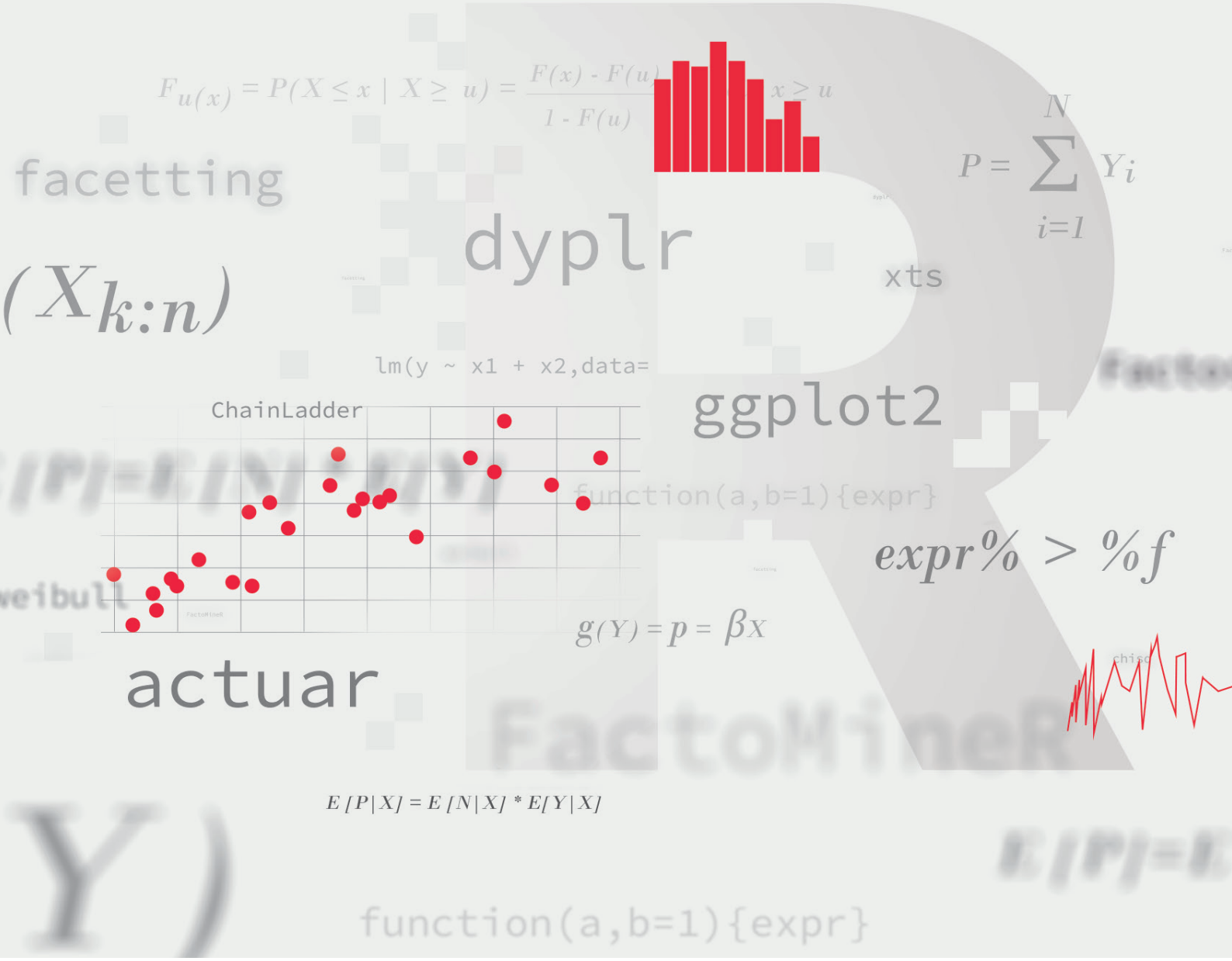
IARD:TARIFER SOUS R

Méthode et pratiques

Annexe



la pratique stratégique,
la vision opérationnelle

SeaBird

Sommaire

ANNEXE À LA PARTIE 1 - LE LANGAGE AU SERVICE DE LA DONNÉE **4**
DATAMINING 4
Chargement des packages et de la base de données 4
Fonctions utiles 4
Enchaîner les opérations 9
Utilisation de pipeline 10
Autres opérations 11
Gestion des valeurs manquantes 12
DATAVISUALISATION 14
Introduction à Ggplot2 14
Personnalisation 18
Représentation de plusieurs geom 18
Utilisation d'esquisse 19

ANNEXE À LA PARTIE 2 - DE LA MODÉLISATION À LA TARIFICATION **20**
Pré-requis à la modélisation 20
Classification 20
Méthode ARIMA 27
Modélisation linéaire généralisée 32

Rédaction
Saïda Derkaoui

LE LANGAGE AU SERVICE DE LA DONNÉE

DATAMINING

Chargement des packages et de la base de données

```
library(tidyverse)

library(dplyr)
library(ggplot2)

dsmall <- diamonds[sample(nrow(diamonds), 100), ]

df<-dsmall
knitr::kable(head(df))
```

carat	cut	color	clarity	depth	table	price	x	y	z
0.91	Premium	H	VS2	60.6	58	4293	6.30	6.25	3.80
0.36	Ideal	E	VS2	61.7	57	684	4.55	4.59	2.82
1.01	Premium	G	VS1	61.7	59	5898	6.38	6.33	3.92
0.71	Very good	F	I1	63.2	55	1448	5.64	5.57	3.54
0.60	Fair	F	SI1	55.3	63	1367	5.67	5.61	3.12
1.19	Ideal	H	SI2	62.0	57	4914	6.77	6.81	4.21

Fonctions utiles

```
### sélection la ligne 3
knitr::kable(slice(df,3))
```

carat	cut	color	clarity	depth	table	price	x	y	z
1.01	Premium	G	VS1	61.7	59	5898	6.38	6.33	3.92

```
### sélection la ligne 3 à 6
knitr::kable(slice(df,3:6))
```

carat	cut	color	clarity	depth	table	price	x	y	z
1.01	Premium	G	VS1	61.7	59	5898	6.38	6.33	3.92
0.71	Very Good	F	I1	63.2	55	1448	5.64	5.57	3.54
0.60	Fair	F	SI1	55.3	63	1367	5.67	5.61	3.12
1.19	Ideal	H	SI2	62.0	57	4914	6.77	6.81	4.21

```
###filtrer
knitr::kable(head(filter(df,color==«E»)))
```

carat	cut	color	clarity	depth	table	price	x	y	z
0.36	Ideal	E	VS2	61.7	57	684	4.55	4.59	2.82
0.46	Premium	E	VS2	61.3	57	1156	5.02	4.99	3.07
1.01	Good	E	SI2	60.7	62	4436	6.43	6.49	3.92
0.31	Ideal	E	VVS2	61.6	57	1046	4.37	4.33	2.68
1.02	Very Good	E	SI1	58.9	58	5232	6.51	6.54	3.84
0.31	Premium	E	SI1	59.2	60	698	4.42	4.36	2.60

```
knitr::kable(head(filter(df,price>100)))
```

carat	cut	color	clarity	depth	table	price	x	y	z
0.91	Premium	H	VS2	60.6	58	4293	6.30	6.25	3.80
0.36	Ideal	E	VS2	61.7	57	684	4.55	4.59	2.82
1.01	Premium	G	VS1	61.7	59	5898	6.38	6.33	3.92
0.71	Very Good	F	I1	63.2	55	1448	5.64	5.57	3.54
0.60	Fair	F	SI1	55.3	63	1367	5.67	5.61	3.12
1.19	Ideal	H	SI2	62.0	57	4914	6.77	6.81	4.21

```
knitr::kable(head(filter(df,price==max(price))))
```

carat	cut	color	clarity	depth	table	price	x	y	z
2.08	Ideal	H	SI1	58.7	60	18760	8.36	8.4	4.92

```
###select
knitr::kable(head(select(df,cut)))
```

cut
Premium
Ideal
Premium
Very Good

cut
Fair
Ideal

```
knitr::kable(filter(select(df,2:5)))
```

cut	color	clarity	depth
Premium	H	VS2	60.6
Ideal	E	VS2	61.7
Premium	G	VS1	61.7
Very Good	F	I1	63.2
Fair	F	SI1	55.3
Ideal	H	SI2	62.0
Premium	I	VS2	59.1
Good	J	VVS2	63.3
Ideal	H	SI1	58.7
Fair	I	VS2	66.8
Good	G	SI2	63.3
Ideal	I	VS2	60.8
Premium	E	VS2	61.3

```
knitr::kable(filter(select(df,-2:-5)))
```

carat	table	price	x	y	z
0.91	58	4293	6.30	6.25	3.80
0.36	57	684	4.55	4.59	2.82
1.01	59	5898	6.38	6.33	3.92
0.71	55	1448	5.64	5.57	3.54
0.60	63	1367	5.67	5.61	3.12
1.19	57	4914	6.77	6.81	4.21
0.90	59	3385	6.28	6.24	3.70
1.50	57	8276	7.25	7.28	4.60
2.08	60	18760	8.36	8.40	4.92
0.46	55	855	4.82	4.77	3.20
0.30	57	394	4.25	4.28	2.70
1.32	57	7079	7.02	7.07	4.28
0.46	57	1156	5.02	4.99	3.07

```
knitr::kable(head(select(df,-cut:-price)))
```

carat	x	y	z
0.91	6.30	6.25	3.80
0.36	4.55	4.59	2.82
1.01	6.38	6.33	3.92
0.71	5.64	5.57	3.54
0.60	5.67	5.61	3.12
1.19	6.77	6.81	4.21

```
###arrange
knitr::kable(head(arrange(df,price)))
```

