HAB619 Introduction to Scientific Computing in Sports Science

#13

De Motu Animalium G.Borelli (1680)

SERDAR ARITAN

serdar.aritan@hacettepe.edu.tr

Biyomekanik Araştırma Grubu
www.biomech.hacettepe.edu.tr
Spor Bilimleri Fakültesi
www.sbt.hacettepe.edu.tr
Hacettepe Universitesi, Ankara, Türkiye
www.hacettepe.edu.tr

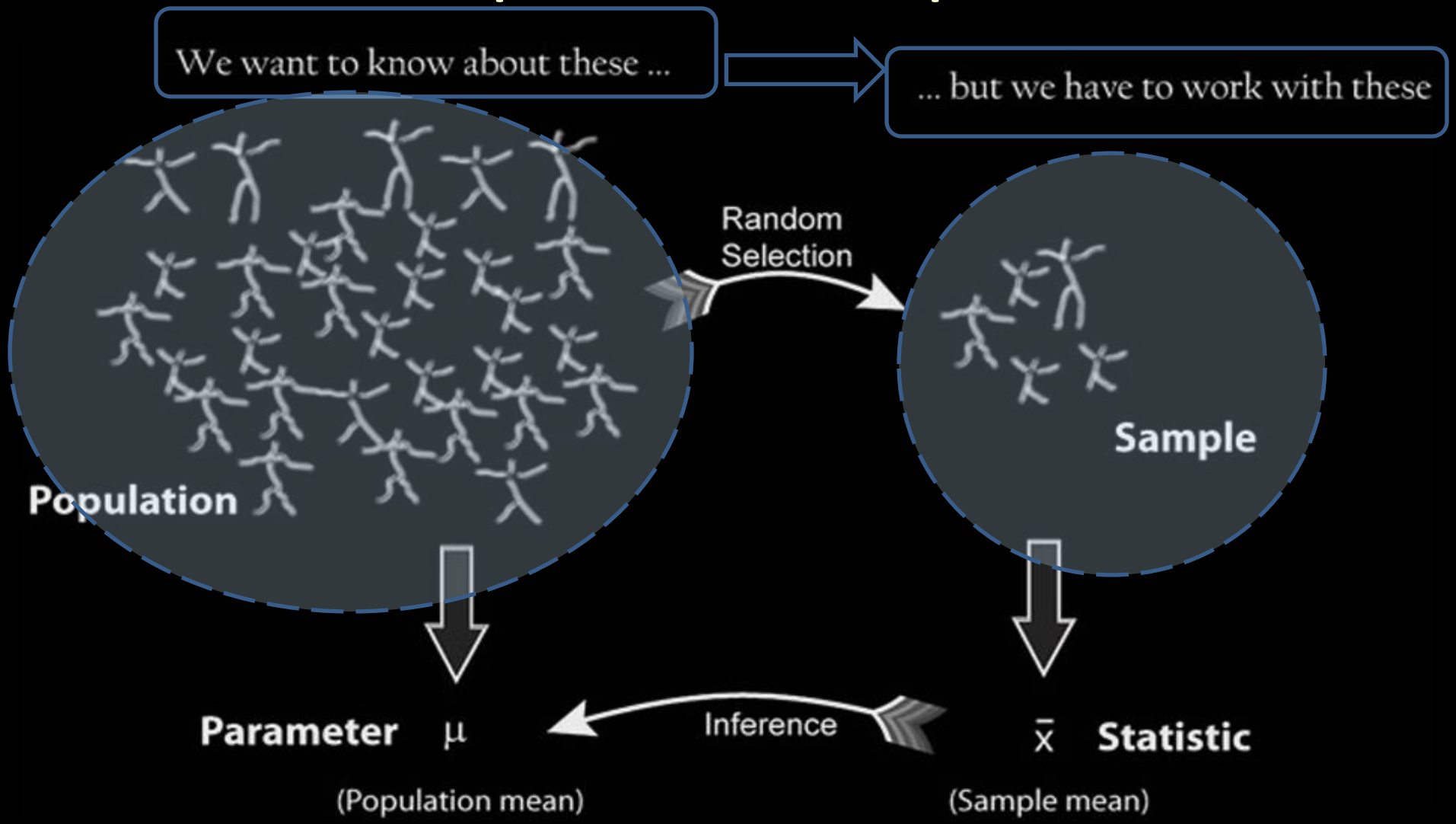
Populations and Samples

Population includes all of the elements from a set of data.

