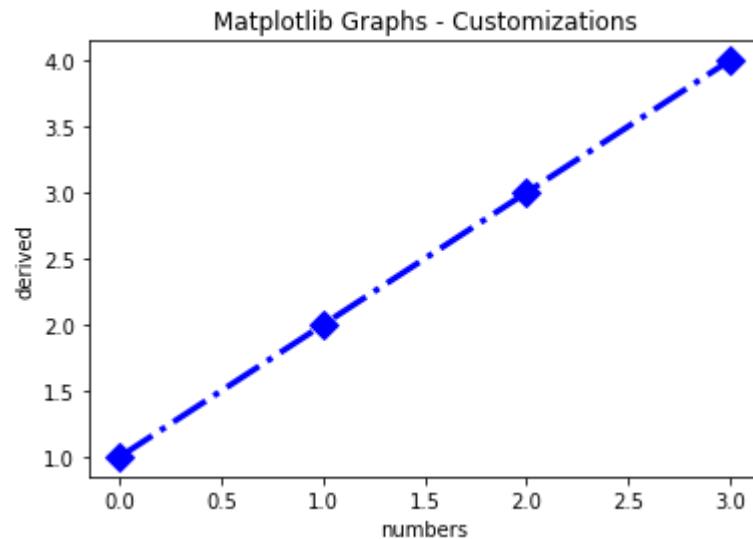
```
In [14]: x = np.arange(1, 5)

plt.title("Matplotlib Graphs - Customizations")

plt.plot(x, color="blue", linestyle="dashdot", linewidth=3, marker="D", markersize=10)

plt.grid(False)
plt.xlabel('numbers')
plt.ylabel('derived')

plt.show()
```

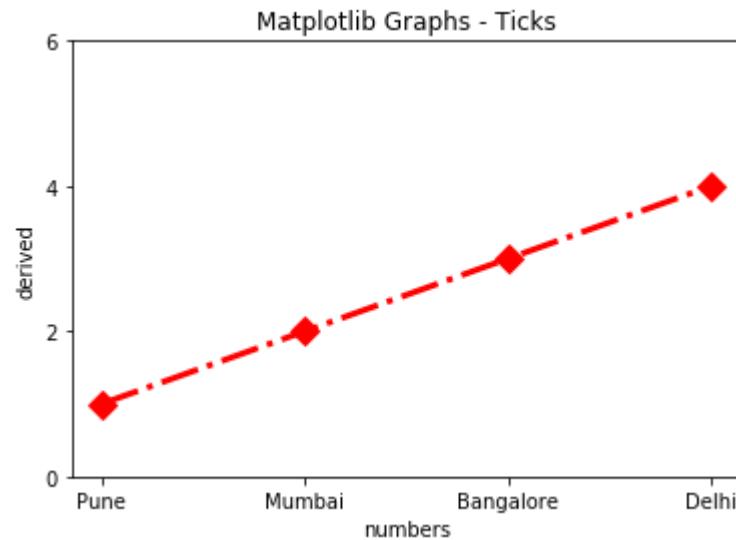


```
In [15]: x = np.arange(1, 5)

plt.title("Matplotlib Graphs - Ticks")

plt.plot(x, color="red", linestyle="dashdot", linewidth=3, marker="D", markersize=10)
plt.xticks(range(len(x)),['Pune', 'Mumbai', "Bangalore", "Delhi"])
plt.yticks(range(0, 8, 2))
plt.grid(False)
plt.xlabel('numbers')
plt.ylabel('derived')

plt.show()
```



Advanced Matplotlib

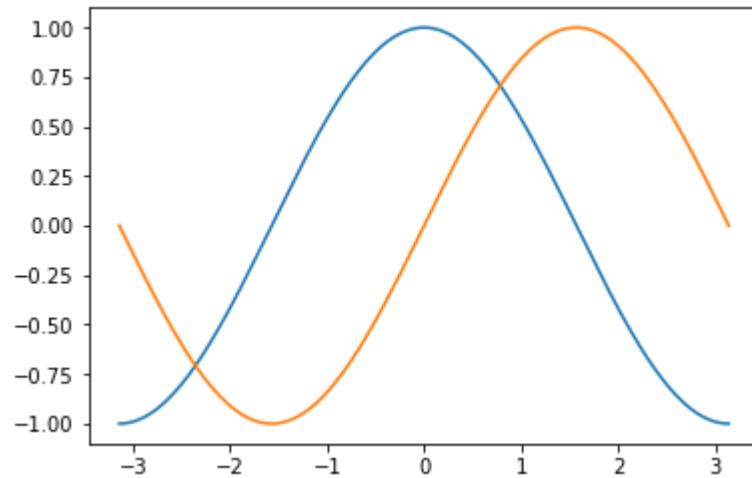
In this section, we want to draw the cosine and sine functions on the same plot. Starting from the default settings, we'll enrich the figure step by step to make it nicer.

```
In [17]: import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt
```

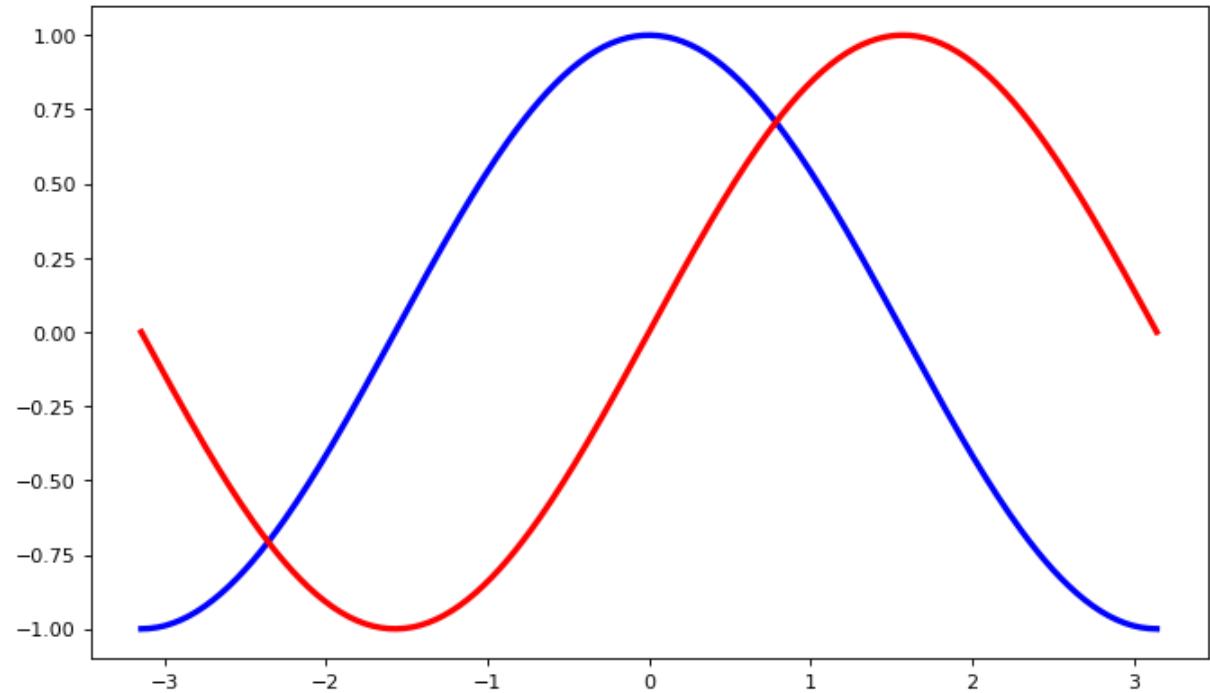
```
In [18]: X = np.linspace(-np.pi, np.pi, 256, endpoint = True)  
C, S = np.cos(X), np.sin(X)
```

X is now a NumPy array with 256 values ranging from $-\pi$ to $+\pi$ (included). C is the cosine (256 values) and S is the sine (256 values).

```
In [19]: plt.plot(X,C)  
plt.plot(X,S)  
plt.show()
```



```
In [20]: plt.figure(figsize=(10,6), dpi = 80)
plt.plot(X,C, color = 'blue', linewidth = 3, linestyle = '-')
plt.plot(X,S, color = 'red', linewidth = 3, linestyle = '-')
plt.show()
```



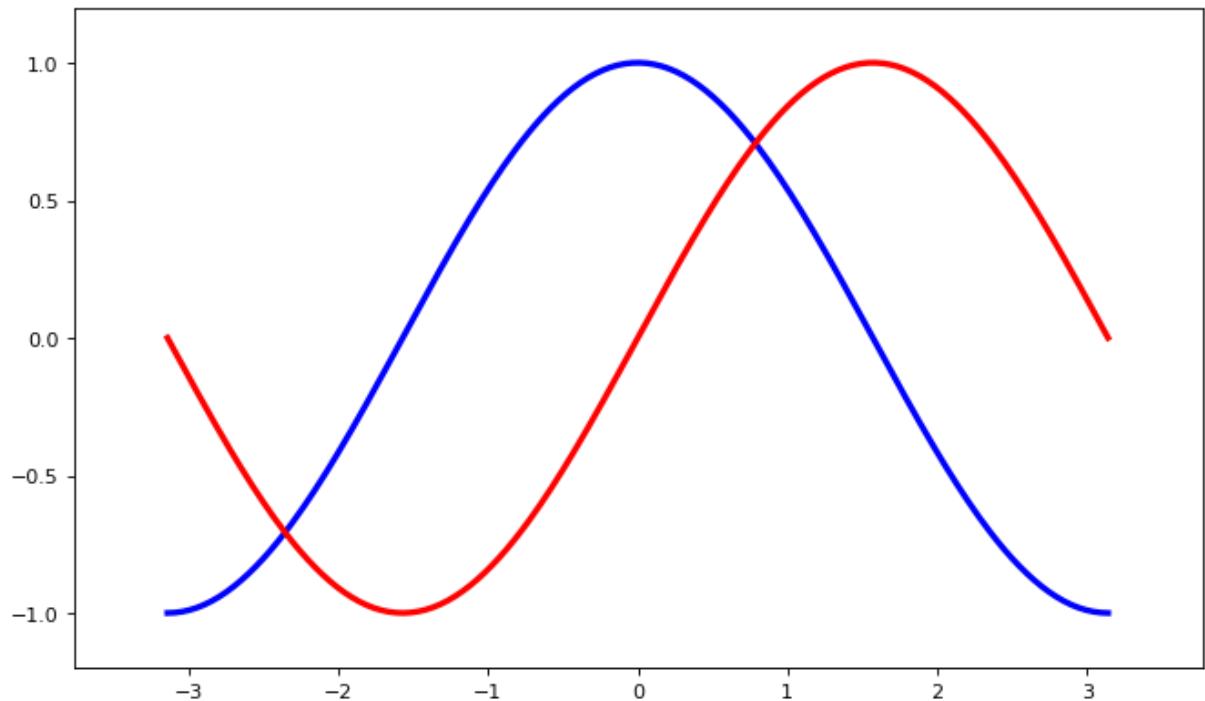
Setting limits

current limits of the figure are a bit too tight and we want to make some space in order to clearly see all the data points

```
In [21]: plt.figure(figsize=(10,6), dpi = 80)
plt.plot(X,C, color = 'blue', linewidth = 3, linestyle = '-')
plt.plot(X,S, color = 'red', linewidth = 3, linestyle = '-')

plt.xlim(X.min()*1.2, X.max()*1.2)
plt.ylim(C.min()*1.2, C.max()*1.2)

plt.show()
```



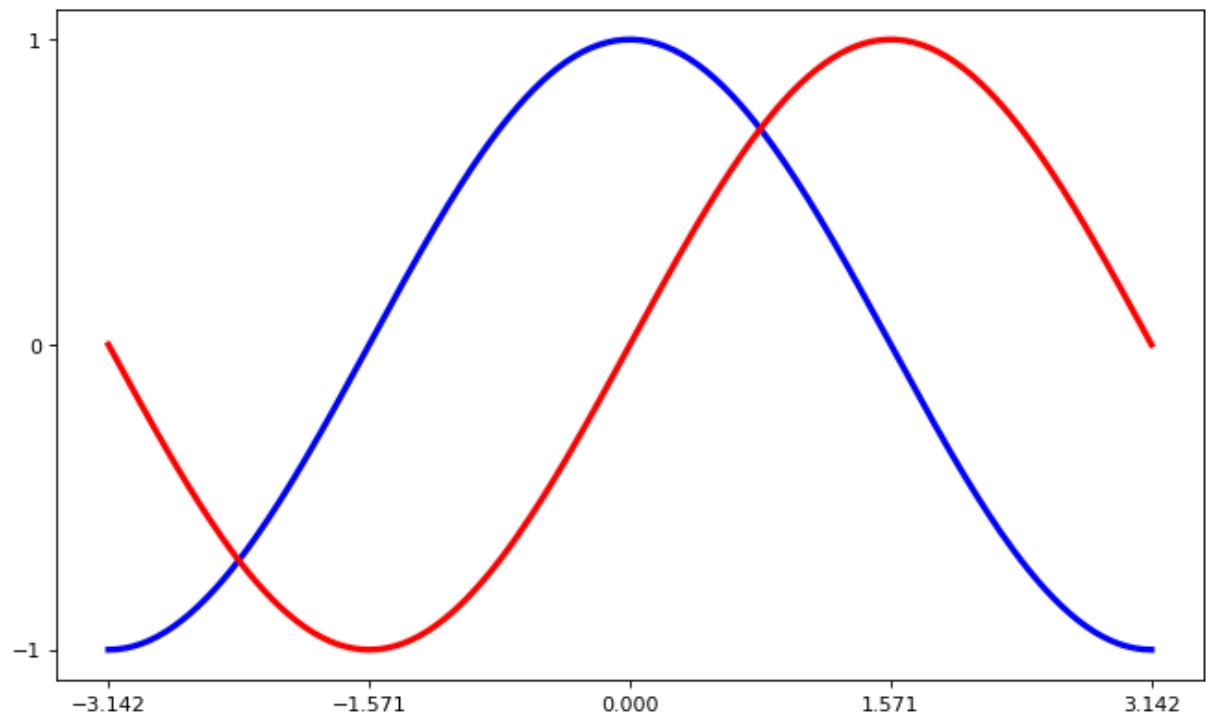
Setting ticks

Current ticks are not ideal because they do not show the interesting values ($+/-\pi, +/-\pi/2$) for sine and cosine. We'll change them such that they show only these values.

```
In [22]: plt.figure(figsize=(10,6), dpi = 80)
plt.plot(X,C, color = 'blue', linewidth = 3, linestyle = '-')
plt.plot(X,S, color = 'red', linewidth = 3, linestyle = '-')

plt.xticks( [-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
plt.yticks([-1, 0, +1])

plt.show()
```



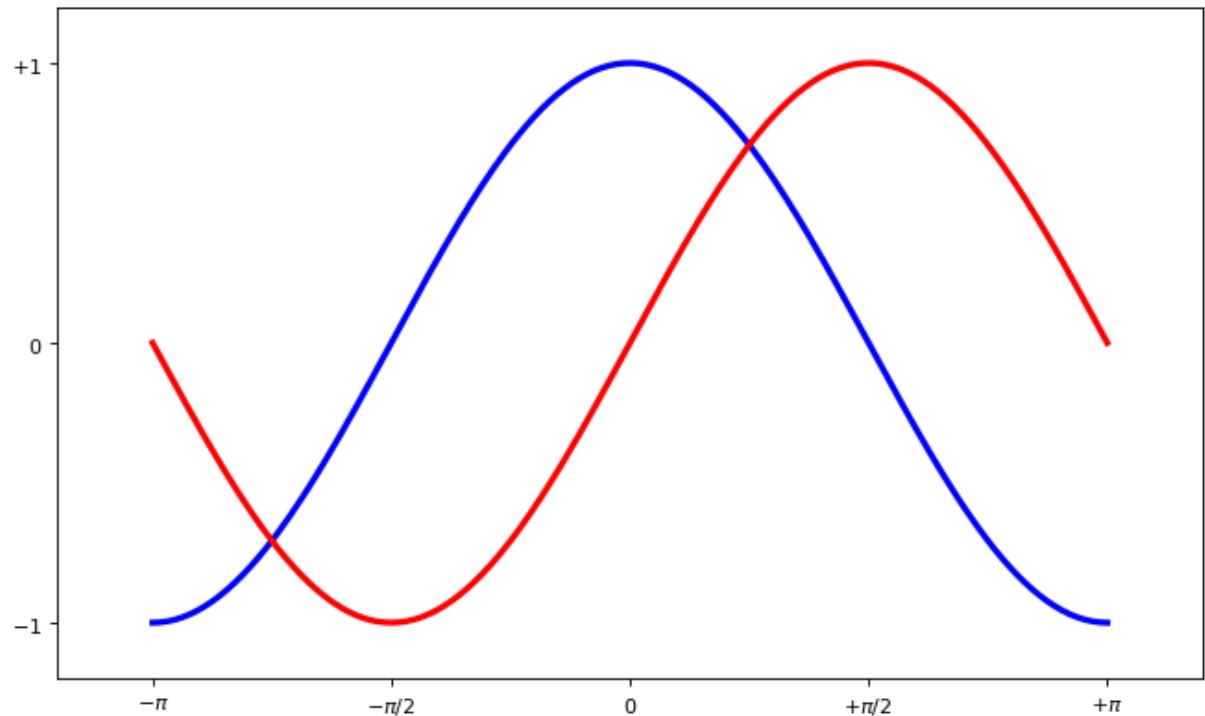
```
In [23]: plt.figure(figsize=(10,6), dpi = 80)
plt.plot(X,C, color = 'blue', linewidth = 3, linestyle = '-')
plt.plot(X,S, color = 'red', linewidth = 3, linestyle = '-')

plt.xlim(X.min()*1.2, X.max()*1.2)
plt.ylim(C.min()*1.2, C.max()*1.2)

# We want to mention the pi values instead of numerical values in the x-axis
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
           ['$-\pi$', '$-\pi/2$', '$0$', '$+\pi/2$', '$+\pi$'])

plt.yticks([-1, 0, +1],
           ['$-1$', '$0$', '$+1$'])

plt.show()
```



Moving spine/ axes

Spines are the lines connecting the axis tick marks and noting the boundaries of the data area. They can be placed at arbitrary positions and until now, they were on the border of the axis. We'll change that since we want to have them in the middle. Since there are four of them (top/bottom/left/right), we'll discard the top and right by setting their color to none and we'll move the bottom and left ones to coordinate 0 in data space coordinates.

```
In [24]: plt.figure(figsize=(10,6), dpi = 80)
plt.plot(X,C, color = 'blue', linewidth = 2, linestyle = '-')
plt.plot(X,S, color = 'red', linewidth = 2, linestyle = '-')

plt.xlim(X.min()*1.2, X.max()*1.2)
plt.ylim(C.min()*1.2, C.max()*1.2)

# We want to mention the pi values instead of numerical values in the x-axis
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
           ['$-\pi$', '$-\pi/2$', '$0$', '$+\pi/2$', '$+\pi$'])

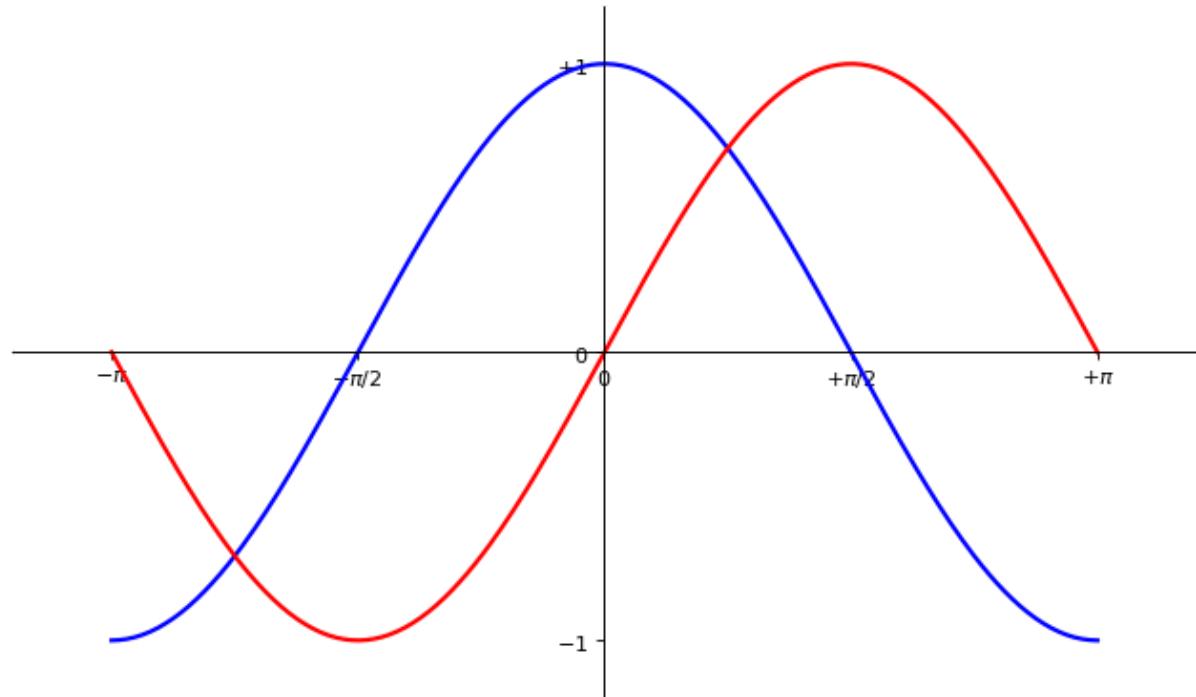
plt.yticks([-1, 0, +1],
           ['$-1$', '$0$', '$+1$'])

ax = plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')

#ax.spines['right'].set_visible(False)
#ax.spines['top'].set_visible(False)

ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

plt.show()
```



Adding legends

```
In [25]: plt.figure(figsize=(10,6), dpi = 80)
plt.plot(X,C, color = 'blue', linewidth = 2, linestyle = '--', label = 'cosine')
plt.plot(X,S, color = 'red', linewidth = 2, linestyle = '--', label = 'sine')

plt.xlim(X.min()*1.2, X.max()*1.2)
plt.ylim(C.min()*1.2, C.max()*1.2)

# We want to mention the pi values instead of numerical values in the x-axis
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
           ['$-\pi$', '$-\pi/2$', '$0$', '$+\pi/2$', '$+\pi$'])

plt.yticks([-1, 0, +1],
           ['$-1$', '$0$', '$+1$'])

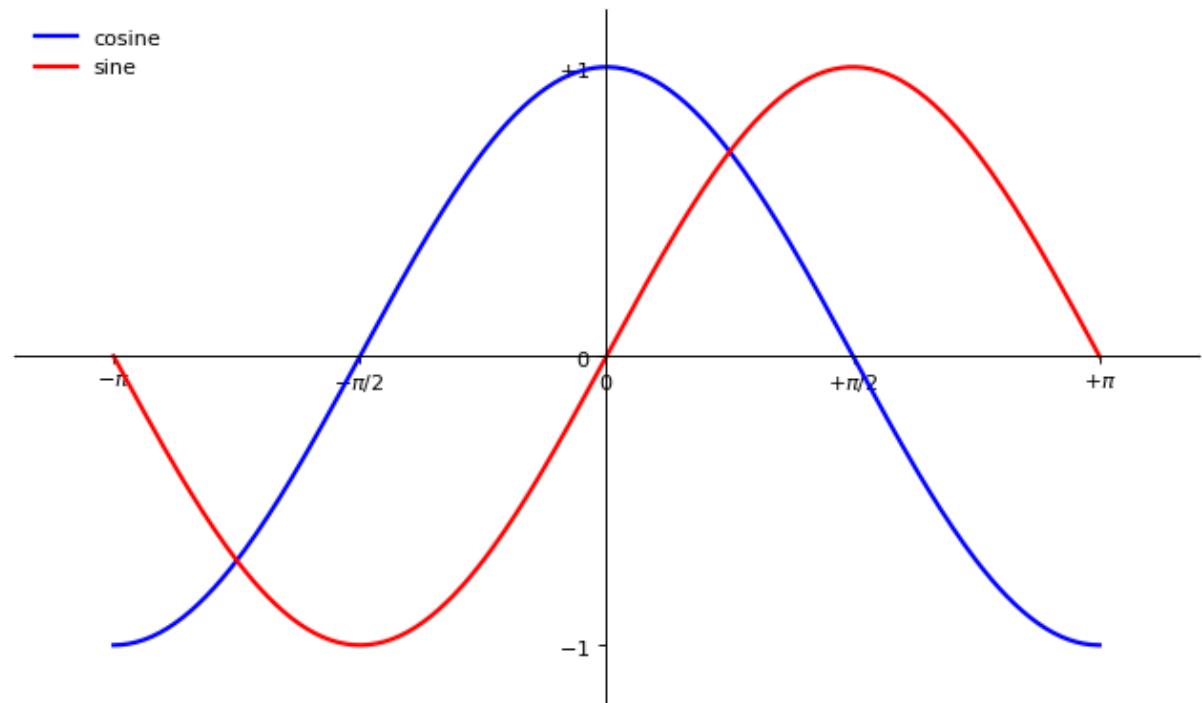
ax = plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')

#ax.spines['right'].set_visible(False)
#ax.spines['top'].set_visible(False)

ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

plt.legend(loc='upper left', frameon = False)

plt.show()
```



```
In [26]: plt.figure(figsize=(10,6), dpi = 80)
plt.plot(X,C, color = 'blue', linewidth = 2, linestyle = '--', label = 'cosine')
plt.plot(X,S, color = 'red', linewidth = 2, linestyle = '--', label = 'sine')

plt.xlim(X.min()*1.2, X.max()*1.2)
plt.ylim(C.min()*1.2, C.max()*1.2)

# We want to mention the pi values instead of numerical values in the x-axis
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi],
           ['$-\pi$', '$-\pi/2$', '$0$', '$+\pi/2$', '$+\pi$'])

plt.yticks([-1, 0, +1],
           ['$-1$', '$0$', '$+1$'])

ax = plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')

#ax.spines['right'].set_visible(False)
#ax.spines['top'].set_visible(False)

ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

t = 2*np.pi/3
plt.plot([t,t],[0,np.cos(t)], color ='blue', linewidth=1.5, linestyle="--")
plt.scatter(t,np.cos(t), 50, color ='blue')

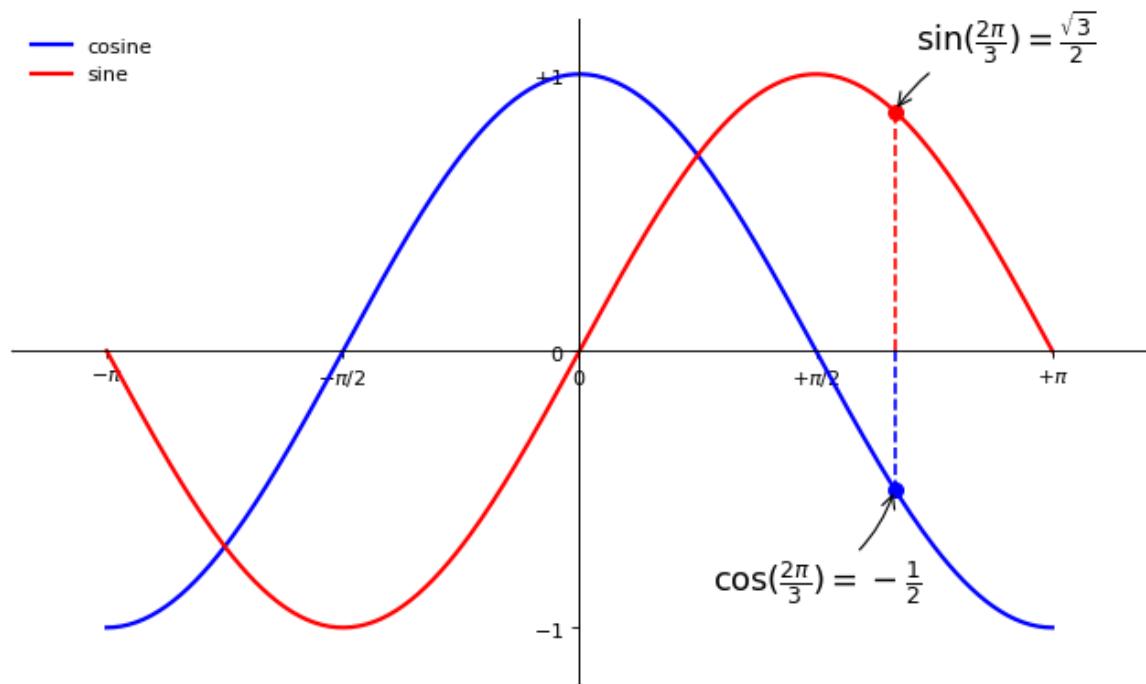
plt.annotate(r'$\sin(\frac{2\pi}{3})=\frac{\sqrt{3}}{2}$',
            xy=(t, np.sin(t)), xycoords='data',
            xytext=(+10, +30), textcoords='offset points', fontsize=16,
            arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))

plt.plot([t,t],[0,np.sin(t)], color ='red', linewidth=1.5, linestyle="--")
plt.scatter(t,np.sin(t), 50, color ='red')

plt.annotate(r'$\cos(\frac{2\pi}{3})=-\frac{1}{2}$',
            xy=(t, np.cos(t)), xycoords='data',
            xytext=(-90, -50), textcoords='offset points', fontsize=16,
            arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))

plt.legend(loc='upper left', frameon = False)

plt.show()
```



Figures and Subplots

You often need to create multiple plots in the same figure, as we'll see in some upcoming examples.