carat	cut	color	clarity	depth	table	price	x	y	z
0.30	Good	G	SI2	63.3	57	394	4.25	4.28	2.70
0.23	Very Good	F	VVS2	60.0	57	530	4.00	4.04	2.41
0.36	Very Good	G	SI1	61.8	60	537	4.53	4.57	2.81
0.40	Ideal	H	SI2	62.2	55	540	4.75	4.77	2.96
0.32	Ideal	G	VS2	61.8	55	561	4.41	4.43	2.73
0.40	Good	J	VS1	64.0	56	567	4.66	4.69	2.99

```
knitr::kable(head(arrange(df,desc(price))))
```

carat	cut	color	clarity	depth	table	price	x	y	z
2.08	Ideal	H	SI1	58.7	60	18760	8.36	8.40	4.92
1.51	Ideal	G	VVS2	61.8	57	14982	7.31	7.36	4.53
1.52	Premium	G	VS2	62.8	60	13768	7.26	7.22	4.55
1.54	Ideal	H	VVS2	62.6	56	12061	7.35	7.42	4.62
1.53	Premium	G	SI1	62.3	58	11421	7.35	7.30	4.56
1.51	Ideal	H	VS2	64.2	59	10959	7.22	7.16	4.62

```
###mutate
knitr::kable(head(mutate(df,com=price*0,10)))
```

carat	cut	color	clarity	depth	table	price	x	y	z	com	10
0.91	Premium	H	VS2	60.6	58	4293	6.30	6.25	3.80	0	10
0.36	Ideal	E	VS2	61.7	57	684	4.55	4.59	2.82	0	10
1.01	Premium	G	VS1	61.7	59	5898	6.38	6.33	3.92	0	10
0.71	Very Good	F	I1	63.2	55	1448	5.64	5.57	3.54	0	10
0.60	Fair	F	SI1	55.3	63	1367	5.67	5.61	3.12	0	10
1.19	Ideal	H	SI2	62.0	57	4914	6.77	6.81	4.21	0	10

Enchaîner les opérations

```
###Enchainer les opérations
knitr::kable(arrange(select(filter(df,cut==«Ideal»),cut:price),price))
```

cut	color	clarity	depth	table	price
Ideal	H	SI2	62.2	55	540
Ideal	G	VS2	61.8	55	561
Ideal	G	VS1	61.5	56	625
Ideal	G	IF	61.2	57	627
Ideal	H	VVS1	61.9	57	665
Ideal	H	VS1	62.1	55	672
Ideal	E	VS2	61.7	57	684
Ideal	I	IF	61.5	55	715
Ideal	E	VS2	61.2	56	723
Ideal	D	VS2	62.0	54	776
Ideal	D	VVS2	61.4	55	777
Ideal	G	VS2	1.3	56	807
Ideal	G	VS2	62.5	54	827

```
#ou
a=filter(df,cut==«Ideal»)
b=select(a,cut:price)
knitr::kable(arrange(b,desc(price)))
```

Utilisation de pipeline

```
df %>% filter(cut==«Ideal») %>% select(cut:price) %>% ar-
range(price)
## # A tibble: 44 × 6
## cut color clarity depth table price
## <ord> <ord> <ord> <dbl> <dbl> <int>
## 1 Ideal H SI2 62.2 55 540
## 2 Ideal G VS2 61.8 55 561
## 3 Ideal G VS1 61.5 56 625
## 4 Ideal G IF 61.2 57 627

## 5 Ideal H VVS1 61.9 57 665
## 6 Ideal H VS1 62.1 55 672
## 7 Ideal E VS2 61.7 57 684
## 8 Ideal I IF 61.5 55 715
## 9 Ideal E VS2 61.2 56 723
## 10 Ideal D VS2 62 54 776
## # ... with 34 more rows


#ou
df %>%
filter(cut==«Ideal»)%>%
select(cut:price)%>%
arrange(price)


## # A tibble: 44 × 6
## cut color clarity depth table price
## <ord> <ord> <ord> <dbl> <dbl> <int>
## 1 Ideal H SI2 62.2 55 540
## 2 Ideal G VS2 61.8 55 561
## 3 Ideal G VS1 61.5 56 625
## 4 Ideal G IF 61.2 57 627
## 5 Ideal H VVS1 61.9 57 665
## 6 Ideal H VS1 62.1 55 672
## 7 Ideal E VS2 61.7 57 684
## 8 Ideal I IF 61.5 55 715
## 9 Ideal E VS2 61.2 56 723
## 10 Ideal D VS2 62 54 776
## # ... with 34 more rows
```

Autres opérations

```
### group_by

knitr::kable(head(df%>%
group_by(price=500,cut=«Ideal»)%>%
summarise(moy_prof = mean(depth))))
```

price	cut	mony_prof
500	Ideal	61.663

```
knitr::kable(head(df%>%
group_by(cut,color,clarity)%>%
arrange(desc(table),.by_group = TRUE)))
```

carat	cut	color	clarity	depth	table	price	x	y	z
0.60	Fair	F	SI1	55.3	63	1367	5.67	5.61	3.12
1.21	Fair	H	SI2	65.6	56	3862	6.69	6.59	4.36
0.46	Fair	I	VS2	66.8	55	855	4.82	4.77	3.20
1.50	Fair	J	SI2	65.5	60	5268	7.07	7.03	4.62
1.01	Good	E	SI2	60.7	62	4436	6.43	6.49	3.92
0.90	Good	F	SI2	64.3	57	3210	6.16	6.06	3.93

```
### summarise

knitr::kable(df%>%
summarise(moy_prix = mean(price), Var_carat = var(carat)*100))
```

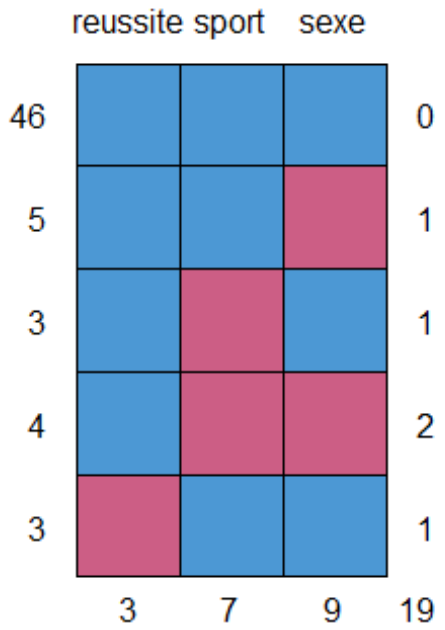
moy_prix	Var_carat
3519.68	19.11446

```
###Exemple du guide

knitr::kable(head(df%>%
group_by(cut,color,clarity)%>%
summarise(moy_prix = mean(price), Var_carat = var(carat)*100)))
```

cut	color	clarity	moy_prix	Var_carat
Fair	F	SI1	1367	NA
Fair	H	SI2	3862	NA
Fair	I	VS2	855	NA
Fair	J	SI2	5268	NA
Good	E	SI2	4436	NA
Good	F	SI2	3210	NA

Gestion des valeurs manquantes

```
reussite <- c(3,3,1,2,3,2,1,3,3,3,NA,2,5,NA,0,rnom(20,5,3),7,6,3,4,10,
NA,3,3,4,5,6,4,3,3,5,6,4,3,3,4,5,3,4,8,6,2) ;length(reussite)

## [1] 61

sexe <- c(NA,«F»,rep(«F»,29),NA,NA,«F»,rep(«M»,15),NA,NA,NA,«M»,«F»,
«F»,«F»,
NA,NA,NA,«M»,«M»);length(sexe)

## [1] 61

sport <- c(NA,«non»,«oui»,rep(«non»,9),«oui»,rep(«oui»,10),rep(«non»,
7),rep(«oui»,12),«non»,NA,rep(«pte»,10),«pte»,«pte»,NA,NA,NA,NA,NA);
length(sport)

## [1] 61

df_2<-data.frame(reussite,sexe,sport)
# Détection
sum(is.na(df_2))

## [1] 19

##which(is.na(df_2),arr.ind = TRUE)

# Utilisation de la library mice
library(mice)
```

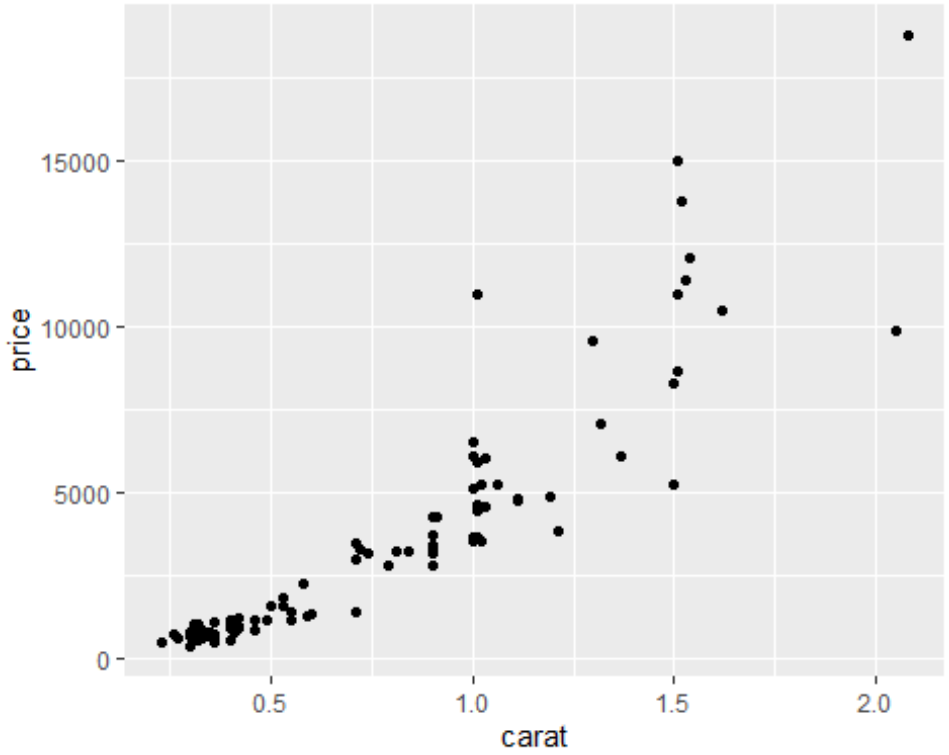



```
##     reussite  sport  sexe
## 46         1     1     1     0
## 5          1     1     0     1
## 3          1     0     1     1
## 4          1     0     0     2
## 3          0     1     1     1
##            3     7     9     19
```

