scmamp, seaborn

Aleš Calábek
469489@mail.muni.cz

Faculty of Informatics, Masaryk University

April 26, 2021

scmamp

- Statistical Comparison of Multiple Algorithms in Multiple Problems [1] [2]
- install .packages('scmamp')
- library (" scmamp")

```
df <- data.frame(DT  = rnorm(10, mean=50, sd=10),
                 RF  = rnorm(10, mean=70, sd= 2),
                 MLP = rnorm(10, mean=70, sd=10),
                 SVM = rnorm(10, mean=65, sd= 5))
```

```
> df/100
          DT        RF       MLP       SVM
1  0.6370958 0.7260974 0.6693361 0.6727725
2  0.4435302 0.7457329 0.5218692 0.6852419
3  0.5363128 0.6722228 0.6828083 0.7017552
4  0.5632863 0.6944242 0.8214675 0.6195537
5  0.5404268 0.6973336 0.8895193 0.6752478
6  0.4893875 0.7127190 0.6569531 0.5641496
7  0.6511522 0.6943149 0.6742731 0.6107770
8  0.4905341 0.6468709 0.5236837 0.6074546
9  0.7018424 0.6511907 0.7460097 0.5292896
10 0.4937286 0.7264023 0.6360005 0.6518061
```

omnibus test

- H0: all algorithms have the same accuracy
- more than 5 algorithms
  - imanDavenportTest()
  - friedmanTest()
- up to 5 algorithms
  - friedmanAlignedRanksTest()
  - quadeTest()

```
> imanDavenportTest(df)

        Iman Davenport's correction of Friedman's rank
            sum test

data:  df
Corrected Friedman's chi-squared = 8.4419, df1 = 3,
    df2 =
27, p-value = 0.0004053
```

```
> friedmanTest(df)

        Friedman's rank sum test

data:  df
Friedman's chi-squared = 14.52, df = 3, p-value =
    0.002276
```

```
> friedmanAlignedRanksTest(df)

        Friedman's Aligned Rank Test for Multiple
           Comparisons

data:  df
T = 12.092, df = 3, p-value = 0.007075
```

```
> quadeTest(df)

        Quade for Multiple Comparisons

data:  df
T = 6.6292, df = 3, p-value = 0.001684
```

post-hoc pair-wise test

- postHocTest() or separate functions
- tests:
  - wilcoxon, t-test, friedman, aligned ranks, quade, tukey
- corrections of p-values:
  - shaffer, bergmann, holland, finner, rom, li
- groupby

```
> pht <- postHocTest(data = df, test = ’friedman’,
    correct = ’finner’)
> pht$summary
            DT        RF      MLP       SVM
[1,] 0.5547297 0.6967309 0.682192 0.6318048
> pht$raw.pval
              DT           RF         MLP        SVM
DT            NA 0.0002755038 0.005583617 0.05674682
RF  0.0002755038           NA 0.386476231 0.08326452
MLP 0.0055836168 0.3864762308          NA 0.38647623
SVM 0.0567468165 0.0832645167 0.386476231         NA

> pht$corrected.pval
             DT          RF        MLP       SVM
DT           NA 0.001651885 0.01665749 0.1102734
RF  0.001651885          NA 0.44358648 0.1222597
MLP 0.016657494 0.443586483         NA 0.4435865
SVM 0.110273432 0.122259652 0.44358648        NA
```

```
> df['param'] <- range(1, 2)
> head(df, 5)
         DT        RF       MLP       SVM param
1 0.6370958 0.7260974 0.6693361 0.6727725     1
2 0.4435302 0.7457329 0.5218692 0.6852419     2
3 0.5363128 0.6722228 0.6828083 0.7017552     1
4 0.5632863 0.6944242 0.8214675 0.6195537     2
5 0.5404268 0.6973336 0.8895193 0.6752478     1

> pht.group <- postHocTest(data = df, test = 'friedman
   ', correct = 'finner', group.by = "param")
> pht.group$summary
  param        DT        RF       MLP       SVM
1     1 0.6133660 0.6882319 0.7323893 0.6379684
2     2 0.4960933 0.7052299 0.6319948 0.6256412
```

