Introduction à matlab pour la banque et la finance

Valérie Monbet

IRMAR, Université de Rennes 1

Matlab

Matlab est un logiciel de calcul numérique, utilisé dans de nombreux domaines d'application. Il se fonde sur le calcul matriciel. Matlab est d'ailleurs un raccourci pour "Matrix Laboratory".

Compétences visées et accès à l'aide

Compétences visées

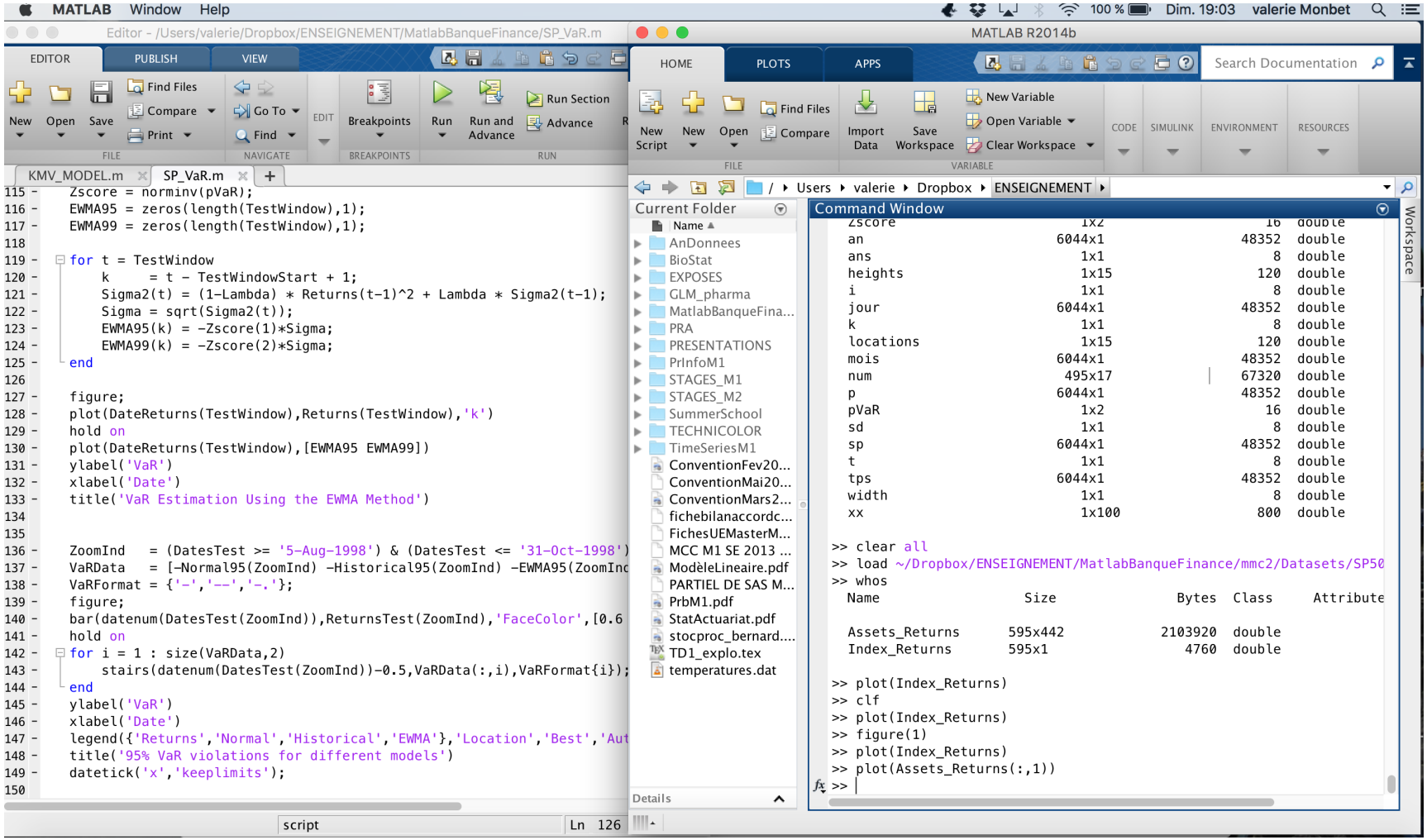
- Environnement matlab : vecteurs, matrices, opérations simples, etc.
- Graphiques
- Importer, exporter des données
- Régression, optimisation
- Copules, simulation Monte Carlo