`matplotlib` has the concept of **figures and subplots** using which you can create *multiple subplots inside the same figure*.

To create multiple plots in the same figure, you can use the method
`plt.subplot(nrows, ncols, nsubplot)` .

```
In [27]: x = np.linspace(1, 10, 100)
y = np.log(x)

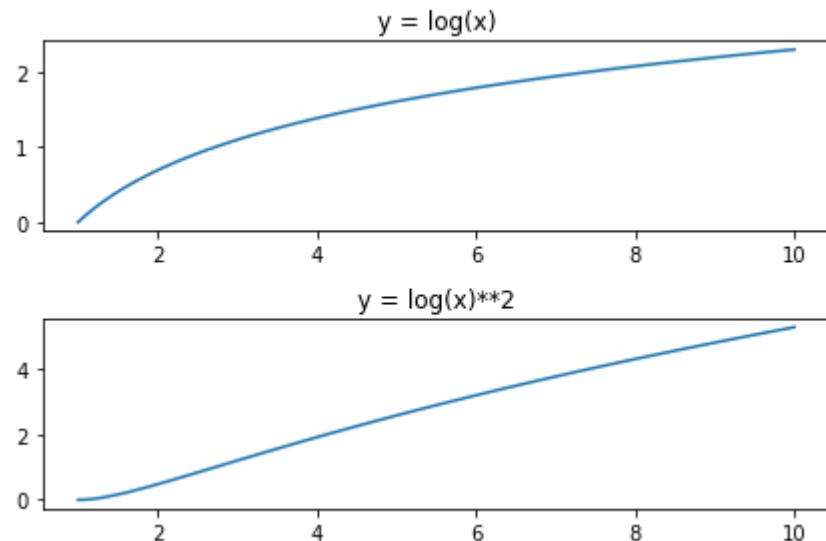
# initiate a new figure explicitly
plt.figure(1)

# Create a subplot with 1 row, 2 columns

# create the first subplot in figure 1
plt.subplot(2,1,1)      # equivalent to plt.subplot(1, 2, 1)
plt.title("y = log(x)")
plt.plot(x, y)

# create the second subplot in figure 1
plt.subplot(212)
plt.title("y = log(x)**2")
plt.plot(x, y**2)

plt.tight_layout()
plt.show()
```



Let's see another example - say you want to create 4 subplots in two rows and two columns.

```
In [28]: # Example: Create a figure having 4 subplots
x = np.linspace(1, 10, 100)

# Optional command, since matplotlib creates a figure by default anyway
plt.figure(1)

# subplot 1
plt.subplot(2, 2, 1)
plt.title("Linear")
plt.plot(x, x)

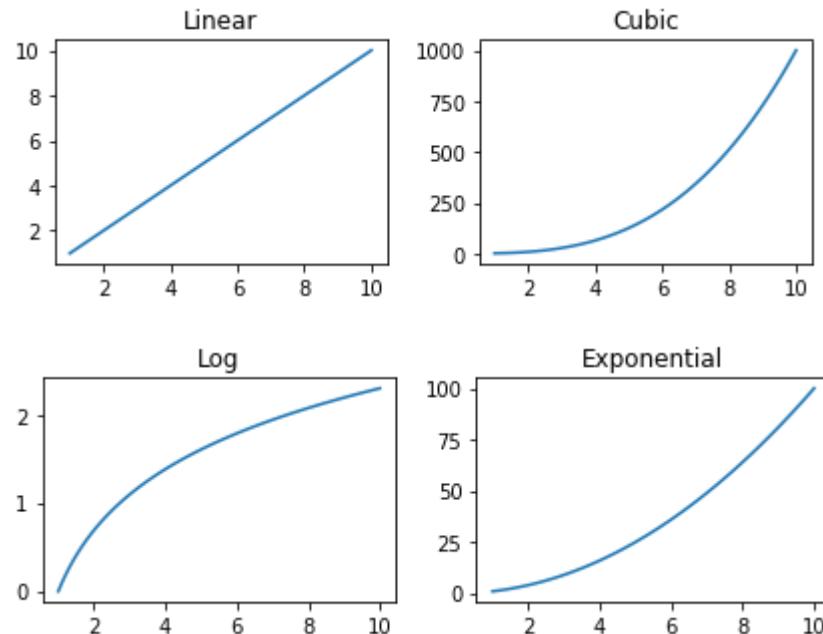
# subplot 2
plt.subplot(2, 2, 2)
plt.title("Cubic")
plt.plot(x, x**3)

plt.tight_layout()

# subplot 3
plt.figure(2)
plt.subplot(2, 2, 1)
plt.title("Log")
plt.plot(x, np.log(x))

# subplot 4
plt.subplot(2, 2, 2)
plt.title("Exponential")
plt.plot(x, x**2)

plt.tight_layout()
plt.show()
```



You can see the list of colors and shapes here:

https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot

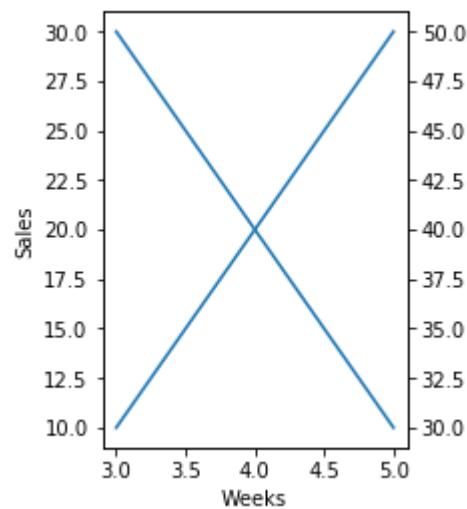
(https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot)

Dual Y Axis

```
In [29]: fig = plt.figure()

plot1 = fig.add_subplot(1, 2, 1)
plot1.plot([3, 4, 5], [10, 20, 30])
plot1.set_xlabel("Weeks")
plot1.set_ylabel("Sales")

plot2 = plot1.twinx()
plot2.plot([3, 4, 5], [50, 40, 30])
plt.show()
```

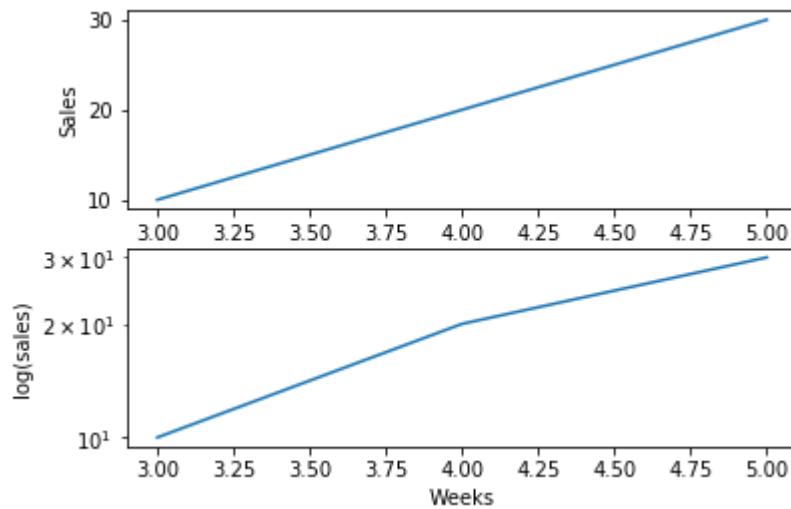


Logarithmic Axis

```
In [30]: fig = plt.figure()

plot1 = fig.add_subplot(2, 1, 1)
plot1.plot([3, 4, 5], [10, 20, 30])
plot1.set_xlabel("Weeks")
plot1.set_ylabel("Sales")

plot2 = fig.add_subplot(2, 1, 2)
plot2.plot([3, 4, 5], [10, 20, 30])
plot2.set_ylabel("log(sales)")
plot2.set_xlabel("Weeks")
plot2.set_yscale('log')
plt.show()
```



Types of Commonly Used Plots

Let's now use the retail store's sales data to create some commonly used plots such as:

- Boxplots
- Histograms
- Scatter plots
- Bar plots

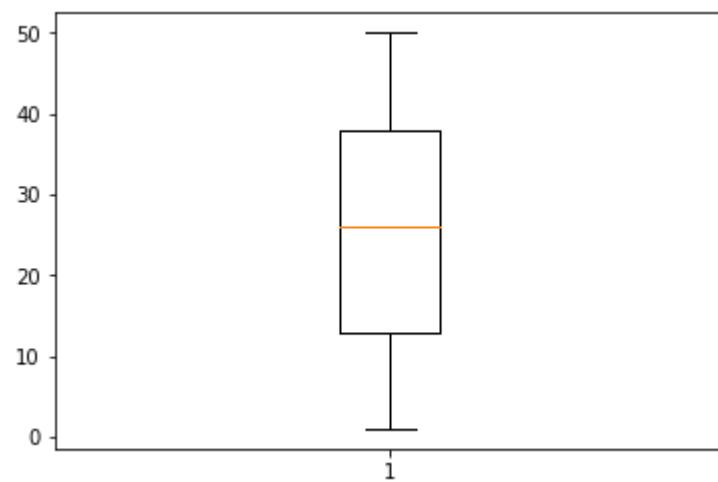
```
In [31]: # Example: Globals sales data  
df = pd.read_csv("market_fact.csv")  
df.head()
```

Out[31]:

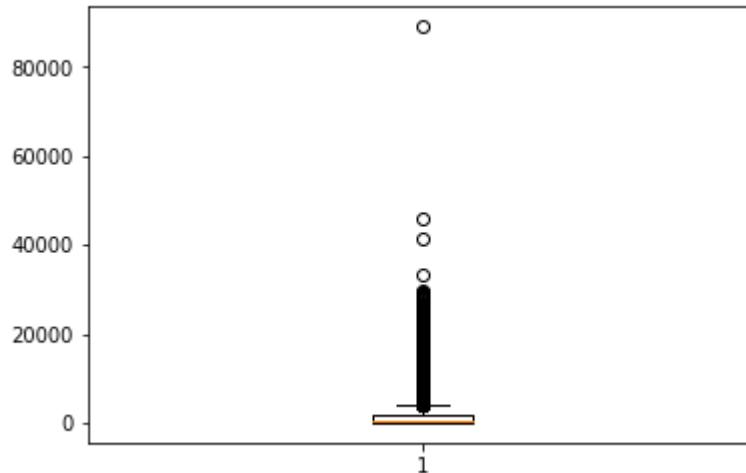
	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	

Boxplot

```
In [32]: # Boxplot: Visualise the distribution of a continuous variable  
plt.boxplot(df['Order_Quantity'])  
plt.show()
```



```
In [33]: # Boxplot of Sales is quite unreadable, since Sales varies  
# across a wide range  
plt.boxplot(df['Sales'])  
plt.show()
```



As you can see, the boxplot of `Sales` is pretty unreadable, since `Sales` varies across a wide range as shown below.

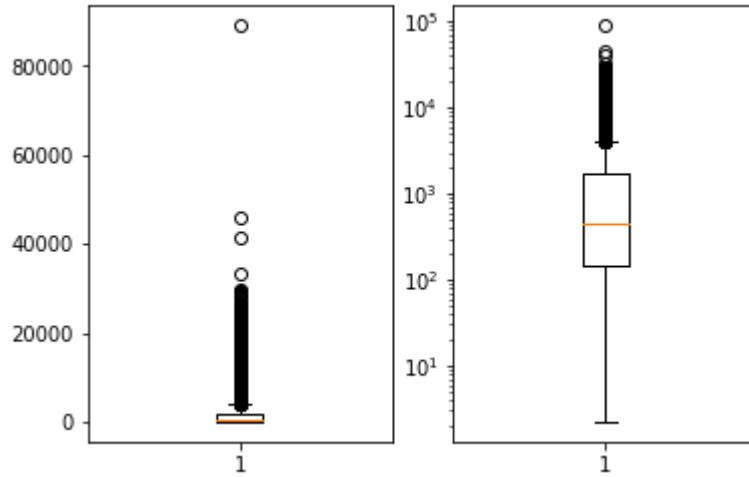
```
In [34]: # Range of sales: min is 2.24, median is 449, max is 89061  
df['Sales'].describe()
```

```
Out[34]: count    8399.000000  
mean      1775.878179  
std       3585.050525  
min       2.240000  
25%      143.195000  
50%      449.420000  
75%     1709.320000  
max     89061.050000  
Name: Sales, dtype: float64
```

The solution to this problem is to **change the scale of the axis** (in this case, the y axis) so that the range can fit into the size of the plot.

One commonly used technique is to transform an axis into the **logarithmic scale**. You can transform the scale of an axis using `plt.yscale('log')`.

```
In [35]: # Usual (linear) scale subplot  
plt.subplot(1, 2, 1)  
plt.boxplot(df['Sales'])  
  
# Log scale subplot  
plt.subplot(1, 2, 2)  
plt.boxplot(df['Sales'])  
plt.yscale('log')  
plt.show()
```



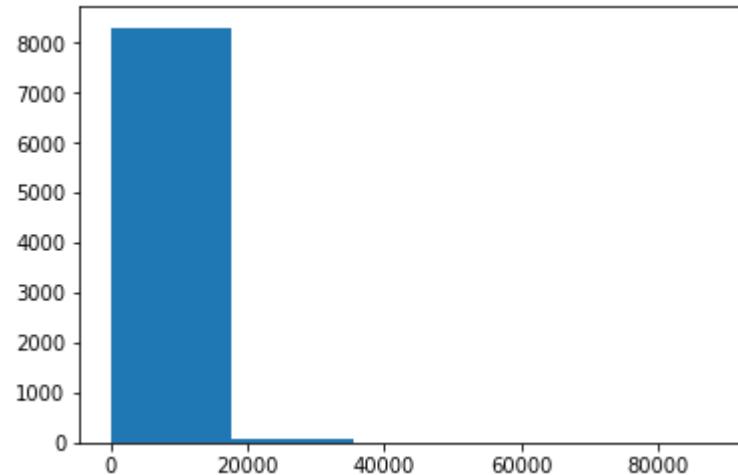
Clearly, the log scale subplot is far more readable - you can infer that the minimum sales is around 0, the median is approximately in the middle of 100 and 1000, and the max is reaching 100,000.

Histogram

Histograms are useful for visualising distribution of single variables.

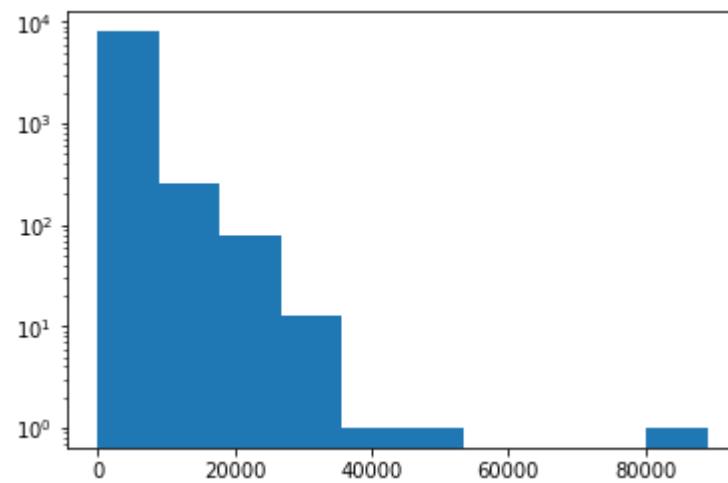
In [36]: # Histograms

```
num_bins = 5 #default number of bins is 10 if not specified separately
plt.hist(df['Sales'],num_bins)
plt.show()
```



In [37]: # The histogram can be made more readable by using a log scale

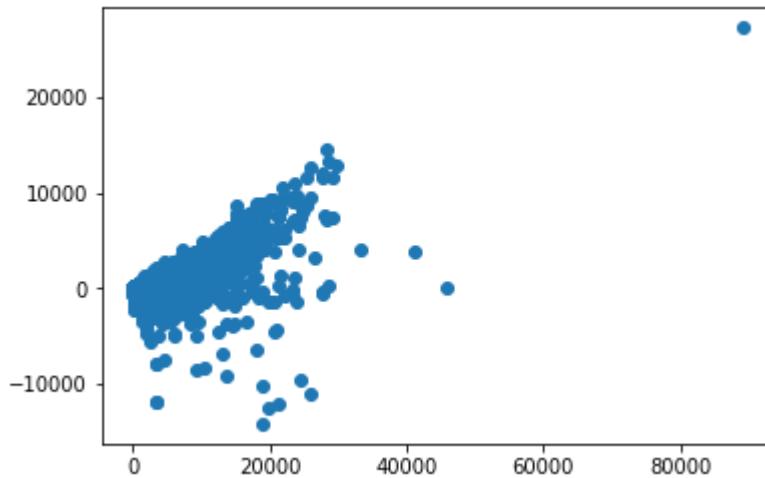
```
plt.hist(df['Sales'])
plt.yscale('log')
plt.show()
```



Scatter Plot

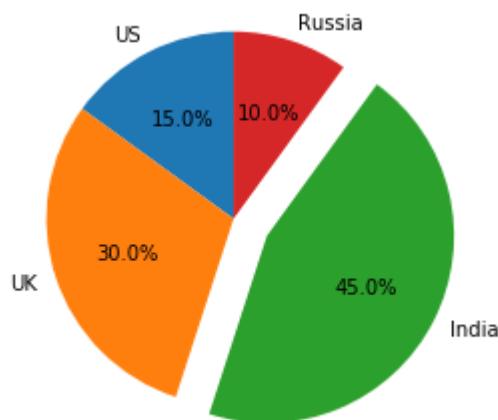
Scatter plots are used to visualise two variables, one on each axis.

```
In [38]: # Scatter plots with two variables: Profit and Sales  
plt.scatter(df['Sales'], df['Profit'])  
plt.show()
```



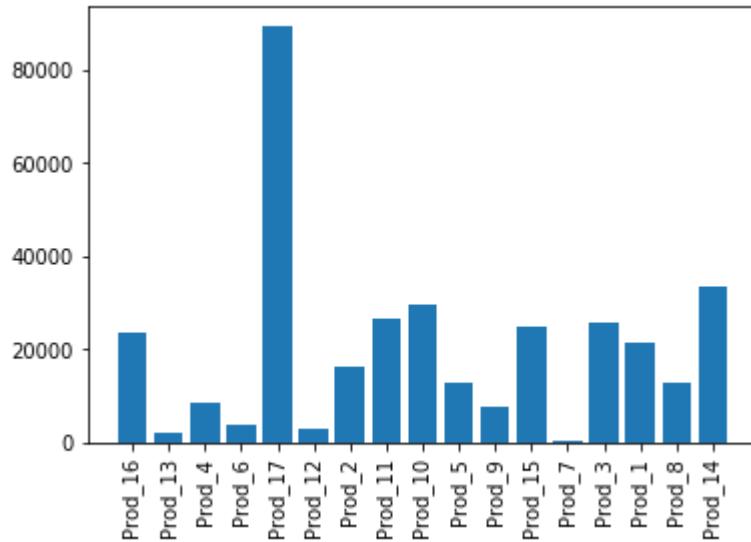
Pie chart

```
In [39]: # Pie chart, where the slices will be ordered and plotted counter-clockwise  
  
#Prepare data  
country_labels = 'US', 'UK', 'India', 'Russia'  
country_sales = [15, 30, 45, 10]  
explode = (0, 0, 0.2, 0) # only "explode" the 2nd slice  
  
fig1, ax1 = plt.subplots()  
ax1.pie(country_sales, explode=explode, labels=country_labels, autopct='%1.1f%%',  
        ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.  
  
plt.show()
```

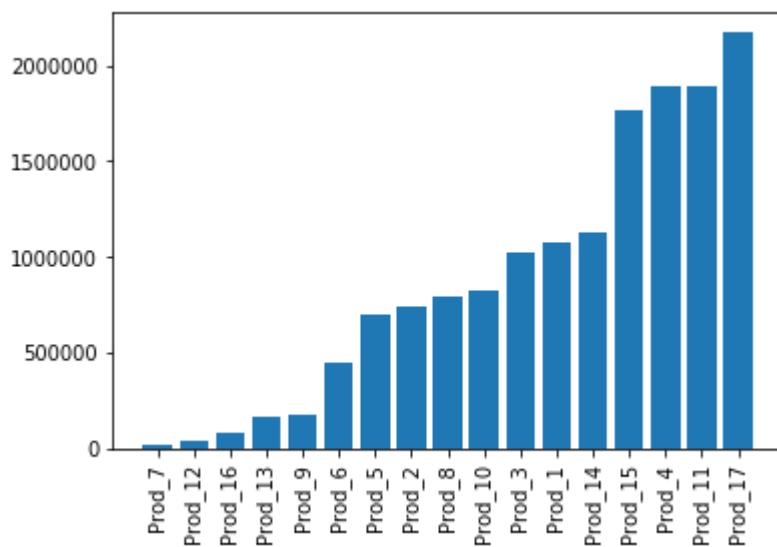


Bar Chart

```
In [40]: plt.bar(df['Prod_id'], df['Sales']) # to produce a horizontal bar chart, use "barh"
plt.xticks(rotation = 90)
plt.show()
```



```
In [41]: # produce the same graph in ascending order.
result = df.groupby(["Prod_id"])['Sales'].aggregate(np.sum).reset_index().sort_values('Sales')
plt.bar(result['Prod_id'],result['Sales'])
plt.xticks(rotation = 90)
plt.show()
```



In [42]: df.head()

Out[42]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	

In [43]: df.describe()

Out[43]:

	Sales	Discount	Order_Quantity	Profit	Shipping_Cost	Product_Base_Material
count	8399.000000	8399.000000	8399.000000	8399.000000	8399.000000	8336.000
mean	1775.878179	0.049671	25.571735	181.184424	12.838557	0.512
std	3585.050525	0.031823	14.481071	1196.653371	17.264052	0.135
min	2.240000	0.000000	1.000000	-14140.700000	0.490000	0.350
25%	143.195000	0.020000	13.000000	-83.315000	3.300000	0.380
50%	449.420000	0.050000	26.000000	-1.500000	6.070000	0.520
75%	1709.320000	0.080000	38.000000	162.750000	13.990000	0.590
max	89061.050000	0.250000	50.000000	27220.690000	164.730000	0.850

```
In [44]: # plotting two variables on the same bar chart
result = df.groupby(["Prod_id"])['Order_Quantity'].aggregate(np.sum).reset_index()
result1=df.groupby(["Prod_id"])['Shipping_Cost'].aggregate(np.sum).reset_index()
_x=np.arange(len(result['Prod_id']))
plt.bar(_x-0.1,result['Order_Quantity'],width=0.2, color='b', align='center')
plt.bar(_x+0.1,result1['Shipping_Cost'],width=0.2, color='g', align='center')
plt.xticks(_x,result['Prod_id'],rotation = 90)
plt.yscale('log')
plt.gca().legend(('Order Quantity','Shipping Costs'))
plt.show()
```



A few good resources for practicing matplotlib visualization

1. [Official documentation](https://matplotlib.org/users/pyplot_tutorial.html) (https://matplotlib.org/users/pyplot_tutorial.html)
2. [Matplotlib tutorial showing a variety of plots](https://github.com/rougier/matplotlib-tutorial) (<https://github.com/rougier/matplotlib-tutorial>)
3. [Matplotlib reference site](https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot) (https://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot)

Data Visualization Session 13

Visualising Distributions of Data

In this section, we will see how to:

- Visualise univariate distributions
- Visualise bivariate distributions

We will also start using the `seaborn` library for data visualisation. Seaborn is a python library built on top of `matplotlib`. It creates much more attractive plots than `matplotlib`, and is often more concise than `matplotlib` when you want to customize your plots, add colors, grids etc.

Let's start with univariate distributions.