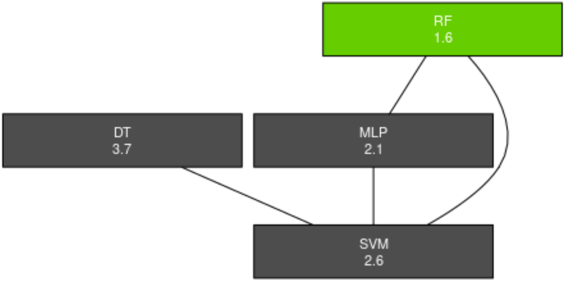
tables, graphs

- writeTabular()
- drawAlgorithmGraph, library(Rgraphviz)

```
> bold <- pht$corrected.pval < 0.05
> bold[is.na(bold)] <- FALSE
> writeTabular(table = pht$corrected.pval, format = ’f
    ’, bold = bold)
\begin{tabular}{|lllll|}
\hline
 & DT & RF & MLP & SVM \\
DT & n/a & {\bf 0.002} & {\bf 0.017} & 0.110 \\
RF & {\bf 0.002} & n/a & 0.444 & 0.122 \\
MLP & {\bf 0.017} & 0.444 & n/a & 0.444 \\
SVM & 0.110 & 0.122 & 0.444 & n/a \\
...
```

	DT	RF	MLP	SVM
DT	n/a	**0.002**	**0.017**	0.110
RF	**0.002**	n/a	0.444	0.122
MLP	**0.017**	0.444	n/a	0.444
SVM	0.110	0.122	0.444	n/a

```
> average.ranking <- colMeans(rankMatrix(df[-5]))
> average.ranking
 DT  RF MLP SVM
3.7 1.6 2.1 2.6
> drawAlgorithmGraph(pvalue.matrix = pht$corrected.
   pval, mean.value = average.ranking, font.size=10)
```



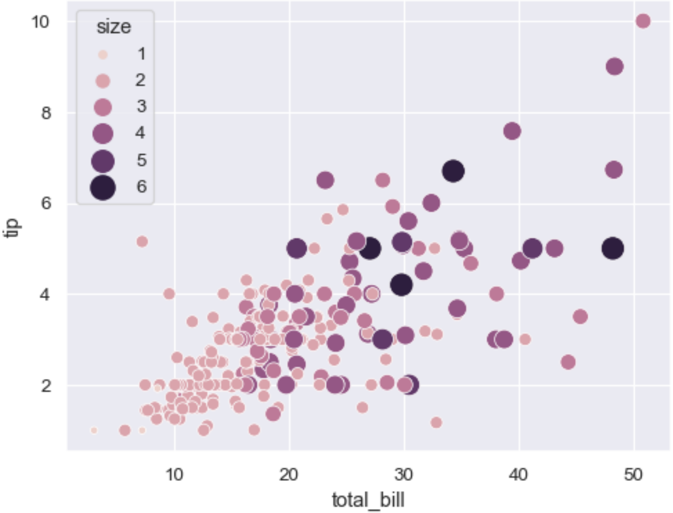
seaborn

- Python data visualization library based on matplotlib [3] [4]
- pip3 install seaborn
- import seaborn as sns

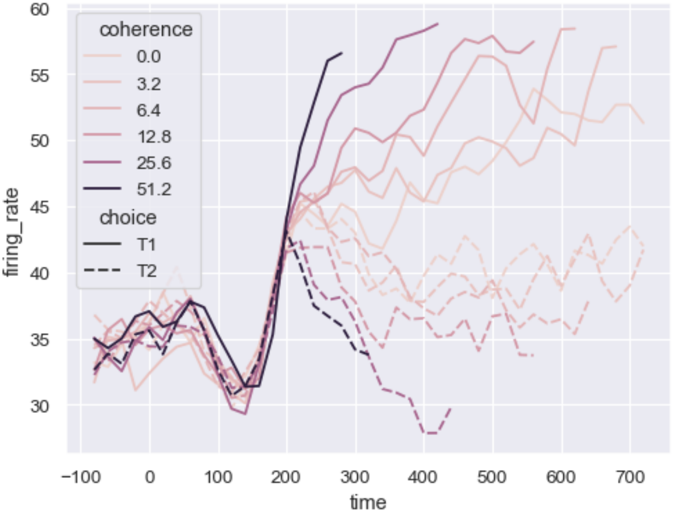
seaborn

- relational plots
- distribution plots
- categorical plots
- regression plots
- matrix plots
- multi-plot grids

