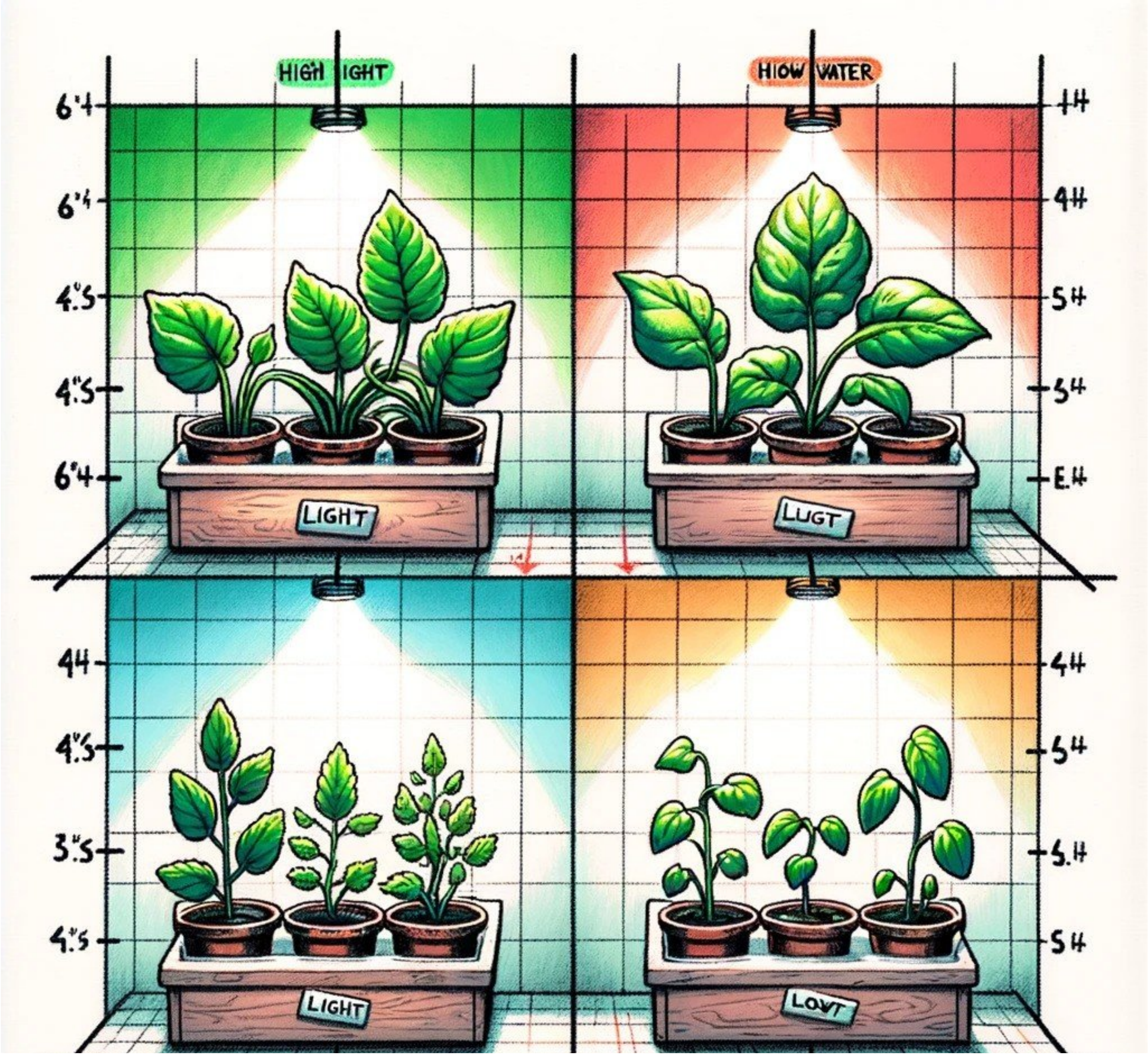
Factorial designs: principles and applications

EXPERIMENTAL DESIGN IN PYTHON

James Chapman
Curriculum Manager, DataCamp

Understanding factorial design

- Study *multiple independent variables*/**factors** in one experiment
- Test *every combination* of factor levels
- Discover direct effects and *interactions* between factors