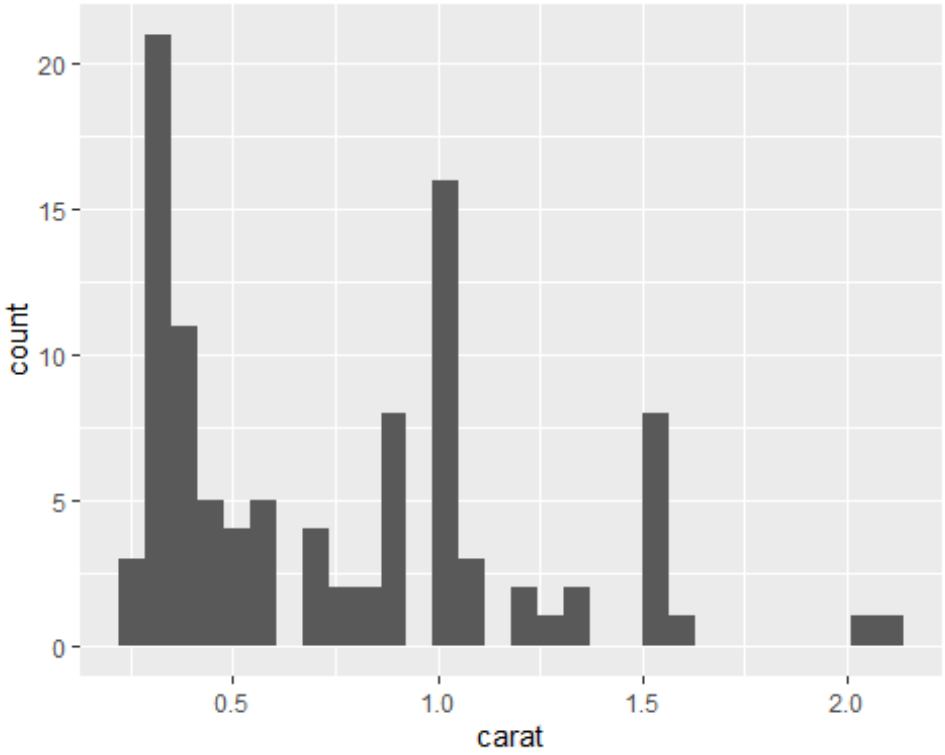
DATAVISUALISATION

Introduction à Ggplot2

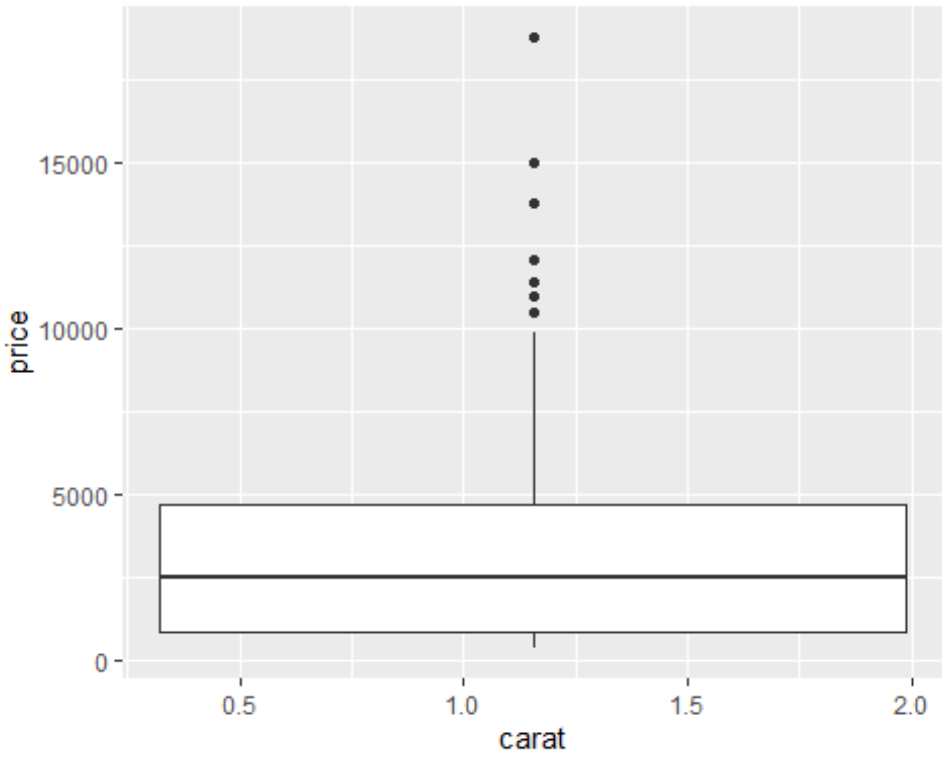
```
ggplot(df)+
  geom_point(aes(x=carat,y=price))
```



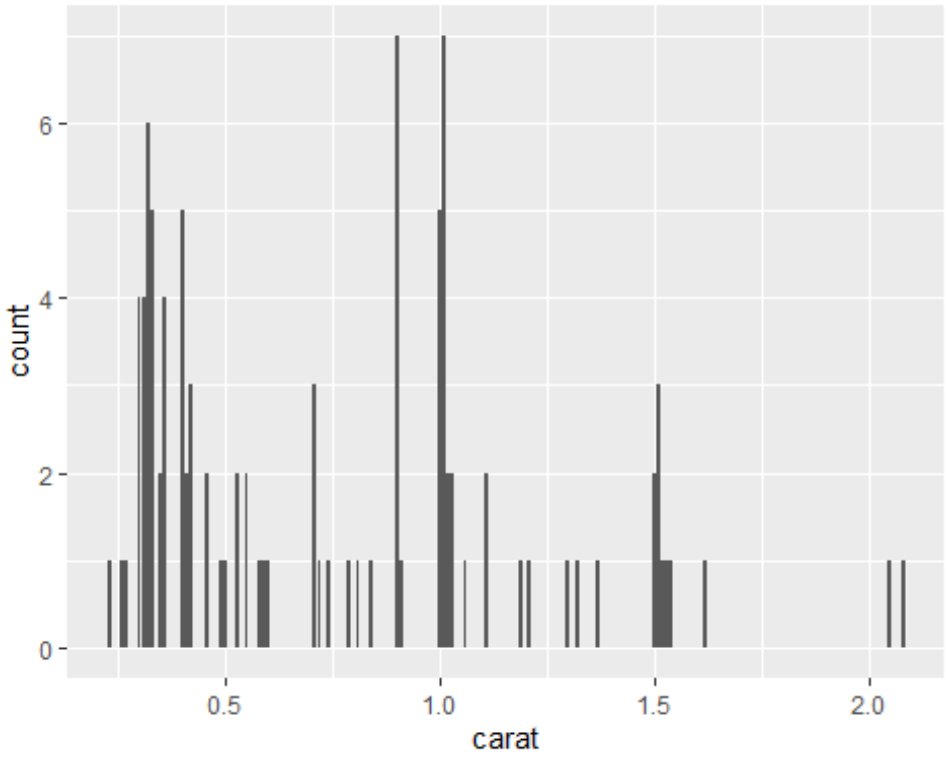

```
ggplot(df)+
  geom_histogram(aes(x=carat),bins = 30)
```



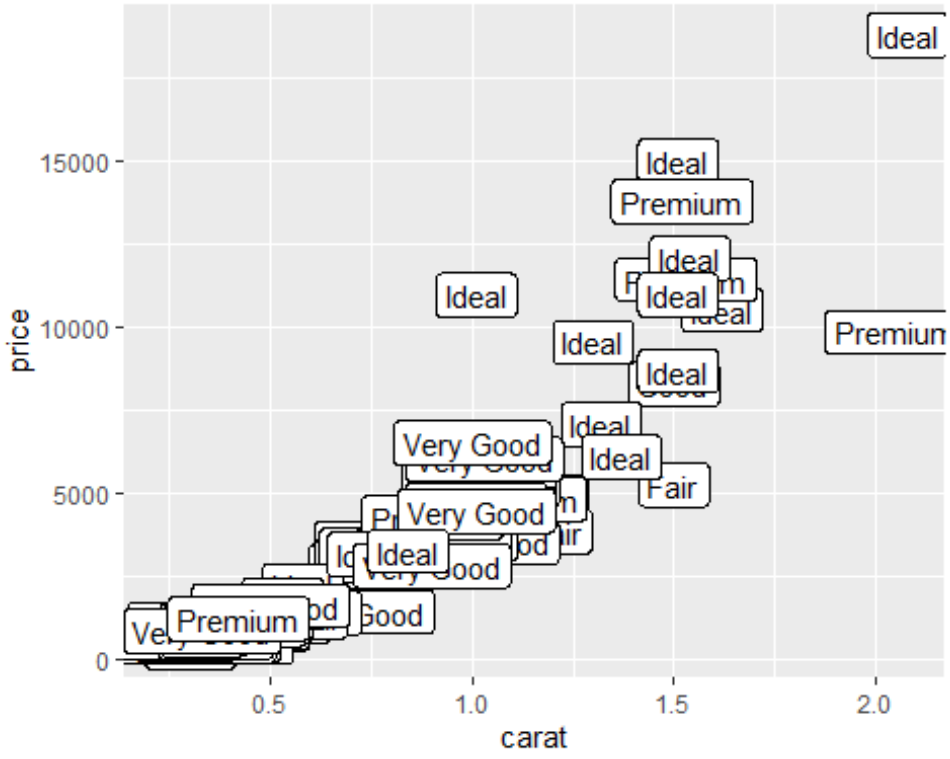

```
ggplot(df)+
   geom_boxplot(aes(x=carat,y=price))
```