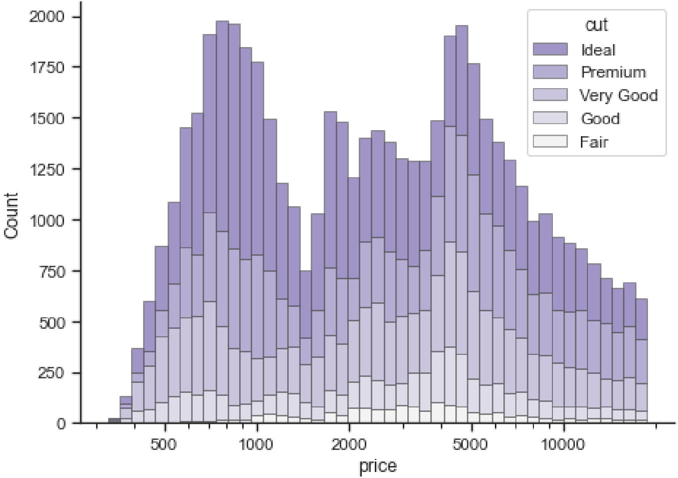
```
sns.scatterplot(
    data=tips, x="total_bill", y="tip", hue="size", s
    sizes=(20, 200), legend="full"
)
```



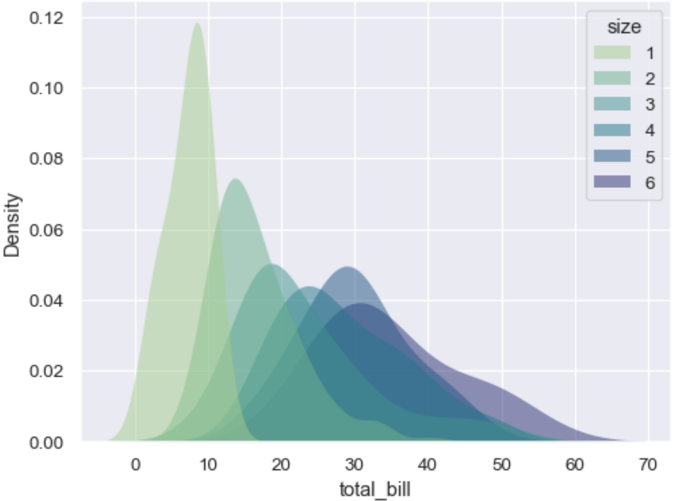
```
sns.lineplot(
    data=dots, x="time", y="firing_rate",
    hue="coherence", style="choice",
)
```



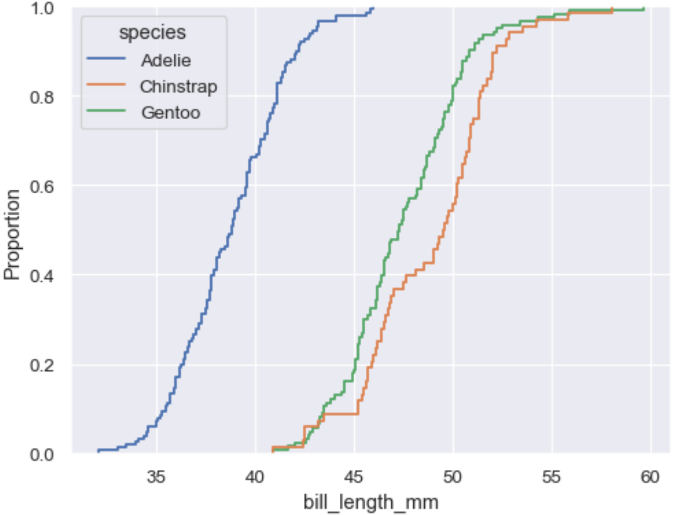
```
sns.histplot(
    diamonds, x="price", hue="cut", linewidth=.5,
    multiple="stack", palette="light:m_r",
    edgecolor=".3", log_scale=True,
)
```



