08/11/2017

Projet d'analyse et modèle de données

Courses Hippiques à Hong Kong

Jean-Baptiste Brun, Charles Cagnon, Pierre Martin
ENSEIGNANTS: MYRIAM MAUMY-BERTRAND & EMMANUELLE CLAEYS

Table des matières

Présentation du dataset

Data set

Ce projet porte sur un data set concernant les résultats de 1561 courses hippiques de Hong-Kong entre 2014 et 2017.
Le data set est disponible sur Kaggle avec l'url suivant :

https://www.kaggle.com/lantanacamara/hong-kong-horse-racing

Courses hippiques

Voici les règles décrivant une course hippique.

Règles

- Le but des courses hippiques est d'arriver premier.
- Chaque participant est associé au départ aléatoirement à une boxe (Draw), correspondant à son couloir de départ.
- Les courses sont entre 900 et 4000m, dépendant du circuit de l'hippodrome
- Les courses s'effectuent avec 16 participants maximum.

Les épreuves

On distingue plusieurs types d'épreuves dans les courses hippiques.

Les épreuves de trots

Le trot attelé

Le jockey est assis sur un char appelé sulky. Le cheval doit alors, tout en trottant, atteindre la ligne d'arrivée le plus rapidement et ainsi arriver premier. Par contre, sous peine de disqualification, il ne doit pas se mettre au galop.

Le trot monté

Le *jockey* est assis normalement sur le cheval sellé. Le rythme imposé est le trop, le galop entrainant une disqualification.

Les épreuves de galop

Le plat

Cette course consiste à faire galoper son cheval le plus vite possible. C'est l'équivalent du 100 mètres (Nous n'avons pas trouvé de cheval ayant pour nom Usain Bolt dans les données, mais il y a Thunder Bolt).

Règles Spéciales

Réservé à une clientèle étrangère, l'hippodrome de Hong Kong suit certaines règles particulières concernant les courses hippiques.

- Il existe un système d'handicap pour rendre les courses plus compétitives. Pour cela, des poids sont ajoutés sur les chevaux en fonction des résultats précédents. Les meilleurs jockeys ayant le poids le plus élevé.
- La liste de chevaux en compétition est faible comparé aux courses d'autres pays, apportant une rivalité accrue à chaque course
- De même, il existe un nombre limité de jockey et entraineur

Prise en main du dataset

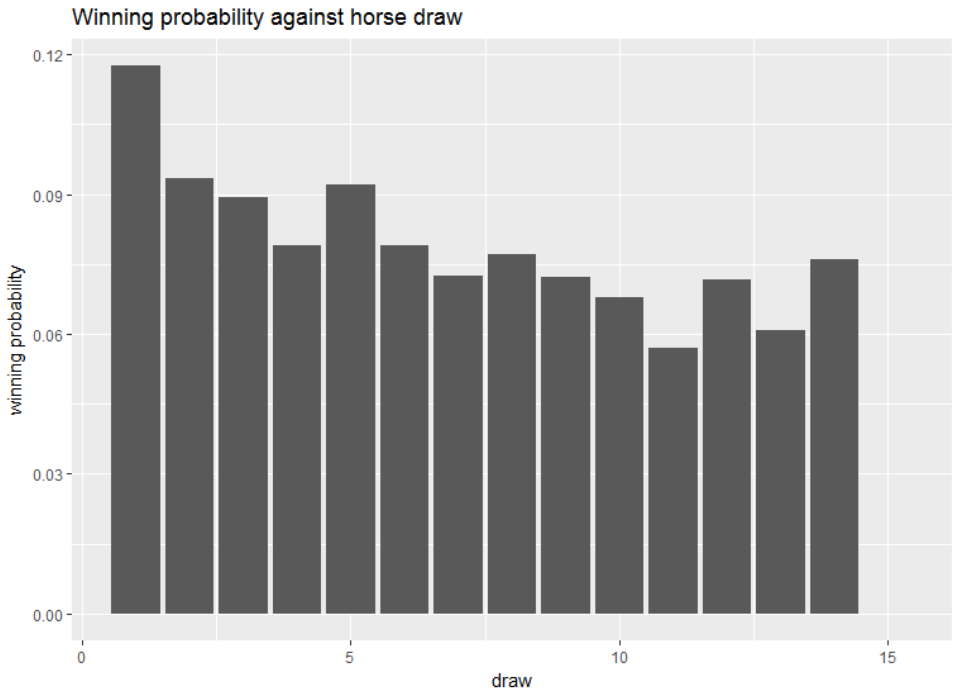
Les données

Le data set est composé de 30189 observations sur 30 variables.

Nom de la variable	Type	Description
Src	Qualitative nominales	Lien url vers la page de la course
Race_date	Qualitative nominales	Date de la course
Race_course	Qualitative nominales	Nom de l'hippodrome
Race_Number	Quantitative discrètes	Numéro de la course (se réinitialise chaque jour)
Race_id	Quantitative discrètes	Identifiant de la course
Race_classe	Qualitative nominales	Classe de la course allant de 1 à 5. La classe 1 correspond aux meilleurs jockeys.
Race_distance	Quantitative continues	Distance de la course
Track_condition	Qualitative nominales	Condition météorologique
Race_name	Qualitative nominales	Nom de la course
Track	Qualitative nominales	Type de circuit
Sectional_time	Quantitative continues	Temps à chaque section de la course
Incident_report	Qualitative nominales	Description des évènements de la course
Finishing_position	Qualitative ordinal	Position finale allant de 1 à 14. Les égalités sont définies dans le dataset par des Die Heat (Ex : 1er ex-aequo = 1 DH). Il est composé aussi de l'ensemble des manières d'être disqualifié.
Horse_number	Quantitative discrètes	Numéro du cheval
Horse_name	Qualitative nominales	Nom du cheval (unique)
Horse_id	Qualitative nominales	Identifiant du cheval
Jockey	Qualitative nominales	Nom du Jockey
Trainer	Qualitative nominales	Nom de l'entraineur
Actual_weight	Quantitative continues	Malus de poids dû au handicap
Declared_horse_Weight	Quantitative continues	Poids du cheval
Draw	Qualitative ordinal	Box de départ
Length_behind_winner	Quantitative discrètes	Distance estimée en taille de cheval derrière le vainqueur
Running_position 1	Qualitative ordinal	Position 1er point de contrôle
Running_position 2	Qualitative ordinal	Position 2ème point de contrôle
Running_position 3	Qualitative ordinal	Position 3ème point de contrôle
Running_position 4	Qualitative ordinal	Position 4ème point de contrôle
Running_position 5	Qualitative ordinal	Position 5ème point de contrôle
Running_position 6	Qualitative ordinal	Position 6ème point de contrôle
Finish_time	Quantitative continues	Temps.
Win_odds	Quantitative discrètes	Côte

Les draws

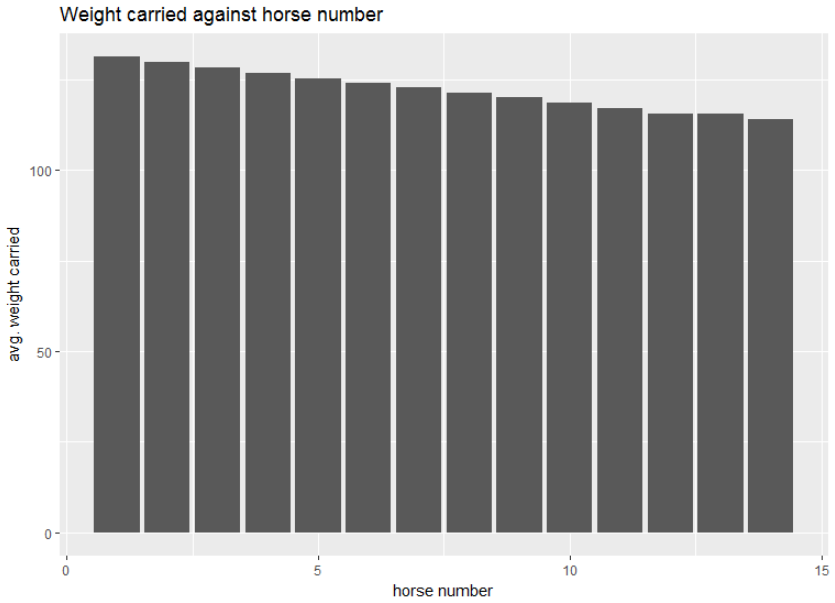
Il faut savoir que le couloir de départ peut donner un avantage ou un désavantage aux différents participants. En effet, il est connu que les chances de gagner sont plus importantes sur les couloirs permettant aux chevaux de prendre une meilleure trajectoire dans les virages.



Ainsi comme on peut voir sur le graphe, les draws dont le numéro est plus petit permettent de meilleures probabilités de victoires. Ce sont ceux placés à vers l'intérieur du circuit.

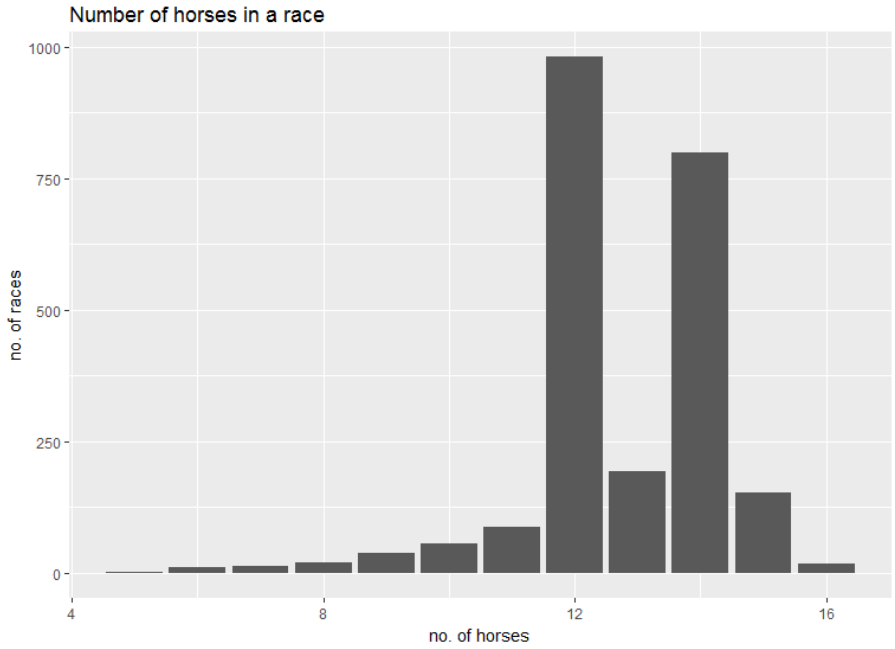
Handicap

Les meilleurs participants subissent un handicap avec l'ajout de poids pour les ralentir et ainsi donner une chance égale à tous. Par convention, les meilleurs chevaux prennent les numéros les plus faibles. Par déduction, les chevaux les plus chargés sont donc ceux aux numéros le plus faible.



Nombre de chevaux par course

A l'hippodrome de Hong Kong, les courses peuvent aller jusqu'à 16 participants. Mais les courses les plus représentées sont à 12 et 14 participants.



Nettoyage des données

Le nettoyage de données est une partie essentielle de l'analyse de données. Cela se décompose en 3 phases :

- Analyser les données afin de détecter les potentiels problèmes
- Choisir le type de transformations à effectuer
- Appliquer ces informations aux données

Tout d'abord, nous avons dû regrouper les deux sets de données que nous avions (un contenant les données sur les chevaux, l'autre sur la course en général). Nous les avons liés grâce à l'ID de la course.

Nous avons vérifié s'il y avait de potentielles duplication mais, heureusement pour nous, il n'y en avait pas.

Les erreurs sémantiques

Nous avons tout d'abord regroupé les plusieurs types de disqualification en un seul car nous avons estimé que faire la distinction entre les différents types de disqualifications était peu utile dans notre recherche.

Ensuite, nous avons décidé de transformer les temps en milliseconde (au lieu du format minutes : secondes : millisecondes).

Fonction transformant les temps en millisecondes

```
timeToMilli <- function(v){
  sapply(strsplit(v, ":", T), function(x) sum(as.numeric(x) * c( 60, 1)[seq(x)]*100))
}
```

Les informations manquantes

Nous nous sommes aperçut que nous avions un certain nombre de données manquantes, notamment à cause du fait qu'un cheval disqualifié n'a pas toutes ses données renseignées (ex : temps final). Une question s'est donc posée : Comment déterminer ces données ?

Nous avions deux choix : soit supprimer ces données, soit les compléter. Nous nous sommes rendu compte qu'il y en avait beaucoup et que celles-ci pouvaient réellement biaiser nos résultats. En effet, si un cheval a fait deux courses, et que sur ces 2 courses il a fini une fois premier et une fois disqualifié, enlever la course où celui-ci est disqualifié reviendrait à dire que ce cheval a gagné toutes ses courses, ce qui est vraiment loin de la réalité. Nous avons donc décidé de compléter ces données.

Le temps d'arrivée

Pour remplacer les données manquantes, nous avons remplacé le temps d'arrivée par le temps que le cheval avait en moyenne dans ce type de courses (1000m, 2000m, ...). Il était important de séparer les types de course car le temps moyen d'une course de petite distance est forcément plus petit que celui d'une course de grande distance.

Pour les chevaux qui n'avaient pas d'autres courses dans ce type de course (donc impossibilité de faire une moyenne), nous avons déterminé que le biais que produisait leur suppression était négligeable (68 sur plus de 30000 individus)

Fonction complétant le temps d'arrivée

```
for (i in 1:length(finish_time$finish_time)){
  #SI LE TEMPS DU CHEVAL EST MANQUANT
  if (is.na(finish_time$finish_time[i])){
    #SI IL N'Y A PAS D'AUTRE TEMPS AVEC CE CHEVAL ON SUPPRIME
    if(finish_time$count[i]<2 || is.na(finish_time$mean_time[i])){
      supr=supr+1
      k=k+1
      list_supr[[k]]<-i
    }
    else{
      #SINON ON MET LE TEMPS MOYEN DU CHEVAL SUR CE TYPE DE COURSES
      finish_time$finish_time[i]<- finish_time$mean_time[i]
    }
  }
}
```

Le couloir

Nous avons trouvé l'idée de compléter les données manquantes de cette section avec des couloirs qui n'étaient attribuées avec aucun cheval. Ainsi, s'il y avait seulement un cheval qui n'avait pas de couloir renseigné, celui se faisait attribuer le couloir qui n'était pas affecté (donc forcément le bon). S'il y avait plusieurs couloirs non assignés, nous les donnions de manière aléatoire aux chevaux qui n'avaient pas de couloirs affectés.

Fonction attribuant les couloirs aux chevaux

```
for(i in levels(draw.modification$race_id)){
  race <- subset.data.frame(draw.modification, race_id==i)
  #SI IL MANQUE UN COULOIR
  if (any(is.na(race$draw))){
    normal.list<-1:nrow(race)
    #ON CHARGE LA LISTE DES COULOIRS MANQUANTS
    missing.values<-setdiff(normal.list,race$draw)
    n<-length(missing.values)
    for(j in rownames(race[which(is.na(race$draw)),])){
      #ON COMPLETE AVEC LES COULOIRS QU'IL MANQUE A ATTRIBUER
      draw.modification[j,"draw"]<-missing.values[n]
      n=n-1
    }
  }
}
```

Mesure entre le cheval et le cheval gagnant

Dans notre set de données, les mesures faites entre le cheval étudié et le cheval gagnant sont en tailles de cheval. Cette mesure peut être exprimée en taille de tête, de nez, en cou, de corps, etc. Nous avions donc des erreurs d'irrégularité. Nous avons dû transformer chaque valeur afin que celles-ci soient uniformes.

Fonction uniformisant les mesures entre le cheval et le cheval gagnant

```
#HEAD
}else if(j=="HD"){
  length.modification$length_behind_winner[i]<-0.1
#NOSE
}else if( j=="NSE" || j=="+NSE"){
  length.modification$length_behind_winner[i]<-0.02
#NECK
}else if(j=="N"){
  length.modification$length_behind_winner[i]<-0.25
#SHORT HEAD
}else if(j=="SH" || j=="+SH"){
  length.modification$length_behind_winner[i]<-0.05
}else{
  lengths<-strsplit(length.modification$length_behind_winner[i],"-")[[1]]
  length.modification$length_behind_winner[i]<-as.numeric(lengths[1])
  if (length(lengths)>1){
    if(lengths[2]=="1/2"){
      length.modification$length_behind_winner[i]<-as.numeric(length.modification$length_behind_winner[i]) + 0.5
    }else if(lengths[2]=="3/4"){
      length.modification$length_behind_winner[i]<-as.numeric(length.modification$length_behind_winner[i]) + 0.75
    }else if(lengths[2]=="1/4"){
      length.modification$length_behind_winner[i]<-as.numeric(length.modification$length_behind_winner[i]) + 0.25
    }
  }
```

Les Poids

Pour les poids manquants, que ce soit le poids du cheval ou porté par celui-ci, nous avons utilisés la même technique que celle pour les données du temps d'arrivée. Nous avons donc remplacé chaque donnée manquante par le poids moyen du cheval.

Fonction complétant les poids des chevaux

```
mean.weigt <- weigth.modification %>%
  group_by(horse_id) %>%
  dplyr::summarize(mean.weigt = mean(declared_horse_weight, na.rm = TRUE))
weigth.modification <- inner_join(mean.weigt, weigth.modification)

mean.actualweigt <- weigth.modification %>%
  group_by(horse_id) %>%
  dplyr::summarize(mean.actualweigt = mean(actual_weight, na.rm = TRUE))
weigth.modification <- inner_join(mean.actualweigt, weigth.modification)


for(i in which(is.na(weigth.modification$declared_horse_weight))){
  weigth.modification$declared_horse_weight[i] <- weigth.modification$mean.weigt[i]
}

for(i in which(is.na(weigth.modification$actual_weight))){
  weigth.modification$actual_weight[i] <- weigth.modification$mean.actualweigt[i]
}
```

Position finale

Pour cette variable nous avions plusieurs choix pour compléter les données manquantes. Ici, les données manquantes n'étaient que pour des chevaux disqualifiés, et donc qui n'avaient pas de position finale. Nous pouvions soit mettre comme valeur 0, soit nous complétions avec la dernière position possible sur cette course (pour une course de 14 chevaux nous pourrions mettre un cheval disqualifié arrivé 14ème). Nous avons choisi de donner une valeur nulle aux chevaux disqualifiés lors d'une course. Pour affiner nos observations, le mieux serait de faire une seconde analyse en prenant la deuxième solution, c'est-à-dire de compléter avec la dernière position possible sur cette course.

Analyse exploratoire

ACP : analyse en composantes principales

Il s'agit d'une méthode de la famille de l'analyse des données, dite multivariées. Elle permet de parcourir et d'observer les *datasets multidimensionnels* composé de variable quantitatives continue. Largement utilisé, cette méthode est l'une des plus connue.

Objectif de l'ACP

L'objectif est de transformer p variables quantitatives liées entre elles (corrélées), en nouvelle variables décorrélées les unes des avec k, $K < p$. On parle alors de variables en ***composantes principales***.

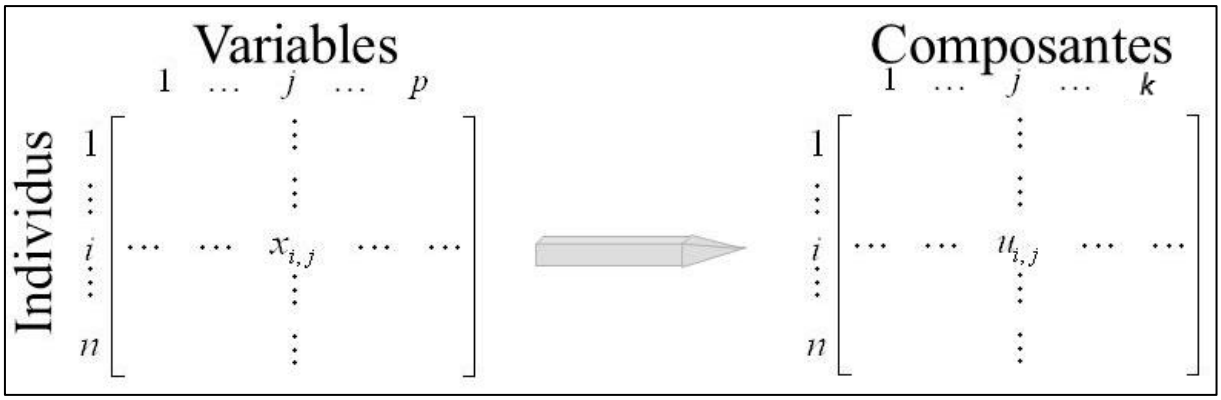


Fig. 1.1 : Composantes principales

Les individus sont représentés dans un espace à p ou n dimensions pour qu'un maximum d'information soit conservée. Ainsi si l'information associé aux 2 ou 3 premier axes représente un pourcentage suffisant de la variable totale du nuage de point (l'inertie ici), il sera possible de représenter les observations **sur un graphique** à 2 ou 3 dimensions. L'interprétation est alors simplifiée.