[1] Image Generated with DALL·E 3

Factorial design data example

- Factor 1 (`Light_Condition`) - two levels: `Full Sunlight` and `Partial Shade`
- Factor 2 (`Fertilizer_Type`) - two levels: `Synthetic` and `Organic`
- Numeric response/dependent/outcome variable: `Growth_cm`

```
plant_growth_data.head()
```

```
   Plant_ID  Light_Condition  Fertilizer_Type  Growth_cm
0         1    Full Sunlight        Synthetic  16.489735
1         2    Partial Shade          Organic  18.361689
2         3    Full Sunlight        Synthetic  18.039459
3         4    Full Sunlight          Organic  12.682425
4         5    Full Sunlight          Organic  21.480601
```

Organizing data to visualize interactions

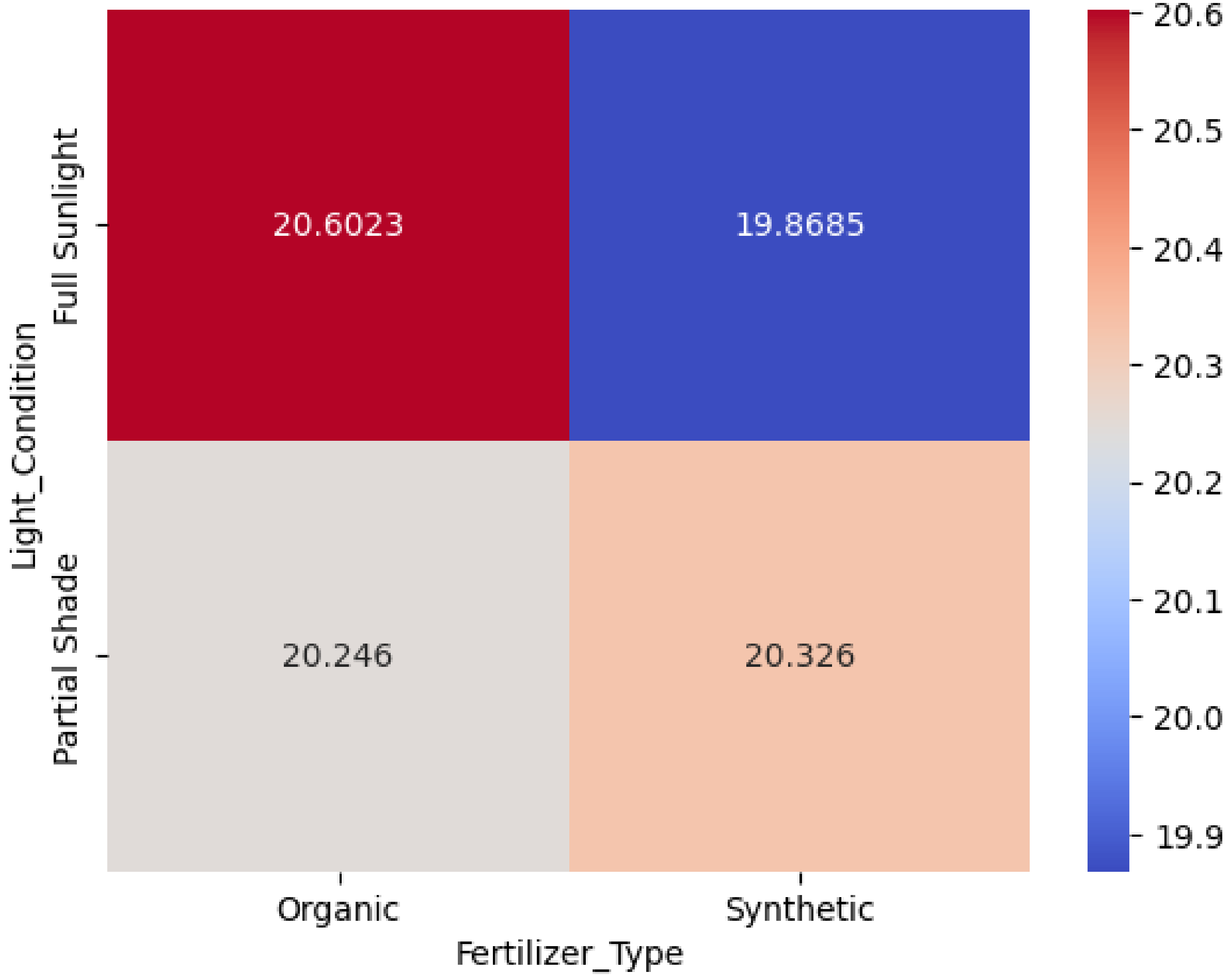
```
plant_growth = pd.pivot_table(plant_growth_data,
                              values='Growth_cm',
                              index='Light_Condition',
                              columns='Fertilizer_Type',
                              aggfunc='mean')

plant_growth
```

```
Light_Condition     Organic   Synthetic
Full Sunlight       20.602       19.869
Partial Shade       20.246       20.326
```

