In [1]:
```
# setup notebook
# notebook formatting
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))

# pretty print all cell's output and not just the last one
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# enable split cells in notebook
# if not installed:  pip install jupyter_contrib_nbextensions; then repeat this cmd
!jupyter nbextension enable splitcell/splitcell

# fix RISE scollbar missing
# if RISE scroll not working fix path to match your jupyter nbconfig, uncomment below and run
# from traitlets.config.manager import BaseJSONConfigManager
# path = "~\user\.jupyter\nbconfig"
# cm = BaseJSONConfigManager(config_dir=path)

# cm.update("livereveal", {
#               "scroll": True,
# });

# imports
import os
import random

# math
import pandas as pd
import numpy as np
np.warnings.filterwarnings('ignore')

from sklearn.linear_model import LinearRegression

# visualization - imports and setting
from matplotlib import pyplot as plt
from matplotlib.pyplot import figure
%matplotlib inline
```

```
Enabling notebook extension splitcell/splitcell...
      - Validating: ok
```

In [2]:
```
%%html
<style>
style="vertical-align: text-top";
table {float:left}
</style>
```

Effective Visualizations

About Me...

- Chris Brousseau
- @surfaceowl
- chris@surfaceowl.com

What I do...

- Founder: Surface Owl
- Data Scientist
- Python Development
- PyBay Diversity & Inclusion Chair

Agenda - Visualizations & Python

1- Understand why effective visualizations are important

2- Think about good approachs to visuals

3- Learn about the python visualization universe

4- Foundation - matplotlib

5- Future - bokeh

1- Understand why effective visualizations are important

Why Visualizations?

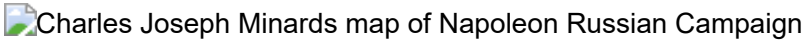
- **to accomplish a goal**
  - share information
  - create understanding
  - convice someone to take action

- Efficient compression of data

- Most sighted people naturally have powerful visual perception abilities

Why visualization is important: *Create Understanding*

Charles Minard's Map of the March to Moscow

 Charles Joseph Minards map of Napoleon Russian Campaign

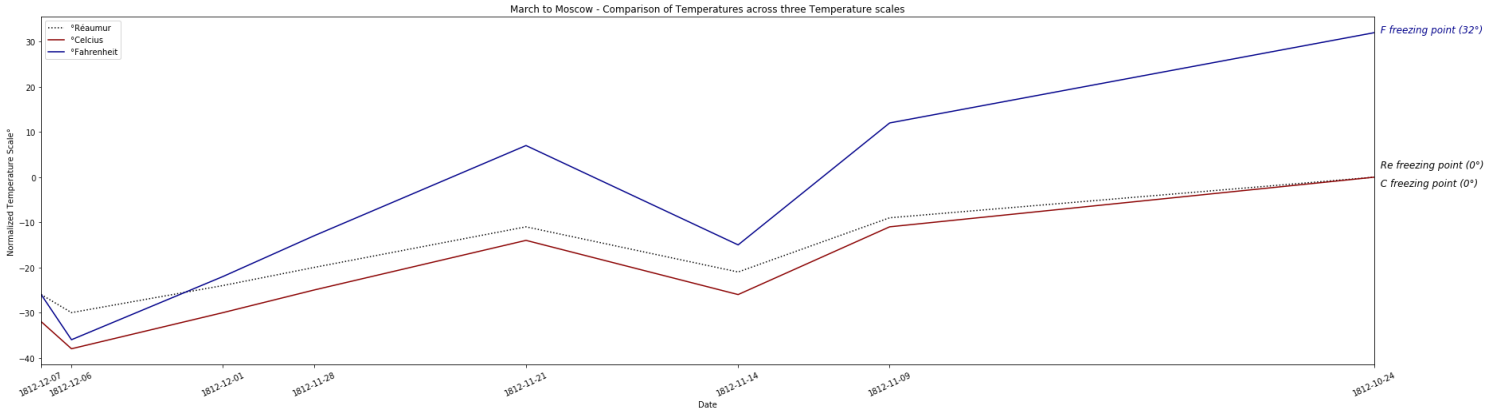
In [3]:
```
# Minard's chart uses a different temperature scaled called the Réaumur scale
# water freeze @ 0°, boils at 80°
# https://en.wikipedia.org/wiki/R%C3%A9aumur_scale
# what are these temperatures in scales we are more familiar with (°F, °C)?
import datetime as dt
import matplotlib.dates as mdates
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()

# get data for chart, format dates for matplotlib;
# data source:  https://www.cs.uic.edu/~wilkinson/TheGrammarOfGraphics/minard.txt
dates = ['12/7/1812', '12/6/1812', '12/1/1812', '11/28/1812', '11/21/1812', '11/14/1812', '11/9/
1812', '10/24/1812']
dates = [dt.datetime.strptime(date,'%m/%d/%Y').date() for date in dates]
total_days_from_moscow = (dates[0] - dates[-1]).days  # total # of days on chart


# temp data - temps in °Re
temps_re = [-26.0, -30.0, -24.0, -20.0, -11.0, -21.0, -9.0, 0.0]

# conversion to Celcius and Fahrenheit
def re_to_c(temp_re):
    return round(temp_re * (5/4), 0)

def re_to_f(temp_re):
    return round((temp_re * 9/4) + 32, 0)

temps_c = [re_to_c(temp) for temp in temps_re]
temps_f = [re_to_f(temp) for temp in temps_re]

print("\n\nSupporting Data:\n\nTemperature data in:")
print("°re: ", temps_re)
print("°c:  ", temps_c)
print("°f:  ", temps_f, "\n")
print("Total # of days marching back from Moscow: ", total_days_from_moscow)

# setup chart
plt.figure(figsize=(30,8));
plt.xlim(dates[0], dates[-1])  # reverse x axis to matches Minard's chart
plt.xticks(dates, dates, rotation=25)
plt.title("March to Moscow - Comparison of Temperatures across three Temperature scales")
plt.xlabel("Date")
```

```
plt.ylabel("Normalized Temperature Scale°")
plt.plot(dates, temps_re, color="black", linestyle="dotted")
plt.plot(dates, temps_c, color="darkred")
plt.plot(dates, temps_f,color="darkblue")
plt.text(dates[-1], 2, "  Re freezing point (0°)", fontsize=12, fontstyle="italic", color="blac
k")
plt.text(dates[-1], -2, "  C freezing point (0°)", fontsize=12, fontstyle="italic", color="blac
k")
plt.text(dates[-1], 32, "  F freezing point (32°)", fontsize=12, fontstyle="italic", color="dark
blue")
plt.legend(("°Réaumur", "°Celcius", "°Fahrenheit"))
plt.show();
```