Sample consists of one or more observations from the population.

When estimating a parameter of a population, e.g., the expected value of the weight of male Europeans, we typically cannot measure all subjects. We have to limit ourselves to investigating a (hopefully representative) sample taken from this group. Based on the sample statistic, i.e., the corresponding value calculated from the sample data, we use statistical inference to draw conclusions about the corresponding parameter in the population.

Populations and Samples



Populations and Samples

Statistic

A numerical value that represents a property of a random sample. Examples of statistics are

- the mean value of the sample data.
- the range of the sample data.
- deviation of the data from the sample mean.

(Empirical) sampling distribution The probability distribution of a given statistic based on a random sample.

Statistical inference Enables you to make an educated guess about a population parameter based on a statistic computed from a representative sample from that population.

Populations and Samples


```
In [1]: import numpy as np
In [2]: x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
In [3]: np.mean(x)
Out[3]: 4.5
In [4]: xWithNaN = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, np.nan]
In [5]: np.mean(xWithNaN)
Out[5]: nan
In [6]: np.nanmean(xWithNaN)
Out[6]: 4.5
In [7]: mat = [[1, 2],
         [3, 4]]
In [8]: np.max(mat)
Out[9]: 4
In [10]: np.max(mat, axis=0)
Out[10]: array([3, 4])
In [11]: np.max(mat, axis=1)
Out[11]: array([2, 4])
```

Populations and Samples


```
In [12]: x = np.arange(1,101)
In [13]: stats.gmean(x)
Out[13]: 37.9927
```

Power Analyses

The determination of the power of a statistical test, and the calculation of the minimum sample size required to reveal an effect of a given magnitude, is called power analysis. It involves four factors:

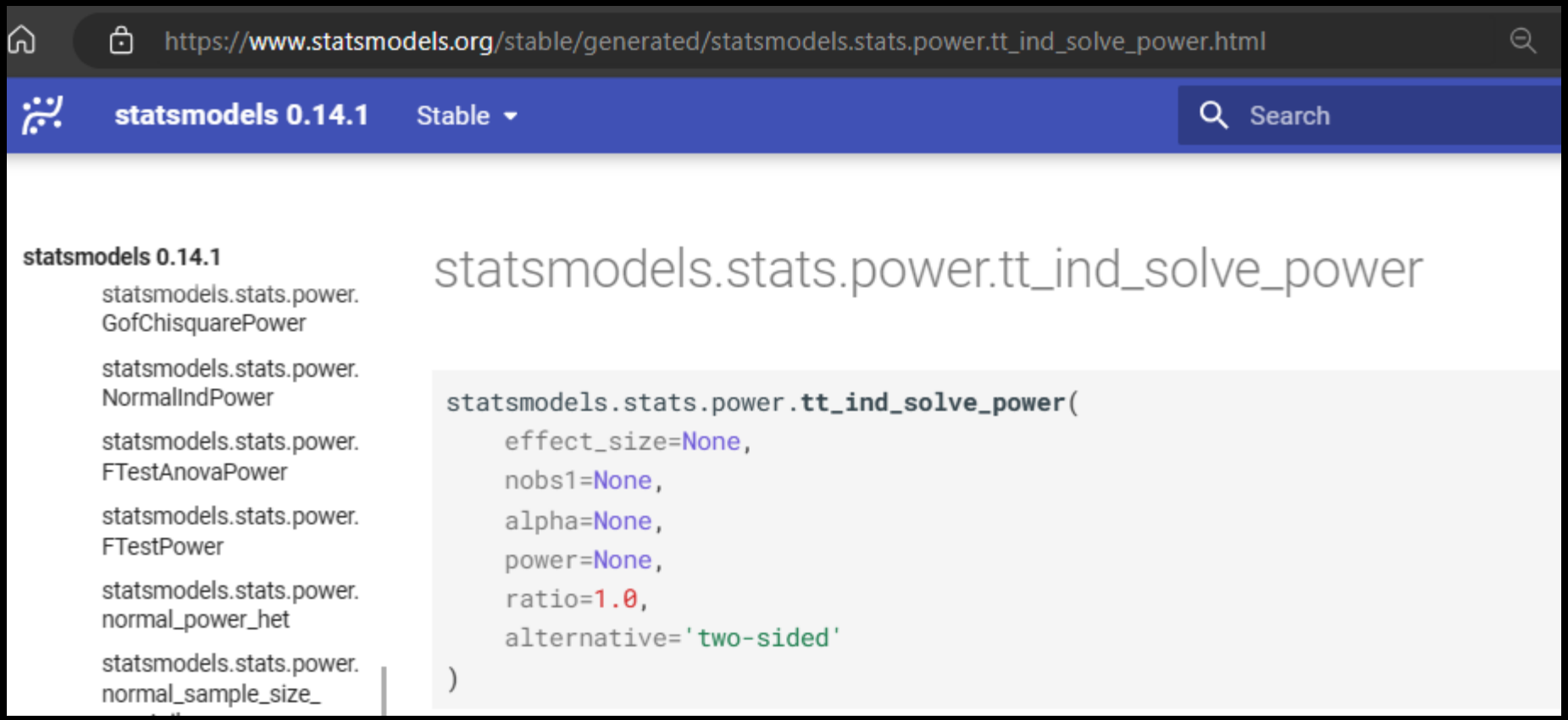
1. α, the probability for type I errors
2. β, the probability for type II errors (⇒ power of the test)
3. d, also called "Cohen's d": the effect size, i.e., the magnitude of the investigated