Visualize interactions with heatmap

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(plant_growth,
            annot=True,
            cmap='coolwarm',
            fmt='g')
plt.show()
```



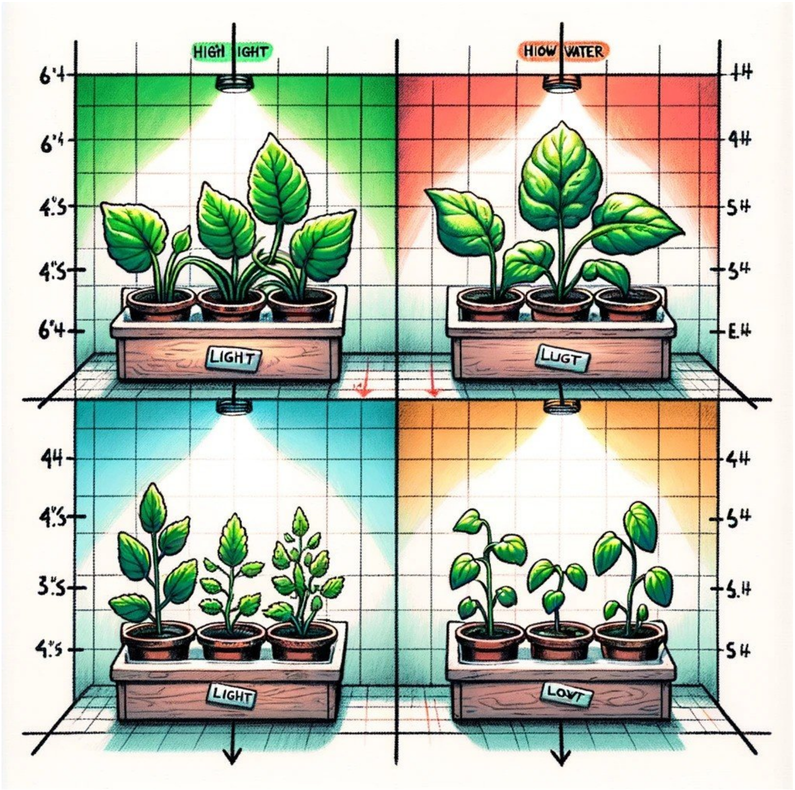
Interpreting interactions

