Data Visualization

2017-06-02

Agenda

- Review
- Visual encoding using color (continued)
  - Neural color signals and HSL color encoding
  - High detail versus low detail
  - Perceptual uniformity
- Visualizing distributions
- Visualizing high dimensional data
- Interactive applications
  - Event loops
  - Examples of increasing complexity in Python
  - Model-View-Controller framework
- Presentations

Review: Purposes of Visualization

- Supporting exploratory data analysis (exploratory)
- Explaining or supporting presentation (explanatory)

Review: Attribute Domains & Visual Encodings

- Nominal (=, ≠)
  - Types and categories (mathematical *set*)
- Ordinal (=, ≠, ≤)
  - Has an order (mathematical *set* with *order relation*)
- Quantitative
  - Interval (=, ≠, ≤, +, -)
    - Has a meaningful difference between values (mathematical *group*)
    - E.g: Dates, location, geometric points, temperature (C and F)
  - Ratio (=, ≠, ≤, +, -, x, ÷)
    - Has a meaningful one and zero point and ratio between values (mathematical *field*)
    - E.g: Distance, mass, temperature (K), time, counts
- Topological
  - Connectivity, inclusion

Review: Graphical Integrity

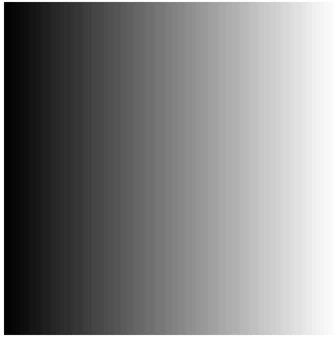
- Proportionality
  - The *physical measurements of the representation of the data* should be proportional to the data itself
- Matching dimensions
  - Beware of pitfalls of using area, volume, and perspective
- Providing context
  - Anchor the audience

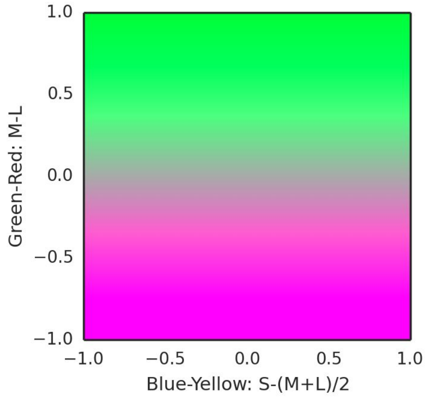
Review: Maximizing Impact

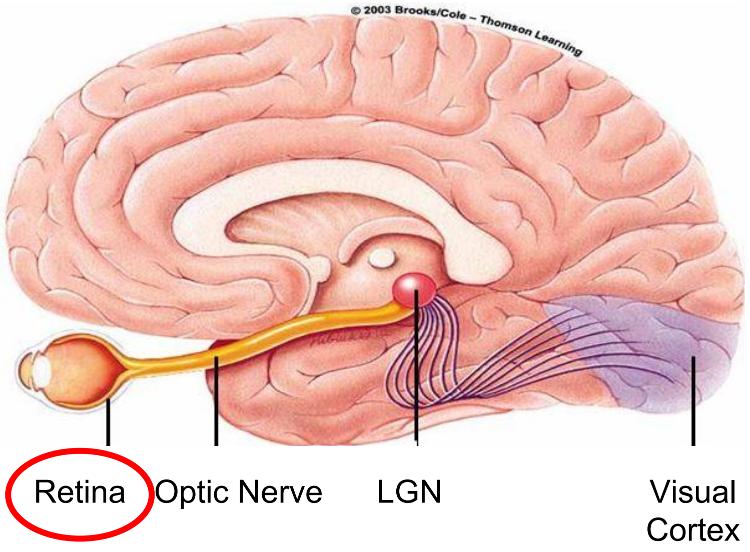
- Erase redundant data
  - Representing the same data multiple times
  - Within reason
- Erase metadata
  - Pixels giving context to the data
  - Within reason
- Avoid chartjunk
  - Not necessary to understand the data
  - Distract the viewer
- Use pre-attentive stimuli when possible