effect relative to σ, the standard deviation of the sample

4. n, the sample size.

Power Analyses

https://www.statsmodels.org/stable/generated/statsmodels.stats.power.tt_ind_solve_power.html



Parametric Statistics

```
import numpy as np
from statsmodels.stats import power

nobs = power.tt_ind_solve_power(effect_size=0.5,alpha=0.05, power=0.8)

print(np.ceil(nobs))
# 64
```

if we compare two groups with the same number of subjects and the same standard deviation, require α = 0.05 and a test power of 80%, and we want to detect a difference between the groups that is half the standard deviation, we need to test 64 subjects.

Parametric Statistics

```
import numpy as np
from statsmodels.stats import power

effect_size = power.tt_ind_solve_power(alpha=0.05,power=0.8, nobs1=25)

print(np.round(effect_size, decimals=2))
# 0.81
```

if we have α = 0.05, a test power of 80%, and 25 subjects in each group, then the smallest significant difference between the groups is 81% of the sample standard deviation.

Populations and Samples

in order to run a parametric analysis, parametric assumptions (e.g., a normality test on the data) have to be ascertained. To check for normality, we can look at the histograms and we can inspect their shape using the **`numpy.histogram()`** function. The **`scipy.stats.kurtosis()`** function returns the kurtosis of the distribution of the input vector. The **`scipy.stats.skew()`** function returns a measure of the asymmetry. To test whether the data are normally distributed, the **`scipy.stats.kurtosistest()`** whether a dataset has normal kurtosis.

Parametric Statistics

ANOVA

The most common application of the F-distribution is the comparison of three or more groups. In that case, the variation between the groups is compared to the variation within the groups. Appropriately, this is called analysis of variance (ANOVA).

An ANOVA compares the size of the variance between two different samples.

Parametric Statistics

ANOVA

For example, a company that produces hip implants, and changes from an old system to a new system. For the old and the new process, we obtain samples with the following femur head diameters (in mm):

An ANOVA compares the size of the variance between two different samples.

```
Old Method: [29.7, 29.4, 30.1, 28.6, 28.8, 30.2, 28.7, 29. ]

New Method: [30.7, 30.3, 30.3, 30.3, 30.7, 29.9, 29.9, 29.9,
 30.3, 30.3, 29.7, 30.3]
```

Parametric Statistics

To compare if the precision of the new system is as good as the old one is used to compare the two variances. A value of F = 1 would indicate that both precisions are equivalent; if the new method were more precise, F would be smaller than 1; and if the new method were less precise, F would be larger than 1.

If the F-value is within the 95%-confidence interval, it would mean that the two methods are not significantly different.

Parametric Statistics

```
import numpy as np
from scipy import stats
old = [29.7, 29.4, 30.1, 28.6, 28.8, 30.2, 28.7, 29.]
new = [30.7, 30.3, 30.3, 30.3, 30.7, 29.9, 29.9, 29.9, 30.3,
30.3, 29.7, 30.3]

f_val = np.var(new, ddof=1)/np.var(old, ddof=1) #-> F=0.244
fd = stats.f(len(new)-1, len(old)-1)
p = fd.cdf(f_val)
#-> p=0.019

if (0.025 < p < 0.975):
        print('No significant difference.')
else:
        print('There is a significant difference ' + \
                'between the two distributions.')
```

Parametric Statistics

Two-Sample t -Test

The **`scipy.stats`** includes two functions performing the two-sample t -test: (1) **`ttest_rel`** for the paired, or repeated measure, t -test, and (2) **`ttest_ind`** for the unrelated, or independent measure, t -test.