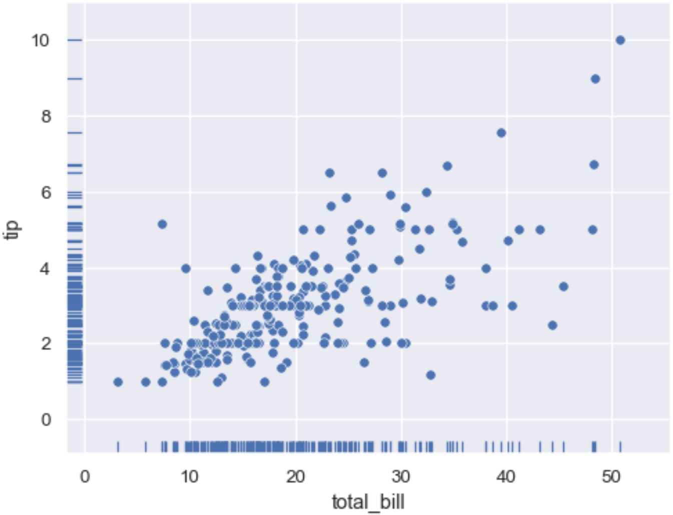
```
sns.kdeplot(
    data=tips, x="total_bill", hue="size",
    fill=True, common_norm=False, palette="crest",
    alpha=.5, linewidth=0,
)
```



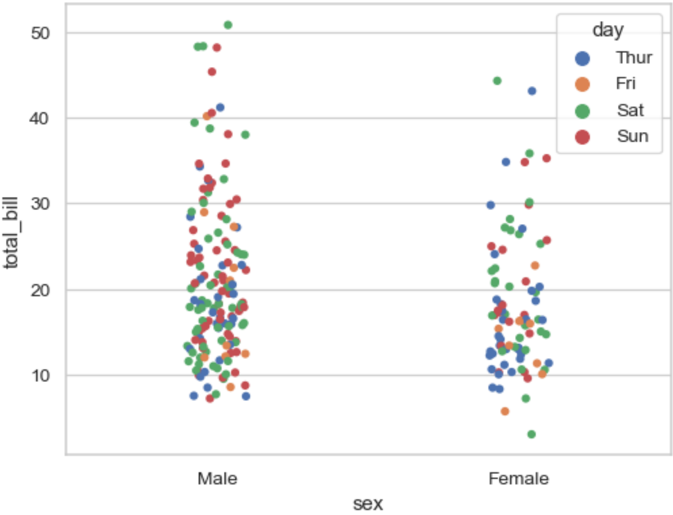
```
sns.ecdfplot(
    data=penguins, x="bill_length_mm",
    hue="species"
)
```



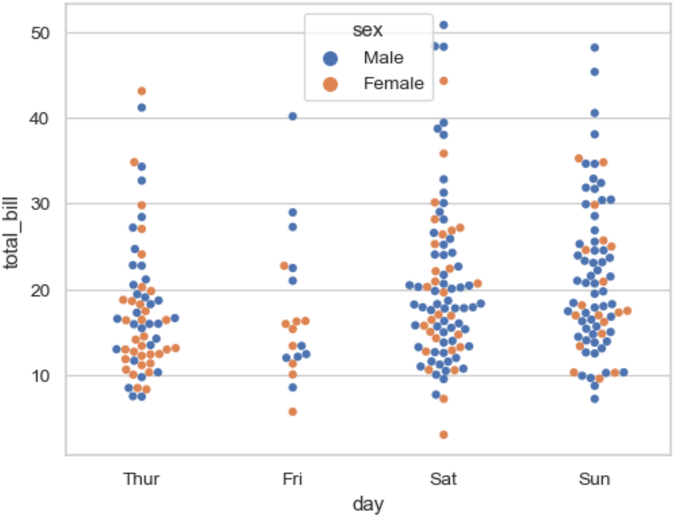
```
sns.scatterplot(data=tips, x="total_bill", y="tip")
sns.rugplot(data=tips, x="total_bill", y="tip")
```



