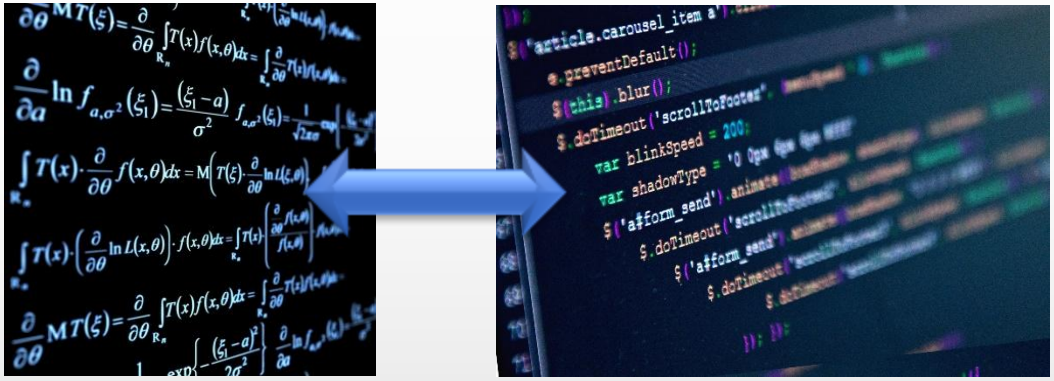
Les statistiques sur R



Bases et astuces sous R studio

Julie Ripoll

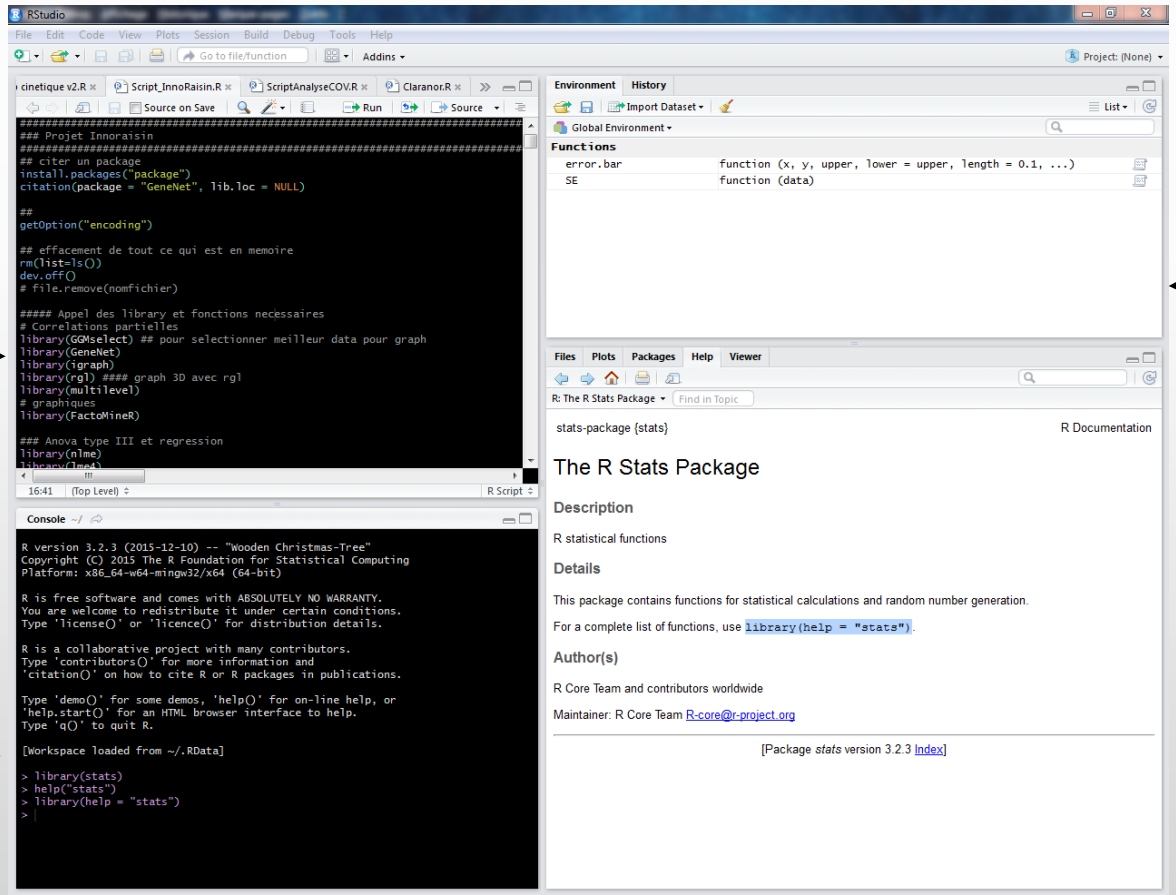
Juin 2016

Interface de travail R studio

- Interface ludique et facile à maitriser
- Logiciel gratuit: https://www.rstudio.com/
- Nécessite d'avoir installer R au préalable sur votre PC en plus de R studio