Let us suppose that you are interested in finding out whether Reaction Times decrease when the color of a probe is red instead of yellow at an alpha level of 0.01. Ten participants are tested, and RTs, in milliseconds, are collected.

Participant id	1	2	3	4	5	6	7	8	9	10
Yellow	300	287	301	400	211	399	412	312	390	412
Red	240	259	302	311	210	402	390	298	347	380

Parametric Statistics

```
yellow = np.array([300,287,301,400,211,399,412,312,390, 412])
red = np.array([240,259,302,311,210,402,390,298,347,380])

t_rel, p_rel = stats.ttest_rel(yellow,red, alternative='greater')
```

we are expecting that RTs in the "red" condition should be faster than in the "yellow" condition. If we pass the vector "yellow" as a first argument, then we are expecting that the difference between the means should be larger than 0. Hence, the right argument has to be passed to the function.

Parametric Statistics

```
Paired t-test with 10 samples yields
t-value of 3.0719 and p-value of 0.0067
```

the probability of finding these results by random chance is equal to 0.0067 (p).

Parametric Statistics

If there were two different groups of participants, that is, one group was tested with the yellow probe only and the other group with the red probe only, then we would need **ttest_ind** as follows:

```
t_ind,p_ind =stats.ttest_ind(yellow,red, alternative='greater')
```

```
Independent t-test with 10 samples yields
t-value of 0.9446 and p-value of 0.1787
```

By looking at the p value, we find out that the probability of obtaining these results by chance with independent samples is 17.87%.

Parametric Statistics

When groups have the same size, the group vector can be implemented directly in the function call to increase readability in the following way:

```
f_oneway (group1' group2' group3' …)
```

Returns:

1. The computed F statistic of the test.
2. The associated p-value from the F distribution.

Parametric Statistics

Imagine that in that experiment, a third group is tested in which participants' RTs are measured when presented with a black probe.

Yellow	300	287	301	400	211	399	412	312	390	412
Red	240	259	302	311	210	402	390	298	347	380
Black	210	230	213	210	220	208	290	300	201	201

Since the number of participants per group is the same, f_oneway can be used in both its ways.

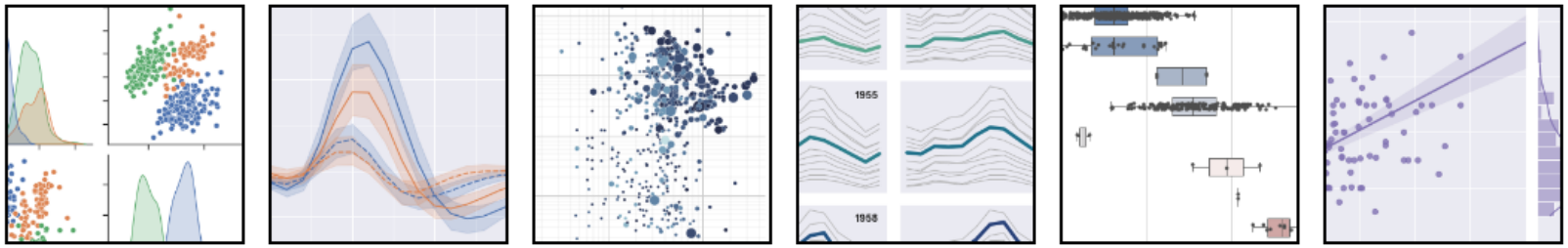
Parametric Statistics

```
from scipy.stats import f_oneway

yellow = [300, 287, 301, 400, 211, 399, 412, 312, 390, 412]
red = [240, 259, 302, 311, 210, 402, 390, 298, 347, 380]
black = [210, 230, 213, 210, 220, 208, 290, 300, 201, 201]

F, pValue = f_oneway(yellow, red, black)

print(f"Computed F statistic is {F:.2f}\n\
p-value from the F distribution is {pValue:.4f}")
```

```
Computed F statistic is 10.15
p-value from the F distribution is 0.0005
```

seaborn Installing Gallery Tutorial API Releases Citing FAQ

seaborn: statistical data visualization



Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the introductory notes or the paper. Visit the installation page to see how you can download the package and get started with it. You can browse the example gallery to see some of the things that you can do with seaborn, and then check out the tutorials or API reference to find out how.