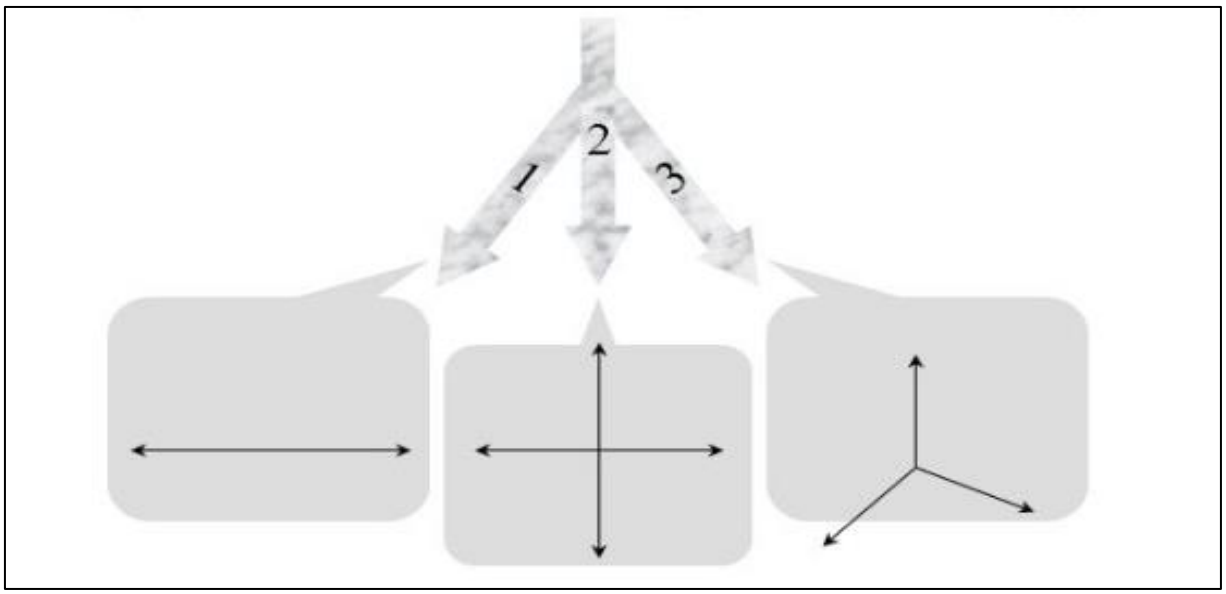


Fig. 1.2 : Représentation en 1,2 et 3 dimension.

1.2 Analyse préliminaire : la corrélation

On cherche à determiner si il y a une forte correlation entre deux ou plus variable, pour detecter la présence ou non d'une rekation linéaire entre deux caractères quantitatif.

```
> tableau_quanti_cor
                     race_number race_distance finishing_position actual_weight declared_horse_weight
race_number               1.0000        0.0628            0.0233        0.0122                0.0104
race_distance             0.0628        1.0000            0.0044        0.0117                0.0136
finishing_position        0.0233        0.0044            1.0000       -0.0086                0.0003
actual_weight             0.0122        0.0117           -0.0086        1.0000                0.0803
declared_horse_weight     0.0104        0.0136            0.0003        0.0803                1.0000
draw                      0.0289        0.0012            0.0762       -0.0103               -0.0015
length_behind_winner      0.0090       -0.0145            0.0001       -0.0198               -0.0014
finish_time               0.0102       -0.0201           -0.0056        0.0198               -0.0413
                        draw length_behind_winner finish_time
race_number           0.0289               0.0090      0.0102
race_distance         0.0012              -0.0145     -0.0201
finishing_position    0.0762               0.0001     -0.0056
actual_weight        -0.0103              -0.0198      0.0198
declared_horse_weight -0.0015              -0.0014     -0.0413
draw                  1.0000              -0.0068      0.0009
length_behind_winner -0.0068               1.0000      0.0270
finish_time           0.0009               0.0270      1.0000
```

Capt. 1.1 : Représentation du tableau de corrélation.

Observation, les corrélations les plus forte sont entre :

- Finish position & draw = 0.0762 : vue dans la partie explication des données, en fonction du couloir (draw) la course est plus ou moins courte. De plus le couloir semble influé sur la finish position à hauteur de 7.6%

- Race distance & race number = 0.0628 : vue dans la partie explication des données, les distance de course sont …… même numéro la journée.
- Declared horse weight & actual weight = 0.080 : analyse préalable explique cette corrélation.

Valeur étonnante :

- Finish time & declared horse weight = 0.0413
- Finish position & declared horse weight = 0.0003

D'instinct la logique voudrait, que Finish time & Finish position soient très corrélés.

Explication/hypothèse : le poids d'un cheval n'influe pas sur le résultat de la course ou très peu 0.03% mais sur son temps de parcours/finish time. Cela implique que Finish time & Finish position sont décorrélés.

Preuve :

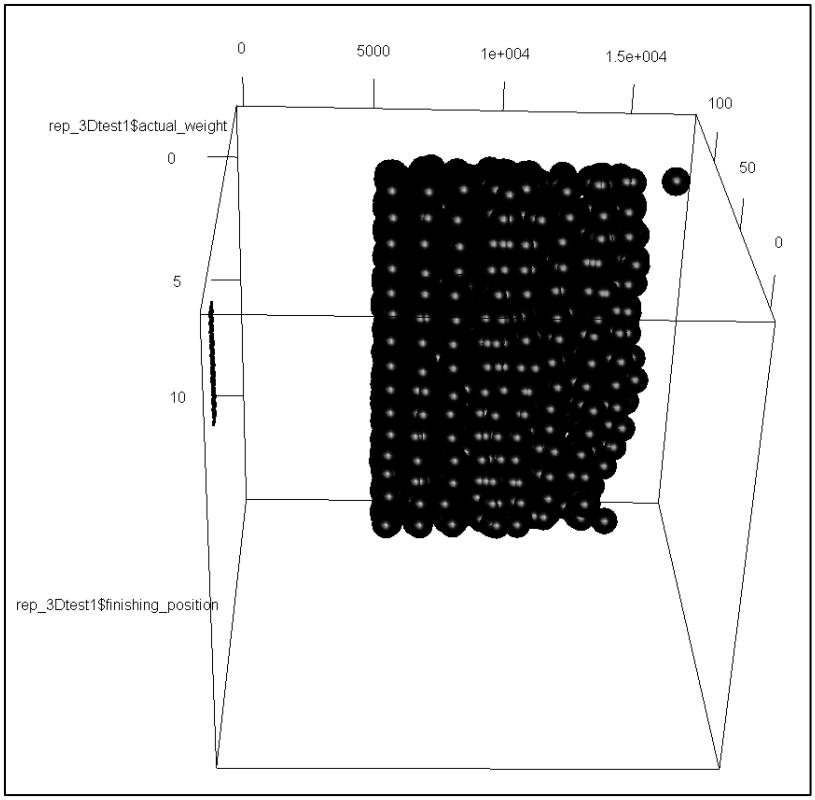
- Finish time & Finish position = 0.0056.

Conclusion : l'étude de la corrélation n'a pas apporté de résultant très concluant, seul 3 valeurs sont supérieures à 5% or ces corrélations s'expliquent par la nature de leurs variables. Il n'y a donc pas relation de linéarité forte ici.

Nous passons donc à l'étude de l'ACP.

1.3 Représentation en 3D normée et non normée.

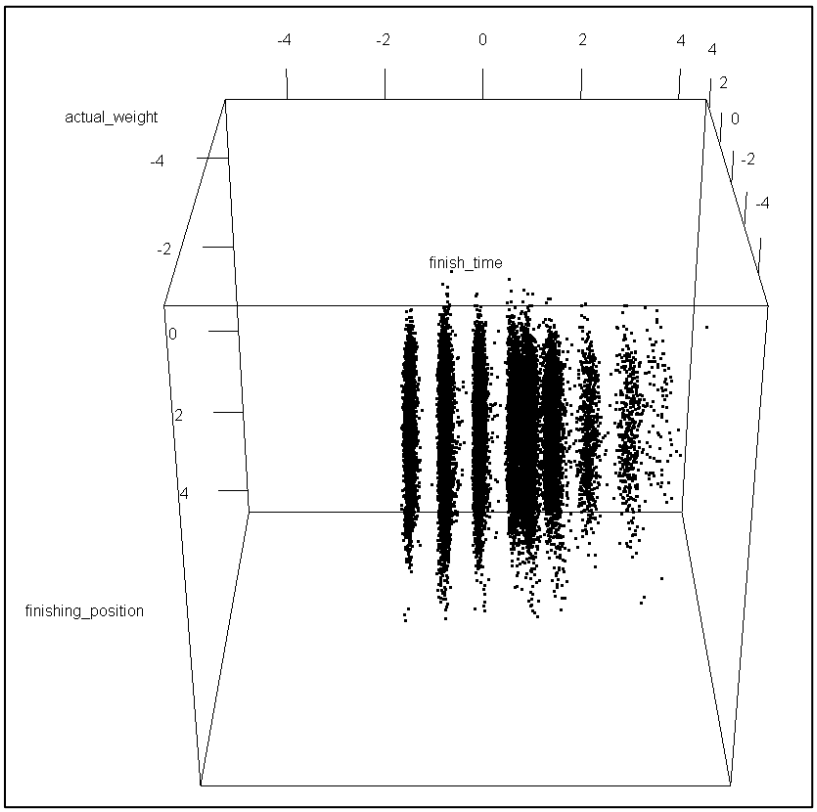
Test 1 : On fait les choix de représenter 3 variables : actual weight, Finish time & Finish position.



Représentation 3D de actual weight, Finish time & Finish position.
Cette représentation est non normée.

Le résultat obtenu n'est pas exploitable.

Capt 1.3 : représentation 3D non normée.

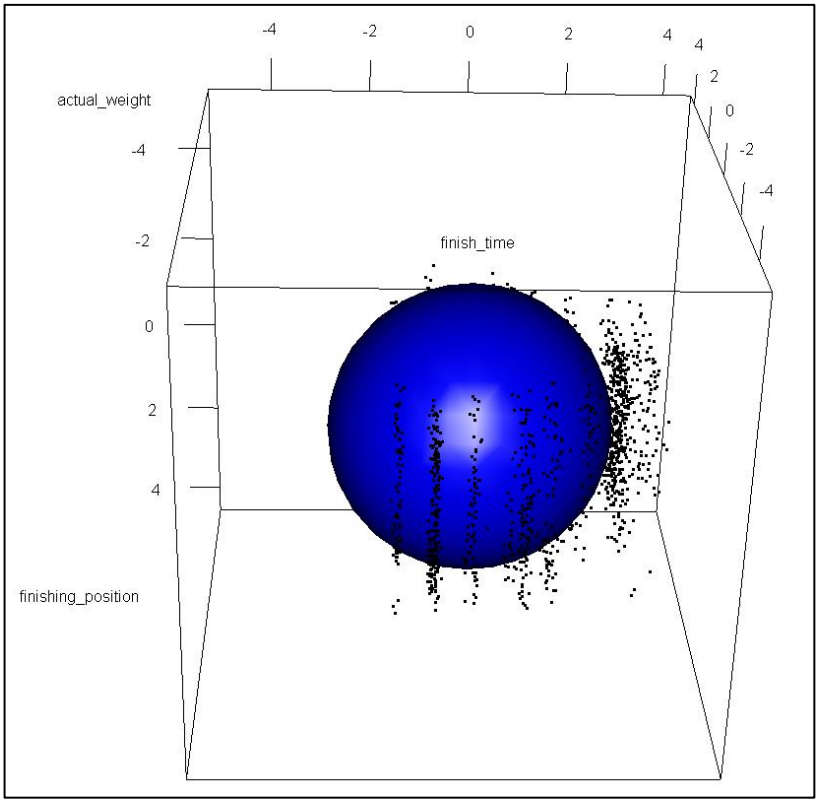


Représentation 3D de actual weight, Finish time & Finish position.
Cette représentation est centrée réduite. Le nuage de point a une forme « cubique ».

Interprétation : ces « couches » correspond au distance 1 000 à 2 400, au nombre de 9.

Capt 1.3 : représentation 3D centré réduite.

On ajoute

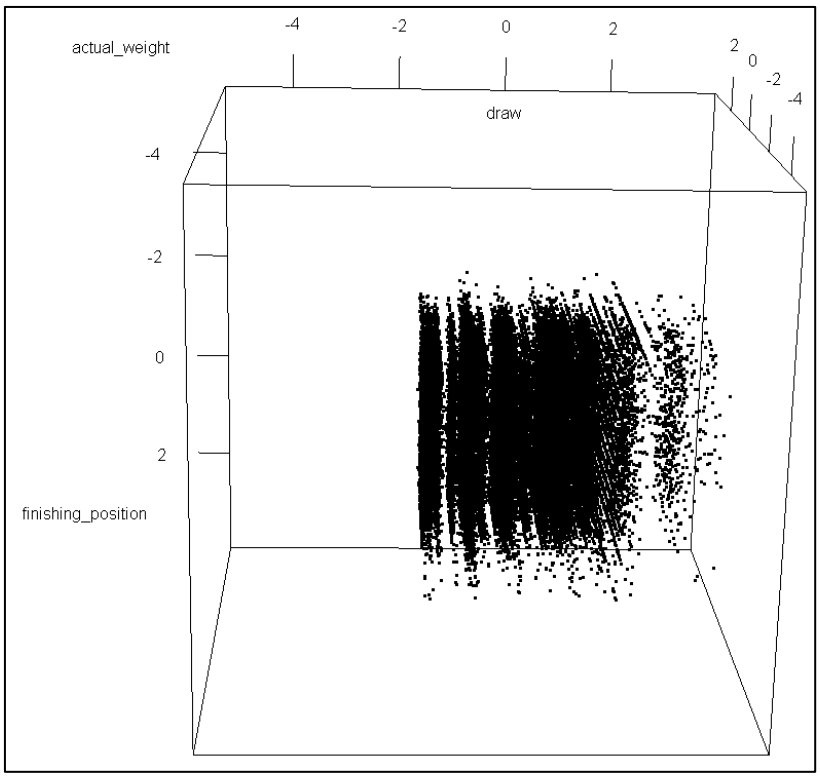


L'ellipse prend une forme particulière celle d'une sphère parfaite (en apparence).

Interprétation : Ceci indique que la réparation est relativement homogène, et s'explique par la forme cubique du nuage de pont.

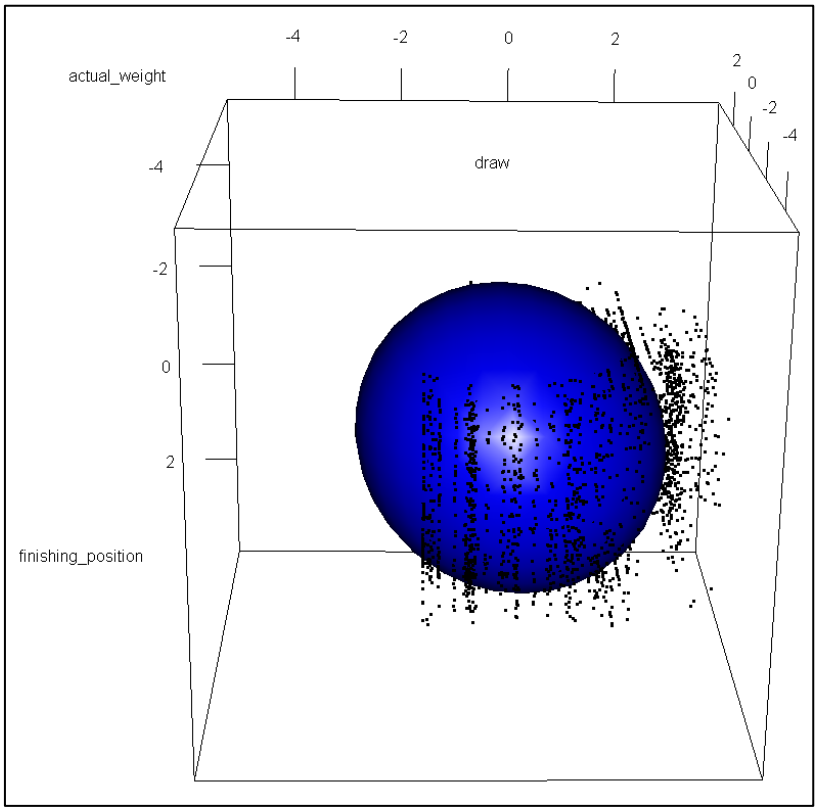
Capt 1.4 : représentation 3D centré réduite avec ellipse.

Test 2 : On étudie l'influence de draw (couloir) sur actual weight & Finish position.



Il y a 16 couche ici qui représente les positionnions (jusqu'à 16 participant).

Capt 1.5 : représentation 3D centré réduite.



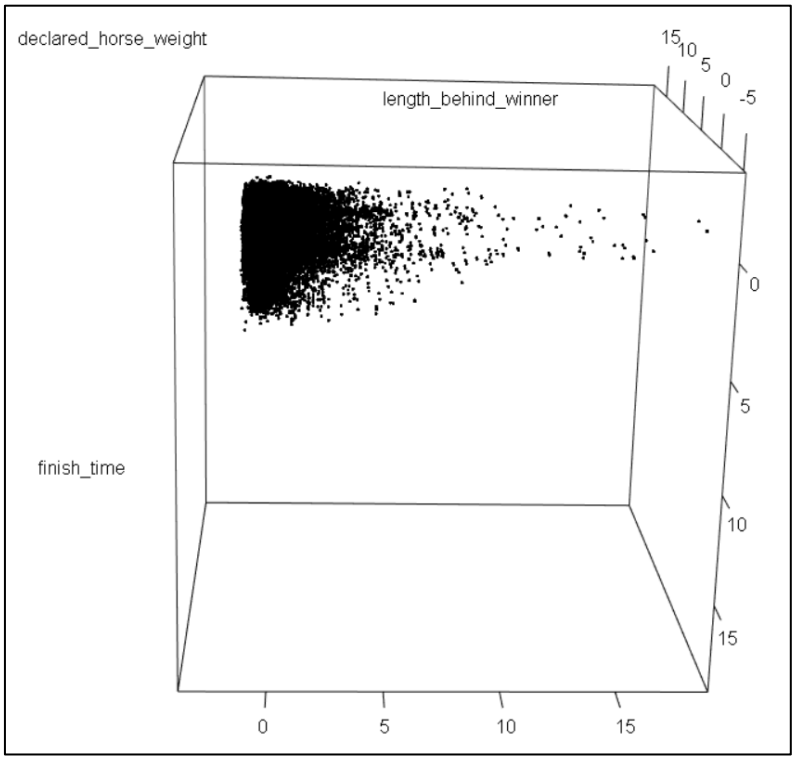
L'ellipse prend une forme particulière celle d'une sphère parfaite comme la représentation précédente.

Capt 1.6 : représentation 3D centré réduite avec ellipse.

Observation : la réparation est plus concentrée, les couches sont moins évidentes & elles ne représentent plus les distances mais les individus. C'est la conséquence de l'intégration de draw.

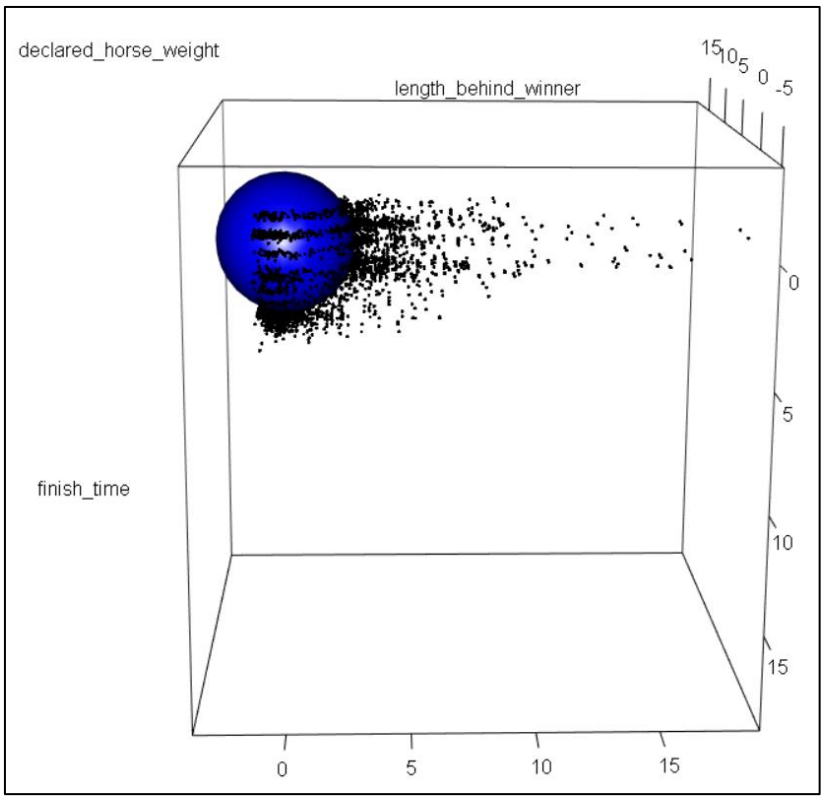
Interprétation : l'explication de ces deux modèles est difficile nous n'avons pas de résultat significatif qui influe sur finish position, qui est notre variable à déterminer.

Test 3 : On fait le choix de ne pas représenter « finish position » qui est la variable que l'on cherche à observer. On représente que des variables qui sont moins sujet à ces effets de couche pour voir si, entre, elles s'influencent.
On cherche aussi à obtenir une représentation moins « cubique », pour étudier le comportement de la répartition. On utilise : declared horse weight, length behind winner & finish time



Capt 1.7 : représentation 3D centré réduite avec ellipse.

