

Pokročilá vizualizace dat

Využití knihoven Matplotlib, Pandas a Seaborn

Vojtěch MRÁZEK

Fakulta Informačních Technologií, Vysoké Učení Technické v Brně
Brno, Czech Republic
mrazek@fit.vutbr.cz



Nástroje pro vizualizaci

■ Matplotlib

- Nízkoúrovňové rozhraní pro tvorbu grafů (největší flexibilita) obsahující metody pro tvorbu základních typů grafů z dat (jakýkoliv iterables)

■ Pandas

- Pandas **DataFrame** a **Series** implementují metodu `plot` (*accessor object* `pandas.plots`, který umožňuje přístup k rozhraní pro kreslení), která pro renderování využívá Matplotlib.

■ Seaborn

- Knihovna pracující nad Pandas **DataFrame** pro tvorbu velkého množství statisticky orientovaných grafů, která pro renderování využívá Matplotlib.

■ Plotly.js

- Interaktivní JavaScript open-source prezentace grafů s wrapperem pro IPython.



Motivace: rozdíl mezi knihovnami

- Příklad: vizualizace dat z datasetu „Titanic“

```
df = sns.load_dataset("titanic")
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	C	Southampton	yes	False

- Cíl: graficky znázornit boxplotem věk pasažéra podle pohlaví a jednotlivých tříd
- Konvence pojmenování knihoven v celé přednášce:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```



Implementace č. 1: matplotlib

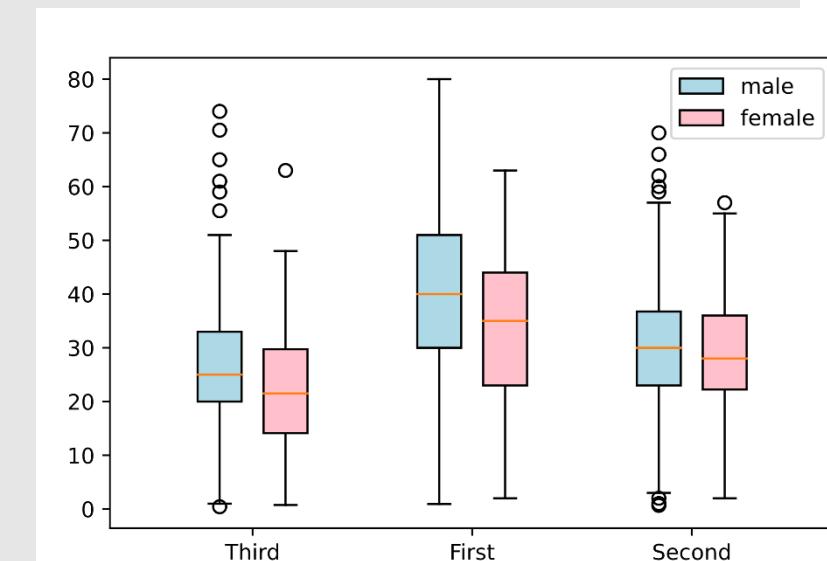
- Funkce plt.boxplot očekává na vstupu pole polí, které musíme manuálně vytvořit.

```

for rid, sex in enumerate(["male", "female"]):
    dd = [
        df.loc[(df["class"] == pclass) & (df["sex"] == sex), "age"].dropna()
        for pclass in classes
    ]
    bp = plt.boxplot(dd, positions=np.arange(3) + (rid - 0.5) * 0.3,
                     widths=0.2, patch_artist=True)
    color = colors[sex]
    for patch in bp['boxes']:
        patch.set_facecolor(color)

from matplotlib.patches import Patch
plt.legend(handles =
            [Patch(facecolor=colors["male"], edgecolor='k',
                   label='male'),
             Patch(facecolor=colors["female"], edgecolor='k',
                   label='female')])
plt.xticks(np.arange(3), classes)

```



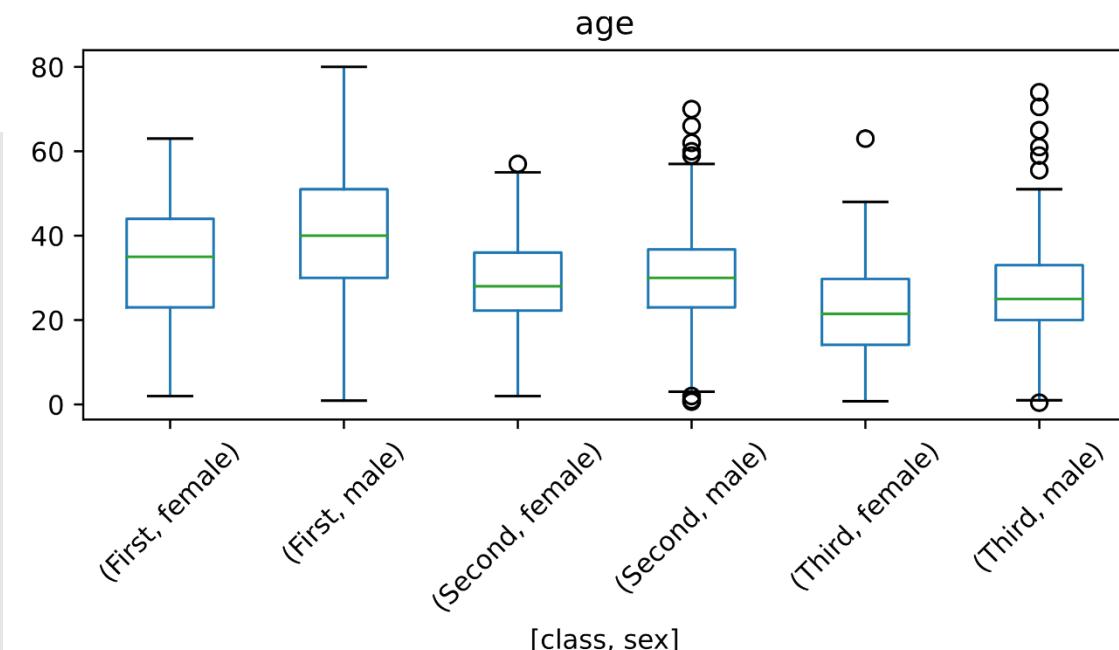
Implementace č. 2: pandas

- S využitím groupby jsme schopni takový graf naimplementovat jednodušeji

```
alld = []
labels = []
for (sex, cls), df_s in df.groupby(["class", "sex"]):
    alld.append(df_s["age"].dropna().to_numpy())
    labels.append((sex, cls))

plt.boxplot(alld)
plt.xticks(np.arange(6) + 1, labels, rotation=45)
```

Boxplot grouped by ['class', 'sex']



- Pandas tuto funkcionalitu implementuje v metodě boxplot
- Pro základní vizualizaci dostatečné, často je nutný manuální zásah.

```
bp = df.boxplot(column="age", grid=False, by=["class", "sex"], rot=45)
```

Implementace č. 2: pandas

Příklad sjednocení vizuálního stylu

- musíme zpět k nízkoúrovňovým funkcím knihovny matplotlib.

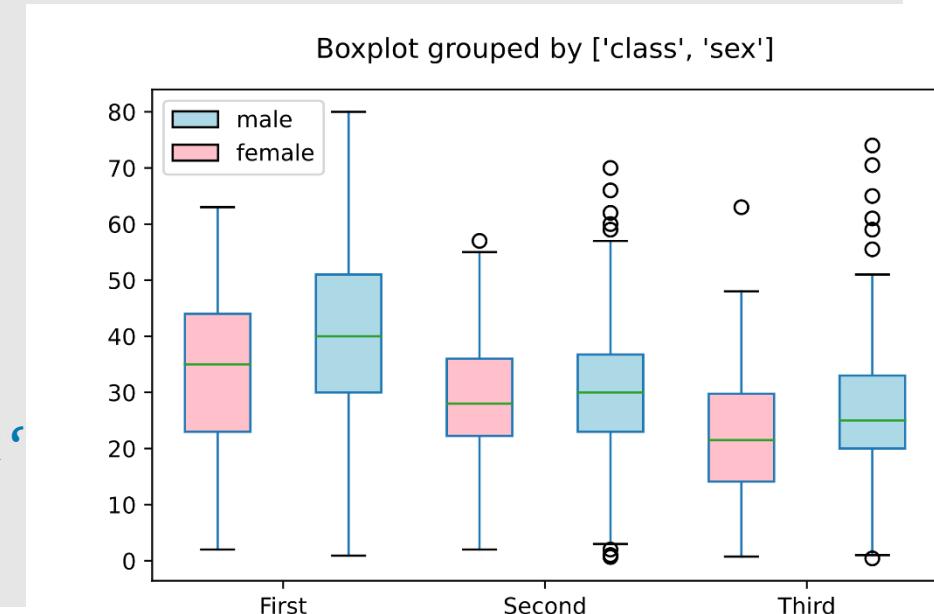
```

ret = df.boxplot(column="age", grid=False, by=["class", "sex"],
                  return_type="both", patch_artist=True)
ax, bp = ret["age"]

for i, patch in enumerate(bp["boxes"]):
    patch.set_facecolor("lightblue" if i % 2 else "pink")

ax.set(xticks=np.arange(3) * 2 + 1.5,
       xticklabels=sorted(classes),
       xlabel="", title="")
from matplotlib.patches import Patch
plt.legend(handles = [
    Patch(facecolor=colors["male"], edgecolor='k',
          label='male'),
    Patch(facecolor=colors["female"], edgecolor='k',
          label='female')
])

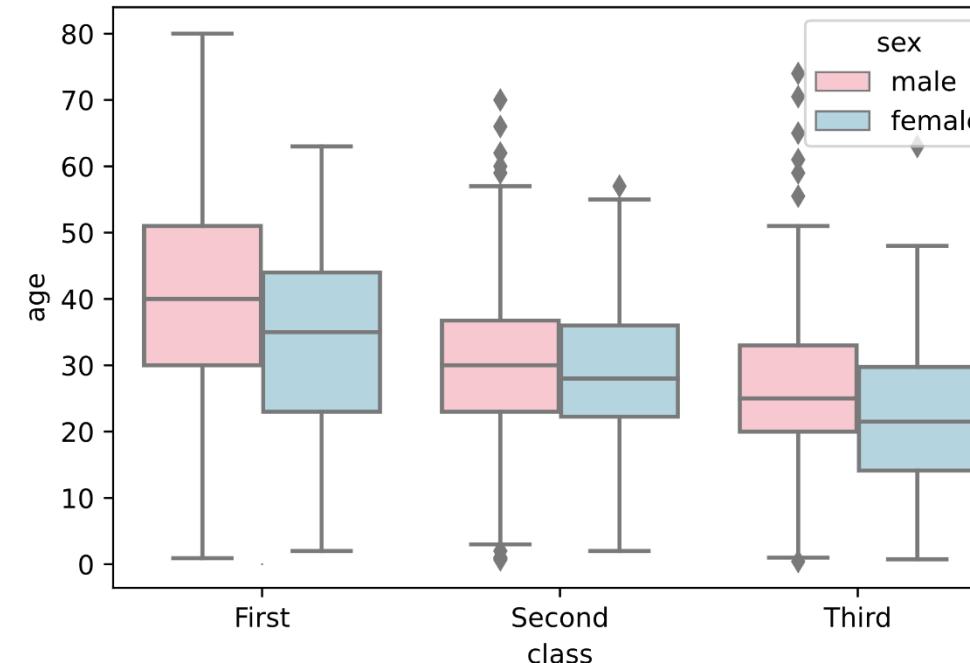
```



Implementace č. 3: seaborn

- Z pohledu abstrakce pracuje na nejvyšší úrovni knihovna seaborn.
 - pestrá škála grafů, dovoluje automaticky vytvářet i grafy komponované z podgrafů.
 - nepředstavuje však vždy nejfektivnější řešení jde-li nám o dobu vytváření grafu

