[Hypothesis Testing] (CheatSheet)

1. Basic Hypothesis Testing

- **One-Sample T-Test**: `scipy.stats.ttest_1samp(data, popmean)`
- **Two-Sample T-Test (Independent)**: `scipy.stats.ttest_ind(sample1, sample2)`
- **Paired T-Test**: `scipy.stats.ttest_rel(sample1, sample2)`
- **One-Way ANOVA**: `scipy.stats.f_oneway(sample1, sample2, sample3)`

2. Normality Tests

- **Shapiro-Wilk Test**: `scipy.stats.shapiro(data)`
- **D'Agostino's K-squared Test**: `scipy.stats.normaltest(data)`
- **Anderson-Darling Test**: `scipy.stats.anderson(data, dist='norm')`

3. Correlation Tests

- **Pearson Correlation Coefficient**: `scipy.stats.pearsonr(x, y)`
- **Spearman's Rank Correlation**: `scipy.stats.spearmanr(a, b)`
- **Kendall's Tau**: `scipy.stats.kendalltau(x, y)`

4. Comparing Variances

- **Levene's Test**: `scipy.stats.levene(sample1, sample2)`
- **Bartlett's Test**: `scipy.stats.bartlett(sample1, sample2)`

5. Non-parametric Tests

- **Mann-Whitney U Test**: `scipy.stats.mannwhitneyu(sample1, sample2)`
- **Wilcoxon Signed-Rank Test**: `scipy.stats.wilcoxon(x, y)`
- **Kruskal-Wallis H Test**: `scipy.stats.kruskal(sample1, sample2, sample3)`
- **Friedman Test**: `scipy.stats.friedmanchisquare(sample1, sample2, sample3)`

By: Waleed Mousa

6. Proportion Tests

- **Z-test for Proportions**: `statsmodels.stats.proportion.proportions_ztest(count, nobs)`
- **Chi-Squared Test for Proportions**: `scipy.stats.chi2_contingency(contingency_table)`

7. Regression Tests

- **Linear Regression Test**: `statsmodels.api.OLS(y, X).fit().summary()`
- **Logistic Regression Test**: `statsmodels.api.Logit(y, X).fit().summary()`

8. Time Series Analysis

- **Augmented Dickey-Fuller Test (Stationarity Test)**: `statsmodels.tsa.stattools.adfuller(data)`
- **Granger Causality Tests**: `statsmodels.tsa.stattools.grangercausalitytests(data, maxlag)`

9. Post-hoc Tests

- **Tukey's Honest Significant Difference Test**: `statsmodels.stats.multicomp.pairwise_tukeyhsd(endog, groups)`

10. Effect Size Calculations

- **Cohen's d for T-Test**: `CohenEffectSize(sample1, sample2)`
- **Eta Squared for ANOVA**: `EtaSquared(ANOVA_Result)`

11. Power Analysis

- **Sample Size for T-Test**: `statsmodels.stats.power.tt_ind_solve_power(effect_size, alpha, power)`
- **Sample Size for ANOVA**: `statsmodels.stats.power.FTestAnovaPower().solve_power(effect_size, alpha, power)`

By: Waleed Mousa

12. Multiple Comparisons Correction

- **Bonferroni Correction**: `scipy.stats.bonferroni(pvals)`
- **False Discovery Rate (FDR) Correction**: `statsmodels.stats.multitest.multipletests(pvals, method='fdr_bh')`

13. Visualization of Statistical Tests

- **QQ Plot for Normality Check**: `statsmodels.api.qqplot(data, line='s')`
- **Boxplot for Comparing Groups**: `seaborn.boxplot(x="group", y="data", data=df)`
- **Histogram for Data Distribution**: `matplotlib.pyplot.hist(data, bins)`

14. Data Preparation for Hypothesis Testing

- **Removing Outliers**: `df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]`
- **Log Transformation for Normality**: `np.log1p(df['column'])`

15. Advanced Statistical Modeling

- **Mixed Linear Models**: `statsmodels.regression.mixed_linear_model.MixedLM(endog, exog, groups).fit()`
- **Survival Analysis**: `lifelines.CoxPHFitter().fit(df, duration_col='T', event_col='E')`

16. Handling Missing Data for Tests

- **Impute Missing Values**: `sklearn.impute.SimpleImputer(strategy='mean').fit_transform(data)`

17. Distribution Fitting

- **Fit Distributions to Data**: `scipy.stats.<distribution>.fit(data)`

By: Waleed Mousa

18. Extracting Test Statistics

- **Extract P-Value and Test Statistic**: `result = scipy.stats.ttest_1samp(data, popmean); p_value = result.pvalue`