```
Light_Condition     Organic   Synthetic
Full Sunlight       20.602       19.869
Partial Shade       20.246       20.326
```

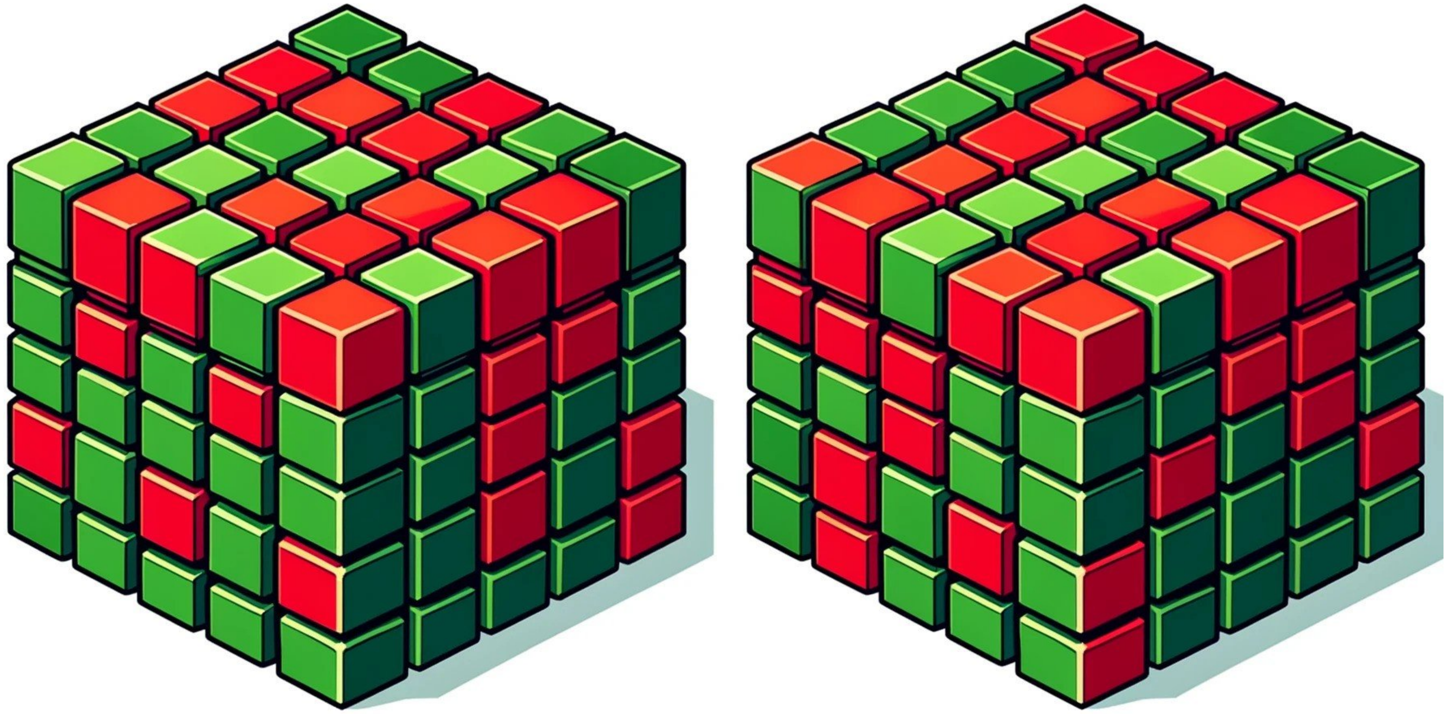
- **Interactions:** how the effect of one factor varies with the level of another factor
- Significant interaction → *factors do not work independently*

Factorial designs vs. randomized block designs

- Multiple treatments and interactions
- Dissect complex multi-variable effects and interactions
- Can require more subjects



- Group similar subjects in randomized designs
- Control within-block variance
- Each treatment is tested within every block



[1] Images Generated with DALL·E 3

Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

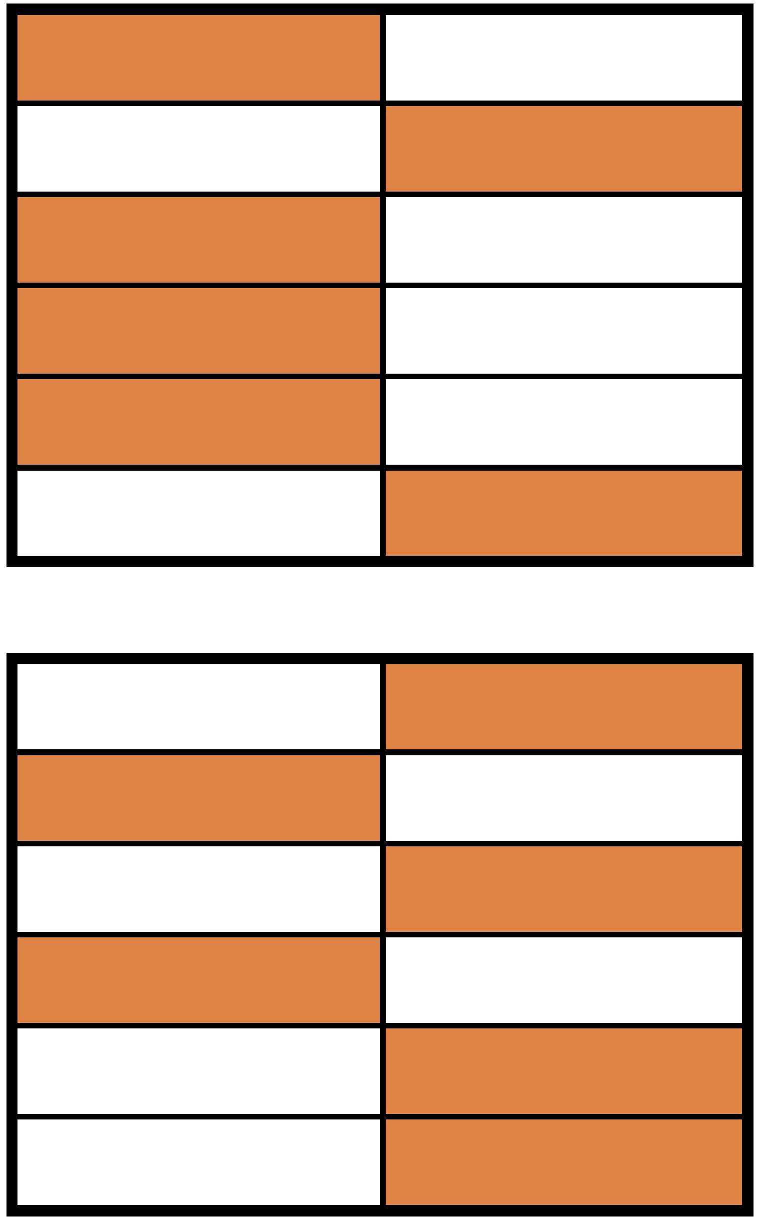
Randomized block design: controlling variance

EXPERIMENTAL DESIGN IN PYTHON

James Chapman
Curriculum Manager, DataCamp

Understanding blocking

- *Reduce variance* by grouping similar units
- Each block receives all treatments
- Focus on treatment effects, *controlling for block effects*



Block design data example

