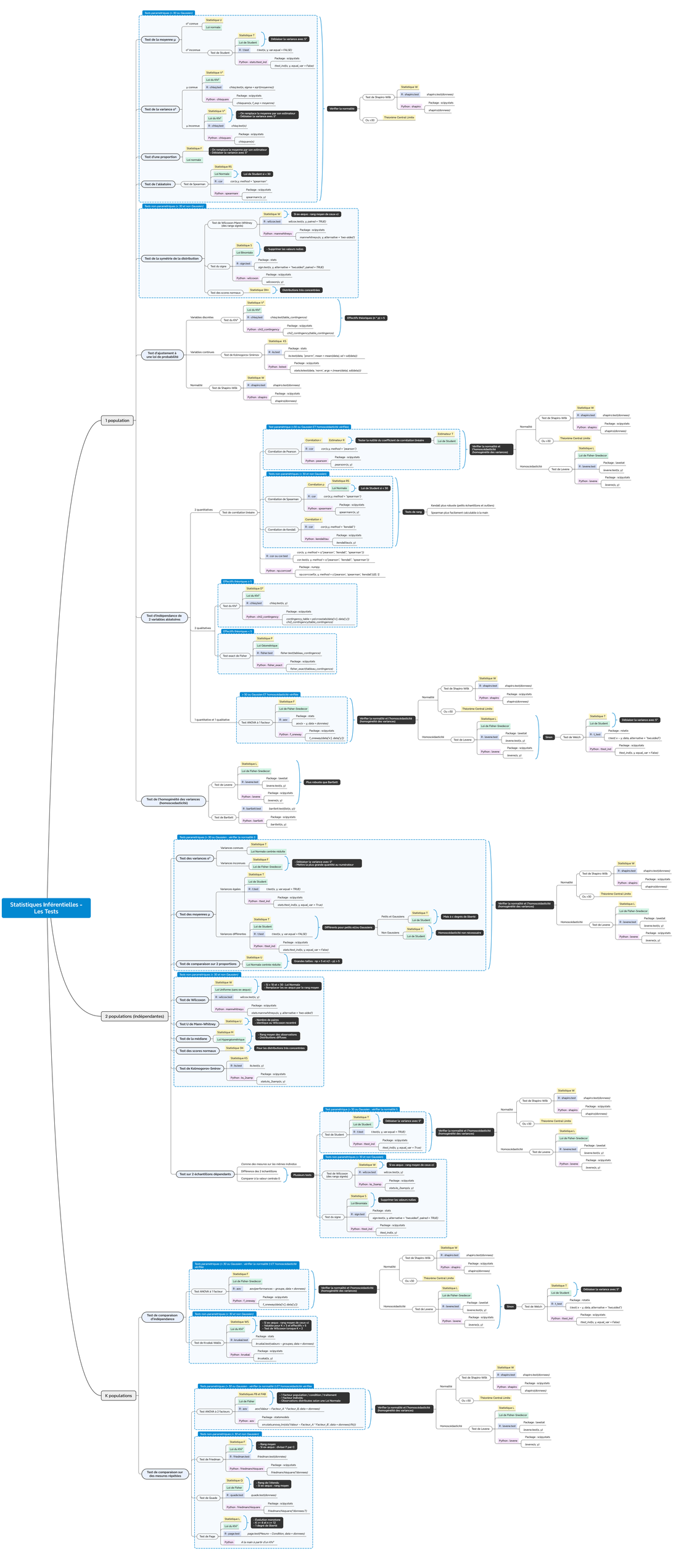
Statistiques Inférentielles – Les Tests
1 population
Tests paramétriques (> 30 ou Gaussien)
Test de la moyenne μ
σ² connue
Statistique U
Loi normale
σ² inconnue
Test de Student
Statistique T
Loi de Student
Débiaiser la variance avec S²
R : t.test
t.test(x, y, var.equal = FALSE)
Python : stats.ttest_ind
Package : scipy.stats
ttest_ind(x, y, equal_var = False)
Test de la variance σ²
μ connue
Statistique V²
Loi du Khi²
R : chisq.test
chisq.test(x, sigma = sqrt(moyenne))
Python : chisquare
Package : scipy.stats
chisquare(x, f_exp = moyenne)
μ inconnue
Statistique V²
Loi du Khi²
- On remplace la moyenne par son estimateur
- Débiaiser la variance avec S²
R : chisq.test
chisq.test(x)
Python : chisquare
Package : scipy.stats
chisquare(x)
Test d'une proportion
Statistique F
Loi normale
- On remplace la moyenne par son estimateur
- Débiaiser la variance avec S²
Test de l'aléatoire
Test de Spearman
Statistique RS
Loi Normale
Loi de Student si < 30
R : cor
cor(x,y, method = "spearman")
Python : spearmanr
Package : scipy.stats
spearmanr(x, y)
Vérifier la normalité
Test de Shapiro-Wilk
Statistique W
R : shapiro.test
shapiro.test(données)
Python : shapiro
Package : scipy.stats
shapiro(données)
Ou >30
Théorème Central Limite
Tests non-paramétriques (< 30 et non Gaussien)
Test de la symétrie de la distribution
Test de Wilcoxon-Mann-Whitney (des rangs signés)
Statistique W
Si on arrque : rang moyen de ceux-ci
R : wilcox.test
wilcox.test(x, y, paired = TRUE)
Python : mannwhitneyu
Package : scipy.stats
mannwhitneyu(x, y, alternative = 'two-sided')
Test du signe
Statistique S
Loi Binomiale
- Supprimer les valeurs nulles
R : sign.test
Package : stats
sign.test(x, y, alternative = "two.sided", paired = TRUE)
Python : wilcoxon
Package : scipy.stats
wilcoxon(x, y)
Test des scores normaux
Statistique SN=
Distributions très concentrées
Test d'ajustement à une loi de probabilité
Variables discrètes
Test du Khi²
Statistique V²
Loi du Khi²
R : chisq.test
chisq.test(table_contingence)
Python : chi2_contingency
Package : scipy.stats
chi2_contingency(table_contingence)
Effectifs théoriques (n * pi) > 5
Variables continues
Test de Kolmogorov-Smirnov
Statistique KS
R : ks.test
Package : stats
ks.test(data, "pnorm", mean = mean(data), sd = sd(data))
Python : kstest
Package : scipy.stats
stats.kstest(data, 'norm', args = (mean(data), sd(data)))
Normalité
Test de Shapiro-Wilk
Statistique W
R : shapiro.test
shapiro.test(données)
Python : shapiro
Package : scipy.stats
shapiro(données)
Test d'indépendance de 2 variables aléatoires
2 quantitatives
Test de correlation linéaire
Test paramétrique (> 30 ou Gaussien ET homocédasticité vérifiée)
Corrélation de Pearson