Supporting Data:

Temperature data in:
°re: [-26.0, -30.0, -24.0, -20.0, -11.0, -21.0, -9.0, 0.0]
°c: [-32.0, -38.0, -30.0, -25.0, -14.0, -26.0, -11.0, 0.0]
°f: [-26.0, -36.0, -22.0, -13.0, 7.0, -15.0, 12.0, 32.0]

Total # of days marching back from Moscow: 44



In [4]:
```
%%html
<!DOCTYPE html>
<html>
<head>
<style>
img.challenger {
  height: auto%;
  width: 90%;
}
</style>

<p style="font-weight:bold; font-size:185%;">Why is visualization important?  <i>Convince people
to Take Action</i></p><br>

</head>
<body>

<table> <tr> <th style="width:50%"></th> <th></th> </tr>
    <tr>
        <td><a href="https://en.wikipedia.org/wiki/Space_Shuttle_Challenger_disaster"><img class
="challenger" src="./images/challenger.photo.1110px-Challenger_explosion.jpg" alt="Space Shuttle
Challenger Disaster - 28 Jan 1986" title="Space Shuttle Challenger Disaster"/></a></td>
        <td style="vertical-align:top; text-align:left; font-size: 160%"> <strong>
            <b>Challenger Disaster - 28 Jan 1986</b><br><br>
            - All seven crew members died<br><br>
            - five NASA astronauts, one payload specialist, and a civilian school teacher<br><br
>
            - caused by the failure of O-ring seals in right solid rocket booster<br><br>
            - NASA disregarded engineers NO-LAUNCH warnings<br><br><br>
            - <i>...but the warnings were confusing</i><br><br></strong> </td>
    </tr>
    <tr>
        <td></td>
        <td></td>
    </tr>
    <tr >
            <td>
            </td>
        <td style="text-align:left; font-size: 120%">
            <a href="https://en.wikipedia.org/wiki/Space_Shuttle_Challenger_disaster"><strong>li
nk: challenger disaster - wikipedia</strong></a></td>
        <td>
```

```
    </tr >
    <tr>
            <td>
            </td>
            <td style="text-align:left; font-size: 120%">
            <a href="https://forum.nasaspaceflight.com/index.php?PHPSESSID=n2pbop2fh60010n76lmcn
h4po7&action=dlattach;topic=8535.0;attach=25186"><strong>link: Rogers Commission Report</strong>
</a>
        </td>

    </tr>
    <tr>
        <td>
        </td>
        <td style="text-align:left; font-size: 120%"> <br>
            <a href="https://en.wikipedia.org/wiki/Rogers_Commission_Report"><strong>link: tldr
 - wikipedia page on Rogers Commission Report</strong></a>
        </td>
    </tr>
        <tr>
            <td>
            </td>
            <td style="text-align:left; font-size: 120%">
                <a href="https://www.vice.com/en_us/article/kbb3qz/could-better-data-design-have
-prevented-challenger"><strong>link: presentation obscured lack of data</strong></a><br><br><str
ong>[credit: Wikipedia](https://en.wikipedia.org/wiki/Space_Shuttle_Challenger_disaster)</strong
>
            </td>
        </tr>
</table>
</body>
</html>
```