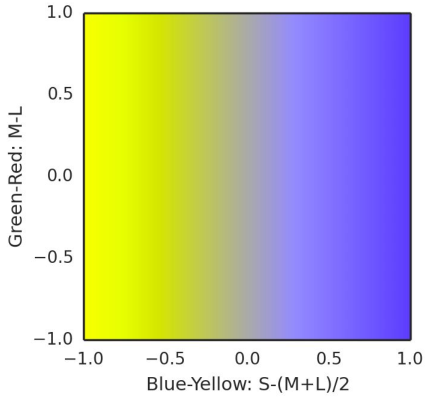
Visual Encoding Using Color

Neural Color Signals

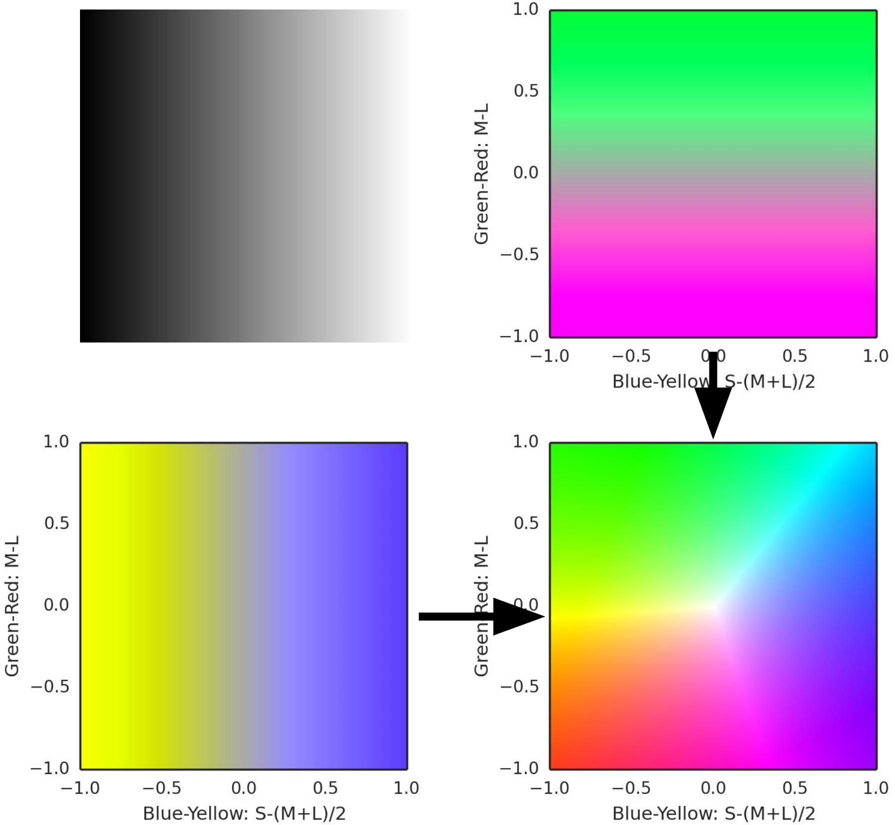


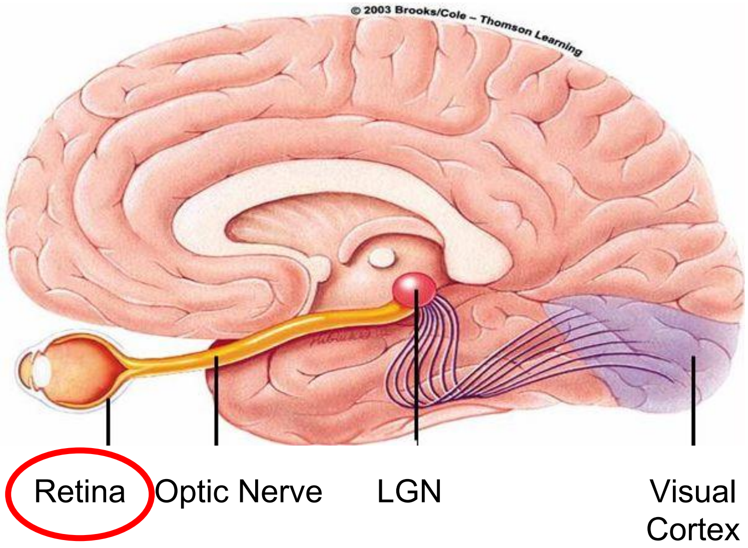




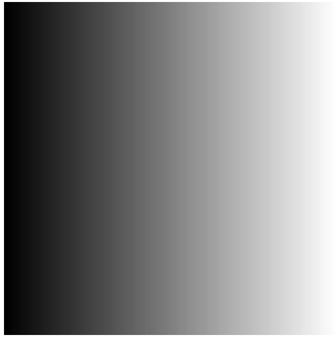


Neural Color Signals



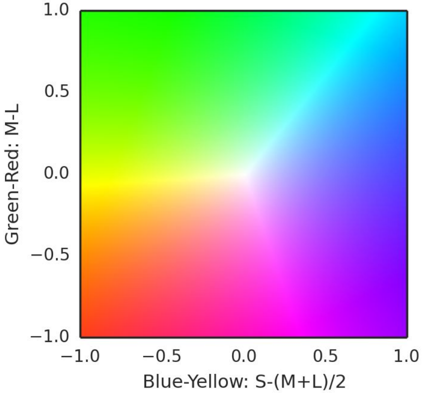


Luminance and Chrominance

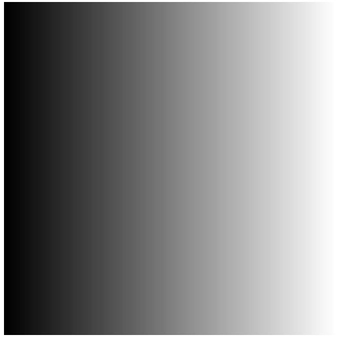


Luminance

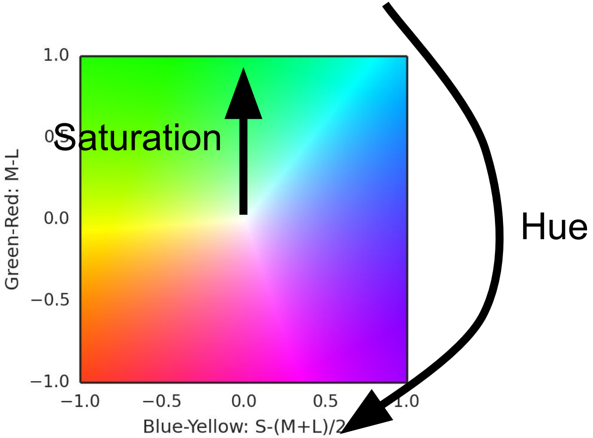
Chrominance



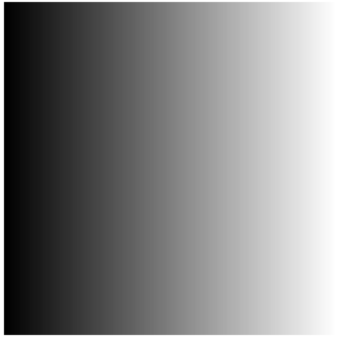
Neural Color Signals to HSL Encoding



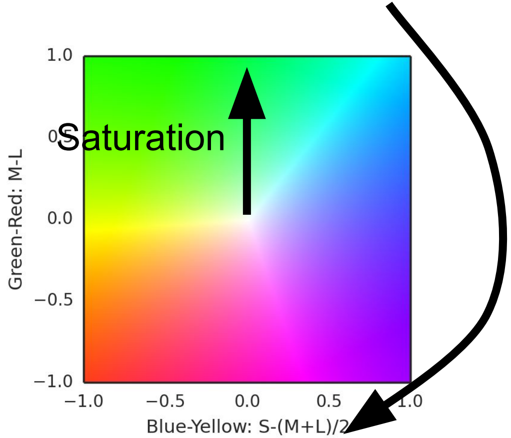
Luminance



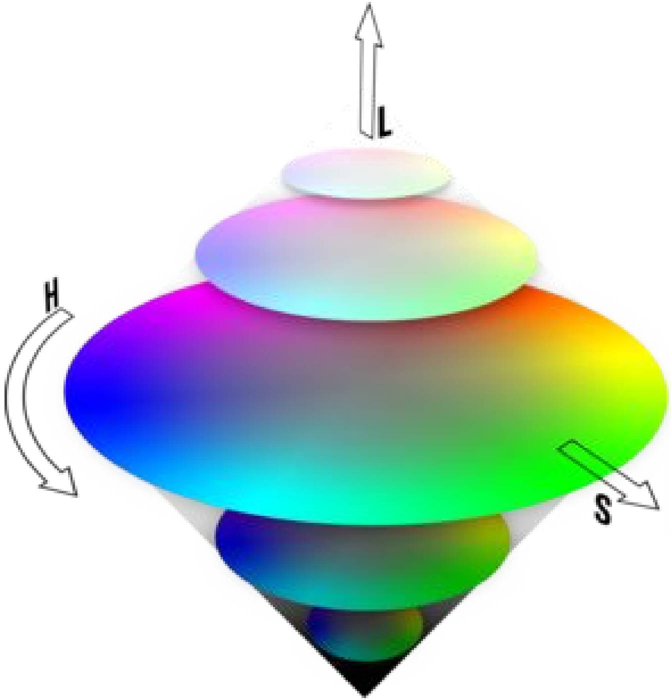
Neural Color Signals to HSL Encoding



Luminance



Hue



HSL Encoding

Hue:

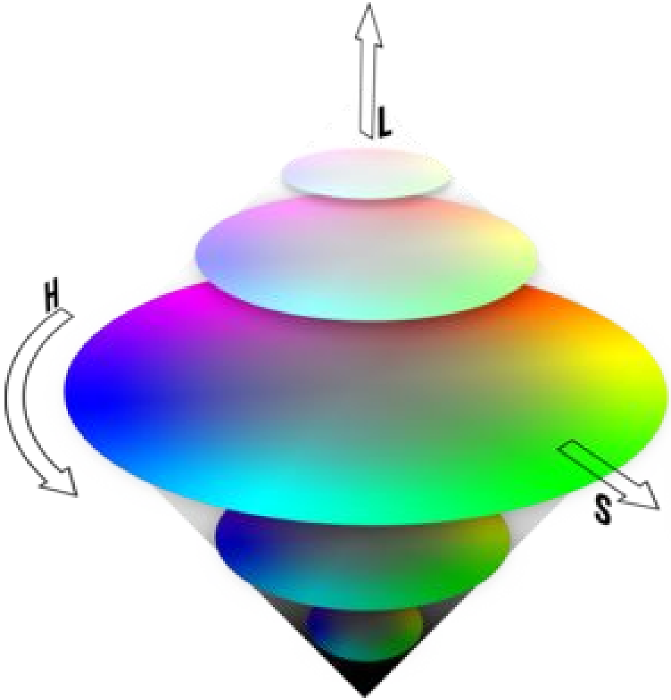
Nominal

Saturation (and opposing processes):

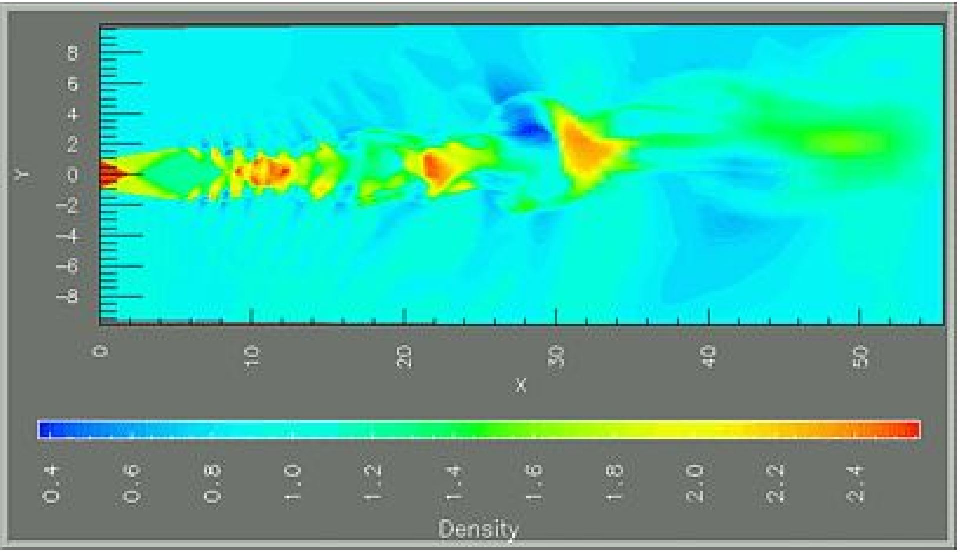
Quantitative

Luminance/Value:

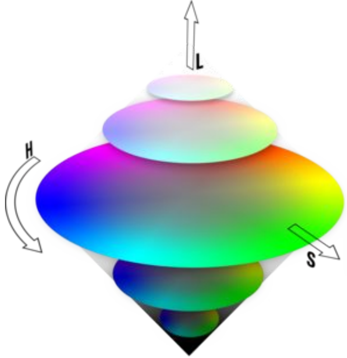
Quantitative

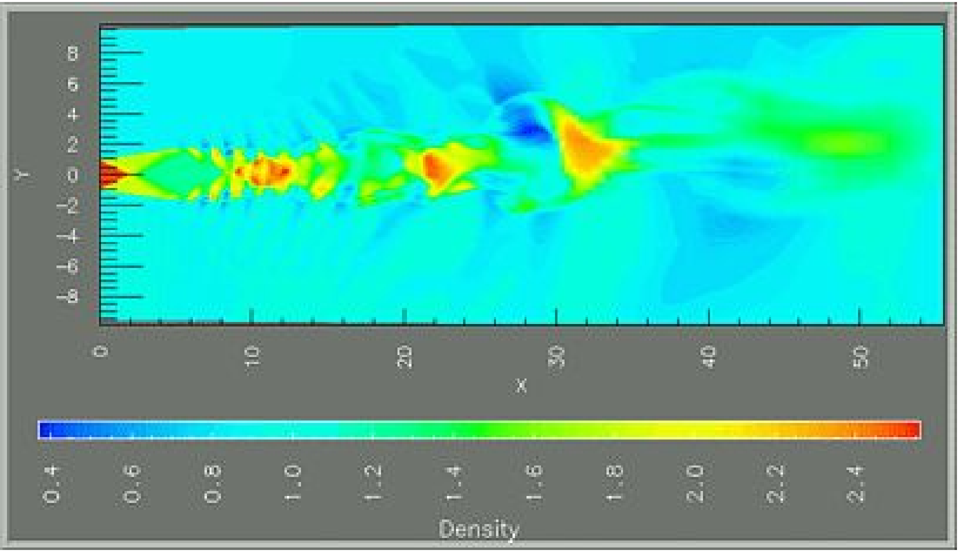


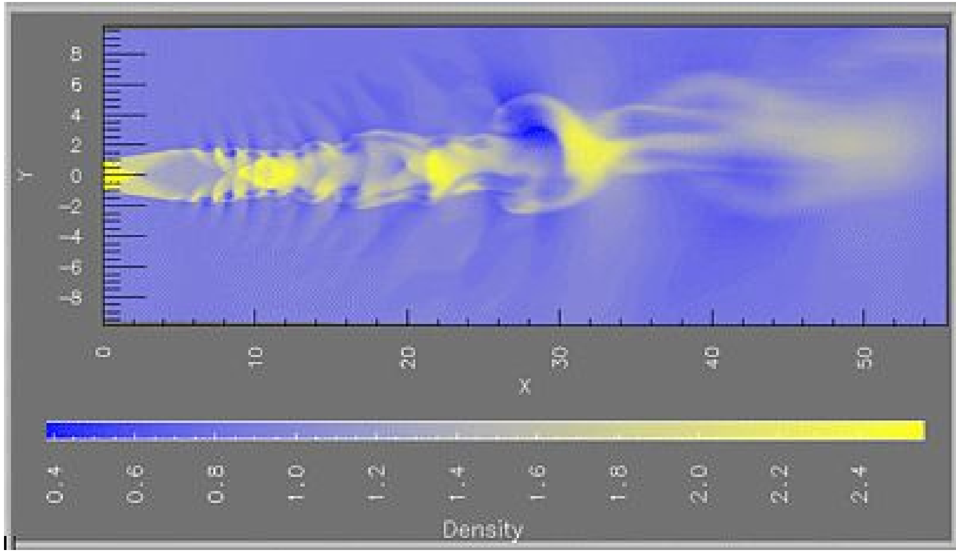
Examples



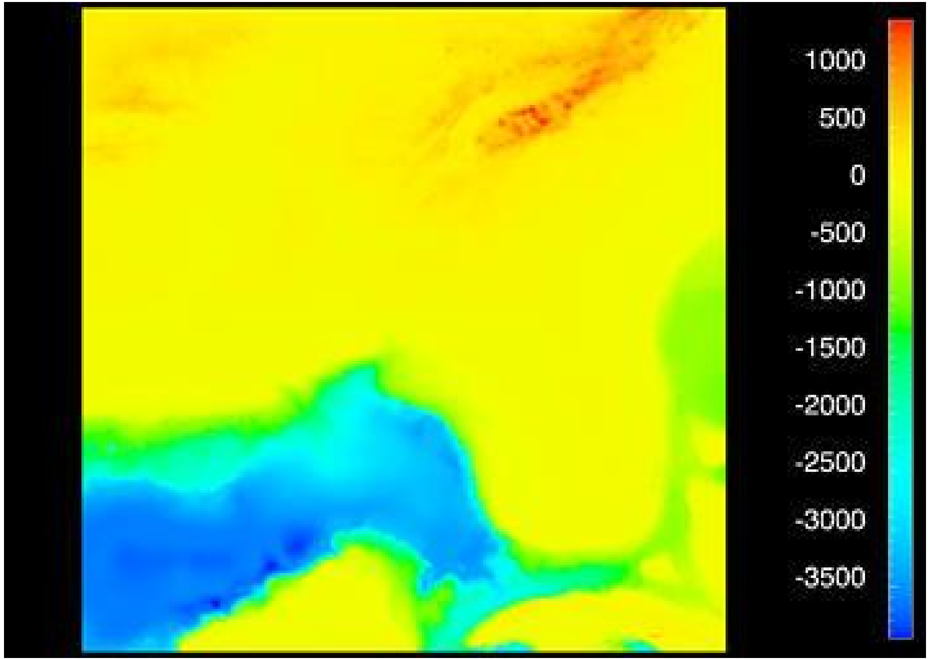
Y
X
Density

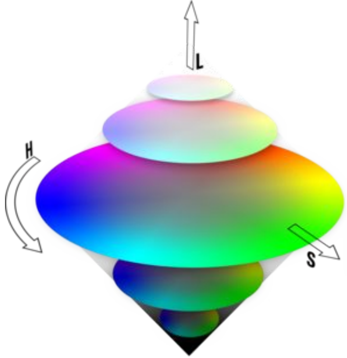

L
H
S


Y
8 6 4 2 0 -2 -4 -6 -8
0 10 20 30 40 50
X
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4
Density