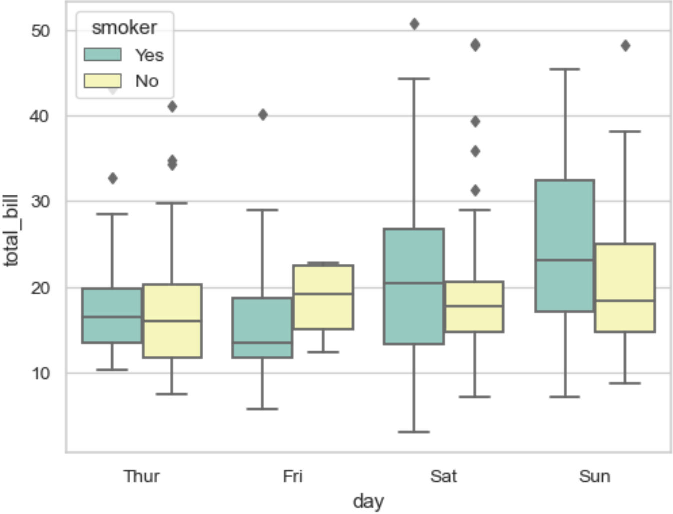
```
sns.stripplot(
    x="sex", y="total_bill", hue="day", data=tips
)
```



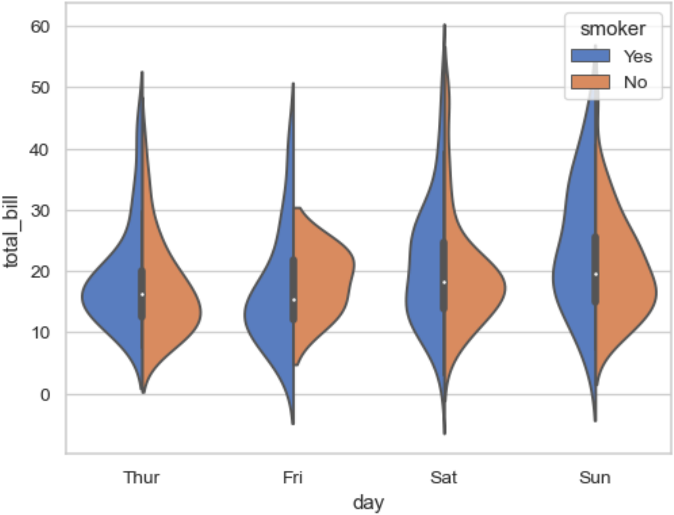
```
sns.swarmplot(
    x="day", y="total_bill", hue="sex", data=tips
)
```



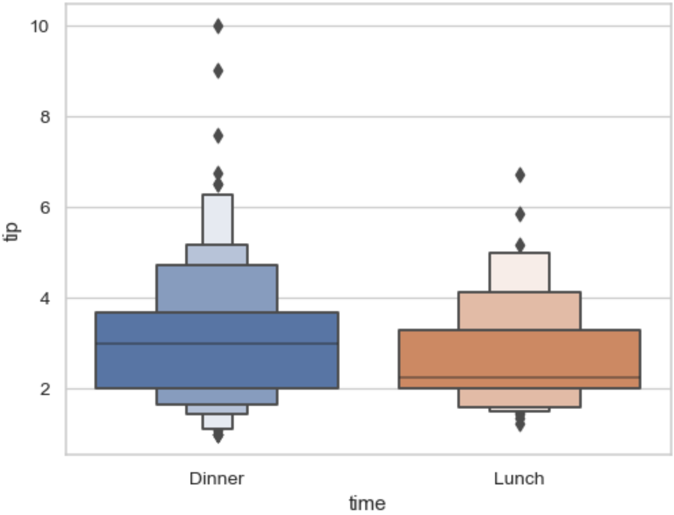
```
sns.boxplot(
    x="day", y="total_bill", hue="smoker",
    data=tips, palette="Set3"
)
```



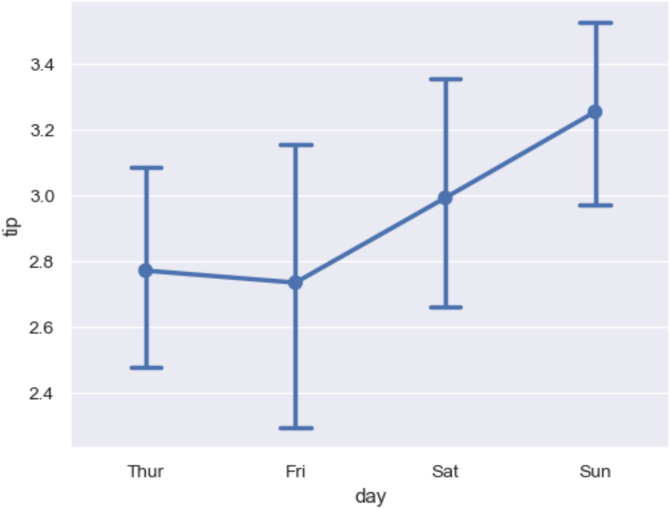
```
sns.violinplot(
    x="day", y="total_bill", hue="smoker",
    data=tips, palette="muted", split=True
)
```