```
athletes.head()
```

```
   Athlete_ID Initial_Fitness_Level  Muscle_Gain_kg
0         113              Beginner        3.225102
1          30              Advanced        3.976548
2         183          Intermediate        5.165449
3         200              Beginner        2.188297
4         194              Beginner        4.724162
```

Implementing randomized block design

- Use `.groupby()` to shuffle within blocks

```
blocks = athletes.groupby('Initial_Fitness_Level').apply(
    lambda x: x.sample(frac=1)
)
blocks = blocks.reset_index(drop=True)
blocks
```

```
     Athlete_ID Initial_Fitness_Level  Muscle_Gain_kg
0           198              Advanced           5.742
1           146              Advanced           6.248
2           157              Advanced           6.049
..          ...                   ...             ...
198         164          Intermediate           6.134
199         178          Intermediate           6.591
```

Implemented randomized blocks

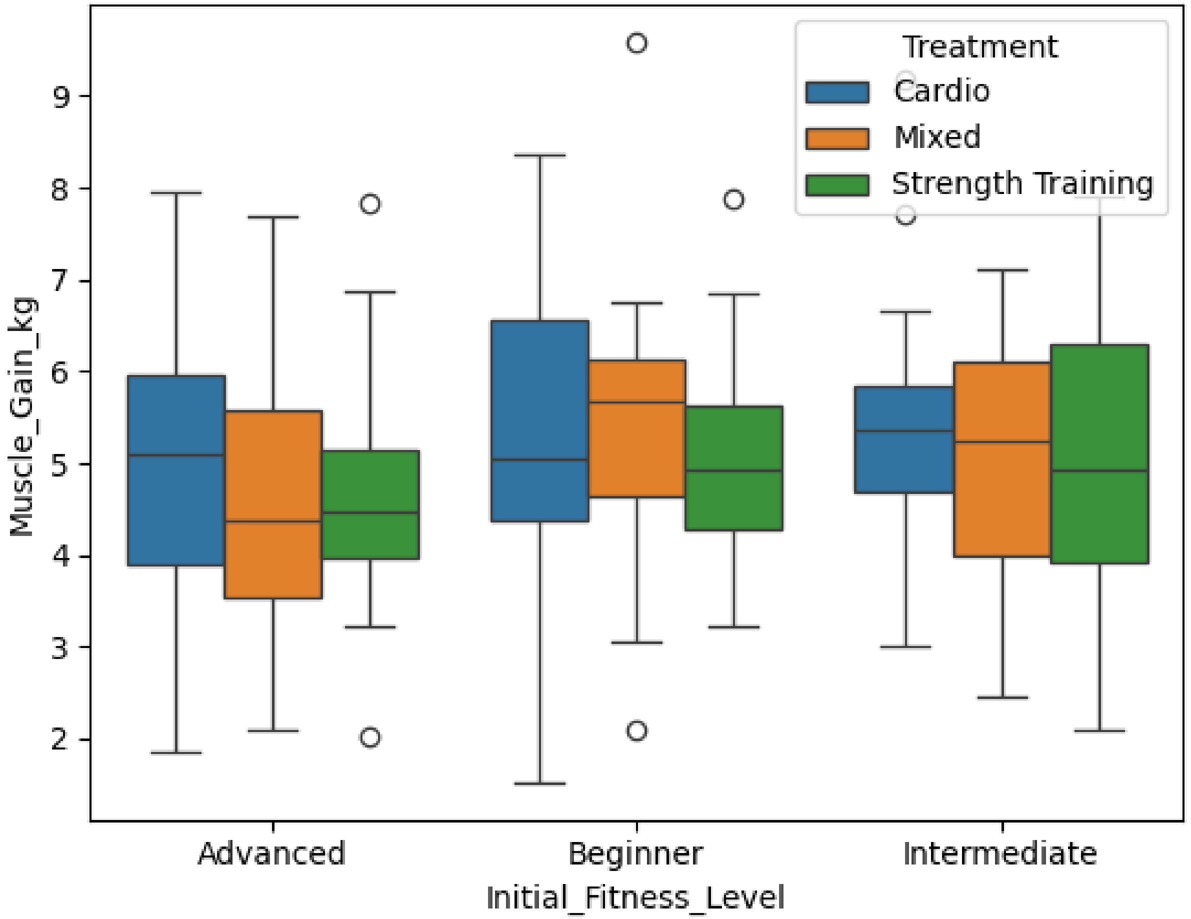
- `numpy.random.choice()` for random treatment assignment within blocks