Fenêtre pour les scripts

Fenêtre de sortie des analyses



Fenêtre de l'environnement (données et fonctions en mémoire)

Plusieurs sous-fenêtres: aide et graphiques principalement

Exemple d'organisation Rstudio

Base du langage R

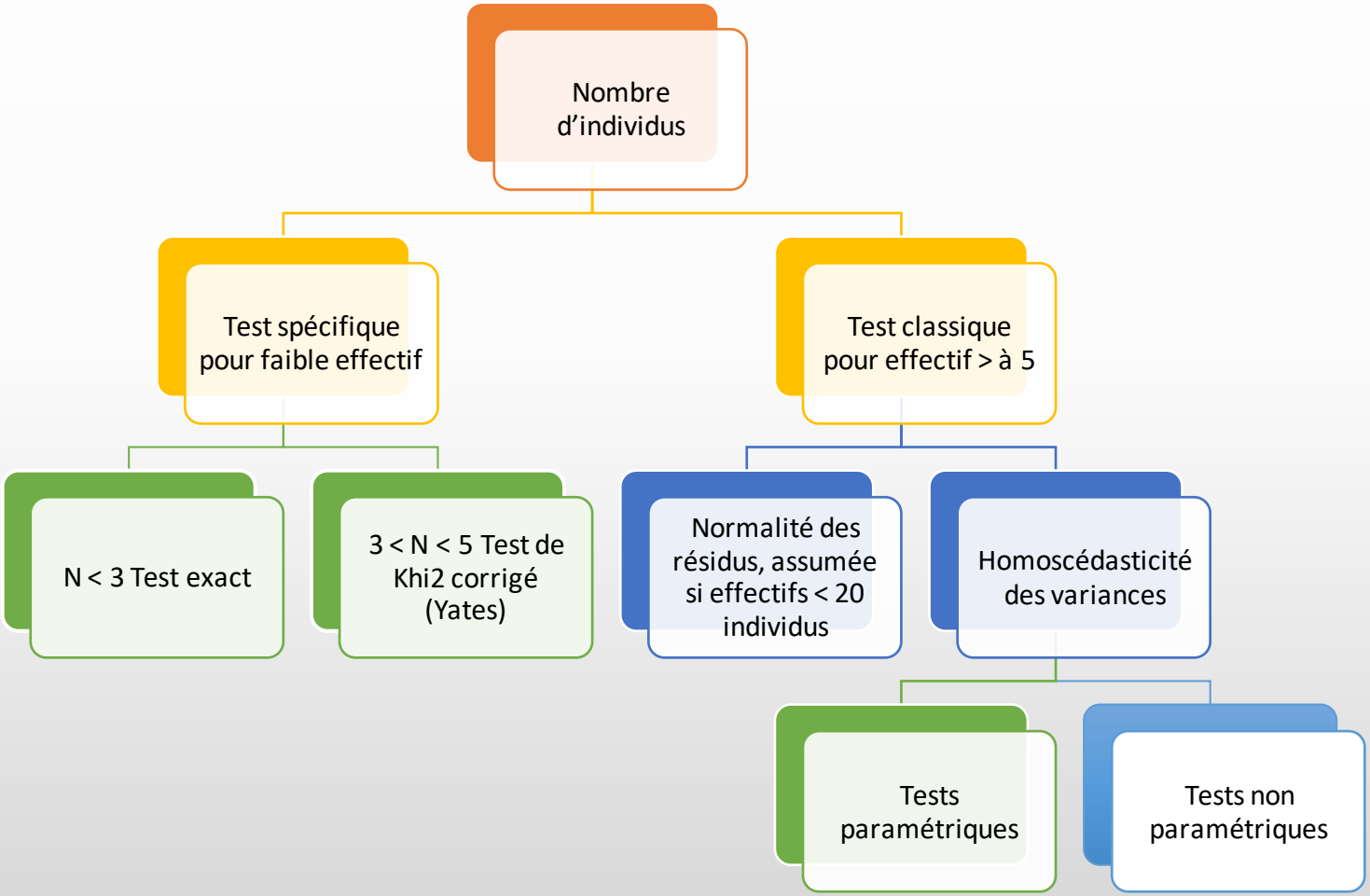
- Fiche des principales « fonctions » reconnues par R