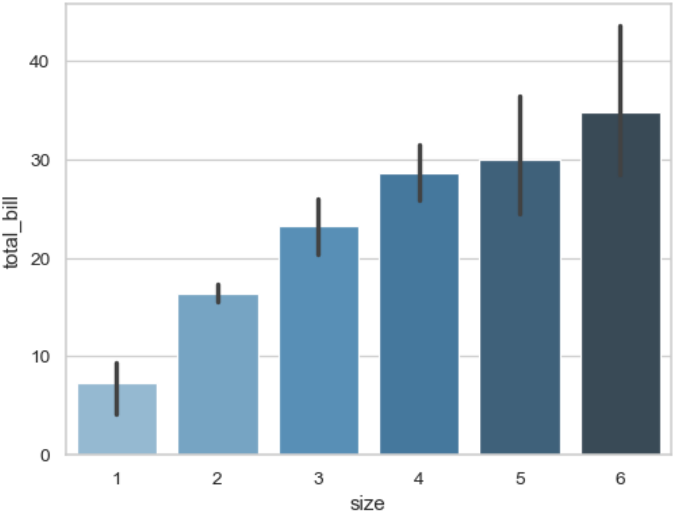
```
sns.boxenplot(
    x="time", y="tip", data=tips,
    order=["Dinner", "Lunch"]
)
```



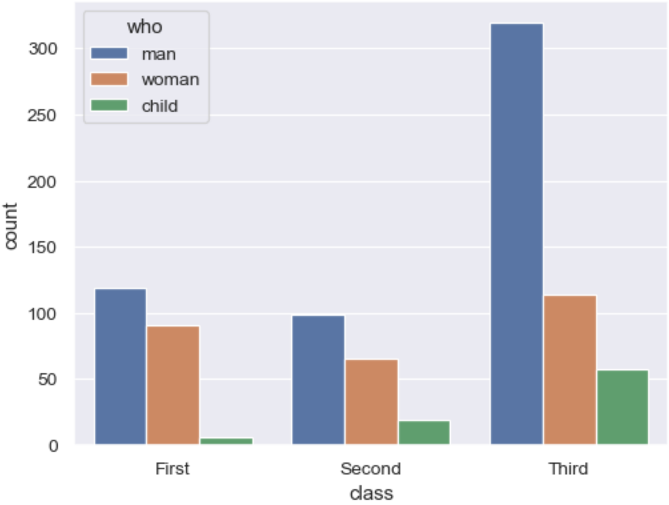
```
sns.pointplot(
    x="day", y="tip", data=tips, capsize=.2
)
```



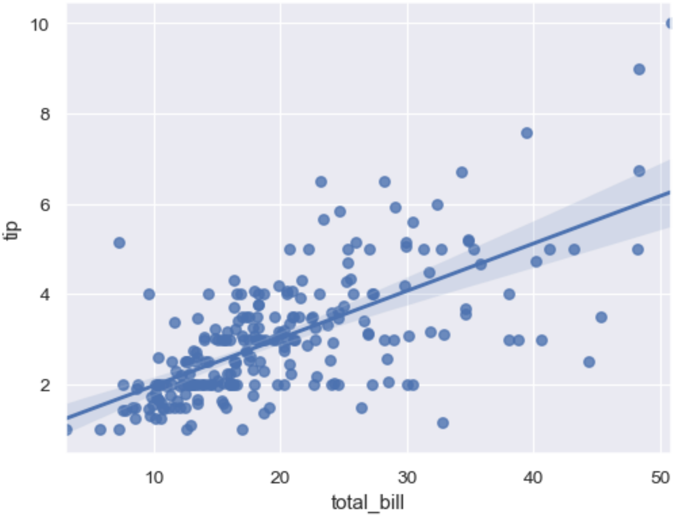
```
sns.barplot(
    x="size", y="total_bill",
    data=tips, palette="Blues_d"
)
```