L'effet de couche est toujours visible signe d'une variable latente, cependant la répartition est moins cubique.
La forme du nuage est celle d'une « mèche » ici.



Capt 1.8 : représentation 3D centré réduite avec ellipse.

La sphère englobe moins de point que sur les autres représentations.

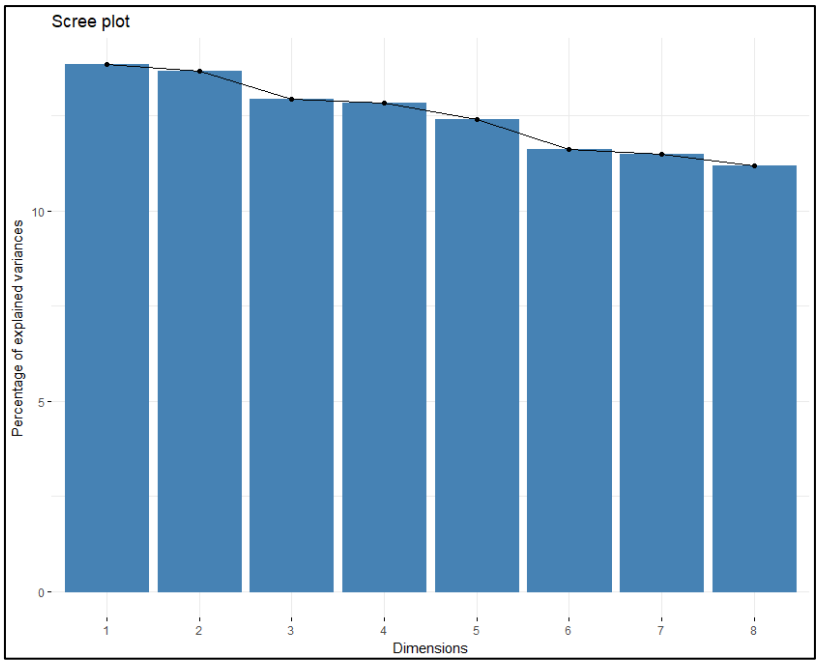
Interpretation : finish time est lié à declared horse weight. La forme du nuage nous renseigne sur l'évolution de lenght behind winner qui peut être très importante d'une course à une autre.

Conclusion : Cette étude montre que les variables influent les unes les autres à des degrés relativement proches. Il serait intéressant d'étudier l'ACP et de regarder l'inertie des vecteurs, pour déterminer les variables latentes ainsi que celle qui contribue le plus.

Pour aller plus loin : une représentation 3D en fonction des classes des chevaux semble plus indiqué. Plus pertinente elle permettrait d'affiner la représentation. En effet entre les classes, les chevaux ont des résultats bien différent (finish time notamment).

1.4 Analyse en Composantes Principales.

1.4.1 Visualisation de l'inertie des axes



Les dimensions sont respectivement les variables du Tableau_quanti.

Observation : l'inertie des dimensions est très proche d'une variable à une autre : 11 à 10.5% d'inerties.

Capt 1.9 : représentation 3D centré réduite avec ellipse.

```
> summary(quanti_acp)
Class: pca dudi
Call: dudi.pca(df = Tableau_quanti[, list.acp], center = TRUE, scale = TRUE,
    scannf = FALSE, nf = 8)

Total inertia: 8

Eigenvalues:
    Ax1     Ax2     Ax3     Ax4     Ax5
 1.1084  1.0936  1.0348  1.0270  0.9919

Projected inertia (%):
    Ax1     Ax2     Ax3     Ax4     Ax5
  13.85   13.67   12.94   12.84   12.40

Cumulative projected inertia (%):
    Ax1   Ax1:2   Ax1:3   Ax1:4   Ax1:5
  13.85   27.52   40.46   53.30   65.70
```

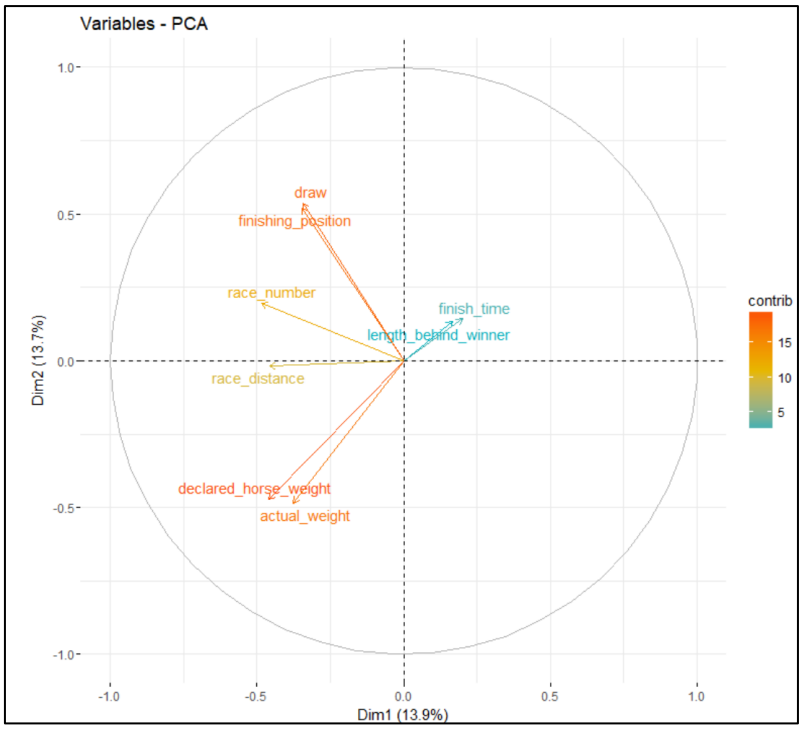
Observation : pour obtenir 80% minimum d'inertie il faut additionner les 7 première valeur.

1.4.2 Cercle des corrélations linéaires

Principe

On étudie donc le s

Dans un permier temps on représents toutes les variables sur notre cercle des corrélations afin d'observer le comportement des variables les unes par rapport aux autres pour ensuite affiner.

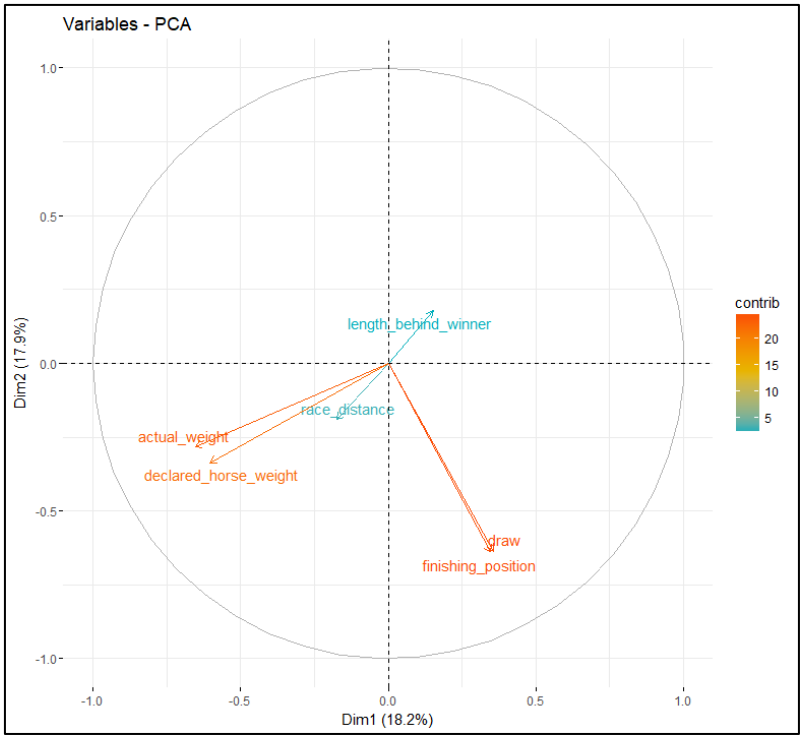


Capt 1.10 : représentation Cercle des corrélations.

Observation : Les axes Dim1 & Dim2 ne représentes additionner que 27% de l'inertie des variables représentées.

Interprétation : certain axes sont peu représentatifs comme finish time & lenght behind winner. Il serait intéressant de retirer des variables pour observer la répartition.

On retire de la représentation Race number & finish time :

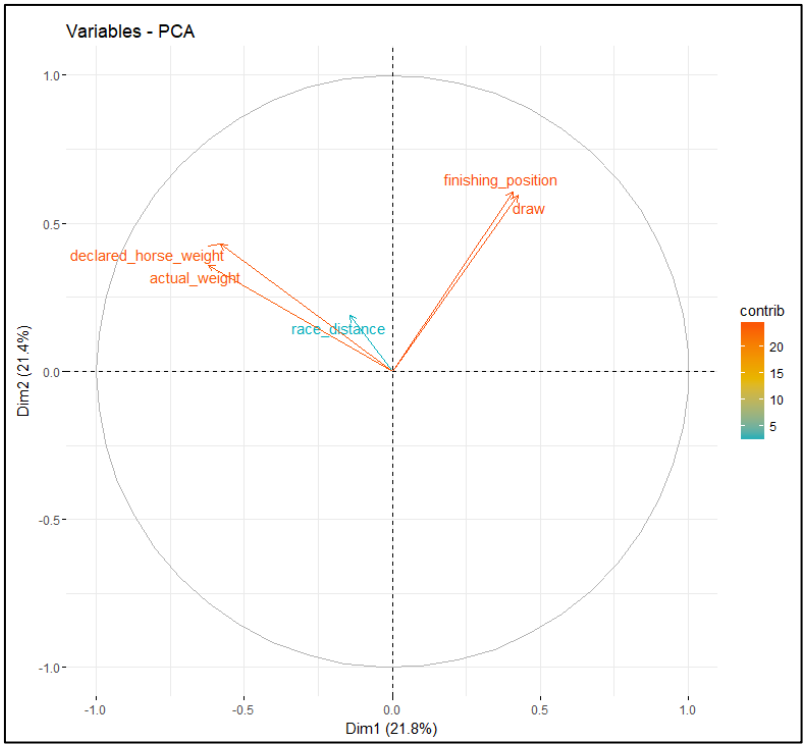


Capt 1.11 : représentation Cercle des corrélations.

Observation : l'inerties des axes augmentes, ils sont plus représentatifs des variables présentes.

Interprétation : une forte corrélation semble exister entre draw & finishinfg position ainsi que actual weight & declared horse weight

On retire maintenant lenght behind winner :

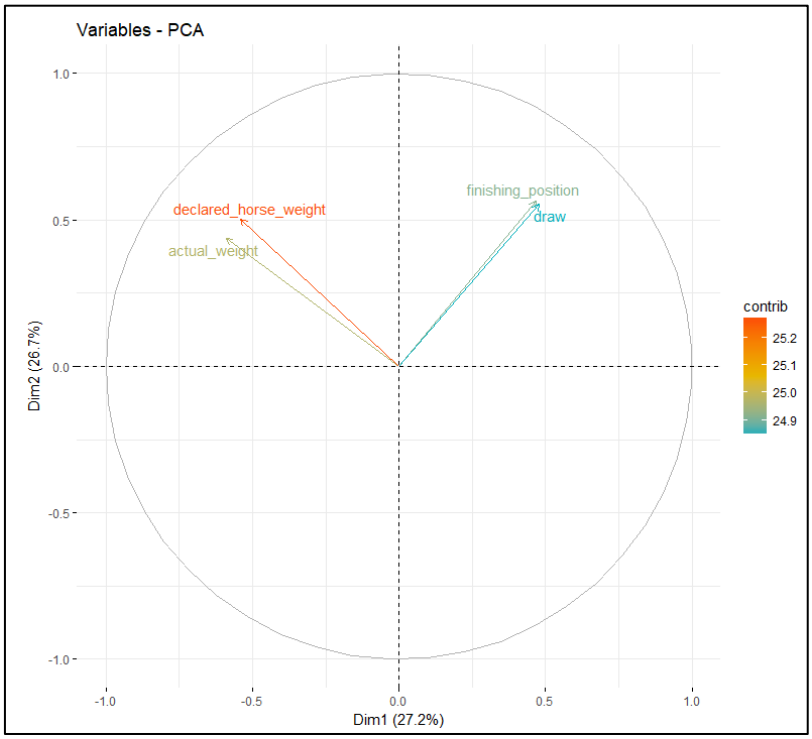


Capt 1.12 : représentation Cercle des corrélations.

Observation : race distance n'est pas représentatif.

Interprétation : race distance tronc les résultant pour ce graphique, il faut donc le retirer.

Cercle de corrélation affiner :



Capt 1.13 : représentation Cercle des corrélations.

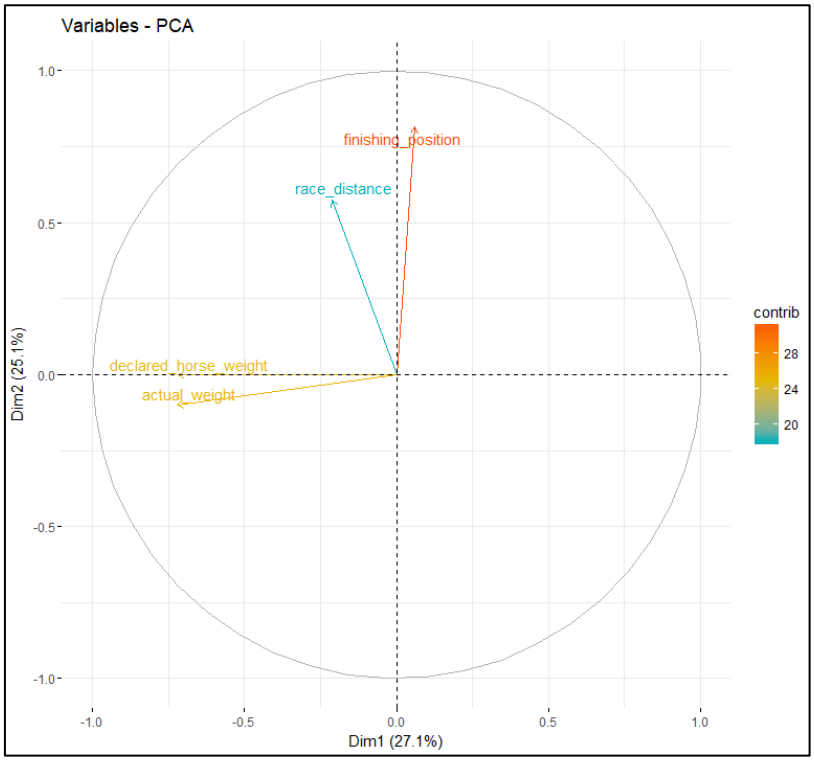
Observation : nous obtenons des résultats très satisfaisant dans notre cas, avec 27% d'inertie pour la Dim1 & 26.6% pour la Dim2

Interprétation : cette représentation peut-être exploiter afin d'extraire quelles valeurs contribues à la finishing position.

Conclusion intermédiaire : La position final lors d'une course dépend fortement du couloir(draw) de départ pour un cheval donné ainsi que du poids (actual weight & declared horse weight) du cheval. Le poids d'un cheval contribue à sa réussite, ceci s'explique par sa masse musculaire plus ou moins importante et sa taille qui est liée à son poids, mais dans notre tableau de donnée nous n'avons pas de variable taille.

Néanmoins lors de notre précédente représentation celle-ci n'était pas propice à la représentation de la race distance en fonction de la finishing position et des autres variables. On se pose donc comme problématique, la distance influe t'elle sur la position finale ?

Représentation affinée : race distance



Capt 1.14 : représentation Cercle des corrélations.

Observation : race distance est peut-être considérer comme représentatif une fois draw retirer.

Interprétation : race distance joue un rôle important sur l'issue de la course.

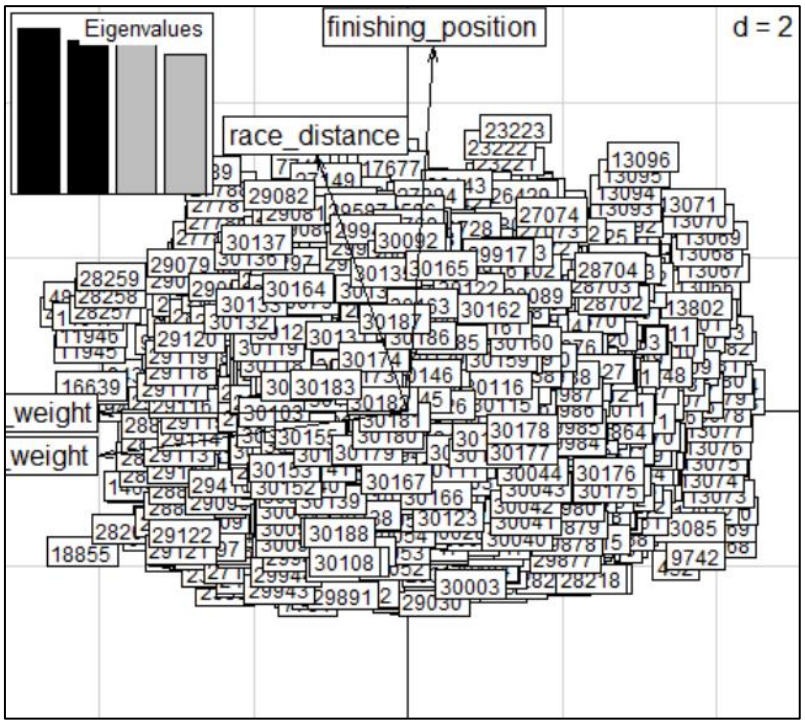
Conclusion intermédiaire : race distance est une des autres variables qui contribue fortement à finishing position. En fonction d'une course donné un cheval ne donne pas de manière général les mêmes résultats.

Précision : On a une représentation similaire pour race number à la place de race distance, ceci s'explique car race distance est une variable latente de race number. *Cf annexe1.1*.

D’un cheval donné

1.4.3 Représentation des individus

Pour des raison pratique de puissance/temps de calcule nous ne représentons que 5% du dataset.



Observation : se dataset n’est pas propice à ce type d’analyse.

Interprétation : en l’état, la forme du tableau mais aussi la quantité d’information importante dans le dataset (nombres d’individus très élevé) n’est pas propice à une étude sur les individus. Il faudrait transformer le tableau pour le rendre praticable.

1.4.5 Conclusion générale de l'ACP

Lors de cette étude nous avons vu que l'inertie de chaque variable était relativement proche les unes des autres. Mais nous avons pu déterminer quelles variables contribuer le plus à la victoire d'un cheval dans une course donnée.

Ainsi nous avons, classé dans l'ordre décroissant :

- Draw
- Poids (actual weight & declared horse weight) contrairement à ce que nous avions pu voir dans l'analyse du tableau de corrélation en début de partie.
- Race distance

Pour aller plus loin : il serait intéressant de construire un second tableau quantitatif dont les lignes seraient constituées des chevaux et non plus du résultat d'un cheval pour une course donnée. Ce nouveau tableau permettrait de représenter et d'étudier la répartition des individus, ce qui est impossible actuellement avec notre tableau.

Analyse factorielle de correspondances

L'analyse factorielle de correspondances est une méthode statistique d'analyse de données.

« L'analyse factorielle traite des tableaux de nombres. Elle remplace un tableau de nombres difficile à analyser par une série de tableaux plus simples qui sont une bonne approximation de celui-ci » Ces tableaux sont « simples », car ils sont exprimables sous forme de graphiques » Rémi Bachelet

Les tests

Pour notre data set, nous avons décidé d'effectuer des tests du Chi-2 entre nos différentes variables qualitatives pour tester l'hypothèse H0 d'indépendance.

Nous avons alors sélectionné nos variables qualitatives pour effectuer les tests avec la fonction chisq .test() :

Remarque : Nous n'avons pas utilisé le test de Shapiro, car il s'agit de variables qualitatives.

```
> chisq.test(race_horses$jockey, race_horses$horse_name)

        Pearson's Chi-squared test

data:  race_horses$jockey and race_horses$horse_name
X-squared = 1199100, df = 226900, p-value < 2.2e-16

> chisq.test(race_horses$draw, race_horses$horse_number)

        Pearson's Chi-squared test

data:  race_horses$draw and race_horses$horse_number
X-squared = 408.86, df = 195, p-value < 2.2e-16

> chisq.test(race_horses$finishing_position, race_horses$trainer)

        Pearson's Chi-squared test

data:  race_horses$finishing_position and race_horses$trainer
X-squared = 2753.9, df = 3290, p-value = 1

> chisq.test(race_horses$finishing_position, race_horses$jockey)

        Pearson's Chi-squared test

data:  race_horses$finishing_position and race_horses$jockey
X-squared = 8816.1, df = 3675, p-value < 2.2e-16

> chisq.test(race_horses$finishing_position, race_horses$horse_name)

        Pearson's Chi-squared test

data:  race_horses$finishing_position and race_horses$horse_name
X-squared = 85617, df = 75635, p-value < 2.2e-16
```

Il en ressort que quasiment tous les tests sont significatifs et rejettent l'hypothèse nulle H0, ayant une p-value inférieure à 0.05. On note l'exception du test entre le trainer et la finishing_position avec une p-value de 1.

