[422]:

```
## Importing packages

# This R environment comes with all of CRAN and many other helpful packag
# You can see which packages are installed by checking out the kaggle/rst
# https://github.com/kaggle/docker-rstats

library(tidyverse) # metapackage with lots of helpful functions
library("scatterplot3d") # load 3d scatter plot
require(boot)

## Running code

# In a notebook, you can run a single code cell by clicking in the cell a
# the blue arrow to the left, or by clicking in the cell and pressing Shi
# you can run code by highlighting the code you want to run and then clic
# at the bottom of this window.

## Reading in files

# You can access files from datasets you've added to this kernel in the "
# You can see the files added to this kernel by running the code below.

list.files(path = "../input")

## Saving data

# If you save any files or images, these will be put in the "output" dire
# can see the output directory by committing and running your kernel (usi
# Commit & Run button) and then checking out the compiled version of your
```

```
'5.R.RData'
```

[423]:

```
getwd()
```

```
'/kaggle/working'
```

[424]:

```r
list.files(path = "../input")
```

'5.R.RData'

[425]:

```r
data = load("../input/5.R.RData")
```

[426]:

```r
# list the contents of a data
data
```

'Xy'

[427]:

```r
# what is Xy ?
class(Xy)
# Xy is a data.frame
```

'data.frame'

[428]:

```r
# query the dimension of a matrix.
dim(Xy)
# 1000 x 3 data frame
```

```
1000   3
```

[429]:

```
Xy[1:5,]
```

X1	X2	y
1.297720	0.8059212	0.2989683
1.267323	0.7990341	0.3181337
1.236882	0.7921693	0.3372015
1.206317	0.7852963	0.3561210
1.175553	0.7783848	0.3748415

[430]:

```
Xy$c4 = y=seq(from=1,length=1000,by=1)
```

[431]:

```
Xy[1:5,]
```

X1	X2	y	c4
1.297720	0.8059212	0.2989683	1
1.267323	0.7990341	0.3181337	2
1.236882	0.7921693	0.3372015	3
1.206317	0.7852963	0.3561210	4
1.175553	0.7783848	0.3748415	5

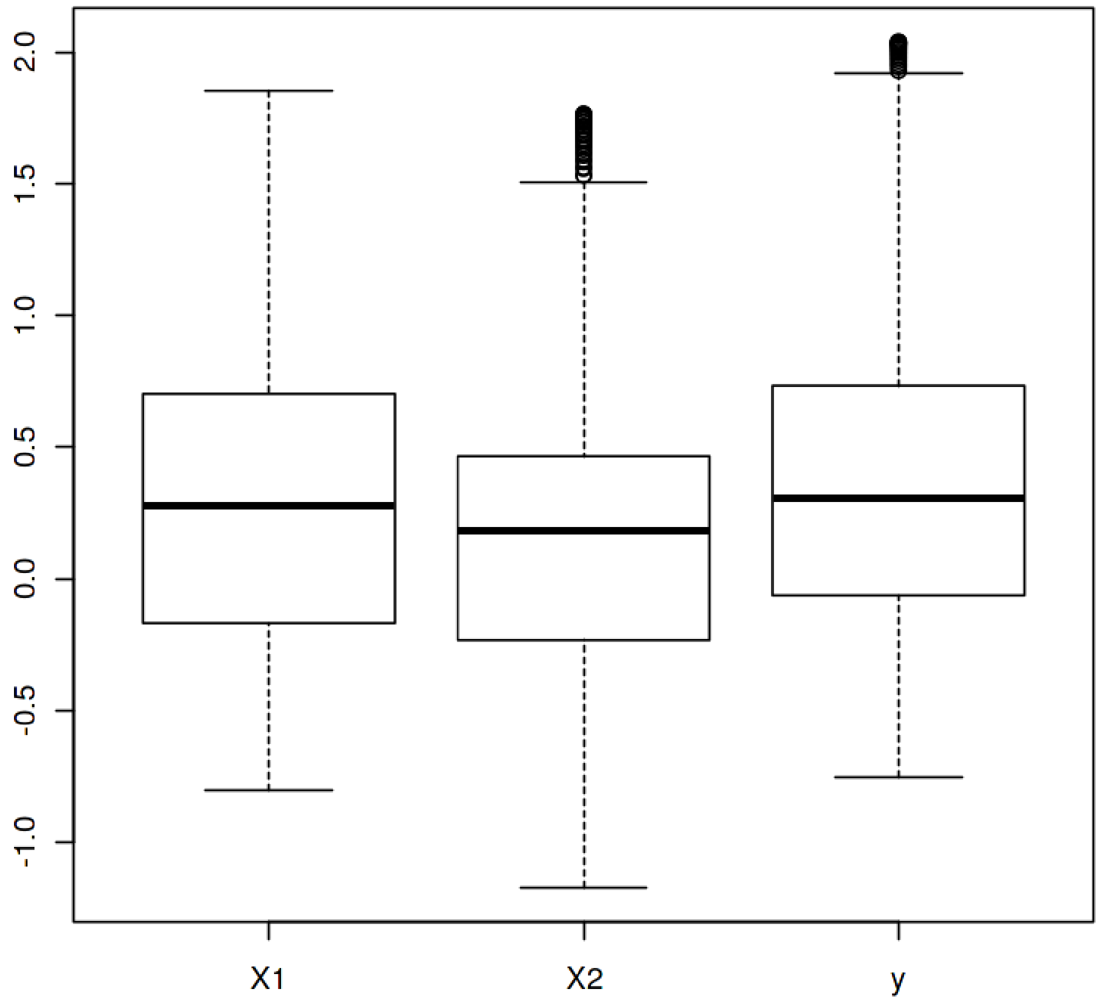
[432]:

```
summary(Xy)
```

```
       X1                 X2                 y                   c4
 Min.   :-0.8068   Min.   :-1.1753   Min.   :-0.75293   Min.   :   1.
0
 1st Qu.:-0.1674   1st Qu.:-0.2339   1st Qu.:-0.06136   1st Qu.: 250.
8
 Median : 0.2798   Median : 0.1824   Median : 0.30452   Median : 500.
5
 Mean   : 0.3337   Mean   : 0.1288   Mean   : 0.35471   Mean   : 500.
5
 3rd Qu.: 0.7017   3rd Qu.: 0.4646   3rd Qu.: 0.73283   3rd Qu.: 750.
2
 Max.   : 1.8531   Max.   : 1.7658   Max.   : 2.03922   Max.   :1000.
0
```

[433]:

```r
boxplot(Xy[,1:3])
```



[437]:

```r
# It tells you what you have available in your working directory.
#ls()
#rm(y)
#rm(beta1_stder,boot.out,data,fit_block_bt,fit1,new.rows,new.Xy,s,se_beta
```