```
ax = sns.boxplot(data=df, y="age", x="class", hue="sex")
```

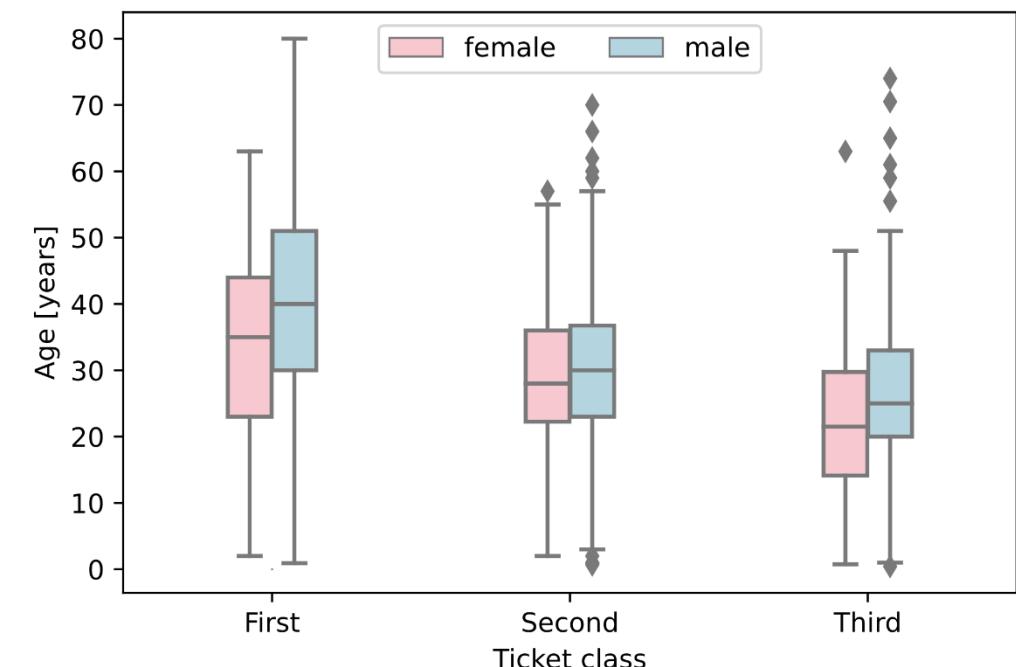


Implementace č. 3: seaborn

- Výstup máme též možnost upravit pomocí matplotlib rozhraní
- Příklad sjednocení vizuálního stylu:

```
ax = sns.boxplot(data=df, y="age", x="class", hue="sex",
                  palette={"female": "pink", "male": "lightblue"},
                  hue_order=["female", "male"], width=0.3)
ax.legend(loc="upper center", ncol=2)
ax.set(ylabel="Age [years]", xlabel="Ticket class")
```

- změn je mnohem méně oproti pandas



Seaborn – koncept grafů

■ Axes-level funkce

- vytváří samostatné grafy v rámci `matplotlib.pyplot.Axes` objektu
- úpravy přímo na úrovni matplotlib.



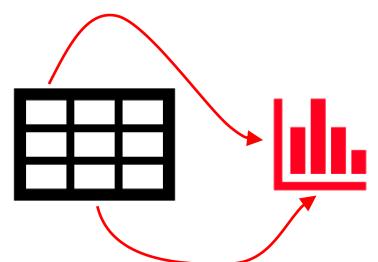
■ Figure-level funkce

- vytvářejí podgrafy, plní je do mřížky tvořené sloupci a řádky
- přístup pro modifikaci přes seaborn objekt **FacetGrid** (a odtud je přístup k matplotlib funkcím)



■ Vstupy a výstupy

- bud' specifikujeme dataset `pd.DataFrame` a jednotlivé parametry grafu jsou sloupce
- nebo můžeme také vkládat přímo `pd.Series` jako parametry.



■ Další funkce

- úprava vzhledu přes téma a jejich modifikace
- načítání standardních datasetů.

Vizualizace vztahu (relace) – čárový graf

- Základní volání obsahuje
 - DataFrame
 - názvy sloupců určující hodnoty použité pro souřadnici X a Y
- Další práce s grafem (zobrazení, uložení)
 - viz matplotlib.

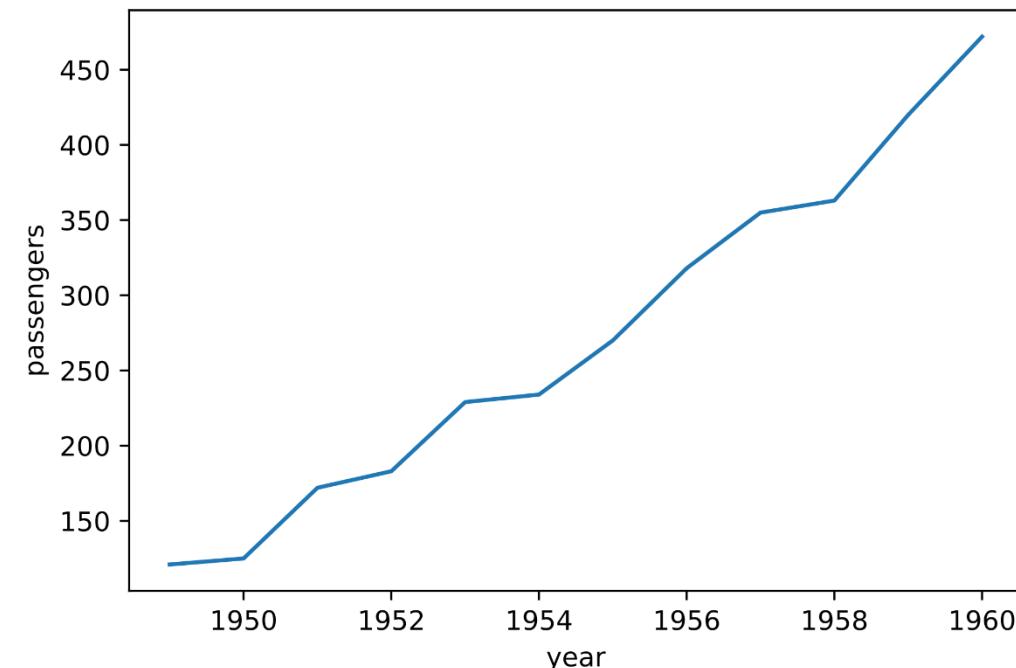
```
sns.lineplot(
    data=flights[flights["month"] == "May"],
    x="year", y="passengers"
)
plt.savefig("fig/line_01.svg")
```

- Místo názvů sloupců lze použít pd.Series

```
fm = flights[flights["month"] == "May"]
sns.lineplot(x=fm["year"], y=fm["passengers"])
```

```
flights = sns.load_dataset("flights")
```

	year	month	passengers
0	1949	Jan	112
1	1949	Feb	118
...
142	1960	Nov	390
143	1960	Dec	432



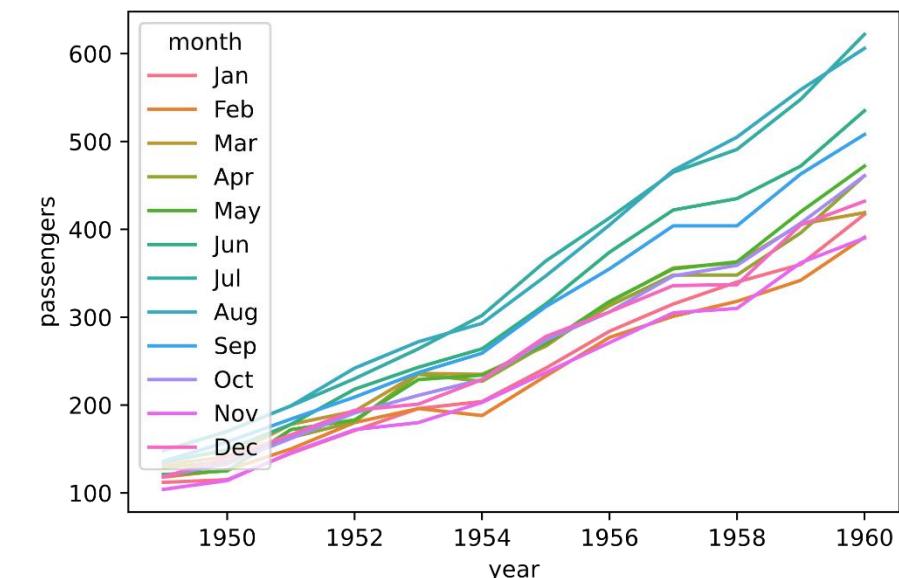
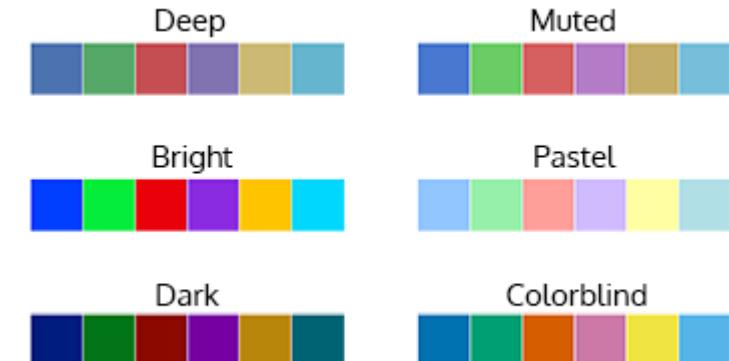
Vizualizace více datových řad a změna parametrů jednotlivých datových řad

- Argument `hue` určuje zdroj definující barvu
 - název sloupce nebo Series
- Argument `palette` určuje lokální paletu barev, avšak paletu je možné definovat i globálně

```
sns.set_palette("dark")
```

- Pořadí vkládání jednotlivých datových řad odpovídá pořadí, jak jsou data uspořádána v DataFrame.
 - Argument `hue_order` dovoluje změnit pořadí

```
sns.lineplot(data=flights,
              x="year", y="passengers",
              hue="month")
```



Styl a šířka čar

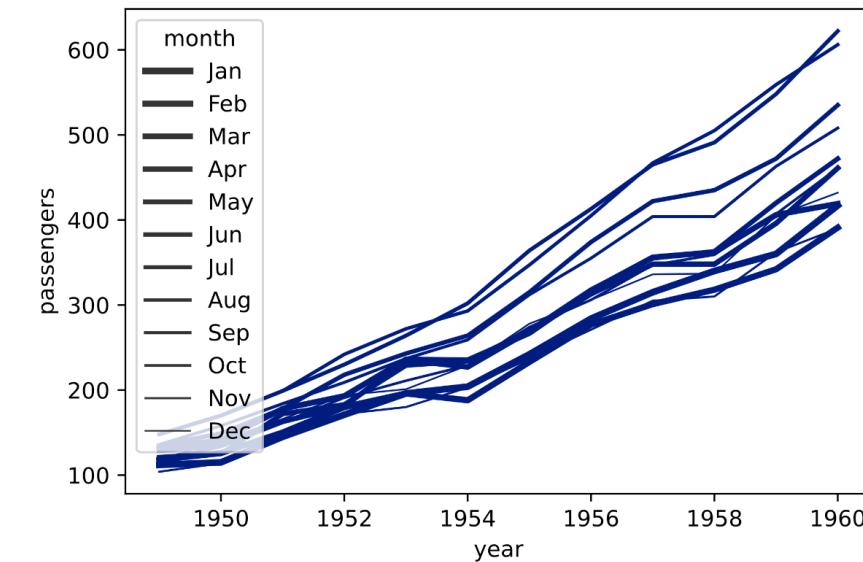
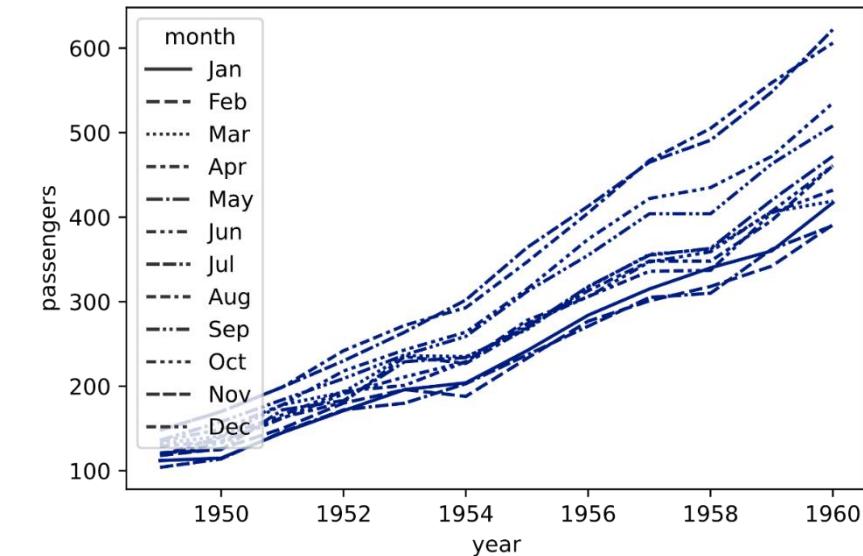
Podobně jako barvu je možné nastavit

- styl čáry pomocí argumentu `style`
 - pořadí může být pře definováno argumentem `style_order`

```
sns.lineplot(data=flights,  
             x="year", y="passengers",  
             style="month")
```

- šířku čáry pomocí argumentu `size`