Why is visualization important? *Convince people to Take Action*


Space Shuttle Challenger Disaster - 28 Jan 1986 (https://en.wikipedia.org/wiki/Space_Shuttle_Challenger_disaster)

Challenger Disaster - 28 Jan 1986

- All seven crew members died

- five NASA astronauts, one payload spec school teacher

- caused by the failure of O-ring seals in r booster

- NASA disregarded engineers NO-LAUN(

- ...but the warnings were confusing

link: challenger disaster - wikipedia (https://en.wikipedia.org/wiki/Space_Shuttle_Challenger_dis

link: Rogers Commission Report (https://forum.nasaspac PHPSESSID=n2pbop2fh60010n76lmcnh4po7&action=dlatt

link: tldr - wikipedia page on Rogers Commission Repo (https://en.wikipedia.org/wiki/Rogers_Commission_Report)

link: presentation obscured lack of data (https://www.vice.com/en_us/article/kbb3qz/could-better-dat challenger)

[credit: Wikipedia](https://en.wikipedia.org/wiki/Space_S

Background: What happened?

- Problem was with a booster rocket

Plume from Booster (https://en.wikipedia.org/wiki/Space_Shuttle_Challenger_disaster)

- Rockets built in sections

- O-rings
1- Sealed joints on booster
2- NOT designed for cold temps ==> NOT flexible
3- Did not stop flames
4- Flames hit the liquid hydrogen fuel tank

- Org culture + decision-making processes were key factors to the accident

- **link: step-by-step graphic** (https://upload.wikimedia.org/wikipedia/commons/4/4f/Challeng

[credit: Wikipedia](https://en.wikipedia.org/wiki/Space_Shuttle_Challenger_disaster)

Why were the warnings confusing?

...because of 13 pages of data like this... (https://history.nasa.gov/rogersrep/v4part6.htm#1)


Challenger - Engineer Warning -SRM Field Joints


Challenger - Engineer Warning - History of O-Ring Damage


Challenger - Engineer Warning - O-Ring Damage vs Temp

- credit: Rogers Commission Report; NASA original source (https://history.nasa.gov/rogersrep/v4part6.htm#1)

These 13 pages...

- Defined the framework for the launch/no-launch decision

- Omitted data from 22 launches

- Obscured a crucial lack of data

- Were just not compelling to decision makers

...There must be a better way

Envisioning Information - Edward Tufte (https://www.edwardtufte.com/tufte/books_ei)

- Distills Challenger data from Roberts Report and makes a compelling visualization

- Let's replicate that in pandas & matplotlib

In [5]:
```
import pandas as pd
df = pd.read_csv("./images/challenger_o-ring_damage_data.csv", encoding="ISO-8859-1")

df["Date"] =  pd.to_datetime(df["Date"].str.replace("."," "))
# df["Date"] =  pd.to_datetime(df["Date"], format='%d.%m.%y')

df[["Erosion incidents", "Blow-by incidents"]] = df[["Erosion incidents", "Blow-by incidents"]].
fillna(axis=1, value=0)
df[["Temperature °F", "Erosion incidents", "Blow-by incidents", "Damage index"]] = df[["Temperat
ure °F", "Erosion incidents", "Blow-by incidents", "Damage index"]].apply(pd.to_numeric).astype(
'int')

df[["Flight", "Comments"]] = df[["Flight", "Comments"]].astype('category')
df["Comments"] = df["Comments"].cat.add_categories("no comment listed")
df[["Comments"]] = df[["Comments"]].fillna("no comment listed")

# sort temperature values to use as X axis, so we can plot results
df = df.sort_values("Temperature °F")
df.head(10)
```

