
# Data visualization

Elizabeth R. Piette

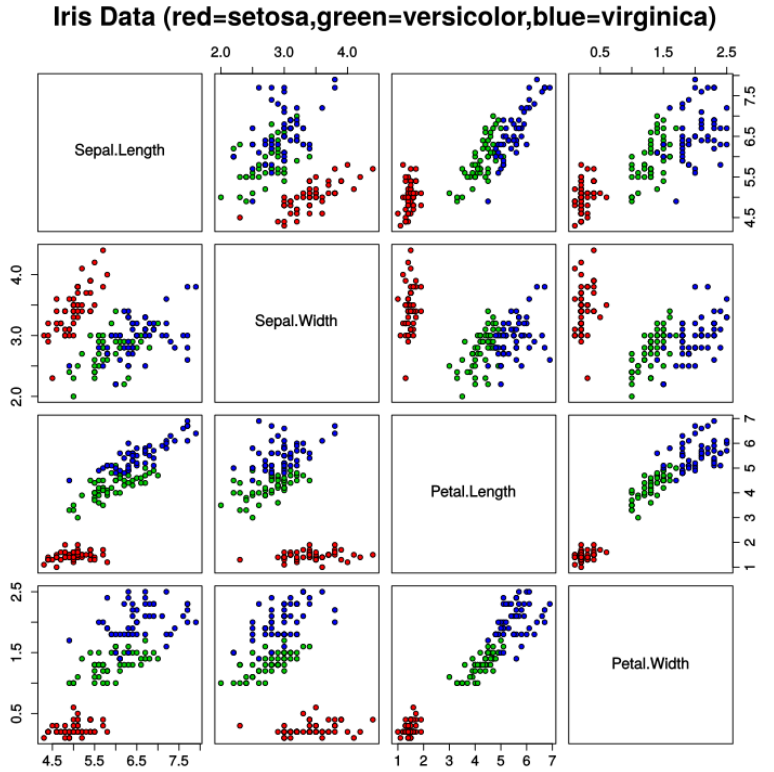
9/28/2017

# Why should we learn how to effectively visualize data?

- Quickly explore properties of a new data set

Fisher's *Iris* Data

| Sepal length | Sepal width | Petal length | Petal width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | *I. setosa* |
| 4.9 | 3.0 | 1.4 | 0.2 | *I. setosa* |
| 4.7 | 3.2 | 1.3 | 0.2 | *I. setosa* |
| 4.6 | 3.1 | 1.5 | 0.2 | *I. setosa* |
| 5.0 | 3.6 | 1.4 | 0.3 | *I. setosa* |
| 5.4 | 3.9 | 1.7 | 0.4 | *I. setosa* |
| 4.6 | 3.4 | 1.4 | 0.3 | *I. setosa* |
| 5.0 | 3.4 | 1.5 | 0.2 | *I. setosa* |
| 4.4 | 2.9 | 1.4 | 0.2 | *I. setosa* |
| 4.9 | 3.1 | 1.5 | 0.1 | *I. setosa* |
| 5.4 | 3.7 | 1.5 | 0.2 | *I. setosa* |
| 4.8 | 3.4 | 1.6 | 0.2 | *I. setosa* |
| 4.8 | 3.0 | 1.4 | 0.1 | *I. setosa* |
| 4.3 | 3.0 | 1.1 | 0.1 | *I. setosa* |
| 5.8 | 4.0 | 1.2 | 0.2 | *I. setosa* |
| 5.7 | 4.4 | 1.5 | 0.4 | *I. setosa* |
| 5.4 | 3.9 | 1.3 | 0.4 | *I. setosa* |
| 5.1 | 3.5 | 1.4 | 0.3 | *I. setosa* |

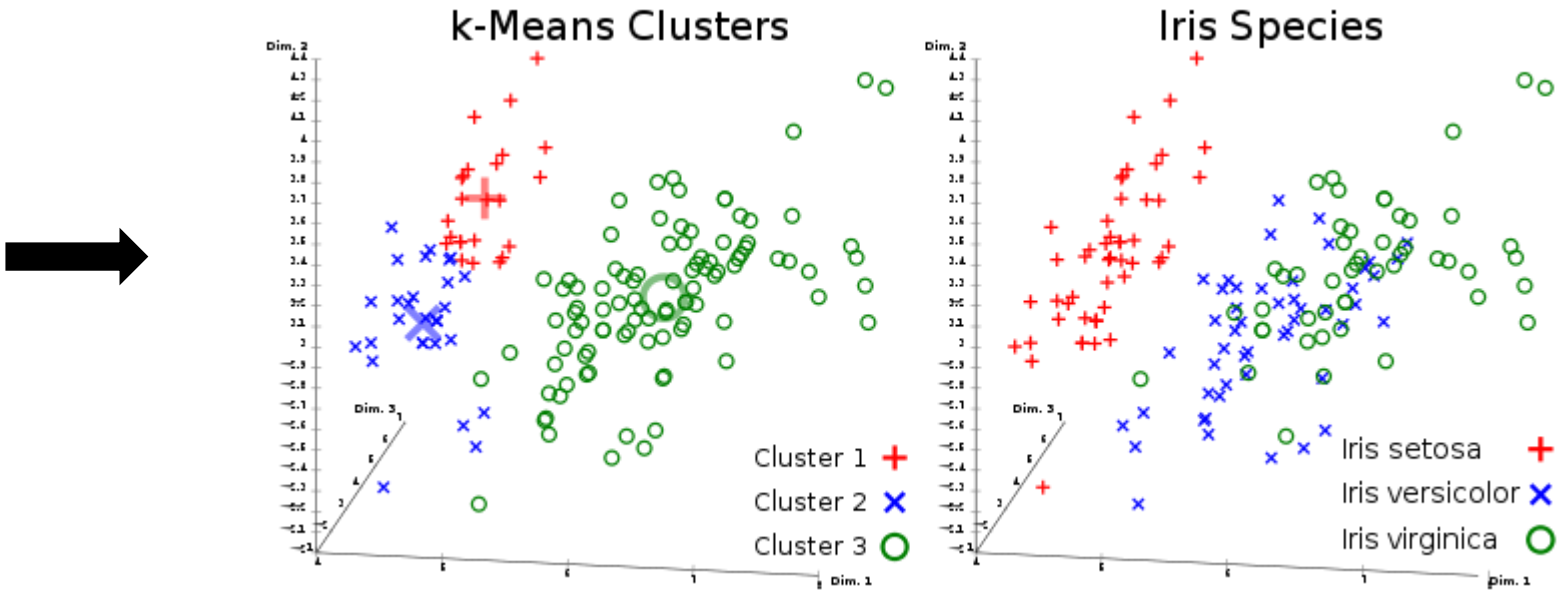



https://en.wikipedia.org/wiki/Iris_flower_data_set

# Why should we learn how to effectively visualize data?

- Quickly explore properties of a new data set
- Share our work in a meaningful and easy to understand way

Descriptive language: k-means clustering with 3 clusters does not satisfactorily separate instances into species clusters



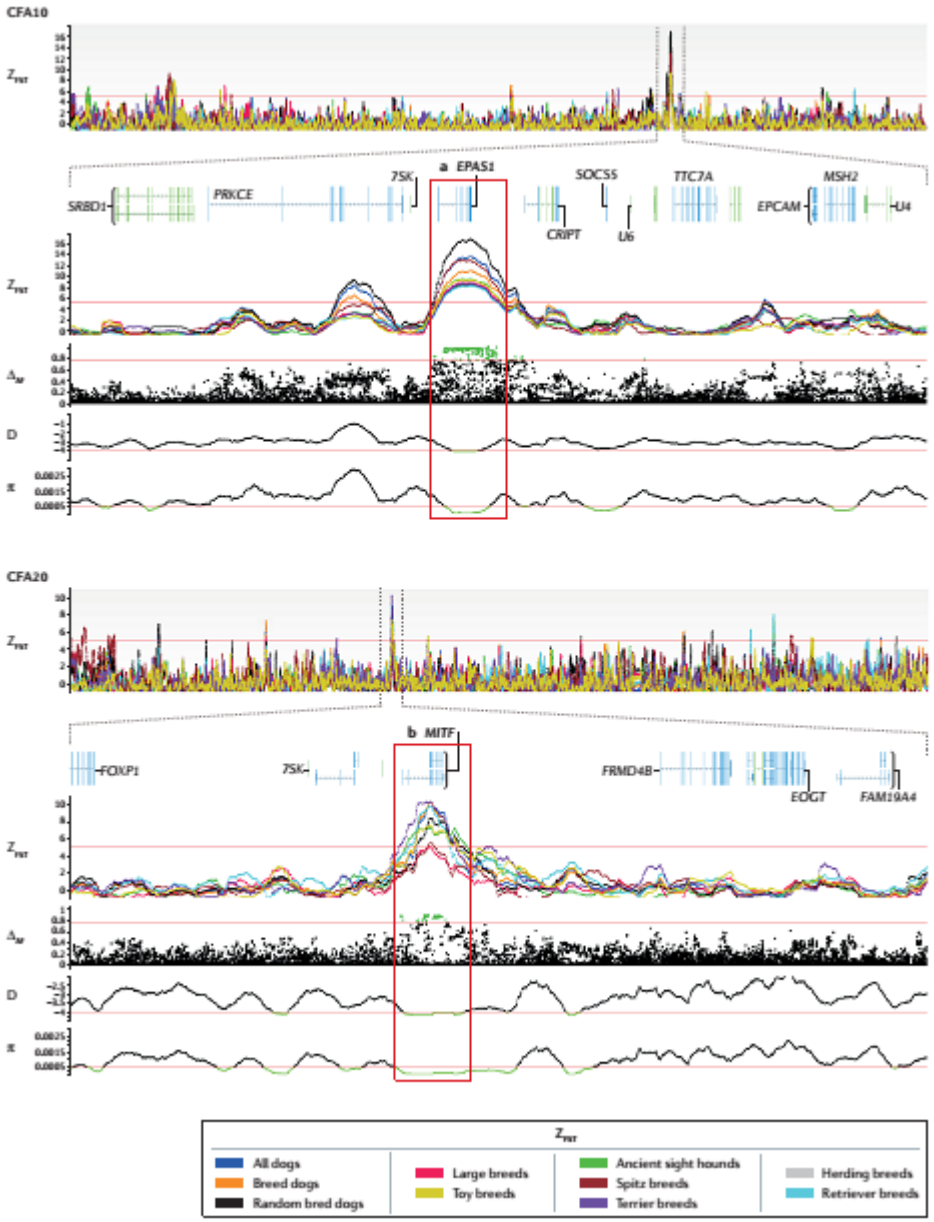

https://en.wikipedia.org/wiki/Iris_flower_data_set

# Learn by example

- Browse journals relevant to your field to familiarize yourself with common practices, such as the types of visualization generally paired with types of data
- Think critically about why specific visualizations are effective and why others fail to communicate the intended message

# Demographic history, selection and functional diversity of the canine genome

*Elaine A. Ostrander[1], Robert K. Wayne[2], Adam H. Freedman[3] and Brian W. Davis[1,4]*




◀ Figure 4 | **Selective sweeps from whole-genome sequencing analysis.** $Z_{FST}$ represents the outlier score for $F_{ST}$, indicating a region with high divergence from others. DeltaAF ($\Delta_{AF}$) is the shift in allele frequency in the population compared to all others, indicating private or enriched variation. Tajima's D (D) is a measure of the nonrandom segregation of alleles, with negative values indicative of selective or demographic influence in the region. Pi ($\pi$) represents nucleotide diversity, which is reduced in regions undergoing selection. **a** | The region surrounding *EPAS1*, a gene linked to hypoxic adaptation at high altitudes in Tibetan Mastiffs[174], on *Canis familiaris* chromosome 10 (CFA10). This gene has also been shown to be under selection in human populations and grey wolves living in the same environment. **b** | A sweep surrounding the *MITF* gene on CFA20 in herding breeds such as the Border Collie and Shetland Sheepdog that is linked to the degree of white colouration present in the coat[77].