- Forums d'aide:
  - StackOverFlow
  - R forum Cirad
  - R community
  - R bloggers

- Avant de commencer vos analyses
  - Poser une question de recherche
  - Emettre une hypothèse
  - Elaborer la procédure expérimentale pour tester l'hypothèse (→ manip)
  - Vérifier les conditions de bases amenant au choix du bon test (types de variables, taille de l'échantillonnage, loi de répartition…)

Tests statistiques de base

- Rappel



Format classique du fichier de données

- Formats lu par R: excel, texte…
- Exemple de fichier data

Identifiants

Facteurs

Variables en colonne

Data_FluoEAU_Innoraisin - Bloc-notes

Fichier Edition Format Affichage ?

Traitement	Date	Fo	Fm	Fv	Fv/Fm	Tfm	Area	F1	F2	F3	F4	F5
WD1	02/07/2014	337	1850	1513	0.818	400	46530	389	436	610	958	1331
WD1	02/07/2014	382	2161	1779	0.823	250	46679	451	515	734	1098	1642
WD1	02/07/2014	384	2156	1772	0.822	400	54313	429	474	609	878	1694
WD1	02/07/2014	352	1885	1533	0.813	200	32889	415	475	686	1072	1435
WD1	02/07/2014	335	1995	1660	0.832	260	43056	398	458	660	992	1541
WD1	02/07/2014	394	2527	2133	0.844	260	51813	475	555	812	1216	1895
WD1	02/07/2014	360	2237	1877	0.839	280	47481	429	498	733	1115	1715
WD1	02/07/2014	338	2050	1712	0.835	220	36808	405	469	691	1057	1568
WD1	02/07/2014	321	1888	1567	0.83	210	33685	384	444	657	1000	1501
WD1	02/07/2014	416	1650	1234	0.748	260	31255	477	534	733	1146	1224
WD1	02/07/2014	347	1819	1472	0.809	270	41348	401	451	638	1058	1348
WD1	02/07/2014	400	1919	1519	0.792	230	37549	462	523	740	1160	1490
WD1	02/07/2014	328	1423	1095	0.77	500	41963	356	384	479	702	1087
WD1	02/07/2014	395	1993	1598	0.802	260	38850	457	514	739	1175	1508
WD2	02/07/2014	405	2022	1617	0.8	230	35380	487	569	826	1250	1542
WD2	02/07/2014	376	2042	1666	0.816	280	45468	444	509	724	1127	1682
WD2	02/07/2014	334	1985	1651	0.832	260	38874	399	466	675	1006	1541
WD2	02/07/2014	411	2303	1892	0.822	200	34921	508	604	918	1404	1828
WD2	02/07/2014	345	1825	1480	0.811	200	29793	414	481	704	1076	1455
WD2	02/07/2014	363	2319	1956	0.843	240	40200	446	528	787	1151	1891
WD2	02/07/2014	347	2012	1665	0.828	290	46296	419	494	730	1114	1507
WD2	02/07/2014	304	1809	1505	0.832	290	41849	354	409	589	909	1344
WD2	02/07/2014	297	1646	1349	0.82	260	34509	346	406	598	962	1210
WD2	02/07/2014	377	2222	1845	0.83	230	40797	453	531	781	1181	1748
WD2	02/07/2014	326	1992	1666	0.836	280	40163	394	457	694	1110	1574
WD2	02/07/2014	383	1822	1439	0.79	270	34295	450	516	746	1212	1381
WD2	02/07/2014	330	1796	1466	0.816	210	37443	390	448	656	1062	1331
WD2	02/07/2014	395	1706	1311	0.768	250	30517	461	529	752	1159	1297
WD2	02/07/2014	365	1579	1214	0.769	220	28468	425	489	687	1046	1155