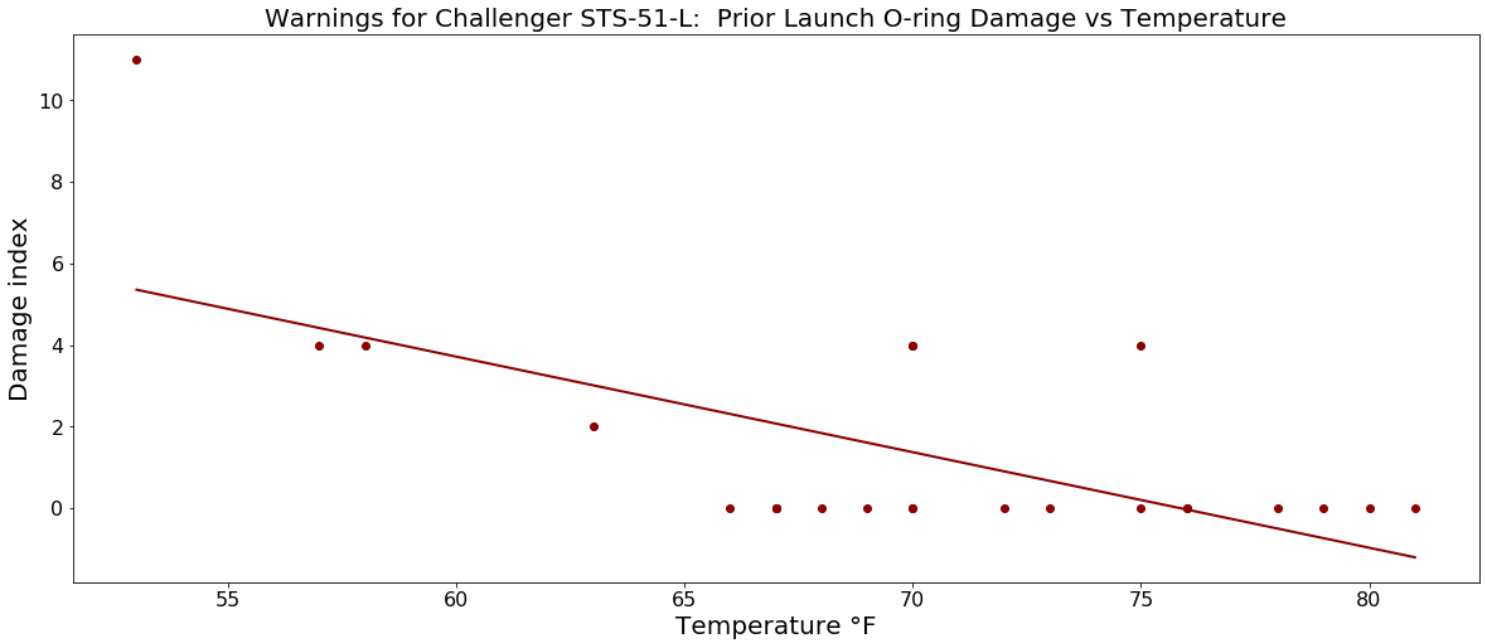
Out[5]:

	Flight	Date	Temperature °F	Erosion incidents	Blow-by incidents	Damage index	Comments
0	51-C	1985-01-24	53	3	2	11	Most erosion any flight; blow-by; back-up ring...
1	41-B	1984-02-03	57	1	0	4	Deep, extensive erosion
2	61-C	1986-01-12	58	1	0	4	O-ring erosion on launch two weeks before Chal...
3	41-C	1984-04-06	63	1	0	2	O-rings showed signs of heating, but no damage.
4	1	1981-04-12	66	0	0	0	no comment listed
5	6	1983-04-04	67	0	0	0	no comment listed
6	51-A	1984-11-08	67	0	0	0	no comment listed
7	51-D	1985-04-12	67	0	0	0	no comment listed
8	5	1982-11-11	68	0	0	0	no comment listed
9	3	1982-03-22	69	0	0	0	no comment listed

In [6]:
```
# what is the relationship between temperature and damage data?
model = LinearRegression()

# reshape our data since there is only one feature
# X = df.loc[:, "Temperature °F"].values.reshape(-1, 1) # get values & convert to a numpy array
 has failed intermittently, while iloc works consistently
# Y = df.loc[:, "Damage index"].values.reshape(-1, 1)  # get values and convert into 1 column nu
mpy array
X = df.iloc[:, 2].values.reshape(-1, 1)  # another way to do this using iloc
Y = df.iloc[:, 5].values.reshape(-1, 1)  # ibid

# run regression
linear_regressor = LinearRegression()  # create object for the class
linear_regressor.fit(X, Y)  # perform linear regression
Y_pred = linear_regressor.predict(X)  # make predictions
```

Out[6]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [7]:
```
# plot the NASA data

plt.figure(figsize=(20,8))
plt.scatter(X, Y, color="darkred")
plt.plot(X, Y_pred, color="darkred", linewidth=2)
plt.title("Warnings for Challenger STS-51-L:  Prior Launch O-ring Damage vs Temperature", fontsi
ze=20)
plt.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel("Temperature °F", fontsize=20)
plt.ylabel("Damage index", fontsize=20);
```