*Elaine A. Ostrander[1], Robert K. Wayne[2], Adam H. Freedman[3] and Brian W. Davis[1,4]*

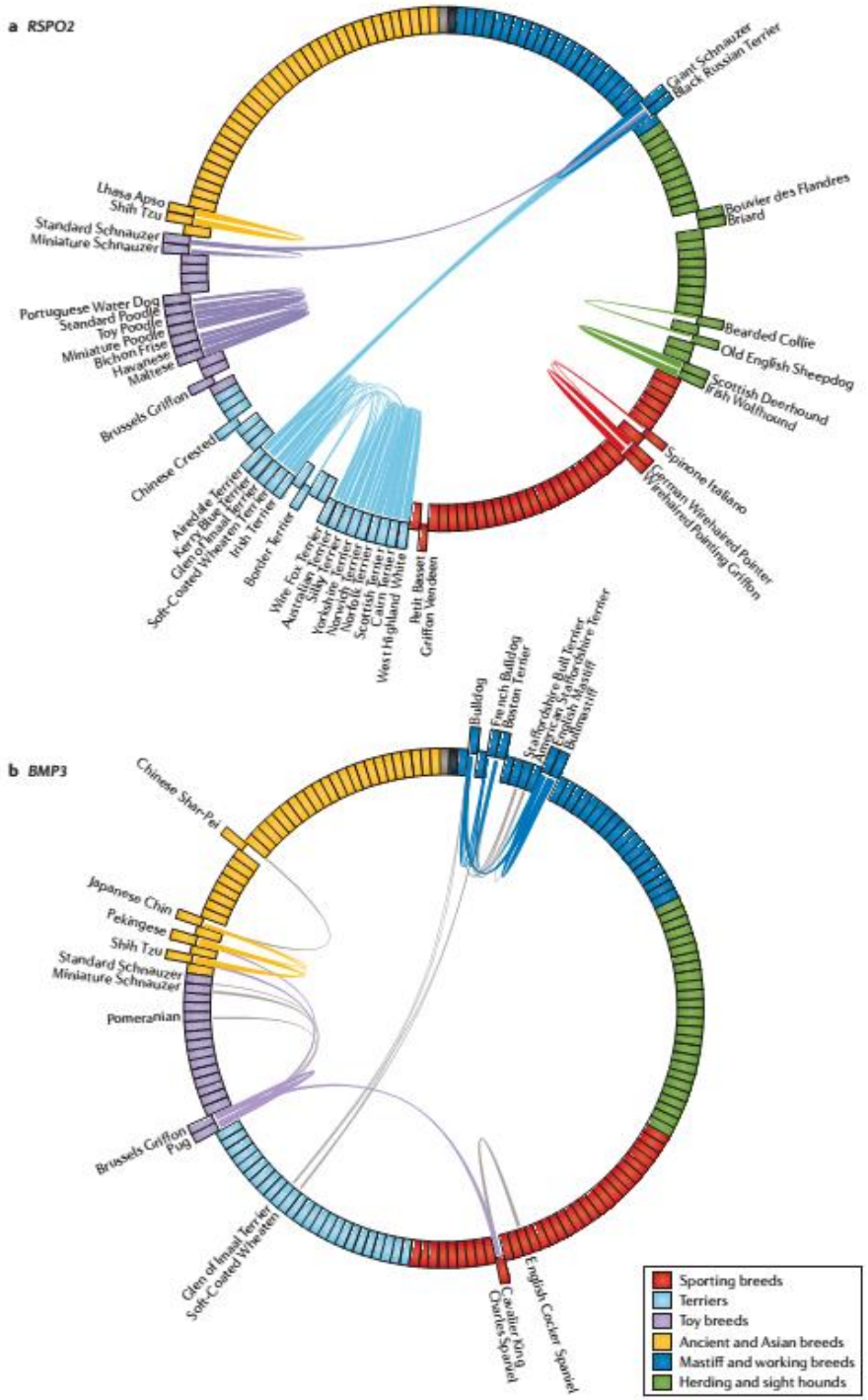



Figure 5 | **Haplotype sharing between breeds indicates the source of a common phenotype for a given trait.** Figures are derived from data published by Parker *et al.*[79]. **a** | An R-spondin 2 (*RSPO2*) mutation causes the 'furnishings' phenotype, typified by a 'moustache and eyebrows' that is observed in the Schnauzer (FIG. 2d) and in ~40 known dog breeds. Excessive haplotype sharing, indicated by the ribbons connecting breeds across the circle, show that the furnishings trait, which appears in the recently created Black Russian Terrier, came from either the Airedale Terrier or the Schnauzer. Breeds with the *RSPO2* mutation are set out from the main circle of breeds. The distribution of breeds with the phenotype suggests the mutation occurred early in the development of breeds. **b** | A nonsense mutation in bone morphogenetic protein 3 (*BMP3*) is associated with the brachycephalic phenotype in multiple breeds. The mutation is also carried by breeds with less severe head shapes. Breeds with the mutation and phenotype are set out from the circle. Coloured ribbons show extensive haplotype sharing between those breeds. Grey ribbons indicate haplotype sharing between brachycephalic breeds and breeds that do not have the phenotype. These breeds do not display the mutation, perhaps due to incomplete penetrance, and the mutation passes throughout the population.

# Genetic and epigenetic features direct differential efficiency of Xist-mediated silencing at X-chromosomal and autosomal locations

Agnese Loda[1,7], Johannes H. Brandsma[2], Ivaylo Vassilev[3], Nicolas Servant[3], Friedemann Loos[4], Azadeh Amirnasr[1], Erik Splinter[5], Emmanuel Barillot[3], Raymond A. Poot[2], Edith Heard[6] & Joost Gribnau[1]

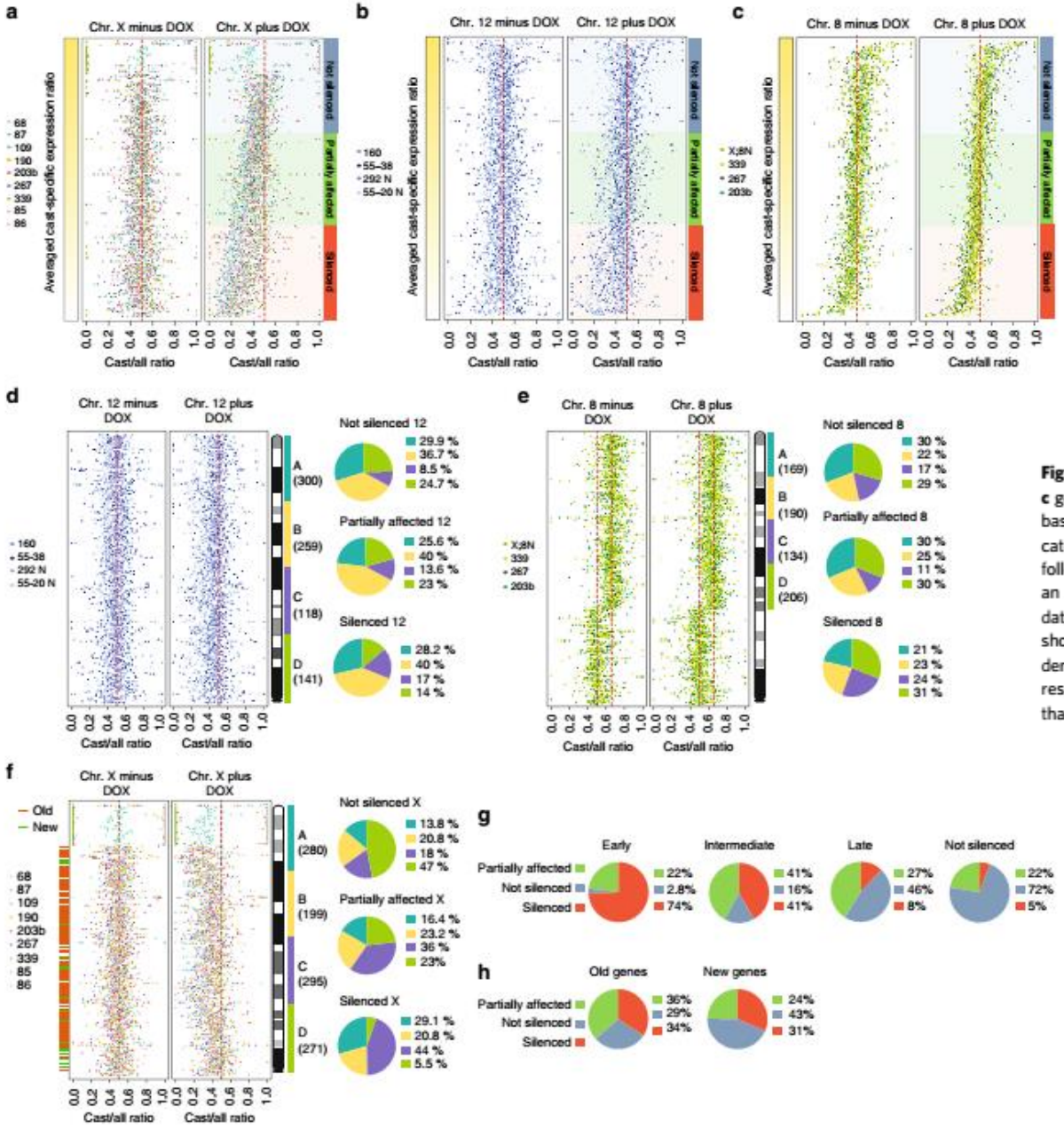



**Fig. 5** Preferential silencing of specific X-linked loci by ectopic Xist RNA. Gene silencing ranking plots for X-linked **a**, chromosome 12 **b**, and chromosome 8 **c** genes. Every *dot* represents the Cast/all expression ratio of a specific gene. About 242 genes are shown in **a**, 351 in **b**, and 336 in **c**. Genes are ranked based on the averaged Cast/all ratio among all clones in each group of clones (Tg-E, Tg-X, Tg-X;8, and Tg-12). Ranked genes are divided in three categories (I) efficiently silenced, (II) partially affected, and (III) not silenced genes. To simplify data visualization, the Cast/all ratios were transformed as follows: (1) For Tg-X;8 clones, carrying Xist transgenes on the 129/sv X chromosome, we used the reciprocal of the Cast/all ratio. Thus, all clones carrying an X-linked Xist transgene behave as if the transgene was integrated on the Cast/Ei X chromosome. (2) For Tg-12 and Tg-X;8 clones, we converted the data from trisomic to disomic by calculating a new Cast/all ratio = (NCast/2)/(NCast/2 + N129). **d–f** Genes are ordered by genomic position. Pie graphs show the amount of overlap between the gene categories denoted in **a–c** and chromosomal regions A, B, C, and D on chromosomes X, 12, and 8. Gene density along chromosomal regions A, B, C, and D is indicated in brackets. In **f**, *red* and *green lines* represent evolutionary old and new X-linked genes, respectively. **g**, **h** Pie graphs showing the proportion of efficiently silenced, partially affected, and not silenced X-linked genes overlapping with gene clusters that show different inactivation dynamics upon cell differentiation (**g**) and evolutionary old and new X-linked genes (**h**)