Control	02/07/2014	361	2260	1899	0.84	290	44505	426	494	709	1071	1792
Control	02/07/2014	390	2268	1878	0.828	290	50032	455	518	751	1198	1645
Control	02/07/2014	329	1859	1530	0.823	260	38665	397	462	661	964	1339
Control	02/07/2014	367	2494	2127	0.853	300	51942	433	500	733	1173	1942
Control	02/07/2014	349	2112	1763	0.835	280	45950	421	490	745	1175	1637
Control	02/07/2014	378	2302	1924	0.836	230	40000	459	542	802	1203	1742
Control	02/07/2014	380	2409	2029	0.842	280	47674	445	507	725	1110	1877
Control	02/07/2014	319	2166	1847	0.853	280	41631	374	434	649	1065	1724
Control	02/07/2014	418	2033	1615	0.794	200	29060	502	586	859	1315	1605
Control	02/07/2014	357	1729	1372	0.794	270	33279	422	483	697	1080	1275
Control	02/07/2014	420	2046	1626	0.795	270	40405	494	565	808	1285	1484
Control	02/07/2014	328	2007	1679	0.837	290	47606	390	449	666	1090	1547
Control	02/07/2014	357	2026	1669	0.824	250	37694	420	481	698	1112	1607
Control	02/07/2014	413	2078	1665	0.801	190	34748	496	581	839	1193	1537
Control	02/07/2014	317	1893	1576	0.833	250	40199	371	421	610	938	1449
WD1	08/07/2014	333	2036	1703	0.836	400	52647	389	441	649	1096	1581
WD1	08/07/2014	339	2212	1873	0.847	500	47836	406	467	709	1208	1821

Ln 1, Col 1

1 ligne du tableau = 1 observation à un temps donné

Charger un ou plusieurs fichiers en mémoire

- Définir le répertoire de travail
  - Code: setwd("~/Nom_du_répertoire/Sous_dossier")

- Sélectionner son fichier
  - Code: file1="Data.txt“
  - On peut lister tous ses fichiers de cette manière pour pouvoir les appeler facilement
  - Note: il est aussi possible de charger le fichier par l’interface de menus de R studio

- Lire le fichier
  - Code: data <-read.delim(file1, dec=".")
  - Note: dans la partie environnement de R studio, vous devez voir apparaitre votre fichier data

Afficher une partie ou toutes les données

- Afficher vos noms de colonnes
  - Code: names(data)
- Afficher les noms de colonnes et les 5 premières lignes
  - Code: head(data)
- Afficher un résumé des données
  - Code: summary(data)
- Afficher une colonne des données
  - Code: data$Nom_de_colonne
- Effacer les fichiers en mémoire
  - Code: rm(list=ls())

Opérations sur les composantes du jeu de données

- Définir une variable ou un vecteur
  - Code: A<-1 ou A=1
  - attribue la valeur 1 à A
  - Code: u1<- c(99.813, 9.9813, 0.99813)
  - u1 est un vecteur composé des 3 valeurs définies

- Rentrer une variable existante du tableau de données en mémoire
  - Code: Traitement<-(data$Traitement)
  - Permet de raccourcir les scripts si besoin

- Enlever une variable d'un jeu de données
  - Code: data2<-data[,-c(8,10,16,54)] # retire les colonnes correspondantes