Nous pouvons alors faire une AFC entre toutes ces variables. Suite à nos recherches sur les courses de chevaux, les experts se rassemblent sur le fait que les chevaux et les jockeys sont des variables influençant énormément sur la position finale (finishing_position) d'une course.

Nous avons alors décidé de vérifier ces observations en effectuant des AFC entre finishing_position et jockey et entre finishing_position et horse_name.

Jockey et position

Pour cela, nous avons modifié les données pour obtenir les fréquences entre chaque position et chaque jockey donnant le tableau suivant :

	1	1 DH	2	2 DH	3	3 DH	4	4 DH	5	5 DH	6	6 DH	7	7 DH	8	8 DH	9	9 DH	10	10 DH	11	11 DH	12
---	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
A Atzeni	3	0	3	0	3	0	6	0	8	0	6	0	9	0	3	0	9	0	7	0	8	0	
A Badel	7	0	8	0	8	0	12	0	11	0	6	0	11	0	11	0	10	0	8	0	14	0	
A de Vries	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
A Sanna	2	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	5	0	1	0	
A Suborics	15	0	24	0	21	0	22	0	24	1	34	1	25	0	30	0	31	0	36	0	38	0	3
B Fayd'herbe	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
B Melham	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
B Prebble	127	0	147	1	137	2	132	0	98	2	116	0	91	0	93	0	102	0	92	0	93	0	6
B Shinn	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	1	0	2	0	0	0	
B Vorster	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
C Brown	1	0	1	0	1	0	2	0	2	0	0	0	1	0	0	0	0	0	1	0	2	0	
C F Wong	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
C Hayes	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
C K Tong	4	0	8	0	11	0	15	0	29	1	33	1	26	0	46	0	54	0	72	0	71	0	8
C Murray	4	0	1	0	5	0	12	1	7	0	7	0	19	0	8	0	12	1	7	0	5	0	
C Perkins	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
C Reith	2	0	5	0	9	0	10	0	6	0	10	0	10	0	11	0	10	0	15	1	8	0	1
C Schofield	62	0	82	1	61	1	83	4	69	0	75	0	86	0	71	1	64	0	64	0	58	1	5
C Soumillon	3	0	4	0	4	0	3	0	3	0	1	0	0	0	1	0	3	0	1	0	3	0	
C Williams	9	0	8	0	10	0	3	0	5	0	4	0	4	0	8	0	5	0	7	0	4	0	
C Y Ho	72	0	72	0	94	2	120	2	112	0	127	2	136	0	112	2	154	1	134	0	130	1	11

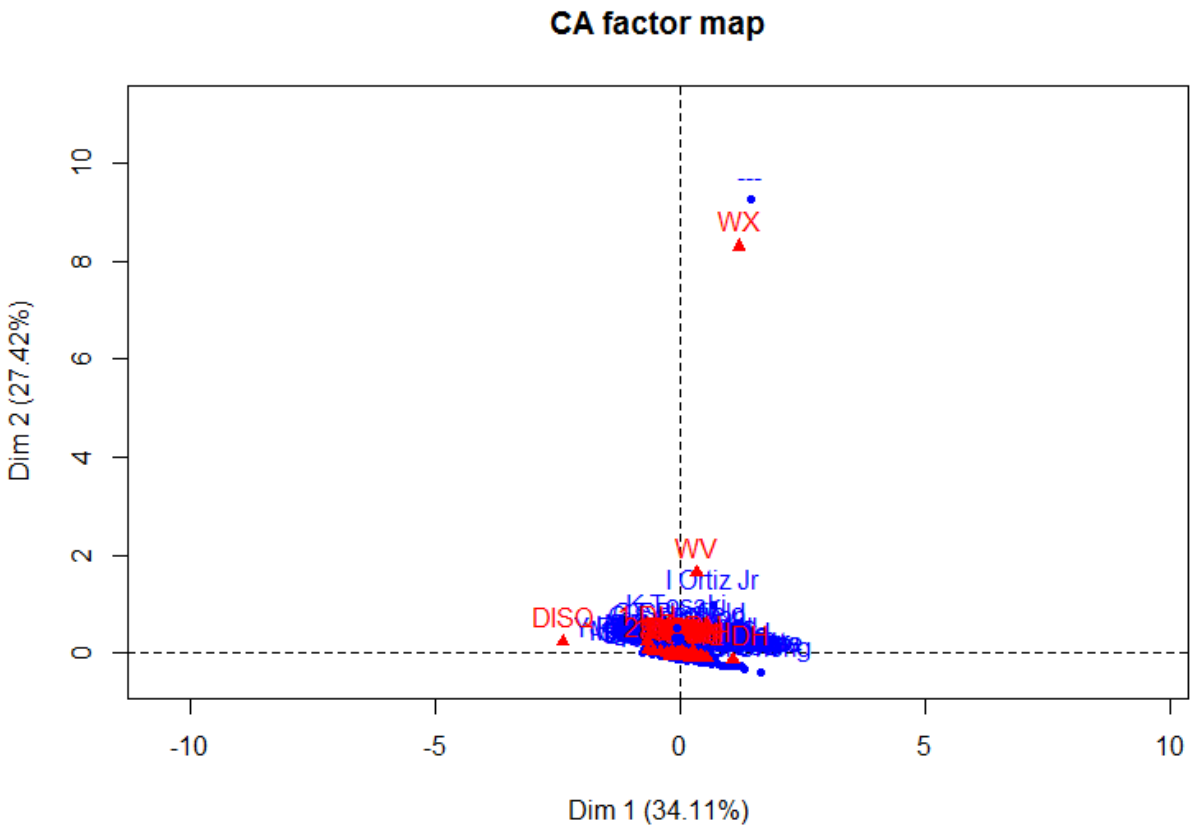
Showing 1 to 23 of 106 entries

Ce tableau est composé de 106 jockeys et des 36 positions différentes (en comptant les disqualifications).

Avec le tableau, nous avons alors les effectifs des resultats de chaque jockey pour chaque position. Cela nous permet maintenant d'éffectuer une AFC.

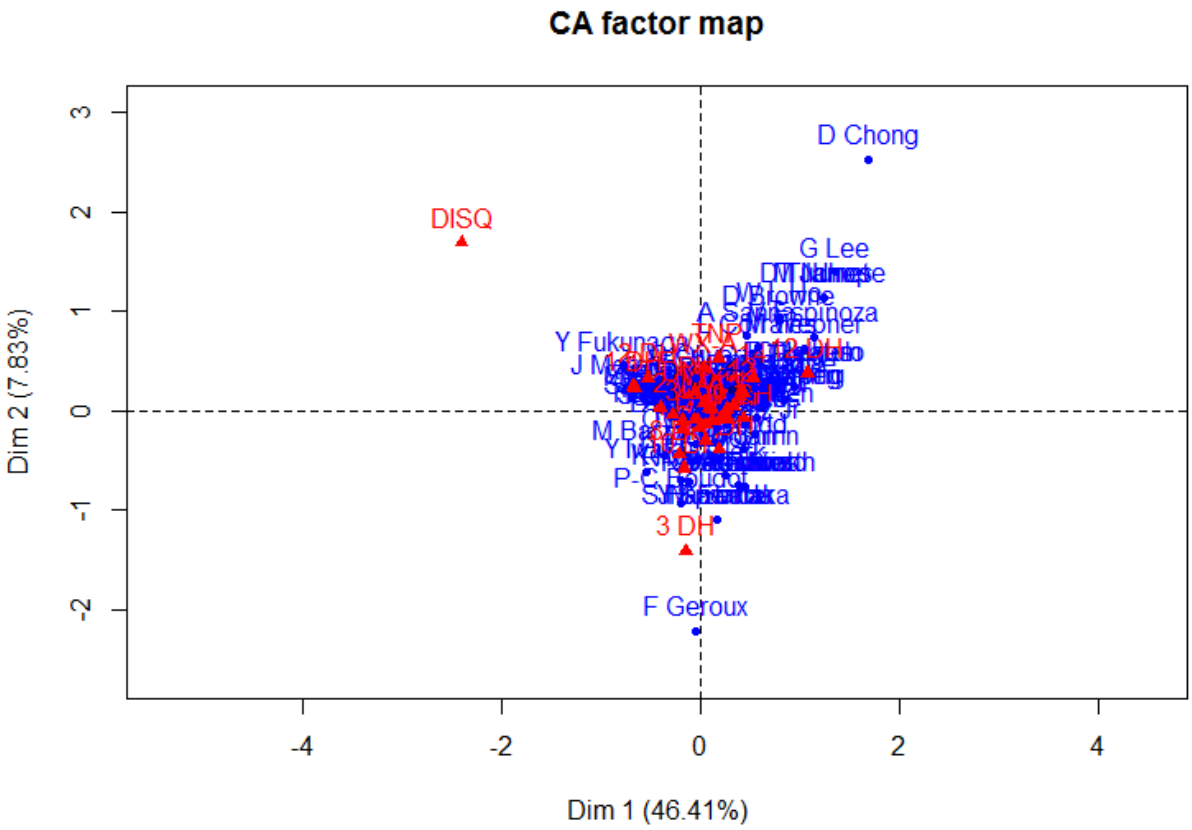
L'afc via la fonction CA() sur le tableau donne alors le résultat suivant :

```
res_exo2 = CA(tableau.freq.jockey)
```



Nous remarquons alors sur le factor map que l'axe 1 représente alors 34% de l'inertie et l'axe 2 représente 27,42% pour un total de 61,53%. En observant le graphe, il est possible de voir que les modalités '---' de jockey et 'WX' de finishing_position sont à part des autres en étant situées en haut du graphe. Après recherche, il se trouve que ces modalités représenteraient les jockeys ayant été disqualifiés ou non présenté à une course. Il s'agit alors d'un cas particulier et avant de continuer l'analyse, il serait préférable de refaire une AFC en mettant ces modalités en supplémentaires.

Voici le résultat obtenu en mettant les modalités précédemment citées en supplémentaire :



Avec cette nouvelle configuration, l'axe 1 augmente alors à 46,41% de l'inertie, alors que l'axe 2 ne représente plus que 7.83% pour un total de 54,24%. Ce qui implique une perte d'information par rapport au cas précèdent. Par contre ce nouveau graphe permet de mieux décrire l'axe 1. Il en ressort que les valeurs à gauche de l'axe représentent les jockeys ayant de bon classement et ceux à droite de mauvais. La modalité DISQ se situant en haut à gauche perturbe cette analyse et pourrait alors être considérée comme une modalité supplémentaire.

Au lieu de refaire une AFC avec une nouvelle modalité supplémentaire, une solution serait de regrouper les positions identiques entre elle. C'est-à-dire fusionner tous les types de disqualification entre une valeur et fusionner les positions et leur ex-aequo (Ex 1 et 1 DH).

Le tableau suivant est le résultat de cette fusion :

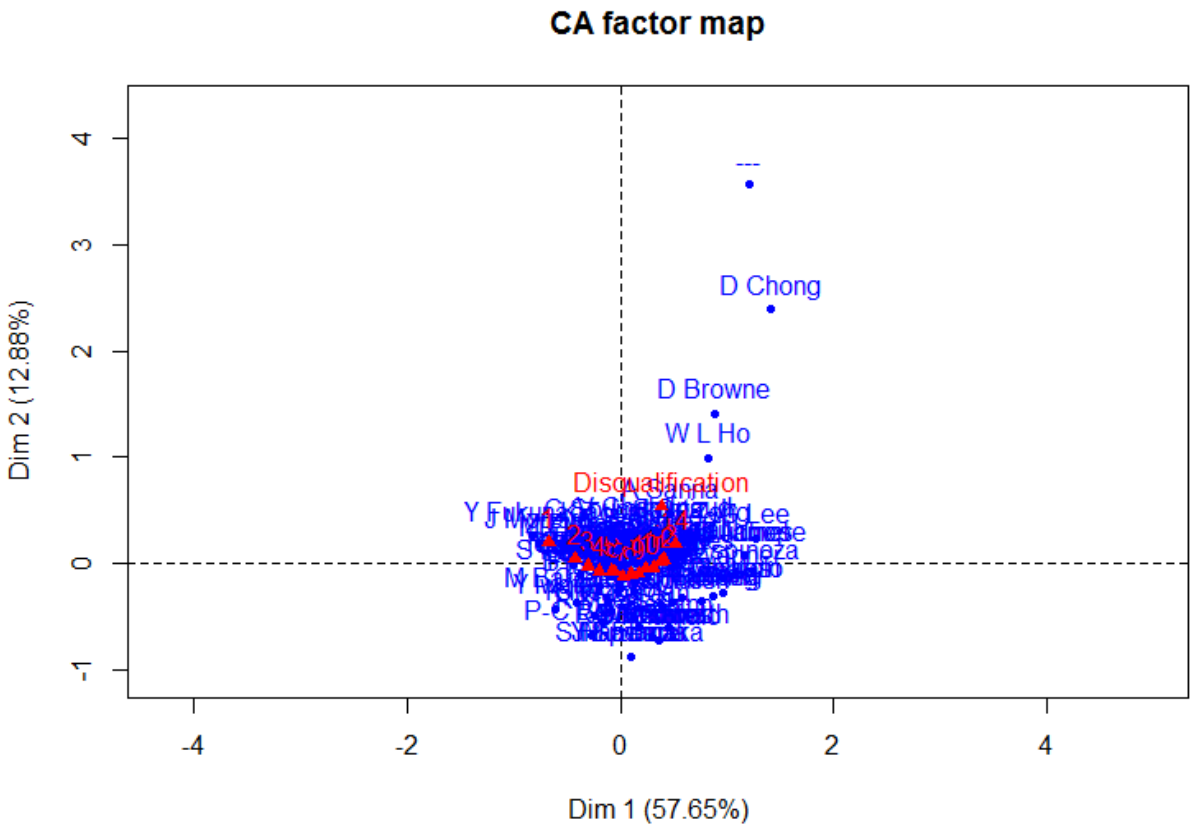
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Disqualification
---	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27
A Atzeni	3	3	3	6	8	6	9	3	9	7	8	7	5	2	3
A Badel	7	8	8	12	11	6	11	11	10	8	14	7	2	2	4
A de Vries	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
A Sanna	2	1	1	1	0	0	0	0	1	5	1	3	2	2	2
A Suborics	15	24	21	22	25	35	25	30	31	36	38	36	13	14	29
B Fayd'herbe	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0
B Melham	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
B Prebble	127	148	139	132	100	116	91	93	102	92	93	61	37	22	51
B Shinn	0	0	0	0	0	0	1	2	1	2	0	0	0	0	0
B Vorster	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C Brown	1	1	1	2	2	0	1	0	0	1	2	0	1	1	1
C F Wong	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
C Hayes	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
C K Tong	4	8	11	15	30	34	26	46	54	72	71	82	47	53	69
C Murray	4	1	5	13	7	7	19	8	13	7	5	7	2	1	4
C Perkins	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
C Reith	2	5	9	10	6	10	10	11	10	16	8	11	1	5	7
C Schofield	62	83	62	87	69	75	86	72	64	64	59	57	27	16	40
C Soumillon	3	4	4	3	3	1	0	1	3	1	3	1	3	1	3
C Williams	9	8	10	3	5	4	4	8	5	7	4	5	3	1	1
C Y Ho	72	72	96	122	112	129	136	114	155	134	131	118	61	50	87
C Y Lui	7	11	10	11	13	14	13	14	8	21	9	15	15	6	9
D Browne	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

Showing 1 to 24 of 106 entries

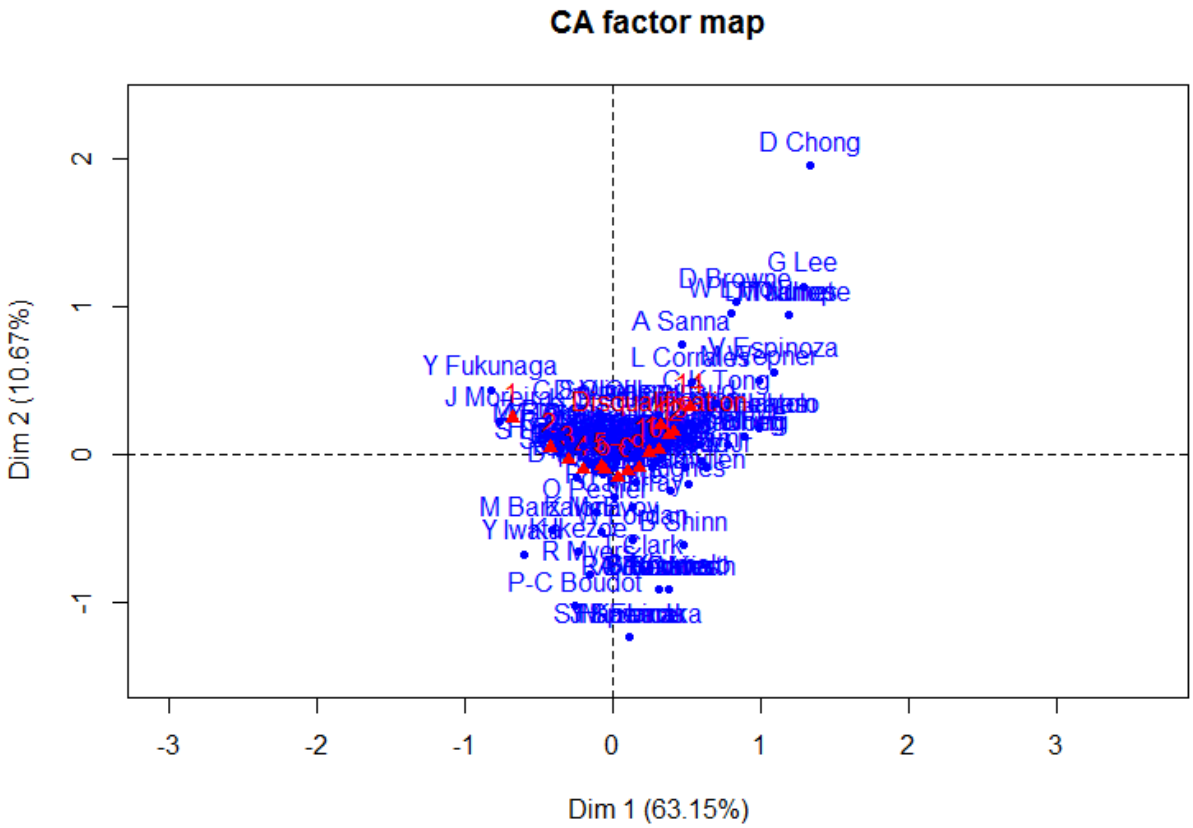
Il existe toujours 106 jockeys, mais avec cette fois-ci, seulement 15 positions :

```
levels(race_horses$finishing_position)
 [1] "1"    "2"    "3"    "4"    "5"    "6"    "7"    "8"    "9"    "10"   "11
"   "12"   "13"   "14"   "DISQ"
```

De nouveau, l'AFC sur le nouveau tableau donne :



Nous observons alors une amélioration de l'inertie de l'axe 1 qui passe alors à 57.65% et de l'axe 2 passant à 12.88% pour un total de 70,75%.
La représentation de l'axe 1 se confirme avec en prime l'observation d'un effet Guttmann avec l'apparition d'une ellipse sur les modalités de positions. Ainsi, les modalités à gauche correspondent belle et bien aux meilleurs jockeys et ceux à droite aux moins bon.
A ce stade il est difficile d'estimer ce que représente réellement l'axe 2.
Pour pousser plus loin, la modalité jockey '---' est alors mis en supplémentaire pour donner le graphe final :



L'inertie de l'axe 1 passe alors à 63.15% et l'axe 2 diminue à 10.67%. Par contre le total passe à 73.82% ce qui indique que notre graphe est le plus représentatif de tous. L'effet Gutmann est d'ailleurs encore plus accru. Ce qui indique qu'il existe une variable latente pour expliquer le lien position/jockey.

Ainsi, l'axe 1 représente clairement la position des jockeys et donc ceux qui sont meilleurs à gauche et ceux moins bon à droite. Y Fukanaga et J Moreira serait alors de bonne pioche pour un pari.

J Moreira	484	298	249	210	151	135	103	81	67	65	45	43	20	11	44

En effet, sur le tableau de fréquence, J Moreira semble abonné aux meilleures places.

Enfin, le second axe représenterait la participation et la dispersion. Les valeurs aux milieux ayant participé au plus de course et dont les résultats seraient proches de la moyenne globale.