# Genetic and epigenetic features direct differential efficiency of Xist-mediated silencing at X-chromosomal and autosomal locations

Agnese Loda[1,7], Johannes H. Brandsma[2], Ivaylo Vassilev[3], Nicolas Servant[3], Friedemann Loos[4], Azadeh Amirnasr[1], Erik Splinter[5], Emmanuel Barillot[3], Raymond A. Poot[2], Edith Heard[6] & Joost Gribnau[1]

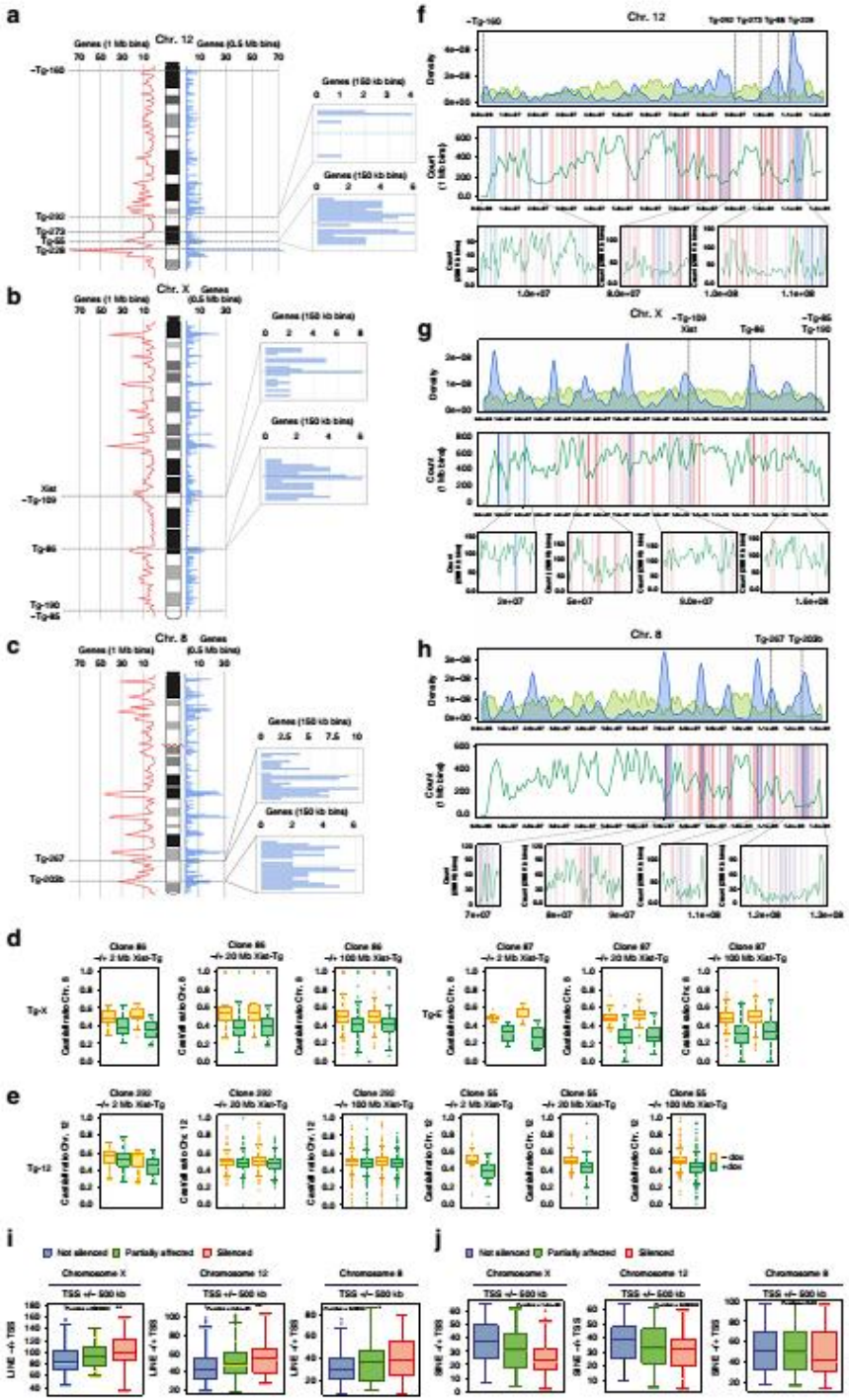


**Fig. 6** LINE and gene density on chromosomes X, 8, and 12. **a**–**c** Gene density along chromosomes 12 **a**, X **b**, and 8 **c**. *Blue histogram bars* represent 0.5 Mb bins, *red frequency lines* correspond to gene distribution in 1 Mb bins. The integration sites of Xist transgenes are indicated and zoom in of the integration loci is shown. *Blue histogram bars* represent 150 kb bins. **d**, **e** *Box plots* showing the Cast/all ratios of X-linked **d** and chromosome 12 **e** genes in 2, 20, and 100 Mb bins around the Xist transgene integration sites as determined by allele-specific RNA-seq analysis. **f**–**h** *Top*: LINE density relative to gene density on chromosome 12 **f**, X **g**, and 8 **h**. *Blue*, gene density; *green*, LINE density. *Middle*: LINE distribution along chromosome 12 **f**, X **g**, and 8 **h**, *green frequency lines* correspond to LINE distribution in 1 Mb bins along the chromosomes. *Blue* and *red lines* indicate "not silenced" and "efficiently silenced" genes defined in Fig. 5, respectively. *Bottom*: zoom in of specific loci. *Green frequency lines* correspond to LINE distribution in 200 kb bins. **i** Box plot showing LINE density in 1 Mb bins around the TSS of genes belonging to the three different categories defined in Fig. 5. Data for chromosomes 12, 8, and X are shown. $^{*}p < 0.05$ and $^{**}p < 0.005$ Wilcoxon rank-sum test. **j** as in **i** but for SINE elements

# ZNF516 suppresses *EGFR* by targeting the CtBP/LSD1/CoREST complex to chromatin

Lifang Li[1], Xinhua Liu[2], Lin He[1], Jianguo Yang[1], Fei Pei[3], Wanjin Li[1], Shumeng Liu[1], Zhe Chen[1], Guojia Xie[1], Bosen Xu[1], Xia Ting[1], Zihan Zhang[1], Tong Jin[1], Xujun Liu[1], Wenting Zhang[1], Shuai Yuan[1], Ziran Yang[1], Chongyang Wu[1], Yu Zhang[1], Xiaohan Yang[1], Xia Yi[1], Jing Liang[1], Yongfeng Shang[1,2] & Luyang Sun[1]

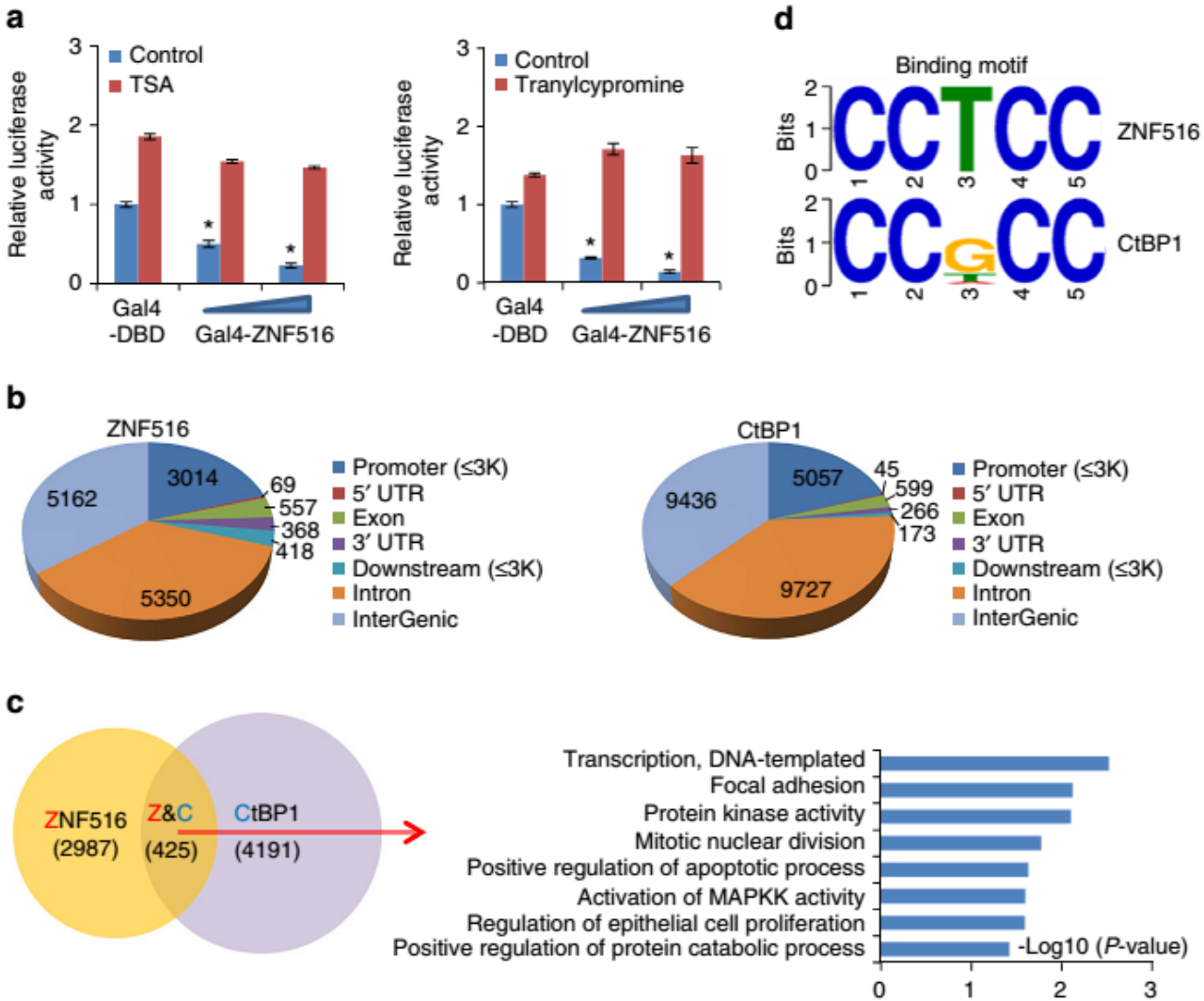
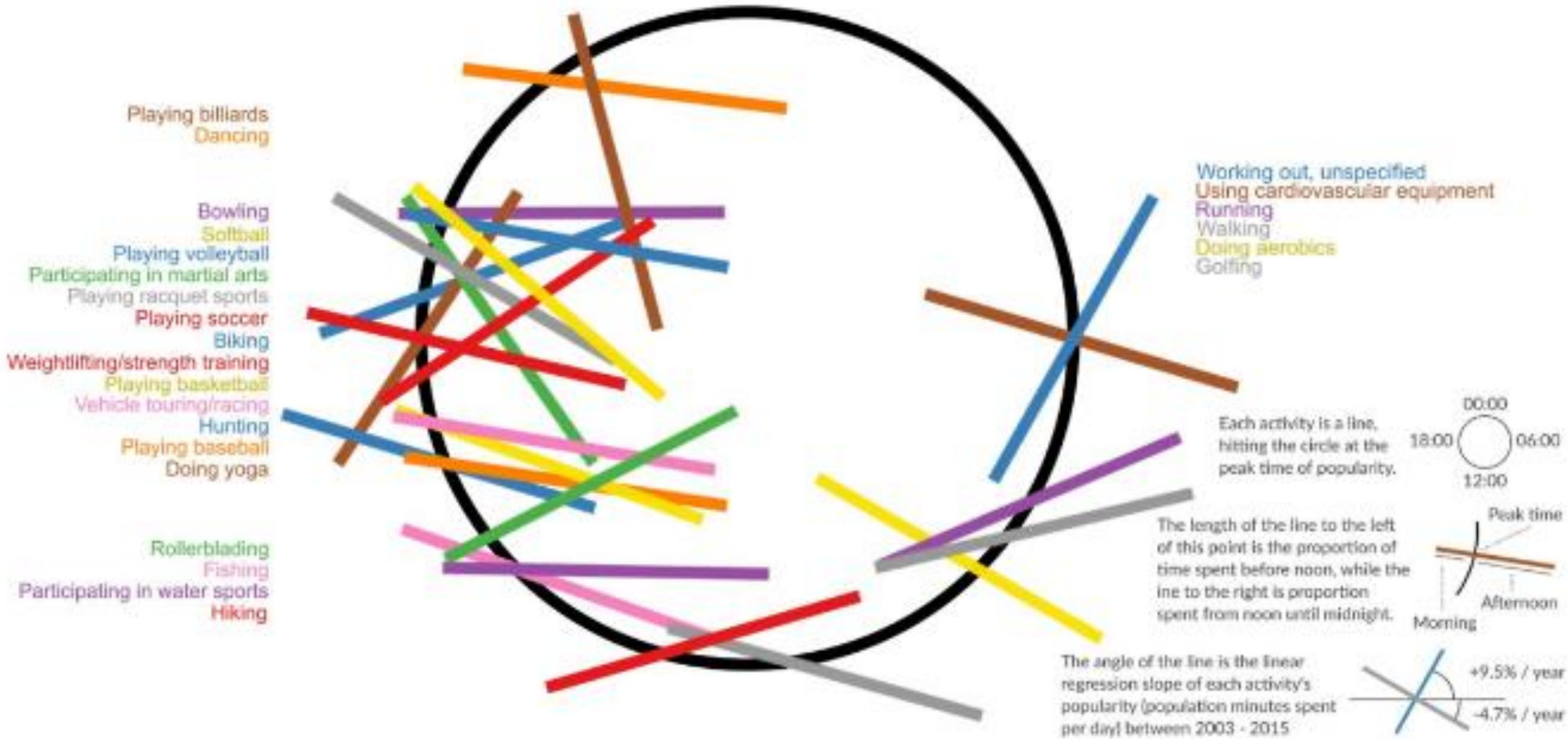