Aide dans matlab

- `helpwin` ouvre une fenêtre contenant la liste des commandes Matlab ainsi que leurs documentations
- `help` donne la liste de toutes les commandes par thèmes
- `help` *nom* décrit la fonction *nom.m*
- `lookfor` *nom* recherche une instruction à partir du mot clé *nom*

Outline

Interface


MATLAB Window Help
Editor - /Users/valerie/Dropbox/ENSEIGNEMENT/MatlabBanqueFinance/SP_VaR.m
EDITOR PUBLISH VIEW
New Open Save Find Files Compare Print Go To Find Breakpoints Run Run and Advance Run Section Advance
KMV_MODEL.m SP_VaR.m
Zscore = norminv(pVaR);
EWMA95 = zeros(length(TestWindow),1);
EWMA99 = zeros(length(TestWindow),1);
for t = TestWindow
k = t - TestWindowStart + 1;
Sigma2(t) = (1-Lambda) * Returns(t-1)^2 + Lambda * Sigma2(t-1);
Sigma = sqrt(Sigma2(t));
EWMA95(k) = -Zscore(1)*Sigma;
EWMA99(k) = -Zscore(2)*Sigma;
end
figure;
plot(DateReturns(TestWindow),Returns(TestWindow),'k')
hold on
plot(DateReturns(TestWindow),[EWMA95 EWMA99])
ylabel('VaR')
xlabel('Date')
title('VaR Estimation Using the EWMA Method')
ZoomInd = (DatesTest >= '5-Aug-1998') & (DatesTest <= '31-Oct-1998')
VaRData = [-Normal95(ZoomInd) -Historical95(ZoomInd) -EWMA95(ZoomInd
VaRFormat = {'-','--','-.'};
figure;
bar(datenum(DatesTest(ZoomInd)),ReturnsTest(ZoomInd),'FaceColor',[0.6
hold on
for i = 1 : size(VaRData,2)
stairs(datenum(DatesTest(ZoomInd))-0.5,VaRData(:,i),VaRFormat{i});
end
ylabel('VaR')
xlabel('Date')
legend({'Returns','Normal','Historical','EWMA'},'Location','Best','Aut
title('95% VaR violations for different models')
datetick('x','keeplimits');
script Ln 126
MATLAB R2014b
HOME PLOTS APPS
New Script New Open Find Files Compare Import Data Save Workspace New Variable Open Variable Clear Workspace
Search Documentation
/ Users valerie Dropbox ENSEIGNEMENT
Current Folder
Name
AnDonnees
BioStat
EXPOSES
GLM_pharma
MatlabBanqueFina...
PRA
PRESENTATIONS
PrInfoM1
STAGES_M1
STAGES_M2
SummerSchool
TECHNICOLOR
TimeSeriesM1
ConventionFev20...
ConventionMai20...
ConventionMars2...
fichebilanaccordc...
FichesUEMasterM...
MCC M1 SE 2013 ...
ModèleLineaire.pdf
PARTIEL DE SAS M...
PrbM1.pdf
StatActuariat.pdf
stocproc_bernard....
TD1_explo.tex
temperatures.dat
Details
Command Window
Zscore 1x2 16 double
an 6044x1 48352 double
ans 1x1 8 double
heights 1x15 120 double
i 1x1 8 double
jour 6044x1 48352 double
k 1x1 8 double
locations 1x15 120 double
mois 6044x1 48352 double
num 495x17 67320 double
p 6044x1 48352 double
pVaR 1x2 16 double
sd 1x1 8 double
sp 6044x1 48352 double
t 1x1 8 double
tps 6044x1 48352 double
width 1x1 8 double
xx 1x100 800 double
>> clear all
>> load ~/Dropbox/ENSEIGNEMENT/MatlabBanqueFinance/mmc2/Datasets/SP50
>> whos
Name Size Bytes Class Attribute
Assets_Returns 595x442 2103920 double
Index_Returns 595x1 4760 double
>> plot(Index_Returns)
>> clf
>> plot(Index_Returns)
>> figure(1)
>> plot(Index_Returns)
>> plot(Assets_Returns(:,1))
>>
Workspace