- Sélectionner ou enlever un traitement en particuliers
  - Code: selectdata<- data[data$Cinétique=="stress",] # sélectionne
  - Code: selectdata<- data[data$Cinétique!="control",] # enlève

Les packages R

- Package de base de R: stats
  - Contient toutes les fonctions nécessaires pour effectuer des analyses statistiques basiques
  - Liste et informations sur tous les packages R existants sur le site officiel Rcran: https://cran.r-project.org/web/packages/available_packages_by_name.html

- Installer et charger un package:
  - Code: install.packages("nom_package")
  - Code: library(nom_package)
  - Note: vous pouvez aussi utiliser l'interface spécifique dans le menus de R studio (choisissez votre site de référence pour le chargement des packages France(Lyon) ou France(Toulouse))

- Obtenir de l'aide sur un package
  - Code: help("stats")
  - Ouvre une page dans la fenêtre spécifique de R studio

- Citer un package
  - Code: citation(package = " nom_package", lib.loc = NULL)

Travail sur les données: moyenne et erreur standard

- Obtenir les moyennes de toutes vos variables en 1 clic
  - Code: `aggregate(data[,-c(1:2)],by=list(data$Genotype,data$Traitement),mean,na.rm=T)`
  - Note: vous pouvez mettre autant de facteurs que vous voulez dans la liste, pensez à enlever les colonnes de texte correspondantes

- Obtenir les erreurs standards
  - Code fonction:
```
SE<-function(data){
sterr<- sqrt(var(data,na.rm=TRUE)/ length(data[is.na(data)!=TRUE]))
return(sterr)
}
```
  - Code: `aggregate(data[,-c(1)],by=list(data$Traitement),SE)`

- Exporter le tableau de résultats des moyennes ou des erreurs standards
  - Code:
```
MEAN<-aggregate(data[,-c(1)],by=list(data$Traitement),mean,na.rm=T)
MEAN<-as.data.frame(MEAN)
write.table(MEAN,"MEAN_data.txt",dec=".",quote=FALSE,row.names=FALSE)
```
  - Créer le fichier texte dans le dossier du répertoire de travail

Travail sur les données: visualisation des données

- Histogramme
  - Code: `hist(data$Var1,c="grey",main="Histogramme Var1",ylab="Frequence",xlab="valeurs Var1")`

- Boite à moustache
  - Code: `boxplot(data)` # l'ensemble des variables
  - Code: `boxplot(data$Var1)` # une variable

- Graph (X,Y) spécifique
  - Code: `plot(data$Var1,data$Var2)` #spécifique 2 variables choisies

- Graph (X,Y) plusieurs données
  - Code: `pairs(data)` #toutes les variables
  - Attention pairs est limité par la capacité d'affichage (autour de 20 variables maximum)

- Graphiques améliorés: beaucoup de packages existants
  - Package: « FactoMine R »
    - Lien vidéos Youtube: https://www.youtube.com/watch?v=1QPRsg3Bxok
    - Lien blog du développeur: http://factominer.free.fr/
  - Package ggplot2: https://thinkr.fr/pdf/ggplot2-french-cheatsheet.pdf

Travail sur les données: définir les contraintes pour les modèles linéaires

- Modèle linéaire = anova et régression linéaire multiple (modèle de prédiction / analyse STEP-WISE)

- Contraintes existantes:
  - contr.treatment: compare l'écart des moyennes de groupes à la moyenne d'un groupe de référence (spécifié par la base, le niveau de base est le 1er facteur)
  - contr.helmert: renvoie les contrastes entre le 2nd niveau et le 1er puis entre le 3ème et la moyenne des deux 1ers…
  - contr.poly : renvoie les contrastes sur la base de polynômes orthogonaux
  - **contr.sum** : compare l'écart des moyennes de groupes à la moyenne générale
  - contr.SAS: utilise les mêmes contraintes que SAS (le niveau de base est le dernier facteur)

