# Simulating random processes in R

# Example of RNG

```
n<-10000
x<-integer(n)
x[1] <- 1093
for(i in 2:n){
  x[i]<- ((2^18+1)*x[i-1]) %% 2^35
}
head(x)
```

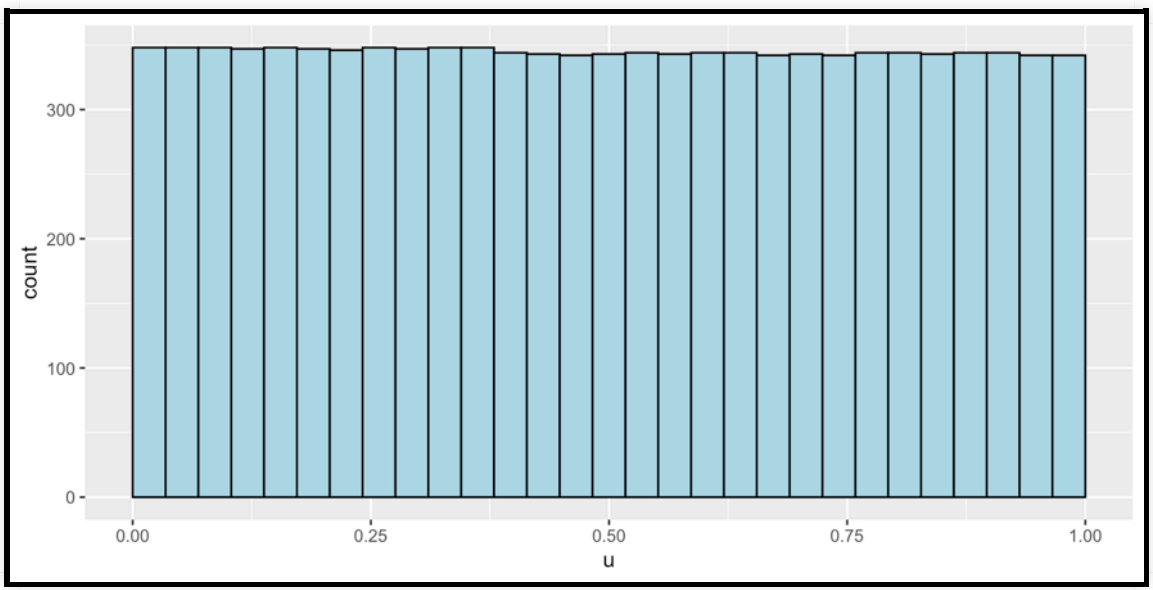
```
[1]       1093  286524485  573047877  859571269 1146094661 1432618053
```

```
head(x/(2^35))
```

```
[1] 3.1810e-08 8.3390e-03 1.6678e-02 2.5017e-02 3.3356e-02 4.1695e-02
```

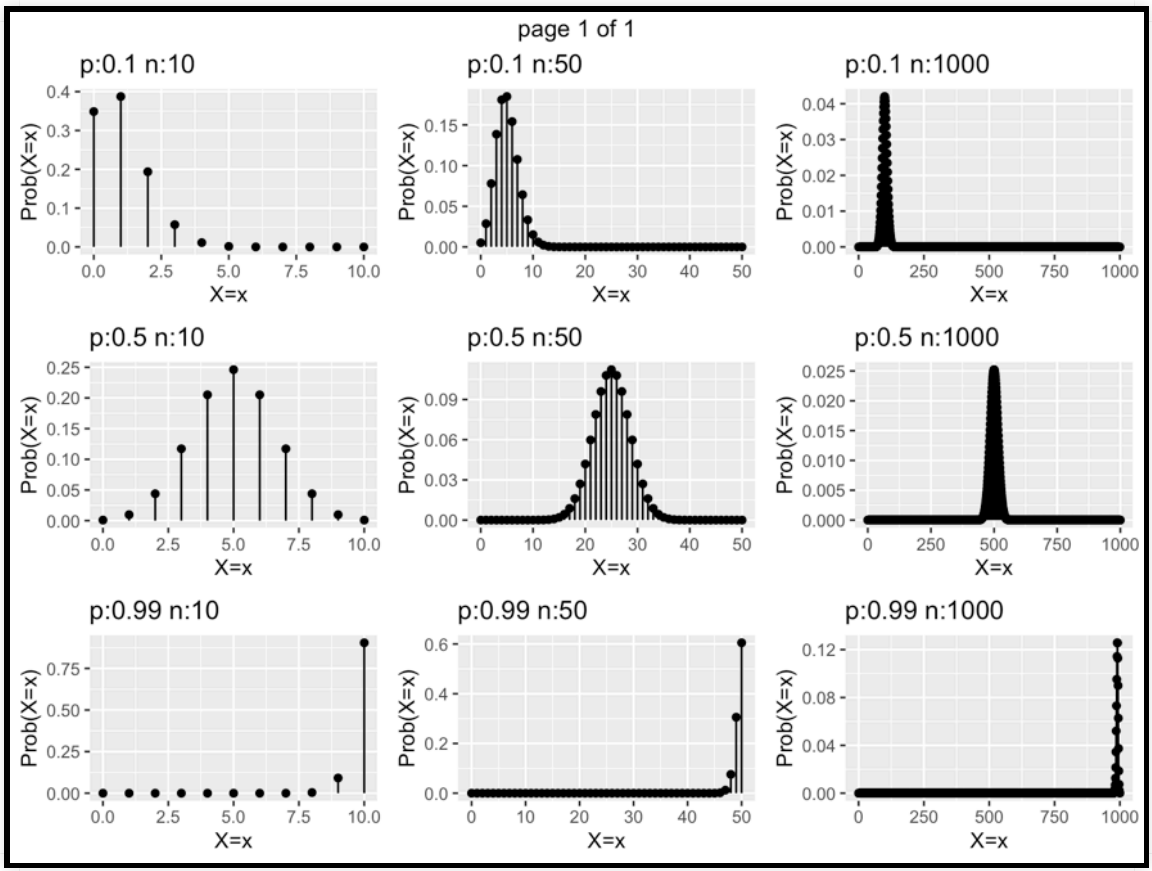
# Example of RNG

```
ggplot(data.frame(u=x/2^35), aes(x=u)) +

        geom_histogram(breaks=seq(0,1,length=30),fill="lightblue",col="
```



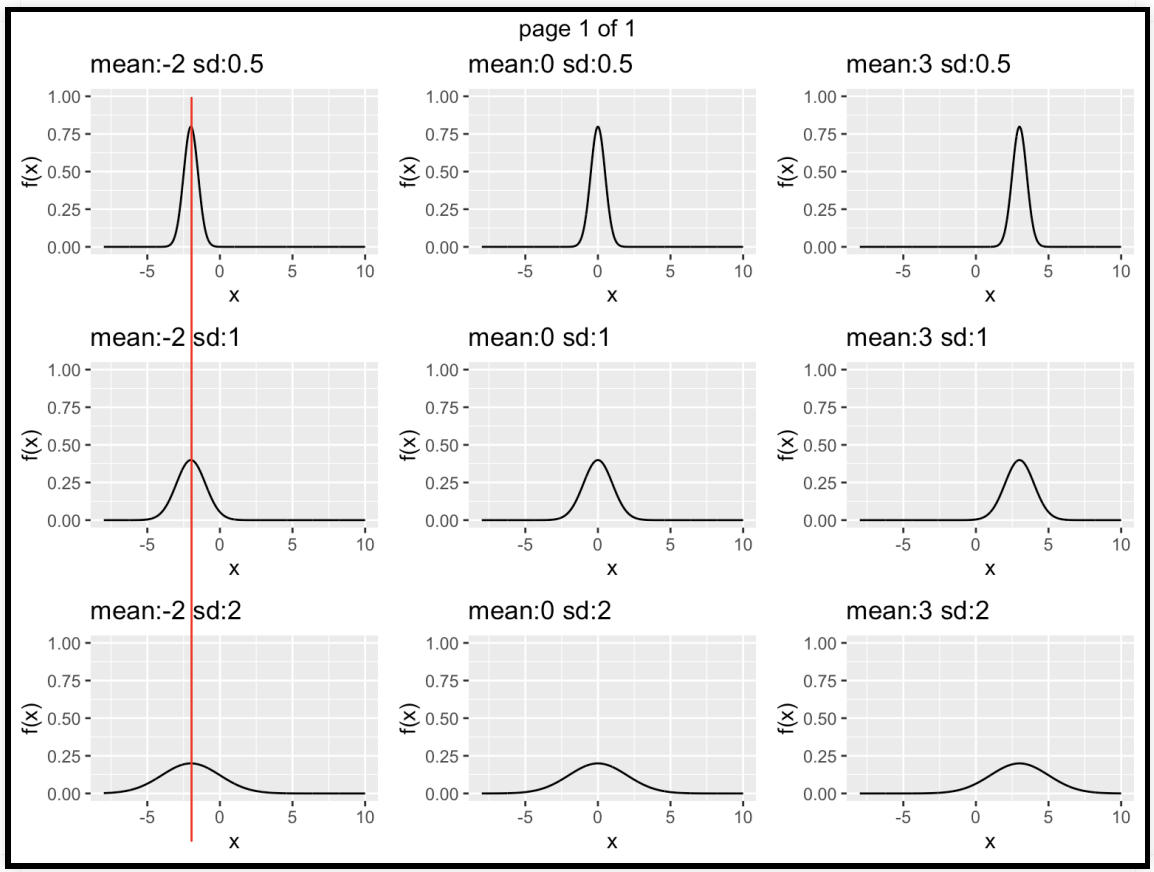
# Binomial distribution with different p and n

parameters.




change of the center will change the spread

# Normal distribution with different $\mu$ and $\sigma$




Could change center and spread separately.