```
sns.lineplot(data=flights,  
             x="year", y="passengers",  
             size="month")
```



Kombinace jednotlivých prvků

- Jednotlivé styly můžeme kombinovat.

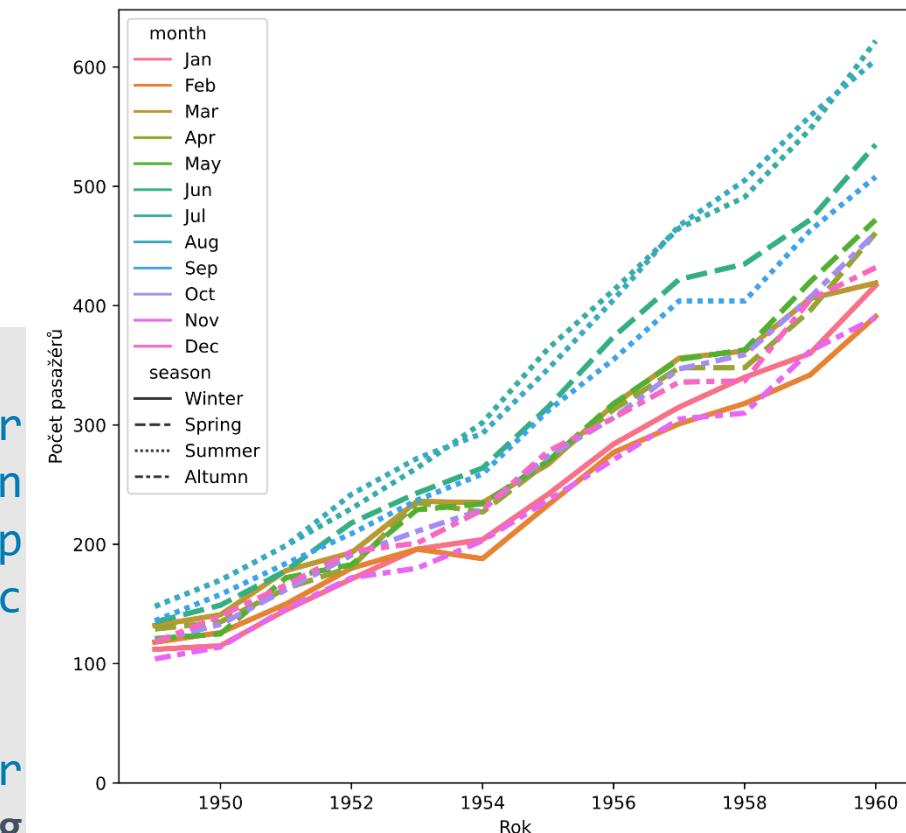
```

flights2 = flights.copy()
flights2.loc[flights2["month"].isin(["Jan", "Feb", "Mar",
flights2.loc[flights2["month"].isin(["Apr", "May", "Jun",
flights2.loc[flights2["month"].isin(["Jul", "Aug", "Sep",
flights2.loc[flights2["month"].isin(["Oct", "Nov", "Dec",

fig = plt.figure(figsize=(8, 8))
ax = sns.lineplot(data=flights2, x="year", y="passenger",
                   hue="month", style="season", lw=3, ax=fig.gca())
ax.set_xlim(1950, 1960)
ax.set_ylim(0, None)
ax.set(xlabel="Rok", ylabel="Počet pasažérů")

fig.savefig("fig/line_all.svg")

```



- Další argumenty, které nejsou zpracovány pandas se stávají argumenty metody [matplotlib.axes.Axes.plot](#)

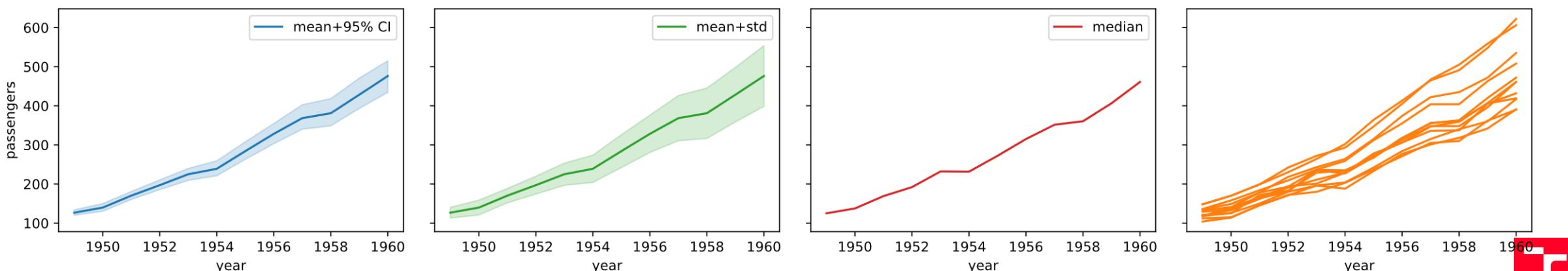
Automatická agregace dat

- Pokud existuje pro jednu kombinaci `x`, `y`, `style`, `hue` a `size` více datových bodů, dochází automaticky k agregaci dat.
 - Seaborn automaticky ukazuje interval spolehlivosti (*confidence interval*)
- Varianta s převzorkováním (výchozí)
 - numerický interval spolehlivosti na 95% (argument `ci=95`)
 - využívá se [bootstrapping](#) – převzorkování na bázi náhodného vzorkování s `n_boot=1000` vzorky
- Varianta se směrodatnou odchylkou
 - argument `ci="sd"`
 - vykreslí průměr a směrodatnou odchylku (očekáváme normalitu)
- Další varianty
 - všechny „čáry“ přes sebe, nebo
 - vlastní agregační funkce

Automatická agregace dat

```
fig, (ax1, ax2, ax3, ax4) = plt.subplots(1, 4, figsize=(16, 3), sharey=True)
sns.lineplot(data=flights2, x="year", y="passengers",
              ax=ax1, color="tab:blue", label="mean+95% CI")
sns.lineplot(data=flights2, x="year", y="passengers",
              ax=ax2, ci="sd", color="tab:green", label="mean+std")
sns.lineplot(data=flights2, x="year", y="passengers",
              ax=ax3, ci=0, estimator=np.median, color="tab:red", label="median")
sns.lineplot(data=flights2, x="year", y="passengers",
              ax=ax4, units="month", estimator=None, color="tab:orange")

fig.tight_layout()
fig.savefig("fig/line_est.svg")
```



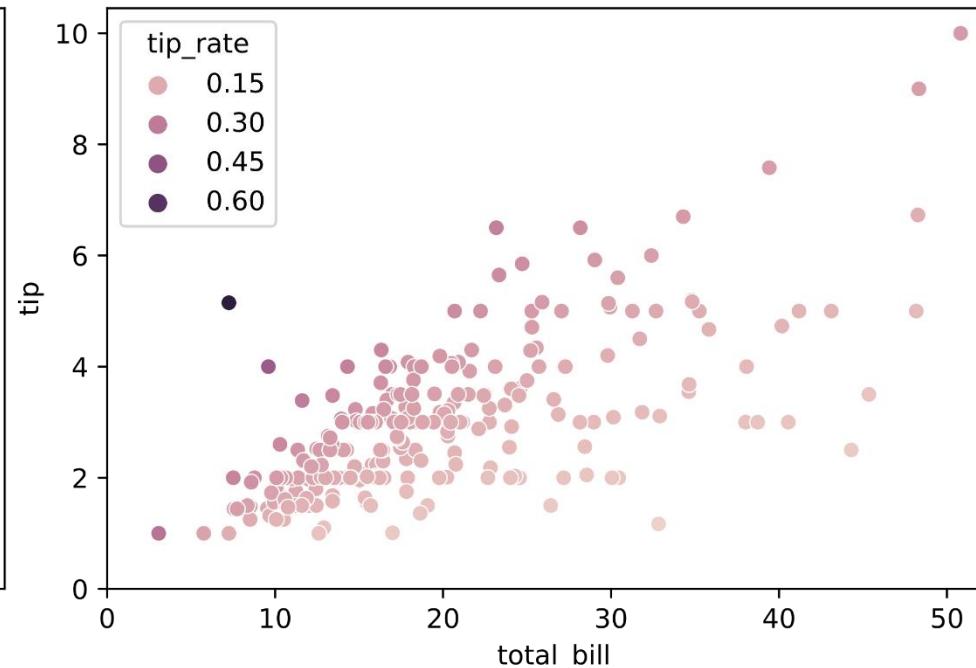
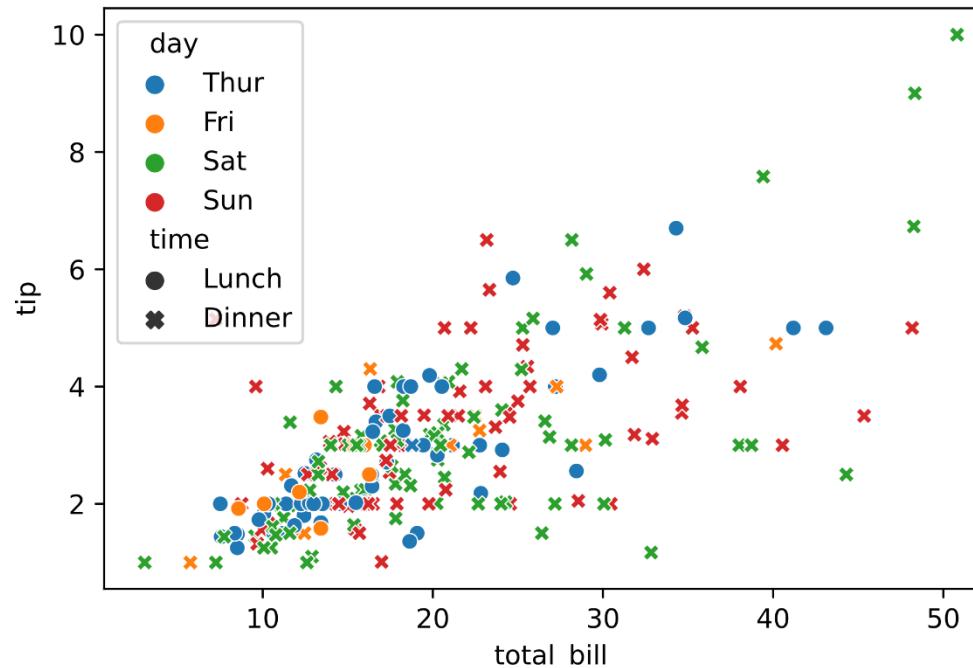
Vizualizace vztahu (relace) - bodový graf

- Axes-level graf
- Podobné chování, jako sns.lineplot
- Neumožňuje počítat intervaly spolehlivosti

```
tips = sns.load_dataset("tips")
```

	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

```
sns.scatterplot(data=tips, x="total_bill", y="tip", style="time", hue="day")
```



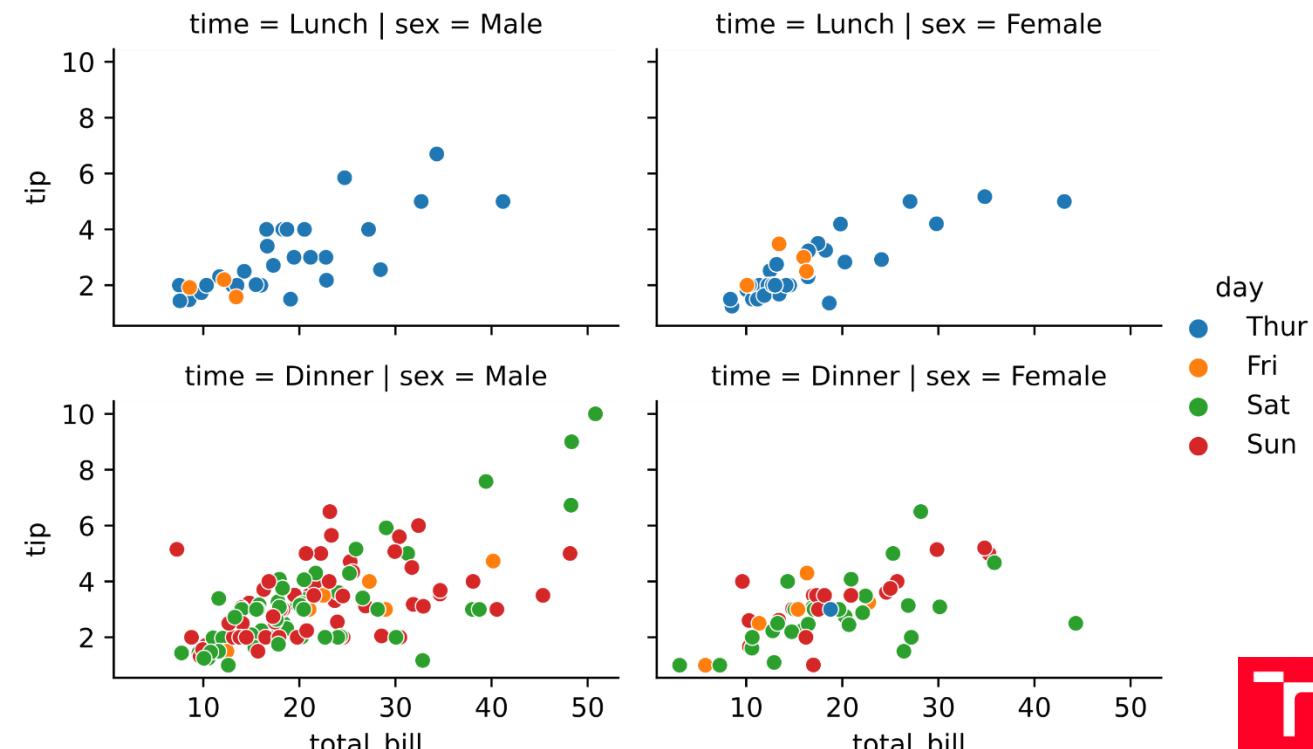
```
sns.scatterplot(data=tips, x="total_bill", y="tip", hue="tip_rate")
```



Vizualizace vztahu (relace) – relplot

- Figure-level graf obsahující jeden nebo více bodových nebo čárových grafů umístěných do pravidelné mřížky
- Metoda přidává argumenty:
 - `col` a `row` (případně `col_wrap`) – odkazy na sloupce v DataFrame, jejichž hodnoty budou definovat umístění v řádcích a sloupcích výsledného grafu
 - `kind` – druh grafu (řetězec scatter nebo line)