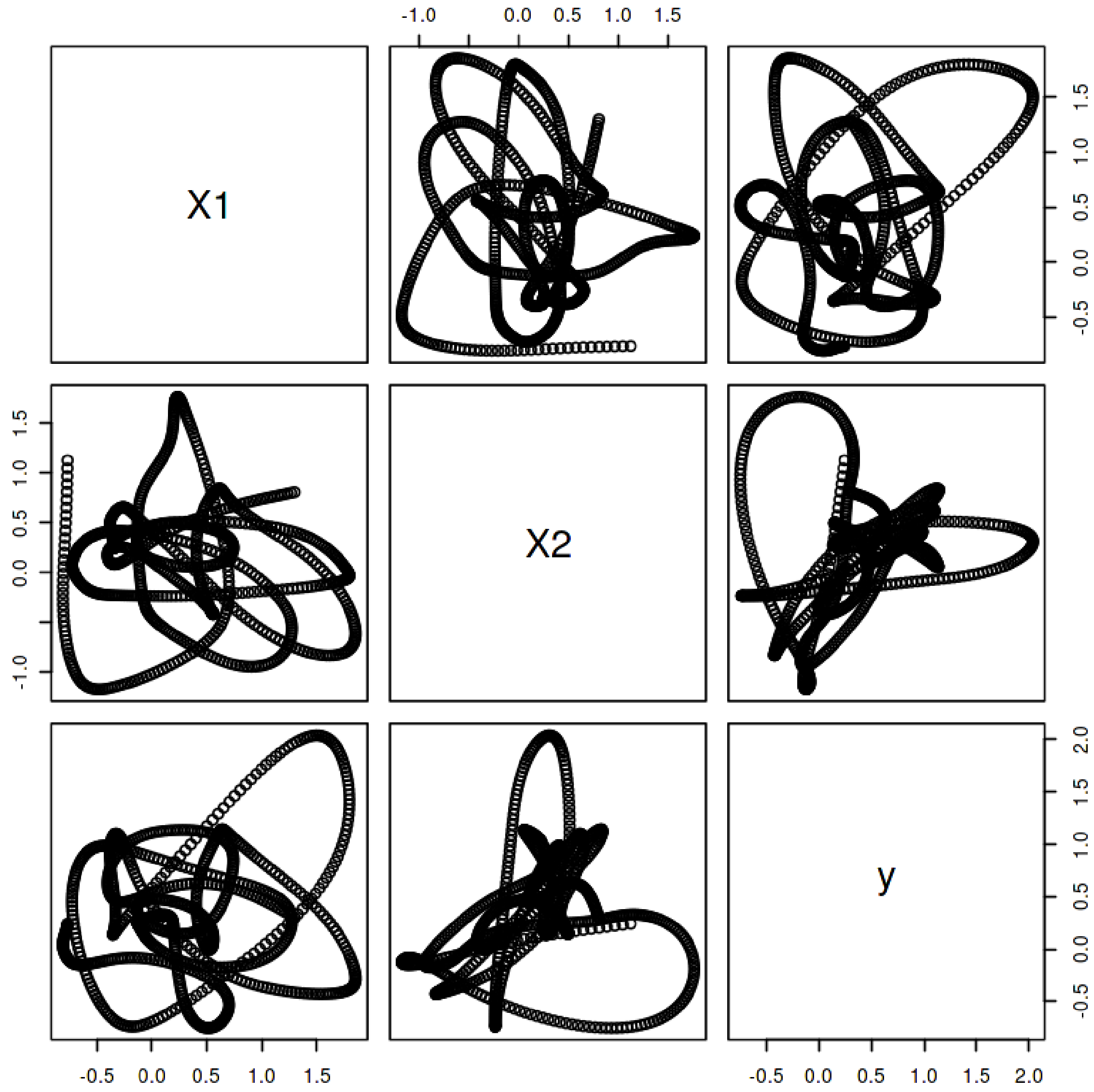
[440]:

```
# attach the dataframe.
attach(Xy)
```

```
The following object is masked _by_ .GlobalEnv:

    y

The following objects are masked from Xy (pos = 3):

    c4, X1, X2, y

The following objects are masked from Xy (pos = 4):

    c4, X1, X2, y

The following objects are masked from Xy (pos = 6):

    c4, X1, X2, y

The following objects are masked from Xy (pos = 7):

    c4, X1, X2, y

The following objects are masked from Xy (pos = 8):

    c4, X1, X2, y

The following objects are masked from Xy (pos = 9):

    c4, X1, X2, y

The following objects are masked from Xy (pos = 10):

    c4, X1, X2, y

The following objects are masked from Xy (pos = 11):

    c4, X1, X2, y

The following objects are masked from Xy (pos = 13):
    X1, X2, y

The following objects are masked from Xy (pos = 14):

    X1, X2, y

The following objects are masked from Xy (pos = 15):

    X1, X2, y

The following objects are masked from Xy (pos = 16):

    X1, X2, y
```