Visualising Univariate Distributions

We have already visualised univariate distributions before using boxplots, histograms etc. Let's now do that using `seaborn`. We'll use the sales data for the upcoming few exercises.

```
In [ ]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# the commonly used alias for seaborn is sns
import seaborn as sns

# set a seaborn style of your taste
sns.set_style("whitegrid") #few other commonly used styles are "darkgrid", "white"

# data
df = pd.read_csv("./global_sales_data/market_fact.csv")
df.head()
```

Histograms and Density Plots

Histograms and density plots show the frequency of a numeric variable along the y-axis, and the value along the x-axis. The `sns.distplot()` function plots a density curve. Notice that this is aesthetically better than vanilla `matplotlib`.

```
In [ ]: # simple density plot
sns.distplot(df['Shipping_Cost']) #unlike matplotlib the default bins value is de
plt.show()
```

You can also plot what is known as the **rug plot** which plots the actual data points as small vertical

bars. The rug plot is simply specified as an argument of the `distplot()` .

```
In [ ]: # rug = True  
# plotting only a few points since rug takes a long while  
sns.distplot(df['Shipping_Cost'][:200], rug=True)  
plt.show()
```

Simple density plot (without the histogram bars) can be created by specifying `hist=False` .

```
In [ ]: sns.distplot(df['Sales'], hist=False)  
plt.show()
```

Since seaborn uses matplotlib behind the scenes, the usual matplotlib functions work well with seaborn. For example, you can use subplots to plot multiple univariate distributions.

```
In [ ]: # subplots  
  
# subplot 1  
plt.subplot(2, 2, 1)  
# plt.title('Sales')  
sns.distplot(df['Sales'])  
  
# subplot 2  
plt.subplot(2, 2, 2)  
# plt.title('Profit')  
sns.distplot(df['Profit'])  
  
# subplot 3  
plt.subplot(2, 2, 3)  
# plt.title('Order Quantity')  
sns.distplot(df['Order_Quantity'])  
  
# subplot 4  
plt.subplot(2, 2, 4)  
# plt.title('Shipping Cost')  
sns.distplot(df['Shipping_Cost'])  
plt.tight_layout()  
plt.show()
```

Boxplots

Boxplots are a great way to visualise univariate data because they represent statistics such as the 25th percentile, 50th percentile, etc.

```
In [ ]: # boxplot
sns.boxplot(df['Order_Quantity'])
# plt.title('Order Quantity')

plt.show()
```

```
In [ ]: # to plot the values on the vertical axis, specify y=variable
sns.boxplot(y=df['Order_Quantity'])
plt.title('Order Quantity')
plt.show()
```

Visualising Bivariate Distributions

Bivariate distributions are simply two univariate distributions plotted on x and y axes respectively. They help you observe the relationship between the two variables.

They are also called joint distributions and are created using `sns.jointplot()`.

```
In [ ]: # joint plots of Profit and Sales

sns.jointplot('Sales', 'Profit', df)
plt.show()

# same as sns.jointplot(df['Sales'], df['Profit'])
```

Notice that both the distributions are heavily skewed and all the points seem to be concentrated in one region. That is because of some extreme values of Profits and Sales which matplotlib is trying to accomodate in the limited space of the plot.

Let's remove that point and plot again.

```
In [ ]: # remove points having extreme values
df = df[(df.Profit < 100) & (df.Sales < 200)]

sns.jointplot('Sales', 'Profit', df, kind="scatter", space = 0, color="b")
plt.show()
```

You can adjust the arguments of the `jointplot()` to make the plot more readable. For e.g. specifying `kind=hex` will create a 'hexbin plot'.

```
In [ ]: # plotting Low Sales value orders
# hex plot
df = pd.read_csv("./global_sales_data/market_fact.csv")
df = df[(df.Profit < 100) & (df.Profit > -100) & (df.Sales < 200)]
sns.jointplot('Sales', 'Profit', df, kind="hex", color="k")
plt.show()
```

The bottom-right region of the plot represents orders where the Sales is high but the Profit is low, i.e. even when the store is getting high revenue, the orders are still making losses. These are the kind of orders a business would want to avoid.

Data Visualization Session 14

Plotting Pairwise Relationships

You'll find it helpful to plot pairwise relationships between multiple numeric variables. For e.g., here we have taken the prices of some popular cryptocurrencies such as bitcoin, litecoin, ethereum, monero, neo, quantum and ripple.

Now, the crypto enthusiasts would know that the prices of these currencies vary with each other. If bitcoin goes up, the others will likely follow suit, etc.

Now, say you want to trade in some currencies. Given a set of cryptocurrencies, how will you decide when and which one to buy/sell? It will be helpful to analyse past data and identify some trends in these currencies.

```
In [14]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

btc = pd.read_csv("crypto_data/bitcoin_price.csv")
ether = pd.read_csv("crypto_data/ethereum_price.csv")
ether.head()
```

	Date	Open	High	Low	Close	Volume	Market Cap
0	Nov 07, 2017	298.57	304.84	290.77	294.66	5407,66,000	28,533,300,000
1	Nov 06, 2017	296.43	305.42	293.72	298.89	5793,59,000	28,322,700,000
2	Nov 05, 2017	300.04	301.37	295.12	296.26	3376,58,000	28,661,500,000
3	Nov 04, 2017	305.48	305.48	295.80	300.47	4164,79,000	29,175,300,000
4	Nov 03, 2017	288.50	308.31	287.69	305.71	6463,40,000	27,547,400,000

In [15]:

```
# reading cryptocurrency files
btc = pd.read_csv("crypto_data/bitcoin_price.csv")
ether = pd.read_csv("crypto_data/ethereum_price.csv")
ltc = pd.read_csv("crypto_data/litecoin_price.csv")
monero = pd.read_csv("crypto_data/monero_price.csv")
neo = pd.read_csv("crypto_data/neo_price.csv")
quantum = pd.read_csv("crypto_data/qtum_price.csv")
ripple = pd.read_csv("crypto_data/ripple_price.csv")

# putting a suffix with column names so that joins are easy
btc.columns = btc.columns.map(lambda x: str(x) + '_btc')
ether.columns = ether.columns.map(lambda x: str(x) + '_et')
ltc.columns = ltc.columns.map(lambda x: str(x) + '_ltc')
monero.columns = monero.columns.map(lambda x: str(x) + '_mon')
neo.columns = neo.columns.map(lambda x: str(x) + '_neo')
quantum.columns = quantum.columns.map(lambda x: str(x) + '_qt')
ripple.columns = ripple.columns.map(lambda x: str(x) + '_rip')

btc.head()
```

Out[15]:

	Date_btc	Open_btc	High_btc	Low_btc	Close_btc	Volume_btc	Market Cap_btc
0	Nov 07, 2017	7023.10	7253.32	7023.10	7144.38	2,326,340,000	117,056,000,000
1	Nov 06, 2017	7403.22	7445.77	7007.31	7022.76	3,111,900,000	123,379,000,000
2	Nov 05, 2017	7404.52	7617.48	7333.19	7407.41	2,380,410,000	123,388,000,000
3	Nov 04, 2017	7164.48	7492.86	7031.28	7379.95	2,483,800,000	119,376,000,000
4	Nov 03, 2017	7087.53	7461.29	7002.94	7207.76	3,369,860,000	118,084,000,000

In [19]: # merging all the files by date

```
m1 = pd.merge(btc, ether, how="inner", left_on="Date_btc", right_on="Date_et")
m2 = pd.merge(m1, ltc, how="inner", left_on="Date_btc", right_on="Date_ltc")
m3 = pd.merge(m2, monero, how="inner", left_on="Date_btc", right_on="Date_mon")
m4 = pd.merge(m3, neo, how="inner", left_on="Date_btc", right_on="Date_neo")
m5 = pd.merge(m4, quantum, how="inner", left_on="Date_btc", right_on="Date_qt")
crypto = pd.merge(m5, ripple, how="inner", left_on="Date_btc", right_on="Date_ri")

crypto.head()
```

Out[19]:

	Date_btc	Open_btc	High_btc	Low_btc	Close_btc	Volume_btc	Market Cap_btc	Date_et	Open
0	Nov 07, 2017	7023.10	7253.32	7023.10	7144.38	2,326,340,000	117,056,000,000	Nov 07, 2017	2!
1	Nov 06, 2017	7403.22	7445.77	7007.31	7022.76	3,111,900,000	123,379,000,000	Nov 06, 2017	2!
2	Nov 05, 2017	7404.52	7617.48	7333.19	7407.41	2,380,410,000	123,388,000,000	Nov 05, 2017	3!
3	Nov 04, 2017	7164.48	7492.86	7031.28	7379.95	2,483,800,000	119,376,000,000	Nov 04, 2017	3!
4	Nov 03, 2017	7087.53	7461.29	7002.94	7207.76	3,369,860,000	118,084,000,000	Nov 03, 2017	2!

5 rows × 49 columns

In [26]: # Subsetting only the closing prices column for plotting

```
clos = crypto[["Close_btc", "Close_et", "Close_ltc", "Close_mon", "Close_neo", "Close_qt", "Close_ri"]]
clos.head()
```

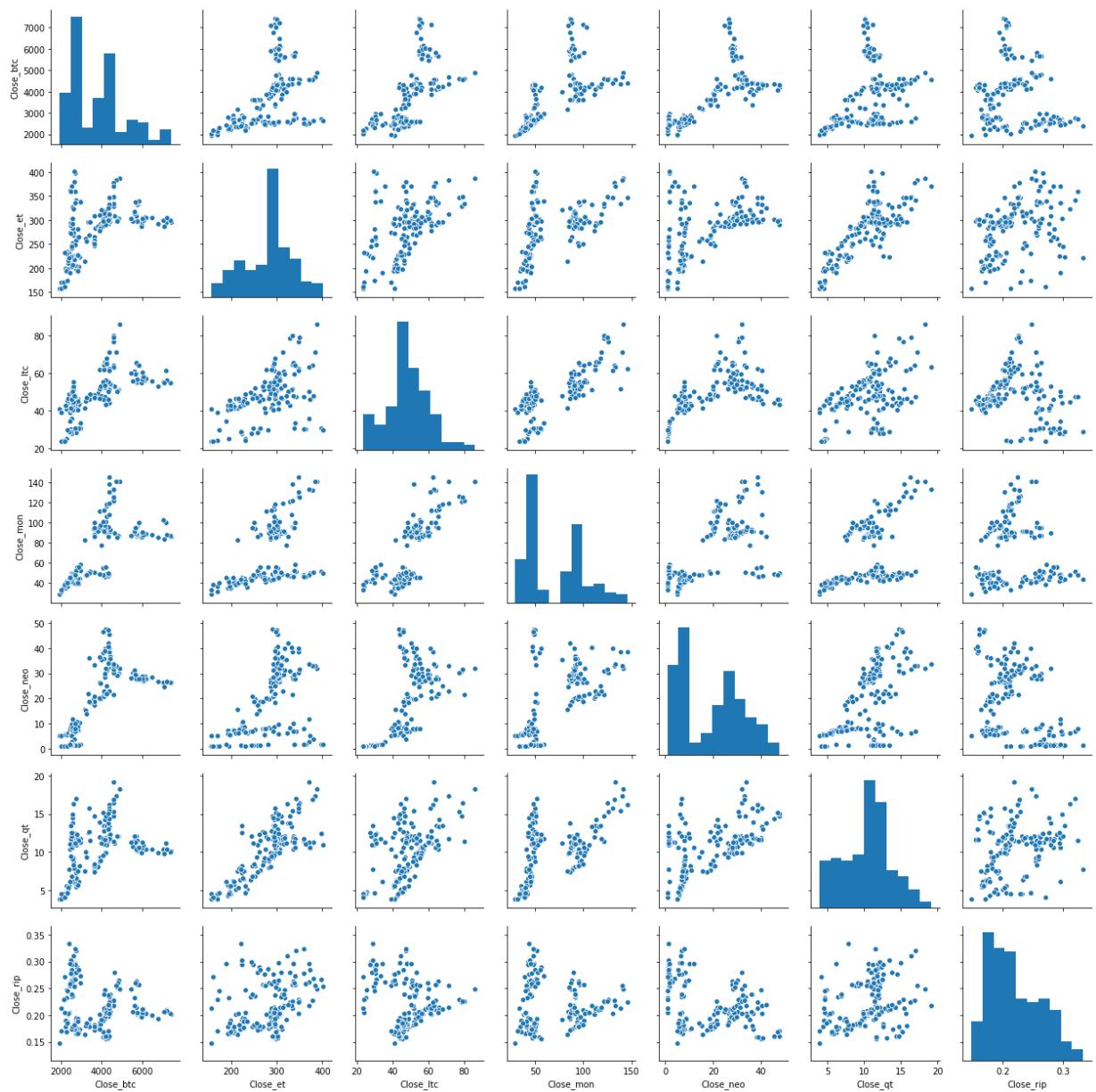
Out[26]:

	Close_btc	Close_et	Close_ltc	Close_mon	Close_neo	Close_qt	Close_ri
0	7144.38	294.66	61.30	99.76	26.23	11.21	0.210354
1	7022.76	298.89	55.17	102.92	26.32	10.44	0.205990
2	7407.41	296.26	54.75	86.35	26.38	10.13	0.202055
3	7379.95	300.47	55.04	87.30	26.49	10.05	0.203750
4	7207.76	305.71	56.18	87.99	26.82	10.38	0.208133

Pairwise Scatter Plots

Now, since we have multiple numeric variables, `sns.pairplot()` is a good choice to plot all of them in one figure.

```
In [27]: # pairplot  
sns.pairplot(clos)  
plt.show()
```



```
In [28]: # You can also observe the correlation between the currencies
# using df.corr()
cor = clos.corr()
round(cor, 3)
```

Out[28]:

	Close_btc	Close_et	Close_ltc	Close_mon	Close_neo	Close_qt	Close_rip
Close_btc	1.000	0.449	0.658	0.697	0.735	0.382	-0.198
Close_et	0.449	1.000	0.490	0.539	0.482	0.791	0.332
Close_ltc	0.658	0.490	1.000	0.793	0.641	0.448	-0.187
Close_mon	0.697	0.539	0.793	1.000	0.669	0.518	-0.086
Close_neo	0.735	0.482	0.641	0.669	1.000	0.557	-0.375
Close_qt	0.382	0.791	0.448	0.518	0.557	1.000	0.292
Close_rip	-0.198	0.332	-0.187	-0.086	-0.375	0.292	1.000

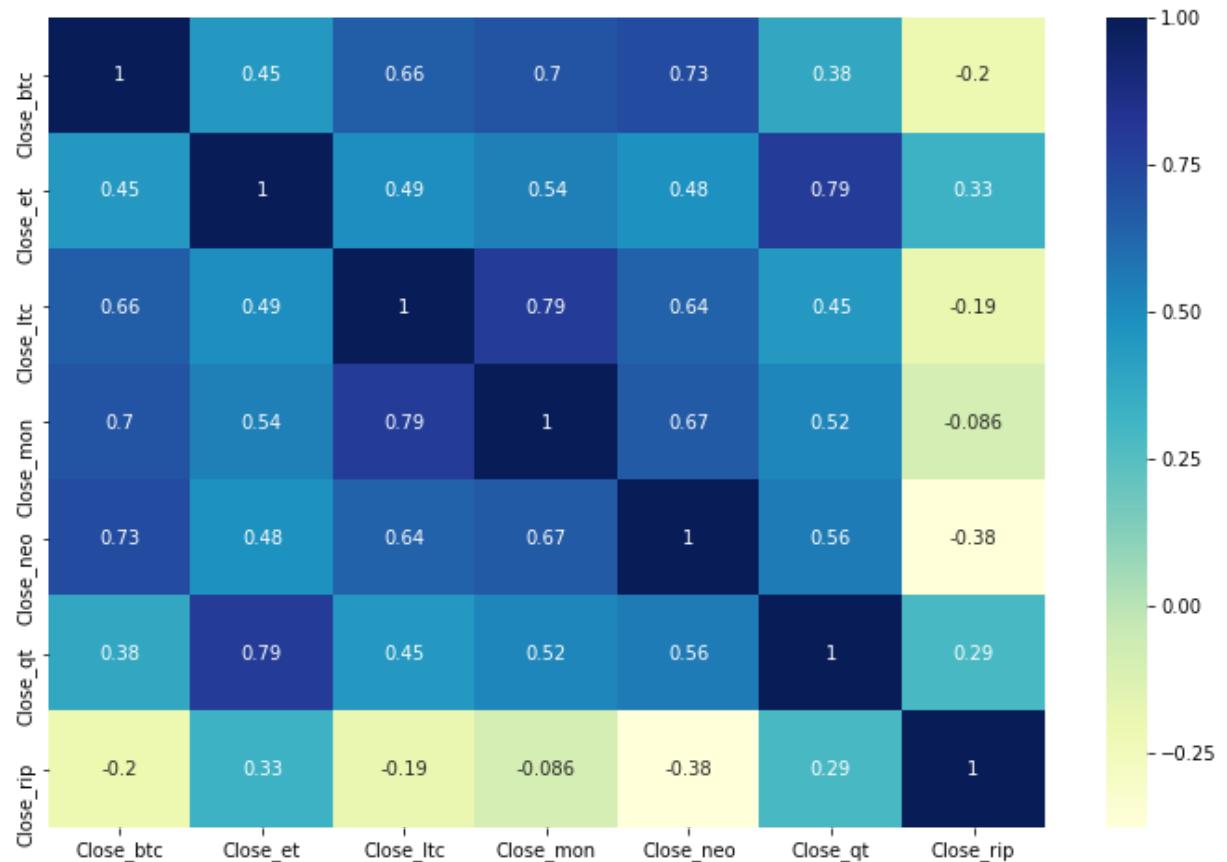
The dataframe above is a **correlation matrix** of cryptocurrencies. Try finding some important relationships between currencies. Notice that quantum and ethereum are highly correlated (0.79).

Heatmaps

It will be helpful to visualise the correlation matrix itself using `sns.heatmap()`.

```
In [29]: # figure size
plt.figure(figsize=(12,8))

# heatmap
sns.heatmap(cor, cmap="YlGnBu", annot=True)
plt.show()
```



The orange boxes show the most correlated currencies. Specifically, **ethereum-quantum (0.79)** and **monero-ltc (0.79)** are the most correlated pairs. Also, **neo-btc (0.73)** are quite highly correlated.

Please note that this data is from a specific time period only.

Thus, from a risk-minimisation point of view, you should not invest in these pairs of cryptocurrencies, since if one goes down, the other is likely to go down as well (but yes, if one goes up, you'll become filthy rich).

Data Visualization Session 13

Plotting Categorical Data

In this section, we will:

- Plot distributions of data across categorical variables
- Plot aggregate/summary statistics across categorical variables

Plotting Distributions Across Categories

We have seen how to plot distributions of data. Often, the distributions reveal new information when you plot them across categorical variables.

Let's see some examples.

```
In [31]: # Loading Libraries and reading the data
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

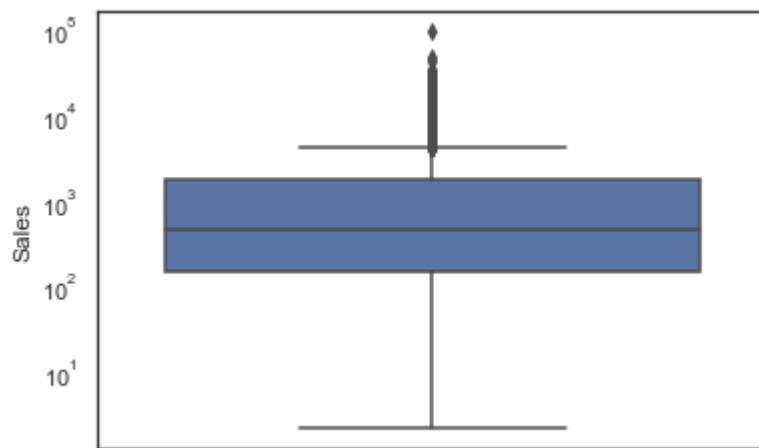
# set seaborn theme if you prefer
sns.set(style="white")

# read data
market_df = pd.read_csv("./global_sales_data/market_fact.csv")
customer_df = pd.read_csv("./global_sales_data/cust_dimen.csv")
product_df = pd.read_csv("./global_sales_data/prod_dimen.csv")
shipping_df = pd.read_csv("./global_sales_data/shipping_dimen.csv")
orders_df = pd.read_csv("./global_sales_data/orders_dimen.csv")
```

Boxplots

We had created simple boxplots such as the ones shown below. Now, let's plot multiple boxplots and see what they can tell us the distribution of variables across categories.

```
In [32]: # boxplot of a variable
sns.boxplot(y=market_df['Sales'])
plt.yscale('log')
plt.show()
```

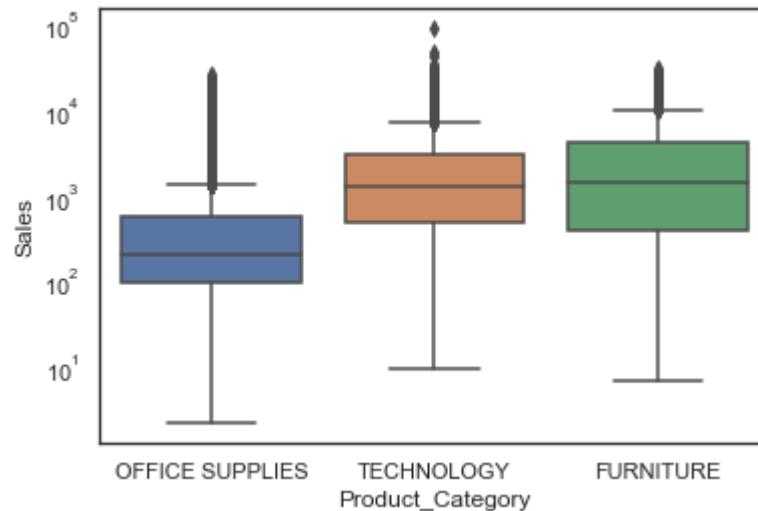


Now, let's say you want to **compare the (distribution of) sales of various product categories**. Let's first merge the product data into the main dataframe.

```
In [33]: # merge the dataframe to add a categorical variable
df = pd.merge(market_df, product_df, how='inner', on='Prod_id')
df.head()
```

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_C
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	3
1	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.05	0.04	27	23.12	3
2	Ord_5484	Prod_16	SHP_7663	Cust_1820	322.82	0.05	35	-17.58	3
3	Ord_3730	Prod_16	SHP_5175	Cust_1314	459.08	0.04	34	61.57	3
4	Ord_4143	Prod_16	SHP_5771	Cust_1417	207.21	0.06	24	-78.64	6

```
In [34]: # boxplot of a variable across various product categories
sns.boxplot(x='Product_Category', y='Sales', data=df)
plt.yscale('log')
plt.show()
```

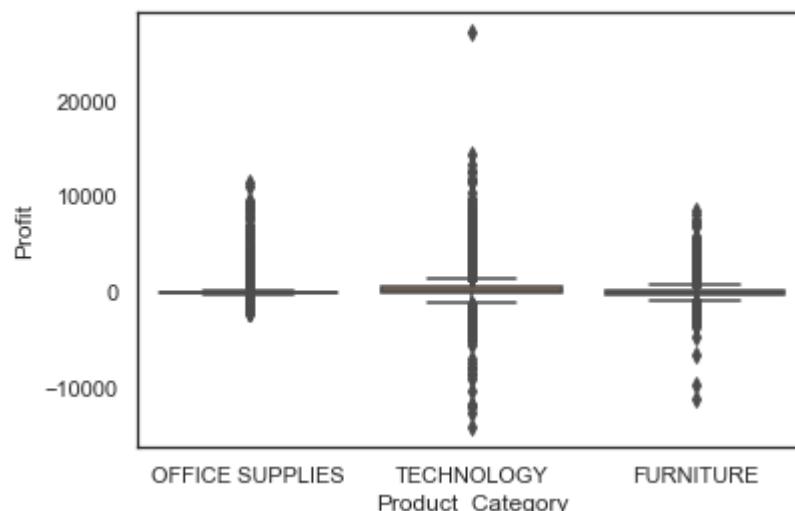


So this tells you that the sales of office supplies are, on an average, lower than the other two categories. The sales of technology and furniture categories seem much better. Note that each order can have multiple units of products sold, so Sales being higher/lower may be due to price per unit or the number of units.

Let's now plot the other important variable - Profit.

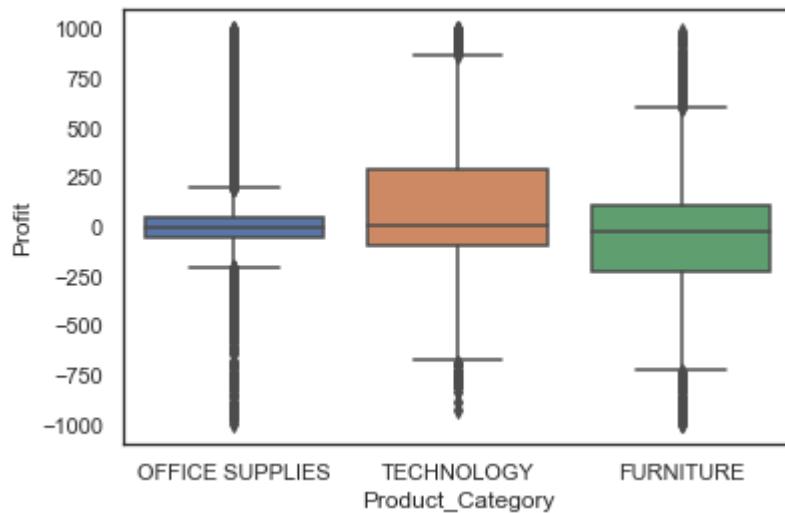
```
In [35]: # boxplot of a variable across various product categories
import seaborn as sns
sns.boxplot(x='Product_Category', y='Profit', data=df)

plt.show()
```



Profit clearly has some *outliers* due to which the boxplots are unreadable. Let's remove some extreme values from Profit (for the purpose of visualisation) and try plotting.

```
In [36]: df = df[(df.Profit<1000) & (df.Profit>-1000)]  
  
# boxplot of a variable across various product categories  
sns.boxplot(x='Product_Category', y='Profit', data=df)  
plt.show()
```



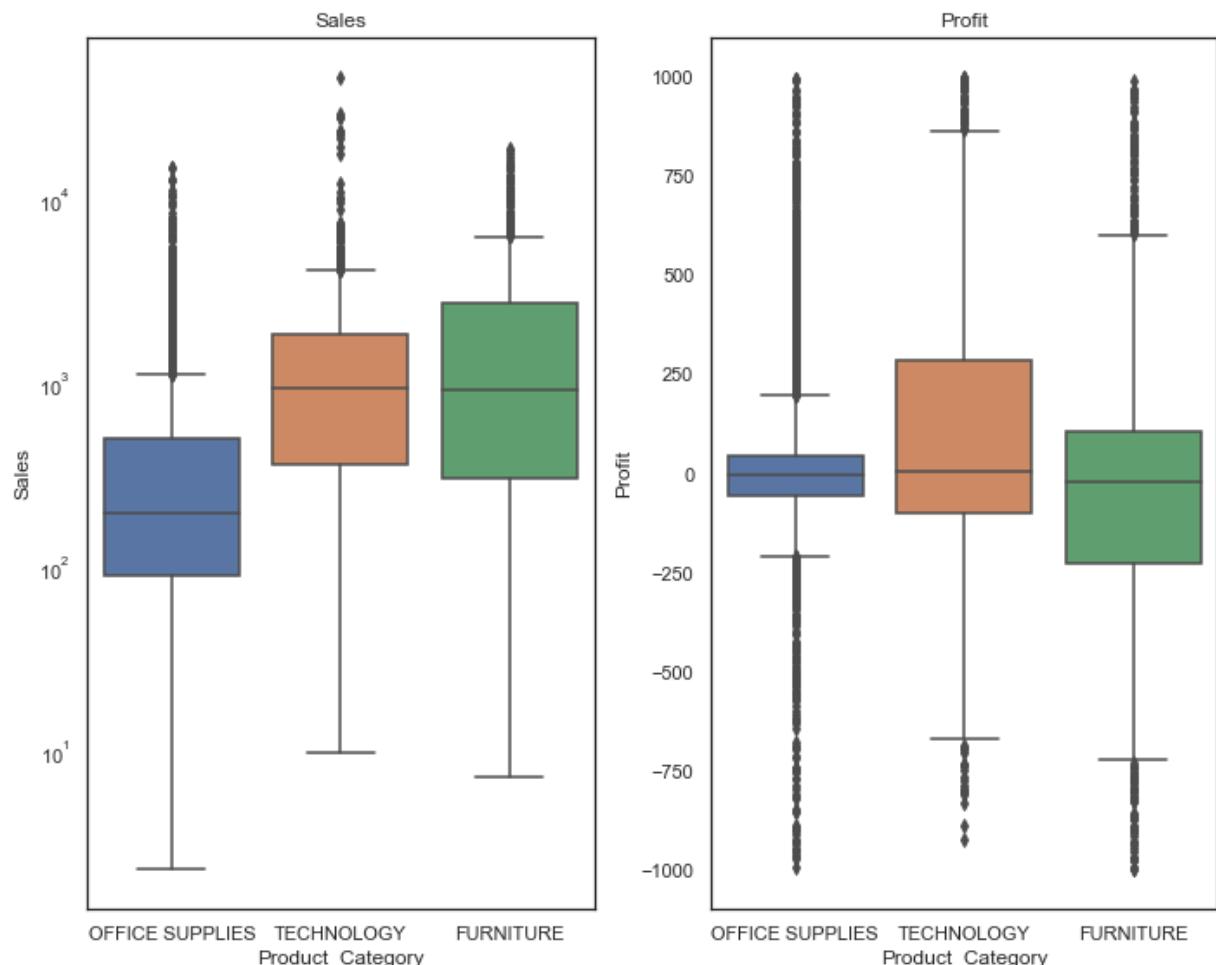
You can see that though the category 'Technology' has better sales numbers than others, it is also the one where the **most loss making transactions** happen. You can drill further down into this.

```
In [37]: # adjust figure size
plt.figure(figsize=(10, 8))

# subplot 1: Sales
plt.subplot(1, 2, 1)
sns.boxplot(x='Product_Category', y='Sales', data=df)
plt.title("Sales")
plt.yscale('log')

# subplot 2: Profit
plt.subplot(1, 2, 2)
sns.boxplot(x='Product_Category', y='Profit', data=df)
plt.title("Profit")

plt.tight_layout()
plt.show()
```



Now that we've compared Sales and Profits across product categories, let's drill down further and do the same across **another categorical variable** - Customer_Segment.