```
sns.relplot(data=tips, x="total_bill",
             y="tip", hue="day",
             col="sex", row="time",
             kind="scatter",
             height=2, aspect=2)
plt.savefig("fig/relplot_basic.svg")
```



Modifikace parametrů grafů a podgrafů

- Argument `facet_kws` umožňuje změnit hromadně řadu parametrů
 - např. sdílení os, barvu pozadí, okraje, apod.
- Pokud potřebujeme upravit konkrétní podgraf, je nutné využít API matplotlib

```

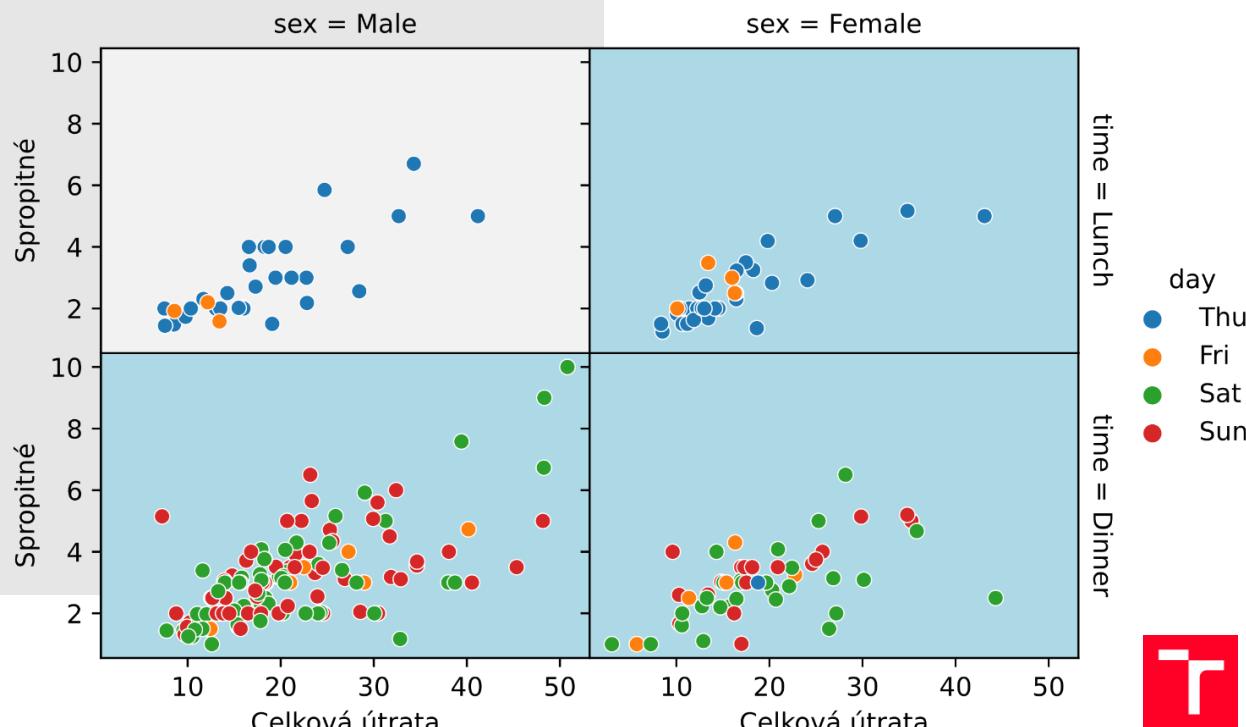
g = sns.relplot(data=tips, x="total_bill", y="tip", hue="day",
                 col="sex", row="time", kind="scatter",
                 facet_kws={"despine":False, "margin_titles": True},
                 height=2, aspect=1.5)

g.fig.subplots_adjust(wspace=0, hspace=0)

g.set(xlabel="Celková útrata",
      ylabel="Spropitné")
for x in g.axes.flatten():
    x.set_facecolor("lightblue")

g.axes[0,0].set_facecolor("0.95")

```



Vizualizace rozložení dat – histogram

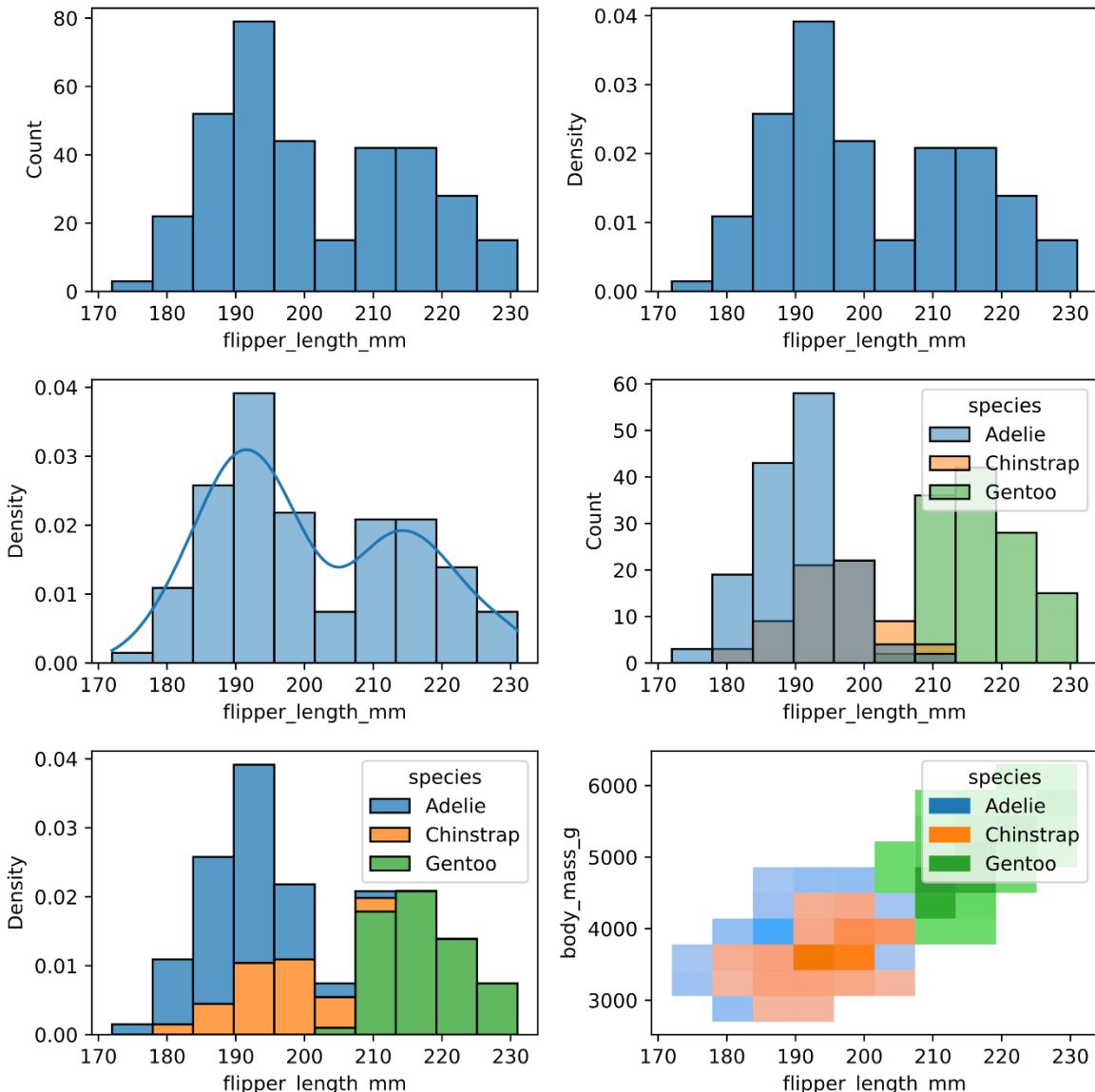
```
fig, axes = plt.subplots(3, 2, figsize=(8, 8))
ax = axes.flatten()

pkw = {"data": penguins,
        "x": "flipper_length_mm", "bins": 10}

sns.histplot(**pkw, ax=ax[0])
sns.histplot(**pkw, stat="density", ax=ax[1])

sns.histplot(**pkw, stat="density", kde=True,
            ax=ax[2])
sns.histplot(**pkw, hue="species", ax=ax[3])

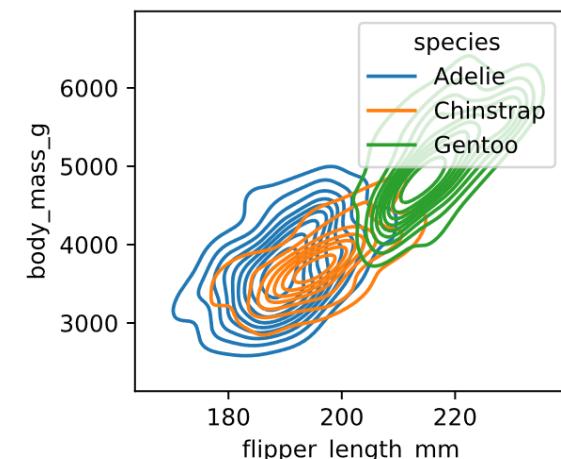
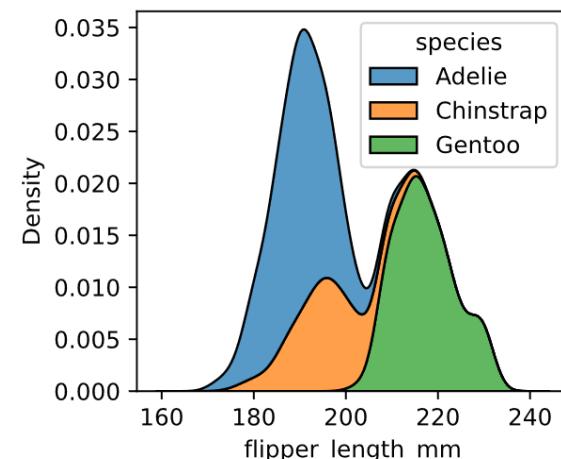
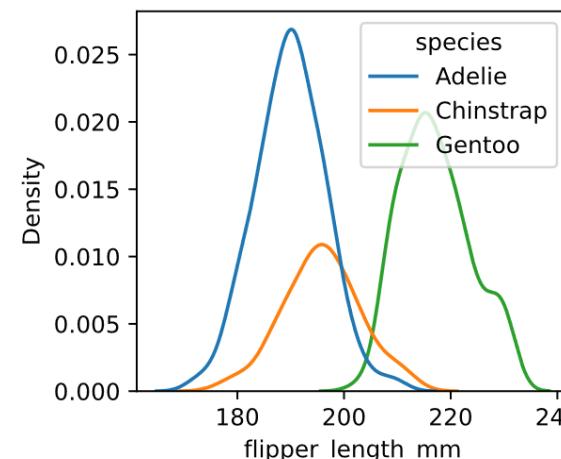
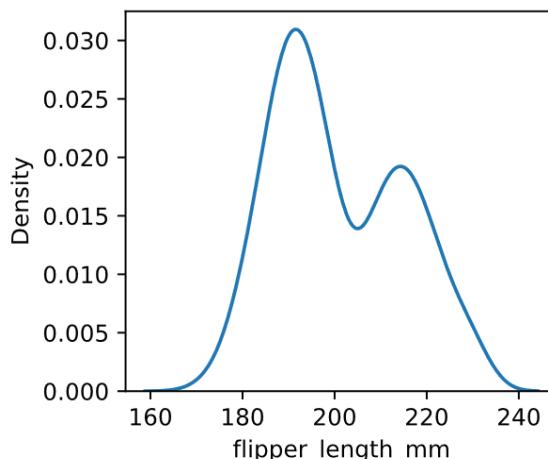
sns.histplot(**pkw, hue="species",
            stat="density",
            multiple="stack", ax=ax[4])
sns.histplot(**pkw, hue="species",
            y="body_mass_g", ax=ax[5])
```



Vizualizace rozložení dat – Kernel density estimate

- Kernel density estimate (KDE) graf je forma vizualizace podobná histogramu, ale data jsou reprezentována pomocí spojité funkce hustoty pravděpodobnosti
 - Funkce odhadnuta pomocí techniky [Kernel Density Estimation](#)
 - Argument `bw` definuje vyhlazení a argument `kernel` jádro (zde Gaussovské) pro KDE.