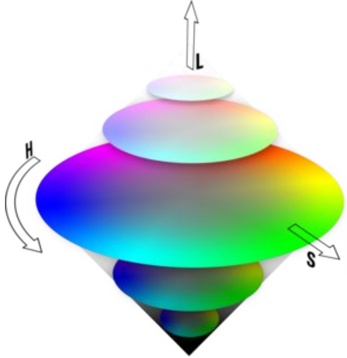

Y
8 6 4 2 0 -2 -4 -6 -8
0 10 20 30 40 50
X
0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4
Density

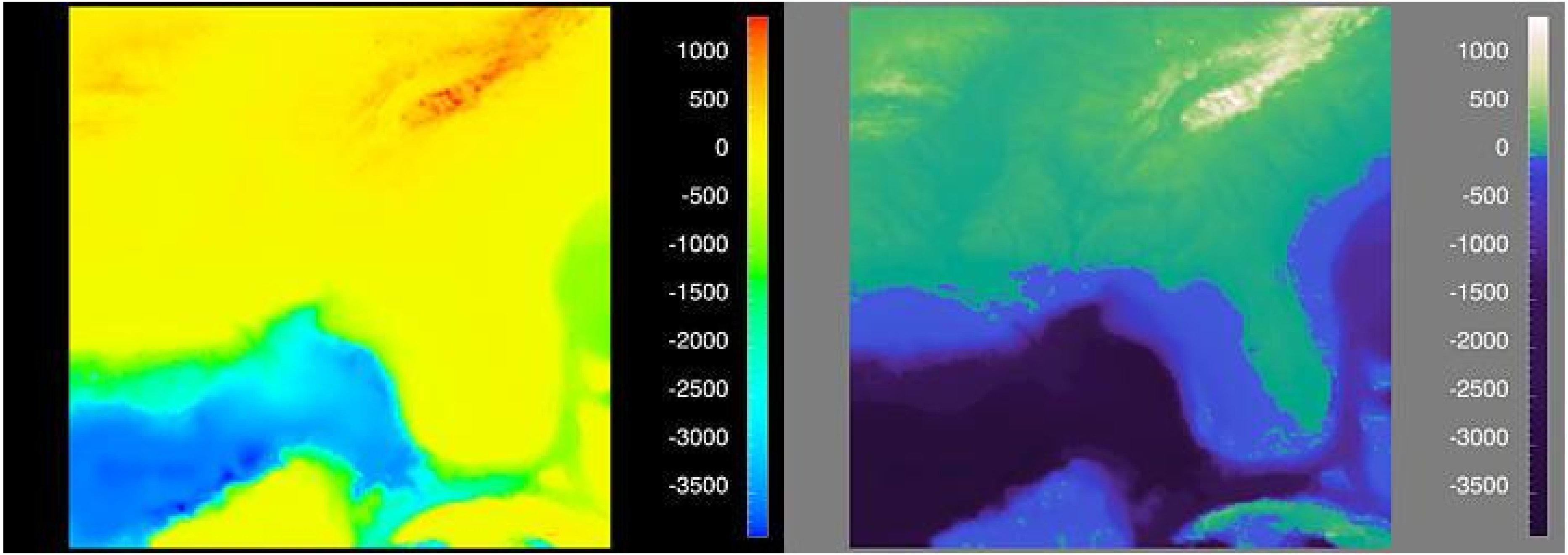
Elevation Example



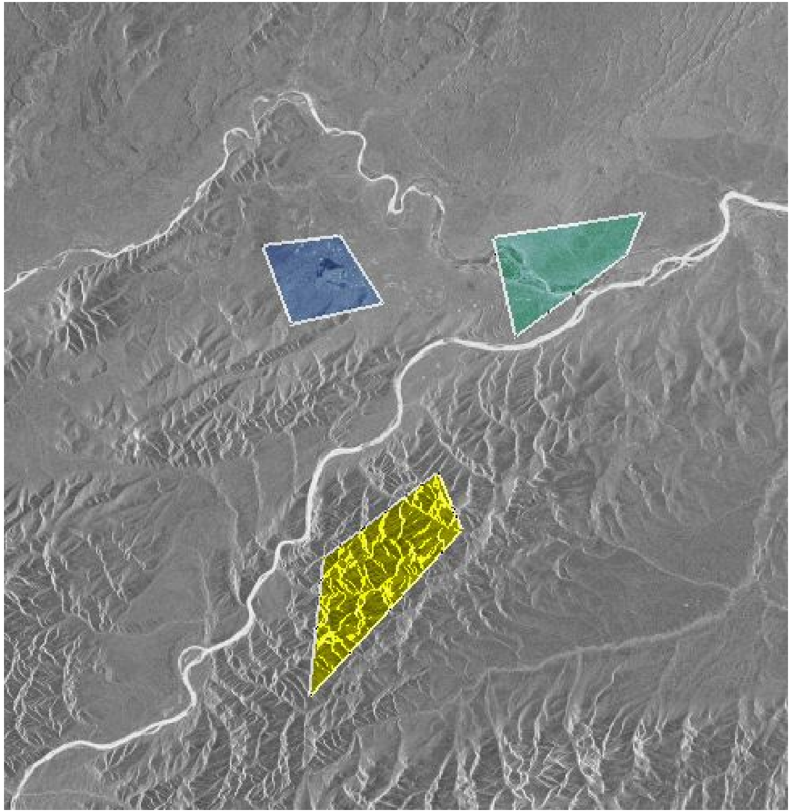


Elevation Example





Example: Encoding multiple dimensions



High vs Low Frequency Data
(an example)

HSL Encoding

Hue:

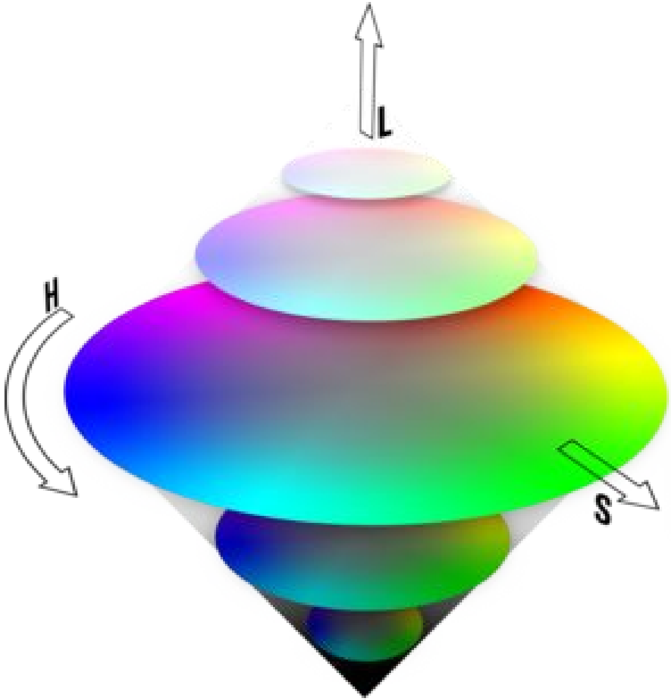
Nominal

Saturation (and opposing processes):

Quantitative

Luminance/Value:

Quantitative



Original

Luminance

+

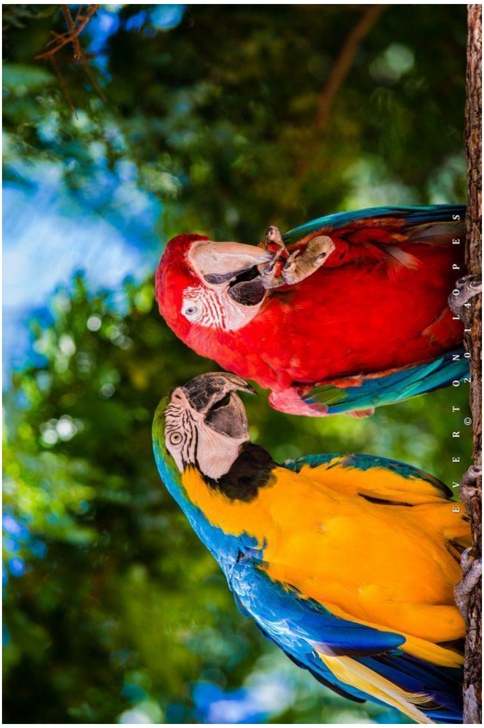
Red-Green

+

Blue-Yellow

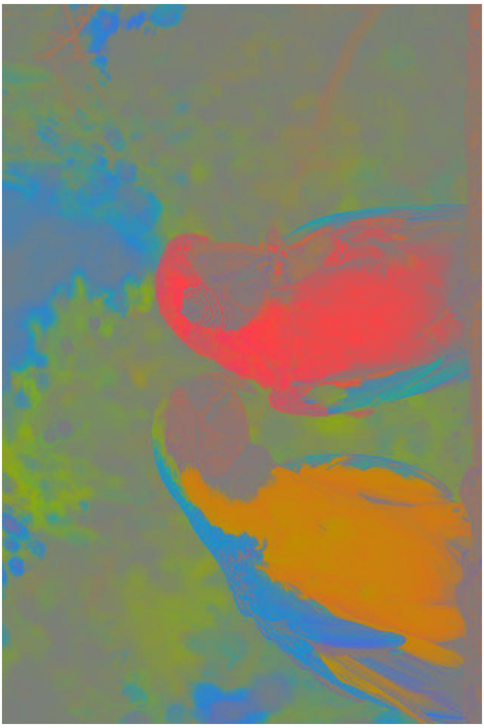
=

Original

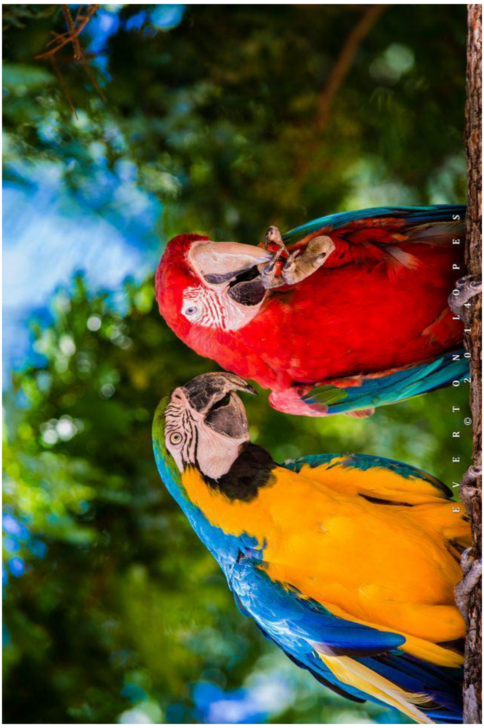


=

Chrominance
(Blue-Yellow and Red-Green)



