**Factorial designs: principles and applications**

**Understanding factorial design**

Factorial designs allow for the simultaneous examination of multiple variables. In this setup, every possible combination of factor levels is tested, which not only measures the direct effects of each factor but also the interactions between them. In the example shown of plant growth in different conditions, implementing a factorial design will mean that we can test the effect of different factors on plant growth, including light conditions and fertilizer type, simultaneously, and identify interactions between them. These interactions can illuminate complex dynamics that might be overlooked in simpler experimental setups.

1. 1 Image Generated with DALL·E 3

**Factorial design data example**

To explain this concept further, we'll work with this plant growth DataFrame. It has 120 rows and four columns: an identifier column, two factors, and one response/dependent variable. Both factors have two levels: Light\_Condition can be Full Sunlight or Partial Shade, and Fertilizer\_Type can be either Synthetic or Organic. The Growth\_cm column is the numeric response, or dependent variable in the experiment.

**Organizing data to visualize interactions**

We next create a pivot table from the DataFrame using pandas' pivot\_table function. It aggregates the Growth\_cm values by taking their mean for each combination of Light\_Condition and Fertilizer\_Type. The resulting table displays these average outcomes, with light values as rows and fertilizer values as columns, illustrating how the growth varies across different levels of the two factors. For example, the value 19.869 represents the average growth for the combination of Full Sunlight from Light\_Condition and Synthetic from Fertilizer\_Type.

**Visualize interactions with heatmap**

The Seaborn heatmap function paints a picture of how these factors interact, with the color intensity revealing the strength and direction of their interactions. Setting annot to True displays the numerical value of the cell, and 'coolwarm' is a color map that ranges from cooler, or bluer colors, to warmer or redder colors. Lastly, the format argument fmt is set to 'g' to avoid scientific notation.

**Interpreting interactions**

The variation in outcomes when changing levels within a factor indicates an interaction. For instance, the decrease from Organic to Synthetic fertilizer within Full Sunlight (from 20.602 to 19.869) contrasts with the modest change within Partial Shade, illustrating how outcomes differ based on factor levels. The differing changes in outcomes between Full Sunlight and Partial Shade across Fertilizer\_Type suggest the factors interact, underscoring the need for nuanced strategies considering the interaction of factors.

**Factorial designs vs. randomized block designs**

Let's conclude by comparing factorial designs to the randomized block design we saw earlier in the course, and that we'll dive deeper into in the next video. Factorial designs investigate multiple treatments and their interactions to understand their combined effects on outcomes. They aim to unravel the effects and interactions of various factors, crucial for complex scenarios with multiple influencing variables. In factorial designs, units experience all treatment combinations, offering thorough exploration but requiring more subjects as treatments grow. Randomized block designs utilize blocks to group similar subjects, minimizing confounding impacts and clearer treatment effects. The focus of randomized block designs is on enhancing experimental precision by managing within-block variability, aiding in the detection of treatment differences. Randomized block designs assign one treatment per unit within blocks, ensuring each treatment's presence in every block to control for block-related variance and bolster treatment effect assessments.

**Randomized block design: controlling variance**

Next, we'll delve further into the concept of blocking in experimental design.

**2. Understanding blocking**

Blocking involves grouping experimental units, often with similar characteristics, to minimize variance within these groups. This ensures that each block, representing a specific level of the blocking factor, receives every treatment. This approach allows us to concentrate on the treatment effects while controlling for variance attributable to the blocking factor, thus improving the precision of our results.

**3. Block design data example**

For this athlete performance DataFrame of 200 rows, blocking is represented by Initial\_Fitness\_Level with categories of Beginner, Intermediate, and Advanced. Muscle\_Gain\_kg is a numeric response variable measured on participants for the year prior to blocks being assigned.

**4. Implementing randomized block design**

To implement a randomized block design, we'll group the rows into blocks based on the Initial\_Fitness\_Level in this case, shuffle the rows within these blocks, and randomly assign a treatment. To shuffle the rows in each Initial\_Fitness\_Level block, we start with .groupby() on Initial\_Fitness\_Level. To shuffle each row in that block, we chain the .apply() method to the groupby, and pass it a lambda function that reads: for each group, denoted by x, we sample all rows with frac=1, effectively shuffling them. We reset the index to not have both an index and column called Block. The grouped data is ordered alphabetically by fitness level.

**5. Implemented randomized blocks**

Then, within each block, we assign exercise program treatments randomly using numpy.random.choice(). This method allows us to control for block effects while focusing on the differences caused by the treatments. Here is a sample of the implemented randomized block DataFrame with the treatment randomly applied within each block.

**6. Visualizing treatment effects within blocks**

A boxplot is an effective tool for visualizing the distribution of treatment effects across different blocks. By plotting the Muscle\_Gain\_kg variable versus the Initial\_Fitness\_Level, coloring by Treatment, we observe the central tendencies and variabilities within each block. Scanning this boxplot, we see similar median values throughout the blocks and treatments. The variability is a bit wider for some, though, such as Cardio for Advanced and Beginner.