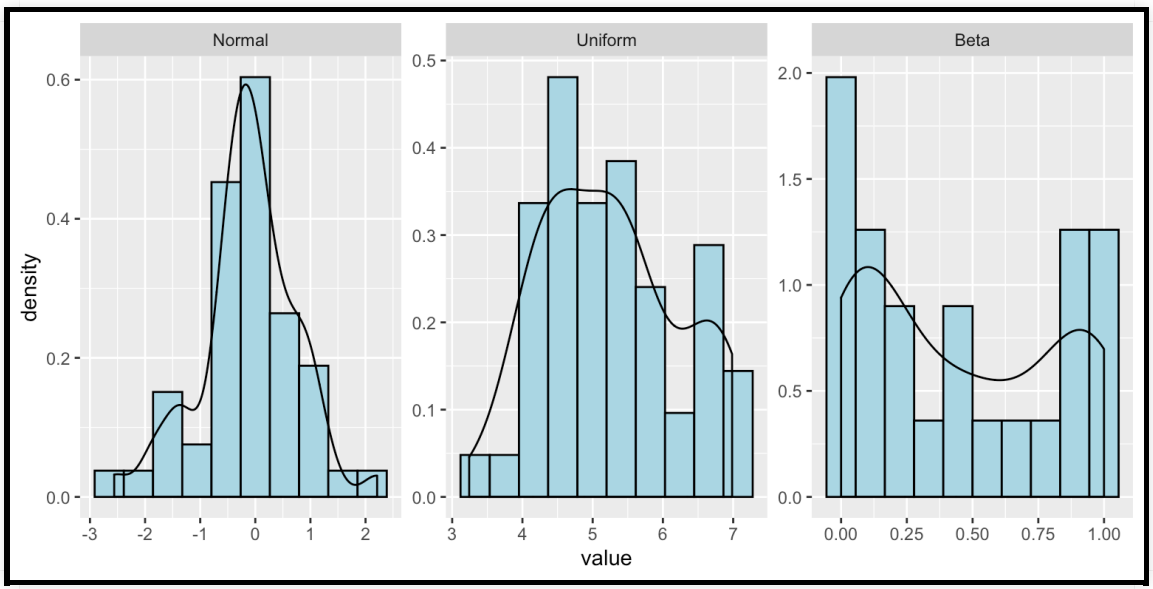
# Generating random variables from a distribution

```
n<-50
my_sims<-tibble(Normal=rnorm(n,mean=0,sd=1),
                Uniform=runif(n,min=3,max=7),
                Beta = rbeta(n,0.5,0.5))
my_sims_long <- my_sims %>% pivot_longer(cols=(Normal:Beta)) %>%
  mutate(name=fct_relevel(name,c("Normal","Uniform","Beta")))
p<-ggplot(my_sims_long, aes(x=value))  +

        geom_histogram(aes(y=..density..),bins=10,fill="lightblue",col="
         +
  geom_density()+
  facet_wrap(~name,scales="free")
```

| name | value |
| --- | --- |
| Normal | ⋮ |
| Uniform | ⋮ |
| Beta | ⋮ |

# N=50

p Not enough.

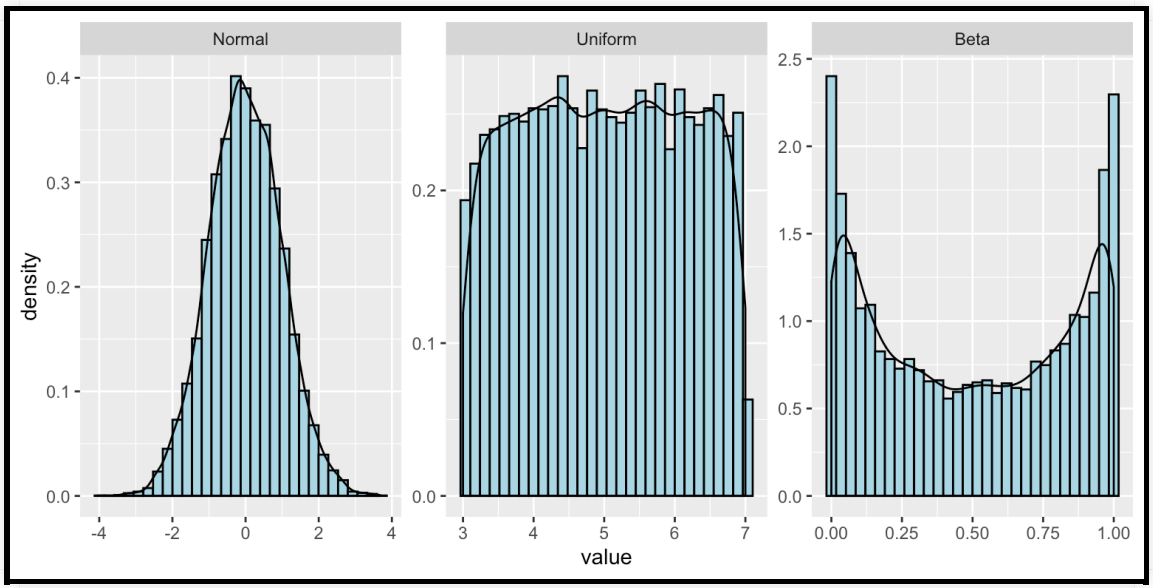


# N=10000

```
n<-10000
my_sims<-tibble(Normal=rnorm(n,mean=0,sd=1),
                Uniform=runif(n,min=3,max=7),
                Beta = rbeta(n,0.5,0.5))

my_sims_long <- my_sims %>% pivot_longer(cols=(Normal:Beta)) %>%
  mutate(name=fct_relevel(name,c("Normal","Uniform","Beta")))

p<-ggplot(my_sims_long, aes(x=value))  +

        geom_histogram(aes(y=..density..),bins=30,fill="lightblue",col="
         +
  geom_density()+
  facet_wrap(~name,scales="free")
```

# N=10000

```
p
```


Normal
Uniform
Beta
density
value
0.4
0.3
0.2
0.1
0.0
-4
-2
0
2
4
3
4
5
6
7
2.5
2.0
1.5
1.0
0.5
0.00
0.25
0.50
0.75
1.00

# Setting the seed for the RNG

```
set.seed(8022012)
rnorm(5)
```

```
[1]  0.18058 -0.40994  0.27908  0.85880 -2.35079
```

```
runif(5)
```

```
[1] 0.76560 0.36905 0.79943 0.66251 0.70653
```

```
set.seed(8022012)
rnorm(6)
```

```
[1]  0.18058 -0.40994  0.27908  0.85880 -2.35079  0.72442
```

```
runif(5)
```

```
[1] 0.79943 0.66251 0.70653 0.37248 0.26014
```

# Back to VaR

# First step

```r
set.seed(19200)
## Set the number of stocks
J <- 10

## Set the weights of the stocks
weights<-rep(50,J)


## If X_j, t+1 are normal
sigma <- 0.2
X_t_plus_1 <- rnorm(J,0,sigma)

## Compute L_t+1
L_t_plus_1 <- - (weights %*% (exp(X_t_plus_1) -1))
```

# First step

```
as.numeric(L_t_plus_1)# Loss
```

```
[1] -25.458
```

```
as.numeric(L_t_plus_1/sum(weights) * 100) # Percent Loss/Gain
```

```
[1] -5.0916
```

# Now to replicates

```
one_VaR_sim <- function(reps=100000,J, weights, sigma,
                        alphas=c(0.95,0.99,0.999)){

  ## If X_j, t+1 are normal
  X_t_plus_1 <- matrix(rnorm(J*reps,0,sigma),ncol=reps)

  ## Compute L_t+1
  L_t_plus_1 <- - weights %*% (exp(X_t_plus_1) -1)

  ## Find VaR for different alphas
  VaR_alpha <- quantile(L_t_plus_1,alphas)

  return(tibble(alpha=alphas,VaR=VaR_alpha))

}
```

# Now to replicates

```
set.seed(19752626)
## One run
one_VaR_sim(reps=1000,J=J,weights=weights,sigma=sigma)
```

```
# A tibble: 3 x 2
  alpha   VaR
  <dbl> <dbl>
1 0.95   39.9
2 0.99   59.1
3 0.999  74.3
```

diff

```
## A second run
one_VaR_sim(reps=1000,J=J,weights=weights,sigma=sigma)
```

```
# A tibble: 3 x 2
  alpha   VaR
  <dbl> <dbl>
1 0.95   44.7
2 0.99   65.7
3 0.999  89.4
```

# Now to replicates

```
set.seed(19752626)
## One run
one_VaR_sim(reps=100000,J=J,weights=weights,sigma=sigma)
```

```
# A tibble: 3 x 2
  alpha   VaR
  <dbl> <dbl>
1 0.95   41.9
2 0.99   61.5
3 0.999  81.8
```

diff decreases

```
## A second run
one_VaR_sim(reps=100000,J=J,weights=weights,sigma=sigma)
```

```
# A tibble: 3 x 2
  alpha   VaR
  <dbl> <dbl>
1 0.95   41.2
2 0.99   61.1
3 0.999  82.2
```

# Easier replication

#times

```
my_VaR_reps<-replicate(100,
                       one_VaR_sim(reps=10000,J=J,
                                   weights=weights,sigma=sigma))
dim(my_VaR_reps)
```

↑ function call

```
[1]   2 100
```

alpha (3x1) ...
VaR (3x1) ...

```
myresults<-apply(aperm(my_VaR_reps,c(2,1)),2,unlist)
```

array version of transpose

⇒ alpha (3x1) ⋮ VaR (3x1) ⋮

# Easier replication

```
head(myresults)
```

```
     alpha    VaR
[1,] 0.950 41.758
[2,] 0.990 62.516
[3,] 0.999 86.176
[4,] 0.950 41.803
[5,] 0.990 61.920
[6,] 0.999 78.671
```

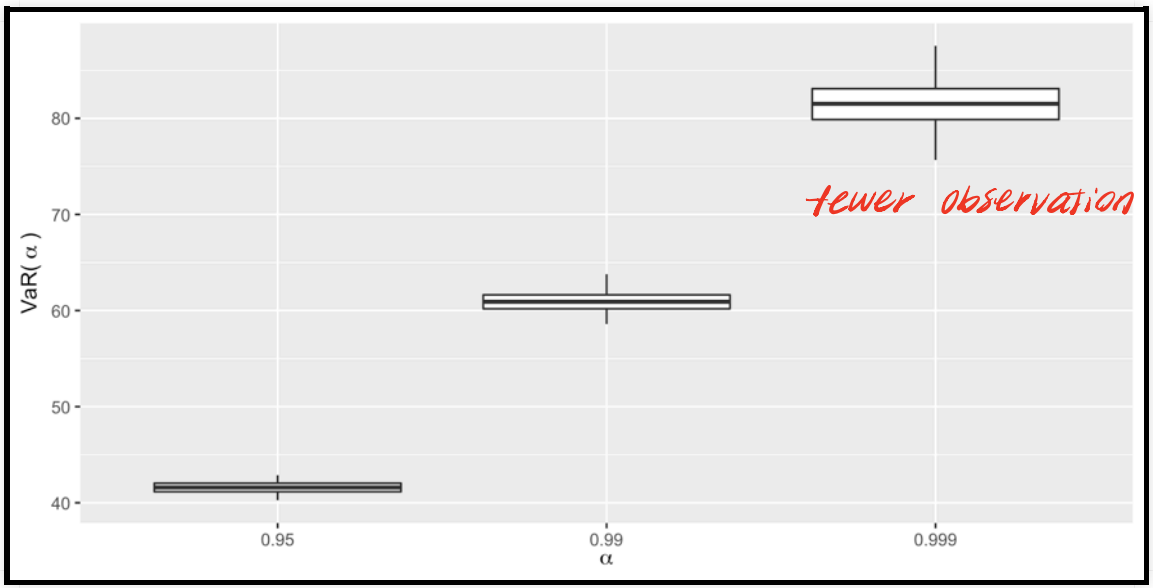
# Easier replication

```
myresults<-as_tibble(myresults) %>% mutate(Rep=rep(1:100,each=3))
head(myresults)
```

```
# A tibble: 6 x 3
  alpha   VaR   Rep
  <dbl> <dbl> <int>
1 0.95   41.8     1
2 0.99   62.5     1
3 0.999  86.2     1
4 0.95   41.8     2
5 0.99   61.9     2
6 0.999  78.7     2
```