19. Bayesian Statistics

- **Bayesian Model Comparison**: `pymc3.compare({model1: trace1, model2: trace2})`

20. Structural Equation Modeling

- **SEM Model Fitting**: `semopy.estimate(sem_model, data)`

21. Multivariate Statistics

- **MANOVA**: `statsmodels.multivariate.manova.MANOVA.from_formula('y1 + y2 ~ x1 + x2', data).mv_test()`

22. Advanced Non-parametric Tests

- **Permutation Test**: `permute.core.permutation_test(x, y, func, method='approximate')`

23. Custom Hypothesis Testing Functions

- **Custom Test Function**: `def custom_test(data): /* implement test logic */; custom_test(data)`

24. Specialized Plots for Hypotheses

- **Scatter Plot with Regression Line**: `seaborn.regplot(x='x', y='y', data=df)`

25. Meta-analysis

- **Meta-Analysis**: `metafor::rma(yi, vi, data=meta_data)`

By: Waleed Mousa

26. Testing Assumptions for Parametric Tests

- **Homogeneity of Variances (Levene's Test)**: `scipy.stats.levene(sample1, sample2)`

27. Reporting Results

- **Formatted Result Output**: `def report_result(pval, alpha): if pval < alpha: print('Reject null hypothesis') else: print('Fail to reject null hypothesis')`

28. Working with Distributions in Hypothesis Testing

- **Drawing Samples from a Distribution**: `np.random.normal(loc=0, scale=1, size=100)`

29. Cross-Validation in Statistical Tests

- **Cross-Validation for Model Testing**: `sklearn.model_selection.cross_val_score(model, X, y, cv=5)`

30. Interpretation of Test Results

- **Interpreting Effect Sizes**: `def interpret_effect_size(d): /* logic to interpret Cohen's d */`

31. Advanced Correlation Analysis

- **Point-Biserial Correlation**: `scipy.stats.pointbiserialr(x, y)`
- **Partial Correlation**: `pingouin.partial_corr(data)`

32. Effect Size Analysis

- **Cohen's d for Independent Samples**: `Cohens_d(group1, group2)`
- **Cohen's d for Paired Samples**: `Cohens_d_paired(sample1, sample2, paired=True)`

By: Waleed Mousa

33. Bootstrapping Methods

- **Bootstrap Resampling**: `bootstrap = [np.random.choice(sample, size=len(sample), replace=True) for _ in range(n_iterations)]`

34. Bayesian Hypothesis Testing

- **Bayesian T-Test**: `pymc3.ttest(x, y)`
- **Bayesian ANOVA**: `pymc3.anova(model)`

35. Nonparametric Bootstrap Confidence Intervals

- **Bootstrap CI for Mean**: `bootstrapped_CI(sample, np.mean, alpha=0.05)`
- **Bootstrap CI for Median**: `bootstrapped_CI(sample, np.median, alpha=0.05)`

36. Power and Sample Size Calculation

- **Calculate Sample Size for T-Test**: `statsmodels.stats.power.tt_solve_power(effect_size, alpha, power)`
- **Calculate Power for T-Test**: `statsmodels.stats.power.TTestIndPower().solve_power(effect_size, nobs1, alpha)`

37. Multiple Testing Correction

- **Benjamini-Hochberg Procedure**: `statsmodels.stats.multitest.multipletests(pvals, method='fdr_bh')`
- **Holm-Bonferroni Method**: `statsmodels.stats.multitest.multipletests(pvals, method='holm')`

38. Exploratory Data Analysis for Hypothesis Testing

- **Pairplot for Visual Exploration**: `seaborn.pairplot(data)`
- **Heatmap for Correlation Analysis**: `seaborn.heatmap(data.corr(), annot=True)`

By: Waleed Mousa

39. Working with Categorical Data

- **Chi-Squared Test of Independence**: `scipy.stats.chi2_contingency(observed)`
- **Fisher's Exact Test**: `scipy.stats.fisher_exact(table)`

40. Tests for Proportions

- **Proportions Z-Test**: `statsmodels.stats.proportion.proportions_ztest(count, nobs)`
- **Test for Equality of Proportions**: `statsmodels.stats.proportion.test_proportions_2indep(success1, nobs1, success2, nobs2)`

41. Regression and ANCOVA

- **Simple Linear Regression Test**: `statsmodels.api.OLS(y, X).fit()`
- **ANCOVA**: `statsmodels.stats.anova.anova_lm(ols_model, typ=2)`

42. Tests for Model Fit

- **Log-Likelihood Ratio Test**: `statsmodels.stats.anova.anova_lm(model1, model2, test='LRT')`
- **AIC and BIC Comparison**: `model.aic, model.bic`