Calculs

- Même fonctionnement et même ordre de priorité que sur toutes les machines à calculer standard : d'abord l'élévation à une puissance ($\hat{}$), puis la multiplication et la division ($*$ et $/$) et enfin l'addition et la soustraction ($+$ et $-$).
- Pour modifier l'ordre des opérations ou encore pour le mettre en évidence, on utilise des parenthèses.
- Fonctions : `sin, cos, tan, cot, asin, acos, atan, acot, cosh, sinh, tanh, coth, acosh, asinh, atanh, acoth, exp, log, log10, sqrt, abs, sign, mod, rem, round, floor, ceil, fix.`
- Exemples

```
x=3/2+1.9
y=1+3^4/2*5
z=(1+3)^4/2*5
a=abs(sin(x))+log(y)+exp(-z)
```

Tracer de courbes

- Pour tracer la courbe représentative de la fonction `sin` sur l'intervalle $[0, 2\pi]$, il suffit de taper

```
x=0:0.01:2*pi;
plot(x,sin(x))
```

- On a l'impression que la première commande définit un intervalle sur lequel est définie la fonction `sin`, et que la seconde produit une courbe continue.
- Ce n'est pas du tout comme cela que MATLAB interprète ces commandes. Pour le voir, lancer les instructions ci-dessus, en remplaçant 0.01 par 1, et en retirant le point-virgule à la fin de la première instruction (qui empêche l'affichage du résultat). MATLAB ignore les intervalles et les fonctions d'une variable réelle, il ne connait que des listes de valeurs numériques. La commande

```
x=0:1:2*pi
```

  crée une liste (aussi appelée vecteur-ligne) de nombres, 0 1 2 3 4 5 6.

- `sin(x)` désigne la liste sin(0) sin(1) sin(2) sin(3) sin(4) sin(5) sin(6).
- Autrement dit, la fonction `sin` est appliquée à chaque terme de la liste `x`. `plot(x,sin(x))` trace une ligne brisée reliant successivement les points $(0, \sin(0))$, $(1, \sin(1))$,..., $(6, \sin(6))$.

Pièges

- Pourquoi l'instruction

  ```
  x=0:1:2*pi;
  y=x*sin(x)
  ```

  provoque-t-elle une erreur ?
- Parce que, pour MATLAB, les vecteurs-lignes sont des cas particuliers de matrices, que les matrices ont une façon bien particulière de se multiplier entre elles, qui diffère de la multiplication terme à terme.
- Au lieu de `y=x*sin(x)`, il faut écrire `y=x.*sin(x)` .
- De même, au lieu de `y=sin(x)/x`, il faut écrire `y=sin(x)./x`
- et au lieu de `y=2^x`, il faut écrire `y=2.^x`, lorsque `x` est un vecteur-ligne.
- **Cette erreur est très fréquente**.

Les matrices

- Pour MATLAB, tout est matrice. Une *matrice* `A`, c'est un tableau rectangulaire de valeurs numériques, appelés les *éléments* de la matrice, et repérés par leur position : `A(i,j)` est l'élément situé sur la *i*-ème ligne et la *j*-ème colonne. On parle de vecteurs-ligne pour désigner les matrices à une seule ligne, et de vecteurs-colonne pour désigner les matrices à une seule colonne.
- **Vecteurs-lignes à coefficients régulièrement espacés** :
  On a déjà rencontré la syntaxe commode
  `u=debut:pas:fin` ou `u=[debut:pas:fin]` et plus simplement
  `v=debut:fin` ou `v=[debut:fin]` lorsque `pas` vaut 1.
  Exemples : `u=-10:4:2` équivaut à `u=[-10 -6 -2 2]`
  `v=2:5` équivaut à `v=2:1:5` c'est-à-dire à `v=[2 3 4 5]`
  `w=5:2` équivaut à `w=5:1:2` c'est-à-dire à `w=[]`