Corrélation r
Estimateur R
Tester la nullité du coefficient de corrélation linéaire
Estimateur T
Loi de Student
R : cor
cor(x,y, method = 'pearson')
Python : pearsonr
Package : scipy.stats
pearsonr(x, y)
Tests non-paramétriques (< 30 et non Gaussien)
Corrélation de Spearman
Corrélation ρ
Statistique RS
Loi Normale
Loi de Student si < 30
R : cor
cor(x,y, method = 'spearman')
Python : spearmanr
Package : scipy.stats
spearmanr(x, y)
Corrélation de Kendall
Corrélation τ
R : cor
cor(x,y, method = 'kendall')
Python : kendalltau
Package : scipy.stats
kendalltau(x, y)
Tests de rang
Kendall plus robuste (petits échantillons et outliers)
Spearman plus facilement calculable à la main
R : cor ou cor.test
cor(x, y, method = c('pearson', 'kendall', 'spearman'))
cor.test(x, y, method = c('pearson', 'kendall', 'spearman'))
Python : npz.corrcoef
Package : numpy
npy.corrcoef(x, y, method = c('pearson', 'spearman', 'kendall'))[0, 1]
2 qualitatives
Test du Khi²
Effectifs théoriques ≥ 5
Statistique D²
Loi du Khi²
R : chisq.test
chisq.test(x, y)
Python : chi2_contingency
Package : scipy.stats
contingency_table = pd.crosstab(data['x'], data['y'])
chi2_contingency(table_contingence)
Test exact de Fisher
Effectifs théoriques < 5
Statistique F
Loi Géométrique
R : fisher.test
fisher.test(tableau_contingence)
Python : fisher_exact
Package : scipy.stats
fisher_exact(tableau_contingence)
1 quantitative et 1 qualitative
Test ANOVA à 1 facteur
n > 30 ou Gaussien ET homocédasticité vérifiée
Statistique F
Loi de Fisher-Snedecor
R : aov
Package : stats
anova = aov(data = données)
Python : f_oneway
Package : scipy.stats
f_oneway(data['x'], data['y'])
Vérifier la normalité et l'homocédasticité (homogénéité des variances)
Normalité
Test de Shapiro-Wilk
Statistique W
R : shapiro.test
shapiro.test(données)
Python : shapiro
Package : scipy.stats
shapiro(données)
Ou >30
Théorème Central Limite
Homocédasticité
Test de Levene
Statistique L
Loi de Fisher-Snedecor
R : levene.test
Package : lawstat
levene.test(x, y)
Python : levene
Package : scipy.stats
levene(x, y)
Sinon
Test de Welch
Statistique T
Loi de Student
Débiaiser la variance avec S²
R : t.test
Package : ntatis
t.test(x ~ y, data, alternative = 'two.sided')
Python : ttest_ind
Package : scipy.stats
ttest_ind(x, y, equal_var = False)
Test de l'homogénéité des variances (homocédasticité)
Test de Levene
Statistique L
Loi de Fisher-Snedecor
R : levene.test
Package : lawstat
levene.test(x, y)
Python : levene
Package : scipy.stats
levene(x, y)
Plus robuste que Bartlett
Test de Bartlett
R : bartlett.test
Package : scipy.stats
bartlett(x, y)
Python : bartlett
2 populations (indépendantes)
Tests paramétriques (> 30 ou Gaussien - vérifier la normalité !)