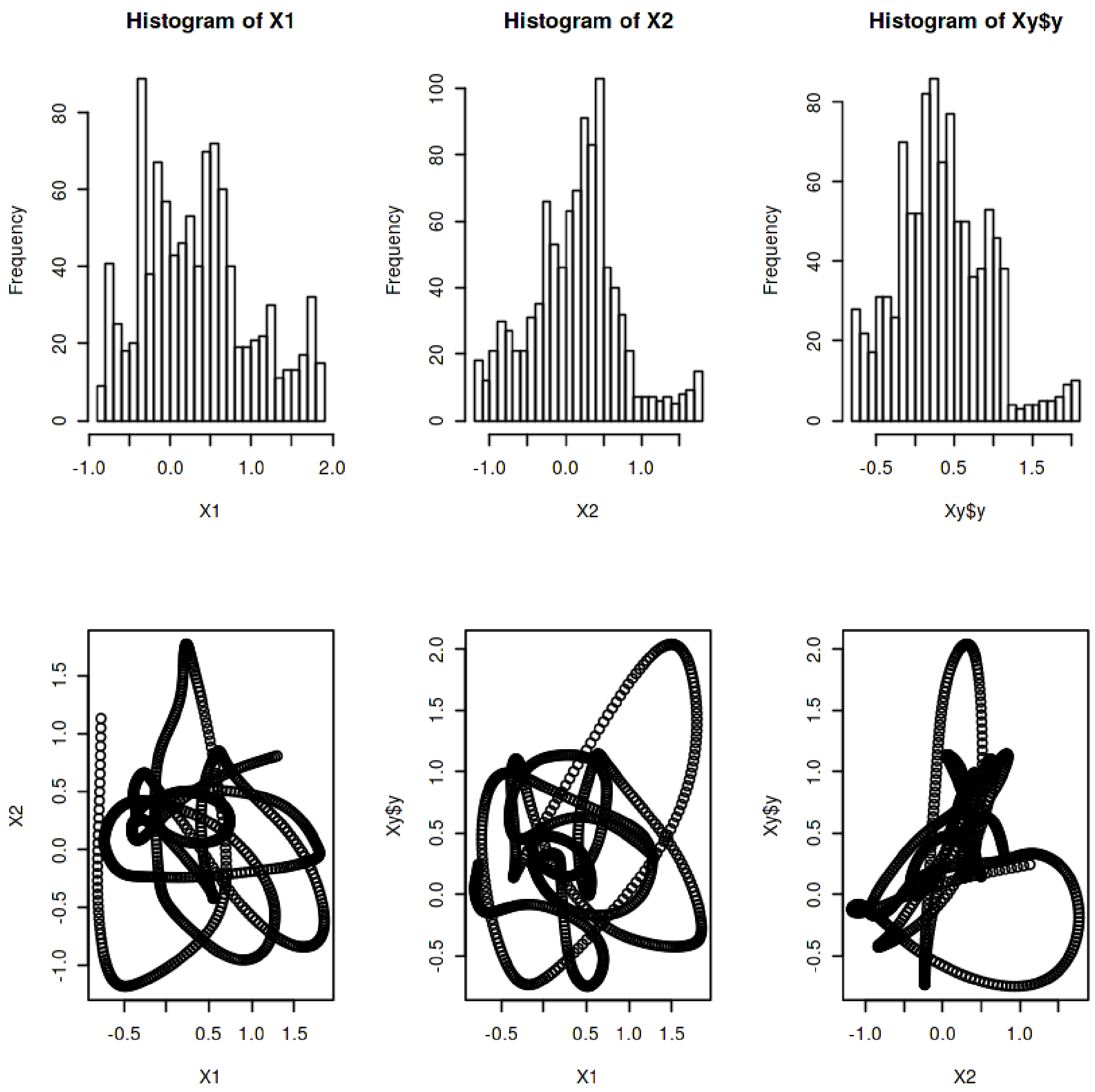
[438]:

```
pairs(Xy[,1:3])
```



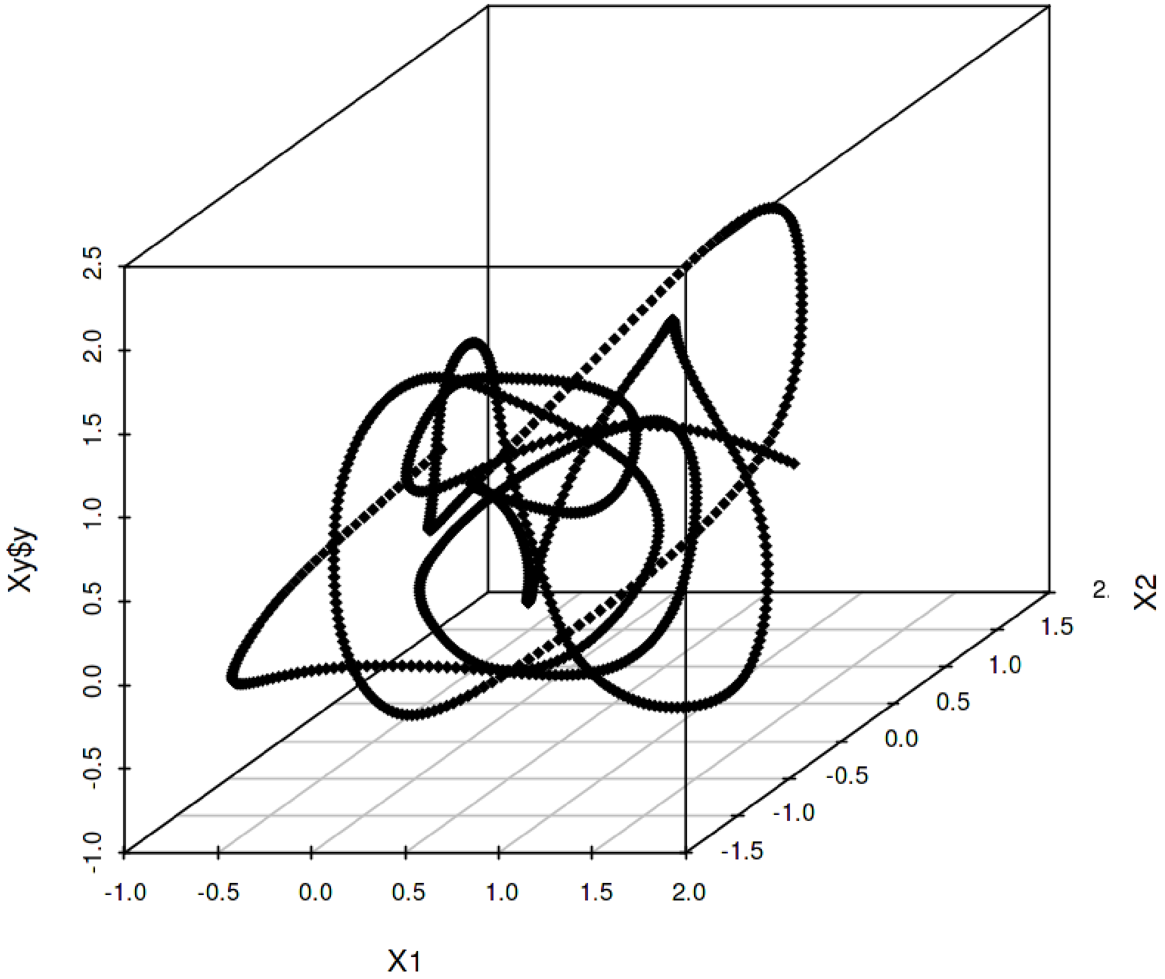
[449]:

```
par(mfrow=c(2,3))
hist(X1,breaks=20)
hist(X2,breaks=40)
hist(Xy$y,breaks=20)
plot(X1,X2)
plot(X1,Xy$y)
plot(X2,Xy$y)
```



[451]:

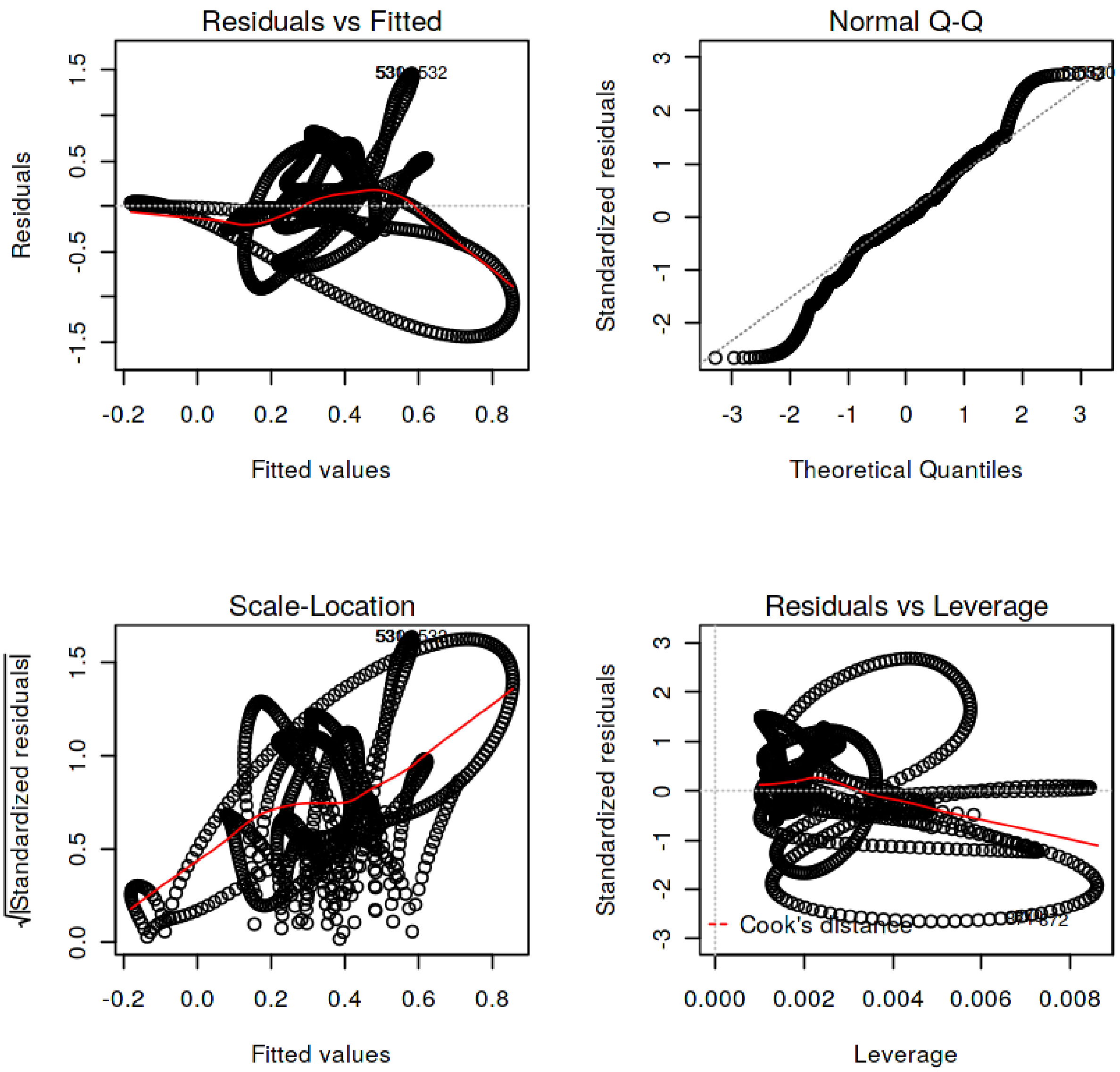
```
par(mfrow=c(1,1))
library("plot3D")

scatterplot3d(X1,X2,Xy$y,pch = 18)

#?scatterplot3d
```



[454]:

```
fit1=lm(y~X1+X2,data=Xy)
fit1
par(mfrow=c(2,2))
plot(fit1)
```

```
Call:
lm(formula = y ~ X1 + X2, data = Xy)

Coefficients:
(Intercept)           X1           X2  
     0.2658       0.1453       0.3134  
```



[455]:

```
summary(fit1)
s=summary(fit1)
```

```
Call:
lm(formula = y ~ X1 + X2, data = Xy)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.44171 -0.25468 -0.01736  0.33081  1.45860 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.26583    0.01988  13.372  < 2e-16 ***
X1           0.14533    0.02593   5.604 2.71e-08 ***
X2           0.31337    0.02923  10.722  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5451 on 997 degrees of freedom
Multiple R-squared:  0.1171,	Adjusted R-squared:  0.1154 
F-statistic: 66.14 on 2 and 997 DF,  p-value: < 2.2e-16
```

[456]:

```
# str(): This function provides a summary of the objects attributes,
str(s)

# show coefficients
s$coefficients

beta1_stder = s$coefficients[2,2]
beta1_stder
#?lm
```