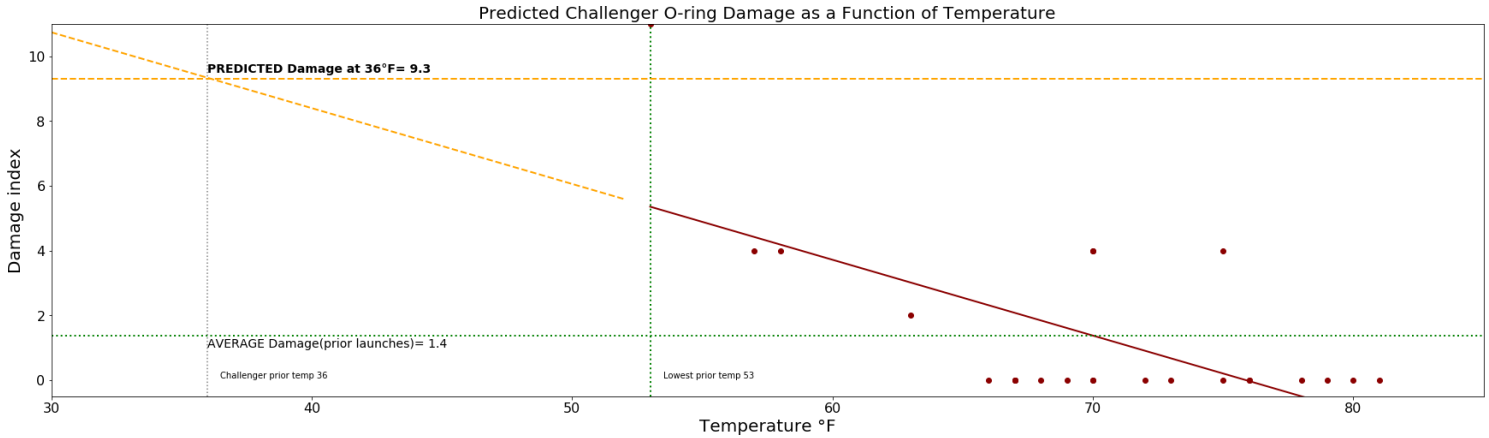


In [8]:
```
# predict damage for temperature at time of launch
challenger_launch_temp = 36

# generate array of lower temps -- below temps of previous launches
Z = [x for x in range(30, 53)]
Z = np.asarray(Z).reshape(-1, 1)
Z_pred = linear_regressor.predict(Z)  # make predictions

# helpful historic datapoints
prior_launch_temp_min = df.iloc[:, 2].min().round(1)
prior_damage_avg = df.loc[:, "Damage index"].mean().round(1)  # using historic data
challenger_launch_temp_np = np.asarray(challenger_launch_temp).reshape(-1, 1)
challenger_damage_predicted = round(linear_regressor.predict(challenger_launch_temp_np)[0][0],1)
```

In [9]:
```
# create plot; use functions so we can build up graph step by step

def plot_history():
    plt.figure(figsize=(30,8))
    plt.title("Predicted Challenger O-ring Damage as a Function of Temperature", fontsize=20)
    plt.tick_params(axis='both', which='major', labelsize=16)
    plt.xlabel("Temperature °F", fontsize=20)
    plt.ylabel("Damage index", fontsize=20)
    plt.xlim(30, 85)
    plt.ylim(-0.5, 11)
    plt.scatter(X, Y, color="darkred")
    plt.plot(X, Y_pred, color='darkred', linewidth=2)

    return True


def plot_key_temps():
    # highlight key temperatures
    plt.axvline(53, color="green", linestyle=":", linewidth=2)
    plt.text(prior_launch_temp_min + 0.5, 0.05, f"Lowest prior temp {prior_launch_temp_min} ", f
ontsize=10)
    plt.axvline(36, color="grey", linestyle=":")
    plt.text(challenger_launch_temp + 0.5, 0.05, f"Challenger prior temp {challenger_launch_tem
p} ", fontsize=10)

    return True


def plot_predicted_damage():
    plt.plot(Z, Z_pred, color='orange', linestyle="--", linewidth=2)

    return True


def plot_annotations():
    plt.axhline(df.loc[:, "Damage index"].mean(), color="green", linestyle="dotted", linewidth=2
)
    plt.axhline(challenger_damage_predicted, color="orange", linestyle="--", linewidth=2)

    # add some labels to hammer it home
    plt.text(36, 1, f"AVERAGE Damage(prior launches)= {prior_damage_avg} ", fontsize=14)
    plt.text(36, 9.5, f"PREDICTED Damage at 36°F= {challenger_damage_predicted} ", fontsize=14,
```

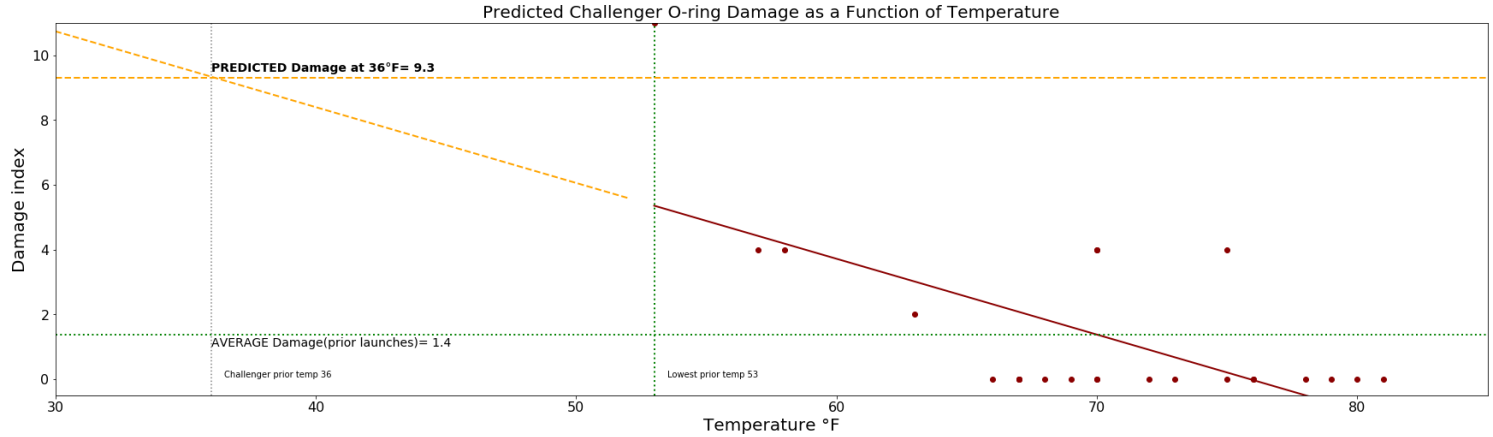
```
fontweight="bold")

    return True
```