43. Spatial and Temporal Data Analysis

- **Mantel Test for Spatial Correlation**: `scipy.spatial.distance.mantel(x, y)`
- **Durbin-Watson Test for Autocorrelation**: `statsmodels.stats.stattools.durbin_watson(residuals)`

44. Working with Survival Data

- **Log-Rank Test for Survival Data**: `lifelines.statistics.logrank_test(event_times_A, event_times_B)`
- **Cox Proportional Hazards Model**: `lifelines.CoxPHFitter().fit(df, duration_col, event_col)`

By: Waleed Mousa

45. Robust Statistical Methods

- **M-estimators for Robust Regression**: `statsmodels.robust.robust_linear_model.RLM(y, X).fit()`
- **Robust Scale Estimators (e.g., MAD)**: `statsmodels.robust.scale.mad(data)`

46. Exploratory Factor Analysis

- **Factor Analysis**: `factor_analyzer.FactorAnalyzer().fit(data)`
- **Rotated Factor Analysis**: `factor_analyzer.FactorAnalyzer(rotation='varimax').fit(data)`

47. Multivariate Statistical Tests

- **Hotelling's T-squared Test**: `hotellings_t_square(sample1, sample2)`
- **MANOVA**: `statsmodels.multivariate.manova.MANOVA.from_formula('y1 + y2 ~ x1 + x2', data).mv_test()`

48. Graphical Representation of Test Results

- **Violin Plot for Distribution Comparison**: `seaborn.violinplot(x='group', y='data', data=df)`
- **ECDF Plot for Empirical Distribution**: `statsmodels.distributions.empirical_distribution.ECDF(data)`
- **Scatter Plot for Association**: `matplotlib.pyplot.scatter(x, y)`

49. Data Transformation for Hypothesis Testing

- **Box-Cox Transformation**: `scipy.stats.boxcox(data)`
- **Yeo-Johnson Transformation**: `scipy.stats.yeojohnson(data)`

50. Advanced Non-parametric Methods

- **Mann-Kendall Trend Test**: `scipy.stats.mstats.kendalltau(x, y)`
- **Sen's Slope for Trend Analysis**: `mk.original_test(data)`

By: Waleed Mousa

51. Hypothesis Testing in Time Series Analysis

- **Granger Causality Test**: `statsmodels.tsa.stattools.grangercausalitytests(data, maxlag)`
- **ADF Test (Augmented Dickey-Fuller)**: `statsmodels.tsa.stattools.adfuller(series)`

52. Spatial Statistics

- **Moran's I for Spatial Autocorrelation**: `esda.moran.Moran(data, w)`
- **Geographically Weighted Regression (GWR)**: `mgwr.gwr.GWRModel(y, X, bw).fit()`

53. Dealing with High Dimensionality

- **Principal Component Analysis (PCA)**: `sklearn.decomposition.PCA(n_components).fit(data)`
- **Factor Analysis for Dimension Reduction**: `sklearn.decomposition.FactorAnalysis(n_components).fit(data)`

54. Working with Complex Datasets

- **Handling Missing Data with Multiple Imputation**: `sklearn.impute.IterativeImputer().fit_transform(data)`
- **Random Forest for Feature Importance**: `sklearn.ensemble.RandomForestClassifier().fit(X, y)`

55. Interpretation and Reporting

- **Effect Size Calculation (Cohen's d)**: `compute_cohens_d(group1, group2)`
- **Confidence Interval Calculation**: `statsmodels.stats.proportion.proportion_confint(count, nobs, alpha=0.05)`

56. Custom Hypothesis Tests

- **Writing Custom Test Functions**: `def custom_test(data): # Implement your test logic`

By: Waleed Mousa

57. Simulation for Hypothesis Testing

- **Simulating Data for Power Analysis**: `numpy.random.normal(loc, scale, size)`
- **Monte Carlo Simulation for P-values**: `perform_monte_carlo_simulation(data, num_simulations)`

58. Advanced Visualization for Test Assumptions

- **Q-Q Plot for Normality**: `scipy.stats.probplot(data, plot=matplotlib.pyplot)`
- **Pair Plot for Multivariate Normality**: `seaborn.pairplot(data)`

59. Post-hoc Analysis

- **Post-hoc Analysis for ANOVA**: `statsmodels.stats.multicomp.pairwise_tukeyhsd(endog, groups)`

60. Handling Time-to-Event Data

- **Kaplan-Meier Estimator**: `lifelines.KaplanMeierFitter().fit(durations, event_observed)`