```
pkw = {"data": penguins, "x": "flipper_length_mm"}
sns.kdeplot(**pkw, ax=ax[0])
sns.kdeplot(**pkw, hue="species", ax=ax[1])
sns.kdeplot(**pkw, hue="species", multiple="stack", ax=ax[2])
sns.kdeplot(**pkw, hue="species", y="body_mass_g", ax=ax[3])
```

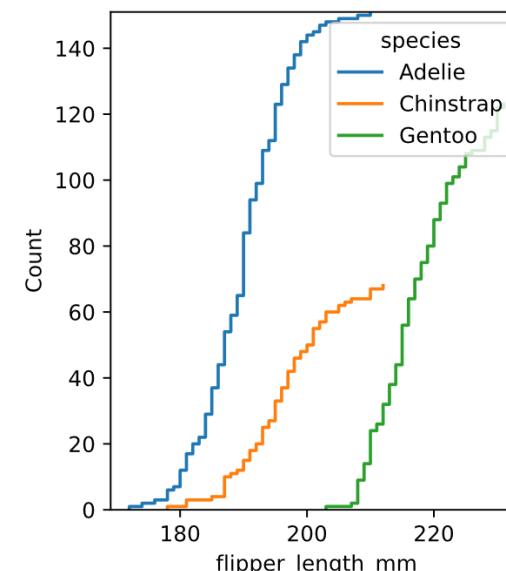
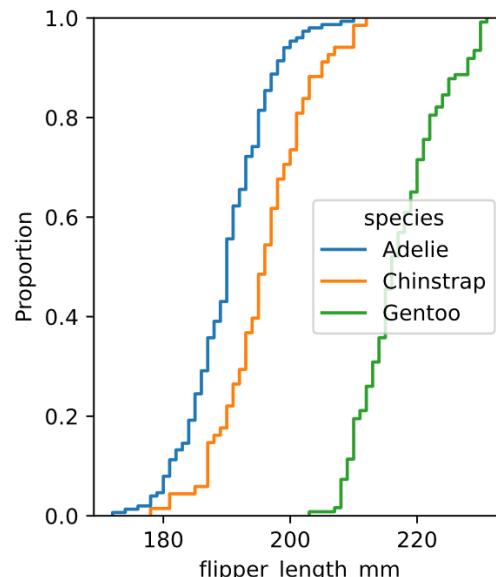


Vizualizace rozložení dat – empirická distribuční funkce a kobercový graf

■ Effective Cumulative Distribution Function

- Počítána jako integrál (suma) histogramu

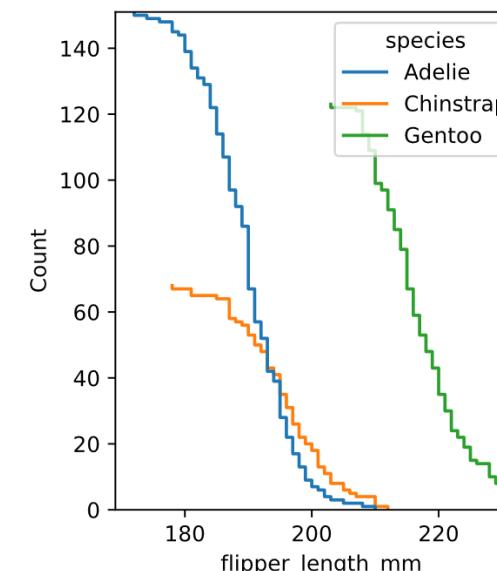
```
pkw = {"data": penguins,
        "x": "flipper_length_mm", "hue": "species"}
sns.ecdfplot(**pkw, ax=ax[0])
sns.ecdfplot(**pkw, stat="count", ax=ax[1])
sns.ecdfplot(**pkw, stat="count",
             complementary=True, ax=ax[2])
```



■ Rugplot

- Ukazuje místa výskytu vzorků
- V kombinaci s jiný distribučním grafem

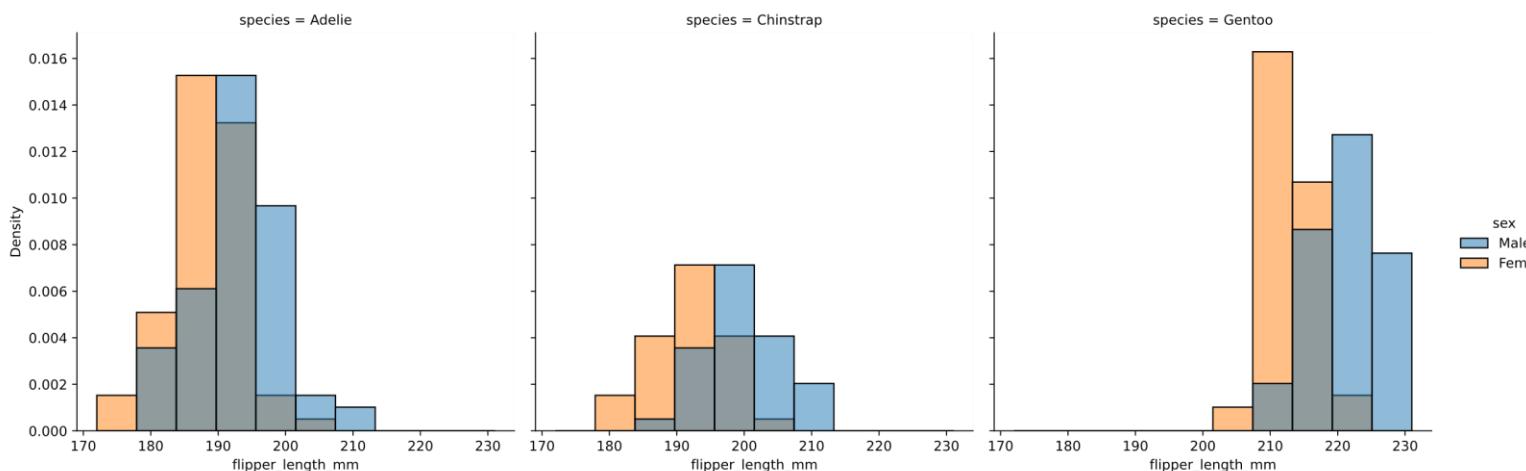
```
sns.kdeplot(**pkw, ax=ax[3])
sns.rugplot(**pkw, ax=ax[3])
```



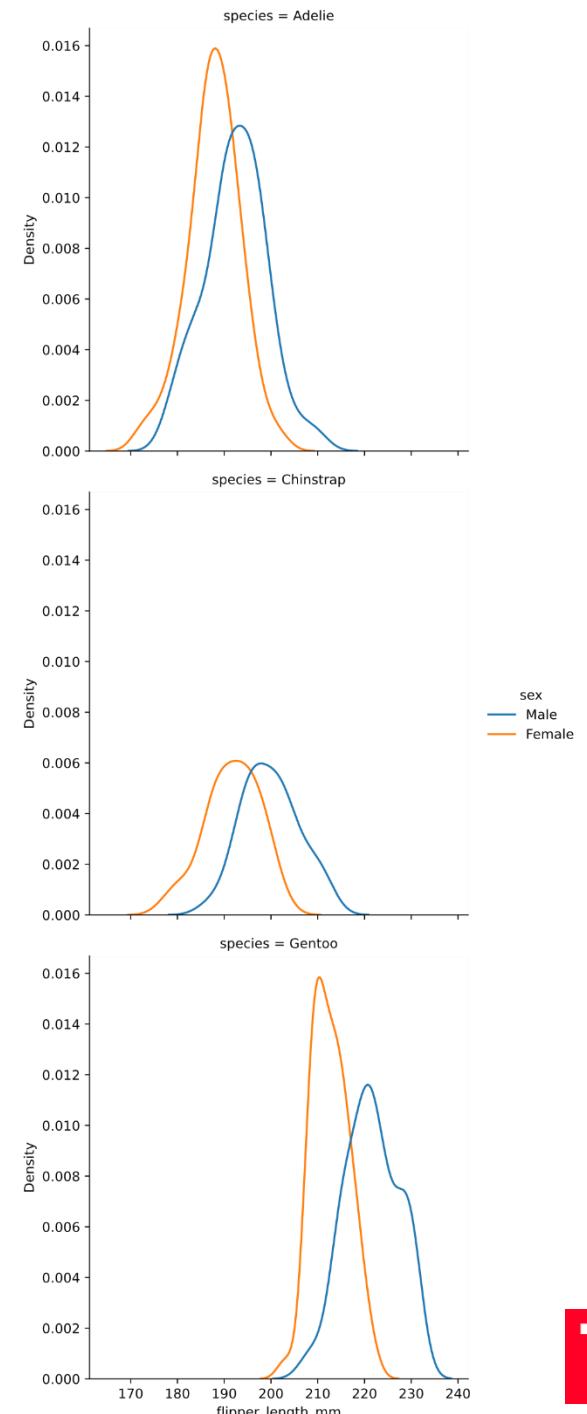
Vizualizace rozložení dat – displot

- Figure-level funkce pro zobrazení distribučních grafů v rámci pravidelné mřížky FacetGrid (viz dále).

```
sns.displot(data=penguins, kind="hist", x="flipper_length_mm",
             hue="sex", col="species", bins=10, stat="density")
```



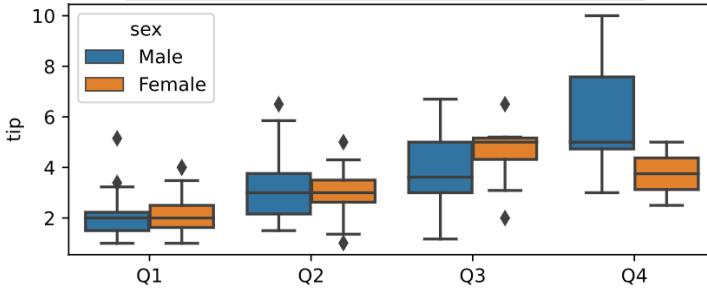
```
sns.displot(data=penguins, kind="kde", x="flipper_length_mm",
             hue="sex", col="species", col_wrap=1)
```



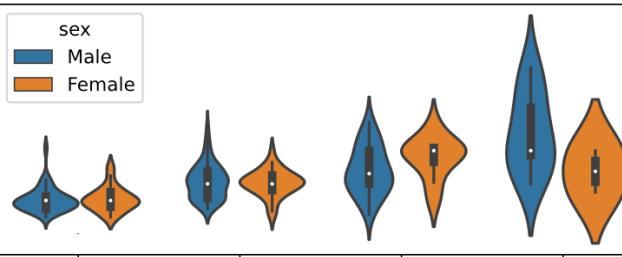
Vizualizace kategorických dat