**7. ANOVA within blocks**

We can use ANOVA to statistically check for these differences. Let's set a significance level at 5% prior to reviewing our results. We group the DataFrame by the blocking column and then apply a lambda function to each group. Within the lambda function, we perform a one-way ANOVA test between the Muscle\_Gain\_kg values for each treatment within each block using f\_oneway from scipy.stats. Finally, it returns the F-statistic and p-value for each block's ANOVA test. Each of the p-values are above the alpha significance level of 5%. This gives evidence that significant differences don't exist across treatments within blocks. This is an ideal goal when setting up randomized block design experiments.

**8. Visualizing effects across blocks**

We can also look for differences in the outcome across randomized blocks. Here, we do not break down further by treatment. These boxplots look similar, so we might guess that none of the blocks has a significantly different mean outcome compared to the others.

**9. ANOVA between blocks**

Next we compute the one-way ANOVA test across the blocks. It compares the Muscle\_Gain\_kg values for each block separately to assess whether there are significant differences in means among the blocks. The function f\_oneway calculates the F-statistic and associated p-value, indicating the likelihood of observing the data if the null hypothesis of equal means across all blocks is true. A p-value greater than 0.05 supports what we saw with the boxplot - that there is no significant difference.

**Covariate adjustment in experimental design**

Let's now explore covariates in experimental design and analysis, and how they can be used to minimize confounding. We'll also learn about ANCOVA, or analysis of covariance, for evaluating treatment effects while controlling for covariates.

**2. Introduction to covariates**

00:17 - 01:42

Recall that covariates are variables that are not of primary interest but are related to the outcome variable and can influence its analysis. Including covariates in statistical analyses is crucial for reducing confounding, which occurs when an external variable influences both the dependent variable and independent variable(s). By adjusting for covariates, researchers can isolate the effect of the independent variable on the outcome, minimizing the influence of confounders. Accounting for covariates in experimental design and analysis controls for variability that is not attributable to the primary variables being studied. This leads to more valid conclusions about the relationship between the independent and dependent variables, as the analysis better reflects the true effect by isolating it from the influence of covariates. Consider the investigation of a new teaching method's effectiveness on student test scores. Here, the primary variables of interest are the teaching method (independent variable) and the student test scores (dependent variable). However, students' prior subject knowledge serves as a crucial covariate because prior knowledge can significantly impact learning outcomes, yet it's not the main focus of the study.

**3. Experimental data example**

01:42 - 01:54

Let's bring back our plant growth data and set it to experimental data as the exp\_data DataFrame, keeping Fertilizer\_Type as treatment and Growth\_cm as response.

**4. Covariate data example**

01:54 - 02:25

The covariate\_data DataFrame also includes Plant\_ID identifiers for each subject, again ranging from 1 to 120, ensuring each subject's covariate data is matched with their experimental data. Watering\_Days\_Per\_Week is another variable measured for each plant. Recall that covariates are additional variables potentially influencing the outcome and are included in analyses to control for their effects.

**5. Combining experimental data with covariates**

02:25 - 02:43

Combining the experimental with covariate data is a crucial step in adjusting for covariates. We use pandas' merge function to combine DataFrames; we do this on the Plant\_ID to ensure each that subject's experimental and covariate data are aligned.

**6. Adjusting for covariates**

02:43 - 03:34

To adjust for covariates in our analysis, we employ ANCOVA, or analysis of covariance, using the ols model from statsmodels. This ols() function takes a formula that specifies the dependent and independent variables. Growth\_cm is the dependent variable we're interested in, which we want to model using the Fertilizer\_Type, the categorical independent variable representing different groups in the experiment, and the potential covariate, Watering\_Days\_Per\_Week, to control for its effects. The first portion of summary output provides details on the significance of the model; it show a large p-value here of 0.531, which implies a lack of support for covariates affecting the model.

**7. Further exploring ANCOVA results**

03:34 - 03:53

Looking at the second and third rows of this second portion of output from summary, we see that the factors and covariate each have large p-values of 0.760 and 0.275, concluding that each of them alone are not significant predictors of growth for this model.

**8. Visualizing treatment effects with covariate adjustment**

03:53 - 04:34

This seaborn lmplot shows treatment effects adjusted for the covariate. The regression lines for each treatment category offer a visual representation of how treatment effects trend across different levels of the covariate. We see that Organic remains relatively constant going from 1 watering to 7 Watering\_Days\_Per\_Week. Synthetic shows an increase. The crossing regression lines suggest we may want to add an interaction term of Watering\_Days\_Per\_Week by Fertilizer\_Type in another model. Parallel lines would suggest a lack of interaction.