- On peut aussi rentrer les coefficients un par un.
  Vecteur-ligne (matrice à 1 seule ligne) : `u=[10 -7 3 8]` ou bien `u=[10,-7,3,8]`
  Vecteur-colonne (matrice à 1 seule colonne) : `v=[2;0;-5]` ou bien `v=[2 0 5]'`
  Tableau à 2 indices : `A=[10 7; 2 4; -5 0]` ou encore, terme à terme

```
A(1,1)=10; A(1,2)=7; A(2,1)=2; A(2,2)=4; A(3,1)=-5;
```

```
A(1,1)=10; A(1,2)=7; A(2,1)=2; A(2,2)=4; A(3,1)=-5;
```

Les matrices, Formes prédéfinies

- Matlab dispose de fonctions définissant des matrices comme `eye, ones, zeros, rand, magic, hilb`...
- Par exemple `eye(3)` est la matrice identité d'ordre 3,
- `zeros(2,4)` est la matrice nulle à deux lignes et 4 colonnes,
- `ones(1,5)` est le vecteur ligne `[1 1 1 1 1]`.
- Attendu que si `A` est un tableau à deux indices, l'instruction `[nl,nc]=size(A)` affecte à la variable `nl` (resp. `nc`) le nombre de lignes (resp. de colonnes) de `A`,
- l'instruction `B=rand(size(A))` crée une matrice `B` de même dimensions que `A` et à coefficients *aléatoires* dans $[0, 1]$.
- Par concaténation de matrices de tailles compatibles :
  Les instructions `u=[1 7 9]` puis `u=[u 3]` équivalent à `u=[1 7 9 3]`.
  L'instruction `M=[ 5*ones(2,3) -eye(2); 1:3 zeros(1,2) ]` crée la matrice

$$M = \begin{pmatrix} 5 & 5 & 5 & -1 & 0 \\ 5 & 5 & 5 & 0 & -1 \\ 1 & 2 & 3 & 0 & 0 \end{pmatrix}$$

Accès aux éléments d'une matrice

- **Accès à un élément à l'aide des parenthèses et des numéros d'indices** :
  `u(2), A(1,3), v(end)`
- **Accès à une section d'une matrice à l'aide de vecteurs d'indices** :
  si `L` et `C` sont des vecteurs d'entiers (lignes ou colonnes peu importe), `A(L,C)` est la matrice composée des éléments `A`(i,j) avec $i \in$ `L` et $j \in$ `C`.
- Par exemple pour la matrice `M` définie précédemment `M([1 3],[2:4])` est la matrice
$$\begin{pmatrix} 5 & 5 & -1 \\ 2 & 3 & 0 \end{pmatrix}$$
- Si `x` est un vecteur, le sous-vecteur formé des composantes d'indice impair de `x` est `x(1:2:end)` tandis que `x(end:-1:1)` est le vecteur déduit de `x` en renversant l'ordre des composantes.
- On utilise le deux-points comme abréviation de `[1:end]`. Ainsi, `A(:,k)` (resp. `A(k,:)`) désigne la colonne (resp. la ligne) d'indice `k` de la matrice `A`. De même, si `C` est un vecteur d'entiers, alors `A(:,C)` est la matrice `A(1:end,C)`.

Opérations sur les matrices

- `A+B, A*B, A-B` désignent les opérations matricielles au sens mathématique lorsque ces opérations ont un sens, c'est-à-dire quand les dimensions des matrices ou vecteurs `A` et `B` sont compatibles.
- Si A est une matrice carrée alors
  `A^n` avec $n \in \mathbb{N}$ est le produit répété n fois de la matrice A,
  `A^(-n)` avec $n \in \mathbb{N}$ est égale à `inv(A)^n`,
  `A^s` avec $s \notin \mathbb{Z}$ est égale à `V*D^s*inv(V)` où `[V,D]=eig(A)`.
- `A\B` est la solution `X` (éventuellement au sens des moindres carrés) de `AX=B` si les dimensions sont compatibles.
  `B/A` est la solution `X` (éventuellement au sens des moindres carrés) de `XA=B` si les dimensions sont compatibles.
- Si `A` est une matrice carrée de dimension $[n, n]$, inversible et si `B` est un vecteur colonne de dimension $[n, 1]$ alors `A\B` et `inv(A)*B` sont identiques mais le calcul n'est pas effectué de la même façon.
- Si `A` est une matrice, si `b` est un scalaire
  - et si `op` est l'un des opérateurs `+, -, *` alors `A op b` ou `b op A` est la matrice de même dimension que `A` dont les éléments sont les `A`(i, j) `op b` (règle analogue si `A` est un vecteur).
  - alors `A.^b` $\equiv$ `A.^(b*ones(size(A)))`
  - alors `b.^A` $\equiv$ `(b*ones(size(A)).^A`