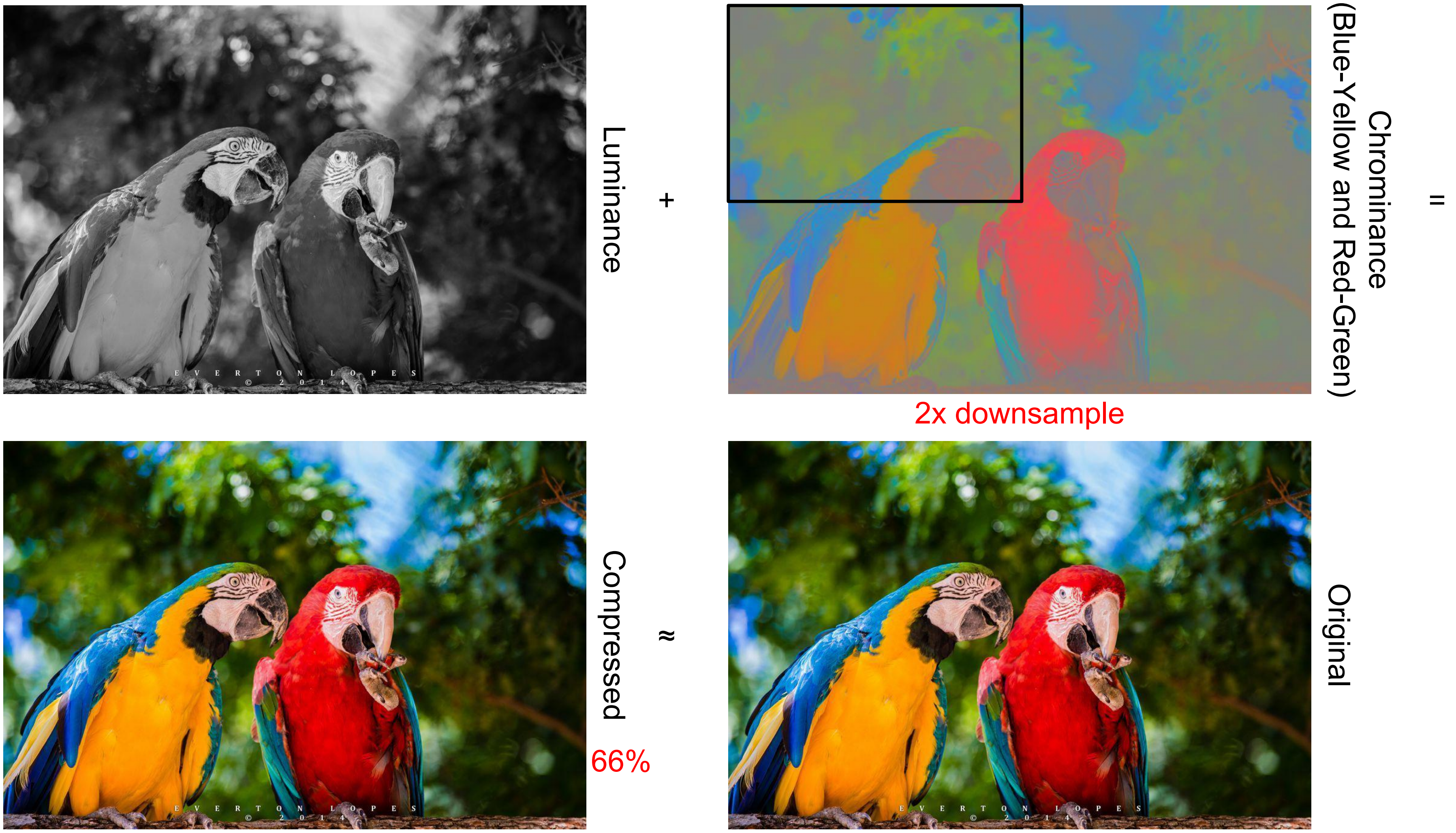
Original

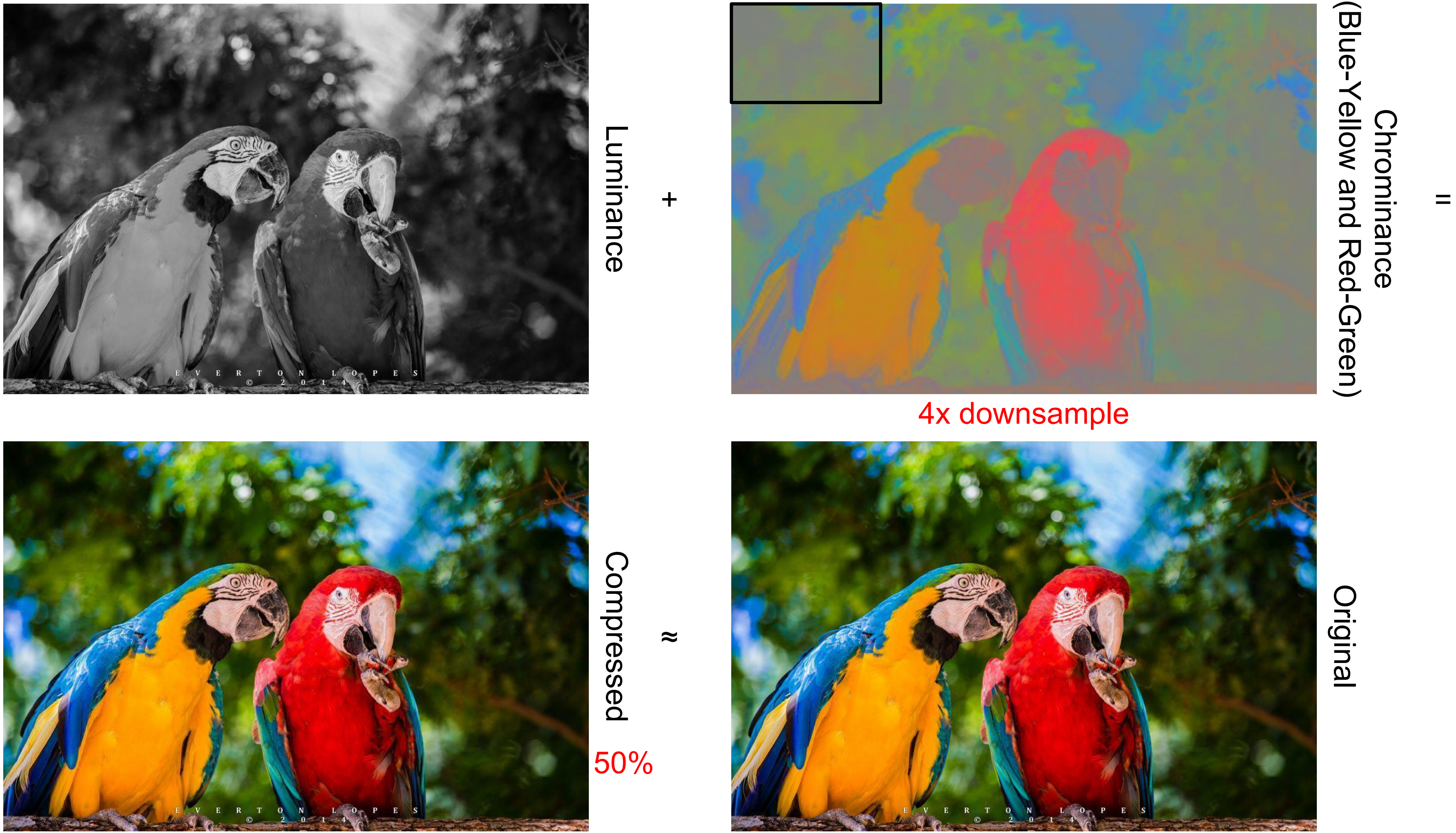


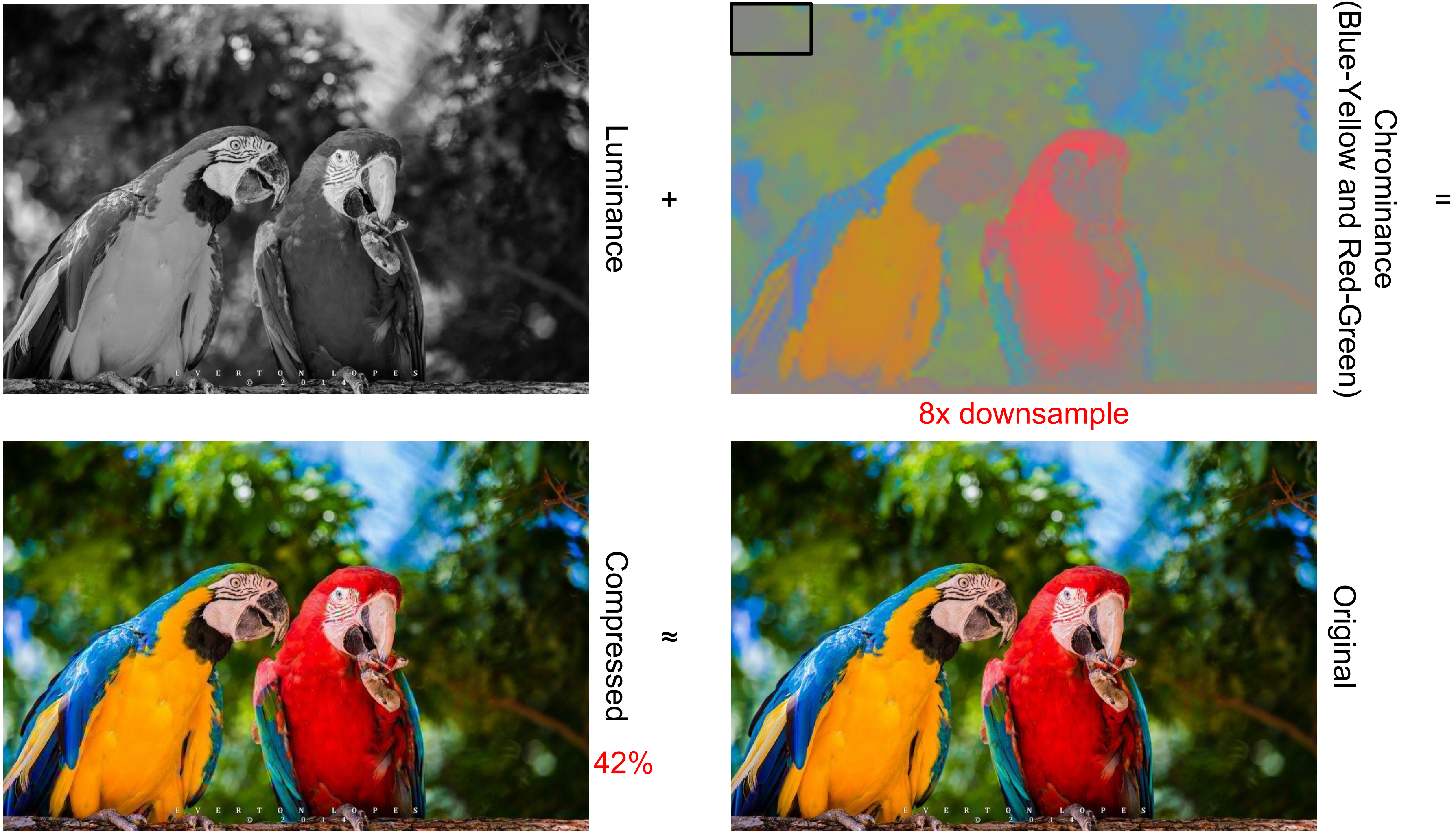
+

=

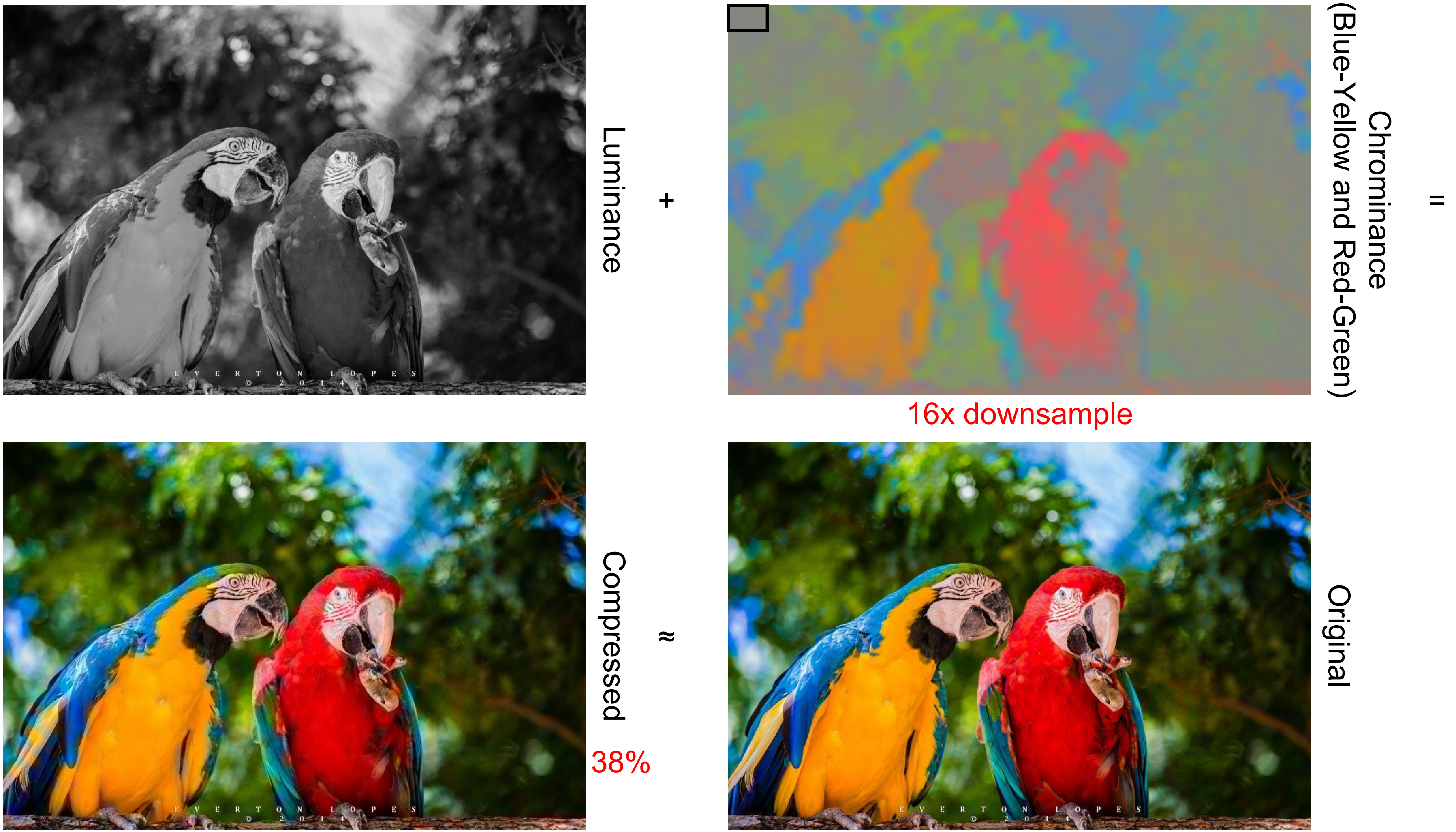
Luminance

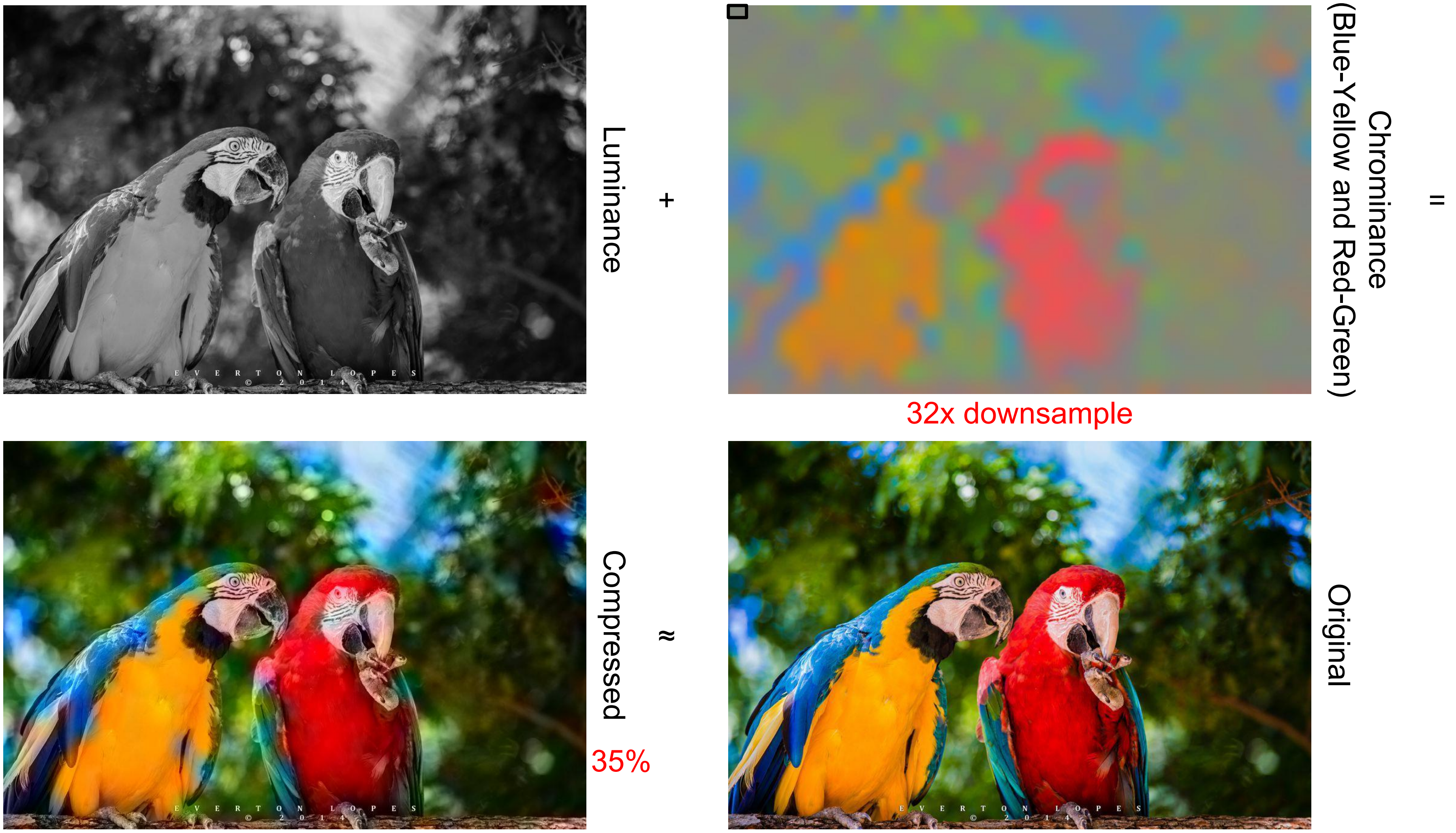

Luminance
+
Chrominance
(Blue-Yellow and Red-Green)
=
2x downsample
Compressed