Test des variances σ²
Variances connues
Statistique T
Loi Normale centrée réduite
Variances inconnues
Statistique F
Loi de Fisher-Snedecor
- Débiaiser la variance avec S²
- Mettre la plus grande quantité au numérateur
Test des moyennes μ
Variances égales
Statistique T
Loi de Student
R : t.test
t.test(x, y, var.equal = TRUE)
Python : ttest_ind
Package : scipy.stats
stats.ttest_ind(x, y, equal_var = True)
Variances différentes
Statistique T
Loi de Student
R : t.test
t.test(x, y, var.equal = FALSE)
Différents pour petits et/ou Gaussiens
Petits et Gaussiens
Statistique T
Loi de Student
Mais à c degrés de liberté
Non Gaussiens
Statistique T
Loi de Student
Homocédasticité non nécessaire
Python : ttest_ind
Package : scipy.stats
stats.ttest_ind(x, y, equal_var = False)
Test de comparaison sur 2 proportions
Statistique U
Loi Normale centrée réduite
Grandes tailles : np ≥ 5 et n(1 - p) ≥ 5
Tests non-paramétriques (< 30 et non Gaussien)
Test de Wilcoxon
Statistique W
Loi Uniforme (sans ex-aequo)
- Si > 10 et < 30 : Loi Normale
- Remplacer les ex-aequo par le rang moyen
R : wilcox.test
wilcox.test(x, y)
Python : mannwhitneyu
Package : scipy.stats
stats.mannwhitneyu(x, y, alternative = 'two-sided')
Test U de Mann-Whitney
Statistique U
- Nombre de paires
- Identique au Wilcoxon recentré
Test de la médiane
Statistique M
Loi Hypergeométrique
- Rang moyen des observations
- Distributions diffuses
Test des scores normaux
Statistique SN
Pour les distributions très concentrées
Test de Kolmogorov-Smirnov
Statistique KS
R : ks.test
ks.test(x, y)
Python : ks_2samp
Package : scipy.stats
stats.ks_2samp(x, y)
Vérifier la normalité et l'homocédasticité (homogénéité des variances)
Normalité
Test de Shapiro-Wilk
Statistique W
R : shapiro.test
shapiro.test(données)
Python : shapiro
Package : scipy.stats
shapiro(données)
Ou >30
Théorème Central Limite
Homocédasticité
Test de Levene
Statistique L
Loi de Fisher-Snedecor
R : levene.test
Package : lawstat
levene.test(x, y)
Python : levene
Package : scipy.stats
levene(x, y)
Test sur 2 échantillons dépendants
Comme des mesures sur les mêmes individus
Différence des 2 échantillons
Comparer à la valeur centrale 0
Plusieurs tests
Test paramétrique (> 30 ou Gaussien - vérifier la normalité !)