```
blocks['Treatment'] = np.random.choice(
    ['Cardio', 'Strength Training', 'Mixed'],
    size=len(blocks))
blocks.sample(n=5)
```

```
     Athlete_ID  Initial_Fitness_Level  Muscle_Gain_kg          Treatment
 87         194               Beginner           4.724             Cardio
 54           3               Advanced           3.731  Strength Training
177          80           Intermediate           6.758              Mixed
146         183           Intermediate           5.165  Strength Training
 60         190               Advanced           3.763             Cardio
```

Visualizing treatment effects within blocks

```
import seaborn as sns
sns.boxplot(x='Initial_Fitness_Level', y='Muscle_Gain_kg', hue='Treatment', data=blocks)
plt.show()
```



ANOVA within blocks

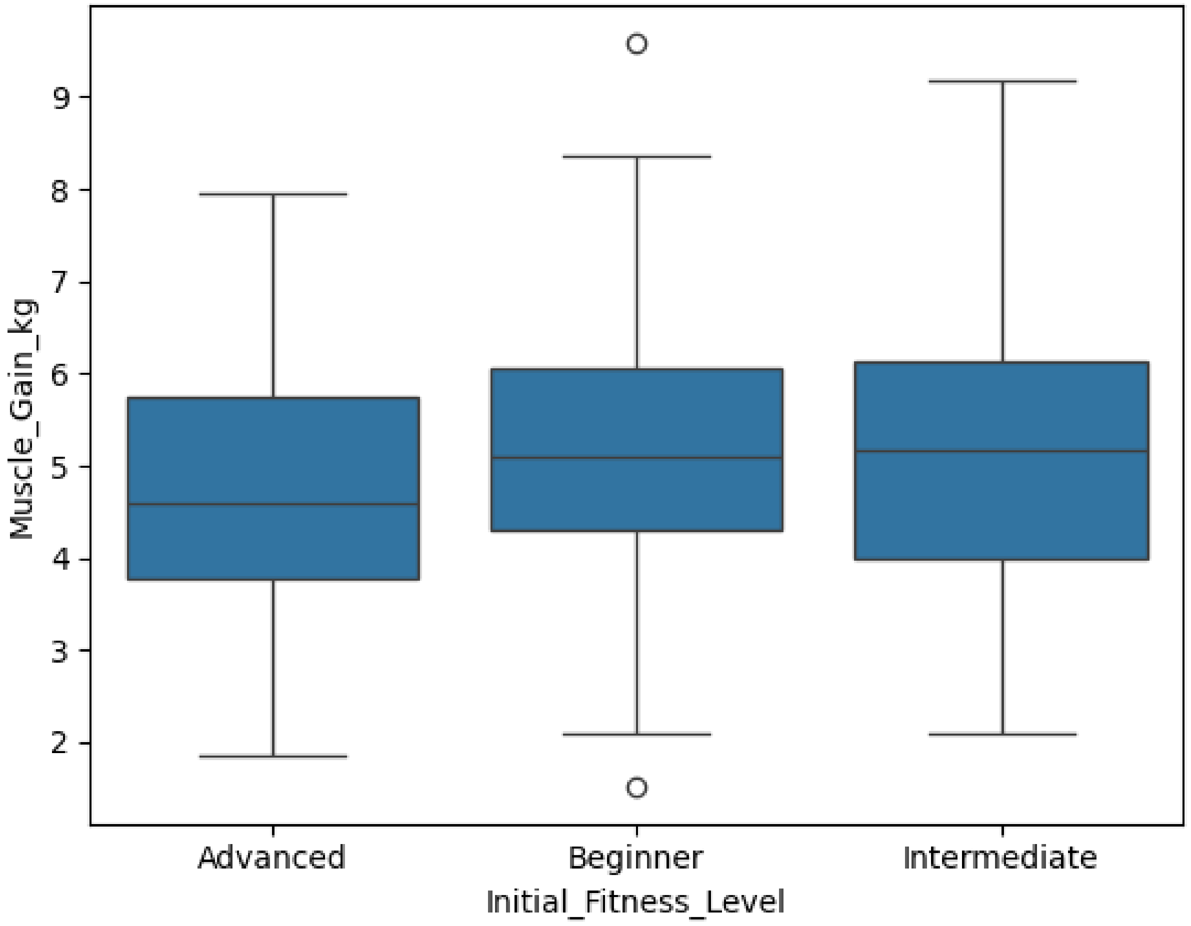
- Assume a significance level α of 0.05