Fonctions de matrices

- **Fonctions de matrices**
  On peut appliquer les fonctions numériques internes (`sin, sqrt, exp, abs, ...`) énumérées en 2.1 et 2.3 à des vecteurs et des matrices. Elles opèrent alors élément par élément. Les fonctions numériques définies par l'utilisateur (cf. section 5) opèrent de même.
- **Fonctions de manipulations de matrices**
  Matlab dispose d'un grand nombre de fonctions intrinsèques sur les matrices : `sum`, `prod`, `max`, `min`, `det`, `cond`, `inv`, `eig`, `svd`, `lu`, `spy`, `diag`, `triu`, `tril`, `mean`, `std`...

Boucles inconditionnelles

- Leur forme générale est

  `for` $k = val$
  liste d'instructions
  `end`

  où k est une variable et val est un **vecteur-ligne**.

- Exemple si (x_n) est la suite récurrente définie par $x_0 = 1$ et $x_{n+1} = \sin(x_n)$ pour $n \geq 0$, on calcule x_{10} en écrivant

```
x=1;
for n=1:10
   x=sin(x);
end
x
```

- Ces boucles peuvent être imbriquées (calcul des 6 premières lignes du triangle de Pascal)

```
A=zeros(6);
for i=1:6
   A(i,1)=1;
   for j=2:i
      A(i,j)=A(i-1,j-1)+A(i-1,j);
   end
end
```

Boucles conditionnelles

- Elles s'écrivent

  `while` *condition*
  liste d'instructions
  `end`

  où la *condition* s'exprime à l'aide des opérateurs arithmétiques

`<`	(strictement inférieur à)	`<=`	(inférieur ou égal à)
`>`	(strictement supérieur à)	`>=`	(supérieur ou égal à)
`~=`	(différent de)	`==`	(égale)

  et des opérateurs logiques

`&`	et
`|`	ou
`~`	non

  A l'exécution, la *liste d'instructions* est répétée aussi longtemps que la *condition* est satisfaite. Par exemple,

```
n=0;
x=0;
while ( x <= 0.99 ) & ( x >= -0.99 )
   n=n+1;
   x=sin(n);
end;
```

  calcule le plus petit entier positif n tel que $|\sin(n)| > 0.99$ et la valeur de $\sin(n)$. On peut afficher ces deux nombres avec l'instruction `[n,x]` ou `disp([n,x])` .

Branchements conditionnels

- Leur forme générale est

  `if` *condition_1*
  liste_1 d'instructions
  `elseif` *condition_2*
  liste_2 d'instructions
  ⋮
  `else`
  liste d'instructions
  `end`

- Le nombre de blocs `elseif` *condition*, *liste d'instructions* est quelconque (éventuellement nul). De même, la présence d'un bloc `else` *liste d'instructions* à la fin est optionnelle.
- Si a, b, c et d sont des variables ayant reçu des valeurs réelles, le rayon spectral ρ de la matrice $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ peut se calculer comme suit

```
delta= (a-d)^2 + 4*b*c;
if delta <= 0
   rho=sqrt(a*d-b*c)
else
   rho=( abs(a+d) + sqrt(delta) )/2
end
```

Outline

Premiers : prédire la population des États-Unis

- On saisie les données, puis on les trace

```
% Time interval
t = (1900:10:2000)';
% Population
p = [75.995 91.972 105.711 123.203 131.669 ...
   150.697 179.323 203.212 226.505 249.633 281.422]';
% Plot
plot(t,p,'b+');
axis([1900 2020 0 400]);
title('Population of the U.S. 1900-2000');
ylabel('Millions');
```