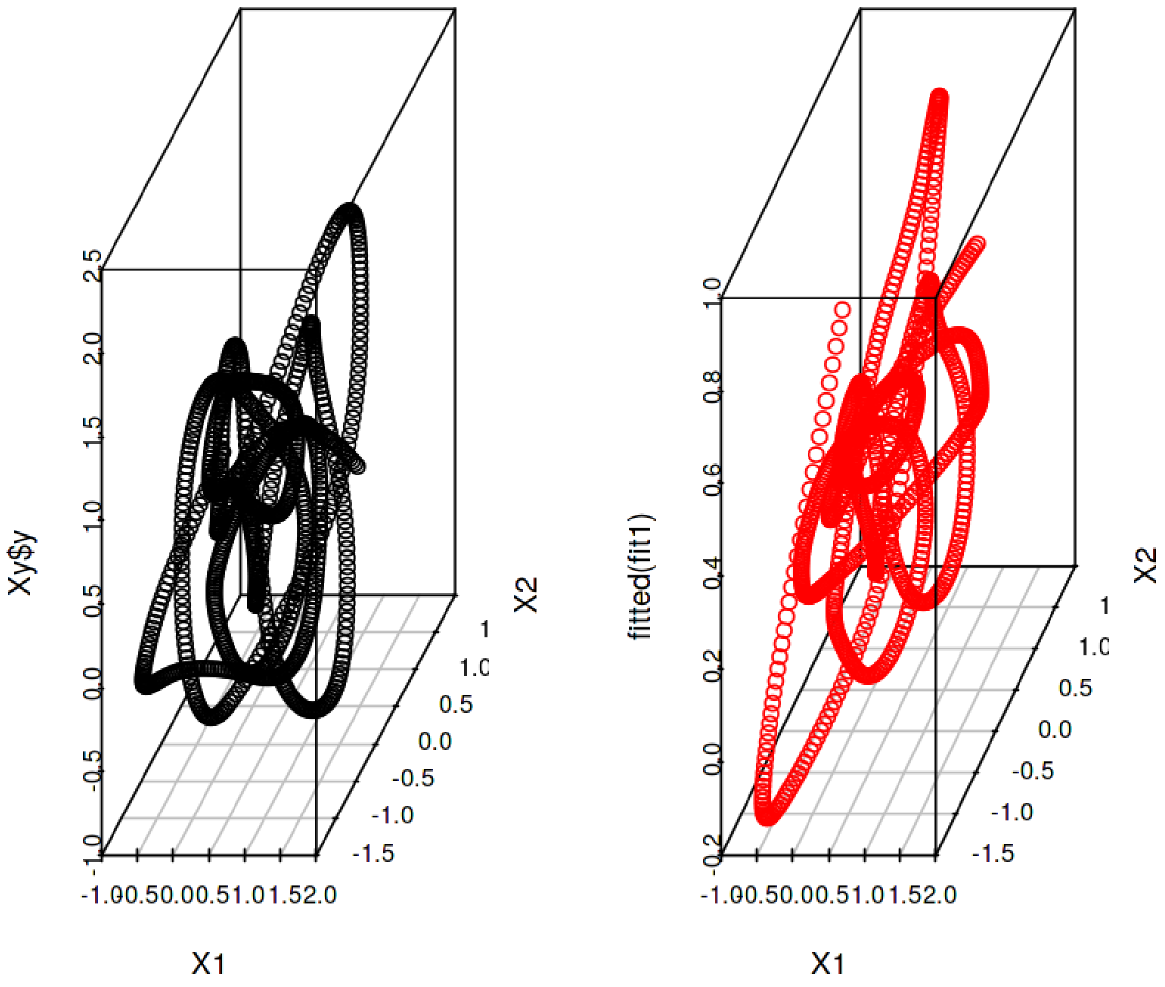
```
List of 11
 $ call         : language lm(formula = y ~ X1 + X2, data = Xy)
 $ terms        :Classes 'terms', 'formula'  language y ~ X1 + X2
  .. ..- attr(*, "variables")= language list(y, X1, X2)
  .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:3] "y" "X1" "X2"
  .. .. .. ..$ : chr [1:2] "X1" "X2"
  .. ..- attr(*, "term.labels")= chr [1:2] "X1" "X2"
  .. ..- attr(*, "order")= int [1:2] 1 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. ..- attr(*, "predvars")= language list(y, X1, X2)
  .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric"
"numeric"
  .. .. ..- attr(*, "names")= chr [1:3] "y" "X1" "X2"
 $ residuals    : Named num [1:1000] -0.408 -0.382 -0.357 -0.331 -0.3
06 ...
  ..- attr(*, "names")= chr [1:1000] "1" "2" "3" "4" ...
 $ coefficients : num [1:3, 1:4] 0.2658 0.1453 0.3134 0.0199 0.0259
...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "(Intercept)" "X1" "X2"
  .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
 $ aliased      : Named logi [1:3] FALSE FALSE FALSE
  ..- attr(*, "names")= chr [1:3] "(Intercept)" "X1" "X2"
 $ sigma        : num 0.545
 $ df           : int [1:3] 3 997 3
 $ r.squared    : num 0.117
 $ adj.r.squared: num 0.115
 $ fstatistic   : Named num [1:3] 66.1 2 997
  ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
 $ cov.unscaled : num [1:3, 1:3] 0.00133 -0.000801 -0.000488 -0.00080
1 0.002263 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "(Intercept)" "X1" "X2"
  .. ..$ : chr [1:3] "(Intercept)" "X1" "X2"
 - attr(*, "class")= chr "summary.lm"
```

	Estimate	Std. Error	t value	Pr(>ltl)
(Intercept)	0.2658349	0.01988032	13.371758	1.249278e-37
X1	0.1453263	0.02593295	5.603925	2.711026e-08
X2	0.3133670	0.02922671	10.721938	1.843565e-25

0.0259329527526028

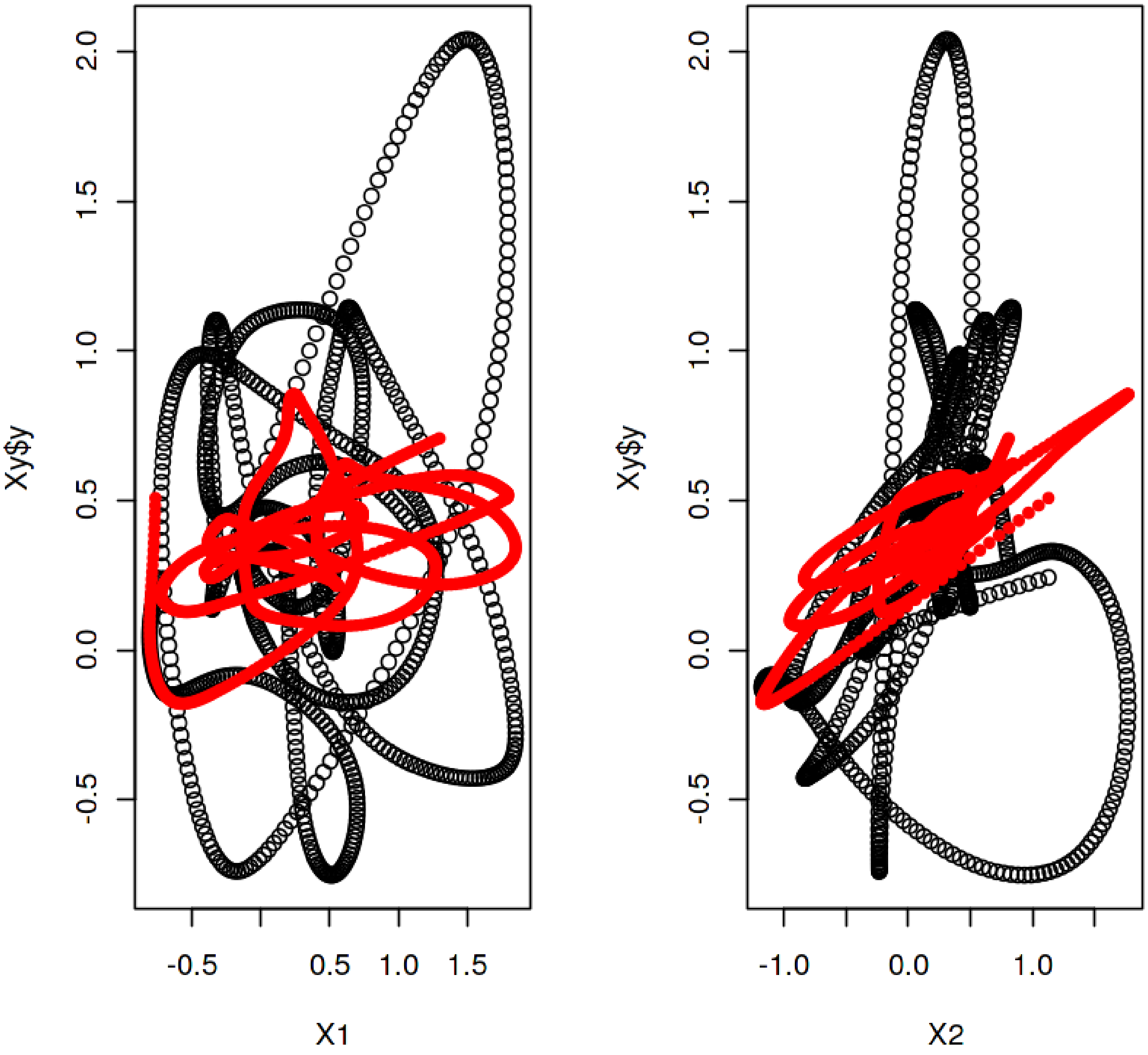
[458]:

```
# add linear model line to the plot
# need to plot before abline
# plot(y~X1,data=Xy)
#points(lstat,fitted(fit6),col="red",pch=20)
par(mfrow=c(1,2))
scatterplot3d(X1,X2,Xy$y)
scatterplot3d(X1,X2,fitted(fit1),color="red")

#?scatterplot3d
```