```
tips["bill_class"] = pd.cut(tips["total_bill"], 4, labels=["Q1", "Q2", "Q3", "Q4"])
pkw = {"data": tips, "x": "bill_class", "y": "tip", "hue": "sex"}
```

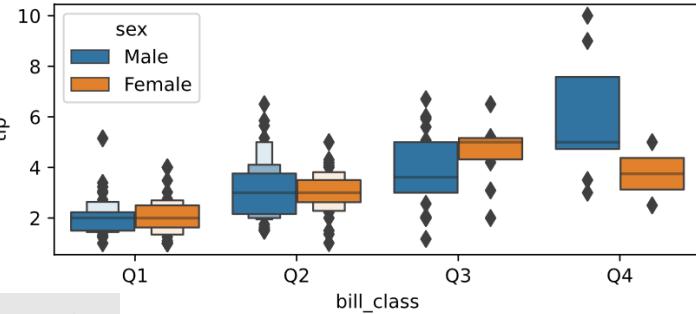
sns.boxplot(**pkw)



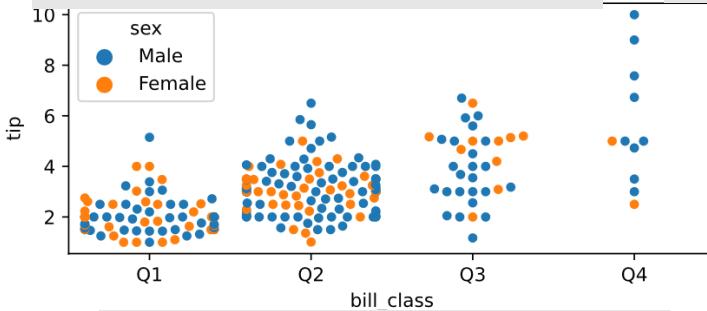
sns.violinplot(**pkw)



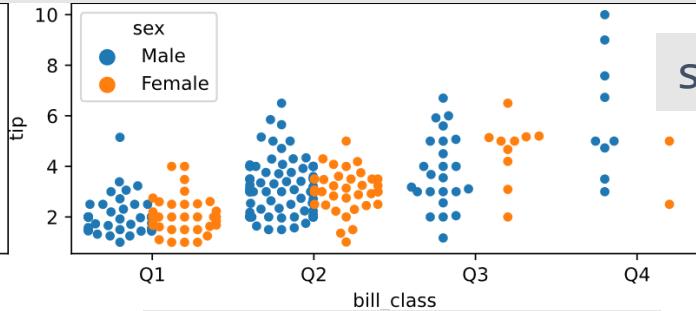
sns.boxenplot(**pkw)



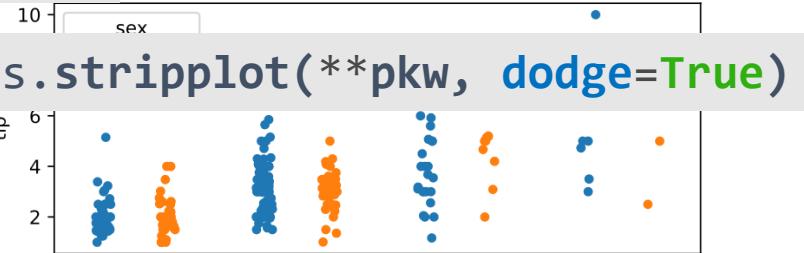
sns.swarmplot(**pkw)



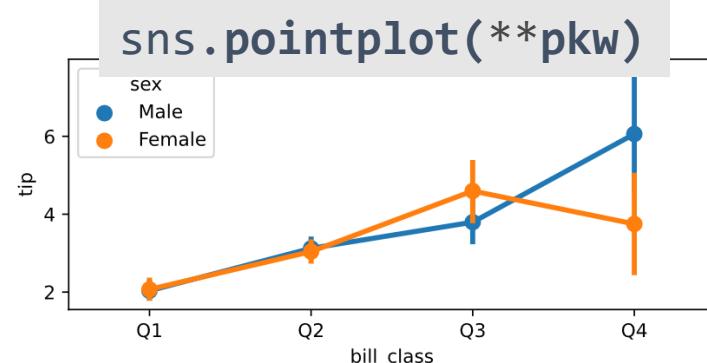
sns.swarmplot(**pkw, dodge=True)



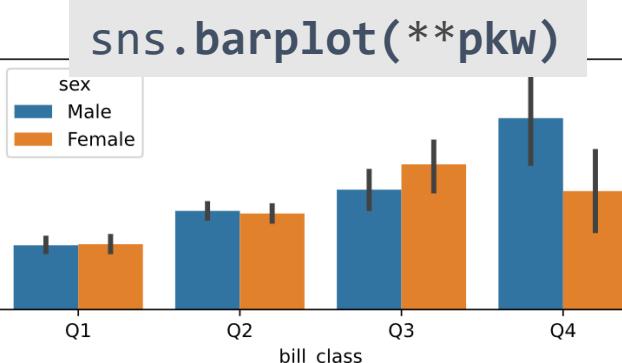
sns.stripplot(**pkw, dodge=True)



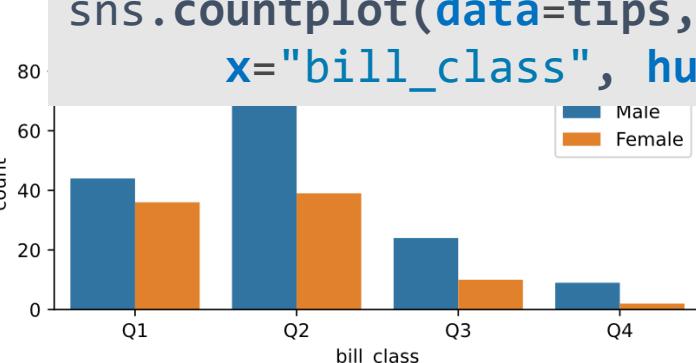
sns.pointplot(**pkw)



sns.barplot(**pkw)



sns.countplot(data=tips, x="bill_class", hue="sex")

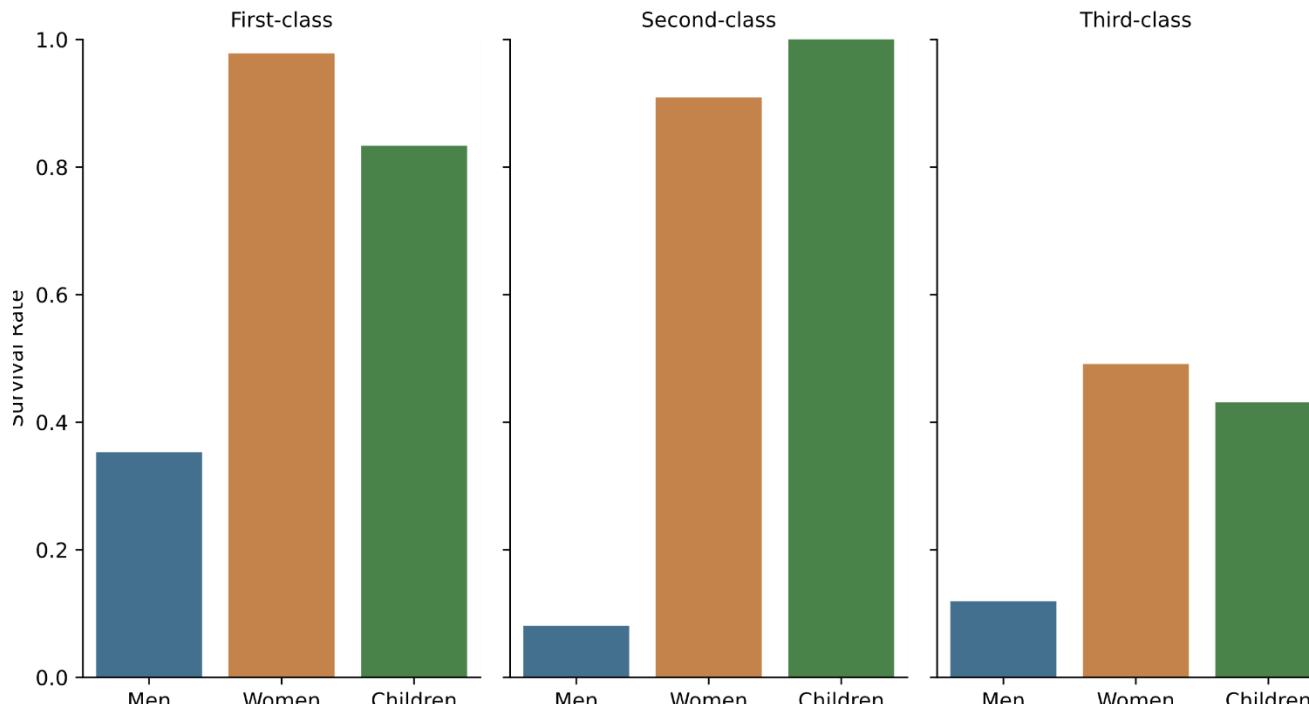


Vizualizace kategorických dat – catplot

■ Figure-level funkce

- argument kind určuje typ grafu

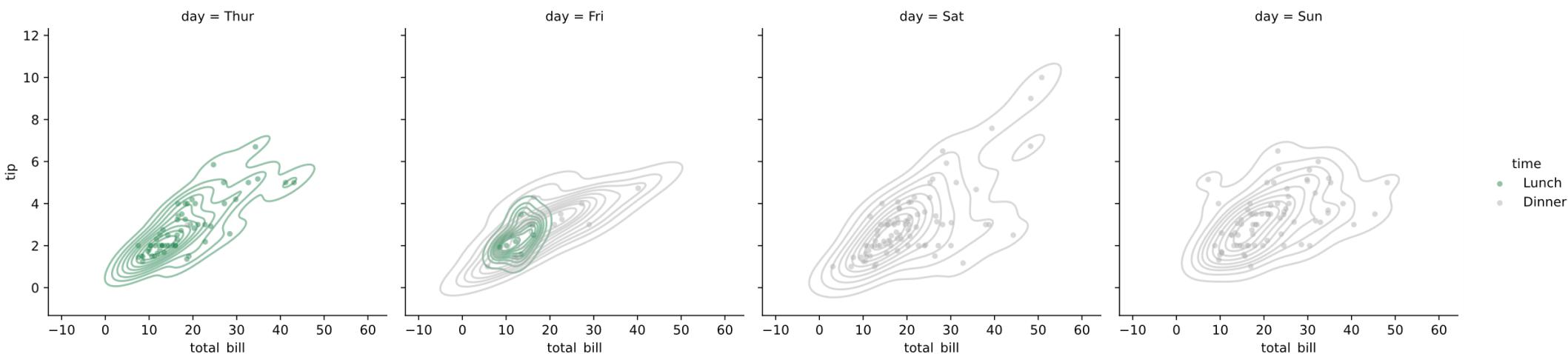
```
g = sns.catplot(x="who", y="survived", col="class",
                 data=titanic, saturation=.5,
                 kind="bar", ci=None, aspect=.6)
(g.set_axis_labels("", "Survival Rate")
 .set_xticklabels(["Men", "Women", "Children"])
 .set_titles("{col_name}-{col_var}")
 .set(ylim=(0, 1)))
```



Objekt FacetGrid

- Umožňuje vizualizaci vztahů v datech formou pravidelné mřížky, kde každá buňka je typicky počítána z určité podmnožiny řádků DataFrame

```
pal = dict(Lunch="seagreen", Dinner=".7")
g = sns.FacetGrid(tips, hue="time", col="day", palette=pal, height=4)
g.map(sns.scatterplot, "total_bill", "tip", s=20, alpha=.5)
g.add_legend()
g.map(sns.kdeplot, "total_bill", "tip", alpha=.5)
```



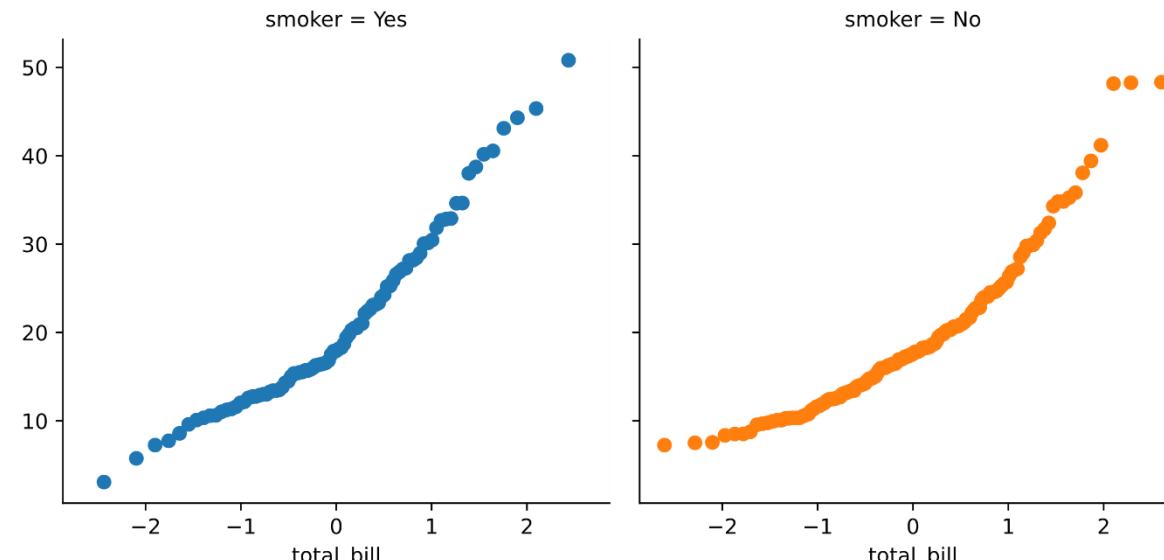
- Je návratovou hodnotou figure-level funkcí relplot, catplot a displot

FacetGrid – vlastní funkce pro vizualizaci

- S využitím metody `map` jsme schopni definovat vlastní vizualizační funkci