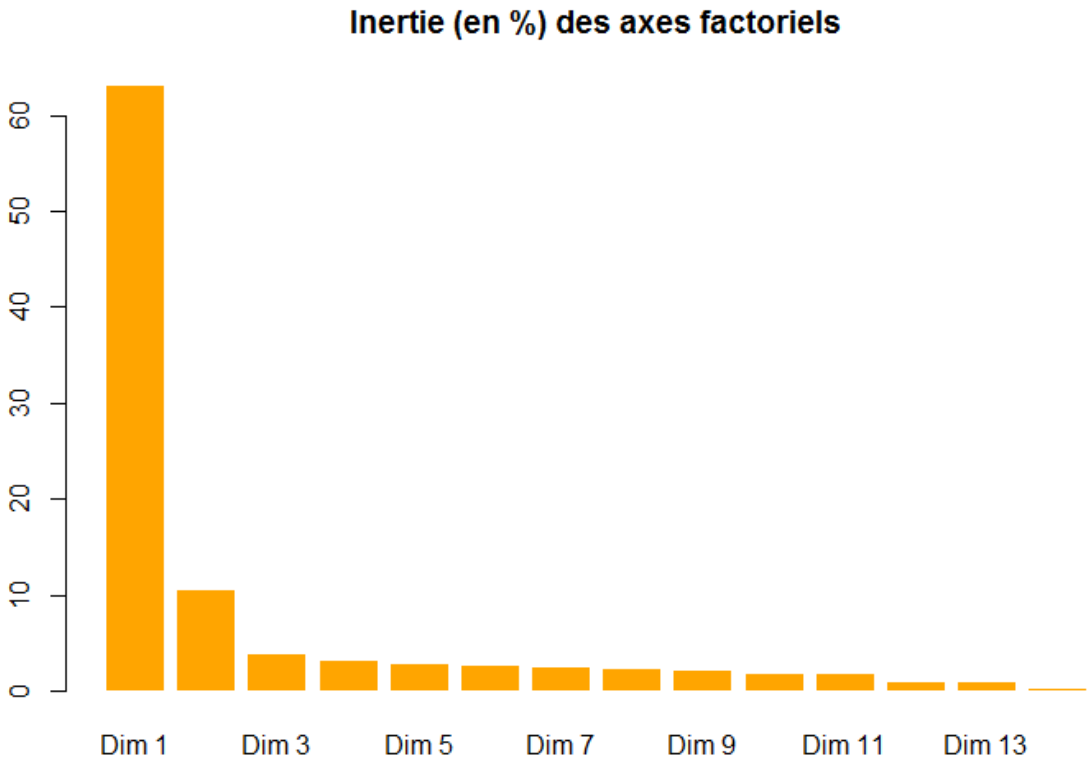
Par exemple D Chong tout en haut à gauche ne possède que deux courses pour une 14ème place et une disqualification. Soit une faible participation et un mauvais score entrainant forte dispersion (moyenne de 14). Il est clairement le jockey avec les pires résultats et représente donc une valeur aberrante.

0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

De même pour PC-C Boudot, ayant fait 2 courses avec des résultats corrects. D'où son positionnement en bas au milieu avec une moyenne de 5.5 ce qui est très raisonnable

P-C Boudot	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0

.



Le Bar plot montre que l'inertie de nos 2 axes atteint en cumulé plus de 70% et qu'il existe une césure entre la dimension 2 et 3. Ce qui est plutôt intéressant que les deux axes de l'AFC représentent bien les données.

Cheval et position

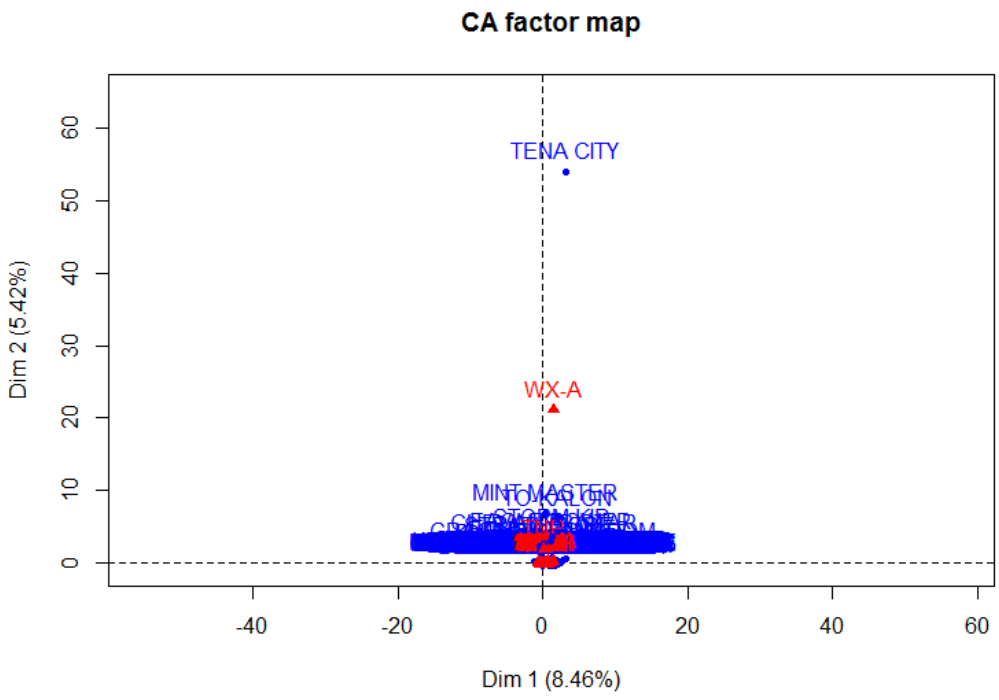
Le même travail est effectué pour les chevaux. Soit la création d'un tableau d'effectif entre horse_name et finishing_position.

	1	1 DH	2	2 DH	3	3 DH	4	4 DH	5	5 DH	6	6 DH	7	7 DH	8	8 DH	9	9 DH	10	10 DH	11	11 DH
A BEAUTIFUL	2	0	0	0	2	0	1	0	1	0	3	0	3	0	2	0	2	0	2	0	4	
A FAST ONE	1	0	2	0	4	0	1	0	1	0	3	0	2	0	1	0	3	0	1	0	0	
A SHIN HIKARI	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
A SHIN HOOF	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
A STAR LUSTER	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	
ABLE DEED	1	0	0	0	2	0	1	0	1	0	1	0	2	0	1	0	1	0	1	0	0	
ABLE FRIEND	7	0	1	0	3	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ABLE TALENT	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	2	0	1	
ABLE WARRIOR	2	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	
ACCEPTED	1	0	2	0	1	0	1	0	2	0	0	0	1	0	2	0	1	0	0	0	0	
ACCESS YEARS	1	0	2	0	2	0	1	0	3	0	3	0	0	0	0	0	0	0	0	0	1	
ACCESSION YEARS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	
ACCLAIMED LIGHT	1	0	1	0	3	0	2	0	0	0	5	0	3	0	2	0	2	0	0	0	0	
ACE KING	3	0	1	1	2	0	3	0	1	0	3	0	2	0	1	0	1	0	0	0	3	
ACTIVISM	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	
ACTUARIAT	0	0	0	0	1	0	2	0	0	0	2	0	0	0	3	0	2	0	3	0	2	
ACUMEN	0	0	0	0	0	0	1	0	1	0	2	0	0	0	0	0	1	0	3	0	0	
ADDOLE	1	0	2	0	3	0	2	1	1	0	1	0	0	0	3	0	3	0	2	0	1	
ADEPTO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0	
ADMIRABLE	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	
ADMIRAL LORD	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
ADVENTURER	4	0	3	0	0	0	1	0	1	0	1	0	0	0	3	0	0	0	1	0	1	

Ce tableau est composé de 2162 chevaux et des 36 cas positions différentes (en comptant les disqualifications).

L'AFC sur le tableau donne le résultat suivant :

-



L’inertie des axes est très faible (inférieur à 10%) donc difficilement exploitable. De plus il existe des valeurs extrêmes comme TENA CITY.

Tout comme pour les jockeys, une fusion sur les différentes positions pour affiner les observations est réalisée.

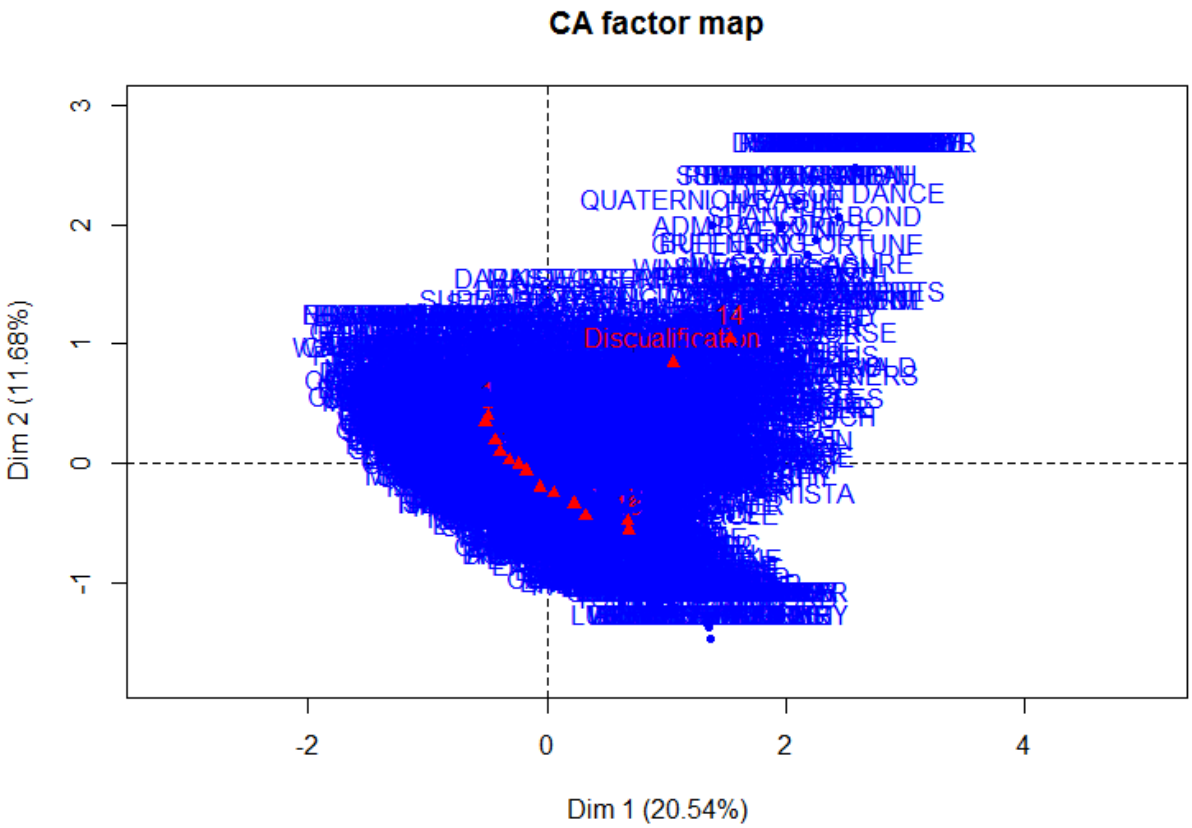
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Disqualification
A BEAUTIFUL	2	0	2	1	1	3	3	2	2	2	4	1	0	0	1
A FAST ONE	1	2	4	1	1	3	2	1	3	1	0	2	0	0	2
A SHIN HIKARI	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
A SHIN HOOF	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1
A STAR LUSTER	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
ABLE DEED	1	0	2	1	1	1	2	1	1	1	0	1	2	0	0
ABLE FRIEND	7	1	3	2	0	1	0	0	0	0	0	0	0	0	1
ABLE TALENT	0	0	0	1	0	0	1	1	0	2	1	3	1	3	4
ABLE WARRIOR	2	0	0	0	1	1	1	0	0	0	0	2	0	0	0
ACCEPTED	1	2	1	1	2	0	1	2	1	0	0	0	0	0	0
ACCESS YEARS	1	2	2	1	3	3	0	0	0	0	1	0	0	0	0
ACCESSION YEARS	0	0	0	0	0	0	0	1	0	0	2	0	0	0	0
ACCLAIMED LIGHT	1	1	3	2	0	5	3	2	2	0	0	0	0	0	0
ACE KING	3	2	2	3	1	3	2	1	1	0	3	1	0	0	0
ACTIVISM	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
ACTUARIAT	0	0	1	2	0	2	0	3	2	3	2	2	2	0	0
ACUMEN	0	0	0	1	1	2	0	0	1	3	0	0	0	0	1
ADDOLE	1	2	3	3	1	1	0	3	3	2	1	1	0	0	1
ADEPTO	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0
ADMIRABLE	0	0	0	0	0	0	1	0	0	1	0	1	0	0	1
ADMIRAL LORD	0	0	0	0	0	1	0	0	0	0	0	0	0	1	2
ADVENTURER	4	3	0	1	1	1	0	3	0	1	1	0	0	0	0
AEROLUMINANCE	1	0	2	1	1	1	1	1	1	0	1	1	2	0	0
AEROSPEED	0	1	1	2	0	4	0	0	1	0	1	0	0	1	1

Showing 1 to 24 of 2,162 entries

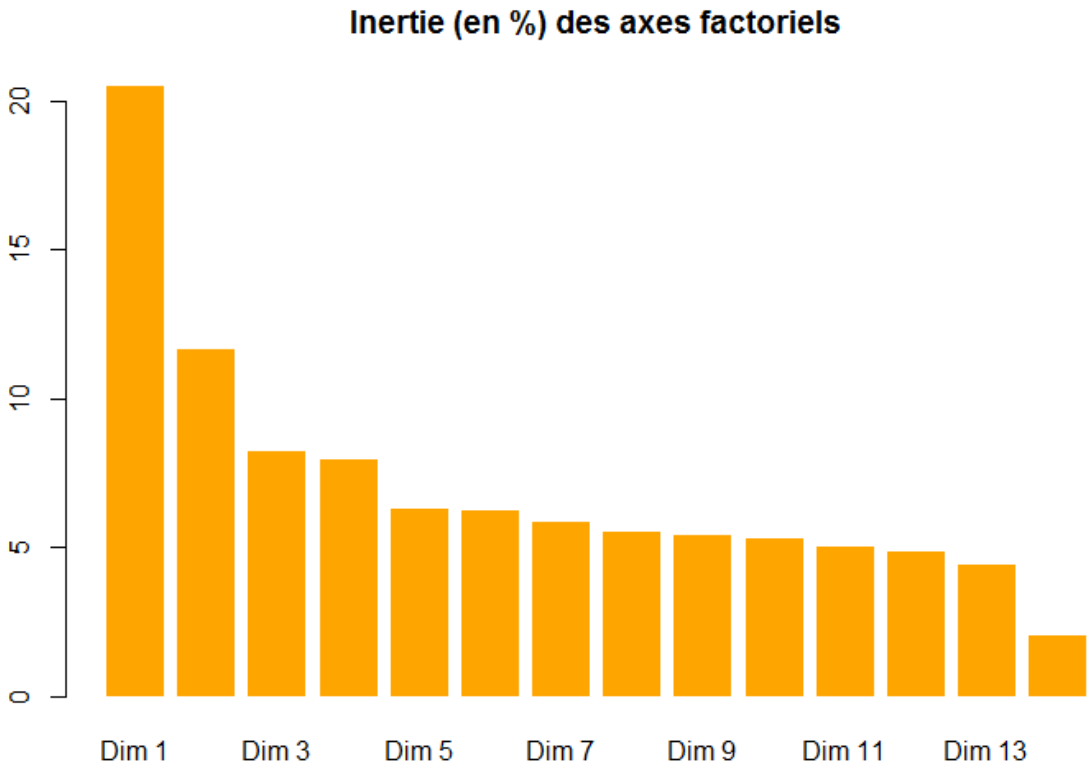
De même, Il existe toujours 2162 chevaux, mais avec cette fois-ci, seulement 15 positions :

```
levels(race_horses$finishing_position)
 [1] "1"    "2"    "3"    "4"    "5"    "6"    "7"    "8"    "9"    "10"   "11
"   "12"   "13"   "14"   "DISQ"
```

Le nouveau tableau donne alors avec la fonction CA() le factor map suivant :



L'inertie de l'axe 1 est alors de 20,54% et celui de l'axe 2 de 11.68%. Comme pour les jockeys, celui-ci présente un effet Guttmann. Il existerait donc une variable latente entre ces deux variables. Ainsi sur l'axe 1, les meilleurs chevaux se situent à gauche et les moins bons à droite. Honnêtement, il est difficile de connaitre la signification de l'axe avec les différentes modalités se superposant. Une des suppositions logiques serait que le comportement est identique à celui pour les jockeys et donc représenterait la participation et dispersion.



On voit sur ce schéma, qu'il existe une césure entre les dimensions 2 et 3. Nous décidons alors de travailler avec cette césure, bien que cela représente 30%, car nous voulons faire la représentation en 2D.

En observant ainsi les différents résultats, la conclusion serait que les jockeys et les chevaux influent tout deux sur la position finale d'une course. Le jockey bien plus d'ailleurs qu'un cheval avec 63.15% contre 20.54%. Mais surtout, il existerait une ou des variables latentes.

Analyse des correspondances multiples

Nous savons avec les différentes AFC précédentes que les jockeys et les chevaux sont liés avec la finishing position.
Nous savons aussi, après de nombreuse recherche d'avis d'expert et d'habitués d'hippodrome qu'un jockey seul ou un cheval seul ne va pas forcement influencer une course. En effet, Les chevaux étant des êtres vivants et dotés de sentiments, tout comme les jockeys. Ceux-ci vont alors développer des relations « amicales » et sportive. Ainsi, la variable latente supposés dans les AFC serait la relation, l'entente qu'il existe entre un cheval et un jockey.

Pour expliquer, même si on prend le meilleur jockey de l'histoire et le cheval le plus performant, Il sera difficile pour un jockey de tirer tout le potentiel d'une bête qui ne peut pas le cadrer.

Pour vérifier les dires des experts, il est préférable de pratiquer une ACM entre jockeys, chevaux et finishing_position et vérifier si la variable expliquant le plus le résultat d'une course était justement ce couple homme/cheval.

Préparation des données

Pour cela, les 3 variables sont sélectionnées dans un nouveau tableau :

```
> str(race_horses.phj)
'data.frame':  30189 obs. of  3 variables:
 $ finishing_position: Factor w/ 36 levels "1","1 DH","2",..: 1 3 5
7 9 11 13 15 17 19 ...
 $ horse_name        : Factor w/ 2162 levels "A BEAUTIFUL",..: 445
1474 781 1888 1935 2098 256 348 1900 1860 ...
 $ jockey            : Factor w/ 106 levels "---","A Atzeni",..: 9
30 104 46 106 6 22 42 55 92 ...
```

Ce tableau est alors composé de 3 variables qualitatives « finishing_position », « horse_name », et « jockey ».

Comme pour les AFC, il faut fusionner les positions et simplifier.

```
> levels(race_horses.phj$finishing_position)
 [1] "1"     "1 DH"  "2"     "2 DH"  "3"     "3 DH"  "4"     "4 DH"
 [9] "5"     "5 DH"  "6"     "6 DH"  "7"     "7 DH"  "8"     "8 DH"
[17] "9"     "9 DH"  "10"    "10 DH" "11"    "11 DH" "12"    "12 DH
"
[25] "13"    "14"    "DISQ"  "DNF"   "FE"    "PU"    "TNP"   "UR"
[33] "WV"    "WV-A"  "WX"    "WX-A"
```

Les différents types de disqualification sous le tag « DISQ » et les ex-aequo sous la position (Ex : « 1 DH » => « 1 »)

Pour cela, nous utilisons les expressions régulières pour repérer les valeurs et les modifier avec la valeur souhaitée.

```
> race_horses.phj$finishing_position <- str_replace_all(race_horses.phj$fini
shing_position,"NA|DNF|FE|PU|TNP|UR|WV(-A){0,1}|WX(-A){0,1}", "DISQ")
> race_horses.phj$finishing_position <- str_replace_all(race_horses.phj$fini
shing_position," DH", "")
levels(race_horses.phj$finishing_position)
 [1] "1"    "2"    "3"    "4"    "5"    "6"    "7"    "8"    "9"
[10] "10"   "11"   "12"   "13"   "14"   "DISQ"
```

Il reste plus qu'à transformer les finishing_position en factor pour l'ACM vu que ce sont des caractères (Chr). Pour cela, il faut utiliser la fonction as.factor().

```
>race_horses.phj$finishing_position <- as.factor(race_horses.phj$finishing_pos
ition)
> str(race_horses.phj)
'data.frame':  30189 obs. of  3 variables:
 $ finishing_position: Factor w/ 15 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8
9 10 ...
 $ horse_name        : Factor w/ 2162 levels "A BEAUTIFUL",..: 445 1474 781 18
88 1935 2098 256 348 1900 1860 ...
 $ jockey            : Factor w/ 106 levels "---","A Atzeni",..: 9 30 104 46 1
06 6 22 42 55 92 ...
```

ACM sur toutes les données

Le tableau est prêt, nous pouvons lancer notre ACM avec dudi.acm.

```
acm.race_horses<-dudi.acm(race_horses.phj, scannf=FALSE)       :
```

Nous obtenons alors le résultat suivant :

```
summary(acm.race_horses)
Class: acm dudi
Call: dudi.acm(df = race_horses.phj, scannf = FALSE)

Total inertia: 760

Eigenvalues:
    Ax1     Ax2     Ax3     Ax4     Ax5
 0.6684  0.6679  0.6677  0.6675  0.6674

Projected inertia (%):
    Ax1     Ax2     Ax3     Ax4     Ax5
0.08795 0.08788 0.08785 0.08782 0.08782
```

Nous remarquons que nous avons énormément de dimensions, 2260.