[461]:

```
par(mfrow=c(1,2))
#plot(X1,X2)

plot(X1,Xy$y)
points(X1,fitted(fit1),col="red",pch=20)

plot(X2,Xy$y)
points(X2,fitted(fit1),col="red",pch=20)
```



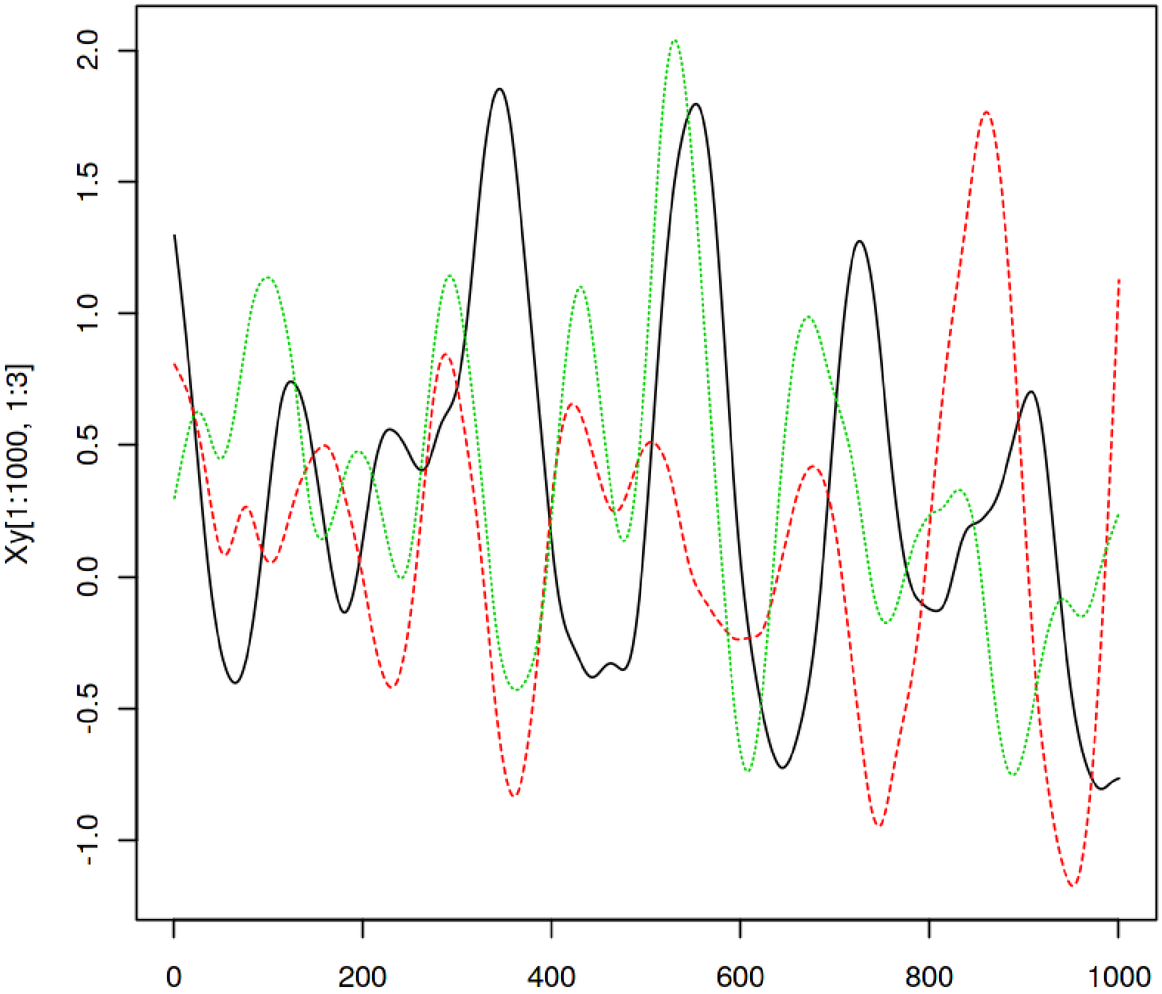
[462]:

```
# find the confident interval for the fit
confint(fit1)
```

	2.5 %	97.5 %
(Intercept)	0.22682280	0.3048470
X1	0.09443689	0.1962158
X2	0.25601406	0.3707199

[463]:

```
matplot(Xy[1:1000,1:3],type="l")
# X1 = black, X2 = red, y = green
#?matplot
```



[464]:

```
# a function to get standard error of beta_1
se_beta1=function(X1,X2,y){
  fit_tmp=lm(y~X1+X2)
  s_tmp=summary(fit_tmp)
  s_tmp$coefficients[2,2]
}
```

[465]:

```
# check the function
se_beta1(X1,X2,Xy$y)
```

```
0.0259329527526028
```