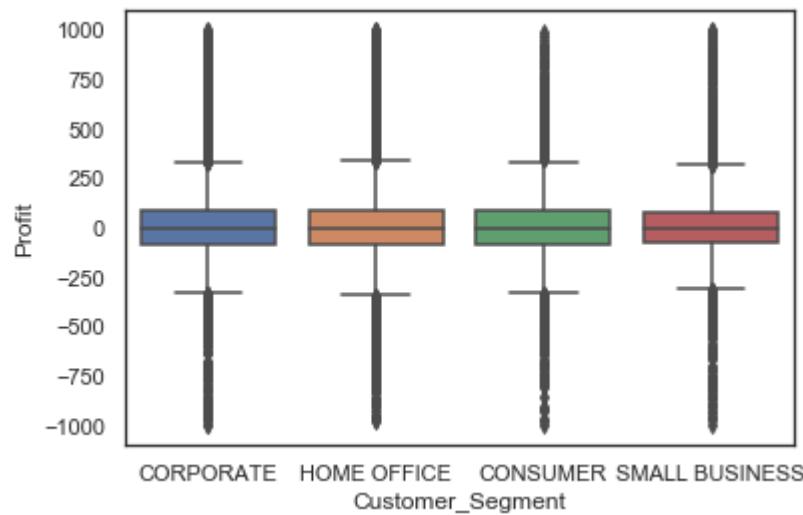
We'll need to add the customer-related attributes (dimensions) to this dataframe.

In [38]: # merging with customers df
df = pd.merge(df, customer_df, how='inner', on='Cust_id')
df.head()

Out[38]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01		23	-30.51
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01		13	4.56
2	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09		43	729.34
3	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.02	0.03		23	-47.64
4	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.05	0.04		27	23.12

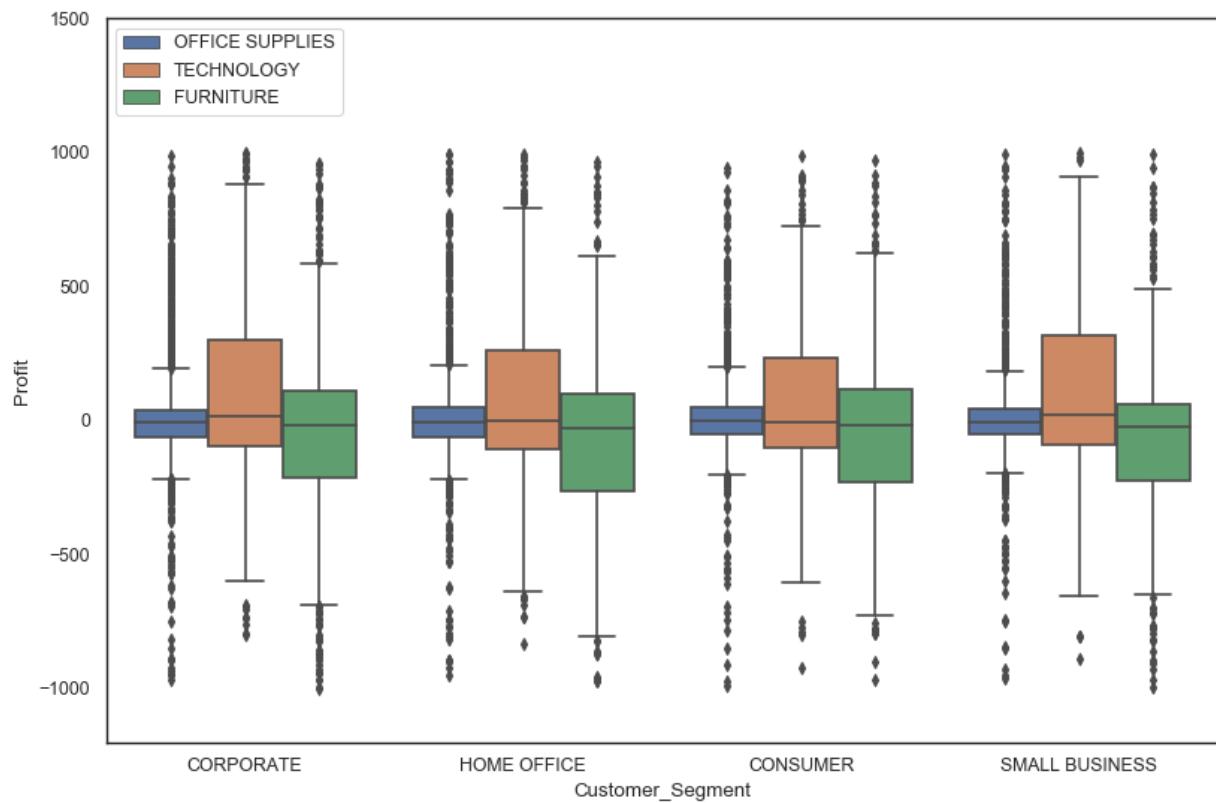
In [39]: # boxplot of a variable across various product categories
sns.boxplot(x='Customer_Segment', y='Profit', data=df)
plt.show()



You can **visualise the distribution across two categorical variables** using the `hue=` argument.

```
In [40]: # set figure size for larger figure
plt.figure(num=None, figsize=(12, 8), dpi=80, facecolor='w', edgecolor='k')

# specify hue="categorical_variable"
sns.boxplot(x='Customer_Segment', y='Profit', hue="Product_Category", data=df)
plt.legend(loc='upper left')
plt.ylim(-1200,1500)
plt.show()
```

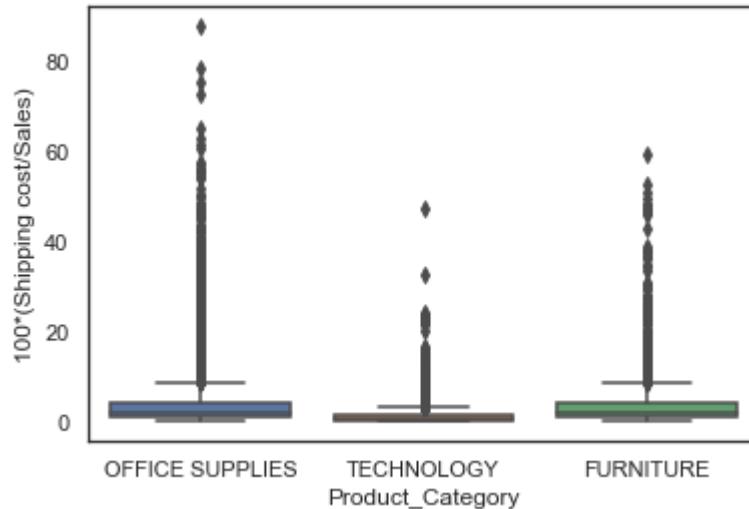


Across all customer segments, the product category `Technology` seems to be doing fairly well, though `Furniture` is incurring losses across all segments.

Now say you are curious to know why certain orders are making huge losses. One of your hypothesis is that the *shipping cost is too high in some orders*. You can **plot derived variables** as well, such as *shipping cost as percentage of sales amount*.

Data Visualization Session 14

```
In [41]: # plot shipping cost as percentage of Sales amount  
sns.boxplot(x=df['Product_Category'], y=100*df['Shipping_Cost']/df['Sales'])  
plt.ylabel("100*(Shipping cost/Sales)")  
plt.show()
```



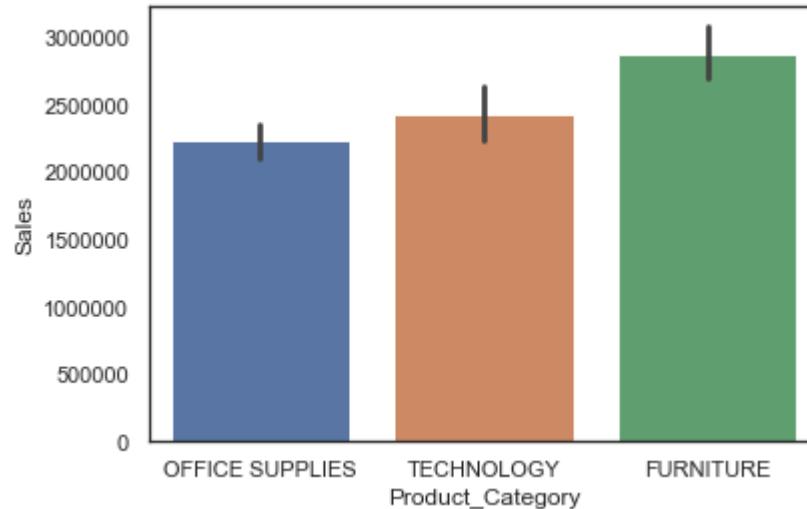
Plotting Aggregated Values across Categories

Bar Plots - Mean, Median and Count Plots

Bar plots are used to **display aggregated values** of a variable, rather than entire distributions. This is especially useful when you have a lot of data which is difficult to visualise in a single figure.

For example, say you want to visualise and *compare the average Sales across Product Categories*. The `sns.barplot()` function can be used to do that.

```
In [45]: # bar plot with default statistic=mean  
sns.barplot(x='Product_Category', y='Sales', data=df, estimator = np.sum)  
plt.show()
```



Note that, **by default, seaborn plots the mean value across categories**, though you can plot the count, median, sum etc. Also, barplot computes and shows the confidence interval of the mean as well.

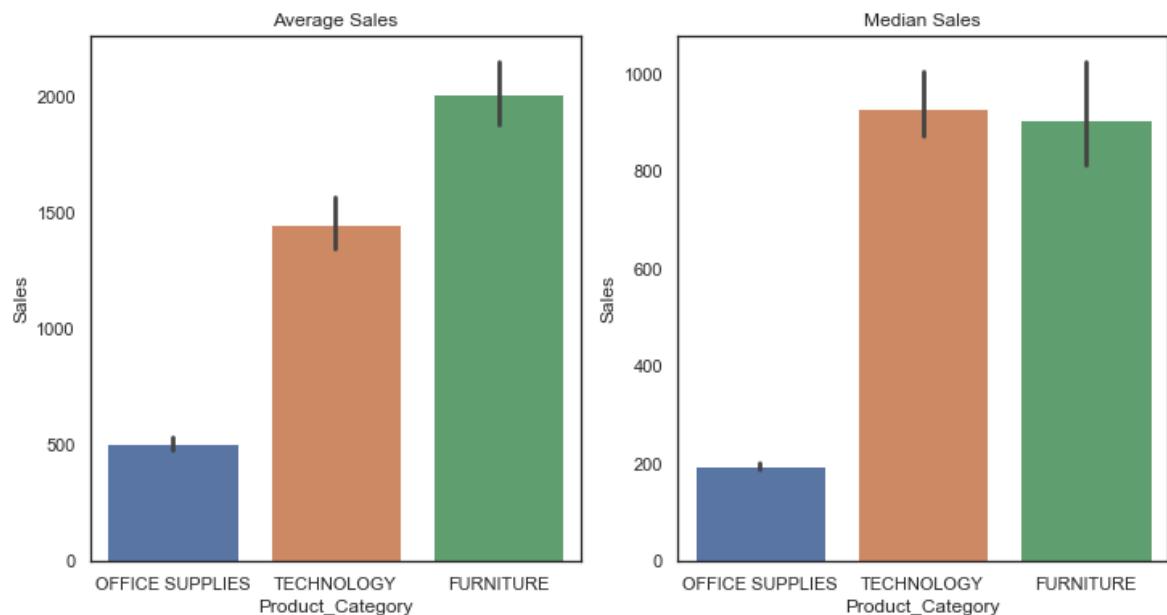
```
In [44]: # Create 2 subplots for mean and median respectively
```

```
# increase figure size
plt.figure(figsize=(12, 6))

# subplot 1: statistic=mean
plt.subplot(1, 2, 1)
sns.barplot(x='Product_Category', y='Sales', data=df)
plt.title("Average Sales")

# subplot 2: statistic=median
plt.subplot(1, 2, 2)
sns.barplot(x='Product_Category', y='Sales', data=df, estimator=np.median)
plt.title("Median Sales")

plt.show()
```

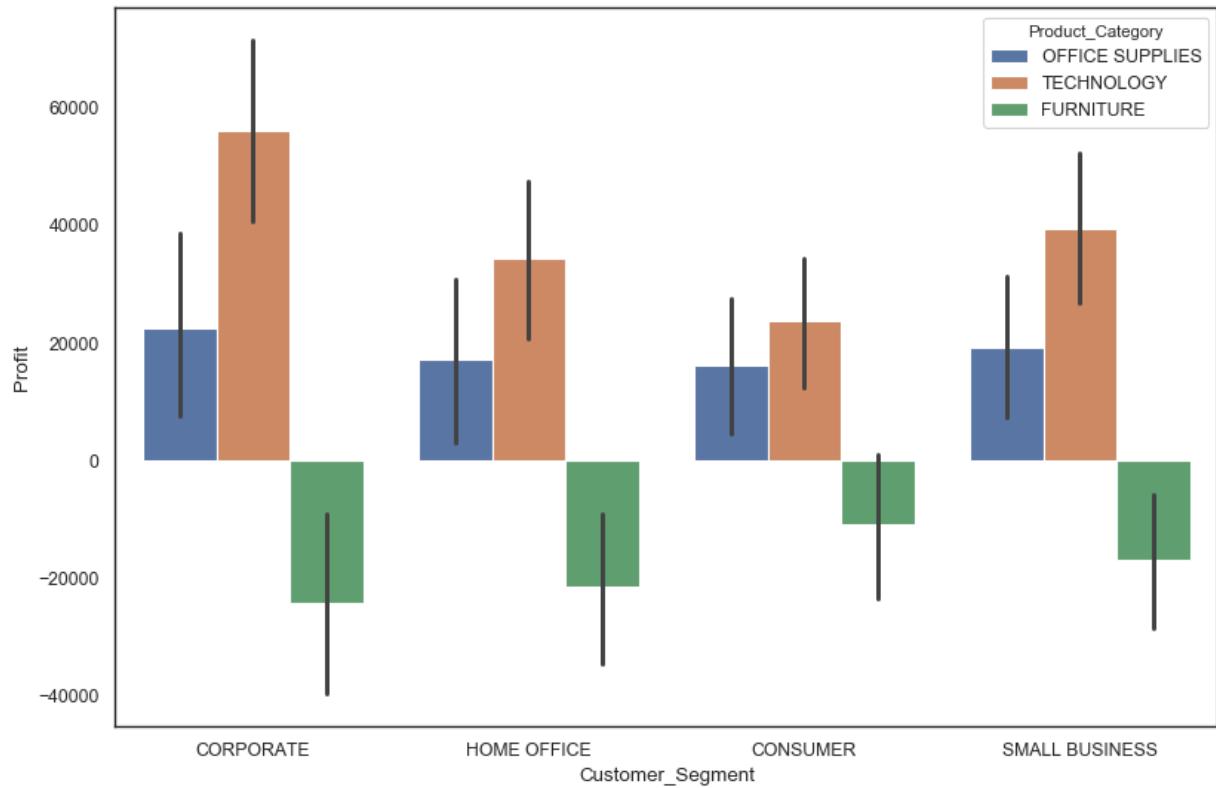


Look at that! The mean and median sales across the product categories tell different stories. This is because of some outliers (extreme values) in the Furniture category, distorting the value of the mean.

You can add another categorical variable in the plot.

```
In [46]: # set figure size for larger figure
plt.figure(num=None, figsize=(12, 8), dpi=80)

# specify hue="categorical_variable"
sns.barplot(x='Customer_Segment', y='Profit', hue="Product_Category", data=df, e:
plt.show()
```



The plot neatly shows the median profit across product categories and customer segments. It says that:

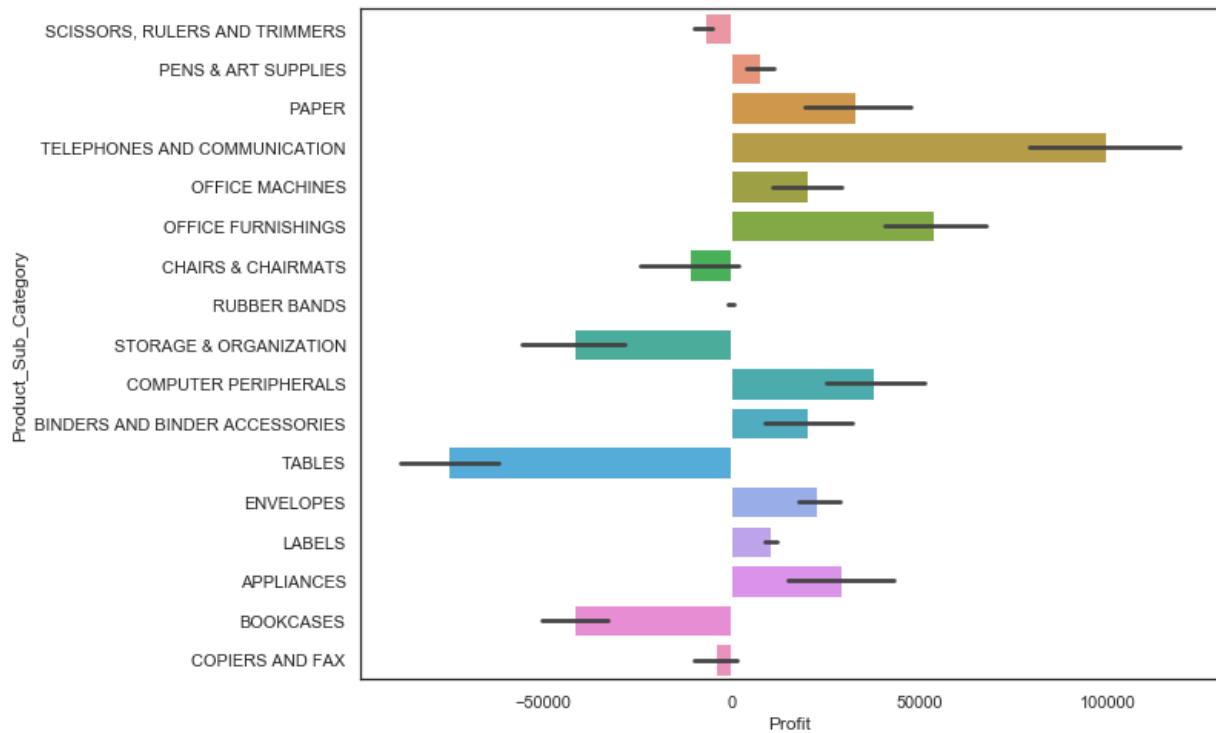
- On an average, only Technology products in Small Business and Corporate (customer) categories are profitable.
- Furniture is incurring losses across all Customer Segments

Compare this to the boxplot we had created above - though the bar plots contains 'lesser information' than the boxplot, it is more revealing.

When you want to visualise having a large number of categories, it is helpful to plot the categories across the y-axis. Let's now *drill down into product sub categories*.

In [48]: # Plotting categorical variable across the y-axis

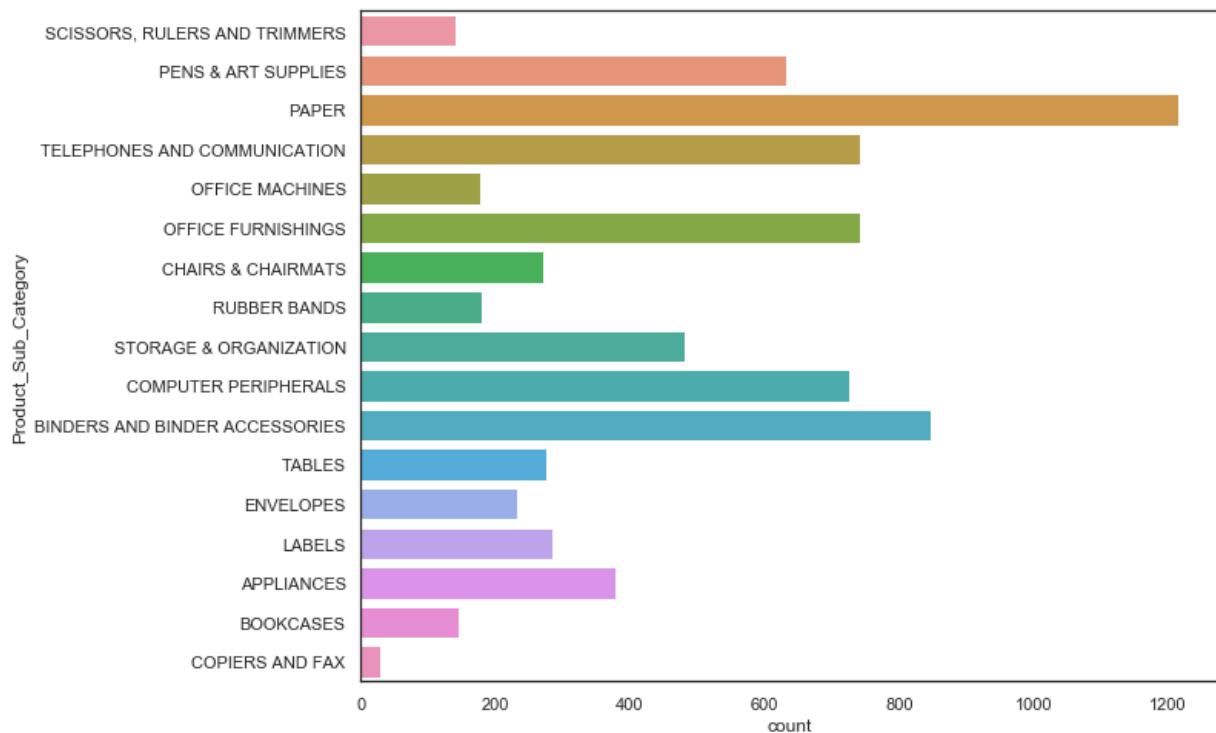
```
plt.figure(figsize=(10, 8))
sns.barplot(x='Profit', y="Product_Sub_Category", data=df, estimator=np.sum)
plt.show()
```



The plot clearly shows which sub categories are incurring the heaviest losses - Copiers and Fax, Tables, Chairs and Chairmats are the most loss making categories.

You can also plot the **count of the observations** across categorical variables using `sns.countplot()` .

```
In [49]: # Plotting count across a categorical variable  
plt.figure(figsize=(10, 8))  
sns.countplot(y="Product_Sub_Category", data=df)  
plt.show()
```



Note that the three most loss making categories - Storage, Tables and bookcases - has a significant number of orders.

Additional Stuff on Plotting Categorical Variables

1. [Seaborn official tutorial on categorical variables](https://seaborn.pydata.org/tutorial/categorical.html)
(<https://seaborn.pydata.org/tutorial/categorical.html>)

Data Visualization Session 14

Visualising Time Series Data

In the section, we will explore ways to visualise data gathered over time. We will:

- Plot simple time series plots
- Derive variables such as month and year and use them for richer visualisations

In [20]: *# Loading Libraries and reading the data*

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# set seaborn theme if you prefer
sns.set(style="whitegrid")

# read data
market_df = pd.read_csv("./global_sales_data/market_fact.csv")
customer_df = pd.read_csv("./global_sales_data/cust_dimen.csv")
product_df = pd.read_csv("./global_sales_data/prod_dimen.csv")
shipping_df = pd.read_csv("./global_sales_data/shipping_dimen.csv")
orders_df = pd.read_csv("./global_sales_data/orders_dimen.csv")
```

Visualising Simple Time Series Data

Let's say you want to visualise numeric variables such as `Sales` , `Profit` , `Shipping_Cost` etc. over time.

In [22]: `market_df.head()`

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	4.56	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	729.34	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	1219.87	

In [21]: `orders_df.head()`

Out[21]:

	Order_ID	Order_Date	Order_Priority	Ord_id
0	3	13-10-2010	LOW	Ord_1
1	293	01-10-2012	HIGH	Ord_2
2	483	10-07-2011	HIGH	Ord_3
3	515	28-08-2010	NOT SPECIFIED	Ord_4
4	613	17-06-2011	HIGH	Ord_5

Since the `Order_Date` variable is in the orders dataframe, let's merge it.

In [23]:

```
# merging with the Orders data to get the Date column
df = pd.merge(market_df, orders_df, how='inner', on='Ord_id')
df.head()
```

Out[23]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01		23	-30.51
1	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00		26	1148.90
2	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.02	0.03		23	-47.64
3	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01		13	4.56
4	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09		43	729.34

```
In [24]: # Now we have the Order_Date in the df
# It is stored as a string (object) currently
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8399 entries, 0 to 8398
Data columns (total 13 columns):
Ord_id           8399 non-null object
Prod_id          8399 non-null object
Ship_id          8399 non-null object
Cust_id          8399 non-null object
Sales            8399 non-null float64
Discount         8399 non-null float64
Order_Quantity   8399 non-null int64
Profit           8399 non-null float64
Shipping_Cost    8399 non-null float64
Product_Base_Margin 8336 non-null float64
Order_ID         8399 non-null int64
Order_Date       8399 non-null object
Order_Priority   8399 non-null object
dtypes: float64(5), int64(2), object(6)
memory usage: 918.6+ KB
```

Since `Order_Date` is a string, we need to convert it into a `datetime` object. You can do that using `pd.to_datetime()`.