De plus chaque axe possède une très faible inertie :

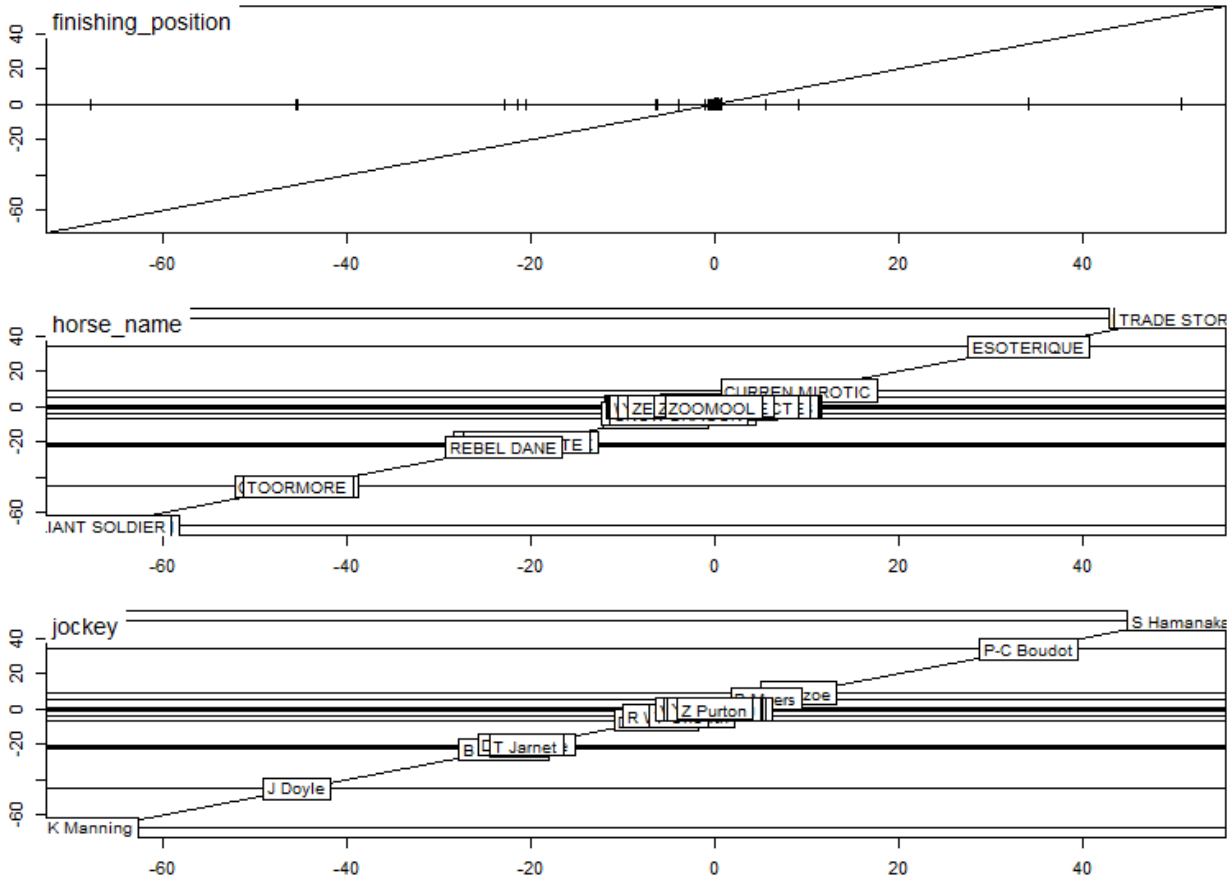
```
head (inertia.dudi(acm.race_horses)$TOT)
      inertia       cum     cum(%)
Ax1 0.6684212 0.6684212 0.08795015
Ax2 0.6678900 1.3363112 0.17583042
Ax3 0.6676556 2.0039668 0.26367984
Ax4 0.6674610 2.6714278 0.35150365
Ax5 0.6674148 3.3388426 0.43932139
Ax6 0.6673080 4.0061507 0.52712507

Cumulative projected inertia (%):
    Ax1   Ax1:2   Ax1:3   Ax1:4   Ax1:5
0.08795 0.17583 0.26368 0.35150 0.43932

(Only 5 dimensions (out of 2260) are shown)
```

En cumulé, les 6 premier axes ne représente que 0.5%. Cela nous pose problème car cela indique que notre ACM est très peu représentative.

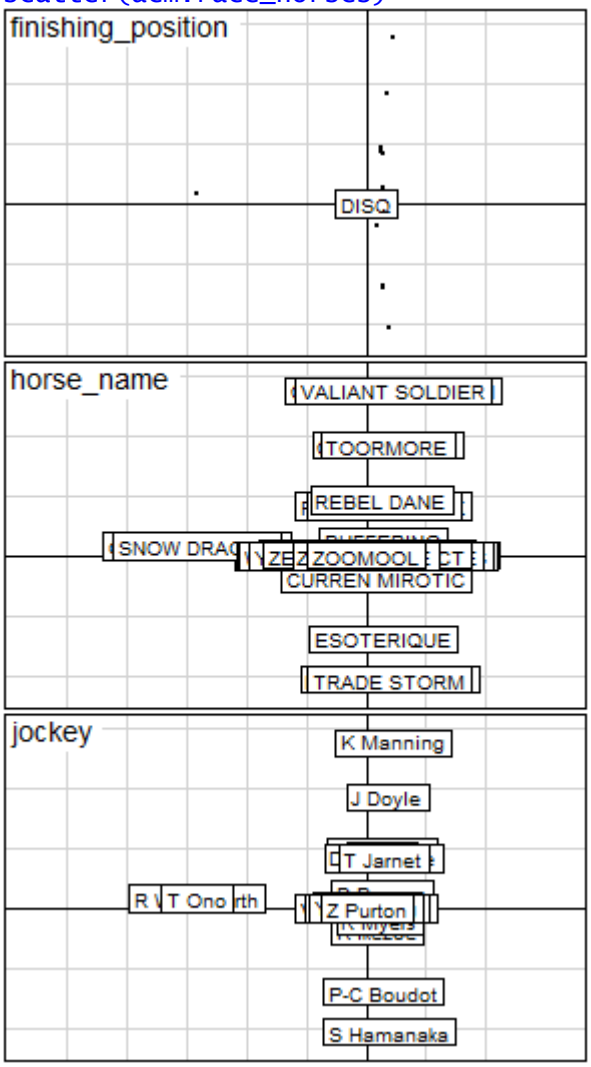
Aucune des inerties de dépasse les 0.6, notre ACM est illisible. Cela est dû au nombre important de modalité.

Les graphiques obtenus avec la fonction score ou scatter n'expliquent pas grand-chose. On ne voit à chaque fois que quelques éléments sur la totalité.

```
score(acm.race_horses ,xax =2, xlim = 100, ylim=1000)
```



```
scatter(acm.race_horses)
```



ACM sur des échantillons.

A ce stade, nous n'avions aucune idée de comment lire, interpréter ou tirer des informations de l'ACM
Le grand nombre de modalité était notre principal problème.

Une idée fut de ne prendre que quelques jockeys pour limiter le nombre de modalité.

Ainsi un dataframe réduit fut mis en place avec 3 jockeys.

Les jockeys étaient sélectionnés aléatoirement avec la fonction sample().

```
select.random_jockey <- sample(1:summary(levels(race_horses$jockey))[1],size = 3,replace=F)
selected_jockey <-list_jockey[[1]][select.random_jockey]
selected_jockey
race_horses.phj_reduce<-race_horses.phj[race_horses.phj$jockey %in% selected_jockey,]
race_horses.phj_reduce
race_horses.phj_reduce <- droplevels(race_horses.phj_reduce[1:1000,])
race_horses.phj_reduce$finishing_position <- factor(race_horses.phj_reduce$finishing_position,
levels=c("1","2","3","4","5","6","7","8","9","10", "11", "12", "13", "14", "DISQ"))
```

```
str(race_horses.phj_reduce)
'data.frame':  1000 obs. of  3 variables:
 $ finishing_position: Factor w/ 14 levels "1","2","3","4",..: 7 5 7 6 9 5 10
7 12 10 ...
 $ horse_name        : Factor w/ 500 levels "A FAST ONE","ABLE TALENT",..: 57
274 384 246 434 488 401 340 404 421 ...
 $ jockey            : Factor w/ 3 levels "C Y Ho","D Lane",..: 1 3 1 3 1 3 1
1 3 1 ...
```

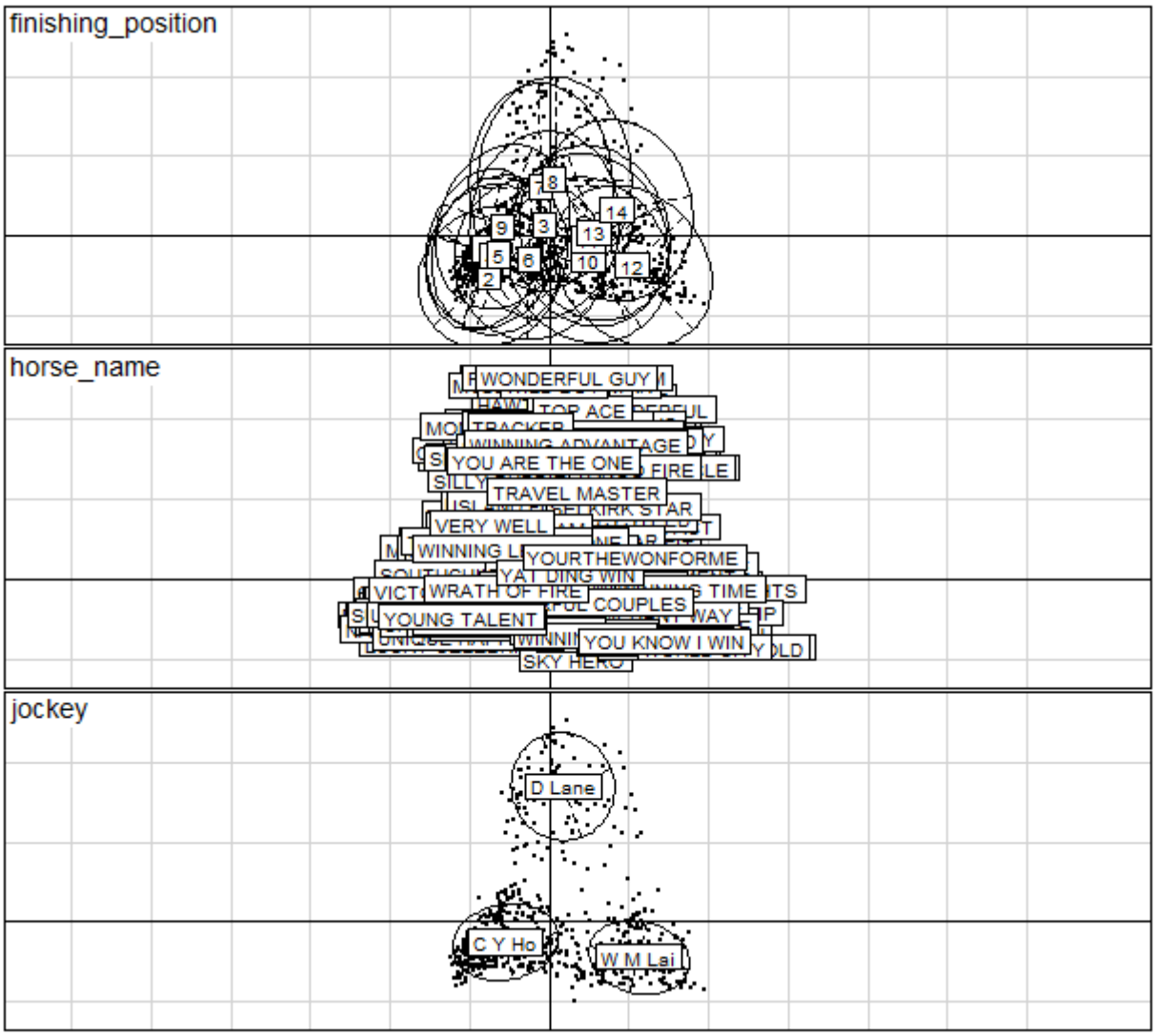
Malheureusement, en fonction des jockeys sélectionnés l'inertie des 5 premiers axes pouvait varier de 2% à 20%.

Dans ce cas, elle est de 2%

```
> head (inertia.dudi(acm.race_horses_reduce)$TOT)
      inertia       cum    cum(%)
Ax1 0.7071183 0.7071183 0.4126893
Ax2 0.6505240 1.3576423 0.7923489
Ax3 0.5972240 1.9548663 1.1409015
Ax4 0.5913509 2.5462172 1.4860265
Ax5 0.5825634 3.1287807 1.8260229
Ax6 0.5763911 3.7051717 2.1624170
```

Cela est bien trop aléatoire pour être exploité.

Par contre, les différents graphiques étaient bien plus faciles à interpréter :



Dans ce cas C Y Ho est plutôt abonné aux bonnes places contrairement à W M Lai.
De même, les jockeys semblent avoir un partager des chevaux.

Exemple avec C Y Ho et Wrath of Fire qui ont finis 3ème ensemble.

finishing_position	horse_number	horse_name	horse_id	jockey
All	All	Wrath	All	C Y
3	11	WRATH OF FIRE	L420	C Y Ho

Alors que Wrath of fire et W M Lai ont finis 9ème.

finishing_position	horse_number	horse_name	horse_id	jockey
All	All	Wrath	All	Lai
9	7	WRATH OF FIRE	L420	H W Lai

ACM J Moreira

Bien que plus lisible en ayant moins de modalité, le fait de tirer des jockeys aléatoires laisser trop de risque de tomber sur des données aberrantes.

Comme l'objectif est de comprendre si en effet, un couple cheval/jockey qui fonctionne bien est la clé du succès. La dernière idée fut de faire un ACM seulement sur un des meilleurs jockeys : J Moreira 484 fois vainqueur.

J Moreira	484	298	249	210	151	135	103	81	67	65	45	43	20	11	44

Pour cela, un dernier cas fut entrepris pour comprendre pourquoi J Moreira gagnait si souvent.

Un nouveau jeu de données fut créé :

```
race_horses.phj_reduceJM<-race_horses.phj[race_horses.phj$jockey == "J Moreira",]
race_horses.phj_reduceJM
race_horses.phj_reduceJM <- droplevels(race_horses.phj_reduceJM[1:1000,])
race_horses.phj_reduceJM$finishing_position <- factor(race_horses.phj_reduceJM$finishing_position,
levels=c("1","2","3","4","5","6","7","8","9","10", "11", "12", "13", "14", "DISQ"))
```

```
> str(race_horses.phj_reduceJM)
'data.frame':  1000 obs. of  3 variables:
 $ finishing_position: Factor w/ 15 levels "1","2","3","4",..: 4 10 4 1 2 14 4
3 1 1 ...
 $ horse_name        : Factor w/ 423 levels "A FAST ONE","ABLE FRIEND",..: 363
329 286 154 191 75 289 397 362 49 ...
 $ jockey            : Factor w/ 1 level "J Moreira": 1 1 1 1 1 1 1 1 1 1 ...
```

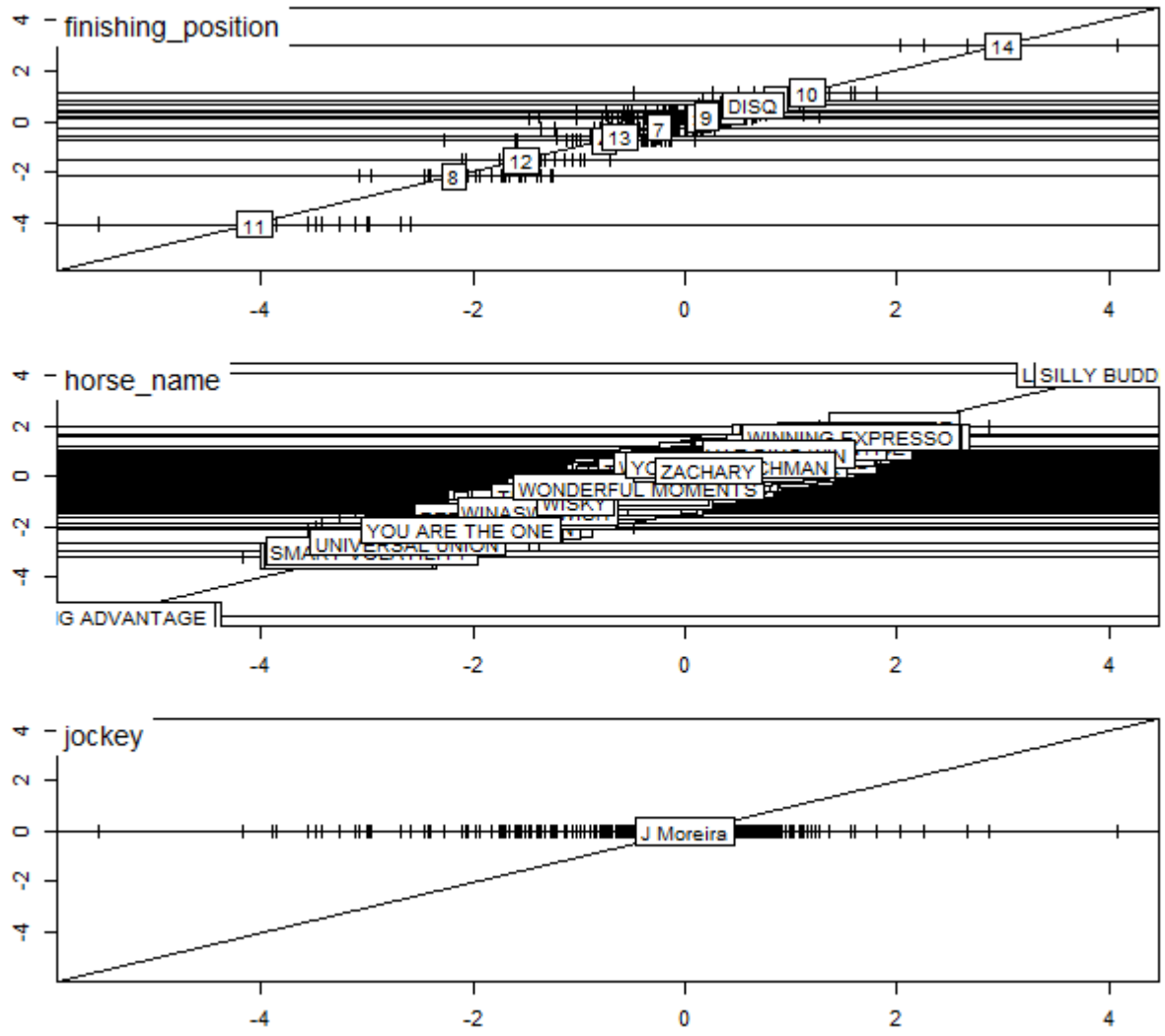
A Noter que Moreira à quand même eut 423 chevaux différents.

Tout comme précédemment, l'inertie des premiers axes est faible.

```
> summary(acm.race_horses_reduceJM$eig)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.06897 0.33333 0.33333 0.33333 0.33333 0.59769
> summary(acm.race_horses_reduceJM)
Class: acm dudi
Call: dudi.acm(df = race_horses.phj_reduceJM, scannf = FALSE)

Total inertia: 145.3

Eigenvalues:
    Ax1     Ax2     Ax3     Ax4     Ax5
 0.5977  0.5782  0.5749  0.5692  0.5626

Projected inertia (%):
    Ax1     Ax2     Ax3     Ax4     Ax5
 0.4113  0.3978  0.3956  0.3916  0.3871

Cumulative projected inertia (%):
    Ax1   Ax1:2   Ax1:3   Ax1:4   Ax1:5
 0.4113  0.8091  1.2047  1.5963  1.9834

(Only 5 dimensions (out of 436) are shown)

> head (inertia.dudi(acm.race_horses_reduceJM)$TOT)
      inertia       cum    cum(%)
Ax1 0.5976938 0.5976938 0.4112572
Ax2 0.5781550 1.1758488 0.8090702
Ax3 0.5749149 1.7507637 1.2046539
Ax4 0.5691797 2.3199433 1.5962913
Ax5 0.5626063 2.8825497 1.9834057
Ax6 0.5601053 3.4426550 2.3687993
```

Cela doit sans doute être dû au nombre important de chevaux utilisé par J Moreira.