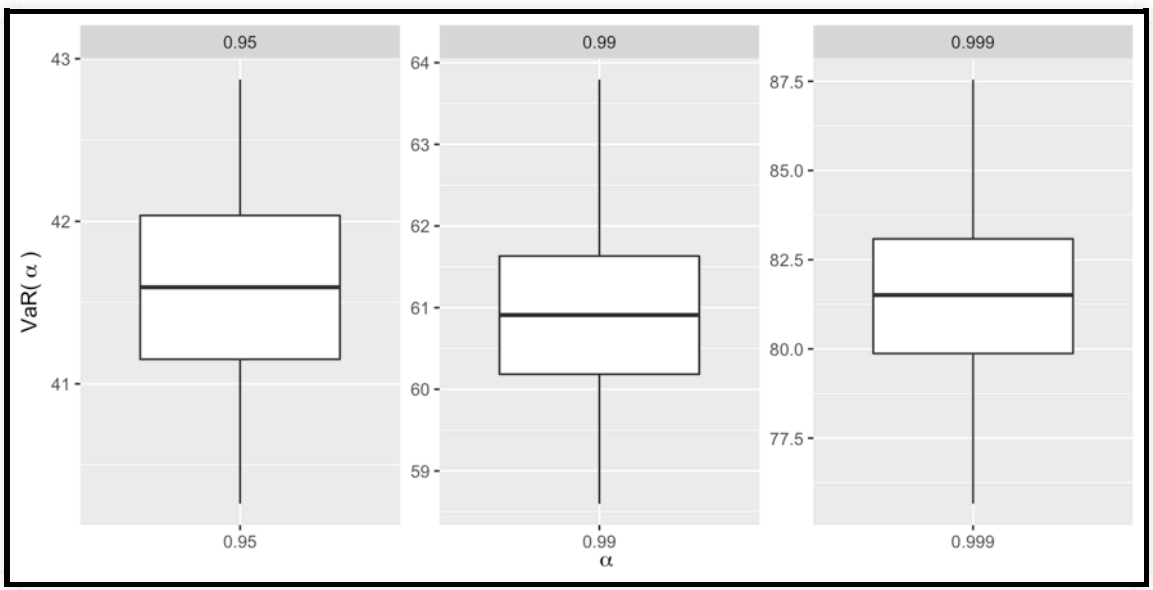
# Easier replication

```
ggplot(myresults, aes(x=factor(alpha),y=VaR)) + geom_boxplot() +
  labs(x=bquote(alpha),y=bquote("VaR("~alpha~")"))
```



# Easier replication

```
ggplot(myresults, aes(x=factor(alpha),y=VaR)) + geom_boxplot() +
  facet_wrap(~factor(alpha),scales="free")+
  labs(x=bquote(alpha),y=bquote("VaR("~alpha~")"))
```



# Even easier replication

```
set.seed(19752626)

my_results_tbl <- c(1:100) %>%
  map_dfr(~one_VaR_sim(reps=10000,J=J,
                       weights=weights,sigma=sigma)%>%
            mutate(Rep=.x))

head(my_results_tbl)
```

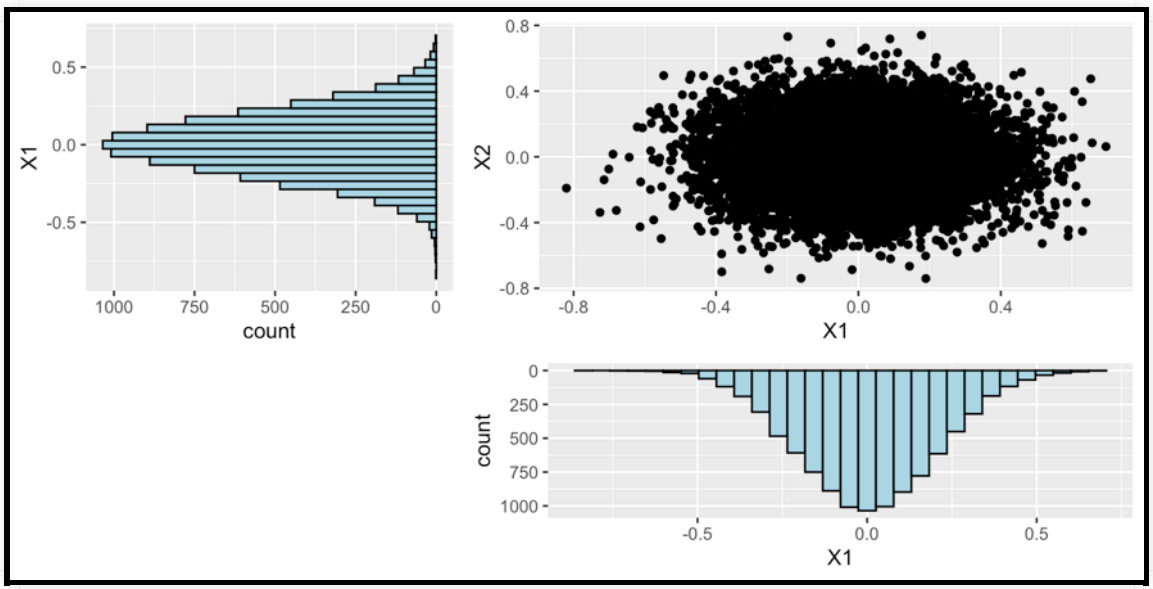
```
# A tibble: 6 x 3
  alpha   VaR   Rep
  <dbl> <dbl> <int>
1 0.95   41.8     1
2 0.99   61.6     1
3 0.999  85.1     1
4 0.95   41.4     2
5 0.99   58.4     2
6 0.999  81.3     2
```

# Assessing sensitivity to assumptions

```
X <- tibble(X1=rnorm(10000,0,0.2),X2=rnorm(10000,0,0.2))
library(gridExtra)
plot_my_stuff<-function(X){
p1<-ggplot(X,aes(x=X1)) + geom_histogram(bins=30,fill="lightblue",
                                         col="black") +
        scale_y_reverse() +
  coord_flip()
p2<-ggplot(X,aes(x=X1,y=X2)) + geom_point()
p3<-ggplot(X,aes(x=X1)) + geom_histogram(bins=30,fill="lightblue",
                                         col="black")+
        scale_y_reverse()
plotlist<-list(p1,p2,p3)
grid.arrange(p1,p2,p3,layout_matrix=
               rbind(c(1,1,2,2,2),c(1,1,2,2,2),
                     c(1,1,2,2,2),c(NA,NA,3,3,3),
                     c(NA,NA,3,3,3)))
```

# Assessing sensitivity to assumptions

```
plot_my_stuff(X)
```



# Simulating dependence

```
rho<-0.5
cov_mat<-matrix(rho,ncol=2,nrow=2)
diag(cov_mat)<-1
cov_mat
```

```
     [,1] [,2]
[1,]  1.0  0.5
[2,]  0.5  1.0
```

```
chol(cov_mat)
```

decomposition

$W = U^T U$

- upper triangular matrix

```
     [,1]    [,2]
[1,]    1 0.50000
[2,]    0 0.86603
```

# Simulating dependence

```
X_star <- matrix(rnorm(10000*2),ncol=2)
X<-as_tibble(X_star%*%chol(cov_mat))
```

```
Warning: `as_tibble.matrix()` requires a matrix with column names or a `
This warning is displayed once per session.
```

```
X<-X*0.2 ### Scaling step
colnames(X)<-c("X1","X2")
```