```
In [25]: # Convert Order_Date to datetime type
df['Order_Date'] = pd.to_datetime(df['Order_Date'])

# Order_Date is now datetime type
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8399 entries, 0 to 8398
Data columns (total 13 columns):
Ord_id           8399 non-null object
Prod_id          8399 non-null object
Ship_id          8399 non-null object
Cust_id          8399 non-null object
Sales            8399 non-null float64
Discount         8399 non-null float64
Order_Quantity   8399 non-null int64
Profit           8399 non-null float64
Shipping_Cost    8399 non-null float64
Product_Base_Margin 8336 non-null float64
Order_ID         8399 non-null int64
Order_Date       8399 non-null datetime64[ns]
Order_Priority   8399 non-null object
dtypes: datetime64[ns](1), float64(5), int64(2), object(5)
memory usage: 918.6+ KB
```

Now, since on each day, multiple orders were placed, we need to aggregate `Sales` using a metric such as mean, median etc., and then create a time series plot.

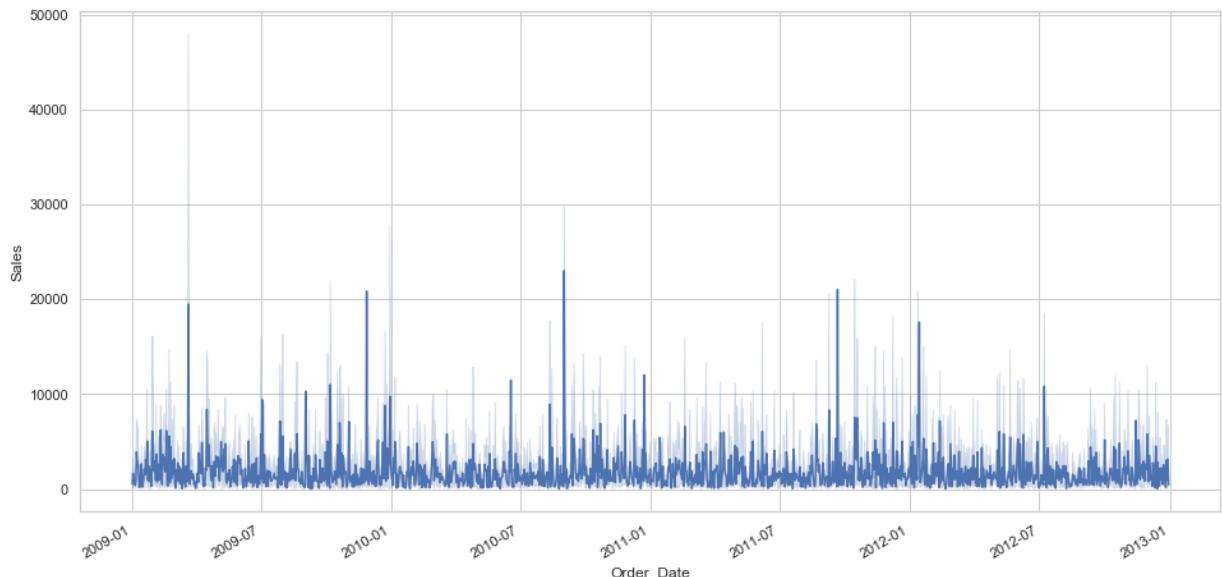
We will group by `Order_Date` and compute the sum of `Sales` on each day.

```
In [26]: # aggregating total sales on each day
time_df = df.groupby('Order_Date')[ 'Sales' ].sum()
print(time_df.head())
print(type(time_df))
```

```
Order_Date
2009-01-01    1052.8400
2009-01-02    5031.9000
2009-01-03    7288.1375
2009-01-04    6188.4245
2009-01-05    2583.3300
Name: Sales, dtype: float64
<class 'pandas.core.series.Series'>
```

We can now create a time-series plot using `sns.tsplot()`.

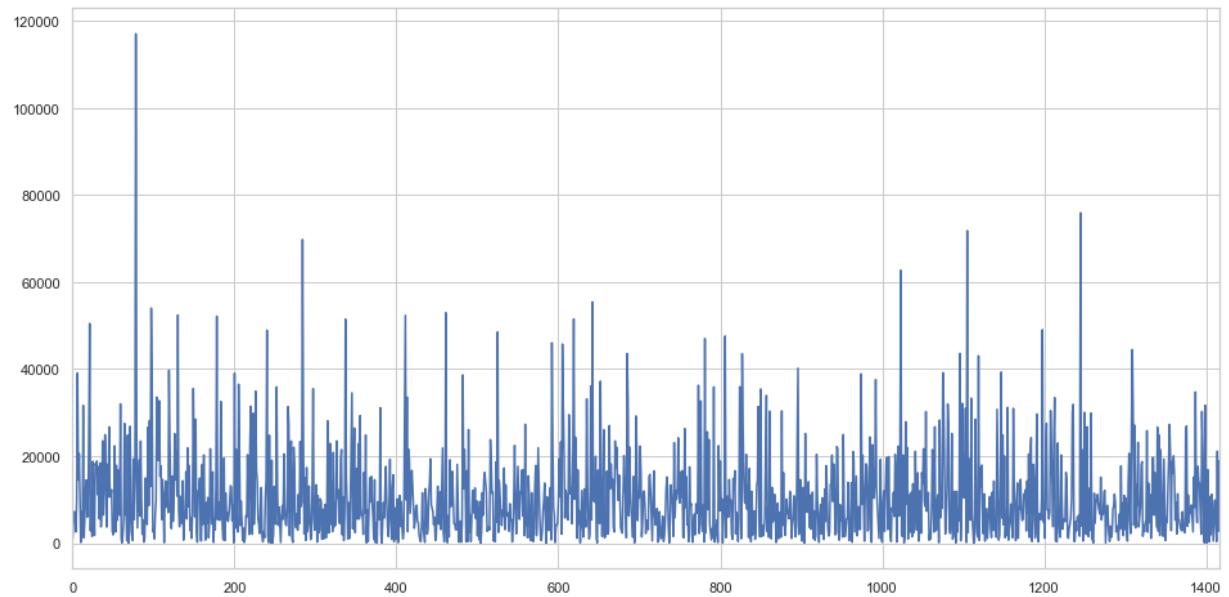
```
In [28]: # time series plot
import matplotlib.dates as mdates
# figure size
fig, ax = plt.subplots(figsize=(16, 8))
# lineplot
sns.lineplot(x='Order_Date',y='Sales',data=df,ax=ax)
ax.format_xdata = mdates.DateFormatter('%Y-%m-%d')
fig.autofmt_xdate()
plt.show()
```



```
In [27]: # time series plot
```

```
# figure size
plt.figure(figsize=(16, 8))

# tsplot
sns.tsplot(time_df)
plt.show()
```



Using Derived Date Metrics for Visualisation

It is often helpful to use derived variables from date such as month and year and using them to identify hidden patterns.

```
In [29]: # extracting month and year from date

# extract month
df['month'] = df['Order_Date'].dt.month

# extract year
df['year'] = df['Order_Date'].dt.year

df.head()
```

Out[29]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping.
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01		23	-30.51
1	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00		26	1148.90
2	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.02	0.03		23	-47.64
3	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01		13	4.56
4	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09		43	729.34

Now you can plot the average sales across years and months.

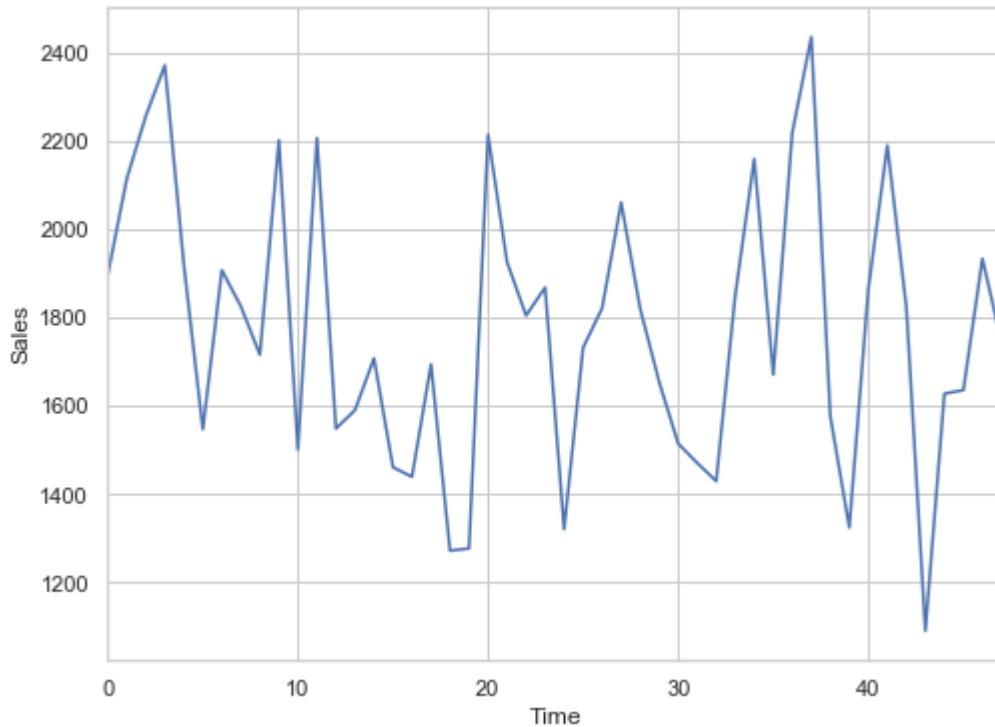
```
In [30]: # grouping by year and month
df_time = df.groupby(["year", "month"]).Sales.mean()
df_time.head()
```

Out[30]:

year	month	Sales
2009	1	1898.475090
	2	2116.510723
	3	2258.661599
	4	2374.155868
	5	1922.317055

Name: Sales, dtype: float64

```
In [31]: plt.figure(figsize=(8, 6))
# time series plot
sns.tsplot(df_time)
plt.xlabel("Time")
plt.ylabel("Sales")
plt.show()
```



There is another way to visualise numeric variables, such as `Sales` , across the year and month. We can pivot the `month` column to create a wide-format dataframe, and then plot a heatmap.

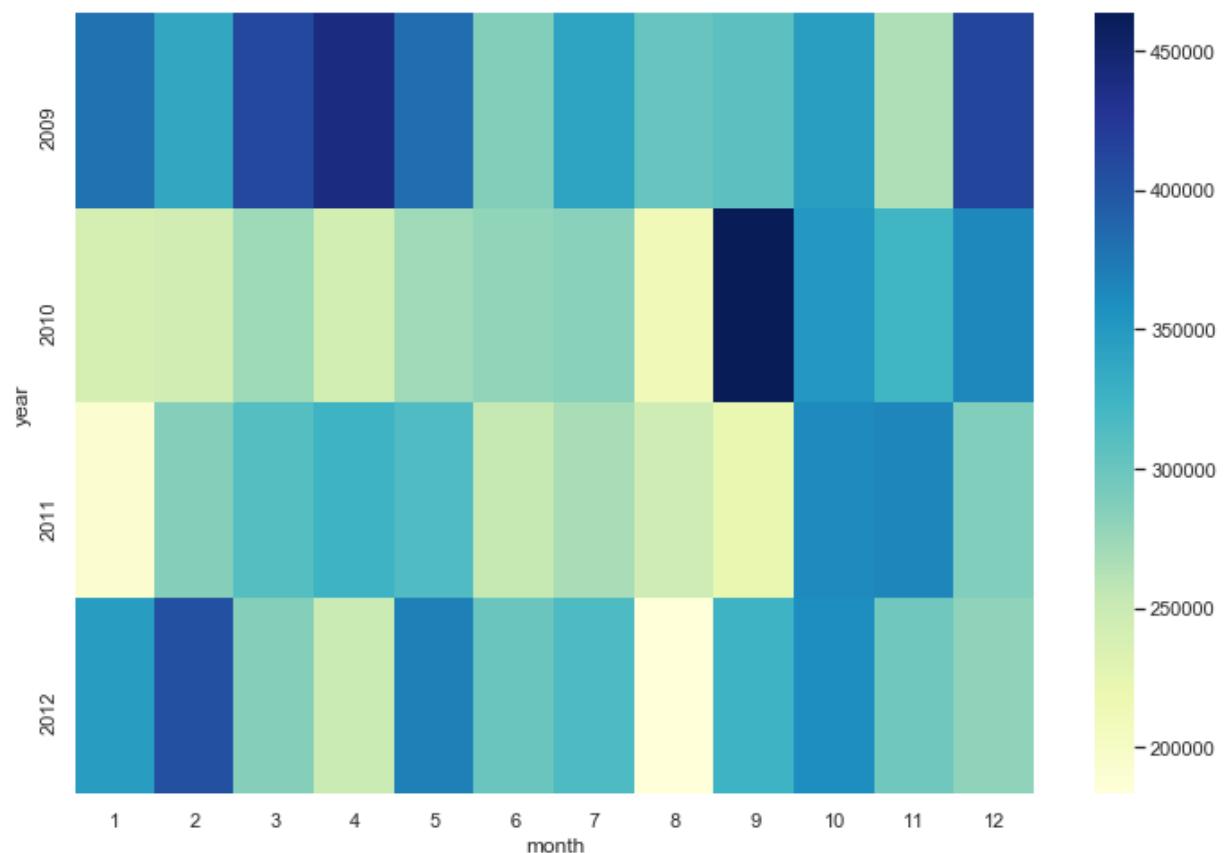
```
In [32]: # Pivoting the data using 'month'
year_month = pd.pivot_table(df, values='Sales', index='year', columns='month', aggfunc='sum')
year_month.head()
```

	month	1	2	3	4	5	6	7
	year							
2009		379695.0180	336525.2050	411076.411	439218.8355	382541.0940	286397.2530	339817.0355
2010		240197.9760	243504.4415	273429.591	242681.2995	272234.3790	279740.5190	283801.7100
2011		191642.3765	286007.3815	311709.165	325909.2735	315211.8710	253306.7455	268256.9980
2012		346449.7220	404735.7155	285669.409	249147.6945	369417.4365	300335.2720	315713.9180

You can now create a heatmap using `sns.heatmap()`.

```
In [33]: # figure size
plt.figure(figsize=(12, 8))

# heatmap with a color map of choice
sns.heatmap(year_month, cmap="YlGnBu")
plt.show()
```



Addtional Reading on Time Series Plots and Heatmaps

1. [Seaborn heatmaps \(documentation\)](https://seaborn.pydata.org/generated/seaborn.heatmap.html)
(<https://seaborn.pydata.org/generated/seaborn.heatmap.html>)



Seaborn

tutorialspoint

SIMPLY EASY LEARNING

network

1-1-h 1-1-th 5-1-h 5-1-th 8-2-h 8-2-th 8-3-h 8-3-th 8-1-h 8-1-th 7-3-h 7-3-th 7-1-h 7-1-th 7-2-h 7-2-th 6-1-h 6-1-th 6-2-h 6-2-th 12-1-h 12-2-h 12-1-th 12-3-h 13-4-h 13-1-h 13-2-h 13-3-h 13-1-th 13-2-th 17-1-h 17-1-th

www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

About the Tutorial

Seaborn is an open source, BSD-licensed Python library providing high level API for visualizing the data using Python programming language.

Audience

This tutorial takes you through the basics and various functions of Seaborn. It is specifically useful for people working on data analysis. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of computer programming terminologies. A basic understanding of Python and any of the programming languages is a plus. Seaborn library is built on top of Matplotlib. Having basic idea of Matplotlib will help you understand this tutorial in a better way.

Copyright & Disclaimer

© Copyright 2017 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial	ii
Audience.....	ii
Prerequisites.....	ii
Copyright & Disclaimer	ii
Table of Contents	iii
1. Seaborn – Introduction	6
Seaborn Vs Matplotlib	6
2. Seaborn – Environment Setup.....	8
Installing Seaborn and getting started	8
3. Seaborn – Importing Datasets and Libraries	10
Importing Libraries	10
Importing Datasets	10
4. Seaborn – Figure Aesthetic	13
Seaborn Figure Styles	16
Removing Axes Spines.....	17
Overriding the Elements	18
Scaling Plot Elements	21
5. Seaborn – Color Palette	23
Building Color Palette.....	23
Qualitative Color Palettes	24
Sequential Color Palettes	25
Diverging Color Palette	25

Setting the Default Color Palette	26
Plotting Univariate Distribution	27
6. Seaborn – Histogram	29
7. Seaborn – Kernel Density Estimates	31
Fitting Parametric Distribution.....	31
Plotting Bivariate Distribution.....	32
Scatter Plot.....	33
Hexbin Plot.....	34
Kernel Density Estimation	35
8. Seaborn – Visualizing Pairwise Relationship.....	37
Axes	37
9. Seaborn – Plotting Categorical Data	1
Categorical Scatter Plots.....	1
10. Seaborn – Distribution of Observations.....	6
Box Plots.....	6
Violin Plots.....	7
11. Seaborn – Statistical Estimation	12
Bar Plot.....	12
Point Plots	14
12. Seaborn – Plotting Wide Form Data.....	16
13. Seaborn – Multi Panel Categorical Plots	20
Factorplot.....	20
What is Facet Grid?	22

14. Seaborn – Linear Relationships	28
Functions to Draw Linear Regression Models	28
Fitting Different Kinds of Models	31
15. Seaborn – Facet Grid	35
Plotting Small Multiples of Data Subsets.....	35
16. Seaborn – Pair Grid	40

1. Seaborn – Introduction

In the world of Analytics, the best way to get insights is by visualizing the data. Data can be visualized by representing it as plots which is easy to understand, explore and grasp. Such data helps in drawing the attention of key elements.

To analyse a set of data using Python, we make use of Matplotlib, a widely implemented 2D plotting library. Likewise, Seaborn is a visualization library in Python. It is built on top of Matplotlib.

Seaborn Vs Matplotlib

It is summarized that if Matplotlib "tries to make easy things easy and hard things possible", Seaborn tries to make a well-defined set of hard things easy too."

Seaborn helps resolve the two major problems faced by Matplotlib; the problems are:

- Default Matplotlib parameters
- Working with data frames

As Seaborn complements and extends Matplotlib, the learning curve is quite gradual. If you know Matplotlib, you are already half way through Seaborn.

Important Features of Seaborn

Seaborn is built on top of Python's core visualization library Matplotlib. It is meant to serve as a complement, and not a replacement. However, Seaborn comes with some very important features. Let us see a few of them here. The features help in -

- Built in themes for styling matplotlib graphics
- Visualizing univariate and bivariate data
- Fitting in and visualizing linear regression models
- Plotting statistical time series data
- Seaborn works well with NumPy and Pandas data structures
- It comes with built in themes for styling Matplotlib graphics

In most cases, you will still use Matplotlib for simple plotting. The knowledge of Matplotlib is recommended to tweak Seaborn's default plots.

2. Seaborn – Environment Setup

In this chapter, we will discuss the environment setup for Seaborn. Let us begin with the installation and understand how to get started as we move ahead.

Installing Seabom and getting started

In this section, we will understand the steps involved in the installation of Seaborn.

Using Pip Installer

To install the latest release of Seaborn, you can use pip:

```
pip install seaborn
```

For Windows, Linux & Mac using Anaconda

Anaconda (from <https://www.continuum.io>) is a free Python distribution for SciPy stack. It is also available for Linux and Mac.

It is also possible to install the released version using conda:

```
conda install seaborn
```

To install the development version of Seaborn directly from github

```
pip install git+https://github.com/mwaskom/seaborn.git
```

Dependencies

Consider the following dependencies of Seaborn:

- Python 2.7 or 3.4+
- numpy
- scipy
- pandas
- matplotlib

3. Seaborn – Importing Datasets and Libraries

In this chapter, we will discuss how to import Datasets and Libraries. Let us begin by understanding how to import libraries.

Importing Libraries

Let us start by importing Pandas, which is a great library for managing relational (table-formatted) datasets. Seaborn comes handy when dealing with DataFrames, which is most widely used data structure for data analysis.

The following command will help you import Pandas:

```
# Pandas for managing datasets  
import pandas as pd
```

Now, let us import the Matplotlib library, which helps us customize our plots.

```
# Matplotlib for additional customization  
from matplotlib import pyplot as plt
```

We will import the Seaborn library with the following command:

```
# Seaborn for plotting and styling  
import seaborn as sb
```

Importing Datasets

We have imported the required libraries. In this section, we will understand how to import the required datasets.

Seaborn comes with a few important datasets in the library. When Seaborn is installed, the datasets download automatically.

You can use any of these datasets for your learning. With the help of the following function you can load the required dataset:

```
load_dataset()
```

Importing Data as Pandas DataFrame

In this section, we will import a dataset. This dataset loads as Pandas DataFrame by default. If there is any function in the Pandas DataFrame, it works on this DataFrame.

The following line of code will help you import the dataset:

```
# Seaborn for plotting and styling
import seaborn as sb
df = sb.load_dataset('tips')
print df.head()
```

The above line of code will generate the following output:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

To view all the available data sets in the Seaborn library, you can use the following command with the **get_dataset_names()** function as shown below:

```
import seaborn as sb
print sb.get_dataset_names()
```

The above line of code will return the list of datasets available as the following output

```
[u'anscombe', u'attention', u'brain_networks', u'car_crashes', u'dots',
u'exercise', u'flights', u'fmri', u'gammas', u'iris', u'planets', u'tips',
u'titanic']
```

DataFrames store data in the form of rectangular grids by which the data can be overviewed easily. Each row of the rectangular grid contains values of an instance, and

each column of the grid is a vector which holds data for a specific variable. This means that rows of a DataFrame do not need to contain values of same data type, they can be numeric, character, logical, etc. DataFrames for Python come with the Pandas library, and they are defined as two-dimensional labeled data structures with potentially different types of columns.

For more details on DataFrames, visit our [tutorial](#) on pandas.

4. Seaborn – Figure Aesthetic

Visualizing data is one step and further making the visualized data more pleasing is another step. Visualization plays a vital role in communicating quantitative insights to an audience to catch their attention.

Aesthetics means a set of principles concerned with the nature and appreciation of beauty, especially in art. Visualization is an art of representing data in effective and easiest possible way.

Matplotlib library highly supports customization, but knowing what settings to tweak to achieve an attractive and anticipated plot is what one should be aware of to make use of it. Unlike Matplotlib, Seaborn comes packed with customized themes and a high-level interface for customizing and controlling the look of Matplotlib figures.

Example

```
import numpy as np

from matplotlib import pyplot as plt

def sinplot(flip=1):

    x = np.linspace(0, 14, 100)

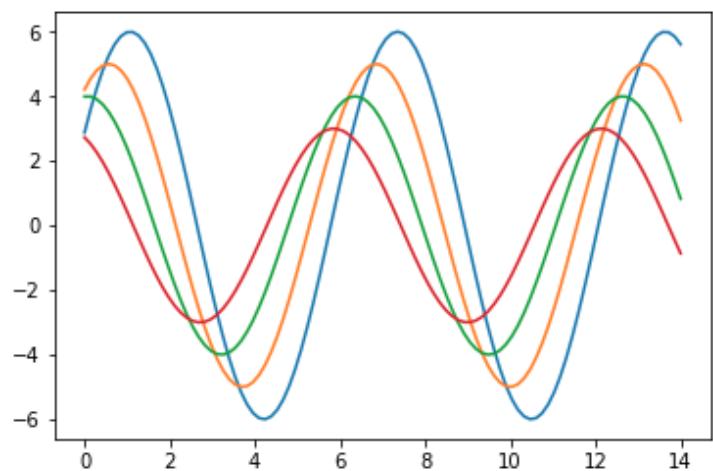
    for i in range(1, 5):

        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)

sinplot()

plt.show()
```

This is how a plot looks with the defaults Matplotlib:



To change the same plot to Seaborn defaults, use the **set()** function:

Example

```
import numpy as np

from matplotlib import pyplot as plt

def sinplot(flip=1):

    x = np.linspace(0, 14, 100)

    for i in range(1, 5):

        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)

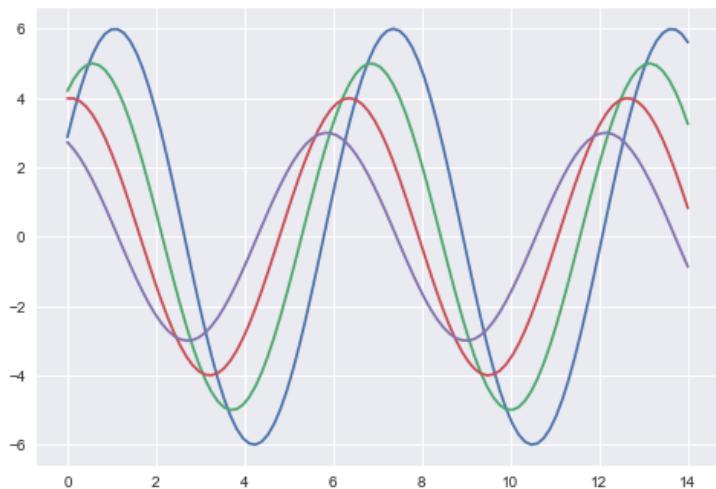
import seaborn as sb

sb.set()

sinplot()

plt.show()
```

Output



The above two figures show the difference in the default Matplotlib and Seaborn plots. The representation of data is same, but the representation style varies in both.

Basically, Seaborn splits the Matplotlib parameters into two groups:

- Plot styles
- Plot scale

Seaborn Figure Styles

The interface for manipulating the styles is **set_style()**. Using this function you can set the theme of the plot. As per the latest updated version, below are the five themes available.

- Darkgrid
- Whitegrid
- Dark
- White
- Ticks

Let us try applying a theme from the above-mentioned list. The default theme of the plot will be **darkgrid** which we have seen in the previous example.

Example

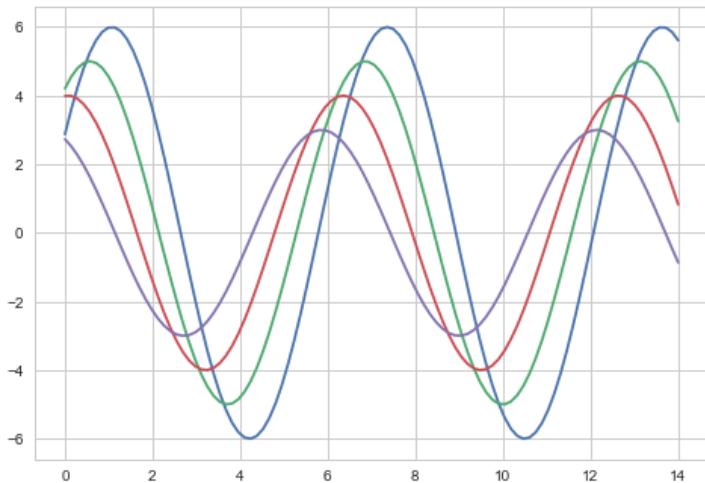
```
import numpy as np

from matplotlib import pyplot as plt

def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 5):
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)

import seaborn as sb
sb.set_style("whitegrid")
sinplot()
plt.show()
```

Output



The difference between the above two plots is the background color.

Removing Axes Spines

In the white and ticks themes, we can remove the top and right axis spines using the `despine()` function.

Example