- Code: `options("contrasts")` ## pour voir les contraintes actuelles

  `options(contrasts=c("contr.sum","contr.sum"))` # pour imposer vos contraintes

Travail sur les données: tests statistiques

- Vérification de la normalité et de l'homoscédasticité des résidus
  - Code: tests de normalité

```
R1 <- aov(Var1~Traitement, data= data)
res<-residuals(R1)
shapiro.test(res) # test de Shapiro
ks.test(res,"plnorm") # test de Kolmogorov-Smirnov
```

  - Code: test des variances

```
bartlett.test(data,residus) # si un seul facteur
library(car)
leveneTest(res~data$Genotype*data$Traitement) # si plusieurs facteurs
```

- Transformation des données
  - Code: pour centrer et réduire les données (nécessaire pour des ACP par exemple)

```
data2=t(scale(t(data),center=TRUE,scale=TRUE)) # penser à enlever les colonnes de texte (facteurs)
```

Travail sur les données: tests statistiques

- Tests paramétriques
  - Code: t.test(data$Var1, y=c(1)) # test de Student, nécessite de grand effectif (n > 30)
  - Code: aov(Var1~Traitement, data= data) # anova 1 facteur, échantillons indépendants
    aov(Var1 ~Genotype + Traitement+ Traitement%in%Genotype, data= data)
    # anova 2 facteurs

- Tests non paramétriques : test de rang principalement
  - Code: test de Wilcoxon (échantillons dépendants)
    wilcox.test(data$Var1,data$ID, conf.level=0.95) # ne fonctionne qu'avec un seul facteur
  - Code: test de Kruskal-Wallis (échantillons indépendants)
    kruskal.test(data$Var1~data$ID) # ne fonctionne qu'avec un seul facteur
  - Code: data$ID<-interaction(data$Genotype, data$Traitement)# création d'un facteur unique pour le test

Travail sur les données: comparaisons multiples

- Pour tests paramétriques
  - Code:
```
R1 <- aov(Var1~ID, data= data)
require(laercio) # package spécifique
LDuncan(R1) # test de Duncan (test de rang)
```
  - Code:
```
TukeyHSD(R1,ordered=TRUE) # test de Tukey (toutes les comparaisons 2à 2)
require(graphics)
plot(TukeyHSD(R1)) # graphique
```

- Pour tests non paramétriques
  - Code:
```
pairwise.wilcox.test(data$Var1, data$ID, p.adjust.method="bonf")
```
  - Code:
```
library(pgirmess)
kruskalmc(data$Var1~data$ID, data=data, probs=0.05)
```

Travail sur les données: corrélations

- Corrélations
  - Code: cor.test(data$Var1,data$Var2,
    alternative = c("two.sided", "less", "greater"),
    method = c("pearson", "kendall", "spearman"), # test à choisir
    exact = NULL, conf.level = 0.95, continuity = FALSE) # corrélation deux à deux

  - Code: library(Hmisc) # package spécifique
    rcorr(as.matrix(data2), type = "pearson") # matrice de corrélations avec test probabilité, nécessité d'avoir normalisé les données par réduction

  - Code: library(corrplot)
    M<-cor(data2)
    corrplot(M, type="upper", order="hclust", tl.col="black", tl.srt=45) # visualisation graphique

  - Code: library(ppcor)
    pcor(data2, method = c("pearson", "kendall", "spearman")) # corrélation partielle permet de connaître la valeur de la corrélation entre deux variables A et B, si la variable C était demeurée constante pour la série d'observations considérées

Travail sur les données: analyse step-wise

- Analyse STEP-WISE = modèle de régression linéaire, permet de définir le plus petit modèle (c'est-à-dire avec le moins de variables possibles) expliquant la variabilité de la variable principale

- Exemple: je souhaite déterminer les composants majeurs de la teneur en matières sèches
  - Code: # penser à définir les bons contrastes pour le modèle linéaire (contr.sum)

```
library(nlme)
library(lme4)
library(MASS)
X<-lm(X.MS ~ Glucose + Fructose + Saccharose + Ac.citrique + Ac.malique + Ac.quinique + luteine + lycopene + beta.carotene + phytoene + VitCtot + VitCred,data=data)
Y<-stepAIC(X,direction=c("both")) # ou forward ou backward
summary(Y)
```