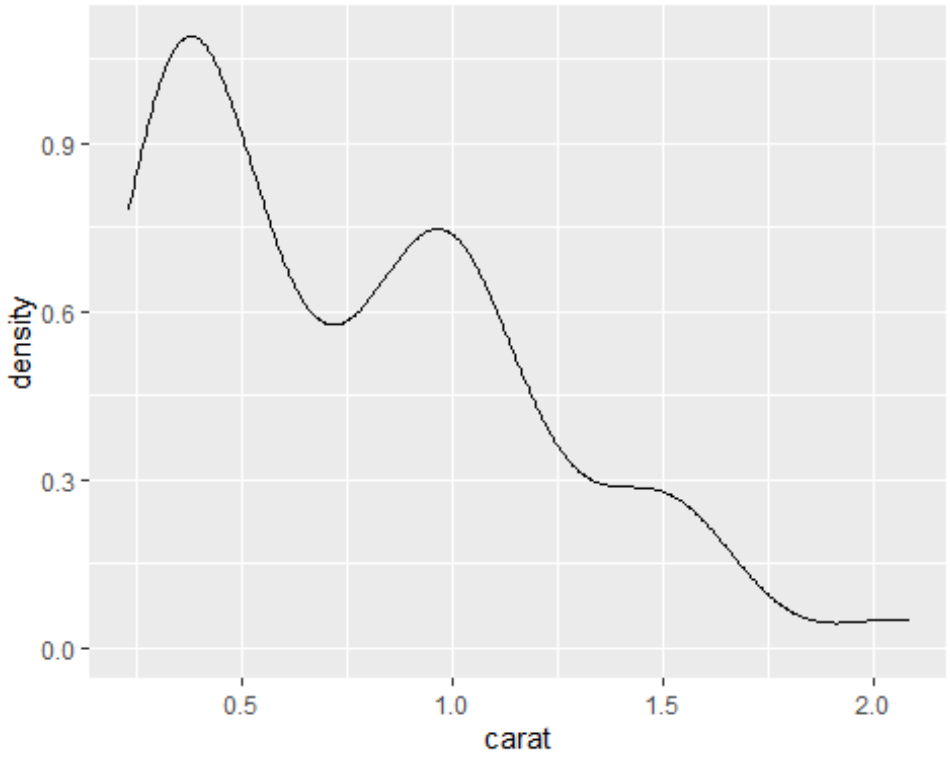
```
ggplot(df)+
   geom_bar(aes(x=carat))
```



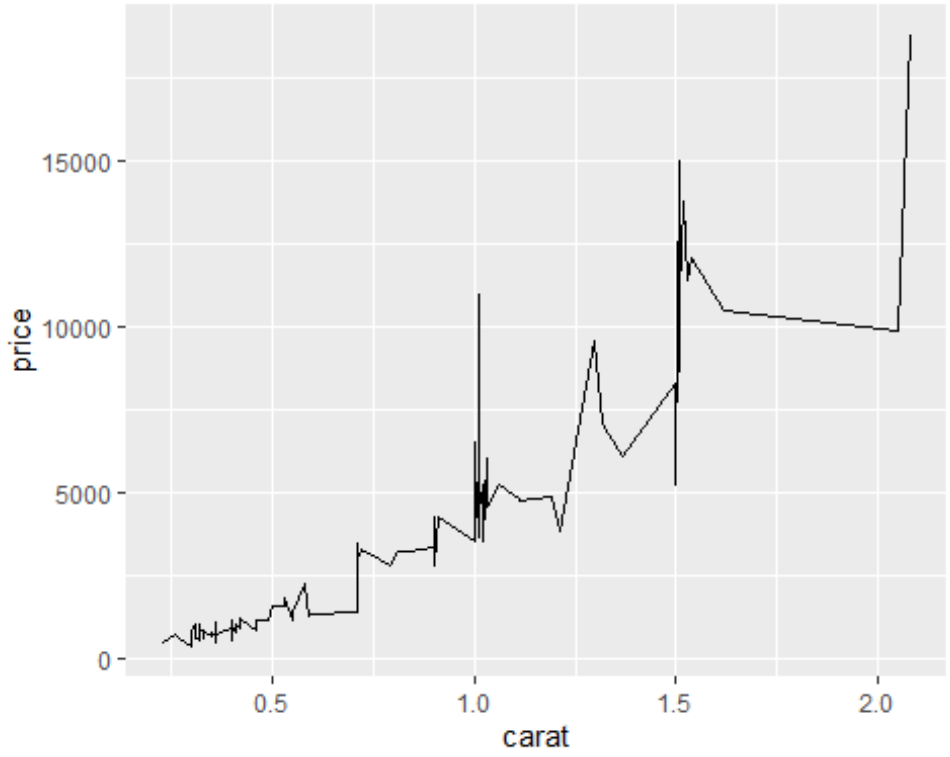

```
ggplot(df)+
   geom_label(aes(x=carat,y=price,label=cut))
```



```
ggplot(df)+
   geom_density(aes(x=carat))
```

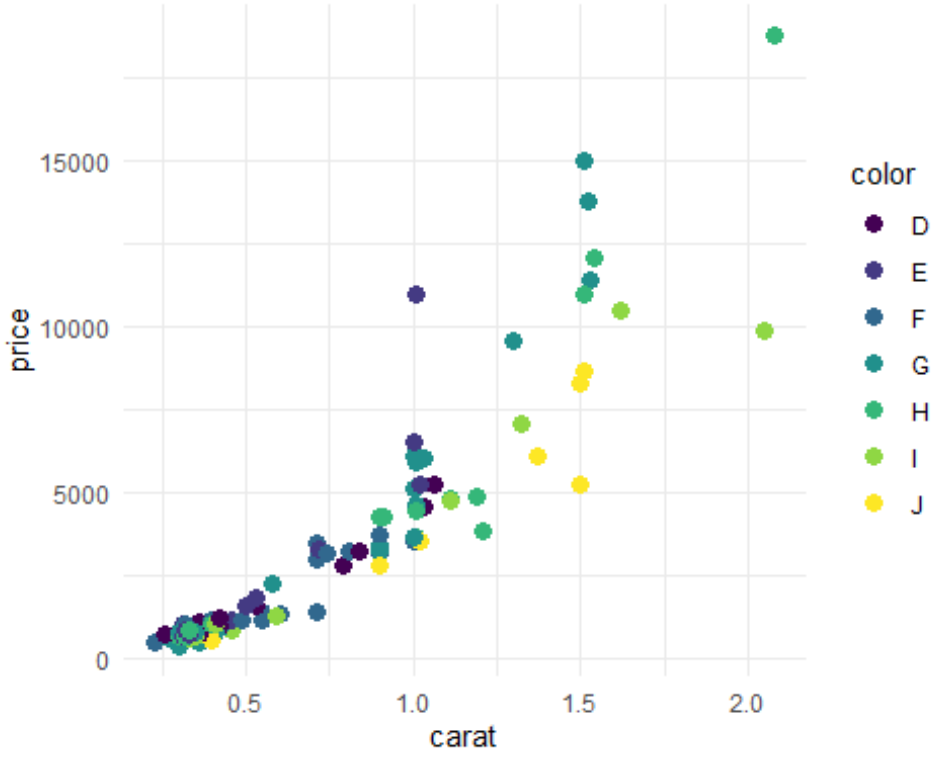



```
ggplot(df)+
   geom_line(aes(x=carat,y=price))
```



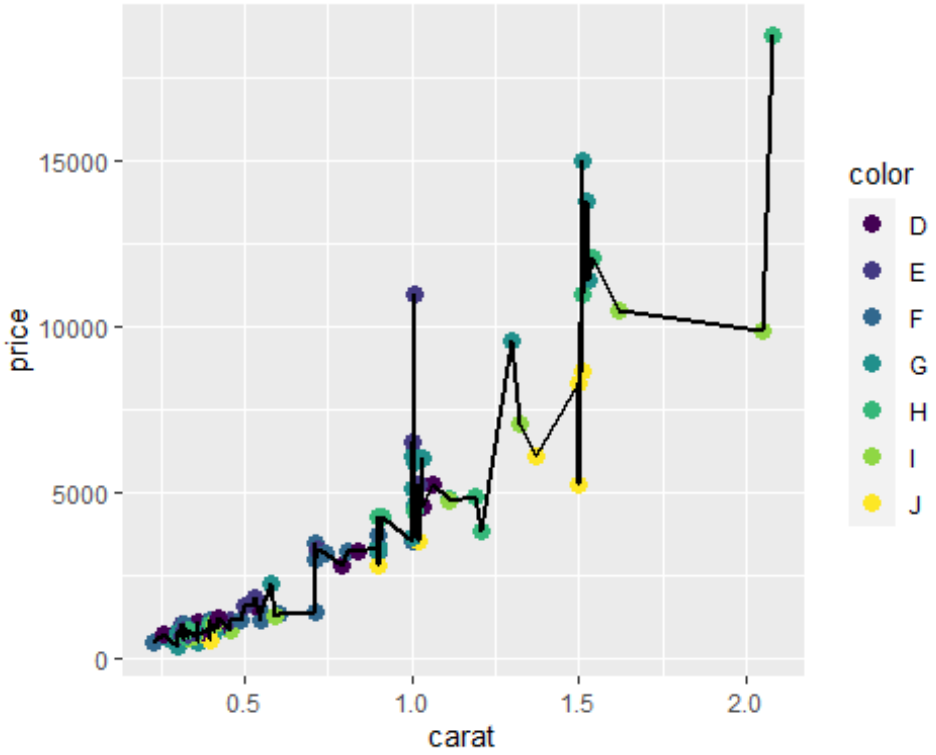
Personnalisation

```
ggplot(df) +
      aes(x = carat, y = price, colour = color) +
      geom_point(shape = «circle», size = 3) +
      theme_minimal()
```