≈
66%
Original
EVERTON LOPES © 2014


Luminance
+
Chrominance
(Blue-Yellow and Red-Green)
=
4x downsample
Compressed
50%
≈
Original

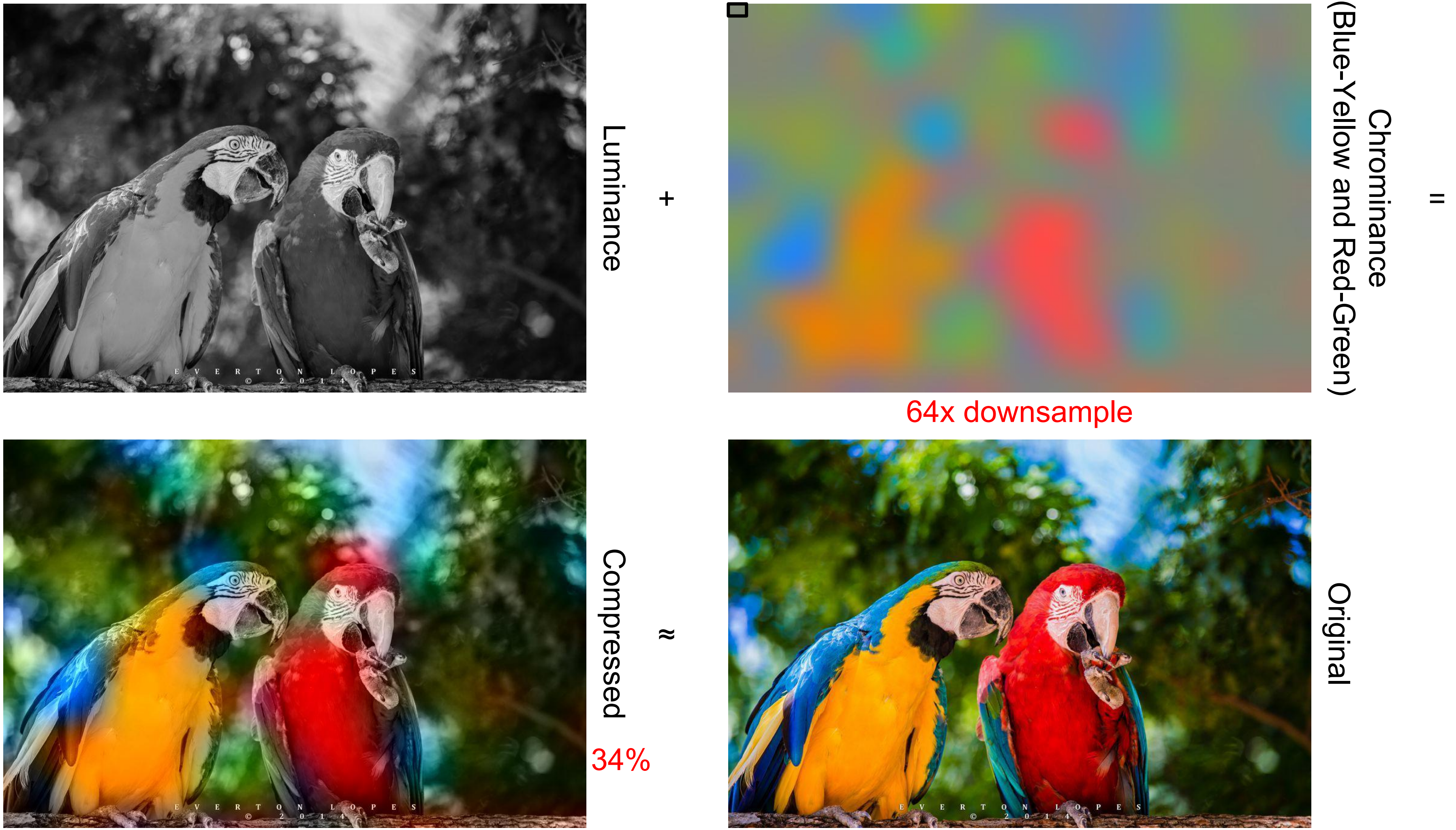


Luminance
+
Chrominance
(Blue-Yellow and Red-Green)
=
8x downsample
Compressed
≈
42%
Original
E V E R T O N L O P E S
© 2 0 1 4