$W_{n\times 2}$ , $A_{2\times 2}$

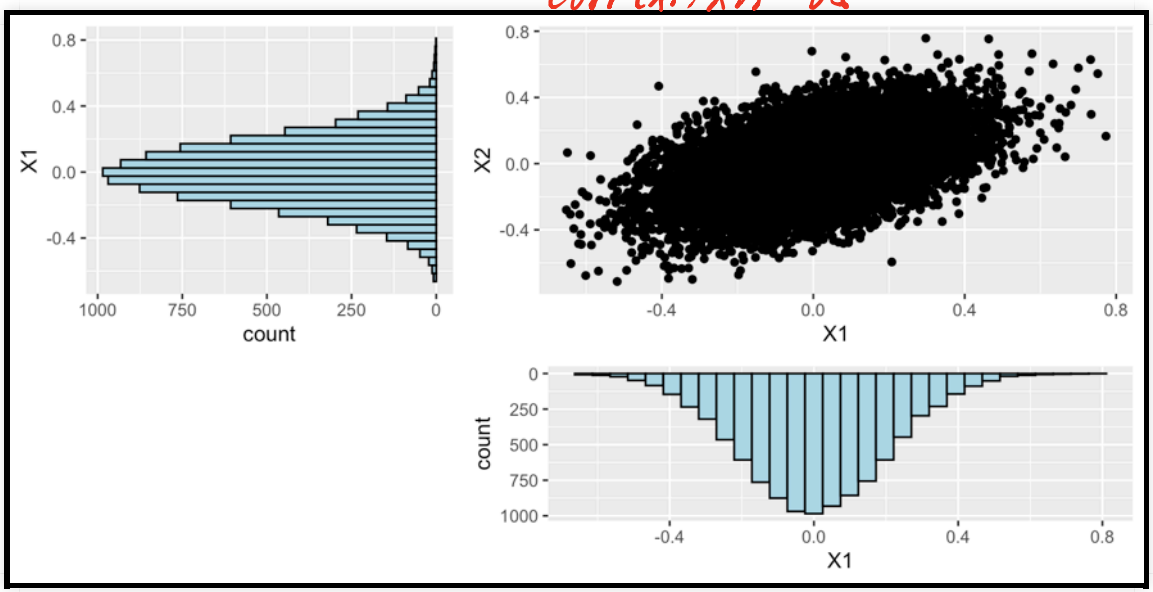
$Var(WA) = A^T Var(W) A$

$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

# Simulating dependence

```
plot_my_stuff(X)
```

$Corr(X_1, X_2) = 0.5$



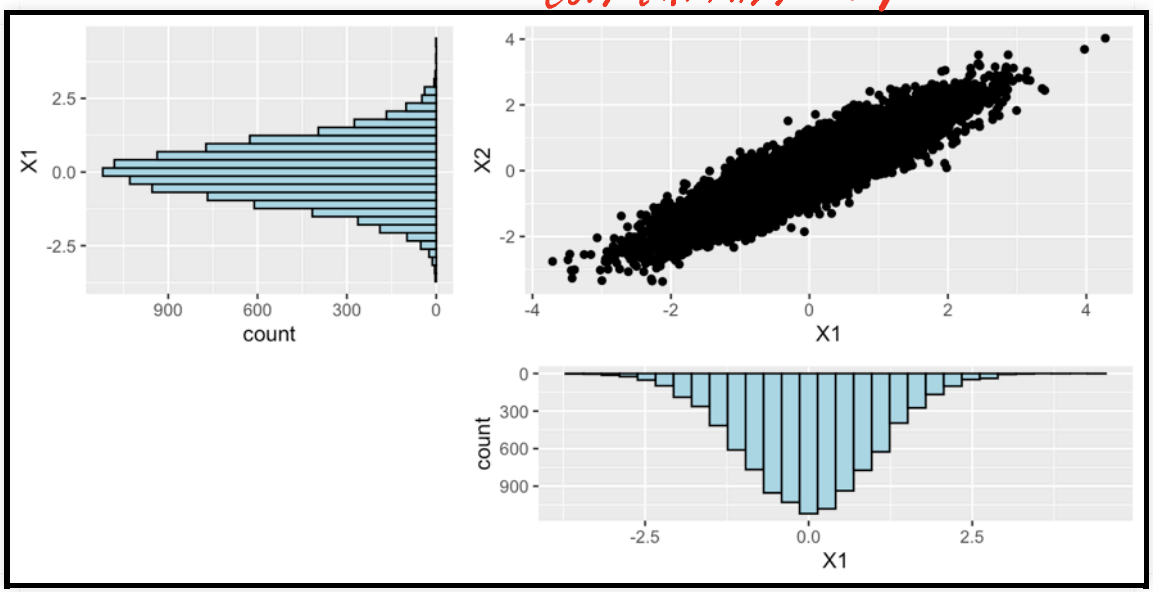
# Simulating dependence

```
rho<-0.9
cov_mat<-matrix(rho,ncol=2,nrow=2)
diag(cov_mat)<-1

X_star <- matrix(rnorm(10000*2),ncol=2)
X<-as_tibble(X_star%*%chol(cov_mat))

colnames(X)<-c("X1","X2")
```

# Simulating dependence

```
plot_my_stuff(X)
```

$Corr(X_1, X_2) = 0.9$



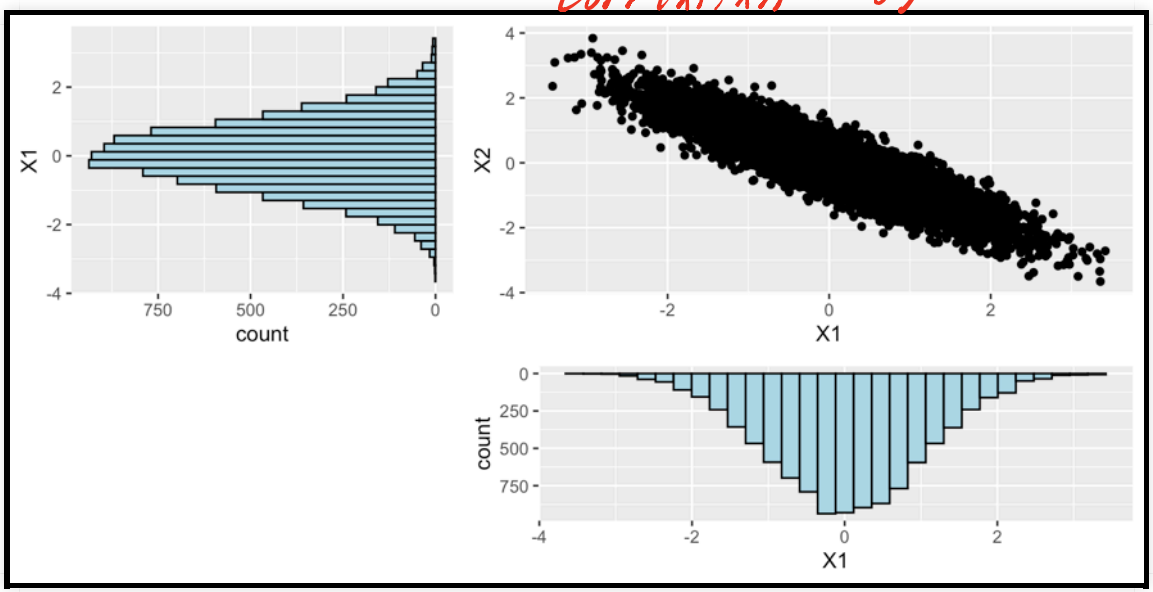
# Simulating dependence

```
rho<- -0.9
cov_mat<-matrix(rho,ncol=2,nrow=2)
diag(cov_mat)<-1

X_star <- matrix(rnorm(10000*2),ncol=2)
X<-as_tibble(X_star%*%chol(cov_mat))

colnames(X)<-c("X1","X2")
```

# Simulating dependence

```
plot_my_stuff(X)
```

$Corr(X_1, X_2) = -0.9$



# Simulating dependence

```r
one_VaR_dep_sim <- function(reps=100000,J, weights, sigma,
                            alphas=c(0.95,0.99,0.999),mean=0,
                            rho=0){
  ## Setting covariance matrix
  cov_mat<-matrix(rho,ncol=J,nrow=J)
  diag(cov_mat)<-1
  X_star <- matrix(rnorm(10000*J),ncol=J)
  X_t_plus_1 <- t(sigma*X_star%*%chol(cov_mat)) ## transpose for old
        code
  ## Compute L_t+1
  L_t_plus_1 <- - weights %*% (exp(X_t_plus_1) -1)
  ## Find VaR for different alphas
  VaR_alpha <- quantile(L_t_plus_1,alphas)
  return(tibble(alpha=alphas,VaR=VaR_alpha))
}
```

independent

dependent normal

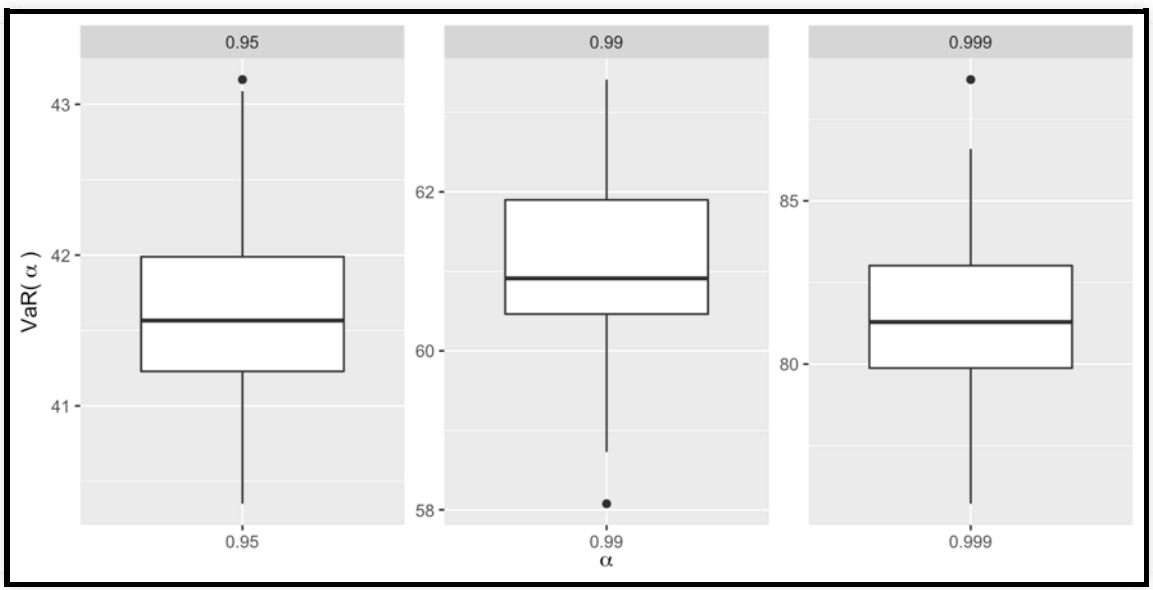
# Simulating dependence

```
set.seed(19752626)

my_results_tbl <- c(1:100) %>%
  map_dfr(~one_VaR_dep_sim(reps=10000,J=J,
                           weights=weights,sigma=sigma,rho=0)%>%
            mutate(Rep=.x))

my_results_tbl %>% group_by(alpha) %>% summarise(Est=mean(VaR))
```

```
# A tibble: 3 x 2
  alpha   Est
  <dbl> <dbl>
1 0.95   41.7
2 0.99   61.1
3 0.999  81.5
```