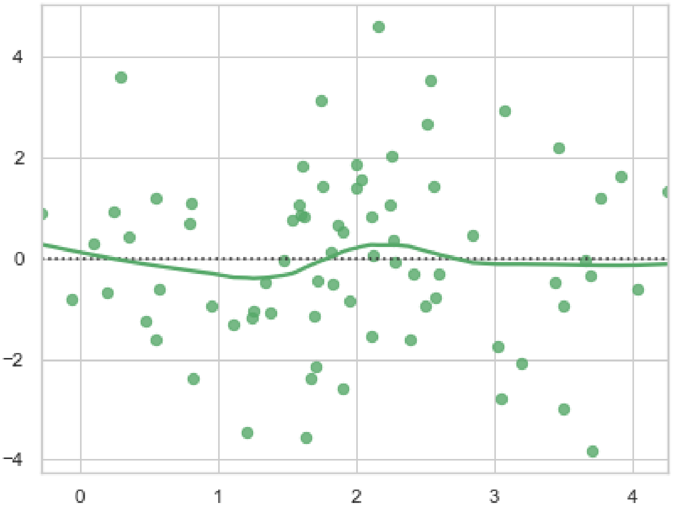
```
sns.countplot(x="class", hue="who", data=titanic)
```



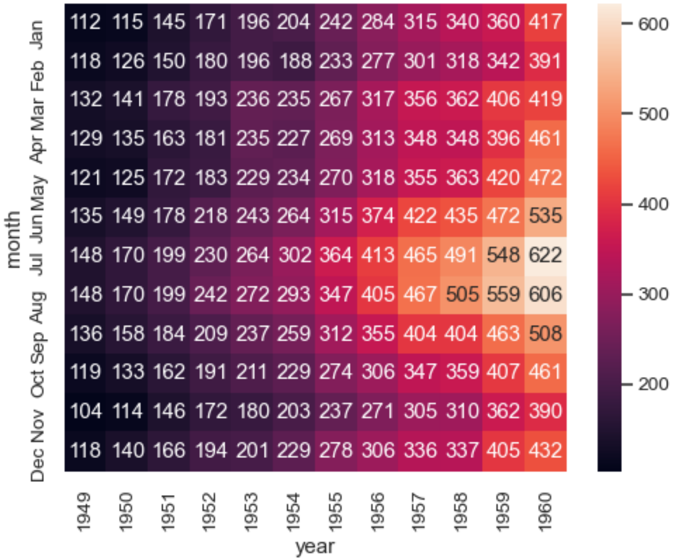
```
sns.regplot(x="total_bill", y="tip", data=tips)
```



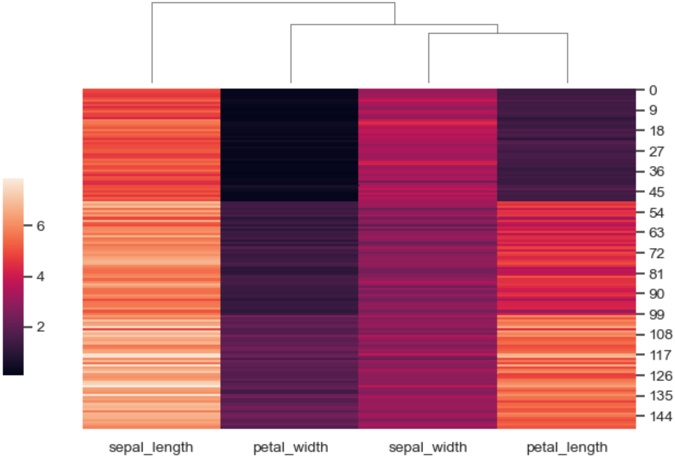
```
sns.residplot(x=x, y=y, lowess=True, color="g")
```



```
sns.heatmap(flights, annot=True, fmt="d")
```