**Fig. 4** Identification of transcriptional targets for ZNF516. **a** MCF-7 cells were transfected with Gal4-DBD or Gal4-ZNF516 plasmids and were treated with trichostatin A (*TSA*) or tranylcypromine. Forty-eight hours later, luciferase activity of Gal4-SV40-LUC was measured. Relative luciferase activity was calculated as firefly luciferase activity divided by renilla luciferase activity and shown relative to the control. Each *bar* represents the mean ± S.D. for triplicate experiments. *P*-values were determined by Student's *t*-test. $^*P < 0.05$. **b** Genomic distribution of ZNF516 and CtBP1 (published in ref. [46]) determined by ChIP-seq analysis in MCF-7 cells. Number of *peaks* in each cluster was indicated. **c** The Venn diagram of overlapping genes targeted by ZNF516 and CtBP1 in MCF-7 cells (*left*). The clustering of the 425 overlapping target genes of ZNF516/CtBP1 into biological process ontologies is shown (*right*). The details of the ChIP-seq experiments were provided in the Materials and Methods and results on the ontologies were provided in Supplementary Data 2. **d** The analysis of ZNF516-bound motifs and CtBP1-bound motifs using MEME suite. Top 10 motifs enriched at ZNF516 peaks were listed in Supplementary Fig. 2

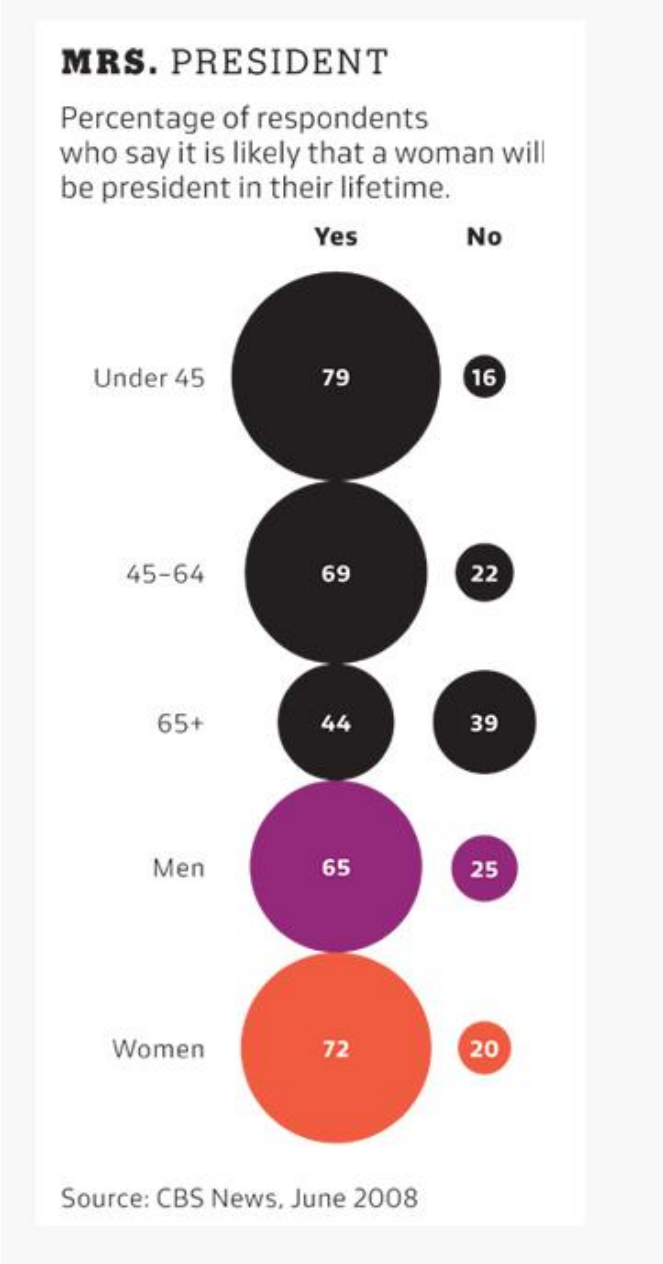
http://viz.wtf/

# Peak time for sports and leisure

@hnrklndbrg | Source: American Time Use Survey



https://eagereyes.org/blog/2008/ny-times-the-best-and-worst-of-data-visualization



https://eagereyes.org/blog/2008/ny-times-the-best-and-worst-of-data-visualization

# Some pointers to keep in mind

- Consider what type of visualization best suits your data and the message you're trying to convey with it
- Keep it simple – avoid needless complexity
- Appropriately direct viewers to important information using color, patterns, line weight, etc.
- Contextualize with a meaningful title, axes, and labels
- Tailor your visualization to the audience

# Let’s get started!

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

# Let’s get started!

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

We’ll be using the popular ‘pylot’ from matplotlib for this lecture

Python ‘magic’ that lets you view plots in your Jupyter notebook

# Basic plotting: plot cos(x)

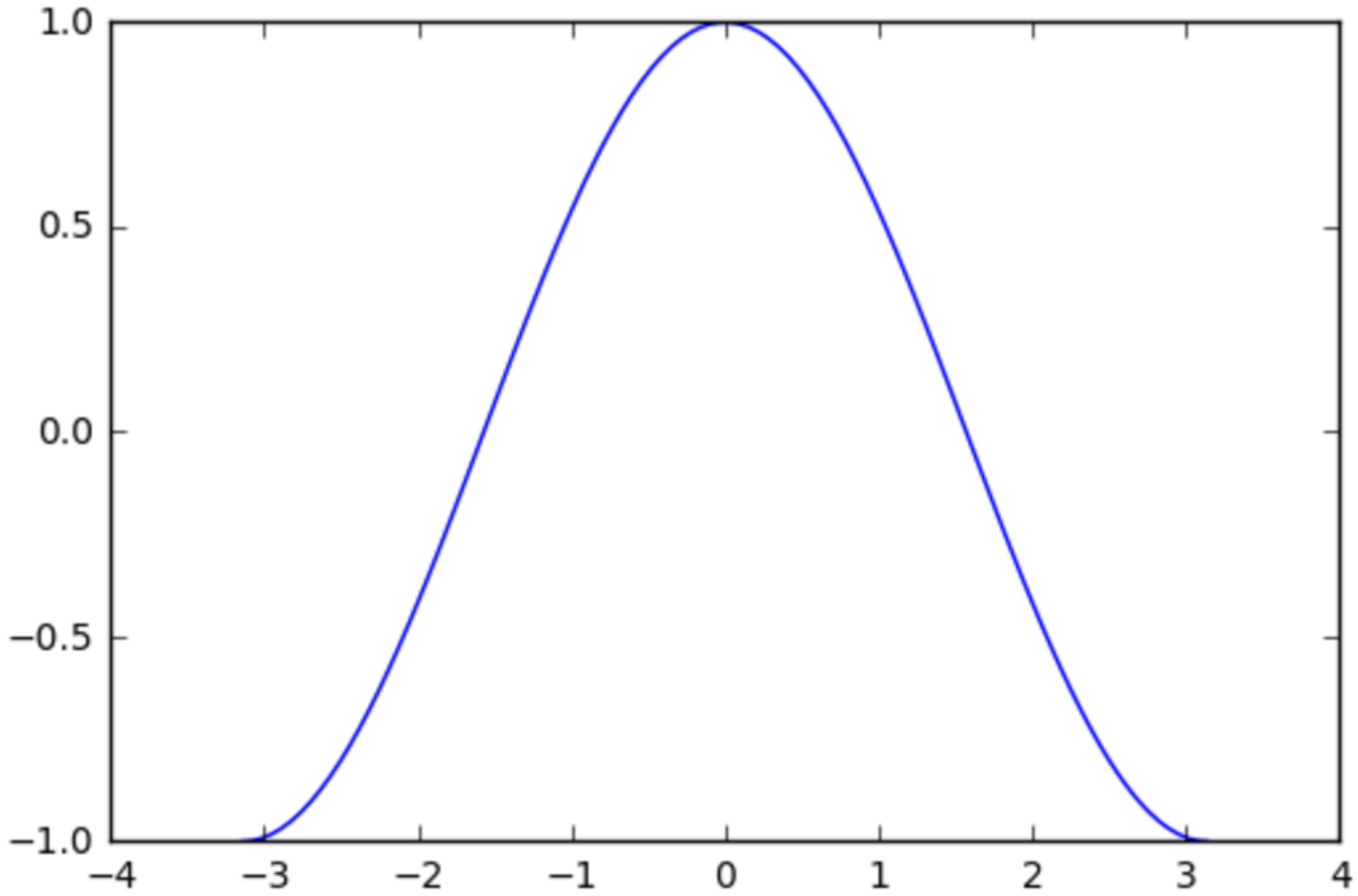
```
x = np.linspace(-np.pi, np.pi, 100, endpoint=True)

cos_x = np.cos(x)

plt.plot(x, cos_x)

plt.show()
```