In [10]:
```
# show the plot
# uncomment lines below through discussion
plot_history()
plot_key_temps()
plot_predicted_damage()
plot_annotations()

plt.show();
```



In [11]:
```
# reference slide:  same slide as previous - with everything in one shot
# create plot; use functions so we can build up graph step by step

def plot_history():
    plt.figure(figsize=(30,8))
    plt.title("Predicted Challenger O-ring Damage as a Function of Temperature", fontsize=20)
    plt.tick_params(axis='both', which='major', labelsize=16)
    plt.xlabel("Temperature °F", fontsize=20)
    plt.ylabel("Damage index", fontsize=20)
    plt.xlim(30, 85)
    plt.ylim(-0.5, 11)
    plt.scatter(X, Y, color="darkred")
    plt.plot(X, Y_pred, color='darkred', linewidth=2)

    return True


def plot_key_temps():
    # highlight key temperatures
    plt.axvline(53, color="green", linestyle=":", linewidth=2)
    plt.text(prior_launch_temp_min + 0.5, 0.05, f"Lowest prior temp {prior_launch_temp_min} ", f
ontsize=10)
    plt.axvline(36, color="grey", linestyle=":")
    plt.text(challenger_launch_temp + 0.5, 0.05, f"Challenger prior temp {challenger_launch_tem
p} ", fontsize=10)

    return True


def plot_predicted_damage():
    plt.plot(Z, Z_pred, color='orange', linestyle="--", linewidth=2)

    return True


def plot_annotations():
    plt.axhline(df.loc[:, "Damage index"].mean(), color="green", linestyle="dotted", linewidth=2
)
    plt.axhline(challenger_damage_predicted, color="orange", linestyle="--", linewidth=2)

    # add some labels to hammer it home
    plt.text(36, 1, f"AVERAGE Damage(prior launches)= {prior_damage_avg} ", fontsize=14)
```

```
    plt.text(36, 9.5, f"PREDICTED Damage at 36°F= {challenger_damage_predicted} ", fontsize=14,
fontweight="bold")

    return True

plot_history()
plot_key_temps()
plot_predicted_damage()
plot_annotations()

plt.show();
```



Why is visualization important? *Numbers don't always show important facts*

Anscome's Quartet:

In [12]:
```
import seaborn as sns
sns.set(style="ticks")

# Load the example dataset for Anscombe's quartet
df = sns.load_dataset("anscombe")

print("Mean of each dataset in df")
display(df.groupby(["dataset"]).mean())
print("\n\nCovariance - measures how changes are associated between variables")
display(df.groupby(["dataset"]).cov().round(1))

print("\n\nPearson's Correlation Coeffient - measures linear correlation")
display(df.groupby(["dataset"]).corr(method="pearson"))
```

Mean of each dataset in df

dataset	x	y
I	9.0	7.500909
II	9.0	7.500909
III	9.0	7.500000
IV	9.0	7.500909

Covariance - measures how changes are associated between variables

dataset		x	y
I	x	11.0	5.5
	y	5.5	4.1
II	x	11.0	5.5
	y	5.5	4.1
III	x	11.0	5.5
	y	5.5	4.1
IV	x	11.0	5.5
	y	5.5	4.1