```
from scipy.stats import f_oneway
blocks.groupby('Initial_Fitness_Level').apply(
  lambda x: f_oneway(x[x['Treatment'] == 'Cardio']['Muscle_Gain_kg'],
                     x[x['Treatment'] == 'Mixed']['Muscle_Gain_kg'],
                     x[x['Treatment'] == 'Strength Training']['Muscle_Gain_kg'])
)
```

```
Block
Initial_Fitness_Level
Advanced        (0.7951054385317405, 0.4555687666120679)
Beginner        (0.1085790370950905, 0.8972754969684291)
Intermediate    (0.5678877824942661, 0.5698403547950377)
dtype: object
```

Visualizing effects across blocks

```
import seaborn as sns
sns.boxplot(x='Initial_Fitness_Level', y='Muscle_Gain_kg', data=blocks)
plt.show()
```



ANOVA between blocks

```
f_oneway(
  blocks[blocks['Initial_Fitness_Level'] == "Advanced"]['Muscle_Gain_kg'],
  blocks[blocks['Initial_Fitness_Level'] == "Beginner"]['Muscle_Gain_kg'],
  blocks[blocks['Initial_Fitness_Level'] == "Intermediate"]['Muscle_Gain_kg']
)
```

```
F_onewayResult(statistic=2.325058605244051, pvalue=0.10045536062209368)
```

Let's practice!

EXPERIMENTAL DESIGN IN PYTHON

Covariate adjustment in experimental design

EXPERIMENTAL DESIGN IN PYTHON

James Chapman
Curriculum Manager, DataCamp

Introduction to covariates

- **Covariates:** potentially affect experiment results but aren't primary focus
- Importance in reducing confounding
- Impact on precision and validity of results

- **Example:** Impact of teaching method on test scores

Does the **teaching method** impact **scores?**

Prior Knowledge = Covariate

Experimental data example

```
exp_plant_data = plant_growth_data[['Plant_ID', 'Fertilizer_Type', 'Growth_cm']]
```

```
   Plant_ID  Light_Condition  Fertilizer_Type  Growth_cm
0         1    Full Sunlight        Synthetic  16.489735
1         2    Partial Shade          Organic  18.361689
2         3    Full Sunlight        Synthetic  18.039459
3         4    Full Sunlight          Organic  12.682425
4         5    Full Sunlight          Organic  21.480601
```

Covariate data example

```
covariate_data
```

```
   Plant_ID  Watering_Days_Per_Week
0         1                       6
1         2                       6
2         3                       4
3         4                       3
4         5                       7
```

Combining experimental data with covariates

```
merged_plant_data = pd.merge(exp_plant_data, covariate_data, on='Plant_ID')
```

```
   Plant_ID  Fertilizer_Type  Growth_cm  Watering_Days_Per_Week
0         1        Synthetic  16.489735                       6
1         2          Organic  18.361689                       6
2         3        Synthetic  18.039459                       4
3         4          Organic  12.682425                       3
4         5          Organic  21.480601                       7
```

Adjusting for covariates