Représentation de plusieurs geom

```
ggplot(df) +
      aes(x = carat, y = price, colour = color) +
      geom_point(shape = «circle», size = 3)+
      geom_line(aes(y=price),size=1, colour = «black»)
```



Utilisation d'esquisse

```
#install.packages(«esquisse»)
#library(«esquisse»)
#esquisser()
```

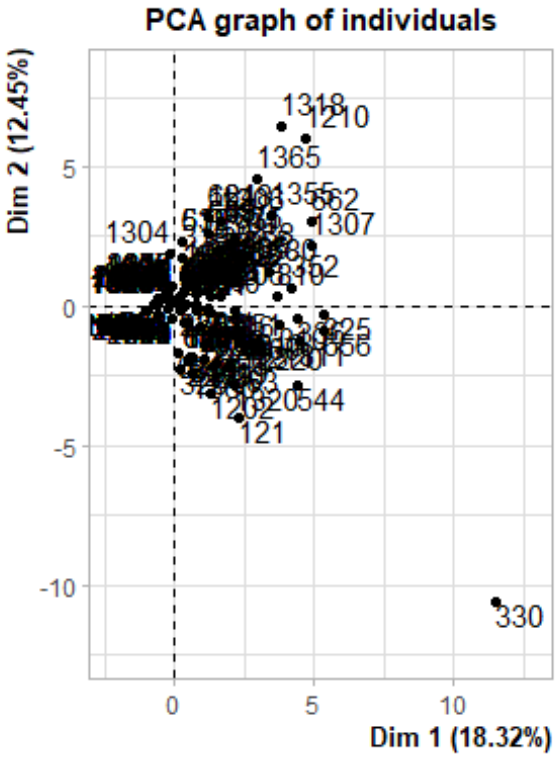
DE LA MODÉLISATION À LA TARIFICATION

Pré-requis à la modélisation

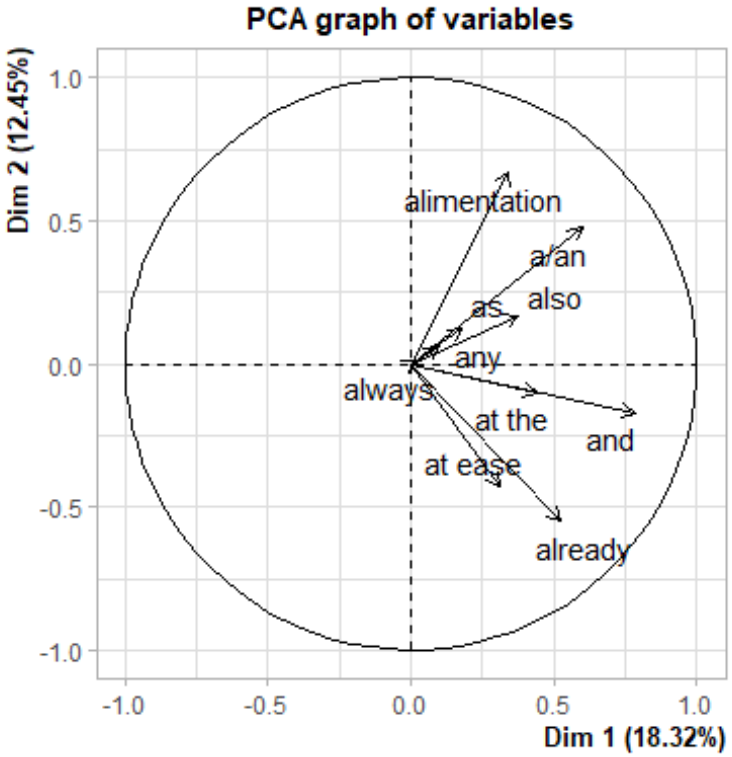
```
u1=runif(1000, 0, 1)
u2=punif(1000,0,1)
ks.test(u1,u2)
##
## Exact two-sample Kolmogorov-Smirnov test
##
## data: u1 and u2
## D = 1, p-value = 0.001998
## alternative hypothesis: two-sided

###teste si les deux échantillons de 1000 valeurs
###le premier d'une loi uniforme et le second d'une loi normale de
mÃame moyenne
###suivent la même distribution.

u3=runif(1000, 0, 4)
u4=rnorm(1000, 2, 1)
ks.test(u3,u4)

##
## Asymptotic two-sample Kolmogorov-Smirnov test
##
## data: u3 and u4
## D = 0.105, p-value = 3.258e-05
## alternative hypothesis: two-sided
```

Classification

```
### Methode Factominer

#install.packages("FactoMineR")
library(FactoMineR)
library(tidyverse)

data(health)
health_2<-health[,c(1:10)]
res.pca <- PCA(health_2, graph = FALSE,n=10)
plot.PCA(res.pca,choix = "ind")
```




```
plot.PCA(res.pca,choix = « var »)
```



```
knitr ::kable(res.pca$eig) # Valeurs propres
```

	eigenvalue	percentage of variance	cumulative percentage of variance
comp 1	1.8316719	18.316719	18.31672
comp 2	1.2447707	12.447707	30.76443
comp 3	1.1254514	11.254514	42.01894
comp 4	1.0854210	10.854210	52.87315
comp 5	1.0525056	10.525056	63.39821
comp 6	0.8933174	8.933174	72.33138
comp 7	0.8587440	8.587440	80.91882
comp 8	0.7885674	7.885674	88.80449
comp 9	0.6227945	6.227945	95.03244
comp 10	0.4967561	4.967561	100.00000

```
res.var <- res.pca$var
knitr::kable(res.var$coord) # Coordonnées
```

	Dim.1	Dim.2	Dim.3	Dim4	Dim.5	Dim.6	Dim.7	Dim.8	Dim.9	Dim.10
a/an	0.600 5330	0.474 6554	0.042 8289	- 0.110 4214	0.230 9853	- 0.192 3734	- 0.020 0351	- 0.036 7264	- 0.507 4714	0.2244 903
alimen tation	0.340 0888	0.667 0224	- 0.202 2089	- 0.288 4263	- 0.011 5399	- 0.042 4781	0.164 7571	0.283 0974	0.453 7053	- 0.0163 551
already	0.521 2365	- 0.549 9623	- 0.179 1362	- 0.196 0255	0.204 5793	- 0.141 6213	0.152 9492	- 0.283 1592	0.264 7395	0.3460 772
also	0.371 9065	0.166 2748	0.318 1268	0.102 2552	- 0.405 2237	0.610 5130	0.298 5991	- 0.302 5104	0.005 9625	0.0688 155
always	0.017 7432	0.008 8455	0.061 7467	0.230 4974	0.818 3339	0.495 1745	- 0.030 7480	0.149 3496	0.067 4253	0.0002 839
and	0.783 4149	- 0.173 2484	- 0.206 2732	0.051 6964	0.057 9855	- 0.035 8690	- 0.059 1887	- 0.152 1610	- 0.038 9814	- 0.5274 468
any	0.094 7556	0.057 0111	0.112 3882	0.787 4124	0.031 6923	- 0.369 5081	0.454 8769	0.075 5086	0.068 8750	0.0149 243
as	0.179 4946	0.120 6685	0.778 6260	0.038 3675	0.083 1791	- 0.232 4453	- 0.434 8464	- 0.171 9342	0.256 0039	- 0.0187 077

	Dim.1	Dim.2	Dim.3	Dim4	Dim.5	Dim.6	Dim.7	Dim.8	Dim.9	Dim.10
at ease	0.307 8168	- 0.431 0017	0.465 2658	- 0.282 8501	- 0.081 7009	0.011 5279	0.237 7638	0.589 5129	- 0.104 9685	- 0.0335 635
at the	0.442 0043	- 0.103 8334	- 0.260 2137	0.428 7890	- 0.324 5433	0.156 6474	- 0.511 5252	0.327 3166	0.043 9355	0.2041 772

```
knitr::kable(res.var$contrib ) # Contributions aux axes
```

	Dim.1	Dim.2	Dim.3	Dim4	Dim.5	Dim.6	Dim.7	Dim.8	Dim.9	Dim.10
a/an	0.6005 330	0.4746 554	0.0428 289	- 0.1104 214	0.2309 853	- 0.1923 734	- 0.0200 351	- 0.0367 264	- 0.5074 714	0.2244 903
alimen tation	0.3400 888	0.6670 224	- 0.2022 089	- 0.2884 263	- 0.0115 399	- 0.0424 781	0.1647 571	0.2830 974	0.4537 053	- 0.0163 551
already	0.5212 365	- 0.5499 623	- 0.1791 362	- 0.1960 255	0.2045 793	- 0.1416 213	0.1529 492	- 0.2831 592	0.2647 395	0.3460 772
also	0.3719 065	0.1662 748	0.3181 268	0.1022 552	- 0.4052 237	0.6105 130	0.2985 991	- 0.3025 104	0.0059 625	0.0688 155
always	0.0177 432	0.0088 455	0.0617 467	0.2304 974	0.8183 339	0.4951 745	- 0.0307 480	0.1493 496	0.0674 253	0.0002 839
and	0.7834 149	- 0.1732 484	- 0.2062 732	0.0516 964	0.0579 855	- 0.0358 690	- 0.0591 887	- 0.1521 610	- 0.0389 814	- 0.5274 468
any	0.0947 556	0.0570 111	0.1123 882	0.7874 124	0.0316 923	- 0.3695 081	0.4548 769	0.0755 086	0.0688 750	0.0149 243
as	0.1794 946	0.1206 685	0.7786 260	0.0383 675	0.0831 791	- 0.2324 453	- 0.4348 464	- 0.1719 342	0.2560 039	- 0.0187 077
at ease	0.3078 168	- 0.4310 017	0.4652 658	- 0.2828 501	- 0.0817 009	0.0115 279	0.2377 638	0.5895 129	- 0.1049 685	- 0.0335 635
at the	0.4420 043	- 0.1038 334	- 0.2602 137	0.4287 890	- 0.3245 433	0.1566 474	- 0.5115 252	0.3273 166	0.0439 355	0.2041 772