- Le modèle ayant le plus petit critère AIC est le meilleur modèle explicatif de la variable
- Si vous avez plus de 20 variables, il faudra utiliser le critère BIC (analyse bayesienne)

Programmation: boucles/conditions

- Boucle « FOR » :
  - Code: i=1
    for (i in c(3:22)) # mettre le nombre de colonne du fichier
    { code }
- Boucle « WHILE » :
  - Code: str<- c(file1,file2,file3,…) # suppose que les fichiers soient au même format
    i=1
    while(i<=length(str)){
    code }
- Conditions « IF » « ELSE »
  - Code: if (p.value<0.05) { code
    } else { code }

- Enregistrement des sorties des analyses dans un fichier texte
  - Code: sink(file="Nom_fichier.txt", append = F) # crée le fichier à enregistrer
    code
    sink() #fermeture du fichier
  - Penser à utiliser la fonction print() pour « imprimer » les résultats souhaités dans le fichier de sortie

Astuces

- Penser à ordonner vos scripts notamment si vous utilisez des boucles: faites les appels des « library » avant d'exécuter la boucle pour éviter les latences

- Lister vos fichiers en amont de votre code si vous devez revenir souvent dessus

- Annoter vos scripts directement dans le texte avec #

- Utiliser les flèches pour revenir sur vos lignes de code précédemment lues dans la fenêtre de sortie si vous souhaitez la relire/modifier plus rapidement

- Utiliser les touches Ctrl+R ou Ctrl+Entrée selon la version de Rstudio pour lire directement vos lignes de code soit les unes après les autres soit toutes d'un coup en sélectionnant les lignes au préalable dans votre fenêtre de script

Exercice

- Test sur un jeu de données de 20 variables quantitatives (en colonne) pour 2 génotypes dans deux conditions témoin et stress

- Ecrivez une boucle permettant de :
  - Déterminer si les conditions d'application de l'anova sont respectées pour chaque variable
  - Sortir les résultats de l'anova si les conditions sont valables
  - Sinon sortir les résultats d'un test non paramétrique (Kruskal-Wallis)

Complément

- Première étape: écrire une boucle

```
data$ID<-interaction(data$Genotype,data$Traitement)
i=1
for (i in c(3:22))  # mettre le nombre de colonne du fichier
{ x<-data[,i] # x sert à définir vos différentes colonnes
R1<-aov(x~data$ID,data=data)
aov1<-anova(R1)
print(aov1)}  # fonction print pour afficher les résultats
```

- Deuxième étape: poser une condition

```
R1<-aov(Diam_moyen~data$ID,data=data)
res<-residuals(R1)
shapiro.test(res)
p.value<-shapiro.test(res)$p.value # on récupère la p-value du test
Leven<-leveneTest(res~data$Genotype*data$Traitement)
p.value2<-Leven$`Pr(>F)`[1] # idem
if (p.value>0.05 & p.value2>0.05) { LDuncan(R1)
} else { Kruskal.test(data$Diam_moyen~data$ID,data=data) }
```

Réponse

```
data <-read.delim(file1, dec=".")
library(laercio)
library(pgirmess)
library(car)
options(contrasts=c("contr.sum","contr.sum"))
sink(file=("Resultats_analyse.txt", append=F)
names(data)
i=1
for (i in c(3:22))
{
 x<-data[ ,i]
 print(i)
 R1 <- aov(x ~ Genotype+Traitement+Traitement%in%Genotype,  data= data)
 aov1<-anova(R1)
 print(aov1)
 res<-residuals(R1)
 P<-shapiro.test(res)$p.value
 Leven<-leveneTest(res~data$Genotype*data$Traitement)
 p.value2<-Leven$`Pr(>F)`[1]
 if (p.value2>0.05 & P>0.05) { print(LDuncan(R1))
 }  else { data$ID<-interaction(data$Genotype,data$Traitement)
 KW<-kruskal.test(x~data$ID)
 p.value3<-KW$p.value
 if (p.value3<0.05) { print(kruskalmc(x~data$ID, data=NULL, probs=0.05))
 } else { print("Aucune  différence stat !!!")}
 }}
sink()
```