```
import numpy as np

from matplotlib import pyplot as plt

def sinplot(flip=1):
    x = np.linspace(0, 14, 100)

    for i in range(1, 5):

        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)

import seaborn as sb

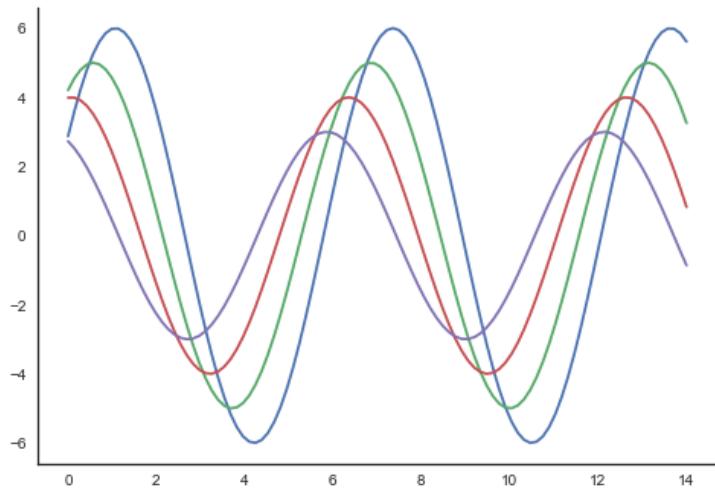
sb.set_style("white")

sinplot()

sb.despine()

plt.show()
```

Output



In the regular plots, we use left and bottom axes only. Using the **despine()** function, we can avoid the unnecessary right and top axes spines, which is not supported in Matplotlib.

Overriding the Elements

If you want to customize the Seaborn styles, you can pass a dictionary of parameters to the **set_style()** function. Parameters available are viewed using **axes_style()** function.

Example

```
import seaborn as sb
print sb.axes_style
```

Output

```
{'axes.axisbelow': False,
'axes.edgecolor': 'white',
'axes.facecolor': '#EAEAF2',
'axes.grid': True,
'axes.labelcolor': '.15',
```

```
'axes.linewidth': 0.0,  
'figure.facecolor': 'white',  
'font.family': [u'sans-serif'],  
'font.sans-serif': [u'Arial',  
    u'Liberation Sans',  
    u'Bitstream Vera Sans',  
    u'sans-serif'],  
'grid.color': 'white',  
'grid.linestyle': u'-',  
'image.cmap': u'Greys',  
'legend.frameon': False,  
'legend.numpoints': 1,  
'legend.scatterpoints': 1,  
'lines.solid_capstyle': u'round',  
'text.color': '.15',  
'xtick.color': '.15',  
'xtick.direction': u'out',  
'xtick.major.size': 0.0,  
'xtick.minor.size': 0.0,  
'ytick.color': '.15',  
'ytick.direction': u'out',  
'ytick.major.size': 0.0,  
'ytick.minor.size': 0.0}
```

Altering the values of any of the parameter will alter the plot style.

Example

```
import numpy as np

from matplotlib import pyplot as plt

def sinplot(flip=1):

    x = np.linspace(0, 14, 100)

    for i in range(1, 5):

        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)

import seaborn as sb

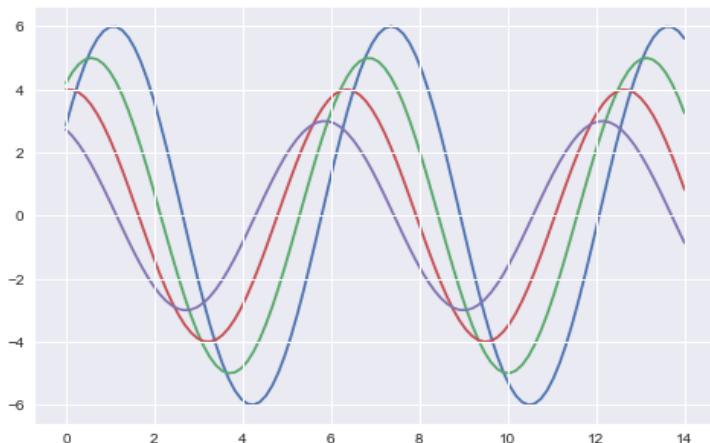
sb.set_style("darkgrid", {'axes.axisbelow': False})

sinplot()

sb.despine()

plt.show()
```

Output



Scaling Plot Elements

We also have control on the plot elements and can control the scale of plot using the **set_context()** function. We have four preset templates for contexts, based on relative size, the contexts are named as follows:

- Paper
- Notebook
- Talk
- Poster

By default, context is set to notebook; and was used in the plots above.

Example

```
import numpy as np

from matplotlib import pyplot as plt

def sinplot(flip=1):

    x = np.linspace(0, 14, 100)

    for i in range(1, 5):

        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)

import seaborn as sb

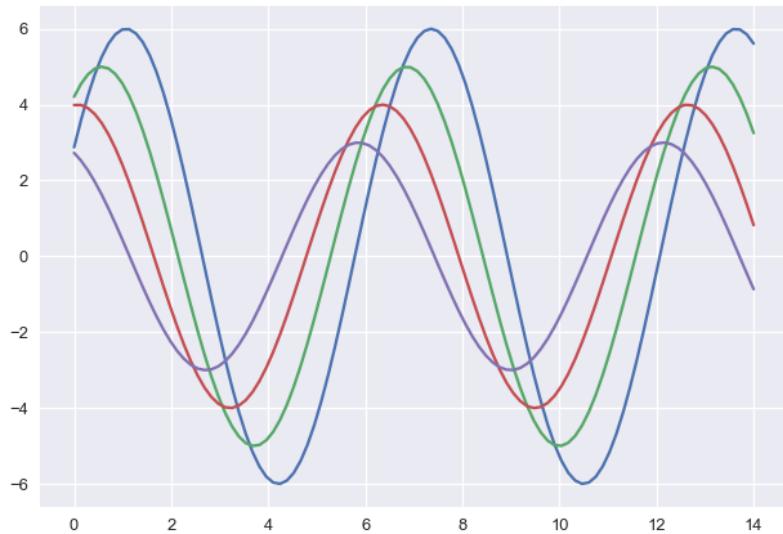
sb.set_style("darkgrid", {'axes.axisbelow': False})

sinplot()

sb.despine()

plt.show()
```

Output



The output size of the actual plot is bigger in size when compared to the above plots.

Note: Due to scaling of images on our web page, you might miss the actual difference in our example plots.

5. Seaborn – Color Palette

Color plays an important role than any other aspect in the visualizations. When used effectively, color adds more value to the plot. A palette means a flat surface on which a painter arranges and mixes paints.

Building Color Palette

Seaborn provides a function called **color_palette()**, which can be used to give colors to plots and adding more aesthetic value to it.

Usage

```
seaborn.color_palette(palette=None, n_colors=None, desat=None)
```

Parameter

The following table lists down the parameters for building color palette:

palatte	Name of palette
n_colors	Number of colors in the palette. If None, the default will depend on how palette is specified. By default the value of n_colors is 6 colors.
desat	Proportion to desaturate each color.

Return

Return refers to the list of RGB tuples. Following are the readily available Seaborn palettes:

- Deep
- Muted
- Bright
- Pastel

- Dark
- Colorblind

Besides these, one can also generate new palettes.

It is hard to decide which palette should be used for a given data set without knowing the characteristics of data. Being aware of it, we will classify the different ways for using **color_palette()** types:

- qualitative
- sequential
- diverging

We have another function **seaborn.palplot()** which deals with color palettes. This function plots the color palette as horizontal array. We will know more regarding **seaborn.palplot()** in the coming examples.

Qualitative Color Palettes

Qualitative or categorical palettes are best suitable to plot the categorical data.

Example

```
from matplotlib import pyplot as plt
import seaborn as sb
current_palette = sb.color_palette()
sb.palplot(current_palette)
plt.show()
```

Output



We haven't passed any parameters in **color_palette()**; by default, we are seeing 6 colors. You can see the desired number of colors by passing a value to the **n_colors** parameter. Here, the **palplot()** is used to plot the array of colors horizontally.

Sequential Color Palettes

Sequential plots are suitable to express the distribution of data ranging from relative lower values to higher values within a range.

Appending an additional character 's' to the color passed to the color parameter will plot the Sequential plot.

Example

```
from matplotlib import pyplot as plt
import seaborn as sb
current_palette = sb.color_palette()
sb.palplot(sb.color_palette("Greens"))
plt.show()
```

Output



Note: We need to append 's' to the parameter like 'Greens' in the above example.

Diverging Color Palette

Diverging palettes use two different colors. Each color represents variation in the value ranging from a common point in either direction.

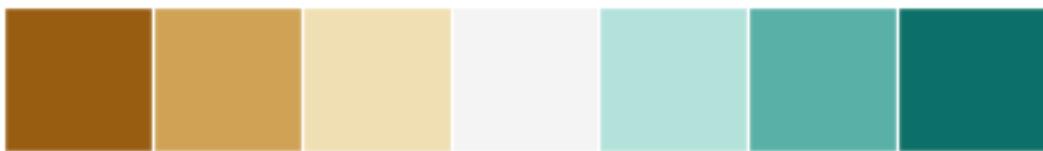
Assume plotting the data ranging from -1 to 1. The values from -1 to 0 takes one color and 0 to +1 takes another color.

By default, the values are centered from zero. You can control it with parameter **center** by passing a value.

Example

```
from matplotlib import pyplot as plt
import seaborn as sb
current_palette = sb.color_palette()
sb.palplot(sb.color_palette("BrBG", 7))
plt.show()
```

Output



Setting the Default Color Palette

The functions **color_palette()** has a companion called **set_palette()**. The relationship between them is similar to the pairs covered in the aesthetics chapter. The arguments are same for both **set_palette()** and **color_palette()**, but the default Matplotlib parameters are changed so that the palette is used for all plots.

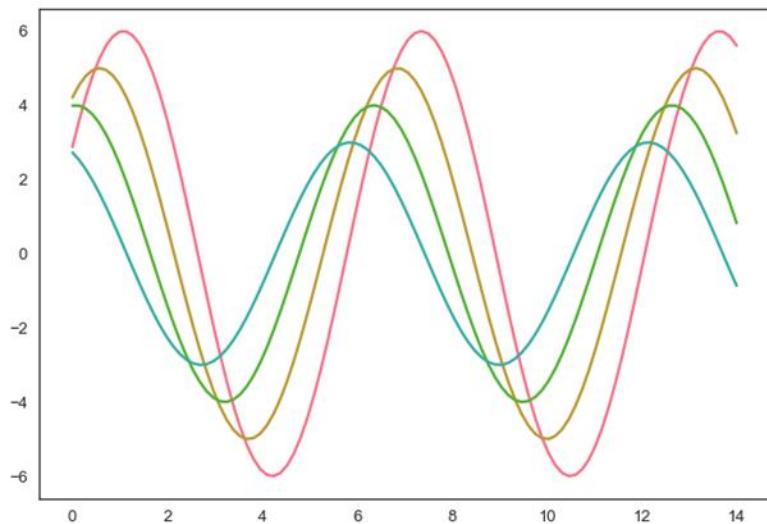
Example

```
import numpy as np
from matplotlib import pyplot as plt
def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 5):
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip)

import seaborn as sb
sb.set_style("white")
```

```
sb.set_palette("husl")
sinplot()
plt.show()
```

Output



Plotting Univariate Distribution

Distribution of data is the foremost thing that we need to understand while analysing the data. Here, we will see how seaborn helps us in understanding the univariate distribution of the data.

Function **distplot()** provides the most convenient way to take a quick look at univariate distribution. This function will plot a histogram that fits the kernel density estimation of the data.

Usage

```
seaborn.distplot()
```

Parameters

The following table lists down the parameters and their description:

Parameter	Description
data	Series, 1d array or a list
bins	Specification of hist bins
hist	bool
kde	bool

These are basic and important parameters to look into.

6. Seaborn – Histogram

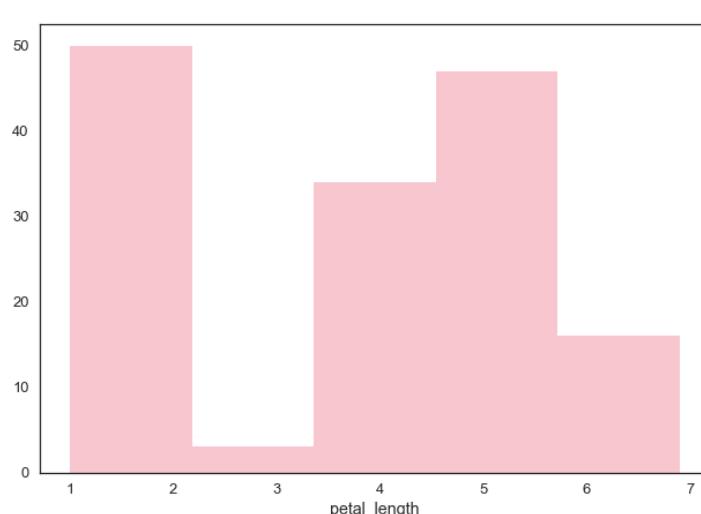
Histograms represent the data distribution by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin.

Seaborn comes with some datasets and we have used few datasets in our previous chapters. We have learnt how to load the dataset and how to lookup the list of available datasets.

Let us use iris dataset for our further analysis.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
sb.distplot(df['petal_length'], kde=False)  
  
plt.show()
```



Here, **kde** flag is set to False. As a result, the representation of the kernel estimation plot will be removed and only histogram is plotted.

Output

7. Seaborn – Kernel Density Estimates

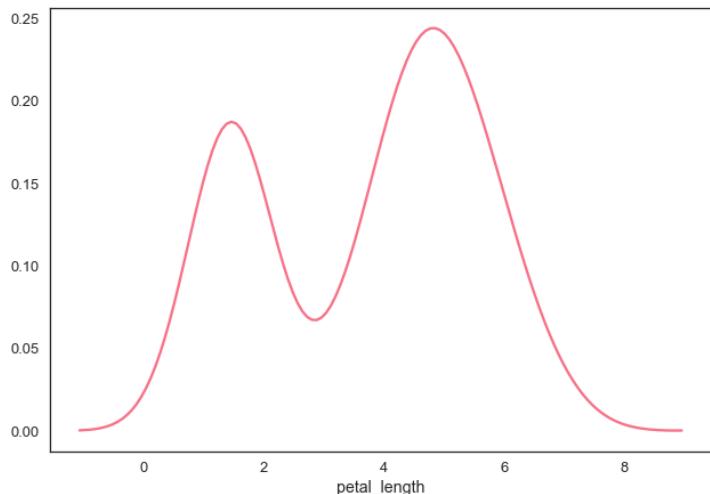
Kernel Density Estimation (KDE) is a way to estimate the probability density function of a continuous random variable. It is used for non-parametric analysis.

Setting the **hist** flag to False in **distplot** will yield the kernel density estimation plot.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
sb.distplot(df['petal_length'], hist=False)  
  
plt.show()
```

Output



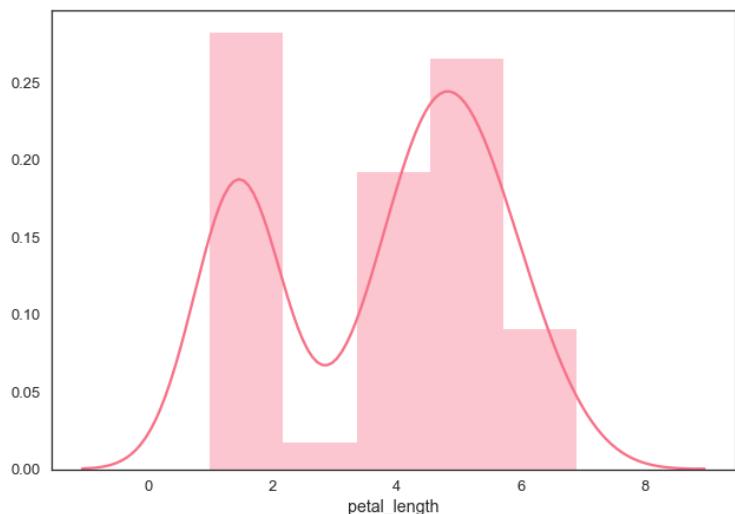
Fitting Parametric Distribution

distplot() is used to visualize the parametric distribution of a dataset.

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
sb.distplot(df['petal_length'])
plt.show()
```

Output



Plotting Bivariate Distribution

Bivariate Distribution is used to determine the relation between two variables. This mainly deals with relationship between two variables and how one variable is behaving with respect to the other.

The best way to analyze Bivariate Distribution in seaborn is by using the **jointplot()** function.

Jointplot creates a multi-panel figure that projects the bivariate relationship between two variables and also the univariate distribution of each variable on separate axes.

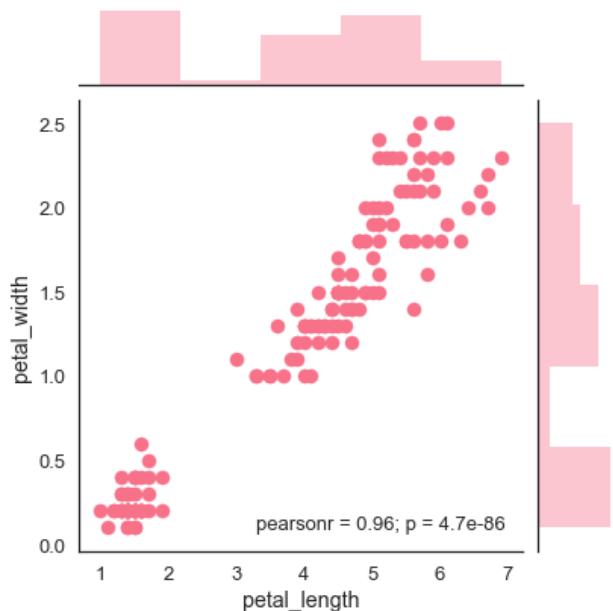
Scatter Plot

Scatter plot is the most convenient way to visualize the distribution where each observation is represented in two-dimensional plot via x and y axis.

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
sb.jointplot(x='petal_length',y='petal_width',data=df)
plt.show()
```

Output



The above figure shows the relationship between the **petal_length** and **petal_width** in the Iris data. A trend in the plot says that positive correlation exists between the variables under study.

Hexbin Plot

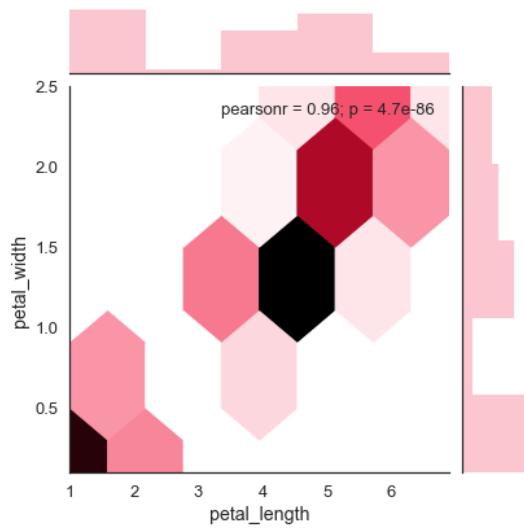
Hexagonal binning is used in bivariate data analysis when the data is sparse in density i.e., when the data is very scattered and difficult to analyze through scatterplots.

An addition parameter called 'kind' and value 'hex' plots the hexbin plot.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
sb.jointplot(x='petal_length',y='petal_width',data=df,kind='hex')  
  
plt.show()
```

Output



Kernel Density Estimation

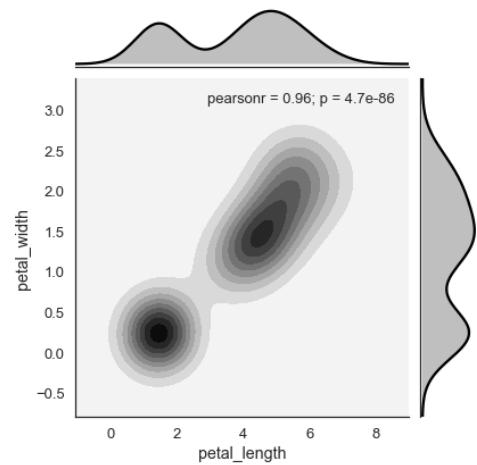
Kernel density estimation is a non-parametric way to estimate the distribution of a variable. In seaborn, we can plot a kde using **jointplot()**.

Pass value 'kde' to the parameter kind to plot kernel plot.

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
sb.jointplot(x='petal_length',y='petal_width',data=df,kind='hex')
plt.show()
```

Output



8. Seaborn – Visualizing Pairwise Relationship

Datasets under real-time study contain many variables. In such cases, the relation between each and every variable should be analyzed. Plotting Bivariate Distribution for (n,2) combinations will be a very complex and time taking process.

To plot multiple pairwise bivariate distributions in a dataset, you can use the **pairplot()** function. This shows the relationship for (n,2) combination of variable in a DataFrame as a matrix of plots and the diagonal plots are the univariate plots.

Axes

In this section, we will learn what are Axes, their usage, parameters, and so on.

Usage

```
seaborn.pairplot(data,...)
```

Parameters

Following table lists down the parameters for Axes:

Parameter	Description
data	Dataframe
hue	Variable in data to map plot aspects to different colors.
palette	Set of colors for mapping the hue variable
kind	Kind of plot for the non-identity relationships. {'scatter', 'reg'}
diag_kind	Kind of plot for the diagonal subplots. {'hist', 'kde'}

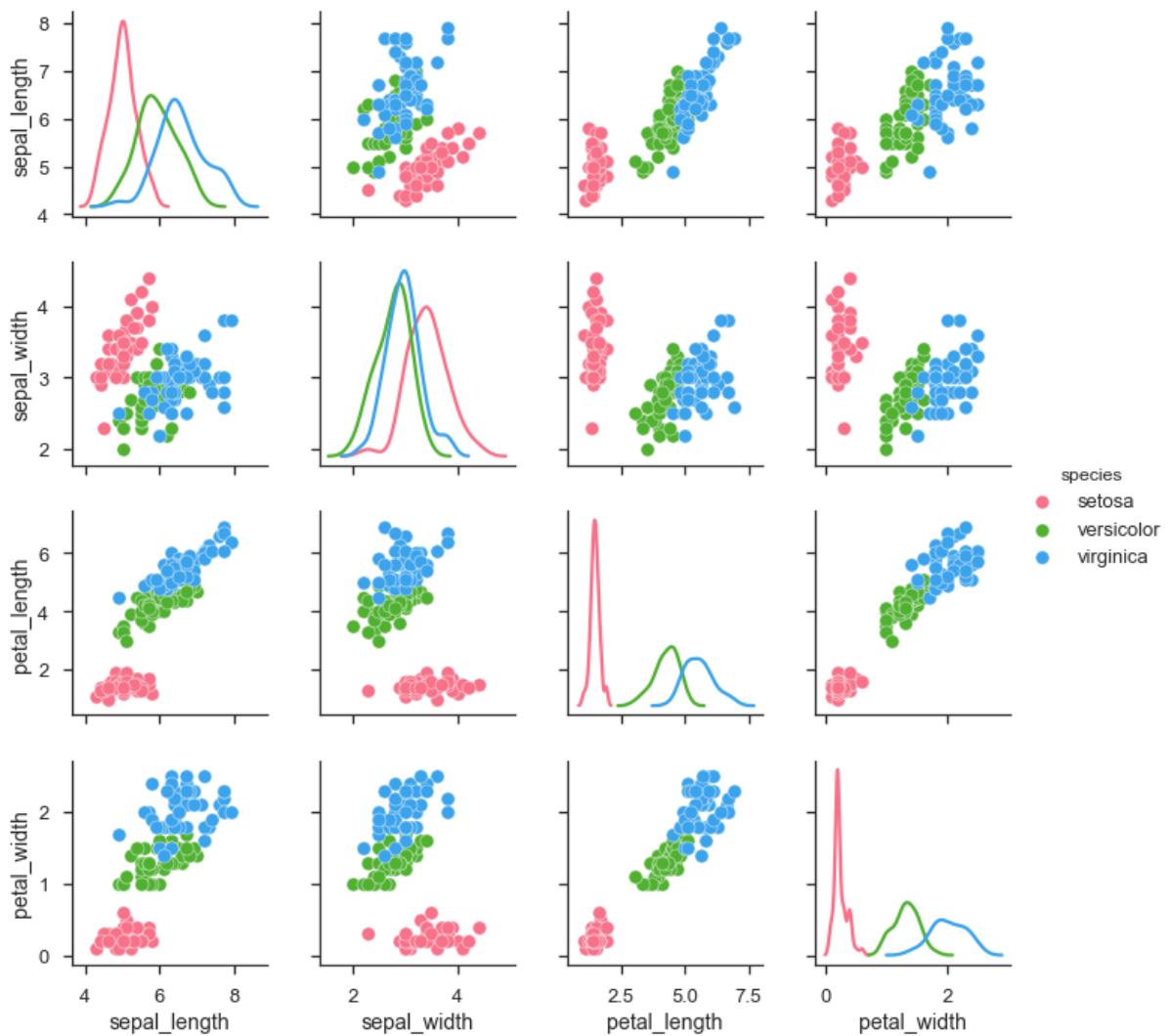
Except data, all other parameters are optional. There are few other parameters which **pairplot** can accept. The above mentioned are often used params.

Example

```
import pandas as pd
```

```
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
sb.set_style("ticks")  
  
sb.pairplot(df,hue='species',diag_kind="kde",kind="scatter",palette="husl")  
  
plt.show()
```

Output



We can observe the variations in each plot. The plots are in matrix format where the row name represents x axis and column name represents the y axis.

The diagonal plots are kernel density plots where the other plots are scatter plots as mentioned.

9. Seaborn – Plotting Categorical Data

In our previous chapters we learnt about scatter plots, hexbin plots and kde plots which are used to analyze the continuous variables under study. These plots are not suitable when the variable under study is categorical.

When one or both the variables under study are categorical, we use plots like `stripplot()`, `swarmplot()`, etc,. Seaborn provides interface to do so.

Categorical Scatter Plots

In this section, we will learn about categorical scatter plots.

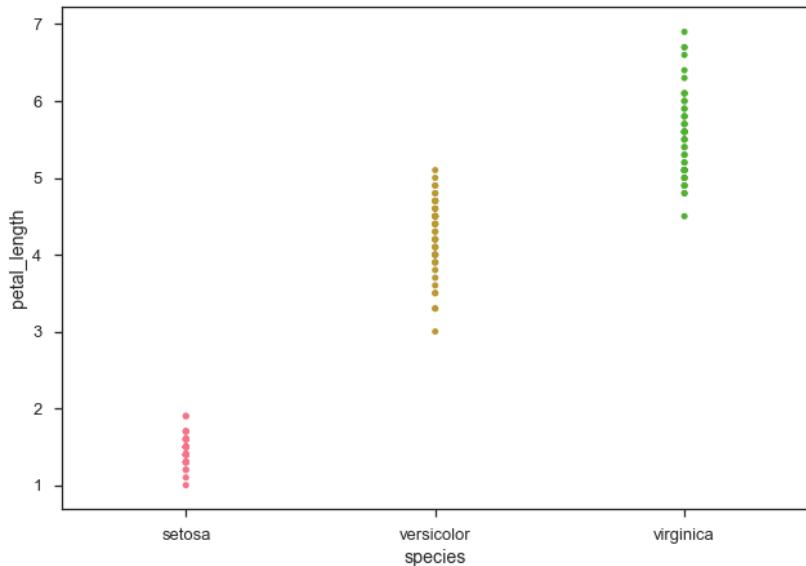
stripplot()

`stripplot()` is used when one of the variable under study is categorical. It represents the data in sorted order along any one of the axis.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
sb.stripplot(x="species", y="petal_length", data=df)  
plt.show()
```

Output



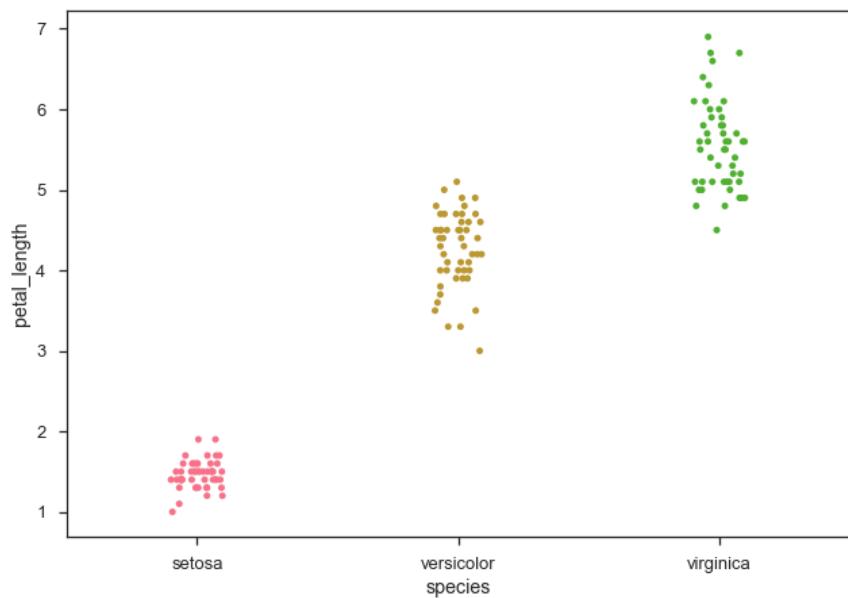
In the above plot, we can clearly see the difference of **petal_length** in each species. But, the major problem with the above scatter plot is that the points on the scatter plot are overlapped. We use the 'Jitter' parameter to handle this kind of scenario.