Test de Student
Statistique T
Loi de Student
Débiaiser la variance avec S²
R : t.test
t.test(x, y, var.equal = TRUE)
Python : ttest_ind
Package : scipy.stats
ttest_ind(x, y, equal_var = True)
Tests non-paramétriques (< 30 et non Gaussien)
Test de Wilcoxon (des rangs signés)
Statistique W
Si on arrque : rang moyen de ceux-ci
R : wilcox.test
wilcox.test(x, y)
Python : ks_2samp
Package : scipy.stats
stats.ks_2samp(x, y)
Test du signe
Statistique S
Loi Binomiale
Supprimer les valeurs nulles
R : sign.test
Package : stats
sign.test(x, y, alternative = "two.sided", paired = TRUE)
Python : ttest_ind
Package : scipy.stats
ttest_ind(x, y)
Vérifier la normalité et l'homocédasticité (homogénéité des variances)
Normalité
Test de Shapiro-Wilk
Statistique W
R : shapiro.test
shapiro.test(données)
Python : shapiro
Package : scipy.stats
shapiro(données)
Ou >30
Théorème Central Limite
Homocédasticité
Test de Levene
Statistique L
Loi de Fisher-Snedecor
R : levene.test
Package : lawstat
levene.test(x, y)
Python : levene
Package : scipy.stats
levene(x, y)
Sinon
Test de Welch
Statistique T
Loi de Student
Débiaiser la variance avec S²
R : t.test
Package : ntatis
t.test(x ~ y, data, alternative = 'two.sided')
Python : ttest_ind
Package : scipy.stats
ttest_ind(x, y, equal_var = False)
K populations
Test de comparaison d'indépendance
Tests paramétriques (> 30 ou Gaussien - vérifier la normalité !! ET homocédasticité vérifiée
Test ANOVA à 1 facteur
Statistique F
Loi de Fisher-Snedecor
R : aov
aov(performances ~ groupe, data = données)
Python : f_oneway
Package : scipy.stats
f_oneway(data['x'], data['y'])
Vérifier la normalité et l'homocédasticité (homogénéité des variances)
Normalité
Test de Shapiro-Wilk
Statistique W
R : shapiro.test
shapiro.test(données)
Python : shapiro
Package : scipy.stats
shapiro(données)
Ou >30
Théorème Central Limite
Homocédasticité
Test de Levene
Statistique L
Loi de Fisher-Snedecor
R : levene.test
Package : lawstat
levene.test(x, y)
Python : levene
Package : scipy.stats
levene(x, y)
Sinon
Test de Welch
Statistique T
Loi de Student
Débiaiser la variance avec S²
R : t.test
Package : ntatis
t.test(x ~ y, data, alternative = 'two.sided')
Python : ttest_ind
Package : scipy.stats
ttest_ind(x, y, equal_var = False)
Tests non-paramétriques (< 30 et non Gaussien)
Test de Kruskal-Wallis
Statistique WS
Loi du Khi²
- Si ex-aequo : rang moyen de ceux-ci
- Si > 10 et < 30 : Loi du Khi² à 1 ddl
- Test de Wilcoxon-Mann-Whitney à 2
R : kruskal.test
Package : stats
kruskal.test(valeurs ~ groupes, data = données)
Python : kruskal
Package : scipy.stats
kruskal(x, y)
Test de comparaison sur des mesures répétées
Tests paramétriques (> 30 ou Gaussien - vérifier la normalité !! ET homocédasticité vérifiée
Test ANOVA à 2 facteurs
Statistiques FB et FAB
Loi de Fisher
- 1 facteur population / condition / traitement
- 1 facteur individu
- Observations distribuées selon une Loi Normale
R : aov
aov(Valeur ~ Facteur_A * Facteur_B, data = données)
Python : aov
Package : statsmodels
sm.stats.anova_lm(ols('Valeur ~ Facteur_A * Facteur_B', data = données).fit())
Vérifier la normalité et l'homocédasticité (homogénéité des variances)
Normalité
Test de Shapiro-Wilk
Statistique W
R : shapiro.test
shapiro.test(données)
Python : shapiro
Package : scipy.stats
shapiro(données)
Ou >30
Théorème Central Limite
Homocédasticité
Test de Levene
Statistique L
Loi de Fisher-Snedecor
R : levene.test
Package : lawstat
levene.test(x, y)
Python : levene
Package : scipy.stats
levene(x, y)
Tests non-paramétriques (< 30 et non Gaussien)
Test de Friedman
Statistique F
Loi du Khi²
- Rang moyen
- Si ex-aequo : diviser F par Q
R : friedman.test
friedman.test(données)
Python : friedmanchisquare
Package : scipy.stats
friedmanchisquare(*données)
Test de Quade
Statistique Q
Loi de Fisher
- Rang de l'étendu
- Si ex-aequo : rang moyen
R : quade.test
quade.test(données)
Python : friedmanchisquare
Package : scipy.stats
friedmanchisquare(*données ?)
Test de Page
Statistique L
Loi du Khi²
- Distributions continue
- X >= 4 et n >= 12
- 1 degré de liberté
R : page.test
page.test(Mesure ~ Condition, data = données)
Python
A la main à partir d'un Khi²
Présenté avec xmind