```
from scipy import stats
def qqplot(x, **kwargs):
    xr, yr = stats.probplot(x, fit=False)
    plt.scatter(xr, yr, **kwargs)

g = sns.FacetGrid(tips, col="smoker", hue="smoker", height=4)
g.map(qqplot, "total_bill")
```



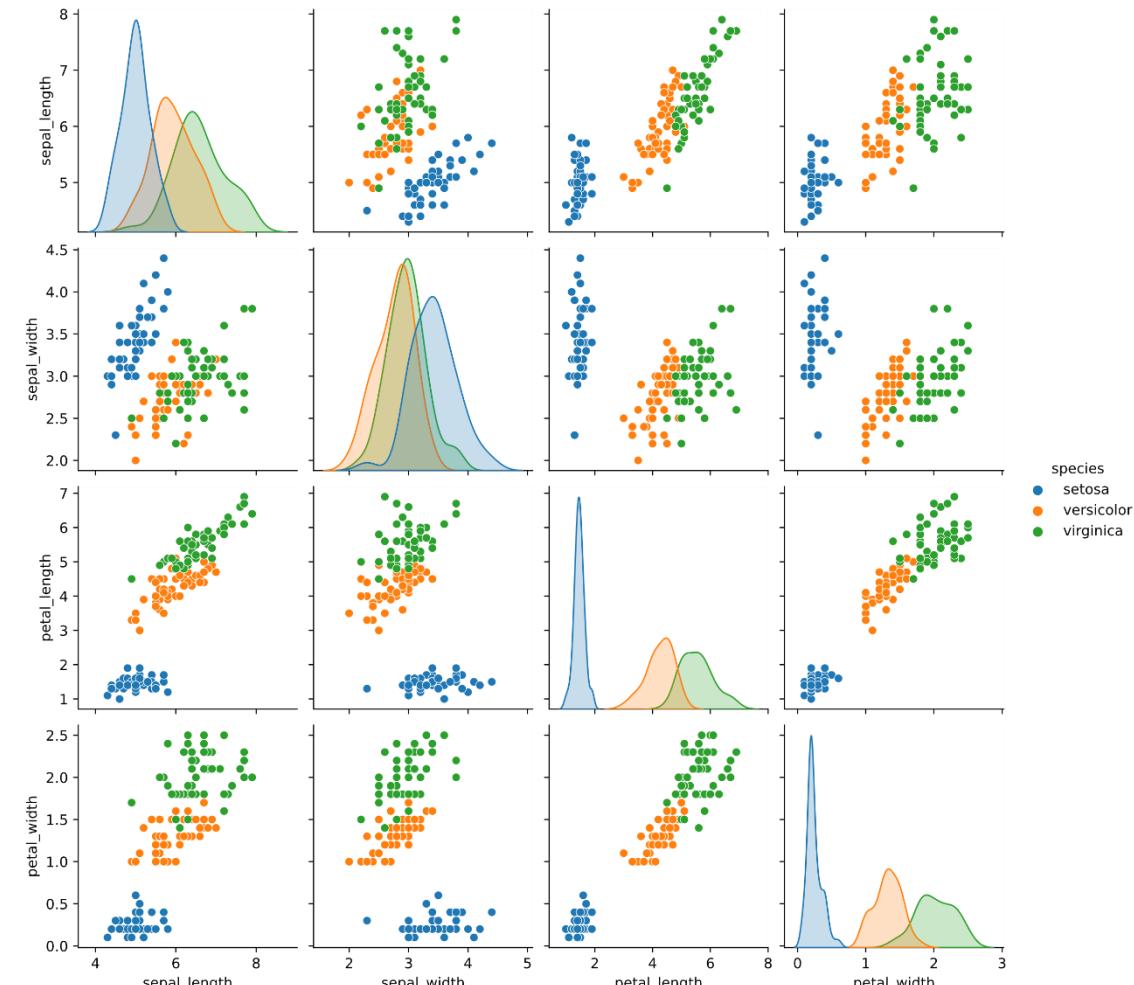
Vizualizace vztahu – párový graf (pairplot)

- Sloupce v DataFrame odpovídají jednotlivým řádkům a sloupcům matice grafu
 - Není-li pře definováno, je výchozím stavu na diagonále histogram

```
g = sns.pairplot(iris, hue="species")
```

- Pokud nechceme zobrazit všechny sloupce, musíme omezit dataset

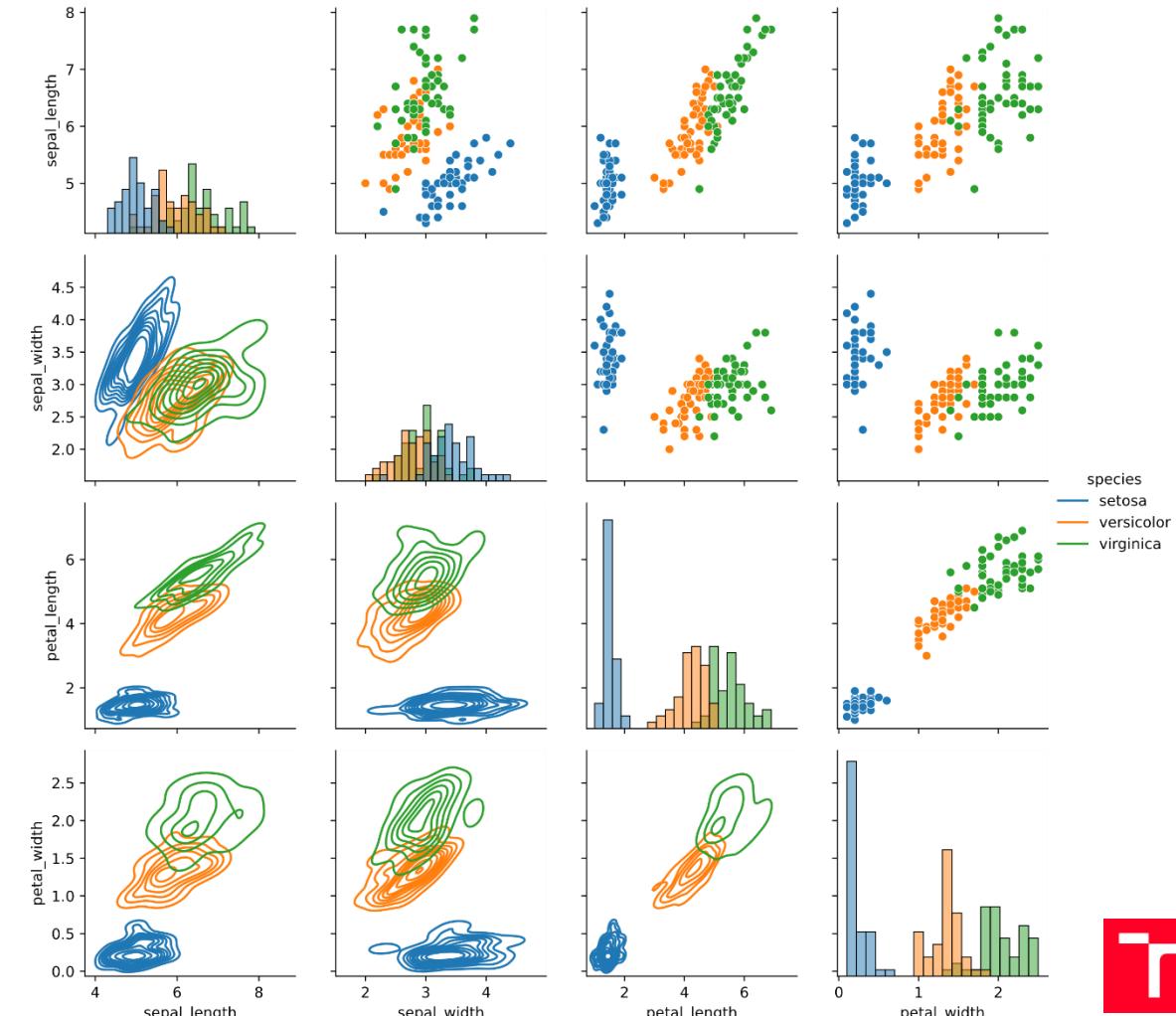
```
g = sns.pairplot(
    iris[['sepal_length',
          'petal_length',
          'species']],
    hue="species")
```



Vizualizace vztahu – PairGrid

- Předcházející funkce nevyužívá FacetGrid, ale speciální objekt PairGrid
 - lze namapovat vlastní zobrazovací funkce pro diagonálu, horní a spodní část

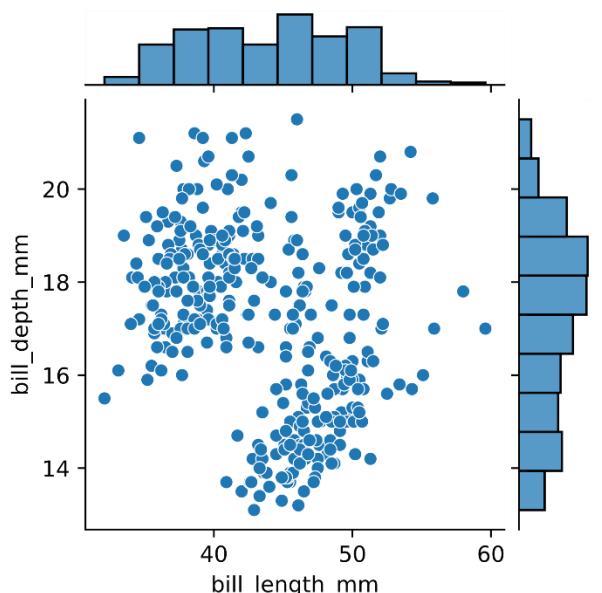
```
g = sns.PairGrid(iris, hue="species")
g.map_upper(sns.scatterplot)
g.map_diag(sns.histplot, bins=20)
g.map_lower(sns.kdeplot)
g.add_legend()
```



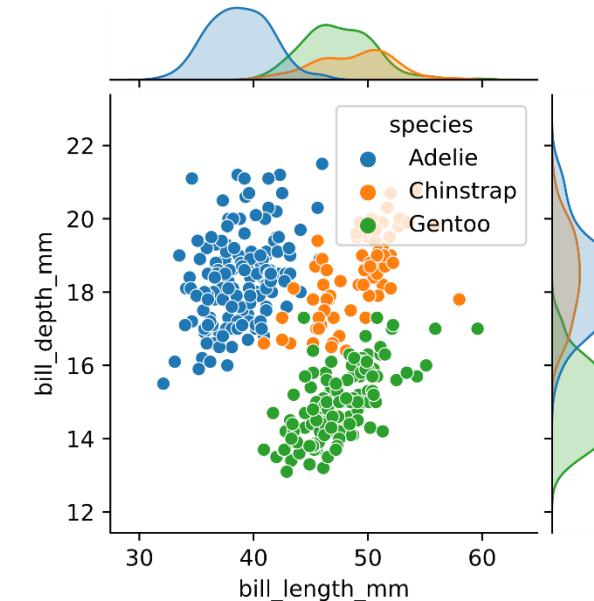
Vizualizace vztahu – jointplot a JointGrid

- Pro vizualizaci korelace mezi dvěma nezávislými proměnnými. Existuje i varianta, kde si můžeme určit funkce sami (JointGrid objekt)

```
sns.jointplot(data=penguins,
    x="bill_length_mm", y="bill_depth_mm",
    height=4)
```



```
sns.jointplot(data=penguins,
    x="bill_length_mm", y="bill_depth_mm",
    height=4, hue="species")
```