- D'après ces données, quelle est selon vous la population en 2010 ?

```
p
```

Ajustement d'une tendance polynomiale

- Ajustement d'une tendance polynomiale et prédiction.
- La solution est obtenue par la résolution d'un système linéaire basé sur la matrice de Vandermonde

```
n = length(t);
s = (t-1950)/50;
A = zeros(n);
A(:,end) = 1;
for j = n-1:-1:1
   A(:,j) = s .* A(:,j+1);
end
```

- On peut visualiser la matrice de différentes façons.

```
A
imagesc(A)
colorbar
imagesc(A(1:10,1:10))
surf(A(1:10,1:10))
```

Calcul des coefficients du polynôme

- Les coefficients c du polynôme de degré d sont obtenus par

$$A[:, n-d:n]c = p$$

- Si $d < 10$, il y a plus d'équations que d'inconnues et on doit chercher la solution des moindres carrés, sinon la solution est obtenue en inversant A.
- Dans les deux cas, on obtient la solution par (exemple pour $d = 3$)

```
d = 3
c = A(:,n-d:n)\p
```

- Afin d'apprécier la qualité de l'ajustement, on peut faire un graphique

```
v = (1900:2020)';
x = (v-1950)/50;
w = (2010-1950)/50;
y = polyval(c,x);
z = polyval(c,w);

hold on
plot(v,y,'k-');
plot(2010,z,'ks');
text(2010,z+15,num2str(z));
hold off
```

Degré du polynôme

- On peut comparer la prédiction obtenue par l'ajustement d'un polynôme cubique ou quadratique

```
c = A(:,n-4:n)\p;
y = polyval(c,x);
z = polyval(c,w);

hold on
plot(v,y,'k-');
plot(2010,z,'ks');
text(2010,z-15,num2str(z));
hold off
```

- On peut aussi calculer des erreurs aux moindres carrés

```
tt = (t-1950)/50
c3 = A(:,n-3:n)\p;
y3 = polyval(c3,tt);
err3 = sum((y3-p).^2)
c4 = A(:,n-4:n)\p;
y4 = polyval(c4,tt);
err4 = sum((y4-p).^2)
```

- En réalité, la vraie valeur est entre les deux prédiction (voir `fr.wikipedia.org/wiki/Recensement_des_%C3%89tats-Unis_de_2010`).

Quand le degré du polynôme augmente

- Quand le degré du polynôme augmente les prédictions deviennent erratiques

```
cla
plot(t,p,'bo')
hold on
axis([1900 2020 0 400])
colors = hsv(8);
labels = {'data'};
for d = 1:8
   [Q,R] = qr(A(:,n-d:n));
   R = R(1:d+1,:);
   Q = Q(:,1:d+1);
   c = R\(Q'*p);    % Same as c = A(:,n-d:n)\p;
   y = polyval(c,x);
   z = polyval(c,11);
   plot(v,y,'color',colors(d,:));
   labels{end+1} = ['degree = ' int2str(d)];
end
legend(labels, 'Location', 'NorthWest')
hold off
```

Outline

Quand le degré du polynôme augmente

- Pour terminer la séance, faire les exercices du TP d'introduction.

Outline

Lecture des données

- Charger les données du fichier `SP_1993_2016.tex` en utilisant la fonction `textread`
- Extraire de la variable date, l'année, le mois et le jour de chaque enregistrement. les dates en utilisant la fonction datenum

```
an = fix(Date/10000) ;
mois =
jour =
tps = datenum(an,mois,jour) ;
```

- Tracer la serie temporelle de l'index ajusté à la fermeture. Mettre des légendes d'axes et un titre.
- La fonction `datetick` permet d'afficher les légendes des tirets de l'axe de la date au format date (par exemple "mm/yyyy").

```
datetick('x','mm/yyyy')
```

- On peut utiliser les commandes en ligne ou les boutons de l'utilitaire des figures.
- Attention les données sont lues dans le sens inverse du temps.

```
AdjClose = AdjClose(end:-1:1) ;
tps = tps(end:-1:1) ;
```