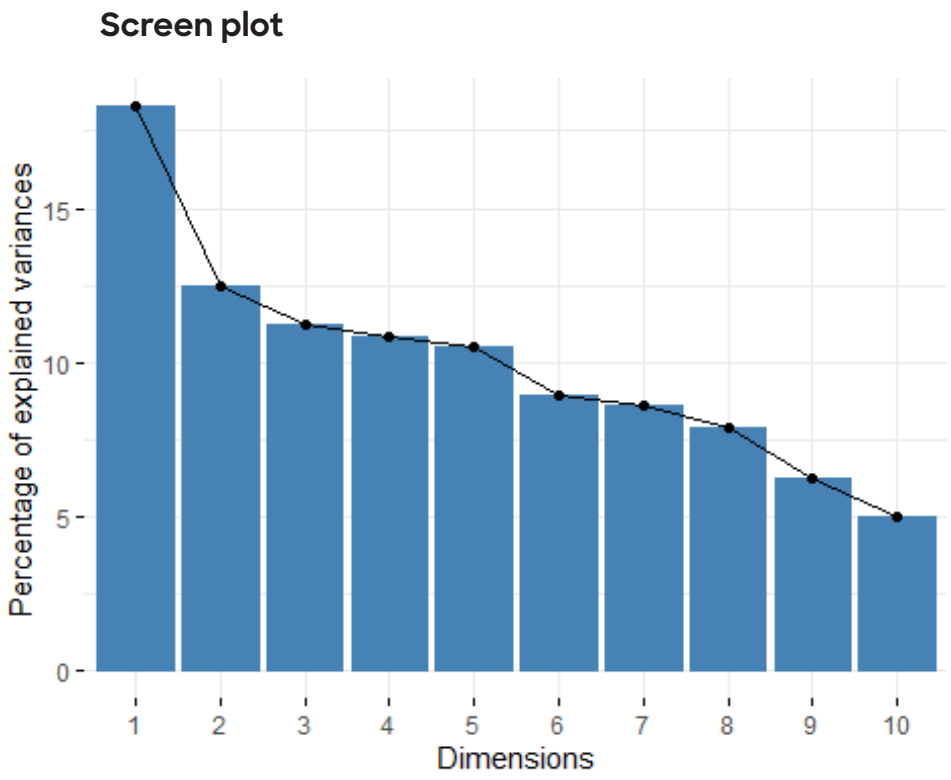
```
knitr::kable(res.ind$contrib ) # Contributions aux axes
```

	Dim.1	Dim.2	Dim.3	Dim4	Dim.5	Dim.6	Dim.7	Dim.8	Dim.9	Dim.10
a/an	19.689 1115	18.099 5382	0.1629 850	1.1233 329	5.0692 580	4.1427 081	0.0467 434	0.1710 482	41.350 2765	10.144 9982
alimen tation	6.3144 701	35.743 0415	3.6330 709	7.6642 830	0.0126 525	0.2019 871	3.1609 987	10.163 2559	33.052 3934	0.0538 469
already	14.832 7573	24.298 3313	2.8512 800	3.5401 945	3.9764 809	2.2451 795	2.7241 497	10.167 6955	11.253 6316	24.110 2998
also	7.5512 658	2.2210 759	8.9923 620	0.9633 244	15.601 4594	41.723 8217	10.382 7744	11.604 9093	0.0057 084	0.9532 987
always	0.0171 876	0.0062 858	0.3387 666	4.8947 885	63.626 3049	27.448 0050	0.1100 957	2.8285 864	0.7299 633	0.0000 162
and	33.507 0351	2.4112 881	3.7805 850	0.2462 192	0.3194 590	0.1440 236	0.4079 560	2.9360 813	0.2439 896	56.003 3632
any	0.4901 876	0.2611 138	1.1223 156	57.122 3837	0.0954 297	15.284 1796	24.094 8368	0.7230 256	0.7616 900	0.0448 377
as	1.7589 569	1.1697 653	53.868 0207	0.1356 219	0.6573 608	6.0483 324	22.019 5315	3.7487 417	10.523 2162	0.0704 524
at ease	5.1729 358	14.923 4261	19.234 2571	7.3707 993	0.6342 046	0.0148 763	6.5830 597	44.070 4832	1.7691 850	0.2267 728
at the	10.666 0922	0.8661 341	6.0163 572	16.939 0526	10.007 3902	2.7468 868	30.469 8539	13.586 1729	0.3099 460	8.3921 141

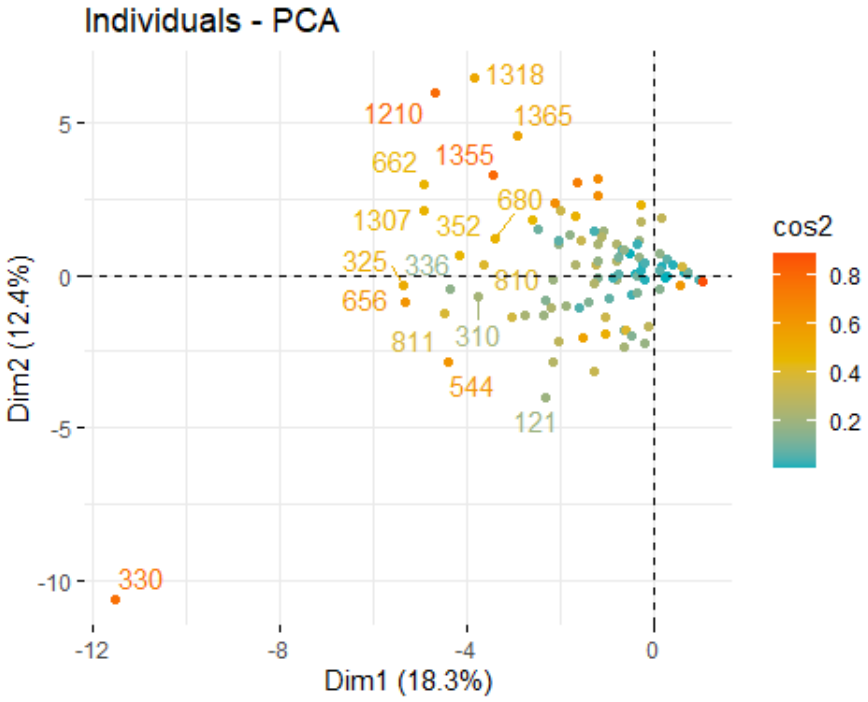
```
knitr::kable(res.ind$cos2 ) # Qualité de représentation
```

	Dim.1	Dim.2	Dim.3	Dim4	Dim.5	Dim.6	Dim.7	Dim.8	Dim.9	Dim.10
a/an	0.3606 399	0.2252 978	0.0018 343	0.0121 929	0.0533 542	0.0370 075	0.0004 014	0.0013 488	0.2575 272	0.0503 959
alimen tation	0.1156 604	0.4449 189	0.0408 884	0.0831 897	0.0001 332	0.0018 044	0.0271 449	0.0801 441	0.2058 485	0.0002 675
already	0.2716 874	0.3024 585	0.0320 898	0.0384 260	0.0418 527	0.0200 566	0.0233 935	0.0801 791	0.0700 870	0.1197 694
also	0.1383 144	0.0276 473	0.1012 047	0.0104 561	0.1642 062	0.3727 262	0.0891 614	0.0915 125	0.0000 356	0.0047 356
always	0.0003 148	0.0000 782	0.0038 127	0.0531 291	0.6696 704	0.2451 978	0.0009 454	0.0223 053	0.0045 462	0.0000 001
and	0.6137 389	0.0300 150	0.0425 486	0.0026 725	0.0033 623	0.0012 866	0.0035 033	0.0231 530	0.0015 196	0.2782 001
any	0.0089 786	0.0032 503	0.0126 311	0.6200 184	0.0010 044	0.1365 362	0.2069 130	0.0057 015	0.0047 438	0.0002 227
as	0.0322 183	0.0145 609	0.6062 584	0.0014 721	0.0069 188	0.0540 308	0.1890 914	0.0295 614	0.0655 380	0.0003 500
at ease	0.0947 512	0.1857 624	0.2164 722	0.0800 042	0.0066 750	0.0001 329	0.0565 316	0.3475 255	0.0110 184	0.0011 265
at the	0.1953 678	0.0107 814	0.0677 112	0.1838 600	0.1053 283	0.0245 384	0.2616 580	0.1071 361	0.0019 303	0.0416 883