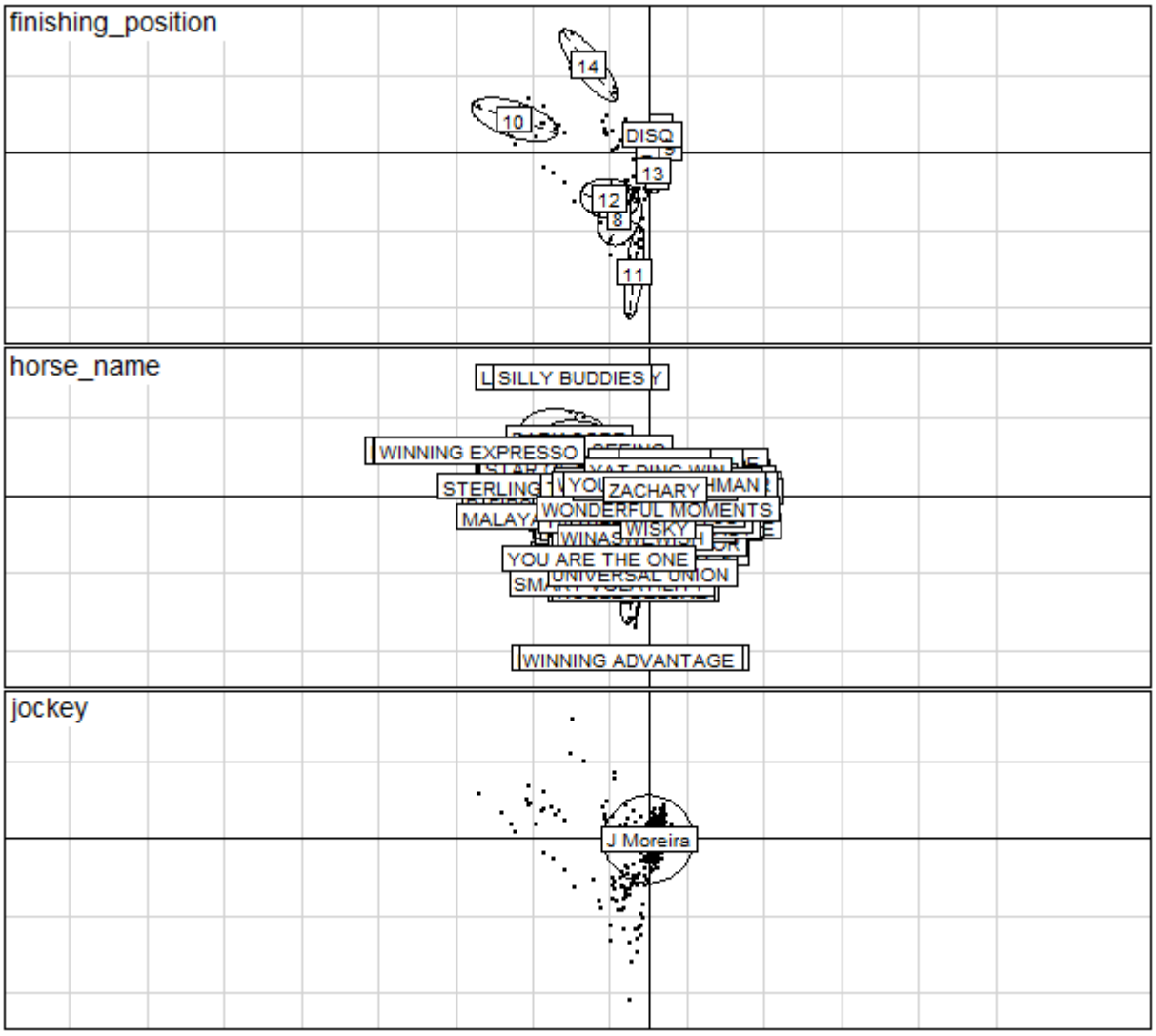
Par contre, les graphes permettent de voir quels sont les chevaux préférés de J Moreira et les résultats qu'ils ont eu.



Par exemple, Silly Buddies et Moreira sont arrivés en mauvaise position à la 14ème place.
De même, on observe que J Moreira a rarement des mauvais résultats.

Ces idées se confirment avec le graphe obtenu avec la fonction scatter où Silly Buddies se retrouve de nouveau à la 14ème place.
De nouveau, J Moreira n'a pas énormément de chevaux ayant finit en mauvaise position, la plupart se concentre sur les meilleures places.

Ce qui n'est pas étonnant vu les statistiques de J Moreira avec par exemple 484 victoires.



Mais ce dernier graphe indique aussi que les chevaux ayant de mauvais résultats sont moins utilisés.

Cela permet de se poser la question suivante :
Le secret de Moreira serait-t-il de garder que les chevaux avec qui il s'entend bien ?

Pour cela, le taux de réutilisation des chevaux a été calculé :

```
couple_JC_JM <- race_horses.phj_reduceJM %>%
  filter(jockey == "J Moreira") %>%
  group_by(jockey,finishing_position,horse_name) %>%
  summarise(frequence = n())
couple_JC_JM

entente_JC_JM <- couple_JC %>%
  filter(jockey == "J Moreira") %>%
  group_by(finishing_position) %>%
  summarise(moyenne = mean(frequence), sd=sd(frequence), total = n())
entente_JC_JM>
```

	finishing_position	moyenne	sd	total
1	1	1.576547	1.0739810	307
2	2	1.366972	0.7456238	218
3	3	1.310526	0.5939686	190
4	4	1.242604	0.5178553	169
5	5	1.118519	0.3466654	135
6	6	1.144068	0.4191041	118
7	7	1.095745	0.2958185	94
8	8	1.038462	0.1935524	78
9	9	1.046875	0.2130420	64
10	10	1.048387	0.2163345	62
11	11	1.046512	0.2130826	43
12	12	1.048780	0.2180848	41
13	13	1.000000	0.0000000	20
14	14	1.000000	0.0000000	11
15	DISQ	1.031250	0.1767767	32

Ainsi, Le jockey J Moreira utilise en moyenne 1.57 un cheval étant arrivé en 1ère position alors qu'un cheval étant arrivé en dernière position ne sera utilisé qu'une seul fois.

Cela veut dire que J Moreira ne réutilise pas un cheval dont la course fut catastrophique.
Cela démontre bien que l'entente cheval/jockey est importante et pris en compte par J Moreira. Un cheval dissident ne sera pas réutilisé.

Pour vérifier que cela ne concerne pas que J Moreira, le taux d'utilisation est de nouveau calculé. Mais cette fois ci sur l'ensemble des jockeys.

```
couple_JC <- race_horses.phj %>%
  group_by(jockey,finishing_position,horse_name) %>%
  summarise(frequence = n())
couple_JC
entente_JC <- couple_JC %>%
  group_by(finishing_position) %>%
  summarise(moyenne = mean(frequence), sd=sd(frequence), total = n())
entente_JC
```

	finishing_position	moyenne	sd	total
1	1	1.322742	0.7287163	1794
2	2	1.233333	0.5644805	1920
3	3	1.183225	0.4781731	2003
4	4	1.152913	0.4197831	2060
5	5	1.128708	0.4153456	2090
6	6	1.129727	0.4069817	2089
7	7	1.101361	0.3507912	2131
8	8	1.108428	0.3616963	2112
9	9	1.108821	0.3718498	2086
10	10	1.096712	0.3340517	2068
11	11	1.089010	0.3375852	2011
12	12	1.092973	0.3222493	1850
13	13	1.048472	0.2244785	949
14	14	1.058081	0.2738689	792
15	DISQ	1.032457	0.1940221	647
16	*NA*	1.000000	0.0000000	2

Ce tableau explique aussi pourquoi J Moreira gagne plus souvent que les autres. Quand on compare le taux de réutilisation de chevaux performants, J Moreira garde et réutilise les meilleurs chevaux plus souvent (1.57 contre 1.32) et abandonne plus facilement ceux avec qu'il il ne s'entend pas alors que les autres les réutilisent (1 contre 1.05). Peut-être que J Moreira possède plus de moyen technique et financier que les autres.

Pour conclure, de nombreux facteurs vont influencer les chances de réussites comme le poids, la distance, le draw de départ, etc. Le cheval va lui aussi influencer le résultat final (comme vu en AFC de 20%) mais la variable la plus importante reste le jockey qui représente le paramètre (63% en AFC) à regarder lors d'un pari. Mais ne s'occuper que du jockey serait une erreur de débutant. Les experts ont bel et bien raison. Une dernière variable est à prendre en compte pour trouver le vainqueur, et c'est tout simplement la relation qui existe entre le cheval et son jockey.

Modélisation

Subset selection

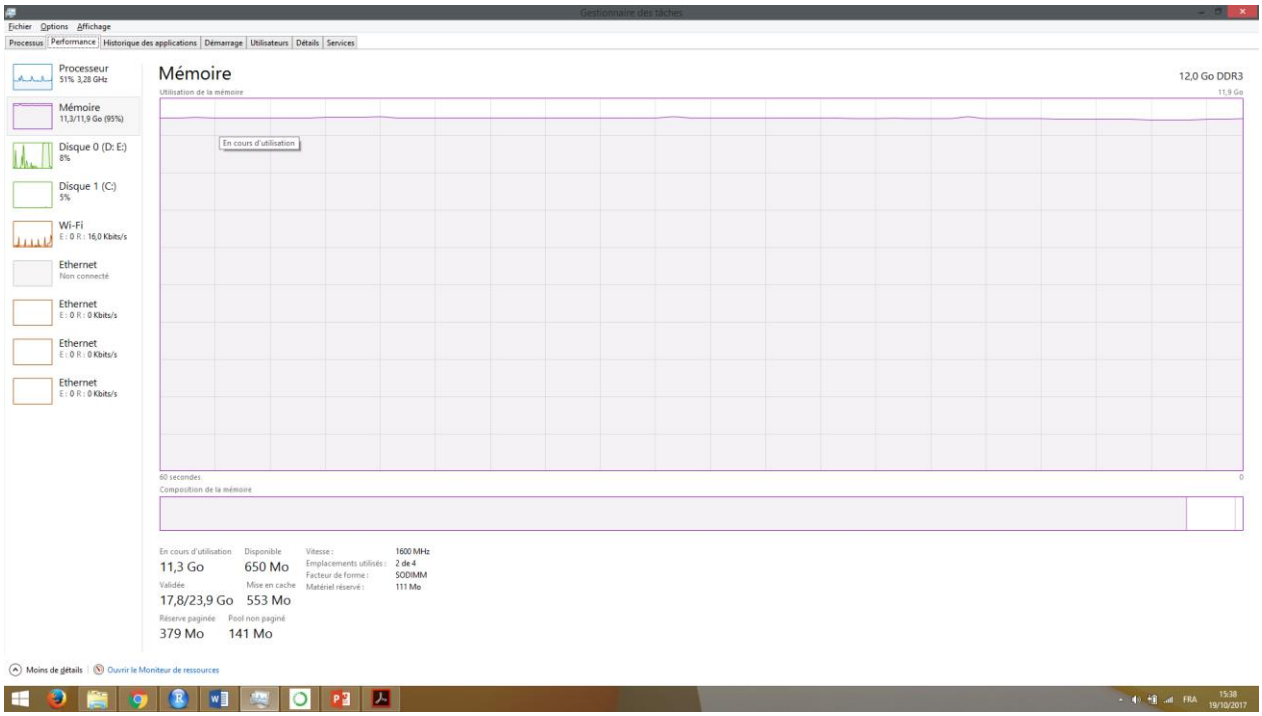
L'objectif de l'utilisation du modèle de lasso est de créer un modèle afin de modéliser la position finale d'un cheval en fonction des variables du dataset.

Au départ, il était prévu d'effectuer la création du modèle avec une complexité de 30, c'est-à-dire en utilisant l'ensemble des variables du data set.

Mais ce fut un doux rêve :

> *La best subset selection teste toutes les combinaisons*
>
> *Possibles de variables et regarde quel est le meilleur modèle. Cette technique exhaustive, a cependant un coût énorme. « Emmanuelle Claeys TD1 »*

Cette solution consomme des ressources, mais il était délicat d'imaginer une telle consommation pour ce data set. Ce fut difficile avec les ressources disponibles.



Traitement des données

Suite à des problèmes de pc en surchauffe et des temps d'attente pouvant attendre les 12 heures.
Une liste de variable fut sélectionnée en fonction des résultats de l'analyse exploratoire.
IL s'agit des variables :

"race_course","race_class","race_number","race_distance","finishing_position","horse_number","horse_name","jockey","actual_weight","declared_horse_weight","draw","trainer"

```
explicative_variable.Horse_Race <-
subset(Horse_Race_quanti, select = c("race_course","race_class","race_number","race_distance","finishing_position","horse_number","horse_name","jockey","actual_weight","declared_horse_weight","draw","trainer"))
```

Pour pouvoir travailler, sur les données en lasso, l'ensemble des valeurs non déterminées ont été supprimé.

```
rh_model=na.omit(explicative_variable.Horse_Race)
```

Une base d'apprentissage et de test ont alors été aléatoirement créées avec pour répartition 70% et 30% et sans remise.

```
training_test.base <- sample(1:nrow(rh_model),size = .7*nrow(rh_model),replace=F)

# The new table with the 30% of the samples would be . . .

rh_model.learning <- rh_model[training_test.base,]

# The table with the 70% of the samples (the remaining) would be . . .

rh_model.test <- rh_model[-training_test.base,]
```

Regsubsets

Puis la fonction regsubsets() fut utilisé pour trouver les meilleurs modèles. En utilisant la fonction coef pour extraire le coefficient de chaque objet, il se trouva que les résultats des 5 premiers modèles était des variables liées avec des modalités.

```
> coef(best_models_forward,1:5)
[[1]]
          (Intercept) horse_nameCHEERFULJET
             13.44938               0.00000

[[2]]
          (Intercept) horse_nameCHEERFULJET      jockeyA de Vries
            13.448919               0.000000              9.551081

[[3]]
          (Intercept) horse_nameCHEERFULJET     horse_nameDARIYAN      jockeyA de Vries
            13.448919               0.000000              0.000000              9.551081

[[4]]
          (Intercept) horse_nameCHEERFULJET     horse_nameDARIYAN      jockeyA de Vries      jockey
S Hamanaka
            13.448554               0.000000              0.000000              9.551446
7.551446

[[5]]
           (Intercept) horse_nameAZTEC EMPIRE  horse_nameCHEERFULJET      horse_nameDARIYAN       j
ockeyA de Vries       jockeyS Hamanaka
            13.454045              -6.689339              0.000000              0.000000
9.545955              7.545955
```

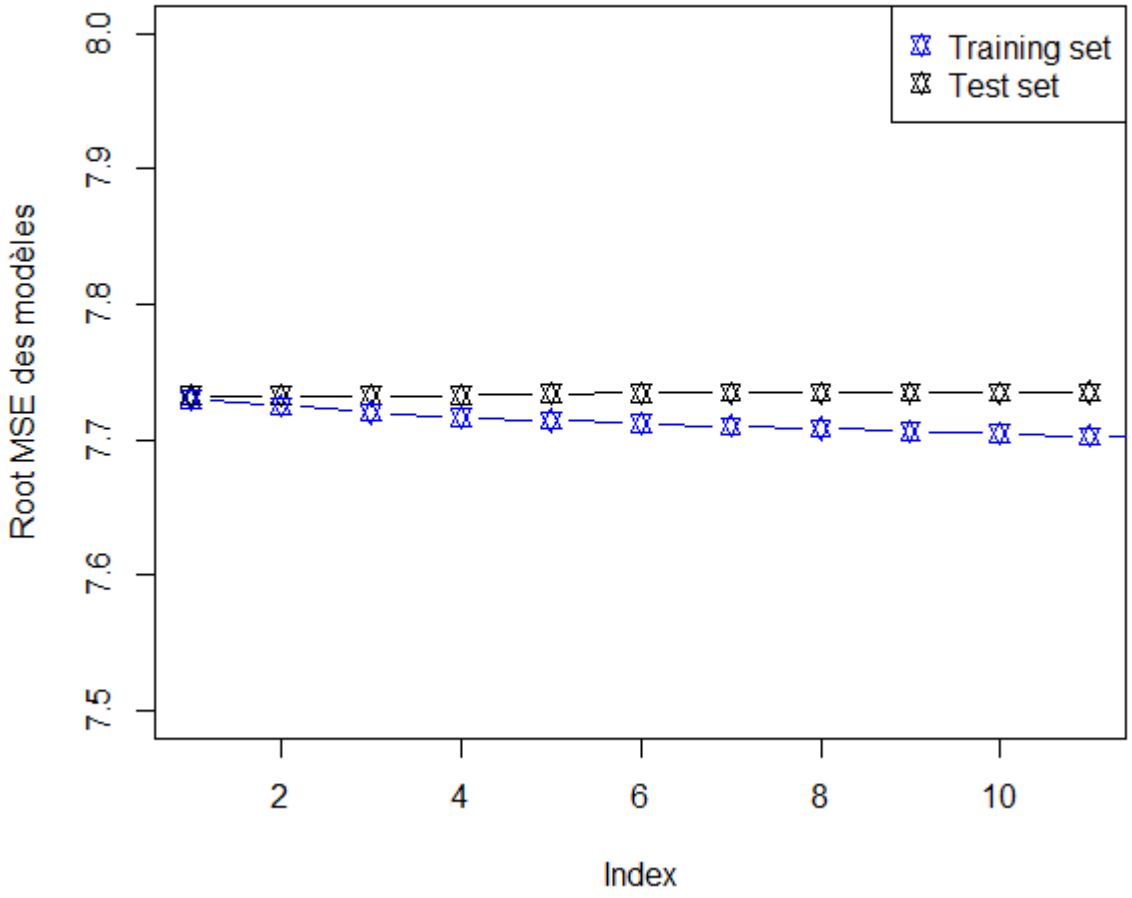
Ce fut surprenant, mais R indique ce cas de figure à travers un message de warning :

"R In leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax, force.in = force.in, : 162 linear dependencies found "

Cela indique aussi que notre modèle a pris en compte tous les couples de chevaux et jockeys et ce qui expliquerait l'utilisation de ressource et le temps d'attente important.

Pour vérifier la modélisation, il faut calculer la root mean square error.
Cela permet de connaitre la différence entre les valeurs prédites et réelles.

Le calcul des RMSE entre les bases d'entrainement et de training donne le graphe suivant :



Il semblerait que le set de test soit plutôt correct, étant proche de la MSE du training set.

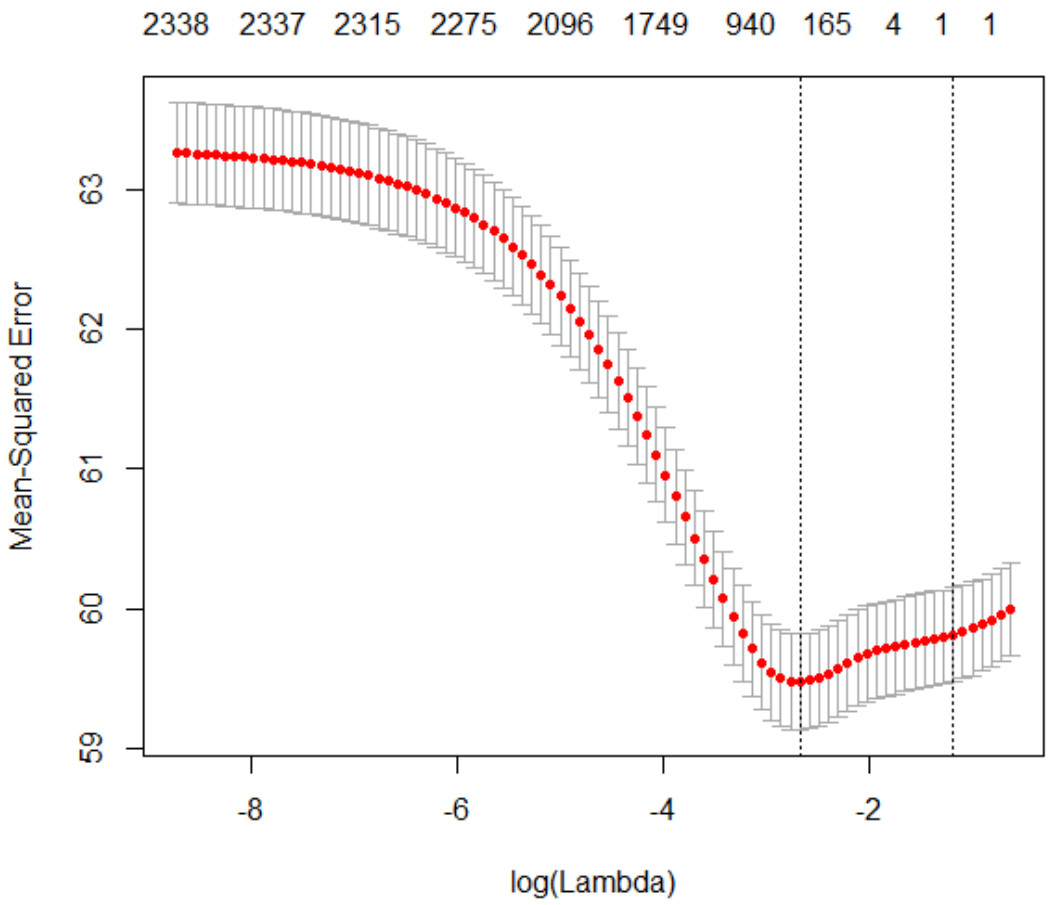
Cela est dû en partie au fait qu'il y a eu plus de 30 000 lignes de données, permettant une précision accrue.

Lasso

Pour finir, la fonction cv.glmnet() permit de faire un k-fold cross validation.

L'objectif de cette technique est de réduire les coefficients les moins pertinents en les associant avec une valeur landa. Cela va alors permettre de d'associer ces coefficients avec des poids pour trouver le meilleur modèle.

On obtient par ailleurs un schéma récapitulant tous les cas de landa possible. Et donnant les meilleurs modèles. Dans notre cas, le meilleur modèle se situe avec un landa entre 940 et 0. C'est à cet endroit que le mse est la plus faible.