# Simulating dependence

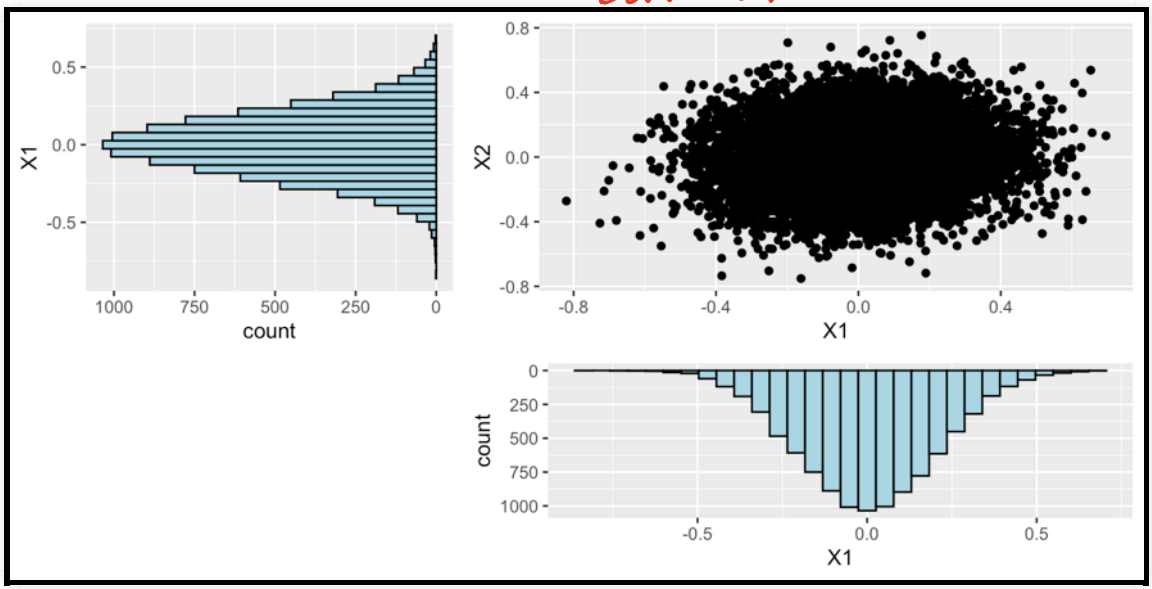
```
ggplot(my_results_tbl, aes(x=factor(alpha),y=VaR)) + geom_boxplot() +
  facet_wrap(~factor(alpha),scales="free")+
  labs(x=bquote(alpha),y=bquote("VaR("~alpha~")"))
```



# Simulating dependence

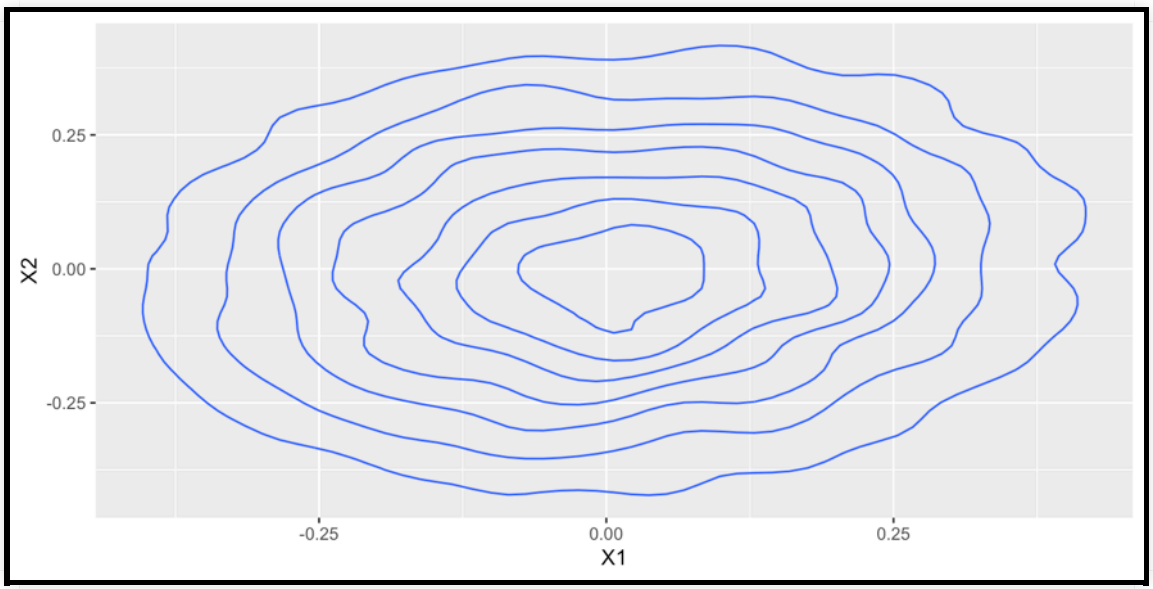
```r
rho<- 0.1
cov_mat<-matrix(rho,ncol=2,nrow=2)
diag(cov_mat)<-1
X_star <- 0.2*matrix(rnorm(10000*2),ncol=2)
X<-as_tibble(X_star%*%chol(cov_mat))
colnames(X)<-c("X1","X2")
plot_my_stuff(X)
```

Corr = 0.1


X1
0.5
0.0
-0.5
1000
750
500
250
0
count
X2
0.8
0.4
0.0
-0.4
-0.8
-0.8
-0.4
0.0
0.4
X1
count
0
250
500
750
1000
-0.5
0.0
0.5
X1

# Simulating dependence

```
ggplot(X,aes(x=X1,y=X2)) + geom_density_2d()
```



# Simulating dependence

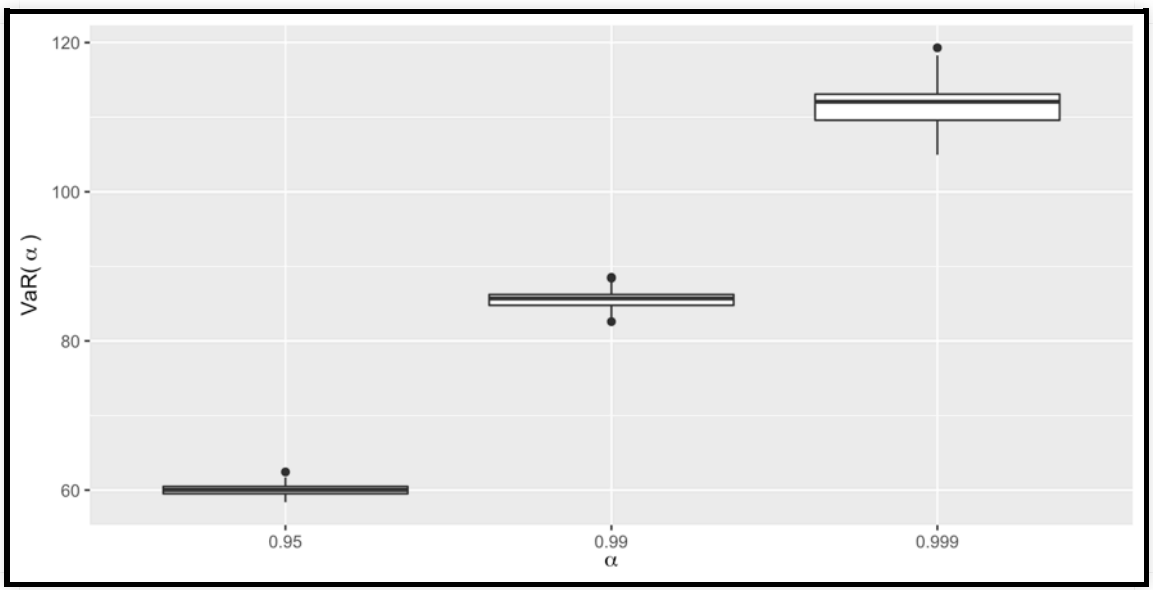
```
set.seed(19752626)

my_results_tbl_0.1 <- c(1:100) %>%
  map_dfr(~one_VaR_dep_sim(reps=10000,J=J,
                           weights=weights,sigma=sigma,rho=0.1)%>%
            mutate(Rep=.x))

my_results_tbl_0.1 %>% group_by(alpha) %>% summarise(Est=mean(VaR))
```

```
# A tibble: 3 x 2
  alpha   Est
  <dbl> <dbl>
1 0.95   60.0
2 0.99   85.6
3 0.999 112.
```

risk increases a lot due to small correlation.

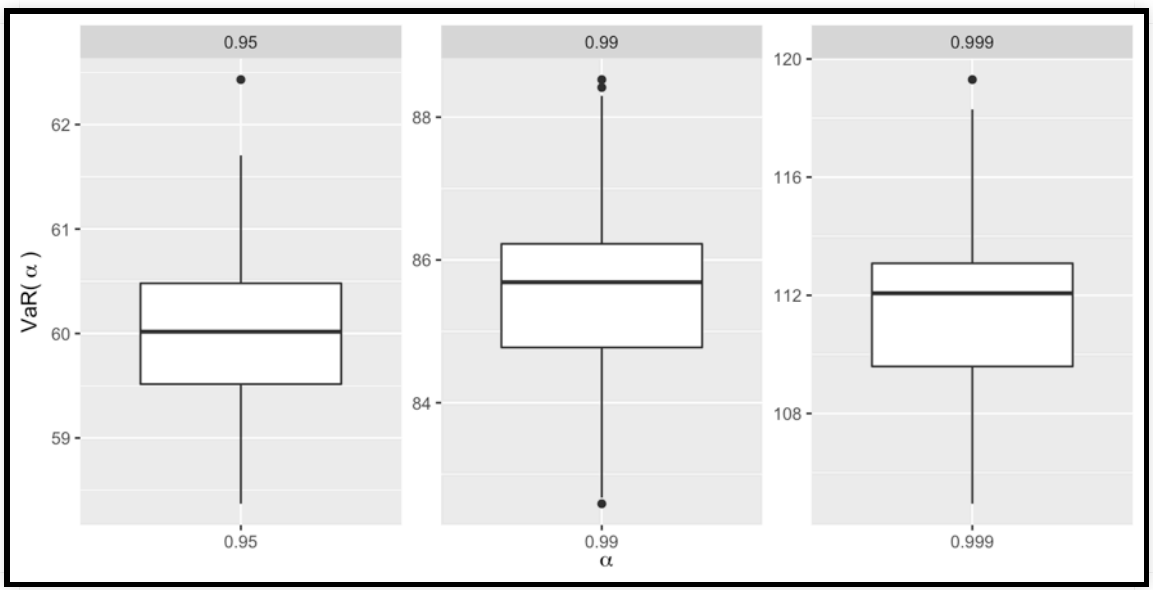
# Simulating dependence

```
ggplot(my_results_tbl_0.1, aes(x=factor(alpha),y=VaR)) + geom_boxplot()
        +
  labs(x=bquote(alpha),y=bquote("VaR("~alpha~")"))
```