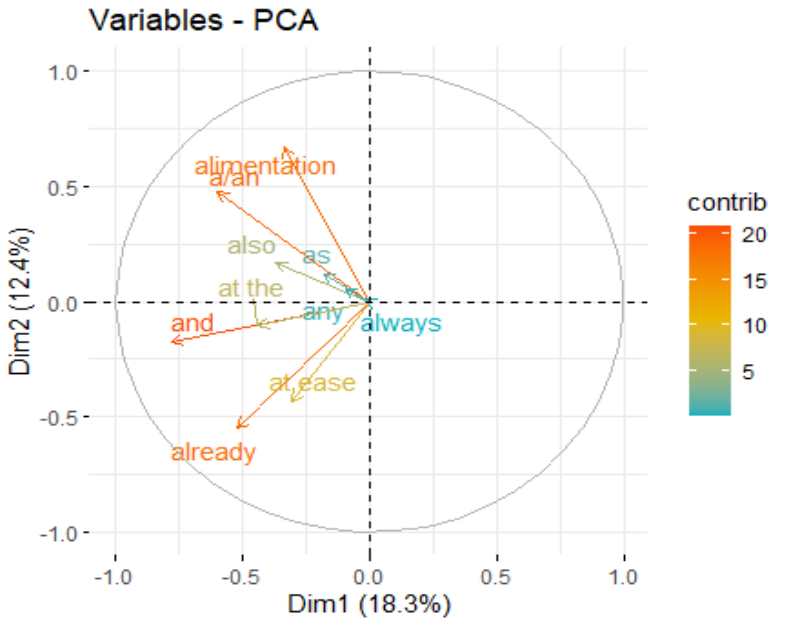
```
###Methode Ade4
#install.packages(«ade4»)
library(ade4)

library(factoextra)

library(magrittr)

res.pca <- dudi.pca(health_2,scannf = FALSE,nf=10)

fviz_eig(res.pca)
```



```
fviz_pca_ind(res.pca,
      col.ind = «cos2»,
      gradient.cols = c(«#00AFBB», «#E7B800», «#FC4E07»),
      repel = TRUE
      )
```

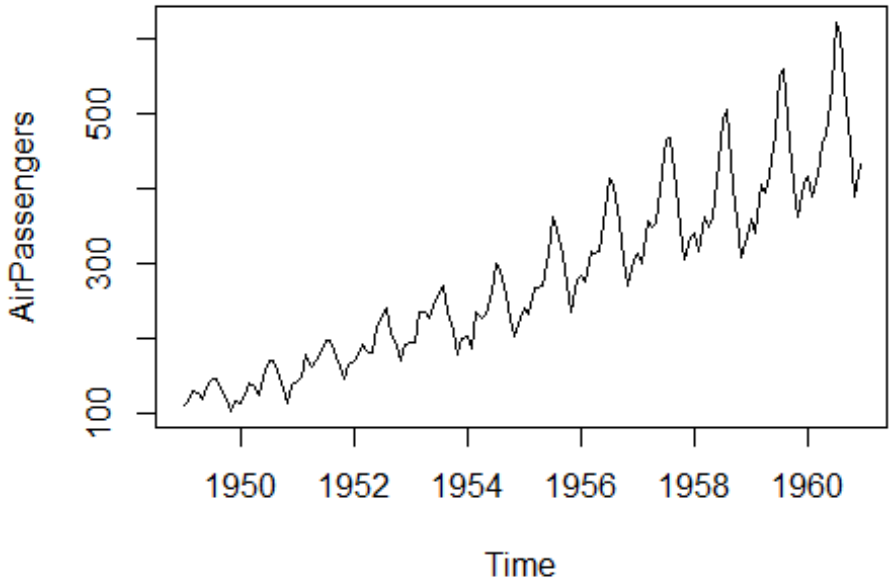



```
fviz_pca_var(res.pca,
      col.var = «contrib»,
      gradient.cols = c(«#00AFBB», «#E7B800», «#FC4E07»),
      repel = TRUE
      )
```

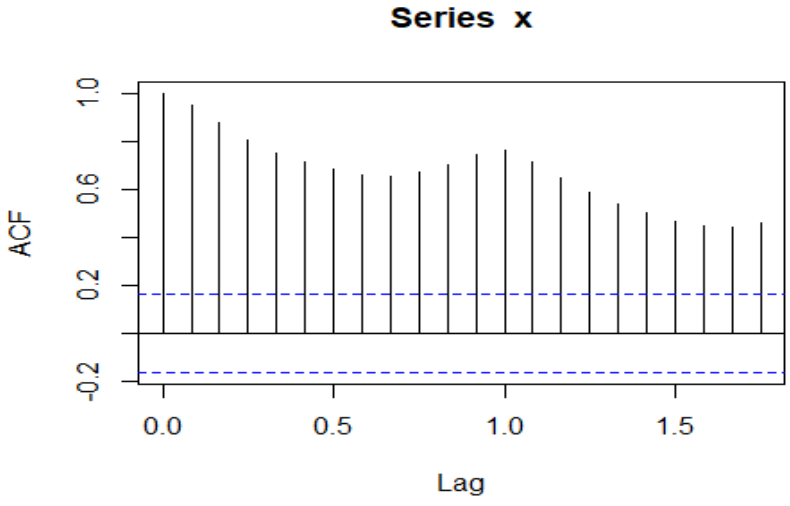


Méthode ARIMA

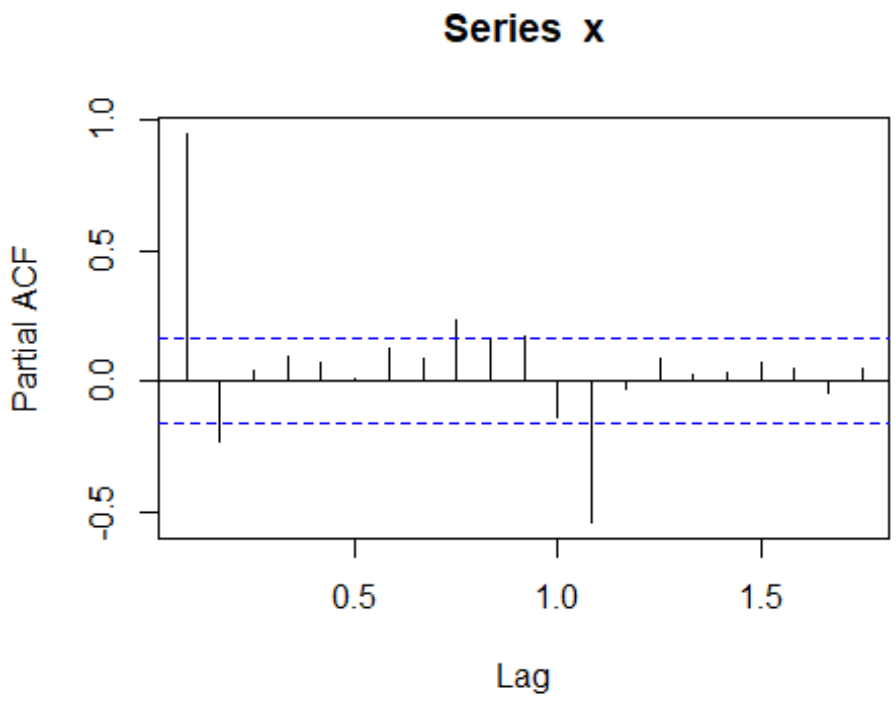
```
# library required for forecasting
library(forecast)

data(AirPassengers)
plot(AirPassengers)
```



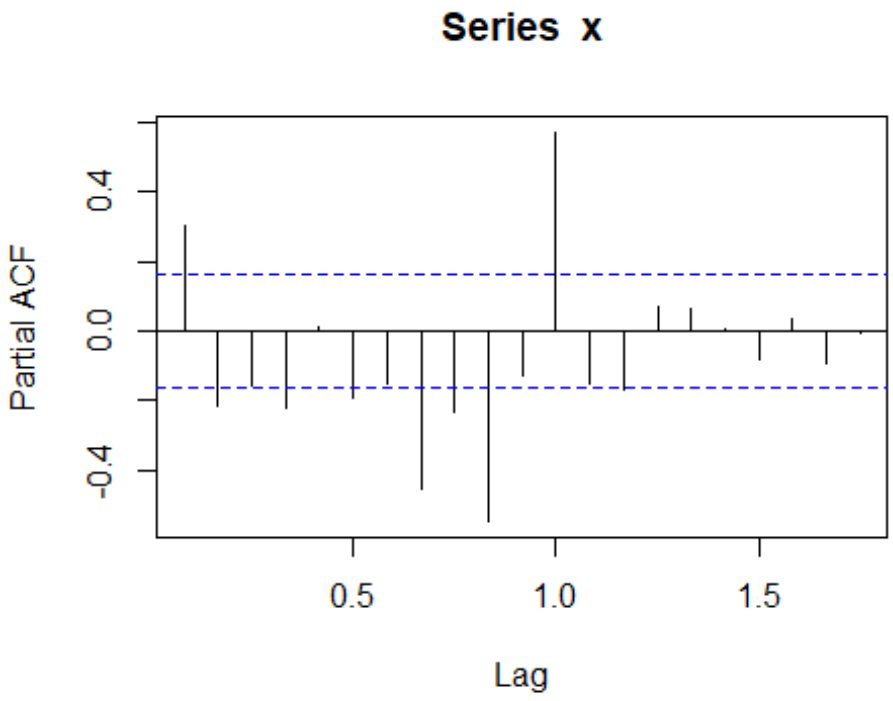

```
x<-AirPassengers
## acf and pacf
( x.acf <- acf(x) )
```



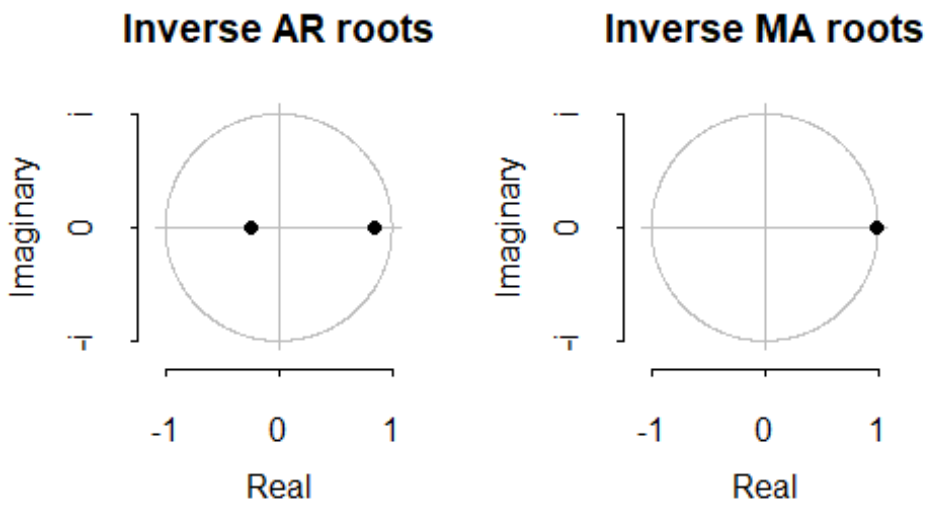
```
##
## Autocorrelations of series ‘x’, by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667
0.7500 0.83
33
## 1.000 0.948 0.876 0.807 0.753 0.714 0.682 0.663 0.656 0.671 0.7
03
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833
1.6667 1.75
00
## 0.743 0.760 0.713 0.646 0.586 0.538 0.500 0.469 0.450 0.442 0.4
57

( x.pacf <- pacf(x) )
```