Rendement

- La "Value at Risk" (VaR) correspond au montant de pertes qui ne devrait être dépassé qu'avec une probabilité donnée sur un horizon temporel donné. Ici, on va l'estimer à partir de quantiles de niveau α élevé.
- Pour estimer la VaR, on définit une fenêtre de 250 points (qui correspondent à environ un an).
- On commence le calcul en 1995.

```
TestWindowStart      = find(year(DateReturns)==1995,1);
TestWindow           = TestWindowStart : SampleSize;
EstimationWindowSize = 250;
```

Calcul par la méthode de la distribution de Gauss

- Pour la méthode de la distribution normale, supposez que les données du portefeuille sont normalement distribuées.
- Utiliser cette hypothèse pour calculez la VaR en multipliant les quantiles de niveau 0.95 et 0.99 (fonction `norminv`) par l'écart-type des rendements (function `std`).
- Calculer la VaR pout tous les temps de la fenêtre et superposer les 2 courbes obtenues à la série temporelle des rendements.
- En utilisant le menu de la figure, changer les couleurs et l'épaisseur des traits pour que les courbes de VaR soient bien visibles. Ajouter une légende.
- L'inconvénient de cette méthode est que l'hypothèse Gaussienne est une hypothèse forte. On peut avoir tendance à sous-estimer les queues de distribution.
- Tracer pour certaines dates, l'histogramme des données qui servent à estimer la VaR (`hist`). Superposer la densité de la loi de Gauss de même moyenne et même écart-type.
- Dans une autre (sous) fenêtre tracer un quantile-quantile plot (`qqplot`).
- La fonction `sunplot`, permet de découper une fenêtre en sous fenêtres.

Calcul par la *Historical Simulation Method*

- C'est une méthode non paramétrique, on ne fait pas d'hypothèse sur la distribution des données.
- En pratique, on utilise les quantiles empiriques.
- Calculer les VaR correspondant aux quantiles 0.95 et 0.99 puis les tracer comme précédemment. On pourra éventuellement superposer les deux graphiques.
- Commentez les résultats.

Calcul par la *Exponential Weighted Moving Average Method*

- Les deux premières méthodes supposent que tous les rendements passés ont le même poids. La méthode de la moyenne pondérée par poids exponentiels (EWMA) donne des poids non égaux. Les rendements les plus récents ont des poids plus forts parce qu'ils influencent le retour "d'aujourd'hui" plus lourdement que les rendements plus éloignés dans le passé.

$$\hat{\sigma}_t^2 = \frac{1}{c}\sum_{i=1}^{W_E} \lambda^{i-1} y_{t-i}^2$$

avec W_E la longueur de la fenêtre et

$$c = \sum_{i=1}^{W_E} \lambda^{i-1} = \frac{1-\lambda^{W_E}}{1-\lambda} \quad \to \frac{1}{1-\lambda} \text{ as } W_E \to \infty.$$

- Pour simplifier les calculs, on peut supposer une fenêtre de longueur infinie

$$\hat{\sigma}_t^2 \approx (1-\lambda)(y_{t-1}^2 + \sum_{i=2}^{\infty} \lambda^{i-1} y_{t-i}^2) = (1-\lambda) y_{t-1}^2 + \lambda \hat{\sigma}_{t-1}^2$$

- On choisit une valeur standard de décroissance : $\lambda = 0.94$.
- Initialiser la méthode EWMA par une période de chauffe sur les premières dates d'observations.
- Calculer les VaR par cette méthode.

Comparaison et validation (*backtesting*)

- Tracer de nouveau les rendements et les VaR estimées par les 3 méthodes.
- Zoomer sur les périodes de fort changement pour mettre en évidence les réactions de chacune des méthodes ; par exemple autre d'Août 1998.

Outline

Optimisation de portefeuille, approche de Markowitz

- Soit une somme fixe d'argent S à répartir entre plusieurs investissements.
- On considère un modèle simplifié d'investissement annuel où chaque actif est modélisé par son rendement annuel par euro investi X_i.
- Ainsi, le rendement moyen $r_i = E[X_i]$ représente le profit réalisé en investissant un euro sur l'actif i.
- En contrepartie de ce profit, l'investisseur encourt un risque, mesuré par la variance $\sigma_i^2 = \mathrm{Var}(X_i)$.
- Le problème est alors le suivant : comment investir son argent de façon à s'assurer un rendement annuel moyen ρ tout en minimisant le risque encouru ?
- La modélisation de ce problème sous la forme d'un problème d'optimisation sous contrainte a conduit à

$$\begin{array}{ll} \min & x^T A x \\ \text{s.c.} & \begin{cases} e^T x = S \\ r^T x \geq \rho \\ x \geq 0 \end{cases} \end{array}$$

où $x, r \in \mathbb{R}^n$, $A \in M_n(\mathbb{R})$, et $e = (1, 1, \ldots, 1)$.