Luminance
+
Chrominance
(Blue-Yellow and Red-Green)
=
16x downsample
Compressed
≈
38%
Original



Luminance
+
Chrominance
(Blue-Yellow and Red-Green)
=
32x downsample
Compressed
≈
35%
Original


Luminance
+
Chrominance
(Blue-Yellow and Red-Green)
=
64x downsample
Compressed
≈
34%
Original
E V E R T O N L O P E S
© 2 0 1 4

HSL Encoding: High vs Low Frequency

Hue:

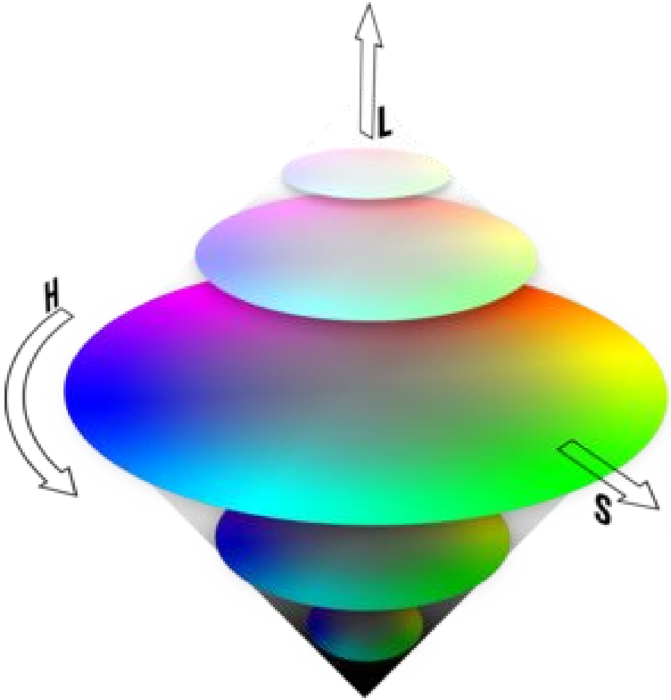
Nominal

Saturation (and opposing processes):

Low Frequency (Non-Detailed) Quantitative

Luminance/Value:

High Frequency (Detailed) Quantitative



Graphical Integrity - Perceptual Uniformity

Remember...

- Proportionality
    - The *physical measurements of the representation of the data* should be proportional to the data itself

...what would that mean for color?

The Just-Noticeable Difference

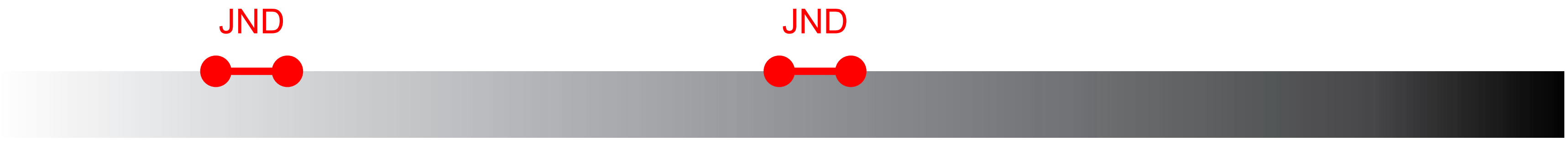
Perceptually uniform: A change of the same amount in a color value should produce a change of about the same visual importance.

Just noticeable difference (JND): the amount a stimulus has to change for someone to notice it.

The Just-Noticeable Difference

Perceptually uniform: A change of the same amount in a color value should produce a change of about the same visual importance.

Just noticeable difference (JND): the amount a stimulus has to change for someone to notice it.



Colormaps in Matplotlib

Considerations:

- Colorblindness
- Perceptual uniformity
- Print compatibility

http://matplotlib.org/examples/color/colormaps_reference.html

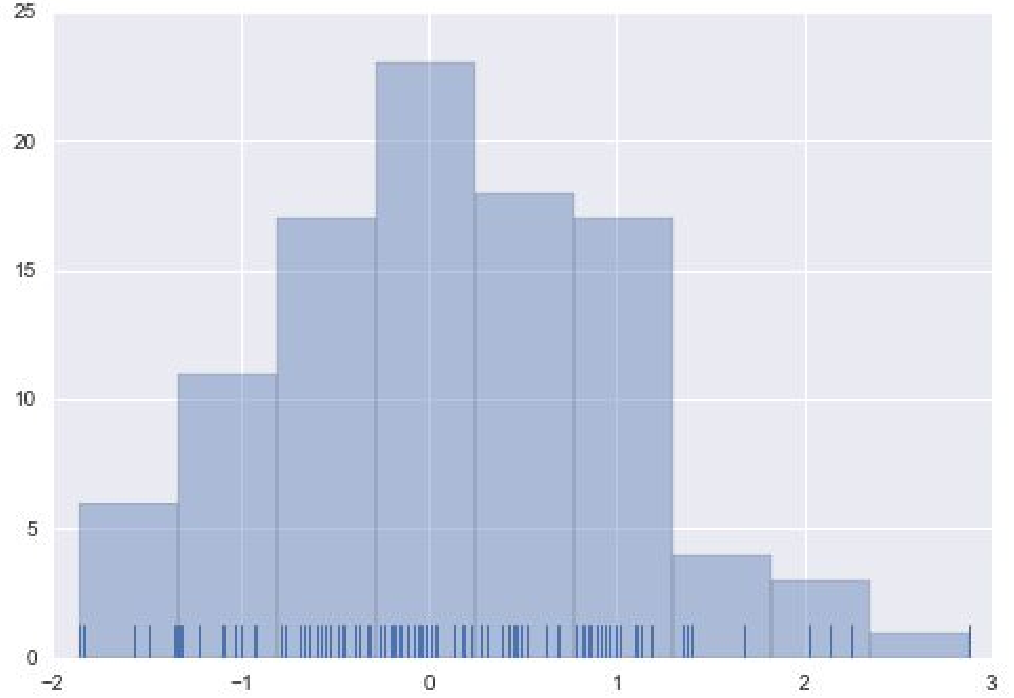
http://matplotlib.org/users/dflt_style_changes.html

Color Takeaways

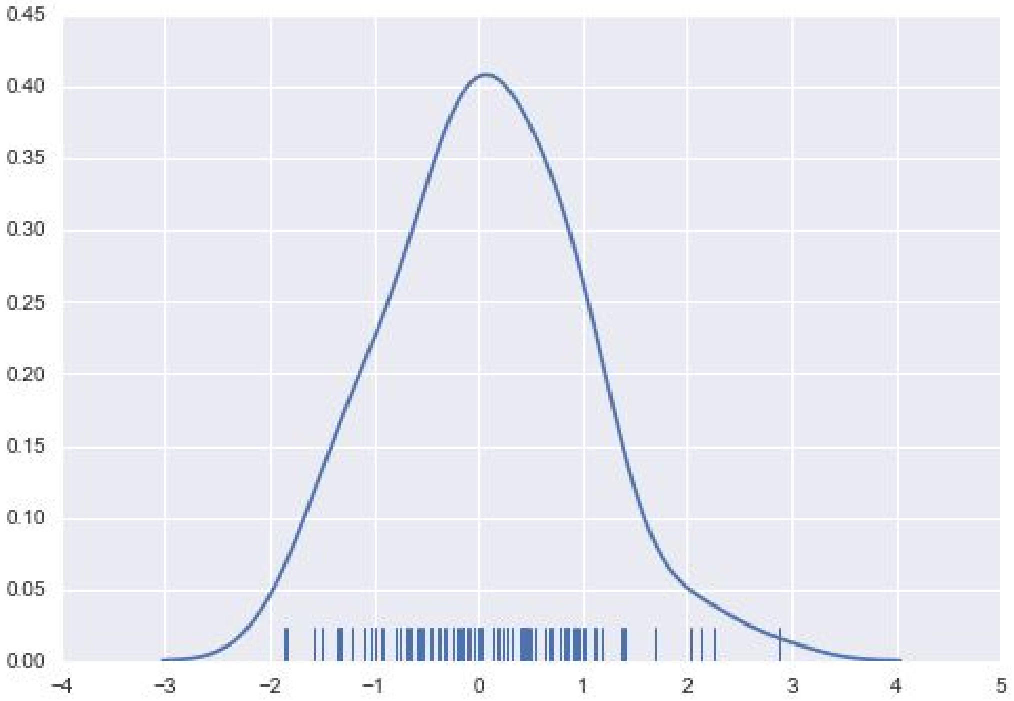
- Never use the jet colormap!
- Use a colormap that is
  - perceptually uniform
  - colorblindness-insensitive
- When making your own color maps:
  - Hue: Nominal
  - Saturation: Low frequency quantitative
  - Luminance: High frequency quantitative
  - Segmentation for context