# Basic plotting: plot cos(x)

```
x = np.linspace(-np.pi, np.pi, 100, endpoint=True)

cos_x = np.cos(x)

plt.plot(x, cos_x)

plt.show()
```

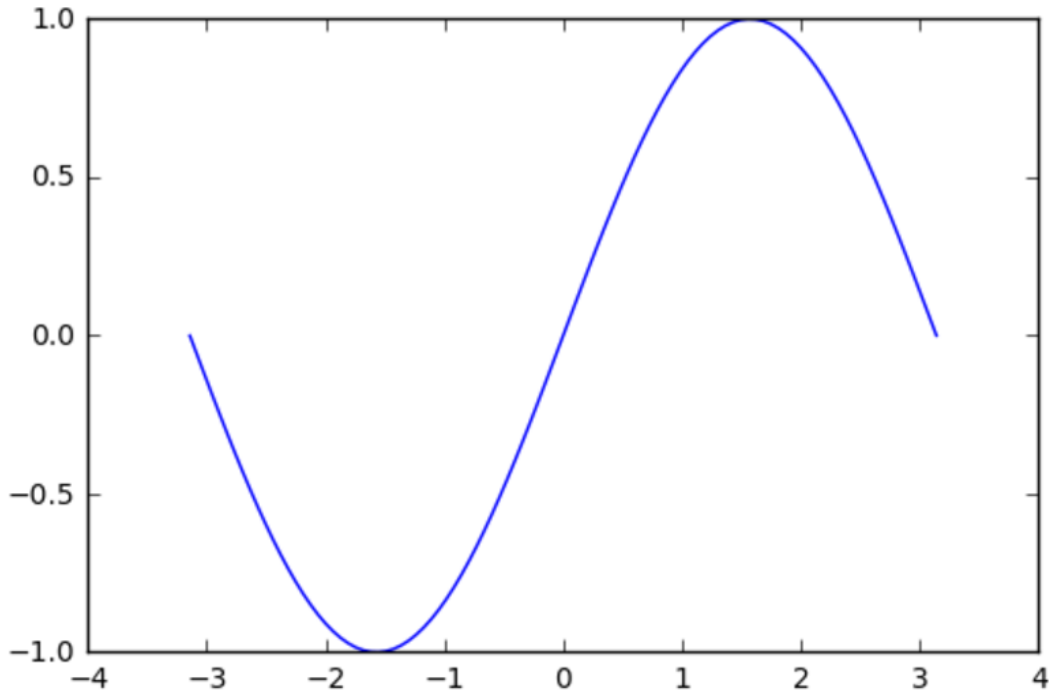
Generates 100 evenly spaced points from –pi to pi

# Your output should look like:

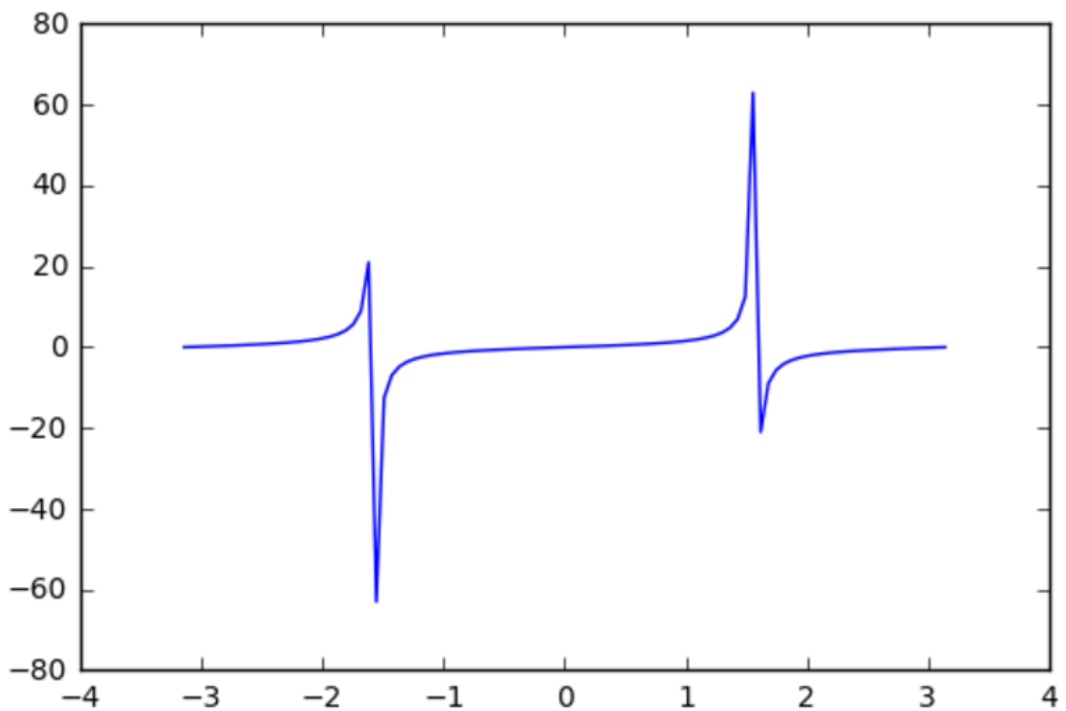


# Now try plotting sin(x) and tan(x)

```
sin_x = np.sin(x)

plt.plot(x, sin_x)

plt.show()
```

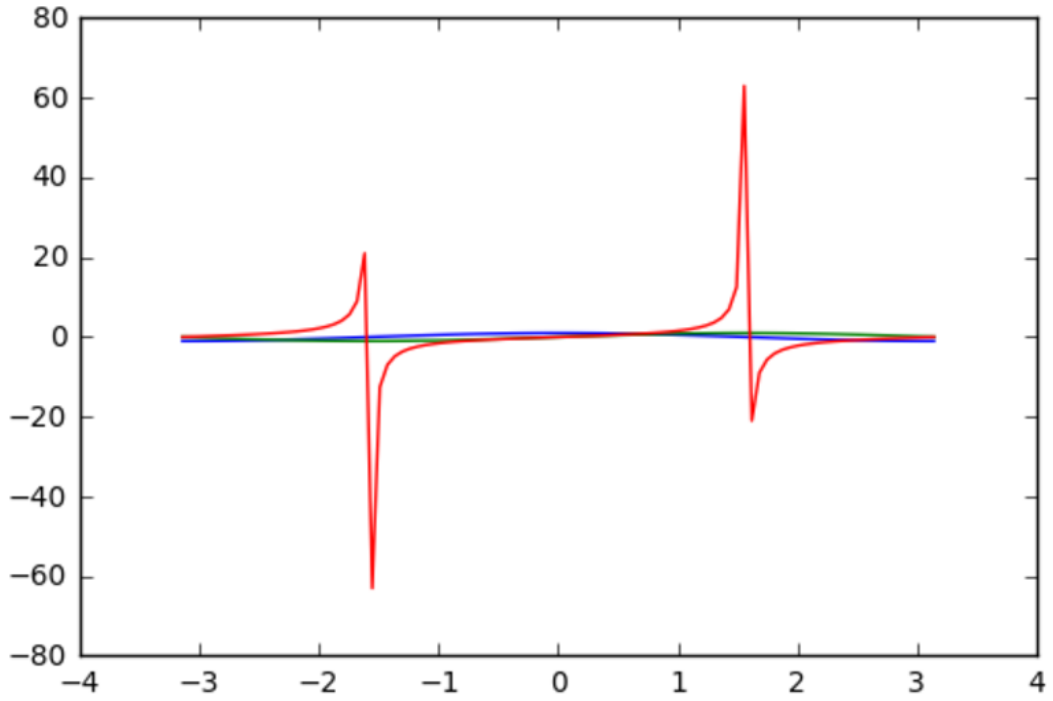



```
tan_x = np.tan(x)

plt.plot(x, tan_x)

plt.show()
```



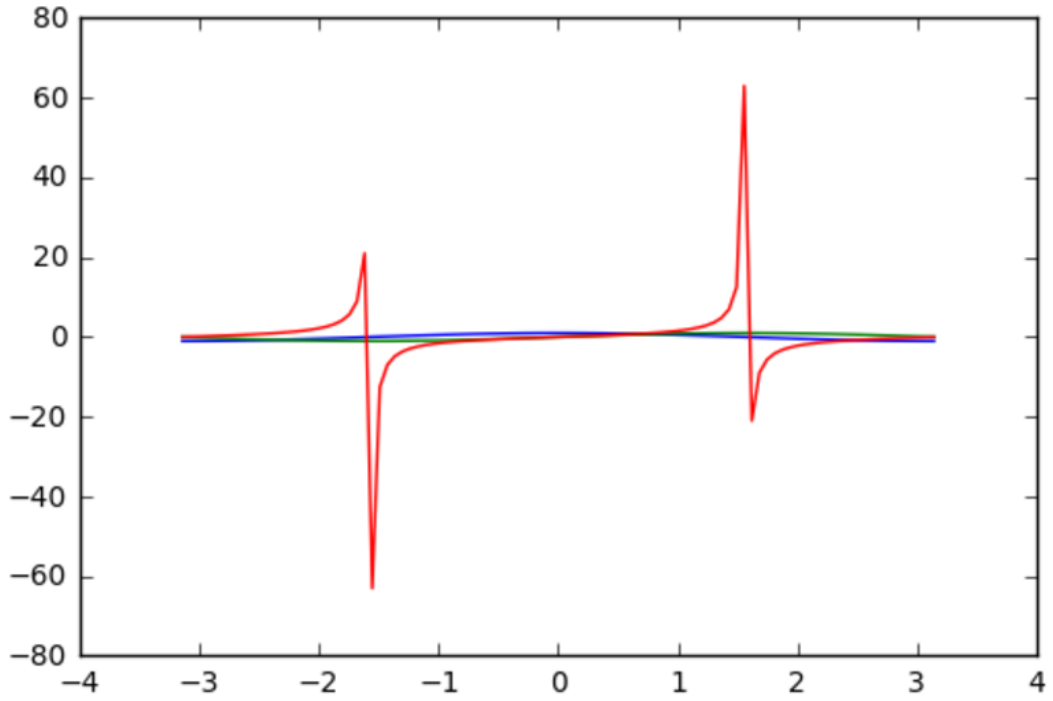
# Issuing multiple plot commands

```
plt.plot(x, cos_x)
plt.plot(x, sin_x)
plt.plot(x, tan_x)

plt.show()
```