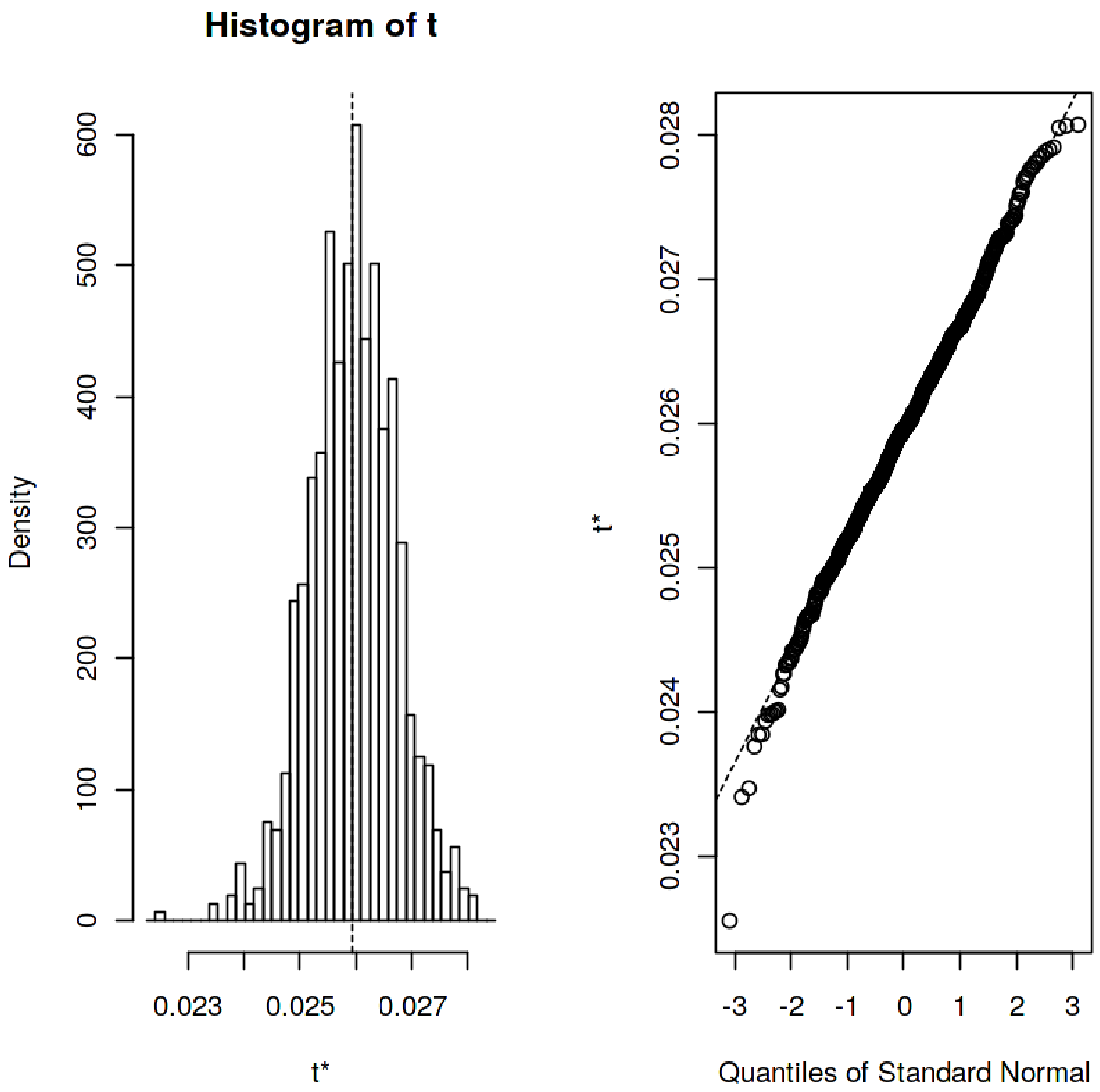
[466]:

```
se_beta1.fn=function(data, index){
  with(data[index,],se_beta1(X1,X2,y))
}
```

[467]:

```
set.seed(1)
#se_beta1.fn(Xy,sample(1:1000,1000,replace=TRUE))
boot.out=boot(Xy,se_beta1.fn,R=1000)
boot.out
plot(boot.out)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Xy, statistic = se_beta1.fn, R = 1000)


Bootstrap Statistics :
      original        bias     std. error
t1* 0.02593295 1.339069e-05 0.0007659365
```


Histogram of t
Density
0
100
200
300
400
500
600
0.023
0.025
0.027
t*
t*
0.023
0.024
0.025
0.026
0.027
0.028
-3
-2
-1
0
1
2
3
Quantiles of Standard Normal

[468]:

```
# test
new.rows=c(101:200, 401:500, 101:200, 901:1000, 301:400, 1:100, 1:100,
new.rows
```

```
101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117
118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133
134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149
150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165
166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181
182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197
198 199 200 401 402 403 404 405 406 407 408 409 410 411 412 413
414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429
430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445
446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461
462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477
478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493
494 495 496 497 498 499 500 101 102 103 104 105 106 107 108 109
110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142
143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158
159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174
175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190
191 192 193 194 195 196 197 198 199 200 901 902 903 904 905 906
907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922
923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938
939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954
955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970
971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986
987 988 989 990 991 992 993 994 995 996 997 998 999 1000 301 302
303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318
319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334
335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350
351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366
367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382
383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398
399 400 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 1 2 3 4 5 6
7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

```
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 801 802 803 804 805 806 807 808 809
810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825
826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841
842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857
858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873
874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889
890 891 892 893 894 895 896 897 898 899 900 201 202 203 204 205
206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221
222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237
238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253
254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285
286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 701
702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717
718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733
734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749
750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765
766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781
782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797
798 799 800
```

[469]:

```
# test
new.Xy = Xy[new.rows, ]
new.Xy
```

	X1	X2	y	c4
101	0.3270281	0.05491268	1.1357421	101
102	0.3586748	0.05375249	1.1343526	102
103	0.3897927	0.05369906	1.1320618	103
104	0.4202234	0.05471397	1.1287803	104
105	0.4498082	0.05675881	1.1244186	105
106	0.4784009	0.05979892	1.1189060	106
107	0.5059039	0.06381470	1.1122458	107
108	0.5322319	0.06879031	1.1044598	108
109	0.5572996	0.07470990	1.0955699	109
110	0.5810215	0.08155763	1.0855978	110
111	0.6033124	0.08931766	1.0745655	111
112	0.6240869	0.09797415	1.0624948	112
113	0.6432683	0.10749860	1.0494115	113
114	0.6608141	0.11781199	1.0353574	114
115	0.6766903	0.12882263	1.0203784	115
116	0.6908629	0.14043885	1.0045204	116
117	0.7032981	0.15256897	0.9878291	117
118	0.7139618	0.16512131	0.9703504	118
119	0.7228203	0.17800418	0.9521302	119
120	0.7298649	0.19112683	0.9331908	120
121	0.7351893	0.20440214	0.9134611	121
122	0.7389127	0.21774390	0.8928465	122
123	0.7411540	0.23106590	0.8712524	123
124	0.7420323	0.24428193	0.8485840	124
125	0.7416668	0.25730580	0.8247469	125
126	0.7401765	0.27005128	0.7996464	126
127	0.7376614	0.28244445	0.7732268	127
128	0.7341449	0.29446045	0.7455887	128
129	0.7296314	0.30608667	0.7168715	129
130	0.7241252	0.31731052	0.6872147	130
⋮	⋮	⋮	⋮	⋮
771	0.109086912	-0.558309351	-0.031764372	771
772	0.087844241	-0.540804599	-0.017574880	772
773	0.067860715	-0.523412706	-0.003378743	773
774	0.049180562	-0.506083192	0.010721767	774
775	0.031848008	-0.488765575	0.024624375	775

	X1	X2	y	c4
776	0.015907284	-0.471409373	0.038226808	776
777	0.001381508	-0.453947673	0.051439617	777
778	-0.011790625	-0.436247820	0.064224645	778
779	-0.023691531	-0.418160726	0.076556559	779
780	-0.034403625	-0.399537304	0.088410027	780
781	-0.044009321	-0.380228466	0.099759715	781
782	-0.052591034	-0.360085124	0.110580293	782
783	-0.060231179	-0.338958190	0.120846426	783
784	-0.067015690	-0.316717850	0.130545076	784
785	-0.073044582	-0.293311384	0.139712378	785
786	-0.078421389	-0.268705342	0.148396763	786
787	-0.083249646	-0.242866278	0.156646659	787
788	-0.087632887	-0.215760744	0.164510496	788
789	-0.091674647	-0.187355292	0.172036703	789
790	-0.095478459	-0.157616475	0.179273710	790
791	-0.099128543	-0.126535711	0.186256600	791
792	-0.102631848	-0.094203880	0.192967068	792
793	-0.105976008	-0.060736727	0.199373463	793
794	-0.109148659	-0.026249999	0.205444135	794
795	-0.112137433	0.009140559	0.211147433	795
796	-0.114929964	0.045319202	0.216451705	796
797	-0.117513888	0.082170184	0.221325302	797
798	-0.119878667	0.119577131	0.225745684	798
799	-0.122021088	0.157421158	0.229726767	799
800	-0.123939766	0.195582753	0.233291579	800

[471]:

```
se_beta1_v2=function(data_tmp){
  fit_tmp2=lm(data_tmp$y~data_tmp$X1+data_tmp$X2)
  s_tmp2=summary(fit_tmp2)
  s_tmp2$coefficients[2,2]
}
```

[472]:

```
# check the function
se_beta1_v2(Xy[,1:3])
```

```
0.0259329527526028
```

[473]:

```
set.seed(1)
tsboot.out = tsboot(Xy, se_beta1_v2, R = 1000, l = 100, sim = "fixed")
#?tsboot
```

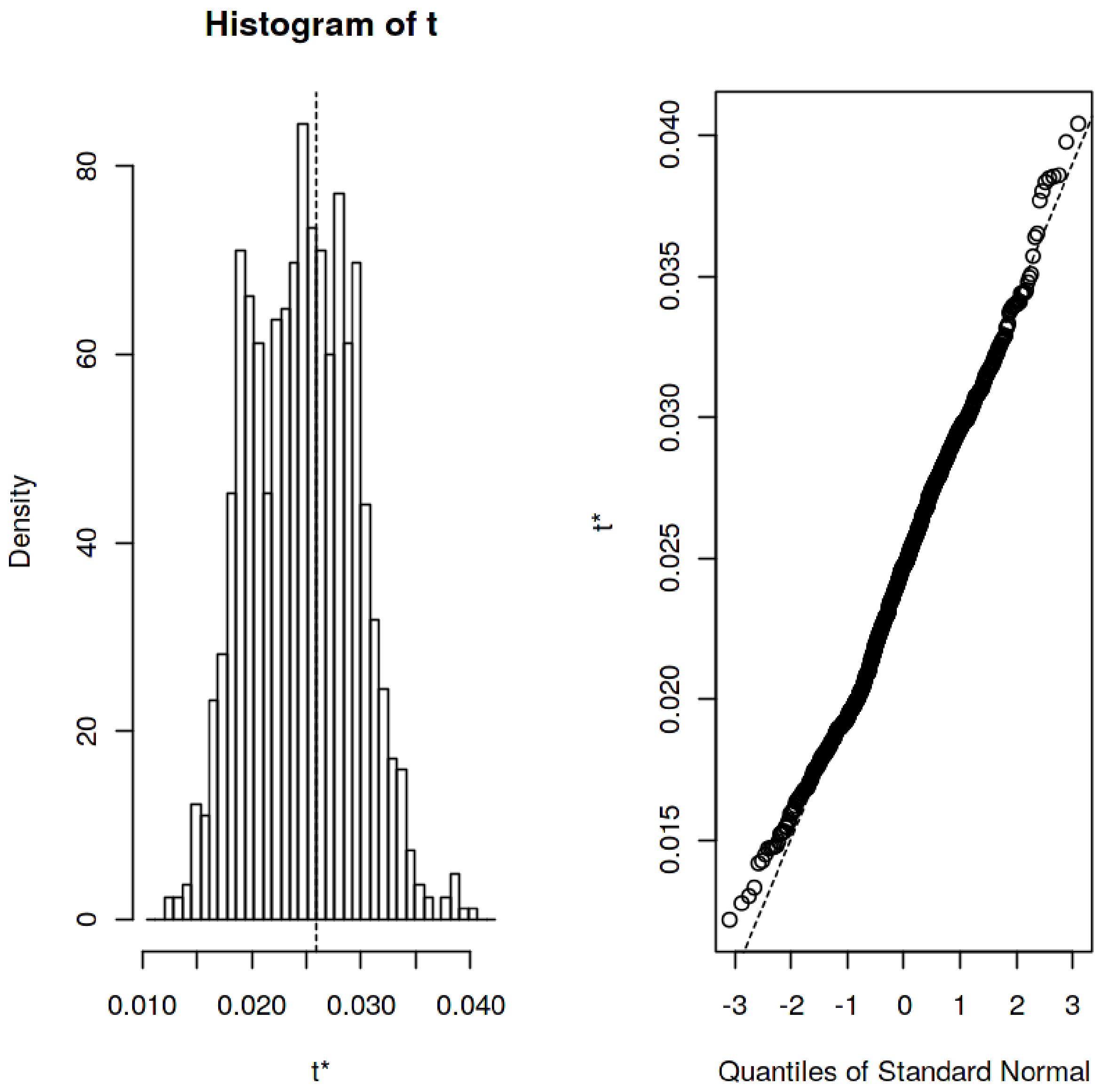
[474]:

```
tsboot.out
# https://stats.stackexchange.com/questions/70593/understanding-the-outpu
```

```
BLOCK BOOTSTRAP FOR TIME SERIES

Fixed Block Length of 100

Call:
tsboot(tseries = Xy, statistic = se_beta1_v2, R = 1000, l = 100,
    sim = "fixed")


Bootstrap Statistics :
      original        bias    std. error
t1* 0.02593295 -0.001283084 0.004794937
```

[475]:

```
#summary(tsboot.out)
plot(tsboot.out)
```



[ ]:

```
```

[ ]:

```
```