Visualizing Distributions

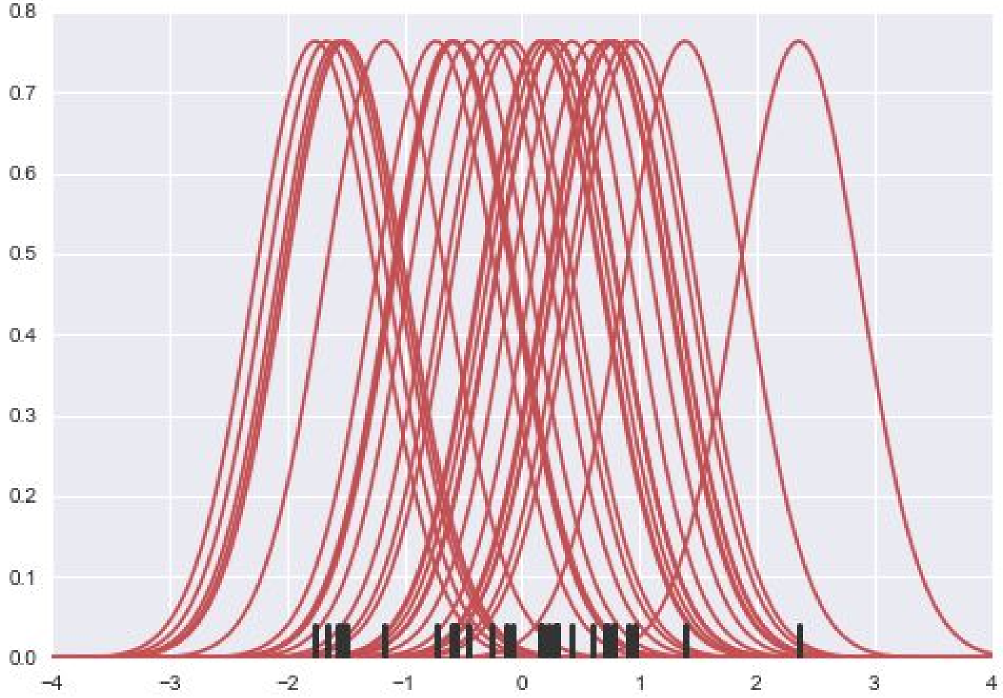
Histograms



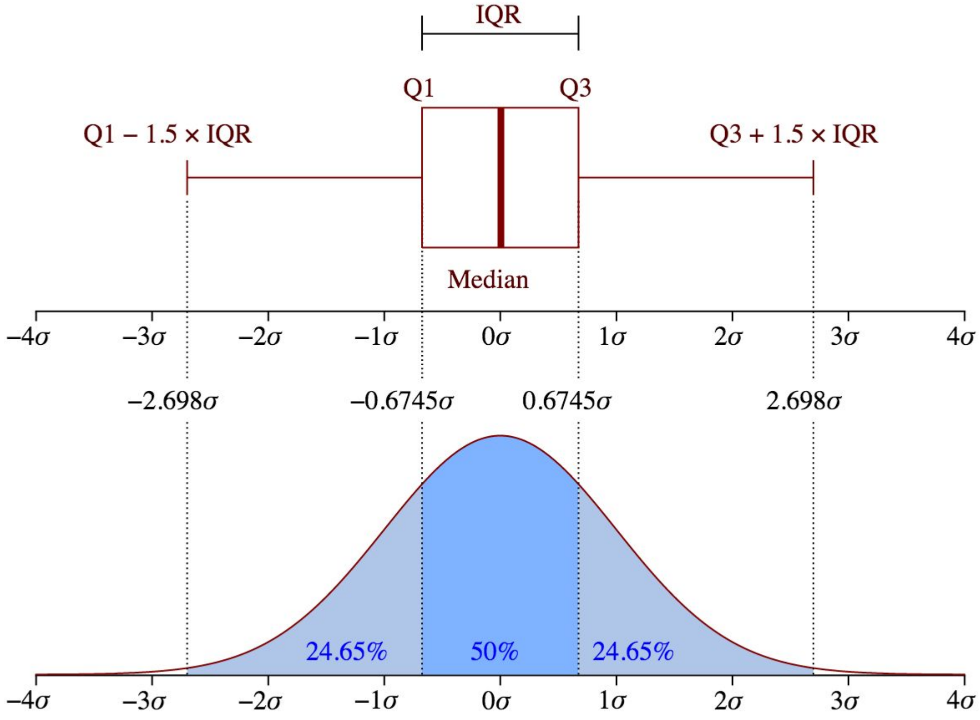
Kernel Density Estimate Plots



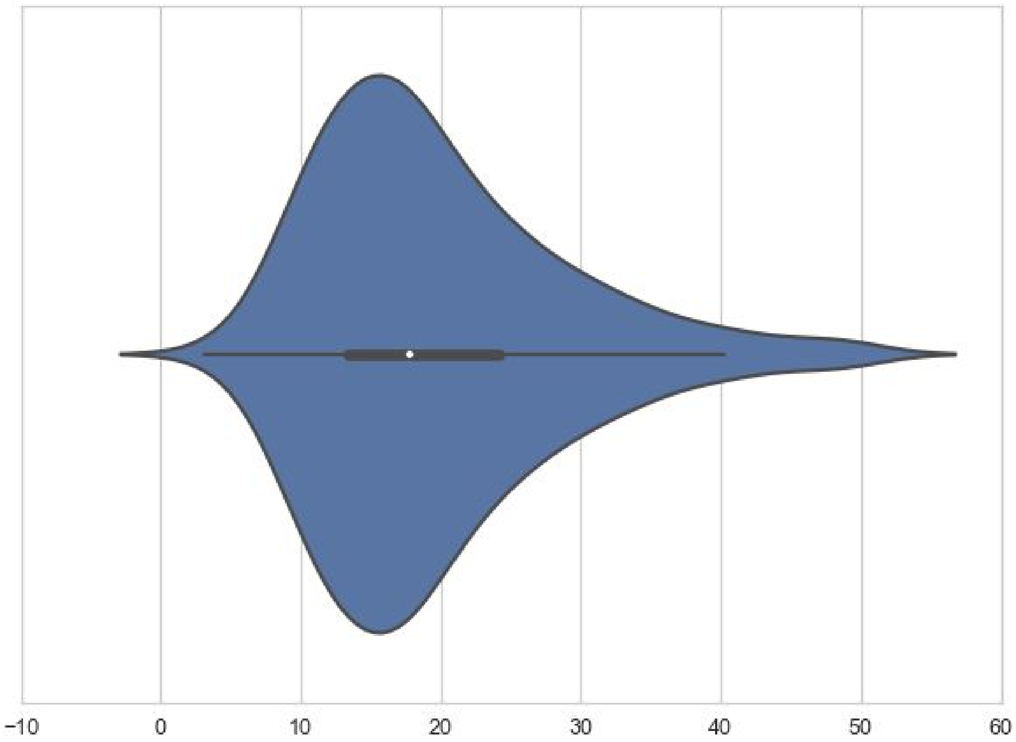
Kernel Density Estimate Plots



Box Plots


IQR
Q1
Q3
Q1 − 1.5 × IQR
Q3 + 1.5 × IQR
Median
−4σ
−3σ
−2σ
−1σ
0σ
1σ
2σ
3σ
4σ
−2.698σ
−0.6745σ
0.6745σ
2.698σ
24.65%
50%
24.65%
−4σ
−3σ
−2σ
−1σ
0σ
1σ
2σ
3σ
4σ

Violin Plots



Notebook Example

Visualizing Distributions Take-Aways

- Histogram vs KDE plot trade-offs
  - Beware of artifacts from parameter choices
- Comparing distributions options
  - Box plots
  - Violin plots
  - Overlapping histograms/KDE plots

Visualizing High Dimensional Data

Senate Data from 2008

Available online

Name	*Motion 1*	*Motion 2*	*Motion 3*	*Motion 4*	...
Alexander	Yea	Yea	Yea	Nay	
Biden	Yea	Nay	Nay	Yea	
Bond	Yea	Abstain	Nay	Nay	
Boxer	Yea	Yea	Yea	Abstain	
...					...

Senate Data from 2008

Available online

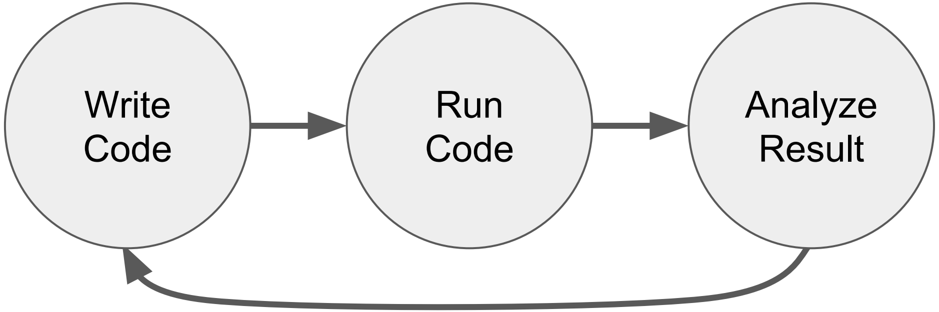
Name	*Motion 1*	*Motion 2*	*Motion 3*	*Motion 4*	...
Alexander	1	1	1	-1	
Biden	1	-1	-1	1	
Bond	1	0	-1	-1	
Boxer	1	1	1	0	
...					...

Back to the Notebook for a Moment

So, who was that outlier?

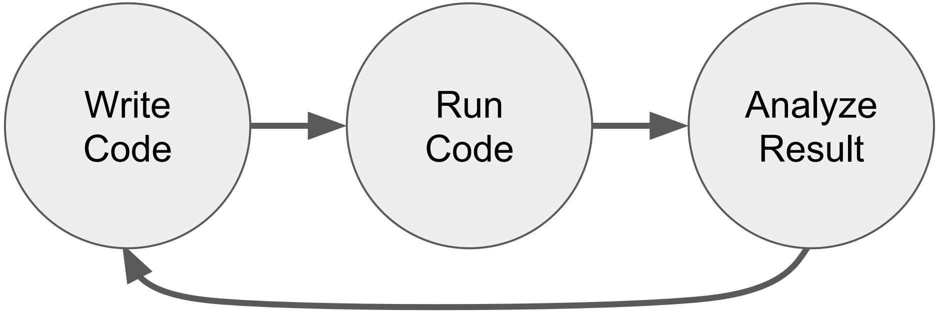
Interactive Applications

Data Scientist’s Workflow:

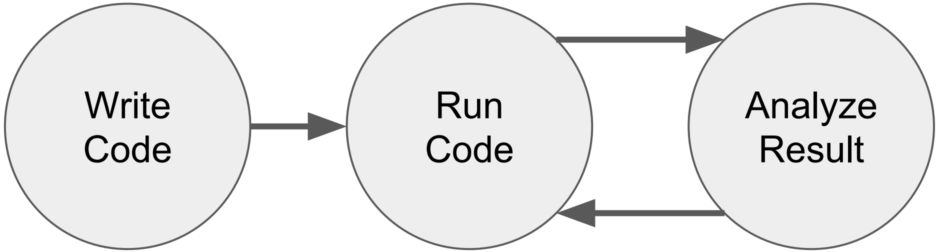


Interactive Applications

Data Scientist’s Workflow:



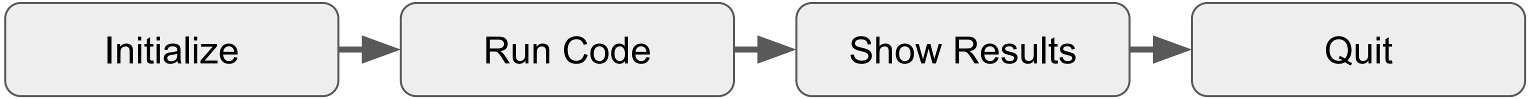
More Efficient Data Scientist’s Workflow:



A non-Notebook example

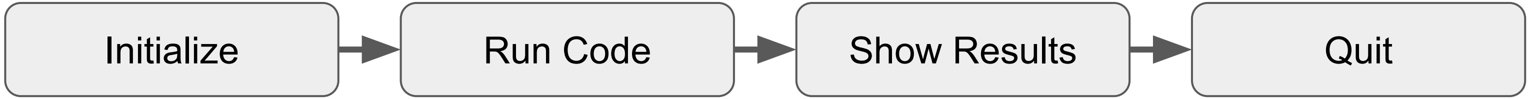
Event Loops

Non-interactive programs

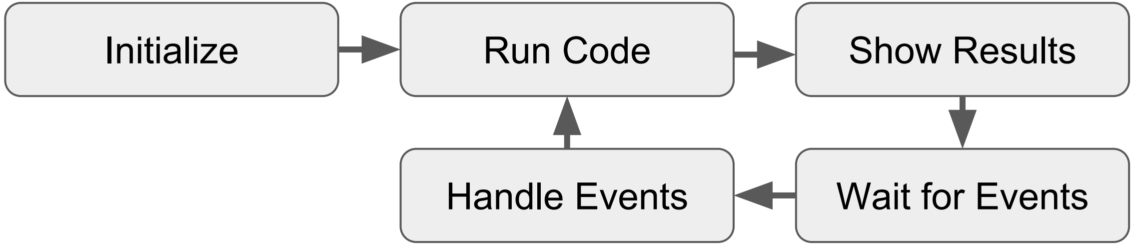


Event Loops

Non-interactive programs



Interactive programs:



So … who WAS that outlier?