Optimisation de portefeuille, approche de Markowitz

- La modélisation de ce problème sous la forme d'un problème d'optimisation sous contrainte a conduit à

$$\begin{array}{ll} \min & x^T A x \\ \text{s.c.} & \begin{cases} e^T x = S \\ r^T x \geq \rho \\ x \geq 0 \end{cases} \end{array}$$

où $x, r \in \mathbb{R}^n$, $A \in M_n(\mathbb{R})$, et $e = (1, 1, \ldots, 1)$.

- A quoi correspondent x et A dans ce modèle ?
- Pourquoi peut-on se limiter au cas où $S = 1$?

Approche de Markowitz, analyse graphique

- Considérons un cas simple (simpliste...) avec seulement 2 actifs corrélés (corrélation -0..22) de rendements respectifs 9.1% et 12.1% et dont l'un est plus risqué que l'autre :

$$\sigma_1 = 16.5, \quad \sigma_2 = 15.8$$

- Faire un graphique représentant la surface $x^T \Sigma x$ ainsi que les contraintes avec Σ la matrice de covariance des rendements des deux actifs.
- Trouver graphiquement, la combinaison d'investissement qui conduit à un rendement d'au moins 11% avec un risque minimum.
- Retrouver cette solution en se ramenant à un problème d'optimisation en dimension 1. On pourra aussi tracer l'évolution du rendement moyen et du risque en fonction de la part d'investissement dans le premier actif.
- Faire un graphique représentant la frontière d'efficience ie la courbe d'évolution du rendement en fonction du risque selon les proportions d'investissement dans les 2 produits. Tracer aussi le point correspondant à la solution optimale définie ci-dessus.
- Trouver la solution bivariée en utilisant la fonction `quadprog2`.

Approche de Markowitz, S& P 500

- Considérons maintenant des données réelles.
- Le fichier `SP500.mat` donne les rendements de 442 produits pour 596 dates entre nov. 2004 et avril 2016.
  Source : verb==.
- Estimer la matrice de risque et les rendements moyens puis positionner les différents produits dans une graphique de risque/rendement.
- Utiliser la fonction `quadprog2` pour tracer quelques points de la frontière d'efficience.

Approche de Markowitz, formulation à risque controlé

- On peut formuler le problème d'optimisation *Mean-Variance* de telle sorte qu'on fixe le risque

$$\begin{array}{ll} \max & x^T r \\ \text{s.c.} & \begin{cases} e^T x = 1 \\ x^T \Sigma x = \sigma^2 \\ x \geq 0 \end{cases} \end{array}$$

- Résoudre ce problème en utilisant la fonction `quadprog2` qui permet de d'optimiser un problème non linéaire.
- Calculer la variance du portefeuille obtenu.

Approche de Markowitz, formulation à risque controlé

- *Risk aversion mean-variance formulation*

$$\begin{array}{ll} \max & x^T r - \lambda x^T \Sigma x \\ \text{s.c.} & \begin{cases} e^T x = 1 \\ x \geq 0 \end{cases} \end{array}$$

Dans cette formulation, λ est choisi par l'utilisateur. Plus il est grand, plus on minimise le risque. Comparer les résultats obtenus avec différentes valeurs de λ.

Approche de Markowitz, formulation pénalisée

- La pénalisation en norme L1 permet aussi de limiter le nombre de produits (robust estimation)

$$\begin{array}{ll} \max & x^T r - \lambda x^T \Sigma x - \lambda_1 ||x||_1 \\ \text{s.c.} & \begin{cases} e^T x = 1 \\ x \geq 0 \end{cases} \end{array}$$

- Résoudre ce problème pour les données de S&P 500. Analyser les solutions quand on fait varier les constantes de pénalisation.