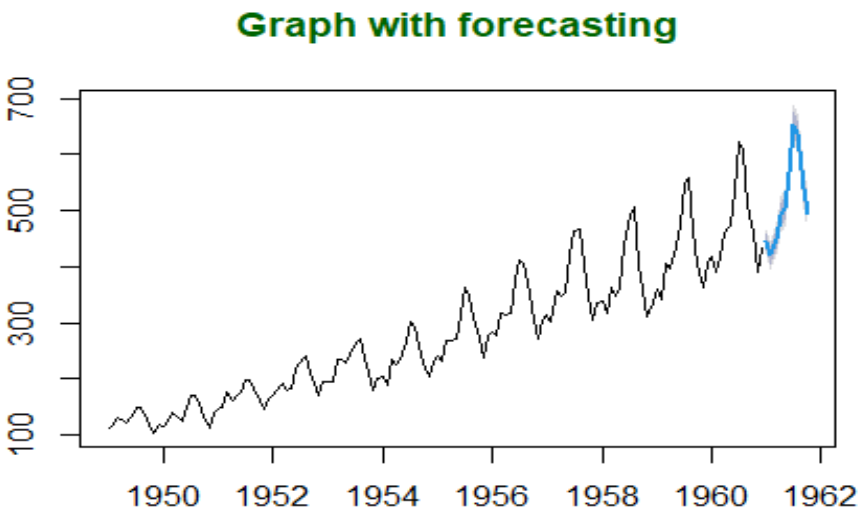
```
##
## Autocorrelations of series ‘x’, by lag
##
## 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667
0.7500 0.83
33
## 1.000 0.303 -0.102 -0.241 -0.300 -0.094 -0.078 -0.092 -0.295 -0.192
-0.1
05
## 0.9167 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833
1.6667 1.75
00
## 0.283 0.829 0.285 -0.106 -0.222 -0.231 -0.062 -0.066 -0.090 -0.297
-0.1
63

( x.pacf <- pacf(x) )
```



```
##
## Partial autocorrelations of series ‘x’, by lag
##
## 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667 0.7500
0.8333 0.91
67
## 0.303 -0.213 -0.160 -0.222 0.010 -0.191 -0.154 -0.455 -0.234 -0.547
-0.1
30
## 1.0000 1.0833 1.1667 1.2500 1.3333 1.4167 1.5000 1.5833 1.6667
1.7500
## 0.571 -0.149 -0.172 0.067 0.062 0.008 -0.080 0.037 -0.095 -0.005

#Arima et prédiction
ap.arima <- auto.arima(AirPassengers)
plot(ap.arima)
```

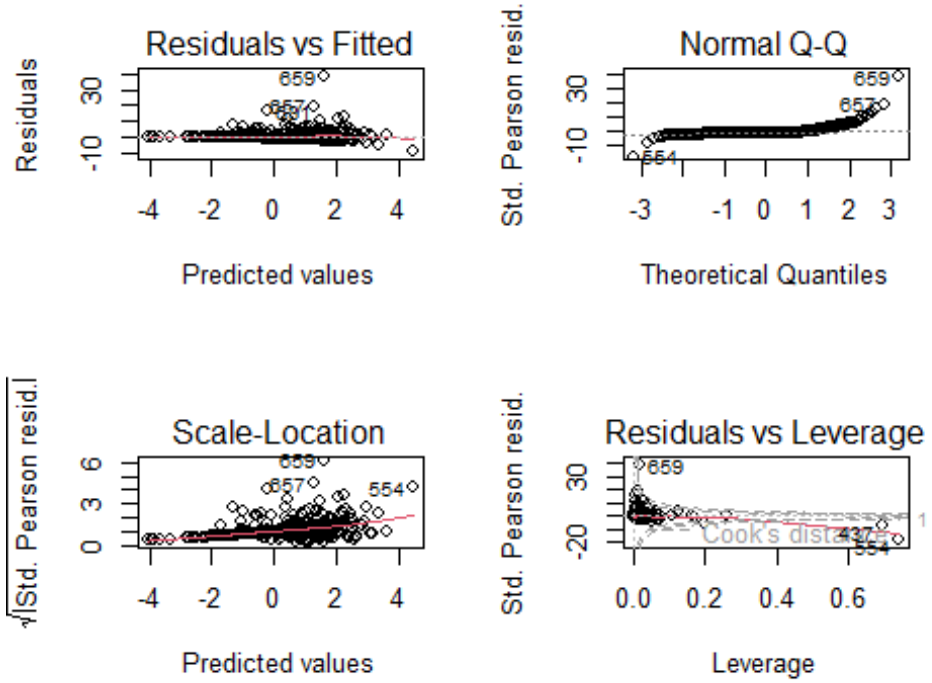


```
forecastedValues <- forecast(ap.arima, 10)

plot(forecastedValues, main = «Graph with forecasting»,
col.main = «darkgreen»)
```



Modélisation linéaire généralisée

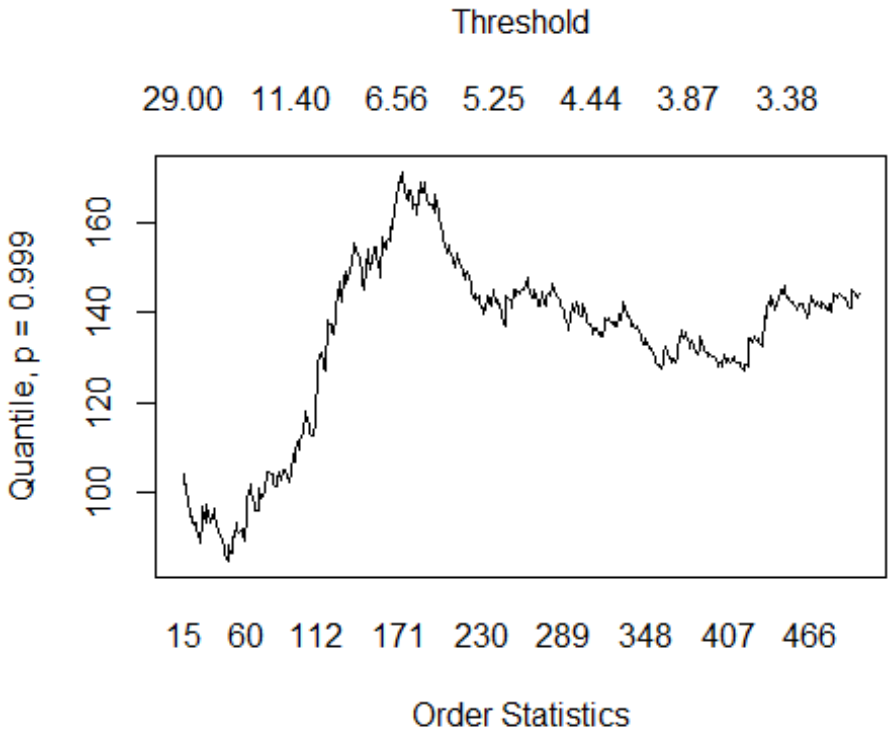
```
library(AER)

data(RecreationDemand)
Reg_pois <- glm(trips ~ ., data = RecreationDemand, family = poisson)

summary(Reg_pois)

##
## Call:
## glm(formula = trips ~ ., family = poisson, data = RecreationDemand)
##
## Deviance Residuals:
## Min 1Q Median 3Q Max
## -11.8465 -1.1411 -0.8896 -0.4780 18.6071
##
## Coefficients:
## Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.264993 0.093722 2.827 0.00469 **
## quality 0.471726 0.017091 27.602 < 2e-16 ***
## skiyes 0.418214 0.057190 7.313 2.62e-13 ***
## income -0.111323 0.019588 -5.683 1.32e-08 ***
## userfeeyes 0.898165 0.078985 11.371 < 2e-16 ***
## costC -0.003430 0.003118 -1.100 0.27131
## costS -0.042536 0.001670 -25.467 < 2e-16 ***
## costH 0.036134 0.002710 13.335 < 2e-16 ***
## ---
## Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ‘ 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
## Null deviance: 4849.7 on 658 degrees of freedom
## Residual deviance: 2305.8 on 651 degrees of freedom
## AIC: 3074.9
##
## Number of Fisher Scoring iterations: 7

par(mfrow=c(2,2))
plot(Reg_pois)
```


Residuals vs Fitted
Residuals
Predicted values
659
657
554
Normal Q-Q
Std. Pearson resid.
Theoretical Quantiles
Scale-Location
√|Std. Pearson resid.|
Residuals vs Leverage
Cook's distance
Leverage

Exemple d'estimateur en TVE

```
### Hill

#install.packages(«evir»)
library(evir)

data(danish)

hill(danish) #Fire insurance Data
```

```
hill(danish, option = «quantile», end = 500, p = 0.999)
```



SeaBird

www.seabirdconseil.com

Suivez-nous : linkedin.com/company/seabird-consulting | @SeaBirdC

Partenaire conseil
des métiers de l'Assurance

15-17 rue de Bucarest
75008 Paris