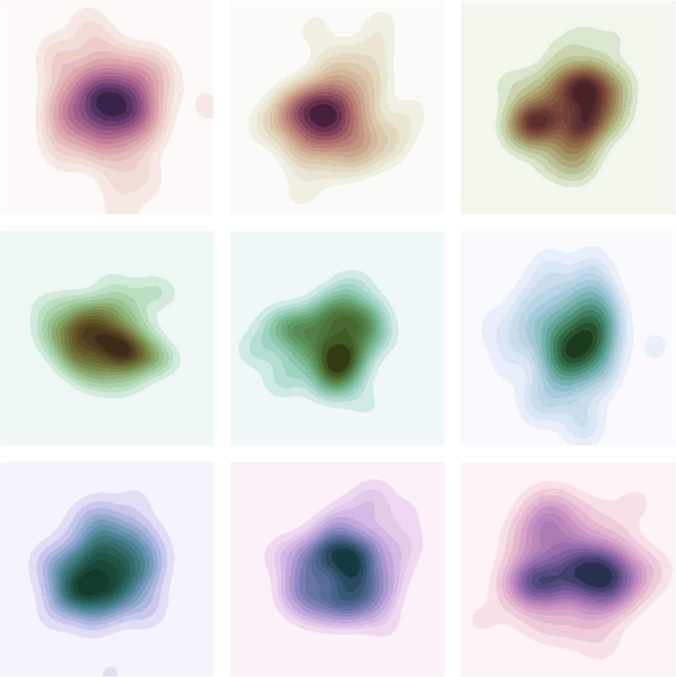


```
sns.clustermap(
    iris, row_cluster=False,
    dendrogram_ratio=(.1, .2),
    cbar_pos=(0, .2, .03, .4)
)
```

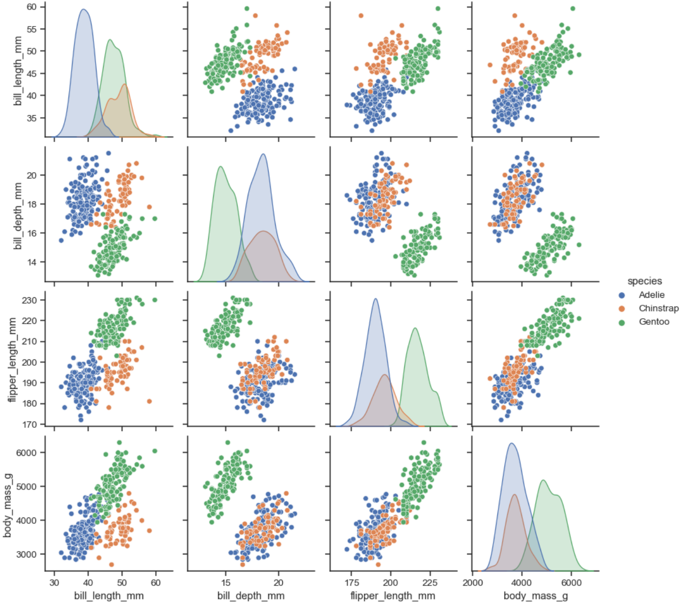


FacetGrid

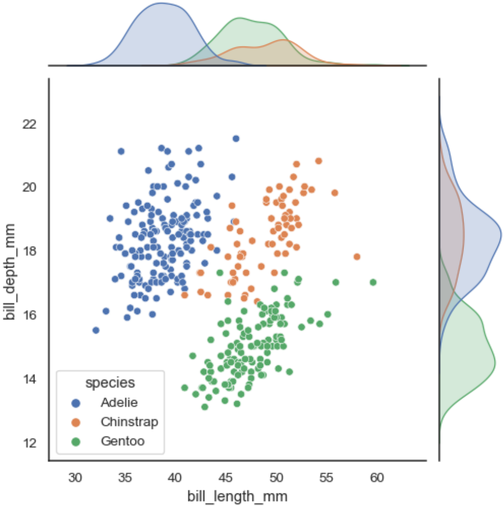
- relplot ()
- displot ()
- catplot ()
- lmplot()



```
sns.pairplot(penguins, hue="species")
```



```
sns.jointplot(
    data=penguins, x="bill_length_mm",
    y="bill_depth_mm", hue="species"
)
```



Bibliography I

[1] Borja Calvo. *scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems*. Oct. 2016. URL: `https://www.rdocumentation.org/packages/scmamp/versions/0.2.55`.

[2] Borja Calvo and Guzmán Santafé Rodrigo. “scmamp: Statistical comparison of multiple algorithms in multiple problems”. In: *The R Journal, Vol. 8/1, Aug. 2016* (2016).

[3] Michael L. Waskom. *seaborn*. URL: `https://seaborn.pydata.org/`.

[4] Michael L. Waskom. “seaborn: statistical data visualization”. In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: `10.21105/joss.03021`. URL: `https://doi.org/10.21105/joss.03021`.

MUNI

FACULTY

OF INFORMATICS