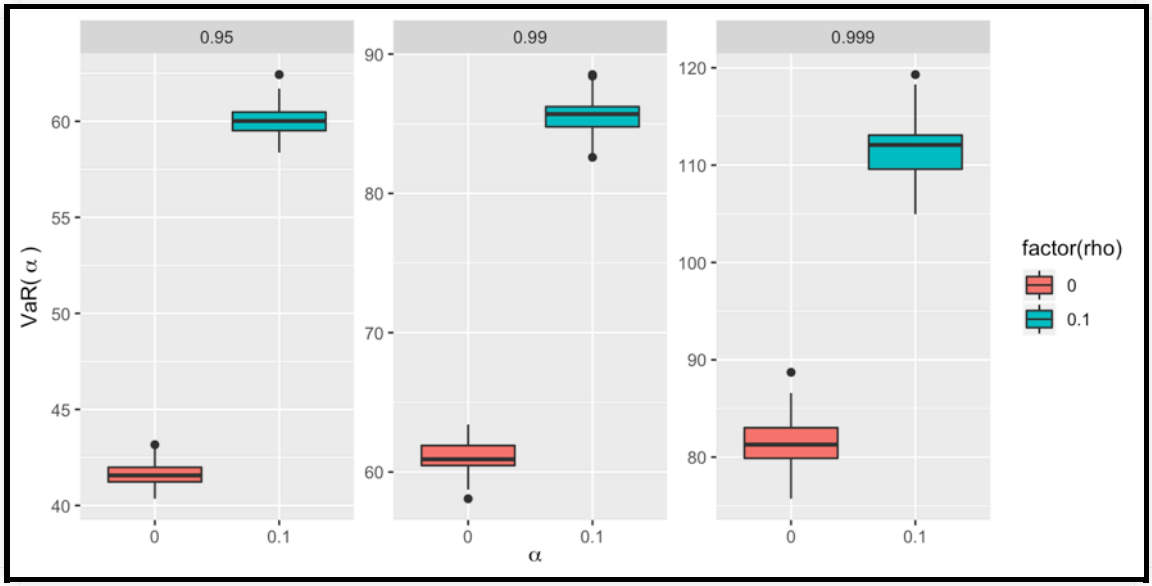
# Simulating dependence

```
ggplot(my_results_tbl_0.1, aes(x=factor(alpha),y=VaR)) + geom_boxplot()
        +
  facet_wrap(~factor(alpha),scales="free")+
  labs(x=bquote(alpha),y=bquote("VaR("~alpha~")"))
```



# Simulating dependence

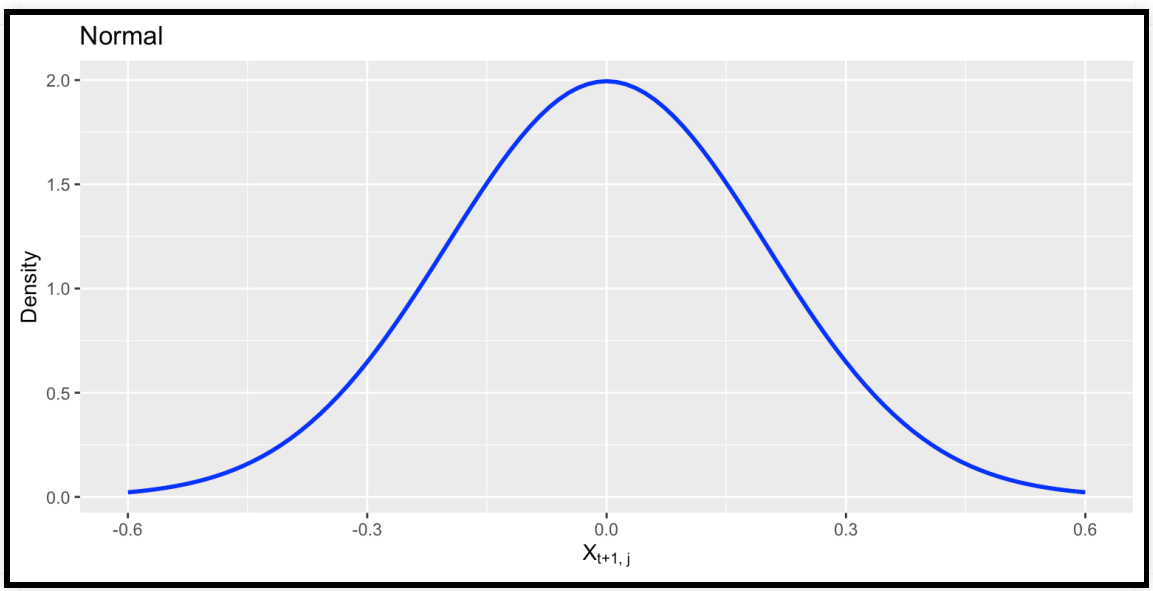
```
my_results_tbl_0.1 <- my_results_tbl_0.1 %>% mutate(rho=0.1)
my_results_tbl <- my_results_tbl %>% mutate(rho=0)
both_results_tbl<-bind_rows(my_results_tbl_0.1,my_results_tbl)
ggplot(both_results_tbl, aes(x=factor(rho),y=VaR,
                             fill=factor(rho))) + geom_boxplot() +
  facet_wrap(~factor(alpha),scales="free")+
  labs(x=bquote(alpha),y=bquote("VaR("~alpha~")"))
```


0.95
0.99
0.999
VaR( α )
α
0
0.1
factor(rho)
0
0.1

# Designing a multi-factor experiment

# Allowing for more extreme observations

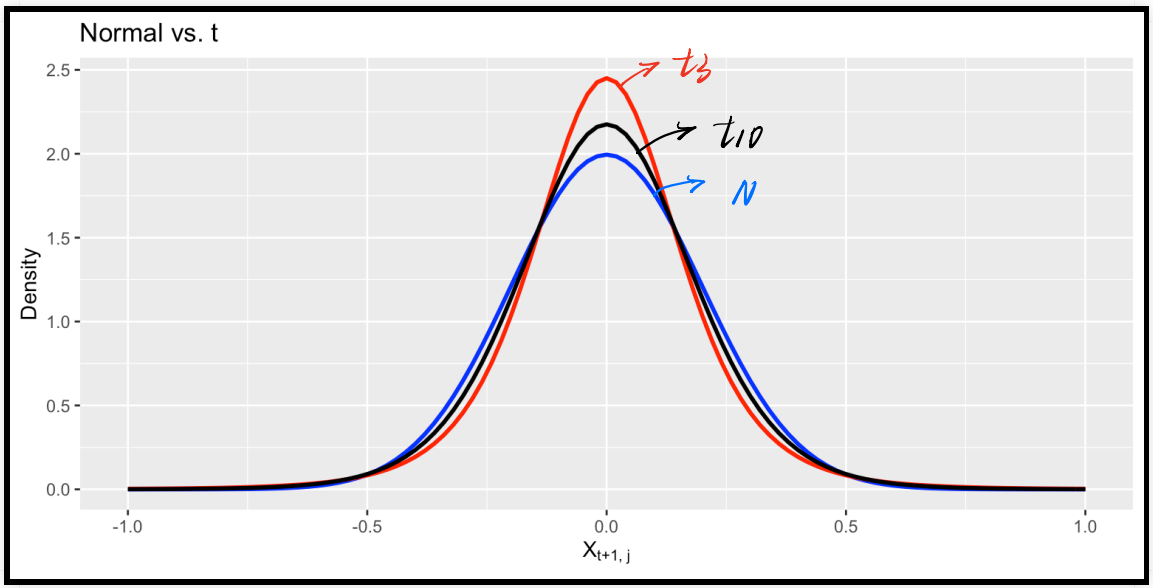
```
plot_grid<-tibble(x=seq(-0.6,0.6,length=1000))
ggplot(plot_grid,aes(x=x)) +
  stat_function(fun=dnorm,args=list(mean=0,sd=0.2),
                lwd=1,col="blue") +
  labs(x=bquote(X["t+1, j"]),y="Density", title="Normal")
```



# Allowing for more extreme observations

```
plot_grid<-tibble(x=seq(-1,1,length=1000))
ggplot(plot_grid,aes(x=x)) +
  stat_function(fun=dnorm,args=list(mean=0,sd=0.2),
                lwd=1,col="blue") +
  labs(x=bquote(X["t+1, j"]),y="Density", title="Normal") +
  stat_function(fun=function(x,df,s){
                    scale_c<-s/sqrt(df/(df-2))
                    1/scale_c*dt(x/scale_c,df)
                  },args=list(df=5,s=0.2), lwd=1,col="red") +
  labs(x=bquote(X["t+1, j"]),y="Density", title="Normal vs. t") +
  stat_function(fun=function(x,df,s){
                    scale_c<-s/sqrt(df/(df-2))
                    1/scale_c*dt(x/scale_c,df)
                  },args=list(df=10,s=0.2),lwd=1,col="black") +
  labs(x=bquote(X["t+1, j"]),y="Density", title="Normal vs. t")
```

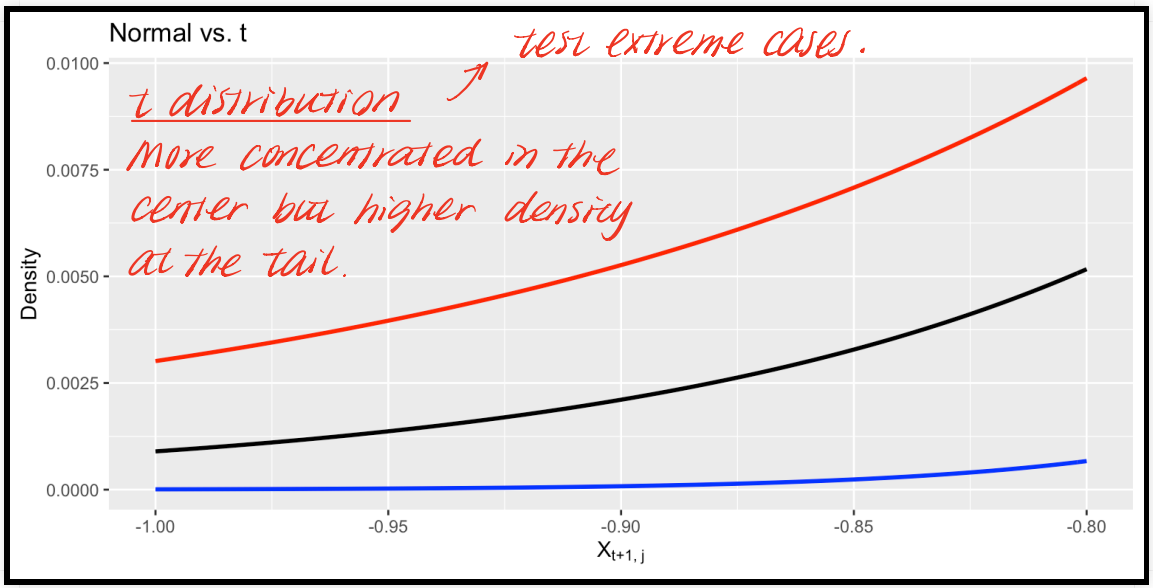
# Allowing for more extreme observations