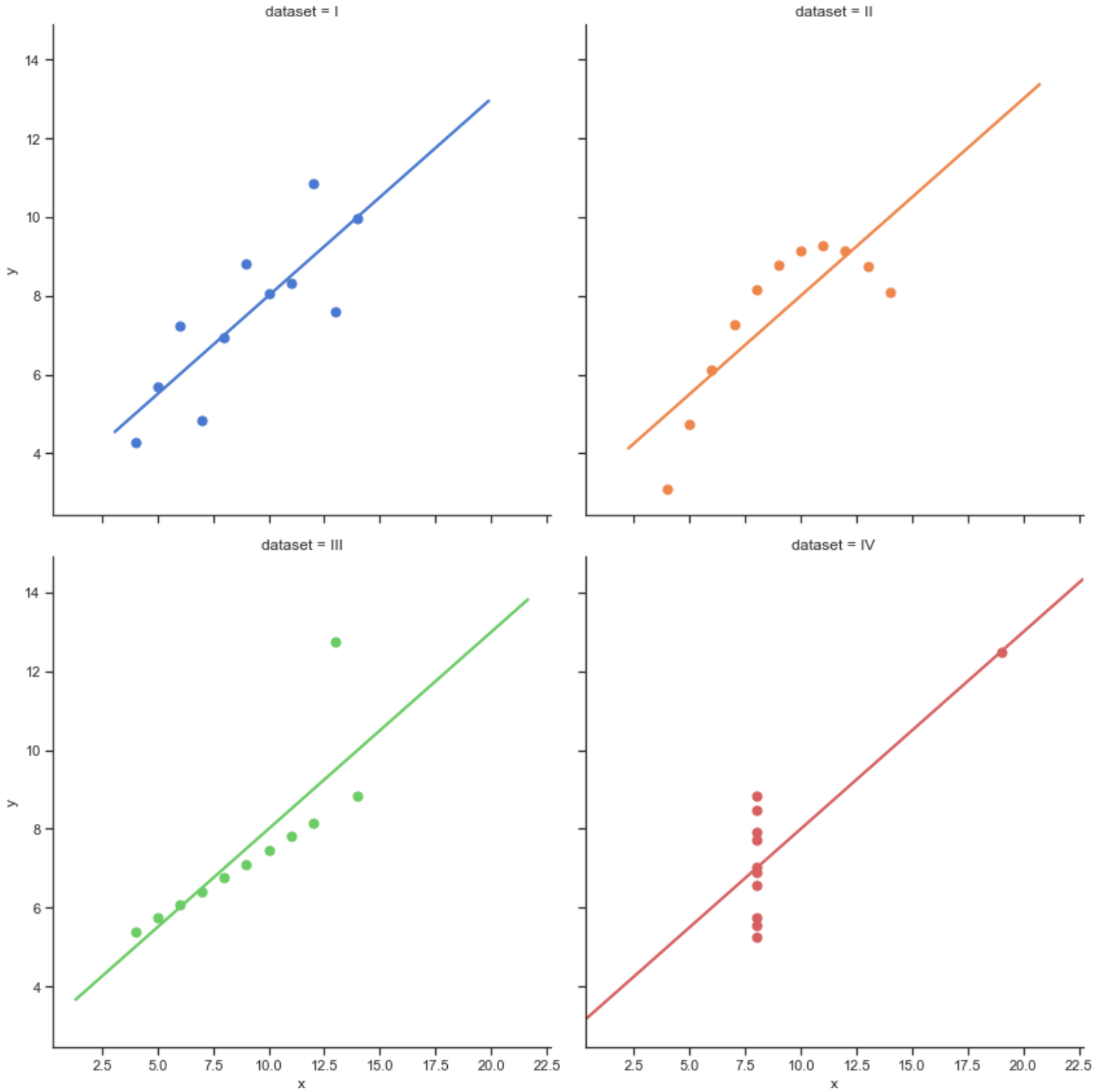
Pearson's Correlation Coeffient - measures linear correlation

		x	y
dataset			
I	**x**	1.000000	0.816421
	y	0.816421	1.000000
II	**x**	1.000000	0.816237
	y	0.816237	1.000000
III	**x**	1.000000	0.816287
	y	0.816287	1.000000
IV	**x**	1.000000	0.816521
	y	0.816521	1.000000

In [13]:
```
# plot these datasets
sns.lmplot(x="x", y="y", col="dataset", hue="dataset", data=df,
           col_wrap=2, ci=None, palette="muted", height=6,
           scatter_kws={"s": 50, "alpha": 1})

print("\nCredit:  https://seaborn.pydata.org/examples/anscombes_quartet.html")
```

Out[13]: <seaborn.axisgrid.FacetGrid at 0x2b608047348>

Credit: https://seaborn.pydata.org/examples/anscombes_quartet.html


dataset = I
dataset = II
dataset = III
dataset = IV
y
x
14
12
10
8
6
4
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5

credit: Wikipedia
[Pure witchcraft from: Alberto Cairo's Datasaurus](https://www.autodeskresearch.com/publications/samestats)

Python/Seaborm page to generate and plot your own (https://seaborn.pydata.org/examples/anscombes_quartet.html)

2- Think about the right visual approach

Grammar of Graphics

- **Construct graphics in a layered approach**
- **From Leland Wilkinson's book of the same name**
- **Drove creation of ggplot (R) and plotnine (python)**

- resources to check out:
https://towardsdatascience.com/murdering-a-legendary-data-story-what-can-we-learn-from-a-grammar-of-graphics-ad6ca42f5e30 (https://towardsdatascience.com/murdering-a-legendary-data-story-what-can-we-learn-from-a-grammar-of-graphics-ad6ca42f5e30)
https://www.slideshare.net/kesarifms/grammar-of-graphics-the-secret-sauce-of-powerful-data-stories (https://www.slideshare.net/kesarifms/grammar-of-graphics-the-secret-sauce-of-powerful-data-stories)

[credit: Leland Wilkinson, Grammar of Graphics](https://www.amazon.com/Grammar-Graphics-Statistics-Computing/dp/0387245448)

How I might think about creating visualiztions

Big Questions

- what data do I have?
- what is my goal?
- who is my audience?