```
from statsmodels.formula.api import ols
model = ols('Growth_cm ~ Fertilizer_Type + Watering_Days_Per_Week',
            data=merged_plant_data).fit()
model.summary()
```

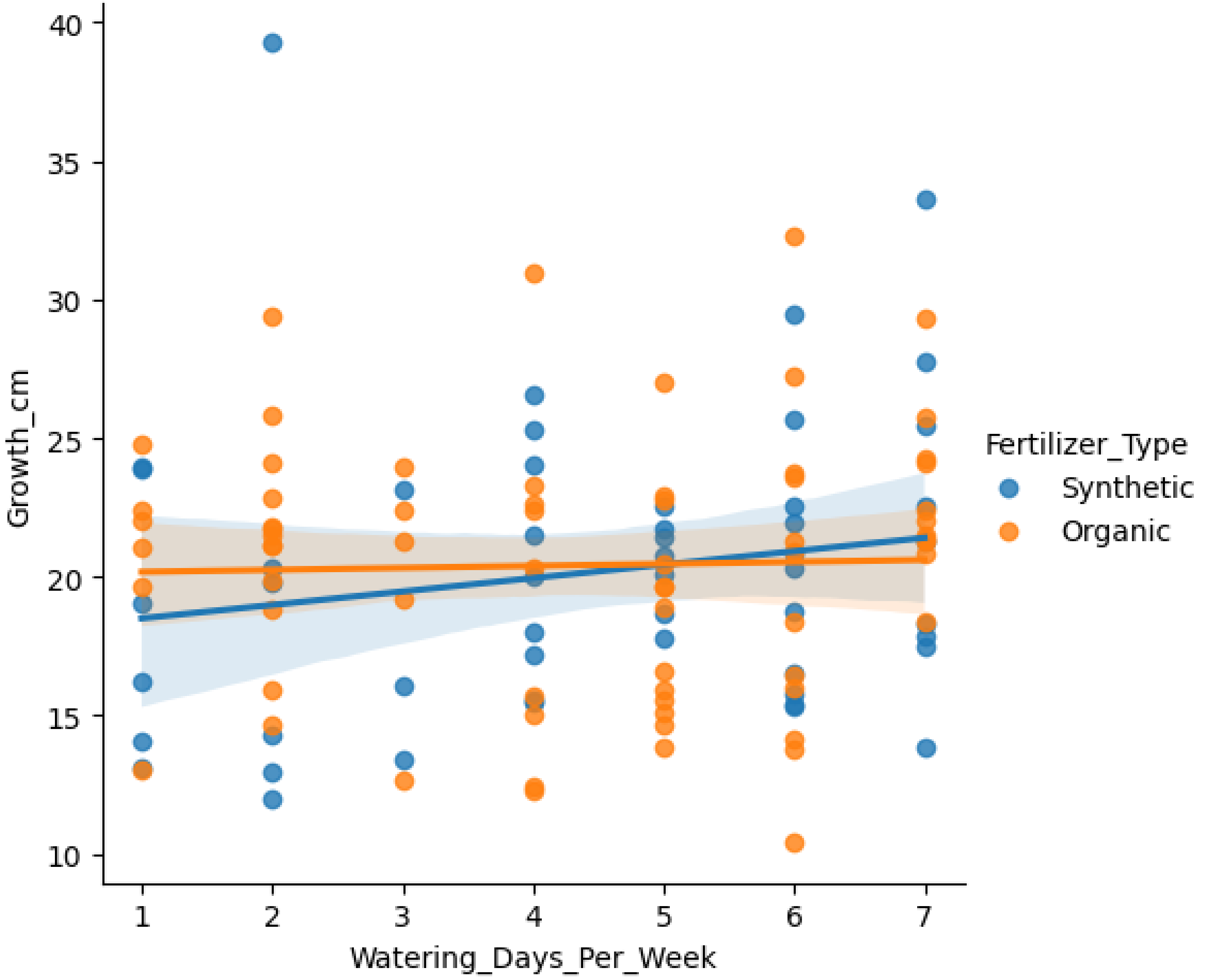
```
                            OLS Regression Results
==============================================================================
Dep. Variable:              Growth_cm   R-squared:                       0.011
Model:                            OLS   Adj. R-squared:                 -0.006
Method:               Least Squares   F-statistic:                    0.6370
No. Observations:                 120   Prob (F-statistic):              0.531 <---
Df Residuals:                     117   Log-Likelihood:                -360.45
Df Model:                           2   AIC:                             726.9
Covariance Type:            nonrobust   BIC:                             735.3
==============================================================================
```

Further exploring ANCOVA results

```
                                     coef      std err            t        P>|t|       [0.025      0.975]
<hr />----------------------------------------------------------------------------------------------
Intercept                         19.3373        1.150       16.820        0.000       17.060      21.614
Fertilizer_Type[T.Synthetic]      -0.2796        0.913       -0.306        0.760 <--   -2.088       1.528
Watering_Days_Per_Week             0.2507        0.229        1.097        0.275 <--   -0.202       0.703
==============================================================================================
Omnibus:                       14.446    Durbin-Watson:                  1.992
Prob(Omnibus):                  0.001    Jarque-Bera (JB):              18.267
Skew:                           0.675    Prob(JB):                    0.000108
Kurtosis:                       4.352    Cond. No.                        13.3
==============================================================================
```

Visualizing treatment effects with covariate adjustment

```python
import seaborn as sns
import matplotlib.pyplot as plt
sns.lmplot(x='Watering_Days_Per_Week',
           y='Growth_cm',
           hue='Fertilizer_Type',
           data=merged_plant_data)
plt.show()
```



Let's practice!

EXPERIMENTAL DESIGN IN PYTHON