# Allowing for more extreme observations

```
plot_grid<-tibble(x=seq(-1,1,length=1000))
ggplot(plot_grid,aes(x=x)) +
  stat_function(fun=dnorm,args=list(mean=0,sd=0.2),
                lwd=1,col="blue") +
  labs(x=bquote(X["t+1, j"]),y="Density", title="Normal") +
  stat_function(fun=function(x,df,s){
                    scale_c<-s/sqrt(df/(df-2))
                    1/scale_c*dt(x/scale_c,df)
                  },args=list(df=5,s=0.2),lwd=1,col="red") +
  labs(x=bquote(X["t+1, j"]),y="Density", title="Normal vs. t") +
  stat_function(fun=function(x,df,s){
                    scale_c<-s/sqrt(df/(df-2))
                    1/scale_c*dt(x/scale_c,df)},
                args=list(df=10,s=0.2),lwd=1,col="black") +
  labs(x=bquote(X["t+1, j"]),y="Density", title="Normal vs. t") +
  xlim(c(-1,-0.8))
```

# Allowing for more extreme observations



# Generating dependent $t-$ random variables

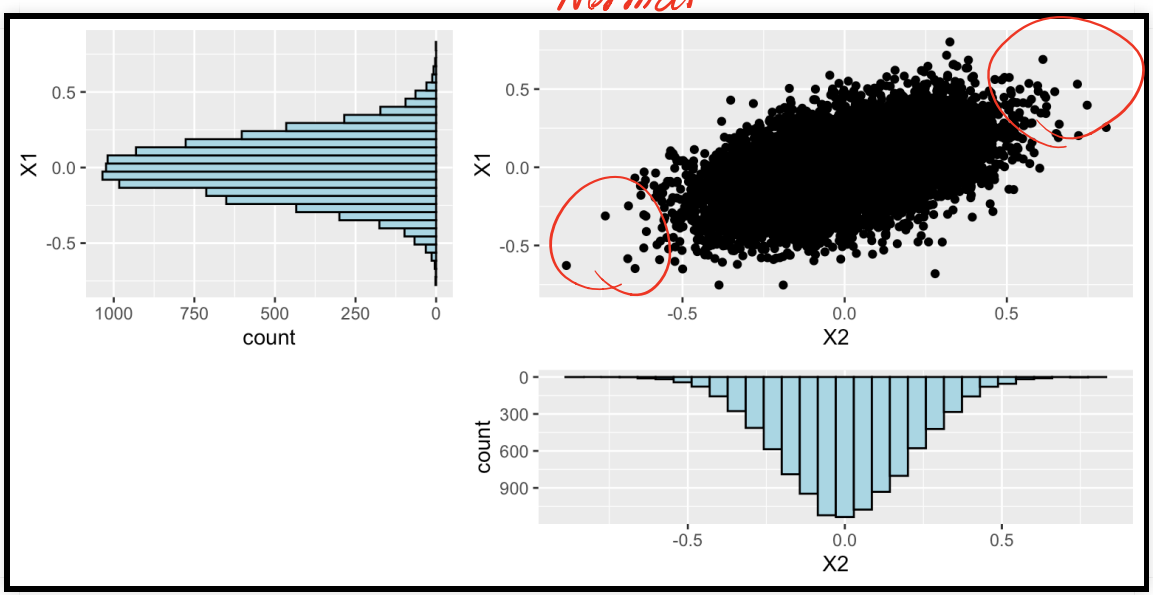
```
rho<-0.5
df<-5
sigma<-0.2
J<-2
cov_mat<-matrix(rho,ncol=J,nrow=J)
diag(cov_mat)<-1
t_scale<-rchisq(10000*J,df)
X_star <- matrix(rnorm(10000*J),ncol=J)
X_tilde<- matrix(rnorm(10000*J)/sqrt(df/t_scale),ncol=J)
X_t_plus_1 <- t(sigma*X_star%*%chol(cov_mat)) ## transpose for old code
X_t_t_plus_1<- t(sigma*X_tilde%*%chol(cov_mat)) ## transpose for old
        code
X<-as_tibble(t(X_t_plus_1))
X_t<-as_tibble(t(X_t_t_plus_1))

colnames(X)<-c("X1","X2")
```

$\chi^2(0,\infty)$

$t$ distribution

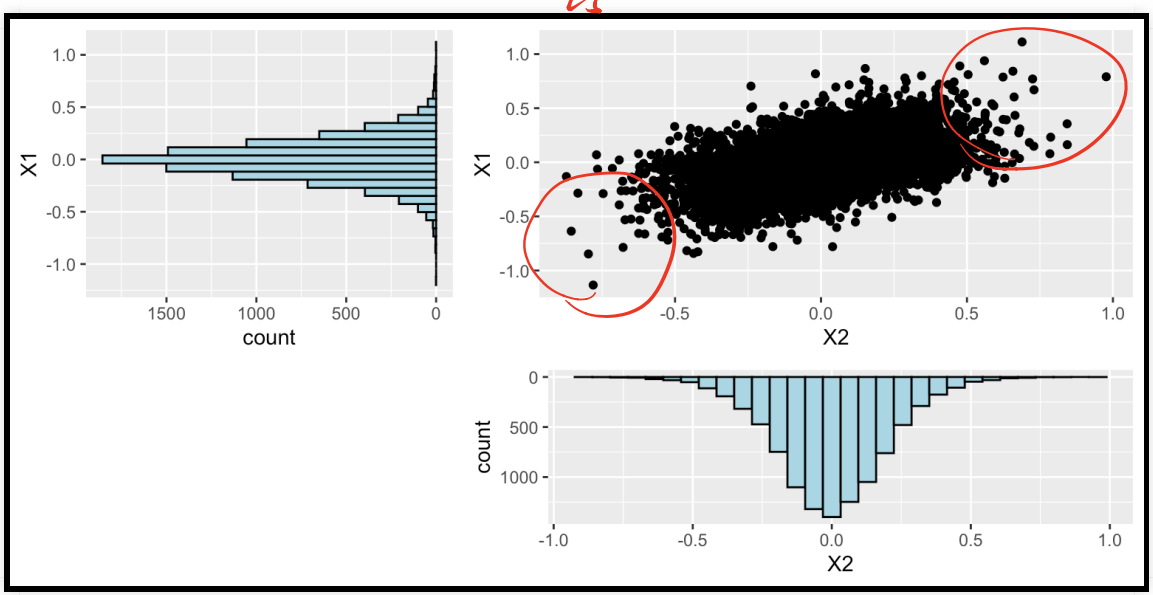
# Generating dependent $t$– random variables

```
plot_my_stuff(X)
```



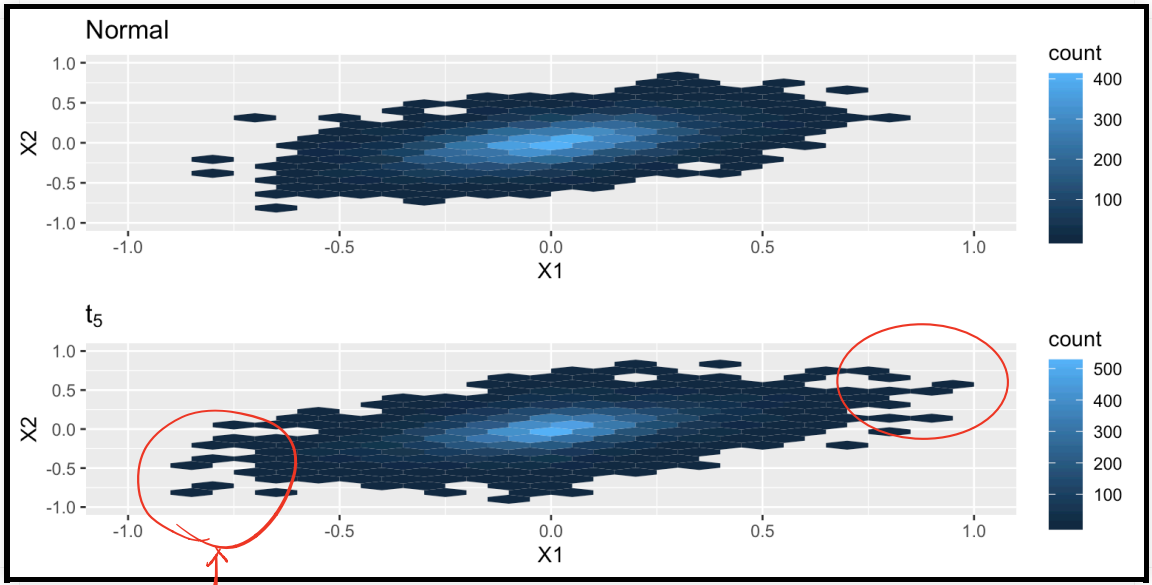
# Generating dependent $t$– random variables

```
plot_my_stuff(X_t)
```



# Generating dependent $t-$ random variables

```
p1<-ggplot(X,aes(x=X1,y=X2)) + geom_hex(bins=20) + ggtitle("Normal") +
  ylim(c(-1,1)) + xlim(c(-1,1))
p2<-ggplot(X_t,aes(x=X1,y=X2)) + geom_hex(bins=20) +
        ggtitle(bquote(t[5])) +
  ylim(c(-1,1)) + xlim(c(-1,1))
grid.arrange(p1,p2)
```


Normal
X2
X1
count
400
300
200
100
$t_5$
X2
X1
count
500
400
300
200
100
big trouble

# Factorial simulation design

```r
### First define correlations and df

rho<-c(0,0.1,0.4)
df<-c(3,10, 100)
iter<-c(1:100) # Number of iterations of each combination

conditions<-crossing(rho,df,iter)
head(conditions %>% arrange(iter))
```

df → ∞  t → N

```
# A tibble: 6 x 3
    rho    df  iter
  <dbl> <dbl> <int>
1   0       3     1
2   0      10     1
3   0     100     1
4   0.1     3     1
5   0.1    10     1
6   0.1   100     1
```

# Factorial simulation design

default arguments

```
one_VaR_dep_t_sim <- function(reps=100000,J=10,weights=rep(50,J),
                              sigma=0.2,
                              alphas=c(0.95,0.99,0.999),
                              rho=0,df=3){
  ## Setting covariance matrix
  cov_mat<-matrix(rho,ncol=J,nrow=J)
  diag(cov_mat)<-1
  t_scale<-rchisq(reps*J,df)
  X_tilde<- matrix(rnorm(reps*J)/sqrt(df/t_scale),ncol=J)
  X_t_t_plus_1<- t(sigma*X_tilde%*%chol(cov_mat)) ## transpose for old
        code
  X_t<-as_tibble(t(X_t_t_plus_1))
  L_t_plus_1 <- - weights %*% (exp(X_t_t_plus_1) -1)
  ## Find VaR for different alphas
  VaR_alpha <- quantile(L_t_plus_1,alphas)
  return(tibble(alpha=alphas,VaR=VaR_alpha))
```