To see the code or report a bug, please visit the GitHub repository. General support questions are most at home on stackoverflow, which has a dedicated channel for seaborn.

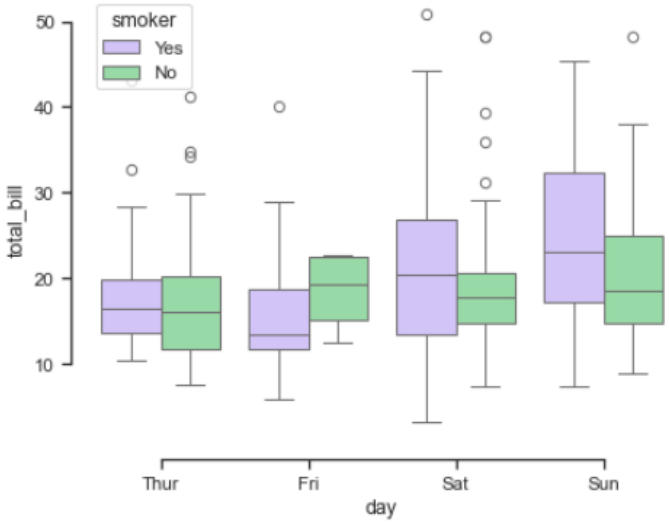
Contents

Features

- New Objects: API | Tutorial
- Relational plots: API | Tutorial
- Distribution plots: API | Tutorial
- Categorical plots: API | Tutorial
- Regression plots: API | Tutorial
- Multi-plot grids: API | Tutorial
- Figure theming: API | Tutorial
- Color palettes: API | Tutorial

https://seaborn.pydata.org/index.html

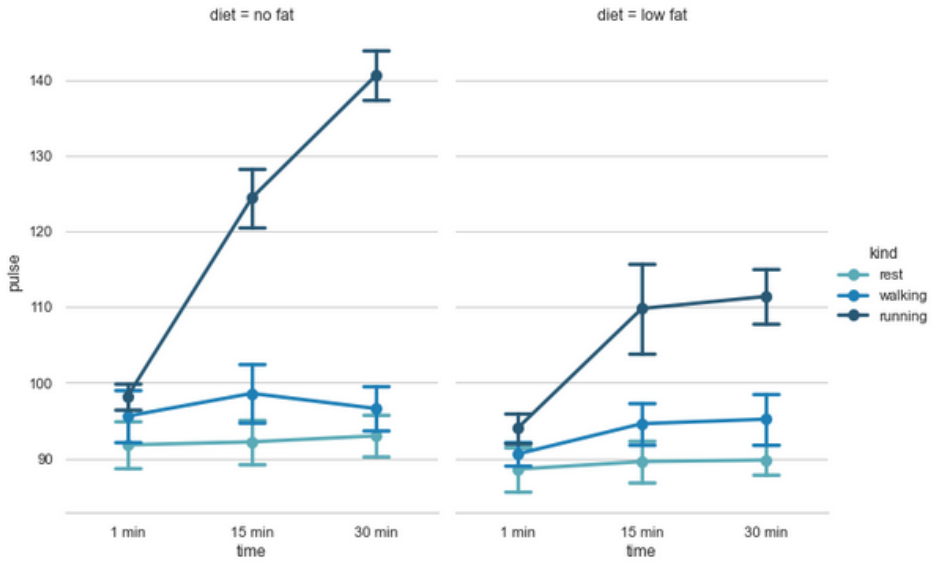
Grouped boxplots



seaborn components used: `set_theme()`, `load_dataset()`, `boxplot()`,

```
import seaborn as sns
sns.set_theme(style="ticks", palette="pastel")

# Load the example tips dataset
tips = sns.load_dataset("tips")

# Draw a nested boxplot to show bills by day and time
sns.boxplot(x="day", y="total_bill",
            hue="smoker", palette=["m", "g"],
            data=tips)
sns.despine(offset=10, trim=True)
```

Plotting a three-way ANOVA



seaborn components used: `set_theme()`, `load_dataset()`, `catplot()`

```
import seaborn as sns
sns.set_theme(style="whitegrid")

# Load the example exercise dataset
exercise = sns.load_dataset("exercise")

# Draw a pointplot to show pulse as a function of three categorical factors
g = sns.catplot(
    data=exercise, x="time", y="pulse", hue="kind", col="diet",
    capsize=.2, palette="YlGnBu_d", errorbar="se",
    kind="point", height=6, aspect=.75,
)
g.despine(left=True)
```