```
g = sns.JointGrid(data=penguins, x="bill_length_mm", y="bill_depth_mm")
g.plot_joint(sns.scatterplot, s=100, alpha=.5)
g.plot_marginals(sns.histplot, kde=True)
```



Teplotní mapa – heat map

- Vizualizace 2D závislosti vyžadující jako vstup 2D tabulku, jejíž obsah je reprezentován pomocí barevné čkály

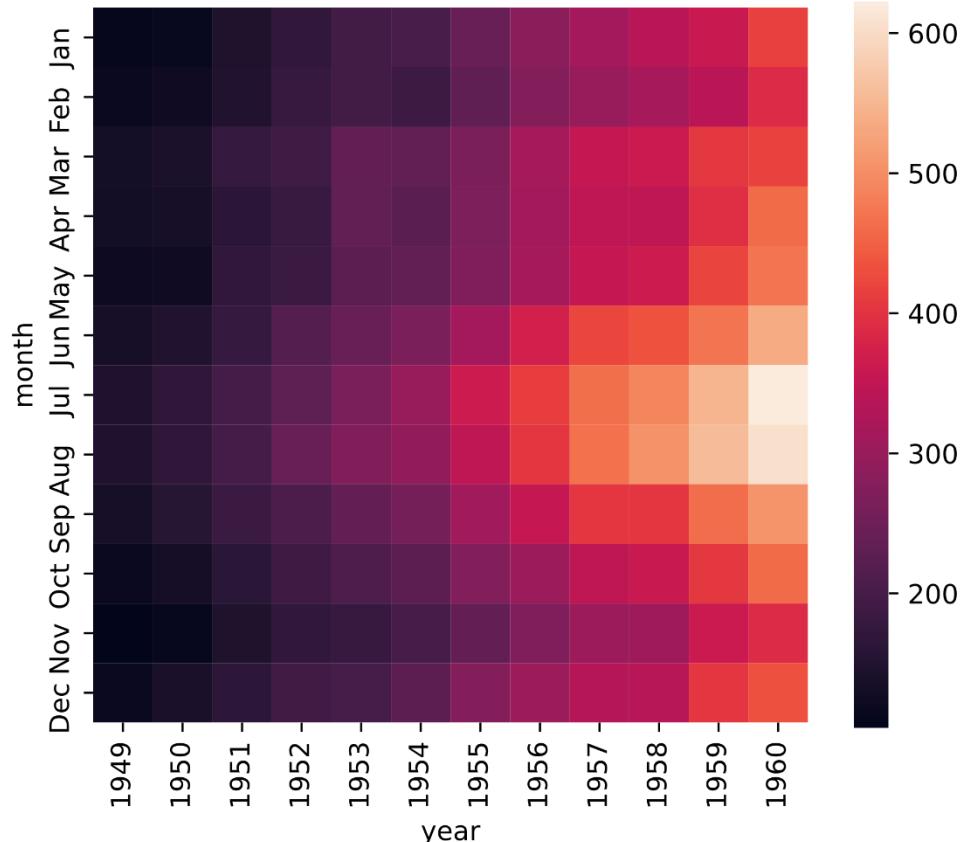
```
flights = flights.pivot("month", "year", "passengers")
```

year	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
month												
Jan	112	115	145	171	196	204	242	284	315	340	360	417
Feb	118	126	150	180	196	188	233	277	301	318	342	391
Mar	132	141	178	193	236	235	267	317	356	362	406	419
Apr	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
Jun	135	149	178	218	243	264	315	374	422	435	472	535
Jul	148	170	199	230	264	302	364	413	465	491	548	622
Aug	148	170	199	242	272	293	347	405	467	505	559	606
Sep	136	158	184	209	237	259	312	355	404	404	463	508
Oct	119	133	162	191	211	229	274	306	347	359	407	461
Nov	104	114	146	172	180	203	237	271	305	310	362	390
Dec	118	140	166	194	201	229	278	306	336	337	405	432

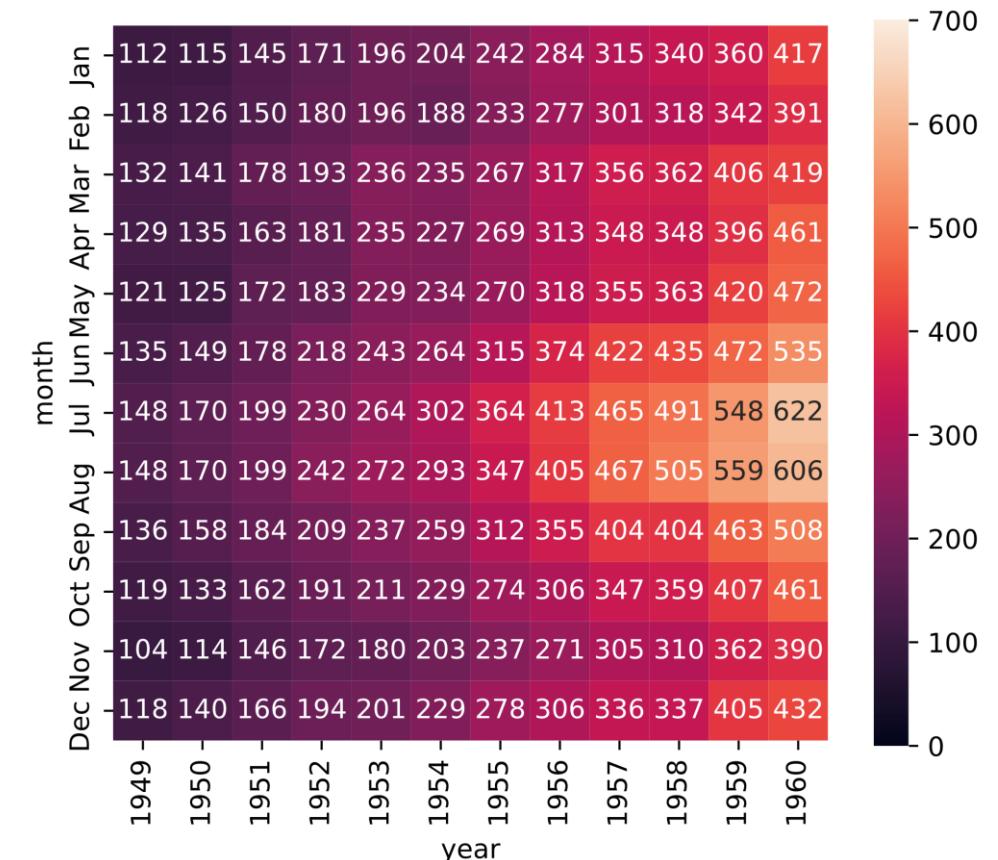


Heat map

```
ax = sns.heatmap(flights, square=True)
```

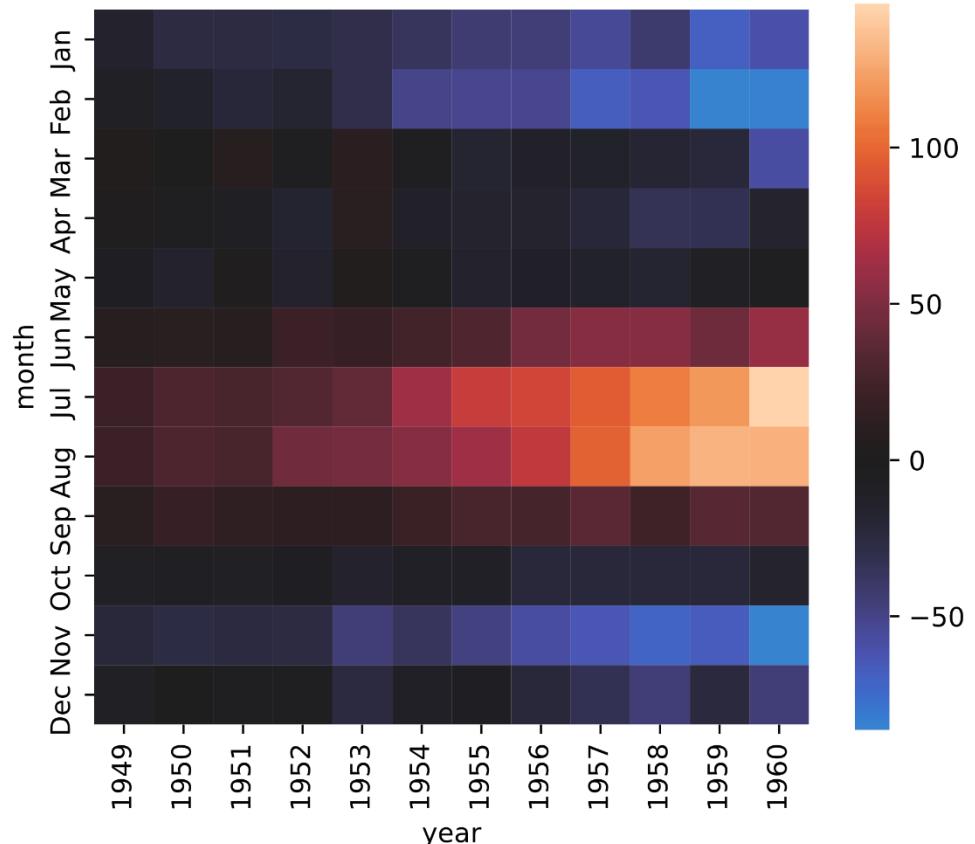


```
ax = sns.heatmap(flights, square=True,  
                 vmin=0, vmax=700,  
                 annot=True, fmt="d")
```

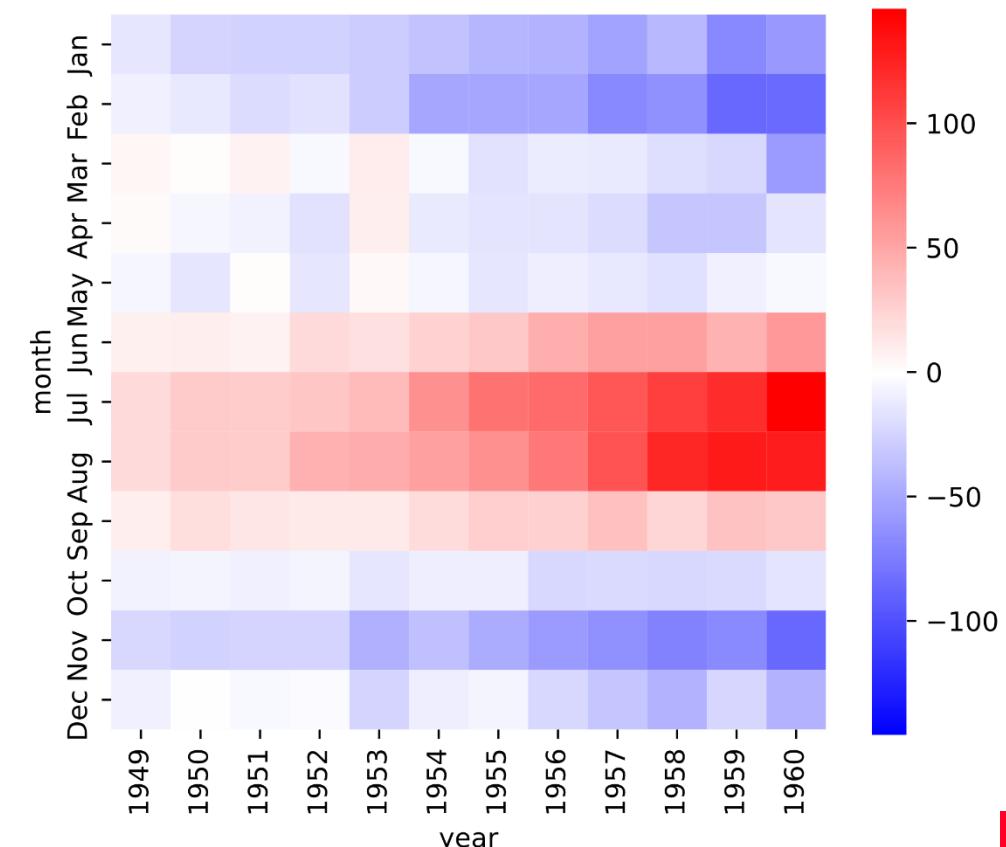


Heat map – divergentní paleta

```
flights_avg = flights - flights.mean()
ax = sns.heatmap(flights_avg, square=True,
                  center=0)
```



```
r = max(
    np.max(-flights_avg.min()),
    np.max(flights_avg.max()))
ax = sns.heatmap(flights_avg, square=True,
                  center=0, cmap="bwr", vmin=-r, vmax=r)
```



Modifikace vzhledu grafů

- Změna stylu

```
sns.set_style(x) # darkgrid, whitegrid, dark, white, ticks
```

```
display(sns.axes_style())
sns.set_style(x, rc)
```

```
with sns.axes_style("darkgrid"):
    sns.lineplot(data=flights2, ...)
```

- Změna stylu na základě cílového použití (kontextu)

```
sns.set_context(x) # paper, notebook, talk, poster
```

- Výběr palety (globálně)

```
sns.set_palette(x)
```

- Vše v jednom

```
sns.set_theme(context='notebook', style='darkgrid', palette='deep',
               font='sans-serif', font_scale=1, color_codes=True, rc=None)
# alternativně
sns.set(...)
```



Shrnutí

- Seaborn je poměrně komplexní knihovna pro vizualizaci statistických závislostí, která umožňuje pracovat s různými typy dat.
- Při tvorbě komplexnějších vizualizací je vhodné nahlédnout do [galerie](#)
- Pro vizualizaci musí mít data vhodný formát
 - typicky může dělat problém tzv. *wide-format* dat a je vhodné využít funkce knihovny Pandas pro převod.
- Vynechali jsme
 - regresní grafy (budou představeny později při regresní analýze).