# Factorial simulation design

```
library(tictoc)
tic()
results<-conditions %>% group_by(rho,df, iter) %>%
  mutate(
    ind_results = pmap(list(rho=rho,df=df),one_VaR_dep_t_sim)

  )
toc()
```

⇒ each row

time

arguments

function

```
234.263 sec elapsed
```

# Factorial simulation design

```
head(results %>% arrange(iter))
```

```
# A tibble: 6 x 4
# Groups:   rho, df, iter [6]
    rho    df  iter ind_results
  <dbl> <dbl> <int> <list>
1   0       3     1 <tibble [3 × 2]>
2   0      10     1 <tibble [3 × 2]>
3   0     100     1 <tibble [3 × 2]>
4   0.1     3     1 <tibble [3 × 2]>
5   0.1    10     1 <tibble [3 × 2]>
6   0.1   100     1 <tibble [3 × 2]>
```

# Factorial simulation design

```
results<-results %>% unnest(cols=ind_results)
head(results)
```

```
# A tibble: 6 x 5
# Groups:   rho, df, iter [2]
    rho    df  iter alpha   VaR
  <dbl> <dbl> <int> <dbl> <dbl>
1     0     3     1 0.95   41.3
2     0     3     1 0.99   60.8
3     0     3     1 0.999  81.9
4     0     3     2 0.95   40.8
5     0     3     2 0.99   59.9
6     0     3     2 0.999  82.1
```

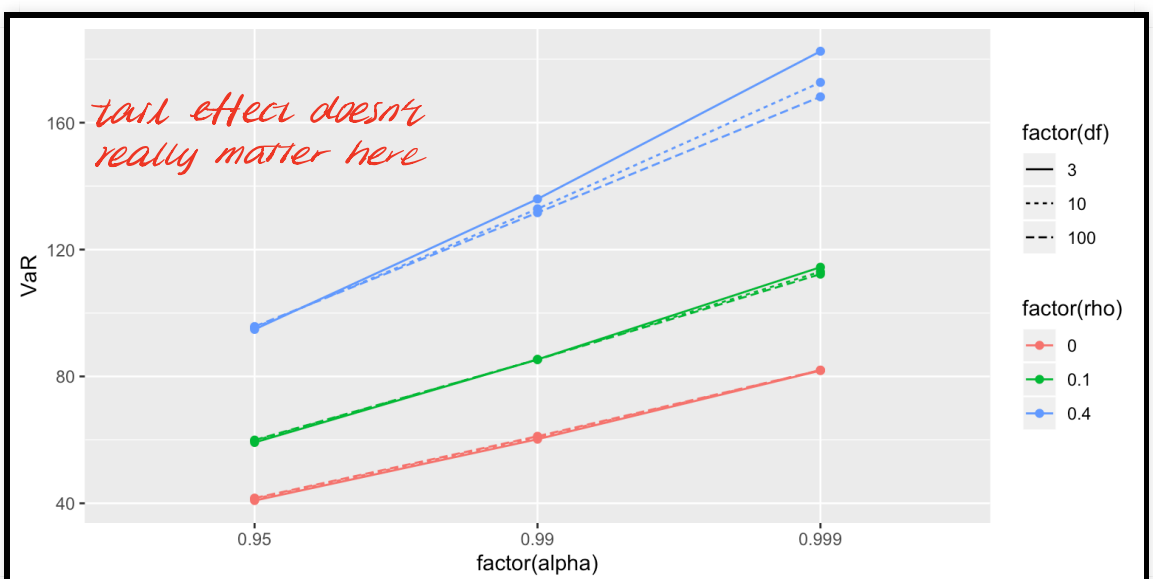
# Factorial simulation design

```
summarise_for_plot<-results %>% group_by(rho,df,alpha) %>%
  summarise(VaR = mean(VaR))
head(summarise_for_plot)
```

```
# A tibble: 6 x 4
# Groups:   rho, df [2]
    rho    df alpha   VaR
  <dbl> <dbl> <dbl> <dbl>
1     0     3 0.95   40.9
2     0     3 0.99   60.2
3     0     3 0.999  81.8
4     0    10 0.95   41.4
5     0    10 0.99   60.8
6     0    10 0.999  82.0
```

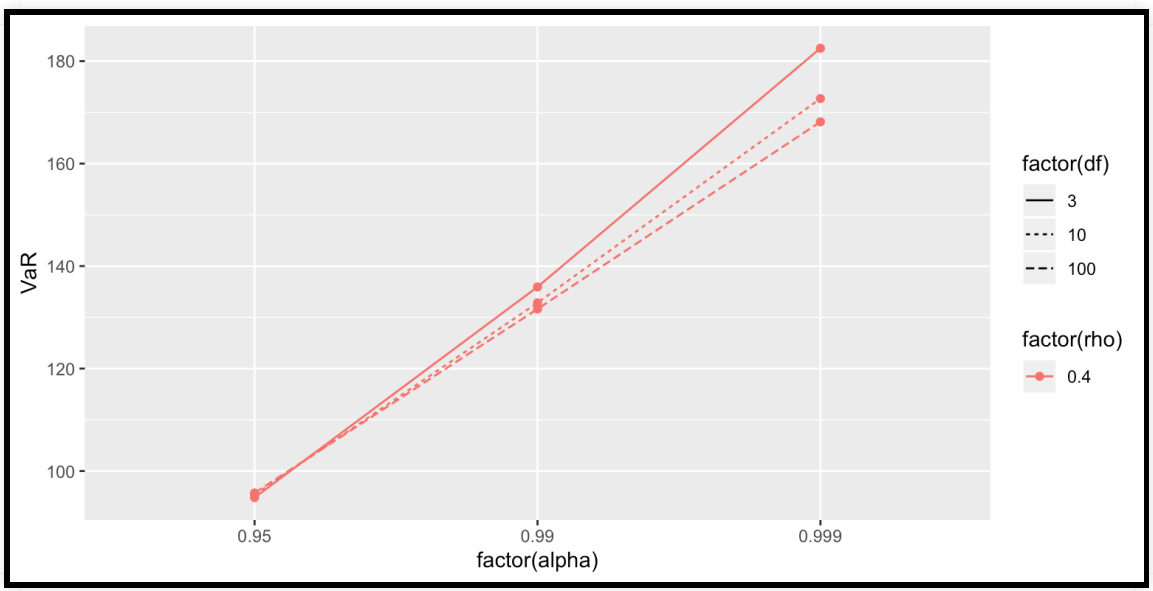
# Factorial simulation design

```
ggplot(summarise_for_plot,aes(x=factor(alpha),y=VaR,
                              col=factor(rho),
                              linetype=factor(df),
                              group=interaction(df,rho))) +
  geom_point() +geom_line()
```



# Factorial simulation design

```
ggplot(summarise_for_plot %>%
         filter(rho==0.4),aes(x=factor(alpha),y=VaR,
                              col=factor(rho),linetype=factor(df),
                              group=interaction(df,rho))) +
  geom_point() +geom_line()
```

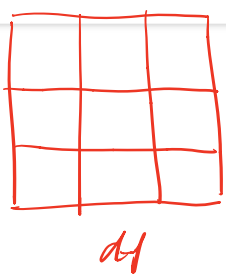


# Simsalapar

Not in final

```
library(simsalapar)
rho<-c(0,0.1,0.4)
df<-c(3,10, 100)

varList<-varlist(
  n.sim=list(type="N",expr=quote(N[sim]), value=100),
  rho = list(type="grid",value=rho),
  df = list(type="grid", value=df)
)
```

number of replication




type = "inner"
to same dataset using different variables.

# Simsalapar

```
do_one_VaR_dep_t_sim <- function(reps=100000,J=10,weights=rep(50,J),
                                 sigma=0.2,
                                 alphas=c(0.95,0.99,0.999),
                                 rho=0,df=3){
  ## Setting covariance matrix
  cov_mat<-matrix(rho,ncol=J,nrow=J)
  diag(cov_mat)<-1
  t_scale<-rchisq(reps*J,df)
  X_tilde<- matrix(rnorm(reps*J)/sqrt(df/t_scale),ncol=J)
  X_t_t_plus_1<- t(sigma*X_tilde%*%chol(cov_mat)) ## transpose for old
        code
  X_t<-as_tibble(t(X_t_t_plus_1))
  L_t_plus_1 <- - weights %*% (exp(X_t_t_plus_1) -1)
  ## Find VaR for different alphas
  VaR_alpha <- quantile(L_t_plus_1,alphas)
  return(VaR_alpha) ### THIS IS DIFFERENT
```

numeric vector

# Simsalapar

do lapply ↘ 1 processors

```
tic()
res<-doMclapply(varList, sfile="res_lapply_seq.rds",
                doOne=do_one_VaR_dep_t_sim,
                monitor=interactive())
toc()
```

```
74.624 sec elapsed
```

```
class(res)
```

```
[1] "array"
```

```
dim(res)
```

```
  rho    df n.sim
    3     3   100
```

# Simsalapar

```
res[1,1,1]
```

```
[[1]]
[[1]]$value
   95%    99%  99.9%
40.945 60.086 82.011

[[1]]$error
NULL

[[1]]$warning
NULL

[[1]]$time
[1] 388
```

# Simsalapar

```
val <- getArray(res)
mytime <- getArray(res,"time")

df<-array2df(val)
str(df)
```

```
'data.frame':   2700 obs. of  5 variables:
 $ Var1 : Factor w/ 3 levels "95%","99%","99.9%": 1 2 3 1 2 3 1 2 3 1 ..
 $ rho  : Factor w/ 3 levels "0.0","0.1","0.4": 1 1 1 2 2 2 3 3 3 1 ...
 $ df   : Factor w/ 3 levels "3","10","100": 1 1 1 1 1 1 1 1 1 2 ...
 $ n.sim: Factor w/ 100 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 .
 $ value: num  40.9 60.1 82 59.5 85.6 ...
```

```
head(df)
```

```
   Var1 rho df n.sim   value
1   95% 0.0  3     1  40.945
2   99% 0.0  3     1  60.086
3 99.9% 0.0  3     1  82.011
4   95% 0.1  3     1  59.463
5   99% 0.1  3     1  85.644
6 99.9% 0.1  3     1 114.964
```