# Issuing multiple plot commands

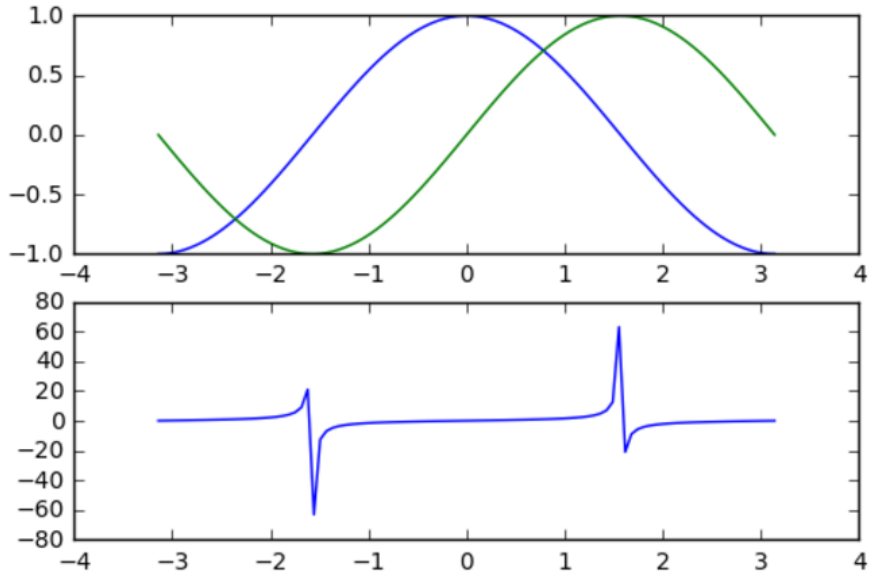
```
plt.plot(x, cos_x)
plt.plot(x, sin_x)
plt.plot(x, tan_x)

plt.show()
```




Perhaps we don’t want to overlay these three functions because it’s difficult to see sin(x) and cos(x) when plotted with the same y-axis as tan(x)…

# Creating a new figure with subplots

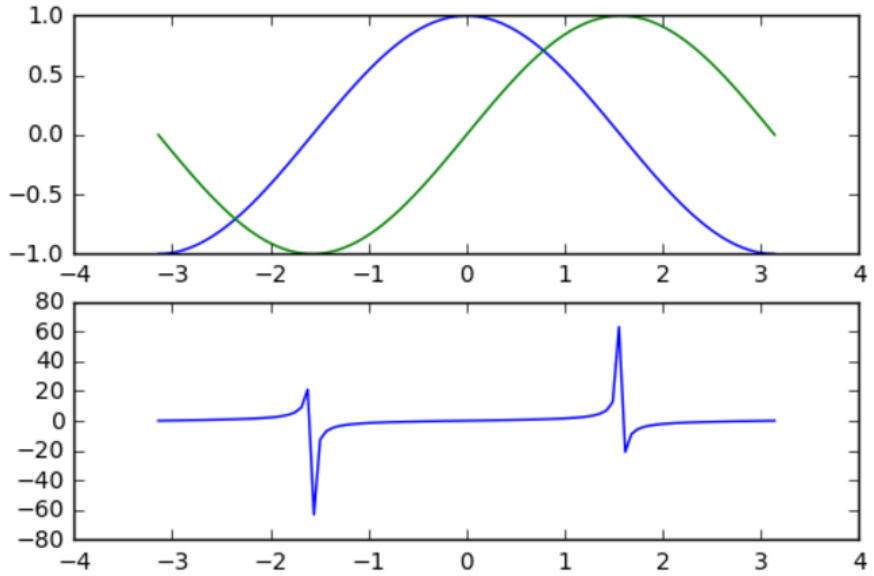
```
figure_1 = plt.figure()

facet_1 = figure_1.add_subplot(2, 1, 1)

plt.plot(x, cos_x)
plt.plot(x, sin_x)

facet_2 = figure_1.add_subplot(2, 1, 2)

plt.plot(x, tan_x)

plt.show()
```



# Creating a new figure with subplots

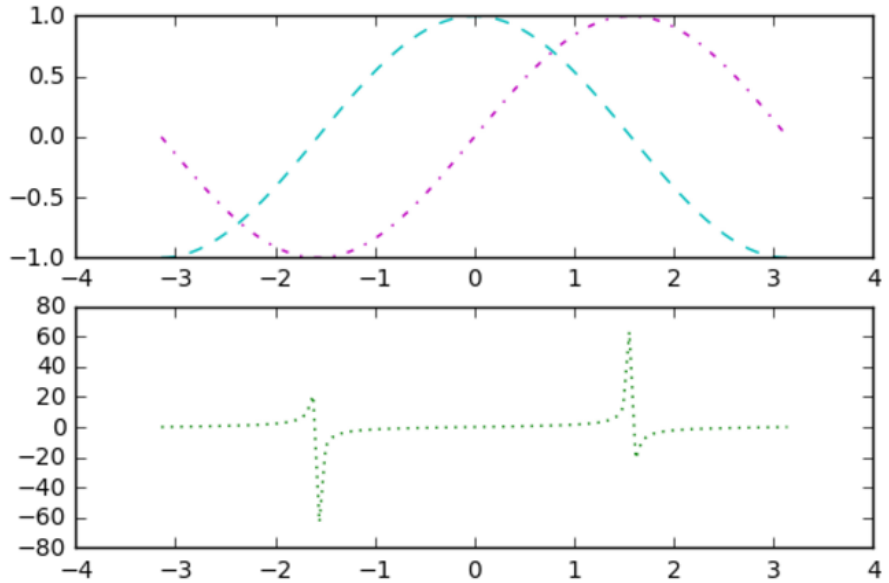
```
figure_1 = plt.figure()

facet_1 = figure_1.add_subplot(2, 1, 1)

plt.plot(x, cos_x)
plt.plot(x, sin_x)

facet_2 = figure_1.add_subplot(2, 1, 2)

plt.plot(x, tan_x)

plt.show()
```

Specify # rows, # cols, and subplot #



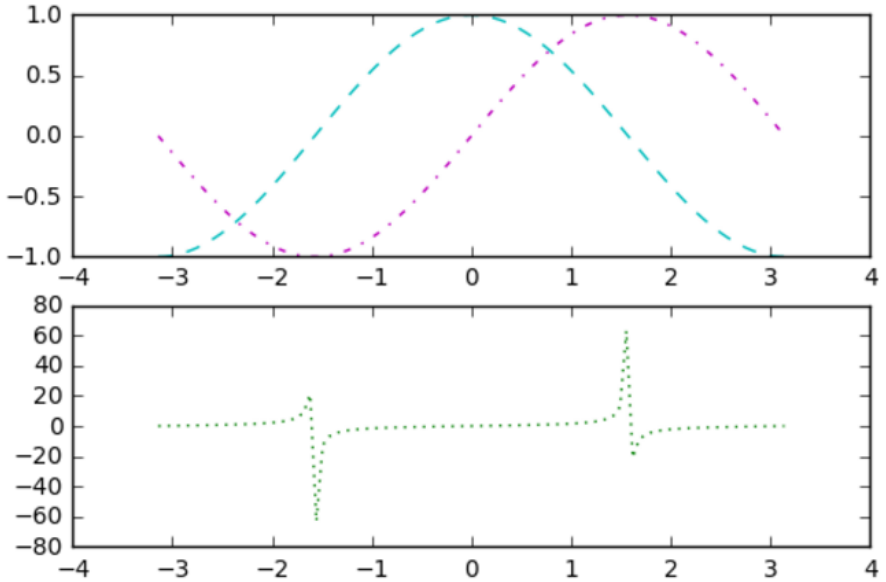
# Changing color and line style

```
figure_1 = plt.figure()

facet_1 = figure_1.add_subplot(2, 1, 1)

plt.plot(x, cos_x, color = "c", linestyle = "--")
plt.plot(x, sin_x, color = "m", linestyle = "-.")

facet_2 = figure_1.add_subplot(2, 1, 2)

plt.plot(x, tan_x, color = "g", linestyle = ":")

plt.show()
```



# Changing color and line style

```
figure_1 = plt.figure()

facet_1 = figure_1.add_subplot(2, 1, 1)

plt.plot(x, cos_x, color = "c", linestyle = "--")
plt.plot(x, sin_x, color = "m", linestyle = "-.")

facet_2 = figure_1.add_subplot(2, 1, 2)

plt.plot(x, tan_x, color = "g", linestyle = ":")

plt.show()
```

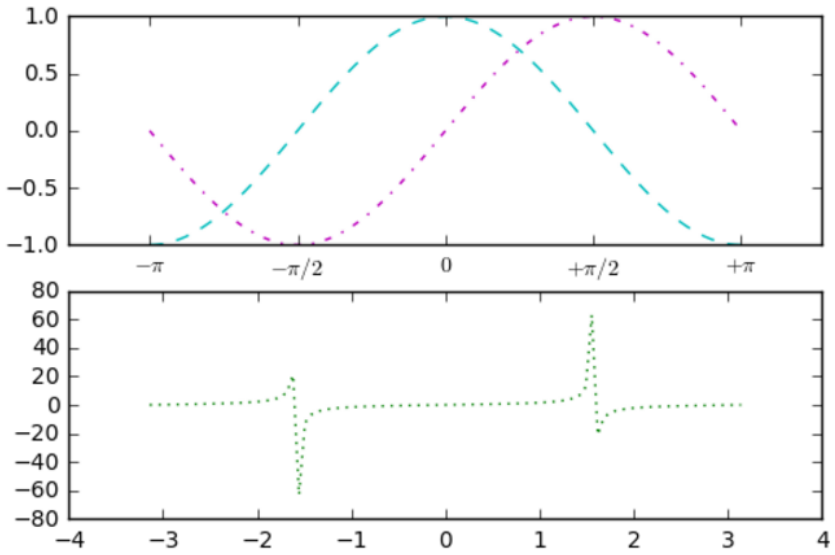



There are some built-in colors you can refer to using a single letter.
You can also use hex, refer to grays as a floats from 0-1, convert to rbg, etc.