Somewhat Generalizing [model-view-controller]

http://www.essenceandartifact.com/2012/12/the-essence-of-mvc.html

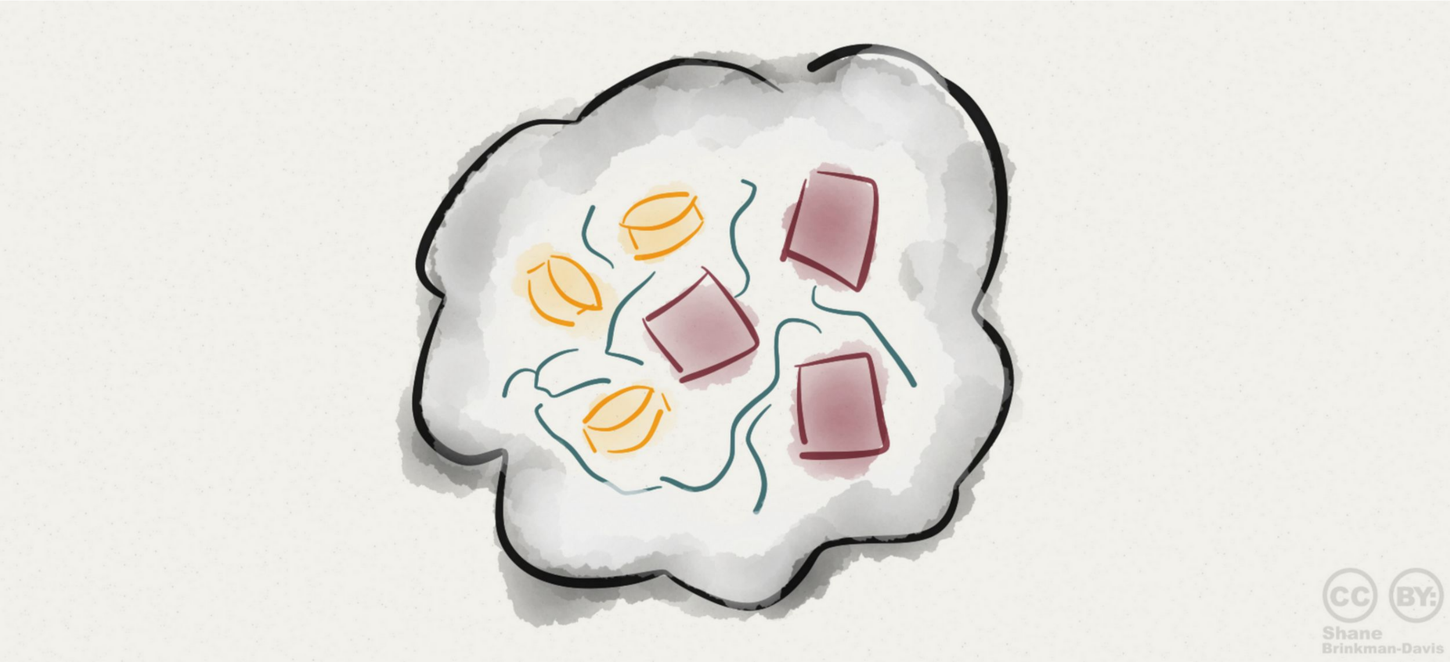
That can get messy...

Code that handles

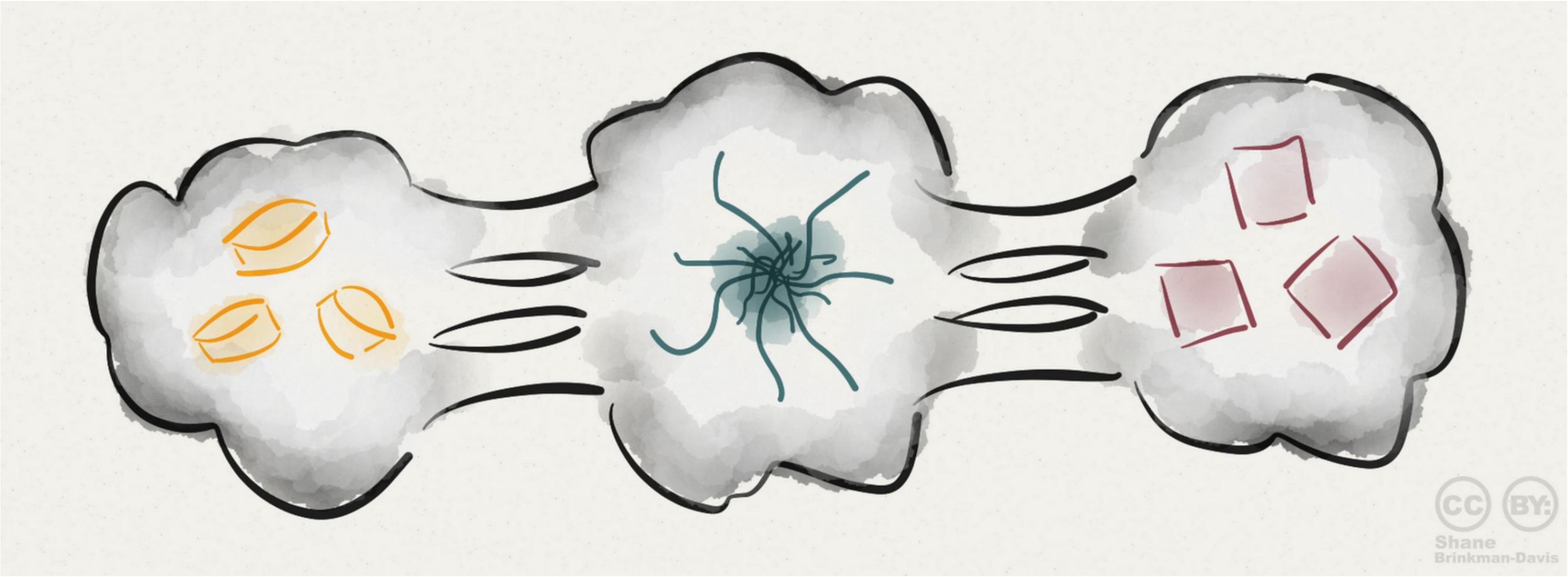
- Data modification
- Handling input events
- Displaying to the user, drawing
- Animating

All living together! Not so happily!

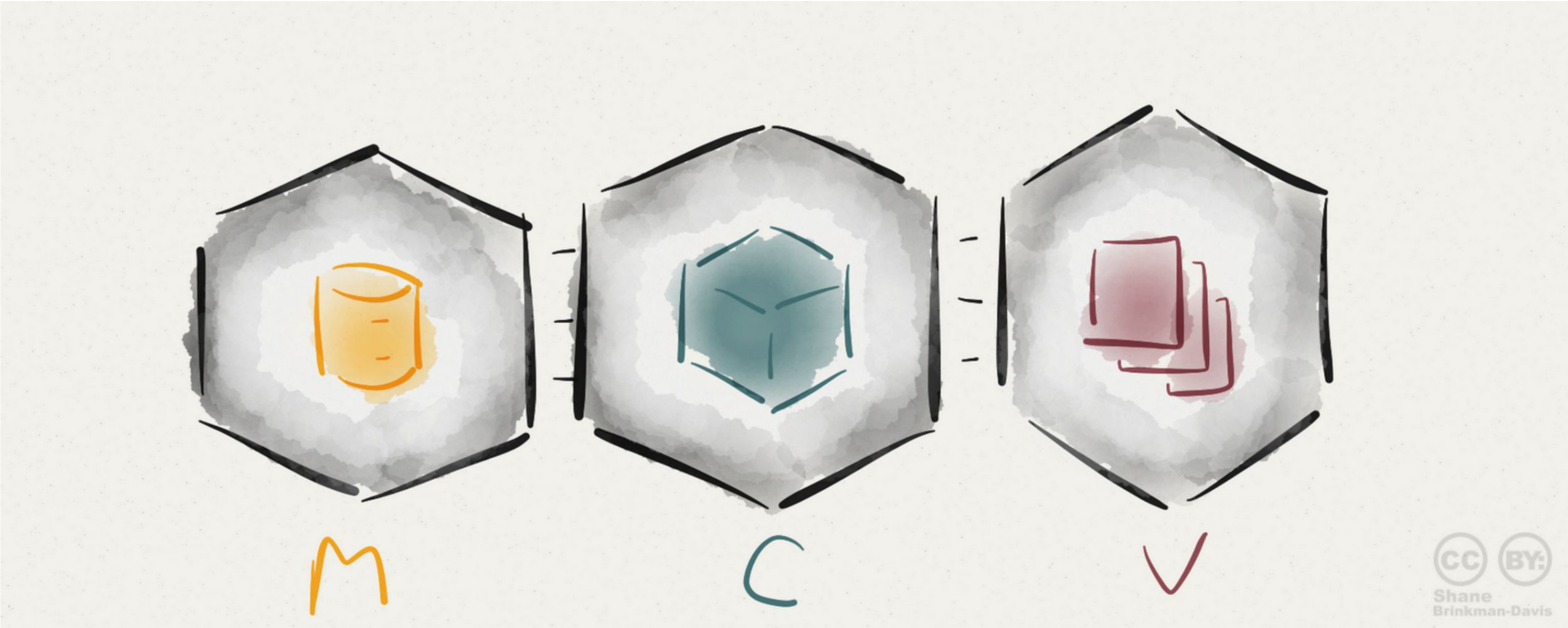
Model-View-Controller



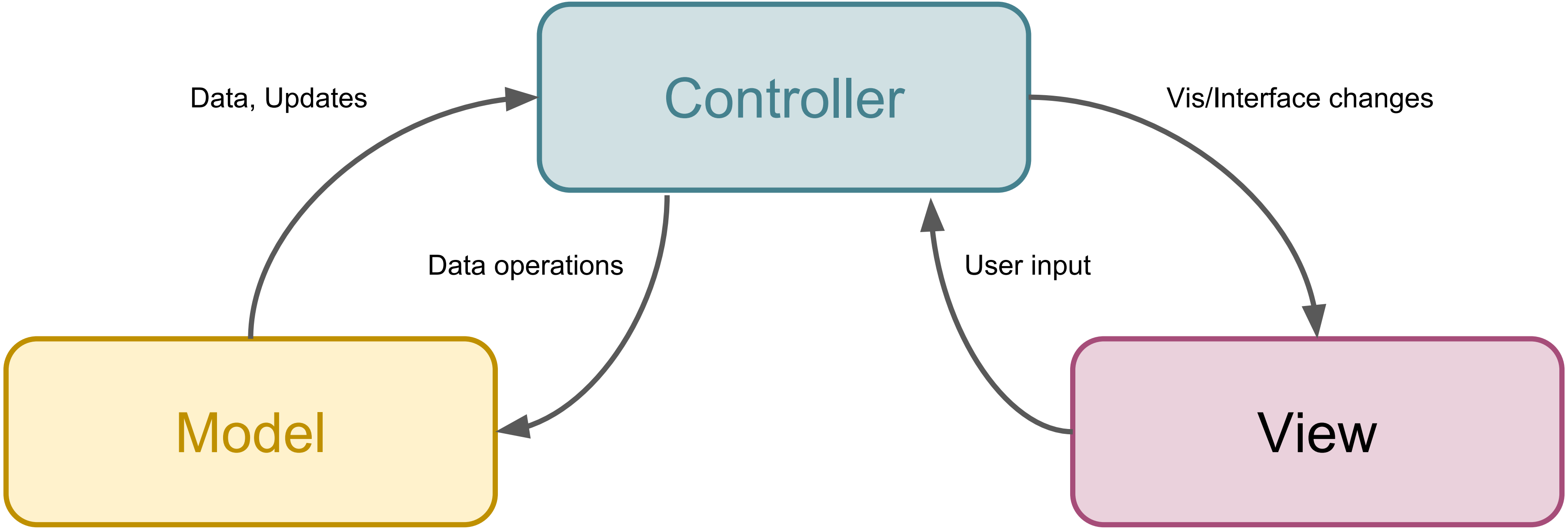
Model-View-Controller



Model-View-Controller



Model-View-Controller



Model-View-Controller

Model:

- store and modify data

View

- present visualization of data to the user, present user interface

Controller

- respond to user inputs and events, update the view and model

Interactivity Take-Aways

- Interactivity
  - Rapid evaluation can save time
  - Use MVC structure to keep code simple