Jitter adds some random noise to the data. This parameter will adjust the positions along the categorical axis.

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
sb.stripplot(x="species", y="petal_length", data=df, jitter=True)
plt.show()
```

Output



Now, the distribution of points can be seen easily.

Swarmplot()

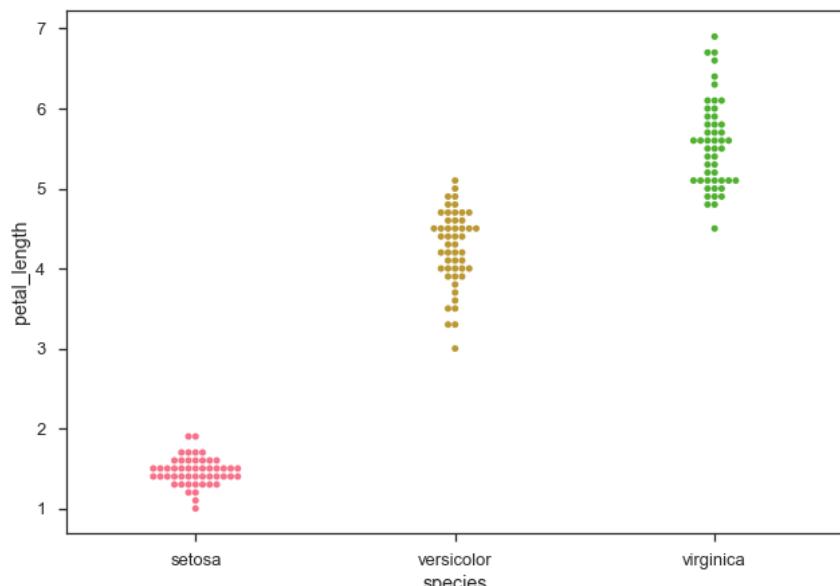
Another option which can be used as an alternate to 'Jitter' is function **swarmplot()**.

This function positions each point of scatter plot on the categorical axis and thereby avoids overlapping points:

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
sb.swarmplot(x="species", y="petal_length", data=df)  
  
plt.show()
```

Output



10. Seaborn – Distribution of Observations

In categorical scatter plots which we dealt in the previous chapter, the approach becomes limited in the information it can provide about the distribution of values within each category. Now, going further, let us see what can facilitate us with performing comparison within categories.

Box Plots

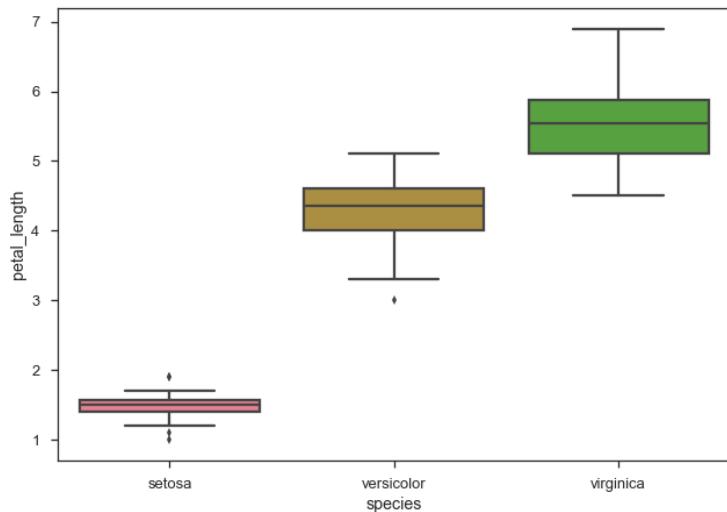
Boxplot is a convenient way to visualize the distribution of data through their quartiles.

Box plots usually have vertical lines extending from the boxes which are termed as whiskers. These whiskers indicate variability outside the upper and lower quartiles, hence Box Plots are also termed as **box-and-whisker plot** and **box-and-whisker diagram**. Any Outliers in the data are plotted as individual points.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
sb.swarmplot(x="species", y="petal_length", data=df)  
  
plt.show()
```

Output



The dots on the plot indicates the outlier.

Violin Plots

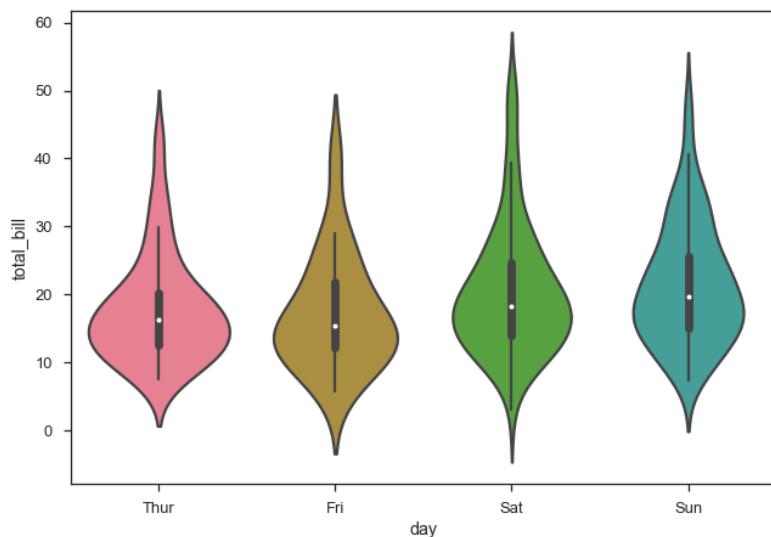
Violin Plots are a combination of the box plot with the kernel density estimates. So, these plots are easier to analyze and understand the distribution of the data.

Let us use tips dataset called to learn more into violin plots. This dataset contains the information related to the tips given by the customers in a restaurant.

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('tips')
sb.violinplot(x="day", y="total_bill", data=df)
plt.show()
```

Output



The quartile and whisker values from the boxplot are shown inside the violin. As the violin plot uses KDE, the wider portion of violin indicates the higher density and narrow region represents relatively lower density. The Inter-Quartile range in boxplot and higher density portion in kde fall in the same region of each category of violin plot.

The above plot shows the distribution of total_bill on four days of the week. But, in addition to that, if we want to see how the distribution behaves with respect to sex, lets explore it in below example.

Example

```
import pandas as pd

import seaborn as sb

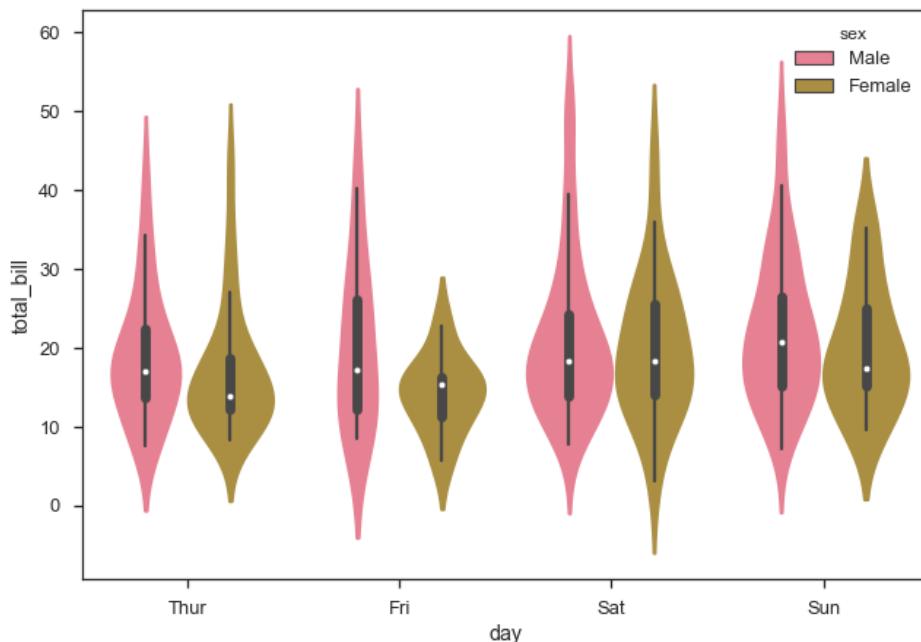
from matplotlib import pyplot as plt

df = sb.load_dataset('tips')

sb.violinplot(x="day", y="total_bill", hue='sex', data=df)

plt.show()
```

Output



Now we can clearly see the spending behavior between male and female. We can easily say that, men make more bill than women by looking at the plot.

And, if the hue variable has only two classes, we can beautify the plot by splitting each violin into two instead of two violins on a given day. Either parts of the violin refer to each class in the hue variable.

Example

```
import pandas as pd
```

```
import seaborn as sb

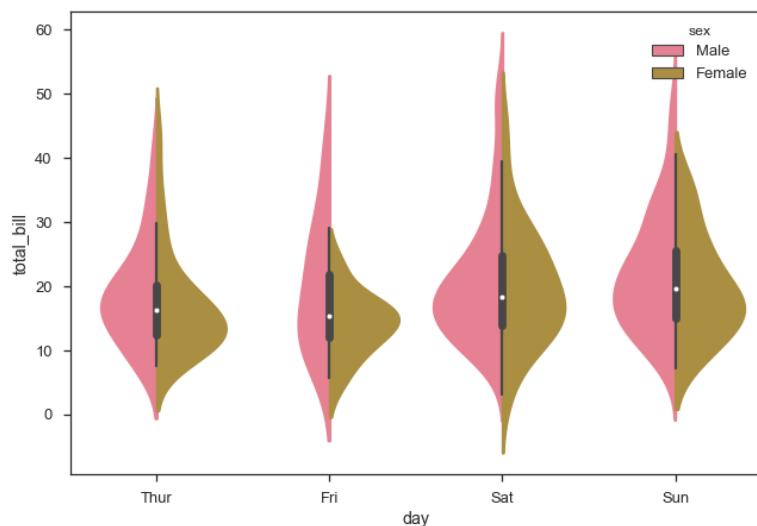
from matplotlib import pyplot as plt

df = sb.load_dataset('tips')

sb.violinplot(x="day", y="total_bill", hue='sex', data=df)

plt.show()
```

Output



11. Seaborn – Statistical Estimation

In most of the situations, we deal with estimations of the whole distribution of the data. But when it comes to central tendency estimation, we need a specific way to summarize the distribution. Mean and median are the very often used techniques to estimate the central tendency of the distribution.

In all the plots that we learnt in the above section, we made the visualization of the whole distribution. Now, let us discuss regarding the plots with which we can estimate the central tendency of the distribution.

Bar Plot

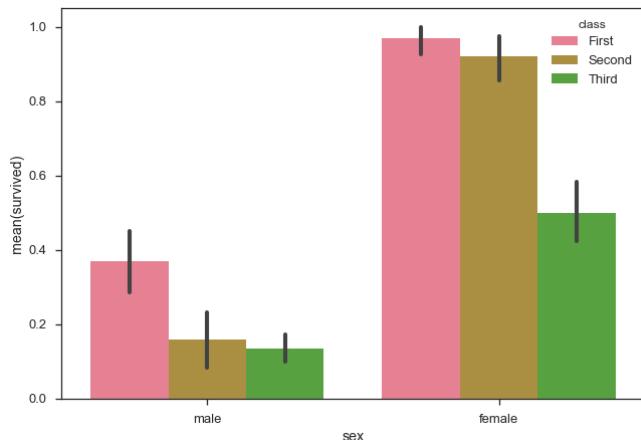
The **barplot()** shows the relation between a categorical variable and a continuous variable. The data is represented in rectangular bars where the length the bar represents the proportion of the data in that category.

Bar plot represents the estimate of central tendency. Let us use the 'titanic' dataset to learn barplots.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('titanic')  
  
sb.barplot(x="sex", y="survived", hue="class", data=df)  
  
plt.show()
```

Output



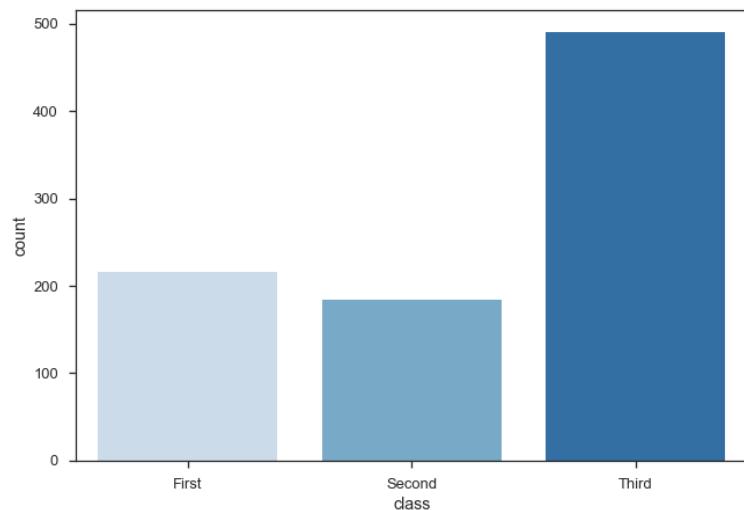
In the above example, we can see that the average number of survivals of male and female in each class. From the plot we can understand that more number of females survived than males. In both males and females more number of survivals are from first class.

A special case in barplot is to show the no of observations in each category rather than computing a statistic for a second variable. For this, we use **countplot()**.

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('titanic')
sb.countplot(x=" class ", data=df, palette="Blues");
plt.show()
```

Output



Plot says that, the number of passengers in the third class are higher than first and second class.

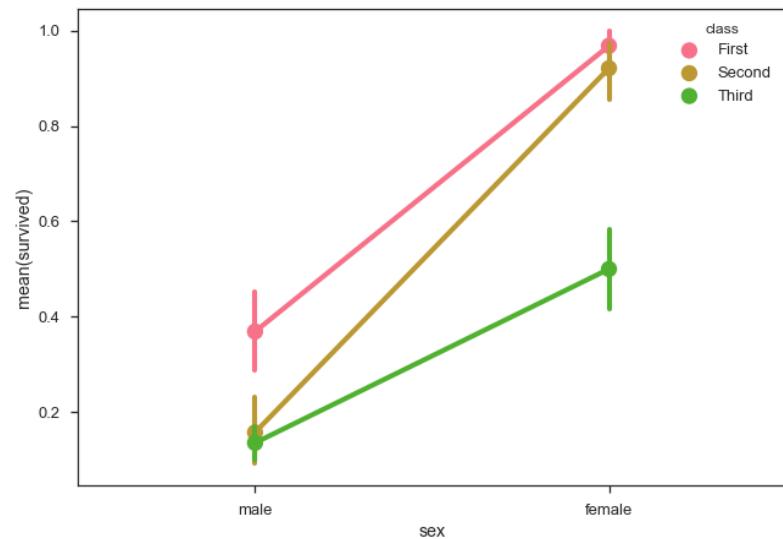
Point Plots

Point plots serve same as bar plots but in a different style. Rather than the full bar, the value of the estimate is represented by the point at a certain height on the other axis.

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('titanic')
sb.pointplot(x="sex", y="survived", hue="class", data=df)
plt.show()
```

Output



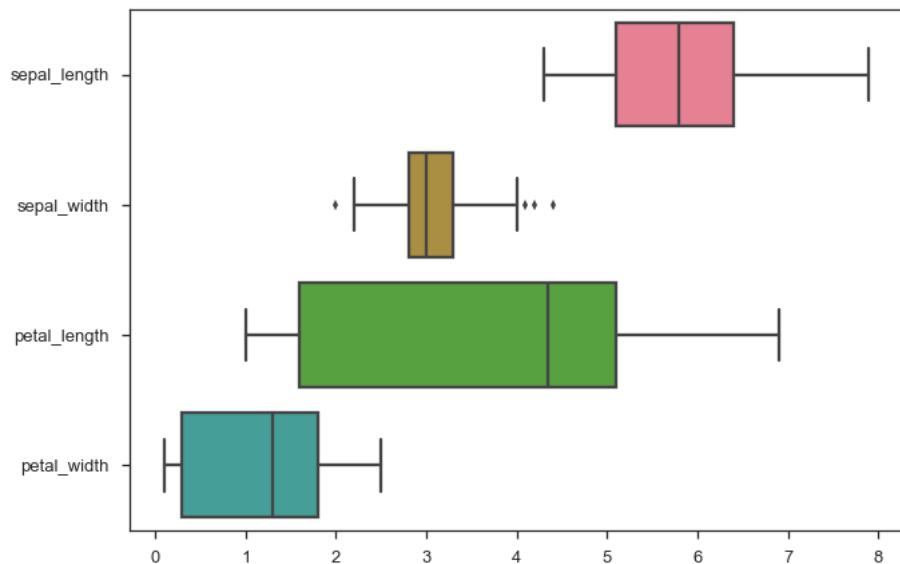
12. Seaborn – Plotting Wide Form Data

It is always preferable to use 'long-from' or 'tidy' datasets. But at times when we are left with no option rather than to use a 'wide-form' dataset, same functions can also be applied to "wide-form" data in a variety of formats, including Pandas Data Frames or two-dimensional NumPy arrays. These objects should be passed directly to the data parameter the x and y variables must be specified as strings

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
sb.boxplot(data=df, orient="h")  
  
plt.show()
```

Output

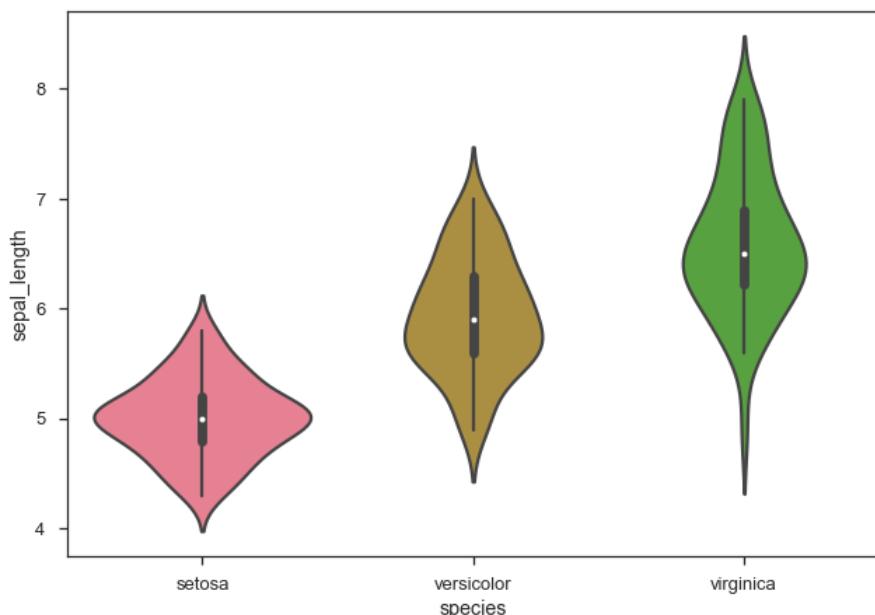


Additionally, these functions accept vectors of Pandas or NumPy objects rather than variables in a DataFrame.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
sb.boxplot(data=df, orient="h")  
  
plt.show()
```

Output



The major advantage of using Seaborn for many developers in Python world is because it can take pandas DataFrame object as parameter.

13. Seaborn – Multi Panel Categorical Plots

Categorical data can be visualized using two plots, you can either use the functions **pointplot()**, or the higher-level function **factorplot()**.

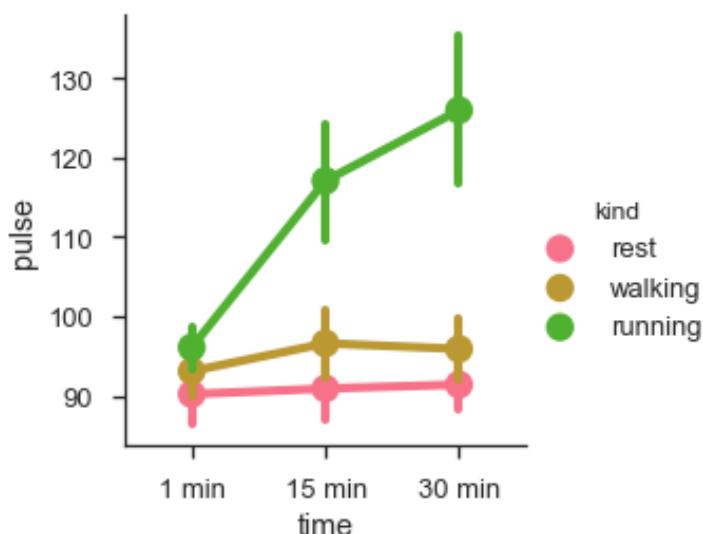
Factorplot

Factorplot draws a categorical plot on a FacetGrid. Using 'kind' parameter we can choose the plot like boxplot, violinplot, barplot and stripplot. FacetGrid uses pointplot by default.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('exercise')  
  
sb.factorplot(x="time", y="pulse", hue="kind", data=df);  
  
plt.show()
```

Output

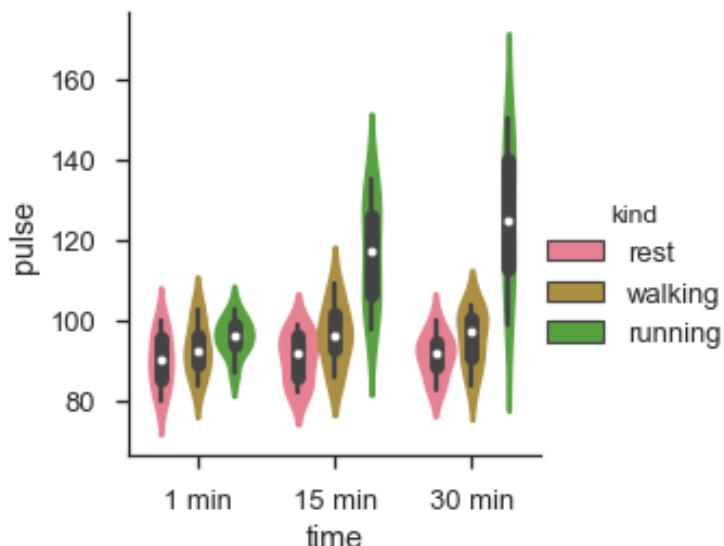


We can use different plot to visualize the same data using the **kind** parameter.