Similarly, there are numerous line styles and markers built in

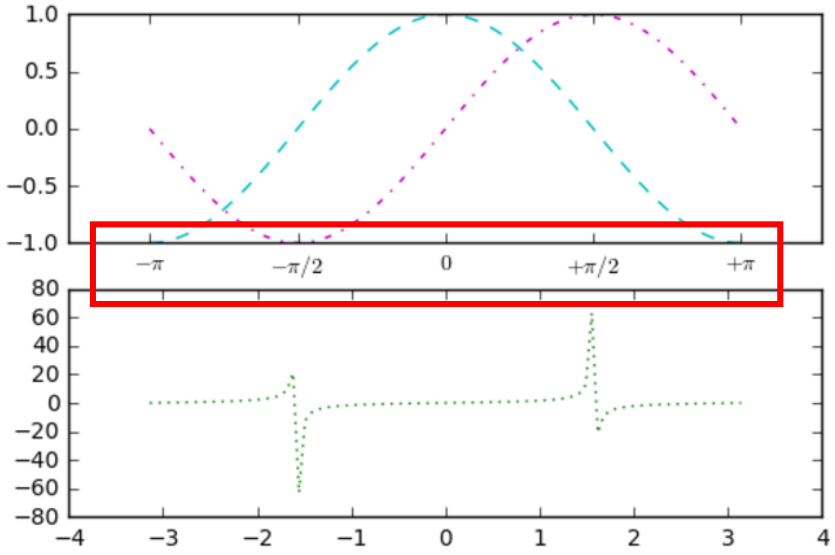
# Axes and labels

```
figure_1 = plt.figure()

facet_1 = figure_1.add_subplot(2, 1, 1)

plt.plot(x, cos_x, color = "c", linestyle = "--")
plt.plot(x, sin_x, color = "m", linestyle = "-.")

facet_1_ticks = facet_1.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_1_labels = facet_1.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

facet_2 = figure_1.add_subplot(2, 1, 2)

plt.plot(x, tan_x, color = "g", linestyle = ":")

plt.show()
```



# Axes and labels

```
figure_1 = plt.figure()

facet_1 = figure_1.add_subplot(2, 1, 1)

plt.plot(x, cos_x, color = "c", linestyle = "--")
plt.plot(x, sin_x, color = "m", linestyle = "-.")

facet_1_ticks = facet_1.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_1_labels = facet_1.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

facet_2 = figure_1.add_subplot(2, 1, 2)

plt.plot(x, tan_x, color = "g", linestyle = ":")

plt.show()
```

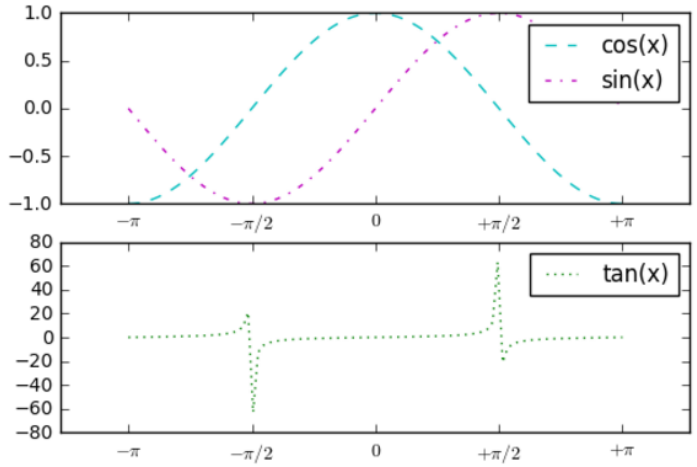



Try adding ticks and labels to the lower subplot.

Try the y-axes as well!

# Legends

```
figure_1 = plt.figure()

facet_1 = figure_1.add_subplot(2, 1, 1)

plt.plot(x, cos_x, color = "c", linestyle = "--", label = "cos(x)")
plt.plot(x, sin_x, color = "m", linestyle = "-.", label = "sin(x)")

facet_1.legend(loc="upper right")

facet_1_ticks = facet_1.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_1_labels = facet_1.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

facet_2 = figure_1.add_subplot(2, 1, 2)

plt.plot(x, tan_x, color = "g", linestyle = ":", label = "tan(x)")

facet_2.legend(loc="upper right")

facet_2_ticks = facet_2.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_2_labels = facet_2.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

plt.show()
```

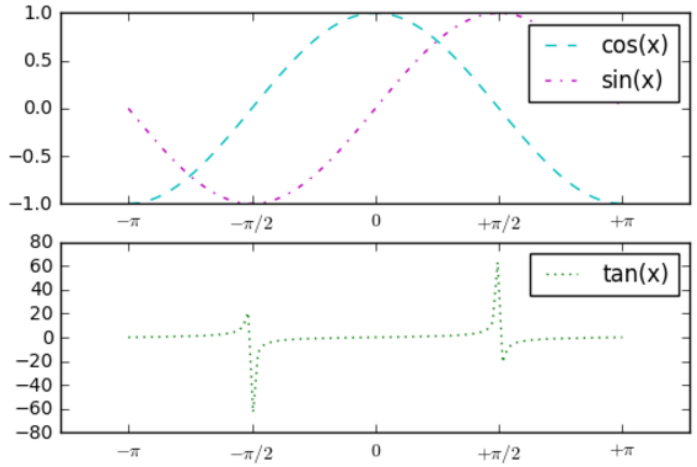


# Legends

```
figure_1 = plt.figure()

facet_1 = figure_1.add_subplot(2, 1, 1)

plt.plot(x, cos_x, color = "c", linestyle = "--", label = "cos(x)")
plt.plot(x, sin_x, color = "m", linestyle = "-.", label = "sin(x)")

facet_1.legend(loc="upper right")

facet_1_ticks = facet_1.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_1_labels = facet_1.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

facet_2 = figure_1.add_subplot(2, 1, 2)
plt.plot(x, tan_x, color = "g", linestyle = ":", label = "tan(x)")

facet_2.legend(loc="upper right")

facet_2_ticks = facet_2.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_2_labels = facet_2.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

plt.show()
```

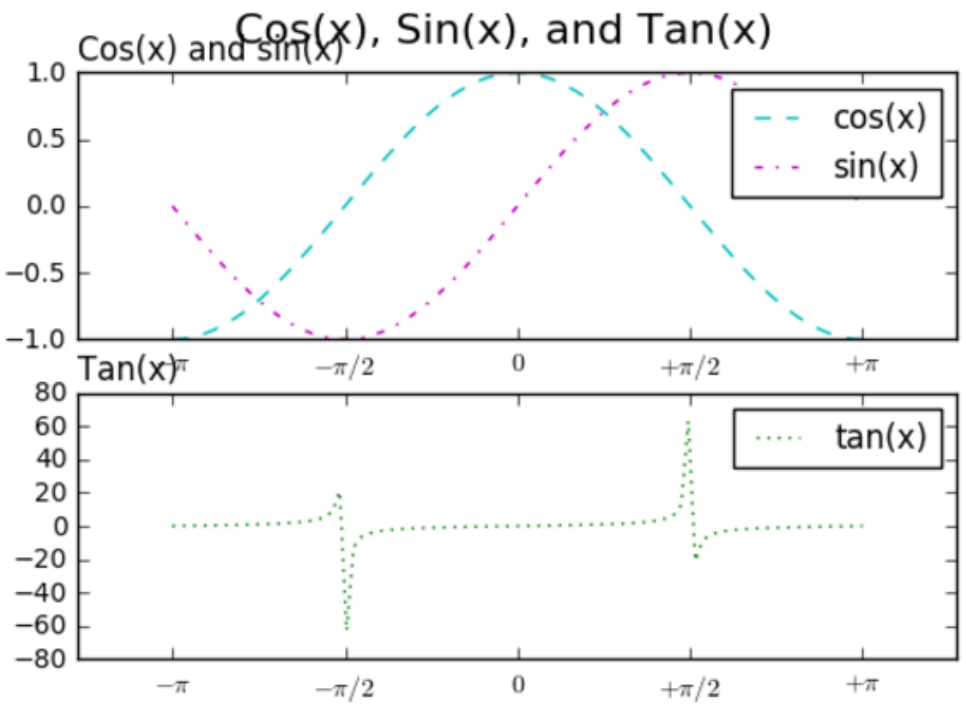
Label text

Legend location



# Titles

```
figure_1 = plt.figure()

figure_1.suptitle("Cos(x), Sin(x), and Tan(x)", fontsize=16)

facet_1 = figure_1.add_subplot(2, 1, 1)

facet_1.set_title("Cos(x) and sin(x)", loc="left")

plt.plot(x, cos_x, color = "c", linestyle = "--", label = "cos(x)")
plt.plot(x, sin_x, color = "m", linestyle = "-.", label = "sin(x)")

facet_1.legend(loc="upper right")

facet_1_ticks = facet_1.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_1_labels = facet_1.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

facet_2 = figure_1.add_subplot(2, 1, 2)

facet_2.set_title("Tan(x)", loc="left")

plt.plot(x, tan_x, color = "g", linestyle = ":", label = "tan(x)")

facet_2.legend(loc="upper right")

facet_2_ticks = facet_2.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_2_labels = facet_2.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

plt.show()
```



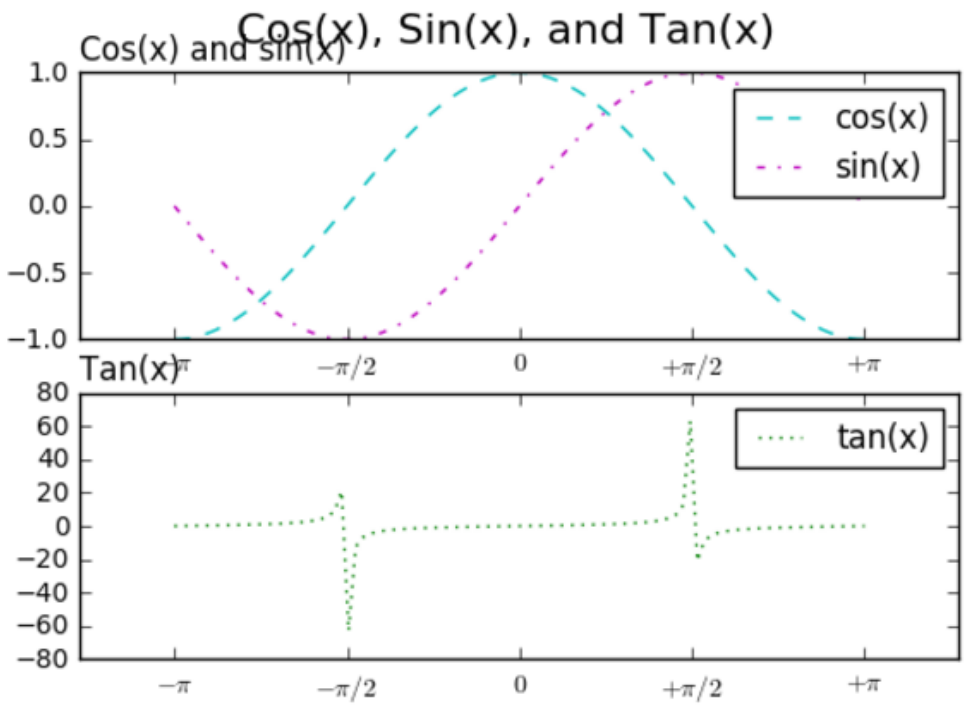
# Titles

Title for the entire figure

```
figure_1 = plt.figure()

figure_1.suptitle("Cos(x), Sin(x), and Tan(x)", fontsize=16)

facet_1 = figure_1.add_subplot(2, 1, 1)

facet_1.set_title("Cos(x) and sin(x)", loc="left")

plt.plot(x, cos_x, color = "c", linestyle = "--", label = "cos(x)")
plt.plot(x, sin_x, color = "m", linestyle = "-.", label = "sin(x)")

facet_1.legend(loc="upper right")

facet_1_ticks = facet_1.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_1_labels = facet_1.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

facet_2 = figure_1.add_subplot(2, 1, 2)

facet_2.set_title("Tan(x)", loc="left")

plt.plot(x, tan_x, color = "g", linestyle = ":", label = "tan(x)")

facet_2.legend(loc="upper right")

facet_2_ticks = facet_2.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_2_labels = facet_2.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

plt.show()
```