Le meilleur modèle est alors :

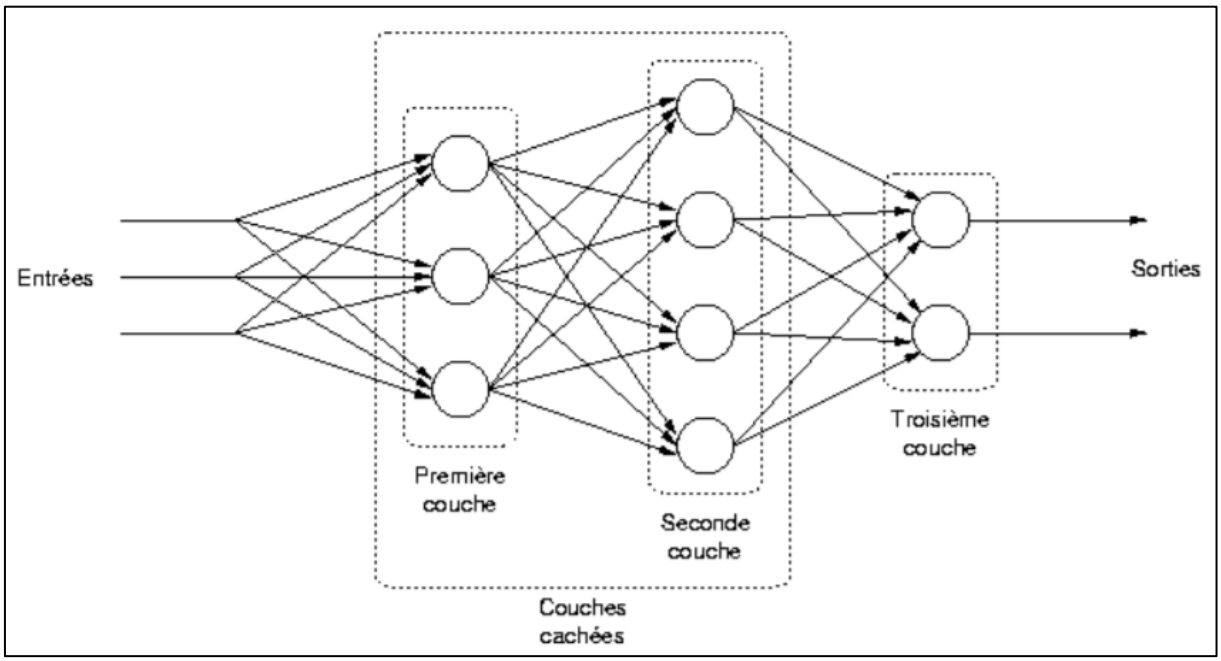
```
> numero_du_best_model
[1] 23
> #c la valeur de son landa
> lasso_model_cv$lambda[numero_du_best_model]
[1] 0.06873946
```

Réseau de neurones

Les réseaux monocouches ont la particularité de pouvoir être entrainés avec des règle d'apprentissage simple. Cependant il se limite le plus souvent à des calculs de fonctions simple.

On fait le choix de crée un réseau plus évolué, contenant par exemple des neurones cachés, c'est à dire des neurones qui ne sont ni de la couche d'entrée ni de la couche de sortie. Pourtant, si ces réseaux ont des capacités de calcul supérieurs leur apprentissage, et en particulier l'attribution des poids des connexions, deviens bien plus difficile.

Ce type réseaux, perceptron multicouche utilise le plus souvent des algorithmes de rétro propagation du gradient descendant. Le schéma si dessous l'illustre le principe des couches :



Capt. 1.4 réseau de neurones

Le perceptron est organisé en plusieurs couches. La première couche est reliée aux entrées, puis ensuite chaque couche est reliée à la couche précédente. C'est la dernière couche qui produit les sorties du PMC. Les sorties des autres couches ne sont pas visibles à l'extérieur du réseau, et elles sont appelées pour cette raison couches cachées.

Pourquoi intégrer dans la partie modèle un réseau de neurones ?

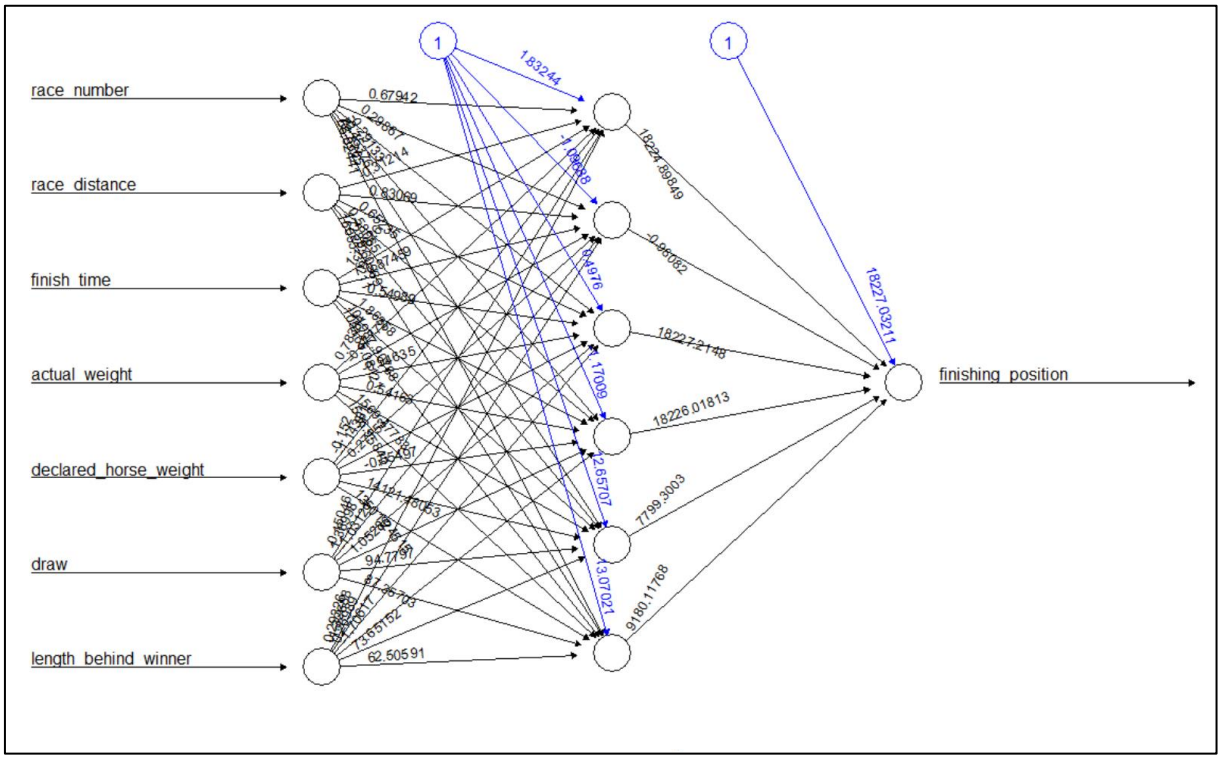
Une fois le réseau réalisé il nous sera possible de déterminer en fonction des paramètres d'entrées : poids du cheval, couloir… Quel sera sa position finale. Ainsi notre classifier s'apparentera à un modèle applicable.

On cherche à déterminer si notre cheval va finir en première position ou non.

On se propose de réaliser un réseau de neurones utilisant le principe de rétro propagation.

Plusieurs prérequis sont nécessaires pour permettre la mise en place le réseau.
Pour pouvoir calculer les coefficients de notre réseau il est indispensable que l'intégralité des vecteurs soit de type numérique. Il nous faut dans un premier temps convertir toutes les données.
On se concentre sur un tableau constitué de valeurs quantitatives.

Résultat obtenu après entrainement du réseau de neurones :



Le nombre de neurones caché a était choisi de telle sorte que l'erreur soit minimisée.

Erreur obtenue :

```
> test100

     1
1  716
2 8242
```

716 représente l'erreur d'un l'échantillon de 8 958 individus : $716 + 8242 = 8\,958$

Observation : Nous avons donc un modèle précis à environ **8%** près. Ce résultat est très satisfaisant.

Interprétation : la précision du modèle est de 8%, celle-ci peut sembler très élever, dans le sens qu'elle ne reflète pas la réalité, aucun modèle n'est capable de déterminer de façon si précise un cheval va finir ou non en première position. Mais il est important de garder à l'esprit que ce modèle est loin d'intégrer toutes les variables qui influe sur la victoire d'un cheval comme le temps, ou l'âge du cheval. En conclusion ce modèle est bien fonctionnel mais il n'est pas applicable telle qu'elle dans la réalité.

Pour aller plus loin : il serait intéressant de convertir les variables factor en num et de les intégrer dans notre réseau de neurones pour l'affiner.

Random Forest

Nous avons décidé d'appliquer l'algorithme Random Forest à notre dataset car c'est un algorithme qui est particulièrement efficace pour repérer des liens entre une variable à expliquer et des variables explicatives, surtout quand celles-ci sont nombreuses. La technique est de classer les variables explicatives en fonction de leurs liens avec la variable à expliquer. Il est aussi possible d'utiliser des variables à la fois quantitatives et qualitatives, ce qui est pratique pour notre cas. Notre modèle va servir à savoir si un cheval va gagner la course ou non.

Sélection des variables

Nous avons tout d'abord enlevé certaines variables qui n'étaient pas utiles comme la source des données. Comme notre modèle va servir à savoir si un cheval va gagner la course ou non, nous avons aussi enlevé les données que nous n'avions pas avant la fin de la course, ainsi nous avons enlevé le temps d'arrivée ou les positions des chevaux aux différents checkpoints de la course. Enfin, comme la fonction de R de Random Forest nous oblige à avoir des levels inférieurs à 53 catégories, nous avons dû supprimer les jockey, entraineurs, et noms de chevaux.

Erreur lorsque l'on a plus de 53 catégories dans un levels

```
Error in randomForest.default(m, y, ...) :
  Can not handle categorical predictors with more than 53 categories.
```

Fonction triant les variables utilisées dans notre modèle

```
Horse_Race$src <- NULL
Horse_Race$race_date <- NULL
Horse_Race$win_odds <- NULL
Horse_Race$incident_report <- NULL
Horse_Race$horse_number <- NULL
Horse_Race$race_name <- NULL
Horse_Race$horse_name <- NULL
Horse_Race$jockey <- NULL
Horse_Race$trainer <- NULL
#On a pas ces données au début de la course
Horse_Race$sectional_time <- NULL
Horse_Race$running_position_6 <- NULL
Horse_Race$running_position_5 <- NULL
Horse_Race$running_position_4 <- NULL
Horse_Race$running_position_3 <- NULL
Horse_Race$running_position_2 <- NULL
Horse_Race$running_position_1 <- NULL
Horse_Race$finish_time <- NULL
```

Ensuite, nous avons séparé notre dataset en deux parties, l'une servant à entrainer notre modèle, l'autre à tester celui-ci. Nous avons choisi d'accorder 70% de notre dataset à la partie entrainement, et le reste pour notre partie test.

Fonction séparant les parties entrainement et test de notre modèle

```
#base train/test (70/30)
DP <- createDataPartition(Horse_Race$finishing_position , p = 0.70 , list = FALSE)
train <- Horse_Race[DP , ]
test <- Horse_Race[-DP , ]
```

Nous avons ensuite vérifié si les données avaient été bien réparties entre les deux parties.

Répartition des positions finales dans le dataset initial

```
         0          1          2          3          4          5          6          7          8
0.02219425 0.07857427 0.07844176 0.07850802 0.07867365 0.07814363 0.07817676 0.07774612 0.07754737
         9         10         11         12         13         14
0.07661985 0.07512919 0.07254538 0.06698026 0.03296012 0.02775937
```

Répartition des positions finales dans la base d'entrainement

```
         0          1          2          3          4          5          6          7          8
0.02100980 0.07949652 0.07982776 0.07736715 0.07836086 0.07651540 0.07878673 0.07907065 0.07651540
         9         10         11         12         13         14
0.07528510 0.07750911 0.07306109 0.06700421 0.03288695 0.02730327
```

Répartition des positions finales dans la base de test

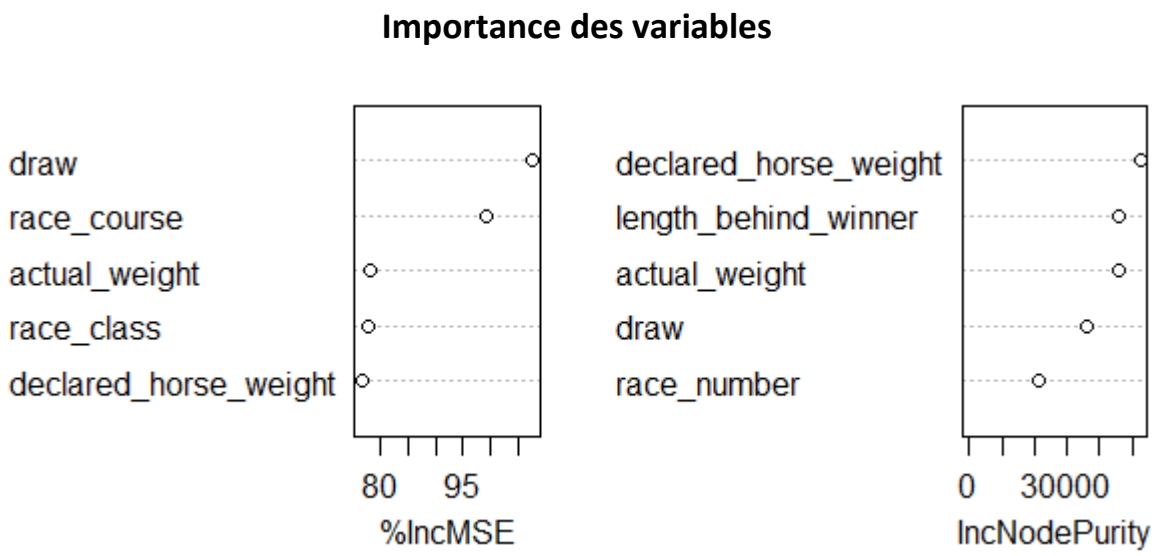
```
         0          1          2          3          4          5          6          7          8
0.02495859 0.07642187 0.07520707 0.08117062 0.07940364 0.08194368 0.07675318 0.07465489 0.07995583
         9         10         11         12         13         14
0.07973495 0.06957482 0.07134180 0.06692435 0.03313087 0.02882385
```

On s'aperçoit que la répartition des données s'est plutôt bien faite, il n'y a pas d'écarts flagrant. Puis nous avons exécuté notre Random Forest à l'aide de la fonction "randomForest()" de la librairie " randomForest". Ci-dessous nous pouvons voir ce que contient notre variable, une fois le Random Forest effectué :

```
Call:
 randomForest(formula = finishing_position ~ ., data = train,      ntree = 1000, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 1000
No. of variables tried at each split: 2

          Mean of squared residuals: 14.88178
                    % Var explained: -2.27
```

Nous pouvons donc tout d'abord voir la formule que nous avons utilisé. Ensuite il y a le type de Random Forest utilisé : classification ou régression. Dans notre cas il s'agit d'une régression car notre variable à expliquer est quantitative (la régression étant accordée pour les variables qualitatives).

Ensuite nous avons fait un plot afin de savoir quelles étaient les variables explicatives les plus importantes dans notre modèle :



Rôles joués par les différentes variables

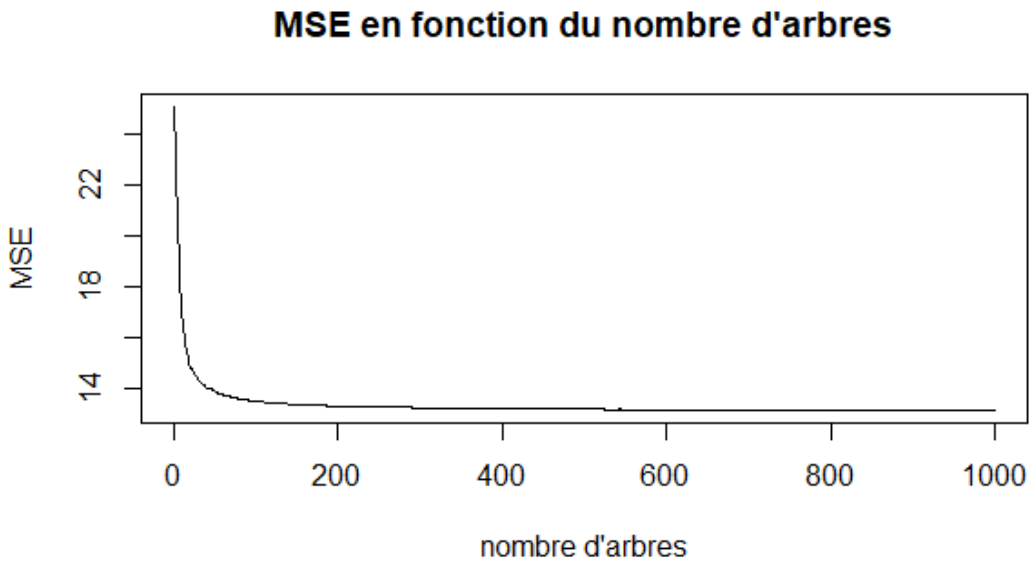
	IncNodePurity	X.IncMSE
draw	35980.626	107.73481
race_course	2294.652	99.20758
actual_weight	45740.857	78.17833
race_class	15151.887	77.72234
declared_horse_weight	52611.115	76.47658
race_distance	13366.171	69.70947
track	17430.117	68.55107
race_number	21466.106	67.99555
track_condition	10653.443	41.44373
length_behind_winner	45854.627	28.30249

On s'aperçoit que la variable qui est de loin la plus importante est le couloir.

Afin d'améliorer notre modèle, nous avons cherché à faire baisser la MSE. Pour cela, nous nous sommes intéressés aux deux principaux paramètres de notre Random Forest : le nombre d'arbres créés et le nombre de variables essayées à chaque split.

Le réglage du nombre d'arbres

Pour régler le nombre d'arbres que nous allions utiliser, nous avons décidé de choisir le nombre d'arbres en gardant la valeur par défaut de "mtry". Nous avons lancé un Random Forest en précisant le nombre d'arbres à 3000. Puis nous avons affiché le taux d'erreur MSE par rapport au nombre d'arbres utilisés pour notre forêt.



On s'aperçoit que la MSE se stabilise aux alentours de 550 arbres. Nous avons donc choisi donc cette valeur.

Le nombre de variables

Maintenant que nous avons notre valeur de nombre d'arbres utilisés, nous nous concentrons sur l'optimisation du nombre de variables essayées à chaque split. Pour cela, il n'y a pas de recette miracle mais juste du réglage à faire. Nous avons donc essayé plusieurs valeurs pour cette variable.

Test du Random Forest pour mtry=1

```
Call:
 randomForest(formula = finishing_position ~ ., data = train,      ntree = 550, mtry = 1
, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 550
No. of variables tried at each split: 1

          Mean of squared residuals: 13.73571
                    % Var explained: 5.55
```

Test du Random Forest pour mtry=3

```
Call:
 randomForest(formula = finishing_position ~ ., data = train,      ntree = 550, mtry = 3
, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 550
No. of variables tried at each split: 3

          Mean of squared residuals: 13.21353
                    % Var explained: 9.14
```

On s'aperçoit ici que le nombre optimum de variables essayées à chaque split est de 3 avec un MSE de 13.21%.

Text Mining

Nous avions une donnée sur les courses qui permettait de faire du Text Mining. En effet, sur chaque course était détaillé les incidents qu'il y a eu sur les chevaux lors de ces courses. Nous nous sommes dit qu'il serait intéressant de faire ressortir les chevaux ayant le plus eu d'incidents.

Nous nous sommes aperçut que les noms des chevaux étaient marqués en majuscules dans le rapport des incidents. Nous avons donc fait un tri afin que seuls les noms de chevaux ressortent de ces textes.

Nous avons décidé d'utiliser l'échantillonnage de Gibbs pour faire notre modèle. Ainsi, le nom des chevaux ayants eus le plus de problème ressort

Top des noms de chevaux ayant eu des problèmes

```
 [1] "plainbluebanner"    "supremeprofit"      "trump"              "kcl"
 [5] "sparklingsword"     "hitthebid"          "luckyday"           "optimism"
 [9] "winningboy"         "gallantrock"        "highlanddragon"     "realfit"
[13] "besttango"          "goodchoice"         "grandharbour"       "happyrocky"
[17] "hitahomerun"        "varapearl"          "pearlwin"           "expedite"
[21] "luckyscepter"       "ââœââ"             "noneother"          "rockthetree"
[25] "styazin"            "winfullpatrol"      "kirov"              "medicswordsman"
[29] "winit"              "archippus"          "regencyking"        "exceloneself"
[33] "speedylongwah"      "ultimateglory"      "uniquejoyful"       "brilliantdream"
[37] "helenââschoice"     "sweetbean"          "dragonbachelor"     "robustmomentum"
[41] "terrificmaster"     "spinningdancer"     "winningleader"      "pikachu"
```

Nous avons aussi pu effectuer un nuage de mots afin de rendre cela plus attractif.

Nuage de mots fait avec les incidents reports

Sources :

Fig. 1.1 & Fig 1.2 tout droit réservé à :
http://slideplayer.fr/slide/1172021/ - Notion de base sur l'ACP –

https://www.xlstat.com/fr/solutions/fonctionnalites/analyse-en-composantes-principales-acp
- Principe de l'AC P –

http://learning.esiea.fr/pluginfile.php/17113/mod_resource/content/1/TD1__Analyse_exploratoire_2017_2018_DTE_DTM.pdf - Tp 1 -

Course de datascience ESIEA : Myriam Maumy-Bertrand & Emmanuelle Claeys

https://fr.wikipedia.org/wiki/Sport_hippique

Annexe

1.1 cercle des corrélations : race distance