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('exercise')
sb.factorplot(x="time", y="pulse", hue="kind", kind='violin', data=df);
plt.show()
```

Output



In factorplot, the data is plotted on a facet grid.

What is Facet Grid?

Facet grid forms a matrix of panels defined by row and column by dividing the variables. Due of panels, a single plot looks like multiple plots. It is very helpful to analyze all combinations in two discrete variables.

Let us visualize the above the definition with an example.

Example

```
import pandas as pd

import seaborn as sb

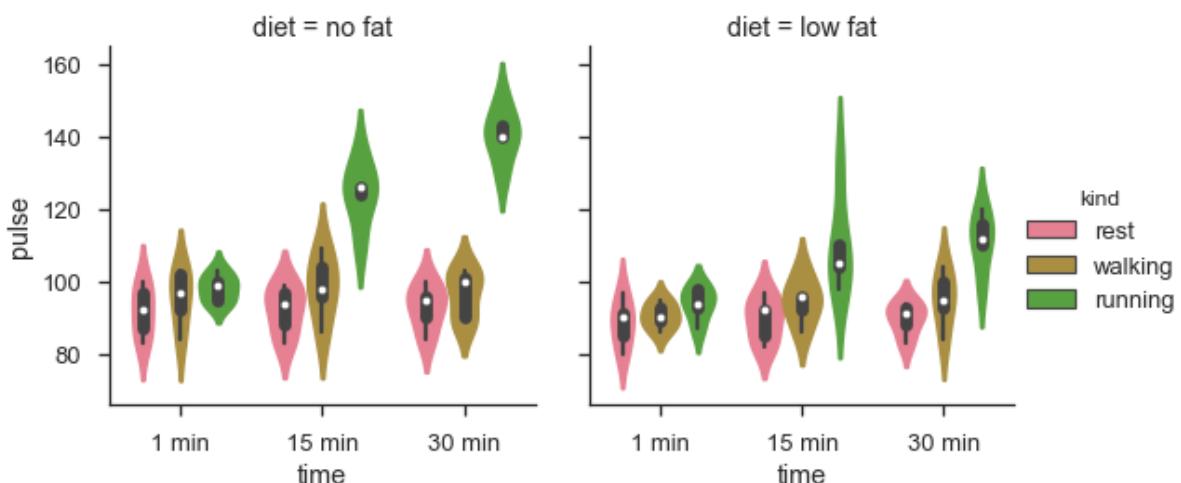
from matplotlib import pyplot as plt

df = sb.load_dataset('exercise')

sb.factorplot(x="time", y="pulse", hue="kind", kind='violin', col="diet",
data=df);

plt.show()
```

Output



The advantage of using Facet is, we can input another variable into the plot. The above plot is divided into two plots based on a third variable called 'diet' using the 'col' parameter.

We can make many column facets and align them with the rows of the grid:

Example

```
import pandas as pd

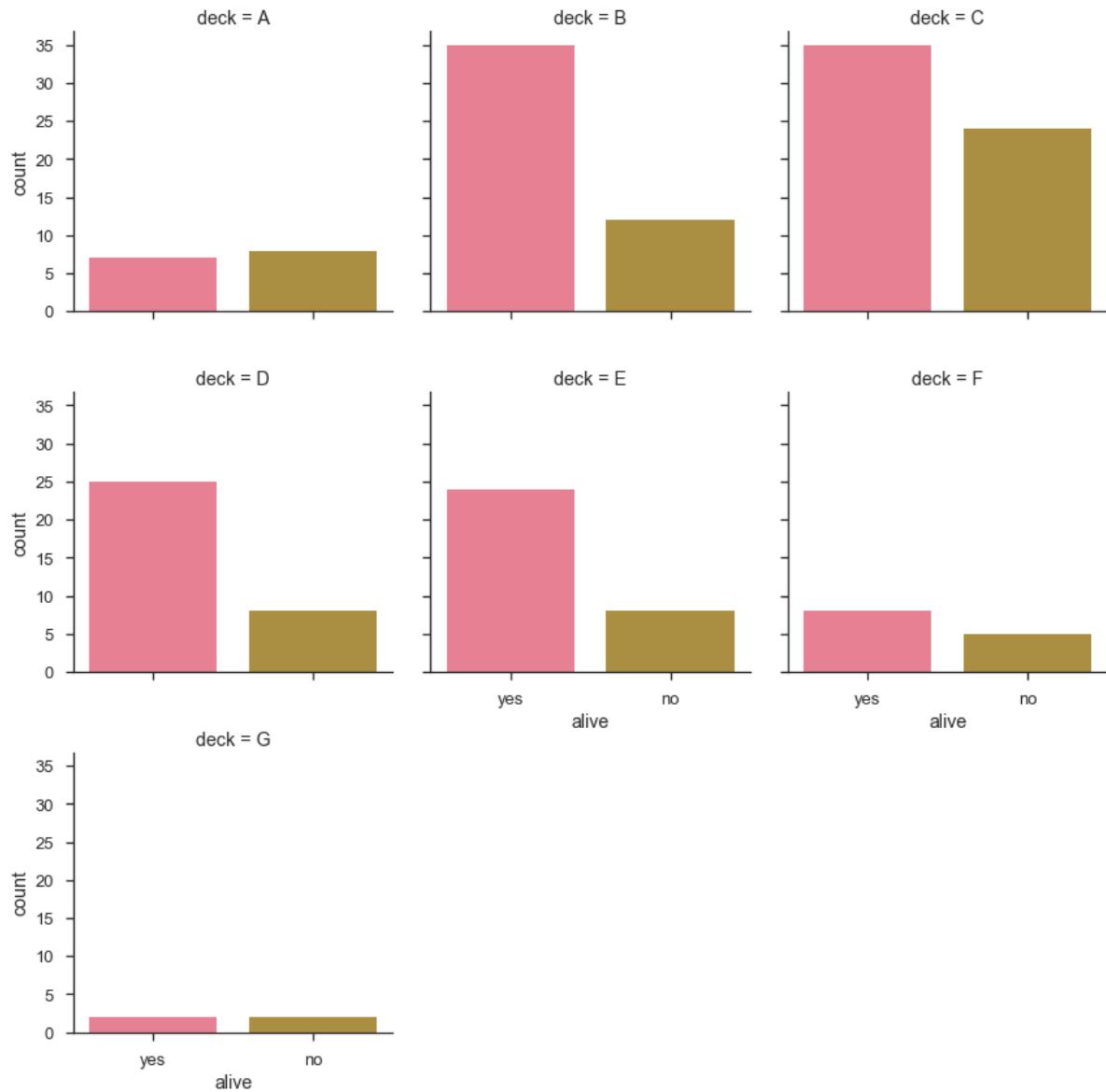
import seaborn as sb

from matplotlib import pyplot as plt

df = sb.load_dataset('titanic')
```

```
sb.factorplot("alive", col="deck",
col_wrap=3,data=df[df.deck.notnull()],kind="count")
plt.show()
```

Output



14. Seaborn – Linear Relationships

Most of the times, we use datasets that contain multiple quantitative variables, and the goal of an analysis is often to relate those variables to each other. This can be done through the regression lines.

While building the regression models, we often check for **multicollinearity**, where we had to see the correlation between all the combinations of continuous variables and will take necessary action to remove multicollinearity if exists. In such cases, the following techniques helps.

Functions to Draw Linear Regression Models

There are two main functions in Seaborn to visualize a linear relationship determined through regression. These functions are **regplot()** and **lmplot()**.

regplot vs lmplot

regplot	lmplot
accepts the x and y variables in a variety of formats including simple numpy arrays, pandas Series objects, or as references to variables in a pandas DataFrame	has data as a required parameter and the x and y variables must be specified as strings. This data format is called "long-form" data

Let us now draw the plots.

Example

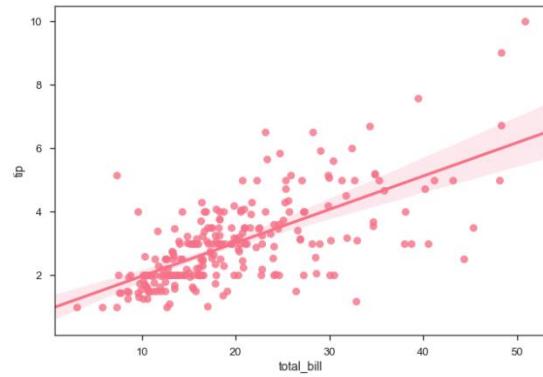
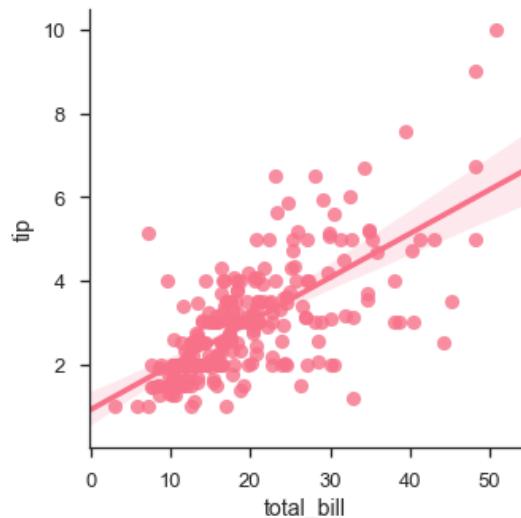
Plotting the regplot and then lm plot with the same data in this example.

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('tips')  
  
sb.regplot(x="total_bill", y="tip", data=df)  
  
sb.lmplot(x="total_bill", y="tip", data=df)
```

```
plt.show()
```

Output

You can see the difference in the size between two plots.

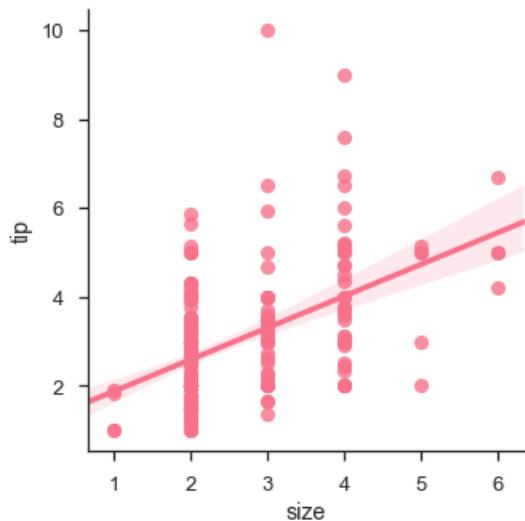


We can also fit a linear regression when one of the variables takes discrete values

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('tips')
sb.lmplot(x="size", y="tip", data=df)
plt.show()
```

Output



Fitting Different Kinds of Models

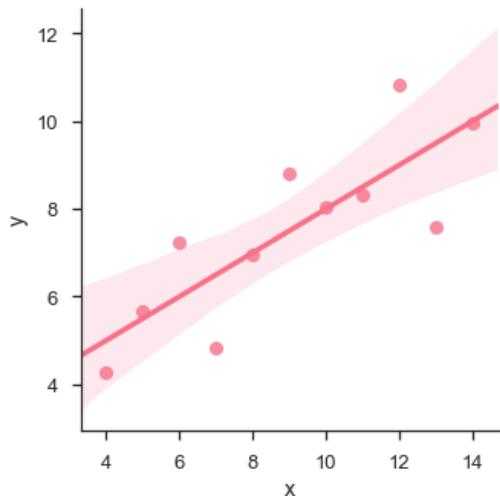
The simple linear regression model used above is very simple to fit, but in most of the cases, the data is non-linear and the above methods cannot generalize the regression line.

Let us use Anscombe's dataset with the regression plots:

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('anscombe')
sb.lmplot(x="x", y="y", data=df.query("dataset == 'I'"))
plt.show()
```

Output



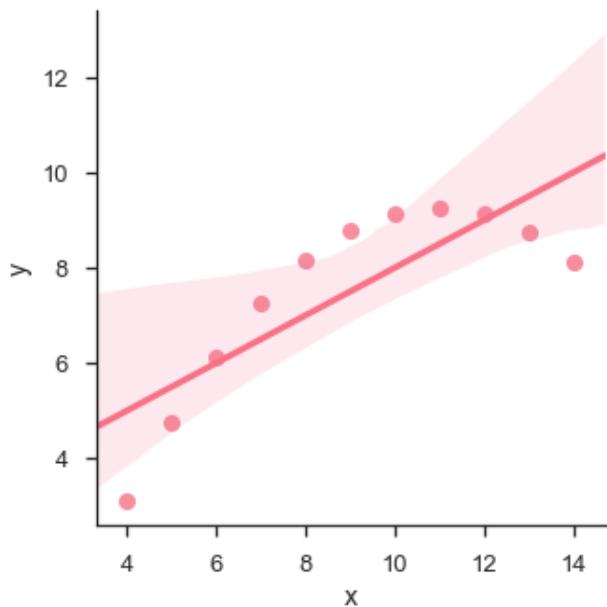
In this case, the data is good fit for linear regression model with less variance.

Let us see another example where the data takes high deviation which shows the line of best fit is not good.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('anscombe')  
  
sb.lmplot(x="x", y="y", data=df.query("dataset == 'II'"))  
  
plt.show()
```

Output

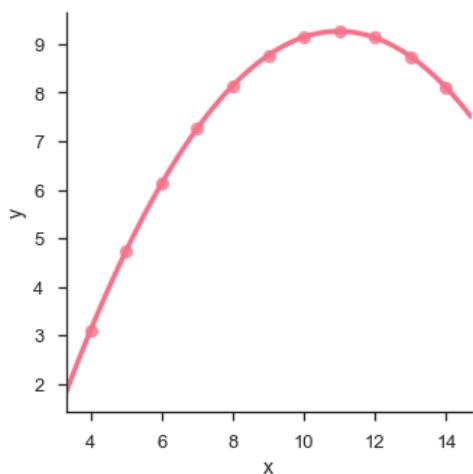


The plot shows the high deviation of data points from the regression line. Such non-linear, higher order can be visualized using the **lmplot()** and **regplot()**. These can fit a polynomial regression model to explore simple kinds of nonlinear trends in the dataset:

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('anscombe')
sb.lmplot(x="x", y="y", data=df.query("dataset == 'II'"),order=2)
plt.show()
```

Output



15. Seaborn – Facet Grid

A useful approach to explore medium-dimensional data, is by drawing multiple instances of the same plot on different subsets of your dataset.

This technique is commonly called as “lattice”, or “trellis” plotting, and it is related to the idea of “small multiples”.

To use these features, your data has to be in a Pandas DataFrame.

Plotting Small Multiples of Data Subsets

In the previous chapter, we have seen the FacetGrid example where FacetGrid class helps in visualizing distribution of one variable as well as the relationship between n multiple variables separately within subsets of your dataset using multiple panels.

A FacetGrid can be drawn with up to three dimensions: row, col, and hue. The first two have obvious correspondence with the resulting array of axes; think of the hue variable as a third dimension along a depth axis, where different levels are plotted with different colors.

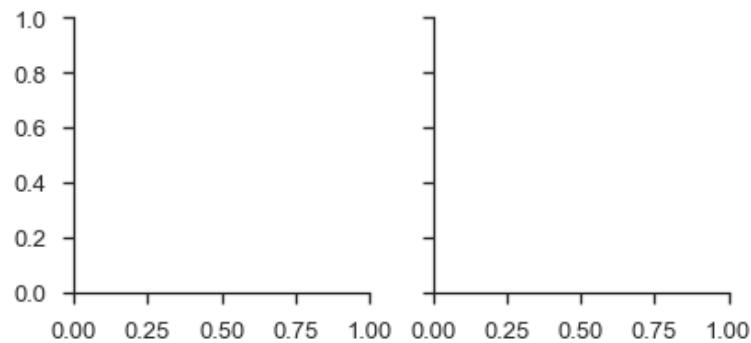
FacetGrid object takes a dataframe as input and the names of the variables that will form the row, column, or hue dimensions of the grid.

The variables should be categorical and the data at each level of the variable will be used for a facet along that axis.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('tips')  
  
g = sb.FacetGrid(df, col="time")  
  
plt.show()
```

Output



In the above example, we have just initialized the **facetgrid** object which doesn't draw anything on them.

The main approach for visualizing data on this grid is with the **FacetGrid.map()** method. Let us look at the distribution of tips in each of these subsets, using a histogram.

Example

```
import pandas as pd

import seaborn as sb

from matplotlib import pyplot as plt

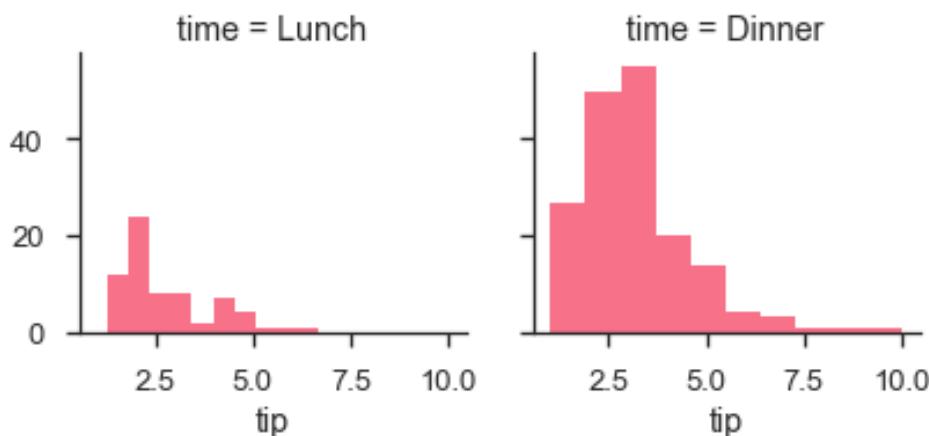
df = sb.load_dataset('tips')

g = sb.FacetGrid(df, col="time")

g.map(plt.hist, "tip")

plt.show()
```

Output



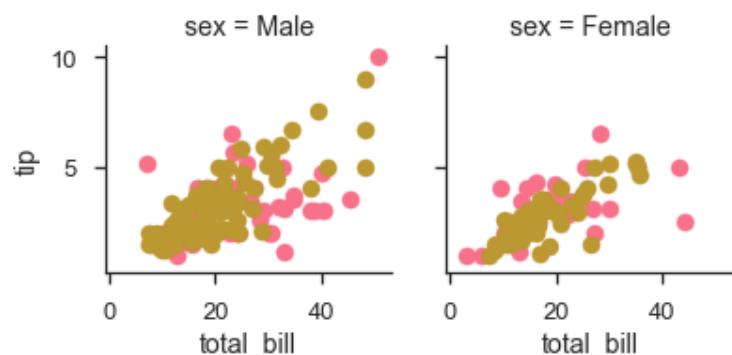
The number of plots is more than one because of the parameter `col`. We discussed about `col` parameter in our previous chapters.

To make a relational plot, pass the multiple variable names.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('tips')  
  
g = sb.FacetGrid(df, col="sex", hue="smoker")  
  
g.map(plt.scatter, "total_bill", "tip")  
  
plt.show()
```

Output



16. Seaborn – Pair Grid

PairGrid allows us to draw a grid of subplots using the same plot type to visualize data.

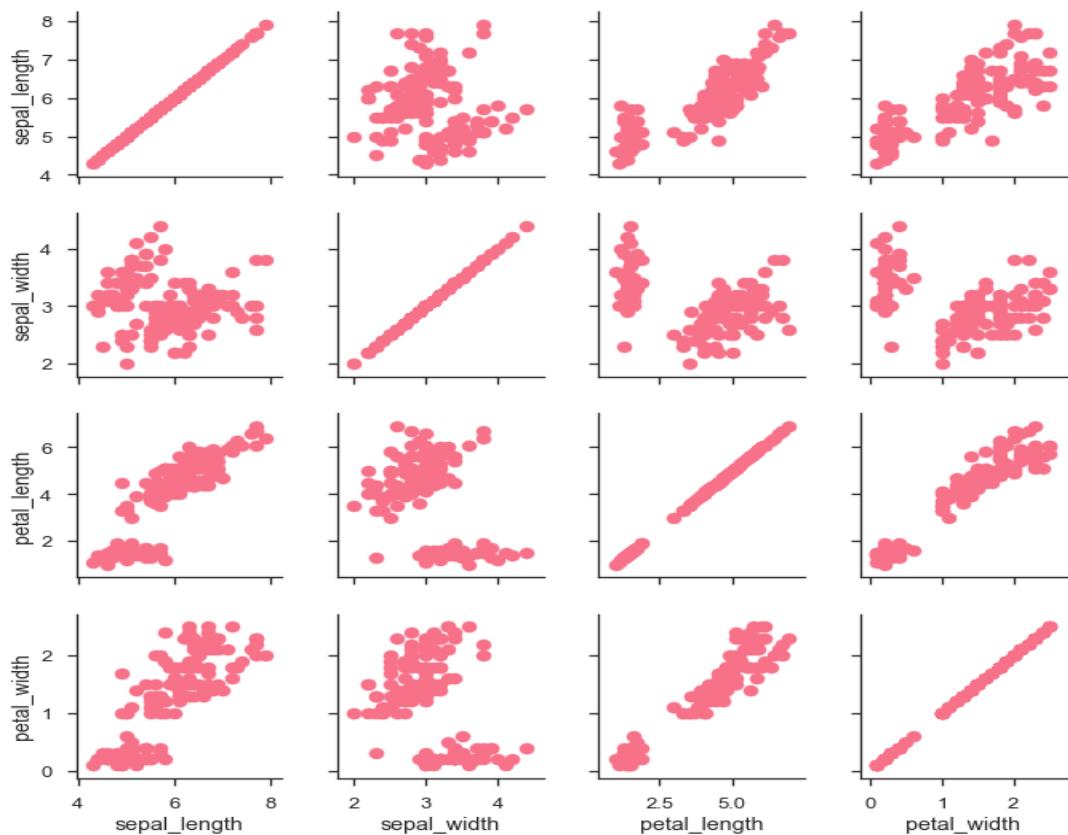
Unlike FacetGrid, it uses different pair of variable for each subplot. It forms a matrix of sub-plots. It is also sometimes called as “scatterplot matrix”.

The usage of pairgrid is similar to facetgrid. First initialise the grid and then pass the plotting function.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
g = sb.PairGrid(df)  
  
g.map(plt.scatter);  
  
plt.show()
```

Output

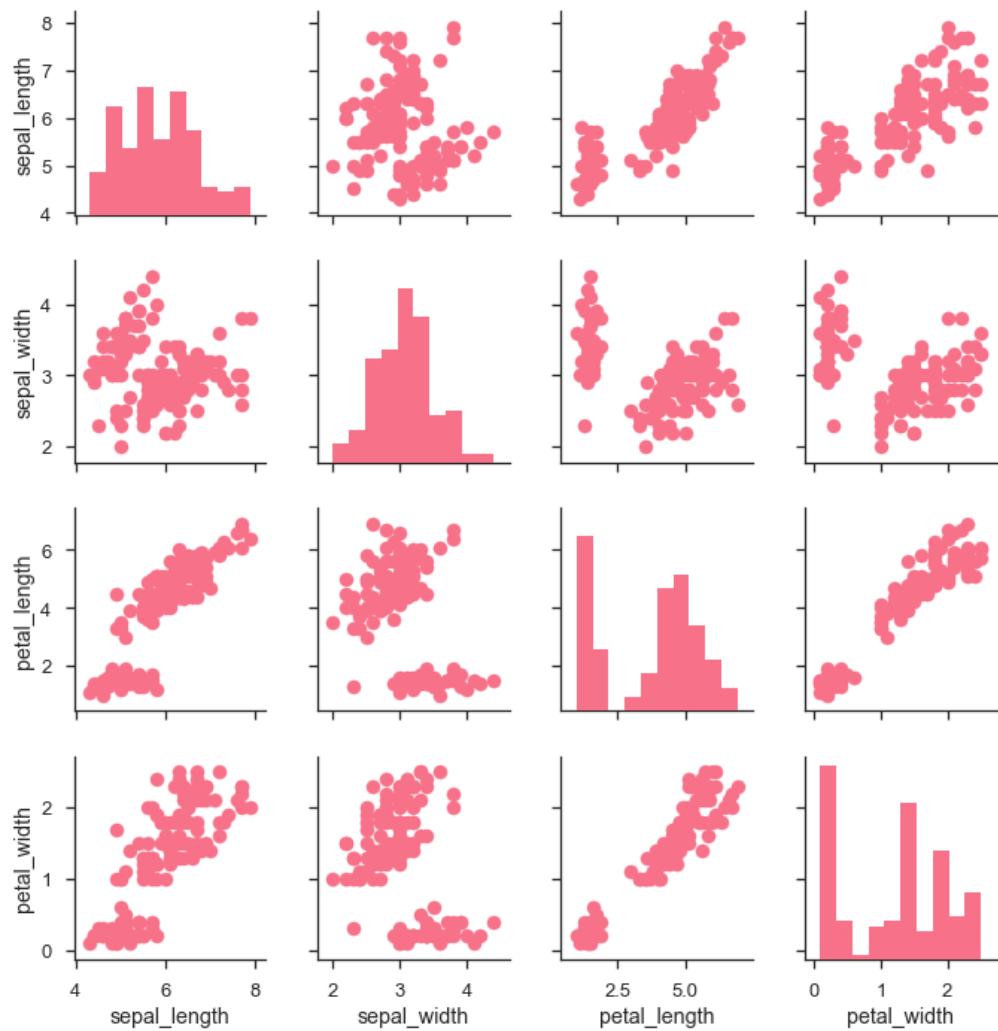


It is also possible to plot a different function on the diagonal to show the univariate distribution of the variable in each column.

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
g = sb.PairGrid(df)
g.map_diag(plt.hist)
g.map_offdiag(plt.scatter);
plt.show()
```

Output

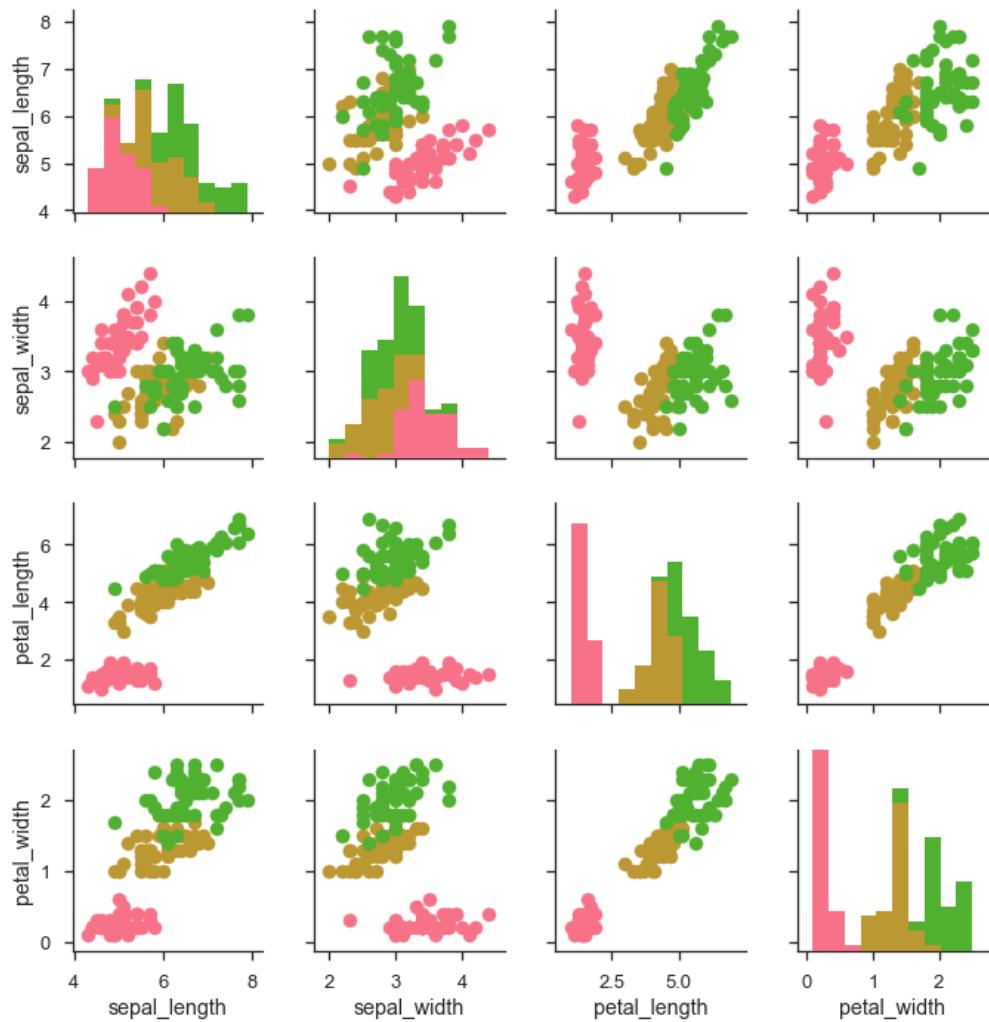


We can customize the color of these plots using another categorical variable. For example, the iris dataset has four measurements for each of three different species of iris flowers so you can see how they differ.

Example

```
import pandas as pd  
  
import seaborn as sb  
  
from matplotlib import pyplot as plt  
  
df = sb.load_dataset('iris')  
  
g = sb.PairGrid(df)  
  
g.map_diag(plt.hist)  
  
g.map_offdiag(plt.scatter);  
  
plt.show()
```

Output



We can use a different function in the upper and lower triangles to see different aspects of the relationship.

Example

```
import pandas as pd
import seaborn as sb
from matplotlib import pyplot as plt
df = sb.load_dataset('iris')
```

```
g = sb.PairGrid(df)
g.map_upper(plt.scatter)
g.map_lower(sb.kdeplot, cmap="Blues_d")
g.map_diag(sb.kdeplot, lw=3, legend=False);
plt.show()
```

Output