Facet 1 title

Facet 2 title



# Titles

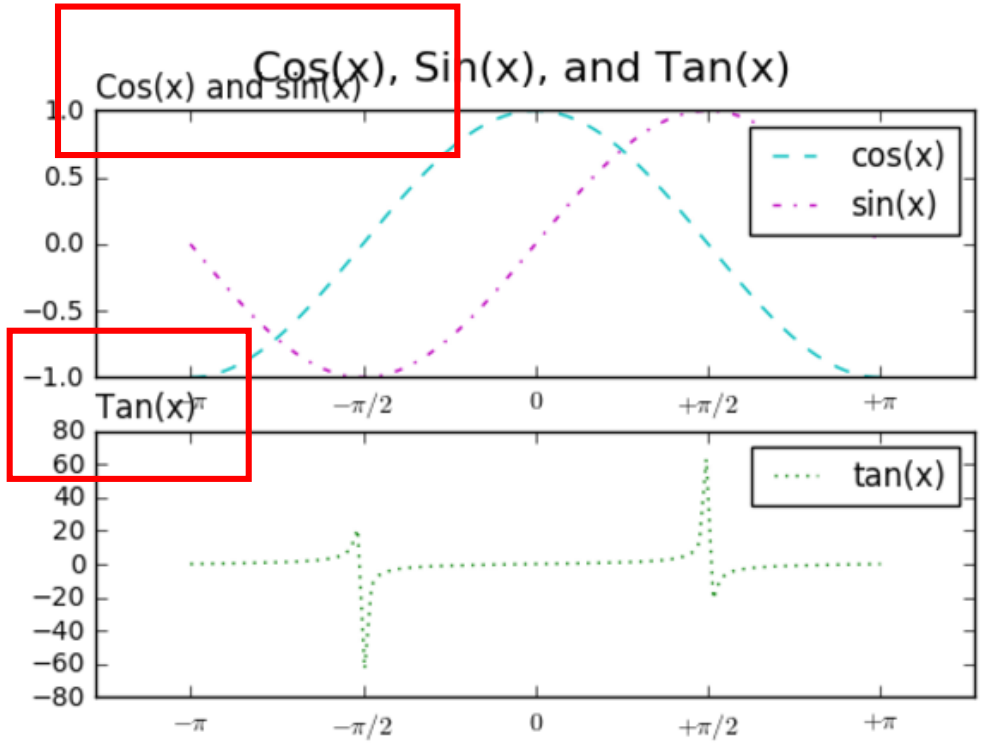
Title for the entire figure

```
figure_1 = plt.figure()

figure_1.suptitle("Cos(x), Sin(x), and Tan(x)", fontsize=16)

facet_1 = figure_1.add_subplot(2, 1, 1)

facet_1.set_title("Cos(x) and sin(x)", loc="left")

plt.plot(x, cos_x, color = "c", linestyle = "--", label = "cos(x)")
plt.plot(x, sin_x, color = "m", linestyle = "-.", label = "sin(x)")

facet_1.legend(loc="upper right")

facet_1_ticks = facet_1.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_1_labels = facet_1.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

facet_2 = figure_1.add_subplot(2, 1, 2)

facet_2.set_title("Tan(x)", loc="left")

plt.plot(x, tan_x, color = "g", linestyle = ":", label = "tan(x)")

facet_2.legend(loc="upper right")

facet_2_ticks = facet_2.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_2_labels = facet_2.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

plt.show()
```

Facet 1 title

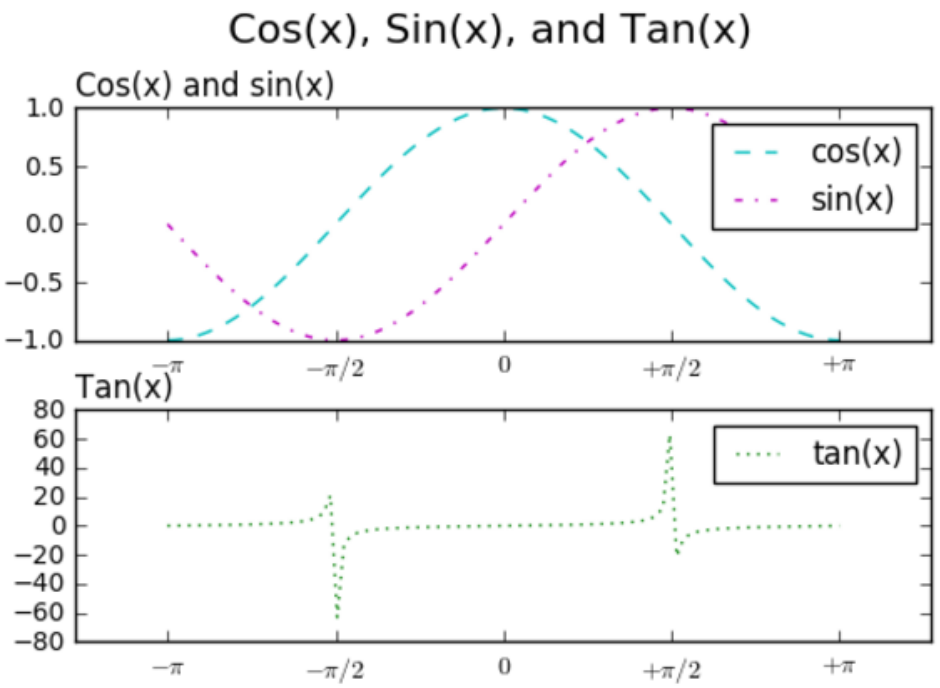
Facet 2 title

These run together in an ugly way. Let's resize!



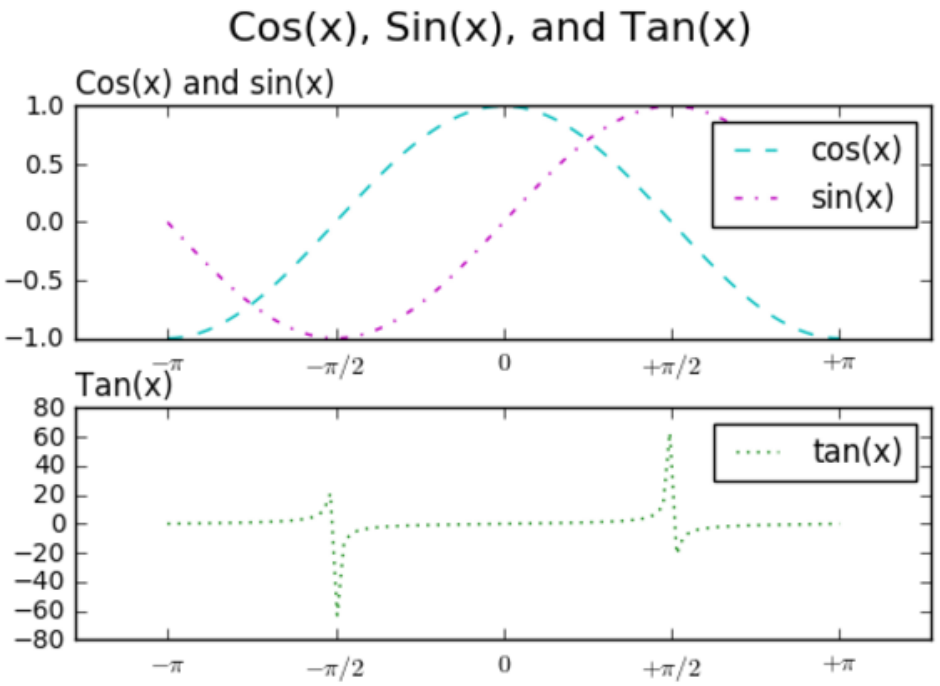
# Adjusting spacing around/between subplots

```
figure_1 = plt.figure()

figure_1.suptitle("Cos(x), Sin(x), and Tan(x)", fontsize=16)

facet_1 = figure_1.add_subplot(2, 1, 1)

facet_1.set_title("Cos(x) and sin(x)", loc="left")

plt.plot(x, cos_x, color = "c", linestyle = "--", label = "cos(x)")
plt.plot(x, sin_x, color = "m", linestyle = "-.", label = "sin(x)")

facet_1.legend(loc="upper right")

facet_1_ticks = facet_1.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_1_labels = facet_1.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

facet_2 = figure_1.add_subplot(2, 1, 2)

facet_2.set_title("Tan(x)", loc="left")

plt.plot(x, tan_x, color = "g", linestyle = ":", label = "tan(x)")

facet_2.legend(loc="upper right")

facet_2_ticks = facet_2.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_2_labels = facet_2.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

figure_1.subplots_adjust(top = 0.85, hspace= 0.3)

plt.show()
```



# Adjusting spacing around/between subplots

```
figure_1 = plt.figure()

figure_1.suptitle("Cos(x), Sin(x), and Tan(x)", fontsize=16)

facet_1 = figure_1.add_subplot(2, 1, 1)

facet_1.set_title("Cos(x) and sin(x)", loc="left")

plt.plot(x, cos_x, color = "c", linestyle = "--", label = "cos(x)")
plt.plot(x, sin_x, color = "m", linestyle = "-.", label = "sin(x)")

facet_1.legend(loc="upper right")

facet_1_ticks = facet_1.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_1_labels = facet_1.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

facet_2 = figure_1.add_subplot(2, 1, 2)

facet_2.set_title("Tan(x)", loc="left")

plt.plot(x, tan_x, color = "g", linestyle = ":", label = "tan(x)")

facet_2.legend(loc="upper right")

facet_2_ticks = facet_2.set_xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
facet_2_labels = facet_2.set_xticklabels([r'$-\pi$', r'$-\pi/2$', r'$0$', r'$+\pi/2$', r'$+\pi$'])

figure_1.subplots_adjust(top = 0.85, hspace= 0.3)

plt.show()
```



# Saving plots

```
plt.savefig('figure_1_path.png', dpi = 300, bbox_inches = "tight")
```

# Configuring matplotlib

Save customized parameters for figure size, color schemes, fonts, etc. for consistent publication-quality figures. Can be done readily using a dictionary:

```
font_options = {'family' : 'monospace',
                'weight' : 'bold',
                'size' : 'small'}

plt.rc('font', **font_options)
```

# Resources for this presentation

This presentation was made based on examples from:

http://www.scipy-lectures.org/intro/matplotlib/

and

“Python for Data Analysis” by Wes McKinney