2nd Order questions:

- keep it simple (less is more)
- choose the right chart structure
- show all the data
- organize / sort data
- accurate scaling
- use predictable patterns
- use color carefully -> implies value
- use text carefully & intentionally
- interactive or static

Find the graphic you need = objective + data you have


Data to Vis Website (https://www.data-to-viz.com/)

credit: Data-to-Viz site - free, interactive tool (https://www.data-to-viz.com/)

Visualization - Leaders to Follow (unordered & incomplete list!)

Theory & Design

Edward Tufte (https://www.edwardtufte.com/tufte/?gclid=EAIaIQobChMIvIul89f55QIVMhh9Ch3Tmwg3EAAYASAAEgKoAfD_BwE)
Alberto Cairo (http://albertocairo.com/)
David McAndless / Information is Beautiful (https://en.wikipedia.org/wiki/David_McCandless)

D3

Mike Bostock - D3 (https://bost.ocks.org/mike/)
Nadieh Bremer (https://www.visualcinnamon.com/)
Shirley Wu (https://sxywu.com/)
Elijah Meeks (https://medium.com/@Elijah_Meeks)
D3 (https://d3js.org/)

Publications

Economist (https://www.economist.com/)
Information is Beautiful (https://informationisbeautiful.net/)
New York Times (https://www.nytimes.com/)
The Guardian (https://www.theguardian.com/technology/data-visualisation)

Apps

what makes a good visualization? source: information is beautiful (https://informationisbeautiful.net/visualizations/what-makes-a-good-data-visualization/)
Observable - notebooks for data viz (https://observablehq.com/)
bl.ocks.org/ (https://bl.ocks.org/)
Visualization Universe (http://visualizationuniverse.com/)

3- Get a snapshot of the python visualization universe

- PyVis landscape overview (https://pyviz.org/overviews/index.html)
- PyCon 2017 - Pythons Visualization Landscape by Jake VanderPlas (https://speakerdeck.com/jakevdp/pythons-visualization-landscape-pycon-2017)
- credit- graphic on following page: Jake VanderPlas (https://speakerdeck.com/jakevdp/pythons-visualization-landscape-pycon-2017)

In [14]:
```
# Focus of this talk
from itertools import cycle
import ipywidgets as widgets
from IPython.display import display
from IPython.display import clear_output

# create list of images to rotate through
image_set = ["./images/pyvis_landscape_overview_2019_landscape-talk_highlights.png", "./images/p
yvis_landscape_overview_2019_landscape-colors.png"]
images = [widgets.Image(value=open(name, "rb").read()) for name in image_set]
imagecycle = cycle(images)  # iterator of images

button = widgets.Button(description="Swap Image");
output = widgets.Output();
credit = "credit: Jake VanderPlas"
display(credit, button, widgets.Image(value=open("./images/pyvis_landscape_overview_2019_landsca
pe-colors.png", "rb").read()))

def on_button_clicked(b):
    display(clear_output(wait=True), credit, button, next(imagecycle))

button.on_click(on_button_clicked)
```

```
'credit: Jake VanderPlas'
```

Community Question - "can you also touch on plotly and dash with pros/cons?"

[Anaconda Articla: "Datavis - why so many libraries" (https://www.anaconda.com/python-data-visualization-2018-why-so-many-libraries/)](https://www.anaconda.com/python-data-visualization-2018-why-so-many-libraries/)

	Matplotlib	Bokeh	Plotly
release year	2003	2013	2013
framework - front end	mpld3	Javascript	Javascript
backend	many renderers	Tornado	Flask
Supporting Libraries	Seaborn	Batteries Included	Plotly Express Cufflinks
Dashboard Framework	None	Panel	Dash
Corporate Sponsor	None	Anaconda	Plotly - the company
Maybe you didn't know	- Oldest visualization library	- Interactive - Intelligent Errors (...similar...) - Great docs	- Interactive - Used to require plotly account - Has supporting gui (Chart Studio)
Pandas	Yes	Yes	Yes
Jupyter Notebook & JupyterLab	Yes	Yes	Yes

Awesome library you should try	**Yes**	**Yes**	**Yes**

Quiz - Tufte

What have we learned so far?

0 - Sighted people have powerful visual perception abilities

1- Images effectively distill info

2- Images give insights pure numbers do not

3- Visualizations can be complex - but there are tools to help

4- Python has a lot of visualization libraries

link to our next chapter - matplotlib (http://localhost:8888/notebooks/notebooks/01_matplotlib_chapter.ip

In [ ]: